Name :- Subina Chhetri
Roll :- 078 BEI 039.

DBMS

Date / /
Page

2080 Bhadra.

1. Define Database and DBMS. How DBMS is differed from file system? Explain 2-tier architecture in DBMS.

→ A database is a structured collection of data that is stored and accessed electronically. Databases are designed to manage large amounts of information by organizing, storing and retrieving data efficiently.

A DBMS is a software that interacts with the users, applications and the database itself to capture and analyze data. It provides a systematic way to create, retrieve, update and manage data.

DBMS is differed from file system in following ways :-

DBMS	File system.
- Minimize data redundancy by using normalization & ensures consistency through constraints.	- Data redundancy occurs frequently & lead to inconsistencies.
- Robust security features including user authentication and authorization.	- Basic security mechanisms, limited integrity constraints.
- Automated backup and recovery mechanisms	- Manual or limited backup and recovery options.

Date / /
Page

2-tier Architecture in DBMS:

- 1-tier Architecture.
→ The database and user interface are integrated into a single system. Users directly access the database.

- 2-Tier Architecture.
→ Consists of a client and a server. The client handles the user interface and application logic, while the server manages the database.

- 3-Tier Architecture.
→ Separates the database, application and presentation layers. The client handles the presentation, the middle tier (application server) handles business logic and the database server manages data storage and retrieval.

- 4-tier Architecture.
→ Extends the 3-tier architecture by adding more layers for specific functions like business logic, presentation and data access, which can be distributed across multiple servers. This provides scalability, manageability and flexing.

Date / /
Page

2. Define Attributes and explain its type with example.

→ Attributes are properties or characteristics of an entity. In a database, an attribute describes the data that can be associated with a particular entity.

Types of Attribute:-

a) Simple and composite attributes.
- A simple (atomic) attribute is one that cannot be broken down into small components.
- Composite attributes can be divided into smaller parts which represent simple attributes with independent meaning.

b) Single valued and multivalued attributes.
- A single valued attribute is the one that holds single value for an entity. Eg:- Date of Birth.
- A multi-valued attribute is the one that ~~may~~ have more than one value for a given instances. Eg: Phone number.

c) Derived attributes.
- A derived attribute is one whose value can be calculated from related attributes value.
- Eg: 'Age' can be derived from 'Date of Birth'.

d) Descriptive Attribute
- A relationship may also have attributes called descriptive attributes.

e) Key attribute : unique identifier for an entity.

Date / /
Page

3. ER diagram for the university database.

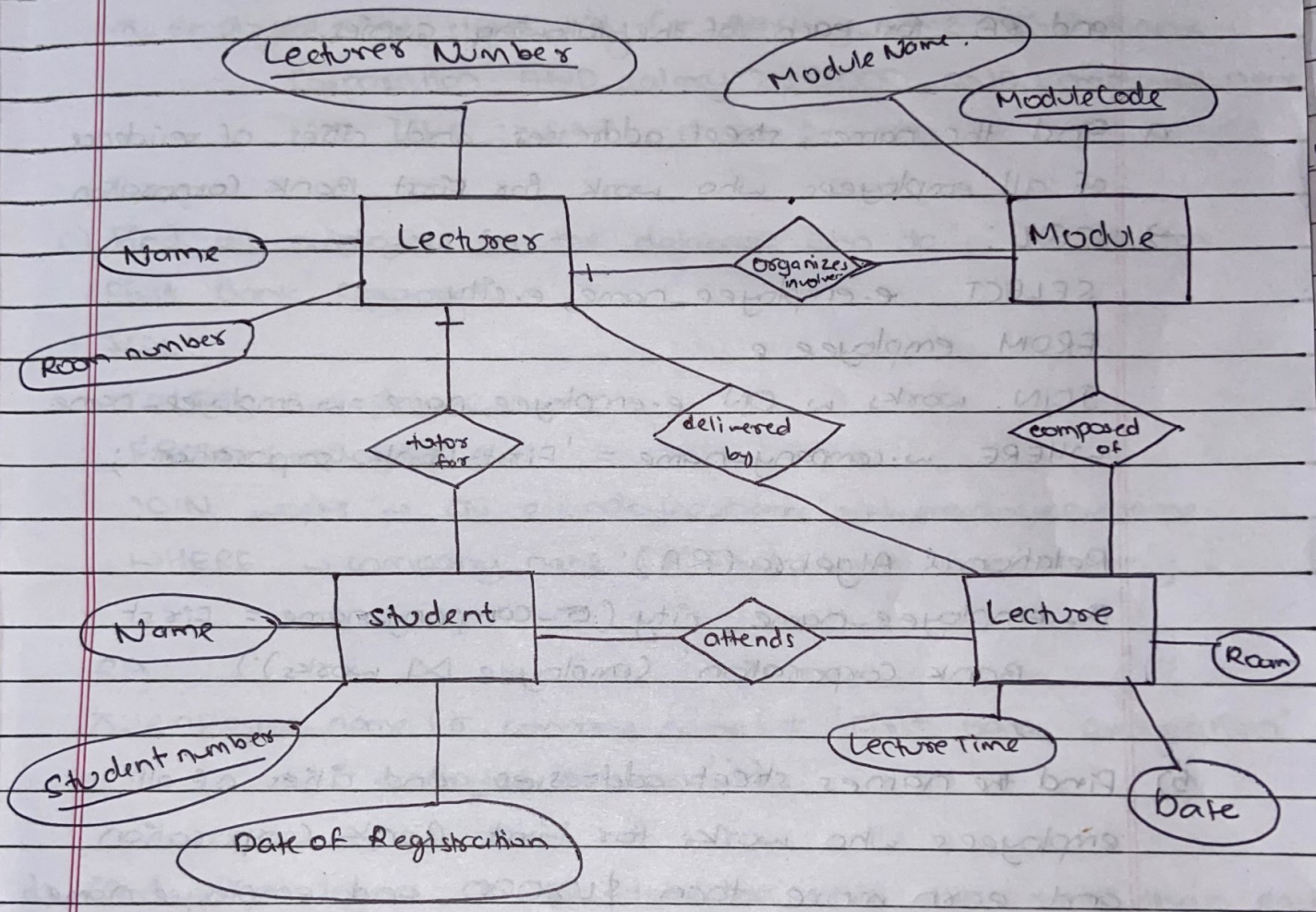
→ List of entities and their attributes are:

