Overview of

DATABASE MANAGEMENT SYSTEM

Er. Anish Baral
Er. Mukesh Kumar Pokhrel

Table of Contents

CHAPTER 1

INTRODUCTION

Database

A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populates with data for a certain task.

Database Management System (DBMS)

It is a software for storing and retrieving user's data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data.

Applications of DBMS:

1. **Telecom:** There is a database to keep track of the information regarding calls made, network usage, customer details etc. without the database systems it is hard to maintain that huge amount of data that keeps updating every milli second.
2. **Industry:** Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database t keep the records of ins and out.
3. **Sales:** To share customer information, production information and invoice details.
4. **Airlines:** To travel through airlines, we make early reservation, this reservation information and invoice details.
5. **Human resources:** To keep database of employee records, salaries and tax deductions.

Objectives

1. A database should provide for efficient storage, update and retrieval of data.
2. A database should be reliable-the stored data should have high integrity and promote user trust in that data.
3. A database should be adaptable and scalable to new and unforeseen requirements applications.
4. A database should identify the existence of common data and avoid duplicate recording selective redundancy is sometimes allowed to improve performance of for better reliability.

History of DBMS

1960 - Charles Bachman designed first DBMS system.

1970 - Codd introduced IBM's information Management system.

1976 - Peter coined and desired the Entity-relationship model also known a the ER-model.

1980 - Relational model becomes a widely accepted database component.

1985 - Objective oriented DBMS develops.

1990 - Incorporation of objective-orientation in relational DBMS.

1991 - Microsoft ships Ms access, a personal DBMS and that displaces al other personal DBMS product.

1995 - First Internet database applications.

1997 - XML applied to database processing. Many renders begin to integrat XML into DBMS products.

Characteristics of DBMS:

- Provide security and removes redundancy.
- Self-describing nature of a database system.
- Insulation between programs and data abstraction.
- Support of multiple views of the data.
- Sharing of data and multiuser transaction processing.
- DBMS allows entities and relations among them to form tables.

DBMS vs Flat File:

DBMS	Flat File management system
• Multi user Access	• It does not support multi-user access.
• Design to fulfill the need for small and large business.	• It is only limited to smaller DBMS system.
• Removes redundancy and integrity.	• Redundancy and integrity issues.
• Expensive. But in the long term Total cost of ownership is cheap.	• It's cheaper.
• Easy to implement complicated transactions.	• No support for complicated transactions.

Data abstraction in DBMS

Database systems are made-up of complex data structure. To case the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called abstraction.

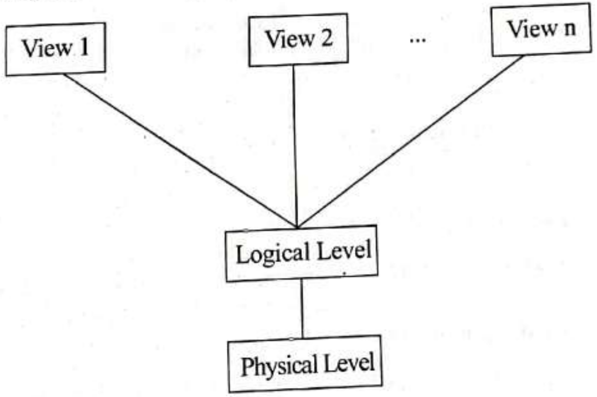


Fig: Three levels of data abstraction.

We have three level of abstraction

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

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

View Level: Highest level of data abstraction. This level describes the user interaction with database system.

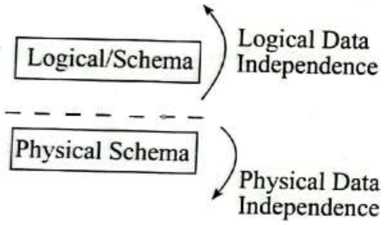
E.g: Let's say we are storing customer information in a customer table. At physical Level these records can be described as blocks of storage in memory. These details are often hidden from the programmers.

At logical level these records can be described as fields and the attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At view level, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how data is stored and what data is stored, such details are hidden from them.

Data Independence

Data Independence is defined as a property of DBMS that helps you to change database schema at one level of data base system without requiring to change the schema at the next higher level.



Logical Data Independence

Logical Data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

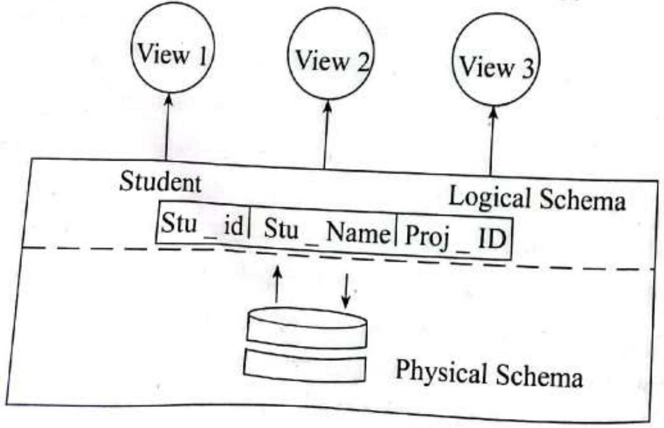
Logical data independence is a kind of mechanism which liberalizes itself from actual data stored on the disk. If we do some changes on table format it should not change the data residing on the disk.

Physical Data Independence

All the schemas are logical and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

Database schema

A database schema is the skeleton structure that represents the logical view of the database. It defines how the data is organized and how the relations among them are associated. It formulates relations among them are associated. It formulates all the constraints that are to be applied on the data.



A database schema can be divided broadly into two categories:

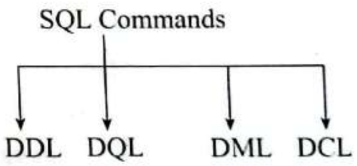
- **Physical Database schema:** This schema pertains to the actual storage of data and its form of storage like files, indices etc. It defines the data will be stored in a secondary storage.
- **Logical Database schema:** This schema define all the logical constraints that need to be applied on the data base stored. It defines tables, view, and integrity constraints.

Database Instance

It is important we distinguish these two terms individuals. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational it is very difficult to make may changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validation, constraints and conditions that the database designers have imposed.

SQL commands



Structured Query (language (SQL) as we all known is the database languages by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like create, Drop, insert etc. to carry out the required tasks.

1. **DDL - Data Definition Language**

   Consists of SQL commands that can be used to define the database schema. It simply deals with description of the database schema and is used to create and modify the structure of database objects in the database.

   E.g:

   - Create - is used to create the database or its objects.
   - Drop - is used to delete objects from the database.
   - Alter - is used to alter the structure of the database.
   - Truncate - is used to remove all records from a table, including all spaces allocated for the records are removed.
   - Comment - is used to add comments to the data dictionary.

2. **DQL (Data Query Language)**

DQL statement are used for performing queries on the data within schema objects. The purpose of DQL command is to get some schema relation based on the query passed out.

3. **DML (Data Manipulation Languages)**

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data manipulation language and this includes most of the SQL statements.

Examples of DML:

- Insert - is used to insert data into a table.
- Update - is used to update existing data with a table.
- Delete - is used to delete records from a data base table.

4. **DCL - Data Control Language**

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

Example of DCL command:

- GRANT - gives user's access privileges to database.
- REVOKE - withdraw user's access privileges given by using the GRANT command.

Old Question Solution

Differentiate between schema and instances. What are the disadvantages of conventional File system? [2076 Baisakh]

The difference between schema and instances are as follows:

Schema	Instances
1. It is the overall description of the database.	1. It is the collection of information stored in a database at a particular moment.
2. Schema is same for whole database.	2. Data instances can be changed using addition, deletion, updation.
3. Does not change frequently.	3. Change Frequently.
4. Defines the basic structure of the database.	4. It is the set of information stored at a particular time.

Disadvantages of Conventional File system

(i) **Data Redundancy**

It is possible that the same information may be duplicated in different files, this leads to data redundancy results in memory wastage.

(ii) **Data Inconsistency**

Because of data redundancy, it is possible that data may not be in consistent state.

(iii) **Difficulty in Accessing Data**

Accessing data is not convinient and efficient in file processing system.

(iv) **Integrity Problems**

Data integrity means that the data contained in the database in both correct and consistent. For this purpose the data stored in database must satisfy correct and constraints.

(v) **Security Problems**

Database should be accessible to user in limited way.

Each user should be allowed to access data concerning his requirement only.

2. **Define data independence and Explain its significance What is importance of aggregation in ER design? Discuss with an example. [2076 Baisakh]**

⇒ **Definition:** See in Page no. 4

Significance of data Independency:

- Helps you to improve the quality of the data.
- Database system maintenance becomes affordable.
- Enforcement of standards and improvement in database security.
- You don't need to alter data structure in application programs.
- Data in congruity is vastly reduced.

Aggregation: See in Page no. 21

3. **What do you mean by schema and instances? Mention the different levels of data abstraction and explain. [2075 Bhadra]**

⇒ 1st part: See in old question solution Q no 1

2nd Part:See in page No.3

4. **Define Data Abstraction. Explain its different levels with example. [2075 Baisakh]**

⇒ See in page no 3

5. **Mention the advantages of the DBMS over the file processing system and explain briefly. [2074 Baisakh]**

⇒ See in page no2

6. **Why is data independence important in data modeling? Differentiate between schema and instances. [2073 Bhadra]**

⇒ 1st part:See in old question no2

2nd part :See in old question no.1

7. **What do you mean by data abstraction. List the various levels of data abstraction and briefly explain. [2073 Magh]**

⇒ See in page no: 3

8. **Why is data independence is importance in data modeling? Differentiate between physical and logical data independence. [2072 Ashwin]**

⇒ See in page no:4

9. **What are the drawbacks of file system to store data? [2072 Magh]**

⇒ Disadvantages of Traditional File System :

- Data redundancy and inconsistency.
- Difficulty in accessing data.
- Data isolation – multiple files and formats.
- Integrity problems
- Unauthorized access is not restricted.
- It co-ordinates only physical access

10. **What difficulties would you face if you used file system directly to implement a database application? What is physical data independence? [2071 Bhadra]**

⇒ 1st part: Old question no.9

2nd part: See in page no.4

11. **Distinguish between a database and a DBMS. What is the advantages of separating the logical level and physical level in database design? [2071 Magh]**

⇒ **Database**

- The database includes the actual data you save.
- The data base is any collection of data whenever you are writing it on the paper or storing it in the digital format.

There are types of database:

- Online transaction processing (OLTP)
- Online Analysis processing (OLAP)

DBMS

- DBMS stands for database Management system.
- DBMS is a kind of software that helps you to retrieve, edit, and store structured data in the database.
- Data Independence is the property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.
- Two levels of data independence are 1) Physical and 2) Logical
- Physical data independence helps you to separate conceptual levels from the internal/physical levels
- Logical Data Independence is the ability to change the conceptual scheme without changing
- When compared to Physical Data independence, it is challenging to achieve logical data independence
- Data Independence Helps you to improve the quality of the data

12. **Briefly explain different levels of data abstraction in a database system.** **[2070 Bhadra]**

⇒ See in page no. 3.

13. **Explain the difference between DDL,DML,DCL along with examples.** **[2070 Magh]**

⇒ See in page no. 5.

14. **Briefly highlight your significant differences between a file processing system and a DBMS.** **[2069 Bhadra]**

⇒ See in page no. 2

□□□

CHAPTER 2

DATA MODELS

Data Models

Data Models defines how the logical structure of a database is modeled. Data models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

Basically Three Types of Data Models:

(i) Conceptual Model

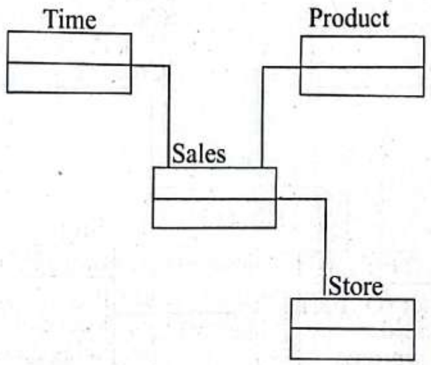
(ii) Physical Model

(iii) Logical Model

Conceptual Model

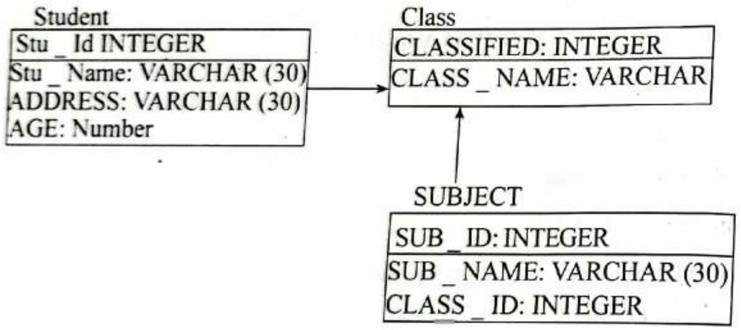
- Established the entities, their attributes and their high - level relationship.
- It is also called domain model.
- There is a little detail.
- If there is attributes defined, there are loosely typed and connectors between entities do not define relationship to specific attribute.
- It is also called entity - based or object based data models (like; ER models, 00 models)

e.g:



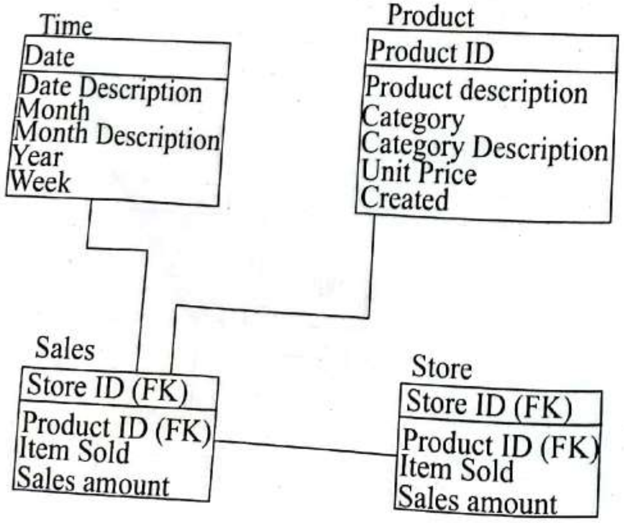
Physical Data Model

Physical data model represents the model where it describes how data are stored in computer memory, how they are scattered and ordered in the memory, and how they would be retrived from memory. Basically physical data model represents the data at data layer or internal constraints like primary key, foreign key, etc. It basically describes how each tables are built and related to each other in DB.



Logical Data Model

Logical Data Model is fully - attributed data models that is independent of DBMS technology, data storage or organizational constraints. It describes the data requirements form the business point of view.



Comparison between data models:

Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	
Entity Relationships	✓	✓	
Attributes.		✓	
Primary key		✓	✓
Foreign key		✓	✓
Table Names			✓
Column Names			✓
Column Data types			✓

E - R Model

Entity Relationship model is a high level conceptual data model diagram. ER modeling helps you to analyze data requirements systematically to produce a well - designed database. The E - R model represents real world entities and relationship between them.

ER - diagrams

E - R diagrams displays the relationship of entity set stored in database. In other words, we can say that ER diagrams help you to explain the logical structure of database. At first look, an ER diagram looks very similar to flowchart. However, ER diagram includes many specialized symbols, and its meaning make this model unique.

Purpose of ER diagrams:

- Helps you to define terms related to entity relationship modeling.
- Provides a preview of how all tables should connect, what fields are going to be on each table.
- Helps to describe entities, attributes, relationship.
- ER diagrams are translatable into relationship tables which allows you to build database quickly.
- ER diagrams allowed you to communicate with the logical structure of the database to users.

Components of ER - diagram

(i) Entities

(ii) Attributes

(iii) Relationship

E.g: For e.g: in a university database, we might have entities for students, course and lectures. Students entity can have attributes like Roll No, Name and Deptip. They might have relationship with courses and lectures.

Entity

It is real world living thing or non - living thing that is easily recognizabl An entity can be person, object, event or a concept, which stores data in th database.

Examples of Entities

Person:	Employee, Student, Patient
Place:	Store, Building
Object:	Machine, Product and Car
Event:	Sale, Registration, Renewal
Concept:	Account, Course

Entity Set

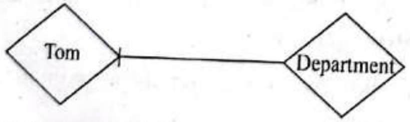
An entity set is a group of similar kind of entities. It may contain entities with attributes sharing similar values. Entities are represented by their properties which also called attributes. All attributes have their separate values. For e.g a student may have a name, age, class, as attributes.

E.g of entity set:

A university may have some departments. All these departments employ various lectures and offer several programs. Some courses make up each program. Students register in a particular program and enroll in various courses. A lecturer from the specific department takes each course, and each lecturer teaches a various group students.

Relationship

Relationship is nothing but association among two or more entities E.g: Tom works in the chemistry department.



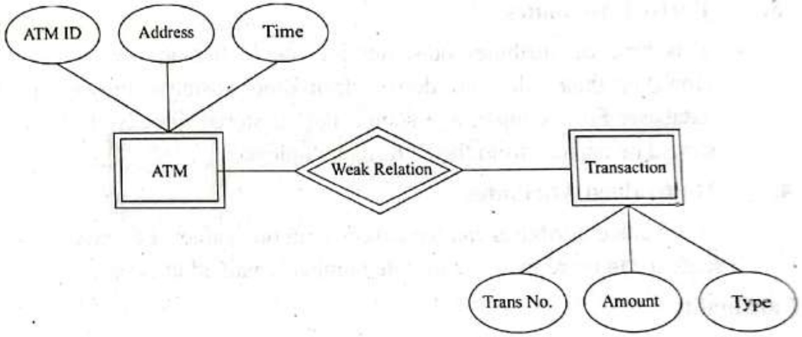
Entities take part in relationship. We often identify relationship with verbs or verb phrases.

For E.g:

- You are attending this lecture.
- I am giving the lecture.
- A student attends a lecture.
- A lecture is giving a lecture.

Weak Entities

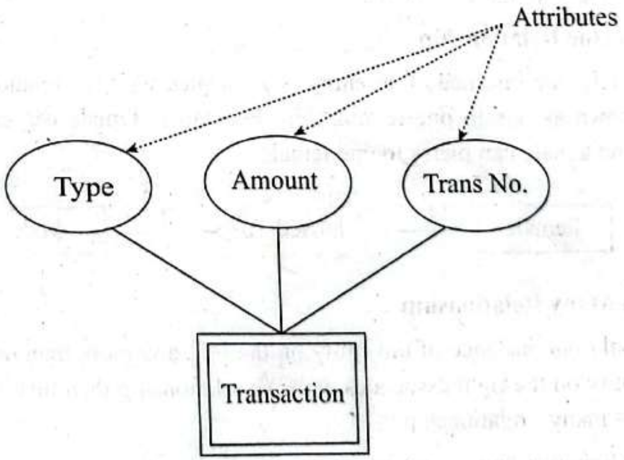
A weak entity is a type of entity which doesn't have its key attribute. It can be identified uniquely by considering the primary key of another entity. For that, weak entity sets needs to have participation.



In above example, "Trans No". is a discriminator within a group of transactions in an ATM.

Attributes

It is a single - valued property of either an entity - type or a relationship - type. For example, a lecture might have attributes: time, date, duration, place, etc. An attribute is represented by an ellipse.



Types of Attributes:

1. **Simple Attributes**

   Simple students can't be further divided. For example, a student's contact numbers. It is also called an atomic value.

2. **Composite Attributes**

It is possible to break down composite attribute. For example, a student's full name and last name divided into first name, second name and last name.

3. **Derived Attributes**

This type of attributes does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB that employee.

4. **Multivalued Attributes**

Multivalued attributes can have more than one values. For examples, a student can more than one mobile number, email address etc.

Cardinality

Defines the numerical attributes of the relationship between two entities or entity set.

Different types of cardinal relationship are:

(i) One to one relationships

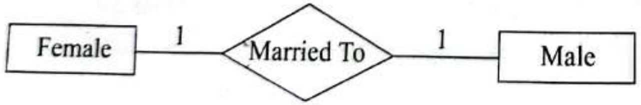
(ii) One to - many relationship

(iii) Many to one relationship

(iv) Many to many relationship

One to One Relationship

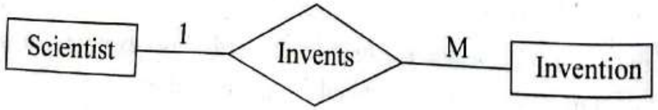
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship. For e.g: A female can carry to one male, and a male can marry to one female.



One To Many Relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as one - to - many - relationship.

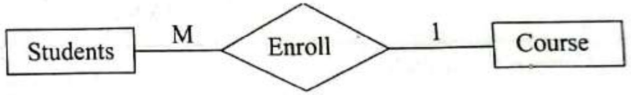
For e.g: Scientist can invent many inventions, but the invention is done by only specific scientist.



Many - To - One - Relationship

When more than one instance of the entity on the left, and only one instance of an entity of an entity on the right associates with the relationship then it is known as many - to - one relationship.

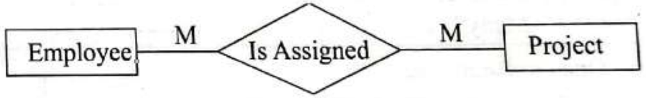
For e.g: Student enrolls that only one course, but a course can have many students.



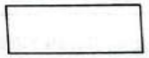
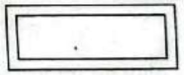
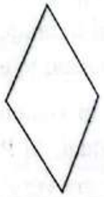
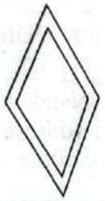
Many To Many relationship

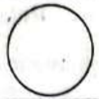
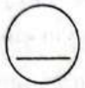
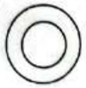
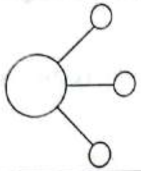
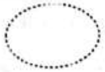
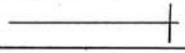
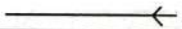
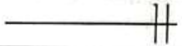
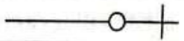
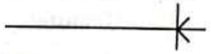
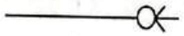
When more than one instance of the entity in the left and more than one instance of an entity on the right associates with the relationship, then it is known as many - to - many relationship.

For e.g: Employee can assign by many projects and projects can have many employees.



Notations in ER - diagrams

Meaning	Symbols
Entity	
Weak entity	
Relationship	
Weak - Relationship	

Attribute	
Key Attribute	
Multivalued Attribute	
Composite Attribute	
Derived Attribute	
One	
Many	
One (and only one)	
Zero or many	
One or many	
Zero or many	

Keys

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationship between tables.

For e.g: In student table, ID is used as a key, because it is unique for each student. In PERSON table, passport _ number, license _ number and SSN are very since they are unique for each person.

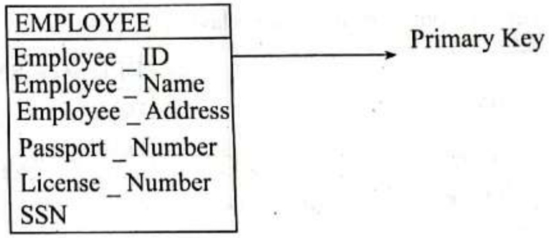
STUDENT
ID
Name
Address
Course

PERSON
Name
DOB
Passport _ Number
License _ Number
SSN

Types of Key

(i) Primary Key

- It is the key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys are we saw in PERSON table. The key which is most suitable from those lists becomes a primary key.
- In the employee table, ID can be primary key since it is unique for each employee. In the employee table, we can even select license - number and passport - number as primary since they are also unique.
- For each entity, selection of the primary key is based on requirement and developers.


EMPLOYEE
Employee _ ID
Employee _ Name
Employee _ Address
Passport _ Number
License _ Number
SSN

(ii) Candidate key

- A candidate key is an attributes or set an attribute which can uniquely indentify a tuple.
- The remaining attributes expect for primary key are considered as a candidate key. The candidate key are as strong as primary key.

  For e.g: In the employee table, id is best suited for the primary key. Rest of the attributes like SSN, passport Number and License - number etc are considered as a candidate key.

(iii) Super key

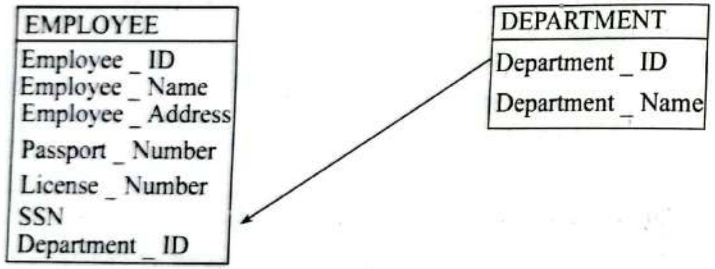
Super is a set of an attribute which can uniquely identify a tuple. Super key is a super set of a candidate key.

For e.g: In above employee table, for (employee _ ID employee _ name) the name of two employees can be the same, but their EMPLOYEE _ ID can't be the same.

The super key would EMPLOYEE _ ID (EMPLOYEE - ID, EMPLOYEE - NAME) etc.

(iv) Foreign Key

- Foreign keys are the column of the table which is used to poi to the primary key of another.
- In a company, every employee works in a specific departmen and employee and department are two different entities. So w can't store the information of the department in the employe table. That's why we link these two tables through the primar key of one table.
- We add the primary key of the DEPARTMENT tabl Department - ID as a new attributes in the EMPLOYEE table.
- Now in the EMPOYEE table, department - ID is the foreig key, and both the tables are related.



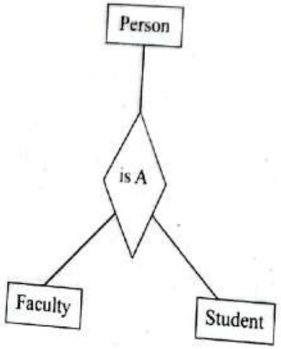
EMPLOYEE
Employee _ ID
Employee _ Name
Employee _ Address
Passport _ Number
License _ Number
SSN
Department _ ID

DEPARTMENT
Department _ ID
Department _ Name

Generalization

- Generalization is like a bottom - up approach in which two or more entities of lower level combine to form a higher level entity if they have some attribute in common.

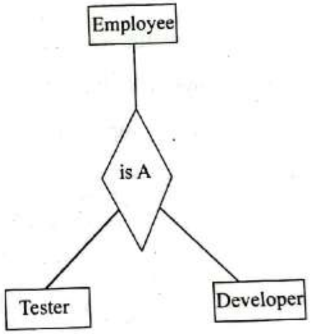
  For e.g: Faculty and student entities can be generalized and create a higher level entity person.



Specialization

Specialization is s top - down approach, and it is approach opposite to generalization. In specialization one higher level entity can be broken down into two lower level entities.

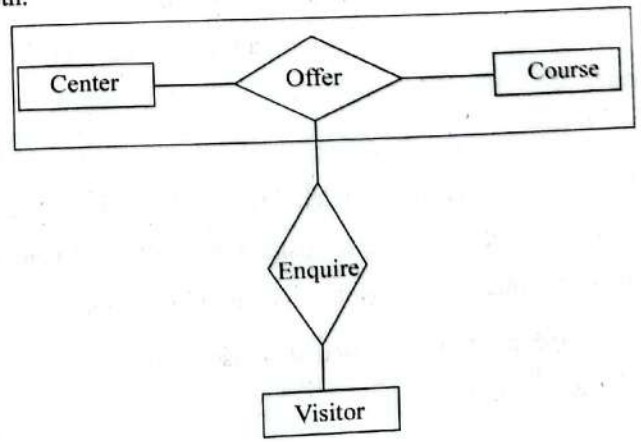
For e.g: In an employee management system EMLPOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



Aggregation

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entries is aggregated into a higher level entity.

For example: center entity offers the course entity acts as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the course only or just about the center instead he will ask the enquiry about both.

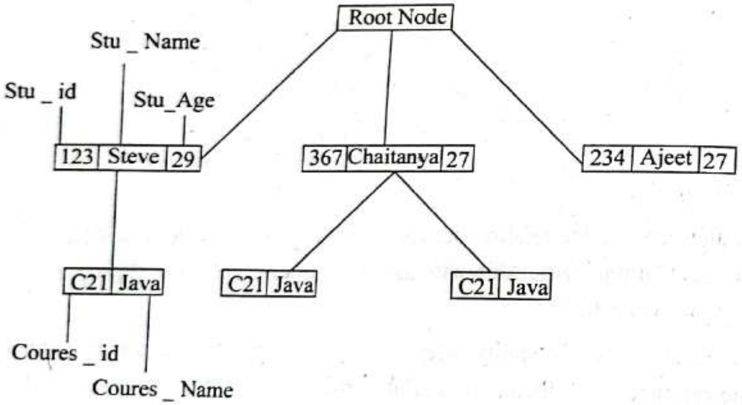


Alternate Data Models

1) Hierarchial Model

In hierarchial model, data is organized into a tree like structure with each records is having one parent record and many children. The main drawback of this model is that, it can have only one to may relationship between nodes.

e.g: Let's say we have few students and few courses and a course can be assigned to a single student only, however a student take any number of courses so this relationship becomes one to many.

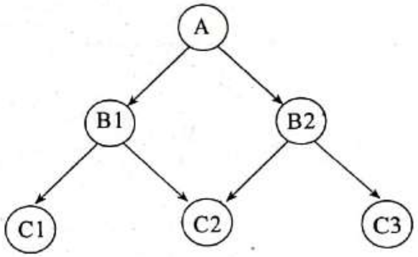


2) Network Model

This is a extension of the Hierarchial model. In this model data is organised more like a graph and are allowed to have more than one parent node.

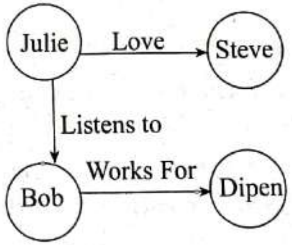
In this database model data is more related as more relationship are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used map many - to many data relationship.

This was the most widely used database model, before relational model was introduced.



3) Graph Model

A graph model, also referred to as a semantic database, is a software application designed to store, query and modify network graph. A network graph is a visual construct that consists of nodes and entity. Each mode represents an entity (such as a person) and each edge represents a connection or relationship between two nodes

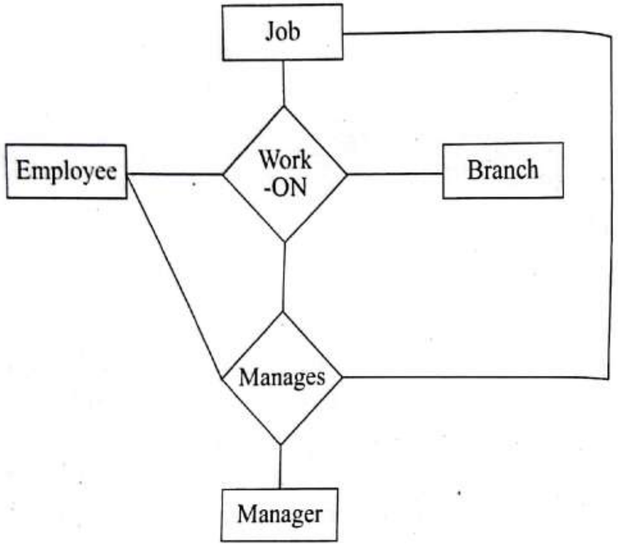


Old Question Solution

1. What is importance of aggregation in ER design? Discuss with an example. [2+2] [2076 Baisakh]

⇒ Aggregation represents relationship between a whole object and its component. Using aggregation we can express relationship among relationships. Aggregation shows 'has - a' or 'is - part - of ' relationship between entities where one represents the 'whole' and other 'part'.

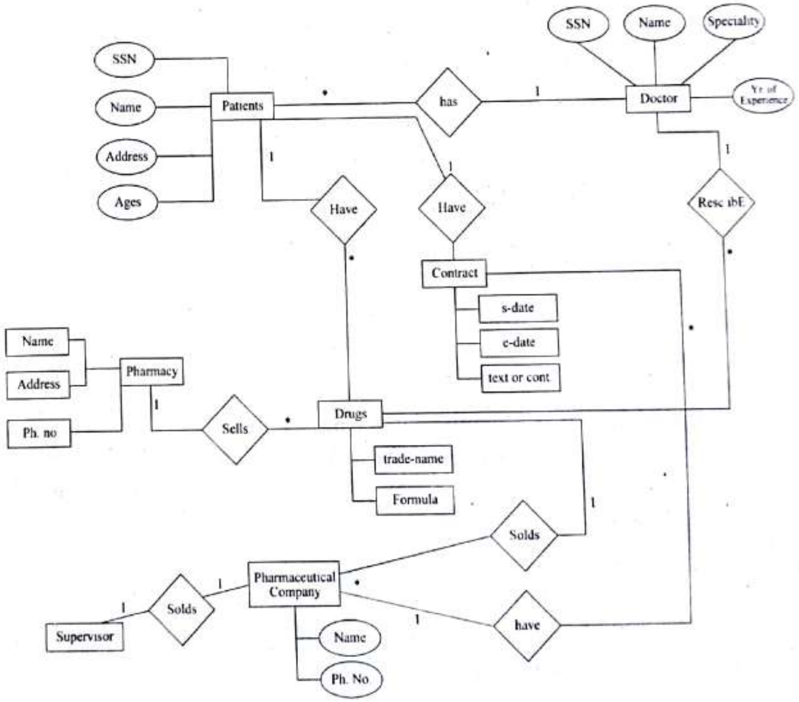
E.g. consider a ternary relationship works - on between employee, branch and manager. Now the best way to model this situation is to use aggregation. So, the relationship - set works - on is a higher level entity - set such an entity - set is treated in the same manner as any other entity - set. We can create a binary relationship manager between works - on and manager to represent who manages what tasks.



2. **Draw an E-R diagram for the given case. [8] [2076 Baisakh]**

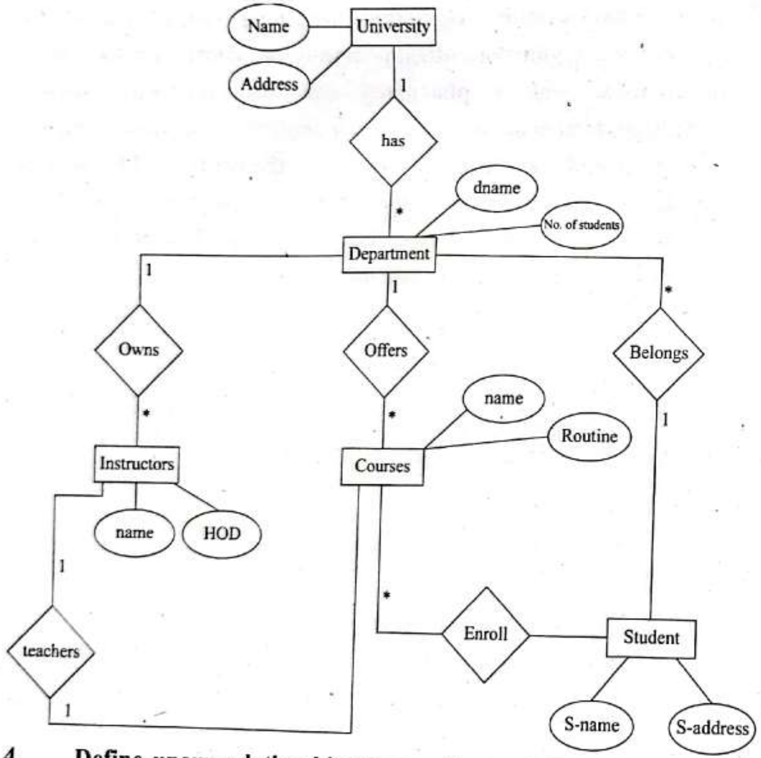
A company having a chain of pharmacies wishes you to design a database for the company. Patients are identified by as SSN, and their names, addresses, and ages must be recorded. Doctors are identified by an SSN. For each doctor, the name, specialty, and years of experience must be recorded. Each pharmaceutical company is identified by name and has a phone number. For each drug, the trade name and formula must be recorded. Each drug is sold by a given pharmaceutical company, and the trade name identifies a drug uniquely from among the products of that company. If a pharmaceutical company is deleted, you need not keep track of its products any longer. Each pharmacy has a name, address, and phone number. Every patient has a primary physician. Every doctor has at least one patient. Each pharmacy sells several drugs and has a prize for each. A drug could be sold at several pharmacies, and the price could vary from one pharmacy to another. Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and a quantity associated with it. You can assume that if a doctor prescribes the same drug for the same patient more than once, only the last such Prescription needs to be stored. Pharmaceutical companies have long-term contracts with pharmacies. A pharmaceutical company can contract with several pharmacies, and a pharmacy can contract with several pharmaceutical companies. For each contract, you have to store a start date, and end date, and the text of the contract. Pharmacies appoint a supervisor for each contract. There must always be a supervisor for each contract, but the contract supervisor can change over the lifetime of the contract.

⇒



3. **Identify relevant attributes and construct an ER diagram with proper mapping constraints for a university which has many departments and each department has multiple instructors; one among them is the head of the department. An instructor belongs to only one department, each department offers multiple courses, each of which is taught by a single instructor. A student may enroll for many courses offered by different departments.**

[6] [2075 Bhadra]



4. **Define unary relationship along with example. How you convert an ER relationship into relation schema? Explain with examples of different cardinalities.** **[2+4] [2075 Bhadra]**

⇒ **Unary relationship**

When there is only on entity set participation in a relationship then such type of relationship s called unary relationship.

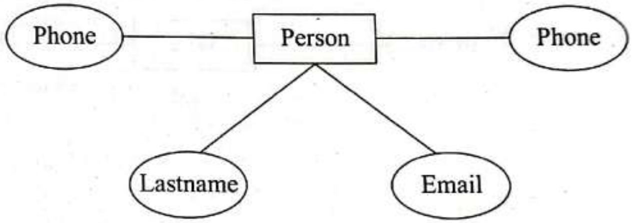
E.g. A person has only one passport and only one passport is given to only one person and hence unary relationship is observed.

Conversion of ER relationship into relation schema:

1. Entities and simple Attributes: An entity type within ER diagram is turned into a table. You may preferably keep the same name for the entity or given it a sensible name but avoid DBMS reserved words as well as avoid the use of special characters.

   Each attributes turns into a column (attribute) in the table. The key attribute of the entity is the primary key of the table which of the entity is the primary key of the table which is usually underlined. It can be composite if required but can never be null.

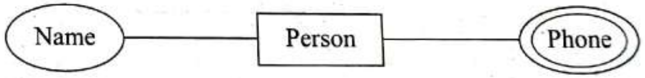
E.g.



Relational Schema

Persons (Personid, name, lastname, email)

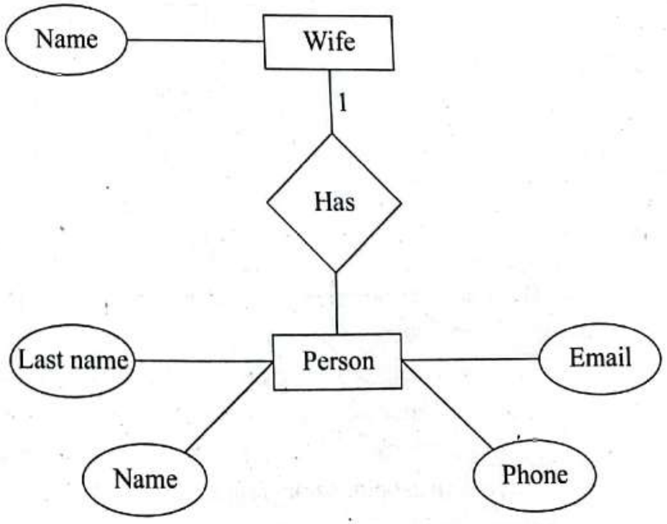
2. **Multivalued Attributes:**



If you have a multi - valued attribute, Take the attribute and turn it into a new entity or table of its own. Then make a 1 : N relationship between the new entity and the existing one. In simple words.

1. Create a table for the attribute.
2. Add the primary (id) column of the parent entity as a foreign key within the new table as shown below.

3. 1 : 1relationship

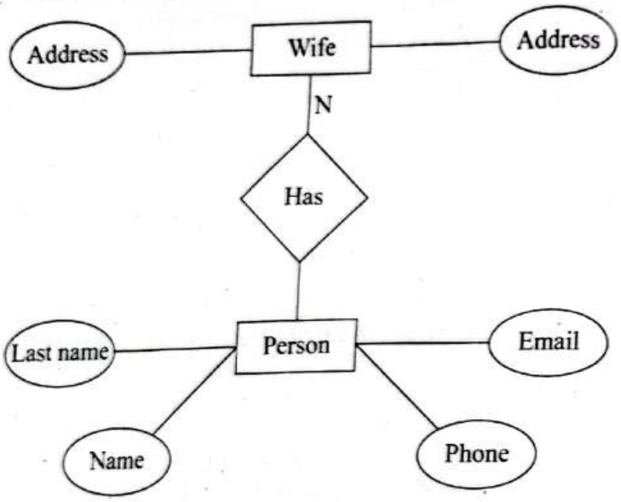


Person (personid, name, last name, email, wifeid)

Wife (wifeid, name)

Here, wifeid is foreign key.

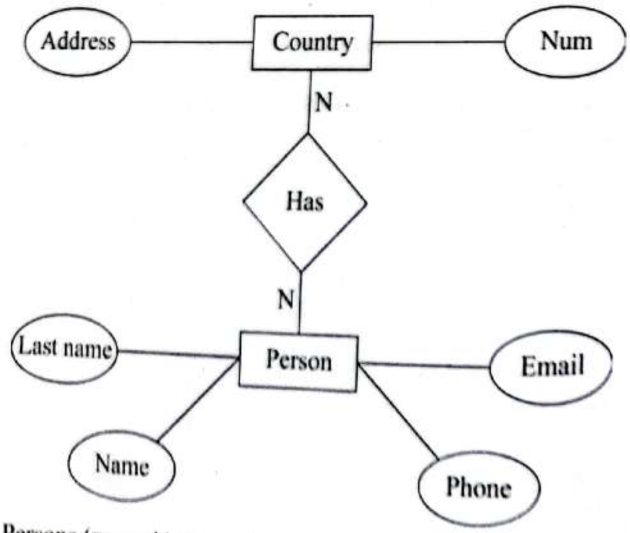
4. 1 : N relationships



Persons (personid, name, lastname, email)

House (housed, num, address, personid)

N : N relationship



Persons (personid, name, lastname, email)

ntries (country, name, code)

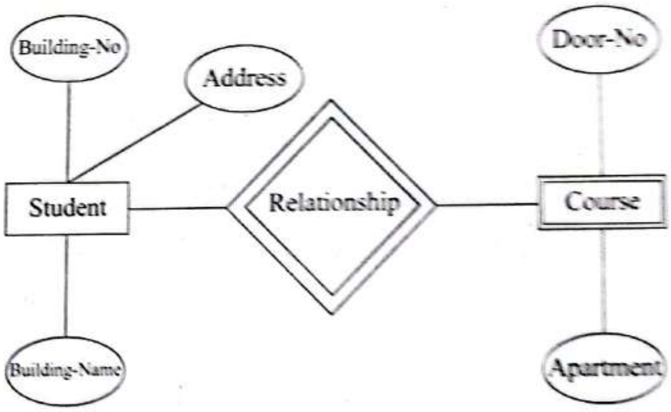
lat (has related, personid, country id)

5. **Explain strong and weak entity sets along with example.** **[2075 Baisakh]**

⇒ (a) **Strong Entity:** A strong entity is not dependent of any other entity in schema. A strong entity will always have primary key. Strong entities are represented by a single rectangle.

(b) **Weak Entity:** A weak entity is dependent on a strong entity to ensure the its existence. Unlike a strong entity, a weak entity does not have any primary key.

Example:



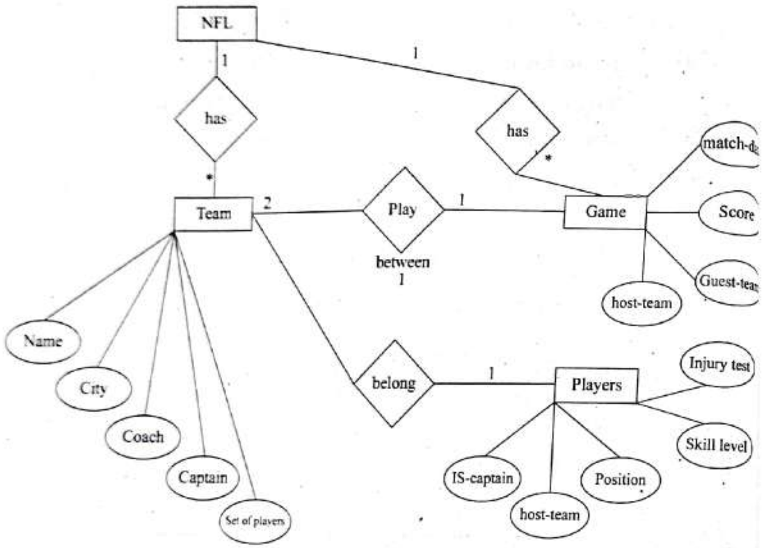
Building is a strong entity because it has a primary key a tribute called building number.

