CMR
INSTITUTE
OF TECHNOLOGY

SCHEME OF EVALUATION

Internal Assessment Test I – Dec. 2023

Sub:	**DATABASE MANAGEMENT SYSTEMS**	**Code:**	21CS53
Date: 21/12/2023 Duration: 90 mins Max Marks: 50	**Sem:** 5	**Branch:**	AIML

Note: ***Answer any five full questions.*** ***(5 X 10 = 50)***

Question No.		**Description**	**Marks Split up**		**Total Marks**
1.		✓ Characteristics ✓ Difference between file system and database approch	5 5	10M	10M
2.	a)	✓ Database Language ✓ Interface	3 3	6M	10M
	b)	✓ Physical data independence ✓ Logical data independence	2 2	4M	
3.	a)	✓ Concept ✓ Types of data model	1 5	6M	10M
	b)	✓ Any four SQL Constraints	4	4M	
4.		✓ Diagram ✓ Explanation of each user	5 5		10M
5.	a)	✓ ER diagram University Database ✓ Enties ✓ Attributes ✓ Primary key ✓ Cardinality	 2 2 1 1	6M	10M
	b)	i. Expression for i ii. Expression for ii	1 1	4M	
6.		✓ Operation of relational algebra with examples			10M

USN

Internal Assessment Test 1 – December 2023

Sub:	**Database Management Systems**				Sub Code:	**21CS53**	Branch:	**AIML**
Date:	**21/12/23**	Duration:	**90 minutes**	Max Marks:	**50**	Sem/Sec:	**V**	**OBE**

		Answer any FIVE FULL Questions	**MARKS**	**CO**	**RBT**
1	a	Discuss the main characteristics of the database approach. How does it differ from traditional file system?	[10]	1	L1
2	a	What are different database schema languages and interfaces?	[6]	1	L1
	b	What is the difference between logical independence and physical data independence?	[4]	1	L1
3	a	Explain the concept of a data model? Explain different categories of data models.	[6]	1	L2
	b	Explain the basic constraints that can be specified in SQL as part of table creation with example (any four).	[4]	2	L2
4	a	With reference to database system environment, describe the component of DBMS and their interaction, with the help of a diagram.	[10]	1	L2
5	a	Write an ER diagram for university data base considering at least four entities.	[6]	1	L3
	b	Consider the relations. EMPLOYEE(emp_id,name) ASSIGNED_TO(projectno,emp_id) PROJECT(projectno,project_name) Express the following queries in Relational Algebra. i) Get details of employee working on both P354 and P345 project numbers. ii)Find the employee number of employee who work on project P678	[4]	2	L3
6	a	List operation of relational algebra and explain the purpose of each with examples.	[10]	2	L1

CI **CCI** **HoD**

---All the Best---

1. Discuss the main characteristics of the database approach. How does it differ from traditional file system?

Characteristics of the Database Approach

Data Independence

One of the fundamental characteristics of database approach in DBMS is data independence. This refers to the ability to modify the structure of a database without impacting the programs that access the data. By separating the logical and physical aspects of the database, changes to the physical structure can be made without affecting the logical structure.

For example, let's consider a database that stores employee information. The logical structure may include fields like employee name, ID, and salary. The physical structure, on the other hand, includes details such as the data's location on disk and the file format. By decoupling these two aspects, database administrators can modify the physical structure, such as moving data to a new disk or changing file formats, without disrupting the programs that interact with the data.

Data Integrity

Another crucial and main characteristics of database approach is data integrity. This ensures the accuracy and consistency of data within the database. Various techniques, such as data validation, data constraints, and data normalization, are employed to achieve data integrity.

Data validation is the process of checking the accuracy and consistency of data entered into the database. For instance, when inputting employee information, a program may verify that the employee ID is unique and that the salary falls within a specific range.

Data constraints are used to enforce rules on the data within the database. An example of a constraint could be ensuring that an employee's salary is greater than zero and less than one million dollars.

