FROM E/R MODEL TO DATABASE SCHEMA

Two Steps

- *Restructure* the ER schema to improve it, based on criteria
- *Translate* the schema into the relational model

1. RESTRUCTURING AN E/R MODEL

Restructuring Overview

Input: E/R Schema

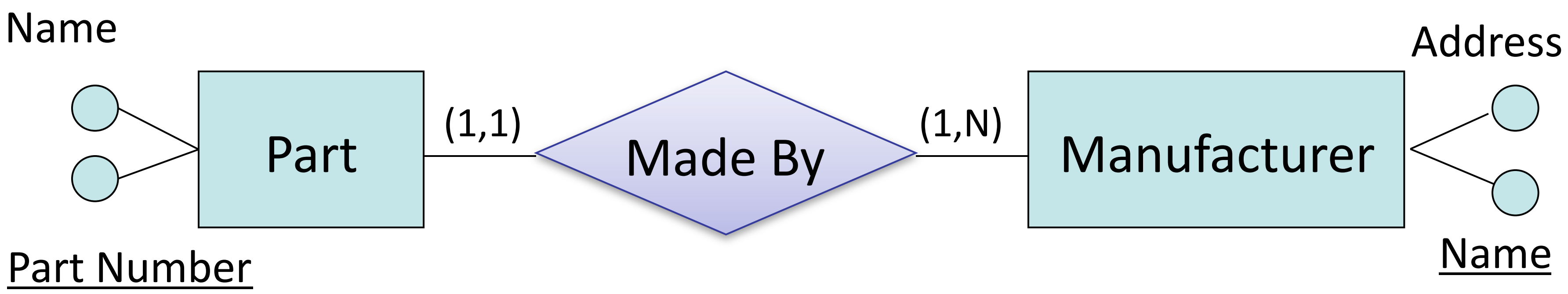
Output: Restructured E/R Schema

Restructuring includes:

- Analysis of redundancies
- Choosing entity set vs attribute
- Limiting the use of weak entity sets
- Selection of keys
- Creating entity sets to replace attributes with cardinality greater than one

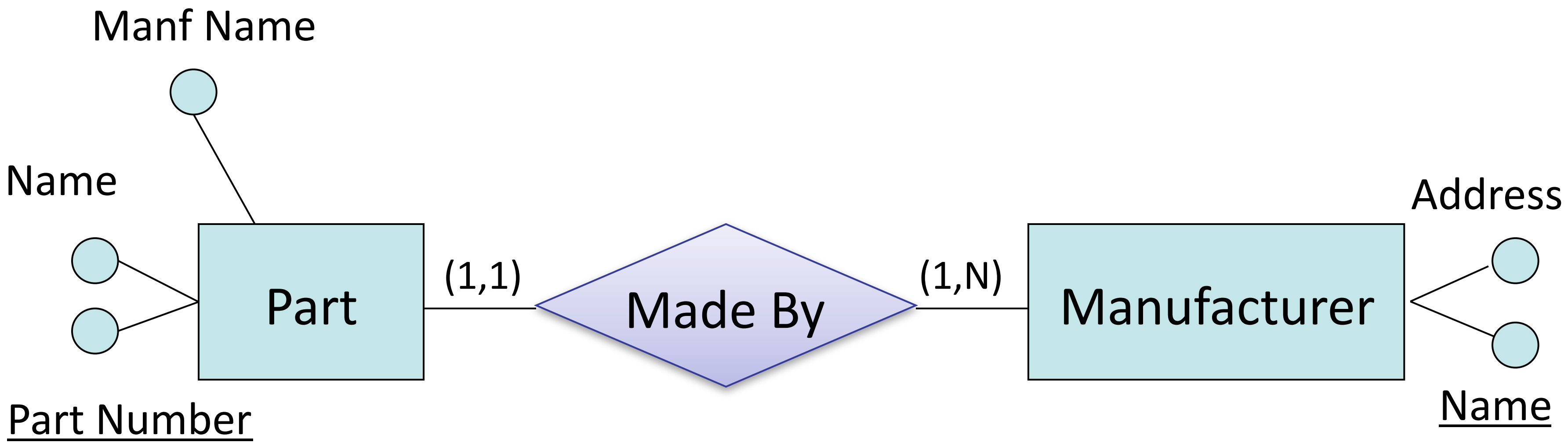
Example: no redundancy

It is not redundant to have Name twice.



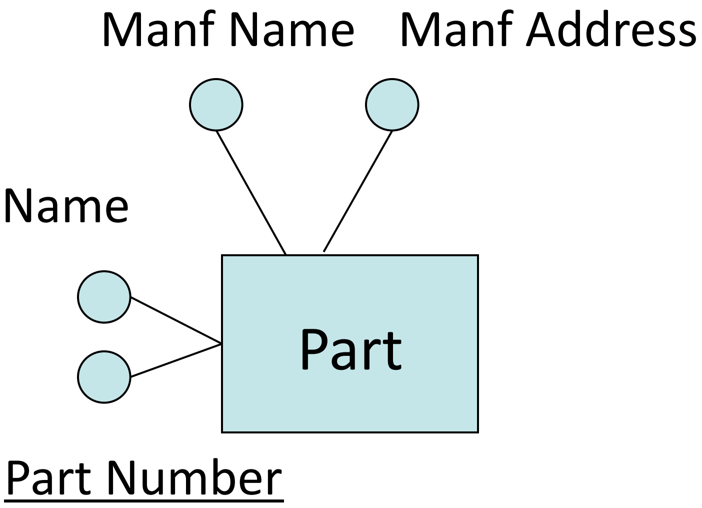
Example: redundancy

What is redundant here?