Apartment is weak entity because it does not have any primary key and door number here acts only as a discriminator.

6. **Construct an ER-Diagram for the following NFL database.[2075 Baisakh]**

You are given the requirement for a simple database for the National Football League (NFL). The NFL has many teams, and each team has a name, a city, a coach, a captain and a set of players. Each player belongs to only one team and each player has a came, a position (such as left wing, mid fielder or a goalkeeper) a skill level, and a set of inquiry records. A team captain is also a player and a game is played between two teams (referred as host team and guest team) and has a match date (such as June 11, 2018) and score (such as 2 to 5).

⇒

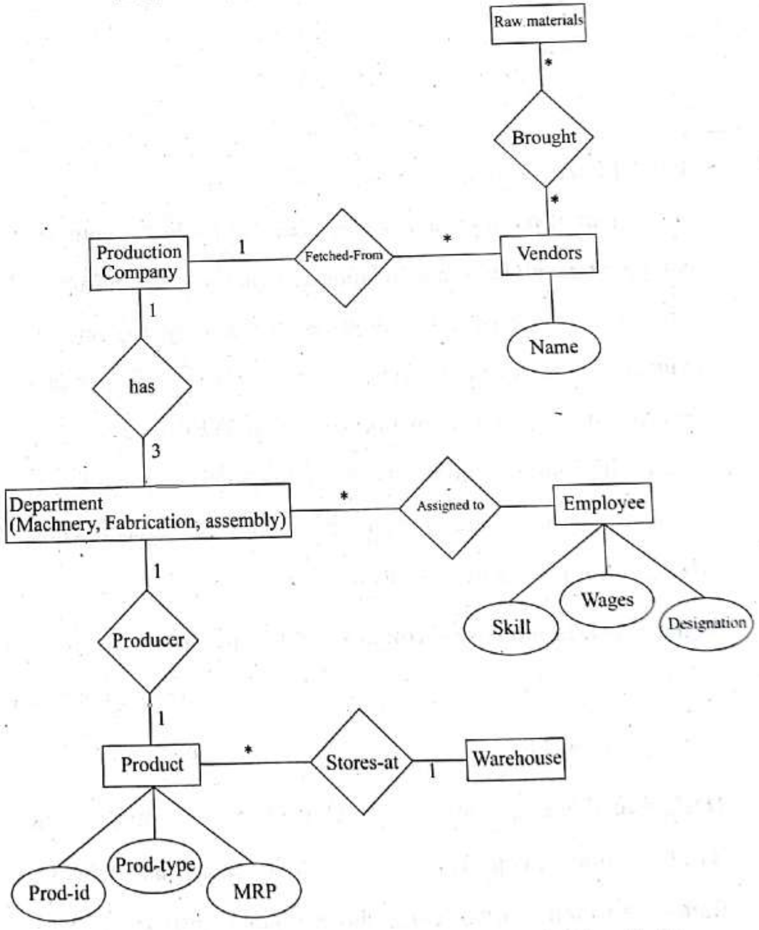


7. **Define discriminator in ER diagram. Explain different keys used in database design.** **[4] [2074 Bhadra]**

⇒ The discriminator of a weak entity set on is a set of a attributes that allows the distinction be made. For e.g. payment number acts as discrimination for payment entity set.2^{nd} **part see on page no.19**

8. **Draw the Entity-Relationship Diagram (ERD) with appropriate mapping cardinalities for the following scenario**

A production company consists of a machining, fabrication and assembly department. Employees are assigned to different departments. Each department is managed by a manager. Each employee has at most one recognized skill, but a given skill may be possessed by several employee is able to operate a given machine-type (e.g. lathe, grinder, welding) of each department. Some of the employees are paid overtime and some of them are paid with daily basis. According to their designation (e.g. mechanic, welder) are supposed to maintain at least one machine-type of their department. Raw materials are bought from different vendors and fetched to the machining department. Parts from machining department are fetched to fabrication department and so on. Many parts are assembled together to form a product. The final products from assembly department are stored in the ware house. Products are labeled with different specification (e.g., Product_Id, Product_type, MRP, etc.) **[8] [2074 Bhadra]**

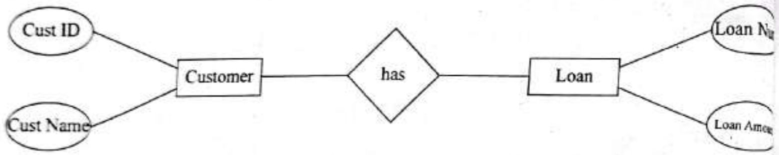


9. **Differentiate total and partial participation with suitable example.** **[2073 Bhadra]**

⇒ **Total Participation:**

Total participation is when each entity in the entity set occurs in at least one relationship in that relationship set.

For instance, consider the relationship borrower betwe customers and loans. A double line from loan to borrower, as show in figure below indicates that each loan must have at least o associated customer.



Partial Participation:

Partial participation is when each entity in the entity set ma not occur in at least one relationship in that relationship set. Fo instance, if a company policy states that employee (manager) mus manage a department, However every employee may not manage department, so the participation of EMPLOYEE in the MANAGE relationship type is partial, meaning that some or part of the set o employee entities are related to some department entity vi MANAGES, but not necessarily all.

10. What are data models? Explain various types of data models.

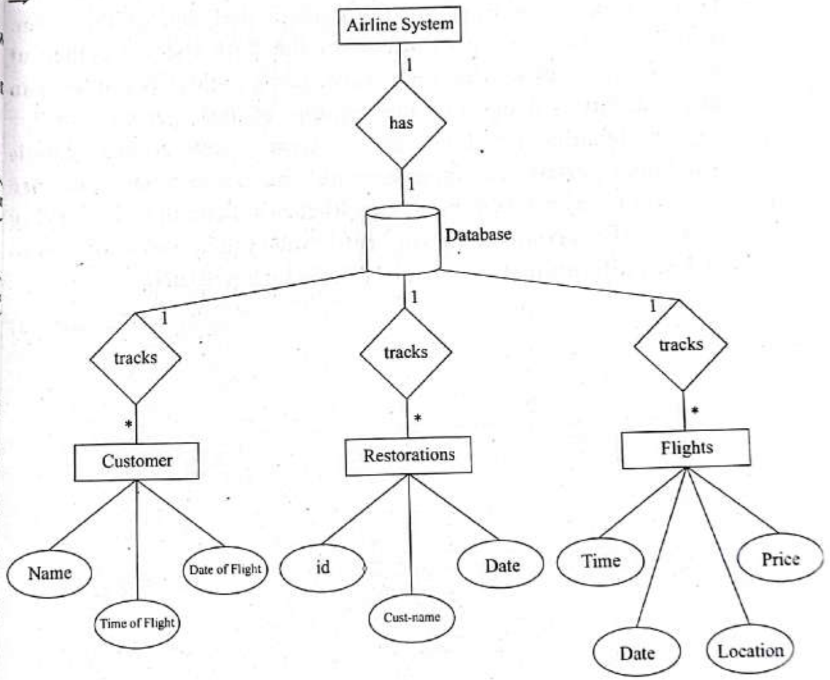
[2073 Magh]

⇒ See in page No:11

11. Design an E-R diagram for a database for an airlines system. The database must keep track of customers and their reservations, flights and their status, seat assignments on individual flights and the schedule and routing of future flights. Apply all the database design constraints as much as possible.

[8] [2073 Magh]

⇒



12. What is the difference between strong and weak entity sets?

[4] [2072 Ashwin]

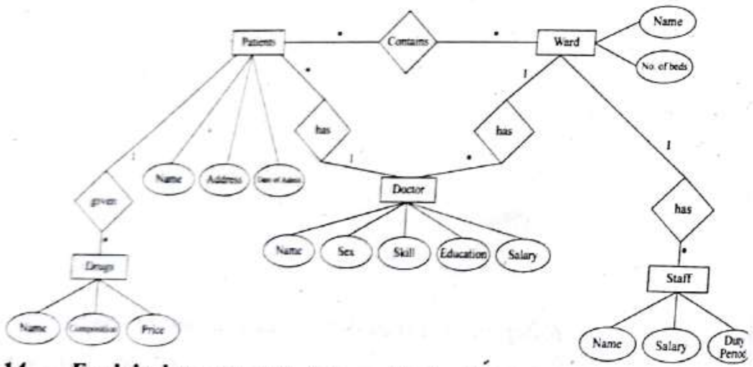
⇒ The differences between strong entity and weak entity are as follows:

Strong Entity	Weak Entity
1. Strong entity always have one primary key	1. Weak entity have a foreign key referencing primary of strong entity.
2. Strong entity is independent of other entities.	2. Weak entity is dependent on strong entity.
3. A strong entity is represented by single rectangle.	3. A weak entity is represented by double rectangle.
4. Relationship between two strong entities is represent by single diamond.	4. Relationship between a strong and weak entity is represented by double diamond.

13. **Draw an ER-diagram for the following mini-case. Patients a treated in a single ward by the doctors assigned to the the Healthcare assistants also attend to the patients; a number these are associated with each ward. Each patient is required take a variety of drugs a certain number of times per day and f varying lengths of time. The system must record deta concerning patient treatment and staff payment. Some staffs a paid part time and doctors and healthcare assistants work varyi amounts of overtime at varying rates, the system will also need track what treatment are required for which patients.**

[2072 Ashwi

⇒



14. **Explain how network data model is different from relation dat model.**

[4] [2072 Magh

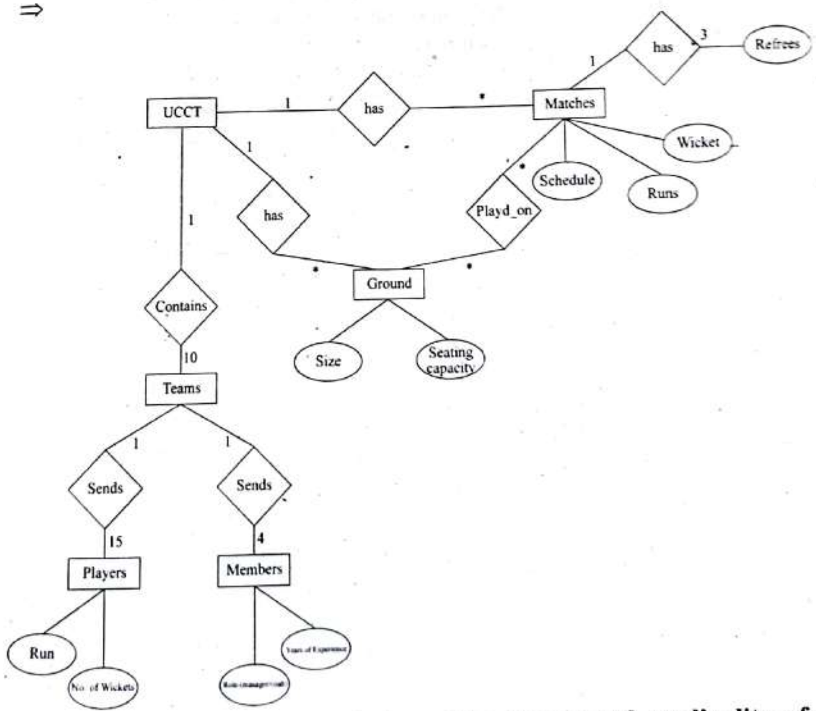
⇒ The distance between network data model and relational data mode are as follows:

Network Data Model	Relational Data Model
1. It organizes records to one another through links or pointers.	1. It organizes records in the form of table and relationship between tables are set using.
2. It organizes records in form of directed graphs.	2. It organizes records in form of tables.
3. Retrieval algorithms are complex but symmetric.	3. Retrieval algorithm are simple and symmetric.
4. There is partial data independence in this model.	4. This model provides data indepence.

15. **An information system is to be designed for keeping the records of Universe Cup Cricket Tournament. There are 10 teams practicing in the tournament. Each country sends 15 players and 4 other members. For players, the runs he scores and the number of wicket taken (so far) are to be recorded. For non-players, the role (manager, coach etc) and the number of years of experience are recorded. There are matches scheduled among the teams on several grounds on fixed dates. Each ground has fixed seating capacity and a size. For 38 matches, 11 referees have been assigned. Each match will have 3 refries. The performance of every player in every match is to be recorded in terms of runs he scored and wicket he took. Draw E-R model of the system.**

[8] [2072 Magh]

⇒



17. **What is the difference between the degree and cardinality of a relationship?** **[8+4] [2071 Bhadra]**

⇒ Degree: This is the number of entities involved in the relationship and it is usually 2 (binary relationship) however unary and higher degree relationship can be exists.

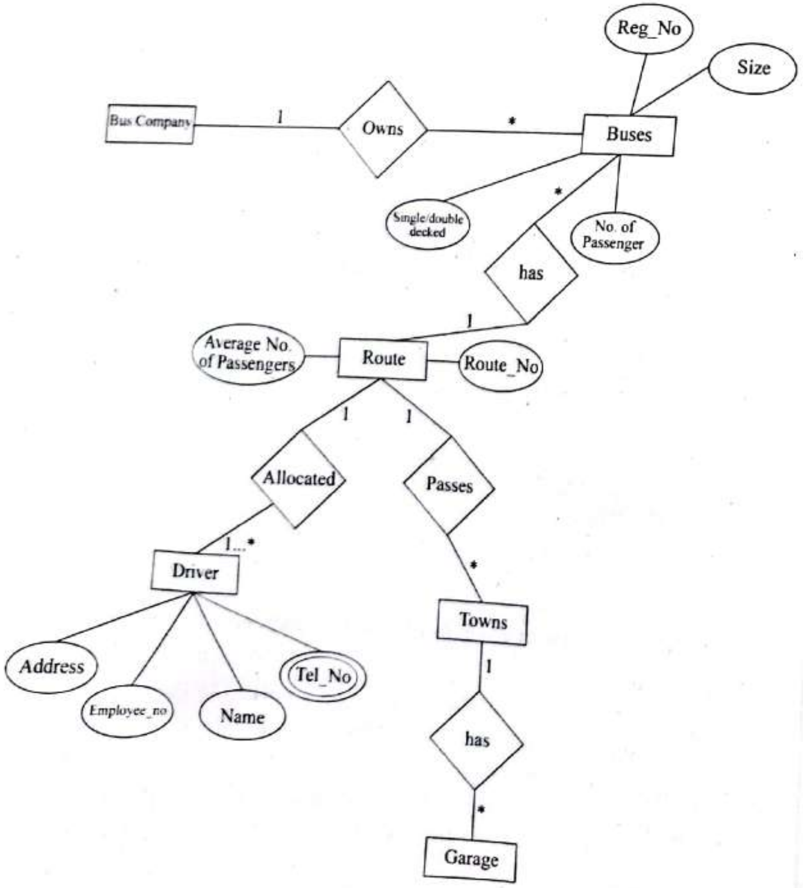
Cardinality: This specifics the number of each entity that is involved in the relationship there are 3 types of cardinality for binary relationship.

- One to One (1 : 1)
- One to Many (1 : n)
- Many to Many (n : m)

18. **Draw a complete ER-diagram for the following case: [2071 Bhadr**

A Bus Company owns a number of buses. Each bus is allocated to particular routte, although some buses may have several buses. Eac rate passes through a number of towns. One or more drivers ar allocated to each stage of a route, which corresponds to a journe through same or all of the buses on a route. Some of the towns have garage where buses are kept and each of the buses are identified b the registration number and can carry different numbers o passengers, since the vehicles vary in size and can be single or double decked. Each route is identified by a route number and information i available on the average number of passengers carried per day for each route. Drivers have an employee number, name, address and sometimes a telephone number."

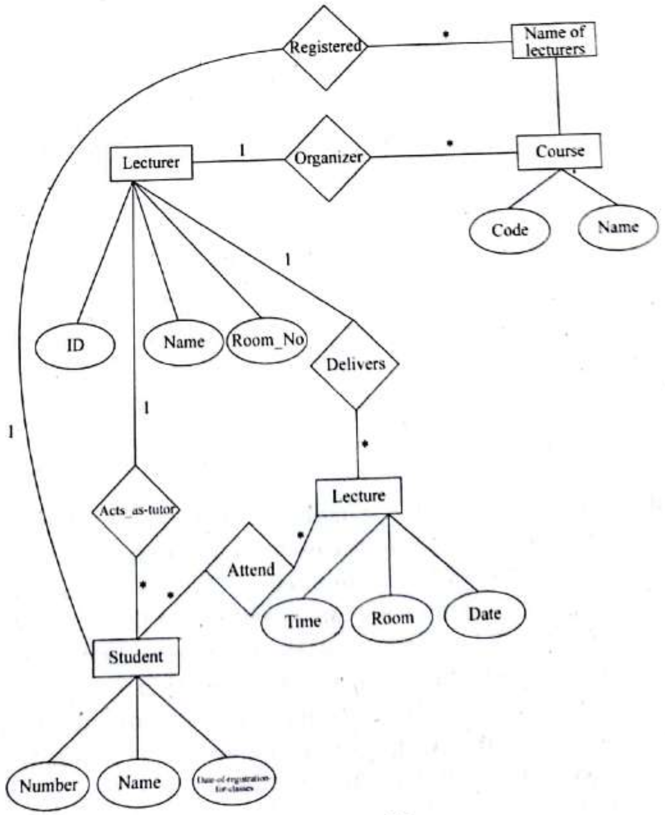
⇒



19. **Draw a complete ER diagram for the following case.**

"A lecturer(having an ID,name and room number)is responsible for organizing a number of course modules. Each module has a unique code and also a name and each module can involve a number of lecturers who deliver part of it..A module is composed of series of lecturers and sometimes lecturers on a given topic can be part of the more than one module. A lecture has a time,room, and date and is delivered by a lecturer and a lecturer may deliver more than one lecture..Students,identified by number and name ,can attend lectures and a student must be registered for a number of modules. We also store the date on which the student first registered for that module. Finally a lecturer acts a tutor for a number of students and each student has any one tutor. Explain generalization and specialization in ER diagram along with example? [2071 Magh]

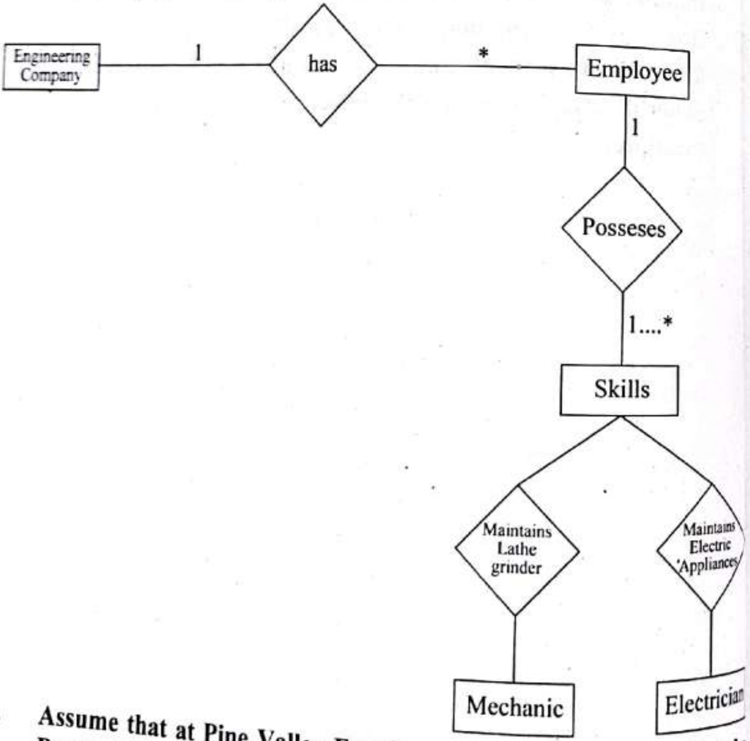
⇒



For Second Part: See in page no 20

20. **Draw an ER-diagram for the following mini –case.What is tl difference between strong and weak entity sets?Each employee i an engineering company has at most one recognized skill,but given skill may be processed by several employeeesAn employee i able to operate given machine-type(e.g.lathe,grinder)if he has o of several skills,but each skill is associated with the operation only one machine type.Possession of a give skill(e.g.mechanic,electrician)allows an employee to maintai several machine-types,although maintenanace of any give machine-type requires a specific skill(e.g. a lathe must b maintained by a mechanic) [2070 Bhadr**

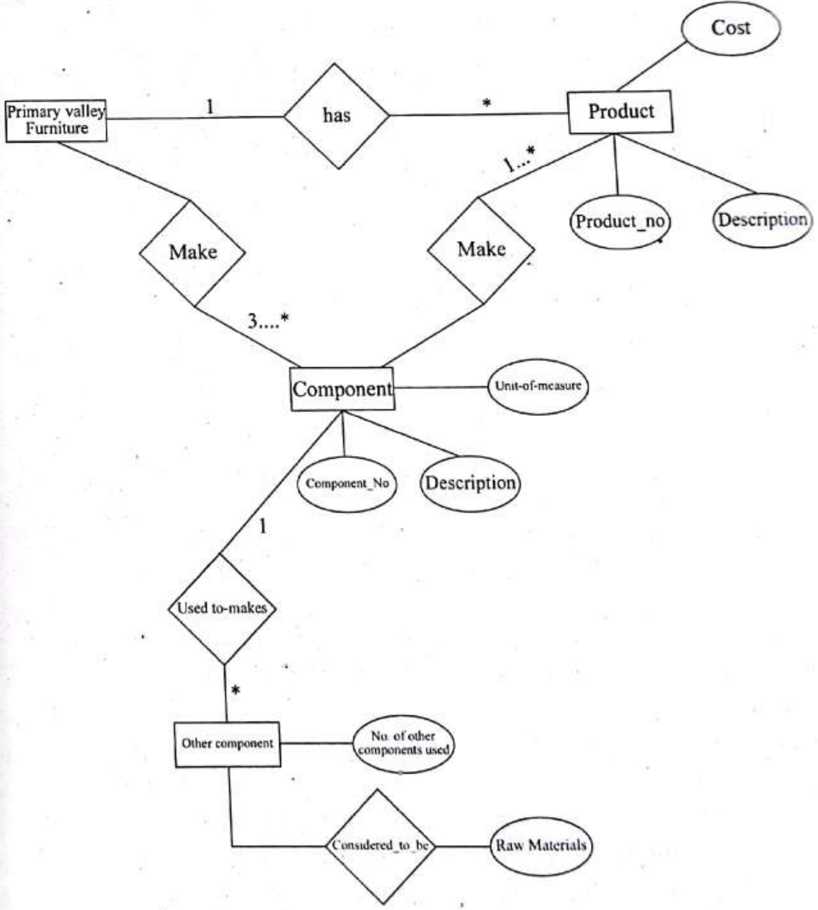
⇒ For 1st part see in old question no 12



21. **Assume that at Pine Valley Furniture each product(described by Product No., Description, and Cost)is comprised of at least three components(described by Component NO., Description and unit of Measure)and components are used to make one or many products(i.e. must be used in at least one product).In addition assume that components are used to make other components and that raw materials are also considered to be components. In both cases of components being used to make other components, we need to keep track of how many components go into making something else.**

Draw an ER-diagram for this case. Describe what is total participation using an ER-diagram example. [2070 Magh]

⇒



For 2nd part: See in old question no. 9.

22. **Draw an ER-diagram for the following mini-case. What is the difference between cardinality and degree of a relationship?**

A university registar's maintains data about the following entities: (a) Course, including number, title, credits, syllabus and prerequisites; (b) Course offerings, including course number, year, semester, section number, instructor(s), timings and classroom; (c) Students, including student-id, name, and program: and (d) instructors, including identification number, name department, and title. Further, the enrollment of students i courses and grades awarded to students in each course they ar enrolled for must be appropriately modeled. [8+4] [2069 Bhadra

CHAPTER 3

RELATIONAL LANGUAGES AND RELATIONAL MODEL

Relational Model

Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute.

Domain

It contains a set of atomic values that can attribute can take.

Attribute

It contains the name of a column in a particular table.

Instance

In the relational database, the relational instance is represented by a finite set of tuples. Relational instances do not have duplicate tuples.

Relational Schema

A relational schema contains the name of the relation and of name all columns or arributes.

Relational Key

In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

e.g:

STUDENT relation

Name	Roll No	Age	Address
Diwash	12	32	Btm
Pradeep	21	30	Damak
Bikash	10	54	Dharan
Prabin	22	24	Itahari

- In the given table, NAME, ROLL NO, age and address are attributes.
- The instance of schema student has 4 tuples.
- t_3 = < Bikash, 10, 54, Dharan >

SQL

SQL stand for structured Query Language. It is standards computer language for a accessing and manipulating database system. SQL works with database program like MS Access, MS SQL server, oracle etc.

Feature of SQL

- Render independent
- High level language
- Easier to understand and comprehensive too for manipulating data.
- Provides multiple views of data
- Extensibility and object technology

SQL Data Type

(i) Int → Store integer value

(ii) Float → Store float value

(iii) Double → Store characters and integer

(iv) Varchar → Store char values

(vi) Date → Store data values.

SQL commands

SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, function and queries of data.

1) **DDL (Data Definition Language)**

- DDL changes the structure of the table like creating a table, deleting a table, altering a table.

Here are some commands that comes under DDL

(a) **Create:** It is used to create a new table in the data base.

syntax:

CREATE TABLE NAME (COLUMN-NAME DATA TYPES [....])

E.g: CREATE TABLE STUDENT (Name VARCHAR (40) Roll No int)

(b) **DROP:** It is used to delete the structure and record stored in the table.

syntax:

DROP TABLE;

E.g: DROP TABLE EMPLOYEES;

(c) **A LTER:** It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probability to add a new attribute.

Syntax:

To add a new column in the table

ALTER TABLE table _ name ADD column _ name column _ description;

To modify existing column in the table

ALTER TABLE MODIFY (COLUMN DEFINATION.....)

E.g: ALTER TABLEE STU-DETAILS ADD (ADDRESS VARCHARS (20);

ALTER TABLE STU-DETAILS MODIFY (NAME VARCHARS (20);

(d) **TRUNCATE:** It is used to delete all the rows from the table and free the space containing the table.

Syntax:

TRUNCATE TABLE table _ name;

E.g: TRUNCATE TABLE EMPLOYEES;

2) **Data Manipulation Language**

DML commands are used to modify the database. It is responsible for all from of changes in the database.

(a) **Insert:** The insert statement is a SQL query. It is used to insert data into row of a table.

Syntax:

INSERT INTO TABLE _ NAME

(Col 1, Col 2, ... Col N)

VALUES (Value 1, Value 2, ... Value N)

E.g: INSERT INTO STUDENT

(Name_ Roll No) VAKUES ("Bikash", 3);

(b) **UPDATE:** This command is used to update or modify the values of column in the table.

Syntax:

UPDATE table-name SET [Column-name 1 = Value 1,.... column-name N = Value N]

[where condition]

For example:

UPDATE students

SET user name = 'ustav'

where student id = '3'

(c) **DELETE:** It is used to remove are or more row from a table.

Syntax:

DELETE FROM table _ name [WHERE condition];

For e.g:

DELETE FROM students

WHERE Author = "Bishal";

3) **Data Query Language**

DQL is used to fetch the data from the database.

- **Select:** This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

  Syntax:

  SELECT Expressions

  FROM TABLES

  WHERE conditions;

  For e.g:

  SELECT emp name

  FROM employee

  WHERE age>20

SQL Operators

(i) **Arithmetic operators**

+	Adds value of both operands
–	Subtract the right hand operand from left-hand operand
×	Multiplying values of both operands
/	Divide the left-hand operand by the right hand operand
%	It is used to divide the left-hand operand by the right-hand operand and returns remainder

(ii) **Comparison operators**

=	Equal to
! =	Not Equal to
< >	Not Equal to
<	Less than
>	Greater than
> =	Greater than equal to
< =	Less than equal to
! <	not less than
! >	not greater than.

4) **SQL Logical operators**

All → compares a value to all values in another value set.

AND → It allows the existence of multiple conditions in an SQL statement.

ANY - It compares the values in the list according to the condition.

BETWEEN - It is used to search for the values that are within a set of values.

IN - It compares a value to that specified list value.

NOT - It reserves the meaning of any logical operator.

OR - It combines multiple conditions in SQL statements.

Exists - It used to search for the presence of a row in a specified table.

LIKE - It compares a value to similar values using wild card operator.

Set Operation

The SQL set operation is used to combine the two or more SQL select statement:

Types of set operation

(i) Union

(ii) Union All

(iii) Intersect

(iv) Minus

Union

The SQL Union Operator is used to combine the result of two or more SQL SELECT queries. The Union operation eliminates the duplicate rows from its result set.

Syntax:

SELECT column _ name FROM table1

UNION

SELECT column _ name FROM table 2;

E.g:

SELECT * FROM table 1

UNION

SELECT * FROM table 2;

Union All

Union All operation is equal to union operation. It returns the set with removing duplication on and sorting the data.

Syntax:

SELECT column _ name FROM table

UNION ALL

SELECT column _ name FROM table 2

Interest

It is used to combine two SELECT statements. It returns the common rov from both the SELECT statements. It has no duplication and it arranges th data in ascending order by default.

Syntax:

SELECT column _ name FROM table 1

INTERESECT

SELECT column _ name FROM table;

Minus

It combines the result of two SELECT statement. Minus operator is used t display the rows which are present in the first query but absent in the secon query. It has no duplicates and data arranged in ascending order by default.

Syntax:

SELECT column _ name FROM table 1

MINUS

SELECT column _ name FROM table 2;

Relations

(i) **Join**

A Join operation combines related tuples from different relations. I and only if a given join condition is satisfied. It is denoted by $\bowtie$

E.g:

EMPLOYEE

EMP _ CODE	EMP _ NAME
101	Stephan
102	Jack
103	Harry

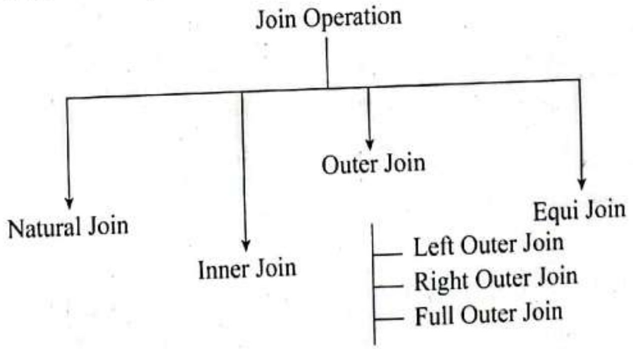
SALARY

EMP _ CODE	SALARY
101	50000
102	30000
103	25000

Operation: (Employee $\bowtie$ salary)

EMP _ CODE	EMP _ NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000

Types o Join operations:



1. **Natural Join:**

- A natural join is that of tuples of all combinations in R and S that are equal on their common attribute names.
- It is denoted by $\bowtie$.

E.g: Let's use the above EMPLOYEE table and SALARY table:

Π Emp _ NAME, SALARY (EMPLOYEE $\bowtie$ SALARY)

Output:

EMP _ CODE	EMP _ NAME
Stephan	50000
Jack	30000
Harry	25000

2. **Outer Join**

The Outer join operation is an extension of the join operatio It is used to deal with missing information.

E.g:

EMPLOYEE

EMP _ NAME	STREET	CITY
Ram	Civil Line	Mumbai
Shyam	Parn street	Kolkata
Ravi	M.G street	Delhi
Hari	Nehru	Hyderabad

FACT _ WORKERS

EMP _ NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

Input: Employee ⋈ fact _ workers

Output:

EMP _ NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil Line	Mumbai	Infosys	10000
Shyam	Park sheet	Kolkata	Wipro	20000
Hari	Nehu Nagar	Hyderbad	TCS	50000

An outer join is basically of three types:

(a) **Left Outer Join**

- Left Outer join contains the set of tuples of all combinations in R and S that are equal on their common attributes names.

- In the left outer join, types in R have no matching tuples in S.
- It is denoted by ⟕

E.g: Using the above EMLPOYEE table and FACT _ WORKERS table.

Input:

EMPLOYEE ⟕ FACT _ WROKERS

EMP _ NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil Line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehu street	Hyderbad	TCS	50000
Ravi	M.G street	Delhi	NULL	NULL

(b) **Right Outer Join:**

- Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In right outer join, tuples in S have no matching tuples in R.
- It is denoted by ⟖

E.g: Using the above EMPLOYEE table and FACT_WORKERS Relation.

Input:

EMPLOYEE ⟖ FACT _ WORKERS

Output:

EMP _ NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderbad
Kuber	HCL	30000	NULL	Null

(c) **Full Outer Join:**

- Full outer join is like a left or right join except that is contains all rows from both tables.

- In full outer join, tuples in R that have on matching tu and tuples in S that have no matching tuples in R th common attribute name.
- It is denoted ⟗

E.g: Using the above EMPOYEE table and FACT_WORKE table.

Input:

Employee ⟗ FACT _ WORKERS

Output:

EMP _ NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil Line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehu street	Hyderbad	TCS	50000
Ravi	M.G street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

Inner Join

An inner join is a join in which the DBMS selects records from two table when the records have the same value in the common field that links th table.

Equi Join

It is a join in which the joining condition is based on equality between value in the common columns, common on columns appear redundantly in the result table.

Embedded SQL

Embedded SQL is a method of inserting in line SQL statement or queries int the code of a programming language, which is known as a host language Because the host language cannot parse SQL, the inserted SQL is parsed b an embedded SQL pre processor.

Embedded SQL is a robust and convenient method of combining the computing power of a programming language with SQL's specialized data management and manipulation capabilities.

Views

- Views in SQL are considered as a virtual table. A view also contains rows and column.

To create the view, we can select the fields from one or more tables present in the database.

Sample Table:

Student _ Detail

STU _ ID	Name	Address
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugram

STU _ ED	NAME	MARKS	AGE
1	Stephan	97	19
2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18

1: **Creating View**

Syntax:

```
CREATE VIEW view _ names AS
SELECT column 1, column 2 ....
FROM table _ name
WHERE condition;
```

2. **Creating view from single table**

Query:

```
CREATE VIEW Detail AS
SELECT NAME, ADDRESS
FROM Student _ Details
WHERE STU _ ID < 4,
```

Just like table query, we can query the view to view the data.

```
SELECT FROM Details view;
```

3. **Creating view From Multiple tables**

- It is known as complex view.

Query:

```
CREATE VIEW MARKS view AS
SELECT Student _ Detail . NAME, Student _ Detail
ADDRESS, Student _ Marks . MARKS
FROM Student _ Detail, Student _ Mark
WHERE Student _ Detail . Name = Student _ Marks . Nam
```

To Display view

```
SELECT * FROM Marks view;
```

4. **Deleting view**

A view can be deleted using the Drop view statement.

Syntax:

```
DROP VIEW view - name;
```

Example:

If we want to delete the view Marks view, we can do this as:

```
DROP VIEW Marks view;
```

Relational Algebra

Every database management system must define a query language to allo users to access the data stored in the database. Relational Algebra is procedural query language used to query to database to access in differe ways.

Operations on Relational Algebra

1. **Select Operation**

- The select operation selects tuples satisfy a given predicate.
- It is denoted by sign (σ)

Where,

σ is used for selection predication

r is used for relation

P is used as a propositional logic formula which may use connectors like: AND, OR and NOT. These relational can use as relational operation like =, ≠, ≥, <, > ≤.

E.g: $\sigma_{\text{BRANCH \_ NAME = "Perry ride"}}$ (LOAN)

2. **Project Operation**

- This operation shows the list of those attributes that we wish to appear in th result. Rest of the attributes are eliminated from the table.
- It is denoted by π.

Notation: π A_1, A_2, A_n (r)

Where;

A_1, A_2, A_3 is used as an attributes name of relation r.

E.g: $\pi_{\text{NAME, CITY}}$ (CUSTORMER)

3. **UNION operator**

- Suppose those are two tuples R and S. The union operation contains all the tuples that are either in R and S or both in R and S. It is denoted by ∪.

Notation: R∪S

E.g: $\pi_{\text{CUSTOMER \_NAME}}$ (BORROW)U $\pi_{\text{CUSTOMER\_NAME}}$ (DEPOSITOR)

4. **Set Intersection**

- Suppose there are two tuples R and S. The set interaction operation contains all tuples that are in both R and S.
- It is denoted by intersection ∩.

Notation: R ∩ S

Example:

$\pi_{\text{CUSTOMER\_NAME}}$ (BORROW) ∩ $\pi_{\text{CUSTOMER\_NAME}}$ (DEPOSITER)

5. **Set Difference**

- Suppose there are two types R and S. The set intersection operation contains all tuples that are in R but not ins.
- It is denoted by intersection minus (–).

Notation: R – S

E.g: $\pi_{\text{CUSTOMER\_NAME}}$ (BORRW) _ $\pi_{\text{CUSTOMER \_ NAME}}$ (DEPOSITER)

6. **Cartesian Product**

- The Cartesian product is used to combine each row in one table with each row in the outer table. It is also known as a cross product.

- It is denoted by X.

Notation: EXD

E.g: EMPLOYEE × DEPARTMENT

7. **Rename Operation**

The rename operation is used to rename the output relation. It i denote by rho (p).

E.g: We can use the rename operator to rename STUDENT relatio to STUDENT 1.

ρ (STUDENT 1, STUDENT)

8. **Aggregate Function Operations**

- Aggregation Function takes a collection of values and returns a single value as a result.
  - avg: Average value
  - min: Minimum value
  - Max: Maximum value
  - Sum: Sum of values
  - Count: number of values
- Aggregate operation in relational algebra

  $G_1, G_2, G_3 \ldots\ldots 9\ F_1(A_1), F_2\ (A_2 \ldots\ldots F_n(A_n)\ (E)$
- Where E is any relational - algebra expression and $G_1, G_2 \ldots\ldots G_n$ is a list of attributes on which to group (can be empty).
- Each F_i is an aggregate function
- Each A_i is an attributes name

9. **Delete Operation**

- A delete request is expressed similarly to a query, except instead of displaying tuples to the user; the selected tuples are removed from the database.
- Can delete only whole tuples; cannot delete value on only particular attributes.
- A deletion is expressed in relational algebra given by
  - $r \leftarrow r - E$

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

- Delete all account record in the New road branch.

  $\text{Account} \leftarrow \text{Account} - \sigma_{\text{branch\_name = "New Road"}}\ \text{(account)}$
- Delete al account at branches located in KTM

  $r_1 \leftarrow 6_{\text{branch\_city = "Need ham"}}\ (\text{account} \bowtie \text{Branch})$

  $r_2 \leftarrow \Pi_{\text{account\_name, branch\_name, balance}}\ (r1)$

  $r_3 \leftarrow \pi_{\text{customer\_name, account\_name}}\ (r2 \bowtie \text{depositer})$

  $\text{Account} \leftarrow \text{account} - r_2$

  $\text{Depositer} \leftarrow \text{depositer} - r_3$

10. **Insertion**

- To insert data into a relation, we either.
- Specify a tuple to be inserted.
- Write a query whose result is a set of tuples to be inserted.
- In relational algebra, an insertion is expressed by:
  - $r \leftarrow r \cup E$

  Where r is a relation and E is a relational algebra expression.
- The insertion of a single tuple is expressed by letting E be a constant relation containing one tuple.
- Insert information in the database specifying that smith has Rs. 1200 in account A - 973 at the Patan branch.

  Account ← account ∪ [("A - 973". "Patan", 1200)]

  Depositer ← depositer ∪ [("Smith", A - 973)]
- Provide as a gift for all loan customers in the Patan branch, Rs. 200 savings account. Let, the loan number serve as the account numbers for the new saving account.

  $r_1 \leftarrow (\sigma_{\text{branch\_name = "Patan"}}\ (\text{borrow} \bowtie \text{loan})$

  $\text{Account} \leftarrow \text{account} \cup \Pi_{\text{loan\_number\_branch\_name,200}}\ (r1)$

  $\text{Depositer} \leftarrow \text{depositer} \cup \Pi_{\text{customer\_name\_loan number}}\ (r1)$

11. **Updating**

- A mechanism to change a value in a tuple without charging all values in the tuples.
- Use the generalized projection operator to do this task.

  $r \leftarrow \Pi_{F1, F2, \ldots\ Ft}\ (r)$

- Each F_i is either
  - The I^{th} attribute of r, if the I^{th} attribute is not updated
  - If the attribute is to be updated F_i is an expressi involving only constants and the attributes for, wh gives the new value for the attribute.
- Pay all accounts with balances over \$ 10,000 6 percent inter and pay all others 5 percent.

  Account ← $\pi_{account\_number,\ branch\_name,\ balance\ *\ 1.06}$ ($\sigma_{BAL,\ balanc}$ $_{1000(account)}$) ∪ $\Pi_{account\_number,\ branch\_name,\ balance\ *\ 1.05}$ (σ_{BAL} $_{1000(account)}$)

Binary Relational Operations JOIN

- The general form of a join operation on two relations R (A_1, ... A_n) and S (B_1, B_2 …… B_m) is:

  R ⋈ < Join conditions > S

  - Where R and S can be any relations that result fro general relational algebra expressions.
- The join condition is called theta.
- Theta can be any general Boolean expression on attributes of l and S.
- Most join conditions involve one or more equality condition "AND" ed together.

12. **Natural - Join Operation**

- Notation r ⋈ s
- Let r and s be relations on schema R and S respectively. Then ⋈ s is a relation on schema R ∪ S obtained as follow
  - Consider each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attribute in R ∩ S, add a tuple to the result, where
    - t has the same value as t_r on r.
    - t has the same value as t_s on s.

E.g:

R = (A, B, C, D)

S = (E, B, D)

Result Schema = (A, B, C, D, E)

r ⋈ s is defined as:

$\Pi_{r.A,\ r.B,\ r,\ r.\ C,\ r.D,\ S.E\ (6r.B\ =\ S.B\ \wedge\ r.D\ =\ S.D\ (r\ \times\ s))}$

Domain Relational Calculus

The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain attributes. Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives ∧ (and), V (or) and (not). It uses existential (∃) and universal quantifiers (∀) to bind the variable.

Notation:

{a_1, a_2, a_3 …… a_n/ p (a_1, a_2, a_3 …… a_n}

Where,

a_1, a_2 are attributes

P stands for formula buit by inner attributes.

For e.g:

{< article, page, subject > ε java point ∧ subject = 'database'}

QBE (Query By Example)

QBE is a query method implemented in most database system, most notably for relational database. QBS was created by Moshe Z 100 F at IBM in the 1970's in parallel tp SQL's development. It is a graphical query language where users can input commands into a table like conditions and example elements. It's a common feature in most database.

QBE is a query language used in relational database that allows users to search for information in fields and tables by providing a simple user interface where the user will b. able to input an example of the data that he or she wants to access.

