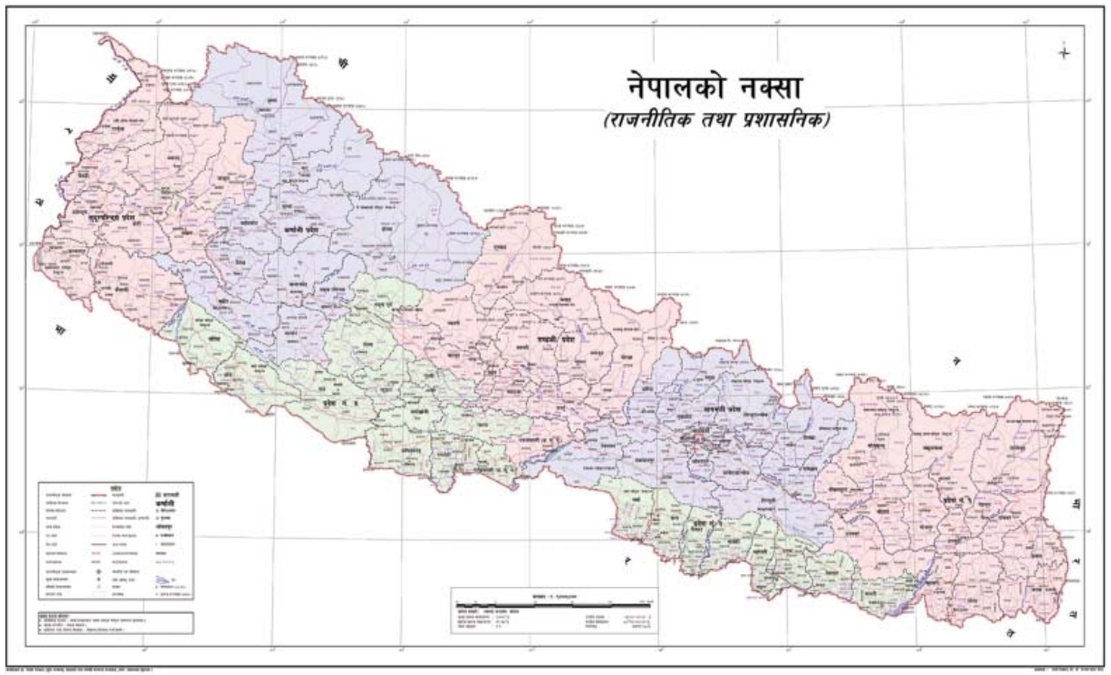
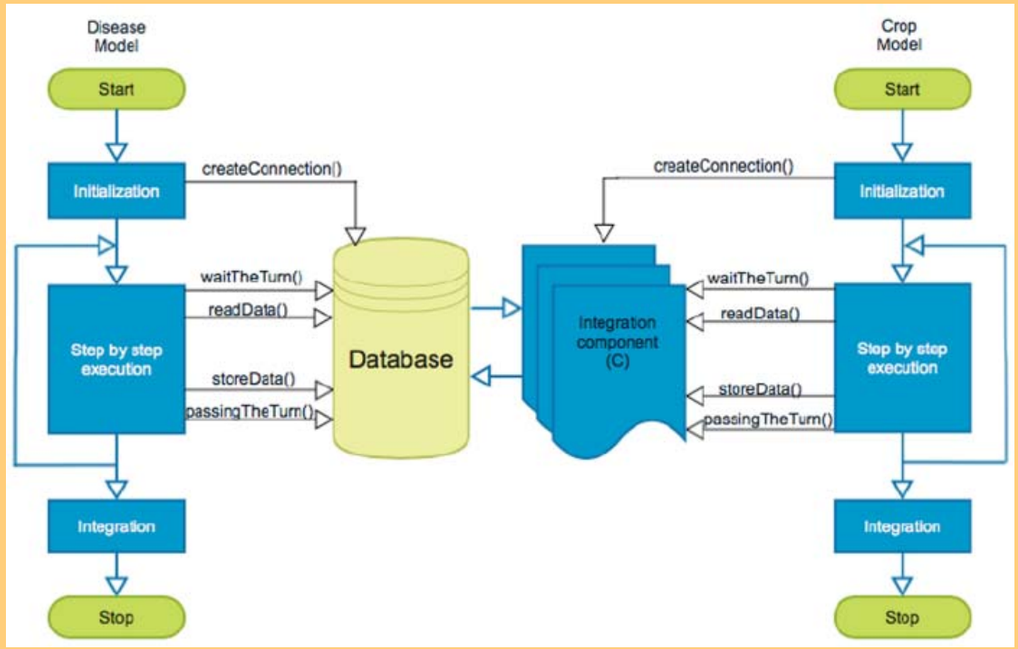


Database Management System



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

Feedback Copy

Technical and Vocational Stream
Learning Resource Material

Database Management System
(Grade 10)

Secondary Level
Computer Engineering

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

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

© Publisher

Layout by Khados Sunuwar

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

Preface

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

In bringing out the learning resource material in this form, the contribution of the Director General of CDC Dr. Lekhnath Poudel, Pro, Dr. Subarna Shakya, Bibha Sthapit, Anil Barma, Yogesh Parajuli, Nabinkrishna Maskey, Rajendra Rokaya, Lina Maharjan, Bimal Thapa, Shankar Yadav is highly acknowledged. The book is written by Satyaram Suwal and the subject matter of the book was edited by Badrinath Timalsina and Khilanath Dhamala. CDC extends sincere thanks to all those who have contributed in developing this book in this form.

This book is a supplimentary learning resource material for students and teachrs. In addition they have to make use of other relevnt materials to ensure all the learning outcomes set in the curriculum. The teachers, students and all other stakeholders are expected to make constructive comments and suggestions to make it a more useful learning resource material.

2076 BS

Ministry of Education, Science and Technology
Curriculum Development Centre

Table of Contents

UNIT-1

Introduction to Database System architecture

Objectives

After completion of this unit students will be able

- To explain the concept of Database System concept
- to state Database Manager and users
- to discuss the needs of organization of database
- to apply the knowledge of Client-Server Architecture

Learning Process and support material:

Learning Process:

- Class demonstration and lecture method.
- Group Discussion and Questionnaire.

Content's Elaboration:

Data: Information in raw or unorganized form (such as alphabets, numbers, or symbols) that refer to, or represent, conditions, ideas, or objects. Data is limitless and present everywhere in the universe.

Information: Information is organized or classified data, which has some meaningful values for the receiver. Information is the processed data on which decisions and actions are based.

For the decision to be meaningful, the processed data must qualify for the following characteristics −

- **Timely** − Information should be available when required.
- **Accuracy** − Information should be accurate.
- **Completeness** − Information should be complete.

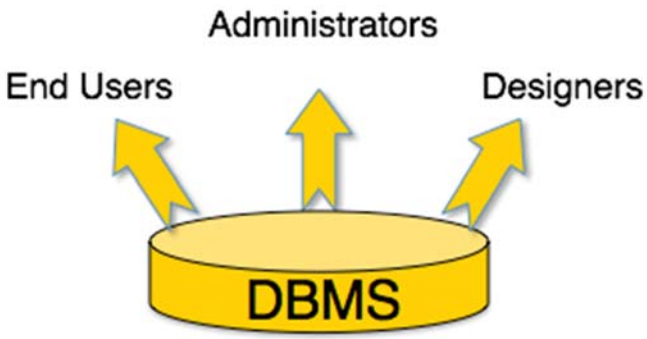
Database: A **Database** is a collection of related data organized in a way that data can be easily accessed, managed and updated. Any piece of information can be a data, for example name of your school. Database is actually a place where related piece of information is stored and various operations can be performed on it.

Database Management system(DBMS)

A DBMS is software that allows creation, definition and manipulation of database. DBMS is actually a tool used to perform any kind of operation on data in database. DBMS also provides protection and security to database. It maintains data consistency in case of multiple users. Here are some examples of popular DBMS, MySQL, Oracle, Microsoft access and dBase

User:

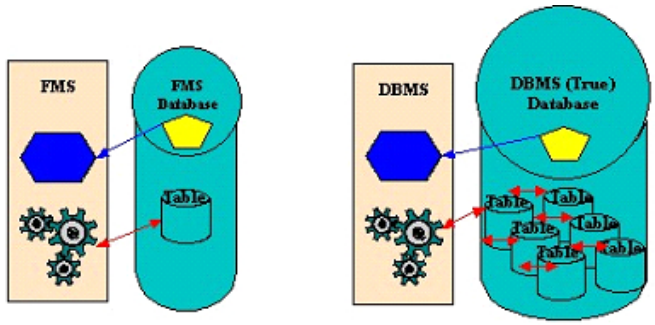
A typical DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows:



- **Administrators** − Administrators maintain the DBMS and are responsible for administrating the database. They are responsible to look after its usage and by whom it should be used. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.
- **Designers** − Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in what format. They identify and design the whole set of entities, relations, constraints, and views.
- **End Users** − End users are those who actually reap the benefits of having a DBMS. End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

File Management System Vs Database Management System

A File Management System (FMS) is a Database Management System that allows access to single files or tables at a time. FMS's accommodate flat files that have no relation to other files. The FMS was the predecessor for the Database Management System (DBMS), which allows access to multiple files or tables at a time (see figure below)



FMS versus DBMS Comparison Diagram (Figure 1)

A Database Management System (DMS): is a combination of computer software, hardware, and information designed to electronically manipulate data via computer processing. Two types of database management systems are DBMS's

Advantages of flat-file Systems: Despite of the number of drawbacks of flat-file systems they are beneficial in many situation and database management system may not be much suitable in such situations. benefits of file processing systems or we can limitations of database management systems are described below:

- Initial investment: To use database systems we need to purchase database management system based on our requirement. Besides this we may need to purchase a powerful computer to run it as database server. Due to this reason initial investment is high if we use database management system.
- Dedicated staff: To manage database system efficiently we need to hire a skilled dedicated staff called database administrator. file processing systems do not need not hire such dedicated
- Overhead: We need to update and maintain database systems in periodic basis to make it efficient in new technologies which is not needed in flat-file systems. Thus overhead cost is high for database management systems.

Drawbacks of File system:

- Data Isolation: Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.
- Duplication of data – Redundant data
- Dependency on application programs – Changing files would lead to change in application programs.

Advantages of DBMS

The database management system has promising potential advantages, which are explained below:

1. Controlling Redundancy: In file system, each application has its own private files, which cannot be shared between multiple applications. this can often lead to considerable redundancy in the stored data, which results in wastage of storage space. By having centralized database most of this can be avoided. It is not possible that all redundancy should be eliminated. Sometimes there are sound business and technical reasons for· maintaining multiple copies of the same data. In a database system, however this redundancy can be controlled.

For example: In case of college database, there may be the number of applications like General Office, Library, Account Office, Hostel etc. Each of these applications may maintain the following information into own private file applications:

General Office	Library	Hostel	Account Office
Roll No Name Class Father_Name Date_of_Birth Address Phone No Previous Record Attendance Marks etc.	Roll No Name Class Address Date at Birth Phone No No of books issued Fine etc	Roll No Name Class Father_Name Date of Birth Address Phone No Mess bill RoomNo etc.	Roll No Name Class Address Phone No Fee Installments Discount Balance Total etc.

