Rohit Chachra
UE228087
BE-IT Sec-2 3rd Sem

DATE
PAGE No.

DBMS

1(a) Compare and contrast various data models.

⇒ Comparison of some common data models :

- **Relational Data Model** –
  - > Structure : Organizes data into tables with rows & columns.
  - > Relationship : Defines relationship between tables using keys.
  - > Flexibility : Provides a structured and rigid format.
  - > Example : SQL databases
- **Entity - Relationship Model (ER Model)** –
  - > Structure : Represents entities, attributes & relation between entities
  - > Relationship : Defines relationship using entities & attributes
  - > Flexibility : Good for representing complex relationships.
  - > Example : Often used in the design phase before implementing a relational database.
- **Object - Oriented Data Model** –
  - > Structure : Represent data as objects, encapsulating attributes & behaviours
  - > Relationships : Defines relation between objects.
  - > Flexibility : Supports inheritance and polymorphism
  - > Example : Object oriented databases (OODBMS)
- **Semi - Structured Data Model** –
  - > Structure : Does not adhere to a rigid schema, data with varying structures
  - > Flexibility : Flexible representation of data with irregularities/variations
  - > Example : JSON, XML (Extensible Markup language)
- **Hierarchical Data Model** –
  - > Structure : Organizes data in a tree-like structure
  - > Relationships : Parent-child relationships between data elements.
  - > Flexibility : Suitable for representing hierarchical data.
  - > Example : XML
- **Network Data Model** –
  - > Structure : Representing data as records and sets in form of directed graphs
  - > Relationship : Complex relationships through pointers
  - > Flexibility : More flexible than hierarchical model
  - > Example : Integrated Data Store (IDS)

1(b) Demonstrate ~~data~~ implementation of data abstraction in DBMS.

⇒ DBMS provides several levels of data abstractions, to simplify users' interaction with the system :

- Physical Level - The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures.
- Logical Level - The next higher level of abstraction describes what data are stored in the database, and what relationship exist among those data. The logical level thus describes the entire database in terms of small number of relatively simpler structures. Database administrators use the logical level
- View level - The highest level describes only part of the entire database. Even though logical level uses simpler structures complexity remains because of the variety of info stored in a large database. The view level of abstraction exists to simplify users interaction with system by allowing them to access the desired part of the database. The system may provide many views for same database