Old Question Solution

1. **Write SQL query. [Consider following relations]**

   Product (<u>Pid</u>, Pname, Price, description)

   Customer(<u>Cid</u>, Cname, Address)

   Sells(Pid, Cid, quantity)

   (a) **Retrieve the record of product who were sold to customer id 12.**

   (b) **Create above table product as indicated.**

   (c) **Find the product whose sells quantity is maximum.**

   (d) **Find the total number of customer whose name start with S.** **[8][2076 Baisakh]**

⇒ (a) SELECT * FROM PRODUCT WHERE Pid = 12;

(b)
```
CREATE  TABLE product (
            Pid int NOT NULL,
            Pname varchar (50),
            Price Float,
            Desription varchar (255),
            PRIMARY KEY (Pid)
            );
CREATE  TABLE Customer (
            Cid int NOT NULL,
            Cname varchar (50),
            Address varchar (50),
            PRIMARY KEY (Cid)
            );
CREATE  TABLE Sells (
            quantity int,
            FOREIGN KEY (Pid) REFERENCES Prod
            (Pid),
            FOREIGN KEY (Cid) REFERENCES Custom
            (Cid)
            );
```

(c) SELECT Pid Max (quantity) FROM Sells;

(d) SELECT COUNT (Cid) FROM Customer where cname LIK ('S%');

2. **Write relational algebra. [Consider following relations.]**
Emp (Eid, Ename, Address, Salary, Dptid]
Depart(Dptid, Dname)
   (a) **Insert a single record in Emp tabl (100,'Ram','Balaju',1000,5)**
   (b) **Retrieve the record of employee who earns more tha 10000 in computer department.**
   (c) **Increase the salary of all employee by 10 percent.**
   (d) **Delete all the record of employee who are from EL department.(Dptid='ELX')** **[8] [2076 Baisakh**

⇒ (a) Emp ← Emp U{("100, 'Ram', 'Balaju', 1000, 5)}

(b) $\Pi_{\text{Eid, Ename, Address, Salary, Dptid}}$ (σ_{salary} > 1000 and Dname "Computer" (Emp $\bowtie_{\text{Emp . Dpitd = Depart. Dptid}}$ Depart)

(c) Emp ← $\Pi_{\text{Eid, Ename, Address, Salary * 1.01, Dpitd (Emp)}}$

(d) Emp ← Emp - $\sigma_{\text{Dpid="ELX"}}$ (Emp)

3. **Consider the following relational data model.** [2×4]
Student (crn, name, address, phone, dob)
Course (courseid, crn, duration, fee)
Enroll (enrolled, cname, courseid, enrolldata, completedata)
   (i) **Write the SQL statements required to create the above relations, including appropriate versions of all primary and foreign key integrity constraints,**
   (ii) **Write an expresion in SQL to find crn, names and enroll data of all students who have taken the course 'java' (cname)**
   (iii) **Write SQL to find the names and address of all the students who have taken both course java and linux.**
   (iv) **Write an expression in SQL to Create a view 'student_course' having the attributes crn, name, phone, coursename, enrolldata** **[2075 Bhadra]**

⇒ (i)
```
Create Table Student (
            crn int NOT NULL,
            name varchar (50),
            Address varchar (200),
            Phone varchar (20),
            dob varchar (20),
            PRIMARY KEY (crn)
            );
Create Table Course (
            Course – id int NOT NULL,
            duration int,
            Fee Float
            FOREIGN  KEY  (crn)  REFERENCES  student
            (crn),
            FOREIGN KEY (course id)
Create Table Enroll (
            Enrolled int NOT NULL,
            cname varchar (50),
            enroll data varchar (50),
            complete data varchar (200),
            FOREIGN KEY (courseid) REFERENCES course
            (courseid)

);
```

(ii) SELECT crn, name, enroll data FROM Student JOIN co ON Student.crn = Course.crn JOIN ENROLL ON Co course id = ENROLL . course id WHERE cname = "JAVA'

(iii) SELECT name, address FROM Student JOIN COURSE (Student . crn = Course . crn JOIN ENROLL ON Cour course =_ id = Enroll . course id WHERE cname = "JAV AND cname = "Linux"

(iv) CREATE view Student _ Course

SELECT crn, name, phone, coursena enrolldata FROM Student JOIN course (Student . crn = Course.crn JOIN Enroll (Course . course id = Enroll . course id

4. **Consider the following relational database**

sailor (sailorid, sname, rating, age)

boat (boatid, boatname, color)

reserves (sailorid, boatid, date)

Write relational algebra expressions for the following:

(i) Find the names of sailor who has reserved boat numbe 105.

(ii) Find the names of sailors who have reserved a red boat.

(iii) Find the names of all sailor who have reserved either a re boat or a green boat.

(iv) Give an expression in QBE to find the sailors name and ag who have reserved a red boat.

[2×4] [2075 Bhadra

⇒ (i) $\Pi_{sname}(\sigma_{boatid = 105}(Sailor \bowtie_{sailor.sailorid = reserves.sailorid} Reserve \bowtie_{boat.boatid = reserve.boatid} boat))$

(ii) $\Pi_{sname}(\sigma_{color} = \text{"Red"} (Sailor \bowtie_{Sailor.Sailorid = reserves.Sailor} Reserves \bowtie_{boat.boatid = reserves.boatid} boat))$

(iii) $\Pi_{sname}(\sigma_{color = \text{"Red"} OR\ color = \text{"Green"}} (Sailor \bowtie_{Sailor.Sailorid = reserves\ sailorid} Reserves \bowtie_{boat.boatid = reserves.boatid} boat))$

(iv) SELECT sname, age FROM Sailor JOIN reserves ON Sailor . Jailorid = reserves . sailorid JOIN boat ON boat . boatid = reserves . boatid.

5. **Consider the following relational schema:**

tblsalesman(s id, name, city, commission)

tblOrders(ord no, prch_amt, ord_date, c_id, s_id)

tblCustomer (c id, name, city, grade, s_id)

Write SQL query expression to **[2×4] [2075 Baisakh]**

(a) Find those salesmen with all information whose name containing the 1st character is 'N' and the 4th character is 'R' and rests may be any character.

(b) Find the highest purchase amount on a data '2017-07-17' for each salesman with their ID.

(c) count the customers with grades above Kathmandu's average.

(d) Increase commission of salesmen by 2% if they are from humla.

⇒ (a) SELECT * FROM tbl salesman WHERE name LIKE ("N_ R"%);

(b) SELECT s_id, MAX (prch _ amt) FROM tbl orders WHERE ord _date = "2017 – 07 – 17"

(c) SELECT COUNT (c _ id) FROM tbl customers WHERE grade > (SELECT grade FROM) tbl customer WHERE City = "Kathmandu")

(d) UPDATE tbl salesman
SET commission = CASE
WHEN City = "humla"
THEN
Commission = 1.2 * commission
END

6. **Consider the following relational database model**

Author(a name, citizenship, birth Year) Book(isbn, title, a_name)

Topic(isbn, subject) **Branch(libname, city)**

Instock(isbn, libname, quantity)

Write relational algebra expressions for the following: **[2×4]**

(a) Give the cities where each book is held.

(b) Give the title and author of each book of which at least two copies are held in a branch located in Kathmandu.

(c) Delete those books that are from author 'xyz'

(d) List total no. of available books of each subject.

⇒ (a) $\Pi_{\text{city, title}}$ (Book $\bowtie_{\text{Instock . isbn = Book . isbn}}$ Instock $\bowtie_{\text{libname = Branch libname}}$ Branch)

(b) $\Pi_{\text{title, author}}$ ($\sigma_{\text{quantity} \geq 2 \text{ AND city = "Kathmandu"}}$ Book $\bowtie_{\text{Instock Book . isbn}}$ Instock $\bowtie_{\text{Instock . libname = Branch . libname}}$ Branch)

(c) BOOK ← BOOK − $\sigma_{\text{a - name = "XYZ"}}$ (BOOK)

(d) title G count (isbn) (BOOK)

7. **Consider the following relational data model [2×4] [2074 Bha**

Employee (<u>empid</u>, ename, age, salary)

Department (<u>deptid,</u> dname, budget, <u>managerid</u>)

Works (<u>empid</u>, <u>depid</u>, hours)

(i) **Write the SQL statements required to create the a relations, including appropriate versions of all primary foreign key integrity constraints.**

(ii) **Write an expression in SQL to find the name of departm whose employee earns the maximum salary.**

(iii) **Write SQL to find the name of the employee, departm name and the number of hours they work.**

(iv) **Write an expression in SQL to give every employee a 20 raise in salary whose age is in between 45 to 50 years.**

⇒ (i)

```
CREATE table Employee (
        empid int NOT NULL,
        ename varchar (30),
        age int,
        Salary Float,
        PRIMARY KEY (empid)
        );
CREATE table Department (
        depitd int,
        dname varchar (30),
        budget Float,
        Management int,
        PRIMARY KEY (deptid),
        PRMARY KEY (managerid)
        );
```

```
CREATE table WORKS (
        hours int,
        FOREIGN KEY (empid) REFERENCES Employee
        (empid)
        FOREIGN KEY (deptid) REFERENCES
        Department (deptid)
        );
```

(ii)

```
SELECT dname, Max (Salary) FROM Employee JOIN ON
Employee . empid = Works . empid Works
JOINS Department ON Works . deptid = Department . depitd
```

(iii)

```
SELECT ename, dname, hours FROM
      Employee JOIN Works ON Employee . emid = Works .
      emid JOIN Department ON Department . deptid =
      Works . deptid
```

(iv)

```
UPDATE Employee
SET Salary = CASE
WHEN age BETWEEN 45 AND 50
THEN
Salary = 1.2 * Salary
END
```

8. **Consider the following relational database [2×4]**

Account (<u>account-number</u>, branch name, balance)

Branch (<u>branch name</u>, branch city, assests)

Customer (<u>cust name</u>, cust street, cust city)

Loan (<u>loan number</u>, branch name, amount)

Depositor (<u>cust name</u>, <u>account number</u>)

Borrower (<u>cust name</u>, loan number)

Write the relational algebra expressions for the following:

(i) **Find the names of customers who has loan at "Koteshwor" branch.**

(ii) **Find the largest account balance.**

(iii) **Find the names of all depositors along with their account number, street and city address.**

(iv) Give an expression in QBE to find the customer name, number and amount for all customers who have a from the "Koteshwor" branch.

⇒ (i) $\Pi_{cust-name}(\sigma_{branch-name} = \text{"Kathmandu"} (\text{Loan} \bowtie_{Loan.loan-number = Borrower.loan\_number} \text{Borrower}))$

(ii) Max (balance) (Account)

(iii) $\Pi_{cust-name, account-number, cust-city}(\text{Customer} \bowtie_{Customer.cust\_ Depositer.cust\_name} \text{Depositer})$

(iv) $\Pi_{cust.-no, loan-number, amount}(\text{Loan} \bowtie_{loan.loan\_number = Borrower.l number} \text{Borrower} \bowtie_{Borrower.cust\_name = customer.cust\_name} \text{custom}$

9. **Consider the following relational schema:**

Employee(<u>Ename</u>,street,city)

Works(Ename,company_name,salary)

Company(<u>company name</u>,city)

Manages(Ename,manager_name)

a) **Write the queries in Relational algebra.**

i) **Find all the employees name who work in NM bank.**

ii) **Find all the employee names who live in the sam city as their company is located.**

iii) **Find the name and city of those employees whos salary is greater than 30000 and lives in 'ktm'city.**

[2×3] [2073 Bhadra

⇒ (i) $\Pi_{name}(\sigma_{company-name} = _{\text{'NMB Bank'}} (\text{Works} \bowtie_{employee.ename = works ename} \text{Employee}_)$

(ii) $\Pi_{name}(\text{Employee} \bowtie_{employee.city = company.city} \text{company})$

(iii) $\Pi_{ename, cirty}(\sigma_{city} = _{\text{'KTM' AND Salary} > 3000} (\text{Employee} \bowtie_{employee.ename = works.ename} \text{works}))$

10. **Write SQL queries for the following:**

i) **Create Employee and works relation with primary key and foreign key constraints.**

ii) **Find the employee,name their company name and city name which ends with pur as substring.**

iii) **Increase the salary of each employees by 25% whose salary is less than 3000.** **[2×3] [2073 Bhadra]**

⇒ (i)

```
CREATE TABLE Employee (
        Ename varchar (255),
        Street varchar (255),
        City varchar (255),
        PRIMARY KEY (Ename),
        FOREIGN KEY (Ename) REFERENCES
        WORKS (Ename)
        );
```

(ii)

```
SELECT Ename, company – name, city – name FROM,
       Employee JOIN Works ON Employee . Ename =
       Works . Ename WHERE Ename Like '1% pur'
```

(iii)

```
UPDATE WORKS
SET Salary = CASE
WHEN Salary < 3000
THEN
Salary = 1.25 * Salary
END
```

(ii) SELECT ename FROM Employee JOIN Works – on ON Employee . empid = Works – on . empid JOIN Project ON project . Pid = Works – on . Pid WHERE balance > 50000

(iii) SELECT dname, count (Pid)

FROM DEPARTUREMENT . JOIN Project on department deptid = Project . deptid GROUP BY deptid

11. **Consider the following relational data model.** **[2×3]**

Employee(empid, name, address, manager_id)

Department(deptid, dname)

Project(pid, title, budget, deptid)

Works_on(empid, pid, hours)

Write down the relational algebraic expression for the following:

(i) **Find the names of all employee from computer department along with their manager name.**

(ii) **Find the names of all the employees who works on proj with budget more than 50000.**

(iii) **Find the total number of projects from each departm along with the department name. [2073 Mag**

⇒ (a) $\pi_{\text{name, manager - name}}$ ($\sigma_{\text{dname = "Computer"}}$ (Department $\bowtie_{\text{departme dept - id = Employee . deptid}}$ Employee))

(b) π_{name} ($\sigma_{\text{budget > 5000}}$ (Project $\bowtie_{\text{project . pid = Works - on . Pid}}$ Works on $\bowtie_{\text{employee . empid = Works - on . empid}}$ Employee)

(c) $H \leftarrow G_{\text{count (Pid)}}$ (Project)

Ans $\leftarrow \Pi_{\text{H1,}}$ dance (Department)

12. **Write down the SQL queries for following:**

i) **Find the name of employees who works on project with th highest budget.**

ii) **Create a view with empid,name,project title and budget**

iii) **Update the budget of all project by 20% where an employee works for more than 12 hours. (2073 Magh**

⇒ (i)

```
SELECT ename, max (budget) FROM EMPLOYEE JOIN
Works – ON ON Employee . empid = Works – on . emid JOIN
Project ON Works – on Pid = Proect . Pid
```

(ii)

```
CREATE view view1

SELECT empid, name, project title and budget FROM
EMPLOYEE JOIN WORKS – ON on Employee . empid =
Works – on . empid JOIN project ON Project . Pid = Works –
on . Pid
```

iii)

```
UPDATE Project JOIN Works – ON Project . pid = Works –
ON Pid

SET = CASE

WHEN hours > 12

THEN

budget = 1.2 * budget

END
```

13. **Write relational algebra queries for(a,b,c).Write the SQL queries for(i,ii,ii)**

a) **Retrieve the detail of employee with eno,add,dob,phone with highest salary.**

i) **Create above table Emp as indicated.**

ii) **Find the employee who earns more than 50000,works in CS department and name contains alphabet a.**

iii) **Increase salary of those employee who earns less than average by 25%.**

b) **Find total amount spent by ECON department for its employee salary**

c) **Find total number of post in CS department. [2072 Ashwin]**

⇒ (a) Relational

Here,

$H1 = \text{Max}_{\text{(salary)}}$

Ans: $\Pi_{\text{eno, add, dob, phone, H1}}$ (Employee)

(b) $H1 = \text{Sum}_{\text{(Salary)}}$ Employee

Ans: Π_{H1} ($\sigma_{\text{dept = 'ECON'}}$ (Employee))

(c) $H1 = \text{Count}_{\text{(post)}}$ (Employee)

Ans: Π_{H1} ($\sigma_{\text{dept = 'CS'}}$ (Employee))

(i)

```
CREATE TABLE Emp (
        eno int NOT NULL,
        add varchar (255)
        dob varchar (255)
        phone int,
        salary int
        );
```

(ii)

```
SELECT emp – name FROM Employee WHERE dept = 'CS
department' AND Salary > 50000 AND emp – name LIKE % a
%;
```

(iii) UPDATE Employee

SET Salary CASE

WHEN Salary < AVG (Salary)

THEN

Salary = 1.25 * (Salary)

END

14. **Consider the following relational scheme:** **[2+6] [2072 M**

Account (<u>account number</u>, branch_name, balance)

Branch (<u>branch name</u>, branch city, assets)

Customer (<u>Customer name</u>, customer_street, customers_city)

Loan (<u>loan number</u>, branch_name, amount)

Depositor (<u>customer name</u>, account number)

Borrower (<u>customer name</u>, loan number)

⇒ (a) SELECT * FROM customer JOIN Depositer ON custome customer – name = Depositer . customer – Name JO Account ON account – no = depositer . account – no JO Branch ON Branch . branch – name = Account . branch name.

(b) SELECT branch – name FROM Branch JOIN Account (Branch – name = Account branch WHERE AVG (balance(50000

(c) UPDATE Account

WHEN SET balance = CASE

balance > 1000 THEN

balance = 1.05 * balance

ELSE

balance = 1.06 * balance

END

(d) SELECT branch – cities COUNT (assets) FROM Branch WHERE assets > 100000

(e) Count account – no (Account $\bowtie_{\text{account . branch\_name = Branch . branch name}}$ Branch)

(f) $\Pi_{\text{amount}}(\text{Loan}) - \Pi_{\text{amount}}(\sigma_{\text{amount} < 1000}(\text{Loan}))$

15. **Consider the following relational database model** **[2071 Bhadra]**

Employee(<u>eid</u>, name, address, supervisor_eid)

Department(<u>dept_id</u>,name)

Project(<u>pid</u>, title, dept_id)

Works_on(<u>eid</u>, <u>pid</u>, hours)

Write relational algebra expresions for the following: **[2×4=8]**

(a) **List the name of all employees from computer department along with the name of their supervisor.**

(b) **Find the name of all employees who work on the "Network monitoring" project for more than 15 hours.**

(c) **Delete all projects which belong to the "Electrical" department.**

(d) **Find the total number of projects from each department, along with the department name.**

⇒ (a) $\Pi_{\text{name, supervisor - name}}(\sigma_{\text{dept - name = "computer"}}(\text{Department} \bowtie_{\text{department . dept - id = Project . dept - id}} \text{Project} \bowtie_{\text{project . Pid = Works - ON . Pid}} \text{Works ON} \bowtie_{\text{employee . eid = Works - ON . eid}} \text{Employee}))$

(b) $\Pi_{\text{name}}(\sigma_{\pi\,\text{title = "Network Monitory" AND hours = 15}}(\text{Project} \bowtie_{\text{Works - on . pid = project . Pid}} \text{Works\_on} \bowtie_{\text{Employee . eid = Works - on . eid}} \text{Employee}))$

(c) $\text{Project} \rightarrow \text{Project} - \pi_{\text{pid}}(\sigma_{\text{name = 'Electrical'}}\ \text{Department} \bowtie_{\text{department . dept\_id = project . dept - id}} \text{project})$

(d) name g $\text{count}_{(\text{Pid})}$ $(\text{Department} \bowtie_{\text{department . dept\_id = project.dept\_id}} \text{Project})$

16. **Consider the relational schema given below.** **[2×4=8]**

Project (<u>pid</u>, name, price, category, maker-pid)

Purchase (buyer-ssn, seller-ssn, quantity, pid)

Company (<u>cid</u>, name, stock price, country)

Person(<u>ssn</u>, name, phone number, city)

(a) **Write an SQL query to find the name and price of all products of "camera" category made in "Japan".**

(b) **Write an SQL query to create a view to expose only the Buyer name, Seller name and product name from all transactions.**

(c) **Write a query in SQL to increase the price of all prod from DELL company by 5%.**

(d) **Write skeleton tables in QBE to find the name and ph number of all persons who purchased products of lap category with price greater than 80,000.**

⇒ (a) SELECT name, price FROM product

JOIN company ON

Product. cid = company . cid

WHERE category = "Camera" AND Country = "Japan"

(b) CREATE VIEW expense As

SELECT Buyer – ssn, seller – ssn, name FROM

Product JOIN purchase ON product . pid = Purchase . pid

JOIN person ON purchase . ssn = person . ssn

(c) UPDATE Product

JOIN company ON product . cid = company . cid

SET price = 1.05 * price

WHERE company . name = "DELL"

(d) QBE

Cid	Name	Phone number	Price	Category
Join	P.X	P.Y	> 8000	Laptop

17. **Consider the following relational data base modal**

Employee(eid, name, address, supervisor_eid)

Department(dept id, name,)

Project(pid, title, dept_id)

Works_on(eid, pid, hours)

Write relational algebra expressions for the following: [2×4=8]

(a) **List the titles of all projects along with the department names.**

(b) **Find the names of all employees who live in"Kathmandu"and are supervised by employee who lives in "Kathmandu".**

(c) **Increase the working hours of all employees who work in the "voter registration"project by 5hrs.**

(d) **Find the total number of employees involved in each project along with the project title.**

[2071 Magh]

⇒ (a) $\pi_{\text{title, name}}$ (Project $\bowtie_{\text{project . dept\_id}}$ = Department . dept – id Department)

(b) π_{name} (σ_{address} = "Kathmandu" AND supervisor – address = "Kathmandu" Employee)

(c) Works – on ← $\pi_{\text{eid, Pid,hours S + 5}}$ (Works – on $\bowtie$ project pid = Works – on . Pid project)

(d) H1 = $G_{\text{count (eid)}}$ (Works – on)

Total No ← $\Pi_{\text{H1, title}}$ (Project)

18. **Consider the following relational database model:**

product(pid,name,price,category,maker_cid)

purchase(buyer ssn,seller ssn,quantity,pid)

company(pid,name,stock-price,country)

person(ssn_name,phone,number,city)

a) **Write an SQL query to find the names of all Japanese companies which sell products of "Computer"category.**

b) **Write an SQL query to create a view to expose only the product id,name,category and marker country.**

c) **Write a query in SQL to decrease the stock price of all markers of"LCD"category products by 1%.**

d) **Write skeleton tables in QBE to find the name and phone number of all persons who said products of "Automobile" category. [2071 Magh]**

⇒ (a) SELECT name FROM

Company JOIN product ON

Product . Pid = company . Pid

WHERE country = "Japan" AND category = "Computer"

(b) CREATE VIEW expose As

SELECT Pid, name, category and

Country FROM

Product JOIN company ON

Product . Pid = company . Pid

(c) UPDATE company

JOIN product ON Product . Pid = company . Pid

SET stock – price = 0.99 * stock price

WHERE category = "LCD"

(d) Here, name = x, Phone number = y

Now, QBE

pid	Name	Phone number	Category
Join	P.X	P.Y	Automobile

19. **Consider the following relational database model:**

employee(<u>employee-name,</u>street,city)

works(<u>employee-name,</u>company-name,salary)

company(<u>company-name,</u>city)

manages(<u>employee-name,</u>manager-name)

Write SQL queries for the following needs:

i) **Modify the database so that Jones now lives in**
Pokhara.

ii) **Give all employees of "Nabil Bank"a 10 percent raise.**

iii) **Give all managers of "Nabil Bank" a 30 percent rai**
unless the salary becomes greater than 100,000.

iv) **Delete employee who has maximum amount of salary.**

[2070 Bhadr

(i) UPDATE employee

SET CITY = "Pokhara"

WHERE employee name = "Jones"

(ii) UPDATE Works

SET Salary = 1.1 * Salary;

(iii) UPDATE Works

JOIN manages ON Works employee . name = manages
employee . name SET Salary = 1.3 * Salary WHERE Salary <
= 100000,

(iv) DELETES FROM employee JOIN works ON employee
employee – name = works . employee – name WHERE Salary
= Max (Salary)

Consider the following relational database model:

Product(<u>pid</u>,name,price,category,cid)

Purchase(buyer-ssn,seller-ssn,quantity,pid)

Company(<u>cid,</u>name,stock,price,country)

manages(<u>ssn,</u>name,phonenumber,city)

a) **Find the ssn and name of all people who have purchased products of category"telephone".**

b) **List the pid and name of all products which is more expensive than $500 and made in China**

c) **Increase the price of all products of "television"category by 10%**

d) **List the ssn and name of each seller along with the total quantity of products sold.** **[2070Magh]**

⇒ (a) $\pi_{\text{ssn, name}} (\sigma_{\text{category}} = \text{"telephone' (Product} \bowtie_{\text{product . Pid = purchase . Pid}} \text{purchase} \bowtie_{\text{purchase . buyer – ssn = person . buyer – ssn}} \text{person})$

(b) $\pi_{\text{pid, name}} (\sigma_{\text{pirce > 500 AND Country = 'China'}} (\text{Product} \bowtie_{\text{product . Cid = company . Cid}} \text{company}))$

(c) $\pi_{\text{price * 1.5}} (\sigma_{\text{category = "Television"}} \text{product})$

(d) $\pi_{\text{ssn, name, COUNT (quantity)}} (\text{Person} \bowtie_{\text{person, ssn = purchase . seller – ssn}} \text{Purcase}))$

21. **Consider the relational schema given below** **[2×4] [2070 Magh]**

Hotel (<u>Hotel No</u>, Name, Address)

Room (<u>Room No, Hotel No</u>, Type, Price)

Booking (<u>Hotel No, Guest No, Date From,</u> Date_To, Room_No)

Guest (<u>Guest No</u>, Name, Address)

a) **Write an SQL query to list all guests who have booked rooms at the Himalayan hotel.**

b) **Write an SQL query to create view to expose only the Hotel_No,Guest_No,Room_No,and Price of the room of all booked rooms.**

c) **Write a query to offer 5% discount on all rooms of types"Delux"for the Everest Hotel.**

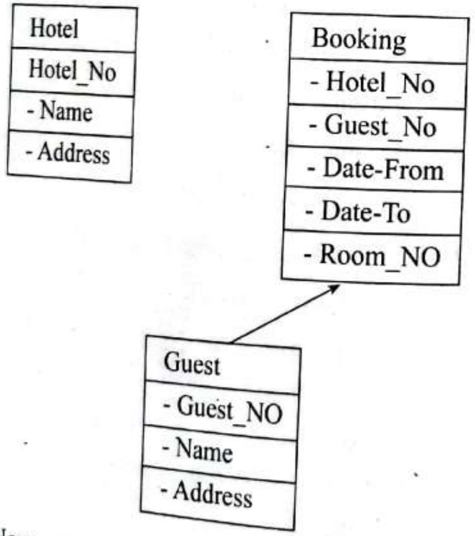
d) **Write skeleton tables in QBE to find the Check-in date and Name of all guests currently booked for Everest hotel.**

⇒ (a) SELECT * FROM GUEST JOIN BOOKING ON GUEST.
Guest no. = Booking . Guest NO JOIN Hotel ON Hotel . Hotel
NO = Booking . Hotel NO WHERE Hotel Name = 'Himalayan'

(b)
```
CREATE VIEW EXPOSE AS
SELECT Hotel – NO, Guest – NO, Room – NO,
Hotel . Hotel – NO = Room . Room – NO JOIN
Booking ON Room . NO = Booking . Hotel – NO
JOIN Guest all Guest . Guest – NO = Booking
Guest – NO WHERE Room . Type = 'Booked'
```

(c)
```
UPDATE ROOM
SET CASE = Price
WHEN TYPE = "Delux"
THEN
Price = 0.95 * price
END
```

(d)



Now,

Selection

Here,

p → display

x → variable

y → variable

Thus,

Guest – NO	Name	Date – From
Join	P.X	P.Y

22. **employee(<u>empname</u>, street, city)** **[4×2] [2069 Bhadra]**

works(<u>empname</u>, companyname, salary)

company(<u>companyname</u>, city)

manages(<u>empname</u>, managername)

For the case of above database schema:

(i) **Write an expression in SQL to create the table employee.**

(ii) **Write an expression in SQL to inset a row into the table works.**

(iii) **Write an expression in SQL to find the name and cities of resident of all the employees who do not work for XYZ Pvt. Ltd.**

(iv) **Write an expression Relational Algebra to find the company name that has the highest number of employees.**

(i)
```
CREATE TABLE employee (
                empname varchar (255)
                street varchar (255)
                city varchar (255)
```

(ii)
```
INSERT INTO WORKS (empname, company name, salary)
VALUES ('XYZ', 'ABC', '$4500')
```

(iii) SELECT * FROM employer join works ON employer. emphane = Works . empname WHERE company name ! = "XYZ pvt. Ltd"

(iv) H1: Count (Empname) (Works)

H2: Max (Count) (H1)

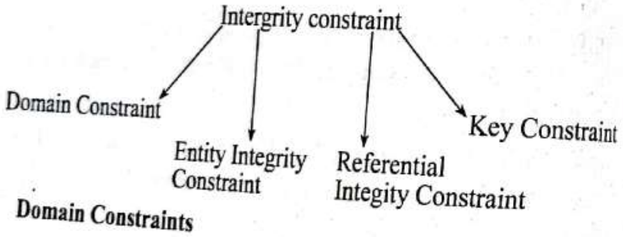
Ans: $\pi_{\text{company name}}$ (Company $\bowtie_{\text{works . company name = employee . company name}}$ H1)

CHAPTER 4 DATABASE CONSTRAINTS AND NORMALIZATIO

Integrity Constraints

- Integrity constraints are set of rules. It is used to maintain the qu of information.
- Integrity constraints ensure that the data deletion, updating and o processes have to be performed in such a way that data integrity is affected.
- Thus, integrity constraints is used to guard against accidental dam to the database.

Types of Integrity Constraints



1. **Domain Constraints**

- Domain constraints can be defined as the definition of a val set of values for an attribute.
- The data type of domain in includes string, character, intege time, date, currency etc. The value of the attribute must b available in the corresponding domain.

E.g:

ID	Name	Semester	Age
1	ABC	1	17
2	XYZ	2	18
3	MNO	5	20
4	PQR	8	A

Not allowed Because AGE is an integer variable.

Entity integrity constraints

- The entity integrity states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify the rows.
- A table can contain a null value other than the primary.

E.g:

Emp _ id	Emp_name	Salary
123	Jack	30000
456	John	40000
146	Harry	5000
	Jonila	15000

Not allowed as primary key can't contain a Null value.

3. **Referential Integrity Constraints**

- A referential integrity constrains is specified between two tables.
- In the referential integrity constraints, if a foreign key in table 1 refers to the primary key of Table 2, then every value of the foreign key in Table 1, must be null or be available in Table 2.

Table 1

Emp _ Name	Name	Age	D - No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

Not allowed as D - No 18 is not defined as a primary key or table 2 and in table 1. D - No is a foreign key defined.

Table 2

D - No	D - location
11	Mumbai
14	Delhi
13	Noida

4. **Key constraint**

- Keys are the entity set that is used to indentify an entity within its entity set uniquely.
- An entity set can be multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

ID	Name	Age
1000	Tom	12
1001	Johnson	14
1002	Suman .	20
1003	Rahul	21
1002	Diwash	22

Not allowed because all row must be unique

Trigger

A trigger is a stored procedure in database which automatically im whenever a special event in the database occurs. For example, a trigger be invoked when arrows is inserted into a specified table or when ce table columns are being updated.

Syntax:

```
Create trigger [trigger _ name]
[before/after]
{insert/update/delete}
on [table _ name]
[For each low]
[trigger _ body]
```

.g: Given student report database, in which student marks assessment ecorded. In such schema, create a trigger so that the total and average ecified marks is automatically inserted whenever a record is insert.

ere, as trigger will invoke before record is inserted so, BEFORE tag is used

L Trigger to problem statement:

```
eate trigger stud _ marks
ore INSERT

ent
ach row
udent . total = student . subject + student . subject + student . subject 3
t . per = student total * 60/100;
```

ssertions

n assertions is a predicate expressing a condition we wish the database to lways satisfy. Domain constraints, functional dependency and referential ntegrity are special forms of assertion.

Where a constraint cannot be expressed in these forms, we use an assertion

.g:

- Ensuring the sum of loan amounts for each branch is less than the sum of all account balances at the branch.
- Ensuring every loan customer keeps a minimum of $1000 in an account.

Syntax:

Create assertion _ name check predicate.

E.g:

Create assertion sum _ constraint check

Functional Dependency

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non - key attribute within a table.

$X \rightarrow Y.$

The left side of FD is known as determinant, the right side of production is known as dependent.

For E.g:

Assume we have an employee table with attributes Emp _ Id, Emp _ Name, Emp _ Address.

Here Emp _ id attributes can uniquely identify the Emp _ Name attribute of employee table because if we know the Emp _ id we can tell that employee name associated with it.

Functional dependency can be written as:

Emp _ id → Emp _ Name

Types of Functional Dependencies

(i) Multi - valued Dependencies

Multi valued dependency occurs in the situation on where there are multiple independent multivalued attributes in a single table.

E.g:

Car _ Model	Year	Color
01	2017	Metalic
02	2018	Green
03	2019	Silver
01	2017	Green
02	2018	Gray
03	2019	Green

In this example, year and color are independent of each othe dependent on car _ model. In this example, these two columns ar to be multivalued dependent on car _ model.

This dependence can be represented

car _ model → year

car _ model → colour

2. **Trivial Functional Dependency**

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

Example:

Consider a table with two columns Employee _ Id and Employe Name.

{Employee _ id, Employee _ Name} → Employee _ Id is a tri functional dependency as

Employee _ Id is a subset of {Employee _ Id, Employee _ Name}

Also, Employee _ Id → Employee _ Id and Employee _ Name → Employee _ Name are trivial dependencies too.

3. **Non - Trivial Functional Dependency**

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

E.g:

ID → Name

Name → DOB

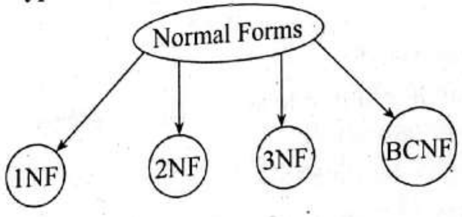
oin Dependency

- Join decomposition is further generalization of multivalued dependencies.
- If the join of R_1 and R_2 over C is equal to relation R, then we can say that a join dependency (JD) exists.
- Where R_1 and R_2 are decompositions R_1 (A, B, C) and R_2 (C, D) of a given relations R (A, B, C, D).
- Alternatively R_1 and R_2 are a lossless decomposition of R.
- A JD $\bowtie$ {R_1, R_2, R_3 ..... R_n} is said to hold over a relation R if R_1, R_2,....... R_n is a lossless - join decomposition.
- The * (A, B, C, D), (C, D) will be a JD of R if the join of join's attributes is equal to the relation R.
- Here, * (R_1, R_2, R_3) is used to indicate that relation R_1, R_2, R_3 and so on are a JD of R.

Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set relations. It is also used to eliminate the undesirable characteristics like insertion, update and deletion anomalies.
- Normalization divides the larger table into the smaller table and likes them using relationship.
- The normal form is used to reduce redundancy form the database table.

There are four types of normal forms:



Normal		Description
1.	1NF	A relation is in 1 NF if it contains an atomic value.

2.	2NF	A relation is in 2NF if it is in 1NF and all n key attributes are fully functional dependen the primary key.
3.	3NF	A relation will be in 3NF if it is in 2NF and transition dependency exists.
4.	4NF	A relation will be in 4 NF if it is in Boyce C normal form and has no multi - val dependencies.
5.	5NF	A relation is in 5NF if it is in 4NF and contains any join dependency and joining sho be lossless.

First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that can attribute of a table cannot hold multiple attribute.

E.g: Employee table

Emp _ id	Emp Name	Emp No.
14	John	7892, 5052
20	Harry	2021
12	Sam	7871, 9990

Relation employee is not in 1NF because of multivalude attribute in Emp N

The decomposition of employee table into 1NF is:

Emp _ id	Emp Name	Emp No.
14	John	7892
14	John	5052
02	Harry	2021
12	Sam	7871
12	Sam	9999

Second Normal Form (2NF)

A relation is in 2NF if, and if, it is in 1NF and every non - key attribute fully dependent on the primary key.

Conversion from 1NF and 2NF

- Identify the primary key for the 1NF relation.
- Identify the functional dependencies in the relation.
- If partial dependencies exist on the primary key remove them b placing them in a new relation along with a copy of their determinant.

E.g:

Teacher	Subject	Age
25	Che	30
25	Bio	30
47	Eng	35
83	Math	38
83	Com	38

In the given table, non - prime attribute Age is dependent on Teacher _ ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two parts:

Teacher _ Detail table:

Teacher _ ID	Age
25	30
47	35
83	38

Teacher _ Subject table

Teacher _ ID	Sub
25	Che
25	Bio
47	Eng
83	Math
83	Com

Third Normal Form (3NF)

- A relation will be 3ND if it 2NF and not contain any transitive partial dependency.

A relations is 3NF if it holds at least one of the following conditions for every non - trivial function dependency X → Y.

1. X is super key
2. Y is a prime attribute i.e. each element of Y is part of some candidate key.

E.g:

Employee _ detail table:

Emp _ id	Emp _ Name	Emp _ zip	Emp _ State	Ci
222	Harry	2010	UP	Noi
333	Stephan	0222	US	Bost
444	Lan	600	, US	Chica
555	Katharine	0630	UK	Nowi
666	John	4620	MP	Bhop

Super - Key in the above table:

{EMP - id} {Emp _ id, Emp _ Name}, {emp _ id, emp _ name, em zip}...... so on.

Candidiate key: {Emp _ id}

Non - prime attribute: In the given table, all attributes except emp _ id. non - prime.

Here, Emp _ state and Emp - city dependen on Emp _ Zip and Emp _ I dependent on Emp _ id. The non - prime attributes (Emp _ State, Emp _ Ci transitively dependent on super key (Emp _ ID). It violates the rule of th Norma form.

That's why we need to move the Emp _ city and Emp _ state to new Employee _ Zip> table, with Emp _ Zip as primary key.

Employee Table:

Emp _ id	Emp _ Name	Emp _ Zip
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

Employee _ Zip table:

Emp _ Zip	Emp _ State	Emp _ City
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Boyce - codd Normal Form (BCNF)

- BCNF is the advance version of 3NF.It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS in super key.

E.g:

Employee table

Emp _ id	Emp _ country	Emp _ dept	Dept _ type	Emp _ dept
264	India	Designing	D334	283
264	India	Testing	D394	300
364	UK	Storing	D283	232
364	UK	Developing	D283	549

In the above table functional dependencies are as follows:

Emp _ id → Emp - country

Emp _ Dept → {Dept _ type, Emp _ dept _ no}

Candidate key: {Emp _ id, Emp _ dept}

The table is not in BCNF because neither Emp _ dept nor Emp _ id alone are keys.

To convert the given table into BCNF, we decompose it into the tables:

Emp _ country table

Emp _ ID	Emp _ Country
264	India
264	India

Emp _ Dept table

Emp _ dept	Dept _ type	Emp _ dept _ no
Designing	D394	283
Testing	D394	300
Storing	D283	232
Developing	D283	549

Emp _ Dept _ mapping table

Emp _ ID	Emp _ Dept
D394	283
D394	300
D283	232
D283	549

Functional Dependencies

Emp → ID → Emp _ country

Emp _ Dept → {Dept _ type, Emp _ dept _ no}

Candidate keys

For first table: Emp _ ID

For second table: Emp _ dept

For third table: {Emp _ ID, Emp _ Dept}

Fourth Normal Form (4NF)

- A relation will be in 4NF if it is in Boyce Codd Normal form and has no multi - valued dependency.
- For a dependency A → B, if for a single valued of A, multiple values of B exists,, then the relation will be a multi - valued dependency.

E.g:

STUDENT

Stu _ id	Course	Hobby
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cirket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the course and hobby are two independent entity. Hence, there is no relationship between course and hobby.

In the student relation, a student with stu - id, 21 contains two courses, computer and math and two hobbies, Dancing and Singing. So, there is a nulti - valued dependency on STU _ ID, which leads to unnecessary epetition of data.

So, to make the above table in 4NF, we can decompose it into two tables:

Student _ course

stu _ id	Course
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

Student _ Hobby

stu _ id	Hobby
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Domain Key Normal Form (DKNF)

- DKNF stands for Domain key Normal form requires the database that contains no constraints other than domain constraints and key constraints.
- In DKNF, it is easy to build a database.
- It avoids general constraints in the database which are not clear domain or key constraints.
- The 3NF, 4NF, 5NF and BCNF are special cases of the DKNF.
- It is achieved when every constraint on the relation is logical consequence of the definition.

Decomposition

- When a relation in the relational a model is not appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies and redundancy.

(i) **Lossless Decomposition**

- If the information is not lost from the relation that i decomposed, then the decomposition is lossless.
- The lossless decomposition gurantees that the join of relations will result in the same relation as it was decomposed.

R = (A, B, C)

A	B	C
1	X	a
2	Y	b
3	Z	c

$R_1 = (A, B)$

A	B
1	X
2	Y
3	Z

$R_1 = (A, B)$

B	A
X	a
Y	b
Z	c

$\Pi_{A, B, C} (R_1 \bowtie R_2)$

A	B	C
1	X	a
2	Y	b
3	Z	c

Thus, it is lossless

(ii) **Dependency Preserving**

- It is an important constraint of the database.
- In the dependency preservation, at least one decomposed table must satisfy every dependency.
- If a relation R is decomposed into relation R_1 and R_2 then the dependencies of R must be a part of R_1 or R_2 or must be derivable from the combination of functional dependencies of R_1 and R_2.
- For example, suppose there is a relation R (A, B, C) with functional dependency.

  A → BC

The relational R is decomposed into R_1 (ABC) and R_2 (AD) which dependency preserving because FD A → BC is part of relation R_1 (ABC).

Old Question Solution

1. **What do you by closure of functional dependency? What is trigger in DBMS? Is it or risky to use trigger? Explain (2076 Baisakh)**

⇒ A closure is a set of FDS is a set of all possible FDS that can be derived from a given set of FDS. It is also referred as a complete set of FDS. If F is used to denote the set of FDS for relation R, then a closure of set of FDS implied by F is denoted by F^+.

Let's us consider:

$F = \{A \rightarrow B,$

$B \rightarrow C,$

$C \rightarrow D\}$

$A^+ = ABCD$

Since,

A → A	[Self Determination]
A → B	Given
A → C	Rule 3 Transitively
A → D	Rule 3 Transitivity

$B^+ = BCD$

$C^+ = CD$

$D^+ = D$

∴ The candidate key is {A}

Triggers are the procedures stored in the database that are executed when the content in the database table is modified. To design a trigger we must meet two requirements.

(i) Specify when a trigger is to be executed.

(ii) Specify the actions to be taken when the trigger is executed.

Trigger may be risky sometimes when:

- You use BULK INSERT to insert data into a table, triggers are not fired unless you include the FIRE _ TRIGGERS option in your bulk insert statement. This may lose data consistency.
- Recursive triggers are even harder to debug than nested triggers.
- If there are many nested trigger it could get very hard to debug and troubleshoot, which consumes development time and resources.

2. **Define normalization and levels of normalization 1NF,2NF an 3NF.Compare the advantages of BCNF over 3NF (2076 Baisak**

⇒ First part:see in theory page no.81

(i) 3NF states that no non - prime attribute must be transitivel dependent on the candidate key of the relation on the oth hands, BCNF states that if a trivial functional dependency X → Y exist for a relation then X must be as super key.

(ii) BCNF is much restrictive than 3NF which help. in normalizin the table more.

(iii) The relation in 3NF has minimum redundancy left which i further removed by the BCNF.

3. **Why do we need normalization? Differentiate primary key an foreign key. Differentiate between 3NF and BCNF (2075 Bhadra)**

⇒ Needs of normalization:

(i) Greater overall database organization

(ii) Reduction of redundant data

(iii) Data consistency within the database

(iv) A much more flexible database design

(v) A better handle on database security

	PRIMARY KEY	FOREIGN KEY
1	A primary key is used to ensure data in the specific column is unique.	A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables.
2	It uniquely identifies a record in the relational database table.	It refers to the field in a table which is the primary key of another table.
3	Only one primary key is allowed in a table.	Whereas more than one foreign key are allowed in a table.
4	It is a combination of UNIQUE and Not Null constraints.	It can contain duplicate values and a table in a relational database.
5	It does not allow NULL values.	It can also contain NULL values.

	3NF		BCNF
1.	No non - prime attribute must be transitively dependent the candidate key.	1.	For any trivial dependency in a relation R say X → Y should be a super key of relation R.
2.	3NF can be obtained without sacrificing all dependencies.	2.	Dependencies may not be prepared in BCNF.
3.	Lossless decomposition can be achieved in 3NF.	3.	Lossless decomposition is hard to achieve in BCNF.

4. **Consider the relation Treatment with the schema. Treatment (doctor ID, doctor Name, Patient ID, diagnosis) and functional dependencies:**

doctor ID → doctor Name and (doctor ID, Patient ID) → diagnosis.

Describes different types of problem that can arise for this relation with records. (2075 Bhadra)

⇒ E.g:

doctor ID	doctor Name	Patient ID	diagnosis
001	XYZ	123	Fever
002	ABC	110	Cold
003	PQR	112	Allergy
002	XYZ	121	Fever

Anomalies

(i) **Insertion anomaly**

For treatment we cannot insert the doctor information like doctor ID and doctor Name without an patient. That is, we need at least one patient to include the doctor information into the table. For e.g. if Dr. XYZ is appointed we need to allocate at least one patient to insert Dr. XYZ information.

(ii) **Deletion Anomaly**

Deleting patients diagnosis could delete the name of their doctor. For e.g. Dr. ABC has only one patient 110 registered for him. If we delete the patient 110, we need to delete Dr. ABC details as well. This is deletion anomaly present in the treatment table.

(iii) **Modification Anomaly**

A doctor may have more than one patient, so an upd anomaly may result if a doctor's name is changed for a giv doctor ID for only one patient. For example in our table I XYZ has two patients. Suppose that we need to changes doc name from XYZ we need to OXYZ. This changes has to done on two records.

5. **What is functional dependency? List the various integri constraints and explain about the referential integrity along wi an example. [2075 Baisakh**

⇒ 1st part See in theory page no 79

2nd part see in theory page no 76

6. **Define 1NF,2NFand 3NF.Illustrate your answer with suitab example. [2075 Baisak**

⇒ See in theory page no 82

7. **What are triggers? Define Domain constraint and Referentia Integrity constraint with an example. [2074 Bhadr**

⇒ 1st part see on page no 78

2nd part see on page no 76

8. **What is the role Functional dependencies in Normalization Explain trivial and non-trivial dependencies. Explain BCNF. (2074 Bhadra**

⇒ 1st part see on page no 86

2nd part see on page no 80

3rd part see on page no 85

9. **What do you mean by functional dependencies? What is BCNF? (2073 Bhadra**

⇒ 1st part see on page no 86

2nd part see on page no 85

10. **What is normalization? Explain 1NF,2NF,3NF and 4NF?(2073 Bhadra)**

⇒ See on page no 81

1. **Define functional dependency. Explain partial and transitive functional dependency with example. (2073 Magh)**

› 1st part see on page no 86

Partial Dependency occurs when a non - prime attribute is functionally dependent on part of a candidate key.

