DBMS

NOTES

Sathya Technologies

2nd Floor, Sri Sai Arcade,
Beside: Aditya Trade Centre,
Ameerpet, Hyderabad - 500038
Tel: 65538958, 65538968

DB

DBMS

Theory — 80"

oracle — 50"

Ramakrishna

Database: collection of inter related items / fields

Management System:

Database Vendors

Backend Products
- ACCESS - MS - O/P
- FoxPro - X

oracle
SQL
DB2
Int.
I

DBMS -

"Database is a collection of inter-related items / fields / attribute."

Ex:

Student

Key attribute (unique value)	Stno	stname	marks	CName
	18	SUBBU	XX	XX
	20	XXX	XXX	XXX

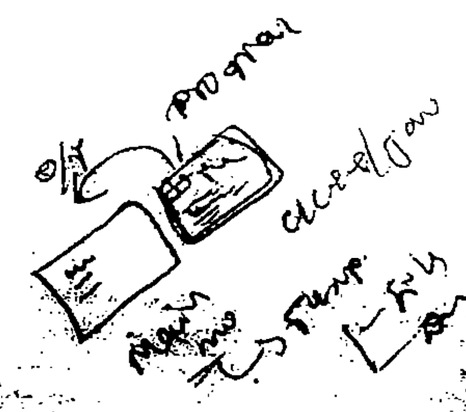
non key attributes

Stno → is called Key attribute where as Stname, marks, cname are called non key attributes.

Management System:

"It is one of the application software use to represent the data base. Technically to perform various operations on data base we perform three operations

1. Insertion
2. Deletion
3. Modification.



Sai

IMPORTANCE OF DBMS:

• We can write any Programme in any language who's output is by default represented in Main Memory. The data which is stored in main memory is temporary data.

• In Information Technology there are two ways to store the data permanently. They are ..

ii) using files
ii) using database.

If we store the data permenently in the form of files then the data can be manipulated by any unauthorised users. hence industry is always recommended data should not be stored in the form of files permenently, but it is always recommended to store the data permenently in the form of database only. Since most of the popular products (oracle, SQL Server, DB2, Ingres, informix, ...) provides 100% security in the form of <u>username</u> & <u>passwords</u>.

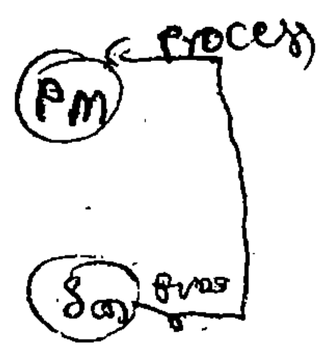
To develop any database effectively we must perform the following sequence of steps.

1. Data Analysis
2. Data Design and application Development
3. Concurency
4. Security issues
   a. OS level Security
   b. Database level Security
   c. Application " "
   d. Physical/human level Security

Sairam

Traditional File Processing System (TFPS vs DBMS)

When we write Any Programme in any language who's Result ~~by~~ is by default Represented in Mainmemory of the Computer. we know that anything is stored in main memory which is Temporary data.



To represent the data Permanently, They are two approaches in information Technology

1. Traditional File Processing System (using Files)
2. DBMS (using database)

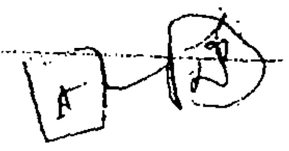
1. DATA Redandancy:

Data Redundancy is nothing but duplication of information. Repetation of Same information in many number of files is known as Data Redandancy.

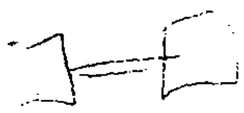
. in TFPS Data Redandancy is more hence we get more memory

Space, more [illegible], less Performance, and in Consistant Result.

In DBMS- Environment, Data Redandancy is minimized. Since DBMS is acting as an interface between Application Programme and Data-base Since Data Redandance is minimised, we get less memory space, less Excution time, more Perfor-mance and Consistance Result.



Hence in DBMS- environment any modification we do on one-Database which will be Reflected on other Databases Also Provided Their exist a Common Data between multiple Databases.



Example

2. Data Intigrity: (Intigrity = correctness of data)

Replace → (call by value)

Intigrity is nothing but Maintaing Correct-ness of the Data.

- In the case of TFPS Data Intigrity Problems are more Since an application Programme is directly manipulating the file. i.e One application Programme Can modified only one file and Those modification never Reflected back to other files.

A.p manipulated / changes by → file

In the Case of DBMS Data Intigrity is maintained Effectively if we

Write any Programme in DBMS Programme and it can change the Content of our DB then those changes which be reflected in all the databases where ever there is the Common information.

3. Data concurrent Access: (concurrent - simultaneous access)

Concurrent Access is nothing but accessing the data Simultaneously in the case of TFPS if any file data is concurrently accessed by multiple application Programes then their is a Possibility of getting in consistent Result.

In the case of DBMS concurrent access can be achieved effectively.

Ex:- Let us assum their exist a balance variable and its initial value is zero. To application Programmes decided to update the balance variable with thousand and two thousand Respectively.

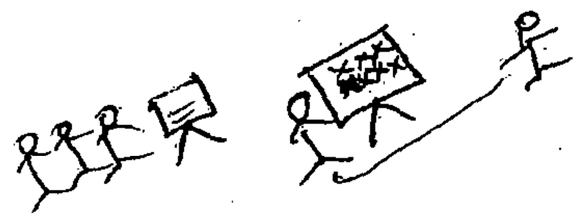
to both the programmes started at same time and in the case of TFPS the final value in the balance variable is either 1000 or 2000 but not 3000. which is inconsistant Result.

unlocked

The balance variable will be application programe 2 and finally balance variable will be locked. after completion of application programme 2, the final value in the balance is 3000. which is consistant Result. Hence in the case of DBMS concurrent access can be achieved in the case of TFPS concurrent access can't be achieved.

Note: when System crashes, there is a possibility of losing integrity of data in the case of TFPS there is No crash Recovery Techniques where as in the case of DBMS we can Recover the state of the database.

4. Security:



Security is one of the important constraint in information technology to protect the contendential information from unallocaised users.

In the case of TFPS, if we store the contendential information in the form of files. Then that data can be accessed by any unallocaised users. Since TFPS Never provides any security on files.

In the case of DBMS, if we store the Contedential information in the form of database Then unallorised users can't Access the Contedential information.

Since DBMS provides 100% Security in the form of usernames and passwords.

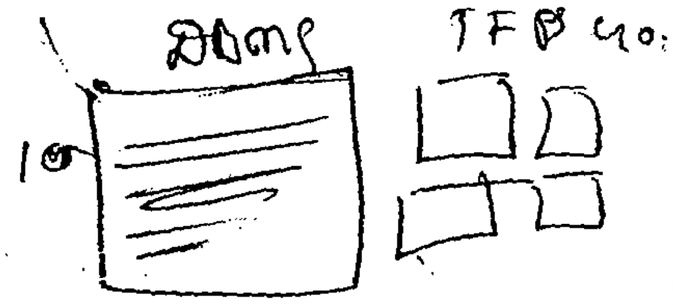
Data independence: it nothing

→ It not input a common Data must be given an input to many number of application programes at the same time. i.e Even then there exist n-number of Application Programmes it is required to create one instance of data only.
(one time memory creation)

In the case of TFPS, for each and every Application Programme it is required to have a separate file, i.e

If we have n-number of Application Programes which are related to same file then it is manadatory to create n-number of files with the same Data to n-number application Programmes. Hence in TFPS Data independence is un-able to achive. But it is achived only Data dependence.

In the case of DBMS, one Data base is enough for n-number of Application Programmes Hence in the case of DBMS data independence is achived.



Application Development time :-

When we write any application Programme in any Environment it is Recommanded to take less application Development time.

In the case TFPS, it their Exist n-number of application Programmes for n-number of files then it takes more time to create n-number of file for Each & Every Application Programme. Hence in TFPS application Development time is always more.

In the case of DBMS irrespective of number of application Programmes (Queries), it is enough to create the Databases only once. Hence Developing an application in DBMS environment always takes less time.

Database Administration:

Data Administration is nothing but Running the Database.

- In the case of TFPS, There is no concept of file administration. Hence files never Provides Effective performance in real world.

• In the case of DBMS, the database data can be accessed in different ways according to the requirement of an application programs. The functionality called database administrator is performed by DBA in real world.

8. Effecient Accessing of the Data:

when we store the data either in the file or DB. the data must be retrieved effectively.

→ In the case of TFPS, we are unable to main-tain column names of the files, no key attributes can be maintained and we need to write a big application programme to retrieve the data from the files. which leads to poor accessing of the data.

• In the case of DBMS, the database can be created with well defined column names, required key attributes and we can develop a small application programme to retrieve the data from data base. which leads to effective accessing of the data.

Hence in DBMS we have various sophi-cated techniques to retrieve the data from database. and it is not the case in TFPS.

n the following terms with the suitable
mples?

Entety/ 2. entity/set 3. attribute

4. relationship set 5. key/ vs non key/attribute

6. funtional Dependency/ 7. Schema.

1. **Entity:** collection of fieldvalues is known as an entity.

eg: consider the following student Database.

entity set

Student

Sno	Sname	Smarks	Cname
10	abc	80	x
11	pqr	70	y
12	abc	70	z
13	abc	80	

fig(1)

attributes

entity / record

Tuple / row

tyset

(10, abc, 80, x) - is an entity/. entity/ is also known as Record (or) Row (or) Tuple.

entity (Record)

entity/set:

Collection of Entities is known as entity/set

Ex: In fig(1) Student is called Entity/set

Attribute:

The well defined column name of an entity/set is known as attribute (or) field.

Ex: In
as a
Relatio
as
Key
alec
wh
is
al

Ex: In fig (1) sno, s.name, s.marks, cname) are known as attributes of student DB

Relationship set:

Association b/w multiple entity/ es known as Relationship set

The General Representation RSS is

$$E = \left[(e_1, e_2 \ldots e_n) / \exists\, e_1 \in E,\ e_2 \in E \ldots e_n \in E \right]$$

Here E - is known as entity/ set

$e_1, e_2 \ldots e_n$ represent entities.

Key vs non key/ attribute:

a key/ attribute is one which always contains Distinct / unique values.

A non key/ attribute is one which contains may or maynot Distinct value i.e a non key/ attribute allowes unique and Duplicate values.

In the fig(1) Sno. is called key attribute

(Sname, Smarks, cname) are known as non key attributes.

Functional Dependency:

It is one of the mathematical expression which express relationship b/w key & non key attributes

Ex: in fig (1)

F.D: Sno → sname, smarks, cname

Hence each & every key attribute is totally functionally determine non key attribute

Schema:

The overall view about the database is known as schema or Relation.

Ex: Define a Schema for student database

student (stno Integer, stnam: String, marks: Real, cname: String)

Integrity constraints in DBMS:

(correctness of data for)

constraints are nothing but rules and They are used for validating the data of the database.

(validation data)

when we design any database, it is mandatory to apply integrity constraints.

in DBMS we have 3 Types of integrity contraints. They are

1. Domain integrity constraints
2. Entity integrity constraints
3. Referential integrity constraints

Domain integrity constraints:

Domain represents column of the database. These constraints are used for validating maintaining correctness of the data for the column.

Domain Integrity constraints are of two types

a. NOT NULL
b. CHECK.

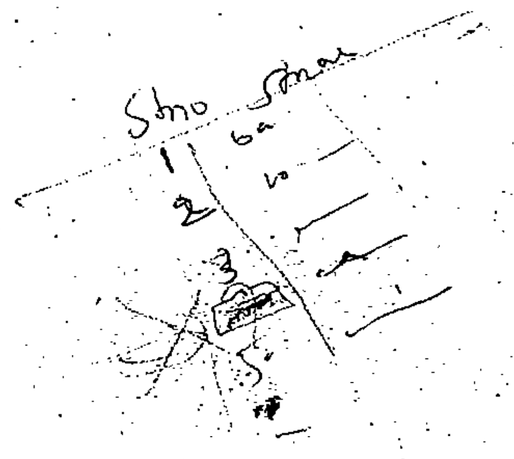
NOT - NULL:

when we create a database with various columns, those columns are by default allows null values which are not recommanded. In order to Enter a value compulsary for a Column Name we must make that column name as not NULL.

CHECK: Check constraint is used for validating the database columns with some pre-defined Conditions.

This contraints must be always applied on Numarical columns.

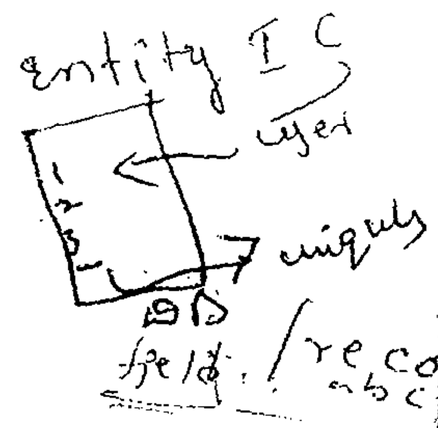
Domain Integrity constraints are also known as Column level Contraints.



Entity Integrity constraints.

Entity Integrity constraints are used for identifying the records of database uniquely.

These constraints are decided into 2 types. They are
1. unique
2. Primary key.



Unique: unique is a constraint which allowes, to maintained Distrinct values (in the case of numericals, characters and alpha Numaric), and it allowes one-time NULL As a value. In DBMS NULL causes various Problems.

Primary key:

Primary key = unique + NOT NULL;

In otherwords Primary key contraint maintaine Different unique values and no NULL values are allowed.

Sai Ram

Referential Integrity constraints: (foreign key)

Referential Integrity constraints are used for establishing the Relationship between multiple Databases. In otherwords

These constraints are used for ref-

ing The data from multiple database.

Consider The following database

Dept DB (or) Parent DB

deptno	dname	LOC
10	IT	HYD
20	MRKT	PCH
30	MCA	MRKP
40	MBA	HYD

no duplicate

Emp DB (or) Child DB

foreign Reference

Eno	Ename	Sal	deptno
100	a	1.2	10
101	b	1.5	20
102	c	0.5	10
103	d	0.6	20

duplicate

The above databases dept database is known as master/parent database where as emp database is known as Detailed/child Database.

using Referential Integrity constraint we can Establish the Relationship between master Detailed DBases.

Rules:

1. foreign key (or) Reference key (deptno of Emp database) Must always Exist in child/Detailed database. foreign key can contain duplicate values. [deptno of Dept DB]
2. Parent is being referenced as primary key i.e no duplicate values

are al

3. For

Show

Data

A. NO

— led

The

5. Ma

Recor

6. NO

Recor

Levels

from

Data

They

are allowed and it should not be NOTNULL.

3. For Each & Every record of master database, their should exist n-number of records in child database.

10 it ury 100.

4. No insertions can be performed in the detailed DB, Provided the Master DB does not contain the reference key value.

5. Master DB record can't be deleted if their exist records in detailed DB.

6. No updations can be performed on master DB records if their exist detailed database records.

Levels of Data Abstraction:

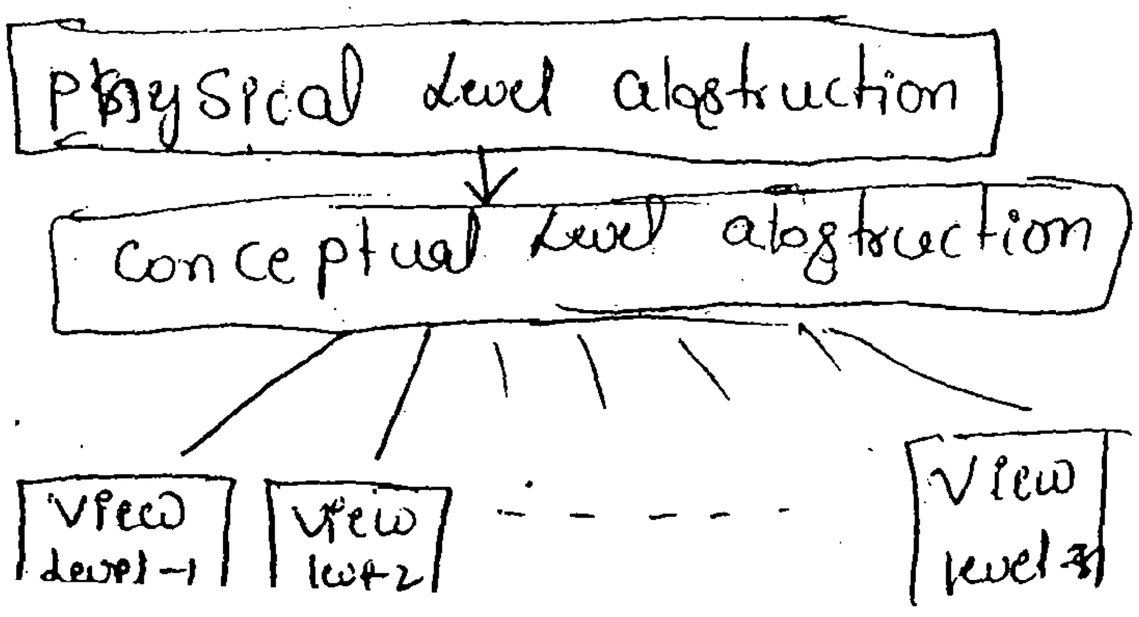
(getting the data)

Data abstraction means retrieving the data from the DB without dealing with background details.

- In DBMS we have three levels of abstraction

They are no 4. Physical level abstraction
2. Logical / conceptual level abstraction
3. View level abstraction.

eno = 100 dname = 'it'



sical level abstraction:

This level of abstraction deals with "How the data is internally organised".

means when we are designing the database, we must choose name of the database, choose the field names, apply various integrity constraints and specify the sizes of the fields.

Ex: Design a database for Employee

Emp (empno, ename, sal);

name of the database

Here empno → is the Primary key
ename → is the NOT NULL
SAL → is the NOT NULL

Emp is called name of the database.

ical/conceptual level abstraction:

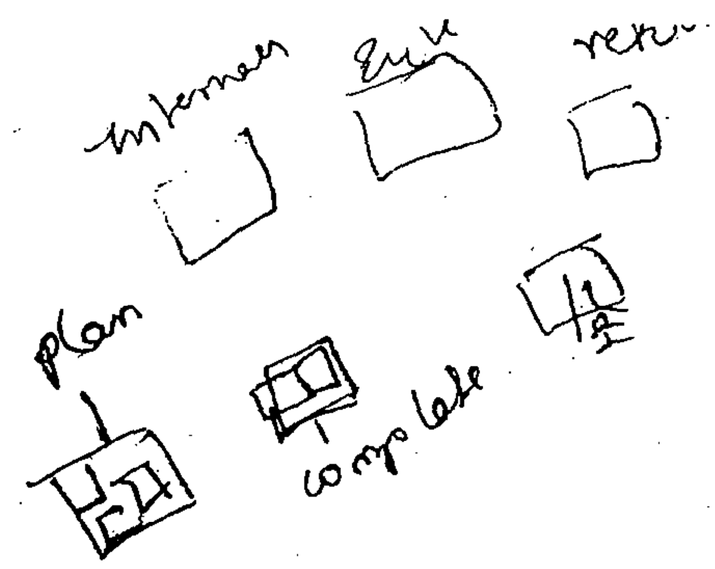
This level of abstraction deals with "How the data is entered" without dealing "How the data is organised".

In This level we entered the data the database depends on the fields of the database

Ex:

Emp

Empno	ename	Sal
10	abc	1.5
11	pqr	1.9
12	xxx	11.11



View level:

This level of abstraction always deals with "How the data is retrieved" without dealing with "How the data is entered" and "How the data is organized".

In otherwords view level always deals with retrieving the data from database in various possibilities by writting various application programme

NOTE: If a DB contains n-no.of columns then we can retrieve the data of the database in n! ways.

Ex: From the above Emp dBase we can retrieve (Empno), (ename), (Sal), (Empno, ename) (ename, Sal), (empno, Sal) & (Empno, ename, Sal).

Constraints (Cardinalities);

mapping constraints are nothing but rules and regulations to be used by the DB Programmer to be applied on the database.

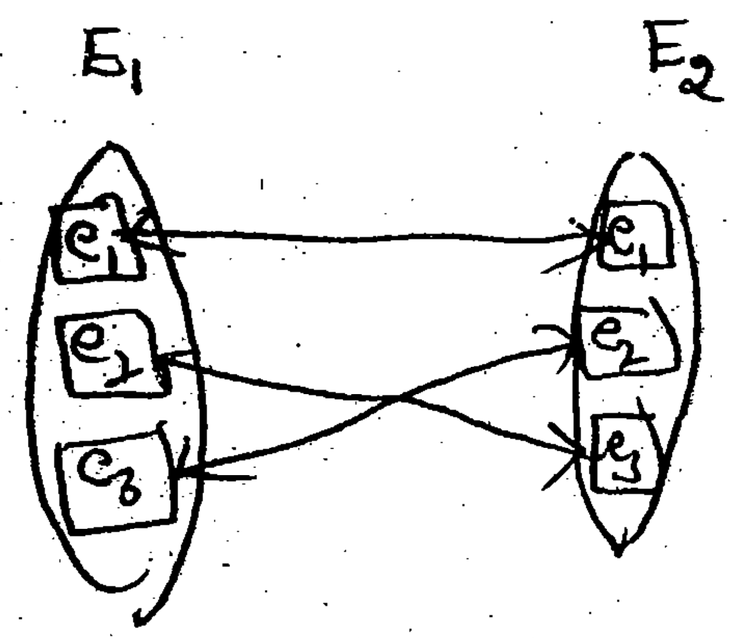
mapping constrains are always makes to understand how the record(s) of one database is associated with another record(s) of another DBase.

based on these rules mapping constraints are devided into 4 types. They are

1. one to one.
2. one to many.
3. many to one.
4. many to many.