Data normalization involves organizing the database to minimize redundancy and enhance consistency. For instance, in a database storing employee and department information, data normalization would involve creating a separate table for department details and establishing a relationship between the employee and department tables.

Data Sharing

The database approach also emphasizes data sharing, enabling multiple users to access and update data simultaneously. Techniques such as locking and concurrency control are employed to facilitate these characteristics of database approach.

Locking involves preventing other users from accessing specific data while it is being updated. For instance, when a user is modifying an employee's salary, the database may lock that employee's record to prevent others from making concurrent changes.

Concurrency control manages access to the database by multiple users. Techniques like time stamping, optimistic concurrency control, and pessimistic concurrency control are employed to ensure data consistency and prevent conflicts.

Backup and Recovery

To safeguard against data loss due to system failures or unexpected events, the database approach incorporates backup and recovery mechanisms. These mechanisms include database backups, transaction logs, and replication.

Database backups involve creating copies of the entire database or specific portions to restore data in case of loss. Regular backup schedules can be established, and backups can be stored on separate servers or in the cloud.

Transaction logs maintain a record of all changes made to the database, such as insertions, updates, and deletions. These logs enable the reconstruction of the database's state at a specific point in time, facilitating data recovery.

Replication involves copying the data from a database to multiple servers, ensuring redundancy in case of data loss. Techniques like master-slave replication and peer-to-peer replication can be employed to enhance data availability and reliability.

Scalability

Scalability is one of the critical characteristics of the database approach, enabling databases to handle vast amounts of data and numerous users without performance degradation. Techniques like horizontal scaling and vertical scaling are utilized to achieve scalability.

Horizontal scaling involves adding more servers to the database system to manage increased workload. Strategies like sharding, which distributes data across multiple servers, and load balancing, which evenly distributes the workload, can be implemented to achieve horizontal scaling.

Vertical scaling involves enhancing the resources, such as memory and CPU power, of a single server to handle increased workload. Hardware upgrades and increasing the number of CPU cores are examples of vertical scaling techniques.

Read our blog on Data Models in DBMS and answer these DBMS Interview Questions.

Security

Security is of paramount importance in the database approach, ensuring that data remains protected from unauthorized access, modification, or deletion. Techniques like authentication, authorization, and encryption are employed to maintain data security.

Authentication verifies the identity of users attempting to access the database. Methods such as username and password authentication or biometric authentication can be utilized.

Authorization controls access to specific database resources based on user roles or permissions. Techniques like access control lists or role-based access control can be employed to enforce authorization policies.

Encryption converts data into a coded format that can only be deciphered by authorized users. Techniques like symmetric key encryption or asymmetric key encryption can be used to encrypt sensitive data

Difference between File System and DBMS

FILE SYSTEM	**DBMS**
Used to manage and organise the files stored in the hard disk of the computer	A software to store and retrieve the user's data
Redundant data is present	No presence of redundant data
Query processing is not so efficient	Query processing is efficient
Data consistency is low	Due to the process of normalisation, the data consistency is high
Less complex, does not support complicated transactions	More complexity in managing the data, easier to implement complicated transactions
Less security	Supports more security mechanisms
Less expensive in comparison to DBMS	Higher cost than the File system
Does not support crash recovery	Crash recovery mechanism is highly supported

2(a) **What are different database schema languages and interfaces?**

Database Language

o A DBMS has appropriate languages and interfaces to express database queries and updates. o Database languages can be used to read, store and update the data in the database.

Types of Database Language:

DDL
DML
DCL
TCL

1.Data Definition Language

o **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern. o It is used to create schema, tables, indexes, constraints, etc. in the database.
o Using the DDL statements, you can create the skeleton of the database.

o Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

2. Data Manipulation Language

DML stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

o **Select:** It is used to retrieve data from a database.
o **Insert:** It is used to insert data into a table.
o **Update:** It is used to update existing data within a table.
o **Delete:** It is used to delete all records from a table.

3. Data Control Language

o **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data. o The DCL execution is transactional. It also has rollback parameters.

