Bachelor of Computer Applications (BCA)

Database Management System

(OBCACO201T24)

Self-Learning Material

(SEM -II)

Jaipur National University
Centre for Distance and Online Education

Established by Government of Rajasthan

Approved by UGC under Sec 2(f) of UGC ACT 1956

&

NAAC A+ Accredited

TABLE OF CONTENTS

EXPERT COMMITTEE

Prof. Sunil Gupta
(Computer and Systems Sciences, JNU Jaipur)

Dr. Satish Pandey
(Computer and Systems Sciences, JNU Jaipur)

Dr. Shalini Rajawat
(Computer and Systems Sciences, JNU Jaipur)

COURSE COORDINATOR

Mr. Satender Singh
(Computer and Systems Sciences, JNU Jaipur)

UNIT PREPARATION

Unit Writer(s)	**Assisting & Proofreading**	**Unit Editor**
Mr. Satender Singh (Computer and Systems Sciences, JNU Jaipur) (Unit 1-5)	Mr. Pawan Jakhar (Computer and Systems Sciences, JNU Jaipur)	Mr. Ramlal Yadav (Computer and Systems Sciences, JNU Jaipur)
Mrs. Rashmi Choudhary (Computer and Systems Sciences, JNU Jaipur) (Unit 6-10)		

Secretarial Assistance

Mr. Mukesh Sharma

COURSE INTRODUCTION

"The computer was born to solve problems that did not exist before."
- Bill Gates

Welcome to the Database Management Systems (DBMS) course! This course offers a comprehensive journey into the world of databases, which are fundamental to the structure and operation of modern information systems. Databases enable efficient data storage, retrieval, and management, forming the backbone of numerous applications and services we rely on daily. From online banking and e-commerce platforms to social media and scientific research, databases play a crucial role in managing vast amounts of data seamlessly and reliably. This course is designed to equip you with the essential knowledge and skills needed to understand and implement effective database systems, covering both theoretical concepts and practical applications.

This course has 3 credits and is divided into 10 Units. The course begins with an exploration of the foundational concepts of databases, including the need for databases, their evolution, and the role of a DBMS in managing data. We will delve into data modeling techniques, starting with the Entity-Relationship (ER) model, which helps in designing a blueprint of a database. You will learn how to translate real-world scenarios into ER diagrams and subsequently into relational schemas. The principles of data normalization will also be covered, ensuring that your database designs are efficient and free from anomalies. As we progress, you will gain proficiency in SQL (Structured Query Language), the standard language for interacting with relational databases. Through hands-on exercises, you will learn to create, modify, and query databases using SQL, enabling you to manipulate data effectively.

Course Outcomes

At the completion of the course, a student will be able to:

1. Identify and organize the information from a DBMS and maintain and retrieve efficiently, and effectively.
2. Illustrate the role of Database Management Systems in information technology applications within organizations and structured query languages to extract information from large datasets
3. Applying contemporary logical design methods and tools for databases and derive a physical design for a database from its logical design.
4. Analyze and design a real database application.
5. Evaluate a real database application using a database management system.

Acknowledgements:

The content we have utilized is solely educational in nature. The copyright proprietors of the materials reproduced in this book have been tracked down as much as possible. The editors apologize for any violation that may have happened, and they will be happy to rectify any such material in later versions of this book.

Unit-1

Basics of Database

Learning Outcomes:

- Students will be capable of learning about the database and database users.
- Students will be able to understand the Characteristics of the database and database systems.
- Students will be able to understand the concepts and architecture of DBMS.
- Students will be able to understand about the schemas and Instances.
- Students will be able to learn about database languagesand Interfaces.

Structure

1.1 Database and Database User

1. Databases and database systems are ubiquitous in modern life, with most individuals interacting with them daily. Banking transactions, hotel or airline reservations, library catalog searches, and online shopping all involve accessing databases. Even routine activities like grocery shopping involve automatic updates to inventory databases. The pervasive influence of databases extends to various fields, including business, electronic commerce, engineering, medicine, genetics, law, education, and library science.
2. A database management system (DBMS) comprises software programs that empower users to create and manage databases. These systems assist users and applications in defining, constructing, manipulating, and sharing databases. Defining a database involves specifying data types, structures, and constraints. The DBMS also stores descriptive information, known as metadata, in the form of a database catalog or dictionary.
3. Database construction involves storing data on a storage medium managed by the DBMS. Databases are indispensable in numerous computer-related fields due to their ability to store and manage structured information efficiently.
4. A database is a collection of related data with inherent meaning. It represents a subset of reality, often referred to as the mini-world or universe of discourse (UoD), and changes in the mini-world are reflected in the database. Logically consistent and designed for specific purposes, databases cater to particular target audiences and applications.
5. There are two major types of databases: relational (or sequence) databases and non-relational (or non-sequence) databases, also known as NoSQL databases. Organizations may utilize these types individually or in combination based on data nature and required functionality.
6. Accuracy and reliability are paramount in database management. Databases must accurately reflect the mini-world they represent, necessitating prompt updates to reflect changes. Large commercial databases, such as Amazon.com, exemplify the scale and importance of database usage in modern commerce.
7. Databases can be manually or automatically created and maintained. While manual databases, like library card catalogs, exist, most databases today are computerized and managed by DBMS or application programs tailored for database management.

Database Users

Database users are granted permissions for various actions such as reading, inserting, updating, and deleting specific objects defined by a set of fields and business rules. These objects can be utilized to modify one or more database tables. To create database users, the Database Access action within the Users application can be employed.

In database management systems (DBMS), database users access and retrieve data from the database through the software's applications and interfaces. Security measures are essential to prevent unauthorized access to the database. Different database users may have distinct login IDs and passwords, restricting access to specific parts of the database associated with their roles.

Database users can be classified based on their interaction with the database. A role, for instance, is a database object that aggregates one or more privileges and can be assigned to users. Users assigned a role inherit all the privileges associated with that role, and a user may have multiple roles. Additionally, role hierarchies are supported in some systems.

Examples of database users include:

1. Naive Users: Railway ticket booking users and bank clerks are examples of naive users who may lack in-depth knowledge of DBMS but utilize the database to perform their assigned tasks.
2. System Analysts: System analysts are users tasked with analyzing the requirements of end users and defining system functionalities.

Database users can be further categorized into seven groups based on the tasks they perform on the databases.

- **Database Administrators (DBA)**

  The most crucial category of database user in DBMS is the Database Administrator (DBA). A person or group of people who define the database schema and manage the database at different organizational levels is known as a database administrator. Database administrators (DBAs), also referred to as super-users, have total control over the database. They work together with programmers to plan and design the workflow, functionalities, and overall layout of the database. All other users' authorization permissions may be granted or revoked at any time by DBAs.
- **Database Designers**

Database designers, as the term suggests, are professionals within a Database Management System (DBMS) who are responsible for planning and creating the structure of a database. This includes designing triggers, indexes, schemas, entity relationships, tables, constraints, and other essential elements. Their primary task involves gathering information based on the database requirements such as layout, functionality, cost considerations, preferred technologies, and implementation techniques before finalizing the database structure for programmers to implement its logic.

Database designers play a crucial role in determining the overall design of the database. They ascertain the types of data that need to be stored, identify the relationships between various entities within the database, and specify the attributes that will be utilized. In essence, they are tasked with creating a blueprint that outlines how the data will be organized and accessed within the DBMS.

- **System Analysts**

  System analysts analyses the requirements of Naive / Parametric End Users using DBMS databases. They must assess whether or not all end-user requirements have been satisfied. Feasibility, economic, and technical analysis are among the main duties of a system analyst in DBMS.They are occasionally in charge of the database's layout, organization, and use. They typically verify and gather all relevant database data, and if necessary, they can update or change the database's final design to satisfy the specifications. System analysts make sure the finished product complies with all requirements.

  A systems analyst is an individual who utilizes analysis and design techniques to address business challenges using information technology. These professionals often act as change agents, identifying organizational improvements, designing systems to enact those changes, and training and motivating others to utilize the systems effectively.

- **Application Programmers / Back-End Developers**

  Computer experts who design application programmes (in C, C++, Java, PHP, Python, and other languages) or the user interface so that other users can interact with the database via these applications are known as application programmers, also referred to as back-end developers.

  Database management systems are well-versed to application programmers (DBMS).For storing and retrieval of the data, they use DML (Data Manipulation Language) queries to

interact with the database. Application programmers may, if necessary, also specify alterations to the database architecture for an application. In any language they are comfortable with, they can create databases.

- **Naive Users / Parametric Users**

  Naive users, which isalso known as Parametric End users, are people who generally have no or little knowledge about database management systems but frequently use database applications to accomplish their goals. The database is mostly used by inexperienced users to fill in or retrieve information through the interface provided by DBMS applications. Inexperienced users do not need to be aware of the database system's presence because they can also interact with it via a menu-driven application interface. Naive / Parametric End Users access the database directly through the developed applications in order to achieve the desired results. Railway ticket booking users, for example, are naive/parametric end users because they are usually not familiar with DBMS and rely solely on the railway booking application to book their tickets.

- **Sophisticated Users**

  Sophisticated users of databases have an extensive grasp of database management systems (DBMS), encompassing familiarity with both Data Definition Language (DDL) and Data Manipulation Language (DML) commands. This group typically includes professionals like business analysts, engineers, scientists, and system analysts, who exhibit proficiency in database operations and can effectively interact with intricate data structures.

- **Casual Users / Temporary Users**

  Casual users, often referred to as temporary users, are individuals who make sporadic use of database services. When accessing the database, they anticipate having all necessary information available in one place. These users are typically unfamiliar with DBMS and may require new information each time they access the database. Examples of casual or temporary users include high-level management personnel with limited DBMS knowledge, who may access the database to input new information or retrieve existing data.

1.2 Characteristics of the Database

- **Self-describing nature of a database system:** In addition to the database itself, a self-describing database system includes metadata that defines and describes the data and

connections between tables. Users of the databases or DBMS software may use this information as necessary. The separation of data and data about data in a database system sets it apart from a conventional file-based system.

- **Insulation between program and data:** In a file-based system, if a user wants to change the structure of a file, all programs that access that file may need to be changed as well because application programs define the structure of the data files. However, the database approach stores the data structure in the system catalog rather than the programs, allowing for easier structural changes with only one modification needed. This concept is referred to as "program-data independence."
- **Support for multiple views of data:** Databases support multiple data views, where a view is a defined subset of the database accessible only to certain system users. Different users' perspectives on the system are possible, and each view may include only information pertinent to one user or group of users.
- **Enforcement of integrity constraints:** Database management systems (DBMS) enable users to define and enforce specific constraints to ensure the accuracy of entered information and maintain data integrity. Constraints serve as restrictions or guidelines governing what can be added to or modified in a table. Examples include enforcing a particular postal code format or allowing only acceptable city names in the City field.
- **Sharing of data and multiuser system:** Databases are designed to allow multiple users to access the same database concurrently, facilitated by concurrency control techniques. These techniques ensure data integrity and accuracy while allowing for better performance compared to earlier, single-user databases.
- **Control of data redundancy:** Ideally, databases should store each individual data item only once, with redundancy minimized to enhance system performance. However, some redundancy may still be used strategically. Integrating all an organization's data into a database system facilitates data sharing among staff members and other authorized users, enabling greater insights from the dataset.
- **Transaction processing:** Database management systems typically include concurrency control subsystems to ensure data consistency and validity during transaction processing, even when multiple users update the same information simultaneously.

- **Backup and recovery facilities:** Database systems offer backup and recovery facilities to prevent data loss. These systems provide separate processes for backing up and recovering data, including network backup, to restore databases to their initial state in case of failures or crashes during update procedures.

1.3 Database Systems

Database Systems or Database Management Systems (DBMS) are essential software components designed to efficiently collect, store, and retrieve electronic and digital records, enabling the extraction of valuable information. The primary objective of a standard database is to facilitate the efficient storage and retrieval of data. For example, Relational Databases are specialized in storing and processing structured data, often utilizing Structured Query Language (SQL) for data access operations.

Databases and DBMSs play a critical role in managing vast, diverse, and rapidly growing datasets. Without a robust Database Management System, businesses would struggle to derive meaningful insights and perform in-depth analytics. The database environment supports a wide range of data-processing operations, enabling organizations to access, modify, control, and present data in a structured and organized manner.

Some of the well-known Database Management Systems (DBMSs) include MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database, and dBase. These systems are designed to organize data efficiently into rows and columns, facilitating the management and retrieval of information with speed and accuracy. The structured format of data storage in DBMSs helps distribute workload pressure effectively.

Furthermore, DBMSs are adept at handling diverse types of data, ranging from numerical and time series data to textual and binary data. This versatility makes them invaluable tools for managing and analyzing data across various domains and industries.

1.4 Concepts and Architecture

1.4.1 Concepts of DBMS

Tiers are classified as follows:

- A database is a structured collection of information or data, typically stored electronically within a computer system. Managing a database is commonly facilitated by a database management system (DBMS). The three primary components of a relational database include tables, keys, and relationships.
- Database architecture involves the creation of specialized software for businesses or organizations using programming languages. It encompasses the design, development, implementation, and maintenance of computer programs that store and organize information for various entities such as businesses, agencies, and institutions. Database architects are responsible for designing and deploying software to meet user needs and requirements.
- The architecture of a DBMS significantly influences its design. It can be structured in various ways, such as centralized, decentralized, or hierarchical. Additionally, the architecture of a DBMS can be categorized as single-tier or multi-tier.

- **Architecture with a single tier**

  In a one-tier or single-tier architecture, all of the required components for a software application or technology are placed on a single server or platform. In essence, a one-tier architecturewill keepall of an application's elements, such as the interface, Middleware, and back-end data, in a singleplace. These systems are usually regarded bythe developers as one of the simplest and most direct method.
- **Architecture with two levels**

  Client Server is a two-tier architecture. A client-server application is similar to a two-tier architecture. The client and the server have direct communication. Between the client and the server, there is usually no intermediary.
- **Three-tiered architecture**

  The tiers of a three-tier architecture are generally determined by the complexity of the users and how they can use the data in the database. It is the most common architecture for creating the DBMS.This architecture has a wide range of applications. It is suitable for use in both the and distributed applications. This architecture is especially effective when applied todistributed systems.
- **Architecture with n levels**

The tiers of a three-tier architecture are determined by the complexity of the users and how they use the database's data. It is the architecture that is most frequently used when creating a DBMS.There are many uses for this architecture. It can be applied to distributed and web applications. Particularly effective in distributed systems is this architecture.

1.4.2 Database Architecture

The architecture of a Database Management System (DBMS) depicts how users interact with data in a database. It is unconcerned with how the database management system handles and processes data. It helps in the creation, design, and maintenance of a database that is used for storing and organizing information forbusinesses. Thebasic concept of a database management system is usually determined by the architecture of the system. Centralized, decentralized, and hierarchical architectures are all possible.

The three levels of DBMS architecture are as follows:

- **External tiers.**

  The segment of a database that pertains specifically to each individual user is known as the external level. At this level, users are shielded from the complexities of the conceptual and internal levels of the database. It represents the highest level of abstraction in database design and comprises multiple external schemas or user views. External levels provide various perspectives of the same database to individual users or user groups. By concealing certain underlying aspects of the database from each user, the external view serves as a robust and adaptable security measure.

- **Conceptual tiers.**

The conceptual level will usually describe what data is usually stored in the database and how the data is related to one another.

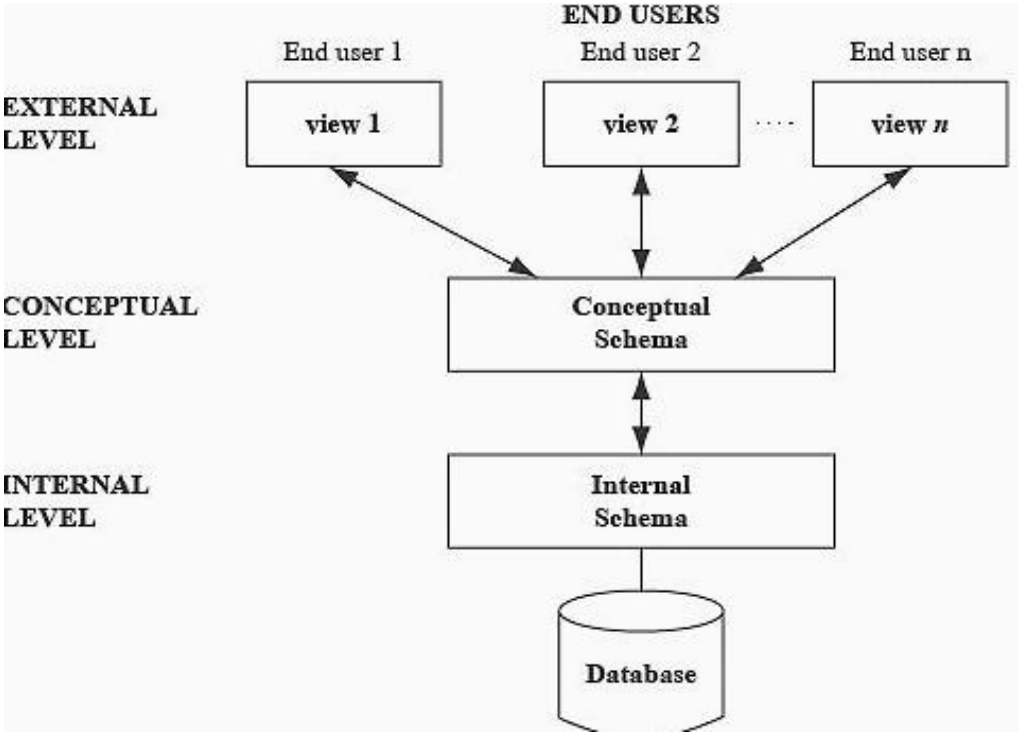
It stands for the following:

- ✓ Information about all the entities, attributes, and relationships.
- ✓ The limitations of the data.
- ✓ Information about security and integrity.

- **Internal tiers.**

The internal level is defined as the computer's physical representation of the database. This level describes how the data in the database is usually stored. It goes over the data structure and file

organization that are used for the storage ofvarious data on the storage devices. The levels of DBMS architecture are depicted in the diagram form below.



- **Knowledge Check 1**

  Fill in the Blanks

  1. The Database Management System (DBMS) architecture depicts how _____ interacts with data in the database.

  2. A database is a _____ collection of information or data that is typically saved electronically in a computer system.

  3. _____ database users in DBMS who will analyses the requirements ofthe Naive / Parametric End users.

- **Outcome-Based Activities 1**

  Try to Illustrate the architecture of Database Management System.

1.5 Schema and Instances

- Both schema and instance are essential concepts in database management systems (DBMS), but they serve distinct purposes.
- The term schema pertains to the overall description or structure of a database. It encompasses the design, organization, and relationships among the database objects such as tables, columns, and constraints. Essentially, the schema defines the blueprint or framework for how data will be stored and organized within the database. It represents the logical and conceptual view of the database and is not typically updated frequently.
- On the other hand, instances refer to the specific collections of data and information stored in the database at a particular moment in time. An instance represents the current state or snapshot of the database contents, including all the data records and values stored within the defined schema. Unlike the schema, instances are dynamic and can change over time as data is added, modified, or deleted from the database.
- In summary, the schema describes the overall design and structure of the database, while instances represent the actual data stored within the database at a given point in time.

The various types of schemas are as follows:

Physical schema refers to the physical design of any of givendatabase. It is defined as hidden beneath the logical schema and can be easily changed without disrupting any of the application programmes.

Logical schema A logical schema is defined as the database design. Programmers build the applications with the logical schema.

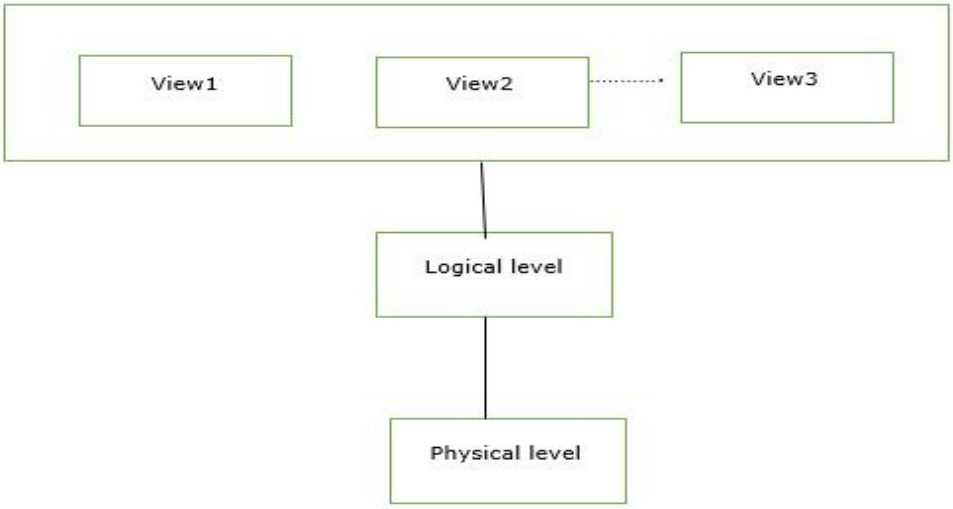
- The external schema, also known as the view level schema, represents the schema that end users interact with. Views, which provide a customized representation of the underlying data, are typically defined at this level within the schema hierarchy.
- A Database Management System (DBMS) typically supports various schema levels, including the physical schema, logical schema, and multiple sub or external schemas. The physical schema defines the database's physical storage structure, while the logical

schema defines the database's logical structure, such as tables, relationships, and constraints.

- The database schema, often referred to as the database layout or format, dictates how data is stored within the database. It serves as a blueprint for organizing and structuring the data, ensuring consistency and integrity. The database schema remains relatively constant over time, specifying the data structure and storage mechanisms unless modifications are made to accommodate changes in data requirements or system enhancements.

A database schema for a person will have fields for name, email, phone and address as shown below –

Name	Email	Phone no



Instances

These are a compilation of all the data and information that is kept on hand at any given time. CRUD operations, such as data and information deletion and addition, can be used to quickly change these instances. It's crucial to remember that no search results result in any changes.

The instances can be changed using specific CRUD operations, like data addition and deletion. It should be noted that no search will alter the instances in any way.

Imagine that we have a table in our database called School with a teacher and 50 records. This means that the instance of the database currently has 50 records, and tomorrow we'll add another 50 records, giving it a total of 100 records. A case is what we're talking about here.

A database instance for the Person database could be (User1,emai.com,11345679, addr), in which case the instance attributes of the person construct would include each user's individual entities.

This is depicted below:

Name	Email	Phone no
BOB	Kysd@yasd.com	2343489
JAY	wwr@sdas.in	5345476
PRIYANKA	wer@sdf.com	2342349

1.6 Data Independence

Data independence refers to the ability to make changes to the database schema without affecting the functionality of newly developed applications or programs. It involves keeping programs and data separate so that modifications to the data structure do not impact the performance of applications or the work of programmers. This principle is central to all three levels of data abstraction within a database.

Ensuring data independence is essential to accommodate changes and expansions in the database over time without disrupting operations at other levels. This approach minimizes the cost and time required for updating the database, as changes can be implemented without causing ripple effects across the system.

There are three levels of abstraction and two levels of data independence.

These are the following:

- **Physical data Independence**

  When we talk about physical data independence, we mean the capacity to modify the physical level without modifying the logical or conceptual levels. We can modify the

database's storage system using this property without having an impact on the logical schema.

The following given techniques may be used to effect changes at the physical level.

Introducing a new storage device, like magnetic tape or a hard disk.

Implementing a new storage data structure.

Adopting different data access methods or file organization techniques.

Relocating the database to a new location. The independence of logical data

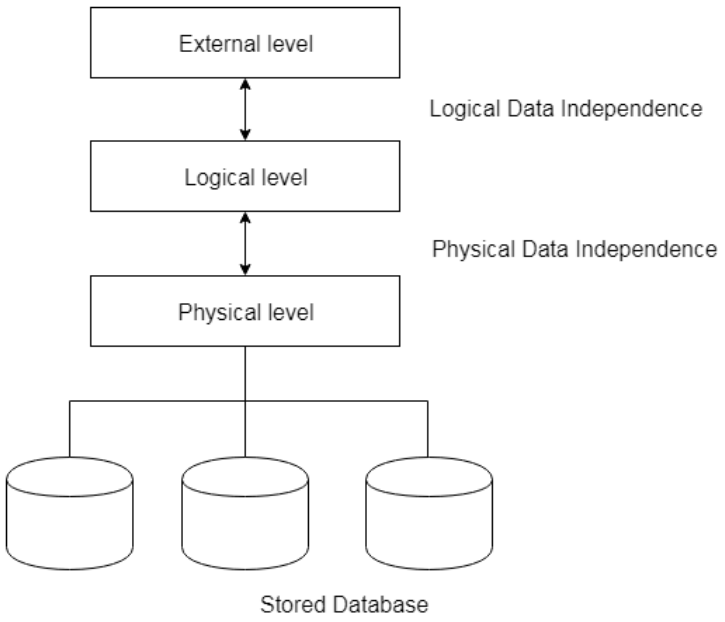
Logical Data Independence

The user's perspective on the data is the logical view of the data. It presents information in an accessible format for end users. Users should be able to manipulate the Logical View of data without being aware of the physical location of the data, in accordance with Codd's Rule of Logical Data Independence. A computer programme or piece of software is used to manipulate the data's logical view. What data should be kept in the database and how to use the logical level of abstraction are decisions made by the database administrator. It offers a thorough understanding of Data. Additionally, it describes the relationship and the data that will be kept in the database.

The database has a simple structure because of the dataindependence. It relies on application domain entities to fulfil functional requirements. It abstracts the functional requirements of the system. The class object diagrams define the static structure for the logical view. Users are not permitted to alter the database's logical structure.

Changes in the logical level could include.

- Modify the data definition.
- Adding, deleting, or updating any new database attribute, entity, or relationship.



Database Languages

DBMS offers the proper languages and interfaces for expressing database updates and queries. Database languages can be used to read, store, and update data in a database. Data must be altered by insertion, deletion, updating, and modification after it has been stored or filled. For these operations, the database management system offers a number of languages (DBMS). Thus, reading, updating, and storing data in a database are all done using database languages.

- Create, Drop, Truncate, and Rename are some of the Data Definition Language (DDL) commands.
- DML (Data Manipulation Language) Select, Insert, Delete, and Update.
- Revoke, Grant are Data Control Language (DCL) commands.
- TCL (Transaction Control Language): Rollback, Commit are the TCL commands.

1.7 Interfaces

The functionality of a DBMS is abstracted in a DBMS interface.It typically refers to the line of communication between a DBMS and its clients or to the abstraction that a DBMS component offers.The implementation of the encapsulated component's functionality is hidden behind a DBMS interface. Without knowing the query language, you can enter queries into a database using a database management system (DBMS) interface.

The following user interfaces may be offered by DBMSs:

Menu-Based Web Client or Browsing Interfaces

These user interfaces present a user with a menu of choices (called options) that help the user create a request. The main advantage of menu usage is that it takes the pressure off of remembering particular commands and query language syntax.

By compiling or choosing options from a menu that the system displays, the query is built step-by-step. In Web-based interfaces, pull-down menus are a common design element.

Forms-Based Interfaces

A forms-based interface displays a form to each user. Users have the option of filling out the entire form to add new data or just a portion of it, in which case the DBMS will redeem the same kind of data for the remaining fields. For users who are not familiar with operating systems, these kinds of forms are typically created, designed, and programmed. Forms specification languages are special languages that assist in the specification of such forms, and they are present in many DBMS.A form-based language like SQL* Forms, for instance, specifies queries using a form created in coordination with the relational database schema.

Graphical User Interface

A schema is typically presented to the user in diagrammatic form by a graphical user interface (GUI).The diagram can then be altered by the user to specify a query. Forms and menus are frequently used in GUIs.The majority of GUIs use a pointing device, like a mouse, to choose a particular area of the displayed schema diagram.

Natural language Interfaces

Natural language interfaces acknowledge and make an effort to comprehend requests made in either English or a different language. The conceptual schema of a database is analogous to the schema of a natural language interface, which also includes a dictionary of keywords.

Speech Input and Output

- While speech interfaces have traditionally seen limited use, their adoption is increasingly widespread and mainstream. Speech is now commonly utilized for various purposes, including querying databases, providing answers to inquiries, and fulfilling requests. This trend reflects the growing prevalence and acceptance of speech technology in diverse applications.
- Speech interfaces are particularly well-suited for applications with constrained vocabularies, such as phone directory inquiries, flight arrival/departure information services, and accessing bank account details. By leveraging speech, individuals can conveniently and efficiently access pertinent information without relying on traditional text-based input methods.
- Overall, the expanding use of speech technology underscores its value in enhancing accessibility and user experience across different domains and applications.

DBA interfaces –

Most of the database systems have exclusive DBA staff-only privileged commands. These instructions include those for establishing system parameters, authorizing accounts, modifying schemas, and rearranging all the database storage structures.

- **Knowledge Check 2**

  State true and false for the following sentences

  1. To read, store, and update data in a database, database clients are used. Once data has been stored or filled, it must be manipulated by insertion, deletion, updating, and modification.
  2. A Database Management System (DBMS) interface is a software interface designed to enable users to interact with a database without requiring knowledge of the underlying query language. It provides a user-friendly platform for entering queries and commands into the database, abstracting away the complexities of the query language.
  3. A graphical user interface (GUI) typically displays a schema in diagrammatic form to the user.

- **Outcome-Based Activities 2**

  Illustrate the difference between Physical and Logical data Independence with Proper diagram.

1.8 Summary

- A database comprises a structured assembly of information or data, usually stored electronically within a computer system. Its management is facilitated by a Database Management System (DBMS).
- Database users within a DBMS context encompass individuals benefiting from database usage, leveraging DBMS-provided applications and interfaces for data access and retrieval.
- Database users are granted access rights for reading, inserting, updating, and deleting specific objects, thereby defining a set of fields and business rules. The creation of database users is typically facilitated through the Database Access action in the Users application.
- The term schema denotes the overall description of a database, while an instance represents the collection of data and information stored within the database at any given point, with the schema remaining consistent throughout.
- Database programming languages enable the definition and manipulation of databases, with four primary types including Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL), and Transaction Control Language (TCL).
- DDL commands are utilized to alter or establish the schema and metadata of a database, while DML commands enable the manipulation of data stored within schema objects.
- A DBMS interface serves as a user-friendly platform allowing users to input queries into a database without necessitating the use of query languages.

