CBCS Scheme

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

16MCA23

# Second Semester MCA Degree Examination, June/July 2017
## Database Management System

Time: 3 hrs. Max. Marks: 80

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



### Module-1

1 a. List the advantages of DBMS over traditional file system. Briefly explain them. **(06 Marks)**

b. Define and explain the importance of database catalog. Explain the internal storage format of a catalog with an example. **(06 Marks)**

c. What are the responsibilities of DBA? **(04 Marks)**

**OR**

2 a. Explain the operations for a two tier client – server architecture for DBMS. **(06 Marks)**

b. What is meant by "Persistant storage for program objects"? Explain. **(04 Marks)**

c. Describe the various steps of an algorithm for ER to relational mapping with the help of Company relational database schema. **(06 Marks)**

### Module-2

3 a. Define the following terms : i) Join ii) Division iii) Cartesian product iv) Union v) Set difference. **(10 Marks)**

b. Consider the following relations and write relational algebra queries :
EMPLOYEE (Fname , SSN , Salary , Super_SSN , Dno) ;
WORKSON (ESSN, Pno , Hours) ;
DEPARTMENT (Dname, Dnumber, Mgr_SSN) ;
DEPENDENT (ESSN , Dependent_name) ;
i) Retrieve the highest salary paid in each department.
ii) Retrieve the name of managers who have more than two dependents.
iii) Retrieve the number of employee's and their average salary working in each department. **(06 Marks)**

**OR**

4 a. What are Integrity constraints? Discuss the various update operations on relations and the type of integrity constraints that must be checked for each update operation. **(10 Marks)**

b. Explain SELECT and PROJECT operation with suitable example. **(06 Marks)**

### Module-3

5 a. Discuss insertion , deletion and updation anomalies by taking suitable examples. **(08 Marks)**

b. Explain i) Aggregate functions ii) Embedded SQL. **(08 Marks)**

**OR**

6 a. How is a view created and dropped? What problems are associated with updating of views? **(08 Marks)**

b. Explain SQL data definition and data types in brief and explain DROP and ALTER command. **(08 Marks)**

## Module-4

7 a. Define Functional dependency. Explain 1NF , 2NF and 3NF, with example for each . **(08 Marks)**

b. What is a Trigger? Explain DML trigger, with an example. **(08 Marks)**

**OR**

8 a. Explain the Informal guidelines for the relational schema. **(08 Marks)**

b. A relation R has four attributes A B C D. For each of the following sets of FD, identify the candidate key and weight normal form.

i) $C \rightarrow D$ , $C \rightarrow A$ , $B \rightarrow C$ ii) $B \rightarrow C$ , $D \rightarrow A$. **(08 Marks)**

## Module-5

9 a. What is Transaction? In what ways it is different from an ordinary program? **(06 Marks)**

b. Explain all the phases involved in ARIES algorithm, with an example. **(10 Marks)**

**OR**

10 a. Explain i) ACID properties ii) Strict two phase locking. **(08 Marks)**

b. Explain the database recovery technique based on different update. **(08 Marks)**

* * * * *

1. **a. List the advantages of DBMS over traditional file system. Briefly explain them.(Previous syllabus)**

Advantages of Database management system over file system.

- No redundant data – Redundancy removed by data normalization
- Data Consistency and Integrity
- Secure
- Privacy Easy access to data
- Easy recovery
- Flexible

b. **Define and explain the importance of database catalog. Explain the internal storage format of a catalog with an example. (Out of Syllabus)**

A DBMS catalog stores the description (structure, type, storage format of each entities) of the database. The description is called meta-data). This allows the DBMS software to work with different databases

A catalog is a directory of information about data sets, files, or a database . A catalog usually describes where a data set, file or database entity is located and may also include other information, such as the type of device on which each data set or file is stored.

| Data Item Name | Starting Position in Record | Length in Characters (bytes) |
|---|---|---|
| Name | 1 | 30 |
| StudentNumber | 31 | 4 |
| Class | 35 | 4 |
| Major | 39 | 4 |

c. **What are the responsibilities of DBA?**

The DBA is responsible for authorizing access to the database coordinating and monitoring its use.