(But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

o **Grant:** It is used to give user access privileges to a database.
o **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and

SELECT. 4. Transaction Control Language

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

o **Commit:** It is used to save the transaction on the database.

o **Rollback:** It is used to restore the database to original since the last

Commit. Interfaces in DBMS

A database management system (DBMS) interface is a user interface which allows for the ability to input queries to a database without using the query language itself.

User-friendly interfaces provide by DBMS may include the following:

1. **Menu-Based Interfaces for Web Clients or Browsing –**These interfaces present the user with lists of options (called menus) that lead the user through the formation of a request. Basic advantage of using menus is that they removes the tension of remembering specific commands and syntax of any query language, rather than query is basically composed step by step by collecting or picking options from a menu that is basically shown by the system. Pull-down menus are a very popular technique in *Web based interfaces*. They are also often used in *browsing interface* which allow a user to look through the contents of a database in an exploratory and unstructured manner.
2. **Forms-Based Interfaces –**A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert a new data, or they can fill out only certain entries, in which case the DBMS will redeem same type of data for other remaining entries. This type of forms are usually designed or created and programmed for the users that have no expertise in operating system. Many DBMSs have *forms specification languages* which are special languages that help specify such forms.

   Example: SQL* Forms is a form-based language that specifies queries using a form designed in conjunction with the relational database schema.b>
3. **Graphical User Interface –**A GUI typically displays a schema to the user in diagrammatic form.The user then can specify a query by manipulating the diagram. In many cases, GUI's utilize both menus and forms. Most GUIs use a pointing device such as mouse, to pick certain part of the displayed schema diagram.
4. **Natural language Interfaces –**These interfaces accept request written in English or some other language and attempt to understand them. A Natural language interface has its own schema, which is similar to the database conceptual schema as well as a dictionary of important words. The natural language interface refers to the words in its schema as well as to the set of standard words in a dictionary to interpret the request.If the interpretation is successful, the interface generates a high-level query corresponding to the natural language and submits it to the DBMS for processing, otherwise a dialogue is started with the user to clarify any provided condition or request. The main disadvantage with this is that the capabilities of this type of interfaces are not that much advance.

5. **Speech Input and Output –**There is an limited use of speech say it for a query or an answer to a question or being a result of a request it is becoming commonplace Applications with limited vocabularies such as inquiries for telephone directory, flight arrival/departure, and bank account information are allowed speech for input and output to enable ordinary folks to access this information.

   The Speech input is detected using a predefined words and used to set up the parameters that are supplied to the queries. For output, a similar conversion from text or numbers into speech take place.

6. **Interfaces for DBA –**Most database system contains privileged commands that can be used only by the DBA's staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, reorganizing the storage structures of a databases.

2(b) What is the difference between logical independence and physical data independence?

There are two kinds:

1. Physical Data Independence
2. Logical Data Independence

Physical Data Independence:

The ability to modify the physical schema without causing application programs to be rewritten.

Logical Data Independence:

The ability to modify the conceptual schema without causing application programs to be rewritten

Usually done when logical structure of database is altered

Logical data independence is harder to achieve as the application programs are usually heavily dependent on the logical structure of the data.

Modifications at this level are usually to improve performance.

3(a) Explain the concept of a data model? Explain different categories of data models.

he Data Model gives us an idea of how the final system would look after it has been fully implemented. It specifies the data items as well as the relationships between them. In a database management system, data models are often used to show how data is connected, stored, accessed, and changed. We portray the information using a set of symbols and language so that members of an organisation may understand and comprehend it and then communicate.

Types of Data Models in DBMS

Though there are other data models in use today, the Relational model is the most used. Aside from the relational model, there are a variety of different data models that we shall discuss in-depth in this article. Data Models in DBMS include the following:

Hierarchical Model

This concept uses a hierarchical tree structure to organise the data. The hierarchy begins at the root, which contains root data, and then grows into a tree as child nodes are added to the parent node.

Network Model

The main difference between this model and the hierarchical model is that any record can have several parents in the network model. It uses a graph instead of a hierarchical tree.