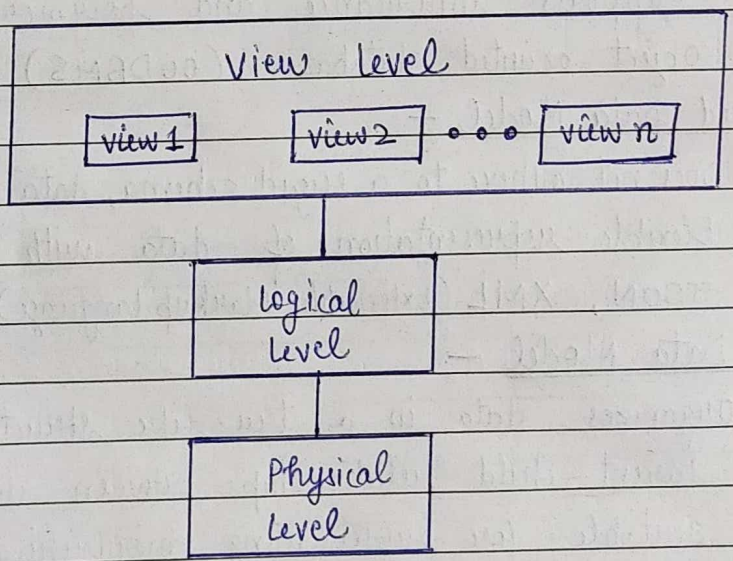
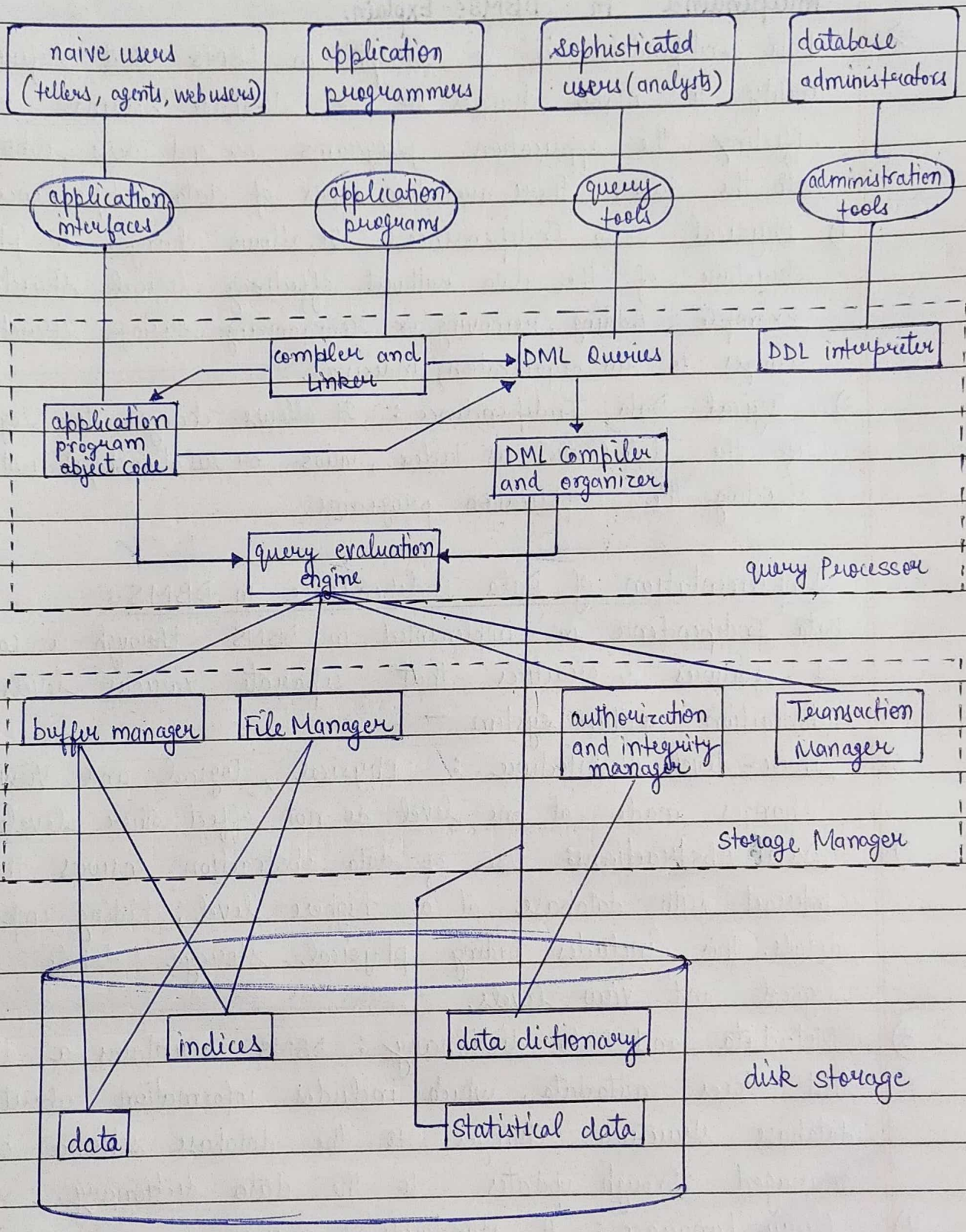


Fig: Three levels of data abstraction

1(c) Explain the architecture of DBMS with a neat sketch

⇒



DBMS ARCHITECTURE

Page :
Date: / /

1(d) Define data independence. How do you implement data independence in DBMS? Explain.

⇒ Data Independence is a concept in DBMS that refers to the ability to make changes to the database schema without affecting the application programs or end users who interact with the data. There are two types of data independence -

1) Physical Data Independence: It allows changes to physical storage structure of the data without affecting logical structure. Example - Adding, removing or reorganizing storage structures (e.g. changes to file organization, indexing).

2) Logical Data Independence: It allows changes to logical structure of the data (such as tables, views or relationships) without affecting the application programs.

Implementation of Data Independence in DBMS:

Data Independence in implemented in DBMS through a combination of features & practices that separate various levels of abstraction within system -

1) Three-Level Architecture: Physical, logical and View level. Changes made at one level do not affect other levels.

2) Data Abstraction: Use of data abstraction ensures that users interact with database at a higher level, hiding implementation details. This includes hiding physical storage details from the logical and view levels.

3) Metadata and Data dictionary: DBMS maintains a data dictionary that stores metadata, which includes information about the database structure. Changes to the database schema can be managed through updates to the data dictionary.

4) Query Language: A standardized query language (SQL) allows users to interact with the database using a high-level, declarative syntax. Changes to database schema are made using data definition language (DDL) statements managed by the DBMS.

