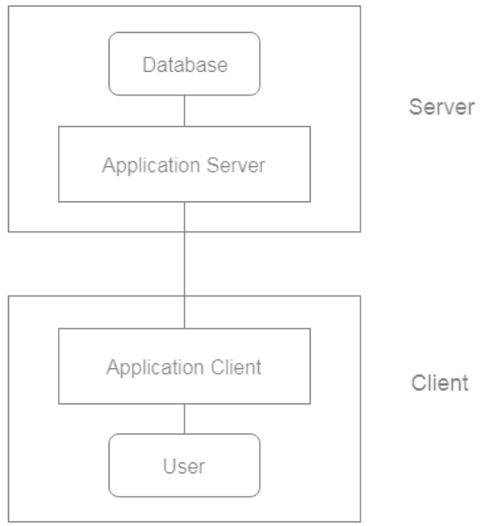
10

Database Management System



Government of Nepal
Ministry of Education, Science and Technology
Curriculum Development Centre
Sanothimi, Bhaktapur
Phone : 5639122/6634373/6635046/6630088
Website- https://www.moecdc.gov.np
Email- info@moecdc.gov.np

Technical and Vocational Stream
Learning Resource Material

Database Management System
(Grade 10)
Computer Engineering

Government of Nepal
Ministry of Education, Science and Technology
Curriculum Development Centre
Sanothimi, Bhaktapur

Publisher: Government of Nepal
Ministry of Education, Science and Technology
Curriculum Development Centre
Sanothimi, Bhaktapur

© Publisher
Layout by Khados Sunuwar

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any other form or by any means for commercial purpose without the prior permission in writing of Curriculum Development Centre.

Preface

The curriculum and curricular materials have been developed and revised on a regular basis with the aim of making education objective-oriented, practical, relevant and job oriented. It is necessary to instill the feelings of nationalism, national integrity and democratic spirit in students and equip them with morality, discipline, self-reliance, creativity and thoughtfulness. It is essential to develop linguistic and mathematical skills, knowledge of science, information and communication technology, environment, health and population and life skills in students. It is also necessary to bring the feeling of preserving and promoting arts and aesthetics, humanistic norms, values and ideals. It has become the need of the present time to make them aware of respect for ethnicity, gender, disabilities, languages, religions, cultures, regional diversity, human rights and social values to make them capable of playing the role of responsible citizens with applied technical and vocational knowledge and skills. This learning resource material for computer engineering has been developed in line with the Secondary Level computer engineering Curriculum with an aim to facilitate the students in their study and learning on the subject by incorporating the recommendations and feedback obtained from various schools, workshops, seminars and interaction programs attended by teachers, students, parents and concerned stakeholders.

In bringing out the learning resource material in this form, the contribution of the Director General of CDC Mr. Yubaraj Paudel and members of the subject committee Dr. Baburam Dawadi, Dr. Sarbim Sayami, Mrs. Bibha Sthapit, Mrs. Trimandir Prajapati is highly acknowledged. This learning resource material is compiled and prepared by Mr. Bimal Thapa, Mr, Bikesh Shrestha, Mr. Nabin Maskey. The subject matter of this material is edited by Mr. Badrinath Timsina and Mr. Khilanath Dhamala. Similarly, the language is edited by Mr. Bijay Kumar Ranabhat. CDC extends sincere thanks to all those who have contributed to developing this material in this form.

This learning resource material contains a wide coverage of subject matters and sample exercises which will help the learners to achieve the competencies and learning outcomes set in the curriculum. Each chapter in the material clearly and concisely deals with the subject matters required for the accomplishment of the learning outcomes. The Curriculum Development Centre always welcomes creative and constructive feedback for the further improvement of the material.

2082 BS

Curriculum Development Centre
Sanothimi, Bhaktapur

Table of Content

Unit	Content	Page No.
1.	Introduction to Database System	1-12
2.	Entity Relationship Model (ER-Model)	13-25
3.	Relational Model	26-33
4.	Structure Query Language Overview	34-45
5.	Relational Design DataBase	46-52
6.	DataBase Transaction	53-60
7.	DataBase Backup, Recovery and Security	61-73

Guidelines to Teachers

A. Facilitation Methods

The goal of this course is to combine the theoretical and practical aspects of the contents needed for the subject. The nature of contents included in this course demands the use of practical or learner focused facilitation processes. Therefore, the practical side of the facilitation process has been focused much. The instructor is expected to design and conduct a variety of practical methods, strategies or techniques which encourage students engage in the process of reflection, sharing, collaboration, exploration and innovation new ideas or learning. For this, the following teaching methods, strategies or techniques are suggested to adopt as per the course content nature and context.

Brainstorming

Brainstorming is a technique of teaching which is creative thinking process. In this technique, students freely speak or share their ideas on a given topic. The instructor does not judge students' ideas as being right or wrong, but rather encourages them to think and speak creatively and innovatively. In brainstorming time, the instructor expects students to generate their tentative and rough ideas on a given topic which are not judgmental. It is, therefore, brainstorming is free-wheeling, non-judgmental and unstructured in nature. Students or participants are encouraged to freely express their ideas throughout the brainstorming time. Whiteboard and other visual aids can be used to help organize the ideas as they are developed. Following the brainstorming session, concepts are examined and ranked in order of importance, opening the door for more development and execution. Brainstorming is an effective technique for problem-solving, invention, and decision-making because it taps into the group's combined knowledge and creative ideas.

Demonstration

Demonstration is a practical method of teaching in which the instructor shows or demonstrates the actions, materials, or processes. While demonstrating something the students in the class see, observe, discuss and share ideas on a given topic. Most importantly, abstract and complicated concepts can be presented into visible form through demonstration. Visualization bridges the gap between abstract ideas and concrete manifestations by utilizing the innate human ability to think visually. This enables students to make better decisions, develop their creative potential, and obtain deeper insights across a variety of subject areas.

Peer Discussion

Peer conversation is a cooperative process where students converse with their peers to exchange viewpoints, share ideas, and jointly investigate subjects that are relevant or of mutual interest. Peer discussion is an effective teaching strategy used in the classroom to encourage critical thinking, active learning, and knowledge development. Peer discussions encourage students to express their ideas clearly, listen to opposing points of view, and participate in debate or dialogue, all of which contribute to a deeper comprehension and memory of the course material. Peer discussions also help participants develop critical communication and teamwork skills by teaching them how to effectively articulate their views, persuasively defend their positions, and constructively respond to criticism.

Peer conversation is essential for professional growth and community building outside of the classroom because it allows practitioners to share best practices, work together, and solve problems as a group. In addition to expanding their knowledge horizon and deepening their understanding, peer discussions help students build lasting relationships and a feeling of community within their peer networks.

Group Work

Group work is a technique of teaching where more than two students or participants work together to complete a task, solve a problem or discuss on a given topic collaboratively. Group work is also a cooperative working process where students join and share their perspectives, abilities, and knowledge to take on challenging job or project. Group work in academic contexts promotes active learning, peer teaching, and the development of collaboration and communication skills. Group work helps individuals to do more together than they might individually do or achieve.

Gallery Walk

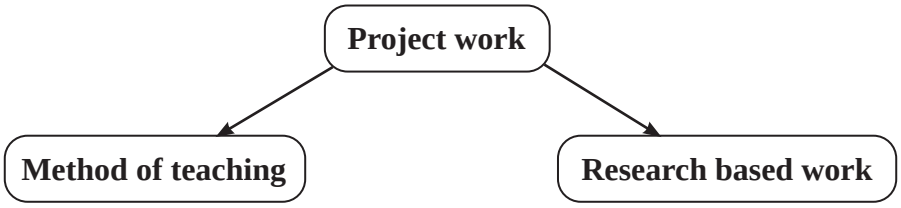
Gallery walk is a critical thinking strategy. It creates interactive learning environment in the classroom. It offers participants or students a structured way to observe exhibition or presentation and also provides opportunity to share ideas. It promotes peer-to-peer or group-to-group engagement by encouraging participants to observe, evaluate and comment on each other's work or ideas. Students who engage in this process improve their communication and critical thinking abilities in addition to their comprehension of the subject matter, which leads to a deeper and more sophisticated investigation of the subjects at hand.

Interaction

The dynamic sharing of ideas, knowledge, and experiences between people or things is referred to as interaction, and it frequently takes place in social, academic, or professional settings. It includes a broad range of activities such as dialogue, collaboration or team work, negotiation, problem solving, etc. Mutual understanding, knowledge sharing, and interpersonal relationships are all facilitated by effective interaction. Interaction is essential for building relationships, encouraging learning, and stimulating creativity in both in-person and virtual contexts. Students can broaden their viewpoints, hone their abilities, and jointly achieve solutions to difficult problems by actively interacting with others.

Project Work

Project work is a special kind of work that consists of a problematic situation which requires systematic investigation to explore innovative ideas and solutions. Project work can be used in two senses. First, it is a method of teaching in regular class. The next is: it is a research work that requires planned investigation to explore something new. This concept can be presented in the following figure.



Project work entails individuals or teams working together to achieve particular educational objectives. It consists of a number of organized tasks, activities, and deliverables. The end product is important for project work. Generally, project work will be carried out in three stages. They are:

- Planning
- Investigation
- Reporting

B. Instructional Materials

Instructional materials are the tools and resources that teachers use to help students. These resources/materials engage students, strengthen learning, and improve conceptual comprehension while supporting the educational goals of a course or program. Different learning styles and preferences can be accommodated by the variety of instructional

resources available. Here are a few examples of typical educational resource types:

- Daily used materials
- Related Pictures
- Reference books
- **Slides and Presentation:** PowerPoint slides, keynote presentations, or other visual aids that help convey information in a visually appealing and organized manner.
- **Audiovisual Materials:** Videos, animations, podcasts, and other multimedia resources that bring concepts to life and cater to auditory and visual learners.
- **Online Resources:** Websites, online articles, e-books, and other web-based materials that can be accessed for further reading and research.

Maps, Charts, and Graphs: Visual representations that help learners understand relationships, patterns, and trends in different subjects.

Real-life Examples and Case Studies: Stories, examples, or case studies that illustrate the practical application of theoretical concepts and principles.

C. Assessment

Formative Test

Classroom discussions: Engage students in discussions to assess their understanding of concepts.

Quizzes and polls: Use short quizzes or polls to check comprehension during or after a lesson.

Homework exercises: Assign tasks that provide ongoing feedback on individual progress.

Peer review: Have students review and provide feedback on each other's work.

Summative Test

Exams: Conduct comprehensive exams at the end of a unit or semester.

Final projects: Assign projects that demonstrate overall understanding of the subject.

Peer Assessment

Group projects: Evaluate individual contributions within a group project.

Peer feedback forms: Provide structured forms for students to assess their peers.

Classroom presentations: Have students assess each other's presentations.

Objective Test

Multiple-choice tests: Use multiple-choice questions to assess knowledge.

True/False questions: Assess factual understanding with true/false questions.

Matching exercises: Evaluate associations between concepts or terms.

Portfolio Assessment

Compilation of work: Collect and assess a variety of student work samples.

Reflection statements: Ask students to write reflective statements about their work.

Showcase events: Organize events where students present their portfolios to peers or instructors.

Observational Assessment

Classroom observations: Observe students' behavior and engagement during class.

Performance observations: Assess practical skills through direct observation.

Field trips: Evaluate students' ability to apply knowledge in real-world settings.

Introduction to Database System

Unit 1

1.1 Concept of Data, Information, Database, and Database Management System

Data

Data is a collection of unprocessed raw facts, figures, or observations. Data alone lacks context, which makes it difficult to interpret without additional processing. Examples of data include a list of numbers, a name without context, or dates without significance. Data can be numerical (e.g., 42, 98, 16), textual (e.g., "apple," "book"), or symbolic.

Data comes in various forms:

a) **Quantitative Data:** Quantitative data refers to numerical information like weight, height, etc.

b) **Qualitative Data:** Qualitative data refers to non-numeric information like opinions, perceptions, etc.

Information

Information is the result of processing data to give it meaning. It provides context, making the data understandable. For example, if data represents scores (e.g., "John scored 100 in Math"), that becomes information by giving context to John's score, indicating that he performed well in Math. In essence, information is data that has been organized or processed in a way that adds value.

Let us compare the data and information in a tabular format

Data	Information
Data is defined as unstructured information such as text, observations, images, symbols, and descriptions.	Information refers to processed, organized, and structured data. It gives context for the facts and facilitates decision making.
In other words, data provides no specific function and has no meaning on its own	In other words, information is processed data that makes sense to us.

Data are the variables that help to develop ideas/conclusions.	Information is meaningful data.
Data are text and numerical values.	Information is refined form of actual data.
Data doesn't rely on information.	While information relies on data.
Bits and bytes are the measuring unit of data.	Information is measured in meaningful units like time, quantity, etc.
Data is a collection of facts, which itself has no meaning.	Information puts those facts into context.
Example of data is student test scores.	Example of information is average score of class that is derived from given data.

Table 1.1: Data Vs Information

Database

A **database** is an organized collection of structured information or data stored electronically, typically in a computer system. A database enables easy access, management, and updating of information. Databases are commonly used in various fields, such as education, healthcare, banking, and retail, where structured information is vital for operations. For example, a hospital database may store patient information, appointment schedules, and treatment records.

