Fifth Semester B.E. Degree Examination
CBCS - Model Question Paper - 1
DATABASE MANAGEMENT SYSTEM

Time: 3 hrs. Max. Marks: 80

Note : ***Answer any FIVE full questions, selecting ONE full question from each module.***

MODULE - 1

1. a. **Discuss the main characteristics of the database approach over the file processing approach.** **(8marks)**

Ans. The main characteristics of the database approach versus the file-processing approach are as follows:

- **Self-describing nature of a database system**

A database system includes a complete definition or description of the database's structure and constraints. This description is stored in a system catalog, which contains a description of the structure of each file, the type and storage format of each field and the various constraints on the data. The information stored in catalog is called as meta-data, which describes the structure of the primary database.

In traditional file processing, data definition is part of the application programs. Hence, these programs are constrained to work with only *one specific database*, whose structure is declared in the 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 any changes to the structure of a file may require *changing all programs* that access that file.

DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalog separately from the access programs. This is termed as program-data independence.

DBMS provides a conceptual or logical view of the data to application programs, so that the underlying implementation may be changed without the programs being modified.

- **Support of multiple views of the data**

Different users have different "views" or perspectives on the database. A view is a subset of the database or it contains virtual data that is derived from the database file. A good Multiuser DBMS has facilities for defining multiple views. This is not only convenient for users, but also addresses security issues of data access.

For example, one user of the database is interested only in accessing and printing the transcript of each student. A second user is interested only in checking that students have taken all the prerequisites of each course for which they register.

- **Sharing of data and multiuser transaction processing**

The Multiuser DBMS includes concurrency control software to ensure that several users trying to update the same data in a controlled manner in order to ensure 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 must 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. **Transaction** is a process that makes one or more accesses to a database and which must have the appearance of executing in *isolation* from all other transactions and of being atomic.

b. Explain the typical component modules of a DBMS and their interactions with a neat diagram. (8marks)

Ans. Below Figure illustrate the typical DBMS components.

It is divided into two parts.

The top part refers to the various users of the database environment and their interfaces. The lower part shows the internals of the DBMS responsible for storage of data and processing of transactions.

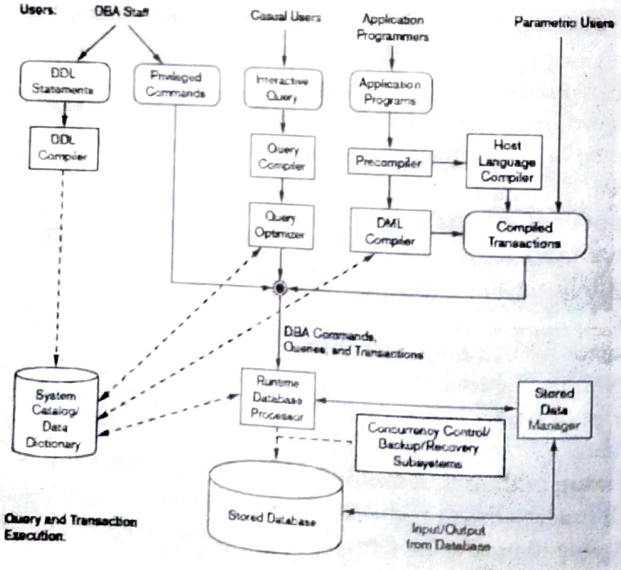
The database and the DBMS catalog are usually stored on disk and **operating system (OS)** controls the accessibility to this disk like read/write.

DBMSs have their own **buffer management** module to schedule disk read/write.

A higher-level **stored data manager** module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.

The DBA staff works on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands.

The DDL compiler processes schema definitions which are specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog. The catalog includes information such as the names and sizes of files, names and data types of data items, storage details of each file, mapping information among schemas, and constraints.



Casual users and persons with occasional need for information from the database interact using some form of interface, which we call the **interactive query** interface.

The **query optimizer** is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution.

The **runtime database processor** executes - the privileged commands, the executable query plans, and the canned transactions with runtime parameters.

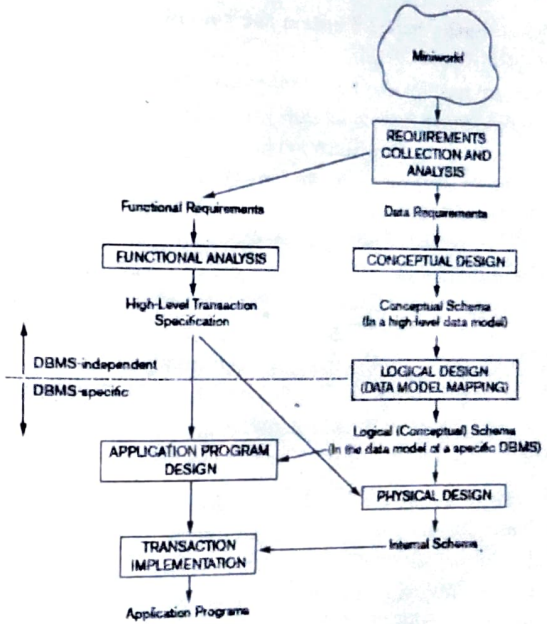
The **precompiler** extracts DML commands from an application program written in a host programming language.

Concurrency control and **backup and recovery systems** are integrated into the working of the runtime database processor for purposes of transaction management.

OR

2. a. With a neat diagram, illustrate the main phases of database design? (8marks)

Ans.



The above figure shows the overview of the database design process. The first step is **requirements collection and analysis**. In this step, the database designers interview database users to understand and document their **data requirements**. The result of this step is a concisely written set of users' requirements.

In parallel with specifying the data requirements, it is useful to specify the known **functional requirements** of the application. It consists of the user-defined **operations** (or **transactions**) that will be applied to the database, including both retrievals and updates.

The next step is to create a **conceptual schema** for the database, using a high-level conceptual data model. This step is called **conceptual design**.
The conceptual schema is a description of the data requirements of the users and includes detailed descriptions of the entity types, relationships, and constraints.
The next step in database design is the actual implementation of the database, using a commercial DBMS. DBMSs use an implementation data model in which the conceptual schema is transformed from the high-level data model into the implementation data model. This step is called **logical design** or **data model mapping.**
The last step is the **physical design** phase, during which the internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files are specified. In parallel with these activities, application programs are designed and implemented as database transactions corresponding to the high-level transaction specifications.

b. What is a weak entity type? Explain the role of partial key in design of weak entity type? **(4marks)**

Ans. An entity type that has no set of attributes that qualify as a key is called **weak.**
Entity of a weak identity type is uniquely identified by the specific entity to which it is related (by a so-called **identifying relationship** that relates the weak entity type with its so-called **identifying** or **owner entity type**) in combination with some set of its own attributes (called a partial key).
Example: A DEPENDENT entity is identified by its first name together with the EMPLOYEE entity to which it is related via DEPENDS_ON.
A weak entity type has a **partial key**, which is the attribute that can uniquely identify weak entities that are related to the same owner entity.
In the above example, 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.

c. What are the responsibilities of DBA and Database Designers? **(4marks)**

Ans. **DBA:** The Data base Administrator (DBA) is the super-user of the system. They oversees and manages the database system (including the data and software).DBA responsibilities include authorizing users to access the database, coordinating/ monitoring its use, acquiring hardware/software for upgrades, etc. In large organizations, the DBA might have a support staff.
Database Designers: Database Designers identifying the data to be stored and choosing an appropriate way to organize the data. They have to communicate with the database users to understand their requirement. They also define **views** for different categories of users. The final design must be able to support the requirements of all the user sub-groups.

Module-2

3. a. Define
i. Domain
ii. Tuple **(6marks)**

Ans. **Domain**: Domain is a set/universe of *atomic* values. "atomic" means that each value in the domain is indivisible (i.e., cannot be broken down into component parts).
Example:
- USA_phone_number: string of digits of length ten

Tuple: A tuple is a mapping from attributes to values drawn from the respective domains of those attributes. A tuple is intended to describe some entity (or relationship between entities) in the miniworld.
Example: A tuple for a PERSON entity might be
{ Name –> "Rumpelstiltskin", Sex –> Male, IQ –> 143 }

b. Discuss the various types of JOIN operations? **(6marks)**

Ans. The **Join** operation denoted by , is used to combine *related tuples* from two relations into single "longer" tuples.
Theta Join: Similar to a CARTESIAN PRODUCT followed by a SELECT. The condition c is called a join condition.
R(A1, A2, ..., Am, B1, B2, ..., Bn) R1(A1, A2, ..., Am) c R2 (B1, B2, ..., Bn)
Equi-Join: The join condition c includes one or more equality comparisons involving attributes from R1 and R2. That is, c is of the form: (Ai=Bj) AND ... AND (Ah=Bk); 1<i,h<m, 1<j,k<n In the above EQUIJOIN operation: Ai, ..., Ah are called the **join attributes** of R1 Bj, ..., Bk are called the **join attributes** of R2
Example of using EQUIJOIN: Retrieve each DEPARTMENT's name and its manager's name:

T <-DEPARTMENT ∞ MGR=SSN EMPLOYEE
RESULT <- Π DNAME,FNAME,LNAME (T)

Natural Join (*): In an EQUIJOIN R R1 c R2, the join attribute of R2 appear redundantly in the result relation R. In a NATURAL JOIN, the redundant join attributes of R2 are eliminated from R. The equality condition is implied and need not be specified. R R1 *(join attributes of R1),(join attributes of R2) R2

c. What is meant by entity integrity constraint? Explain the importance of referential integrity constraint. **(6marks)**

Ans. **Entity integrity constraint**: In a tuple, none of the values of the attributes forming the relation's primary key may have the (non-)value **null**. Or is it that at least one such attribute must have a non-null value.
The **Referential integrity constraint** is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an *existing tuple* in that relation.
For example: The attribute Dno of EMPLOYEE gives the department number for

which each employee works; hence, its value in every EMPLOYEE tuple must match the Dnumber value of some tuple in the DEPARTMENT relation.

A set of attributes FK in relation schema $R1$ is a **foreign key** of $R1$ **that references** relation $R2$ if it satisfies the following rules:

1. The attributes in FK have the same domain(s) as the primary key attributes PK of $R2$; the attributes FK are said to **reference** or **refer to** the relation $R2$.

2. A value of FK in a tuple $t1$ of the current state $r1(R1)$ either occurs as a value of PK for some tuple $t2$ in the current state $r2(R2)$ *or is NULL*. In the former case, we have $t1[FK] = t2[PK]$, and we say that the tuple $t1$ **references** or **refers to** the tuple $t2$.

In this definition, $R1$ is called the **referencing relation** and $R2$ is the **referenced relation**. If these two conditions hold, a **referential integrity constraint** from $R1$ to $R2$ is said to hold

OR

4. a. Describe the six clause in the syntax of an sql retrieval query. Show what type of constructs can be specified in each of the six clauses. Which of the six clauses are required and which are optional. (8marks)

Ans. A retrieval query in SQL consists of six clauses, in which first two clauses i.e SELECT and FROM are mandatory and the remaining four clauses are optional. The clauses are specified in the following order.

```
SELECT <attribute and function list>
FROM <table list>
[WHERE <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING <group condition>]
[ORDER BY <attribute list>]
```

The SELECT clause lists the attributes or functions to be retrieved.

The FROM clause specifies all relations (tables) needed in the query.

The WHERE clause specifies the conditions for selection of tuples from these relations.

Example:

Retrieve the birth date and address of the employee(s) whose name is 'John B Smith'

```
SELECT   b date,address
FROM     EMPLOYEE
WHERE    Fname='John' AND
         Minit='B' AND
         Lname='smith';
```

Group by specifies grouping attributes whereas having specifies a condition on the groups being selected rather than on the individual tuples. The builtin Aggregate functions COUNT,SUM,MIN,MAX AND AVG are used in conjunction with grouping.

Example:

For each project on which more than two employees work, retrieve the project number,project name and the number of employees who work on the project.

```
SELECT    Pnumber,Pname,count(*)
FROM      PROJECT,WORKS_ON
WHERE     Pnumber=Pno
GROUP BY  Pnumber,Pname
HAVING    COUNT(*)>2;
```

ORDER BY specifies an order for displaying the result of a query. Syntax is shown below

```
ORDER BY <column name [<order>]
{, <column name [<order>]}
<order> : =(ASC/DESC)
```

b. Explain Relationship Sets (without Constraints) to Tables? (8marks)

Ans. Relationship set, like an entity set, is mapped to a relation in the relational model. Consider a relationship sets without key and participation constraints.

To represent a relationship, each participating entity is identified and give values to the descriptive attributes of the relationship.

Thus, the attributes of the relation include:

- The primary key attributes of each participating entity set, as foreign key fields.
- The descriptive attributes of the relationship set.

The set of nondescriptive attributes is a superkey for the relation. If there are no key constraints, this set of attributes is a candidate key.

Module-3

5. a. Write a note on: (8marks)

Ans. i. Views in SQL ii. Aggregate functions in SQL

i) Views in SQL : A Views in SQL is a single table that is derived from other tables. These other tables can be *base tables*. A view does not necessarily exist in physical form; it is considered to be a **virtual table.**

In SQL, the command to specify a view is **CREATE VIEW**.

The view is given a (virtual) table name (or view name), a list of attribute names, and a query to specify the contents of the view.

Example: CREATE VIEW WORKS_ON1 **AS SELECT** Fname, Lname, Pname, Hours **FROM** EMPLOYEE, PROJECT, WORKS_ON **WHERE** Ssn=Essn **AND** Pno=Pnumber;

The **DROP VIEW** command to dispose of it.

For example:

V1A: DROP VIEW WORKS_ON1;

ii) Aggregate functions in SQL: 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.

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 query. 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.

Example:

SELECT SUM (Salary), **MAX** (Salary), **MIN** (Salary), **AVG** (Salary) **FROM** EMPLOYEE;

b. How triggers and assertions defined in SQL? Explain **(8marks)**

Ans. In SQL, users can specify general constraints- via **declarative assertions**, using the **CREATE ASSERTION** statement of the DDL.

Each assertion is given a constraint name and is specified via a condition similar to the WHERE clause of an SQL query.

For example, to specify the constraint that the salary of an employee must not be greater than the salary of the manager of the department that the employee works for in SQL.

CREATE ASSERTION SALARY_CONSTRAINT **CHECK (NOT EXISTS (**
SELECT *
FROM EMPLOYEE E, EMPLOYEE M, DEPARTMENT D
WHERE E.Salary>M.Salary **AND** E.Dno=D.Dnumber **AND** D.Mgr_ssn=M.Ssn)
);

The constraint name SALARY_CONSTRAINT is followed by the keyword CHECK, which is followed by a **condition** in parentheses that must hold true on every database state for the assertion to be satisfied.

The constraint name can be used to refer to the constraint or to modify or drop it.

Any WHERE clause condition can be used, but many constraints can be specified using the EXISTS and NOT EXISTS style of SQL conditions.

Another important statement in SQL is **CREATE TRIGGER**. The trigger can be written as below

CREATE TRIGGER SALARY_VIOLATION **BEFORE INSERT OR UPDATE OF** SALARY, SUPERVISOR_SSN **ON** EMPLOYEE **FOR EACH ROW**
WHEN (**NEW**.SALARY > (**SELECT** SALARY **FROM** EMPLOYEE **WHERE** SSN = **NEW**.SUPERVISOR_SSN)) INFORM_SUPERVISOR (**NEW**.Supervisor_ssn, **NEW**.Ssn);

Trigger has three components:

i. The **event(s)**: These are usually database update operations that are explicitly applied to the database. In this example the events are: inserting a new employee record, changing an employee's salary, or changing an employee's supervisor.

ii. The **condition** that determines whether the rule action should be executed: Once the triggering event has occurred, an optional condition may be evaluated. If no condition is specified, the action will be executed once the event occurs.

iii. The **action:** The action is usually a sequence of SQL statements, but it could also be a database transaction or an external program that will be automatically executed

OR

6. a. Briefly explain the Properties of Cursors? **(8marks)**

Ans. The general form of a cursor declaration is:

DECLARE *cursomame* [INSENSITIVE] [SCROLL] CURSOR [WITH HOLD]
FOR *some query* [ORDER BY order-item-list]
[FOR READ ONLY I FOR UPDATE]

- A cursor can be declared to be a read-only cursor (FOR READ ONLY) or, if it is a cursor on a base relation or an updatable view, to be an updatable cursor (FOR UPDATE).
- The UPDATE andDELETE commands allow us to update or delete the row on which the cursor is positioned.

