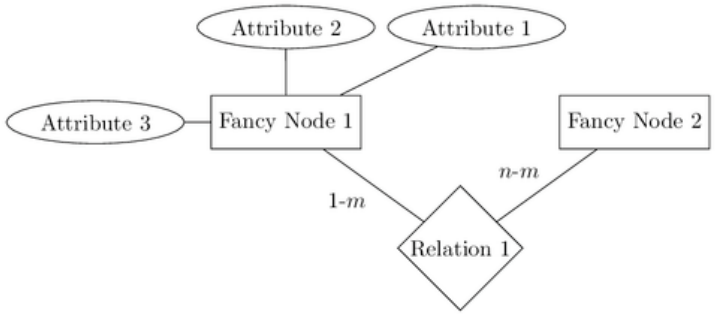
UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Information Storage and Management I

Dr. Alejandro Arbelaez

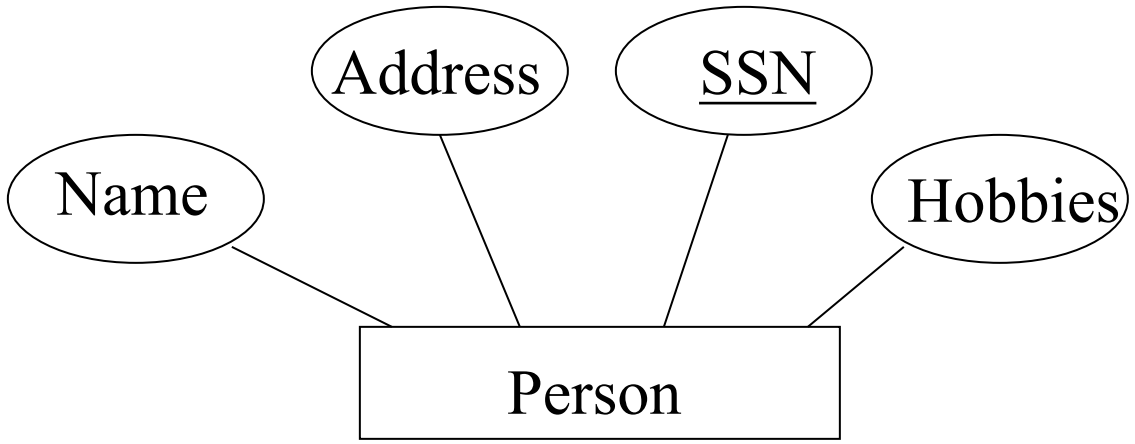


Entity Relationship Models

Person Entity (Example)

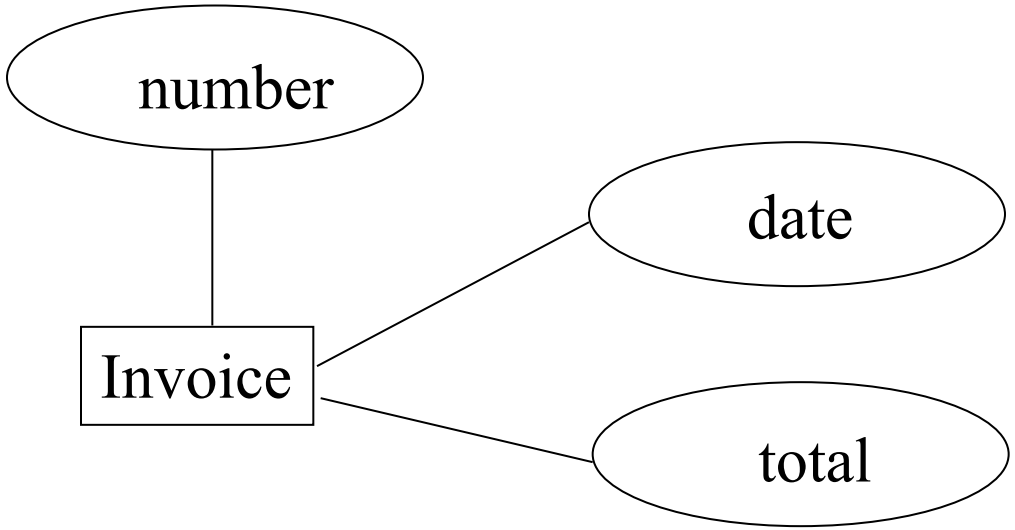
We illustrate attributes using ovals.

- Graphical Representation in ER diagram:



We use a rectangular symbol to represent an entity set in the Entity Relationship Diagram.

Invoice Entity (Example)

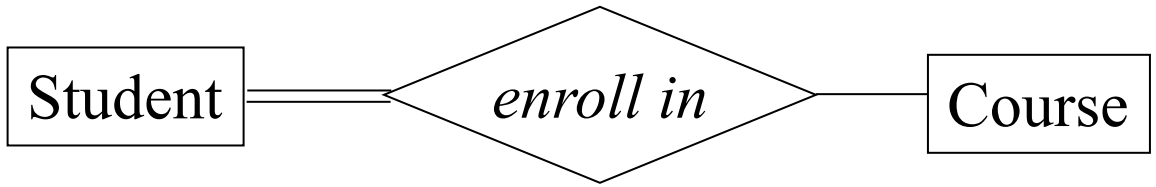


An Invoice entity may be described by attributes including:

- invoice number
- invoice date
- invoice total

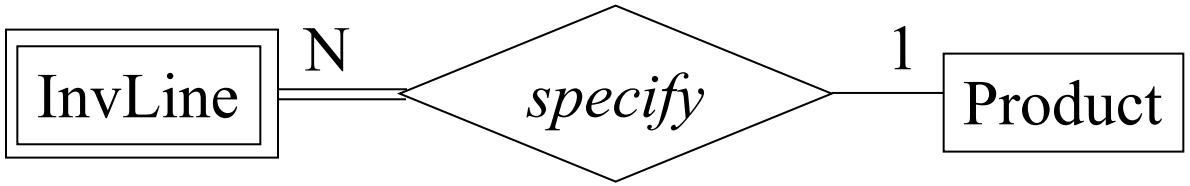
Relationships

- **The single relationship line** means that ‘enroll in’ is optional to a course
- A course can exist and have zero students enrolled
- **The double line** means that ‘enroll in’ is mandatory for student
- Each Student must be enrolled in at least one course.



Relationships

Cardinality is a constraint on a relationship specifying the number of entity instances that a specific entity may be related to via the relationship.



An Invoice Line will specify exactly one Product. A Product may appear on any number, zero or more, Invoice Lines.

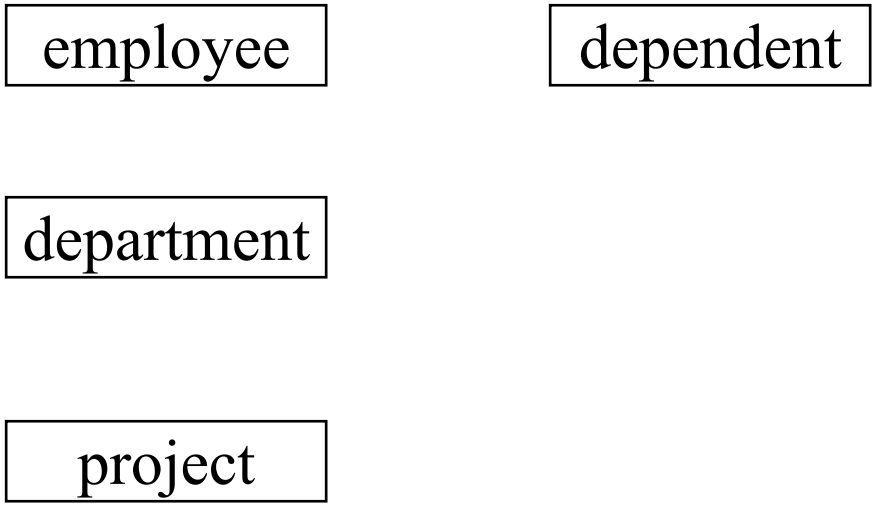
Example – Company DB

- The company database keeps track of a company's employees, departments, and projects
- **Requirements -- Concerning the department**
  1. Company is organized into departments
  2. A department has a unique name, a unique number, and a given employee would be the manager
  3. We track the start date for the manager function
  4. A department may be in several locations
  5. A department controls a number of **projects**
- **Concerning the project:**
  1. A project has a unique name, a unique number, and is in a single location

Example

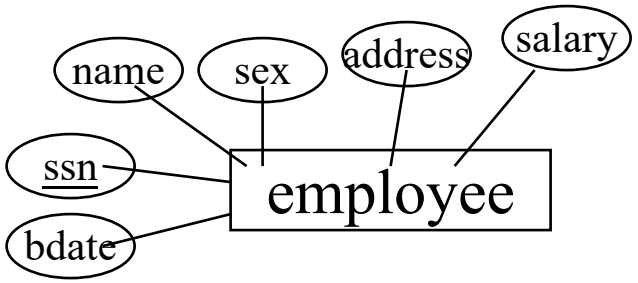
- **Concerning the employee:**
  1. Each employee has a name, social insurance number, address, salary, sex, and birth date
  2. An employee is assigned to one department but may work on several projects which are not necessarily controlled by the same department
  3. We track of the direct supervisor of each employee
  4. We track the dependents of each employee (for insurance purposes)
- **Concerning the dependent:**
  1. We record each dependent's first name, sex, birth date, and relationship to the employee

Entities



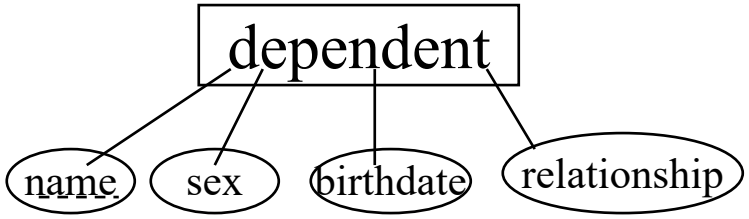
Example

- **Concerning the employee:**
  1. Each employee has a name, social insurance number, address, salary, sex, and birth date
  2. An employee is assigned to one department but may work on several projects which are not necessarily controlled by the same department
  3. We track of the direct supervisor of each employee
  4. We track the dependents of each employee (for insurance purposes)



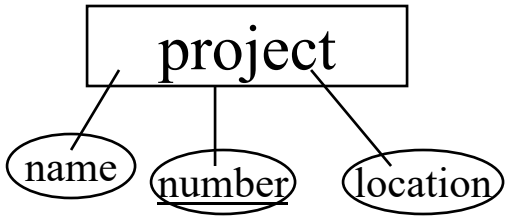
Example

- **Concerning the dependent:**
    1. We record each dependent's first name, sex, birth date, and relationship to the employee



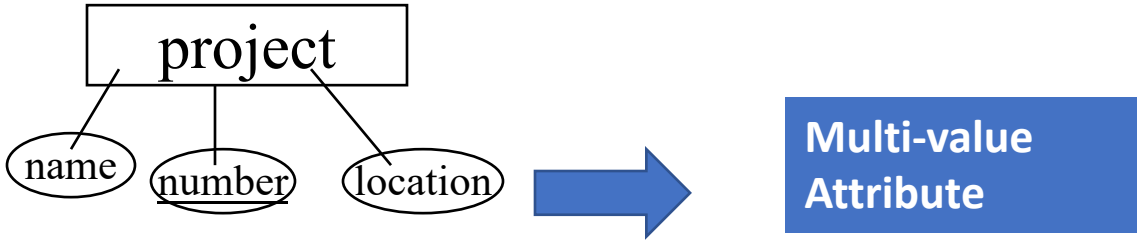
Example

- **Concerning the project:**
  1. A project has a unique name, a unique number, and is in a single location

