COMP9311 - Database Systems

Week 1

Introduction, Data Modeling, and ER Notation

School of Computer Science and Engineering (CSE)

Lecturer: Dr. Rachid Hamadi

Email: rhamadi@cse.unsw.edu.au

Course Website:

http://www.cse.unsw.edu.au/~cs9311

Course Information

- Lectures: **Wednesdays 15:00 - 18:00**

  Location: **ChemScM17**

- Labs: **Weeks 2 – 13, 2 hours per week in labs**

- Consultation: **Wednesdays 14:00 - 15:00**

  Location: **K17-203**

Course Information (cont'd)

- This semester, there will be three (3) assignments and a final exam
- Assignments are worth **40%**:
  - Assignment 1: ER Diagram / Relational Mapping (**10%**) (due **Week 5**)
  - Assignment 2: SQL Queries and Functions (15%) (due **Week 8**)
  - Assignment 3: Normalisation / Relational Algebra / Transaction (**15%**) (due **Week 11**)
- **Penalty for late submission:** 10% reduction of the total mark for each day late.

Course Information (cont'd)

- **Final Exam:**

  Two (2) hours **written** exam

  Worth **60%** during exam period

- **Final Mark:**

  FinalMark = A1Mark + A2Mark + A3Mark + ExamMark

- **Hurdle:**

  To pass this course, you must obtain **at least 40%** in the final exam with a combined total of at least 50%

Course Information (cont'd)

- Textbook:
  - Fundamentals of Database Systems , Elmasri and Navathe, 7th edition, 2016, Addison-Wesley

- Reference books:
  - Database System Concepts , Silberschatz, Korth, Sudarshan, 6th edition, 2010, McGraw-Hill
  - Database Management Systems , Ramakrishnan and Gehrke, 3rd edition, 2003, McGraw-Hill
  - Database Systems: The Complete Book , Garcia-Molina, Ullman, Widom, 2nd edition, 2008, Prentice-Hall
  - Database Systems: An Application-Oriented Approach Kifer, Berstein, Lewis, 2nd edition (Complete Version), 2006, Addison-Wesley

Note: Earlier editions are fine for this course

Course Schedule

The following is an **approximate** guide to the sequence of topics in this course.

Week	Lectures	Assignments
1	Introduction, Data Modelling, ER Notation	
2	Relational Model, ER-Relational Mapping, SQL Schemas	
3	DBMSs, Databases, Data Modification	
4	SQL Queries	
5	More SQL Queries, Stored Procedures, PLpgSQL	**Assignment 1 Due**
6	Extending SQL: Queries, Functions, Aggregates, Triggers	
7	More Triggers, Programming with Databases	
8	Catalogs, Privileges	**Assignment 2 Due**
9	Relational Design Theory, Normal Forms	
-	Non-teaching week (mid-semester break)	-
10	Relational Algebra, Query Processing	
11	Transaction Processing, Concurrency Control	**Assignment 3 Due**
12	Course Review	

Databases in CSE

COMP9311 introduces foundations & technology of databases

- Skills: how to build database-backed applications
- Theory: how do you know that what you built was any good

After COMP9311 you can go on to study ...

- COMP9315: how to build relational DBMSs (write your own Oracle)
- COMP9318: techniques for data mining (discovering patterns in DB)
- COMP6714: information retrieval, web search (dealing with text data)
- COMP9319: web search and data compression (dealing searching compressed web data)
- COMP932(1|2|3): service-oriented computing, which relies on DB background

Introduction

- Database Applications:
  - Banking System,
  - Stock Market,
  - Transportation,
  - Social Network,
  - Marine Data Analysis,
  - Criminal Analysis and Control,
  - Now, Big Data ...

Introduction (cont’d)

