Part I DBMS - Database Management System

Course Outcome 1 :- Introduction. 3m or 6m or 7m

Databases - conventional file processing - data modelling for a databases - three level architecture - data independence - component of database management system - characteristics - Advantages & disadvantages of a DBMS - roles - Data base administrator - functions & responsibilities - In-memory database.

(1) An introduction to database systems - Bipin C Desai (Free download available)

(2) Principles of database management - James Martin

Evolution

File processing system (History)

drawbacks

data base } (2)

Layers of database software

3 level architecture - (1).

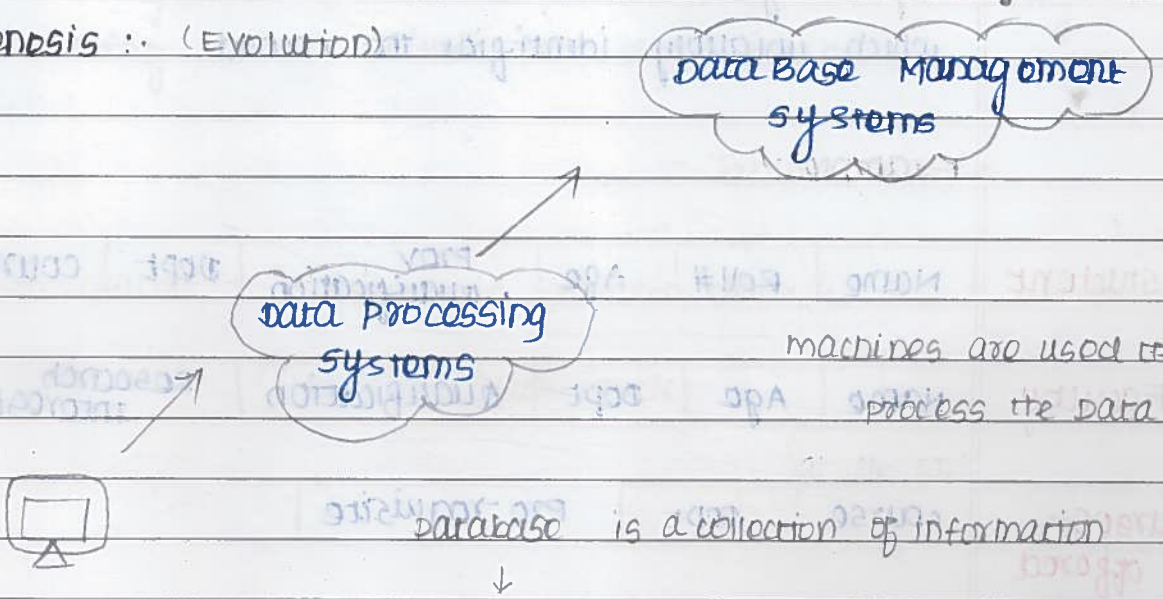
structure of a DBMS

Advantages / disadvantages.

Data Models

Kinds of DBMS

Genesis :- (Evolution)



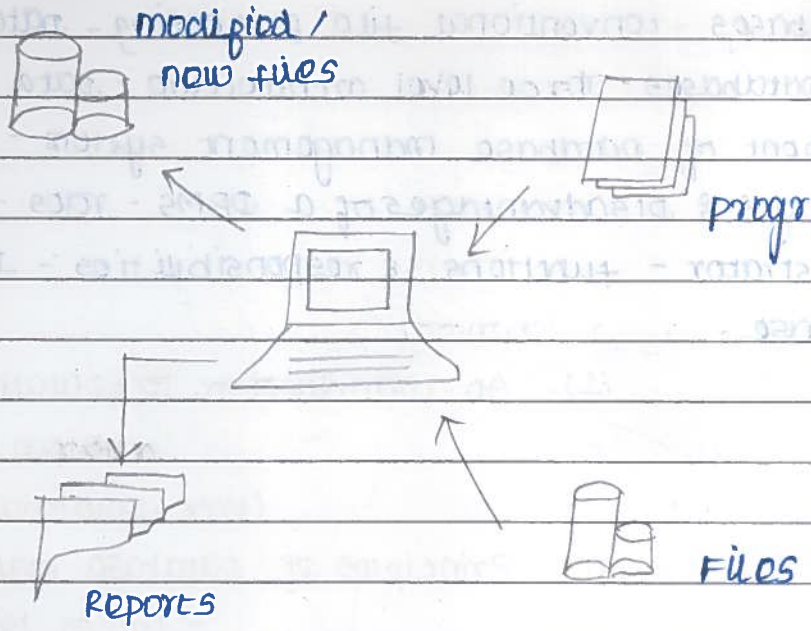
Database is a collection of information

↓

DBMS → software → reservoir of data that manages

conventional
File processing

File Management system.



File - a collection of Records.

Record - a collection of Fields

Field - a collection of characters

- numeric character (123)
- Alphabetic character (MCA)
- Alpha numeric character (22MX222)

(*)

Key - key is an important Field in a record in which uniquely identifies the record.

Example:

Student	Name	Roll#	Age	Prev qualification	Dept	course
Faculty	Name	Age	Dept	Qualification	Research interest	
courses-offered	course	Dept	Pre-requisite			

courses taught	course	Fac_name

template

student record type :

Roll #	Name	course	Address

→ record types

Key ↔ unique identity.

values of the record type is called records.

records:

97MXD1	RAM	BE	CAL
97MXD2	JOHN	BE	CBE

Data Base :-

A collection of data designed to be used by different programmers is called a data base;

A collection of interrelated data stored together with controlled redundancy to serve one or more application in an optical fashion;

Data Independence / the data are stored so that they are independent of programs which use the data;

a common and controlled approach is used in adding new data and modifying and retrieving existing data within the data base.

- James Martin

Interrelated → "relationship between data"

Data Model

Scriby Date __/__/__ Page 06

Demerits of the File Management system :-

Redundancy : A field that repeatedly used in multiple places in the file.

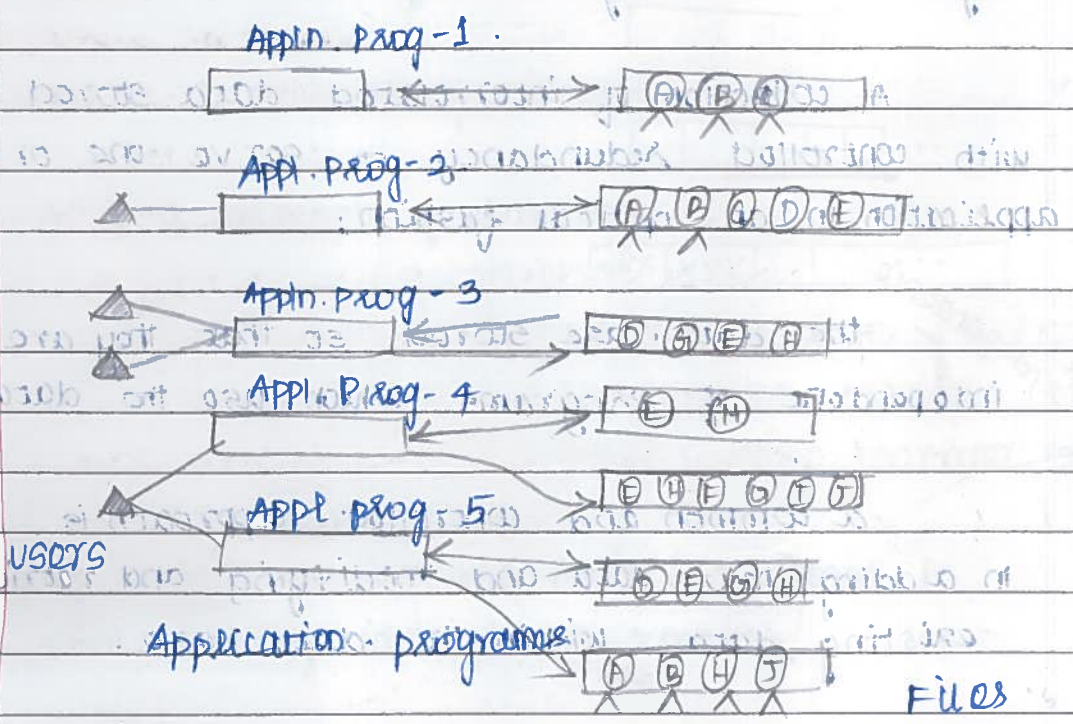
↓

Ex : course field is used in many table that makes redundancy.

Inconsistent : When the changes are not properly carried out in the redundant field, then the system becomes inconsistent.

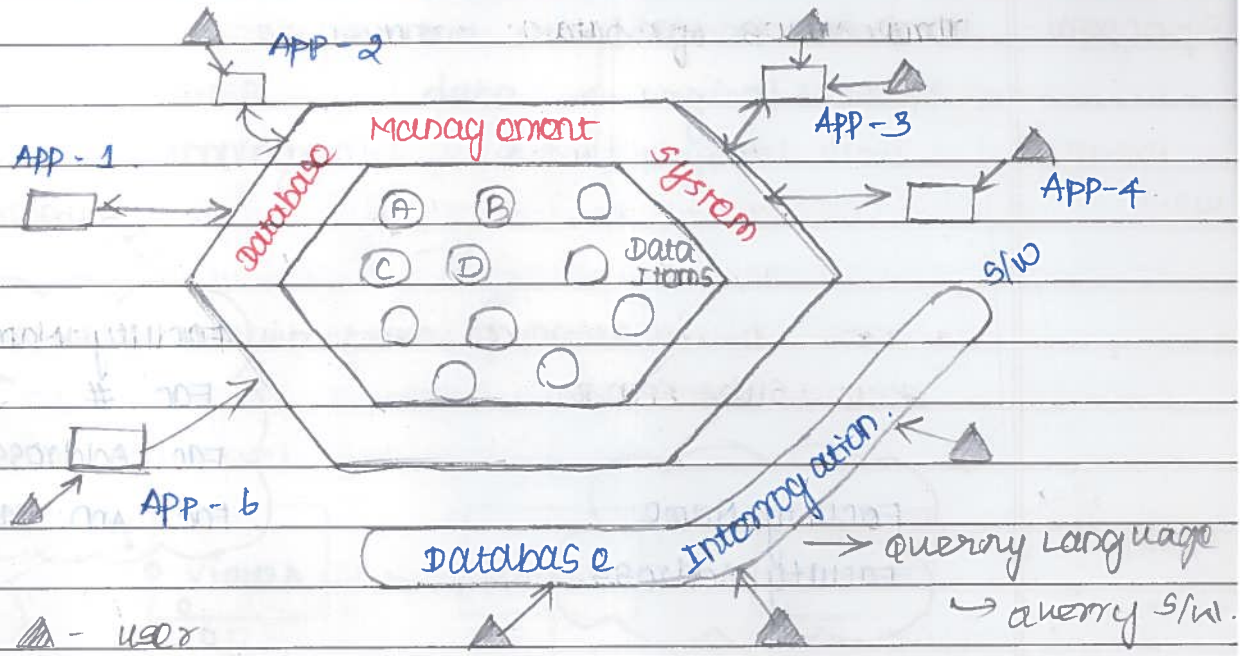
→ Redundancy that leads to inconsistent.

03/10 :



File management system

James Martin



Database management system.

Data interrogation - query.
→ without an application program, we can directly access to the data base with the help of query.

Input - Output security.

Layers of DBMS :

Application programs

Data manipulation Language.

subschema description Language.

subschema

data security.

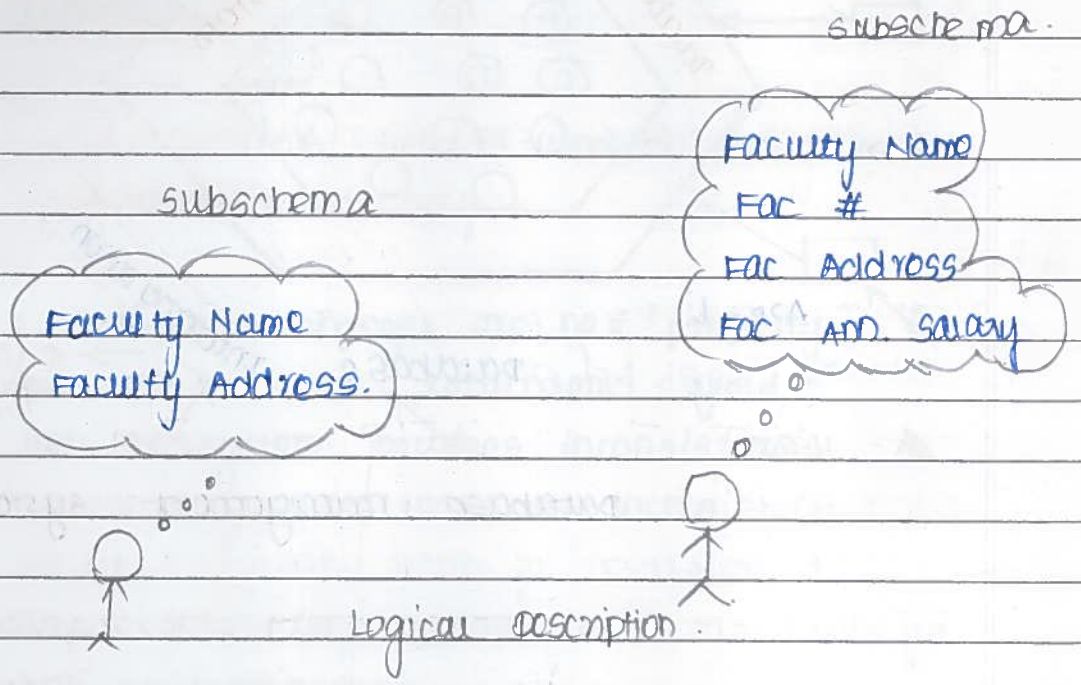
schema description Language.

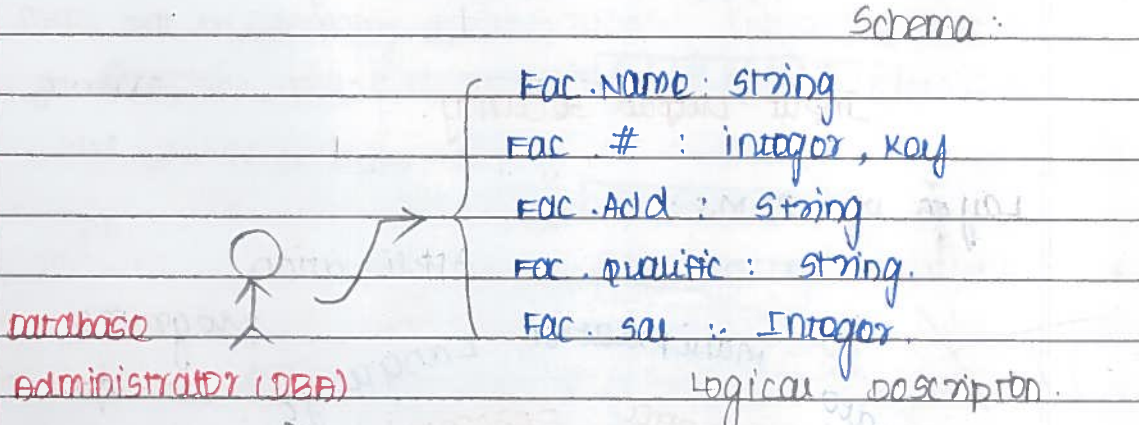
schema

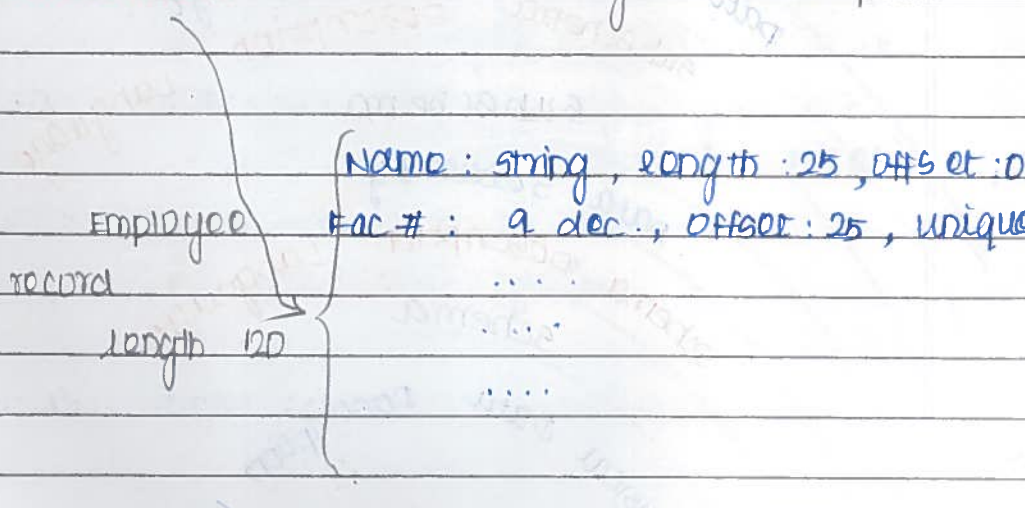
physical Data Description Language

schema - logical description of Data.

Bipin C Desai - Three views of data :-





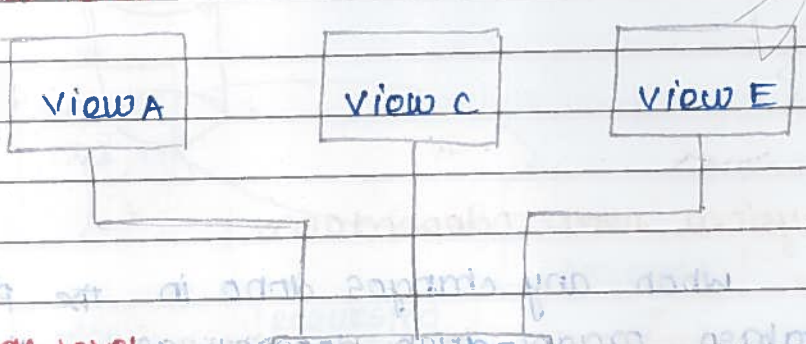


06/10

Database technology } data { subschema, Logical view / schema → global view, physical view

3 Level architecture of DBMS:

External level



Conceptual level

Mapping supplied by DBMS

Internal level

mapping supplied by DBMS / OS

Data independence:

→ Physical Data Independence. When any changes done in the physical DB does not affect the logical DB or Appln Pro

Application Program

Logical database organization

Physical database organization

→ Logical data independence

Appln Prog A | Appln Prog B

Logical database organization.

physical database organization.

Physical data independence :

When any changes done in the physical database organization doesnot affect the logical database organization or Application program.

Logical data independence :

When any changes done in the logical database organization doesnot affect the application Programs and any change in any one of the application program also doesnot affect the other application program.

Database characteristics :

→ Data independence
→ Speedy handling of spontaneous requests
→ Non redundancy
→ versatility in representing relationship between data items
→ Security protection.
→ Real time accessibility. (many cases)

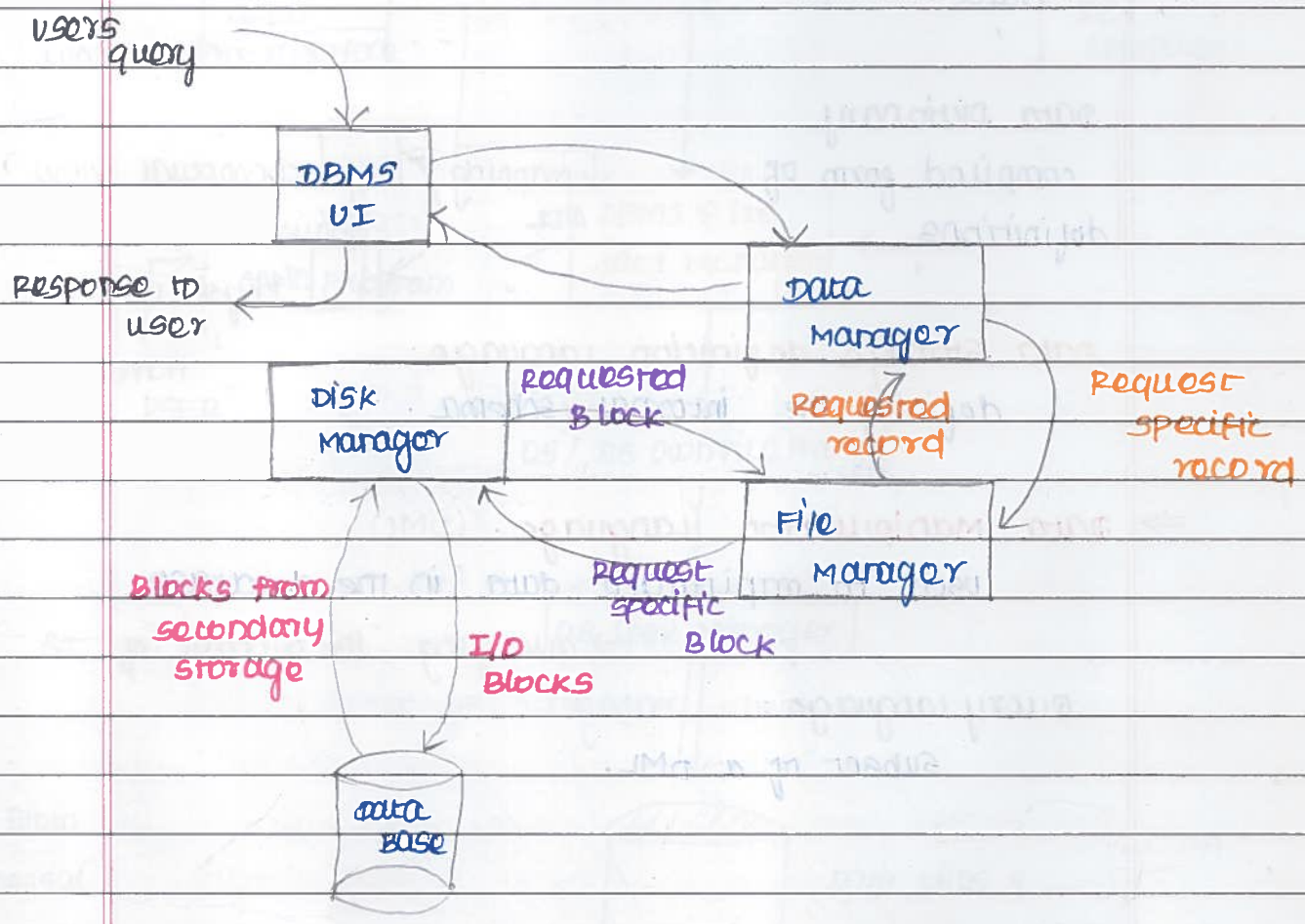
Data processing :

Offline / Batch processing

online processing /

Real time processing - fast response

Steps in data access :

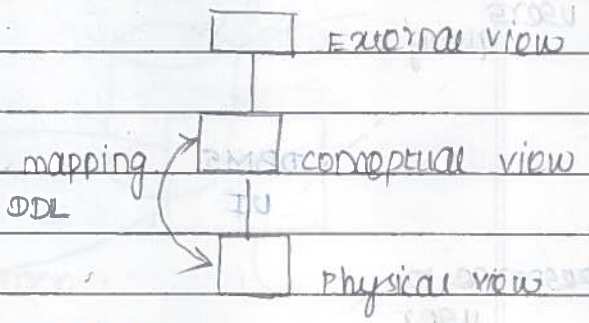


Database Management System Facilities:

⇒ **Data Definition Language:** (DDL)

Used to define the conceptual schema and give details about how to implement the schema in the physical devices used to store the data.

Data Dictionary
compiled form of definitions.



Data Storage definition language:
defining the internal schema.

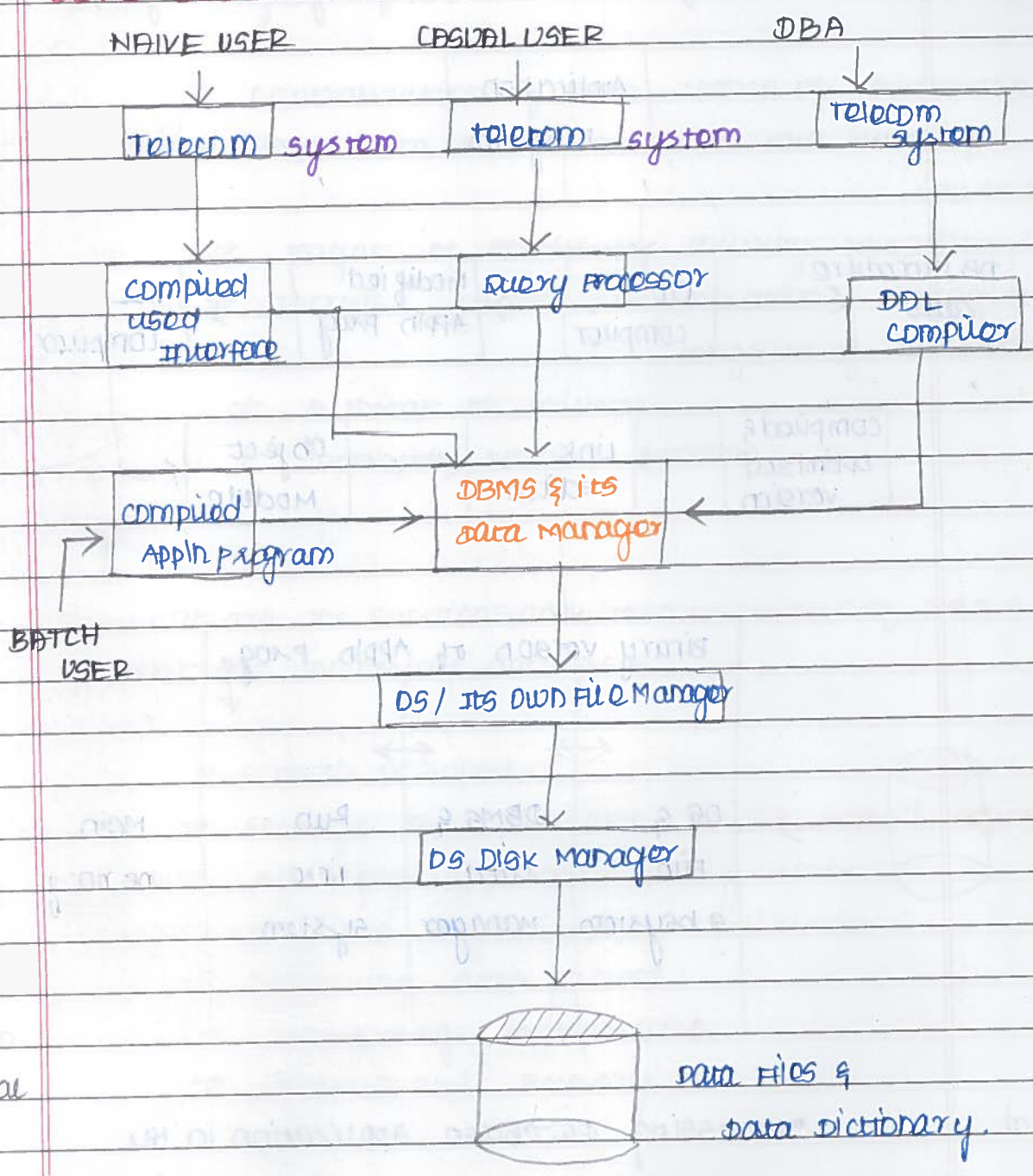
⇒ **Data Manipulation Language.** (DML)

used to manipulate data in the database.