One to one:

Each record of one database must be associated with unique record of another DB.



Ex: each student must have unique student number.

ii. Each Employee must work only in only one Department no.

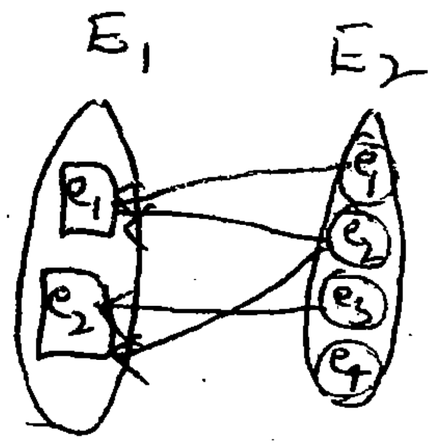
One to Many:

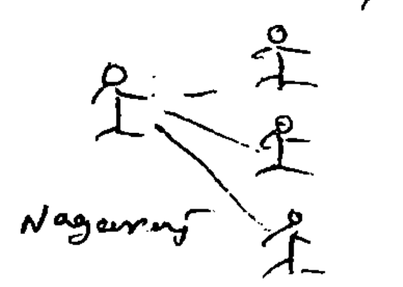
Each Record of one database must be associated with many Records of another Database

(OR)

Many Records of another Database are mapped with unique Record on one Database.

Ex:





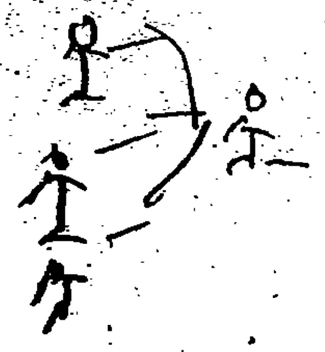
Ex: one college can contain many courses.

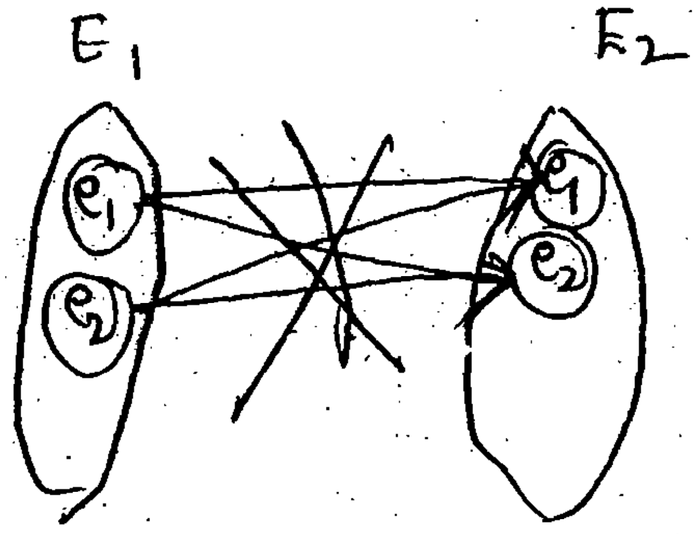
one customer can have many accounts.

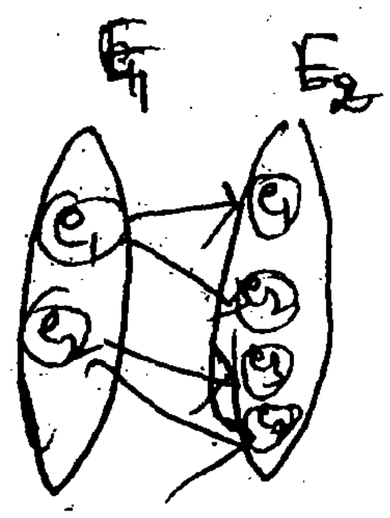
Many to one:

Many Records of one Database related with unique Record in another Database (or) one Record of one Database is associated with many Records of another Database.

Ex:





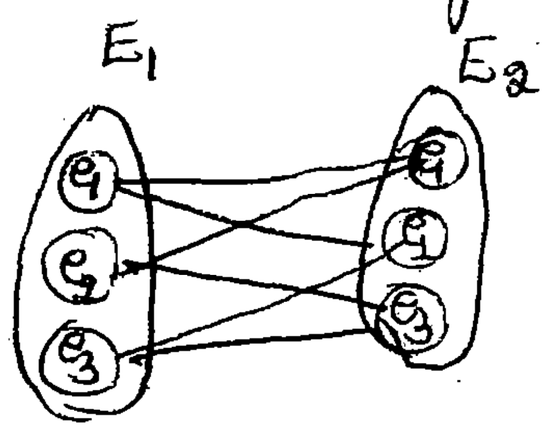


Many colleges containg one common Course.

Many ~~Accounts~~ Banks Can Containg One common Costemer.

Many to Many:

Each record of one database must be associated with many records of another Data-base (or) Each record of other Database must be associated with Many records of Other Database.



Ex: Many Colleges offers Many courses (or) Many courses offered by Many colleges.

DATA MODELS:

Data model is nothing but Simplified Representation of Reality.

In DBMS we have four types of General Data models. They are

1. Network model
2. Hierarchal model
3. Relational model
4. ER (Entity Relationship) model

Network model & Heirarchical models are the older Data models and the industry is not using Becoz of Their Complexities and hardness.

Relational model:

- In the Year 1970 the industry Expert 'Codd' was developed this relation model for Eliminating the difficulties and complexities of network model and heirarchical model.

① In Relational model the Data is Represented in the form of Rows and columns in the table. ② A table is a DB object in which the Data is organised in the form of rows and columns.

③ In this model name of the DB is nothing but Relational Schema. (which gives overall view about the Database.

④ In this model the columns of the table are known as attributes or fields, Each attribute must be Qualified with an appropriate domain. (type of the column known as domain).

⑤ In this model the Relational schema (Database) must be enforced by various integrity constraints for performing various operations of the Database Effectively.

(6) The total no. of records in a particular relation is known as relational instance.

Consider the following Employee Database.

Emp (empno: Integer, ename: string, sal: Real)

Emp

	Empno	ename	sal
1 —	100	abc	1.5
2 —	101	pqr	1.6
3 —	102	aaa	1.5
4 —	103	abc	1.5

(7) Here Emp is known as Relation name or DB name or table name.

(8) • Empno, Ename, sal → are fields.

• (100, abc, 1.5) ... (103, abc, 1.5) are known as Records Tuple/r

• The instance of Emp database is 4.

(9) Relational Model is simple and elegent, m representing the data in the form of tables for retrieving and performing various operations on the database.

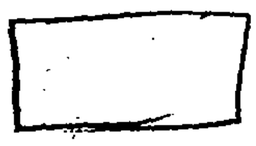
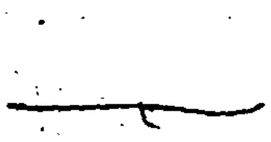
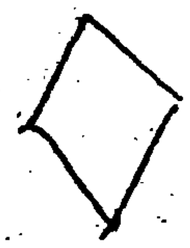
(10) • Relation model was popularised by IB IBM and it is used in most of the popular DB Products. (oracle, DB2, Sy informix, Pa

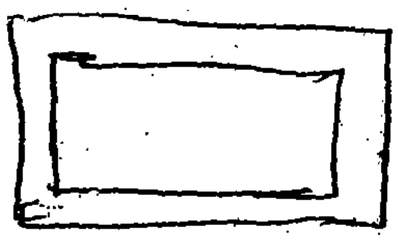
ER-model: (Entity relation Model)

At the time of designing the DB, we need to conduct requirement analesys for obtaining information from Organisation. To describe the information we got from the organisation can be describe diagramatically who a model called ER-model.

ER-model gives overal logical view about database design or deborial representation of database is kyocon as ER-model. [Between 2 entity sets their exist only one relationship set, Then wise it is invalid ER-diagram]

Symbols used in ER-model:

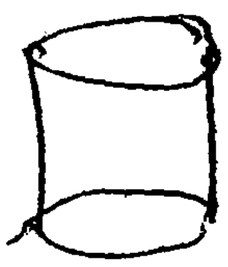
Symbol	Name	Purpos
	Rectangle	• used for representing Entity set/relation name/ DB Name.
	Elipse	• user for representing fields of the Entity set
	line	represent the bloco b/w Entity set to attribute and Entity set to relationship set.
	Rombus/ Diamond	• used for Representing the relationship b/w multiple DB



Double Rectangle — used for representing weak entity set. Ex: Con... mode...

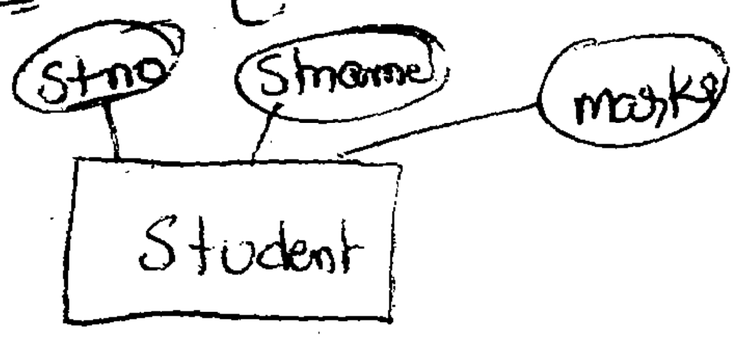
Double Ellipse — used for representing weak field.

→ Double Diamond / Rambus → used for representing weak relationship set.



→ Cylinder → used for representing Master DB. B → Con... Ship

Eg: construct an ER-model for student info... mode...



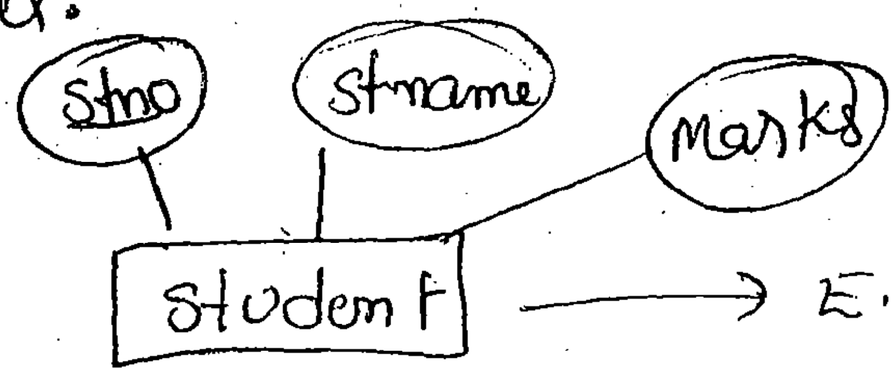
(A) Converting ER-diagramme into Relation... we c... model.

when simple ER-model is given we can convert... follo... The ER-model into relational model the... following the below steps

1. Take entity set name as a relation...
2. Take attribute names as fields...
3. Each & Every field in the relation model must be qualified with domain (data type)..

(a) ... -cipc...
(b) ... rela...

Ex: Convert the following ER-model into relation model.



→ E.R-model

student (stno: Integer, stname: string, marks: Real):

(B) Converting multiple ~~DB~~ along with relationship set into relational model.

(or)

Translating Relationship set into Relational model.

Sol: when ER-model is given with relationship, we must convert into relational model.

In order to convert the relationship set into relational model we must follow the following steps.

(a) Define Relational model for first participating Entity set

Ex: Dept(deptno: Integer, dname: String, loc: String)

(b) convert second participating Entity set into relational model

Ex: emp(empno: Integer, ename: string, sal: Real)

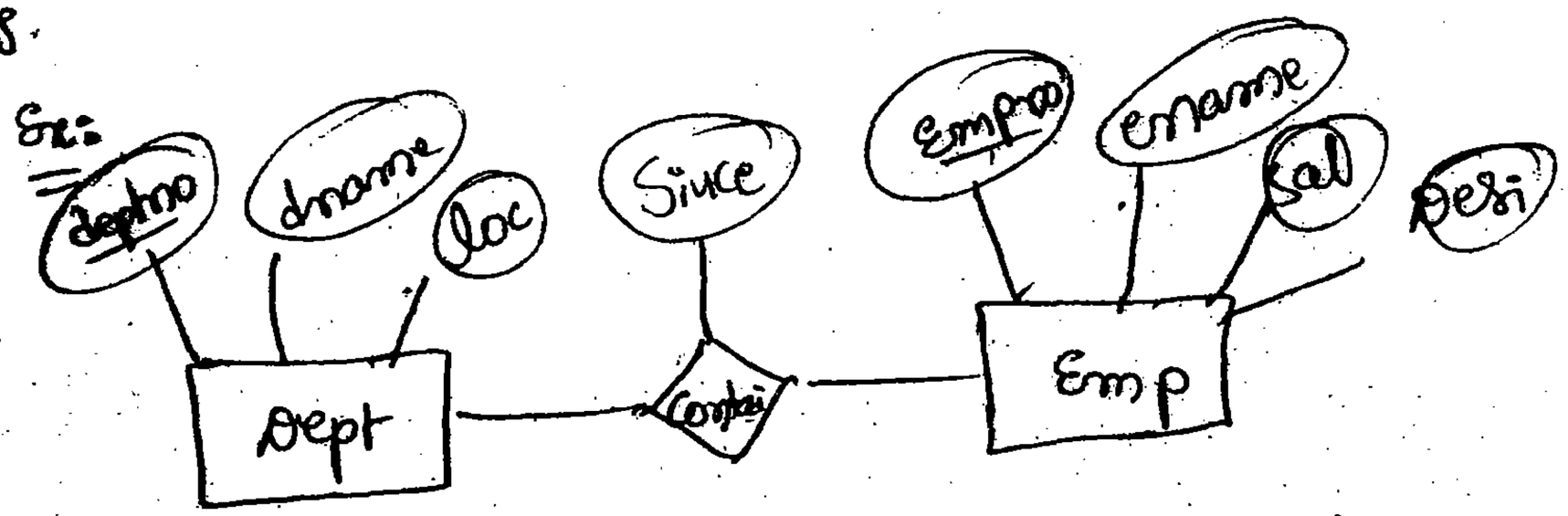
(c) The attributes for the Relationship set are Primary key of first Participating Entity set and Primary key of second participating Entity set.

Ex: Contains(deptno: Integer, Empno: Integer)

Converting Relationship set with Discriptive attribute into relational model.

Discriptive attribute is the Name of the field To be represented on Relationship set. Descriptive attributes always express more logical relationship b/w participating entity sets.

Ex:



To convert the above ER-model into Relational model we must follow the following steps

1. The Relational model for first participating Entity set contains all the attributes associated with it.

Ex: Dept (deptno: Integer, dname: string, loc: string)

2. The Relational model for Second Participating Entity set contains all the attributes associated with it.

Ex: Emp (Eno: Integer, Ename: string, sal: real, designation)

3. The Relational model for relationship set is primary key of first participating Entity set, primary key of second participating Entity set and descriptive attribute of relationship set.

Ex: contains (deptno: Integer, empno: Integer, Since: date)

Aggregation:

It is not possible to write multiple relationship sets between two Entity sets. In order to construct an ER-diagramme with multiple relationship set along with two or more Entity sets, we must use the concept of aggregation. Aggregation always makes to understand write the Relationship set b/w

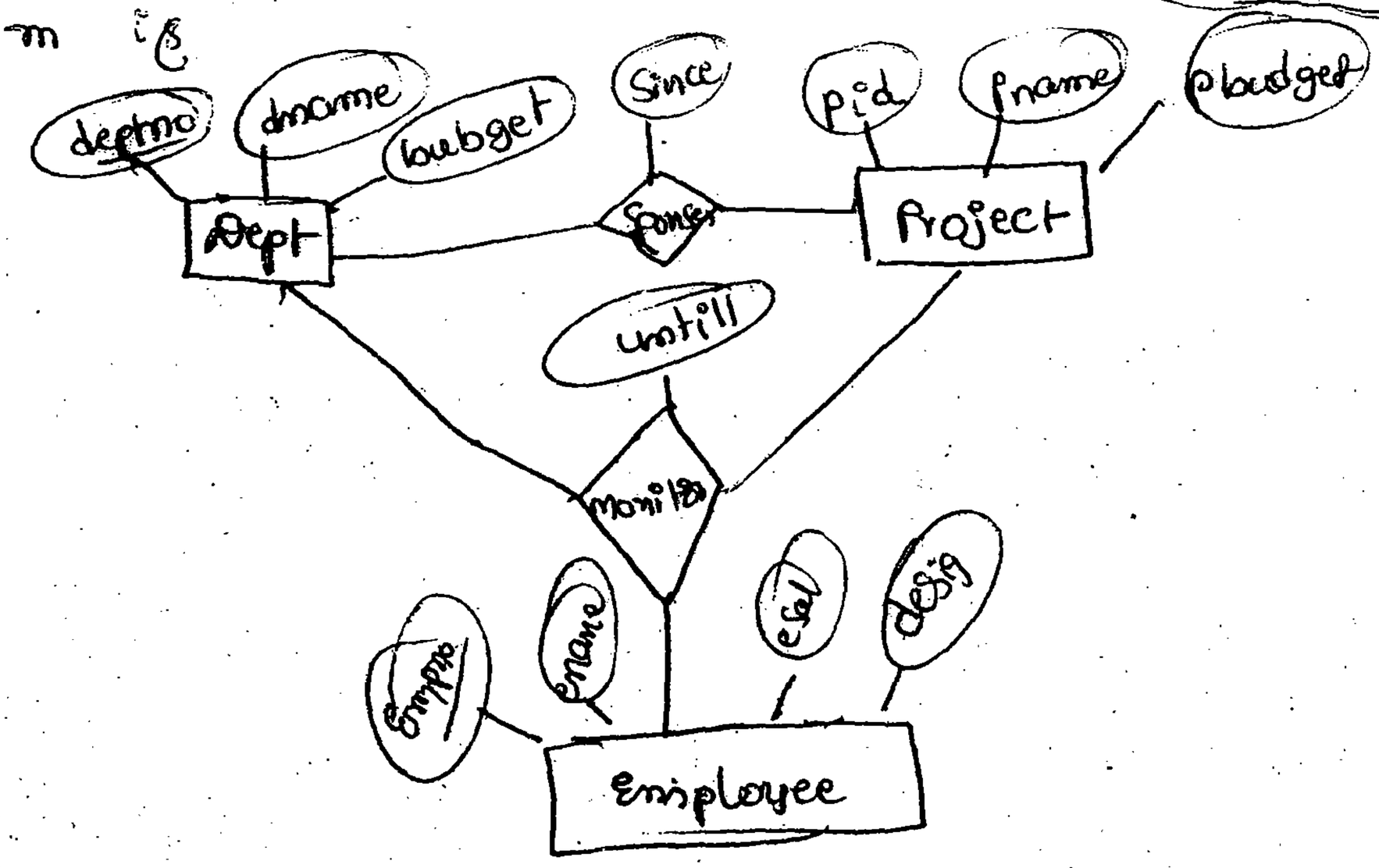
Relationship set with entity set

(OR)

writting relationship set b/w one ER-diagram one and one entity set.

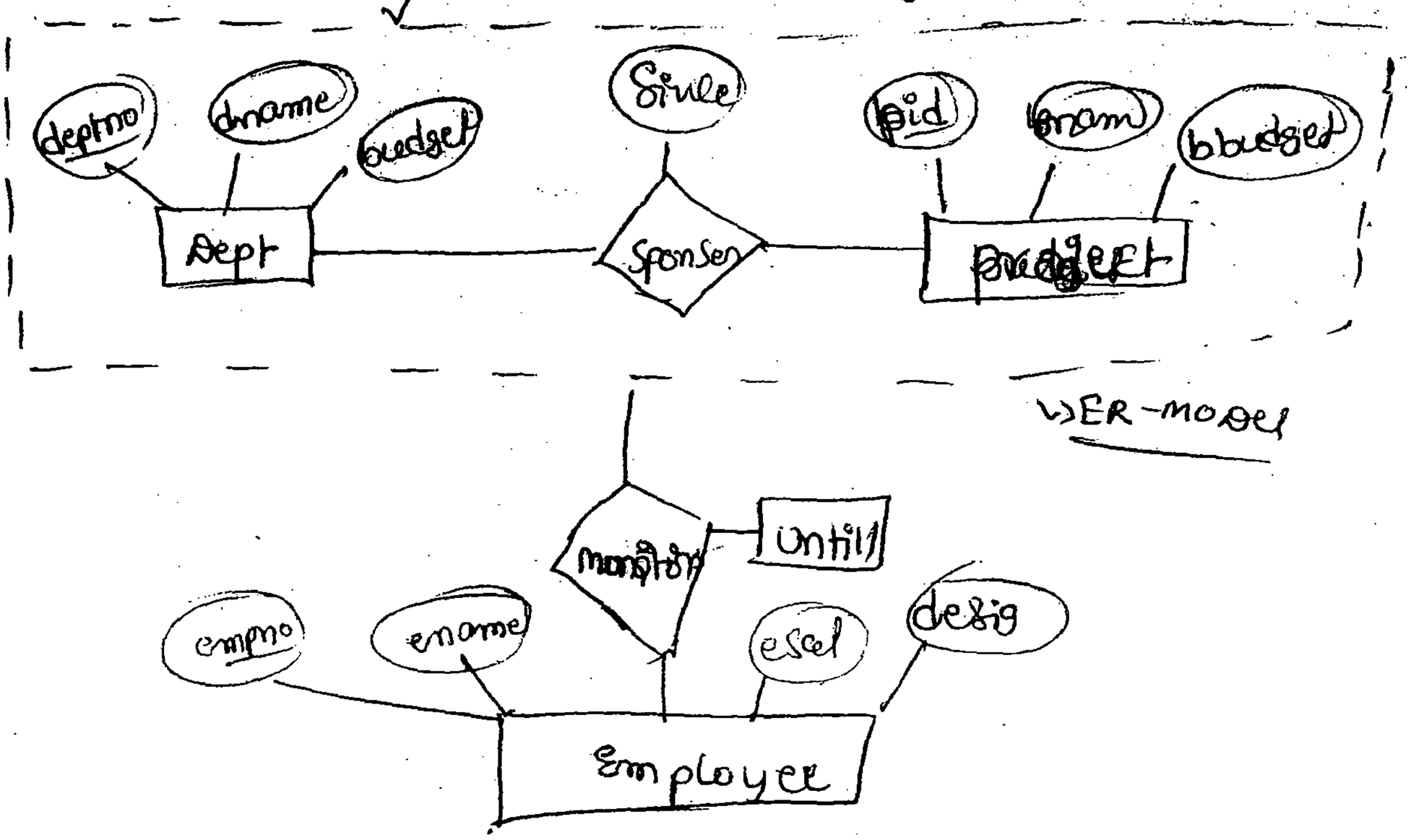
Ex: Their exist various departments, each deptment is sponsoring various projects. to develop the projects departments are recruiting various employees for monitoring and developing for a period of time. Each every employee must be associated with their projects and their departments.

