DATABASE MANAGEMENT SYSTEM

Second Year : B.A / B.Com.

SEMESTER – IV

Lesson Writers

Sri. Y. Suresh Babu
M.Com, M.C.A.,
Lecturer,
Dept. Of Computer Science,
JKC College, Guntur.

Dr.Vasantha Rudramalla
M.Tech.,Ph.D.
Faculty,
Dept. of Computer Science & Eng.
Acharya Nagarjuna University.

Editor Editor

Prof. I. Ramesh Babu
M.E., Ph.D.
Dept. of Computer Science,
Acharya Nagarjuna University,
Nagarjuna Nagar, Guntur.

Director

Dr. Nagaraju Battu
MBA, MHRM, LLM, M.Sc. (Psy.), MA (Soc.), M.Ed., M.Phil. , Ph.D.
Centre for Distance Education
Achaarya Nagarjuna University
Nagarjuna Nagar 522 510

Ph : 0863 - 2346208, 2346222, 2346259 (Study Material)
Website : www.anucde.info
e-mail : anucdedirector@gmail.com

2nd Year B.A / B.Com. Semester – IV

DATABASE MANAGEMENT SYSTEM

First Edition : 2023

No. of Copies :

Published by :

Dr. NAGARAJU BATTU,
Director
Centre for Distance Education,
Acharya Nagarjuna University

Printed at :

FOREWORD

Since its establishment in 1976, Acharya Nagarjuna University has been forging ahead in the path of progress and dynamism, offering a variety of courses and research contributions. I am extremely happy that by gaining a ‘A’ Grade from the NAAC in the year 2014, the Acharya Nagarjuna University is offering educational opportunities at the UG, PG levels apart from research degrees to students from over 285 affiliated colleges spread over the two districts of Guntur and Prakasam.

The University has also started the Centre for Distance Education with the aim to bring higher education within reach of all. The centre will be a great help to those who cannot join in colleges, those who cannot afford the exorbitant fees as regular students, and even housewives desirous of pursuing higher studies. With the goal of bringing education in the door step of all such people. Acharya Nagarjuna University has started offering B.A, and B, Com courses at the Degree level and M.A, M.Com., L.L.M., courses at the PG level from the academic year 2021-22 on the basis of Semester system.

To facilitate easier understanding by students studying through the distance mode, these self-instruction materials have been prepared by eminent and experienced teachers. The lessons have been drafted with great care and expertise in the stipulated time by these teachers. Constructive ideas and scholarly suggestions are welcome from students and teachers invited respectively. Such ideas will be incorporated for the greater efficacy of this distance mode of education. For clarification of doubts and feedback, weekly classes and contact classes will be arranged at the UG and PG levels respectively.

It is aim that students getting higher education through the Centre for Distance Education should improve their qualification, have better employment opportunities and in turn facilitate the country’s progress. It is my fond desire that in the years to come, the Centre for Distance Education will go from strength to strength in the form of new courses and by catering to larger number of people. My congratulations to all the Directors, Coordinators, Editors and Lesson -writers of the Centre who have helped in these endeavours.

Prof. P.Rajasekhar
Vice –Chancellor,
Acharya Nagarjuna University

ACHARYA NAGARJUNA UNIVERSITY-GUNTUR

Structure of B.Com (Computer Applications) Programme under Revised CBCS

Semester-wise Syllabus under CBCS (w.e.f. 2020-21 Admitted Batch)

I Year B Com (CA), Semester- IV

407BCO21-COURSE 4C: Database Management System

………………………………………………………………………………………………..

Unit-I (Overview of Database Management System)

Introduction, Data and Information, Database, Database Management System, Objectives of DBMS, Evolution of Database Management System, Classification of Database Management System.

UNIT-II(File-Based System)

File Based System. Drawbacks of File-Based System, DBMS Approach, Advantage of DBMS, Data Models, Components of Database System, Database Architecture, DBMS Vendors and their products.

UNIT-III (Entity-Relationship Model)

Introduction, The Building Blocks of an Entity-Relationship, Classification of Entity Set, Attribute Classification, Relationship Degree, Relationship Classification, Generalization and Specialization, Aggregation and Composition, CODD's Rules, Relational Data Model, Concept of Relational Integrity.

UNIT-IV (Structured Query Language)

Introduction, History of SQL Standards, Commands in SQL, Data types in SQL, Data Definition Language (DDL),Selection Operation Projection Operation, Aggregate Functions, Data Manipulation Language, Table Modification, Table Truncation, Imposition of Constraints, Set Operations.

UNIT-V (PL/SQL)

Introduction, Structure of PL/SQL,PL/SQL Language Elements, Data Types, Control Structure, Steps to Create a PL/SQL Program, Iterative Control Cursors, Steps to Create a Cursor, Procedure, Functions, Packages, Exceptions Handling, Database Triggers, Types of triggers.

LIST OF THE DBMS LAB PROGRAMS

1. Create tables department and employee with required constraints.
2. Initially only the few columns (essential) are to be added. Add the remaining columnsseparately by using appropriate SQL command.
3. Basic column should not be null
4. Add constraint that basic should not be less than 5000.
5. Calculate hra, da, gross and net by using PL/SQL program.
6. The percentage of hra and da are to be stored separately.
7. When the da becomes more than 100%, a message has to be generated and with userpermission da has to be merged with basic.
8. Empno should be unique and has to be generated automatically.

MODEL QUESTION PAPER (407BCO21)

B. Com.(Comp. Appl.s) Degree Examination

Second Year – Fourth Semester

Part – II : Commerce

Paper – IV : DATA BASE MANAGEMENT SYSTEM

Time : Three hours **Maximum Marks : 70**

Section – A

Answer any FIVE of the following questions. (5 × 4 = 20 Marks)

1) Explain Data and information, DBMS.
డేటా మరియు ఇన్ఫర్మేషన్, DBMS గురించి వివరించండి.

2) Write about File Based system.
ఫైల్ బేస్డ్ సిస్టం గురించి వ్రాయండి.

3) Explain the concept of relational integrity.
రిలేషనల్ ఇంటెగ్రిటీ (సమగ్రత) భావనను వివరించండి.

4) Explain Data types in SQL.
SQL లోని డేటా టైప్స్ ను వివరించండి.

5) Explain structures of PL/SQL.
PL/SQL యొక్క నిర్మాణంను వివరించండి.

6) Write the advantages of DBMS.
DBMS యొక్క ప్రయోజనాలను వ్రాయండి.

7) Explain relationship classification.
రిలేషన్ షిప్ వర్గీకరణను వివరించండి.

8) Explain Aggregate functions in SQL.
SQL లోని Aggregate ఫంక్షన్లను వివరించండి.

Section – B

Answer the following questions. (5 x 10 = 50 Marks)

9) (a) Explain about objectives of DBMS.
DBMS యొక్క ఆబ్జెక్టివ్స్ ను వివరించండి.

Or

(b) Explain classification of database.
డేటాబేస్ యొక్క వర్గీకరణను వివరించండి.

10) (a) Explain components of Database system.
డేటాబేస్ సిస్టమ్ యొక్క components ను వివరించండి.

Or

(b) Give a brief note on Data Models.
డేటా మోడల్స్ గురించి క్లుప్తంగా వ్రాయండి.

11) (a) Explain building blocks of an entity relationship with suitable example.
Entity రిలేషన్ షిప్ ను వాటి యొక్క బిల్డింగ్ బ్లాక్స్ ను సరియైన ఉదాహరణతో వివరించండి.

Or

(b) Explain CODD's rules.
CODD's రూల్స్ ను వివరించండి.

12) (a) Explain selection and projection operation with example.
ఉదాహరణతో selection మరియు ప్రొజెక్షన్ ఆపరేషన్స్ ను వివరించండి.

Or

(b) Explain DDL with commands.
DDL ను వాటి యొక్క కమాండ్స్ తో వివరించండి.

13) (a) Explain control structures in PL/SQL.
PL/SQL యొక్క కంట్రోల్ నిర్మాణంను వివరించండి.

Or

(b) What is database trigger? Explain types of triggers.
డేటా ట్రిగ్గర్ అనగానేమి? వాటి యొక్క రకాలను వివరించండి. .

CONTENTS

LESSON – 1

OVER VIEW OF DATABASE MANAGEMENT SYSTEMS

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ Understand the concept of the database system
- ❖ Discuss the advantages and disadvantages of DBMS over a file system
- ❖ Explain the architecture of DBMS
- ❖ Define entities, entity sets, attributes and keys
- ❖ Describe the Entity-Relationship model in detail
- ❖ Identify the various kinds of relationships that exist between entities
- ❖ Evaluate the various design choices for ER diagrams
- ❖ Explain different data models

STRUCTURE OF THE LESSON:

1.1 INTRODUCTION:

You know that the Database System is a computerized record-keeping system that allows you to electronically organize and manipulate data using computers. It consists of databases and a Database Management System (DBMS). A database stores interrelated data. It is a software program that helps to store and manipulate data in a database system. This data can be data of a software system such as employee information system and student information system, used in the organization.

The basic DBMS concepts like entities, entity sets and attributes will be explained in this unit. Entities are the collection of real world objects in a database. It is a collection or set of entities. The attributes in DBMS define the characteristics and properties of an entity.

In this you will also learn about the important concept of keys, which include the primary key and candidate key. The primary key uniquely identifies the tuple in a table stored in the database. The candidate keys are keys that can become primary keys, but are not explicitly defined as primary keys. In DBMS, the Entity-Relationship (ER) model is also an important concept. The ER model is the conceptual model for representing any database application.

1.2 OVERVIEW OF DATABASE SYSTEM:

A database system allows the users to store and manipulate data by performing various operations such as updating and deleting according to the requirements. The data stored in the database system can be of any information, which is required for running the business of an organization. For example, an organization can maintain the employee database that stores the personal information of the employees such as name, address, birth date, date of joining and telephone numbers using some DBMS software. Figure below shows a simple database system.
Introduction to Database Management System

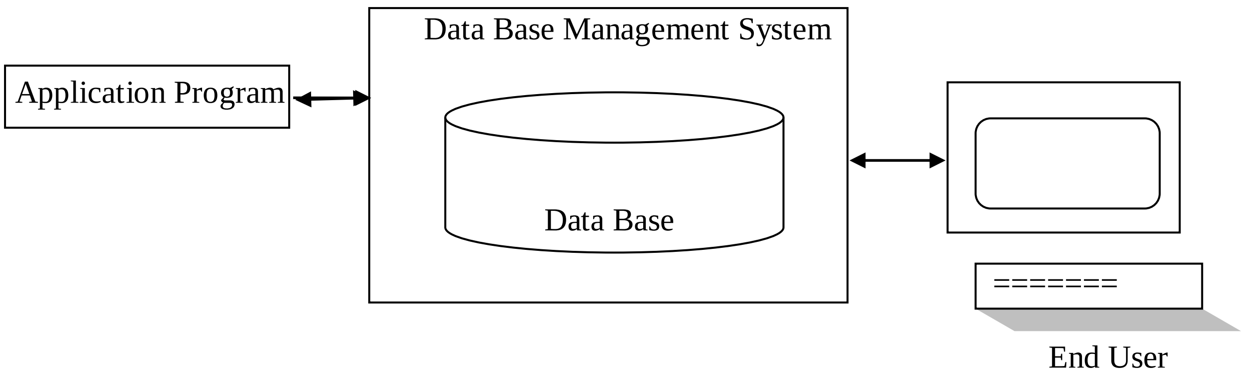


Figure: The Database System

A database system consists of the following four components:

1. Data
2. Hardware
3. Software
4. Users

1.2.1 Data:

It refers to the different types of information used in the organization that can be stored in the database. Data can be both- integrated and shared, and the different types of data can be stored in the database in the integrated manner. For example, a database contains the data related to the employees of an organization in one file and the department to which that employee belongs in another file. The employee file can include data such as name, address, designation, e-mail id and salary of the employee. The department file can include data such as department number and department name. Thus, there is distinct data included in both the files, but the files are stored in a single database. This is known as the integration of data.

The data stored in the database is shared between the users and each user can have access to the data. For example, the human resource department and the accounts department can access the data stored in the employee file of the database.

1.2.2 Hardware:

The hardware component of a database system includes the secondary storage devices such as magnetic disk, processor and main memory. The secondary storage device contains the data that needs to be stored. The processor and the main memory help in the execution of the DBMS software. The hardware component of the database system includes I/O devices, I/O device controllers and I/O channels that help in performing the I/O operations for transferring the data.

1.2.3 Software:

The software component or the database system helps the end users to access and use the data stored on a computer. The software component is also called the database manager or database server. The man component of the database system is basically the DBMS software, which allows you to add or remove files from a database and update the data contained in these files. Besides DBMS, the software component also includes other software such as application development tools, software for designing the user interface of database application and transaction manager software.

1.2.4 Users:

In a database system, users perform different operations such as storing, retrieving, updating and deleting data. The users are people who need to interact with databases. Users can be categorized according to their requirements of data. There are three types of users- application programmers, end users and database administrators who perform different operations in a database system. Application programmers write database applications using a programming language like .Net or Java. These database applications contain the code for performing operations such as accessing, inserting, updating and deleting data from a database using a database language like SQL or Functional Database Language (FDL). End users interact with the database system using a computer and a database application created by the application programmer. The end users can also use an interface provided by a DBMS to interact with the database system. Database administrators administer the database system using a database language and the DBMS software. A database administrator is responsible for making the strategy and policy decisions regarding the organization of data in the database. A DBA also provides technical support in implementing the decisions which are taken by the data administrator. A database administrator performs the following functions:

- Defining the conceptual schema

- Defining the internal schema
- Liaising with users
- Defining security and integrity constraints
- Defining dump and reload policies
- Monitoring performance and responding to changing requirements

Besides the data administrator, there can be database designers and database managers who may be involved in the use of database systems. A database designer identifies the data to be stored in a database. The designer is also responsible for choosing the right database structure to represent and store this data in the database. The tasks of identifying the data and choosing the structure are performed before the implementation of a database. The database designer communicates with the database users to understand their requirements. Database manager, also called DBMS, refers to the software that helps use and manage the data stored in a database. The database manager handles the requests of database users to access the data items from database. The database manager also provides facilities such as support for a query language, to retrieve and update the database. The facilities provided by the database manager depend on the design of the database manager. For example, if the data manager is designed to handle one request at a time, then multiple users cannot access data simultaneously.

1.2.5 Need for Database Systems:

A database system allows a user to create a database which stores data in an organized manner in the form of tables. Before the introduction of databases, large files were used to store data on the hard drives of computer systems. This method of storing data in files was called manipulating or handling file systems. In the file system, each user defines and implements the files needed for a specific application as a part of programming the application. The basic disadvantage of a file system is that each user has to maintain a separate copy of the same file. For example, in the main office of a company, the program to calculate the salary of employees may use the employee data file. The billing department also keeps the same information in a separate file to keep the track of employees' conveyance bills, etc. Although, both the users are interested in the employees' data, each of them has to maintain separate files of records. Instead, you can create a database for storing data related to employees of an organization that can be accessed by the main office as well as the billing department for retrieving the required data. In addition, the process of electronically storing data in a database is much faster than storing data in a file system. The process of storing data in a file is repetitive and tedious as compared to storing data in a database. Thus, organizations prefer to store data in a database rather than in files. A database system is useful for business organizations when data has to be manipulated. Manipulation of data involves the tasks of retrieving, inserting, modifying and deleting unnecessary data from a database. With the help of a database system, these tasks can be performed easily and quickly.

1.3 DESCRIBING A DATABASE:

A database consists of persistent data that can be of different types and is used in the organization. The persistent data refers to the data that is permanently stored in the database and is deleted by an explicit request through the database language that is sent to the DBMS. A database contains tables that stores related data. For example, the data related to an employee is stored in the Employee table. You can query a database to retrieve the data from it using the database language.

1.3.1 Tables in a Database:

The table's stores the data related to an entity such as employee and student that is a part of the software system used in the organization. For example, consider an employee information software system that stores the data related to the employees, department and their salary. The organization can create tables in the database such as employee_records for storing all the data related to the employee information software system. The employee table can store the personal data such as name and address related to the employee. The department table can store the data such as department number and department name related to the departments in an organization. The salary table contains data such as basic salary and House Rent Allowance (HRA) related to the salary of an employee. Figure below shows the employee table, with the data related to the employees of an organization.

Employee Code	**Employee Name**	**Employee Address**	**Employee Designation**	**Employee Salary**	**Employee Department**
001	Deepak	23,Ganga Apartments, Geeta Colony	Reseaercher	20000	10
002	Shalini	45, Hemagiri Apartments, Shri Colony	Programmer	18000	20
003	Sachin	67,DLF Qutab, Gurgoan	Accountant	17000	30

Figure: The Employee Table

1.3.2 Columns in a Database:

Columns in the database represent different characteristics such as name and address of an entity. For example, the employee entity in an employee information software system can contain the characteristics such as name, address, e-mail id, salary and department. Thus, the column in the employee table, which stores the data related to employee entity, is as follows:

- Employee code: It contains numbers, which uniquely identify the employees of an organization.
- Employee name: It contains the names of all the employees of an organization.
- Employee address: It contains the addresses of all the employees of an organization.
- Employee e-mail id: It contains the e-mail addresses of all the employees of an organization.
- Employee salary: It contains the salaries of all the employees of an organization.
- Employee department: It contains the department number, which uniquely identifies the departments to which the employees belong.

Similarly, the department table, which stores the data related to the department entity, can contain two columns- Department number and Department name. The Department number column contains numbers, which uniquely identifies the departments of an organization and the Department name column contains the name of the departments of an organization.

The salary table, which stores the data related to the salary of employee entity, can contain the following columns:

- **Basic salary:** It contains the basic salary of all the employees of an organization.
- **HRA:** It contains data related to HRA given to all the employees of an organization.
- **PF:** It contains the data related to the Provident Fund deducted for all the employees of an organization.
- **TA:** It contains the data related to the Travel Allowance (TA) given to all the employees of an organization.

1.4 OBJECTIVES OF DBMS:

DBMS is a set of programs that allows you to create and maintain databases. The advantages of the database management system are:

- It controls data redundancy.
- It restricts unauthorized access.
- It provides persistent storage for the program objects and data structures.
- It provides multiple user interfaces.
- It provides integrity constraints.
- It provides backup and recovery.

1.4.1 Data redundancy:

Database management system also helps to control data redundancy. For example, an organization stores the information about the employees. The accounts department needs to store employee information regarding pay roll and query for the leaves taken by the employee. Similarly, the HR department needs to store the employee information for querying about the leaves and application forms for any company-specific issue and performance related updates. In this way, the accounts and HR department both store the information about the employees. In this process both the departments store same information about the employees. Storing the same information more than once is known as data redundancy. Data redundancy has some disadvantages, which are stated below:

- You need to perform a logical update such as entering the data for a new employee in each of the files maintained by various user groups.
- The storage space is wasted when the same data is stored repeatedly.
- The files representing the same data may become inconsistent if each of the user group updates the files independently.

In the database approach of storing data, the information such as employee name or date of joining is stored in a single place. Therefore, storing data in a database prevents the data redundancy and saves the storage space.

1.4.2 Unauthorized access:

Multiple users can access the data stored in the database, however all users are not allowed to access all the data in the database. Only authorized users can access the data, to which access permissions are granted. For example, for the financial data stored in a database, only a few users are allowed to access the financial information.

In addition for the security reasons you can also specify the users who are allowed to retrieve information and the users who are allowed to update the information in the financial database of the organization.

1.4.3 Persistent storage:

Databases provide constant storage for the data structures such as class definitions in C++ and program variables used in the programming languages. If the program variable is a data structure which is used in the program is not stored in the permanent memory file, then the value of the program variable is left behind when the program terminates. The database management system is compatible with languages such as .NET and Java. Therefore, data structures can be stored permanently in the database management system. Such data stored in the database management system is called persistent data because the data can be called by another program even after the program is terminated.

1.4.4 Multiple user interfaces:

The database management system provides multiple user interfaces for users with varying levels of technical knowledge. For example, query language for casual users and programming language interfaces for the application programmers.

1.4.5 Integrity constraints:

When you store data in the database, you can specify an integrity restriction that holds for the data. DBMS allows you to define and implement the constraints. For example, you can specify a constraint that each value in the employee_ id column must have a unique value. The database designers need to identify the integrity constraints during the database design.

1.4.6 Backup and recovery:

Data stored in a database can be recovered in case of hardware or software failures. The backup and recovery subsystem of the database management system allows in taking backup of whole data when database is successfully working and recovering of data when hardware or software failure occurs. For example, if hardware failure occurs when the database is being updated, then the data stored in the database is restored and the program is resumed where it was interrupted.

Disadvantages of DBMS:

Database provides a centralized storage of data. The users need to access the data from different locations. A database provides the facility of online access to the authorized users. The accessibility of database by a huge number of users involves the risk of data manipulation and becomes the disadvantage of the database management system. A database management system is vulnerable in the following areas:

❖ **Data integrity:** A large number of users can access the database through the Internet. It becomes difficult to maintain the integrity of data with increase in the volume of users of the database. Data integrity becomes vulnerable when multiple users try to update the data at the same time. Data quality: As the data is accessible by the remote users, it increases the chances of reduced data quality. The remote users can change, manipulate or damage the data. Adequate controls are needed to secure the data from manipulation.

- **Data security:** In a centralized database, the data is available to remote users. It increases the chances of data abuse or data theft. To reduce the chances of unauthorized users accessing the important information, it becomes necessary to take administrative and technical measures.
- **Enterprise vulnerability:** The centralization of all the enterprise information in the database makes the database an indispensable resource for the organization. The security of central database becomes a cumbersome task for the organization, as the survival of the organization may depend on the security of the database.
- **The cost of using a DBMS:** DBMS provides a flexible approach of data access to multiple users from different locations. At the same time, the investment in designing and maintaining the database makes it a costly affair for small organizations.

1.5 SUMMARY:

In this lesson, you have learnt about the database system that allows users to store and manipulate data by performing various operations such as updating and deleting. The various components of a database system are data, hardware, software and users. The basic concepts related to data models have also been discussed in this unit. A data model helps in describing the structure of a database with regard to the data types used, constraints applied and relationships between various entity sets. You have also studied the relationship types such as one-to- one and one-to-many between two or more entity sets. You have also learnt about the different types of data models, which include relational, hierarchical, network and object oriented data models. The relational data model is a model in which data is stored in tables, which contain rows and columns. The hierarchical data model stores data in the form of a tree structure. The network data model is that model in which data is stored in the form of set constructs. Each set in the set construct consists of owner and member of record type and name.

1.6 KEY TERMS:

- **Database System:** Database System is a computerized record keeping system that allows you to electronically organize and manipulate data using computers.
- **Data:** It is the information related to a software system used in an organization that can be stored in a database and has a meaning.
- **Hardware:** The hardware component of a database system includes the secondary storage devices, such as magnetic disk, processor and main memory. The secondary storage device contains data, which needs to be stored.
- **Software:** The software component of a database system helps the end users to access and use the data stored on a computer. The software component is also called the database manager or database server. Table: The tables store data related to an entity such as employee and student that is a part of the software system used in an organization. Column: Columns in a database represent the different characteristics such as name and address of an entity.
- **Database Management System (DBMS):** Database Management System is a software solution that allows you to create and maintain databases in which you can store the data.
- **Internal level architecture:** The internal level of the database specifies the way in which the data is physically stored in a database.
- **External level architecture:** The external level of the database specifies the way in which the data stored in a database and is viewed by the users. Conceptual level

architecture: The conceptual level specifies the level of interaction between the internal level and the external level of system architecture.

- ❖ **System Architecture of DBMS:** System Architecture of the DBMS describes what data is stored, how the data is stored and who has access to it.
- ❖ **Attribute:** An attribute specifies the particular properties of the entity.
- ❖ **Degree of relationship set:** It is the number of entity sets that participate in a relationship.
- ❖ **Domain of the attribute:** The set of possible values that an attribute can possess is called the domain of the attribute.
- ❖ **Data model:** A data model in DBMS describes the structure of a database. The structure of a database refers to the data types, constraints and relationships used on the data.
- ❖ **Relational data model:** In a relational data model the data is stored in tables, which are also called as relations.
- ❖ **Object-oriented data model:** The object-oriented (OO) data model consists of a collection of entities. Entity refers to the concept or object described in a database and is represented as a class in the object-oriented data model.

1.7 SELF – ASSESSMENT QUESTIONS:

1. Give an overview of a database system.
2. Give a brief description of tables and queries in DBMS.
3. What are the features of DBMS?
4. What are the advantages and disadvantages of DBMS?
5. What do you understand by the DBMS? Also, explain its different features, advantages and disadvantages
6. What are the various roles performed in the DBMS?

1.8 FURTHER READINGS:

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LESSON - 2

EVOLUTION OF DATABASE MANAGEMENT SYSTEM

OBJECTIVES:

After going through this lesson, you will be able to:

❖ The evolution of Database Management Systems (DBMS) .

❖ It involves understanding the historical development and progression of database technologies. This can encompass various aspects

❖ This gain a comprehensive understanding of the field's history, the challenges faced and overcome, and the trends shaping the future of data management.

STRUCTURE OF THE LESSON:

2.1 INTRODUCTION:

A database is essentially a structured collection of data, organized in a way that facilitates efficient storage, retrieval, and manipulation of information. In a database, data is typically organized into tables, which consist of rows (records) and columns (attributes). These tables can be interlinked through relationships, allowing for more complex and sophisticated data management.

2.2 DATABASE MODELS:

2.2.1 Flat files (1960s – 1980s) :

Flat file database is a database that stores information in a single file or table. This type of database is simple and easy to manage, but it may not be as efficient or powerful as more complex database systems for handling complex relationships or performing advanced queries.

Advantages:

- ❖ A flat file database can be a suitable choice for small databases
- ❖ It is easy to understand and implement. Fewer skills are required to handle a flat file database.
- ❖ Less hardware and software skills are required to maintain a flat file database.

Disadvantages:

- ❖ A flat file can contain fields that duplicate data because there is no automation in flat files.
- ❖ When a record needs to be deleted from the flat file database, all relevant information in the various fields must be removed manually, making data manipulation inefficient.
- ❖ A flat file database wastes space on your computer because it has to store information about items that are not logically available. Searching for information in a large database takes a long time.

2.3 IMPLEMENTATION OF A FLAT FILE DATABASE:

Flat file database is implemented in:

- ➢ Berkeley DB
- ➢ SQLite
- ➢ Mimesis
- ➢ The Integration Engineer etc.
- ➢ Hierarchical database (1970s – 1990s)

As the name suggests, a hierarchical database contains data in the form of hierarchically organized data. In more detail, you can think of it as a family tree in which a parent-child relationship exists. Each parent can have multiple children, but a child can only have one parent, which is a one-to-many relationship. Its hierarchical structure contains levels or segments that correspond to the file system record type. All attributes of a particular record are listed under the entity type.

- ❖ In a hierarchical database, the entity type is the main table, the rows in the table represent records, and the columns represent attributes.
- ❖ In this, CUSTOMER is the parent and it has two children (CHCKACCT & SAVEACCT).
- ❖ In hierarchical database, the entity type is the main table, rows of a table represent the records and columns represent the attributes.
- ❖ In this CUSTOMER is the parent and it has two children (CHCKACCT & SAVEACCT).

Advantages:

- ❖ In a hierarchical database, access to information is quick thanks to predefined paths. This increases database performance.
- ❖ The connections between the different entities are easy to understand.

Disadvantages:

- ❖ The hierarchical database model lacks flexibility. If you need to establish a new relationship between two entities, you will need to build a new, possibly redundant, database structure.
- ❖ Data maintenance is inefficient in a hierarchical model. Changes to the mappings may require manual reorganization of the data.
- ❖ This model is also inefficient for non-hierarchical access.

2.4 NETWORK DATABASE (1970S – 1990S) :

The creator of the network model is Charles Bachmann. In contrast to the hierarchical database model, a network database allows multiple parent-child relationships, i.e. H. it maintains a many-to-many relationship. A network database is essentially a graphical structure. The network database model was created to achieve three main goals:

- ❖ Represent complex data relationships more effectively.
- ❖ To improve database performance.
- ❖ To implement the database standard.

In a network database, a relationship is called a set. Each set consists of two types of records: an owner record, which corresponds to the parent type in a hierarchical database model, and a member record, which is similar to a child record in a hierarchical database model.

Advantages:

- ❖ The network database model makes data access quite easy and efficient as the application can access the owner record and all member records as a whole.
- ❖ This model is conceptually simple to design.
- ❖ This model ensures data integrity as no member can exist without an owner.
- ❖ The user must then enter the owner entry and then save the member.
- ❖ The network model also ensures data independence because the application runs independently of the data.

Disadvantages:

- ❖ Models lack structural independence and cannot make changes to the database structure. The application program must also be modified before accessing the data.
- ❖ The network model does not allow for easy-to-use database management systems.

Implementation of network database:

Network database is implemented in:

- ➢ Digital Equipment Corporation DBMS-10
- ➢ Digital Equipment Corporation DBMS-20
- ➢ RDM Embedded

- Turbo IMAGE
- Univac DMS-1100 etc.

2.4 RELATIONAL DATABASE (1980S – PRESENT):

The relational database model was developed by E.F. suggested Morue. After the hierarchy and network models, the birth of this model was a huge advancement. Allows entities to be linked using a common attribute. So, to connect two tables (elements), it is enough that they have a common attribute. Tables contain primary keys and alternative keys. Primary keys form a relationship with alternative keys. This feature makes this model extremely flexible.

This allows you to store large amounts of information in small tables when using a relational database. Data access is also very efficient. All the user has to do is enter a query and the application will provide the requested information.

Relational databases are created using the computer language Structured Query Language (SQL). This language forms the basis of all database applications available today, from Access to Oracle.

Advantages:

- A relational database supports a range of mathematical operations such as union, intersection, difference and Cartesian product. It also supports select, project, relational merge and split operations.
- The relational database uses a normalization framework that makes it easier to achieve data independence.
- Security controls can also be implemented more effectively by performing authorization checks on sensitive attributes in the table.
- The relational database uses a simple, human-readable language.