For example, if *sinfa* is an updatable cursor and open, the following statement is executed:

UPDATE Sailors S
SET S.rating = S.rating ~ 1
WHERE CURRENT of sinfo;

- If the keyword SCROLL is specified, the cursor is scrollable, which means that variants of the FETCH command can be used to position the cursor in very flexible ways; otherwise, only the basic FETCH command, which retrieves the next row, is allowed.
- If the keyword INSENSITIVE is specified, the cursor behaves as if it is ranging over a private copy of the collection of answer rows. Otherwise, and by default, other actions of some transaction could modify these rows, creating unpredictable behavior.

For example, while rows is fetched using the *sinfa* cursor, then *rating* values is modified in Sailor rows by concurrently executing the command:

UPDATE Sailors S
SET S.rating = S.rating -

b. Explain Three-Tier Application Architectures? **(8marks)**

Ans. The three-tier architecture separates application logic from data management:

i. Presentation Tier: Users require a natural interface to make requests, provide input, and to see results. The widespread use of the Internet has made Web-based interfaces increasingly popular.

At the presentation layer, the user can issue requests, and display responses that the middle tier generates.

It is important that this layer of code be easy to adapt to different display devices and formats; for example, regular desktops versus handheld devices versus cell phones.

ii. Middle Tier: The application logic executes here. An enterprise-class application reflects complex business processes, and is coded in a general purpose language such as C++ or Java.

The middle layer runs code that implements the business logic of the application: It controls what data needs to be input before an action can be executed, determines the control flow between multi-action steps, controls access to the database layer, and

often assembles dynamically generated HTML pages from database query results. For example, consider the a customer who wants to buy an item (after browsing or searching the site to find it). Before a sale can happen, the customer has to go through a series of steps: She has to add items to her shopping basket, she has to provide her shipping address and credit card number (unless she has an account at the site), and she has to finally confirm the sale with tax and shipping costs added. Controlling the flow among these steps and remembering already executed steps is done at the middle tier of the application.

iii. Data Management Tier: Data-intensive Web applications involve DBMSs.

Module-4

7. a. Explain informal design guidelines for relation schemas. (8marks)

Ans. The four informal measures of quality for relation schema

i. Semantics of relations attributes

Specifies how to interpret the attributes values stored in a tuple of the relation. In other words, how the attribute value in a tuple relate to one another.

Guideline 1: 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.

ii. Reducing redundant values in tuples. Save storage space and avoid update anomalies.

Insertion Anomalies

To insert a new employee tuple into EMP_DEPT, it must include either the attribute values for that department that the employee works for, or nulls. It's difficult to insert a new department that has no employee as yet in the EMP_DEPT relation. The only way to do this is to place null values in the attributes for employee. This causes a problem because SSN is the primary key of EMP_DEPT, and each tuple is supposed to represent an employee entity - not a department entity.

Deletion Anomalies

If an employee tuple is deleted from EMP_DEPT that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database.

Modification Anomalies

In EMP_DEPT, if the value of one of the attributes of a particular department is changed- say the manager of department 5- then all employees in the tuple who work in that department have to be updated.

Guideline 2: Design the base relation schemas so that no insertion, deletion, or modification anomalies occur. Reducing the null values in tuples. e.g., if 10% of employees have offices, it is better to have a separate relation, EMP_OFFICE, rather than an attribute OFFICE_NUMBER in EMPLOYEE.

iii. Reducing the null values in tuples

NULLs can have multiple interpretations, such as the following:

- The attribute does not apply to this tuple. For example, Visa_status may not apply

to U.S. students.

- The attribute value for this tuple is *unknown*. For example, the Date_of_birth may be unknown for an employee.
- The value is known but absent; that is, it has not been recorded yet. For example, the Home_Phone_Number for an employee may exist, but may not be available and recorded yet.

Guideline 3: Avoid placing attributes in a base relation whose values are mostly null.

iv. Generation of Spurious Tuples: Spurious Tuples are the tuples that are not in the original relation but generated by natural join of decomposed subrelations.

Guideline 4: Design relation schemas so that they can be naturally JOINed on primary keys or foreign keys in a way that guarantees no spurious tuples are generated.

b. What is normalization? Explain third normal form with example (8marks)

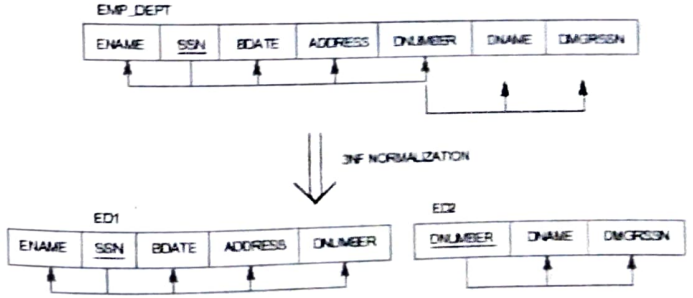
Ans. **Normalization of data** is the process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anomalies

Third normal form is based on the concept of transitive dependency.

A functional dependency X Y in a relation is a transitive dependency if there is a set of attributes Z that is not a subset of any key of the relation, and both X Z and Z Y hold.

In other words, a relation is in 3NF if, whenever a functional dependency X→A holds in the relation, either (a) X is a superkey of the relation, or (b) A is a prime attribute of the relation. Practical Rule: "Eliminate Columns not Dependent on Key," i.e., if attributes do not contribute to a description of a key, remove them to a separate table.

Formal Definition: A relation 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.



1NF: R is in 1NF iff all domain values are atomic.

2NF: R is in 2 NF iff R is in 1NF and every nonkey attribute is fully dependent on the key.

3NF: R is in 3NF iff R is 2NF and every nonkey attribute is non-transitively dependent on the key.

OR

8. a. Write the algorithm for testing non additive join property. (8marks)

Ans. Testing for Nonadditive Join Property

Input: A universal relation R, a decomposition $D = \{R1, R2, ..., Rm\}$ of R, and a set F of functional dependencies.

1. Create an initial matrix S with one row i for each relation Ri in D, and one column j for each attribute Aj in R.

2. Set $S(i, j) := bij$ for all matrix entries. (* each bij is a distinct symbol associated with indices (i, j) *).

3. For each row i representing relation schema Ri {for each column j representing attribute Aj

{if (relation Ri includes attribute Aj) then set $S(i, j) := aj$;};}; (* each aj is a distinct symbol associated with index (j) *).

4. Repeat the following loop until a *complete loop execution* results in no changes to S {for each functional dependency $X \rightarrow Y$ in F {for all rows in S *that have the same symbols* in the columns corresponding to attributes in X {make the symbols in each column that correspond to an attribute in Y be the same in all these rows as follows: If any of the rows has an a symbol for the column, set the other rows to that *same* a symbol in the column. If no a symbol exists for the attribute in any of the rows, choose one of the b symbols that appears in one of the rows for the attribute and set the other rows to that same b symbol in the column ;} ; } ;};

5. If a row is made up entirely of a symbols, then the decomposition has the nonadditive join property; otherwise, it does not.

b. Consider R={ A B C D E F}
Fd's{ AB→C, B→E, A→DF}. Check whether decomposition is lossless. (8marks)

Ans. Key=AB

$R_1 = \{A,B,C\}$

$R_2 = \{A,D,F\}$

$R_3 = \{B,F\}$

Decomposition is lossless if for any 2 relations R_1 and R_2

$R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$

$R_1 \cap R_1 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$

Here this condition is not satisfied so the decomposition is lossy.

Module-5

9. a. Write a short note on: (8marks)
i. Transaction support in SQL. ii. Write ahead log protocol.

Ans. i. Transaction support in SQL

An SQL transaction is a logical unit of work (i.e., a single SQL statement).

- The *access mode* can be specified as *READ ONLY* or *READ WRITE*. The default is *READ WRITE*, which allows update, insert, delete, and create commands to be executed. A mode of READ ONLY, as the name implies, is simply for data retrieval.

The *diagnostic area size* option specifies an integer value n, indicating the number of conditions that can be held simultaneously in the diagnostic area.

The *isolation level* option is specified using the statement *ISOLATION LEVEL*. where the value for <isolation> can be READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, or SERIALIZABLE.

the default isolation level is *SERIALIZABLE*.

If a transaction executes at a lower isolation level than SERIALIZABLE, then the following three violations may occur:

i. Dirty read. ii. Nonrepeatable read. iii. Phantoms.

ii. Write ahead log protocol.

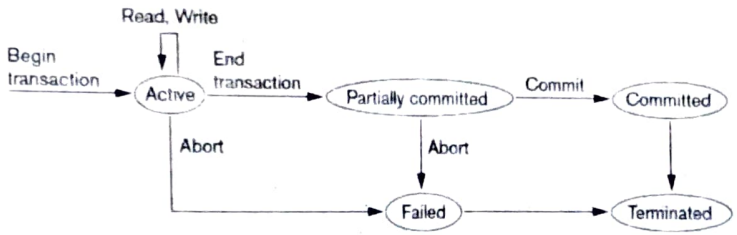
Write-ahead logging (WAL) protocol for a recovery algorithm that requires both UNDO and REDO:

1. The before image of an item cannot be overwritten by its after image in the database on disk until all UNDO-type log records for the updating transaction— up to this point—have been force-written to disk.

2. The commit operation of a transaction cannot be completed until all the REDO-type and UNDO-type log records for that transaction have been force written to disk.

b. With a neat state transition diagram, discuss the different states of a transition. (8marks)

Ans.



- **BEGIN_TRANSACTION.** This marks the beginning of transaction execution.
- **READ or WRITE.** These specify read or write operations on the database items that are executed as part of a transaction.
- **END_TRANSACTION.** This specifies that READ and WRITE transaction operations have ended and marks the end of transaction execution. At this point it may be necessary to check whether the changes introduced by the transaction can be permanently applied to the database (committed) or whether the transaction has to be aborted because it violates serializability or for some other reason.
- **COMMIT_TRANSACTION.** This signals a *successful end* of the transaction so that any changes (updates) executed by the transaction can be safely **committed** to the database and will not be undone.
- **ROLLBACK (or ABORT).** This signals that the transaction has *ended unsuccessfully*, so that any changes or effects that the transaction may have applied to the database must be **undone**.

OR

10. a. Explain Time stamp ordering algorithms? **(8marks)**

Ans. A schedule in which the transactions participate is serializable, and the *only equivalent* serial schedule permitted has the transactions in order of their timestamp values. This is called **Timestamp Ordering (TO)**

Basic TO Algorithm: The concurrency control algorithm must check whether conflicting operations violate the timestamp ordering in the following two cases:

1. Whenever a transaction T issues a write_item(X) operation, the following is checked:

a. If read_TS(X) > TS(T) or if write_TS(X) > TS(T), then abort and roll back T and reject the operation. This should be done because some *younger* transaction with a timestamp greater than TS(T)—and hence *after* T in the timestamp ordering—has already read or written the value of item X before T had a chance to write X, thus violating the timestamp ordering.

b. If the condition in part (a) does not occur, then execute the write_item(X) operation of T and set write_TS(X) to TS(T).

2. Whenever a transaction T issues a read_item(X) operation, the following is checked:

a. If write_TS(X) > TS(T), then abort and roll back T and reject the operation. This should be done because some younger transaction with timestamp greater than TS(T)—and hence *after* T in the timestamp ordering—has already written the value of item X before T had a chance to read X.

b. If write_TS(X) ≤ TS(T), then execute the read_item(X) operation of T and set read_TS(X) to the *larger* of TS(T) and the current read_TS(X).

Strict Timestamp Ordering (TO):

Thomas's Write Rule: A modification of the basic TO algorithm, known as **Thomas's write rule**, does not enforce conflict serializability, but it rejects fewer write operations by modifying the checks for the write_item(X) operation as follows:

1. If read_TS(X) > TS(T), then abort and roll back T and reject the operation.

2. If write_TS(X) > TS(T), then do not execute the write operation but continue processing. This is because some transaction with timestamp greater than TS(T)—and hence after T in the timestamp ordering—has already written the value of X. Thus, we must ignore the write_item(X) operation of T because it is already outdated and obsolete. Notice that any conflict arising from this situation would be detected by case (1).

3. If neither the condition in part (1) nor the condition in part (2) occurs, then execute the write_item(X) operation of T and set write_TS(X) to TS(T).

b. Describe the shadow paging recovery technique. Under what circumstances does it not require a log? **(8marks)**

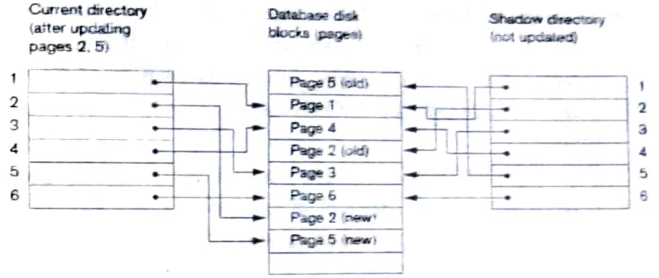
Ans. This recovery scheme does not require the use of a log in a single-user environment. In a multiuser environment, a log may be needed for the concurrency control method. Shadow paging considers the database to be made up of a number of fixedsize disk

pages (or disk blocks)—say, n—for recovery purposes. A **directory** with n entries is constructed, where the ith entry points to the ith database page on disk. The directory is kept in main memory if it is not too large, and all references—reads or writes—to database pages on disk go through it.

When a transaction begins executing, the **current directory**—whose entries point to the most recent or current database pages on disk—is copied into a **shadow directory**. The shadow directory is then saved on disk while the current directory is used by the transaction.

During transaction execution, the shadow directory is *never* modified. When a write_item operation is performed, a new copy of the modified database page is created, but the old copy of that page is *not overwritten*. Instead, the new page is written elsewhere—on some previously unused disk block. The current directory entry is modified to point to the new disk block, whereas the shadow directory is not modified and continues to point to the old unmodified disk block. Below **Fig** illustrates the concepts of shadow and current directories. For pages updated by the transaction, two versions are kept. The old version is referenced by the shadow directory and the new version by the current directory.

An example of shadow paging



In a multiuser environment with concurrent transactions, logs and checkpoints must be incorporated into the shadow paging technique.

Fifth Semester B.E. Degree Examination

CBCS - Model Question Paper - 2

DATABASE MANAGEMENT SYSTEM

Time: 3 hrs. Max. Marks: 80

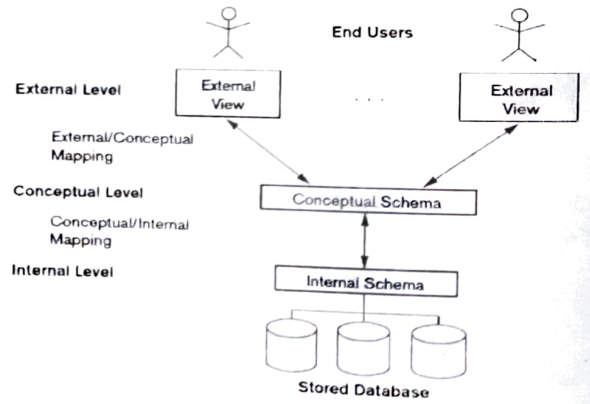
Note : ***Answer any FIVE full questions, selecting ONE full question from each module.***

MODULE - 1

1. a. Explain the three level DBMS architecture, with a neat diagram. Why do we need mappings between schema levels? (08 marks)

Ans. This idea was first described by the ANSI/SPARC committee in late 1970's. The goal is to separate (i.e., insert layers of "insulation" between) user applications and the physical database.

