Unit - I

- Data - Facts
- Database - Collection of related data
- DBMS - Collection of programs that allows users to create & manage the database
  → General Purpose Software System

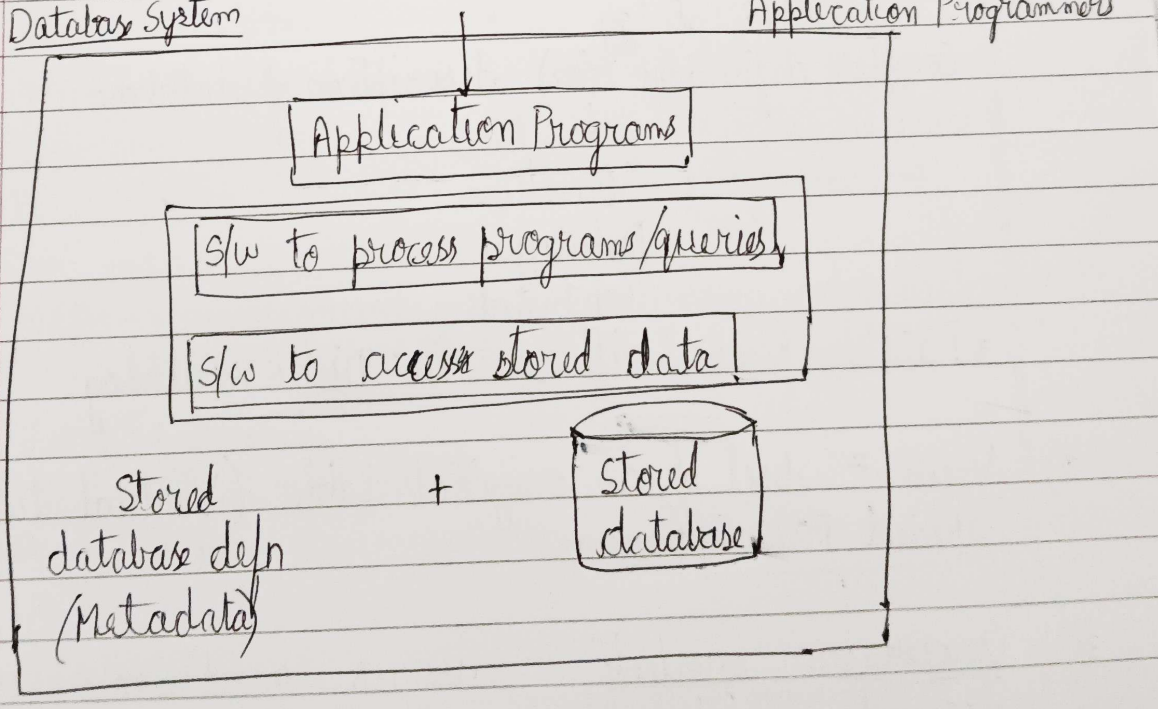
3 processes facilitated by DBMS

1. Define → structure, data types, constraints
2. Construct
3. Manipulate → Query - update, generate reports

Database System

Stored database definition + Stored database

s/w to process programs/queries
s/w to access stored data



Eg. University Database
STUDENT - Student name, USN ...
COURSE - Course name, Course Code ...
GRADE-REPORT

Traditional File Processing Approach v/s Database Approach

Traditional	Database
• Redundancy - wastage of storage	• Single repository

X Characteristics of Database approach

1) Self describing nature
2) Support of multiple views
3) Sharing of data & multi-user transaction processing
4) Insulation b/w programs of data

1) Metadata
DBMS Catalog
Complete definitions (or) description of database
↓
Structure
Datatypes
Constraints
→ Database users will use the DBMS Catalog

2) View - Subset of an original table (Virtual table)
Student transcript view

3) Concurrency module - ensures multiple users can access the data, where only a single user can update data at a time to maintain consistency of data

4) Data Abstraction
Program data independence + & Program operation

File processing
Structure of file → Application programs
→ Structure should be changed in appl. program

Database approach
→ Maintained separately
→ Object oriented relational database
→ Users can define operations as a part of database definition
→ Interface - Operation name, no. of arguments
→ Implementation

Database users
2 types
1) Actors on the scene — Eg. → Database Administrator (DBA)
2) Workers behind the scene

Actors on the scene - Eg. DBA, Database Designers, End users, System designers, Application programmers

1) DBA → Gives privileges to database, administering the resources, monitoring usage of database,
→ security
→ Poor response time
→ H/w and S/w resources

2) Database designers
- Understand requirements of users (communication)
- Identify the data stored in the database

96 = 69 Kavi

3) End users
1. Casual - Have knowledge about query language. Eg. managers
2. Naive or Parametric - No knowledge about query lang. Eg. clerks
3. Sophisticated - Have thorough knowledge about DBMS software
4. Standalone -

Resources allocated by DBA
- Primary resource - Database
- Secondary resource - DBMS and related S/W

* Transaction that involves querying and updating the database is called Canned Transaction

4) System Designers and ~~Implementors~~ Application Programmers
- Develop specifications for canned transactions - done by system designers. Implemented by ~~implementors~~ appl. programmers as programs.

Workers behind the scene
- Design, develop and implement the DBMS software and system environment

Categories
1. DBMS System designers and Implementers
   ↳ DBMS Modules/components:
   Concurrency control module, Recovery subsystem, catalog, Query Language, security module, interface

2. Tool Developers
   ↳ S/W packages
   For database design, performance

3. Operators and Maintenance Personnel
- Running and maintenance of H/W and S/W environment

Data Mo
Collecti
The stru
- Include
in the

1) Low le
2) High l
3) Represe

1) Low l
- Gives
- Recor

2) High
Eg.
- Gives

→ Enti
→ Attr
→ Rela

3) Rep
- Eas
- Us
- Al

Sc
Da

E

Data Models

Collection of concepts that can be used to describe the structure of the database

- Includes set of operations for updating and retrieval in the database

1) Low level or Physical Data Model
2) High level or Conceptual - " -
3) Representational or Implementational - " -

1) Low level/Physical data model
   - Gives information on how the data is stored internally
   - Record format + Access paths

2) High level/Conceptual data model
   Eg. ER Model (Entity Relationship)
   - Gives information about entities, attributes, relationships

→ Entity - Real World object
→ Attribute - property of entity
→ Relationship b/w 2 entities Eg. Employee → works for → Dept.

