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

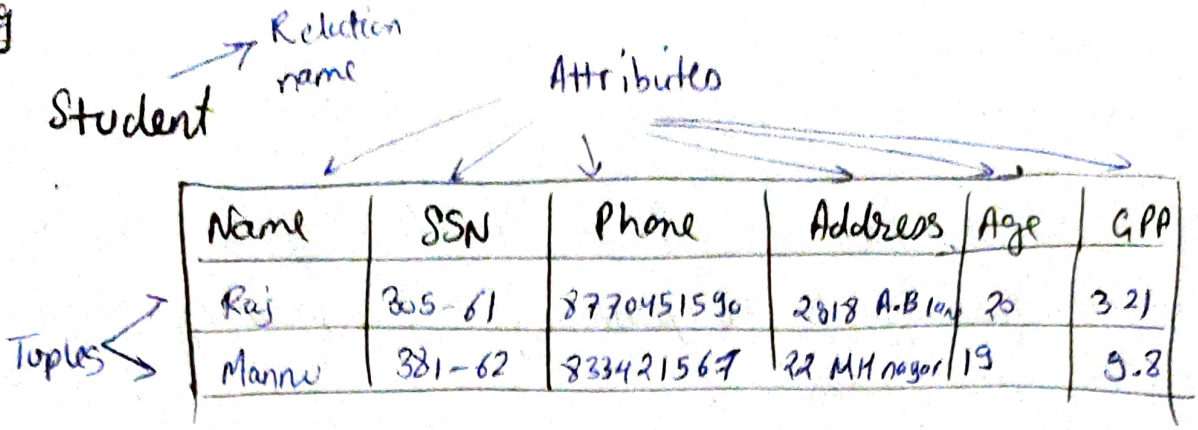
Let $R \subset S_1 \times S_2$

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.