Example: redundancy

What is redundant here?



Entity Sets Versus Attributes

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one non-key attribute.

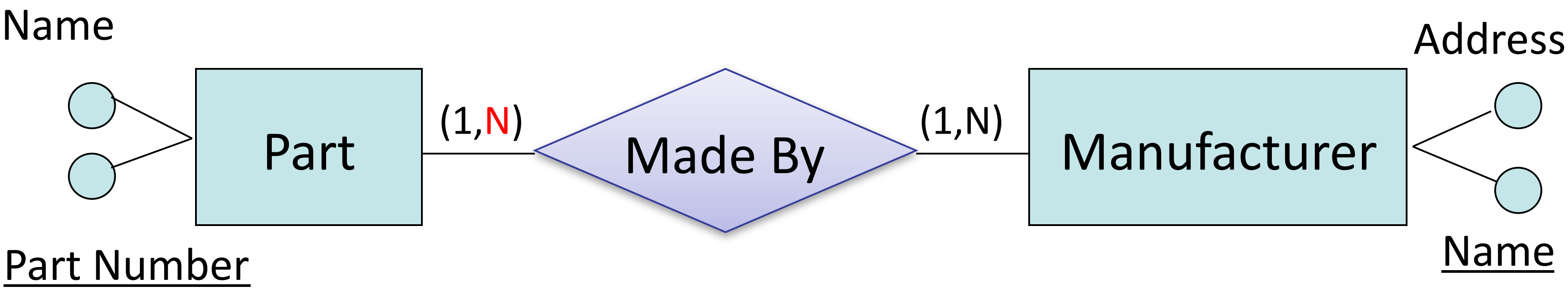
    or

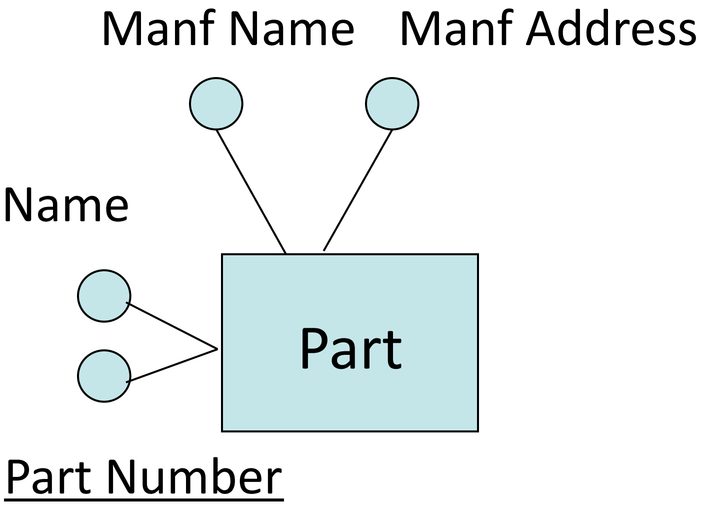
  - It is the “many” in a many-one or many-many relationship.
- Rules of thumb
  - A “thing” in its own right => Entity Set
  - A “detail” about some other “thing” => Attribute

Really this is just about avoiding redundancy

E.S. vs. attributes: examples

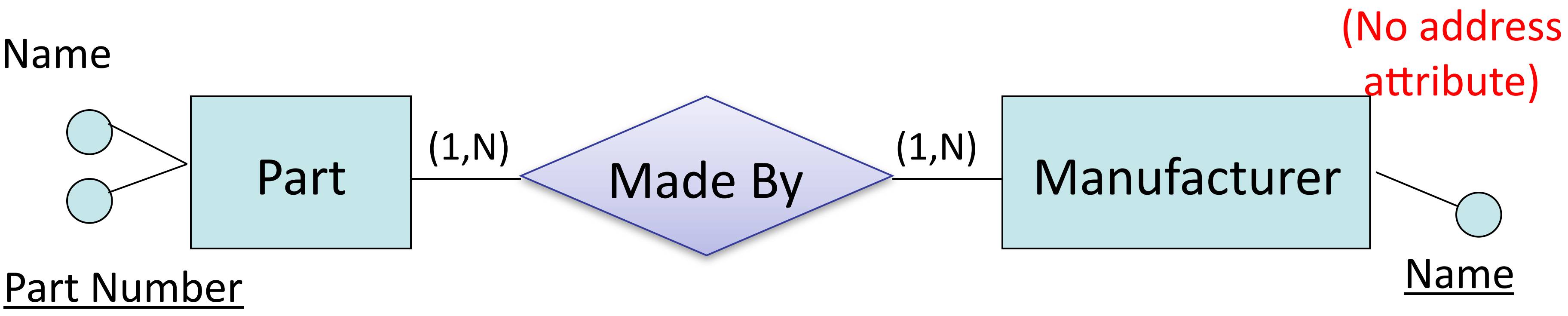
Domain fact change: A part can have more than one manufacturer …