Entities	Attributes
• Lecturer	- Lecturer Number (Primary key)
	- Name
	- Room number
• Module	- Module Code (Primary key)
	- Module Name.
• Lecture	- Lecture Time
	- Room
	- Date.
• Student	- StudentNumber (Primary key)
	- Name
	- Date of Registration (for the module)

→ Relationships:

- Lecturer organizes Module :- one-to-many relationship between lecturer and Module.
- Module involves Lecturers :- many to many
- Module composed of Lectures :- one to many
- Lecture delivered by lecturer - one to many
- Students attends Lectures :- many to many
- student registered for Modules :- many to many
- Lecturer acts as Tutor for students :- one to many relationship.

ER diagram:



4). Consider the employee database Schema

employee (employee name, street city)

works (employee name, company name, salary)

company (company name, city)

manages (employee name, manager name), where the

Date / /
Page

primary keys are underlined. Give an expression in SQL and RA for each of the following queries.

a. Find the names, street addresses and cities of residence of all employees who work for First Bank Corporation

→ SQL:

```
SELECT e.employee_name, e.city
FROM employee e
JOIN works w ON e.employee_name = w.employee_name
WHERE w.company_name = 'First Bank Corporation';
```

Relational Algebra (RA):

$\Pi_{employee\_name, city}(\sigma_{company\_name = 'First\ Bank\ Corporation'}(employee \bowtie works))$

b) Find the names, street addresses and cities of all employees who works for First Bank Corporation and earn more than $10,000 and employee name must not start with the letter "z".

→ SQL:

```
SELECT e.employee_name, e.street, e.city
FROM employee e
JOIN works w ON e.employee_name = w.employee_name
WHERE w.company_name = 'First Bank Corporation'
    AND w.salary > 10,000
    AND e.employee_name NOT LIKE 'Z%';
```

RA:

π_employee_name, street, city (σ_company_name = 'First Bank Cooporation' AND salary > 10000 AND employee_name NOT LIKE 'Z.I.' (employee $\bowtie$ works))

c) Find all employees in the database who do not work for First Bank Corporation.

→ SQL:

```
SELECT e.employee_name
FROM employee e
JOIN works w ON e.employee_name = w.employee_name
WHERE w.company_name != 'First Bank Corporation';
```

RA:

π_employee_name (σ_company_name $\neq$ 'First Bank Corporation' (employee $\bowtie$ works))

d) Find all employees in the database who earn more than each employee of small Bank Corporation.

→ SQL:

```
SELECT e.employee_name
FROM works e
WHERE e.salary > ALL (SELECT w.salary FROM works w
    WHERE w.company_name = 'small Bank Corporation');
```

RA:

π_employee_name (σ_salary > ALL (π_salary (σ_company_name = 'small Bank Corporation' (works))) (works))

e) Assume that the companies may be located in several cities. Find all companies located in every city in which small Bank corporation is located.