- Maintaining all databases required for development, testing, training and production usage
- Maximizing uptime of databases
- Managing share resources used amongst applications
- Administrates all database objects, including tables, views, indexes, stored procedures, functions, packages, sequences and clusters
- Enforces and maintains database constraints to ensure integrity of the database
- Installation and configuration of DBMS server software and related products.
- Upgrading and patching/hot-fixing of DBMS server software and related products.
- Assists with impact analysis of any changes made to the database objects.
- Migrate database to another server.
- Evaluate DBMS features and DBMS related products.
- Ensure that the site is running the products that are most appropriate
- ensure that any new product usage or release upgrade takes place with minimal impact
- Establish and maintain sound backup and recovery policies and procedures.
- Take care of the Database design and implementation.
- Implementing HA Solution (Replication, Clustering, Mirroring and Log Shipping)
- Implement and maintain database security (create and maintain logins, users and roles, assign privileges).
- Performance tuning and health monitoring on DBMS, OS and application.
- Setting Up Server Level, Database Level and Operating System Alerts
- Implementation of robust maintenance plan (Index Defrag, Stats Update, DBCC, Reorg etc)
- SQL Server T-SQL/ Oracle PL-SQL Tuning
- Setup and maintain documentation and standards.
- Perform reviews on the design and code frequently to ensure the standards are being adhered to
- Plan growth and changes (capacity planning).
- Do general technical troubleshooting and give consultation to development teams.
- Troubleshooting on DBMS and Operating System performance issue
- Documentation of any implementation and changes (database changes, reference data changes and application UI changes etc)

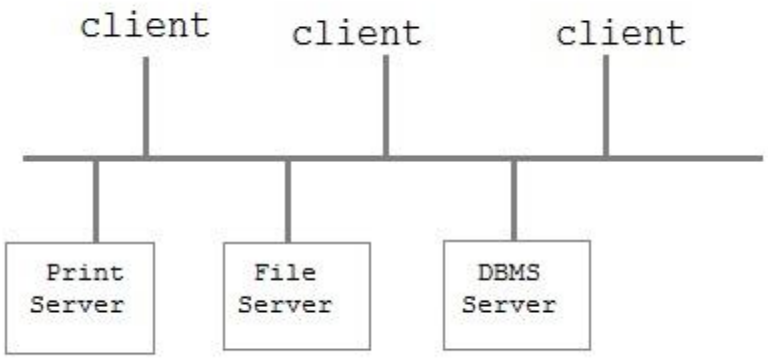
Be able to provide a strategic database direction for the organization.

Expert level knowledge of DBMS Architecture, all features in all versions and troubleshooting skill

- Excellent knowledge of DMBS backup and recovery scenarios.
- Good skills in all DMBS tools.
- A good knowledge of DMBS security management.
- A good knowledge of how DMBS acquires and manages resources.
- Sound knowledge of the applications at your site.

2. **a Explain the operations for a two-tier Client-Server architecture for DBMS**.

Two-tier Client / Server Architecture




Two-tier Client / Server architecture is used for User Interface program and Application Programs that runs on client side. An interface called ODBC(Open Database Connectivity) provides an API that allow client side program to call the dbms. Most DBMS vendors provide ODBC drivers. A client program may connect to several DBMS's. In this architecture some variation of client is also possible for example in some DBMS's more functionality is transferred to the client including data dictionary, optimization etc. Such clients are called Data server.

**b What is meant by "Persistent storage for program objects"? Explain. (Out of Syllabus)**

Persistent storage is any data storage device that retains data after power to that device is shut off. It is also sometimes referred to as non-volatile storage.