→ For the above problem the proposed ER-diagram is



In the above diagramme, b/w two entity sets, their is more than one relationship set. which is invalid according to ER-diagrammes Rules.

- to solve the above, we must apply the concept of aggregation. Hence the exact ER-diagramme for the ~~above~~ proposed ER diagramme is



Sol: The Relational model for the above ER-diagram is given below.

Dept (deptno: Integer, dname: string, budget: Real)

Project (pno: Integer, pname: string, pbudget: Real)

Sponser (deptno: Integer, pno: Integer, Since: Date)

Employee (Empno: Integer, ename: string, esal: Real, desig: Str.)

monitor (deptno: Inter, pno: Integer, Eno: Integer, Since: Date, until: Date)

Generalization (or) Specialization:

- The Process of Developing two or more Specibised classes from the General class is known as Generalization or Specialization.

- The representation of Generalization is known as Class Heirarchy.

- Generalization can be represented by a symbol known as ▽ (is-A).

Ex:- Consider there is a organisation where two types of Employees are working namely hourly Employees and contact Employees.

In General every Employee contains social Security number (SSN), Employee name, and Designation. Then Employees receive the salary from organisation the basis of number of hours they work. Where as contact Employees receives fixed Amount irrespective of no. of hours they work.

Sol: From the about Description the General class Employee with SSN, Name, Designation.

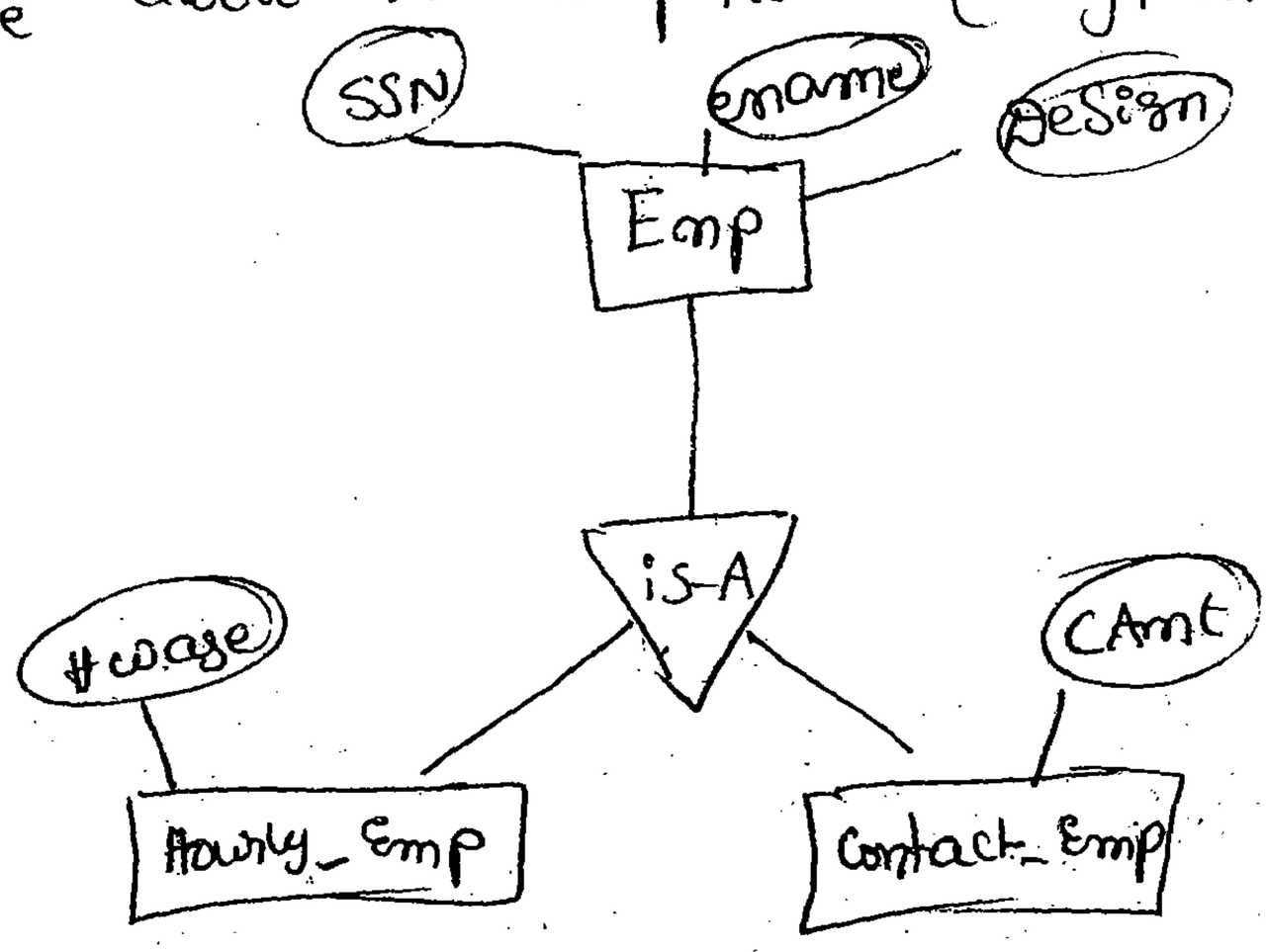
The two Specialised classes are Hourly Employee with SSN, ename, Designation and hourly wage.

Contact_Emp with SSN, ename, Designation, contract amount.

In the above Entity sets SSN, ename, Designation are common for all three Entity sets Hence they must be written for Generalised class.

The attribute Hourly wage is specially related to Hourly_Emp Entity set and

The attribute Amount is specially realated to Contact_Emp. The Exact ER-model for the above Description is given below.



The interpretation of the above ER-diagram is

An Employee is-a hourly Employee.

An Employee is-a Contact Employee.

Hence in the above diagramme an entity set Employee is known as Generalised class and the entity sets hourly_Emp and contact_Emp are

known as specified entity sets.

The relational model for the above Generalisation is

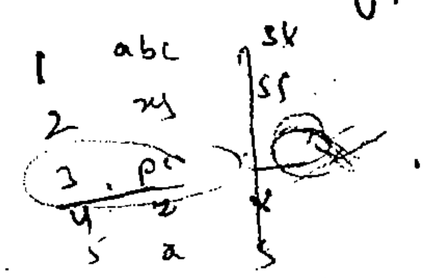
~~Employee~~

Emp(SSN: Integer, Ename: string, Design : string);

hourly_Emp(SSN: Integer, Hwage : Real) ;

Contact_Emp(SSN: Integer, cAmt : Real);

~~Normalization:~~



Pitfalls in DB Design:

Pitfalls are nothing but Problems. where we are Designing any DB for any organisation we can across two types of Pitfalls.

1. Data Redundancy ✓
2. unable to represent certain information even we know some other information.

Ex: Let us Consider the following DB Design

Deposit branch (acno, ename, bal, bname, asset, city)

(Designed by P_1)

Deposit (acno, ename, bal bname) } Designd by P_2
branch (bname, asset, city) }

In the DB Design which is Designed by P_1 there is a problem called the Data Redundancy

Since if their exist 100 customers of the same branch, we need to write 100 different customers information. But we are writing same branch information repeatedly 100 times. which leads to inconsistence result and high memory cost.

Hence irrespective of number of customer the branch information is repeated. Hence data redandancy is occured.

∴ The DB designed by P_1 is not effective.

In the DB design which is designed by P_2, we can represent n-no. of customer information in deposite DB and their branch details is represented in branch database only once irrespective of no. of customers.

Hence data redandancy is minimize by in the database design which is designed by P_2. and it is effective.

(2) In database design which is designed by we know the branch details but no customer available. Hence we are able to represent branch details but unable to represent details. Hence this is not effective design.

∴ In this database design we are unable to represent certain information even we know some other information. which leads to more updation problems.

In the DB design which is designed by P_2. is only branch details are there without customer details then represent the branch details, in branch database and we need not to represent any details about customer in deposit database.

Hence in the database design which is designed by P_2. is not having a representing and unknown information problem.

In the above DB design, P_1 - database design is not an effective. Since it contains two pitfalls of database design where as the database design which is designed by P_2 - is completely elemenated pitfalls of database designed by using normalization concept.

Normalization:

Normalization is used for eliminating pitfalls (problems) of database design. In the database design, we use normalization for obtaining well structured relation's / schema's.

once we have well-structured relations we can perform the database operations like insertions, deletion and updation effective

A Database May contain two types of attributes They are Key and nonkey attributes.

Key attribute is one which always contains District/unique values. Key attributes are always used to identified The records uniquely and Each & Every key attribute must Belongs to Primary key.

A non key attribute is one which always may contain Either unique or duplicate or more.

A nonkey attribute must always belongs to NOT NULL Constraints. The relationship b/w key & non key attribute Can be Expressed by functional Dependency.

functional Dependency states that a non key attribute is fully functionally determining list of key attributes. In Other words, The list of non key attribute are fully functionally Determind by key attribute.

In order to get well structured relations we have the following normal forms

(a) 1NF (b) 2NF (c) 3NF (d) BCNF

(e) 4NF (f) 5NF

2NF :

Def "Elemenate Partial Dependency"

(OR)

"No non-key attributes depends on part of the Primary key"

Primary key = unique + not null

Def of Partial Dependency:

If any non key attribute depends on part of the Primary key Then it is known as Partial Dependency.)

Ex: Empno → sal, ename, Desig
PK — Non Key all

Ex: Give the Relation which is in 1 Normal form but not in 2NF.

Primary not

Ans: fig(2)

in fig(2) in order to determine Date_Completed effectively we must make Empno, and Course as a key attribute.

Empno and course is known as composit key attribute.

The attribute Date_completed is fully functionally Determind by Empno and Course.

Emp
10
10
10

FD

but

also

as

Part

fig

for

co

by

FD : (Empno, course) → Date_complet

but a part of the Primary key Empno is also determining the attributes Ename, sal, Design.

Hence this type of Determination is known as partial ~~Determ~~ Dependency.

According to 2NF we must elemenate partial dependency from fig(2).

To Elemenate Partial Dependency from fig(2), we must Decompose fig(2) into the following relations.

Emp_details (Empno, Ename, Sal, Design)

course_details (Empno, course, date_complet)

The Sample Database for Emp_details and course_details are given below.

Emp_details

Empno	Ename	Sal	Design
100	abc	1.5	PL
101	pqr	1.0	SE
102	xyz	1.5	DE

course_details

Empno	course	date_comple
100	C++	30/09/0
100	Java	29/09/0
101	Java	30/09/0
101	C++	29/09/9

fig(3)

Irrespective of Number of courses learned by an Employee, we are representing Employee details only once in Emp_details DB.

Hence Data Redundancy is minimized.

If any Employee is not learning any Course, the Course details are not maintained in Course_details Database Hence we are able to represent complete information about Courses which are learned by Employees.

∴ If any relation satisfies 2NF then all pitfalls of database Design are completely eleminated and we get well structured Relations.

1 NORmal form (1NF):

"Each & Every field must be atomic" [an atomic field is one which must Contain a value] (OR)

"intersection of rows & columns must Contain a value" (OR)

→ "Elemenate multivalue attributes".

(multivalue attribute is one which contains multiple value)

Relation

Ex:

Consider the following Relation

Emp_course(Empno; ename, sal, Design, course, date comp)

The Sample Database for the above relation is

Emp_course - RD/R

Empno	Ename	Sal	Design	course	date comp
100	abc	1.5	P2	C++	30/09/09
~~100~~	abc	1.5	P2	Java	25/09/09
101	pqr	1.0	SE	Java	30/09/09
~~101~~	pqr	1.0	SE	C++	25/09/09

fig(1)

In fig (1) The Employee abc is learning two courses where as PQR is also learning two courses. The fieldnames Ename, Sal, Design, Course, Date_Comp are containing multiple values for the Empno field.

According to 1NF eliminating multivalue attributes. Multivalued attributes can be elemenated in fig (1) by writting Empno for the fields Ename, Sal, Design, Course, Date complete separately.

fig (1) is converted into first Normal form and it is shown below.

Emp_course

Empno	Ename	Sal	Design	Course	Date_comp
100	abc	1.5	P2	C++	30/09/09
100	abc	1.5	P2	Java	25/09/09
101	PQR	1.0	SE	C++	30/09/09
101	PQR	1.0	SE	Java	25/09/09

fig (2).

3rd Normal form:

"Every determinent is a Key" (or)

"Elemenating Transitive Dependency"

A Transtive Dependency is one in which There Exist three attributes x, y, z. and x determines (x→y), y determines z (y→z) and finally x determines z).

Ex:- Give the Relation which is in 1st NF, 2NF But not in 3NF

worker(wid, skill, Bonus_rate)

The Sample database for worker_relation is given below

wid (x)	Skill (y)	B_rate (z)
100	Electrician	5.00
101	gardener	6.50
102	plumber	7.00

With the above Database the key attribute is fully functionally Determining skill

$x \rightarrow y \quad y \rightarrow z$

$x \rightarrow z$

It is able to determind other nonkey attributes. Hence skill attribute is known determinent.

fD : wid → skill;

TD: skill → B_rate; ⇒ wid ⇒ B_rate.

According to 3rd NF Transitive dependency must be Eliminated by making skill attribute as key attribute

After applying 3rd NF on worker relation, it will be Decomposed into the following relations.

worker_details (wid, skill);

worker_skill (skill, B_rate);

The Sample database for the above Relation is given below.

work_details

wid	skill
100	Electrician
101	Gardian
104	plumber

fD: wid → skill

skill	B_rate
Electrician	5:00
Gardian	6:00
plumber	7:00

TD: skill → B_rate.

BCN

Greve

not

Dom

skill

Ft

-01

BC

If the relation is NOT in 3NF we get 3 - dis - advantage / anamalies.

They are

1. Insertion Anamolees.
2. Deletion "
3. updation "

BCNF: (Bi code Normal form)

"Every determinent is a key"

(or)

Eliminating Transitive Dependency"

Give the Relation which is in 3rd NF but not in BCNF.

worker_details (wid, skill)

~~worker~~ skill_details (Skill, Exp, Bonus_rate)

In General by observing the above Relation Bonus rate must be Determind by Experiance and skill. Here Experiance is an non key attribute. Hence it must be made as key attribute.

after applying BCNF (skill, Exp) made as Comp-osit key in skill_details Database.

The Difference b/w 3rd NF and BCNF is a single Primary key is always

available in 3NF and composite primary key is available in BCNF.

The Sample database for worker_details, and SKILL_details is given below.

work_details

WID	SKILL
100	Electrician
101	Electrician
104	Gardiner

FD: WID → SKILL

SKILL_details

SKILL	Exp	B_rate
Electrician	10	5:00
Electrician	5	2:50
Gardiner	5	3:00
Gardiner	10	6:00

FD: (SKILL, Emp) → B_rate

4th NORMal form:

"Elemenating multivalued Dependency"

Def Multivalued Dependency:

Multivalued Dependency is one which will exist with minimum three attributes say (a, b, c). For each value of a their exist well defined set of values of b and for each value of a their exist well defined set of values of c. However set of values of b must be independent from set of values of c. or set of values of c are total independent from set of values of b.

Ex: Give a Relation which is not in 4th Normal form.

course-details (coursename, Books, faculty)

NOT in 4NF

In the above Relation, for each & every coursename, their Exist well Defined set of books and for each & every coursename, their Exist well Defined set of faculty.

However set of books must be independent from set of faculty (or) set of values of faculty is independent from set of values for books. Hence the Relation course-details contains multivalued Dependency.

According to 4NF, multivalued Dependency must be Eliminated.

After Eliminating multivalued Dependency from course_details relation, it ~~contai~~ Decomposes in the following Relations.

course_books (coursename, Books);
course_faculty (Course Name, faculty);

are in 4NF.

5 Normal Form:

"Eliminating lossy-less join De-composition".

In 5NF we are composing various Relations into single Relation for obtaining consistant Result.

Ex: Give the Relation which are Not in 5th NF.

R_1 (acno, bname, bal)

R_2 (bal, bname)

The Sample Database for R_1 and R_2 Relations are

R_1

acno	cname	bal
1000	a	100
1001	b	200
1002	c	100

bal	bname
100	SBH
200	SBI
100	ICICI

On the above two Databases if we apply the following Query there is a possibility of getting in consistent Result.

Q) find branch name of that customer whose acno is 1000.

Ans: SBH
ICICI

which is an inconsistant result.

According to SNF we must join the seperated relations into single relation for obtaining consistant result.

Deposit (acno, cname, bal, bname)

The above relation is obtained by ~~Deco~~ composing R_1 and R_2 relation.

The sample database for Deposite is give below.

Deposit

acno	cname	bal	bnam
1000	a	100	SBH
1001	b	200	SBJ
1002	c	100	ICICI

Q) find the branch name of that customer whose acno is 1000 Ans: SBH

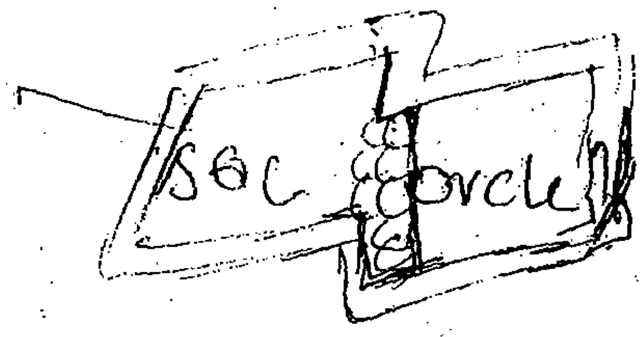
which is consistant result.

1000 a 100 SBH

Sad

Unit - II

1) Relational Algebra
2) Relation calculas



Query: Retriving the data from database. State

1) Relation algebra
2) Type of relations calculas
3)
4)
5)

SQL: structured Query language

Priliminaries of a Query langage:

when a Query is given to answer that Query we must follow the following basic Priliminaries.

1. Find which database is to be refered
2. find (or) identified the type of condition tos to be refered.
3. find which coloumns / attributes / fields values ~~attrib~~ to be retrived.

(9) To Retrieve the Data from multiple Data Bases should Exist Some Common Coloumn.

4) wher
we
but,
5) m
Rela
6) Π
Re
7) Ea
an
the
8) a
multi
Comc
must
Comc
Rela
=
Jam
Rel
by
an

4) when we are retriving data from the DB we must specify coloumn names of the DB, but not positions of the coloumns of the DB.

5) In Query Evaluation, instances (Records) of a Relation are Participating as an input.

6) The o/p of a Query is also instances of a Relation.

7) Each & Every Relation may be Qualified with an alternative name. The alternative name of the Relation is Known as Relation.

8) when we are Retriving the Data from multiple Databases and multiple Databases contain Comon coloumns Then the common coloumn must be referred by Relationname/alias name. Common Coloumn name.

Relational ALGEBRA:

1. Relational algebra is one of the Query language developed retriving Data from any Relational model.

2. In Relational algebra Queries can be Developed by using Some set of operators. Those operators are Devided into unary and Binary operators

Unary operators are those which can be used to retrieve the data from single database and such type of queries are known as simple queries.

When we use binary operators for retrieving data from multiple databases by writing various queries and such type of queries are known as complex queries.

Relation algebra is a kind of mathematical representation and it is known as procedural language.

Operators in Relational algebra:

(1) Select (σ)
(2) Project (π)
(3) Intersect ($\cap$)
(4) union ($\cup$)
(5) set Difference ($-$)
(6) Rename (ρ)
(7) assignment ($\leftarrow$)
(8) Cartesian product ($\times$)
(9) Natural Join ($\bowtie$)

Sailors

Consider the following databases.

I → Emp (empno, ename, sal, design, deptno)
→ Dept (deptno, dname, location)

II → Sailors (sid, sname, rating)
→ Boat (Bid, Bname, color)
→ Reserves (sid, Bid, date)

III → Customer (cname, street, city)
→ Deposit (acno, cname, bal, bname)
→ Borrow (loanno, cname, Amt, bname)
→ Branch (bname, asset, city)

SELECT (σ):

This operator is used for representing list of conditions. If the condition is true then we get the details about complete record. Comparision starts from the begging of the Records of the Database.

If the condition is false we never get any record details.

Syntax:

$\sigma_{(condition\ list)}$ (relation name)

Relation name represents name of the Database
condition list represent Either one or more.

Q) find the details of an Employee who's Empno is 100

$\sigma_{(Empno=100)}(Emp)$

Q) when the above Query is Executed we get complete Details about Empno, Ename, Sal, Desig, Deptno.

(Q2) find the details of the Sailors who's rating range between 6 and 8.

$\sigma_{((rating \geq 6) \text{ and } (rating \leq 8))}(Sailor)$

Sid	Sname	rating
10	x	10
20	y	7
30	z	8
40	a	6

→ Compound State

PROJECT (π):

This operator is used for selecting two column values of the Database when the condition is satisfied.