→ SQL:

```
SELECT DISTINCT c1.company_name
FROM company c1
WHERE NOT EXISTS (
        SELECT c2.city
        FROM company c2
        WHERE c2.company_name = 'small Bank Corporation'
          AND c2.city NOT IN (
                SELECT c3.city
                FROM company c3
                WHERE c3.company_name = c1.company_name
                )

);
```

RA:

π_company_name (company) − π_company_name (σ_city (σ_company_name = 'small Bank corporation' (company)))
(company))

f) Find the company that has the most employees

→ SQL:

```
SELECT w.company_name
FROM work w
```

Date / /
Page

```
GROUP BY w.company_name
ORDER BY COUNT (w.employee_name) DESC
LIMIT 1;
```

RA:

π_company_name (σ_count = MAX (γ_company_name; COUNT (employee_name) → count (works)))

5. Why normalization is important in DBMS? Explain anomalies in DBMS with example. In relation R = (A, B, C, D, E). The set of functional dependencies is: (A → BC, CD → E, B → D, E → A) decomposed it into R_1 = (A, B, C), R_2 = (A, D, E). show that this decomposition is a lossless-join decomposition.

→

Normalization is the process of efficiently organizing data in a database to minimize data redundancy and improve data integrity. It is important in DBMS for following reasons:-

- Elimination of redundant data storage
- Closed modeling of real world entities, processes and their relationship
- structuring of data so that model is flexible.
- Less storage data
- Easier to add data
- Flexible structure.

Date / /
Page

* Anomalies in DBMS.

i) Insertion Anomaly

→ Occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

Example: In a table storing employee and department information, you might not be able to add a new department unless an employee is assigned to it.

ii) Update Anomaly

→ Occurs when one or more instances of duplicated data are updated, but not all instances are updated consistently.

Example: If an employee's department name is stored redundantly, changing the department name requires updating multiple records, and any missed updates lead to inconsistency.

iii) Deletion Anomaly

→ Occurs when the deletion of data about one entity results in unintentional loss of data about another entity.

Example: If you delete an employee record that includes department details, you might lose information about the department as well.

Date / /
Page

Given Relation R = (A, B, C, D, E) with functional dependencies {A → BC, CD → E, B → D, E → A} and decomposition R1 = (A, B, C) and R2 = (A, D, E)

To prove the decomposition is loseless-join:

1) Compute Attribute Closure for R1 and R2:
   - (R1) = {A, B, C}
   - (R2) = {A, D, E}

2) Check Attribute Closure:
   - A+ in R1 : A → BC, so A+ = {A, B, C}
   - A+ in R2 : E → A and A → BC, so A+ = {A, B, C, D, E}

Since the closure of A in R2 covers all attributes in R, the decomposition is lossless-join.

6) Explain basic steps of query processing. Transform the following relational algebra expression using equivalence rule. show each step involved.

instructor (ID, name, dept_name, salary)
teaches (ID, course_id, sec_id, semester, year)
course (course_id, title, dept_name, credits)

$\Pi_{name, title} (\sigma_{dept\_name = "Music" \wedge year = 2017} (instructor \bowtie (teaches \bowtie \Pi_{course\_id, title} (course))))$

→ Query Processing - activities involved in retrieving data from the database.

• Basic steps in Query Processing

i) Parsing and Translation
- Convert SQL query into an internal format, typically a parse tree.

ii) Optimization
- Transform the internal representation into an optimized logical plan.

iii) Execution.
- Translate the logical plan into a physical plan then execute it on the database.

→ Given relational algebra expression

$\Pi_{name, title}(\sigma_{dept\_name = "Music" \wedge year = 2017}(instructor \bowtie (teaches \bowtie \Pi_{course\_id, title}(course))))$

Step By step Transformation

1) Combine selection:

$\sigma_{dept\_name = "Music" \wedge year = 2017}(instructor \bowtie teaches \bowtie \Pi_{course\_id, title}(course))$

2) Push selection Down:

$instructor \bowtie teaches \bowtie \sigma_{depth\_name = "Music"}(\Pi_{course\_id, title}(\sigma_{year = 2017}(course)))$

Date / /
Page

3) simplify inner selection

instructor ⋈ teaches ⋈ ($\pi_{course\_id, title}$ ($\sigma_{year = 2017 \wedge dept\_name =}$ "Music" (course)))

4) Combine Joins:

$\pi_{name, title}$ ($\sigma_{dept\_name = "Music" \wedge year = 2017}$ (instructor ⋈ teaches ⋈ course))

7. What is the role of index in DBMS? Explain primary index and secondary index with example. What are the characteristics of B+ tree?

→ Role of index in DBMS:

Indexes improve the speed of data retrieval operations on a database at the cost of additional writes and storage space to maintain the index data structure.

Primary Index:

- An index on a set of fields that includes the primary key fields.
- Ensure unique values and typically sorted to improve search efficiency.

Example: An index on 'employee_id' in an 'Employee' table.

Secondary Index

- An index on non-primary key columns.
- Used to improve the performance of queries that filter on

non-key columns.
Example: An index on 'department_name' in an 'Employee' table.

* B^+ tree:
B^+ tree indices are an alternative to indexed-sequential files.
Characteristics of B^+ Tree are:
- Balanced Tree structure: Ensures all leaf nodes are at same level
- Leaf Nodes Linked: Leaf nodes are linked to provide ordered traversal of data.
- Efficient insertions/deletions: Maintains balance through efficient node splitting & merging.
- Range Queries: supports efficient range queries due to its ordered nature.
- Multiple Levels: Internal nodes store keys and child pointers, while leaf nodes store actual data pointers.

