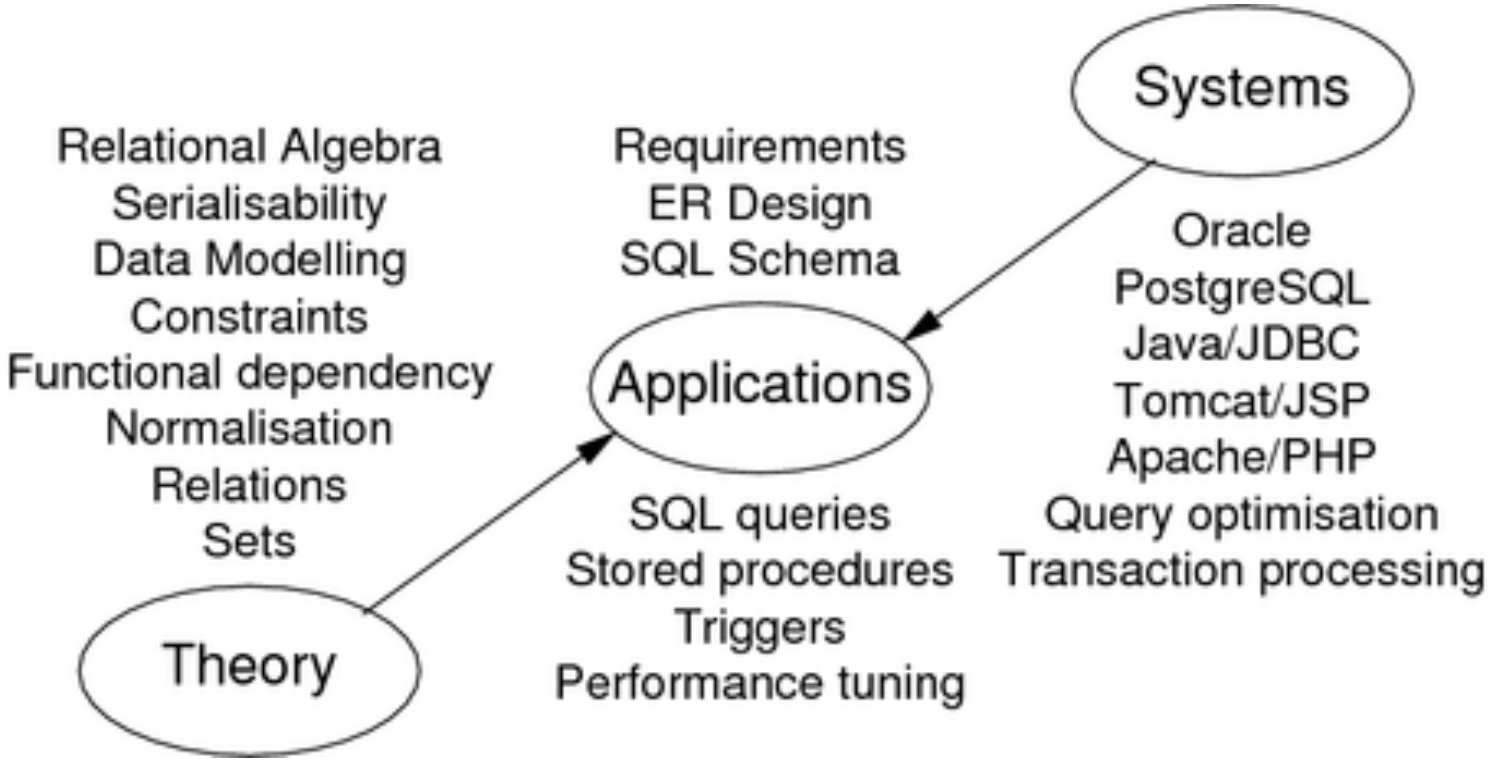
# Data Modelling

textbook:

Chapter 3 Data Modelling using the Entity Relationship (ER) Model

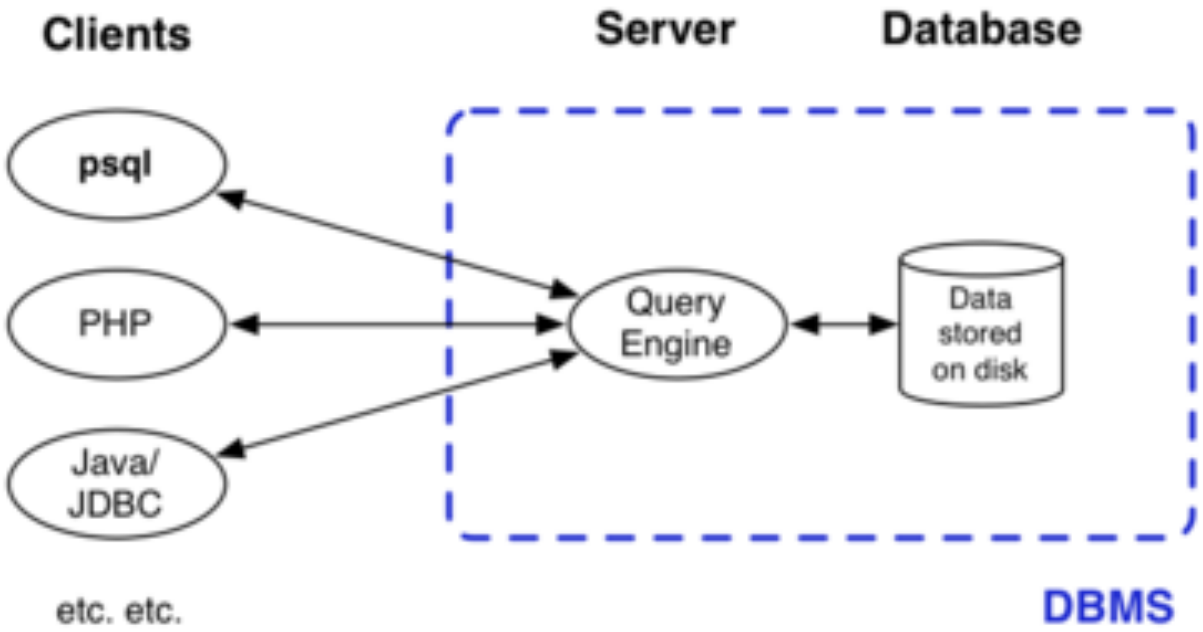
Chapter 4 Extended ER …

# Overview of the Databases Field



# A high-level view of a database application

The typical environment for a modern DBMS is:




SQL queries and results travel along the client↔server links

# Database Application Development

A variation on standard software engineering process:

- analyse application requirements
- develop a data model to meet these requirements
- define operations (transactions) on this model
- implement the data model as relational schema
- implement transactions via SQL and procedural PLs
- construct an interface to these transactions
- At some point, populate the database (may be via interface)

# Data Modelling

Aims of data modelling:

- describe what *information* is contained in the database

  (e.g., entities: students, courses, accounts, branches, patients, ...)

- describe *relationships* between data items

  (e.g., John is enrolled in COMP3311, Andrew's account is held at Coogee)

- describe *constraints* on data

  (e.g., 7-digit IDs, students can enrol in no more than four courses per semester)

Data modelling is a *design* process

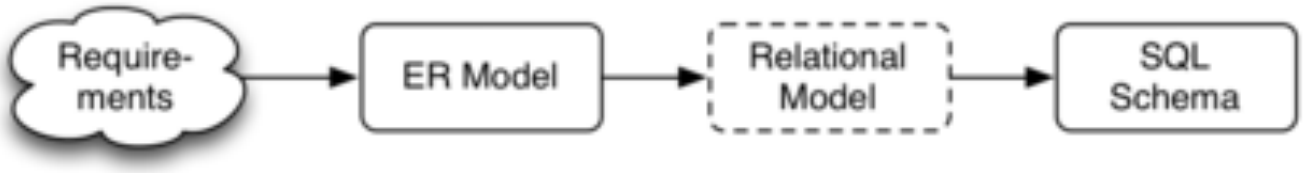
- converts requirements into a data model

# Data Modelling

Kinds of data models:

- *logical*: abstract, for conceptual design, e.g., ER, ODL (object data language)
- *physical*: record-based, for implementation, e.g., relational

Strategy: design using abstract model; map to physical model



# 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 is available

# Exercise 1: Gmail Data Model

Consider the Google Mail System:

More information about Gmail from https://www.howtogeek.com/school/gmail-guide/

Let's develop an informal data model for it by identifying:

- the data items involved (objects and their attributes)
- relationships between these data items
- constraints on the data and relationships

[suggested solution will be posted later]

# 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 has **three** major modelling **constructs**:

- *attribute*: data item describing a property of interest
- *entity*: collection of attributes describing object of interest
- *relationship*: association between entities (objects)

The ER model is not a standard, so many variations exist.

