CBCS SCHEME

USN | | | | | | | | | | |

18CS53

Fifth Semester B.E. Degree Examination, Jan./Feb. 2023
Database Management System

Time: 3 hrs. Max. Marks: 100

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

Module-1

1 a. Describe the characteristics of database approach. **(08 Marks)**

b. List and explain the criteria for classification of DBMS. **(08 Marks)**

c. Write an ER diagram to represent CAR entity type with 2 key attributes Registration and Vehicle ID. **(04 Marks)**

OR

2 a. Write an UML class diagram notation for company conceptual schema. **(10 Marks)**

b. Define the following terms : i) Data Model ii) Schema iii) Instance iv) Canned transaction v) Data Manupulation Language (DML). **(10 Marks)**

Module-2

3 a. Explain the concepts of specialization and Generalization, with the help of VEHICLE superclass. **(08Marks)**

b. Explain the different Relational Model constraints. **(06 Marks)**

c. Create a table for the Works_In relationship shown in Fig. Q3(c). **(06 Marks)**

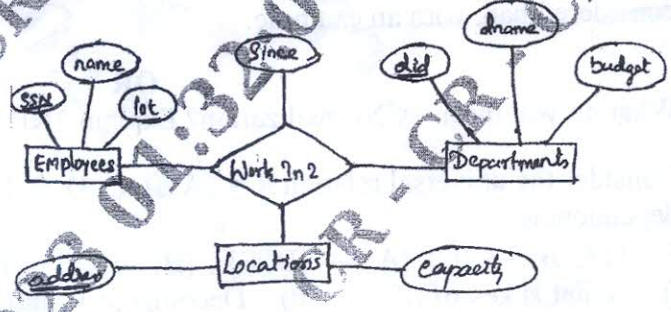


Fig. Q3(c)

OR

4 Considered the COMPANY DATABASE
EMPLOYEE (Fname , Minit , Lname , Ssn , Bdata , Address , Sex , Salary , Super_Ssn , Dno).
DEPARTMENT (Dname , Dnumber , Mgr_Ssn , Mgr_Start_data)
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 managers last name , address and birth date. **(06 Marks)**

b. Retrieve the birth date and address of the employees whose name is 'John B, Smith'. **(06 Marks)**

c. Retrieve the name and address of all employees who work for the 'Research' department. **(06 Marks)**

d. Retrieve the salary of every employee. **(02 Marks)**

Module-3

5 a. Discuss EXISTS and UNIQUE functions in SQL. Consider the COMPANY database given in Question number 04, write a query to list the name of the Manager who have atleast one dependent. **(08 Marks)**

b. With a real World example, explain the following :
i) JDBC ii) Correlated nested queries iii) Stored Procedures
iv) Schema change statements in SQL. **(12 Marks)**

OR

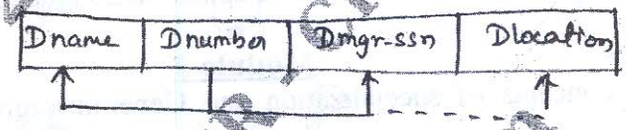
6 a. Explain the usage of Aggregate function in SQL. Write an SQL query to find sum of the salaries of all employees, the maximum salary, the minimum salary and the average salary by renaming the columns in a single row table. **(10 Marks)**

b. Create an HTML form to collect user id and password fields and it also has to have two buttons one for reset and another for login. **(06 Marks)**

c. Write a short note on : i) JavaScript ii) CGI **(04 Marks)**

Module-4

7 a. Describe the 3 main techniques to achieve first normal form for the relation by taking following examples schema. **(04 Marks)**



b. Discuss the Informal guidelines to determine the quality of relations schema design with a suitable example. **(10 Marks)**

c. Discuss the Insertion , Deletion and Modification anomalies. Illustrate, why are they considered bad, with an example. **(06 Marks)**

OR

8 a. What do you mean by Normalization? Explain 2NF and BCNF, with a suitable example. **(06 Marks)**

b. 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}}.
i) What is key of R? ii) Decompose R into 2NF and then 3NF relation. **(06 Marks)**

c. Write an algorithm to find a minimal cover F for a set of functional dependencies E. **(08 Marks)**

Module-5

9 a. Discuss the ACID properties of database transaction. **(08 Marks)**

b. Why concurrency control is needed? Demonstrate with an example. **(12 Marks)**

OR

10 a. Briefly explain 2 phase locking protocols. **(05 Marks)**

b. Explain Transaction support in SQL. **(05 Marks)**

c. Write a short note on :
i) Single user and Multiuser system.
ii) Transaction roll back and Cascadding roll back.
iii) Shadow paging.
iv) Database backup and recovery from catastrophic failure.
v) Deadlock prevention protocol. **(10 Marks)**

* * * * *

Model Solution for VTU Question Paper- Feb-2023

Database Management System (18CS53)

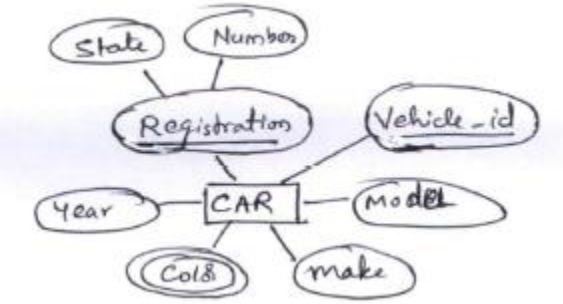
Q. No. 1a)Describe the characteristics ofDatabase Approach