8. Explain about the state diagram of transaction. How do test conflict serializability of a schedule S, explain in details with example.

→ State diagram of transaction:
- Active state: The transaction is executing.
- Partially committed: The transaction has completed

its execution but not yet committed.

- Committed : The transaction has failed during execution.
- Aborted : The transaction has been rolled back and undone.

* Conflict Serializability

- Determine if a schedule is serializable by constructing a precedance (or dependency) graph.
- Nodes represented transactions and directed edges represent conflicting operations (read/write conflicts)

Example :

1) Schedule S :
   - T1 : R(A), W(A)
   - T2 : R(A), W(B)
   - T3 : W(A), R(B)

2) Construct Precedance Graph :
   - T1 → T2 (T1 reads A before T2 writes A)
   - T2 → T3 (T2 writes B before T3 reads B)
   - T3 → T1 (T3 writes A before T1 reads A)

3) check for cycles :
   - The graph has a cycle (T1 → T2 → T3 → T1), so the schedule is not conflict serializable.

9) What are the different types of failure in DBMS ? Explain shadow page recovery technique with example.

→ Different types of failure in DBMS are as follows :-

a) Transaction Failure : Errors in transaction logic, input or integrity constraints.

b) System Failure : Hardware / software failures causing system crashes.

c) Media Failure : Physical damage to storage devices.

d) Application Failure : Errors in application programs causing data corruption.

* Shadow Page Recovery Technique :

a) Maintain two copies of the database pages :

- Current Page Table : Points to the current state of the pages.
- shadow Page Table : Points to the stable state before transaction begins.

b) During transaction :

- Make changes to copies of pages
- Update the current page table.

c) Commit

- Atomically switch the current page table to the new page table.

d) Roll back

- Discard changes by referencing the shadow page table.

Date / /
Page

Example:
- Initial Pages: P1 (current), P2 (shadow)
- Transaction modifies P1: New P1 written
- Commit: Update page table to point to new P1.
- Rollback: Discard new P1, use shadow P2.

10. Define transaction. What are the ACID properties in DBMS.

→ A transaction is a unit of work performed within a database management system (DBMS) against a database, treated in a coherent and reliable way independent of other transactions.

The ACID properties in DBMS are:
- Atomicity: Ensures that all operations within a transaction are completed; if not, the transaction is aborted.
- Consistency: Ensures that the database remains in a consistent state before and after the transaction.
- Isolation: Ensures that transaction are isolated from each other until they are completed.
- Durability: Ensures that once a transaction is committed, the changes are permanent, even in the event of a system failure.

11. Explain Distributed Database system with its types. What are different types data fragmentation techniques?

→

A distributed database system is a collection of multiple, logically interrelated databases distributed over a computer network.

Types of Distributed Database systems:

- Homogeneous : All sites use the same DBMS software
- Heterogeneous : Sites use different DBMS software, requiring middleware for integration
- Federated : Independent databases cooperating together, maintaining their autonomy.

* Data Fragmentation Techniques

- Horizontal Fragmentation : Divides a relation into subsets of tuples

  Example: Splitting 'Employee' table by department

- Vertical Fragmentation : Divides a relation into subsets of attributes

  Example : Splitting 'Employee' table into 'Employee-Personal' and 'Employee-Professional'.

- Hybrid Fragmentation : Combines horizontal and vertical for fragmentation.

  Example : First split 'Employee' table by department (horizontal) then split each fragment by attributes (vertical).

Date / /
Page

2079 Bhadra.

1. What are the advantages of Database Management System? List roles and responsibilities.

→ Advantages of Database Management System are:-

a) Data Redundancy Control:
- Minimizes data redundancy by integrating data across multiple applications.

b) Data consistency:
- Ensures data consistency through constraints and normalization.

c) Data security:
- Provides robust security features including user authentication and authorization.

d) Data integrity:
- Maintains data accuracy and integrity through constraints and data validation rules.

e) Data independence
- separates data structure from application programs, ensuring data independence.

f) Efficient Data Access
- Provides efficient query processing and indexing to facilitate fast data retrieval.

g) Concurrent Access
- supports concurrent data access for multiple users, ensuring data integrity.

h) Backup and Recovery
- offers automated backup and recovery options to protect against data loss.

Date / /
Page

Roles and Responsibilities of Database Administrator (DBA):

a) Installation & configuration
- Install and configure DBMS and related software.

b) Database Design:
- Design database schema, tables and relationships.

c) Performance Tuning:
- Optimize database performed by tuning SQL queries, indexes and configurations.

d) Backup & Recovery
- Implement and manage backup and recovery processes to ensure data safety.

e) Data integrity
- Ensure data integrity through constraints and validation rules.

f) Monitoring & Maintenance
- Monitor database performance and perform regular maintenance tasks.

g) Data Migration
- Manage Data Migration and conversion between different systems.

