DBMS Copy Notes

ICT 4th Semester

by

Mukesh Singh
ICT 5th batch, 2073
Sukuna Multiple Campus
Sundarharaincha-12, Morang

Unit-1

Database System Introduction

1.1. Basic Terminologies: Data vs information, Data Hierarchy, Database, Database Management System, Database System, Relational Database Management Systems:

Data vs Information:

Data

Data is raw facts or figures which are composed of alphabets, digits, and other symbols. It may or may not give any sense.

For example: 1, Priya, 5.4, 2, Anjali etc.

Information:

The output of data processing is called information. It is an organized collection of data which gives a complete sense.

for example:

SN	Name	Height (feet, inch)
1.	Priya	5.4'
2.	Anjali	5.3'
3.	Sandhya	5.5'

Data hierarchy:

Data hierarchy refers to the systematic organization of data, often in a hierarchical form. Data organization involves characters, fields, records, files & so on.

Database:

Database is collection of interrelated data of entities or objects which is stored in a computer in such a way that it can be easily accessed by user. It is stored in tabular form. It provides a base or foundation for managing a large volume of data in well organized manner.

For example- Phone diary, Result sheet, Customer record, Price lists etc.

Database Management System (DBMS)

A DBMS is computerized records keeping system. It is software that defines, manipulates and manages the database. The DBMS provides users and programmers with a systematic way to create, retrieve, update and ~~manipulate~~ manage data.

For example- FoxPro, DBase, Sybase, MS Access, MySQL, MS SQL Server, Oracle, DB2 etc.

Relational Database Management System (RDBMS):

The database system which stores and displays data in tabular format of row and columns like spreadsheet, is known as RDBMS.

For example- Oracle, SQL, MS Access, MySQL etc.

Database System / Model:

There are different DBMS, each characterized by the way where data are defined and structured, called database model.

It is of four types:

- Hierarchical Database Model
- Network Database Model
- Relational Database Model
- Entity Relational Database Model

1.2 Data Management Approaches: File Management Systems, Database Management Systems, Limitations, Advantages and Applications

1. File Management System:

A file management system is a type of software that manages data files in a computer system. It has limited capabilities and is designed to manage individual or group files, such as special office documents and records.

Advantages:

i) Backup
ii) Compactness → Possible to store data compactly.
iii) Data retrieval
iv) Editing
v) Remote Access
vi) Sharing

Limitations / disadvantages:
i) Data redundancy
ii) Data inconsistency
iii) Difficulty in accessing data
iv) Limited data sharing
v) Security problems

2. Database Management System
A DBMS is computerized record keeping system.
Advantages:
i) Sharing data
ii) Data backup and recovery
iii) Data integrity → (data accuracy, consistency)
iv) Multiple user interface
v) Reduced data redundancy (repitition of data)

Disadvantages / limitations:
i) Expensive
ii) Changing technology
iii) Need technical training
iv) Back up is needed.

Applications
i) Railway reservation system
ii) Library management system
iii) Banking
iv) Universities and Colleges
v) Credit card transactions
vi) Social media sites
vii) Military
viii) Online shopping
ix) Manufacturing
x) Telecommunications

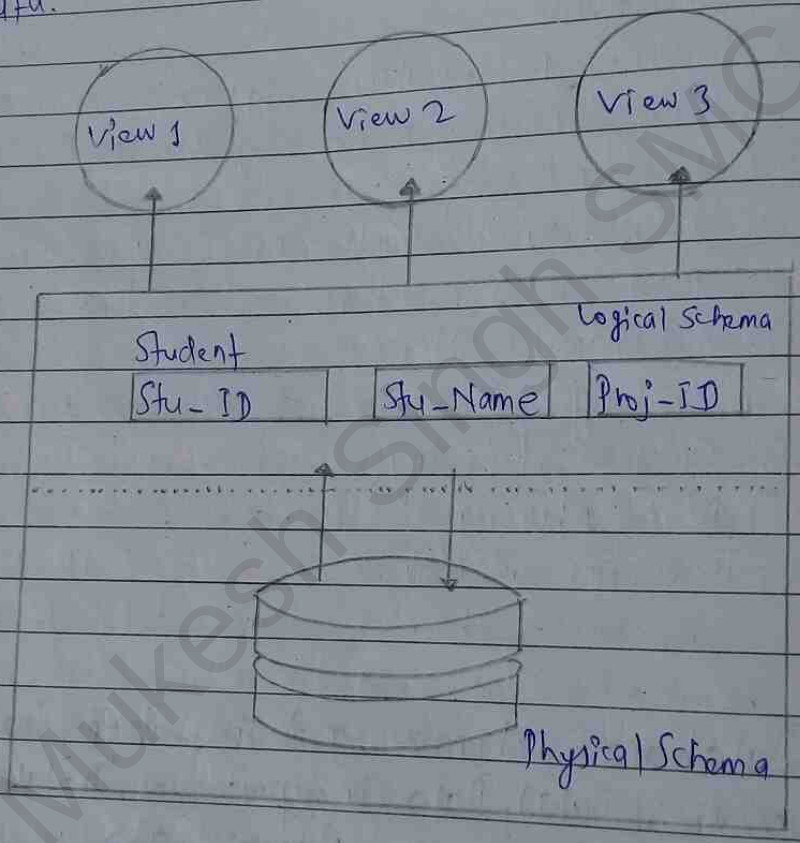
1.3. Database schema and instance, Data Abstraction (Views of Data), Data independence, Database Languages, Database Users and Administrator

Database schema and instance:

Database Schema

A database schema is the skeleton structure that represents the logical view of the

entire database. It defines how the data is organized and how the relations among them are associated. A database schema does not contain any data or information. It forms the constraints that are to be applied on the data.



A database schema can be broadly divided into two categories:

1. Physical database schema: The design of database at physical level is called physical schema. It defines how the data will be stored in a secondary storage.

2. Logical database schema:
The design of database at logical level is called logical schema. It defines tables, views, and integrity constraints.

(Optional) View Schema – Design of database at view level is called view schema. This generally describes end user interaction with database systems.

Database Instance
The data stored in database at particular moment of time is called instance of database. It contains a snapshot of database. Database schema defines the variable declaration in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

Data Abstraction (views of data):
The process of hiding irrelevant details from user is called data abstraction.

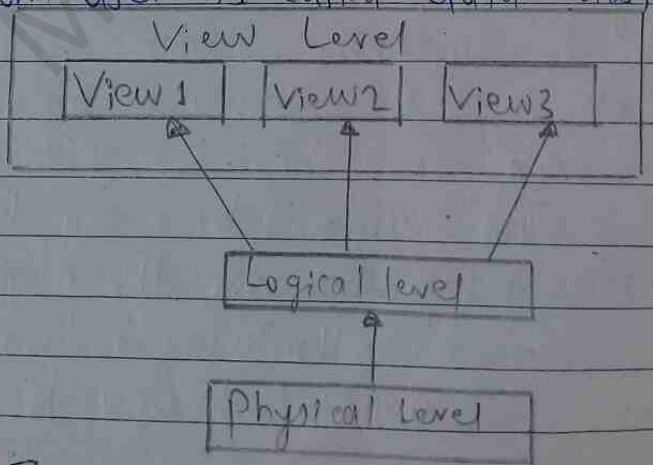


Fig: Three Levels of database abstraction

Sigma
schema

There are three levels of data abstraction.

1) Physical level:
This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get complex data structure at this level.

2. Logical level:-
This is the middle level of 3-level data abstraction. It describes what data is stored in database.

3. View Level:
It is the highest level of data abstraction. This level describes the user interaction with database system.

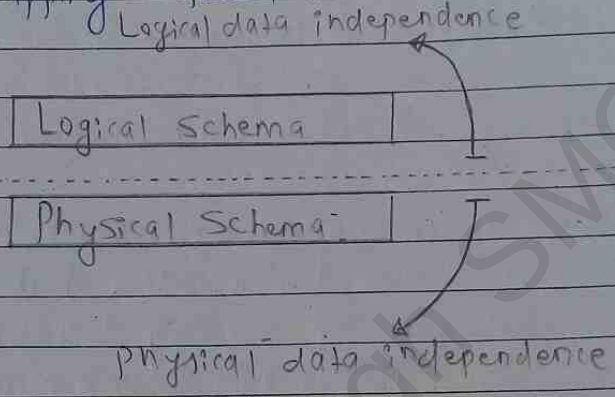
Data Independence:
Data independence is the type of data transparency that matters for a centralized DBMS. It refers to the immunity of user applications to changes made in the definition and organization of data.

Or

~~Two Types~~ Data independence can be defined ~~1. Logical data independence~~ as the capacity to change the schema ~~of the~~ at one level of a database system without having to change the

schema at the next higher level.

Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between two levels is changed.



1. Logical data independence:

It is the capacity to change the conceptual schema without having to change external schema or application program.

We may change the conceptual schema to expand the database, to change constraints, or to reduce the database.

2. Physical data independence:

It is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schema need not to be changed as well.

Changes for the internal schema may be needed because some physical files had to be reorganized.

Database Languages:

A generic term referring to a class of languages used for defining and accessing databases. Database languages are used to read, update and store data in a database. There are several such languages that can be used for this purpose; one of them is SQL.

Types of database languages:

1) Data Definition Language (DDL)

Database language that is used to create, delete or modify database schema is called DDL. All DDL commands are auto-committed. That means it saves all changes permanently in the database.

Examples of DDL are:

- CREATE: to create database
- ALTER: alters the structure of the existing database
- DROP: delete objects from the database
- TRUNCATE: remove all records from a table including all spaces allocated for the records are removed.

2 Data Manipulation Language (DML)

A DML is a language that enables users to access or manipulate data as organized by the appropriate data model.

DML commands are not auto-committed. It means changes are not permanent to database,

they can be rolled back.

Examples of DML are:

- SELECT: retrieve data from database
- INSERT: insert data into a table
- UPDATE: updates existing data in a table
- DELETE: delete all records from a database table

There are two types of DML:

1) Procedural DML: require a user to specify what data are needed and how to get those data.

2) Non-Procedural DML: require a user to specify what data are needed without specifying how to get those data.

3. Data Control Language (DCL)

DCL is used for granting and revoking user access on a database.

- GRANT - To grant access to user
- REVOKE - To revoke access from user

4. Transaction Control Language (TCL)

Manage transactions in the database using TCL.

Examply

- COMMIT - to permanently save
- ROLLBACK - to undo change
- SAVEPOINT - to save temporarily

Database users and Administrators

* Database Users

Database users are the one who really use and take the benifits of database. There are different types of users depending on their need and way of accessing the database.

1. Application Programmers:

Application programmers are computer professionals who write application programs. Application programmer can choose from many tools to develop user interfaces. Rapid Application Development (RAD) tools are tools that enable an application programmer to construct forms and reports with minimal programming effort.

2. Sophisticated users:

Sophisticated users interact with the system without writing programs. Instead, they form their requests either by using a database query language or by using tools such as data analysis software. Analysts who submit queries to explore data in the database fall in this category.

3. Specialized Users:

Specialized users are sophisticated users

who write specialized database applications that do not fit into the traditional data-processing framework.

Among these applications are computer aided design systems, knowledgebase and expert systems, systems that store data with complex data types. Scientists, researchers fall in this category.

4. Native users (or Naive Users)

Native users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

for example: A customer who uses ATM simply invokes the program which checks his/her user name, password, balance and finally allows to withdraw certain amount from his account

* Database Administrators:

The person who has central control over a database system is called Database Administrator (DBA).

The database administrator has the following functions in a database system.

- Schema definition:

The DBA creates the original database schema of the database by using DDL (data definition language).

- Schema and Physical Organization modification:
The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.

- Granting of authorization for data access:
The DBA can decide which parts of the database can be accessed by a user, by using the different types of authorization methods.

- Database maintenance (Routine maintenance)
The database maintenance includes the following process:
→ Regular backing up of the database
→ Ensuring the disk space for performing the required operations
→ Monitoring the jobs running on the database

Types of DBA:
a) Administrative DBA
b) Development DBA
c) Database Architect
d) Data Warehouse DBA
e) Application DBA
f) OLAP DBA

1.4. Data Models: Hierarchial, Network, Entity Relationship, Relational and Object Oriented data model.

A data model is a logical structure of database. It describes the design of database to reflect entities, attributes, relationship among data, constraints, etc.

Types of data models:

1. Hierarchial Database model:

Hierarchial data model is the first and one of the oldest type of database models. It represents the data in a hierarchial tree structure. It uses pointer to navigate between the stored data.

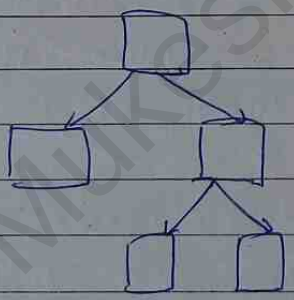


Fig: Hierarchial model

Advantages

- Easiest model
- Has one or more attributes
- Searching is fast & easy

Disadvantages

- Old fashioned/outdated model
- Increases redundancy

2. Network data model:

Network Database Model is same like Hierarchial model, but the only difference is that it allows a record to have more than one

parent. In this model, there is no need of parent to child association like the hierarchical model. It replaces the hierarchial tree with a graph. This model is easy to design & understand.

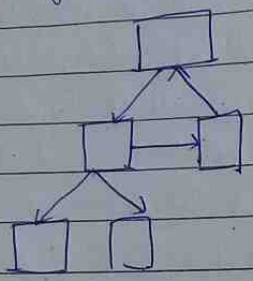


Fig: Network model

Advantages
- More flexible
- Searching is faster
- Simple & easy to design

Disadvantages
- less security
- Difficult to handle

3. Relational Model

The database ~~model~~ system which stores and displays data in tabular format of rows and columns, like spreadsheet, is known as Relational model. It is also known as RDBMS.

Examples - Oracle, SQL, MS-Access, Mysql etc.