It is clear from the above file systems, that there is some common data of the student which has to be mentioned in each application, like Roll number, Name, Class, Phone No~ Address etc. This will cause the problem of redundancy which results in wastage of storage space and difficult to maintain, but in case of centralized database,

data can be shared by number of applications and the whole college can maintain its computerized data with the following database:

General Office	Library	Hostel	Account Office
Rollno Name Class Father_Name Address Phone - No Date_of_birth Previous_Record Attendance Marks etc.	Rollno No_of_books_issued Fine etc.	Rollno RoomNo Mess_Bill etc.	Rollno Fee Installments Discount Balance Total etc.

It is clear in the above database that Rollno, Name, Class, Father_Name, Address, Phone_No, Date of_birth which are stored repeatedly in file system in each application, need not be stored repeatedly in case of database, because every other application can access this information by joining of relations on the basis of common column i.e. Rollno. Suppose any user of Library system need the Name, Address of any particular student and by joining of Library and General Office relations on the basis of column Rollno he/she can easily retrieve this information.

Thus, we can say that centralized system of DBMS reduces the redundancy of data to great extent but cannot eliminate the redundancy because RollNo is still repeated in all the relations.

2. Integrity can be enforced: Integrity of data means that data in database is always accurate, such that incorrect information cannot be stored in database. In order to maintain the integrity of data, some integrity constraints are enforced on the database. A DBMS should provide capabilities for defining and enforcing the constraints.

For Example: Let us consider the case of college database and suppose that college having only BTech, MTech, MSc, BCA, BBA and BCOM classes. But if a user enters the class MCA, then this incorrect information must not be stored in database and must be prompted that this is an invalid data entry. In order to enforce this, the integrity constraint must be applied to the class attribute of the student entity. But, in

case of file system this constraint must be enforced on all the application separately (because all applications have a class field).

3. Inconsistency can be avoided: When the same data is duplicated and changes are made at one site, which is not propagated to the other site, it gives rise to inconsistency and the two entries regarding the same data will not agree. At such times the data is said to be inconsistent. So, if the redundancy is removed chances of having inconsistent data is also removed.

Let us again, consider the college system and suppose that in case of General_Office file it is indicated that Roll_Number 5 lives in Amritsar but in library file it is indicated that

Roll_Number 5 lives in Jalandhar. Then, this is a state at which title two entries of the same object do not agree with each other (that is one is updated and other is not). At such time the database is said to be inconsistent.

An inconsistent database is capable of supplying incorrect or conflicting information. So there should be no inconsistency in database. It can be clearly shown that inconsistency can be avoided in centralized system very well as compared to file system ..

4. Data can be shared: As explained earlier, the data about Name, Class, Father __name etc. of General_Office is shared by multiple applications in centralized DBMS as compared to file system so now applications can be developed to operate against the same stored data. The applications may be developed without having to create any new stored files.

5. Standards can be enforced : Since DBMS is a central system, so standard can be enforced easily may be at Company level, Department level, National level or International level. The standardized data is very helpful during migration or interchanging of data. The file system is an independent system so standard cannot be easily enforced on multiple independent applications.

6. Restricting unauthorized access: When multiple users share a database, it is likely that some users will not be authorized to access all information in the database. For example, account office data is often considered confidential, and hence only authorized persons are allowed to access such data. In addition, some users may be permitted only to retrieve data, whereas other are allowed both to retrieve and to

update. Hence, the type of access operation retrieval or update must also be controlled. Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database. A DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions. The DBMS should then enforce these restrictions automatically.

7. Solving Enterprise Requirement than Individual Requirement: Since many types of users with varying level of technical knowledge use a database, a DBMS should provide a variety of user interface. The overall requirements of the enterprise are more important than the individual user requirements. So, the DBA can structure the database system to provide an overall service that is "best for the enterprise".

For example: A representation can be chosen for the data in storage that gives fast access for the most important application at the cost of poor performance in some other application. But, the file system favors the individual requirements than the enterprise requirements

8. Providing Backup and Recovery: A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update program, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the program started executing.

9. Cost of developing and maintaining system is lower: It is much easier to respond to unanticipated requests when data is centralized in a database than when it is stored in a conventional file system. Although the initial cost of setting up of a database can be large, but the cost of developing and maintaining application programs to be far lower than for similar service using conventional systems. The productivity of programmers can be higher in using non-procedural languages that have been developed with DBMS than using procedural languages.

10. **Concurrency Control:** DBMS systems provide mechanisms to provide concurrent access of data to multiple users.

Disadvantages of DBMS:

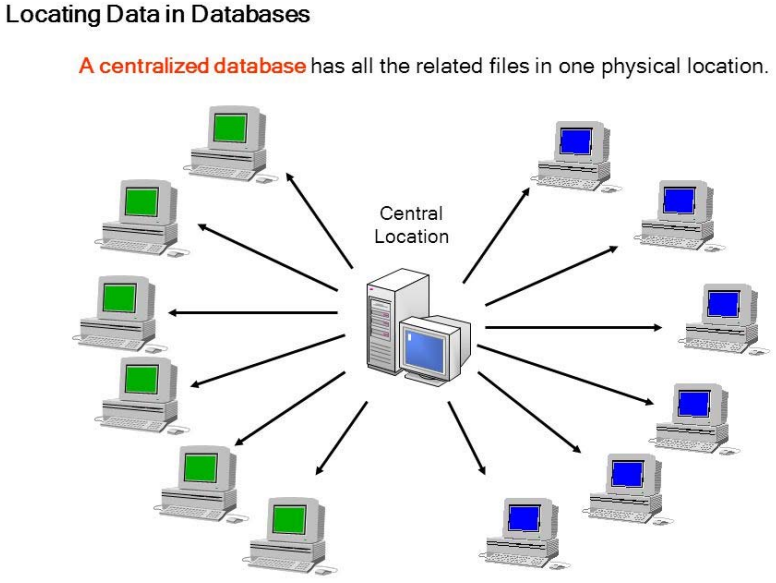
- DBMS implementation cost is high compared to the file system
- Complexity: Database systems are complex to understand
- Performance: Database systems are generic, making them suitable for various applications. However, this feature affects their performance for some applications

Database Architecture

Centralized database

Earlier architectures used mainframe computers to provide the main processing for all system functions, including user application programs and user interface programs, as well as all the DBMS functionality. The reason was that most users accessed such systems via computer terminals that did not have processing power and only provided display capabilities. Therefore, all processing was

performed remotely on the computer system, and only display information and controls were sent from the computer to the display terminals, which were connected to the central computer via various types of communications networks.

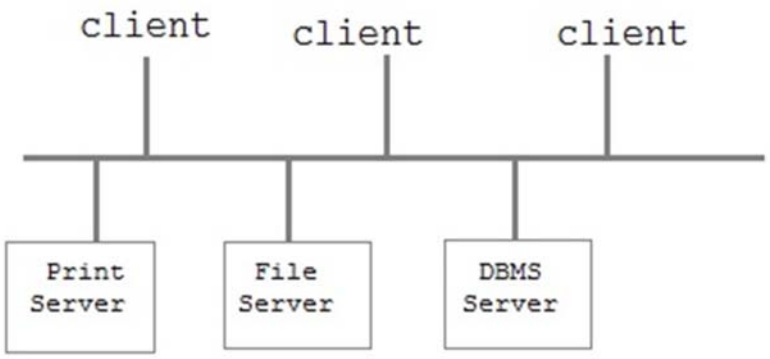


A**centralized** DBMS in which all the DBMS functionality, application program execution, and user interface processing were carried out on one machine. Figure 2 illustrates the physical components in a centralized architecture. Gradually, DBMS systems started to exploit the available processing power at the user side, which led to client/server DBMS architectures.

Client Server Architecture

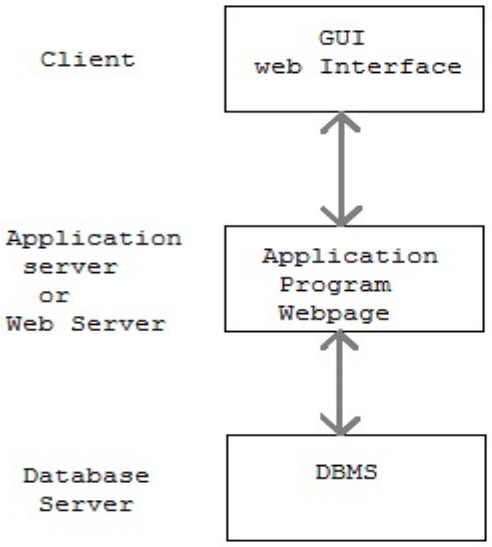
- Client Server architecture is logically divided into two types.
- Logical two-tier Client / Server architecture
- Logical three-tier Client / Server architecture

Two-tier Client / Server Architecture



Two-tier Client / Server architecture is used for User Interface program and Application Programs that runs on client side. An interface called ODBC (Open Database Connectivity) provides an API that allow client side program to call the dbms. Most DBMS vendors provide ODBC drivers. A client program may connect to several DBMS's. In this architecture some variation of client is also possible for example in some DBMS's more functionality is transferred to the client including data dictionary, optimization etc. Such clients are called Data server.

Three-tier Client / Server Architecture



Three-tier Client / Server database architecture is commonly used architecture for web applications. Intermediate layer called Application server or Web Server stores the web connectivity software and the business logic(constraints) part of application used to access the right amount of data from the database server. This layer acts like medium for sending partially processed data between the database server and the client.

Applications where we use Database Management Systems are:

Database System are widely used in different areas because of their numerous advantages. Some of the most common database application are listed here.

- **Telecom**: There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
- **Industry**: Where it is a manufacturing unit, warehouse or distribution center, each one needs a database to keep the records of ins and outs. For example, distribution center should keep a track of the product units that supplied into the center as well as the products that got delivered out from the distribution center on each day; this is where DBMS comes into picture.

- **Banking System**: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
- **Education sector**: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.
- **Online shopping**: You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

Key Points

- Data is a raw or unorganized form that refer to any idea or object.
- Information is meaningful values for the receiver.
- Database is a collection of related data in such a way that data can be easily accessed, managed and updated.
- A DBMS is software that allows creation, definition and manipulation of database.
- A centralized DBMS in which all the DBMS functionality, application program execution, and user interface processing were carried out on one machine
- Client/Server architecture, the processing power of the computer system at the user's end utilized by processing the user-interface on that system.
- A client is a computer system that sends request to the server connected to the network
- Server is a computer system that receive the request, process it, and returns the requested information back to the client.

Glossary:

- Redundancy: Data repetition
- Integrity: Accurate data in database
- Inconsistency: Data is duplicate and changes are made at one site is not reflect to other site.
- Backup: A copy of a file or other item of data made in case the original is lost or damaged.
- ODBC driver: An ODBC driver uses the Open Database Connectivity (ODBC) interface by Microsoft that allows applications to access data in database management systems (DBMS) using SQL as a standard for accessing the data.
- Data server: Data server is the phrase used to describe computer software and hardware (a database platform) that delivers database services.
- Web Server: It enables a computer to host one or more websites that can be accessed over the Internet using a Web browser.

