DBMS college notes (3)

Topic → Relational model
- Relational Integrity constraints

ER → Relational model

- **RELATION** (A table of values)

1) A relation may be thought of as set of rows. Each row represents a fact that corresponds to a real-world entity or relationship.

2) Each column typically is called by its column name or column header or attribute name.

3) Schema of relation : $R(A_1, A_2 - - A_n)$ relation schema R is defined over attributes $A_1, A_2 - - A_n$

eg CUSTOMER (cust-id, Cust-name, Address, Phone#)

- Here CUSTOMER is a relation defined over four attributes Cust-id, Cust-name, Address, Phone#, each of which has a domain or set of valid values. for eg domain of cust-id is 6-digit numbers.
- A domain may have a data-type or format defined for it. eg Dates have domain as yyyy-mm-dd.

- **TUPLE**

1) A tuple is ordered set of values. each value is derived from appropriate domain.

eg CUSTOMER table may be referred to as a tuple in table as. <632895, "John", 101 Main Street", "(404)894-2000">

2) Relation may be regarded as set of tuples (row)

- Formal defination

A relation is formed over the cartesian Product of the sets ; each set has values from a domain , that domain is used in specific role which is conveyed by attribute name

formally,

Given $R(A_1, A_2, \ldots A_n)$

$r(R) \subset \text{domain}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n)$

R: Schema of relation

r of R : specific "value" or population of R

R is also called intension of a relation

r is also called extension of a relation .

eg Let $S_1 = \{0, 1\}$ $S_2 = \{a, b, c\}$

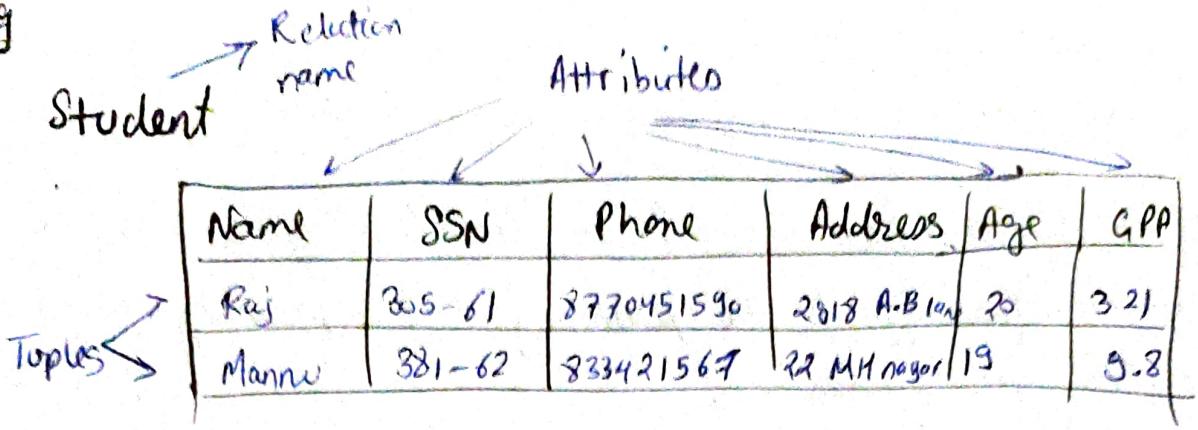
Let $R \subset S_1 \times S_2$

eg $r(R) = \{\langle 0, a\rangle, \langle 0, b\rangle, \langle 1, c\rangle\}$

is one possible "state" or "population" or "extension" r of relation R over domain S_1 & S_2 , it has 3 tuples.

Informal terms	Formal terms
Table	Relation
Column	Attribute / Domain
Row	Tuple
values in column	Domain
Table Defination	Schema of Relation
Populated table	Extension

eg



Relational Integrity Constraint

Constraints are conditions that must hold on all valid relation instances. There are 3 main types of constraints:

1) key constraint
2) Entity integrity constraint
3) Referential integrity constraint

Note
4) can even add (Domain constraint
values inside an attribute lies in a domain.
- values can't be composite value

Key Constraints

- Superkey of R : A set of attributes SK of R such that no two tuples in any valid relation instance r(R) will have same value of SK. That is, for any distinct tuples t_1 and t_2 in r(R), $t_1[SK] \neq t_2[SK]$
- Key of R: A "minimal" superkey, that is superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

eg CAR (State, Reg#, Serial No, Make, model, year)

has two keys K1 = {State, Reg#}, K2 = {Serial No}

which are also superkeys,
{Serial No, Make} is a superkey but not a key.

- If a relation has several candidate keys, one is chosen arbitrarily to be primary key. primary key attributes are underlined.

Entity Integrity Constraint

- Relational Database Schema: A set S of relation schemas that belong to same database, S is name of database

$$S = \{R_1, R_2 - - R_n\}$$

- Entity Integrity: Primary key attributes PK of each relation schema R in S can not have null values in any tuple of r(R). This is because primary key values are used to identify the individual tuples.