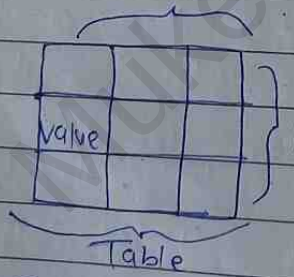


Fig: Relational Model

Advantages
- Processing is faster
- Less redundancy

Disadvantages
- More complex than others
- Needs more powerful computers

4. Entity Relationship model:

The entity relationship data model (ER model) is based on a perception of a real world that contains a collection of basic objects, called entities and of relationship among these objects and characteristics of an entity.

OR, The diagrammatic representation of entities, attributes and relationship is called ER diagram.

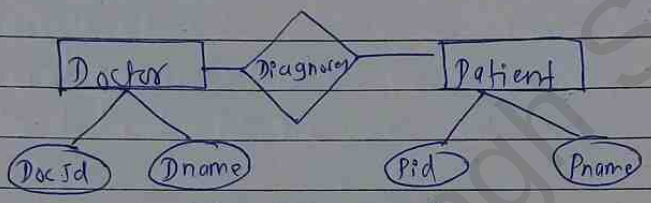


Fig: ER model

- Rectangle represents the entities. Eg: Doctor & Patient
- Ellipse represents the attributes. Eg: DocId, Dname, Pid, Pname.
- Diamond represents the relationship in ER diagrams. Eg: Doctor diagnoses the patient.

5. Object Oriented data model

Object model stores the data in the form of objects, classes and inheritance. This model handles more complex applications, such as Geographic information system (GIS), scientific experiments, manufacturing etc.

It is used in File Management System. It represents real world objects, attributes and

behaviour.

1.5 Database Application Architecture, Classification of DBMSs.

Database Application Architecture:

Database architecture can be 2-tier or 3-tier architecture based on how user are connected to the database to get their request done. They can either directly connect to the database or their request is received by intermediatory layer which synthesizes the request and then it sends to database.

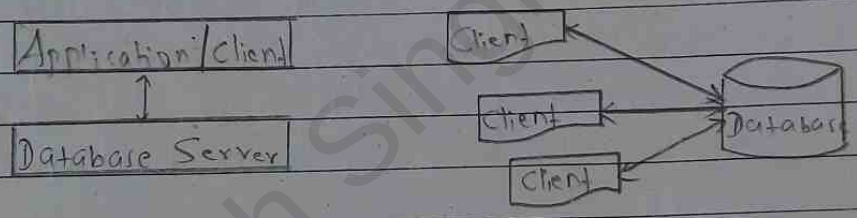
a) 2-tier Architecture:

2-tier architecture includes an Application layer between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the user.

An application interface known as ODBC (Open Database Connectivity) provides an API that allows client side program to call the DBMS. Most DBMS vendors provide ODBC drivers for their DBMS.

a) 2-tier architecture:

In 2-tier architecture, application program directly interacts with the database. There will not be any user interface or the user involved with database interaction. Imagine a front end application of School, where we need to display the reports of all the students who are opted for different subjects. In this case, the application will directly interact with the database and retrieve all required data. Here no inputs from the user are required. This involves 2-tier architecture of the database.



Advantages

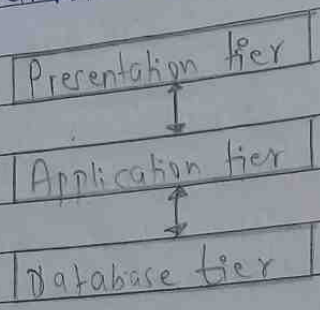
- Easy to understand as it directly communicates with the database.
- Requested data can be retrieved very quickly.
- Easy to modify
- Easy to maintain

Disadvantages

- Time consuming, when huge no. of users.
- It is little cost effective

b) 3-tier architecture:

3-tier architecture is most widely used database architecture. It can be viewed as below.



Layer = tier (same)

i) Presentation (user) tier

It is the layer where user uses the database. He does not have any knowledge about underlying database. He simply interacts with the database as though he has all data in front of him.

ii) Application (middle) tier:

It is the underlying ~~layer~~ program which is responsible for saving the details that you have entered and retrieving your details to show us in the page.

iii) Database (data) tier:

It is the layer where actual database resides. In this layer, all the tables, their mappings and the actual data is present. When you want to view your details in

the ~~database~~ browser, a request is sent to database layer by application layer. The database layer fires queries and gets the data. These data are then transferred to the browser (presentation layer) by the programs in the application layer.

Advantages
- Easy to maintain and modify.
- Improved security.
- Good performance

Disadvantages
- More complex
- More effort is required in terms of hitting the database.

Classification of DBMSs.

Unit-2

Entity Relationship Data Modeling

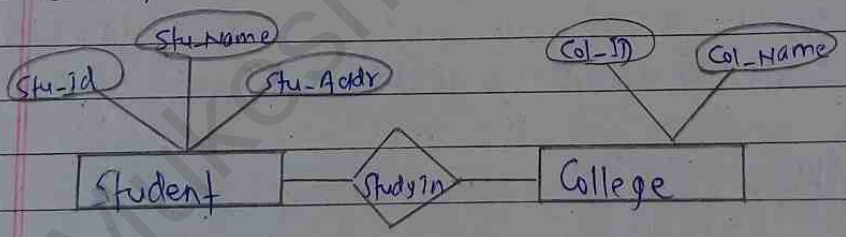
2.1. ER model and ER Diagrams, Components of ER Model, Types of Attributes.

Entity Relationship model (ER model) and ER Diagrams:

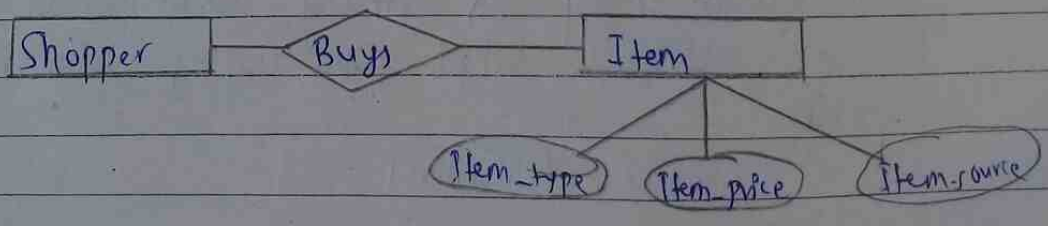
ER model is a model used for design and representation of relationships between data.

OR. An ER model, also called an ER diagram, is a graphical representation of an information system that depicts the relationship ~~between~~ among people, objects, places, concepts or events within that system.

Examples.



(1) Fig: Sample ER Diagram of Student

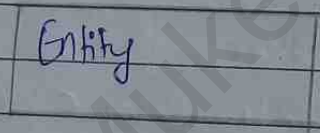


(2) Fig: Sample E-R diagram of shop

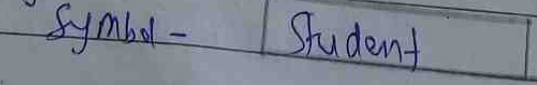
Importance of ER Model:
- ER model is plain and simple for designing the structure.
- It saves time.
- Without ER diagram you can't make a database structure and write production code.
- It displays the clear picture of the database structure.

Components of ER Model:

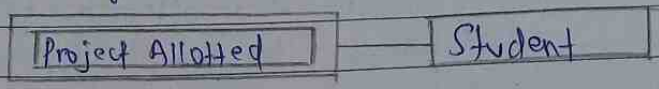
1. Entity:
Entities are definable things - such as objects, persons, concept or event that can have data stored about them.
Example - a customer, student, car or product.
* Typically shown as a rectangle



Two types of entities:
a) ~~Weak~~ Strong entity: A strong entity has primary key attribute which uniquely identifies each entity.
Symbol -



b) Weak entity: A weak entity does not have a primary key attribute and depends on other entity via a foreign key attribute

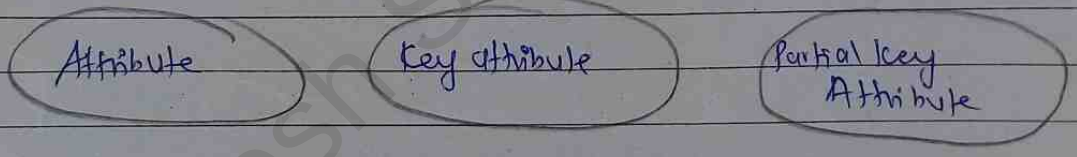
Symbol – 

weak entity.

2. Attributes:

Attributes are properties or characteristics of entities. An ERD attribute can be denoted as a primary key, which identifies a unique attribute, or a foreign key, which can be assigned to multiple attributes.

Often shown as an oval or circle.



Eg- ① Customer = (customer-id, customer_name, customer_city)

② account = (account_no, balance)

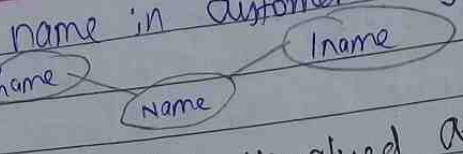
Types of attributes

@ Simple and Composite attribute

Attributes which cannot be devided into sub parts is called simple attribute for eg: customer_id in customer entity set.

Attributes that can be further divided into sub parts is called composite attribute: for eg:

Customer name in customer entity set. (fname, mname, lname)

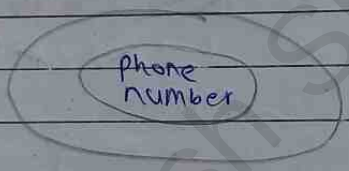


b) Single valued & multivalued attribute:

Attribute that can take only one value in every entry is called single valued attribute. for eg: Ramesh, kamal and Raj are the instances of entity 'student' and each of them owned a separate roll number.

An attribute which can hold more than one value in any entity, then it is called multi-valued attribute.

for eg: phone number of person.



Multi-valued attribute

c) Derived attribute:

Attribute whose value can be derived from the values of other related attribute or entities is called derived attribute.

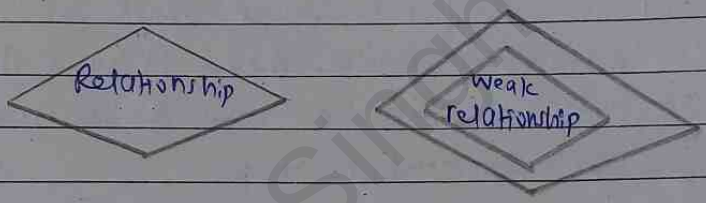
for eg: attribute age is derived from attribute date of birth.

3. Relationship:

A relationship is defined as bond or attachment between 2 or more entities. Normally, ~~a sentence~~ a verb in a sentence signifies a relationship.

A diamond is used to symbolically represent a relationship in the ER Diagram.

Relationship



2.2. Degree of Relationship, Constraints on ER model (Mapping Cardinalities and Participation Constraints), Keys and Types of Keys, Weak Entity Sets

Degree of Relationship

It signifies the number of entities involved in a relationship. Degree of relationship can be classified into following types:

a) Unary relationship:→ If only single entity is involved in a relationship, then it is a unary relationship. Example – An employee (manager)

Supervises another employee.

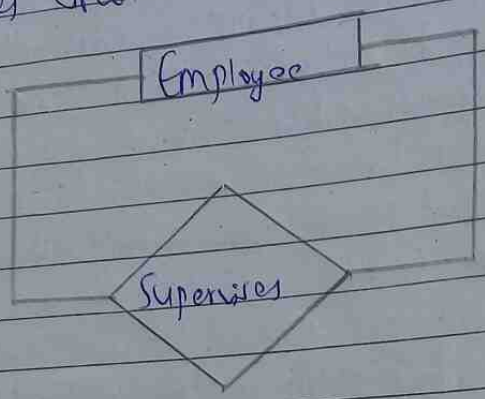


Fig- Unary relationship

b) Binary relationship → When two entities are associated to form a relation, then it is called binary relationship. For example - A person works in a company. The teacher - Student example

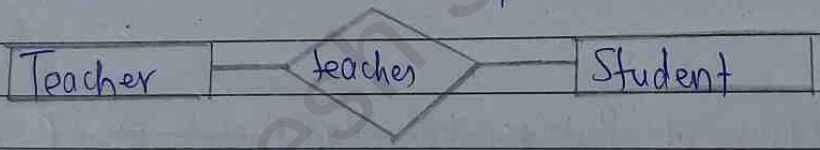


Fig- Binary relationship

Constraints on ER Model:

E-R model has the capability to enforce constraints. Two most important types of constraints are:

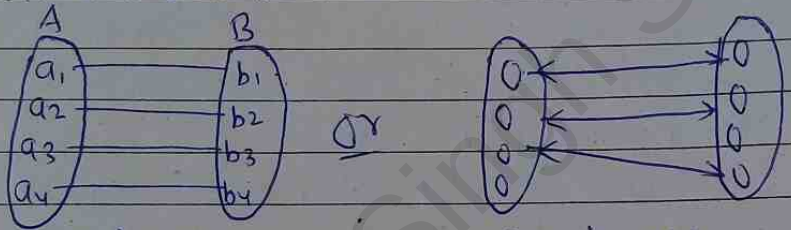
1. Mapping Cardinalities
2. Participation Constraints

1. 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.

a) One-to-one

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

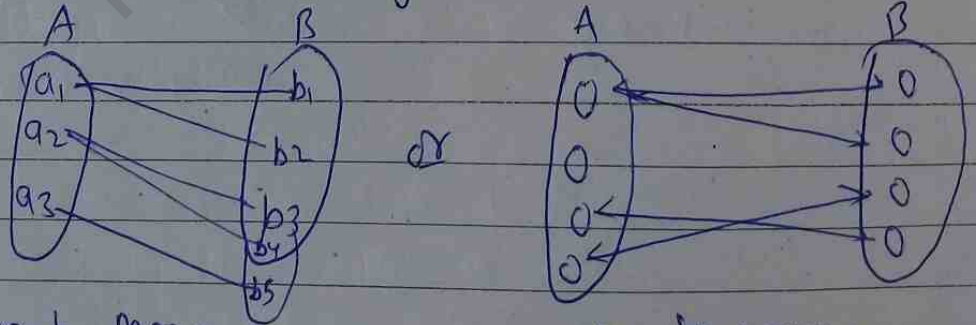


one to one

one-to-one

b) One-to-many

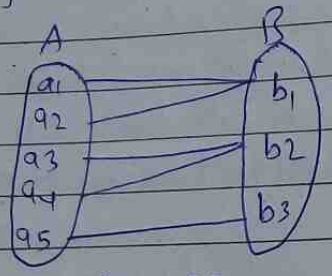
One entity from entity set A can be associated with more than one entities of entity set B but entity in B can be associated with atmost one entity in A



One-to-many

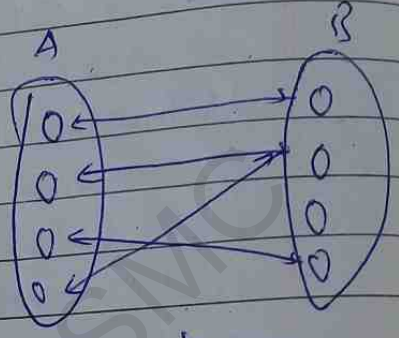
One-to-many

c) Many-to-one:
An entity in set A can be associated with atmost one entity in B but an entity in B can be associated with one or more entities in A.