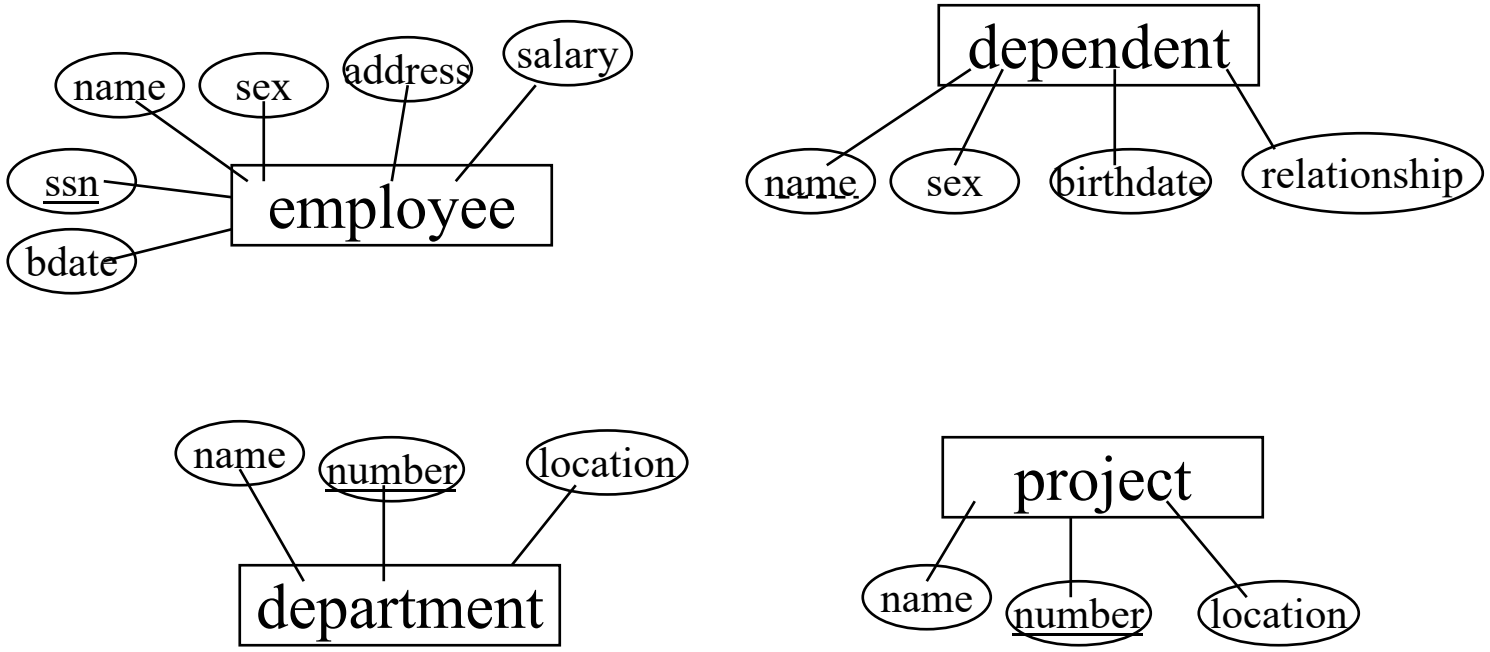


Example

- **Requirements -- Concerning the department**
  1. Company is organized into departments
  2. A department has a unique name, a unique number, and a given employee would be the manager
  3. We track the start date for the manager function
  4. A department may be in several locations
  5. A department controls a number of **projects**

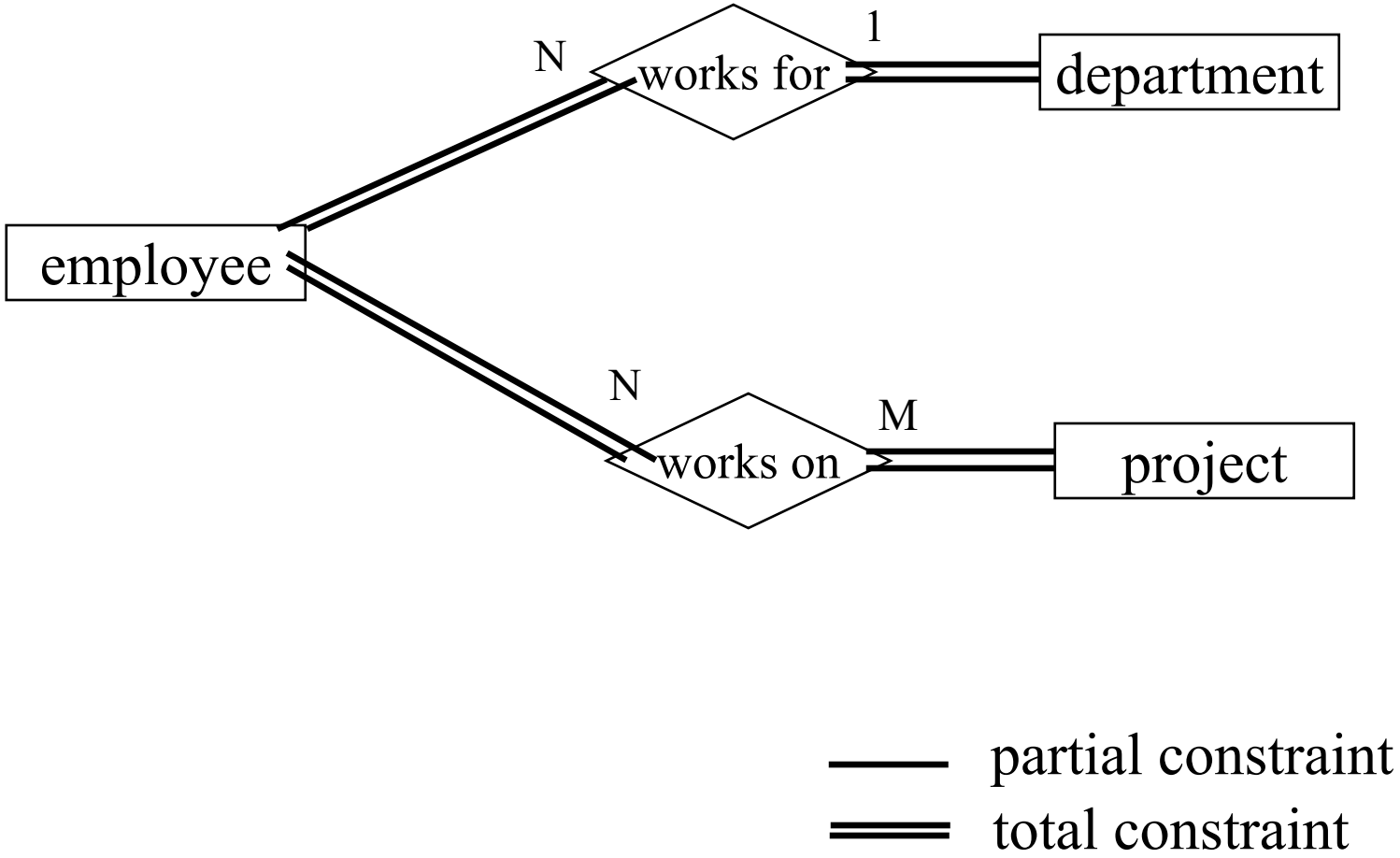


Entities


name
sex
address
salary
ssn
employee
bdate
dependent
name
sex
birthdate
relationship
name
number
location
department
project
name
number
location

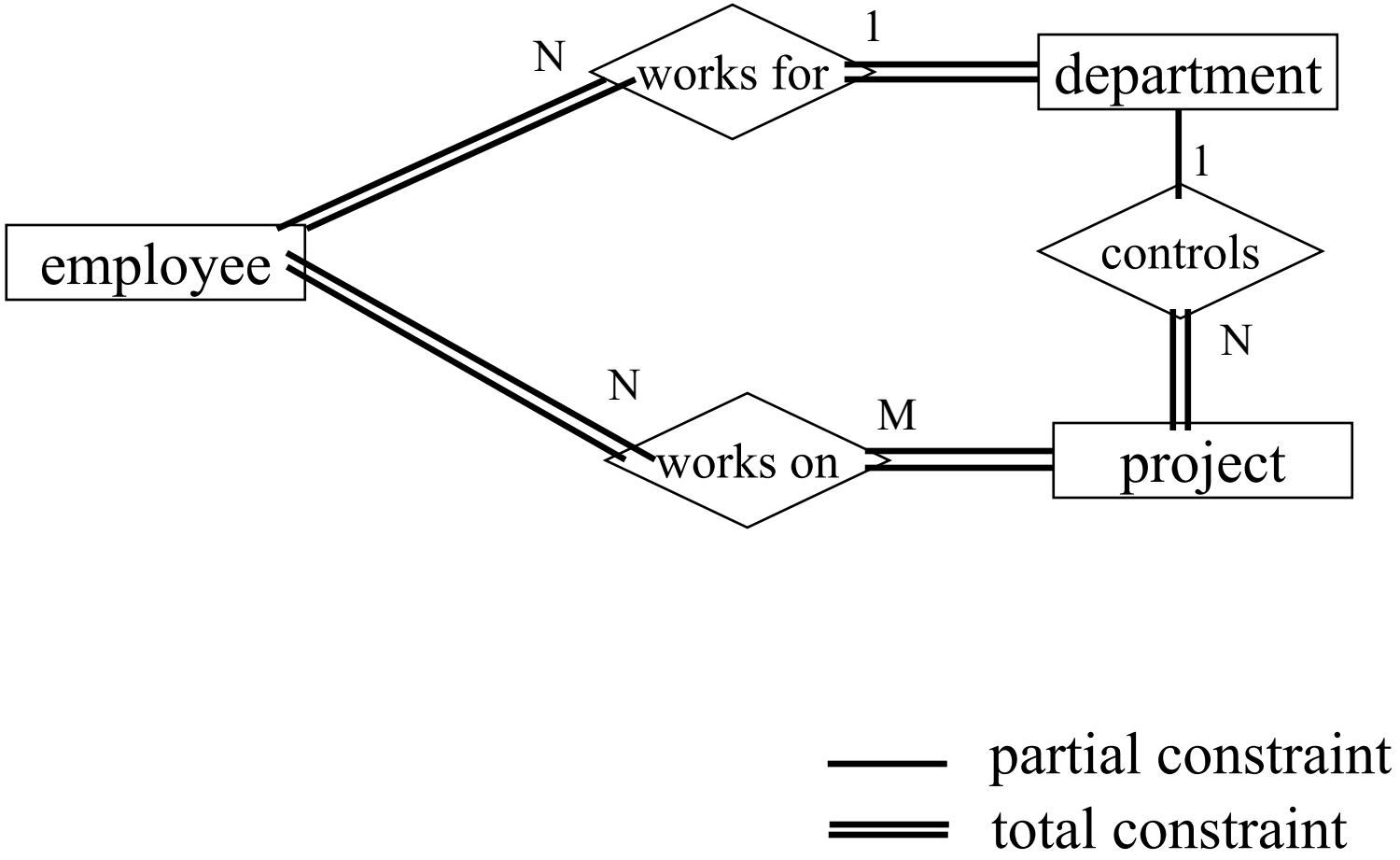
Relationships

An employee is assigned to one department but may work on several projects which are not necessarily controlled by the same department