$$t[PK] \neq \text{null} \quad \text{for any tuple } t \text{ in } r(R)$$

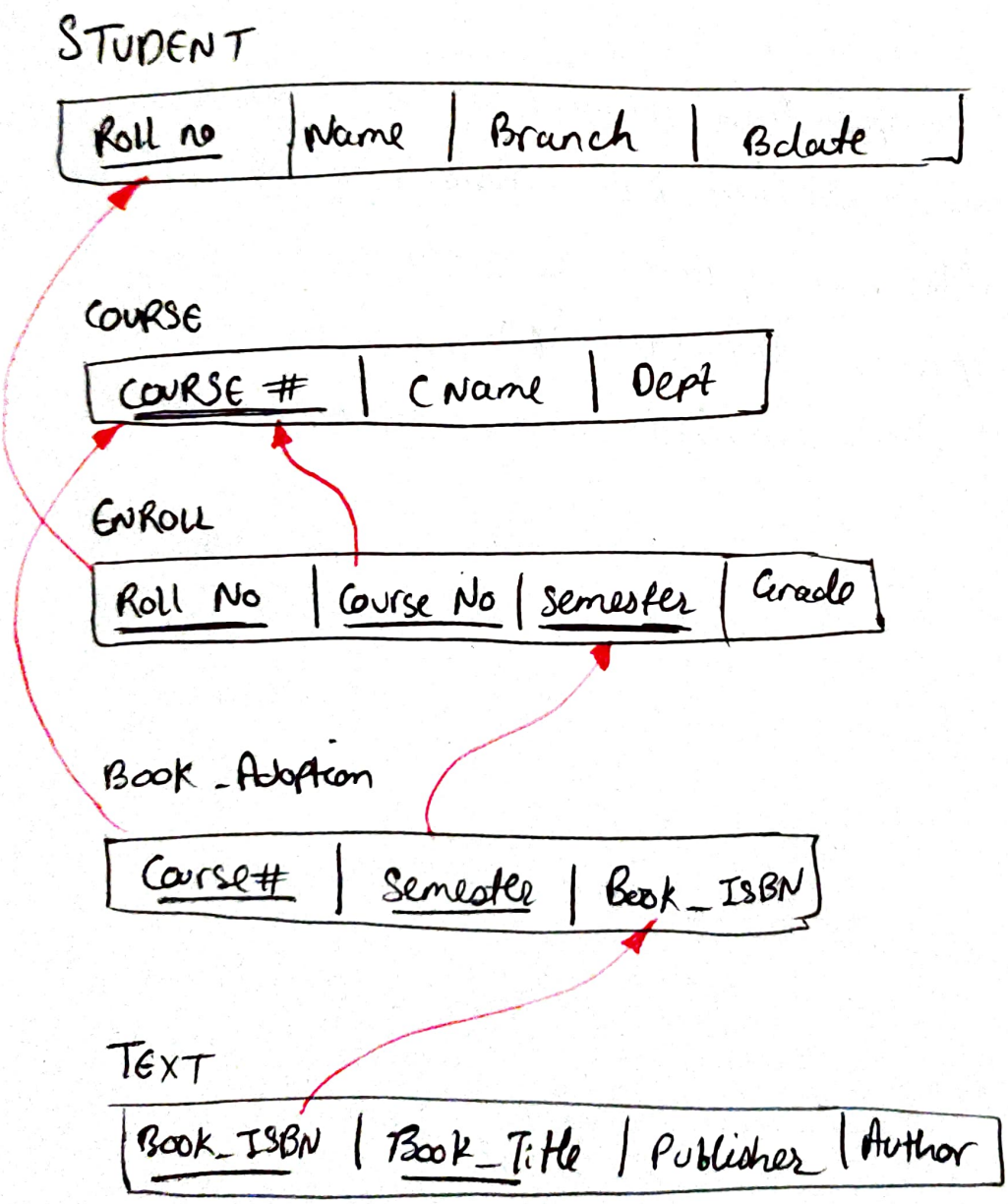
Referential Integrity Constraint

- A Constraint involving two relations
- Used to specify a relationship among tuples in two relations: referencing relation and referenced relation
- Tuples in referencing relation R_1 have attributes FK (called foreign key) that reference the primary key attributes PK of referenced relation R_2. tuple t_1 in R_1 is said to reference tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$

- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1.FK$ to R_2.

eg Book adoption by student

relational database schema diagram



Step 3
: Mapping of weak Entity

- for each weak entity type W in ER schema with owner entity type E, create a relation R & include all simple attributes of W as attributes of R.
- In addition, include as foreign key attributes of R the primary key attribute(s) of relation that corresponds to owner entity type.
- Primary key of R is combination of primary key(s) of owner(s) & partial key of weak entity type W, if any

eg Create relation DEPENDENT, include primary key SSN of Employee relation as foreign key attribute of Dependent (renamed to ESSN)
So primary key of DEPENDENT relation is {ESSN, DEPENDENT_Name}

Step 4 :
Mapping of Binary 1:N Relationship Type

- for each regular binary 1:N relationship type R, identify relation S that represent participating entity type at N-side of relationship type
- Include as foreign key in S the primary key of relation T that represents other entity type participating in R
- Include any attribute of 1:N relation type attributes of S.

eg: 1:N relationship types WORKS_FOR. here we include primary key DNUMBER of DEPARTMENT relation as foreign key in EMPLOYEE relation & call it DNO.

Steps

Mapping of Binary M:N relationship types.

- for each regular binary M:N relationship type R, create a new relation S to represent R.
- Include as foreign key attributes in S the primary keys of relations that represent participating entity types, their combination will form primary key of S.
- Also include any simple attributes of M:N relationship type as attributes of S.

eg WORKS_ON

Primary key is {ESSN, PNO} — fk of Employee & project.

→ WORKS_ON {ESSN, PNO, hours}

STEP 6

Mapping of Binary 1:1 relation Types

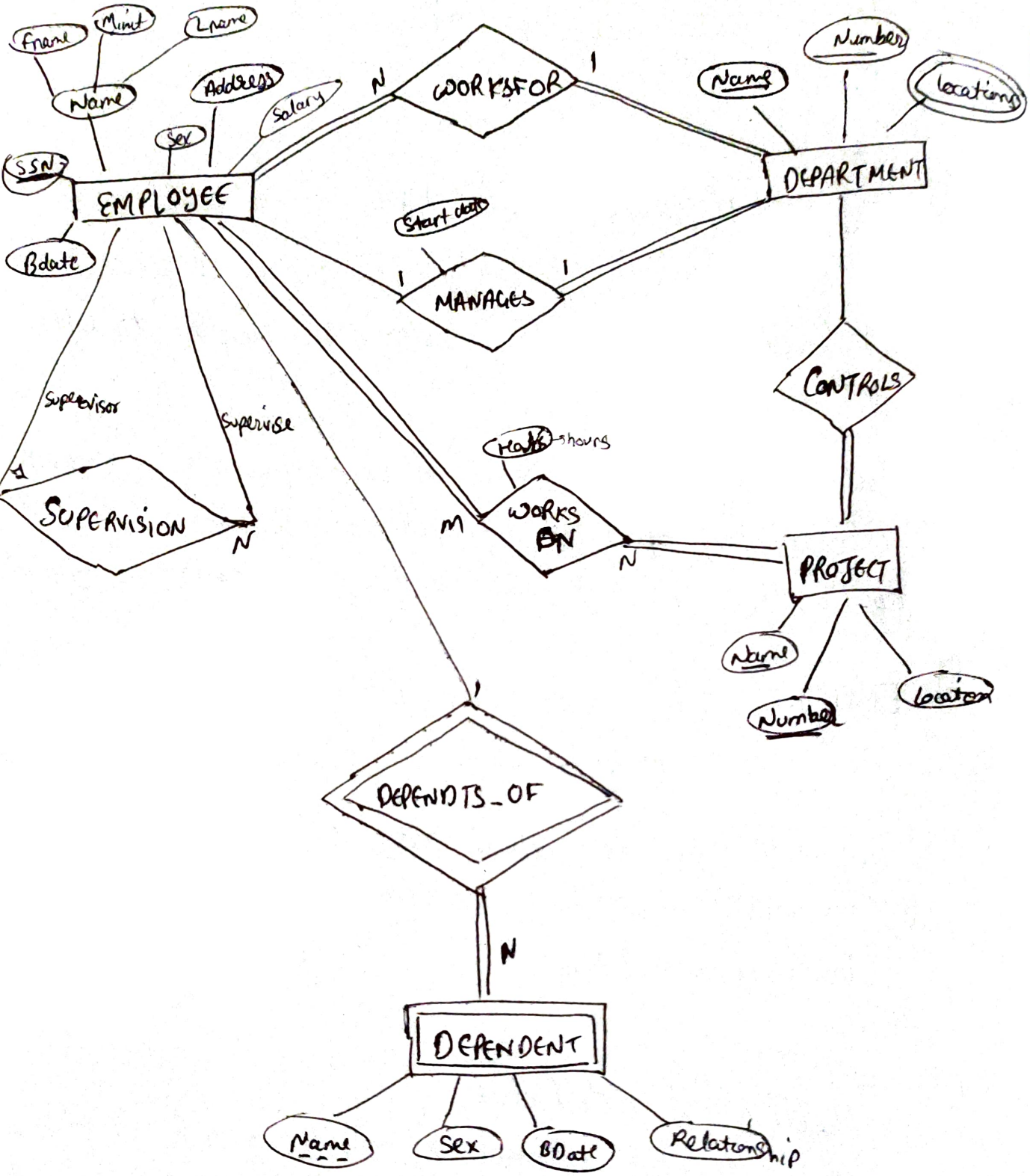
Let relations S and T that corresponds to entity types participating in R. There are 3 possible approaches.