Disadvantages:

- The response to a query becomes time-consuming and inefficient if the number of tables between which the relationships are established increases.

Implementation of Relational Database:

- Oracle
- Microsoft
- IBM
- My SQL
- PostgreSQL
- SQLite

2.5 OBJECT-ORIENTED DATABASE (1990S – PRESENT):

An object-oriented database management system is a database system in which data or information is represented in the form of objects, similar to an object-oriented programming language. In addition, an object-oriented database management system also facilitates transaction management, language for various queries, and indexing options. Additionally, these database systems are capable of processing data efficiently across multiple servers.

In contrast to relational databases, an object-oriented database works within real programming languages such as JAVA or C++.

Advantages:

- ❖ When complex (many-to-many) relationships exist between entities, an object database handles them much faster than any database model discussed above.
- ❖ Navigating the data is much easier.
- ❖ Objects require no assembly or disassembly, saving programming and execution time.

Disadvantages:

- ❖ Lower performance when data or relationships are simple.
- ❖ Data is accessed through a specific language and API, which is not the case with relational databases.

2.6 OBJECT-RELATIONAL DATABASE (1990S – PRESENT):

In simple terms, an object-oriented relational database management system has a changed object-oriented user view compared to an already implemented relational database management system. When various programs interact with this modified database management system, they generally assume that data is stored as objects.

The basic principle of this database management system is that it translates useful data into organized tables arranged in rows and columns and now manages the data in the same way as a relational database system. When the user needs to access the data, it is also brought back from the processed form to the composite form.

Advantages:

- ❖ Data remains encapsulated in an object-relational database. The terms inheritance and polymorphism can also be implemented in this database.

Disadvantages:

- ❖ The object-relational database is complex. Proponents of the relational approach believe that the simplicity and purity of the relational model has been lost.
- ❖ It's also expensive.

2.7 WEB-ENABLED DATABASE (1990S TO PRESENT):

The Web-Enabled Database simply populates a database with a web interface.

This means that there can be a separation of interests. This means the web designer does not need to know the details of the underlying database design. Likewise, the database designer must deal with the web interface of the database.

An online database uses three functional layers: a presentation layer, an intermediate layer, and a database layer.

Advantages:

- ❖ An online database allows users to retrieve the information they need from a central repository on demand.
- ❖ The database is simple and user-friendly.
- ❖ Accessing the data is easy thanks to the online database.

Disadvantages:

- ❖ The main disadvantage is that it can be easily hacked. Internet-enabled databases support the full range of database operations, but must be “downloaded” for ease of use.

2.9 CLASSIFICATION OF DATABASE MANAGEMENT SYSTEM:

Database management systems can be classified based on several criteria, such as the data model, user numbers and database distribution, all described below.

2.9.1 Classification Based on Data Model:

The most commonly used data model today is the relational data model. This model is supported by well-known DBMS systems such as Oracle, MS SQL Server, DB2 and My SQL. Other traditional models such as hierarchical data models and network data models are still used in the industry, mostly on mainframe platforms. However, due to their complexity, they are rarely used. They are all called traditional models because they preceded the relational model.

In recent years, new object-oriented data models have been introduced. This model is a database management system in which information is represented as objects, as used in object-oriented programming. Object-oriented databases are different from relational databases, which are table-oriented. Object-oriented database management systems (OODBMS) combine the functionality of databases with the functionality of an object-oriented programming language.

Object models have not been as successful as expected and are therefore not widely used. Some examples of object-oriented DBMSs are O2, Object Store and Jasmine.

2.10 CLASSIFICATION BASED ON USER NUMBERS:

A DBMS can be classified based on the number of users it supports. This can be a single-user database system that supports one user at a time, or a multi-user database system that supports multiple users at the same time.

2.10.1 Classification Based on Database Distribution:

Database Management Systems (DBMS) can be classified based on their distribution systems. There are four main distribution systems for database systems:

2.10.2 Centralized systems:

With a central database system, the DBMS and database are stored in a location that is also used by numerous other systems.

2.10.3 Distributed database system:

In a distributed database system, the actual database and DBMS software are distributed from multiple locations connected through a computer network.

2.10.4 Homogeneous distributed database systems:

Homogenously distributed database systems use the same DBMS software from multiple locations. Data exchange between these different parties can be easily managed. For example, library information systems from the same vendor, such as Geac Computer Corporation, use the same DBMS software, allowing easy data exchange between different Geac Library facilities.

2.10.5 Heterogeneous distributed database systems:

In a heterogeneous distributed database system, different sites may use different DBMS programs, but there is common additional software that supports data exchange between sites. For example, several library database systems use the same machine-readable cataloging format (MARC) to support data exchange of library records.

2.11 DBMS APPROACH:

A Database Management System (DBMS) is a software suite that interacts with the user, applications, and the database itself to capture and analyze data. The DBMS approach is a systematic way of managing, storing, and retrieving data in an organized and efficient manner.

Here are some key aspects of the DBMS approach:

- ❖ **Data Definition:** providing a way to define and build the database
- ❖ **Data Manipulation:** providing a way to insert and update data in the database
- ❖ **Query Execution:** recovering information from the data in the database
- ❖ **Data Integrity:** ensuring that data stored is well-formed
- ❖ **Data Security:** enforcing restrictions about who is able to access what data
- ❖ **Provenance:** logging capabilities to provide an audit trail for data changes
- ❖ **Multiuser Concurrency:** supporting the activities of many users at the same time

As can be seen from the above list, a Database Management System (DBMS), a sophisticated software application with varying features across different systems. While not all DBMSs possess the same set of characteristics, these outlined features generally define contemporary DBMSs. Proficiency and specific knowledge about the chosen DBMS are prerequisites for utilizing a database effectively. Notable DBMSs in use today include MySQL, Microsoft SQL Server, Oracle, Postgre SQL, Microsoft Access, and IBM's DB2.

The second element in the database approach pertains to data. Although the physical storage and organization of data may impact performance, the method or location of storage does not determine whether a system adheres to the database approach. As long as the DBMS can access and fulfill its responsibilities regarding the data, the intricacies of storage are inconsequential.

The final component of the database approach is the application, often referred to as "front end" software. This application software interacts with the DBMS, serving information to users and facilitating user invocation of other DBMS functionalities. Notably, the DBMS itself lacks a visual interface, and users do not engage with it directly. Any software that provides a user interface to invoke DBMS procedures falls under the category of application software.

Once the application has determined what the application sends a request to the DBMS, instructing changes to data or seeking specific information. Standardized across all relational databases, Structured Query Language (SQL) is employed to formulate and process these requests.

The DBMS, upon receiving a request, verifies the user's authorization for the specified operation. If permitted, the DBMS executes the operation and communicates the outcome back to the application, which then relays the information to the user. In cases where the operation is unauthorized or encounters errors, the DBMS responds with an appropriate message, with the application responsible for presenting this information to the user. Crucially, the database approach mandates that the application never circumvents the DBMS to directly access stored data.

2.12 SUMMARY:

In this lesson the evolution of DBMS reflects a journey from simple file-based systems to sophisticated, distributed, and cloud-enabled databases. Each stage addressed challenges and embraced technological advancements, leading to the diverse landscape of database systems we have today. The continuous evolution is driven by the need for more efficient, scalable, and flexible solutions to manage the ever-growing volumes of data in various contexts.

2.13 KEY WORDS:

- ❖ **File-based Systems:** Early data storage systems that use flat files to store and manage data.
- ❖ **Hierarchical Model:** A database model that organizes data in a tree-like structure with parent-child relationships.
- ❖ **Network Model:** A database model that represents data as interconnected records, allowing for more complex relationships.
- ❖ **Relational Database Model:** A database model introduced by Edgar Codd, organizing data into tables with rows and columns to establish relationships between entities.
- ❖ **Object:** refers to a self-contained unit that consists of both data and the functions that operate on that data. Objects are instances of classes, which are like blueprints or templates defining the structure and behavior of the objects.
- ❖ **Object-Oriented Databases:** Database systems that represent data as objects, facilitating a more natural representation of real-world entities.

2.14 SELF- ASSESSMENT QUESTIONS:

1. Explain the evolution of Database Management Systems (DBMS)?
2. Define Web-Enabled Database?
3. Write the differences between hierarchical, centralized, and relational database models?

2.15 FUTHUR READINGS:

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LESSON - 3

FILE BASED SYSTEM

OBJECTIVES :

After going through this lesson, you will be able to:

❖ A file-based system is to provide a way for computer systems to organize and manage data stored in files.

❖ In a file-based system, the data is typically stored in individual files, and each file may contain records or information related to a specific entity or concept.

STRUCTURE OF THE LESSON:

3.1 INTRODUCTION:

A File-Based System (FBS) serves as a foundational method for organizing and managing data within a computerized environment. In the early stages of computer technology, before the beginning of complicated database management systems, file-based systems played a important role in data storage and retrieval. The fundamental concept revolves around the idea of storing information in individual files, each containing records that represent specific entities or concepts. This approach provides a structured means of data organization, enabling users to perform operations such as reading, writing, updating, and deleting records.

File-based systems are characterized by their simplicity and ease of implementation. Each file acts as a container for related data, fostering a straightforward approach to information storage. Despite their historical significance, file-based systems have certain limitations, including challenges in handling complex relationships between data sets and managing large volumes of information efficiently.

As technology has advanced, modern database management systems (DBMS) have largely supplanted file-based systems in many applications. However, understanding the principles and characteristics of file-based systems remains valuable for grasping the historical evolution of data management and appreciating the foundations upon which more sophisticated systems have been built. In this exploration, we delve into the key features, functions, and limitations of file-based systems, shedding light on their role in the evolution of information management in the realm of computing.

3.2 DATABASE MANAGEMENT SYSTEM:

DBMS is a software solution that allows you to create and maintain databases in which you can store the data. It basically refers to the system, which helps to store and retrieve the data systematically from the database. The different user such as database manager performs separate roles to manage the database in the DBMS, which supports the multiple layered architecture that provides physical and logical data independence. You can use DBMS to define the database that involves in specifying the data types such as structures and constants. DBMS also allows you to construct and manipulate the database. Constructing a database is a process of storing the data on some storage medium such as floppy drive, compact disk (CD) or hard disk drive (HDD). Manipulating the database involves in performing functions such as querying the database to retrieve the specific data such as updating the database to reflect changes made by the user and generating reports from the data.

3.2.1 Features of DBMS:

To understand the basics of database management system, you must know the terms and definitions that are used in DBMS. These terms and definitions constitute the DBMS terminology. DBMS is a software program, which runs on the user machine or on the computer server. DBMS accepts the queries from the user and responds to these queries. It has the following features:

- ❖ **Structured data:** DBMS enables you to structure the data as tables, records or objects.
- ❖ **Query language:** DBMS provides a query language, such as SQL to process the user requests.
- ❖ **Multiuser access:** DBMS allows several users to access the data stored in the database and it provides security features, which restricts some users from viewing or manipulating the data.
- ❖ **Data dictionary:** DBMS provides a data dictionary, which contains the structure of the database.

3.3 ADVANTAGES OF DBMS:

DBMS is a set of programs that allows you to create and maintain databases. The advantages of the database management system are:

- ❖ It controls data redundancy.
- ❖ It restricts unauthorized access.
- ❖ It provides persistent storage for the program objects and data structures.
- ❖ It provides multiple user interfaces.
- ❖ It provides integrity constraints.
- ❖ It provides backup and recovery.

Backup and recovery:

Data stored in a database can be recovered in case of hardware or software failures. The backup and recovery subsystem of the database management system allows in taking backup of whole data when database is successfully working and recovering of data when hardware or software failure occurs. For example, if hardware failure occurs when the database is being updated, then the data stored in the database is restored and the program is resumed where it was interrupted.

A file system, on the other hand, is a type of software that is responsible for storing entire files on a storage medium. In the file system, all files are organized in directories and folders and sometimes the same file can be duplicated in multiple locations. This means that the risk of data inconsistencies with file systems is much higher. In addition, file systems typically have significantly less storage capacity than DBMSs and can only change the metadata of a given file, not its contents. Examples of file systems are Microsoft's NTFS and Apple's Hierarchical File System.

Advantages of file systems include:

- Ease of use
- Cost effective
- Greater efficiency
- Suitable for managing personal data
- Disadvantages of file systems include:
- Limits its capacity
- Limited functionality 98 7Lower security
- Increased data inconsistency
- No backup and restore functions

3.4 DRAWBACKS OF FILE BASED SYSTEM:

3.4.1 Data Redundancy and Inconsistency:

Since data is located in different private data files, there is a risk of redundancy and resulting inconsistency. For example, in the example above, the same customer may have both a savings account and a mortgage. In this case, customer data may be duplicated because programs that support two functions store the corresponding data in two different data files. This creates redundancy in customer data. Because the same data is written to two files, an inconsistency occurs when a data change made in one file is not reflected in the other file

3.4.2 Unanticipated Queries:

In a file-based system, handling unexpected/ad hoc requests can be difficult because they require changes to existing programs.

3.4.3 Data Isolation:

Although data used by different programs in an application may be linked together, they reside in isolated data files.

3.4.4 Concurrent Access Anomalies: In large multi-user systems, the same file or data set may require concurrent access by multiple users. Managing this on file-based systems is difficult.

3.4.5 Security Issues:

For data-intensive applications, data security is a key concern. Users should be given access only to required data and not the whole database. In a file-based system, this can be handled only by additional programming in each application.

3.4.6 Integrity Problems:

In any application, there will be certain data integrity rule which needs to be maintained. These could be in the form of certain conditions/constraints on the elements of the data records. In the savings bank application, one such integrity rule could be "Customer ID, which is the unique identifier for a customer record, should be non-empty". There can be multiple such integrity rules. In a file-based system, all of these rules must be explicitly programmed into the application program. You may have noticed that we do not mean to say that solving the above problems, such as: Some issues, such as concurrent access, security issues, integrity issues, etc., are not possible in a file-based system. However, the real problem was precisely that. All these problems were typical of every data-intensive application, each application had to solve them on its own. An application developer not only has to deal with implementing business rules for applications, but also dealing with common problems.

3.5 COMPONENTS OF DATABASE SYSTEM:

Hardware, software, data, database access language, procedures and users together make up the components of a DBMS.

Let's discuss each element individually in detail.

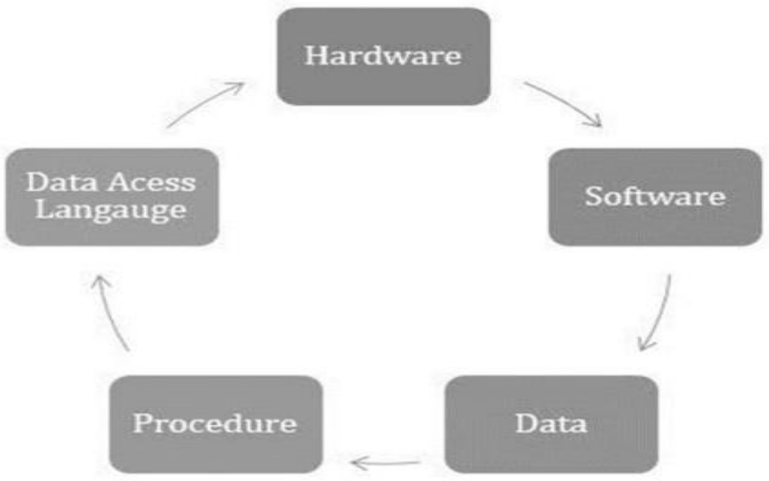
- ❖ **Hardware:** Hardware is the actual computer system used to store and access a database. Traditional DBMS hardware consists of additional storage devices such as hard drives. Databases run on a variety of machines, from microcomputers to mainframes.
- ❖ **Software:** Software is the actual DBMS between the physical database and the users of the system. All user access requests to the database are processed by the DBMS.
- ❖ **Data:** It is an important part of the database management system. The main task of the DBMS is data processing. Databases are used to store, retrieve, and update data in and from databases.
- ❖ **Users: There** are numerous users who can access or download data on demand through the applications and interfaces provided by the DBMS.

The users of the database can be classified into different groups they are

- Native Users- directly interact with the database using the inherent tools, interfaces, and query languages provided by the specific DBMS.
- Online Users- entities actively accessing and interacting with the database in real-time through applications or interfaces that facilitate immediate data transactions and updates.
- Sophisticated Users- professionals possessing in-depth knowledge and expertise, often engaging with complex database functionalities and optimization strategies beyond basic interactions.
- Specialized Users- In a Database Management System (DBMS) are individuals with expertise in specific domains or functionalities, utilizing the database for specialized tasks or analysis within their specialized field.

- Application Users-These are individuals who interact with the database indirectly through software applications, relying on predefined interfaces for accessing and manipulating data.
- Database Administrator (DBA) - It is a professional responsible for the overall management, security, and performance optimization of the database system.

The components of DBMS are given below in pictorial form:



3.6 SUMMARY:

File-based systems organize and manage data through individual files, aiming for efficient storage, retrieval, and organization. They enable operations like reading, writing, updating, and deleting records. While providing some data independence, file-based systems have limitations, such as a lack of data relationships. Modern database management systems have largely replaced them due to advanced features and improved data integrity.

3.7 KEY WORDS:

- ❖ **File-based system:** It is a method of organizing and managing data through the storage of information in individual files.
- ❖ **Data sharing:** The capability of sharing data among different programs and users through files.
- ❖ **Data security:** Measures implemented to control access to files and protect data from unauthorized users.
- ❖ **Relationships:** Connections or associations between different sets of data within the system.

3.8 SELF-ASSESSMENT QUESTIONS :

1. Define what a "record" is in the context of a file-based system.
2. How does a file-based system handle the storage of data?
3. Write the Data Redundancy and Inconsistency
4. Explain the Components Of Database System?

3.9 FURTHER READINGS :

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LESSON - 4

DATA MODELS

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ The data models are to provide a structured framework for organizing and representing data, facilitating efficient storage and retrieval.
- ❖ These models ensure data integrity by enforcing constraints and rules, preventing anomalies that could compromise accuracy.
- ❖ It supports seamless integration of data, enhancing interoperability between different components of a system.
- ❖ Data models in DBMS also aim to optimize query performance through normalization, minimizing redundancy and improving overall database efficiency.
- ❖ The objectives collectively contribute to the development of reliable, scalable, and adaptable database systems.

STRUCTURE OF THE LESSON:

4.1 INTRODUCTION:

The Data models play a crucial role in shaping the architecture and organization of stored information. The primary objectives of data models revolve around providing a structured representation of data, ensuring its integrity, and optimizing the efficiency of database operations. These models serve as a blueprint for designing databases, defining relationships between entities, and enforcing constraints to prevent inconsistencies. By supporting efficient querying, retrieval, and storage, data models contribute to the seamless integration of data within a system. In essence, the objectives of data models in DBMS aim to create a robust foundation for reliable, scalable, and well-organized databases that meet the diverse information needs of applications and users.

4.2 DATA MODEL:

A data model in DBMS describes the structure of a database. The structure of a database refers to the data types, constraints and relationships used on the data. In addition, the data model also includes the basic operations of retrieving data from a database and updating the data on the database.

4.2.1. A data model describes:

Structure: It represents how the data is organized in a database. The data can be organized using hierarchical, network, relational, or object- oriented data model.

Integrity: It provides a definition of rules to indicate whether or not the defined structure can be used to organize data in a database.

Manipulation: It provides a language in which you can update the data in a database.

Querying data: It provides a language in which the data in the database is queried.

For DBMS implementation, you can use various data models, which include all database-related concepts, for describing the structure of a database. The various data models used in DBMS are as follows:

- Hierarchical
- Network
- Relational
- Object-oriented

4.2.2 Hierarchical Data Model:

In the hierarchical data model, data is organized in a tree structure. The tree structure consists of the hierarchy of parent and child data segments. The hierarchical data model can have repeated information in the child data segments. Consider the tree structure of Figure below that stores the hierarchical information of an organization.

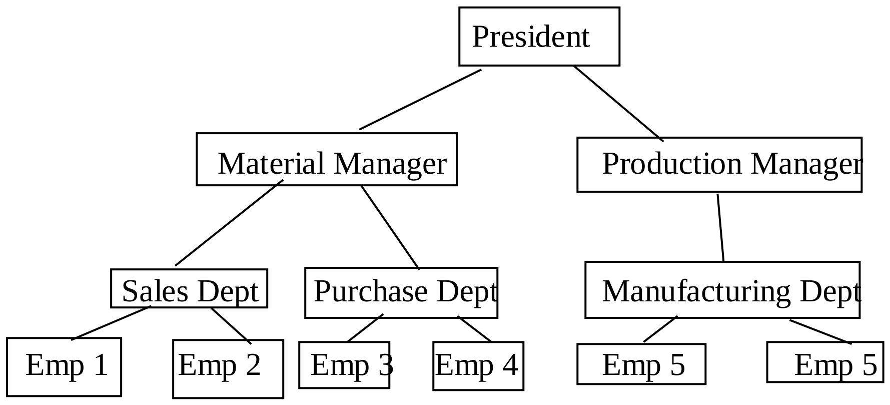


Figure: Hierarchical Data Model

The above figure shows the hierarchical data model of an organization database in which the president of the organization is at the top of the hierarchy and the employees in a department are at the bottom of the hierarchy.

The advantages of the hierarchical data model are:

- ❖ It is simple to construct and operate on the data in the hierarchical model.
- ❖ It helps to improve the hierarchically organized domains such as product information in manufacturing and employee information in organization.
- ❖ The disadvantages of the hierarchical data model are:

- ❖ It requires navigational and procedural processing of data.
- ❖ It provides less scope of query optimization.

4.2.3. Network Data Model:

In the network data model, which is also called the CODASYL model; data is organized using set constructs. A set consists of an owner record type, a set name and a member record type. Unlike the hierarchical model, a child segment in a network model can have multiple parent segments.

The advantages of the network data model are as following:

- ❖ It represents complex relationships and the affect of operations such as add and delete on the relationships.
- ❖ It uses constructs such as FIND, FIND owner and FIND NEXT within a set that allows the users to navigate through the database.

The disadvantages of the network data model are:

- ➢ It provides navigational and procedural processing of data.
- ➢ It provides complex array of pointers that is read through a set of records.
- ➢ It provides less scope for query optimization.

4.2.4. Relational Data Model:

In this model the data is stored in tables, which are also called relations. The related tables or relations in the relational data model form a database. The properties of relational data model are:

- ❖ Each row in a table is unique from every other row in the table.
- ❖ Each row contains atomic data, which implies that data is not repeated and does not contain structures such as arrays.

In a relational model, tables are used to organize data. A table consists of columns or fields that represent attributes of an entity. Each row or tuple in a table represents the occurrence of an entity and must consist of the value that uniquely identifies the row. Such a column that uniquely identifies the rows is called the primary key. The relational model also consists of foreign keys that allow joining data of two tables. To understand the relational model, consider Table that represents the customer database.

Cust name	**Cust id**	**Cust city**
John	1001	Xyz
Tom	1002	Pqr
Ken	1003	Abc

In the above table, Cust_name represents the name of the customers, Cust_id is the unique number for each customer and Cust_city represents the city of the customers.

The relational data model uses the set theory and it is based on the concept of mathematical relation, which contains several data elements. The basic characteristics of the relational model are relational algebra and relational calculus. The relational algebra is a set of operations for manipulating relations and specifying queries. The relational calculus provides a declarative way to specify the database queries. The relational algebra and the relational calculus are two different means used for representing the database queries. Any relational algebraic expression can be converted into a corresponding expression in the relational calculus and vice versa.

4.2.5. Object-Oriented Data Model:

The object-oriented (OO) data model consists of a collection of entities. Entity refers to the concept or object described in the database and is represented as a class in the object-oriented data model. The instances of a class are called objects. An attribute provides additional information to describe an entity. The attributes of a class help to distinguish an object from the other object.

Consider the employee database having employee as an entity set. The different attributes of the entity employee can be employee_name, employee_code and employee_salary. Then the employee database can be represented in object- oriented data model as follows:

Class

Employee

Attributes

Employee_name

Employee_code

Employee salary

Behaviour

Works for manager

The object-oriented data model consists of:

- Object: It represents a real world entity.
- Attributes and methods: These represent the set of values for the attributes of the object and set of methods to operate on the attributes of the object.
- Class: It is a group of all the objects that share the same set of attributes and methods.
- Class hierarchy and inheritance: It represents the derivation of a new class from an existing class, which is called a superclass. The new class is called subclass and it inherits all the attributes and methods of the existing class. In addition, it also consists of additional attributes and methods of the class.

4.3. DBMS ARCHITECTURE:

The DBMS architecture is designed so that the storage details of data in a database are hidden from users. The users accessing a database need to work with the data rather than any concern for how the data is physically stored in the database. The DBMS architecture allows:

- ❖ Different views of the database for the users. It provides customized view to each of user accessing the same data from the database. Each user view is independent and any change to the view does not affect other views.
- ❖ The internal structure of the database remains unaffected by changes made to the physical storage of data.
- ❖ The database administrator to change the structure of a database without the user views of the database being affected.

The architecture of the DBMS is also called American National Standards Institute/Standards Planning and Requirements Committee (ANSI/SPARC) model.

The (ANSI/SPARC) model or DBMS architecture is divided into three levels, which are as following:

Internal level: It specifies the way in which the data is physically stored in the database. The internal or physical level of the database system architecture also provides description of the relationship that exists between the data.

External level: It specifies the way in which the data stored in the database is viewed by the users.

Conceptual level: It specifies the level of interaction between the internal level and the external level of system architecture.

Figure below shows the Different Levels of DBMS Architecture

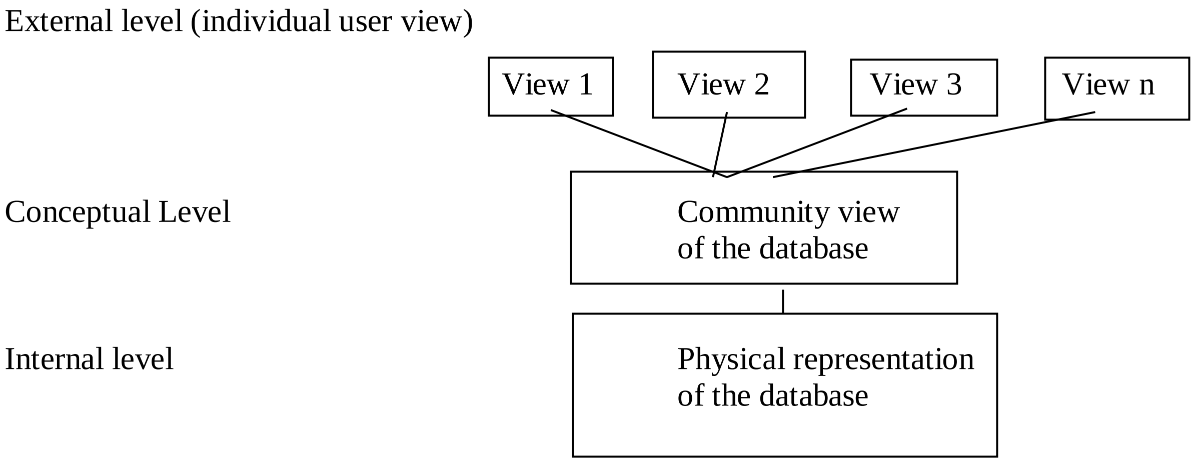

External level (individual user view)
View 1
View 2
View 3
View n
Conceptual Level
Community view of the database
Internal level
Physical representation of the database

Figure: Different Levels of DBMS Architecture

To understand the concept of levels in DBMS architecture, consider an employee database that contains details of the employee such as employee number and department number. The internal level of architecture for the employee database can be represented as:

```
Stored_emp BYTES-20
Prefix  TYPE=BYTE (5), OFFSET=0
Emp     TYPE=BYTE (6), OFFSET=5,            INDEX=EMPX
Dept    TYPE=BYTE (2), OFFSET=10
Pay     TYPE=FULLWORD, OFFSET=10
```

In the preceding example for the internal level, stored record types Stored_emp represent employees, which is twenty bytes long. The Stored_emp consists of four stored fields that are prefix, emp, dept and pay. The prefix contains control information such as flags or pointers. The other data fields represent three properties of employees and also the records in Stored_emp are indexed by using an index.At the conceptual level, database contains the information about an entity. For example, for an employee database, the conceptual level includes the information about the employee entity such as employee_number, dept_number and salary. The conceptual level of the architecture for the employee database can be represented as:

```
employee
employee_number          CHARACTER (6)
department_number        CHARACTER (4)
salary                   NUMERIC (5)
```

At the external level, the view of the database consists of two fields- employee number and salary. The external view shows only the fields that a user needs to view. For example, for an employee database the external level of architecture consists of two fields, emp# and salary which can be represented as:

```
 DCL 1        empp,
2 emp#          CHAR (6),
2 sal FIXED BIN (30);
```

The various data fields can have different names in the various views of the database. For example, the employee number at the internal level is represented by emp and at the conceptual level it is represented by employee_ number.

4.4. DBMS VENDORS AND THEIR PRODUCTS:

DBMS Vendors and their products are as shown below table

Vendor	**Product**
IBM	-DB2/MVS
	-DB2/UDB
	-DB2/400
	-Informix Dynamic Server (IDS)
Microsoft	-Access
	-SQL Server
	-DesktopEdition(MSDE)
Open Source	-MySQL
	-PostgreSQL
Oracle	-OracleDBMS
	-RDB
Sybase	-Adaptive Server Enterprise(ASE)

	-Adaptive Server Anywhere(ASA)
	-Watcom

4.5. SUMMARY :

The Data models in Database Management Systems (DBMS) serve as conceptual frameworks for organizing and representing information in a structured manner. Their primary purpose is to provide a clear blueprint for database design, defining the relationships and constraints that govern data integrity. By minimizing redundancy and optimizing storage, data models enhance the efficiency of data retrieval and manipulation operations. They play a crucial role in facilitating seamless integration between different components of a system, promoting interoperability. Overall, data models in DBMS contribute to the creation of reliable, scalable, and well-organized databases, ensuring that information is managed systematically to meet the diverse needs of applications and users

4.6. KEY WORDS :

- **Relationship:** A relationship signifies an association between two or more entities, describing how they are connected.
- **Primary Key:** A primary key is a unique identifier for each record in a table, ensuring the uniqueness and integrity of the data.
- **Foreign Key:** A foreign key is a field in a table that refers to the primary key in another table, establishing a link between the two tables.