- **Internal level**: has an internal/physical schema that describes the physical storage structure of the database using a low-level data model. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
- **Conceptual level**: has a conceptual schema describing the (logical) structure of the whole database for a community of users. It hides physical storage details, concentrating upon describing entities, data types, relationships, user operations, and constraints. It is described using either high-level or implementation data model.
- **External/view level**: includes a number of external schemas (or user views), each of which describes part of the database that a particular category of users is interested in, hiding rest of database. It is described using either high-level or implementation data model.



The DBMS transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for

processing over the stored database. If the request is database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are called mappings.

b. What are the advantages and the disadvantages of DBMS? Explain? (08 marks)

Ans. Advantages of DBMS:

Controlling Redundancy: Data redundancy (occurs in the "file processing" approach) leads to **wasted storage space**, **duplication of effort** (when multiple copies of a datum need to be updated), and a higher likelihood of the introduction of **inconsistency**.

In the database approach, during database design the views of different user groups are integrated. i.e each logical data details are stored in *only one place* in the database .This is termed as **data normalization**, and it ensures consistency and saves storage space. In order to improve the performance of queries, it is necessary to use **controlled redundancy**

A DBMS should provide the capability to automatically enforce the rule that no inconsistencies are introduced when data is updated.

- **Restricting Unauthorized Access:** A DBMS provides a **security and authorization subsystem**, which is used for specifying restrictions on user accounts. DBMS will enforce these restrictions automatically. Allow read-only access (no updating), or access only to a subset of the data are some kinds of restrictions.
- **Providing Persistent Storage for Program Objects:** Object-oriented database systems make it easier for complex runtime objects (e.g., lists, trees) to be saved in secondary storage so as to survive beyond program termination and to be retrievable at a later time.
- **Providing Storage Structures for Efficient Query Processing:** The DBMS maintains indexes (typically in the form of trees and/or hash tables) that are utilized to improve the execution time of queries and updates. (The choice of which indexes to create and maintain is part of *physical database design and tuning* (see Chapter 16) and is the responsibility of the DBA.
- The **query processing and optimization** module is responsible for choosing an efficient query execution plan for each query submitted to the system.
- **Providing Backup and Recovery:** The subsystem having this responsibility ensures that recovery is possible in the case of a system crash during execution of one or more transactions.

Disadvantages of DBMS:

- A DBMS is a complex piece of software, optimized for certain kinds of workloads and its performance may not be adequate for certain specialized application.
- The abstract view of the data presented by the DBMS does not match the application's needs and actually get in the way.
- In most situations calling for large scale data management, DBMS have become an indispensable tool.

OR

2. a. **Explain the different types of attributes that occur in the ER Model. Write their corresponding notations?** **(08 marks)**

Ans. Attributes are classified as:

- Simple/Atomic vs. Composite

A **composite** attribute is one that is *composed* of smaller parts, which represent basic attributes with independent meanings. Composite attributes forms a hierarchy.
Example: The Address attribute of the EMPLOYEE entity can be subdivided into Street_address, City, State, and Zip.

An **atomic** attribute is indivisible or indecomposable.

- Single-valued vs. Multi-valued (or set-valued)

The attribute that have a single value for a particular entity is called as Single-valued attribute.
Example: Age is a single valued attribute of a person.
The attribute that have a number of values for a particular entity is called as Multi-valued attribute.
Example: The color attribute of the car. The car can have two or three colors at most.

- Stored vs. Derived :

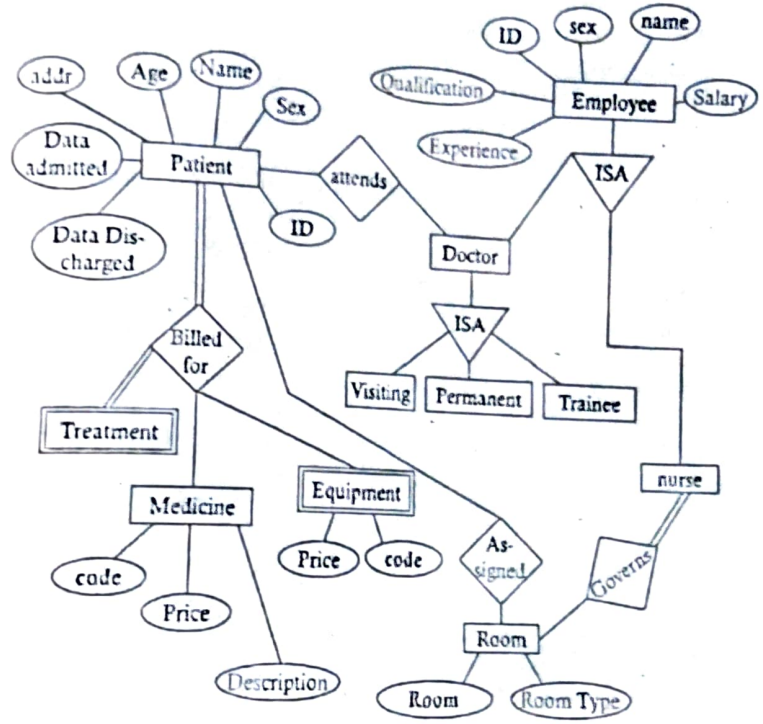
A derived attribute is one whose value can be calculated from the values of other attributes.
Example: Age can be calculated from BirthDate.
A stored attribute is one from which a derived attribute is calculate.
Example: BirthDate is a stored attribute from which age is calculated.

- NULL Values:

A particular entity may not have an applicable value for an attribute.
Example: a College_degrees attribute applies only to people with college degrees.

b. **Write an ER diagram of hospital management system. Assume your own entities (minimum 4), attributes and relations.** **(08 marks)**

Ans.



Module-2

3. a. **Explain the characteristics of relations?** **(08 marks)**

Ans. **Ordering of Tuples**: A relation is a set of tuples; hence, there is no order associated with them.

When a relation is depicted as a table, the tuples are necessarily listed in *some* order, of course, but you should attach no significance to that order.

when tuples are represented on a storage device, they must be organized in *some* fashion, and it may be advantageous, from a performance standpoint, to organize them in a way that depends upon their content.

Ordering of Attributes: A tuple is best viewed as a mapping from its attributes to the corresponding values. Hence, the order in which the attributes are listed in a table is irrelevant.

Values of Attributes: For a relation to be in *First Normal Form*, each of its attribute domains must consist of atomic values.

Much of the theory underlying the relational model was based upon this assumption.

The **Null** value: used for *don't know*, *not applicable*.

Interpretation of a Relation: Each relation can be viewed as a **predicate** and each

tuple in that relation can be viewed as an assertion for which that predicate is satisfied for the combination of values in it. In other words, each tuple represents a fact.
Example: The first tuple listed means: There exists a student having name Benjamin Bayer, having SSN 305-61-2435, having age 19, etc.
The **closed world assumption** states that the only true facts about the miniworld are those represented by whatever tuples currently populate the database.

b. What is constraint? Give the detailed explanation of key constraints. (08 marks)

Ans. Constraints on databases can be categorized as follows:

- **inherent model-based:** Example: no two tuples in a relation can be duplicates (because a relation is a set of tuples)
- **schema-based:** can be expressed using DDL, this kind is the focus of this section.
- **application-based:** are specific to the "business rules" of the miniworld and typically difficult or impossible to express and enforce within the data model. Hence, it is left to application programs to enforce.

Key Constraints: A relation is a *set* of tuples, and each tuple's "identity" is given by the values of its attributes. Hence, it makes no sense for two tuples in a relation to be identical. That is, no two tuples may have the same combination of values in their attributes.
Usually the miniworld dictates that there be subsets of attributes for which no two tuples may have the same combination of values. Such a set of attributes is called a **superkey** of its relation. From the fact that no two tuples can be identical, it follows that the set of all attributes of a relation constitutes a superkey of that relation.
A **key** is a minimal superkey, i.e., a superkey such that, if we were to remove any of its attributes, the resulting set of attributes fails to be a superkey.
Example: The faculty member is uniquely identified by Name and Address and also by Name and Department, but by no single one of the three attributes mentioned. Then { Name, Address, Department } is a (non-minimal) superkey and each of { Name, Address } and { Name, Department } is a key (i.e., minimal superkey).

OR

4. a. Write the SQL Queries for the following database schema (08 marks)
STUDENT (USN, NAME, BRANCH, PERCENTAGE)
FACULTY (FID, FNAME, DEPT, DESIGNATION, SALARY)
COURSE (CID, CNAME, FID)
ENROLL (CID, USN, GRADE)
i. Retrieve the name of all students enrolled for the course 'cs-54'
ii. List all the department having an average salary of the faculties above Rs 10000
iii. List the names of the students enrolled for the course 'cs-54' and having 'B' grade.

Ans. i.Select USN from STUDENT, ENROLL,COURSE where STUDENT. USN=ENROLL.USN and COURSE.CID = ENROLL.CID and COURSE. CNAME='Cs-54'

ii. Select DEPARTMENT from FACULTY where avg(SALARY) > 10000 group by DEPARTMENT
iii. Select NAME from STUDENT,ENROLL.COURSE where STUDENT.USN = ENROLL.USN and ENROLL..CID=COURSE.CID And COURSE.CNAME = 'CS-51' and ENROLL.GRADE = 'B'

b. Explain how the group by clause works. What is the difference between the WHERE and HAVING clause? (08 marks)

Ans. Aggregate function can be applied to subgroups in a relation, where the subgroups are based on some attribute values.
The relation is partitioned into non-overlapping subsets (or groups) of tuples. Each group consists of the tuples that have the same value of some attributes called the grouping attributes.
SQL has a GROUP BY clause for this function. It specifies the grouping attributes, which can also be used with SELECT clause.
Example: For each department, retrieve the department number, the number of employees in the department and their average salary.

```
SELECT    Dno, COUNT (*),AVG (salary)
FROM      EMPLOYEE
GROUP BY  Dno;
```

Difference between WHERE and HAVING clause

WHERE CLAUSE	HAVING CLAUSE
1. Where clause can be used other than select statement also.	1. Having is used only with the select statement.
2. Where clause applies to each and single row.	2. Having clause applies to summarize rows
3. In where clauses, the data is fetched from memory according to condition.	3. In Having clause, the completed data is fetched firstly and then separated according to condition.
4. Where clause is used before GROUP BY clause.	4. Having clause is used to improve condition on GROUP function and is used after GROUP BY clause in the query.

Module-3

5. a. What is a view? Explain how to create the view and how view can be dropped? What problems are associated with updating views? (08 marks)

Ans. A view is a single table that is derived from other tables. The other tables can be *base tables* or previously defined views.
In SQL, the command to specify a view is **CREATE VIEW**. The view is given a (virtual) table name (or view name), a list of attribute names, and a query to specify the contents of the view.
CREATE VIEW DEPT_INFO(Dept_name, No_of_emps, Total_sal)
AS SELECT Dname, **COUNT** (*), **SUM** (Salary)
FROM DEPARTMENT, EMPLOYEE

WHERE Dnumber=Dno
GROUP BY Dname;
The **DROP VIEW** command is used to dispose the view.
For example, to get rid of the view V1, we can use the SQL statement in **V1A:**
V1A: DROP VIEW WORKS_ON1;
Updating of views is complicated and can be ambiguous. An update on a view defined on a *single table* without any *aggregate functions* can be mapped to an update on the underlying base table under certain conditions.
If the view involves joins, an update operation may be mapped to update operations on the underlying base relations in *multiple ways*. Hence, it is often not possible for the DBMS to determine which of the updates is intended.
Example: Consider the WORKS_ON1 view, and suppose that the command to update the PNAME attribute of 'John Smith' is issued from 'ProductX' to 'ProductY'. This view update is shown below:
UPDATEWORKS_ON1
SET Pname = 'ProductY'
WHERE Lname='Smith' **AND** Fname='John'
AND Pname='ProductX';

b. Explain the rules for dealing with NULL values in SQL? (08 marks)

Ans. NULL is used to represent a missing value.
There are three different interpretations
- value *unknown* (exists but is not known)
- value *not available* (exists but is purposely withheld)
- value *not applicable* (the attribute is undefined for this tuple).
- Consider the following examples.

1. Unknown value. A person's date of birth is not known, so it is represented by NULL in the database.
2. Unavailable or withheld value. A person has a home phone but does not want it to be listed, so it is withheld and represented as NULL in the database.
3. Not applicable attribute. An attribute LastCollegeDegree would be NULL for a person who has no college degrees because it does not apply to that person.
SQL uses a three-valued logic with values TRUE, FALSE, and UNKNOWN instead of the standard two-valued (Boolean) logic with values TRUE or FALSE. It is necessary to define the results of three-valued logical expressions when the logical connectives AND, OR, and NOT are used.

OR

6. a. Explain the Stored Procedures? (08 marks)

Stored procedures are beneficial for software engineering. Once a stored procedure is registered with the database server, different users can re-use the stored procedure, eliminating duplication of efforts in writing SQL queries or application logic, and making code maintenance easy.
Creating a Simple Stored Procedure


```
CREATE PROCEDURE AddInventory (
IN book_isbn CHAR(IO),
IN addedQty INTEGER)
UPDATE Books
SET
WHERE
qty_in_stock = qtyjn_stock + addedQty
bookjsbn = isbn
```

Stored procedures can also have parameters. These parameters have one of three different modes: IN, OUT, or INOUT.
IN parameters are arguments to' the stored procedure. OUT parameters are returned from the stored procedure; it assigns values to all OUT parameters that the user can process. INOUT parameters combine the properties of IN and OUT parameters. They contain values to be passed to the stored procedures, and the stored procedure can set their values as return values. Stored procedures enforce strict type conformance: If a parameter is of type INTEGER, it cannot be called with an argument of type VARCHAR.

Calling Stored Procedures from JDBC

Stored procedures from JDBC is called using the CallableStatment class. CallableStatement is a subclass of PreparedStatement and provides the same functionality.
A stored procedure contain multiple SQL statements or a series of SQL statements- thus, the result could be many different ResultSet objects. We illustrate the case when the stored procedure result is a single ResultSet.

```
CallableStatement cstmt=
COIl. prepareCall(" {call ShowNumberOfOrders}");
ResultSet rs = cstmt.executeQueryO
while (rs.next())
```

Calling Stored Procedures from SQLJ

The stored procedure 'ShowNumberOfOrders' is called as follows using SQLJ:

```
/ / create the cursor class
#sql !terator CustomerInfo(int cid, String cname, int count);
/ / create the cursor
CustomerInfo customerinfo;
/ / call the stored procedure
#sql customerinfo = {CALL ShowNumberOfOrders};
while (customerinfo.nextO) {
System.out.println(customerinfo.cid() + "," +
customerinfo.count()) ;
}
```

b. Explain the Common Gateway Interface and Servlets? (08 marks)

Ans. Common Gateway Interface

The Common Gateway Interface connects HTML forms with application programs.