1) Foreign key approach :-

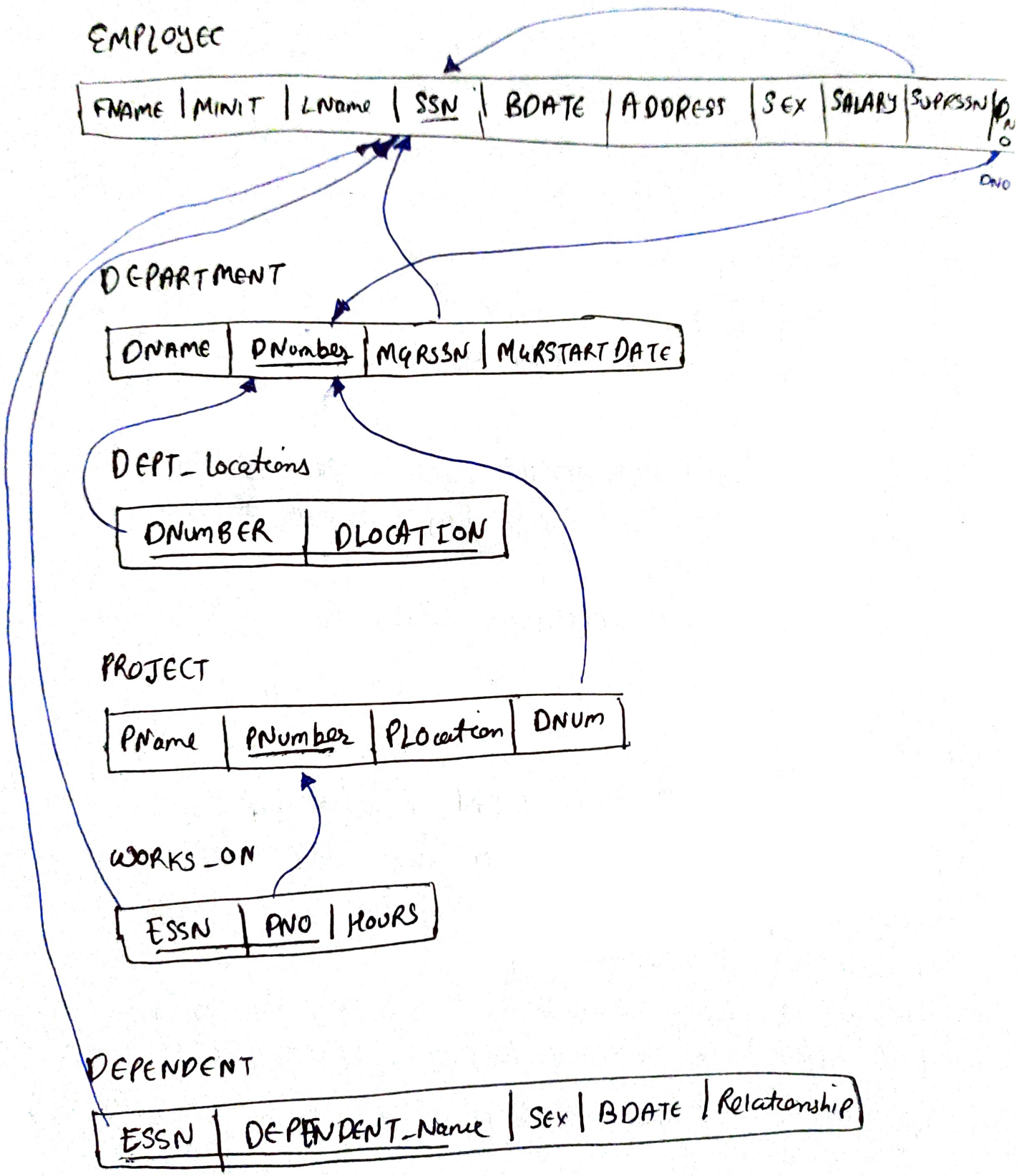
Chose one of the relation S, say - and include foreign key in S the primary key of T. It is better to chose an entity type with total participation in R in role of S.

eg 1:1 relation MANAGES, is mapped by choosing participating entity type DEPARTMENT to serve in role of S, because its participation in MANAGES relationship type is total.

2) Merged relation option : An alternating mapping of 1:1 relationship type is possible by merging two entity types & relationship into single relation. Use it when both participations are total

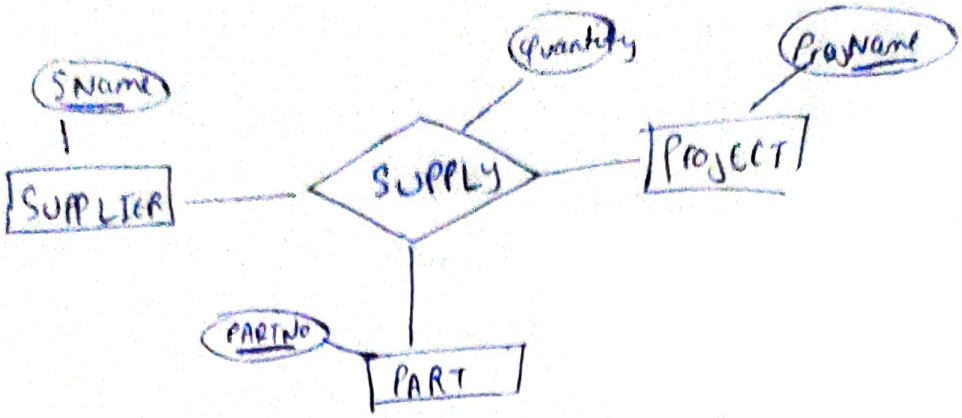

Fname
Minit
Lname
Name
Address
Salary
Sex
SSN
EMPLOYEE
Bdate
N
WORKS FOR
1
Number
Name
Locations
DEPARTMENT
Start date
1
MANAGES
1
CONTROLS
Supervisor
Supervise
SUPERVISION
N
Hours
M
WORKS ON
N
PROJECT
Name
Number
Location
1
DEPENDTS_OF
N
DEPENDENT
Name
Sex
BDate
Relationship

ER schema into relational schema



- For N-ary relationship

eg



SUPPLIER

SName	...

PROJECT

PROJName	...

PART

PARTNO	...