Database Management System (DBMS)

A Database Management System (DBMS) is software designed to define, create, manipulate, retrieve, and manage data in a database. It serves as an interface between the user and the database, providing tools to create, read, update, and delete data efficiently. Examples of popular DBMSs include MySQL, Oracle, PostgreSQL, and Microsoft SQL Server. DBMS ensures data integrity, security, and concurrent access to data, making it a crucial technology for modern data management.

Characteristics of DBMS

a) It uses a digital repository established on a server to store and manage the information.

b) It can provide a clear and logical view of the process that manipulates data.

c) DBMS contains automatic backup and recovery procedures.

d) It contains ACID properties which maintain data in a healthy state in case of failure.

e) It can reduce the complex relationship between data.

f) It is used to support manipulation and processing of data.

g) It is used to provide security of data.

h) It can view the database from different viewpoints according to the requirements of the user.

DBMS allows users the following tasks

a) **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.

b) **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.

c) **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.

d) **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

Some of the Leading DBMS are:

a) MySQL(Open Source, From Oracle)

b) Oracle (Proprietary Software from Oracle)

c) MS SQL Server (Proprietary Software from Microsoft)

d) IBM DB2 (Proprietary Software from IBM)

e) Postgre SQL(Open Source Database)

f) MongoDB (NoSQL Database)

1.2 Limitations of File System

File systems were the traditional method for storing data before the advent of databases. A file system is a method used by an operating system to store, organize, and retrieve files on storage devices like hard drives, SSDs, and USBs. It provides a way to manage how data is saved and accessed on these devices. Despite being relatively simple, file systems have several limitations. They are follows:

a) **Data Redundancy and Inconsistency:** Data is often duplicated in multiple files, leading to inconsistencies if updates are not applied everywhere. For example, a customer's updated contact information in one file might differ in another.

b) **Data Isolation:** Data stored in separate files is hard to access in a unified way. For instance, retrieving a student's records requires accessing multiple files individually.

c) **Limited Data Security:** File systems lack strong security controls, making data more vulnerable to unauthorised access compared to the user access controls available in DBMS.

d) **Concurrency Issues:** When multiple users try to access the same data simultaneously, file systems often struggle, leading to potential data conflicts or corruption.

e) **Inefficient Data Retrieval:** Retrieving data is time-consuming as each file must be searched separately, which is inefficient, especially with large datasets.

f) **No Centralised Control:** File systems lack centralized data management, making it hard to maintain data integrity, backups, and updates consistently.

1.3 Advantages and Disadvantages of Database Systems

Advantages of Database Systems

a) **Data Integrity and Consistency:** DBMS enforces integrity constraints, ensuring that data remains accurate and consistent across all database records.

b) **Enhanced Data Security:** DBMS provides advanced security features such as user authentication, encryption, and access control, protecting data from unauthorised users.

c) **Reduced Data Redundancy:** By storing data centrally, DBMS reduces duplication, there by optimizing storage and preventing inconsistent information.

d) **Improved Data Access:** Databases allow concurrent data access for multiple users, making collaboration and multi-user operations easier.

e) **Efficient Data Management:** DBMS automates data storage, retrieval, and manipulation processes, enhancing overall system efficiency.

f) **Backup and Recovery Mechanisms:** DBMS offers tools for automatic backup and recovery, ensuring data safety even in case of system failures.

g) **Support for Large Datasets:** DBMS is designed to handle large amounts of data, making it ideal for applications with extensive data storage needs.

Disadvantages of Database Systems

a) **High Cost:** Implementing a DBMS can be costly due to software licenses, hardware requirements, and skilled personnel needed for maintenance.

b) **Complexity:** Setting up and managing a DBMS requires technical knowledge, which can be challenging for organizations without IT support.

c) **Hardware Requirements:** DBMS requires powerful computing resources, especially when managing extensive databases, which can lead to additional hardware costs.

d) **Backup and Recovery Overhead:** Regular backups are necessary to ensure data availability, which can be resource-intensive and slow down performance.

1.4 Applications of Database Systems

Database systems are essential across a wide range of industries, where they help streamline and automate various processes. Here are some common applications:

a) **Banking:** Banks use databases to manage customer information, account details, and transaction history, ensuring secure and efficient banking operations.

b) **Healthcare:** Databases store patient records, medical history, appointment schedules, and treatment data, helping healthcare providers offer timely and effective care.

c) **Education:** Schools and universities use databases to manage student enrollment, grades, schedules, and teacher information.

d) **Retail and E-commerce:** Retailers use databases to track product inventory, sales, and customer data, which is essential for managing supply chains and improving customer experience.

e) **Telecommunications:** Telecommunication companies manage customer information, billing, call records, and data plans in databases to provide seamless connectivity services.

f) **Human Resources:** Companies use databases to store employee information, payroll, attendance, and performance records, simplifying HR management.

1.5 Types of Database Users

A database user is defined as a person who interacts with data daily, updating, reading, and modifying the given data. Database users can access and retrieve data

from the database through the Database Management System (DBMS) applications and interfaces. Database users know the value of data security, privacy, and integrity, as well as how to query data from databases using Structured Query language (SQL) or other tools. Here are the types of database users:

a) **Navie Users:** Navie User are the unsophisticated user who don't have any DBMS knowledge but they frequently use database application in their daily life to get the desired results. For example, Railway's ticket booking user are naive users. These users require minimal technical knowledge, relying on the application to perform tasks like retrieving information or making simple data updates.

b) **Sophisticated User:** Sophisticated users are skilled individuals who use the database for advanced data retrieval and analysis.They often write complex queries or scripts to retrieve specific data for research, analysis, or reporting. Typical sophisticated users include data scientists, engineers, or researchers working with large datasets.They are usually proficient in SQL or other query languages and may use specialized software for data manipulation and analytics.

c) **Application Programmers:** Application programmers are responsible for developing and maintaining applications that interact with the database. They write code that enables users to perform tasks such as data entry, data retrieval, and updates within the application. For example, a developer may create a web application for online shopping, which interacts with a product and user database.These programmers require strong programming skills and an understanding of how to connect applications to databases securely and efficiently.

d) **System Analysts:** System analysts design and plan database systems to meet organizational requirements.They work on structuring the database to ensure it meets the needs of the business and can handle necessary workflows and processes.For example, a system analyst may plan a database to manage inventory and sales for a retail company.Their expertise includes analyzing business requirements, database design, and working closely with database administrators and application programmers.

1.6 DBMS Architecture

The DBMS design depends upon its architecture. The basic client/server architecture

is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.The client/server architecture consists of many PCs and a workstation which are connected via the network. DBMS architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Architecture

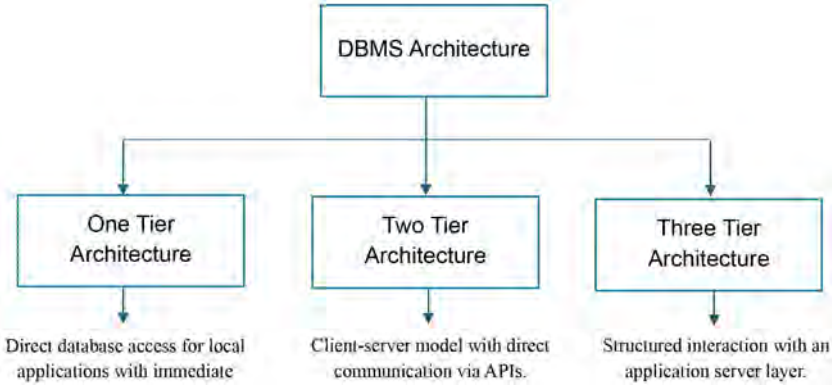


Fig: 1.1: DBMS Architecture

a) One Tier Architecture

The simplest DBMS architecture is a single tier architecture. In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and use it. Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users. The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

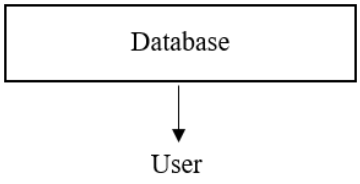


Fig 1.2: One Tier Architecture

b) Two Tier Architecture

The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: ODBC, JDBC are used.The user interfaces and application programs are run on the client-side. The server side is responsible to

provide the functionalities like: query processing and transaction management. To communicate with the DBMS, client-side application establishes a connection with the server side.

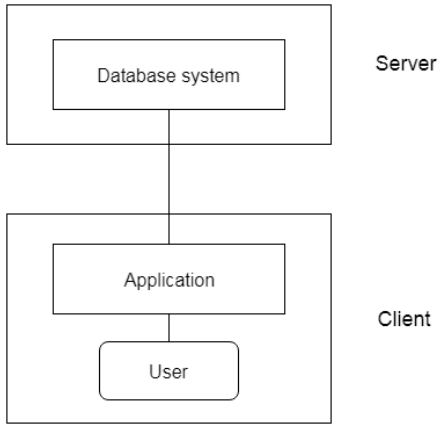


Fig 1.3: Two Tier Architecture

c) Three Tier Architecture

The three tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server. The application on the client-end interacts with an application server which further communicates with the database system.End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.The 3-Tier architecture is used in case of large web application.

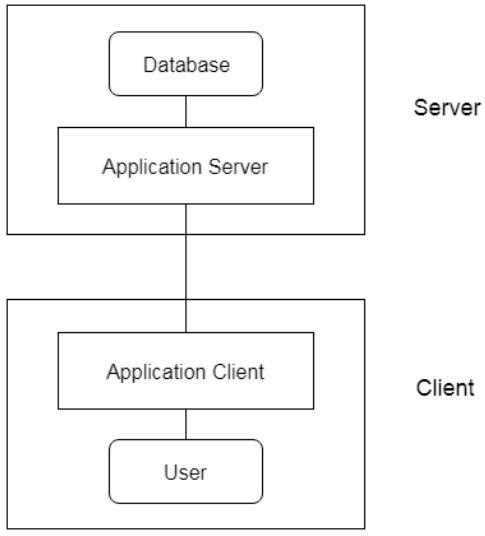


Fig 1.4: Three Tier Architecture

1.7 Database Models

Different database models represent data and relationships in varied ways, making it easy to handle complex data structures. Some of the main models include:

- **Hierarchical Model:** Hierarchical model organizes data in a tree-like structure where each record has a single parent, and each parent can have multiple children. This model is suitable for applications with a clear hierarchical relationship, such as file systems.

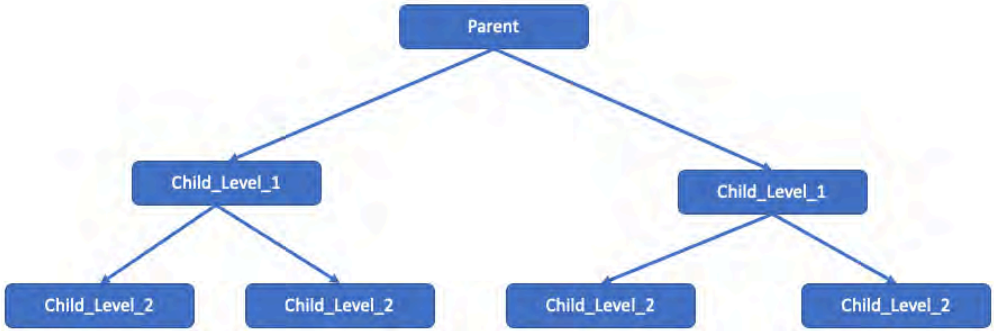


Fig. 1.5 Hierarachical Model

- **Network Model:** Network model is an extension of the hierarchical model, allowing multiple relationships. Each child can have multiple parents, making it flexible but complex to manage.

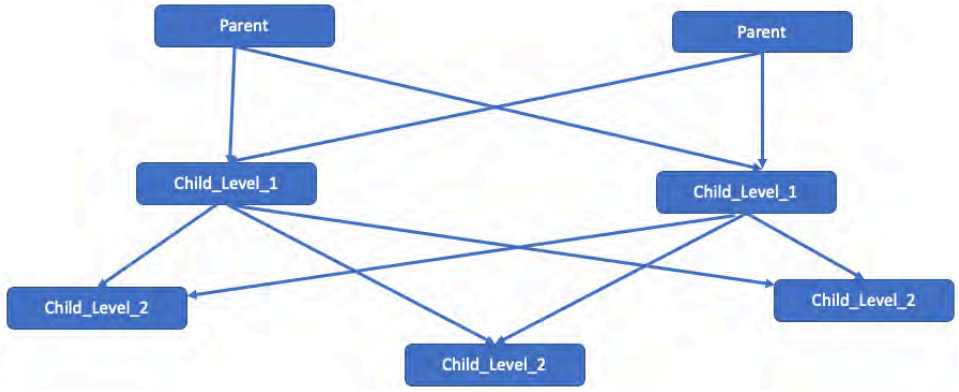


Fig. 1.6 Network Model

- **Relational Model:** Relational model is the most widely used model, which organizes data into tables (relations) where rows represent records and columns represent attributes. It allows for flexible querying using SQL and is ideal for applications requiring extensive data manipulation.