1.9 Self-Assessment Questions

1. What is a Database?
2. What are the different types of Database users?

3. What are the Characteristics of the database?
4. What is Database Systems?
5. Explain the concept and architecture of the Database.
6. What is schemas and instances?
7. What is Data Independence?
8. What are the different types of DBMS Languages?

1.10 References

- Atkinson, M.P. (1981) *Database*. Maidenhead: PergamumInfoTech.
- Frank, L. and Helmersen, O. (1988) *Database theory and practice*. Wokingham, England: Addison-Wesley Publishing Company.
- *Database users: 2nd Toronto conference: Selected papers* (1990). Canadian Association for Information Science.
- *Database* (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) *Database*. Rockville, MD: Computer Science Press.

Unit 2

Relational Database Design

Learning Outcomes:

- Students will be capable of learning about Functional Dependencies and its implication.
- Students will be able to understand the closure rules and normalisation.
- Students will be able to understand the concepts of the Decomposition and Synthesis approach.
- Students will be able to understand about the 3NF and BCNF.
- Students will be able to learn about the Lossless join and Dependency Preserving decomposition.
- Students will learn about the Multi-valueddependency and 4NF.
- Students will understand the concept of Join dependency and 5NF.

Structure

- Knowledge Check 2
- Outcome-Based Activities 2

2.10 Summary

2.11 Self-Assessment Questions

2.12 References

2.1 Functional Dependencies and its implication

Functional dependency (FD) is a constraint that defines the relationship between an attribute and another attribute in a database management system (DBMS). Maintaining the integrity of the data in the database is made easier by functional reliance. It's critical to distinguish between poor and excellent database design. A functional dependence is represented by a "arrow". Y represents X's functional reliance on Y.

X->Y

The left side of the functional dependency is known. Let's assume that our employee table has the following columns: Emp Id, Emp Name, and Emp Address.

This is because we can determine the employee name associated with an Emp Id if we know it; the Emp Id attribute, in this case, can be used to uniquely identify the Emp Name attribute of the employee table. Functional dependability is expressed as:

Emp Id->Emp Name

In the above example,Emp_Name is functionally dependent on Emp Id.

2.1.1Types of Functional Dependency

1. Trivial functional dependency

The functional dependency A->B is straightforward if B is a subset of A. Trivial dependencies include, for example, A -> A and B ->B.

2. Non-trivial functional dependency

There exists a non-trivial functional dependence between A and B if B is not a subset of A.When the intersection of A and B is NULL, we say that they are entirely non-trivial.

1. ID → Name,
2. Name → DOB

2.2 Closure rules

Closure of an attribute x can be defined as the set of all the attributes that are generally functional dependencies on attribute X with respect to the F. It is usually denoted by the $X^{+,}$ which means what X can generally determine.An attribute's closure is a group of additional attributes that can be inferred from it in terms of functionality.

A set of FDs is closed when the set F+ of all FDs can be deduced from F.The set X+ of all attributes that are functionally determined by a set of attributes X with respect to F is known as closure.

Algorithm

Let's examine the X+ computation algorithm.

First 1: X+ = X

Step 2: iterate until X+ stays the same.

For each FD in F, Y->Z

If Y ⊆ X+ then X+ = X+ U Z

Pseudocode

The closure of X under functional dependency set F is X+, so ascertain this.

X Closure: = X itself will be contained;

Continue the procedure as follows:

X closure = old X Closure;

Do this for each functionally related P and Q in the FD set.

X Closure: = X Closure U Q if X Closure is a subset of P.

Until (X closure = old X Closure), repeat;

2.3 Normalisation

Normalization is the systematic procedure of structuring data within a database. It entails the creation of tables and the establishment of relationships between them based on predetermined guidelines. The primary objectives of normalization are to protect data integrity and enhance database flexibility by removing redundant information and inconsistent dependencies.

Normalization primarily focuses on reducing redundancy within relationships or groups of relationships. Relational redundancy can lead to anomalies during data insertion, deletion, and updates. By adhering to normalization rules, relational redundancy is minimized or eliminated, thereby enhancing data integrity and consistency.

Normalization is typically carried out through a series of stages known as normal forms. These normal forms define specific criteria that a relation must meet to be considered normalized. Each normal form addresses different aspects of data redundancy and dependency.

Normalization helps in organizing data efficiently and ensures that databases are structured in a way that facilitates data management, retrieval, and maintenance. By eliminating redundancy and adhering to standard formats, normalization enhances the overall integrity and reliability of the database system.

The basic reason for normalisation is the removal of the data anomalies. Dataredundancy is the main reason for the failure of eliminating anomalies, and it results in the data integrity and others such problems as the database increases.

Normalization is the process of structuring the data in the database. Normalization is used to minimize redundancy from a relation or group of relations. Another use for it is the elimination of unwanted features such as Insertion, Update, and Deletion Anomalies. During normalisation, the bigger table is divided into smaller ones, which are linked by relationships.

- An insertion anomaly occurs when there is not enough data to allow the insertion of a new tuple into a relationship.
- A circumstance known as a "deletion anomaly" occurs when some significant data is mistakenly erased together with other data.
- Update Anomaly: An update anomaly is when modifying one data value requires modifying several data rows.

2.3.1 Types of Normal Form

1. First Normal Form

When a relation possesses atomic values, it conforms to the first normal form (1NF) in Database Management Systems (DBMS). Simply put, 1NF dictates that each attribute within a table should contain only a single value and should not accommodate multiple values.

In other words, a relation is considered to be in the first normal form when none of the following characteristics are present, or it violates the first normal form if it includes composite or multi-valued attributes. A relation is said to adhere to the first normal form if every attribute within it is a single-valued attribute.

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	INDIA
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

Figure- 2.1 Conversion to 1NF

2. Second Normal Form

In Database Management Systems (DBMS), when a relation meets the criteria of the First Normal Form and contains no non-prime attribute that is functionally dependent on any proper subset of any candidate key, it achieves the Second Normal Form (2NF).

To qualify for the Second Normal Form, a relation must already adhere to the rules of the First Normal Form and should not display any partial dependencies. A relation attains the 2NF status when there are no partial dependencies observed between any non-prime attribute (attributes not included in any candidate key) and any proper subset of any candidate key in the table. Partial dependency occurs when a non-prime attribute is determined by only a portion of a candidate key.

3. Third Normal Form

The Third Normal Form (3NF) is a schema design strategy in relational databases aimed at reducing data redundancy, preventing anomalies, maintaining referential integrity, and facilitating data management through normalization principles.

A relation achieves the third normal form if it is already in the second normal form and does not contain any transitive dependency for non-prime attributes. In 3NF, each non-trivial functional dependency must be explained by at least one of the following criteria:

- $X \rightarrow Y$, where X is a super key.
- Y is a prime attribute (each element of Y is part of some candidate key).

4. Fourth Normal Form

Fourth Normal Form, or 4NF in DBMS terminology, is the state in which a relation adheres to Boyce-Codd Normal Form and does not possess any multi-valued dependencies. In a relation $A \rightarrow B$, a multi-valued dependency exists when there are multiple values of B for a single value of A.

A relation qualifies for Boyce-Codd Normal Form (BCNF) if it meets the criteria of the Third Normal Form and the left-hand side (LHS) is a super key for each functional dependency. In other words, if X is a super key in each non-trivial functional dependency $X \rightarrow Y$, the relation satisfies BCNF.

5. Fifth Normal Form

Fifth Normal Form (5NF), also referred to as Projection-Join Normal Form (PJ/NF), represents an advanced level of database normalization aimed at resolving redundancy in relational databases housing multi-valued facts by isolating semantically related multiple relationships.

In 5NF, the focus is on ensuring that if a relation is already in the Fourth Normal Form (4NF) without any join dependencies, the joining process should be lossless. This means that when combining multiple relations, no information should be lost, maintaining data integrity throughout the normalization process.

2.3.2 Advantages of Normalisation

Data redundancy is reduced with the aid of normalisation.

- A better general organization of the database.
- The internal data consistency of the database.
- A database architecture that is far more flexible.
- The concept of relationship integrity is upheld.

2.3.3 Disadvantages of Normalisation

- You need to determine the user's needs before building the database.
- The performance degrades as the relations are normalized to higher normal forms, like 4NF and 5NF.
- Restoring higher-degree connections to normalcy is a difficult and time-consuming task.
- Inadequate decomposition can lead to a badly designed database, which can lead to major problems.

2.4 Decomposition

When a table is broken up into several tables, the DBMS decomposition process aids in the removal of redundancy, inconsistencies, and anomalies from the database. Decomposition is the process of breaking down a relation X into smaller parts, such as X1, X2, and so on. Decomposition is both lossless and dependent preserving.

The process of disassembling a table in a database into its component pieces is known as decomposition. Decomposition, then, substitutes a set of several smaller relations for a given connection.

The process of breaking down a database table into its component parts is called decomposition. Hence, decomposition substitutes a collection of several smaller connections for a single relation. Therefore, each table in a database may be divided into many tables to collect a particular collection of data. It is always necessary to apply lossless decomposition. By employing the decomposed relations, we can ensure that the data or information that was contained in the original connection can be precisely rebuilt. An improper breakdown of the partnership may ultimately result in issues like information loss.

2.4.1 Types of Decomposition

There are two types of Decomposition

1. Lossless Decomposition

A decomposition is considered lossless when joins from the decomposed tables allow one to reconstruct the original relation R. It's the choice that people prefer the most. In this way, the knowledge remains intact even after we break down the relationship. In the end, a lossless join would yield a result that was extremely close to the original relation.

For instance,

Consider "A" as the relational schema that contains an instance of "a." Consider that it is decomposed into: A1, A2, A3, ... An; with instance: a1, a2, a3, ...an, If a1 ⋈ a2 ⋈ a3 . . . .⋈ an, then it is known as 'Lossless Join Decomposition'.

2. Lossy Decomposition

As the name implies, whenever we break down a relation into multiple relational schemas, we inevitably lose data or information when trying to recover the original relation.

The following qualities must be present for decomposition:

1. Lossless Decomposition Is Required

Information from a decomposed relation must never be lost, so decomposition must always be lossless. By doing this, we are given a guarantee that, when the relations are joined, the join will ultimately result in the same relation as its decomposed form.

2. Maintenance of Dependence

Dependency is a crucial restriction on a database; each dependency must be satisfied by at least one deconstructed table. Functional dependence between the two sets is required if {P → Q} is true. Therefore, if both of these are set in the same relation, it becomes quite helpful when testing the dependence. Only when the functional dependence is maintained can we use this decomposition characteristic. Furthermore, we may examine different changes with this attribute without having to compute the natural join of the database structure.

3. Insufficient Data Redundancy

It is also frequently referred to as a repetition of information or facts. This property states that data redundancy must not affect decomposition. Careless deconstruction may have negative

effects on the database's total data. We can easily attain the property of no redundant data when we execute normalisation.

2.5 Synthesis Approach

The synthesis approach is predicated on the notion that a database can be defined by a collection of characteristics and a set of functional dependencies. 3NF or BCNF relations, or fragments thereof, are then synthesised based on the set of dependencies.

- **Knowledge Check 1**

  Fill in the Blanks “

  1. A relation must be in first normal form in order to be in_____, and it cannot have any partial dependencies.
  2. If a relation R is in Third Normal Form and LHS is super key for every FD, it is said to be in _____.
  3. _____describes the process of dissecting a database table into different components or elements. “

- **Outcome-Based Activities 1**

  List all the types of normalisation with example.

2.6 3NF and BCNF

1. Third Normal Form

According to the principles of the Third Normal Form (3NF), no non-prime attribute should be transitively dependent on the relation's candidate key. When a relation attains the Second Normal Form (2NF) and lacks any non-key attribute transitively reliant on the primary key, or when there are no transitive dependencies, the relation is deemed to be in the third normal form (3NF). Furthermore, it must satisfy one of the following prerequisites:

- The functional dependency C -> D must be a part of the candidate key, with C being a super key and a prime attribute.

The adoption of 3NF aims to ensure data integrity and minimize data redundancy.

For instance: In the case of the relationship R(L, M, N, O, P), with functional dependencies L->M, MN->P, and PO->L: The candidate keys will be: {LNO, MNO, NOP}

- Closure of LNO = {L, M, N, O, P}
- Closure of MNO = {L, M, N, O, P}
- Closure of NOP = {L, M, N, O, P}

Given that the relation is already in 2NF and lacks a transitive dependency, it qualifies for 3NF. Additionally, no non-prime attribute derives from another non-prime attribute.

On the other hand, BCNF stipulates that if a relation possesses a minimal functional dependency (X -> Y), then X must be a super key. The transition from 3NF to BCNF is possible without losing any dependencies.

For example: In the relationship R(A, B, C, D) with functional dependencies A->B, A->C, C->D, C->A: The candidate keys will be: {A, C}

- Closure of A = {A, B, C, D}
- Closure of C = {A, B, C, D}

This relation is in BCNF since it already satisfies 3NF (no non-prime attribute derives from another non-prime attribute), and there exists a candidate key on the left side of the functional dependency.

2. Lossless Join and Dependency Preserving Decomposition

A lossless join decomposition of a relational schema R into two schemas with attribute sets X and Y ensures that, for each instance of R satisfying the dependencies in F, a natural join of X and Y will produce the same instance r. This method divides a relation into two or more relations while guaranteeing that neither extra nor fewer tuples are generated, nor any loss of information occurs during the decomposition.

Every dependency in the dependency preservation must be satisfied by at least one decomposed table. The dependencies of a relation R must either be a component of relation R1 or relation R2, or they must be derived from the union of functional dependencies of relation R1 and relation R2 when R is decomposed into R1 and R2.

3. Fourth Normal Form and Multi-valued Dependency

In relational database management systems, the term "4NF" refers to a relation that adheres to Boyce-Codd Normal Form (BCNF) and lacks any multi-valued dependencies. A multi-valued dependency occurs when multiple values of attribute B exist for a single value of attribute A.

A table exhibits multi-valued dependency (MVD) when several rows have MVD, indicating the existence of multiple additional rows in the same table. Therefore, 4NF cannot be achieved with a multi-valued dependency, as it necessitates at least three attributes in a table.

An MVD implies that different values of attribute "y" may exist for different values of attribute "x". The format of MVD can be represented as: x -> -> y

For a legal relation q(Q), if all pairings of tuples p1 and p2 in q are such that one attribute, say x, can determine another attribute, y, then MVD exists.

4. Fifth Normal Form and Join Dependency

Fifth Normal Form (5NF), also known as the project-join normal form, is attained when a relation is in 4NF and cannot be losslessly decomposed into smaller tables. Additionally, 5NF is achieved if the candidate key determines every join dependency in the relationship.

Join dependency is based on the concept of 5NF and is analogous to functional or multi-valued dependencies. It can only be satisfied if the relation is a join of a certain number of projections. A relation exhibits join dependency if it can be reconstructed by connecting multiple other tables, each containing a subset of the original table's attributes. This constraint is closely associated with 5NF, as a relation can only achieve 5NF if it also satisfies 4NF and cannot be further decomposed.

- **Knowledge Check 2**

  State true and false for the following sentences.

  1. A table is considered a Multi-valued dependency if it can be recreated by simply connecting many other tables, each of which contains a subset of the table's attributes.
  2. The term "4NF" in relational database management systems refers to a relation that is in Boyce Codd Normal Form and does not have any Join-valued dependencies.
  3. The project-join normal form is another name for the 5NF (Fifth Normal Form)

- **Outcome-Based Activities 2**

List some examples of Multi-valued dependency and Join dependency.

2.10 Summary

- A functional dependence (FD) is a relationship between two attributes; in a database, this relationship is frequently between the primary key (PK) and other non-key attributes. If the value of attribute X affects the value of Y in a different way for each valid instance of X, then attribute Y in any relation R is functionally dependent on attribute X (often the PK).
- The closure of a functional dependency is the whole set of all attributes that may be functionally inferred from a certain functional dependency using the inference criteria known as Armstrong's criteria. "F+" can be used to denote the closure of a functional dependency if "F" is a functional dependency.
- Normalization is the process of organizing the data in the database.Normalization is used to remove redundancy from a connection or group of relations.Another reason for it is to eliminate undesirable features like Insertion, Updating, and Deletion Anomalies.
- The normalization procedure arranges the data in a database.To ensure data security and improve database flexibility by eliminating redundancy and inconsistent dependence, tables must be created and their associations must be defined according to rules.
- In a multi-valued dependency, a determinant is linked to a number of different values.The MVD does not exhibit modification abnormalities when isolated.
- A constraint called join dependence, often known as JD, is comparable to FD (functional dependency) or MVD (multi-valued dependency).Only when the relation in question is a join of a certain number of projections is JD satisfied. Thus, this kind of constraint is referred to as a join dependency.
- A functional dependence (FD) is an association between two qualities, most often the PK and other non-key properties in a database.Attribute Y is functionally dependent on attribute X for any relation R if the value of attribute X (often the PK) influences the value of attribute Y in a distinct manner for each legitimate occurrence of X.

2.11 Self-Assessment Questions

1. What is Functional Dependencies and its implications?
2. What isClosure rules?

3. Explain Normalisation.
4. What is Decomposition?
5. What is the Synthesis approach?
6. Explain 3NF and BCNF.
7. What is Lossless Join and Dependency Preserving?
8. What is Join Dependency?

2.12 References

- Atkinson, M.P. (1981) *Database*. Maidenhead: Pergamum InfoTech.
- Frank, L. and Helmersen, O. (1988) *Database theory and practice*. Wokingham, England: Addison-Wesley Publishing Company.
- *Database users: 2nd Toronto conference: Selected papers* (1990). Canadian Association for Information Science.
- *Database* (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) *Database*. Rockville, MD: Computer Science Press.

Unit 3

Relational Database Model

Learning Outcomes:

- Students will be capable of learning about the Relational Data Models
- Students will be able to understand the Concepts and Relational database constraints.
- Students will be able to understand the concepts of Database design using ER.
- Students will be able to understand about the EER to Relational Mapping and Relational Algebra.

Structure

- Outcome-Based Activities 2

3.5 Summary

3.7 Self-Assessment Questions

3.7 References

3.1 Relational Data Models

- The relational model (RM) represents a database as a collection of relations, where each relation is essentially a table of values. Each row within these tables comprises relevant data values, symbolizing real-world entities like people, places, objects, or connections. The utilization of table and column names aids in comprehending the significance of the values in each row. Despite the physical storage method, the logical organization of data remains independent in the relational model.
- The relational model for database management serves as a conceptual framework for representing and managing data within a database. Data within this model is structured into a series of two-dimensional tables or relations, proposed by E.F. Codd. Subsequently, after designing the conceptual database model using an Entity-Relationship (ER) diagram, it becomes imperative to translate it into a relational model feasible for implementation using various RDBMS languages such as Oracle SQL, MySQL, etc. Relational databases store data based on the relational model, where data is stored in relations or tables.

Think about the relation STUDENT, which has the Table 1 attributes ROLL NO, NAME, ADDRESS, AGE

ROLL NO	NAME	ADDRESS	AGE
1	RAHUL	RAIPUR	25
2	KIRTI	BILASPUR	22
3	SWATI	BHILAI	21
4	ROHAN	JAMSHEDPUR	24

Some of the popular Relational Database Management System are:

- DB2 and Informix Dynamic Server by IBM

- Oracle and RDB by Oracle
- SQL Server and Access by Microsoft

3.1.1 Characteristics of Relational Model

• Data is organized in rows and columns, typically represented as relations.

• The relational model describes the relationships between tables where data is stored.

• The relational model supports operations like data definition, manipulation, and transaction management.

• Each column is uniquely named and represents an attribute.

• Each row represents a single entity.

• Relation names are unique within the database.

• Each cell in a relation contains exactly one atomic (single) value.

• Attributes have unique names.

• The attribute domain defines the type of data a column can hold.

• Values within a tuple (row) are unique.

• Tuple order may vary across different instances.

3.1.2 Important terms related to the Relational Database

- Each column in a table represents an attribute, defining the characteristics of a relation.
- Examples of attributes include "Name," "Student Roll no," etc.
- Tables, also known as relations, store data in a tabular format within the relational model. Rows represent records, while columns represent attributes.
- A tuple refers to a single table row, representing a single record.
- Relation Schema encompasses the relation's name and its attributes.
- Degree indicates the total number of attributes in a relation.
- Cardinality refers to the total number of rows in a table.
- Columns define the range of values for a specific attribute.
- A relation instance in an RDBMS comprises a finite set of tuples.
- Duplicate tuples do not exist within relation instances.
- The relation key refers to one or more attributes that uniquely identify each row.

- Each attribute has a predefined value and domain, known as its attribute domain.

a. Concepts and Relational database constraints

Constraints are rules applied to a table's data columns, ensuring data integrity and reliability by filtering the types of data allowed into tables. These constraints can be imposed at both the column and table levels.

In the relational model, constraints are essential requirements that database data must meet to maintain validity. Before executing any database operation, such as insertion, deletion, or updating, these constraints are checked. If any constraint is violated, the operation will fail. Integrity constraints are of various types in DBMS, with three main categories:

1. **Domain Constraints**: These constraints ensure that attribute values adhere to the related domain and correct data type. Each attribute's value within a tuple must comply with domain requirements. Common data types such as integers, real numbers, characters, Booleans, and variable length strings are examples of this.
2. **Key Constraints**: Key constraints enforce uniqueness within a set of attributes, ensuring that each tuple has a unique identifier or key.
3. **Entity Integrity Constraints**: Entity integrity constraints guarantee the existence of a primary key in every table, preventing null values in primary key attributes.
4. **Referential Integrity Constraints**: These constraints maintain consistency between related tables, ensuring that foreign key values in one table correspond to primary key values in another, preventing orphaned records.

For example

Create DOMAIN Customer Name

CHECK (value not NULL)

For example

Eid	Ename	Phoneno
1	Rohan	9994678990,9690578999

Due to the fact thatthe Name in the relationship is a composite attribute and the Phone is a multi-valued attribute, the domain constraint is broken.

Key Constraints

Each relation in the database must contain at least one set of attributes that uniquely defines a tuple, known as keys. For instance, in a STUDENT relation, ROLL NO serves as an example of a key. This ensures that there are no duplicate tuples within the relation, making it a uniqueness constraint.

A relation may have multiple keys or candidate keys, where a candidate key is the minimal super key. From these candidate keys, one key is chosen as the primary key. While there are no strict rules for selecting the primary key, it is generally recommended to choose the candidate key with the fewest attributes.

The primary key cannot include null values, and therefore, the Not Null constraint is a requirement for the key attributes. This ensures that each tuple in the relation has a valid and non-null value for the primary key attributes.

The following are the two characteristics of a key:

- For each tuple, it must be different.
- It cannot contain any NULL values.

For Example

EID	ENAME	AGE
1	Rahul	22
2	Rohan	21
3	Rohit	23
4	Amar	21
1	Armaan	20

The first and last tuples in the above table have the same value for EID, which is 01, therefore breaking the key constraint.

Entity Integrity Constraints:

No primary key can have a NULL value, according to entity integrity constraints, because we need primary keys to uniquely identify each tuple in a relation.

EID	ENAME	AGE
1	Rahul	20
2	Rohan	21
44	Karan	18
NULL	Prem	23

In the relationship mentioned above, EID is designated as the primary key. As the primary key cannot contain NULL values, it is violated in the third tuple.

Referential IntegrityConstraints

Referential integrity ensures that an attribute in a relation can only contain values that exist in another attribute within the same relation or in another relation. This constraint is maintained through the use of foreign keys in DBMS.

A foreign key is a vital attribute within a relation that references attributes in other relations. It establishes a connection between different tables, enforcing relational integrity constraints. However, it is essential that the referenced attribute exists in the related table to maintain referential integrity.

EID	ENAME	DNO
1	Rohan	12
2	Rohit	22
4	Amit	14

Dno	Place
12	Jaipur
13	Bhilai
14	Raipur

The DNO of Table 1 is the foreign key, and the DNO of Table 2 is the primary key in the tables mentioned above.DNO = 22 cannot be used in the foreign key of Table 1 since it is not defined in Table 2's primary key. Referential integrity constraints are thus broken in this situation.

Features of Relational Database Model –

The relational database features are discussed briefly.

- Simplicity of Model The relational database model is simpler compared to other models, requiring straightforward SQL queries for data handling, without the need for complex processing or structuring.
- Ease of Use Users can efficiently retrieve required information using Structured Query Language (SQL) without facing the complexities often associated with databases.
- Accuracy Relational databases are well-structured, minimizing data duplication and ensuring accuracy in data representation.
- Data Integrity Relational Database Management Systems (RDBMS) ensure consistency across all tables, enhancing data integrity and reliability.
- Normalization Normalization breaks down complex data into manageable chunks, reducing storage size and ensuring uniformity in data structure, thus maintaining data integrity.
- Collaboration Multiple users can access and retrieve information simultaneously from the database, even during data updates.
- Security RDBMS systems offer robust security features, allowing only authorized users to access data directly, ensuring data confidentiality and integrity.

3.2.2 Advantages of the Relational Model

- It is simplethan the network and the hierarchical model.
- It is very easy and very simple to understand.
- The structure of the relational Model can be changed according to the requirements.
- It is Adaptable
- It is Safe.
- Data reliability
- Data reliability
- Applications for operations are simple.

3.2.3 Disadvantages of the Relational Model

- Limited Scalability for Huge Databases: Relational databases may encounter scalability issues when handling large volumes of data, which can impact performance.
- Complex Relationship Management: Managing relationships between tables can sometimes be challenging, requiring careful design and maintenance.
- Complexity and Usability: Relational databases can be complex to set up and maintain, requiring developers and programmers to invest significant time and effort in database management tasks.
- Maintenance Challenges: As data volume increases over time, maintaining a relational database can become more challenging, leading to higher maintenance costs.
- Cost of Setup and Maintenance: Setting up and maintaining a relational database system can incur significant costs, including licensing fees, hardware expenses, and ongoing maintenance costs.
- Performance Issues with Large or Distributed Databases: Large or distributed databases may experience latency and availability issues, affecting overall performance.
- Limitation in Representing Complex Relationships: Relational databases store data in tabular form, which may not adequately represent complex relationships between objects, posing challenges for applications that require intricate data models.
- Performance Degradation with Increased Complexity: As the number of tables and data complexity grows, relational databases may experience slower response times and increased query complexity, leading to performance issues during peak usage times.

3.2.4 Operations in Relational Model

The relational database model supports the following four fundamental update operations:

- Data is inserted into the relation using insert.
- To remove tuples from a table, use delete.
- You can modify an existing tuple to change the values of certain of its attributes.
- Select enables you to pick a certain set of facts.

Integrity restrictions set in the relational database structure must never be broken when one of these operations is used. If the tuple being deleted is referred to by foreign keys from other tuples in the same database, the delete operation may not be referentially sound.

- **Knowledge Check 1**

  Fill in the blanks with the following sentences.

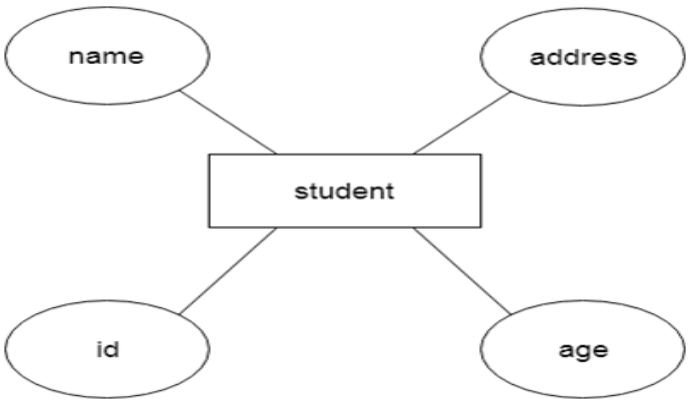
  1. Foreign Keys serve as the foundation for DBMS referential integrity constraints.
  2. _____are the requirements that must be met for database data in order for the Relational model to be valid.
  3. _____occurs when an attribute of a relation can only take values from an attribute of the same relation or from any other relation.

- **Outcome-Based Activities 1**

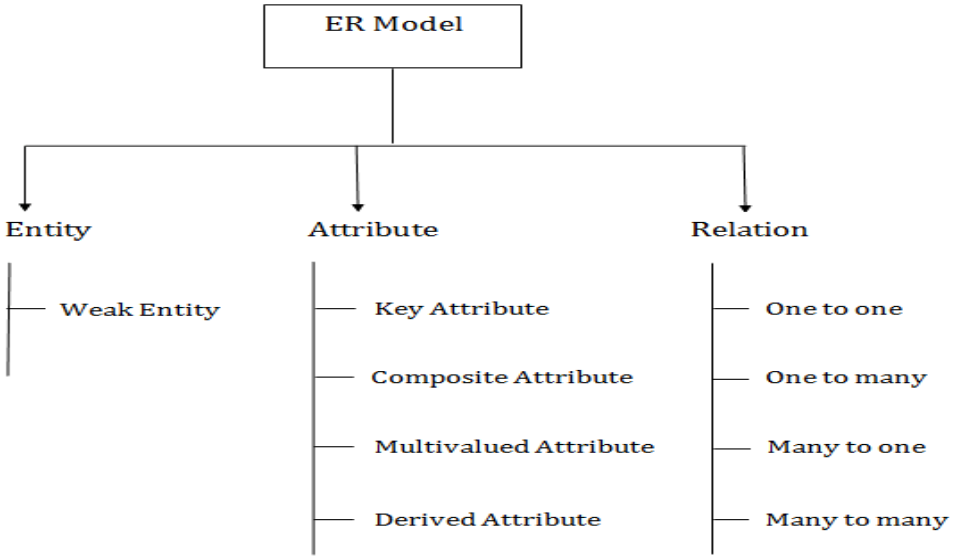
  List out all the Relational Database Constraints.