SUPPLY

SNAME	PROJName	PARTNO	Quantity

ER Model

- Entity type
- 1:1 or 1:N relationship type
- M:N relationship type
- value set
- key attribute
- Multivalued attribute
- Composite attribute

Relational Model

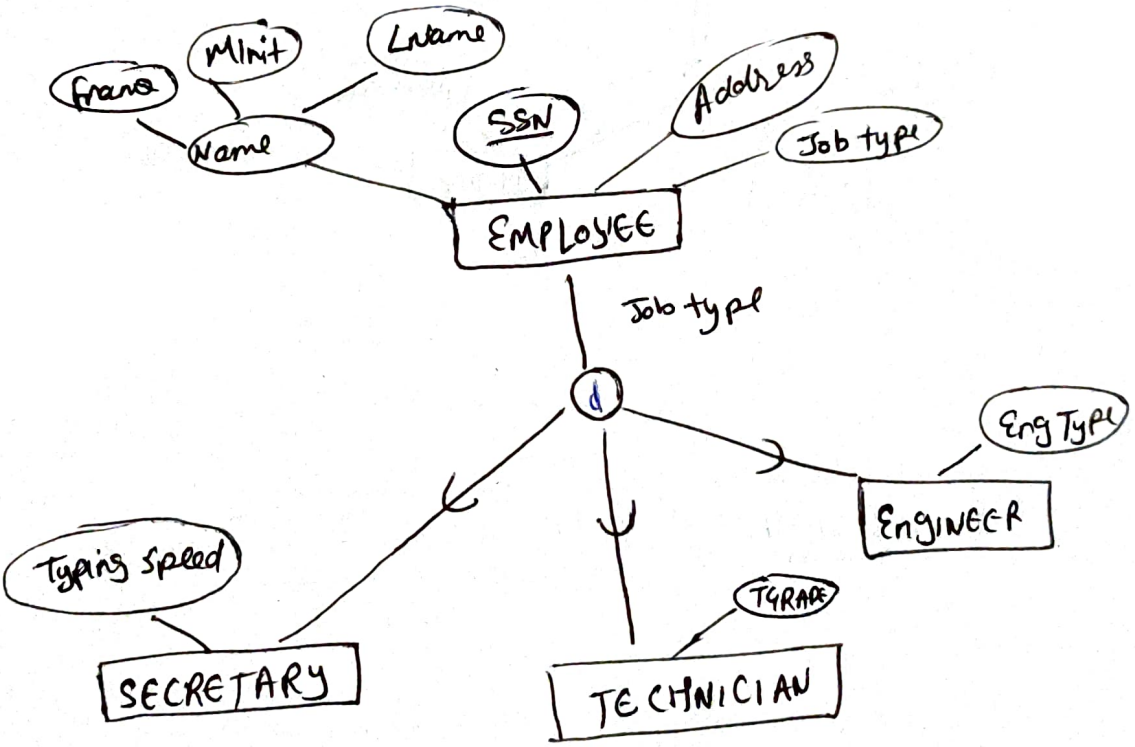
- "Entity" relation
- Foreign key (or "relationship" relation)
- "Relationship" relation & 2 foreign key
- Domain
- Primary key
- Relation & foreign key
- Set of simple component attributes

• EER to Relational

Method 1

Multiple relations - Superclass & subclass

Create a relation L for C with attributes Attrs(L) = {k, a1, ..an} and PK(L) = k. Create a relation Li for each subclass Si, 1 < i < m, with attributes Attrs(Li) = {k} ∪ {attributes of Si} and PK(Li) = k. It works for any Specialization (total, partial, disjoint, overlapping)



→ EMPLOYEE

SSN	FName	Minit	LName	BDate	Address	JobType

SECRETARY

SSN	TypingSpeed

TECHNICIAN

SSN	GRade

ENGINEER

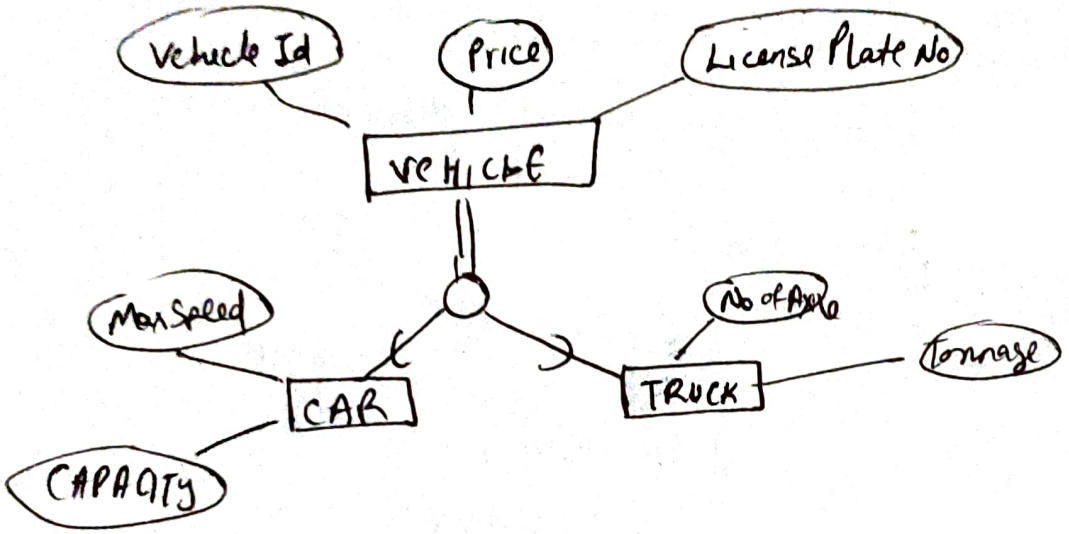
SSN	EngType

Method 2

Multiple relations - Subclass relations only.

Create a relation L_i for each subclass S_i, $1 < i < m$, with attributes $Attrs(L_i) = \{\text{attributes of } S_i\} \cup \{k, a_1, a_2, \ldots a_n\}$ and $PK(L_i) = k$. This option works for specialization whose subclasses are total (every entity in superclass must belong to (at least) one of the subclasses)

eg



CAR

VehicleId	LicensePlateNo	Price	MaxSpeed	CAPACITY

TRUCK

VehicleId	LicensePlateNo	Price	No of Axle	Tonnage

DBMS college Notes (2)

Topic → ER modeling, Enhanced ER Specialization Generalization

ER model

eg Company Database

- Requirements of the company

1) The company is organized into DEPARTMENTS. Each department has a name, number and an employee who manages the department. We keep track of start date of department manager.
2) Each department controls number of PROJECTS. Each project has a name, number & is located at single location.

ER model concepts

Entities & Attributes

- Entities are specific objects or things in mini-world that are represented in database. For eg the EMPLOYEE John Smith, Research DEPARTMENT, the product X PROJECT.
- Attributes are properties used to describe an entity. for eg an EMPLOYEE entity may have NAME, SSN, Address, Sex, Birth Date.
- A specific entity will have a value for each of its attributes for ex. a specific employee entity may have Name = 'John Smith', SSN = '12345', Address = '731, fondren, Houston', Sex = 'M'.

Each attribute has a value set (or data type) associated with it eg Integer, String, Subrange, enumerated type.

Types of Attributes

1) Simple
Each entity has a single atomic value for attribute eg SSN, Sex

2) Composite
The attribute may be composed of several components. for eg Name (FirstName, MiddleName, LastName), Address (Apt#, House#, Street, city, State, Zipcode, Country).

3) Multivalued
An entity may have multiple values for attribute. for eg COLOR of a CAR or Previous Degrees of a STUDENT Denoted as {COLOR} or {Previous Degrees}.