3) Representational/Implementational data model
   - Easily understood by the end users
   - Used in traditional commercial DBMS
   - Also called Record-Based data model

Schema, Instance

Database schema - Structure of database

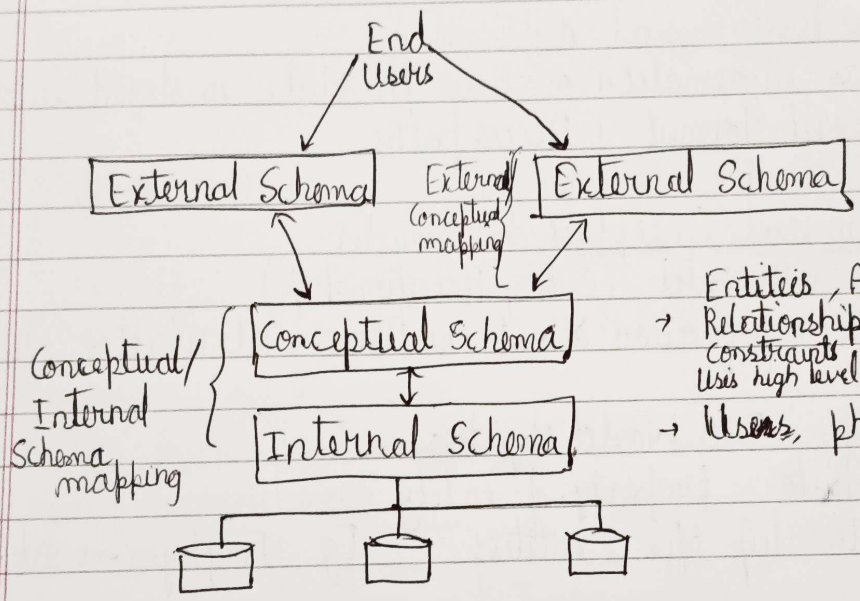
Eg. Employee

E-name	E no.	E age	E Sal

Instance - Values in the database

Note:
Database Schema - Intension
Database State - Extension

Three Schema Architecture
Used to separate the physical database and user applications



Mapping - Process of transforming the request and response b/w the levels is called mapping

Data Independence
Capacity to change the schema at one level of a database system without any need to change the schema at the next higher level
Two types:
1) Logical data independence
2) Physical data independence

1) Logical data independence
→ Change in conceptual schema
Eg. change in relationships

2) Physical data independence
→ Change in internal schema
Eg. change the access paths, change the data structure

07/12/23 Lab

SQL Queries

- Creating database - create database database_name;
- Using database - use database_name
- Creating table - create Table tablename (Ename varchar(10), Eno. int, EAddr varchar(20));
  (employee)
- Table rows - Tuple
  Table column - Attribute
- Inserting in table - insert into employee values ('XX', 20, 'XYZ');
  + insert into employee (Ename, Eno., EAddr) values ('XX', 20, 'XYZ'), (. . . .);
- Delete values : delete from tablename where Eno. = 10;
  → delete from tablename where Ename = 'XX';
  → The row is deleted
- Delete all contents - delete from tablename;
- drop Table tablename - Removing entire structure of the Table from the database
- Checking contents - select * from tablename;
  → select * from employee where Eno. = 10;
  ↳ Displays the rows where Eno. is 10
- Add attributes - alter table employee add column job varchar(15)
- show databases; - List all databases

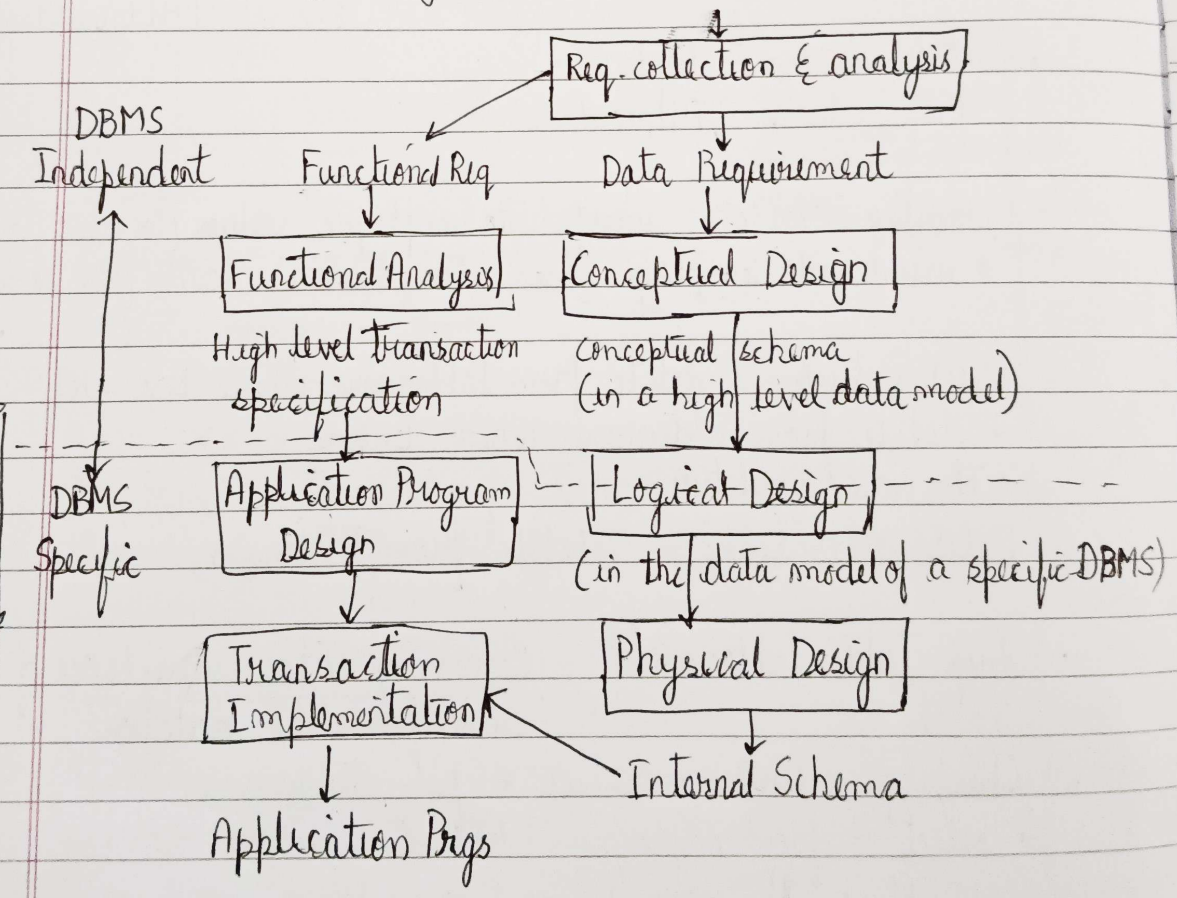
To implement the database:
Conceptual and Internal Schema
DDL - Data Definition Language → used by DBA + database designer