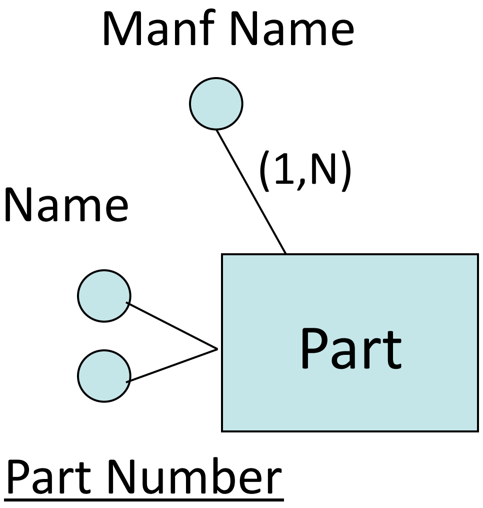


E.S. vs. attributes: examples

Domain fact change: Not representing Manufacturer address …

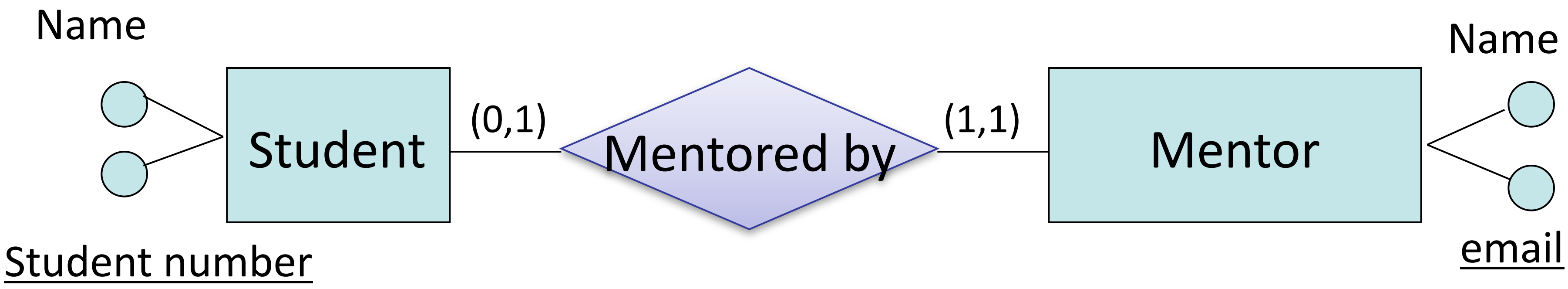


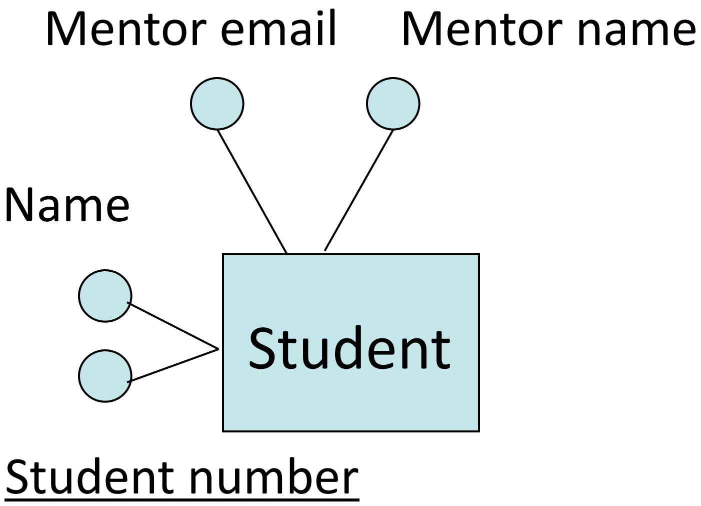
(No address attribute)