Hard disk drives and solid-state drives are common types of persistent storage. This can be in the form of file, block or object storage. In containerization, persistent storage refers to storage volumes -- usually associated with state ful applications such as databases -- that remain available beyond the life of individual containers. Persistent storage volumes can be contrasted with ephemeral storage volumes that live and die with containers and are associated with stateless apps.

Persistent objects are the fundamental logical units of data storage in an Objectivity database.

Persistence is relevant for objects with an internal state. The state needs to be retained between object deactivation and object activation

Persistent objects are created by applications. Each persistent object has an object identifier (OID) and is an instance of a class. The class may be defined in an object-oriented programming language, or it may be a class defined by Objectivity/DB (for example, ooObj). In either case, the class is described in the federated database's schema.

A persistent object continues to exist and retain its data beyond the duration of the process that creates it.

**c. Describe the various steps of an algorithm for ER to relational mapping with the help of company relational database scheme.**

Step 1: Mapping of regular entity type
Step 2: Mapping of weak entity type
Foreign key approach
Merged relation approach
Cross reference or relationship relation approach
Step 3: Mapping of binary 1:1 relation types
Step 4: Mapping of binary 1:N relationship types.
Step 5: Mapping of binary M:N relationship types.
Step 6: Mapping of multivalued attributes
Step 7: Mapping of N-ary relationship types.

**3. a Define the following terms: 1) Join 2) Division 3)Cartesian product 4)Union 5)Set difference**

1. Join (2 M)

   A SQL join clause combines records from two or more tables in a relational database. It creates a set that can be saved as a table or used as it is. A JOIN is a means for combining fields from two tables (or more) by using values common to each.

2. Cartesian Product(2 M)

   A Cartesian product of n sets, also known as a n-foldCartesian product, can be represented by an array of n dimensions, where each element is an n-tuple. An ordered pair is a 2-tuple or couple

3. Union(2 M)

   It performs binary union between two given relations and is defined as –

$$r \cup s = \{ t \mid t \in r \text{ or } t \in s\}$$

Notion – r U s

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

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

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

∏ author (Books) ∪ ∏ author (Articles)

4. Set Difference (2 M)

   The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

   Notation – r – s

   Finds all the tuples that are present in r but not in s.

∏ author (Books) − ∏ author (Articles)

5. Division

The formal definition of division is as follows:

$$A/B = \pi x(A) - \pi x((\pi x(A) \times B) - A)$$

The division operator denoted by ÷ is suited to queries that include the phrase "for all". suppose that we wish to find all customers who hav3e an account at all the branches located at Brooklyn. We obtain all branches in Booklyn by the expression

r1= ∏ br_name amount (σ br_city = "Brooklyn" (branch)

we can find all customer-name, br_name pairs for which the customer has an account at a branch by writing

r2=∏ customer_name,br_name (account |x| depositor)

now we have to find customers who appear in r2 with every branch name in r1.the operation that provide this is r2 ÷ r1.

**4.** a **What are integrity constraints? Discuss the various update operations on relations and the type of integrity constraints that must be checked for each update operation.**

Integrity constraints provide a way of ensuring that changes made to the database by authorized users do not result in a loss of data consistency. We saw a form of integrity constraint with E-R models: key declarations: stipulation that certain attributes form a candidate key for the entity set.

CHECK constraints let you enforce very specific integrity rules by specifying a check condition. The condition of a CHECK constraint has some limitations:

It must be a Boolean expression evaluated using the values in the row being inserted or updated, and

It cannot contain sub queries; sequences; the SQL functions SYSDATE, UID, USER, or USERENV; or the pseudo columns LEVEL or ROWNUM.

UNIQUE constraint: The UNIQUE constraint ensures that all values in a column are different. Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns. A PRIMARY KEY constraint automatically has a UNIQUE constraint. However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

```
CREATE TABLE Persons (
   ID int NOT NULL UNIQUE,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int
);
```

b **Explain Select and Project operation with suitable example.**

**Select Operation (σ)**

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

Notation − σp(r)

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

For example −

σsubject = "database"(Books)

Output − Selects tuples from books where subject is 'database'.