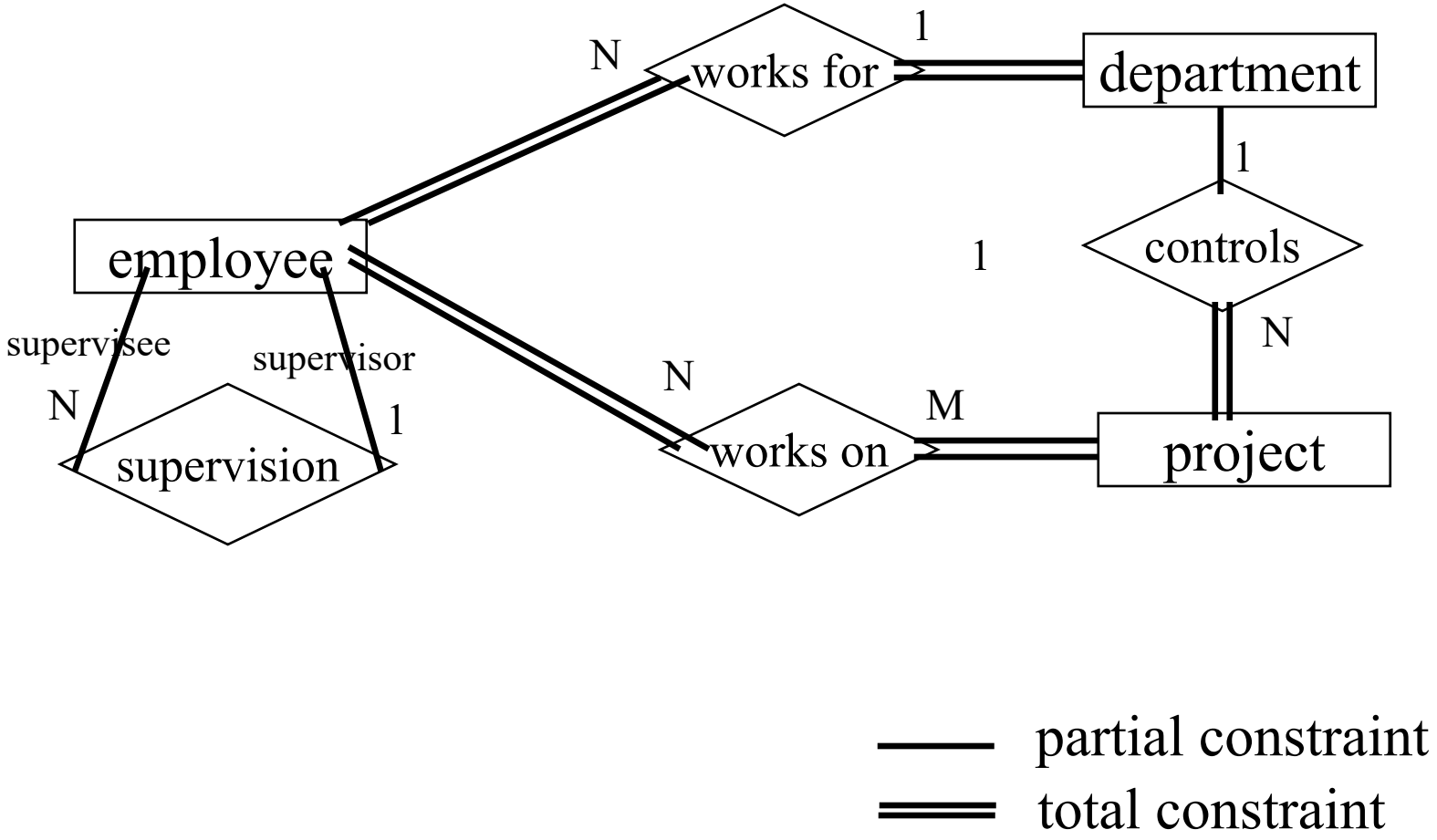
A department controls a number of projects

Relationships



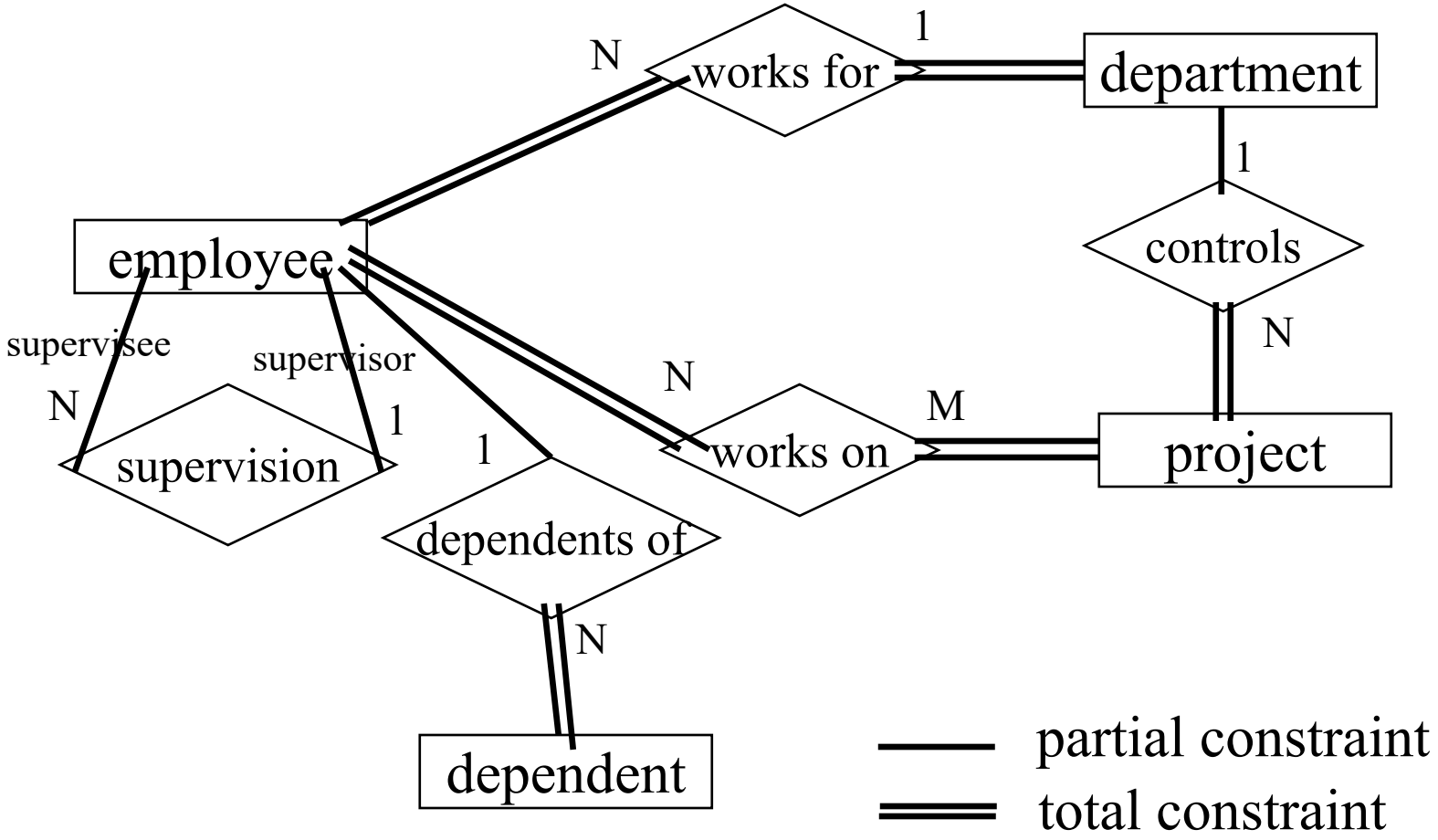
We track of the direct supervisor of each employee

Relationships



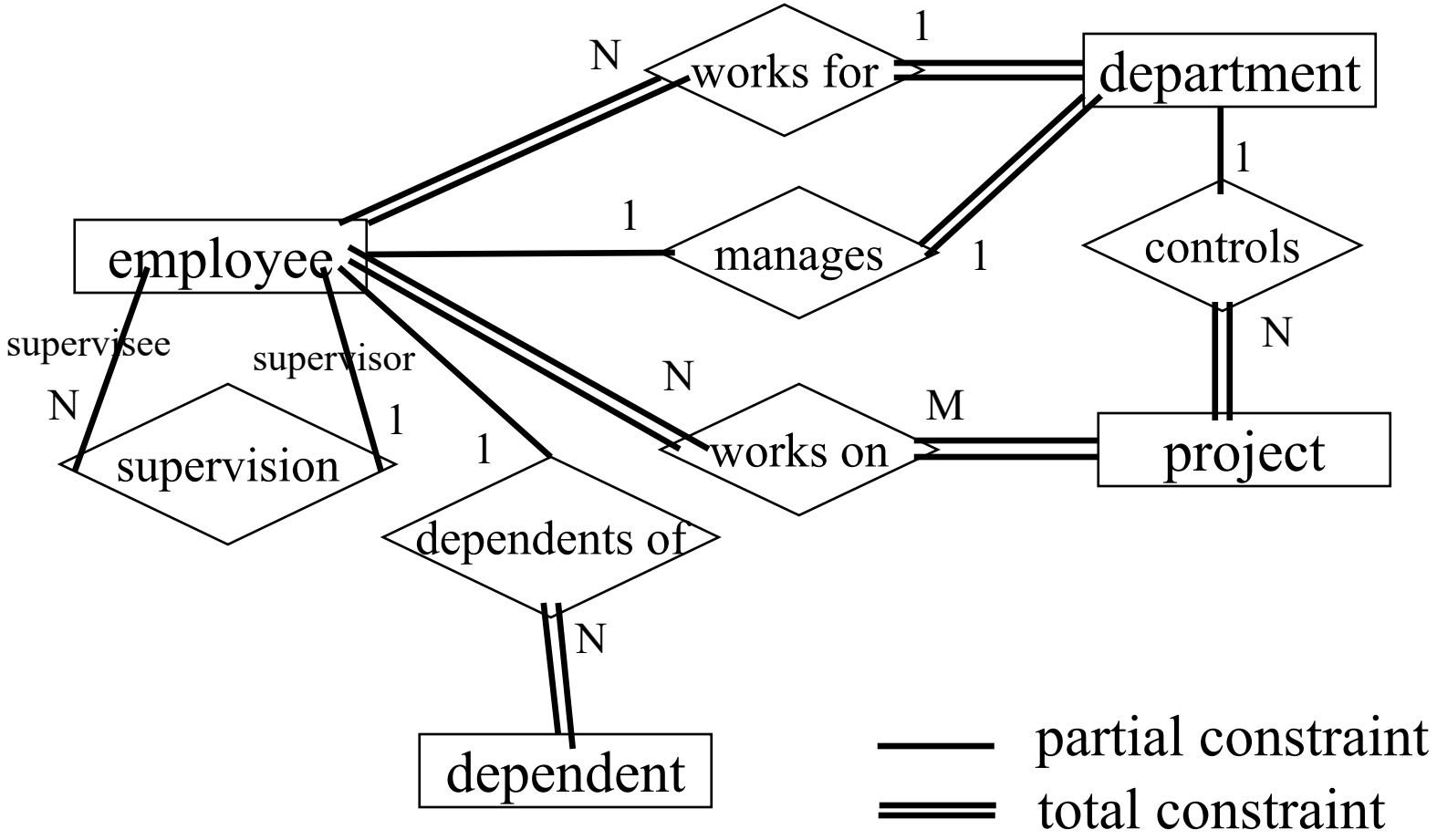
We track the dependents of each employee (for insurance purposes)