Intelligent Transportation

Business Services

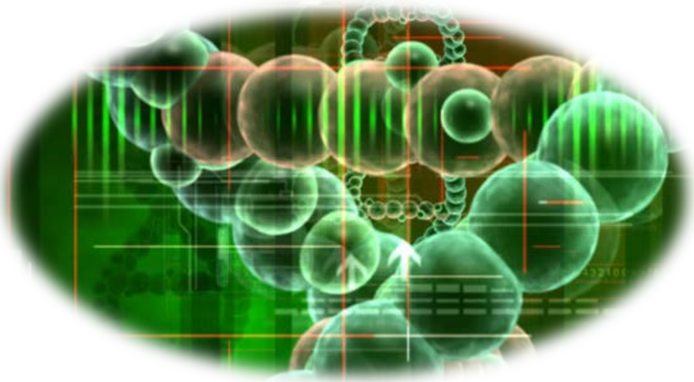
Natural Disasters

Public Health

Modern Military

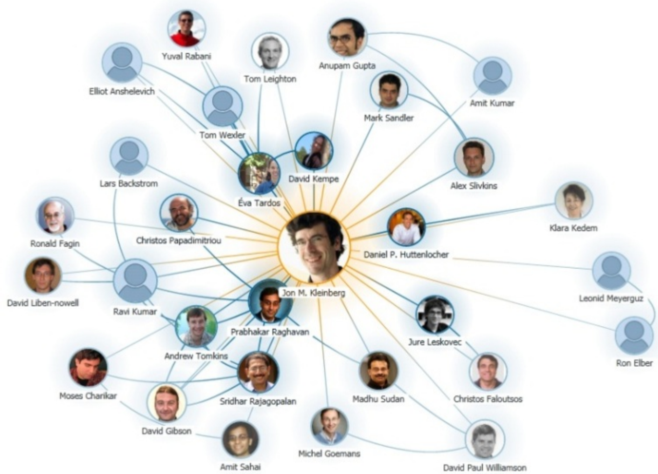
Tourism Development

More Applications



Chemical Compounds

Social Network



Collaboration Graph

Road Network

Graphs Could be Big!

- 1.23 billon active users in 2013
- 117 billion friendships in 2013

- 645 million users in 2013
- 1.7 billion tweets/month in 2013

Databases: Important Themes

The field of **databases** deals with:

- **Data** ... representing application scenarios
- **Relationships** ... amongst data items
- **Constraints** ... on data and relationships
- **Redundancy** ... one source for each data item
- **Data manipulation** ... declarative, procedural
- **Transactions** ... multiple actions, atomic effect
- **Concurrency** ... multiple users sharing data
- **Scale** ... massive amounts of data

What is data?

- *Data* (Elmasri/Navathe)
  - Known facts that can be recorded and have explicit meaning
- *Example* - a student records database
  - Contains information identifying students, courses they are enrolled in, results from past courses, …

Item	Type of data	Stored as
Family name	String	Character strings?
Birthdate	Date	3 integers?
Weight	Real number	Floating point number?
…		

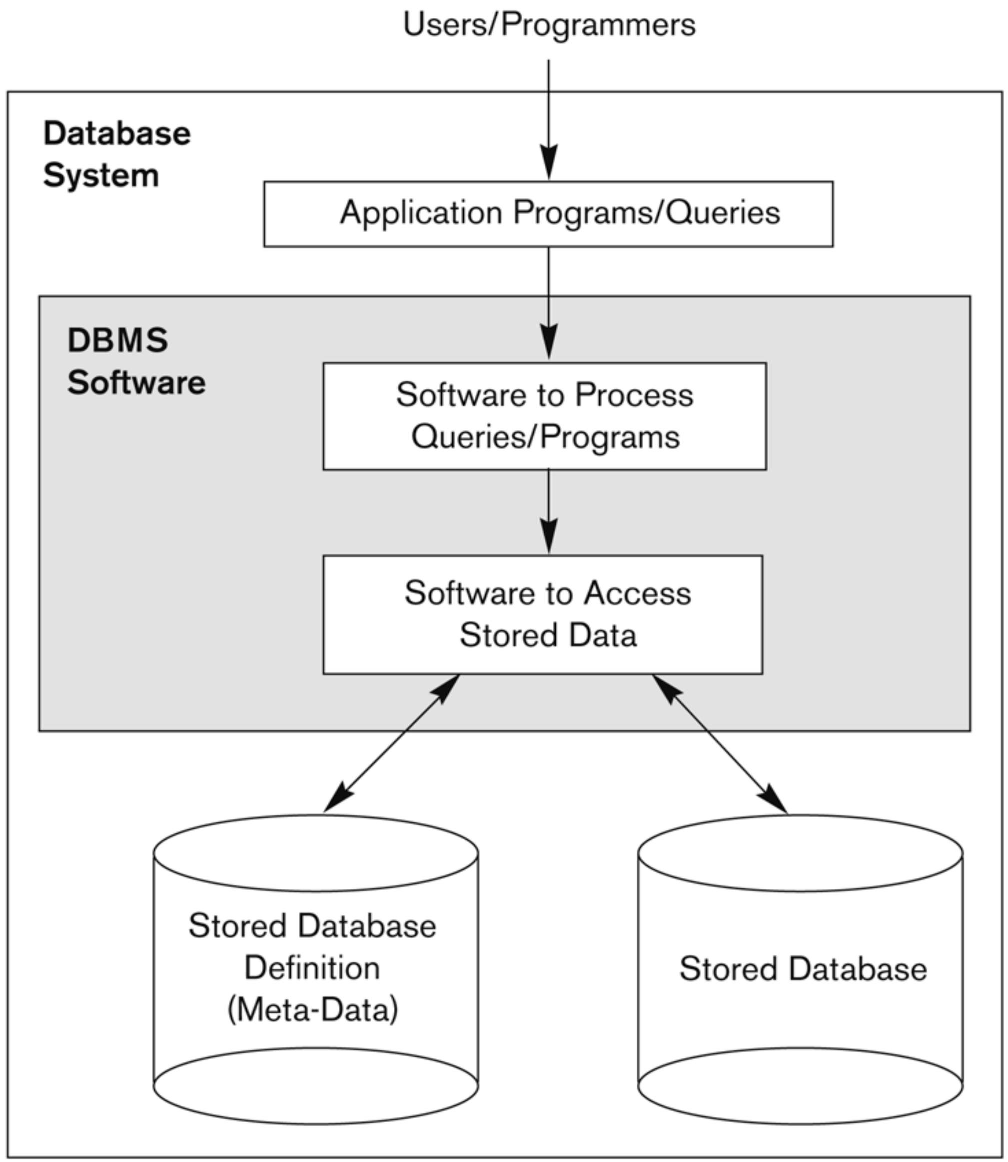
What is a database?