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

Output − Selects tuples from books where subject is 'database' and 'price' is 450.

σsubject = "database" and price = "450" or year > "2010"(Books)

Output − Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

**Project Operation (∏)**

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

Notation − ∏A1, A2, An (r)

Where A1, A2 , An are attribute names of relation r.

Duplicate rows are automatically eliminated, as relation is a set.

For example −

∏subject, author (Books)

**5. a Discuss insertion, deletion and updation anomalies by taking suitable examples.**

Insertion, Updation and Deletion Anomalies are very frequent if Database is not Normalized. To understand these anomalies let us take an example of Student table.

| S_id | S_Name | S_Address | Subject_opted |
|---|---|---|---|
| 401 | Adam | Noida | Bio |
| 402 | Alex | Panipat | Maths |
| 403 | Stuart | Jammu | Maths |
| 404 | Adam | Noida | Physics |

---

Updation Anamoly : To update address of a student who occurs twice or more than twice in a table, we will have to update S_Address column in all the rows, else data will become inconsistent.

Insertion Anamoly : Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert NULL there, leading to Insertion Anamoly.

Deletion Anamoly : If (S_id) 401 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

**b Explain i) Aggregate functions**

Include COUNT, SUM, MAX, MIN, and AVG

Find the maximum salary, the minimum salary, and the average salary among all employees.

```
SELECT      MAX(SALARY),
MIN(SALARY), AVG(SALARY)
        FROM  EMPLOYEE
```

– Some SQL implementations may not allow more than one function in the SELECT-clause

Retrieve the total number of employees in the company (Q17), and the number of employees in the 'Research' department (Q18).

```
SELECT      COUNT (*)
            FROM  EMPLOYEE

SELECT      COUNT (*)
            FROM  EMPLOYEE,
        DEPARTMENT
            WHERE       DNO=DNUMBER AND
        DNAME='Research
```

Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

```
SELECT      MAX(SALARY), MIN(SALARY),
        AVG(SALARY)
        FROM  EMPLOYEE, DEPARTMENT
        WHERE       DNO=DNUMBER AND
        DNAME='Research'
```

**ii) Embedded SQL:**

To illustrate the concept of embedded SQL let's consider C as host programming language.

Within an embedded SQL commands. We may refer to specially declared C rogram variables.

Shared Variables: It is used in both C and SQL statement.

SQLCode and SQL State to communicate errors and exception conditions between the program and the DBMS.

Communicating between the program and the DBMS.

SQL code – declared as int

Value 0 – SQL statement is successfully executed

SQLCODE > 0 – especially SQLCODE = 100 - no more data are available in a query result.

Sqlcode < 1 – error while executing sql statement

// Program Segment E1:

```
Loop=1;
While(loop)
{
Prompt("Enter a social security number:",ssn);
EXEC SQL
Select fname, minit, lname, address,salary into :fname,:mint,:lname, :address, :salary from employee where ssn=:ssn;
If(sqlcode==0)
Printf(fname,minit, lname,address,salary)
Else
Printf("SSN does not exists:",SSN);
Prompt("More SSN(enter 1 for yes, 0 for no):",loop);
}
```

6. **a How is a view created and dropped? What problems are associated with updating of views?**

   A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

   **Create VIEW command**:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

```
CREATE VIEW [Products Above Average Price] AS
SELECT ProductName, UnitPrice
FROM Products
WHERE UnitPrice > (SELECT AVG(UnitPrice) FROM Products);
```

We can **delete a view with the DROP VIEW** command.

```
DROP VIEW view_name;
```

- Consider Q is the query that defines view V based on database D.
- U is the update specification.
- V' is the updated view.
- T is the translator for U, i.e., the actual update applied to D.
- D' is the updated database.