Entity-Relationship Model

The real-world problem is depicted in visual form in this model to make it easier for stakeholders to comprehend. The ER diagram also makes it very simple for developers to comprehend the system.

Relational Model

The data in this model is kept in the form of a table that is two-dimensional. All of the data is kept in the form of rows and columns. Tables are the foundation of a relational paradigm.

Object-Oriented Data Model

Both the data and the relationship are contained in a single structure that is known as an object in this model. We can now store audio, video, pictures, and other types of data in databases, which was previously impossible with the relational approach (Although you can store video and audio in relational DB, it is advised not to store them in the relational database).

Object-Relational Data Model

It is a hybrid of relational and object-oriented models. This model was developed to bridge the gap between the object-oriented and relational models.

Flat Data Model

It’s a straightforward model in which the DB is depicted as a table with rows and columns.

Semi-Structured Data Model

The relational model has evolved into the semi-structured model. In this model, we can’t tell the difference between data and schema.

Associative Data Model

It is a model in which the data is separated into two sections. Everything that has its own existence is referred to as an entity, and the relationships between these entities are referred to as associations. Items and links are two types of data that are separated into two components.

Context Data Model

The Context Data Model is made up of various models. This includes models such as network models and relational models, among others.

3(b). Explain the basic constraints that can be specified in SQL as part of table creation with example.

Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

The available constraints in SQL are:

- **NOT NULL**: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
- **UNIQUE**: This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.
- **PRIMARY KEY**: A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.
- **FOREIGN KEY**: A Foreign key is a field which can uniquely identify each row in a another table. And this constraint is used to specify a field as Foreign key.
- **CHECK**: This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.
- **DEFAULT**: This constraint specifies a default value for the column when no value is specified by the user.

How to specify constraints?
We can specify constraints at the time of creating the table using CREATE TABLE statement. We can also specify the constraints after creating a table using ALTER TABLE statement.
Syntax:
Below is the syntax to create constraints using CREATE TABLE statement at the time of creating the table.

```
CREATE TABLE sample_table

(

column1 data_type(size) constraint_name,

column2 data_type(size) constraint_name,

column3 data_type(size) constraint_name,

....

);
```

sample_table: Name of the table to be created.
data_type: Type of data that can be stored in the field.
constraint_name: Name of the constraint. for example- NOT NULL, UNIQUE, PRIMARY KEY etc.
Let us see each of the constraint in detail.

1. NOT NULL –
If we specify a field in a table to be NOT NULL. Then the field will never accept null value. That is, you will be not allowed to insert a new row in the table without specifying any value to this field.
For example, the below query creates a table Student with the fields ID and NAME as NOT NULL. That is, we are bound to specify values for these two fields every time we wish to insert a new row.

```
CREATE TABLE Student

(

ID int(6) NOT NULL,
NAME varchar(10) NOT NULL,
ADDRESS varchar(20)
);
```

2. UNIQUE –
This constraint helps to uniquely identify each row in the table. i.e. for a particular column, all the rows should have unique values. We can have more than one UNIQUE columns in a table.
For example, the below query creates a table Student where the field ID is specified as UNIQUE. i.e, no two students can have the same ID.

```
CREATE TABLE Student
(
ID int(6) NOT NULL UNIQUE,
NAME varchar(10),
ADDRESS varchar(20)
);
```

3. PRIMARY KEY –
Primary Key is a field which uniquely identifies each row in the table. If a field in a table as primary key, then the field will not be able to contain NULL values as well as all the rows should have unique values for this field. So, in other words we can say that this is combination of NOT NULL and UNIQUE constraints.
A table can have only one field as primary key. Below query will create a table named Student and specifies the field ID as primary key.

```
CREATE TABLE Student
(
ID int(6) NOT NULL UNIQUE,
NAME varchar(10),
ADDRESS varchar(20),
PRIMARY KEY(ID)
);
```