1)Self-Describing Nature of a Database System.
2)Isolation between Programs and Data, and Data Abstraction
3)Support for Multiple Views of the Data
4)Sharing of knowledge and Multi-user Transaction Processing

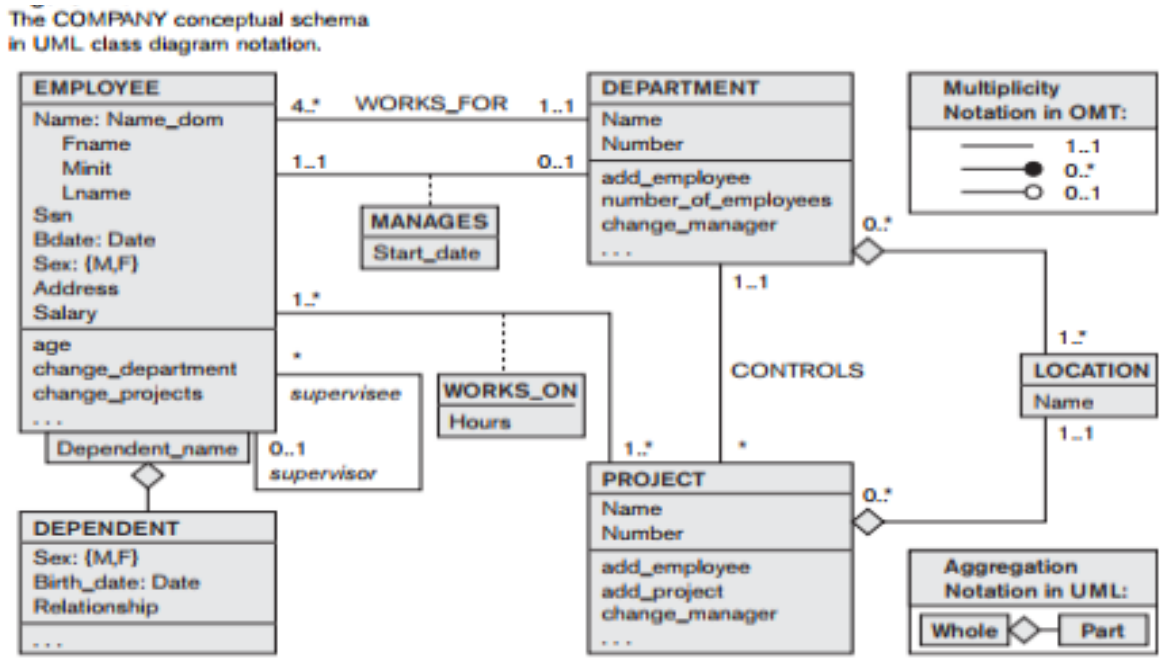
1b) List and explain the criteria for classification of DBMS

1)Classification Based on User Numbers.
2)Classification Based on Database Distribution
3) Centralized System
4)Distributed System
5)Homogeneous distributed database systems
6)Heterogeneous distributed database system.
7)Multi user database system
8)Single user database system
9)object oriented data model

1c)



Q. No. (2)a



b) Describe the Following:

DataModel:Data models describe how a database's logical structure is represented.Data models specify how data is linked to one another, as well as how it is handled and stored within the system.In a database management system, data models are often used to show how data is connected, stored, accessed, and changed.

Schema:A database schema defines how data is organized within a relational database; this is inclusive of logical constraints such as, table names, fields, data types, and the relationships between these entities.

Instance: In DBMS, the data is stored for a particular amount of time and is called an instance of the database. The database schema defines the attributes of the database in the particular DBMS. The value of the particular attribute at a particular moment in time is known as an instance of the DBMS.

Canned Transaction: The process of constantly querying and updating the database, using standard types of queries and updates.

DML(Data Manipulation Language):A DML (data manipulation language) refers to a computer programming language that allows you to add (insert), delete (delete), and alter (update) data in a database and merge contents of more than one table into a single.

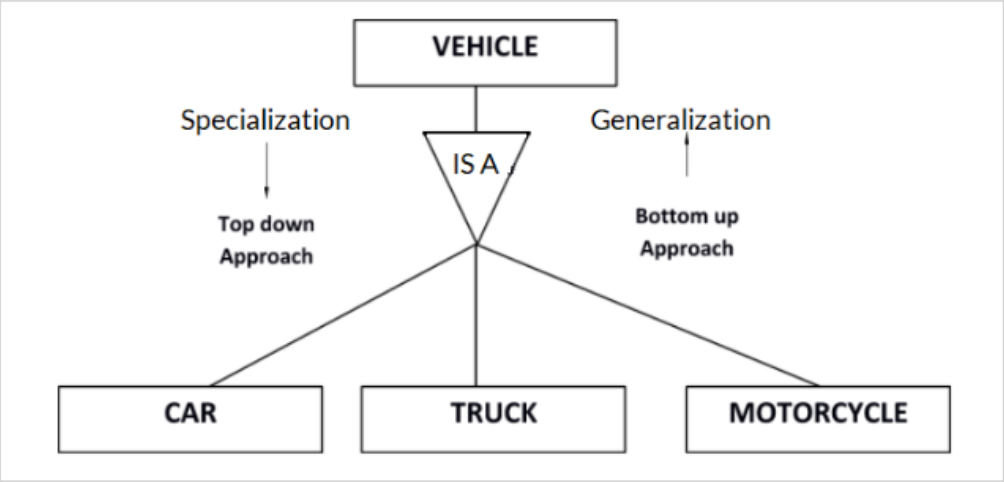
Q. No. (3)

a) Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

Generalization:It works on the principle of bottom up approach. In Generalization lower level functions are combined to form higher level functions which are called as entities. This process is repeated further to make advanced level entities.