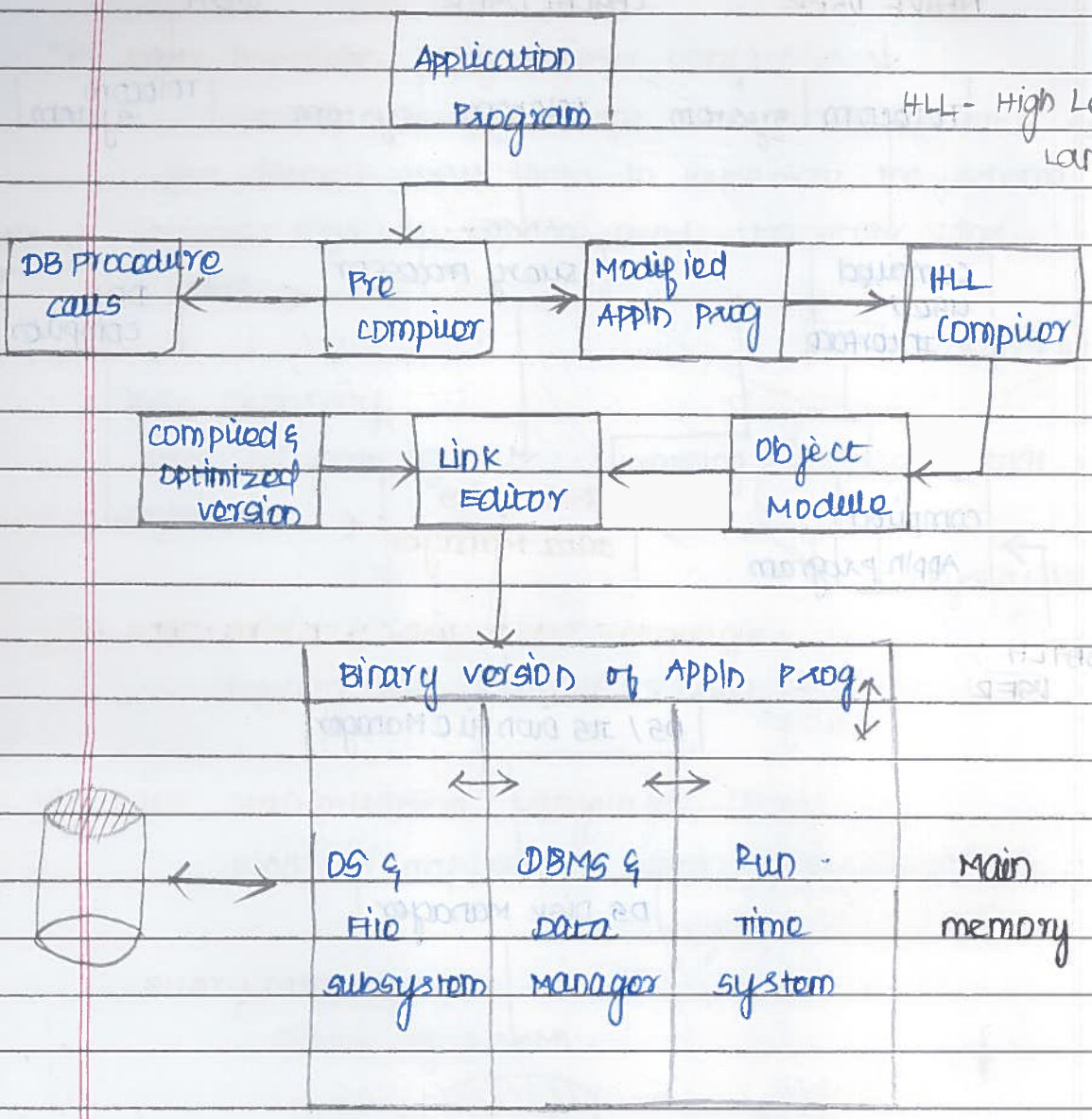
↳ querying the content of DB

Query language:
Subset of a DML.

DBMS USERS :


NAIVE USER
CASUAL USER
DBA
Telecom system
telecom system
Telecom system
compiled used interface
Query processor
DDL compiler
compiled Appln program
DBMS & its data manager
BATCH USER
OS / its own File Manager
OS Disk Manager
Data Files & Data Dictionary.

Structure of a database Management System:



Bipin Dossai

Processing Database Application in HLL

* Advantages of DBMS:
  - → Reduction of Redundancies
  - → shared Data
  - → Integrity - accuracy
  - → Security
  - → conflict Resolution
  - → Data Independence

* Disadvantages of DBMS:

→ centralization of data - sense of ownership lost.
→ Inaccurate data - extensive data integrity / validation.
→ target of ~~sensitivity~~ security breaches
→ Increase political / organizational struggles - conflicts of interest
→ A threat to privacy.
→ Improper design - problems

(i) What are the Functions and responsibilities of DBA?
(ii) What are inmemory databases?

(i) → Schema definition
→ Storage structure and Access Method definition
→ Assisting application Programmers
→ Physical organization modification
→ Approving data Access
→ Monitoring performance
→ Backup and Recovery

(ii)

13/10

Scribd Date __/__/__ Page 17

Part - II

COURSE OUTCOMES : Introduction

Data Models: Hierarchical - Network data model - ER model : Entity Relationship diagram - Data association - Entities, attributes, relationships - structural constraints - Extended ER diagram Generalization - Aggregation - composition - Mapping ER diagram to relations, hierarchical & network models - applications

Bipin Dessai

DATA MODELS :

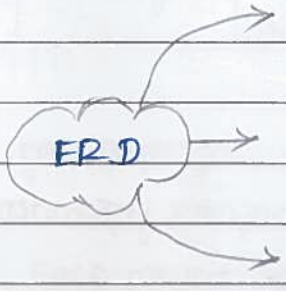
↳ Blueprint / Technical drawing

Data models defines how data is connected to each other and how they are processed and stored inside the system.

A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities.

Entity - Relationship Data Model (ERD)

↳ Foundation



Hierarchical Data Model

Network Data Model

Relational Data Model.

An Entity-Relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship diagram (ER Diagram)

An ER model is a design or blueprint of a database that can later implemented as a database.

ER modelling is based on two concepts:

Entities, defined as tables that hold specific information (data).

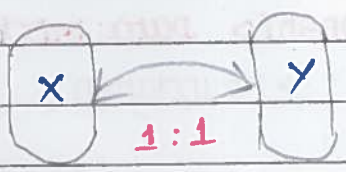
Relationships, defined as the associations or interaction between Entities.

ER Relationships:

(SET)

→ one

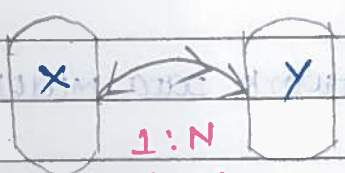
→→ many



One to one

Ex: citizen ↔ Aadhar

M and N are the general alphabets used to denote many.

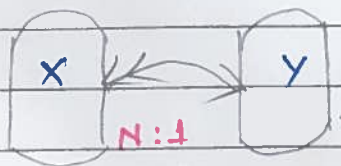


One to many

Ex: student ↔→ course

many to many

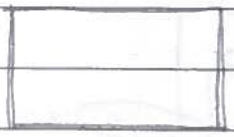
Ex: Hospital ↔→ Lab



many to one

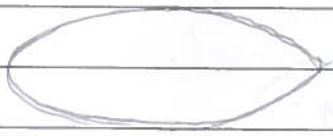
Ex: students ↔ tutor

ER Data Model :



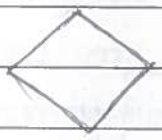
STRONG ENTITY.

Ex : Academics : Student, Faculty.

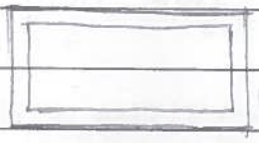


ATTRIBUTE. - characteristics of Entity.

Ex :- Student :- Roll.no, name.



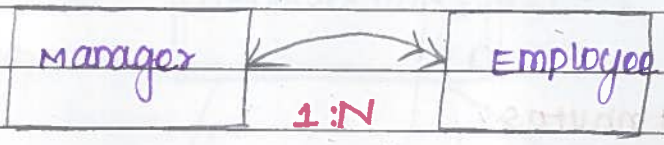
RELATIONSHIP



WEAK ENTITY.

Example :

Company :-



i) One manager can have multiple Employees

ii) Each Employee can only report to one manager.

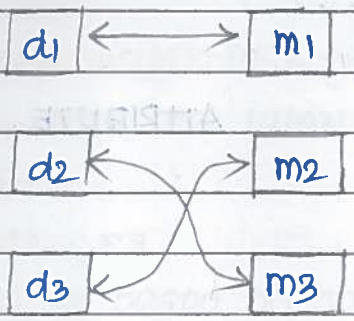
i) One Employee can work in many project

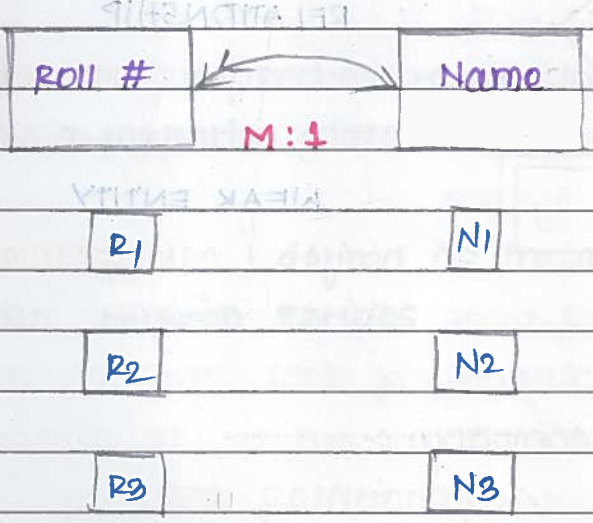
ii) Each project can have multiple employee in it.

14/10

Relationship between Entities:

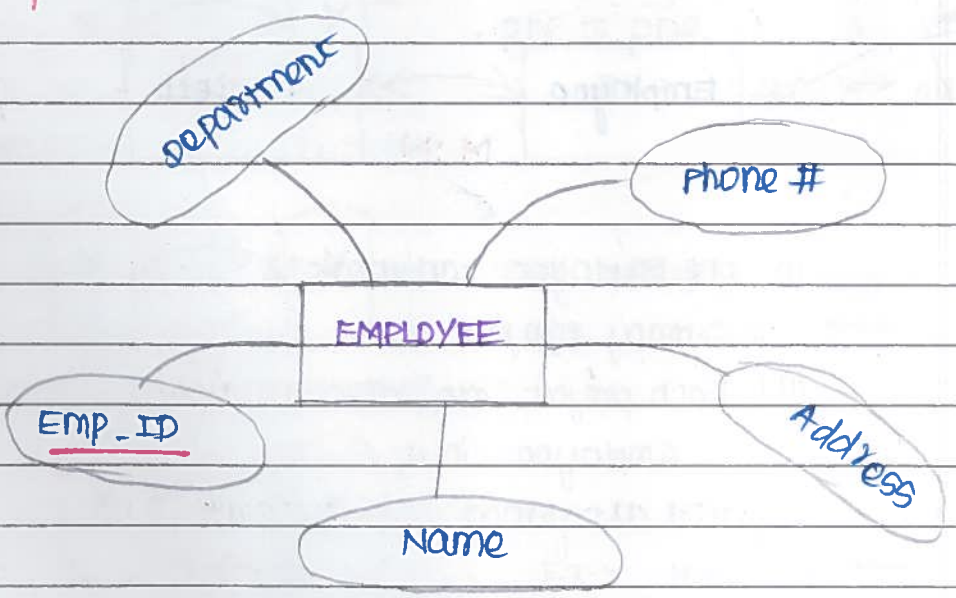
collection of instances





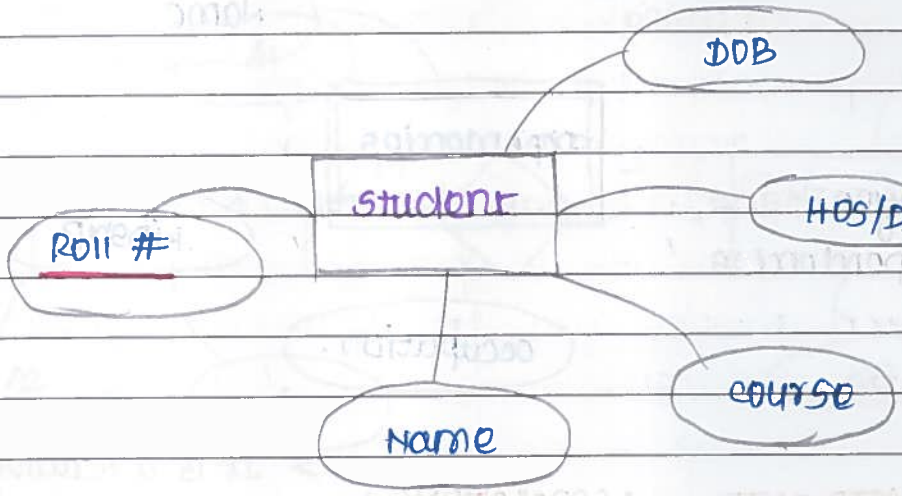
Entity Attributes:

An Entity is described by Attributes.



Employee Entity.

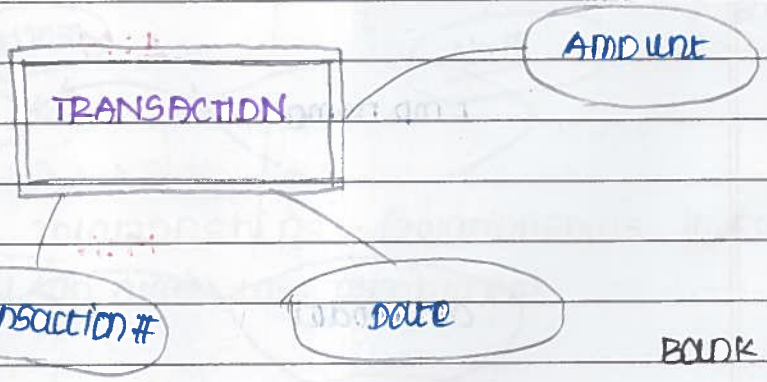
Keys →



Student Entity.

Those attributes name underlined are used to identify the strong attribute is called key.

WEAK ENTITY :



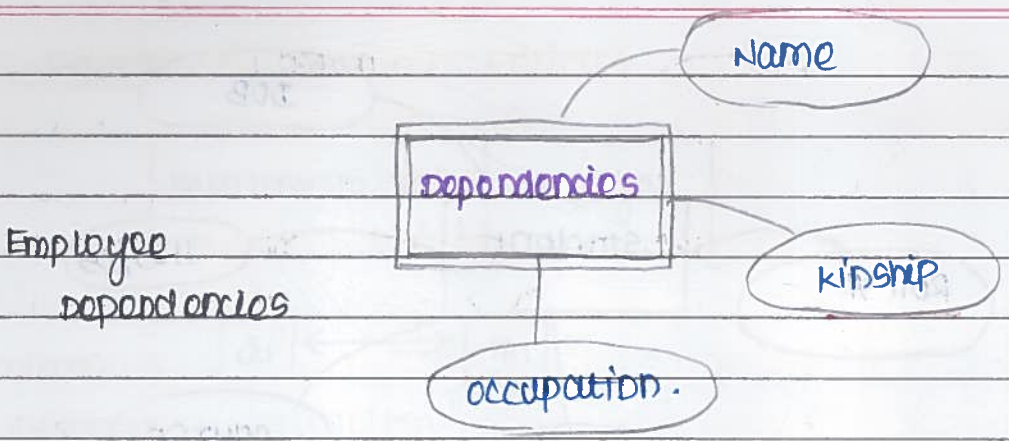
Weak entity does not have any key.

Bank Transactio

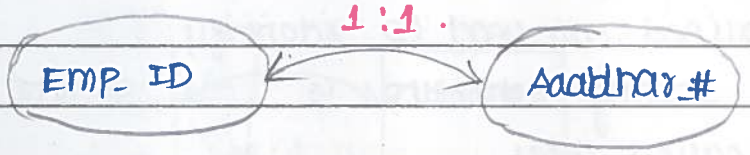
Discriminator :

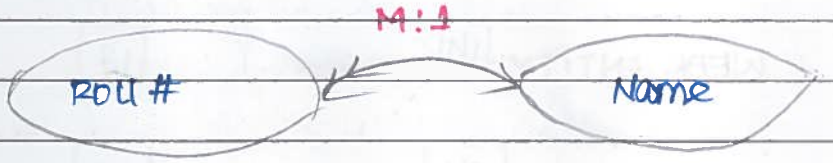
It is a collection of attribute in a weak entity that can describe uniquely the entity.
(It is not strong as the primary key)

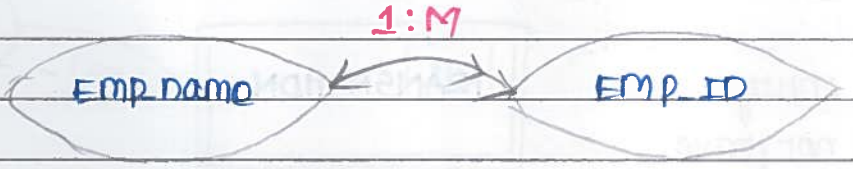
In the above case transaction # and date, together can act as a Discriminator to unique identification.

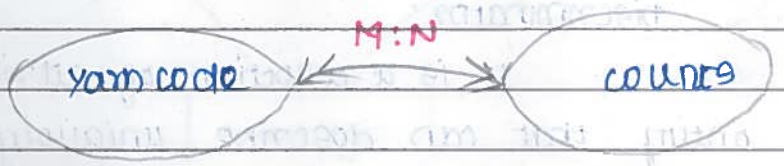


ATTRIBUTE ASSOCIATIONS : → It is a relationship between attributes.



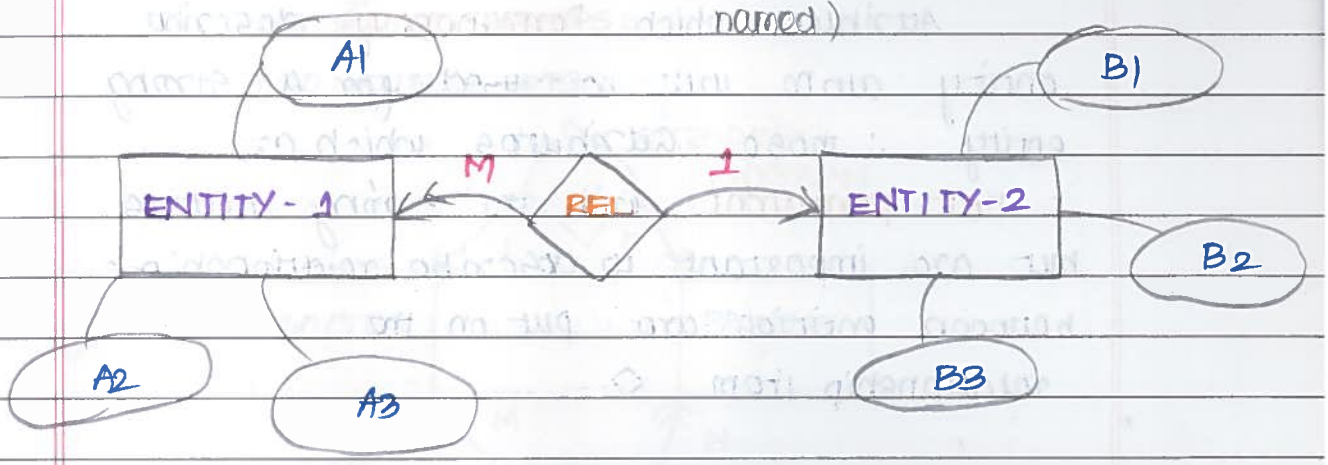






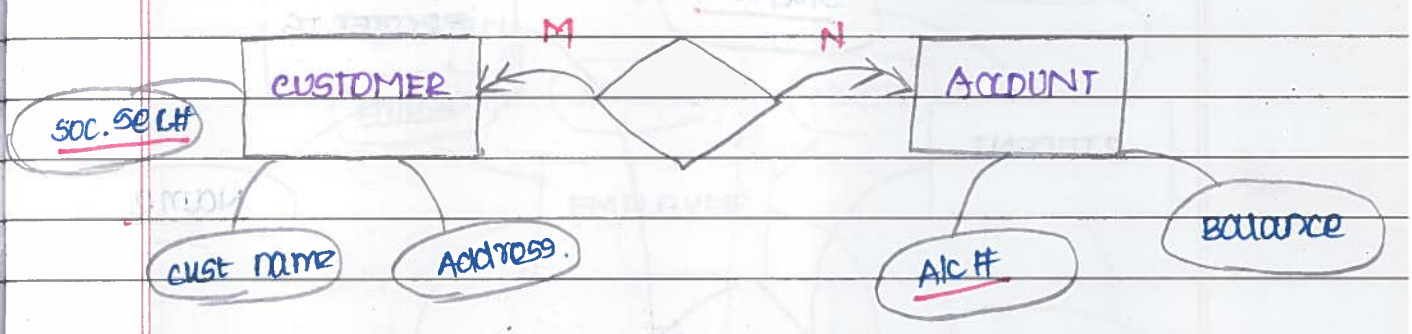
ENTITY - RELATIONSHIPS

(Every Relationship should be named)

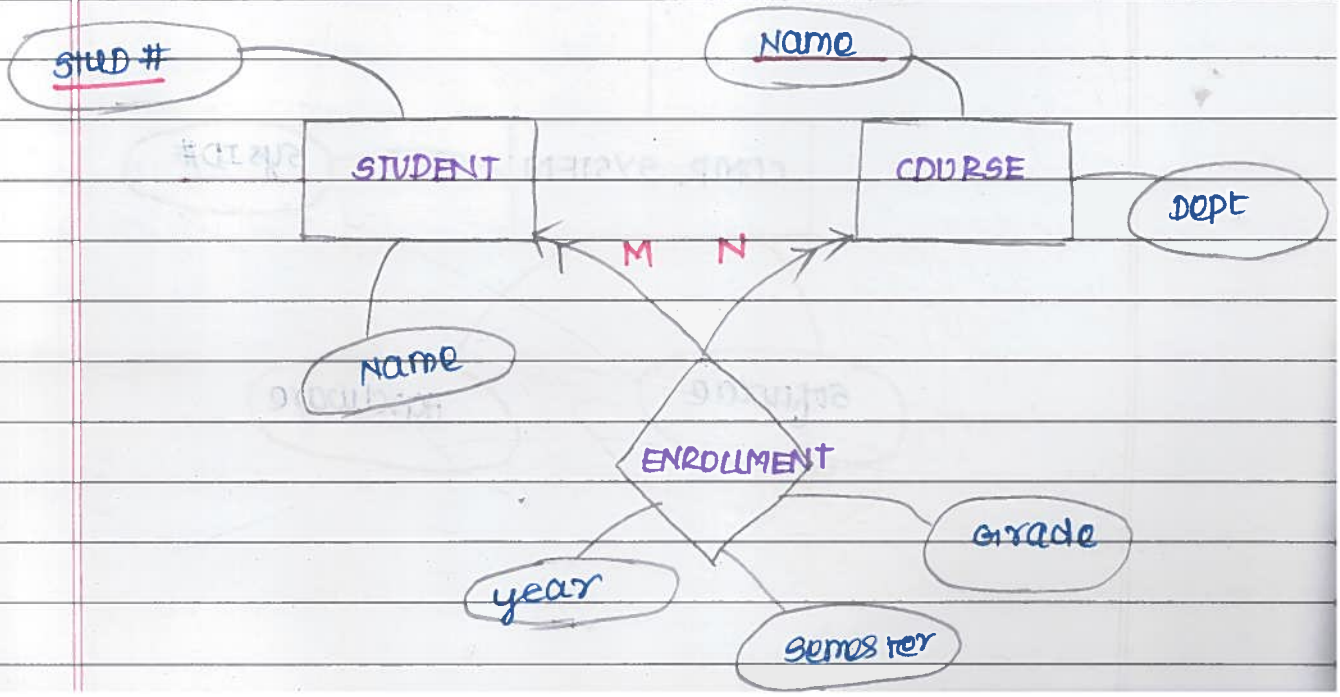


Example :-

Bank Details :-

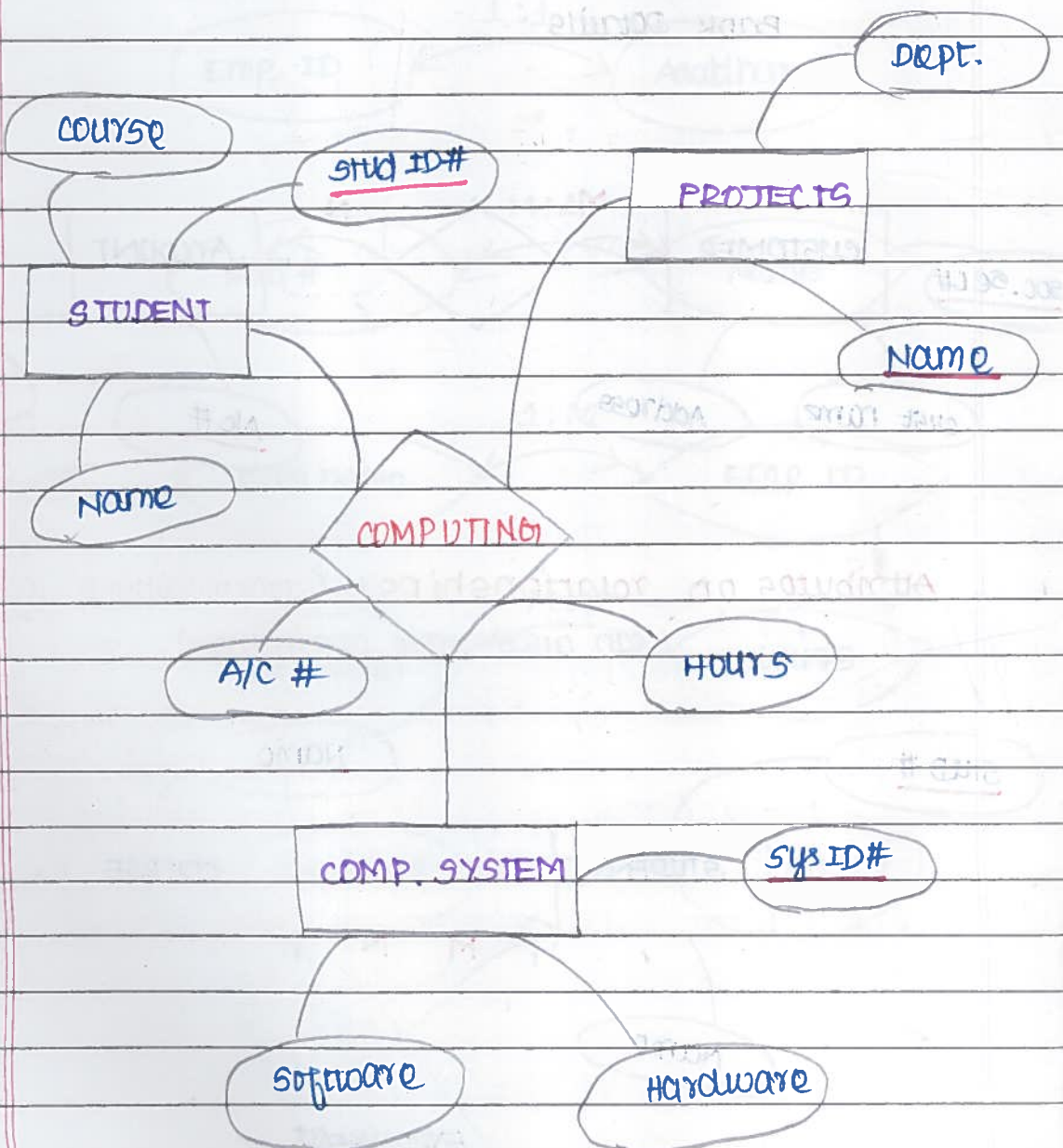


Attributes on relationships :- (relationships in ERD can also have attributes)



Attribute which Permanently describe entity alone will be used for a strong entity. ∴ those attributes which are not permanent with the strong entities but are important to describe relationship between entities are put on the relationship item ◇

TERNARY RELATIONSHIP



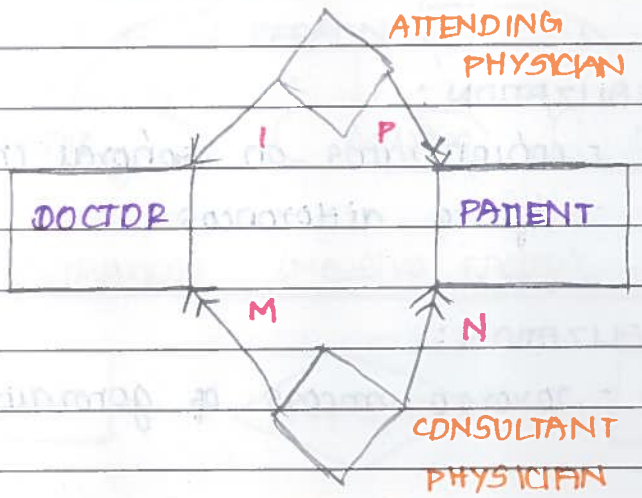
17/10

n-ary relationship.