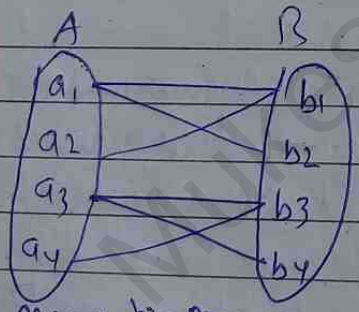
many-to-one

OR



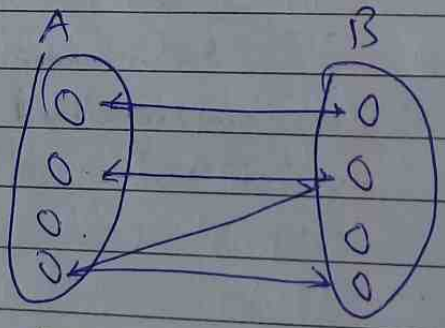
many-to-one

d) Many-to-many
One entity from A can be associated with more than entity from B and vice versa.



Many-to-many

OR

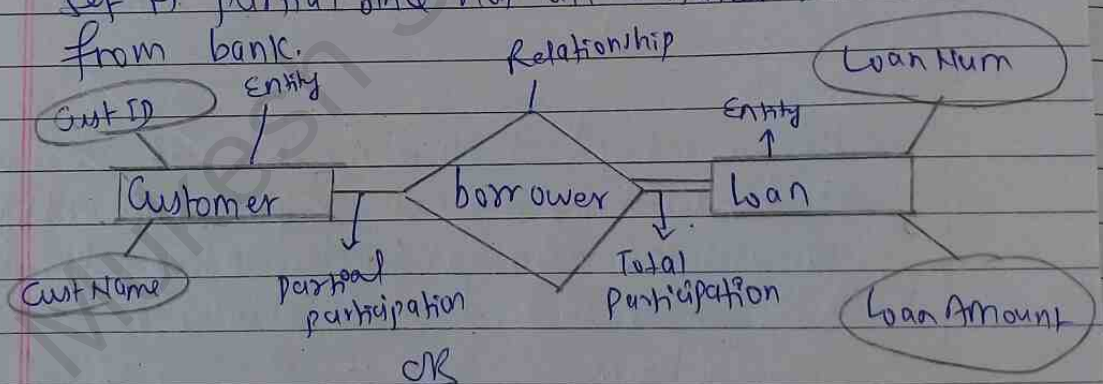


many-to-many

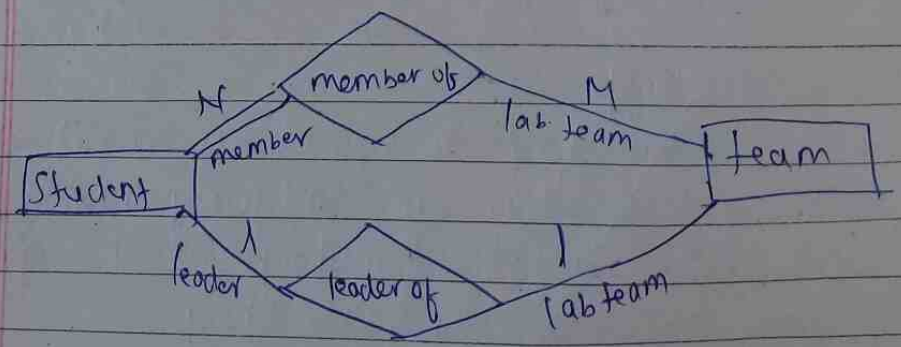
2) Participation Constraints

a) Total Participation:
Each entity in E is involved in the relationship R. Total participation is represented by double lines.
Example - Participation of loan in the relationship set borrower.

b) Partial Participation:
Not all entity in E are involved in the relationship R. Partial participation is represented by single lines.
Example - Customer entity set in borrower relationship set is partial since not all customer take loan from bank.



OR



Keys and Types of keys, Weak entity sets

Keys

A key is a field, or combination of fields, in a database table used to retrieve and sort rows in the table based on certain requirements.

Key plays an important role in relational database. It is used for identifying rows from table. It also establishes relationship among tables.

Eg

Customer id	Customer name	Address
C001	Amita	Itahari
C002	Sandhya	Dharan
C003	Amita	Biratnagar
C004	Sandhya	Kathmandu

To distinguish one entity from another entity in entity set, there must exist attributes whose values must not duplicate in entity set. It ensures no two entities in entity set can exist with same values for all attributes.

Types of keys

a) Super key:

Super key is defined as a set of attributes within a table that can uniquely identify each record within a table. Superkey is a superset of primary key and candidate key.

The super key contains a set of attributes, including the primary key, which can uniquely identify any data row in a table.
Example: STUD_NO (STUD_NO, STUD_NAME) etc

b) Candidate key:
A key with no redundant key or attribute is known as candidate key.
Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or set of attribute that can act as primary key for a table to uniquely identify each record in a table. There can be more than one candidate key.
Example - ① STUD_NO in STUDENT relation.
② STUD_NO as well as STUD_PHONE both

c) Primary key:
A primary key is a column or a group of columns in a table that uniquely identifies the rows in that table.
The primary key is selected from one of the candidate keys and becomes the identifying key of a table. It can uniquely identify any data row of the table.
Example - STUD_NO can be chosen as primary key from two candidate keys - STUD_NO & STUD_PHONE.

d) Secondary or Alternate key:
The candidate key other than primary key is called an alternate key.
Out of all candidate keys, only one gets selected as primary key, remaining keys are primary or secondary keys.
Example - STUD_NO as well as STUD_Phone both are candidate keys for relation STUDENT but STUD_Phone will be alternate key.

e) Composite key:
Key that consist of two or more attributes that uniquely identify any record in a table is called composite key.

f) Foreign key:
Foreign keys are columns (attribute value) in a table that acts as primary key in another table. Hence, the foreign key is useful in linking together two tables.
Example - OrderNo is primary key of ORDERS table and Customer No is a foreign key that points to primary key in the CUSTOMER table.

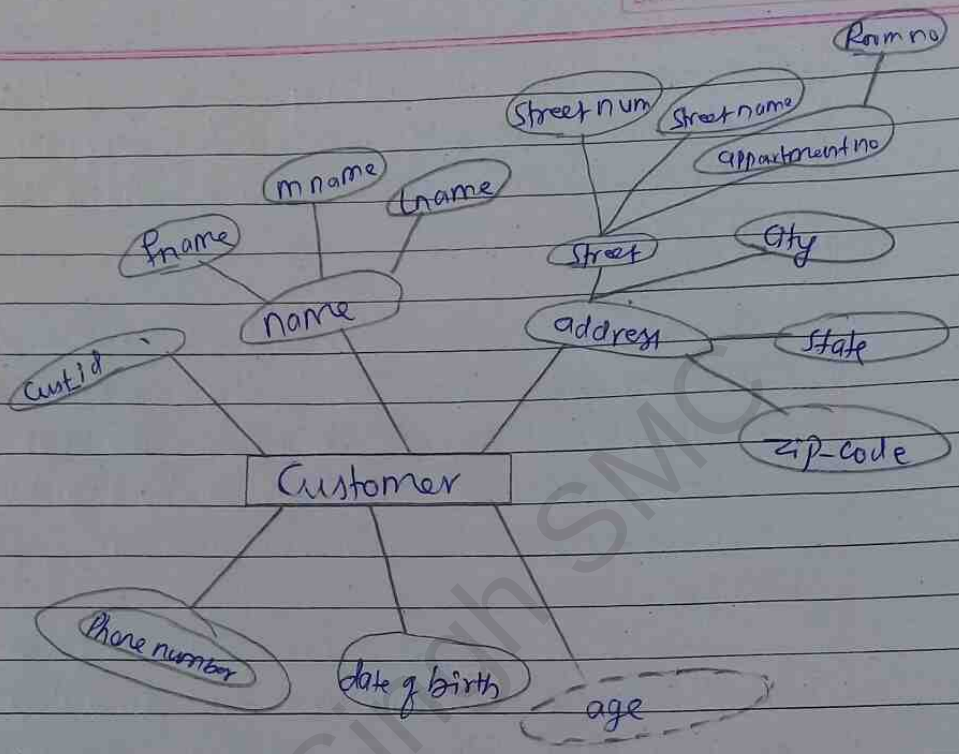


Fig: ER diagram with composite, multivalued and derived attributes.

2.3 Extended ER modelling: Subclass / Superclass Relationship, Specialization and Generalization, Constraints on Specialization / Generalization Aggregation

EER is a high-level data model that incorporates the extensions to the original ER model. Enhanced Entity-Relationship diagrams are advance database diagrams very similar to regular ER diagrams which represents the requirements and complexities of complex databases.

It is a diagrammatic technique for displaying Sub class and Superclass, Specialization, and Generalization, Union or category, Aggregation.

Features of EER model:

- EER creates a design more accurate to database schemas.
- It reflects the data property and constraints more precisely.
- It includes all modelling concepts of ER model.
- Diagrammatic technique helps for displaying the EER schema.
- It includes the concept of specialization and generalization.

A. Subclass and Superclass Relationship

Relation between sub class & super class is denoted by (d) symbol.

#Superclass

- Superclass is an entity type that has relationship with one or more subtypes.
- An entity cannot merely exist in database by being member of any superclass.

For Example: ~~Square, Circle, Triangle~~ Shape super class & have 3 sub groups as square, circle, triangle.

Subclass:

- Sub class is a group of entities with unique attributes.
- Sub class inherits properties and attributes from its superclass.

For example: Square, Circle, Triangle are the subclass of 'Shape' super class.

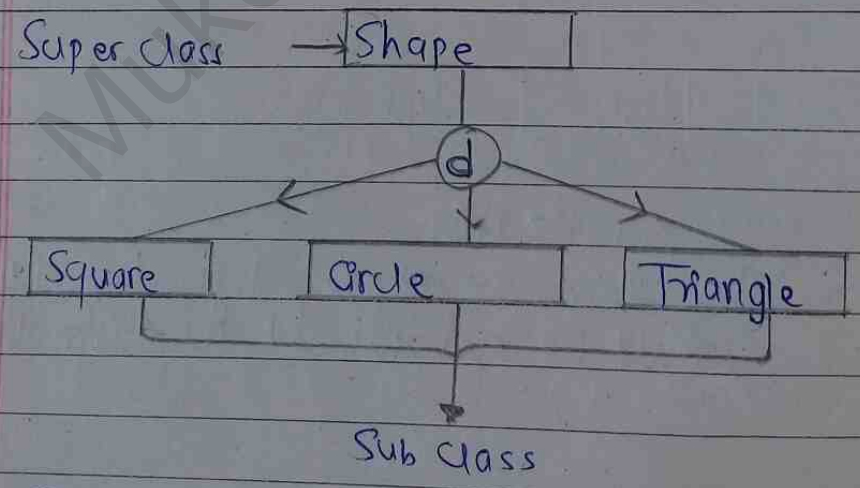


Fig: Superclass / Sub class Relationship.

B. Specialization and Generalization:

1. Generalization:

- Generalization is the process of generalizing the entities which contain the properties of all the generalized entities.
- It is a bottom up approach, in which two lower level entities combine to form a higher level entity.
- Generalization is the reverse process of Specialization.
- Example

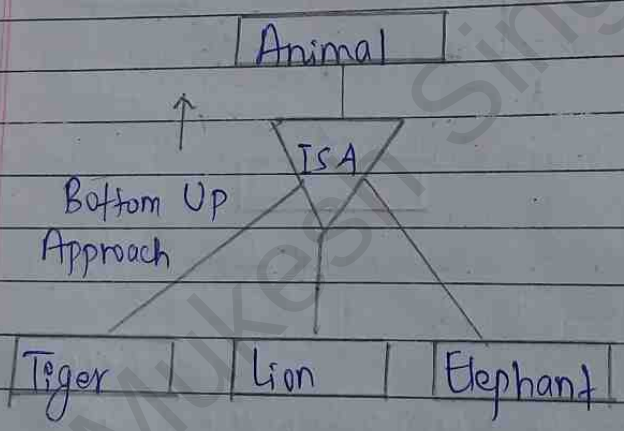


Fig: Generalization

In the above example Tiger, Lion, Elephant can all be generalized as Animals.

2. Specialization

- Specialization is a process ~~which~~ that defines a group of entities which is divided into sub groups based on their characteristic.
- It is a top-down approach, in which one higher entity can be broken down into two lower level entity.
- It defines one or more sub class for the super class and also forms the superclass/ sub class relationship.