- *Database* (Elmasri/Navathe)
  - . . . a collection of **related** data . . .
- Data items alone are relatively useless
- We need the data to have some structure
- Database can be manipulated by a **database management system**

What is a database management system (DBMS)?

- Elmasri/Navathe
  - *DBMS*: … a collection of programs that enables users to create and maintain a database …
  - *Database system*: … the database and DBMS together …

Simplified Database System Environment
(Elmasri/Navathe)



Database Users

- *Database Administrator (DBA)*
  - Responsible for authorizing access to the database, coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations

- *Database Designer*
  - Responsible for defining the structure, the constraints, and transactions. Must communicate with the end-users and understand their needs

Database Users (cont’d)

- *End Users* - Use the data for queries, reports and some of them update the database content. They can be categorized into:
  - **Casual**: access database occasionally when needed
  - **Naive** (or Parametric): they make up a large section of the end-user population.
    - They use previously well-defined functions in the form of “canned transactions” against the database.
    - Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.

Categories of End-Users (cont'd)

- **Sophisticated**: these include business analysts, scientists, engineers, others who are thoroughly familiar with system capabilities. Many use tools in the form of software packages that work closely with the stored database.

- **Stand-alone**: mostly maintain personal databases using ready-to-use packaged applications. An example is a user who maintains her/his own address book

Database Management Systems

DBMS for practical work

- PostgreSQL 9.3.3 (open-source, free, full-featured)

Comments on using a specific DBMS:

- the primary goal is to learn SQL (a standard)
- the specific DBMS is not especially important
- but, each DBMS implements non-standard features
- we will use standard SQL as much as possible
- an exception is PL/pgSQL (but close to Oracle's PL/SQL)
- PostgreSQL documentations describe all deviations from standard

Database Management Systems (cont'd)

Comments on PostgreSQL vs Oracle

- Oracle is resource hungry (>800MB vs <200MB for PostgreSQL)
- PostgreSQL is a commercial-strength (ACID) RDBMS

  ... but, being open source, you can see how it works
- PostgreSQL has been object-relational longer than Oracle

  ... and its extensibility model is better than Oracle's
- PostgreSQL is more flexible than Oracle

  ... allows stored procedures via a range of programming languages

But note: PostgreSQL and Oracle have very close SQL and PL/SQL languages

Database Management Systems (cont'd)

Comments on PostgreSQL vs MySQL

- both open source* and reasonably efficient
- most Web/DB developers use MySQL
- until v4/5, MySQL lacked many serious DB concepts
  - no transactions, foreign keys, subselects, views, procedures, …
- MySQL's SQL often ignores SQL standards
- MySQL is hacked together from "imported components"
  - multiple storage engines (some still w/o transactions)