Employee_Name	Employee_Post	Employee_Salary
Ram	Manager	50000
Sita	A.Manager	40000
Laxman	Guard	15000

Employee_Name	Employee_Address	Employee_houseno
Ram	Butwal	23
Sita	Pokhara	15
Laxman	Palpa	67

Fig. 1.7 Relational Model

- **Object-Oriented Model:** Object-oriented model represents data as objects, similar to object-oriented programming, which can handle complex data types and relationships. It is beneficial for applications needing support for advanced data types and relationships.

1.8 Database Schema

A database schema defines the logical structure of the database, describing tables, fields, relationships, and constraints. It serves as a blueprint for organizing data, setting rules for data storage, and ensuring that data is structured consistently.

Types of Database Schemas

1. **Physical Schema:** Physical Schema describes how data is physically stored on storage devices, addressing indexing, partitions, and file structures.
2. **Logical Schema:** Logical Schema defines the tables, columns, and relationships between data without focusing on how they are stored. This schema is critical for data organization and relationships within the database.
3. **View Schema:** View Schema is also known as the external schema, it provides different users with tailored views of the data, simplifying their interaction with the database while maintaining data privacy.

Exercise

Choose the correct answer from the given alternatives.

1. What is the full form of DBMS?

   a. Data of Binary Management System
   b. Database Management System
   c. Database Management Service
   d. Data Backup Management System

2. What is a database?

   a. Organized collection of information that cannot be accessed, updated, and managed
   b. Collection of data or information without organizing
   c. Organized collection of data that can be accessed, updated, and managed
   d. Organized collection of data that cannot be updated

3. Which type of data can be stored in the database?

   a. Image oriented data
   b. Text, files containing data
   c. Data in the form of audio or video
   d. All of the above

4. Which of the following is not an example of DBMS?

   a. MySQL
   b. IBM DB2
   c. Microsoft Access
   d. Google

5. Which of the following is not a feature of DBMS?

   a. Minimum Duplication and Redundancy of Data
   b. High Level of Security
   c. Single-user Access only
   d. Support ACID Property

6. In general, a file is basically a collection of all related_____.

   a. Rows & Columns
   b. Database
   c. Fields
   d. Records

7. How many types of DBMS architectures are there?

   a. 1 b. 2 c. 3 d. 4

8. Basic client-server model is similar to –

   a. 2-tier architecture
   b. 3-tier architecture
   c. 4-tier architecture
   d. 5-tier architecture

9. The term "NTFS" refers to which one of the following?

   a. New Technology File System
   b. New Table type File System
   c. New Tree File System
   d. Both a and b

10. ODBC stands for

   a. Object Database Connectivity
   b. Oral Database Connectivity
   c. Oracle Database Connectivity
   d. Open Database Connectivity

Write short answer to the following questions.

1. Explain DBMS with its advantages and disadvantages.
2. Differentiate between data and information.
3. What is the role of DBMS in modern business?
4. What are types of database users?
5. Describe the limitations of file system management.
6. Describe the application of DBMS.
7. Discuss the three level architecture of database system.
8. Explain the following terms: Data, Information, Database and Database Management System (DBMS).

Write long answer to the following questions.

1. Explain Database architecture with figures with examples?
2. Explain the concept of data, information, database and Database Management System (DBMS).
3. Differentiate between 1-tier, 2-tier and 3-tier architecture.
4. Explain the types of database user with the help of an examples.
5. Explain the client server architecture. Also write advantages and disadvantages of it.
6. Explain Hierarchical, Network and Relational Database model with help of some examples.
7. What is DBMS? What are the advantages and disadvantages offered by such system as compared to file processing system? Explain.

Entity Relationship Model (ER-Model) Unit 2

2.1 Introduction of ER-Model

The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.

The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, the ER diagram is the structural format of the database.

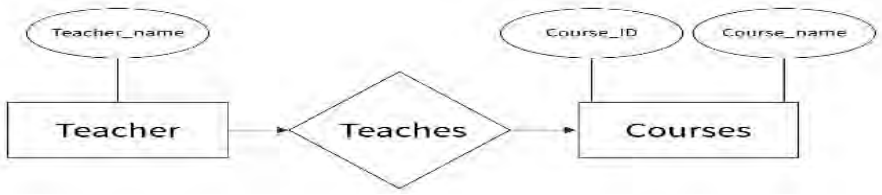


Fig 2.1: An example of ER model

Use of ER Diagram

1. ER diagrams play a vital role in conceptualizing and designing databases by helping to outline the relationships between different data entities clearly.
2. These diagrams can serve as a reference guide or documentation, offering an overview of the system structure and relationships, which can be particularly helpful for new team members or stakeholders to understand the database schema quickly.
3. ER diagrams can be used to identify and solve potential issues in the database schema during the design phase, preventing complications at later stages of development.
4. For maintenance and upgrade purposes, ER diagrams can be used to understand the impact of changes on different components and relationships within the system.
5. These diagrams are very easy to understand and easy to create even for a naive user.
6. It can play a key role in setting up useful database to analyse the data.

2.2 Components of ER-Model

ER Diagrams are composed of entities, relationships and attributes.

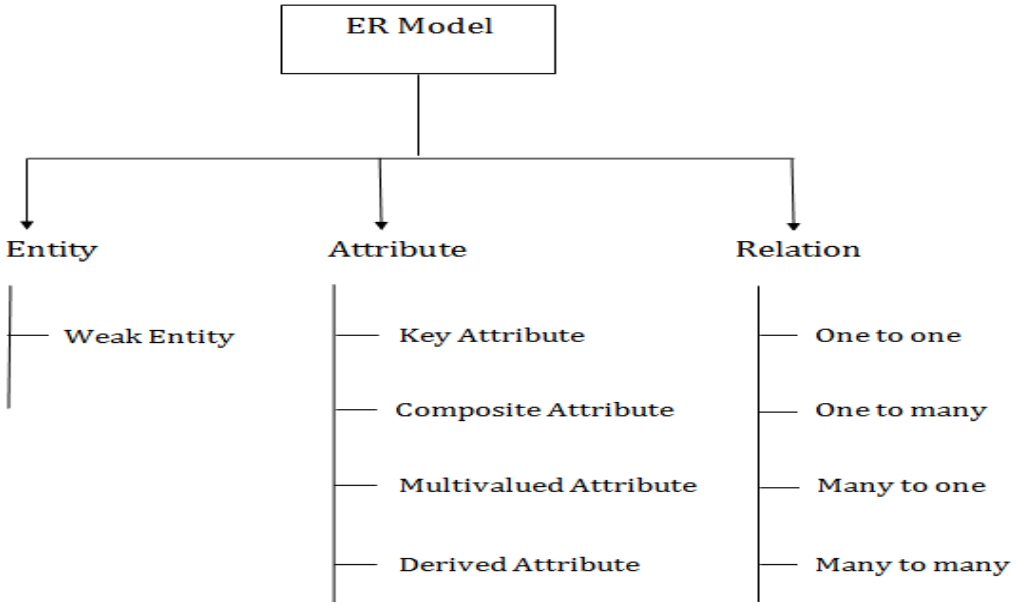


Fig 2.2: Component of ER-model

2.2.1 Entity, Weak Entity, Entity Set

Entity

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

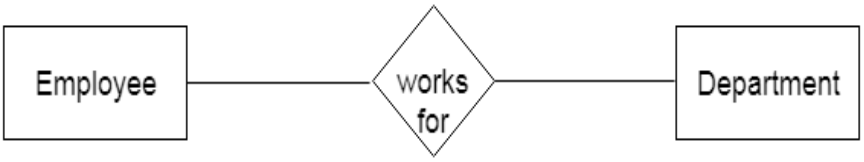


Fig 2.3: Entity

Weak Entity

An entity that depends on another entity is called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.

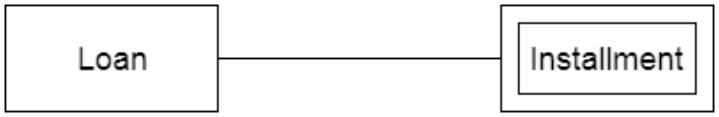


Fig 2.4: Weak Entity

Entity Set

An entity set is a group of entities of the same entity type. For example, an entity set of students, an entity set of motorbikes, an entity of smartphones, an entity of customers, etc.

Entity sets can be classified into two types:

i. Strong Entity Set

In a DBMS, a strong entity set consists of a primary key. For example, an entity of motorbikes with the attributes, motorbike's registration number, motorbike's name, motorbike's model, and motorbike's colour.

ii. Weak Entity Set

In a DBMS, a weak entity set does not contain a primary key. For example, An entity of smartphones with its attributes, phone's name, phone's colour, and phone's RAM.

2.2.2 Attributes, Types of Attributes

The attribsute is used to describe the property of an entity. Eclipse is used to represent an attribute. For example, id, age, contact number, name, etc. can be attributes of a student.

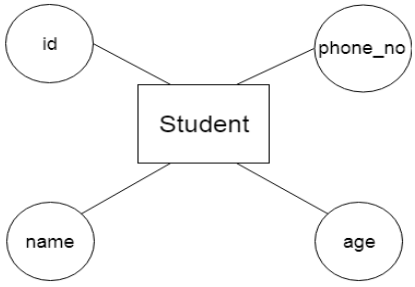


Fig 2.5: Attribute

Types of Attributes

i. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents

a primary key. The key attribute is represented by an ellipse with the text underlined. Roll number can be uniquely identied a student from group of students.

Fig 2.6: Key Attribute

ii. Composite Attribute

An attribute that is composed of many other attributes is known as a composite attribute. For Example,

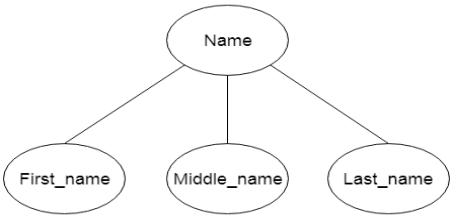


Fig 2.7: Composite Attribute

iii. Multivalued Attribute

Multivalued Attribute is a attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.

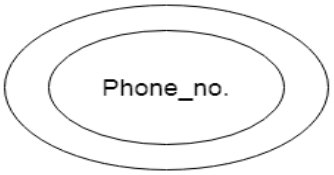


Fig 2.8: Multivalued Attribute

iv. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like date of birth.

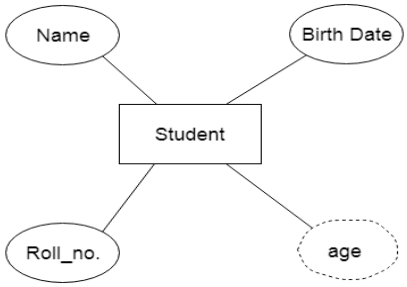


Fig 2.9: Derived Attribute

2.2.3 Relationship, Types of Relationship

Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

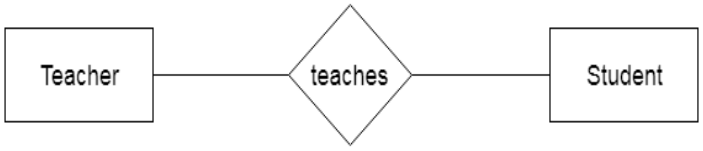


Fig 2.10: Relationship

Types of Relationship

i. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

Fig 2.10: Relationship

ii. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship. For example, scientist can invent many inventions, but the invention is done by the only specific scientist.

Fig 2.11: One-to-one Relationship

iii. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship. For example, student enrolls for only one course, but a course can have many students.

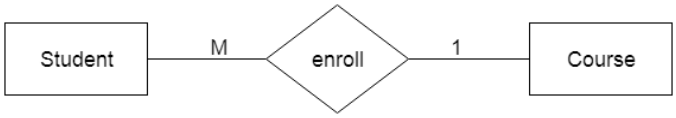


Fig 2.12: Many-to-one Relationship

iv. Many-to-Many Relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship. For example, employee can be assigned by many projects and project can have many employees.

Fig 2.13: Many-to-many Relationship

2.3 Mapping Cardinalities

There are four types of Cardinality Mapping in Database Management Systems:

i. One to one

ii. Many to one

iii. One to many

iv. Many to many

i. One to One Cardinalities

One to one cardinality is represented by a 1:1 symbol. In this, there is at most one relationship from one entity to another entity. There are a lot of examples of one-to-one cardinality in real life databases.

For example, one student can have only one student id, and one student id can belong to

only one student. So, the relationship mapping between student and student id will be one to one cardinality mapping.

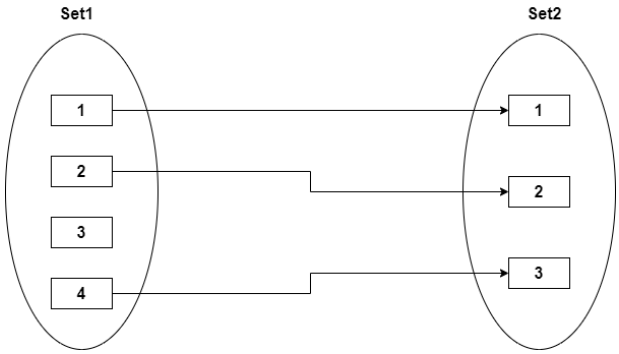


Fig 2.14: One-to-one Cardinality

ii. Many to One Cardinality

In many to one cardinality mapping, from set 1, there can be multiple sets that can make relationships with a single entity of set 2. Or we can also describe it as from set 2, and one entity can make a relationship with more than one entity of set 1.