Example

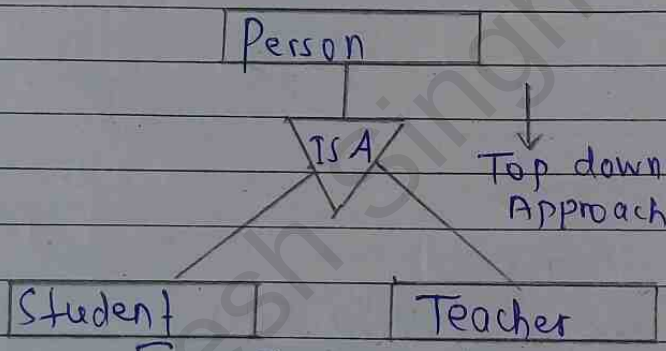


Fig: Specialization

In the above example, Person can be specialized as teacher or student, based on what role they play in school as entities.

C. Category or Union
- Category represents a single super class or sub class relationship with more than one super class.
- It can be total or partial participation.

For example: Car booking, Car owner can be a person, a bank (holds a possession on a car) or a company. Category (sub class) → Owner is a subset of the union of the three super class → Company, bank and person. A category member must exist in at least one of its super classes.

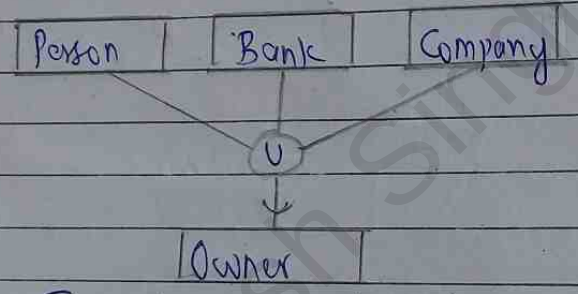


Fig: Categories (Union Type)

D. Aggregation:
- Aggregation is a process that represents a relationship between a whole object and its component parts.
- It abstracts a relationship between objects and viewing the relationship as an object.
- It is a process when two entity is treated as a single entity.

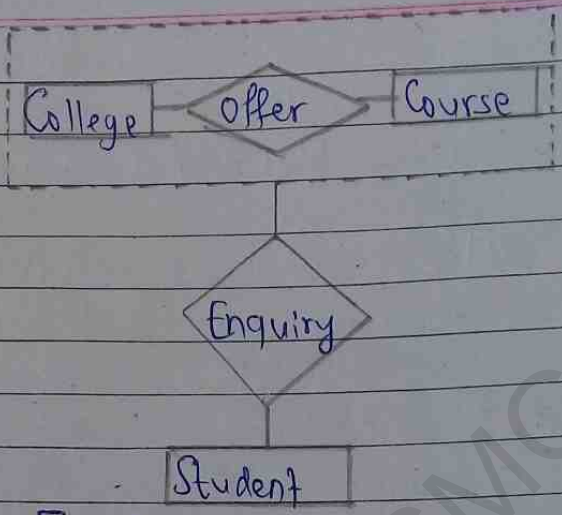


Fig: Aggregation

In the above example, the relation between College and Course is acting as an Entity in Relation with Student.

2.4. Relational Model: Introduction, Structure of Relational Databases, Schema Diagram, Mapping ER model to Relational Database

The database system which stores and displays data in tabular format of rows and columns, like spreadsheet, is known as Relational model. It is also known as Relational model. Examples – ORACLE, SQL, Ms-Access, Mysql etc.

Structure of Relational Databases:

In relational model, table is a major construct for representing data. Relational database consists of set of tables. A row in a table represents a relationship among set of values. So, table can be referred as a collection of such relationship.

Basic Construct

To illustrate the basic structure of database let us consider a table account:

a/c number	branch-name	balance
A1	Ktm	500
A2	Butwal	300
A3	Itahari	700
A4	Dharan	600

The columns of table "account" are: a/c number, branch-name & balance referred as

attributes. The set of permitted values for each attributes is known as domain of that attribute. For eg: set of all a/c number is a domain of attibutes a/c number.

Let D_1 denotes set of all a/c numbers, D_2 denotes set of all branch names and D_3 denotes set of all balances. Any row of table a/c must consist 3 ~~tpp~~ tuple (v_1, v_2, v_3) where v_1 is a/c number (v_1 is a domain of D_1) v_2 is branch name & v_3 is a/c balance. In general, we can express table account will contain only subset of all possible rows i.e a/c is a subset of $D_1 \cdot D_2 \cdot D_3$.

Customer-schema = (customer-id, customer-name, customer city)

Branch-schema = (branch-name, branch-city, assets)

Account-schema = (account-number, branch-name, balance)

Customer-schema = (customer-id, customer-name, balance)

Customer-schema = (customer-id, customer-name, customer-street, customer-city)

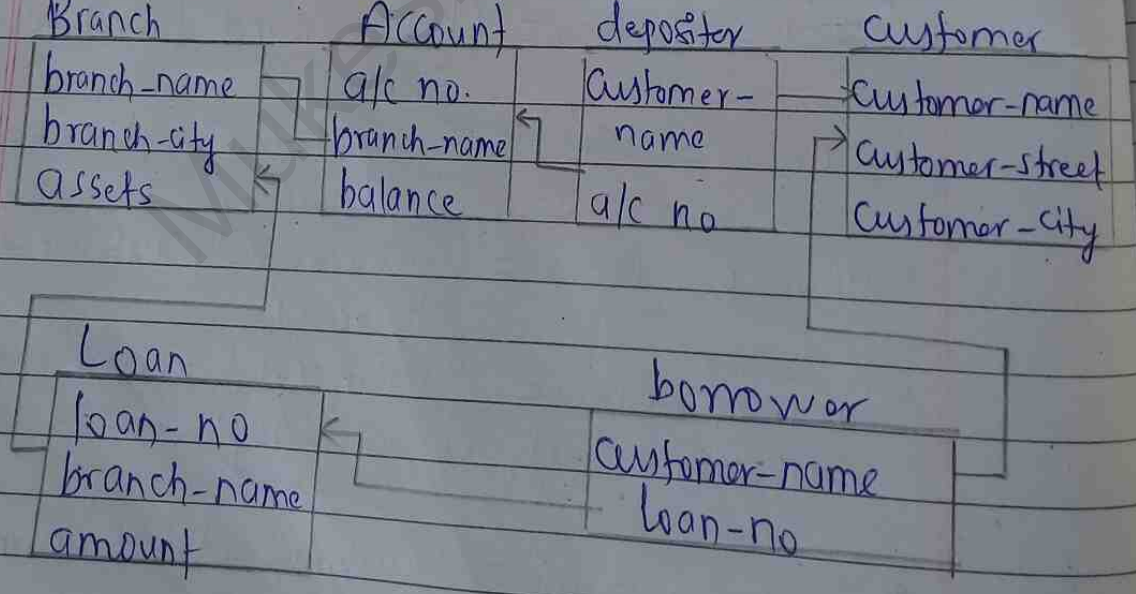
Depositor-schema = (customer-id, a/c-number)

Loan-schema = (loan-number, branch-name, amount)

Borrower-schema = (customer-id, loan-no)

Schema Diagram:

It is a graphical representation of database schema along with primary key and foreign key dependencies. In schema diagram, each relation is represented by box, where attributes are listed inside the box & relation name is specified above it. Primary key in relation is placed above the horizontal line that crosses the box. Foreign key in schema diagram appear as arrow from foreign key attribute of the referencing relation to the primary key of the referenced relation.



Unit-3

Relational Algebra

Relational database systems are expected to be equipped with a query language that can assist its user to query the database instances. There are two kinds of query languages - relational algebra and relational calculus. Every database management system must define a query language to allow users to access the data stored in database.

Query language is a language through which user request information from database. Query language can be categorized into procedural and non-procedural language.

In procedural language, user is required to specify sequence of operations to system to compute desired information.

Whereas in non-procedural language user need to specify required information without specifying special procedural for obtaining that information.

3.1 Introduction of Relational Algebra (RA), Fundamental operations of RA: select, Project, Set Union, Set Difference, Cartesian product and Rename Operations.

Introduction of Relational Algebra (RA)

Relational Algebra is a procedural query language used to query the database tables to access data in different ways.

Relational Algebra is ~~prod~~ procedural query language, which takes relation as input and generate relation as output. Relational algebra mainly provides theoritical foundation for relational databases and SQL.

Fundamental Operations:

1. Select Operation (σ)

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

Notation - $\sigma_p(r)$

where 'σ' stands for selection predicate and 'r' stands for relation. 'p' is prepositional logic formula which may use connectors like and, or, ~~and~~ & not. These terms may use relational operators like $=, \neq, \geq, <, >, \leq$ etc.

For example:

→ $\sigma_{Subject = "database"}(Books)$

Output - Selects tuples from books where subject is 'database'.

→ $\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$

Output - Selects tuples from books where subject is 'database' and 'price' is 450 ~~or those books~~ those books published after 2010.

→ $\sigma_{subject = "database" \text{ and } price = "450"}(Books)$

Output - Selects tuples from books where subject is 'database' and 'price' is 450.

2 Project Operation(Π)

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

Notation - $\Pi_{A_1, A_2, A_n}(r)$

where, A_1, A_2, A_n are attributes names of relation r.

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

for example:

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

→ Selects and projects columns named as subject and author from the relation Books.

Eg 2 - find account number and balance from account relation.

$\Pi_{Account\_number, balance}(Account)$

3. Set Union Operation (∪)

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

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

Notation - $r \cup s$

Where r and s are either database relations or relation result set (temporary relation)

For a union relation to be valid, the following conditions must hold-

- r and s must have the same number of attributes.
- Attribute domains must be compatible
- Duplicate tuples are automatically eliminated.

Example

$\Pi_{author}(Books) \cup \Pi_{author}(Articles)$

output → Projects the names of the authors who have either written a book or an article or both.

4. Set Difference (−)

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

Notation → $r - s$

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

For example

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

output - Provides the name of authors who have written books but not articles.

5. Cartesian Product (X)

Combines information of two different relations into one.

Notation - $r \times s$

Where r and s are relations and their output will be defined as -

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

for example

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

Output - Yields a relation, which shows all the books and articles written by tutorialspoint.

6. Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter rho ρ.

Notation - $\rho_x(E)$

where the result of expression E is saved with name of x.

3.2 Additional Relational Algebra Operations : Set Intersection, Natural join, Division and Assignment Operation.

1. The Set Intersection Operation ($\cap$)
   It finds tuples in both the relations. It is denoted by $\cap$.
   It does not add any power as:
   $r \cap s = r - (r - s)$
   To find all customers having both a loan and an account at the SFU branch, we write.
   $\Pi_{cname}(\sigma_{bname = "SFU"}(borrow)) \cap \Pi_{cname}(\sigma_{bname = "SFU"}(deposit))$.
   $r \cap s = \{t : t \in r \text{ and } t \in s\}$

2. The Natural join Operation ($|x|$)
   It is a binary operation and a combination of certain selections and a Cartesian product into one operation.
   · It is denoted as $|x|$. It is associative.

· For example, to find all customer having a loan at the bank and the cities in which they live, we need borrow and customer relations.

$\Pi_{borrow.cname, cCity}(\sigma_{borrow.cname = customer.cname}(borrow \times customer))$

To illustrate, we can rewrite it as
$\Pi_{cname, cCity}(borrow \; |x| \; customer)$

To find all customers who have both an account and a loan at the SFU branch.
$\Pi_{cname}(\sigma_{bname = "SFU"}(borrow \bowtie deposit))$

In contrast
It forms a Cartesian product of its two arguments. Then performs a selection forcing equality on those attributes those appear in both the relations. And finally removes duplicate attributes.

$r(R)$: r is a relation with attribute R.
$s(S)$: s is a relation with attribute S.
If $R \cap S = \phi$ ie they have no attribute in common then $r \bowtie s = r \times s$.

3. The division (quotient operation $(\div)$)
It is denoted as $\div$
It is suited to queries that include the phrase "for all".
Suppose we want to find all the customers who have an account for all branches located in Brooklyn.
Strategy: think of it as three steps

We can obtain the names of all branches located in Brooklyn by
$r_1 = \Pi_{bname}(\sigma_{bcity = "Brooklyn"}(branch))$

We can also find all cname, bname pairs for which the customer has an account by

$r_2 = \Pi_{cname, bname}(deposit)$

Now, we need to find all customers who appear in r_2 with every branch name in r_1.

The divide operation provides exactly those customers.

$\Pi_{cname, bname}(deposit) \div \Pi_{bname}(\sigma_{bcity = "Brooklyn"}(branch))$

which is simply $r_2 \div r_1$.

Formally,

Division can also be defined in terms of the relational algebra. We have $r(R)$ and $s(S)$ relations with $S \subseteq R$.

Then

$r \div s = \Pi_{R-S}(r) - \Pi_{R-S}((\Pi_{R-S}(r) \times s) - \Pi_{R-S, S}(r))$

4. The Assignment Operation

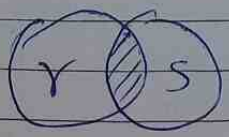
- Notation: $temp \leftarrow E$

  assigns the result of expression E to a temporary relation temp.
- Used to break complex queries to small steps.
- Assignment is always made to a temporary relation variable.

Example 1: Write r ∩ s in terms of ∪ and −

temp1 ← r − s

r − temp1



Example 2: Find all customer who has taken loan from bank as well as he/she has bank account.

Temp 1 ← Π Customer name (borrower)

Temp 2 ← Π Customer name (depositor)

Result ← Temp 1 ∩ Temp 2

Example 3: Cartesian product operation (×)

A	B
α	1
β	2

relation 'r'

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

relation 's'

r × s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Example of set intersection operation:

A	B
α	1
α	2
β	1

relation 'r'

A	B
α	2
β	3

relation 's'

A	B
α	2

r ∩ s

Example of natural join operation:

relation 'r'

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

relation 's'

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	1	β	b	δ

r ⋈ s

Example of Division operation:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

B
1
2

relation's'

A ÷ S

A
α
δ
ϵ

3.3. Extended Relational Algebra Operations: Generalized projection, Outer join and aggregate functions

1. Generalized Projection:

It extends the projection operation by allowing arithmetic functions to be used in projection list. The general structure is $(\Pi_{F_1, F_2, F_3 \ldots F_n}(E))$, where E is relational algebra expression.