↳ more than 3 entities involved in a relation

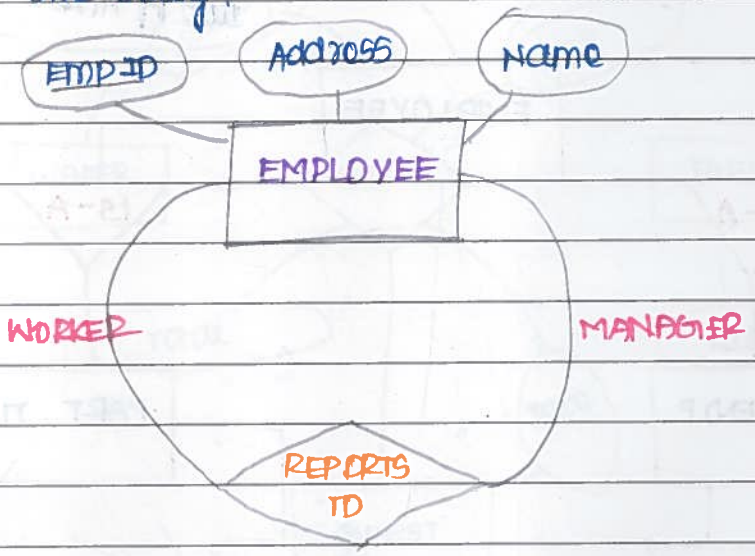
MULTIPLE RELATIONSHIPS :-

- TWO ENTITY SETS



ROLES :- Self kind of relationship.

One Entity.



▽ → Magenta triangle IS-A stands for Abstraction in the ERD.

ABSTRACTION → Extension of ERD.

summarization.

→ GENERALIZATION:

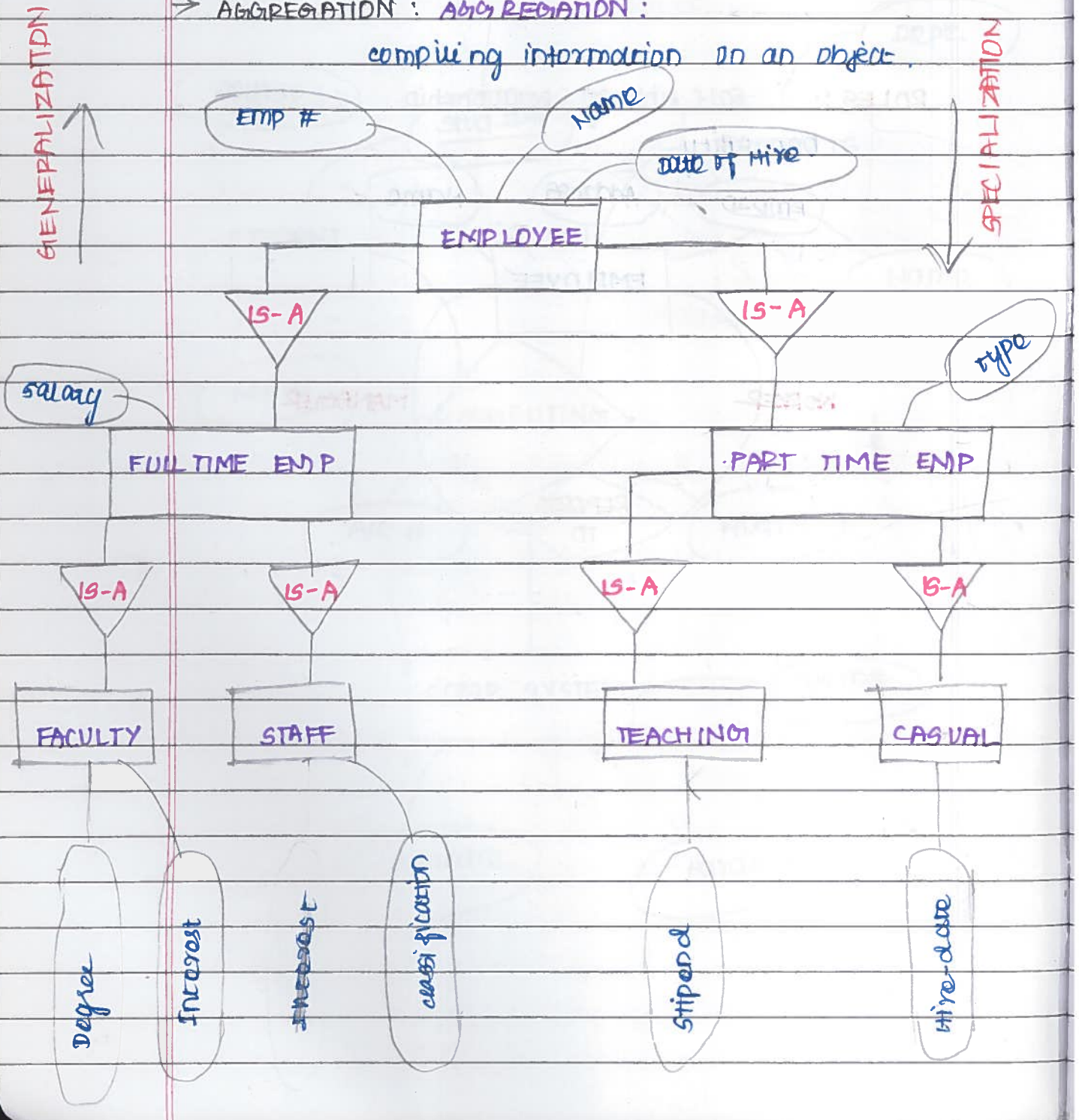
- concentrates on general characteristics
- ignore differences.

→ SPECIALIZATION:

- reverse process of generalization.

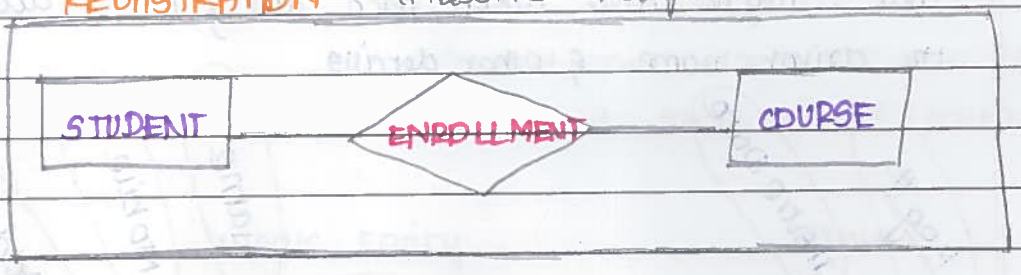
→ AGGREGATION: AGGREGATION:

compiling information in an object.

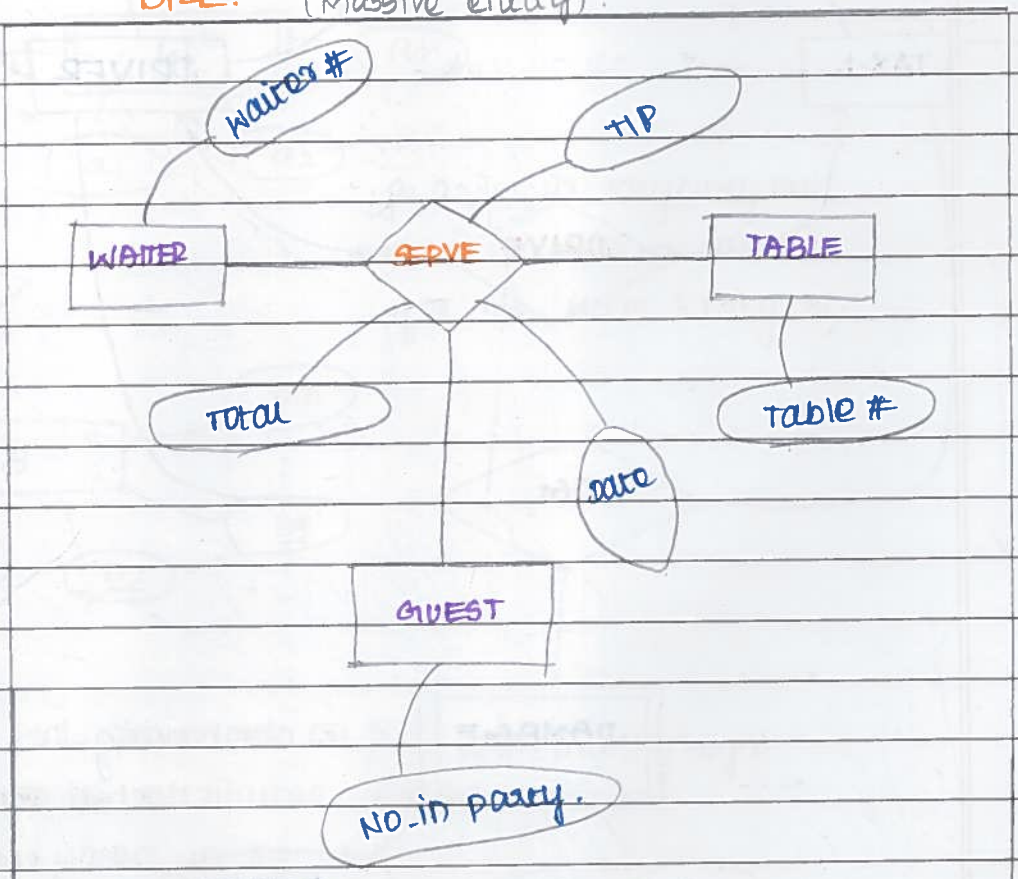


Example of Aggregation:

REGISTRATION (Massive Entity)

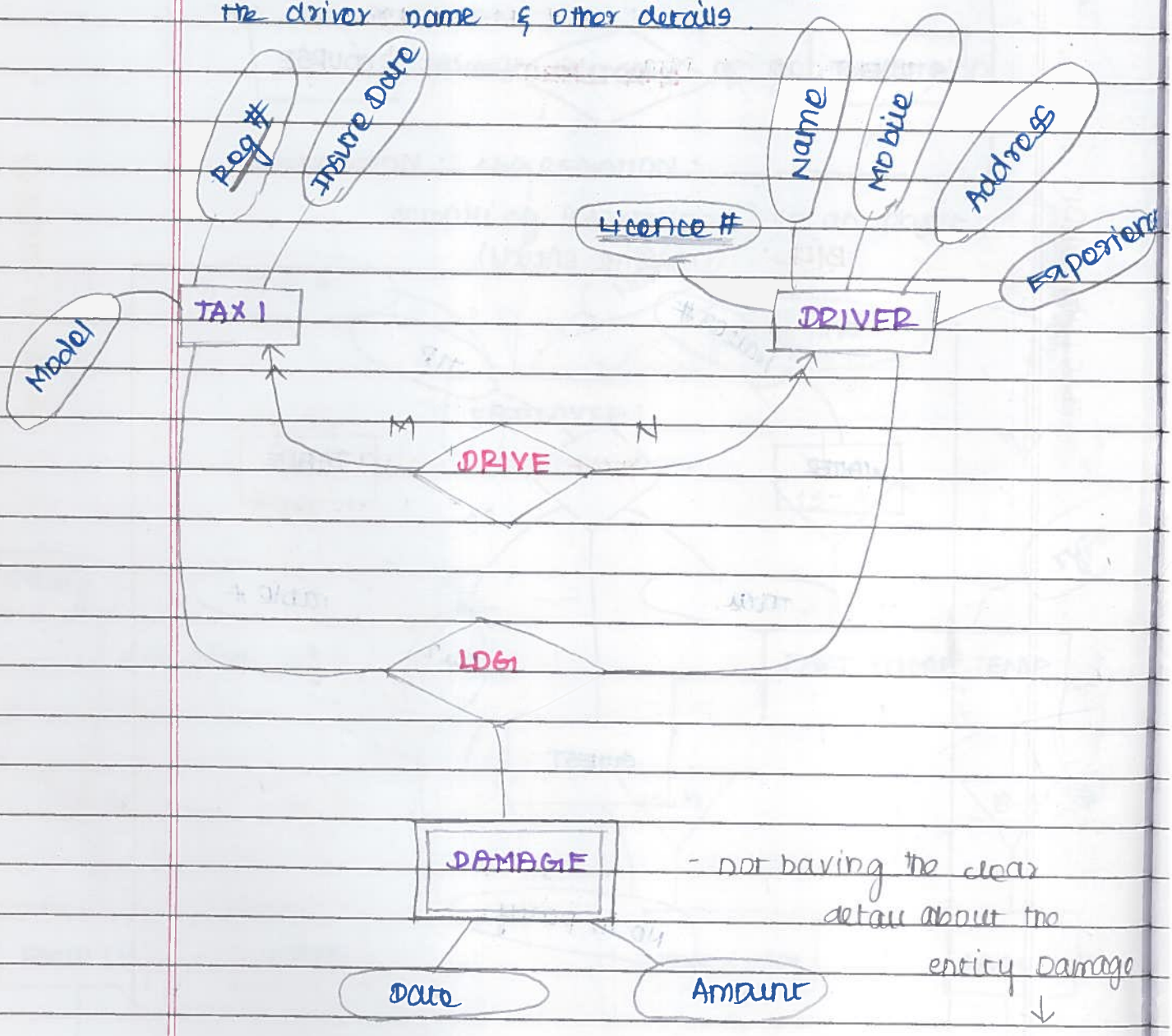


BILL: (Massive entity)



EXERCISE 1 : Design an ER Data Model

DRIVE SAFE Travels maintain a Fleet of private taxis. The list of drivers and their reg# (licence) is also maintained by them. For every car a log book records details about any damage / accident that might have taken place along in the date & the driver name & other details.



- not having the clear detail about the entity Damage

↓

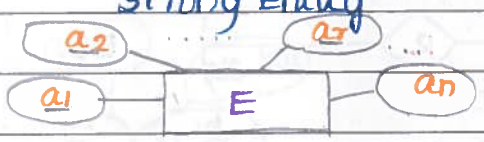
Weak Entity

18/10

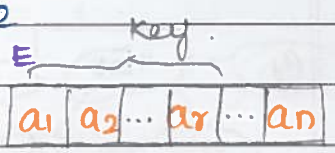
Scribd Date / / Page 29

Transformation / conversion: (ERD into Table)

Strong Entity



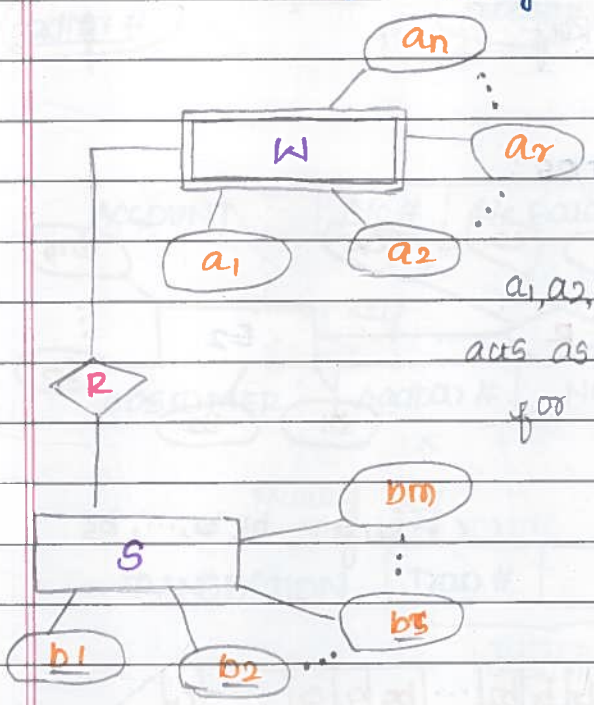
Table



→ Entity E is converted into a table named E

→ And the attributes are converted as a table fields (column).

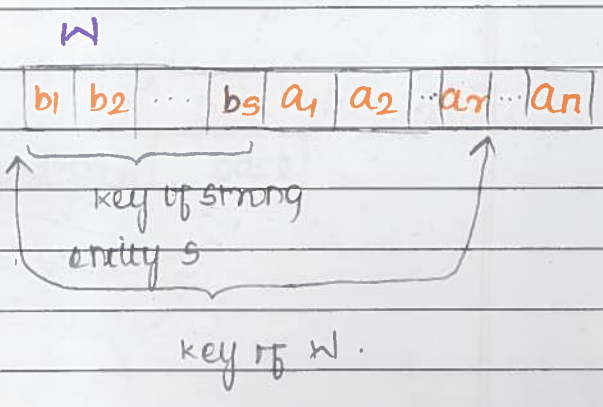
Weak Entity



a1, a2, ..., as may/may not acts as a discriminator for the Weak Entity W.

Table

W is dependent on S → of S (primary keys)
So the attributes will be placed in the W



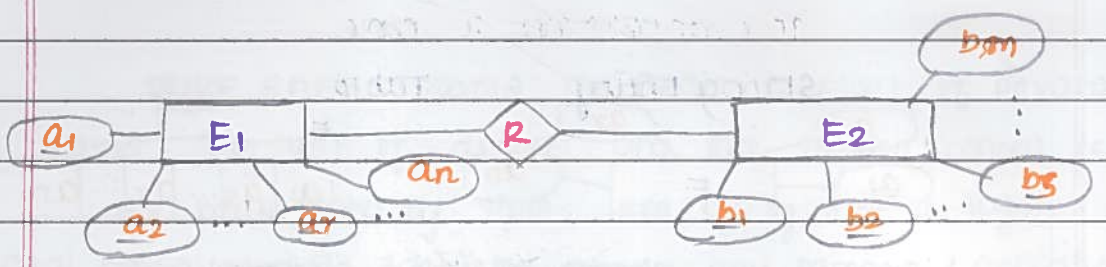
Both primary key of S & discriminant of W1 acts as a key for W1.

Note:-

All relations in the ERD will become a table while conversion.

Scriby Date 1/1 Page 20

Relation with ND attribute :



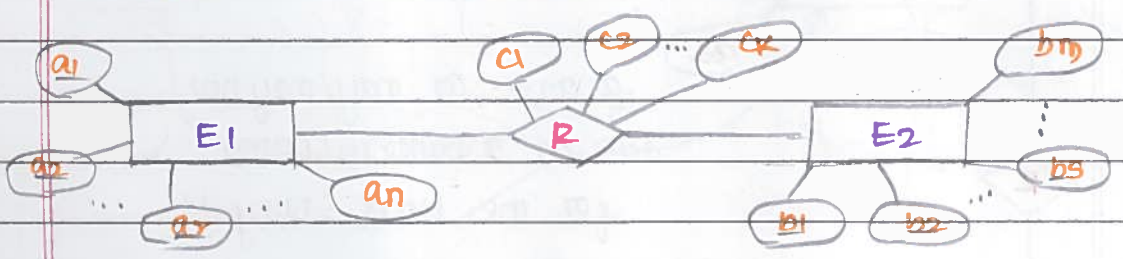
keys :- $a_1, a_2, \cdots, a_r$ keys :- $b_1, b_2, \cdots, b_s$

R

a_1	a_2	$\cdots$	a_r	b_1	b_2	$\cdots$	b_s

key.

Relation with attributes :



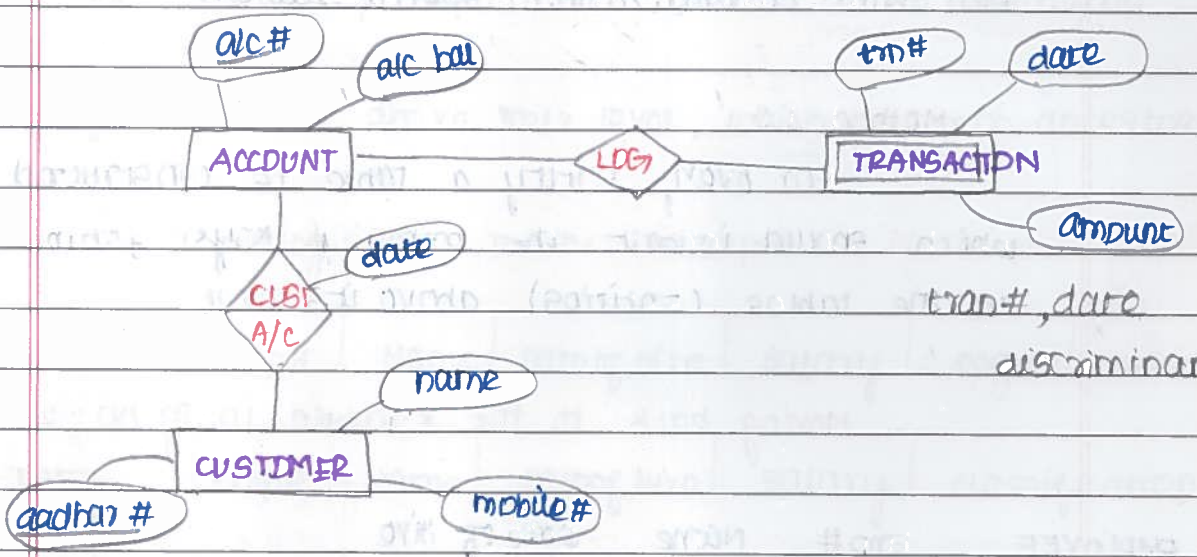
keys :- $a_1, a_2, \cdots, a_r$ keys :- $b_1, b_2, \cdots, b_s$

R

a_1	a_2	$\cdots$	a_r	b_1	b_2	$\cdots$	b_s	c_1	c_2	$\cdots$	c_k

key. attributes of relation.

The following is a snapshot of an ERD in a Banking Enterprise and convert into a table.



tran#, date discriminant.

ACCOUNT | A/C # | A/C Balance |

↑ primary key.

CUSTOMER | Aadhar # | Name | Mobile # |

↑ primary key

TRANSACTION | Tran # | Date | A/C # | Amount |

Tran # + Date → discriminant key

A/C # → primary key

discriminant key + primary key → Key

LOG | A/C # | Tran # | Date |

CUST A/C | A/C # | Aadhar # | Date |

Transformation / conversion of an ERD with Generalization / specialization :-

Method 1 :-

For every Entity a table is constructed which should inherit the primary keys from all the tables (Entities) above its level.

Moving back to the Example in Pg no : 26

EMPLOYEE	Emp#	Name	Date of Hire
FULL TIME EMPLOYEE	Emp#	Salary	
PART TIME EMPLOYEE	Emp#	type	
FACULTY	Emp#	Degree	Interest
STAFF	Emp#	classification	
TEACHING	Emp#	Stippend	
CASUAL	Emp#	Hiredate	

Method 2:-

Only the lowest level entities are converted into tables, & these tables inherit all the attributes of the entities above their level, which are their ancestor.

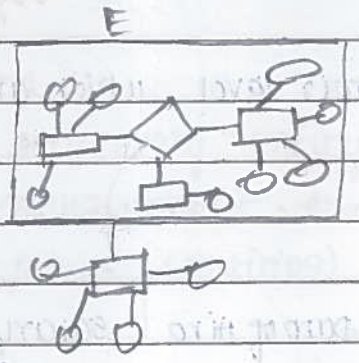
Moving back to the example in Pg no: 26

FACULTY	Emp#	Name	Date of Hire	Salary	Degree	Interest.
STAFF	Emp#	Name	Date of Hire	Salary	Classification.	
TEACHING	Emp#	Name	Date of Hire	type	Stipend	
CASUAL	Emp#	Name	Date of Hire	type	Hire date	

Emp# acts as a key attribute in all the tables.

Transformation / conversion of an ERD with Aggregation.

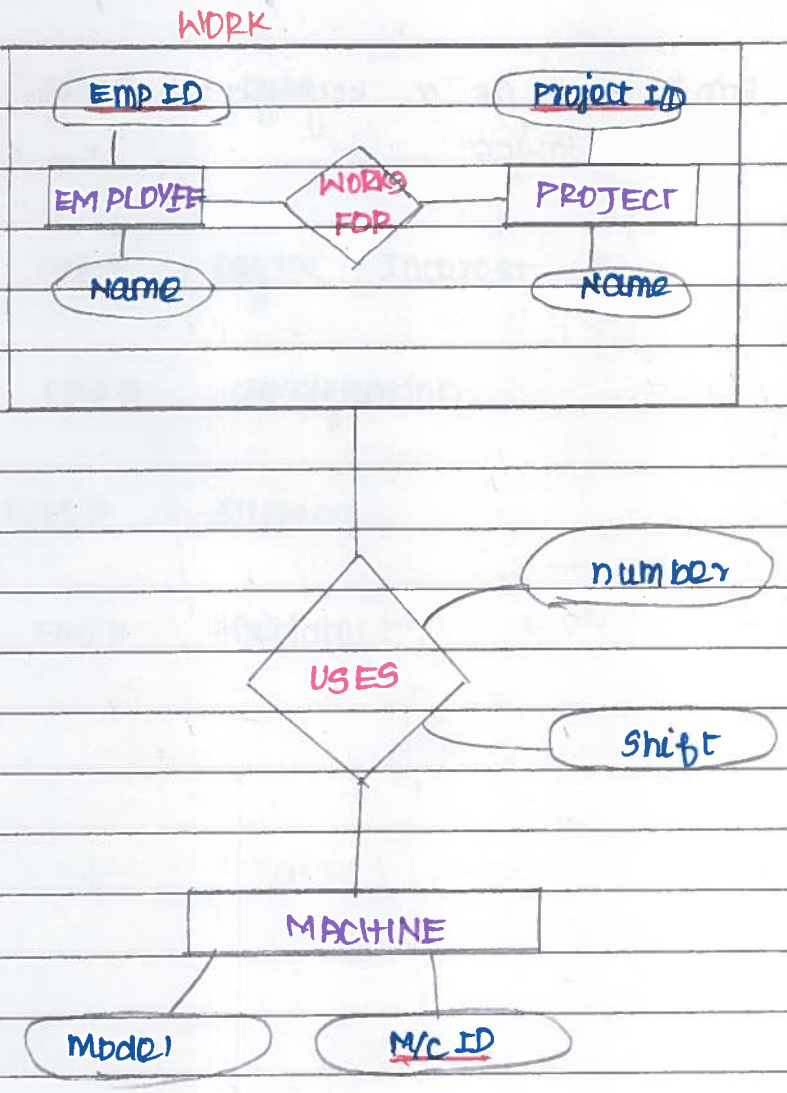
Example available in pg no: 27



dominant relationship will be used

In the case of Aggregation, dominant relationship plays a significant role, in the conversion of ERD into tables.

Example:



EMPLOYEE	EmpID	Name

PROJECT	ProjectID	Name

MACHINE	M/C ID	Model

WORKS FOR	Emp ID	Project ID

USES	M/C ID	Number	Shift	EmpID	Project ID

EXERCISE 2.

A hospital has several laboratories attached, several wards & several doctors. Patients are admitted in wards, the wards have supporting staff, several tests are conducted on patients in the labs.

(i) Design an ERD for the above Enterprise.