4.7 SELF – ASSESSMENT QUESTION :

1. What is Data model?
2. Explain the Data models in DBMS?
3. Describe the DBMS architecture?
4. Write the DBMS vendors and their products?

4.8 FURTHER READINGS :

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LESSON – 5

ENTITY-RELATIONSHIP MODEL

OBJECTIVES:

After going through this lesson, you will be able to:

❖ In Data Base Management Systems the Entity-Relationship (ER) model is a conceptual data modeling technique used in Database Management Systems (DBMS) to represent the structure of a database in a clear and concise manner.

❖ The main objectives of the Entity-Relationship model are to provide a clear and visual representation of the data structure and to facilitate communication between stakeholders involved in the design and development of a database system.

STRUCTURE OF THE LESSON:

5.1 INTRODUCTION:

In this lesson we formalize data modeling based on the powerful concept of business rules and we describe the entity-relationship (E-R) data model. Business rules are derived from policies, procedures events, functions, and other business objects, and state constraints on the organization. Explained the guidelines for the clear naming and definition of data objects in a business. In terms of conceptual data modeling, names and definitions must be provided for entity types, attributes, and relationships.

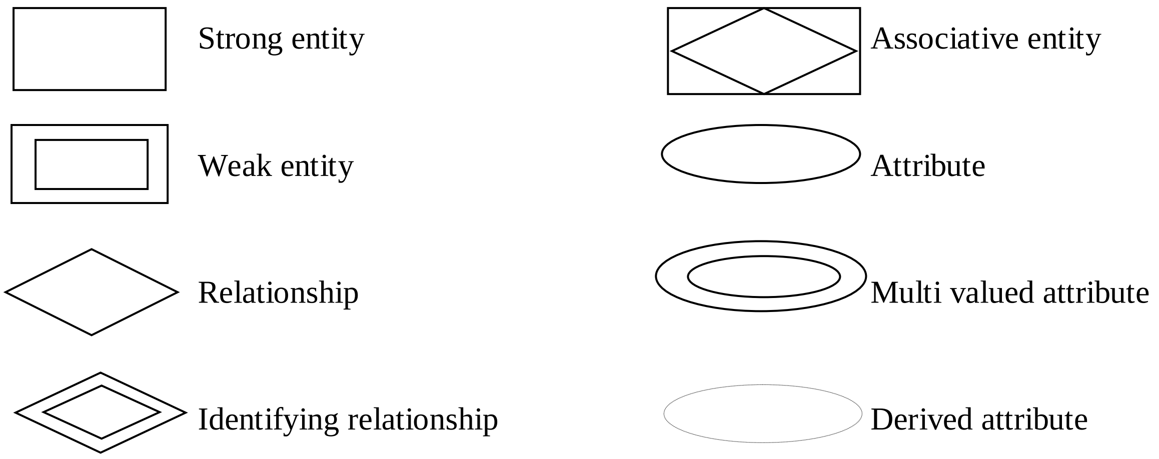
The E-R model is used to construct a conceptual data model, which is representation of the structure and constraints of a database that is independent of the software and its associated data model that will be used to implement the database. We introduced three important concepts as- sociated with relationships: the degree of a relationship, cardinality of relationship, and participation constraints in a relationship.

5.2 ENTITY-RELATIONSHIP MODEL CONSTRUCTS:

The basic constructs of the entity-relationship are entities, relationships and attributes. As shown in model allows numerous variations for each of these constructs. The rich- ness of the E-R model allows designers to model real-world situations accurately and expres- sively, which helps account for the popularity of the model.

Figure Basic E-R notation

Basic symbols



5.2.2 Entities:

Entity: A person, place, object event or concept in the user environment about which the organization wishes to maintain data.

Person:	**EMPLOYEE, STUDENT, PATIENT**
Place:	**STORE, WARESHOUSE, STATE**
Object:	**MACHINE, BUILDING, AUTOMOBILE**
Event:	**SALE, REGISTRATION, RENEWAL**
Concept:	**ACCOUNT, COURSE, WORK CENTRE**

There is an important distinction between entity types and entity instances. In an E-R diagram the entity name is placed inside the box representing the entity type. (See above Figure).

Entity Type: A collection of entities that share common properties or characteristics.

Entity Instance: A single occurrence of an entity type.

Strong Entity: An entity that exists independently of other entity types.

Entity Type:	EMPLOYEE
Attributes:	
EMPLOYEE NUMBER	CHAR(10)
NAME	CHAR(25)
ADDRESS	CHAR(30)

CITY	CHAR(20)

Two instances of EMPLOYEE

111-12-6789	222-23-5678
RAMA KRISHNA	PRASAD
JKC COLLEGE	KOTHAPET
GUNTUR	GUNTUR

Example STUDENT, EMPLOYEE, COURSE
Weak Entity: An entity type whose existence depends on some other entity type.

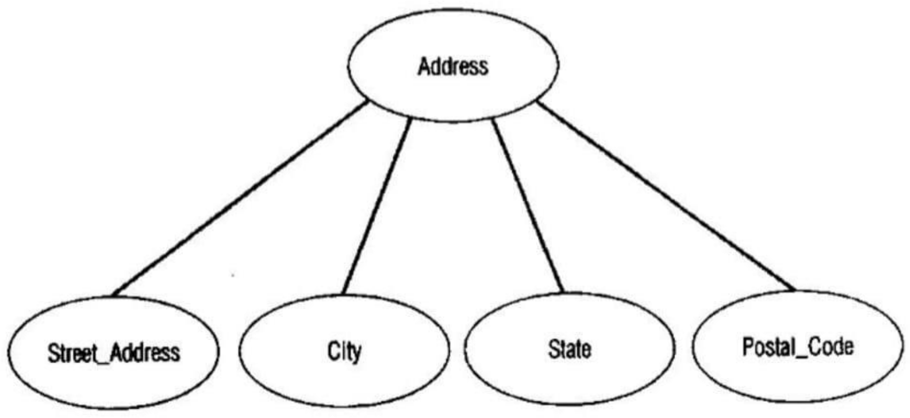
5.3 RELATIONSHIPS:

- ❖ A relationship type is a meaningful relation between or among entity types.
- ❖ A relationship is an association among one or more entities.

Note: Most of our work centers on relationship types.

- ❖ You can use the diamond shape to represent a relationship or you can write the relation- ships one the relationship line at each end:
- ❖ Consider an example of a company that tracks the training courses its employees are taking. The analysts developed the following Employee-Course ERD.

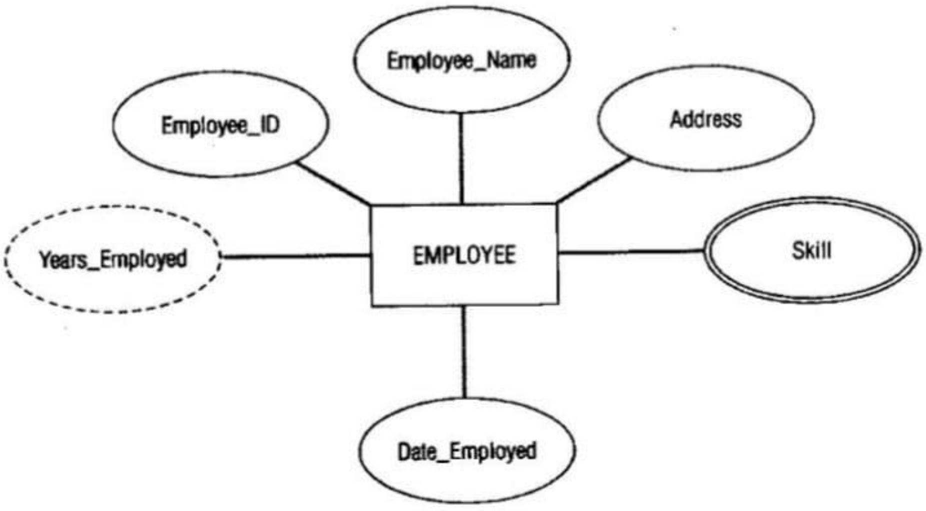
Figure : A composite attribute Address



Multivalued Attribute: A multivalued attribute is an attribute that may take on more than one value for a given entity instance. For example, the employee entity type in given picture has an attribute name Skill, whose values record the skill (or skills) for that employee.

Derived Attribute: An attribute whose values can be calculated from related attribute values. For example, in an organization, the EMPLOYEE entity type has a Date_Employed attribute. If users need to know how many years a person has been employed using Date_Employed and today's date. So, Years_Employed is a derived attribute (which is shown in the below picture)

Figure: Entity with multivalued attribute and derived attribute



Identifier (Key Attribute): An identifier is an attribute (or combination of attributes) that uniquely identifies individual instances of an entity type. The identifier for STUDENT entity is Student_Id. An attribute such as Student_Name is not a candidate identifier, since many students may potentially have the same name. Each entity instance must have a single value for the attribute and the at- tribute must be associated with the entity. We underline identifier names on the E-R diagrams, which are shown in below picture.

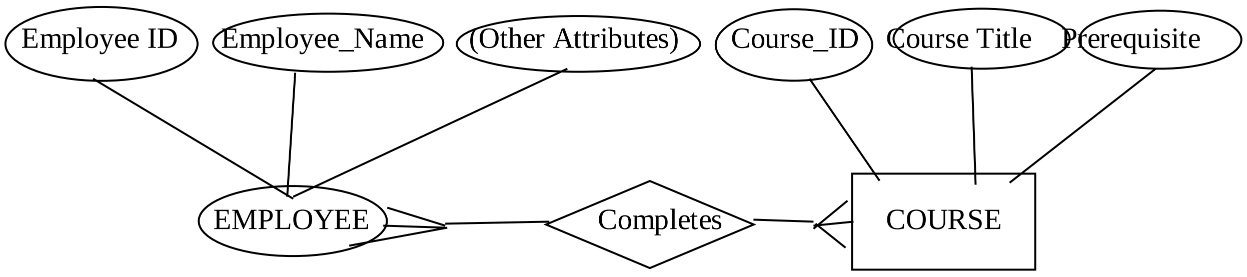
5.3.1 Basic Concepts and Definitions in Relationships:

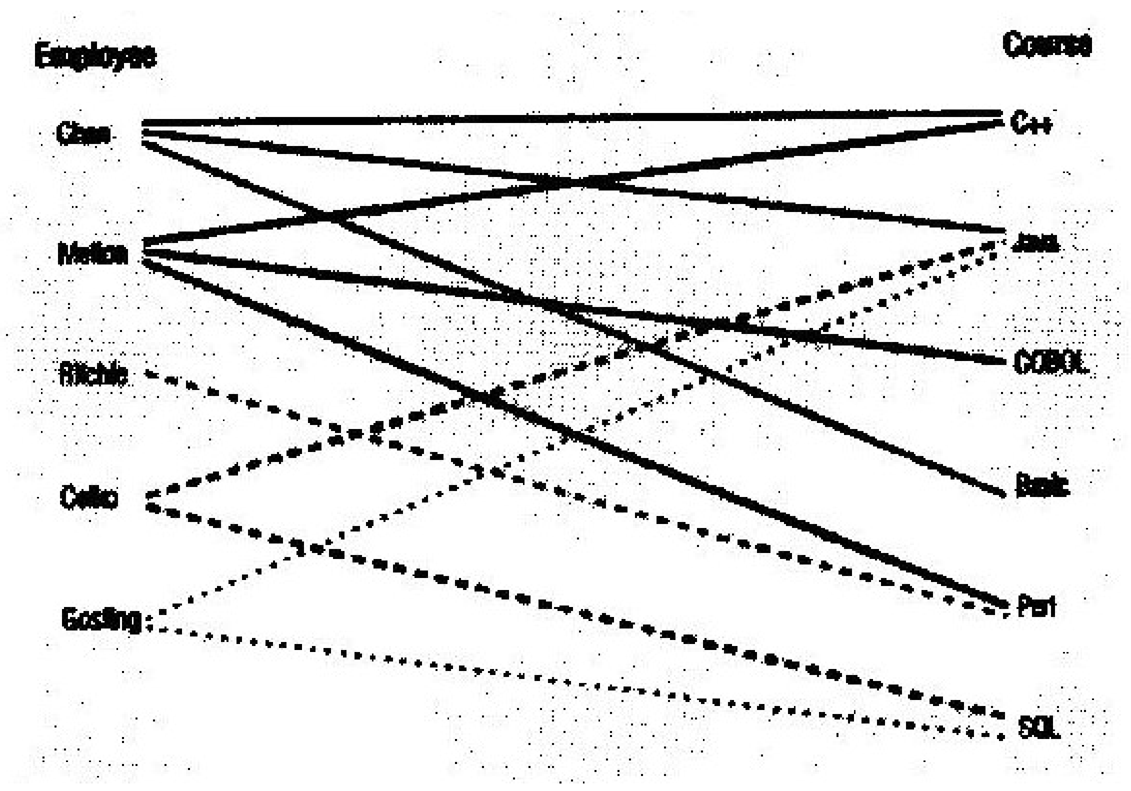
Relationship: Relationship is an association among the instances of one or more entity types. **Relationship Type:** It is a meaningful association between (or among) entity types. A relationship type is denoted by a diamond symbol containing the name of the relationship.

Figure: Relationship type and instances

(a) Relationships type

(b) Relationship instances

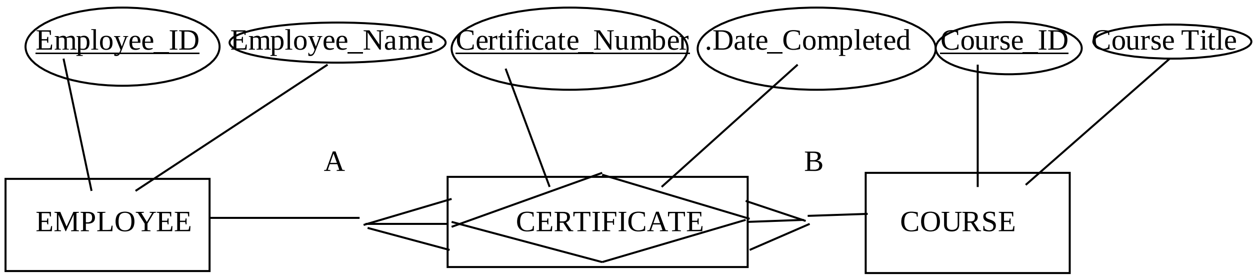




Relationship Instance: It is an association between (or among) entity instances where each relationship instance includes exactly one entity from each participating entity type.

Associative Entity: It associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.

Figure below An associative entity (a) Attribute on a relationship



The associative entity CERTIFICATE is represented with the diamond relationship symbol enclosed within the entity box.

Analysts use the relations symbols on the diagram while others do not. For example, the relation "Takes" makes sense when we read the ERD from left to right, as: an employee Takes courses", but it doesn't make sense when we say "Course Takes employee". Thus, the use of relations is optional.

The above ERD represents a 'many-to-many' relation, stating "an employee takes many courses and a course may be attended by many employees". In order to understand this relation- ship, let's review the relationship instances, showing the occurrences of the relations between entities. The relationship instances below shows that employee named Chen takes

C++, Java and Basic. Similarly, the course C++ was attended by many students: Chen and Melton.

5.3.2 Attributes or Relationship:

Attribute may be associated with a many-to-many relationship, as well as with an entity. For example, suppose the organization wishes to record the date when an employee compiletes each course. This attributes is named Date_Complted.

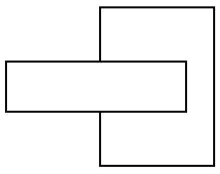
Where the attributes Date_Completed should be placed on the E-R diagram. Refferening to Figure below you will notice that Date_Completed has not been associated with either the EMPLOYMEE or COURSE entity. That is because Dtae_ Completed is a property of the relation- ship Completeds, rather than a property of either entity. In other words, for each instance of the relationship Completes, there is a value for Date_Completed. One such instance shows that the employee named Melton completed the course titled C++ on 06/2000.

5.3.3 Degree of a Relationship:

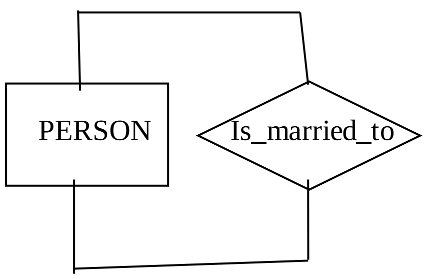
The degree of relationship is the number of entity types that participate in that relationship. The three most common relationship degrees in E-R models are

1. Unary Relationship
2. Binary Relationship
3. Ternary Relationship

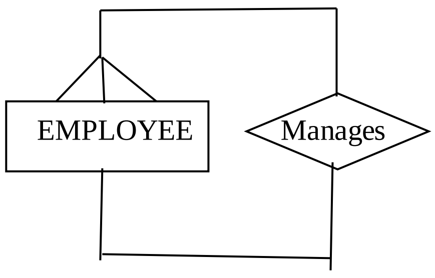
Unary Relationship: A relationship between the instances of a single entity type. This is also called as a recursive relationship.



Unary



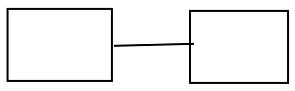
One-to-one

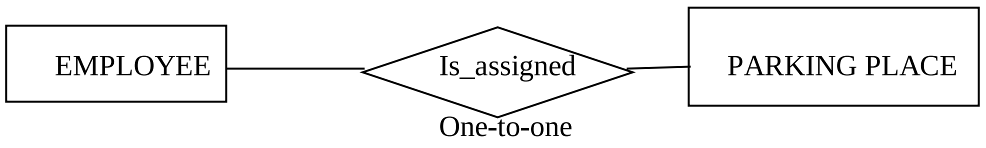


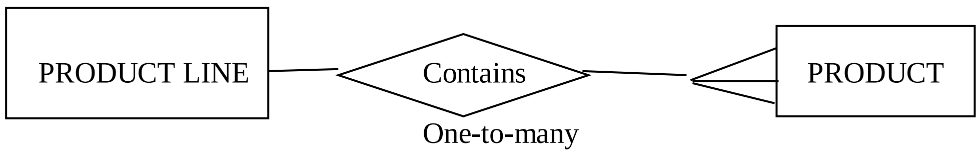
One-to-many

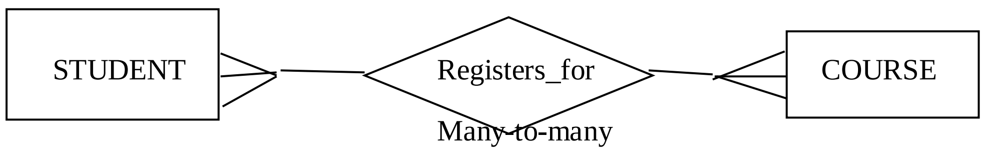
Unary relationships are also called recursive relationships. In the above example "Is_married-to" is shown a one-to-one relationship between instances of the PERSON entity type.

Binary Relationship: A binary relationship is a relationship between the instances of two entity types.


Binary

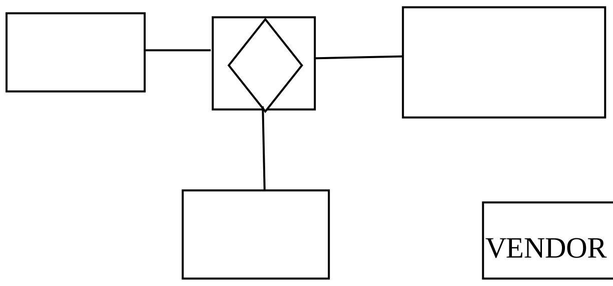


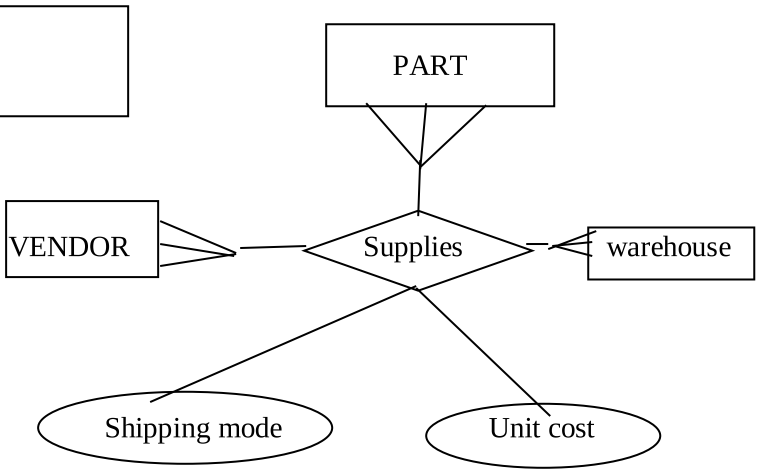




In the above example an employee is assigned one parking place, each parking place is assigned is assigned to one employee.

Ternary Relationship: A Ternary relationship is a simultaneous relationship among the instances of three entity types.


Ternary



In the above example vendors can supply various parts to warehouses. The relationship supplies is used to record the specific parts that are supplied by a given vendor to a particular warehouse.

5.3.4 Cardinality Constraints:

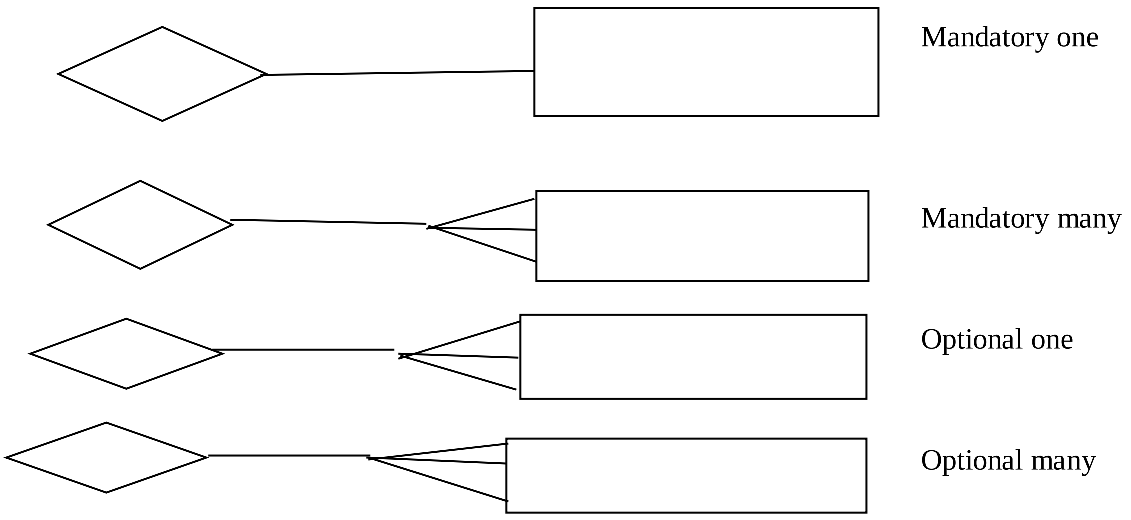
Cardinality Constraints: Cardinality Constraint Specifies the number of instances of one entity that can (or must) be associated with each instance of another entity.

Minimum Cardinality: The Minimum number of instances of one entity that may be associated with each instance of another entity.

Maximum Cardinality: The Maximum number of instances of one entity that may be associated with each instance of another entity.

As a relationship line is followed from an entity to another, near the related entity two sym- bols will appear. The first of those is the optionality indicator. A circle (TM) indicates that the relationship is optional-the minimum number of relationships between each instance of the first entity and instances of the related entity is zero. One can think of the circle as a zero, or a letter O for "optional." A stroke (1) indicates that the relationship is mandatory-the minimum number of relationships between each instance of the first entity and instances of the related entity is one.

Relationship cardinality



The second symbol indicates cardinality. A stroke (1) indicates that the maximum number of relationships is one. A "crows-foot" () indicates that many such relationships between instances of the related entities might exist.

The following diagram indicates all of the possible combinations:

5.4 MODELING TIME - DEPENDER DATA:

Database contents vary over time. Fox example, in a database that contains product information, the unit rice for each product may be changed as material and labor costs and market conditions. A time-stamp is simply a time value that is associated over time when we need to maintain a history of those data values. Time stamps may the value becomes valid or stops being valid, or the time when critical actions were performed. The use of time stamping (given example) is often adequate for modeling time-dependent data.

we can conceptualize this requirements as a series of prices and the effective data for each price. This results in the multi-valued attribute named Price-History. The components of Price-History are Price and Effective-Date. An important characteristics of such a composite, multi-valued attribute is that the component attributes go together. Thus, each Price is paired with the corresponding Effective-Date.

5.4.1 Naming and Defining Relationship:

In addition to the general guidelines for naming data objects, there are a few special guidelines for naming relationship, which follows.

A relationship name is verb phase. Relationships represented access being taken, usually in the present tense. A relationship name states the action taken, no the result of the action. The name states the essence of the interaction between the participating entity types, not the process involved.

You should avoid vague names, such as Has of Is_ related_ to. Use descriptive verb phrases, often taken from the action verbs found in the definition of the relationship.

There are also some specific guidelines of defining relationships, which follows:

- ❖ A relation definition explains what actions are being taken and possibly why it is important.
- ❖ It may also be important to give examples to clarity the action.
- ❖ The definition should explain any optional participation.
- ❖ A relationship definition should also explain the reason for any explicit maximum cardinality other than many.
- ❖ A relationship between should explain any mutually exclusive relationships.
- ❖ A relationship definition should explain any restriction on participation in the relation- ship
- ❖ A relationship definition should explain the extent of history that is kept in the relation- ship.
- ❖ A relationship definition should explain whether an entity instance involved in a relationship instance can transfer participation to another relationship instance.

5.5. ATTRIBUTE CLASSIFICATION :

Attribute classification in Database Management Systems (DBMS) refers to the categorization and organization of attributes or fields within a database. An attribute is a characteristic or property of an entity, and proper classification is crucial for designing an efficient and well-structured database.

Attributes can be classified based on various criteria:

1. **Simple and Composite Attributes:**
   - Simple Attribute: It cannot be divided any further. For example, the "Age" of a person.
   - Composite Attribute: It can be divided into sub-parts, each representing a more basic attribute. For instance, an address attribute can be decomposed into street, city, and postal code.
2. **Single-valued and Multi-valued Attributes:**
   - Single-valued Attribute: It holds a single value for each entity. For example, the "Date of Birth" of a person.

- Multi-valued Attribute: It can hold multiple values for each entity. An entity might have multiple phone numbers, and the "Phone Number" attribute would be multi-valued.

3. **Stored and Derived Attributes:**
    - Stored Attribute: The value is physically stored in the database. For example, the "Age" attribute can be stored since it can be calculated from the "Date of Birth."
    - Derived Attribute: Its value is not stored but derived from other attributes via a calculation. An example is the "Total Price" of an item in an invoice, which can be derived from the quantity and unit price.
4. **Key Attributes:**
    - Key Attribute: An attribute that is used to uniquely identify an entity within an entity set. For example, the "EmployeeID" in an employee database.
5. **Single-valued and Multi-valued Composite Attributes:**
    - Single-valued Composite Attribute: A composite attribute that can have only one value for each entity. For instance, an attribute representing a person's full name.
    - Multi-valued Composite Attribute: A composite attribute that can have multiple values for an entity. An example is an attribute representing a person's academic degrees.
6. **Null Attributes:**
    - An attribute that may not have a value for every entity is classified as a null attribute. It indicates the absence of a known value and is important for handling missing or unknown information.
7. **Simple and Complex Attributes:**
    - Simple Attribute: A basic atomic attribute that cannot be further divided.
    - Complex Attribute: An attribute composed of sub-parts that can have their attributes. For example, an attribute representing a car's engine, which may have attributes like horsepower and fuel type.

Understanding attribute classification is crucial during the database design phase as it helps in creating a well-structured schema, normalization, and optimization of queries. Properly classified attributes contribute to the overall efficiency, integrity, and reliability of a database system.

5.6 SUMMARY :

The Entity-Relationship model is a foundational tool in database design, offering a visual and abstract representation of data structure. It aids in communication, planning, and the creation of a robust database schema, supporting the efficient management and utilization of data in a database system.

5.7 KEY WORDS :

- **Entity:** An entity is a real-world object or concept that has a distinct and independent existence, represented in the database. Entities have attributes that describe their characteristics.
- **Attribute:** An attribute is a property or characteristic of an entity that provides additional information about the entity. Attributes are the building blocks of entities.
- **Relationship:** A relationship describes an association between two or more entities. It illustrates how entities are connected and interact with each other.

- **Cardinality:** Cardinality specifies the number of instances of one entity that can be associated with another entity. It defines the relationship's multiplicity.

5.8 SELF-ASSESSMENT QUESTIONS :

1. What is entity?
2. Describe the entity-relationship model constructs?
3. Explain the Basic Concepts And Definitions In Relationships?
4. Write the Degree of a Relationship?

5.9 FURTHER READINGS :

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LESSON – 6

RELATIONSHIP DEGREE, RELATIONSHIP CLASSIFICATION

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ The Studying Relationship Degree and Relationship Classification in a Database Management System (DBMS) typically revolve around understanding, applying, and optimizing the representation of relationships between entities in a database.

STRUCTURE OF THE LESSON:

6.1 INTRODUCTION:

In the Database Management Systems (DBMS), the art of designing an efficient and logically structured database goes beyond merely capturing individual entities and their attributes. The effectiveness of a database hinges on the thoughtful representation of relationships between these entities, forming the backbone of data integrity, retrieval, and overall system functionality. Two fundamental concepts that play a pivotal role in shaping these relational landscapes are Relationship Degree and Relationship Classification.

6.2 RELATIONSHIPS:

A relationship refers to the association between two or more entities. For example, you can define a relationship that associates customer name, cust_name and customer's social security number, cust_ssn.