Each $F_1, F_2, F_3 \ldots F_n$ are arithmetic expression involving constants and attributes in the schema in the E.

For eg: Suppose a relation

credit_info (Customer-name, Credit limit, Credit_balance)

Query: Find how much more each person can spend.

$\Pi_{\text{Customer\_name, Credit\_limit} - \text{Credit\_balance}}(\text{credit\_info})$

Eg 2: Suppose a relation

Employee (employee_id, ename, salary)

Query :- Find employee & their salary with corresponding bonus, assume that bonus for each employee is 10% of his/her salary.

$\Pi_{\text{ename, salary} * 1.10 \text{ as bonus}}(\text{employee})$

100 + 10

110/100 % = 1-10%

2. Outer join:

The outer join operation is extension of natural join operation. It has a capability to deal with missing information. There are three form of outer join operation.

i) left outer join (⟕)

Takes all tuples in the left relation t. If there are any tuples in the right relation that does not match with tuple in left relation, simply pad these right relation tuples with null value.

ii) Right outer join (⟖)

Takes all tuples in the right relation. If there are any tuples in left relation that doesn't match with tuple in right relation, simply pad these left relation tuples with null value.

iii) Full outer join (⟗)

→ Pad tuples from the left relation that does not match any tuples from the right relation.
→ Pad tuples from the right relation that does not match any tuples from the left relation.

For eg:
Consider relations

Loan_number	branch_name	amount
L01	B1	500
L02	B2	600
L03	B1	700

Relation loan

Customer_name	Loan_number
X	L01
Y	L02
Z	L07

relation_borrower

Loan_number	branch_name	amount	customer_name
L01	B1	500	X
L02	B2	600	Y

Left Outer join

Loan ⟕ borrower

Loan_no.	branch_name	amount	customer name
L01	B1	500	X
L02	B2	600	Y
L03	B1	700	null

Right outer join

Loan ⟖ borrower

Loan_no	branch_name	amount	customer_name
L01	B1	500	X
L02	B2	600	Y
L07	null	null	Z

full outer join

Loan ⟗ borrower

Loan_no	branch_name	amount	customer_name
L01	B1	500	X
L02	B2	600	Y
L03	B1	700	null
L07	null	null	Z

Aggregate functions & Operations

Aggregate function

They take a Collection of values and return a single value as a result some aggregate functions are avg: Avh: average value.

MIN: Minimum values
SUM: Sum of values
Count: number of values

The aggregate operation in relation

algebra is denoted by symbol 'g'. The general structure is $G_1, G_2, \ldots G_n$ g $F_1(A_1), F_2(A_2) \ldots F_n(A_n)$ (E), where E is any relational algebra expression.

→ $G_1, G_2, G \ldots G_n$ is a list of attribute on which is to be grouped. (It could be empty)
→ Each F_i is aggregate function.
→ Each A_i is attribute name

for eg:

A	B	C
α	α	7
α	β	7
α	β	3
β	β	10

Relation 'r'

$g_{sum(c)}(r)$ →

Sum-C
27

2) branch-name g sum (balance) as sum_balance (a/c)

3) branch-name g count (a/c-number) (account)

3.4. Database Modification: Insert, Delete and Update Operation

1) Insertion

Insertion of tuples can be viewed as a union operation that is made permanent. To insert data into relation, we can specify a tuple to be inserted or write a query whose result is a set of tuples to be inserted.

Insertion is expressed by:

$$r \leftarrow r \cup E$$

Where r is relation and E is a relational algebra expression.

Example

Insert information in the database specifying customer 'X' has 5000 in account A_1 at the Kathmandu city.

account ← account ∪ {(A_1, "Ktm", 5000)}

depositor ← depositor ∪ {("X", A_1)}

2) Deletion

Deletion of tuples is effectively a set-difference operation that is "permanent". Thus, we can write deletion as:

$$r \leftarrow r - E.$$

r is relation and E is relational algebra expression that determines the tuples

that are to be removed from r.

Example.

i) Delete all account in the B1 branch.

account ← account − σ branch_name = "B1" (account)

ii) Delete all records with amount in the range 0 to 50.

→ loan ← loan − σ amount ≥ 0 ∧ amount ≤ 50 (loan).

3.) Update

An update modification changes one or more values in a tuple without changing all values in tuple. In relational algebra, update operation is expressed by $r \leftarrow \pi_{F_1, F_2 \ldots F_n}(r)$

Where Each F_i is either:

→ i^{th} attribute of r, if the i^{th} attribute is not updated or

→ expression involving only constant & attributes of r, if the attribute is to be updated. It gives the new value for the attribute.

Example

① Increase balance by 5% to all branches

account ← $\pi_{account\_number, branch\_name, balance * 1.05}$ (account)

Eg:2

- Increase the balance by 6% for those account / whose balance is over 5000 and further rest of account increase balance by 5%.

→ account ← $\Pi_{account\_number,\ branch\_name}$ (balance * 1.06 ($\sigma_{balance > 5000}$ (account)) ∪ $\Pi_{account\_number,\ branch\_name,\ balance \times 1.05}$ ($\sigma_{balance \leq 5000}$ (account))

3.5 Null Values, Advantages and Limitations of Relational Algebra

Tuples may not have any values for some of the attributes. At that time the attribute is said to have null value. and is denoted by null. It simplifies an unknown value or a value that does not exist. The result of any arithmetic or comparison involving null is null.

Unit-4

Structured Query Language

Page No.
Date 07/10/2016

SQL was developed in 1970s in an IBM laboratory "San Jose Research Laboratory (now the Amaden Research Center). SQL is derived from the ~~sequ~~ SEQUEL one of the database language popular during 1970's. SQL established itself as the standard Relational Database Language. Two standard organizations ANSI and ISO currently promote SQL standards in Industry.

Basic terms

a) Query

Query is a statement requesting the retrieval of information.

b) Query Language:

Language through which user request information from database. These languages are generally higher level language than programming language. There are two types of query language:

i) Procedural language

User instructs the system to perform sequence of operation on the database to complete the desired result.

Eg: Relational algebra

ii) Non-Procedural language
User describes the desired information without giving a specific procedural for obtaining the desired information.
E.g: Tuple relational calculus and Domain relational calculus

Database languages

i) Data definition language (DDL)
Specify the database schema.
E.g: The following statement in SQL defines relation named "student"

```
CREATE TABLE student
(
student-id VARCHAR2(3);
address VARCHAR(30);
);
```

Student id	address

ii) Data Manipulation language:
A data manipulation language is a language that enables user to access or manipulate the data in database. The data manipulation means:
→ Retrieval of information stored in database
→ The insertion of new information into the database
→ Deletion of information from the database.

→ Modification of data in database

DDL:

SQL DDL provides commands for defining relational schemas, deleting schema, deleting ~~schemas~~ relation and modifying relational schema.

E.g: CREATE, ALTER, DROP

```
CREATE TABLE dept
(
dept_no NUMBER(2) PRIMARY KEY
dname VARCHAR2(20) NOT NULL
);
ALTER TABLE dept ADD (address, VARCHAR2(10));
ALTER TABLE dept MODIFY(dept_no NUMBER(10));
ALTER TABLE dept ADD UNIQUE (dname);
DROP TABLE dept
```

Data manipulation language in SQL:

SQL provides the following basic data manipulation statements.

E.g: SELECT, UPDATE, DELETE, INSERT

The SELECT Statement:

The ~~set~~ SELECT statement is most commonly used SQL statement. It is only a data retrieval statement in SQL.

The basic syntax for SELECT statement

Ans:
SELECT [DISTINCT/ALL] <attribute>1 FROM <relation>2 [WHERE <predicate>]3

where,
1. <attribute> → Columns name
2. <relations> → table name
3. <predicate> → Conditions

- SELECT, FROM are necessary clause. WHERE is optional clause.
- DISTINCT/ALL is optional clause.
- SELECT clause is used to list the attributes that are required in the result in query.
- FROM clause list the relation from where specified attributes are to be selected.
- WHERE clause are used to specify the condition while we require retrieving of particular data. One or more condition can be specified using where clause by using WHERE clause by using SQL logical connectives. (<, <=, >, >=, =, <>)

- DISTINCT keyword is used to eliminate duplicate values.

- ALL keyword is used to explicity allow duplicate.

Example:
Consider a relational database

Student (roll, name, address)
teacher (tid, tname, taddress, salary)
teaches (roll, tid, tsubject, semester)

Q1. Find the name of the student who live in Kathmandu.

→ SELECT Sname FROM student
Saddress = 'Kathmandu'

Q2. Display all the records of the teacher table.

→ SELECT * FROM teacher

Q3. Find the name of a teacher who teaches DBMS.

→ SELECT tname FROM teacher, teaches
WHERE teacher.tid = teaches.tid AND
tsubject = 'DBMS'

Q4. Find the name of the students studying in 4th semester.

→ SELECT Sname FROM student, teaches
WHERE student.roll = teaches.roll AND
Semester = '4th'.

Q5. Find the name of the teacher who has salary > 50000.

→ SELECT fname FROM teacher.
WHERE salary > 50000

Insert Statement:

The insert statement is used to insert or populate the data into the database table.
The common syntax of Insert statement is

INSERT INTO <tablename> Values
(Value1, Value2, ... ValueN);

Eg:
INSERT INTO student (1, 'Ramprakar Shah', 'Itahari');
OR
INSERT INTO Student (Roll, Sname, Saddress)
VALUES (1, 'Ramprakar Shah', 'Itahari')

DELETE Statement

The DELETE statement is used to remove or delete the record from the database table.
The common syntax of DELETE statement is
DELETE FROM <tablename> WHERE <Condition>

* DELETE the records of students whose address is Itahari.

→ DELETE FROM student WHERE Saddress = "Itahari".

UPDATE statement

The UPDATE statement is used to modify the value of the attribute of the table. The common syntax is:

UPDATE <tablename> SET <attribute> = <newvalue> WHERE <Condition>

E.g: Increase the salary of employ by 10% of officer level.

→ UPDATE employee SET Salary = 1.1 * Salary WHERE Level = 'Officer'

4.6 DDL: Domain Types in SQL, Create, Alter and Drop Statements

1. Create Statement:
It is used to create database or table

Syntax - To define database
CREATE DATABASE dbname;
eg: CREATE DATABASE Sulkuna;

* To define table
CREATE TABLE <tablename>
(<attribute 1> <Domain-Type 1>, ~~<Attribute 2>~~
<attribute 2> <Domain-Type 2>,
:
<Constraint> <Const. name>);

Eg:
CREATE TABLE student (Roll INT NOT NULL
Name VARCHAR(50),
Address VARCHAR(50),
Level CHAR (10)
PRIMARY KEY (Roll);

2. ALTER statement:

The ALTER statement is used to modify the database schema ~~to modify the~~ or tables. The common use of ALTER statement is to add new field and adding the constraint in the existing database.

Eg(1): Suppose the table employee already exist and ~~new~~ Now, we want to make Dno attribute as foreign key.

→ ALTER TABLE employee ADD CONSTRAINT FOREIGN KEY (Dno) REFERENCES Depart (Dno)

Eg(2) Suppose There is a table student with roll, name, address as an attribute and we want to add another attribute level.

→ ALTER TABLE Student ADD Column Level CHAR(10);

3. DROP statement

The DROP statement is a DDL statement which is used to remove ~~or~~ or delete table from the database or remove attribute from the table. The DROP statement removes the table or attribute permanently from the

database

To drop table
DROP TABLE <tablename>
To drop dubibute
ALTER TABLE <tablename> DROP <Columnname>

Questions for practice

Creak:
① branch (branch_name, branch city, assets)
② Customer (Customer_id, customer_name, customer_city, & branch_name)
③ Account (account_number, branch_name)
④ Depositor (account_number, customer_id)
⑤ loan (loan_number, loan amount)
⑥ borrower (loan_number, customer_id)

Solutions

① Sol^n
→ CREATE TABLE BRANCH
(branch_name VARCHAR(50) NOT NULL,
branch_city VARCHAR(50),
assets NUMERIC(10,2)
PRIMARY KEY (branch_name));

(2)

→ CREATE TABLE Customer
(Customer_id INT NOT NULL,
Customer_name VARCHAR(50) NOT NULL,
Customer_city VARCHAR(50),
branch_name VARCHAR(50);
PRIMARY KEY (customer_id),
FOREIGN KEY (branch_name)
REFERENCE branch (branch_name))

(3)

→ CREATE TABLE account
(account_number INT NOT NULL,
branch_name VARCHAR(20),
PRIMARY KEY (account_number),
FOREIGN KEY (branch_name) REFERENCE
branch (branch_name)

(4)

→ CREATE TABLE depositer
(account_number CHAR(10), NOT NULL
Customer_id INT,
PRIMARY KEY (account_number, Customer_id)
FOREIGN KEY (account_number) REFERENCE
account (account_number)
FOREIGN KEY (account_number) REFERENCE
account (account_number)
FOREIGN KEY (~~account~~ Customer_id) REFERENCE
Customer (Customer_id))

(5)

→ CREATE TABLE loan
loan (loan_number, loan amount)
(loan_number INT,
loan amount INT,
PRIMARY KEY (loan_number)
)

(6)

→ CREATE ~~loan~~ TABLE borrower
(
loan_number INT,
Customer_id INT,
PRIMARY KEY (loan_number, customer_id)
FOREIGN KEY (loan_number) REFERENCE
loan(loan_number),
FOREIGN KEY (Customer_id) REFERENCE
customer(Customer_id)
).

9.4 Aggregate functions, Group by clause and having clause

The aggregate function takes the collection of values as input and returns a single value as output. The common aggregate function in SQL are AVERAGE, SUM, COUNT, MIN, MAX

AVERAGE – Returns an average value from the setup value.

SUM – Returns a sum of the collectional values.

COUNT – Count the number of values or tuples.

MIN – Returns the minimum value from the collectional values.

MAX – Returns the maximum value from the collectional values.