Lecture notes use simple notations. We can use this as 'COMP9311 standard'.

# Entity-Relationship (ER) Diagrams

*ER diagrams* are a graphical tool for data modelling.

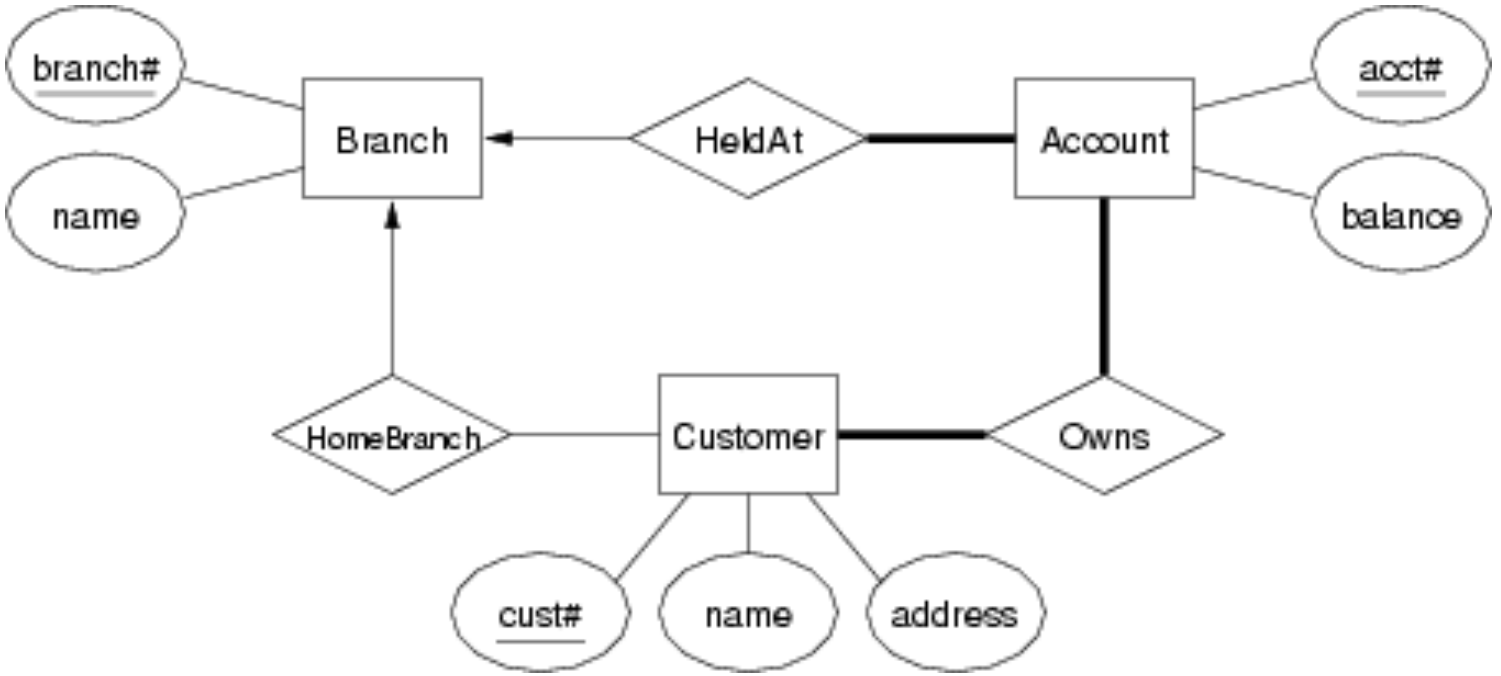
An ER diagram consists of:

- 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

Terminology: when discussing "entity sets", we frequently say just "entity"

