Databases
Data-Modelling with ER

COMP 1531, 17s2

Aarthi Natarajan

Week 9

Story so far…

Last week, we studied

- **relational data model**
- how to map the relational model to define the **database schema** (e.g., for a relational data model, define the tables, rows and columns or field names and types of fields, constraints and relationships)

This week, we will look at:

- **ER model**
- Mapping an ER model **to** a relational model

Entity-Relationship (ER) Model

Designing a database

***Data modelling*:** an important early stage of database application development (aka "database engineering")

Typical steps in a database design

1. requirements analysis (identify data and operations)
2. **data modelling** (high-level, abstract)
3. database schema design (detailed, relational model/tables)
4. database implementation (create instance of schema)
5. build operations/interface (SQL, stored procedures, GUI)
6. performance tuning (physical re-design)
7. schema evolution (logical schema re-design)

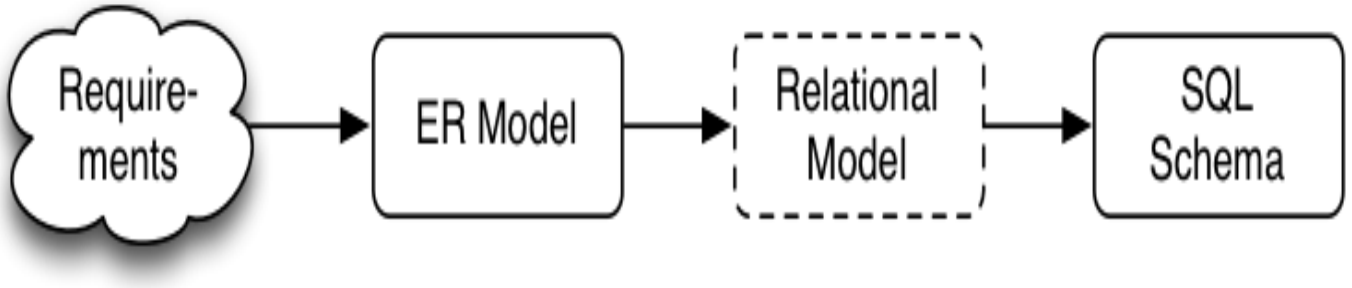
Designing a database

Two data models

- Logical: abstract model e.g., ER Model, OO Model
- Physical: record-based models e.g., relational model

A strategy for designing a database

- Design using abstract model (conceptual-level modelling)
- Map to physical model (implementation-level modelling)



Some Design Ideas

Consider the following while we work through exercises:

- start simple ... evolve design as problem better understood
- identify objects (and their properties), then relationships
- most designs involve kinds (classes) of people
- keywords in requirements suggest data/relationships (rule-of-thumb: nouns → data, verbs → relationships)
- don't confuse operations with relationships (operation: he **buys** a book; relationship: the book **is owned** by him)
- consider all possible data, not just what's available

Quality of Designs

- There is no single "best" design for a given application
- Most important aspects of a design (data model):
  - correctness (satisfies requirements accurately)
  - completeness (all reqs covered, all assumptions explicit)
  - consistency (no contradictory statements)
- Potential inadequacies in a design:
  - omits information that needs to be included
  - contains redundant information ($\Rightarrow$ inconsistency)
  - leads to an inefficient implementation
  - violates syntactic or semantic rules of data model

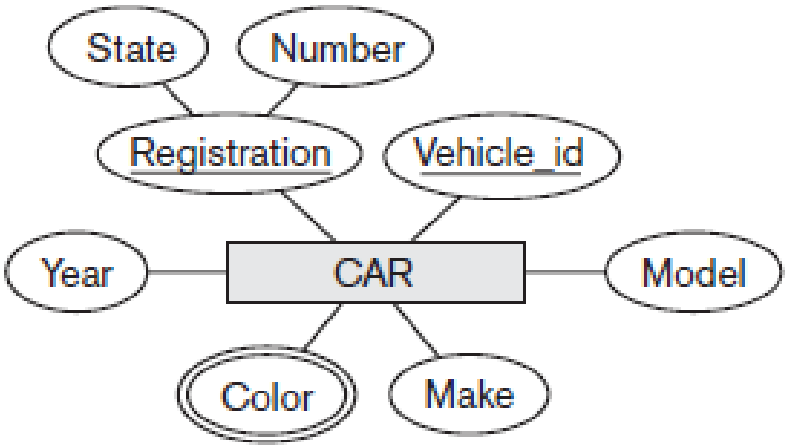
Entity-Relationship Data Modelling

The world is viewed as a collection of inter-related entities.

ER modelling uses three major modelling constructs:

- **entity**:
  - a **thing** or **object** of interest in the real-world and is distinguishable from other objects
- **attribute**:
  - a **data item** or property of interest describing the entity
    e.g., Joe (entity) described by name, address, age (attributes)
- An **entity-set** (aka: entity-type) can be viewed as either:
  - a set of entities with the same set of attributes
  - an abstract description of a class of entities e.g., students, courses, accounts

e.g.,



An entity-set CAR with two key attributes (registration and vehicle_id), three single-valued attributes (year, model, make) and a multi-valued attribute (color)

CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

An entity set CAR with three entities

Relationship Sets

Relationship: relates two or more entities

- e.g Joe Smith (entity) *is enrolled in* (relationship) COMP1531 (entity)

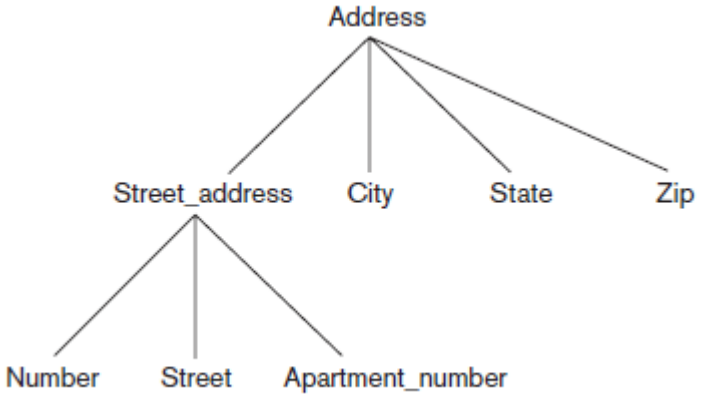
Relationship Set (aka relationship type) : set of similar relationships

- **degree** = # entities involved in the relationship (in ER model, ≥ *2*)
- **cardinality** = # associated entities on each side of relationship

Attributes of an entity-set

In contrast to relational model, attributes in an ER model can be:

- Simple (attribute cannot be broken into smaller sub-parts)
  - e.g., `age` attribute for entity type Employee
- Composite (have a hierarchy of attributes)
  - e.g., entity type EMPLOYEE has a composite attribute `Address`



- Single-valued (have only one value for each entity)
  - e.g., an `vin_chassis` attribute for an entity type CAR
- Multi-valued (have a set of values for each entity)
  - e.g., a `Colors` attribute for CAR = (blue,black)

What if two entities have the same set of attribute values?

- They're regarded as the same entity.
- So, each entity must have a distinct set of attribute values.

  One approach:

  Define a key (superkey) : It is any set of attributes
  - whose set of values are distinct over entity set
  - natural (e.g. name + address + birthday) or artificial (e.g. SSN)
- Candidate key = any superkey such that **no subset** is also a superkey)

  e.g. (name + address) is a superkey, but not (name) or (address)
- Primary key = a candidate key chose by DB designer that uniquely identifies an entity e.g., SSN

Example (bank customer entities)

Customer = (custNo, name, address, taxFileNo)

- Definite superkeys:
  - any set of attributes involving custNo or taxFileNo
- Possible superkeys:
  - (name,address)
- Unlikely superkeys:
  - (name), (address)

ER model vs OO model

Analogy between ER and OO models:

- an entity is like an object instance
- an entity set is like a class

Differences between ER and OO models:

- ER modelling doesn't consider operations (methods)

Entity Relationship Diagrams

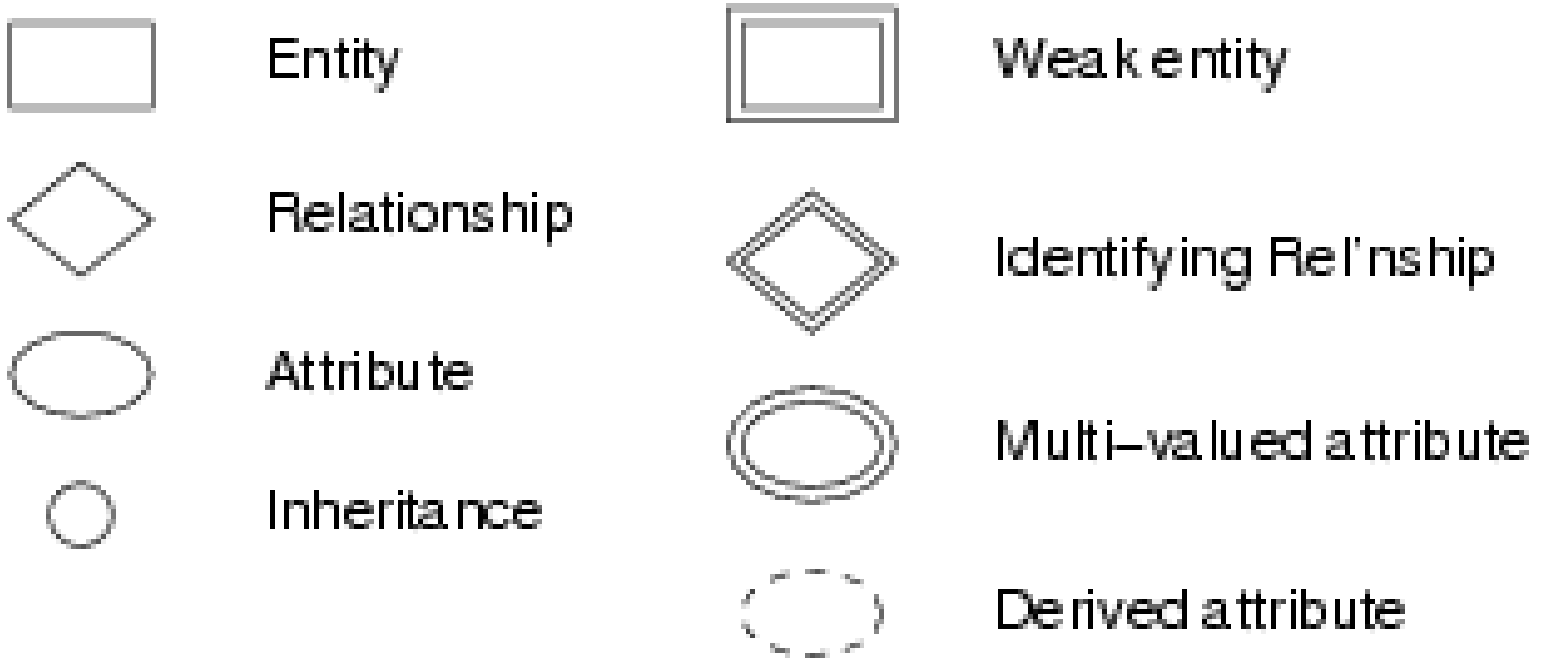
- ER diagrams are a graphical tool for data modelling
- An ER diagram consists of:
  - a collection of *entity set* definitions
  - a collection of *relationship set* definitions
  - *attributes* associated with entity and relationship sets
  - connections between entity and relationship sets

Warning: 99% of the time ...