**Problem:** there may be more than one translator T for a given update specification U.

**b Explain SQL data definition and datatype in brief and explain DROP and ALTER command.**

1. Numeric : (Integer/Int , Real , Decimal(i,j),Dec)
2. Character – string (Char(n) , Varchar(n)
   Value assigned is enclosed ' ' , case sensitive
   | | (double vertical bar) : used for concatenation. Eg: 'abc'||'xyz' => abcxyz

- Bit String : Bit(n) / Bit Varying(n)
- Boolean : True / False / Unknown
- Date YYYY-DD-MM
- Time HH:MM:SS
- Timestamp

Alter Command:

- Used to add an attribute to one of the base relations
- The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is *not allowed* for such an attribute
- Example:

**ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12);**

The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple. This can be done using the UPDATE command.

Drop Command:

- Used to remove a relation (base table) *and its definition*
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists
- Example:

  **DROP TABLE DEPENDENT;**
  **DROP TABLE <table name> CASCADE**
  **DROP TABLE <table name> RESTRICT**

**7. a Define functional dependency. Explain 1NF, 2NF and 3NF with example for each.**

Functional dependency is a relationship that exists when one attribute uniquely determines another attribute. If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X->Y, which specifies Y is functionally dependent on X.

Normalization rule are divided into following normal form.

First Normal Form Second Normal Form Third Normal Form

First Normal Form (1NF)

As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

The Primary key is usually a single column, but sometimes more than one column can be combined to create a single primary key. For example consider a table which is not in First normal form

Student Table :

| Student | Age | Subject |
|---|---|---|
| Adam | 15 | Biology, Maths |
| Alex | 14 | Maths |

| | | |
|---|---|---|
| Stuart | 17 | Maths |

In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be :

| Student | Age | Subject |
|---|---|---|
| Adam | 15 | Biology |
| Adam | 15 | Maths |
| Alex | 14 | Maths |
| Stuart | 17 | Maths |

Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

---

Second Normal Form (2NF)

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails Second normal form.

In example of First Normal Form there are two rows for Adam, to include multiple subjects that he has opted for. While this is searchable, and follows First normal form, it is an inefficient use of space. Also in the above Table in First Normal Form, while the candidate key is {Student, Subject}, Age of Student only depends on Student column, which is incorrect as per Second Normal Form. To achieve second normal form, it would be helpful to split out the subjects into an independent table, and match them up using the student names as foreign keys.

| Student | Age |
|---|---|
| Adam | 15 |
| Alex | 14 |
| Stuart | 17 |

New Student Table following 2NF will be :

In Student Table the candidate key will be Student column, because all other column i.e Age is dependent on it.

New Subject Table introduced for 2NF will be :

| Student | Subject |
|---|---|
| Adam | Biology |
| Adam | Maths |
| Alex | Maths |
| Stuart | Maths |

In Subject Table the candidate key will be {Student, Subject} column. Now, both the above tables qualifies for Second Normal Form and will never suffer from Update Anomalies. Although there are a few complex cases in which table in Second Normal Form suffers Update Anomalies, and to handle those scenarios Third Normal Form is there.

Third Normal Form (3NF)

Third Normal form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So this transitive functional dependency should be removed from the table and also the table must be in Second Normal form. For example, consider a table with following fields.

Student_Detail Table :

| Student_id | Student_name | DOB | Street | city | State | Zip |
|---|---|---|---|---|---|---|

In this table Student_id is Primary key, but street, city and state depends upon Zip. The dependency between zip and other fields is called transitive dependency. Hence to apply 3NF, we need to move the street, city and state to new table, with Zip as primary key.

New Student_Detail Table :

| Student_id | Student_name | DOB | Zip |
|---|---|---|---|

Address Table :

| Zip | Street | city | state |
|---|---|---|---|

The advantage of removing transtive dependency is,

Amount of data duplication is reduced.Data integrity achieved.