The Projecter operator will select those values which are unique i.e this operator displays only unique values.

Syntax: $\pi_{(attribute\ list)}(relation\ name)$

Attribute list represents list of coloumn values which we want to retrive.

(Q3) find the Name of the customers who are living in Hyderabad.

$$\Pi_{cname}\left(\sigma_{(city='Hyderabad')}(customer)\right)$$

Q4) Find loan Number, customer name and Branch name who's loan amount > 5.0.

$$\Pi_{lnone, cnas, bnae}\left(\sigma_{(Amt>5.0)}(Borrow)\right)$$

(Q5) Find Boat id and its colour who's name is interlake

$$\Pi_{Bid, color}\left(\sigma_{(Bname='Interlake')}(Boat)\right)$$

$R_1(1,2)$ $R_2(3,4)$

$R_1 \times R_2 = ((1,3)(1,4)(2,3)(2,4))$

Cartessian Product (X):

cartiesian product is used for retriving the data from multiple databases.

If we apply (X) b/w two Relations R_1 and R_2 and R_1 - contains m - coloumns and p records. and Relation R_2 - contains n - coloumns and q records Then $R_1 \times R_2$ contains m+n - coloumns and p*q records.

Ex: when we apply Cartician Product b/w Emp and Dept i.e dept X Emp then we get Seven Coloumn

Namely (Dept. Deptno, dname, loc, empno, ename, sal, Emp. deptno)

Q> Syntax:

$R_1 X R_2$ where R_1, R_2 are well structured relations.

> find Name of the Dept where Empno 1 is working

(Hint: Two Answer any Query which is retrieving the data for multiple databases we get the following conditions.

1. Direct Condition (given in Query) → (empno = 1)

2. Indirect Condition [Comman coloumn Equality]

$\boxed{\text{Dept. Deptno} = \text{Emp. deptno}}$

$$\Pi_{\text{Dname}} \left(\sigma_{(\text{Empno}=1) \text{ and } (\text{dept. deptno} = \text{Emp. dept}} \; \text{Dept} \times \text{Emp} \right)$$

* find Name of The Sailors who reserves boat 103.

Res (Sid, Bid, date)

$$\pi_{Sname}\left(\sigma_{(Bid = 103) \text{ and } (Reserves.Sid = Sailors.Sid)}\left(Sailors \times Reserves\right)\right)$$

* find The Name of The Sailors who's reservers Red boat

$$\pi_{Sname}\left(\sigma_{(colour = Red) \text{ and } (boat.bid = Reserve.bid) \text{ and } (Reserves.Sid = Sailors.Sid)}\left(Boats \times Reserves \times Sailors\right)\right)$$

* find The colours of the boats Reserved by andy. (Sname)

$$\pi_{bcolour}\left(\sigma_{(Reserved = 'andy') \text{ and } (S.Sid = R.Sid) \text{ and } (R.bid = B.Bid)}\left(Sailors \times boat \times Reserved\right)\right)$$

* find colour of the boat and Sailor name who's Reserves a boat on 1-9-2009.

$$\pi_{bcolour, Sname}\left(\sigma_{Date = (1\text{-}9\text{-}2009) \text{ and } (Reserves.Bid = Boat.Bid) \text{ and } (Reserves.Sid = Sailor.S}\left(boat \times sailors \times Reserves\right)\right)$$

Natural JOIN: (⋈)

Natural Join is also used for retriving the data from multiple databases.

Mathematically natural Join = (cartesian Product + implicit ~~Coloumn~~ common coloumn Equality)

Syntax: $R_1 \bowtie R_2$

where R_1, R_2 are two relations which containg m and n coloumns respectively.

The total Number of coloumns in $R_1 \bowtie R_2$ =

(No. of coloum in R_1 + No. of coloum in R_2) − Total Number of common coloumns in R_1 and R_2.

Sai

Ex: Consider The following databases.

Deptno

Deptno	dname	loc
10	IT	Hyd
20	HR	loc

Emp

Empno	Ename	Sal	deptno
100	abc	1.5	10
101	pqr	1.2	20
102	aaa	1.3	10

Deptno ⋈ Emp

Deptno	dname	loc	Empno	Ename	Sal	deptno
10	IT	Hyd 100	abc	1.5	1.5	
10	IT	101	pqr	1.2		
20						

* find Name of the department where Employee 100 is working

$$\Pi_{dname}\left(\sigma_{(empno=100)}(dept \bowtie emp)\right)$$

* find the Name of the department which is paying more than 1.2 lacs

$$\Pi_{dname}\left(\sigma_{(sal>1.2)}(dept \bowtie emp)\right)$$

$$\Pi_{dnm}\left(\sigma_{(sal>1.2) \wedge (dept.deptno = emp.deptno)}(dept \times emp)\right)$$

* find the customer names and their Branch who's loan amount greater than 1.5 lacs and also print asset of branch

(x)

$$\Pi_{cname,\ branch,\ bname,\ asset}\left(\sigma_{(amt>1.5)\ and\ (borrow.name = branch.bnam)}(borrow \times branch)\right)$$

→ NJ (⋈)

$$\Pi_{asset,\ cname,\ Bran.bname}\left(\sigma_{(amt>1.5)}(borrow \bowtie Branch)\right)$$

Q) Consider the following Relations.

Suppliers (Sid, Sname, address)
Parts (pid, pname, colour) → I.
Catalog (Sid, pid, cost)

Q1) find the Name of the suppliers who's supplies red parts.

Q2) find Supplier Name and Their address who's Supplies NUTS.

(Q3) Find Name of the Suppliers who'are not supplies any parts.

(Q4) Find details of the Suppliers who's supplies Either NUT or Bolt or Both.

(Q5) find details of the Suppliers who's supplies both Red and Green parts.

Set operators:

INTERSECT (∩):

Set operators are basically used for combining two are More Query results.

Syntax: r_1 Set operator r_2

r_1, and r_2 are the two relations which containg Temporary results of the Queries.

Set Operator represents Either union or intersection or set Difference.

Rules:

1) Number of coloumns of first Query must be Equal to Number of coloumns of Second Query (r_2).

2) The domain of all the columns of first Query (r_1) must be similar to Domain of all the columns of second Query (r_2).

INTERSECT (∩):

Intersect operator is used for obtaining the common Records which are available in multiple databases.

Syntax: $R_1 \cap R_2$

Here intersect (∩) operator used for retrieving common records of r_1 and r_2.

ex:

Emp

Emp	ename	Sal	deptno
100	abc	1	10
101	abc		10
~~102~~	pqr		

Dept

deptno	dname	loc
10	IT	Hyd
20	HR	Sec

Emp

Sol

Q) find the Name of the Employee who's working in both IT and HR Department.

Ans: find the Name of the Employee who is working in 'IT' department

r_1 — Π_{ename} ((Dept × Emp) σ (dept.name = 'IT') and (Dept.Deptno = Emp.Deptno))

find the Name of the Employee who is working in 'HR' Department.

r_2 $\Pi_{ename}\left(\sigma_{(dname='HR') \text{ and } (Dept.deptno = Emp.Deptno)} (Dept \times Emp)\right)$

5 a

$r_1 \cap r_2$ — give the Name of the Employees who are working in both IT and HR Department.

(OR)

$\Pi_{ename}\left[\left(\sigma_{(dname='IT') \text{ and } (dept.deptno = Emp.deptno)} (Emp \times Dept)\right) \cap \left(\sigma_{(dname='HR') \text{ and } (dept.deptno = Emp.deptno)} (Emp \times Dept)\right)\right]$

=

Sailors (

r find the name of the Sailors who's reserves both red and green boats on the same day

i) find Name of the Sailors who's reserves in read boat

$\Pi_{sname}\left(\sigma_{(color='Red') \text{ and } (Boat.bid = Reserves.bid) \text{ and } (Reserves.sid = Sailors.sid)} (Boat \times Reserves \times Sailors)\right)$

r_1

r_2

$r_1 \cap$

Π cnc

r_1

* fin

Acou

Shar

r_2

$$\Pi_{sname}\left(\sigma_{colour='green' \text{ and } (boat.bid = Reserves.bid) \text{ and } (Reserves.Sid = Sailors.Sid)}\left(Boat \times Reserves \times Sailors\right)\right)$$

$r_1 \cap r_2$ gives the name of the Sailors who ~~are Contain~~ Reserves both red and green colors of boats.

* find the Name of the costumer who is having Account and taken loan from SBH.

r_1

$$\Pi_{cname}\left(\sigma_{(bname = 'SBH') \text{ ~~and (Deposit)~~}}\left(deposit \times borrow\right)\right)$$

r_2

$$\Pi_{cname}\left(\sigma_{(bname = 'SBH') \text{ ~~and (Loan)~~}}\left(deposit \times borrow\right)\right)$$

$r_1 \cap r_2$ gives name of the costumer who is having Account and taken a loan from SBH both Dc

UNION [∪]:

This operator is used for obtaining the total records from multiple queries.

Syntax: $r_1 \cup r_2$

Here r_1 and r_2 → are two Relations which containg temporary reserves of Queries.

The operator union combines Either the result of r_1 or r_2 or both.

Ex: find Name of the Employees who works in Either it or in marketting?

$$\Pi_{ename}\left(\sigma_{(dname='it') \text{ and } dept.deptno = emp.deptno}(Dept \times Emp)\right)$$

r_1

$$\Pi_{ename}\left(\sigma_{(dname='HR') \text{ and } (dept.deptno = emp.deptno)}(Dept \times Emp)\right)$$

r_2

$r_1 \cup r_2$ gives Name of those Employees who works in both Either in it or in HR or in both.

* find the Name of the branches which are available in Hyd, banglore who's assets more than 5.4lac

$$\Pi_{bname}\left(\sigma_{(city = 'HYD') \text{ and } (asset > 5.4)}(Branch)\right)$$

r_1

$$\Pi_{bname}\left(\sigma_{(city = 'bang') \text{ and } (asset > 5.4)}(branch)\right)$$

r_2

$r_1 \cup r_2$ gives those name of the branches which are located either in HYD or in bangalore or in both.

Set Difference : (−)

Set Difference is used for removing the common records from multiple relations and it displays remaining records from first relation.

Syntax: $r_1 - r_2$

Here r_1 and r_2 are the two relations which contains the result of two Queries.

The operator − is known as set Difference operator and it is used for eliminating common records from both r_1 and r_2 relations and Displays the remaining

vector as from r_1 Relation.

* find the Deptname which does not contain any employee.

a). Get the dname which are available in Dept database.

$$r_1 \leftarrow \Pi_{dname}(Dept)$$

b). get the name of the Dept which are available in Employee Database

$$r_2 \leftarrow \Pi_{dname}\left(\sigma_{Dept.Deptno = Emp.Deptno}(Dept \times Emp)\right)$$

iii) $r_1 - r_2$ gives dname which are not containing any Employee.

* find the Name of the Sailors who didn't reserve any boat.

i) get the Name of the Sailors from Sailor Database

$$\Pi_{Sname}(Sailor)$$

ii) get the Name of the Sailor from Reserve Database

$$\Pi_{Sname}\left(\sigma_{R}(Reserve \times Sailor)\right)$$

$$\Pi_{sname}\left(\sigma_{(Boat.bid = Reserves.Bid) \text{ and } (Reserves.Sid = Sailors.Sid)}(Boat \times Reserve \times Sailors)\right)$$

iii) $r_1 - r_2$ gives didnot reserve any boat Name of the Sailors.

(*) find the Name of the Depositors from SBH and who does not have any loan from SBH (or) find the Name of the Depositors who is having an account in SBH

i. find the Name of the Depositors who is having an account in SBH.

$$r_1 \leftarrow \Pi_{cname}\left(\sigma_{bname = 'SBH'}(Depositor)\right)$$

ii. get the Name of the customers who has taken loan from SBH

$$r_2 \leftarrow \Pi_{cname}\left(\sigma_{bname = 'SBH'}(borrow)\right)$$

iii) $r_1 - r_2$ gives name of the customers who is having accounting in SBH but not a loan in SBH.

<u>Assignment operator:</u> ($\leftarrow$) this operator is used for assigning R.H.S Query Expression result to a single L.H.S Temporary relation (Temporary is the relation is the Variable name)

Syntax: $r_1 \leftarrow$ Query Expression

Query Expression is a composition of collection of relational algebra operators. r_1 is the temporary relation name. (Ex— we can see (*

Rename (ρ) :

This operator is used for renaming the old Relation name (Database name) to new Relation name.

In other words it is used for giving an alias name to the existing Relation.

Syntax: $\rho_{(\text{New Relationname})}(\text{Old Relationname})$

Ex:- $\rho_{(C)}(\text{customer})$

Here 'C' is the alias name to customer relation Hence the columns of customer relation can be represented as C.Cname, C. street, C. city

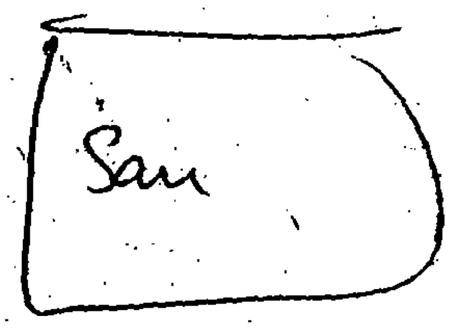
Q) find the Name of the customer's who is living in same street where John is living?

(OR)

find the name of the customer's who's street is similar to John street.

Cnam	Street	ndt
x	Ame	—
y	H	—
John	A	

C



$$\Pi_{cname}\left(\sigma_{(customer.name = 'John') \text{ and } (customer.street = c.street) \text{ and } (c.cname != 'John')}\left(customer \times \rho_{c}(customer)\right)\right)$$

* find name of the Employee who's Designation is Similar to Ravi

Emp	Ename	Sal	Dept
10	X	1.2	—
20	Ravi	1.2	—

$$\Pi_{Ename}\left(\sigma_{(Employee.ename = 'Ravi') \text{ and } (employee.desig = e.designation) \text{ and } (e.ename != 'Ravi')}\left(employee \times \rho_{e}(employee)\right)\right)$$

* find the Name of the Supplier who's Supplies the parts which are Similar to xyz supplier.

$$\Pi_{Sname}\left(supplier \times \rho_{(s)}(supplier)\right)$$

(*-1)

Ans

$$\pi_{S.Sname}\left(\sigma_{(Supplier.sname='xyz') \text{ and } (suppliers.sid = catalog.sid) \text{ and } (catalog.pid = parts.pid) \text{ and } (parts.pname = p.pname) \text{ and } (p.pid = c.pid) \text{ and } (c.sid = s.sid) \text{ and } (s.sname != 'xyz')}\left(suppliers \times \rho_s \times catalog \times \rho_c \times parts \times \rho_p\right)\right)$$

Ans:

* Find the Name of the Sailors ~~which~~ are similar to who Reserved Simone the boat which are similar to ~~'intendake'~~ ake

Sail

Divisi

(7-1) (Q): find the name of the suppliers who is not supplied red parts.

Ans:

Ans:

$$r_1 \leftarrow \pi_{sname}(\text{supplier})$$

$$r_2 \leftarrow \pi_{sname}(\sigma_{(color! = 'red') \& (part.pid = catalog.pid) \& (catalog.sid = supplier.sid)}(\text{parts} \times \text{cost} \times \text{supplier}))$$

Sai//

Division operation:

Relational Calculas. [Tuple relational calculas)
[TRC]

i) It is an alternative arrangement for relational algebra for Querying the data.

ii) Queries can be expressed even in relational calculas for retrieving the Data in relational model.

iii) Relational calculas is one of the non-procedural language and it does not contain the operators like relational algebra. Relational calculas is having big influence in Developing commercial languages like SQL.

iv) The Syntax for a Query in tuple relational calculas is

$$\{t / p(t)\}$$

Here 't' is one of the tuple variable. tuple variable is one which contains output of the relational calculas Query.

'p(t)' → represents predicate or first order logic. first order logic is one in which we write logic of the Query.

Ex:

$$\{t / \exists s \in \text{Sailors} \ldots\}$$

the columns of Sailors will be retrieved Query as S.Sid, S.Sname, S.rating

once the condition is satisfied the tuple variable 't' contains output.

Sai

x: find the Name of the Sailors whose rating > 7.

Ans: $\{t / \exists s \in Sailors\ (\underbrace{s.rating > 7}_{condition} \wedge \underbrace{t.sname = s.sname}_{Assignment})\}$

Note: In the above Query '$\wedge$' is known as and it is used for Seperating Condition (s) and assignments (result)

(2) find the Name of the Sailors and their id's whose ranges rating between 5 & 8.

$\{t / \exists s \in Sailors\ (s.rating >= 5 \wedge s.rating <= 8 \wedge t.sname = s.sname, t.sid = s.sid)\}$

(3) Display all the details of Sailors

$\{t / \exists s \in Sailors\ (t.sname = s.sname, t.sid = s.sid, t.rating = s.rating)\}$

(or)

$\{t / \exists t \in Sailors\}$

Note: when ever the alias name (t) and tuple variable is Same the result of retrieved Db is Copied into tuple variable.

4) find the Name of the Sailors who reserves the red boat

$\{t / \exists B \in Boats, R \in Reserves, S \in Sailors\ (B.color = 'red' \wedge (B.bid = R.bid) \wedge (R.sid = S.sid) \wedge (t.sname = S.sname)$

Note:

(5) not

{t

(6) for an

(7) {

Si

Note:

(

(8) f

the

∧

Note: $!=$ not Equal to In Relational algebra
$\neg$ not Equal to in Relational calculas.

(5) find the Name of the Sailor who's rating is not seven.

$\{t \mid \exists s \in Sailors\ (s.rating \neg 7 \wedge t.sname = s.sname)\}$

(6) find the name of the customer who is having an account and loan in SBH.

$\{t \mid \exists d \in Deposit\ (d.bname = 'SBH') \wedge$

$b \in Borrow\ (b.bname = 'SBH' \wedge d.cname = b.cname$

$\wedge\ t.cname = b.cname)\}$

(7) find the name of the suppliers who supplies Either red or Green boats or both

$\{t \mid \exists p \in parts,\ s \in Sailors,\ c \in catalog\ (p.color = red \wedge$

$p.pid = c.pid \wedge c.sid = s.sid \wedge t.sname = s.sname)$

$\vee\ (p.color = 'Green' \wedge p.pid = c.pid \wedge c.sid = s.sid \wedge$

$t.sname = s.sname)\}$

Note: 'v' this operator treat as union (U) operator of Relational algebra.

(8) find the name of the customer who living the same street where 'abc' lives.

$\{t \mid \exists c_1 \in customer,\ c_2 \in customer\ (c_1.cname = 'abc'$

$\wedge\ c_1.street = c_2.street \wedge c_2.cname \neg 'abc' \wedge t.cname =$

$c_2.cname)\}$

(1) find the name of the branches who does not have customers, Deposits.

$\{ t \mid \exists B \in Branch (t.Bname = B.Bname) - \exists D \in Deposit ($
$t.Bname = D.bnand \}$

(or)

$\{ t \mid \exists B \in Branch, D \in Deposit (t.Bname = B.Bname) -$
$(t.Bname = D.Bname) \}$

Q:

Consider the following relations

Person (pid, Pname, addr)

Company (pid, cname, sal)

Works (pid, cname, bnget, city)

Answer the following Queries in both Relational algebra Tuple Relational calculus.

(1) Find the Name of the Person who works in HCL.

RA :

$\pi_{Pname} (\sigma_{(Company.cname = HCL) \text{ and } (Company.pid = Person}$ (Company × Person))

RC:

$\{ t \mid \exists c \in Company, p \in person (c.cname = 'HCL' \wedge c.pid =$
$p.pid \wedge t.pname = p.Pname$

(2) find the Name of the persons who is working in those Companys which are located in Bangalore

$\pi_{pname} (\sigma_{(city = Banglore) \wedge (works.pid = person.pid)}$ (works × person)

$\{ t \mid \exists w \in works, p \in person (w.city = Banglore \wedge$
$w.pid = p.pid \wedge$
$t.pname = p.pname) \}$

Sem V

Expressive Power of Relational algebra and Relational Calculus.

Relational algebra and Relational calculus are the two formal languages which are having the Power of expressing the Queries in relational model.

Similarities:

(1). Relationalalgebra & R.C are two formal language (Since no Software Exist)

(2). Using This two languages we can retrieve the data only from Relational model.

(3) Both is languages ~~Does~~ not contain any operators for Performing insertion and ~~Deletion~~ operations and updation operations.

(4) This two languages gives enough influence for ~~developing~~ Commercial Database product like SQL, DBQ, SQL Server Etc. (Known as RDBMS Product)

(5) This two languages are unable to provide the group functions or aggregate functions for numarical calculations. (sum(), avg(), min(), count(), max())

Differences:

Relational algebra	Relational calculus
1) It is one of the Procedural language.	1) It is one of the non-procedural language
2) The Queries which we write in Relational algebra are known as Safe Queries.	2) the Queries which we write in Relational calculus are known as un-safe Queries.
3) Relational algebra Queries are treated as Relationally Complete.	3) Relational calculus Queries are not relationally Complete
4) In Relational algebra Queries can be Expressed Easily since we have Different operators for different Purposes.	(4) In Relational calculus, it may be difficult to Express the Query since we Does not have Distinct operators for different Purposes.

Domain Relational calculus:

It is also one of the non-Procedural language which can be used for Expressing the Queries in Relational Model.

The General formate of DRC Query is

$$\{\langle v_1, v_2 \ldots v_n \rangle \mid \exists (v_1, v_2 \ldots v_n) \in \text{Relation Name and}$$

first order logic of DRC $\}$

$\langle v_1, v_2 --- v_n \rangle \rightarrow$ represent alternative names to the column names of a particular Relation. at the time of Retrieving the Data from the Relation we must Specify alternative names of the Columns Only but not actual columns.

x - find Sailorname and their id whos rating more than 7.

$$\{\langle s, n \rangle \mid \exists (s, n, r) \in Sailor \wedge r > 7\}$$

Note: $\langle s, n, r \rangle$ are alternative names to Sid, Sname, rate are Sailor relation.