E.S. vs. attributes: examples

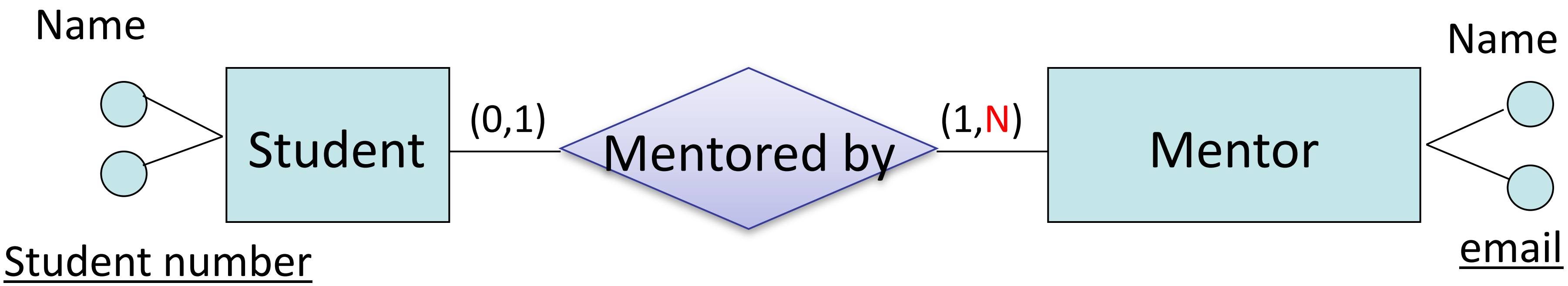
New domain

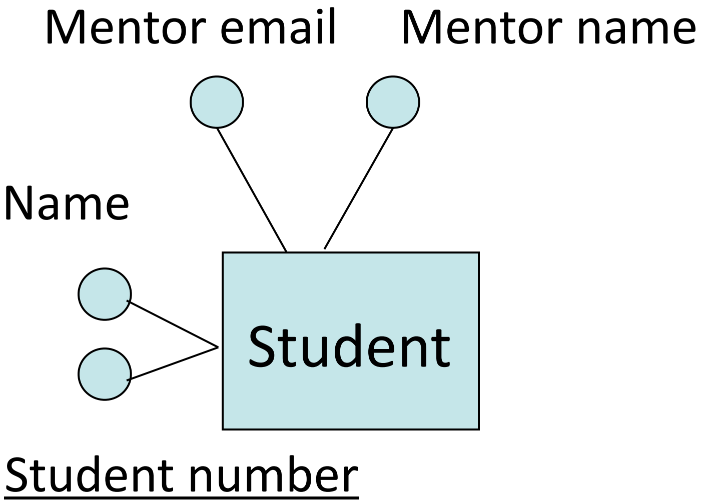




E.S. vs. attributes: examples

Domain fact change: A mentor can have more than one mentee …





When to use weak entity sets?

- The usual reason is that there is no global authority capable of creating unique ID's
- Example: it is unlikely that there could be an agreement to assign unique student numbers across all students in the world

Don’t Overuse Weak Entity Sets

- Beginning database designers often doubt that anything could be a key by itself
  - They make all entity sets weak, supported by all other entity sets to which they are linked
- It is usually better to create unique IDs
  - Social insurance number, automobile VIN, etc.
  - Useful for many reasons (next slide)

Selecting a Primary Key

- Every relation must have a primary key
- The criteria for this decision are as follows:
  - Attributes with null values cannot form primary keys
  - One/few attributes is preferable to many attributes
  - Internal keys preferable to external ones (weak entities depend for their existence on other entities)
  - A key that is used by many operations to access instances of an entity is preferable to others

Keeping keys simple

Multi-attribute and/or string keys…

- … are wasteful
  - e.g. Movies(<u>title</u>, <u>year</u>, …): 2 attributes, ~16 bytes
  - Number of movies ever made << 2^{32} (4 bytes)

  => Integer movieID key saves 75% space and a lot of typing
- … break encapsulation
  - e.g. Patient(<u>firstName</u>, <u>lastName</u>, <u>phone</u>, …)
  - Security/privacy hole

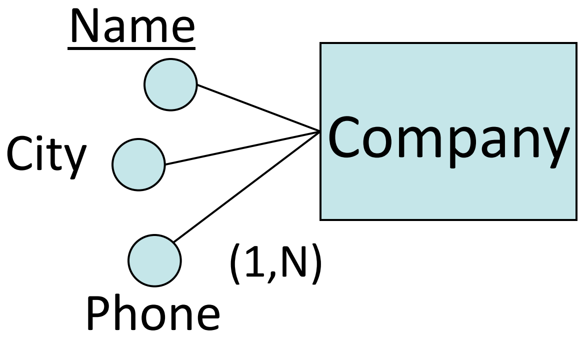
  => Integer patientID prevents information leaks
- … are brittle (nasty interaction of above two points)
  - Name or phone number change? Parent and child with same name?
  - Patient with no phone? Two movies with same title and year?

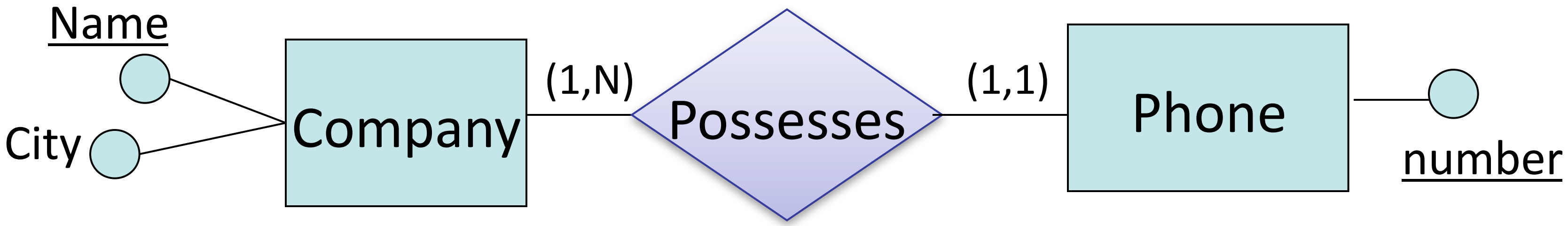
  => Internal ID always exist, are immutable, unique

Also: computers are really good at integers…

Attributes with cardinality > 1

- The relational model doesn't allow multi-valued attributes. We must convert these to entity sets.