The function of an entity in a relationship is known as the role of that entity. The entity sets that participate in a relationship are distinct and the roles of the entity are not specified. It is specified when you need to describe the relationship that exists between the two entities. This is done when an entity participates in the relationship in different roles. Such a relationship in which you need to specify the function of each entity participating in the relationship is called the recursive relationship. For example, consider an entity set 'employee' that provides information about all the employees working in the organization. You can create a relationship 'working_for' between the ordered pairs of the employee entity. In the 'working_for' relationship, the first employee in the ordered pair takes the role of the manager and the second employee in the pair takes the role of a worker. Therefore, the ordered pair for working relationship would be (manager, worker). A relationship set is the set of similar relationships. If X1, X2, ..Xn are the entity sets, then a relationship set R is a subset of:

((x1, x2,...,xn) | x1, x2 € X2,........xn € Xn}

where (x1,x2,...xn) is a relationship

For example, consider two entity sets customer and loan. You can define a relationship set to represent the association between customer and loan. Similarly, you can define a relationship set branch-loan to represent relationship between loan and branch entity sets. The association between the entity sets is also known as participation. Therefore, you can say that entity sets E1, E2, ... En participate in relationship set R.

A relationship can exist between two entity sets or more than two entity sets. If the relationship exists between two entity sets, then the relationship is called a binary relationship set. If a relationship 'borrower and loan-branch' exists between customer, loan and branch entity sets, then the relationship can be called as ternary relationship.

The degree of relationship set is the number of entity sets that participate in the relationship. For a binary relationship set the degree of relationship set is two and for a ternary relationship set it is three.

Relationship between two entity sets describes the way in which two different types of entities are related. For example, consider two entity sets employee and manager. The two entity sets are related to each other since the employee works

The various types of relationships that exist between two entities are:

- One-to-one One-to-many
- Many-to-one
- Many-to-many

6.2.1 One-to-One Relationship:

It is a relationship between two entities exists when the occurrence of an entity is related to only one occurrence of the another entity. Figure below shows the one-to- one relationship for the two entity sets Cust and Loan that are related by the relationship set Borrower.

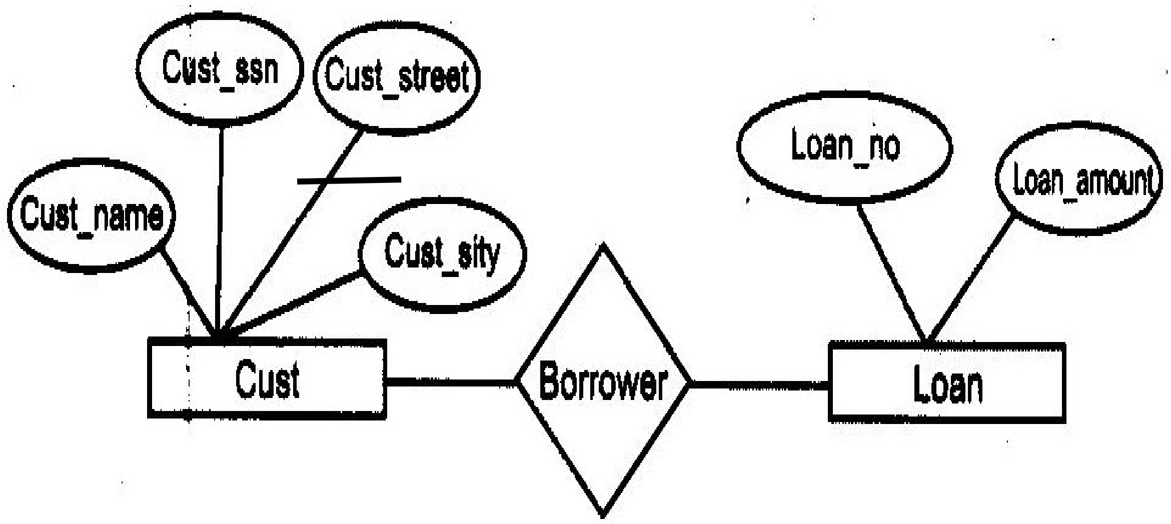


Figure : one-to- one relationship

6.2.2 One-to-Many Relationship:

It is relationship between two entities exists when the occurrence of one entity is associated with the several occurrences of the other entity. For example, consider two entity sets, employee and manager, which are related using working_for relationship set. In this relationship, an employee is working for a single manager. However a manager can have several employees working for him. Similarly, Figure below shows an ERD representing one-to-many relationship.

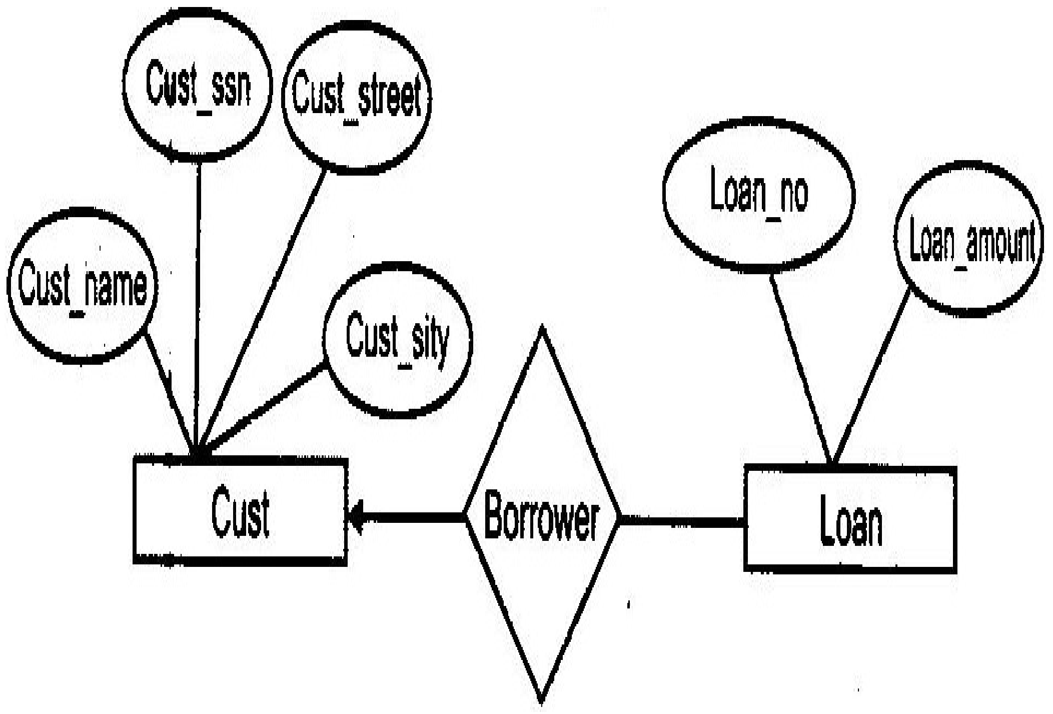


Figure: one-to-many relationship.

6.2.3 Many-to-One Relationship:

It is a relationship between two entities represents that many occurrences of one entity that is associated to a single occurrence of another entity. For example, Figure below shows the many-to-one relationship that exists between the two entity sets, customer and loan.

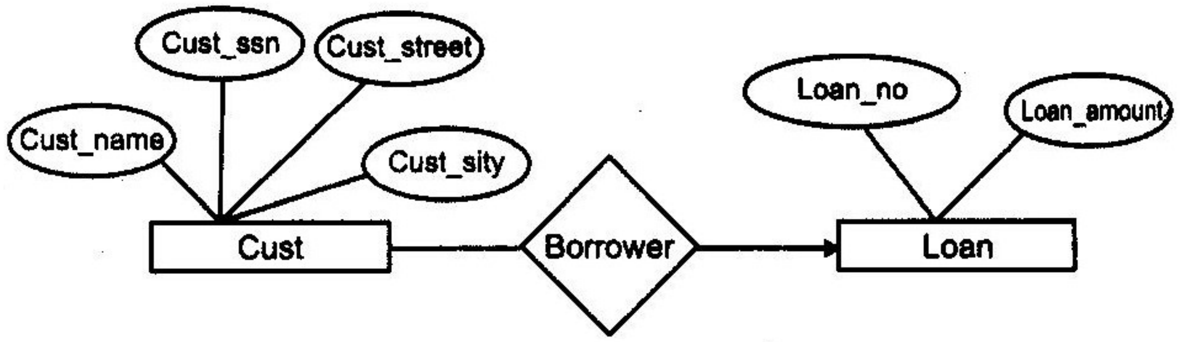


Figure: many-to-one relationship

6.2.4 Many-to-Many Relationship:

It is a relationship between two entities represents that many occurrences of one entity are related to many occurrences of the other entity. For example, the Student and College entity sets are relcollegesated by using the relationship set Study in . In this relationship, a Student is Study in to several . Figure below shows many-to- many relationship existing between the two entity sets.

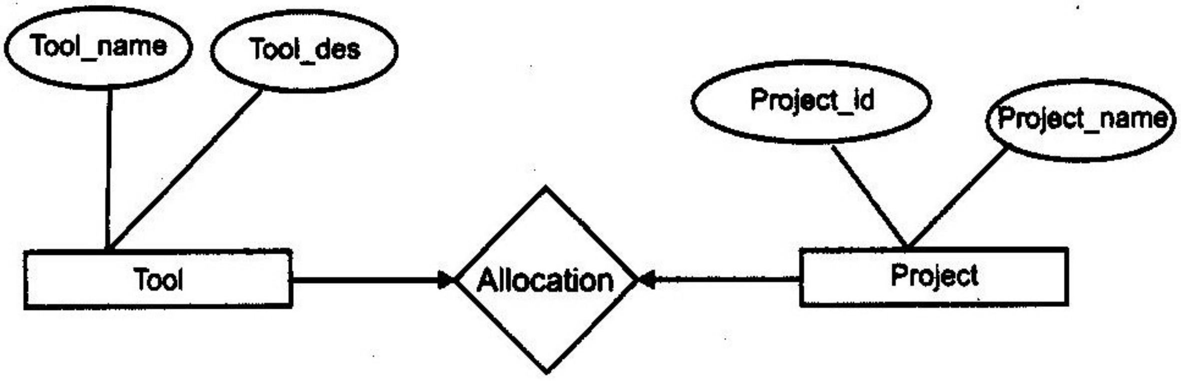


Figure: many-to- many relationship

6.3 SPECIALIZATION AND GENERALIZATION :

Specialization is a process, which defines the sets of subclasses of an entity type and this entity type is called the superclass. The specialization process defines different characteristics of the entities, which helps to categorize the members of superclass into subclass. For example, the subclasses such as TECHNOLOGY STUDENT and MANAGEMENT_STUDENT specialization of STUDENT superclass that distinguishes among the STUDENT entities. The specialization process allows you to perform the following tasks:

- ❖ It defines a set of subclasses of an entity type.
- ❖ It defines specific attributes with each subclass.
- ❖ It defines specific relationship within each subclass and between different subclasses.

Figure below shows that STUDENT is a superclass, which is converted into two different subclasses using specialization process.

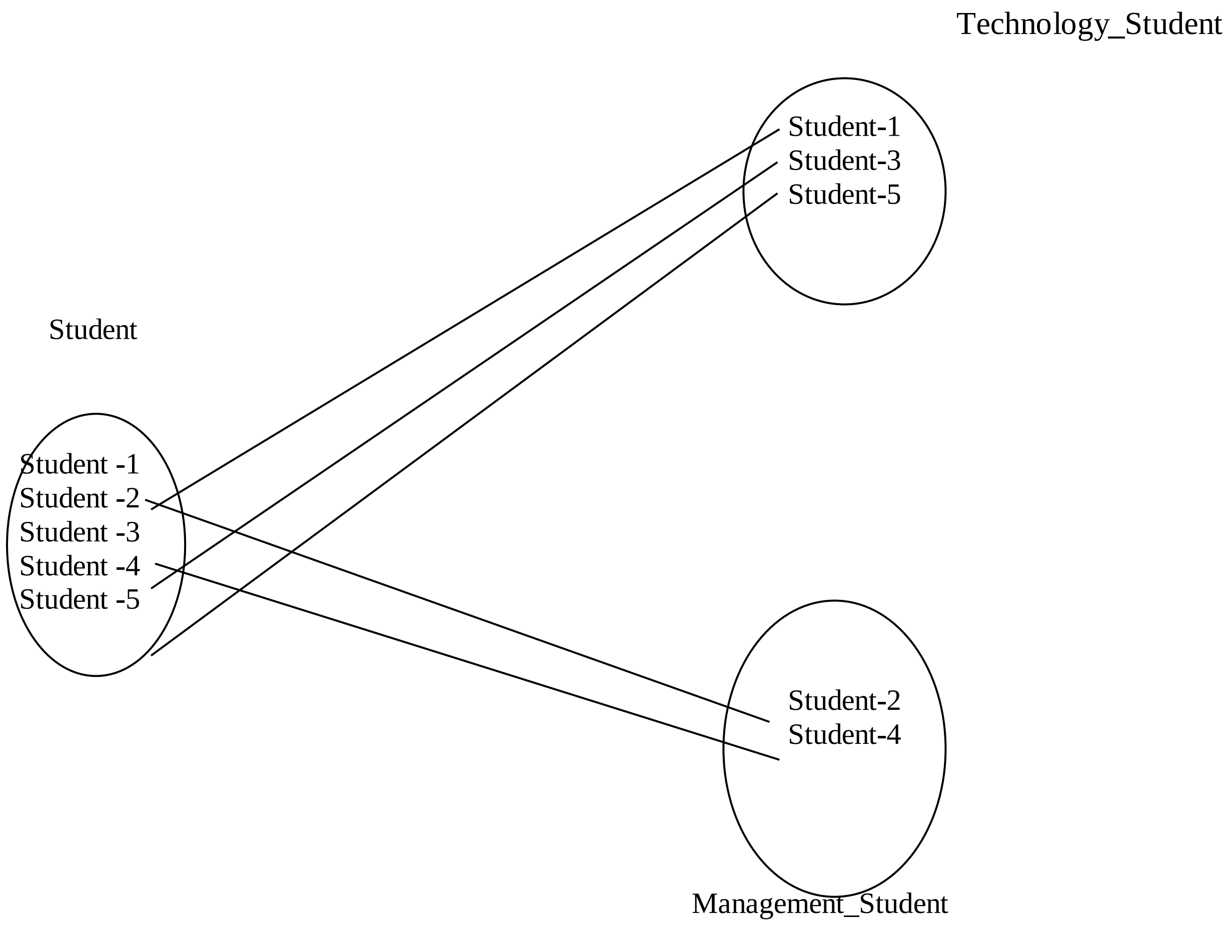


Figure: specialization of STUDENT superclass

Generalization is the reverse process of specialization, in that you combine the various classes by identifying their common features and generalize them into a superclass. For example, CAR and TRUCK are two different entity types, where the CAR entity has attributes such as MAX_SPEED, COST, NO_OF_SEATS and VEHICLE_NO and the TRUCK entity has attributes such as MAX_SPEED, COST, NO_OF_AXIES and VEHICLE_NO. You can generalize the CAR and TRUCK entities according to their common attributes, which are MAX_SPEED, COST and VEHICLE_NO under the

VEHICLE superclass. Figure below shows that VEHICLE is generalized entity of two entities, CAR and TRUCK.

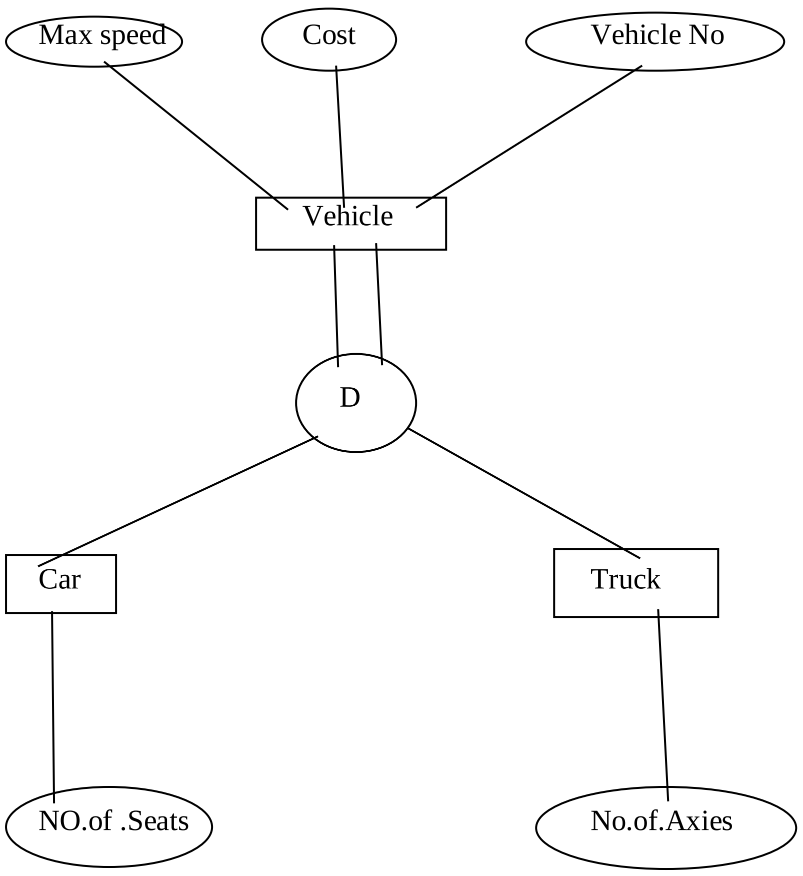


Figure: Generalization of CAR and TRUCK

6.4 CONSTRAINTS ON SPECIALIZATION AND GENERALIZATION:

In general, you may define on the same entity type with several subclasses for a single superclass using specializations concept. Following are the various constraints on specialization and generalization:

- If you determine exactly those entities, which are the members of each subclass by using a common condition, such type of subclasses are called predicate-defined or condition-defined subclasses.
- Condition is a constraint that helps to determine the subclass members.
- You can display a predicate-defined subclass by simply writing condition next to the subclass.
- If all subclasses in a specialization have membership condition on same attribute of the superclass then such a specialization is called as attribute defined-specialization. example, TECHNOLOGY STUDENT and MANAGEMENT STUDENT StudentType is the defining attribute of the specialization of STUDENT.
- If there is no condition to determine for the membership of entity, the subclass is called user-defined.

- Membership in the subclass may be determined by the users of database by applying an operation to add an entity to the subclass.
- Membership in the subclass may be specified by the database users individually for each entity in the superclass.

Following are the two other constraints that may apply to a specialization and generalization:

6.4.1 The disjointness constraint:

This constrains specify that:

- Subclasses defined using specialization must be disjointed. In other words an entity must be a member of atmost one of the subclass in the processes of specialization.
- This constraint in EER diagram is denoted by d.
- If the specialization is not disjointed and may be overlaped, which determines that the same entity may be a member of more than one subclass in the specialization.
- This constraint in EER diagram is denoted by o.

6.4.2 Completeness constraint:

Completeness constraints can be specified by two types that are total and partial:

- Total completeness constraint specifies that every entity in the superclass must be a member of some subclass in the specialization or generalization
- This constraints in EER diagram is denoted by the double line.
- Partial completeness constraint allows an entity not to belong to any of the subclasses.
- This constraints in EER diagram is denoted by a single line.

Aggregation:

Aggregation is a concept for creating composite objects from their component objects. The aggregation allows to use a relationship set as an entity set for the purposes of participation in other relationships Following are the three types of situation in which you can relate this concept to the EER model:

- ❖ **Casel:** This is the situation where you can aggeregate attribute values of an object to form the other object.
- ❖ **Case2:** This is the situation where you represent an aggregation relationship as a simple relationship.
- ❖ **Case3:** This is the situation where you may combine objects that are related by a specified relationship. This is sometimes useful when a higer level aggeregate object is itself to be related to the object.

You may call the relationship between the primitive objects and their aggregate object IS-A-PART-OF and the inverse of it is called the IS-A-COMPONENT- OF.

6.5 COMPOSITION:

The composition represents an integral aspect of aggregation, illustrating the relationship between a whole and its constituent parts. This concept emphasizes the dependency existing between a composite (parent) and its parts (children), signifying that the removal of the composite leads to the deletion of its associated parts. This relationship typically occurs among objects of similar nature.

Here are some key aspects related to composition in DBMS:

1. **Tables:**
   - The fundamental unit of organization in a relational database is a table. This tables consist of rows and columns, where each row represents a record, and each column represents an attribute of that record.
   - Tables are composed of fields or attributes, each defining a specific type of data, such as text, numbers, dates, etc.
2. **Fields/Attributes:**
   - Fields or attributes represent the individual pieces of information stored in a table. For example, in a table of employees, attributes might include employee ID, name, date of birth, and salary.
   - Each field has a data type, specifying the kind of data it can store (e.g., VARCHAR for variable-length character strings, INT for integers).
3. **Data Types:**
   - DBMS supports various data types to accommodate different kinds of information. Common data types include integers, floating-point numbers, strings, dates, and Boolean values.
4. **Keys:**
   - Keys play a crucial role in the composition of a database. The Primary keys uniquely identify each record in a table, ensuring that there are no duplicate records. Foreign keys establish relationships between tables.
5. **Relationships:**
   - In a relational database, tables are often related to each other through keys. The relationships can be one-to-one, one-to-many, or many-to-many. They help maintain data integrity and consistency.
6. **Normalization:**
   - The Normalization is a process used to minimize redundancy and dependency by organizing fields and table of a database. It involves breaking down large tables into smaller, related tables to eliminate data redundancy.
7. **Indexes:**
   - Indexes are structures that provide quick access to data in a table. They improve the speed of data retrieval operations but can also affect the performance of data modification operations.
8. **Views:**
   - Views are virtual tables derived from the result of a SELECT query. They allow users to access specific data without exposing the underlying table structure. Views can simplify complex queries and enhance security.
9. **Stored Procedures and Triggers:**
   - Stored procedures are precompiled and stored in the database. They allow the execution of a set of SQL statements, and they can be reused.
   - Triggers are special kinds of stored procedures that automatically execute in response to specific events, such as data modifications.

Understanding and effectively managing the composition of a database is crucial for designing efficient, scalable, and maintainable database systems. The proper composition ensures data accuracy, integrity, and accessibility.

6.6 SUMMARY :

In the Database Management Systems (DBMS), the concepts of Relationship Degree and Relationship Classification serve as keystones in designing efficient and meaningful database structures.

6.7 KEY WORDS:

- ❖ **One-to-one Relationship:** In a one-to-one relationship, each record in one table is associated with at most one record in another table, and vice versa. It implies a unique and singular correspondence between records in both tables.
- ❖ **One-to-many Relationship:** In a one-to-many relationship, each record in one table can be associated with multiple records in another table, but each record in the second table is associated with at most one record in the first table.
- ❖ **Many-to-one Relationship:** Many-to-one is essentially the reverse of a one-to-many relationship. Multiple records in one table can be associated with at most one record in another table.
- ❖ **Many-to-many Relationship:** In a many-to-many relationship, multiple records in one table can be associated with multiple records in another table, and vice versa.
- ❖ **Associative Relationship:** A type of relationship where entities are connected in a meaningful way but don't represent the same real-world object.

6.8 SELF ASSESSMENT QUESTION:

1. Define a Relationship?
2. Explain Specialization And Generalization in Database Management Systems?
3. What is Composition? Write the Composition in DBMS?

6.9 FUTHUR READINGS :

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LESSON - 7

CODD'S RULES

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ Codd's Rules, proposed by Dr. E.F. Codd, outline a set of principles that define the characteristics of a relational database management system (RDBMS).
- ❖ The objective of these rules is to ensure the consistency, integrity, and reliability of data within a relational database.
- ❖ Codd's Rules are designed to guide the design and implementation of relational databases to achieve certain fundamental properties.

STRUCTURE OF THE LESSON:

7.1 INTRODUCTION:

Edgar F. Codd, a computer scientist, laid the foundation for the theory of relational databases with his groundbreaking work in the early 1970s. Codd's seminal paper, "A Relational Model of Data for Large Shared Data Banks," introduced the concept of a relational database and presented a set of principles known as Codd's Rules. These rules serve as a framework for evaluating the effectiveness and adherence to the relational model within a Database Management System (DBMS).

The simplifying and organizing data management, Codd proposed a set of twelve rules that a database system should follow to be considered truly relational. These rules aim to ensure the consistency, integrity, and flexibility of data representation, retrieval, and manipulation. By adhering to Codd's Rules, a relational database management system can achieve a level of efficiency, reliability, and ease of use that distinguishes it as a robust solution for storing and managing structured information.

This introduction marks the beginning of a journey through the fundamental principles outlined by Codd, each rule contributing to the establishment of a standard for relational database systems. From the systematic treatment of null values to the principles of logical and physical data independence, Codd's Rules continue to shape the design and implementation of modern database systems, playing a crucial role in the development of effective and scalable solutions for managing structured data.

7.2 CODD's Rules:

Codd's rule in DBMS also known as Codd's 12 rules/commandments is a set of thirteen rules (numbered 0 to 12) that classify a database to be a correct Relational Database Management System (RDBMS). If a database follows Codd's 12 rules, it is called a True relational database management system. These rules were originally set out in 1970 by Edgar F. Codd and were developed further by him in 1985.

Rule 0: The Foundation Rule

The DataBase must be structured in a relational manner so that the system's relational capabilities can manage the DataBase.

Rule 1: The Information Rule

The data stored in the database, whether user data or metadata, must must the value of a table cell. Everything in the database should be stored in tabular form.

Rule 2: The Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means can be used to access data.

Rule 3: The Systematic Treatment of NULL Values

NULL values in the database must be handled systematically and consistently. This is a very important rule because a NULL value can be interpreted as missing data, unknown data, or not applicable data.

Rule 4: The Dynamic/Active Online Catalog on the basis of the Relational Model

A description of the structure of the entire database must be stored in an online directory, called a data dictionary, to which authorized users have access. Users can access the directory using the same query language that they use to access the database itself.

Rule 5: The Comprehensive Data Sub-Language Rule

The database can only be accessed using a linear syntax language that supports data definition, data manipulation and transaction management operations. This language can be used directly or through an application. If the database allows access to the data without using this language, it is considered a violation..

Rule 6: The View Updating Rule

All database views that can theoretically be updated must also be updated by the system.

Rule 7: The Relational Level Operation (or High-Level Insert, Delete, and Update) Rule

The database must support high-level inserts, updates, and deletes. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: The Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: The Logical Data Independence

The logical data in a database must be independent of its user’s view (application). Any change in logical data must not affect the applications using it. For example, merging two tables or splitting one of them into two different tables should not have any impact or change in the user's application. This is one of the most difficult rules to apply.

Rule 10: The Integrity Independence

The database must be independent of the application that uses it. All integrity constraints can be changed independently without requiring any changes to the application. This rule makes the database independent of the frontend application and its interface.

Rule 11: The Distribution Independence

The end user cannot see that the data is scattered in different locations. Users should always feel like their data is in one place. This principle is considered the basis of distributed database systems.

Rule 12: The Non-Subversion Rule

If the system has an interface that allows access to low-level data sets, this interface must not be capable of subverting the system and bypassing security and integrity restrictions.

7.3 BASIC DEFINITIONS IN RELATIONAL DATA MODEL:

Data Structure: Data are organized in the form of tables with rows and columns

Data Manipulation: Powerful operations (using the SQL) are used to manipulated data stored in the relations.

Data Integrity: Facilities are included to specify business rules that maintain the integrity of data when they are manipulated.

7.4 RELATIONAL DATA MODEL:

The relational Data Model represents data in the form of tables. The relational model is based on mathematical theory and therefore has a solid theoretical foundational. However, we need only a few simple concepts to describe the Relational Model.

Relational Data Structure:

Relation: A relation is a named, two-dimensional table of data. Table is made up of rows (records), and columns (attribute or field). Not all tables qualify as relations.

The Requirements to a table become as a Relation

1. Every relation has a unique name.
2. Every attribute value is atomic (not multivalued, not composite).
3. Every row is unique (can't have two rows with exactly the same values for all their fields).
4. Attributes (columns) in tables have unique names.
5. The order of the columns is irrelevant.
6. The order of the rows is irrelevant.

Correspondence with ER Model

A. Relations (tables) correspond with entity types.
B. Rows correspond with entity instances.
C. Columns correspond with attributes.

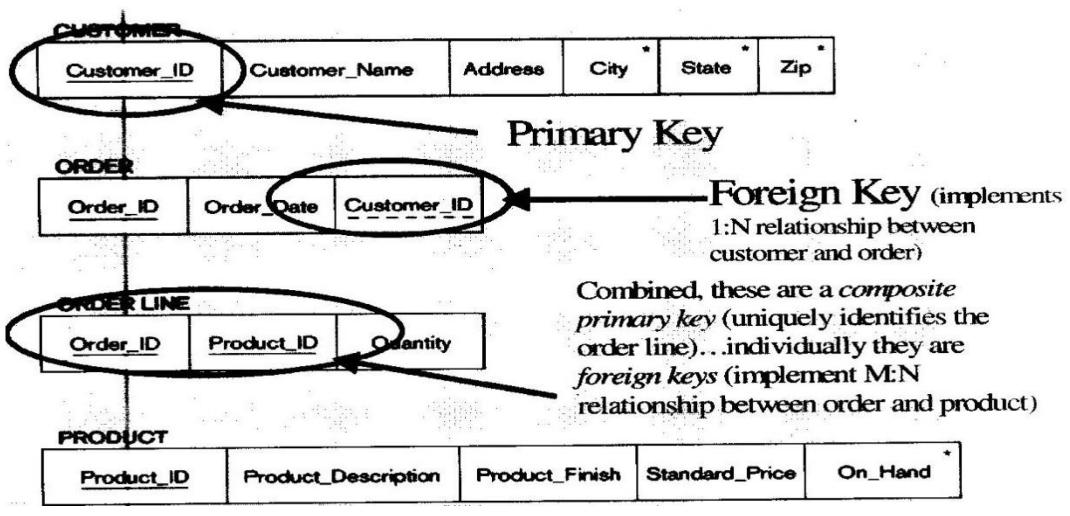
E-R MODEL	**RELATIONS**
Entity type	Table
Entity instance	Record
Attribute	Column

Keys : Keys are special fields that serve two main purposes:

Primary key :Primary keys are unique identifiers of the relation in question. Examples include employee numbers, social security numbers, etc. This is how we can guarantee that all rows are unique.