One to One Cardinality is the subset of Many to One Cardinality. It can be represented by M:1.

For example, there are multiple patients in a hospital who are served by a single doctor, so the relationship between patients and doctors can be represented by Many to one Cardinality.

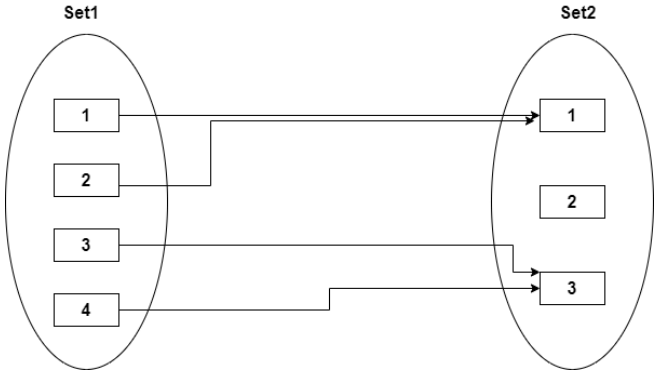


Fig 2.15: Many-to-one Cardinality

iii. One to Many Cardinalities

In One-to-Many Cardinality mapping, from set 1, there can be a maximum single

set that can make relationships with a single or more than one entity of set 2. Or we can also describe it as from set 2, more than one entity can make a relationship with only one entity of set 1.

One to One Cardinality is the subset of One-to-Many Cardinality. It can be represented by 1: M.

For example, in a hospital, there can be various compounders, so the relationship between the hospital and compounders can be mapped through One-to-Many Cardinality.

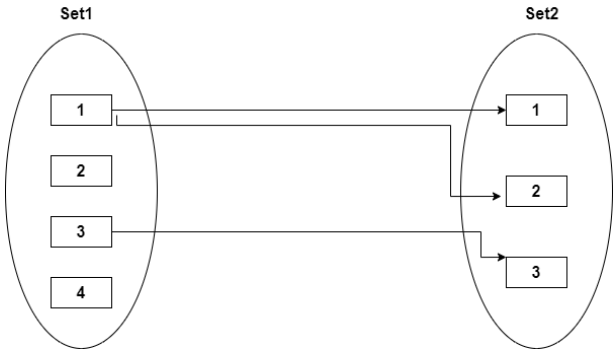


Fig 2.16: One-to-many Cardinality

iv. Many to Many Cardinalities

In Many to Many cardinalities mapping, there can be one or more than one entity that can associate with one or more than one entity of set 2. In the same way from the end of set 2, one or more than one entity can make a relation with one or more than one entity of set 1.

It is represented by M: N or N: M.

One to One Cardinality, One to Many Cardinalities, and Many to one cardinality is the subset of the many to many cardinalities. For example, in a college, multiple students can work on a single project, and a single student can also work on multiple projects. So, the relationship between the project and the student can be represented by Many to Many Cardinalities.

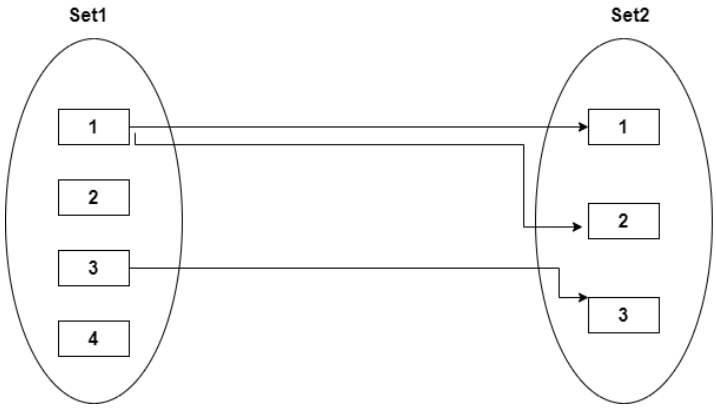


Fig 2.17: Many-to-many Cardinality

2.4 Keys in DBMS

Different types of database keys

1. Primary Key

The primary key refers to a column or a set of columns of a table that helps us identify all the records uniquely present in that table. A table can consist of just one primary key. Also, this primary key cannot consist of the same values reappearing/ repeating for any of its rows. All the values of a primary key have to be different, and there should be no repetitions.

The PK (Primary Key) constraint that we put on a column/set of columns won't allow these to have a null value or a duplicate. Any table can consist of only a single primary key constraint. A foreign key that refers to it can never change the values present in the primary key.

2. Super Key

A super key refers to the set of all those keys that help us uniquely identify all the rows present in a table. It means that all of these columns present in a table that can identify the columns of the table uniquely act as the super keys.

A super key is a candidate key's superset (candidate key has been explained below). We need to pick the primary key of any table from the super key's set so as to make it the table's identity attribute.

3. Candidate Key

The candidate keys refer to those attributes that identify rows uniquely in a table. In a table, we select the primary key from a candidate key. Thus, a candidate key has similar properties as that of the primary keys that we have explained above. In a table, there can be multiple candidate keys.

4. Alternate Key

As we have stated above, any table can consist of multiple choices for the primary key. But, it can only choose one. Thus, all those keys that did not become a primary key are known as alternate keys.

5. Foreign Key

We use a foreign key to establish relationships between two available tables. The foreign key would require every value present in a column/set of columns to match the referential table's primary key. A foreign key helps us to maintain data as well as referential integrity.

6. Composite Key

The composite key refers to a set of multiple attributes that help us uniquely identify every tuple present in a table. The attributes present in a set may not be unique whenever we consider them separately. Thus, when we take them all together, it will ensure total uniqueness.

Exercise

Choose the correct answer from the given alternatives.

1. An _____ is a set of entities of the same type that share the same properties, or attributes.

   a. Entity set
   b. Relation set
   c. Attribute set
   d. Entity model

2. Entity is a _____

   a. Object of relation
   b. Thing in real world
   c. Present working model
   d. Model of relation

3. The number of entities to which another entity can be related through a relationship set is called.....

   a. Cardinality
   b. Entity
   c. Schema
   d. Attributes

4. Select the attributes which made up of more than one single attribute.....

   a. Multi-value attribute
   b. Single value attribute
   c. Derived attribute
   d. Composite attribute

5. If two entities have many to many relationships mostly results in how many tables...

   a. Three c. One b. Two d. Four

6. Select from the following the multi-valued attribute.....

   a. Date of birth
   b. Name
   c. Phone number
   d. All of the above

7. The attribute AGE is calculated from DATE_OF_BIRTH. The attribute AGE is.....

   a. Single valued c. Composite b. Multi value d. Derived

8. Which of the following is a single valued attribute?

   a. Register number
   b. Address
   c. Subject taken
   d. Reference

9. The attribute name could be structured as an attribute consisting of first name, middle name, and last name. This type of attribute is called.....

   a. Simple attribute
   c. Multivalued attribute
   b. Composite attribute
   d. Derived attribute

10. The primary key in a many to one relationship, acts as a foreign key on which side...

   a. On the side where many relationships are defined

   b. On the side where a single (one) relationship is defined

   c. On both the sides

   d. Neither of them

Write short answer to the following questions.

1. What do you mean by Entity-Relationship Diagram? Explain
2. Define the term entity and give an example.
3. What are the components of the ER-model?
4. Define the attribute and its type with suitable example.
5. What is derived attribute? Explain it with an appropriate example. 6. Define relationship and give an example.
6. Define degree of relationship.
7. What is key? Explain the major features of primary key.

Write long answer to the following questions.

1. What is key? Explain the major features of primary key.
2. Explain the distinctions among the terms primary key, candidate key and super key.
3. What are the major components of E-R model? Discuss each of them.
4. Explain the difference between single valued attribute and multivalued attribute. Brifly.
5. Explain the differences between a weak and a strong entity set.
6. Explain the various notations used in Entity-Relationship model.
7. Differentiate between primary key and foreign key with the help of suitable example.

D. Project Work

1. Design an E-R diagram for a library system that tracks the books and members.
2. Create an E-R diagram for a school management system that keeps track of students, teachers, and courses.
3. Design an E-R diagram for a restaurant management system that tracks men. items, orders, and customers.
4. Create an E-R diagram for a medical clinic that tracks patients, appointments and medical records.

Unit 3

Relational Model

3.1 Introduction to Relational Model

The relational model represents how data is stored in relational databases. A relational database consists of a collection of tables, each of which is assigned a unique name. The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-based model.

Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE, and AGE shown in the table.

Table Student

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	Thimi	9455123451	18
2	RAMESH	Bhaktapur	9652431543	18
3	SUJIT	Kathmandu	9156253131	20
4	SURESH	Lalitpur	9812345678	18

Merits of Relational Model

Following are the various merits of relational model:

1. This provides an abstract view of the data. It abstracts the physical structure from the logical structure of data.
2. This model is very easy to design. Tables can use different attributes as per requirements.
3. The relational model supports data independence. In a relational database the data is stored in tables so that we can modify the data without changing the physical structure.

4. Relational database helps the user to use a query language to query the database.
5. It offers more flexibility than other models.
6. It is useful for representing most real world objects and the relationships between them.

Demerits of Relational Model

Following are the various demerits of relational model:

1. The main disadvantage of relational models is that they do not support binary data for example: images, documents, spreadsheets etc.
2. The relational model can easily adapt to new hardware so incurs large hardware overhead.
3. Relational databases use a simple mapping of logical tables to physical structures.
4. Enforcing data integrity in relational models is difficult because no single piece of hardware has control over the data.
5. The relational model is suitable for small databases but not suitable for complex databases because the user needs to know the complex physical data storage details.

Relational Model Terminologies

Following are the terminologies of Relational Model:

Relation	Table
Tuple	Row, Record
Attribute	Column, Field
Domain	It consists of set of legal values
Cardinality	It consists of number of rows
Degree	It contains number of columns

3.2 Key Concept in Relational Model

Let's explain each term one by one in detail with the help of example.

Example: STUDENT Relation

Stu_No	S_Name	PHONE_NO	ADDRESS	Gender
10112	Rama	9874567891	Kathmandu	F
12839	Shyam	9026288936	Lalitpur	M

33289	Laxman	8583287182	Bhaktapur	M
27857	Mahesh	7086819134	Bhaktapur	M
17282	Ganesh	9028939884	Pokhara	M

1. **Relation:** A table with rows and columns in a database. Each row represents a record. Example: A "Student" table with columns like ID, Name, and Address.
2. **Tuple:** A single row in a table, representing a specific record. Example: A row with values (10112, Rama, 9874567891, Kathmandu, F).
3. **Data Item:** The smallest unit of data in a table, found at the intersection of a row and column. Example: 10112 or "Rama".
4. **Domain:** The set of allowed values for a column. Example: The "Gender" column can only contain "M" or "F".
5. **Attribute:** A column in a table, representing a specific piece of data. Example: Columns like Stu_No, S_Name, and PHONE_NO.
6. **Cardinality:** The total number of rows in a table at a specific time. Example: If there are 3 rows in the "Student" table, the cardinality is 3.
7. **Degree:** The total number of columns in a table. Example: If a table has 5 columns, its degree is 5.
8. **Relational Instance:** A specific set of rows in a table at a given time, with no duplicate rows.
9. **Relational Schema:** A blueprint of a table, defining its name and column names.
10. **Relational Key:** A column or combination of columns that uniquely identify a row in a table.

3.3 Properties of Relations

1. Each attribute in a relation has only one data value corresponding to it i.e. they do not contain two or more values.
2. Name of the relation is distinct from all other relations.
3. Each relation cell contains exactly one atomic (single) value
4. Each attribute contains a distinct name
5. Attribute domain has no significance

6. tuple has no duplicate value
7. Order of tuple can have a different sequence
8. It also provides information about metadata

3.4 Mapping ER-Model to Relational Model

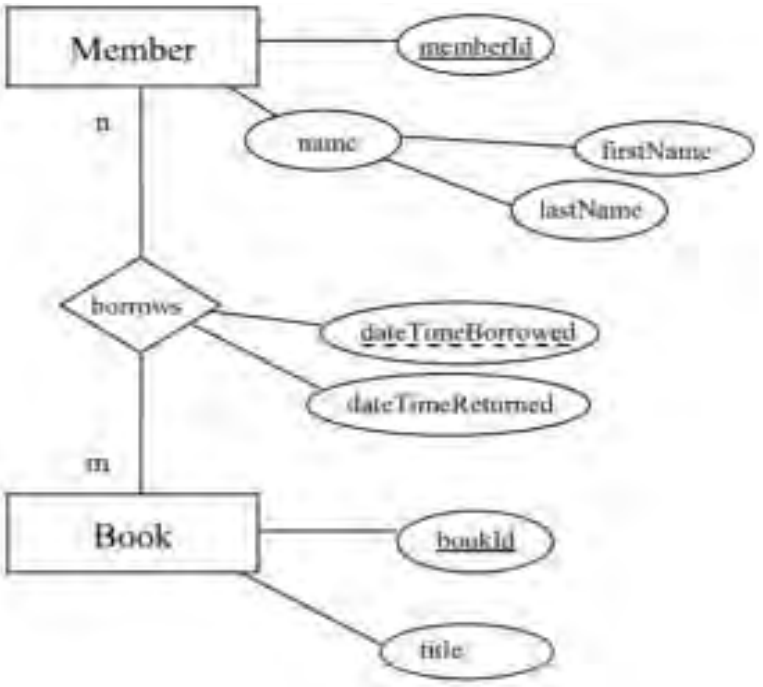
Converting an Entity-Relationship Diagram (ERD) into a Relational Model means turning the design of a database into a format that can be used in software like MySQL or Oracle. Each entity in the ERD becomes a table, and its attributes (properties) become the columns of that table. The unique identifier of the entity, called the primary key, is used to identify each record in the table.