2. TRANSLATING AN E/R MODEL INTO A DB SCHEMA

Translation into a Logical Schema

Input: E/R Schema

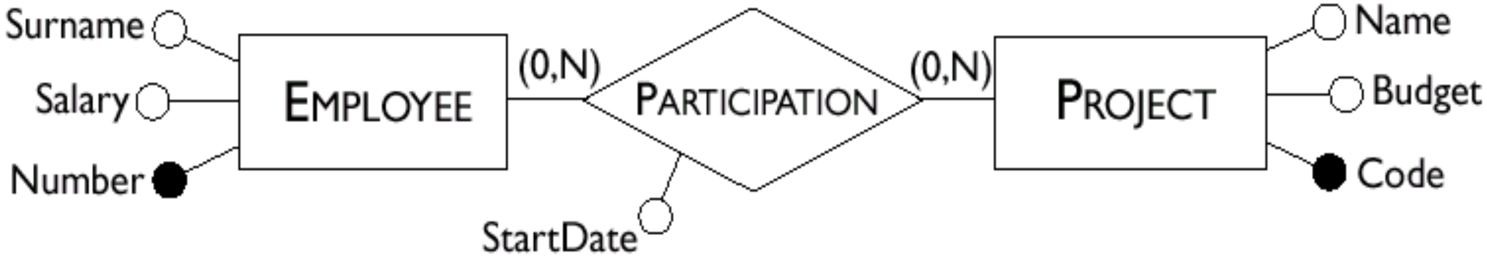
Output: Relational Schema

- Starting from an E/R schema, an equivalent relational schema is constructed
  - "equivalent": a schema capable of representing the same information
- A good translation should also:
  - not allow redundancy
  - not invite unnecessary null values

The general idea

- Each entity set becomes a relation. Its attributes are
  - the attributes of the entity set.
- Each relationship becomes a relation. It’s attributes are
  - the keys of the entity sets that it connects, plus
  - the attributes of the relationship itself.
- We’ll see opportunities to simplify.

Many-to-Many Binary Relationships

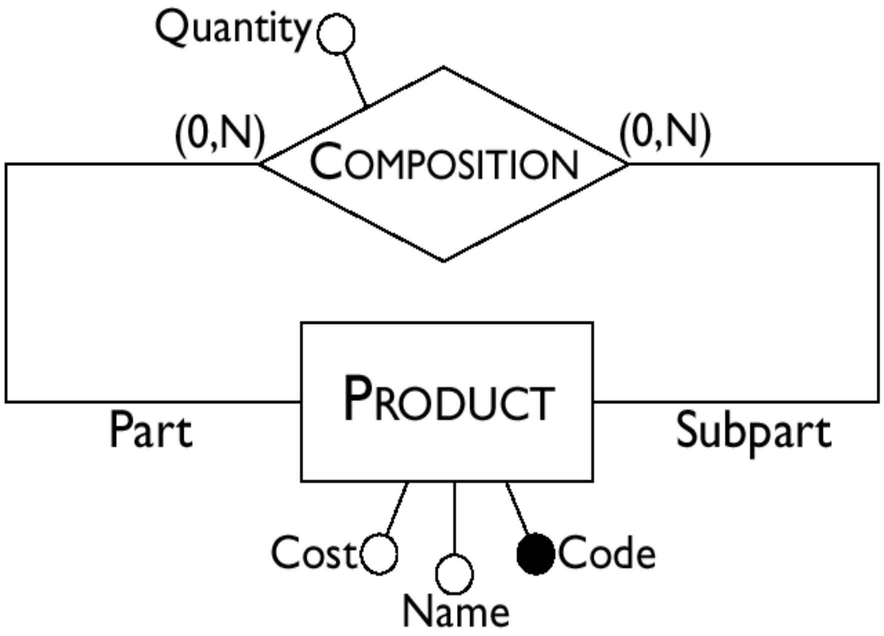


Employee(Number, Surname, Salary)

Project(Code, Name, Budget)

Participation(Number, Code, StartDate)