In the Generalization process properties are drawn from particular entities and thus we can create generalized entities. We can summarize the Generalization process as it combines subclasses to form superclasses.

We can have three sub entities as Car, Truck, Motorcycle and these three entities can be generalized into one general super class as Vehicle.



It is a form of abstraction that specifies two or more entities (sub class) having common characters that can be generalized into one single entity (super class) at higher level hiding all the differences.

b)The Different Relational Model constraints are:

Constraints in the databases can be categorized into 3 main categories:

1. Constraints that are applied in the data model are called **Implicit constraints**.

2. Constraints that are directly applied in the schemas of the data model, by specifying them in the DDL(Data Definition Language). These are called schema-based **constraints or Explicit constraints**.
3. Constraints that cannot be directly applied in the schemas of the data model. We call these Application based or **semantic constraints**.

1. **Implicit constraints deal with:**

Mainly Constraints on the relational database are of 4 types:

1. Domain constraints
2. Key constraints
3. Entity Integrity constraints
4. Referential integrity constraints

1. Domain constraints :

1. Every domain must contain atomic values(smallest indivisible units) it means composite and multi-valued attributes are not allowed.
2. We perform datatype check here, which means when we assign a data type to a column we limit the values that it can contain. Eg. If we assign the datatype of attribute age as int, we cant give it values other then int datatype.

Example:

Example:

EID	Name	Phone
01	Bikash Dutta	123456789
		234456678

Explanation:

In the above relation, Name is a composite attribute and Phone is a multi-values attribute, so it is violating domain constraint.

2. Key Constraints or Uniqueness Constraints :

1. These are called uniqueness constraints since it ensures that every tuple in the relation should be unique.
2. A relation can have multiple keys or candidate keys(minimal superkey), out of which we choose one of the keys as primary key, we don't have any restriction on choosing the primary key out of candidate keys, but it is suggested to go with the candidate key with less number of attributes.
3. Null values are not allowed in the primary key, hence Not Null constraint is also a part of key constraint.

Example:

EID	Name	Phone
01	Bikash	6000000009
02	Paul	9000090009
01	Tuhin	9234567892

Explanation:

In the above table, EID is the primary key, and first and the last tuple has the same value in EID ie 01, so it is violating the key constraint.

3. Entity Integrity Constraints :

1. Entity Integrity constraints says that no primary key can take NULL value, since using primary key we identify each tuple uniquely in a relation.

Example:

EID	Name	Phone
01	Bikash	9000900099
02	Paul	600000009
NULL	Sony	9234567892

Explanation:

In the above relation, EID is made primary key, and the primary key cant take NULL values but in the third tuple, the primary key is null, so it is a violating Entity Integrity constraints.

4. Referential Integrity Constraints :

1. The Referential integrity constraints is specified between two relations or tables and used to maintain the consistency among the tuples in two relations.
2. This constraint is enforced through foreign key, when an attribute in the foreign key of relation R1 have the same domain(s) as the primary key of relation R2, then the foreign key of R1 is said to reference or refer to the primary key of relation R2.
3. The values of the foreign key in a tuple of relation R1 can either take the values of the primary key for some tuple in relation R2, or can take NULL values, but can't be empty.

Example:

Example:

EID	Name	DNO
01	Divine	12
02	Dino	22
04	Vivian	14

DNO	Place
12	Jaipur
13	Mumbai
14	Delhi

Explanation:

In the above, DNO of the first relation is the foreign key, and DNO in the second relation is the primary key.

DNO = 22 in the foreign key of the first table is not allowed since DNO = 22

is not defined in the primary key of the second relation. Therefore, Referential integrity constraints is violated here

c)create a table for Works_on

create table works_on(since year primary key);

create table employee(ssn int(5) primary key,name varchar(15),lot int(5),since year ,constraint emp_fk foreign key(since) references works_on(since));

create table Department(did int(5) primary key, dname varchar(15),budget int(9),constraint dept_fk foreign key(since) references works_on(since));

create table location(address varchar(16) primary key,capacity int(5),since year,constraint loc_fk foreign key(since) references works_on(since));

Q. No. (4)

Considered the COMPANY DATABASE
EMPLOYEE (Fname , Minit , Lname , Ssn , Bdata , Address , Sex , Salary , Super_Ssn , Dno).
DEPARTMENT (Dname , Dnumber , Mgr_Ssn , Mgr_Start_data)
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 managers last name , address and birth date. **(06 Marks)**

```
Select P.Pnumber, P.Dnum, E.Lname,E.Address, E.Bdata
from EMPLOYEE E, PROJECT P, DEPARTMENT D
WHERE P.Dnum=D.Dnumber and D.Mgr_Ssn=E.Ssn and PLocation = 'Stafford';
```

b) Retrieve the birth date and address of the employees whose name is 'John B, Smith'. **(06 Marks)**

```
Select Bdata, Address from EMPLOYEE where Fname='John' and Minit='B' and Lname='Smith';
```

c) Retrieve the name and address of all employees who work for the 'Research' department. **(06 Marks)**

```
Select E.Fname,E.Minit,E.Lname,E.Address
from EMPLOYEE E, DEPARTMENT D
WHERE E.Dno=D.Dnumber and Dname='Research';
```

d) Retrieve the salary of every employee. **(02 Marks)**

```
Select E.Ssn, E.Salary from EMPLOYEE;
```