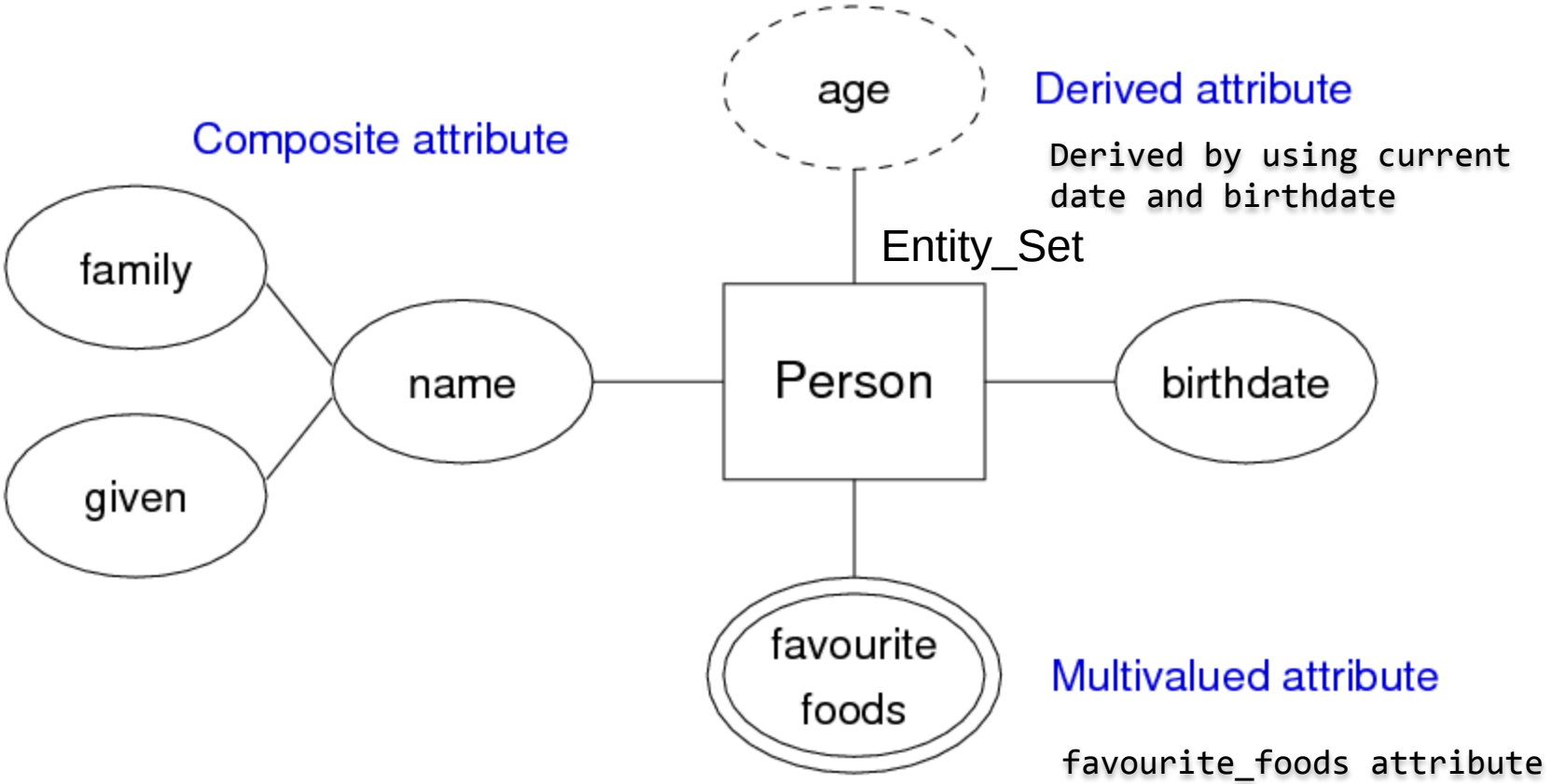
- we say "entity" when we mean "entity set"
- we say "relationship" when we mean "relationship set"
- If we want to refer to a specific entity, we generally say "entity instance"

Entity Relationship Diagrams

Specific visual symbols indicate different ER design elements:



Example of attribute notations

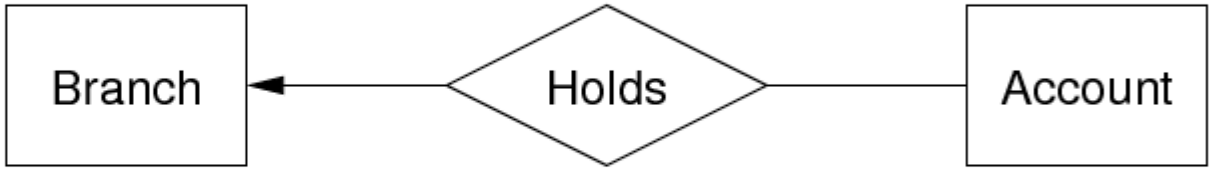


Cardinality in Relationship Sets

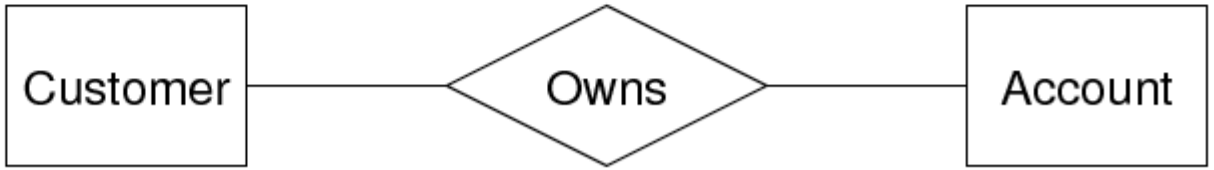
Examples:

one–to–one

one–to–many

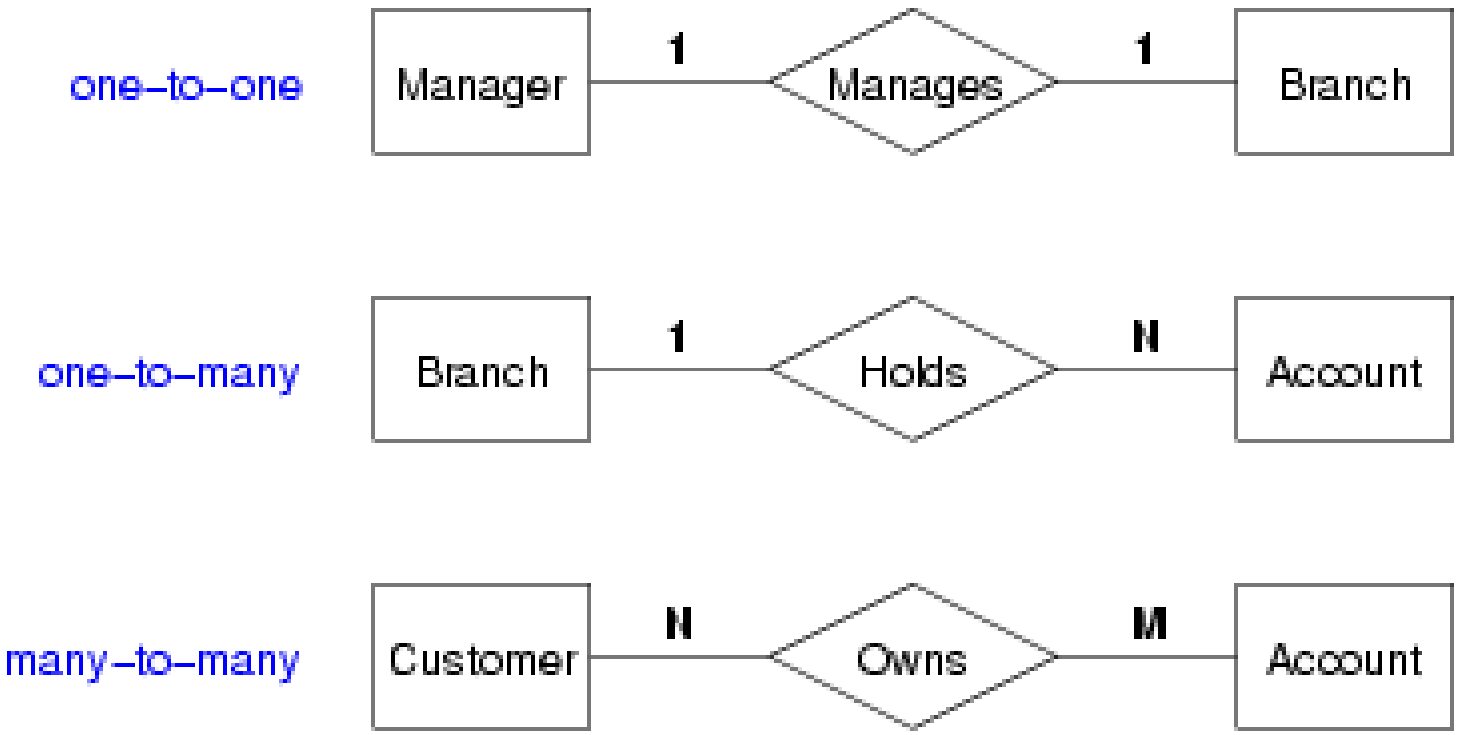


many–to–many



An alternative explicit notation

Examples:



Relationship Sets in ER diagrams

Level of participation constraint = a type of relationship constraint defined as:

Participation in relationship set *R* by entity set A may be:

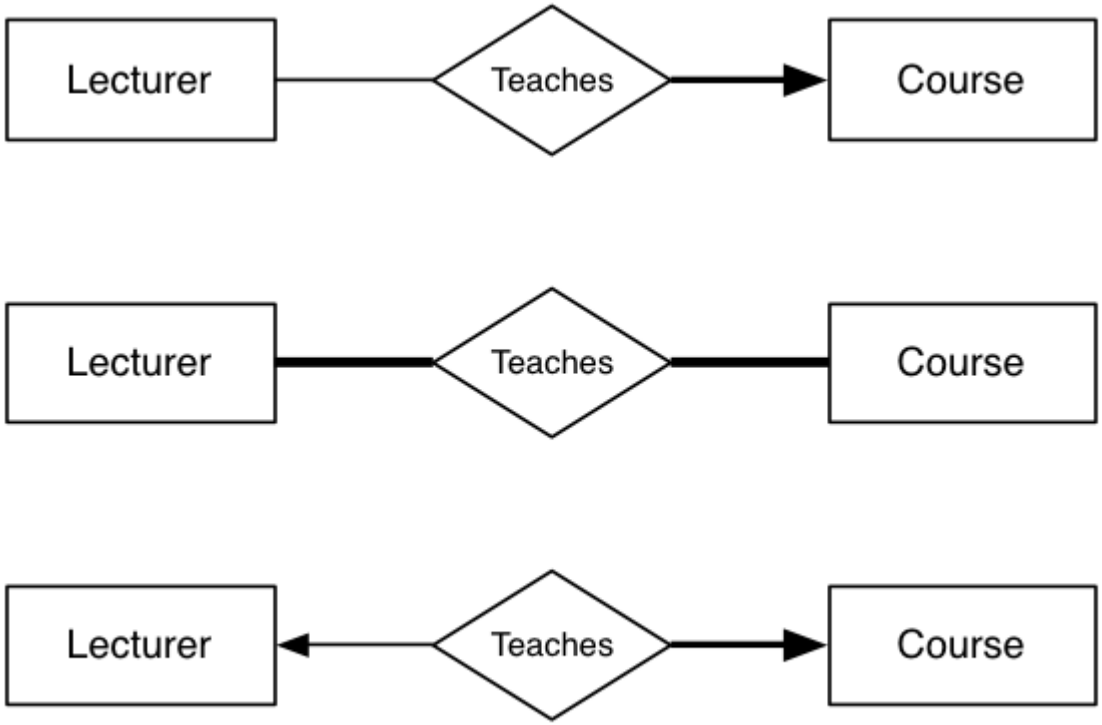
- total - every $a \in A$ participates in ≥ 1 relationship in *R*
- partial - only some $a \in A$ participate in relationships in *R*

Example:

- every bank loan is associated with at least one customer
- not every customer in a bank has a loan

Exercise 1: Relationship Semantics

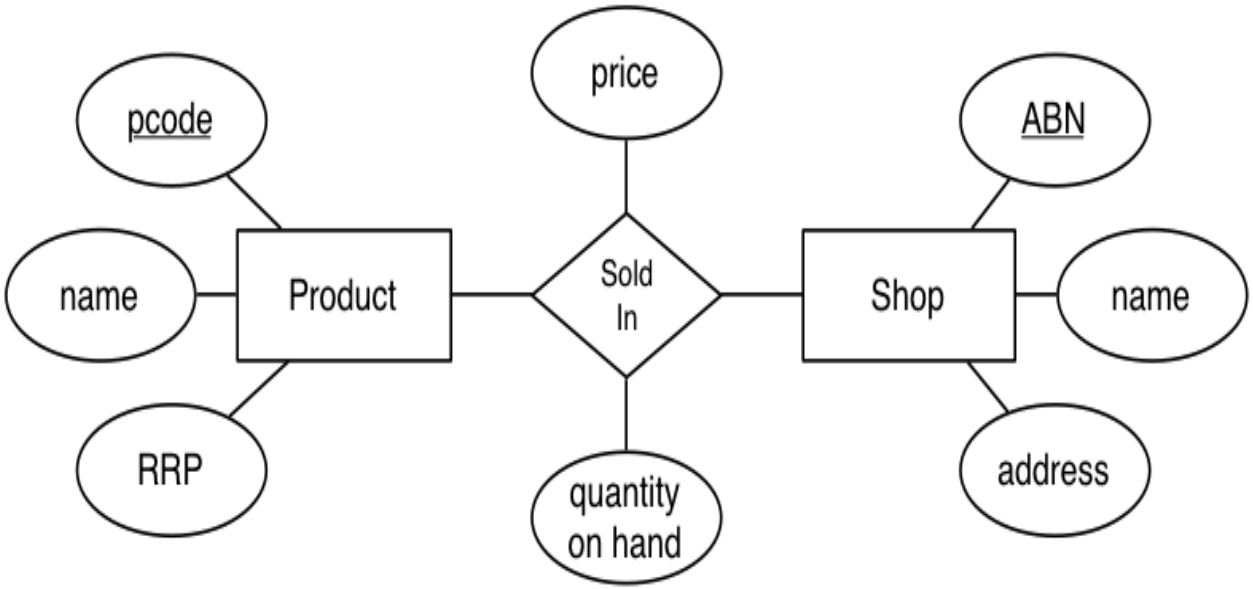
Describe precisely the scenarios implied by the following relationships:



Relationship Type with attributes

In some cases, a relationship needs associated attributes

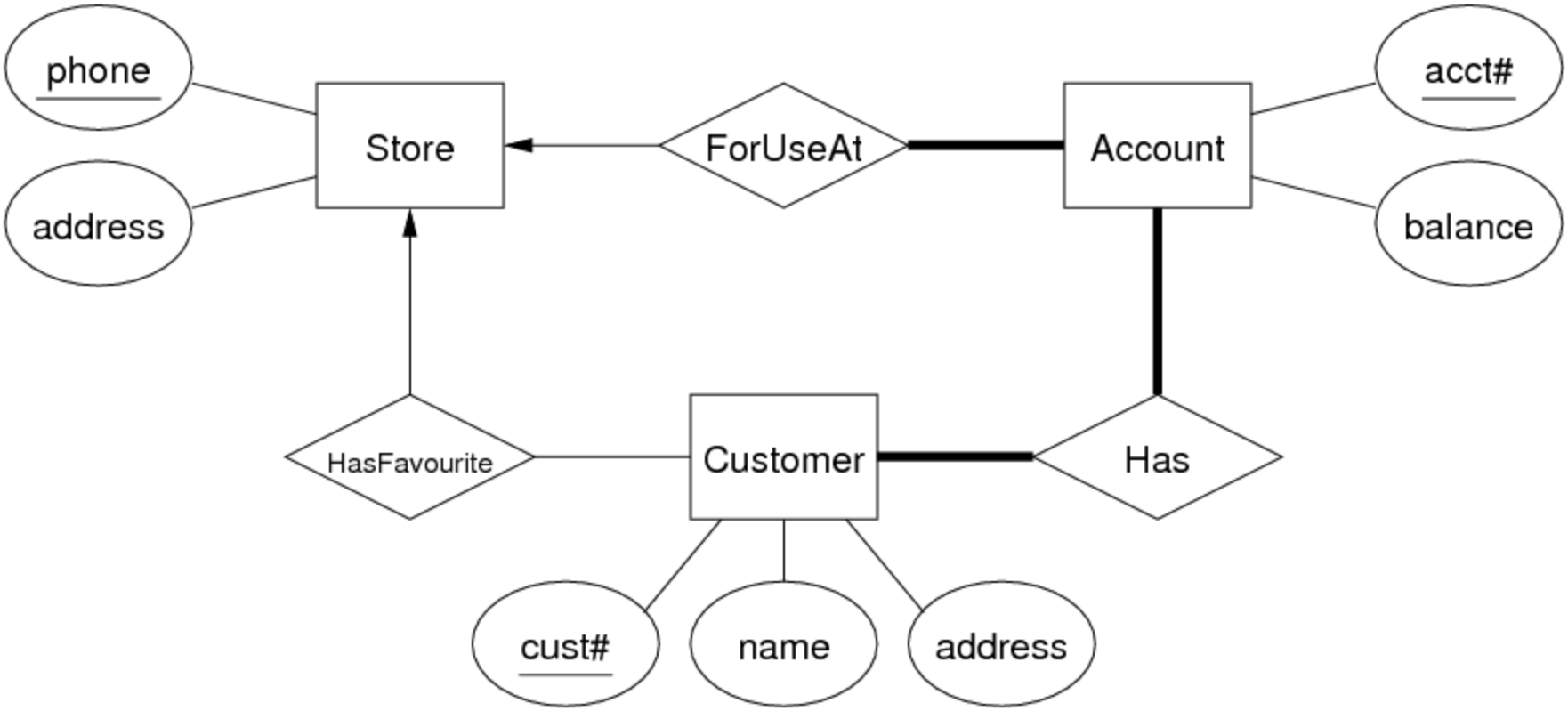
Example:



(price and quantity are related to products in a particular shop)

Putting it all together…

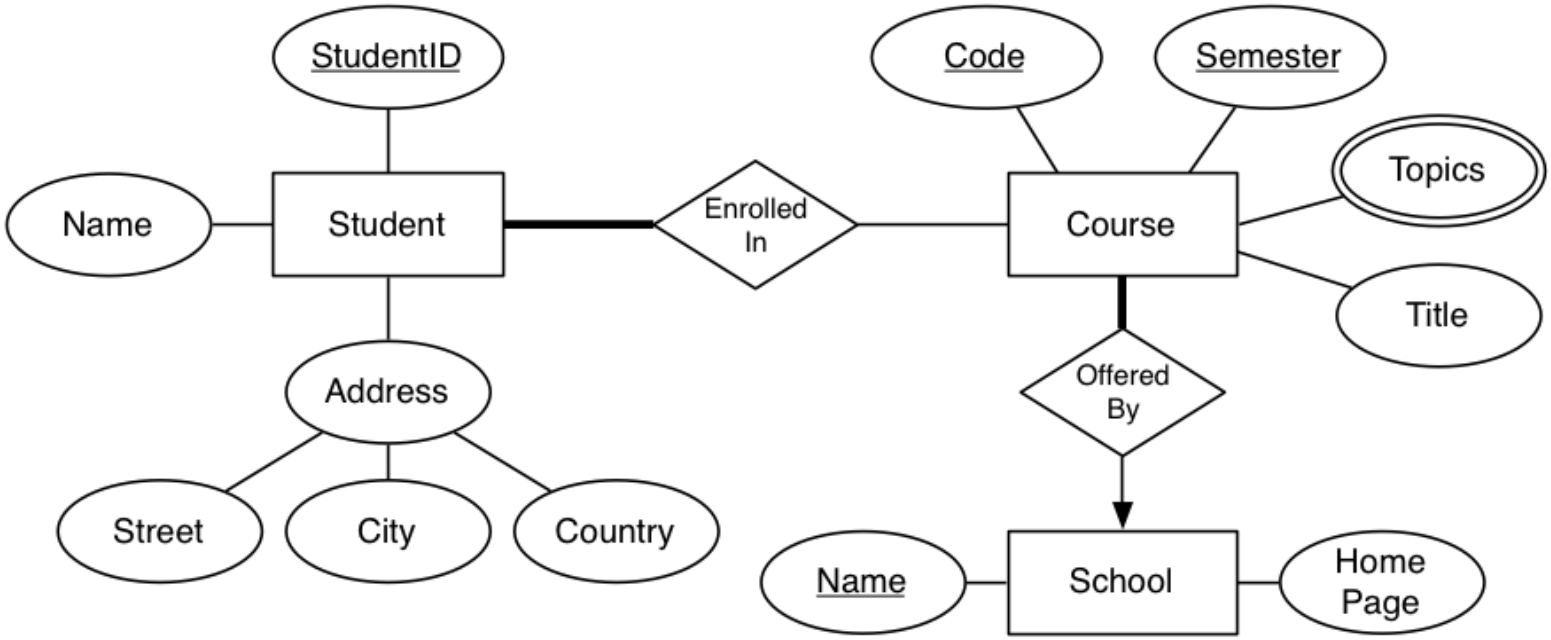
Example1: - a complete ER Diagram



primary key attributes are underlined e.g. cust#

Example 2:

Entities, relationships, attributes, keys, cardinality, participation, ...

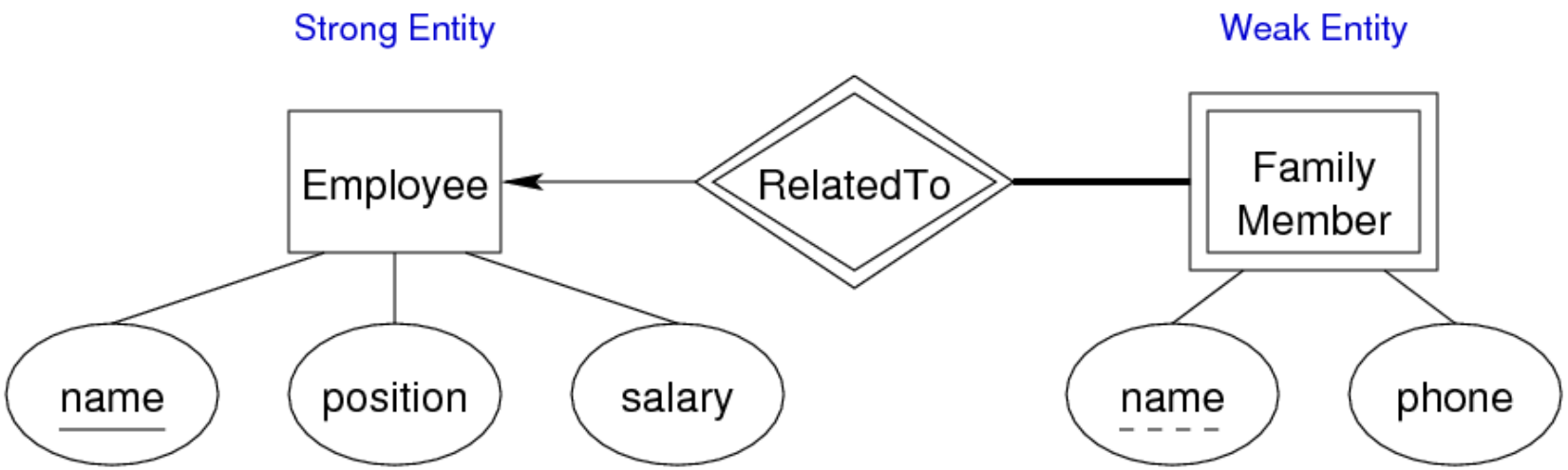


Weak Entity Set

A Weak entity set

- has no key of its own;
- exist only because of association with strong entities

Example:



Subclasses and Inheritance

A subclass of an entity set *A* is a set of entities:

- with all attributes of *A*, plus (usually) it own attributes
- that is involved in all of *A*'s relationships, plus its own

Properties of subclasses:

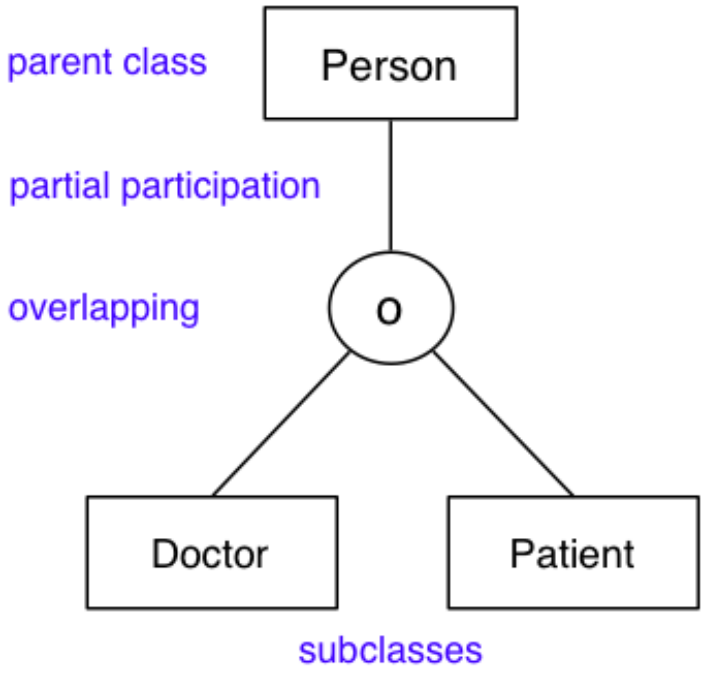
- overlapping or disjoint (can an entity be in multiple subclasses?)
- total or partial (does every entity have to also be in a subclass?)