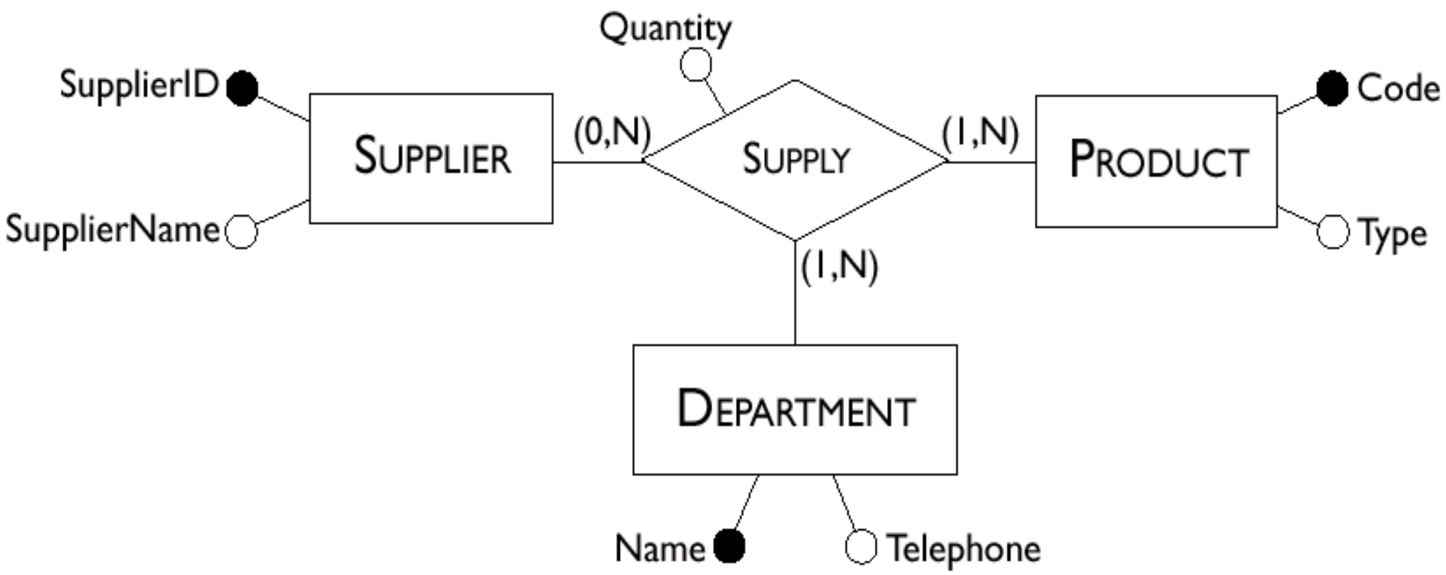
Many-to-Many Recursive Relationships



Product(Code, Name, Cost)

Composition(Part, SubPart, Quantity)

Many-to-Many Ternary Relationships



Supplier(<u>SupplierID</u>, SupplierName)

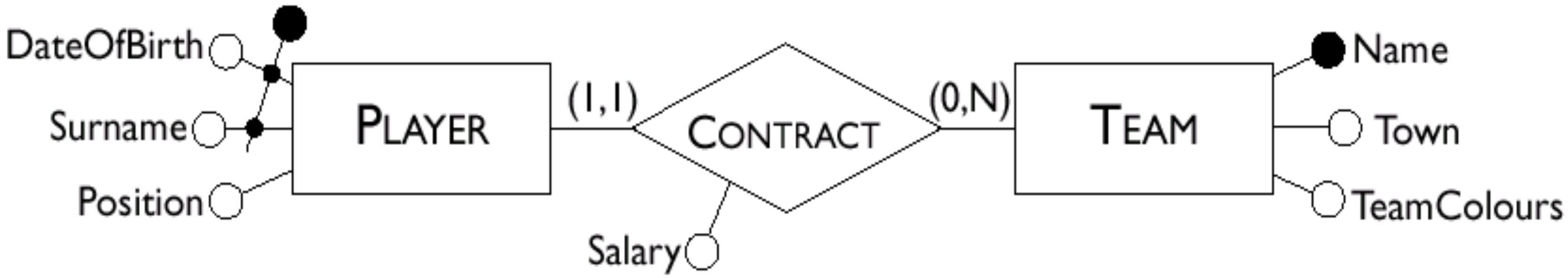
Product(<u>Code</u>, Type)

Department(<u>Name</u>, Telephone)

Supply(<u>Supplier</u>, <u>Product</u>, <u>Department</u>, Quantity)

One-to-Many Relationships

with mandatory participation for one



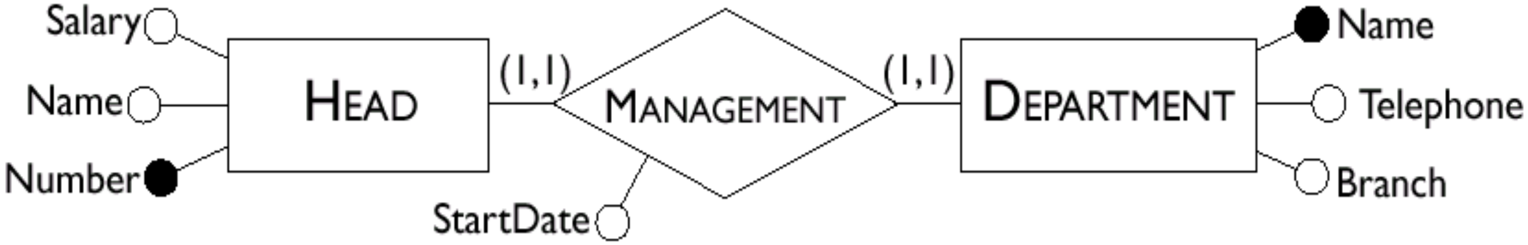
Player(Surname,DateOfBirth, Position)
Team(Name, Town, TeamColours)
Contract(PlayerSurname, PlayerDateOfBirth, Team, Salary)