Q. No. (5)

a) Discuss EXISTS and UNIQUE functions in SQL. Consider the COMPANY database given in Question number 04, write a query to list the name of the Manager who have atleast one dependent. **(08 Marks)**

EXISTS and UNIQUE are Boolean functions that return TRUE or FALSE; hence,
they can be used in a WHERE clause condition. The EXISTS function in SQL is used to check whether the result of a 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.

UNIQUE(Q), which returns TRUE if there are no duplicate tuples in the result of query Q; otherwise, it returns FALSE. This can be used to test whether the result of a nested query is a set (no duplicates) or a multiset (duplicates exist).

- SELECT Fname, Lname FROM EMPLOYEE WHERE EXISTS (SELECT * FROM DEPENDENT WHERE Ssn = ESsn) AND EXISTS (SELECT * FROM DEPARTMENT WHERE Ssn = Mgr_Ssn);

We have specified two nested correlated queries; the first selects all DEPENDENT tuples related to an EMPLOYEE, and the second selects all DEPARTMENT tuples managed by the EMPLOYEE. If at least one of the first and at least one of the second exists, we select the EMPLOYEE tuple.

b) With a real World example, explain the following :
i) JDBC ii) Correlated nested queries iii) Stored Procedures
iv) Schema change statements in SQL. (12 Marks)

JDBC: JDBC is SQL Class Library for Java Programming. The class libraries and associated function calls for SQL are accessed in java using JDBC. The Java programming language was designed to be platform independent—that is, a program should be able to run on any type of computer system that has a Java interpreter installed. Because of this portability, many RDBMS vendors provide JDBC drivers so that it is possible to access their systems via Java programs.

A JDBC driver is basically an implementation of the classes and associated objects and function calls specified in JDBC for a particular vendor's RDBMS. Hence, a Java program with JDBC objects and function calls can access any RDBMS that has a JDBC driver available. Because Java is object-oriented, its function libraries are implemented as classes. Before being able to process JDBC function calls with Java, it is necessary to import the JDBC class libraries, which are called java.sql.*. These can be downloaded and installed via the Web.

To load a JDBC driver explicitly, the generic Java function for loading a class can be used. For example, to load the JDBC driver for the Oracle RDBMS, the following command can be used:

Class.forName("oracle.jdbc.driver.OracleDriver")

Example:

```
//Program JDBC1:
import java.io.* ;
import java.sql.*

class getEmpInfo {
public static void main (String args []) throws SQLException, IOException {
try { Class.forName("oracle.jdbc.driver.OracleDriver")
} catch (ClassNotFoundException x) {
```

```
System.out.println ("Driver could not be loaded") ;
}
String dbacct, passwrd, ssn, lname ;
Double salary ;
 dbacct = readentry("Enter database account:") ;
 passwrd = readentry("Enter password:") ;
 Connection conn = DriverManager.getConnection
 ("jdbc:oracle:oci8:" + dbacct + "/" + passwrd) ;
 String stmt1 = "select Lname, Salary from EMPLOYEE where Ssn = ?" ;
 PreparedStatement p = conn.prepareStatement(stmt1) ;
 ssn = readentry("Enter a Social Security Number: ") ;
 p.clearParameters() ;
 p.setString(1, ssn) ;
 ResultSet r = p.executeQuery() ;
 while (r.next()) {
 lname = r.getString(1) ;
 salary = r.getDouble(2) ;
 system.out.printline(lname + salary) ;
 } }
 }
```

Following steps are followed in the order in above example:

Import the JDBC class library, Load the JDBC driver, Create appropriate variables, Create Connection object, create Prepared Statement object, Setting the statement parameters, Binding the statement parameters, Executing the SQL statement, Processing the ResultSet object.

Correlated nested queries:

Whenever a condition in the WHERE clause of a nested query references some attribute of a relation declared in the outer query, the two queries are said to be correlated. We can understand a correlated query better by considering that the nested query is evaluated once for each tuple (or combination of tuples) in the outer query.

Example:

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

In this example:

For each EMPLOYEE tuple, evaluate the nested query, which retrieves the Essn values for all DEPENDENT tuples with the same sex and name as that EMPLOYEE tuple; if the Ssn value of the EMPLOYEE tuple is in the result of the nested query, then select that EMPLOYEE tuple.

Stored Procedures:

Database program modules—procedures that are stored and executed by the DBMS at the database server are called stored procedures. They are also called persistent stored modules because these programs are stored persistently by the DBMS.

Stored procedures are useful in the following circumstances:

- If a database program is needed by several applications, it can be stored at the server and invoked by any of the application programs. This reduces duplication of effort and improves software modularity.
- Executing a program at the server can reduce data transfer and communication cost between the client and server in certain situations.
- These procedures can enhance the modeling power provided by views by allowing more complex types of derived data to be made available to the database users via the stored procedures. Additionally, they can be used to check for complex constraints that are beyond the specification power of assertions and triggers.