It is a protocol that defines how arguments from forms are passed to programs at the server side. We do not go into the details of the actual CGI protocol since libraries enable application programs to get arguments from the HTML form. Example of CGI script, written in Perl.

```
#!/usr/bin/perl
use CGI;
### part 1
$dataIn = new CGI;
$dataIn-I,headerO;
$authorName = $dataIn-I,param(`authorName`);
### part 2
print (II<HTML><TITLE>Argument passing test</TITLE> II) ;
print (II The user passed the following argument: II) ;
print (II authorName: ", $authorName);
### part 3
print ("</HTML>");
exit;
```

Perl is an interpreted language that is often used for CGI scripting and many Perl libraries, called **modules,** provide high-level interfaces to the CGI protocol.

The CGI module is a convenient collection of functions for creating CGI scripts.

Java servlets

Java servlets are pieces of Java code that run on the middle tier, in either webservers or application servers. There are special conventions on how to read the input from the user request and how to write output generated by the servlet. Servlets are truly platform-independent, and so they have become very popular with Web developers. All servlets must implement the Servlet interface. In most cases, servlets extend the specific HttpServlet class for servers that communicate with clients via HTTP. The HttpServlet class provides methods such as doGet and doPost to receive arguments from HTML forms, and it sends its output back to the client via HTTP. Servlets that communicate through other protocols (such as ftp) need to extend the class GenericServlet.

Servlets usually handle requests from HTML forms and maintain state between the client and the server.

Module-4

7. a. Define and explain the first, second and third normal forms (08 marks)

Ans. First Normal Form (1NF)

First normal form is now considered to be part of the formal definition of a relation. It states that the domains of attributes must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. Practical Rule: "Eliminate Repeating Groups," i.e., make a separate table for each set of related attributes, and give each table a primary key. Formal Definition: A relation is in first normal form (1NF) if and only if all underlying

simple domains contain atomic values only.

Second Normal Form (2NF)

Second normal form is based on the concept of fully functional dependency.

A functional X—> Y is a fully functional dependency is removal of any attribute A from X means that the dependency does not hold any more. A relation schema is in 2NF if every nonprime attribute in relation is fully functionally dependent on the primary key of the relation. It also can be restated as: a relation schema is in 2NF if every nonprime attribute in relation is not partially dependent on any key of the relation.

Practical Rule: "Eliminate Redundant Data," i.e., if an attribute depends on only part of a multi valued key, remove it to a separate table.

Formal Definition: A relation is in second normal form (2NF) if and only if it is in 1NF and every non key attribute is fully dependent on the primary key.

Third Normal Form (3NF) Third normal form is based on the concept of transitive dependency. A functional dependency X—>Y in a relation is a transitive dependency if there is a set of attributes Z that is not a subset of any key of the relation, and both X —>Z and Z —>Y hold. In other words, a relation is in 3NF if, whenever a functional dependency X —> A holds in the relation, either (a) X is a super key of the relation, or (b) A is a prime attribute of the relation. Practical Rule: "Eliminate Columns not Dependent on Key," i.e., if attributes do not contribute to a description of a key, remove them to a separate table.

Formal Definition: A relation is in third normal form (3NF) if and only if it is in 2NF and every non key attribute is non transitively dependent on the primary key.

b. What is a functional dependency? Explain? (08 marks)

Ans. A functional dependency (FD) is a constraint between two sets of attributes from the database.

It is denoted by X —> Y

"*Y* is **functionally dependent** on *X*". X is called the left-hand side of the FD. Y is called the right-hand side of the FD.

A functional dependency is a property of the semantics or meaning of the attributes, i.e., a property of the relation schema. They must hold on all relation states (extensions) of R. Relation extensions r(R).

A FD X—> *Y* is a full functional dependency if removal of any attribute from X means that the dependency does not hold any more; otherwise, it is a partial functional dependency. Examples:

1. SSN ENAME
2. PNUMBER {PNAME, PLOCATION}
3. {SSN, PNUMBER} HOURS

FD is property of the relation schema R, not of a particular relation state/instance

Let R be a relation schema, where X R and Y R t1, t2 r, t1[X] = t2[X] t1[Y] = t2[Y]

The FD X Y holds on R if and only if for all possible relations r(R), whenever two tuples of r agree on the attributes of X, they also agree on the attributes of Y.

- the single arrow denotes "functional dependency"
- X Y can also be read as "X determines Y"
- the double arrow denotes "logical implication"

OR

8. a. Explain the following **(08 marks)**
i) Inclusion dependencies ii) Domain key Normal Form

Ans. i. Inclusion dependencies were defined in order to formalize two types of interrelational constraints:

- The foreign key (or referential integrity) constraint cannot be specified as a functional or multivalued dependency because it relates attributes across relations.
- The constraint between two relations that represent a class/subclass relationship also has no formal definition in terms of the functional,multivalued, and join dependencies.

An **inclusion dependency** $R.X < S.Y$ between two sets of attributes— X of relation schema R, and Y of relation schema S—specifies the constraint that, at any specific time when r is a relation state of R and s a relation state of S

ii. Domain-key normal form (DKNF) is to specify the "ultimate normal form" that takes into account all possible types of dependencies and constraints.

A relation is said to be in **DKNF** if all constraints and dependencies that should hold on the relation can be enforced simply by enforcing the domain constraints and key constraints on the relation.

b. What is the dependency preservation property for decomposition? Why is it important? **(08 marks)**

Ans. It would be useful if each functional dependency $X \rightarrow Y$ specified in F either appeared directly in one of the relation schemas Ri in the decomposition D or could be inferred from the dependencies that appear in some Ri. This is the *dependency preservation condition*.

The dependencies should be preserved because each dependency in F represents a constraint on the database. If one of the dependencies is not represented in some individual relation Ri of the decomposition, This constraints cannot be enforced by dealing with an individual relation. Multiple relations have to be joined so as to include all attributes involved in that dependency.

It is not necessary that the exact dependencies specified in F appear themselves in individual relations of the decomposition D. It is sufficient that the union of the dependencies that hold on the individual relations in D be equivalent to F.

Definition. Given a set of dependencies F on R, the **projection** of F on Ri, denoted by $\pi Ri(F)$ where Ri is a subset of R, is the set of dependencies $X \rightarrow Y$ in $F+$ such that the attributes in $X \cup Y$ are all contained in Ri. Hence, the projection of F on each relation schema Ri in the decomposition D is the set of functional dependencies in $F+$, the closure of F, such that all their left- and right-hand-side attributes are in Ri. Decomposition $D = \{R1, R2, ..., Rm\}$ of R is **dependency-preserving** with respect to F if the union of the projections of F on each Ri in D is equivalent to F; that is,

$((\pi R1(F)) \cup ... \cup (\pi Rm(F)))+ = F+$.

If a decomposition is not dependency-preserving, some dependency is **lost** in the decomposition. To check that a lost dependency holds, two or more relations in the decomposition must be joined to get a relation that includes all left and right-hand-side attributes of the lost dependency, and then check that the dependency holds on the result of the JOIN.

Module-5

9. a. Briefly discuss different type of locks used in concurrency control. **(08 marks)**

Ans. Binary Locks.

A **binary lock** have two **states** or **values:** locked and unlocked (or 1 and 0).

A distinct lock is associated with each database item X. If the value of the lock on X is 1, item X cannot be accessed by a database operation that requests the item. If the value of the lock on X is 0, the item can be accessed when requested, and the lock value is changed to 1.It includes two operations, lock_item and unlock_item.

If the simple binary locking scheme is used, every transaction must obey the following rules:

1. A transaction T must issue the operation lock_item(X) before any read_item(X) or write_item(X) operations are performed in T.

2. A transaction Tmust issue the operation unlock_item(X) after all read_item(X) and write_item(X) operations are completed in T.

3. A transaction T will not issue a lock_item(X) operation if it already holds the lock on item X.1

4. A transaction T will not issue an unlock_item(X) operation unless it already holds the lock on item X.

shared/exclusive:

Shared/exclusive or **read/write** locks have three locking operations: read_lock(X), write_lock(X), and unlock(X).

A lock associated with an item X, LOCK(X).

A **read-locked item** is also called **share-locked** because other transactions are allowed to read the item.

write-locked item is called **exclusive-locked** because a single transaction exclusively holds the lock on the item.

When the shared/exclusive locking scheme is used, the system must enforce the following rules:

1. A transaction T must issue the operation read_lock(X) or write_lock(X) before any read_item(X) operation is performed in T.

2. A transaction T must issue the operation write_lock(X) before any write_item(X) operation is performed in T.

3. A transaction T must issue the operation unlock(X) after all read_item(X) and write_item(X) operations are completed in T.3

4. A transaction T will not issue a read_lock(X) operation if it already holds a read (shared) lock or a write (exclusive) lock on item X. This rule may be relaxed, as we discuss shortly.

5. A transaction T will not issue a write_lock(X) operation if it already holds a read (shared) lock or write (exclusive) lock on item X. This rule may also be relaxed, as we discuss shortly.
6. A transaction T will not issue an unlock(X) operation unless it already holds a read (shared) lock or a write (exclusive) lock on item X.

b. Explain the type of failures? Why recovery is needed? (08 marks)

Ans. Whenever a transaction is submitted to a DBMS for execution, the system is responsible for making sure that either
(1) all the operations in the transaction are completed successfully and their effect is recorded permanently in the database, or
(2) the transaction has no effect whatsoever on the database or on any other transactions.
The DBMS must not permit some operations of a transaction *T* to be applied to the database while other operations of *T* are not. This may happen if a transaction fails after executing some of its operations but before executing all of them.

Types of Failures

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. In addition, 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. Notice that an exception condition, such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal, to be canceled. This exception should be programmed in the transaction itself, and hence would not be considered a failure.

4. Concurrency control enforcement: The concurrency control method may decide to abort the transaction, to be restarted later, because it violates serializability or because several transactions are in a state of deadlock.

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.

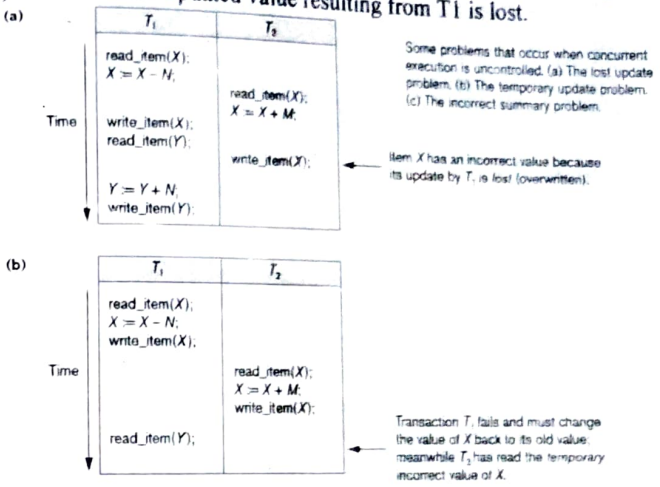
6. Physical problems and catastrophes: This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft, sabotage, overwriting disks or tapes by mistake, and mounting of a wrong tape by the operator.

OR

10. a. Explain the problems that can occur when concurrent transactions are executed. Give example. (12 Marks)

Ans. The Lost Update Problem.

This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect. Suppose that transactions T1 and T2 are submitted at approximately the same time, and suppose that their operations are interleaved then the final value of item *X* is incorrect, because T2 reads the value of *X before* T1 changes it in the database, and hence the updated value resulting from T1 is lost.



The Temporary Update (or Dirty Read) Problem.

This problem occurs when one transaction updates a database item and then the transaction fails for some reason. The updated item is accessed by another transaction before it is changed back to its original value. Above figure (b) shows an example where T1 updates item *X* and then fails before completion, so the system must change *X* back to its original value. Before it can do so, however, transaction T2 reads the "temporary" value of *X*, which will not be recorded permanently in the database because of the failure of T1. The value of item *X* that is read by T2 is called *dirty data,* because it has been created by a transaction that has not completed and committed yet; hence, this problem is also known as the *dirty read problem.*

The Incorrect Summary Problem.

If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated. For example, suppose that a transaction T3 is calculating the total number of reservations on all the flights; meanwhile, transaction T1 is executing. If the interleaving of operations shown in Figure (c) occurs, the result of T3 will be off by an amount *N* because T3 reads the value of *X after N* seats have been subtracted from it but reads the value of *Y before* those *N* seats have been added to it.

(c)

T_1	T_3

	sum := 0;
	read_item(A);
	sum := sum + A;
	⋮

read_item(X);
X := X − N;
write_item(X);	

	read_item(X);
	sum := sum + X;
	read_item(Y);
	sum := sum + Y;

read_item(Y);
Y := Y + N;
write_item(Y);	

T_3 reads X after N is subtracted and reads Y before N is added; a wrong summary is the result (off by N).

Another problem that may occur is called **unrepeatable read,** where a transaction T reads an item twice and the item is changed by another transaction T' between the two reads. Hence, T receives *different values* for its two reads of the same item. This may occur, for example, if during an airline reservation transaction, a customer is inquiring about seat availability on several flights. When the customer decides on a particular flight, the transaction then reads the number of seats on that flight a second time before completing the reservation.

b. Write a note on check pointing. **(04 marks)**

Ans. A check point is a record written into the log periodically at that point when the system writes out to the database on disk all DBMS buffers that have been modified. The recovery manager of DBMS decides at what intervals to take a check point. The interval could be measure in time-say every m minutes or in the number t of committed transactions since the last check point, where the values of m or t are system parameters.

Taking a check point consists of the following actions

1. Suspend execution of transaction temporarily.
2. Force right all main memory buffers that have been modified to disk.
3. Write a [check point] record to the log and force- write the log to disk.
4. Resume executing transactions.

Fifth Semester B.E. Degree Examination
CBCS - Model Question Paper - 3
DATABASE MANAGEMENT SYSTEM

Time: 3 hrs. Max. Marks: 80

Note : ***Answer any FIVE full questions, selecting ONE full question from each module.***

MODULE - 1

1. a. Briefly explain the history of database application? **(08 marks)**

Ans. Early Database Applications: The Hierarchical and Network Models were introduced in mid 1960s and dominated during the seventies.

A bulk of the worldwide database processing still occurs using these models.

These database systems were implemented on large and expensive mainframe computers.

The main drawbacks of early database systems were the intermixing of conceptual relationships with the physical storage and placement of records on disk. And these systems did not provide sufficient data abstraction and program-data independence capabilities.

- **Relational Model based Systems:**

Relational model was originally introduced in 1970, was heavily researched and experimented Relational model with in IBM Research and several universities.

Relational model separated the physical storage of data from its conceptual representation and also provide a mathematical foundation for data representation and querying.

The relational data model also introduced high-level query languages that provided an alternative to programming language interfaces,

- **Object-oriented and emerging applications:**

Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications. Many relational DBMSs have incorporated object database concepts, leading to a new category called *object-relational* DBMSs (ORDBMSs)

Mainly used in specialized applications such as engineering design, multimedia publishing, and manufacturing systems.

- **Extended relational systems add further capabilities** (e.g. for multimedia data, XML, and other data types)

Relational DBMS Products emerged in the 1980s

- Interchanging Data on the Web for E-Commerce Using XML

Data on the Web and E-commerce Applications:

- Web contains data in HTML (Hypertext markup language) with links among pages.
- This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language).
- Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database.
- Extending Database Capabilities for New Applications

New functionality is being added to DBMSs in the following areas:

- Scientific Applications

- XML (eXtensible Markup Language)
- Image Storage and Management
- Audio and Video data management
- Data Warehousing and Data Mining
- Spatial data management
- Time Series and Historical Data Management
- The above gives rise to *new research and development* in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.
- Also allow database updates through Web pages.

b. Define database? Explain the implicit properties of database? **(04 marks)**

Ans. The term **database** refers to any collection of related data. According to Elmasri & Navathe, database is not only a collection of related data, but a database must have the following properties:

- It represents some aspect of the real world, called the **miniworld**. Changes to the miniworld are reflected in the database. For example, a UNIVERSITY miniworld concerned with students, courses, course sections, grades, and course prerequisites.
- It is a logically coherent collection of data, to which some meaning can be attached.
- It has a purpose: there is an intended group of users and some preconceived applications that the users are interested in employing.

c. Explain the different categories of data models **(04 marks)**

Ans. **A data model** is an abstract, self-contained, logical definition of the objects, operators, and so forth, which together constitute the *abstract machine* with which users interact.

i. High-level/conceptual: provides a view close to the way users would perceive data; uses concepts such as

- **entity**: real-world object or concept (e.g., student, employee, course, department, event)
- **attribute**: some property of interest describing an entity (e.g., height, age, color)
- **relationship**: an interaction among entities (e.g., works-on relationship between an employee and a project)

ii. Representational/implementation: It is the intermediate level of abstractness. It provide concepts that can be easily understood by end users but that are not too far removed from the way data is organized in computer storage.