Exercises:

a. Very Short Question:
   a. What is Data in Database ?
   b. Define information
   c. What is user ?
   d. who is administrator ?
   e. Give any two examples of Database
b. Short Question
   a. What is Database Management System ?
   b. List any four DBMS .
   c. What is mean by data redundancy ?
   d. Define data integrity with example
   e. What is data integrity ?
c. Long Question
   a. Define user, administrator, designer and end users.
   b. What is DBMS? Write the advantage of DBMS
   c. Differentiate between client server and centralized database system
   d. explain difference types of client server architecture in detail
   e. Write the application area of Database System.

References:

https://www.studytonight.com/dbms/architecture-of-database
https://www.tutorialspoint.com/dbms/dbms_data_models.htm
https://www.wikipedia.com

UNIT -2

Entity Relationship Model (ER-Mode)

Objective:

After completion of this unit students will be able

- To distinguish between entity and attributes
- To identify entity set and keys
- To practice the knowledge of mapping Cardinalities

Learning Process and support material:

- Class demonstration with chart paper drawings.
- Case Study

Content's Elaboration:

Entity Relationship Model

Entity–relationship modeling was developed for database design by Peter Chen and published in a 1976 paper. However, variants of the idea existed previously. Some ER models show super and subtype entities connected by generalization-specialization relationships, and an ER model can be used also in the specification of domain-specific ontologies.

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity

Entity Relationship Diagram

An entity-relationship (ER) diagram is a graphical representation of entities (which will become your tables) and their relationships to each other.

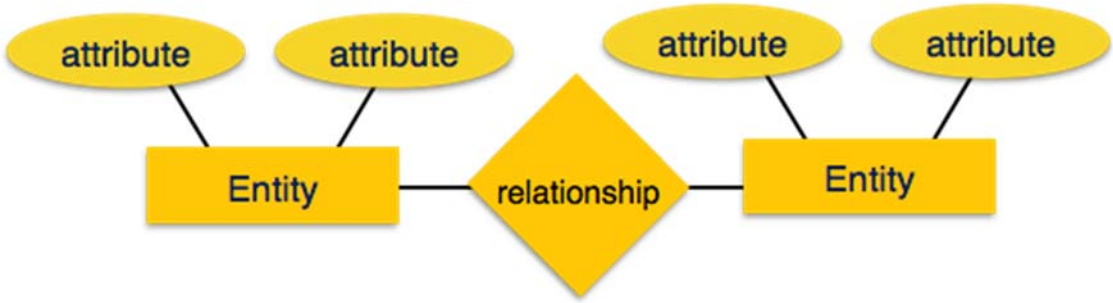
ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

ER Model is based on −

- **Entities** and their *attributes.*
- **Relationships** among entities.

These concepts are explained below.

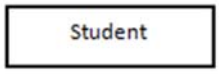


Entity

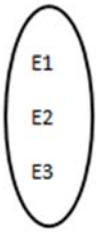
An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.

Entity Set:

An Entity is an object of Entity Type and set of all entities is called as entity set. e.g; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



Entity Type



Entity Set

Attribute:

An attribute defines the information about the entity that needs to be stored. An attribute describes the facts, details or characteristics of an entity. If the entity is employee, attributes could include: name, employee ID, health plan enrollment and work location.

Types of Attributes

Attributes can be classified as having many parts to them or just a single unbreakable attribute.

Simple and Composite Attributes

The composite attribute is an attribute that can be subdivided into other single attributes with meanings of their own. A simple attribute is just an attribute that cannot be subdivided into parts.

Example:

Imagine from the entity **Student** that instead of having the three attributes: ***stu_LastName, stu_MiddleName, stu_FirstName*** it had one attribute called ***stu_Name***. The attribute ***stu_Name*** would be considered a composite attribute since it can be subdivied into the other three attributes: ***stu_LastName, stu_MiddleName, stu_FirstName***. The rest of attributes would be considering single attributes since they can't be subdivided into parts.

Single-valued and multi-valued Attributes

Attributes can be classified as single or multi-value. The single-value attribute can only have one value, while the multi-valued attributes usually can store multiple data in them.

Example:

In the entity **Student**, ***stu_Address*** could be considered a multi-value attribute since a student could have multiple addresses where he lives at. An example of a single-value attribute would be ***stu_LastName*** since a student usually has josiose last name that uniquely identifies him/her.

Derived Attributes

The last category that attributes can be defined is called a derived attribute, where one attribute is calculated from another attribute. The derived attribute may not be stored in the database but rather calculated using algorithm.

Example:

In the entity **Student**, ***stu_Age*** would be considered a derived attribute since it could be calculated using the student's date of birth with the current date to find their age.

Keys and non-keys Attributes

In every entity an attribute or grouped attributes uniquely identify that entity. These attributes are the key attributes and range from Primary key (single attribute identifier) to a Composite Key (Multi attribute Identifier). The rest of the attributes

after the identifier are considered the non-key attributes or descriptors, which just describe the entity.

Example:

Above in the table Student there is only one unique identifier, ***stu_LastName,*** which is the primary key of the table. The rest of the attributes are descriptors.

Database Keys

Keys are very important part of Relational database. They are used to establish and identify relation between tables. They also ensure that each record within a table can be uniquely identified by combination of one or more fields within a table.

Super Key

Super Key is defined as a set of attributes within a table that uniquely identifies each record within a table. Super Key is a superset of Candidate key.

Candidate keys

Candidate keys are defined as the set of fields from which primary key can be selected. It is an attribute or set of attribute that can act as a primary key for a table to uniquely identify each record in that table.

Primary Key

Primary key is a candidate key that is most appropriate to become main key of the table. It is a key that uniquely identify each record in a table.

Relationship − The logical association among entities is called ***relationship***. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

Mapping cardinalities −

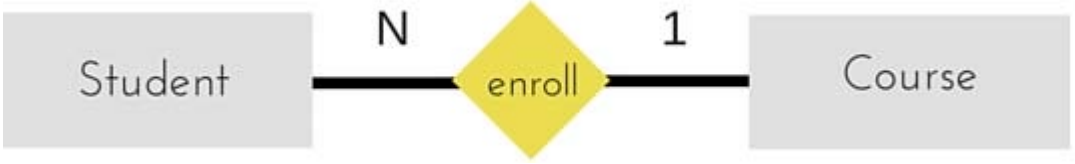
- one to one
- one to many
- many to one
- many to many

The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in relationship.

One to Many: It reflects business rule that one entity is associated with many number of same entity. The example for this relation might sound a little weird, but this means that one student can enroll to many courses, but one course will have one Student.

The above diagram describes that one student can enroll for only one course.

Many to One: It reflects business rule that many entities can be associated with just one entity. For example, Student enrolls for only one Course but a Course can have many Students.



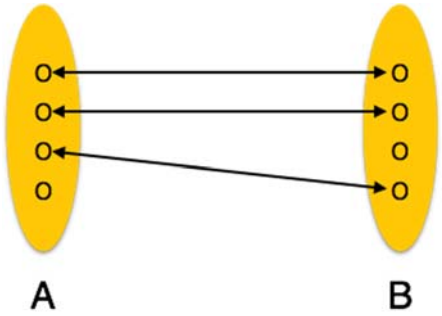
The above diagram describe many students can enroll in one course.

Many to Many:

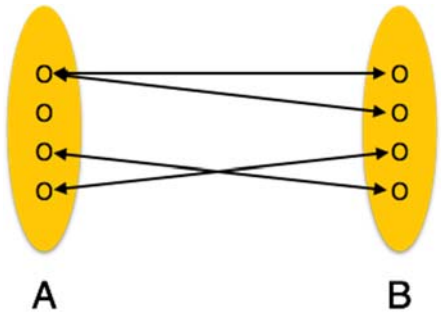
Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

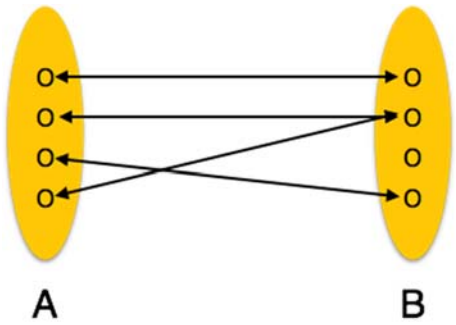
- **One-to-one** − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



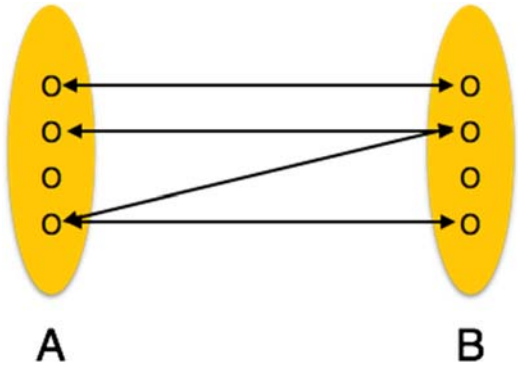
- **One-to-many** − One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



- **Many-to-one** − More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



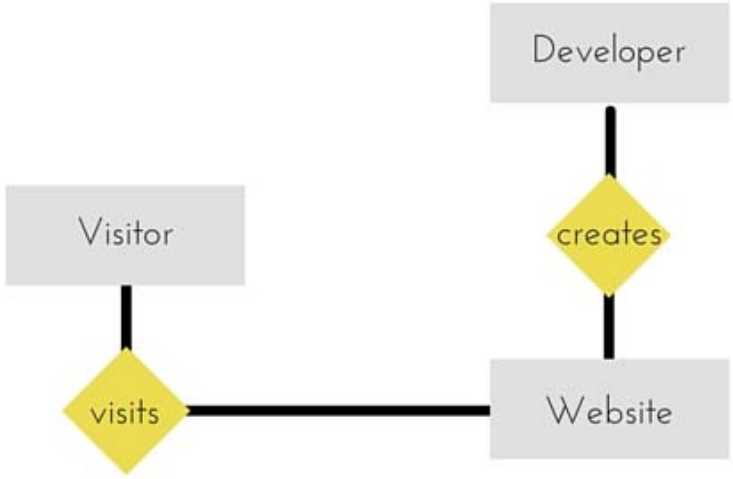
- **Many-to-many** − One entity from A can be associated with more than one entity from B and vice versa.



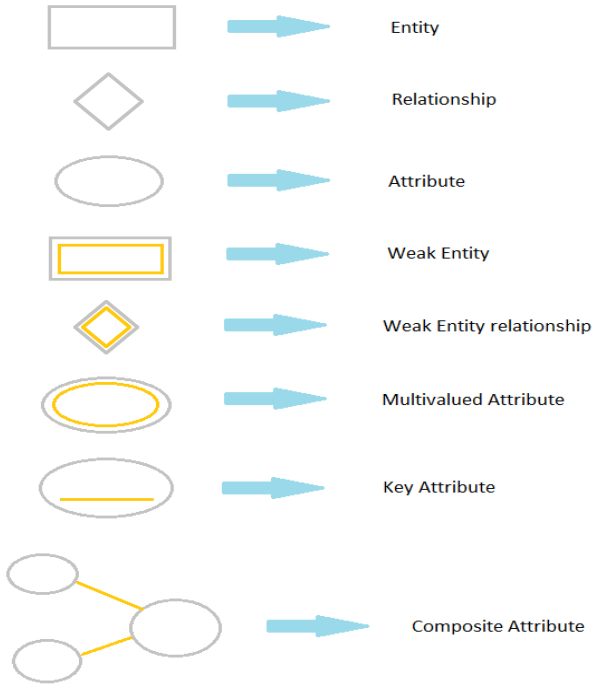
The above diagram represents that many students can enroll for more than one courses.