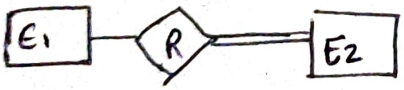
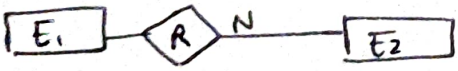
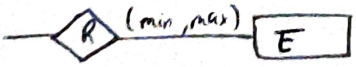
Nesting between them is possible

eg Previous Degrees of a STUDENT is composite multi-valued attribute denoted by {Previous Degrees (college, year, degree, field)}

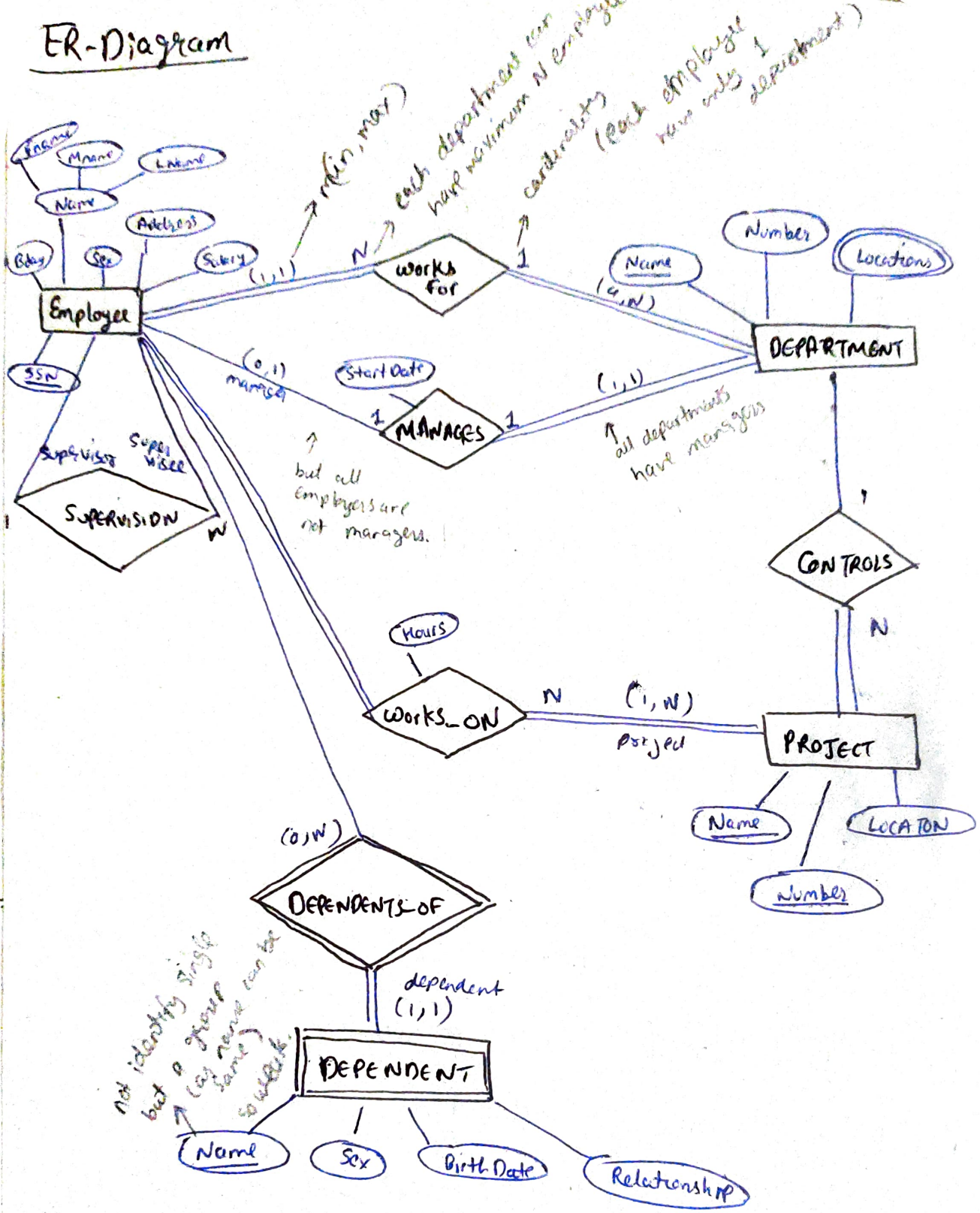
Relationships & Relationship types

- A relationship relates two or more distinct entities with a specific meaning. for eg EMPLOYEE JohnSmith works on ProductX PROJECT or EMPLOYEE Franklin Wong manges the Resarch Department.
- The degree of relationship type in number of participating entities type. Both MANAGES & WORK_ON are binary relationship.

Notations for ER Schemas

Symbol	Meaning
	Entity Type
	Weak Entity Type
	Relationship Type
	Identifying relationship type
	Attribute
	Key Attribute
	Multivalued attribute
	Composite attribute
	Derived attribute (like age can be derived from DoB.)
	Total participation of E_2 in R
	Cardinality ratio of 1:N for $E_1 : E_2$ in R
	Structural Constraint (min, max) on Participation of E in R.

ER-Diagram



- Relationship types are : works_for , Manages , Work_on , Controls , SUPERVISION , dependents_of
- weak entity type is : Dependent
  Identifying relationship is dependents_of
- Recursive Relationship type is : - SUPERVISION

Weak Entity types

→ A entity that does not have key attribute

- A weak entity must participate in identifying relationship with owner or identifying entity type.
- Entities are identified by combination of:
  - A partial key of weak entity type
  - The particular entity they are related to in identifying entity type.

eg DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via identifying relationship type DEPENDENT_OF.

Constraints on Relationships

Cardinality Ratio

Specifies maximum number of relationship instances that an entity can participate in.

- One-to-one (1:1)
- One-to-many (1:N) or Many-to-one (N:1)
- Many-to-Many

Participation

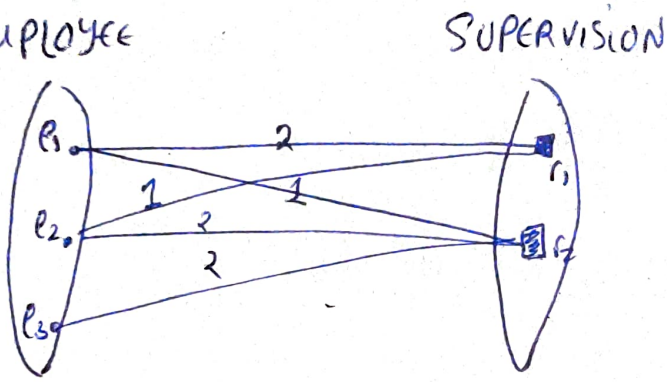
- Also called minimum cardinality constraint
- Specifies minimum number of relationship instances that each entity can participate in.

* Weak entity always comes with total participation

- Recursive relationship type

1) Both participations are same entity type in different roles.
2) Eg: SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) & (another) EMPLOYEE (in role of subordinate)
3) In ER diagram, need to display role names to distinguish participations.