Special case: entity has one subclass ("B is-a A" specialisation)

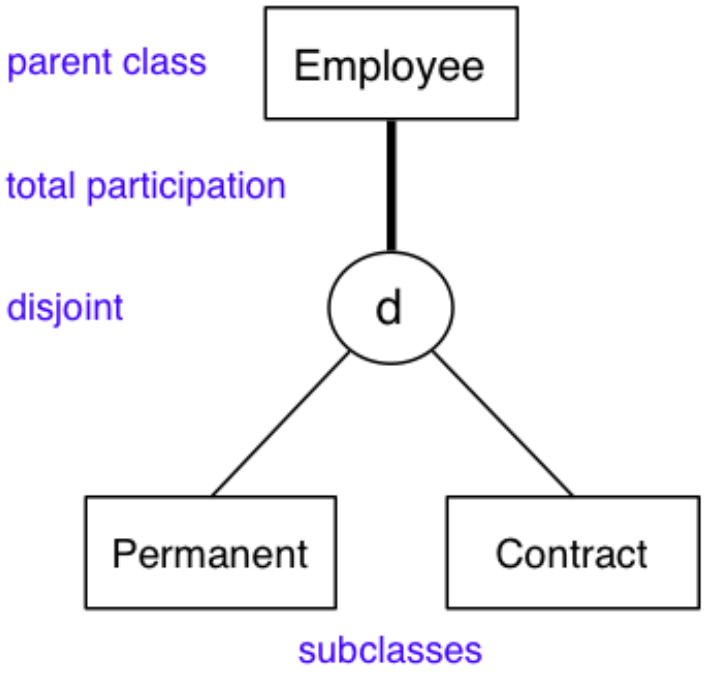
Subclasses and Inheritance

Example:

A person may be a doctor and/or may be a patient or may be neither



Every employee is either a permanent employee or works under a contract



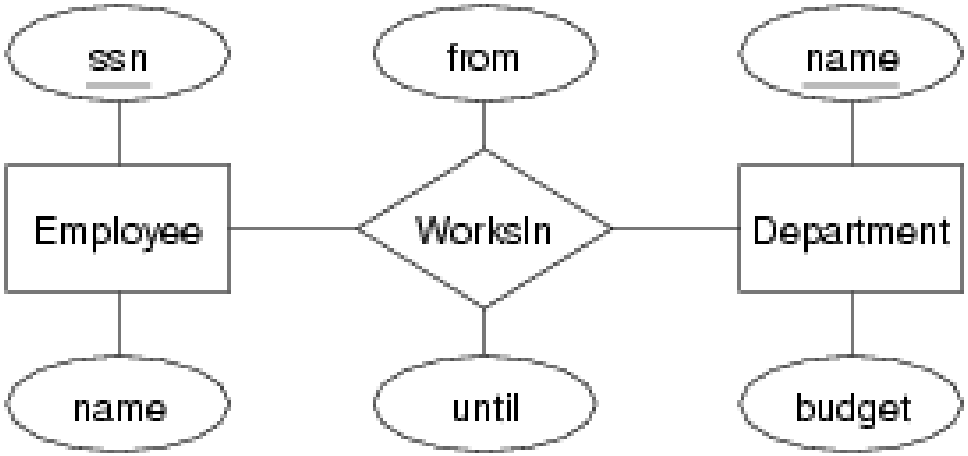
Design considerations using the ER model

- should an "object" be represented by an attribute or entity?
- is a "concept" best expressed as an entity or relationship?
- should we use *n*-way rel^{n}ship or several 2-way rel^{n}ships?
- is an "object" a strong or weak entity? (usually strong)
- are there subclasses/superclasses within the entities?

Answers to above are worked out by *thinking* about the application domain.

Design considerations (cont…)

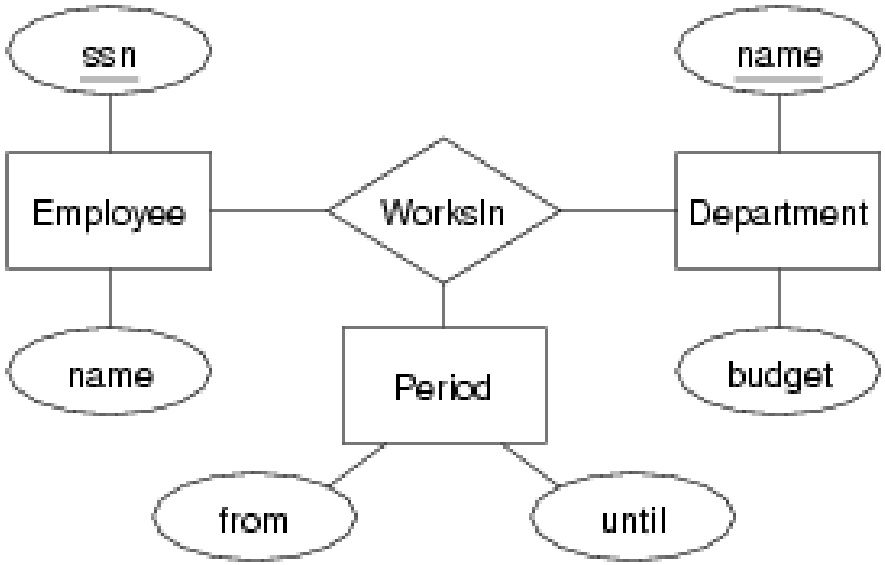
Attribute vs Entity Example (v1)



Employees can work for several departments, but cannot work for the same department over two different time periods.

Design considerations (cont…)

Attribute vs Entity Example (v2)



Employees can work for the same department over two different time periods.

Design using the ER model

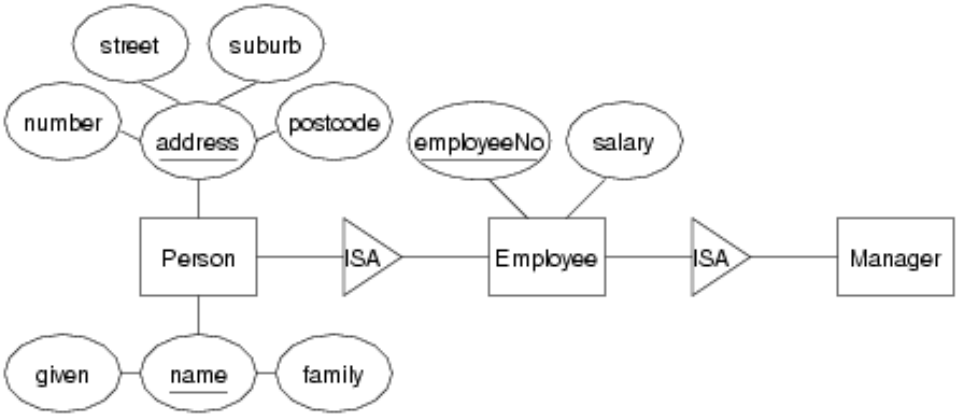
ER diagrams are typically too large to fit on a single screen. (or a single sheet of paper, if printing)

One commonly used strategy:

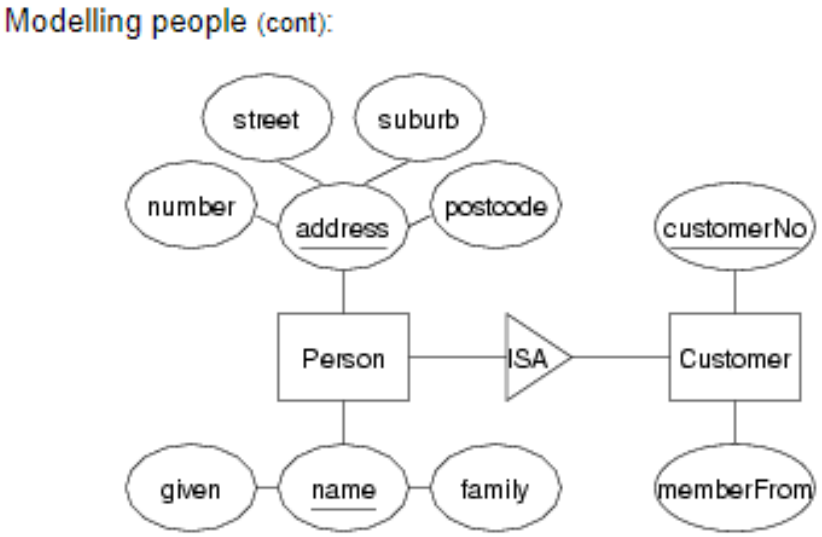
- define entity sets separately, showing attributes
- combine entities and relationships on a single diagram (but without showing entity attributes)
- if very large design, may use several linked diagrams as seen in the example in the next three set of slides

e.g. an ER model for a Bank

(1) Modelling people (employees)

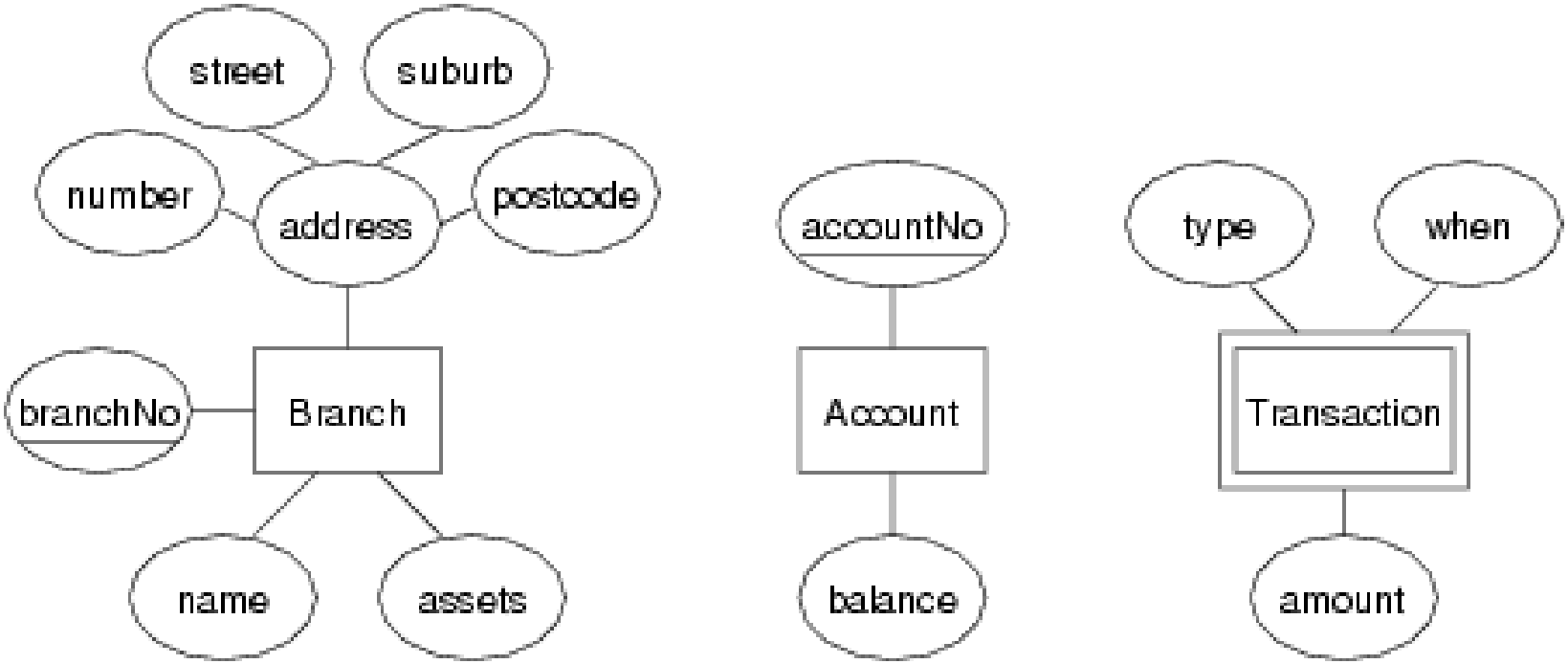


(2) Modelling people (customer)