E.g:

Student Project

Student IP	Project No.	Student Name	Project Name
501	199	OXY	Geo super
502	200	ZMO	Sony

The prime attributes are student ID and project NO. Non, prime attributes are student Name and project Name should be functionally dependent on part of a candidate key to be partial dependent.

The student Name can be determined by student ID, which makes the relation partial dependent.

The project Name can be determined by project No. which makes the relation partial dependent.

Transitive F.D

A transitive F.D is a type of F.D which happens when it is indirectly formed by two functional dependencies.

E.g:

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Alibaba	Jack Ma	54

Company → CEO

CEO → Age

Therefore accordance to rule of transitive F.D.

Company → Age

12. **Define decomposition and its desirable properties. Explain 3NF and BCNF. (2073 Magh)**

⇒ When a relation in the relational model is not appropriate normal form then the decomposition of a relation is required. In a database, breaking down the table into multiple tables termed as decomposition. The properties of a relational decomposition are listed below :

1. **Attribute Preservation:** Using functional dependencies the algorithms decompose the universal relation schema R in a set of relation schemas D = { R1, R2, ….. Rn } relational database schema, where 'D' is called the Decomposition of R. The attributes in R will appear in at least one relation schema Ri in the decomposition, i.e., no attribute is lost. This is called the *Attribute Preservation* condition of decomposition.

2. **Dependency Preservation:** If each functional depende X->Y specified in F appears directly in one of the rela schemas Ri in the decomposition D or could be inferred f the dependencies that appear in some Ri. This is *Dependency Preservation.*

If a decomposition is not dependency preserving so dependency is lost in decomposition. To check this conditi take the JOIN of 2 or more relations in the decomposition.

For example:

R = (A, B, C)

F = {A ->B, B->C}

Key = {A}

R is not in BCNF.

Decomposition R1 = (A, B), R2 = (B, C)

R1 and R2 are in BCNF, Lossless-join decompositio Dependency preserving. Each Functional Dependen specified in F either appears directly in one of the relations i the decomposition.

It is not necessary that all dependencies from the relation appear in some relation Ri.

It is sufficient that the union of the dependencies on all th relations Ri be equivalent to the dependencies on R.

3. **Non Additive Join Property:** Another property o decomposition is that D should possess is the *Non Additi Join Property*, which ensures that no spurious tuples ar generated when a NATURAL JOIN operation is applied t the relations resulting from the decomposition.

4. **No redundancy:** Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies and redundancy. If the relation has no proper decompositio then it may lead to problems like loss of information.

5. **Lossless Join:** Lossless join property is a feature of decomposition supported by normalization. It is the ability to ensure that any instance of the original relation can be identified from corresponding instances in the smaller relations.

For example: R : relation, F : set of functional dependencies on R, X, Y : decomposition of R, A decomposition {R1, R2, ..., Rn} of a relation R is called a lossless decomposition for R if the natural join of R1, R2, ..., Rn produces exactly the relation R.

A decomposition is lossless if we can recover: R(A, B, C) -> Decompose -> R1(A, B) R2(A, C) -> Recover -> R'(A, B, C) Thus, R' = R. Decomposition is lossless if: X intersection Y -> X, that is: all attributes common to both X and Y functionally determine ALL the attributes in X. X intersection Y -> Y, that is: all attributes common to both X and Y functionally determine ALL the attributes in Y If X intersection Y forms a superkey of either X or Y, the decomposition of R is a lossless decomposition.

2nd part see on page no 83

13. **Suppose that we decompose the schema R = (A, B, C, D, E) into (A, B, C) and (C, D, E). Is it lossless decomposition? It is dependency preserving? Consider that following set F of functional dependencies hold.**

A → BC

CD → E

B → D

E → A

(2072 Ashwin)

⇒

	A	B	C	D	E
R_1	α_A	α_B	α_C	B_{1D}	B_{1E}
R_2	B_{2A}	$B\alpha_B$	α_C	α_D	α_E

There are not any rows containing all α. So, it is not lossless and not dependency preserving.

14. **Suppose that we decompose the scheme R = (A, B, C) into R_1 = (A, B), R_2 = (A, C). Show that decomposition is a lossless join decomposition and not dependency preserving if the F = { A → B**

B → C

}

(2072 Magh)

⇒ To find lossless or not

	A	B	C
R_1	α_A	α_B	B_{1C}
R_2	α_A	α_B	α_C

Now, using B → C have all α in column.

So,

Now, it is lossless

	A	B	C.
R_1	α_A	α_B	α_C
R_2	α_A	α_B	α_C

Now, to find functional dependent or not

R_1 (A, B)	R_1 (A, B)
$A \rightarrow B$	$A \rightarrow C$ x
$B \rightarrow A$ x	$C \rightarrow A$ x

$R_1 \cup R_2 = A \rightarrow B \neq R$

So, it is not functional dependent preserving.

R_1 (A, B, C)	R_2 (A, B, C)
$A \rightarrow B$, $B \rightarrow A$	$C \rightarrow D$, $C \rightarrow E$, $D \rightarrow C$,
$AB \rightarrow C$, $C \rightarrow AB$	$E \rightarrow D$, $\boxed{CD \rightarrow E}$
$AC \rightarrow B$, $B \rightarrow AC$	$E \rightarrow CD$, $EC \rightarrow D$,
$BC \rightarrow A$, $\boxed{A \rightarrow BC}$	$D \rightarrow EC$, $DE \rightarrow C$,
$A \rightarrow C$, $C \rightarrow A$	$D \rightarrow DE$

$R_1 = A \rightarrow BC$

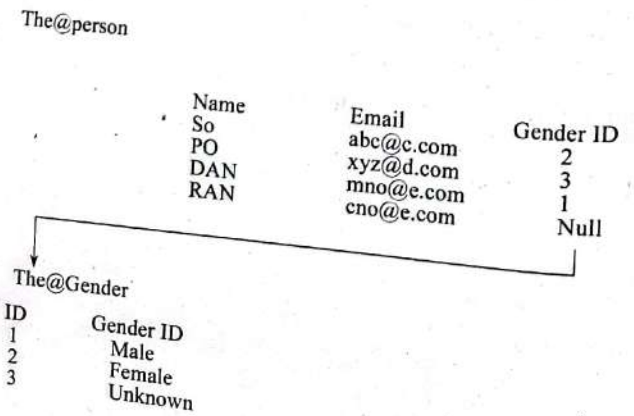
$R_2 = CD \rightarrow E$

$R_1 \cup R_2 \neq R$

So, not dependency preserving

15. **Explain cascading actions in refrential integrity constraints.** **(2071 Bhad**

⇒



The@person

Name	Email	Gender ID
So	abc@c.com	2
PO	xyz@d.com	3
DAN	mno@e.com	1
RAN	cno@e.com	Null

The@Gender

ID	Gender ID
1	Male
2	Female
3	Unknown

Cascading refrential integrity constraint allows to define the actions Microsoft SQL server should take when or user attempts to delete or update a key to which an existing foreign keys points.

For example, If you delete row with ID = 1 from the @ gender table, then row with ID = 3 From The @ person table becomes an orphan record. You will not be able to tell the gender for this row. So, cascading refrential integrity constraint can be used to define actions Microsoft SQL. Server should take when this happens. By default, we get an error and the DELETE or UPDATE Statement us rolled back.

16. **Explain the necessary condition for decomposing a relational database table into two table** **(2070Magh)**

⇒ (i) **Attribute Preservation**

The attribute in R will appear in at least one relation schema R_i in the decomposition i.e. no attribute is lost.

(ii) **Dependency Preservation**

If each functional dependency $X \rightarrow Y$ specified I F appears directly in one of the relation schema R_i in the decomposition D or could be inferred from the dependencies that appear in some R_i.

(iii) **Non - Additive join property**

This ensures that no spurious tuples are generated when a NATURAL JOIN operation is applied to relations resulting from the decomposition.

(iv) **NO Redundancy**

Decomposition is used to eliminate some of the problems of bad design like anomalies, in consistencies and redundancy.

(v) **Lossless Join**

Lossless join property is a feature of decomposition supported by normalization. It is the ability to ensure that any instance of the original relation can be identified from corresponding instances in the smaller relations.

17. **When database de - normalization is preferred?** **(2069 Bhadra)**

⇒ Conditions:

(i) Retrieving data is faster since we do fewer joins.

(ii) Queries to retrieve can be simpler since we need to look at fewer tables.

□□□

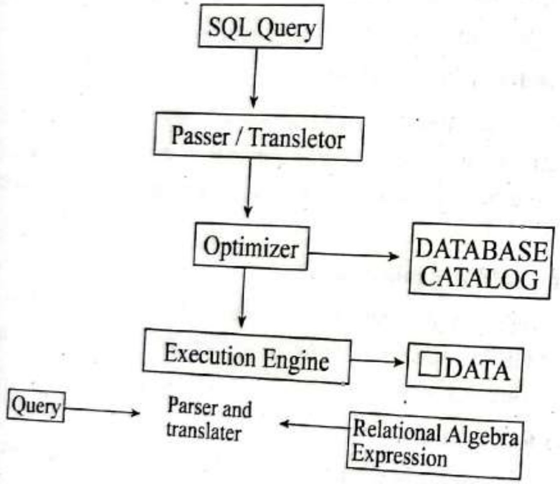
CHAPTER 5

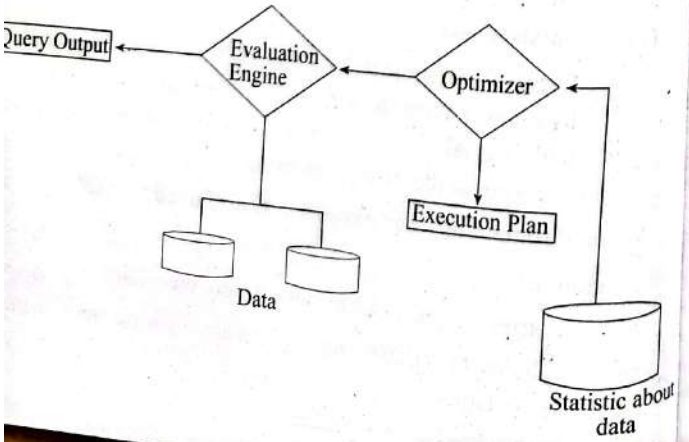
QUERY PROCESSING AND OPTIMIZATION

Query processing is the activity performed in extracting data from data It takes various steps for fetching the data from the database. The involved are:

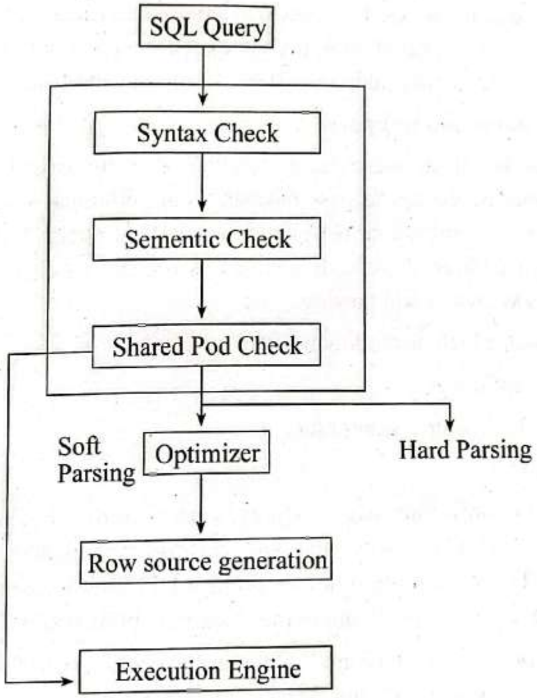
1. Parsing and transaction
2. Optimization
3. Evaluation

Block Diagram of Query processing is as:





Detailed Diagram is drawn as:



It is done in the following steps:

- step 1:

Parser: During parse call, the database performs the following check.

(i) syntax check

(ii) semantic check

&(iii) shared pool check, after converting the query into relational algebra.

parser performs the following checks as:

(1) syntax check - concludes SQL syntactic validity example.

SELECT * form EMPLOYEE

Here error of wrong spelling of FROM is given by this check.

2. Semantic check - determines whether the statement is meaningful or not.

Example: Query contains a table name which does not exist is checked by this check.

3. Shared pool check - Every query possess a has code during execution, so, this check determines existence of written code in shared pool, if code exists in shared poll then datab will not take additional steps.for optimization and execution.

Hard parse and soft parse

If there is a fresh query and its has code does not exist in shared p then that query has to pass through from additional steps known hard parsing otherwise if has code exists then query does not pass through additional steps. It just passes directly to execution engi This is known as soft parsing.

Hard parse includes following steps-

(i) optimizer

(ii) Row source generation.

Optimizer -

During optimization stage, database must perform a hard parse at lea one unique DML statement and perform optimization during th parse. This optimization never optimize DPL unless it includes DM component such as sub query that required optimization.

It is a process in which multiple query execution plan for satisfying query are examined and mot efficient query plan is satisfied fo execution.

Database catalog stores the execution plans and then optimizer parse the lowest cost plan for execution.

Row source Generation-

The Row Sources Generation is a software that receives a optimal execution plan from the optimizer and produces and an iterative execution plant that is usable by the rest of the database. The database. The iterative plan is the binary program that when executed by the sql engine produces the result set.

Step 3:

Finally runs the query and display the required result.

Query Cost Estimation

The query evaluation cost is dependent on different resources:

(i) Disk accesses

(ii) CPU time to execute the query cost

(iii) Cost of data transmission (for distributed/parallel system)

(i) Disk accesses

The cost to access data from disk is most important for large database since the disk accesses are slow compare to main memory access

(ii) CPU time

Time to execute the query code.

Generally disk access time>>CPU time, hence we ignore cpu time and consider only disk access costs to measure the query evaluation cost.

Disk access: To measure the cost of data access from disk we consider :

Number of block transfer and number of disk seeks form disk.

$t_{T:}$ Avg. time to transfer a single block of data (in sec)

ts: Avg. block access time (disk seek) plus rotation latency (in sec) let go of blocks be $b\ell$ and no of seeks be se, then

The cost = ($bl*l_T$ + Se & t_T) sec moreover, block access = block read + block write

$t_{br} < t_{bw}$

The cost of all the algorithm that we consider deepens on the size of the buffer memory.

Best case: when all data (all tables) reside into main memory.

Worst case: store few blocks or parts of the block and were Ned many swaps.

Practically, actual disk access < estimated disk cost. Since we consider worst cases most of the time.

Query operation

1. Selection operation

File block and test all records to see whether they satisfy the selection condition.

→ Cost estimate = b_r block transfers + 1seek

br denotes number of block containing records from relation r.

→ If selection is an key attribute, can stop ion finding record. cost = (br/2) block transfer = 1 seek

→ linear search can be applied regardless of

(i) selection condition, or

(ii) ordinary of records in the file or

(iii) availability of indices.

(b) A2 (binary search)

If the file is ordered on an attribute, and the selection condi kg an equally comparison on the attribute, we can use a binary sea on the blocks of the file.

(c) Index scan

Search algorithm that use index.

→ search - key of index.

(d) A3 (primary index, equality on key)

Retrieve a single record that satisfies the corresponding equal condition.

→ cost = $(h_i + 1) * (t_T + t_s)$

(e) A_4 (secondary index, equality on non key)

- Retrieve a single record if the search key is a candidate key.

  → cost = $(h_i + 1) * (t_T + t_s)$

- Retrieve multiple record if search - key is not a candidate key

  → each of n matcxhing records may be on a different block

  → cot = $(h_i + n) * (t_T + t_s)$

  → can be very expensive

(f) A5/primary index, comparison)

(g) A7 (conjuctive selection using one index)

(h) A10 (disjunctive selection union of identifier)

) · **Sorting**

It is the technique of sorting the record in ascending or descending order of one or more columns. It is useful because some of the queries will ask us to return sorted records or in operations like joins will be more efficient in sorted records.

(i) **Quick sort**

Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value

Quick sort pivot algorithm

Step 1 - choose the highest index value has pivot.

Step 2 - Take tow variables to point left and right of the list excluding pivot

Step 3 - left points to the low index

Step 4 - right points to the high

step 5 - while value at left is less than pivot move right.

step 6 - while value at right is greater than pivot more left.

step 7 - If both step 5 and step 6 does not match swap left and right

step 8 - If left ³ right, the point where they met is new pivot.

Step sort algorithm

step 1 - Make the right-most index value pivot

step 2 - partition he array using pivot value.

step 3 - quick sort left partition recursively.

sep 4 - quick sort partition recursively.

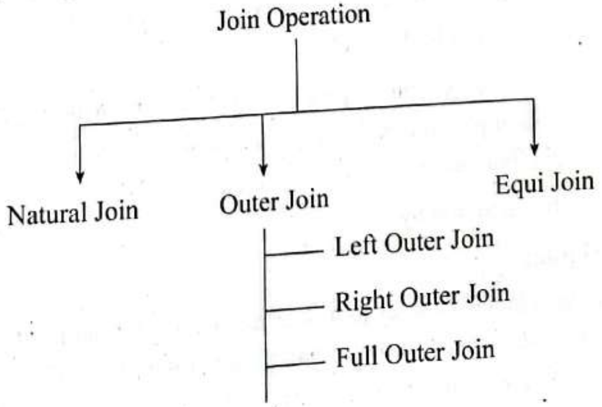
(ii) **Merge sort**

For the larger tables which cannot be accommodated in the current memory, this type of sorting is used. It has better performance compared to quick sort.

(3) **Join operation**

A join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ∞.

Types of join operation.



1. **Natural join:**

- A natural join is the set of tuples of all combination in R and that are equal on their common attribute names.
- It is denoted by ∞

2. **Outer join:**

The outer join operation is an extension of he join operation, I is used to deal with missing information.

An outer join is basically of three types.

(a) Left outer join

(b) Right outer join

(c) Full outer join

(a) Left outer join:

- Left outer join contains the set of tuples of all combinations in R and S that are equal or their common attributes names.
- In the left outer join, tuples in R have no matching tuples in S.
- It is denoted by ∞.

(b) Right outer join:

- Right outer join contains the set of tuples of all combination in R and S that are equal on their common attribute names.
- In right outer join, tuples in S have no matching tuples in R.
- It is denoted by ∞.

(c) Full outer join:

- Full outer join is like left or right join except that in contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuple sin S have no matching in R in their common attribute name.
- It is denoted by ×.

Equi join:

t is also known as inner join. It is the most common in. It is based on atched data as per the equality condition the qui join uses the mparison operator ⇔ (=)

Evaluation of Expression

For evaluating an expression that carries multiple operations in it, we can perform the computation of each operation one by one. However, in the query system, we use to methods for evaluating an expression carrying multiple operations. These methods are:

1. Materialization
2. Pipelining

1. **Materialization**

In this method, the given expression evaluates are relational operation at a time. Also, each operation is evaluated in an appropriate sequence or order. After evaluating all the operations, the outputs are materialized in a temporary relation for their subsequent uses. It leads the materialization method to a disadvantage.

The disadvantage is that it needs to construct those temporary relations for materializing the results of the evaluated operations, respectively. These temporary relations are written on the disks unless they are small in size.

2. **Pipelining**

Pipelining is an alternate method or approach to the materialization method. In pipelining, it enables us to evaluate each relations operation of the expression simultaneously in a pipeline. In this approach, after evaluating one operation, its output is passé don to the next operation, and the chain continues till all the relational operations are evaluated thoroughly. Thus, there is no requirement of storing a temporary relation in pipelining. Such an advantage of pipelining makes it a better approach as compared to the approach used in the materialization method.

Even the costs of both approaches can have subsequent difference in-between. But, both approaches perform the best role in different cases. Thus, both ways are feasible at their place.

Query optimization

There are two method of query optimization.

1. **Cost based optimization (physical)**

   This is based on the cost of the query. The query can use different path based on indexes, constraints, sorting method etc. This kind mainly uses the statistics like record size, number of records, number of records per block number of blocks, table size whether whole table first in a block, organization of tablės, uniqueness of column value, size of column etc. Even the costs of both approaches can have subsequent difference in-between. But, both approaches perform the best role in different cases. Thus, both ways are feasible at their place.

2. **Heuristic optimization (logical)**

   This method is also known as ruled based optimization. This is based on the relational expressions, hence the number of combination of queries get reduces here. Hence the cost of the query too reduces.

   These algorithm have polynomial time and space complexity which is lower than the exponential complexity of exhausted search-based algorithms. However these algorithm do not necessarily produce the best query plan.

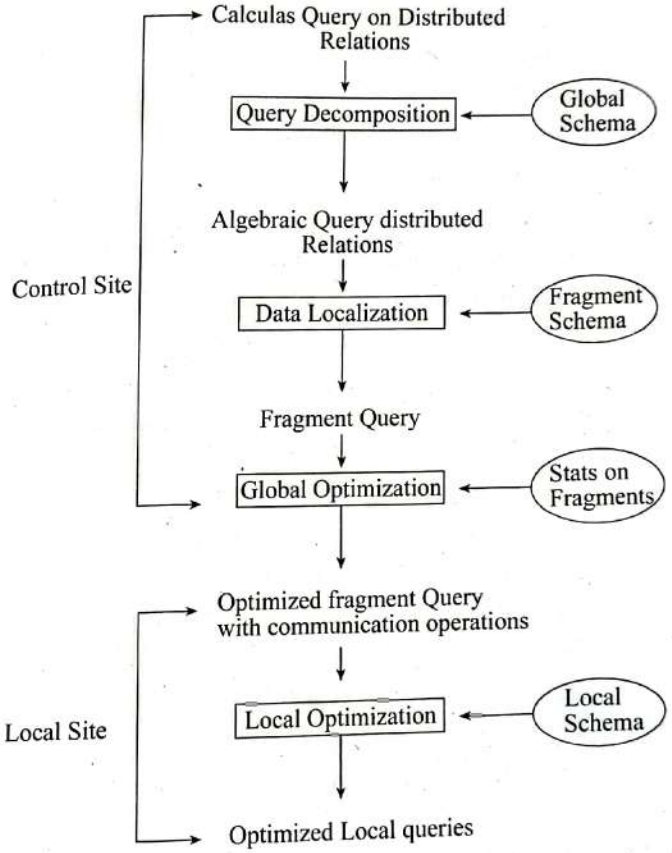
Some of the common heuristic rules are -

- Perform select and project operations before join operations. This is done by moving the select and project operations down the query tree. This reduces the number of tuples available for join.
- Perform the most restrictive select/project operations at first before the other operations.
- Avoid cross-product operation since they result in very large-sized intermediate tables.

Query decomposition

- The query decomposition is the first phase of query processing whose aims are to transform a high-level query into a relational algebra query and to check whether that query is syntactically and semantically correct.
- Thus, a query decomposition phase starts with a high-level query and transforms into a query graph of low-level operations (algebraic expressions), which satisfies he query.

Distributed Query Processing Step



- Mapping of calculus query (SQL) to algebra operations/select, projects join, rename).
- Both input and output queries refer to global relations, without knowledge of the distribution of data.
- The output query is semantically correct and good in the sense that redundant work is avoided.

Steps of Query decomposition

Query decomposition consists of 4 steps.

1. Normalization - Transform query to a normalized form.
2. Analysis: Detect and reject incorrect queries, possible only for a subset of relational calculus.
3. Elimination of redundancy: Eliminate redundant predicates.
4. Rewriting: Transform query to RA and optimize query DDB.

Step i: Normalization

- Normalization refers to transform the query to a normalized form facilitate further processing.
- It consist of mainly two step:

i) Lexical and syntactic analysis

ii) Put into Normal form

→ Lexical and syntactic analysis:

- Check validity (similar to compilers)
- Check for attributes and relations.
- Type checking on the qualification.

Step ii: Analysis

- It aims to indentify an reject type incorrect or semantical incorrect queries.
- Type incorrect:
- Check whether the attribute and relation names of a query a defined in the global schema.
- Check whether the operation on attributes do not conflict wi the types of the attributes,

e.g. a comparison > operation with an attribute of type string.

- Semantically incorrect

→ Check whether the components contribute in any way to th generation of the results.

→ Only a subset of relational calculus queries can be tested fo correctness, i.e. those that do not contain disjunction an negation.

- Typical data structure used to detect the semantically incorre queries are:

1. Conjunction graph (query graph)

2. Join graph

Step iii:-

Elimination of Redundancy

- Elimination of redundancy:

Simplify the query by eliminate redundancies, e.g., redundant predicates

- Redundancies are often due to semantic integrity constraint expressed in the query language e.g. queries on view are expanded into queries on relations that satisfy cetin integrity and security constraints.
- p∧p·p • P∧P⇔false
- p∨p⇔p • P•∧∨⌉P⇔true
- p⊥∧(PivPz) ÛP^
- p⊥∨ (P⊥∧Pz)ÛP1
- It convert relational calculus query to relational algebra query an find an efficient expression.
- Transform query to RAQ and optimize query.

Old Question Solution

1. **Explain the basic steps in query processing with diagram? What is pipelining in query evaluation. Explain with an example.**

[5+3] [2076 Baisakh]

⇒ 1st part See in theory Page No. 98

2nd part see in theory Page No.105

2. **What is the task of evaluation engine in query processing? Explain cost based query optimization and Heuristic optimization.**

[4+4] [2075 Baisakh]

⇒ Query evaluation engine is important part of SQL structured query language) because all the query evaluated in SQL with help of query evaluation engine it executes low - level instruction generates by complier and provides specific output.

For Second Part: See in Theory 106

3. **Explain with diagram about process of query processing in RDBMS.** **[5] [2075 Bhadra]**

⇒ Consider the relational algebra expression for the query "Find the names of all customers who have an account at any branch located at KTM"

Then relational algebra for this is:

$\Pi_{cust_name}(\sigma_{branch_city} = $ "KTM"(branch account depositor)

This expression constructs a large intermediate relation b account depositer. By reducing the number of tuples of the br relation that we need to access, we reduce the size of the intermed result. Then the query is now represented by the relational alg expression.

$\Pi_{customer\_name}(\sigma_{branch\_city = "KTM"}(branch)(account\ depositer))$

This is equivalent to our original algebra expression but wh generates smaller intermediate relations. This can be depicted by diagram which is as follows:

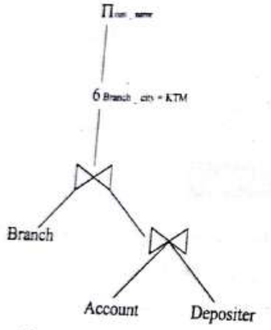


Fig (a) Initial Expression

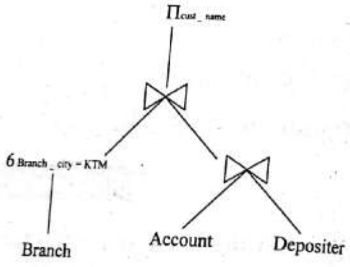


Fig (b) Transforred expression

4. **How are equivalence rules for relational algebra helpful for quer optimization? Explain with example.** **[3] [2075 Bhadr**

⇒ See in internet

5. **Explain about the steps involved in query optimization. How i pipelining approach different from the materialization approach?** **[3+5] [2074 Bhadr**

⇒ See in theory page no 106

The differences between pipelining and materialization approach ar as follows:

Pipelining	Materialization
1. It is a modern approach to evaluate multiple operations.	1. It is a traditional approach to evaluate multiple operations.
2. It does not use any temporary relations for storing the results of the evaluated operations.	2. It uses temporary relation for storing results of the evaluated operations.
3. It is a more efficient way of query evaluation as it quickly generates the results.	3. It is less efficient as it takes time to generate the query results.
4. Poor performance if trashing occurs.	4. No trashing occurs in materlization.
5. It optimizes the cost of query evaluation.	5. The overall cost includes the cost of operations plus the cost of reading and writing. results on the temporary storage.

6. **How can you optimize the following query?** **[5] [2071 Bhadra]**

$\pi_{name,\ title}(\sigma_{dept\_name = "MUSIC"}(Instructor \bowtie \pi_{course\_id,\ title}(Teaches \bowtie course)))$

⇒ $\pi_{name,\ title}(\sigma_{dept\_name = "MUSIC"}(Instructor \bowtie \pi_{course\_id,\ title}(Teaches \bowtie course)))$

Here, dept _ name is a field f only the instructor table, Hence, we can select out the music instructor before joining the tables, hence reducing query time.

Optimized Query

Rule

$\sigma_{\theta 1 \wedge \theta 2}(E_1 \bowtie {}_{\theta}E_2) = (\sigma_{\theta 1}(E_1)) \bowtie_{\theta} (\sigma_{\theta 2}(E_2))$

Performing the selection as early as possible reduces the size of the relation to be joined.

$\pi_{name,\ title}((\sigma_{dept\_name = "Music"}(instructor) \bowtie ({}_{teacher\ ?} \bowtie \pi_{course\_id,\ title}(course))$

7. **How can pipelining approach improve query evaluation efficiency?** **[4] [2070 Bhadra]**

⇒ Pipelining helps in improving the efficiency of the query evaluation b decreasing the production of a number of temporary files. Actually we reduce the construction of the temporary file by merging the multiple operations into a pipeline. The result of one currently executed operations passes to the next operation for its execution, and the chain continues till all operations are completed, and we get the final output of the expression. If we combine the root operator of a query evaluation plan in a pipeline with its inputs, the process of generating query results becomes quick. As a result, it is beneficial for the users us they can view the results of their queries as soon as outputs get generated.

CHAPTER 6 FILE STRUCTURE AND HASHING

File

A file is name collection of related information that is recorded on second storage such as magnetic disks, magnetic tables and optical disks.

Records

The database is stored as a collection of files. Each file is a sequence records. A record is a sequence of fields.

Logical Structure of a file

There are two things which makes a file.

(i) Field

Smallest (Fixed) individual logical of file. A filed holds a part of som data value.

(ii) Record

A set of logically related fields. Size of the record may be fixed o variable size.

A field
SSN
Name
Age
Phone
A record

SSN	Name	Age	Phone

wo types of record:

- Fixed - length records
- Variable - length records

ced - Length Records

ixed length record is one where the length of the fields in each record has n set to be a certain maximum numbers of characters long.

Table Name

Type deposit = record

account _ number: char (10);
Branch _ name: char (22);
Balance: real;

Let assume

1 char = 1 byte
1 Real = 8 bytes

So our one records becomes

Account _ Numb char 10 × 1 = 10 byte
Branch _ Name char 22 × 1 = 22 byte
Balance Real 8 × 1 = $\frac{8 \text{ bytes}}{40 \text{ bytes}}$

Then we can follow an approach

→ Use a first 40 bytes for the first record.
→ The next 40 bytes for the second record.

But there are two problems with this simple approach.

1. Deleting the record from this will create an empty block which may occur waste of memory space. So to overcome this problem we must fill the block with record of the file, or we must have way of marking deleted records so that can be ignored.

A - 101	Perryridge	400
A - 302	Round Hill	350
A - 101	Mianus	700
A - 201	Downtown	500
A - 218	Red wood	900
A - 420	Perryridge	750

Approaches for deletion

(i) Move records i + 1 …… n to I, ……. n - 1(Requires more access to move than record)

(ii) Don't move records, but link all free records on a free list.

(iii) Store the address of the first deleted record in the file header.

(iv) This first records to store the address of the second deleted record and so on.

Variable - Length Records

Variable length records arise in database systems in several ways:

- Storage of multiple records systems in a file.
- Record types that allow variable length for one or more fields.
- Record types that allow repeating fields (used in some older data models)

Mainly two approaches

1. Byte string Representation
2. Fixed - length Representation

Byte - string Representation

- Attach a special end - of - record (1) symbol to the end o record.
- Disadvantage:
  - Not easy to reuse space occupied by deleted record.
  - No space for the records to grow longer. If the records becom longer it must be moved.
- Modified form of byte - string representation on called slotted pa structure.
- Slotted page header contains:
  - Number of record entries
  - End of free space in the block
  - Location and size of each block

0	Perryridge	A - 102	400	A - 201	900	A - 218	700	1
1	Brend Hill	A - 305	350	1				
2	Mianus	A - 215	700	1				
3	Downtwon	A - 103	500	A - 103	600	1		
4	Red wood	A - 222	700	1				
5	Brignton	A - 217	750	1				

Fig: Byte - string Representation

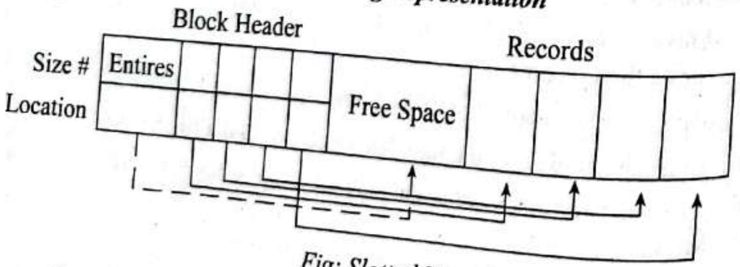


Fig: Slotted Page Structure

Records can be moved around within a page to keep them contiguous with no empty space between them, entry in the header must be updated.

Pointers should not point directly to record - instead they should point to the entry for the record in header.

Fixed - Length Representation

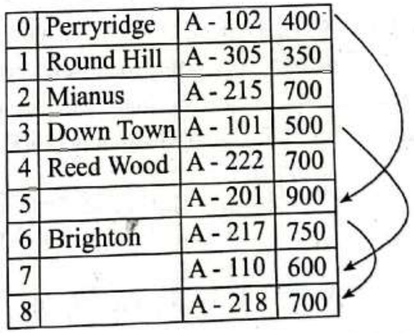
- Use one more fixed length records:

1. **Reserved Space**

   Can use fixed - length records of a known maximum length, unused space in shorter records filled with a null.

0	Perryridge	A - 102	400	A - 201	900	A - 218	700
1	Round Hill	A - 305	350	1	1	1	1
2	Mianus	A - 215	700	1	1	1	1
3	Downtwon	A - 103	500	A - 110	600	600	1
4	Red word	A - 222	700	1	1	1	1
5	Brignton	A - 217	750	1	1	1	1

2. **List representation represent by a list of fixed. Length records chained together.**



0	Perryridge	A - 102	400
1	Round Hill	A - 305	350
2	Mianus	A - 215	700
3	Down Town	A - 101	500
4	Reed Wood	A - 222	700
5		A - 201	900
6	Brighton	A - 217	750
7		A - 110	600
8		A - 218	700

- Disadvantage to pointer structure; space is wasted in all records except the first in a chain.
- Solution is to allow two kinds of blocks in a file.
  - Another block: Contains the first records of chain.
  - Overflow: Contains records other than those that are the first records of chairs.

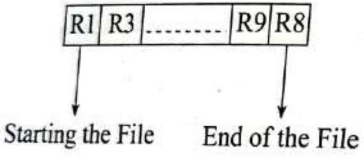
Records Organization in a File

(i) Sequential File organization

This is the simplest method for file organization. In this method, files are stored sequentially. This method can be implemented in two ways:

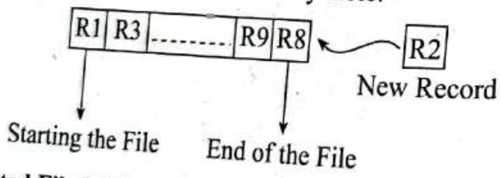
1. **File method**

- It is a quite simple method. In this method, we stor record in a sequence, i.e. one after another. Here record will be inserted in the order in which they inserted into tables.
- In case of updating or deleting of any record, the rec will be searched in the memory blocks. When it found, then it will be marked for deleting, and the n record in inserted.



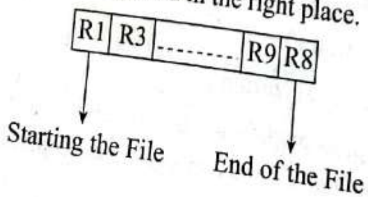
Insertion of the new record

Suppose we have four records R_1, R_2, and so on upto R_9 and l in a sequence. Hence, records are nothing but a row in th table. Suppose we want to insert new record R_2 in th sequence, then it will be placed at the end of the file. Her record are nothing but a row in any table.



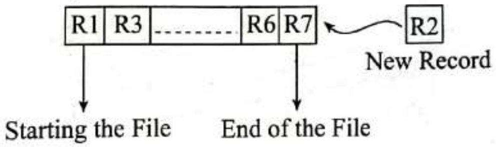
2. **Sorted File Method**

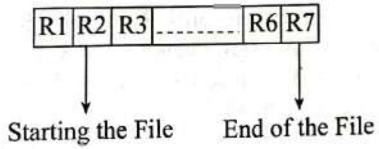
- In this method, the new record is always inserted at the file' end, and then it will sort the sequence in ascending or descending order. Sorting of records is based on the primary key or any other key.
- In the case of modification of any record. it will update the record and then sort the file, and lastly the updated record is placed in the right place.



Insertion of the new record

Suppose there is a preexisting sorted sequence of four records R_1, R_2, and son on upto R_6 and R_7. Suppose a new record R_2 has to be inserted in the sequence, then it will be inserted at the end of the file, and then it will sort the sequence.





Advantage of Sequential File Organization

- It contains a fast and efficient method for huge amount of data.
- In this method, files can be easily stored in cheaper storage mechanism like magnetic tapes.
- It is simple in design. It requires no much effort o store the data.
- This method is used for report generation or statistical calculations.

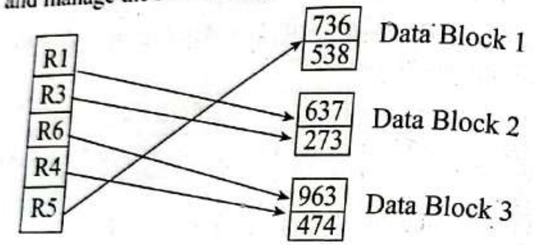
Disadvantage of Sequential File Organization

- It will waste time as we cannot jump on a particular record that is required but we have to move sequentially which takes our time.
- Sorted file method takes more time and space: For sorting the records.

(ii) **Heap File Organization**

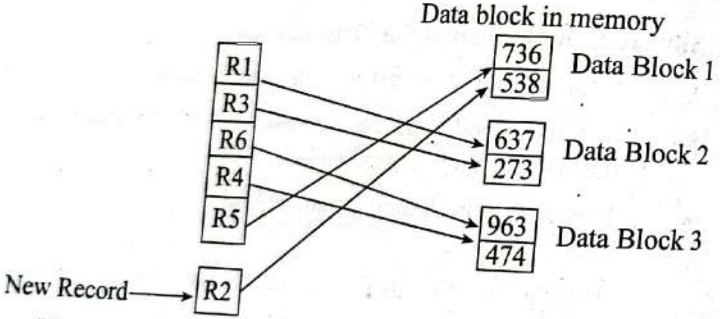
- It is the simplest and most basic type of organization. It works with data blocks. In heap file organization the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
- When the data books is full, the new record is stored in some other block. This new data block need not to be the very next data block, but it can select any data block in the memory to store new record. The heap file is also known as unordered file.

- In the file, every record has a unique id, and every page i file is of the same size. It is the DBMS responsibility to s and manage the new records.



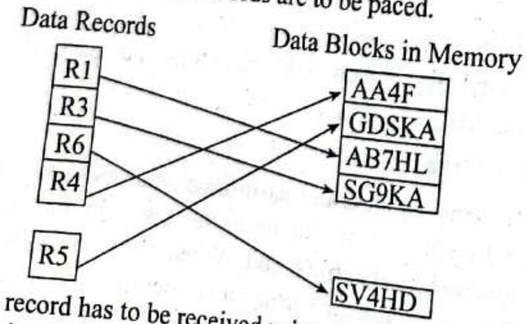
Insertion of a new record

Suppose we have five records R_1, R_2, R_3 R_4 and R_5 in a heap an suppose we want to insert a new record R_2 in a heap. If the data bloc 3 is full then it will be inserted in any of the database selected by th DBMS let's say data block 1.



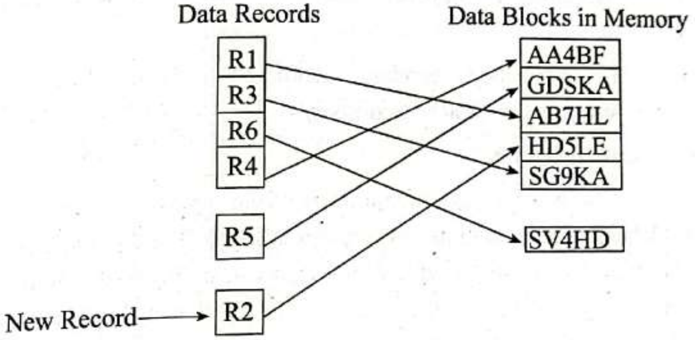
(iii) Hash File Organization

Hash file organization uses the computation of hash function on some filed of the records. The hash function's output determines the location of disk block where the records are to be paced.



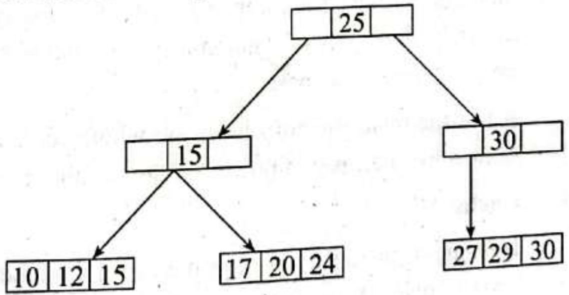
When a record has to be received using the hash key columns, then the address is generated, and the whole record is retrieved using that address. In the same way, when a new record has to be inserted, then the address is generated. The same process is applied in case of delete and update.

In their method, there is no effort for searching and sorting the entire file. In this method, each record will be stored randomly in the memory.



(iv) B$^+$ File Organization

- B$^+$ tree file organization is the advanced method of an indexed sequential access method. It uses tree - like structure to store records in file.
- It uses the same concept of key - index where the primary key is used to sort the records; For each primary key, the value of the index is generated and mapped with the record.
- The B$^+$ tree is similar to a binary search tree (BST) but it can have more than two children. In this method, all the records are stored only at leaf node. Intermediate nodes acts as a pointer to the lead nodes. They do not contain any records.



The above B$^+$ tree shows that

The is one root node of the tree i.e. 25

- There is an intermediary layer with nodes. They do not store the actual record. They have only pointers to the leaf node.
- The nodes to the left of the root node contain the prior value of the root and nodes to the right contain next value of the root i.r. 15 and 30 respectively.

- There is only one leaf node which has only values i.e. 10, 17, 20, 27, and 29.
- Searching for any record is easier as all the leaf nodes balanced.
- In this method, searching record can be traversed through single path and accessed easily.

Storage System in DBMS

A database system provides an ultimate view of the stored data. Several ty of data storage exist in most computer systems. The storage media a classified by the speed with which data can be accessed, by the c per unit of data to buy the medium, and by the mediums reliabili Among the media we classify them as:

1. **Primary Storage**

It is the primary area that offers quick access to the stored data. W also know the primary storage as volatile storage. It is because th type of memory does not permanently store data. As soon as th system leads to a power cut or crash, the data also get lost. Mai memory and cache are the types of primary storage.

(i) **Main Memory**

It is the one that is responsible for operating the data that i available by the storage medium. The main memory handle each instruction of a computer machine. This type of memory can store gigabytes of data on a system, but is small enough t carry the entire database.

A last the main memory loses the whole content id the system shuts down because of power failure or other reasons.

(ii) **Cache**

It is one of the costly storage media on the other hand, it is the fastest one. A cache is a tiny storage media which is maintained by the computer hardware usually.

Secondary Storage

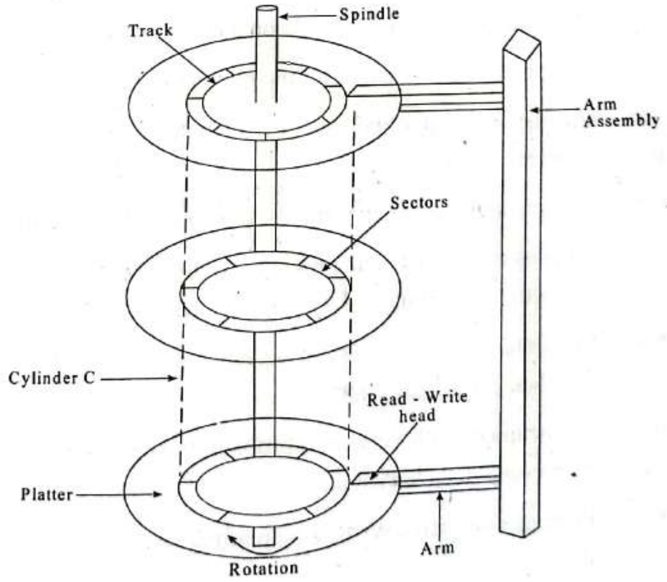
Secondary storage is also called as online storage. It is the storage area that allows the user to save and store data permanently. This type of memory does not lose the data due to any power failure or system crash. That's why we call it non - volatile storage.