For relationships, if one entity is connected to many others (one-to-many), the primary key of the "one" side is added as a column (foreign key) in the table of the "many" side. For many-to-many relationships, we create a new table to store the connection, which includes the primary keys of both related entities. If two entities have a one-to-one relationship, we add a column for the foreign key in either of the two tables.

If an entity has properties with multiple values (like a person with multiple phone numbers), we create a separate table for those values and link it back to the main table. Rules like primary keys and foreign keys help keep the database accurate and connected. This process helps us turn the visual ERD into a working database that can store and manage information.

Example 1

Consider the ERD Diagram



The mapping rules lead to the relations:

Book	
bookId	title

Member		
memberId	firstName	lastName

Borrow			
memberId	**bookId**	**dateTimeBorrowed**	dateTimeReturned

Mapping Process

- Create table for an entity and relationship.
- Add the primary keys of all participating entities as fields of table with their respective data types.
- If relationship has any attribute, add each attribute as field of table.
- Declare a primary key composing all the primary keys of participating entities.
- Declare all foreign key constraints

Exercise

Choose the correct answer from the given alternatives.

1. Who is credited with the creation of the relational database model?

   a. Edgar F. Codd b. Charles Bachman

   c. Larry Ellison d. James Gosling

2. In the relational model, data is organized into:

   a. Trees b. Graphs c. Tables d. Arrays

3. What is the primary key in a relational database?

   a. A unique identifier for each row in a table.

   b. A non-unique column.

   c. A key that can have null values.

   d. A foreign key.

4. In a relational database, what is a foreign key?

   a. A key that uniquely identifies a row

   b. A key that points to a primary key in another table

   c. A non-unique identifier

   d. A key that does not allow null values

5. Which language is most commonly used to interact with relational databases?

   a. C++ b. Python c. SQL d. Java

6. In the relational model, a row is also known as a:

   a. Column b. Tuple c. Attribute d. Key

7. In the relational model, a column is also known as a:

   a. Tuple b. Relation c. Attribute d. Domain

8. What is a relation in the context of the relational database model?

   a. A table with rows and columns b. A function that maps keys to values

   c. A hierarchy of data d. A network of connected nodes

9. Which of the following is a characteristic of a relational database table?

   a. Each column has a unique name.

   b. Each row must have a unique key.

   c. Columns can have different data types.

   d. All of the above

10. What is a foreign key?

   a. A foreign key is a primary key of a relation which is an attribute in another relation

   b. A foreign key is a super key of a relation which is an attribute in more than one other relations

   c. A foreign key is an attribute of a relation that is a primary key of another relation

   d. A foreign key is the primary key of a relation that does not occur anywhere else in the schema

Write short answer to the following questions.

1. What is the relational model?
2. What is a table in the relational model?
3. What is a tuple in the relational model?
4. What is an attribute in the relational model?
5. What is a relational schema?
6. What is a relational key in the relational model?
7. What are the properties of relations in the relational model?
8. What is the process of mapping an ER model to a relational model?

Write long answer to the following questions.

1. What is relation? Differentiate between a relation schema and a relation instance.
2. What are the major properties of the relation? Explain
3. Write down the steps for mapping ER-Model to relational model with an example.
4. What are the key concepts in the relational model, and how do they differ from other database models?
5. What is the role of tables in the relational model, and how do they represent data in

a relational database?

6. Explain the relational model with the help of suitable example.

7. Write notes on the following topic:

   a. Attribute, Degree and Domain

   b. Tuple, Cardinality and Column

   c. Relational Instance and Relational Schema.

D. Project Work

1. A company has an ER model for its employee database that includes entities for employees, departments, and projects. How would you convert this ER model into a relational model, including steps for determining primary keys, defining relationships, and normalizing the data?

2. A large retail company has an ER model for its customer database that includes entities for customers, orders, and products. How would you convert this ER model into a relational model, and what steps would you take to ensure data consistency and integrity in the resulting relational database?

Unit 4 Structure Query Language Overview

4.1 Introduction to SQL

SQL (Structured Query Language) is used for managing data in relational databases. It enables users to create, read, update, and delete data in tables. SQL is supported by various RDBMS like MySQL, Oracle, and SQL Server. It allows users to query databases using simple, English-like statements, enabling efficient data retrieval and modification. Instead of specifying how operations should be performed, SQL defines what data is needed, with the DBMS handling the execution.

Uses of SQL

1. SQL can execute queries against a database.
2. SQL can retrieve data from a database.
3. SQL can insert records in a database.
4. SQL can update records in a database.
5. SQL can delete records from a database.
6. SQL can create new databases.
7. SQL can create new tables in a database.
8. SQL can create views in a database.

Advantages of SQL

SQL has many advantages which makes it popular and highly demanded. It is a reliable and efficient language used for communicating with the database. Some advantages of SQL are as follows:

1. **Faster Query Processing:** Large amount of data is retrieved quickly and efficiently. Operations like insertion, deletion, manipulation of data is also done in almost no time.

2. **No Coding Skills:** For data retrieval, large number of lines of code is not required. All basic keywords such as SELECT, INSERT INTO, UPDATE, etc are used and also the syntactical rules are not complex in SQL, which makes it a user-friendly language.

3. **Standardized Language:** Due to documentation and long establishment over years, it provides a uniform platform worldwide to all its users.

4. **Portable: I**t can be used in programs in PCs, server, laptops independent of any platform (Operating System, etc). Also, it can be embedded with other applications as per need/requirement/use.

5. **Interactive Language:** It is easy to learn and understand, answers to complex queries can be received in seconds.

Disadvantages of SQL

Although SQL has many advantages, still there are a few disadvantages.

1. Complex Interface: SQL has a difficult interface that makes few users uncomfortable while dealing with the database.

2. Cost: Some versions are costly and hence, programmers cannot access it.

3. Limited Flexibility: SQL databases are less flexible than NoSQL databases when it comes to handling unstructured or semi-structured data, as they require data to be structured into tables and columns.

4. Lack of Real-Time Analytics: SQL databases are designed for batch processing and do not support real-time analytics, which can be a disadvantage for applications that require real-time data processing.

4.2 Types of SQL

SQL Commands are mainly categorized into five categories:

DDL – Data Definition Language

DML – Data Manipulation Language

DCL – Data Control Language

DDL (Data Definition Language)

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc. All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are the DDL (Data Definition Language) commands along with their syntax:

Command	Description	Syntax
CREATE	Create database or its objects (table, index, function, views, store procedure, and triggers)	CREATE TABLE table_name (column1 data_type, column2 data_type, ...);
DROP	Delete objects from the database	DROP TABLE table_name;
ALTER	Alter the structure of the database	ALTER TABLE table_name ADD COLUMN column_name data_type;
TRUNCATE	Remove all records from a table, including all spaces allocated for the records are removed	TRUNCATE TABLE table_name;
RENAME	Rename an object existing in the database	RENAME TABLE old_table_name TO new_table_name;

DML (Data Manipulation Language)

DML commands are used to modify the database. It is responsible for all form of changes in the database. The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are the main DML (Data Manipulation Language) commands along with their syntax:

Command	Description	Syntax
SELECT	It is used to retrieve data from the database	SELECT column1, column2, ...FROM table_name WHERE condition;
INSERT	Insert data into a table	INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);
UPDATE	Update existing data within a table	UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;
DELETE	Delete records from a database table	DELETE FROM table_name WHERE condition;

DCL (Data Control Language)

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

Two important DCL commands and their syntax are:

Command	Description	Syntax
GRANT	Assigns new privileges to a user account, allowing access to specific database objects, actions, or functions.	GRANT privilege_type [(column_list)] ON [object_type] object_name TO user [WITH GRANT OPTION];
REVOKE	Removes previously granted privileges from a user account, taking away their access to certain database objects or actions.	REVOKE [GRANT OPTION FOR] privilege_type [(column_list)] ON [object_type] object_name FROM user [CASCADE];

Important SQL Commands

Some of the most important SQL commands that you are likely to use frequently include:

1. SELECT: Used to retrieve data from a database.
2. INSERT: Used to add new data to a database.
3. UPDATE: Used to modify existing data in a database.
4. DELETE: Used to remove data from a database.
5. CREATE TABLE: Used to create a new table in a database.
6. ALTER TABLE: Used to modify the structure of an existing table.
7. DROP TABLE: Used to delete an entire table from a database.
8. WHERE: Used to filter rows based on a specified condition.
9. ORDER BY: Used to sort the result set in ascending or descending order.
10. JOIN: Used to combine rows from two or more tables based on a related column between them.

4.3 SQL Clause

A clause in SQL is a built-in function that helps to fetch the required records from a database table. A clause receives a conditional expression, i.e. a column name or some terms involving the columns. The clause calculates the result based on the given statements in the expression. When a large amount of data is stored in the database, clauses are helpful to filter and analyze the queries. SQL clauses are

essential components of SQL queries that help filter, sort, and manipulate data stored in databases. They allow users to specify conditions, patterns, and limits to retrieve or modify specific data. There are different types of clauses in SQL that are available for fetching the desired data, and these are mentioned below:

1. With clause

WITH clause acts as a temporary view as it is available only during the execution of SELECT, UPDATE, INSERT, DELETE, etc. statements. It is used to simplify complex and long queries.

2. OR clause

The OR clause is used when multiple conditions are specified in a query and returns a dataset when one of those conditions gets satisfied.

3. AND clause

The AND clause is used when multiple conditions are specified in a query and returns a dataset when all the conditions given in the AND clause meet the requirements.

4. WHERE clause

The WHERE clause in SQL is used to retrieve the specific data from the database that specifies the conditions exactly that are given in the UPDATE, DELETE, etc. statements.

5. ORDER BY clause

The ORDER BY clause in SQL is used for sorting the records of the database tables.

4.4 SQL Join

SQL JOIN clause is used to query and access data from multiple tables by establishing logical relationships between them. It can access data from multiple tables simultaneously using common key values shared across different tables.

Types of JOIN in SQL

There are many types of Joins in SQL. Depending on the use case, you can use different type of SQL JOIN clause.

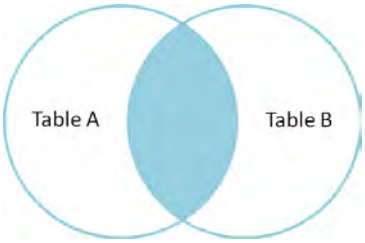
Here are the frequently used SQL JOIN types:

- INNER JOIN

- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

1. SQL INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e. value of the common field will be the same.



Syntax

The syntax for SQL INNER JOIN is:

SELECT table1.column1, table1.column2, table2.column1, ...

FROM table1

INNER JOIN table2

ON table1.matching_column = table2.matching_column;

INNER JOIN Example

This query will show the names and age of students enrolled in different courses.

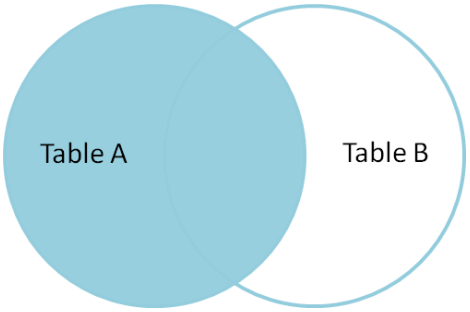
SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE FROM Student

INNER JOIN StudentCourse

ON Student.ROLL_NO = StudentCourse.ROLL_NO;

2. SQL LEFT JOIN

LEFT JOIN returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.



Syntax:

The syntax of **LEFT JOIN** in SQL is:

SELECT table1.column1, table1.column2, table2.column1, ...

FROM table1

LEFT JOIN table2

ON table1.matching_column = table2.matching_column;

LEFT JOIN Example

SELECT Student.NAME,StudentCourse.COURSE_ID

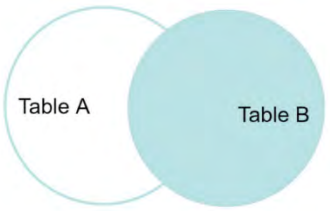
FROM Student

LEFT JOIN StudentCourse

ON StudentCourse.ROLL_NO = Student.ROLL_NO;

3. SQL RIGHT JOIN

RIGHT JOIN returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. It is very similar to LEFT JOIN For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.



Syntax

The syntax of **RIGHT JOIN** in SQL is:

SELECT table1.column1, table1.column2, table2.column1, ...