Example is relational data model. Also called as **record-based** model.

iii. Low-level/physical: This level describes how data is stored in computer system, such as record formats, orderings of records, access paths.

An **access path** is a structure that makes the search for particular database records efficient.

OR

2. a. Define **(08 marks)**

Ans. **i. Entity type:** An **entity type** serves as a template for a collection of **entity instances**, all of which are described by the same collection

ii. Entity set: An **entity set** is the collection of all entities of a particular type that exist, in a database, at some moment in time.

iii. Snapshot: The data in the database at a particular time is called the **state** of the database, or a **snapshot**. It is also called the current set of **occurrences** or **instances** in the database.

iv. Participation Constraints: The **participation constraint** specifies whether the existence of an entity depends on its being related to another entity via the relationship type.

b. DESIGN an ER-diagram for the movie database considering the following requirements: **(08 marks)**

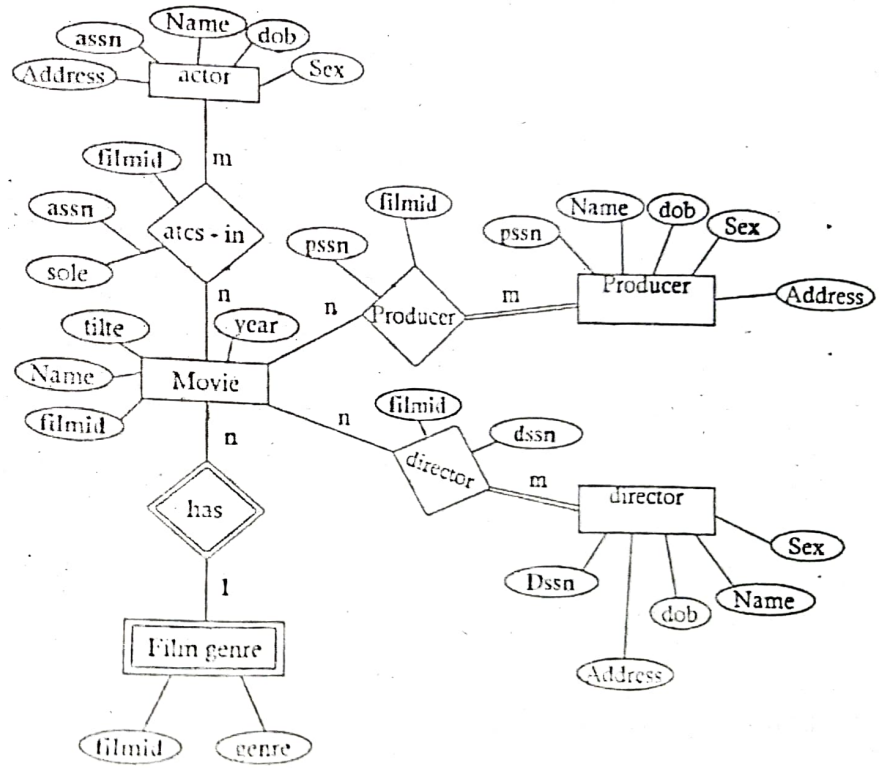
a. Each movie is identified by its title and year of release, it has length in minutes and can have zero of more quotes, languages.

b. Production companies are identified by name, they have address and each production company can produce one or more movies.

c. Actors are identified by name and date of birth, they can act in one or more movies and each actor has a role in a movie

d. Director is identified by Name, Dssn, dob, sex, address.

Ans.



Module-2

3. a. Briefly explain how Attribute Constraints and Attribute Defaults are Specified (08 marks)

Ans. Constraint NOT NULL is specified if NULL is not permitted for a particular attribute. It is implicitly specified for the attributes that are part of the primary key, but it can be specified for any other attributes whose values are required not to be NULL.

Example:

Dnumber INT **NOT NULL CHECK** (Dnumber > 0 **AND** Dnumber < 21);

A default value for an attribute can be specified by appending the clause **DEFAULT** <value> to an attribute definition. The default value is included in any new tuple if an explicit value is not provided for that attribute.

Example: CREATE TABLE EMPLOYEE

(. . . , Dno INT **NOT NULL DEFAULT** 1,.....)

If no default clause is specified, the default "default value" is NULL for attributes that do not have the NOT NULL constraint. Another type of constraint can restrict attribute or domain values using the **CHECK** clause.

The CHECK clause can also be used in conjunction with the CREATE DOMAIN statement.

Example:

CREATE DOMAIN D_NUM **AS** INTEGER **CHECK** (D_NUM > 0 **AND** D_NUM < 21);

b. Explain the SELECT and PROJECT operations in relational algebra with example. (08 marks)

The SELECT operation is used to choose a *subset* of the tuples from a relation that satisfies a **selection condition**. The SELECT operation keeps only those tuples that satisfy a qualifying condition.

In general, the SELECT operation is denoted by

σ<selection condition>(R)

where the symbol σ (sigma) is used to denote the SELECT operator and the selection condition is a Boolean expression (condition) specified on the attributes of relation R.

Example:

σ(Dno=4 **AND** Salary>25000) **OR** (Dno=5 **AND** Salary>30000)(EMPLOYEE)

The **PROJECT** operation selects certain *columns* from the table and discards the other columns.

The general form of the PROJECT operation is

π<attribute list>(R)

where Π (pi) is the symbol used to represent the PROJECT operation, and <attribute list> is the desired sublist of attributes from the attributes of relation R.

Example:

πSex, Salary(EMPLOYEE)

OR

4. a. Explain with example in SQL
i. Insert command ii. Delete command iii. Update command (08 marks)

Ans. Insert command

The **Insert** operation provides a list of attribute values for a new tuple t that is to be inserted into a relation R.

Example:

Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.

Delete command

The DELETE command removes tuples from a relation. It includes a WHERE clause, to select the tuples to be deleted. Tuples are explicitly deleted from only one table at a time.

Example:

DELETE FROM EMPLOYEE

WHERE Lname='Brown';

Update command

The **UPDATE** command is used to modify attribute values of one or more selected tuples. A WHERE clause in the UPDATE command selects the tuples to be modified from a single relation. An additional **SET** clause in the UPDATE command specifies the attributes to be modified and their new values.

For example:

UPDATE PROJECT

SET Plocation = 'Bellaire', Dnum = 5

WHERE Pnumber=10;

b. Explain how Relationship Sets is translated with Key Constraints (08 marks)

If a relationship set involves 'n' entity sets and some 'm' of them are linked via arrows in the ER diagram, the key for anyone of these 'm' entity sets constitutes a key for the relation to which the relationship set is mapped. Hence in 'm' candidate keys, one can be designated as the primary key.

Consider the relationship set in which the table corresponding to Manages has the attributes *ssn, did, since.* However, because each department has at most one manager, no two tuples can have the same *did* value but differ on the *ssn* value. The *did* is itself a key for Manages.The set *did, ssn* is not a key.

The Manages relation can be defined using the following SQL statement:

CREATE TABLE Manages (ssn CHAR (11),
did INTEGER,
since DATE,
PRIMARY KEY (did),
FOREIGN KEY (ssn) REFERENCES Employees,
FOREIGN KEY (did) REFERENCES Departments

A second approach to translating a relationship set with key constraints is often superior because it avoids creating a distinct table for the relationship set. The idea is to include the information about the relationship set in the table corresponding to the entity set with the key, taking advantage of the key constraint.

In the Manages example, because a department has at most one manager, A key fields can be added to the Employees tuple denoting the manager and the *since* attribute to the Departments tuple. This approach eliminates the need for a separate Manages relation, and queries.

The only drawback to this approach is that space could be wasted if several departments have no managers. In this case the added fields would have to be filled with *null* values.

Module-3

5. a. Expalin Attribute Data Types for SQL? **(08 marks)**

Ans. **Numeric** data types include integer numbers of various sizes (INTEGER or INT, and SMALLINT) and floating-point (real) numbers of various precision (FLOAT or REAL, and DOUBLE PRECISION). Formatted numbers can be declared by using DECIMAL(*i,j*)—or DEC(*i,j*) or NUMERIC(*i,j*)—where *i*, the *precision*, is the total number of decimal digits and *j*, the *scale*, is the number of digits after the decimal point.

Character-string data types are either fixed length-CHAR (*n*) or CHARACTER (*n*), where *n* is the number of characters-or varying length-VARCHAR (*n*) or CHAR VARYING (*n*) or CHARACTER VARYING (*n*), where *n* is the maximum number of characters.

Bit-string data types are either of fixed length *n*—BIT(*n*)—or varying length—BIT VARYING(*n*), where *n* is the maximum number of bits.

Boolean data type has the traditional values of TRUE or FALSE. In SQL, because of the presence of NULL values, a three-valued logic is used, a third possible value for a Boolean data type is UNKNOWN.

The **DATE** data type has ten positions, and its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD. The TIME data type has at least eight positions, with the components HOUR, MINUTE, and SECOND in the form HH:MM: SS.

A **timestamp** data type (TIMESTAMP) includes the DATE and TIME fields, plus a minimum of six positions for decimal fractions of seconds and an optional WITH TIME ZONE qualifier.

b. Explain EXISTS and GROUP BY Functions in SQL **(08 marks)**

Ans. The EXISTS function in SQL is used to check whether the result of a correlated nested query is *empty* (contains no tuples) or not. The result of EXISTS is a Boolean value **TRUE** if the nested query result contains at least one tuple, or **FALSE** if the nested query result contains no tuples.

Example:

```
SELECT E.Fname, E.Lname
FROM EMPLOYEE AS E
WHERE EXISTS ( SELECT *
FROM DEPENDENT AS D
WHERE E.Ssn=D.Essn AND E.Sex=D.Sex
AND E.Fname=D.Dependent_name):
```

GROUP BY : The relation is partitioned into nonoverlapping subsets (or **groups**) of tuples. Each group (partition) will consist of the tuples that have the same value of some attribute(s), called the **grouping attribute(s)**. Group by function can be applied to each such group independently to produce summary information about each group.

The GROUP BY clause specifies the grouping attributes, which should *also appear in the SELECT clause*, so that the value resulting from applying each aggregate function to a group of tuples appears along with the value of the grouping attribute(s).