(i) **Flash Memory**

Data survives despite of power failure. Data can be written at a location only once, but location can be erased and written to again. It can support only a limited numbers (10 K - 1 M) of write/erase cycles. Erasing of memory has to be done to an entire bank of memory. Reads are roughly as fast as main memory. It is widely used in embedded devices such as digital cameras.

(ii) **Magnetic - Disk**

This type of storage media is also known as online storage media. A magnetic disk is used for storing the data entire database. It is the responsibility of the computer system to make availability of the data from a disk to the main memory for further accusing. Also, if the system performs any operation on over the data, the modified data should be written back to the disk.



Read - write head

- Positioned very close to the platter surface (almost touching it)
- Reads or writes magnetically encoded information.

→ Surface of platter divided into circular tracks.

- Over 50 K - 100 K tracks per platter on typical hard disks.

→ Each track is divided into sectors.

- A sector is the smallest unit of data that can be read or writ
- Sector size typically 512 bytes.
- Typically sectors per track: 500 (on inner tracks) to 100 outer track)

→ To read/ Write a sector

- Disk arm swings to position head on right track.
- Platter spins continually; data is read/written as sector pas understood

→ Head - disk assemblies

- Multiple disk platters on a single spindle (1 to 5 usually)
- One should per platter, mounted on a common arm.
- Multiple disk platters on a single spindle (1 to 5 usually)
- One head per platter, mounted on a common arm.

→ Cylinder I consists of ith track of all the platters.

→ Disk controller - interfaces between the computer system and the dis drive hardware.

- Accepts high - level commands to read or write a sector.
- Initiates actions such as moving the disk arm to the right trac and actually reading and writing the data.
- Computers and attaches check sums to each sectors to veri that data is read back correctly.

→ If data is corrupted, with very high probability stored checksum wo match the recomputed checksum.

- Ensures successful/Writing by reading back sector after writin it.
- Performs remapping of bad sectors.

Performance Factors of Disk

1. **Disk Access Time:**

   The total time required by the computer to process a read/write reques and then retrieve the required data from the disk storage.

2. **Seek Time**

   The time required by the read/write head to move from the current track to the requested track.

3. **Rotational Latency**

   The time required by the read/write head to move from the current sector to the requested sector.

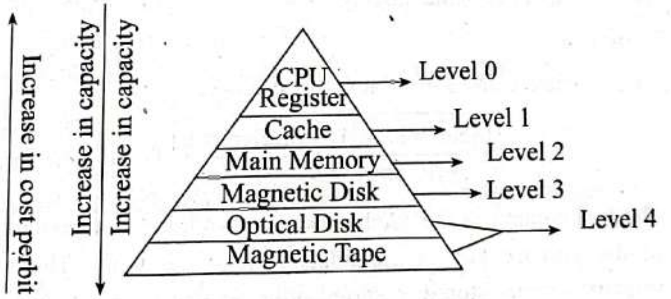
4. **Data Transfer Time**

   Data transfer Time is defined as the time required to transfer data between the system and the disk.

5. **Mean Time To Failure (MTTF)**

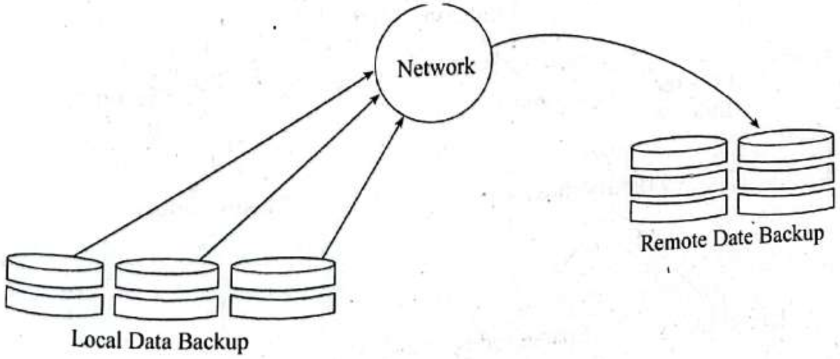
   The measure of how reliable a disk or component is.

Memory Hiearchy



Remote Backup System

Remote backup provides a sense of security in case the primary location where the database is located gets destroyed. Remote backup can be offline or real - time or online. In case it is offline it is maintained manually.



Online backup systems are more real - time and life savers for data administrative and investors. An online backup system is a mechanism w every bit of the real - time data is backed up simultaneously at two dis places. One of them is directly connected to the system and the other one kept at a remote place as backup.

As soon as the primary database storage fails, the backup system senses t failure and switches the user system to the remote storage. Sometimes th so instant that the users can' even realize a failure.

Indexing in DBMS

- Indexing is used to optimize the performance of database i minimizing the number of disk access required when a query processed.
- The index is a type of data structure. It is used to locate and access t data in a database table quickly.

Index Structure

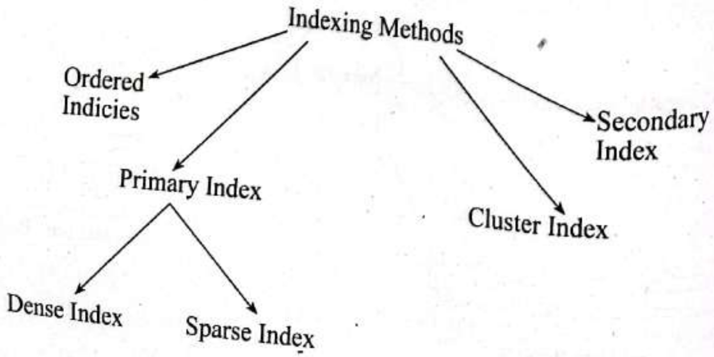
Index can be created using some database columns.

Search Key	Data Reference

Fig: Strucuture of Index

- The first column of the database is the search key that contains a cop of the primary key or candidate key of the table. The values o primary key are stored in sorted order so that the corresponding da can be accessed easily.
- The second column of the database is the data reference. It contains a set of pointer holding the address of the disk where the value of th particular key can be found.

Indexing methods



Ordered Indices

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

E.g: Suppose we have an employee table with thousands of records and each of which is 10 bytes long. If their IDS start with 1, 2, 3, ….. and so on and we have to search with ID - 543.

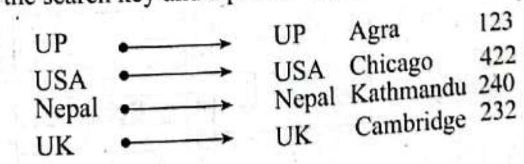
- In the case of a database with no index, we have to search the disk block from starting till it reaches 543. The DBMS will read the record after reading 5443 * 10 = 5430 bytes.
- In the case of an index, we will search using indexes and the DBMS will read the record after reading 542 * 2 = 1084 bytes which are very less compared to the pervious case.

Primary Index.

- 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 reach record and contain 1 : 1 relation between the records.
- As primary keys are stored in sorted order, the performance of the searching operation is quite efficient.
- The primary index can be classified into two types. Dense index and Sparse index.

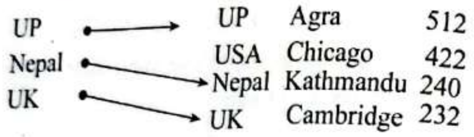
Dense Index

- The dense index contains an index record for every search key value in the data file. It makes searching faster. In this, the number of records in the index table.
- It needs more space to store index record itself. The index records have the search key and a pointer to the actual records on their disk.



Spare Index

- In the data file, index records appears only for a few items. Each item points to a block.
- In this, instead of pointing to each record in the main table, the index points to the records in the main table in the group.

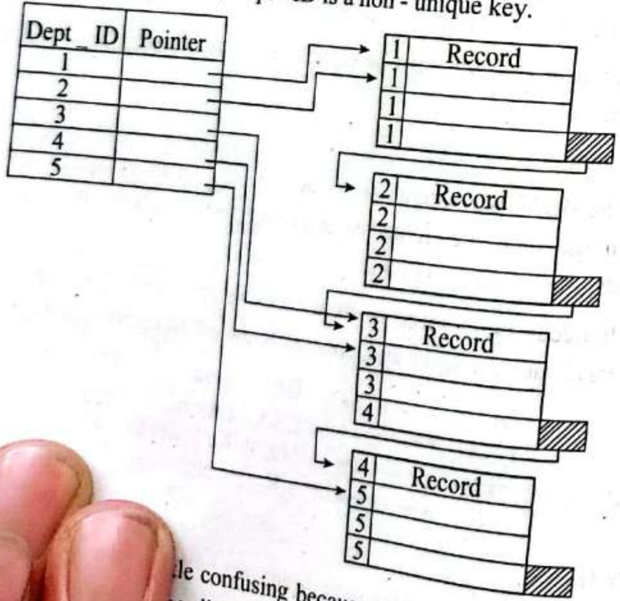


Clustering Index

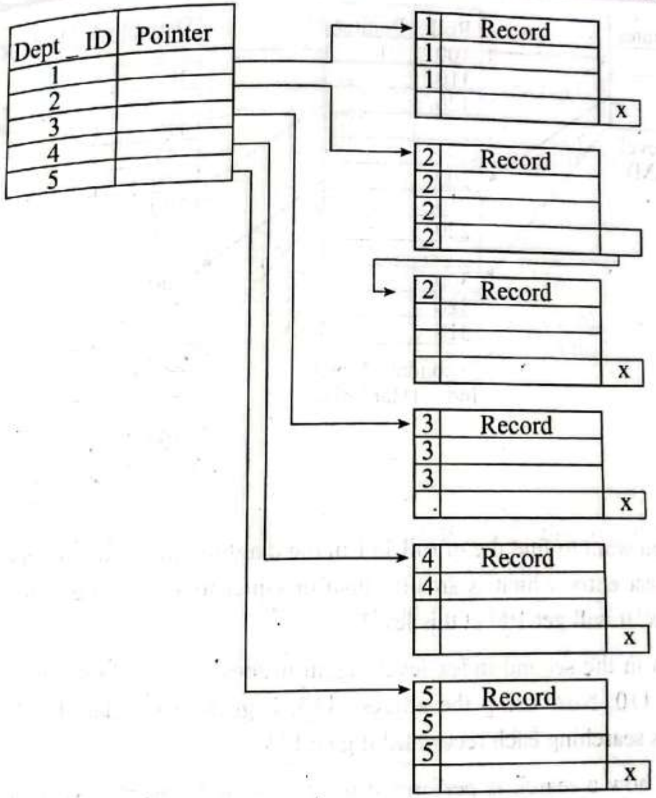
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may not unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. T method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



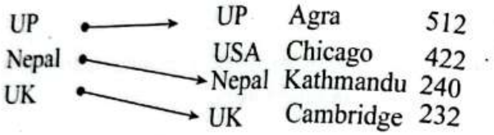
Th [illegible] re [illegible] le confusing because one disk block is shared by he different cluster. If we use separate disk block is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

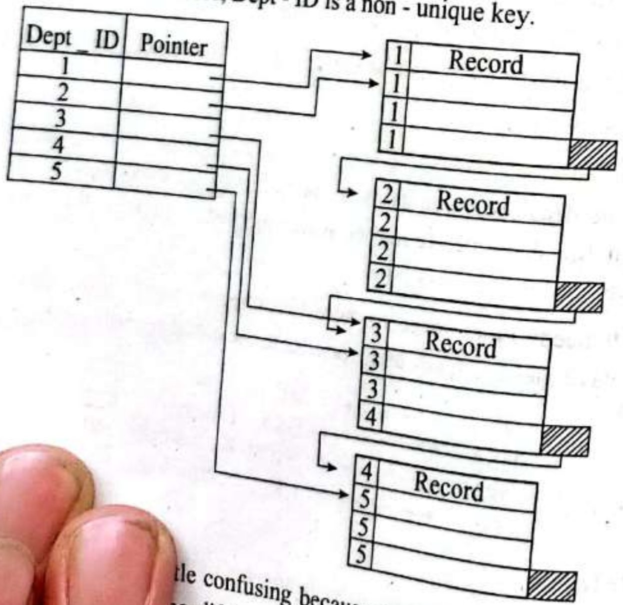


Clustering Index

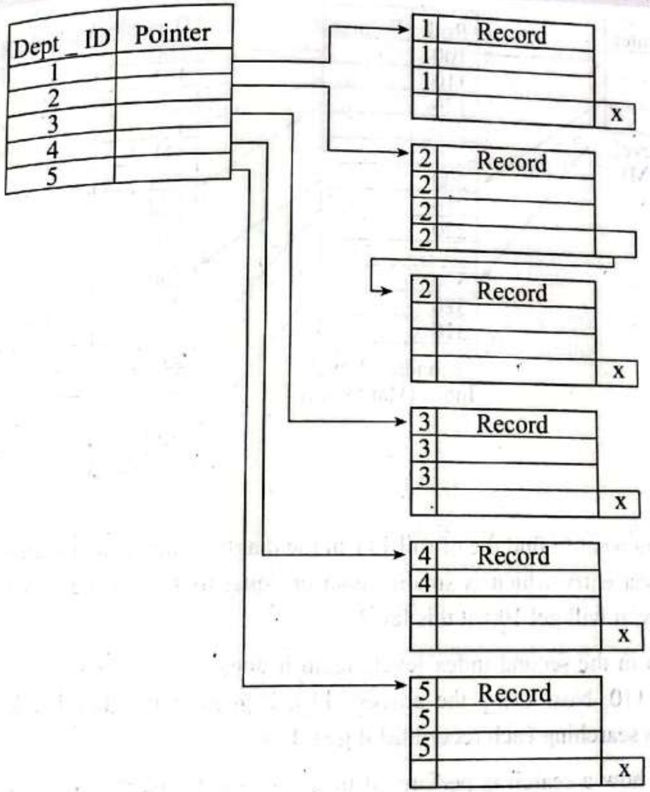
- A clustering index can be defined as an ordered data file. Sometin the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



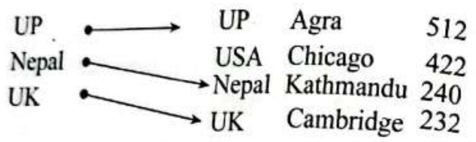
The e [illegible] tle confusing because one disk block is shared by [illegible] he different cluster. If we use separate disk block [illegible] is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

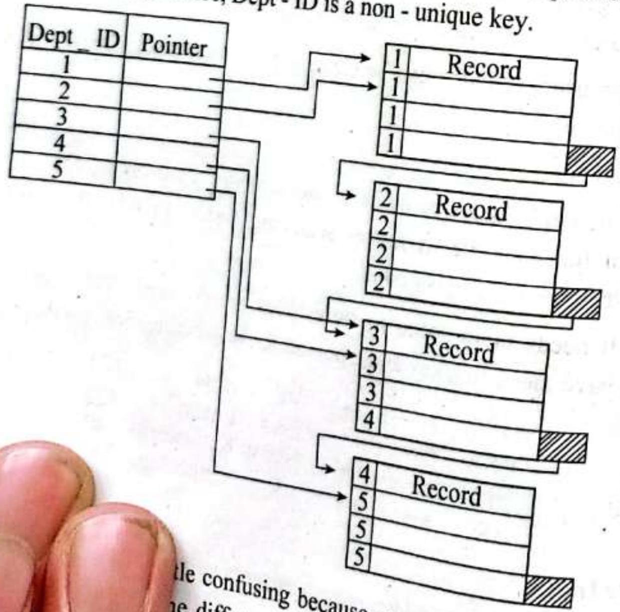


Clustering Index

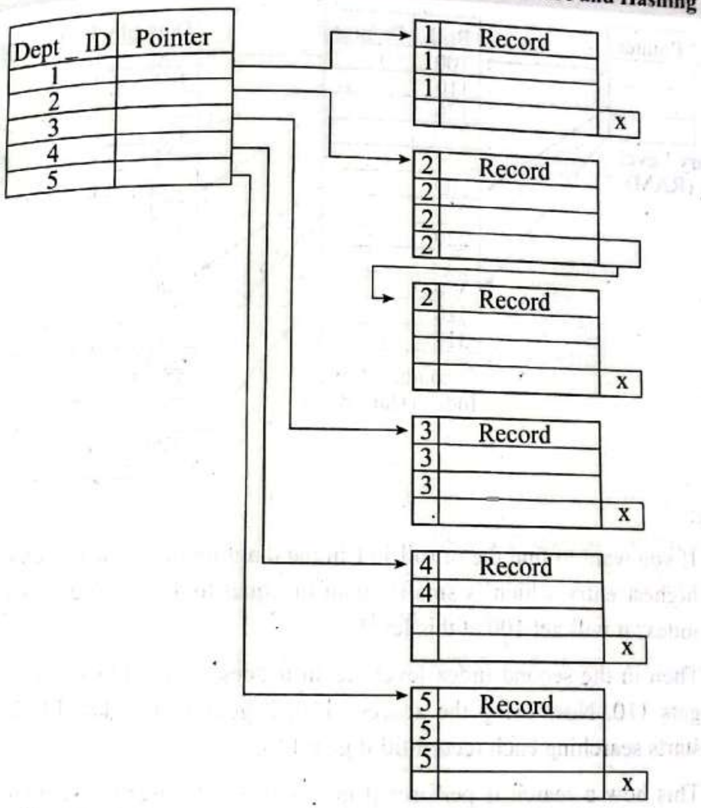
- A clustering index can be defined as an ordered data file. Sometin the index is created on non - primary key columns which may not unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



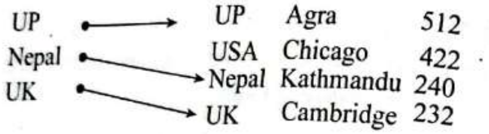
The e le confusing because one disk block is shared by he different cluster. If we use separate disk block is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

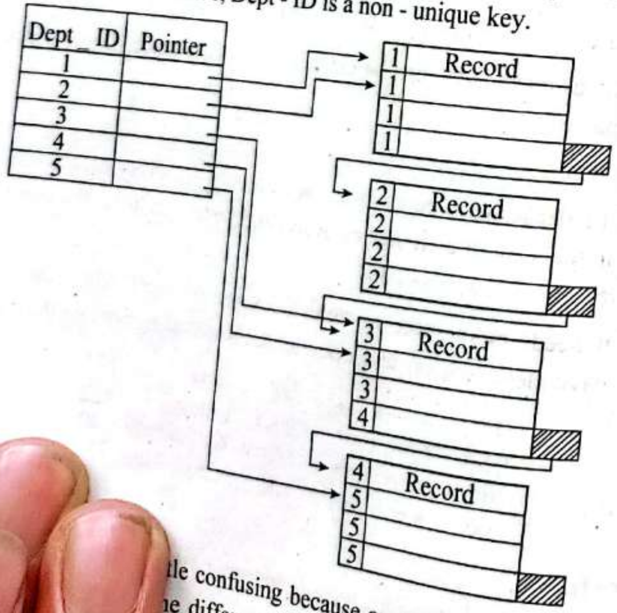


Clustering Index

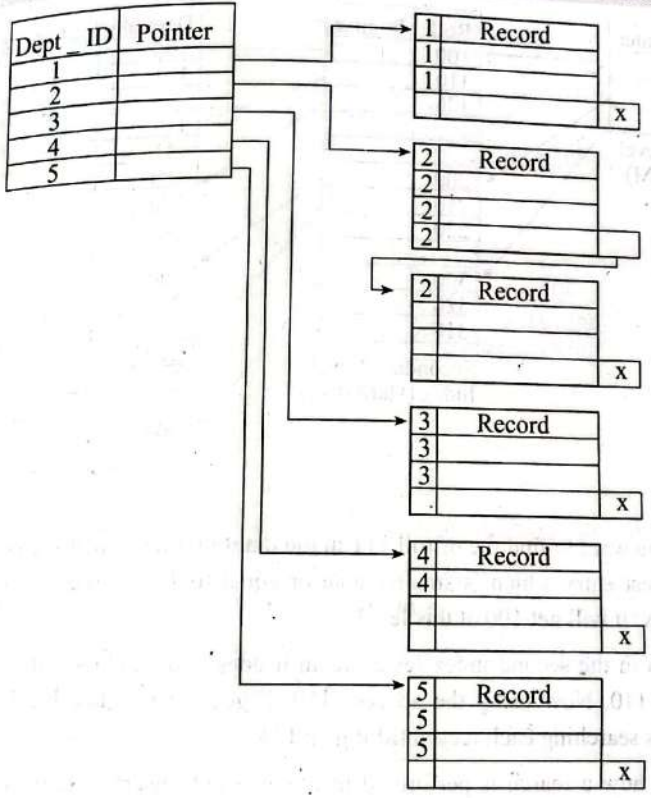
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or mo columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers poin the cluster as a whole. Here, Dept - ID is a non - unique key.



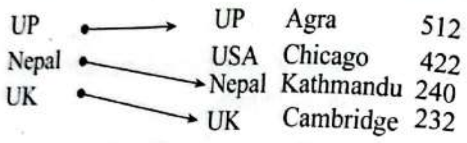
Th
e
tle confusing because one disk block is shared by
he different cluster. If we use separate disk block
is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)



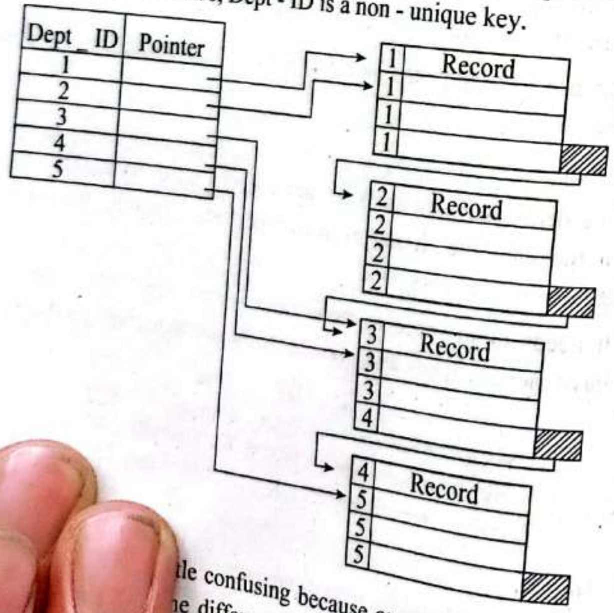
UP	Agra	512
USA	Chicago	422
Nepal	Kathmandu	240
UK	Cambridge	232

Clustering Index

- A clustering index can be defined as an ordered data file. Sometim the index is created on non - primary key columns which may not unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. T method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

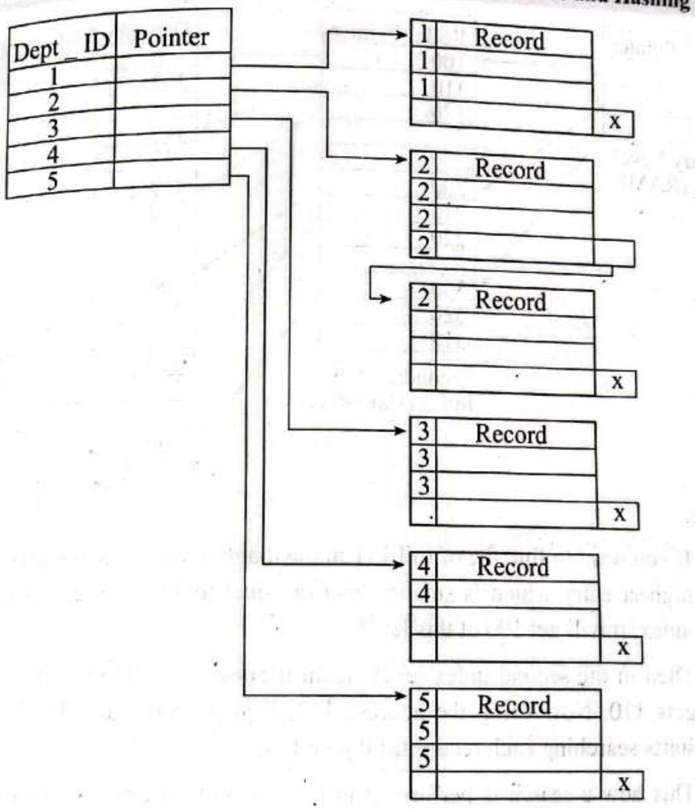
Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



Th
e

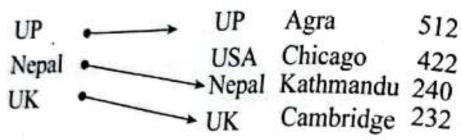
le confusing because one disk block is shared by
he different cluster. If we use separate disk block
is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

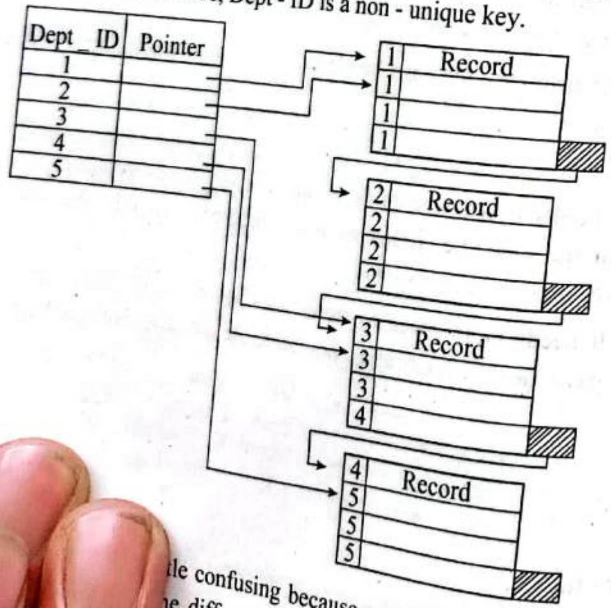


Clustering Index

- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

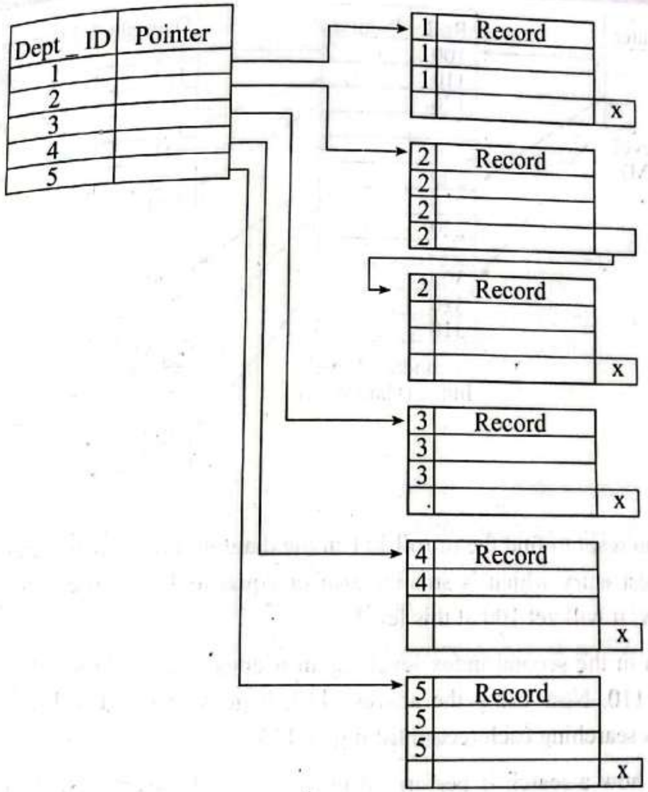
Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



Th
e

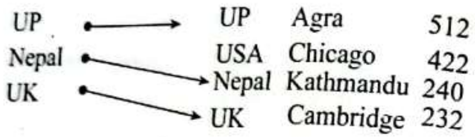
...le confusing because one disk block is shared by ...e different cluster. If we use separate disk block ...is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

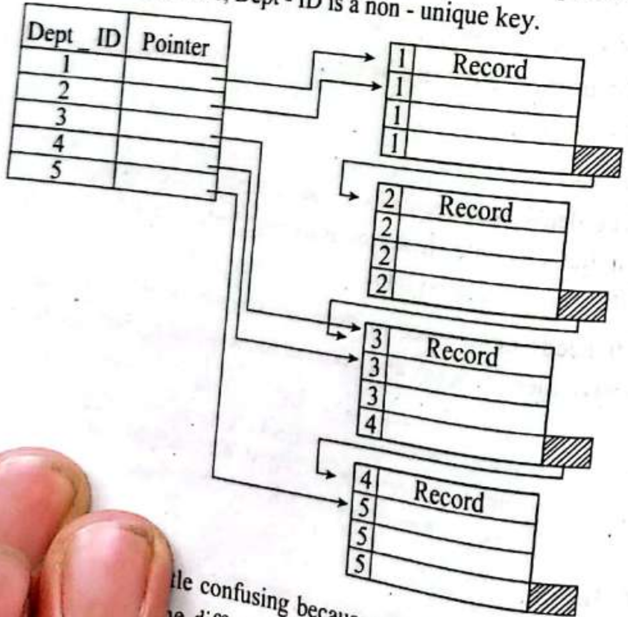


Clustering Index

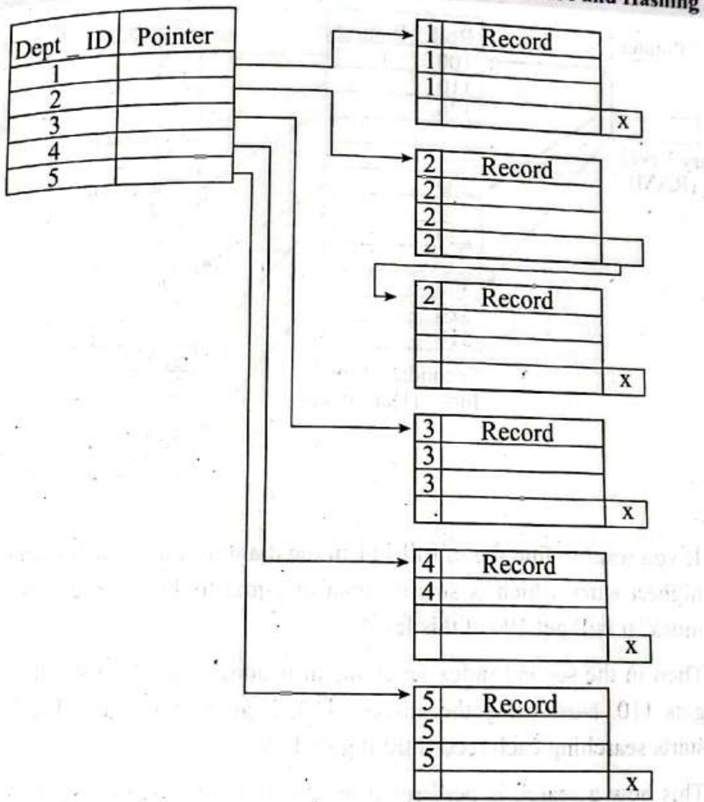
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may not unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. T method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



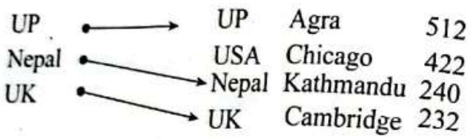
le confusing because one disk block is shared by ne different cluster. If we use separate disk block is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

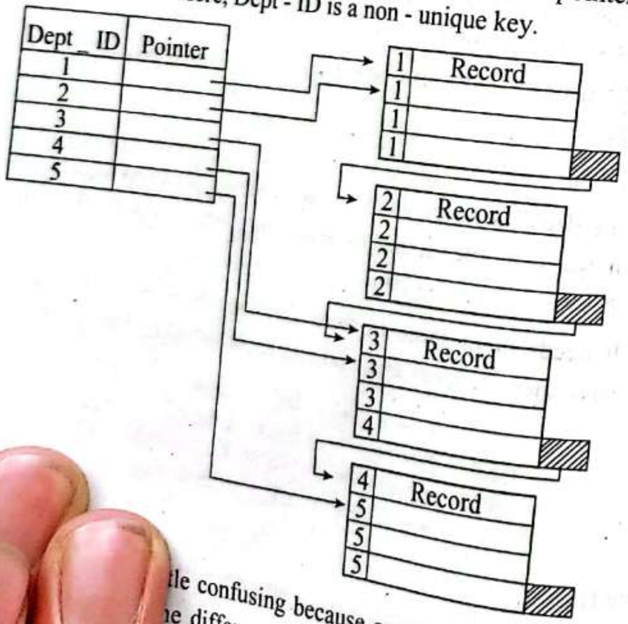


Clustering Index

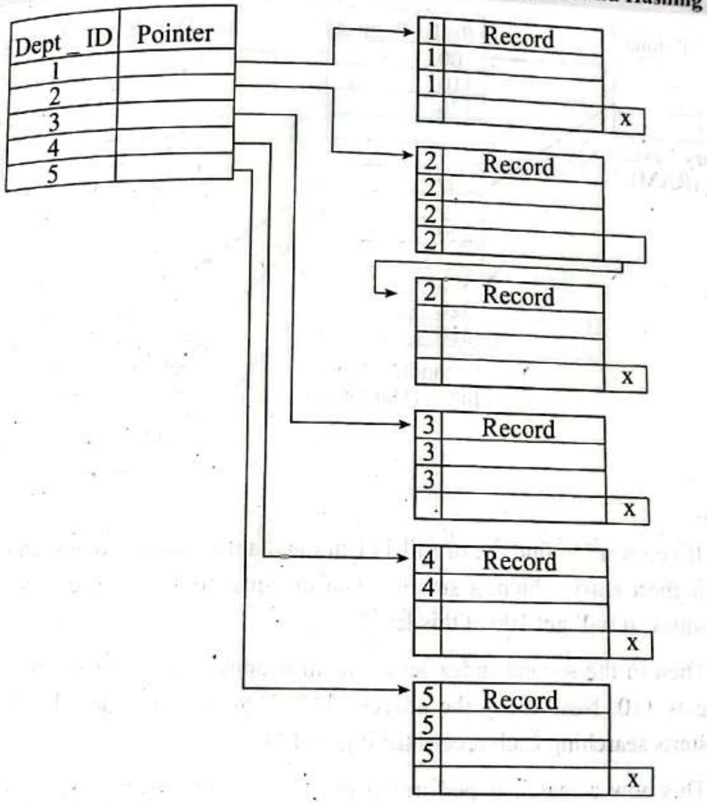
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may not unique for each record.
- In this case, to identify the record faster, we will group two or mo columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppo we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



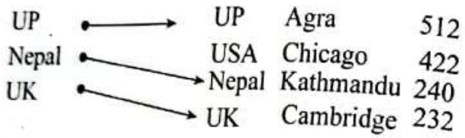
he... le confusing because one disk block is shared by ... he different cluster. If we use separate disk block ... is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

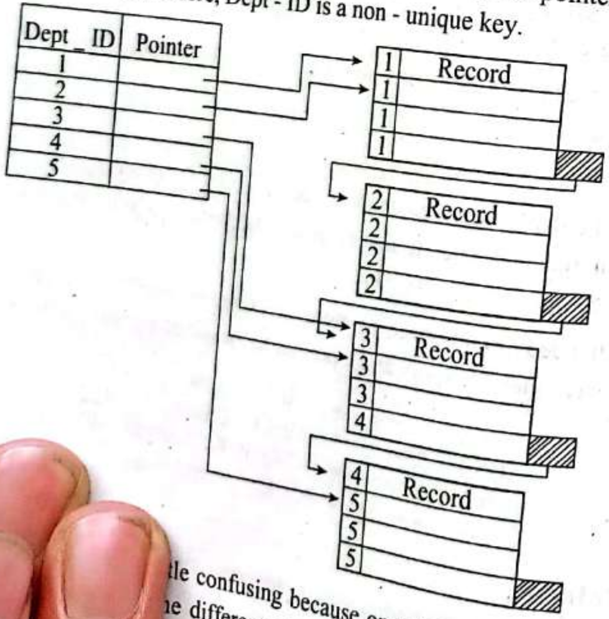


Clustering Index

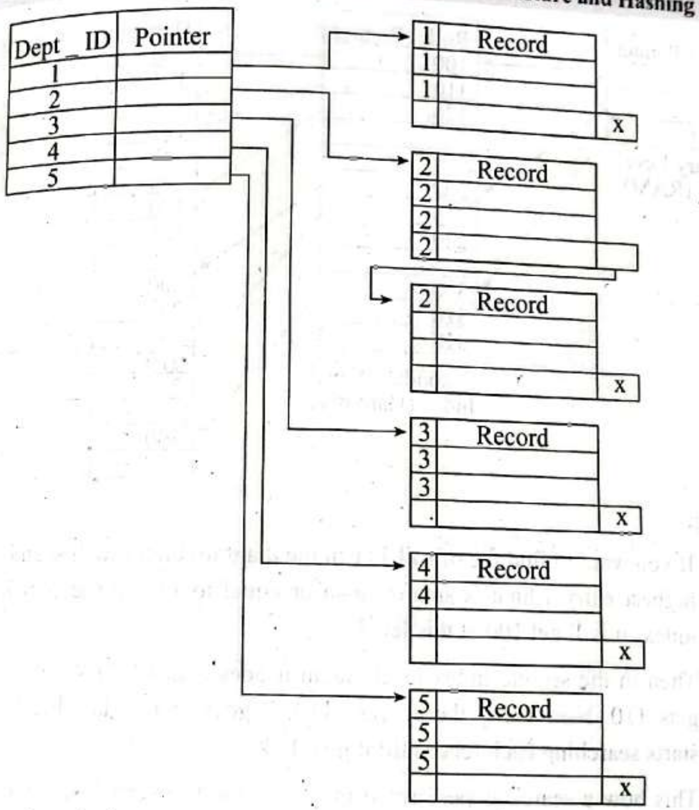
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or mo columns to get the unique value and create index out of them. T method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



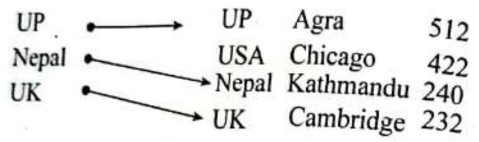
The e ...le confusing because one disk block is shared by ...e different cluster. If we use separate disk block ...is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

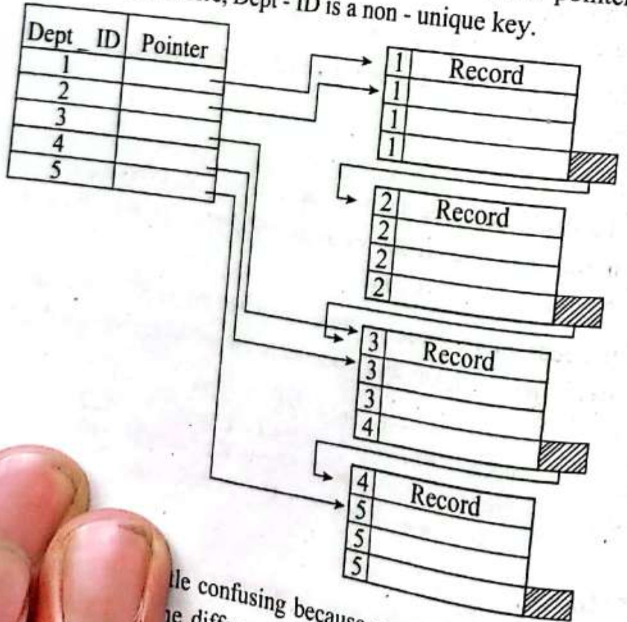


Clustering Index

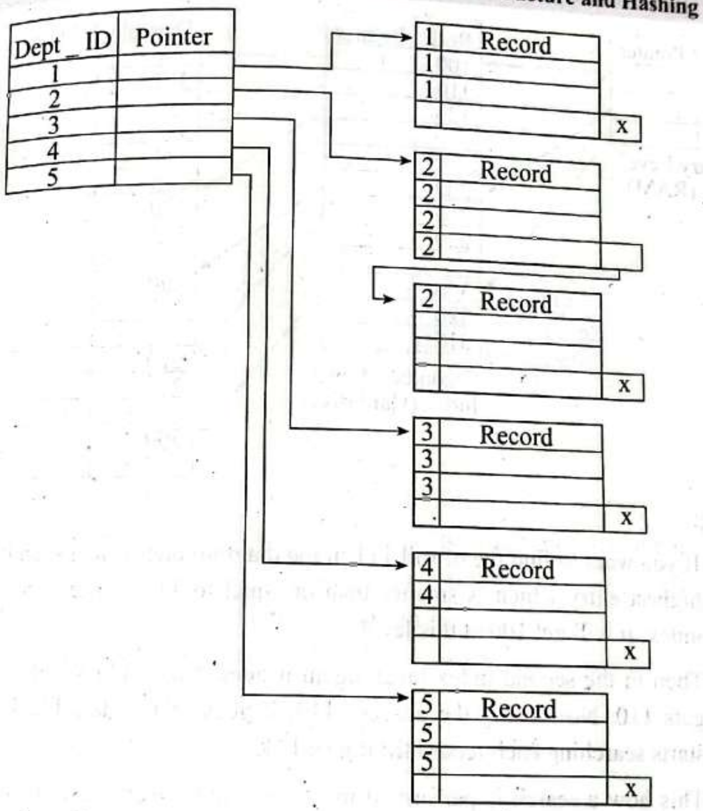
- A clustering index can be defined as an ordered data file. Sometim the index is created on non - primary key columns which may not unique for each record.
- In this case, to identify the record faster, we will group two or mo columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



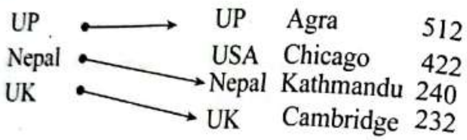
le confusing because one disk block is shared by he different cluster. If we use separate disk block is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

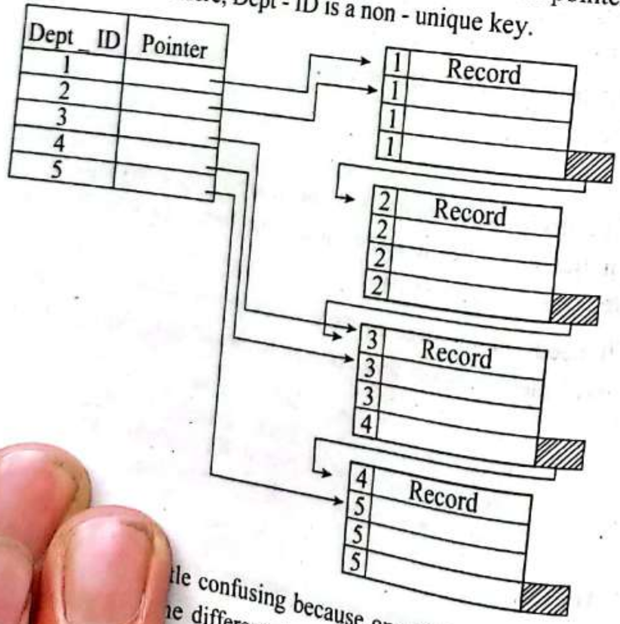


Clustering Index

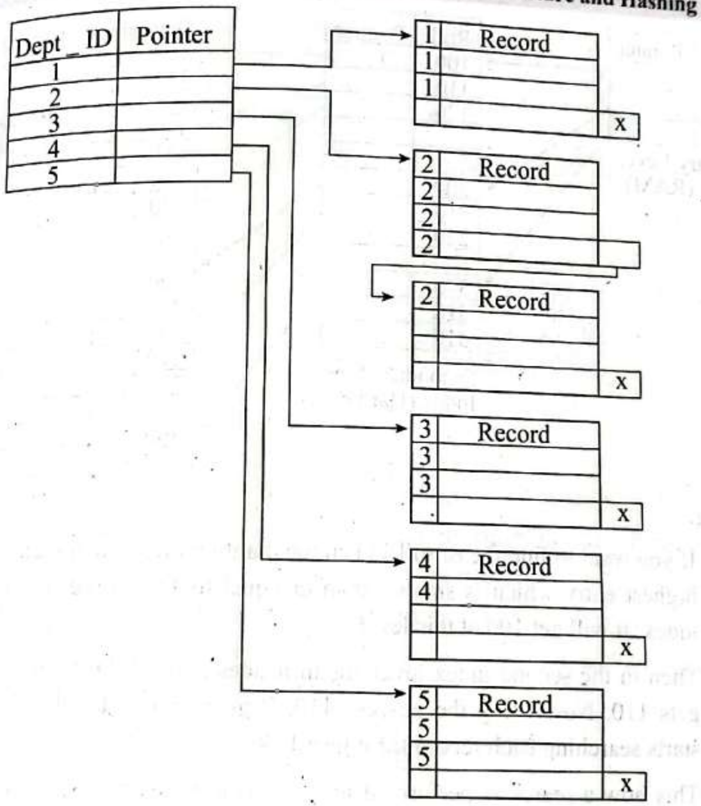
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may not unique for each record.
- In this case, to identify the record faster, we will group two or mo columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



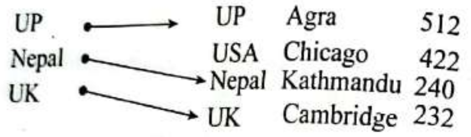
The ...le confusing because one disk block is shared by ...e different cluster. If we use separate disk block ...is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

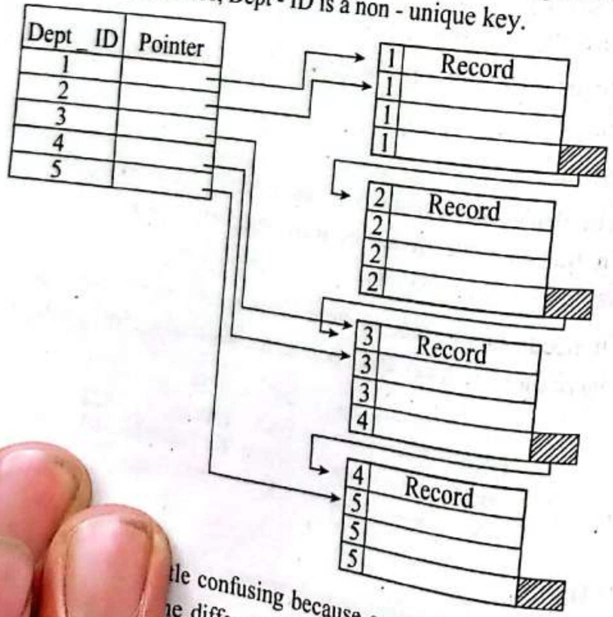


Clustering Index

- A clustering index can be defined as an ordered data file. Sometim the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or mo columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

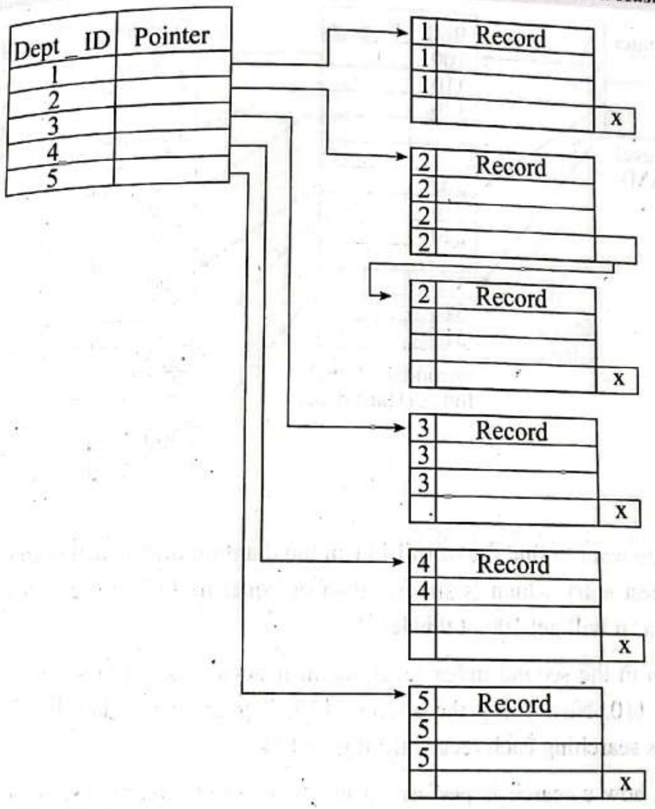
Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



The
e

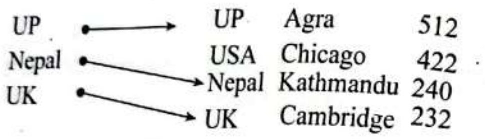
tle confusing because one disk block is shared by
e different cluster. If we use separate disk block
is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

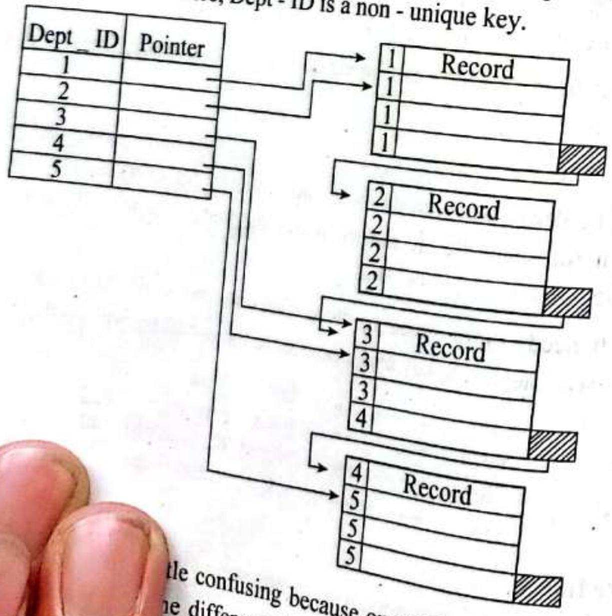


Clustering Index

- A clustering index can be defined as an ordered data file. Sometin the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. T method is called a clustering index.
- The records which have similar charactersitics are grouped, a indexes are created for these group.