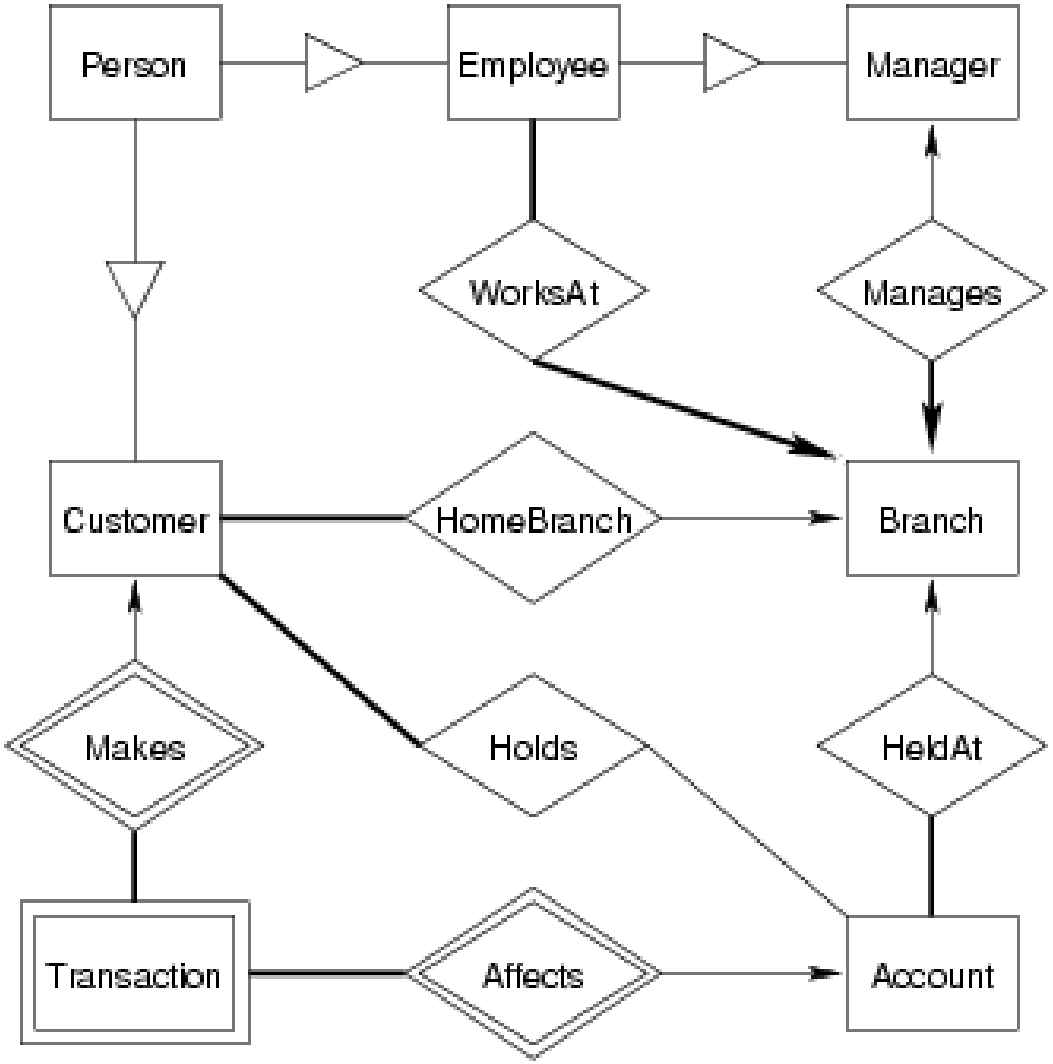
e.g. an ER model for a Bank

(3) Modelling branches, accounts, transactions



e.g. an ER model for a Bank

(4) Putting it all together with relationships



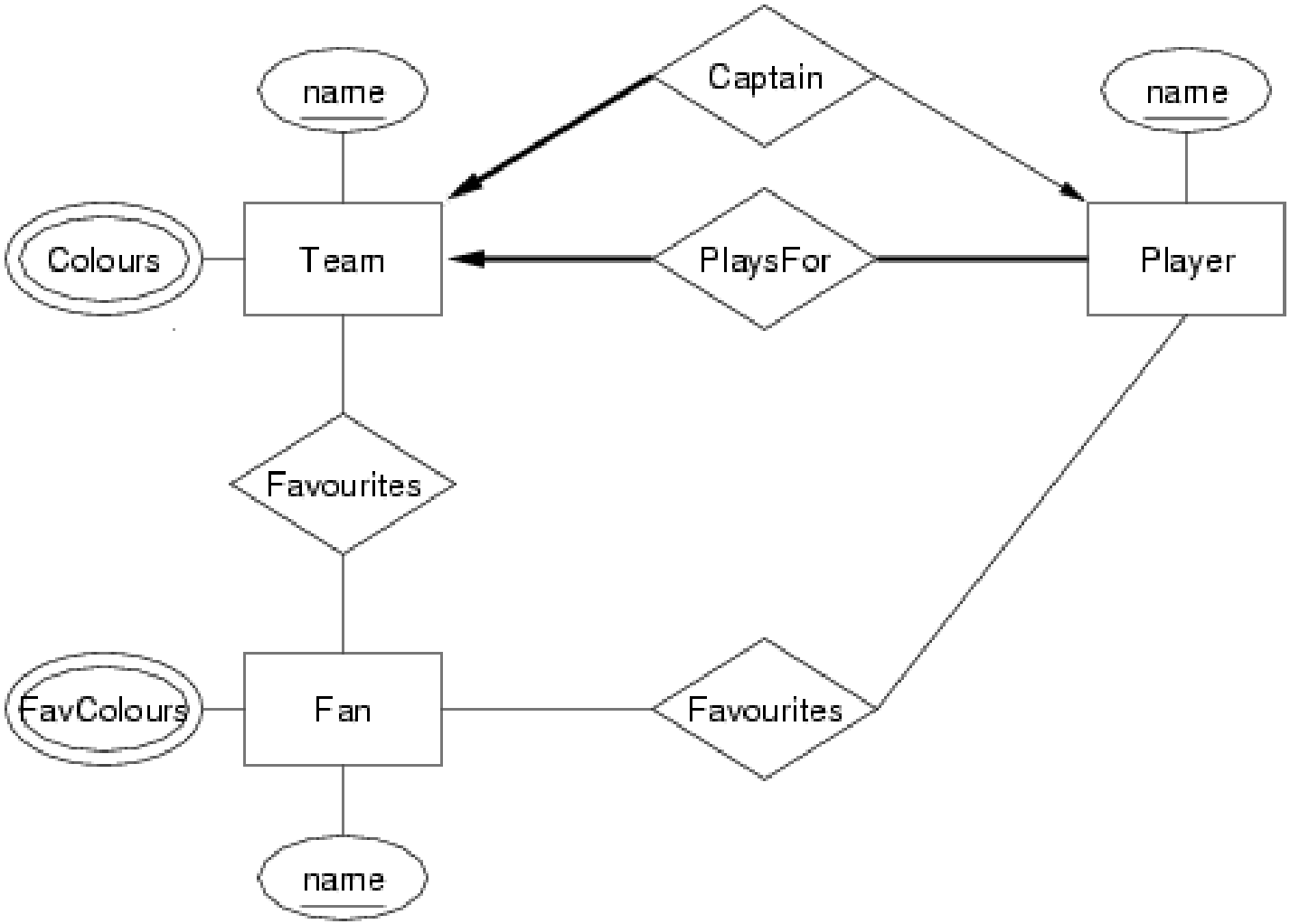
Exercise 2:

Develop an ER design for the following scenario:

A database records information about teams, players, and their fans, including:

- For each team, its name, its players, its captain (one of its players) and the colours of its uniform.
- For each player, their name and team.
- For each fan, their name, favourite teams, favourite players, and favourite colour.

Exercise 2 (Solution): ER Design



Mapping ER Designs to Relational Model

ER to Relational Mapping

- A formal mapping exists for ER model → Relational model.
- This maps "structures"; but additional info is needed, e.g.
  - concrete domains for attributes and other constraints

Relational Model vs Entity Model

Correspondences between relational (R) and ER data models:

- ER attribute → relational attribute
- ER entity → relational tuple
- ER entity-set → relational table (relation)
- ER relationship → relational table (relation)
- ER key → relational primary key

Differences between relational and ER models:

- Relational uses *relations* to model *entities* and *relationships*
- Relational has **no** *composite* or *multi-valued* attributes (only atomic)
- Relational has **no** *object-oriented notions* (e.g. subclasses, inheritance)

(1) Mapping Strong Entities

An *entity* consists of:

- a collection of attributes;
  - attributes are simple, composite and multi-valued

A *relation schema* consists of:

- a collection of attributes;
  - all attributes have atomic data values

So, even the mapping from entity to relation schema is not simple

(1) Mapping Strong Entities

An obvious mapping

- an entity set *E* with atomic attributes $a_1, a_2, \ldots A_n$

maps to

- a relation (table) *R* with attributes (columns) $a_1, a_2, \ldots A_n$

Each row in relation *R* corresponds to an entity in *E*

Example:

(Note: the key is preserved in the mapping)

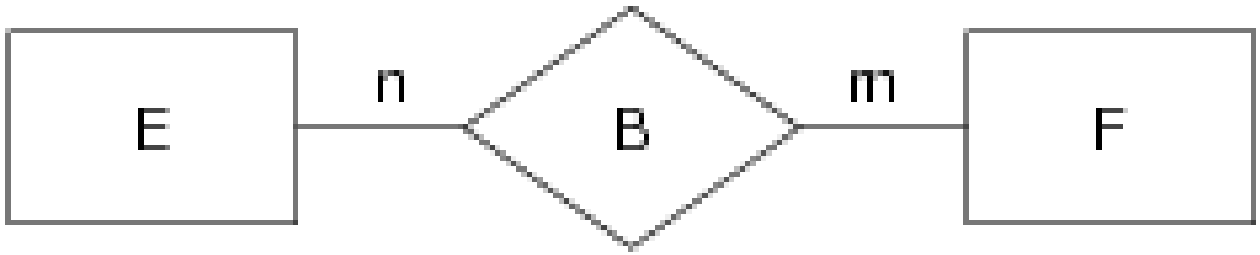
(2) Mapping Relationships

ER **relationship** → relational **table** (relation)

- Identify one entity as “parent”
- other entity as “child”
- as general rule,
  - **PK of parent is added to child as FK**
- Any attributes of the relationship
  - are added to ***child*** relation

(2a) Mapping N:M Relationships

A binary relationship set *B* between entity sets *E* and *F* gives associations between pairs of entities in *E* and *F*



We can represent

- entity set *E* by relation *S* (using attribute mappings as above)
- entity set *F* by relation *T* (using attribute mappings as above)

But how to represent *B*?

One possibility –

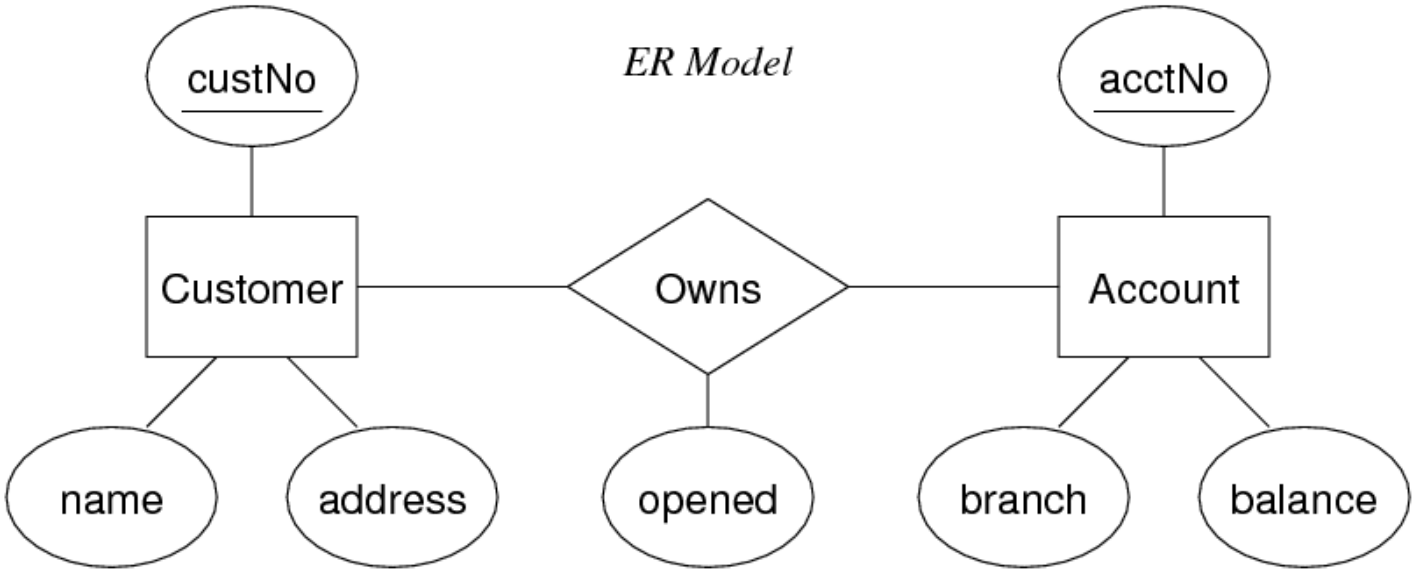
- Represent the relationship set *B* explicitly by a relation *R* containing:
  - all attributes from the primary keys of *S* and *T*
  - all attributes associated with the relationship set *B*

- where *S* and *T* are relations representing entity sets *E* and *F*
- and the *key* for *R* is the union of the key attributes for *S* and *T*

And this approach works generally for:

- relationship degree ≥ 2
- relationship multiplicity 1:1, 1:N, N:M
- associated attributes are simply included in *R*