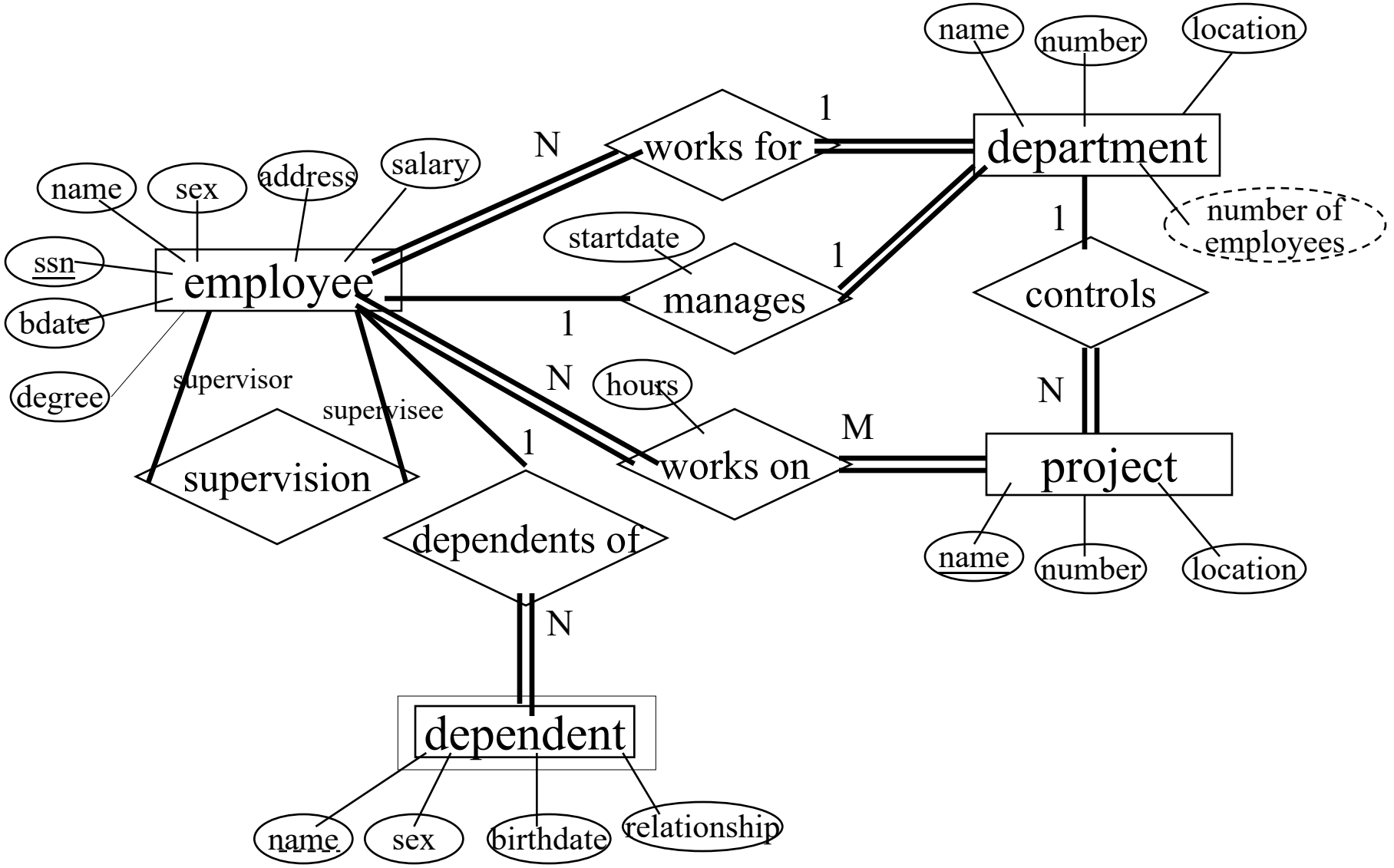
Relationships

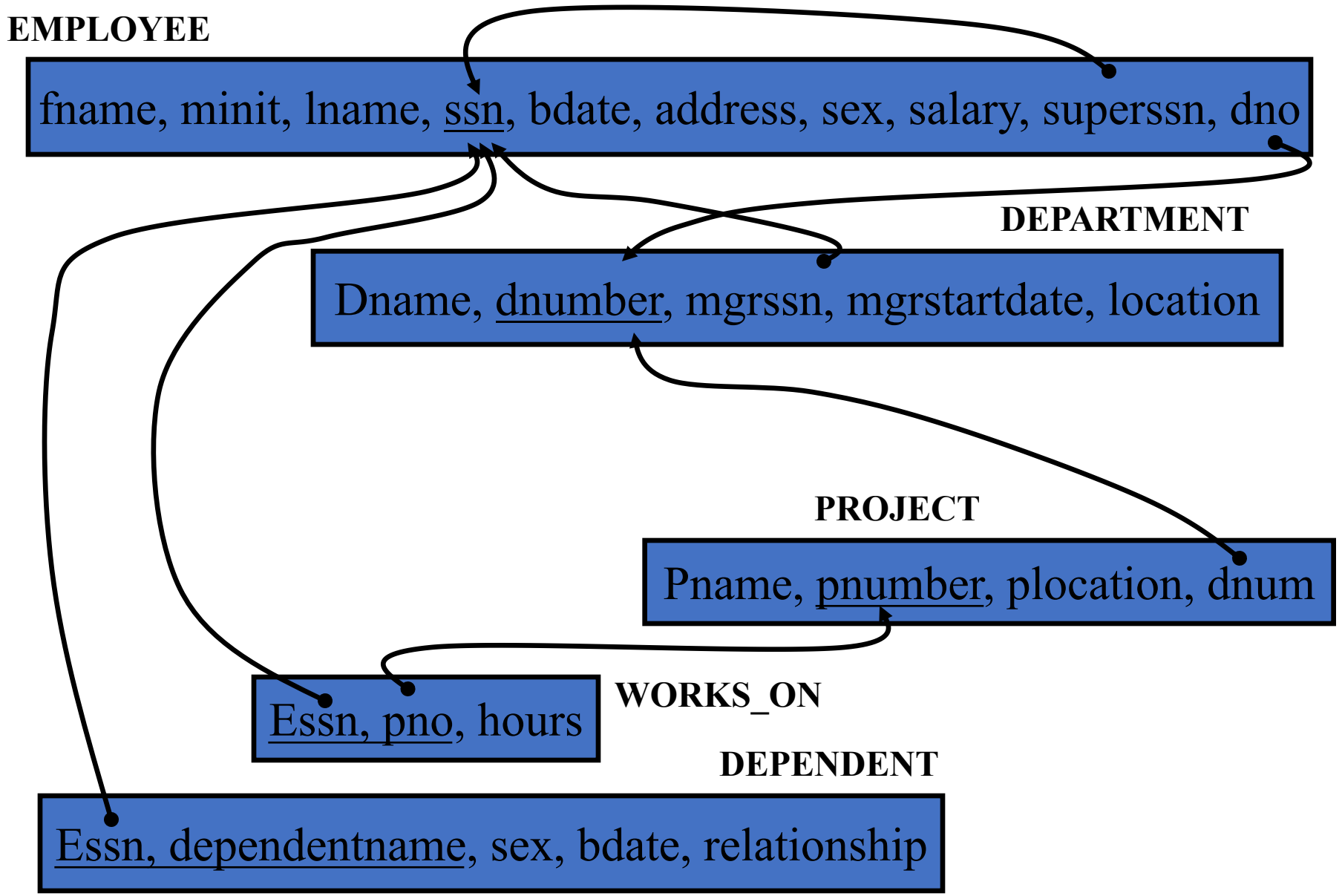


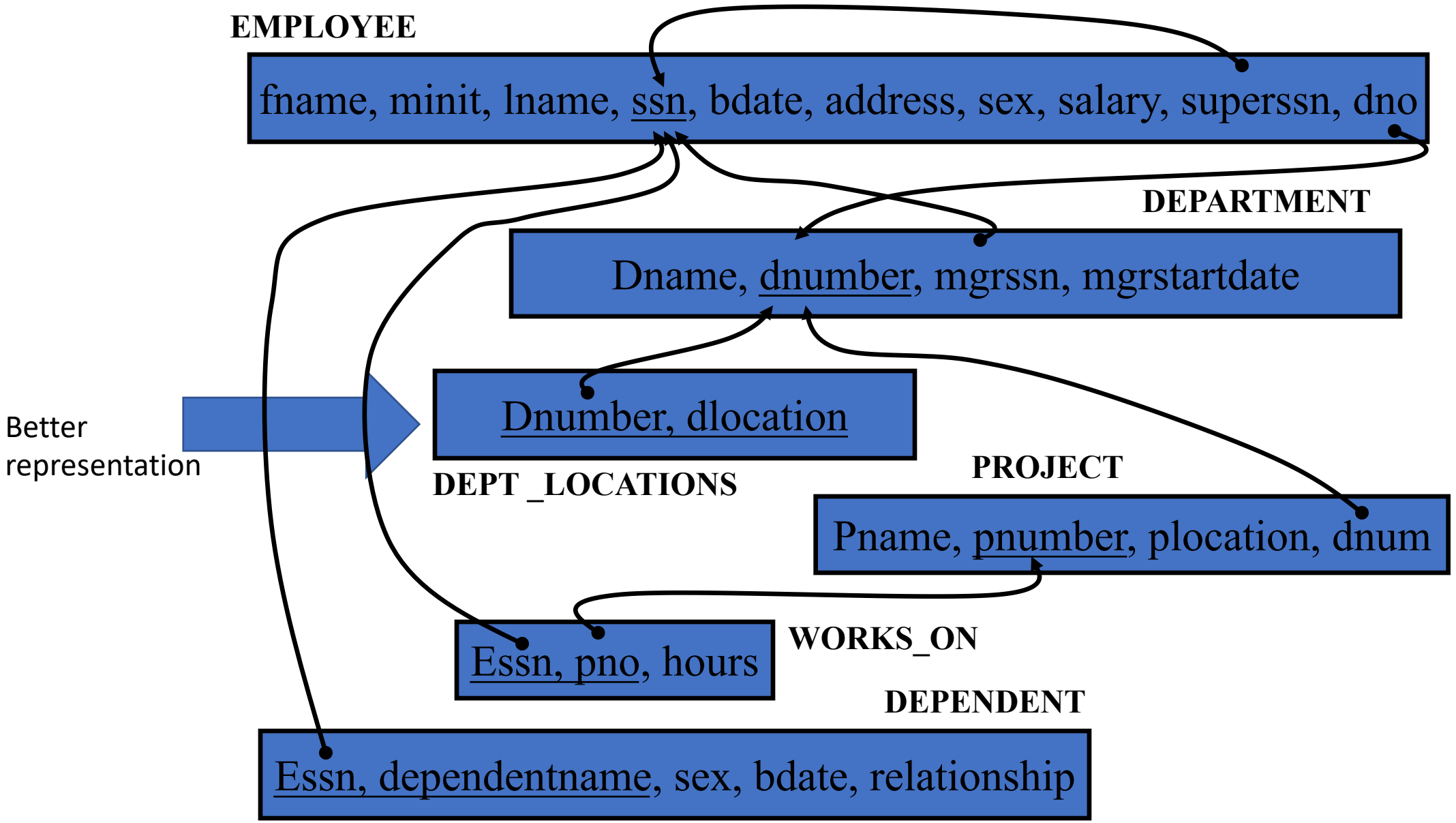
A given employee would be the manager of a department

Relationships




name
number
location
department
works for
1
N
name
sex
address
salary
employee
ssn
bdate
degree
startdate
manages
1
1
number of employees
controls
1
N
supervisor
supervisee
N
hours
works on
M
project
supervision
1
dependents of
N
name
number
location
dependent
name
sex
birthdate
relationship


EMPLOYEE
fname, minit, lname, ssn, bdate, address, sex, salary, superssn, dno
DEPARTMENT
Dname, dnumber, mgrssn, mgrstartdate, location
PROJECT
Pname, pnumber, plocation, dnum
WORKS_ON
Essn, pno, hours
DEPENDENT
Essn, dependentname, sex, bdate, relationship


EMPLOYEE
fname, minit, lname, ssn, bdate, address, sex, salary, superssn, dno
DEPARTMENT
Dname, dnumber, mgrssn, mgrstartdate
Better representation
Dnumber, dlocation
DEPT _LOCATIONS
PROJECT
Pname, pnumber, plocation, dnum
Essn, pno, hours
WORKS_ON
DEPENDENT
Essn, dependentname, sex, bdate, relationship

Mapping an ER to a Relational Database

Mapping an ER to a Relational Database

To map an ER to a relational database, five rules are defined to govern how tables are constructed.