- find the Name of the Sailor who reserved red boat.

$$\{\langle a \rangle \mid \exists \langle B, n, c \rangle \in Boats, \langle s, i, b \rangle \in Reserves, \langle d, a, r \rangle \in Sailor, c = 'Red' \wedge B = b \wedge s = d\}$$

UNIT - III

→ Data Organised

a. File Organisation & Types of Indexes
S. I. U. P. E
C. I. V. U. C. I

b. Tree-based Indexing
— ISAM
— B-Tree *

c. Hash Based Indexing
- Static Hashing
- Extendable
- Liner Hashing
- E. H V SL. H

(i) The basic aim of DBMS is to organised the data in terms of collection of records. In General in DBMS a table is also known as file.

(ii) File Organisation is a Mechanism of organising the data in the form of records. It provides the effective operations with optimised time. i.e the database operations like insertion, Deletion, and updation of record must take less amount of time.

(iii) An Indexing is the kind of data structure which allocates to maintain a separate column as a primary key for locating the record uniqly by taking less time.

- tion
Iden-
appr
4
file
are
Page
Prima
Knoc
Bubber
layer
addi
memor
DISK S
which
Space
inter
new
map

Hence to perform any Database operation effectively first we must locate index identify the key and key will point to the appropriate record in the Database / file.

(4) Internally the data is organising in the file in the form of records and records are internally placed in pages in the file.

(5) Page :-

— A unit of information Transfered b/w Primary Memory and Secondary memory is known as Page. (or)

— A unit of information read from (or) written to Harddisk (file) is known as Page.

(6) Buffer Manager:

Buffer Manager is one of the software layer in DBMS which exchanges the pages from Harddisk into Primary memory and primary memory to Harddisk.

Disk Space Manager is also one of the software layer which is used for allocating Enough memory space in the Harddisk for inserting new records.

This layer generates the following interaptions

(a) If diskspace is not available then new record may not be inserted and interaption may be Generated.

(b) when we are trying to create a file which is already existing then Disk Space Manager may not be creating the file once again and it may generate an interrupt.

(c) when we remove any records from the database (file) then records will be removed from the file and file contains empty pages (dirty pages) and empty pages are collected by Disk Space Manager and given to operating system.

Hence allocation of proper memory space and deallocation of memory space can be performed by Disk Space Manager while we are performing database operations.

- The channel between buffer Manager and DiskSpace Manager is known as file layer / access methods layer.

All the records in the DB are equality with a unique column name called rid (record id).

in oracle rid is used as rowid or rownum

INDEXing:

An index is a data structure which contains various data entries which are used for locating records in the file/database.

The concept of index can be developed for fast access, performing fast database operations like, insertion, deletions, updations.

Using indexing we can reduce the cost of page I/O will be reduced. Hence it increases performance of database applications.

An index can maintain three types of entries/alternatives for processing the records in the database.

1. The data entry of an index can be simply a search key which is denoted as 'K' (value of search key)

2. The data entry of an index can be combination of search key and list of rid's and it is denoted as <K, rid>

3. The data entry of an index can be combination of search keys and range of rid's which is denoted as <K, ridlist>

1. Search key(K)
2. <K, rid>
3. <K, ridlist>

Types of Indexes:

→ In DBMS we have two types of in

They are No 1. clusted vs unclusted

2. Primary vs Secondary

Clustered

Clustered index is one. in which all the entries in the index are in the one order which are available in the file, i.e which ever order the records are available in the file, in the same order entries of index also will be Maintained.

→ In clustered index searching time of a record is very less.

clustered index gives high performance for database applications.

→ Most of the popular DB Products allows clustered indexes.

un-clustered

un-clustered is one which entries of index is random without ordering in which order The records are Maintain in the file.

b. In un-clustered index searching time of a record takes more time

c. un-clustered index gives less performance

d. DB products may not be following

→ Same order
→ searching time less
→ high performance
→ Most of the popular DB products follows clustered

random order
more
performance may not be better

Prim

Primary

taing

tabase

ta En

using

peration

faster Sin

identity

3. Primary

allow R

4. In Pr

page

indexing

index:

1) Hash

Organ

Ins

Primary Index	Secondary Index
1. Primary index is one which contains Primary key of the database is available as Data entry in the index.	1. Secondary index is one which contains some entry in the index which is other than Primary key.
2. using Primary index the operation Searching is made faster since we are able to identify the records uniqly	2. In Secondary index the operation Searching takes more amount of time since we are unable to identify the record of the Database uniqly
3. Primary index doesn't allow Duplicates	Secondary index allowses to have Duplicates.
4. In Primary index cost of page I/O is less.	In Secondary index cost of page I/O is more.

Indexing Structure (or) Approches for Maintaining indexes

In DBMS we have two types of indexing structures they are

1) Hashbased indexing
2) Tree Based indexing

1) Hashbased indexing:

(a) Hashing is a mechanism of Organising the Data in the Main Memory of the computer by calculating hashcodes

b) A hash code is a numerical number which can be calculated by a defined hash function. In This indexing scheme the have the following hash function.

$$H(x_i) = x_i \bmod n.$$

'x_i' - represents set of key values.

'n' represents null of buckets.

(d) In general DB file contains collection of records which are internally available in n-number of pages. Each page is represented in '1' bucket i.e each record is represented in one bucket.

(e) Hash function in This schema is used for locating an appropriate bucket number.

(f) If a bucket is already contain (one record) or a page then that bucket contains Primary Page. If insert any new record and it is placing (inserting) in the same bucket which already contains a primary page then the new record will be inserted in the same bucket as a overflow page.

Hash based indexing contains 3 types. They are

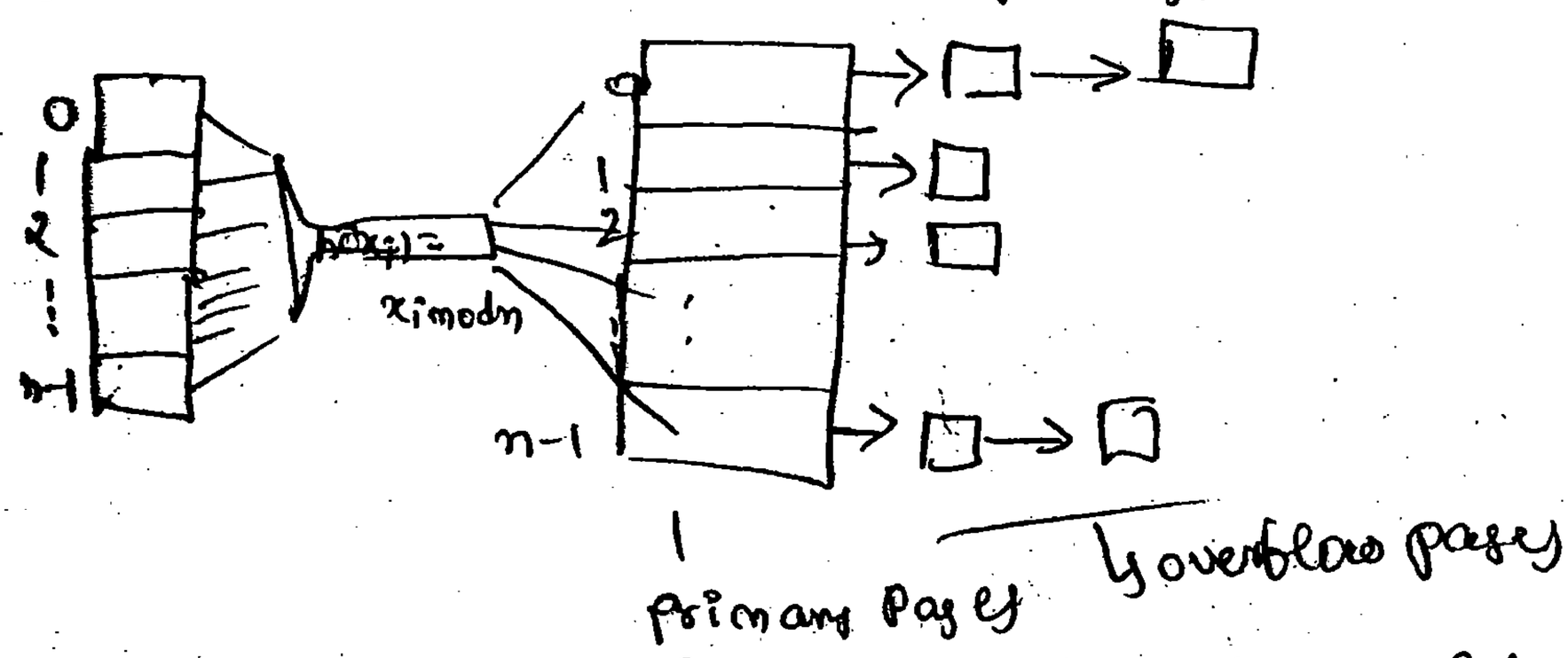
1) static Hashing
2) Extendable Hashing
3) Liner Hashing.

(i) **Static Hashing:** Static Hashing is one in which the no of records are represented in fixed no of buckets. In this mechanism each bucket is containing one primary page. If new record is inserted into some bucket which is already containing a primary page then the new record will be added to the same bucket as a overflow page. In this key the bucket number are ranges from 0 to n-1 where n represents no of buckets, which are fixed. to calculate bucket numbers we use the following hash function. $h(x_i) = x_i \bmod n$.

Here x_i - represents set of key values. Practically 'x_i' represents primary key columns of the DB.

(Empno, Sno, SSN etc (Social Security no)

The General representation of static hashing scheme is illustrated in the following diagr



Ex: construct the static hash based indexing scheme for the following DB Emp

Empno	Ename	Sal	Design
100	abc	1.2	PL
101	xyz	1.3	SA
102	pqr	1.4	|
|	|	|	|
100	xx	0.5	SE

$h(x_i) = x_i \bmod n$

$h(100) = 100 \bmod 5 = 0$

$h(101) = 101 \bmod 5 = 1$

$h(102) = 102 \bmod 5 = 2$

$h(103) = 103 \bmod 5 = 3$

$h(104) = 104 \bmod 5 = 4$

$h(105) = 105 \bmod 5 = 0$

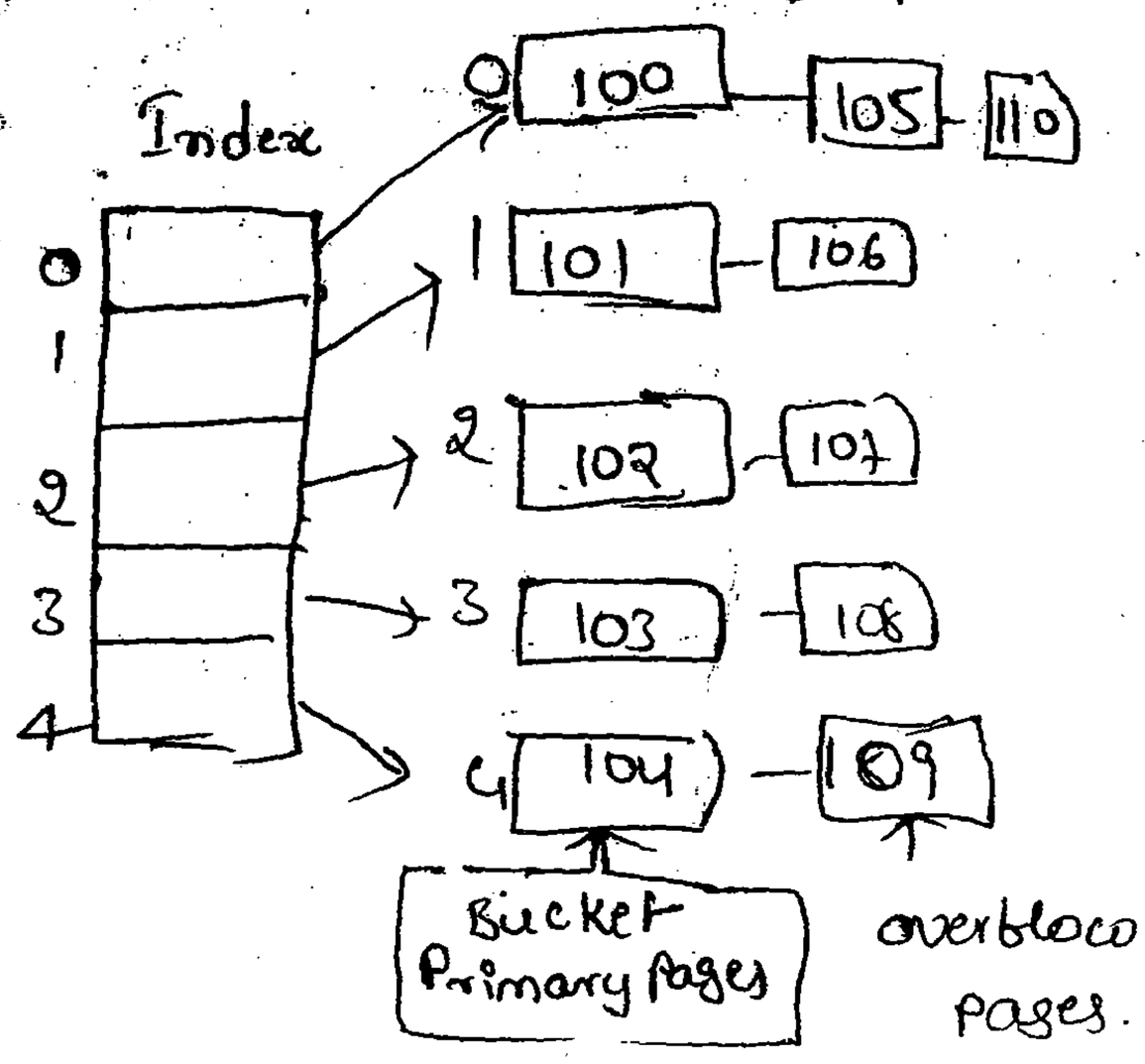
$h(106) = 106$ " $= 1$

$h(07) = 107$ " $= 2$

$h(108) = 108$ " $= 3$

$h(109) = 109$ " $= 4$

$h(110) = 110$ " $= 0$



2(4-0
2(2-0
1

32	16	8	4	2	1
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	0

Dis-advantages of static hashing:

- In static hashing we will be given fixed number of buckets. hence we are unable to represent Dinamic no.of records in the Primary Page of the bucket.

- If we Remove Some Records from the Existing index, There is a possibility that Empty overflow Pages and Primary Pages. which leads to wastage of memory space.

as a
for a
of the

Ex:
Sol.

case a:

as and when more no of overflow pages occures for a Primary page which leads to more amount of time for searching operation.

4a Extentable hashing Technique:

ex: Construct the indexing structure for the following set of key values using extandable hashing. 4, 1, 10, 15, 12, 5, 13, 21, 7, 19, 32, 16, 20, 9

Sol.

32	16	4	2	1
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0

h(4) = 0100

h(1) = 0001

h(10) = 1010

h(15) = 1111

h(12) = 1100

h(5) = 0101

h(13) = 1101

h(21) = 10101

h(7) = 0111

h(19) = 10011

h(32) = 10000

h(16) = 1000

h(20) = 1100

h(9) = 1001

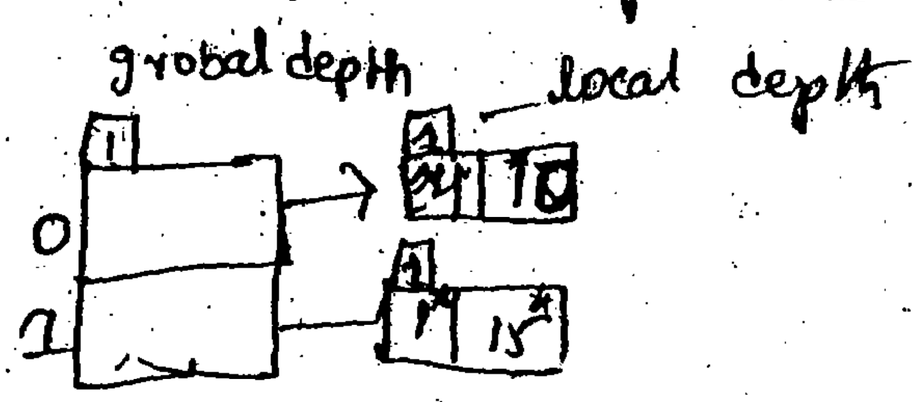
2 (9, -1)

2 (4, 0)

2 (2, 0)

1

ase a: Consider Global depth = 1.



inserting keys 4, 1, 10, 15

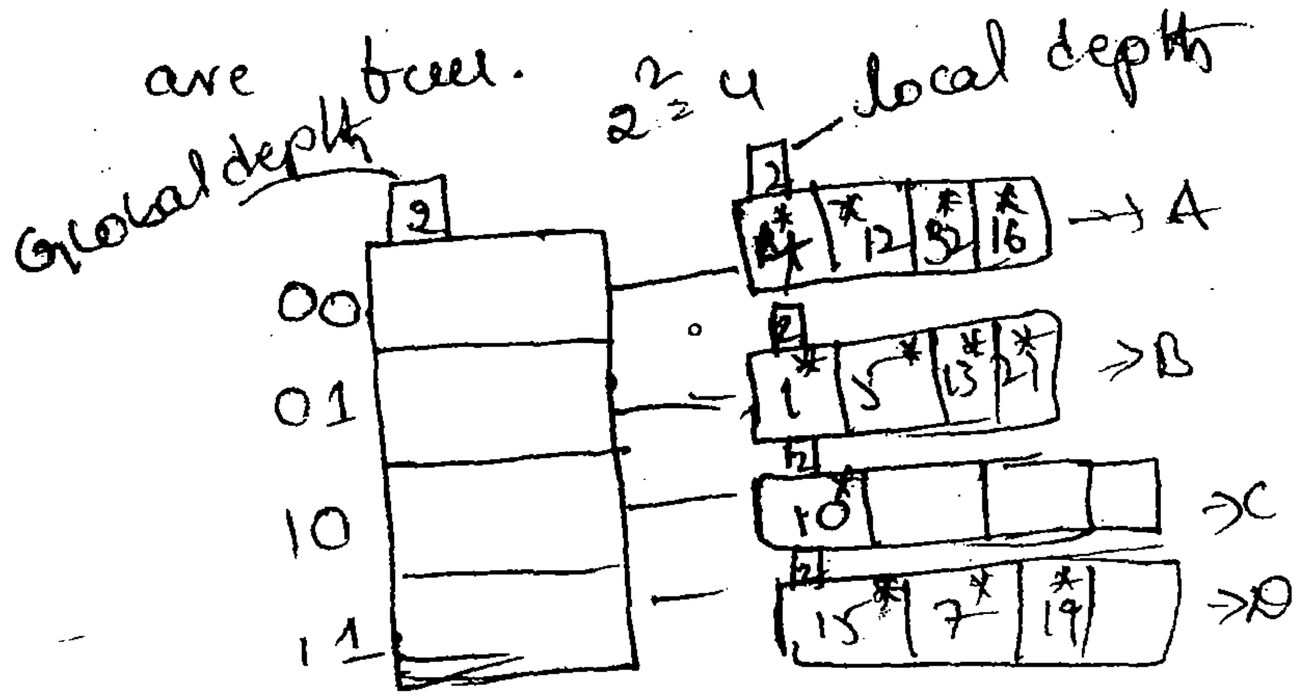
inserting the key 12

is not possible. Since all the buckets are bu[illegible] case c:

In order to insert the rest of the key values we must Double the directory pointers by increasing global depth by one.

000
001
010
011
100
101
110
111

case b: Consider global depth = 2. Hence local Depth also becomes two since all the bu[illegible] are full. $2^2 = 4$ local depth



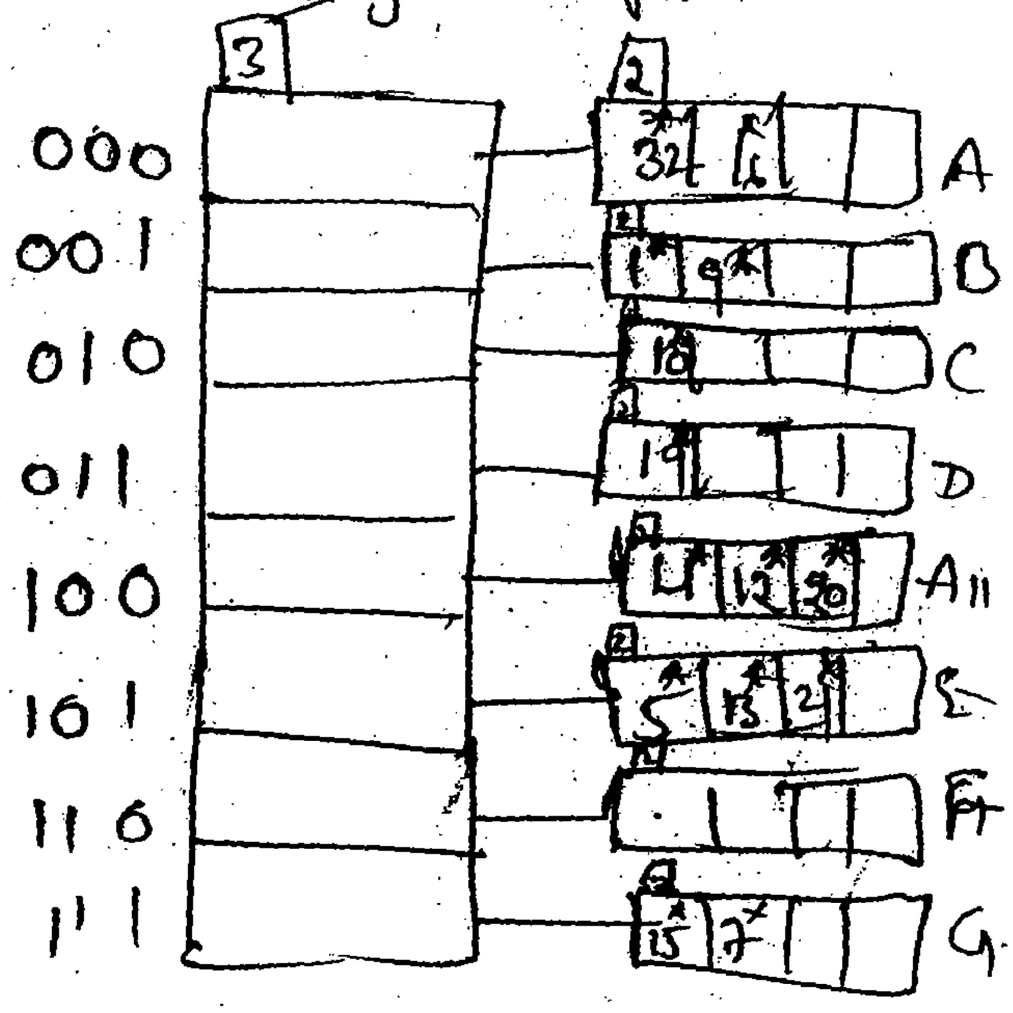
Inserting 4, 12, 32, 16, 1, 5, 13, 21, 10, 15, 7, 19.