**b What is a trigger? Explain DML trigger, with an example.**

Triggers are stored programs, which are automatically executed or fired when some events occur.
Triggers are, in fact, written to be executed in response to any of the following events −
A database manipulation (DML) statement (DELETE, INSERT, or UPDATE)
A database definition (DDL) statement (CREATE, ALTER, or DROP).
A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated. To demonstrate triggers we will be using the CUSTOMERS table we had created and used in the previous chapters −
Select * from customers;

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
```

```
| 1 | Ramesh   | 32 | Ahmedabad |  2000.00 |
| 2 | Khilan   | 25 | Delhi     |  1500.00 |
| 3 | kaushik  | 23 | Kota      |  2000.00 |
| 4 | Chaitali |  25 | Mumbai    |  6500.00 |
| 5 | Hardik   | 27 | Bhopal    |  8500.00 |
| 6 | Komal    | 22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

The following program creates a row-level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values −

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  sal_diff := :NEW.salary  - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

When the above code is executed at the SQL prompt, it produces the following result −

Trigger created.

**8. a Explain the Informal guidelines for the relational schema.**

INFORMAL DESIGN GUIDELINES FOR RELATIONAL SCHEMA

1.Semantics of the Attributes

2.Reducing the Redundant Value in Tuples.

3.Reducing Null values in Tuples.

4.Dissallowing spurious Tuples.

1. **Semantics of the Attributes** : Whenever we are going to form relational schema there should be some meaning among the attributes.This meaning is called semantics.This semantics relates one attribute to another with some relation. Eg: USN No Student name Sem

2. **Reducing the Redundant Value in Tuples** Mixing attributes of multiple entities may cause problems Information is stored redundantly wasting storage Problems with update anomalies Insertion anomalies Deletion anomalies Modification anomalies Student name Sem Eg: Dept No Dept Name If we integrate these two and is used as a single table i.e Student Table USN No

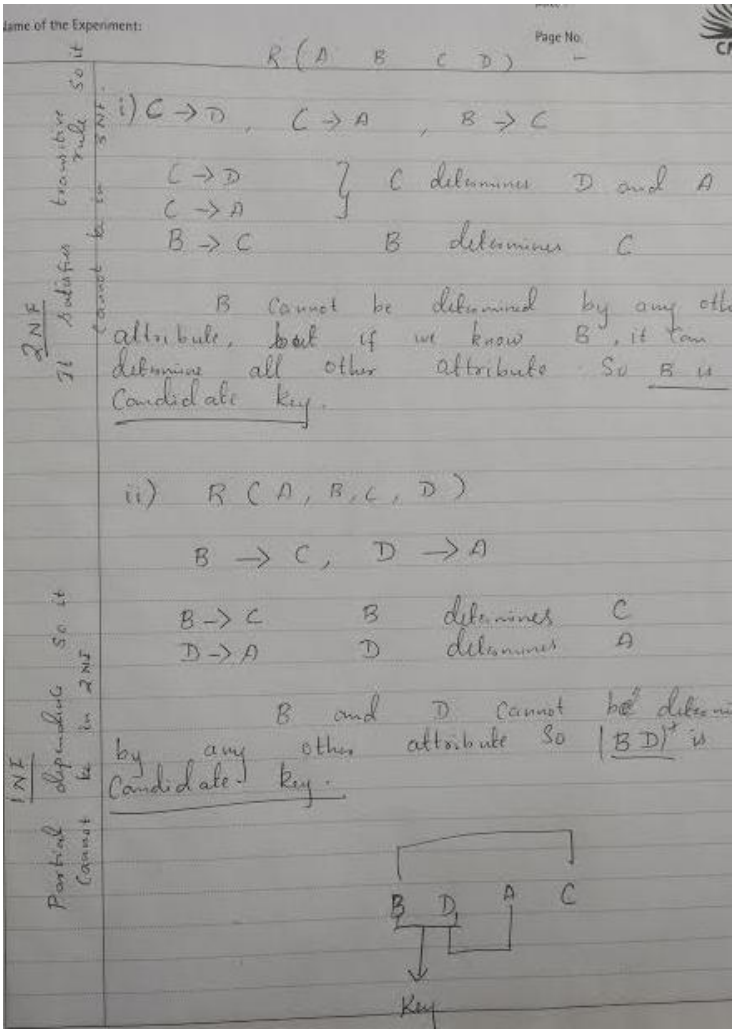
Student name Sem Dept No Dept Name Here whenever if we insert the tuples there may be 'N' stunents in one department,so Dept No,Dept Name values are repeated 'N' times which leads to data redundancy. Another problem is updata anamolies ie if we insert new dept that has no students. If we delet the last student of a dept,then whole information about that department will be deleted If we change the value of one of the attributes of aparticaular table the we must update the tuples of all the students belonging to thet depy else Database will become inconsistent. Note: Design in such a way that no insertion ,deletion,modification anamolies will occur