Example① Find the minimum salary of the teacher.

→ SELECT MAX(Salary) as max-salary from teacher.

Eg② Find the number of students.
→ SELECT COUNT(Roll) FROM student.

Eg(3) Find the minimum, average, maximum salary
→ SELECT MIN(Salary), AVG(Salary, MAX salary)
FROM teacher.

String operator 'LIKE':
In SQL, there is a string operator LIKE which is used to match the pattern of string or character. The % symbol is used for string and _ symbol is used for the character matching.

Example

1. (a) List the name of the student whose name starts from S.
→ SELECT Name FROM student WHERE name LIKE S%

2. Find the name and address of the student whose address consists of man.
→ SELECT name, address FROM student WHERE address LIKE % man %

3. Find the name of the teacher whose name consists of l in third position.
→ SELECT fname FROM teacher WHERE fname LIKE '_ _ l %'

Ordering

The clause ORDER By is used to order (sort) the value use. Asc is used for ascending order. DESC is used for descending order. By default, it is ascending.

Example

1 find the name and address of student & order by name in ascending order.

→ SELECT name, address FROM student ~~order by~~ ORDER BY name ASC.

SET Operations:

In SQL, there are some operations: union, intersect. & EXCEPT are called SET operations

a) UNION Operation

The UNION Operation is used to perform the union ~~UNION~~ of results from the relation. It automatically removes the duplicate. To retain duplicate we use UNION ALL.

b) INTERSECT Operation

It is used to find common values.

c) EXCEPT operation

It is used to retrieve the data which is equivalent to set difference.

Example

1) Find all customer who have a loan and account or both.

→ (SELECT Customer_name FROM depositor)
UNION
(SELECT customer_name FROM borrower)

2) Find all customer who have both loan and account.

→ (SELECT Customer_name FROM depositor)
INTERSECT
(SELECT Customer-name FROM borrower)

3) Find all customers who have an account but not loan.

→ (SELECT Customer_name FROM depositor)
EXCEPT
(SELECT Customer_name FROM borrower)

GROUP By Clause

The GROUP BY clause is used to group the similar value. It is used with aggregate function.

Example

1) Find the number of depositor from each branch.

→ SELECT branch_name, COUNT (DISTINCT customer_name) FROM depositor, account WHERE depositor account_number = account. account_number

Having Clause

The having clause is also used with aggregate function.

Eg:

@ find the name of branches where the average account balance is more than 1200.

→ SELECT branch_name, AVG(balance) FROM account GROUP BY branch_name HAVING AVG(balance) > 1200

Nested ~~SUBQUERIES~~: Subsequenes

The SQL has a mechanism for the nesting subqueries. A sub-query is a ~~set from~~ SELECT-FROM-WHERE expression ~~where~~ that is nested within another query.

The common use of sub-query is to perform test for set membership, set comparison and set cardinality. The connection IN, NOT IN, SOME, ALL are closed in nested sub-queries.

4.7

Eg.
① find all the customers who have both an account and a loan at the bank.

→ SELECT DISTINCT Customer-name FROM borrower WHERE Customer-name IN (SELECT Customer-name FROM depositor)

② find all branches that have greater assets than that of some branch located in Brooklyn.

→ SELECT branch-name FROM branch WHERE assets > some (SELECT assets FROM branch WHERE branch-city = 'Brooklyn'

Joint Operation (reffered to relational algebra)

4.7. View and Modification of Views, Embedded and Dynamic SQL:

→ A view in SQL terminology is a single table that is derived from other tables. These other tables can be base tables or previously defined views. A view does not necessarily exist in physical form; it is considered a virtual table in contrast to base table, whose tuples are actually stored in the database. This limits the possible update operations that can be applied to view.

Updating of view is complicated and can be ambiguous. In general, an update on a view defined on a single table without any aggregate function can be mapped to an update on the underlying base table under certain condition.

Syntax:

CREATE VIEW <view name> AS <query expression>

e.g.

CREATE VIEW vw-emp AS SELECT emp.no, ename, job FROM emp

Some Domain types in SQL (Data types in SQL)

1.) CHAR(n):
Fixed length character string with user specified length 'n'.

2) VARCHAR2(n) - A variable length character string with user specified with length 'n'.

3) NUMBER(P,S): Holds fixed or floating point numbers. P determines the maximum length of data and S determines the number of place to the right of decimal.

4) NUMBER(n) - Holds fixed number with specified length 'n'.

5) INT - An integer or small int.

6) FLOAT(n) - A floating point number with precision of at least 'n' digits.

7) DATE - Represents data and time. The Standard format is : DD-MM-YYYY

~~8) Long~~

8) LONG - Used to store variable length character string containing upto 2 gb.

Unit-5

Integrity Constraints

5.1. Concept and Importance of Integrity Constraints, Data Integrity

Integrity Constraints are those constraints in database system which guard against invalid database operation or accidental damage to the database by ensuring the authorized changes to the database. It does not allow to loss of data consistency in database. In fact, integrity constraints provide a way of ensuring that changes made to the database by authorized user, do not result in a loss of data consistency.

Example of integrity constraints in E-R model:

- Key declaration : candidate key, primary key
- Form of relationship : mapping cardinalities :- one to one, one to many.

In database management system, we can enforce any arbitrary predicate as integrity constraints. But it adds overhead to the database system.

Integrity constraints are the rules that are to be applied on database columns to ensure the validity of data.

Data integrity is the maintenance of, and the assurance of the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes or retrieves data. It is opposite of data corruption. Data integrity is not to be confused with data security, the discipline of protecting data from unauthorized parties.

5.2. Domain Constraints: Not Null Constraints, Unique Constraints, Primary key Constraints, Check Constraints.

Domain constraints are rules that define legal values for columns.

Domain Constraint is one of the elementary form of integrity constraint that helps to maintain accuracy and consistency.

Domain Constraints are user defined data types and we cannot define them like this:

Domain Constraint = data type + Constraints (NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT).

→ Domain Constraints helps to avoid common error such as date: 30th February, 2010...

→ A domain is defined as set of all unique values that can be allowed for an attribute.
for example; a domain of day-of-week is Sunday, Monday, Tuesday ---- Saturday.

CHECK, UNIQUE, NOT NULL, PRIMARY KEY are examples of domain constraints.

1. Not NULL Constraints:
This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
for example, the below query creates a table Student with the fields ID and NAME as NOT NULL. That is, we are bound to specify values for these two fields every time we wish to insert a new row.

Eg

```
CREATE TABLE Student
(
ID INT(6) NOT NULL,
NAME varchar(10) NOT NULL,
ADDRESS varchar(20)
);
```

2) Unique Constraint

This constraint helps to uniquely identify each row in a table i.e. for a particular column, all the rows should have unique values (must not be repeated). We can have more than one UNIQUE columns in a table.

For example, the below query creates a table student where the field ID is specified as UNIQUE i.e no two students can have the same ID. ~~Unique~~

Eg

```
CREATE TABLE Student
(
ID INT(6) NOT NULL UNIQUE,
NAME varchar(10),
ADDRESS varchar(20)
);
```

3 Primary Key Constraints:

Primary key is a field which uniquely identifies each row in the table. If a field in a table as primary key, then the field will not be able to contain NULL values as well as all the rows should have unique values for the field. So, in other words, we can say that this is combination of NOT NULL and UNIQUE constraints.

A table can have only one field as

primary key. Below query will create a table named student and specifies the field ID as primary key.

```
CREATE TABLE Student
(
ID int(6) NOT NULL UNIQUE,
NAME varchar(10),
ADDRESS varchar(20),
PRIMARY KEY (ID)
);
```

4. Check Constraints:

Using CHECK Constraint, we can specify a condition for a field, which should be satisfied at the time of entering values for this field.

For example, the below query creates a table student and specifies the condition for the field AGE as (AGE >= 18). That is, the user will not be allowed to enter any record in the table with AGE < 18.

```
CREATE TABLE Student
(
ID int(6) NOT NULL,
NAME varchar(10) NOT NULL,
AGE int NOT NULL CHECK (AGE >= 18)
);
```

5. foreign key Constraint

foreign key is a field in a table which uniquely identifies each row of another table. That is, this field points to primary key of another table. This usually creates a kind of link between the tables.

Syntax

```
CREATE TABLE Orders.
(
O_ID int NOT NULL,
ORDER_NO int NOT NULL,
C_ID int,
PRIMARY KEY (O_ID),
FOREIGN KEY (C_ID) REFERENCES Customers (C_ID)
)
```

6 Default Constraints

This constraint is used to provide a default value for the fields. That is if at the time of entering new records in the table if the user does not specify any value for these fields then the default value will be assigned to them.

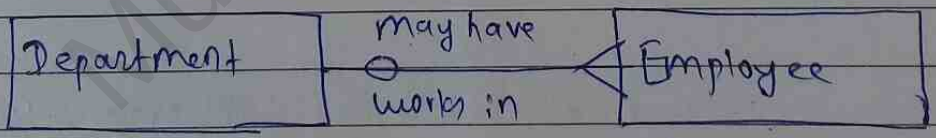
for example, the below query will create a table named Student and specify the default value for the field AGE as 18.

```
CREATE TABLE Student
(
ID int (6) NOT NULL,
NAME varchar(10) NOT NULL,
AGE int DEFAULT 18
);
```

5.3 Referential Integrity: Using referential integrity, Cascading Actions

Referential integrity is the relationship between the tables of a database, i.e. the data of one table does not contradict the data of another table. Specifically, every foreign key value in a table must have a matching primary key value in the related table. This is the most common type of integrity constraint. This is used to manage the relationships between primary key and foreign keys.

Example:



** Department No

** Employee No
* Department No.

department (depart no, dname)
employee (emp no, ename, depart no).

Here, Department no. is the foreign key in the employee table and the primary key in department table

Example 2

Primary Table

Company ID	Company Name
1	Apple
2	Samsung

?

Related Table

Company ID	Product ID	Product Name	~~Associated Record~~
1	1	iPhone	Associated Record ✓
15	2	Mustang	Orphaned Record ✗

Here, the related table contains a foreign key value that doesn't exist in the primary key field of the primary table (i.e Company field). This has resulted in "orphaned record".

So, referential integrity will prevent users from:

- Adding records to a related table if there is no associated record in the primary table.
- Changing values in a primary table that result in orphaned records in a related table.
- Deleting records from a primary table if there are matching related records.

Cascading Actions

Cascading referential integrity constraints are foreign key constraints that tell SQL server to perform certain actions when a primary key field in a primary key-foreign key relationship is updated or deleted.

Syntax

```
Create account
foreign key (branch_name) references branch
On delete cascade
on update cascade
- - - - -
- - - - - - - )
```

- On delete cascading :→ If a deleting tuples in branch results referential integrity constraints violation, it also delete tuples in relation on account that refers to the branch that was deleted.

- On update cascading :→ If an update tuples in branch results referential integrity constraints violation, it also ~~delete~~ update tuples in relation on account that refers to the branch that was updated.

5.4. Assertions and Triggers: Creating and Deleting Assertions, Creating and Deleting Triggers, Assertions vs Triggers

Assertions

An assertion is a predicate expressing a condition we wish the database to always satisfy.
- Domain constraints, functional dependency and referential integrity are special forms of assertion.

If a constraint cannot be used in this form, we use an assertion.

Creating Assertions

Syntax:

```
CREATE ASSERTION <Assertion_name>
CHECK (predicate)
```

Example

```
CREATE ASSERTION assert
CHECK (0=(
        SELECT COUNT(*)
        FROM Video
        WHERE my_date = CURRENT Date
        GROUP
          BY my_user
        HAVING COUNT(*) >= 10
        ));
```

Triggers:

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The triggers are mostly used for maintaining the integrity of the information on the database.

For example, when a new record (representing a new worker) is added to the employees table, new records should also be created in the tables of the taxes, vacations and salaries. Triggers can also be used to log historical data, for example to keep track of employees' previous salaries.

Creating Triggers

Syntax

~~CREATE trigger [trigger_name]~~

CREATE TRIGGER [trigger_name]
[BEFORE | AFTER]
{INSERT | UPDATE | DELETE} [of Column]
On [table_name]
[For EACH Row] [WHEN Conditions]
[TRIGGER body]

Example

CREATE TRIGGER new_employee
AFTER INSERT ON hr_table
BEGIN
INSERT INTO benefits_table ('name', 'street', 'workid')
VALUES ('john', '123 Any street', '3022')
END.

Unit-6

Relational Database Design

6.1. Introduction, Database Modification Anomalies, functional Dependencies (FDs), Types of FDs, FD. Inference Rules

A relational database is a set of formally described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables. The standard user and application programming interface (API) of a relational database is the Structured Query Language (SQL). SQL statements are used both for interactive queries for information from a relational database and for gathering data for reports.

Database modification Anamolies:

To understand the anamolies let us take an example of:

Student

roll no	name	branch	HOD	Office-tel
401	Ram	ICT	Mr. X	53337
402	Shyam	ICT	Mr. X	53337
403	Hari	ICT	Mr. X	53337
404	Gita	ICT	Mr. X	53337

In the above table, we have data of four ICT students as we can see data for the fields, branch, HOD, office telephone are repeated for the students who are in the same branch

in the College. This is data redundancy.

i) Insertion Anamoly:
An insertion anamoly is the inability to add data to the database due to absence of other data.
Suppose, for a new admission until and unless a student enrolls for a branch, data of the student cannot be inserted or else we will have to set the branch information as null. Also if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students. These scenaries are nothing but an insertion anamolies.

ii) Update Anamoly
What if Mr. X leaves the College or is no longer the HOD of ICT department? In that case all the student records will have to be updated, and if by mistake, we miss any record, it will lead to data inconsistency. This is update anamoly.

iii) Deletion Anamoly
In our student table, two different informations are kept together i.e student information and branch information. Hence, at

the end of academic year, if students' records are deleted, we will loose the branch information. Then is deletion anamoly.

Functional ~~Dependency~~ Dependencies (FDs)

Functional dependency is a relationship that exists when one attribute uniquely determines another attribute.