In following figure, first role participation labeled with 1 and second role participation labeled with 2.



A relationship type can have attributes, for example, HoursPerWeek of WORKS_ON; its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on PROJECT.

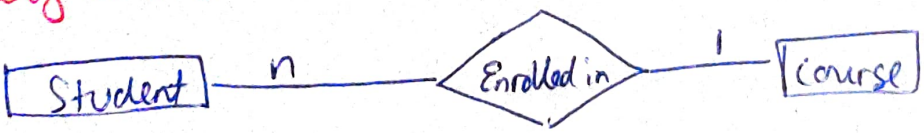
Alternative (min, max) notations for relationship structural constraints

- Specified on each participation of an entity type E in relationship type R.
- Specifies that each entity e in E participates in at least min & at most max relationship instances in R.

- Default (no constraint) : min = 0 , max = n
- Eg.
  - A department has exactly one manager & an employee can manage at most one department
    - Specify (0, 1) for participation of EMPLOYEE in MANAGES
    - Specify (1, 1) for participation of DEPARTMENT in MANAGES

- An employee can work for exactly one department but a department can have any no. of employee.
  - Specify (1,1) for participation of Employee in WORKS_FOR
  - Specify (0,N) for participation of DEPARTMENT in WORKS_FOR

* Many to one relationship



- n students enrolled in one course
- for one student there is one course

Enhanced - ER (EER) Model (E2R or EER)

- Includes all modeling concept of basic ER.
- Additional concepts : subclass / super classes, specialization generalization, categories, attribute inheritance
- It includes some object-oriented concepts like inheritance.

Subclasses or Superclasses

- An entity may have additional meaningful subgroupings of its entities.

eg EMPLOYEE may be further grouped into SECRETARY, ENGINEER, MANAGER, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE.

- Each is called subclass of EMPLOYEE
- EMPLOYEE is superclass for each of these subclasses.

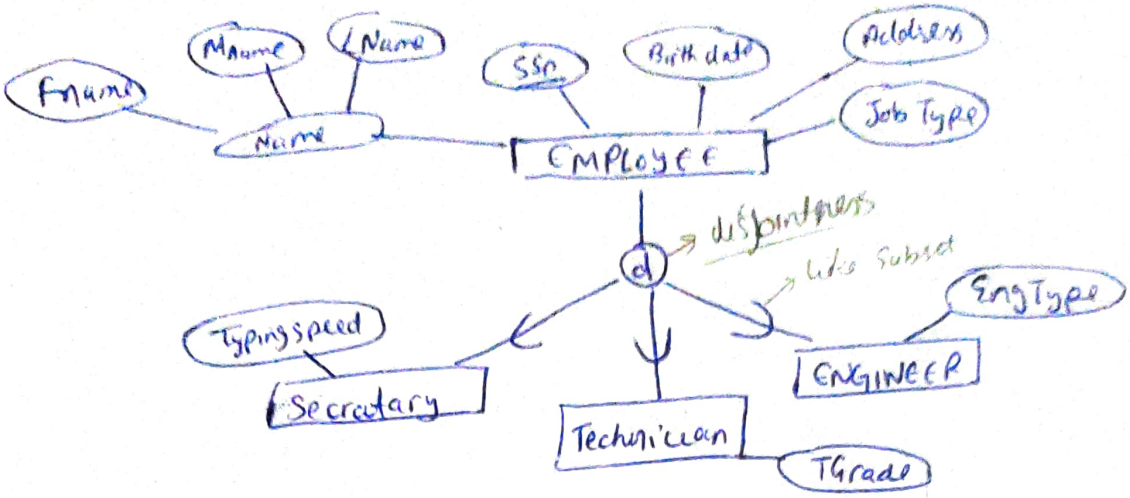
→ These are called superclass / subclass relationship
→ EMPLOYEE / SECRATARY, EMPLOYEE / TECHNICIAN

- These are also called IS-A relationship
  eg Secretary IS-A employee.
- An entity that is member of subclass inherits all attributes of entity as a member of superclass, it also inherits all relationships

Specialization

- Is process of defining a set of subclasses of superclass.
- Ex. {Secretary, Engineer, Technician} is specialization of EMPLOYEE based upon job type.
- Another specialization of EMPLOYEE based on Method of pay is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.

Attribute of Subclass are called specific attributes. For eg. TypingSpeed of SECRETARY.



Generalization

- Reverse of specialization process
- eg VEHICLE is generalization of CAR and TRUCK

- **Disjointness constraint**
  - Specifies that the subclasses of specialization must be disjointed (an entity can be member of at most one of the subclasses of of specialization)
  - Specified by d in EER diagram
  - If not disjointed, overlap; that is the same entity may be a member of more than one subclass of specialization.
  - Specified by o in EER digram.
- **Completeness Constraint**
  - Total specifies that every entity in Superclass must be a member of some subclass in specialization / generalization
  - Shown in EER digram by double line
  - Partial allows an entity not to belong to any of subclass
  - Shown in EER diggrams by single line

Hence we have 4 types of specialization / generalization

1) Disjoint, total 2) Disjoint, partial 3) Overlapping, total
4) Overlapping, partial

→ Generalization usually is total because superclass is derived from subclasses.

⇒ Hierarchy has a constraint that every subclass has only one superclass (single inheritance)

⇒ In a Lattice, a subclass can be subclass of more than one superclass (multiple inheritance)

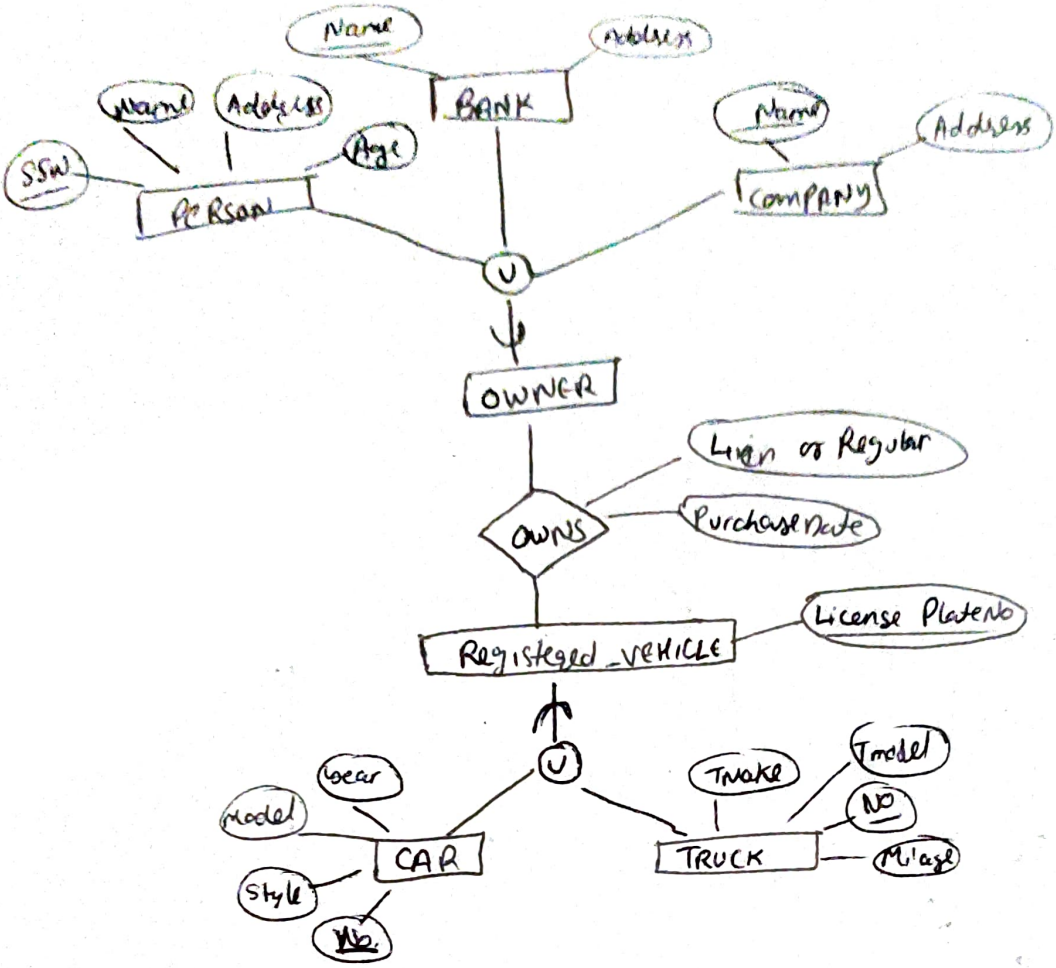
In lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of its predecessor superclass.