(ii) Convert the ERD into Table

(iii) Check whether the SQL queries for the following questions produce the appropriate results.

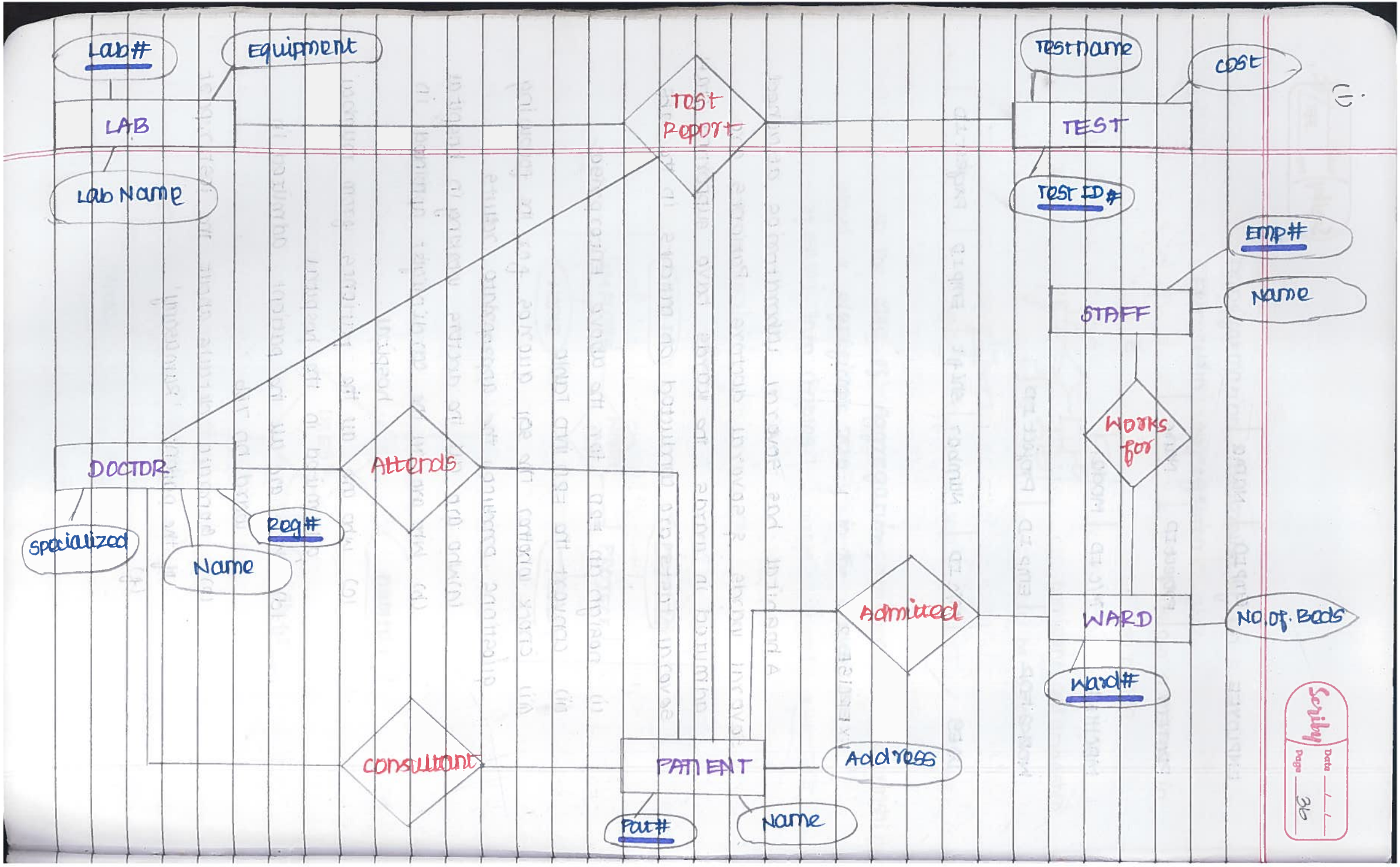
(a) Who are all the doctors working in hospital.

(b) Who are all the cardiologist appointed in hospital.

(c) Who are all the patients from mumbai addmited in the hospital

(d) Who are all the patient admited in ward no 756

(e) Generate details about the test report of the patient 'Ramasamy'


Lab#
Equipment
LAB
Lab Name
TEST REPORT
TestName
Cost
TEST
Test ID#
Emp#
Name
STAFF
WORKS for
DOCTOR
Attends
Specialized
Name
Reg#
Admitted
WARD
No. of Beds
Ward#
CONSULTANT
PATIENT
Address
Pat#
Name

(ii)

DOCTOR	Reg# ↑ PK	Name	specialized
PATIENT	Pat # ↑ Pk	Name	Address
WARD	Ward # ↑ Pk	NO.Of. Beds	
STAFF	Emp # ↑ Pk	Name	
LAB	Lab # ↑ Pk	Lab Name	Equipment
TEST	Testname	COST	Test ID# ↑ Pk
ATTENDS	Reg #	Pat #	
CONSULT	Reg #	Pat #	
ADMITTED	Pat #	Ward #	
WORKS FOR	Ward #	Emp #	
TEST REPORT	Lab #	Reg #	Test ID #

(iii)

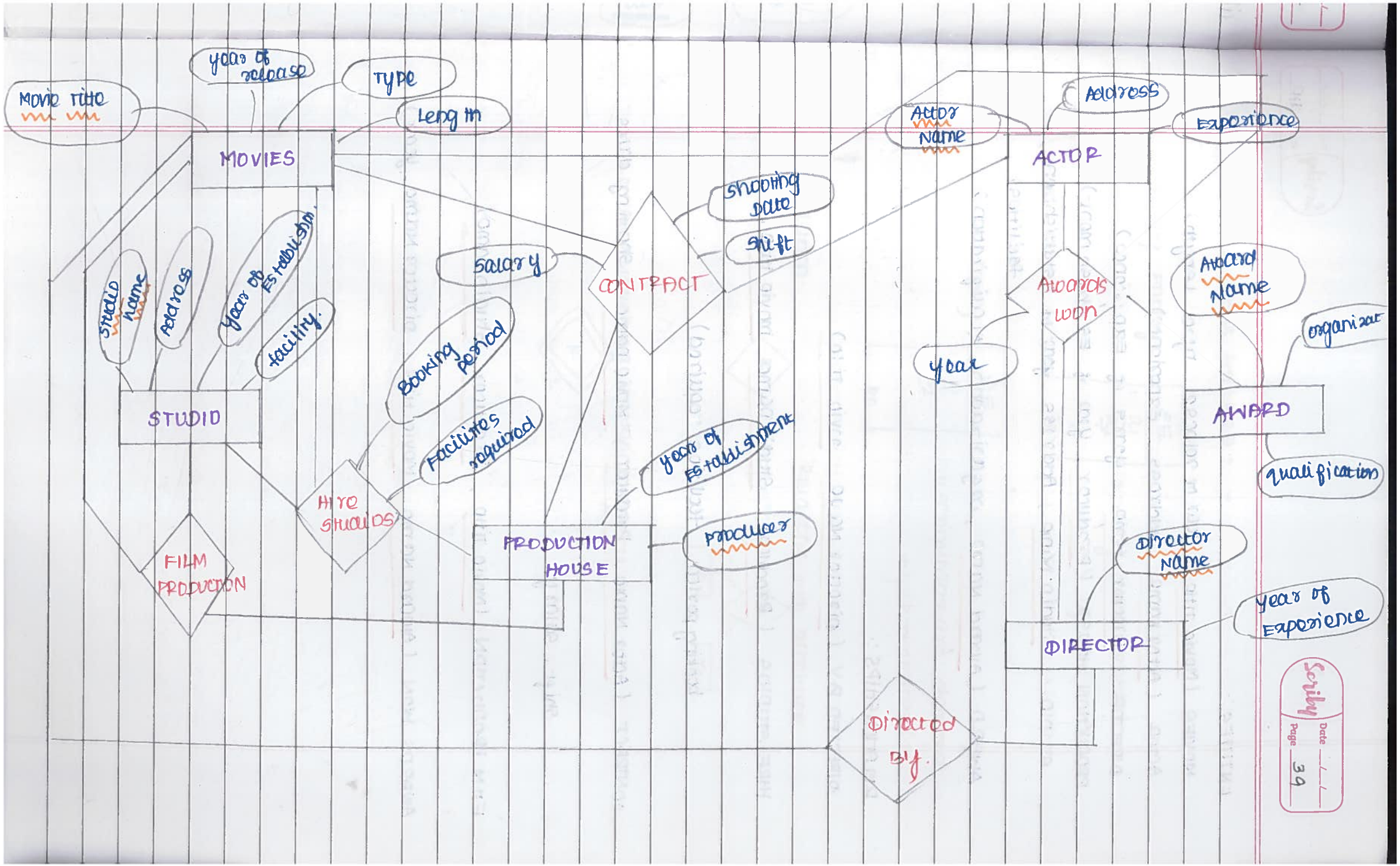
(a) Select * from DOCTOR;

(b) Select * from DOCTOR where specialized = 'CARDIOLOGIST

(c) Select * from PATIENT where Address = 'Mumbai';

(d) Select Pat # from ADMITTED where Ward # = 756;

(e) Select * from TEST where TestID = (Select Test ID from TESTREPORT where Pat # = (Select Pat # from PATIENT where Name = 'Ramasamy'));


Movie title
year of release
type
Length
MOVIES
Studio name
Address
Year of Establishment
facility
STUDIO
Salary
Booking period
facilities required
HIRE STUDIOS
FILM PRODUCTION
CONTRACT
shooting date
shift
PRODUCTION HOUSE
year of Establishment
producer
Actor Name
Address
Experience
ACTOR
Awards won
year
Award Name
organizer
AWARD
qualification
Director Name
year of Experience
DIRECTOR
Directed By.

ENTITIES :-

MOVIES (Movie title, year of release, type, Length)

ACTOR (Actor name, Address, Experience)

DIRECTOR (director name, years of Experience)

PRODUCTION HOUSE (Producer, year of Establishment)

STUDIO (Studio Name, Address, year of Establishment, facilities)

AWARD (Award Name, organization, qualification)

RELATIONSHIPS :-

DIRECTED BY (director Name, Movie title)

HIRE STUDIOS (Producer, Studio name, movie title, booking period, facilities required)

CONTRACT (Actor Name, Producer, Movie name, Shooting dates, shift, salary)

FILM PRODUCTION (Movie title, Producer, studio name)

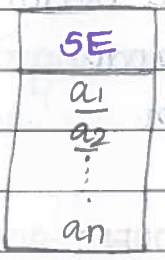
AWARDS WON (Award Name, Movie title, director Name, year)

27/10

Scribe Date / / Page 41

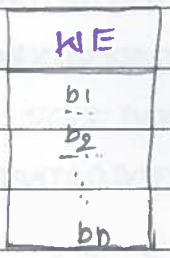
Alternate icons for ERD :

Strong Entity.



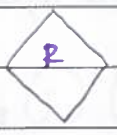
keys are denoted by line (—)

Weak Entity

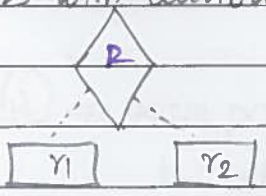


discriminator are denoted by broken lines (- - - -)

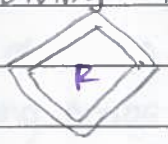
Relation



Relations with attribute.



Relations involving in weak Entity.



Data Model: → developed by IBM

* Hierarchical data model (HDM)
* Network data model → HDBMS
* Relational model

HIERARCHICAL DATA MODEL: (Historical data model)

Designed, commercialized & marketed by IBM.

1960's
↳ mainframes

IMS - Information management system
↑ contains all the characteristics of DBMS.
↓ makes use of Hierarchical data model (HDM)

IMS ↑ obsolete

HDM ⇒ HDBMS

Legacy databases retains IMS
→ Banks
→ Insurance companies
→ Hospitals.

1970's or 1980's
MRI system corporation :
↓
DBMS - Hierarchical data model
↓
HDBMS
↳ system 2000 (software)

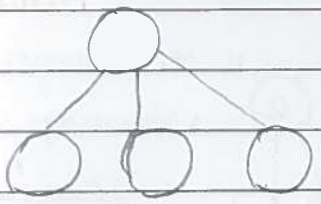
taken over → Intel.

RDM mobile
↳ Embedded mobile database } Hierarchical data model (HDM)

Windows registry → Hierarchical data model.
↑ part of OS

HDM structure: HDM makes use of tree structure.

Tree structure organizes data in Hierarchy.

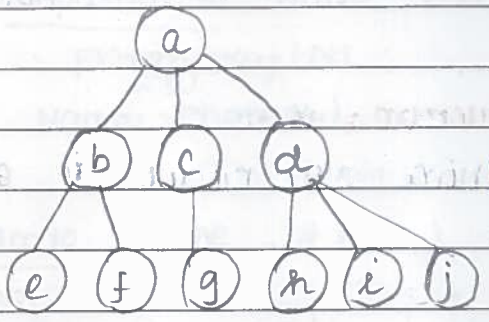


top node of the tree
↳ root node.

In a tree only we can have one root node.

○ → nodes

—— → Edges / Branches

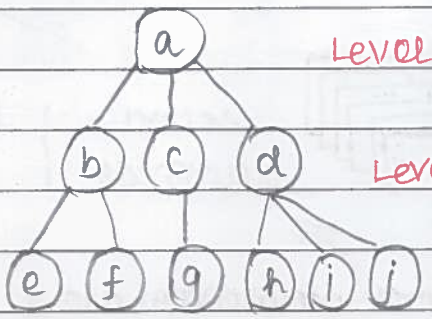


→ root node
→ child node
→ parent node
→ Leaf node
→ sibling node

a - root node
b, c, d - child nodes of root node a
a is a parent node of b, c, d.

e, f, g, h, i, j - Leaf node
↓
last level of tree that do not give raises to any branch nodes.

b, c, d are the sibling nodes of the parent node a ← children of the same parent.



hierarchical concept

Height - maximum level of the tree.

Height() = 3

Hierarchical path - sequence of edges between root & the ~~leaf~~ specific node.

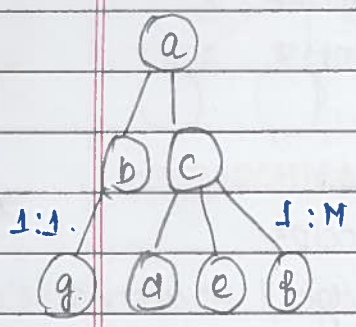
Hierarchical path (j) = a → d → j

one-one, one-many,
many-one, many-many.

Relationship between data → IBM stucked to only one relationship.

Parent-child relationship ←

↳ 1:1, 1:M } naturally supported by parent-child relationship



Drawback : (M:N)

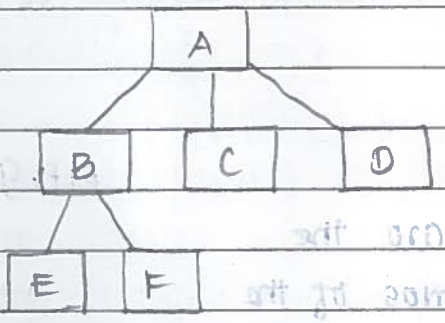
Handling of many-many relation is not done in HDM.

Database structure :

. Hierarchical data model is structured liked this.

Record type - ▭

Entity - is denoted as Record type

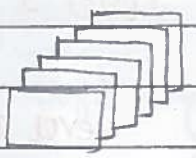


A - Root record type

B, C, D - child record type

B - Parent record type.

Record types are cutted open to see the attributes

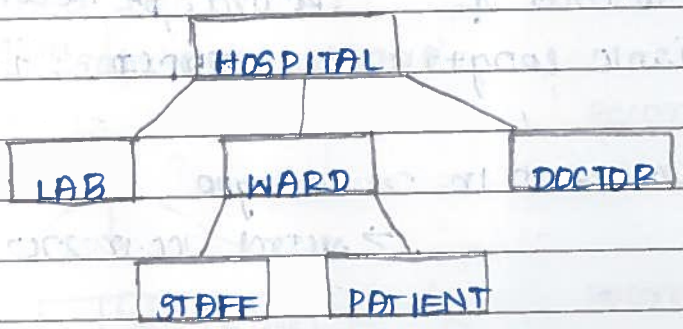


record occurences } Bunch of record occurences are denoting record type.

Each record occurences are bundled with their respective child with the help of pointer.

A case study Hierarchical data model.

Hospital Management system



Hospital (Hospital code, Name, Address, Phone #, # beds)

Primary key (PK)

Lab (Lab #, Name, Phone #)

PK

Ward (Wardcode, Name, # beds)

PK

Staff (Emp #, Name, duty, shift, salary)

PK

Patient (Reg #, Bed #, Name, Address, DOB, Gender)

PK

Doctor (DOC #, Name, speciality)

PK

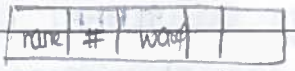
31/10

Scriby Date / / Page 46

Implementation of Hierarchical database:

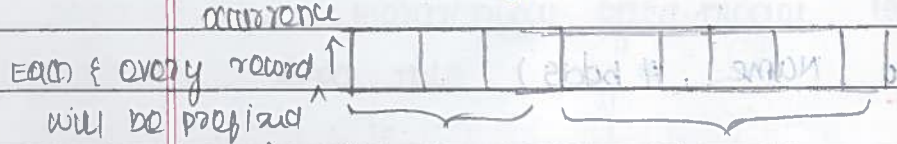
→ All the records of variable length → Record type ↳ template → record type HOSPITAL

→ giving values to the record type
↳ record occurrence.

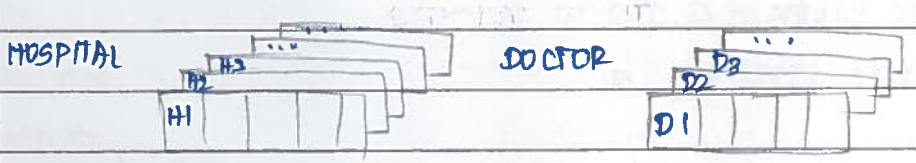


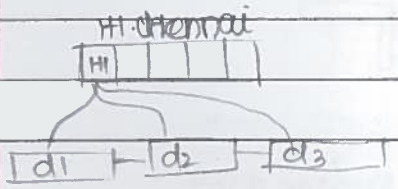
values of record type

→ various record occurrence with distinct variable length will be stored as.



Structure of hierarchical database.





Every parent occurrence need to be linked with the respective child occurrences.

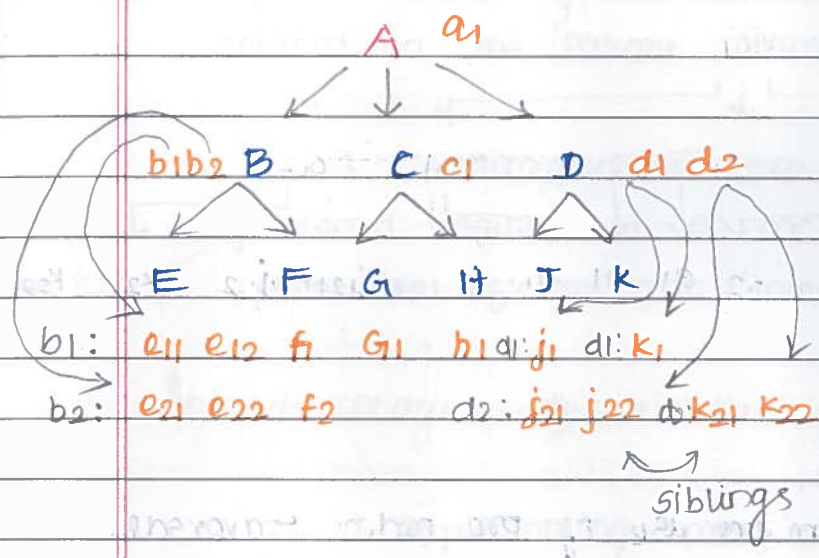
↓

clusters of data link will be created.

Method of storage in the database:

Method 1:

sequential storage:



Record type:
A, B, C, D, E, F, G, H, J, K

Record occurences:
$a_1, b_1, b_2, c_1, d_1, d_2, e_{11}, e_{12}, e_{21}, e_{22}$

sequential order is followed to store the data inside memory.

→ a_1 b_1 e_{11} e_{12} f_1 b_2 e_{21} e_{22} f_2 c_1 g_1 h_1 d_1 j_1 k_1 d_2 j_{21} j_{22} k_{21} k_{22}.

sequentially completing each branch by connecting to the records

Method 2: (pointers)

child / sibling pointers.

Each record occurence has two types of pointers

child pointers → pointing to its left most child record occurrence

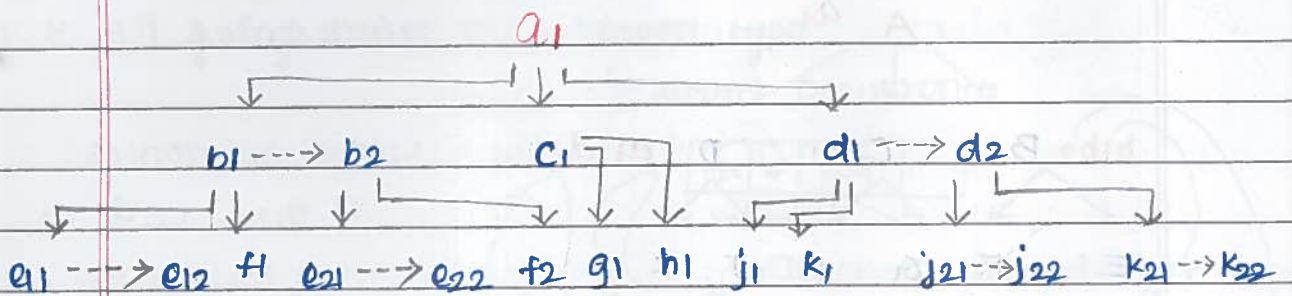
sibling Pointer → to its right sibling.

Therefore, each record has one sibling pointer & as many child pointers.

Legend :-

→ - child pointers

⇢ - sibling pointers.



Method 3 :

Method 3 makes use of pre-order traversal.

characteristic of hierarchical data base :-

* A hierarchical data base is a hierarchical tree - one root record type - no parent record type for the root record type.

* Root can have any no of child record type - each of which can form a sub tree

* Each child record type can have only one parent record type.

* Data in a parent record applise to all its child records

* Each occurrence (value to the record type) of a record type can have any no of occurrences of each of its child.

* A child record occurrence must have a parent record occurrence.

* Deleting a parent record occurrence requires deleting all child record occurrences that related to the parent record occurrences.

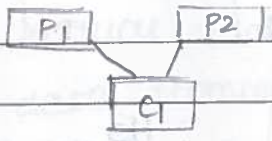
* A hierarchical tree can have any number of record occurrences on each record type at each level of the tree including the root.

Disadvantages of Hierarchical Data base:

(M:N)

Representing many-many relationship

The case where a record type participates as a child in more than one parent-child record type.



(not possible)

N-ary relationships with more than two participating record types.

01/11

Scribe Date __/__/__ Page 50

Handling of M:N relationships in HDM:

Replication Method
Virtual records

Replication Method:

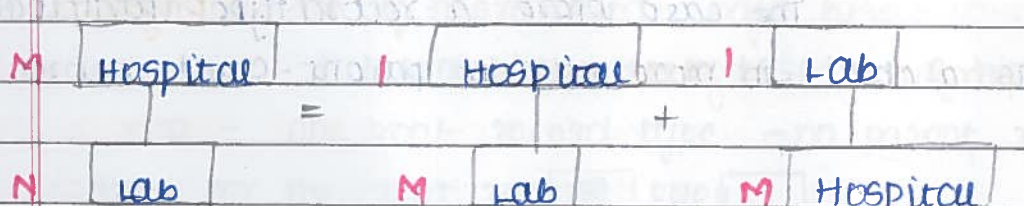
duplicating multiple times

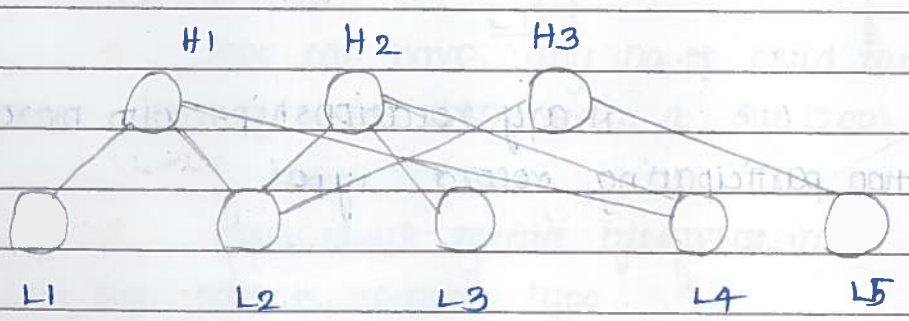
M | Hospital | Parent

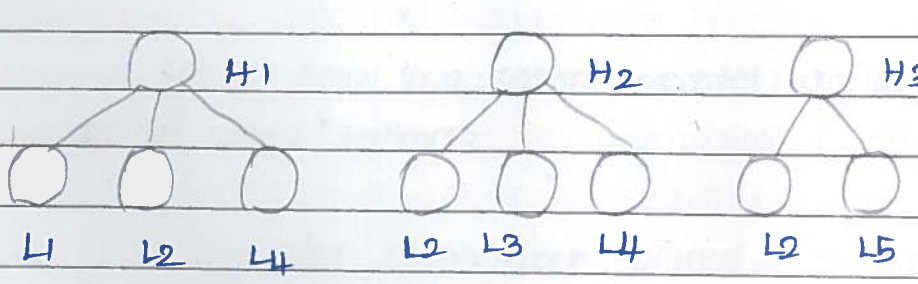
N | Lab | child

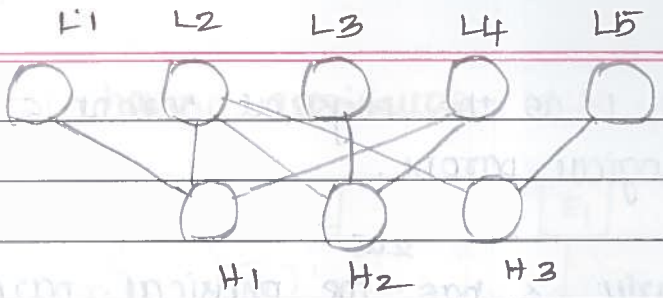
Many-Many relationship can be converted into One-Many relationship as follow:

M:N → 2 (1:M) relationships.









The disadvantage of this method is redundancy.

Replication (duplication) is where a M:N relationship is treated as a combination of two 1:M relationship at the cost of redundancy.

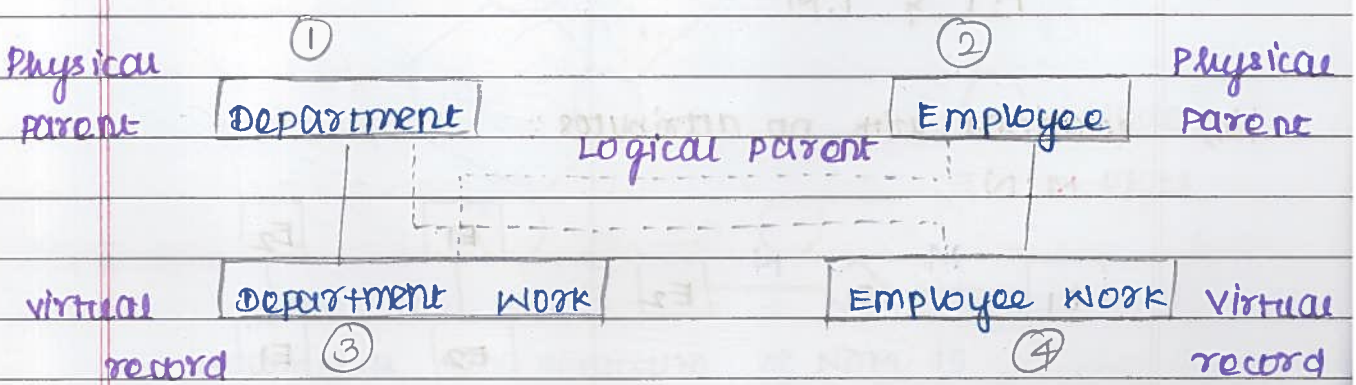
Virtual records :

In this method to tackle M:N relationship, we introduce 2 virtual record types, each of the virtual record type having a physical parent & logical parent.