Example - Mapping N:M Relationship



Relational Version

Customer

custNo	**name**	**address**

Account

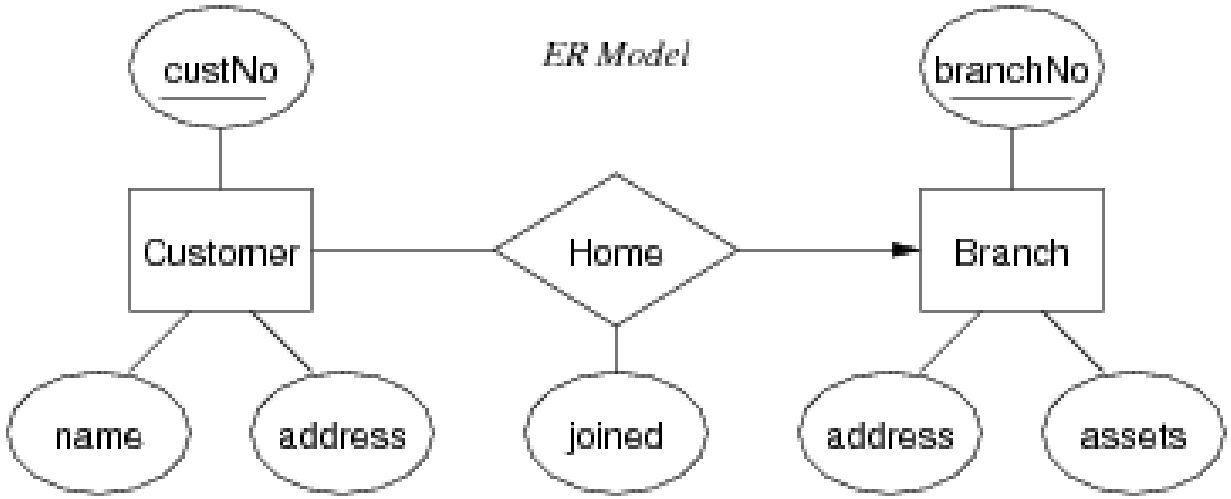
acctNo	**branch**	**balance**

Owns

custNo	**acctNo**	**opened**

(2b) Mapping 1:M Relationships

Example:



Relational Version

Generic Mapping

Customer

custNo	name	address

Branch

branchNo	address	assets

Home

custNo	branchNo	joined

Optimised Mapping

Customer

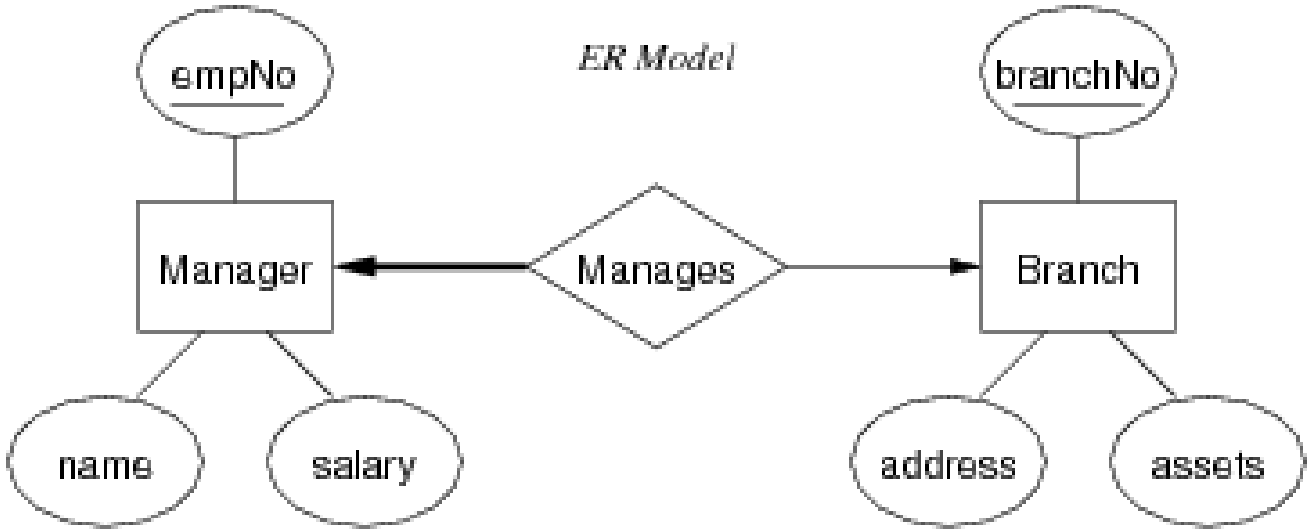
custNo	name	address	branchNo	joined

Branch

branchNo	address	assets

(2c) Mapping 1:1 Relationships

Example:

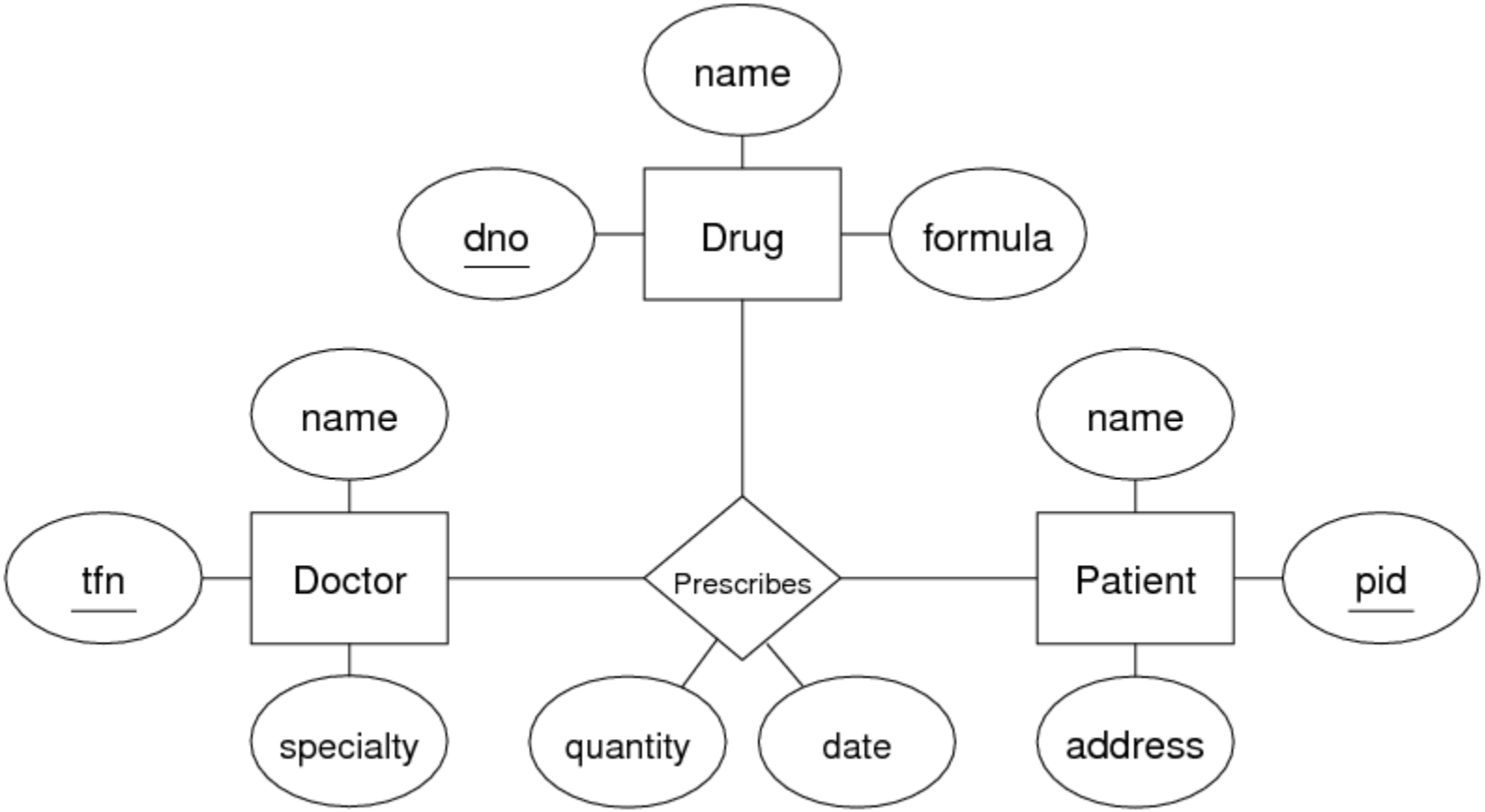


- Handled similarly to 1:N relationships
- For a 1:1 relationship between entity sets *E* and *F* (*S* and *T*):
  - choose one of *S* and *T* (e.g. *S*) (*Note : Choose the entity set that participates totally, if only one of them does*)
  - add the attributes of *T*'s primary key to *S* as foreign key
  - add the relationship attributes as attributes of *S*

(2d) Mapping n-way relationships

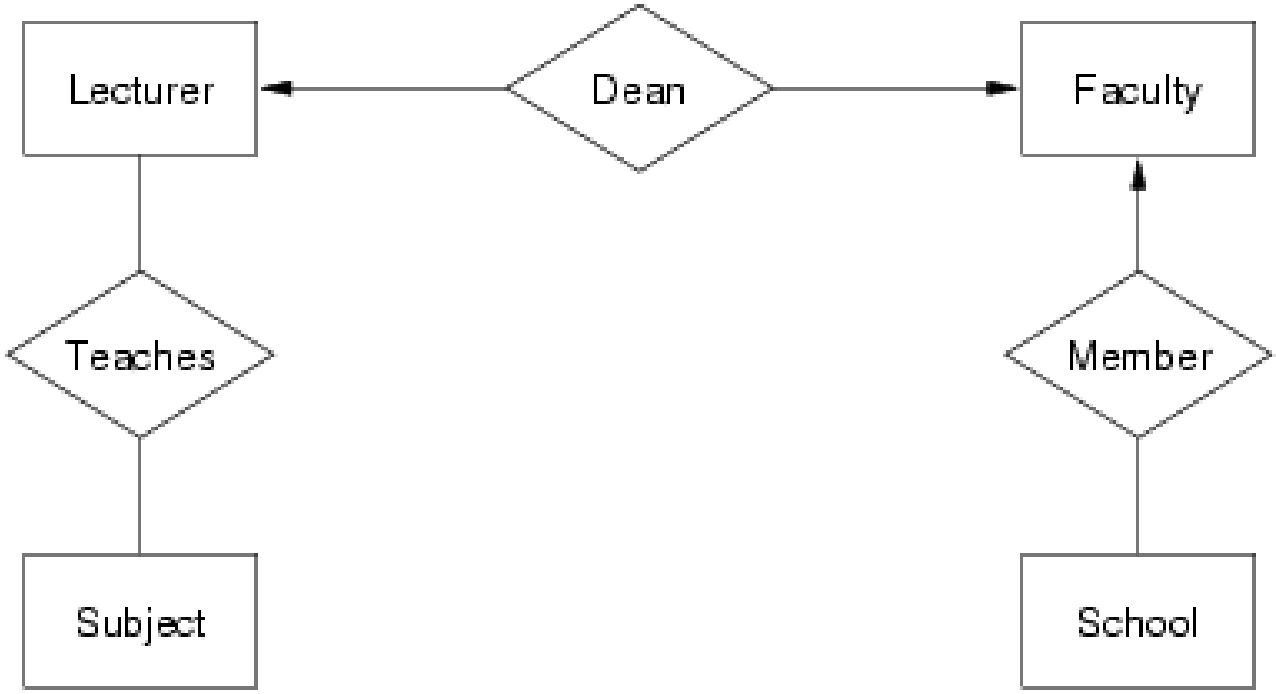
Exercise:

Convert the following ER design into a relational data model



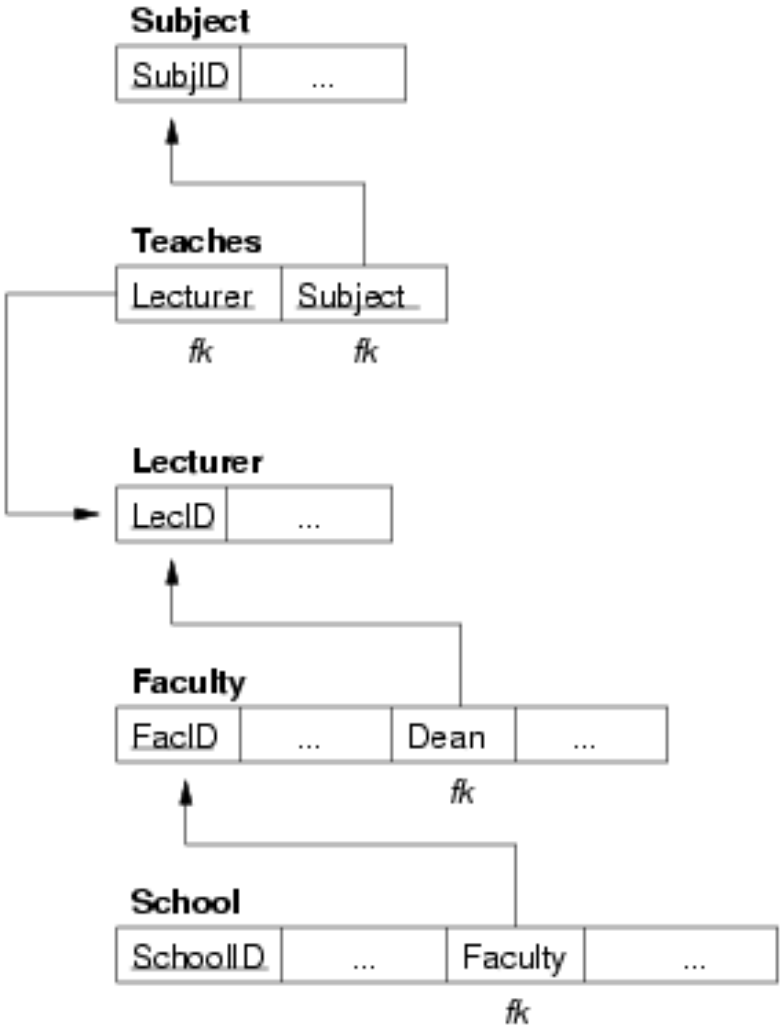
Exercise3:

Convert the following ER design into a relational data model



You can assume that each attribute contains (at least) a suitably-named attribute containing a unique identifying number (e.g. the Lecturer entity contains a LecID attribute).