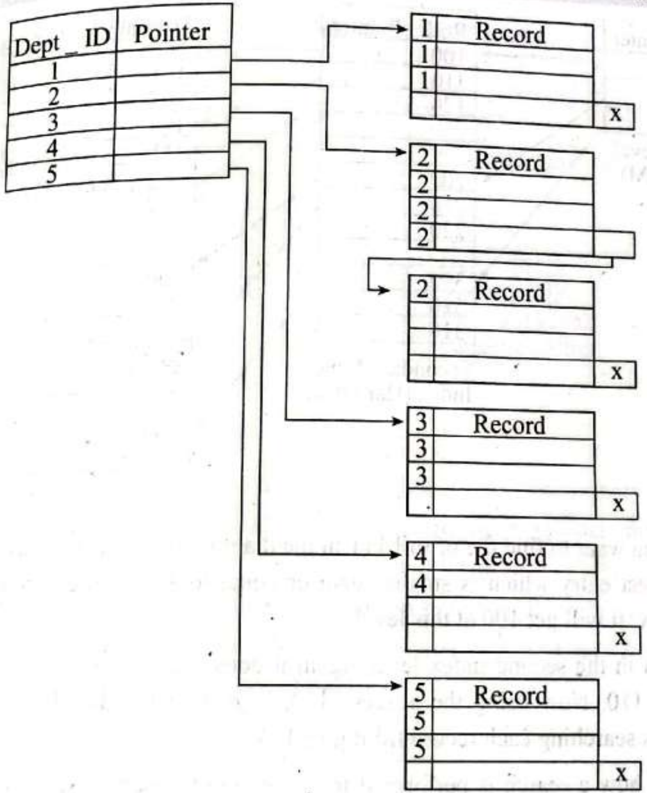
Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



Th

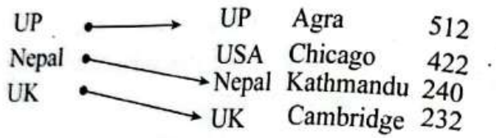
tle confusing because one disk block is shared by ne different cluster. If we use separate disk block is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

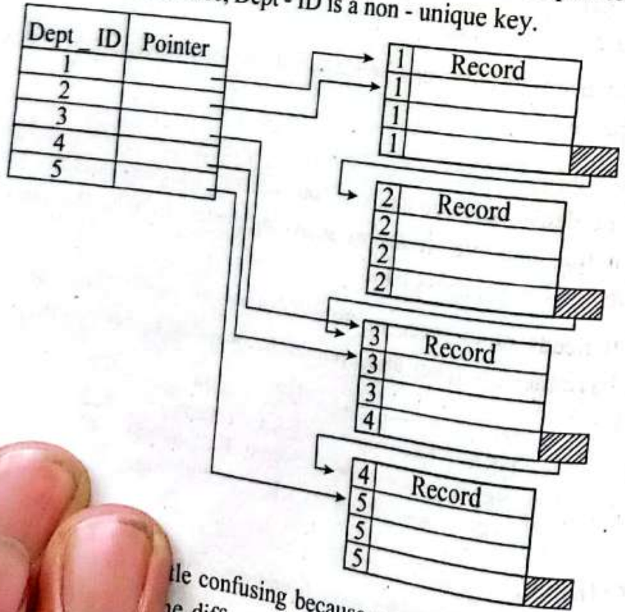


Clustering Index

- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

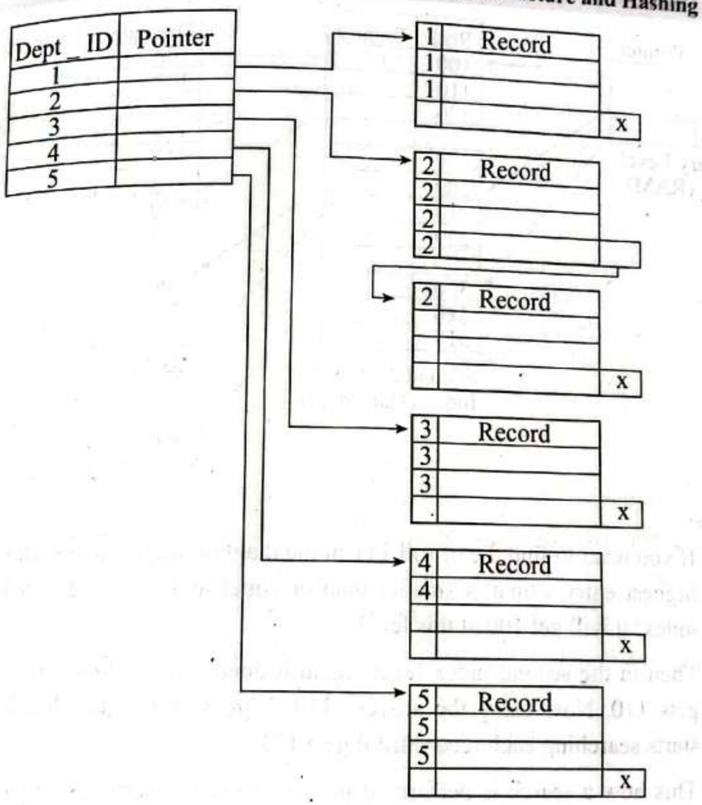
Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers poin the cluster as a whole. Here, Dept - ID is a non - unique key.



The

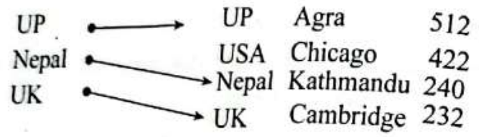
le confusing because one disk block is shared by
e different cluster. If we use separate disk block
is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

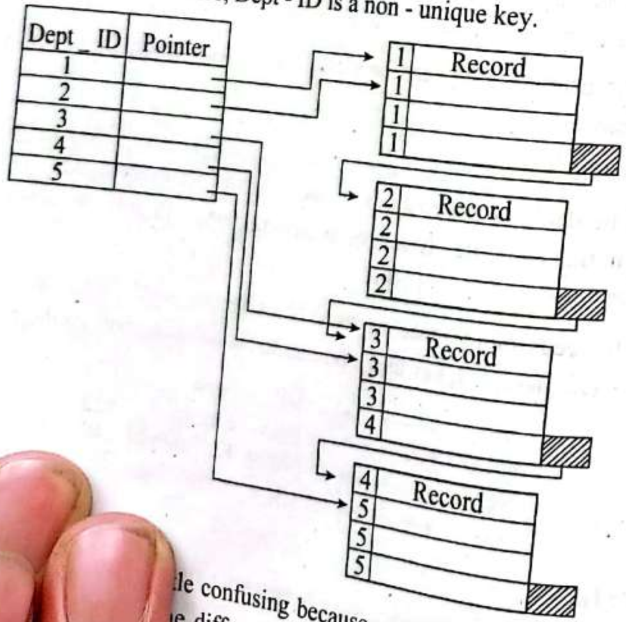


Clustering Index

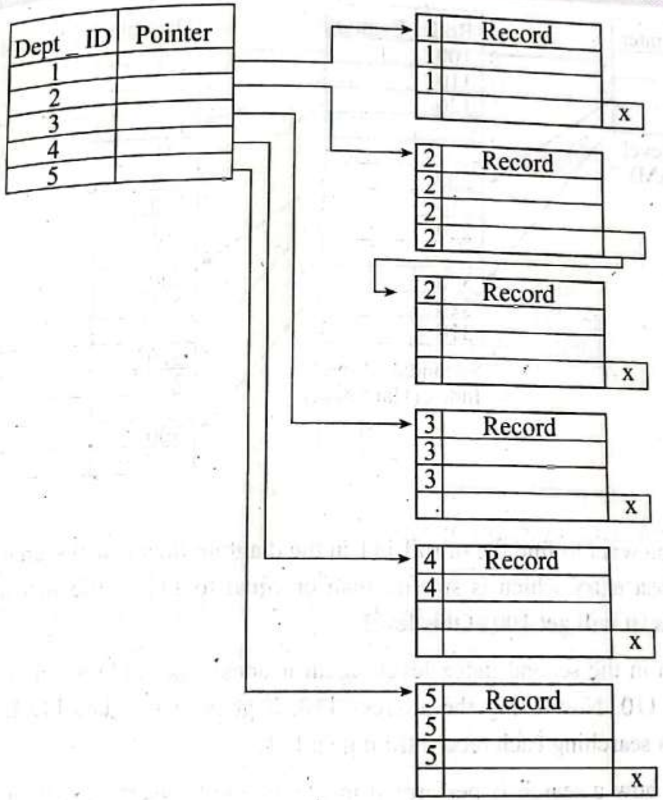
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. T method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppo we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



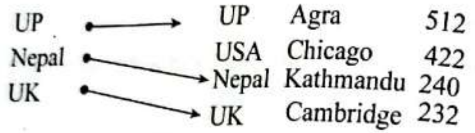
le confusing because one disk block is shared by e different cluster. If we use separate disk block s called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

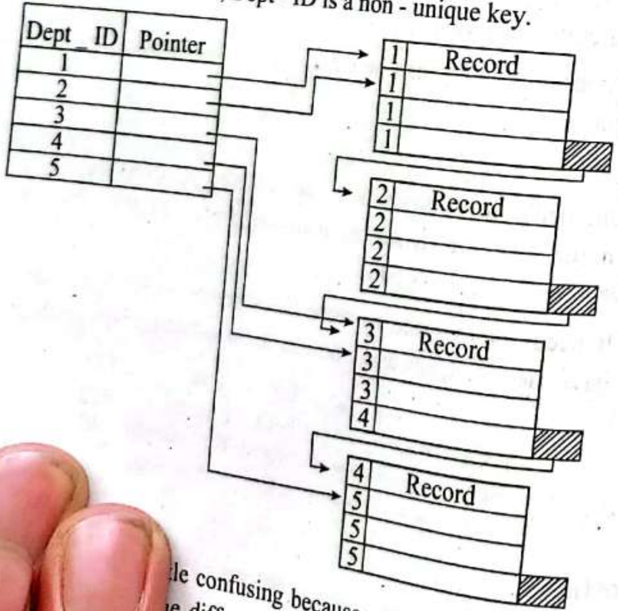


Clustering Index

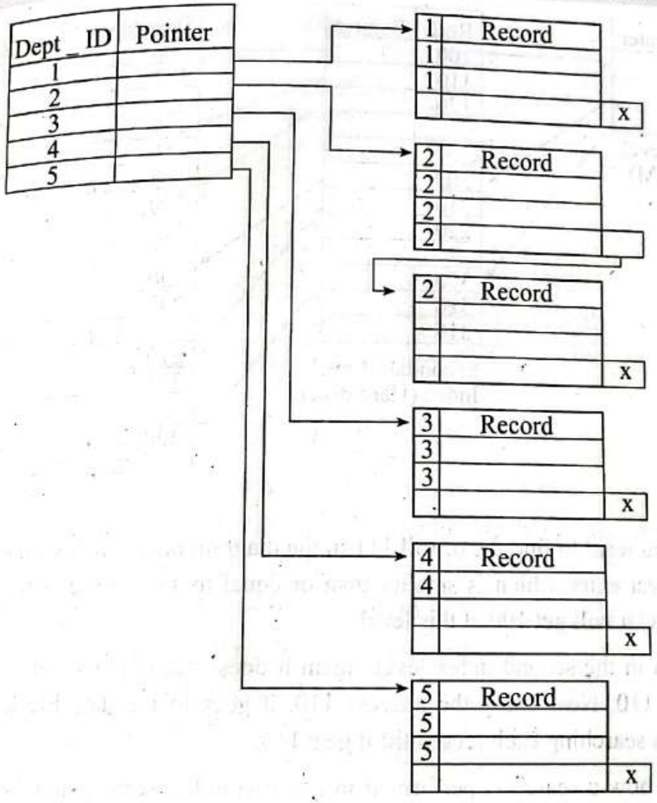
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. T method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



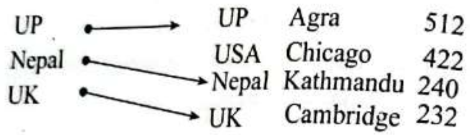
le confusing because one disk block is shared by e different cluster. If we use separate disk block s called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

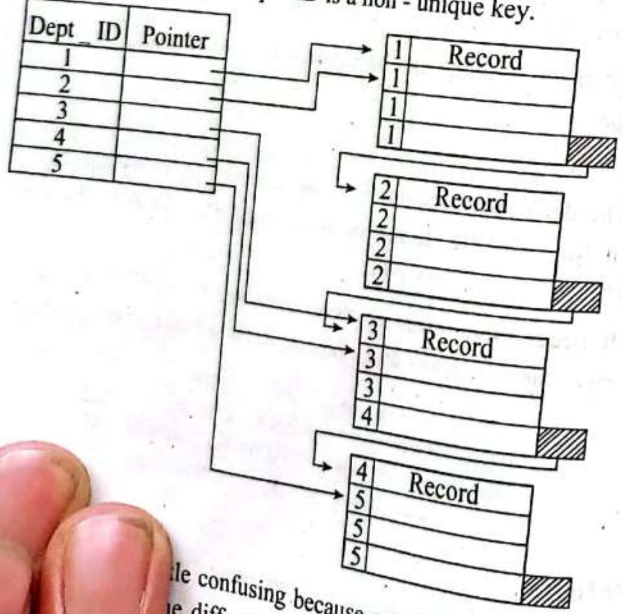


Clustering Index

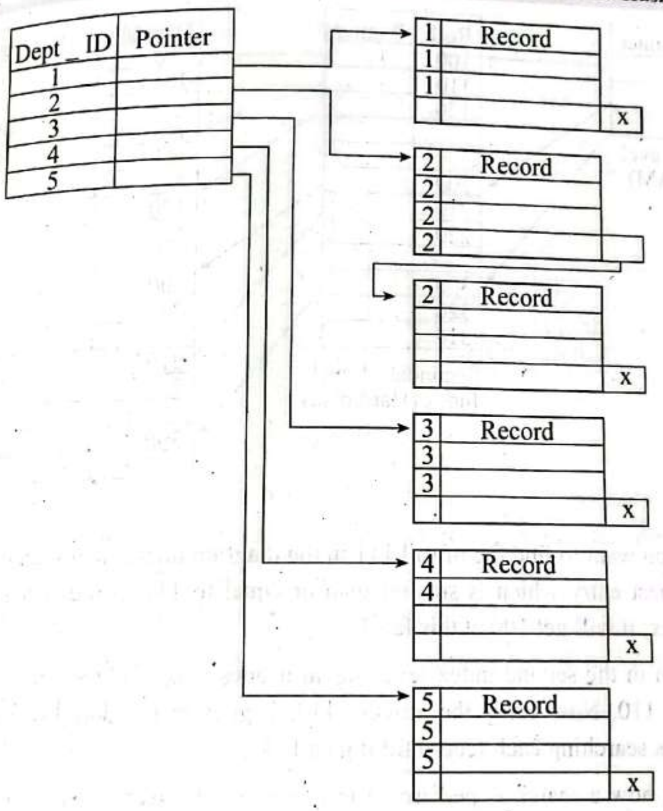
- A clustering index can be defined as an ordered data file. Sometin the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or mo columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



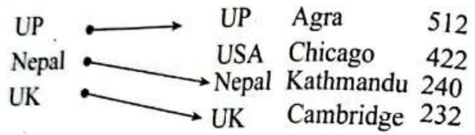
le confusing because one disk block is shared by e different cluster. If we use separate disk block is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

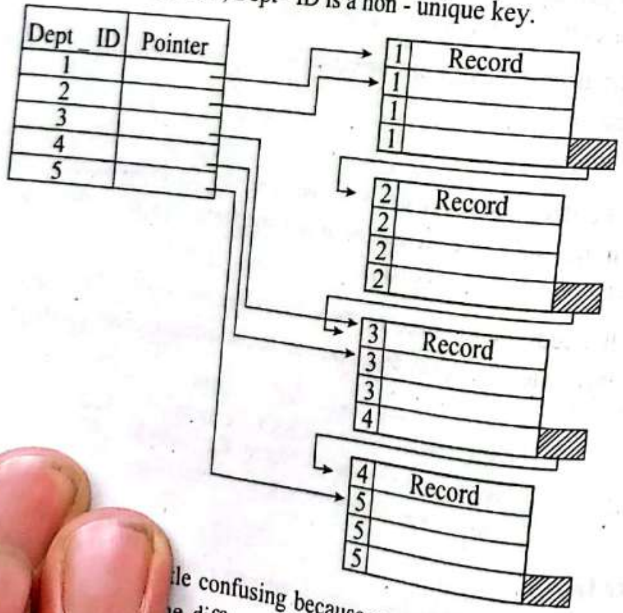


Clustering Index

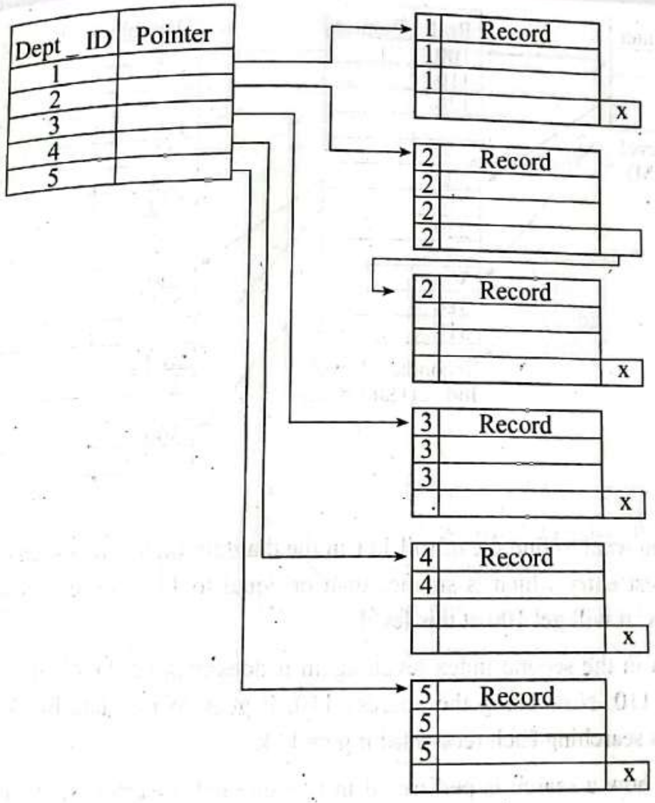
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. T method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppo we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



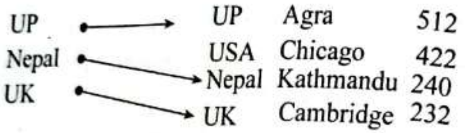
le confusing because one disk block is shared by e different cluster. If we use separate disk block is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

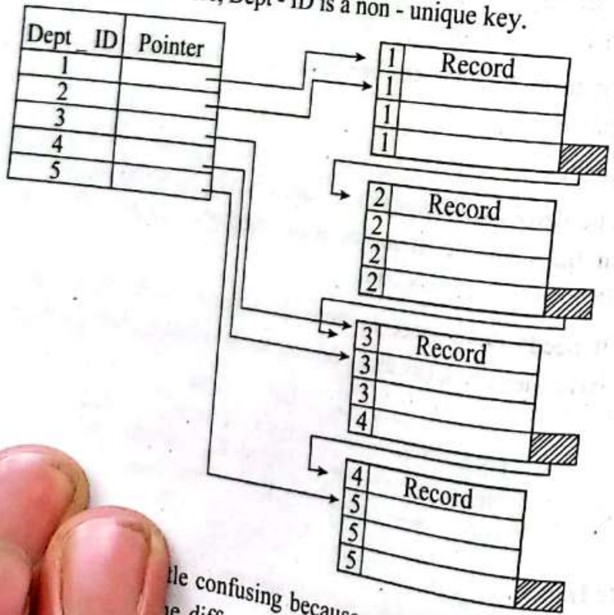


Clustering Index

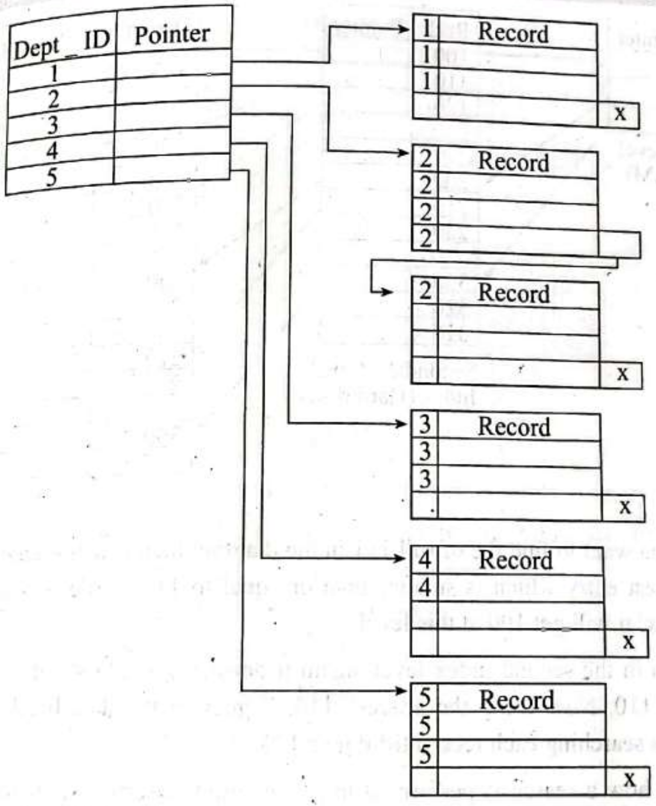
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may no unique for each record.
- In this case, to identify the record faster, we will group two or m columns to get the unique value and create index out of them. T method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



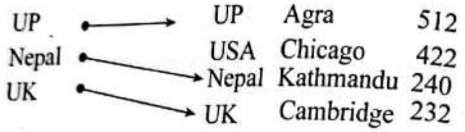
le confusing because one disk block is shared by ne different cluster. If we use separate disk block is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

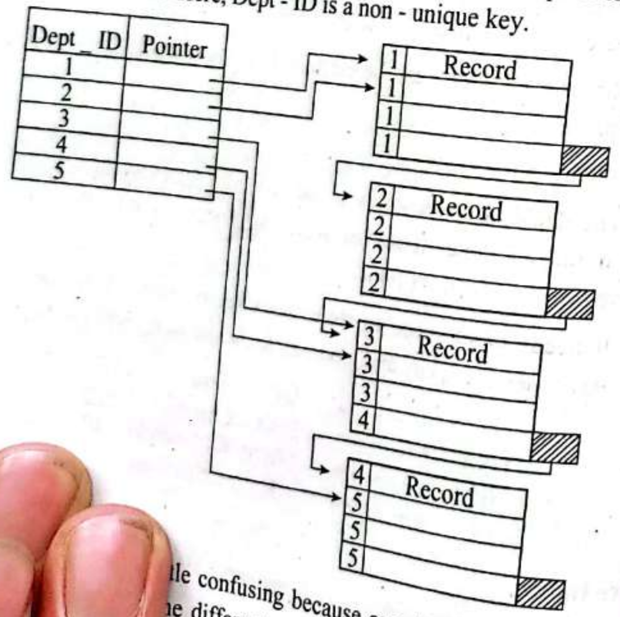


Clustering Index

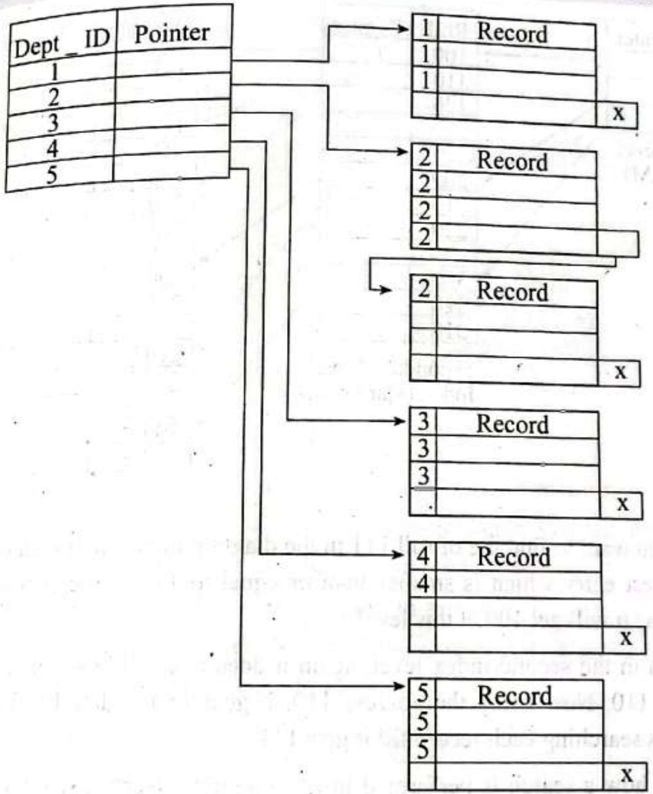
- A clustering index can be defined as an ordered data file. Someti the index is created on non - primary key columns which may not unique for each record.
- In this case, to identify the record faster, we will group two or mo columns to get the unique value and create index out of them. Th method is called a clustering index.
- The records which have similar characterstics are grouped, a indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppos we use a clustering index, where all employees which belong to the sam Dept - ID are considered within a single cluster, and index pointers point the cluster as a whole. Here, Dept - ID is a non - unique key.



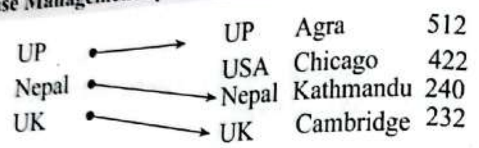
le confusing because one disk block is shared by he different cluster. If we use separate disk block is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)

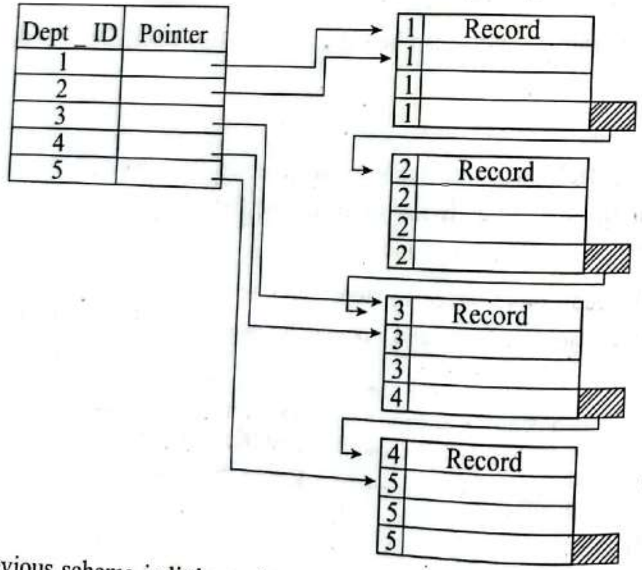


Clustering Index

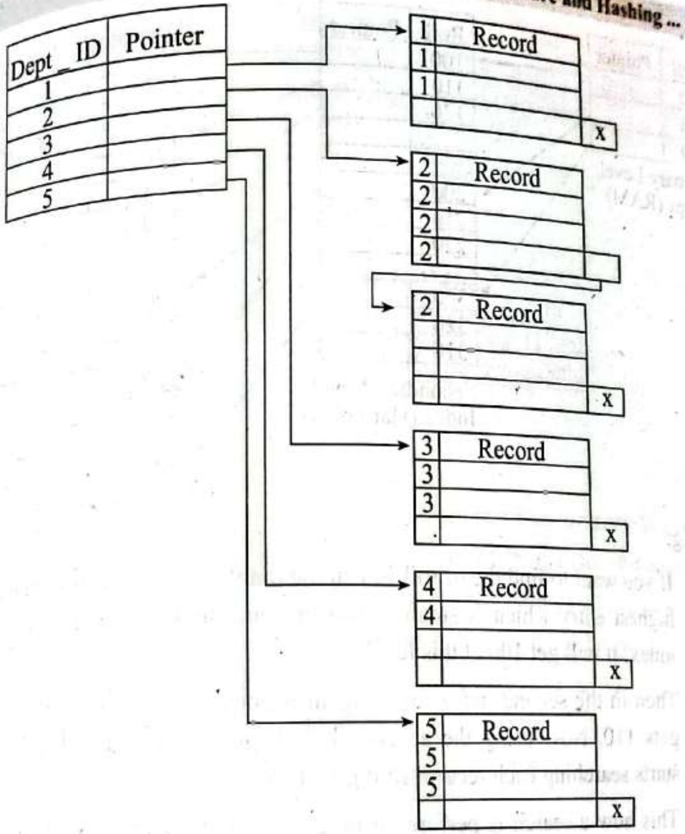
- A clustering index can be defined as an ordered data file. Sometimes the index is created on non - primary key columns which may not be unique for each record.
- In this case, to identify the record faster, we will group two or more columns to get the unique value and create index out of them. This method is called a clustering index.
- The records which have similar characterstics are grouped, and indexes are created for these group.

Example:

Suppose a company contains several employed in each department. Suppose we use a clustering index, where all employees which belong to the same Dept - ID are considered within a single cluster, and index pointers point o the cluster as a whole. Here, Dept - ID is a non - unique key.



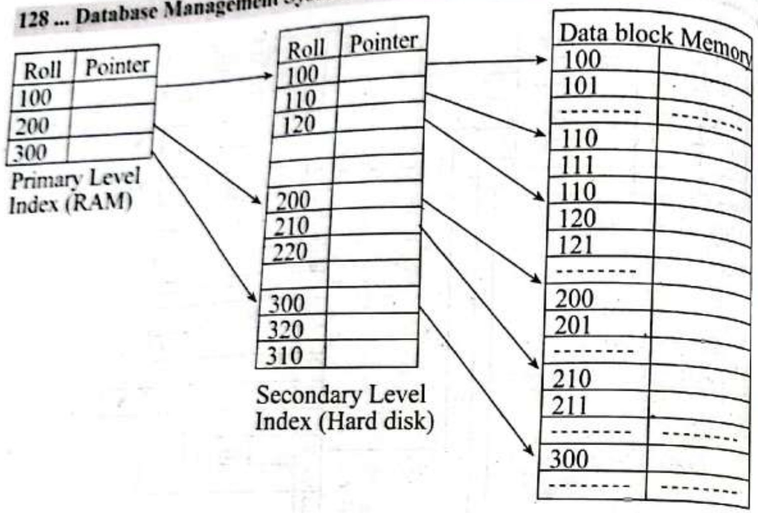
The previous schema is little confusing because one disk block is shared by records which belong to the different cluster. If we use separate disk block for separate cluster, then it is called better technique.



Secondary Index

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk)



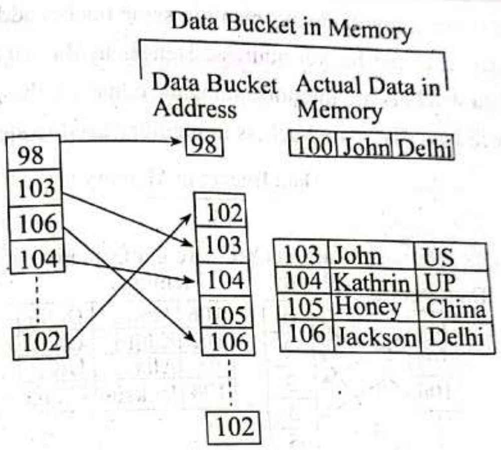
For e.g:

- If you want to find the of roll 111 in the diagram then it will search the highest entry which is smaller than or equal to 111 in the first level index. It will get 100 at this level.
- Then in the second index level, again it does max (111) < = 111 and gets 110. Now using the address 110, it goes to the data block and starts searching each record till it gets 111.
- This how a search is performed in this method. Inserting updating or deleting is also done in the same manners.

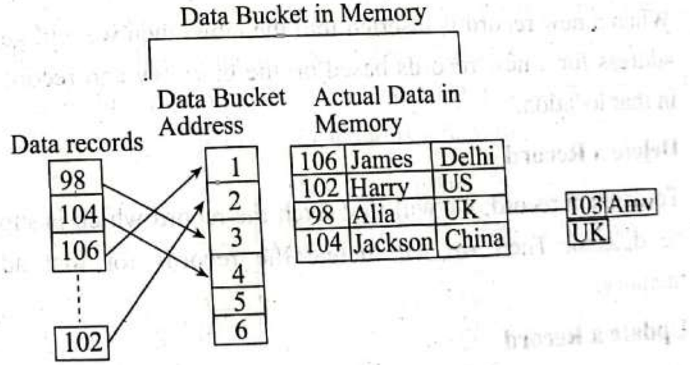
Hashing

Ina huge database structure, it is very inefficient to search all the index values and reach the desired data. Hashing technique is used to calculate the direct location of a data record on the disk without using index structure.

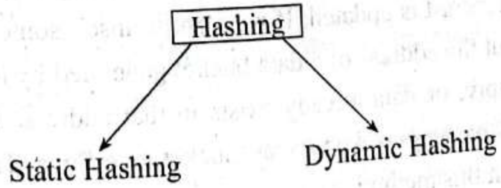
In this technique, data in stored at the data blocks whose address is generated by using the hashing function. The memory location where these records are stored is known as data bucket or data blocks In this records, a hash function can choose any of the column value to generate the address. Most of the time, the hash function uses the primary key to generate the address of the data block. A hash function is a simple mathematical function to any complex mathematical function we can even consider the primary key itself as the address of the data block. That means each row whose address will be the same as a primary key stored in the data block.



The above diagram shows data block address same as primary key value. This hash function can also be a simple mathematical function like exponential, mod, cos, sin etc. Suppose we have mod (5) hash function to determine the address of the data block. In this case, it applies mod (5) hash function on the primary keys and generates mod 3, 3, 1, 4, and 2 respectively, and stored in those data block addresses.

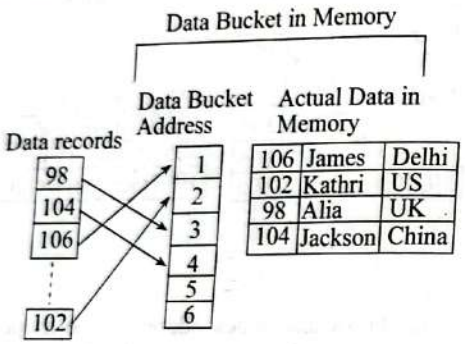


Types of Hashing



Static Hashing

In static hashing, the resultant data bucket address will always be the same. That means if we generate an address For EMP _ ID = 103 using the hasl function mod (5) then it will always result in same bucket address 3. Here there will be no change in the bucket address. Hence, in this static hashing, the number of data buckets in memory remains constant throughout. In this example, we will have five data buckets in memory used to store the data.



(i) Operations of Static Hashing

Searching a record when a records needs to the searched, then the same hash function retrieves the address of the bucket where the data is stored.

(ii) Insert a Record

When a new record is inserted into the table, then we will generate an address for a new records based on the hash key and record is stored in that location.

(iii) Delete a Record

To delete a record, we will first fetch the record which is supposed to be deleted. Then we will delete the records for that address in memory.

iv) Update a Record

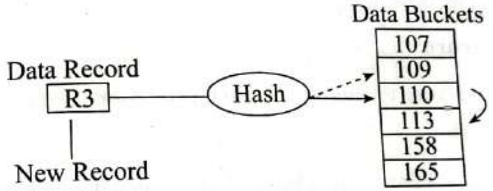
To update a record, we will first search it using a hash function, and then data record is updated. If we want to insert some new record into the file but the address of a data bucket generated by the each function is not empty, or data already exists in that address. This situation in the static hashing is known as bucket overflow. This is a critical situation in this method.

To overcome this situation, there are various methods. some commonly used method are as follows:

1. Open Hashing

When a hash function generates an address at which data is already stored, then the next bucket will be allocated to it. This mechanism is called as linear probing.

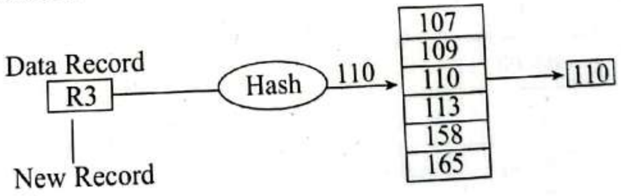
For E.g: Suppose R_3 is a new address which needs to be inserted, the hash function generates as 112 for R_3. But the generated address is already full. So the system searches next available data bucket, 113 and assigns R_3 to it.



2. Close Hashing

When buckets are full, then a new data bucket is allocated for the same hash result and it linked after the previous one. This mechanism is known as overflow chaining.

For E.g: Suppose R_3 is a new address which needs to be inserted into the table, the hash function generates address as 110 for it. But this bucket is full to store the new data. In this case, a new bucket is inserted at the end of 110 bucket and is linked to it.



Dynamic Hashing

- The dynamic hashing method is used to overcome the problems of static hashing like bucket overflow.
- In this method, data buckets grow or shrink as the records increases or decreases. This method is also known as extendable hashing method.
- This method makes hashing dynamic, i.e. it allows insertion or deletion without resulting is poor performance.

How to search a key

- First, calculate the hash address of the key.
- Check how many bits are used in the directory and these bits a called as i.
- Take the least significant i bits of the hash address. This given a index of the directory.
- Now using the index, go to the directory and find bucket addres where the records might be.

How to insert record

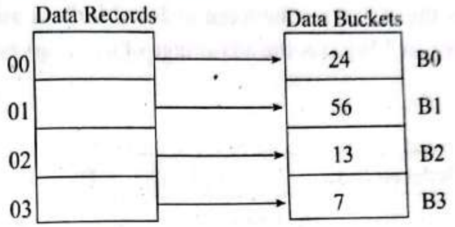
- Firstly, you have to follow the same procedure for retrieval ending u in some bucket.
- If there is still space in that bucket, then place the record in it.
- If the bucket is full, the we will split the bucket and redistribute the records.

For E.g:

Consider the following grouping of key into buckets, depending on the prefix of their hash address:

Key	Hash Address
1	11010
2	00000
3	11110
4	00000
5	01001
6	10101
7	10111

The last two bits of 2 and 4 are 00. So it will go into bucket B_0. The last two bits of 5 and 6 are 01, so it will go into bucket B_1. The last two bits of 1 and 3 are 10, so it will go into bucket B_2. The last two bits of 7 are 11, so it will go into B_3.



Advantages of dynamic Hashing

1. Performance does not decrease as the data grows in the system.
2. In this method, memory is well utilized as it grows and shrinks with the data.
3. This methods is good for the dynamic database where data grows and shrinks frequently.

Disadvantage f dynamic Hashing

1. In this method, if the data size increases then the bucket size is also increased.
2. In this case, the bucket overflow situation will also occur. But it might take little time to reach this situation than static hashing.

Old Question Solution

1. **What do you mean by hashing and indexing? Differentiate between dense index and sparse index? (2076 Baiskh)**

⇒ Hashing is an effective technique to calculate the direct location of a data record on the disk without using index structure.

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.

Dense Index VS Sparse Index

	Dense Index		Spare Index
1.	Index size is larger.	1.	Index size is smaller.
2.	Records in data file need not be clustered.	2.	Records in data file need to be clustered.
3.	Timer to locate data is less.	3.	Time to locate data is more.
4.	Computing time in RAM is less.	4.	Computing time in RAM is more.
5.	Overhead for insertions and deletions are more.	5.	Overhead for insertion and deletions are less.

2. **What is the difference between ordered indices and hash indices in a database? What is the advantage of using sparse index? (2071 Baisakh)**

⇒

Ordered Indices	Hash Indices
1. A data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done.	1. An effective technique to calculate the direct location of a data record on the disk without using index structure.
2. Uses data reference that holds the address of the disk block with the value corresponding to the key.	2. Uses mathematical functions called hash functions to calculate direct records on the disk.
3. Does not work well for large databases.	3. Works well for large database.