If there is seperation b/w the 2 layers
1. DDL - for conceptual schema
2. SDL - Storage definition Language - for internal schema

VDL - View definition Language - for mapping user views

DML - Data Manipulation Language - includes operations like insertion, updation, retrieval

Data Modeling using ER Model



- Logical design includes actual implementation of database.
  → Most commercial implementations use Relational data Model

- Physical design contains internal schema → organisation of the files, access paths

ER Model
Describes data with respect to entities, attributes and relationships

Entity

→ Entity type
Collection (or) set of entities having the same set of attributes

→ Entity set
Actual collection of entities

Eg. Entity Type
Employee (Ename, Eno, Esal)

e1 (XX, 10, 10k)
e2 (YY, 20, 20k)
:
en

$e_1, e_2 \ldots e_n$ → Entity set

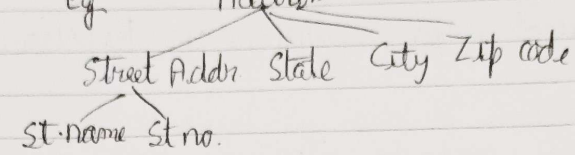
→ Attribute - Property that describes an entity

Types:
1. Simple vs. Composite attribute
2. Single vs. Multivalued - " -
3. Stored vs. Derived

1) Simple Attribute - Attribute whose values aren't divisible. Eg. Age

Composite attribute - Multiple components

Eg. Address



2) Single attribute - Only one value
Eg. Age

Multivalued Attribute - More than one value
Eg. Person - degree

3) Stored attribute - All attribute values are related.

Derived attribute - Attribute values are derived from stored attributes

Eg. Age is derived from Birth Date

Birth date (stored)

↓

Age (derived)

NULL value - Implies no value

Complex attribute - Combination of composite and multivalued attributes.

→ Composite represented with ( )
Multivalued — " — { }

Eg. Person can have more than 1 residence & each residence can have single address & multiple phones

Symbol
Eg. If phone no. has 2 components :- (Area_code, Ph.no.)

For residence eg.

~~{ DI {~~
{ Address_Phone ({ Phone (Area_code, Ph_no) },
Address (Street_Addr (street_name, St_name_Apt_no),
state, city, zip)) }

Key Attribute - Distinct values for all entities in the entity table.

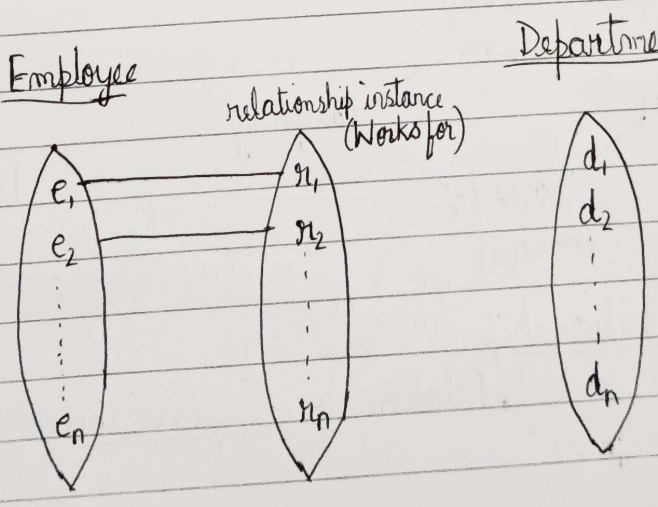
Eg. Employee
Eno - key

→ Cannot have NULL value
→ Two attribute values together may form the key

Value sets (Domain)
Value sets are domains of the attributes
• Each attribute is associated with a value set
Eg. Age - (25 to 60)

→ Cannot be included in the ER diagram

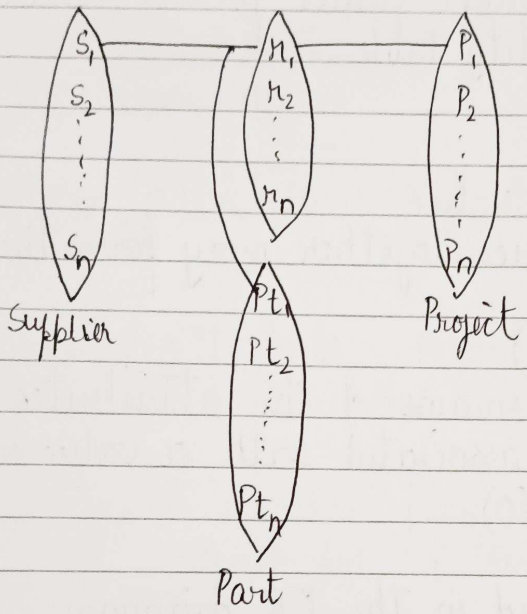
Relationship



Degree of a relationship is the no. of participating entity types

1. Binary – 2 entity types. Eg Employee works for Department

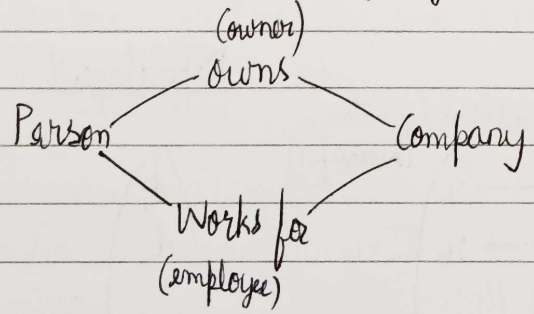
2. Ternary
   Supplier
   Project
   Part



Role names

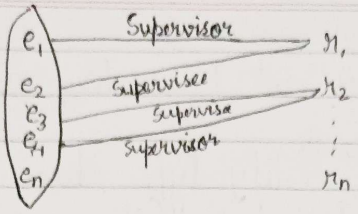
- Needed when same entity type participates in the relationship

Eg.



Recursive Relationship

Employee — Relationship – Supervision



- Should specify the role names

Constraints

1) 1:1 Eg. Employe $\xrightarrow{1}$ Manages $\xrightarrow{1}$ Department

2) N:1 Eg. Employees $\xrightarrow{N}$ Works for $\xrightarrow{1}$ Department

3) M:N Eg Employees $\xrightarrow{M}$ Works on $\xrightarrow{N}$ Projects

(1) → Cardinality ratio — [The above 3 come under cardinality ratio]
No. of instances entity can participate in.