Example: For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
SELECT Dno, COUNT (*), AVG (Salary)
FROM EMPLOYEE
GROUP BY Dno;
```


OR

6. a. Explain the Classification of drivers in JDBC? **(08 marks)**

Ans. Drivers in JDBC are cla.ssified into four types depending on the architectural relationship between the application and the data source:

- **Type I Bridges:** This type of driver translates JDBC function calls into function calls of another API that is not native to the DBMS.

An example is a JDBC-ODBC bridge.

An application can use JDBC calls to access an ODBC compliant data source. The application loads only one driver, the bridge.

Bridges have the advantage that it is easy to piggyback the application onto an existing installation, and no new drivers have to be installed. One drawback is increased number of layers between data source and application affects performance.

- **Type II Direct Translation to the Native API via Non-Java Driver:** This type of driver translates JDBC function calls directly into method invocations of the API of one specific data source.

The driver is written using a combination of C++ and Java. It is dynamically linked and specific to the data source. This architecture performs significantly better than a JDBC-ODBC bridge.

One disadvantage is that the database driver that implements the API needs to be installed on each computer that runs the application.

- **Type III Network Bridges:** The driver talks over a network to a middleware server that translates the JDBC requests into DBMS-specific method invocations.

In this case, the driver on the client site is not DBMS-specific.

The JDBC driver loaded by the application can be quite small, as the only functionality it needs to implement is sending of SQL statements to the middleware server.

The middleware server can then use a Type II JDBC driver to connect to the data source.

- **Type IV-Direct Translation to the Native API via Java Driver:** Instead of calling the DBMS API directly, the driver communicates with the DBMS through Java sockets.

In this case, the driver on the client side is written in Java, but it is DBMS-specific.

It translates JDBC calls into the native API of the database system. This solution does not require an intermediate layer, and since the implementation is all Java, its performance is usually quite good.

b. Briefly explain the advantages of the Three-Tier Architecture **(08 marks)**

Ans. The three-tier architecture has the following advantages:

- **Heterogeneous Systems:** Applications can utilize the strengths of different platforms and different software components at the different tiers. It is easy to modify or replace the code at any tier without affecting the other tiers.
- **Thin Clients:** Clients only need enough computation power for the presentation layer. Clients are Web browsers.
- **Integrated Data Access:** In many applications, the data must be accessed from several sources. This can be handled transparently at the middle tier, where we can centrally manage connections to all database systems involved.

- **Scalability to Many Clients:** Each client is lightweight and all access to the system is through the middle tier. The middle tier can share database connections across clients, and if the middle tier becomes the bottle-neck, we can deploy several servers executing the middle tier code. Clients can connect to anyone of these servers, if the logic is designed appropriately.
- **Software Development Benefits:** By dividing the application cleanly into parts that address presentation, data access, and business logic, we gain many advantages. The business logic is centralized, and is therefore easy to maintain, debug, and change. Interaction between tiers occurs through well-defined, standardized APIs. Therefore, each application tier can be built out of reusable components that can be individually developed, debugged, and tested.

Module-4

7. a.

Book_title	Auth_name	Book_type	Listprice	Affiliation	Publication

FD'S {Book_title->Book_type, Publication
Auth_name->Affiliation
Book_type->Listprice}
What normal form is the relation in? explain your answer. Apply normalization until you cannot decompose the relations further. State the reasons behind each decomposition.? (08 marks)

Ans.

- The relation is in 1NF and not in 2NF as no attributes are fully functionally dependent on the key (BookTitle and AuthorName). It is also not in 3NF.
- The relation is not in 2NF because:

  BookTitle →Publisher, BookType
  BookType →ListPrice
  AuthorName →AuthorAffiliation
- Thus, these attributes are not fully functionally dependent on the primary key. The 2NF decomposition will eliminate the partial dependencies.
- 2NF decomposition:

  Book1(BookTitle, AuthorName)
  Book2(BookTitle, BookType, ListPrice, Publisher)
  Book3(AuthorName, AuthorAffiliation)
- The relations are not in 3NF because:

BookTitle → BookType → ListPrice

- Thus, BookType is neither a key itself nor a subset of a key and ListPrice is not a prime attribute.
- The 3NF decomposition will eliminate the transitive dependency of Listprice.
- 3NF decomposition:

  Book1(BookTitle, AuthorName)
  Book2A(BookTitle, BookType, Publisher)
  Book2B(BookType, ListPrice)
  Book3(AuthorName, AuthorAffiliation)

b. Explain the Join dependency and fifth Normal Form? **(08 marks)**

Ans. A **join dependency (JD)**, denoted by JD{R1, R2, ..., 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 lossless join decomposition into R1, R2, ..., Rn; that is, for every such r we have * (ΠR1(r), (ΠR2(r) , ... (ΠRn(r)) = r
Lossless-join property refers to when we decompose a relation into two relations - we can rejoin the resulting relations to produce the original relation. However, sometimes there is the requirement to decompose a relation into more than two relations.
A relation schema R is in **fifth normal form (5NF)** (or **project-join normal form (PJNF)**) with respect to a set F of functional, multivalued, and join dependencies if, for every nontrivial join dependency JD(R1, R2, ..., Rn) in F+ (that is, implied by F),18 every Ri is a superkey of R.

OR

8. a. Explain the Inclusion Dependencies? **(08 marks)**

Ans. Inclusion dependencies were defined in order to formalize two types of interrelational constraints:

- The foreign key (or referential integrity) constraint cannot be specified as a functional or multivalued dependency because it relates attributes across relations.
- The constraint between two relations that represent a class/subclass relationship also has no formal definition in terms of the functional, multivalued, and join dependencies.

Definition. An **inclusion dependency** $R.X < S.Y$ between two sets of attributes- X of relation schema R, and Y of relation schema S—specifies the constraint that, at any specific time when r is a relation state of R and s a relation state of S, we must have $\pi X(r(R)) \leq \pi Y(s(S))$

b. Explain fourth normal form and Multivalued Dependency. (08 marks)

Ans. A relation schema R is in 4NF with respect to a set of dependencies F (that includes functional dependencies and multivalued dependencies) if, for every nontrivial multivalued dependency X →→ Y in F+17 X is a superkey for R.

- An all-key relation is always in BCNF since it has no FDs.
- An all-key relation such as the EMP relation, which has no FDs but has the MVD Ename→→ Pname | Dname, is not in 4NF.
- A relation that is not in 4NF due to a nontrivial MVD must be decomposed to convert it into a set of relations in 4NF.
- The decomposition removes the redundancy caused by the MVD.

A multivalued dependency $X \rightarrow\rightarrow Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples $t1$ and $t2$ exist in r such that $t1[X] = t2[X]$, then two tuples $t3$ and $t4$ should also exist in r with the following properties,15 where we use Z to denote $(R - (X \cup Y))$:

- $t3[X] = t4[X] = t1[X] = t2[X]$.
- $t3[Y] = t1[Y]$ and $t4[Y] = t2[Y]$.
- $t3[Z] = t2[Z]$ and $t4[Z] = t1[Z]$.

Module-5

9. a. Explain why a transaction execution should be atomic? Explain ACID propertie (08 marks)

Ans. Transactions should posses the following (ACID) properties: Transactions should possess several properties. These are often called the **ACID properties,** and they should be enforced by the concurrency control and recovery methods of the DBMS. The following are the ACID properties:

1. **Atomicity:** A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
2. **Consistency preservation:** A transaction is consistency preserving if its complete execution take(s) the database from one consistent state to another.
3. **Isolation:** A transaction should appear as though it is being executed in isolation from other transactions. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.
4. **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.

The atomicity property requires that we execute a transaction to completion. It is the responsibility of the transaction recovery subsystem of a DBMS to ensure atomicity. If a transaction fails to complete for some reason, such as a system crash in the midst of transaction execution, the recovery technique must undo any effects of the transaction on the database.

b. What is schedule? Explain the algorithm which is used to test a schedule for conflict serializability. (08 marks)

Ans. A **schedule** (or **history**) S of n transactions $T1$, $T2$, ..., Tn is an ordering of the operations of the transactions. Operations from different transactions can be interleaved in the schedule S.

Two operations in a schedule are said to **conflict** if they satisfy all three of the following conditions: (1) they belong to *different transactions*; (2) they access the *same item* X; and (3) *at least one* of the operations is a write_item(X).

Testing Conflict Serializability of a Schedule S

1. For each transaction Ti participating in schedule S, create a node labeled Ti in the precedence graph.

2. For each case in S where Tj executes a read_item(X) after Ti executes a write_item(X), create an edge (Ti-> Tj) in the precedence graph.

3. For each case in S where Tj executes a write_item(X) after Ti executes a read_item(X), create an edge (Ti->Tj) in the precedence graph.

4. For each case in S where Tj executes a write_item(X) after Ti executes a write_item(X), create an edge (Ti-> Tj) in the precedence graph.

5. The schedule S is serializable if and only if the precedence graph has no cycles.

OR

10. a. Discuss the problems of deadlock and starvation, and the different approaches to dealing with these problems. (08 marks)

Ans. **Deadlock** occurs when *each* transaction T in a set of *two or more transactions* is waiting for some item that is locked by some other transaction T_ in the set

One way to prevent deadlock is to use a **deadlock prevention protocol**.

One deadlock prevention protocol, which is used in conservative two-phase locking, requires that every transaction lock all the items it needs in if any of the items cannot be obtained, none of the items are locked. Rather, the transaction waits and then tries again to lock all the items it needs. A second protocol, which also limits concurrency, involves *ordering all the items* in the database and making sure that a transaction that needs several items will lock them according to that order. This requires that the programmer is aware of the chosen order of the items, which is also not practical in the database context.

Another group of protocols that prevent deadlock do not require timestamps. These include the no waiting (NW) and cautious waiting (CW) algorithms.

In the **no waiting algorithm**, if a transaction is unable to obtain a lock, it is immediately aborted and then restarted after a certain time delay without checking whether a deadlock will actually occur or not.

The **cautious waiting** algorithm was proposed to try to reduce the number of needless aborts/restarts. Suppose that transaction Ti tries to lock an item X but is not able to do so because X is locked by some other transaction Tj with a conflicting lock.

Another problem that may occur when we use locking is **starvation**, which occurs when a transaction cannot proceed for an indefinite period of time while other transactions in the system continue normally.

One solution for starvation is to have a fair waiting scheme, such as using a **first-come-first-served** queue; transactions are enabled to lock an item in the order in which they originally requested the lock. Another scheme allows some transactions to have priority over others but increases the priority of a transaction the longer it waits, until it eventually gets the highest priority and proceeds.

b . Explain two multiversion techniques for concurrency control. (08 marks)

Ans. i. Multiversion Technique Based on Timestamp Ordering

In this method, several versions $X1$, $X2$... Xk of each data item X are maintained.

For *each version*, the value of version Xi and the following two timestamps are kept:

1. read_TS(Xi). The **read timestamp** of Xi is the largest of all the timestamps of transactions that have successfully read version Xi.

2. write_TS(Xi). The **write timestamp** of Xi is the timestamp of the transaction that wrote the value of version Xi.

Whenever a transaction T is allowed to execute a write_item(X) operation, a new version Xk+1 of item X is created, with both the write_TS(Xk+1) and the read_TS(Xk+1) set to TS(T). Correspondingly, when a transaction T is allowed to read the value of version Xi, the value of read_TS(Xi) is set to the larger of the current read_TS(Xi) and TS(T).

ii. Multiversion Two-Phase Locking Using Certify Locks

In this multiple-mode locking scheme, there are *three locking modes* for an item: read, write, and *certify*.

The state of LOCK(X) for an item X can be one of read-locked, writelocked, certify-locked, or unlocked.

In the standard locking scheme, once a transaction obtains a write lock on an item, no other transactions can access that item.

The idea behind multiversion 2PL is to allow other transactions $T\_$ to read an item X while a single transaction T holds a write lock on X.

Other transactions can continue to read the *committed version* of X while T holds the write lock. Transaction T can write the value of $X\_$ as needed, without affecting the value of the committed version X. However, once T is ready to commit, it must obtain a **certify lock** on all items that it currently holds write locks on before it can commit.

The certify lock is not compatible with read locks, so the transaction may have to delay its commit until all its write-locked items are released by any reading transactions in order to obtain the certify locks.

Fifth Semester B.E. Degree Examination, CBCS - Dec 2017 / Jan 2018

Database Management Systems

Time: 3 hrs. **Max. Marks:** 80

Note : *Answer any FIVE full questions, selecting ONE full question from each module.*

Module - 1

1. a. Explain the main characteristics of the database approach versus the file processing approach.

Ans. Refer Q. No. 1. a., Model Question Paper - 1 **(08 marks)**

b. Explain the three scheme architecture with neat diagram. Why do we need mappings among schema levels? How do different scheme definition languages support this architecture?

Ans. Refer Q. No. 1. a., Model Question Paper - 2 **(08 marks)**

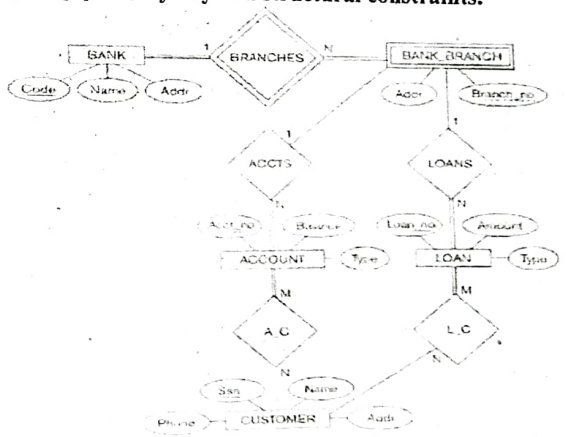
- **Data Definition Language (DDL)** is used by the DBA and by database designers to define both internal and conceptual level schemas.
- **Storage Definition Language (SDL)** is used to specify the internal schema.
- The **View Definition Language (VDL)** is used to specify user views and their mappings to the conceptual schema.
- **Data Manipulation Language (DML)** is used to manipulate the data in the database.

OR

2. a. Discuss with example, different types of attributes **(7marks)**

Ans. Refer Q. No. 2. a., Model Question Paper - 2

b. Draw an ER diagram for a BANK database schema with atleast five entity types. Also specify primary key and structural constraints. **(09 marks)**



Primary key: Code, Branch_no, Loan_no, Acc_no, Ssn.

Module-2

3. a. Describe the characteristics of relations with suitable examples for each. (08 marks)

Ans. Refer Q. No. 3. a., Model Question Paper - 2

b. What are the basic operations that can change the states of relations in the database? Explain how the basic operations deal with constraint violations. (08 marks)

Ans. The basic operations that can change the states of relations in the database are given below:

Refer Q. No. 4. a., Model Question Paper - 3

The basic operations deal with constraint violations are

- Domain constraint is violated if a given attribute value does not appear in the corresponding domain.
- Key constraint is violated if the key value already exists.
- Entity integrity constraint is violated if a primary key value is declared as NULL.
- Referential integrity constraint is violated if foreign key value refers to a tuple that does not exist.
- Delete operation violates only referential integrity constraint.
- Update operation violates all four constraint.

OR

4. a. Describe the steps of an algorithm for ER- to - relational mapping. (10 marks)

Ans. Step 1: Mapping of Regular Entity Types:

For each regular (strong) entity type *E* in the ER schema, create a relation *R* that includes all the simple attributes of *E*. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of *E* as the primary key for *R*. If the chosen key of *E* is a composite, then the set of simple attributes that form it will together form the primary key of *R*.

Step 2: Mapping of Weak Entity Types:

For each weak entity type *W* in the ER schema with owner entity type *E*, create a relation *R* and include all simple attributes of *W* as attributes of *R*.

The primary key of *R* is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type *W*, if any.

Step 3: Mapping of Binary 1:1 Relationship Types.

For each binary 1:1 relationship type *R* in the ER schema, identify the relations *S* and *T* that correspond to the entity types participating in *R*.

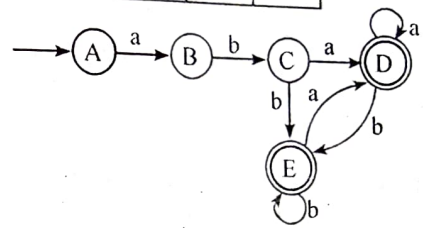
There are three possible approaches: (1) the foreign key approach, (2) the merged relationship approach, and (3) the cross reference or relationship relation approach.

Foreign key approach: Choose one of the relations—*S* and include as a foreign key in *S* the primary key of *T*. It is better to choose an entity type with *total participation* in *R* in the role of *S*. Include all the simple attributes of the 1:1 relationship type *R* as attributes of *S*.

Since no new state, will stop

Since no new state, will stop

δ	a	b
A	B	ϕ
B	ϕ	C
*C	D	E
*D	D	E
*E	D	E

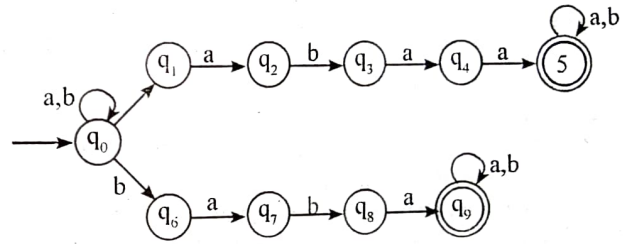


OR

2. a. Draw a DFSM to accept the language,

1 = {ω∈ {a, b}* : ∀x, y∈ {a, b}* ((ω = x abbaay) v (ω = x babay))} (03 Marks)

Ans.



b. Define distinguishable and indistinguishable states. Minimize the following DFSM,

S	0	1
A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

```
( . . . ,Dno INT NOT NULL DEFAULT1,
CONSTRAINT EMPPK
PRIMARY KEY (Ssn),
CONSTRAINT EMPSUPERFK
FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE (Ssn)
ON DELETE SET NULL ON UPDATE CASCADE,
CONSTRAINT EMPDEPTFK
FOREIGN KEY (Dno) REFERENCES DEPARTMENT (Dnumber)
ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

- A constraint NOT NULL is specified if NULL is not permitted for a particular attribute.
- The default value is included in any new tuple if an explicit value is not provided for that attribute.
- The **PRIMARY KEY** clause specifies one or more attributes that make up the primary key of a relation.
- ON DELETE SET NULL and ON UPDATE CASCADE for the foreign key Super_ssn of EMPLOYEE is set. This means that if the tuple for a supervising employee is deleted, the value of Super_ssn is automatically set to NULL for all employee tuples that were referencing the deleted employee tuple.

Module-3

5. a. Consider the COMPANY DAT ABASE
EMPLOYEE (Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, super-ssn, Dno)
DEPARTMENT (Dname, Dnumber, Mgr_ssn, Mgr_st_date)
DEPART_LOCATIONS(Dnumber, Dlocation)
PROJECT (Pname, Pnumber, Plocation, Dnum)
WORKS_ON (Essn, Pno, Hours)
DEPENDENT (Essn, Dependent_name, Sex, Bdate, Relationship).
Specify the following queries in SQL on the database schema given above :

a. For every project located in Stafford, list the project number the controlling department number and the department manager's last name, address and birth date. (04 Marks)

b. List the names of all employees who have a dependent with the same first name as themselves. (02 Marks)

c. For each project, list the project name and the total hours per week (by all employees) spent on that project. (04 Marks)

d. Retrieve the name of each employee who works on all the projects controlled by "Research' department. (06 Marks)

Ans.
```
Select Pnumber, Dnum, Lname, Address, Bdate
from PROJECT, DEPARTMENT, EMPLOYEE
where Dnum= Dnumber and Mgr_Ssn = Ssn and Plocation='stafford';
```



```
b. Select Fname, Lname
from EMPLOYEE, DEPENDENT
where Ssn= Essn and Fname= dependent_name;
c. Select Pname,Sum(Hours)
from PROJECT, WORK_ON
where Pnumber= Pno group by Pname;
d. Select Fname, Lname
from EMPLOYEE
where NOT EXISTS (( select Pnumber
from PROJECT, DEPARTMENT
where Dname= 'Research' and Dnumber= Dnum)
MINUS   (select Pno from WORK_ON
 Where Ssn=Essn));
```

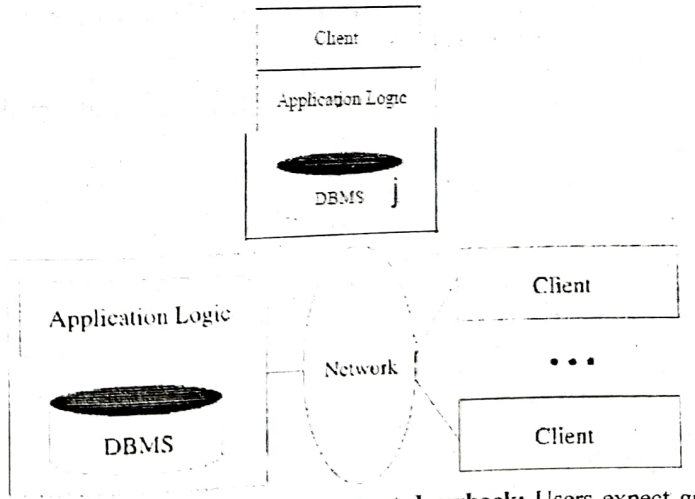
OR

6. a. Define Stored Procedure. Explain the creating and calling of stored procedure with suitable example. (08 Marks)

Ans. Refer Q. No. 6. a., Model Question Paper -2

b. Explain the Single - tier and Client - server architecture, with neat diagram. (08 Marks)

Ans. A data-intensive application is combined into a single tier, including the DBMS, application logic, and user interface, as illustrated in below Figure. The application typically ran on a mainframe, and users accessed it through *dumb terminals* that could perform only data input and display. This approach has the benefit of being easily maintained by a central administrator.