4. FOREIGN KEY –
Foreign Key is a field in a table which uniquely identifies each row of a another table. That is, this field points to primary key of another table. This usually creates a kind of link between the tables.
Consider the two tables as shown below:

Orders

O_ID	**ORDER_NO**	**C_ID**
1	**2253**	**3**
2	**3325**	**3**
3	**4521**	**2**
4	**8532**	**1**

Customers

C_ID	**NAME**	**ADDRESS**
1	**RAMESH**	**DELHI**
2	**SURESH**	**NOIDA**
3	**DHARMESH**	**GURGAON**

As we can see clearly that the field C_ID in Orders table is the primary key in Customers table, i.e. it uniquely identifies each row in the Customers table. Therefore, it is a Foreign Key in Orders table.
Syntax:

```
CREATE TABLE Orders

(

O_ID int NOT NULL,

ORDER_NO int NOT NULL,

C_ID int,

PRIMARY KEY (O_ID),

FOREIGN KEY (C_ID) REFERENCES Customers(C_ID)

)
```

(i) CHECK –
Using the CHECK constraint we can specify a condition for a field, which should be satisfied at the time of entering values for this field.
For example, the below query creates a table Student and specifies the condition for the field AGE as (AGE >= 18). That is, the user will not be allowed to enter any record in the table with AGE < 18. Check constraint in detail

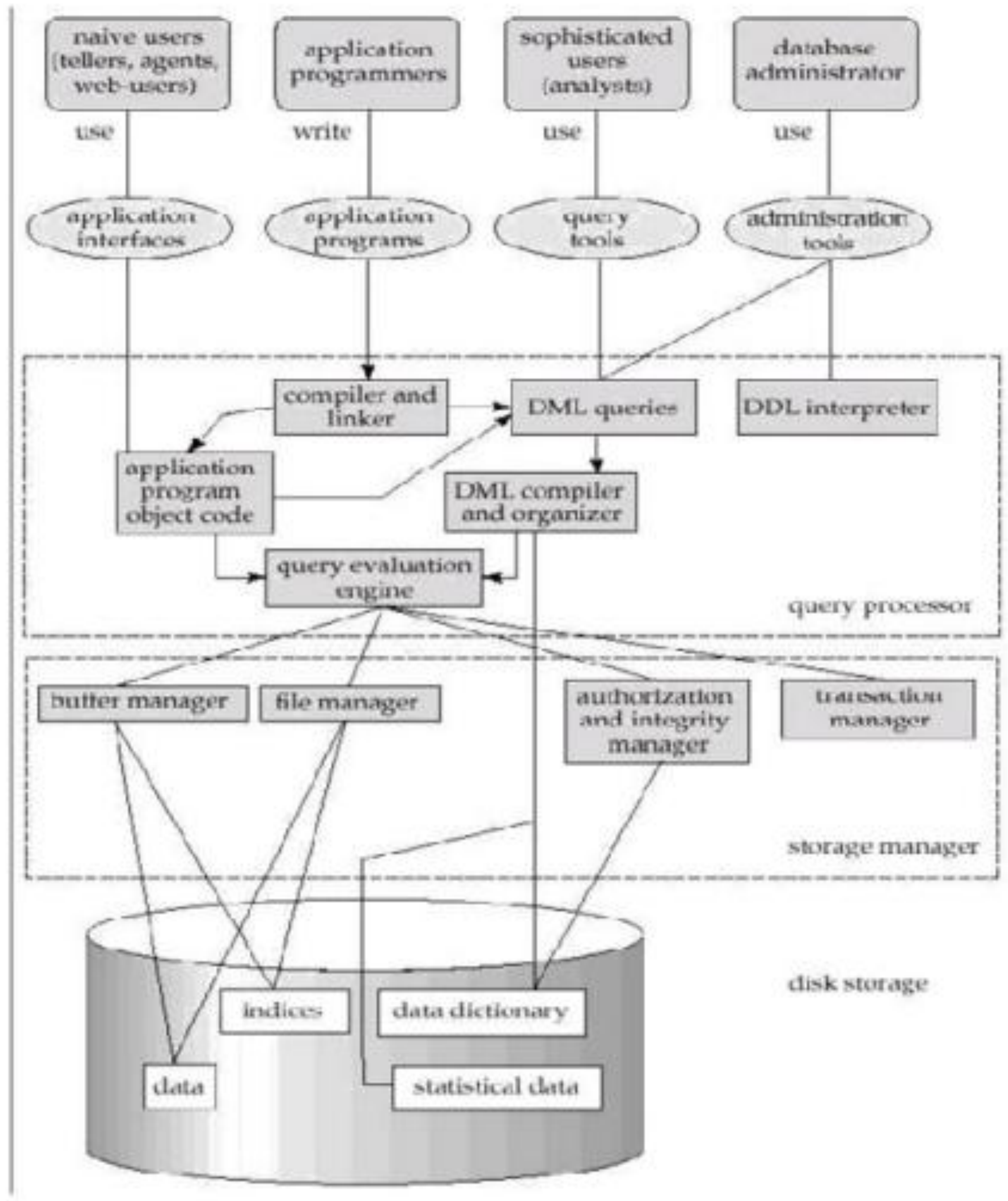
```
CREATE TABLE Student

(

ID int(6) NOT NULL,

NAME varchar(10) NOT NULL,

AGE int NOT NULL CHECK (AGE >= 18)

);
```