FROM table1

RIGHT JOIN table2

ON table1.matching_column = table2.matching_column;

RIGHT JOIN Example

SELECT Student.NAME,StudentCourse.COURSE_ID

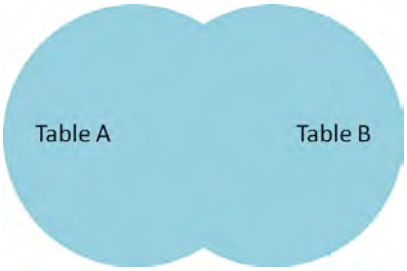
FROM Student

RIGHT JOIN StudentCourse

ON StudentCourse.ROLL_NO = Student.ROLL_NO;

4. SQL FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.



Syntax

The syntax of SQL FULL JOIN is

SELECT table1.column1, table1.column2, table2.column1, ...

FROM table1

FULL JOIN table2

ON table1.matching_column = table2.matching_column;

FULL JOIN Example

SELECT Student.NAME,StudentCourse.COURSE_ID

FROM Student

FULL JOIN StudentCourse

ON StudentCourse.ROLL_NO = Student.ROLL_NO;

4.5 SQL VIEW

Views in SQL are a type of virtual table that simplifies how users interact with data across one or more tables. Unlike traditional tables, a view in SQL does not store data on disk; instead, it dynamically retrieves data based on a pre-defined query each time it's accessed.

SQL views are particularly useful for managing complex queries, enhancing security, and presenting data in a simplified format. In this guide, we will cover the SQL create view statement, updating and deleting views, and using the WITH CHECK OPTION clause.

CREATE VIEWS in SQL

We can create a view using CREATE VIEW statement. A View can be created from a single table or multiple tables.

Syntax

CREATE VIEW view_name

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Exercise

Choose the correct answer from the given alternatives.

1. What does SQL stand for?

   a. Structured Query Language
   c. Simple Query Language
   b. Standard Query Language
   d. Sequential Query Language

2. Which of the following is NOT a valid keyword in SQL?

   a. SELECT
   c. WHERE
   b. DELETE
   d. INCLUDE

3. Which of the following operators is used to compare two values in SQL?

   a. + c. <> b. = d. &

4. What is the purpose of the SQL keyword “DISTINCT” in a SELECT statement?

   a. To retrieve unique values from a column
   b. To filter NULL values
   c. To delete duplicate records
   d. To sort the result set

5. Which of the following tasks CANNOT be accomplished using SQL?

   a. Creating and modifying database structures
   b. Writing complex algorithms for data analysis
   c. Retrieving specific data from a database
   d. Adding new data to a database

6. Which of the following is basis for SQL?

   a. SQL Server b. DBMS c. RDBMS d. Oracle

7. Which character is used to separate SQL statements in database systems?

   a. : c. _ b. % d. ;

8. Which statement(s) are mandatory in a simple SQL SELECT statement?

   a. Select, From
   c. Select, Where
   b. Select, OrderBy
   d. Select, GroupBy

9. Which of the following is a default join type?

a. Right join
c. Inner join
b. Left join
d. Outer join

10. Which of the following statement is true about views in SQL?

a. We can delete but not insert rows in a view
b. We cannot insert and delete rows in a view
c. We can insert but not delete rows in a view
d. We can insert and delete rows in a view

11. Which of the following SQL statement selects only unique values from 'section' column of table 'school'?

a. SELECT section FROM school;
b. SELECT DISTINCT section FROM school;
c. SELECT * FROM school;
d. SELECT ALL section FROM school;

Write short answer to the following questions.

1. What are the different categories of SQL commands?
2. Write the importance characteristics of SQL.
3. List out the aggregate function.
4. Difference between DROP and DELETE COMMAND.
5. Write down the syntax of INSERT and UPDATE command with suitable example.
6. What does the function of SELECT, FROM and WEHRE command?
7. Write down the syntax and example of UPDATE statement.
8. What are the usages of SQL?
9. What is the purpose of DDL Language?
10. What is the purpose of DML Language?
11. Differentiate between the ALTER and UPDATE command.

Write long answer to the following questions.

1. What is Data Definition Language (DDL) in SQL and what are its main commands?

2. Explain DML with its commands.
3. What is data control language (DCL) in SQL and what are its main commands?
4. Briefly describe SQL joins and why they are used in SQL?
5. Discuss SQL clauses and what is their purpose in SQL?

D. Project Work

1. Write DDL code to create a table named "customers" with columns: (varchar), and "country" (varchar). "customer_id" (integer), "customer name" (varchar), "contact_name"
2. Write DML code to insert five records into the "customers" table with the following values:
    a. customer_id=1, company_name="ABC Company", contact_name="RAM", address="Thimi"
    b. customer_id=2, company_name="XYZ Inc.", contact_name="Anil", address="Bhaktapur"
    c. customer_id=3, company_name="World Coporation", contact_name="Hari", address="Lalitpur"
    d. customer_id=4, company_name="Dolls House", contact_name="Salina", address="Kathmandu".
    e. customer_id=5, company_name="Lakoul Industry", contact_name="Garima", address="Thimi"

Answer the following questions with the help of above information.

i. Write DML code to update the record in the "customers' table with customer_id=3 and set the value of the "address" column to "Kathmandu".
ii. Write DML code to delete the record in the "customers" table with customer_id=5.
iii. Write SELECT and WHERE clauses to select all records from the "customers" table where the address is "Thimi".
iv. Write SELECT clauses to select all records from the "customers" table and order them by customer_name in ascending order.
v. Select the record where customer_id=3.

Unit 5 Relational Design DataBase

5.1 Functional Dependency and its Type

A functional dependency occurs when one attribute uniquely determines another attribute within a relation. It is a constraint that describes how attributes in a table relate to each other. If attribute A functionally determines attribute B we write this as the A→B.

Functional dependencies are used to mathematically express relations among database entities and are very important to understanding advanced concepts in Relational Database Systems.

For example

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.

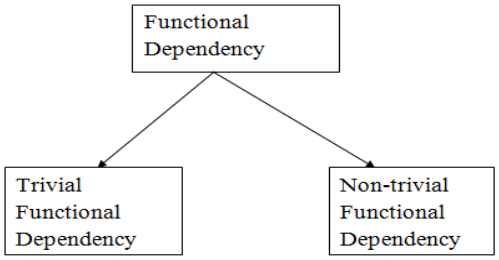
Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

Emp_Id → Emp_Name

We can say that Emp_Name is functionally dependent on Emp_Id.

Types of Functional Dependency



1. Trivial Functional Dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B

- _Name are trivial dependencies too.

2. Non-trivial Functional Dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.

Properties and Axiomatization of Functional Dependencies

- Reflexivity: If Y is a subset of X, then X → Y.
- Augmentation: If X → Y, then XZ → YZ.
- Transitivity: If X → Y and Y → Z, then X → Z.

5.2 Normalization

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

First Normal Form – 1NF

For a table to be in the first normal form, it must meet the following criteria:

- A single cell must not hold more than one value (atomicity).
- There must be a primary key for identification.
- No duplicated rows or columns.
- Each column must have only one value for each row in the table.

Example: Before 1NF

Employee_ID	E_Name	Phone_Number
001	Ram	1234567890, 2347895014
002	John	1122334455

After 1NF:

Employee_ID	E_Name	Phone_Number
001	Ram	1234567890
001	Ram	2347895014
002	John	1122334455

Second Normal Form – 2NF

A table is said to be in 2NF if it meets the following criteria:

- It's already in 1NF
- It has no partial dependency i.e. all non-key attributes are fully dependent on a primary key.

Example: Before 2NF

Employee_ID	Dept_ID	E_Name	Dept_Name
001	10	Ram	HR
002	20	John	IT

Here, **Dept_Name** is only dependent on **Dept_ID**, not on the whole primary key (**Employee_ID, Dept_ID**). This is a partial dependency.

After 2NF

- Creates a separate table for Department
- **Employee Table:** Store Employee_ID, E_Name and Dept_ID
- **Department Table:** Store Dept_ID and Dept_Name

Employee Table:

Employee_ID	Dept_ID	E_Name
001	10	Ram
002	20	John

Department Table:

Dept_ID	Dept_Name
10	HR
20	IT

Third Normal Form – 3NF

A table is said to be in 3NF if it meets the following criteria:

- It's already in 2NF
- There are no transitive dependencies. This means that non-key attributes should not depend on other non-key attributes. Every non-key attributes must depend only on the primary key, not on any other non-key attributes

Example: Before 3NF:

Employee_ID	E_Name	Dept_ID	Dept_Name	Dept_Head	Dept_Location
001	Ram	10	HR	Sarah	Building A
002	John	20	IT	Mike	Building B

Here, **Dept_Head** depends on **Dept_ID**, not directly on the primary key **Employee_ID**. This is a transitive dependency.

After 3NF

Employee Table:

Employee_ID	E_Name	Dept_ID
001	Ram	10
002	John	20

Department Table:

Dept_ID	Dept_Name	Dept_Info_ID
10	HR	D1
20	IT	D2

Department_Info Table

Dept_Info_ID	Dept_Head	Dept_Location
D1	Sarah	Building A
D2	Mike	Building B

Difference between 1NF and 2NF

1NF	2NF
In order to be in 1NF any relation must be atomic and should not contain any composite or multi-valued attributes.	In order to be in 2NF any relation must be in 1NF and should not contain any partial dependency.
First Normal form only deals with the schema of the table and it does not handle the update anomalies.	Second normal form handles the update anomalies.

A relation in 1NF may or may not be in 2NF.	A relation in 2NF is always in 1NF.
The primary key in case of first normal form can be a composite key.	The primary key in case of second normal form cannot be a composite key in case it arises any partial dependency.
The main goal of first normal form is to eliminate the redundant data within the table.	The main goal of second normal form is to actually ensure the data dependencies.

Difference between 2NF and 3NF

2NF (Second Normal Form)	**3NF (Third Normal Form)**
1. It is already in 1NF.	1. It is already in 1NF as well as in 2NF also.
2. In 2NF non-prime attributes are allowed to be functionally dependent on non-prime attributes.	2. In 3NF non-prime attributes are only allowed to be functionally dependent on Super key of relation.
3. No partial functional dependency of non-prime attributes are on any proper subset of candidate key is allowed.	3. No transitive functional dependency of non-prime attributes on any super key is allowed. .
4. Stronger NF than 1NF but lesser than 3NF	4. Stronger normal form than 1NF and 2NF.
5. It eliminates repeating groups in relation.	5. It virtually eliminates all the redundancies.

Difference between 1NF and 3NF

1NF	**3NF**
1. In order to be in 1NF any relation must be atomic and should not contain any composite or multi-valued attributes.	1. In order to be in 3NF there should be no transitive dependency that is no non prime attribute should be transitively dependent on the candidate key.
2. 1NF is considered less stronger normal form.	2. 3NF is considered as a stronger normal form than the 1NF.
3. 1NF contains candidate keys which automatically comply with 2NF.	3. 3NF form will require decomposing a table that is in the 2NF or 1NF.
4. It eliminate duplicate columns from the same table.	4. It remove columns that are not dependent upon the primary key.
5. There are No Composite Attributes.	5. There are No Transitive Functional Dependencies.

Exercise

Choose the correct answer from the given alternatives.

1. A ___ is normalized after it has been organized.

   a. Table
   c. Row
   b. Database
   d. Column

2. Redundancy is reduced in a database table by using the ___ form.

   a. Abnormal
   c. Special
   b. Normal
   d. None

3. In practical applications, how many types of Normal Forms are there?

   a. 3
   c. 5
   b. 4
   d. 6

4. Which of the following is not a type of Normal Form?

   a. 1NF
   c. 3NF
   b. 2NF
   d. 10NF

5. When a relation contains an atomic value, it is a ___ relation.

   a. 1NF
   c. 3NF
   b. 2NF
   d. BCNF

6. 2NF relations are those that are in 1NF with all the attribute types dependent on the ___ key.

   a. Primary
   c. Composite
   b. Foreign
   d. Alternate

7. When a relation is in 2NF and there is ___, it is in 3NF.

   a. Transition Dependency
   b. No Transition Dependency
   c. Relational Dependency
   d. No Relational Dependency

8. Two attributes can be functionally ___ on/of each other.

   a. Dependent
   b. Independent
   c. Both A and B
   d. None of the above

9. How many types of functional dependencies are there?

   a. 1
   c. 3
   b. 2
   d. 4

10. Which of the following is a type of functional dependency?

    a. Trivial Functional Dependency

    b. Non-trivial Functional Dependency

    c. Both A and B

    d. None of the above

11. Which of the following is a trivial functional dependency?

    a. A->B, if B is a subset of A

    b. A->A

    c. B->B

    d. All of the above

Write short answer to the following questions.

1. What is normalization? List out different normal forms and explain any two of them.
2. Compare 2NF and 3NF normal forms.
3. Briefly explain how to normalize a database from unnormalized form to INF, 2NF and 3 NF.
4. What is the importance of normalization?
5. Define functional dependency. Explain partial and transitive functional dependency with suitable example.
6. Write down the advantages of normalization.
7. Consider the following Student table. Convert this table into the third normal form.