E-R Diagram

ER-Diagram is a visual representation of data that describes how data is related to each other.



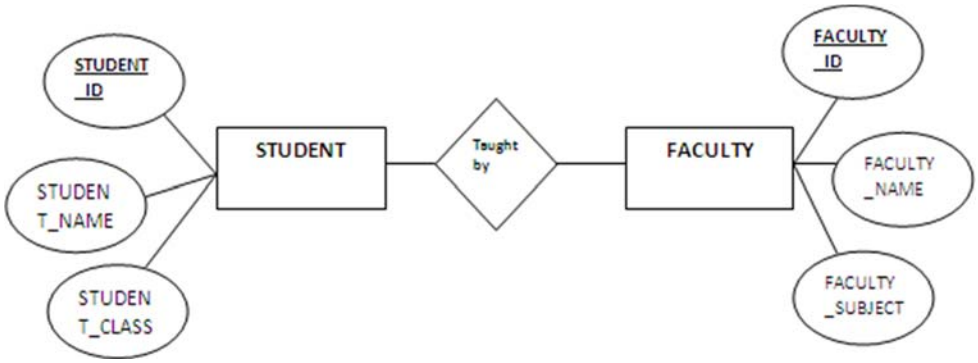
Symbols and Notations



Components of E-R Diagram

The E-R diagram has three main components.

1) Entity
2) Attribute
3) Relationship



KEY POINTS

- **Entity-Relationship (ER)**diagram, is a graphical representation of entities (which will become your tables) and their relationships to each other.
- **An entity** in an ER Model is a real-world entity having properties called **attributes**
- **An attribute** defines the information about the entity that needs to be stored
- **Keys** are very important part of Relational database. They are used to establish and identify relation between tables.
- **Super Key** is defined as a set of attributes within a table that uniquely identifies each record within a table
- **Candidate keys** are defined as the set of fields from which primary key can be selected
- **Primary key** is a candidate key that is most appropriate to become main key of the table.
- The logical association among entities is called **relationship.**
- **Cardinality** defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.
- **The E-R diagram** has three main components.
  1) Entity
  2) Attribute
  3) Relationship

Glossary:

- **Ontology** :An **ontology** is a formal naming and definition of the types, properties, and interrelationships of the entities that really exist in a particular domain of discourse.
- Foreign key: A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables. It acts as a cross-reference between tables because it references the primary key of another table, thereby establishing a link between them.

Exercises:

1. **Very Short Question**
    a. Define entity in Database System
    b. What is relationship mean in Database System
    c. What is super key ?
    d. Define one to one relationship
    e. What is cardinalities means?
2. **Short Question**
    a. List out entity sets use in ER diagram
    b. Write an example of ER diagram
    c. What is key in Database system? list out the different keys use in Database.
    d. Define primary keys
    e. Give an example of many to many relationships with example.
3. **Long question**
    a. Write an ER diagram of student and teacher
    b. Explain different types of relationship with examples.

what is mapping cardinalities? Discuss with an example

References.

https://www.tutorialspoint.com/dbms/dbms_data_models.htm

https://en.wikipedia.org/wiki/Ontology_(information_science)

https://www.techopedia.com/definition/7272/foreign-key

UNIT-3

Introduction to Relational Database, SQL and Relational Model Basics

Objectives:

After completion of this unit students will be able

- To memorize Relational Database Model
- To tell Structure Query Language.

Learning Process and Study Materials

- Class demonstration and practical
- Computer lab with RDBMS software.

Content's Elaboration:

Entity-Relationship model

An entity-relationship model (ERM) is a theoretical and conceptual way of showing data relationships in software development. ERM is a database modeling technique that generates an abstract diagram or visual representation of a system's data that can be helpful in designing a relational database. These diagrams are known as entity-relationship diagrams, ER diagrams or ERDs. Entity-relationship patterns were first proposed by Peter Pin-Shan Chen of the Massachusetts Institute of Technology (MIT) in 1976.

The building blocks of an ERD are entities, relationships and attributes. Entities have entity types, which are known as instances of the corresponding entities. Each entity type can exist independently of another; for example, the entity "vehicle" can have the entity types "car" and "bus." Relationship is the property that links the entity types together. For example, the entity type husband is related to the entity type wife by a relationship known as "is-married-to." Attributes are properties that belong to the entity types as well as to the relationships. There are a number of ER diagramming tools available on the market. The most common ones are MySQL Workbench and OpenModelSphere.

Relational database

Relational database is a collection of organized set of tables from which data can be accessed easily. Relational Database is most commonly used database. It consists of number of tables and each table has its own primary key.

Table

In Relational database, a table is a collection of data elements organized in terms of rows and columns. A table is also considered as convenient representation of relations. But a table can have duplicate tuples while a true relation cannot have duplicate tuples. Table is the simplest form of data storage. Below is an example of Employee table.

ID	Name	Age	Salary
1	Ram	34	13000
2	Pema	28	15000
3	John	20	18000
4	Yadav	42	19020

Record

A single entry in a table is called a Record or Row. A Record in a table represents set of related data. For example, the above Employee table has 4 records. Following is an example of single record.

RAM	34	13000

Field

A table consists of several records(row); each record can be broken into several smaller entities known as Fields. The above Employee table consist of four fields, ID, Name, Age and Salary.

Column

In **Relational** table, a column is a set of value of a particular type. The term Attribute is also used to represent a column. For example, in Employee table, Name is a column that represent names of employee.

Name
Adam
Alex
Stuart
Ross

Students Table

Student	ID*
John Smith	084
Jane Bloggs	100
John Smith	182
Mark Antony	219

Activities Table

ID*	Activity1	Cost1	Activity2	Cost2
084	Tennis	$36	Swimming	$17
100	Squash	$40	Swimming	$17
182	Tennis	$36		
219	Swimming	$15	Golf	$47

Above diagram describe the relationship between table in relational database model.

Constraints in Relational Model

While designing Relational Model, we define some conditions which must hold for data present in database are called Constraints. These constraints are checked before performing any operation (insertion, deletion and updation) in database. If there is a violation in any of constrains, operation will fail.

Domain Constraints: These are attribute level constraints. An attribute can only take values which lie inside the domain range. e.g.,; If a constrains AGE>0 is applied on STUDENT relation, inserting negative value of AGE will result in failure.

Key Integrity: Every relation in the database should have at least one set of attributes which defines a tuple uniquely. Those set of attributes is called key. e.g.; ROLL_NO in STUDENT is a key. No two students can have same roll number. So a key has two properties:

- It should be unique for all tuples.
- It can't have NULL values.

Referential Integrity: When one attribute of a relation can only take values from other attribute of same relation or any other relation, it is called referential integrity. Let us suppose we have 2 relations

STUDENT

ROLL_NO	NAME	ADDRESS	PHONE	AGE	BRANCH_CODE
1	RAM	DELHI	9455123451	18	CS
2	RAMESH	GURGAON	9652431543	18	CS
3	SUJIT	ROHTAK	9156253131	20	ECE
4	SURESH	DELHI		18	IT

BRANCH

BRANCH_CODE	BRANCH_NAME
CS	COMPUTER SCIENCE
IT	INFORMATION TECHNOLOGY
ECE	ELECTRONICS AND COMMUNICATION ENGINEERING
CV	CIVIL ENGINEERING

BRANCH_CODE of STUDENT can only take the values which are present in BRANCH_CODE of BRANCH which is called referential integrity constraint. The

relation which is referencing to other relation is called REFERENCING RELATION (STUDENT in this case) and the relation to which other relations refer is called REFERENCED RELATION (BRANCH in this case).

Structure Query Language

SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus. SQL comes as a package with all major distributions of RDBMS.

SQL comprises both data definition and data manipulation languages. Using the data definition properties of SQL, one can design and modify database schema, whereas data manipulation properties allows SQL to store and retrieve data from database.

Code Rules

Code rules were proposed by E.F. Code which should be satisfied by relational model.

1. **Information Rule:** Data stored in Relational model must be a value of some cell of a table.
2. **Guaranteed Access Rule:** Every data element must be accessible by table name, its primary key and name of attribute whose value is to be determined.
3. **Systematic Treatment of NULL values:** NULL value in database must only correspond to missing, unknown or not applicable values.
4. **Active Online Catalog:** Structure of database must be stored in an online catalog which can be queried by authorized users.
5. **Comprehensive Data Sub-Language Rule:** A database should be accessible by a language supported for definition, manipulation and transaction management operation.
6. **View Updating Rule:** Different views created for various purposes should be automatically updatable by the system.
7. **High level insert, update and delete rule:** Relational Model should support insert, delete, update etc. operations at each level of relations. Also, set operations like Union, Intersection and minus should be supported.
8. **Physical data independence:** Any modification in the physical location of a table should not enforce modification at application level.
9. **Logical data independence:** Any modification in logical or conceptual schema of a table should not enforce modification at application level. For

example, merging of two tables into one should not affect application accessing it which is difficult to achieve.

10. **Integrity Independence:** Integrity constraints modified at database level should not enforce modification at application level.
11. **Distribution Independence:** Distribution of data over various locations should not be visible to end-users.
12. **Non-Subversion Rule:** Low level access to data should not be able to bypass integrity rule to change data.

Join in SQL

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. **Join** Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is **(n-1)** where **n**, is number of tables. A table can also join to itself known as, **Self-Join**.

Types of Join

The following are the types of JOIN that we can use in SQL.

- Inner
- Outer
- Left
- Right

Outer JOIN

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

- Left Outer Join
- Right Outer Join
- Full Outer Join

Left Outer Join

The left outer join returns a result table with the **matched data** of two tables then remaining rows of the **left** table and null for the **right** table's column.

- Left Outer Join syntax is,
- SELECT column-name-list
- from*table-name1*

LEFT OUTER JOIN

table-name2
on table-name1.column-name = table-name2.column-name;
Left outer Join Syntax for **Oracle** is,
select column-name-list
from*table-name1*,
table-name2
on table-name1.column-name = table-name2.column-name(+);

Example of Left Outer Join

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

The **class_info** table,

ID	Address
1	KATHAMANDU
2	BHAKTAPUR
3	POKHARA
7	CHITWAN
8	PATAN

Left Outer Join query will be,

SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id=class_info.id);

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	KATHAMANDU
2	adam	2	BHAKTAPUR
3	alex	3	POKHARA
4	anu	null	null
5	ashish	null	null

Right Outer Join

The right outer join returns a result table with the **matched data** of two tables then remaining rows of the **right table** and null for the **left** table's columns.