If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X -> Y, which specifies Y is functionally dependent on X. Here X is a determinant set and Y is a dependent attribute. Each value of X is associated with precisely one Y value.

Functional dependency in a database serves as a constraint between two sets of attributes. Defining functional dependency is an important part of relational database design and contributes to aspect normalization.

Functional dependency is an association between two attributes of the same relational database table.

Types of functional Dependencies

1. fully functional Dependency
An attribute is fully functional dependent on another attribute, if it is functionally Dependent on that attribute and not on any of its proper subset.
For a given relation R, functional dependency $X \rightarrow Y$, Y is said to be fully functionally dependent on X if there is no Z, where Z is the proper subset of X such that $Z \rightarrow Y$.

Example
In a relation R(A, B, C, D) with candidate keys AB and BC the functional dependency $ABC \rightarrow D$ is a full functional dependency.
where as.
$B \rightarrow D$ is not full dependency.

2. 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, $A \rightarrow B$ is trivial functional dependency if B is a subset of A (i.e $B \subseteq A$).
Also, $A \rightarrow A$ & $B \rightarrow B$ are trivial.

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}.

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

3. Partial functional Dependency:
Partial FD is a kind of FD that occur when primary key must be candidate key and non prime attribute depends on the subset/part of candidate keys.
Partial FD occurs when a non prime attribute is functionally dependent on a part of a candidate key.

Example 1
Seller (FD, Product, Price

Candidate key: Id, Product
Non primitive attribute: Price

Price attribute only depends on Product attribute which is a subset of candidate key not the whole candidate key (Pid, Product) key. It is called partial dependency.
So, we can say that Product → Price is partial dependency.

Example
Table: Stud_Id, Course_Id, Stud_name, Course_name

Where, Primary key = Stud_Id + Course_Id

Then, to determine name of student we use only Stud_id, which is part of primary key.
{Stud_Id} → {Stud_Name}

Hence, Stud_name is partially dependent on Stud_Id. This is partial dependency.

4. Transitive functional dependency:

- When an indirect relationship causes functional dependency, it is called Transitive functional dependency.
If P → Q and Q → R is true, then P → R is a transitive dependency.

- A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. A transitive dependency can occur only in a relation of three or more attributes.
For Example, let A, B and C are three distinct attributes in a relation. Suppose all three of the following conditions hold:

- A -> B
- B does not -> A
- B -> C

Then the functional dependency ~~A B~~ A -> C (which follows from 1 and 3 by the axiom of transitivity) is a transitive dependency.

Example
Table: Book, Author, Author ~~Age~~ Nationality

The functional dependency {Book} -> {Author ~~Age~~ Nationality} applies; that is if we know the book we know the author's nationality.

- {Book} -> {Author}
- {Author} does not -> {Book}
- {Author} -> {Author Nationality}

Therefore, {Book} -> {Author Nationality} is a transitive dependency.

Transitive dependency occured because a non-key attribute (Author) was determining another non-key attribute (Author Nationality).

5. Multivalued Dependency

When existence of one or more rows in a table implies one or more other rows in the same table, then the Multi-valued dependencies occur.

If a table has attributes P, Q and R, then Q and R are multi-valued facts of P.

It is represented by double arrow:

->->

Example

P ->-> Q

Q ->-> R

In the above case, Multivalued dependency exits only if Q and R are independent attributes.

FD Inference Rules

The inference rule is a type of assertion. It can apply to set of FD to derive other FD. Using the inference rule, we can derive additional functional dependency from the initial set.

There functional dependency has 6 types of inference rules.

i) Reflexive Rule (IR_1)

In the reflexive rule, if Y is a subset of X, then X determines Y.

i.e If $X \supseteq Y$ then $X \rightarrow Y$

Example

$X = \{a, b, c, d, e\}$

$Y = \{a, b, c\}$

ii) Augmentation Rule (IR_2)

The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

i.e If $X \rightarrow Y$ then $XZ \rightarrow YZ$

Example

For R(ABCD), if $A \rightarrow B$ then $AC \rightarrow BC$

iii) Transitive Rule (IR_3).

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

i.e If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

iv) Union Rule (IR_4)

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

i.e If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

v) Decomposition Rule (IR5)
Decomposition rule is also known as project rule. It is the ~~reserve~~ reverse of union rule. The rule says, if X determines Y and Z, then X determines Y and X determines Z separately.
i.e. If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

vi) Pseudo transitive Rule (IR6)
In Pseudo transitive Rule, if X determine Y and YZ determines W, then XZ determines W.
i.e If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XY \rightarrow W$.

~~6.2 Normalization : Purpose and Concept of Normalization, forms of Normalization : 1-NF, 2-NF, 3-NF, BCNF~~

Advantages of Functional Dependency:

→ FD avoids data redundancy.
→ It helps to maintain the quality of data in the database.
→ It helps to define meanings and constraints of databases.
→ It helps you to identify bad designs.
→ It helps to find the facts regarding the database design.

6.2 Normalization: Purpose and Concept of Normalization, forms of Normalization:- 1-NF, 2-NF, 3-NF, BCN.

Concept

The process of converting complex data structure into simpler, stable data structure is called normalization.

Or

Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.

It divides larger tables into smaller tables and links them using relationships.

Purpose / Need of Normalization

- To minimize data redundancy (duplicate data).
- To minimize or avoid data modification issues
- To simplify queries.
- To reduce the need to reorganize data as modified or enhanced.

Forms of Normalization:

1. First Normalization form (1NF)
- Unique rows
- No multivalued attributes
- All relations are in 1NF

Example

ID	Name	Course
1	A	C1
1	A	C2
2	E	C3
3	M	C1
3	M	C2

The table is in 1NF bcz there is no Multivalued attribute.

2. Second Normalization form (2NF)

For a table to be in second normal form,
- It should be in First Normal form (1NF).
- It should not have partial dependency.

Example

Consider following functional dependencies in relation R(A, B, C, D)

AB -> C [A and B together determine C]
BC -> D [B and C together determine D]

In the above relation, AB is the only candidate key and there is no partial dependency, i.e any

proper subset of AB doesn't determine any non-prime attribute.

3. Third Normal form (3NF)

A table is said to be in Third Normal form when,

- It is in 2NF.
- It does not have Transitive dependency

Example

Consider a relation R(A, B, C, D, E)

A -> BC,

CD -> E,

B -> D,

E -> A,

All possible candidate keys in above relation are {A, E, CD, BC}. All attribute are on right sides of all functional dependencies are prime.

4. Boyce and Codd Normal form (BCNF).

Boyce Codd Normal form is a higher version of Third Normal form. This form deals with certain type of anamoly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3NF
- For each functional dependency (X -> Y), X should be a super key.

Example
Consider a relation R(A, B, C)
A -> BC,
B ->
A and B both are super keys so above relation is in BCNF.

3. Decomposition

Decomposition of relation is done when a relation in relational model is not in appropriate normal form.

Decomposition in DBMS removes redundancy, anamolies and inconsistencies from a database by diving the table into multiple tables.

Types

(6.3) a) Lossless decomposition

If we decompose a relation R into relations R_1 and R_2, decomposition is lossless, if $R_1 \bowtie R_2 = R$

(b) Lossy decomposition

If we decompose a relation R into relations R_1 and R_2, decomposition is lossy if $R_1 \bowtie R_2 \supset R$

Unit-7

Database Security and Indexing

Ajanta

Page No.
Date

Database security refers to the collective measure used to protect and secure a database or database management software from illegitimate use and malicious threats and attacks.

Database security covers and enforces security on all aspects and components of databases. They includes

- Data stored in database
- Database server
- DBMS
- Other database workflow applications

Some of the ways ways to analyze and implement database security are:

i) Restricting unauthorized access
ii) Isolate sensitive databases
iii) Respond to suspicious behaviour
iv) Use web application and database firewalls
v) Encrypt your data
vi) Audit and moniter database activity.

7.1 Authentication vs Authorization, Classification of DB Security, Levels of DB Security

Authentication vs Authorization

	Authentication	Authorization
1.	It is the process of verifying who you are.	It is the process to confirm what you are authorized to perform.
2	It confirms your identity to grant access to the system.	It determines whether you are authorized to access the resources.
3.	It determines whether user is what he claims to be.	It determines what user can and cannot access.
4	Authentication is the first step of authorization so always comes first.	Authorization is done after succesful authentication.
5.	It requires login and password. i.e Authentication = login + password (who you are)	It requires permission i.e Authorization = permission (what you are allowed to do).

Page No.
Date

Classification of DB Security

1. Access Control:
The purpose of access control must always be clear. It is applied to know situations, to known standards, to achieve known purposes. Control always has to be appropriate to the situation.

2 Auditing
Database auditing is observing a database as to be aware at the actions of database users.

3. Authentication
The client has to establish the identity of the server and the server has to establish the identity of the client. In client-server systems where data is distributed, the authentication may be acceptable from a peer system.
Authentication involves verifying the validity as at least one form of identification.

4 Encryption
Encryption is the process of encoding the message or information in such a way that only authorized parties can access it and those who are not authorized cannot access it.

5. Backup
It is the process of backing up or copying data

into an archieve file of computer so it may be used to restore the original data after a data loss event.

Levels of DB Security:

1. Physical
   The sites containing the computer systems must be secured against armed or surreptitious entry by intruders.

2. Human
   Users must be authorized carefully to reduce the chance of any such user giving access to an intruder in exchange for a bribe or other favors.

3. Operating System
   No matter how secure the database system is, weakness in OS security may serve as a means of unauthorized access to the database.

4. Network
   Software-level security within the network software is as important as physical security, both on the Internet and in networks private to an enterprise.

5. Database System
It is responsibility of the database system to ensure that these authorization restrictions are not violated.

7.2 Types of Authorization, Creating users, Granting and Revoking Authorizations in SQL, Concept of Roles, Authorization using Roles

Types of Authorization

1. Resource authorization
Authorization to access any system resource
eg sharing of database, printer etc

2. Alternation Authorization
Authorization to add attributes or delete attributes from relations.

3. Drop Authorization
Authorization to drop a relation

Creating User

CREATE USER statement is used to Create a user.

Syntax

CREATE USER Ram IDENTIFIED By Password

Example

CREATE USER Ram IDENTIFIED By *********

Granting of privileges

A system privilege is the right to perform an action on any schema object of a particular type. An authorized user may pass on this authorization to other users. This process is called granting of privileges.

OR

Granting of privileges is giving the access or privileges to the user of the database.

Syntax

GRANT <privilege_name>
ON <relation name or view name>
To <user/role name>;

Example

GRANT SELECT ON employee TO user1;

Revoking of Privileges

Revoking of privileges is taking away the granted privileges (authorizations) from the users of the database.

Syntax

REVOKE <privilege_name>
ON <relation name or view name>
FROM <user/role name>;

Example

REVOKE SELECT ON employee FROM user1;

Concept of Roles

Roles are a collection of privileges or access rights. When there are many users in database, it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users.

Creating Roles

Syntax

CREATE ROLE Role_name
[IDENTIFIED BY Password];

Example

CREATE ROLE developer
[IDENTIFIED BY pwd];

Authorization using Roles

Eg: CREATE ROLE buyer AUTHORIZATION Ram;
GRANT buyer to Shyam.

7.3 Concept of Indexing, Index file vs Data File, Index key structure, Types of Indices

Concept of Indexing

Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done.

Indexing in database systems is similar to what we see on books.

An index or database index is a data structure which is used to quickly locate and access the data in a database table.

~~Structure of index~~

Search Key	Data Reference

Fig: Structure of Index

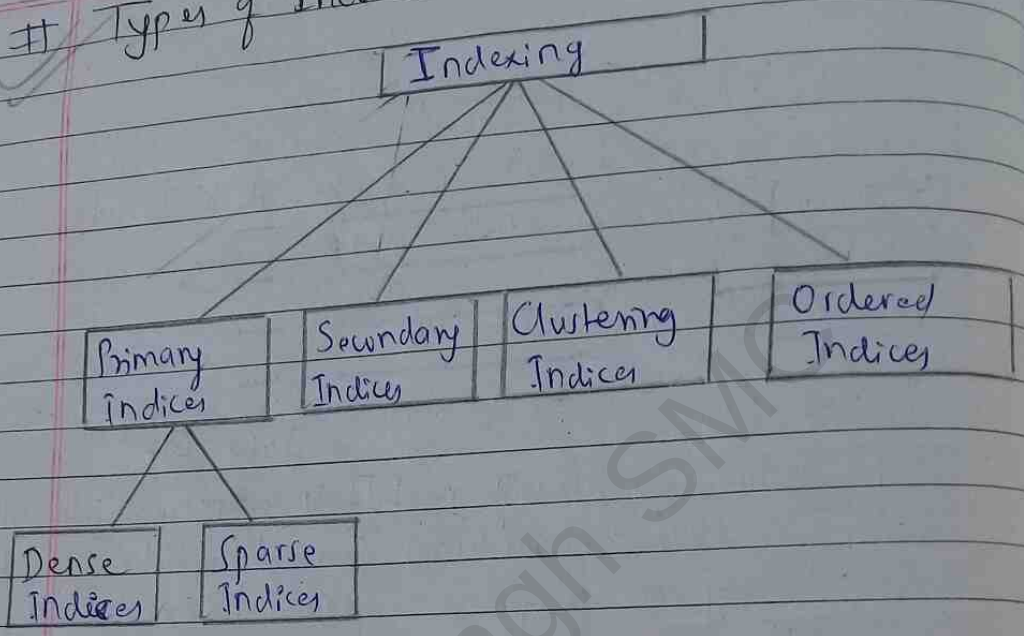
- ① First column is search key that contains a copy of the primary key or candidate key of the table.

② Second column is Data Reference which contains a set of pointers holding the address of the disk block where that particular key can be found.

Index file vs Data file

	Index file	Data file
1.	It is a computer file with an index that allows easy random access to any record given its file key.	1. It is a computer file which stores data to be used by a computer application or system.
2.		2.