3. **Reducing Null values in Tuples**. Note: Relations should be designed such that their tuples will have as few NULL values as possible Attributes that are NULL frequently could be placed in separate relations (with the primary key) Reasons for nulls: attribute not applicable or invalid attribute value unknown (may exist) value known to exist, but unavailable

4. Disallowing spurious Tuples Bad designs for a relational database may result in erroneous results for certain JOIN operations The "lossless join" property is used to guarantee meaningful results for join operations Note: The relations should be designed to satisfy the lossless join condition. No spurious tuples should be generated by doing a natural-join of any relations.

**b A relation R has four attributes A B C D. For each of the following sets of FD, identify the candidate key and weight normal form (context not clear).**

**1. C ->D, C->A, B->C 2. B->, D->A.**




**9. a What is transaction? In what ways it is different from an ordinary program?**

To ensure integrity of the data, we require that the database system maintain the following properties of the transactions:

- Atomicity. Either all operations of the transaction are reflected properly in the database or none are.
- Consistency. Execution of a transaction in isolation (that is no other transaction executing concurrently) preserves the consistency of the database.
- Isolation. Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions Ti and Tj it appears to Ti that either Tj finished execution before Ti started, or Tj started execution after Ti finished. Thus, each transaction is unaware of other transactions executing concurrently in the system.
- Durability. After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

These properties are called ACID properties, with acronym derived from the first letters of the above four properties.Even though database permanently resides on disk, some portion of it is temporarily residing in main memory.

Transactions access data using two operations:

Read(x), which transfers the data item x from the database to a local buffer belonging to the transaction that executed the read operation.
Write(x), which transfers the data item x from the local buffer of the transaction that executed the write back to the database.
In a real database system, the write operation does not necessarily result in the immediate update of the data on the disk. The write operation may be temporarily stored in memory and executed on the disk later. For now, assume write is done immediately.

Let Ti be a transaction that transfers $50 from account A to B. this transaction is

```
Ti: read(A);
    A := A – 50;
    Write(A);
    Read(B);
    B := B+50;
    Write (B);
```

**b Explain all the phases involved in ARIES algorithm, with an example.**

ARIES recovery involves three passes

Analysis pass: Determines

- Which transactions to undo
- Which pages were dirty (disk version not up to date) at time of crash
- RedoLSN: LSN from which redo should start

Redo pass:

- Repeats history, redoing all actions from RedoLSN
  - RecLSN and PageLSNs are used to avoid redoing actions already reflected on page

Undo pass:

- Rolls back all incomplete transactions
  - Transactions whose abort was complete earlier are not undone
    - Key idea: no need to undo these transactions: earlier undo actions were logged, and are redone as required

**Analysis pass**

Starts from last complete checkpoint log record

- Reads DirtyPageTable from log record
- Sets RedoLSN = min of RecLSNs of all pages in DirtyPageTable
  - In case no pages are dirty, RedoLSN = checkpoint record's LSN
- Sets undo-list = list of transactions in checkpoint log record
- Reads LSN of last log record for each transaction in undo-list from checkpoint log record

Scans forward from checkpoint

Scans forward from checkpoint

If any log record found for transaction not in undo-list, adds transaction to undo-list