(2) Participation

1. Total → All the entities should participate in the given relationship

2. Partial → Some entities participate in the given relationship

Attributes of Relationship

Employee (Ename, Eno, Esal)
Attributes

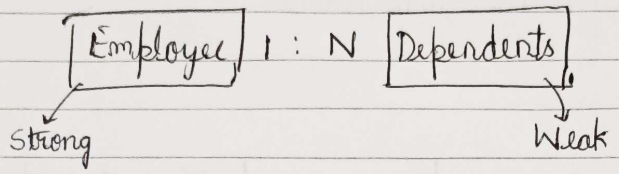
For Manages - Relationship,
Manager_start_date
is the attribute

For Employee → Works on → Project
No. of years
is the attribute

Weak entity type

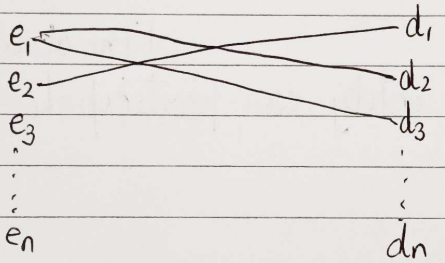
Entity type that does not have key attributes on its own.

- All the entities of the weak entity type is related to entities of another entity type. (strong entity type)



- Strong entity type has key attribute

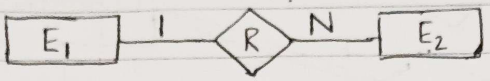
Employee	Dependents
• Eno. is the key	• Dname is the partial key



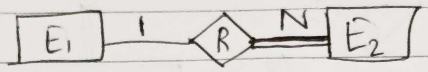
ER Notations

1. → Entity Type → Weak entity
2. → Attribute
3. → Key Attribute → Partial key
4. → Composite Attribute
5. → Multivalued

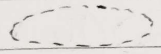
6.  → Relationship  → Connected to weak entity

7. 

→ Binary Relationship with Cardinality Ratio 1:N

8. 

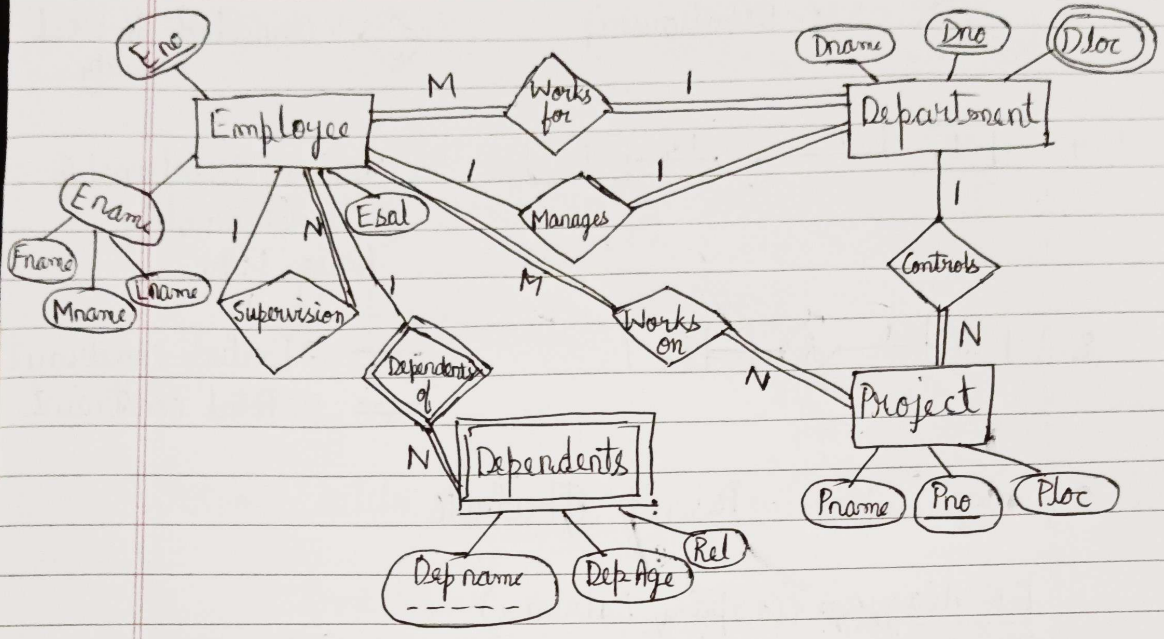
—— : Partial constraint
═══ : Total constraint

9.  → Derived attribute

ER diagram (Company Database)
Requirements: Employee, Department, Project

Dept - Dname, Dno, Particular employee manages dept., start-date, several locations, controls no. of projects,
Project - Project name, Pno, Single location

Emp - Ename, Eno, Addr, Esalary
→ Assigned to one dept.
→ Many employees can work on several projects but not necessarily controlled by the same dept.
→ No of hours per week working on the project
→ Keep track of supervisor of each employee, dependents



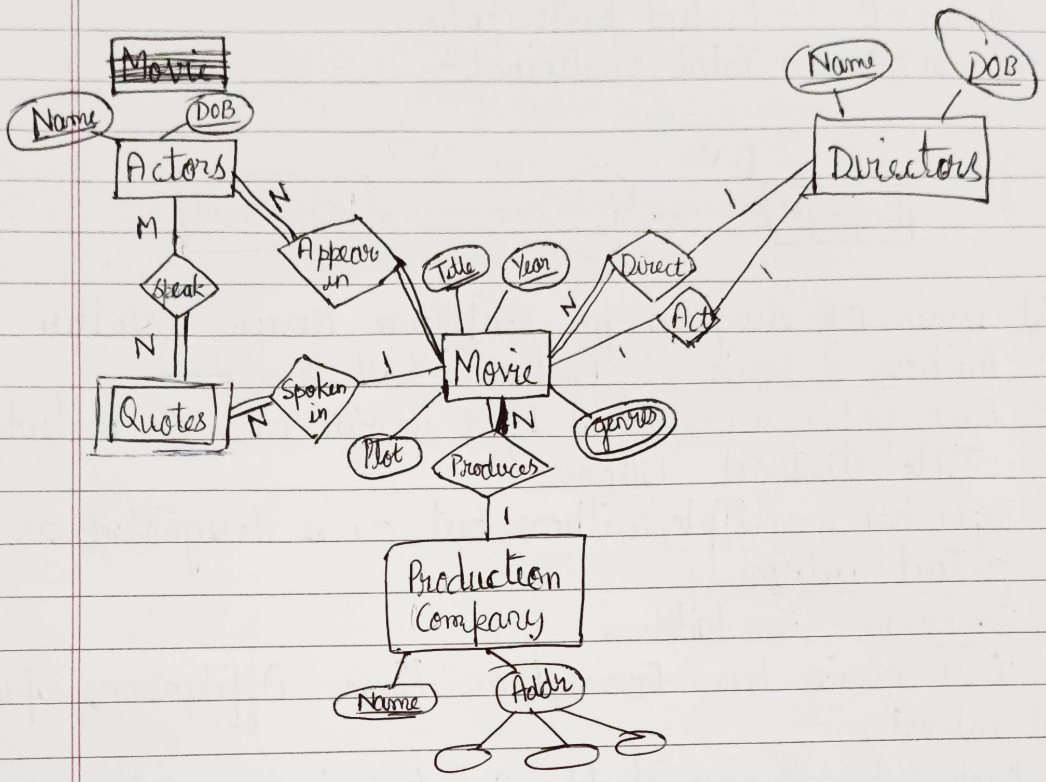
Q) Design movie database

Movie - Title and Year of Release, Length in minutes,
- Production Company
- Genres (Horror, action, drama etc.)
- Plot outline
- 0 (or) more quotes
  ↳ Spoken by a particular actor appearing in the movie