3.1 Database design using ER

- The Entity-Relationship (ER) model, often referred to as the ER model, operates at a high level, defining data items and relationships within a system. It serves as the foundation for conceptual database design, offering a clear and concise view of the data structure. An entity-relationship diagram (ER diagram) visually represents the database structure in ER modeling, illustrating the relationships between entity sets stored in the database.
- ER diagrams aid in describing the logical organization of databases by depicting entities, attributes, and relationships. Introduced by Peter Chen in 1971, ER diagrams establish a standard convention for conceptual modeling in relational databases and networks. In these diagrams, rectangles represent entities, ovals denote attributes, and diamond shapes represent relationships.
- For instance, consider creating a school database. In this scenario, a student would be an entity within the database, encompassing attributes such as residence, name, ID, and age. Additionally, attributes related to the student's address, such as city, street name, and pin code, may be treated as separate entities with a relationship to the address.



3.3.1 Components of the ER Model



3.3.2ER Diagram Symbols and Notations

The rectangle, oval, and diamond are the three primary symbols used in entity relationship diagrams to show relationships between objects, entities, and characteristics. In the ERD Diagram, several sub-components are based on the major elements. Using various ERD Symbols and Notations, an ER Diagram is a visual representation of data that shows how data are related to one another.

Rectangles: This entity-type symbol is used in entity relationship diagrams.

Ellipses: Symbols for attributes

Diamonds: Relationship kinds are represented by this symbol.

Lines: It connects entity types with other relationship types and attributes to entity types.

Primary key: highlighted qualities

Double Ellipses: Represent properties with multiple values

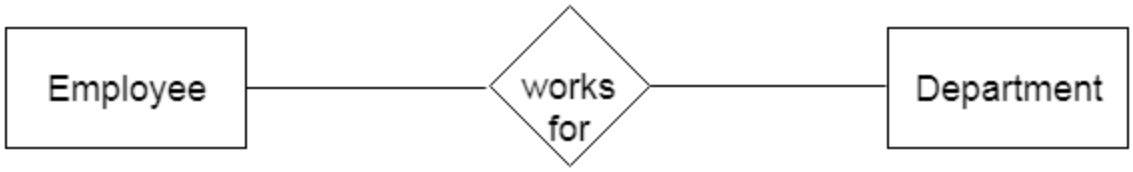
3.3.3 Components of the ER Diagram

This model is basically based on three basic concepts:

- Entities
- Attributes
- Relationships

Entities

Any item, class, person, or location can be considered an entity. Rectangles in the ER diagram can be used to represent an entity. Take a company as an example; a manager, a product, an employee, a department, etc., can all be considered as separate entities.



A weak entity is one that is dependent on another entity. The weak object is devoid of any unique key attribute. The weak entity is represented by a double rectangle.

Attributes

It is a relationship-type or entity-type single-valued property. A lecture, for instance, might have the following attributes: time, date, duration, location, etc. In instances of ER Diagrams, an

Ellipse is used to represent an attribute.The property of an entity is described by the attribute. Eclipse is a symbol for an attributes.

Types of Attributes

1.Key Attribute

The key attribute of an entity represents its primary attributes, often denoted as the main key. Key attributes are typically underlined within an ellipse to signify their importance. Simple attributes, on the other hand, cannot be further subdivided and represent basic characteristics such as a student's telephone number, known as atomic values.

Composite attributes are composed of multiple other attributes and are depicted within an ellipse with ellipses connecting their components. For example, a name attribute may comprise first name, middle name, and last name.

Multi-valued attributes allow several values to be assigned to an attribute, represented by a double oval. An example is a student having multiple phone numbers.

Derived attributes are created from other attributes and are denoted by a dashed ellipse. For instance, the date of birth can be derived from the age attribute.

Relationships between entities are represented by rhombuses or diamonds. One-to-one relationships entail only one instance of each entity involved. For example, a female can marry one man, and vice versa.

- **One-to-Many Relationships**

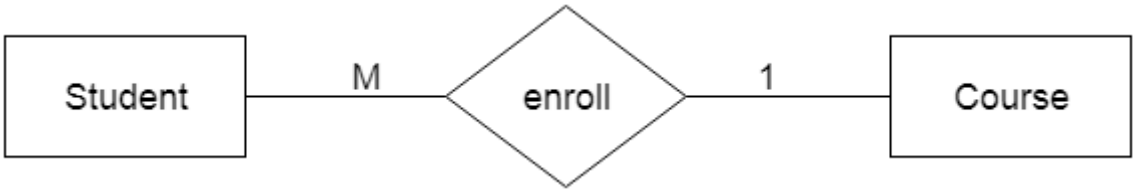
  A one-to-many relationship exists when there is only one instance of the entity on the left and multiple instances of the entity on the right that are associated with the relationship.

  For instance, a scientist can come up with a number of inventions, but only one scientist can really create them.

- **Many to One Relationships**

  A many-to-one relationship is one in which there are multiple instances of the left-hand entity and only one instance of the right-hand entity that are linked to the relationship.

  As an illustration, a Student only enrolls in one course, while a course can have numerous students.

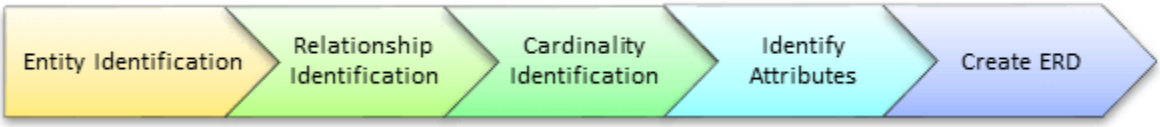


- **Many-to-Many Relationships**

  A many-to-many relationship is one in which there are multiple instances of both the entity on the left and the entity on the right that are associated with the relationship.

  For instance, numerous projects may allocate employees to one employee, and vice versa.

3.3.4 Create an Entity Relationship Diagram (ERD)



3.3.5 Advantages of ER Diagram

- ER diagrams provide a clear understanding of terminology related to entity relationship modeling.
- They outline the relationships between tables and the fields each table will include.
- Entities, properties, and relationships are described in detail.
- ER diagrams facilitate the quick creation of databases as they can be converted into relational tables.

- Database designers use ER diagrams to plan out the implementation of data in software applications.
- ER diagrams help designers better understand the data to be stored in the database.
- They allow users to interact with the logical structure of the database.
- The ER model enables the creation of a database design.
- ER diagrams are widely used graphical tools for modeling data and represent a GUI representation of a database's logical structure.
- They aid in identifying the entities present in a system and their relationships.

3.4 ER to Relational Mapping and Relational Algebra

3.4.1 ER to Relational Mapping

When visualized into diagrams, the ER Model provides an excellent overview of entity-relationship that is simpler to comprehend. Relational schema can be created using ER diagrams, which means that they can be mapped to relational schema. An approximation schema can be built; however, we are unable to import all ER constraints into the relational model.

The following are the general steps involved in converting an E-R diagram into a relational model:

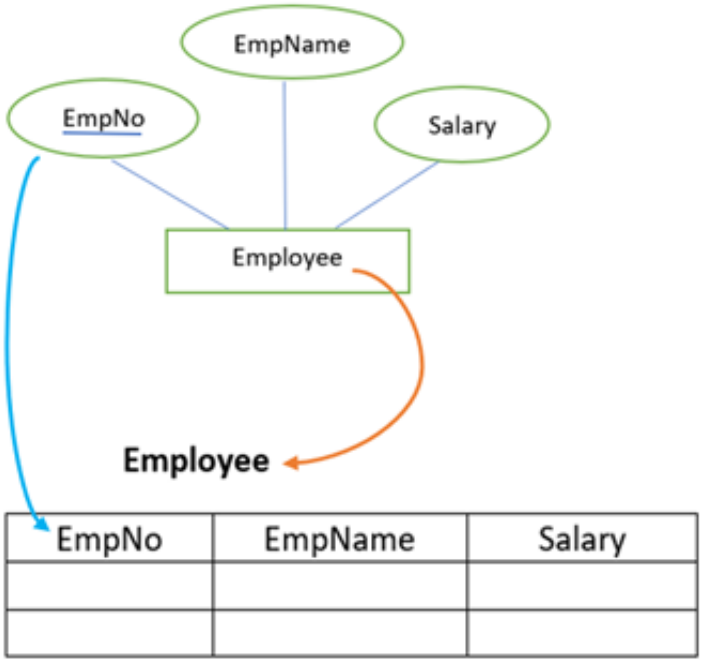
- Mapping of an entity set into database relations (tables).
- The attributes of an entity are among the attributes of a table.
- The main key of the relation is the key attribute of an entity

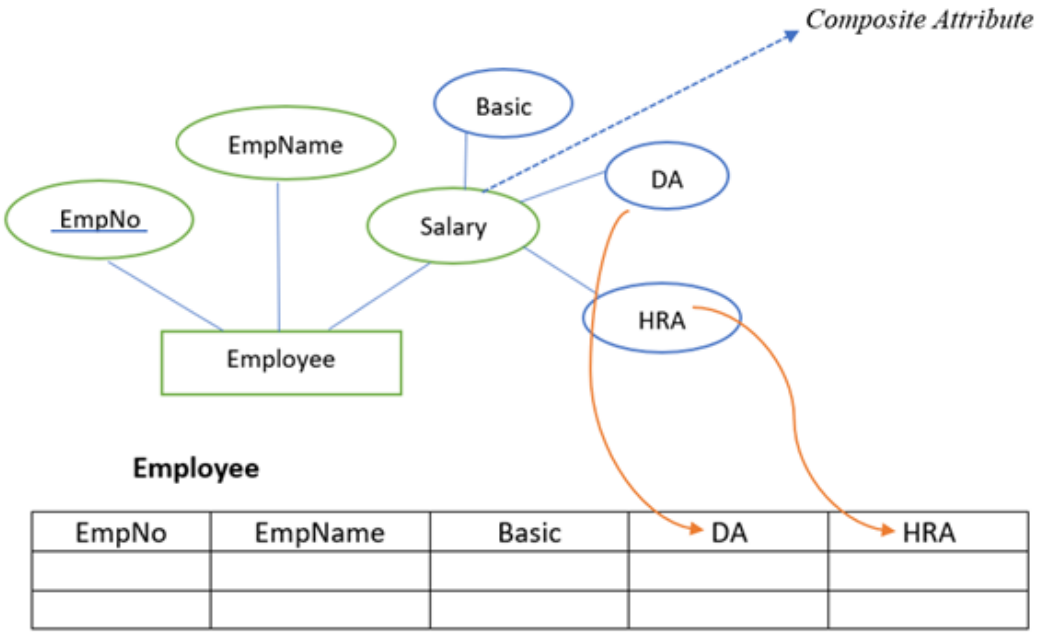
ER-to-Relational Mapping Algorithm

1. Step 1: Mapping of Regular Entity Types.
2. Step 2: Mapping of Weak Entity Types.
3. Step 3: Mapping of Binary 1:1 Relation Types.
4. Step 4: Mapping of Binary 1: N Relationship Types.
5. Step 5: Mapping of Binary M:N Relationship Types.
6. Step 6: Mapping of Multi-valued attributes.

Mapping Process

- Create table for weak entity set.
- Add all its attributes to table as field.
- Add the primary key of identifying entity set.
- Declare all foreign key constraints.
- Create tables for all entities at higher levels.
- Create tables for entities at a lower level.
- In the table of lower-level entities, add the main keys of higher-level entities.
- Add all additional properties from lower-level entities to lower-level tables.
- Declare the lower-level table's main key as well as the primary key for the higher-level table.
- Declare Constraints on the foreign keys.





3.4.2 Relational algebra

Relational algebra is a procedural query language that operates on instances of relations, accepting them as input and producing instances of relations as output. Queries in relational algebra are executed using a set of operators, which can be both unary and binary. These operators take relations as input and output, treating them recursively. The intermediate results obtained through the recursive application of relational algebra are also regarded as relations:

- Select
- Project
- Union
- Set difference
- Cartesian product
- Rename

1. Select Operation (σ)

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

Symbol– $\sigma_p(r)$

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

For example −

$\sigma_{subject = "database"}(Books)$

2. Project Operation (∏)

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

Notation − $\prod_{A1, A2, An} (r)$

Where A_1, A_2 , A_n are attribute names of relation **r**.

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

For example −

$\prod_{subject, author} (Books)$

3.Union Operation (∪)

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

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

4. Set Difference (−)

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

Notation − **r** − **s**

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

$\prod_{author} (Books) - \prod_{author} (Articles)$

5. Cartesian Product (X)

Combines information of two different relations into one.

Notation − r X s

Where **r** and **s** are relations, and their output will be defined as −

$r X s = \{ q\ t \mid q \in r \text{ and } t \in s\}$

$\sigma_{author = 'tutorialspoint'}(Books\ X\ Articles)$

6. Rename operation (ρ)

Even though the outcomes of relational algebra are relations, they lack names. We can rename the output relation using the rename operation. The little Greek character rho is used to indicate the "rename" procedure.

Notation − ρ_x (E)

Where the result of expression **E** is saved with name of **x**.

Additional operations are −

- Set intersection
- Assignment
- Natural join

- **Knowledge Check 2**

  State True and False for the Following Sentences.

  - A Composite attribute is an attribute that can be constructed from multiple other attributes. It is often represented using a dashed ellipse.
  - In a many-to-many relationship, multiple instances of both entities on the left and entities on the right are connected through the relationship.
  - Derived attributes are attributes that can have multiple values assigned to them.

- **Outcome-Based Activity 2**

  List out all the steps of ER to Relational Mapping and Relational Algebra.

3.5 Summary

The relational model within Database Management Systems (DBMS) is an abstract framework utilized for organizing and managing data stored in a database. It structures information into two-dimensional tables or relations, where each row represents an entity and each column denotes its attributes.

Relational mapping is a method that converts object data member types into corresponding representations in a relational database (SQL) data source. This process facilitates the mapping of an object model to a relational data model.

In addition to domain constraints, relational databases employ various constraints to further constrain data. The five primary types of constraints in a relational database are:

- Domain constraints
- Key constraints
- Entity Integrity constraints
- Referential integrity constraints
- Tuple Uniqueness constraint

Relational algebra is a procedural query language that accepts instances of relations as input and generates instances of relations as output. It employs operators, including both unary and binary, to execute queries effectively.

3.6 Self-Assessment Questions

1. What are Relational Data Models?
2. What are Relational Database Constraints?
3. What is ER Diagram?
4. What is database design using ER?
5. What is Relational Mapping?
6. What is Relational algebra?
7. Explain ER to Relational Mapping.
8. Explain ER to Relational algebra.

3.7 References

- Atkinson, M.P. (1981) Database. Maidenhead: Pergamum InfoTech.
- Frank, L. and Helmersen, O. (1988) Database theory and practice. Wokingham, England: Addison-Wesley Publishing Company.
- Database users: 2nd Toronto conference: Selected papers (1990). Canadian Association for Information Science.
- Database (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) Database. Rockville, MD: Computer Science Press.

Unit 4

Query Processing

Learning Outcomes:

- Students will be capable of learning about the Basics of Query Processing.
- Students will be able to understand the Processing of Joins.
- Students will be able to understand the concepts of Materialized Vs Pipelined Processing.
- Students will be able to understand about the DB transactions and ACID Properties.
- Students will be able to learn about the Interleaved Executions and Schedules.
- Students will learn about the Serializability.
- Students will understand the concept of Database Recovery and Backup.

Structure

4.1 Basics of Query Processing

Query processing is the process of converting high-level queries into low-level expressions that are often utilized at the file system's physical level, optimizing the query, and actually running the query to obtain the desired outcome.

Query processing is the process of taking data out of a database. The process of retrieving data from the database during query processing involves many phases.

The involved actions are:

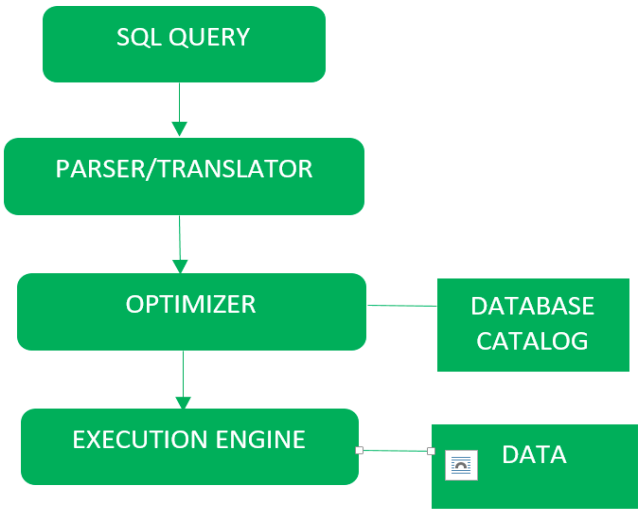
Parsing and translation represent the initial steps in query processing. High-level database languages like SQL are utilized to translate user queries into phrases that can be further applied at the physical level of the file system. These queries undergo evaluation along with various query-optimizing modifications. However, while SQL is an optimal choice for human readability, it may not be entirely suitable for the system's internal representation of queries.

For the internal representation of queries, relational algebra serves as an effective approach. The query parser within the system functions similarly to a translation process during query processing. It first assesses the syntax of the query, then checks the database for the relation name, tuple, and the required attribute value. A parse-tree, or tree of the query, is created by the parser, which is then converted into a relational algebraic form. Queries utilizing views are entirely replaced in this process.

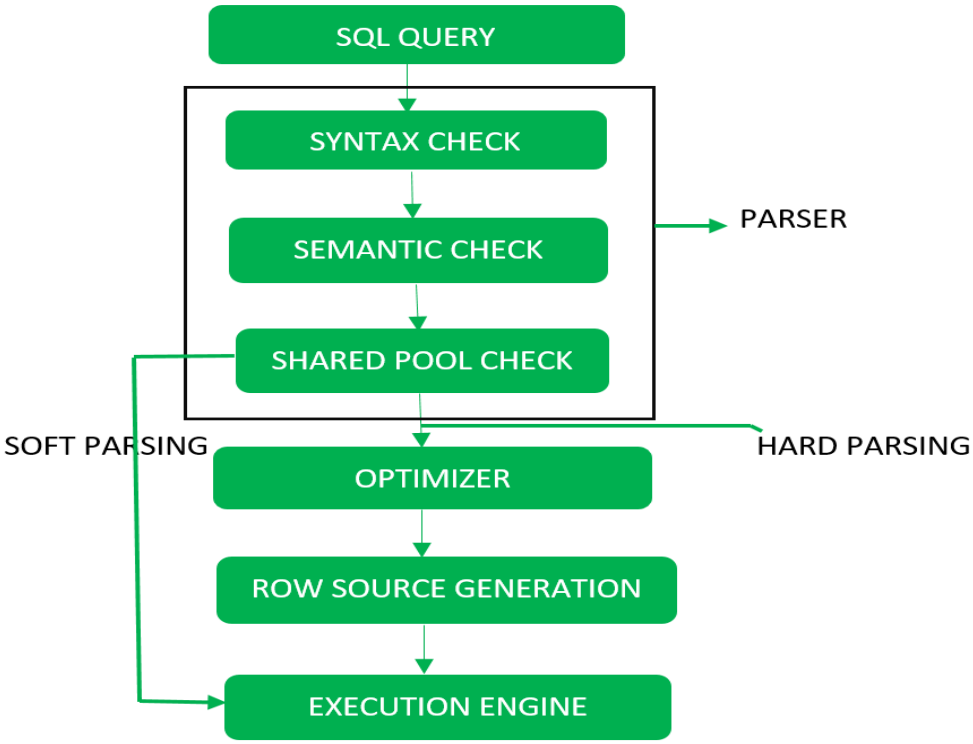
Optimization is a crucial stage in query processing. Different types of queries incur varying costs during examination. Although users are not obligated to write their queries efficiently, the database system is responsible for generating a cost-effective query evaluation plan to enhance efficiency. Query optimization refers to the task performed by the database system in this regard. The query optimizer analyzes the estimated cost of each action before optimizing a query, taking into account execution costs, memory allocations, and other relevant factors.

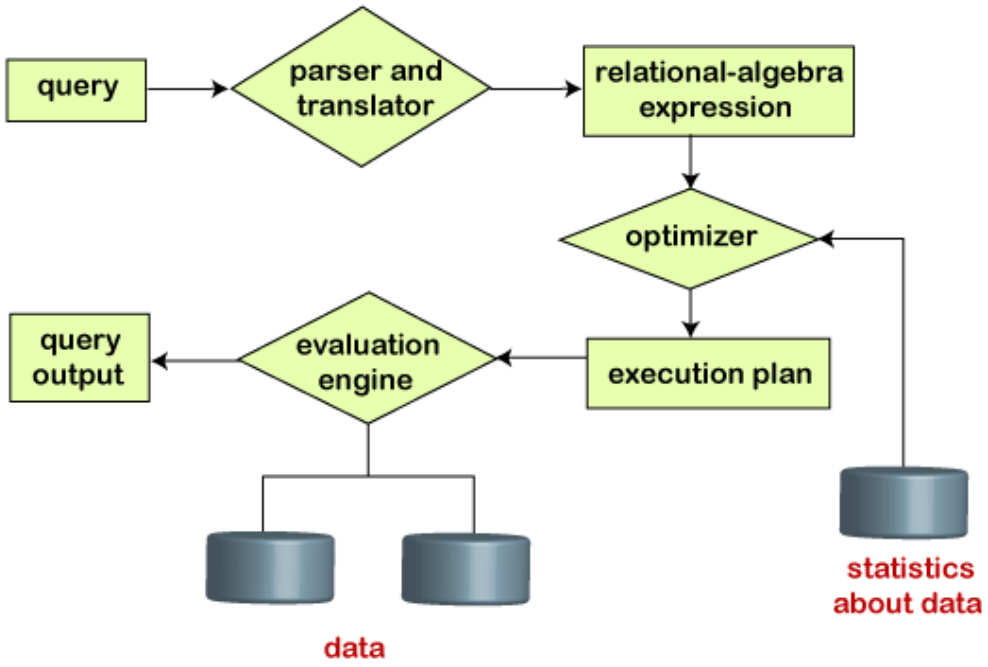
Evaluation, the final stage, involves annotating criteria for defining and evaluating each operation on the translated relational algebra expression. Once the user inquiry has been translated by the system, a query evaluation plan is executed. The system then evaluates the query and generates the query output based on the chosen evaluation plan.

Block Diagram



Detailed Diagram





Steps in query processing

4.2 Processing of Joins

A join in a database management system (DBMS) is a binary action that combines selection with a join product in one statement. A join condition is used to allow data from two or more DBMS tables to be combined. In database management systems, tables are linked together using primary keys and foreign keys. Related tuples from different relations are merged via a join operation if and only if a certain join condition is satisfied. It is denoted by ⋈.

Employee

EMP_Code	EMP Name
101	John
102	Rahul
103	Rohit

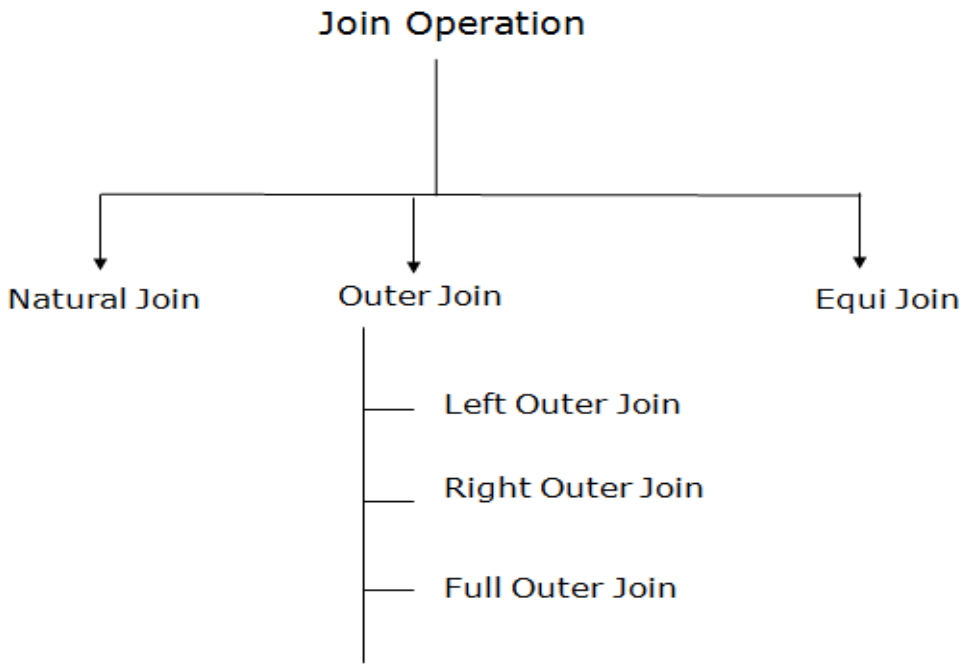
Salary

EMP_CODE	SALARY
101	40000
102	50000
103	35000

Operation: (EMPLOYEE ⋈ SALARY)

EMPCODE	EMPNAME	SALARY
101	**John**	**40000**
102	**Rahul**	**50000**
***103**	**Rohit**	**35000**

Types of Joins



4.3 Materialized vs Pipelined Processing

Although both methods are utilized for evaluating expressions with numerous operations, they don't differ much from one another.

The table below lists the difference points.

Pipelined Processing	Materialized
It is described as the contemporary method of assessing several procedures.	It is described as the conventional method of assessing several procedures.
The outcomes of the assessed activities are not stored using any temporary relations.	The outcomes of the assessed activities are stored using temporary relations. Thus, it requires additional I/O and temporary files.
As a result of its speedy results generation, it is a more effective method of query evaluation.	It is less efficient than pipelined Processing as it takes time to generate the query results.
For producing outputs, memory buffers are needed often. Thrashing will result from insufficient memory buffers.	For query evaluation, it doesn't have any more stringent memory buffer requirements.
The expense of query evaluation is optimized. Since the cost of reading and writing temporary storages is not included.	The overall cost includes the cost of operations as well as the cost of reading and writing findings to the temporary storage.
Performance issues if trashing happens	In materialization, trashing doesn't happen. Hence, materialization performs better in these circumstances.

4.4 DB transactions

A group of tasks collectively referred to as a transaction is an activity treated as a cohesive unit that either occurs entirely or not at all. In the context of database management systems (DBMS), a database transaction is an independent, consistent, and reliable unit of work performed against a database. It encompasses any modification made to the database and consists of a sequence of actions that are executed as a single logical unit, ensuring that either all operations are completed or none are. Though the state of the database may temporarily be inconsistent during the execution of a transaction, changes are only applied when the transaction is committed or concluded.

For instance, consider a simple transaction involving a bank employee transferring Rs. 500 from account A to account B. This transaction involves multiple low-level operations, such as debiting

the amount from the source account and crediting it to the destination account. It is crucial that the entire process is successfully executed; halting midway would result in an undesirable state where the money is deducted from one account but not deposited into the other.

In modern databases, transactions are utilized to prevent partial data writes and ensure data integrity. However, the fundamental principle remains constant: transactions guarantee the integrity of the data being manipulated.

4.5 ACID properties

ACID properties, which stand for Atomicity, Consistency, Isolation, and Durability, define the criteria that transactions must satisfy to ensure correctness, completeness, and data integrity within a database system.

- Atomicity ensures that transactions are indivisible units of work, meaning that either all operations within a transaction are executed successfully or none are.
- Consistency mandates that any attempt to commit an invalid change will fail, reverting the system to its previous valid state. Changes can only occur if the new state of the system remains valid.
- Isolation ensures that each transaction is executed as if it were the only one in the system, preventing interference from concurrent transactions. Changes made by one transaction are isolated from other transactions until they are completed.
- Durability guarantees that once a transaction is committed, its changes are permanent and will not be lost, even in the event of system failures or crashes. The system ensures that committed transactions are preserved and persisted, providing reliability and data consistency.

- **Knowledge Check 1**

  Fill in the Blanks “

  1. The property of _____states that all transactions will be carried out and processed as if they are the only transactions in the system in a database system when several transactions are being executed concurrently and in parallel.
  2. A state change can be represented by a transaction. Ideal transactions contain the four characteristics, referred to as _____ property.

3. An autonomous, coherent, and dependable unit of work that is conducted against a database using a database management system (or a system comparable to it) is referred to as _____. ”

- **Outcome-Based Activity 1**

  List out all the acid Properties of Database Transactions.

4.6 Interleaved Executions

During a single-query execution, interleaved execution modifies the unidirectional boundary between the optimization and execution phases and enables plans to adjust in light of the updated estimates. Many users frequently share a DBMS.To reduce the time it takes for these users' queries to execute, transactions from these users can be interleaved.

Users do not have to wait for other users' transactions to finish completely before starting their own transaction because queries can be interspersed.

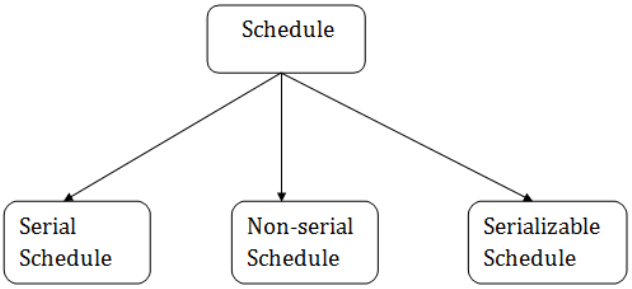
The problem occurs when two different database transactions interleave (i.e., execute read/write operations on the same database items concurrently), which causes the items' values to be erroneous and the database to become inconsistent.

The committed transactions from some anomalies associated with interleaved execution (DBMS) could be executed against a consistent database and leave it in an inconsistent state. If at least one of the operations is a write, then two actions on the same data object conflict.

The purpose of interleaved execution is to recompile and run the remainder of the query using the result and result cardinality from a portion of the query that may be executed independently.

4.7 Schedules

A schedule refers to a set of actions taken from one transaction to the next. It serves to maintain the sequential order of each individual transaction's operations.



1. Serial Schedule

A sort of schedule known as a serial schedule requires that one transaction be finished before beginning another. When the first transaction in the serial schedule completes its cycle, the next transaction is carried out.

For instance:

Assume there are two transactions with operations, T1 and T2.