(ii) DEFAULT –
This constraint is used to provide a default value for the fields. That is, if at the time of entering new records in the table if the user does not specify any value for these fields then the default value will be assigned to them.
For example, the below query will create a table named Student and specify the default value for the field AGE as 18.

```
CREATE TABLE Student

(

ID int(6) NOT NULL,

NAME varchar(10) NOT NULL,

AGE int DEFAULT 18

);
```

4. With reference to database system environment, describe the component of DBMS and their interaction, with the help of a diagram.

Answer: Diagram



A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.

The storage manager is important because databases typically require a large amount of storage space. Some Big organizations Database ranges from Giga bytes to Tera bytes. So the main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed.

The query processor also very important because it helps the database system simplify and facilitate access to data. So quick processing of updates and queries is important. It is the job of the database system to translate updates and queries written in a nonprocedural language.

StorageManager:

A storage manager is a program module that provides the interface between the low level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The storage manager translates the various DML statements into low-level file-system commands. Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

Storage Manager Components:

Authorization and integrity manager which tests for the satisfaction of integrity constraints and checks the authority of users to access data.

Transaction manager which ensures that the database itself remains in a consistent state despite system failures, and that concurrent transaction executions proceed without conflicting.

File manager: which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

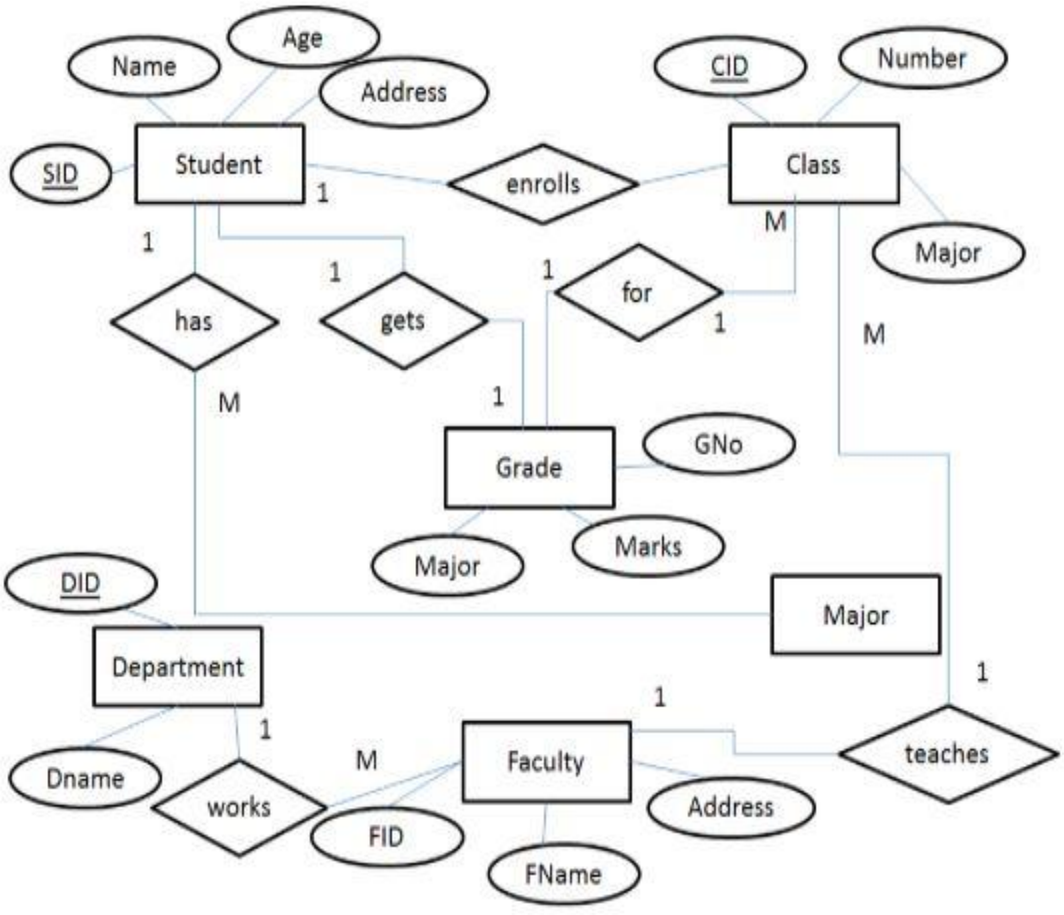