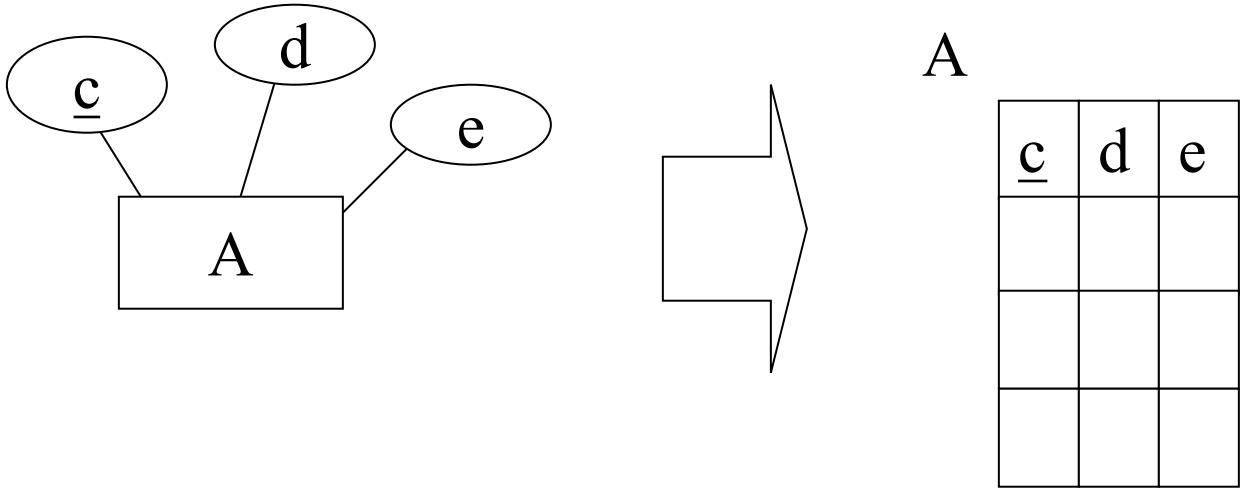
1. Rule for entity types
2. Rule for relationships
3. Rule for attributes
4. Rule for generalization/specialization hierarchies (will not be discussed in this course)
5. Rule for participation constraint

Mapping an ER to a Relational Database

- Entities: each entity set is implemented with a separate relation.
- The PK (Primary Key) is chosen from the set of keys available.

Mapping an ER to a Relational Database

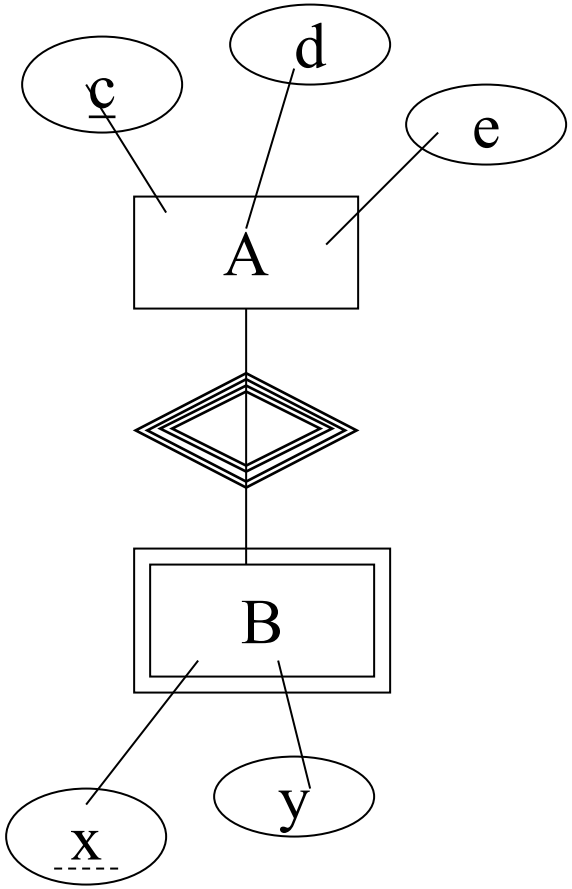
Example: Entities

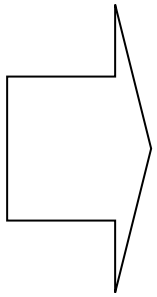


Mapping an ER to a Relational Database

- **Entities**: each entity is implemented with a separation relation
- **Weak Entities** are mapped to their own relation
- The PK of the weak entity is the combination of the PKs of entities related through identifying relationships and the discriminator (partial key) of the weak entity.

Mapping an ER to a Relational Database





B

c	x	y

Mapping an ER to a Relational Database

- Relationships We'll consider binary relationships only.

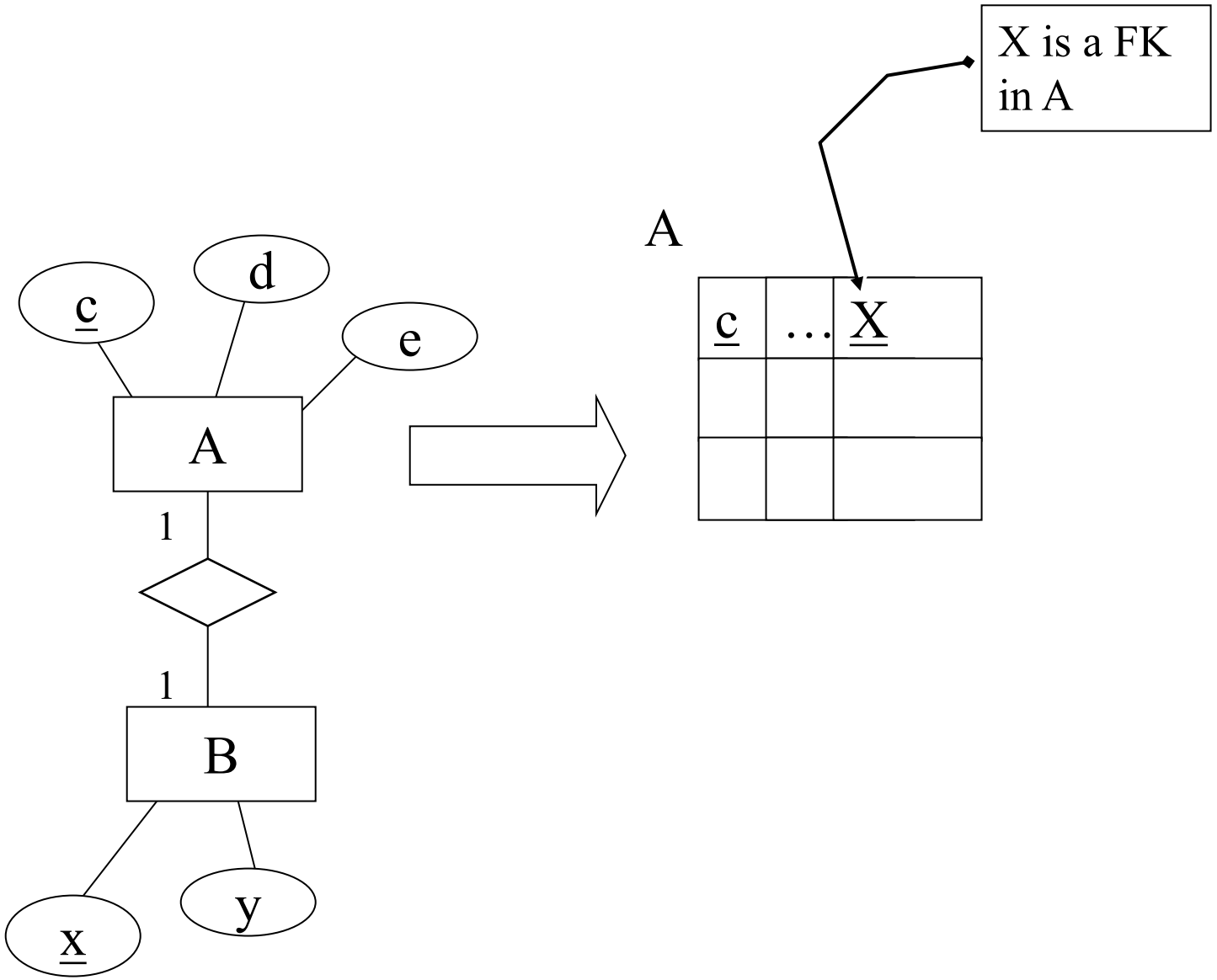
  All relationships involve the use of foreign keys: the PK attribute of one entity set is added as an attribute to the relation representing the other entity set.

- Binary One-To-One

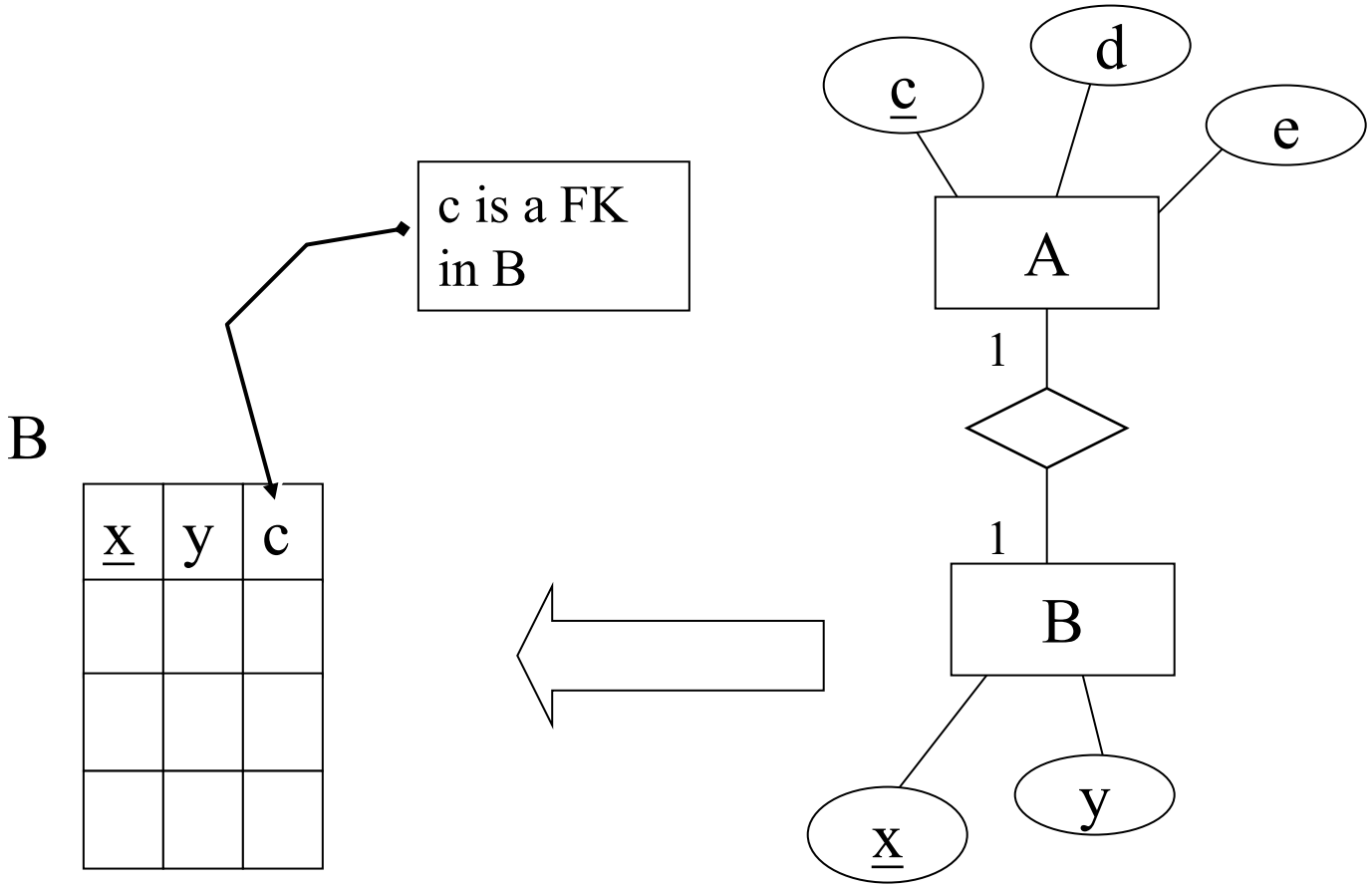
  In general, with a one-to-one relationship, you have a choice regarding where to implement the relationship. You may choose to place a **foreign key** in one of the two relations, or in both.