In order to insert the key 20 in the bucket A, there is no space is available. Hence bucket a must split into 2. They are

Bucket A, Bucket A_1 (Split Image)

In order to insert element 20, double the Directory of pointers by increasing global dept by 1 and also increase local depth for bucket a. Dat A by 1.

Consider Global depth = 3.



inserting 21, 1, 10, 18, 12, 5, 13, 21, 7, 19, 32, 16, 20, 9

Sai

Hash to → Elim dis stru

(i) It is one of the Hashbased indexing structure. using this indexing we can Eliminate all the Disadvantages of static hashing.

(ii) In this Technique we use a hash function which will Generate binary code for a Given keyvalue. using this binary code we locate or insert or update or select or Delete the record of the file.

while we are inserting the key as a data Entry in the index, first we must consider the least Significant bit of the binary code of a search key, in the

Second level we must consider trust two least Significant bits and in Third level we consider trust Three least Significant bits and so on.

4) As and when no. of levels of changing as and when Primary Pages are full at certain level, the Directry of Pointers will be Double and increase the Global Depth of the index Structure by '1'.

→ If the Primary Page is full while we are inserting a new data Entry. Then split the Primary page in two they are old Primary Page (Old bucket) and "Split Image" (New bucket)

→ when the Primary Page is split into two Pages Their local Depth may or may not be change in. Local Depth of those two Pages will be also increased when they contain full of values. otherwise Previous local Depth only will be continued.

→ while we are changing local Depth and global Depth values, we must ensure that the value of local Depth must be ≤ the Value of ~~Global Dep~~ Global Depth

→ As and when Directory pointers are doubled the will be reorganised into various Primary pages and Split images. This Type of re-organisation of the data is known as "Skewed Distribution".

→ As the above process is repeated untill all the Key values are properly maintained in index Structure.

Dis-advantages:

1. ~~The~~ when Number of Pointers are Doubled, The Existing data will be re-organised. for re-organisation of the data it takes more time.

2. the Number of page I/O cost is more and Their is a possibility of loosing Performance.

3. As and when global Depth is increasing the Directory pointers are Doubled. in Some of the Extreme Situations, for a large data base, main memory contains more number of pointers and it takes more time to process.

- re organisation takes more time
- no. of page I/O cost is more & loosing Performance
- more time to process

Linear indexing:-

Construct a linear indexing structure for the following set of key values.

32, 9, 14, 31, 44, 25, 5, 18, 10, 35, 7, 11, 36, 30, 43, 37, 29, 22, 66, 34, 50.

Sol:

32 16 8 4 2 1

$h(32) = 100000$

$h(9) = 1001$

$h(14) = 1110$

$h(31) = 11111$

$h(44) = 101100$

$h(25) = 11001$

$h(5) = 101$

$h(18) = 10010$

$h(10) = 1010$

$h(35) = 100011$

$h(7) = 111$

$h(11) = 1011$

$h(36) = 100100$

$h(30) = 11110$

$h(43) = 101011$

$h(37) = 100101$

$h(29) = 11101$

$h(22) = 10110$

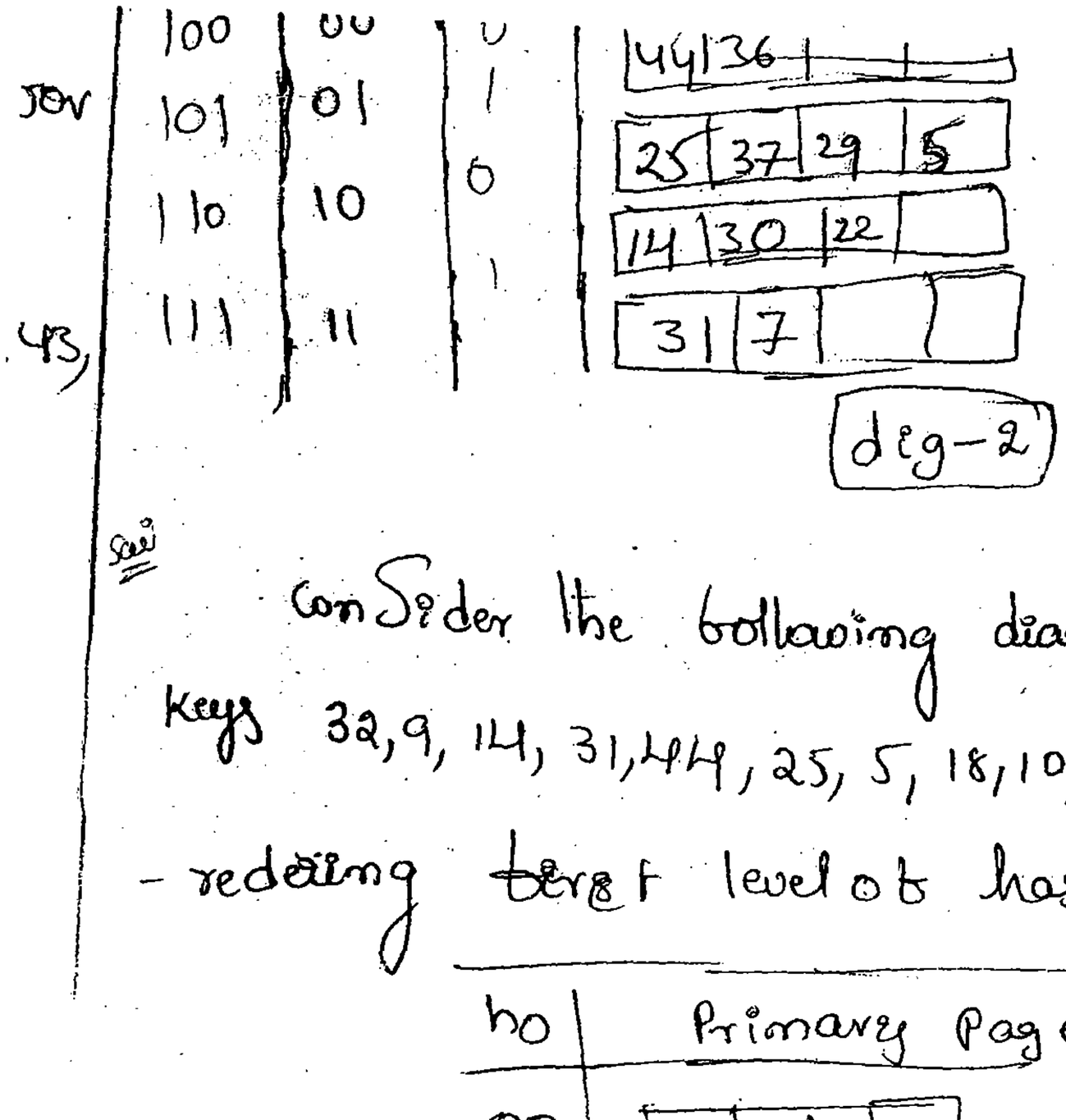
$h(66) = 1000010$

$h(34) = 100010$

$h(50) = 110010$

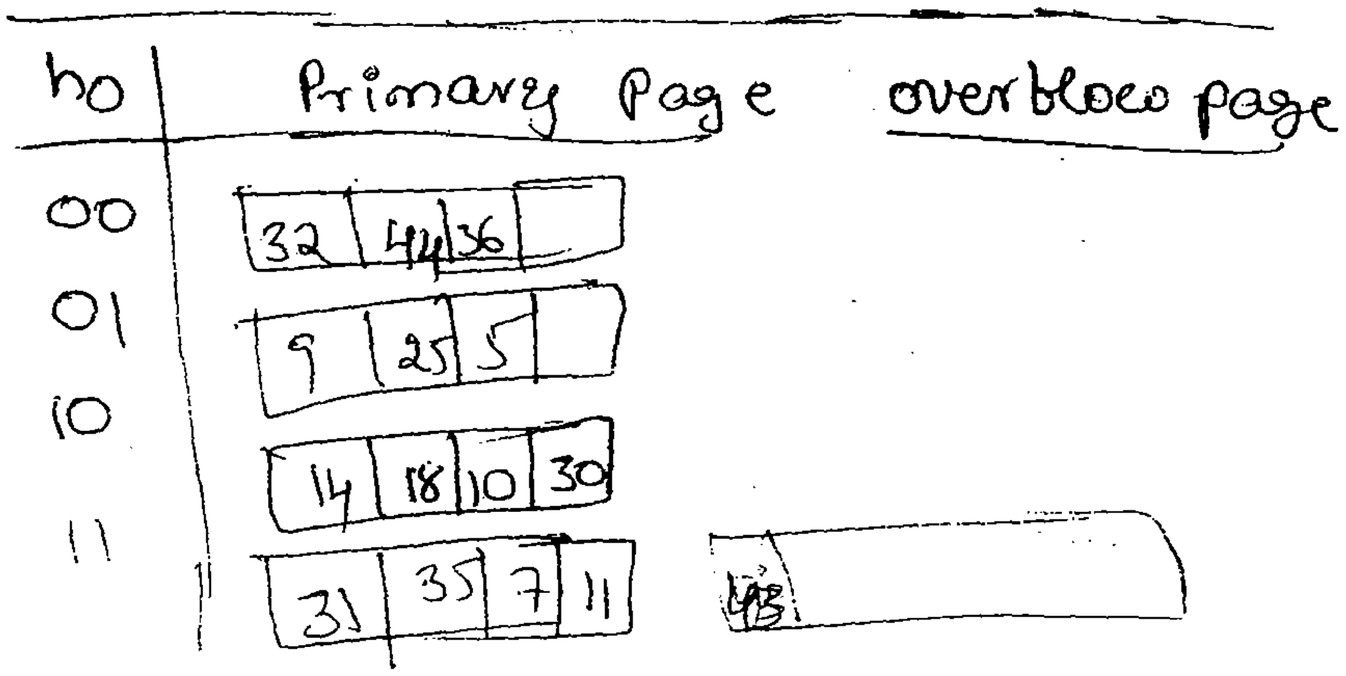
2|66 — 0
2|33 — 1
2|16 — 0
2|8 — 0
2|4 — 0
2|2 — 0

h_2	h_1	h_0	Primary pages	Overflow pages
000	00	0	35	
001	01	1	9, 25	
010	10	0	18, 10, 34, 66, 50	66
011	11	1	31, 11, 43	



Sol

Consider the following diagram for inserting the keys 32, 9, 14, 31, 44, 25, 5, 18, 10, 35, 7, 11, 36, 30 by considering first level of hash function i.e h_0.



The key value 43 can't be inserted at one-one of h_0. Hence the key value 43 is added as a over block to the above diagrame.

Linear hashing never maintains overblock pages. If at all any overblow pages coming then which will be merged with existing primary pages in the next levels.

→ dig-2

In the above diagramme we have inserted the keys 37, 29, 66, 34 the key value 50 is unable to insert at 010 of

h_1 hence the value 50 is added as a over- primar
blow page and it will be merged in the
next levels i.e h_2. in the

This is also one of the indexing scheme is calls
for maintaining various data entries. This scheme
contains a hash which will generate bi- an c
-nary code of the given key values.

page -
level

The basic idea in this scheme is
that we use family of hash functions to
map the key values into one primary page out of
m- primary pages. in particular hash function
h_0 maps a key value in one of the primary
page out of 4 primary pages, hash function h_1
maps a value in one primary page out of 8 primary
pages and so on.

-ged

u

never

we c
-catic

main
memo

in General hash function h_i maps a
value in one of the primary page out of m-pri
-mary pages, hash function h_{i+1} maps a key
value in one of the primary page out of
$2m$ primary pages and so on.

→ del

linear

&

D we
of p

therefore as and when level of
hash function is increasing, number of

1- Primary pages are increased by multiples of two

At the time of inserting a new value in the primary page and if the primary page is full then a new value cannot be inserted in same primary page but it can be added as an overflow page.

When ever we get first overflow page in linear hashing, we must go to next higher level hash function.

Hence the overflow page will be merged with any of the existing primary pages.

In extendable hashing, linear hashing maintains a directory of pointers. Hence improve the performance of an application.

Since it never takes much amount of main memory space.

→ Differenciate b/w Extendable hashing and linear hashing.

Extendable	linear
1) we maintain Directy of pointers pointing primary pages	1) we maintain different levels of hash functions for mapping the key/values into various primary pages

1) No overflow pages are Maintained.	(2) overflow pages are occuring and they are finally rounded with Primary pages.
3) Re-organisation of the Data is more.	(3) Re-organisation of the Data is less.
4) we Maintain Split Image of a bucket when the bucket is full.	(4) we never maintain Split Image (instead of it we maintaind overflow page)
5) cost of page I/O is more.	(5) cost of page I/O is very less.
6) average page cost Performing various DB operations is more.	(6) avarage Page cost for Performing various DB operations is less.

Tree Bashed Indexing

(1) It's also one of the technique for achieving Indexing of a Database/file. in this Mechanism the collection of keys maintained as an index in the form of tree. a tree is a Data Structure in which there exist a Specially Disignated node called root node and other nodes are particular into Different Disjoint Subtrees. Subtrees are again trees.

Tree Based Indexing contains two types they are

NO 1). Indexed Sequencial access method (ISAM)

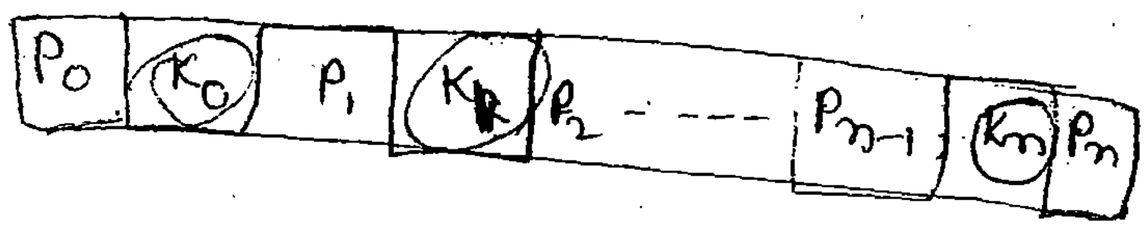
2) B^+ trees.

ISAM:

① This is one of the indexing Scheme following the concept of trees for maintaining an index.

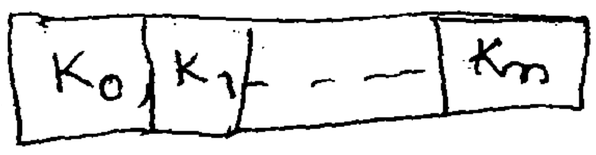
② In this Scheme the Data is Organising in the form of non leaf and leaf pages.

The General formate of non leaf page is



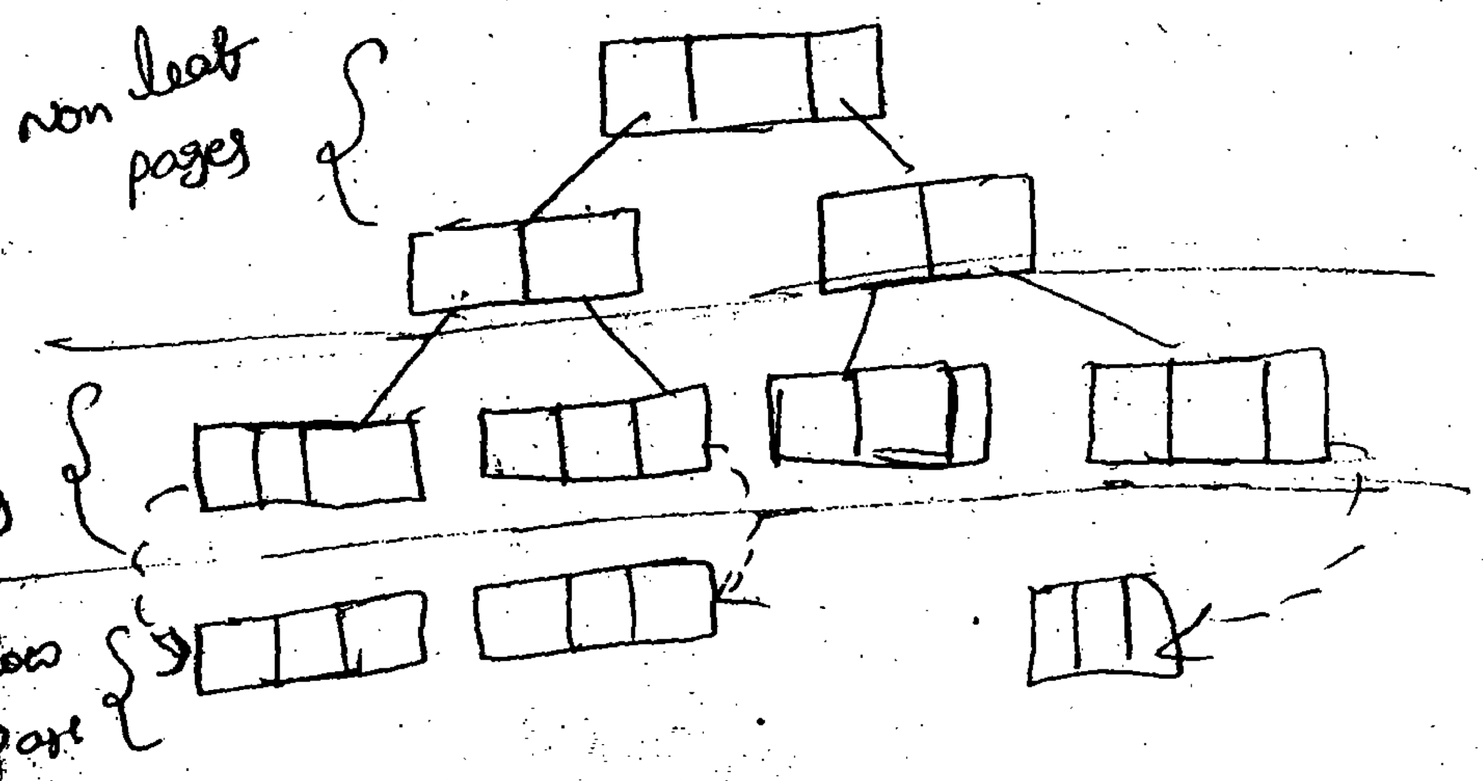
Here $P_0, P_1 ---- P_{n-1}$ and P_n are the Pointers Pointing to left and Right leaf pages.

The General formate of leaf page is



K_0, K_1, K_n represents list of key values.

The architectural Diagram for ISAM is given below



Ex: Construct an indexing structure using ISAM. for the following key values.

Min no key = 1

min no pointer

10, 15, 20, 27, 33, 37, 40, 46, 51, 55, 63, 97, 23, 48, 41, 42. assume the order of the node or maximum no of Pointers = 3

Sol: when the set of key values are given, the Data must be in Sorted Order.

→ Order of a node = m = 3.