Example:

```
CREATE FUNCTION Dept_size(IN deptno INTEGER)
RETURNS VARCHAR [7]
DECLARE No_of_emps INTEGER ;
SELECT COUNT(*) INTO No_of_emps
FROM EMPLOYEE WHERE Dno = deptno ;
IF No_of_emps > 100 THEN RETURN "HUGE"
ELSEIF No_of_emps > 25 THEN RETURN "LARGE"
ELSEIF No_of_emps > 10 THEN RETURN "MEDIUM"
ELSE RETURN "SMALL"
END IF ;
```

This example the procedure returns a string value (line 1) describing the size of a department within a company based on the number of employees. There is one IN integer parameter, deptno, which gives a department number. A local variable NoOfEmps is declared in line 2. The query in lines 3 and 4 returns the number of employees in the department, and the conditional branch in lines 5 to 8 then returns one of the values {'HUGE', 'LARGE', 'MEDIUM', 'SMALL'} based on the number of employees.

Schema Change statements in SQL:

The DROP Command: The DROP command can be used to drop named schema elements, such as tables, domains, types, or constraints. One can also drop a whole schema if it is no longer needed by using the DROP SCHEMA command.

Example:

```
DROP SCHEMA COMPANY;
```

DROP TABLE DEPENDENT;

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

Q. No. (6)

a) Explain the usage of Aggregate function in SQL. Write an SQL query to find sum of the salaries of all employees, the maximum salary, the minimum salary and the average salary by renaming the columns in a single row table. **(10 Marks)**

Aggregate functions are used to summarize information from multiple tuples into a single-tuple summary. Grouping is used to create subgroups of tuples before summarization. Grouping and aggregation are required in many database applications. 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. These functions can be used in the SELECT clause or in a HAVING clause (which we introduce later). The functions MAX and MIN can also be used with attributes that have nonnumeric domains if the domain values have a total ordering among one another. We illustrate the use of these functions with several queries.

SELECT SUM (Salary) AS Total_Sal, MAX (Salary) AS Highest_Sal, MIN (Salary) AS Lowest_Sal, AVG (Salary) AS Average_Sal FROM EMPLOYEE;

b) Create an HTML form to collect user id and password fields and it also has to have two buttons one for reset and another for login. **(06 Marks)**

```
<!DOCTYPE html>
<html lang="en">
<head> <meta charset="UTF-8">
```

```
<title>Login Form</title>
</head>
<body>
<form id="login-form">
        <label for="userid">User ID:</label>
        <input type="text" id="userid" name="userid"><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password"><br>
        <input type="reset" value="Reset">
        <input type="submit" value="Login">
</form>
</body>
</html>
```

c) Write a short note on : i) JavaScript ii) CGI (04 Marks)

JavaScript is a programming language that is widely used in web development. It is a high-level language that is designed to be easy to read and write, and is interpreted by web browsers. JavaScript is primarily used to add interactivity and dynamic behavior to web pages, such as creating responsive user interfaces, validating form input, and updating content without requiring a page reload. JavaScript is also used to create web applications that run entirely in the browser, known as single-page applications. This is made possible by JavaScript frameworks and libraries such as React, Angular, and Vue, which provide powerful tools for building complex web applications. In addition to web development, JavaScript is also used in server-side programming with Node.js, which allows developers to use JavaScript on the server to build scalable and high-performance web applications. JavaScript has a vast ecosystem of tools and libraries that make it easy to build and maintain web applications, and it continues to evolve rapidly, with new features and standards being added regularly.

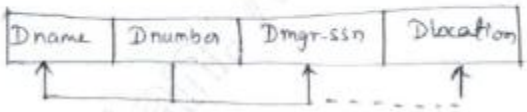
CGI (Common Gateway Interface) is a protocol that allows web servers to execute scripts and programs on the server in response to HTTP requests. CGI was created in the early days of the web to allow web servers to dynamically generate content and respond to user input, such as form submissions. With CGI, a web server receives an HTTP request from a client, such as a web browser, and sends the request to a CGI script or program. The script or program generates a response, which is sent back to the web server, which then sends it back to the client in the form of an HTTP response. CGI scripts can be written in a variety of programming languages, including Perl, Python, Ruby, and C++. The output of a CGI script can be in any format, such as HTML, XML, JSON, or plain text. While CGI was once a popular way to build dynamic web applications, it has largely been replaced by more modern technologies, such as server-

side scripting with PHP, Node.js, and other frameworks. However, CGI scripts can still be useful in certain situations, such as when building custom server-side functionality or working with legacy applications.

Q. No. (7)

a)

a. Describe the 3 main techniques to achieve first normal form for the relation by taking following examples schema. **(04 Marks)**



1. Remove the attribute Dlocation as it violates 1NF and place it in separate relation
2. Expand the key so that there is a separate tuple in the relation Department.
3. If Dlocation is a multivalued attribute, replace it with atomic value.

b)

b. Discuss the Informal guidelines to determine the quality of relations schema design with a suitable example. **(10 Marks)**

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

(1) Imparting Clear Semantics to Attributes in Relations

•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

if a relation schema corresponds to one entity type or one relationship type, it is straightforward to interpret and to explain its meaning. if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained.