The following two outcomes are possible if there is no interleaving of operations:

Complete all of the T1 operations, which were then followed by all of the T2 operations. Complete all of the T1 operations, which were then followed by all of the T2 operations.

Schedule A in the accompanying (a) figure depicts a serial schedule with T1 and T2 coming after each other. Schedule B is depicted in the supplied (b) figure as a serial schedule, with T2 coming after T1.

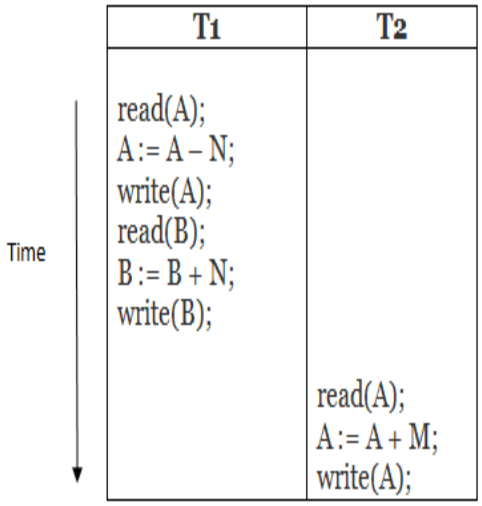
2. Non-serial Schedule

There will be a non-serial schedule if the interleaving of operations is permitted. It contains a wide range of potential orders in which the system could carry out the many transactions' separate processes. Schedules C and D in the accompanying figures (c) and (d) are non-serial schedules. The operations are interleaved.

3. Serializable schedule

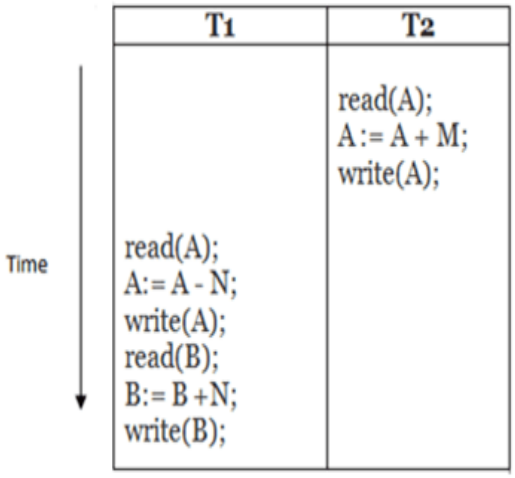
Finding non-serial schedules that permit the transaction to execute concurrently without interfering with one another is done using the serializability of schedules. When the transaction's executions have their activities interleaved, it shows which schedules are correct. If the outcome of a non-serial schedule is the same as the outcome of its serially executed transactions, the schedule can be serialized.

(a)



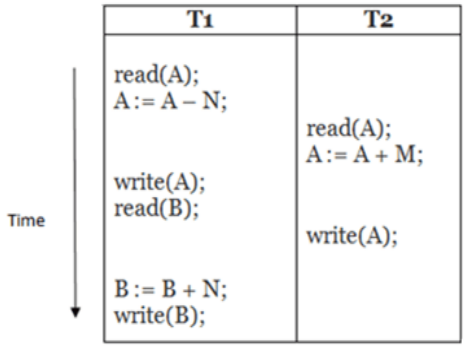
Schedule A

(b)



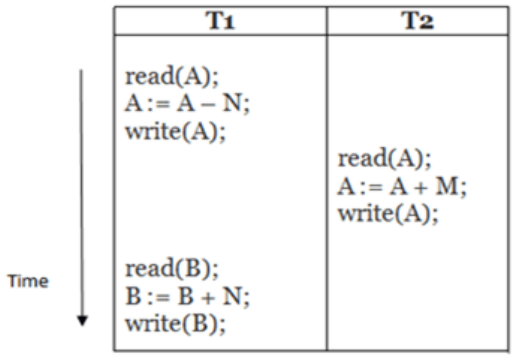
Schedule B

(c)



Schedule C

(d)



Schedule D

Schedules A and B are serial schedule. On-serial schedules include Schedule C and Schedule D.

4.8 Serializability

A system's serializability is a feature that describes how various processes use shared data. When there is no execution overlap and the outcome is the same whether the operations are carried out in any particular order, the system is said to be serializable.

A system's serializability refers to how various processes interact with shared data.If a system's outcome is the same as if the operations were carried out in a sequential order with no overlap in execution, the system is serializable.

Data can be locked using a database management system (DBMS) to prevent access from other processes while it is being read or written.

Types of serializability

There are two types of serializability −

1. **View serializability**

   When viewed similarly to a serial timetable, a schedule has view-serializability.

   It adheres to the following rules:

   - After T1 has finished reading A's initial value, A's initial value is also read by T2.
   - T1 reads the value that was written by T2, and T2 then reads the value that was also written by T1.
   - Following T1's write operation, T2 also uses the write operation as the final value.

2. **Conflict serializability**

   Similar to some serial execution, it arranges any incompatible operations. If two operations share the same data item and one of them is a write operation, they are said to conflict.

   It implies

   - ✓ Readi(x) readj(x) - non-conflict read-read operation.
   - ✓ Readi(x) writej(x) - conflict read-write operation.
   - ✓ Writei(x) readj(x) - conflict write-read operation.
   - ✓ Writei(x) writej(x) - conflict write-write operation.

4.9 Concept of Database Recovery and Backup

Backup and recovery is the process of creating and maintaining copies of data that may be used to protect organizations against data loss. This has also been referred to as operational recovery. In order to ensure safety, database backup often comprises building and keeping a copy of the database's contents on a backup server. Together with the database data, transaction logs are also included in the backup as without them, the database data would be worthless.

In the unlikely event that the original database is lost or corrupted, a backup copy of the database is required. With the help of this backup, the database may be returned to its initial condition.

Backup Methods

The various database backup techniques include:

Complete Backup - This method is time-consuming since a complete copy of the database, including all data and transaction records, is created.

Transaction Log - In this method, only the transaction logs are stored as a backup. The prior transaction log information is removed after creating a new backup record in order to keep the backup file as short as feasible.

Differential Backup-It stores both the data and the transaction records, making it identical to full backup in that regard. Unfortunately, the backup only contains the data that has changed since the last complete backup. Differential backup results in smaller files as a result. **"**

Recovery Methods

For database recovery, there are essentially two ways.

Which are:

1.**Log-based recovery** - In log-based recovery, each database transaction log is stored in a secure area so that in the case of a system failure, the database may retrieve the data. All log data, including the transaction's time, contents, etc., should be saved prior to the transaction being executed.

2. Shadow paging -With shadow paging, the data from the transaction is automatically saved for future reference after it has been finished. The database will not reflect changes made by the system if it crashes in the middle of a transaction.

- **Knowledge Check 2**

  State True and False for the following Sentences.

  1. With shadow paging, the data from the transaction is automatically saved for future reference after it has been finished.
  2. Schedules are the process of creating and maintaining data copies that may be utilized to protect companies against data loss.
  3. A schedule refers to a set of actions taken from one transaction to the next.

- **Outcome-Based Activity 2**

  List out all types of DBMS Schedules.

4.10 Summary

- During query processing, high-level queries undergo transformation into low-level expressions, representing a systematic procedure that occurs at various stages, including the physical level of the file system, query optimization, and actual execution of the query to produce the result. This process requires a fundamental grasp of relational algebra and file organization principles.
- In a database management system, a transaction is a discrete unit of labor or logic that may consist of many operations. Any logical computation done consistently is called a database transaction.
- In the context of transaction processing, the abbreviation ACID stands for atomicity, consistency, isolation, and durability—the four fundamental properties of a transaction. Updates to the data are treated as if they were one continuous activity.
- During a single-query execution, interleaved execution modifies the unidirectional boundary between the optimization and execution phases and enables plans to adjust in light of the updated estimates.
- A schedule refers to a set of actions taken from one transaction to the next. It serves to maintain the sequential order of each individual transaction's operations.
- A system's serializability refers to how various processes interact with shared data. A system is serializable if the result is the same as if the operations were performed in a non-overlapping, sequential manner.

4.11 Self-Assessment Questions

1. What is query Processing?
2. Explain the Processing of joins.
3. What is the difference between Materialized and pipelined processing?
4. What is DB transactions?
5. Explain ACID Property.
6. What is Interleaved Executions?
7. What isthe Schedules?
8. What is the concept of database backup and recovery?

4.12 References

- Atkinson, M.P. (1981) Database. Maidenhead: Pergamum InfoTech.
- Frank, L. and Helmersen, O. (1988) Database theory and practice. Wokingham, England: Addison-Wesley Publishing Company.
- Database users: 2nd Toronto conference: Selected papers (1990). Canadian Association for Information Science.
- Database (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) Database. Rockville, MD: Computer Science Press.

Unit 5

Relational Database Design Using PLSQL

Learning Outcomes:

- Students will be capable of learning about the PLSQL, Datatypes, and Language Structure.
- Students will be able to understand about the Controlling of Program Flow and Conditional Statements.
- Students will be able to understand the concepts of Loops and Stored Procedures.
- Students will be able to understand aboutthe Stored Functions.
- Students will be able to learn about the Handling Errors and Exceptions.
- Students will learn about the Cursors and Triggers.

Structure

5.1 Introduction to PLSQL

PL/SQL is a procedural language designed to incorporate SQL statements within its syntax. Program units written in PL/SQL are compiled by the Oracle Database server and stored within the database, allowing for optimal efficiency at runtime.

Developed by Oracle Corporation in the late 1980s, PL/SQL serves as a procedural extension language for SQL and the Oracle relational database. It is a block-structured language, meaning logical blocks within PL/SQL programs can be further subdivided into sub-blocks.

PL/SQL has been integrated with Oracle databases since version 7, with its capabilities expanding with each new release. While closely related to SQL, PL/SQL introduces additional programming constructs not found in SQL. Unlike SQL, PL/SQL is case-insensitive, except for string and character literals.

PL/SQL text is composed of lexical units, categorized into delimiters, identifiers, literals, and comments. Some noteworthy facts about PL/SQL include its high performance, portability across different platforms, integration with SQL*Plus command-line interface, and its support by other databases like IBM DB2 and Times Ten's in-memory database.

Overall, PL/SQL provides a robust and versatile environment for developing transaction processing applications, offering an integrated and interpreted programming environment that is independent of the operating system.

- **Features of PL/SQL**

The following characteristics of PL/SQL include

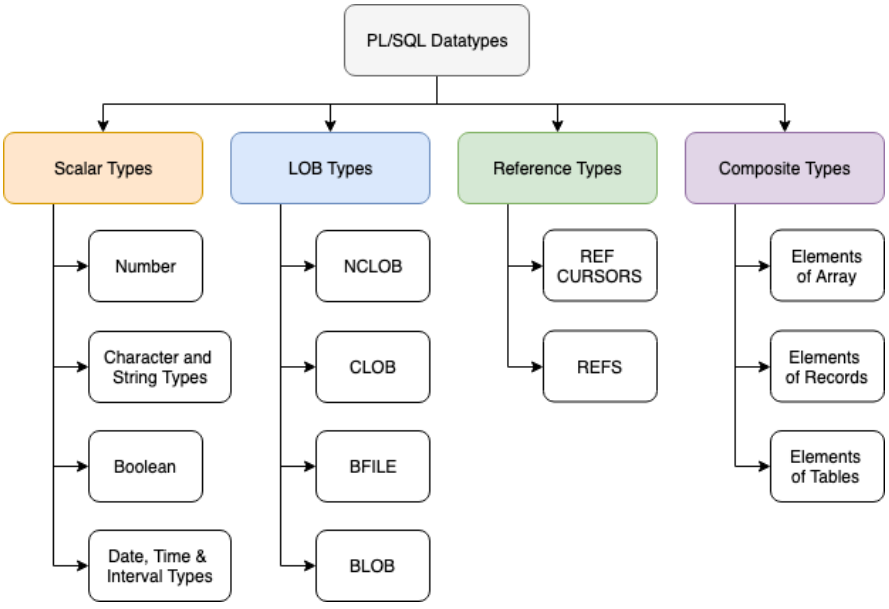
1. SQL and PL/SQL work together quite closely.
2. It provides thorough mistake checking.
3. It provides a variety of data types.
4. It provides a selection of coding structures.
5. Using its functions and procedures, it promotes organized programming.
6. Programming that is object-oriented is supported.
7. It facilitates the creation of server pages and web applications**.**

5.2 PLSQL: Datatypes, Language structure

- **Datatypes**

Many predefined datatypes are offered by PL/SQL.Integer, floating point, character, Boolean, date, collection, reference, and big object (LOB) types are only a few examples.You can also create your own subtypes using PL/SQL. As with any other programming language, PL/SQL datatypes can be utilized in the PL/SQL block in addition to writing SQL queries. When any PL/SQL code block is executed, the provision of a datatype tells Oracle how to store and process any data. Datatype identifies the type of data being utilized, such as whether it is a single character, a word (string), or another type of data. Depending on the type of data required, the following datatypes can be utilized in PL/SQL:

- **Scalar Types**-Basic data types known as scalar types often carry a single value, such as a single digit or a string of characters. In the image above, there are four categories for scalar types: number types, character and string types, Boolean kinds, date and time types, etc.
  1. **LOB Types**: This datatype is used to describe the location of huge items like text files, pictures, and other types of objects that are typically not stored outside of a database.
  2. **Reference Types:** This sort of data is used to store pointer values, which typically store the addresses of other programme components.
  3. **Composite Types:**Last but not least, as the name implies, this form of data is an amalgamation of independent data that can be handled and processed separately.



1.NUMBER (p, s)

Range: **p= 1 to 38 s= -84 to 127**

This datatype is generally used for storing numeric data. Here, p is precision s is scale.

Example:

Age NUMBER (2),where**Age** is a variable that is used to store**two**digits

2.CHAR(size)

Range: 1 to 2000 bytes

An alphabetical string of a specific length is stored using this datatype.

Single quotations are used to indicate its value. It occupies the entire amount of memory that has been specified, even if the space is not being used by the data.

Example: -rank CHAR (10), where **rank** is a variable that can store upto 10 characters.

3.VARCHAR(size)

Range: **1 to 2000 bytes**

Alphanumeric strings of various lengths are stored using this datatype. Single quotations are used to indicate its value. It occupies the entire amount of memory that has been specified, even if the space is not being used by the data.

Example:

address is a variable that may hold up to 10 bytes of data and has the type VARCHAR (10). It can contain alphanumeric values. Waste occurs in unused areas.

4.VARCHAR2(size)

Range: **1 to 4000 bytes**

Variable-length alphanumeric strings are stored using this datatype. Its value is in single quotation marks. It frees up memory space that isn't being used, hence conserving space.

Example

name is a variable that can store up to 10 bytes of data in memory, or an alphanumeric value, using the format VARCHAR2(10). Memory that isn't being used is released.

5.Date

Range: **01-Jan-4712 BC to 31-DEC-9999**

The information is kept in date format, DD-MON-YYYY.This datatype's value is enclosed in single quotes.

Example:

DOB DATE, where DOB is a variable that holds the date of birth in a predetermined format (for example, "13-FEB-1991")

6. %TYPE

When we want a variable to inherit the datatype of a table column, but its datatype is unknown, this function stores the value of that variable. Also, because its value is typically taken from an already-existing database table, it adopts the datatype of the column for which it is utilized.

Student sno %TYPE; where Student is the name of the table created in the database and sno is a variable whose datatype is unknown, and %TYPE is used to store its value.

7. BOOLEAN

a. Conditional statements use this datatype.
b. The values are kept logically.
c. Either it is TRUE, or it is FALSE.

Example:

Admin is a variable whose value can be either TRUE or FALSE, depending on the condition being checked. Its type is BOOLEAN.

- **Structure**
  - PL/SQL, known as a block-structured language, encompasses fundamental building elements such as procedures, functions, and anonymous blocks, all organized into logical blocks. These logical blocks can contain nested sub-blocks and are designed to address specific problems or sub-problems within a program.
  - In PL/SQL, a block represents the smallest meaningful collection of code, similar to other procedural languages. It serves as an execution and scoping boundary for variable declarations and exception handling. Named blocks, which can take the form of packages, procedures, functions, triggers, or object types, as well as anonymous blocks, which are code blocks without names, can be written in PL/SQL.

A semicolon marks the end of every PL/SQL query (;). Using BEGIN and END, PL/SQL blocks can be nested inside of other PL/SQL blocks. The fundamental design of a PL/SQL block is as follows:

DECLARE
<declarations section>
BEGIN

<executable command(s)>

EXCEPTION

<exception handling>

END;

Only one of the four components in a PL/SQL block must be mandatory:

1.Header

only utilized for named blocks. The header specifies how to call the named block or programme. It is Optional Part.

2.Section on Declaration

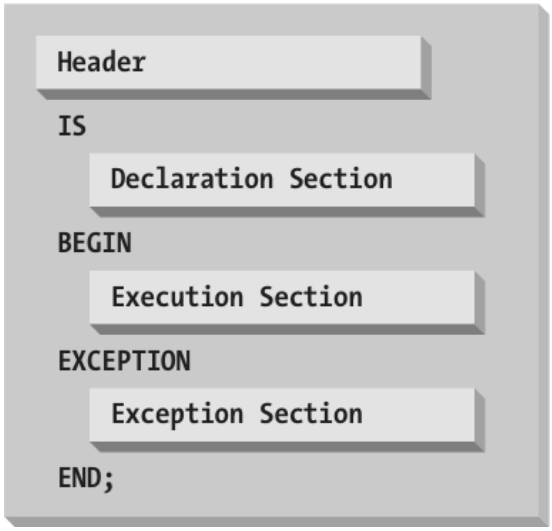
Describes the variables, cursors, and sub-blocks that the execution and exception sections relate to.It is Optional Part.

3.Section of execution

Statements At runtime, the PL/SQL runtime engine will operate.It is Mandatory Part.

4.Section on exceptions

Handles deviations from the standard processing (warnings and error conditions). It is Optional Part.



5.3 Controlling the Program Flow

Sequential control statements, loop statements, and conditional selection statements are the three types of control statements in PL/SQL.

The PL/SQL control statement categories are:

1. Conditional selection statements, which execute several statements depending on the value of the input data. IF and CASE are the conditional selection statements.
2. Statements that use loops to repeat the same instructions with a variety of data values.The fundamental LOOP, FOR, and WHILE loop statements are the loop statements.

   The EXIT statement moves the focus to a loop's conclusion.

   The CONTINUE statement ends the current loop iteration and hands off control to the following one.

   The WHEN clause, which is optional for both EXIT and CONTINUE, allows you to specify a circumstance.
3. PL/SQL programming does not require sequential control statements. GOTO, which jumps to a specific statement, and NULL, which does nothing, are the sequential control statements.

5.4 Conditional Statements

For various data values, the conditional selection statements CASE and IF execute various statements. The CASE statement executes the corresponding statement based on a selection from a list of circumstances.

These are the forms for the CASE statement:

1. IF THEN
2. IF THEN ELSE
3. IF THEN ELSIF

The CASE statement executes the appropriate statement by picking one from a list of circumstances.These are the forms for the CASE statement:

   a. Simple, which compares a single expression to a range of possible values.
   b. Several conditions are assessed by searched, which selects the first one that is true.

When a separate course of action is to be taken for each choice, the CASE statement is appropriate.

1. IF THEN Statement

Depending on a condition, the IF THEN statement executes or skips a series of one or more statements.

Here is the structure of the IF THEN statement:

IF condition THEN

statements

END IF;

The statements execute if the condition is true; otherwise, the IF statement does nothing.

2. IF THEN ELSE Statement

The form of the IF THEN ELSE statement is as follows: If the conditional value is true, the statements execute; otherwise, the else statements execute.

Here is the structure of the IF THEN ELSE statement:

IF condition THEN

statements

ELSE

else statements

END IF;

3. IF THEN ELSIF Statement

This is the structure of the IF THEN ELSIF statement:

IF condition_1 THEN

statements_1

ELSIF condition_2 THEN

statements_2

[ELSIFcondition_3 THEN

statements_3

]...

[ELSE

else statements

]

END IF;

The initial sentences whose condition is true are executed by the IF THEN ELSIF clause. Conditions that still exist are not assessed. If none of the conditions are met, the IF THEN ELSIF statement does nothing; otherwise, the else statements, if any, run.

4. Simple CASE Statement

The basic CASE statement is composed as follows:

CASE selector

WHEN selector_value_1 THEN statements_1

WHEN selector_value_2 THEN statements_2

...

WHEN selector_value_n THEN statements_n

[ELSE

else statements]

END CASE;]

The expression that the selector is (typically a single variable). Every selector value has two possible values: literal and expression. The straightforward CASE statement executes the first set of statements when selector value == selector.

Conditions that still exist are not assessed.

If no selector value equals selector, the CASE statement raises the predefined exception CASE NOT FOUND and executes else statements if they are present.

5. Searched CASE Statement

The structure of the CASE query is as follows:

CASE

WHEN condition_1 THEN statements_1

WHEN condition_2 THEN statements_2

...

WHEN condition_n THEN statements_n

[ELSE

else_statements]

END CASE;]

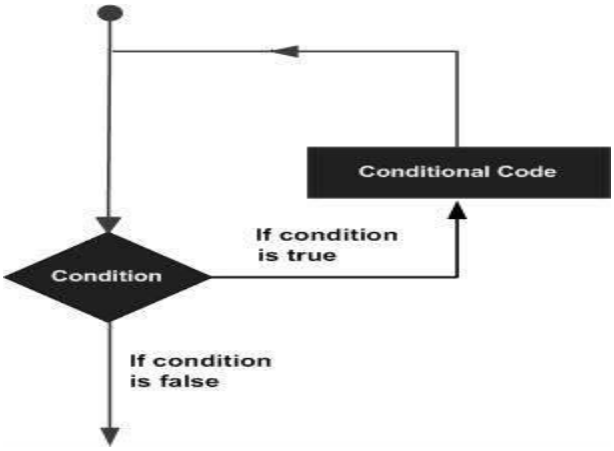
The initial statements that the searching CASE statement executes when the condition is true. The remaining circumstances are not assessed. The CASE statement raises the predefined exception CASE NOT FOUND if none of the conditions are true and executes else statements if they are present.

5.5 Loops

In many programming scenarios, there arises a need to execute a block of code repeatedly. Typically, statements within a function execute sequentially, with each statement following the one before it. However, various control structures in programming languages provide options for more complex execution paths.

One common construct for repetitive execution is the loop statement. This statement enables the execution of one or more statements multiple times. The general form of a loop statement across most programming languages allows for the iteration of a statement or a collection of statements.

Loop statements offer flexibility in controlling the flow of execution, allowing developers to automate repetitive tasks efficiently. They are essential tools in programming for handling scenarios where actions need to be performed iteratively until certain conditions are met.



To meet the looping needs, PL/SQL offers the following types of loops.

1. **PL/SQL Basic LOOP**

   Between the LOOP and END LOOP statements is a sequence of statements that makes up a basic loop structure. The series of statements are carried out on each iteration before the control returns to the loop's beginning.

Syntax:

```
LOOP
Sequence of statements;
END LOOP;
```

A single statement or a block of statements may be included in this sequence of statements.

The loop must be broken using either an EXIT or an EXIT WHEN statement.

2. PL/SQL WHILE LOOP

As long as a specified condition is true, a target statement is continuously executed using a WHILE LOOP statement in the PL/SQL computer language.

Syntax

```
  WHILE condition LOOP
sequence_of_statements
  END LOOP;
```

3. PL/SQL FOR LOOP

The repetition control structure known as a FOR LOOP makes it simple to create a loop that must run a certain number of times.

Syntax

```
FOR counter IN initial_value ..final_value LOOP
sequence_of_statements;
END LOOP;
```

The direction of control in a for Loop is as follows:

a. The first and only execution of the initial step occurs initially. Any loop control variables can be declared and initialized at this phase.
b. The condition, consisting of the initial value and final value, is then assessed. The body of the loop is executed if it returns TRUE. If it returns FALSE, the loop's body is skipped, and the statement immediately following the for loop takes control.
c. The counter variable's value is raised or lowered following the execution of the for loop's main body. The situation is now reevaluated.
d. If it is TRUE, the loop is carried out, and the procedure is repeated (body of loop, then increment step, and then again condition).
e. The FOR-LOOP ends when the condition is determined to be FALSE.

4. Nested loops in PL/SQL

One loop may be used inside another in PL/SQL Every other basic loop, while, or for loop may include one or more loops.

Syntax

LOOP

Sequence of statements1

LOOP

Sequence of statements2

END LOOP;

END LOOP;

The Loop Control Statements

Statements used to control loops divert execution from the usual path. All automatic objects produced within a scope are deleted when execution exits it.

The following control statements are supported by PL/SQL.Putting the control outside of a loop is facilitated by labelling the loops.

1. EXIT statement

Following the END LOOP, control moves to the statement after the Exit statement has finished the loop.

2. CONTINUE the statement

It causes the loop to skip the rest of its body and retest its state right away before repeating.

3. GOTO statement

It puts the labelled statement in charge. While using the GOTO statement in your programme is not recommended.

- **Knowledge Check 1**

  Fill in the Blanks

  1. As a procedural extension language for SQL and the Oracle relational database, the PL/SQL programming language was created by _____in the late 1980s.
  2. Sequential control statements, loop statements, and conditional selection statements are the three types of _____in PL/SQL.
  3. Only one of the _____components in a PL/SQL block must be mandatory.

- **Outcome-Based Activity 1**

  List out all the Conditional and Loop statements in PLSQL.

5.6 Stored Procedures

A set of SQL and other PL/SQL computer language statements are logically grouped together into a procedure or function as a schema object to carry out a specified purpose. A procedure is a subroutine, similar to a subprogram in a conventional programming language, saved in a database and is frequently referred to as a "stored procedure."

A procedure has a name, a list of parameters, and a SQL command (s)A user's schema is used to build procedures and functions, which are then saved in a database for later usage.

PL/SQL blocks are PL/SQL subprograms that can be called with a variety of arguments.PL/SQL offers two different types of subprograms.

a. **Functions:** These programmers' subroutines typically compute and return a single value.
b. **Procedures:** Mostly employed to carry out actions, these subprograms don't immediately return a value.

Creating A Procedure

The CREATE OR REPLACE PROCEDURE statement generates a procedure. The following is the CREATE OR REPLACE PROCEDURE statement's shortened syntax:

CREATE [OR REPLACE] PROCEDURE procedurename
[(parameter name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
<procedure body >
END procedure name;

5.7 Stored Functions

Subprograms that can be written and saved in the database as database objects include procedures and functions. They can also be referenced or invoked inside of other blocks.The PL/SQL Function and PL/SQL Procedure are extremely similar. A function must always return a value, whereas a procedure may or may not do so. This is the primary distinction between a procedure and a function. Other than this, PL/SQL function has all the same properties as PL/SQL procedure.

A return statement is a requirement for the function.

a. The data type you intend to return from the function is specified by the RETURN clause.
b. The executable portion is contained in the function body.
c. For defining a solo function, the AS keyword is used rather than the IS keyword.

Syntax:

1. CREATE [OR REPLACE] FUNCTION function name [parameters]
2. [(parameter name [IN | OUT | IN OUT] type [, ...])]
3. RETURN returndatatype
4. {IS | AS}
5. BEGIN
6. < function body >
7. END [function name];

Example

```
create or replace function adder (n1 in number, n2 in number)
return number
is
n3 number (8);
begin
n3 :=n1+n2;
return n3;
end;
```

5.8 Handling Errors and Exceptions

In PL/SQL, an error that happens while the programme is running is referred to as an exception. Programmers can capture these occurrences using exception blocks in PL/SQL, and the error condition is dealt with appropriately. A PL/SQL error that is raised during programme execution, either expressly by your programme or unintentionally by TimesTen, is an exception. An exception can be handled by propagating it to the caller environment or catching it using a handler.

Two categories of exceptions exist:

1. Exceptions defined by the system

2. Exceptions created by the user

Syntax for exception handling:

DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling goes here >

WHEN exception1 THEN

exception1-handling-statements

WHEN exception2 THEN

exception2-handling-statements

WHEN exception3 THEN

exception3-handling-statements

.....

WHEN others THEN

exception3-handling-statements

END;

5.9 Cursors

Oracle creates a memory space called a context area when a SQL statement is processed there. The pointer to this context area is the cursor. It includes all the data required to process the statement. The context area in PL/SQL is managed by Cursor. The rows of data accessed by a select statement are listed in a cursor.

A programme is referred to as a cursor when it is used to retrieve and handle each row returned by a SQL statement individually. Cursors come in two varieties:

- **Implicit Cursors**

  If you don't use an explicit cursor for the statement, Oracle will create implicit cursors while the SQL statement is performed. They are automatically created when DML commands, like INSERT, UPDATE, DELETE, etc., are executed to process the statements.

To examine the state of DML activities, Orcale offers several attributes referred to as Implicit cursor's attributes.

%FOUND, %NOTFOUND, %ROWCOUNT, and %ISOPEN are a few of them.

- **Explicit Cursors**

  To have additional control over the context area, programmers specify explicit cursors.

  These cursors need to be defined in the PL/SQL block's declaration section.It is built on a SELECT operation that produces several rows.

  Syntax

  CURSOR cursor name IS select statement;;

  Steps

These actions must be carried out when using an explicit cursor.

1. Declare the memory cursor's initialization.
2. For memory allocation, move the cursor.
3. To retrieve data, recover the cursor.
4. To release the allotted memory, close the cursor.

5.10 Triggers

When a specific event takes place, the Oracle engine automatically calls the trigger. When a certain condition is met, the trigger is frequently called from a database. When a certain event occurs, triggers are stored programmes that are automatically executed or fired.

The following occurrences can cause triggers to be written to be executed.

1. The statement for database manipulation (DML) (DELETE, INSERT, or UPDATE).
2. The statement is defining a database (DDL) (CREATE, ALTER, or DROP).
3. An operation on a database (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

The table, view, schema, or database that the event is related with could all have triggers defined on them.

Advantages of using Triggers

1. Trigger automatically creates some derived column values.
2. The enforcement of referential integrity
3. Recording events and keeping track of table access information Auditing

4. Synchronized table replication
5. The imposition of security authorizations
6. Avoiding erroneous transactions

- **Knowledge Check 2**

  State true and False for the following Sentences.