Advantages of sparse Index

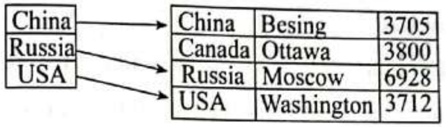
1. It reduces the size of the index, saving space and decreasing maintenance of the index.
2. Not necessary to generate unnecessary index entries.

3. **What is the use of RAID storage device? How is a record searched from a sparse sequential index? (2073 Bhadra)**

⇒ The use of RAID storage device are as follows:

- An improvement in cost - effectiveness because lower priced disks are used in large numbers.
- The use of multiple hard drives enable RAID to improve on the performance of a single hard drive.
- Increased computer speed and reliability after a crash - depending on configuration.

In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk. To search a record, we first proceed by index record and reach at the actual location of the data. If the data we are looking for is not where we directly reach by following the index, then the system starts sequential search until the desired data is found.



China	Besing	3705
Canada	Ottawa	3800
Russia	Moscow	6928
USA	Washington	3712

4. **What is RAID? Which RAID level would you prefer the best for safety of application and why? (2073 Magh)**

⇒ RAID or Redundant Array of independent Disks is a technology to connect multiple secondary storage devices and use them as a single storage device.

I would prefer the RAID 6 level because:

- It gives more unable capacity the more disks you add.
- It required more processing power.
- It can always protect against two simultaneous disk failures.

5. **How would you choose the best RAID level for your database server? (2071 Bhadra)**

⇒ The best RAID level 1 would choose for database server are:

1. **RAID level 0 (Disk stripping)**

   The data is divided into blocks and spread among all the disks in an arrau.

2. **RAID level 1 (Disk Mirroring)**

   Provides a redundant identical copy of a selected disk.

3. **RAID level 5 (Stripping With Parity)**

   The parity is also written across all the disks, which also means the data is redundant.

6. **Write the SQL syntax to create an index. (2071 Magh)**

⇒ CREATE INDEX index _ name

ON table _ name (column1, column2, ...);

7. **What are the advantages and disadvantages of mirroring? (2070 Bhadra)**

⇒ **Advantages**

1. Database mirroring supports full text catlogs.
2. Database mirroring architecture is more robust and efficient than database log shipping.
3. Does not require special hardware (such as shared storage, heart - beat connection) and clusterware, thus potentially has lower infrastructure cost.
4. It has automatic server failure and client failover mechanism.

Disadvantages

1. Mirror server / database in not available for user operation.
2. Automatic server failover may not be suitable for applicatio using multiple database.
3. Potential data lost is possible in asynchronous operation mode.

8. Explain limitation of static hashing. How extendable hashing overcome such limitation. **(2069Bhadra)**

⇒ Limitation of static hashing

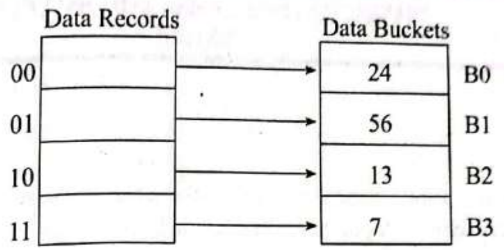
- Static hashing is not a good option for largely sized database.
- Time taken for this function is higher than normal, as the hash function has to go through all the addresses of the storage memory in order to perform operations in the BDMS system.
- It doesn't work well with scalable database.
- The ordering process is no efficient compared to other hashing techniques.

The dynamics hashing method is used to overcome the problems of static hashing like bucket overflow. In this method data buckers grow or shrinks as the record increases or decreases. This method is known as extendable hashing method.

E.g: Consider the following grouping of keys into buckets, depending on the prefix of their hash address.

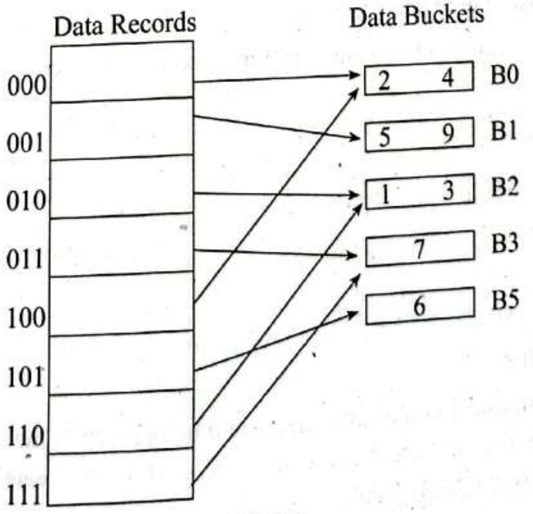
Key	Hash Address
1	11010
2	00000
3	11110
4	00000
5	01001
6	10111

The last two bits of 2 and 4 are 00. So it will go into bucket B0. The last two bits of 5 and 6 are 01, so it will go into bucket B1. The last two bits of 1 and 3 are 10, so it will go into bucket B2. The last two bits of 7 are 11, so it will go into B3.



Insert key 9 with address 10001 into the above structure:

- Since key 8 hash address 10001, it must go into the first bucket. But bucket B1 is full so it will get split.
- The splitting will separate 5, 9 from 6 since last three bits of 5, 9 are 001, so it will go into bucket B1, and the last three bits of 6 are 101, so it will go I int bucket B5.
- Key 2 and 4 are still in B0. The record in Bo pointed by the 000 and 100 entry because last two bits of both the entry are 00.
- Keys 1 and 3 are still in B2. The record in B2 pointed by the 010 and 110 entry because last two bits of both the entry are 10.
- Key 7 are still in BB3. The record in B3 pointed by the 111 and 001 entry because the two bits of both the entry are 11.



CHAPTER 7

TRANSACTION PROCESSING AND CONCURRENCY CONTROL

Transaction

A transaction is a single logical unit of works which accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.

ACID properties of transaction

A transaction in a database system must maintain Atomicity, consistency, Isolation and Durability - commonly known as ACID properties.

(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.

(ii) **Consistency**

The database must remain in a consistent state after any transaction.

(iii) **Isolation**

Transaction should be executed in isolation from other transactions (no locks)

(iv) **Durability**

After successful completion of a transaction, the changes in the database should persist.

E.g: Transaction to transfer 50 from account A to Account B.

```
read (A)
A: = A - 50
Write (A)
read (B)
B: = B + 50
Write (B)
```

Atomicity

If the transaction fails after step 3 and before step 6, the system should ensure that its updates are not reflected in the database, else an inconsistency will result.

Consistency

The sum of A and B is unchanged by the execution of the transaction.

(iii) **Isolation:** If between step 3 and 6, another transaction is allowed to access the partially updated database, it will see an inconsistent data base (the sum A + B will be less than it should be)

(iv) **Durability**

One the user has been notified that the transaction has completed, the update to the database by the transaction must persist despite failures.

Transaction States

Transaction goes through many different states throughout its life cycle. This states are called as transaction states.

Transaction States are as Follows:

(i) **Active State**

- This is the first state in the life cycle of a transaction.
- A transaction is called in an active state as long as its instruction are getting executed.

(ii) **Partially Committed State**

- After the last instruction of transaction has executed, it enters into a partially committed state.

(iii) **Committed State**

- After all the changes made by the transaction have been successfully stored into the database, it enters into a committed state.

(iv) **Failed State**

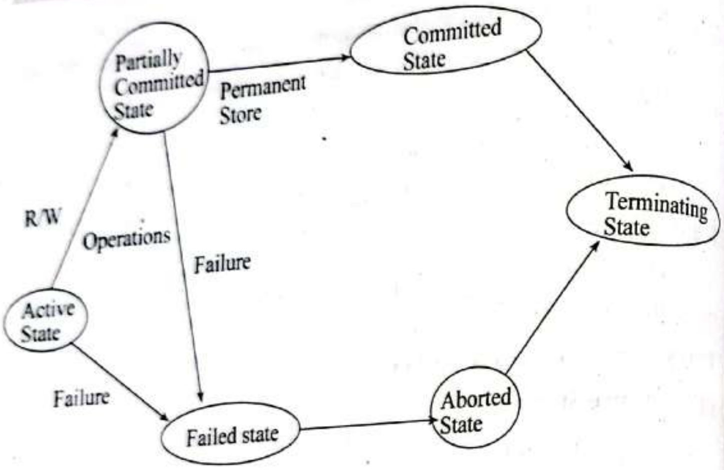
- When a transaction is getting executed in the active state or partially committed state and some failure occurs due to which it becomes impossible to continue the execution, it enters into a failed state.

(v) **Aborted State**

After the transaction has failed and entered into a failed state, all the changes made by it have to be undone. After the transaction has rolled back completely, it enters into an aborted state.

(vi) **Terminated State**

After entering the committed state or aborted state, the transaction finally enters into a terminated state which its life cycle finally comes to an end.



Concurrent Executions

The allowance to run multiple transactions in the system is known as concurrent executions.

Advantages of Concurrent Executions

(i) Increased processor and disk utilization

(ii) Reduced average Response Time

Concurrency control

It is the process of managing simultaneous execution of transactions in a DBMS without having them interfere with one another.

Problems of Concurrency control

Three major Problems: ,

1. Lost updates
2. Dirty Read (or Uncommitted data)
3. Unrepeatable Read (or inconsistent retrievals)

Lost Update Problem

A lost update problem occurs when two transactions that access the same database items have their operations in a way that makes the value of some \ database item incorrect. In other words, if transaction T_1 and T_2 both read a record and then update it, the effects of the first update will be overwriting by the second update.

Transactions	Time	Transaction - Y
—	t_1	—
Read A	t_2	—
—	t_3	Read A
Update A	t_4	—
—	t_5	Update A
—	t_6	—

Here,

- At time t_2, transaction - X reads A's value.
- At time t_3, transaction - Y reads A's value.
- At time t_4, transaction - X writes A's value on the basis of the value see at time t_2.
- At time t_5, transaction - Y writes A's value of the basis of the value seen at time t_3.
- So at time T_5, the update of Transaction - X is lost because Transaction Y overwrite it without looking at its current value.
- Such type of problem is known as los update problem as update mage by one transaction is lost here.

Dirty Read

The dirty read occurs in the case when one transaction updates an item of the database and then the transaction fails for some reason. The updated database item is accessed by another transaction before it is changed back to the original value.

Transactions	Time	Transaction - Y
—	t_1	—
—	t_2	Update A
Read A	t_3	—
—	t_4	Rollback
—	t_5	—

- A transaction T_1 updates a record which is read by T_2. If T_1 aborts then T_2 now has values which have never formed part of the stable database.
- At time t_2, transaction - Y writes A's value.
- At time t_3, transaction - X writes A's value.
- A's value back to that of prior to t_1.
- So, transaction - x now contains a value which has never become part of stable database.
- Such type of problem is known as Dirty Read problem, as one transaction reads a dirt value which not been committed.

Inconsistent Retrivals Problem

Inconsistent Retrievals problem is also known as unrepeatable read. Whe transaction calculates some summary function over a set of data while other transactions are updating the data, then the inconsistent retrieva problem occurs. A transaction T_1 reads a record and then does some oth processing during which the transaction t_2 updates the record. Now when th transaction T_1 reads the record, then the new value will be inconsistent wi the previous value.

E.g: Suppose two transactions operate on those accounts.

Account - 1	Account - 2	Account - 3
Balance = 200	Balance = 250	Balance = 150

Transactions	Time	Transaction - Y
——	t_1	——
Read Balance of Acc - 1 Sum < - - 200 Read Balance of Acc - 2	t_2	——
Sum < - - sum + 250 = 450	t_3	——
——	t_4	Read Balance of Acc - 3
——	t_5	Update Balanc of Acc - 3 150--➝ 150 - 50 --- 100
——	t_6	Read balance of Acc - 1
——	t_7	Update Balanc of Acc - 1 200--➝ 200 + 50 --- 250
Read Balance of Acc - 3	t_8	Commit
Sum < -- sum + 250 = 550	t_9	——

- Transaction - X is doing the sum of all balance while transaction - Y is transferring an amount 50 from Account - 1 to Account - 3.
- Here, transaction - X produces the result of 550 which is incorrect. If we write this produced result in the database, the database will become an inconsistent state because the actual sum is 600.

Here, transaction - X has seen an inconsistent state of the database.

Concurrency control Protocol

Concurrency control Protocol ensure atomicity, isolation and serializability of concurrent transactions. The concurrency control protocol can be divided into three categories:

1. Lock based protocol
2. Time - Stamp protocol
3. Validation based protocol

Lock - Based Protocol

In this type of protocol, any transaction cannot read or write data until it acquires an appropriate lock on it. There are two types of lock.

1. **Shard Lock**
   - It is also known as a read - only lock. In a shared lock, the data item can only ready by the transactions.
   - It can be shared between the transactions because when the transactions holds a lock, then it can't update the data on the data item.
2. **Exclusive Lock**
   - In the exclusive lock, the data item can be both reads as well as written by the transaction.
   - This lock is exclusive and in this lock, multiple transactions do not modify the same data simultaneously.

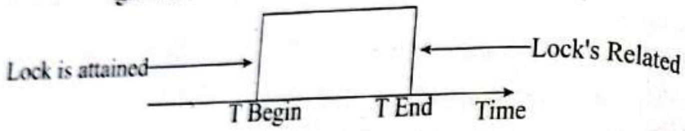
There are four types of lock protocol available:

1. **Simplistic Lock Protocol**

   It is the simplest way of locking the data while transaction. Simplistic lock - based protocols allow all the transactions to get the lock on the data before insert or delete or update on it. It will unlock the data item after completing the transaction.

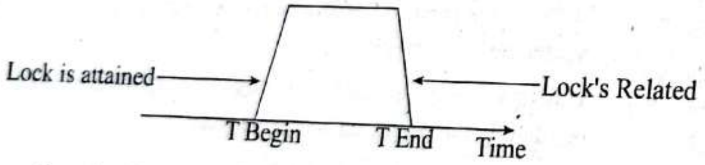
2. **Pre - Claiming Lock Protocol**
   - Pre claiming lock protocol evaluate the transaction to list all the data items on which they need locks.
   - Before initiating an execution of the transaction, it request DBMS for all the lock on all those data items.
   - If all the locks are generated then this protocol allows the transaction to begin when the transaction is completed then it all the lock.

- If all the locks are not granted then this protocol allows that the transaction to rolls back and waits until all the locks are granted.



3. **Two - phase locking (2PL)**

- The two - phase locking protocol divides the execution phase of the transaction into three points.
- In the first part, when the execution of the transactions starts, it seeks permission for the lock it requires.
- In the second part, the transactions acquires all the locks. The third phase is started as soon as the transaction releases its first lock.
- In the third phase, the transaction cannot demand any new locks. It only releases the acquired locks.



Growing Phase

In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released.

Shrinking Phase

In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired.

In the below example, if look conversion is allowed then the following phase can happen.

1. Upgrading of lock (From S (a) to X (a)) is allowed in growing phase.
2. Downgrading of lock (From X (a) to S (a)) must be done in shrinking phase.

	T₁	T₂
0	Lock - S (A)	
1		Lock - S (A)
2	Lock - X (B)	
3	—	—
4	Unlock (A)	
5		Lock - X (C)
6	Unlock (B)	
7		Unlock (A)
8		Unlock (C)
9	—	

The following way shows how unlocking and locking work with 2 - PL:

Transaction T_1:

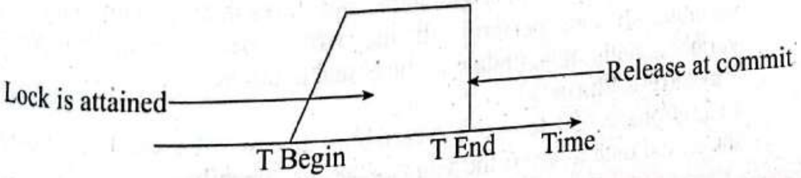
- Growing Phase: From step 1 - 3
- Shrinking Phase: From Step 5 - 7
- Lock point: at 3

Transaction T_2:

- Growing Phase: From Step 2 - 6
- Shrinking Phase: From Step 8 - 9
- Lock Point: at 6

4. **Strict Two - Phase Locking (Strict - 2PL)**

- The first phase of strict - 2PL is similar to 2PL. In the first phase, after acquiring all the locks, the transaction continues to execute normally.
- The only difference between 2PL and strict 2PL is that strict - 2Pl does not release a lock at a time.
- Strict - 2PL waits until the whole transactions to commit and then it release all the locks at a time.
- Strict - 2PL protocol does not have shrinking phase of lock release.



Lock Compatibility Matrix

- Lock Compatibility Matrix controls whether multiple transactions can acquire locks on the same resource at the same time.

	Shared	Exclusive
Shared	True	False
Exclusive	False	False

- If a resource is already locked by another transaction, then a new lock request can be granted only if the mode of the requested lock is compatible with the mode of the existing lock.
- Any number of transactions can hold shared locks on an item, but if any transaction holds an exclusive lock on item, no other transaction may hold any lock on the item.

Time Stamp Ordering Protocol

The time stamp - ordering protocol ensures serializability among transactions in their conflicting read and write operations. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp value of the transactions.

- The timestamp of transactions T_i is denoted as $T_s(T_i)$
- Read time - stamp of data item X is denoted by R - timestamp (X)
- Write time - stamp of data - item X is denoted by W - time stamp (X)

Time Stamp ordering protocol works as follows:

(i) If a transaction T_i issues a read (X) operation:

- If $T_s(T_i) <$ W - timestamp (X)
  → Operation rejected
- If $T_s(T_i) >=$ W - time stamp (X)
  → Operation executed
- All data - item timestamps updated.

(ii) If a transaction T_i issues a write (X) operation

- If $T_s(T_i) <$ R - timestamp (X)
  → Operation Rejected
- If $T_s(T_i) <$ W - timestamp (X)
  → Operation rejected and T_i rolled back
- Otherwise operation executed

Validation Based Protocol

Validation Phase is also known as optimistic concurrency control technique. In the validation based protocol, the transaction is executed in the following three phased.

1. **Read Phase**

   In this phase, the transaction T is read and executed. It is used to be read the value of various data items and stores them in temporary local variables It can perform all the write operations on temporary variables without an update to the actual database.

2. **Validation Phase**

   In this phase, the temporary variable value will be validated against the actual data to see if the violates the serializability.

3. **Write Phase**

   If the validation of the transaction is validated, then the temporary results is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Schedule

A sequences of instructions that specify the chronological order in which transactions are executed.

Serializability

A schedule S is serial if, for every transactions T participating in the schedule, all operations of T is executed consecutively in the schedule A schedule of n transactions is serializable if it is equivalent to some serial schedule of the same n transactions.

Two Types of Serializability

(i) Conflict Serializability

(ii) View Serializability

Conflict Serializability

A schedule is called conflict serializability if after swapping of non - conflicting operations, it can transform into a serial schedule. The schedule equivalent to a serial schedule.

Conflicting Operations

(i) Both belong to separate transactions

(ii) They have the same data item

(iii) They contain at least one write operation

E.g: Swapping is possible only if S1 and S2 are logically equal.

1. **T_1: Read (A) T_2: Read (A)**

T_1	T_2
Read (A)	
	Read (A)

Schedule S_1

Swapped →

T_1	T_2
	Read (A)
Read (A)	

Schedule S_2

Here, $S_1 = S_2$. That means it is non - conflict.

2. **T_1: Read (A) T_2: Write (A)**

T_1	T_2
Read (A)	
	Write (A)

Schedule S_1

Swapped →

T_1	T_2
	Write (A)
Read (A)	

Schedule S_2

Here, $S_1 \neq S_2$. This means it is conflict.

⇒ **Conflict Equivalent**

In the conflict equivalent, one can be transformers to another by swapping non - conflicting operations. In the given example, S_2 is conflict equivalent to S_1 (S_1 can be converted to $S_2$by swapping non - conflicting operations)

Two schedules are said to be conflict equivalent if and only if:

1. They contain the same set of the transaction.
2. If each pair conflict operations are ordered in the same way.

E.g: Non - serial schedule

T_1	T_2
Read (A)	
Write (A)	
	Read (A)
	Write (A)
Read (B)	
Write (B)	
	Read (B)
	Write (B)

T_1	T_2
Read (A)	
Write (A)	
Read (B)	
Write (B)	
	Read (A)
	Write (A)
	Read (B)
	Write (B)

Schedule S_2

Schedule S_2 is a serial schedule because, in this all operations of T_1 are performed before starting any operation of T_2 are Schedule S_1 can be transformed into a serial schedule by swapping non - conflicting operations of S_1.

After swapping of non - conflict operations the schedule S_1 becomes.

T_1	T_2
Read (A)	
Write (A)	
Read (B)	
Write (B)	
	Read (A)
	Write (A)
	Read (B)
	Write (B)

Since, S_1 is conflict serializable

View Serializability

A schedule will view serializable if it is view equivalent to a serial schedule.

If a schedule is conflict serializable, then it will be view serializable.

The view serializable which does not conflict serializable contains blind writes.

View Equivalent

Two schedules S_1 and S_2 are said to be view equivalent if they satisfy the following conditions:

1. **Initial Read**

An initial read of both schedule must be the same. Suppose two schedule S_1 and S_2. In schedule S_1, if a transaction $T_1$is reading the data item A, then in S_2, transaction T_1 should also read A.

T_1	T_2
Read (A)	
	Write (A)

Schedule S_1

T_1	T_2
	Write (A)
Read (A)	

Schedule S_2

Above two schedules are view equivalent because initial read operation in S_1 is done by T_1 and in S_2 it is also done by T_1.

2. **Update Read**

In schedule S_1, if T_i is reading A which is updated by T_j then in S_2 also, T_i should read A which is updated by T_j.

T_1	T_2	T_3
Read (A)		
	Write (A)	
		Read (A)

Schedule S_1

T_1	T_2	T_3
	Write (A)	
Write (A)		Read (A)

Schedule S_2

Above two schedules are not view equal because in S_1, T_3 is reading A updated by T_2 and in S_2, T_3 is reading A updated by T_1.

3. **Final Write**

A final write must be the same between both the schedules. In schedule S_1, if the transaction T_1 updates A at last then is S_2, final writes operations should also be done by T_1.

T_1	T_2	T_3
Write (A)		
	Read(A)	
		Write (A)

Schedule S_1

T_1	T_2	T_3
	Read (A)	
Write (A)		
		Write (A)

Schedule S_2

Above two schedule is view equal because final write operation on in S_1 is done by T_3 and in S_2, the final write operation is also done by T_3.

T_1	T_2	T_3
Read (A)		
	Write (A)	
		Write (A)
Write (A)		

Schedule's

With 3 transactions, the total number of possible schedule.

= 3! = 6

$S_1 = < T_1\ T_2\ T_3 >$

$S_3 = < T_2\ T_3\ T_1 >$

$S_5 = < T_3\ T_1\ T_2 >$

$S_2 = < T_1\ T_3\ T_2 >$

$S_4 = < T_2\ T_1\ T_3 >$

$S_6 = < T_3\ T_2 T_1 >$

Taking first schedule S_1:

T_1	T_2	T_3
Read (A)		
Write (A)		
	Write (A)	Write (A)

Schedule S_1:

Step 1:Final updation on data items

In both schedules S and S_1, there is no read except the initial read that's why we don't need to check that condition.

Step 2: Initial Read

The initial read operation in S is done by T_1 and in S_1, it is also done by T_1.

Step 3: Final write

The final write operation in S is done by T_3 and is S_1, it is also done by T_3. So, S and S_1 are view equivalent.

The first schedule S_1 satisfies all the three conditions so we don't need to check another schedule.

$T_1 \rightarrow T_2 \rightarrow T_3$

Recovrability of Schedule

Sometime a transaction may not execute completely due to a software issue, system crash or hardware failure. In that case, the failed transaction has to be roll back. But some other transaction may also have used value produced by the failed transaction. So we have to roll back those transactions.

T_1	T_1's buffer space	T_2	T_1's buffer space	Database
Read (A);	A = 6500			A = 6500
A = A - 500;	A = 6000			A = 6500
Write (A);	A = 6000			A = 6500
				A = 6000
		Read (A);	A = 6000	A = 6000
		A = A + 1000;	A = 7000	A = 7000
		Write (A);	A = 7000	
ailure Point		Commit;		
ommit;				

The above table 2 shows a schedule with two transactions. Transaction T_1 reads and writes A, and that value is read and written by transaction T_2. But later on, T_1 fails. Due to this, we have to roll back T_1. T_2 should be roll back because T_2 has read the value written by T_1.

As t has not committed before T_1 commits so we can roll back transaction T_2 as well. So, it is recovered with cascade roll back.

Recoverable with cascading roll back

The schedule will be recoverable with cascading rollback if T_j reads the updated value of T_i commit of T_j is delayed till commit of T_i.

T_1	T_1's buffer space	T_2	T_1's buffer space	Database
Read (A);	A = 6500			A = 6500
A = A - 500;	A = 6000			A = 6500
Write (A);	A = 6000			A = 6500
Commit;				A = 6000
				A = 6000
		Read (A);	A = 6000	A = 6000
		A = A + 1000;	A = 7000	A = 7000
		Write (A);	A = 7000	
		Commit;		

The above table 3 shows a schedule with two transactions. Transaction T_1 reads and write A and commits, and that value is read and written by T_2. So, this is a cascade less recoverable schedule.

Deadlock in DBMS

In deadlock is a condition where two or more transactions are waiting indefinitely for one another to give up locks. Deadlock is said to be one of the most feared complications in DBMS as no task ever gets finished and is in waiting state forever.

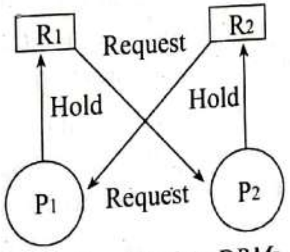


Fig: Dead lock in DBMs

Consider the following two transactions:

T_1 : Write (X) T_2: Write (X)

Write (Y) Write (Y)

Schedule with deadlock:

T1	T1
Lock - X on X Write (X)	
	Lock - X on Y Write (Y) Wait for lock - X on X
Wait for lock - X on Y	

Deadlock Prevention

Deadlock Prevention protocols ensure that the system will never enter into a deadlock state.

(i) First approach ensures that no cyclic waits can occur by ordering the requests for locks, requiring all locks to be acquired together. It requires that each transaction locks all its data item before execution. Moreover, either all are locked in one step or none are locked. There are two main disadvantages.

- It is often hard to predict, before the transaction begins, what data items need to be locked.
- Data item utilization may be very low, since many of the data items may be locked but unused for a longtime.

(ii) Second approach performs transaction rolled back instead of waiting for a lock, whenever the wait could potentially result in a deadlock. It uses preemption and transaction roll back. These schemes use time stamps just for deadlock prevention.

(a) Wait - die scheme (non - preemptive)

When transaction T_i requests a data item currently held by T_j. T_i is allowed to wait only if it has a timestamp smaller than that of T_3. Otherwise T_i rolled back (dies)

- For example, suppose that transaction T_1, T_2 and T_3 have time stamp 20, 30 and 40 respectively. If T_1 request a data item held by T_2, then T_2 will wait. If T_3 requests a data item held by T_2, then T_3 will be rolled back.

(b) Wound - wait schema (Preemptive)

- When transaction T_i requests a data item currently held by T_j . T_i is allowed to wait only if it has a timestamp larger than that of T_j otherwise T_i rolled back.

- For example, suppose that transactions T_1, T_2 and T_3 have timestamps 20, 30 and 40 respectively. If T_1 request a data item held by T_2, then the data item will b preempted from T_2 and T_2 will be rolled back. If T_3 requests a data item held by T_2, then T_3 will wait.
- Both in wait - die and wound - wait schemes, a rolled back transaction is restarted with its original time stamp. Older transactions thus have precedence over newer ones, and starvation is hence avoided.

(c) Time - Based Schemes

- A transaction waits for a lock only for a specified amount of time. After that, the wait times out and the transactions is rolled back.
- Thus deadlock are not possible.
- Simple to implement, but starvation is possible. Also difficult to determine good value of the time out interval.

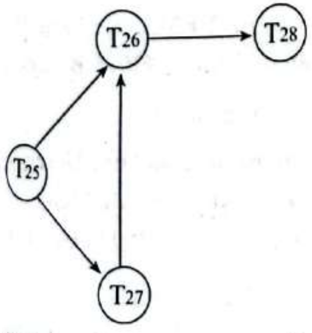
Deadlock Detection

Deadlock can be described as a wait for graph, which consist of a pair a = (V, E).

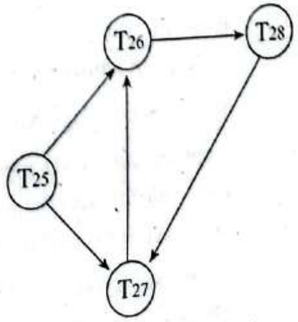
Where,

V is set of vertices (all transaction in the system) E is a set of edges, each element is an ordered pair $T_i \rightarrow T_j$.

If $T_i \rightarrow T_j$ is in E, then there is a directed edge fro, T_i and T_j, implying that T_i is waiting for T_j to release being held by T_j, then the edge T_i T_j is inserted in the wait - for graph. This edge is removed only when T_j, is no longer holding a data item needed by T_i. The system is in a deadlock state if and only if the wait - for graph has a cycle.



Wait for graph without a cycle



Wait for graph with a cycle

Deadlock Recovery

When deadlock is detected:

- Some transaction will have to rolled back (made a victim) to breaks deadlock. Select that transaction as a victim that will incur minimum cost.
- Roll back - determine how for to roll back transaction.
  - **Total roll back:** Abort the transaction and then restart it.
  - More effective to roll back transaction only as for as necessary to break dead lock.
- Starvation happens is same transactions is always chosen as victim. Include the number of rollbacks in the cost factor to a avoid starvation.

Old Question Solution

1. **What is transaction? What are ideal properties of transaction? [2076 Baisakh]**

⇒ See in page no.138

2. **Describe strict two-phase locking protocol(2PL) (2076 Baisakh)**

⇒ See in page no 145

3. **Define transaction and explain various states of transaction with a transition diagram. Describe about two phase locking protocol for concurrent transaction along with its limitations. [2075 Bhadra]**

⇒ 1st part See in page No138. 2nd part see on page no 144

4. **Describe about two phase locking protocol for concurrent transaction along with its limitations. [2075 Bhadra]**

⇒ See in Page No. 144

5. **Explain the possible transaction states in DBMS. Explain the concept of conflict searilizablity with an example. [2075 Baisakh]**

⇒ 1st part See in Page No: 139 2nd part see in page no. 147

Explain the ACID properties of a database transaction. Describe how conflict serializablity differs from the view serializablity for concurrent execution of transactions. [20074 Bhadra]

1st part see in page no 138

2nd part see in page no 147

7. **What are schedules? Describe the concept of view seriazablity for concurrent execution of transactions. [2073 Bhadra]**

⇒ 1st part see in page no 147

2nd part see in page no 148

8. **How deadlocks arise while processing transactions? Explain the deadlock prevention strategies. [2073 Bhadra]**

⇒ See in page no 151 and 152

9. **Define ACID properties of a transactions. Describe the concept of conflict serializablity for concurrent execution of transactions. [20073 Magh]**

⇒ 1st part see in page no 138

2nd part see in page no 147

10. **How two phase locking protocol helps in concurrency control? Explain. [2073 Magh]**

⇒ See in page no 144

11. **Explain different states of a transaction along with state transaction diagram. Explain conflict serializablity with example. [2072 Ashwin]**

⇒ 1st part see in page no 139

2nd part see in page no 147

12. **Explain briefly two phase locking protocol for Concurrency Control. [2072 Ashwin]**

⇒ See in page 144

13. **What is transaction? Explain ACID properties with examples. [2072 Magh]**

⇒ See in page 138

14. **Describe the different types of locks used for concurrency control. Draw the lock compatibility matrix. [2072 Magh]**

⇒ First Part: See in page no. 143; Second Part: See in page no. 146.

15. **Explain Atomicity and Isolation properties of database transaction. Describe the concept of conflict serializablity for concurrent execution of transactions. [2071 Bhadra]**

⇒ 1st part see in page no 138

2nd part see in page no 147

16. **What do you mean by seriazablity of a schedule? What do you understand by granularity of looking for concurrency control?** **[2071 Magh]**

⇒ 1st part see in page no 147

The granularity of locks in a database refers to how much of the data is locked at one time.

17. **What is transaction? What are the properties of a transaction should satisfy in a database system?** **(2071 Magh)**

⇒ See in page no 138

18. **During its execution a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which a transaction may pass. Explain why each state transaction may occur.** **(2070 Bhadra)**

⇒ See in page No 139

19. **How two phase locking protocol helps in avoiding deadlock? Explain with examples.** **(2070 Bhadra)**

⇒ See in page no 144

20. **What do you understand by the ACID properties of transaction? Explain with examples.** **(2070 Magh)**

⇒ See in page no. 138

21. **Explain conflict seriazablity with example.** **(2069 Bhadra)**

⇒ See in page no. 147

22. **Differentiate between fine granularity and coarse granularity locking in multiple granularity locking protocol.** **(2069 Bhadra)**

⇒

Fine Granularity	Coarse Granularity
1. Locking when necessary and unlocking as soon as possible.	1. Locking for long period of time.
2. They provide maximum parallelism.	2. They provide less parallelism.
3. Overhead of taking and releasing lock.	3. No overhead of taking and releasing lock.

CHAPTER 8

CRASH RECOVERY

Failure classification

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

1 Transaction failure

2. System crash

3. Disk failure.

1 Transaction failure

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

Reasons for a transaction failure could be

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

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

2. System crash

- System failure can occur due to power failure or other hardware or software failure. Example: operating system error.
- Fail-stop assumption: non-volatile storage contents are assumed not to be corrupted.

3. Disk failure.

- It occurs when hard-disk drives or storage drives used to fail frequently. It was a common problem in the early days of technology evolution.
- Disk failure occurs due to the formation of bad sectors, disk head crash, and unreachably to the disk or any other failure, which destroy all or part of disk storage.

Storage structure

The storage structured can be divided into two categories.

(i) Volatile storage

(ii) Non-volatile storage

(i) **Volatile storage:** As the name suggests, a volatile storage cannot survive system crashes. Volatile storage are placed very close to the CPU, normally they are embedded onto chipset itself. For example, main memory and cache memory are examples of volatile storage. They are fast but can store only a small amount of information.

(ii) **Non-volatile storage:** These memory are made to survive system crashes. They are huge in data storage capacity, but slower in accessibility. Example may

Note:

Stable storage:

- Information residing in it never lost
- a mythical form of storage that survives all failures.

Let's assume there is a transaction to modify the city of a student. The following logs are written for this transaction.

- When the transaction is initiated, then it writes 'start' log.

  <Tn, start>

- When the transaction modifies the city from 'Noida' to Bangalore', then another log is written to the file.

  <Tn, city, 'Noida', Banglore'>

- When then transaction is finished, then it writes another log to indicate the end of transaction.

  <Tn, commit>

There are two approaches to modify the database:

1. Deferred database modification:

- The deferred modification technique occurs of the transaction does not modify the database until it has committed.
- In this method, all the logs are created and stored in the sable storage, and the database is updated when transaction commits include hard-disks, magnetic tapes, flash memory and non volatile (battery backed up) RAM.

Log-Based Recovery

- The log is sequence of records log of each transaction is maintained in some stable storage so that any failure occurs, then it can be recovered from there.
- If any operation is performed on the debase, then it will be recorded in the log.
- If any operation is performed on the database, then it will be recorded in the log.
- But the process of storing the logs should be done before the actual transaction is applied in the database.

2. Immediate database modification

- The immediate modification technique occurs if database modification occurs while the transaction still active.
- In this technique, the database is modified immediately after every operation. It follows an actual database modification.

* Rectory using log records.

When the system is crashed, then the system consults he log to find which transactions need to be undone and which need to be redone.

1. If the log contains the record <Ti, start> and <Ti, commit> or <Ti commit>, then the transaction Ti needs to be redone.
2. If the log contains record < In, start> but does not contain the record either <Ti, commit> or <Ti, abort>, then the transition Ti needs to be undone.

* **Recovery and Atomicity**

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

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

- It should check the status of all the transaction, which were being executed.
- A transaction may be in the middle of some operation: the DBMS must ensure the atomicity of the transaction in this case.
- It should check whether the transaction can be completed now or it need to be rolled back.

- No transaction would be allowed to leave the DBMS in a inconsistent state.

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

* Maintain the log of each transaction, and writing them onto some stable storage before actually modifying the database.

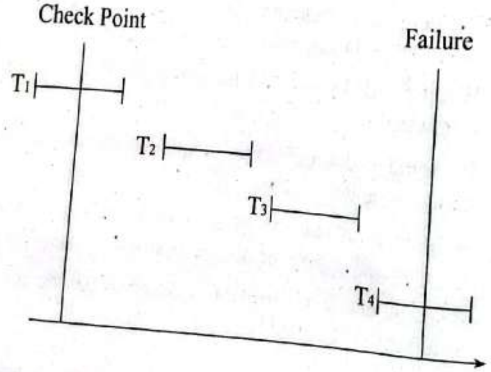
- Maintaining the shadow paging, where the changes are done on a volatile memory and later, the actual database is updated.

* **Check point**

- The check point is a type of mechanism where all the previous logs are removed from the system and permanently stored in the storage disk.
- The checkpoint is, like a bookmark. While the execution of the transaction, such checkpoints are marked, and the transaction is executed the using the steps of the transaction, the log files will be created.
- When it reaches to the checkpoint, then the transaction will b updated into the database, and till that point, the entire log file will be removed from the file. Then the log file is updated with the new step of transaction till next checkpoint and so on.
- The checkpoint is used to declare a point before which the DBMP was in consistent state, and all transaction were committed.

Recovery using checkpoint

In the following manner, a recovery system recovers the database from this failure:



- The recovery system read log files from end to short. It reads log files from T4 and T1.
- Recovery system maintains two lists, a redo-list, and undo-list.
- The transaction is put into redo state if the recovery system sees a log with <Tn, start>, and <Tn, commit> or just <Tn, commit>. In this redo-list and their previous list, all th transaction are removed and then redone before saving their logs.
- For example: In the log file, transaction T_2 and T_3 will have <Tn, strt> and <Tn, commit>. Th T1 transaction will have only. <Tn, commit> in the log file. That is why the transaction is committed after the checkpoint is crossed. Hence it puts T1, T2 and T3 transaction into redo list.
- The transaction is put into undo state if the recovery system sees a log with <Tn, start> but no commit or abort log found. In the undo-list, all the transaction are undone, and their logs are removed.
- For example: Transaction T4 will have <Tn, start>. So, T4 will be put into undo list since this transaction is not yet complete and failed amid.

* **Shadow paging**

- Shadow paging a technique for providing atomicity and durability in database systems.
- Shadow paging is a copy - on -write technique for voiding in - plane updates of pages. Instead, when a page is to be modified, a shadow page is allocated.
- Since the shadow page has no references (from other page on disk), it can be modified liberally, without concern for consistency constraints, etc. When the page is ready to become durable, all pages that referred to the original are updated to refer to the new replacement page instead. Because the page is 'activated' only when if is ready, it is atomic.
- This increases performances significantly by avoiding many writes on hotspots high up in referential hierarchy at the cost to high commit latency.

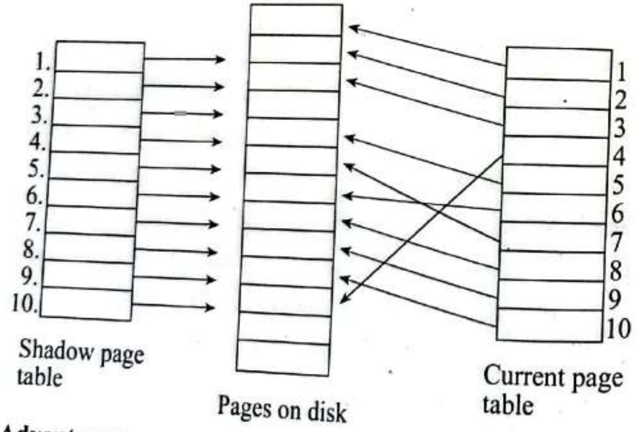
Shadow paging considers:

1. The databases is petitioned into fixed length blocks referred to as PAGES.
2. Page table has n entries - one for each database page.
3. Each contain pointer to a page on disk (1 to 1st page on database and so on ....).

The idea is to maintain 2 pages tables during the life of transaction.

1. The current page table
2. The shadow page table. When the transaction starts, both page tables are identical.

1. The shadow pager table is never chinned over the duration of the transaction.
2. The current page table may be changed when a transaction performs a write operation.
3. All input and output operations use the current page table to locate database pages on disk.



Advantages:

- No overhead for writing log records.
- No undo/No Redo algorithm.
- Recovery is faster.

Disadvantages

- Data gets fragmented or scattered
- After every transaction completion database page s containing old version modified data need to be garbage collected.
- Hard to extend algorithm to allow transaction to run concurrently.

Advance Recovery algorithm

- Support for high-concurrency lacking techniques, such as these used for B^+-tree concurrency control, which release lack early.
- Supports "logical undo"
- Recovery based on "repeating history', whereby recovery executes exactly the same actions as normal processing.
- Including redo of log records of incomplete transaction, followed by subsequent undo.
- key benefits
  - → support logical undo
  - → easier to understand/show correctness

Logical undo logging

- Operations like B^+ - tree insertions and deletion release locks early.
  - → They cannot be undone restoring and values (physical undo),since once a lock is released other transactions may have updated the B^+ - tree.
  - → Instead, insertions (resp. deletion) are undone by exciting a deletion (resp. insertion) operation (known as logical undo)
- For such operations, undo log records should contain the undo operation to be executed.
  - → Such logging is called logical undo logging, in contrast to physical undo logging.
  - → Operations are called logical operations.
- Other examples:
  - → delete of tuple, to undo inset of tuple.
  - → allows early lock release on space allocation information.
  - → subtract amount deposited, to undo deposit.
    - allows early lock release on bank balance.

Physical redo

- Redo information is logged physical (that, ps, new value for each written even for operations with logical undo.

  → Logical redo is very complicated since database state on disk may not be "operation consistent" when recovery starts.

Operation logging

- Operation logging is done as follows:

1. When operation starts log <Ti, Oj, operation begin) Here, Oj is a unique identifier of the operation instance.
2. While operation is executing, normal log records with physical redo and physical undo information are logged.
3. When operation completes, <Ti, Oj, operation-end, U> is logged when contains information needed to perform a logical undo information.

- Example: insert of (key, record-id) pair (KS, RID7) into index I9

<TI, 01, operation-begin>

.....

<T_1, X, 10, k5>
<T_1, Y, 45, RID7>
} Physical redo of steps in insert

<T_1, 01, operation-end, (delete I0, KS, RID7)>

- If cash/rollback occurs before operations completes:
  → If the operation - end log record is not found and
  → the physical undo information is used to undo operation.
- If crash/roll back occurs after the operations completes:
  → the operation - end log record is found, and in this case
  - logical undo is performed using U; the physical undo information for the operation is ignored.
- Redo of operation (after crash) still uses physical redo information.

TXn Rollback

Rollback of transaction Ti is done as follows:

- Scan the logs backwards:

1. If a log records <Ti, X,V_1, V_2>
   is found perform the undo and log a special redo-only log records <Ti, X, V_1>

2. If a <Ti, Oj, operation - end, U> record is found.
   → Rollback the operation logically using the undo information U.
   - At the end of the operation roll back, instead of logging an operation end record, generate a record.

   <Ti, Oj, operation - abort>

   → Skip all preceding log records for Ti until the record <Ti, Oj, operation - begin> is found.
3. If a redo-only record is found ignore it
4. If a <Ti, Oj, operation - abort> record is found:
5. Stop the scan when the record
   <Ti, start> is found.
6. Add a <Ti, abort> record to the log.

:: Some points to note:

(i) Cases 3 and 4 above can occur only it the database crashes while a transaction is being rolled back.

(ii) Skipping of log records as in case 41s important to prevent multiple rollback of the same operation.

Crash Recovery:

The following actions are taken when recovering from system crash.

1. **(Redo phase):** Scan log forward from last <checkpoint L> record till end of log

(i) Repeat history by physically re-doing all updates of all transactions:

(ii) Create an undo-list during the scan as follows

- Undo-list is set to L initially.
- Whenever <Ti start> is found Ti is added to undo-list.

This brings database to state as of crash, with committed as well as uncommitted transactions having been redone.

Now undo-list contains transaction that are incomplete, that is have neither neither committed nor been fully rolled back.

(2) (Undo phase):

Scan log backwards, performing undo on log records of transactions found in undo-list.

- Log records of transactions being rolled back are processed as described earlier, as they are found.
- When <Ti start> is found for a transaction Ti in undo-list, write a <Ti abort> log record.
- Stop scan when <Ti start> records have been found for al T_i in undo-list.

This undoes the effects of incomplete transactions (those with neither commit nor abort log record). Recovery is now complete.