Examples of Violating Guideline 1: Company Database

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION

logically correct but they violate Guideline 1 by mixing attributes from distinct real-world entities:

•EMP_DEPT mixes attributes of employees and departments

•EMP_PROJ mixes attributes of employees and projects and the WORKS_ON relationship

(2) Removing Redundant Information in Tuples and Update Anomalies

§One goal of schema design is to minimize the storage space used by the base relations
§Grouping attributes into relation schemas has a significant effect on storage space

update anomalies
- insertion anomalies
- deletion anomalies,
- modification anomalies

(3) Removing NULL Values in Tuples

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

(4) Removing Spurious Tuples

Spurious Tuples are those rows in a table, which occur as a result of joining two tables in wrong manner. They are extra tuples (rows) which might not be required
Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated
§Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

c)

c. Discuss the Insertion , Deletion and Modification anomalies. Illustrate, why are they considered bad, with an example. **(06 Marks)**

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555

Insertion Anomalies

1) To insert a new employee tuple into EMP_DEPT, we must include either the attribute values for the department that the employee works for, or NULLs.

- 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
- In the design of Employee in fig 1, we do not have to worry about this consistency problem because we enter only the department number in the employee tuple; all other attribute values of department 5 are recorded only once in the database, as a single tuple in the DEPARTMENT relation

2.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
- This violates the entity integrity for EMP_DEPT because Ssn is its primary key
- This problem does not occur in the design of Figure 1 because a department is entered in the DEPARTMENT relation whether or not any employees work for it, and whenever an employee is assigned to that department, a corresponding tuple is inserted in EMPLOYEE.

Deletion Anomalies

§ The problem of deletion anomalies is related to the second insertion anomaly situation just discussed
- 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 we change the value of one of the attributes of a particular department—say, the manager of department 5—we must update the tuples of all employees who work in that department; 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.

Q. No. (8)

a)

a. What do you mean by Normalization? Explain 2NF and BCNF, with a suitable example. **(06 Marks)**

Normalization of data can be considered a process of analysing 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

Second Normal Form (2NF)

•In the 2NF, relational must be in 1NF.
•In the second normal form, all non-key attributes are fully functional dependent on the primary key
Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Boyce Codd normal form (BCNF)

•BCNF is the advanced version of 3NF. It is stricter than 3NF.

Rules for BCNF in DBMS

A table or relation is said to be in BCNF (Boyce Codd Normal Form) if it satisfies the following two conditions :

•It should satisfy all the conditions of the Third Normal Form (3NF)

•For any functional dependency (A->B), A should be either the super key or the candidate key. In simple words, it means that A can't be a non-prime attribute if B is given as a prime attribute.

In this example, we have a relation R with three columns: Id, Subject, and Professor.

Id	Subject	Professor
101	Java	Mayank
101	C++	Kartik
102	Java	Sarthak
103	C#	Lakshay
104	Java	Mayank

In this Example , we will decompose the table into two tables: the Student table and the Professor table to satisfy the conditions of BCNF.

Student Table

P_Id	S_Id	Professor
1	101	Mayank
2	101	Kartik
3	102	Sarthak
4	103	Lakshay
5	104	Mayank

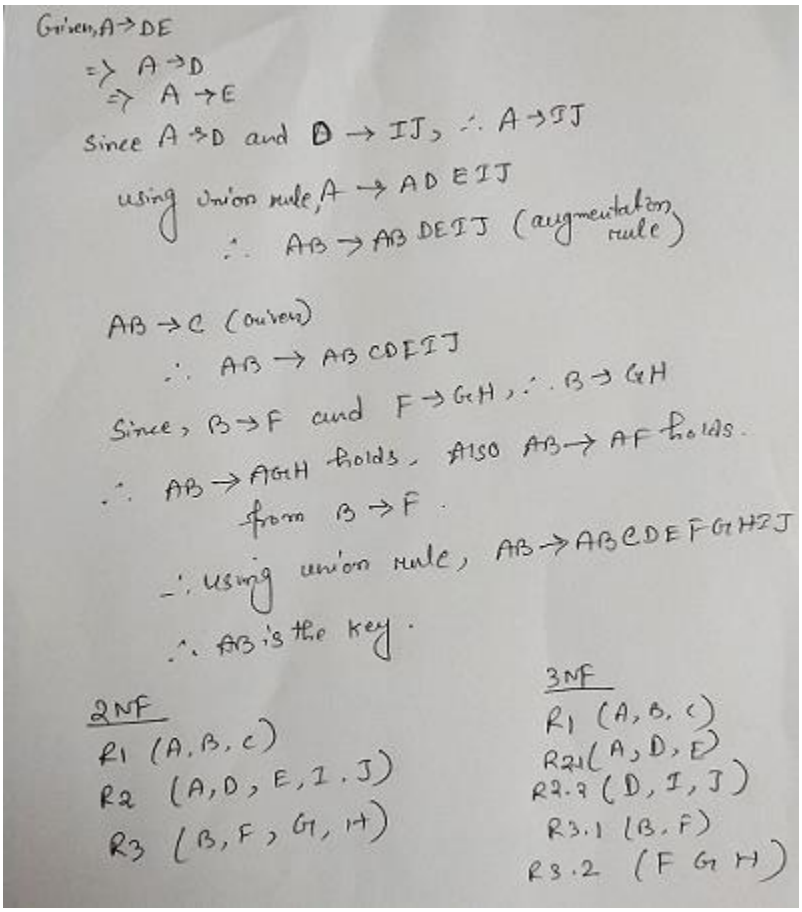
Professor Table

Professor	Subject
Mayank	Java
Kartik	C++
Sarthak	Java
Lakshay	C#
Mayank	Java

Professor is now the primary key and the prime attribute column, deriving the subject column. Hence, it is in BCNF.