Right Outer Join Syntax is,

select column-name-list

from*table-name1*

Right Outer Join

table-name2

on table-name1.column-name = table-name2.column-name;

Right outer Join Syntax for Oracle is,

select column-name-list

from*table-name1*,

table-name2

on table-name1.column-name(+) = table-name2.column-name;

Example of Right Outer Join

The class table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

The **class_info** table,

ID	**Address**
1	KATHAMANDU
2	BHAKTAPUR
3	POKHARA
7	CHITWAN
8	PATAN

Right Outer Join query will be,

SELECT * FROM class RIGHT OUTER JOIN class_info on (class.id=class_info.id);

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	KATHAMANDU
2	adam	2	BHAKTAPUR
3	alex	3	POKHARA
null	null	7	CHITWAN
null	null	8	PATAN

INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the query.
Inner Join Syntax is,
SELECT column-name-list
from*table-name1*

INNER JOIN

table-name2
WHERE table-name1.column-name = table-name2.column-name;
Example of Inner JOIN
The **class** table,

ID	NAME
1	abhi
2	adam
3	alex

4	anu

The **class_info** table,

ID	Address
1	KATHAMANDU
2	BHAKTAPUR
3	POKHARA

Inner JOIN query will be,

SELECT * from class, class_info where class.id = class_info.id;

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	KATHAMANDU
2	adam	2	BHAKTAPUR
3	alex	3	POKHARA

Structured Query Language(SQL)

SQL is a standard computer language for relational database management and data manipulation. SQL is used to query, insert, update and modify data. Most relational databases support SQL, which is an added benefit for database administrators (DBAs), as they are often required to support databases across several different platforms.

First developed in the early 1970s at IBM by Raymond Boyce and Donald Chamberlin, SQL was commercially released by Relational Software Inc. (now known as Oracle Corporation) in 1979. The current standard SQL version is voluntary, vendor-compliant and monitored by the American National Standards

Institute (ANSI). Most major vendors also have proprietary versions that are incorporated and built on ANSI SQL, e.g., SQL*Plus (Oracle), and Transact-SQL (T-SQL) (Microsoft).

One of the most fundamental DBA rites of passage is learning SQL, which begins with writing the first SELECT statement or SQL script without a graphical user interfaces (GUI). Increasingly, relational databases use GUIs for easier database management, and queries can now be simplified with graphical tools, e.g., drag-and-drop wizards. However, learning SQL is imperative because such tools are never as powerful as SQL.

SQL defines following data languages to manipulate data of RDBMS.

DDL: Data Definition Language

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Command	Description
create	to create new table or database
alter	for alteration
truncate	delete data from table
drop	to drop a table
rename	to rename a table

DML: Data Manipulation Language

DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Command	Description
insert	to insert a new row
update	to update existing row

delete	to delete a row
merge	merging two rows or two tables

TCL: Transaction Control Language

These commands are to keep a check on other commands and their effect on the database. These commands can annul changes made by other commands by rolling back to original state. It can also make changes permanent.

Command	Description
commit	to permanently save
rollback	to undo change
save point	to save temporarily

DCL: Data Control Language

Data control language provides command to grant and take back authority.

Command	Description
grant	grant permission of right
revoke	take back permission.

Key points:

- An entity-relationship model (ERM) is a theoretical and conceptual way of showing data relationships in software development
- Relational database is a collection of organized set of tables from which data can be accessed easily. Relational Database is most commonly used database
- A table is a collection of data elements organized in terms of rows and columns
- A single entry in a table is called a Record or Row. A Record in a table represents set of related data.
- A table consists of several records(row); each record can be broken into several smaller entities known as Fields.
- While designing Relational Model, we define some conditions which must hold for data present in database are called Constraints.
- Every relation in the database should have at least one set of attributes which defines a tuple uniquely. Those set of attributes is called key. e.g.; ROLL_NO in STUDENT is a key
- SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus
- The left outer join returns a result table with the matched data of two tables then remaining rows of the left table and null for the right table's column
- The right outer join returns a result table with the matched data of two tables then remaining rows of the right table and null for the left table's columns.
- INNER Join is a simple JOIN in which the result is based on matched data as per the equality condition specified in the query.
- All DDL commands are auto-committed. That means it saves all the changes permanently in the database
- DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back
- Transaction Control Language: These commands are to keep a check on other commands and their effect on the database. These commands can annul changes made by other commands by rolling back to original state. It can also make changes permanent.
- Data control language provides command to grant and take back authority

Glossary:

- DBA: A database administrator (DBA) directs or performs all activities related to maintaining a successful database environment. Responsibilities include designing, implementing, and maintaining the database system
- oracle: Oracle Corporation is an American multinational computer technology corporation, headquartered in Redwood Shores, California. The company specializes primarily in developing and marketing database software and technology, cloud engineered systems and enterprise software products
- **Primary key**: A primary key is a special relational database table column (or combination of columns) designated to uniquely identify all table records.
- My SQL: is an open-source relational database management system(RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter,] and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements
- **Constraint:** limitation or restriction.
- **Referential integrity (RI)**: Referential integrity (RI) is a relational database concept, which states that table relationships must always be consistent
- **Query:** A question, especially one expressing doubt or requesting information.

Exercise

Very Short Question

a. Define table in RDBMS
b. What is record?
c. What is field?
d. What is join in RDBMS
e. Define DCL in RDBMS.
f. Choose best answer

The operation is intended to join two relations on the basis of comparison operation is called

a. outer join b. theta join c. Natural joins c. Cartesian produce.

Modification of the database is done by the operator

a. Insertion b. deletion c. Update d. all of the above.

Short Question

a. List out the different types of joins.
b. What is the major drawback of natural join?
c. What is right outer join?
d. What is SQL? Where does it use?
e. List out different languages use in SQL

Long Question

a. What is RDBMS? Explain with examples.
b. What is constrain in relational Model? Illustrate with example.
c. Explain referential integrity with example.
d. Describe Code rules in detail.
e. Explain different types of joins with example.

References:

https://www.techopedia.com/definition/1245/structured-query-language-sql

https://www.tutorialspoint.com/dbms/dbms_data_models.htm

UNIT-4

Database Design

Objective

After completion of this unit students will be able

- To distinguish between Trivial and Non-trivial
- To demonstrate closure of a set of Attributes
- To practice the knowledge of Normalization

Learning Process and Study Materials

- Class demonstration and practical
- Group discussion
- Case study

Content's Elaboration:

Introduction: while designing a database out of an entity-relationship mode, the main problem existing in that raw database is redundancy. redundancy is storing the same data item in more one place. A redundancy creates several problems like the following:

- Extra storage space: strong the same data in many places takes large amount of disk pace.
- Entering same data more than once during data insertion.
- Deleting data from more than one place during deletion.
- Modifying data in more than one place.
- Anomalies may occur in the database if insertion, deletion, modification etc. are no done properly.
- It creates inconsistency and unreliability in the database.

To solve this problem the raw database needs to be normalized. Normalization is a step by step process of removing different kinds of redundancy and normally at each step. At each step a specific rule is followed to remove specific kind of impurity in order to give the database a slim and clean look. The process of reducing data

redundancy and removing database modification anomaly in a relational database is called normalization.

Functional Dependencies

The attributes of a table are said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table.

For example: Suppose we have a student table with attributes: Stu_Id, Stu_Name, Stu_Age. Here Stu_Id attribute uniquely identifies the Stu_Name attribute of student table because if we know the student id we can tell the student name associated with it. This is known as functional dependency and can be written as Stu_Id->Stu_Name or in words we can say Stu_Name is functionally dependent on Stu_Id.

Formally:

If column A of a table uniquely identifies the column B of same table then it can represented as A->B (Attribute B is functionally dependent on attribute A)

Types of Functional Dependencies

- Trivial functional dependency
- non-trivial functional dependency

Trivial functional dependency

The dependency of an attribute on a set of attributes is known as trivial functional dependency if the set of attributes includes that attribute.

Symbolically: A ->B is trivial functional dependency if B is a subset of A.

The following dependencies are also trivial: A->A & B->B

For example: Consider a table with two columns Student_id and Student_Name.

{Student_Id, Student_Name} ->Student_Id is a trivial functional dependency as Student_Id is a subset of {Student_Id, Student_Name}. That makes sense because if we know the values of Student_Id and Student_Name then the value of Student_Id can be uniquely determined.

Also, Student_Id ->Student_Id&Student_Name ->Student_Name are trivial dependencies too.

Non trivial functional dependency

If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called non trivial Functional dependency.

For example: An employee table with three attributes: emp_id, emp_name, emp_address. The following functional dependencies are non-trivial:emp_id ->emp_name (emp_name is not a subset of emp_id) emp_id ->emp_address (emp_address is not a subset of emp_id)

On the other hand, the following dependencies are trivial:{emp_id, emp_name} ->emp_name [emp_name is a subset of {emp_id, emp_name}]

Closure of a Set of Functional Dependencies

1. We need to consider *all* functional dependencies that hold. Given a set *F* of functional dependencies, we can prove that certain other ones also hold. We say these ones are **logically implied** by *F*.
2. Suppose we are given a relation scheme *R*=(*A*,*B*,*C*,*G*,*H*,*I*), and the set of functional dependencies:

   A $\rightarrow$ B

   A $\rightarrow$ C

   CG $\rightarrow$ H

   CG $\rightarrow$ I

   B $\rightarrow$ H

Then the functional dependency $A \rightarrow H$ is logically implied.

Closure of Attribute Sets

1. To test whether a set of attributes α is a superkey, we need to find the set of attributes functionally determined by α.
2. Let α be a set of attributes. We call the set of attributes determined by α under a set *F* of functional dependencies the **closure** of α under *F*, denoted α^+.
3. The following algorithm computes α^+:

result := α

while (changes to *result*) **do**

for each functional dependency $\beta \rightarrow \gamma$

in *F* **do**

begin

if $\beta \subseteq$ *result*

then *result* := *result* $\cup \gamma$;

end

1. If we use this algorithm on our example to calculate (AG^+) then we find:
   - We start with *result* = AG.
   - $A \rightarrow B$ causes us to include B in *result*.
   - $A \rightarrow C$ causes *result* to become ABCG.
   - $CG \rightarrow H$ causes *result* to become ABCGH.
   - $CG \rightarrow I$ causes *result* to become ABCGHI.
   - The next time we execute the while loop, no new attributes are added, and the algorithm terminates.
2. This algorithm has worst case behavior quadratic in the size of *F*. There is a linear algorithm that is more complicated.

Normalization

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

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

- **Update anomalies** − If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
- **Deletion anomalies** − We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
- **Insert anomalies** − We tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring the database to a consistent state.

First Normal Form

First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.

Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

We re-arrange the relation (table) as below, to convert it to First Normal Form.

Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

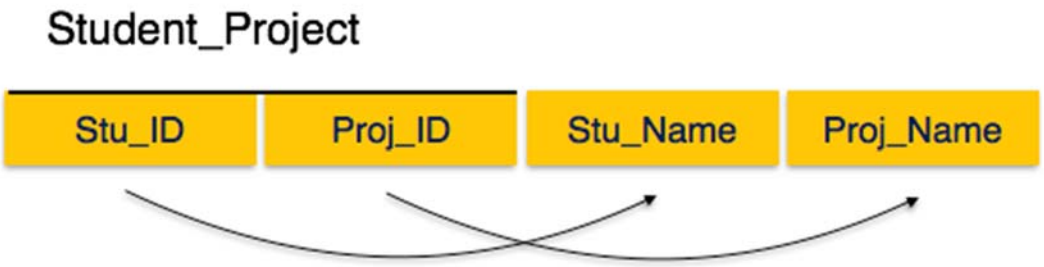
Each attribute must contain only a single value from its pre-defined domain.