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

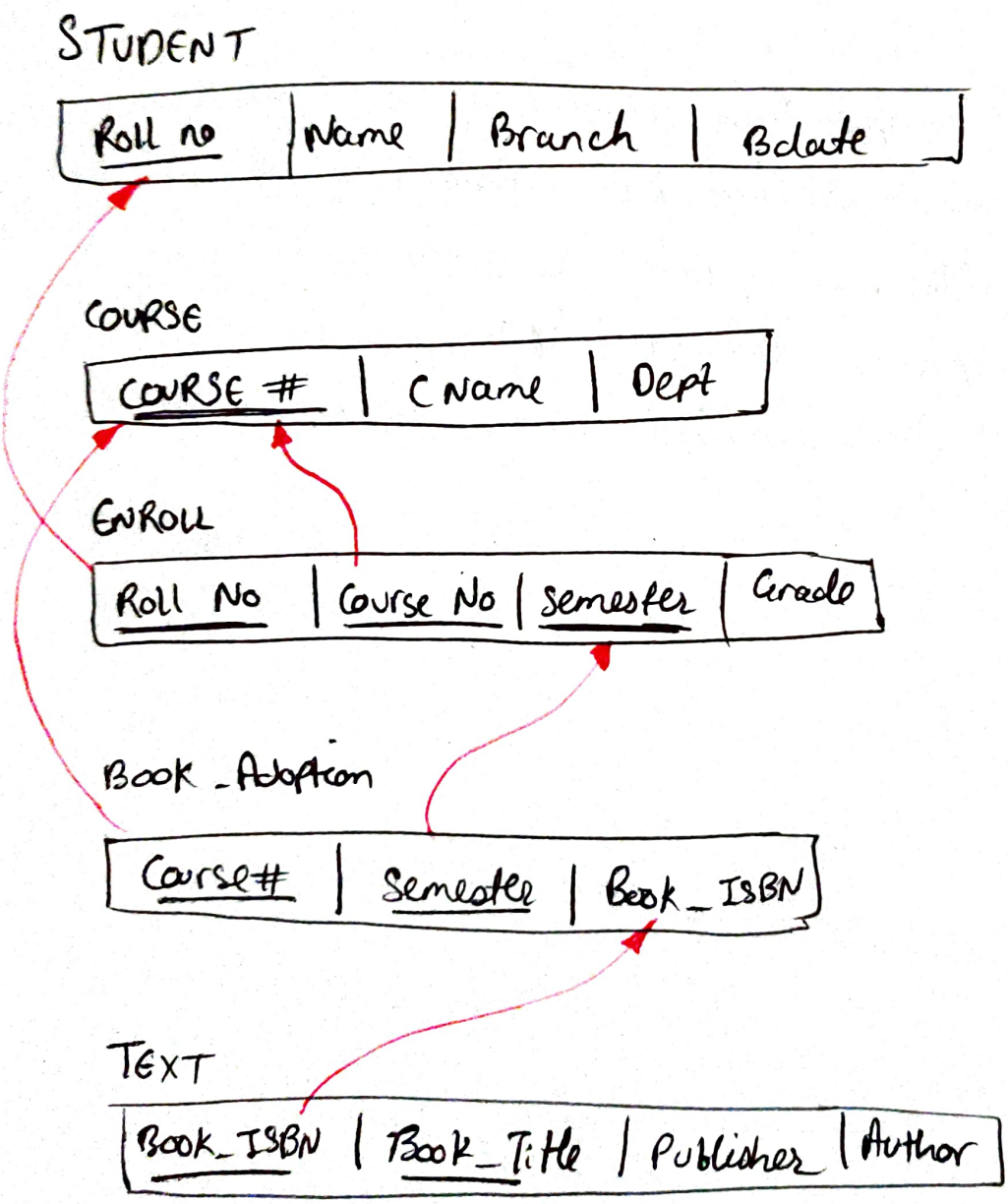
Referential Integrity Constraint

- 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

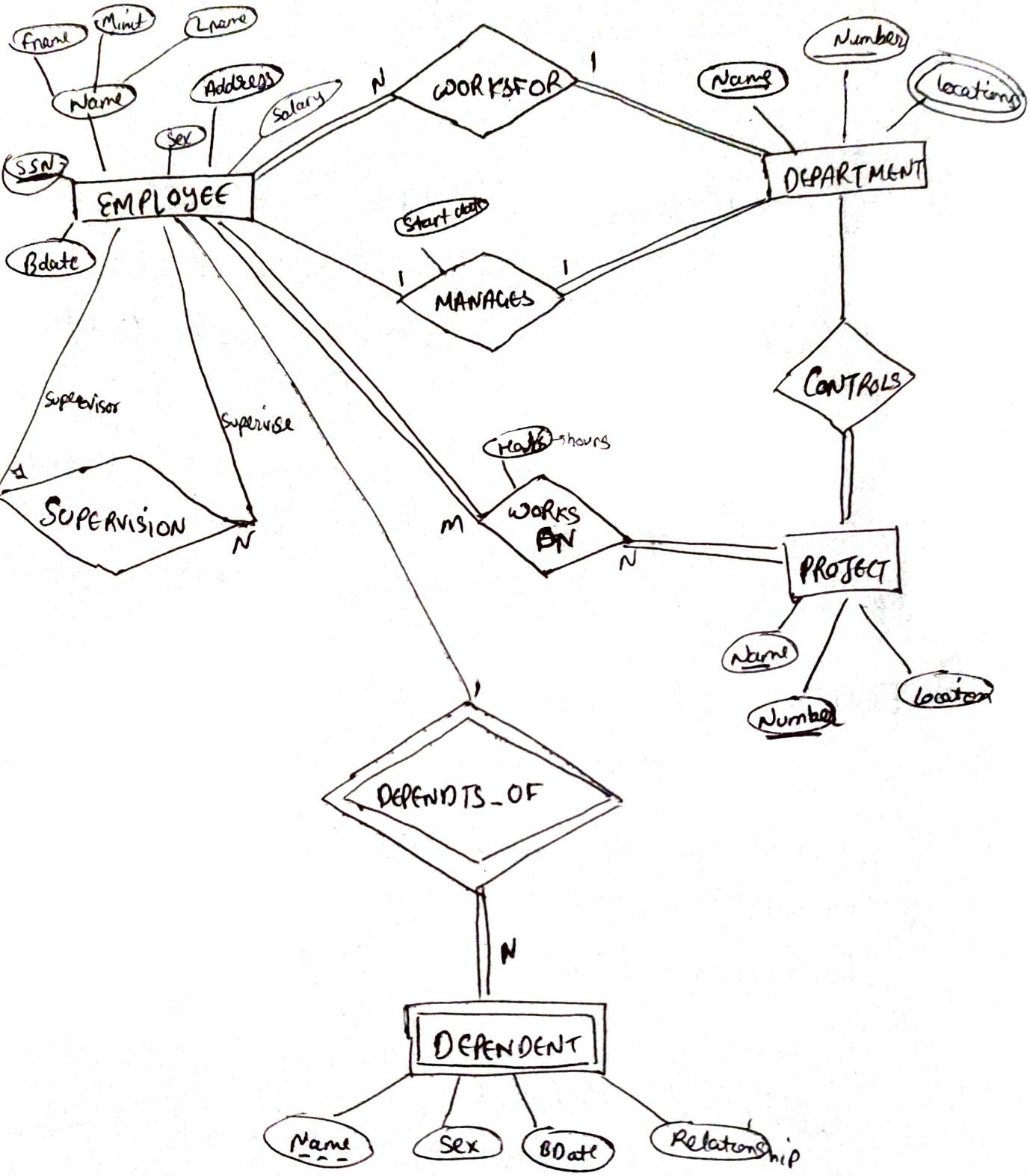
Let relations S and T that corresponds to entity types participating in R. There are 3 possible approaches.

1) Foreign key approach :-

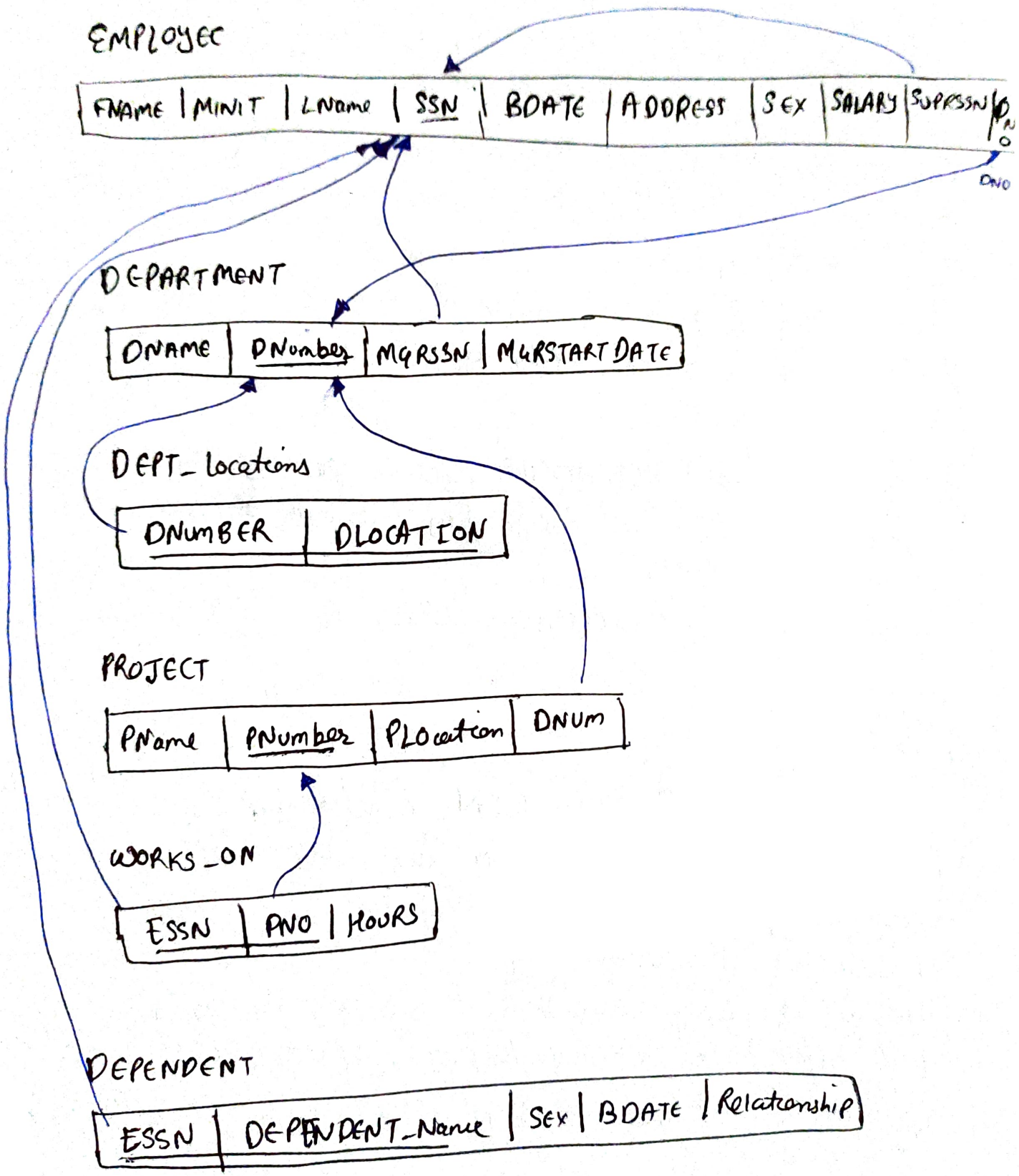
Chose one of the relation S, say - and include foreign key in S the primary key of T. It is better to chose an entity type with total participation in R in role of S.

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

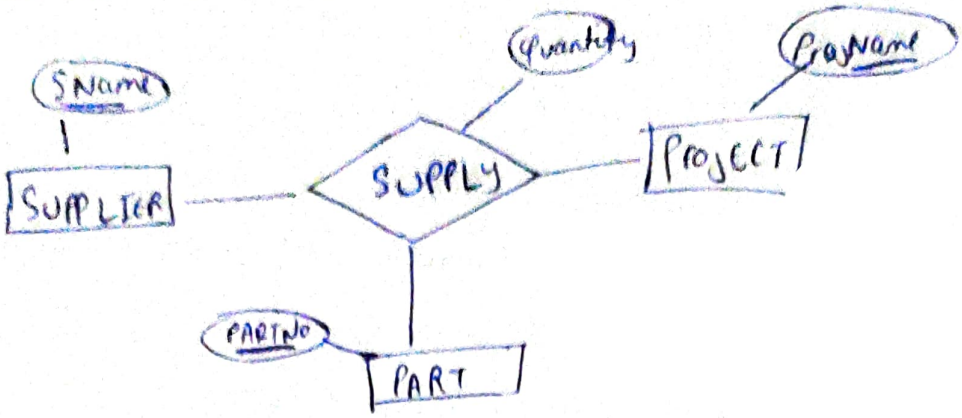

Fname
Minit
Lname
Name
Address
Salary
Sex
SSN
EMPLOYEE
Bdate
N
WORKS FOR
1
Number
Name
Locations
DEPARTMENT
Start date
1
MANAGES
1
CONTROLS
Supervisor
Supervise
SUPERVISION
N
Hours
M
WORKS ON
N
PROJECT
Name
Number
Location
1
DEPENDTS_OF
N
DEPENDENT
Name
Sex
BDate
Relationship

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

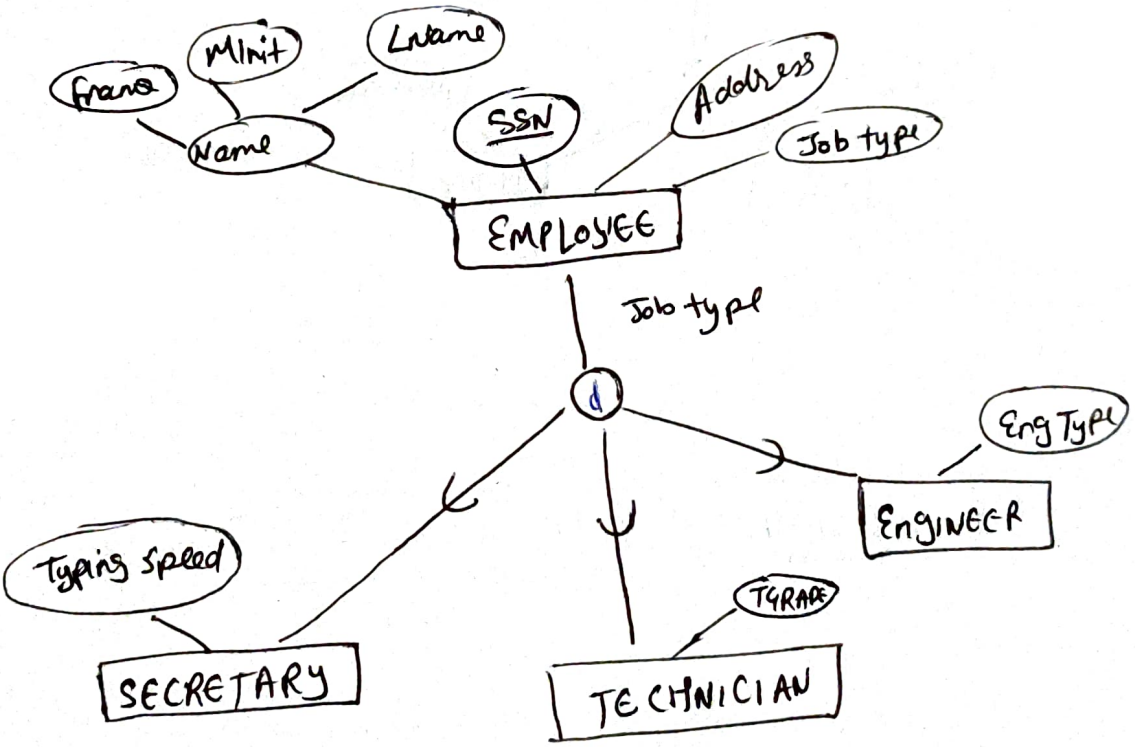
- "Entity" relation
- Foreign key (or "relationship" relation)
- "Relationship" relation & 2 foreign key
- Domain
- Primary key
- Relation & foreign key
- Set of simple component attributes

• 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

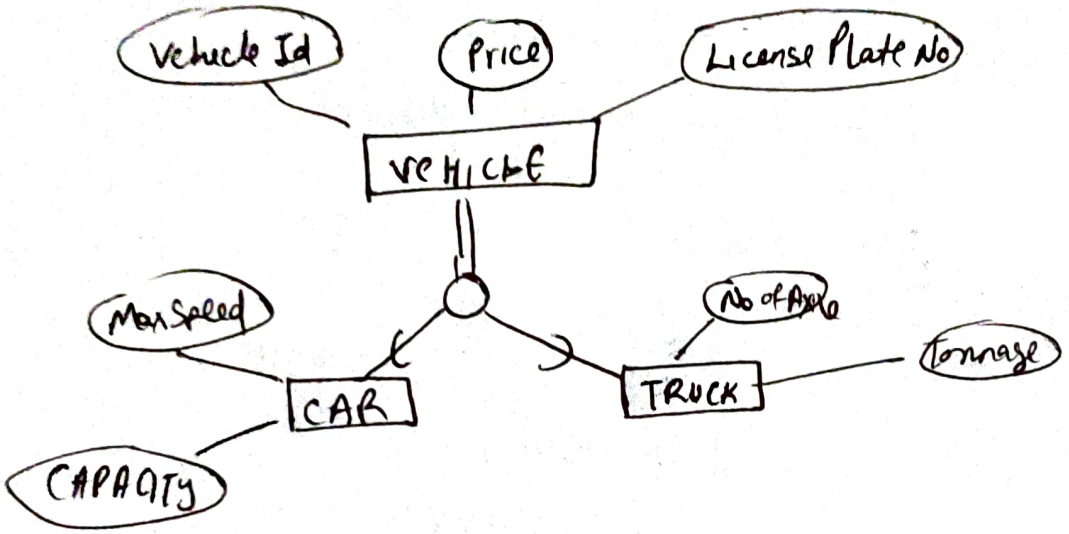
SSN	EngType

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