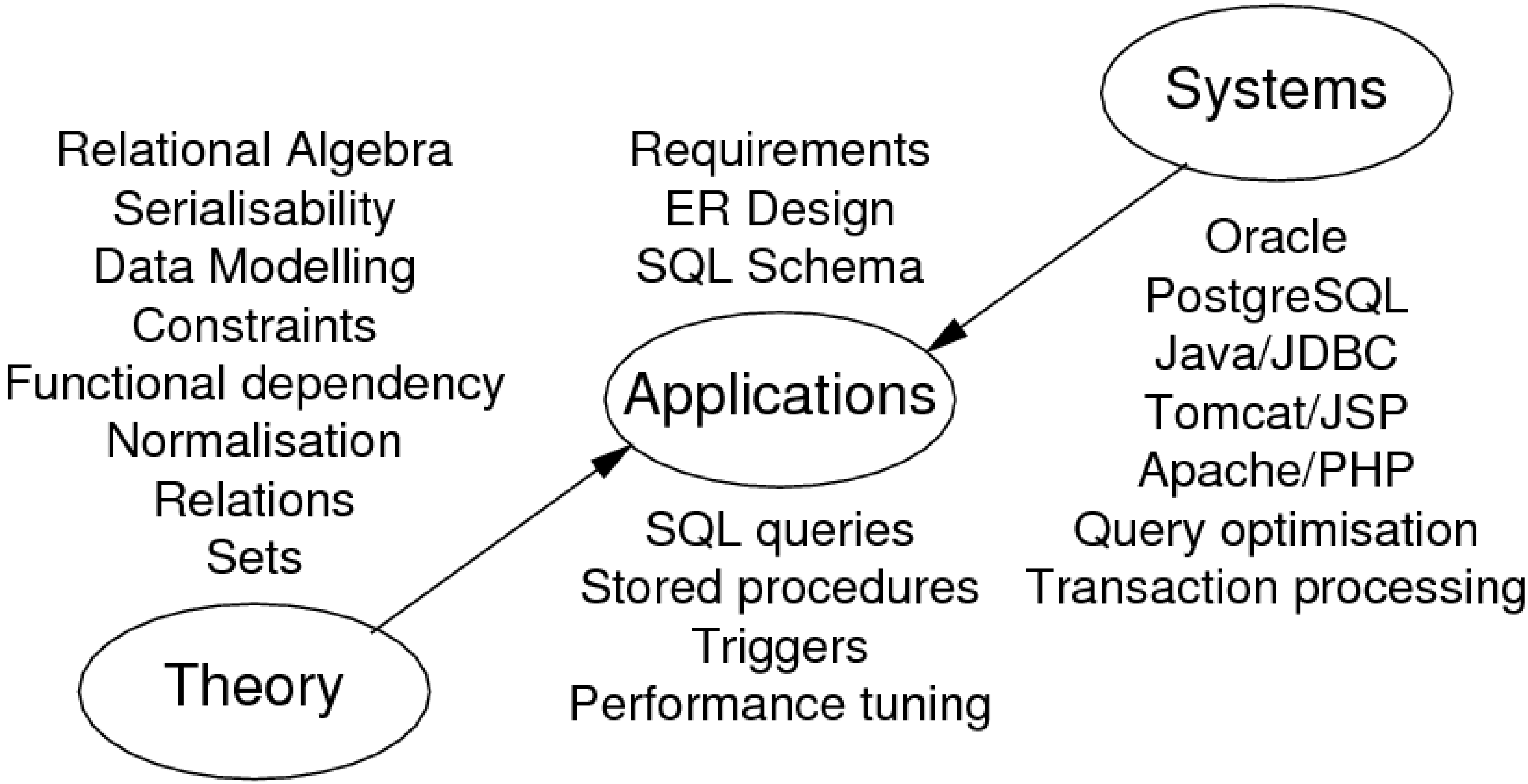
PostgreSQL is better engineered; MySQL is more popular

* But Oracle now controls MySQL ⇒ open-source status unclear

Working from Home

- PostgreSQL has very good on-line documentation
- This semester we will be using PostgreSQL 9.3.3
- If you install them at home
  - Get a version "close to" that (the latest stable is 9.6)
  - Test all assignment work at CSE before submitting
- Alternative to installing at home
  - Run them on CSE servers (grieg) as you would do in labs
  - Use, e.g., Putty to log in to a CSE server from home
  - PostgreSQL via Putty ok, since command-line based

Overview of the Database Areas



Database Application Development

A variation on standard software engineering process

1. analyse application requirements
2. develop a data model to meet these requirements
3. define operations (transactions) on this model
4. implement the data model as relational schema
5. implement transactions via SQL and PLs
6. construct a web interface to these transactions

At some point, populate the database (via interface?)