Types of Indices



7.4

1. Primary Indices:
   - If the index is created on the basis of the primary key of the table, then it is known as primary indexing.
   - These primary keys are unique to each record and contain 1:1 relation between the records.

   Primary indexes can be classified into two types:

a) Dense Indices:

The dense index contains an index record for every search key in the data file. It makes searching faster.

- The number of records in the index is same as the number of records in the main table.
- It needs more space to store index record itself.

Example

Ram	→	Ram	Ktm	98
Shyam	→	Shyam	Brt	97
Hari	→	Hari	Ith	95

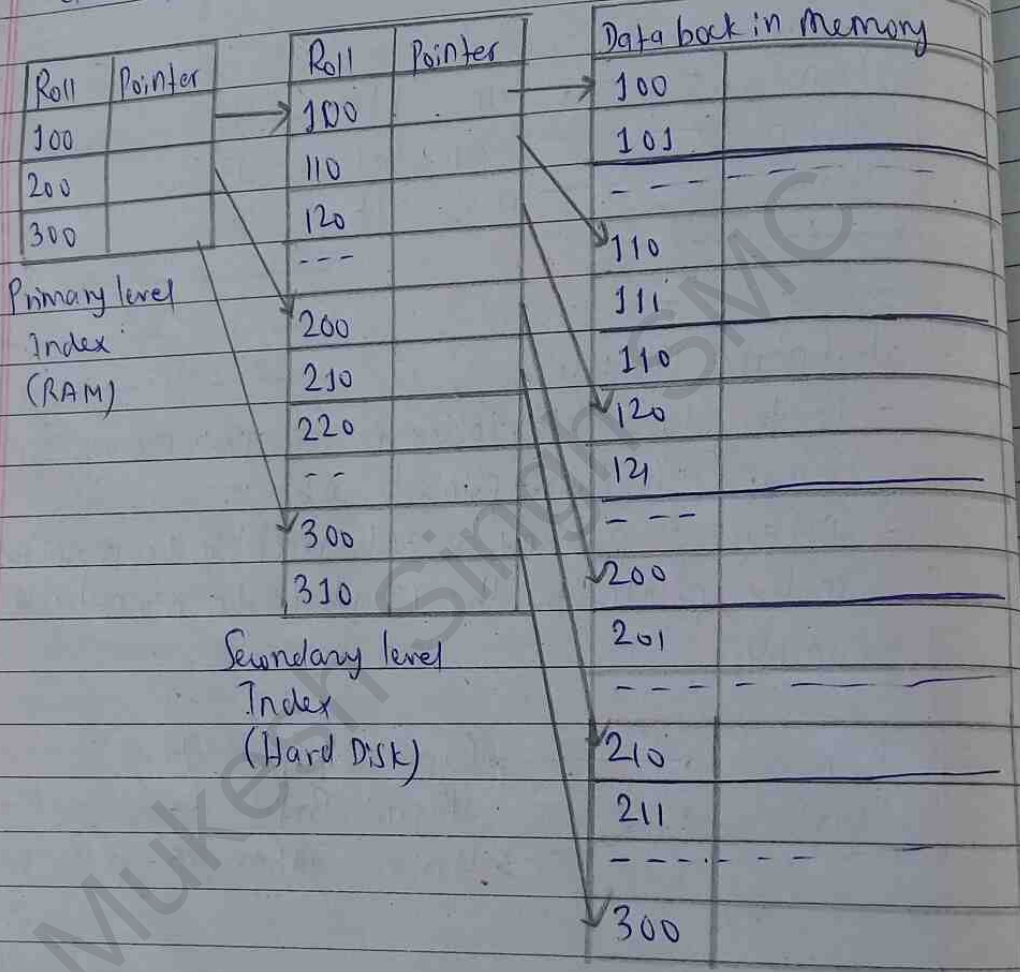
b) Sparse Indices
- In the data file, index record appears only for a few items. Each item points to a block.
- Instead of pointing to each record in the main table, the index points to the records in the main table in a gap.

Example

Ram	→	Ram	Ktm	98
Hari	→	Shyam	Brt	97
	→	Hari	Ith	95

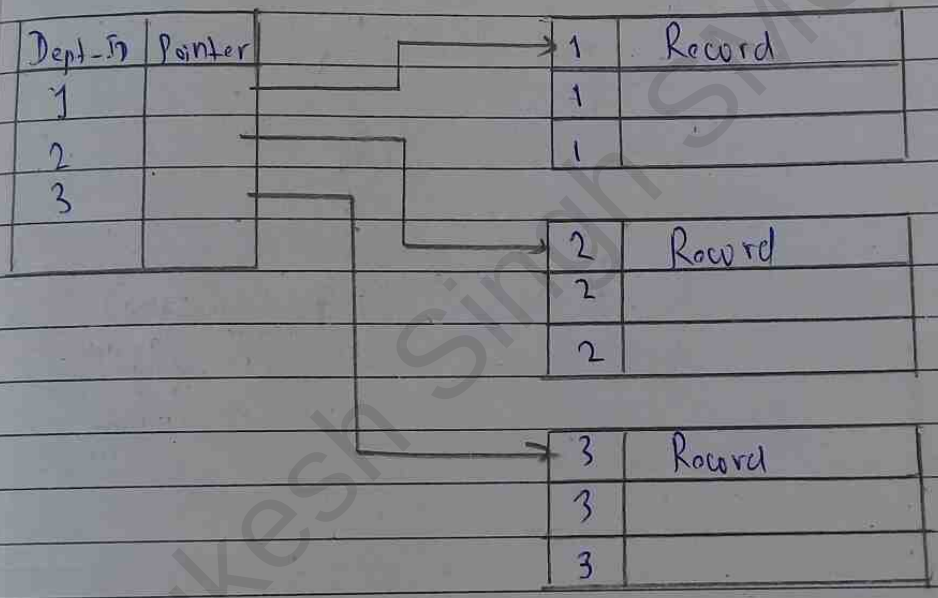
2. Secondary Indices
- Two levels of indexing are used in order to reduce the mapping size of the first level and in general.
- The huge range for the columns is selected so initially so that the mapping size of the first level becomes small.
- Then each range is divided into small ranges.

- Mapping of first level is stored in primary memory whereas mapping of second level and actual data are stored in the secondary memory.



3. Clustering Indices
   - A clustered index can be defined as an ordered data file.
   - The data file is ordered on a non-key field.

- Sometimes, the index is created on non-primary key columns which may not be unique for each record.
- In order to identify the records faster, we will group two or more columns together to get the unique values and create index out of them. This method is known as clustering index.
- Basically, records with similar characteristics are grouped together and indices are created for these groups.



4. Ordered Indices

The indices are usually sorted to make searching faster. The indices which are ~~sorted~~ sorted are known as ordered indices.

Unit-8

Introduction to Transaction & Recovery

8.1 Transaction Processing: Transaction Concepts, Transaction Operations, Desirable Properties of Transaction, Transaction States, Schedule, Serial, Non-Serial and Non Serializable Schedule

Transaction Concepts:

A transaction is a single logical unit of work which accesses and possibly modifies the contents of a database.

Transactions access data ~~during~~ using read and write operations. In order to main consistency in a database, before and after transaction, certain properties are followed. These properties are called ACID properties:

Properties of Transaction

A i) 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.

C ii) Consistency → The database must remain in consistent state after any transaction.

D iii) Durability → The database should be durable enough to hold all the latest updates even if the system fails or restarts.

I iv) Isolation → No transaction will affect the existence of any other transaction.

Transaction Operations

i) Read Operation
To read a database object, it is first brought into the main memory from disk, and then its value is copied into a program variable.

ii) Write Operation
To write a database object, an in-memory copy of the object is first modified and then written to disk.

Transaction States:

i) Active → In this state, the transaction is being executed. It is the initial state of every transaction.

ii) Partially Committed → When a transaction executes its final operation, it is said to be in a partially committed state.
OR
After the final statement has been executed.

iii) failed → When the normal execution can no longer proceed.

iv) Aborted → After the transaction has been rolled

back and the database has been restored to its state prior to the start of the transaction.

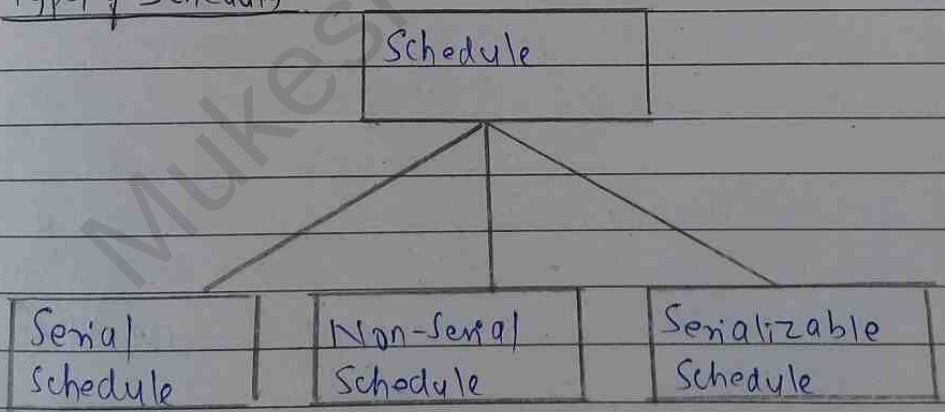
v) Committed → If a transaction executes all its operations succesfully, it is said to be committed.
OR
After succesful completion.

Schedule

A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each ~~comprising~~ comprising of a number of instructions / tasks. OR

A series of operation from one transaction to another transaction is known as schedule. It is used to preserve the order of the operation in each of the individual transaction.

Types of Schedule



1. Serial Schedule

In this schedule, when the first transaction completes its cycle, then the next transaction is executed.

T_1	T_2
R(A)	
A=A+50	
W(A)	R(A)
A=150	A=100
	W(A)

A = 150 + 100
= 250

2. Non-Serial Schedule

If interleaving of operations is allowed, there will be non-serial schedule.

Non-serial schedule interleave the action of other transactions.

for eg:

T_1	T_2
R(A)	
A=A+50	
	R(A)
W(A)	A=A+100
	W(A)

3. Serializable Schedule

A non-serial schedule will be serializable if its result is equal to the result of its transaction executed serially.

8.2 Concurrency Control: Introduction, Need of Concurrency Control, Lock-based protocols

Concurrency Control:

Concurrency control is the procedure in DBMS for managing simultaneous operations without conflicting with each other.

The multiple transactions can be executed simultaneously. It may affect the transaction result.

Problems of Concurrency Control
1) Lost Updates
2) Dirty read
3) Unrepeatable read

Need of Concurrency Control
i) To update data correctly when multiple transactions are executed concurrently.
ii) To apply isolation through mutual exclusion between conflicting transactions.
iii) To resolve read-write and write-write conflict issues.
iv) To preserve database consistency
v) To ensure serializability

Concurrency Control Protocols

Lock-Based Protocols

A lock is a data variable which is associated with a data item. Locks help synchronize access to the database item by concurrent transactions.

Database systems are equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock.

Lock are of two kinds:

i) Binary locks:→ A lock on a data item can be in two states; it is either locked or unlocked.

ii) Shared/exclusive → This type of locking mechanism separates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is called an exclusive lock.

8.3 Database Recovery: Need of Recovery, Concept of Recovery, Log Based Recovery, Write Ahead Logging, Checkpointing

Concept of Recovery:

Database recovery is the method of restoring the database to its correct state in the event of failure at the time of the transaction or after the end of a process.

Or

It is the process of restoring the data that has been lost accidentally, deleted, corrupted or made inaccessible.

Need of Database Recovery

→ To bring the database into the last consistance-state which existed prior to the failure.

→ To preserve transaction properties (ACID properties).

→

Lock Based Recovery

Log is a sequence of records, which maintains the records of actions performed by a transaction. It is important that the logs are written prior to the actual modification and stored on a stable storage media, which is failsafe.

Log-based recovery works as follows.

- The log file is kept on a stable storage media.
- When a transaction enters the system and starts execution, it writes a log about it.
  $\langle T_n, \text{Start} \rangle$
- When the transaction modifies an item X, it writes logs as follows.
  $\langle T_n, X, V_1, V_2 \rangle$
- It reads T_n has changed the value of X, from V_1 to V_2.
- When the transaction finishes, it logs -
  $\langle T_n, \text{Commit} \rangle$
- If any operation is performed on the database, then it will be recorded in the log.

Write Ahead Logging

Write Ahead Logging (WAL) is a commonly used technique in database systems to maintain atomicity and durability of writes.

The changes are first recorded in the log, which must be written to stable storage, before the changes are written to the database.

In a system using WAL, all modifications are written to a log before they are applied. Usually both redo and undo information is stored in the log.

Checkpointing

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~~ 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.

Unit-9

Emerging Database Technology & Application

9.1. Concept of Big data

Big data is a phrase used to mean a massive volume of both structured and unstructured data that is so large it is difficult to process using traditional database and software techniques.

Big data is an umbrella term for datasets that cannot reasonably be handled by traditional computers or tools due to their volume, velocity and variety.

9.2 Concept of NoSQL

NoSQL is a non-relational database that stores and accesses data using key-values.

A NoSQL database does not require a structured schema that defines each table and the related columns. This provides a much more flexible approach to storing data than a relational database.

9.3. Concept of mobile and multimedia data

Mobile data: A mobile data is a database that decides on a mobile device such as a PDA, Smart phone or Laptop. Such devices are often limited in resources such as memory, computing power and battery power.

Multimedia data

The multimedia database is a collection of related multimedia data. The multimedia data include one or more primary media data types such as text, images, graphics, animation sequences, audio and video etc.

9.4 Concept of Gis database

Gis stands for Geographical Information System.

A GIS is a system designed to capture, store, manipulate, analyze, manage and present spatial or geographical data.

9.5 Concept of data warehouse and data mining

A data warehouse is the collection of database that work together. Distributed databases are used to store a databases at multiple computer sites to improve data access and processing.

Data mining is the process of analyzing data and summarizing it to produce useful information.