Actors - Name, DOB
- Appear in one (or) more movies
- Each actor has specific role in the movie

Directors - Name, DOB
- Direct one (or) more movies
- Can also act

Production Company - Name, Address
- Produces one (or) more movies

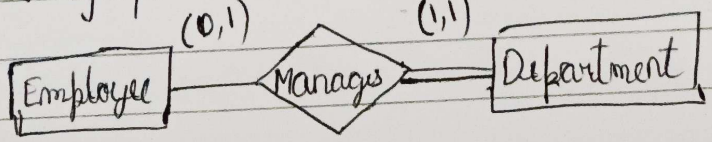


Structural Constraints on Relationship

(min, max) – It associates a pair of integers with each participation of an entity type E in a relationship type R

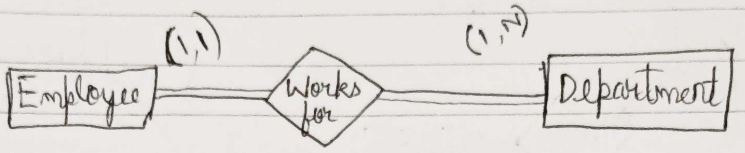
$0 \leq min \leq max$ and $max \geq 1$

- For each entity e in E, e must participate in atleast min and at most max relationship instances in R at any point in time



Note:

- min = 0 → Partial participation
- min > 0 → Total participation



Q) Draw ER diagram for Conference Review database

1) Authors of papers - First name, Last name
2) Each paper is identified by a unique id, it includes Title, Abstract, Filename
3) Paper has multiple authors but one is designated as contact author
4) Reviewers - Emailaddress.
   → Each reviewer has Fname, Lname, Ph no., Affiliation, Topics of interest
5) Each paper is assigned b/w 2 to 4 reviewers. Reviewer rates each paper on a scale of 1-10
   4 categories
   → Technical
   → Merit
   → Readability
   → Originality
6) Each reviewer provides an overall recommendation
   Each review - 2 types of written comments
   → one to be seen by review committee
   → other - feedback to authors

ER to relational model mapping

Steps:

1) Mapping of strong entity type

For each strong entity type E in ER, create a new relation R.

Eg. For company database

Employee

EFname	EMname	ELname	Eno	ESal	EAge

Department.

Dname	Dno

[Do not include multivalued attributes]
In this case DLoc is multivalued

Project

Pname	Pno	Plocation

2) Mapping of weak entity type

For each weak entity type W in ER diagram, create a new relation

Dependent

Foreign key

Dep_name	Dep_Age	Relation	Emp_no

Primary key → Eno + Dep_name

3) Mapping of Binary 1:N Relationship

(i) Employee $\overset{N}{\underset{\text{Works for}}{\text{———}}}\overset{1}{}$ Department

(ii) Department $\overset{1}{\underset{\text{Controls}}{\text{———}}}\overset{N}{}$ Projects

(iii) Employee $\overset{1}{\underset{\text{Supervision}}{\text{———}}}\overset{N}{}$ Employee

- Identify the entity type having N
- The entity type on the other side's primary key should be included as foreign key on the relation on ~~N's~~ N's side

(i) Employee $\overset{N}{\underset{\text{Works for}}{\text{———}}}$ $\overset{1}{}$ Department

EFname	.....	Eno	...	Dnum
		PK		FK

(ii) Department $\overset{1}{\underset{\text{Controls}}{\text{———}}}$ $\overset{N}{}$ Projects

Pname	Pno	Plocation	Dnum
	PK		FK

(iii) Employee $\overset{1}{\underset{\text{Supervision}}{\text{———}}}$ $\overset{N}{}$ Employee

EFname	.....	Eno	Dnum	Super_Eno
		PK	FK	FK

[Named Super_Eno to avoid Eno rename]

4) Mapping of Binary M:N Relationship

Employee $\overset{M}{\underset{\text{Works on}}{\text{———}}}$ $\overset{N}{}$ Project

Works on

Eno	Pno	Hours
FK	FK	

PK → Eno + Pno

5) Mapping of Multivalued attribute

Department entity type → Dlocation (multivalued attr)

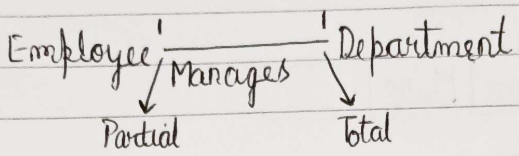
For each multivalued attribute A, create a new relation

Dlocation	Dnum

FK - Dnum
PK - Dlocation + Dnum

6) Mapping of Binary 1:1 Relationship

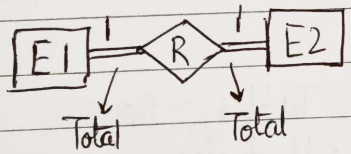
(i) Foreign key Approach



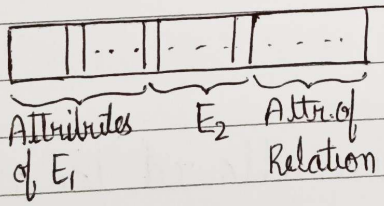
Department

Dname	Dno	DMgr_eno	Mgr_start_date
	PK	FK	

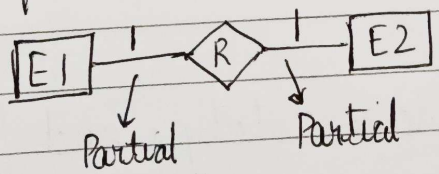
(ii) Merged Relation Approach



Relation



(iii) Cross reference (or) Relationship relation approach



Create a new relation R for the relationship

Attr. of Rel	PK of Rel	PK of Rel
	FK wrt E_1	FK wrt E_2

7) Mapping of Nary Relationship

Supplier

Sname	Sno	....

Project

Pname	Pno	...

Part

Pt_name	Pt_no	....

Supply

Sno	Pno	Pt-no	.....