OR

Player(Surname,DateOfBirth, Position, TeamName, Salary)
Team(Name, Town, TeamColours)

One-to-One Relationships

with mandatory participation for both



Head(<u>Number</u>, Name, Salary, Department, StartDate)

Department(<u>Name</u>, Telephone, Branch)

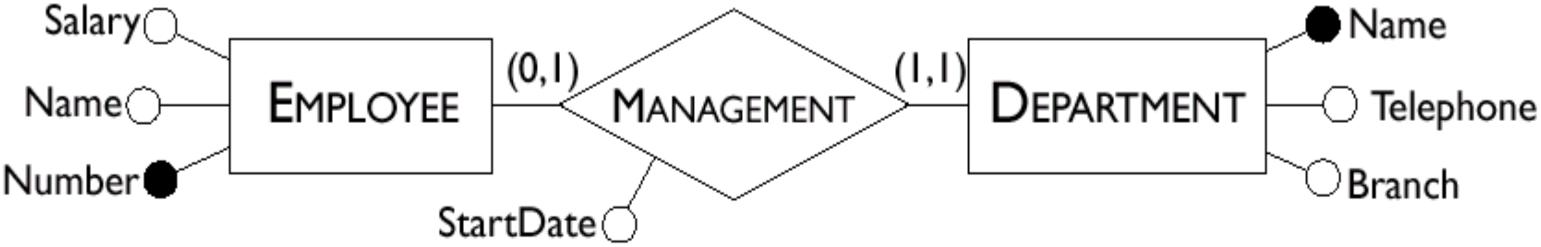
Or

Head(<u>Number</u>, Name, Salary, StartDate)

Department(<u>Name</u>, Telephone, HeadNumber, Branch)

One-to-One Relationships

with optional participation for one



Employee(Number, Name, Salary)

Department(Name, Telephone, Branch, Head, StartDate)

Or, if both entities are optional

Employee(Number, Name, Salary)

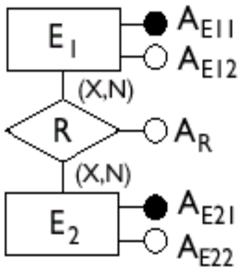
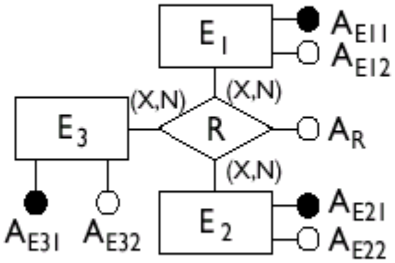
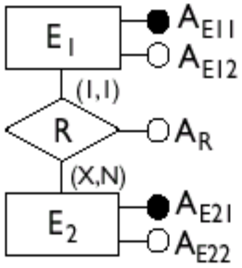
Department(Name, Telephone, Branch)

Management(Head, Department, StartDate)

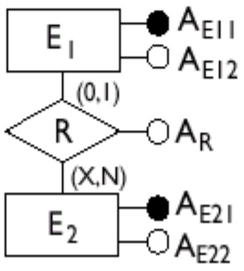
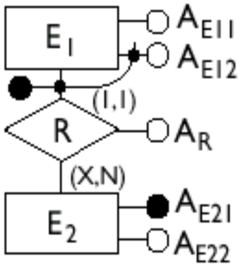
Summary of Types of Relationship

- many-to-many (binary or ternary)
- one-to-many
  - mandatory: (1,1) on the "one" side
  - optional: (0,1) on the "one" side
- one-to-one
  - both mandatory: (1,1) on both sides
  - one mandatory, one optional: (1,1) on one side and (0,1) on the other side
  - both optional: (0,1) on both sides

Summary of Transformation Rules

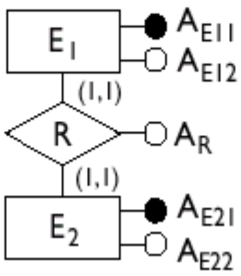
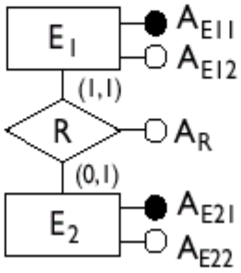
Type	Initial schema	Possible translation
Binary many-to-many relationship	 E1 A_E11 A_E12 (X,N) R A_R (X,N) E2 A_E21 A_E22	$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$
Ternary many-to-many relationship	 E1 A_E11 A_E12 (X,N) (X,N) E3 R A_R (X,N) A_E31 A_E32 E2 A_E21 A_E22	$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $E_3(\underline{A_{E31}}, A_{E32})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, \underline{A_{E31}}, A_R)$
One-to-many relationship with mandatory participation	 E1 A_E11 A_E12 (1,1) R A_R (X,N) E2 A_E21 A_E22	$E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

...More Rules...