Single-tier architectures have an important drawback: Users expect graphical

interfaces that require much more computational power than simple dumb terminals. Centralized computation of the graphical displays of such interfaces requires much more computational power than a single server available, and thus single-tier architectures do not scale to thousands of users.

Two-tier architectures, often also referred to a client-server architectures, consist of a client computer and a server computer, which interact through a well-defined protocol.

In the traditional client server architecture, the client implements just the graphical user interface, and the server. Implements both the business logic and the data management; such clients are often called thin clients, and this architecture is illustrated above Figure.

Compared to the single-tier architecture, two-tier architectures physically separate the user interface from the data management layer.

Module-4

7. a. Explain informal design guidelines used as measures to determine the quality of relation schema design. (08 marks)

Ans. Refer Q. No. 7. a., Model Question Paper - 1

b. Define Normal forms. Explain 1NF, 2NF and 3NF with suitable examples for each. (08 marks)

Ans. Refer Q. No. 7. a., Model Question Paper - 2

OR

8. a. Write the algorithm for testing non additive join property. (08 marks)

Ans. A minimal cover of a set of functional dependencies E is a minimal set of dependencies that is equivalent to E

Algorithm: Finding a Minimal Cover F for a Set of Functional Dependencies E

Input: A set of functional dependencies E.

1. Set $F := E$.

2. Replace each functional dependency $X \rightarrow \{A1, A2, ..., An\}$ in F by the n functional dependencies $X \rightarrow A1, X \rightarrow A2, ..., X \rightarrow An$.

3. For each functional dependency $X \rightarrow A$ in F

for each attribute B that is an element of X

if $\{ \{F - \{X \rightarrow A\}\} \rightarrow \{(X - \{B\}) \rightarrow A\}\}$ is equivalent to F

then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in F.

4. For each remaining functional dependency $X \rightarrow A$ in F

if $\{F - \{X \rightarrow A\}\}$ is equivalent to F,

then remove $X \rightarrow A$ from F.

Sol: E : {B→A, D→A, AB→D}

BB → AB (IR2) AB→D

AB → D

B → D (IR3)

Minimal Cover : {B→D, D→A}

b. Consider the universal relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies (08 Marks)

F = {{A, B} → {C}, {A}→{D, E}, {B}→{F}, {F}→{G, H}, {D}→{I, J}}. Determine whether each decomposition has the lossless join property with respect to F. D, = {R_1,R_2,R_3} ; R_1 = {A,B,C,D,E} ; R_2={B,F,G, H} ; R_3={D, I, J}.

Ans. Given :

R= { A,B,C,D,E,F,G,H,I,J}

R1= { A,B,C,D,E}

R2= { B,,F,G,H}

R3= {D,I,J}

Functional Dependencies

{ A,B} → {C}{A} → {D,E}

{B} → {F}{F} → {G,H}{D} → {I,J}

R	A	B	C	D	E	F	G	H	I	J
R1	a_1	a_2	a_3	a_4	a_5	b_{16}	b_{17}	b_{18}	b_{19}	a_{10}
R2	b_{21}	a_2	b_{23}	b_{24}	b_{25}	a_6	a_7	a_8	b_{29}	b_{20}
R3	b_{31}	b_{32}	b_{33}	a_4	b_{35}	b_{36}	b_{37}	b_{38}	a_9	a_{10}

According to Functional Dependencies

R	A	B	C	D	E	F	G	H	I	J
R1	a_1	a_2	a_3	a_4	a_5	b_{16}	b_{17}	b_{18}	b_{19}	a_{10}
	a_6	a_7	a_8	a_9						
R2	b_{21}	a_2	b_{23}	b_{24}	b_{25}	a_6	a_7	a_8	b_{29}	b_{20}
R3	b_{31}	b_{32}	b_{33}	a_4	b_{35}	b_{36}	b_{37}	b_{38}	a_9	a_{10}

It is lossless join because first row contains all values of 'a'.

Module-5

9. a. Why Concurrency control is needed demonstrate with example? (12 Marks)

Ans. Refer Q. No. 10. a., Model Question Paper - 2

b. Discuss the desirable properties of transactions. (04 Marks)

Ans. Refer Q. No. 9. a., Model Question Paper - 3

OR

10. a. When deadlock and starvation problems occurs? Explain how these problems can be resolved. (09 Marks)

Ans. Refer Q. No. 10. a., Model Question Paper - 3

b. Explain how shadow paging helps to recover from transaction failure. (07 Marks)

Ans. Refer Q. No. 10. a., Model Question Paper - 1

Fifth Semester B.E. Degree Examination, CBCS - June/July 2018

Database Management Systems

Time: 3 hrs. **Max. Marks:** 80

Note : *Answer any FIVE full questions, selecting ONE full question from each module.*

Module - 1

1. a. Discuss the main characteristics of the database approach and how it differs from traditional file systems. **(04 marks)**

Ans. Refer Q. No. 1. a., Model Question Paper - 1

b. Describe the three-schema architecture. Why do we need mappings among schema levels. **(4 marks)**

Ans. Refer Q. No. 1. a., Model Question Paper – 2

c. Discus various components of a DBMS, with a neat diagram. **(8 marks)**

Ans. Refer Q. No. 1. b., Model Question Paper – 1

OR

2. a. Define an Entity and Attribute. Explain the different types of attributes that occur in an ER - diagram model, with an example. **(6 marks)**

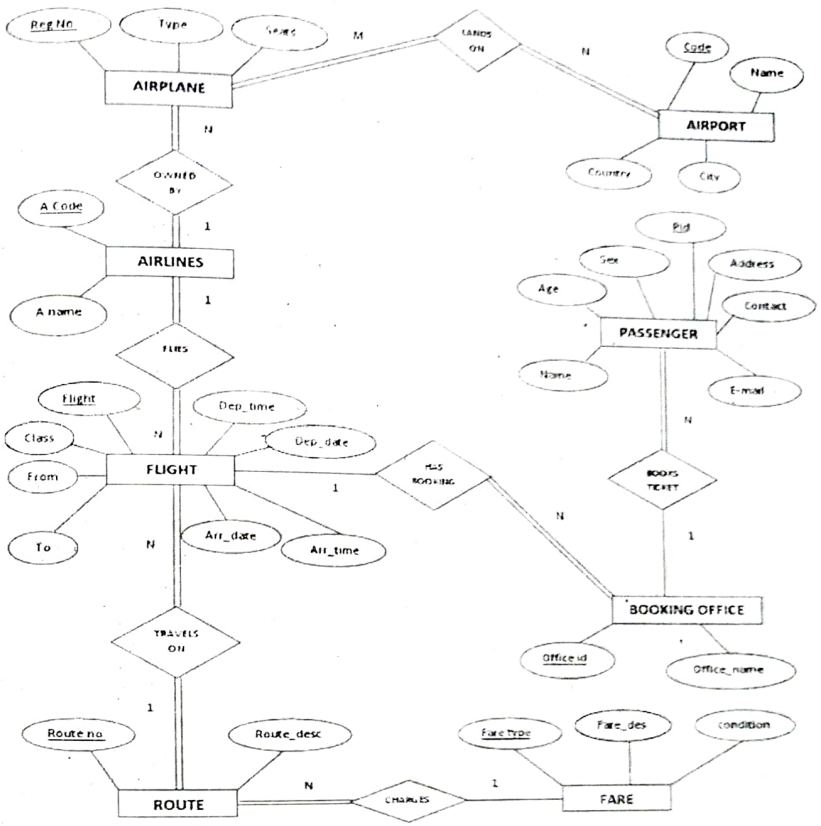
Ans. Refer Q. No. 2. a., Model Question Paper – 2

Entity: It is a thing or object in the real world with an independent existence.
Example: Employee.

Attribute: The property that describes the entity.
Example: Ei, Ename, Eaddress of Employee.

b. Draw an ER - diagram of an Airline reservation system, taking into account at least five entities. Indicate all keys, constraints and assumptions that are made. **(10 marks)**

Ans.



Primary key: A code, code, Reg No, Pid, flight, office id, Route no, fare type.

Module-2

3. a. Explain the data types available for attribute specification in SQL. **(04 marks)**

Ans. Refer Q. No. 5. a., Model Question Paper - 3

b. Explain briefly violations in entity integrity constraint, key and referential integrity constraints, with example. **(06 marks)**

Ans. The basic operations that can change the states of relations in the database are given below:

Refer Q. No. 3. b., Model Question Paper - 2

- Key constraint is violated if the key value already exists.
- Entity integrity constraint is violated if a primary key value is declared as NULL.
- Referential integrity constraint is violated if foreign key value refers to a tuple that does not exist.
- Delete operation violates only referential integrity constraint.
- Update operation violates all four constraint.

c. Consider the following RESORT database,
RESORT (resortno, resortname, resorttype, resortaddr, resortcity, numsuite)
SUITE (suiteno, resortno, suiteprice)
RESERVATION (reservationno, resortno, visitorno, checkin, checkout, totalvisitor, suiteno)
VISITOR (visitorno, firstname, lastname, visitoraddr)
i) Wite the SQL to list full details of all the resorts on Los Angeles
ii) Wite the SQL to list full details of all the resorts having number of suites more than 30.
iii) Wite the SQL to list visitors in ascending order by firstname. **(06 Marks)**

Ans. i. Select * from resort where resortname = 'Los Angeles';
ii. Select * from resort where numsuite > 30;
iii. Select * from visitor order by firstname;

OR

4. a. Explain how constraints are specified in SQL during table creation, with suitable example. **(04 marks)**

Ans. Refer Q. No. 4. b., Dec- 2017

b. Consider the following relations for a database that keeps track of student enrolments in courses and the books adopted for each course. **(06 marks)**
STUDENT (SSn, Name, Major, bdate)
COURSE (Courseno, Cname, dept)
ENROLL (SSn, Courseno, Quarter, grade)
BOOK–ADOPTION (Courseno, Quarter, book_isbn)
TEST (book_isbn, book_title, Publisher, Author)
Write the following queries in realtional algebra on the database schema :
i) List the number of courses taken by all students named John Smith in winter 2009 (i.e., Quarter = W09).
ii) Produce a list of text books (include courseno, book_isbn, book_title) for courses offered by the 'CS' department that have used more than two books.
iii) List any deprartment that has all its adopted books publised by 'Pearson' publishing

Ans. i. Π courseno (σ quarter = w09 ((σ name = ' john smith' (student) ? enroll))
ii. cs_adoption. Π Course#, book_isbn (dept = 'cs'(course) book_adoption) book_ count, course no σ count book_isbn(cs_adoption) course_needed course# (count (book_isbn) > 2), book_ count))
iii. dept_pubs. Π dept,publisher (course, book_adoption, text) result Π dept(course) Π dept (publisher 'pearson' (dept_pub)

c. Give an example of mapping of generalization or specialization into relation schemas. **(06 marks)**

Ans. Options for Mapping Specialization or Generalization.
Convert each specialization with m subclasses {S1, S2,...,Sm} and (generalized)

superclass C, where the attributes of C are {k,a1,...an} and k is the (primary) key, into relation schemas using one of the following options:

■ **Option A:** Multiple relations—superclass and subclasses. Create a relation L for C with attributes Attrs(L) = {k, a1,..., an} and PK(L) = k. Create a relation Li for each subclass Si, $1 \le i \le m$, with the attributes Attrs(Li) = {k} ∪ {attributes of Si} and PK(Li) = k. This option works for any specialization (total or partial, disjoint or overlapping).

■ **Option B:** Multiple relations—subclass relations only. Create a relation Li for each subclass Si, $1 \le i \le m$, with the attributes Attrs(Li) = {attributes of Si} ∪ {k, a1, ..., an} and PK(Li) = k. This option only works for a specialization whose subclasses are total (every entity in the superclass must belong to (at least) one of the subclasses). Additionally, it is only recommended if the specialization has the disjointedness constraint. If the specialization is overlapping, the same entity may be duplicated in several relations.

■ **Option C:** Single relation with one type attribute. Create a single relation L with attributes Attrs(L) = {k, a1, ..., an} ∪ {attributes of S1} ∪ ... ∪ {attributes of Sm} ∪ {t} and PK(L) = k. The attribute t is called a type (or discriminating) attribute whose value indicates the subclass to which each tuple belongs, if any. This option works only for a specialization whose subclasses
are disjoint, and has the potential for generating many NULL values if many specific attributes exist in the subclasses.

■ **Option D:** Single relation with multiple type attributes. Create a single relation schema L with attributes Attrs(L) = {k, a1, ..., an} ∪ {attributes of S1} ∪ ... ∪ {attributes of Sm} ∪ {t1, t2, ..., tm} and PK(L) = k. Each ti, $1 \le i \le m$, is a Boolean type attribute indicating whether a tuple belongs to subclass Si.

Module-3

5. a. Discuss how each of the following constructs is used in SQL and discuss the various options for each construct : **(06 Marks)**
i) Nested Queries **ii) Aggregate functions** **iii) Triggers**
iv) Views and their updatability **v) Schma change statements**
vi) Group by and having clause.

Ans. **i. Nested Queries:**

- **A nested query is** a query in which the existing values in the database is fetched first and then used in a comparison condition.
- It is a complete select-from-where blocks within the WHERE clause of another query. That other query is called the **outer query**.
- The comparison operator **IN**, which compares a value *v* with a set (or multiset) of values *V* and evaluates to **TRUE** if *v* is one of the elements in *V*.

Example:
SELECT DISTINCT Essn
FROM WORKS_ON
WHERE (Pno, Hours) **IN** (**SELECT** Pno, Hours

FROM WORKS_ON
WHERE Essn='123456789');

- This query will select the Essns of all employees who work the same (project, hours) combination on some project that employee 'John Smith' (whose Ssn ='123456789') works on. The IN operator compares the subtuple of values in parentheses (Pno, Hours) within each tuple in WORKS_ON with the set of type-compatible tuples produced by the nested query

ii. Aggregate function:

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.

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 query. 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.