1. Output all log records in memory to stable storage.
2. Output to disk all modified buffer blocks.
3. Output to log on sable storage a <checkpoint L> record.

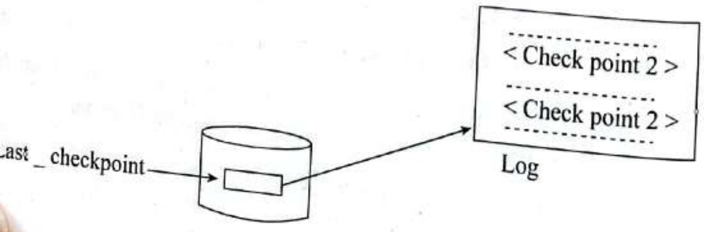
Transaction are not allowed to perform any action while check pointing is in progress.

- Fuzzy check pointing allows transactions to progress while the most time consuming parts of check pointing are in progress
- Fuzzy check pointing is done as follows:

1. Temporally stop all updates by transactions
2. Write a <checkpoint L> log record and force log to stable storage.
3. Note list M of modified buffer blocks.
4. Now permits transactions to proceed with their actions.
5. Output to disk all modified buffer blocks in list M.

- Blocks should not be updated while being output.
- Follow WAL: all log records pertaining to a block must be output before the block is output.

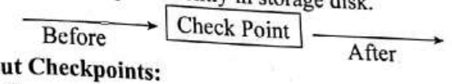
6. Store a pointer to the check point record in a fix position last _ checkpoint on disk.



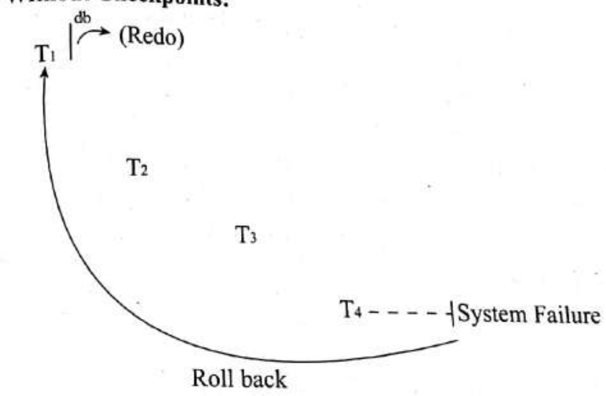
Q. **Explain checkpoint. How does it help in reducing the amount of time required during recovery?**

Checkpoints:

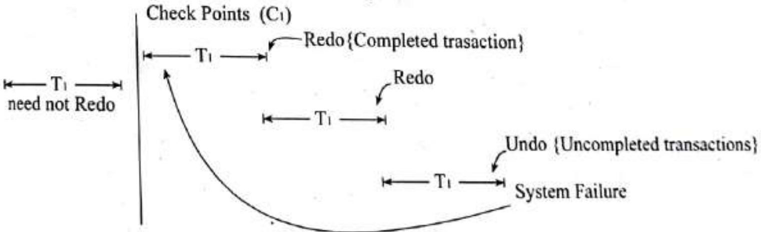
It is the mechanism where all the previous logs are removed from the system and permanently in storage disk.



Without Checkpoints:



- Extra-time in processing entire log file.
- Redoing of unnecessary transaction.



Old Question Solution

1. **Define term Recovery and Atomicity in database. Consider the following log contents when a crash occurs. Explain how recovery would be done for each state.** **[2+4] [2076 Baisakh]**

<T_0 start>	**<T_0 start>**	**<T_0 start>**
<T_0, A, 1000, 950>	**<T_0, A, 1000, 950>**	**<T_0, A, 1000, 950>**
<T_0, B, 2000, 2050>	**<T_0, B, 2000, 2050>**	**<T_0, B, 2000, 2050>**
	<T_0 commit>	**<T_0 commit>**
	<T_1 start>	**<T_1 start>**
	<T_1, C, 700, 600>	**<T_1, C, 700, 600>**
		<T_1 commit>
(a)	**(b)**	**(c)**

⇒ **First Part:** See in theory 159

Recovery actions in each case above as:

(a) **Undo (To):** B is restored to 2000 and A to 1000, and log records.

< To, B, 2000 >, < To, A, 1000 >, < To, abort > are Written out.

(b) **Redo (To) and undo (T_1):** A and B are set to 950 and 2050 and C is restored to 700. Log records < T_1, C, 700 >, < T_1, abort > are written out.

(c) **Redo (To) and redo (T_1):** A and B are set to 950 and 2050 respectively. Then C is set to 600.

2. **Write the different types of failures that may occur in system. Differentiate between shadow paging and log-based recovery.**

[3+3] [2075 Bhadra][2073 Bhadra]

⇒ **For First Part:** See in theory 157

The difference between shadow paging and log - based recovery are as follows:

Shadow Paging	Log-based Recovery
1. Recovery is faster due to elimination of overhead of log - record output.	1. Recovery process is slower.
2. Locality property is lost	2. Locality property is retained.
3. Garbage collection is lost necessary.	3. Garbage collection is not necessary.

3. **Explain the idea of log based recovery.**

[2075 Baisakh][2072 Ashwin]

⇒ See in theory page no. 159

4. **What is the purpose of implementing check points in data recovery mechanism?** **[2] [2074 Bhadra]**

⇒ Checkpoint - Recovery is a common technique for imbuing a program or system with fault tolerant quantities, and grew from the ideas used in systems which employ transactions processing. Its allows systems to recover after some fault interrupts the system, and causes the task to fail, or be aborted in some way.

5. **What are the recovery actions performed if failure arises at the end of the given transaction states?**

<T_0 start>	**<T_0 start>**
<T_0, A, 1000, 950>	**<T_0, A, 1000, 950>**
<T_0, B, 2000, 2050>	**<T_0, B, 2000, 2050>**
	<T_0 commit>
	<T_1 start>
	<T_1, C, 700, 600>

⇒ (a) **Undo (To):** B is restored to 2000 and A to 1000, and log records.

< To, B, 2000 >, < To, A, 1000 >, < To, abort > are Written out.

(b) **Redo (To) and undo (T_1):** A and B are set to 950 and 2050 and C is restored to 700. Log records < T_1, C, 700 >, < T_1, abort > are written out.

6. **What is stable storage? Explain the log based recovery mechanism.** **[2+4] [2073 Magh]**

⇒ **Stable Storage**

This is said to be third form of storage structure but it is same as non volatile memory. In this case, copies of same non volatile memories are stored at different places/. This is because, in case of any crash and data loss, data can be recovered from other copies. This is even helpful if there one of non - volatile memory is last due to fire or flood. It can be recovered from other network location. But there can be failure while taking the backup of DB into different stable storage devices.

For Second Part: See in theory 159

7. **Explain redo phase and undo phase of log based failure recovery mechanism.** **[2072 Magh][2070 Magh][2069 Bhadra]**

⇒
- Undo of a log record < T_1, T_2, V_2 > writes the old value V_1 to X.
- Redo of a log record < T_1, T_2, V_2> writes the new value V_2 t0 X.
- Undo and Redo of Transactions.

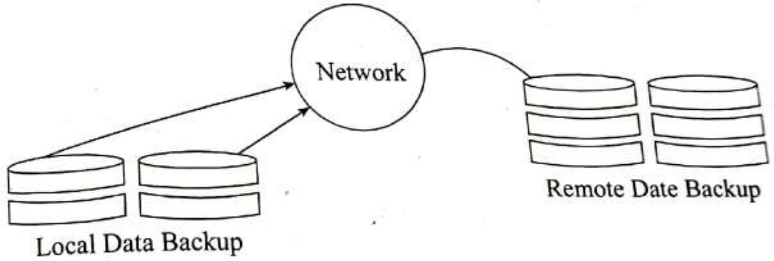
- Undo (T_1) restores the value of all data items updated by T_i top their old values, going backwards, from the last log record for T_i.
  - → Each time a data item x is restored to its value V a special log record (called redo - only) < T_i, X, V > is written out.
  - → When undo of transaction is complete, a log record < T_i, abort > is written out.
- Redo (T_1) sets the value of all data time updated by T_i to the new values, going forward from the first log record for T_i.
- The undo and redo operations are used in several different circumstances:
  - The undo is used for transactions roll back during normal operations.
  - The undo operation are used during recovery from failure.
- We need to deal with the case where during recovery from failure another occurs prior to the system having fully recovered.
- When recovering after failure:
  - Transaction T_i needs to be undone if the log.
    - → Contains the to be undone if the log
    - → But does not contain either the record <T_i commit> or < T_I abort >
- Transaction T_i needs to be redone if the log
  - → Contains the records < T_i Start >
  - → But does not contain either the record < T_i, commit > or < T_I abort >
- It may seem storage to redo transaction T_i if the record < T_i, abort > record is in the log. To see why this works note that if < T_i abort > is in the log, so are the redo - only records written by the undo operation. Thus, the end result will be to undo T_i 's modification is this case. This sight redundancy simplifies the recovery algorithm and enables faster overall recovery time.

8. **Briefly explain the idea of a stable storage. Explain the architecture of a remote backup system. [3+3] [2071 Bhadra]**

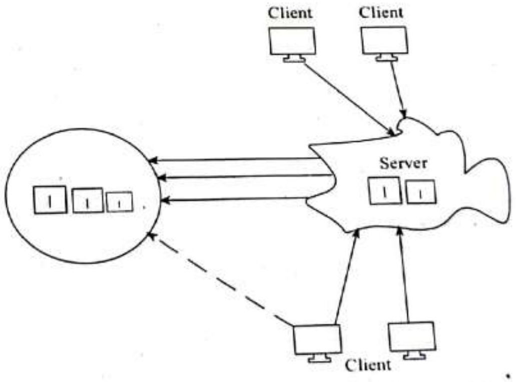
⇒ **For First Part:** See in above Q.No. 6

Remote Backup System

Remote backup system provides a sense of security in case the primary location where the database is located gets destroyed. Remote backup can be offline or real - time or online. In case it if offline, it is a maintained manually.



Architecture



9. **Distinguish between immediate modification and differed modification in the context of log-based database recovery. What is the significance of checkpoints in a log? [4+2] [2071Magh]**

⇒ The difference between immediate update and deferred - update in the context of log based database recovery are as follows:

Deferred	Immediate update
1. The changes are not applied immediately to the database.	1. The changes are applied directly to the database.
2. The log file contains all the changes that are to be applied to the database.	2. The log file contains both old as well as new values.

3.	In this method one roll back is done all record of log file are discarded and no changes are applied to the database.	3.	Concept of shadow paging is used in immediate update method.
4.	Concept of buffering and caching are used in deferred update method.	4.	Concept of shadow paging is used in immediate update method.

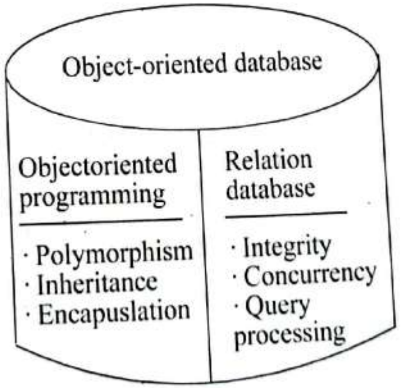
For Second Part: See in above Q.No. 4.

□□□

CHAPTER 9

ADVANCE DATABASE CONCEPTS

- An object oriented database is a collection of objected-oriented programming and relational database.
- An object-oriented database is a database that is based on object-oriented programming, The data is represented and stored in the forms of objects.
- A database is data storage. A software system that is used to manage database called a database management system (DBMS).
- Object database are commonly used in application that requires high performance, calculations, and faster result. Some of the common applications that use object database are real-time system, architectural and engineering for 3D modeling telecommunications, and scientific products, molecular science, astronomy.
- Object oriented database model is combination of OUP principles (inheritance, Encapsulation, polymorphism) and Relational database features (integrity, concurrency, Query processing).



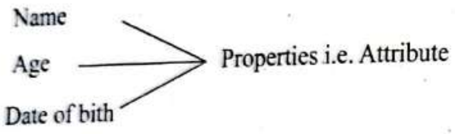
- Both data and their relationship are organized or contained in a single structure known as object.

  Object includes information about relationship between the facts within object, as well as information about its relationship with other object.
- Object oriented database models also known as semantic data model.

Components of OODM

- **An object** is the abstraction of th real world entity.
- **Attributes:** It describes the property of object.

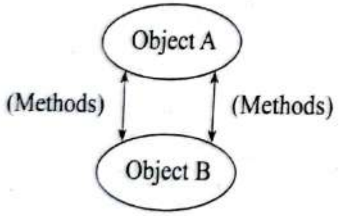
Person - object/class



- **Class:** Objcts that are similar in characterize are grouped in class. So class is a collection of similar objects with shared structure (Attributes) and Behaviour (Method).

Method:

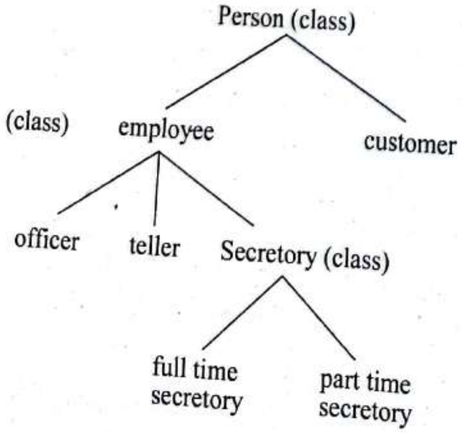
Methods represents a real world action such as finding a selected person's Name, changing person name or printing a person address.



Class are organized in class hierarchy and it resemble son upside-Down tree in which each class has only one parent (Head).

Inheritance

Inheritance is an object ability to inherit the attributes and methods (messages) of the class above it.



UML and XML are mainly used to represent the structure of class, objects, attributes and methods.

Advantage of OODBM

- Complex data sets can be save and retrieved quickly and easily.
- Objects IDS are assigned automatically.
- Works well with object-oriented programming languages.

Disadvantages

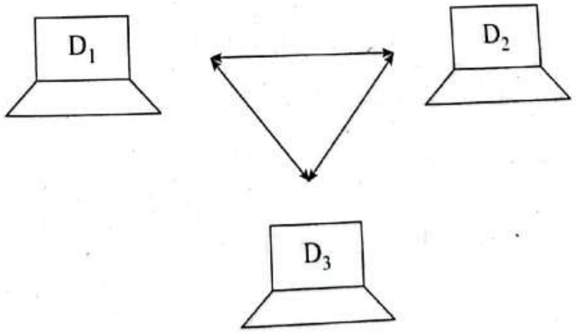
- Object databases are not widely adopte3d.
- In some situations, the high complexity can cause performance problems.

Distributed Database Model

- Distributed database is a system in which storage devices are not connected to a common processing unit.

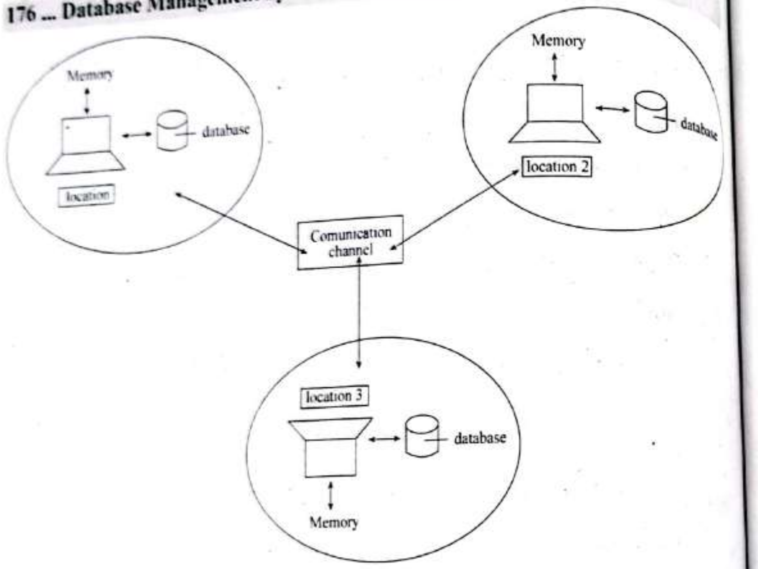
  Common processing unit i.e. each storage devices has its own processing unit.

- Database is controlled by Distributed database management system and data may be stored at the same location or spread over the interconnected network. It is loosely coupled system.



Distributed database system is loosely coupled system so that every system can access data whenever it required.

- A typical example of distributed data based system in which communication channel is used to communication with the different locations and every systems has its own memory of database.



The fig. shows that each system has is own memory and locations.

Mainly two types of distributed data system.

1) Heterogeneous Database

2) Homogenous Database

- Homogeneous Database

→ All the sits use identical DBMS and operating System

→ The sites use very similar software

→ Much easier to design and manage.

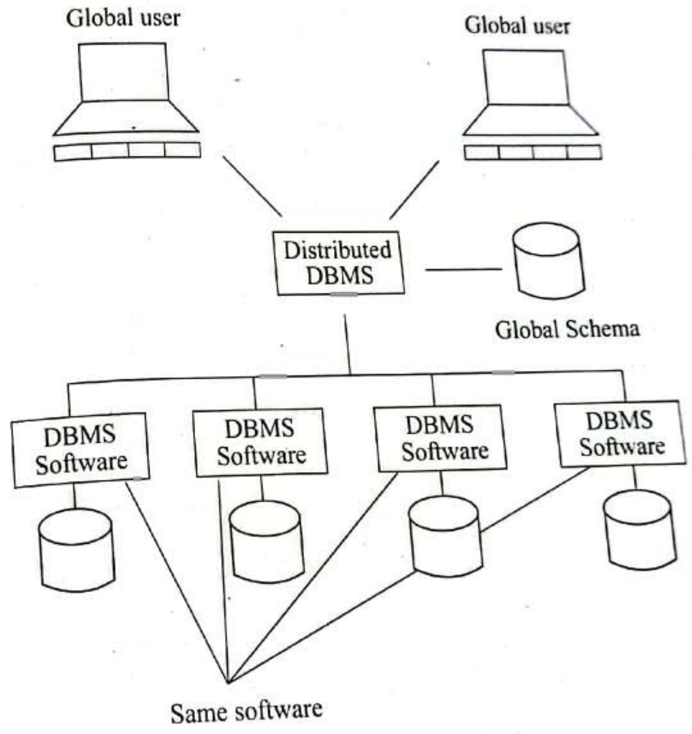
→ It appears to users as a single system.

Advantages:

easy to use, manage, design

Disadvantages:

→ Difficult for most organizations to force a homogeneous environment.



Homogenous database

2) Heterogeneous DBMS

- In this type of database, Different data center may run different DBMS products, with possibly different underlying dada models.
- Different sizes may use different sachems and software

Advantages

- Huge data can be stored in one global center from different databases center.
- Remote access sis done using the global schema.
- Different DBMS may be used at each node.

Disadvantage

- Difficult to design & manage.

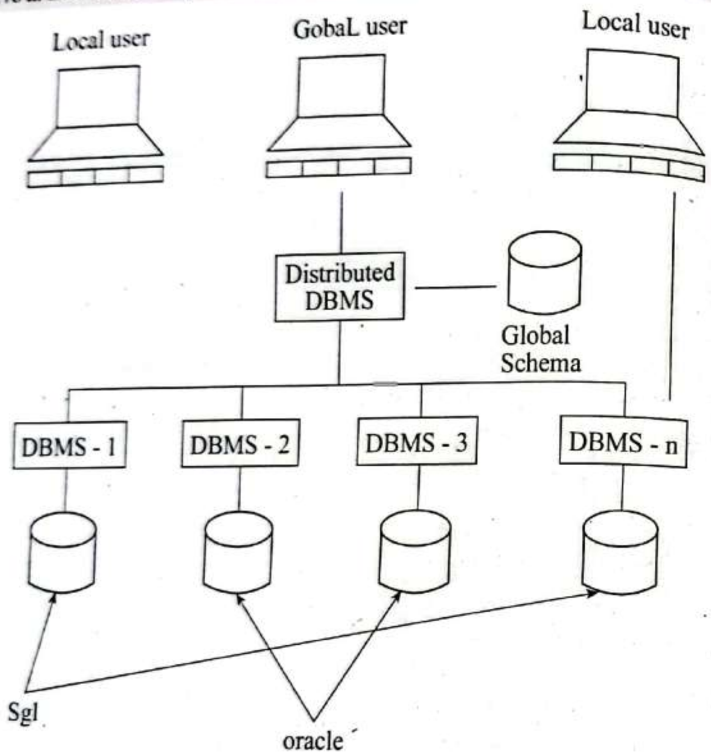


Fig:- Heterogenous DBMS

Difference between Homogenous and Heterogeneous Distribute DBMS

1)	All site have same DBMS and operating system	2)	All sites does not have same DBMS.
2)	Easy to install	2)	Complex to install
3)	Easy to administrate	3)	Complex to administrate.
4)	Provides full data access facility	4)	Provides limited data access facility
5)	Updating data is easy	5)	Updating data is complex
5)	Conceptual schemas is same	6)	Conceptual schemas is different
)	Provides high performance	7)	Provides relatively law performance.
)	Easy transaction processing	8)	Difficult to process transaction.
)	Communication between two different sites is easy.	9)	Communication between two different types require transaction.

Distributed data storage

It is used to refer to distributed data base where users store information in more than one site or node.

Two approaches

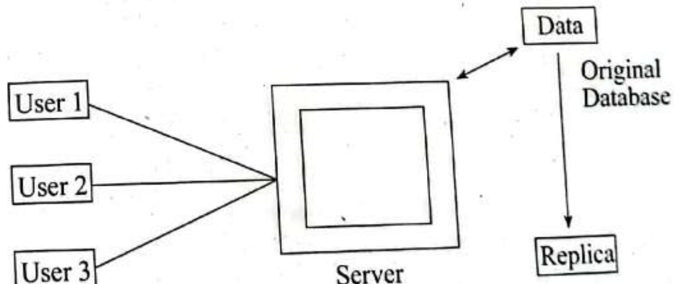
1) Data replication 2) Data fragmentation

1) Data replication

Data replication is the process of storing the same data in multiple locations to improve data variability and accessibility and to improve system reliance and reliability. It is simply copying data from a database from one server to another so that all the users can share the same data without any inconsistency.

Full replication

- Replication of whole database at every site in the distributed system. This will improve the availability of the system because system can continue to operate as long as at least one site is up.



Full replication

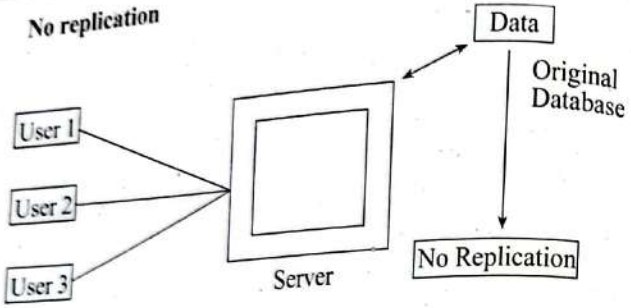
Advantage of full replication

- High avaiulability of DATA
- Improves the performance of retrieval of global queries as the result can be obtained locally from any of local sites.
- Faster execution of queries.

Disadvantage of full replication

- Concurrency is difficult to achieve in full replication.

No replication



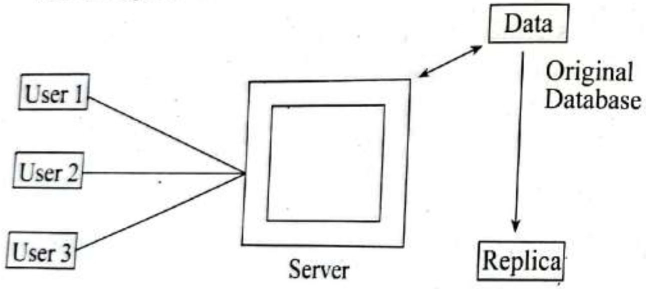
Advantage

- The data can be easily recovered.
- Concurrency can be achieved in no replication.

Disadvantage

- The data is not easily available as there is no.

Partial Replication



Partial Replication

In this type of replication some fragments of the database may be replicated whereas others may not.

Advantages of partial replication

- The number of copies of fragment depends upon the importance of data.

Advantages of DATA REPLICATION

- To provide a consistent copy of data across all the database nodes.
- To increase the availability of data
- The reliability of data of increased through data replication
- Data replication supports multiple users and gives high performance.

- To remove any data redundancy, the databases are merged and solve databases are updated with outdated or incomplete data.
- To perform faster execution of queries.

Disadvantage of DATA REPLICATION

- More storage space is needed as storing the replica of same data at different sites consume more space.
- Data Replication becomes expensive when the replica at all different sites need to the be updated.
- Maintaining data consistency at all different sites involves complex measures.

2) **Data Fragmentation:**

Data fragmentation occurs when a collection of data in memory is broken up into many places that are not close together. It is typically the result of attempting to insert a large object into storage that has already suffered external fragmentation.

Dividing the whole table data into smaller chunks and storing them in different the distributed Database management is called data fragmentation.

Advantages

→ It allows easy usage to Data

→ It makes mast frequently accessed set of data near to the user.

Fragmentation can be of three types.

1) horizontal

2) vertical

3) hybrid (combination of horizontal and vertical)

Parallel database system

Parallel database system improves performance of data processing using multiple resources in parallel like multiple CPU and disk are used parallel.

It also performs may parallelization operations like data loading and query processing.

Goal of parallel Databases

1. **Improve performance**

The performance of the system can be improved by connecting multiple CPU and disk in parallel. Many small processors can also be connected in parallel.

2. **Improve availability of data:**

   Data can be copied to multiple locations to improve the availability of data.

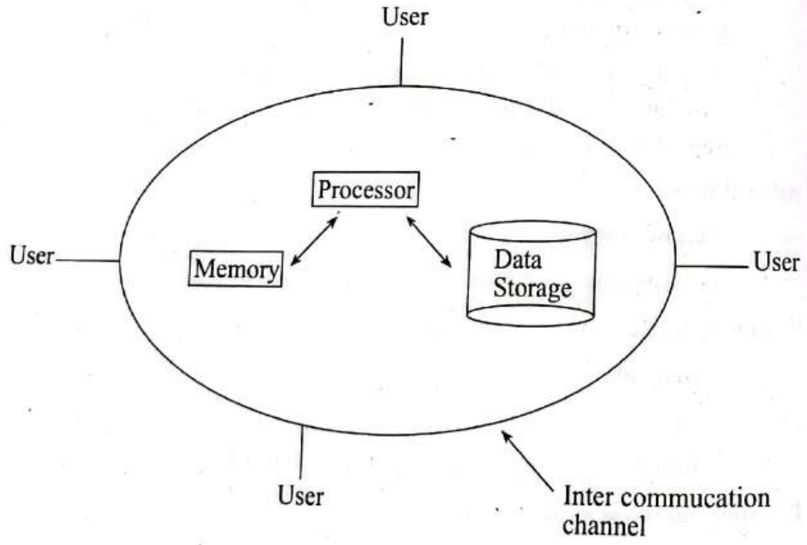
   For example: If a module contains a relation (table in database) which is available then it is important to make it available from another module.

3. **Improve reliability:**

   Reliability of system is improved with completeness, accuracy; and availability of data.

4. **Provide distributed accesses of data:**

   Companies having many branches in multiple cities can acdcss data with the help of parallel database system.



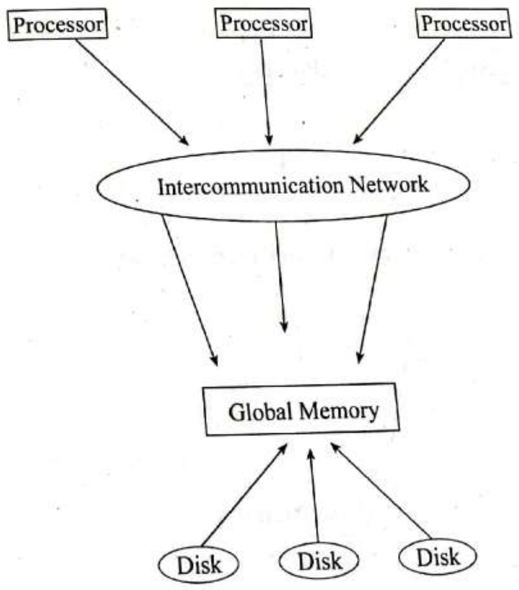
Parallel database system

Parallel Database Architecture

- Shared memory - processor share a common memory
- Shared disk - processor share a common disk
- Shared nothing - processor share neither a common memory nor common disk.
- Hierarchical - hybrid of the above architecture.

(i) **Shared memory**

- Shared memory system uses multiple processors which is attached to a global shared memory via intercommunication channel, or communication bars.
- Shared memory system have large amount of cache memory at each processors, so referencing of the shared memory is avoided.
- If a processor performs a write operation to memory location, the data should be updated or removed from that location.



Shared memory system in parallel database

Advantages of shared memory system

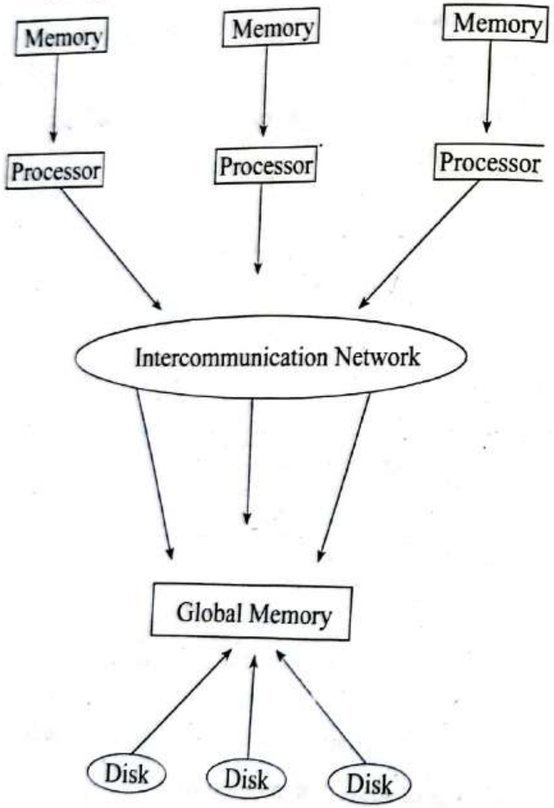
- Data is easily accessible to any processor
- One processor can send message to other efficiently.

Disadvantage of shred memory system.

- Waiting time of processor is increased due to more number of processor.
- Band width problem.

(ii) Shred Disk System

- Shared disk system uses multiple processors which are accessible to multiple disks via intercommunication channel and every processor has local memory.
- Each processor has its own memory so that data sharing is efficient.
- The system built around this system are called as clusters.



Shared disk system in parallel database

Advantages of shared disk system

- Fault tolerance is achieved using shared disk system.

  Fault tolerance: If a processor

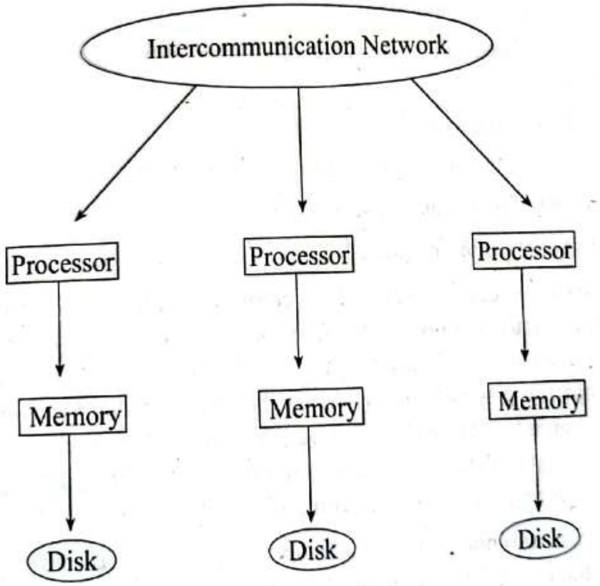
  or its memory fails, the other processor can complete the tasks. This is called as fault tolerance.

Disadvantage of shred disk system.

- Shared disk system has limited scalability as large amount of data travels through interconnection channel.
- If more processors are added the existing processors are slowed down.

(iii) Shared nothing disk system

- Each processor in the shared nothing system has its own local memory and local disk.
- Processors can communicate with each other through intercommunication channel.
- Any processor can act as a server to serve the data which is stored on local disk.



Shared nothing disk system in Nepal Database

Advantages of shared nothing disk system.

- Number of processors and disk can be connected as per the requirement in share nothing disk system.
- Shared nothing disk system can support. For many processor, which makes the system more scalable.

Disadvantages of shared nothing disk system

- Data partitioning is required in shred nothing disk system.
- Cost of communication for accessing local disk is much higher.

(iv) Hierarchical system or Non-uniform Memory Architecture.

- Hierarchical model system is a hybrid of shared memory system shared disk system and shared nothing system.
- Hierarchical model is also known as Non-uniform Memory Architecture (NUMA)
- In this system each group of processor has a local Memory. But processors from other groups can access memory which is associated with the other group in coherent.
- NUMA uses local and remote memory (Memory from other group), hence it will take longer time to communicate with each other.

Advantages of NUMA

- Improves the scalability of the system
- Memory bottleneck (shortage of memory) problem is minimized in this architecture.

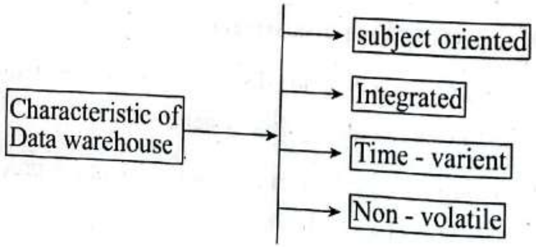
Disadvantages of NUMA

The cost of the architecture is higher compared to other architectures.

* Concept of data warehouse database.

What is Data Warehouse?

- A data warehouse essentially combines information from several sources into one comprehensive database. For, ex, in the business world, a data warehouse might incorporate customer information from company's point-of-sale systems (the cash registers), its website the, its mailing lists and its comment cards, Alternatively, it might incorporate all the information about employs, including time cards, demographic data, salary information, etc.
- A data warehouse is subject-oriented, integrated, time-variant, in volatile collection of data in support of management's decision making process.



1. **Subject-oriented**

A data warehouse is always a subject oriented as it delivers information about a theme instead of organizations current operations. It can be achieved on specific them. That means the data work housing process if proposed to handle with a specific them is more defined. These themes can be sales, distributions, marketing etc.

A data warehouse never put emphasis only current operations. Instead, it focuses on demonstrating and analysis of data to more various decisions.

2. **Integrated**

- Data warehouse is constructed by integrating multiple heterogeneous sources.
- Data processing are applied to ensure constancy.
- The data ware is a centralized, consolidated database that integrated data derived from the entire organization.
  - Multiple sources.
  - Diverse sources
  - Diverse formats.

3. **Time-variant:**

- In this data is maintained via differen4t intervals of time such as weekly, monthly, or annually etc. It founds various time limit which are structured between the large data sets and are held in online transaction process. (OLTP) . The time limits for data warehouse are wide-ranged, than that of operation system.
- The data warehouse represents the flow of data through time.
- Can contain projected data form statistical models.
- Data is periodically uploaded then time-dependent data is recomputed.

(iv) **Nonvolatile**

- Once data is entered it is never removed. That means the data resided in data warehouse is permanent. It also means that data is not eronosed or deleted when new data is inserted.
- Data is read-only and refreshed at particular intervals.
- Two types of data operations done in data warehouse are:
  - Data loading
  - Data Access.

Function of path warehouse:

It works as a collection of data and here is organized by various communities that endures the feature to recover the data function.

The major function are:

1. Data consolidation
2. Data clearing
3. Data integration

Rules that define a Data warehouse

1. Data warehouse and operational environment are separated.
2. Data are integrated.
3. Contains historical data over a long time horizon.
4. Snapshot data captured at al given point in time.
5. Subject-oriented
6. Mainly read-only
7. Development is data drive: the classical approach is process driven.
8. Contains data with several levels of details.
9. Characterized by read-only transactions to very large data sets.
10. Traces data resources, transformation and storage.
11. Metadata are a critical component of this environment.
12. Contains a charge-back mechanism for resources usage.

Phases of Data Mining

(a) Data understanding: Review the data that you have, document ii, identify data management and daa quality issues. Tasks for this phase include.:

- Gathering data
- Describing
- Exploring
- Verifying quality

(b) Data preparation: Get your data ready to use for modeling. Tasks for this phase include.

- Selecting data
- Clearing data
- Constructing
- Integrating

(c) Modeling: Use mathematical techniques to identity patterns within your data. Tasks for this phase include.

- Selecting technique
- Designing tests.
- Building models
- Accessing models

(d) Evaluation:

Review the pattern you have discovered and assess their potential for business use. Tasks for this phase include:

- Evaluating results
- Reviewing the process
- Determining the next process

(e) Deployment:

Put your discoveries to work in everyday business. Task for this phase include:

- Planning deployment (your methods for integrate data mining discoveries into use)
- Reporting find results.
- Reviewing final results.

Need for data warehouse:

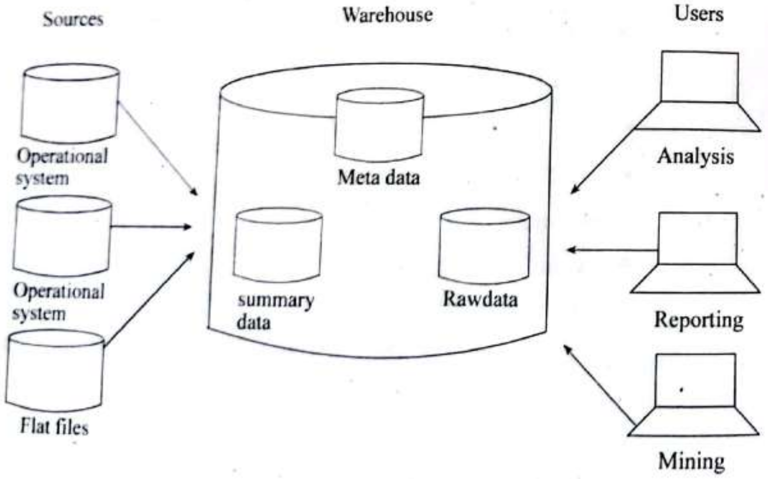
- Industry have huge amount of operational data. Knowledge worker wants to turn this data into useful information. This information is used by them to support strategic decision making.
- It is a platform for consolidated historical data for analysis.
- It is stores data of good quality so that knowledge worker can make correct decisions.

- From business perspective.
  - → It is latest marketing weapon
  - → Helps keep customers by learning more about their needs.
  - → Valuable tool in today's competitive fast evolving world.

Application Area of Data warehouse.

- OLAP (Online Analytical Processing)
- DSS (Decision Support Systems)
- Data mining

* **Data Warehouse Architecture.**



Operational system:

An operational system is a method used in data warehousing to refer to a system that is used to process the day-to-day transactions of an organization.

Flat Files:-

A Flat Files system is a system of files in which transactional data is stored, and every file in the system is must have a different name.

Meta data:

A set of data that defines and gives information about other data.

Metadata used in Delaware house for a variety of purpose, including:

Meta data summarizes necessary information about data, which can make finding and work with particular instances of data more accessible. Metadata is used to direct a query to most appropriate data sources.

Clients:

- Query and reporting tools
- Analysis tools
- Data mining tools

Spatial Database System

- A spatial database is a database that is optimized for storing and querying data that represents objects defined in a geometric space.
- Spatial data, which means data related to space.
- Spatial data includes location, phase size and orientation.
- A spatial database system is a database system with additional capabilities for handling spatial data.

→ **Application area:**

- Geographical information shyster

  Eg. data: road network and places of interest.

  uses: driving directions.
- Environmental system·

Eg: → data: land cover, climate, rainfall, and forest fire

usage: find total rainfall precipitation

- Corporate Decision - support systems

  Eg→ data: store locations and customer locations.

Usage: determine the optimal location for a new store

- Battlefield soldier Monitoring systems

  Eg: data: locations of soldier (w/wa medial equipments)

  Usage: monitor soldier that may need help from each on with medical equipment.

a) What is the difference between Database and Data warehouse?

(i) Database is a collection of related data that represent some elements of the real world whereas data warehouse is an information system that stores historical and commutative data form single or multiple sources.

(ii) Database is designed to record data whereas the data warehouse is designed to analyze data.

(iii) Database is application - oriented - collection of data whereas Data warehouse is the subject - oriented collection of data.

(iv) Data uses online Transactional (OLTP) whereas data warehouse uses online analytical processing (OLAP)

(v) Database tables and joins are complicated because they are normalized. Whereas Data warehouse tables and joins are easy because they are demoralized.

(vi) ER modeling techniques are used for designing Database whereas data modeling technique are used for designing Data Warehouse.

Old Question Solution

1. **Write Short notes on:** **[2076 Baisakh, 2073 Bhadra]**

⇒ Distributed databases

See on page No. 175

Write about data warehouse with its components.

[2075 Bhadra, 2071 Bhadra]

See on page No. 186

Write about spartial database.

[2075 Bhadra, 2075 Baisakh, 2074 Bhadra, 2073 Bhadra]

See on page No. 191

Explain homogenous and heterogenous distributed database.

[2075 Baisakh]

See on page No. 176

Describe briefly about object oriented database. **[2073 Magh]**

See on page No. 173

6. **Explain the differences between homogenous and heterogenous distributed database.** **[2073 Magh]**

⇒ See on page No. 178

7. **Explain the importance of data warehouse in decision making. Write the application areas of spartial database.** **[2072 Ashwin]**

⇒ See on page No. 189

Applications areas of spartial database

1. GIS (Geographic Information System)
2. MMIS (Multimedia Information System)
3. CAD (Computer Aided Design)

These can be used for capturing, storing, manipulating, analysing, managing and presenting all types of spartial or geographical data.

8. **What is the significance of object - oriented database? Briefly explain parallel database architecture.** **[2072 Magh]**

⇒ Significance of OOD are:

- Complex data sets can be saved and retrieved quickly and easily.
- Objects IDS are assigned automatically.
- Works well with object oriented programming language.

2nd part: See on page No. 181

9. **Write short notes on the following**

Types of distributed database **[2071 Bhadra]**

⇒ See No. 175

10. **Write Short notes on the following.**

a) **Object - relational mapping** **[2071 Magh]**

When we work with an object - oriented system, there is a mismatch between the object model and the relational database. RDMS represent data in a tabular format whereas object - oriented languages, such as JAVA or C# represent it as an interconnected graph of objects. Consider the following JAVA class with proper constructed and associated public function:

```
Public Class Employee {
        private int id;
        private string First - name;
        private string last - name;
        private int salary;
    Public Employee ( ) { }
    Public Employee (string F name, string l name, int salary)}
        this First - name = F name;
        this last - name = l name;
        this salary = salary;
    }
    Public int get Id (){
        return id;
    }
    Public string get First Name ( ) {
        return First - name;
    Public string get Last Name ( ) {
        return last - name;
    }
    Public int get salary ( ) {
        return salary:
    }
}
```

Consider the above objects are to be stored and retrieved in to the following RDBMS, table-

```
Create table EMPLOYEE (
        id INT NOT NULL auto - increment,
        First - name VARCHAR (20) default NULL,
        last - name VARCHAR (20) default NULL,
        salary INT default NULL,
        primary key (id)
);
```

b. **Parallel database architecture** **[2071 Magh]**

⇒ See in theory page No. 181

11. **Briefly explain properties of distributed database.** **[2070 Bhadra]**

⇒
- A collection of logically related shared data.
- n the data is split into a number of fragments.
- Fragments may be replicated.
- Fragments/replicas are allocated to sites.
- Each DBMS participates in at least one global application.

12. **Briefly explain horizontal and vertical fragmentation in distribute databases.** **[2071 Magh]**

⇒ **Vertical Fragmentation**

In vertical Fragmentation, the field or columns of a table are grouped into fragments. In order to maintain reconstructiveness, each fragments should contain the primary key field (S) of the table. Vertical Fragmentation can be used to enforce privacy of data.

Horizontal Fragmentation

Horizontal fragmentation groups the tuples of a table in accordance to values of one or more fields. Horizontal Fragmentation should also confirm to the rule of reconstructiveness. Each horizontal fragment must have all comes of the original base table.

13. **Write Short notes on Data Warehouse and associated applications.**

⇒ See in page No. 186

14. **What is object - oriented database? Explain briefly** **(2069 Bhadra)**

⇒ See in page No. 173

15. **Explain the benefit of parallel database?**

⇒ See in page No. 181

□□□

Bibliography

H. F. Korth and A. Silberschatz, "*Database system concepts*", McGraw Hill, 2010.

A. K. Majumdar and P. Bhattacharaya, "*Database Management Systems*", Tata McGraw Hill, India, 2004.

https://www.geeksforgeeks.org

https://www.javapoint.com/

https://www.tutorialspoint.com/

□□□