Date / /
Page

2a) Design an ER diagram for a company human resource database, "The Company has a set of branch offices. Each branch office has a set of departments. Each departments has a set of ~~departments~~ employees, a set of projects. Each employee has a job history, academic qualification. For each job type, the employee also has a salary history"

→ (PK = Primary Key, FK = Foreign key)

List of entities and their attributes are:

Entities	Attributes
• Branch Office	BranchID (PK), Location
• Department	DeptID (PK), BranchID (FK), DeptName
• Employee	EmpID (PK), DeptID(FK), EmpName, JobHistory, AcadQualification
• Project	ProjectID (PK), DeptID (FK) ProjectName.
• JobType	JobTypeID (PK), JobDescription, Salary History.

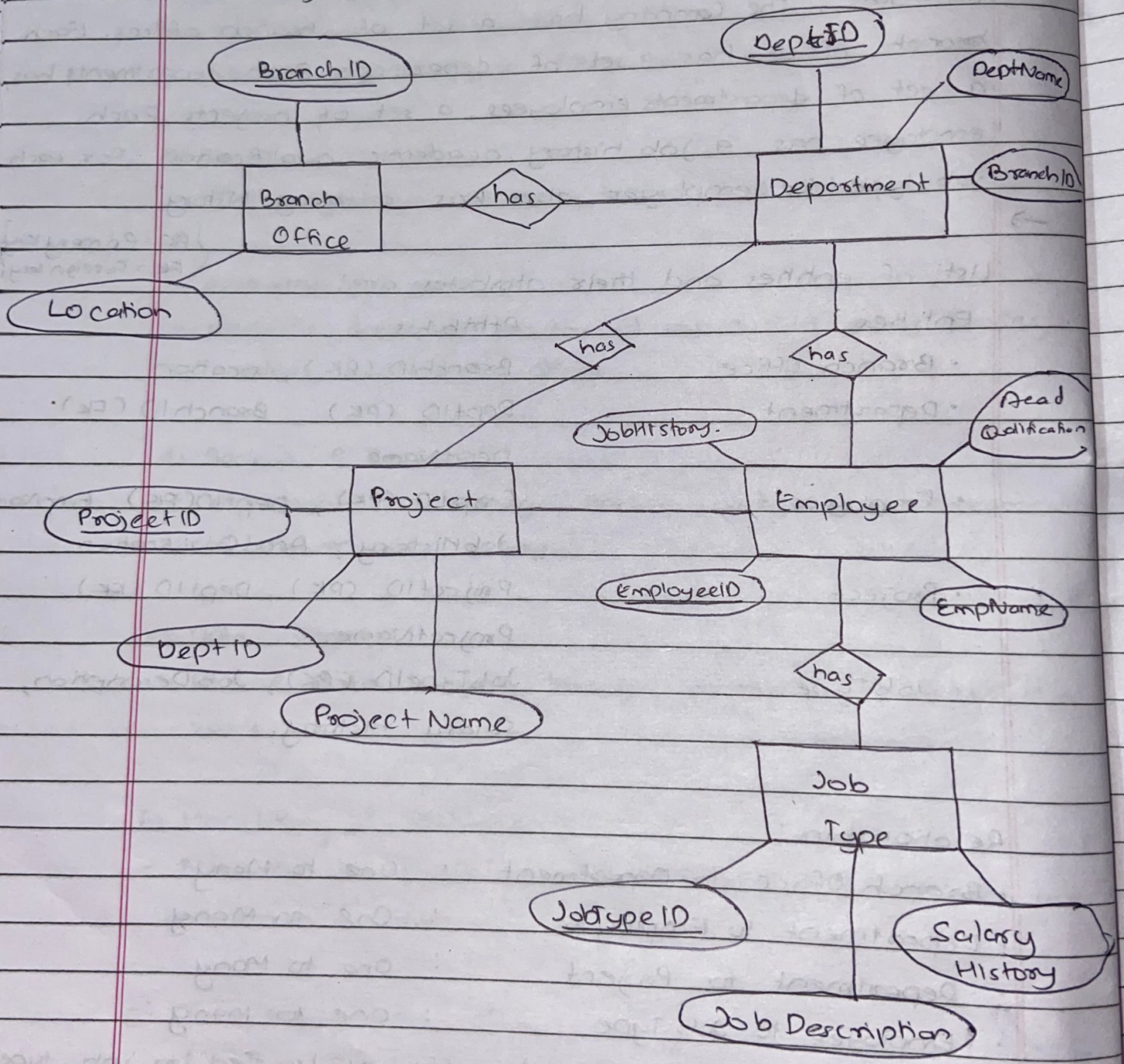
Relationships:

- Branch Office to Department : One to Many
- Department to Employee : One to Many
- Department to Project : One to Many
- Employee to Job Type : One to Many

(considering salary history is tied to job type)

Date / /
Page

ER Diagram:



Date / /
Page

26. What is key attribute? List out the types of keys and explain them briefly.

→ A key attribute is a unique identifier for an entity in a database. It uniquely identifies a record and helps maintain data integrity.