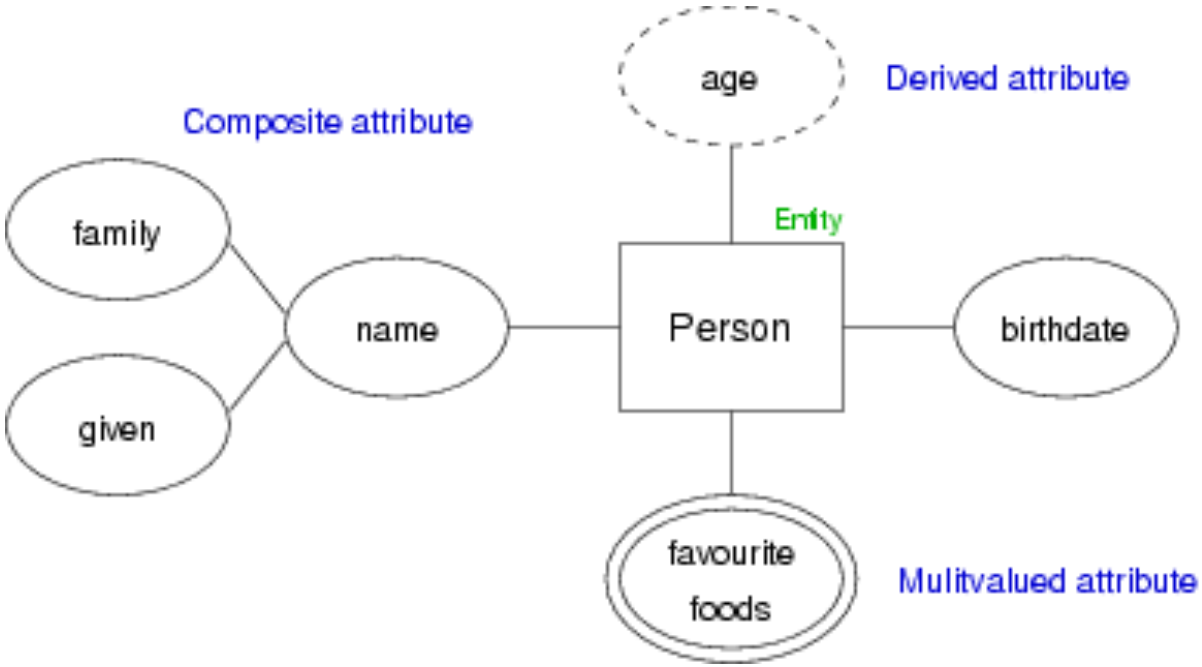
# Entity-Relationships

Example ER Diagram



# Entity-Relationships

Example of attribute notations



# Entity Sets

An *entity set* can be viewed as either:

- a set of entities with the same set of attributes (extensional)
- an abstract description of a class of entities (intensional)

*Key* (*superkey*): 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* = minimal superkey (no subset is a key)

*Primary key* = candidate key chosen by DB designer

Keys are indicated in ER diagrams by underlining

# Relationship Sets

*Relationship*: an association among several entities

- e.g., Customer(9876) is the owner of Account(12345)

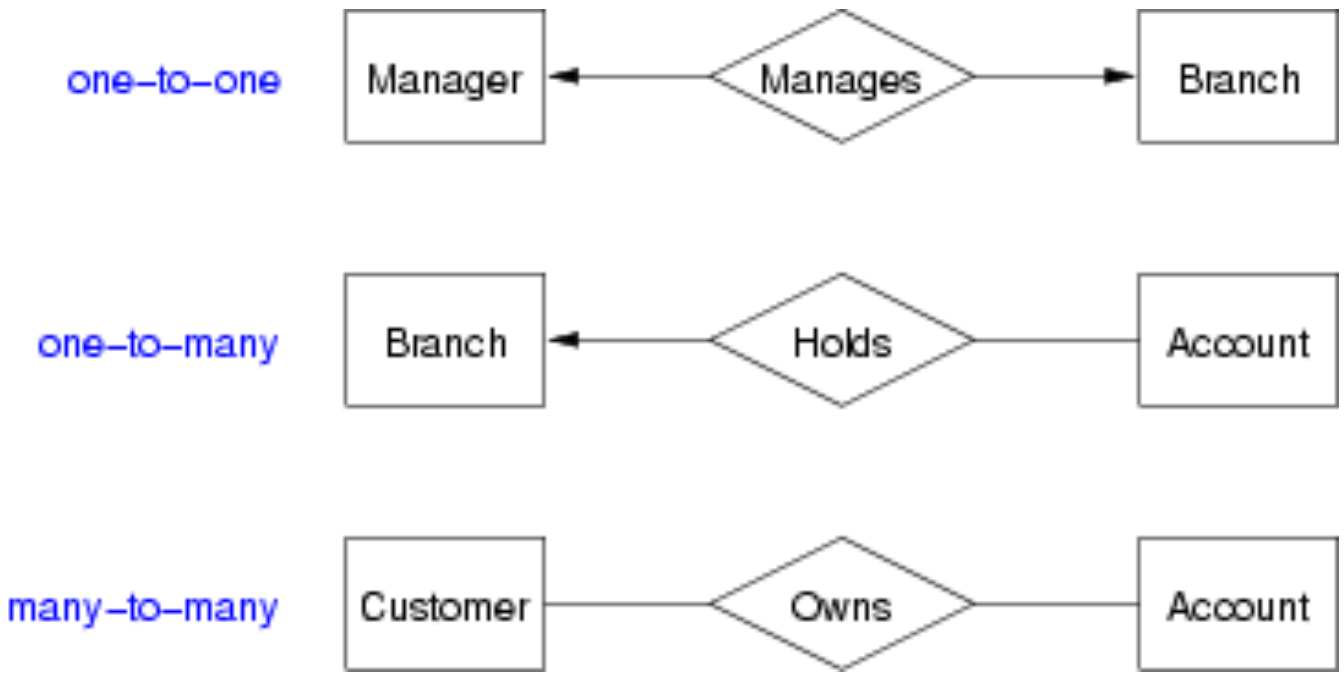
*Relationship set*: collection of relationships of the same type