Whenever an update log record is found

- If page is not in DirtyPageTable, it is added with RecLSN set to LSN of the update log record

If transaction end log record found, delete transaction from undo-list

Keeps track of last log record for each transaction in undo-list

- May be needed for later undo

At end of analysis pass:

RedoLSN determines where to start redo pass

RecLSN for each page in DirtyPageTable used to minimize redo work

All transactions in undo-list need to be rolled back

**Redo Pass**: Repeats history by replaying every action not already reflected in the page on disk, as follows:

Scans forward from RedoLSN. Whenever an update log record is found:

1. If the page is not in DirtyPageTable or the LSN of the log record is less than the RecLSN of the page in DirtyPageTable, then skip the log record
2. Otherwise fetch the page from disk. If the PageLSN of the page fetched from disk is less than the LSN of the log record, redo the log record

**Undo pass**:

Performs backward scan on log undoing all transaction in undo-list

Backward scan optimized by skipping unneeded log records as follows:

- Next LSN to be undone for each transaction set to LSN of last log record for transaction found by analysis pass.
- At each step pick largest of these LSNs to undo, skip back to it and undo it
- After undoing a log record
  - For ordinary log records, set next LSN to be undone for transaction to PrevLSN noted in the log record

- For compensation log records (CLRs) set next LSN to be undo to UndoNextLSN noted in the log record
  - » All intervening records are skipped since they would have been undone already

Undos performed as described earlier

**10 a. Explain**

**ACID properties :**

To ensure integrity of the data, we require that the database system maintain the following properties of the transactions:

- Atomicity. Either all operations of the transaction are reflected properly in the database or none are.
- Consistency. Execution of a transaction in isolation (that is no other transaction executing concurrently) preserves the consistency of the database.
- Isolation. Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions Ti and Tj it appears to Ti that either Tj finished execution before Ti started, or Tj started execution after Ti finished. Thus, each transaction is unaware of other transactions executing concurrently in the system.
- Durability. After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

## Strict two phase locking:

Consider the partial schedule of the figure. Each transaction observes the two-phase locking protocol, but the failure of T5 after the read(A) step of T7 leads to cascading rollback of T6 and T7.

| T5: | T6: | T7: |
|---|---|---|
| lock-X(A); | | |
| Read (A); | | |
| lock-S(B); | | |
| Read (B); | | |
| Write(A); | | |
| Unlock(A) | | |
| | lock-X(A); | |
| | Read (A); | |
| | Write(A); | |
| | Unlock(A) | |
| | Lock-S(B); | |
| | | Lock-S(A); |
| | | Read (A); |

Cascading rollbacks can be avoided by a modification of two-phase locking called the strict two-phase locking protocol. This protocol requires not only that locking be two phase, but also that all exclusive-mode locks taken by a transaction be held until that transaction commits. This requirement

ensures that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits, preventing any other transaction form reading the data.

**b Explain the database recovery technique based on different update. (Out of Syllabus)**

**Deferred Update:** Deferred update also called NO-UNDO/REDO is a technique used to recover/support transaction failures that occur due to operating system, power, memory or machine failures. When a transaction runs, any updates or alterations made to the database by the transaction are not done immediately. They are recorded in the log file. Data changes recorded in the log file are applied to the database on commit. This process is called “Re-doing”. On rollback, any changes to data recorded in the log file are discarded; hence no changes will be applied to the database. If a transaction fails and it is not committed due to any of the reasons mentioned above, the records in the log file are discarded and the transaction is restarted. If the changes in a transaction are committed before crashing, then after the system restarts, changes recorded in the log file are applied to the database.

Even though Deferred Update and Immediate Update are two methods for recovering after a system failure, the process that each method uses is different. In differed update method, any changes made to the data by a transaction are first recorded in a log file and applied to the database on commit. In differed update method, records in the log file are discarded on roll back and are never applied to the database. One disadvantage of deferred update method is the increased time taken to recover in case of a system failure.