The types of keys are as follows:

i) Primary key
- A unique identifier for a record in a table. It ensures each record is unique.
- Example: 'EmpID' in Employee.

ii) Candidate key.
- An attribut or a set of attributes that can uniquely identify a record. A table can have multiple candidate keys, but only one primary key.
- Example: 'EmpID' and 'EmpEmail' if both are unique.

iii) Super key
- A set of attributes that can uniquely identifies a record. A super key may contain additional attributes that are not necessary for unique and identification.
- Example: '{EmpID, EmpMail}'

iv) Foreign key.
- An attribute in one table that refers to the primary key in another table. It establishes relationship between tables. Example: Dept ID in Employee.

v) Composite Key

- A primary key composed of multiple attributes. It is used when a single attributes is not sufficient to uniquely identify a record.
- Example: '{courseID, sectionID}' in a course offering table.

6. What is record organization? Explain the way of file organization. Compare secondary index and multilevel indexing techniques.

→ Record organization refers to how data records are stored within a file. This organization impacts how efficiently data can be accessed, inserted, updated and deleted.

File Organization Methods:

a. Heap (unordered) Files:

- Records are stored in the order they are inserted.
- Pros: simple to implement, efficient for bulk loading
- Cons: slow for search operations, especially for large datasets.

b. Sequential Files:

- Records are stored in a sorted order based on a key field.
- Pros: Efficient for range queries and sequential access.

- Cons: Insertion and deletion can be costly due to the need to maintain order.

c) Hashed Files:
- Records are stored based on a hash function applied to a key field.
- Pros: Very efficient for equality searches.
- Cons: Not suitable for range queries, potential for hash collisions.

d) Clustered Files:
- Related records from different tables are stored close together based on a clustering field.
- Pros: Improves Performance for join operations.
- Cons: Complex to maintain, especially with frequent updates.

* Comparison of Secondary Index and Multilevel Indexing Techniques.

Secondary Index	Multilevel Indexing
- It can be generated by a field which has a unique value for each record and it should be a candidate key.	- It is created when a primary index does not fit in memory.
- Relatively simple to implement	- More complex to implement.

transactions to maintain cons

Date / /
Page

7. Define transaction and explain its ACID properties. Define schedule and give proper examples. What is a serializable schedule?

→ A transaction is a unit of work performed within a database management system (DBMS) that is treated in a coherent and reliable way independent of other transactions. A transaction ensures that a sequence of operations is completed sucessfully and the database is left in a consistent state.

ACID properties

a. Atomicity
- Ensures that all operations within a transaction are completed, if not, the transaction is aborted.
- Eg: A bank transfer transaction either full completes (debiting one account and crediting another) or does not execute at all.

b. Consistency:
- Ensures that a transaction takes the database from one consistent state to another.
- Eg: If a transaction violates database integrity constraints it will be rolled back.

c. Isolation
- Ensures that the operation of a transaction are isolated from other transactions.

- Example: Two transactions modifying the same data will not interfer with each other.

d. Durability

- Ensures that the results of a transaction are permanently recorded in the datase.
- Example: Once a bank tranfer is completed, the changes are not lost even if the system crashes.

→ Schedule: A schedule is an ordered sequence of transactions. It can be:

- ~~Serials~~ Serializable
  - A schedule is serialization if its result is the same as if the transactions were executed serially.

- Non-serializable
  - A schedule that may result in data inconsitencies or anomalies.

Eg: ~~A~~ Consider two transactions, T1 and T2:

- T1: Read(A), A = A + 100, WRITE(A)
- T2: Read(A), A = A * 2, WRITE(A)

A serialize schedule could be:

1. T1: Read(A), A = A + 100, Wrik(A)
2. T2: Read(A), A = A * 2, Wrik(A).

the UNDO operation reverses the
transactions

Date / /
Page

8. Define checkpointing with example. How REDO and UNDO operations performed in log based recovery mechanism?

→ Checkpointing is a process used in DBMS to reduce the amount of time required to recover from a system crash. It involves periodically saving current state of the database and the transaction log to stable storage.

Example:

Suppose a database performs a checkpoint every 5 minutes. At each checkpoint, the DBMS performs the following actions.

a. Write all in-memory modified data (dirty pages) to disk.

b) Record the checkpoint in the transaction log.

c) Flush the transaction log to disk.

If a crash occurs, the system can restart from the last checkpoint, reducing the number of transactions that need to be reprocessed.

→ REDO and UNDO Operations in log-Based Recovery.

In log-based recovery, the DBMS uses a log (a sequence of log records) to record all changes made to the database. This log is crucial for ensuring data integrity and consistency in the event of a crash or failure. Each log record typically contains information about a transaction, such as its start, the changes it made and its commit or abort.