Second Normal Form

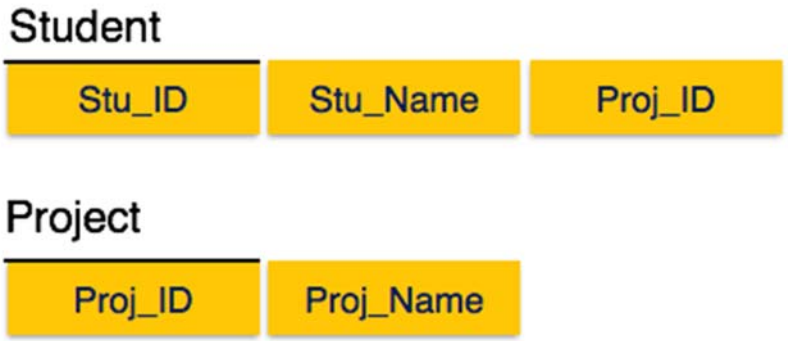
Before we learn about the second normal form, we need to understand the following−

- **Prime attribute** − An attribute, which is a part of the candidate-key, is known as a prime attribute.
- **Non-prime attribute** − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.



We see here in Student_Project relation that the prime key attributes are Stu_ID and Proj_ID. According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually. But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently. This is called **partial dependency**, which is not allowed in Second Normal Form.



We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

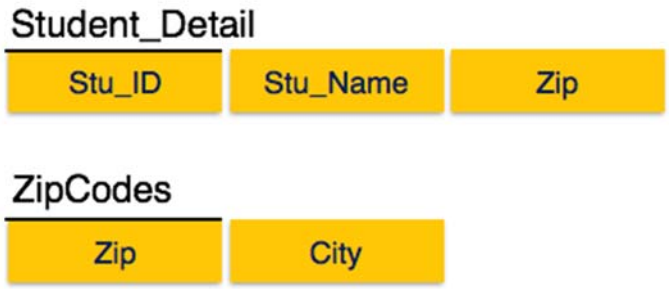
Third Normal Form

For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy −

- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency, X → A, then either −
  - X is a superkey or,
  - A is prime attribute.

We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute. We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a superkey nor is City a prime attribute. Additionally, Stu_ID → Zip → City, so there exists **transitive dependency**.

To bring this relation into third normal form, we break the relation into two relations as follows −



Key points:

- **Normalization** is a step by step process of removing different kinds of redundancy and normally at each step.
- **Functional Dependencies:** The attributes of a table is said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table.
- Trivial functional dependency: The dependency of an attribute on a set of attributes is known as trivial functional dependency if the set of attributes includes that attribute.
- Non trivial functional dependency: If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called non trivial Functional dependency.

Glossary:

- Trivial: of very little importance or value; insignificant
- Nontrivial:The valueof at least one variable of the equation is not equal to zero.
- Anomalies : A deviation from the common rule, type, arrangement, or form.
- Domain: A data domain refers to all the values which a data element may contain.
- Ttransitive dependency: A transitive dependency in a database is an indirect relationship between values in the same table that causes a functional dependency

Exercise

Very Short Question

a. What is the miniumum condition to be 2nd normal form
b. If all non-prime attributes are fully functional dependent on the relation key then what is the relation called ?
c. List out the types of functional dependencies.
d. What is prime attribute ?
e. Define normalization.

Short Question

a. What is functional Dependencies ? write its types
b. Differentiate between Trivial and Non trivial functional dependency
c. Why normalization is important in database ?
d. Define second normal form with its condition
e. Define third normalization with its minimum condition

Long Question

a. Discuss un-normalized form with example.
b. Explain types of functional dependency with examples.
c. What do you mean by closure of a set of functional dependencies ?
d. Compare fully and partially functional dependency.
e. Explain Normalization process with examples.

References:

https://beginnersbook.com/2015/04/functional-dependency-in-dbms/

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter6/node12.html

https://www.tutorialspoint.com/dbms/database_normalization.htm

UNIT 5

Concurrency Control and recovery

Objectives

After completion of this unit students will be able

- To identify the transaction
- To describe the properties of transaction
- To illustrate the knowledge of state of transaction
- To memorize the requirements for recovery

Learning Process and Study Material:

- Class demonstration and Case Study.
- Practical woks can be done in recovery part.

Content's Elaboration:

- Introduction Of transaction

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Let's take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

A's Account

```
Open_Account(A)
Old_Balance = A.balance
New_Balance = Old_Balance - 500
A.balance = New_Balance
Close_Account(A)
```

B's Account

```
Open_Account(B)
```

```
Old_Balance = B.balance
New_Balance = Old_Balance + 500
B.balance = New_Balance
Close_Account(B)
```

Write and Read-Write Operations

You can manage the specific behavior of concurrent write operations by deciding when and how to run different types of commands. The following commands are relevant to this discussion:

- COPY commands, which perform loads (initial or incremental)
- INSERT commands that append one or more rows at a time
- UPDATE commands, which modify existing rows
- DELETE commands, which remove rows

COPY and INSERT operations are pure write operations, but DELETE and UPDATE operations are read-write operations. (In order for rows to be deleted or updated, they have to be read first.) The results of concurrent write operations depend on the specific commands that are being run concurrently. COPY and INSERT operations against the same table are held in a wait state until the lock is released, then they proceed as normal.

UPDATE and DELETE operations behave differently because they rely on an initial table read before they do any writes. Given that concurrent transactions are invisible to each other, both UPDATEs and DELETEs have to read a snapshot of the data from the last commit. When the first UPDATE or DELETE releases its lock, the second UPDATE or DELETE needs to determine whether the data that it is going to work with is potentially stale. It will not be stale, because the second transaction does not obtain its snapshot of data until after the first transaction has released its lock.

Potential Deadlock Situation for Concurrent Write Transactions

Whenever transactions involve updates of more than one table, there is always the possibility of concurrently running transactions becoming deadlocked when they both try to write to the same set of tables. A transaction releases all of its table locks at once when it either commits or rolls back; it does not relinquish locks one at a time.

For example, suppose that transactions T1 and T2 start at roughly the same time. If T1 starts writing to table A and T2 starts writing to table B, both transactions can proceed without conflict; however, if T1 finishes writing to table A and needs to start writing to table B, it will not be able to proceed because T2 still holds the lock on B. Conversely, if T2 finishes writing to table B and needs to start writing to table A, it will not be able to proceed either because T1 still holds the lock on A. Because neither transaction can release its locks until all its write operations are committed, neither transaction can proceed.

In order to avoid this kind of deadlock, you need to schedule concurrent write operations carefully. For example, you should always update tables in the same order in transactions and, if specifying locks, lock tables in the same order before you perform any DML operations.

Properties of transaction

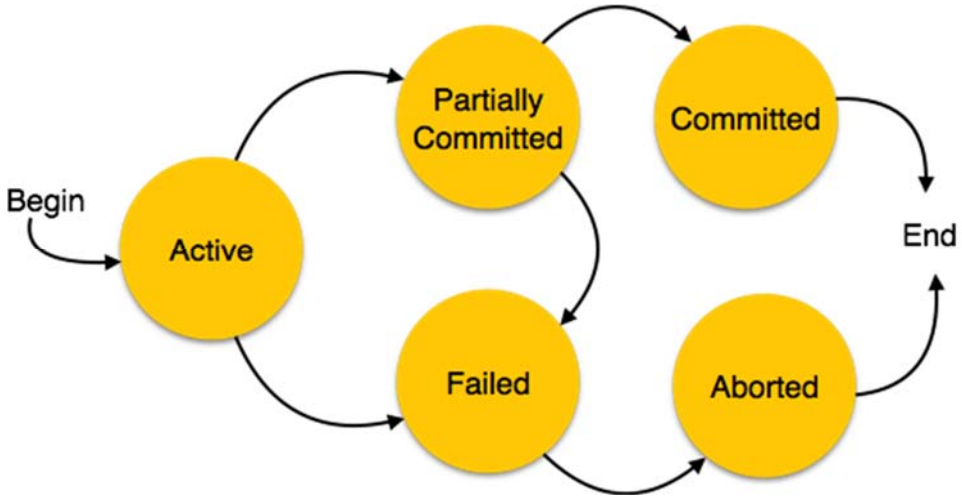
A transaction is a very small unit of a program and it may contain several low level tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

- **Atomicity** − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
- **Consistency** − The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
- **Durability** − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

- **Isolation** − In a database system where more than one transaction being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

States of Transactions

A transaction in a database can be in one of the following states −



- **Active** − In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** − When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The

database recovery module can select one of the two operations after a transaction aborts −

- Re-start the transaction
- Kill the transaction

Committed − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

Data Recovery

DBMS is a highly complex system with hundreds of transactions being executed every second. The durability and robustness of a DBMS depends on its complex architecture and its underlying hardware and system software. If it fails or crashes amid transactions, it is expected that the system would follow some sort of algorithm or techniques to recover lost data.

Requirements for recovery

When a system crashes, it may have several transactions being executed and various files opened for them to modify the data items. Transactions are made of various operations, which are atomic in nature. But according to ACID properties of DBMS, atomicity of transactions as a whole must be maintained, that is, either all the operations are executed or none.

When a DBMS recovers from a crash, it should maintain the following −

- It should check the states of all the transactions, which were being executed.
- A transaction may be in the middle of some operation; the DBMS must ensure the atomicity of the transaction in this case.
- It should check whether the transaction can be completed now or it needs to be rolled back.
- No transactions would be allowed to leave the DBMS in an inconsistent state.

There are two types of techniques, which can help a DBMS in recovering as well as maintaining the atomicity of a transaction −

- Maintaining the logs of each transaction, and writing them onto some stable storage before actually modifying the database.
- Maintaining shadow paging, where the changes are done on a volatile memory, and later, the actual database is updated.

Recovery with Concurrent Transactions

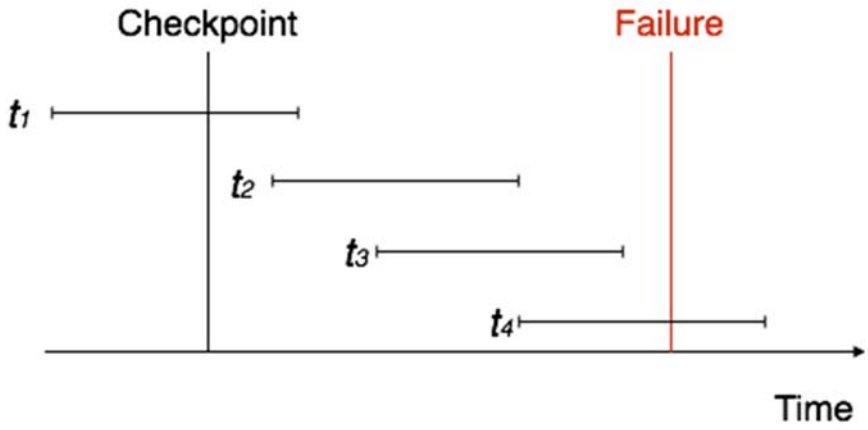
When more than one transaction are being executed in parallel, the logs are interleaved. At the time of recovery, it would become hard for the recovery system to backtrack all logs, and then start recovering. To ease this situation, most modern DBMS use the concept of 'checkpoints'.