PK ← Sno

FK → Pno, Pt-no

Relational Model

Relation - Table

Student

Sname	Sno	SAge

Tuples - Rows

Attributes - Columns

Domain

Set of atomic values that are allowed for a particular attribute.

- Values are indivisible
  Eg. Age
- A domain is given a name, datatype and format
  Eg. Emp_Phone

Q) Write the d

Relation Sc

- Structure of
- Intension
- Format: R(A
- Eg. Emp (En

Relational

Employe

Departme

Project

Depende

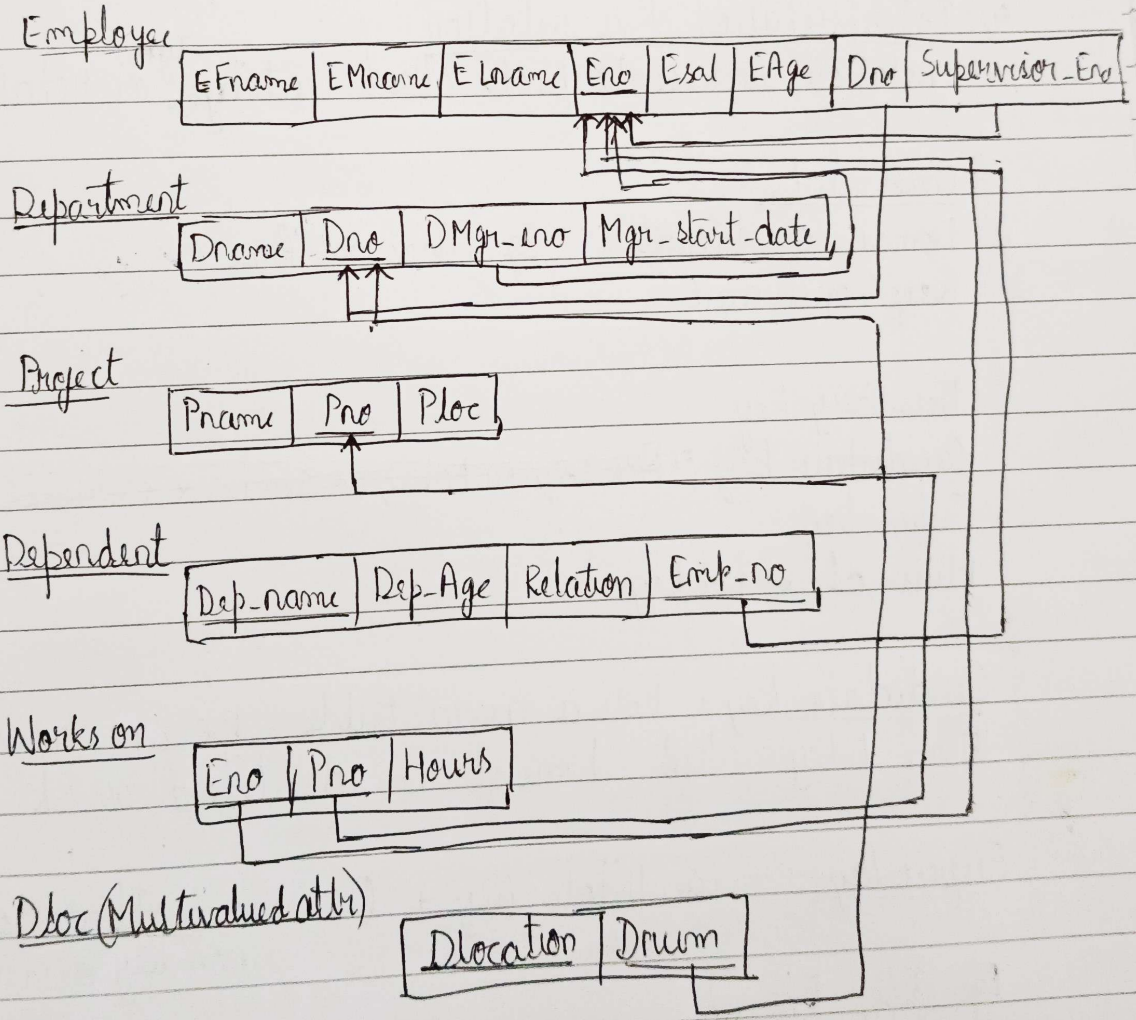
Works

Dloc

Q) Write the difference b/w relation schema and relation instance

Relation Schema	Relation Instance
• Structure of the ~~database~~ table	• Values in the ~~database~~ table
• Intension	• Extension
• Format: $R(A_1, A_2 \ldots A_n)$	• $t_i = V_1, V_2 \ldots V_n$ [i^{th} tuple]
• Eg. Emp (Ename, Eno ... Esal)	• Represented by $r(R)$
	• $r(R)$ is a subset of cartesian product of the domains
	~~$r(R) \subset (D_1 \times D_2)$~~
	• $r(R) \subseteq (dom(A_1) \times dom(A_2) \ldots dom(A_n))$

Relational Model for Company database



Characteristics of Relation

1) Ordering of tuples
   Any order can be given

2) Ordering of ~~tuples~~ values within tuples
   Any order can be given
   t_1 = <name, XX> <no, 10>
   t_2 = <no, 10> <name, XX>
   } Same

3) Values in tuples
   - Atomic value
   - If it doesn't exist, give NULL Value

4) Interpretation of a relation
   - Relation represents facts about entities, type or relation

Constraints
1) Domain constraint - Eg: Phone no, Age
2) Key constraint

Primary key
Candidate key ~~- For a given table, we can have more than 1 key,~~
Super key
Minimal super key

Candidate Key - For a given table, we can have more than 1 key, but 1 must be identified as PK

Super key - Candidate Key + 0 or more ~~attributes~~ non key attributes

Eg. (Eno, Ename)
(Eno, Ename, EAge)

· No two rows must have the same values.
(Violates key constraint)

Minimal superkey - Min set of attributes or values required to identify rows uniquely. (nothing but primary key)

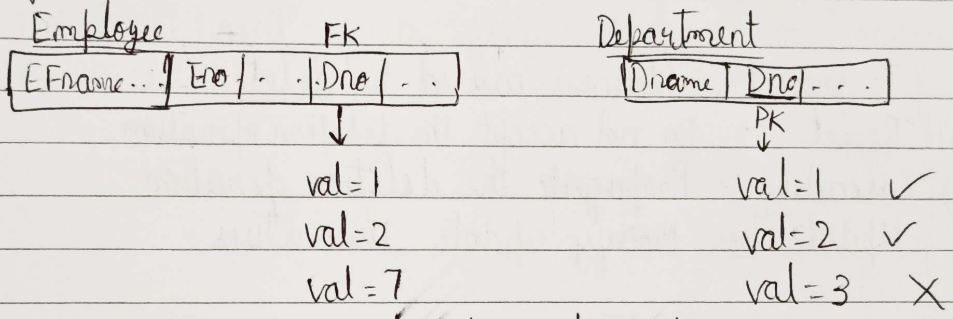
3) Entity integrity ~~and~~ constraint

· No null value for the PK attribute.
· Key constraint & entity integrity is specified on a single relation

4) Referential integrity constraint

· Need two relations
· ~~Eg~~ Data type and values of FK and PK must be same

Eg.