Century

Date / /
Page

- REDO operation:

The REDO operation ensures that all the changes made by committed transactions are reflected in the database. This is necessary because some changes may not have been written to the disk before the cash.

Steps:

- Scan the log from last checkpoint to the end of the log.
- For each log record, if the transaction is committed, apply the changes to the database.

Example:

<checkpoint start>
<checkpoint end, T1, T2>
<T1, start>
<T1, write, A, 100, 200> -- T1 changed A from 100 to 200
<T1, commit>
<T2, start>
<T2, write, B, 300, 400> -- T2 changed B from 300 to 400.

If a crash occurs after these log records, the recovery process will:- REDO the changes made by T1 : set A to 200.

- UNDO operation:

The UNDO operation reverses the changes made by uncommited transactions to maintain consistency. This is necessary because

Century

ed from file system in following

changes from uncommited transactions should not be present in database.

Steps:

- scan the log backward from the end to last checkpoint
- For each long record, if the transaction is not committed, reverse the changes made.

Example:

<checkpoint start>

<checkpoint end, T1, T2>

<T1, start>

<T1, write, A, 100, 200> -- T1 changed A from 100 to 200

<T1, commit>

<T2, start>

<T2, write, B, 300, 400> -- T2 changed B from 300 to 400.

If a crash occurs, the recovery process will :- UNDO the change made by T2 : set B block back to 300.

Date / /
Page

9. Write short notes on:

a) Advantages of object oriented database model.

→ The object-oriented database model is a database management approach that integrates object-oriented programming principles with database technology. This model aims to address the limitations of traditional relational databases, especially when dealing with complex data types and relationships. Its advantages are:-

- Data Abstraction: supports complex data types and structures.
- Encapsulation: combines data and behaviour, ensuring data integrity
- Inheritance: Reusability of code and structure.
- Polymorphism: Ability to process objects differently based on their data type or class.
- Extensibility: Easy to extend database schema with new data types and methods.

b.) Parallel Database Architecture

→ A parallel database architecture employs multiple processors and storage devices to perform database operations concurrently, thereby improving performance and scalability. Two main performance measures:

- throughput - the number of tasks that can be completed in a given time interval.
- response time - the amount of time it takes to complete a single task from the time it is submitted.

system in ...

Date / /
Page

Advantages:
- scalability: can handle large volumes of data and users.
- speed: Faster query processing and transaction throughput.
- Fault Tolerance: Better reliability and availability due to distributed components.

c) Data Warehousing
→ A centralized repository for storing large volumes of structured data from multiple sources, optimized for query and analysis.
Its characteristics are:
- subject-Oriented: organized around key business subjects.
- Integrated: Combines data from various sources.
- Time-Variant: Historical data storage for trend analysis.
- Non-Volatile: Data is read-only, not subject to changes after insertion.

Advantages:
- Improved Decision Making: Provides comprehensive data analysis capabilities.
- Data Consolidation: Combines disparate data sources into a single view.
- Performance: Optimized for complex queries and large-scale ~~reading~~ reporting.

Date / /
Page

3 a) Consider the following relational database model:
Product (product_id, pname, price, pdescription)
Customer (customer_id, cname, address, phone)
Purchase (product_id, customer_id, quantity, sales_mid)
Salesman (sales_mid, sname, salary).
Write SQL statement for the following:

i) Create table 'Purchase' with foreign keys:

→
```
CREATE TABLE Purchase (
    product_id INT,
    customer_id INT,
    quantity INT,
    sales_mid INT,
    FOREIGN KEY (product_id) REFERENCES Product (product_id),
    FOREIGN KEY (customer_id) REFERENCES Customer (customer_id),
    FOREIGN KEY (sales_mid) REFERENCES Salesman (sales_mid)
);
```

ii. List name and address of all customers who purchased the product SSD:

→
```
SELECT c.cname, c.address
FROM Customer c
JOIN Purchase p ON c.customer_id = p.customer_id
JOIN Product pr ON p.product_id = pr.product_id
WHERE pr.pname = 'SSD';
```

ed from file system in

iii) Find the name of the product which purchase quantity is maximum.

→
```
SELECT pname
FROM Product
WHERE product_id = (
    SELECT product_id
    FROM Purchase
    GROUP BY product_id
    ORDER BY SUM(quantity) DESC
    LIMIT 1
);
```

iv) Increase the salary of all salesmen by 5%. who have sold at least 10 SSD.

→
```
UPDATE Salesman
SET salary = salary * 1.05
WHERE sales_mid IN (
    SELECT p.sales_mid
    FROM Purchase p
    JOIN Product pr ON p.product_id = pr.product_id
    WHERE pr.pname = 'SSD'
    GROUP BY p.sales_mid
    HAVING SUM(p.quantity) >= 10
);
```

3b→ For the relational database model given in the Question No. 3 (a). write relational algebraic

Date / /
Page

expression for the following:

i. Display name of the customers who are from Kathmandu and name starts with 'R'.