Buffer manager which is responsible for fetching data from disk storage into main memory. Storage manager implements several data structures as part of the physical system implementation. Data files are used to store the database itself. Data dictionary is used to stores metadata about the structure of the database, in particular the schema of the database.

Query Processor Components:

DDL interpreter: It interprets DDL statements and records the definitions in the data dictionary.

DML compiler: It translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands. **Query evaluation engine:** It executes low-level instructions generated by the DML compile.

5(a) Write an ER diagram for university data base considering at least four entities.



5(b) Consider the relations.
EMPLOYEE(emp_id,name)
ASSIGNED_TO(projectno,emp_id)
PROJECT(projectno,project_name)
Express the following queries in Relational Algebra.
i)Get details of employee working on both P354 and P345 project numbers.
Answer:
Π emp_id,name(∂project_no.=P354 (EMPLOYEE*ASSIGNED_TO)) ∩ Π emp_id,name (∂project_no.=P345 (EMPLOYEE*ASSIGNED_TO))

ii)Find the employee number of employee who do not work on project P678
Answer:
Π emp_id(∂project_no.<>P678 (EMPLOYEE*ASSIGNED_TO))

6. List operation of relational algebra and explain the purpose of each with examples.
Relational Algebra is a procedural query language. Relational algebra mainly provides a theoretical foundation for relational databases and SQL. The main purpose of using Relational Algebra is to define operators that transform one or more input relations into an output relation. Given that these operators accept relations as input and produce relations as output, they can be combined and used to express potentially complex queries that transform potentially many input relations (whose data are stored in the database) into a single output relation (the query results).
Fundamental Operators
These are the basic/fundamental operators used in Relational Algebra.
1. Selection(σ)
2. Projection(π)

3. Union(U)
4. Set Difference(-)
5. Set Intersection(∩)
6. Rename(ρ)
7. Cartesian Product(X)

1. Selection(σ): It is used to select required tuples of the relations.
Example:

A	B	C
1	2	4
2	2	3
3	2	3
4	3	4

For the above relation, **σ(c>3)R** will select the tuples which have c more than 3.