Categories (Union types)

- A shared subclass is subclass in more than one distinct superclass / subclass relationships where each relationships has a single superclass (multiple inheritance)
- Superclasses represent different entity type. Such a subclass is called a category or UNION Type.
  eg. Database for vehicle registration, vehicle owner can be person, a bank or a company.

→ Category (subclass) OWNER is subset of union of 3 superclasses COMPANY, BANK & PERSON

→ A category member must exist in at least one of its superclasses.



FORMAL DEFINATION of EER

- **Class C** : A set of entities ; could be entity type, subclass, superclass, category.
- **Subclass S** : A class whose entities must always be subset of entities in another class, called the superclass C of superclass / subclass (or IS-A) relationship S/C :

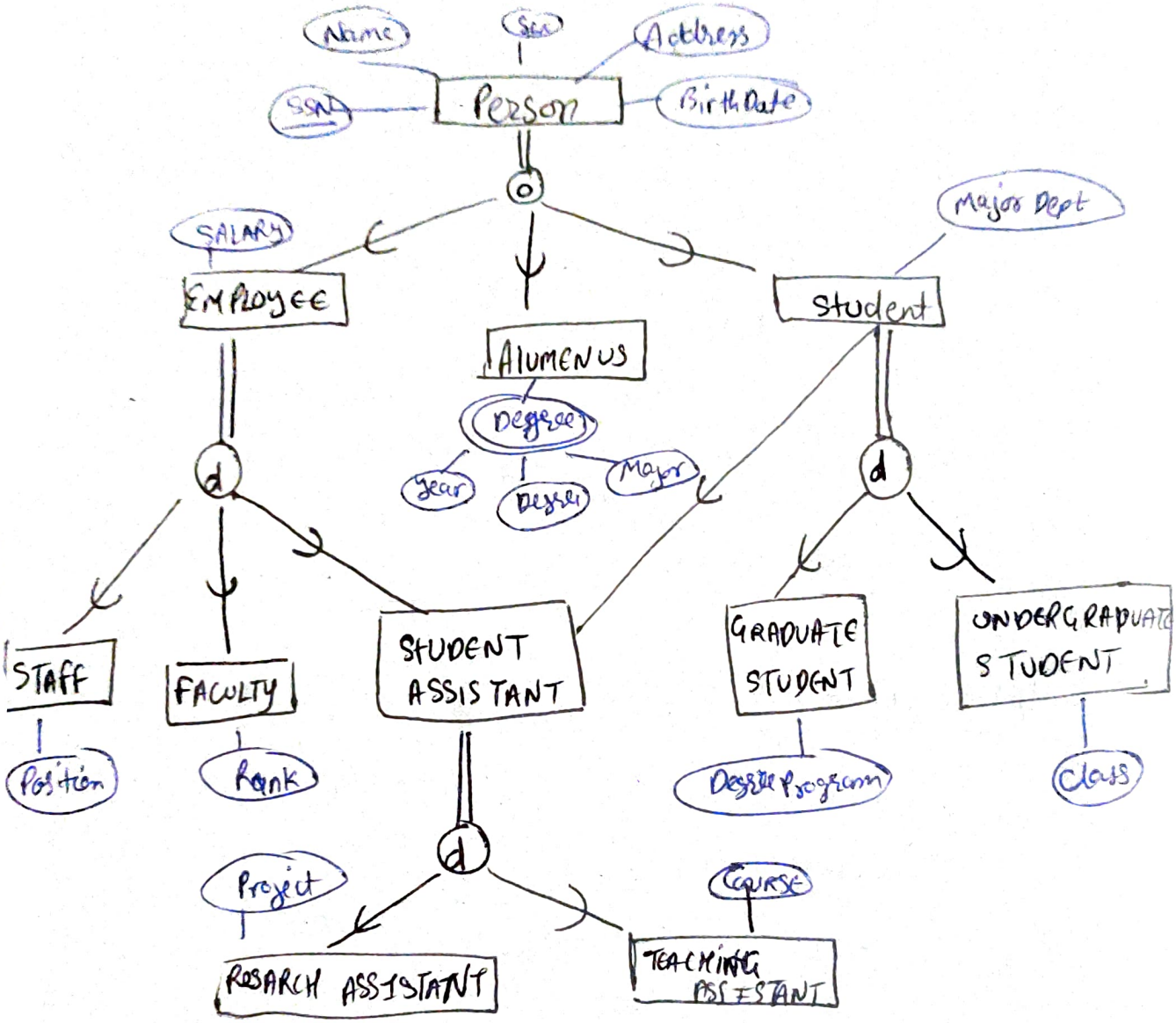
  $S \subseteq C$.

- **Specialization Z** : $Z = \{S_1, S_2 - - S_n\}$ a set of subclasses with same superclass G, hence G/S_i is superclass relationship for $i = 1, 2 - - n$
  - G is called generalization of subclasses $\{S_1, S_2 - - S_n\}$
  - Z is total if we always have :
    $S_1 \cup S_2 \cup - - S_n = G$ otherwise Z is partial
  - Z is disjoint if we always have :
    $S_i \cap S_j$ empty set for $i \neq j$ otherwise Z is overlapping

With Specialization / Generalization concept

UNIVERSITY

O - overlapping



difference betn o & d :-

- The disjoint rule states that an entity instance of a supertype can only be a member of one subtype. The overlap rule states that an entity instance of supertype can be member of multiple subtype.