↳ 7 does not exist, so violates relational integrity constraint

Different Types of operations and dealing with constraint violations

1) Insert
2) Delete
3) Update

1) Insert operations can violate all constraints
Eg insert into employee <'XX', 'YY', 'ZZ', NULL, 5000, 28, 1, 20>
→ Violates entity integrity constraint

· Assume Eno 10 already exists. If Eno 10 is inserted again, duplicate key values, so violates key constraint
insert into employee('XX', 'YY', 'ZZ', 10, 5000, 28, 1, 20>

Domain constraint violation

Insert into employee <'XX', 'XY', 'ZZ', 10, 'abc', 80, 2, 70>

→ Salary is 'abc', but it should be a number

→ Age is 80, but should be b/w 25 & 60

2) Delete

Delete operation violates referential integrity constraint

3) Update

Update operation can violate all constraints

To prevent violations caused by delete:

1) Reject - Do not accept the deletion operation
2) Cascading - Propagate the deletion operation
3) Update value - Modify/update the values

For update operation:

If there is no Dno 15, and the Dno value is updated to 15, it violates referential integrity

Note:

~~Consider F~~

No. of super keys = 2^{N-1} N - Total no. of attributes

Eg $R(a_1, a_2, a_3)$

Consider $R(a_1, a_2 \ldots\ldots a_n)$

If $CK \Rightarrow a_1 a_2 a_3$

No. of SK = 2^{N-3}

Relational Algebra

Set of operations that are used for the relational mode

1) Select, Project, Join
2) Union, Intersection, Difference, Cartesian Product

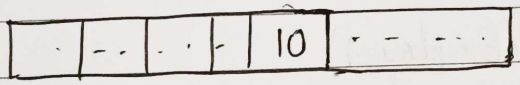
Input - 1 or 2 relations
Output - Single relation

Select operation (σ)

It is used to select subset of tuples from a given relation based on a given predicate or condition.

Eg.

$\sigma_{eno=10}(Employee)$



Note: Can use $=, \neq, >, <, <=, >=$ operators

AND, OR

Eg. $\sigma_{cond1\ AND\ cond2}(R)$

Commutative property of σ

$\sigma_{cond2}(\sigma_{cond1}(R)) \Rightarrow \sigma_{cond1}(\sigma_{cond2}(R))$

Project Operation (Π)

It is used to project (select) set of attributes

Eg

ename	eno
XX	10
YY	20
ZZ	30

$\Pi_{ename, eno}(Employee)$

- Project operation is NOT commutative, associative

Eg. $\Pi_{ename, eno}(\Pi_{eno}(Employee))$

→ Only eno has been projected, so cannot project ename

- Project operation eliminates duplicates

Eg. $\Pi_{esal}(Employee)$

esal
60000
70000
80000

Sequences of operations and Renaming of attributes

$\Pi_{ename, eno}(\sigma_{eno>10}(Employee))$

O/p

ename	eno
XX	10

The operations can also be written as:

$T_1 \leftarrow \sigma_{eno>10}(Employee)$

$T_2 \leftarrow \Pi_{ename, eno}(T_1)$

1/1/24

Renaming of Attributes

$\rho_S(R)$

$\rho_{(B_1, B_2, B_3 \ldots B_n)}(R)$ → Renaming attributes

$\rho_{S(B_1, B_2 \ldots B_n)}(R)$ → Renaming both attributes & relation

Set operations

1) Union
2) Intersection
3) Difference (Minus)

1) Union

Two relations should be union compatible if they have the same degree and domain.

- Degree - No. of attributes in a given relation

$R(A_1, A_2, A_3 \ldots A_n)$
$S(B_1, B_2, B_3 \ldots B_n)$

Domains should be same:

$$dom(A_i) = dom(B_i)$$

- Union: $R \cup S$
- Commutative: $R \cup S = S \cup R$
- Associative: $R \cup (S \cup T) = (R \cup S) \cup T$
- Union operation is commutative and associative
- Union includes all the tuples that are either in R or in S

Eg.

Student

Fname	Lname
XX	A
YY	B
ZZ	T
PP	S

Instructor

Fn	Ln
XX	A
PP	S
TT	L

Student ∪ Instructor

Fname	Lname
XX	A
YY	B
ZZ	T
PP	S
TT	L

2) Intersection

$R \cap S$

- Intersection is both commutative & associative
- $R \cap S = S \cap R$
- $R \cap (S \cap T) = (R \cap S) \cap T$

Eg. Consider the student and instructor tables,

Student ∩ Instructor

Fname	Lname
XX	A
PP	S

3) Minus

$R - S$

- Minus is NOT commutative, associative
- $R - S \neq S - R$
- $R - (S - T) \neq (R - S) - T$
- Query: List out all students who are not instructors

Student - Instructor

Fname	Lname
YY	B
ZZ	T

Cartesian Product

$R \times S$ (No need to be union compatible)

- Includes all combinations of tuples that are in R as well as in S
- If no. of rows in R is n_R, no. of rows in S is n_S, no of rows in $R \times S = n_R * n_S$

Eg. Student

Sid	Sname
10	XX
20	YY
30	ZZ

Stud-Grade

Sid	Grade
10	A
20	B

Student × Stud_Grade

Sid	Sname	Sid	Grade	
10	XX	10	A	✓
10	XX	20	B	✗
20	YY	10	A	✗
20	YY	20	B	✓
30	ZZ	10	A	✗
30	ZZ	20	B	✗

[Sid is different]

- To display only rows with same Sid:

$$\sigma_{\text{student.Sid} = \text{Stud\_Grade.Sid}} (\text{Student} \times \text{Stud\_Grade})$$

∴

Sid	Sname	Sid	Grade
10	XX	10	A
20	YY	20	B

→ Disadvantage - Repetition of Sid
↳ So project to avoid this:

$$\Pi_{\text{Sid, Sname, Grade}} \left(\sigma_{\text{student.Sid} = \text{Stud\_Grade.Sid}} (\text{Student} \times \text{Stud\_Grade}) \right)$$