- *Degree* = # entities involved in reln (in ER model, ≥ *2*)
- *Cardinality* = # associated entities on each side of reln
- *Participation* = must every entity be in the relationship

**Example:** relationship participation

# Relationship Sets

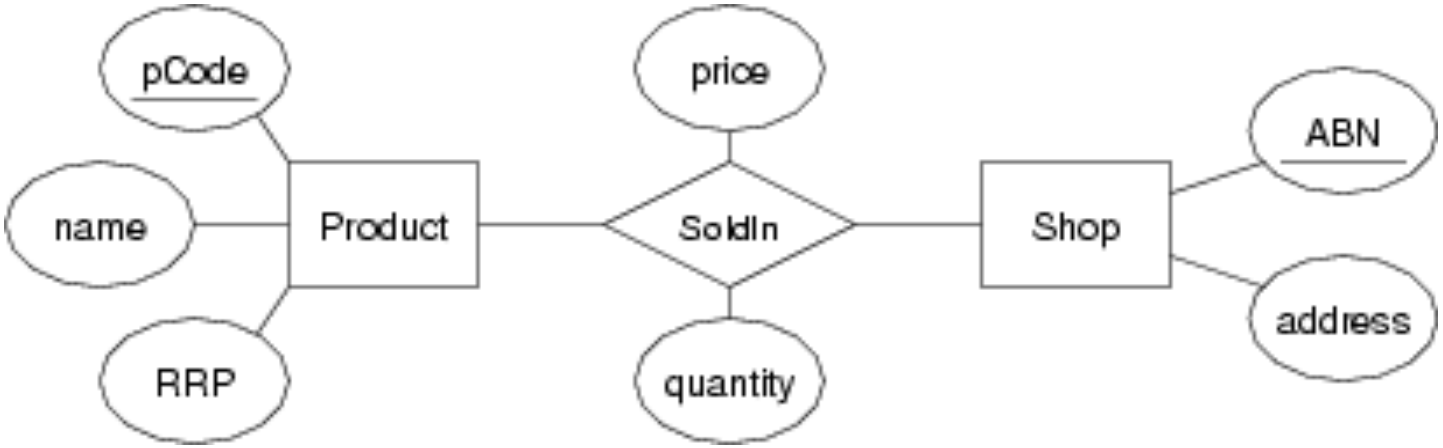
Examples: Relationships



# Relationship Sets

In some cases, a relationship needs associated attributes.