Checkpoint

Keeping and maintaining logs in real time and in real environment may fill out all the memory space available in the system. As time passes, the log file may grow too big to be handled at all. Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.

Recovery

When a system with concurrent transactions crashes and recovers, it behaves in the following manner −



- The recovery system reads the logs backwards from the end to the last checkpoint.
- It maintains two lists, an undo-list and a redo-list.
- If the recovery system sees a log with <T_n, Start> and <T_n, Commit> or just <T_n, Commit>, it puts the transaction in the redo-list.
- If the recovery system sees a log with <T_n, Start> but no commit or abort log found, it puts the transaction in undo-list.

All the transactions in the undo-list are then undone and their logs are removed. All the transactions in the redo-list and their previous logs are removed and then redone before saving their logs.

Key Points

- A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.
- One can manage the specific behavior of concurrent write operations by deciding when and how to run different types of commands
- The following commands are relevant to this discussion:
- COPY commands, which perform loads (initial or incremental)
- INSERT commands that append one or more rows at a time
- UPDATE commands, which modify existing rows
- DELETE commands, which remove rows
- Whenever transactions involve updates of more than one table, there is always the possibility of concurrently running transactions becoming deadlocked when they both try to write to the same set of tables
- Properties of transaction
    - Atomicity
    - Consistency
    - Durability
    - Isolation
- States of Transactions
    - Active
    - Partially Committed
    - Failed
    - Aborted
    - Committed
- In computing, data recovery is a process of salvaging (retrieving) inaccessible, lost, corrupted, damaged or formatted data from secondary storage, removable media or files, when the data stored in them cannot be accessed in a normal way.

Glossary:

- **Deadlock:** a deadlock is a state in which each member of a group is waiting for some other member to take action, such as sending a message or more commonly releasing a lock.
- **Algorithm:** It is a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
- **Atomicity :** is part of the ACID model (Atomicity, Consistency, Isolation, Durability), which is a set of principles used to guarantee the reliability of database transactions

Exercises:

Very Short Question

a. Define transaction in Database System
b. What are the commands use in write and read operation
c. What is mean by deadlock ?
d. List out the porperties of transaction
e. What are the states of transactions

Short Question

a. What is potential deadlock situation ?
b. Write Atomicity properties of transaction.
c. Explain about states of transactions.
d. What is checkpoint in DBMS?
e. What is data recovery ?

Long Question

a. Explain the write and read -write operation in detail.
b. write briefly about the properties of transaction.
c. Illustrate with figure about the states of transactions.
d. List out he requirements for recovery in detail.
e. Explain data recovery with figure.

References:

https://www.tutorialspoint.com/dbms/dbms_data_recovery.htm

https://www.techopedia.com/definition/1245/structured-query-language-sql

https://docs.aws.amazon.com/redshift/latest/dg/c_write_readwrite.html

UNIT-6
Security

Objectives
After completion of this unit students will be able

- To state the security of DBMS
- To recognize Access Control
- To tell statistical Database
- To discuss with data encryption technique.

Learning Process and Study Materials
- Class demonstration and Questionnaire
- Group discussion and practical work

Content's Elaboration:

Database Security and Threats

Data security is an imperative aspect of any database system. It is of particular importance in distributed systems because of large number of users, fragmented and replicated data, multiple sites and distributed control.

Threats in a Database

- **Availability loss** − Availability loss refers to non-availability of database objects by legitimate users.
- **Integrity loss** − Integrity loss occurs when unacceptable operations are performed upon the database either accidentally or maliciously. This may happen while creating, inserting, updating or deleting data. It results in corrupted data leading to incorrect decisions.
- **Confidentiality loss** − Confidentiality loss occurs due to unauthorized or unintentional disclosure of confidential information. It may result in illegal actions, security threats and loss in public confidence.

Measures of Control

The measures of control can be broadly divided into the following categories −

- **Access Control** − Access control includes security mechanisms in a database management system to protect against unauthorized access. A user can gain access to the database after clearing the login process through only valid user accounts. Each user account is password protected.
- **Flow Control** − Distributed systems encompass a lot of data flow from one site to another and also within a site. Flow control prevents data from being transferred in such a way that it can be accessed by unauthorized agents. A flow policy lists out the channels through which information can flow. It also defines security classes for data as well as transactions.
- **Data Encryption** − Data encryption refers to coding data when sensitive data is to be communicated over public channels. Even if an unauthorized agent gains access of the data, he cannot understand it since it is in an incomprehensible format.

Role of DBA in Database Security

Keeping data secure is only one of the roles of the DBA. The typical tasks of a DBA include:

- Controlling access to the database, including creating logins for users and setting roles for each user. Some users may only need to query the data, while others are involved in entering new data.
- Providing support services to the end users, such as making sure all users know how to use the database.
- Managing procedures for backup and recovery of data, in case of errors made by users or system crashes. You don't want to lose all the valuable data just because the power went out.
- Ensuring data integrity, which means that data are complete, accurate and current for the tasks at hand.
- Controlling data security, including preventing unauthorized access to the data and protecting against other security threats.
- Setting data privacy, which means that only authorized individuals are able to see certain data. For example, there is no need for everyone in the organization to be able to see all the personnel files of all the employees.

What is Threat in Database System ?

A threat is something that may or may not happen, but has the potential to cause serious damage. Threats can lead to attacks on computer systems, networks and more.

Threats are potentials for vulnerabilities to turn into attacks on computer systems, networks, and more. They can put individuals' computer systems and business computers at risk, so vulnerabilities have to be fixed so that attackers cannot infiltrate the system and cause damage.

Threats can include everything from viruses, Trojans, back doors to outright attacks from hackers. Often, the term blended threat is more accurate, as the majority of threats involve multiple exploits. For example, a hacker might use a phishing attack to gain information about a network and break into a network.

Threats to Database security

Almost all organizations use databases in some form for tracking information such as customer and transaction records, financial information, and human resources records. Much of the information contained in databases is sensitive and can be sold for cash or, such as in cases of theft by a disgruntled employee or by a hacker with political motivations, to cause the organization loss of business or reputation, especially if the organization is found to be in breach of regulations or industry standards that demand high levels of data security.

According to technology vendor Application Security, Inc., the following are the top 10 threats related to databases:

1. Default or weak passwords
2. SQL injection
3. Excessive user and group privileges
4. Unnecessary DBMS features enabled
5. Broken configuration management
6. Buffer overflows
7. Privilege escalation
8. Denial of service
9. Un-patched RDBMS
10. Unencrypted data

Database Security Essentials

There are 5 key steps to ensuring database security, according to Applications Security, Inc.

1. Isolate sensitive databases—maintain an accurate inventory of all databases deployed across the enterprise and identify all sensitive data residing on those databases.
2. Eliminate vulnerabilities—continually assess, identify and remediate vulnerabilities that expose the database.
3. Enforce least privileges—identify user entitlements and enforce user access controls and privileges to limit access to only the minimum data required for employees to do their jobs.
4. Monitor for deviations—implement appropriate policies and monitor any vulnerabilities that cannot be remediated for any and all activity the deviates from authorized activity.
5. Respond to suspicious behavior—alert and respond to any abnormal or suspicious behavior in real time to minimize risk of attack.

Classification of Database Security

The database security can be broadly classified into physical and logical security. Database recovery refers to the process of restoring database to a correct state in the event of a failure.

- **Physical Security**: Physical security refers to the security of the hardware associated with the system and the protection of the site where the computer resides. Natural events such a fire, foods, and earthquakes car be considered as some of the physical threats. It is advisable to have backup copies of databases in the face of massive disasters.
- **Logical security**: Logical security consists of software safeguards for a organization's systems including user identification and password access, authenticating, access rights and authority levels. these measures are to ensure that only authorized users are able to perform actions or access information in a network or a workstation. It is a subset of computer security.

Discretionary Access Control (DAC)

Discretionary access control (DAC) is a type of security access control that grants or restricts object access via an access policy determined by an object's owner group and/or subjects. DAC mechanism controls are defined by user identification with supplied credentials during authentication, such as username and password. DACs are discretionary because the subject (owner) can transfer authenticated objects or information access to other users. In other words, the owner determines object access privileges.

In DAC, each system object (file or data object) has an owner, and each initial object owner is the subject that causes its creation. Thus, an object's access policy is determined by its owner.

A typical example of DAC is Unix file mode, which defines the read, write and execute permissions in each of the three bits for each user, group and others. DAC attributes include:

- User may transfer object ownership to another user(s).
- User may determine the access type of other users.
- After several attempts, authorization failures restrict user access.
- Unauthorized users are blind to object characteristics, such as file size, file name and directory path.
- Object access is determined during access control list (ACL) authorization and based on user identification and/or group membership.

DAC is easy to implement and intuitive but has certain

Disadvantages, including:

- Inherent vulnerabilities (Trojan horse)
- ACL maintenance or capability
- Grant and revoke permissions maintenance
- Limited negative authorization power

Mandatory Access Control (MAC)

Mandatory Access Control (MAC) Is a set of security policies constrained according to system classification, configuration and authentication. MAC policy management and settings are established in one secure network and limited to system administrators.

MAC defines and ensures a centralized enforcement of confidential security policy parameters.

or best practices, MAC policy decisions are based on network configuration. In contrast, certain operating systems (OS) enable limited Discretionary Access Control (DAC).

MAC advantages and disadvantages depend on organizational requirements, as follows:

- MAC provides tighter security because only a system administrator may access or alter controls.
- MAC policies reduce security errors.
- MAC enforced operating systems (OS) delineate and label incoming application data, which creates a specialized external application access control policy.

Statistical Database

It is a database used for statistical analysis purposes. It is an OLAP (online analytical processing), instead of OLTP (online transaction processing) system. Modern decision, and classical statistical databases are often closer to the relational model than the multidimensional model commonly used in OLAP systems today.

Statistical databases typically contain parameter data and the measured data for these parameters. For example, parameter data consists of the different values for varying conditions in an experiment (e.g., temperature, time). The measured data (or variables) are the measurements taken in the experiment under these varying conditions.

Many statistical databases are sparse with many null or zero values. It is not uncommon for a statistical database to be 40% to 50% sparse. There are two options for dealing with the sparseness:

(1) leave the null values in there and use compression techniques to squeeze them out or

(2) remove the entries that only have null values.

Data Encryption

Encryption is a security method in which information is encoded in such a way that only authorized user can read it. It uses encryption algorithm to generate cipher text that can only be read if decrypted.