Example: 1-1

If the participation constraint on an entity set is mandatory, we must choose the table for that entity set.

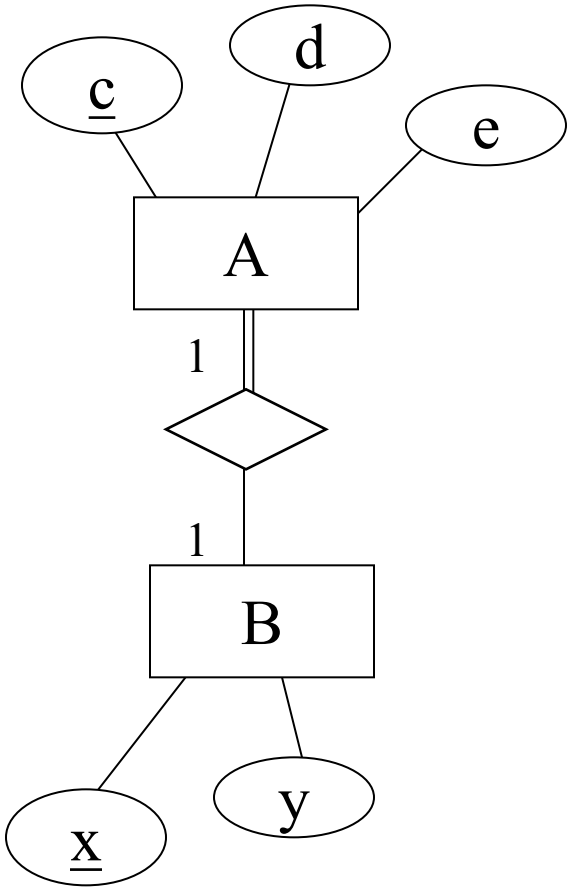


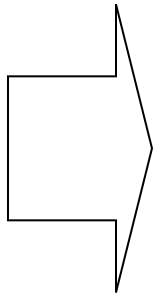
Example: 1-1

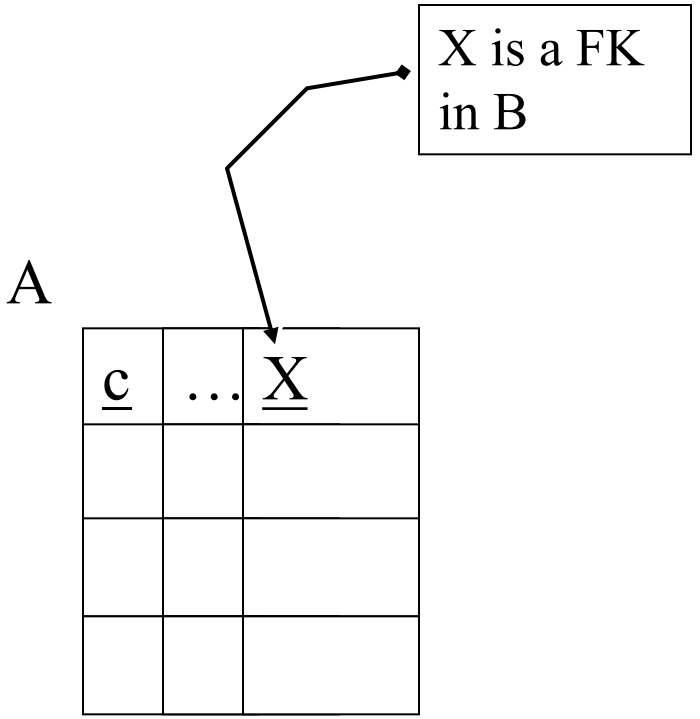


If the participation constraint on an entity set is mandatory, we must choose the table for that entity set.

If the participation constraint on an entity set is mandatory, we must choose the table for that entity set.



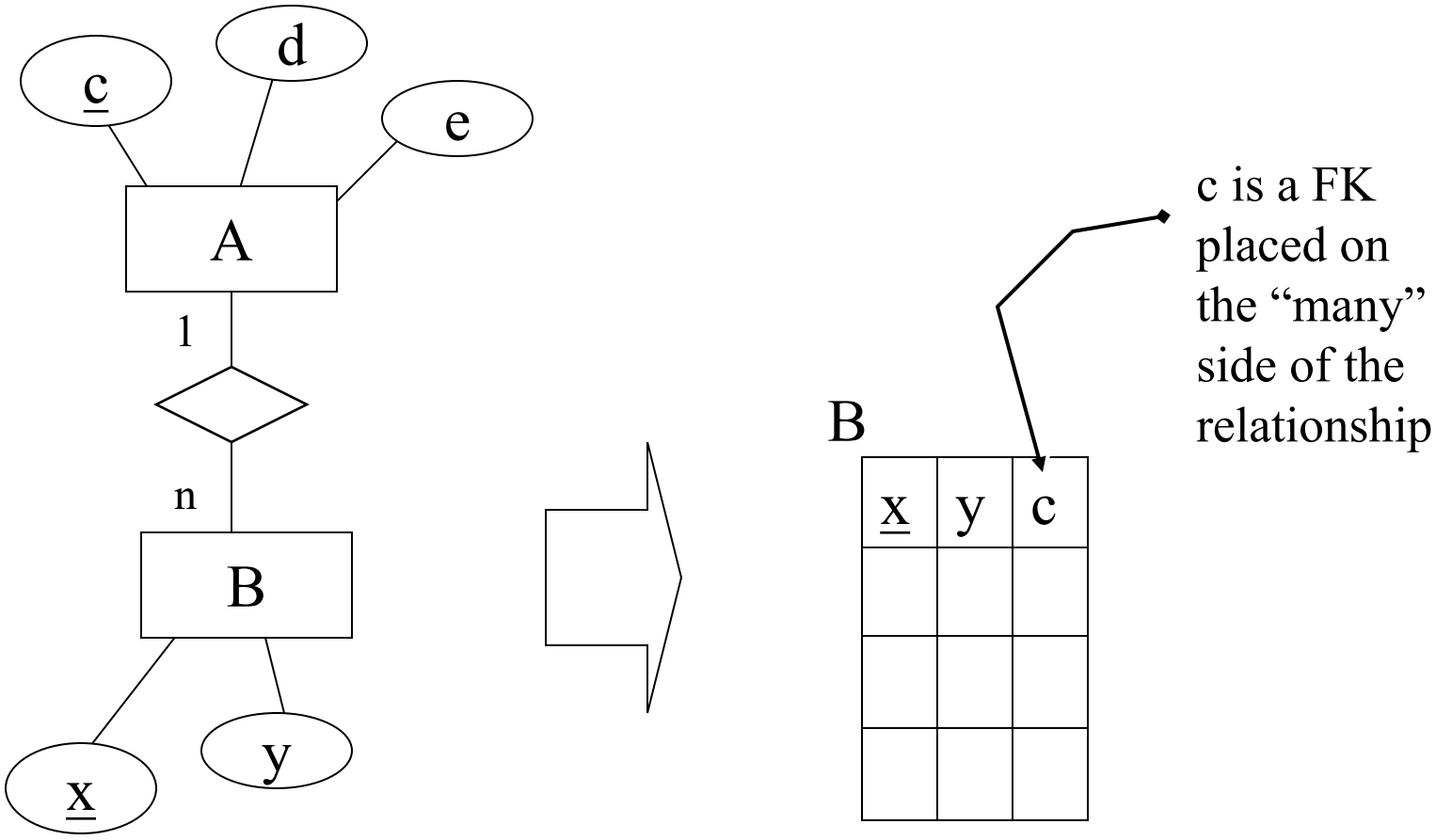




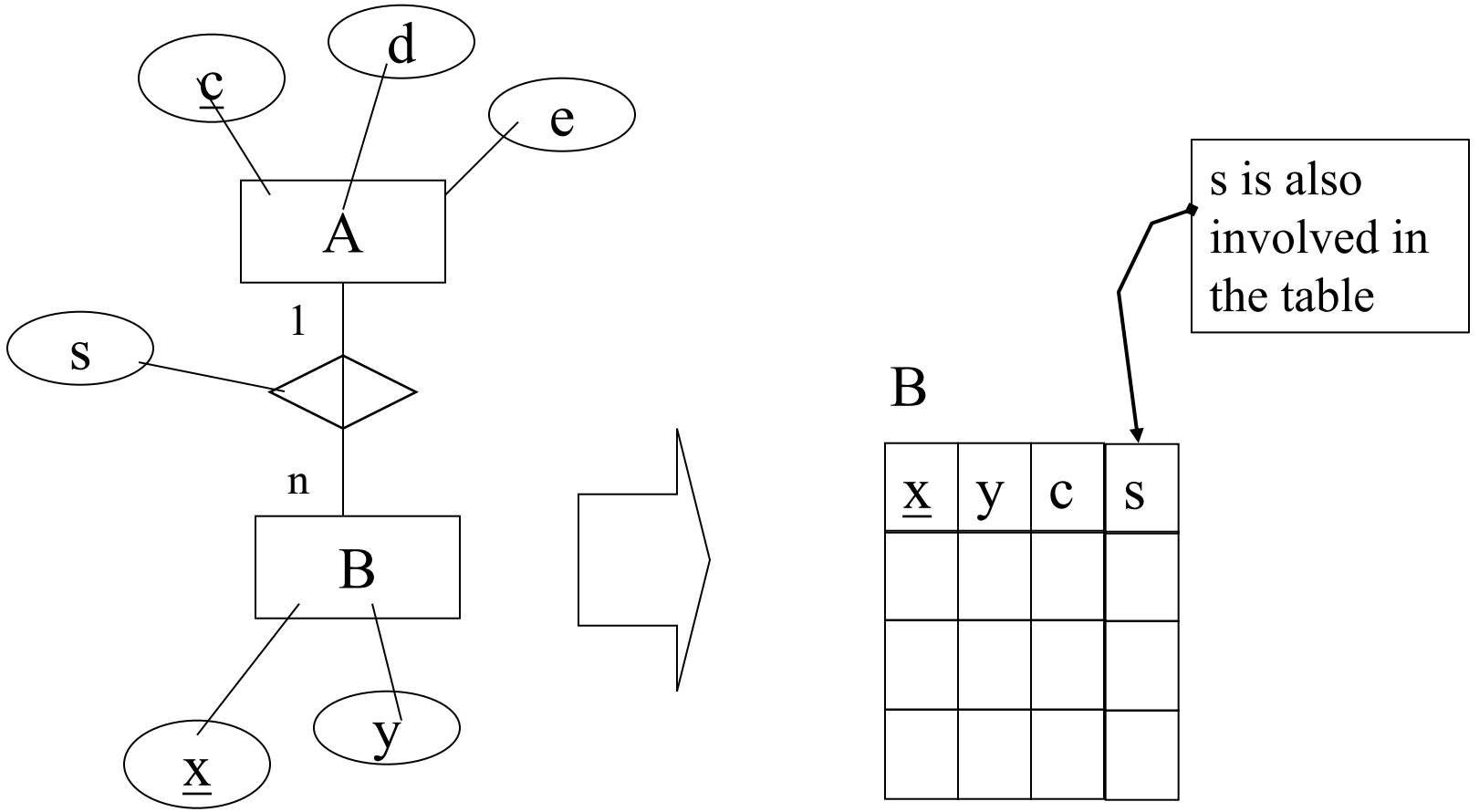
Binary One-To-Many

- With a one-to-many relationship you must place a foreign key in the relation corresponding to the *many* side of the relationship.