The virtual record can contain more instruction data (common data) along with some relevant fields.

Eg :

Factory :



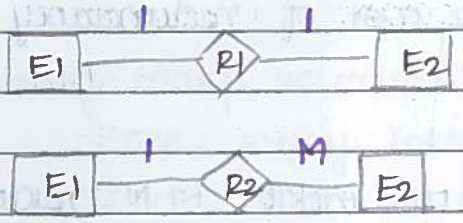
3 has 1 as the physical parent & 2 as the logical parent.

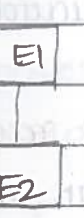
Similarly 4 has the physical parent & 2as 1 as the logical parent.

Conversion of ER data Model (ERD) into a Hierarchical data model (HDM):-

ERD | HDM

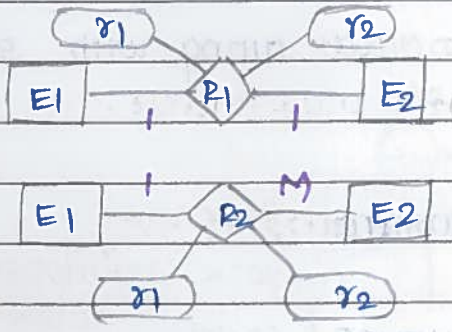
(i)

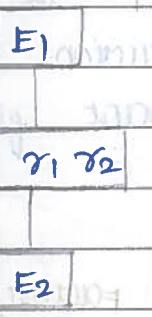




1 : M & 1 : 1

(ii) Relations having attributes:-

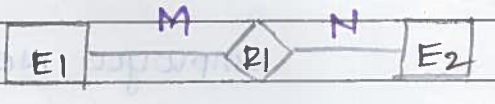


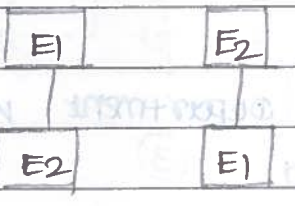


1 : 1 & 1 : M

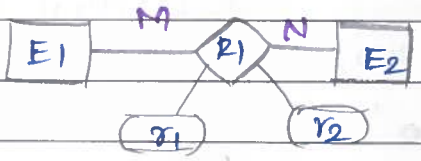
(iii) Relations with no attributes:-

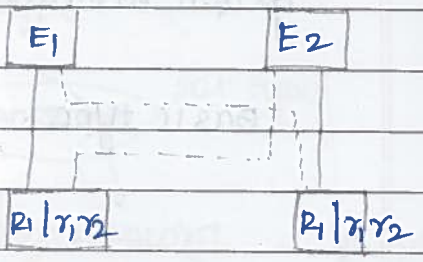
M : N





(iv) Relations having attributes M:N





03/11 **NETWORK DATA MODEL :: (NDM)**

↳ Network Database Management System (NDBMS)

Language :: CODOL (traditional File processing)

common Business oriented Languages.

↳ proposed by the committee namely CODASYL

↓

conference on Data systems Language.

Traditional File processing

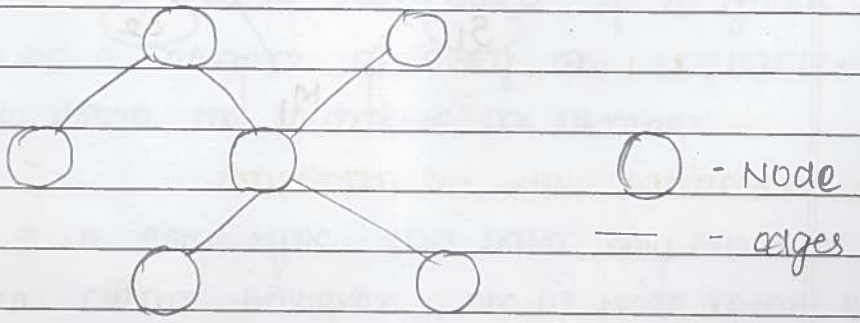
↓

DBTG :: Database TASK Group.

↳ cobol → Network Data Model.

HDM was flopped because of NDM.

DBTG came up with → Network / graph data structure.

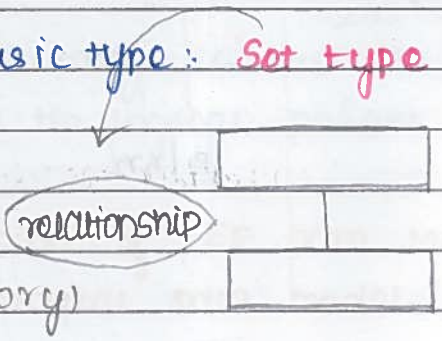


Fundamental data structure of NDM is Network / graph data structure.

Graph is a collection of Nodes & Edges.

Our data model is structured base of network / graph structure.

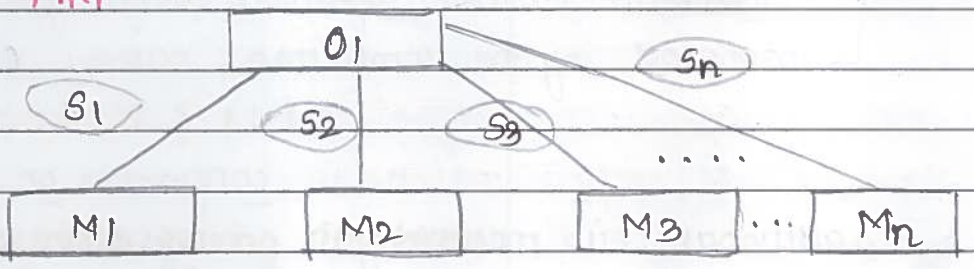
Basic type : Set type



Labelling of relationship is set type (mandatory)

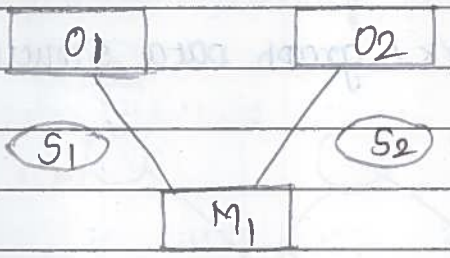
(i) Each and every owner can have N number of member record type with each and every unique set type.

1 : M



(ii) One member can have many owner record type with distinct set type.

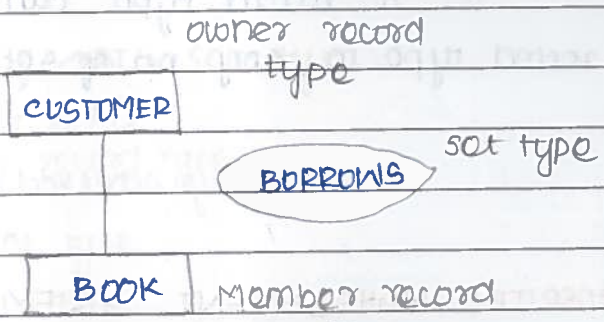
M : 1



set type → DBTG set type.

Example:

Library:



Note:

Network Data Model is the collection of DBTG sets.

characteristics:

⇒ DBTG set construct and characteristic:

* A set type should be named and must have one owner record type & one or more member record type(s).

* A record occurrence (record with values) of a given record type can be owner of only one occurrence of a set type where a record type is the owner.

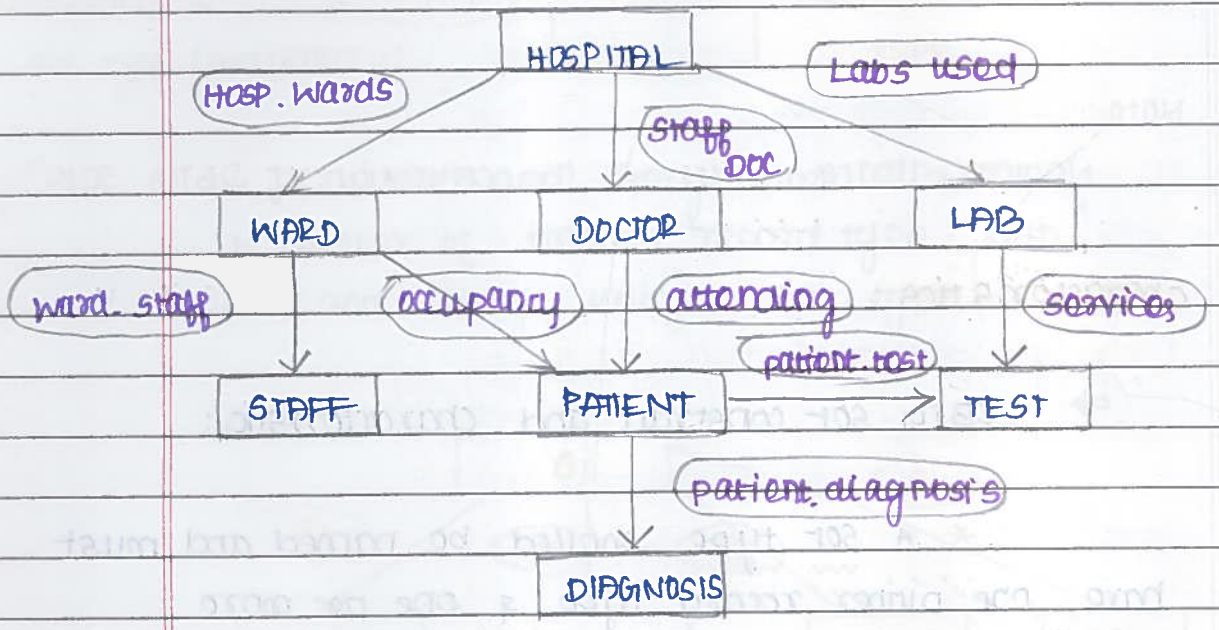
* A record occurrence of a given record type can be a member of only one occurrence of a set type where the record is the member.

* A set type can have only one type of record as owner. However, one or more record types can be members of the set.

* A record type can be the owner record type in any no of set types.

* A record type can be the member record type in any no of set types.
(M:N relationship can be easily achieved)

Example: HOSPITAL MANAGEMENT SYSTEM



HOSPITAL (Hosp. code, Name, Address, Ph#, # of beds)

WARD (Ward code, Name, # of beds)

DOCTOR (DOC#, Name, speciality)

LAB (Lab#, Name, Address, Ph #)

STAFF (Emp #, Name, duty, shift, salary)

PATIENT (Reg #, Bed #, Name, Address, DOB, Gender)

TEST (Test code, type, date, specimen)

DIAGNOSIS (diag code, type, complications)

7/11

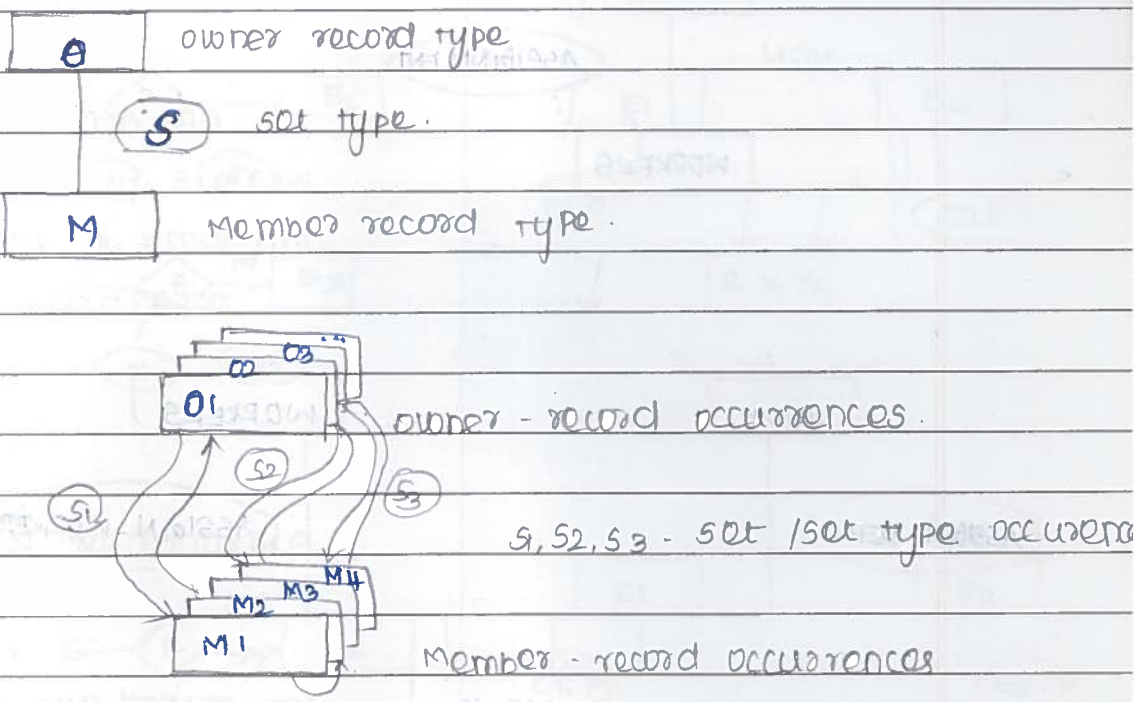
ERD → SQL.

Scriby Date / / Page 57

Data models :

NDM

Implementation of DBTG set type :-

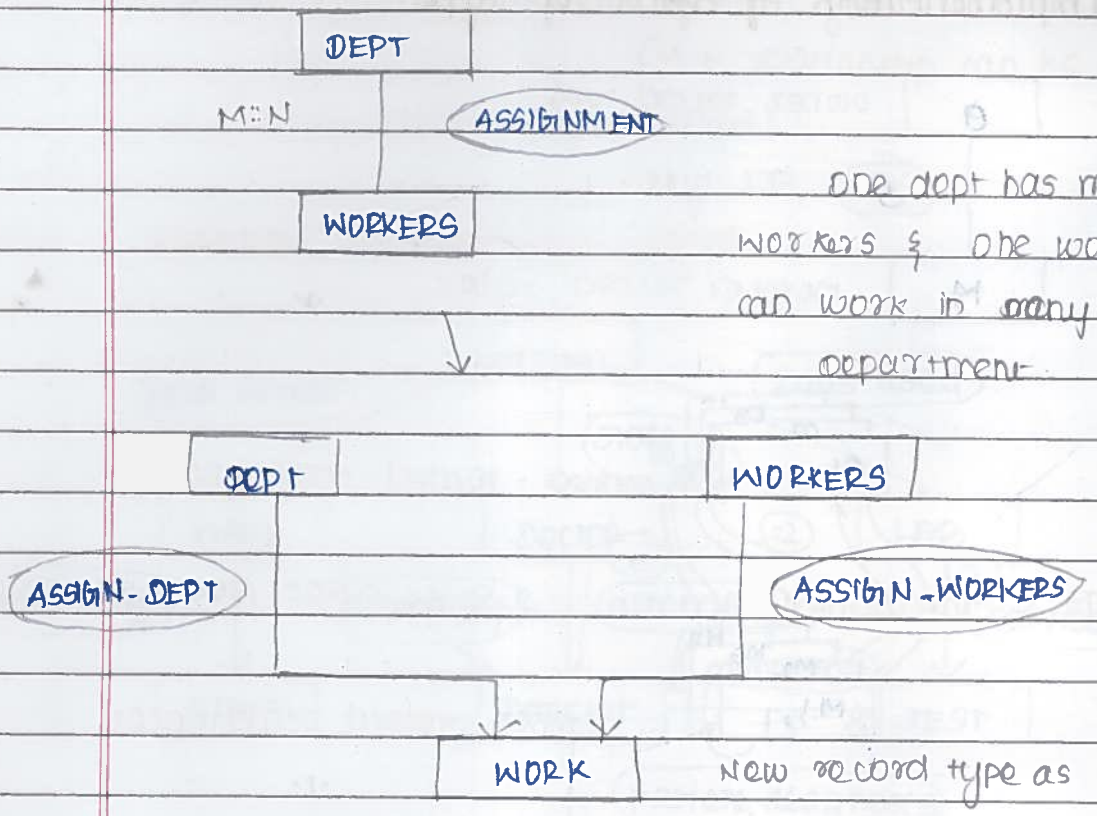


Implementation of DBTG set type → single pointer record base unit.

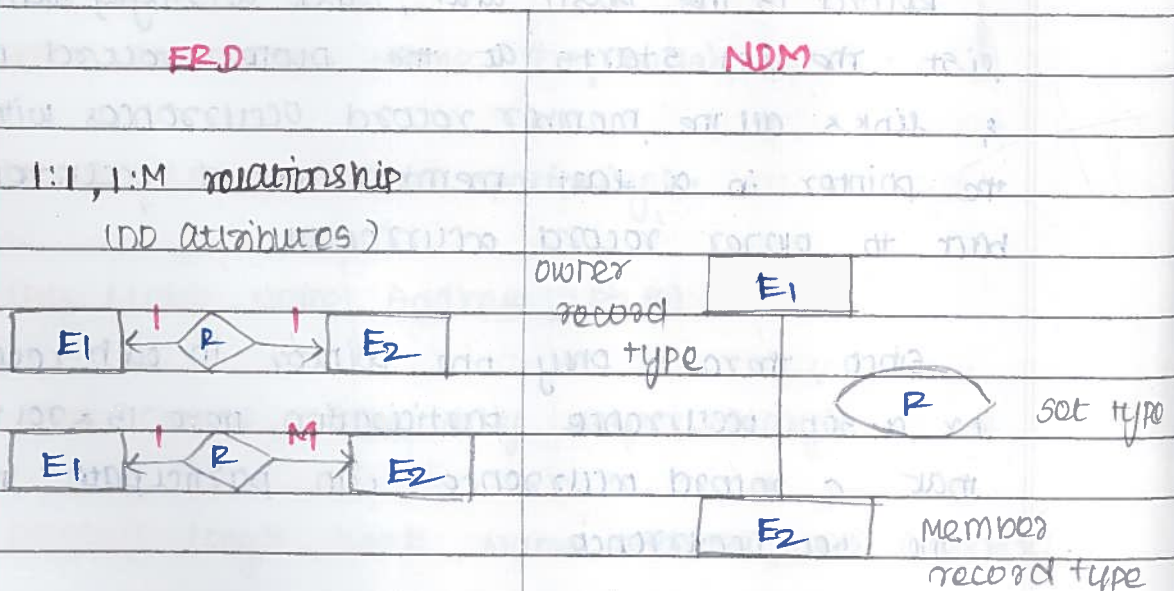
Record is the basic unit, and a singly linked list. The list starts at the owner record occurrence & links all the member record occurrences with the pointer in a last member record occurrence pointing back to owner record occurrences.

Since, there is only one pointer in each record for a set occurrence participation there is a restriction that a record occurrence can participate in only one set occurrence.

Implementation of M:N relationship in DBTG-NDM:

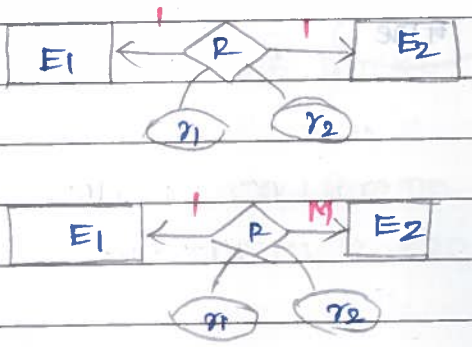


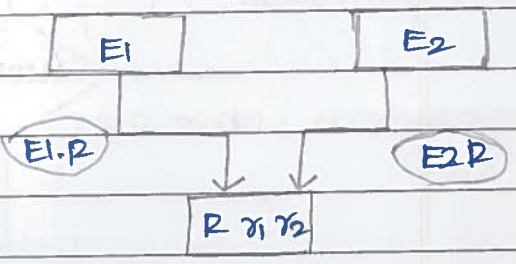
Conversion of ER data Model (ERD) into a Network data Model (NDM)



ERD | NDM

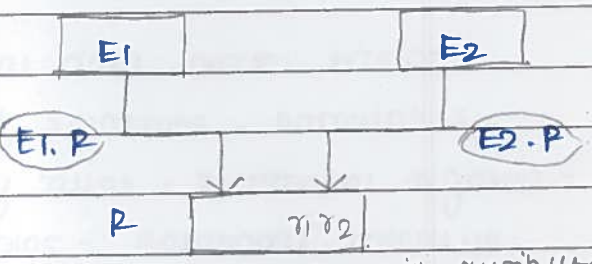
1:1, 1:M relationship (with attributes)



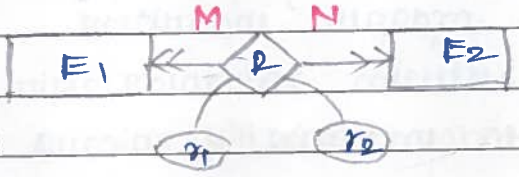


M:N relationship

without attributes



with attributes

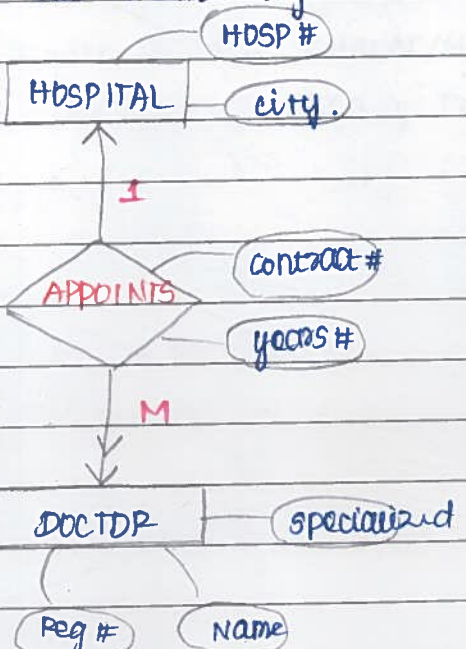


↳ with attributes

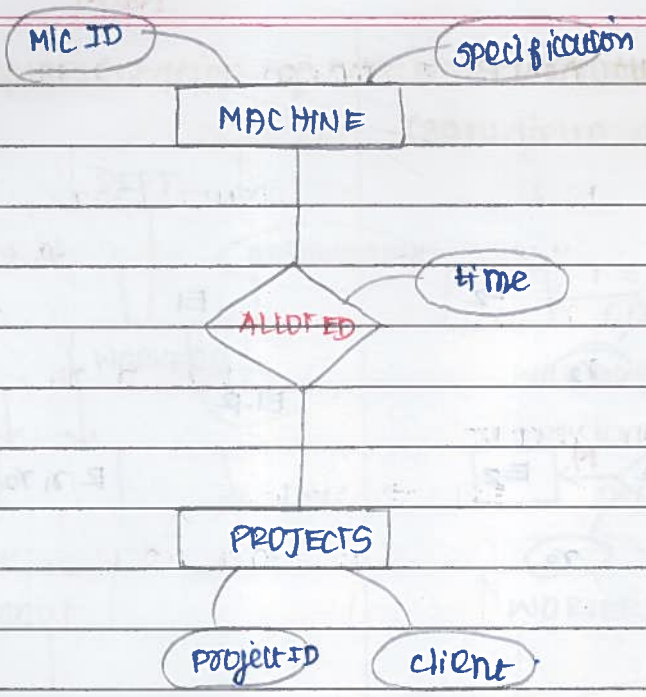
In practice R can contain any common information relevant to E1 & E2.

Map the following ERD into HDM and NDM.

1.



2.


MC ID
specification
MACHINE
time
ALLOTED
PROJECTS
projectID
client

course outcome 1 : 25 marks : 45 mins.

(a) 3 marks → introduction, advantages, disadvantages, structure

(b) 7 marks → data model, characteristics.

(c) 15 marks → Application.
↳ ERD → table.
↳ other model conversion.

course outcome 2 :

Relational Model : Relational data model basics - codd's rule - properties of relations - domains & key constraints . Integrity rules - Relational algebra - Relational algebra queries - Relational calculus : Tuple Relational calculus, Domain Relational calculus - Queries in Relational calculus.

↳ assignment presenta

(a) } Integrity rules
(b) }

(c) 15 marks.
↳ Relational algebra query
DRC , TRC

Mathematical Relation.

RELATIONAL MODEL :

(TABLE) Relation

	C_1	C_2	....	C_n
r_1				
r_2				
r_m				

Data Models :
- HDM
- NDM
- RDM (E.F. Codd)

Relation in the Relational data Model pictorial represents table but actually means that the Mathematical relation.

Mathematical Relation
- Relational Algebra
- Relational calculus
  - DRC — Domain Relational calculus
  - TRC — Tuple Relational calculus.

Definition of a Relation :

A relation is defined as a subset of the cross product (×) of its underlying domains.

Given sets $D_1, D_2, D_3, \cdots, D_n$ with not necessarily distinct, R is a relation if

$$R \subseteq D_1 \times D_2 \times \cdots \times D_n$$

D_i - Domain

Domain is a set from which the values are taken.

8/11

Scriby Date / / Page 63

Domain Example: column A is taking values from Domain

$D1 = \{a_1, a_2, a_3\}$

R:

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

column B is taking value from D2

$D2 = \{b_1, b_2, b3\}$.

column C is taking value from D3

$D3 = \{c_1, c_2, c_3\}$

column A, B, C cannot take values other than their corresponding Domain

$D1 \times D2$ (a,b) ordered pairs

$D1 \times D2 \times D3$ (a,b,c) triple

$D1 \times D2 \times D3 = \{ (a_1,b_1,c_1), (a_1,b_1,c_2), (a_1,b_1,c_3), (a_1,b_2,c_1), (a_1,b_2,c_2), (a_1,b_2,c_3), (a_1,b_3,c_1), (a_1,b_3,c_2), (a_1,b_3,c_3), (a_2,b_1,c_1), (a_2,b_1,c_2), (a_2,b_1,c_3), (a_2,b_2,c_1), (a_2,b_2,c_2), (a_2,b_2,c_3), (a_2,b_3,c_1), (a_2,b_3,c_2), (a_2,b_3,c_3), (a_3,b_1,c_1), (a_3,b_1,c_2), (a_3,b_1,c_3), (a_3,b_2,c_1), (a_3,b_2,c_2), (a_3,b_2,c_3), (a_3,b_3,c_1), (a_3,b_3,c_2), (a_3,b_3,c_3) \}$

$D1 \times D2 \times D3 \times \cdots \times Dn$ (a,b,c,....,z) n-tuple

$|D1 \times D2 \times D3| = |D1| * |D2| * |D3|$

cardinality → no. of elements in set

$= 3 * 3 * 3 = 27$

$\{(a_1,b_1,c_1), (a_2,b_2,c_2), (a_3,b_3,c_3)\} \subseteq D1 \times D2 \times D3$

Terminologies:

R	A1	A2	A3	....	An
rows					

columns

Given a relation R with columns $A_1, A_2, \ldots$ An & rows.

The columns $A_1, A_2, \ldots, A_n$ are described as Attributes. & If there are n attributes that is n-columns then n is known as degree of R (or) arity of R

Relation with 2 Attributes - Binary
3 Attributes - Ternary
⋮
n Attributes - n-ary

The rows are described as tuple & If there are m rows that is m is known as cardinality of R

Every row was known as tuple.
A relation is made up of tuple.

Every relation must have a subset of the attributes that uniquely identify a tuple in the relation. → Key.

this notion was called as a key. A key in a relational model should satisfy the following two characteristics:

⇒ P_1: Unique Identification
In each tuple of the relation the value of the key uniquely identifies the tuple that is no two tuples (rows) can have the same value for the attributes in the key taken as a whole.