Example:

SELECT SUM (Salary), **MAX** (Salary), **MIN** (Salary), **AVG** (Salary) **FROM** EMPLOYEE;

iii. Triggers:

Trigger has three components:

i. The **event(s)**: These are usually database update operations that are explicitly applied to the database. In this example the events are: inserting a new employee record, changing an employee's salary, or changing an employee's supervisor.

ii. The **condition** that determines whether the rule action should be executed: Once the triggering event has occurred, an optional condition may be evaluated. If no condition is specified, the action will be executed once the event occurs.

iii. The **action:** The action is usually a sequence of SQL statements, but it could also be a database transaction or an external program that will be automatically executed

The trigger can be written as below

For Example:

CREATE TRIGGER SALARY_VIOLATION **BEFORE INSERT OR UPDATE OF** SALARY, SUPERVISOR_SSN **ON** EMPLOYEE **FOR EACH ROW WHEN** (**NEW**.SALARY > (**SELECT** SALARY **FROM** EMPLOYEE **WHERE** SSN = **NEW**.SUPERVISOR_SSN)) INFORM_SUPERVISOR (**NEW**.Supervisor_ssn, **NEW**.Ssn);

iv. Views and their updatability:

A **Views in SQL** is a single table that is derived from other tables. These other tables can be *base tables*. A view does not necessarily exist in physical form; it is considered to be a **virtual table.**

In SQL, the command to specify a view is **CREATE VIEW**.

The view is given a (virtual) table name (or view name), a list of attribute names, and a query to specify the contents of the view.

Example: CREATE VIEW WORKS_ON1 **AS SELECT** Fname, Lname, Pname, Hours **FROM** EMPLOYEE, PROJECT, WORKS_ON **WHERE** Ssn=Essn **AND** Pno=Pnumber;

The **DROP VIEW** command to dispose of it.

For example:

V1A: DROP VIEW WORKS_ON1;

v. Scheme change statement:

The DROP Command

The DROP command can be used to drop named schema elements, such as tables, domains, or constraints. One can also drop a schema.

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

The ALTER Command

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, the command.

ALTER TABLE COMPANY.EMPLOYEE ADD COLUMN Job VARCHAR(12);

vi. Group by and having clause:

Group by specifies grouping attributes whereas having specifies a condition on the groups being selected rather than on the individual tuples. The built in Aggregate functions COUNT, SUM, MIN, MAX AND AVG are used in conjunction with grouping.

b. Draw and explain 3 - tier architecture adn technology relevant to each tier. Write the advantages of 3 - tier architecture. (06 marks)

Ans. Refer Q. No. 6. b., Model Question Paper -1

c. What is CGI? Why was CGI introduced? What are the disadvantages of an architecture using CGI scripts? (04 Marks)

Ans. Refer Q. No. 6. b., Model Question Paper -2

OR

6. a. What is Dynamic SQL and how is it different form Embedded SQL? (08 marks)

Ans.

- Application must accept commands from a user and, based on what the user needs, generate appropriate SQL statements to retrieve the necessary data.
- SQL provides some facilities to deal with such situations; these are referred to as Dynamic SQL .
- The two main commands, PREPARE and EXECUTE, through a simple example: char c_sqlstring[] = {"DELETE FROM Sailors WHERE rating>5"};

EXEC SQL PREPARE readytogo FROM :csqlstring;
EXEC SQL EXECUTE readytogo;

- The first statement declares the C variable *c_sqlstring* and initializes its value to the string representation of an SQL command.
- The second statement results in this string being parsed and compiled as an SQL command, with the resulting executable bound to the SQL variable *readytogo*.
- The third statement executes the command.
- The preparation of a Dynamic SQL command occurs at run-time and is run-time.
- The preparation of a Dynamic SQL command occurs at run-time and is run-time.
- Using Dynamic SQL, parameters can be passed from the host language program to the SQL statement.

b. What is SQL, J and how is it different from JDBC?

Ans.

- SQLJ - 'SQL-Java' was developed by the SQLJ Group, a group of database vendors and Sun.
- SQLJ was developed to complement the dynamic way of creating queries in JDBC with a static model. It is therefore very close to Embedded SQL.
- SQLJ statement binds host language variables title, price, and author to the return values of the cursor books.

```
#sql books = {
SELECT title, price INTO :title, :price
FROM Books WHERE author = :author
};
```

- Comparing the JDBC and SQLJ code, the SQLJ code is much easier to read than the JDBC code. Thus, SQLJ reduces software development and maintenance costs.

c. Consider the following company database:
EMP(name, Ssn, salary, superssn, dno)
DEPT (dnum, dname, mgrssn)
DEPT_LOC(dnum, dlocation)
PROJECT (Pname, Pnumber, Plocation, dnum)
WORKS_ON(Essn, dept_name, sex)
Write SQL queries for the following:
i) Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees.
ii) Retrieve the names of employees who make atlest 10,000 more than the employee who is paid the least in the company.
iii) A view that has the employee name, supervisor name and employee salary for each employee who works in the 'Research' department.
iv) A view that has the project name, controlling department name, number of employees and total hours worked per week on the project for each project with more than one employee working on it. **(08 Marks)**

Ans. i. select name from emp where dno = (select dno from emp where sal= (select

max(sal) from emp));
ii. select name from emp where sal >=10000 + (select min(sal) from emp);
iii. create view vname as
select ename as name , e1.ename as super_name, e.salary from emp e, emp e1, dept d where e.super.ssn = e1.ssn and d.dnumber – e.dno and d.dname= ' research';l
iv. create view proj_det as
select p.pname, d.dname, count(w.essn), sum(w.hrs) from project p, dept d, work_on w where p.num=d.number and p.number=w.pno group by p.pname having count(w.essn) > 1;

Module-4

7. a. Discuss insertion, delection and modification anomalies. Why are they considered bad? Illustrate with examples. **(04 Marks)**

Ans. **Insertion Anomalies:** Insertion anomalies can be differentiated into two types, illustrated by the following examples based on the EMP_DEPT relation:

■ To insert a new employee tuple into EMP_DEPT, either the attribute values are included for the department that the employee works for, or NULLs (if the employee does not work for a department as yet). For example, to insert a new tuple for an employee who works in department number 5, we must enter all the attribute values of department 5 correctly so that they are consistent with the corresponding values for department 5 in other tuples in EMP_DEPT.

■ It is difficult to insert a new department that has no employees as yet in the EMP_DEPT relation. The only way to do this is to place NULL values in the attributes for employee.

Deletion Anomalies: The problem of deletion anomalies is related to the second insertion anomaly situation. If we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database.

Modification Anomalies: In EMP_DEPT, if the value of one of the attributes of a particular department is changed—i.e , the manager of department 5 - the tuples of all employees who work in that department must be updated; otherwise, the database will become inconsistent. If we fail to update some tuples, the same department will be shown to have two different values for manager in different employee tuples, which would be wrong.

b. Define Multivated dependency. Explain fourth normal form, with an example. **(06 Marks)**

Ans. A multivalued dependency X Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t1 and t2 exist in r such that t1[X] = t2[X], then two tuples t3 and t4 should also exist in r with the following properties, where Z denotes (R – (X ∪ Y)).

- A relation schema R is in 4NF with respect to a set of dependencies F (that includes functional dependencies and multivalued dependencies) if, for every nontrivial multivalued dependency X →→ Y in F+17 X is a superkey for R.

The following points are stated:

- An all-key relation is always in BCNF since it has no FDs.
- An all-key relation such as the EMP relation, which has no FDs but has the MVD Ename→→ Pname | Dname, is not in 4NF.
- A relation that is not in 4NF due to a nontrivial MVD must be decomposed to convert it into a set of relations in 4NF.
- The decomposition removes the redundancy caused by the MVD.

c. Consider the universal relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F = {{A, B} → {C}, {A} → {D, E}, {B} → {F}, {F} → {G, H}, {D} → {I, J}.
What is key of R? Decompose R into 2NF and then 3NF relations. (06 Marks)

Ans. Key of {A.B}

2 NF: AB → C
A → DE & D → I J
B → F & F → G H

3 NF: AB → C
A → DE
D → I J
B → F
F → G H

OR

8. a. Define Non-additive join property of a decomposition and write an algorithm of testing for non-additive join property. (04 Marks)

Ans. A decomposition D = {R1, R2, ..., Rm} of R has the lossless (nonadditive) join property with respect to the set of dependencies F on R if, for every relation state r of R that satisfies F, the following holds, where * is the NATURAL JOIN of all the relations in

D: *(πR1(r), ..., πRm(r)) = r.

Testing for Nonadditive Join Property

Input: A universal relation R, a decomposition D = {R1, R2, ..., Rm} of R, and a set F of functional dependencies.

1. Create an initial matrix S with one row i for each relation Ri in D, and one column j for each attribute Aj in R.
2. Set S(i, j):= bij for all matrix entries. (* each bij is a distinct symbol associated with indices (i, j) *).
3. For each row i representing relation schema Ri
{for each column j representing attribute Aj
{if (relation Ri includes attribute Aj) then set S(i, j):= aj;};}; (* each aj is a distinct symbol associated with index (j) *).
4. Repeat the following loop until a complete loop execution results in no changes to S
{for each functional dependency X→Y in F
{for all rows in S that have the same symbols in the columns corresponding to attributes in X

{make the symbols in each column that correspond to an attribute in Y be the same in all these rows as follows: If any of the rows has an a symbol for the column, set the other rows to that same a symbol in the column.
If no a symbol exists for the attribute in any of the rows, choose one of the b symbols that appears in one of the rows for the attribute and set the other rows to that same b symbol in the column ;} ; } ;};
5. If a row is made up entirely of a symbol, then the decomposition has the non-additive join property; otherwise, it does not.

b. A relation R(A, C, D, E, H) satisfies the following FDs : A → C, AC → D, E → AD, E → H. Find the Canonical cover for this set of FD's. (06 marks)

Ans. Given :

R= {A, C, D, E, H}
Functional Dependencies
A → {C} AC → D
E → AD E → H
Redundant check: E → H
Essential check: A → C
E → AD
AC → D
Minimal set of FD's:
A → C, E → D, E → A, E → H.

c. Consider two set of functional dependencies :
F = {A → C, AC → D, E → AD, E → H} and G = {A → CD, E → AH}
Are they equivalent? (06 Marks)

Ans. When F covers G & G covers F, then they are equivalent, apply inference rules.

F covers G
A → C A → CD
AC → D A → D
C → D
E → AD E → A
E → D E → AH
E → H

Module-5

9. a. Discuss ACID propertis of a database transaction. (04 marks)

Ans. Refer Q. No. 9. a., Model Question Paper - 3

b. Explain transaction support in SQL. (06 marks)

Ans.

- SQL transaction is a logical unit of work and is guaranteed to be atomic. Transaction initiation is done implicitly when particular SQL statements are encountered.
- Every transaction must have an explicit end statement, which is either a COMMIT

or a ROLLBACK. These characteristics are specified by a SET TRANSACTION statement in SQL. The characteristics are the *access mode*, the *diagnostic area size*, and the *isolation level*.

- The **access mode** can be specified as READ ONLY or READ WRITE. The default is READ WRITE, unless the isolation level of READ UNCOMMITTED is specified in which case READ ONLY is assumed. A mode of READ WRITE allows select, update, insert, delete, and create commands to be executed. A mode of READ ONLY, is simply for data retrieval.
- The **diagnostic area size** option, DIAGNOSTIC SIZE *n*, specifies an integer value *n*, which indicates the number of conditions that can be held simultaneously in the diagnostic area. These conditions supply feedback information to the user or program on the *n* most recently executed SQL statement.
- The **isolation level** option is specified using the statement ISOLATION LEVEL <isolation>, where the value for <isolation> can be READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, or SERIALIZABLE.15 The default isolation level is SERIALIZABLE, although some systems use READ COMMITTED as their default. The use of the term SERIALIZABLE here is based on not allowing violations that cause dirty read, unrepeatable read, and phantoms

- A sample SQL transaction might look like the following:

```
EXEC SQL WHENEVER SQLERROR GOTO UNDO;
EXEC SQL SET TRANSACTION
READ WRITE
DIAGNOSTIC SIZE 5
ISOLATION LEVEL SERIALIZABLE;
EXEC SQL INSERT INTO EMPLOYEE (Fname, Lname, Ssn, Dno, Salary)
VALUES ('Robert', 'Smith', '991004321', 2, 35000);
EXEC SQL UPDATE EMPLOYEE
SET Salary = Salary * 1.1 WHERE Dno = 2;
EXEC SQL COMMIT;
GOTO THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END: . . . ;
```

The above transaction consists of first inserting a new row in the EMPLOYEE table and then updating the salary of all employees who work in department 2. If an error occurs on any of the SQL statements, the entire transaction is rolled back. This implies that any updated salary would be restored to its previous value and that the newly inserted row would be removed.

c. Discuss the UNDO and REDO operations and the recovery techniques that use each **(06 marks)**

Ans.

- The **deferred update** techniques do not physically update the database on disk until *after* a transaction reaches its commit point; then the updates are recorded in the database.
- Before reaching commit, all transaction updates are recorded in the local transaction workspace or in the main memory buffers that the DBMS maintains (the DBMS main memory cache). Before commit, the updates are recorded persistently in the log, and then after commit, the updates are written to the database on disk.
- If a transaction fails before reaching its commit point, it will not have changed the database in any way, so UNDO is not needed. It may be necessary to REDO the effect of the operations of a committed transaction from the log, because their effect may not yet have been recorded in the database on disk. Hence, deferred update is also known as the **NO-UNDO/REDO algorithm**.
- In the **immediate update** techniques, the database *may be updated* by some operations of a transaction *before* the transaction reaches its commit point. However, these operations must also be recorded in the log *on disk* by force-writing *before* they are applied to the database on disk, making recovery still possible. If a transaction fails after recording some changes in the database on disk but before reaching its commit point, the effect of its operations on the database must be undone; that is, the transaction must be rolled back. In the general case of immediate update, both *undo* and *redo* may be required during recovery. This technique, known as the **UNDO/REDO algorithm**, requires both operations during recovery, and is used most often in practice. A variation of the algorithm where all updates are required to be recorded in the database on disk *before* a transaction commits requires *undo* only, so it is known as the **UNDO/NO-REDO algorithm.**

OR

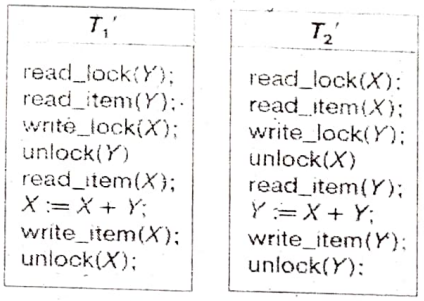
10. a. What is two-phase looking protocol? How does it guarantee serializability? **(04 marks)**

Ans.

- The main techniques used to control concurrent execution of transactions are based on the concept of locking data items.
- 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.
- The transaction can be divided into two phases: an **expanding** or **growing (first) phase**, during which new locks on items can be acquired but none can be released and a **shrinking (second) phase**, during which existing locks can be released but no new locks can be acquired.
- If lock conversion is allowed, then upgrading of locks i.e. from read-locked to

write-locked must be done during the expanding phase, and downgrading of locks (from write-locked to read-locked) must be done in the shrinking phase. Hence, a read_lock(X) operation that downgrades an already held write lock on X can appear only in the shrinking phase.

- Assume transactions $T1$ and $T2$ do not follow the two-phase locking protocol because the write_lock(X) operation follows the unlock(Y) operation in $T1$, and similarly the write_lock(Y) operation follows the unlock(X) operation in $T2$.
- To enforce two-phase locking, the transactions can be rewritten as $T1\_$ and $T2\_$, as shown in Figure 2.4.



b. What is Serializability? How can serializability be ensured? Do you need to restrict concurrent execution of transaction to ensure serializability? Justify your answer. **(06 marks)**

Ans. Refer Q. No. 10. a., Model Question Paper - 1

c. Discuss the time - stamp ordering protocol for concurrency control.(06 Marks)

Ans. Refer Q. No. 10. a., Model Question Paper - 1