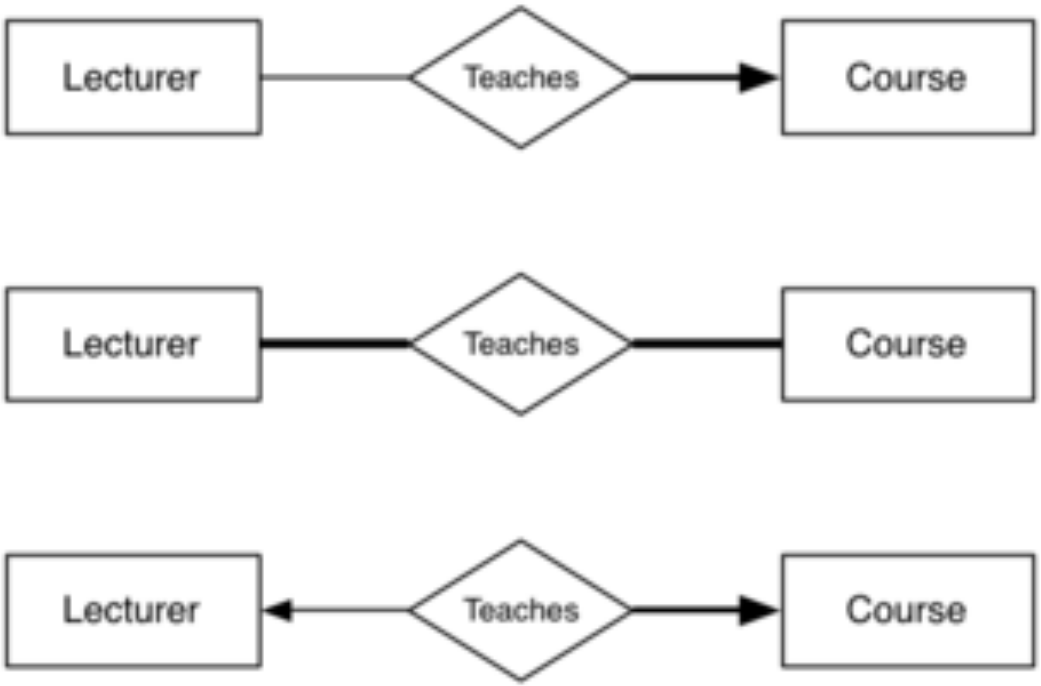
Example:




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

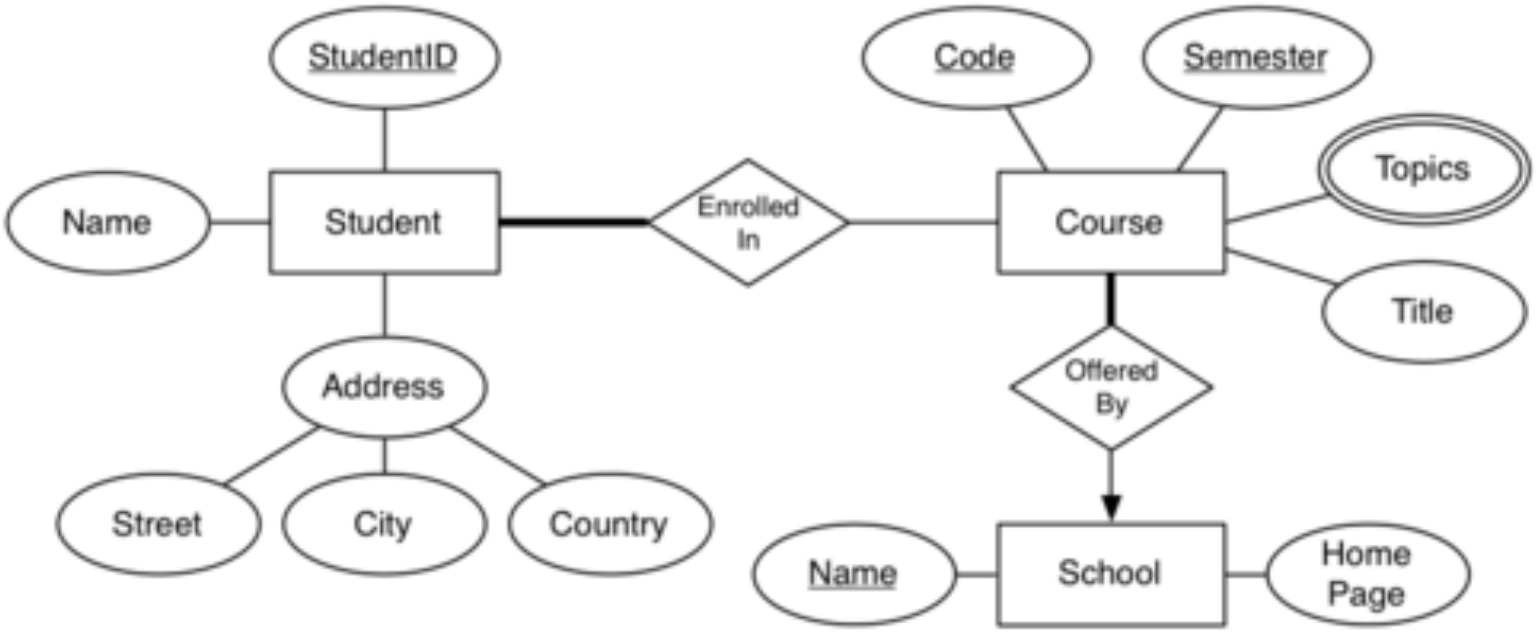
# Exercise 2: Relationship Semantics

Describe precisely the scenarios implied by the following relationships:



# ER: the story so far

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

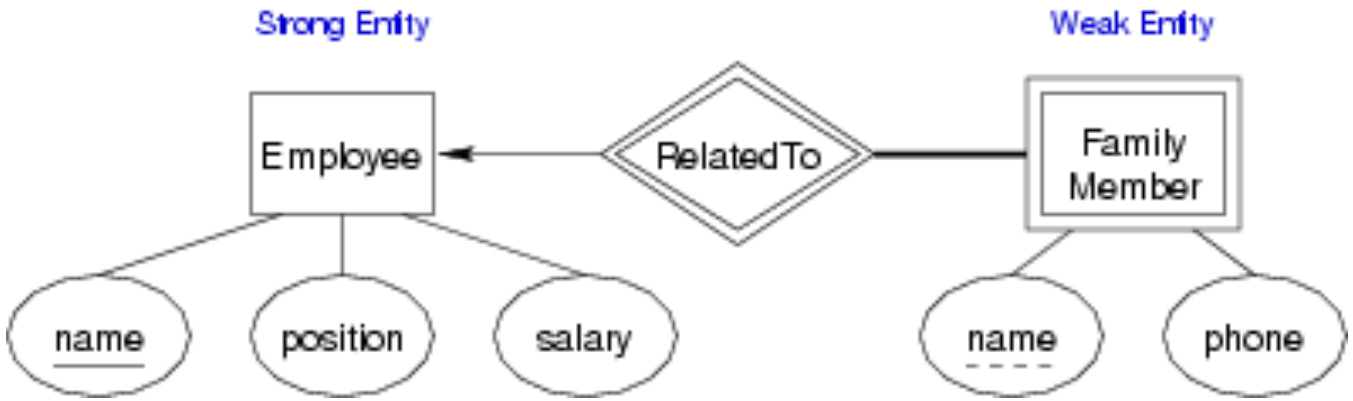


# Weak Entity Sets

Weak entities

- exist only because of association with strong entities.
- have no key of their own; have a *discriminator*