Foreign Key: Foreign keys are identifiers that enable a dependent relation (on the many side of a relationship) to refer to its parent relation (on the one side of the relationship).

Figure Schema for four relations


CUSTOMER
Customer_ID
Customer_Name
Address
City
State
Zip
Primary Key
ORDER
Order_ID
Order_Date
Customer_ID
Foreign Key (implements 1:N relationship between customer and order)
ORDER LINE
Order_ID
Product_ID
Quantity
Combined, these are a composite primary key (uniquely identifies the order line)...individually they are foreign keys (implement M:N relationship between order and product)
PRODUCT
Product_ID
Product_Description
Product_Finish
Standard_Price
On_Hand

7.5 CONCEPT OF RELATIONAL INTEGRITY:

Relational integrity is a fundamental concept in database management systems (DBMS) that ensures the accuracy, consistency, and reliability of data within a relational database. It is enforced through the use of constraints, which are rules defined on tables to maintain the integrity of the relationships between them. There are two main types of relational integrity: entity integrity and referential integrity.

1. **Entity Integrity:**
    - *Primary Key Constraint:* Every table in a relational database must possess a primary key, a unique identifier for each record in that table. This key guarantees the absence of duplicate records and mandates a valid, non-null value for each record.
2. **Referential Integrity:**
    - *Foreign Key Constraint:* This constraint establishes connections between tables. A foreign key in one table references the primary key in another, creating a relationship between the two. It ensures that values in the foreign key column correspond to values in the primary key column of another table.
    - *Cascade Operations:* Referential integrity often involves specifying actions for referenced record updates or deletions. For instance, cascade updates or deletes can be configured, signifying that alterations in the primary key of one table automatically reflect in the foreign key of another.

The enforcement of these integrity constraints is vital for preventing data inconsistencies and maintaining the validity of relationships between tables over time. If a user attempts to breach these constraints through data insertion, updating, or deletion that jeopardizes integrity, the DBMS will raise an error and reject the operation, thereby preserving the database's reliability.

The relational integrity is a pivotal aspect of database design and management, providing a framework to sustain the accuracy and consistency of data within a relational database. This is accomplished through primary and foreign key constraints, coupled with the enforcement of rules governing table relationships.

7.6 SUMMARY:

Codd's Rules, formulated by Edgar F. Codd, establish a set of principles that define the characteristics of a true relational database management system (RDBMS). These rules aim to ensure the consistency, integrity, and flexibility of data within a relational database. The twelve rules cover various aspects of database design and management, including the organization of information, access mechanisms, handling of null values, and the independence of data representation from application programs. Adherence to Codd's Rules results in databases that are more reliable, maintainable, and provide a high level of data integrity. The rules address key aspects such as comprehensive data sublanguages, physical and logical data independence, distribution independence, and the systematic treatment of null values. Following Codd's Rules guides the development of relational databases, contributing to the creation of robust and effective systems for storing and managing structured data.

7.7 KEY WORDS:

- **Table (Relation):** In the relational model, data is organized into tables, also referred to as relations. Each table is a two-dimensional structure with rows and columns. Each row in a table represents a record, and each column represents an attribute or field.
- **Row (Tuple):** A row in a table represents a single record or data instance. Each row contains values for each attribute defined in the table.
- **Column (Attribute):** A column in a table represents a specific attribute or characteristic shared by all records in the table. It defines the type of data that can be stored in that column.
- **Primary Key:** A primary key is a unique identifier for each record in a table. It ensures that each row in a table can be uniquely identified.
- **Foreign Key:** A foreign key is a column in a table that refers to the primary key of another table. It establishes a relationship between two tables.
- **Normalization:** The relational model supports the concept of normalization, which involves organizing data to minimize redundancy and dependency. Normalization helps in maintaining data integrity and reducing the likelihood of data anomalies.

7.8 SELF ASSESSMENT QUESTIONS:

1. Explain the relational Data Model?
2. Discuss the relational data structure?
3. Write the concept of relational integrity?

7.9 FURTHER READINGS :

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000

LESSON – 8

INTRODUCTION-HISTORY OF SQL STANDARDS

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ It is provide the readers with a foundational understanding of the database management system (DBMS) and its pivotal role in organizing and managing data.
- ❖ It sets the stage for exploring the SQL (Structured Query Language) standards by establishing the context of their significance in database management.
- ❖ The objective is to familiarize readers with the fundamental concepts of DBMS, emphasizing its importance in modern information systems.

STRUCTURE OF THE LESSON:

8.1 INTRODUCTION:

The objective of a database is to store the data in such a manner that it can be easily accessed and altered by the user. DBMS uses a database language to store and retrieve the data in a database. The database languages also allow the database administrator (DBA) to retrieve, update and remove the data from the database. The DBA uses SQL to maintain the database. It is a database language used for accessing and managing the data from the database. It defines various methods to create and manipulate the relational databases on different platforms. It is used with different database applications such as Microsoft Access, Microsoft SQL Server, Oracle and Sybase. The Programming Language SQL (PL/SQL) is a development tool that extends the features provided in the SQL database language. The PL/SQL allows you to provide the flow control and logic design to the unstructured SQL command blocks.

Commands in SQL:

Oracle is a Relational Database Management System (RDBMS). Oracle being RDBMS, stores data in tables called relations. These relations are two-dimensional representation of data, where rows called tuples represent records and columns called attributes represent pieces of information contained in the record.

Oracle provides a rich set of tools to allow design and maintenance of the database. Major tools are,

RDBMS Kernel	Database Engine
SQL	: Structured Query Language
SQL* PLUS	: Addition to SQL
PL/SQL	: Procedural Language SQL, allows Procedural processing of SQL statements
SQL * DBA	: Database Administrator's tool set
DEVELOPER 2000	: ORACLE'S GUI tool for Forms.

8.2 THE SQL ENVIRONMENT:

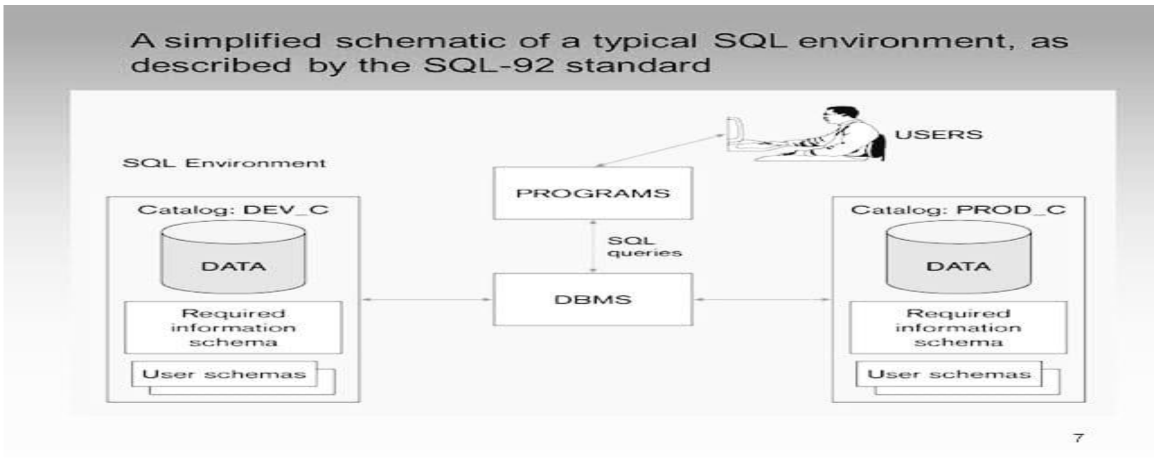
An SQL environment, It includes an instance of a relational DBMS along with the actual databases which are accessible by the DBMS. The environment supports multiple databases, such as a production (live) and a development (for debugging new programs) database. Each database is contained in a catalog which describes all the objects in the database (for all schemas). Each database has a named schema(s) associated with a catalog.

8.3 DEFINING A DATABASE IN SQL

The schema is a collection of related objects created by the user, such as tables, views, domains, constraints, triggers, etc. In other words, it is everything you create as it relates to a particular database.

The diagram below shows SQL environments for two catalogs one used for development purposes, and the other the actual production level DBMS (live version with real data and all monitoring and security protection)

Note that information in the DBMS is created and maintained using SQL commands. The use of simple SQL commands will be used to carry out complex operations on the database. We will study how this is done over the next two Lecture Sets.


A simplified schematic of a typical SQL environment, as described by the SQL-92 standard
USERS
SQL Environment
PROGRAMS
Catalog: DEV_C
DATA
Required information schema
User schemas
SQL queries
DBMS
Catalog: PROD_C
DATA
Required information schema
User schemas
7

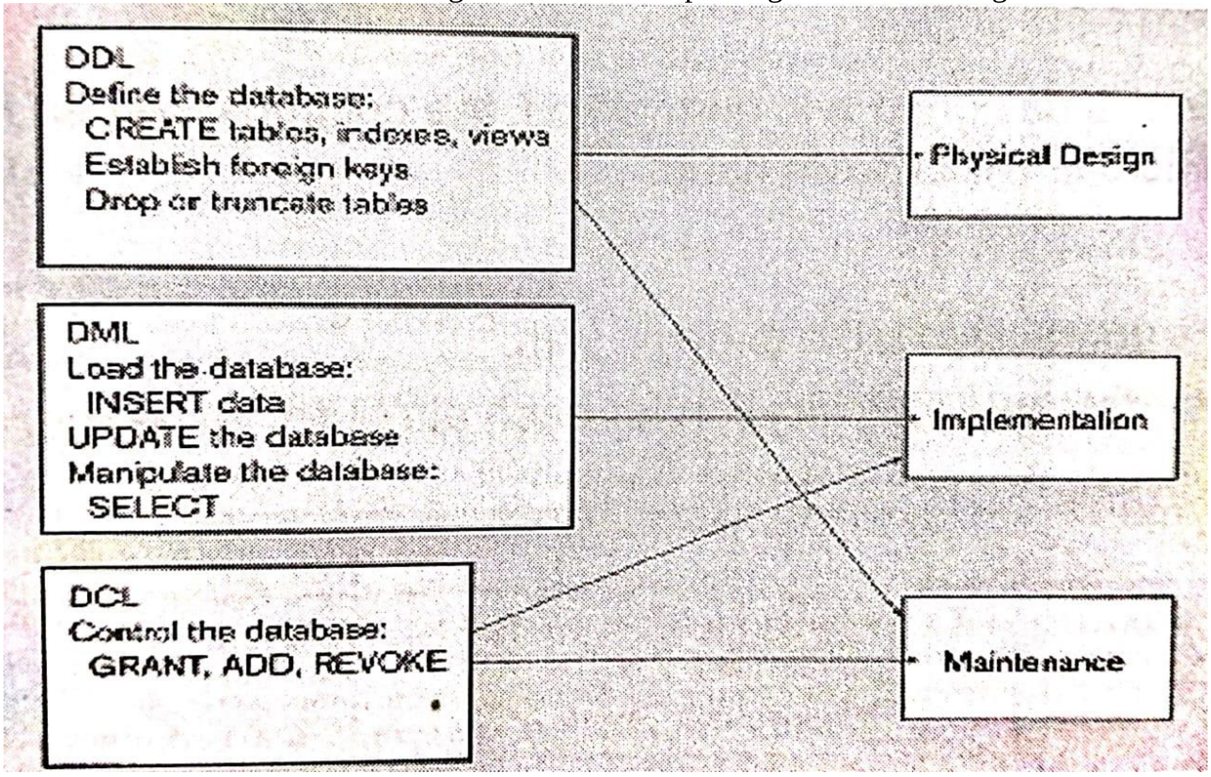
8.4 GENERATING SQL DATABASE DEFINITIONS:

SQL commands are classified into three types:

Data definition language (DDL) commands: Are used to create, alter and drop tables and views.

Data manipulation language (DML) commands: Are used for inserting, updating, modifying and querying data in the database. DML commands may be issued interactively or imbedded within programs written in 3GL languages such as C or COBOL.

Data control language (DCL) commands: Are used by the DBA (database administrator) to con- trol the database such as grant or revoke privileges for accessing the database.



Generating SQL Database Definitions

Three SQL DDL CREATE commands are included in Entry-Level SQL-92.

CREATE SCHEMA used to define that portion of the database that a particular user owns. Schemas are dependent on a catalog, and contain schema objects including tables, views, domains and constraints etc

CREATE TABLE Defines a new table and its columns. Tables are dependent on the schema. Table can be a base table or a derived table. Tables are dependent on schema and are created by executing an SQL query that creastes a TABLE (rather than a query)

CREATE VIEW Defines a logical table from one or more tables or views.

DROP TABLE will destroy a table including its name, definitions and content. The DROP com- mand pretty much acts as the inverse of opposite of CREATE.

DROP SCHEMA and DROP VIEW - Used to destroy schemas and views.

ALTER TABLE may be used to change the definition of an existing table by adding, dropping, or changing a column or dropping a constraint.

There are 5 other create commands in the base SQL-92 language standard. Your SQL could meet the Entry and Intermediate level SQL-92 standard without them.

CREATE CHARACTER SET - This is especially useful for languages other than English
CREATE COLLATION - Specifies the order that a character set assumes. Does not have to be ASCII. Each character in a character set is assigned a numeric value that determines its collating order.

CREATE TRANSLATION - Maps one character set to another in case of translations for ex- ample, English to French, or ASCII to EBCDIC, etc..

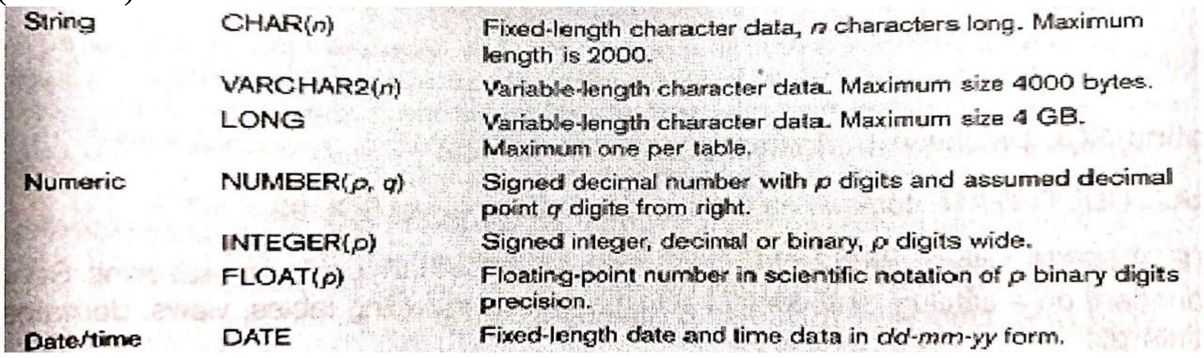
CREATE ASSERTION - A schema object that can be used to create new CHECK constraints.

CREATE DOMAIN - A schema object that establishes a domain or set of valid values for an at- tribute. The domain can consist of a data type, default value, collation, or other constraint.

8.5 CREATING TABLES:

Once the data model is designed and normalized, the columns needed for each table can defined using the SQL CREATE TABLE command.

Steps for creating a table:
Be sure you have a normalized table to begin with.
Provide a complete data type specification for each attribute, especially length and precision (if needed)



String	CHAR(*n*)	Fixed-length character data, *n* characters long. Maximum length is 2000.
	VARCHAR2(*n*)	Variable-length character data. Maximum size 4000 bytes.
	LONG	Variable-length character data. Maximum size 4 GB. Maximum one per table.
Numeric	NUMBER(*p, q*)	Signed decimal number with *p* digits and assumed decimal point *q* digits from right.
	INTEGER(*p*)	Signed integer, decimal or binary, *p* digits wide.
	FLOAT(*p*)	Floating-point number in scientific notation of *p* binary digits precision.
Date/time	DATE	Fixed-length date and time data in *dd-mm-yy* form.

Be sure you know which columns can be NULL and which cannot. Note that once it is established it is enforced for all data.

Identify those columns that have to be UNIQUE — that is, no two table instances can have the same data value for such a column. Each such column is a candidate key (but only one can be chosen as PRIMARY KEY). UNIQUE and PRIMARY KEY are called column constraints.

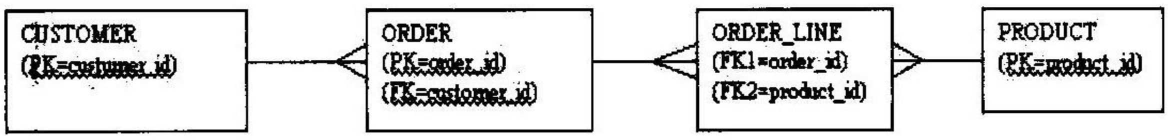
Identify all foreign key-primary key pairs (either as table is created or later, by adding a constraint. The "parent table should be created first so that the referencing or child table can reference an existing table when it is created. REFERENCES can be used to enforce referential integrity.

Determine default values those values to be used in a particular column when no valueis entered for an instance of an entity.

Identify any columns for which domain constraints need to be tightened or stronger than given. You can use CHECK to place a tighter column constraint. Note that "Spruce" or "White Maple" are not legal PRODUCT_FINISH values as shown in the table example a little further down. All legal values are included in the specified set of values.

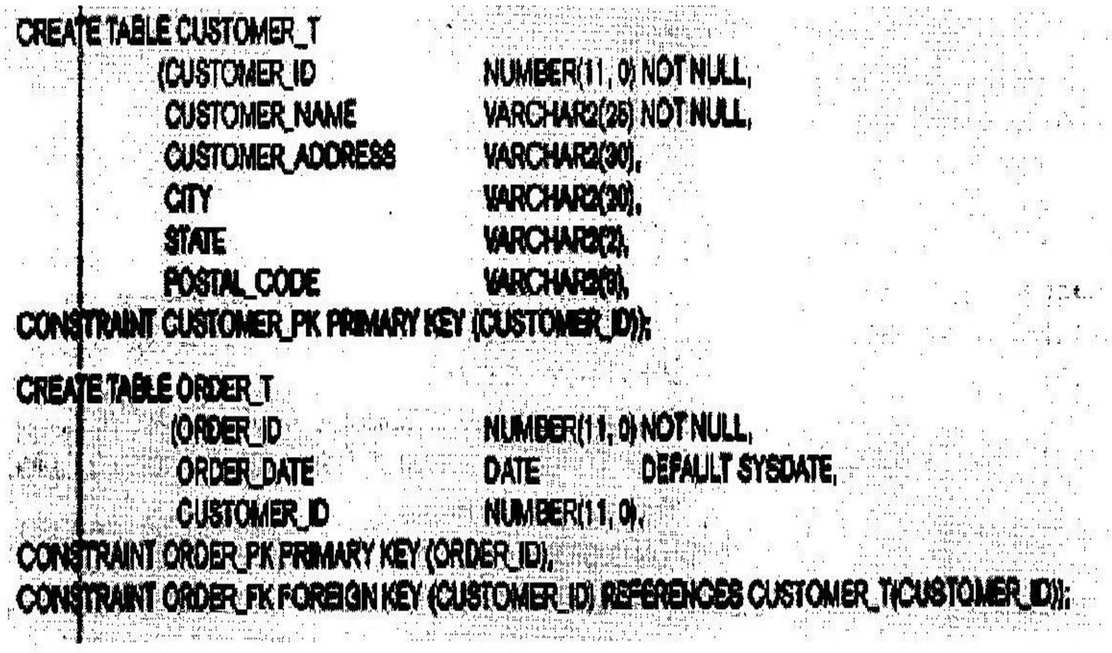
Create the table and any desired indexes using the CREATE TABLE and CREATE INDEX statements.

Example:

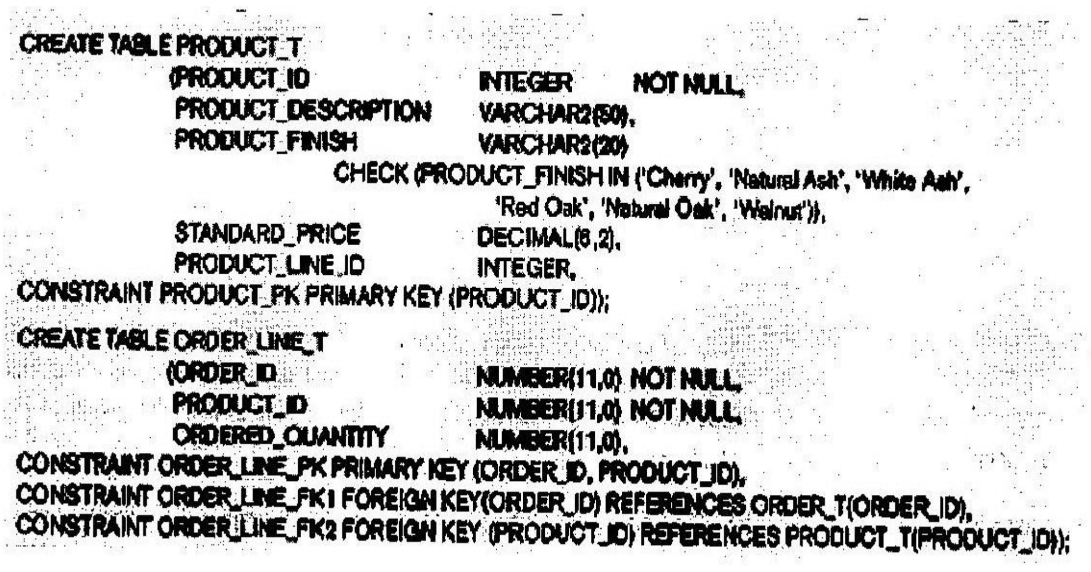


Let's create tables for the ERD defining the relations between CUSTOMER, ORDER, ORDER_LINE and PRODUCT.

PK = Primary Key, FK = Foreign Key.



```
CREATE TABLE CUSTOMER_T
        (CUSTOMER_ID            NUMBER(11, 0) NOT NULL,
         CUSTOMER_NAME          VARCHAR2(25) NOT NULL,
         CUSTOMER_ADDRESS       VARCHAR2(30),
         CITY                   VARCHAR2(20),
         STATE                  VARCHAR2(2),
         POSTAL_CODE            VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
         (ORDER_ID              NUMBER(11, 0) NOT NULL,
          ORDER_DATE            DATE          DEFAULT SYSDATE,
          CUSTOMER_ID           NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));
```



```
CREATE TABLE PRODUCT_T
           (PRODUCT_ID              INTEGER        NOT NULL,
            PRODUCT_DESCRIPTION     VARCHAR2(50),
            PRODUCT_FINISH          VARCHAR2(20)
                     CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                            'Red Oak', 'Natural Oak', 'Walnut')),
            STANDARD_PRICE          DECIMAL(6,2),
            PRODUCT_LINE_ID         INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
           (ORDER_ID                NUMBER(11,0) NOT NULL,
            PRODUCT_ID              NUMBER(11,0) NOT NULL,
            ORDERED_QUANTITY        NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Note that names such as those with extensions _FK1 and PK are useful aliases to give to column names. If you ever have occasion to read a constraint table, these names will help make the table more readable.

8.6 USING AND DEFINING VIEWS:

A view is like a window through which data on tables can be viewed or changed. A view is derived from another table or view, which is referred as the base table. A view is stored as a SE- LECT statement only but has no data of its own. It manipulates data in the underlying base table. Syntax: CREATE [OR REPLACE] VIEW view_name[col1,col2, . . .]

AS SELECT statement;

Ex 1: create view v_dept30 as select * from emp where deptno-30;

Ex 2: create view dept_summary(name,min_sal,max_sal,avg_sal)

as select dname,min(sal),max(sal), avg(sal) from emp e,dept d

where e.deptno=d.deptno group by dname;

View text can be viewed using a data dictionary view USER_VIEWS.

Views can be deleted using DROP command.

Syntax,

DROP VIEW view_name;

8.7 CREATING DATA INTEGRITY CONTROLS:

Constraints: Constraint is a rule that can be applied on a table or a column of a table. Con- straints can also be used to prevent mistakes in data entry. Constraints can be categorized into following,

Entity Integrity Constraints.

- Primary key
- Unique

Domain Integrity Constraints

- Not null
- Check

Referential Constraint

Constraints can be attached to a table or column in two ways as follows,

Table Constraints: Defining separately from definitions of columns in the table for one or more columns (using ALTER command).

Column Constraints: Defining within the definition of columns for single column (within CRE- ATE command).

Primary Key Constraint: A key field, which can be used to identify records (entity values) uniquely. A key field can be a composite. A table may contain more than one key fields, but one of the key fields can be treated as primary key field.

Syntax:

Table Constraint:

[CONSTRAINT constraint_name] PRIMARY KEY(column, column,...)

Ex: alter add primary key(deptno) ...

Column Constraint:

[CONSTRAINT constraint_name] PRIMARY KEY

Ex: create ... deptno number(3) primary key,...

Unique Constraint: Column with UNIQUE constraint specifies that all entities having different (unique) values for that column.

Syntax:

Table Constraint:

[CONSTRAINT constraint_name] UNIQUE(column,column,..)

Ex: alter... add unique(ssno) ...

Column Constraint:

[CONSTRAINT constraint_name] UNIQUE

Ex: Create ... Ssno varchar2(8) unique, . . .

Null / Not Null Constraint: This constraint can be defined only as Column constraint. If a column has NOT NULL constraint, null values are not permitted (i.e. It would not allow you to omit value for that column). Columns without NOT NULL, is referred as NULL (i.e. it would allow you to omit value for that column if necessary is treated as null value).
EX: create ... eno number(3) not null, apt_number varchar2(10), ...

Check Constraint: This constraint is for defining a condition that each row must satisfy. Same Syntax for both table constraint and column constraint.

Syntax:

[CONSTRAINT constraint_name] CHECK(condition)

Ex 1: create eno number(3) check(eno>0 and eno <=500), ...

Ex 2: alter table emp add constraint ck_emp check(sal + comm <10000);

Foreign Key Constraint: It provides referential integrity rules either within a table or between table. This can be used in relationship with either a primary or unique key elsewhere. Usually foreign key of one table (called child table ex. emp) refers primary key of another table (called parent table ex. dept).

Syntax:

Table Constraint:

[CONSTRAINT constraint_name] FOREIGN KEY (column, column, . . .) REFERENCES table_name(column,column, ...) [ON DELETE CASCADE]

Ex: alter ... add constraint fk_dno foreign key(dno) references dept(dno);

Column Constraint:

[CONSTRAINT constraint_name] REFERENCES table_name(column,column, . . .)

Ex: create table dept(dno number(3) primary key, dname varchar2(15));

Ex: create table emp(eno number(3) primary key, . . . dno number(3) references dept(dno));

As the result of FOREIGN KEY constraint, a record in parent (ex.dept) could not be deleted if rows exists in child (ex.emp) with the same value of reference field. The ON DELETE CASCADE is the option can be appended to FOREIGN KEY syntax (only for table constraint), and can be used to delete child automatically if parent is deleted.

...

Ex: alter add constraint fk_dno foreign key (dno) references dept (dno) on delete cas- cade;

8.8 CREATING TABLE DEFINITIONS:

Table definitions may be changed by ALTERing column specifications.

ALTER TABLE command may be used to add new columns to an existing table.

We can use three commands to alter tables, ADD, DROP, and ALTER. For example, to an an attribute (column) such as a cell phone number to a table, we can specify

ALTER TABLE CUSTOMER_T

ADD (cell phone VARCHAR (10));

←.

a new attribute (cell phone) is added

Note that this add has no impact on any of the code (queries and reports) written except those in which this attribute is to be printed. All others remain unchanged. This would not likely be the case on old file-based systems.

This command is invaluable for the always expected modifications that have to be made to databases after they have been created. (As invaluable as it is, it is easier to add an attribute from the design view of a table.)

Note that the attribute is added to all instances of the table. Normally, when adding an attribute, we want to allow the values to be NULL at least until we have entered the values for all table instances.

8.9 REMOVING TABLE:

DROP TABLE customer_t;

(Note, the Security privileges (GRANT/REVOKE) are not implemented in Access. The limited security of Access is created using the Tools/Security menu.

The DROP command can only be executed by a person that was granted the DROP ANY TABLE system privilege. Furthermore, to maintain data integrity the DROP may be qualified by the RESTRICT or CASCADE. If restrict is specified, the command fails if there are any dependent objects such as views or constraints that reference the table. If CASCADE is specified, all depen- dent objects are removed.

8.10 INSERTING, UPDATING & DELETING DATA:

Once tables and views have been created, it is necessary to populate them with data and maintain those data before queries can be written. The INSERT command which is used add the records into table.

INSERT INTO customer_t

VALUES (001, 'Contemporary Casuals', '100 Main St.', 'Phila', 'PA', '19111);

(F all data attributes are added.)

INSERT INTO customer_t (customer_id, customer_name)

VALUES (100, 'New customer'); only 2 attributes are entered

INSERT INTO ca_customer_ SELECT FROM customer_t

WHERE state = 'CA' t

(Populating a table from another table (not in Access)

INSERT INTO order_t (order_id, customer_id, order_date)

VALUES (123, 100, '25-jan-01') ← entering date

Note that data values to be inserted in the same order as the columns appear in the table. If only some fields are to be defined by the insert, NULL values can be used for the others or you can specify the particular fields to be filled in (for the indicated attribute). When fields are specified, they should be in the same order as they appear in the table and the data must also be in that order

8.11 BATCH INPUT:

The INSERT command is used to enter one row of data at a time or to add multiple rows as the result of a query. Some versions of SQL have a special command or utility for entering multiple rows of data as a batch: the INPUT command. Oracle includes a program, SQL*Loader, which runs from the command line, and can be used to load form a file into the database. This popular program is tricky to use and is not included in the scope of this taxt.

Deleting Database Contents

```
UPDATE product_t
        SET unit_price = 775.00
                WHERE product_id = 7;
```

: It allows you to remove one or more rows from a table.