Maximum no. of keys = m-1 = 2,

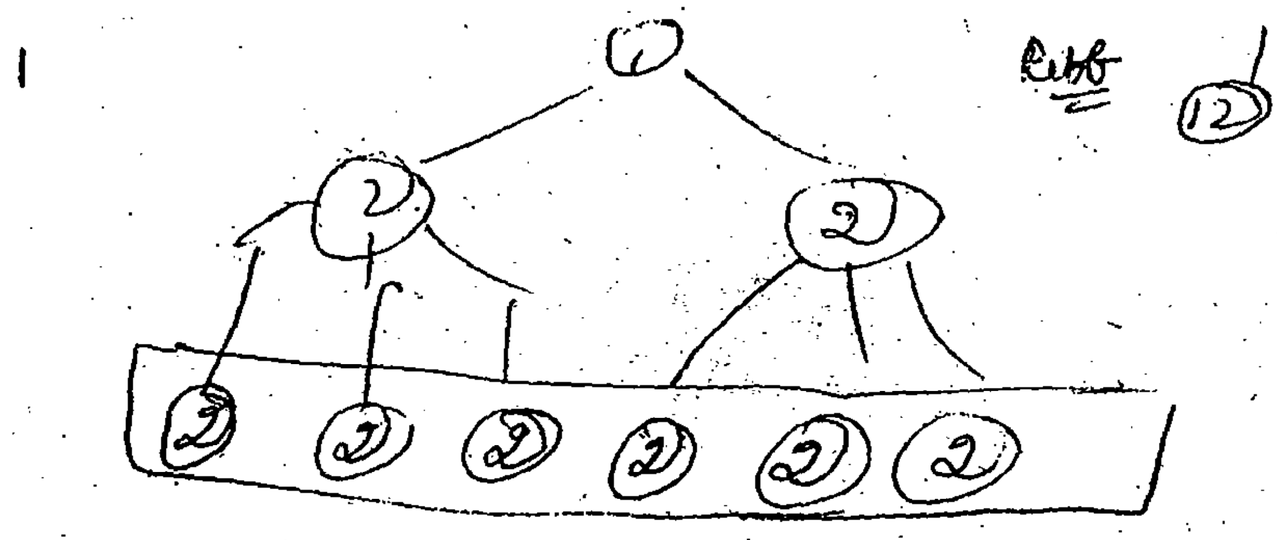
Minum no. of Pointer = $\lceil m/2 \rceil = \lceil 3/2 \rceil = 2$

Minimum no. of key = $\left\lceil \frac{m-1}{2} \right\rceil = \lceil 2/2 \rceil = 1$

root node Exception:

minum no. of key = 1.

minimum no. of pointer = 2,



Page as

in Se

48, 4

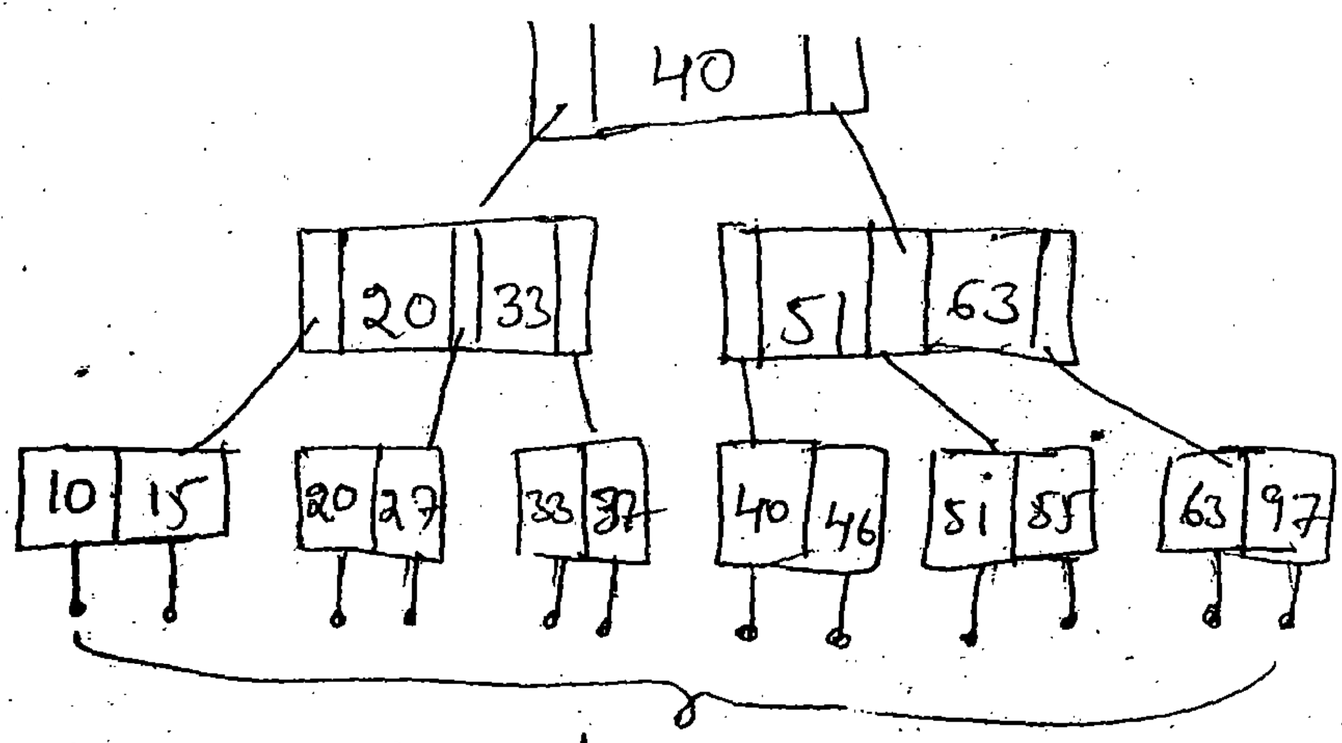
in Se

41

Overflo

mo di

non l

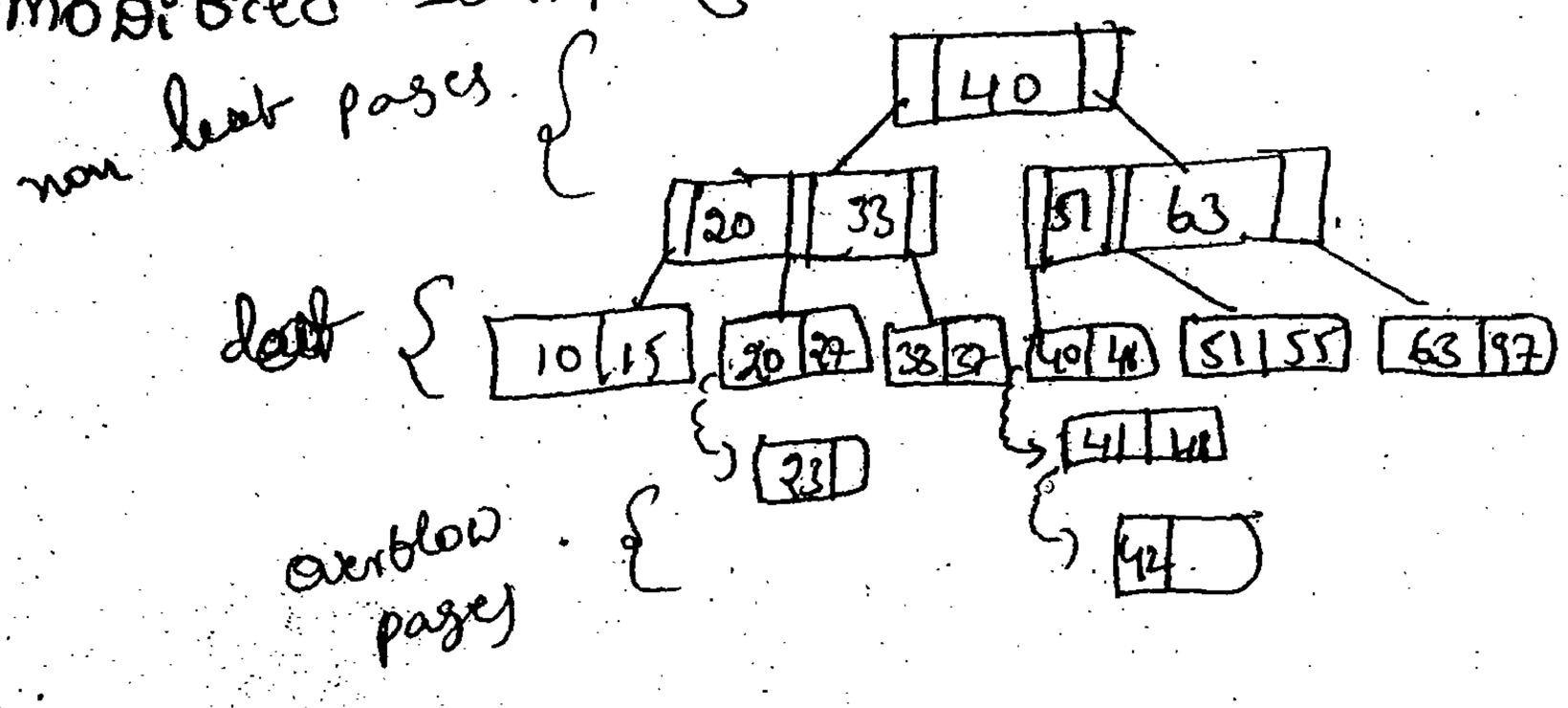


Pointing to actual records of database.

The key value 23 can't be inserted in leaf page [20|27]. Hence the key value 23 is added as overflow page.

The key values [48|41] can't be inserted in leafpage [40|46]. Hence the keys 48, 41 added as a overflow page.

The key value 42 can't be inserted in the leaf page [40|46] and also it can't be inserted in the overflow page [41|48]. Hence the key 42 is inserted as overflow page for [41|48] overflow page. Hence the modified ISAM is



→ ISAM The Most Suitable for those files which are having fixed no. of records. i.e ISAM is static in Nature.

→ If we construct an indexing structure growing → (4) using ISAM, It the grows rapidly then their exist more number of overflow pages. which leads to more performance.

→ From the indexing structure It we remove more no. of records then their is the waste of memory space.

→ In the indexing structure of ISAM, Both leaf and non leaf pages Duplicate keys are represented i.e Data Redandancy is available. which leads to higher memory cost and inconsistant result.

(1) . Serching takes more amount of time. Since it's starts from non leaf and terminates at either at leaf pages or overflow pages. which are attached to leaf pages.

- static in Nature
- more no. of records
- remove more record
- D.R → higher
- Searching →

B^+ TREE :

(1) B^+ Tree is one of the Dynamic indexing Structure.

(2) . when we are creating indexing using B^+ Tree the data entries are represented in the form of

leaf and non leaf pages.

(3) • In B^+ Trees leaf pages are also known as External nodes where as non leaf pages are also known as internal nodes.

(4) • The format of non leaf / internal node is Given below for representing the data entry in the index.

P_0	K_0	P_1	- - - -	P_{n-1}	K_n	P_n

Here P_0, P_{n-1}, P_n → represents set of pointers pointing to the left and right leaf/non leaf pages.

Here $K_0, K_1 \ldots K_n$ represents set of keys which are nothing but Primary keys of the database.

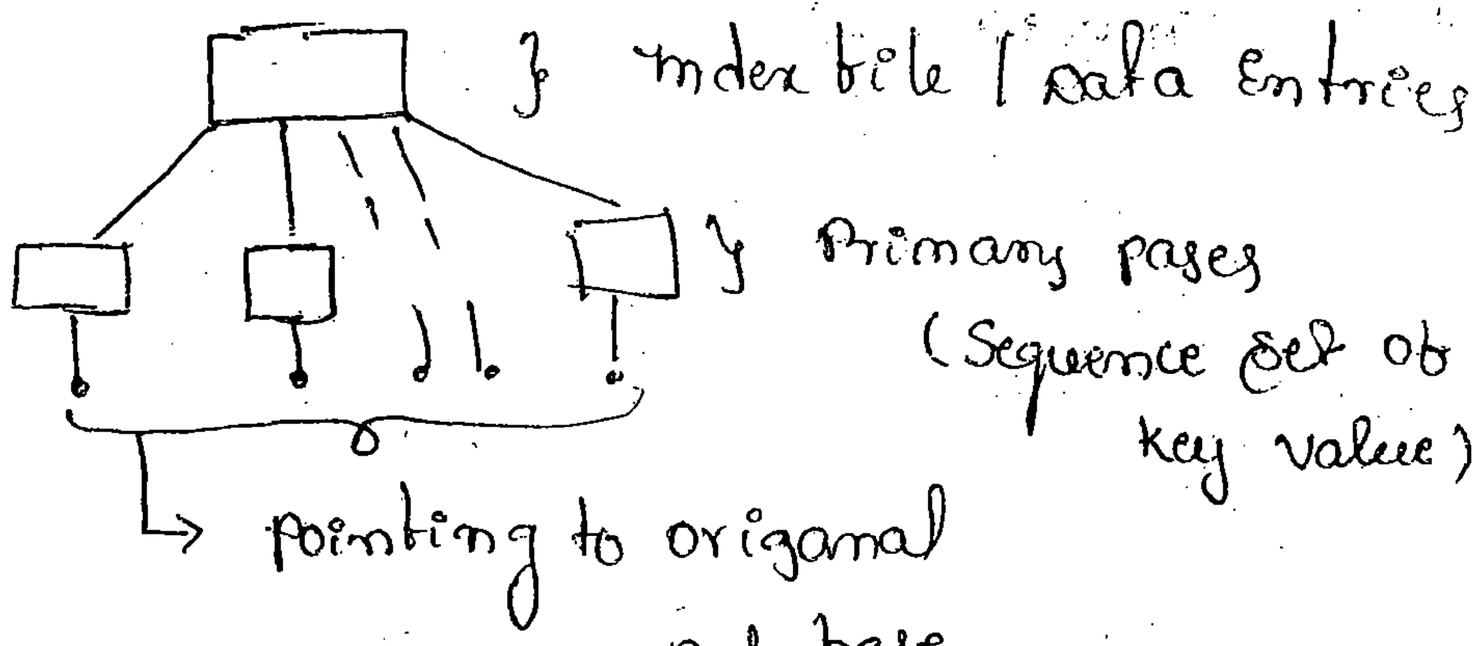
(5) • The General format of leaf / external nodes for representing actual key values.

K_0	K_1	- - -	K_n

when we are representing the key values in the external nodes, They May present in multiple external nodes, and they doesn't contain any Duplicates.

• The Keys which are available in leaf nodes are pointing to actual records in the database.

• The architectural Diagram of B^+ Tree is shown below.



characteristics of B^+ Tree:

- ALL the leaf / non leaf pages must be at same level (or) what ever the operations are performed and B^+ Tree Those operations Makes the B^+ Tree as balanced.

- B^+- Tree gives Guarantee of 50% occupancy regarding data Entries.

- In B^+- Trees Searching starts from root (non leaf), visits various non leaf pages (if required) and finally search terminate at leaf page.

Advantages of B^+- Tree:

- No overflow Pages are Maintained for leaf pages. Hence it may minimizes Searching time.

- It is one of the Dynamic indexing structure. Hence it never consider growing of the file and ~~stricki~~ shrinking of the file

- It gives always more occupancy in the leaf pages of B^+ tree (50% above)

- **Formula for Constructing B^+-tree:**

To construct a B^+ tree we need to formulas

→ 1) Root node exception

min. no. of keys = 1,

min. no. of pointers = 2,

General formula:

(a) max. no. of pointers = m (c) min. node pointers = $\lceil \frac{m}{2} \rceil$

(b) max. no. of keys = m-1 (d) min. no. of keys = $\lceil \frac{m-1}{2} \rceil$

Ex: Construct a B^+-tree for the following set of key values.

2, 3, 5, 7, 14, 16, 19, 20, 22, 24, 27, 29, 33, 34, 38, 39.

Assume the no. of keys can fit in one page are 4.

Sol: When we are constructing B^+ tree we must ensure that the data must be in sorted order.

Root node exception: min no. of keys = 1

min no. of pointers = 2

General formula:

max. no. of pointers = 5 min. no. of pointers = $\lceil \frac{5}{2} \rceil = 2$

max no. of keys = 4 min. no. of keys = $\lceil \frac{4}{2} \rceil = 2$

UNIT-IV

Transaction Management

A ~~Transaction~~

A Transaction can be defined as

"Any Execution of user programes in DBMS Environment"

(OR)

"A part of an application is known as Transactions"

(OR)

"List of actions performed on DB objects" (Tables)

(or)

A Series of actions performed on DB objects for performing some opperations"

~~or~~

In DBMS we perform two types of Transactions They are

1. Serial Transactions.
2. concurent Transactions.

(S) A Serial Transaction is one which always executes one by one and it must ensure the consistancy of the database.

(T) concurrent Transactions are those which will execute all at one's simultaneously.

and they must leave the DB in consistance state

③ on database objects we perform various database operations like insertion, deletion, update and selection. These operations makes use of internally read and write operations.

④ If we perform any read transactions and database object O, then it is denoted by $T_R(O)$ ⑤ where as if we perform write transaction and database object O then it is denoted by $T_W(O)$

⑥ In General $T_R(O)$, $T_W(O)$ is represented by $R(O)$, $W(O)$ respectively.

⑦ In DBMS environment a transaction can have two types of states they are

1) commited
2) Aborted

⑧ A commited Transaction is one in which transaction has executed successfully and the modified data will be written to disk from primary memory.

⑨ Every commited Transaction must have <start> ---- <commit> → If a Transaction is executing and it is complete with <start> ---- <commit> then

DBMS internally calls an operation called
Redo(T_i) → T_i – represents Transaction.

An aborted transaction is one in which Transaction is not successfully completed due to the following reasons.

1. Transaction is aborted because of internal reasons of RDBMS Products.

2. Transaction aborted because of System crashes while the Transaction is in progress.
(power failure)

3. Transaction is aborted because of reading an excepted value.

Programatically some of the (transaction)/ actions are not executing which are enclosed between <Start> ---- <Commit> then that transactions are known as aborted / un-committed Transactions.

Properties of Transactions:

Any Transaction which is executing in DBMS environment must follow certain property for achieving consistency of the database whether we are executing the

Transaction serially or concurrently.

- The Properties of Transaction must also achiving the consistancy of DB from System crashes

Every Transaction contains four Properties They are

1. Atomicity
2. ~~Constim~~ Sen Consistancy
3. Isolation
4. Durability

ACID

Atomicity:

Atomicity says that "any group of transactions to be count execute then execute all the transaction otherwise don't execute at all"

Each every transaction in DBMS must be atomic.

Consistancy: Consistancy DB is nothing but Maintaining correctness of the DB.

When we Develop any Transactions which are treated as independent and Dependent transactions and their executing either Serially or currently those transaction must leave the DB consistance state.

Hence the Goal of Each every transaction in DBMS must achive consistancy of DB.

Isolation:

A DB Programmer is always treated various transactions which are developed in different context treated as isolated transactions.

In DBMS all the isolated transactions must be leave the DB in consistance state by executing either serially or concurrently.

Durability:

When we perform any transaction and it is successfully executed DBMS informs a message to the user regarding completness of the transaction. When the transaction is completed, DBMS will tranbord the temporary result from log to secondary memory.

The process of storing the data permanently with in less time in secondary memory is known as durability i.e Amount of time taken to achive the persistancy is known as Durability.

Transactions and Schedules:

A Transaction is defined as any excution of user defined programmes in DBMS.

- If we run a single transaction n-no of times then DBMS will treat a single transaction as n-no of transactions.

(1) In DBMS Environment, a transaction may be successfull or unsuccessfull.

(i) Successfull transaction are known as commit. Transaction where as

(ii) unsuccessfull transaction are known as aborted or uncommited transaction.

(2) Any transaction which is performing read operation on DB object 'O' is denoted as R(O) or $T_R(O)$.

(3) Any transaction which is performing write operation on database object 'O' and it is denoted as W(O) or $T_W(O)$.

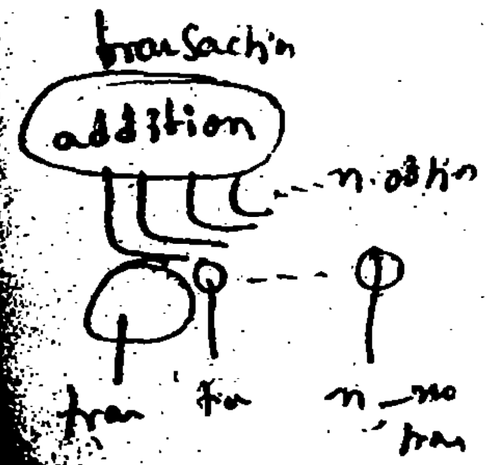
→ (4) "A schedule is defined as collection of transaction" (or)

Serial of transactions grouped in a single unit and this single unit is known as schedule".

In General DBMS express the schedule in the following ways

T_1	T_2
R(A)	
W(A)	
	R(B)
	W(B)
	R(A)
	R(B)
R(B)	
W(B)	
	Commit
Commit	

- t_1 and t_2 are collectively known schedule
- Must exchanges their messages through objects



(5) In the above transaction t_1 and t_2 are collective known as Schedule.

(6) The Transactions in the schedule must ~~Execut~~ exchanges their messages Throw database Object.

(7) All the multiple transactions on the schedule will perform their operations on multiple dependent Objects (DB table) but not on independent DB object. If we remove one dependent object from Group of dependent object, the state of the transaction in the schedule will be changed.

(8) If all the transactions in the schedule contains <Start> - - - - <Commit> Then those set of Transactions in the schedule are known as complete scheduled

(9) Transactions. where as if any one of the Transaction in the Schedule Doesnot contain either Start or Commit Then those transactions in the schedule are known as incomplete scheduled transactions.

(10) If all the transactions in the schedule are executing one by one Then those transactions in the schedule are known as Serial scheduled transaction

(11) where as if the transactions in the Scheduled are executing Simultaniously Then those type of transaction in the Schedule are known as concurrent Scheduled transactions.

(12) Hence all the transactions in the schedule make execute serially or execute concurrently finally all those transactions in the schedule must leave the DB in consistancy state.

Note:
when the transaction is in progress, because of various reasons the control of DBMS changes to some other new transaction by leaving currently executing transaction. The old transaction is said to be interrelated transactions

Sai ✓

Concurrent Execution of transaction:

① Any application we develop in DBMS Environment, there are by default executing serially which leads to more amount of time for executing.

② when we write any user programer and submitted to a system, it internally performs transferring the page (which contains the record) from secondary memory to primary memory this operation can be done by input device.

- At the time of getting the page from S.m to p.m the other device like cpu sits idle. It is recommended in DBMS, when one IO operation is performing, cpu also must perform other activities like processing the records. If both the devices are functioning simultaneously. which increases system throw put (with in a no. of jobs completed in a given amount of time)

when once system throwput is increase the response time will be reduced.

- This two factors leads in DBMS to execute the transactions concurrently.

Serializability:

"It is the Process of executing all the transactions serially and who's net effect leave the database in consistant state."

- In Serializability Process transactions are executing one by one and DBMS Environment may take more amount of time for transforing Primary memory Data (block Data) to secondary memory.
- In other words Serializability takes more amount of responce time and System Throuput also less.

- ex: Serializable Schedule

T_1	T_2
R(A)	
W(A)	
	R(B)
	W(B)
	R(A)
	W(A)
R(B)	
W(B)	
Commit	

In the above Serializable Schedule, transactions t_1 and t_2 are executing one by one and who's responce time is less.