Example: 1-n


c
d
e
A
1
n
B
x
y
B
x
y
c
c is a FK placed on the "many" side of the relationship

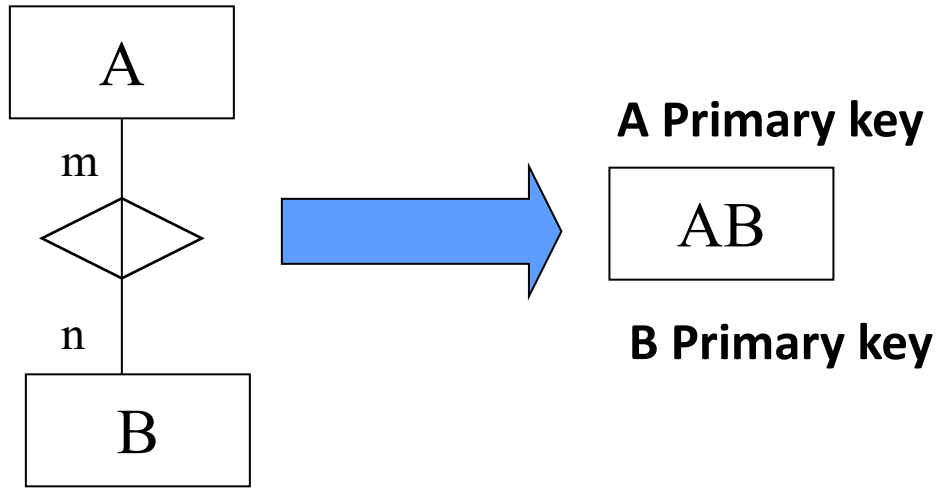
Example: 1-n



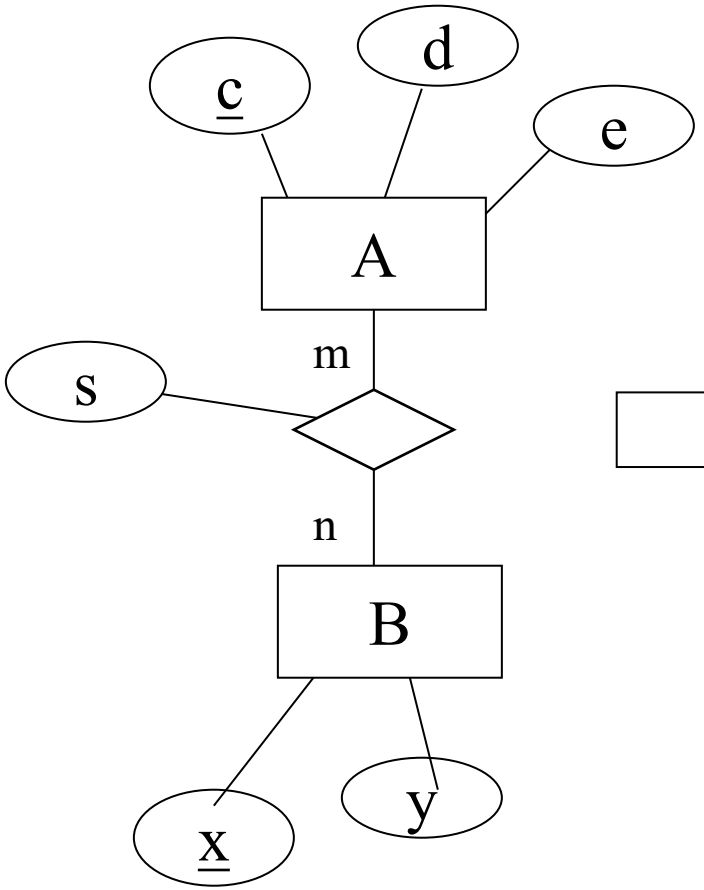
Many-to-Many

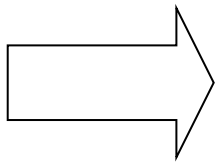
A many-to-many relationship must be implemented with a **separate relation** for the relationship.

This new relation will have a composite primary key comprising the primary keys of the participating entity sets plus any discriminator attribute.



Many-to-Many





AB

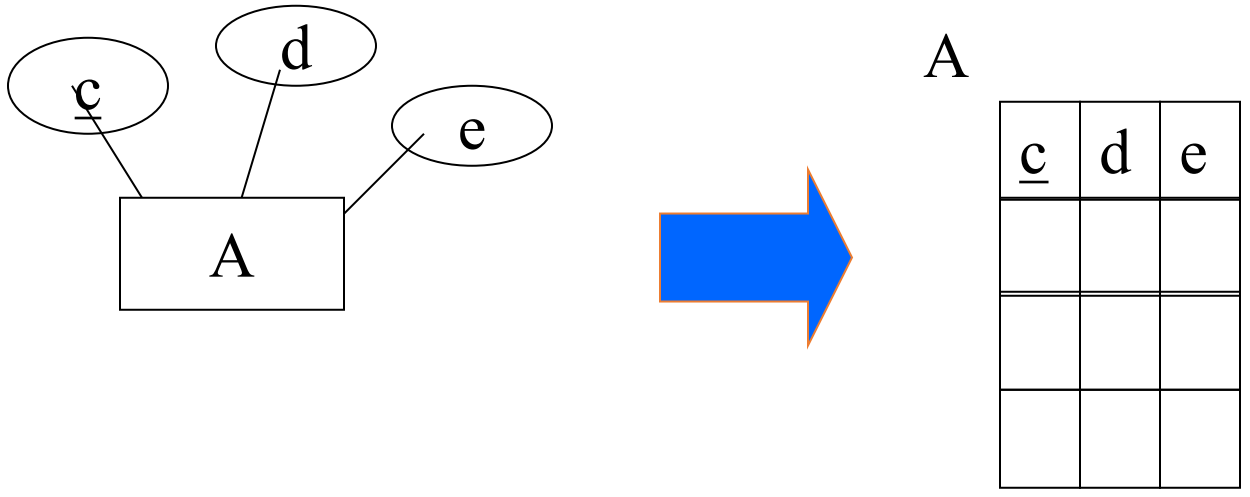
x	c	s

x and c are FKs, and together they form the PK of AB

Attributes

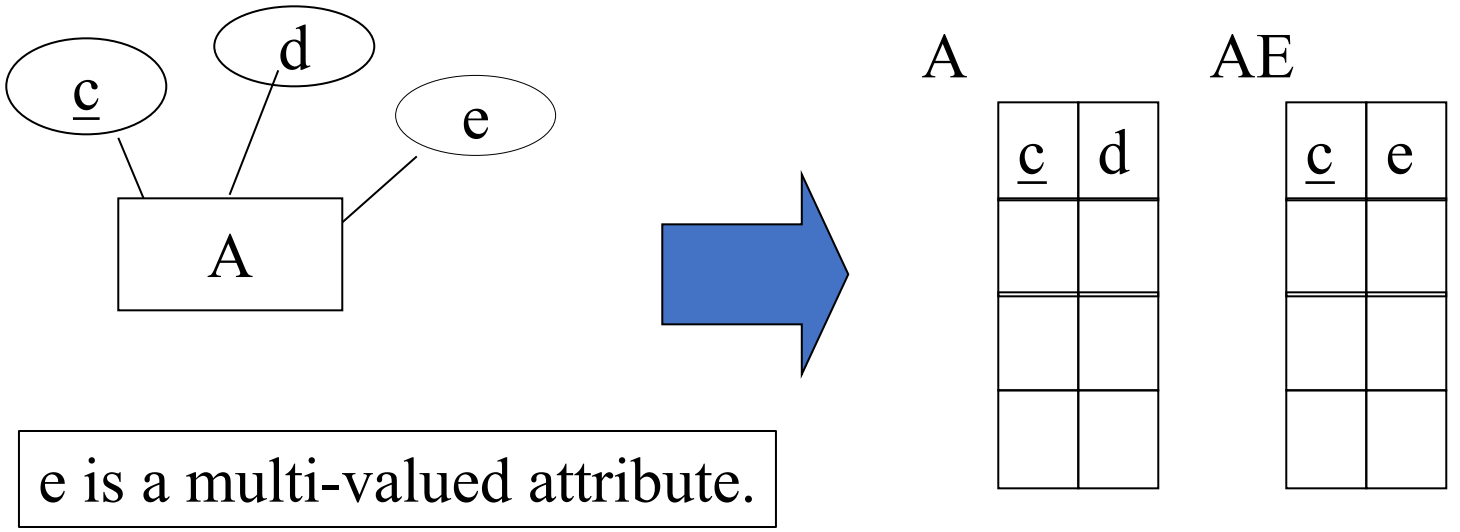
- All attributes, with the exception of derived and composite attributes, must appear in relations.
- Simple, atomic

  These are included in the relation created for the pertinent entity set, many-to-many relationship, or *n*-ary relationship.



Multi-valued attributes

- Each multi-valued attribute is implemented using a new relation. This relation will include the primary key of the original entity set.
- The primary key of the new relation will be the primary key of the original entity set and the multi-valued attribute. Note that in this new relation, the attribute is no longer multi-valued.