Exercise3(Solution) : Relational Model



Relational model for a very small University ER model

Wednesday

So far …

Summarising ER → Relational Mapping

Mapping entities and attributes

- ER attribute → relational attribute
- ER entity → relational tuple
- ER entity-set → relational table (relation)
- ER key → relational primary key

Mapping Relationships

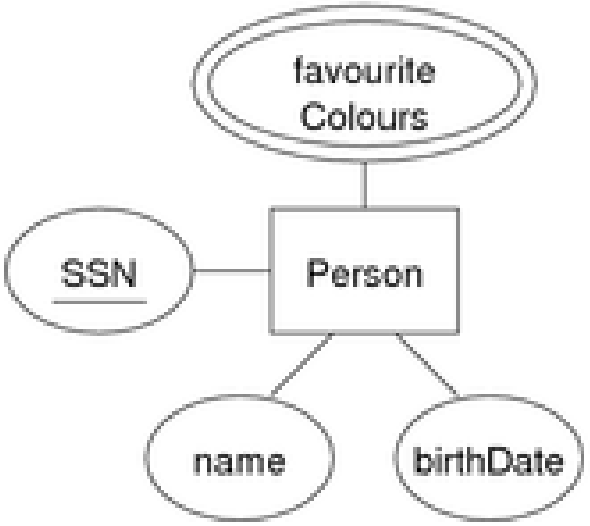
- ER relationship → create a new relational table (relation)
  - N:M relationship → add FK for each participating entity plus relationship attributes)
  - 1:N relationship → FK plus relationship attributes
  - 1:1 relationship → FK plus relationship attributes

(3) Mapping multi-valued attributes

- treat like an N:M relationship between entities and values
- create a new relation where each tuple contains:
  - the primary key attributes from the entity
  - one value for the multi-valued attribute from the corresponding entity

Example:

ER Model



Relational Version

Person

SSN	name	birthdate

FavColour

SSN	colour

(3) Mapping multi-valued attributes contd…

Example: the two entities

```
Person(12345, John, 12-feb-1990, [red,green,blue])
Person(54321, Jane, 25-dec-1990, [green,purple])
```

would be represented as:

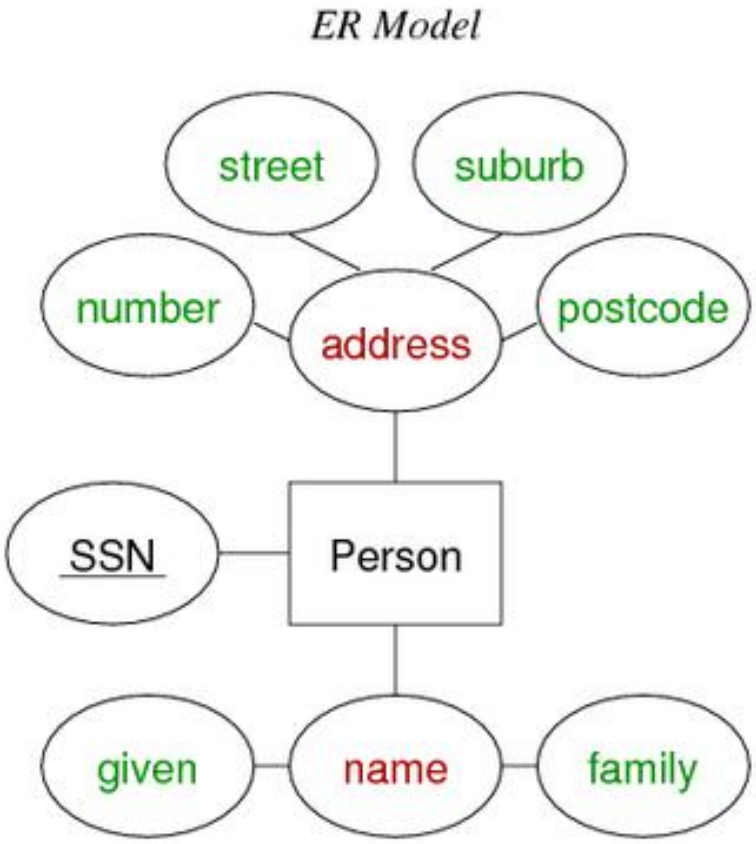
```
Person(12345, John, 12-feb-1990)
Person(54321, Jane, 25-dec-1990)

FavColour (12345, red)
FavColour(12345, green)
FavColour(12345, blue)
FavColour(54321, green)
FavColour(54321, purple)
```

(4) Mapping composite attributes

Mapped by concatenation or flattening

Example:



Relational Version #1

Person

SSN	name	address

Relational Version #2

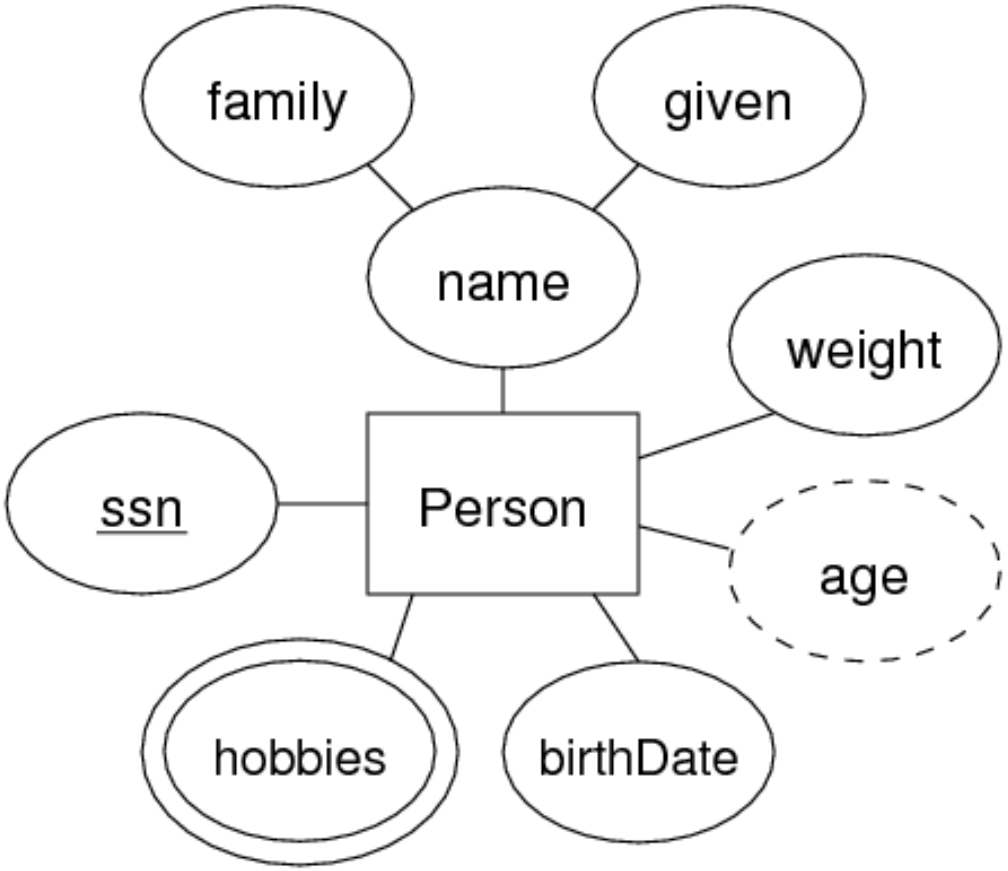
Person

SSN	givenName	familyName	...

... | number | street | suburb | postcode |

Exercise 4:

Convert the following ER design to relational form

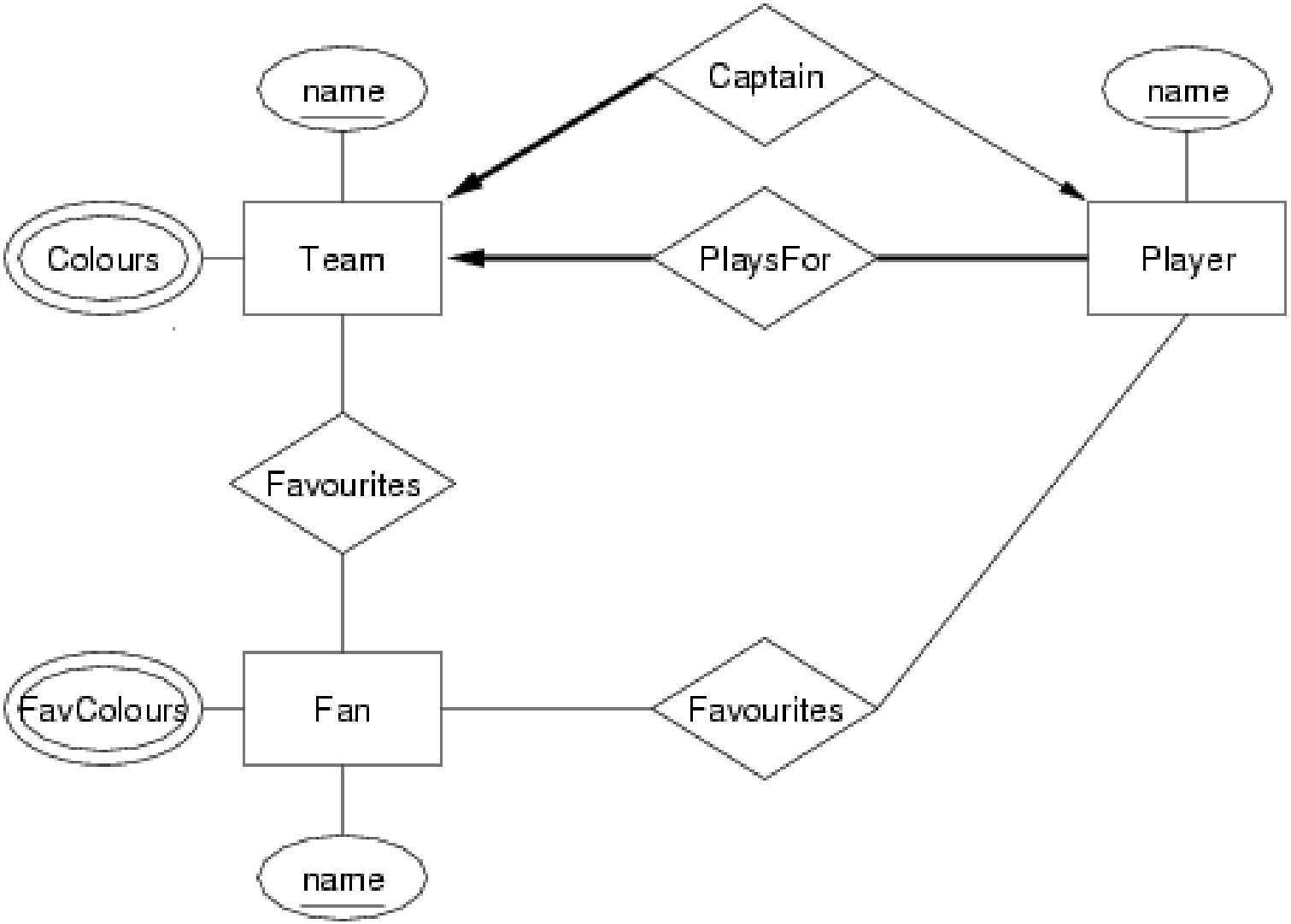


Exercise 5:

Convert the following ER design into a relational data model

(1) first as a box-and-arrow diagram

(2) as a sequence of statements in the SQL data definition language:



(5) Mapping sub classes

Three different approaches to mapping subclasses to tables:

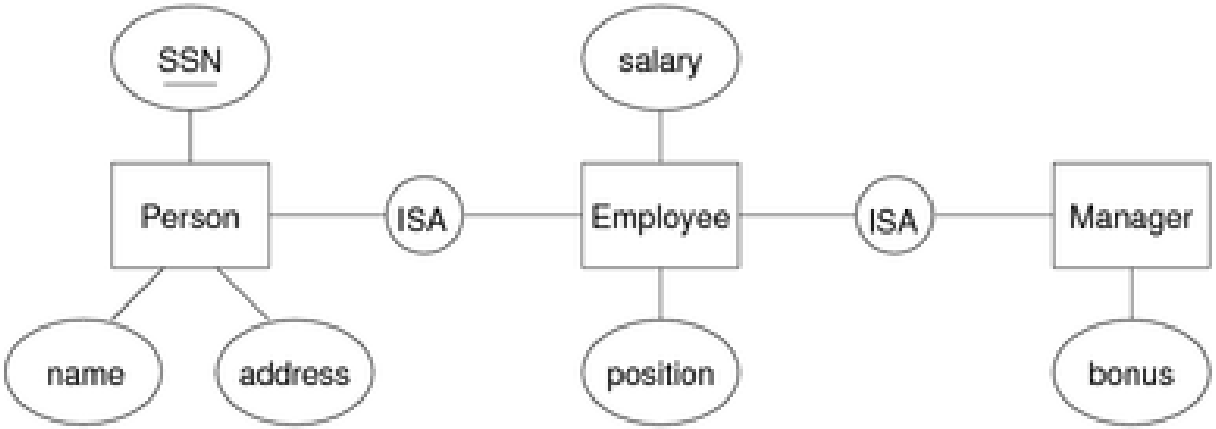
- ER style
  - each entity becomes a separate table,
  - containing attributes of subclass + FK to superclass table
- object-oriented
  - each entity becomes a separate table,
  - inheriting all attributes from all superclasses
- single table with nulls
  - whole class hierarchy becomes one table,
  - containing all attributes of all subclasses (null, if unused)

Which mapping is best depends on how data is to be used

(5) Mapping sub classes in ER style

The subclass relation contains:

- all of the subclass-specific attributes
- uses the superclass primary key to capture the association



Relational Model

Person

SSN	name	address

Employee

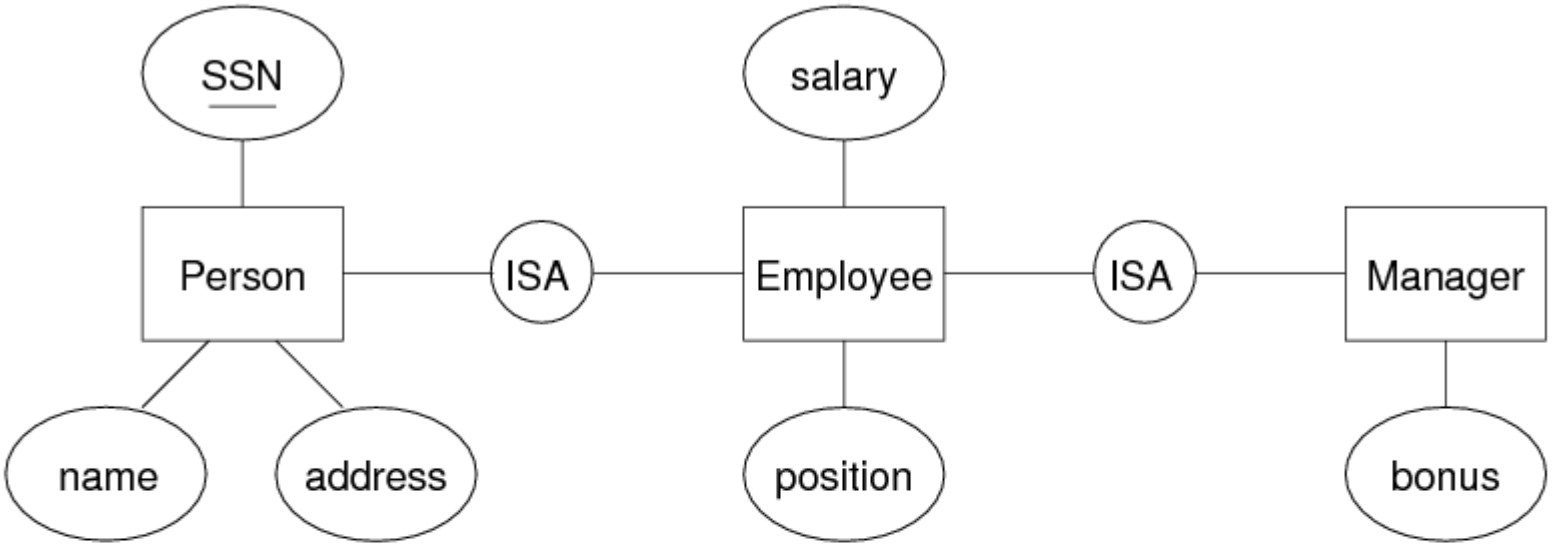
SSN	salary	position

Manager

SSN	bonus

(5) OO Mapping of sub classes

Entity–Relationship Model



Relational Model

Person

SSN	**name**	**address**

Employee

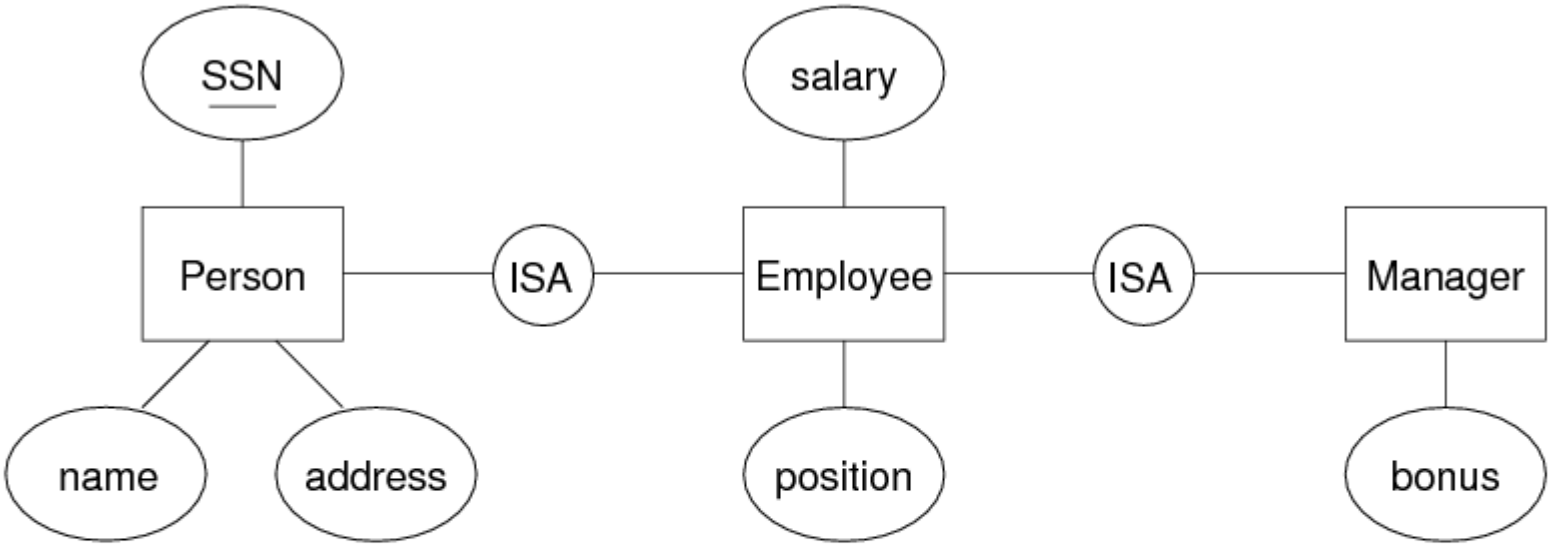
SSN	**name**	**address**	**salary**	**position**

Manager

SSN	**name**	**address**	**salary**	**position**	**bonus**

(5) Single table with nulls mapping

Entity–Relationship Model



Relational Model

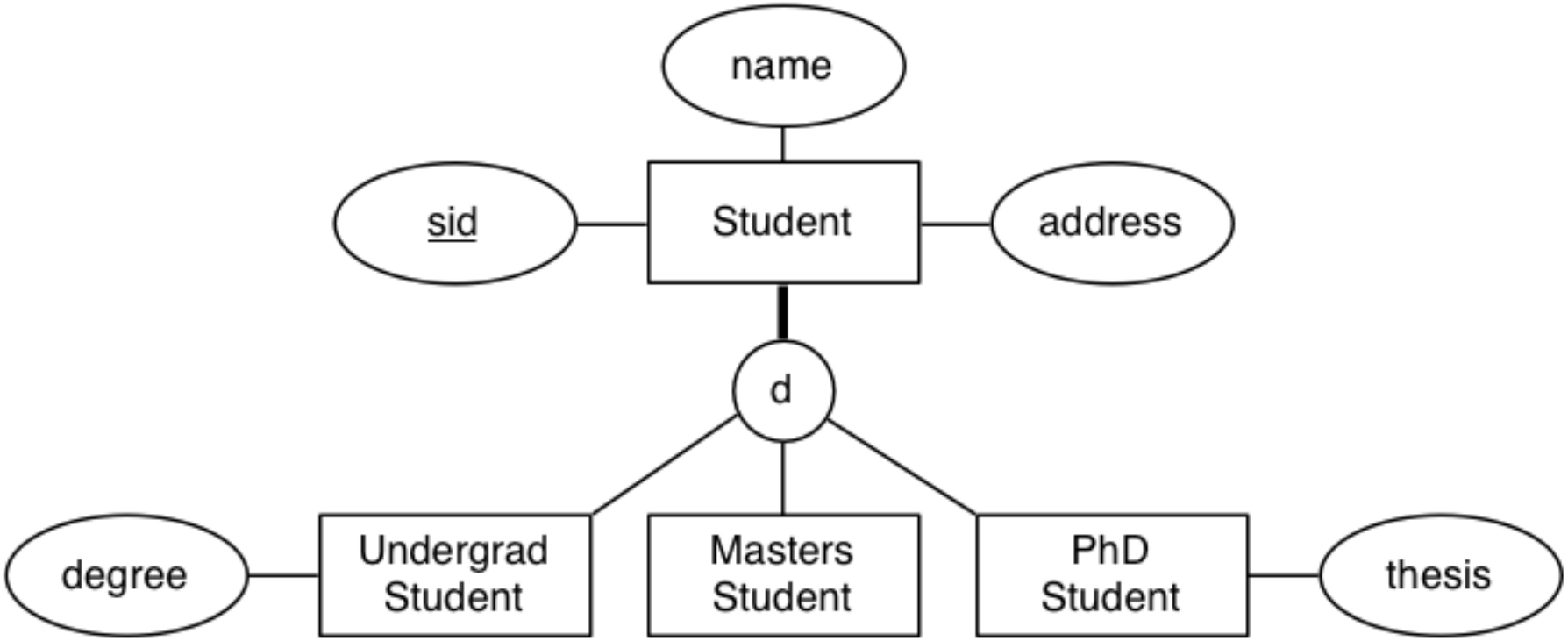
Person

SSN	name	address	salary	position	bonus

NULL for Person who is not Employee

NULL for Employee who is not Manager

(5) Exercise 6: Disjoint subclasses



Use (a) ER-mapping, (b) OO-mapping, (c) 1-table-mapping
Are there aspects of the ER design that can't be mapped?

Exercise 7: Give an ER design to model the following scenario ...

- Patients are identified by an **SSN**, and their **names**, **addresses** and **ages** must be recorded.
- Doctors are identified by an SSN. For each doctor, the name, specialty and years of experience must be recorded.
- Each pharmacy has a name, address and phone number. A pharmacy **must** have a manager.
- A pharmacist is identified by an SSN, he/she can only work for one pharmacy. For each pharmacist, the name, qualification must be recorded.
- For each drug, the trade name and formula must be recorded.
- Every patient has a primary physician. Every doctor has at least one patient.
- Each pharmacy **sells** several drugs, and has a price for each. A drug could be sold at several pharmacies, and the price could vary between pharmacies.
- Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and quantity associated with it.