  1. A procedure is a subroutine, similar to a subprogram in a conventional programming language, saved in a database and is frequently referred to as a Program.
  2. When a specific event takes place, the Oracle engine automatically calls the trigger.
  3. In PL/SQL, an error that happens while the programme is running is referred to as an event.

- **Outcome-Based Activity 2**

  Try to Enable, Disable and Drop Triggers in the SQL Server.

5.11 Summary

- A procedural language called PL/SQL was created with SQL statements already built into its syntax. The Oracle Database server builds PL/SQL programme units and stores them in the database.
- The use of conditional statements allows you to "test" various assumptions and then execute the statements based on the results. In PL/SQL, there are two varieties of conditional control statements:
  - ✓ IF clauses
  - ✓ CASE declarations
- Many predefined datatypes are offered by PL/SQL.Integer, floating point, character, Boolean, date, collection, reference, and big object (LOB) types are only a few examples. You can also create your own subtypes using PL/SQL.
- A series of statements within a PL/SQL code block is repeated using the LOOP statement.
- Each entry into the loop body is immediately followed by an evaluation of the condition.
- The four types of loop statements offered by PL/SQL are basic loop, WHILE loop, FOR loop, and cursor FOR loop.

- An application programme uses a cursor, also known as a named control structure, to identify and pick a row of data from a result set. You can use a cursor to read and execute the query result set one row at a time rather than running the query in its whole.
- Indirect Cursors and Direct cursors are two types of Cursors.
- An insert, update, or delete operation on a table will cause a PL/SQL trigger, which is a named database object, to specify and carry out a certain set of actions. The CREATE TRIGGER statement in PL/SQL is used to construct triggers.

5.12 Self-Assessment Questions

1. What is PLSQL?
2. What are PLSQL data types?
3. What are Control Flow Statements in PLSQL?
4. What is Conditional Statements in PLSQL?
5. Explain Loops in PLSQL.
6. What is Stored Procedures and Stored Functions?
7. What is Cursors?
8. What is Triggers?

5.13 References

- Atkinson, M.P. (1981) *Database*. Maidenhead: Pergamum InfoTech.
- Frank, L. and Helmersen, O. (1988) *Database theory and practice*. Wokingham, England: Addison-Wesley Publishing Company.
- *Database users: 2nd Toronto conference: Selected papers* (1990). Canadian Association for Information Science.
- *Database* (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) *Database*. Rockville, MD: Computer Science Press.

Unit 6

Concurrency Control

Learning Outcomes:

- Students will be capable of learning about the Concurrency Control techniques.
- Students will be able to understand the Locking and management of locks.
- Students will be able to understand the concepts of 2PL.
- Students will be able to understand about the Locking Techniques deadlocks.
- Students will be able to learn about Optimistic Concurrency Control.
- Students willlearn about the Comparison of Concurrency control methods.

Structure

6.1 Concurrency Control Techniques

Concurrency control in a database management system (DBMS) is a crucial function that ensures the smooth execution of concurrent processes, particularly in multi-user environments. By managing simultaneous processes effectively, concurrency control minimizes conflicts and maintains data consistency.

Advantages of Concurrency Control:

1. Reduced Waiting: With concurrency control, there is less waiting time for processes, enhancing overall system efficiency.
2. Slowed Reaction Time: Processes can respond promptly without being hindered by conflicts, leading to improved system responsiveness.
3. Increased Resource Utilization: Efficient management of concurrent processes allows for optimal resource utilization within the system.
4. Enhanced System Performance: By preventing conflicts and ensuring smooth execution, concurrency control contributes to the overall performance and efficiency of the system.

Techniques for Concurrency Control:

1. Locking: Locking mechanisms ensure that transactions have exclusive access to specific data objects, preventing conflicts. This includes shared locks for read-only access and exclusive locks for both read and write operations.
2. Time Stamping: Time stamps are used to track the beginning and completion times of transactions, enabling the system to prioritize and manage concurrent processes effectively.
3. Optimistic Concurrency Control: This approach assumes that conflicts are rare, allowing transactions to proceed without unnecessary delays. It verifies data consistency after transactions are completed.
4. Multiversion Concurrency Control: Multiversion schemes maintain previous versions of data items to facilitate concurrent access. Each write operation generates a new version of the data item, identified by timestamps.
5. Validation Concurrency Control: Transactions are executed without restrictions until they are committed, based on the assumption that conflicts are infrequent. Validation occurs after transactions are completed to ensure data consistency.

Locking and Management:

Lock-based protocols ensure that transactions obtain the necessary locks before accessing data, preventing conflicts. Shared locks allow read-only access, while exclusive locks provide exclusive write access. These locks help eliminate concurrency issues by isolating transactions and controlling access to data.

Shared Locks: Allow read-only access to data items, ensuring read integrity and preventing updates while reading. Exclusive Locks: Provide exclusive write access to data items, preventing other transactions from accessing or updating the data simultaneously.

By effectively managing locks, DBMS ensures data consistency and integrity while allowing for concurrent access by multiple users.

The data item can be read and written by the Transaction while in the exclusive lock. This lock is exclusive, and different transactions cannot edit the same data at the same time while it is locked.An exclusive or write lock grants a process exclusive access to the defined portion of the file for writing.No other process can lock that section of the file while a write lock is in place.Exclusive locks can be activated or deactivated.

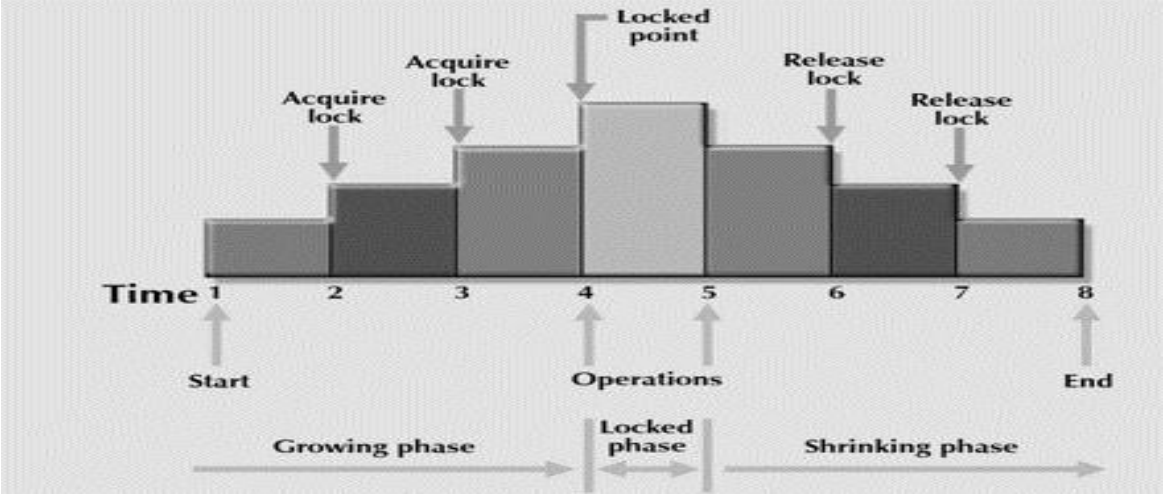
Exclusive locks safeguard both recoverable and non-recoverable file resource modifications.Just one Transaction can own them at a time.If another task presently has an exclusive lock or a shared lock against the requested resource, any transaction requiring an exclusive lock must wait.

6.3 2PL Locking Protocol

Each Transaction involves locking and unlocking the data item twice.

Growth Phase – At this phase, all locks are distributed. After all modifications to data items are committed, locks are not released, and the second phase (the shrinking phase) then begins.

Phase of shrinkage: No locks are granted during this time, and any changes to data items are documented (stored) before locks are released.



A transaction in the growth phase reaches the point when it has gained all the locks it might require. This location is known as LOCK POINT. The Transaction goes through a shrinking phase after reaching the lock point.

Types

There are two types of two-phase locking.

1. **The strict two-phase locking protocol**

   After the lock point, a transaction can release a shared lock but not an exclusive lock until the Transaction commits. This technique produces a schedule with fewer cascade

   Cascading schedule: A transaction in this schedule is dependent on another transaction in a cascading schedule. Hence, if one must roll back, the other must also.

2. **The rigorous two-phase locking protocol**

   Every lock, whether shared or exclusive, cannot be released by a transaction until it commits. Serializability is guaranteed by the 2PL protocol, but deadlock cannot be prevented.

6.4 Locking Techniques

Four different lock protocols are available:

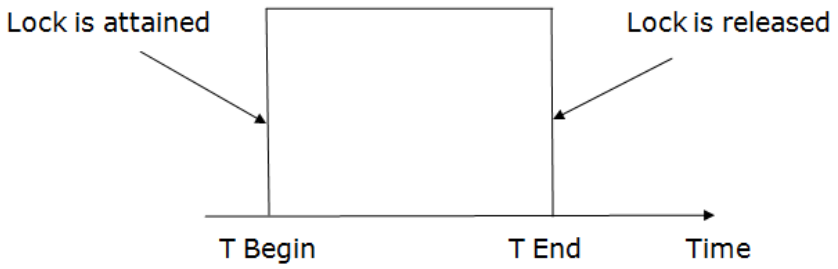
1. **Simplistic lock protocol**

The easiest way to lock the data while a transaction is to do it this way. All transactions can obtain a lock on the data before inserting, deleting, or updating it thanks to simple lock-based protocols. After the Transaction is finished, it will unlock the data item.

2. Pre-claiming Lock Protocol

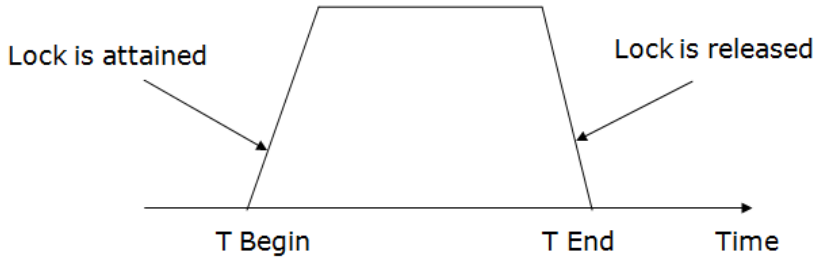
Pre-claiming Lock

In order to identify all the data objects on which they require locks, protocols assess the Transaction. It asks the DBMS for a lock on each of those data objects before starting the Transaction's execution. TheTransaction can start under this protocol if all the locks are granted. All locks are released once the Transaction is finished. This protocol enables the Transaction to roll back if all the locks are not provided and waits until they are all granted.



3. Two-phase locking (2PL)

The execution phase of a transaction is managed by the two-phase locking protocol, which divides it into three sections. In the first phase, the transaction begins execution and requests the locks it requires. The second phase involves acquiring all necessary locks by the transaction. Once the transaction releases its first lock, the third phase begins, during which the transaction is not allowed to request additional locks. Only the locks that have been acquired are released during this phase.

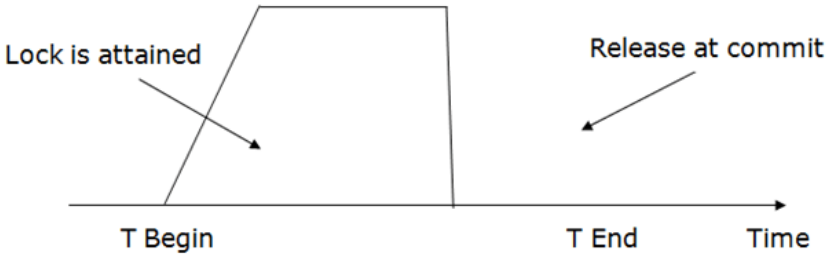


The 2PL process comprises two stages:

- Growing phase: A new lock on the data item may be acquired by the Transaction during the growing phase, but none may be released.
- The phase of shrinkage: During the phase of shrinkage, any locks already held by the Transaction may be released, but no additional locks may be purchased.

4. Strict Two-phase locking (Strict-2PL)

Strict-2PL's initial phase resembles 2PL in certain ways. After obtaining all the locks in the first step, the Transaction continues to run smoothly. Only the fact that Strict-2PL does not release a lock after utilizing it distinguishes 2PL from 2PL strictly. Strict-2PL releases each lock individually after waiting for the entire Transaction to commit. There is no shrinking step of lock release in the strict-2PL process.As opposed to 2PL, it lacks cascading abort



- **Knowledge Check 1**

Fill in the Blanks for the Following Sentences.

1. It is a DBzMS technique that allows us to manage _____concurrent processes so that they run without conflicting with one other, which is common in multi-user systems.

2. By locking or isolating a specific transaction to a_____, lock-based protocols serve to eliminate the concurrency problem in DBMS for simultaneous transactions.

3. During the _____ any locks already held by the Transaction may be released, but no additional locks may be purchased.

- **Outcome-Based Activity 1**
- **List out all the types of Locking Techniques in DBMS.**

- **6.5 Deadlocks**
  - A deadlock in a database occurs when multiple transactions are waiting for each other to release locks, resulting in a situation where none of the transactions can proceed. For example, Transaction A may hold a lock on certain rows in the Accounts table while needing to update certain rows in the Orders table. At the same time, Transaction B holds locks on rows in the Orders table that Transaction A needs to update.
  - Deadlocks are one of the most concerning issues in database management systems (DBMS) because they prevent tasks from completing and keep them in a waiting state indefinitely. When a deadlock occurs, transactions are stuck waiting for each other to release locks, causing all activities to come to a standstill. The DBMS must recognize the deadlock and terminate one of the transactions to resolve the deadlock.
  - Deadlock avoidance is preferred over termination or restart when a database is deadlocked because it conserves time and resources. Techniques such as the "wait-for graph" are used to anticipate deadlock situations. However, deadlock prevention techniques are more suitable for larger databases. In deadlock detection, the DBMS identifies whether a transaction is stuck in a deadlock by maintaining a wait-for graph.
  - In deadlock prevention approaches, the database management system allocates resources in a way that prevents deadlocks from occurring. The system examines transactional

operations to determine if a deadlock situation is likely to arise and prevents transactions that may cause deadlocks from being executed.

- One such deadlock prevention scheme is the Wait-Die scheme, where the DBMS compares the timestamps of two transactions if one transaction requests a resource currently held by another transaction with a conflicting lock. The older transaction is allowed to postpone execution until the resource becomes available.

2. Wound wait scheme

In a wound wait strategy, the younger Transaction is compelled to kill the Transaction and release the resource if the elder Transaction asks a resource that it is holding. The younger Transaction is reopened with the same timestamp after the little pause. If the younger Transaction requests a resource that the elder Transaction has retained, the younger Transaction is instructed to wait until the older Transaction releases the requested resource.

6.6 Optimistic Concurrency Control

OCC, commonly referred to as optimistic locking, is a concurrency control technique used in transactional systems such as software transactional memory and relational database management systems.OCC makes the assumption that several transactions can commonly finish one after the other without interfering.

Phases - The three steps of optimistic concurrency control are described here.

1.Read phase

Reading and storing various data elements in temporary variables (local copies). These variables undergo all actions without causing the database to be updated.

2. Validation Phase

To verify that serializability will not be validated if the transaction updates are actually made to the database, all concurrent data items are examined. TheTransaction rolls back if the value is altered. The write-sets and read-sets are kept up to date, and the transaction timestamps are used.

The following conditions must exist in order to verify that Transaction A does not obstruct transaction B.

- ✓ Before TransA begins the read phase, TransB has finished its write phase.
- ✓ TransB's write phase is finished before TransA's write phase begins, and there are no shared items between the read and write sets of the two transactions.
- ✓ TransB finishes reading before TransA finishes reading, and the read set and write a set of TransA have no items in common with TransB's write set.

3.Write phase

Upon successful validation, the Transaction updates the database. In such a case, transactions are stopped and restarted while updates are discarded. As it doesn't employ any locks, it is deadlock-free; however, data item starvation issues could happen.

6.7 Comparison of Concurrency control methods

The coordination of a transaction's simultaneous execution in a multi-user database system is known as concurrency control. The primary goal of concurrency control is to assure the serializability of the Transaction in a multi-user database system. The necessary data pieces can only be accessed in a single, mutually exclusive way. That is, a transaction can modify a data item while it is being accessed by another transaction. The most popular way to achieve this criterion is to restrict access to a data item to transactions that are currently in possession of a lock on it.

One-phase or two-phase locking methods can be used with locking-based concurrency control systems.

- ✓ One-phase Locking Protocol
- ✓ Two-phase Locking Protocol
- ✓ Distributed Two-phase Locking Algorithm
- ✓ Distributed Timestamp Concurrency Control
- ✓ Conflict Graphs
- ✓ Distributed Optimistic Concurrency Control Algorithm.

There are Various methods of concurrency control

Locking data items in a database can be done in several ways:

1. Shared Mode (S): Transactions with shared mode locks on a data item Q can only read Q but cannot write to it.
2. Exclusive Mode (X): Transactions with exclusive mode locks on a data item Q can both read and write to Q. When a transaction requests a lock, it sends a request to the concurrency control manager, and the transaction can proceed only if the manager grants the lock.

In a lock-based protocol, the fundamental principle is that a lock must be acquired before accessing a data item, and the item should be released immediately after use to prevent conflicts. Two types of locks are used: exclusive locks and shared locks.

In the two-phase locking protocol, transactions go through two phases: growing and shrinking. During the growing phase, a transaction can acquire locks but cannot release them, while during the shrinking phase, a transaction can release locks but cannot acquire new ones.

Rigorous two-phase locking eliminates the shrinking phase, ensuring that transactions cannot acquire locks and release any locks simultaneously. However, this protocol may experience deadlock.

Strict two-phase locking allows the freeing of shared locks during the shrinking phase, providing an advancement over rigorous two-phase locking.

Conservative two-phase locking removes the growth phase, requiring transactions to acquire all locks before performing any reads or writes. While it ensures deadlock independence and conflict serializability, it may suffer from recoverability and cascades.

In the timestamping protocol, each transaction is assigned a timestamp upon acceptance into the system. Data items are also timestamped, and transactions can read and write data items based on their timestamps.

These locking protocols and timestamping strategies play crucial roles in ensuring data integrity, concurrency control, and transaction management within a database.

- **Knowledge Check 2**

  State True and false for the following Sentences.

  1. A deadlock in a database occurs when one transactions are waiting for each other to release locks.

2. If a transaction has obtained a shared lock on a data item Q, it can only execute read operations on the item on its own, but if it has obtained an exclusive lock, it can perform both read and write operations.
3. In a wait-die strategy, the younger Transaction is compelled to kill the Transaction and release the resource if the elder Transaction asks a resource that it is holding.

- **Outcome-Based Activity 2**

  Describe all the types of Deadlock Prevention Techniques in DBMS.

6.8 Summary

- In a shared database, the concurrent control technique is a way to control how many transactions are executed at once. The read-write operation of transactions causes conflicts, which are resolved by enforcing the isolation of various transactions and maintaining database consistency.
- A data variable that is linked to just one data item is referred to as a lock. This lock denotes that data item operations are permitted. Locks fall into one of two categories: Shared and individual locks.
- Database management systems use the two-phase locking (2PL) protocol, commonly referred to as basic 2PL, to lock data from running concurrent transactions. For instance, one user could need to read data from a record while another tries to modify or update that data at the same time.
- A deadlock occurs when many transactions are awaiting the release of their locks in a database. For instance, in order to complete, Transaction A can have a lock on some entries in the Orders table and need to change some rows in the Accounts table.
- A concurrency control technique used in transactional systems like relational database management systems and software transactional memory is optimistic concurrency control (OCC), commonly referred to as optimistic locking.
- Several transactions can typically complete without interfering with one another, according to OCC.

6.9 Self-Assessment Questions

1. What is concurrency control Techniques?
2. What is Locking in DBMS?
3. Explain 2PL.
4. What are the different types of Locking Techniques?
5. What is deadlocks?
6. What is Optimistic Concurrency Control Techniques?
7. Explain the Management of Locks.
8. What are the different types of Locks?

6.10 References

- Atkinson, M.P. (1981) *Database*. Maidenhead: Pergamum InfoTech.
- Frank, L. and Helmersen, O. (1988) *Database theory and practice*. Wokingham, England: Addison-Wesley Publishing Company.
- *Database users: 2nd Toronto conference: Selected papers* (1990). Canadian Association for Information Science.
- *Database* (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) *Database*. Rockville, MD: Computer Science Press.

Unit 7

Other Databases

Learning Outcomes:

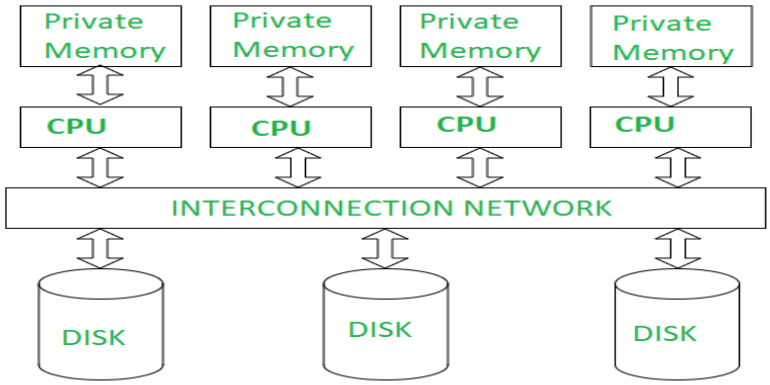
- Students will be capable of learning about the Parallel and Distributed Databases.
- Students will be able to understand about the Object-Based Databases.
- Students will be able to understand the concepts of XML Databases.
- Students will be able to understand about the NoSQL Database.
- Students will be able to learn about the Big Data Databases.

Structure

7.1 Introduction to Parallel and Distributed Databases

1.Parallel Database

A parallel DBMS is one that utilizes several processors and is intended to carry out operations concurrently whenever practical. A parallel DBMS connects a number of smaller machines to produce the same throughput as one large system would. Data loading and query processing are carried out simultaneously. Database systems that are centralized and client-server-based are not strong enough to accommodate applications that require quick processing. Applications for online transaction processing and decision assistance benefit greatly from parallel database systems. A large task is divided into smaller ones in parallel processing, and each smaller work is carried out simultaneously on various nodes. As a result, a bigger task can be finished more rapidly.



Features of Parallel Database

- There is CPU parallel processing.
- It breaks down large jobs into numerous smaller activities, which enhances performance.
- It Finishes tasks very rapidly.

Architectural Models

For parallel machines, there are numerous architectural paradigms.

The following are the most crucial ones:

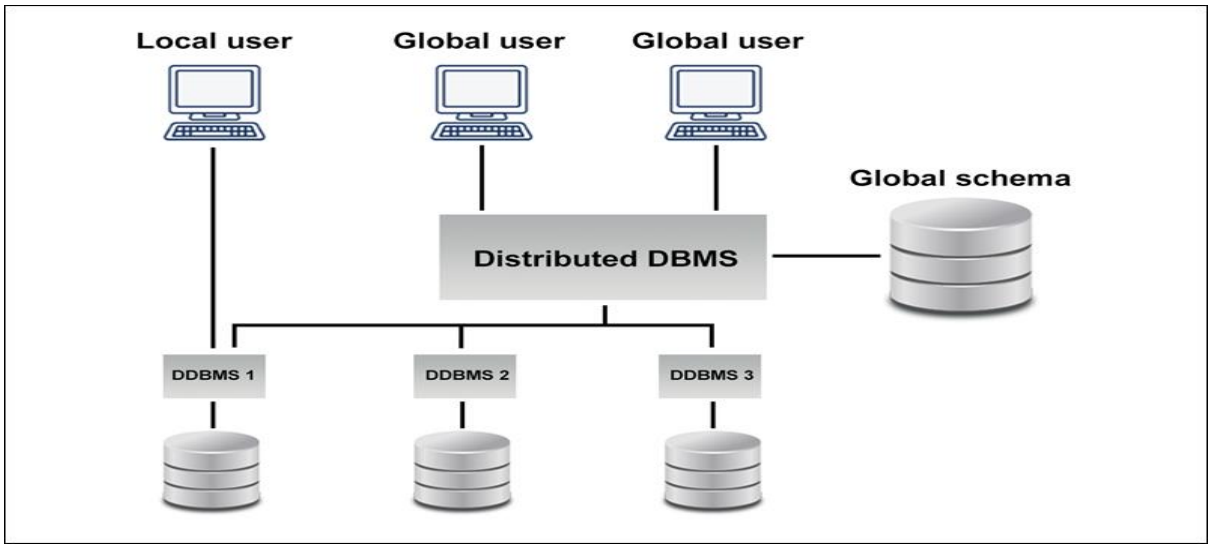
- **Shared-memory multiple CPU**-In this case, the computer has numerous CPUs that are all operating at once and connected to a network for communication. These CPUs share a single main memory and a single disc storage array.

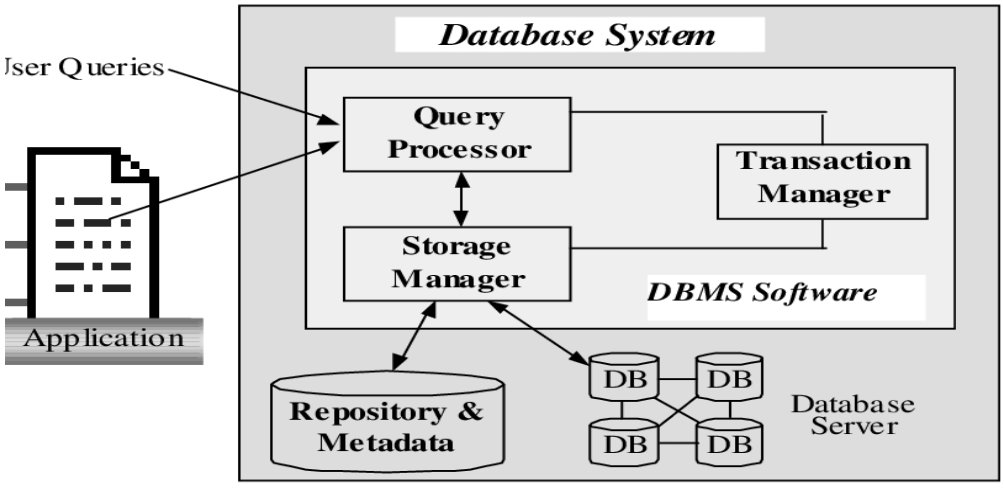
- **Shared disk architecture**-In this design, all nodes share mass storage, but each has their own primary memory. Each node actually has a number of processors.
- **Shared nothing architecture**-Each node in a shared-nothing architecture has its own main memory and mass storage.

Distributed Databases

A shared, logically connected collection of data that is physically dispersed across a computer network at various locations is known as a distributed database.

The software that makes distributed data accessible to users and enables the management of distributed databases is known as a distributed database management system (DBMS).





Features of Distributed Databases

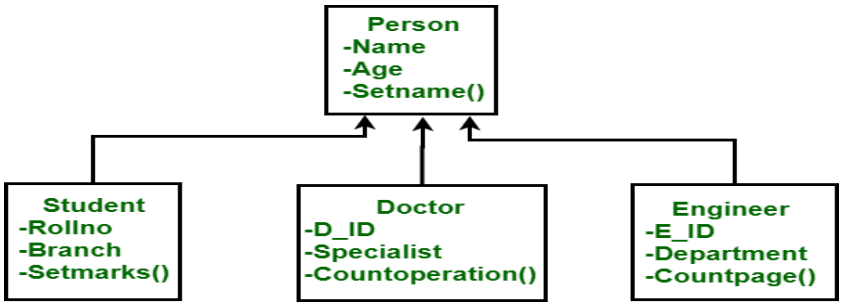
- A collection of information that is communicated logically.
- The information is divided into numerous pieces.
- There could be fragment replication.
- There is a communication network connecting the locations.
- The phrase "modular development" refers to the idea that adding nodes to the existing network will allow us to expand the functionality of the system we are now using to additional locations without interfering with it.
- Improves Reliability The phrase "increases reliability" means that if one node on a network fails, the system may continue running by dividing the work among the other nodes in the network.
- Increases Performance– A huge database is divided into smaller databases that are scattered across many places and are easier to manage with greater performance. We all know that a little database is easier to handle than a large database.
- Increased Availability-As data may be accessed from numerous different network nodes, the failure of one node won't have an impact on the availability of the data.
- Faster Reaction – When data is accessible locally, data retrieval becomes more effective.

The Primary difference between the parallel and distributed databases is that the former is tightly coupled and then later loosely coupled.

7.2 Introduction to Object-Based Databases

- An object-oriented database (OOD) is a database system capable of handling complex data objects resembling those used in object-oriented programming languages (OOP). Objects are fundamental elements in OOP, and OODs emerged as a response to the constraints imposed by the relational model when used in diverse applications like computer-aided design and geographic information systems. Object-based databases were introduced to address these limitations, enabling the management of complex data types.
- In object-based databases, each entity is treated as an object and stored in tables. The concept of inverse reference is utilized to maintain relationships between objects and to organize similar items into classes and subclasses.

- Object-based data models serve as the foundation for a database's structure, providing conceptual tools for expressing data, relationships, semantics, and constraints. The object-oriented data model extends the entity-relationship (ER) model by incorporating concepts such as encapsulation, methods, and object identity. Core principles of object-oriented programming, such as inheritance, object identity, encapsulation, and interface definition, are applied in data modeling.
- The object-oriented data model supports a more comprehensive type system, including structured and collection types. It combines features of both object-oriented and relational data models, resulting in an object-relational data model supported by most database providers. This model enhances the traditional relational paradigm with features like structured and collection types, object orientation, and more.



Advantages of Object-Based Data Models

- Data that is more complex and diverse than MySQL data types.
- Data may be saved and promptly recovered with ease.
- Seamless interface with programming languages that use objects.
- Advanced real-world problems are simple to model.
- Custom data types provide for extensibility.

Disadvantages of Object-Based Data Models

- It is not used as frequently as relational databases.
- There's no global data model. There are no theoretical norms or grounds.
- It contradicts the ideas.
- Performance issues result from high complexity.
- There are no adequate access controls or security measures in place for the objects.