⇒ P2: Non redundancy (minimality)

Key with two attributes

STUDENT

	ROLLNO	DEPT	NAME	DOB
PK (ROLLNO, DEPT)				
↳ It is possible & it survives				

↓

But minimality characteristic is not satisfied.

A key should be minimal.

NO attribute that is the part of key can be removed without destroying property P1, that is the key should be minimal.

refer & write. CODD'S RULES :- 12 rules. (*)

The Foundation rule:

The database must be in relational form. so that the system can handle the database through its relational capabilities.

Information rule:

A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.

Guaranteed Access rule:

Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name and column name.

Systematic Treatment of Null values:

This rule defines the systematic treatment of

Null values in database records the null value has various meanings in the database, like missing the data. no value in a cell. inappropriate information, unknown data and the primary key should not be null.

Active dynamic online catalog based on the relational model:

It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database and implement a similar query language to access the database.

Comprehensive data sublanguage Rule

The relational database support various languages, and if we want to access the database, the language must be the explicit, linear or well-defined syntax, character strings and supports the comprehensive: data definition, view definition. data manipulation, integrity constraints, and limit transaction management operations. If the database allows access to the data without any language, it is considered a violation of the database.

View updating Rule:

All views table can be theoretically updated and must be practically updated by the database systems.

Relational level operation (High-level insert, update & delete) rule:

A database system should follow high-level

2. RUSTional operations such as insert, update & delete in each level or a single row. It also supports union, intersection and minus operation in the database system.

Physical data independence:

All stored data in a database or an application must be physically independent to access the database. Each data should not depend on other data or an application. If data is updated or the physical of the database is changed, it will not show any effect on external applications that are accessing the data from database.

base.

Logical data independence rule:

It is similar to physical data independence. It means, if any changes occurred to the logical level (table structures), it should not affect the user's view (application).

Distribution independence rule:

The distribution independence rule represents a database that must work properly, even if it is stored in different locations and used by different end-users. Suppose a user accesses the database through an application; in the case, they should not be aware that another user uses particular data, and the data they always get is only located on one site. The end users can access the database, and these access data should be independent for every user to perform the SQL queries.

10/11

no(attribute) → degree / arity.

attributes

Relation	A	B	C
→			
→			
→			
tuples →			

Key.
(i) unique identification
(ii) Non-redundancy. (minimality)

Relational Model

no(tuple) → cardinality.

Relational Algebra : (SQL)
↳ makes use of Relational Algebra.

- Basic operations
- Additional operations.

Basic operations : Fundamental operations:
- → Union
- → set difference
- → cartesian product
- → Projection
- → selection

UNION :

Let R and S be two relations then R ∪ S which is the union operation is the set of tuples in R (or) S (or) both.

Note :-

R and S must have same arity of attributes the relation which is output doesn't carry any attributes.

Example :

R

A	B	C
a	b	c
d	a	f
c	b	d

S

D	E	F
b	g	a
d	a	f

In some SQL version union permits duplicate.

→ no attribute names.

R ∪ S

a	b	c
d	a	f
c	b	d
b	g	a
~~d~~	~~a~~	~~f~~

duplicate tuples are not permitted.

Note:- arity(R) = 3 arity(S) = 2. (FDs example)

If arity of (R) ≠ arity of (S), union operation cannot be done.

3 ≠ 2

SET DIFFERENCE :-

It is denoted by R-S. It is the set of tuples in R but not present in S.

Note:-

R and S must have same arity of attribute the relation which is output doesn't carry any attributes.

Example:

R

A	B	C
a	b	c
d	a	f
c	b	d

S

D	E	F
b	g	a
d	a	f

R-S

a	b	c
c	b	d

CARTESIAN PRODUCT :

It is denoted by R*S.

R x S is the cartesian cross product of the tuples in R with the tuples in S.

Example :

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	D	E	F
	b	g	a
	d	a	f

R x S	A	B	C	D	E	F
	a	b	c	b	g	a
	a	b	c	d	a	f
	d	a	f	b	g	a
	d	a	f	d	a	f
	c	b	d	b	g	a
	c	b	d	d	a	f

Important points :

If arity (R) = K1 and arity (S) = K2

cardinality (R x S) = K1 * K2

arity (R x S) = K1 + K2

K1 need not be equal to K2

If cardinality (R) = m1 and cardinality (S) = m2

cardinality (R x S) = m1 * m2

PROJECTION :

Projection is denoted by Π.

Given a relation R, the projection operation helps to remove any of the columns (attributes) and/or rearrange the columns (attributes) in any order.

Example:

Let R be a relation

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

$\Pi_{AC}(R)$

project A & C attribute from relation R

A	C
a	c
d	f
c	d

$\Pi_{CBA}(R)$

rearranging columns.

C	B	A
c	b	a
f	a	d
d	b	c

NOTE :

Projection will always works on the relation and the components on which it works as subscript to Π.

After projection operation → Duplicates will be removed automatically.

R	A	B	C
	a	b	c
	d	e	f
	g	h	i
	w	e	f

$\Pi_{BC}(R)$

R	B	C
	b	c
	e	f
	h	i
~~	~~e~~	~~f~~

14/11

Scriby Date / / Page 72

Relational algebra · Basic operations

SELECTION : → rows

Selection is denoted by σ sigma

$\sigma_F(R)$

Selection works on tuples.

where F is a Formulla that comprises of

(i) operands that are constants (or) components (column attributes)

(ii) Arithmetic comparison operators (relational operators)
(>, >=, <, <=, =, ≠)

(iii) Logical operators
(And, Or, Not)
↳ ∧ ↳ ∨ ↳ ¬
} Boolean operations.

Example :

Let R be a relation, S be a relation.

R	A	B	C
	a	b	c
	d	a	f
	c	b	a

S	D	E	F
	b	g	a
	d	a	f

$\sigma_{B=b}(R)$ select * from R where B = b ;

A	B	C
a	b	c
c	b	a

$\sigma_{A=a \wedge C=c}(R)$ select * from R where A=a and C=c ;

A	B	C
a	b	c

~~Additi~~

$\sigma_{A=a \wedge F=a}(R \times S)$

A	B	C	D	E	F
a	b	c	b	g	a

$\Pi_{B,E}(R \times S)$

B	E	
b	g	
b	a	} 4 distinct records
a	g	
a	a	
b	g	} Duplicates removed
b	a	

Additional Operations :-

(Algebraic operations)

These operations can be expressed in terms of basic operations.

- → Intersection (∩)
- → Quotient (÷)
- (※) → θ - Join, → Equi - Join, → Natural Join } (⋈)

INTERSECTION : (∩)

It is set of all tuples which are all common in both R and S.

Where R and S are relations.

Example :-

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	D	E	F
	b	g	a
	d	a	f

R ∩ S

d	a	f

→ No mod.

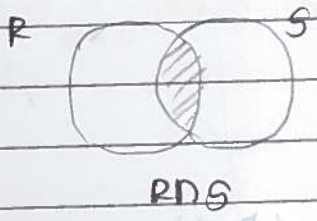
Note :

R and S have same arity of attributes

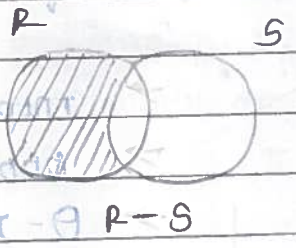
Attribute labels will not be outputted.

Intersection can be expressed as follows :-

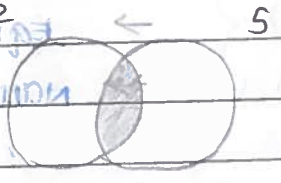
$R \cap S = R - (R - S)$



R ∩ S



R − S



R − (R − S)

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

Hence proved.

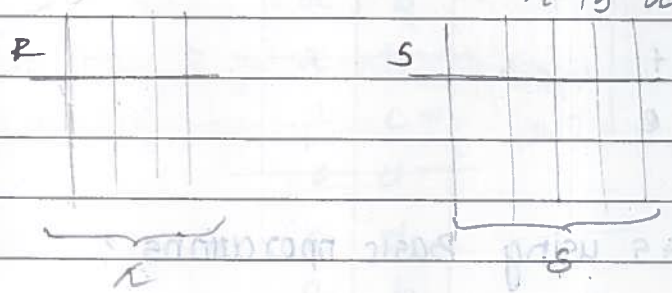
QUOTIENT : (÷)

Let R be a relation of arity r
S be a relation of arity s
where r > s → (*)

R & S can have different arity

R ÷ S is the set of r-s tuples t such that ∀ s tuples u in S, tuple t.u is in R

↳ t is a prefix



Example :

Let R be a relation.

R	1	2	3	4
	a	b	c	d
	a	b	e	f
	b	c	e	f
	e	d	c	d
	e	d	e	f
	a	b	d	e

Arity(R) = r = 4

S	1	2
a/b	c	d
a/b	e	f

Arity(S) = s = 2.

r > s → we can do quotient otherwise not possible.

R ÷ S → No need

a	b
e	d

Quotient operation expressed as basic operations :

$$R \div S = \Pi_{1,2,\ldots,r-s}(R) - \Pi_{1,2,\ldots r-s}\left[(\Pi_{1,2,\ldots,r-s}(R) \times S) - R\right]$$

④ ③ ① ②

common prefix that is available in R for all tuples available in S.

15/11

Additional operations:

~~θ JOIN~~

Example:

For the following relation R and S

R	1	2	3	4
	a	b	c	d
	a	b	e	f
	b	c	e	f
	e	d	c	d
	e	d	e	f
	a	b	d	e

S	1	2
	c	d
	e	f

Obtain R ÷ S using Basic operations:

$$R \div S = \pi_{1,2}(R) - \pi_{1,2}\left[(\pi_{1,2}(R) \times S) - R\right]$$

④ ③ ① ②

$\pi_{1,2}(R)$

1	2
a	b
a	b
b	c
e	d
e	d
a	b

$\pi_{1,2}(R) \times S$

1	2	1	2
a	b	c	d
a	b	e	f
b	c	c	d
b	c	e	f
e	d	c	d
e	d	e	f

$(\pi_{(1,2)} R \times S) - R$

b	c	c	d

$\pi_{1,2}\left[(\pi_{1,2}(R) \times S) - R\right]$

1	2
b	c

$\pi_{1,2}(R)$

1	2
a	b
a	b
b	c
e	d
e	d
a	b

$\pi_{1,2}(R) - \pi_{1,2}\left[(\pi_{1,2}(R) \times S) - R\right]$

1	2
e	d
a	b

θ- JOIN :- (⋈)

θ- Join is written as $R \underset{i\theta j}{\bowtie} S$.

where R, S are relations & i, j are columns (attributes) and θ is a arithmatic comparisson operator namely $<, <=, >, >=, =, \neq$.

θ-Join of R and S are those tuples in the cartesian product of R and S, such that the i^{th} component (column / attribute) of R stands in relation θ to the j^{th} component (column / attribute) of S

(i) cartesian product R×S
(ii) comparison $i\theta j$

The expression of θ Join in terms of Basic operation is given by.

$$\sigma_{i\theta j}(R \times S)$$

Example:

Let R and S be the relation

R	A	B	C
	1	2	3
	4	5	6
	7	8	9

S	D	E
	3	1
	6	2

Find : $R \underset{B<D}{\bowtie} S$

R × S :

R×S	A	B	C	D	E
	1	2	3	3	1
	1	2	3	6	2
	4	5	6	3	1
	4	5	6	6	2
	7	8	9	3	1
	7	8	9	6	2

$R \bowtie_{B<D} S$

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

$\sigma_{B<D}(R \times S)$

EQUI-JOIN : $(\bowtie)$

It is a special case of θ-join when θ is the operator = then is is called Equi-join.

Example :

$R \bowtie S$

$i \Rightarrow A = E \leftarrow j$

θ

$R \bowtie_{A=E} S$

A	B	C	D	E
1	2	3	3	1

NATURAL JOIN : $(\bowtie)$

$R \bowtie S$ called as Natural Join is applicable only when both R and S have the same (common) attributes. (※)

$R \bowtie S$ is obtained by

(i) compute $R \times S$

(ii) select those tuples whose values are the same in the common attributes of R and S.

The expression of Natural join in terms of Basic operation is given by:

$$\Pi_{i_1, i_2, \ldots i_m} \left(\sigma_{(R.A_1 = S.A_1) \wedge (R.A_2 = S.A_2) \wedge \ldots \wedge (R.A_n = S.A_n)} (R \times S) \right)$$

Where $A_1, A_2, \ldots A_n$ are common attributes of R and S.

$i_1, i_2, \ldots, i_m$ are distinct columns of $R \times S$ except $S.A_1 \ldots S.A_n$

Find the Natural join of Relations R and S

R	A	B	C
	a	b	c
	d	b	c
	b	b	f
	c	a	d

S	B	C	D
	b	c	d
	b	c	e
	a	d	b

R ⋈ S

R X S	A	B	C	B	C	D
	a	b	c	b	c	d
	a	b	c	b	c	e
	a	b	c	a	d	b
	d	b	c	b	c	d
	d	b	c	b	c	e
	d	b	c	a	d	b
	b	b	f	b	c	d
	b	b	f	b	c	e
	b	b	f	a	d	b
	c	a	d	b	c	d
	c	a	d	b	c	e
	c	a	d	a	d	b

$\sigma_{R.B=S.B \wedge R.C=S.C.}$	A	B	C	B	C	D
	a	b	c	b	c	d
	a	b	c	b	c	e
	d	b	c	b	c	d
	d	b	c	b	c	e
	c	a	d	a	d	b

$\Pi_{A,B,C,D}$

A	B	C	D
a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

17/11

APPLICATION PROBLEMS OF RELATIONAL ALGEBRA :-

(1) Given the following relation INSTRUCTOR (ID, NAME, DEPT, SALARY), write Relational Algebra queries for the following.

INSTRUCTOR

ID	NAME	DEPT	SALARY
10114	Sri	CS	65000
2623	Wu	Finance	90000
5678	Moz	Music	30000
2345	Ravi	Music	32000
2189	Sandy	Finance	198000

(ii) List the names of instructors and their salaries. Their IDs must occur first.

$\Pi_{ID, NAME, SALARY}$ (INSTRUCTOR)

ID	NAME	SALARY
10114	Sri	65000
2623	Wu	90000
5678	Moz	30000
2345	Ravi	32000
2189	Sandy	198000

(ii) List the ID's and names of all instructors working in the Music department.

$$\Pi_{ID, NAME} \left(\sigma_{DEPT = "MUSIC"} (INSTRUCTOR) \right)$$

ID	NAME
5678	Moz
2345	Ravi

(OR) Step by step :

$$TEMP = \sigma_{DEPT = "MUSIC"} (INSTRUCTOR)$$

$$RESULT = \Pi_{ID, NAME} (TEMP)$$

(iii) Who are the instructors working in the Finance department and earning a salary greater than Rs 30000?

$$\Pi_{NAME} \left(\sigma_{DEPT = "FINANCE" \wedge SALARY > 30000} (INSTRUCTOR) \right)$$

NAME
Wu
Sandy

(2) In a juice parlour enterprise which serves fruit juices and drinks of different types, viz, milk shakes, soda base, plain juices etc., to its clients through its different parlours in various parts of the cities in India, with different serving times (morning / evening), the following relations are available.

FREQUENTS (CLIENT NAME, JUICE PARLOUR NAME, LOCATION, CITY)

RED - Relation Name

PURPLE - Primary key

BLACK - non key attribute

SERVES (JUICE PARLOUR NAME, FRUIT DRINK NAME, SERVING TIME)

LIKES (CLIENT NAME, FRUIT DRINK NAME)

FRUIT DRINK (FRUITDRINK NAME, MAIN FRUIT, ADDED FRUIT, TYPE)

(i) List all juice parlours frequented by clients in Mumbai city.

$$\Pi_{\text{JUICE PARLOUR NAME}} (\sigma_{\text{CITY} = \text{"Mumbai"}} (\text{FREQUENTS}))$$

(ii) What are the favourite drinks of clients who frequent the juice parlour in "M G Road" of Bangalore.

$$\text{TEMP1} = \sigma_{\text{LOCATION} = \text{"MGRoad"} \wedge \text{CITY} = \text{"BANGALORE"}} (\text{FREQUENTS})$$

RESULT =

$$\text{TEMP2} = \text{TEMP1} \bowtie \text{LIKES}$$

$$\text{RESULT} = \Pi_{\text{FRUIT DRINK NAME}} (\text{TEMP2})$$

(iii) List all juice parlours where a client "Prince charles can have the fruit drinks that he likes.

$$\Pi_{\text{JUICE PARLOUR NAME}} (\Pi_{\text{FRUIT DRINK NAME}} (\sigma_{\text{CLIENT NAME} = \text{"Prince charles"}} (\text{LIKES})) \bowtie \text{SERVES})$$

(ii) List the ID's and names of all instructors working in the Music department.

$$\Pi_{ID, NAME} \left(\sigma_{DEPT = "MUSIC"} (INSTRUCTOR) \right)$$

ID	NAME
5678	Moz
2345	Ravi

(OR) Step by step :

$$TEMP = \sigma_{DEPT = "MUSIC"} (INSTRUCTOR)$$

$$RESULT = \Pi_{ID, NAME} (TEMP)$$

(iii) Who are the instructors working in the Finance department and earning a salary greater than Rs 30000?

$$\Pi_{NAME} \left(\sigma_{DEPT = "FINANCE" \wedge SALARY > 30000} (INSTRUCTOR) \right)$$

NAME
Wu
Sandy

(2) In a juice parlour enterprise which serves fruit juices and drinks of different types, viz, milk shakes, soda base, plain juices etc., to its clients through its different parlours in various parts of the cities in India, with different serving times (morning / evening), the following relations are available.

FREQUENTS (CLIENT NAME, JUICE PARLOUR NAME, LOCATION, CITY)

Red - Relation Name

Purple - primary key

Black - non key attribute

SERVES (JUICE PARLOUR NAME, FRUIT DRINK NAME, SERVING TIME)

LIKES (CLIENT NAME, FRUIT DRINK NAME)

FRUIT DRINK (FRUITDRINK NAME, MAIN FRUIT, ADDED FRUIT, TYPE)

(i) List all juice parlours frequented by clients in Mumbai city.

$$\Pi_{\text{JUICE PARLOUR NAME}} (\sigma_{\text{CITY} = \text{"Mumbai"}} (\text{FREQUENTS}))$$

(ii) What are the favourite drinks of clients who frequent the juice parlour in "M G Road" of Bangalore.

$$\text{TEMP1} = \sigma_{\text{LOCATION} = \text{"MG Road"} \wedge \text{CITY} = \text{"BANGALORE"}} (\text{FREQUENTS})$$

RESULT =

$$\text{TEMP2} = \text{TEMP1} \bowtie \text{LIKES}$$

$$\text{RESULT} = \Pi_{\text{FRUIT DRINK NAME}} (\text{TEMP2})$$

(iii) List all juice parlours where a client "Prince charles" can have the fruit drinks that he likes.

$$\Pi_{\text{JUICE PARLOUR NAME}} \left(\Pi_{\text{FRUIT DRINK NAME}} \left(\sigma_{\text{CLIENTNAME} = \text{"Prince charles"}} (\text{LIKES}) \right) \bowtie \text{SERVES} \right)$$

(iv) List all juice parlours that serve milk shake type Fruit drinks with apple as the main fruit in the mornings.

$$Temp1 = \sigma_{TYPE = \text{"Milkshake"} \wedge \text{MAIN FRUIT} = \text{"APPLE"}} (FRUIT\ DRINK)$$

$$Temp2 = \Pi_{FRUIT\ DRINK\ NAME} (Temp1)$$

$$Temp3 = Temp2 \bowtie SERVES$$

$$Temp4 = \sigma_{SERVING\ TIME = \text{"Morning"}} (Temp3)$$

$$Result = \Pi_{JUICE\ PARLOUR\ NAME} (Temp4)$$

(v) Who are the clients, who like Fruit Drinks with Mango as the main fruit and Apple as the added fruit.

$$Temp1 = \sigma_{\text{MAIN FRUIT} = \text{"Mango"} \wedge \text{ADDED FRUIT} = \text{"Apple"}} (FRUIT\ DRINK)$$

$$Temp2 = LIKES \bowtie Temp1$$

$$result = \Pi_{CLIENT\ NAME} (Temp2)$$

21/11/22

Scribe Date __/__/__ Page 85

Relational Calculus :-

↳ calculi → computing

Relational computing

INGRES - RDBMS proposed by University of California Berkeley.

IBM RDBMS ≈ System R.

Interactive Graphics and Retrieval System.

Query Language : QUEL

built on

completely ↑ Relational calculus

set theory :- {a, b, c, d, e} → set → collection of elements.

set Builder notation :-

$A = \{x \mid x \geq 5\}$

set of all elements such that $x \geq 5$

Relational calculus

- Tuple Relational calculus (TRC)
- Domain Relational calculus (DRC)

Tuple Relational calculus (TRC) :

TRC has a structure $\{t \mid P(t)\}$ where t denotes tuple and P(t) denotes predicate (condition) which should be satisfied by the tuples t.

Definition of TRC is as follows :

A TRC Formula is built from Atoms.

An Atom is

A1 : $x \in R$ where x is a tuple variable and R is a relation.

A2 : $x \theta y$ or $x \theta c$ where (x, y) are domain compatible variables, c is a domain compatible constant and θ is a arithmetic operators (<, >, <=, >=, ==, !=)

A well formed Formula (WFF) in TRC is built from Atoms using the following Rules :

B1 : An atom is a Formula

B2 : If f, g Formulae then

$\neg f$, $f \vee g$, $f \wedge g$, $f \Rightarrow g$, $f = g$.

not or and implies (if ... then) equality.

B3 : If $f(x)$ is a Formula then

$\exists x f(x)$, $\forall x f(x)$ are also Formula and

there exist for all

in such a case x is a bound variable.

Notations :-

TRC makes use of following notations :

1. $\{t \mid P(t)\}$
2. $t[A]$ A - Attribute $t[A]$ denotes the value of tuple attribute A.
3. $t \in R$
4. $\exists \in P(t)$
5. $\forall \in P(t)$
6. $\vee, \wedge, >, >=, <, <=, ==, !=, =, \neg, \Rightarrow$

Problem :-

1. Given the following tables frame the query using tuple Relational calculus :-

CUSTOMER (customer_name, street, city)

Br_name - Branch name

BORROW (Br_name, loan #, customer_name, Loan_amt)

DEPOSIT (Br_name, a/c#, customer_name, Balance_amt)

(i) Find the branch name, loan#, customer name & loan amount for all loans > 10L rupees.

$$\{t \mid t \in BORROW \wedge t[loan\_amount] > 10,00,000\}$$

(ii) Find all customers (complete details) from Mumbai city who have availed a loan of 20 Lakh and above

$$\{t \mid t \in CUSTOMER \wedge t[city] = \text{"Mumbai"} \wedge \exists u (u \in BORROW \wedge u[loan\_amount] > 20,00,000 \wedge t[customer\_name] = u[customer\_name])\}$$

22/11/22

Domain relational calculus :: (DRC)

DRC has centers on the Domain variables and therefore has a structure

$\{ \langle x_1, x_2, \ldots, x_n \rangle \mid F(x_1, x_2, \ldots, x_n) \}$

$\langle \rangle \rightarrow$ angular brackets

where $\langle x_1, x_2, \ldots, x_n \rangle$ represent a set of domain variables (attribute name) and $F(x_1, x_2, \ldots x_n)$ is a formula.

DRC formulas are built from Atoms.

An Atom is

A1 : $x \in R$ where

A2: $x \theta y$ or $x \theta c$

where θ is a comparison operator, x and y are domain compatible variables, c is a domain compatible constant.

A well formed formula (WFF) in DRC are built from Atoms using the following rules :-

B1 : An atom is a formula

B2 : If f and g are formulae then

$\neg f$, $f \vee g$, $f \wedge g$, $f \Rightarrow g$, $f = g$

B3 : If $f(x)$ is a formula then

$\exists x f(x)$, $\forall x f(x)$, are also formula and in such a case x is a bound of variable.

Given the following database schema; answer the following queries using domain relational calculus.

PRODUCT (Manufacturer, Model, PType)

PC (Model, Speed, RAM, Harddisk, CD, price)

PType refers to product type which is PC / LAPTOP / Printer

(i)

Find all Manufacturers of printers.

$\{ \langle ma \rangle \mid \exists \langle ma, m, pt \rangle \{ \langle ma, m, pt \rangle \in PRODUCT \wedge pt = \text{"Printer"} \} \}$

(ii) Find the model number, speed and Harddisk size for all PC's who's price is under 70,000 ₹

$\{ \langle m, s, h \rangle \mid \exists \langle m, s, r, h, c, p \rangle \{ \langle m, s, r, h, c, p \rangle \in PC \wedge p \leq 70,000 \} \}$

Frame domain relational calculus for (ii) in Banking Enterprice.

$\{ \langle cu, s, ci \rangle \mid \exists \langle cu, s, ci \rangle \{ \langle cu, s, ci \rangle \in CUSTOMER \wedge ci = \text{"Mumbai"} \wedge \exists \langle b, l, c, la \rangle \{ \langle b, l, c, la \rangle \in BORROW \wedge la \geq 20,00,000 \wedge c = cu \} \} \}$

For the following database schema shown below associated with computer sales express the following queries using the concepts labelled as A/B.

PRODUCT (Manufacturer, Model, PType)

PC (Model, Speed, RAM, Harddisk, CD, PRICE)

NOTEBOOK (Model, Speed, RAM, Harddisk, screen_size, price)

PRINTER (Model, printer_type, colour, price)

(i) Find the Model, RAM and screen size for Notebooks costing more than 1,00,000.

A: Relational Algebra

B: TRC

(ii) Find all available brands of Printers and list their models

A: Relational Algebra

B: DRC

(iii) Find the Model, speed and HD for all PC's who's price is under 40,000

A: Relational Algebra

B: TRC

(iv) Find all laser printers manufactured by 'HP' which prints in colour and who's price is less than 60,000

A: TRC

B: Relational algebra

(v) Write SQL query for all the above (iv) questions

Soln:

(i) Relational Algebra:

$$Temp1 = \sigma_{price > 100000}(NOTEBOOK)$$

$$Result = \Pi_{Model, RAM, screensize}(Temp1)$$

Tuple Relational calculus (TRC):

$$\{ t[model], t[RAM], t[screensize] \mid t \in NOTEBOOK \wedge t[price] >= 100000 \}$$

Domain Relational calculus (DRC):

$$\{ \langle m, r, ss \rangle \mid \exists \langle m, s, r, h, ss, p \rangle \{ \langle m, s, r, h, ss, p \rangle \in NOTEBOOK \wedge p >= 100000 \} \}$$

(ii) Relational Algebra:

$$\Pi_{Manufacturer, Model}(\sigma_{Ptype = "printer"}(PRODUCT))$$

Tuple Relational calculus (TRC):

$$\{ t[Manufacturer], t[Model] \mid t \in PRODUCT \wedge t[Ptype] = 'printer' \}$$

Domain Relational calculus (DRC):

$$\{ \langle ma, mo \rangle \mid \exists \langle ma, mo, Pt \rangle \{ \langle ma, mo, Pt \rangle \in PRODUCT \wedge Pt = "printer" \} \}$$

(iii) Relational Algebra :-

$$Temp\ 1 = \sigma_{price <= 40000}(PC)$$

$$Result = \Pi_{model, speed, HardDisk}(Temp\ 1)$$

Tuple Relational calculus (TRC) :-

$$\{ t[model], t[speed], t[HardDisk] \mid t \in PC \wedge t[price] <= 40000 \}$$

domain Relational calculus (DRC) :-

$$\{ <m, s, hd> \mid \exists <m, s, r, hd, cd, p> \{ <m, s, r, hd, cd, p> \in PC \wedge p <= 40000 \}$$

(iv) Relational Algebra :-

$$Temp\ 1 = \sigma_{printerType = "Laser" \wedge color = "color" \wedge price <= 60000}(PRINTER)$$

$$Temp\ 2 = PRODUCT \bowtie Temp\ 1$$

$$Temp\ 3 = \sigma_{manufacturer = "HP"}(Temp\ 2)$$

$$Result = \Pi_{model}(Temp\ 3)$$

Tuple Relational calculus (TRC) :-

$$\{ t[model] \mid t \in PRINTER \wedge t[printer\_type] = "Laser" \wedge t[color] = "color" \wedge t[price] <= 60000 \wedge \exists u (u \in PRODUCT \wedge u[manufacturer] = "HP" \wedge t[model] = u[model]) \}$$

∧ m = "HP"

Domain Relational calculus (DRC) :-

$\{ \langle m \rangle \mid \exists \langle m, md, pt \rangle [\langle m, md, pt \rangle \in PRODUCT \wedge \exists \langle md \rangle \exists \langle md, pt, c, p \rangle \{ \langle m, pt, c, p \rangle \in PRINTER \wedge pt = \text{"Laser"} \wedge c = \text{"color"} \wedge p <= 60000 \wedge m = md \}] \}$

(v) SQL Queries :- Structured Query Language

(i) Select Model, RAM, screensize from NOTEBOOK
where price >= 100000 ;

(ii) Select Manufacturer, Model from Product
where ptype = 'Printer' ;

(iii) Select Model, speed, Harddisk from PC
where Price <= 40000 ;

(iv) Select P.model from PRODUCT P, PRINTER PR
where P.Model = PR.Model AND
PR.Printer.type = "Laser" AND
PR.color = "color" AND PR.Price <= 6000;

29/11/22

CO4

Database design theory :- Functional dependencies. Normal forms - Normalization : INF to 5NF - Domain key Normal Form - lossless join and dependency preserving decomposition - Denormalization - Database tuning. Database transaction and security : transaction processing - properties - security and integrity threats - security violations - identification and authentication - discretionary access control based on grant and revoking privilege - mandatory control and role based access control :

DATABASE DESIGN THEORY.