In DBMS) Environment when multiple transactions are executing concurrently there is a possibility of leaving the database in inconsistance state. To avoid this inconsistency we must use the concept of two phase locking Protocol (or) strict 2PL.

Two phase locking Protocol:

Ex: Let us assume their exist a DB object account with a column balance and whose intial value is zero. Their exist two transactions T_1 and T_2 and they want to update balance column with 10 and 20 respectively and both the transactions want to execute at the same time.

- when two transactions T_1 and T_2 excutes at the same time and after completion of their excution the final value in the balance column is either 10 or 20. But not 30. which leads to inconsistant result of the DB.
- In order to get the consistant Result 30 we must use 2PL.
- The Protocol 2PL contains some set of Rules for achiving consistancy of the database.

Rule 1: If a transaction want to read (respectively modified) Then The Transaction gets shared lock (respectively Exclusive lock) on the database object.

- when The transaction is in Progress, DBMS Environment will decide the types of locks required by the transaction and They will be granted.

what ever the locks which are granted on a DB object to a particular transaction, who's details are maintained by locking manager of DBMS.

Rule 2: If the transaction completes it's Excution Then The Transaction will Release all the locks which are held with transaction.

- If a transaction want to read The Data from DB object, It has to obtained shared lock and It is generally denoted by $S(O)$ or $S_T(O)$.

on one DB object we can obtain any no. of shared locks.

- If any transaction want to write (modify) the data to the DB object Then the transaction has to obtain exclusive lock on DB object which is denoted by $X(O)$ or $X_T(O)$. on one particular DB object, at a particular point of time

one Exclusive lock can be obtained that is multiple transaction can't obtained exclusive lock on the same Database object.

Transaction Support in SQL:

Any execution of user programme in DBMS Environment is known as transaction. In DBMS we have 6 levels of transactions which are supported in SQL.

SQL is a common approach for various DB products.

- selections
- insertion
- updations
  - updation
  - Grant
  - Revoke.

Selection: This Transaction level is used for retrieving the Data from DBby specifying some condition if required.

In SQL, to select records from the DB we use a command called select.

Syntax for Select Command:

select <column list> from <table list> where <condition list>;

Deletion: This operation is used for deleting the record from the database.

In SQL, to delete the record we use a command called Delete.

Syntax: delete
from <table list> where <condition list>;

Ex: Delete from student where marks ≤50;

- Delete the records whose marks less than are equal to 50.

Updation: This operation is used for updating or modifying the records in the DB. In SQL, to update the records in the DB we use a command called update.

Syntax: update <table list> set col_1 = $value_1$, col_2 = $value_2$ ... col_n = val where <condition>

val_1, val_2 -- $value_n$ represents the list of values which we use to update the records in the DB.

Ex: Give 5% Bonus marks to all the students.

update student set marks = marks + (0.5* marks);

- change the name of the student as 'aaa' where sno = 12.

update student set sname = 'aaa' where sid = 12;

Grant: This operation is used for granting the set of previlazers to some other user.

set of previlazers are nothing but select, insert, delete, and update.

Syntax: Grant <set of previlazers> on <tablelist>

to <username 1>, <username 2> --- <username n>;

ex: Grant ~~set of previla~~ select, delete, update on Emp to MCA020;

If no set of previlazers are specified, automatically all the previlazers in the DBMS are granted to the specified user.

Revoke: This operation is used for getting back the set of previlazers which are granted to the other users.

Syntax: Revoke <set of previlazers> on <tablelist>

from ~~to~~ <username 1> <username 2> --- <usernamen>;

ex: Revoke select, delete, update on Emp from MCO20

Once the previlazers are revoked from the specified user, no longer the specified users ~~the set~~ can't use the set of previlazers.

Note: Grant and Revoke Commands Can be applied in Client Server Architecture where an RDBMS products is installed. i.e one Cannot use Grant and Revoke Commands in standalone machions where RDBMS products is installed.

Note: find the Name of the Employee who's drawing Second highest Salary.

select max(sal), ename from emp where Sal < (select max(sal) from emp)

/* obtaint all the Salaries which are ≤ highest sal and applying max(sal) */

Sai

* Crash Recovery:

Crash Recovery is nothing but rollback the aborted transaction to its original state.

When the transaction starts executing, the corresponding record will be transfered into main-memory under the same time for processing the transaction in DBMS environment some amount of memory space is created and this is known as log record. A transaction can be either successfull or unsuccessfull. Successfull transactions are known as Commited transactions.

hen the transaction is commited what ever the changes are performed in log are forced to write to a DB. which deciding in the harddisk. In other words the modifications of log are transfered to disk by performing an operation called redo(T_i)

- when the transaction is aborted, the temporary modifications of log are rolled back i.e who's modifications are neglected. In other words log modifications are rolled back by using undo(T_i).

- when the transaction is aborted the log record is known <u>Dirty read page</u>.

- when Multiple transactions are aborted, it is responsability of recovery Manager to collect all dirty pages handover then to O/S system.

- when group of transactions are started their execution, some of the transactions are successfull and some of them are unsuccessfull.

Identification of successfull transactions & unsuccessfull transactions will be done by <u>transaction Manager</u>. which ever transaction are successfull who's modification will be

commited to original database by performing Redo (T_i) where as which our transactions are aborted They are rolledback by recovery manager by calling undo (T_i). (This will be in the case of independent Transactions).

when N-no of Sheduled transactions are executed in which some of the transactions may be successfull and some of them are unsuccessfull. in this context the transactions in the sheduled are rolled back by performing undo(T_i). If all the transactions in the sheduled are successfully executed then the modification of transactions will be commited to the original Database. by performing redo(T_i).

Introduction to ARIES:

The basic aim of ARIES is to recovery the System from crash. This Process contain 3 phases They are

1. Analasis
2. Redo
3. undo

Analasis: when group of transactions started executing, some of the transactions may be successfull or may not be successfull. Analasis phase makes to understand identification of which transactions are successfull and which of them

are unsuccessfull.

- Analysis phase is performed by transaction manager.

Redo: In analysis phase which ever transactions are identified as successfull transactions, who's tempo-rary modifications available in log, are transfered to Hard Disk by performing redo.

Undo: In Analysis phase which ever transactions are identified as un-successfull transactions (aborted), those transactions modifications are rolled backe by performing undo (Ti) in this phase, the Dirty read pages are handover to O.S by the recovery manager.

Security in Transaction Management:

- Security is an important principle in DBMS to protect the confidential information from unauthorised users.

- In DBMS Security can be available at 4 level. They are
  1. OS level
  2. product level
  3. logical level
  4. human level

OS level Security:

when we want use any DB product representing the data permanently. It is mandatory for us to have an operating system.

Before using any database product, some level of Security must provided by the O.S (Windows NT, XP, Unix, etc)

Product level Security:

to create any database according to the organisational requirements, one must use a popular Database product which will provide product level Security in the form of usernames and passwords.

Some of the Databases like Dbase, Dbase III plus, foxpro, MSaccess are not able to provide usernames and passwords. Hence these Products are very weak in Security. Hence industry never uses.

The popular DB Products like oracle, Sql Server, DB2 .. etc are providing effective Security in the form of usernames and passwords.

Logical Level Security:

logical level Security is one which must be developed by a database Programmer for validating Database users.

most of the popular DB products provide a facility to the programmers to develop their Secur

Logical:

The above 3 levels of Securities are available at nation level.

Human level Security:

The above 3 levels of Securities can't Protect the confedential information from Theft, fire, etc to protect These Kind of Problems we must have an Effective human level Security.

30/07/

UNIT - V

Introduction to security:

In Information technology security is one of the important principal used in various applications. for protecting the confidential information from un authorised users.

In Information technology Data can be stored permanately in two ways. They are.

1) Through - files
2) Through - DataBase.

If we store the data permanatly in the form of files, files never provide any security to the Data. Hence Industry never uses files to store the data permanately.

Industry always stores the data permanately in the form of DataBase. Since most of the popular DataBase products (like Oracle, SQL server, DB2 etc.) provides 100% security to the Data in the form of Usernames and passwords. (product level security)

When we are designing any DataBase Application, The DataBase programmer must maintain the following objectives.

1) secracy: " confidential information should not be disclosed to unauthorized users".

For Examaple the Grades of the student can be viewed by Apartical student only. But not

by another student. Similarly the PIN No of any Bank customer can be used by themself but not by others.

Hence each and every database application must contain secrecy of Information in the form of user names and passwords.

Integrity:- "The processes of maintaining correct nes of the data" is known as Integrity.

For eg:- The Banking operations, various employees are responsible for ~~form~~ performing these operations on a particular day. One employee of one responsibility is interfering in another responsibility of other employee, then is a possibility of losing integrity.

Student are allowed to view Grades of various students but they should not be allowed for updating the Grades. If so Then we loose Integrity.

Availability:- " The process of Giving the DataBase Data to all users is known as availability".

Eg:- PIN NO of one Customer can not be changed by other customers. Grades of the students are available to every student but Grades of the student can not be changed by students. But they can be changed by authorised persons like faculty.

Access Controls:-

Once we develop a database, the Database Data must be provided to authorized users only.

In DBMS we have two types of access controls

1) mandatory access controls (Bell-Lapadula module)
2) Discremanary Access controls

mandatory Access controls:-

To provide control about the Database Access, they are some loop holes Discremanary Access controls. To avoid those loop holes to use mandatory Access controls.

mandatory Access control discribed by a model called Bell-Lapadula module. This module uses the following terms to discribe mandatory Access controls.

1) Database Objects (tables, views, clusters etc). Database objects are denoted by 'O'.

2) Subjects:- (users programs, Queries, transcation, schedules) subjects are denoted by 'S'.

3) Security classes:- for each every database object and subjects we must assign some security class.

In DBMS we have the following security classes - they are. a) Top secrete (TS).

b) secrete (S)

c) Confidential (C)

d) Unclassified (U)

The order of maintaining the security for DataBase objects, subjects is that TS > S > C > U.

7) Clearence:-

This is subjected to clearing Read and write operations on Data Base Objects. with respective subjects.

Bell-Lapadule model provides two restrictions for reading and writing from | to data base objects 'O' with subject 'S'.

1) Simple security principal:-

If security class(S) ≥ security class(O) then
class(S) ≥ class(O)
the subject S can read the Data Base Object 'O'

Otherwise.

subject 'S' cannot read the Data from Data Base object 'O'

Ex:- A Top secrete subject (TS(S)) can read the data from Top secrete database object, secrete, confidensial, unclassifide DataBase object (i.e)

TS(S) ≥ TS(O)
TS(S) ≥ S(O)
TS(S) ≥ C(O)
TS(S) ≥ U(O)

then A secrete subject (S(S)) cannot read the data from Top secrete DataBase object (TS(O)).

* - Principal:-

class(s) ≤ class(o)

If security class ('s') ≤ security class(O)
Then a subject 's' can write the data on DataBase object 'O'.
Otherwise subject 's' can't write the data on DataBase object 'O'.

Ex:- Unclassified subject (US) can write the data to unclassified, confidential, security, Top security DataBase objects But A Top secreate subject (TS(S)) cannot write the data to security class of DataBase object.

Discremany Access controls:-

These controls provides ~~confidential~~ security to the database by using two types of controls which are available in various RDBS products. They are

1) Grant
2) Revoke.

Grant:- Grant is one of the discrenary Access control which will Grant the set of privilizes to other users. In DBMS set of privilizes represents the following

1) selecting the records from the DataBase. (select)
2) Inserting the records in the DataBase (Insert)
3) Deleting the records from the DataBase (Delete)
4) Updating the records of the DataBase (Update).

Syntax for Grant

grant < set of privilige on Database object> to <Username>
<Username> . . . . <Username>

set of privileges represents either select or insert or update (or) delete (or) All. ~~Data~~

Database objects represents table names, Usernames, Usernamen --- Username represents set of users who would like to use the set privileges on set of database objects.

Eg:- Grant select, delete, update on emp. to MCA01, MCA02;

Once we are Grant Access control for Granting set of privileges to some other users, then users make create a table with Data of User of Granted privileges. This is considered as one of the loophole of Grant option.

Ex:- Create table EmpDup as (select empno, ename, sal from emp);

EmpDup is the table which is created by some other user which contains complete data of emp table ~~Granted~~ Granted the privileges.

Revoke:- This is one of the Discretionary Access control which will take back the privileges which are Granted with Grant option.

Syntax of Revoke:-

Revoke <set of previleges> on <database object> from <U n-1> <U n-2> --- <U n-n>

set of previleges represent either select (or) delete (or) update (or) Insert (or) All.

DataBase objects represent the name of database/table

U(n-1), U n-2 --- <U n-n> represents set of users from whom the set of privileges are Revoked which are Granted by Grant option.

Eg:- Revoke select, Delete, Insert on Emp from mera, mira.

The limitation of Revoke option is that if a database programmer who Granted set of privileges are not Revoking back completly from the DataBase users then their is a possibility of loosing security for the Data which is present in the DataBase.

To avoid above loopholes of discretionary Access Controls we use mandatory ~~Access~~ Access Control.

<u>Additional issues related to security</u>:- 6

In DBMS environment, the Data which is present in DataBase is confidential that is DataBase Data must be protected from unauthorised users. By providing usernames names and passwords.

In DataBase environment ~~A person~~ called DataBase Administrator who is responsible for protecting the confidential information of various DataBase users by providing Usernames and passwords.

Along with the major roles of DBA security is one of the major responsibility to be provided all the DataBase users.

<u>Ex:-</u> In oracle their is a common Username and password i.e scott and tiger. Due to this the DataBase programmers may not be protecting their confidential Information from another DataBase users. Hence In DataBase environment common User names and passwords are not recommended. Therefore DataBase Administrator must Create a separate Usernames & passwords for

all the Data Base programmes. Due to this we can achieve effective security for the Database Data

Note:- The process of assigning a security class to the database object is known as poly Instantiation.

Check pointing:-

This is one of the machanisam to avoid Crash Recovery of the transcations. Check point is one of the small Book-keeping objects which makes to understand upto what level of transcation is completed and without how much transcation is not executed (ie) which ever transcations ~~point~~ part is completed, DBMS environment maintains a check point and for the rest of the transcation no check point will be maintain. When the transcation is in progress, DBMS environment maintaince for a check point for a regular Theoritical intervals of time for the completed part. When the transcation crashed and it is restarted further, DBMS environment executes same transcation after the check point onwards. (ie) If the check point is not maintained, the same transcation starts executing from beging onwards.

S.No	Data type	size (in bytes)	Range	Accuracy
1	float	4	+ve to -(32)	8
2	double	8	+ve to -(32)	16

The category contains two types of datatypes and their are given in the ~~follow~~ above table

In order to treate Real constant values as a float value that Real constant value must be followed by a letter f (10.75f) but

where as if a real constant value is not followed by a letter 'f' then that Real constant will be treated as higher category datatype in float category. (ie) double datatype.

10.75 treated as double datatype

10.75f treated as float datatype.

9/10/09

Media Recovery:-

In larger organizations, when transcations are executing, internally the result of the transcation will be taken as a Backup for avoiding server crashes. of a particular organization

The processes of taking Backup for the on-going transcations in a separate file for a periodical regular intervals of time is known as media Recovery.

media Recovery used in Real time to avoid sudden server crashes.

In most of the Real time organizations if any natural disaster (like floods, fire, etc) occures there is a possibility of losing the data to get the data which lost, organization will use media Recovery to procede further transcations by using old data of the organization.

Backups can be either off-line or On-line. In off-line Backup, in DBMS, it maintance only committed transcation details. But not uncommitted transcation details for a regular periodical intervals of time.

On-line Back up is one in which the Back up of each and every transcation will be maintained for each and every second without considering either it is committed or aborted.

Hence media recovery helps to recover the server from its sudden crash either by using on-line or off-line Backup techniques.

In media recovery DBMS uses FUZZY-check pointing for committed transcations.

Write about log:-

A log is nothing but a journal. When a transcation starts executing the appropriate records will be transford from Disk into main memory and it is stored in a separate place called private Area.

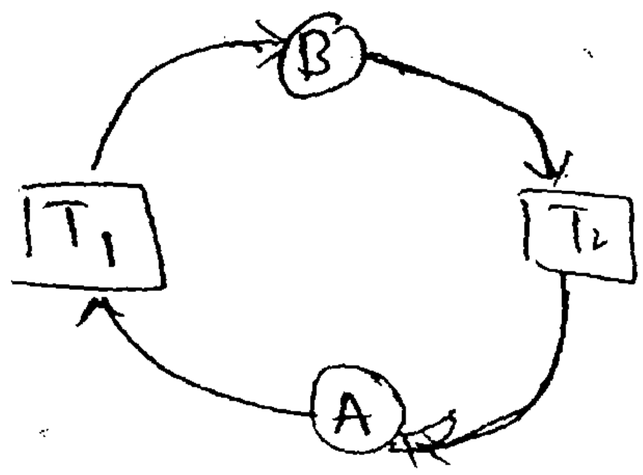
private Area & the main memory is known as log in which transcations modification can taken place. A log may contain committed transcations or un committed transcations. If a log contains committed transcation whose Details are used to modify the original Data Base. By using using Redo(T_i). If a log contains un committed transcations then whose details are cleared / Rolled back or undo of T_i. A log is temparary memory space for performing various Operations on the records.

Dead Lock: If two transcations (say T_i and T_j) started for executing, transcation T_i obtains an exclusive lock on Database object 'A', Transcation T_j obtains Exclusive lock on database object 'B'. T_i is requesting for Exclusive lock on Database object 'B',

which was already held with transcation T_j. Transcation T_i requesting an Exclusive lock on Database object A, which already held with transcation T_i. Hence transcations T_i and T_j are waiting for the Database Objects [B and A] respectively.

Hence T_i and T_j cannot execute their transcations, this model is known as <u>Dead Lock</u>.

In DBMS Dead-Lock's can be expressed in a graph called <u>wait-for</u> Graph. The <u>wait-for</u> Graph for the above problem is



In the above Graph transcations T_1 and T_2 are in Dead lock and their transcation execution are not in progress.

<u>Dead Lock Prevention:-</u>

When a Deadlock occure we use the concept of Deadlock prevention to Break the Dead Lock. Dead lock prevention uses the following principles/techniques to avoid the Deadlock.

Let us assume two transcation T_i and T_j started executing concurrently.

Wait - Die:-

If T_i is having high priority T_j waits, otherwise abort T_j

Wound - wait:-

If T_i is having high priority, ~~Abort~~ T_j, otherwise wait T_i.

In wait and Die approach technique, less priority transcations started executing by stoping higher priority transcations.

In wound and wait technique higher priority transcations started executing by stopping lower priority transcations.

priority of the transcation will be given Based on time stamp. lesser the time ~~stop~~ stamp, higher the priority of the transcation is. Higher the time stamp, lower the priority of the transcation is

① Find the name of the sailors who reserves all the boats

Sol:- Sailors(sid, sname, rating)
boats (Bid, Bname, colour)
Reserves (sid, Bid, date)

Sol:-
1) $r_1 \leftarrow \Pi_{sname, Bname}\left(\sigma_{(Sailors.sid = Reserves.sid) \text{ and } (Reserves.Bid = Boats.Bid)}(Sailor \times boats \times Reserves)\right)$

2) $r_2 \leftarrow \Pi_{Bname}(Boats)$

3) $r_1 \div r_2$

Imp Questions.

UNIT-I :-

① Define the following terms.
a) Entity b) Attributes c) Entity sets d) Relationship
e) Relationship sets

② Explain about E-R model Along with it features and symbols.

③ What is Relational model? How do you convert an ER model in to Relational model.

④ List and Discribe Integrity Constraints.?

⑤ Discuss about tables, views and altering tables and views (In SQL)

6) Define the following terms?
a) schema. b) Functional Dependency c) Partial Dependency
d) Transitive Dependency.

7) List and Describe various Normal forms? Long with Suitable examples.

UNIT - II

8) Explain primitives of Relational Algebra and Relational calculus.

9) List and Describe the set of operators available in Relational Algebra.

10) Distinguish b/w Relational Algebra & Relational calculus.

11) Explain the following

a) Dynamic SQL b) Triggers c) Active DataBase.

d) Embedded SQL e) Aggregate operators / Functions

Exercises The Queries:-

UNIT - III

12) What is an Index? Explain the properties of Indexes.

13) Distinguish b/w

a) clustered vs Un-clustered

b) primary vs secondary.

14) Explain about tree structured Indexing (ISAM)

15) Discuss about B+ tree along with its operations.

16) (Do the problems)

16) Explain about Hash Base Indexing?

17) Write a note on a) static Hashing

b) Extendable Hashing

c) Linear Hashing

18) compare and contrast two Extendable Hashing List or Hash

UNIT – IV

19) Explain the concept of transcation? List and discuss properties of transcations.

20) Discuss about schedules?

21) Explain the concept of concurrent execution of transcation

22) Hint: motivation for concurrent Execution, serializability, Anamalies – RW, WR, WW, RR.

23) Discuss about log based concurrence control? (Hint 2PL Deed lock)

24) How do you recover the system from crash Aries (Hint One process)

25) Explain specialized locking techniques Pr 1) Bmd 2

26) Explain the concept of concurrence control without locking (Hint:- optimastic concurrence control, time stamp based concurrence control)

Unit-V

27) Write a note on Data Base security.

28) List and discribe Access Controls. (Hint Discretionary Access, mandatory Access control)

29) Explain about
a) Aries process
b) media Recovery
c) check point

MCA II year I sem	MCA I year II sem
DCN → 24/11/09	AFM 4/11/09
✓ DBMS → 26/11/09	JAVA 6/11/09
ICG → 30/11/09	MIS 9/11/09
✓ OS → 2/12/09	DS 11/11/09
✓ OR → 4/12/09	CD 13/11/09