```
        DELETE FROM table_name [WHERE condition];
DELETE FROM customer_t           ← deleting rows that meet a certain criterion
        WHERE state = 'PA';
DELETE FROM customer_t;                  ← All rows are deleted
```

Changing the Database Contents

UPDATE: It allows you to change values of rows in a table.

```
        UPDATE table_name SET column=expression
        [WHERE condition];
UPDATE product_t
        SET unit_price = 775.00
                WHERE product_id = 7;
```

8.12 INTERNAL SCHEMA DEFINITION IN RDBMSS:

The internal schema of a relational database can be controlled for processing and stor- age efficiency. The ISO standard does not address most efficiency issues so a number of these definitions are not available in all DBMS systems.

Some of the things we can do to tune a DB are indicated below

Creating Indexes

We can index primary of secondary keys to increase the speed of row selection, table joining or row ordering.

This in turn can provide more efficient sequential and random access to base table data.

CREATE INDEX name_idx ON CUSTOMER_T (customer_name);

It is important to note that creating indexes takes up more table space so you want to carefully choose the keys you decide to index. Overhead is also required every time and indexed attribute value is changed.

Dropping Indexes

Dropping an index (to increase the speed of table updates):

DROP INDEX name_idx

8.13 DATA TYPES IN SQL:

In SQL (Structured Query Language), data types are define the type of data that a particular column or variable can hold in a database. Different database management systems (DBMS) may have slightly different implementations of data types, but there are some common types that are widely used. The following categories are the some data types in SQL:

1. **Numeric Types:**
   - **INTEGER or INT:** Used for whole numbers.
   - **SMALLINT:** Similar to INT but with a smaller range.
   - **BIGINT:** Used for larger whole numbers.
   - **DECIMAL or NUMERIC:** Used for fixed-point numbers.
   - **FLOAT and REAL:** Used for floating-point numbers.
2. **Character String Types:**
   - **CHAR(n):** Fixed-length character string with a specified length 'n'.
   - **VARCHAR(n):** Variable-length character string with a maximum length of 'n'.
   - **TEXT:** Variable-length character string with no specified maximum length.
3. **Date and Time Types:**
   - **DATE:** Stores date values in the format YYYY-MM-DD.
   - **TIME:** Stores time values in the format HH:MM:SS.
   - **DATETIME or TIMESTAMP:** Stores both date and time values.
4. **Boolean Type:**
   - **BOOLEAN or BOOL:** Represents true or false values.
5. **Binary Types:**
   - **BINARY(n):** Fixed-length binary string with a specified length 'n'.
   - **VARBINARY(n):** Variable-length binary string with a maximum length of 'n'.
   - **BLOB (Binary Large Object):** Variable-length binary data, used for storing large amounts of binary data like images or documents.
6. **Miscellaneous Types:**
   - **ENUM:** Represents a set of predefined values.
   - **SET:** Similar to ENUM but can store multiple values from a predefined set.
   - **JSON:** Stores JSON data.
   - **XML:** Stores XML data.

7. **Spatial Types:**
   - **GEOMETRY, POINT, LINESTRING, POLYGON:** Used for storing spatial data, such as geographical information.

When creating a table in SQL, you specify the data type for each column, indicating the kind of data that column can store. For example:

```
CREATE TABLE Employees (

   EmployeeID INT,

   FirstName VARCHAR(50),

   LastName VARCHAR(50),

   BirthDate DATE,

   Salary DECIMAL(10, 2)

);
```

In the above example, EmployeeID is of type INT, FirstName and LastName are of type VARCHAR, BirthDate is of type DATE, and Salary is of type DECIMAL. The choice of data type depends on the nature of the data you want to store and the operations you want to perform on that data.

8.14 SUMMARY :

The Database Management Systems (DBMS) and their crucial role in organizing and managing data. It serves as a gateway to understanding the significance of SQL standards in the context of database management. Key themes include the importance of DBMS in modern information systems and the necessity of a structured approach to querying and manipulating data.

8.15 KEY WORDS :

- **Structured Query Language (SQL):**A standardized programming language used for managing and manipulating relational databases. SQL allows users to define, query, update, and control data within a database.
- **SQL Standardization:** The process of establishing and maintaining a set of rules and guidelines to ensure uniformity and compatibility in the implementation of the Structured Query Language across different database management systems.
- **Evolution of SQL:** The gradual development and refinement of SQL over time, marked by the introduction of new features, improvements, and modifications to address emerging needs and challenges in database management.

8.16 SELF – ASSESSMENT QUESTIONS:

1. Define A Database In SQL?
2. How to Create Data Integrity Controls?
3. Explain Batch Input?
4. Write the different Data Types In SQL?

8.17 FURTHER READINGS:

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000

LESSON – 9

DATA DEFINITION LANGUAGE (DDL)

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ The objectives of DDL in DBMS revolve around defining, managing, and maintaining the structure of the database to ensure data accuracy, integrity, security, and adaptability to changing business needs.

STRUCTURE OF THE LESSON:

9.1 INTRODUCTION:

The structural and organizational aspects of data are crucial for effective data storage, retrieval, and manipulation. This is where Data Definition Language (DDL) plays a pivotal role. DDL is a subset of SQL (Structured Query Language) responsible for defining and managing the structure of a database.

Unlike Data Manipulation Language (DML), which focuses on the manipulation of data within the database, DDL is concerned with the definition of the database schema. In simpler terms, DDL is the language used to specify the database schema or the blueprint that outlines how data is organized, stored, and related to other data within the database.

The primary functions of DDL include the creation, modification, and deletion of database objects such as tables, indexes, views, and constraints. Through DDL commands, database administrators and developers can establish the foundation upon which data operations will be conducted. This involves specifying data types, defining relationships between tables, setting constraints to ensure data integrity, and establishing the overall framework that governs the database structure.

DDL commands are powerful tools that enable users to control the fundamental aspects of a database, shaping it according to the requirements of the organization and the applications it serves. The precision and accuracy with which DDL statements are crafted significantly influence the efficiency, reliability, and security of the database system.

The role of DDL is to provide a means for users to articulate the architecture of a database, ensuring that it aligns with the intended data model and supports the functional requirements of the applications relying on it. As a cornerstone of database management, DDL empowers database administrators and developers to orchestrate the foundational elements of a database, laying the groundwork for efficient data storage, retrieval, and management.

RELATIONAL LANGUAGES:

Relational languages are used by the DBA and the database designers to specify the conceptual and internal schemas for the database. The relational languages can be divided into three categories. These categories are:

Data Definition Language (DDL): It is used by the DBA to define the internal as well as the conceptual schema of the database. DDL is used to specify the database schema as a set of definitions. Each DBMS has a DDL compiler that is used to compile the DDL statements to produce the schema description. This schema description is then stored in the special DBMS catalog file, which is also called data dictionary or data directory. This data dictionary contains the meta data that is the data about data. Extensible Markup Language (XML) is the best example of DDL. In other words, DDL is used to define the structure of the database and creates the objects in the database. The database objects can be tables, views, stored procedures and triggers. However the number of database objects depends up on the DBMS used. The SQL also provides certain commands that come under the category of the DDL. These commands are:

- ❖ CREATE statements
- ❖ DROP statements
- ❖ ALTER statements:

Storage Definition Language (SDL): It is used to create the internal schema of the database. This means, SDL is used to describe the physical storage structure of the database. It is also used to specify the complete details of the data storage and access paths for the database. Data Manipulation

Language (DML): It is the language of the database that is used after the database schema has been compiled by DDL compiler and the database is filled with data. This language

9.2 OPERATIONS IN RELATIONAL ALGEBRA:

In relational algebra, some operators are primal and the others can be defined exclusively in terms of the primal ones are not primal. Some of the primal operators are as follows:

- ❖ Selection
- ❖ Projection
- ❖ Cartesian product (also cross product or cross join)
- ❖ Set union
- ❖ Intersection
- ❖ Set difference or complement
- ❖ Rename

These seven operators are primal in the sense that if you omit one of them you will certainly lose expressive power.

9.3 SELECTION OPERATION AND PROJECTION OPERATION:

SELECTION OPERATION:

In relational algebra, a selection (or restriction) is a unary operation (an operation with only one operand) written as $\sigma_{a\theta b}$ (R) or $\sigma_{a\theta\, v}$ (R)

Where

a and b are the names of the attribute

O is a binary operation over the set {<, >, =, >, <}

v is a value constant.

R is a relation.

The selection Gob (R) selects all those tuples in R for which lies between the a and the b attribute.

The selection av (R) selects all those tuples in R for which lies between the a attribute and the value v.

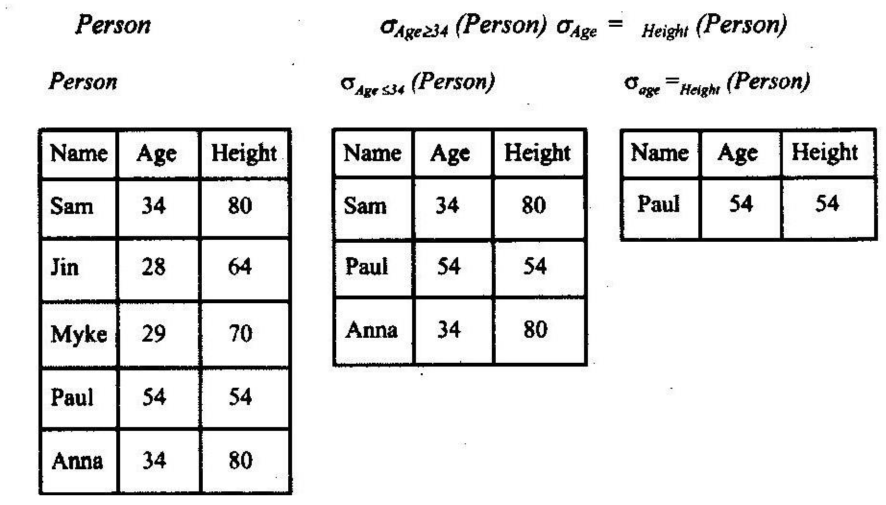
For example, consider the following tables in Figure below , where the first table gives the relation Person, the second table gives the result of OAge≥34 (Person) and the third table gives the result of OAge Height (cm)(Person).

=

A more formal definition of the selection is given as follows:

$\sigma_{a\,\theta b}$ (R) = {t:t∈ R, t(a) θ t (b)}

$\sigma_{a\theta v}$ (R)= {t:t∈R, f(a) θ v}



Person $\sigma_{Age\geq 34}$ *(Person)* σ_{Age} = $_{Height}$ *(Person)*

Person

Name	Age	Height
Sam	34	80
Jin	28	64
Myke	29	70
Paul	54	54
Anna	34	80

$\sigma_{Age\leq 34}$ *(Person)*

Name	Age	Height
Sam	34	80
Paul	54	54
Anna	34	80

$\sigma_{age} =_{Height}$ *(Person)*

Name	Age	Height
Paul	54	54

Figure: An example of selection operation

Projection Operation:

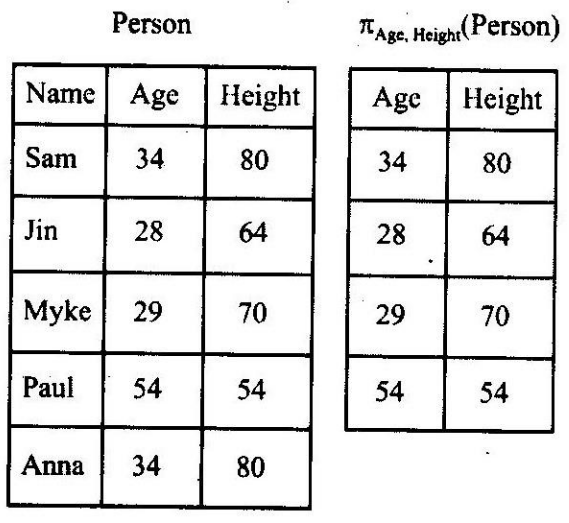
In relational algebra, a projection is a unary operation written as $\pi_{a_l,...,a_n}(R)$, where

$A_1,...,a_n$ is a set of attribute names. The result of this kind of projection is defined as the set

that is obtained, when all tuples in R are restricted to the set $\{a_i,...,a_n)$.

For example, consider the relations listed in the following two tables of Figure below. This below figure contains two relations that are Person and the relation that has been formed by

applying the projection operation on the attributes age and height.



Person

Name	Age	Height
Sam	34	80
Jin	28	64
Myke	29	70
Paul	54	54
Anna	34	80

$\pi_{Age, Height}$(Person)

Age	Height
34	80
28	64
29	70
54	54

Figure: An example of projection

A more formal definition of the projection is given as follows:

$\pi_{al.......an}$, (R) = $\{t[a_1,....a_n]: t \in R\}$

Where t $[a_1,..., a_n]$ is the selection of the tuple t to the set $\{a_1,...,a_n\}$ so that:

t $[a_1,...., a_n] = \{(a',v) \mid (a',v) \in t, a' \in a_1,..., a_n\}$

9.3.1 Cartesian product:

Cartesian product (or direct product) of two sets -X and Y, denoted X x Y, is the set of all possible ordered pairs whose first component is a member of X and second component is a member of Y. That is:

$$XxY = \{(x, y)\ x\in X \text{ and } y \in Y\}$$

An ordered pair is a collection of two items; such that one can be separated as the first element and the other as the second element. An ordered pair with the first element a and the second element b is often written as (a, b).

For example, the Cartesian product of 2-element sets {1,2} and {3,4) is a 4- element set {(1,3), (1,4), (2,3), (2,4)}.

9.3.2 Set union:

If A and B are two sets, then the set having all the elements of A and B is called the union of A and B. The union of A and B, is written as AU B. According to the formal definition:

x is an element of A∪B if and only if:

x ЄA or x∈B.

For example, the union of the sets {1, 2, 3} and {2, 3, 4} is {1, 2, 3, 4}.

Figure below hows the union of sets A and B.

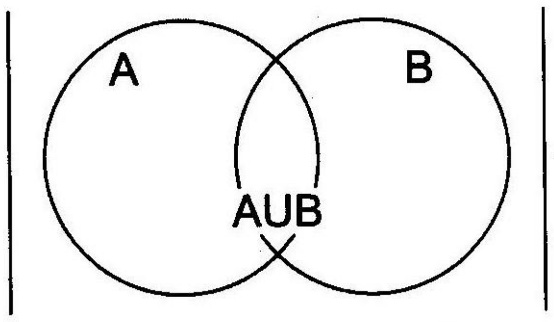


Figure: The union sets of A and B

Intersection:

The intersect operator allows you to create a new relation from the two relations that contains all the rows, which are common to both. For example, consider two relations A and B. The relation A consists of two rows-a and b and the relation B consists of four rows-a, b, c and d. Therefore, the new relation formed by using the intersect operator will consists of two rows-a and b.

Difference or complement

The two kinds of complements are:

• The Relative complement

• The Absolute complement

The Relative Complement: The relative complement of X in Y is the set of elements which are in Y but not in X. This is written as Y - X (also Y\X).

Y-X = {x € Y l x £ X}.

For example:

{1,2,3} - {2,3,4}= {1} and

{2,3,4} - {1,2,3} = {4}

If X, Y, and Z are the sets, then the following expressions hold:

Z-(X∩Y)=(Z-X) U(Z — Y)

Z-(Y-X) = (X NZ) U(Z — Y)

Z-(X∪Y)=(Z-X) п (Z-Y)

(Y-X) UZ = (YUZ) − (X- Z)

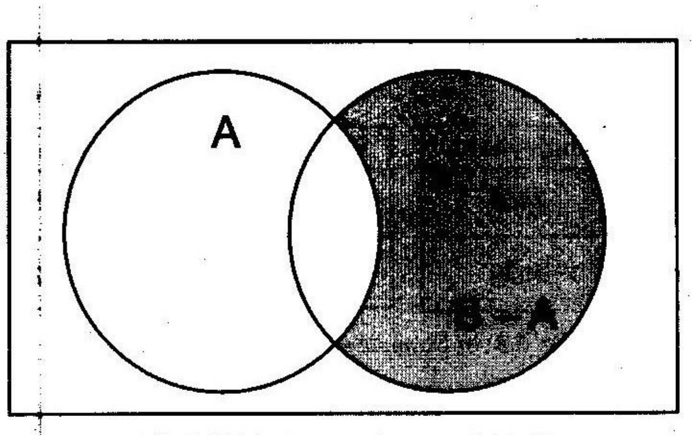
(Y-X) NZ = (YNZ) - X= Yn (z-X)

X-X= Ø

X-Ø=X

Ø -X= Ø

Where Ø is an empty set, with no elements in the set. Figure below shows the relative complement of Sets A and B.



The Absolute Complement: The relative complement of A in the universal set U is called the absolute complement (or simply complement) of A and is denoted by A^C (or A'), that is:

A^c=U-A

The Universal set U is a set that is so large that it can include all the sets that one may suppose to use.

For example, you can take the set of natural numbers as the universal set, and then the complement of the set of all odd numbers will be the set of all even numbers. Figure below shows the absolute complement of Set A.

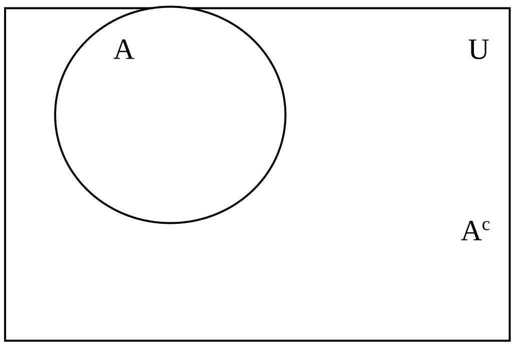


Figure The Absolute Complement or Complement of Set A

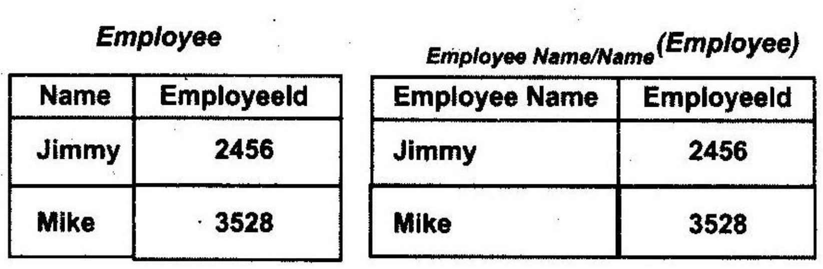
9.4 RENAME:

A rename is a unary operation written as $\rho_{a/b}$ (R), where:

a and b are the attribute names

R is a relation

The result is identical to R except that the 'b' field in all tuples is renamed to 'a' field. For example, consider the Employee relation and its renamed version in the Figure below



Employee

Name	EmployeeId
Jimmy	2456
Mike	3528

$_{Employee\ Name/Name}$(Employee)

Employee Name	EmployeeId
Jimmy	2456
Mike	3528

Figure: Relational Data Model and Relational Languages

9.4.1 Completeness of Relational Algebra:

The set of relation algebric operation {σ, л, ×, -} is a complete set of relational algebra, from which any of the other relational operations can be expressed as a combination of operations. For example, you can express INTERSECTION operation using UNION and DIFFERENCE. The following equation is used to express INTERSECTION operation using UNION and DIFFERENCE:

$R \cap S = (R U S) - ((R-S) U (S - R))$

9.4.2 The Aggregate Functions:

The aggregate functions are used to calculate the collection of values in the database. The aggregate functions are applied on the multiple values in the database and produce a single value as the output.

The aggregate functions include the following functions:

- ❖ MAXIMUM: It is used to retrieve the maximum value among the multiple values in a database.
- ❖ MINIMUM: It is used to retrieve the minimum value among the multiple values in a database.
- ❖ SUM: It returns the sum of all the values in a database.
- ❖ AVERAGE: It returns the average of the total values in a database.
- ❖ COUNT: It returns the count of all the numbers in a database.

9.5 SUMMARY:

In this you have learnt about the relational data model which represents the database as a collection of relations. A relation can be seen as a table of values. The six basic operations in relational algebra-selection, projection, Cartesian product, union, complement and rename have also been explained in this unit. Some additional operators are also included in relational algebra such as join, outer join and division.

And also studied about SQL. It is a query language that is used for retrieving and manipulating data stored in a database and also states the various SQL objects. SQL statements are used to create and access a database using the SQL server enterprise manager. For creating a data model for a complex database, you can use the EER model which includes additional concepts such as subclasses, superclasses and inheritance of semantic data modelling. In addition, EER model also provides the concepts of specialization and generalization that helps to reduce data redundancy.

9.6 KEY WORDS:

- ❖ **Relation:** It is a two-dimensional table, which is used to represent the data in the form of rows and column.
- ❖ **Domain:** It is a set of atomic values.
- ❖ **Attribute:** It is a column header in a relation that represents the attributes of an entity.
- ❖ **Relational schema:** It is the description of the database specified during the designing.
- ❖ **Constraints:** They are defined as a condition, which needs to be satisfied while storing data in a database.
- ❖ **Entity integrity:** It is a mechanism that allows uniquely identified rows in a table.
- ❖ **Relational algebra:** It is a set of relations, and these relations are closed under certain operators of the relational algebra.
- ❖ **Stored procedure:** It is a group of SQL statements that is used to perform a particular task.
- ❖ **Table:** It is a two-dimensional arrangement of data consisting of rows and columns.
- ❖ **Tuple:** In RDBMS, a row is termed as tuple that gives complete information of an entity.
- ❖ **View:** It is a logical table that consists data collated from the database after you executes a query to retrieve data from the database. Alias: It is symbolic or secondary name for a collection of data in a database.

9.7 SELF- ASSESSMENT QUESTIONS:

1. Explain a domain?
2. Define a tuple?
3. Define a relation?
4. Relational Data Model and Relational Languages?
5. What do you mean by relative complement? Explain with an example.
6. Explain the selection and projection operations in relational algebra?
7. Discuss the various operations in relational algebra with examples. Create a student database table using the SELECT statement to retrieve data from the table?
8. Give the steps to map an ER diagram into tables. Illustrate with the help of an example?

9.8 FURTHER READINGS:

1. Date, C. J., An Introduction To Database Systems. 7th Edition. Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LESSON – 10

DATA MANIPULATION LANGUAGE

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ The Data Manipulation Language (DML) in Database Management Systems (DBMS) are focused on the manipulation, retrieval, and modification of data stored within the database

STRUCTURE OF THE LESSON:

10.1 INTRODUCTION:

In the last unit, you learnt about the basics of DBMS. In this unit, you will learn about the relational data model. Relational data model makes use of the set theory and is based on the concept of the mathematical relation, which contains several data elements. The Relational algebra, which is a set of operations, can be used for manipulating the relations and specifying the queries. Functional dependencies are the constraints on the set of relations in a database.

In addition, you will also study about Structured Query Language (SQL) in this unit. It is a standard interactive and programming language used for accessing, querying, managing and updating data in Relational Database Management Systems (RDBM). By using

SQL, you can also create databases and various database objects such as tables, columns and views. This unit will explain how to map an Entity Relationship (ER) diagram or model into related tables. The ER model has been further enhanced as an Enhanced Entity Relational (EER) model, which is useful for representing any complex database.

10.2 MODIFYING DATA:

SQL provides statements for modifying the data in a database table. You can use these statements to insert, update and delete information from the database table. To modify data you can perform the following functions:

- ❖ Inserting data Updating data
- ❖ Deleting data
- ❖ Changing column name
- ❖ Eliminating duplicate information

10.3 INSERTING DATA:

You can insert data in the database table by specifying the table and column in which you want to insert the data. The general syntax to insert data is:

INSERT INTO table name (column names) VALUES (data values) You can insert data by using the following statements:

A. The INSERT statement
B. The SELECT statement

10.3.1 The INSERT statement:

The INSERT statement allows you to insert records to an existing table in the database. For example, you need to add details of a new customer to the cust details database table. To add a new customer in the cust details database table, you need to execute the following command:

```
INSERT INTO cust_details

values ('ken', 'burton', 012, 1000)
```

The above SQL statement adds the details of the new customer to the cust_details table. The cust_details database table has cust_firstname, cust_lastname, cust_id and cust_amount columns and the values for all these columns are specified in the SQL statement.

10.3.2 The SELECT statement:

The SELECT statement can provide the data to be inserted into the table, using INSERT INTO statement. The SELECT statement and INSERT INTO statement allow you to copy the particular data from one table to the other table. Consider the following example to insert records in the table:

```
INSERT INTO EmpTable

SELECT EmpName, EmpID, Salary

FROM Employees
```

The above SQL statement inserts the records such as EmpName and Salary in EmpTable from the Employees table.

10.3.3 Updating Data:

You need to update the existing data present in the database table. You can use the UPDATE statement to update the data that allows you to set which field is to be updated and the data to updated in that field.

10.3.4 The UPDATE statement:

The UPDATE statement allows you to modify the information, which is, contained within a table. Consider the following UPDATE statement to update the data:

```
UPDATE table name
SET column name = value
Where condition
```

For example, a company maintains the cust_details table to store the details of its regular customers. The company decides to give a 3% discount every year to all the regular customers. Therefore, you need to apply the changes to the cust_amount column of the cust_details table. The following SQL command allows you to apply the changes to the cust_details table:

```
UPDATE cust details

SET cust amount = cust amount* 0.97
```

Consider that you want to make the changes to the details of the specific customer in the cust_details table. The following command allows you to add an amount of $2000 to the cust_amount column of a customer with id 012.

```
UPDATE cust_details
SET cust amount = cust amount + 2000
WHERE cust id = 012
```

10.3.5 Deleting Data:

You can use the DELETE statement to delete data from the database table. You need to specify the table from which you want to delete data

10.3.6 The DELETE statement:

The DELETE command allows you to remove a record from the table. For example, if you need to delete the records of a customer with cust_id 011, you can execute the following command:

```
DELETE FROM cust details
WHERE cust id = 011
```

Changing Column Name : You can rename or change the database column name using the RENAME keyword in the ALTER statement.

10.4 THE ALTER STATEMENT:

To rename a column in a database table you use the ALTER statement and to specify the new name for the column you use the RENAME keyword with the ALTER statement. Consider the following syntax to rename a column in a database table:

```
ALTER TABLE Table name RENAME COLUMN Column name TO
New column name
```

The SQL statement renames a specified column with a new column name in the specified database table. Consider the following syntax to rename Address database column to Contact in the Mytable database table:

```
ALTER TABLE Mytable RENAME COLUMN Address TO Contact
```

The ALTER statement changes the name of Address column to Contact column in the database table Mytable.

Eliminating the Duplicate Information:

To eliminate the duplicate information or to retrieve unique data records from the database table, you use DISTINCT keyword with the SELECT statement. The following syntax determines the general SELECT statement with a DISTINCT keyword to retrieve only the unique records from the database table:

```
SELECT
DISTINCT Column_name1, Column_name2, FROM Table name
Column_namen
```

Consider the following example where you need to retrieve the unique data records from the Customers table.