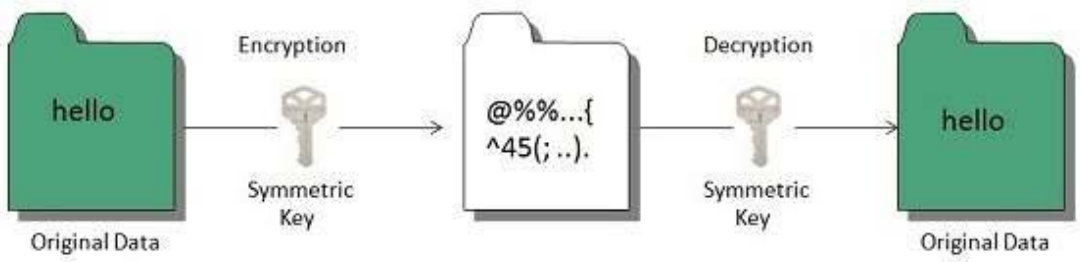
Types of Encryption

There are two types of encryptions schemes as listed below:

- Symmetric Key encryption
- Public Key encryption

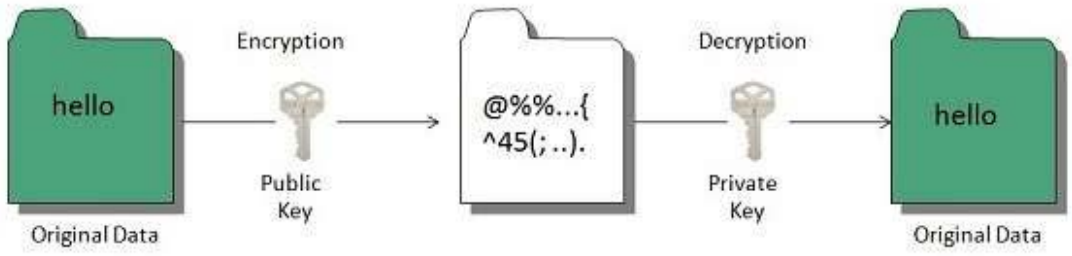
Symmetric key encryption

Symmetric key encryption algorithm uses same cryptographic keys for both encryption and decryption of cipher text.



Public Key Encryption

Public key encryption algorithm uses pair of keys, one of which is a secret key and one of which is public. These two keys are mathematically linked with each other.

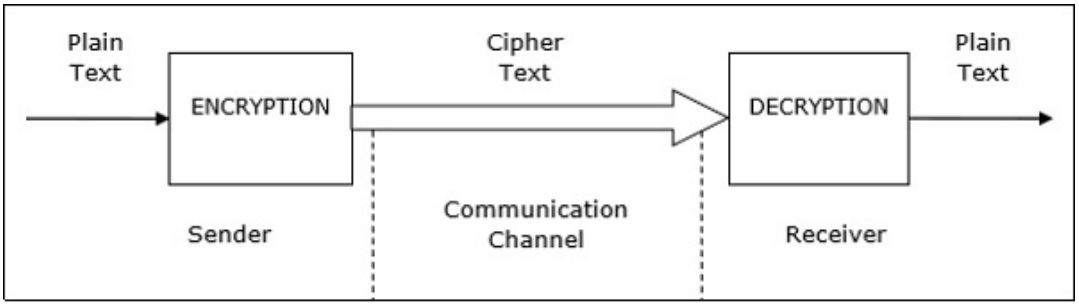


Cryptography

Cryptography is the science of encoding information before sending via unreliable communication paths so that only an authorized receiver can decode and use it.

The coded message is called **cipher text** and the original message is called **plain text**. The process of converting plain text to cipher text by the sender is called encoding or **encryption**. The process of converting cipher text to plain text by the receiver is called decoding or **decryption**.

The entire procedure of communicating using cryptography can be illustrated through the following diagram −



Private Key Cryptography

In conventional cryptography, the encryption and decryption is done using the same secret key. Here, the sender encrypts the message with an encryption algorithm using a copy of the secret key. The encrypted message is then send over public communication channels. On receiving the encrypted message, the receiver decrypts it with a corresponding decryption algorithm using the same secret key.

Security in conventional cryptography depends on two factors −

- A sound algorithm which is known to all.
- A randomly generated, preferably long secret key known only by the sender and the receiver.

The most famous conventional cryptography algorithm is **Data Encryption Standard** or **DES**.

The advantage of this method is its easy applicability. However, the greatest problem of conventional cryptography is sharing the secret key between the communicating parties. The ways to send the key are cumbersome and highly susceptible to eavesdropping.

Public Key Cryptography

In contrast to conventional cryptography, public key cryptography uses two different keys, referred to as public key and the private key. Each user generates the pair of public key and private key. The user then puts the public key in an accessible place. When a sender wants to sends a message, he encrypts it using the public key of the receiver. On receiving the encrypted message, the receiver decrypts it using his private key. Since the private key is not known to anyone but the receiver, no other person who receives the message can decrypt it.

The most popular public key cryptography algorithms are **RSA** algorithm and **Diffie– Hellman** algorithm. This method is very secure to send private messages. However, the problem is, it involves a lot of computations and so proves to be inefficient for long messages.

The solution is to use a combination of conventional and public key cryptography. The secret key is encrypted using public key cryptography before sharing between the communicating parties. Then, the message is send using conventional cryptography with the aid of the shared secret key.

Digital Signatures

A Digital Signature (DS) is an authentication technique based on public key cryptography used in e-commerce applications. It associates a unique mark to an individual within the body of his message. This helps others to authenticate valid senders of messages.

Typically, a user's digital signature varies from message to message in order to provide security against counterfeiting. The method is as follows −

- The sender takes a message, calculates the message digest of the message and signs it digest with a private key.
- The sender then appends the signed digest along with the plaintext message.
- The message is sent over communication channel.
- The receiver removes the appended signed digest and verifies the digest using the corresponding public key.
- The receiver then takes the plaintext message and runs it through the same message digest algorithm.
- If the results of step 4 and step 5 match, then the receiver knows that the message has integrity and authentic.

Hashing

In terms of security, hashing is a technique used to encrypt data and generate unpredictable hash values. It is the hash function that generates the hash code, which helps to protect the security of transmission from unauthorized users.

Hash function algorithms

Hashing algorithm provides a way to verify that the message received is the same as the message sent. It can take a plain text message as input and then computes a value based on that message.

Essential Points

- The length of computed value is much shorter than the original message.
- It is possible that different plain text messages could generate the same value.

Here we will discuss a sample hashing algorithm in which we will multiply the number of a's, e's and h's in the message and will then add the number of o's to this value.

For example, the message is " the combination to the safe is two, seven, thirty-five". The hash of this message, using our simple hashing algorithm is as follows:

2 x 6 x 3) + 4 = 40

The hash of this message is sent to John with cipher text. After he decrypts the message, he computes its hash value using the agreed upon hashing algorithm. If the hash value sent by Bob doesn't match the hash value of decrypted message, John will know that the message has been altered.

For example, John received a hash value of 17 and decrypted a message Bob has sent as "You are being followed, use backroads, hurry"

He could conclude the message had been altered, this is because the hash value of the message he received is:

(3x4x1)+4 = 16

This is different from then value 17 that Bob sent.

Key points:

- Data security is an imperative aspect of any database system. It is of particular importance in distributed systems because of large number of users, fragmented and replicated data, multiple sites and distributed control.
- The measures of control can be broadly divided into the following categories −
  - Access Control
  - Flow Control
  - Data Encryption
  - A threat is something that may or may not happen, but has the potential to cause serious damage. Threats can lead to attacks on computer systems, networks and more.
  - Classification of Database Security
    - Physical Security
    - Logical security
- **Discretionary access control** (DAC) is a type of security access control that grants or restricts object access via an access policy determined by an object's owner group and/or subjects
- **Mandatory Access Control** (MAC) is a set of security policies constrained according to system classification, configuration and authentication
- **Statistical** is a database used for statistical analysis purposes. It is an OLAP (online analytical processing), instead of OLTP (online transaction processing) system. Modern decision, and classical statistical databases are often closer to the relational model than the multidimensional model commonly used in OLAP systems today.
- **Encryption** is a security method in which information is encoded in such a way that only authorized user can read it
- Symmetric key encryption algorithm uses same cryptographic keys for both encryption and decryption of cipher text.
- Public key encryption algorithm uses pair of keys, one of which is a secret key and one of which is public. These two keys are mathematically linked with each other.
- **A Digital Signature (**DS) is an authentication technique based on public key cryptography used in e-commerce applications.
- Hashing is a technique used to encrypt data and generate unpredictable hash values.

Glossary:

- **A Digital Signature (**DS) is an authentication technique based on public key cryptography used in e-commerce applications.
- Hashing is a technique used to encrypt data and generate unpredictable hash values.
- **Fragmented:** Break or cause to break into fragments.
- **Confidentiality:** The state of keeping or being kept secret or private.
- **Encryption:** Confidentiality is a set of rules or a promise that limits access or places restrictions on certain types of information.
- **logins:** an act of logging in to a computer, database, or system.A password or code used when logging in.
- **unauthorized:** Not having official permission or approval
- **Trojans Horse:** A **trojan horse** (Sometimes called a **trojan**) is a kind of malware. It is special computer program that pretends to do a certain thing, but in reality it does something else, such as allow a stranger to access the computer and change it and read its information. In some cases the user notices, in other cases they do not. Spyware programs are current examples of programs that work as trojans.
- **Back doors:** A backdoor is a means to access a computer system or encrypted data that bypasses the system's customary security mechanisms.
- **Hackers:** A **hacker** is any skilled computer expert that uses their technical knowledge to overcome a problem. While "**hacker**" can refer to any skilled computer programmer, the term has become associated in popular culture with a "security **hacker**", someone who, with their technical knowledge,
- **Vulnerabilities:** A **vulnerability** is a weakness which allows an attacker to reduce a system's information assurance. **Vulnerabilities** are the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw.
- **privileges:** When multiple users can access **database** objects, authorization can be controlled to these objects with **privileges**. Every object has an owner. **Privileges** control if a user can modify an object owned by another user
- **grants: GRANT** statement is use to give privileges to a specific user or role, or to all users, to perform actions on **database** objects.

- revoke:REVOKE statement is use to remove privileges from a specific user or role, or from all users, to perform actions on **database** objects.
- **cipher:** In cryptography, **ciphertext** or cyphertext is the result of encryption performed on plaintext using an algorithm, called a cipher.

Exercises:

Very Short Question Answer

a. Define Database security.
b. What is thereat ?
c. What is encryption mean in database security ?
d. Define cipher text.
e. Who is responsible for overall security of the database system.

Short Question Answer

a. List out the different threats in Database System.
b. What are the measures of control use in Database System?
c. List out the role of DBA in Database Security.
d. Write the threats found in Database.
e. Classify the Database Security.

Long Question Answer

a) Explain the role of DBA in Database Security in detail.
b) Describe the method of Database Security Essentials.
c) What is Discretionary Access Control? Write its attributes.
d) Differentiate between Private Key cryptography and Public Key cryptography.
e) What is digital signature? whey it is important ?

References:

https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_database_security_cryptography.htm

http://www.dbta.com/Editorial/Think-About-It/5-Key-Steps-to-Ensuring-Database-Security-95307.aspx