Database System Languages

Requests to DBMS

- **queries**, **updates** in data manipulation language (DML) (e.g. SQL)
- **data structures**, **constraints** in data definition language (DDL) (e.g. SQL)
- **create** and **drop** databases, **indexes**, **functions** **(**e.g. PL/pgSQL)

Results/effects from DBMS requests

- **tuples** (typically, sets of tuples)
- changes to underlying data store

Data Modeling

Aims of data modeling

- 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 COMP9311, Paul's account is held at Coogee)
- describe **constraints** on data
  (e.g. 7-digit IDs, students can enrol in no more than 30UC per semester)

Data modelling is a **design** process

- converts requirements into a data model

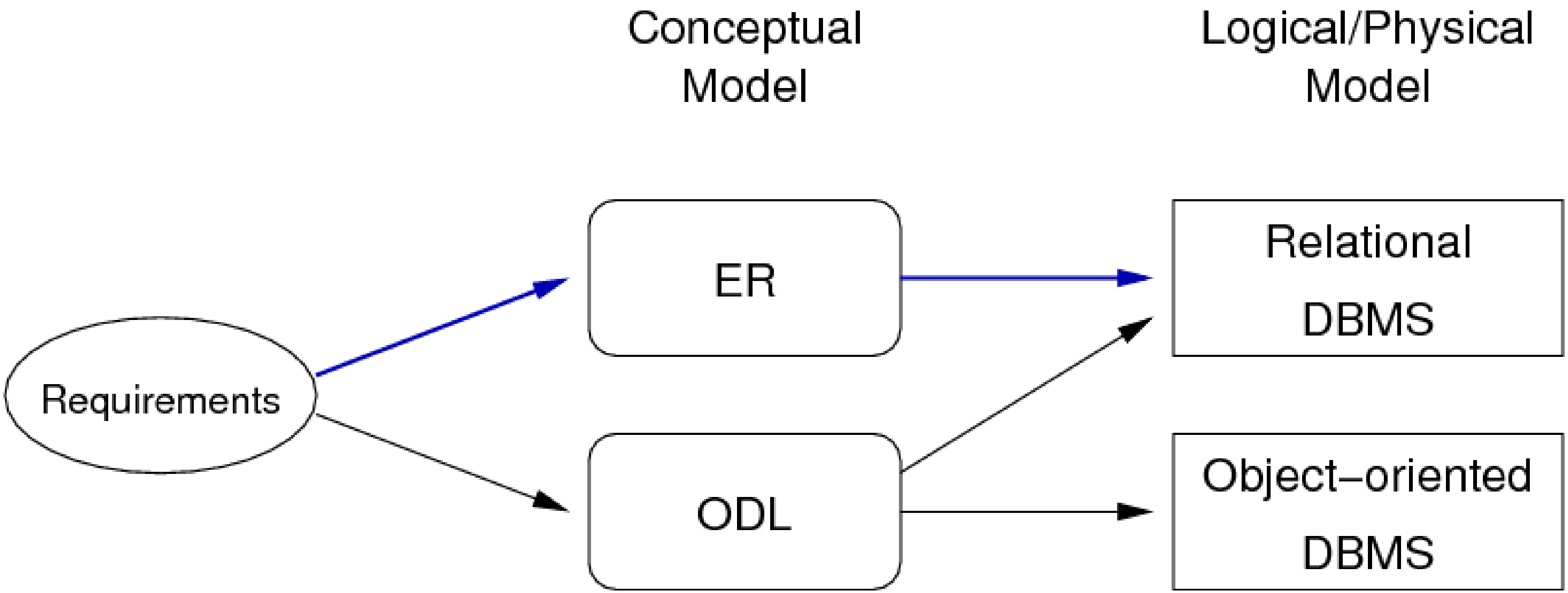
Data Modeling (cont'd)

Kinds of data models:

- **logical**: abstract, for conceptual design
  e.g., Entity Relationship (ER), Object Definition Language (ODL)
- **physical**: record-based, for implementation
  e.g., relational