```
SELECT
            DISTINCT City

            FROM Customers;
```

The SQL statement returns the unique records of the City column from the Customers database table, even if a particular city name occurs multiple times in the Customers table.

10.5 TABLE MODIFICATION:

Table: The table is a to store the data

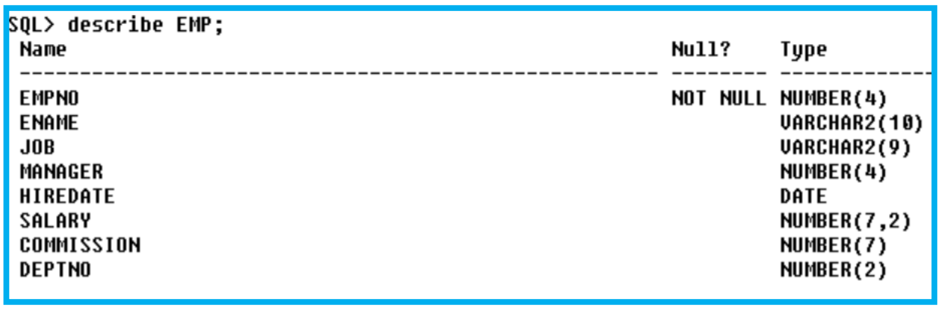
Table Modification Commands

To modify the structure of a table use, ALTER TABLE command. It is given below:

- ALTER: This command is used to modify the structure of the table. Using this command, we can perform four different operations. They are:
  - ALTER – Modify
  - ALTER – ADD
  - ALTER – RENAME
  - ALTER – DROP

1. ALTER – Modify: This command allows you to change the data type of a column from an existing data type to a new data type and also increase or decrease the size of the Syntax: ALTER TABLE <Table name> MODIFY <Column name> data type(size);

   E.g., Structure of EMP table



```
SQL> describe EMP;
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ------------
 EMPNO                                                 NOT NULL NUMBER(4)
 ENAME                                                          VARCHAR2(10)
 JOB                                                            VARCHAR2(9)
 MANAGER                                                        NUMBER(4)
 HIREDATE                                                       DATE
 SALARY                                                         NUMBER(7,2)
 COMMISSION                                                     NUMBER(7)
 DEPTNO                                                         NUMBER(2)
```

   Now to change the column size of ename from 10 to 20, we use Alter table command as follows:

   ALTER TABLE EMP MODIFY ENAME VARCHAR2(20);

Syntax for modify more than one column

```
ALTER TABLE <Table name> MODIFY
(
   Column-1 data type(size),
   Column-2 data type(size),
   ----,
   Column-n data type(size)
);
```

E.G., ALTER TABLE emp MODIFY (
 empno number (5), ename char (10)
);

Restrictions on ALTER MODIFY:

You can reduce the size of a column. Column only if the existing column values fit in the new dimension.

To change the data type of a column, the column must be empty.

1. ALTER ADD: This command is used to add a new column to the existing table.
2. Syntax: ALTER TABLE<Table name> ADD column name datatype(size);

E.g., ALTER TABLE EMP ADD phone_no NUMBER (10);

Syntax to Add more than one column:

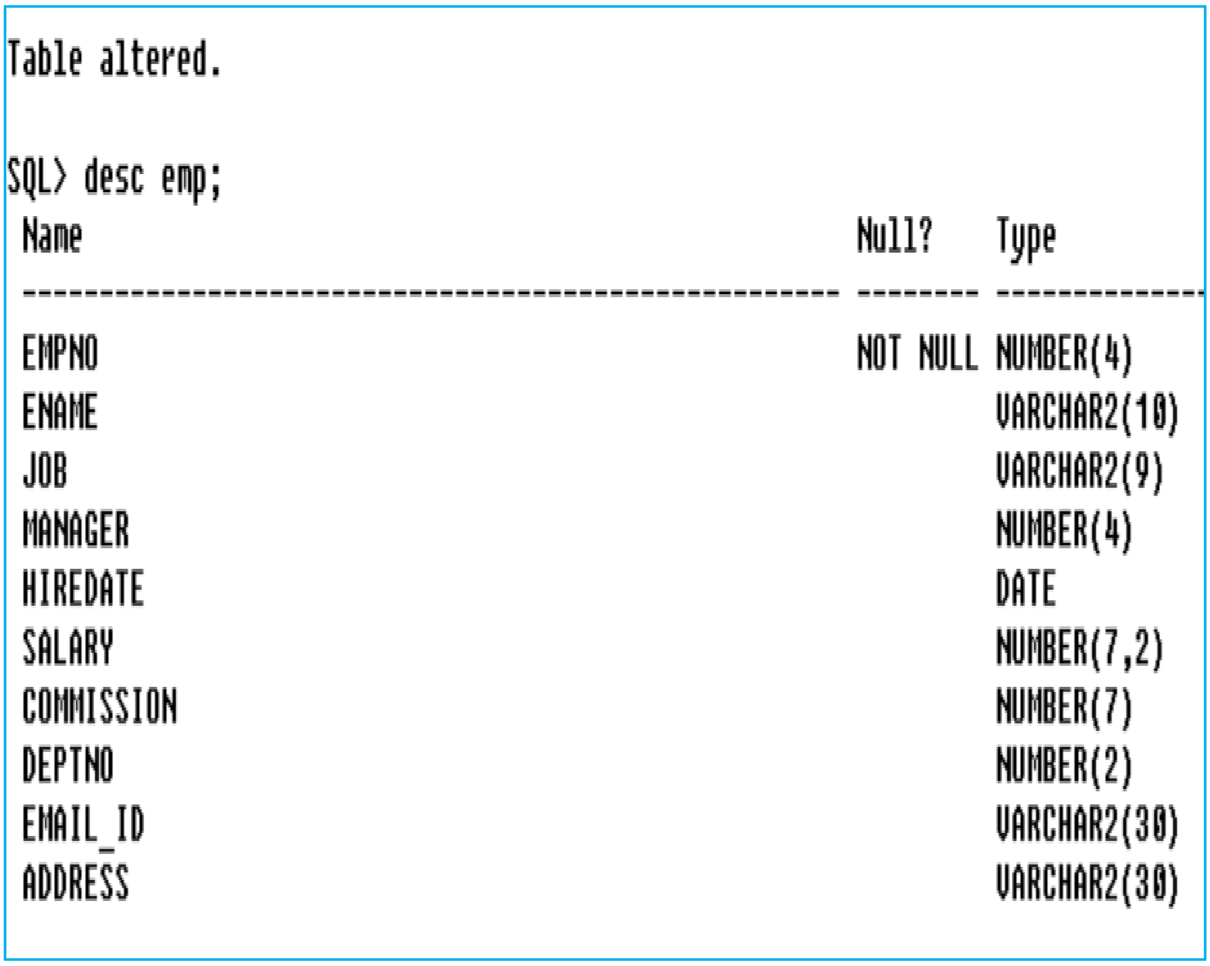
```
ALTER TABLE<table name>
ADD (column-1 datatype(size),
      column-2 datatype(size), … ,
      column-n datatype(size)
    );
```

```
ALTER TABLE EMP ADD
(
mail_id varchar2(30),
address varchar2(30)
);
```

Note: Whenever we add a new column to a table the new column is always added to the end of the table only.

3. ALTER RENAME: This command is used to change the column name from old column name to new column name and also to change the table name from old to new name.

Syntax:



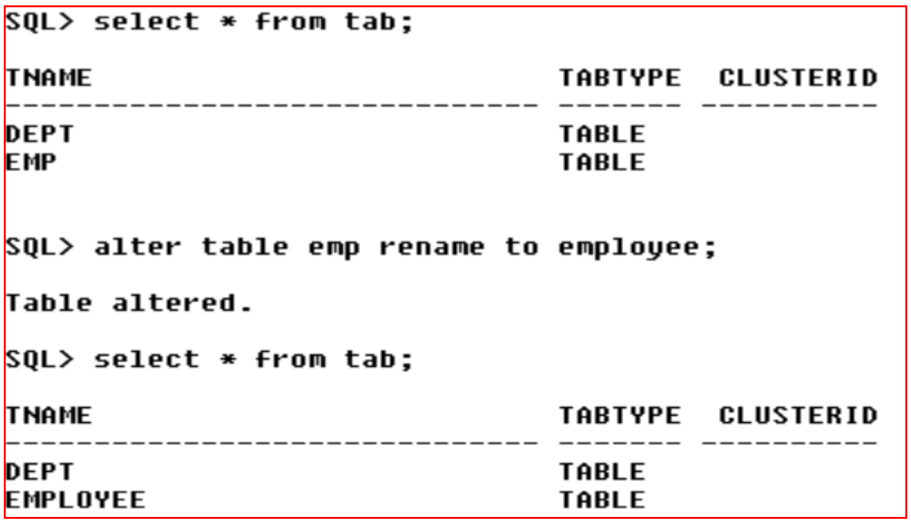
```
Table altered.

SQL> desc emp;
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- -------------
 EMPNO                                                 NOT NULL NUMBER(4)
 ENAME                                                          VARCHAR2(10)
 JOB                                                            VARCHAR2(9)
 MANAGER                                                        NUMBER(4)
 HIREDATE                                                       DATE
 SALARY                                                         NUMBER(7,2)
 COMMISSION                                                     NUMBER(7)
 DEPTNO                                                         NUMBER(2)
 EMAIL_ID                                                       VARCHAR2(30)
 ADDRESS                                                        VARCHAR2(30)
```

ALTER TABLE<table-name> RENAME COLUMN <old column name> to <new column name>;

E.g., ALTER TABLE EMP RENAME COLUMN SALARY TO SAL;

Syntax change table name:

ALTER TABLE<old table name> RENAME TO <New table name>; E.g., ALTER TABLE EMP RENAME TO EMPLOYEE;



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE


SQL> alter table emp rename to employee;

Table altered.

SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMPLOYEE                       TABLE
```

4. ALTER – DROP: This command is used to remove the column from the existing table. Syntax: ALTER TABLE< table name>DROP COLUMN column-name; E.g., ALTER TABLE EMP DROP COLUMN email_id;

   Syntax to drop more than one column:

   ALTER TABLE EMP DROP (email_id, address); /*Here we can't drop all columns. */

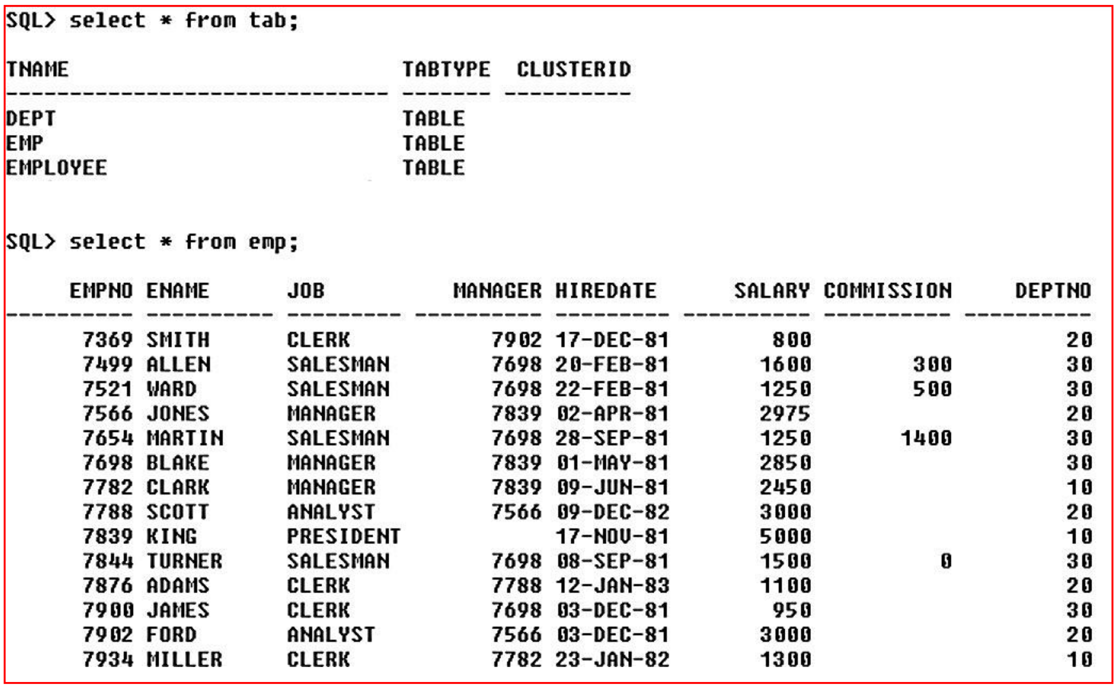
10.6 TABLE TRUNCATION:

TRUNCATE:

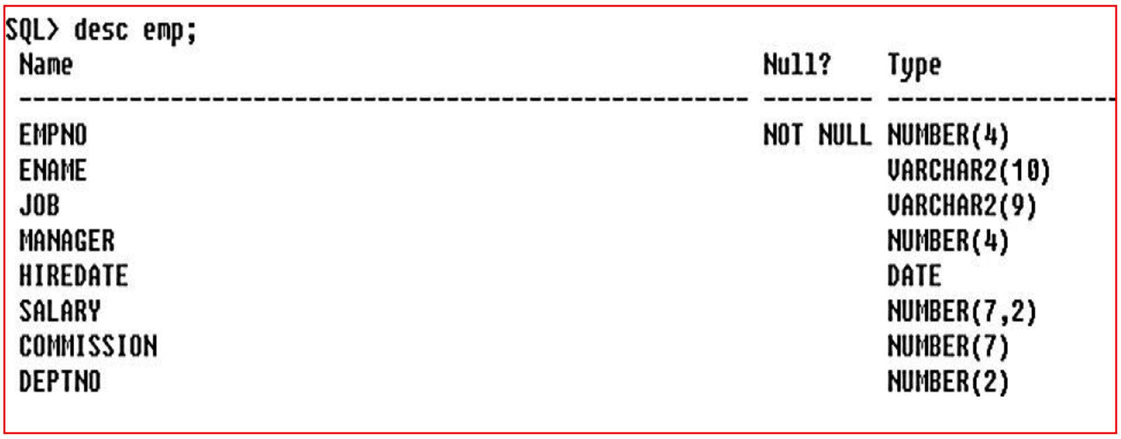
This command is used delete all records permanently from the existing table but the table structure will remain as it is.
 It is used to remove all the rows from the table and to release the storage space used by the table.
It will not facilitate for rollback
Only owner of the table is allowed to apply the truncate command on the table.

<u>Table Truncation Command</u>

Syntax: TRUNCATE TABLE <table name>;

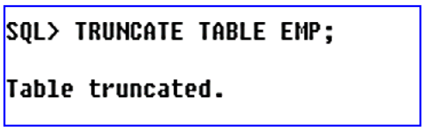


```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
EMPLOYEE                       TABLE


SQL> select * from emp;

     EMPNO ENAME      JOB          MANAGER HIREDATE      SALARY COMMISSION     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-81        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 09-DEC-82       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 12-JAN-83       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
```



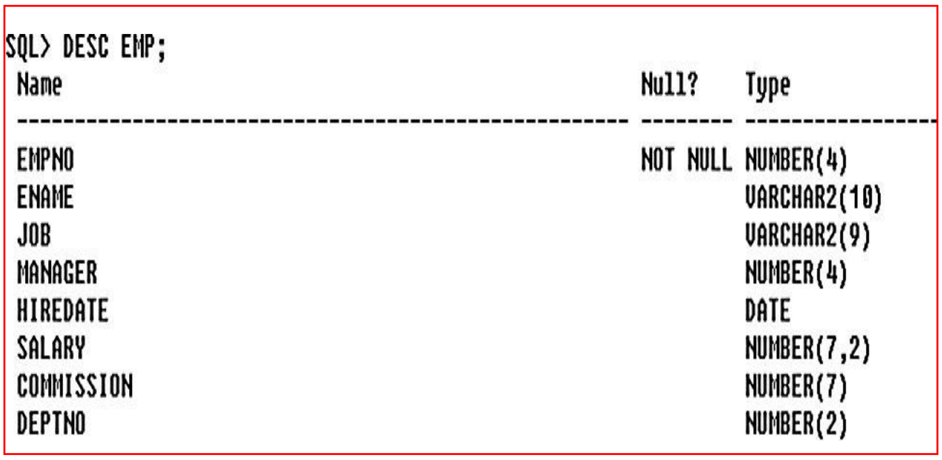
```
SQL> desc emp;
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ----------------
 EMPNO                                                 NOT NULL NUMBER(4)
 ENAME                                                          VARCHAR2(10)
 JOB                                                            VARCHAR2(9)
 MANAGER                                                        NUMBER(4)
 HIREDATE                                                       DATE
 SALARY                                                         NUMBER(7,2)
 COMMISSION                                                     NUMBER(7)
 DEPTNO                                                         NUMBER(2)
```

E.g., TRUNCATE TABLE EMP;



```
SQL> TRUNCATE TABLE EMP;

Table truncated.
```

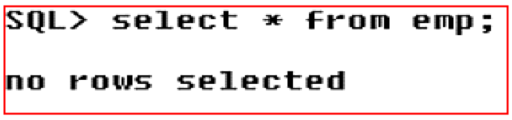
Now, let us verify whether ‘emp’ table is existing in the database by executing ‘desc’ command. As it will also show the structure of the table, if it is existing.



```
SQL> DESC EMP;
 Name                                                  Null?    Type
 ----------------------------------------------------- -------- ----------------
 EMPNO                                                 NOT NULL NUMBER(4)
 ENAME                                                          VARCHAR2(10)
 JOB                                                            VARCHAR2(9)
 MANAGER                                                        NUMBER(4)
 HIREDATE                                                       DATE
 SALARY                                                         NUMBER(7,2)
 COMMISSION                                                     NUMBER(7)
 DEPTNO                                                         NUMBER(2)
```

From the above output, we understood that the 'emp' table is still existing in the database even after applying truncate table command upon 'emp'.

Let us display the data present in the 'emp' table by executing 'select' command.



```
SQL> select * from emp;

no rows selected
```

From the output of 'select' command, we can conclude that truncate table command only removes all the records present in the given table but the table structure remains in the database. It is generally used to free the storage space occupied by the database tables after taking the backup.What is the difference between Drop Table Command Truncate Table Command?

10.7 TRUNCATE TABLE COMMAND:

- Removes all rows from a table without logging the individual row deletions.
- TRUNCATE TABLE is similar to the DELETE statement with no WHERE clause.
- TRUNCATE TABLE is faster than DELETE command and uses fewer system and transaction log resources.

❖ **Imposition of Constraint:** Imposing constraints in a database management system "DBMS" is an important aspect of maintaining data integrity, consistency and accuracy. Constraints are rules or conditions imposed on the data in a database to maintain data quality and reliability. The following are some common types of constraints in a DBMS:

❖ **Primary Key Constraint:** A primary key is a unique identifier for a record in a table. It ensures that each record can be uniquely identified.

❖ It enforces the uniqueness of a column or combination of columns and ensures that they cannot contain null values.

❖ **Unique Constraint:** A unique constraint ensures that values in a column (or combination of columns) are unique across a table.

❖ Unlike a primary key, a unique constraint can allow null values.

- **Foreign Key Constraint:** A foreign key establishes a relationship between two tables. It ensures that values in a column (or combination of columns) in one table match values in another table.
- This helps maintain referential integrity between related tables.
- **Check Constraint:** A check constraint specifies a condition that must be true for a record to be inserted or updated in a table.
- It can be used to enforce business rules or other conditions on the data.
- **Default Constraint:** A default constraint specifies a default value for a column. If no value is provided during an insert operation, the default value will be used.
- **Not Null Constraint:** This constraint ensures that a column cannot contain null values. It enforces that every record must have a value for that column.
- **Entity Integrity Constraint:** This is a high-level constraint that states that a primary key cannot have a null value.
- **Referential Integrity Constraint:** This is a high-level constraint that ensures that foreign keys refer to existing primary keys in another table.
- **Domain Constraint:** A domain constraint defines the allowable values for a column. It can be used to restrict the range of valid data.
- **Multi-Attribute Constraints:** These constraints involve conditions that span multiple attributes or columns.
- **User-Defined Constraints:** Some DBMSs allow you to define custom constraints using a programming language like PL/pgSQL or PL/SQL.
- **Triggers:** Although not technically a constraint, triggers are often used to apply complex business rules or conditions that cannot be easily handled with simple constraints.

By using these restrictions, the DBMS ensures that the data is consistent, accurate, and reliable. Helps prevent errors, anomalies and inconsistencies in the database. Additionally, it provides a layer of security by preventing unauthorized access or modification of sensitive data.

10.8 SET UNION:

If A and B are two sets, then the set having all the elements of A and B is called the union of A and B. The union of A and B, is written as AU B. According to the formal definition:

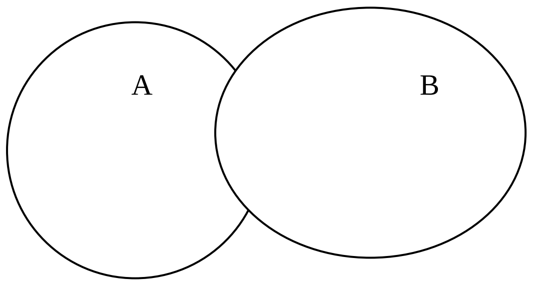
x is an element of AB if and only if:

x ЄA or

xЄB.

For example, the union of the sets {1, 2, 3} and {2, 3, 4} is {1, 2, 3, 4}.

Figure below shows the union of sets A and B.



10.9 SUMMARY:

The Data Manipulation Language is instrumental in the practical utilization of a database. It provides the tools necessary for users to interact with and manipulate data, facilitating the dynamic and responsive nature of database-driven applications. DML is an essential aspect of effective data management, enabling users to perform operations that keep the database aligned with the evolving needs of the organization.

10.10 KEY WORDS:

- ❖ **Relation:** It is a two-dimensional table, which is used to represent the data in the form of rows and column.
- ❖ **Domain:** It is a set of atomic values.
- ❖ **Attribute:** It is a column header in a relation that represents the attributes of an entity.
- ❖ **Relational schema:** It is the description of the database specified during the designing.
- ❖ **Constraints:** They are defined as a condition, which needs to be satisfied while storing data in a database.
- ❖ **Entity integrity:** It is a mechanism that allows uniquely identified rows in a table.
- ❖ **Relational algebra:** It is a set of relations, and these relations are closed under certain operators of the relational algebra.
- ❖ **Stored procedure:** It is a group of SQL statements that is used to perform a particular task.
- ❖ **Table:** It is a two-dimensional arrangement of data consisting of rows and columns.
- ❖ **Tuple:** In RDBMS, a row is termed as tuple that gives complete information of an entity.
- ❖ **View:** It is a logical table that consists data collated from the database after you executes a query to retrieve data from the database. Alias: It is symbolic or secondary name for a collection of data in a database.

10.11 SELF- ASSESSMENT QUESTIONS:

1. Explain modifying data?
2. Write the inserting data?
3. Differences between update, delete, alter and modification?
4. Discuss the truncate table command?

10.12 FURTHER READINGS:

1. Date, C. J., An Introduction To Database Systems. 7th Edition. Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LESSON - 11

INTRODUCTION, STRUCTURE OF PL/SQL

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ PL/SQL in Database Management Systems (DBMS) is to provide a powerful and efficient procedural programming language that seamlessly integrates with SQL.
- ❖ PL/SQL, or Procedural Language/Structured Query Language, serves as an extension to standard SQL by incorporating procedural constructs such as loops, conditional statements, and exception handling.

STRUCTURE OF THE LESSON:

11.1 INTRODUCTION:

PL/SQL stands for Procedural Language/Structured Query Language and is a powerful extension of SQL (Structured Query Language) used for writing procedural code in Oracle databases. It combines the query and data manipulation capabilities of SQL with the flexibility and control structures of a programming language.

Here are some key concepts you should understand about PL/SQL:

Blocks: PL/SQL code is organized in blocks. A block can contain one or several hundred lines. It consists of three sections: declaration, executable and exception handling.

Declarations section: Here you declare the variables, constants and cursors used in the block. Variables can store values, while constants store values that do not change.

```
DECLARE

   my_variable NUMBER := 10;

   my_constant CONSTANT NUMBER := 100;
```

Executable Section: This is where you write the main logic of your code. It can **contain** SQL statements for querying and manipulating data, as well as control structures **such as** loops and conditional statements.

```
BEGIN

   IF my_variable > my_constant THEN

      DBMS_OUTPUT.PUT_LINE('Variable is greater than constant');

   ELSE

      DBMS_OUTPUT.PUT_LINE('Variable is less than or equal to constant');

   END IF;

END;
```

Exception Handling Section : This is where you handle exceptions or errors that may occur during code execution. You can define custom exceptions and specify how they should be handled.

```
EXCEPTION

   WHEN OTHERS THEN

      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
```

Variables and Data Types : PL/SQL supports a variety of data types, including numeric, character, date, and boolean types. Variables are used to store and manipulate data in a block.

```
my_variable NUMBER := 10;

my_string VARCHAR2(50) := 'Hello, World!';

my_date DATE := SYSDATE;
```

Control Structures: PL/SQL contains standard programming constructs such as loops and conditional statements.

```
FOR i IN 1..5 LOOP

   DBMS_OUTPUT.PUT_LINE('Iteration ' || i);

END LOOP;
```

Procedures and Functions: PL/SQL allows you to define reusable code blocks such as procedures and functions. Unlike functions, procedures do not return values.

```
CREATE OR REPLACE PROCEDURE my_procedure AS

BEGIN

   -- code goes here

END;
```

Cursors: Cursors are used to process individual rows returned by a SQL query. They allow you to browse different results.

Packages: Packages allow you to group related procedures, functions, variables, and cursors. They help you organize and manage your code.

```
CREATE OR REPLACE PACKAGE my_package AS

   PROCEDURE procedure_1;

   PROCEDURE procedure_2;

END my_package;
```

Triggers: Triggers are special blocks of code that are executed in response to specific events, such as: B. inserting, updating or deleting a table.

```
CREATE OR REPLACE TRIGGER my_trigger

BEFORE INSERT ON my_table

FOR EACH ROW

BEGIN

   -- code goes here

END;
```

These are just some of the fundamental concepts of PL/SQL. It's a versatile language that allows for complex data manipulation and control flow, making it a valuable tool for working with Oracle databases. Remember to practice and experiment with PL/SQL to gain proficiency in using it effectively.

11.2 STRUCTURE OF PL/SQL:

PL/SQL (Procedural/Structured Query Language) is a procedural extension of SQL from Oracle Corporation. It allows you to combine SQL queries with procedural constructs such as loops, conditional statements and exception handling. Here is an overview of the basic PL/SQL structure.

Declaration section: This section is optional. It is used to declare variables, constants, cursors and other program structures. The declarations are placed at the beginning of the PL/SQL block.

Example:

```
DECLARE

  -- Variable declarations

  num1 NUMBER;

  name VARCHAR2(50);

  -- Constant declaration

  PI CONSTANT NUMBER := 3.14;
```

Execution Section: This is the main part of the PL/SQL block where you write the logic. It contains SQL statements along with procedural constructs like loops, conditional statements, etc.

Example:

```
BEGIN

  -- SQL Statements and Procedural Logic

  num1 := 10;

  name := 'John Doe';

  IF num1 > 5 THEN

    DBMS_OUTPUT.PUT_LINE('Number is greater than 5');

  ELSE

    DBMS_OUTPUT.PUT_LINE('Number is less than or equal to 5');

  END IF;

END;
```

Exception Handling Section (Optional) : This section allows you to handle exceptions that might occur during the execution of the block. It helps in gracefully handling errors.

Example:

```
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No data found');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
```

SQL Statements: Inside the execution section, you can use standard SQL statements like SELECT, INSERT, UPDATE, DELETE, etc.

Example:

```
INSERT INTO employees (emp_id, emp_name) VALUES (1, 'John Doe');
```

Procedural Constructs: PL/SQL supports various procedural constructs like loops (FOR, WHILE), conditional statements (IF-THEN-ELSE, CASE), and more.

Example (FOR Loop):

```
FOR i IN 1..10 LOOP
  DBMS_OUTPUT.PUT_LINE('Iteration ' || i);
END LOOP;
```

Subprograms (Optional):

You can define procedures and functions within a PL/SQL block. These are named blocks of code that can be reused.

Example (Procedure): CREATE OR REPLACE PROCEDURE my_procedure IS

```
BEGIN
  -- Procedure logic here
END my_procedure;
```

Triggers (Optional):

PL/SQL can be used to define triggers, which are special stored procedures that are executed in response to certain events on a table.

Example:

```
CREATE OR REPLACE TRIGGER my_trigger
```

```
BEFORE INSERT ON my_table

FOR EACH ROW

BEGIN

  -- Trigger logic here

END;
```

Anonymous Blocks: These are PL/SQL blocks that are not stored as database objects. They are executed once and discarded.

Example:

```
BEGIN

  -- Anonymous block logic here

END;
```

Remember that this is a basic overview of the PL/SQL structure. In practice, PL/SQL can be quite complex, especially in larger applications where there may be many complex procedures, functions, packages and logic.

11.3 PL/SQL LANGUAGE ELEMENTS:

PL/SQL (Procedural Language/Structured Query Language) is a programming language extension for Oracle databases. Combine SQL with procedural programming language constructs. Here are some of the key elements of the PL/SQL language:

- ❖ **Blocks:** PL/SQL programs are organized in blocks, which can be anonymous or named. A block consists of three parts: declaration, execution and exception handling.
- ❖ **Variables:** Variables are used to store data that can be manipulated within a block or procedure. They must be declared before use.
- ❖ **Data Types:** PL/SQL supports a variety of data types including numeric, character, date, logical and more.
- ❖ **Constants:** Constants are variables whose values cannot be changed. They are declared with the CONSTANT keyword.
- ❖ **Operators:** PL/SQL supports a wide range of operators, including arithmetic, comparison, logical, and string concatenation operators.
- ❖ **Control Structures:** IF-THEN-ELSE: Conditional statement to execute other code based on a condition.
- ❖ **CASE:** A switch type statement to select one of several alternatives based on a value.
- ❖ **LOOP:** Construct for repeatedly executing a block of code.
- ❖ **WHILE-LOOP:** A loop that executes a block of code while a condition is true.
- ❖ **FOR-LOOP:** A loop that repeats a specified number of times.
- ❖ **Cursors:** Cursors are used to process multiple rows returned from a SQL query. They allow you to browse different results.

- **Exception Handling:** PL/SQL provides error and exception handling mechanisms. This includes an EXCEPTION block where you can specify what to do if an error occurs.
- **Procedures:** Procedures are called blocks of code that accept parameters and can be called from other parts of the program.
- **Functions:** Functions are similar to procedures but return a value.
- **Packages:** A package is a set of related procedures, functions, variables, and other elements. It helps you organize and manage your code.
- **Trigger:** A trigger is a special type of stored procedure that is executed automatically in response to specific events in a specific table or view.
- **PL/SQL Records and Collections:** Records are used to store related data items, and collections allow you to work with records.
- Dynamic SQL: Allows dynamic creation and execution of SQL statements at runtime.
- **Exception Handling:** PL/SQL provides a robust error and exception handling mechanism using constructs such as EXCEPTION, RAISE and EXCEPTION_INIT.
- **Packages and Types:** PL/SQL allows you to create packages to encapsulate related procedures, functions, variables, and other elements.
- **SQL Statements:** PL/SQL contains SQL statements for database operations like SELECT, INSERT, UPDATE, DELETE etc.
- **Packages and Types:** PL/SQL allows you to create packages that contain procedures, functions, variables, and other related elements.
- **Object Types:** PL/SQL supports the creation of custom object types, allowing modeling of real-world objects.
- **Transactions:** PL/SQL supports managing transactions using commands such as COMMIT and ROLLBACK.

Remember that PL/SQL is specific to Oracle databases and some of its features and syntax may differ from those of other database systems. If you are working with a different database, the procedural language is similar but the details are different.

11.4 DATA TYPES:

Data Types in dbms

In a Database Management System (DBMS), data types define the type of data that a particular column or variable can hold. Different DBMSs may have their own set of data types, but I`ll provide a general overview of common data types used in many relational database systems:

1. **Numeric Types:** INTEGER or INT: Used for whole numbers without decimal points.

SMALLINT: Similar to INTEGER, but typically with a smaller range of values.

BIGINT: Used for very large integers.

2. Decimal or Numeric Types: DECIMAL(p, s) or NUMERIC(p, s): Used for numbers with decimal points. 'p' represents the total number of digits, and 's' represents the number of digits after the decimal point.

3. Floating-Point Types:

FLOAT(p): A floating-point number with a specified precision.

REAL: A single-precision floating-point number.

DOUBLE PRECISION: A double-precision floating-point number.

4. Character String Types:

CHAR(n): Fixed-length character string with a specified maximum length of 'n'.

VARCHAR(n) or VARCHAR2(n): Variable-length character string with a maximum length of 'n'.

TEXT: A type for storing large amounts of text.

5. Date and Time Types:

DATE: Stores date only (year, month, day).

TIME or TIMESTAMP: Stores time only or both date and time.