A	B	C
1	2	4
4	3	4

Note: The selection operator only selects the required tuples but does not display them. For display, the data projection operator is used.
2. Projection(π): It is used to project required column data from a relation.
Example: Consider Table 1. Suppose we want columns B and C from Relation R.
π(B,C)R will show following columns.

B	C
2	4
2	3
3	4

Note: By Default, projection removes duplicate data.
3. Union(U): Union operation in relational algebra is the same as union operation in set theory.
Example:

FRENCH

Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13

Student_Name	**Roll_Number**
Geeta	17

GERMAN

Student_Name	**Roll_Number**
Vivek	13
Geeta	17
Shyam	21
Rohan	25

Consider the following table of Students having different optional subjects in their course.

π(Student_Name)FRENCH U π(Student_Name)GERMAN

Student_Name
Ram
Mohan
Vivek
Geeta
Shyam
Rohan

Note: The only constraint in the union of two relations is that both relations must have the same set of Attributes.

4. Set Difference(-): Set Difference in relational algebra is the same set difference operation as in set theory.

Example: From the above table of FRENCH and GERMAN, Set Difference is used as follows

π(Student_Name)FRENCH - π(Student_Name)GERMAN

Student_Name
Ram
Mohan

Note: The only constraint in the Set Difference between two relations is that both relations must have the same set of Attributes.

5. Set Intersection(∩): Set Intersection in relational algebra is the same set intersection operation in set theory.

Example: From the above table of FRENCH and GERMAN, the Set Intersection is used as follows π(Student_Name)FRENCH ∩ π(Student_Name)GERMAN

Student_Name
Vivek
Geeta

Note: The only constraint in the Set Difference between two relations is that both relations must have the same set of Attributes.
6. Rename(ρ): Rename is a unary operation used for renaming attributes of a relation.
ρ(a/b)R will rename the attribute 'b' of the relation by 'a'.
7. Cross Product(X): Cross-product between two relations. Let’s say A and B, so the cross product between A X B will result in all the attributes of A followed by each attribute of B. Each record of A will pair with every record of B.
Example:

A

Name	**Age**	**Sex**
Ram	14	M
Sona	15	F
Kim	20	M

B

ID	**Course**
1	DS
2	DBMS

A X B

Name	**Age**	**Sex**	**ID**	**Course**
Ram	14	M	1	DS
Ram	14	M	2	DBMS
Sona	15	F	1	DS
Sona	15	F	2	DBMS
Kim	20	M	1	DS
Kim	20	M	2	DBMS

Note: If A has ‘n’ tuples and B has ‘m’ tuples then A X B will have ‘ n*m ‘ tuples.
Derived Operators
These are some of the derived operators, which are derived from the fundamental operators.

1. Natural Join(⋈)
2. Conditional Join

1. Natural Join(⋈): Natural join is a binary operator. Natural join between two or more relations will result in a set of all combinations of tuples where they have an equal common attribute.

Example:

EMP

Name	**ID**	**Dept_Name**
A	120	IT
B	125	HR
C	110	Sales
D	111	IT

DEPT

Dept_Name	**Manager**
Sales	Y
Production	Z
IT	A

Natural join between EMP and DEPT with condition :

EMP.Dept_Name = DEPT.Dept_Name

EMP ⋈ DEPT

Name	**ID**	**Dept_Name**	**Manager**
A	120	IT	A
C	110	Sales	Y
D	111	IT	A

2. Conditional Join: Conditional join works similarly to natural join. In natural join, by default condition is equal between common attributes while in conditional join we can specify any condition such as greater than, less than, or not equal.

Example:

R

ID	**Sex**	**Marks**
1	F	45
2	F	55
3	F	60

S

ID	**Sex**	**Marks**
10	M	20
11	M	22
12	M	59

Join between R and S with condition **R.marks >= S.marks**

R.ID	**R.Sex**	**R.Marks**	**S.ID**	**S.Sex**	**S.Marks**
1	F	45	10	M	20
1	F	45	11	M	22
2	F	55	10	M	20
2	F	55	11	M	22
3	F	60	10	M	20
3	F	60	11	M	22
3	F	60	12	M	59