b)

b. 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}}.
i) What is key of R? ii) Decompose R into 2NF and then 3NF relation. **(06 Marks)**



c)

c. Write an algorithm to find a minimal cover F for a set of functional dependencies E. **(08 Marks)**

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 $X \rightarrow \{A_1, A_2, ..., A_n\}$ in F by the n functional dependencies $X \rightarrow A_1, X \rightarrow A_2, ..., X \rightarrow A_n$.
3. For each functional dependency $X \rightarrow A$ in F
   for each attribute B that is an element of X
   if $\{ \{F - \{X \rightarrow A\}\} \cup \{ (X - \{B\}) \rightarrow A\} \}$ 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.

- Step 2 places FDs in a canonical form for subsequent testing
- Step 3 constitutes removal of an extraneous attribute B contained in the left-hand side Xof a functional dependency X->A from F when possible
- Step 4 constitutes removal of a redundant functional dependency x->A from F whenpossible

Q. No. (9)

a)

a. Discuss the ACID properties of database transaction. **(08 Marks)**

Transactions should possess several properties, often called the **ACID properties**

A Atomicity:

a transaction is an atomic unit of processing and it is either performed entirely or not at all.

C Consistency Preservation:

a transaction should be consistency preserving that is it must take the database from one consistent state to another.

I Isolation/Independence:

A transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executed concurrently.

D Durability (or Permanency):

if a transaction changes the database and is committed, the changes must never 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.

The preservation of **consistency** is generally considered to be the responsibility of the programmers who write the database programs or of the DBMS module that enforces integrity constraints.

The **isolation** property is enforced by the concurrency control subsystem of the DBMS. If every transaction does not make its updates (write operations) visible to other transactions until it is committed, one form of isolation is enforced that solves the temporary update problem and eliminates cascading rollbacks

Durability is the responsibility of recovery subsystem.

b)

b. Why concurrency control is needed? Demonstrate with an example. **(12 Marks)**

Several problems can occur when concurrent transactions execute in an uncontrolled manner

Example:

•We consider an Airline reservation DB

•Each records is stored for an airline flight which includes Number of reserved seats among other information.

Types of problems we may encounter:

1.The Lost Update Problem[WW conflict]

2.The Temporary Update/ Dirty Read Problem[WR conflict]

3.The Incorrect Summary Problem

4.The Unrepeatable Read Problem

[RW conflict]

T_1	T_2

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);
	read_item(X);
	$X := X + M$;
	write_item(X);

Transaction T1

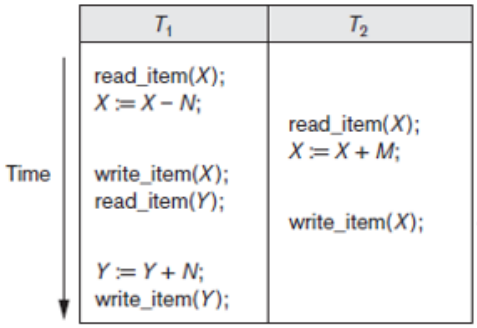
•transfers N reservations from one flight whose number of reserved seats is stored in the database item named X to another flight whose number of reserved seats is stored in the database item named Y.

Transaction T2

•reserves M seats on the first flight (X)

1. The Lost Update Problem [WW conflict]

- Occurs when two transactions that access the same DB items have their operations interleaved in a way that makes the value of some DB item incorrect
- Suppose that transactions T1 and T2 are submitted at approximately the same time, and suppose that their operations are interleaved as shown in Figure below



Item X has an incorrect value because its update by T_1 is *lost* (overwritten).

- 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.

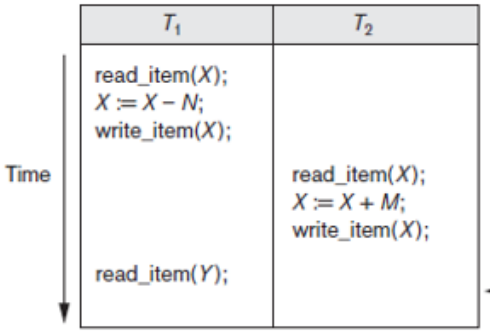
- For example:

X = 80 at the start (there were 80 reservations on the flight)
N = 5 (T1 transfers 5 seat reservations from the flight corresponding to X to the flight corresponding to Y)
M = 4 (T2 reserves 4 seats on X) The final result should be X = 79.

- The interleaving of operations shown in Figure is X = 84 because the update in T1 that removed the five seats from X was lost.

2. The Temporary Update / Dirty Read Problem [WR conflict]

- occurs when one transaction updates a database item and then the transaction fails for some reason
- Meanwhile the updated item is accessed by another transaction before it is changed back to its original value



Transaction T_1 fails and must change the value of X back to its old value; meanwhile T_2 has read the *temporary* incorrect value of X.