→ $\Pi_{name}(\sigma_{address = 'kathmandu' \text{ AND } cname \text{ LIKE } 'R.*'}(customer))$.

ii. List the name of product purchased by customer 'sita' from the salesman 'Ram'.

→ $\Pi_{pname}($
$(\sigma_{cname = 'Sita'}(customer) \bowtie Purchase)$
$\bowtie (\sigma_{sname = 'Ram'}(salesman) \bowtie Product)$
$)$

iii. Find the product wise total purchased quantity.

→ $\gamma_{product\_id,\ SUM(quantity)}(Purchase)$

iv) Update the price of all products by 8.1.

→ *RA doesn't directly support updates*

UPDATE Product SET price = price * 1.08

4) What is Normalization? Why is it important? How can you convert an Unnormalized table to third Normal Form? Explain with example.

→ Normalization is a database design technique that

create, retrieve, update

Date / /
Page

organizes tables in a manner that reduces redundancy and dependency by dividing large tables into smaller ones and defining relationships among them. The main goals are to eliminate redundancy, avoid data anomalies, and ensure data integrity.

Importance of Normalization:

- Eliminates Redundancy: By dividing data in related tables, normalization minimizes duplicate data.
- Improves Data Integrity: Ensures consistency of data through constraints and relationships.
- Avoids Anomalies: Reduces insertion, update and deletion anomalies by organizing data properly.
- Optimizes Queries: Simplifies the database schema, making queries more efficient and reducing the complexity of joins.

→ Converting Unnormalized Table to Third Normal Form (3NF):

a. Unnormalized Table (UNF)

- Contains repeating groups or multiple values in a single field.
- Example:

OrderID	Customer	Items	Price
1	Alice	Pen, Notebook	10, 20
2	Bob	Pencil, Eraser	5, 3

Date / /
Page

b. First Normal Form (1NF)

- Remove repeating groups, ensuring each field contains only atomic values.

Example:

OrderID	Customer	Item	Price
1	Alice	Pen	10
1	Alice	Notebook	20
2	Bob	Pencil	5
2	Bob	Eraser	3

c) Second Normal Form (2NF)

- Ensures the table is in 1NF and all non-key attributes are fully functional dependent on the primary key. Remove partial dependencies.

Eg:

Split into two tables.

- Orders table:

Order ID	Customer
1	Alice
2	Bob

- Order Items table:

Order ID	Item	Price
1	Pen	10
1	Notebook	20
2	Pencil	5
2	Eraser	3.

Date / /
Page

d. Third Normal Form

Ensures the table is in 2NF and all attributes are functionally dependent only on the primary key. Remove transitive dependencies.

Eg:

Split into three tables:

- Orders table

OrderID	Customer
1	Alice
2	Bob.

- Items table:

ItemID	Item
1	Pen
2	Notebook
3	Pencil
4	Eraser.

- Orders Items table

OrderID	ItemID	Price
1	1	10
1	2	20
2	3	5
2	4	3

Date / /
Page

5. Explain the steps of query processing with examples. Compare cost based evaluation and heuristic optimization method.

→ The steps of query processing involves:

a) Parsing
- The query is parsed to check syntax and semantics.
- Example: 'SELECT * FROM Employee where age > 30;

b) Translation
- The parsed query is translated into a relational algebra expression.
- Eg: 'σ_age > 30 (Employee)'

c) Optimization.
- The relational algebra expression is optimized to improve efficiency.
- Eg: Use indexes to speed up the selection.

d) Execution Plan Generation
- An execution plan is created detailing the operation to be performed.
- Eg: Scanning the Employee table using an index on the age column.

e) Execution:
- The DBMS executes the plan and retrieves the data.
- Eg: Returns all employees with age > 30.

Date / /
Page

Example for Query Processing steps:

→ Consider the query.

SELECT name FROM Employee WHERE age > 30
AND department = 'HR';

a. Parsing
- ~~SELECT name~~ & check the syntax and semantics of the query.

b) Translation
- translate to relational algebra

$\pi_{name}(\sigma_{age > 30 \wedge department = 'HR'}(Employee))$

c) Optimization
- Apply heuristics or cost-based optimization.

$\sigma_{department = 'HR'}(\sigma_{age > 30}(Employee))$

d. Execution plan Generation
- Generate an execution plan, potentially using indexes on 'age' and 'department'.

e. Execute
- Execute the plan and return the names of employees matching the criteria.

→ Comparison between cost based evaluation and Heuristic Optimization Method.

Cost Based Evaluation	Heuristic Optimization Method.
- Considers various possible execution plans and evaluates the cost of each based on factors like I/O, CPU usage and memory usage.	- Uses a set of heuristic rules to transform the query into an efficient form without evaluating the cost.
- High precision due to detailed cost estimation.	- Low precision due to reliance on general rules
- Complex to implement as it requires detailed statistics and cost models.	- Simple to implement.
- Better for complex queries.	- Efficient for simple, common queries.