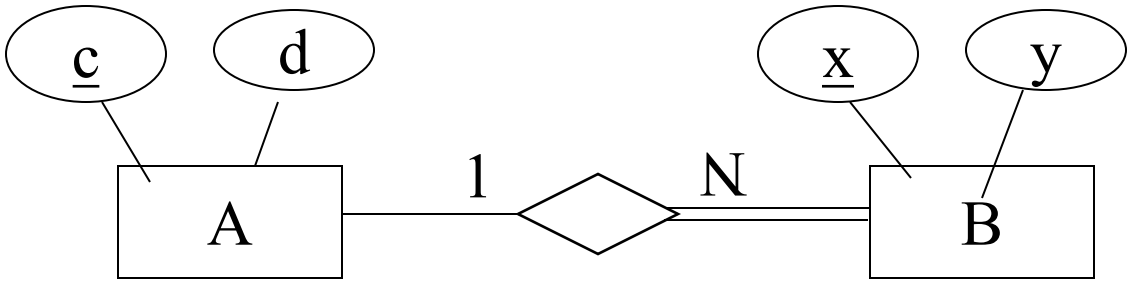


Participation in Constraints

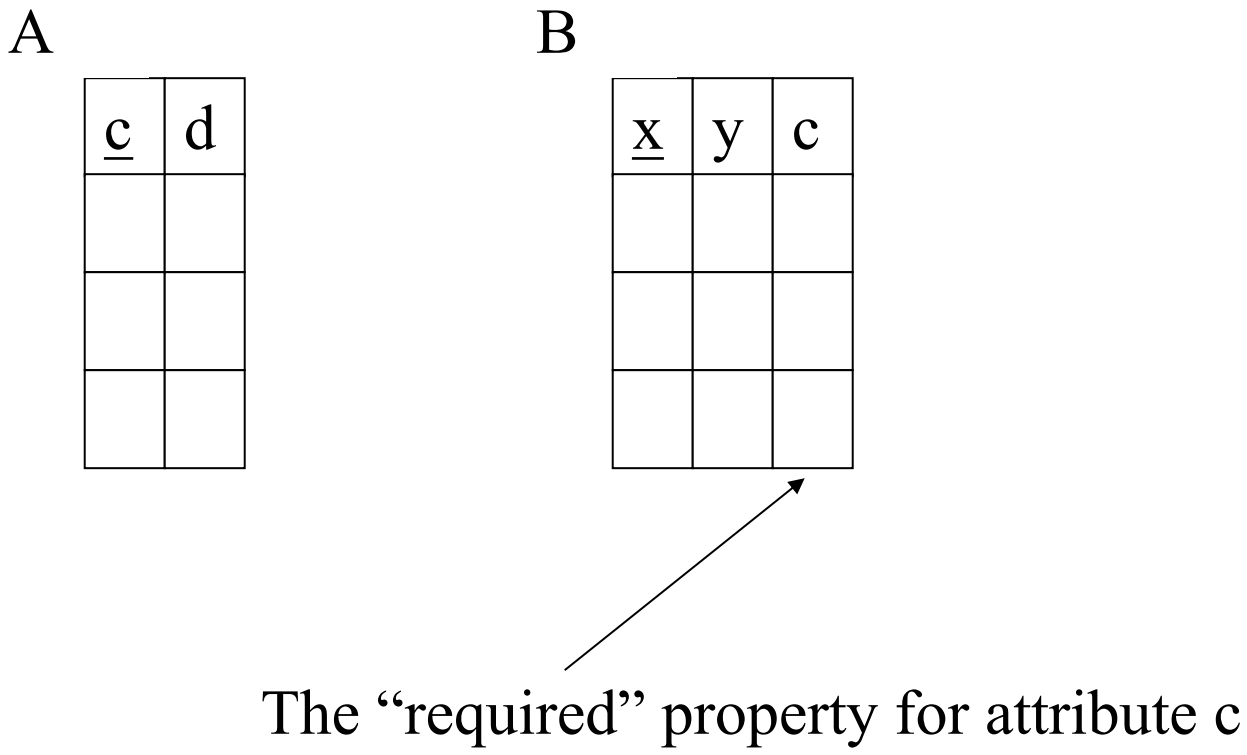
If a relationship is mandatory for an entity set, then if the entity set is on the "many" side of the relationship, then a specification is required to ensure a foreign key has a value, and that it cannot be null

setting the 'required' property for the FK in MySQL, or

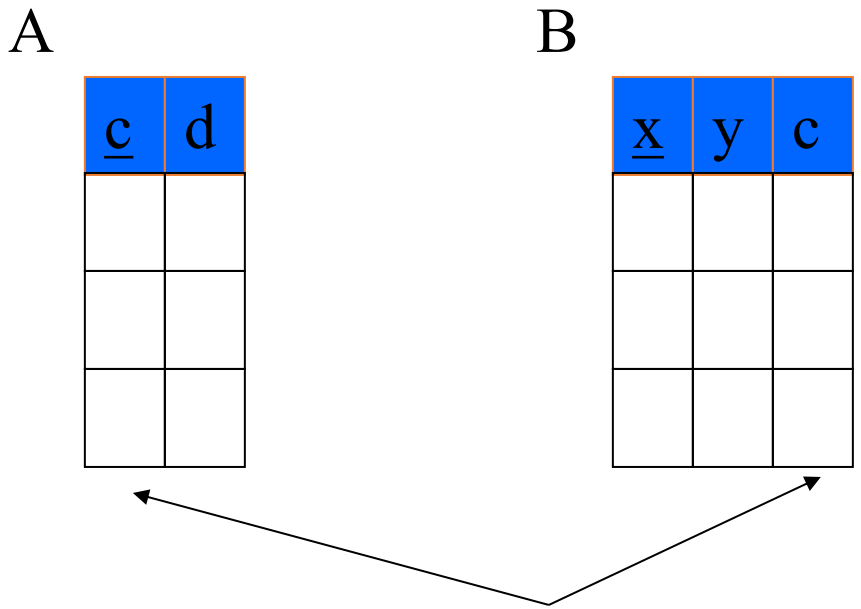
NOT NULL constraint in the DDL.



Participation in Constraints



Participation in Constraints



A program should be produced to check that any value appearing in c-column in table A must appear at least once in c-column in table B.

Summary

- **Entities:**
  - Each entity set is implemented with a separate relation.
- **Weak Entities**
  - Weak entities are mapped to their own relation
  - The PK of the weak entity is the combination of the PKs of entities related through identifying relationships and the discriminator (partial key) of the weak entity;

Foreign Keys

All relationships involve foreign keys. If the relationship is identifying then the primary key of the strong entity must be propagated to the relation representing the weak entity.

- Binary **One-To-One**
  - In general, with a one-to-one relationship, you have a choice regarding where to implement the relationship. You may choose to place a foreign key in one of the two relations, or in both.

- Binary **One-To-Many**
  - With a one-to-many relationship you must place the foreign key in the relation corresponding to the *many* side of the relationship.

Foreign Keys

- Binary **Many-To-Many**
  - A many-to-many relationship must be implemented with a separate relation for the relationship.
  - This new relation will have a composite primary key comprising the primary keys of the participating entity sets plus any discriminator attribute.

- **n-ary**, n>2
  - new relation is generated for the n-ary relationship.
  - This new relation will have a composite primary key comprising the primary keys of the participating entity sets plus any discriminator attribute.
  - If the cardinality related for any entity set is 1, then the primary key of that entity set is only included as a foreign key and not as part of the primary key of the new relation.

Foreign Keys, Referential Integrity

- Foreign key : Set of fields in one relation that is used to `refer' to a tuple in another relation. (Must correspond to primary key of the second relation.) Like a `logical pointer'.
- E.g. sid is a foreign key referring to Students:
  - Enrolled(sid: string, cid: string, grade: string)
  - If all foreign key constraints are enforced, referential integrity is achieved, i.e., no dangling references.
  - Can you name a data model w/o referential integrity?
    - Links in HTML!

Foreign Keys, Referential Integrity

- sid is a foreign key referring to Students:
  - Enrolled(sid, cid, grade)
  - If all foreign key constraints are enforced, referential integrity is achieved, i.e., no dangling references

Enrolled

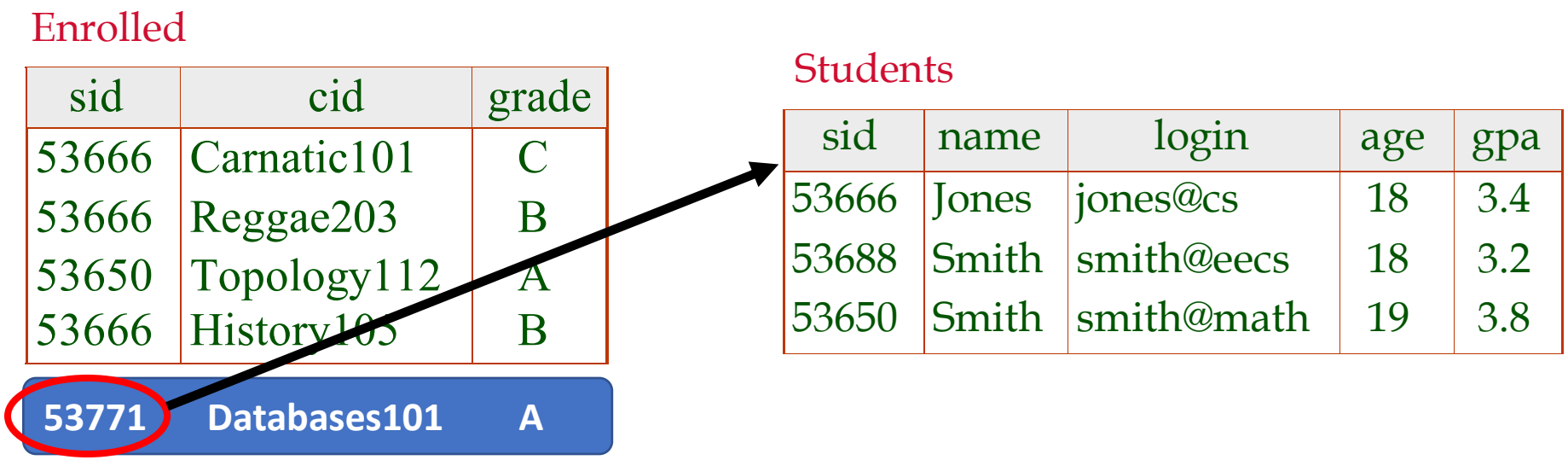
sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enforcing Referential Integrity

- Consider Students and Enrolled; sid in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted?



Enforcing Referential Integrity

- Consider Students and Enrolled; sid in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted? **Reject it!**

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B
53771	Databases101	A

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enforcing Referential Integrity

- Consider Students and Enrolled; sid in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted? Reject it!
- Options if a Students tuple is deleted:
  - Also delete all Enrolled tuples that refer to it (“CASCADE”)
  - Disallow deletion of a Students tuple that is referred to (“NO ACTION”).
  - Set sid in Enrolled tuples that refer to it to a default sid (“SET DEFAULT”).
  - In SQL, also: Set sid in Enrolled tuples that refer to it to a special value null, denoting `unknown’ or `inapplicable’ (“SET NULL”)
- Similar if primary key of Students tuple is updated.

Foreign Keys, Referential Integrity

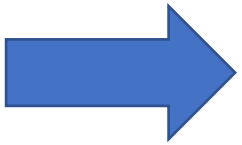
Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

```
CREATE TABLE Enrolled (
        sid VARCHAR(20)
        cid  VARCHAR(20),
        grade VARCHAR(2),
        PRIMARY KEY  (sid,cid)
)
```



```
CREATE TABLE Enrolled (
   sid VARCHAR(20),
    cid VARCHAR(20),
    grade VARCHAR(2),
    PRIMARY KEY  (sid,cid),
    FOREIGN KEY (sid)
      REFERENCES Students(sid)
            ON DELETE CASCADE
)
```

Referential Integrity in SQL

```
CREATE TABLE Enrolled (
  sid CHAR(20),
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid)
    REFERENCES Students (sid)
          ON DELETE CASCADE
)
```

SQL Supports all 4 operations on deletes and updates

- Default is NO ACTION (delete/update is rejected)
- CASCADE (also delete all tuples that refers to delete tuple)
- SET NULL/ SET DEFAULT (set foreign key value on referencing tuple)

Example

```
INSERT INTO Students VALUES ('53666', 'Jones', 'Jones@cs', 18, 3.4);
INSERT INTO Students VALUES ('53688', 'Smith', 'Smith@eecs', 18, 3.2);
INSERT INTO Students VALUES ('53650', 'Smith', 'smith@math', 19, 3.8);
```

sid	name	login	age	gpa
53650	Smith	smith@math	19	4
53666	Jones	Jones@cs	18	3
53688	Smith	Smith@eecs	18	3

Example

```
INSERT INTO Enrolled VALUES ('53666', 'Carnatic101', 'C');
INSERT INTO Enrolled VALUES ('53666', 'Reggae203', 'B');
INSERT INTO Enrolled VALUES ('53650', 'Topology112', 'A');
INSERT INTO Enrolled VALUES ('53666', 'History', 'B');
```

sid	cid	grade
53650	Topology112	A
53666	Carnatic101	C
53666	History	B
53666	Reggae203	B

Example

sid	name	login	age	gpa
53650	Smith	smith@math	19	4
53666	Jones	Jones@cs	18	3
53688	Smith	Smith@eecs	18	3

X

```
DELETE FROM Students WHERE sid = '53650';
```

sid	cid	grade
~~53650~~	~~Topology112~~	~~A~~
53666	Carnatic101	C
53666	History	B
53666	Reggae203	B