Strategy: design using abstract model; map to physical model

Data Modeling (cont'd)



Some Design Ideas

Consider the following while modeling data:

- 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: Gmail Data Model

Consider the Gmail system

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

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 requirements 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 (ER) Model

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. Hence, many variations exist

Lecture notes use notation from SKS and GUW books (simple)

Entity-Relationship (ER) Diagrams

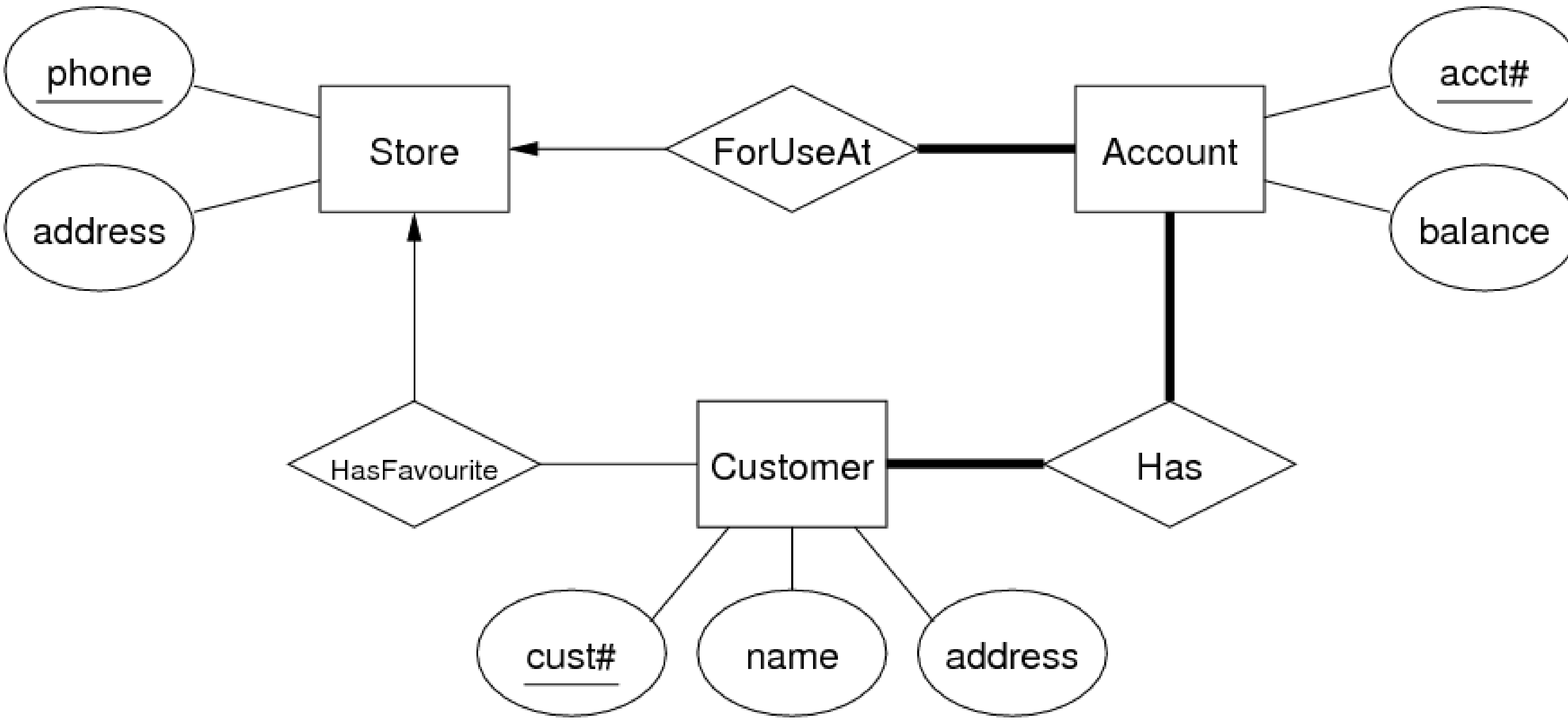
ER diagrams are a graphical tool for data modelling