3. The Incorrect Summary Problem

•If one transaction is calculating an aggregate summary function on a number of DB items while other transactions are updating some of these items, the aggregate function may calculate some values before they are updated and others after they are updated.

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).

4. The Unrepeatable Read Problem [RW conflict]

- Transaction T reads the same item twice and gets different values on each read, since the item was modified by another transaction T` between the two reads.
- for example, if during an airline reservation transaction, a customer inquires 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, and it may end up reading a different value for the item.

Q. No. (10)

a)

a. Briefly explain 2 phase locking protocols. **(05 Marks)**

Two-phase locking (2PL) protocol divides the execution phase of a transaction into three parts.
In the first part, when the transaction starts executing, it seeks permission for the locks it requires.
It is a concurrency control method that guarantees serializability
The second part is where the transaction acquires all the locks.
As soon as the transaction releases its first lock, the third phase starts.
•In this phase, the transaction cannot demand any new locks.
•It only releases the acquired locks.

•The protocol utilizes locks, applied by a transaction to data, which may block other transactions from accessing the same data during the transaction's life.

A transaction is said to follow the two-phase locking protocol if all locking operations (read_lock, write_lock) precede the first unlock operation in the transaction

§ Such a transaction can be divided into two phases:

§ **Expanding or growing (first) phase**, during which new locks on items can be acquired but none can be released

§ **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 (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.

b)

b. Explain Transaction support in SQL. **(05 Marks)**

Transaction Support in SQL

The basic definition of an SQL transaction is, it is a logical unit of work and is guaranteed to be atomic

A single SQL statement is always considered to be atomic—either it completes execution without an error or it fails and leaves the database unchanged.

Every transaction must have an explicit end statement, which is either a COMMIT or a ROLLBACK.

The characteristics are :

•***The access mode***

- can be specified as READ ONLY or READ WRITE
- The default is READ WRITE
- A mode of READ WRITE allows select, 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***

- DIAGNOSTIC SIZE n, specifies an integer value n, which indicates the number of conditions that can be held simultaneously in the diagnostic area.

•***The isolation level***

- specified using the statement ISOLATION LEVEL <isolation>, 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 one or more of the following three violations may occur:

1.*Dirty read:* A transaction T1 may read the update of a transaction T2, which has not yet committed.

2.*Nonrepeatable read:* A transaction T1 may read a given value from a table. If another transaction T2 later updates that value and T1 reads that value again, T1 will see a different value.

3. *Phantoms:* A transaction T1 may read a set of rows from a table, perhaps based on some condition specified in the SQL WHERE-clause. Now suppose that a transaction T2 inserts a new row that also satisfies the WHERE-clause condition used in T1, into the table used by T1. If T1 is repeated, then T1 will see a phantom, a row that previously did not exist.

c)

c. Write a short note on :
   i) Single user and Multiuser system.
   ii) Transaction roll back and Cascadding roll back.
   iii) Shadow paging.
   iv) Database backup and recovery from catastrophic failure.
   v) Deadlock prevention protocol. **(10 Marks)**

i) One criterion for classifying a database system is according to the number of users who can use the system concurrently

Single-User versus Multi User Systems

▪ A DBMS is

• single-user

- at most one user at a time can use the system
- Eg: Personal Computer System

• multi user

- many users can use the system and hence access the database concurrently
- Eg: Airline reservation database

ii) If a transaction fails for whatever reason after updating the database, but before the transaction commits, it may be necessary to roll back the transaction

▪ If any data item values have been changed by the transaction and written to the database, they must be restored to their previous values (BFIMs)

▪ The undo-type log entries are used to restore the old values of data items that must be rolled back

If a transaction T is rolled back, any transaction S that has, in the interim, read the value of some data item X written by T must also be rolled back

▪ Similarly, once S is rolled back, any transaction R that has read the value of some data item Y written by S must also be rolled back; and so on.

▪ This phenomenon is called cascading rollback, and can occur when the recovery protocol ensures recoverable schedules but does not ensure strict or cascadeless schedules

iii) 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 fixed size disk pages (or disk blocks)—say, n—for recovery purposes

A directory with n entries5 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.

iv) A key assumption has been that the system log is maintained on the disk and is not lost as a result of the failure.

- Similarly, the shadow directory must be stored on disk to allow recovery when shadow paging is used.
- The recovery techniques use the entries in the system log or the shadow directory to recover from failure by bringing the database back to a consistent state.
- The recovery manager of a DBMS must also be equipped to handle more catastrophic
- failures such as disk crashes.
- The main technique used to handle such crashes is a database backup, in which the whole database and the log are periodically copied onto a cheap storage medium such as magnetic tapes or other large capacity offline storage devices.
- In case of a catastrophic system failure, the latest backup copy can be reloaded from the tape to the disk, and the system can be restarted.
- Data from critical applications such as banking, insurance, stock market, and other databases is periodically backed up in its entirety and moved to physically separate safe locations.

v) 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 advance. 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 (or the system) is aware of the chosen order of the items
- Both approaches impractical

▪ Some of these techniques use the concept of transaction timestamp TS(T), which is a unique identifier assigned to each transaction

▪ The timestamps are typically based on the order in which transactions are started; hence, if transaction T1 starts before transaction T2, then TS(T1) < TS(T2).

▪ The older transaction (which starts first) has the smaller timestamp value.

▪ Protocols based on a timestamp

• Wait-die

• Wound-wait