7.3 XML Databases

A data persistence software system called an XML database is used to store an enormous volume of data in XML format. It offers a safe location to keep XML documents. With XQuery, you can export and serialize your saved data into the desired format. Document-oriented databases and XML databases are frequently used together. Huge amounts of data are stored in XML format using XML databases. Having a secure location to store the XML documents is necessary because XML usage is expanding across all industries. The database's data can be queried with XQuery, serialized, and exported into any format needed.

Types

XML databases come in two different varieties.

- **XML-enabled database**

  The extension offered for the conversion of an XML document is what an XML-enabled database is. In a relational database like this one, data is kept in tables made up of rows and columns. The tables include a collection of records, which themselves are composed of fields. The operation of an XML-enabled database is identical to that of a relational database. It resembles an extension offered for the conversion of XML documents. In this database, data is kept in tables with rows and columns.

- **Native XML database (NXD)**

  Large amounts of data are stored in native XML databases. Native XML databases are based on container format rather than table structure. Data can be accessed using XPath expressions. Because it is much more capable of storing, maintaining, and querying XML content, native XML databases are favoured over XML-enabled databases. Path expressions are used to query native XML databases. The advantage of a native XML database over an XML-enabled database. It is much more capable than an XML-enabled database to store, query, and maintain an XML document.

Example

```
<? xml version="1.0"?>
<contact-info>
<contact1>
<name>Arun Sharma </name>
```

<company>SSMT.org</company>
<phone>9588306809</phone>
</contact1>
<contact2>
<name>Manish Gupta </name>
<company>SSMT.org</company>
<phone>09990449645</phone>
</contact2>
</contact-info>

- **Knowledge Check 1**

  Fill in the Blanks for the following Sentences.

  1. A data persistence software system called an _____is used to store the enormous volume of data in XML format.
  2. A database system that can handle complex data objects, or objects that resemble those used in object-oriented programming languages, is known as an _____.
  3. A shared, logically connected collection of data that is physically dispersed across a computer network at various locations is known as_____.

- **Outcome-Based Activity 1**

  Illustrate architecture of Parallel and distributed databases with Proper diagram.

7.4 NoSQL Database

- NoSQL databases diverge from traditional relational databases by storing data in documents rather than relational tables. They are aptly classified as "not only SQL" due to their flexible data models, which encompass various types such as document databases, key-value stores, wide-column databases, and graph databases.
- NoSQL databases are highly proficient in managing extensive amounts of unstructured and semi-structured data, providing a flexible approach to data management unlike the fixed schemas found in relational databases. They excel in adapting to evolving data structures and have the capability to scale horizontally to meet the demands of increasing data volumes.

- Initially coined as "non-SQL" or "non-relational" databases, the term "NoSQL" has evolved to encompass a wide range of database architectures and data types. DynamoDB, MongoDB, and SimpleDB are prominent examples of NoSQL databases, each catering to different use cases and data management needs.
- In contrast to relational databases, NoSQL databases eschew the tabular relations paradigm and do not rely on tables to store data. Instead, they offer an alternative method for data storage and retrieval, making them well-suited for real-time web applications and managing vast amounts of data.

History

Flat File Systems are in use at the start of the 1970s.The main issues with flat files are that each organization uses its own flat files and that there are no standards. Data was saved in flat files. Because there is no standardized method for storing data, it is exceedingly challenging to save data in files and retrieve data from files.

After that, E.F. Codd developed the relational database, which provided a solution to the issue of the lack of a common method for storing data. However, relational databases later developed the issue of being unable to handle large amounts of data. As a result, a demand for a database that could handle all problems arose, and the NoSQL database was created.

E.F.Codd's relational database was created in order to overcome the lack of a standardized mechanism for data storing. However, relational databases later developed the issue of being unable to handle large amounts of data. As a result, a demand for a database that could handle all problems arose, and the NoSQL database was created**.**

When to employ NoSQL:

When storing and retrieving a large amount of data is required.

The connections between the data you store are not really significant.

The data is unstructured and evolves over time.

At the database level, support for constraints and joins is not necessary.

You must frequently scale the database to handle the data because it is constantly expanding.

Types of NoSQL Databases

There are several different kinds of NoSQL databases, and the name of the database system that fits that category is:

- Graph Databases: Examples – Amazon Neptune, Neo4j
- Key value store: Examples – Memcached, Redis, Coherence
- Tabular: Examples – Hbase, Big Table, Accumulo
- Document-based: Examples – MongoDB, CouchDB, Cloudant

Features of NoSQL Databases

1. NoSQL databases feature a dynamic schema, which allows them to adapt to changing data structures without the need for migrations or schema changes.
2. It supports Horizontal Scalability. NoSQL databases are made to scale up by adding more nodes to a database cluster, which makes them suitable for managing massive volumes of data and heavy traffic.
3. A document-based data model is used by several NoSQL databases, including MongoDB, to store data in semi-structured formats like JSON or BSON.
4. Redis and other NoSQL databases use a key-value data model, which stores data as a set of key-value pairs.
5. Data is arranged into columns rather than rows in a column-based data model, which is used by several NoSQL databases like Cassandra.
6. Distributed and high availability: NoSQL databases are frequently made to be highly available and to take care of data replication over numerous nodes in a database cluster as well as node failures.
7. With support for numerous data types and dynamic data structures, NoSQL databases give developers the freedom to store and retrieve data in a flexible and dynamic manner. It Provides Flexibility.

Benefits of NoSQL

Working with NoSQL databases like MongoDB and Cassandra offers numerous advantages, with high availability and scalability being among the most notable benefits. These databases support query languages, deliver swift performance, and enable horizontal scaling.

1. **High Scalability**: NoSQL databases employ sharding for horizontal scaling, partitioning data across multiple machines while preserving its original structure. Horizontal scaling is simpler to implement compared to vertical scaling, making NoSQL databases like MongoDB and Cassandra well-suited for managing large datasets.

2. **Flexibility**: NoSQL databases excel at handling semi-structured or unstructured data and can easily adapt to evolving data models. This flexibility makes them an excellent choice for applications with fluctuating data requirements.
3. **High Availability**: NoSQL databases feature auto-replication, ensuring high availability by automatically copying data to maintain a stable state in case of failures. This redundancy enhances reliability and minimizes downtime.
4. **Scalability**: The inherent scalability of NoSQL databases enables them to efficiently handle massive amounts of data and traffic, making them ideal for applications with demanding scalability requirements.
5. **Performance**: NoSQL databases are optimized for managing large volumes of data and traffic, resulting in superior performance compared to traditional relational databases. This enhanced performance is particularly beneficial for high-throughput applications.
6. **Cost-Effectiveness**: NoSQL databases are often more cost-effective than relational databases due to their simplified design and reduced hardware or software requirements. This affordability makes them a compelling option for organizations seeking to minimize infrastructure costs while maximizing performance.

Disadvantages of NoSQL Databases

ChatGPT

1. **Lack of Standardization**: The wide variety of NoSQL databases, each with its own strengths and weaknesses, can make it challenging to choose the best option for a particular application due to the absence of standardized practices.
2. **Absence of ACID Compliance**: NoSQL databases do not fully adhere to the ACID (Atomicity, Consistency, Isolation, Durability) standard, which can lead to inconsistencies, integrity issues, and data durability concerns, especially for applications requiring strong data consistency guarantees.
3. **Narrow Focus**: While NoSQL databases excel at storage, they often lack robust features for transaction management, making them less suitable for applications with complex transactional requirements compared to relational databases.

4. **Open-Source Nature**: Many NoSQL databases are open-source, which can lead to variations in quality, reliability, and support among different database systems, making it crucial to carefully evaluate each option.
5. **Limited Support for Complex Queries**: NoSQL databases may struggle with handling complex queries and data analysis tasks due to their focus on storage efficiency rather than comprehensive querying capabilities.
6. **Large Document Size**: NoSQL databases like MongoDB and CouchDB often use JSON format for data storage, resulting in large document sizes that can impact network performance and data processing speed, particularly for Big Data applications.
7. **Lack of Maturity**: NoSQL databases are still evolving and may not be as mature or stable as traditional relational databases, potentially leading to reliability and security concerns.
8. **Backup Challenges**: Certain NoSQL databases, such as MongoDB, may lack robust mechanisms for consistent data backup, posing challenges for ensuring data integrity and disaster recovery.
9. **Management Difficulty**: Managing NoSQL databases, especially at scale, can be more complex compared to traditional databases, requiring specialized skills and tools for installation, configuration, and maintenance.
10. **Lack of GUI Tools**: Graphical user interface (GUI) tools for interacting with NoSQL databases may be limited compared to relational databases, requiring developers and administrators to rely more on command-line interfaces and scripting for database management tasks.

Multimedia Databases:

Multimedia databases encompass a wide range of multimedia data types, including text, graphics, photos, animations, videos, and audio, organized and managed within a multimedia database management system (DBMS). These databases are classified into three main classes: dimensional media, dynamic media, and static media.

Key Components of Multimedia Database Management Systems:

1. **Media Data**: Actual data representing multimedia objects.

2. **Media Format Data**: Information about the format of media data after acquisition, processing, and encoding.
3. **Media Keyword Data**: Keywords describing the generation of data, also known as content descriptive data.
4. **Media Feature Data**: Content-dependent data such as color distribution, texture types, and shapes present in the data.

Applications of Multimedia Databases:

- **Repository Applications**: Store large amounts of multimedia data with metadata for retrieval, such as repositories for engineering drawings and satellite photographs.
- **Presentation Applications**: Distribute multimedia data over time, requiring the DBMS to supply data at a specific rate while maintaining optimal viewing or listening experiences.
- **Collaborative Work with Multimedia Information**: Involve collaborative tasks utilizing multimedia data, such as intelligent healthcare networks.

Multimedia databases find applications in industries, knowledge dissemination, education and training, product promotion, shopping, entertainment, and travel, facilitating various tasks like record management, knowledge sharing, learning resources, digital libraries, and virtual tours.

- Real-time monitoring and control: When combined with active database technology, multimedia information display can be a very useful tool for keeping track of and in charge of challenging activities. Control of manufacturing operations, for instance.

Issues with multimedia databases

- Modeling: Documents within multimedia databases represent distinct fields, warranting specialized focus on database retrieval methods rather than conventional information retrieval techniques.
- Design: The conceptual, logical, and physical design aspects of multimedia databases remain incompletely resolved due to the complexity introduced by the inclusion of various formats such as JPG, GIF, PNG, and MPEG. Converting between these formats presents significant challenges, leading to more intricate performance and tuning considerations at each design level.

- Storage - When a multimedia database is stored on a normal disc, representation, compression, mapping to device hierarchies, archiving, and buffering during input-output operations become issues. Untyped bitmaps can be saved and retrieved using a DBMS feature called "BLOB" (Binary Big Object).
- Performance-Physical constraints rule when it comes to applications that involve video playback or audio-video synchronization. Although such techniques have not yet reached their full potential, the utilization of parallel processing may be able to solve some issues. In addition to this, multimedia databases use a lot of processing power and bandwidth.
- Queries and retrieval - With multimedia data like images, videos, and audio, accessing data through a query raises a number of difficulties that need to be addressed, including effective query formulation, query execution, and query optimization.

7.5 Big Data Databases

Big Data refers to vast amounts of data that are characterized by their large volume, variety, and velocity. Traditional data management systems are often inadequate for storing and processing such data due to its sheer size and complexity. The term "Big Data" is often represented by the three Vs:

1. **Volume**: Big Data encompasses large volumes of data, often ranging from terabytes to petabytes and beyond. This includes both structured and unstructured data generated from various sources such as social media, sensors, logs, and transaction records.
2. **Variety**: Big Data comes in diverse formats and types, including text, images, videos, sensor data, and more. Managing and analyzing this variety of data requires specialized tools and techniques capable of handling different data formats and structures.
3. **Velocity**: Big Data is generated at high speeds and arrives rapidly from various sources. This continuous influx of data requires real-time or near-real-time processing capabilities to extract valuable insights and make informed decisions.

In addition to the three Vs, Big Data may also be characterized by other attributes such as veracity (data quality and reliability), value (the potential insights and benefits derived from analyzing the data), and variability (the inconsistency in data flows).

To effectively manage and analyze Big Data, organizations rely on specialized technologies and platforms known as Big Data solutions. Examples of such solutions include Amazon Redshift, Azure Synapse Analytics, Microsoft SQL Server, Oracle Database, MySQL, IBM DB2, and others. These platforms offer advanced capabilities for storing, processing, and analyzing massive volumes of data to uncover valuable insights and drive business decisions.

Overall, Big Data represents a significant opportunity for organizations to gain deeper insights, enhance decision-making processes, and drive innovation across various industries. However, effectively harnessing the power of Big Data requires the right tools, technologies, and expertise to manage its inherent challenges and complexities.

Sources of Big Data

- **Social networking sites:** Social networking sites, such as Facebook, Google, and LinkedIn, produce enormous amounts of data every day due to their massive user bases.
- **E-commerce website:** Sites like Amazon, Flipkart, and Alibaba produce a significant amount of logs that can be used to track customer purchasing patterns.
- **Weather station:** All weather stations and satellites provide enormous amounts of data, which are then stored and processed to forecast the weather. Because they have billions of members globally, social networking sites like Facebook, Google, and LinkedIn all produce enormous amounts of data every day.
- **E-commerce website:** Websites like Amazon, Flipkart, and Alibaba produce a significant number of logs from which customer purchasing patterns can be identified.
- **Weather station:** Every weather station and satellite provides a ton of data that is kept and processed to forecast the weather.
- **Telecom company:** Telecom behemoths like Airtel and Vodafone research customer trends and then announce their plans in line with those findings. To do this, they store the data of their million subscribers.
- **Share Market:** Through their everyday transactions, stock exchanges all over the world produce a significant amount of data.

3V's of Big Data

- Velocity: The data is growing very quickly. The amount of data is expected to double every two years.
- Variety: Data are no longer kept in rows and columns today. Both structured and unstructured data exist. CCTV video and log files are unstructured data. Structured data, such as the bank's transaction data, can be saved in tables.
- Volume: The data that we work with is extremely huge, weighing in at petabytes.

Types of Big Data

The following are the types of Big Data:

1. Structured Structured data refers to any data that can be accessed, processed, and stored in a fixed format. An ‘Employee’ table in a database is an example of Structured Data

2. Unstructured All data that can be retrieved, processed, and stored in a set format is referred to as structured data. Example: The results of a "Google Search"

3. Semi-structured Both types of data are compatible with semi-structured data. Although semi-structured data appears to be structured, it is not defined, for example, by a table definition in a relational DBMS.A data set that is represented in an XML file is an example of semi-structured data. Advantages of Big Data Processing

• Companies can use external intelligence to help them make decisions. Organizations are now able to fine-tune their business plans because to the availability of social data from search engines and websites like Facebook and Twitter.

• Enhanced client Services-New systems created using Big Data technology are replacing conventional consumer feedback systems. Big Data and natural language processing technologies are being employed in these new systems to read and assess customer feedback.

• Improved operational efficiency

• Early detection of any harm to the goods or services.

• Among the benefits of big data include better decision-making, higher operational efficiency, and

1. **Structured Data**: Structured data refers to data that has a predefined format and can be easily organized, processed, and stored in a relational database or similar system.

Examples include data stored in tables with rows and columns, such as transaction records, customer information, and inventory data.

2. **Unstructured Data**: Unstructured data does not have a predefined format and cannot be easily organized into a traditional database structure. This type of data includes text documents, emails, social media posts, multimedia content (images, videos), sensor data, and other forms of raw data that do not fit neatly into rows and columns.
3. **Semi-structured Data**: Semi-structured data lies somewhere between structured and unstructured data. It has some organization or structure but does not conform to the strict schema of structured data. Examples include data in XML files, JSON documents, log files, and NoSQL databases. While semi-structured data may have some level of organization, it does not adhere to a rigid schema like structured data.

Advantages of Big Data Processing:

1. **Informed Decision Making**: Big Data enables organizations to analyze vast amounts of data from various sources, including social media, web logs, and customer interactions, to gain insights and make informed decisions. By leveraging external intelligence, companies can fine-tune their strategies and stay competitive in the market.
2. **Enhanced Customer Service**: Big Data technologies allow organizations to improve customer service by analyzing customer feedback, sentiment analysis, and interactions to understand customer preferences and behavior. This enables companies to personalize their services, address customer concerns proactively, and enhance overall customer satisfaction.
3. **Operational Efficiency**: Big Data analytics can streamline business operations by optimizing processes, identifying bottlenecks, and automating repetitive tasks. By analyzing large datasets, organizations can identify areas for improvement, reduce costs, and increase efficiency across various departments and functions.
4. **Early Detection of Issues**: Big Data analytics can help organizations detect potential issues or anomalies in real-time by monitoring data streams and identifying patterns that indicate potential problems. This proactive approach allows companies to address issues promptly, prevent downtime, and minimize disruptions to operations.

- **Knowledge Check 2**

  State True and False for the Following Sentences.

  1. Big data is a type of data that is small. Big data is defined as data that is more varied, arriving at a faster rate and in larger volumes.
  2. The Four Vs are another name for the Big Data.
  3. The term "multimedia database" refers to a collection of associated multimedia data that may comprise text, graphics (sketches, drawings), photos, animations, video, audio, and other types of data from numerous sources.

- **Outcome-Based Activity 2**

  Describe all types of Practical Applications of Big Data.

7.7 Summary

A distributed database is one where no single CPU controls all the storage devices.

• A parallel database helps to increase performance by parallelizing a variety of tasks, including data loading, creating indexes, and analyzing queries.

• A database system that can handle complex data objects, or objects that resemble those used in object-oriented programming languages, is known as an object-oriented database (OOD). Every element in object-oriented programming (OOP) is an object.

• Data can be supplied and occasionally stored in XML format using an XML database, which is a software solution for data persistence. It is possible to query, modify, export, and then return this data to the caller system.

• A subset of document-oriented databases, which in turn are a subset of NoSQL databases, are XML databases.

• NoSQL, often known as "not simply SQL" or "non-SQL," is an approach to database design that makes it possible to store and query data outside of the conventional structures seen in relational databases.

• A collection of related multimedia data known as a "multimedia database" may include text, graphics (sketches, drawings), images, animations, video, audio, and other types of data from many sources.

• Big data is defined as data that has a wider variety, arrives in greater numbers, and moves more quickly. The three Vs are another name for this. Big data is simply larger, more complex data sets, particularly from new data sources.

7.8 Self-Assessment Questions

1. What is a Parallel and Distributed Database?
2. Explain Object-Based Database.
3. What is XML Database?
4. What is NoSQL Database?
5. Explain Multimedia Database.
6. What is Big Data?
7. Explain the three v's of Big Data.
8. What is Big Data Processing?

7.9 References

- Atkinson, M.P. (1981) *Database*. Maidenhead: Pergamum InfoTech.
- Frank, L. and Helmersen, O. (1988) *Database theory and practice*. Wokingham, England: Addison-Wesley Publishing Company.
- *Database users: 2nd Toronto conference: Selected papers* (1990). Canadian Association for Information Science.
- *Database* (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) *Database*. Rockville, MD: Computer Science Press.

Unit 8

Introduction to Distributed Databases

Learning Outcomes:

- Students will be capable of learning about the Transparencies in a distributed DBMS.
- Students will be able to understand the Distributed DBMS Architecture.
- Students will be able to understand the concepts of Global directory issues and Alternative Design Strategies.
- Students will be able to understand the Distributed design issues.
- Students will be able to learn about Fragmentation and Data allocation.
- Students will learn about view management and Data Security.
- Students will understand the concept of Semantic Integrity Control.

Structure

8.1 Transparencies in a distributed DBMS

Transparency in DBMS refers to the requirement that a DBMS system provides the user with a transparent distribution;in other words, it shields the user from implementation specifics. Transparency is classified into four types: distribution transparency, transaction transparency, performance transparency, andDBMS transparency. In the distributed DBMS, transparency refers to the transparent conveyance of information from the system to the user. It helps to conceal the information that the user must use. Assume that in a standard database management system (DBMS), data independence is a sort of transparency that aids in concealing from the user changes to the definition and organisation of the data. But they all have the same overall purpose in mind.In other words, treat the distributed database as you would a centralised database.

There are four different types of transparency in distributed database management systems, and they are as follows:

1. Transaction transparency
2. Performance transparency
3. DBMS transparency
4. Distribution transparency

1. Transaction Transparency

Due to this transparency, distributed databases' integrity and regularity are maintained throughout all transactions. It is also important to realise that distribution transaction access refers to data held across many places. Another thing to keep in mind is that the DDBMS is in charge of ensuring that each sub-transaction is atomic, which means that it either completes directly or not at all. Becauseof the DBMS's use of fragmentation, allocation, and replication, it is quite complex.

Transaction transparency is a DDBMS property that ensures that database transactions will maintain the distributed database'sconsistency. A distributed transaction accesses data stored at more than one location. Transaction transparency ensures that the transaction will be completed only when all database sites involved in the transaction complete their part of the transaction.

2. Performance Transparency

A DDBMS must function as a centralised database management system in order to achieve this transparency. Also, because of its distributed architecture, the system shouldn't experience any performance drops. A distributed query processor, similar to this, is required by a DDBMS to translate a data request into an orderly series of activities on the local database. The fragmentation, replication, and allocation structures of DBMS add an additional layer of complexity to this, which must be taken into account.

Durability, or all changes that have been saved permanently in a database but must not be lost in the case of any kind of failure, is a requirement for performance transparency in a transaction. Accessing and updating data storages located in several locations must be made more complicated through distributed transactions.

3. DBMS transparency

This transparency is only applicable for diverse types of Distributed DBMS since it hides the possibility of a distinct local DBMS (Databases with several sites and using various operating systems, products, and data models). One of the most difficult generalisations is the generalisation of this transparency.

4. Distributed Transparency

When a DDBMS supports distribution data transparency, the user is not required to be aware that the data is fragmented and can instead see the database as a single object or logical entity.

The five types of distribution transparency are listed below.

a. **Fragmentation Transparency-**Database accesses are based on the global schema because there is no requirement for the user to be aware of fragmented data in this sort of transparency. Like SQL views, the user may be unaware that they are using a view of a table rather than the table itself.

   This distribution transparency level is the highest. The feature allows users to query any table as though it has already been divided up. The fact that the tables being retrieved are, in reality, a fragment or a union of several fragments is concealed. Stated differently, the user is unaware of the fragmentation of the tables. Because of this, fragment names and data locations are not required to be specified by the user because database accesses are all based on global schema.

b. **Location Transparency**-Database accesses are based on the global schema due to the lack of requirement for the user to be aware of fragmented data in this form of transparency.The user may be unaware of the fact that they are utilising a table view rather than the table itself, as with the SQL views.

   The intermediate degree of distribution transparency is location transparency. It guarantees that any tables or fragments may be queried by the user just as if they were locally stored on their site. The user has to be aware of how the data has been divided when there is this level of openness. They are unaware of the data's location, nevertheless. A database management system (DBMS) has to have access to an updated date dictionary and a directory containing the location information of data in order to implement location transparency.

c. **Replication transparency-** The user is unaware of fragment copying in replication transparency.Transparency in the replication is associated with concurrency and failure transparency.When a user modifies a particular data item, the change isalso reflected in all table copies.Yet, the user should be unaware of this operation.

   The user is not aware of the fragments' replication when there is replication transparency. It lets users query a table as though there were just one copy of the table. Concurrency and failure transparency are related to replication transparency. All copies of the table reflect the changes made by the user to a data item. Location transparency implies replication transparency as well.

d. **Local Mapping transparency-** In order to avoid duplications, the user must define both the fragment names and the positioning of the data items in the local mapping transparency. This is a more difficult and time-consuming query for the user in terms of Distributed DBMS transparency.

   the least transparent distribution level. A user must describe the location of data items as well as the names of the fragments in this type of transparency, taking into account any potential duplication.

e. **Naming transparency**- Both DBMS and Distributed DBMS are centralised database management systems. In other words, each item in this database must have its own name. Distributed DBMS must also verify that no database objects with the same name are created by two sites. To address the issue of naming transparency, there are two options:

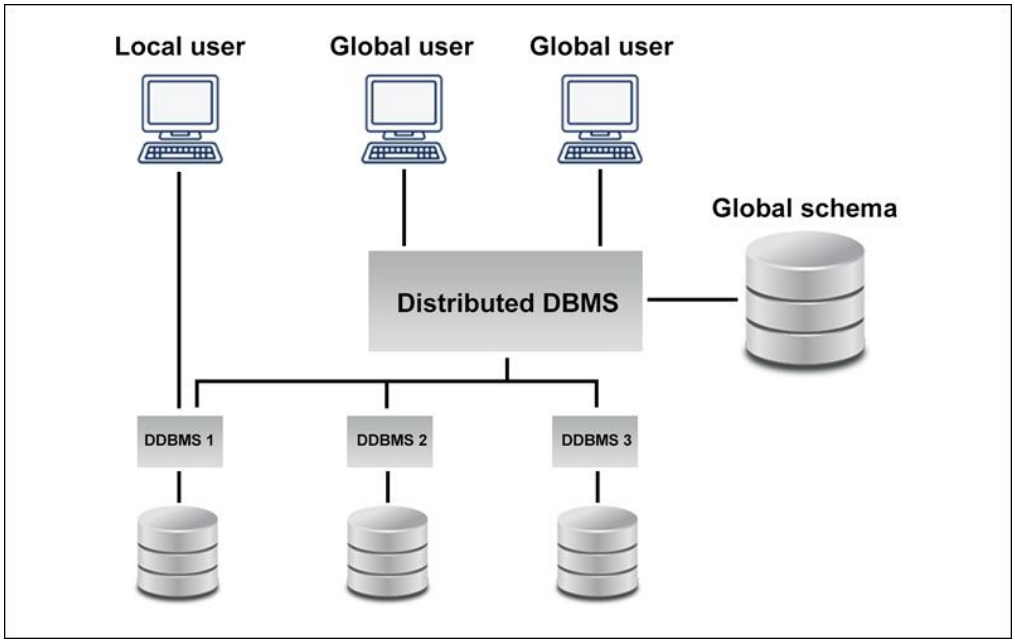
build a central name server to produce unique names for system objects or add an item that starts with the creator site's identity.

8.2 Distributed DBMS Architecture

Distributed database design allows applications to access data from databases that are located both locally and remotely.An Oracle database is present in every database in a homogenous distributed database system.In a heterogeneous distributed database system, there is at least one database that is not an Oracle database. Servers and clients have different levels of functionality in this two-tier architecture. It is separated into client and server sections. Data management, query processing, optimization, and transaction management are the main tasks of the server. User interface is mostly included in client functions.

The core server functions include data management, query processing, optimisation, and transaction administration. The user interface is the most common client function.

Distributed DBMS Architecture



Architectures for Distributed DBMS

Distributed DBMS architectures are typically designed in the three parameters that are mentioned below as follows:

1. **Distribution -**Distributionis used for describing the physical distribution of the various data across distinct sites.
2. **Autonomy**– Autonomy denotes the distribution of database system control and the degree to which each constituent DBMS can operate independently.
3. **Heterogeneity**-It relates to the resemblance or dissimilarity of data models, system components, and databases.

Some common architectural models are as follows:

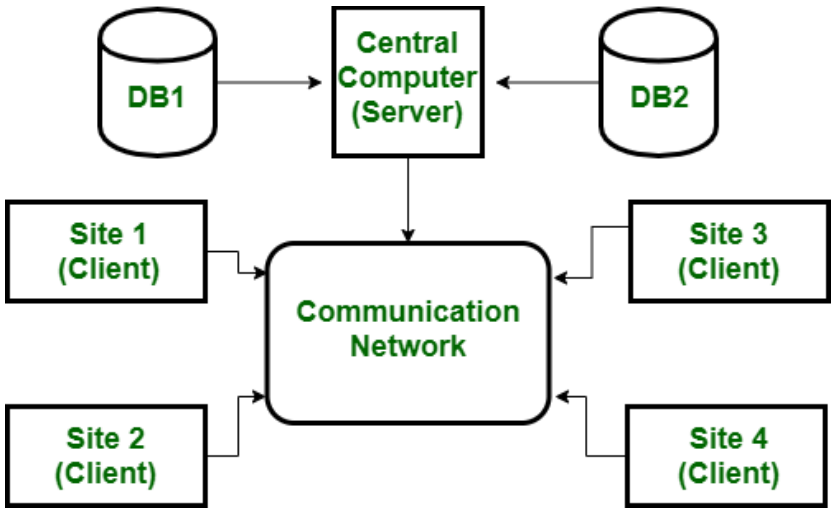
a. Client-Server DDBMS Architecture
b. Peer-to-Peer DDBMS Architecture
c. Multi - DBMS Architecture

a. Client-Server DDBMS Architecture

This two-level design divides functionality between servers and clients. The core server functions include data management, query processing, optimisation, and transaction administration. The user interface is the most common client function. Nonetheless, they do provide some services, such as consistency checking and transaction management.