DATE : / /20
PAGE NO

2 (a) How do you represent cardinalities, roles, weak entities and weak relations in E/R diagram. Explain

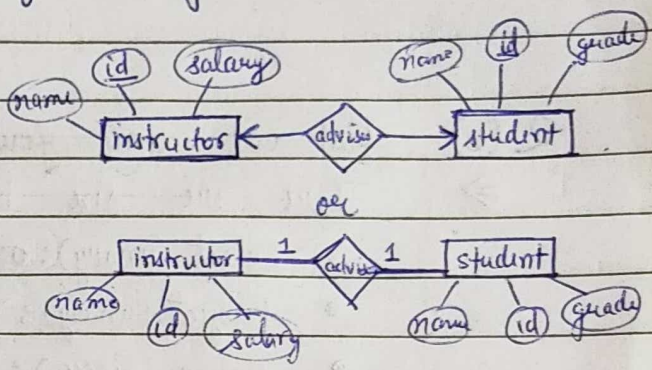
⇒ • **Cardinalities**

Draw lines between the relationship and entity as follows-

- directed line (→), signifying "one"
- undirected line (—), signifying "many"

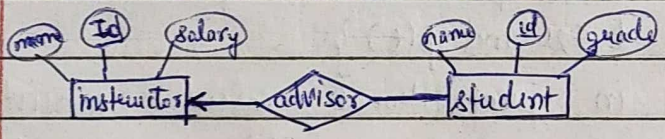
One-to-one Relationship

e.g. student is associated with at most one instructor & instructor is associated with at most one student via advisor



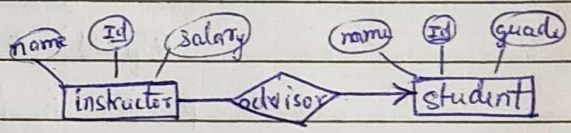
one-to many Relationship

e.g student associated with at most one instructor & instructor associated with several students via advisor



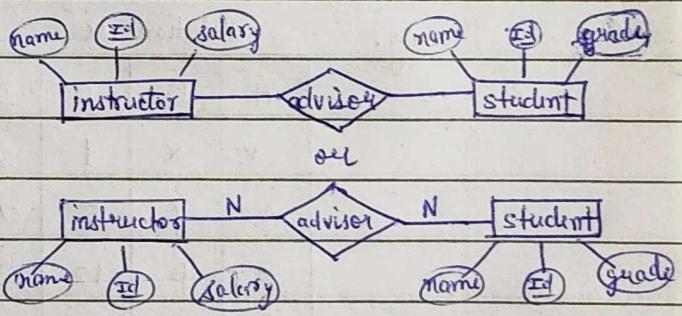
Many-to-one Relationship

e.g student associated with several instructors & instructor associated with at most one student via advisor



Many-to-Many Relationship

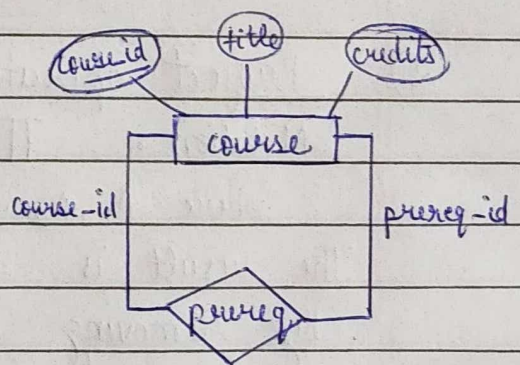
e.g student associated with several instructors & instructor associated with several students via advisor



• **Roles**

We indicate roles in ER diagram by labeling the lines that connect diamond and rectangles.

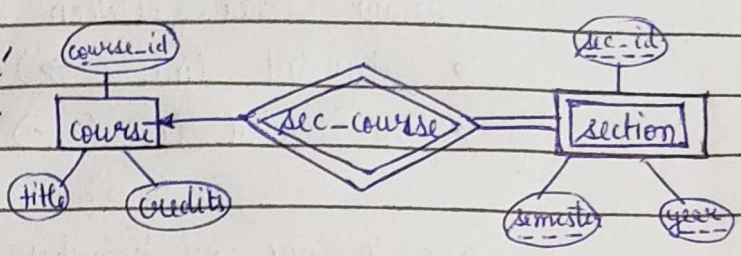
e.g course-id & prereq-id indicates roles between course entity set & prereq relationship set.



DATE : / 20
PAGE No

- **Weak Entity and Weak Relations**

Weak entity sets are indicated by double rectangles
Weak relationship sets are indicated by double diamonds
e.g. 'section' is a weak entity set and 'sec_course' is a weak relationship set



2 (b) List out any four operations in relational algebra. Explain

⇒ There are six basic operations in Relational Algebra :

- Select (unary) : σ
- Project (unary) : Π
- Union (binary) : $\cup$
- Set Difference (binary) : $-$
- Cartesian Product (binary) : $\times$
- Rename (unary) : ρ

1) **Select operation**

Selects tuples that satisfy a given predicate
Notation : $\sigma_p(r)$ where p is selection predicate

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

where p is a formula in propositional calculus consisting of terms connected by $\wedge$ (and), $\vee$ (or), $\neg$ (not) and $=, \neq, >, \geq, <, \leq$

eg.

Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{A=B \wedge D>5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

2) **Project operation**

Notation : $\Pi_{A_1, A_2, \ldots A_k}(r)$

where $A_1, A_2 \ldots A_k$ are attribute names & r is a relation name. The result is defined as the relation of k columns obtained by removing columns that are not listed. Duplicate rows are removed from result.

DATE : / /20
PAGE NO

e.g. Relation r

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

3) Union operation

Notation : $r \cup s$ where r & s are relation name

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

For $r \cup s$ to be valid,

- r, s must have the same arity (same no. of attributes)
- The attribute domain must be compatible (domains of the i^{th} attribute of r and i^{th} attribute of s must be same for all i.).

e.g. Relation r

A	B
α	1
α	2
β	1

Relation s

A	B
α	2
β	3

$r \cup s$ =

A	B
α	1
α	2
β	1
β	3

4) Set Difference operation

Notation : $r - s$

$$r - s = \{ t \mid t \in r \text{ and } t \notin s \}$$

Set difference operation is valid for compatible relations, i.e

- r and s must have same arity
- attribute domains of r and s must be compatible

e.g Relation r

A	B
α	1
α	2
β	1

Relation s

A	B
α	2
β	3

$r - s$ =

A	B
α	1
β	1

DATE: / 20
PAGE No

3 (a) Explain any four SQL Aggregate operators with an example.

⇒ Aggregate operators in SQL:

Employees

e-Id	Salary
E1	10,000
E2	20,000
E3	8,000
E4	50,000
E5	35,000
E6	27,000

1. COUNT:

Counts the number of rows in a result set.

Example: SELECT COUNT(*) AS TotalEmployees
FROM Employees.

output: Total Employees
6

2. SUM:

Calculates the sum of values in a numeric column

Example: SELECT SUM(Salary) AS TotalSalary
FROM Employees

output: TotalSalary
150000

3. AVG:

Calculates the average of values in a num. column

Example: SELECT AVG(Salary) AS AverageSalary
FROM Employees

output: AverageSalary
25000

4. MAX:

Retrieves the Maximum value in a column

Example: SELECT MAX(Salary) AS MaxSalary
FROM Employees

output: MaxSalary
50000

5. MIN:

Retrieves the minimum value in a column

Example: SELECT MIN(Salary) AS MinSalary FROM Employees

output: MinSalary
8000

Date / /
Page No.

3. (b) Explain the operators in SQL with examples
a) ANY b) IN c) EXISTS d) EXCEPT

⇒ Table: Employees

e-id	e-name	salary	Department
1	A	10,000	Marketing
2	B	20,000	Sales
3	C	15,000	Accounts
4	D	30,000	Sales
5	E	50,000	R&D

Table: Projects

e-id	p-id	Budget
2	B	1 Cr
4	D	1 Cr
5	E	50 Lakh

a) ANY operator : The 'ANY' operator returns true if any of the subquery results satisfy the comparison condition.

Example : SELECT e-name FROM Employees
WHERE Salary > ANY (SELECT Salary FROM Employees
WHERE Department = 'Accounts');

output :

e-name
B
D
E

b) IN operator: The 'IN' operator returns true if the value matches any value in the set or result of the subquery.

EXAMple : SELECT e-name FROM Employees
WHERE Department IN ('Marketing', 'Sales');

output :

e-name
A
B
D

c) EXISTS operator : The 'EXISTS' operator ~~its~~ is used to check whether a subquery returns any result. It returns true if the subquery returns one or more rows ; otherwise, it returns False.

Example : SELECT e-name FROM Employees E
WHERE EXISTS (SELECT * FROM Projects P WHERE P.e-id = E.e-id);

Output :

e-name
B
D
E

Date / /
Page No.

d) EXCEPT operator : The 'EXCEPT' operator is used to retrieve distinct rows from the left query that do not appear in the result set of the right query.

Example : SELECT e-id FROM Employees EXCEPT SELECT e-id FROM Projects

output :

e-id
1
3

3(c) Explain nested queries and correlated queries with Examples.

⇒ Nested queries : A nested query is a SQL query embedded within another query. The result of the nested query is used as a condition in the outer query.

Example : SELECT e-id, e-name FROM Employees
WHERE Salary > (SELECT AVG(Salary) FROM Employees);

output :

e-id	e-name
4	D
5	E

Correlated Queries : A correlated query is a type of nested query where the inner query references columns from the outer query.

Example : SELECT Department, e-name FROM Employee E1
WHERE Salary > (SELECT AVG(Salary) FROM Employees E2
WHERE E1.Department = E2.Department)

output :

Department	e-name
Sales	D