**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

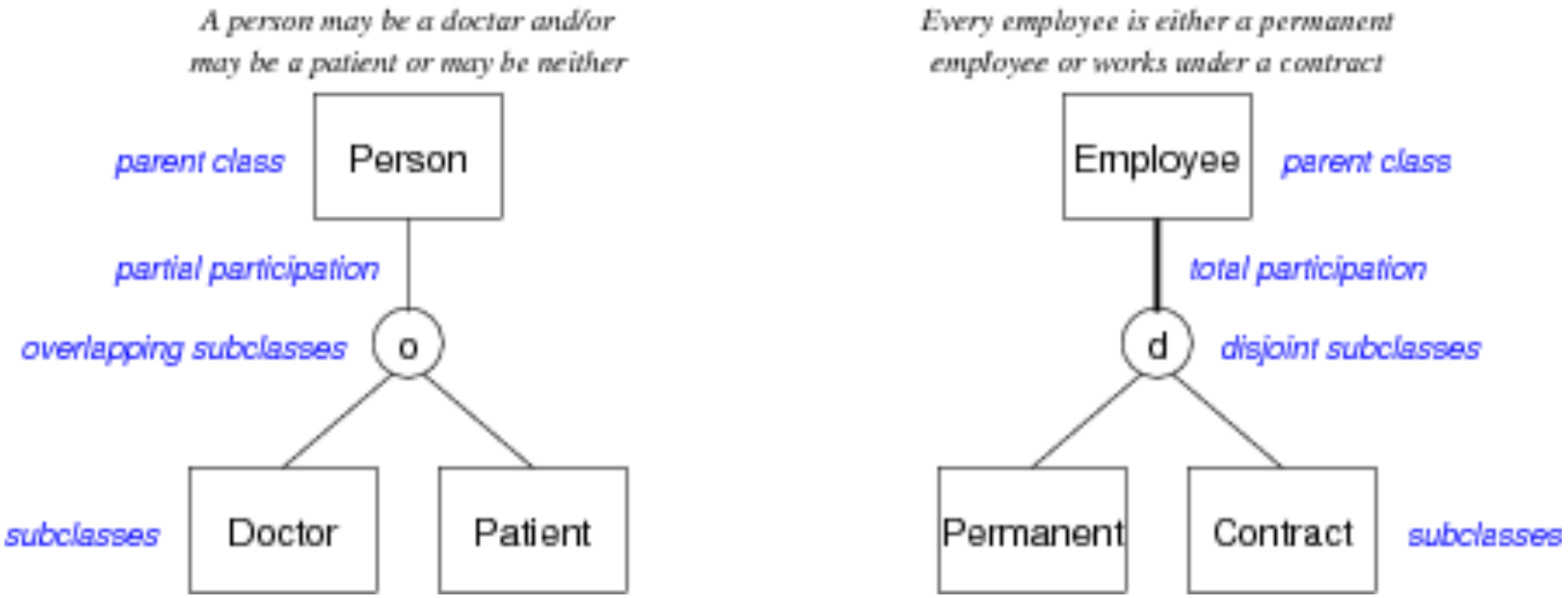
Properties of subclasses:

- *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**:



# Design Using the ER Model

ER model: simple, powerful set of data modelling tools

Some considerations in designing ER models:

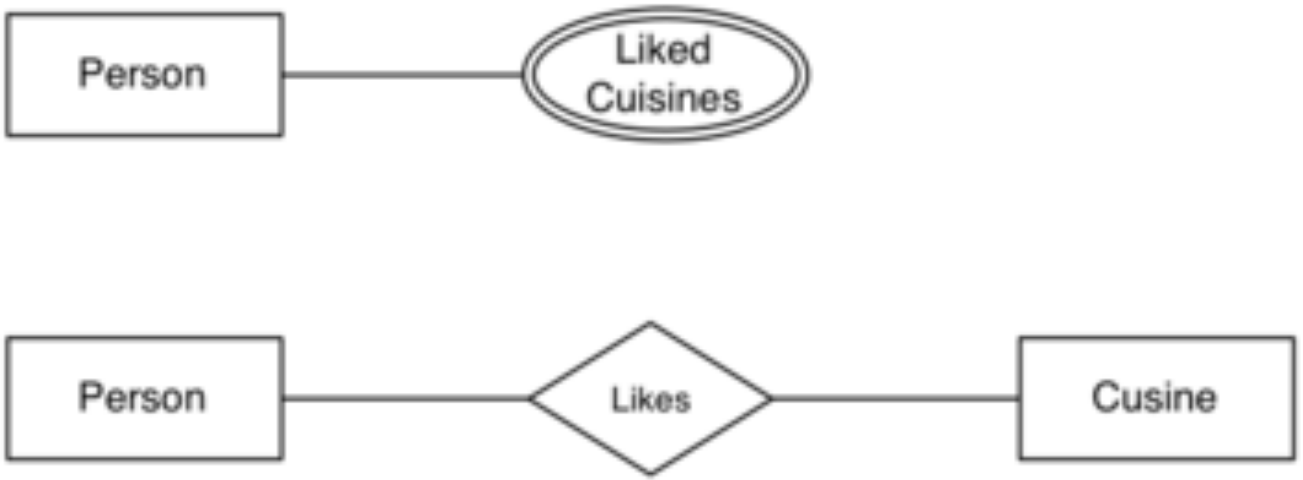
- should an "object" be represented by an attribute or entity?
- is a "concept" best expressed as an entity or relationship?
- should we use *n*-way relationship or several 2-way relationships?
- is an "object" a strong or weak entity? (usually strong)
- are there subclasses/superclasses within the entities?

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

# Exercise 3: ER Design Choices

The following two diagrams both represent

a person has some types of food that they like




Why might we favour one over the other?

# 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

# Exercise 4: Medical Information

Develop an ER design for 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.