Functional dependencies :- (FD)

Syntax : $X \rightarrow Y$

relation/table

X - collection of attributes

Y - collection of attributes

Definition : If two tuples t_1, t_2 of relation R have the same values for the X attributes then if the corresponding values of the Y attributes are identical (same) then $X \rightarrow Y$ it is pronounced as X functionally determines Y (or) X determines Y.

Example :

consider the following relation

A	B	C
a_1	b_1	c_2
a_2	b_2	c_1
a_3	b_1	c_2
a_3	b_3	c_3
a_1	b_2	c_1
a_4	b_4	c_4
a_3	b_2	c_1
a_5	b_4	c_4

For all the tuples at first instance B and C takes some value as b_1, c_2 in all the cases when B takes $b_1 \Rightarrow$ C should only have c_2. similarly for all the cases we have to check.

In the above example :-

$B \rightarrow C$

$B \not\rightarrow A$ → doesn't determines

$AC \rightarrow B$

combination of attributes can also be possible.

Trivial Functional dependency :-

An FD, $X \rightarrow Y$ is said to be trivial if $Y \subseteq X$.

Axioms of Inference for FD :

Given R to be a relation and X, Y, Z, W to be subsets of R. The following axioms hold good. Here $\models$ means "logically implies".

F1: Reflexivity.

$X \rightarrow X$

F2: Augmentation. (in addition)

$X \rightarrow Y \models XZ \rightarrow Y$ and $XZ \rightarrow YZ$

F3: Transitivity

$X \rightarrow Y$ and $Y \rightarrow Z \models X \rightarrow Z$.

F4: Additivity.

$X \rightarrow Y$ and $X \rightarrow Z \models X \rightarrow YZ$

F5: Projecting (Projectivity)

$X \rightarrow YZ \models X \rightarrow Y$ and $X \rightarrow Z$

F6: Pseudo Transitivity.

$X \rightarrow Y$ and $YZ \rightarrow W \models XZ \rightarrow W$

Axioms F1, F2, F3 are known as Armstrong's Axiom.

A	B	C	D	E
a_1	b_1	c_2	d_1	e_1
a_2	b_2	c_1	d_2	e_2
a_3	b_1	c_2	d_1	e_3
a_3	b_3	c_3	d_3	e_4
a_1	b_2	c_1	d_2	e_5
a_4	b_4	c_4	d_4	e_6
a_3	b_2	c_1	d_2	e_7
a_5	b_4	c_4	d_4	e_8

(1). Check the following:

(i) $E \Rightarrow D$? true

(ii) $D \rightarrow C \Rightarrow$? true

(2) (i) Does B determines CD? If so do they obey Projectivity Axiom:

$B \rightarrow CD$

Projectivity $\Rightarrow$ $B \rightarrow C$ and $B \rightarrow D$.

satisfies.

(ii) Test for the transitivity axiom between E, B and C.

(i) $E \rightarrow B$ and $B \rightarrow C \models E \rightarrow C$

✓ satisfies.

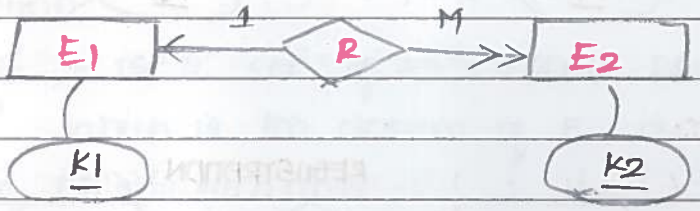
5/12/22

Scribly Date __/__/__ Page 97

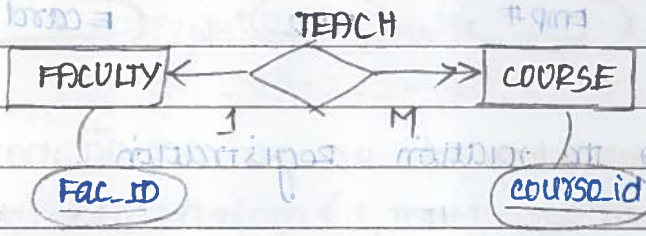
Functional dependency (FD) and Entity relationship:

Consider a relation R between two Entities E_1 and E_2.

one-many relationship.



Example:



Consider the relation teach

TEACH

FAC_ID	COURSE_ID
f_1	c_1
f_2	c_3
f_3	c_4
f_4	c_5
f_1	c_2
f_2	c_6
f_3	c_8

FACULTY $\not\rightarrow$ COURSE

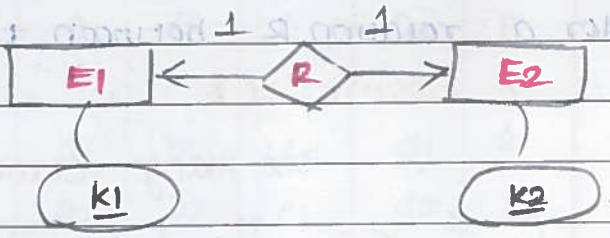
COURSE $\rightarrow$ FACULTY

When there is one-many relationship between two entities x and y.

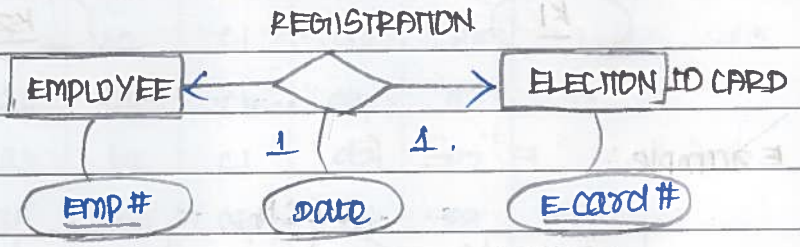
$x \rightarrow y$ does not imply $y \rightarrow x$

consider the following Relation R as follows.

one-one relationship



Example :-



consider the relation Registration.

REGISTRATION

EMP #	E-CARD #	DATE
e1	eid1	d1
e1	eid1	d1
e2	eid2	d2
e2	eid2	d2

EMPLOYEE → ELECTION ID CARD

ELECTION ID CARD → EMPLOYEE

When there is one-one relationship between two entities x and y

$X \to Y$ implies that $Y \to X$

CLOSURE OF a set of FD'S:

The set of FD's which is logically imply by F is called a closure of F and is written as F^+

definition:

If F is a set of FD's on a relation R then F^+, which is the closure of F is the smallest set of FD's such that

$$F^+ \supseteq F \quad (\supseteq \text{ : superset})$$

F^+ contains F

and no FD can be derived from F by using the inference axioms that are not contained in F^+.

Thus,

$$F^+ = \{ X \to Y \mid F \models X \to Y \}$$

Example:

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

and $F = \{ A \to B, A \to C, BC \to D \}$

Find atleast 3 members of F^+.

soln:

$$F = \{ \underset{(i)}{A \to B}, \underset{(ii)}{A \to C}, \underset{(iii)}{BC \to D} \}$$

$F^+ = F \cup \{ \{A \to BC\}$ by Additivity of (i) and (ii) $\}$

$$F^+ = \{ \underset{(i)}{A \to B}, \underset{(ii)}{A \to C}, \underset{(iii)}{BC \to D}, \underset{(iv)}{A \to BC} \}$$

$F^+ = F^+ \cup \{ \{A \to D\}$ by transitivity of (iv) and (iii) $\}$

$F^+ = \{ A \to B, A \to C, BC \to D, A \to BC, A \to D \}$

$F^+ = F^+ \cup \{ (A \to A, B \to B, C \to C, D \to D \}$ by reflexivity }

$F^+ = \{ A \to B, A \to C, BC \to D, A \to BC, A \to D,$
$A \to A, B \to B, C \to C, D \to D \}$

$F^+ = F^+ \cup \{ \{ AC \to D \}$ by pseudo transitivity of $A \to B$ and $BC \to D$ }

$F^+ = \{ A \to B, A \to C, BC \to D, A \to BC, A \to D, A \to A,$
$B \to B, C \to C, D \to D, AC \to D \}$
...

Note:
It appears that F^+ can become very large but in practice the objective is not to list all the members of F^+ but only to find if a given FD $X \to Y$ can be inferred from F (i.e.,) we are interested to know $F \models X \to Y$.
↳ F^+

To do this, instead of finding the entire universe of F^+ with inference axioms, we adopt a shortcut algorithm to find if $F \models X \to Y$

Algorithm to compute closure:

Computing F^+ is a lengthy process, however the practical need arises only to know if $F \models X \to Y$ this is true only when $X \to Y \in F^+$.
However to find this we adopt an alternative method as given below, which depends on generating X^+, which is the closure of X, under F.

6/12/22

Anomalies.

to eliminate it drawbacks → Normalization.

Definition :-

the closure of X under F, a set of FD's, written as X^+ is the set of attributes $\{A_1, A_2, \ldots, A_m\}$ of the relation R such that the FD $X \to A_i$ for $A_i \in X^+$ follows from F by the inference axioms for FD's.

> To test if $F \models X \to Y$, find X^+ and check if $Y \subseteq X^+$

Computation of X^+ (work rule):-

To find X^+

(i) start with X^+ initialize to X.
which is the LHS of $X \to Y$, that is to be tested

(ii) For each FD, $W \to Z$ in F
if $W \subseteq X^+$, modify X^+ as a union of X^+ and Z. $(X^+ \cup Z)$

(iii) terminate the computation process when there is no change in X^+.

Example :-

consider a relation R(A, B, C, D, E, H) with

$$F = \left\{ \begin{array}{ll} A \to BC, & ABH \to BD \\ CD \to E, & DH \to BC \\ E \to C, & \\ D \to ABH, & \end{array} \right\} \quad X = BCD$$

AEH

Let $X = BCD$ Find X^+ ?

Soln :- closure $X^+ = ABCDEH$

$X^+ = X$

$X^+ = (BCD)$ step (i)

$X^+ = (BCDE)$ $(BCD) \cup (E)$ $\because CD \to E$ step (ii)

$X^+ = (ABCDEH)$ $(BCDE) \cup (AEH)$ $\because D \to AEH$ step (ii)

Process terminates, no more attributes can be added to X^+

Find x^+. (i) $X = E$

(ii) $X = AEH$

(i) $X = E$

$x^+ = E$ Step (i)

$x^+ = (E) \cup (C) = EC$ Step (ii) $E \rightarrow C$

process terminates no more attribute can be added to x^+

(ii) $X = AEH$

$x^+ = AEH$ Step (i)

$x^+ = ABCEH$ $(AEH) \cup (BC)$: Step (ii) $A \rightarrow BC$

$x^+ = ABCDEH$ $(ABCEH) \cup (BD)$: Step (iii) $ABH \rightarrow BD$

process terminates.

all the attributes of the relation R were added to x^+. No more attributes can be added into x^+

Example 2:

[Test if $BCD \rightarrow H$ is inferable in relation R using F given in Example 1] $\left(\begin{array}{l} X = BCD \\ x^+ = ABCDEH \end{array} \right)$

(closure of x)

$H \in x^+ \Rightarrow$ yes

Soln:

(i) To test this Find x^+ for $X = BCD$

(ii) check H is a subset of x^+, $BCD \rightarrow H$.

soln:

(i) $x^+ = ABCDEH$ (as shown in Example 1).

(ii) $H \subseteq x^+$

From this H is a subset of x^+.

$BCD \rightarrow H$ is inferable from x.

Given a relation R(X, Y, Z, H, C, E, G) with

$$F = \begin{cases} X \to YZ \\ XZH \to ZE \\ ZE \to G \\ E \to XGH \end{cases}$$

(i) Test if $XZ \to G$ is inferable from F

(ii) Test if $F \models XH \to GH$?

(i) $A = XZ$

$A^+ = XZ$ Step (i)

$A^+ = XYZ$ $(XZ) \cup (YZ)$ $X \to YZ$ Step (ii)

Process terminates

$XZ \to G$

$G \notin A^+$ $\therefore XZ \to G$ is not inferable from F.

(ii) $A = XH$

$A^+ = XH$ Step (i)

$A^+ = XHZY$ $(XH) \cup (YZ)$ $X \to YZ$ Step (ii)

$A^+ = XHZEY$ $(XHZY) \cup (ZE)$ $XZH \to ZE$ Step (ii)

$A^+ = XHZEGY$ $(XHZEY) \cup (G)$ $ZE \to G$ Step (ii)

$A^+ = XHZEGY$ $(XHZEYG) \cup (XGH)$ $E \to XGH$ Step (ii)

Process terminates, no more attributes can be added to A^+

$F \models XH \to GH$

$GH \subseteq A^+$ ✓

$\therefore XH \to GH$ is inferable from F

$F \models XH \to GH$ ✓

Functional dependencies and keys:

Given a relation R $(A_1, A_2, \ldots, A_n)$ and a set of FD's F, we need to test if a set of attributes are the right choice to behave as a key K for the relation R

12/12/22

(Scientific method of checking Primary key)

Testing for a key requires the following to be true:

(i) key K should have unique identification property.

(ii) key K should be minimal (non redundancy)

To understand the above, the following rules should be satisfied

(i) If $R(A_1, A_2, \ldots, A_n)$ is a relation with a set of FD's F and $K = A_1, A_2, \ldots, A_k$ then to test if K is a key

> Test if $K \to A_1, A_2, \ldots A_n \in F^+$
> (i.e.,) Find if $K^+ = A_1, A_2, \ldots, A_n$

(ii)

{a, b, c}

↓

Proper subsets.

{a, b, c,

ab, bc, ac}

> Find $Y \subset K$ for all Y (Y: proper subset)
> and show that
> $Y \to A_1, A_2, \ldots, A_n \in F^+$
> (i.e.), compute Y^+ and show
> that $Y^+ \neq A_1, A_2, \ldots, A_n$

Important Note:

Most authors are satisfied with keys to have the unique identification property and therefore, tends to ignore the property of minimality.

In such a case a superkey (a superset of K is called super key). A superkey and a key is the same.

But ullman, he claims that a key should be minimal. Therefore, according to ullman superkey ≠ key.

Example:

Given $R(A,B,C,D,E,H)$ and

$$F = \begin{cases} A \to BC \\ CD \to E \\ E \to C \\ D \to AEH \\ ABH \to BD \\ DH \to BC \end{cases}$$

Find whether $K = (C,D)$ is a key of R?

To test if $K = CD$ is a key

(i) Unique identification.

Find if $K^+ = (CD)^+ = (A,B,C,D,E,H)$

compute $(CD)^+$

$= CD$

$= (CD) \cup (AEH)$

$= ACDEH$

$= (ACDEH) \cup (BC)$

$(CD)^+ = ABCDEH$

CD uniquely identifies the relation R.

property (i) satisfied.

To test if K satisfies minimality. CD → proper subsets {C} {D}

(ii) Find $Y \subset K = CD$

$Y = \{C\}$

compute $Y^+ = C^+$

show whether $C^+ \neq ABCDEH$.

$C^+ = C$ $\quad C \neq ABCDEH$ ✓

$Y = \{D\}$

compute $Y^+ = D^+$

show whether $D^+ \neq ABCDEH$

$D^+ = D$

$= D \cup AEH$

$= ADEH$

$= ADEH \cup BC$

$D^+ = ABCDEH$ ✗ $D^+ \neq ABCDEH$.

condition not satisfied.

CD doesnot satisfies the property of minimality.

CD satisfies the property of unique identification but doesn't satisfy the property of minimality.

2. For the same relation R and FD's

Find whether

(i) BC a candidate key, show how?

(ii) AD ~~is a~~ and ~~EB~~ keys are super keys according to uiman.

soln:

(i) BC - a candidate key?

unique identification:

$K^+ = (BC)^+$

$= BC \notin ABCDEH$

∴ BC does not uniquely identifies the relation R.

(ii) superkeys according to ullmann.

It should satisfy the unique identification.

Let K = AD ; test whether it satisfies unique identification.

$(K)^+ = (AD)^+$

$= AD$

$= (AD) \cup BC \quad [\because A \to BC]$

$= ABCD.$

$= ABCD \cup E \quad [\because CD \to E]$

$= ABCDE$

$= ABCDE \cup AEH \quad [\because D \to AEH]$

$= ABCDEH.$

No more attributes can be added.

Hence, AD satisfies the unique identification property then AD is a Super Key.

$(K)^+ = (EB)^+$

$= EB.$

$= EB \cup C \quad [\because E \to C]$

$= EBC \neq ABCDEH.$

No more attributes can be added.

EB does not satisfies the unique identification property. then it is not a Super Key.

15/12

Scriby Date / / Page 109

Full Functional dependency: (FFD)

Definition: Given a relational schema R and an FD: $X \to Y$, Y is a Fully Functionally dependend on X.

If there is no Z, where Z is a proper subset of X, such that Z determines Y.

$Z \to Y$

Example:

Let R be a relation (ABCDEH)

$$F = \begin{cases} A \to BC \\ CD \to E \\ E \to C \\ CD \to AH \\ ABH \to BD \\ DH \to BC \end{cases}$$

Test if (i) $A \to BC$ and

(ii) $ABH \to BD$ are Fully Functional Dependencies?

(i) $A \to BC$

To check if $A \to BC$ is an FFD?

$A \to BC$ ↳ no subsets

Since, the proper subsets of A is null, we have $A \to BC$ to be an FFD.

(ii) $ABH \to BD$

To check if $ABH \to D$ is an FFD?

Proper subsets $ABH = \{A, B, H, AB, AH, BH\}$

(i) Test if $A \to D$

$A^+ = A \Rightarrow ABC \Rightarrow A^+ = ABC$

D is not a subset of $A^+ = ABC$.

$A \not\to D$

(iii) Test if $B \to D$

$B^+ = B \Rightarrow B^+ = B$

D is not a subset of B^+.

$B \not\to D$

(ii) Test if $H \to D$

$H^+ = H$

$D \notin H^+$

$H \not\to D$

(iv) Test if $AB \to D$

$(AB)^+ = AB = ABC$

D is not a subset of $(AB)^+$

$AB \not\to D$

(v) Test if $AH \to D$

$(AH)^+ = AH = ABCH = ABCDH = ABCDEH$

D is a subset of $(AH)^+$

$\therefore AH \to D$.

Since $AH \to D$, we conclude that $ABH \to BD$ is not fully functionally determines.

(vi) Test if $BH \to D$

$(BH)^+ = BH$

D is not a subset of $(BH)^+$

$BH \not\to D$

Prime & Non-Prime attribute.

An attribute A in a relation R is a prime attribute are simply prime. If A is a part of any candidate key of the relation R.

If A is not a part of any candidate key of relation R then A is called Non-prime attribute. or simply Non-prime

Partial Dependencies:

Given a relational schema R with FD's, F defined on the attributes of R and k is a candidate key, if X is a proper subset of K and if $F \models X \to A$, then A is said to be partially dependent on key K.

Example:

consider a relation R(ABCD) with FD's, F

$$F = \{ AB \to C,\ B \to D \}$$

(i) test if AB is a key K of R?

Yes, it is a key.

(ii) test if D is partially dependent on the key K=AB

ii) Test if $A \to D$ or $B \to D$

$A^+ = A$ $B^+ \to B = BD$

$D \notin A^+$ $D \subset B^+$

$A \not\to D$ $B \to D$

$B \to D$, ∴ B is partially dependent on key K.

Transitive dependency:

Given a relation scheme R with FD's, F defined on the attributes of R. Let X, Y be subsets of R and let A be an attribute of R such that $X \not\subset Y$, $A \not\subset XY$.

If the set of FD's $\{X \to Y, Y \to A\}$ is implied by F i.e., $F \models X \to Y \to A$ and $F \not\models Y \to X$ (doesn't infer) then A is transitively dependent on X.

Example:

R(ABCDE)

FD's $F = \{AB \to C, B \to D, C \to E\}$

(i) Test if AB is a key!

$K = AB \Rightarrow (AB)^+ = AB = ABC = ABCDE \Rightarrow (AB)^+ = ABCDE$ (ABCD)

$Y \subset K \Rightarrow Y = \{A\}$ $Y^+ = A^+ \Rightarrow A^+ = A$ $A^+ \neq ABCDEH$ ✓

$\Rightarrow Y = \{B\}$ $Y^+ = B^+ \Rightarrow B^+ = B = BD$ $B^+ \neq ABCDEH$ ✓

AB satisfies the properties of unique identification and the property of minimality.

(ii) Is E transitively dependent on the key K = AB.

$AB \to C \to E$

$AB \to E$ $\quad C \not\to AB$

In addition to above, we need to show that $C \not\to AB$

$C^+ = C = CE$

$AB \notin C^+$ ∴ $C \not\to AB$

19/12/22

Scribe Date / / Page 113

NORMALIZATION - Design Process

Normalization is a design process that involves FD, FFD and transitive dependancy.

Normalization is a design process where data elements are grouped into tables representing entities and the relationship.

It is used to minimize redundancy from the relation R, a set of relations.

It is used to eliminate undesirable characteristics such as Insertion, Deletion, and Update anamolies.

The process of normalization, proceeds step by step:

STEP 1 :- It consists of transforming Data items into a two dimensional table.

Definition :- In the first step, an unnormalized relation is defined as follows:

It is one in which, the relation / table contains non atomic values (i.e.,) multiple values in a single row.

(PTO)

Example :

Enterprise - Patient - surgeons - surgery - post operative drug - side effect.

step 1 :

unnormalized relation :

Patients #	surgeon's LICENSE NUMBER	surgery Date	patient Name	patient Address	surgeons name	sur
1234	243	Apr. 5. 1976	John	N. Delhi	Dr. Ram	co
	272	Nov. 4. 1999	John	N. Delhi	Dr. Ray	ge
7890	5761	Apr. 6. 2001	Sima	Pune	Dr. Lisa	
		Jan. 7. 2022	Sima	Pune	Dr. Lisa	

STEP 2 :

Unnormalized relation needs to be structured in such a way that the information stored in the table are uniquely identifiable by a primary key such a relation is called as FIRST NORMAL FORM (1NF)

Definition : A First normal Form is a relation / table in which the primary key attributes functionally determine the non key attributes (i.e.)

Prime attributes → Non Prime attributes.
(Primary key)

(First normal Form Deals with Functional Dependency (FD))

Let us choose Patient #, Surgeon's license #, surgery date as the primary key.

ve

Surgery	post operative drug.	side effects
cataract	Penicilin	Fever
gall stone	Bacterion	swelling.
Heart	Mycins	Headache
Heart	Mycins	Headache.

The 1 NF TABLE will looks like

Primary key.

patient #	surgeon's Licence number	surgery date	patient name	Address	surgeon's name	surgery
1234	243	Apr 5 1976				
1234	272	Nov 4 1999				
7890	5761	Apr. 6. 2001				and so on.
7890	5761	Jan 7 2002				

Anomalies in the 1 NF :-

Insertion Anomaly :-

A new patient who has not completed the process of diagnizing the surgery and hence for whom no surgeon has been assigned is getting admitted in the hospital.

This is an anomaly since it is not possible to enter the appropriate tuple in the table, hence it is an insertion anomaly.

Update Anomaly:

Let us suppose, a patient enters the hospital multiple times for the surgery and if between one of his visits, his address changes then update has to be done by scanning all the tuples of patient and making the appropriate changes.

this is an Update anomaly.

20/12/22

Deletion Anomaly:

Let us suppose, a patient on whom a surgery was done is deceased, then we delete the tuple(s) in the table. However, if the surgery was done by a new surgeon for whom this patient was the first client and it was the only surgery that he had done.

(this is an deletion anomaly)

then when the tuple is deleted all facts about the surgeon's can be lost.

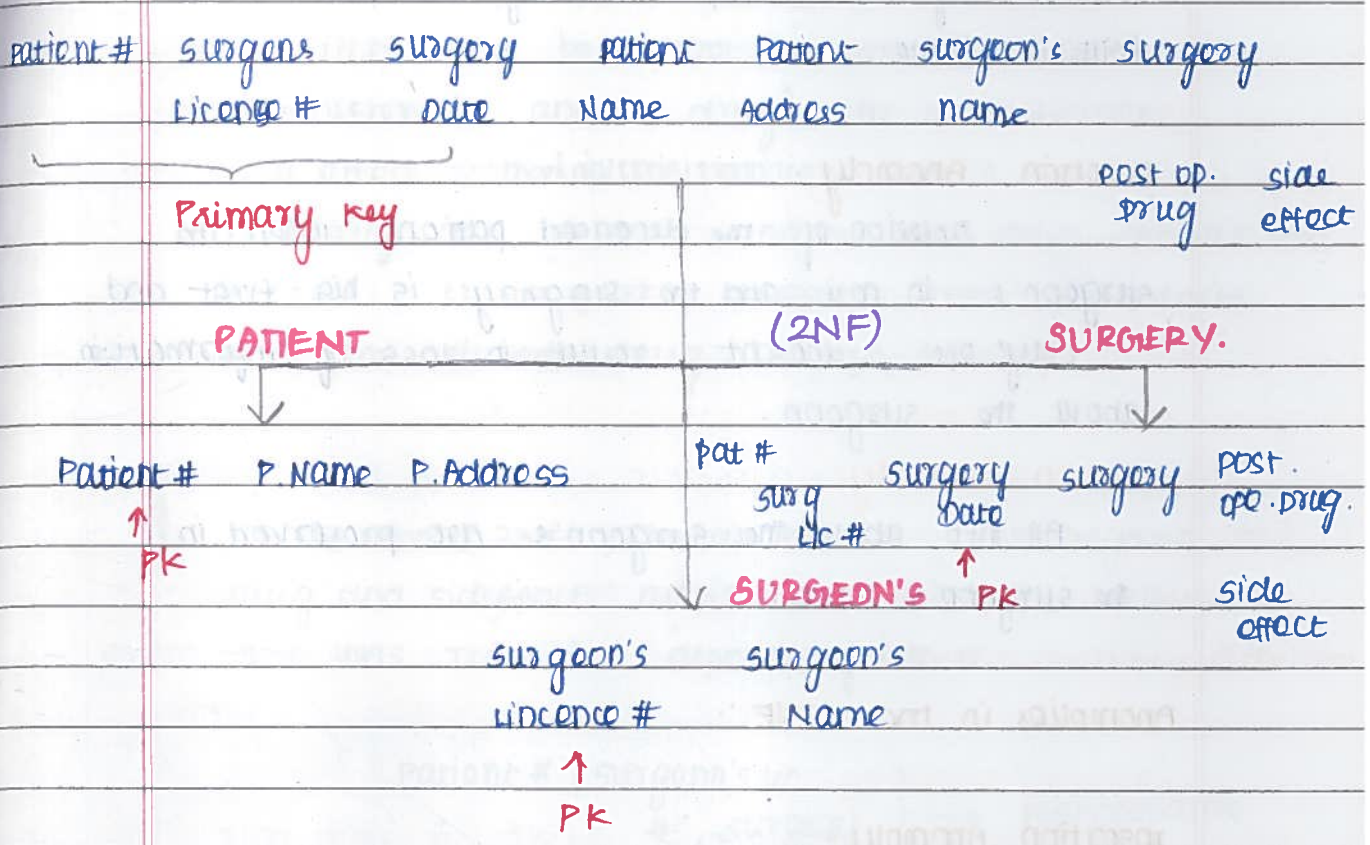
The above INF suffers from Insertion, update and deletion anomalies. Since, the non prime attributes are only Functionally dependent on the primary key but they are not Fully Functionally dependent on the primary key.

to handle this (break the table), we undertake decomposition.