   STUD_ID, STUD_NAME, STUD_STATE, STUD_COUNTRY, STUD_AGE
8. Explain the purpose of normalization.

Write long answer to the following questions.

1. What is the functional dependency? Explain the types of functional dependency with an example.
2. What are the advantage of normalization of database? Explain 1 NF, 2NF and 3NF.
3. Write short notes on the following:

   a. Trivial Functional Dependency

   b. Non-Trivial Functional Dependency

   c. 3NF

DataBase Transaction

Unit 6

6.1 Introduction to Transaction

In Database Management Systems (DBMS), a transaction is a fundamental concept representing a set of logically related operations executed as a single unit. Transactions are essential for handling user requests to access and modify database contents, ensuring the database remains consistent and reliable despite various operations and potential interruptions.

Operations of Transaction

A user can make different types of requests to access and modify the contents of a database. So, we have different types of operations relating to a transaction. They are discussed as follows:

i) Read

A read operation is used to read the value of **X** from the database and store it in a buffer in the main memory for further actions such as displaying that value.

Such an operation is performed when a user wishes just to see any content of the database and not make any changes to it. For example, when a user wants to check his/her account's balance, a read operation would be performed on user's account balance from the database.

ii) Write(X)

A write operation is used to write the value to the database from the buffer in the main memory. For a write operation to be performed, first a read operation is performed to bring its value in buffer, and then some changes are made to it, e.g. some set of arithmetic operations are performed on it according to the user's request, then to store the modified value back in the database, a write operation is performed.

For example, when a user requests to withdraw some money from his account, his account balance is fetched from the database using a read operation, then the amount

to be deducted from the account is subtracted from this value, and then the obtained value is stored back in the database using a write operation.

iii) Commit

This operation in transactions is used to maintain integrity in the database. Due to some failure of power, hardware, or software, etc., a transaction might get interrupted before all its operations are completed. This may cause ambiguity in the database, i.e. it might get inconsistent before and after the transaction. To ensure that further operations of any other transaction are performed only after work of the current transaction is done, a commit operation is performed to the changes made by a transaction permanently to the database.

iv) Rollback

This operation is performed to bring the database to the last saved state when any transaction is interrupted in between due to any power, hardware, or software failure.

In simple words, it can be said that a rollback operation does undo the operations of transactions that were performed before its interruption to achieve a safe state of the database and avoid any kind of ambiguity or inconsistency.

6.2 Concurrency in Transaction

Executing many transaction simultaneously is the simplest definition of concurrency. To improve time efficiency, it is necessary. Inconsistency develops when many transactions attempt to access the same piece of data. Data consistency requires the use of concurrency control.

Advantages of Concurrency

In general, concurrency means, that more than one transaction can work on a system. The advantages of a concurrent system are:

- **Waiting Time:** It means if a process is in a ready state but still the process does not get the system to get execute is called waiting time. So, concurrency leads to less waiting time.
- **Response Time:** The time wasted in getting the response from the cpu for the first time, is called response time. So, concurrency leads to less response time.
- **Resource Utilization:** The amount of resource utilization in a particular system is called resource utilization. Multiple transactions can run parallel in a system. So,

concurrency leads to more resource utilization.

- **Efficiency:** The amount of output produced in comparison to given input is called efficiency. So, concurrency leads to more efficiency.

Disadvantages of Concurrency

- Deadlocks: Deadlocks can occur when two or more transactions are waiting for each other to release resources, causing a circular dependency that can prevent any of the transactions from completing.
- Reduced concurrency: Concurrency control can limit the number of users or applications that can access the database simultaneously. This can lead to reduced concurrency and slower performance in systems with high levels of concurrency.
- Complexity: Implementing concurrency control can be complex, particularly in distributed systems or in systems with complex transactional logic.
- Inconsistency: In some cases, concurrency control can lead to inconsistencies in the database. For example, a transaction that is rolled back may leave the database in an inconsistent state, or a long-running transaction may cause other transactions to wait for extended periods, leading to data staleness and reduced accuracy.

6.3 ACID Properties

ACID stands for Atomicity, Consistency, Integrity, and Durability in database management. These are the four properties that ensure the reliability and consistency of database transactions despite failures. ACID features are used by Database Management Systems (DBMS) to maintain data integrity and accuracy

Atomicity

Atomicity guarantees that all of the commands that make up a transaction are treated as a single unit and either succeed or fail together. This is important in the event of a system failure or power outage, in that if a transaction wasn't completely processed, it will be discarded and the database maintains its data integrity.

Consistency

Consistency guarantees that changes made within a transaction are populated across the database system (e.g., nodes) and in alignment with DBMS constraints. If data consistency is going to be negatively impacted by a transaction in an inconsistent state, the entire transaction will fail.

Isolation

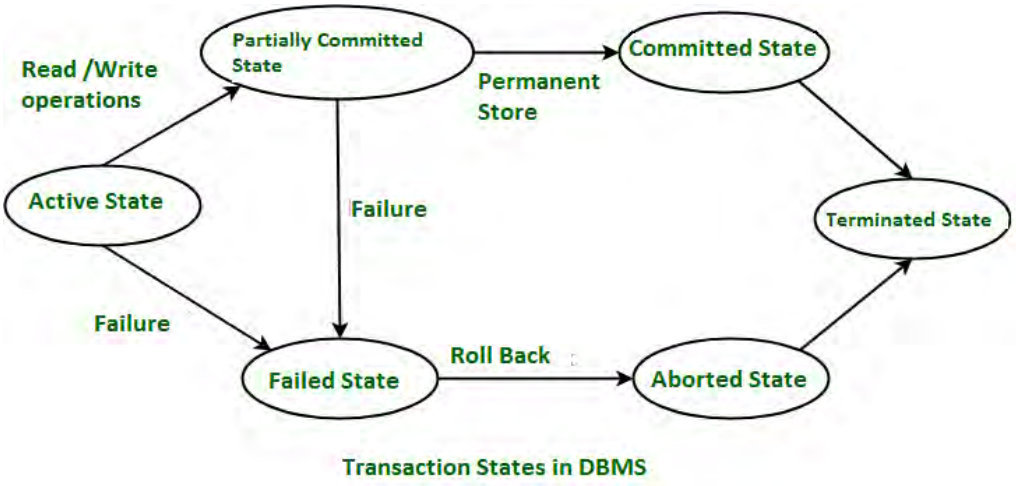
Each transaction is isolated from the other transactions to prevent data conflicts. This also helps database operations in relation to managing multiple entries and multi-level transactions. For example, if two users are trying to modify the same data (or even the same transaction), the DBMS uses a mechanism called a lock manager to suspend other users until the changes being made by the first user are complete.

Durability

Durability guarantees that once the transaction completes and changes are written to the database, they are persisted. This ensures that data within the system will persist even in the case of system failures like crashes or power outages. The concept of durability is a key element in data reliability.

6.4 State of Transaction

A Transaction log is a file maintained by the recovery management component to record all the activities of the transaction. After the commit is done transaction log file is removed.



Transaction States in DBMS

These are different types of transaction states:

1. **Active State –** When the instructions of the transaction are running then the transaction is in active state. If all the 'read and write' operations are performed without any error then it goes to the "partially committed state"; if any instruction fails, it goes to the "failed state".

2. **Partially Committed –** After completion of all the read and write operation the

changes are made in main memory or local buffer. If the changes are made permanent on the database then the state will change to "committed state" and in case of failure it will go to the "failed state".

3. **Failed State –** When any instruction of the transaction fails, it goes to the "failed state" or if failure occurs in making a permanent change of data on database.

4. **Aborted State –** After having any type of failure the transaction goes from "failed state" to "aborted state" and since in previous states, the changes are only made to local buffer.

5. **Committed State –** It is the state when the changes are made permanent on the data base and the transaction is complete and therefore terminated in the "terminated state".

6. **Terminated State –** If there isn't any roll-back or the transaction comes from the "committed state", then the system is consistent and ready for new transaction and the old transaction is terminated.

Exercise

Choose the correct answer from the given alternatives.

1. A _____ consists of a sequence of query and/or update statements.

   a. Transaction c. Rollback

   b. Commit d. Flashback

2. Which of the following makes the transaction permanent in the database?

   a. View c. Rollback

   b. Commit d. Flashback

3. Consider the Following Action:

   TRANSACTION.....

   Commit;

   ROLLBACK;

   What does Rollback do?

   a. Undoes the transactions before commit

   b. Clears all transactions

   c. Redoes the transactions before commit

   d. No action

4. In order to maintain the consistency during transactions, database provides

   a. Commit b. Atomic c. Flashback d. Retain

5. A transaction that completes its execution is said to be

   a. Committed b. Aborted c. Rolled back d. Failed

6. _____ will undo all statements up to commit?

   a. Transaction b. Flashback c. Rollback d. Abort

7. Which ACID property would be violated if a transaction read data that was modified by another concurrent transaction but not yet committed?

   a. Atomicity b. Consistency

   c. Isolation d. Durability

8. Which mechanism is often used to implement the Isolation property in a database system?

   a. Checkpoints

   b. Two-phase commit protocol

   c. Locking and concurrency control

   d. Redundant arrays of independent disks (RAID)

9. What does the Durability property ensure about a transaction's effects?

   a. They are temporary until confirmed by the user.

   b. They are rolled back if a failure occurs.

   c. They are preserved and not lost due to system failures.

   d. They are subject to being overridden by other transactions

10. Which of the following scenarios demonstrates a failure to uphold the consistency property?

    a. A transaction that successfully commits after ensuring no conflicting updates

    b. A transaction that fails to complete, but all its operations are rolled back

    c. A transaction that leaves the database in a state where constraints are violated

    d. A transaction that ensures its changes are visible to all subsequent transactions

11. In a DBMS, how can Durability be ensured?

    a. By using rollback segments

    b. By maintaining transaction logs and ensuring that changes are written to non-volatile storage

    c. By using isolation levels to control visibility of transaction changes

    d. By employing deadlock detection mechanisms

12. Which ACID property is tested when verifying that a transaction does not interfere with other concurrently executing transactions?

    a. Atomicity b. Consistency c. Isolation d. Durability

13. What does the Atomicity property ensure in the context of a transaction?

    a. Transactions are executed in a way that maintains database consistency.

    b. All operations of a transaction are completed, or none are executed.

c. Transactions do not interfere with each other's execution.

d. Changes made by a transaction are permanent even after a failure.

14. If a transaction causes an update to a database that cannot be rolled back, which ACID property is at risk?

a. Atomicity b. Consistency

c. Isolation d. Durability

Write short answer to the following questions.

1. What do you mean by transaction? Explain it with illustration.
2. Describe the ACID properties in transaction.
3. Explain the state of transaction with the help of suitable example.
4. Discuss the ACID properties of database transaction with suitable example.
5. Why do we need concurrency in transaction? Explain in short.

Write long answer to the following questions.

1. Draw a state diagram, and discuss the typical state that a transaction goes through during transaction.
2. Explain the major problem with the concurrent execution of Transaction.
3. Write short on the following:
   a. Atomicity
   b. Consistency
   c. Isolation &
   d. Durability

DataBase Backup, Recovery and Security

Unit 7

7.1 Introduction to Backup

A database backup is the process of creating a copy of data from a database to ensure its availability in case of data loss, corruption, or system failure. This backup can be used to restore the database to its previous state, safeguarding critical information from unexpected issues. Backups can be performed regularly and stored on-premises or in cloud environments, providing a reliable safeguard for valuable business data. A database backup is a process for safeguarding your data against loss, corruption, or system failure by regularly creating copies of your database. Backups protect your business from accidental data loss, ensure rapid recovery from disruptions, and safeguard against cyber threats, helping maintain business continuity.

Advantage of Database Backup

1. Protect your Data from Accidental Loss

Backups help ensure that you can recover data even if it's accidentally deleted or altered. By restoring your database to its previous state, you reduce the impact of human error, maintain data accuracy, and improve cloud ROI by preventing costly data recovery efforts and downtime.

2. Ensure Business Continuity

With a reliable backup in place, you can typically restore operations and minimize downtime in case of system failures. This helps keep your business running smoothly, even during unexpected outages or disruptions.

3. Safeguard Against Cyber Threats

Backups are reliable, safe spots that can allow you to restore clean database versions after a cyberattack. This reduces the risk of data breaches and helps you recover quickly without compromising sensitive information.

4. Simplify Disaster Recovery

Backups enable faster recovery from disasters, reducing the impact on your business and minimizing operational disruptions. Whether it's a natural disaster or a hardware

failure, having a backup helps ensure that you can resume normal operations efficiently.

5. **Security**

   By regularly backing up your data, you can ensure that even if your system is compromised or suffers a hardware failure, you won't lose any critical information. This can be especially important for businesses that deal with sensitive customer information or financial records.

7.2 Types of Backup

A database backup is a copy of storage that is stored on a server. Backup is used to prevent unexpected data loss. If original data gets lost, then with the help of a backup, it is easy to gain access to the data again.

There are two types of database backup.

- Physical backup
- Logical backup

Physical Backup

Physical database backups are backups of physical files that are used to store and recover databases. These include different data files, control files, archived redo logs, and many more. Typically, physical backup data is kept in the cloud, offline storage, magnetic tape, or on a disc.