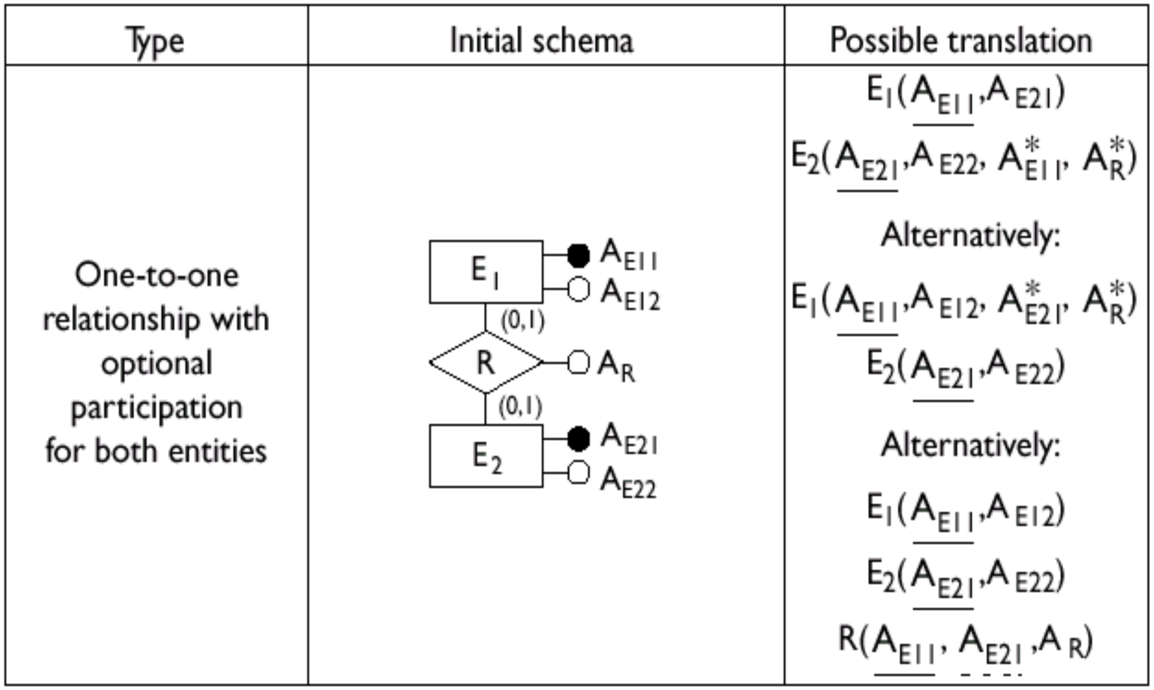
Type	Initial schema	Possible translation
One-to-many relationship with optional participation	 E_1 A_{E11} A_{E12} (0,1) R A_R (X,N) E_2 A_{E21} A_{E22}	$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$ Alternatively: $E_1(\underline{A_{E11}}, A_{E21}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$
Relationship with external identifiers	 E_1 A_{E11} A_{E12} (1,1) R A_R (X,N) E_2 A_{E21} A_{E22}	$E_1(\underline{A_{E12}}, \underline{A_{E21}}, A_{E11}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

✻ Indicates that nulls are allowed

…Even More Rules…

Type	Initial schema	Possible translation
One-to-one relationship with mandatory participation for both entities	 E_1 A_{E11} A_{E12} (1,1) R A_R (1,1) E_2 A_{E21} A_{E22}	$E_1(\underline{A_{E11}}, A_{E12}, \underset{\text{- - - -}}{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Alternatively: $E_2(\underline{A_{E21}}, A_{E22}, \underset{\text{- - - -}}{A_{E11}}, A_R)$ $E_1(\underline{A_{E11}}, A_{E12})$
One-to-one relationship with optional participation for one entity	 E_1 A_{E11} A_{E12} (1,1) R A_R (0,1) E_2 A_{E21} A_{E22}	$E_1(\underline{A_{E11}}, A_{E12}, \underset{\text{- - - -}}{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

…and the Last One…



Type	Initial schema	Possible translation
One-to-one relationship with optional participation for both entities	E$_1$ A_{E11} A_{E12} (0,1) R A_R (0,1) E$_2$ A_{E21} A_{E22}	$E_1(\underline{A_{E11}}, A_{E21})$ $E_2(\underline{A_{E21}}, A_{E22}, A^*_{E11}, A^*_R)$ Alternatively: $E_1(\underline{A_{E11}}, A_{E12}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Alternatively: $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$

Will the schema be "good"?

- If we use this process, will the schema we get be a good one?
- The process should ensure that there is no redundancy.
- But only with respect to what the E/R diagram represents.
- Crucial thing we are missing: functional dependencies. (We only have keys, not other FDs.)
- So we still need FD theory.

Redundancy can be *desirable*

- Disadvantages of redundancy:
  - More storage (but usually at negligible cost)
  - Additional operations to keep the data consistent
- Advantages of redundancy:
  - Speed: Fewer accesses necessary to obtain information
- How to decide to maintain or eliminate a redundancy? Examine:
  - the cost of operations that involve the redundant information and
  - the storage needed for the redundant information

  with and without the redundancy.

Performance analysis is required to decide about redundancy