6. Boolean Type:

BOOLEAN or BOOL: Stores true or false values.

7. Binary Large Object Types:

BLOB: Used to store binary data like images, audio, or video files.

CLOB: Used to store large amounts of character data.

8. Row Identifier Types:

SERIAL or AUTO_INCREMENT: Automatically generates a unique value for each new row.

9. Array or Collection Types (may not be supported in all DBMSs):

ARRAY: An ordered collection of elements of the same type.

MULTISET: A collection of elements that can be of different types.

10. User-Defined Types:

Some DBMSs allow users to define their own custom data types based on existing types.

11. XML Types:

For storing and manipulating XML data.

12. Geospatial Types (may not be supported in all DBMSs):

For handling geospatial data, like points, lines, and polygons.

13. JSON Types (may not be supported in all DBMSs):

For storing and querying JSON data.

Remember that the availability and specific syntax of data types may vary for different DBMSs (e.g., Oracle, MySQL, PostgreSQL, SQL Server). It's essential to refer to the documentation of the specific DBMS you are using for precise information about its supported data types and their usage.

11.5 CONTROL STRUCTURE:

PL/SQL control structures are used to control the flow of execution.

PL/SQL provides various types of statements that provide this procedural functionality. These commands are almost identical to commands in other languages.

Control statement flow can be divided into the following categories:

- Conditional Control
- Iterative Control
- Sequential Control

Conditional Control: In PL/SQL, you can use the IF statement to control the execution of a block of code. In PL/SQL, you can use the IF -THEN - ELSIF - ELSE - END IF syntax within a code block to specify specific conditions for executing a particular block of code.

Syntax:

```
IF < Condition > THEN

   < Action >

ELSIF <Condition> THEN

   < Action >

ELSE < Action >

END IF;
```

Example:

create file named "condi.sql"

```
DECLARE

     a Number := 30;      b Number;

BEGIN

     IF a > 40 THEN

       b := a - 40;

       DBMS_OUTPUT.PUT_LINE('b=' || b);

    elseif a = 30 then

       b := a + 40;

       DBMS_OUTPUT.PUT_LINE('b=' || b);
```

```
      ELSE
        b := 0;
        DBMS_OUTPUT.PUT_LINE('b=' || b);
      END IF;
END;
/
```

Output:

```
SQL>set  serveroutput on
SQL>start d://condi.sql
b=70
PL/SQL successfully completed.
```

Iterative Control: A Iterative control specifies the ability to repeat or skip sections of a code block. A loop marks a sequence of statements that must be repeated. The keyword loop must be placed before the first statement in a sequence of repeated statements. On the other hand, the keyword end loop is placed immediately after the last statement in the sequence. Once executed, a loop continues forever. Therefore, loops are always accompanied by conditional statements that control how often the loop executes.

11.5.1 PL/SQL supports the following constructs for iteration control:

Simple loops: In simple loops, the loop keyword must be placed before the first statement of the sequence, and the keyword end loop must be written at the end of the sequence. Exit the loop.

Syntax:

```
Loop
  < Sequence of statements >
 End loop;
```

Example:

create file named it.sql

```
DECLARE
   i number := 0;
BEGIN
    LOOP
   dbms_output.put_line ('i = '||i);
   i:=i+1;
            EXIT WHEN i>=11;
    END LOOP;
END;
/
```

Output:

```
SQL>set serveroutput on

SQL>start d://it.sql
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
PL/SQL successfully completed.
```

11.5.2 WHILE loop:

The while loop executes commands in its body as long as the condition remains true

Syntax :

```
WHILE < condition >
LOOP
      < Action >
END LOOP
```

Example :

find reverse of given number using while loop

```
DECLARE
   num Number(3) :=123;
   ans Number(3) :=0;
   i Number(3) :=0;
 BEGIN
   WHILE num != 0
    LOOP
        i:=mod(num,10);
        ans:=(ans * 10 ) + i;
        num:=floor(num/10);
    END LOOP;
   dbms_output.put_line('reverse of given number is: ' || ans);
END;
/
```

Output :

```
SQL>set serveroutput on
SQL>start d://rev.sql
reverse of given number is: 321
PL/SQL successfully completed.
```

11.5.4 The FOR Loop:

The FOR loop can be used when the number of iterations to be executed are known.

Syntax :

```
FOR variable IN [REVERSE] start..end
 LOOP
   < Action >
 END LOOP;
```

The variable in the For Loop need not be declared. Also the increment value cannot be specified. The For Loop variable is always incremented by 1.

Example :

```
DECLARE
     i number ;
 BEGIN
    FOR i IN 1 .. 10
     LOOP
         dbms_output.put_line ('i = '||i);
     END LOOP;
 END;
 /
```

Output :

```
SQL>set serveroutput on


SQL>start d://it.sql
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
PL/SQL successfully completed.
```

Sequential Control:

11.5.5 GOTO Statement:

The GOTO statement changes the flow of control within a PL/SQL block. This statement allows execution of sections of code that are not in the normal flow of control. Entry points to such code blocks are marked with a <<userdefined>> tag. A GOTO statement can use this custom name to jump to and execute this code block.

Syntax :

```
GOTO jump;
....
<<jump>>
```

Example :

```
DECLARE

 BEGIN
     dbms_output.put_line ('code starts');
     dbms_output.put_line ('before GOTO statement');
    GOTO down;
     dbms_output.put_line ('statement will not get executed..');
     <<down>>
     dbms_output.put_line ('flow of execution jumped here.');
 END;
 /
```

Output :

```
SQL>set serveroutput on

SQL>start d://a.sql
code starts
before gotostatements
flow of execution jumped here.
PL/SQL successfully completed.
```

11.6 SUMMARY :

The structure of PL/SQL is designed to provide a disciplined and efficient way to write procedural code within a database environment. The clear organization of declarations, executable code, and exception handling ensures readability, maintainability, and robust error management in database-centric applications.

11.7 KEY WORDS :

❖ **SELECT:** Retrieves data from one or more tables based on specified conditions, allowing users to query and retrieve specific information from the database.

- ❖ **INSERT:** It Adds new records (rows) to a table, specifying values for each column or providing data from a subquery.
- ❖ **UPDATE:** Modifies existing records in a table by changing the values of specific columns based on specified conditions.
- ❖ **TRUNCATE:** Removes all rows from a table, effectively deleting all data, but unlike DELETE, it does not generate individual row deletion logs and is usually faster.

11.8 SELF ASSESSMENT QUESTIONS:

1. Explain Structure Of Pl/SQL?
2. Describe the data types?
3. Write the Pl/SQL Supports The Following Constructs For Iteration Control?
4. Define GOTO Statement?

11.9 FURTHER READINGS :

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LESSON - 12

STEPS TO CREATE A PL/SQL PROGRAM

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ Creating a PL/SQL (Procedural Language/Structured Query Language) program in Database Management Systems (DBMS) are aimed at leveraging the power of a procedural language for efficient and effective database processing.

STRUCTURE OF THE LESSON:

12.1 INTRODUCTION:

Structured Query Language (SQL) is the standard language for interacting with relational database management systems (DBMS). It provides a powerful set of tools for managing and manipulating data. However, when it comes to developing more complex and robust database applications, it often becomes necessary to extend SQL's capabilities. This is where Procedural Language/Structured Query Language (PL/SQL) comes into play.

PL/SQL is a powerful and versatile programming language designed specifically for Oracle Database. It integrates seamlessly with SQL, allowing developers to combine the declarative power of SQL with the procedural constructs of a programming language. Creating a PL/SQL program involves a systematic approach to defining, organizing, and implementing the logic that will be executed on the database server.

The essential steps to create a PL/SQL program in a Database Management System (DBMS). Whether you are a seasoned developer or just starting with database programming, understanding these steps will empower you to harness the full potential of PL/SQL for developing efficient and scalable database applications.

12.2 STEPS TO CREATE A PL/SQL PROGRAM:

PL/SQL is a procedural language extension of SQL that is used to create programs that run on Oracle databases. Here are the steps to create a PL/SQL program:

1. **Open Oracle SQL Developer**: Oracle SQL Developer is an Integrated Development Environment (IDE) that can be used to create and manage PL/SQL programs.
2. **Create a connection to the database**: Before creating a PL/SQL program, you need to connect to the database where you want to create the program.
3. **Create a new PL/SQL file**: Once you have connected to the database, you can create a new PL/SQL file by right-clicking on the connection and selecting "New" > "PL/SQL File".
4. **Write your PL/SQL code**: In the new file, you can write your PL/SQL code. You can use SQL statements, control structures, and variables in your code.
5. **Save your PL/SQL file**: After writing your code, save the file with a .sql extension.
6. **Compile your PL/SQL program**: To compile your program, right-click on the file and select "Compile".
7. **Run your PL/SQL program**: Once your program has been compiled successfully, you can run it by right-clicking on the file and selecting "Run".

12.3 ITERATIVE CONTROL CURSORS:

In the field of database management systems, cursors serve as control structures that facilitate navigation between records in a database. This allows sequential processing of individual rows within a query result set. Cursors are used in various databases such as Oracle, SQL Server, and MySQL. These are especially useful when executing data manipulation language (DML) statements such as Update, Insert, and Delete.

- ❖ **There are two different types of cursors within PL/SQL:** implicit cursors and explicit cursors. Implicit cursors are automatically generated by Oracle when you execute a SQL statement. Explicit cursors, on the other hand, are defined by the programmer for more control over the context area. These must be specified in the declaration section of the PL/SQL block. Typically these are set for SELECT statements that produce multiple rows as output. This allows you to iterate over the dataset and perform any desired operations.
- ❖ **Steps to Create a Cursor:**A cursor is a temporary work area created in the system memory when DML statements like Insert, Update, and Delete queries are executed. A cursor allows you to process the result set one row at a time, instead of loading the whole result set at once. Cursors are useful for working with different kinds of databases, such as Oracle, SQL Server, MySQL, and so on. You can use cursors with DML statements like Update, Insert, and Delete to manipulate data.
- ❖ **In PL/SQL, there are two kinds of cursors:** Implicit cursors and Explicit cursors. Implicit cursors are automatically created for DML statements and queries that return only one row. You don't need to declare them explicitly. Explicit cursors are defined by the programmers to have more control over the result set. You have to declare them in the declaration section of the PL/SQL block. They are usually used for queries that return more than one row.
- ❖ **There are two types of cursors available in PL/SQL:** Implicit cursors and Explicit cursors . Implicit cursors are available for DML statements in PL/SQL i.e. no need to declare the cursor, and even for the queries that return 1 row. Explicit cursors are defined by the programmers to have more control area on the context area. It has to be defined in the declaration section of the PL/SQL Block. Usually, it is defined on a SELECT Statement and it returns more than one row as output .

Here are the steps involved in creating an explicit cursor:

1. Cursor Declaration for initializing the memory: CURSOR <cursorName> IS SELECT

<Required fields> FROM <tableName>;

2. Cursor Opening to allocate the memory: OPEN <cursorName>;
3. Cursor Fetching to retrieve the data: FETCH <cursorName> INTO <Respective columns>
4. Cursor Closing to release the allocated memory: CLOSE <cursorName>;

Here's an example of how to create an explicit cursor:

```
DECLARE
  empId employees.EMPLOYEEID%type;
  empName employees.EMPLOYEENAME%type;
  empCity employees.EMPLOYEECITY%type;
  CURSOR c_employees is SELECT EMPLOYEEID, EMPLOYEENAME,
EMPLOYEECITY FROM employees;
BEGIN
  OPEN c_employees;
  LOOP
    FETCH c_employees into empId , empName , empCity;
    EXIT WHEN c_employees %notfound;
    dbms_output.put_line (empId || ' ' || empName || ' ' || empCity);
  END LOOP;
  CLOSE c_employees ;
END;
```

12.4 STEPS TO CREATE CURSORS:

Definition: Cursors are used for processing rows individually for a result set returned by a query. They allow operations to be performed on one row at a time.

Advantages:

Fine-Grained Control: Cursors provide fine-grained control over the result set.

Flexibility: They allow for complex data manipulation.

Example:

```
DECLARE cursor1 CURSOR FOR

SELECT * FROM employees;

OPEN cursor1;

FETCH cursor1 INTO emp_id, emp_name;

CLOSE cursor1;
```

Dynamic SQL:

Definition: Dynamic SQL involves generating and executing SQL statements dynamically at runtime, rather than having them hard-coded in a procedure or application.

Advantages:

Flexibility: It allows for the creation of dynamic queries based on changing conditions.

Dynamic Table and Column Names: It enables querying tables and columns whose names are not known in advance.

Example:

```
SET @query = 'SELECT * FROM ' || table_name;

PREPARE stmt FROM @query;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;
```

Database management relies on these procedures, which enable the wrapping of intricate logic, boosting security, enhancing performance, and facilitating automation of tasks within a database setting.

12.5 PROCEDURE:

A Database Management System (DBMS) consists of a set of procedures that can manipulate the data stored in a database. These procedures are either pre-defined by the DBMS or defined by the user, and they can be called by applications or other components of the DBMS to perform various operations on the database. The following are some common types of procedures in a DBMS:

Stored Procedures:

Definition: Stored procedures are pre-compiled SQL statements that are stored in the database. They can be called by name, and they may accept parameters and return results.

Advantages:

Modularity: They promote modular programming by allowing you to encapsulate complex logic in a single procedure.

Performance: They can improve performance as they are precompiled and stored in the database, reducing the need for parsing and optimization.

Security: They can be used to restrict direct access to tables and views, allowing controlled access through procedures.

Example:

```
CREATE PROCEDURE sp_GetCustomerDetails (IN custID INT)

BEGIN
```

```
   SELECT * FROM customers WHERE customer_id = custID;
END;
```

Execution

```
CALL sp_GetCustomerDetails(101);
```

User-Defined Functions (UDFs):

Definition: UDFs are similar to stored procedures, but they return a value. They can be used within SQL queries and can be part of expressions.

Advantages:

Reusability: Functions can be used in multiple queries or procedures.

Encapsulation: They allow for logical grouping and abstraction of operations.

Example:

```
CREATE FUNCTION fn_CalculateTotal (price INT, quantity INT)
RETURNS INT
BEGIN
   RETURN price * quantity;
END;
```

Usage:

```
SELECT fn_CalculateTotal(10, 5); -- Returns 50
```

12.6 SUMMARY:

Creating a PL/SQL program in DBMS are centered around implementing procedural logic, manipulating data, ensuring data integrity, promoting modularity and reusability, and adhering to best practices for readability, security, and performance. These are collectively contribute to the development of robust and efficient database applications.

12.7 KEY WORDS :

- ❖ **DECLARE:** The DECLARE keyword is used to declare variables, constants, and cursors within the PL/SQL program. It establishes the data structures that will be used in the subsequent executable section.
- ❖ **BEGIN:** The BEGIN keyword marks the beginning of the executable section of the PL/SQL block. It signifies the start of the procedural logic and the actual implementation of the program.
- ❖ **END: Definition:** The END keyword signifies the end of the PL/SQL block. It marks the conclusion of the procedural logic and the termination of the program.
- ❖ **PROCEDURE:** The PROCEDURE keyword is used to define a named PL/SQL procedure. A procedure is a modular unit of code that can be called and executed independently. It promotes code modularity and reusability.
- ❖ **FUNCTION:** The FUNCTION keyword is used to define a named PL/SQL function.

A function returns a value and can be called within SQL queries or other PL/SQL code. Like procedures, functions promote modularity and reusability.

12.8 SELF-ASSESSMENT QUESTIONS:

1. Write the steps to create a PL/SQL program?
2. Explain the iterative control cursors?
3. Differences between triggers and procedures?

12.9 FURTHER READINGS:

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000

LESSON - 13

FUNCTIONS, PACKAGES, EXCEPTIONS HANDLING

OBJECTIVES:

After going through this lesson, you will be able to:

- ❖ In database management systems (DBMS), functions, packages, and exception handling play crucial roles in enhancing the efficiency, maintainability, and robustness of database applications.

STRUCTURE OF THE LESSON:

13.1 INTRODUCTION:

Functions in DBMS are subprograms that can be called by SQL statements. They are utilized to typify a particular activity or a bunch of tasks that can be reused in various pieces of the code. Functions can be used to return a single value or a table of values.

Packages are a way of organizing related functions, procedures, and other objects in a single unit. They are used to group related functionality together and provide a namespace for the objects they contain. Packages can be used to simplify code maintenance and improve performance by reducing the number of calls to the database.

Exception handling is a mechanism in DBMS that permits you to deal with mistakes that happen during the execution of a program. It gives an approach to gracefully handle errors and recover from them. In PL/SQL, you can use the EXCEPTION block to handle exceptions. There are two types of exceptions: user-defined and system-defined. User-defined exceptions are created by the programmer, while system-defined exceptions are predefined by Oracle.

13.2 FUNCTIONS:

In database management systems (DBMS), functions play an important role in data processing and manipulation. These allow you to perform certain tasks and calculations within the database. Here are some common function types in DBMS: Aggregate Functions

These functions perform calculations on a set of values and return a single result. Examples include:

SUM(): Calculates the sum of a set of values.

AVG(): Computes the average of a set of values.

MIN(): Returns the minimum value in a set.

MAX(): Returns the maximum value in a set.

COUNT(): Counts the number of rows in a result set.

Scalar Functions: Scalar functions operate on a single value and return a single value. Common scalar functions include:

UPPER(), LOWER(): Convert text to uppercase or lowercase.

LENGTH(), CHAR_LENGTH(): Return the length of a string.

CONCAT(): Concatenates two or more strings.

SUBSTRING(), LEFT(), RIGHT(): Extract parts of a string.

Date and Time Functions: These functions help in manipulating and working with dates and times. Examples include:

NOW(), CURRENT_DATE, CURRENT_TIME: Retrieve the current date and time.

DATEADD(), DATEDIFF(): Add or subtract days, months, or years from dates.

Mathematical Functions: These functions perform mathematical operations on numeric data. Common mathematical functions include:

ROUND(), CEIL(), FLOOR(): Round numbers to the nearest integer.

ABS(): Return the absolute value of a number.

SQRT(): Calculate the square root of a number.

String Functions: String functions manipulate text data. Examples include:

LEFT(), RIGHT(), SUBSTRING(): Extract parts of a string.

CHARINDEX(), LEN(): Find the position of a character or return the length of a string.

REPLACE(): Replace occurrences of a substring within a string.

Conversion Functions: These functions convert data from one type to another. Common conversion functions include:

CAST(): Convert one data type to another.

CONVERT(): Convert data from one type to another in a DBMS-specific way.

User-Defined Functions (UDFs): Some DBMSs allow users to create their own functions using a defined syntax and programming language. These features can be customized to perform specific tasks based on your needs. Window functions: These functions operate on a "window" of rows that are related to the current row. These are used for tasks such as calculating running totals, rankings, and finding moving averages. These functions are important for performing various operations on data stored in the database. They help retrieve, transform, and represent data in meaningful ways and are a fundamental part of database queries and data manipulation.

13.3 PACKAGES:

In the context of a database management system (DBMS), a "package" typically refers to specific functionality provided by a DBMS to facilitate tasks related to database management. Different DBMS systems may have their own packages or libraries. Here are some of the common packages or features you might encounter in a DBMS.

- ❖ **PL/pgSQL (PostgreSQL):** PostgreSQL has a procedural language called PL/pgSQL that allows you to create stored procedures and functions. These can be organized into packages, which are essentially collections of functions and procedures.
- ❖ **PL/SQL (Oracle):** PL/SQL is Oracle's SQL procedural language extension. Oracle uses packages to group related procedures, functions, variables, and other constructs into a named unit.
- ❖ **Stored Procedures and Functions:** Many DBMS systems allow you to create procedures and functions and store them in the database. These procedures and functions can be called from other parts of your application or database.
- ❖ **Database Triggers:** Triggers are a special type of stored procedure that are automatically executed (or "fired") in response to certain events, such as inserting, updating, or deleting data in a table.
- ❖ **Built-in functions and procedures:** Most DBMS systems have a number of built-in functions and procedures that provide common functionality such as mathematical operations, string manipulation, and date and time manipulation.
- ❖ **User-defined functions (UDFs):** Some DBMS systems allow users to define their own functions to extend the functionality of the DBMS.
- ❖ **Standard SQL Functions:** Standard SQL does not itself provide packages, but it does provide a wide range of functions for querying and manipulating data in a database.

13.4 EXCEPTIONS HANDLING:

In a Database Management System (DBMS), exceptions handling refers to the process of dealing with errors or exceptional conditions that may occur during the execution of database operations. These exceptions could be due to various reasons such as data integrity violations, concurrency issues, resource constraints, or system errors. Proper handling of exceptions is crucial to ensure data consistency and system reliability.

Here are some common approaches to exception handling in DBMS:

- ❖ **Transaction Rollback:** If an error occurs during the execution of a transaction, the system can roll back the transaction to a previous consistent state. This ensures that the database remains in a consistent state even in the presence of errors.
- ❖ **Error Codes and Messages:** The DBMS may provide error codes and descriptive error messages that indicate the nature of the error. These codes and messages can be used in your application or user interface to notify the user or take appropriate action.
- ❖ **Logging and Auditing:** The DBMS may log details of exceptions and errors. This log can be used to troubleshoot, monitor, and record system operation.
- ❖ **Limitations and Validation:** A DBMS can apply limits to data to prevent invalid or inconsistent entries. For example, you can enforce referential integrity constraints,

unique constraints, etc.

- **Deadlock detection and resolution:** In a multi-user environment, deadlocks can occur when two or more transactions are waiting for mutually held resources. The DBMS can detect and resolve deadlocks to ensure progress.
- **Resource Management:** The DBMS may handle exceptions related to resource allocation. B. There is insufficient storage space, memory, or other system resources.
- **Isolation Level:** The isolation level of a transaction determines the level of visibility of changes made by other transactions. Different isolation levels require different approaches to dealing with concurrency and potential conflicts.
- **Savepoints:** Save points allow transactions to specify points within the transaction that can be used for partial rollbacks. This is useful for reversing certain parts of a transaction in case of an error.
- **Custom Error Handling:** Some DBMSs allow developers to define custom exception handlers or triggers. These can be used to perform specific actions in response to specific exceptions.
- **Recovery Mechanisms:** A DBMS may have mechanisms to recover from system failures such as: For example, restoring from backups or using transaction logs for point-in-time recovery.

Note that the specific implementation and exception handling features may vary depending on the DBMS used (e.g. MySQL, PostgreSQL, Oracle, etc.) and the programming language or environment used to interact with the database. .

For developers and database administrators, it is important to have a good understanding of the exception handling mechanisms of a particular DBMS in order to build robust and reliable database applications.

13.5 DATABASE TRIGGERS:

A named set of SQL statements that are considered when a data modification (INSERT, UPDATE, DELETE) occurs. If a condition stated within the triggers met, then a prescribed action is taken.

Constraints can be thought of as a special case of triggers. They also are applied automatically as a result of data modification commands. Triggers have three parts, the event, the condi- tion, and the action, and these parts are reflected in the coding structure for triggers.

Definition: Triggers are special types of stored procedures that are automatically executed (or "triggered") in response to certain events on a particular table or view.

Advantages:

Enforcement of Business Rules: Triggers can enforce business rules or perform actions based on changes in the data.

Logging and Auditing: They can be used for logging or auditing purposes.

```
CREATE TRIGGER after_insert

AFTER INSERT ON orders

FOR EACH ROW

BEGIN
```

```
    INSERT INTO order_logs (order_id, action) VALUES (NEW.order_id, 'Inserted');
END;
```

Functions-routines that return values and take input parameters

Procedures-routines that do not return values and can take input or output parameters

Simplified Trigger syntax

```
CREATE TRIGGER trigger_name
(BEFORE I AFTER I INSTEAD OF) (INSERT I DELETE | UPDATE) ON
table_name
[FOR EACH (ROW I STATEMENT)] [WHEN (search condition)] <triggered SQL
statement here>;
```

13.6 TYPES OF TRIGGERS:

Triggers are procedures that are automatically executed by the DBMS in response to changes made to the database. They are used to specify certain integrity and referential constraints that cannot be specified using SQL's constraint mechanism. There are different types of triggers that can be defined for each table in a database. There are six types of triggers that can be defined for each table:

1. AFTER INSERT: Activated after data is inserted into the table.
2. AFTER UPDATE: Takes effect after data in the table has been modified.
3. AFTER DELETE: Enabled after data is deleted/deleted from the table.
4. BEFORE INSERT: Valid before data is inserted into the table.
5. BEFORE UPDATE: Takes effect before any data in the table is modified.
6. BEFORE DELETE: Enabled before data is deleted/deleted from the table.

The difference between BEFORE and AFTER triggers is that BEFORE triggers execute before the triggering event (such as an insert, update, or delete) is executed on the table, whereas AFTER triggers execute after the triggering event has been executed on the table.

13.7 SUMMARY :

The functions, packages, and exception handling in DBMS contribute to the development of robust, modular, and maintainable database applications by addressing issues related to code organization, reusability, security, and error management.

13.8 KEY WORDS :

❖ **String:** A string in DBMS refers to a sequence of characters. Strings are a fundamental data type used to represent textual information within a database. They can store names, addresses, descriptions, or any other type of alphanumeric data.

❖ **Exception handling:** in DBMS refers to the process of managing and responding to errors or exceptional conditions that may arise during the execution of database operations. These exceptional conditions could include things like data integrity violations, constraint violations, or unexpected errors.

- **Package**: It is a container that holds related database objects, such as procedures, functions, types, and variables. It provides a way to organize and structure database code into modular units.

13.9 SELF ASSESSMENT QUESTIONS :

1. Define functions? Write the functions?
2. Explain packages? With example.
3. Write the exception handlings?
4. Discus the different types of triggers?

13.10 FURTHER READINGS :

1. Date, C. J., Database System. 7th Edition, New Delhi: Pearson Education, 2005.
2. Navathe, Elmasri, Fundamentals of Database Systems. 3rd Edition, Essex, UK: Pearson Education Asia, 2000.

LAB MANUAL

LIST OF THE DATA BASE MANAGEMENT SYSTEMS LAB PROGRAMS

1. Create tables department and employee with required constraints.
2. Initially only the few columns (essential) are to be added. Add the remaining columns separately by using appropriate SQL command.
3. Basic column should not be null
4. Add constraint that basic should not be less than 5000.
5. Calculate hra, da, gross and net by using PL/SQL program.
6. The percentage of hra and da are to be stored separately.
7. When the da becomes more than 100%, a message has to be generated and with user permission da has to be merged with basic.
8. Empno should be unique and has to be generated automatically.

1. Create tables department and employee with required constraints.

CREATION OF TABLES:

QUERY:

```
create table emp (empno  number(10) primary key, ename varchar2(10), job
varchar2(10), mgr  number(10),hiredate date, sal number(10),comm  number(10),deptno
number(10));
```

EXPECTED OUTPUT:

Table Created

OUTPUT:

QUERY:

```
Create table dept( deptno number(10) primary key, dname varchar2(20), description
varchar2(40));
```

EXPECTED OUTPUT:

Table Created

OUTPUT:

2. Initially only the few columns (essential) are to be added. Add the remaining columns separately by using appropriate SQL command

QUERY

```
Alter table emp (add basic number(10),hra number(10),da number(10));
```

EXPECTED OUTPUT:

```
Table Altered
```

OUTPUT:

3. Basic column should not be null

QUERY:

```
Create table employee (empno number(10) not null, basic number(10) not null, ename varchar2(10));
Insert into employee values (null,null,'kumar');
```

Note: The Above we kept not null for both empno, basic and we tried to insert null values in to both
columns but it has not accepted.

EXPECTED OUTPUT:

```
Cannot insert null value in basic column( not null constraint violated)
```

OUTPUT:

4. Add constraint that basic should not be less than 5000.

QUERY:

```
alter table employee add check (basic>5000)
```

EXPECTED OUTPUT:

```
Table Altered
```

OUTPUT:

5. Calculate hra, da, gross and net by using PL/SQL program.

BASIC	HRA	DA
15000	12%	8%
12000	10%	6%
9000	7%	4%
OTHERS	5%	200/-

PL/SQL PROGRAM

```
DECLARE
BASIC NUMER
HRA NUMBER;
DA NUMBER;
NET NUMBER;
BEGIN
BASIC := &BASIC_SALARY;
IF BASIC > 15000 THEN
HRA := BASIC * .12;
DA := BASIC * .08;
ELSIF BASIC > 12000 THEN
HRA := BASIC * .1;
DA := BASIC * .06;
ELSIF BASIC > 9000 THEN
HRA := BASIC * ..07;
DA := BASIC * .04;
ELSE
HRA := BASIC * .05;
DA := BASIC * 200;
END IF;
NET := BASIC + HRA + DA;
DBMS_OUTPUT.PUT_LINE (‘BASIC: ‘ || BASIC);
DBMS_OUTPUT.PUT_LINE (‘HRA: ‘ || HRA);
DBMS_OUTPUT.PUT_LINE (‘DA: ‘ || DA);
DBMS_OUTPUT.PUT_LINE (‘NET: ‘ || NET);
END;
```

EXPECTED OUTPUT:

```
pl/sql procedure successfully completed
```

OUTPUT:

6. The percentage of hra and da are to be stored separately.

```
ALTER TABLE EMP ADD HRAP NUMBER(5,2),DAP NUMBER(5,2);
```

7. When the da becomes more than 100%, a message has to be generated and with user permission da has to be merged with basic.

PL/SQL PROGRAM

```
Create or replace trigger da1
Before insert or update on employee
for each row
begin
if :new.da>sal then
 raise_application_error (-20456,'the da is not accepted, so the da is merge to basic’);
end if;
 end;
```

EXPECTED OUTPUT:

```
Trigger Created Successfully.
```

OUTPUT:

8. Empno should be unique and has to be generated automatically.

QUERY:

```
Create or replace sequence s1
Increment by 1
Start with 501
End with 590
No cache
No cycle;
Insert into employee values (s1.nextval,’siva’, 9000, 900, 4500);
Insert into employee values (s1.nextval,’siva’, 9000, 900, 4500);
```

EXPECTED OUTPUT:

```
Sequence created.
```

OUTPUT: