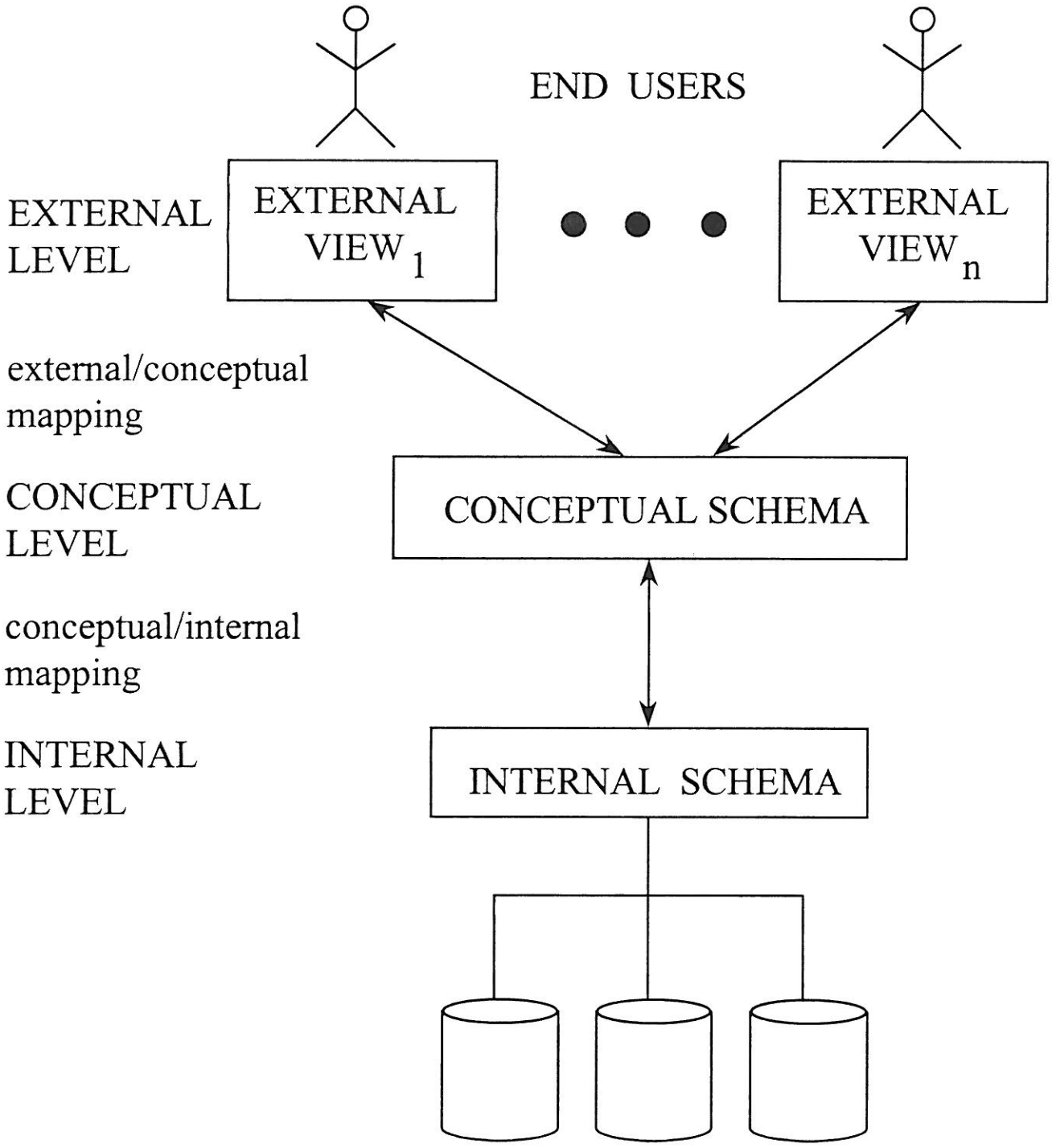
Introduction to Database Modeling

Peter Scheuermann

Dept. of EECS

peters@eecs.northwestern.edu



ANSI / SPARC ARCHITECTURE

OUTLINE:

1. DATABASE ARCHITECTURE FRAMEWORK
   file processing versus database approach

2. DATA MODELING WITH THE ENTITY-RELATIONSHIP APPROACH

3. INTRODUCTION TO THE RELATIONAL MODEL

4. TRANSFORMATION OF E-R SCHEMA TO RELATIONAL SCHEMA

5. NORMALIZATION IN THE RELATIONAL MODEL

6. OVERVIEW OF SQL

THE ENTITY-RELATIONSHIP MODEL

- DESCRIPTION OF THE "REAL WORLD" - an important tool in the communication among users, system analysts, database administrators

1. **ENTITY:** anything that can be distinctly identified, i.e. has existence in the real world

   - **ENTITY SET:** a group of entities with identical properties

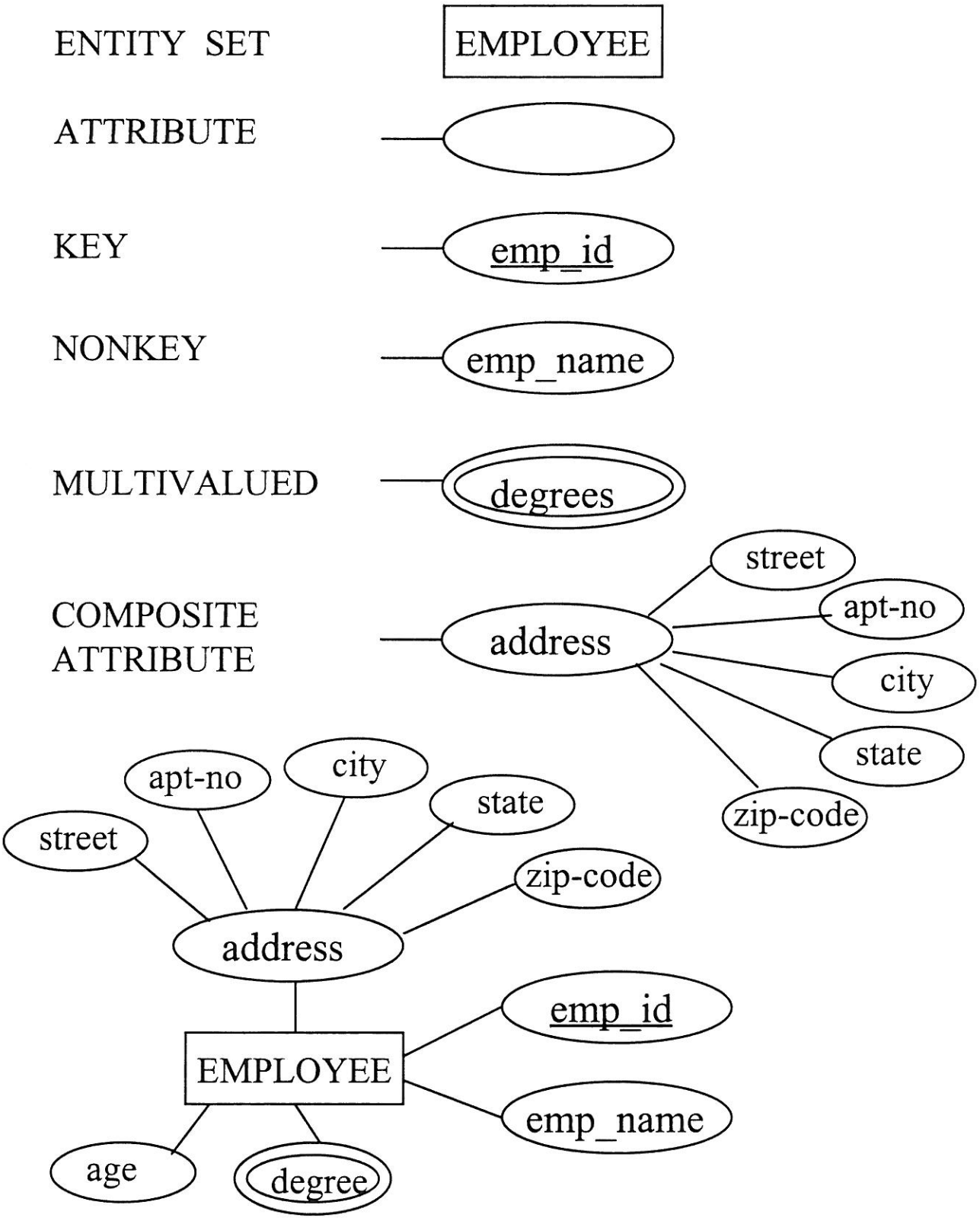
EMPLOYEE

   - **DESCRIPTION OF ENTITIES**
     - **ATTRIBUTES:** ~ characteristics of attributes that provide descriptive detail

     Ex: employee name, age, salary

     - each entity attribute can have *single* or *multiple values*

CONCEPTS AND THEIR REPRESENTATION



•**ENTITY KEY:** an attribute (or set of attributes) which uniquely identifies an entity within an entity set

•MOST ATTRIBUTES ARE *SINGLE VALUED*

•THE MEANINGS OF *NULL VALUES*:

(a) **NOT APPLICABLE**
single family home has no appt-no.

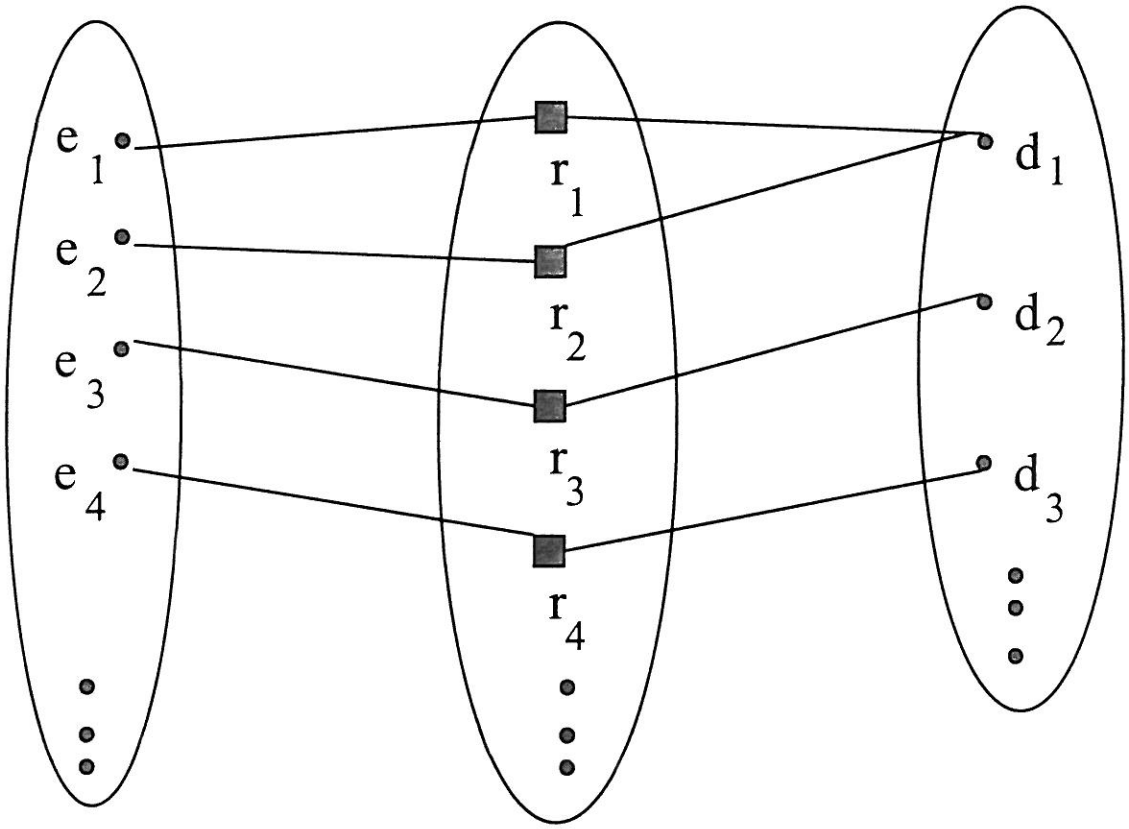
(b) **UNKNOWN**
we don't know the state where Mary Smith lives.

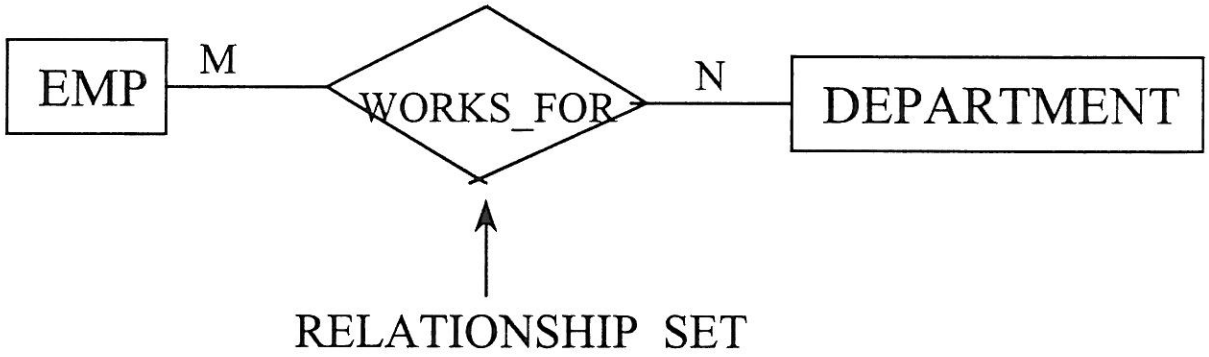
•**VALUE SETS** (DOMAINS) :

each simple attribute is associated with a value set which specifies the set of values that may be legally assigned to that attribute

Ex: age value set : [16 , 70]

2. RELATIONSHIP: association (mapping) among entities where association includes exactly one entity from each participating entity set





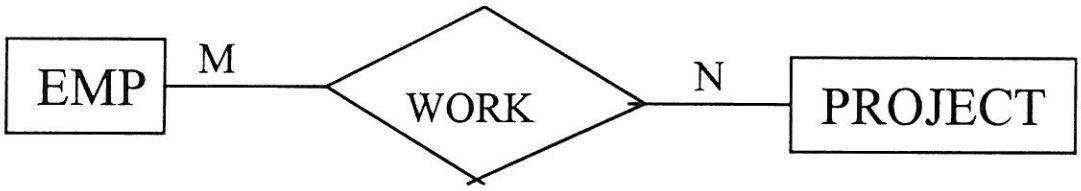
•**RELATIONSHIPS** ARE CHARACTERIZED IN TERMS OF

- •*degree*
- •*cardinality*
- •*constraints*
- •*attributes*

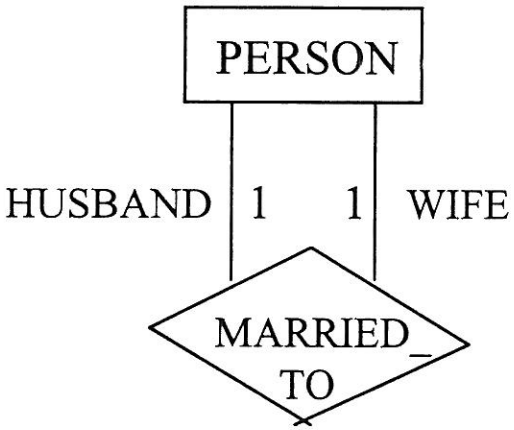
•**DEGREE OF RELATIONSHIP SET:**

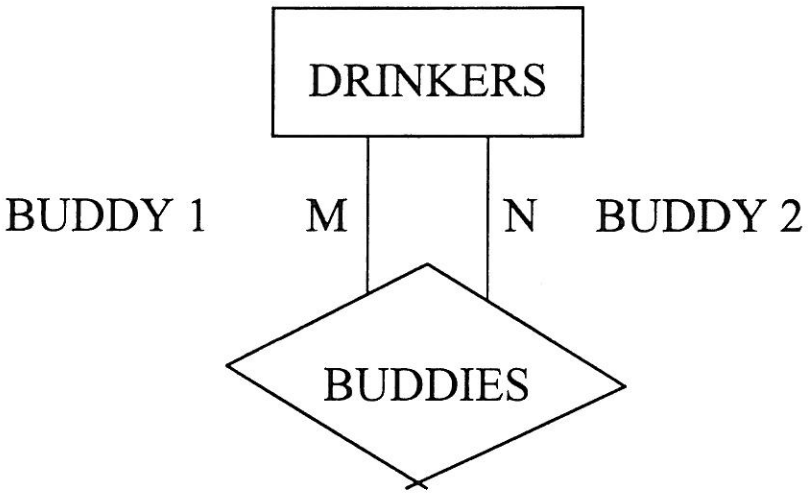
no. of participating entity sets

BINARY:



RECURSIVE BINARY:





Buddy 1	Buddy 2
d_1	d_2
d_1	d_3
d_2	d_1
d_2	d_4

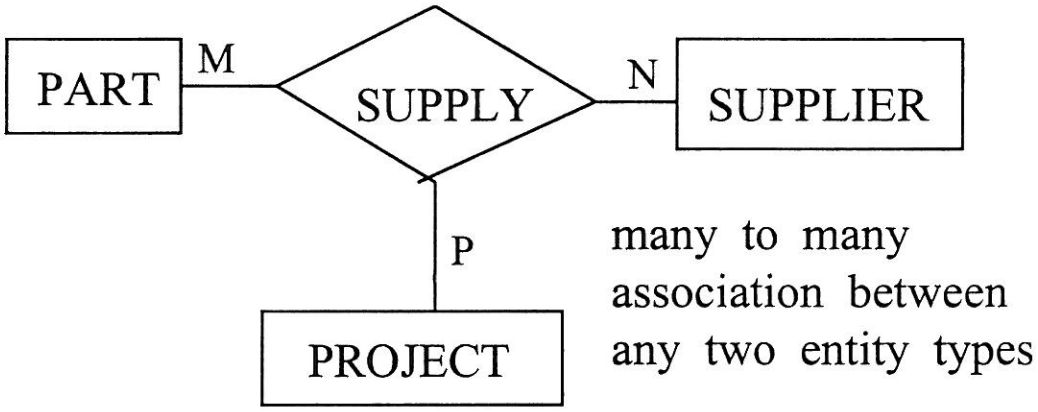
Buddies IS symmetric, *married* not

➢ no way to model "symmetric" in E/R

Design question:

Should we replace husband and wife by one relationship spouse?

TERNARY:



many to many association between any two entity types

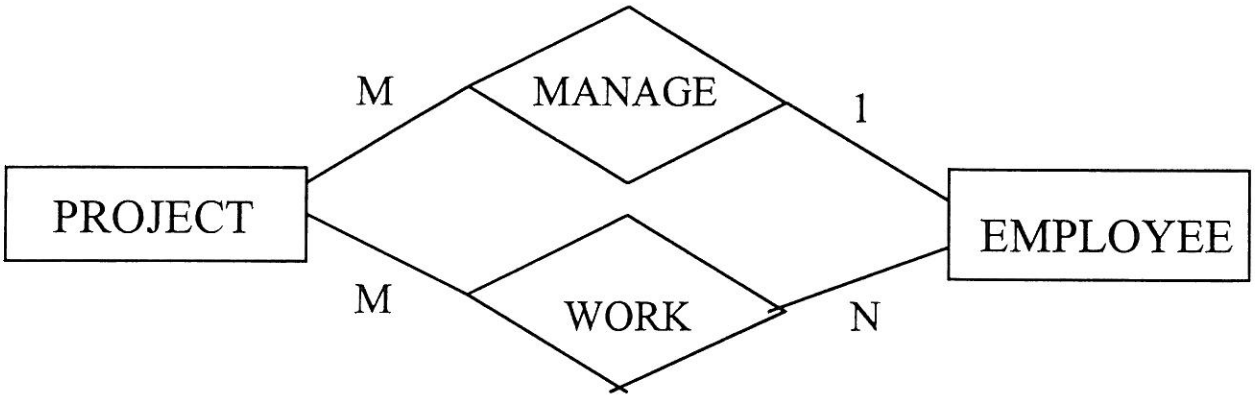
RELATIONSHIP SET TABLE:

PART#	SUPPLIER#	PROJ#
25	4	1
25	5	2
10	4	2
17	4	3
17	2	1
17	5	1

- *ternary* relationships (and higher) should be avoided if possible

- but *ternary* relationship is necessary when two (or three) *binary* relationships are not equivalent

•**CARDINALITY RATIO:** specifies the number of relationship instances that an entity can participate in

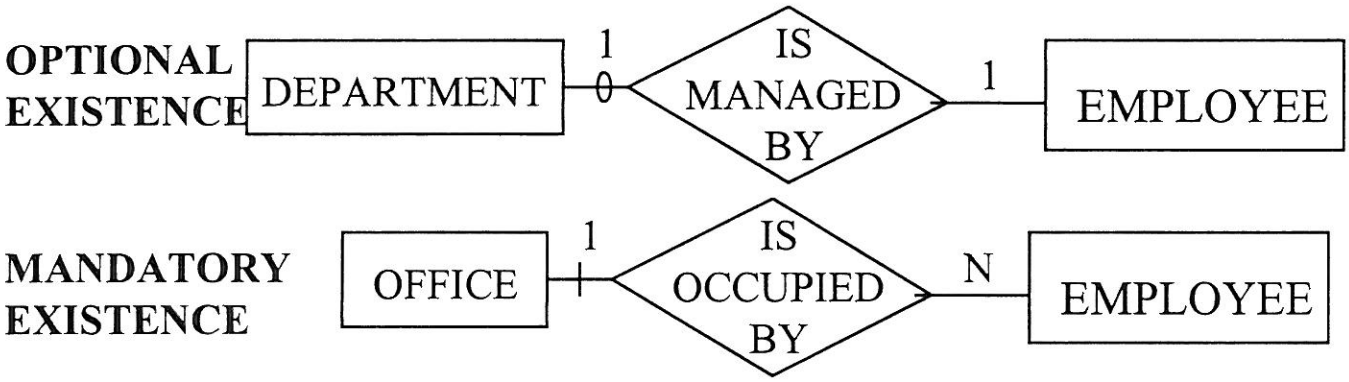


POSSIBLE MAPPINGS ARE 1:1 , M : 1 , 1 : M , M : N

•each project has only *one* manager , but an employee can manage *many* projects

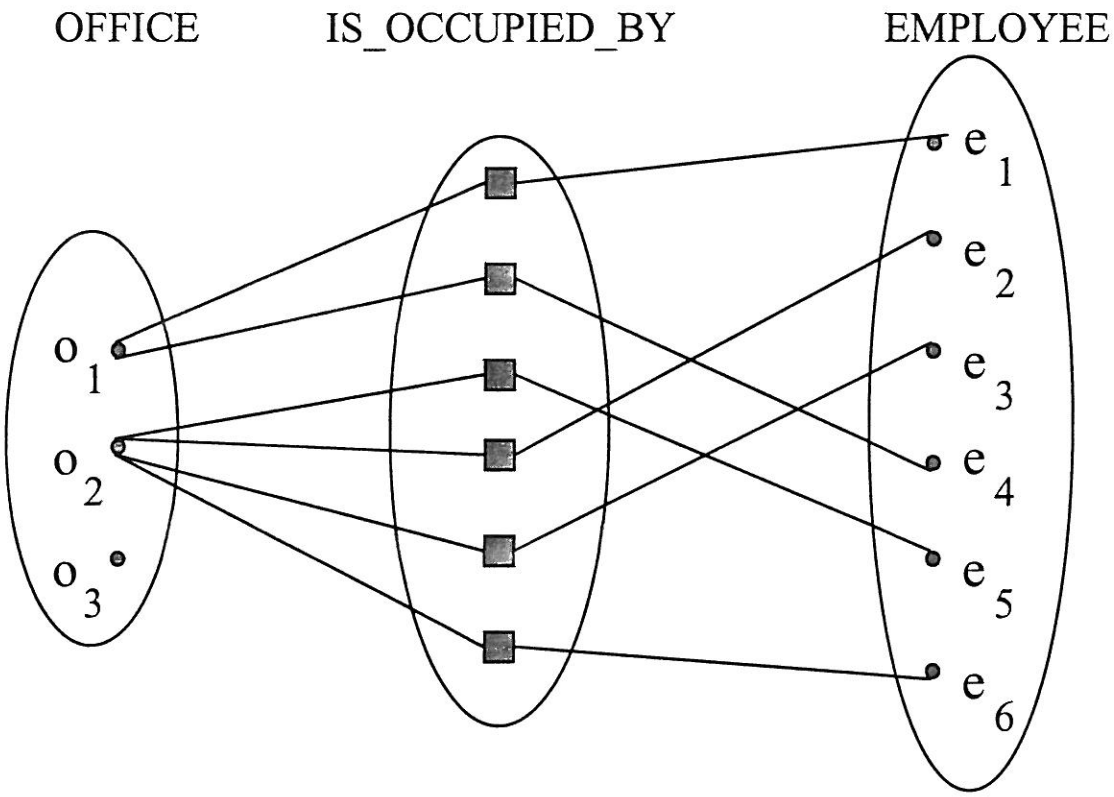
IMPORTANT: *one* and *many* refers to maximum cardinality

•CONSTRAINTS ON PARTICIPATION:



OPTIONAL EXISTENCE: defines a minimum cardinality of zero

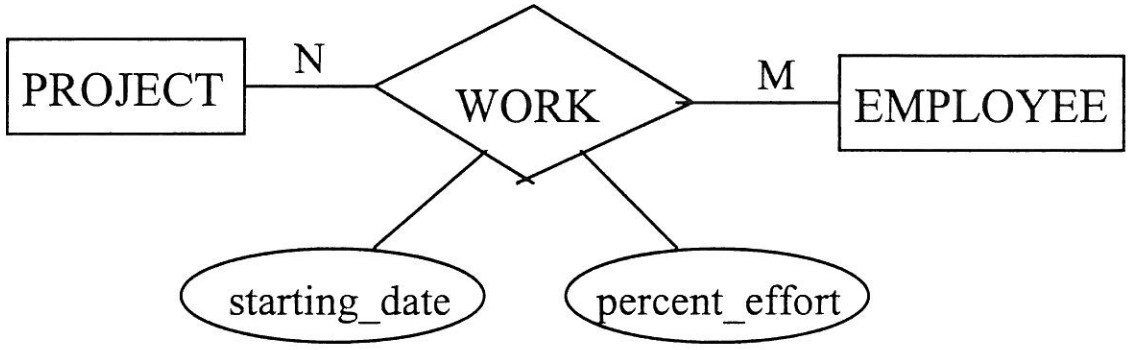
MANDATORY EXISTENCE: defines a minimum cardinality of one



OFFICE HAS *MANDATORY* EXISTENCE IN IS_OCCUPIED_BY

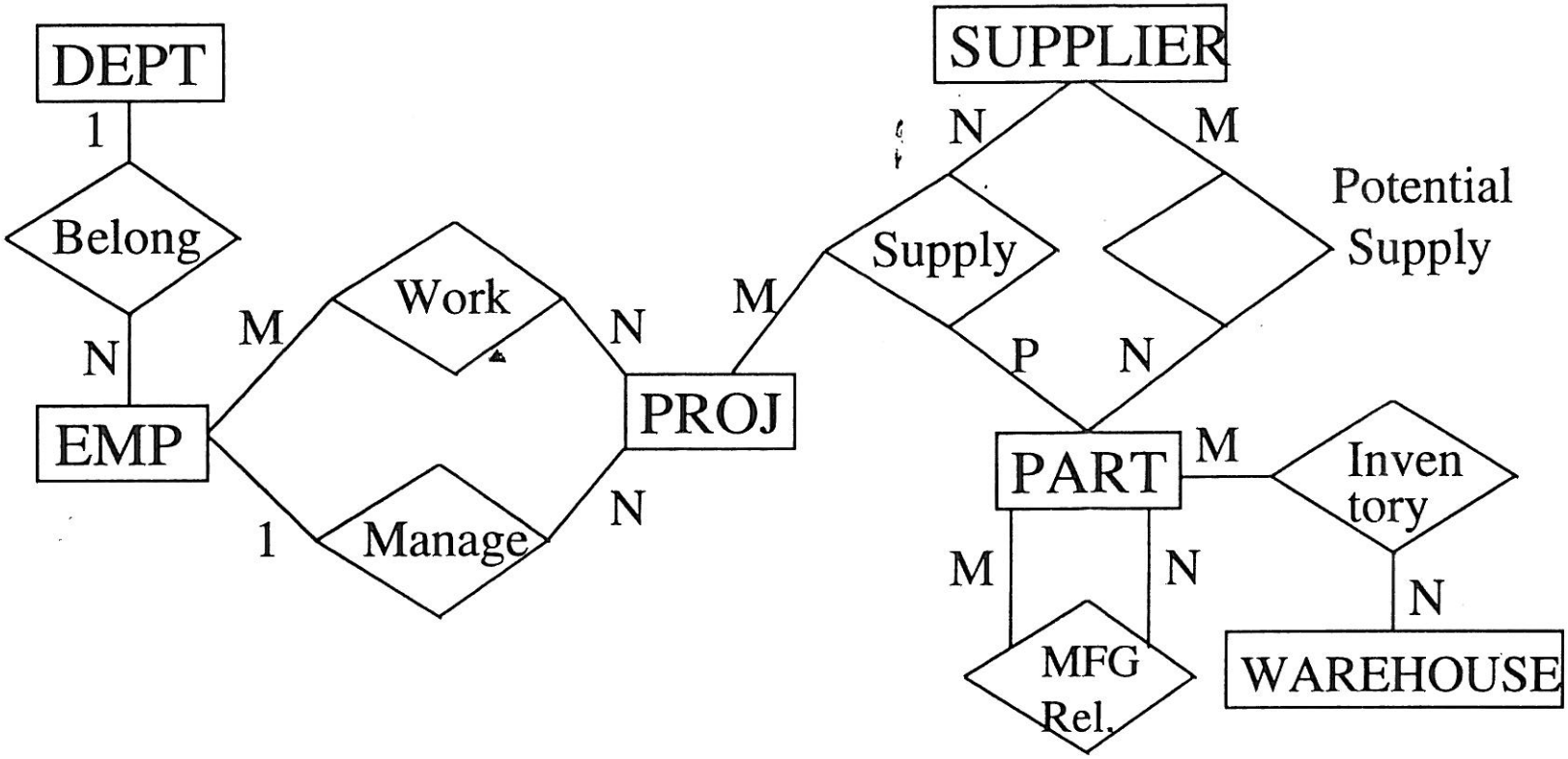
every employee is assigned exactly one office

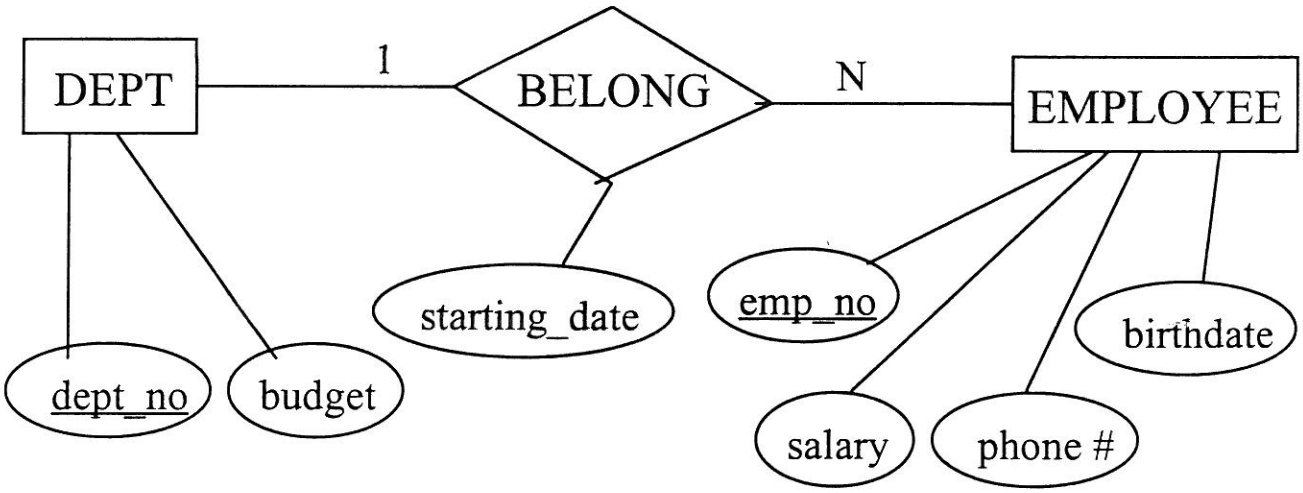
•RELATIONSHIPS HAVE ATTRIBUTES TOO



THE ENTITY - RELATIONSHIP MODEL

Example : Manufacturing Company





NOTE: attributes *birthdate* and *starting_date* are defined on the same value-set

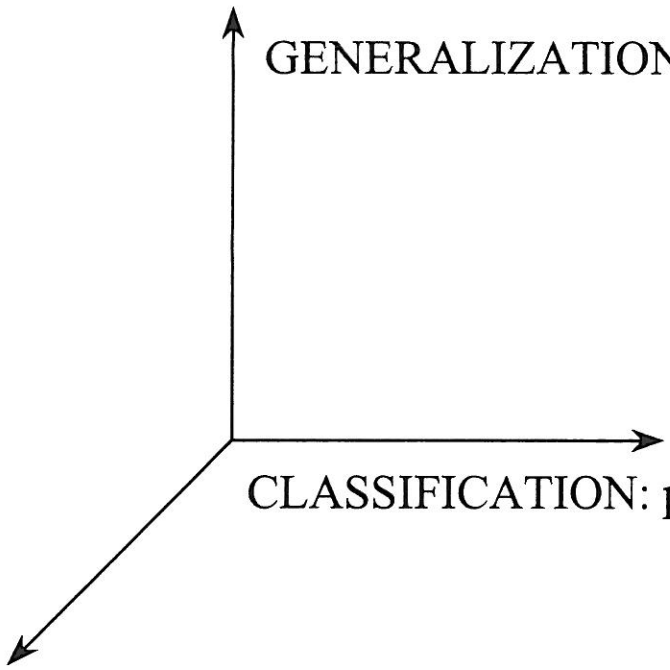
•ADVANCED CONCEPTS

•MODELING STRUCTURAL ABSTRACTIONS

(A) **CLASSIFICATION:** collection of objects with similar properties viewed as a class

(B) **AGGREGATION:** abstraction by which an object is constructed from its constituent objects

(C) **GENERALIZATION:** set of dissimilar objects is regarded as a single generic object



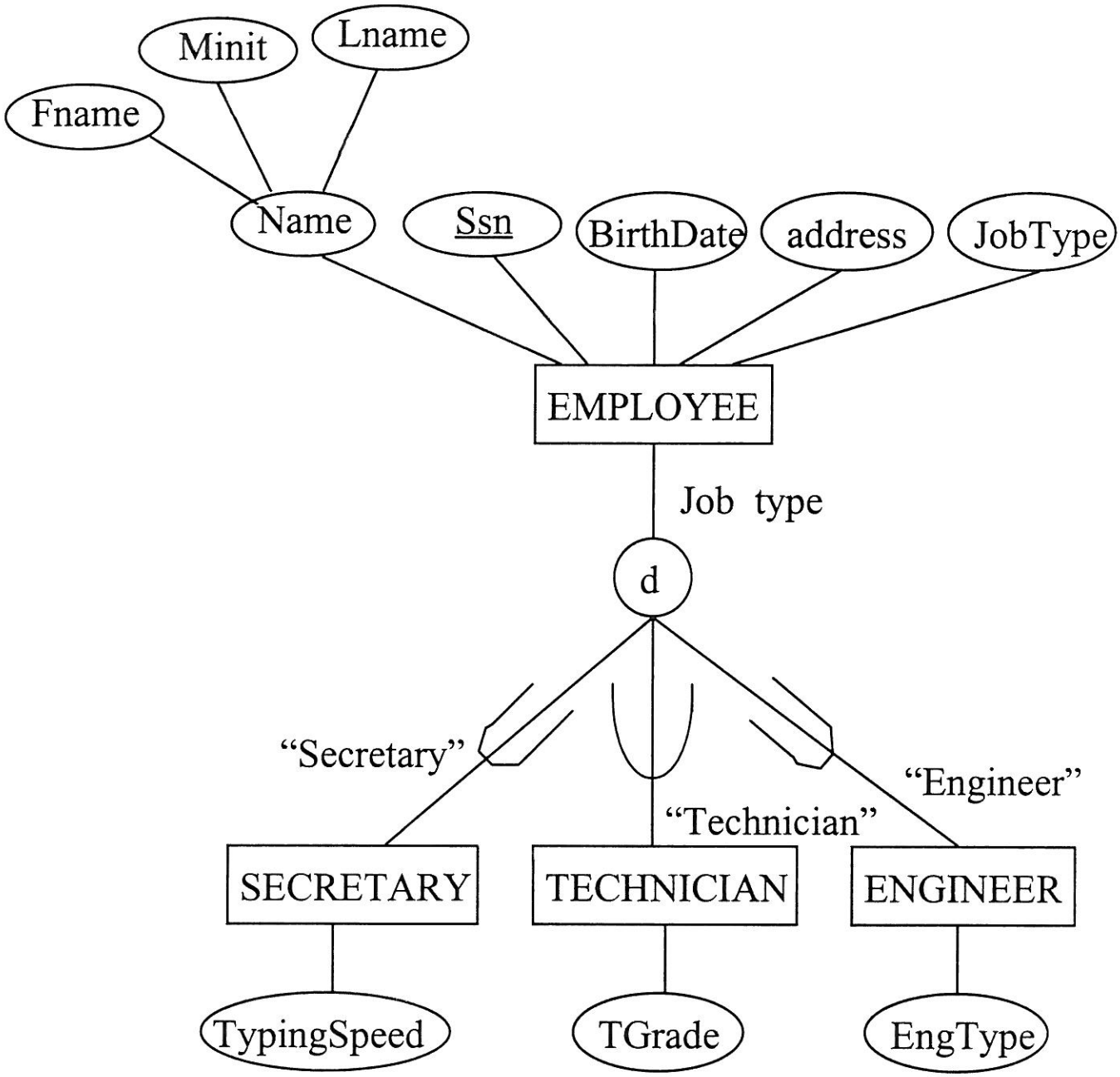
GENERALIZATION: employee is generalization of secretary, trucker, assistant

CLASSIFICATION: person : {John, Mary, …}

AGGREGATION: Person is aggregate of name, address, profession

GENERALIZATION HIERARCHY

(*class/subclass hierarchy*)



Note: subset symbol on a line indicates direction of class/subclass relationship.

employee - generalized class contains only attributes in common

secretary, engineer, trucker - specializations of employee

•ATTRIBUTE INHERITANCE

an entity that is a member of a subclass inherits all the attributes of the entity as a member of the superclass. the entity will also inherit all *relationship* instances for relationship types in which the superclass participates.

•IMPLIED CONSTRAINTS

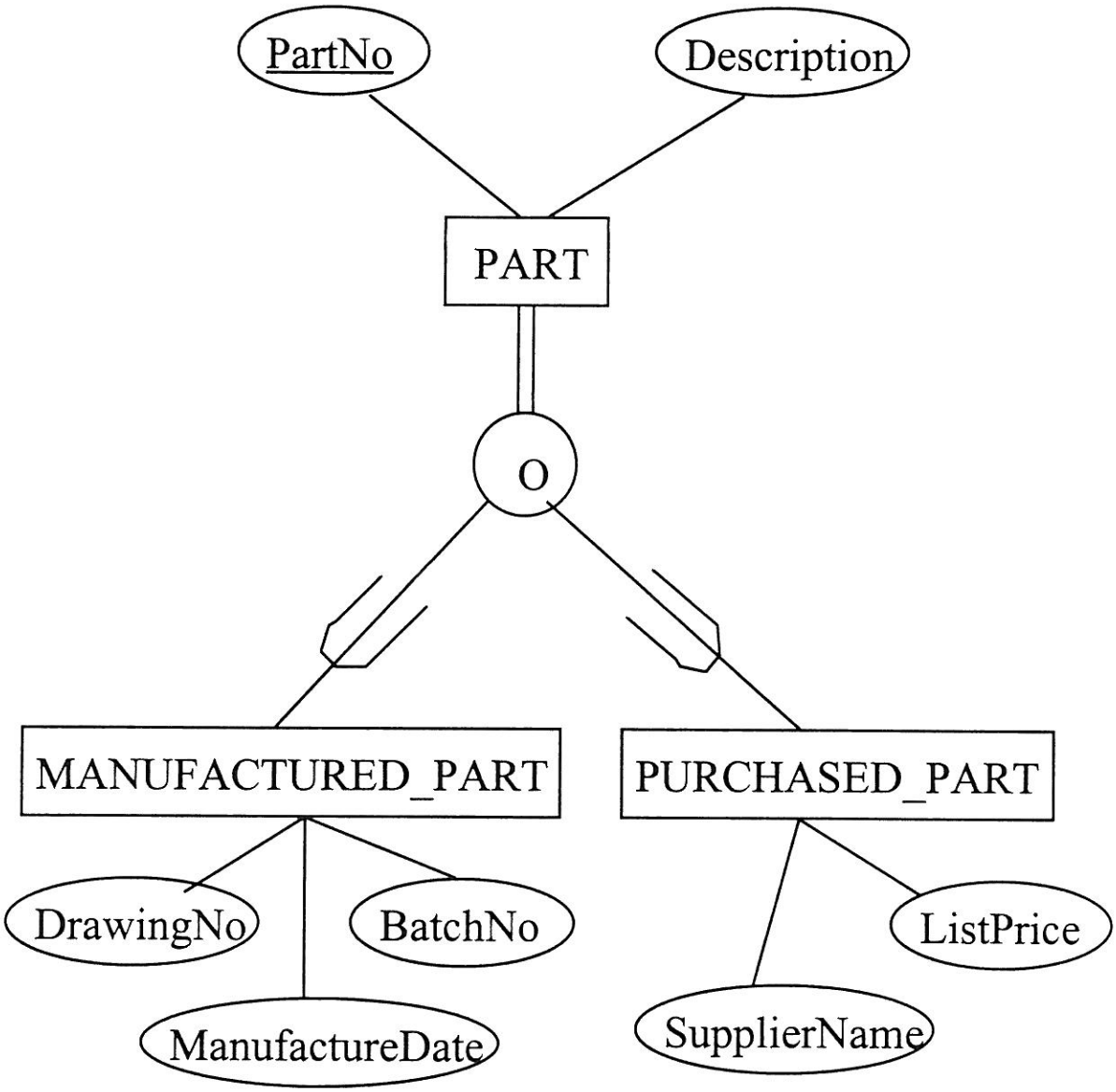
an entity cannot exist in database by being only a member of a subclass, it must also be member of superclass (reverse is not true).

ADDITIONAL CONSTRAINTS ON SPECIALIZATION

(A) DISJOINTNESS CONSTRAINT:

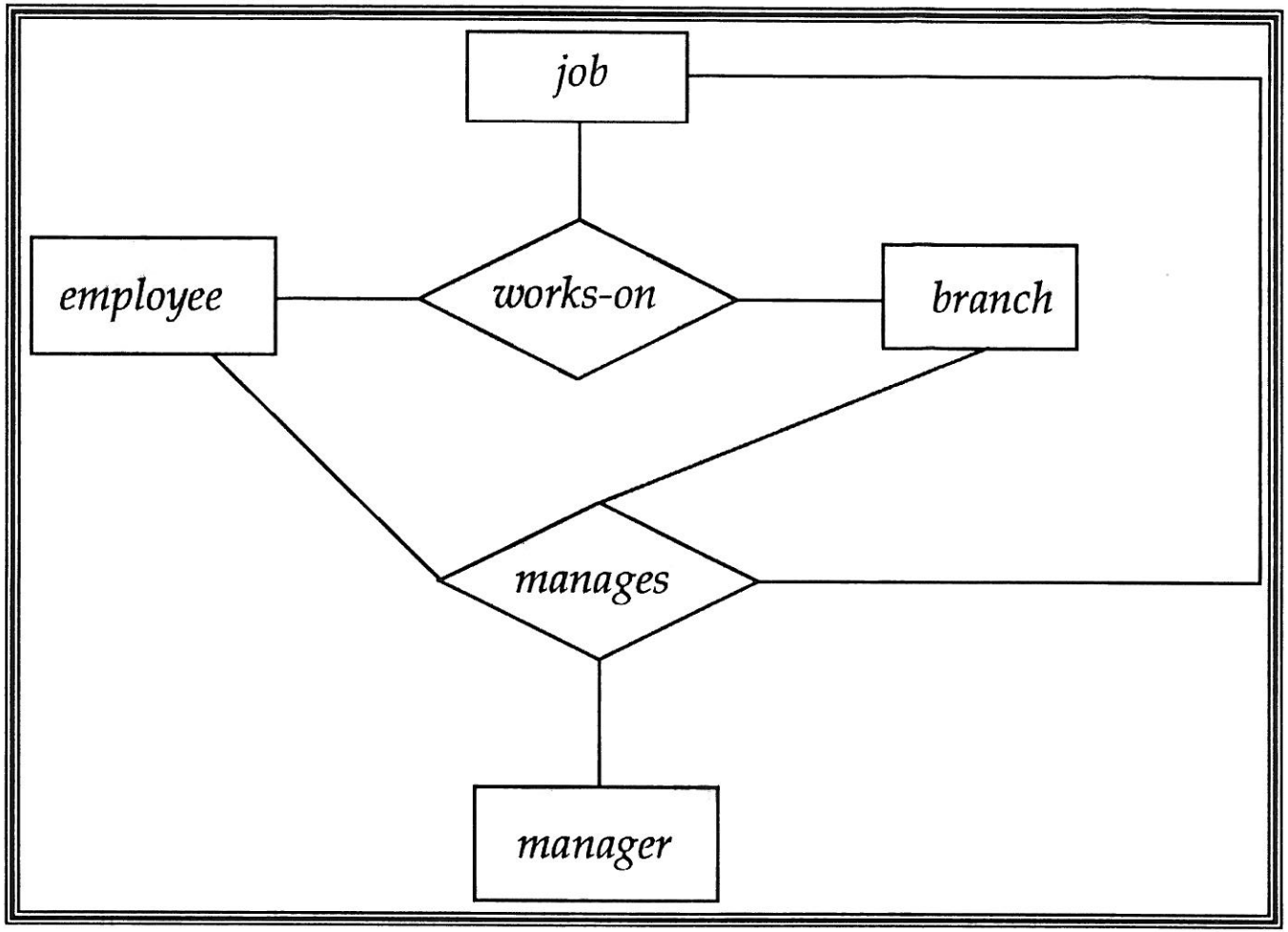
Subclasses of specialization are disjoint

(B) OVERLAPPING SUBCLASSES

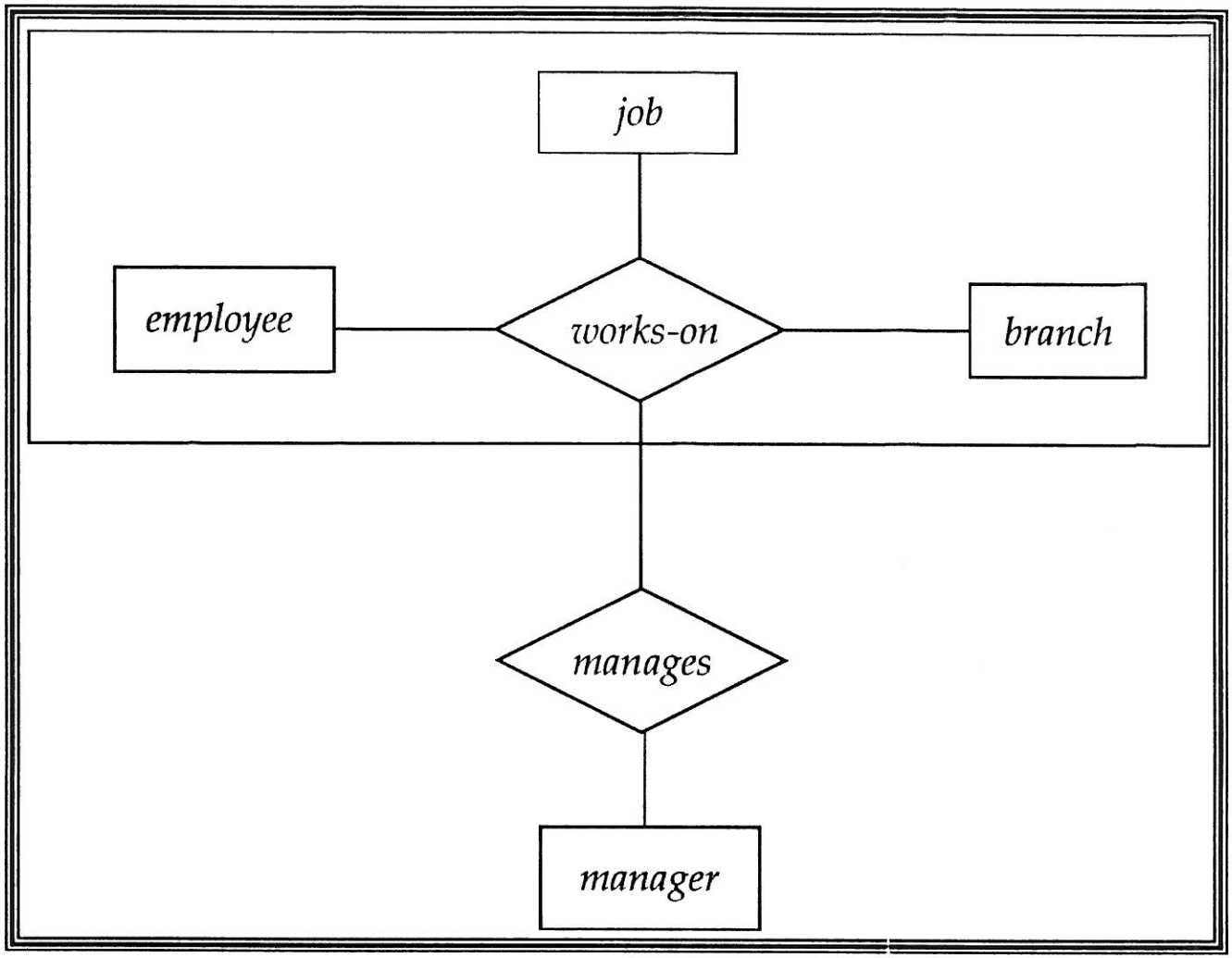


Aggregation

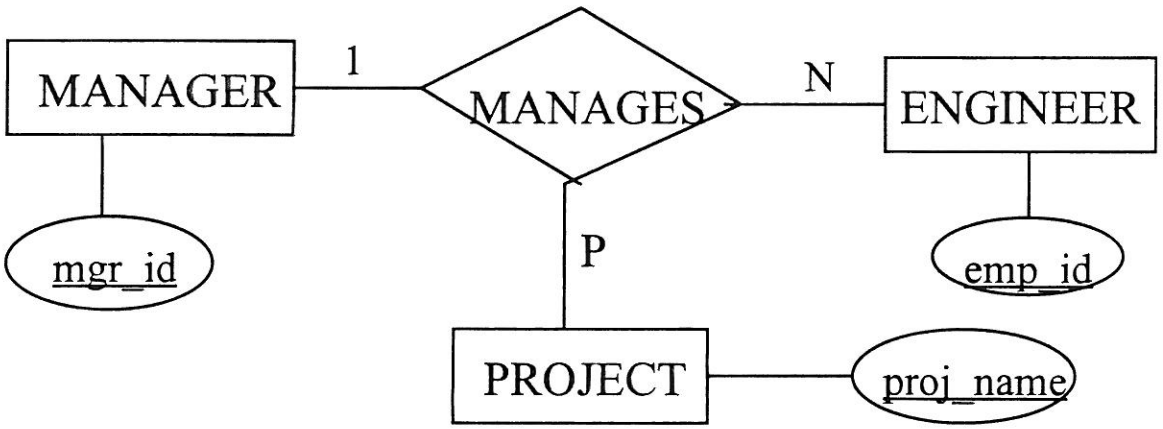
- Consider the ternary relationship *works-on*
- Suppose we want to record managers for tasks performed by an employee at a branch



E-R Diagram With Aggregation



DETERMINING THE CARDINALITY OF TERNARY RELATIONSHIPS



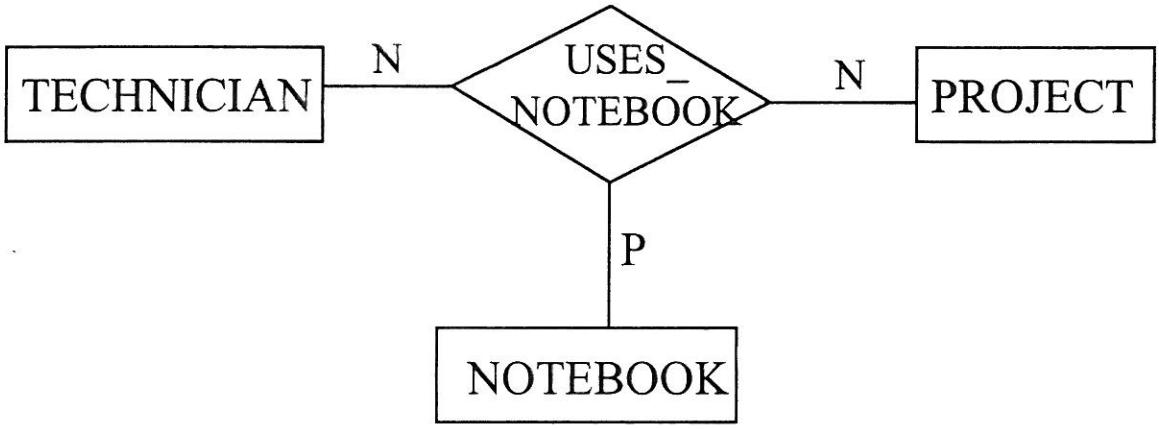
- **ASSERTION 1 :** one engineer, working under one manager could be working on many projects
- **ASSERTION 2 :** one project, under the direction of one manager, could have many engineers
- **ASSERTION 3 :** one engineer, working on one project, must have only a single manager

EQUIVALENT *FUNCTIONAL DEPENDENCY*

emp_id , proj_name → mgr_id

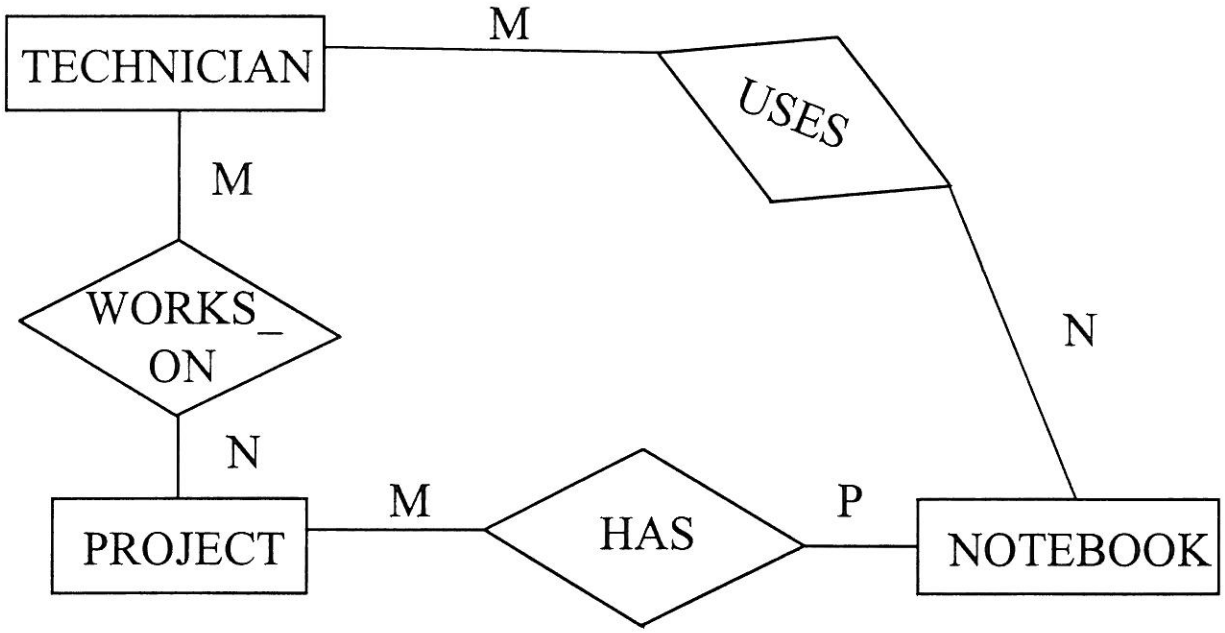
keys of associated entities

CONVERTING TERNARY RELATIONSHIPS

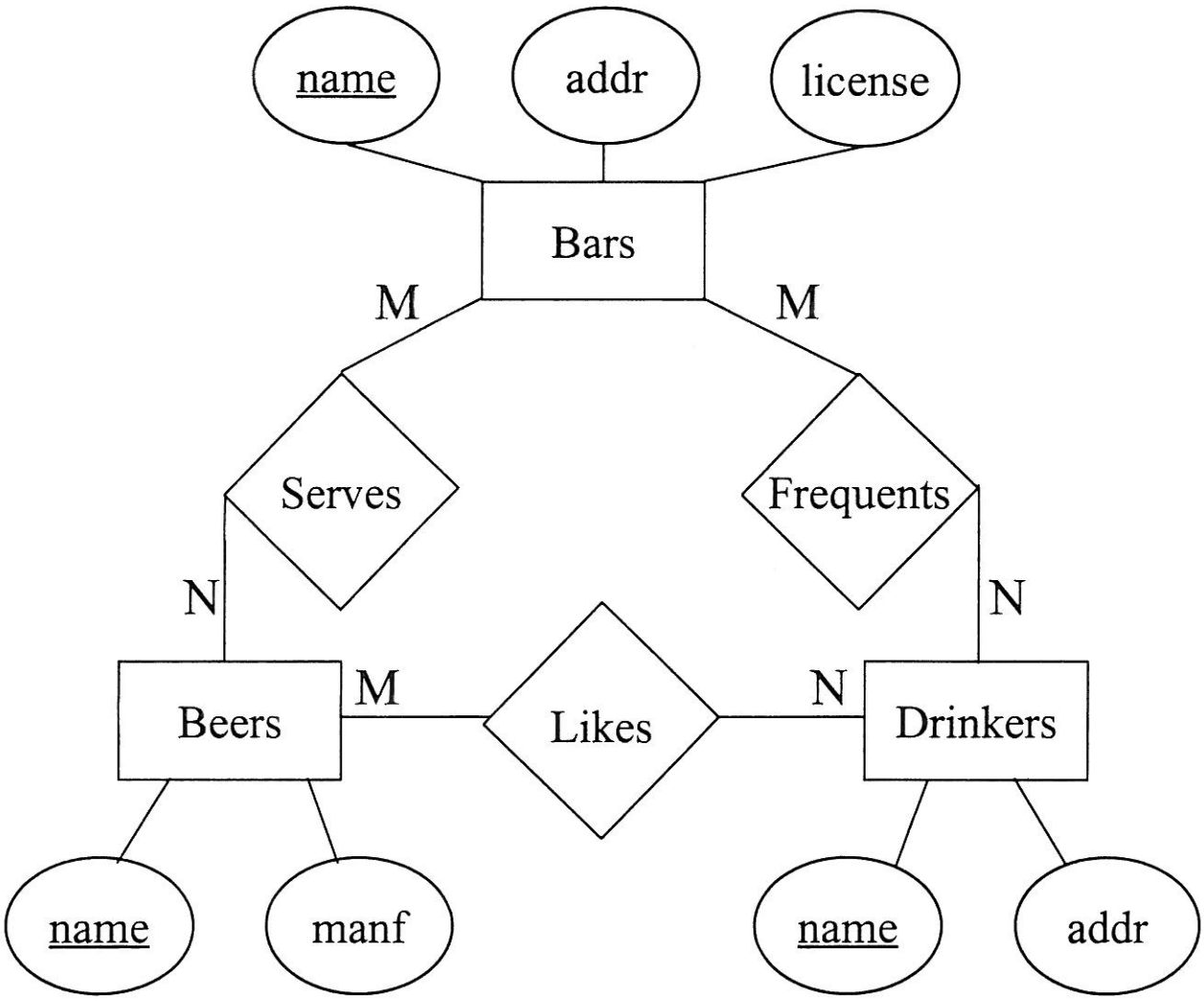


if each technician can be working on several projects and uses the same notebooks on each project;

=> can *decompose* 3-ary relationship into binary relationships

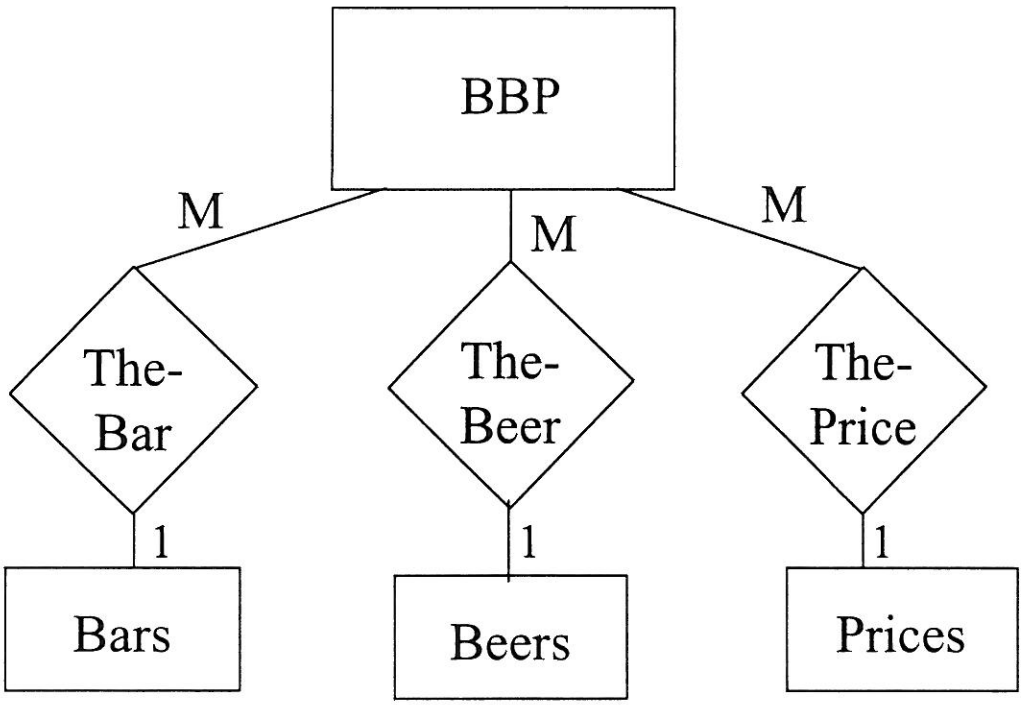


Beers-Bars-Drinkers Example



Converting Multiway to 2-Way

- Awkward in E/R, but necessary in certain "object-oriented" models.
- Create a new *connecting* E.S. to represent rows of a relationship set.
  - ➢ e.g. (Joe's Bar, Bud, $2.50) for the *Sells* relationship.
- Many-one relationships from the connecting E.S. to the others.



Weak Entity Sets

- Entity set E has only *partial key*
  Its set is formed by combining partial key with key(s) of *identifying owner(s)*

Conditions:

- Owner entity set (s) + weak entity set E must participate in 1 : M relationship set

- Identifying owner has *mandatory* existence in identifying relationship set

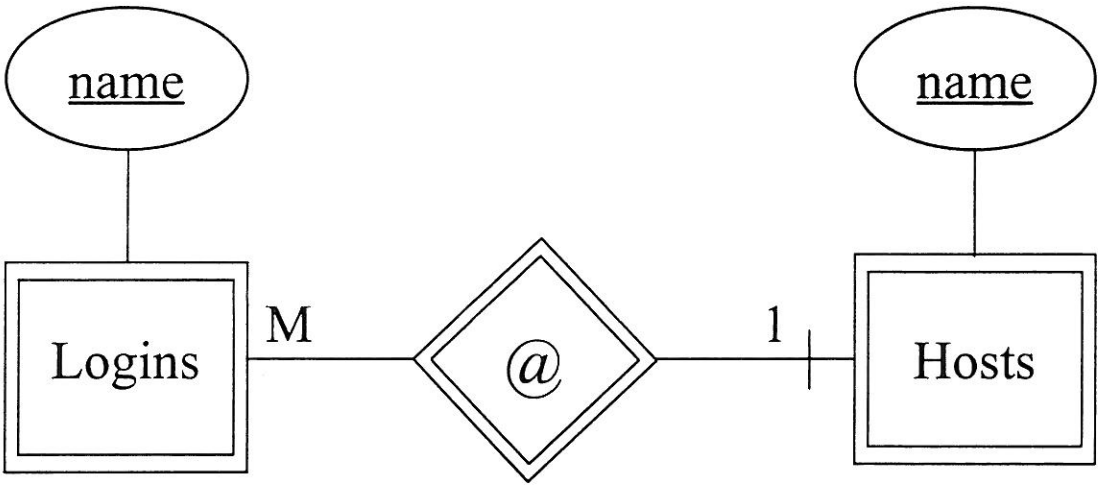
NOTE: If identifying owner is itself weak, then the key of E is formed recursively

Example: Logins (Email Addresses)

Login name = user name + host name, e.g.

- A "login" entity corresponds to a user name on a particular host, but the password table doesn't record the host, just the user name, e,g, . peters.

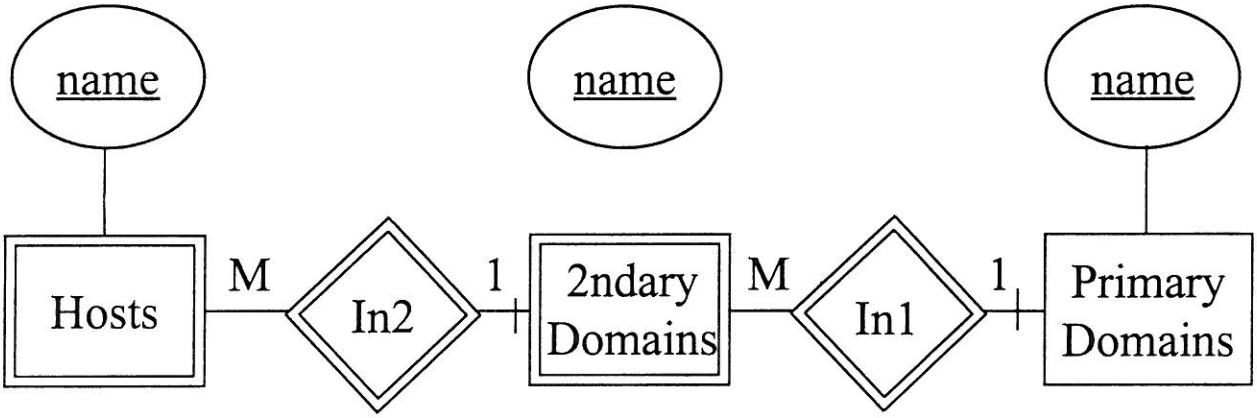
- Key for a login = the user name at the host (which is unique for that host only) + the IP address of the host (which is unique globally) .



- Design Issue: Under what circumstances could we simply make login-name and host- name be attributes of logins, and dispense with the weak E.S.?

Example: Chain of "Weakness"

Consider IP addresses consisting of a primary domain (e.g. edu) subdomain (e.g. northwestern) and host (e.g. begonia).



- Key for primary domain = its name.
- Key for secondary domain = its name + name of primary domain
- Key for host = its name + key of secondary domain = its name + name of secondary domain + name of primary domain

STEPS OF LOGICAL DESIGN WITH E-R MODEL

(1) CLASSIFY ENTITIES AND ATTRIBUTES
- entities should contain descriptive information
- classify multivalued attributes as entities

(2) IDENTIFY GENERALIZATION HIERARCHIES

(3) DEFINE RELATIONSHIPS
check for redundant relationships

(4) PERFORM *VIEW INTEGRATION* AND CONSTRUCT ONE CONCEPTUAL SCHEMA

(5) TRANSFORM E-R MODEL INTO SQL TABLES

(6) NORMALIZE THE TABLES

THE RELATIONAL MODEL

RELATION => TABLE => SET

- no two tuples are identical
- ordering of tuples is immaterial

RELATION SCHEMA

DEPT (DNAME, THIS-YEAR-BUDGET, LAST-YEAR-BUDGET)

D.Name	This-year-budget	Last-year-budget
ENG.	200 K	100 K
SALES	800 K	600 K
MANUF	500 K	300 K

relation instance

tuple

EMP (EMP-NO, SALARY, OFF-PHONE, HOME-PHONE, BIRTHDATE)

Emp-no	Salary	Off.Phone	Home Phone	Birthdate
101	60 K	1-6181	328-0152	1952
117	45 K	1-3298	675-2191	1965
250	90 K	1-5276	321-0712	1943
390	35 K	1-9913	275-3172	1961

BELONG (EMP-NO, DEPT-NAME, START-DATE)

Emp-no	Dept-name	Start-date
101	ENG	1985
117	SALES	1990
250	ENG	1970
390	MANUF	1992

•**ALL ATTRIBUTE VALUES ARE ATOMIC**

SUPPLY (SUPPLIER-NO, PART-NO)

Supplier-no	Part-no
S2	P1
S2	P2
S4	P2
S4	P4
S4	P5

correct !

Supplier-no	Part-no
S2	{P1,P2}
S4	{P2,P4,P5}

values are sets (repeating groups)

wrong !

DEFINITION: a relation is in *first normal form* (1nf) if it contains only atomic values, that is there are no repeating groups within a tuple

•**RELATION SCHEMA** $R(A_1, A_2, \ldots, A_n)$

•**ATTRIBUTE** A_i : role played by domain D_i

•**DEGREE** of relation is the number of attributes (n) in its relation schema

•KEYS

- **•SUPERKEY:** a set of one or more attributes which taken collectively allows us to uniquely identify a tuple in a relation

- **•KEY:** a *minimal superkey* that is a superkey from which we cannot remove any attributes and still have the uniqueness hold

Ex : *EMP'*(EMP-NO, EMP-NAME, SALARY, BIRTH-DATE)

has a number of *Candidate Keys*

- **•PRIMARY KEY:** candidate key chosen by the system (user) and used to identify tuples in the relation (usually *index* also constructed on *primary key*)

•**FOREIGN KEY** (FK) : attribute *FK* (possibly composite) of relation R_2 is a foreign key if and only if it satisfies :

1. Each value of *FK* is wholly null or wholly non-null

2. There exists a relation R_1 with primary key *FK* such that each non-null value of *FK* is identical to the value of *FK* in some tuple of R_1.

Notes :

•relations R_1 and R_2 need not be distinct

•R_2 is called the *referencing relation*

•R_1 is called the *referenced relation*

Ex :

In relation belong {emp_no, dept_name} are two FKs.

What makes a system relational (Codd)

=> Two main integrity rules:

A. **Entity integrity rule**: no component of the primary key of a relation is allowed to accept NULLs.

(remember NULL means: "value unknown" or "does not apply")

Reason: if two or more tuples had nulls for primary keys, we are not able to distinguish between them

B. **Referential integrity rule**: the database must not contain any unmatched foreign key values.

foreign key & referential integrity rule are defined in terms of one another.

Important: these rules do not say how the system should enforce them!

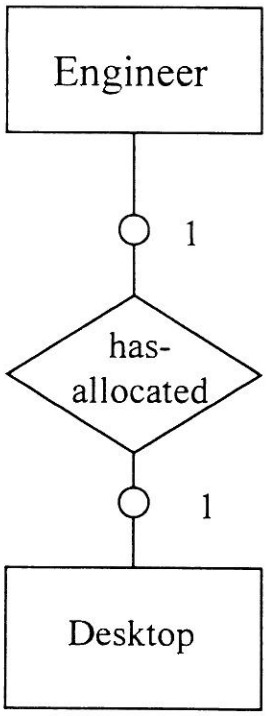
Transformation rules: From ER model to SQL tables

1. An entity set becomes all table with the same attributes

2. Binary relationship set rules:

2.1 One-to-one

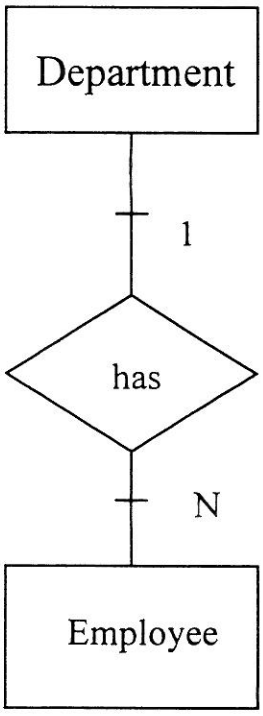
relationship set **embedded** in either entity table and the primary key of one entity table appears as **foreign key** in the other.




```
create table engineer
        (emp_id char(10),
        desktop_no integer,
        primary key (emp_id),
        foreign key (desktop_no)  references
            desktop
        on delete set null on update cascade);

create table desktop
        (desktop_no integer,
        emp_id char(10),
        primary key (desktop_no),
        foreign key (emp_id) references
            engineer on delete set null
        on update cascade);
```

2.2 **One-to-many**

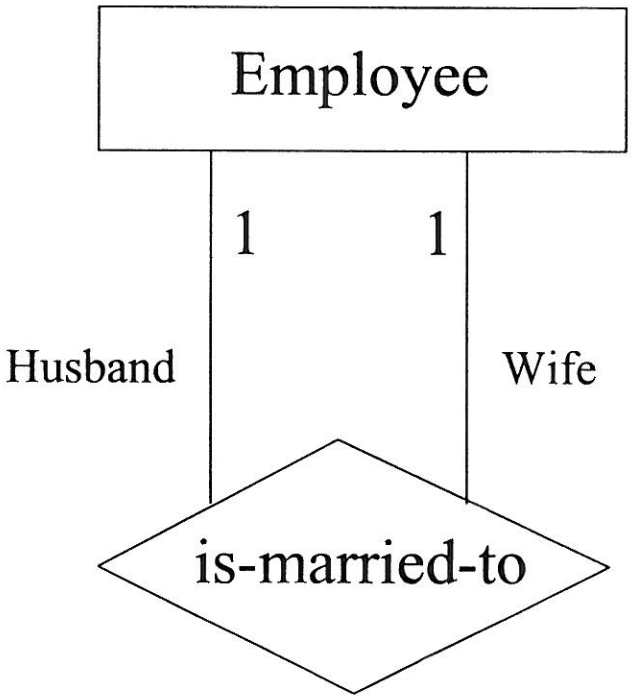
relationship set **embedded** in entity table corresponding to "**child**" and primary key of parent table appears as foreign key of child table



create table **department**
(dept_no integer,
dept_name char(20),
primary key (dept_no));

create table **employee**
(emp_id char(10),
emp_name char(20),
dept_no integer not null,
primary key (emp_id),
foreign key (dpt_no) reference **department**
on delete set default on update cascade);

Recursive Binary Relationship

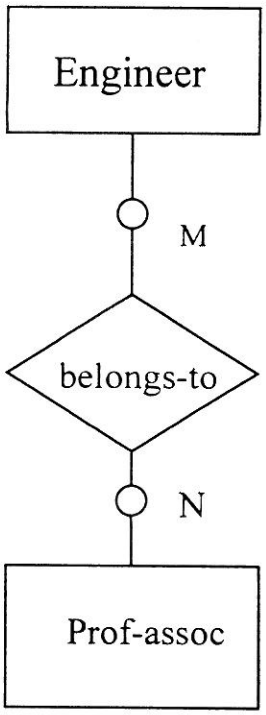


Create table **employee**
(emp_id char(10),
emp_name char(20),
spouse_id char(10),
primary key (emp_id),
foreign key (spouse_id) references **employee** on delete set null on update cascade);

What happened to Roles?

2.3 **Many-to-many**

relationship set represented as a **separate** table and the primary key on this table consists of the primary keys of the related entity tables

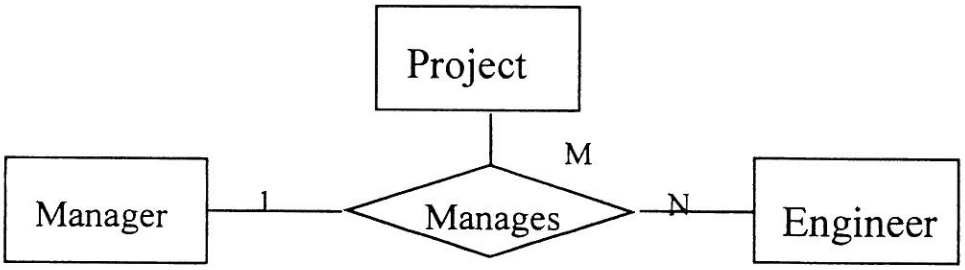



```
create table engineer
        (emp_id char(10),
        primary key (emp_id));

create table prof_assoc
        (assoc_name varchar(256),
        primary key (assoc_name));

create table belongs_to
        (emp_id char(10),
        assoc_name varchar(256),
        primary key (emp_id, assoc_name),
        foreign key (emp_id) references engineer
            on delete cascade on update cascade,
        foreign key (assoc_name) references
            prof_assoc
            on delete cascade on update cascade);
```

3. Ternary and N-ary relationships rule:

In all cases create a **separate** table containing the primary keys of all entities

However the primary key of this table may consist of less than n attributes



create table **project**
(project_name char(20),
primary key (project_name));
create table **manager**
(mgr_id char(10),
primary key (mgr_id));
create table **engineer**
(emp_id char(10),
primary key (emp_id));
create table **manages**
(project_name char(20),
mgr_id char(10) not null,
emp_id char(10),
primary key (project_name, emp_id),
foreign key (project_name) references **project**
on delete cascade on updte casecade,
foreign key (mgr_id) references **manager**
on delete cascade on update cascade,
foreign key (emp_id) references **engineer**
on delete cascade on update cascade);

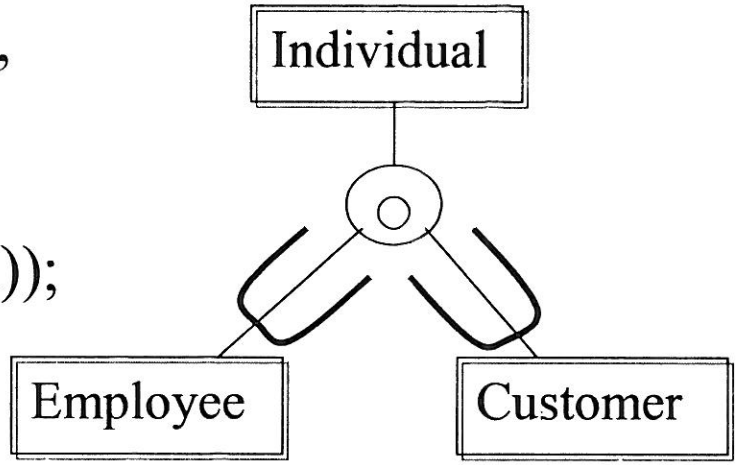
4. Generalization rule

- create a separate table for the generic type and each of its subtype

- The supertype table contains the supertype key and all common attributes

- Each subtype table contains the supertype key [as primary and foreign key] and only the attributes that are specific to that subtype

Generalization Abstraction Transformation Rules




```
create table individual (indiv_id char(10),
                         indiv_name char(20),
                         indiv_addr char(20),
                         primary key (indiv_id));
create table employee (emp_id char(10),
                       job_title char(15),
                       primary key (emp_id),
                       foreign key (emp_id) references individual
                          on delete cascade on update cascade);
create table customer (cust_no char(10),
                       cust_credit char(12),
                       primary key (cust_no),
                       foreign key (cust_no) references individual
                          on delete cascade on update cascade);
```

Create table

- Basic data definition statement in **SQL'92**

Constraints on attributes

- Not null
- Unique for candidate keys
- Primary key => implies not null & unique
- Foreign key

How to enforce the foreign key rule ?

- two options :

(A) Reject any operations that result in **illegal state**

or

(B) System performs additional **compensating actions** to guarantee that overall result is a **legal state**

Default Policy: Rejecting Violating Modifications

1) Try to insert a new **employee** tuple whose *dept_no* is not known or is NULL.

2) Try to update an **employee** tuple to change the *dept_no* to a non-NULL value that is not a legal value in **department** table.

3) Try to delete a **department** tuple and its *dept_no* appears as a component of an **employee** tuple(s).

4) Try to update a **department** tuple in such a way that we change the *dept_no* and the old *dept_no* appears as a component of an **employee** tuple(s).

ALL ARE REJECTED

SQL '92 solution

- On update

  Cascade - The update operation on the PK in the referenced table cascades to all tuples in the referencing table with matching FKs.

  Set-Null - Foreign keys are set to NULL when they match the old PK of the updated tuple in the referenced table.

  Set-Default - Foreign keys are set to default when they match the old PK …
  [Note: broader interpretation of NULL]

- On Delete

  Cascade - The delete operation on the referenced table cascades to all tuples in the referencing table with matching FKs

  => One user operation may trigger many other operations

  SET-NULL ...

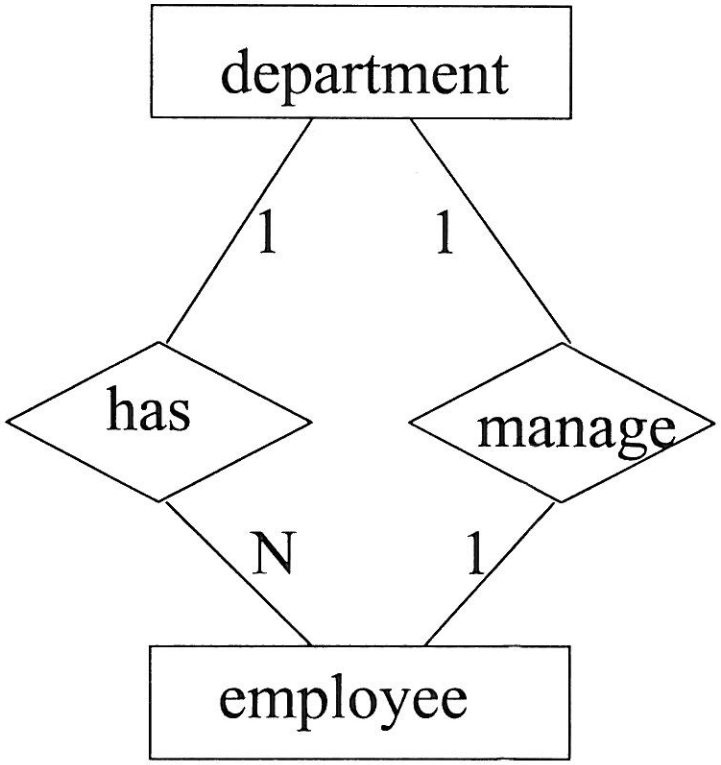
  Set-Default ...

Previous Solution Handles only Modifications of Types 3 and 4

1) *insert into* ***employee***
   values ('chair01', 'abejones'); ⇨ System cannot use NULL for foreign key

2) *update* ***employee***
   set dept_no = 14
   where emp_id = 'bigboss'; ⇨ If dept_no = 14 is a new value not in **department** table action is rejected

Embed Two One-to-many Relationship




```
create table department
  (dept_no integer,
  dept_name char (20),
  emp_id char (10),
  primary key (dept_no),
  foreign key (emp_id) references employee
     on delete set default on update cascade);
```

Question: Can we insert tuples
with new *dept_no* or new *emp_id*?