The two distinct client-server architectures are

i. Single Server Multiple Client
ii. Multiple Server Multiple Client



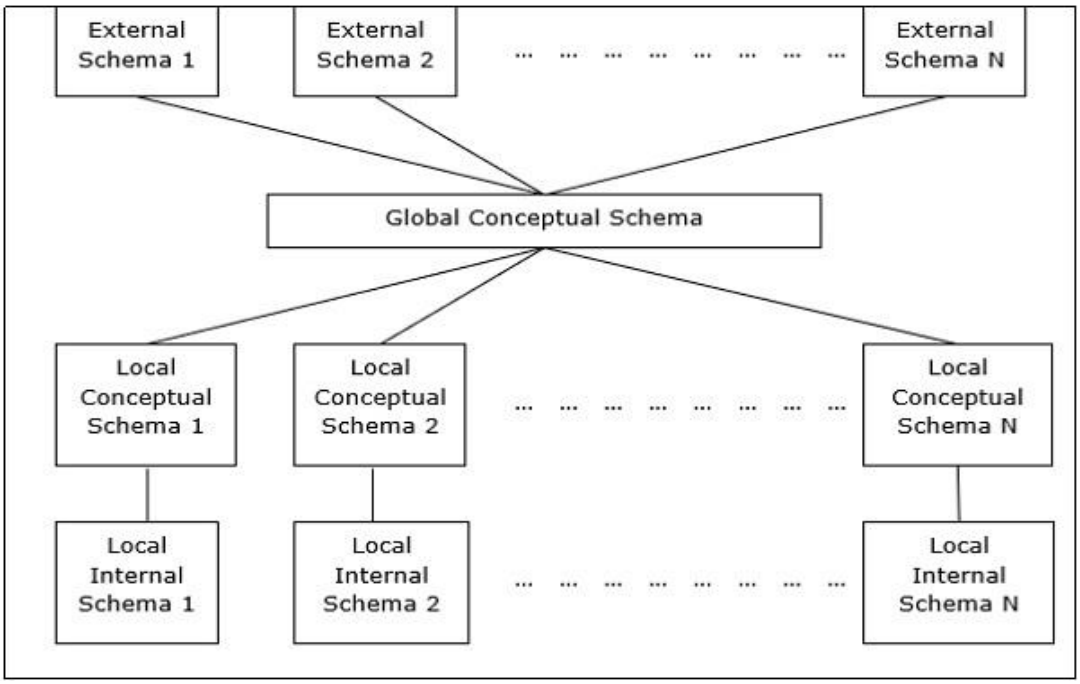
b. Peer-to-Peer Architecture for DDBMS

In these types of systems, each peer serves as both a client and a server for providing the database services.Peers pool their resources and collaborate to coordinate their actions. In general, this architecture has four levels of schemas.

i. **Global Conceptual Schema** – It represents the overall logical picture of data.

ii. **Local Conceptual Schema** – It will show the logical data organisation at each location. **Local Internal Schema** – It will show the physical data arrangement at each location.

iii. **External Schema** – It will represent the user's perspective on data.



c. Multi - DBMS Architectures

An integrated database system is composed of two or more independent database systems. Multi-DBMS can be defined using six levels of schemas.

i. Multi-database View Level– It depicts a number of user views that are subsets of the integrated distributed database.

ii. Multi-database Conceptual Level–It is used to express a multi-database integrated structure composed of global logical multi-database structure specifications.

iii. **Multi-database Internal Level**- It displays data distribution across several sites as well as multi-database to local data mapping.

iv. **Local database View Level**–It represents the public view of the local data.

v. **Local database Conceptual Level**–It displays the organisation of local data at each site.

vi. **Local database Internal Level**– It depicts the physical structure of the data at each location. There are two design possibilities for multi-DBMS, which are as follows:

   a. Model at the conceptual level of many databases.
   b. The model's conceptual level in the absence of a multi-database.

8.3 Global Directory Issues

It Contains information about the fragments' location as well as their composition.The directory is a database in its own right, including meta-data about the actual data stored in the database.

A global conceptual schema is used by remote DBMSs and multi-DBMSs. Global Directory is an addition to the traditional directory that contains details on the composition and placement of the fragments.

- **Global Directory Issues**
   a. Applicable to distributed DBMS or multi-DBMS that use a global conceptual schema.
   b. Includes information about the location and composition of the fragments.
   c. The directory is a database in its own right, containing meta-data about the actual data stored in the database.
   d. A directory might be global to the entire database or site-specific.
   e. The directory may be kept centrally in one location or distributed over numerous locations.
   f. The directory is always distributed if the system is distributed.
   g. Replication can be done in a single or multiple copy.Multiple copies would boost reliability.

8.4 Alternative Design Strategies

The distribution design possibilities for tables in a Distributed DBMS are as follows:

a. Non-replicated and non-fragmented
b. Fully replicated

c. Partially replicated
d. Fragmented
e. Mixed

a. Non-replicated &Non-fragmented

Several tables are positioned in various positions in this design variant.Data is placed near the location where it is most frequently used.It is best suited for database systems when the proportion of queries required to join data from disparate tables is low. This design choice contributes to minimise transmission costs during data processing if a suitable distribution strategy is utilised.

b. Fully Replicated

Each site in this architecture choice holds a single copy of all database tables. Queries are highly fast and incur negligible communication costs because each site has its own copy of the whole database.

In contrast, significant data redundancy needs extravagant expenses during update operations. As a result, this is suited for systems that must process a large number of requests while performing a small number of database updates.

c. Partially Replicated

Table copies or pieces of tables can be found in a variety of places.The tables are arranged in descending order of frequency of access.This takes into account the fact that the frequency with which the tables are consulted varies significantly among sites.The number of copies of the tables (or portions of the tables) is determined by the frequency of access queries that generates the access queries.

d. Fragmented

This architecture divides a table into two or more portions or partitions, and each fragment can be retained in separate locations.This accounts for the fact that it is uncommon for all data recorded in a table to be required at a single location.In addition, fragmentation enhances

parallelism and disaster recovery.Each fragment in this situation has only one copy in the system, resulting in no redundant data.

The three fragmentation techniques thatare given below as follows−

1. Vertical fragmentation
2. Horizontal fragmentation
3. Hybrid fragmentation

e. Mixed Distribution

Mixed Distribution is the result of incomplete replications and fragmentation.First, the tables are broken up into smaller parts in either direction (horizontal or vertical), and then the pieces are replicated in part over other locations according to how often the individual sections are accessed.

8.5 Distributed Design Issues

The two critical design concerns are fragmentation (the separation of the database into fragments) and distribution (the best distribution of pieces). A distributed information system is defined as "a collection of networked computers linked together via a network with the goal of sharing information among them."A distributed information system is made up of numerous independent computers that communicate with one another and share data via a computer network.

1. Heterogeneity:This word refers to different computer hardware, operating systems, networks, and implementations used by developers.A key element of the client-server ecosystem of heterogeneous distributed systems is middleware.Middleware is a collection of services that facilitates communication between applications and end users in a heterogeneous distributed system.

2. Openness: The ease with which new resource-sharing services may be made accessible to consumers determines how open a distributed system is.Open systems are distinct in that their essential interfaces are published.It is predicated on a widely used interface for public resource

access and a common communication protocol.Many different pieces of hardware and software may be used to build it.

3. Scalability: The system's scalability should remain efficient even when the number of users and resources linked grows significantly. It should make no difference whether a programme has 10 or 100 nodes; performance should be consistent. Scaling a distributed system necessitates consideration of several factors, including size, geography, and administration.

4. Security: Information system security has three components. Confidentiality, integrity, and accessibility are all important considerations. Encryption safeguards shared resources and conceal sensitive information while it is transmitted.

5. **Failure Handling:**It is necessary to establish remedial procedures to deal with the scenario when hardware and software malfunctions lead to inaccurate results or halt before completing the intended computation. Because partial failures in distributed systems occur when some components fail while others continue to operate, managing these kinds of failures can be challenging.

6. Concurrency: Concurrency happens when numerous clients attempt to access the same shared resource at the same time. Many users make requests to read, write, and update the same resources. Each resource in a concurrent context must be safe. Any object representing a shared resource in a distributed system must ensure that it performs correctly in a concurrent environment.

7. Transparency: By being transparent, a distributed system is perceived as a single entity by users or application programmers, as opposed to a group of independent entities that collaborate. The changeover from a local to a remote workstation should be smooth, and the user should be oblivious of where the services are located.

- **Knowledge Check 1**

  Fill in the Blanks

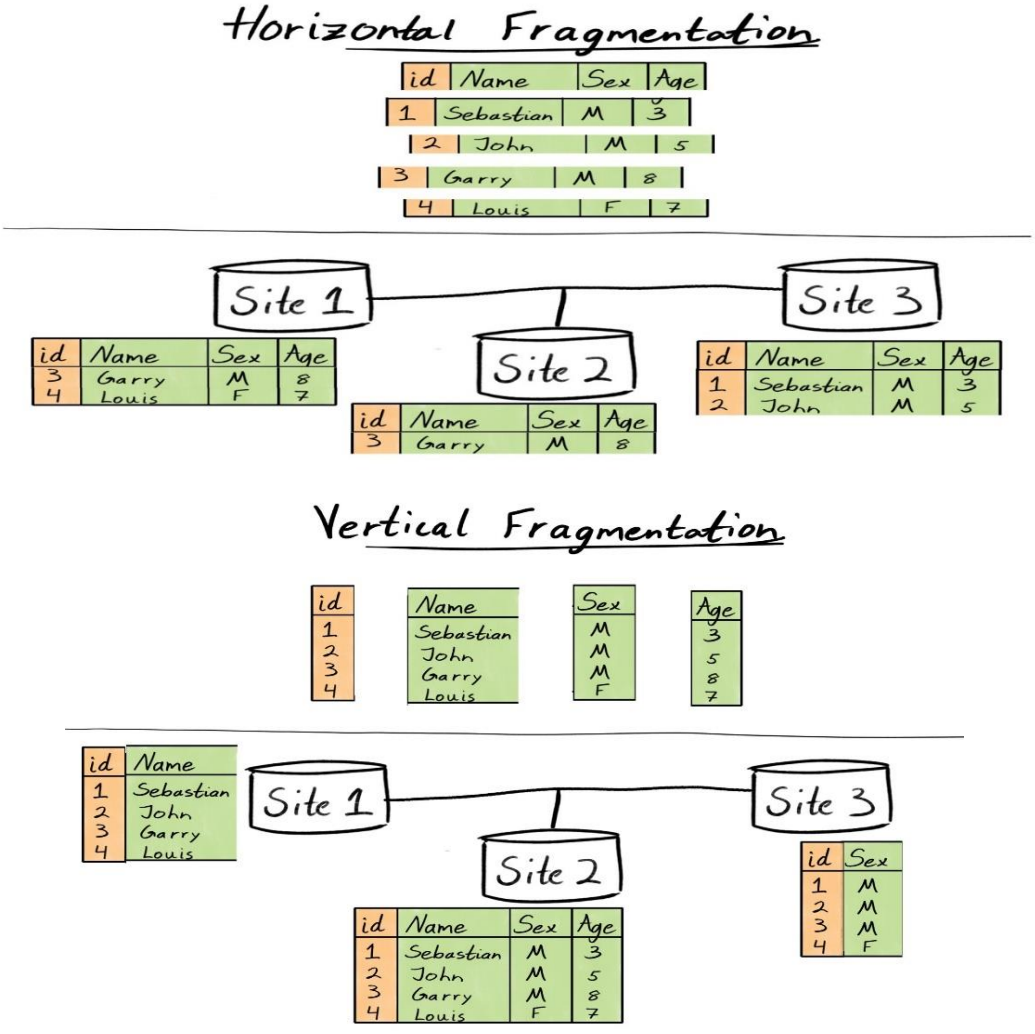
  1. _____ in DBMS refers to the requirement that a DBMS system provides the user with a transparent distribution
  2. _____is an extension of the conventional directory that includes information about the location of the fragments as well as the makeup of the fragments
  3. _____tiers of schemas can be used to define multi-DBMS.”

- **Outcome-Based Activity 1**

  Describe thevarious types of architecture of the Distributed DBMS.

8.6 Fragmentation

The process of breaking a table up into smaller tables is called fragmentation. We refer to the table's subsets as fragments. The process of breaking a table up into smaller tables is called fragmentation.The table's subsets are called fragments.Three forms of fragmentation are distinguished: hybrid, which combines vertical and horizontal elements, and horizontal. Primary and derived horizontal fragmentation are two further categories into which horizontal fragmentation techniques may be separated. It is necessary to fracture the table in order to reassemble the original from the pieces. This is needed in case it becomes essential to reconstruct the original table using the pieces. That's what we call "reconstruction."



Advantages of Fragmentation

1. When data is stored close to the point of use, the database system's performance improves.
2. Local query optimisation procedures are sufficient for some queries because the data is available locally.
3. Fragmentation aids in the confidentiality and privacy of the database system.

Disadvantages of Fragmentation

1. Access times may be extremely slow when data from numerous fragments is requested.
2. Rebuilding recursive fragmentations will need the deployment of costly techniques.
3. If a site fails, a lack of backup copies of data on other sites may leave the database inoperable**.**

- **Types of Fragmentation**

1. Vertical fragmentation

Vertical fragmentation separates the fields or columns of a table into fragments. To maintain reconstructive, each piece should have the primary key field of the table (s). Vertical data fragmentation can be used to safeguard data privacy**.**

2. Horizontal Fragmentation

The tuples in a table are organised horizontally based on the values of one or more fields. Horizontal fragmentation should also support the rule of reconstructive. Each horizontal segment must have all columns from the original base table.

3. Hybrid fragmentation

The horizontal and vertical fragmentation techniques are combined in a hybrid fragmentation approach. This is the most adaptive fragmentation method since it produces fragments with the least amount of extraneous information. Sadly, replicating the original table is sometimes an expensive procedure.

There are two methods for hybrid fragmentation.

1. Create a collection of horizontal pieces first, then create vertical fragments from one or more of the horizontal fragments.
2. Create a collection of vertical pieces first, then create horizontal fragments from one or more of the vertical fragments.

8.7 Data Allocation

Data Allocation is the intelligent allocation of your data bits (called data fragments) to increase database performance and data availability for end users. It seeks to lower total transaction processing costs while also providing reliable data quickly in your DDBMS systems. Despite providing numerous advantages over centralised database systems, data allocation is one of the fundamental design difficulties of a distributed database system. Data Allocation is an important phase in the development of Distributed Database Systems.Data Fragmentation and Data Replication are two typical tactics used in efficient data allocation.

The strategies used for distributed database design are classified as follows:

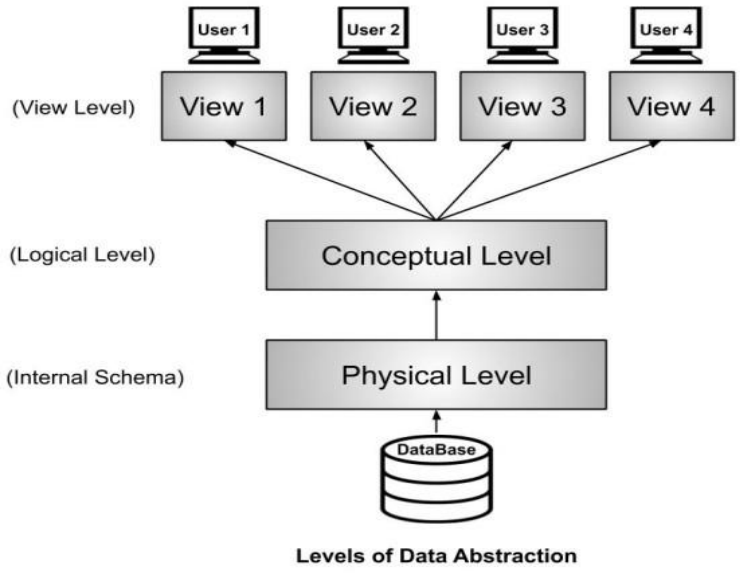
1. **Fragmentation:** A technique for breaking up a database into logical chunks is called asfragmentation.Fragments can be assigned to different storage locations.Vertical fragmentation, horizontal fragmentation, and hybrid fragmentation are the three types of fragmentation procedures.The fragmentation technique provides several advantages, including increased efficiency, local query optimisation, security, and privacy. In a horizontal fragmentation, each tuple of a relation r is assigned to one or more pieces. Vertical fragmentation divides a relation r's schema into several smaller schemas that share a common candidate key and a specific property.
2. **Replication:** Data replication is a mechanism that allows specific data or copies of a database to be kept in more than one location. It is a well-known distributed database fault tolerance approach. Furthermore, some of the primary benefits of the data replication technique include reliability, faster response, reduced network traffic, and simpler transactions. Yet, there are several downsides to Distributed Database Replication. To ensure precise and proper responses to user inquiries, data must be regularly updated and synchronised. Failure to do so can result in data inconsistencies, which might stymie corporate goals and judgements made by other departments.
3. **Allocation:** The allocation technique is used to allocate pieces or clones of fragments for storage at different locations.

   A global directory stores all data fragmentation, allocation, and replication information. The DDBS apps can access this directory as needed.

8.8 View management

A view in a relational database management system is defined as a "virtual table" derived from a specific query on one or more underlying tables. A view can be defined using relational operations such as join, restrict, and project, as well as statistical summaries of data. View offers complete logical data independence. Views are virtual relations that are defined as the result of a query on base relations. Views are not usually realised. It is a dynamic window that displays all important database updates.

The view definition in DDBMSis similar to that of centralised DBMS.A view in a DDBMS, on the other hand, can be derived from fractured relations maintained at separate locations.



Levels of Data Abstraction

Advantages of View Management

1. It Provides Data Integrity.
2. It helps in proving Logical Data Independence.
3. Views are also value in context of the security.
4. It helps in Shieldingfrom change.
5. It provides Easier querying.

8.9 Data Security

A distributed system needs more security measures than a centralised one since it has more users, diversified data, various sites, and dispersed control.

In this chapter, we will examine the many facets of distributed database security.

In distributed communication systems, intruders fall into two types:

1. Passive eavesdroppers watch messages and obtain sensitive information.
2. In addition to keeping an eye on communications, active attackers actively contribute new data or modify existing data.

Security measures include communications security, data security, and data auditing.

- **Communications Safety**

Due to the diverse location of data, users, and transactions in a distributed database, a great deal of data communication occurs. As a result, it necessitates safe communication between users and databases, as well as between database environments.

The following aspects of communication security are included:

Data should not be corrupted during transfer, and the communication connection should be secure against both passive and aggressive eavesdroppers.

Well-defined security methods and protocols should be employed to meet the aforementioned standards.

End-to-end secure communications are ensured by two common and consistent technologies.

a. Secure Socket Layer Protocol (SSL) or Transport Layer Security Protocol (TLSP).
b. Virtual Private Networks (VPN).

Data Security

Aside from communications, it is critical in distributed systems to implement measures to secure data.

The data security measures are.

a. Authentication and authorisation are access control procedures that ensure only authorised users have access to the database. Digital certificates are used to establish authentication. In addition, login is restricted to a username/password combination.
b. There are two ways for data encryption in distributed systems. The user apps encrypt the data before putting it in the database, which is inherent in the distributed database strategy. Before using it, the programmes retrieve encrypted data from the database and decrypt it. Outside of the distributed database system: The distributed database system has its own encryption

capabilities. The user applications save and retrieve data without realising that the data is encrypted in the database.

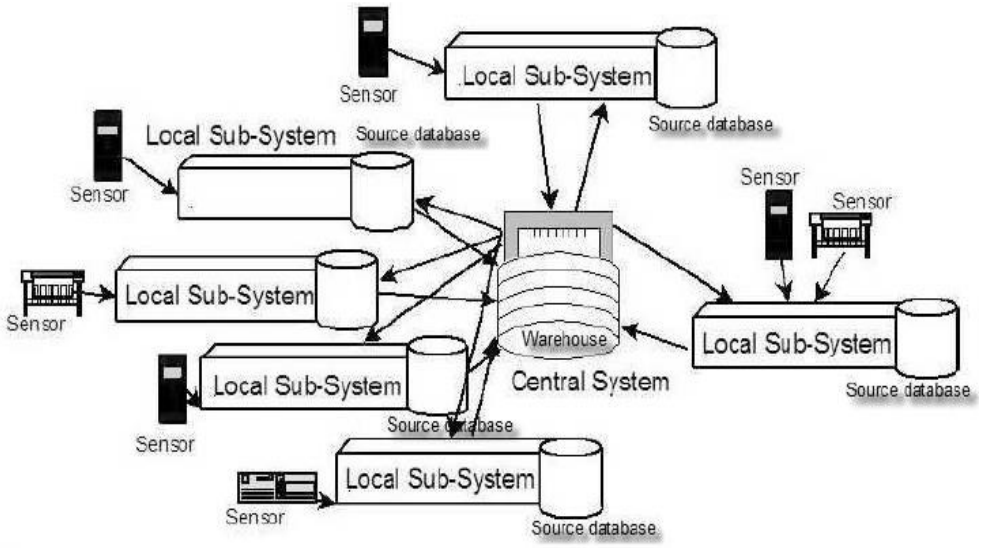
c. Validated input - Under this security measure, the user application validates each input before it can be used to change the database.
d. Invalidated input can be used to launch a number of attacks, including buffer overruns, command injection, cross-site scripting, and data corruption.

- **Data Auditing**

A database security system must detect and monitor security infractions in order to determine which security measures to implement.It is frequently difficult to detect security breaches as they occur. Checking audit logs is one method of finding security flaws.

8.10 Semantic Integrity Control

Semantic integrity ensures that data entered into a row conforms to a legitimate row value. The value must be inside the column's domain or allowable range of values. In the items table, for example, the quantity column only allows numbers. The integrity constraints of the database system are defined and enforced via semantic integrity control.



The following are the integrity constraints:

a. Data type integrity constraint
b. Entity integrity constraint
c. Referential integrity constraint

1. Data Type Integrity Constraint

The range of values and kinds of operations that can be carried out on a field of a certain data type are limited by a data type constraint. Examine the "HOSTEL" table, which consists of three fields: capacity, hostel name, and hostel number. The hostel can accommodate up to 150 people, and its number must start with the capital letter "H" and cannot be NULL.

2. Entity Integrity Control

The regulations are upheld via entity integrity control, enabling each tuple to be distinguished from the others. A primary key is defined for this. A primary key comprises the bare minimum of information necessary to uniquely identify a tuple. No two tuples in a table may have the identical primary key value, and no primary key field can have a NULL value, according to the entity integrity criteria.

3. Referential Integrity Constraint

The referential integrity constraint establishes foreign key constraints. A field in one data table that serves as the main key in another is known as a foreign key. The referential integrity constraint states that the value of the foreign key field cannot be NULL at all or one of the values of the referenced table's primary key.

- **Knowledge Check 2**

  State True and False

  1. Replication technique for breaking up a database into logical chunks is called as fragmentation.
  2. A view managementstores all data fragmentation, allocation, and replication information. The DDBS apps can access this directory as needed.
  3. The referential integrity constraint establishes the restrictions for foreign keys.

- **Outcome-Based Activity 2**

  Given the following relation EMP and the predicates p1: SAL > 23000, p2: SAL <23000

ID	Name	Salary
1289	Guru	12000
8907	Siva	67050
7643	Shalini	51980
0988	Kavin	23000
6543	Surya	28760
0986	Kavitha	23000
2345	Anees	29999

Perform a horizontal table fragmentation based on the predicates provided.

8.11 Summary

- Transparency in DBMS refers to a DBMS system that provides a transparent distribution to the user. In other words, it shields the user from implementation details. Transparency is classified into four types: distribution transparency, transaction transparency, performance transparency, and DBMS transparency.
- The global directory Contains information about the fragments' location as well as their composition. The directory is a database in and of itself, including meta-data about the real data saved in the database.
- The task of separating a table into smaller tables is known as fragmentation. The table's subsets are referred to as fragments.
- There are three types of fragmentation: horizontal, vertical, and hybrid (a combination of horizontal and vertical).
- The primary goal of data allocation in a distributed database is to divide data pieces over multiple sites in such a way that the total data transmission cost is reduced while running a series of queries.
- Data security measures are implemented. Authentication and authorisation are access control procedures that are used to ensure that only authorised users have access to the database. Digital certificates are used to verify authentication. Login is also restricted to a username/password combination.

- Semantic integrity assures that data placed into a row reflects an acceptable value for that row. The value must fall inside the domain, or permissible set of values, for that column. For example, the quantity column in the items table only accepts numbers.

8.12 Self-Assessment Questions

1. What are Transparencies in distributed DBMS?
2. Explain DBMS architecture.
3. What are Global directory issues?
4. What are Alternative design strategies?
5. Explain Distributed design issues.
6. What are Fragmentation and Data allocation?
7. What is view management?
8. What is Semantic Integrity Control?

8.13 References

- Atkinson, M.P. (1981) Database. Maidenhead: Pergamum InfoTech.
- Frank, L. and Helmersen, O. (1988) Database theory and practice. Wokingham, England: Addison-Wesley Publishing Company.
- Database users: 2nd Toronto conference: Selected papers (1990). Canadian Association for Information Science.
- Database (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) Database. Rockville, MD: Computer Science Press.

Unit 9

Query Processing in Distributed Database

LearningOutcomes:

- StudentswillbecapableoflearningabouttheObjectivesofqueryprocessing.
- StudentswillbeabletounderstandtheCharacterisationofqueryProcessors.
- StudentswillbeabletounderstandtheconceptsofLayersofqueryprocessing.
- StudentswillbeabletounderstandtheQuerydecomposition.
- StudentswilllearnabouttheLocalisationofdistributeddata.

Structure

9.1 Objectives of query processing

Query Processing

• The translation of high-level requests into low-level phrases is known as query processing. It is a step-by-step procedure that may be used to achieve the result at the physical level of the file system, query optimisation, and actual query execution. It requires a basic understanding of relational algebra and file structure. Query optimisation is the process of determining an efficient execution strategy for processing a query. The query processor is the subcomponent of the data server that processes SQL requests. The SQL requests can access a single database or file system or reference multiple types of databases or file systems. Query processing is the action of extracting data from a database.

In query processing, multiple procedures are used to retrieve data from the database.

The steps are as follows:

1. Parsing and translation
2. Optimisation
3. Evaluation

Objectives of Query Processing

The main and primary objectives of distributed query processing are to translate a high-level query on a distributed database, which users perceive as a single database, into an efficient execution strategy defined in a low-level language in local databases. The goal of query optimisation is to find an execution plan that reduces the time required to run a query. To reach this optimisation goal, we must carry out two critical tasks. The first is to determine the most efficient method of accessing the database, and the second is to reduce the time it takes to run the query plan.

a. The purpose of distributed query processing is to translate a high-level query from a distributed database into a low-level language that can be executed on local databases. There are several stages to the query transformation.

b. The optimiser attempts to provide the optimal execution plan possible for a SQL statement. The optimiser chooses the plan with the lowest cost among all candidate plans considered. The optimiser computes expenses based on the information provided. The cost computation takes

into account query execution factors such as I/O, CPU, and connectivity for a certain query in a given context

c. As the database has so many internal statistics and tools at its disposal, the optimiser is frequently in a better position to determine the best way to execute a statement than the user. As a result, all SQL statements make use of the optimiser.

d. There is a query, for example, that requests information about students in positions of leadership, such as class reps. If the optimiser statistics show that half of the students are in positions of leadership, the optimiser may decide that a complete table search is the most efficient. However, if data indicate that just a small number of students are in positions of leadership, reading an index followed by database access by row id may be more efficient than a full table scan.

9.2 Characterisation of query Processors

The first four features apply to both centralised and distributed query processors, whereas the next four are exclusive to the distributed query processors in closely integrated distributed DBMSs.

a. Languages
b. Optimisation Types
c. Optimisation 1 3 3 5 13 14 16 19
d. Timing
e. Statistics
f. Sites of Decision
g. Use of Network Topology
h. Replicated Fragment Exploitation
i. Use of Semi joins

1. **Types of Optimisation**

a. Exhaustive search- The goal of query optimisation is to find the "best" point in the solution space of all possible execution techniques. Search the solution space for the cost of each approach and select the strategy with the lowest cost. Although this method is successful for finding the optimum strategy, the optimisation itself may entail large processing costs. The

problem is that the solution space can be enormous, which means that even with a limited number of relations, there may be numerous comparable techniques.

b. Heuristics Search- Heuristics are a popular method of decreasing the cost of an exhaustive search by restricting the solution space to a few methods that gather common subexpressions. Do selection and projection first, then replace a join with a sequence of semijoins, reorder processes to reduce intermediate relation size, and optimise individual operations to minimise data transfer.

c. Randomised strategies- Randomised strategies Pick a really good solution; it may not necessarily be the best one, but avoid the large cost of optimisation in terms of memory and time usage.

2**. Optimisation Timing**

Optimisation can be done statically before running the query or dynamically while it runs.

a. Static Static query optimisation occurs during query compilation. As a result, the expense of optimisation can be spread across numerous query runs. This time frame is suitable for usage with the exhaustive search strategy. As the sizes of a strategy's intermediate relations are unknown until run time, they must be calculated using database statistics. 1 1

b. Dynamic Dynamic run-time optimisation database statistics are not required to estimate the size of intermediate outcomes. The fundamental advantage of dynamic query optimisation over static query optimisation is that the query processor may see the actual sizes of intermediary relations, reducing the likelihood of a wrong choice. The fundamental drawback is that query optimisation, an expensive job, must be done for each query run. As a result, this method is suitable for ad hoc queries.

c. Hybrid It gives the benefits of static query optimisation. The technique is primarily static, but dynamic query optimisation may occur during run-time if a significant difference between expected and actual sizes of intermediate relations is observed. If the error in estimate sizes exceeds a certain threshold, re-optimise at run time.

3. Statistics

a. The efficiency of query optimisation is dependent on database statistics.

b. Statistics are required for dynamic query optimisation to decide which operators should be run first.

c. Static query optimisation is substantially more challenging due to the need to calculate the size of intermediary relations using statistical data.

d. Statistics for query optimisation frequently include fragment cardinality and size, as well as the size and number of distinct values for each feature.

e. More extensive statistics, such as histograms of attribute values, are sometimes used to lessen the risk of error.

f. Statistics are kept up to date on a regular basis, which ensures their accuracy.

g. When using static optimisation, significant changes in the statistics used to optimise a query may result in query optimisation.

4. Decision Sites

- Centralised decision approach The approach that determines the "optimal" timetable is generated by a single site. Simpler knowledge of the complete distributed database is also needed to be required.
- Distributed decision approach The distributed decision approach requires just local knowledge to decide the timetable (elaboration of the optimum strategy).
- Hybrid decision approach One site makes important decisions that influence the global timetable. Other sites make local decisions that maximise the local sub-queries.

5. Network Topology

Network topology is the physical and logical organisation of nodes and links in a computing network. The distributed query optimisation problem is divided into two parts: selecting a global execution strategy based on inter-site communication and selecting each local execution strategy based on a centralised query processing algorithm.

a. Wide area networks (WAN) - The cost of point-to-point communication will be the most important. It disregards all other cost variables. Local scheduling based on centralised query optimisation.

b. Local area networks (LAN)- Communication costs are similar to I/O costs. It improves parallel execution while increasing communication costs. The broadcasting capability of some local area networks can be successfully exploited to optimise join operator processing. Algorithms for star networks are unique.