∴

Sid	Sname	Grade
10	XX	A
20	YY	B

Q) Relational database schema:
Hotel (hotelno (PK), hname, city)
Room (rno (PK), hotelno (FK), type, price)
Booking (hotelno (FK), guestno (FK), datefrom, dateto) (PK)
Guest (guestno (PK), guestname, guestaddr)

(i) List all single rooms with a price below Rs. 3000 per night

(ii) List the price & type of all the rooms at the Sagar hotel

(iii) List all the guests currently staying at the Sagar hotel

(iv) Project the guest name & address

Soln (i) $\sigma_{(Room.type = 'single') \text{ AND } (Room.price < 3000)}(Room)$

(iv) $\Pi_{guestname, guestaddr}(Guest)$

* **Note:**
  $R * S \rightarrow$ natural join

(ii) $\Pi_{price, type}\left(\sigma_{hname = 'Sagar'}(Hotel * Room)\right)$

(iii)

Join Operation $R \bowtie S$

Related tuples are included in the relation

Categories:

- Inner Join
  - Theta Join
  - Natural Join
- Outer Join
  - → Left outer join
  - → Right outer join
  - → Full outer join

Eg. $R \bowtie_{EID = ID} S$

EID - Attribute of relation R

ID - Attribute of relation S

Can also write like: $R \bowtie_{R.EID = S.ID} S$

Natural Join $R * S$

Attribute name should be same in both the relations

Theta Join

A general join condition which is of the form <cond1> and <cond 2> and <cond 3>, where each condition is of the form $A_i \theta B_j$

Note: If the operator is =, then it is Equijoin. else, it is not equijoin

Q) Consider 2 relations with R_1 and R_2

EID	Dept
10	D1
20	D2
30	D3
40	D4

ID	Salary
10	5000
20	6000
60	7000
80	8000

$R_1 \bowtie_{EID=ID} R_2$ → Inner Join (Equijoin)

EID	Dept	ID	Salary
10	D_1	10	5000
20	D_2	20	6000

For natural join, rename EID to ID,

$R_1 * R_2$ → No need to specify the condition

$\rho_{(ID)} R_1(EID)$

Outer join

Non-matching tuples will also be included

$R_1 ⟕ R_2$ → Left outer join

$R_1 ⟖ R_2$ → Right outer join

$R_1 ⟗ R_2$ → Full outer join

$R_1 \bowtie R_2$

EID	Dept	ID	Salary
10	D_1	10	5000
20	D_2	20	6000
30	D_3	NULL	NULL
40	D_4	NULL	NULL

$R_1 \bowtie R_2$

EID	Dept	ID	Salary
10	D1	10	5000
20	D2	20	6000
NULL	NULL	60	7000
NULL	NULL	80	8000

$R_1 \bowtie R_2$

EID	Dept	ID	Salary
10	D_1	10	5000
20	D_2	20	6000
30	D_3	NULL	NULL
40	D_4	NULL	NULL
NULL	NULL	60	7000
NULL	NULL	80	8000

Division Operation

$R(X) \div S(Y)$

X - Set of attributes of relation R
Y - Set of attributes of relation S

Condition: Y should be a subset of X

Eg:

R_1

Ename	Pno
XX	1
XX	2
YY	1
Zz	2
ZZ	1
SS	2

R_2

Pno
1
2

$R_1 \div R_2$

Ename
XX
ZZ

[All the ename who are working on both 1 and 2]

Q) Student (id, name)
enrolledin (id, code)
Subject (code, lecturer)

(i) Which subject is handled by a particular staff?

Soln $\Pi_{code}(\sigma_{lecturer='xx'}(Subject))$

(ii) Who teaches the course CS 1500 ?

Soln $\Pi_{lecturer}(\sigma_{code='CS1500'}(Subject))$

(iii) Who teaches the course CS1500 or CS 2500

$\Pi_{lecturer}(\sigma_{code='CS1500' \text{ OR } code='CS2500'}(Subject))$

(iv) List out the names of students who are enrolled in a particular course

$\Pi_{name}(\sigma_{code='xx'}(Student * enrolledin))$

(v) List out the names of students who have enrolled in both CS1500 and CS2500. Use set operation

$\Pi_{name}(\sigma_{code='CS1500'}(student * enrolledin) \cap \sigma_{code='CS2500'}(student * enrolledin))$

(vi) ~~Two~~ ~~teo~~ Who teaches atleast 2 different subjects.

Sol^n Consider the Subject relation as R and S
→ R and S are both subject tables → Recursive join

$$\Pi_{lecturer}\left(\sigma_{R.lecturer = S.lecturer \text{ AND } R.code \neq S.code}(R * S)\right)$$

(vii) List out the names of students enrolled in atleast 2 different subjects.

Sol^n Consider the Enrolled in relation as R and S

$$\Pi_{name}\left(Student * \left(\sigma_{R.id = S.id \text{ AND } R.code \neq S.code}(R * S)\right)\right)$$

(viii) List out the names of the students taking a subject taught by a particular lecturer

Sol^n $$\Pi_{name}\left(Student * \left(enrolledin * \left(\sigma_{lecturer = 'xx'}(Subject)\right)\right)\right)$$

Q) Members (M_id, Name, Desg, Age)
Books (Bid, Btitle, BAuth, BPublisher, BPrice)
Reserves (M_id, Bid, Date)

(i) Find the names of members who are older than 45 year

$$\Pi_{Name}\left(\sigma_{Desg = 'Professor' \text{ AND } Age > 45}(Members)\right)$$

(ii) Find the titles of books reserved by professors

$$\Pi_{Btitle}\left(Books * \left(Reserves * \left(\sigma_{Desg = 'Professor'}(Members)\right)\right)\right)$$

(iii) Find authors & titles of books reserved on a particular date

$$\Pi_{BAuth, Btitle}\left(Books * \left(\sigma_{Bid = Date = 'xx'}(Reserves)\right)\right)$$

(iv) Find nam

Sol^n

~~Π_{name}~~

Π_{name}

(v)

HW Q) Supplier
Parts (P
Catalog

(i) Find t
(ii) Find
(iii) Find S
(iv) Find S
(v) Find

(iv) Find names of members who have reserved all books

Sol^n

~~Π_{name} (Members * (Reserves ÷ Π_{B}~~

Π_{name} (Members * ($\Pi_{M\text{-}id, Bid}$ (Reserves) ÷ Π_{Bid} (Books)))

(v)

HW Q) Suppliers (Sid, Sname, Asddr)
Parts (Pid, Pname, color)
Catalog (Sid, Pid, Cost)

(i) Find the names of suppliers who supply some red part.
(ii) Find Sid of suppliers who supply some red or some green part
(iii) Find Sid of suppliers who supply every part
(iv) Find Sid of suppliers who supply every red part
(v) Find Sid of suppliers who supply every red or every green part