↳ table will be broken out into multiple tables.

The above table is now decomposed into 3 different

tables so that, all the non prime attributes are fully functionally dependent on the prime attributes.



The above tables are in second Normal Form (2NF)

Definition:- A relation or a table are said to be in 2NF if all the non prime attributes are fully functionally dependent on the prime attributes.

Rectification of anomolies of the 1NF in the 2NF tables:-

Insertion Anomoly:- (rectification)

Details regarding the new patient for whom no surgeon's was assigned and the date of surgery was not yet fixed, can now be inserted in the PATIENT Table.

Update Anomaly :- (Rectification)

The update of patient's has to be done only in one tuple in only one place in the PATIENT table.

Deletion Anomaly :- (Rectification)

Deletion of the deceased patient when the surgeon's in row and the surgery is his first and only one, doesn't result in loss of information about the surgeon.

All info about the surgeon's are preserved in the surgeon's table.

Anomalies in the 2NF :-

Insertion Anomaly :-

The fact that a particular drug has a particular side effect cannot be recorded unless it is given to a patient for whom a surgery and a surgeon have been fixed.

Hence, no information about new drugs and their side effect can be entered.

Update Anomaly :-

If the manufacturer of the particular drug changes the formula so that the side effect of the particular drug changes, then we have to search the whole table and update the side effects of the drug in all the tuples.

Deletion Anomaly:

Assume that a patient received, a post operative drug with the particular side effect (severe fever due to penicillin) and to control the side effect, he was administered another drug with a different side effect, then the earlier details of the first drug are deleted and the new drug details are entered. It is quite possible that the details about the deleted drug are lost forever.

In the above 2NF, the anomalies with regard to drug and sideeffect attribute occurred because there was transitive dependency between non key attributes

Patient #, surgeon's lic #, surgery date } → Post operative drug.

Post operative drug } → side effect — Functionally dependent

Hence the anomalies will be removed, when there is no transitive dependency in the non prime attributes.

Patient # | surgeon's licence # | surgery date | surgery | Post operative drug | Side effect.

3NF

PATIENT- SURGERY | DRUG.

Patient # | surgeon's licence # | surgery date | surgery POD | Post operative drug | Side Effect.

26/12/2022

Scribd Date __/__/__ Page 120

Rectification of the anomalies in 2NF

Insertion Anomaly: (Rectification)

All details about new drugs and their side effects can be directly entered into the DRUG Relation. There is no need, that the drug has been administered to the patient, who's surgery has been fixed.

Update Anomaly: (Rectification)

When the side effect of the post operative drug changes, the update has to be done only in one tuple in the drug Relation, not in all the places as it was done in Earlier.

Deletion Anomaly: (Rectification)

When the patient is given a new post operative drug with its side effect then deleting the old post operative drug and side effect may lead to loss of information. However, in 3NF the drug table preserves all details about all the drugs and hence, there is no information loss.

Hence the process of Normalization Finally yields the following 4 tables.

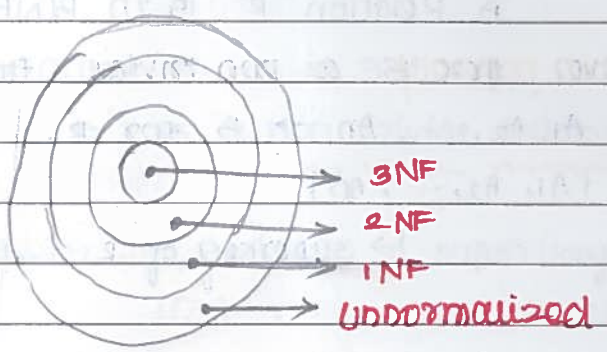
PATIENT Patient # Pat Name Pat Address

SURGERY surgeon's licence # surgeons Name

PATIENT-SURGERY Patient # surgeon's licence # surgery date surgery POD

DRUG POD side effect

summary:



1NF (table/relation has a key and obeys Functional dependency)

2NF (table/relation has Fully Functional dependency)

3NF (table/relation, does'nt have transitive dependency among the non key attributes)

BCNF (Boyce codd Normal Form) (3.5NF)
↳ Raymond Boyce
↳ SQL Design team

The goal of decomposition during normalization process involves replacing a relation with several relations so that no anomolies can occur.

Boyce-codd investigated a simple condition which can ensure that no anomolies can exist. This condition is called as Boyce-codd Normal Form.

Trivial dependency.

$X \rightarrow Y$

$\Rightarrow Y \subseteq X$.

BCNF definition:

A Relation R is in BCNF if and only if whenever there is a non trivial functional dependency (FD) $A_1, A_2, \ldots, A_n \rightarrow B$ for R.

$\{A_1, A_2, \ldots, A_n\}$

↳ superkey of R.

That is the left hand side of every non trivial FD for a BCNF should contain a key.

↳ primary

Example:

⇒ NOT BCNF

(i) MOVIE { TITLE, YEAR, LENGTH, FILMTYPE, STUDIO, STAR NAME)

key is safe but FD is not complied. { TITLE, YEAR → LENGTH, FILMTYPE, STUDIO }

(ii) MOVIE-1 (TITLE, YEAR, LENGTH, FILMTYPE, STUDIO)

BCNF. FD { TITLE, YEAR, STUDIO → LENGTH, FILMTYPE }

27/12/22

Alternative definition for 3NF :-

A relation R is in 3NF if, whenever $A_1, A_2, \ldots, A_n \rightarrow B$ is a non trivial dependency either

(i) $\{A_1, A_2, \ldots, A_n\}$ is a super key.

(OR)

(ii) B is a member of some key.

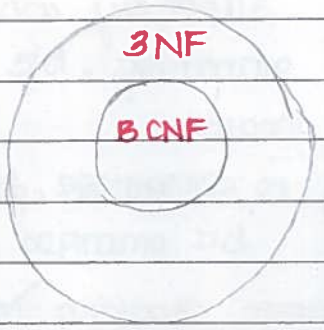
B can be a member of any candidate key.

↓

multiple attributes acts as a key.

Note the difference between 3NF and BCNF which is a condition on

B is a member of some key. In other words 3 NF is a liberalized BCNF.



Every BCNF is a 3NF but vice versa is not true.

Example :-

(i) Let R be a relation R(ABC)

FD { A→C, C→B } → Non trivial FD's

IS R in BCNF ? It is not a BCNF since the left hand side of the FD's doesn't contain the key A, B.

IS R in 3NF ?

FD's are non trivial, fails

(i) Superkey is not true. (or)

(ii) C is not a member of key AB, condition fails Hence it is not in 3NF.

(ii)

BOOKING (TITLE, CITY, THEATER)

$$FD = \left\{ \begin{array}{l} THEATER \rightarrow CITY \\ TITLE, CITY \rightarrow THEATER \end{array} \right\}$$

Also {theater, TITLE} and {title, city} are the candidate keys.

check whether

i) Booking is in 3NF (or) BCNF (or) both ?

The given FD's are non trivial.

(i) check for BCNF ?

since, in the first FD the left hand side theater is not a super key the condition fails.

∴ BOOKING is not in BCNF

(ii) check for 3NF?

(i) In the firs FD the left hand side theater is not a super key. super key condition fails

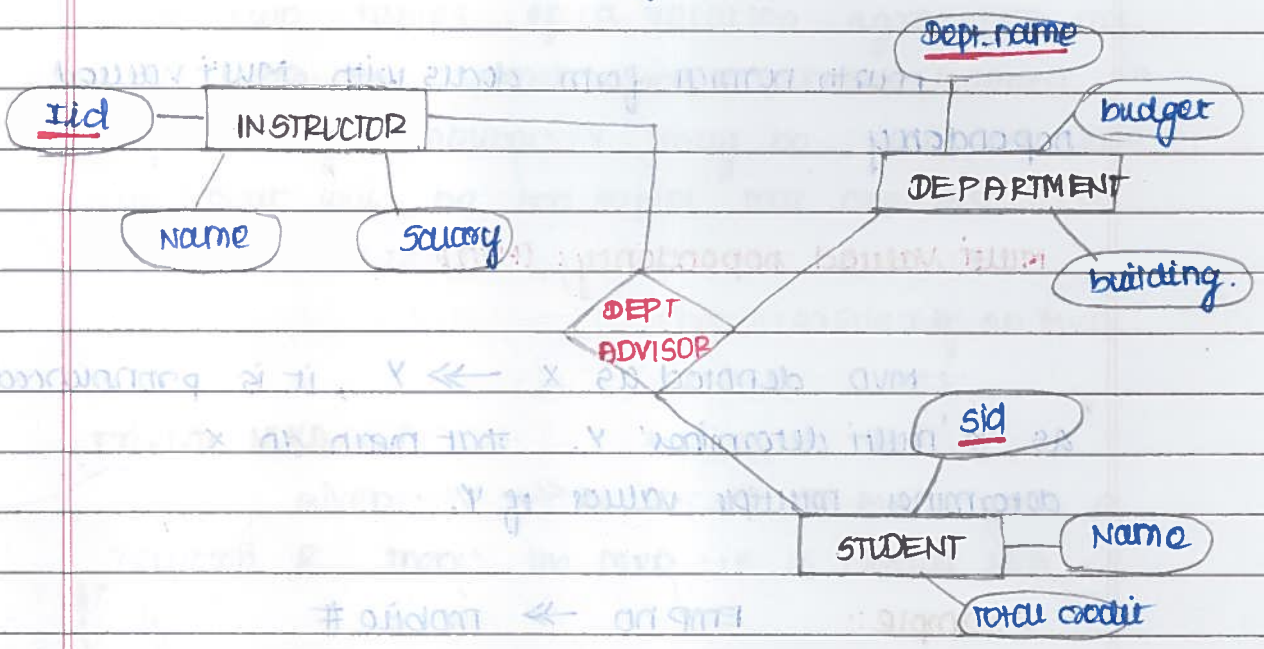
(OR)

(ii) city and theater

the RHS of FD's are a part of two candidate keys.

∴ The relation Booking is in 3NF.

(iii) Consider the following ERD.



(i) Obtain the relation for Department Advisor ?

DEPT ADVISOR {Iid, deptname, sid}

(ii)

FD { Iid → dept_name
sid, dept_name → Iid }

Test whether sid, Deptname is a super key ?

Let x = sid, Deptname

x^+ = sid, Deptname

x^+ = sid, Deptname, Iid ∵ sid, deptname → Iid

✓

Hence, it is a super key.

(iii) check whether Dept_Advisor is in 3NF ?

FD { Iid → dept_name
sid, dept_name → Iid }

The FD is in non-trivial.

(i) Iid is not a super key (ii) Iid is a part of Primary key (or) { dept_name is a part of super key

∴ It is in 3NF.

29/12/22

MVD

→ Multi-valued Dependency.

4th Normal Form: (4 NF)

Fourth normal form deals with multivalued dependency.

Multi valued dependency: (MVD)

MVD denoted as $X \twoheadrightarrow Y$, it is pronounced as X 'multi determines' Y. That mean an X determines multiple values of Y.

Example:: Emp no → mobile #

Emp no → dependents

MVD requires atleast 3 attributes (columns) because it consists of atleast two attributes that are dependent on the third.

Example:

Emp name	Project	Hobby
x	unix	music
x	oracle	reading
x	unix	reading
x	oracle	music

{ Emp Name → Project
{ Emp Name → Hobby.

This is a table with the above MVD, which can leads to inconsistency, when new details are inserted. Therefore, an MVD is described as follows:

fact

(i) An MVD $X \twoheadrightarrow Y$ is an assertion that if two tuples of a relation agree on all attributes of X then the components (values) in the set of attributes Y may be swapped and the result will be two tuples that are also in a relation.

↳ characteristic of an MVD.

Trivial MVD :-

Given $V \twoheadrightarrow W$ to be an MVD, over a relation R then the MVD is a trivial MVD

if

(i) W is a subset of V - $W \subseteq V$

(or)

(ii) $VW = R$ (union of V and W attributes equals the original relation)

Note :-

An MVD, $X \twoheadrightarrow Y$ is considered to be trivial if $Y = \emptyset$ (null set)

4th Normal Form (4NF)

Definition :-

Given a relation schema R such that the set D of FD's and MVD's are satisfied, consider a set of attributes X and Y, where

$X \subseteq R$ and $Y \subseteq R$

Relation R is in 4NF if

(i) For all MVD's of the form $X \twoheadrightarrow Y \in D^+$ either

$X \twoheadrightarrow Y$ is a trivial MVD (or)

→ FD has to be checked.

(ii) X is a superkey of R

Example:

consider the following relation R

Emp no	salary	year
E123	70K	1998
E123	80K	2006
E345	60K	1995
E345	70K	2009

$$D = \{ \text{Emp no} \twoheadrightarrow \text{salary}, \text{year} \}$$

given relation R is in 4NF form

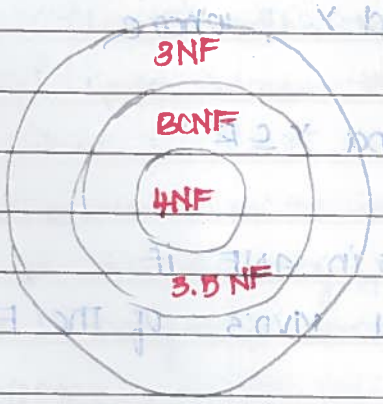
since D is a trivial MVD. → condition (i) satisfied.

$V \twoheadrightarrow W \Rightarrow VW =$ Emp no, salary, year

all the columns are given. it is trivial

since there are no FD's in D, second condition (checking of superkey) is ignored.

→ A relation R which is in 4NF is also in BCNF.

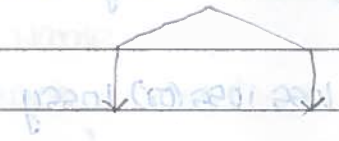


If a relation R which is in BCNF it is also in 3NF.

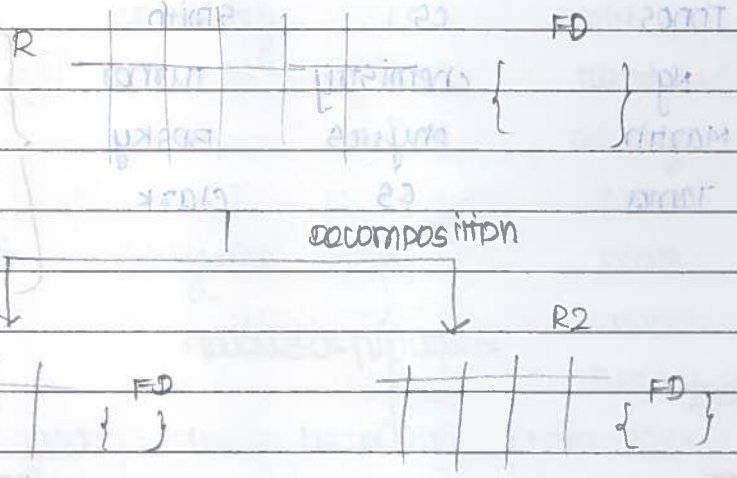
2/1/2023

Decomposition :

Two characteristics :



(lossless join)



$R_1 \bowtie R_2 = R$ lossless join.

when $R_1 \neq R_1 \bowtie R_2$ (lossy decomposition)

Lossless join & dependency preserving :

↳ (content Preserving)

Lossless join & dependency preserving are two important criteria for decomposition.

Decomposition requires that information contained in the original relation should be maintained. This in turn requires that the decomposition should be such that

(i) a join of the decomposed relation gives the same set of tuples as the original relation and

(ii) the dependencies (FDs) of the original relation are preserved.

Example:

A decomposition of a following table has been done as given.

Is the decomposition loss less (or) lossy?

(R1)

STUDENT ADVISOR

Name	Dept	Advisor
Jones	CS	Smith
Ng	chemistry	Turner
Martin	physics	Bosky
James	CS	Clark

FD's

{ Name → Dept
Name → ADVISOR
ADVISOR → Dept }

Decomposition

(R2)

STUDENT_DEPT

Name	Dept
Jones	CS
Ng	chemistry
Martin	Physics
James	CS

(R3)

DEPT_ADVISOR

Dept	ADVISOR
CS	Smith
chemistry	Turner
Physics	Bosky
CS	Clark

FD's

{ Name → Dept }

FD's

{ ADVISOR → Dept }

To test for lossless decomposition.

$R_1 = R_2 \bowtie R_3$

STUDENT_ADVISOR = STUDENT_DEPT ⋈ DEPT_ADVISOR

STUDENT_DEPT × DEPT_ADVISOR

Name	Dept	Dept	ADVISOR
Jones	CS	CS	Smith ✓
"	"	C	Turner
"	"	P	BOSky
"	"	CS	Clark ✓
Ng	Chemistry	CS	Smith
"	"	C	Turner ✓
"	"	P	BOSky
"	"	CS	Clark
Martin	Physics	CS	Smith
"	"	C	Turner
"	"	P	BOSky ✓
"	"	CS	Clark
James	CS	CS	Smith ✓
"	"	C	Turner
"	"	P	BOSky
"	"	CS	Clark ✓

Π σ STUDENT_DEPT.DEPT = DEPT_ADVISOR.DEPT

R2 × R3

Name	DEPT	DEPT	ADVISOR
Jones	CS	CS	Smith
X ← Jones	CS	CS	Clark → X
Ng	Chemistry	Chemistry	Turner
Martin	Physics	Physics	BOSky
X ← James	CS	CS	Smith → X
James	CS	CS	Clark.

STUDENT_ADVISOR ≠ STUDENT_DEPT ⋈ DEPT_ADVISOR

Since original relation has only 4 tuples but the join of the decomposed relation has 6 tuples, which is incorrect.

∴ the given decomposition is lossy decomposition

definition:

A decomposition of a relation schema are $R\langle S, F\rangle$ into relation schemas R_i, $1 \le i \le n$, is said to be a lossless join decomposition (or) simply lossless if for every relation R that satisfies the FD's F, the natural join of the projections of R gives the original relation R.

$$R = \Pi_{R_1}(R) \bowtie \Pi_{R_2}(R) \bowtie \cdots \bowtie \Pi_{R_n}(R)$$

If R is a subset of $\Pi_{R_1}(R) \bowtie \Pi_{R_2}(R) \bowtie \cdots \bowtie \Pi_{R_n}(R)$

then the decomposition is lossy decomposition.

Another property to be noted is dependency preserving decomposition:

Definition:

Given a relation schema $R\langle S, F\rangle$ where S is the set of attributes & F is the set of FD's, and R is decomposed into the relation schemas $(R_1, R_2, \ldots, R_n)$ with the FD's $(F_1, F_2, \ldots, F_n)$.

Decomposition of R is dependency preserving if the closure of F' where $F = F_1 \cup F_2 \cup \cdots \cup F_n$ is equal to F^+

(i.e)

$$(F')^+ = (F_1 \cup F_2 \cup \cdots \cup F_n)^+ = F^+$$

Example:

S of STUDENT_ADVISOR (R)

$$F = \begin{Bmatrix} \text{Name} \to \text{Dept} \\ \text{Name} \to \text{ADVISOR} \\ \text{ADVISOR} \to \text{Dept} \end{Bmatrix}$$

(R1) FD of STUDENT_DEPT (R2) FD of DEPT_ADVISOR

$F_1 = \{ Name \rightarrow Dept \}$ $F_2 \{ Advisor \rightarrow Dept \}$

To check

$F_1^+ = (F_1 \cup F_2)^+$

$= \{ Name \rightarrow Dept, Advisor \rightarrow Dept \}$

To check whether $Name \rightarrow Advisor \in (F_1 \cup F_2)^+$

To prove that $X \rightarrow Y \in F_1^+$

Find x^+ and show that Y is a subset of x^+.

$Name \rightarrow Advisor$

$(Name)^+ = Name$

$= Name, Dept$

$Advisor \notin (Name)^+$

$\therefore Name \rightarrow Advisor \notin (F_1 \cup F_2)^+$

$\therefore F_1^+ \neq (F_1 \cup F_2)^+$

$\therefore$ The given decomposition is not dependency preserving.

03/01/2022

Scribe Date / / Page 134

Alternative method to check for lossless join decomposition:-

The decomposition of a relation $R(\langle x,y,z\rangle, F)$ into $R_1(\langle x,y\rangle, F_1)$ and $R_2(\langle x,z\rangle, F_2)$ where F_1, F_2 are the FD's of R_1 and R_2 then the decomposition is lossless if the common attributes x of R_1 and R_2 is a superkey of atleast one of that relations R_1 and R_2. (i.e.,) $x \to Y$ (or) $x \to Z$.

Example:-

Given the relation $R(A,B,C,D)$, FD F

$$F = \{A \to B,\ A \to C,\ C \to D\}$$

which is decomposed into

$R_1(A,B,C)$ and $R_2(C,D)$

$$F_1 = \{A \to B,\ A \to C\} \quad \text{and} \quad F_2 = \{C \to D\}$$

(i) Test whether the decomposition is lossless (or) lossy.

common attribute in the decomposition of R_1 and R_2 : C

Goal: If the decomposition of R_1 and R_2 is lossless then we have to prove that C is either superkey of R_1 or C is the superkey of R_2.

To check is C is a superkey of R_1.

$C^+ = ABC$.

$C^+ = C$

$\neq ABC$

$\therefore$ C is not a superkey of R_1

To check C is a superkey of R2.

$C^+ = CD$.

$C^+ = C$

$= CD$ $\because (C \to D)$

✓

$\therefore$ C is a superkey of R2.

since C is a superkey of R2, the given decomposition is a lossless decomposition.

(ii) Test whether the decomposition is dependency preserving

To check the decomposition is dependency preserving we have to prove that

$F^+ = (F_1 \cup F_2)^+$

$$F = \begin{Bmatrix} A \to B \\ A \to C \\ C \to D \end{Bmatrix} \quad F^+ = \begin{Bmatrix} A \to B \\ A \to C \\ C \to D \end{Bmatrix}^+ \to (1)$$

$$F_1 \cup F_2 = \begin{Bmatrix} A \to B \\ A \to C \end{Bmatrix} \cup \{C \to D\} = \begin{Bmatrix} A \to B \\ A \to C \\ C \to D \end{Bmatrix}$$

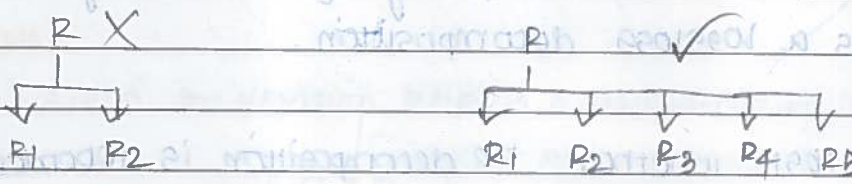
$$(F_1 \cup F_2)^+ = \begin{Bmatrix} A \to B \\ A \to C \\ C \to D \end{Bmatrix}^+ \to (2)$$

From (1) and (2), it is proved that the given decomposition is dependency preserving.

5NF : (project - join Normal Form PJNF)

Join Dependency.

Join dependency is a term used to indicate the property of a relation R that cannot be decomposed losslessly into two simpler relations R1 and R2. but can be decomposed losslessly into three or more simpler relations



lossy decomposition lossless decomposition.

Definition 5NF :

Consider a relation R and a set D of Functional or multivalued or join dependency. the relation R is in 5NF or project-join Normal Form (PJNF) they respect to D if for every join dependency (denoted as * [R1, R2, . . . , Rn]) that is applicable to R and that is implied by D the following is true

(i) Join dependency is trivial.

(OR)

(ii) Every R_i is a superkey of R.

Note :

A Join dependency is trivial if one of the projections of R is R itself. (i.e) one of the decomposed relations should be R

Example:-

Let R(A,B,C,D,E) be a relation with FD's

$$F = \begin{Bmatrix} A \to BCDE \\ C \to ABDE \\ D \to ABCE \end{Bmatrix}$$

Let us suppose the relation R satisfies the join dependency $*[(ABE)(CD)(ABCD)]$ (R1, R2, R3)

Test whether Relation R is in 5NF / PJ-NF?

Soln:- Since one of the projection of R ≠ ABCDE. The join dependency is not trivial.

To check whether R1 is a superkey of R.

To test whether R1 is a superkey of R we have to prove that $(ABE)^+ = ABCDE$.

$(ABE)^+ = ABE$

$= ABCDE \quad \because A \to BCDE$

$(ABE)^+ = ABCDE$

Hence ABE (or) R1 is a superkey of R.

R2 ⇒

$(CD)^+ = CD$

$= ABCDE \quad \because C \to ABDE$

$(CD)^+ = ABCDE$

Hence CD (or) R2 is a superkey of R.

R3 ⇒

$(ABCD)^+ = ABCD$

$= ABCDE \quad \because A \to BCDE$

$(ABCD)^+ = ABCDE$ Hence ABCD (or) R3 is a superkey of R.

Since R1, R2, R3 are a superkey of R. The given relation R is in 5NF (or) PJ-NF.

5/1/23

Domain Key Normal Form :- (DKNF)

Domain Key Normal Form (DKNF) is a normal form which requires that the database contains no constraints other than domain constraints and key constraints.

↳ Every table should have a key

Unlike other normal forms DKNF is not defined in terms of FD's, MVD's, Join dependency
$\rightarrow$ $\twoheadrightarrow$ *[ ]

∴ Note :- DKNF would require that they do not exhibit insertion, deletion and update anomalies.

Database tuning :-

Database tuning describes a group of activities used to optimize and homogenize the performance of a database. It usually overlaps query tuning but refers to design of the database files, selection of the DBMS application and configuration of the database environment (OS, CPU, etc.,)

Database tuning aims to maximize use of system resources to perform work as efficiently and rapidly as possible. Most systems are designed to manage their views of system resources but through database tuning there is scope to improve their efficiency by customizing the settings and configuration of the database & DBMS.

Denormalization :

Denormalization is the strategy that database managers use to increase the performance of a database infrastructure. It involves adding redundant data to a normalized database, to reduce certain types of problems with database queries that combine data from various tables into a single table.

In many cases denormalization involves creating separate tables or structures so that queries on one piece of information will not affect any other information tied to it.

Example:

Views in SQL is some kind of denormalization in practical.

Security & Integrity :

R1 R2

Test 3 :

CO3 : (25 marks)

SQL { DML, DDL, constraints,
Joins, subqueries, views, triggers, union,
Transactions

CO4 : (25 marks) (Definition)

MVD, FFD

Closure, TD

Key,

* * * (NORMALIZATION) * * *

Security / Integrity

Decomposition