6. Languages

Relational DBMSs employ relational calculus.Object DBMSs employ Object calculus, which is an extension of RDBMS. XML is a data model that uses Xquery and XPath for storing and transmitting data over the internet. XPath is an xml path language that is used to utilise queries to pick nodes from an xml document.XQuery extracts and manipulates data from xml documents, relational databases, and Microsoft Office documents that support an xml data source.

9.3 Layers of Query Processing

Query processing has four layers which are given below as follows: -

1. Query Decomposition
2. Data Localisation
3. Global Query Optimisation
4. Distribution Query Execution

1. Query Decomposition

The first layer is used for converting the calculus query into an algebraic global relationship query. The information required for this transition is contained in the global conceptual schema reflecting global relations.

Query decomposition can be viewed as four successive steps

a. Normalisation

b. Analysis

c. Simplification

d. Restructure

i. The calculus question is first rewritten in a normalised form that can be modified further. Normalising a query frequently requires modifying query quantifiers and query qualification via logical operator priority.

ii. Second, the normalised question is semantically analysed in order to detect and reject incorrect requests as early as possible. Techniques for identifying erroneous queries are

available in just a subset of relational calculus. They frequently use graphs to capture the semantics of the query.

iii. Third, the correct question (which is still expressed in relational calculus) is simplified. Removing superfluous predicates is one method for simplifying a query.It should be noted that when a query is the result of system transformations done on the user query, it is likely that redundant queries may occur. These transformations are used to govern semantic data (views, protection, and semantic integrity control).

iv. Fourth, the calculus query is reorganised as an algebraic query. The standard way for obtaining a "better" algebraic specification is to start with an algebraic inquiry and adjust it to reach a "objective."

2. **Data Localisation**

As input, the second layer receives an algebraic query on global relations. The major goal of the second layer is to localise the query's data using data distribution information from the fragment schema. This layer determines which fragments are involved in the query and translates it to a query on fragments.By applying the fragmentation criteria and then producing a programme of relational algebra operators that act on parts, a global relation can be restored. Repetition: Relationships are fragmented, with each kept in a different location. Fragmentation is specified by fragmentation rules or schemes. This layer also detects which fragments are in the query and converts the distributed query into a fragment query. By substituting each distributed relation with its reconstructive programme, the distributed query is mapped into a fragment query (materization program). The fragment query is simplified and rebuilt to yield another "excellent" query (applying the same rules used in the decomposition layer.)

A fragment query is created in two steps.

a. The query is first turned into a fragment query by replacing each relation with its reconstruction programme (also called the materialisation program).

b. The fragment question is then simplified and restructured to generate another "excellent" query.

3. Global Query Optimisation

The third layer is fed an algebraic query on parts. The goal of query optimisation is to identify a query execution strategy that is close to optimal. Earlier stages have already optimised the

query, such as by deleting duplicated expressions. This optimisation, however, is unaffected by fragment properties like as fragment allocation and cardinalities. The process of obtaining the "optimal" ordering of the query's operators, which includes communication operators that minimise a cost function, is known as query optimisation. The query optimisation layer generates an algebraic query that is optimised and includes communication operators on fragments. It's frequently represented and saved as a distributed query execution plan (for future executions). Query optimisation is the process of determining the "optimal" ordering of the query's operators, including communication operators that minimise a cost function. The cost function, which is 2 20 frequently expressed in terms of time units, refers to computer resources like disc space, disc I/Os, buffer space, CPU cost, communication cost, and so on.It is, in general, a weighted combination of I/O, CPU, and communication expenses. Yet, as previously said, an early distributed DBMS simplification was to consider communication cost as the most important criterion.This used to be true for wide-area networks, where restricted bandwidth rendered communication significantly more expensive than local processing. This is no longer the case because communication costs can be lower than I/O costs. To choose an operator ordering, it is important to forecast the execution costs of alternative candidate orderings. Calculating execution costs prior to query execution (i.e., static optimisation) is based on fragment statistics and formulas for predicting the cardinalities of relational operator results. Hence, optimisation decisions are influenced by fragment allocation and accessible information on fragments that are recorded in the allocation schema.

Query optimisation consists of

Identifying the optimum order of operations in the fragment query,

- Identifying the communication procedures that minimise a cost function.
- The cost function refers to computational resources such as disc space, disc I/Os, buffer
- space, CPU cost, communication cost, and so on. Instead of join operators, the semi-join operator is used to optimise queries.

4. **Distribution Query Execution**

   The last layer is processed by all sites that have fragments in the query. Each sub-query executing at a single site, known as a local query, isthen optimised and executed using the site's localschema. At this point, the algorithms for performing the relational

operations can be selected. Local optimisation leverages centralised system methods. The purpose of distributed query processing is to identify a suitable execution strategy that minimises a system cost function that incorporates I/O, CPU, and communication costs given a calculus query on a distributed database. An execution strategy is given in terms of relational algebra operators and communication primitives (send/receive) applied to local databases (i.e., the relation fragments). As a result, the complexity of relational operators affecting query execution performance is critical in the design of a query processor.

Knowledge Check 1

Fill in the Blanks.

1. _____is the process of determining the "optimal" ordering of the query's operators, which includes communication operators that minimise a cost function.
2. _____layer is executed by all sites that have fragments in the query
3. _____primary function is to localise the query's data using data distribution information from the fragment schema.

Outcome-Based Activity 1

- List out all the steps of Query Decomposition.

9.3 Query decomposition

The query decomposition phase of query processing tries to turn a high-level question into a relational algebra query and to validate that query for syntactic and semantic correctness. The first stage of query processing is query decomposition. The basic goals of query decomposition are to convert a high-level query into a relational algebra query and to validate the query for syntactic and semantic correctness. The typical steps of query decomposition include analysis, normalisation, semantic analysis, simplification, and query rearrangement. The first phase of query processing is query decomposition which converts a relational calculus

- query into a relational algebra query. Without knowing the distribution of data, both input and output queries pertain to global relations. As a result, question decomposition is

identical in centralised and distributed systems. The input query is assumed to be syntactically correct in this section. When this stage is

- completed properly, the output query issemantically correct and good in the sense that it avoids unnecessary labour. The following are the steps in question decomposition:
  (1) standardisation,
  (2) analysis,
  (3) redundancy elimination, and
  (4) rewriting. Steps 1-4 rely on the notion that for a given query, multiple transformations need to be
- equivalent, and some may perform better than others. We give the first three phases of tuple relational calculus (e.g., SQL). The query is only converted to relational algebra in the final phase. 1 4 4 8 Query decomposition can be viewed as four successive steps. a. Normalisation b. Analysis c. Simplification d. Restructure Example of Query Processing Query: select salary from instructor where salary < 75000; This query can alternatively be expressed as one of the following relational algebra expressions: σsalary
- <75000(Πsalary (instructor)) Πsalary (σsalary
- <75000(instructor))

9.4 **Localisation of distributed data**

Data localisation is the practice of retaining data within the location from whence it originated.

- For example, if a company obtains data in the United Kingdom, it will keep it there rather than send it to another country for processing. As input, the second layer receives an algebraic query on global relations. The primary and main purpose of the second layer is to localise the query's data using fragment
- schema data distribution information. This layer determines which fragments are involved in the query and converts it to a query onfragments.
- A global relation can be recreated by applying the reconstruction criteria and generating relational algebra programme whose operands are the pieces; this process is known as the localisation Program.

- A global relation can be restored by applying the fragmentation criterion and then constructing a programme of relational algebra operators that act on pieces.
- The localisation layer converts an algebraic query on global relations into an algebraic query on Physical Fragments.

12 21 Localisation takes the use of data stored in the fragment schema.

- Fragmentation is defined
- by fragmentation rules, which can be expressed as relational queries. Purpose It applies data distribution information to algebra operations and assists in determining which
- fragments are commonly involved. It replaces global queries with queries on fragments.
- It improves the global query"
- Knowledge Check 2
- State true and false for the Following Sentences.

  1. The first phase of query processing is query decomposition which converts a relational calculus query into a relational algebra query.

  2. Query decomposition can be viewed as six successive steps.

  3. Data localisation is the practice of retaining data within the location from whence it originated.

Outcome-Based Activity 2

- Illustrate all the steps of Query Processing.

9.5 **Summary**

Query Processing is the process of converting high-level inquiries into low-level expressions.

- It is a step-by-step procedure that can be utilised at the physical level of the file system, query optimisation, and actual query execution to obtain the result. It necessitates a basic understanding of relational algebra and file structure.

There are four Layers of Query Processing.

a. Query Decomposition.

b. Data Localization.

c. Global Query Optimisation.

d. Distribution Query Execution.

The query decomposition phase is the initial stage of query processing, and its goals are toconvert a high-level query into a relational algebra query and to validate that query forsyntactic and semantic correctness.

The practice of storing and processing data inside a certain geographic location is referred to as data localisation.

It might be prompted by a number of considerations, including national rules or regulations, security and privacy concerns, or a desire to keep data closer to users.

To reduce data transfer across sites, the global optimiser develops a distributed execution plan.

The plan describes the sequence in which query stages must be performed, as well as the activities involved in transmitting intermediate results.

All sites with query fragments execute the final layer. Each sub-query done at a single site is then optimised and executed using the local schema of that site.

9.7 Self-Assessment Questions

1. What is Query Processing?
2. What are the objectives of Query Processing?
3. Explain the various Characterisation of Query Processing.
4. What are the different layers of Query Processing?
5. What is Query decomposition?
6. What is the Localisation of distributed data?
7. What are the steps involved in Query Decomposition?
8. What is the goal of the Localization of distributed data?

9.8 References

Atkinson, M.P. (1981) Database. Maidenhead: Pergamum InfoTech.

•Frank, L. and Helmersen, O. (1988) Database theory and practice. Wokingham, England:

- Addison-Wesley Publishing Company. Database users: 2nd Toronto conference: Selected papers (1990). Canadian Association for•Information Science. Database (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) Database. Rockville, MD: Computer Science Press.

Unit10

Distributed Query Optimisation

Learning Outcomes:

- Students will be capable of learning about the Factors governing query optimisation.
- Students will be able to understand theCentralised query optimisation.
- Students will be able to understand the Ordering of Fragment Queries.
- Students will be able to understand Distributed Query Optimization algorithms.

Structure

10.1 Factors governing query optimisation

An effective way to access the database is through query optimization. It's an art to get needed information in a timely, consistent, and dependable way. The practice of changing a query into a form that is equivalent and can be assessed more quickly is known as query optimization.

Selecting the optimal execution plan for a given query within the restrictions of available resources is the aim of query optimization. The query describes the intended result, or the user's purpose, but it makes no mention of how the output should be generated.

Query optimization is the technique used to ensure that a database query runs as efficiently as possible. In a Database Management System (DBMS), query optimization involves identifying the most effective way to execute a SQL statement. Since SQL is a nonprocedural language, the optimizer has the flexibility to combine, reorganize, and process data in various orders. Query optimization plays a crucial role in database design, significantly impacting the performance and efficiency of database operations.

The following are the various principles of Query Optimisation:

1. Learn how your database executes your query. Understanding how the database performs is the first step towards query optimisation. Various databases have different commands for this. To inspect the query plan in MySQL, for example, use the "EXPLAIN [SQL Query]" keyword. To inspect the query plan in Oracle, use the "EXPLAIN PLAN FOR [SQL QUERY]" command.
2. Retrieve as little data as possible Since the more information is restored from the query, the more resources the database has to extend in order to analyse and keep these entries. If you only need to retrieve one column from a table, don't use 'SELECT *'.
3. Keep intermediate results.The logic for a query might be rather complex at times. Subqueries, inline views, and UNION-style statements can be used to achieve the desired results. The transitional results for those techniques are not retained in the database, but are used directly within the query. This can cause problems, especially if the transitional results have a large number of rows.

Many factors influence query optimisation in a database management system (DBMS):

1. The number of I/O requests connected with a single file system access.
2. The amount of CPU works necessary to identify which rows satisfy the query criteria.

3. The resources needed to sort or group the data.
4. A variety of things can have an impact on query performance. The following data, cluster, and database actions all have an impact on how rapidly your queries are processed.
5. The number of compute nodes, processors, or slices - A compute node is divided into slices. More nodes imply more processors and more slices, allowing your queries to run quicker by running portions of the query concurrently across the slices. But, more nodes equal more expense, so you must find the right mix of cost and performance for your system. See Data warehouse system architecture for additional information on Amazon Redshift cluster architecture.
6. Node types -Amazon Redshift clusters can use either dense storage or dense computing nodes. Dense storage node types are recommended for large data storage requirements, whilst dense compute node types are intended for performance-intensive tasks.
7. Data Distribution: The way data is distributed across the database and indexed can have a substantial impact on query performance. A well-designed database with adequate indexes can boost query performance significantly.
8. Joins and Subqueries - Because joining numerous tables or utilising subqueries can be computationally expensive, it's crucial to keep the amount of joins and subqueries in a query to a minimum.
9. Selectivity and Cardinality - A query's selectivity and cardinality refer to how specific or selective it is, as well as the number of rows it is expected to return. Queries with a smaller cardinality will often run faster than those with a greater cardinality.
10. Server Resources - The amount of server resources available, such as RAM, CPU, and I/O, will affect query performance
11. Query Complexity - The query's complexity, such as the number of calculations and tables, will influence how much work the database has to do to execute the query.
12. Database Configuration - Database configuration, such as buffer pool settings, can have an impact on query performance.
13. Execution Plan - The execution plan is the sequence of steps that the DBMS takes to execute a query; the optimiser will generate various plans and select the best one based on specific criteria.

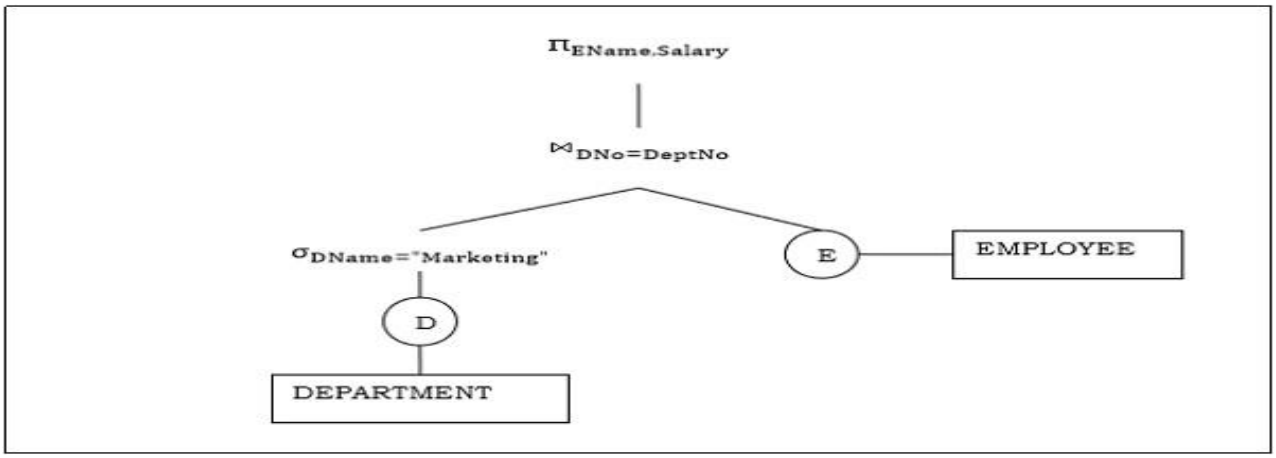
14. Statistics - The DBMS relies on statistics to determine how to execute a query, and the correctness of the information can influence the optimiser's conclusion.

You can help ensure that your database queries perform as effectively as possible by knowing and optimising all the above-mentioned factors.

10.2 CentralisedOptimisation

In a centralised system, query processing is carried out with the following goals in mind:

1. Minimisation of query response time (the time it takes to deliver results for a user's query).
2. Increase system throughput (the number of requests that are processed in a given amount of time).
3. Decrease the amount of processing memory and storage required.
4. Boost parallelism.
5.



1. **Query Parsing and Translation**

   The SQL query is initially scanned. It is then parsed to check for syntactical faults and data type correctness. If the query passes this test, it is divided into smaller query chunks. After that, each block is transformed into an equivalent relational algebra statement.

2. **Steps for Query Optimisation**

   Query optimisation consists of three steps: query tree generation, plan generation, and plan code generation.

Step 1 − Query Tree Generation

A query tree is a tree data structure that represents a relational algebra statement. The query tables are represented as leaf nodes. Internal nodes represent relational algebra operations. The root reflects the query as a whole.When an internal node's operand tables are available, it is performed. The result table is then used to replace the node. This process is repeated for all internal nodes until the root node is performed, and the result table takes its place.

Step 2 − Query Plan Generation

After generating a query tree, a query plan is developed. This query plan is an expanded version of the query tree that outlines the access paths for each operation within the tree. Access paths detail how the relational operations in the tree should be executed. For instance, a selection operation might include an access path indicating the use of a B+ tree index for selection. Additionally, a query plan specifies how intermediate tables should be transferred between operators, the use of temporary tables, and how operations should be pipelined or combined for optimal performance.

Step 3− Code Generation

The final phase in query optimisation is code generation. It is the query's executable form, which varies depending on the type of underlying operating system. The Execution Manager executes the query code and generates the results.

- **Approaches to Query Optimisation**

Exhaustive search and heuristics-based algorithms are the most commonly utilised methodologies for query optimisation.

1. **Exhaustive Search Optimisation**

   These strategies produce all feasible query plans for a query and then select the best plan. Whilst these strategies yield the optimum solution, they have an exponential time and space complexity because to the huge solution space. For instance, consider the dynamic programming technique.

2. **Heuristic Based Optimisation**

   For query optimisation, heuristic-based optimisation employs rule-based optimisation methodologies. The time and space complexity of these algorithms is polynomial, as opposed

to the exponential complexity of exhaustive search-based algorithms. These algorithms, however, do not always yield the optimum query strategy.

Some examples of typical heuristic rules are given below as follows:

a. Before joining operations, perform choose and project operations. This is accomplished by relocating the select and project operations lower in the query tree. This decreases the amount of tuples that can be joined.
b. Execute the most restrictive select/project procedures first, followed by the others.
c. Cross-product operations should be avoided since they produce very large intermediate tables.

- **Knowledge Check 1**

  Fill in the Blanks

  1. The final phase in query optimisation is _____.
  2. _____strategies produce all feasible query plans for a query and then select the best plan.
  3. _____is very important in the centraliseddatabase, and it is more important in the distributed database as join between the fragments may increase the communication time.

- **Outcome-Based Activity 1**

  Illustrate centralize Optimisation with Proper Diagram.

10.3 Ordering of Fragment Queries

Fragmentation is a feature in database servers that allows control over data storage at the table level. It enables the definition of groups of rows or index keys within a table based on a specific algorithm or scheme.

Join order is very important in the centraliseddatabase, and it is more important in the distributed database as join between the fragments may increase the communication time.

Join ordering

- Distributed INGRES
- System R*

Semi join ordering

- SDD-1

There are many assumptions that are related withthe main issues.

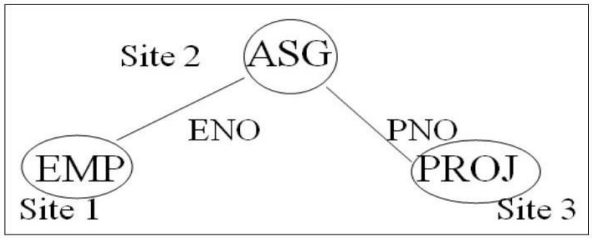
1. Relationships and fragments are indistinguishable.
2. The expense of local processing is removed.
3. Relations are transferred one set at a time.
4. The cost of transferring data to the result location to produce the final result is removed.

There are two basic approaches for order joins in the fragment queries

1. Direct optimisation of the ordering of the joins(e.g. in the distributed INGRES algorithm)
2. Replacementof the joins by the combination of the semi joins in order tominimise the communication costs.

Join Ordering - Example

Consider: $\mathbf{PROJ} \bowtie_{\mathbf{PNO}} \mathbf{ASG} \bowtie_{\mathbf{ENO}} \mathbf{EMP}$

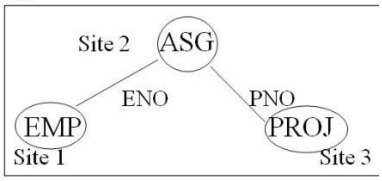


Join Ordering – Example (*cont.*)

$\mathbf{PROJ} \bowtie_{\mathbf{PNO}} \mathbf{ASG} \bowtie_{\mathbf{ENO}} \mathbf{EMP}$

- Execution alternatives:

1. EMP → Site 2
   Site 2 computes EMP'=EMP⋈ASG
   EMP' → Site 3
   Site 3 computes EMP'⋈PROJ

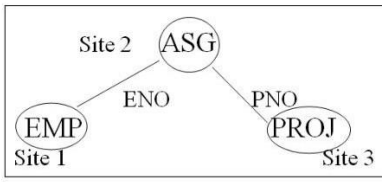


2. ASG → Site 1
   Site 1 computes EMP'=EMP⋈ASG
   EMP' → Site 3
   Site 3 computes EMP'⋈PROJ

Join Ordering – Example (*cont.*)

PROJ $\bowtie_{PNO}$ ASG $\bowtie_{ENO}$ EMP

3. ASG → Site 3
   Site 3 computes ASG'=ASG⋈PROJ
   ASG' → Site 1
   Site 1 computes ASG'⋈EMP

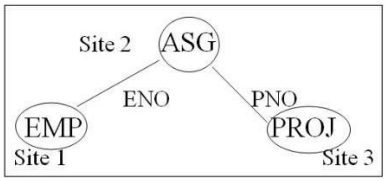


4. PROJ → Site 2
   Site 2 computes PROJ'=PROJ⋈ASG
   PROJ' → Site 1
   Site 1 computes PROJ' ⋈ EMP

Join Ordering – Example (*cont.*)

PROJ $\bowtie_{PNO}$ ASG $\bowtie_{ENO}$ EMP

5. EMP → Site 2
   PROJ → Site 2
   Site 2 computes EMP⋈ PROJ⋈ASG



Semi join Algorithms

There are many shortcomings of the joining method that are mentioned below as follows:

- Transfer of the entire relations, which may also contain some of the useless tuples.
- Semi-join also reduces the size of the operand relations that needs to be transferred.

Semi join Algorithms is highly useful if the cost of developing and sending to the other site is less than the cost of sending the entire relation.

Semijoin Algorithms (*cont.*)

- Consider the join of two relations
  - R[A] (located at site 1)
  - S[A] (located at site 2)

- Alternatives
  1. Do the join $R \bowtie_A S$
  2. Perform one of the semijoin equivalents

$$R \rhd\lhd_A S \Leftrightarrow (R \rhd\!\!<_A S) \rhd\lhd_A S \Leftrightarrow R \rhd\lhd_A (S \rhd\!\!<_A R)$$
$$\Leftrightarrow (R \rhd\!\!<_A S) \rhd\lhd_A (S \rhd\!\!<_A R)$$

Semijoin Algorithms (*cont.*)

- Perform the join
  - Send R to site 2
  - Site 2 computes $R \bowtie_A S$
- Consider semijoin $(R \rhd\!\!<_A S) \rhd\lhd_A S$
  - S' = Π_A(S)
  - S' → Site 1
  - Site 1 computes $R' = R \rhd\!\!<_A S'$
  - R' → Site 2
  - Site 2 computes $R' \rhd\lhd_A S$

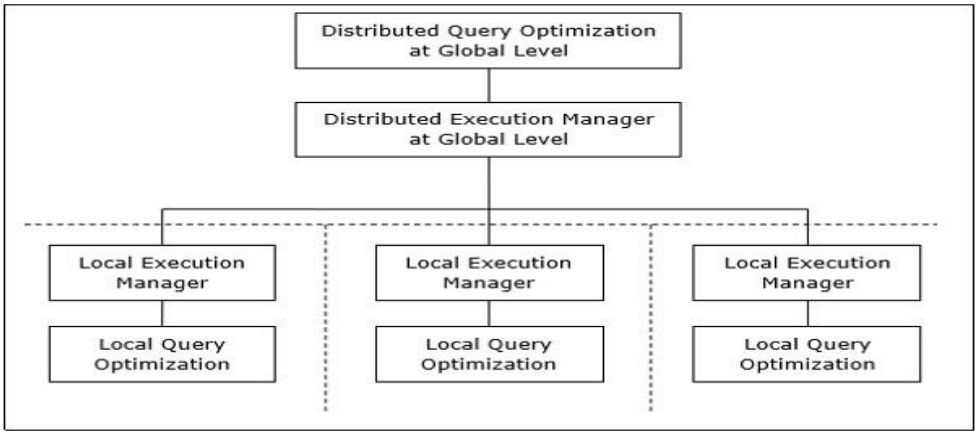
Semijoin is better if $(size(\Pi_A(S)) + size(R \rhd\!\!<_A S)) < size(R)$

10.4 Distributed Query Optimisation Algorithms

Distributed query optimization refers to the process of devising a plan for executing a query in a distributed database system. This strategy is known as a query execution plan. In a distributed database system, schemas and queries represent logical data units. For instance, in a relational distributed database system, relations are logical units of data. At the fundamental physical level, these units can be fragmented. In a distributed system, these fragments, which may be redundant and duplicated, are allocated to various database servers.

Distributed query optimization requires evaluating numerous query trees, each capable of producing the desired query results. This complexity arises from the extensive presence of

replicated and fragmented data. Therefore, the objective is to identify an optimal solution ratherthan the absolute best one.



The primary concerns for distributed query optimisation are as follows:

1. The most efficient use of distributed system resources.
2. Consider trading.
3. The query's solution space is reduced.

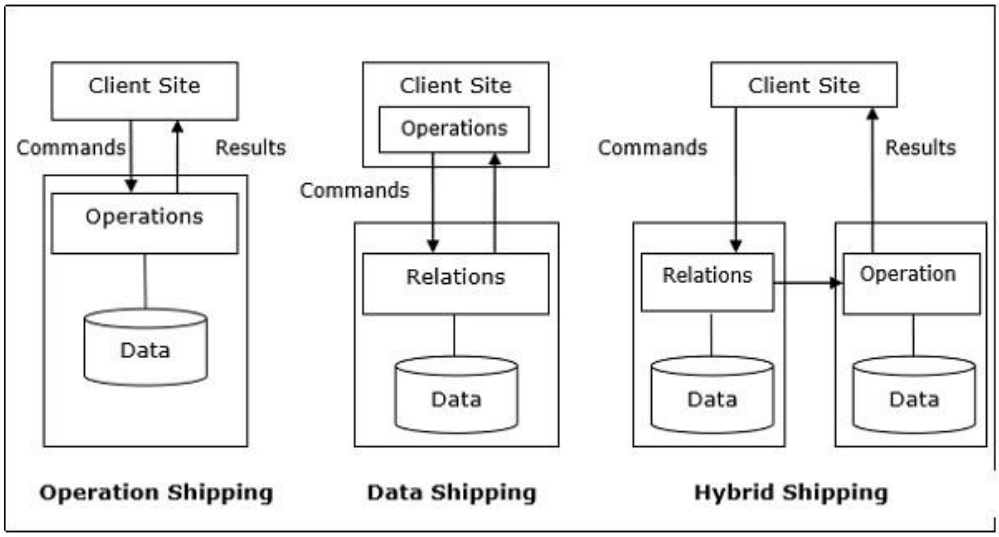
- **Optimum Resource Usage in a Distributed System**

A distributed system uses several database servers spread over several locations to handle query-related tasks.

The methodologies for optimal resource usage are as follows:

1. **Operation Shipping-** Rather than at the client site, the process is performed at the location where the data is kept. After then, the client's website receives the findings. For operations when both operands are present at the same place, this is helpful. Two instances are Select and Project operations.
2. **Data Shipping-**The database server receives the data pieces and executes the actions on them. This is applied to operations in which the operands are dispersed among several places.This is also suitable for systems with low connection costs and processors local to the client-server network that operate much more slowly.

3. **Hybrid Shipping**-This combines operation shipping with data. High-speed processors receive the data chunks and execute the operation there. After then, the client's website receives the findings.



- **Knowledge Check 2**

  State True and False for the Following Sentences.

  1. The process of creating a plan for the execution of a query to a distributed database system is referred to as distributed query optimisation.
  2. In hybrid Shipping, data pieces are transmitted to the database server, where the operations are performed.
  3. Join order is very important in the centraliseddatabase, and it is more important in the distributed database as join between the fragments may decreases the communication time.

- **Outcome-Based Activity 2**

  List out some of the examples of Join and Semi-join algorithms**.**

10.5 Summary

- Query optimization is the process of figuring out how to run a SQL statement as efficiently as possible. The optimiser can combine, restructure, and process data in any sequence as SQL is a nonprocedural language.

- The process of query optimization involves three stages: the creation of query trees, plans, and plan codes.
- The following objective guides query processing in a centralized system: Shorten the time it takes to respond to a user's inquiry in terms of response time. Boost the system's throughput, or the quantity of requests handled in a predetermined length of time.
- Join ordering is very important in the centraliseddatabase, and it is more important in the distributed database as join between the fragments may increase the communication time.
- By examining different regions of the combinatorial search space, which is defined as the set of possible query execution plans, the distributed query optimization algorithm selects an optimum or satisfied plan. The objective function is the cost function that has to be optimized.

10.6 Self-Assessment Questions

1. What is query optimisation?
2. What are the factors that governs query optimisation?
3. What is Centralised query optimisation?
4. What is the ordering of fragment queries?
5. What are the different approaches of query optimisation in a Centralised System?
6. What are the different steps of Query optimisation in a centralisedSystem?
7. Explain Query Tree generation.
8. Explain Distributed Query optimisation algorithms.

10.7 References

- Atkinson, M.P. (1981) Database. Maidenhead: Pergamum InfoTech.
- Frank, L. and Helmersen, O. (1988) Database theory and practice. Wokingham, England: Addison-Wesley Publishing Company.
- Database users: 2nd Toronto conference: Selected papers (1990). Canadian Association for Information Science.
- Database (no date). Weston Ct.: Online, Inc.
- Kambayashi, Y. (1982) Database. Rockville, MD: Computer Science Press.