DBMS college notes (1)

Topic → Basic, classifications of DBMS, DBMS Architecture

---

1) Basic

- **Data** → known fact that can be recorded & have an implicit meaning.
- **Database** → A collection of related data representing some aspects of real world.
- **Miniworld** → Some part of real world about which data is stored in database.
- **Database Management System** → General purpose software system that facilitates the purpose of defining, constructing & manipulating databases for various applications
  Eg: MySQL, Oracle

1. Defining database – specifying data types, structures & constraints for data
2. Constructing database or Load database – Process of storing data on some medium that is controlled by DBMS.
3. Manipulating a database – functions like querying, updating, insertions, deletions

3) Example of Database

Mini world for example = University Environment

Some mini-world entities:

1) Students
2) Courses
3) Sections (of courses)
4) (academic) Departements
5) INSTRUCTORS

above could be expressed in Entity-Relationship data model

Some mini-world Relationships :-

- SECTIONS are of specific COURSES
- STUDENTs take SECTIONs
- COURSES have prerequisite COURSES
- INSTRUCTORS teach SECTIONs
- STUDENTS major in DEPARTMENTS

it can be represented as ER model

Student	Name	Students no.	class	Major
	Smith	17	1	CS
	Brown	8	2	CS

Grade	Students No.	Section Identifier	Grade
	17	112	B
	17	119	C
	8	85	A

Courses	Course Name	Course No.	Credit Hours	Department
	Intro to Programming	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Maths	MA2410	3	Maths
	Database	CS3380	3	CS

Prerequisite	Course Number	Prequisite
	CS3380	CS3320
	CS3380	MA2410
	CS3320	CS1310

Section	Section Identifier	Course No.	Semester	Year	Instructor
	85	MA2410	Fall	98	King
	92	CS1310	Fall	98	Andrew
	102	CS3320	Spring	99	Knuth
	112	MA2410	Fall	99	Chang
	119	CS1310	Fall	99	Stone

③ Relations

Relation_name	No_of_colums
STUDENT	4
COURSE	4
SECTION	5
GRADE	3
PREREQUISITE	2

Main characteristics of Database approach

1) Insulation betn program & data – Called program data independence. Allows changing data storage structures & operations without having to change DBMS access programs.

2) Data Abstraction – A data model is used to hide storage details & present the user with conceptual view of database.

3) Support of multiple views of data :- Each user may see a different view of database, which describe only data of interest of that user.

Database Users

Users are of 2 type

1. Those who actually use & control the content (called actor on the scene) eg: Database administrators, Database designers, End-users.
2. Those who enable database to be developed & DBMS software to be designed & implemented (called "Actors behind the scene") eg: System designers & Implementors, Tool Developers, operators & Maintenance Personnel.

DBMS ARCHITECTURE

- Data model → A set of concepts to describe structure of a database & certain constraints a database should poses.

Categories of data models :-

- Conceptual (high-level, semantic) data models :
  Provide concepts that are close to the way many users perceive data (Also called entity-based or object-based data models.)
- Physical (low-level, internal) data models :
  Provide concepts that describe details of how data is stored in computer.
- Implementation (representational) data models :
  Provide concepts that fall between the above two, balancing user views with some computer storage details.

Schemas vs instances :-

- Database Schema :- The description of a database. Includes description of data base structure & constraints that should hold on database.
- Schema Diagram :- A diagrammatic display of (some aspects of) database schema.
- Schema construct :- A component of schema or an object within the schema, eg :- STUDENT, COURSE
- Database instance :- The actual data stored in data at particular moment in time. Also called database state (or occurance)

⑤ Schema diagram

Schema is called 'intension', whereas state is called extension.

eg of Database State

COURSE

Name	Course No.	Credit	Department
Data Structure	CS 3320	4	CS
Discrete Maths	MA 2410	3	MATH

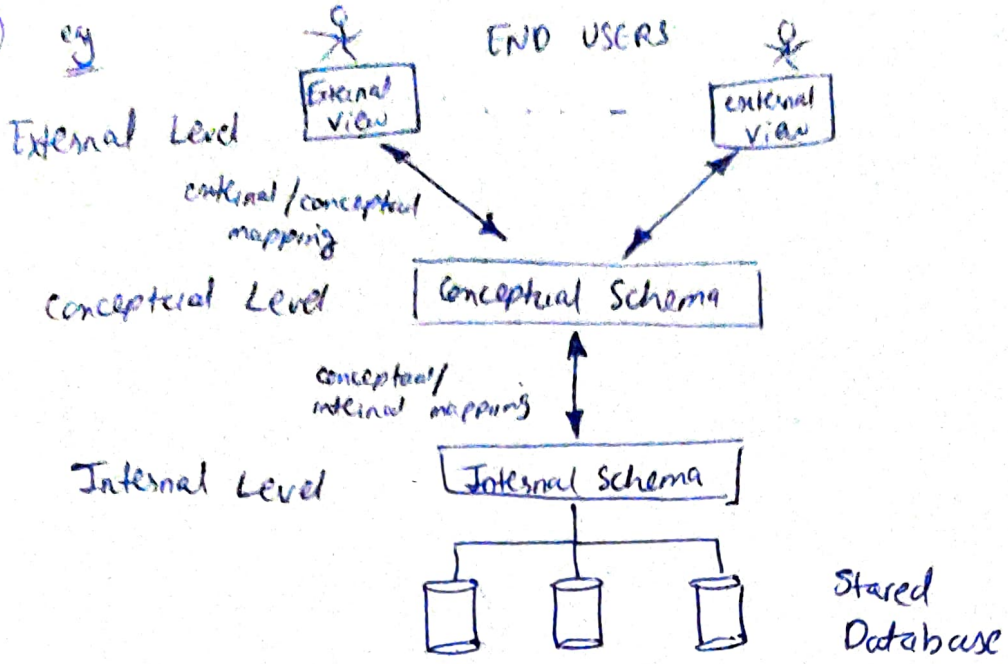
- **Three Schema Architecture**

Defines DBMS schemas at three levels.

1) Internal Schema :- at internal level to describe physical storage structures & access paths. Typically uses a physical data model.
2) Conceptual schema :- at conceptual level to describe the structure & constraints for whole database for a community of users. Uses a conceptual or an implementation data model.
3) External schemas :- at external level to describe various user views. Usually uses the same data model as conceptual level

→ Mappings among schema levels are needed to transform request & data. Program refer to an external schema, and are mapped by DBMS to internal schema for execution

⑥ eg



Data Independence

1) Logical Data Independence :- The capacity to change conceptual schema without having to change the external schemas and & their application programs.

2) Physical Data Independence :- The capacity to change internal schema without having to change conceptual schema

DBMS LANGUAGES

1) Data Definition Language (DDL): Used by DBA & database designers to specify conceptual schema of a database. In many DBMS, the DDL is also used to define internal & external schemas (views).

2) Data Manipulation Language (DML): Used to specify database retrievals & updates

- DML commands (data sublanguage) can be embedded in general-purpose programming language (host language) such as C or an Assembly language
- Alternatively, stand-alone DML commands can be applied directly (query Language)

⑦ Classification of DBMS

- Based on data model used :
  - Relational , Network , hierarchical
  - Object - oriented , Object - relational
- Other classifications :
  - Single - user (typically used with micro - computers) vs multi user .
  - Centralized (uses single computer with one database) vs distributed (uses multiple computers , multiple databases)