There are two methods to perform a physical backup:

1. Operating system utilities
2. Recovery manager

This type of backup is useful when the user needs to restore the complete database in a short period. It is beneficial to provide details of transactions and changes made in databases. It is considered the foundation of the recovery mechanism. This form of backup has the drawback of slowing down database operations.

Advantages

- It is useful when the user needs to restore the complete database in a short period.
- They provide details of transactions and changes made in databases.

Disadvantage

- This slows down database operations
- Requires large storage space
- Slower backup and restore time

Logical Backup

It contains logical data which is retrieved from the database. It contains a view, procedure, function, and table. This is useful When users want to restore or transfer a copy of the database to a different location. Logical backups are not as secure as physical backups in preventing data loss. It only provides structural details. Every week, complete logical backups should be performed. Logical backups are used as a supplement to a physical backup.

Advantages

- This is useful when the user needs to restore the complete database to the last time.
- It was more complex and provides granular recovery capabilities.

Disadvantages

- Critical for recovery of special components
- Less secure compared to physical backup.
- It only provides structural details.

Physical Backup Vs Logical Backup

Physical Backup	Logical Backup
1. Physical database backups are backups of physical files that are used to store and recover databases.	1. Logical database backups are backups of logical files that are retrieved from the database.
2. It contains data files, control files, and archived redo logs.	2. It contains a view, a procedure, a function, and a table.
3. It copies data files when data is running or stopped.	3. Using the EXPORT keyword Logical backup is done
4. A user needs to restore the complete database in a short period of time.	4. This is useful when users want to restore or transfer a copy of the database to a different location.

Physical Backup	Logical Backup
5. More secure than logical backup.	5. Less secure as compared to Physical backup.

7.3 Reasons for Database Failure

Failure of a database is the inability of the database to carry out the requested transactions or the loss of database data. A database is susceptible to several failures and each of these failures requires a unique management strategy. It can occur for variety of causes.

Failure Classification

To find that where the problem has occurred, we generalize a failure into the following categories:

1. Transaction failure
2. System crash
3. Disk failure

1. Transaction Failure

The transaction failure occurs when it fails to execute or when it reaches a point from where it can't go any further. If a few transaction or process is hurt, then this is called as transaction failure.

Reasons for a transaction failure could be -

a. Logical errors: If a transaction cannot complete due to some code error or an internal error condition, then the logical error occurs.

b. Syntax error: It occurs where the DBMS itself terminates an active transaction because the database system is not able to execute it. For example, the system aborts an active transaction, in case of deadlock or resource unavailability.

2. System Crash

A system crash usually occurs when there is some sort of hardware or software breakdown. Some other problems which are external to the system and cause the system to abruptly stop or eventually crash include failure of the transaction, operating system errors, power cuts, main memory crash, etc.

These types of failures are often termed soft failures and are responsible for the data

losses in the volatile memory. It is assumed that a system crash does not have any effect on the data stored in the non-volatile storage and this is known as the fail-stop assumption.

3. Disk Failure

It occurs where hard-disk drives or storage drives used to fail frequently. It was a common problem in the early days of technology evolution.

Disk failure occurs due to the formation of bad sectors, disk head crash, and unreachability to the disk or any other failure, which destroy all or part of disk storage.

7.4 Methods of Database Backup

1. Full backups

The most basic and complete type of backup operation is a full backup. As the name implies, this type of backup makes a copy of all data to a storage device, such as a disk or tape. The primary advantage to performing a full backup during every operation is that a complete copy of all data is available with a single set of media. This results in a minimal time to restore data, a metric known as a recovery time objective. However, the disadvantages are that it takes longer to perform a full backup than other types (sometimes by a factor of 10 or more), and it requires more storage space.

2. Incremental Backups

An incremental backup operation will result in copying only the data that has changed since the last backup operation of any type. An organization typically uses the modified time stamp on files and compares it to the time stamp of the last backup. Backup applications track and record the date and time that backup operations occur in order to track files modified since these operations.

Because an incremental backup will only copy data since the last backup of any type, an organization may run it as often as desired, with only the most recent changes stored. The benefit of an incremental backup is that it copies a smaller amount of data than a full. Thus, these operations will have a faster backup speed, and require less media to store the backup.

3. Differential Backups

A differential backup operation is similar to an incremental the first time it

is performed, in that it will copy all data changed from the previous backup. However, each time it is run afterwards, it will continue to copy all data changed since the previous full backup. Thus, it will store more backed up data than an incremental on subsequent operations, although typically far less than a full backup. Moreover, differential backups require more space and time to complete than incremental backups, although less than full backups.

4. Synthetic full backups

Synthetic full backups are a great way to reduce the impact of long incremental chains without the impact of pulling all the data again as you would in a full backup. A synthetic full backup is created by using the previous full backup and the incremental backups to create a new full backup with all the incremental changes. This is a reset point for your incremental backup strategy. This method is sometimes called 'incremental forever.' A good solution would do the synthetic full creation at the backup storage layer.

5. Mirror Backups

A mirror backup is comparable to a full backup. This backup type creates an exact copy of the source data set, but only the latest data version is stored in the backup repository with no track of different versions of the files. All the different backed up files are stored separately like they are in the source.

One of the benefits of mirror backup is a fast data recovery time. It's also easy to access individual backed up files. One of the main drawbacks, though, is the amount of storage space required. With that extra storage, organizations should be wary of cost increases and maintenance needs. If there's a problem in the source data set, such as corruption or deletion, the mirror backup experiences the same. As a result, it is not to rely on mirror backups for all the data protection needs and have other backup types for the data.

7.5 Concept of Recovery, Redo/Undo

Database systems like any other computer system, are subject to failures but the data stored in them must be available as and when required. When a database fails it must possess the facilities for fast recovery. It must also have atomicity i.e. either transactions are completed successfully and committed (the effect is recorded permanently in the database) or the transaction should have no effect on the database.

Types of Recovery Techniques in DBMS

Database recovery techniques are used in Database Management Systems (DBMS) to restore a database to a consistent state after a failure or error has occurred. The main goal of recovery techniques is to ensure data integrity and consistency and prevent data loss.

There are mainly two types of recovery techniques used in DBMS

i. Rollback/Undo Recovery Technique

The rollback/undo recovery technique is based on the principle of backing out or undoing the effects of a transaction that has not been completed successfully due to a system failure or error. This technique is accomplished by undoing the changes made by the transaction using the log records stored in the transaction log. The transaction log contains a record of all the transactions that have been performed on the database. The system uses the log records to undo the changes made by the failed transaction and restore the database to its previous state.

ii. Commit/Redo Recovery Technique

The commit/redo recovery technique is based on the principle of reapplying the changes made by a transaction that has been completed successfully to the database. This technique is accomplished by using the log records stored in the transaction log to redo the changes made by the transaction that was in progress at the time of the failure or error. The system uses the log records to reapply the changes made by the transaction and restore the database to its most recent consistent state.

7.6 Introduction to Database Security

Database security refers to the various measures organizations like yours take to ensure their databases are protected from internal and external threats. Database security includes protecting the database itself, the data it contains, its database management system, and the various applications that access it. Organizations must secure databases from deliberate attacks such as cybersecurity threats and misuse of data and databases by those who can access them.

Benefits of using Database Security

1. Avoiding Financial Losses

Data breaches can be incredibly costly. According to a report by IBM, the average cost of a data breach in 2021 was $4.24 million. By investing in robust database security and compliance measures, businesses can mitigate these financial risks.

2. Supporting Business Continuity

A secure and compliant database infrastructure is critical for business continuity. In the event of a cyberattack or data breach, having a robust security framework in place ensures that the business can quickly recover and continue its operations.

3. Protecting Sensitive Information

The primary benefit of database security is the protection of sensitive information. Databases often contain personal data, financial information, and proprietary business details. Implementing robust security measures such as encryption, access controls, and regular security audits can significantly reduce the risk of data breaches

4. Ensuring Regulatory Compliance

Compliance with data protection regulations is not optional; it is a legal requirement for many businesses. Non-compliance can result in hefty fines and legal actions. Implementing compliance measures helps businesses avoid these penalties and ensures that they handle data in a manner that respects user privacy and data security.

5. Building Customer Trust

In an era where data breaches are increasingly common, customers are more concerned than ever about the security of their personal information. Businesses that prioritize database security and compliance can build stronger relationships with their customers by demonstrating their commitment to protecting sensitive information. This trust can translate into increased customer loyalty and a competitive advantage in the marketplace.

6. Enhancing Operational Efficiency

Effective database security and compliance can also enhance operational efficiency. By standardizing data handling procedures and implementing automation tools, businesses can streamline their operations and reduce the risk of human error. Automated security measures can include real-time monitoring for suspicious activities, automated backups, and quick restoration processes in case of a breach. These efficiencies save time and resources, allowing businesses to focus on their core activities.

7.7 Common Threats in Database Security

Database attacks come in many forms, from malware-infected computer systems to targeted cyberattacks on your database systems themselves. The most common types

of database attack include:

1. **SQL Injection:** This is when an attacker inserts malicious code into a website's login page to obtain information.
2. **Denial of Service (DoS):** This is an attempt to overload a database or website with superfluous requests, causing it to slow down or even shut down.
3. **Database Exploitation:** This refers to the unauthorized use of sensitive data. Code modification: This occurs when attackers change the code of a database.
4. **Malicious Insiders:** This are people who have been given access to a database, but misuse their privileges.
5. **Hidden Flaws in Code:** These can be exploited by attackers to steal sensitive data or cause damage to your system.

Exercise

Choose the correct answer from the given alternatives.

1. What is the primary purpose of database backup?

   a. To improve database performance

   b. To restore the database to a previous state in case of failure or data loss

   c. To optimize SQL queries

   d. To manage user access controls

2. Which type of backup involves copying all the data in the database, regardless of previous backups?

   a. Incremental Backup b. Differential Backup

   c. Full Backup d. Log Backup

3. In which backup type are only the changes made since the last backup saved?

   a. Full Backup b. Differential Backup

   c. Incremental Backup d. Snapshot Backup

4. Which backup strategy involves making a backup of the data since the last full backup, but not the incremental backups?

   a. Incremental Backup b. Differential Backup

   c. Full Backup d. Log Backup

5. What is a transaction log backup?

   a. A backup of all the data in the database

   b. A backup of only the changes to the database since the last log backup

   c. A backup of the database schema

   d. A backup of the database's transaction logs to enable point-in-time recovery

6. What does the term 'point-in-time recovery' refer to?

   a. Restoring the database to the state it was in at a specific point in time

   b. Restoring only the schema of the database

   c. Recovering data from a backup made yesterday

   d. Restoring the database to the most recent backup

7. Which backup strategy requires the least amount of storage space and backup time?

   a. Full Backup
   b. Differential Backup
   c. Incremental Backup
   d. Mirror Backup

8. What is the primary advantage of a differential backup over a full backup?

   a. It provides a complete backup of the database
   b. It is faster to perform and requires less storage space compared to a full backup
   c. It allows for point-in-time recovery
   d. It is more secure than full backups

9. Which of the following backup methods is typically used to protect against data loss due to user errors or system crashes?

   a. Full Backup
   b. Incremental Backup
   c. Differential Backup
   d. Log Backup

10. Which of the following is a primary goal of database security?

    a. Increase transaction speed
    b. Ensure data consistency
    c. Protect data from unauthorized access and breaches
    d. Optimize query performance

11. What is the purpose of authentication in database security?

    a. To verify the identity of users accessing the database
    b. To encrypt data stored in the database
    c. To ensure data integrity during transmission
    d. To manage user roles and permissions

Write short answer to the following questions.

1. What is backup? Explain is the role of backup in database management system?
2. Describe logical and physical backup with the help of suitable example.
3. Why is it important to back up a database regularly?
4. What are the different types of backups that can be performed on a database?
5. What are the common mistakes to avoid while taking database backups?

6. How do you choose the backup frequency for a database?
7. What are the tools available for taking database backups?
8. What are the factors to consider when selecting a backup and recovery solution?
9. Write short notes on the following:
    i. Full backup
    ii. Differential Backup &
    iii. Incremental Backup

Write long answer to the following questions.

1. What is backup and why is it important for databases?
2. What are the two types of backup and how do they differ from each other?
3. What is physical backup and what components of the database does it backup?
4. What is logical backup and what type of data does it backup?
5. What are the common reasons for database failure and how can they be prevented?
6. What are the different methods of database backup and what are the advantages and disadvantages of each?
7. What is recovery and how does it relate to database backup?
8. What is the concept of redo and undo in database backup and recovery?
9. Why is database security important and what are the potential threats to databases?
10. How can organizations protect their databases from security threats and ensure the integrity, availability, and confidentiality of their data?

Reference

Book References

"Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan

"Fundamentals of Database Systems" by Ramez Elmasri and Shamkant B. Navathe

"Database Management Systems" by Raghu Ramakrishnan and Johannes Gehrke

"SQL: The Complete Reference" by James R. Groff and Paul N. Weinberg

"An Introduction to Database Systems" by C.J. Date

Web References

https://www.geeksforgeeks.orgc

https://www.tutorialspoint.com

https://www.techtarget.com

https://en.wikipedia.org

https://www.splunk.com

https://www.w3schools.com