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

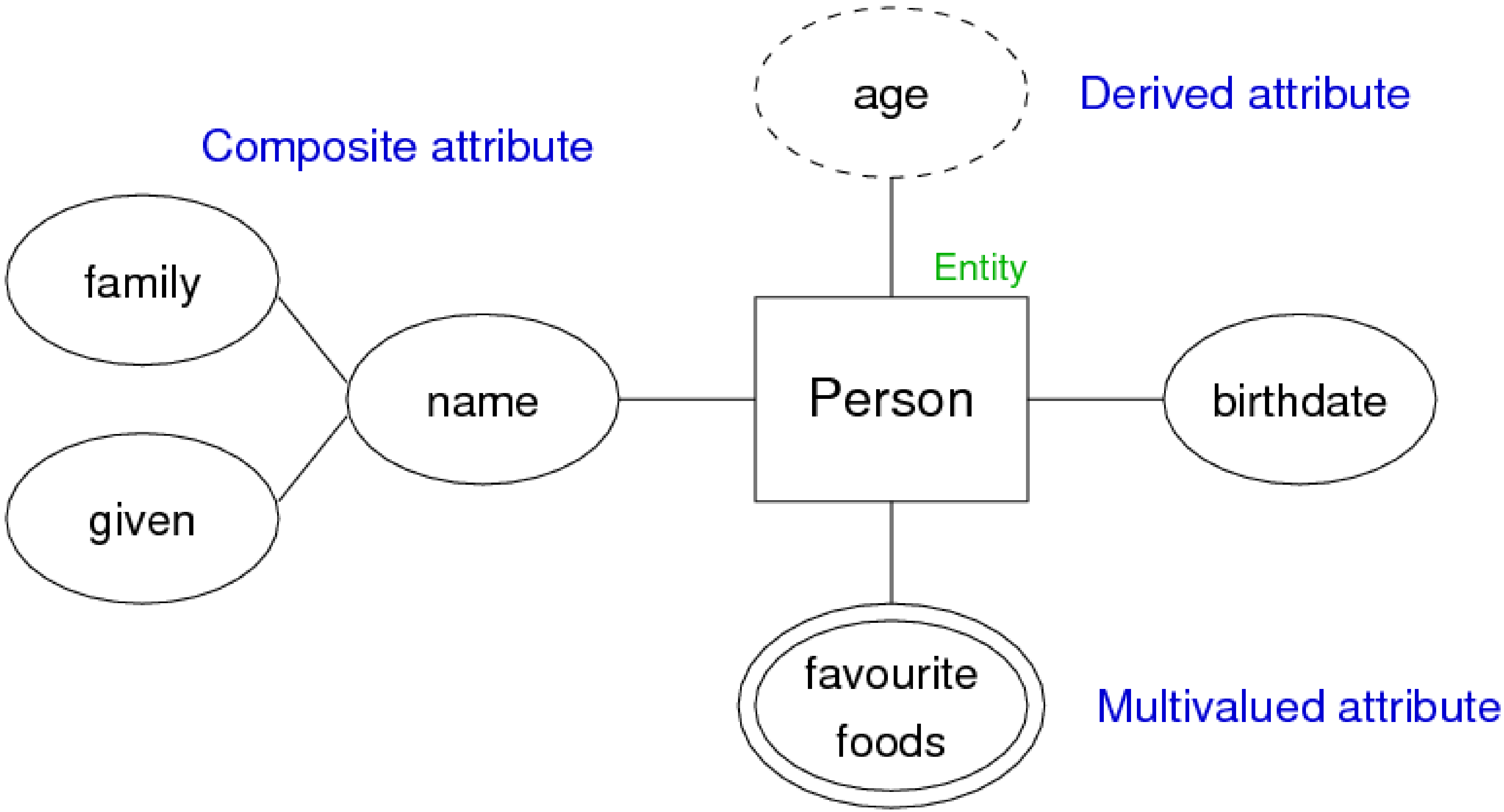
Terminology abuse: "entity" means "entity set" or "entity instance"?

Entity-Relationship (ER) Diagrams (cont'd)



ER Diagram Example

Entity-Relationship (ER) Diagrams (cont'd)



Attribute Notations Example

Entity Sets

An **entity set** can be viewed as either:

- a set of entities with the same set of attributes (**extensional view** of entity set)
- an abstract description of a class of entities (**intentional view** of entity set)

An entity may belong to more than one entity sets

"Data" in a database ≅ collection of (extensional) entity sets

Keys

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)

A **candidate key** is any superkey such that

- no proper subset of its attributes is also a superkey

A **primary key**:

- is one candidate key chosen by the database 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 relationship (in ER model, ≥ 2)

Cardinality = # associated entities on each side of relationship

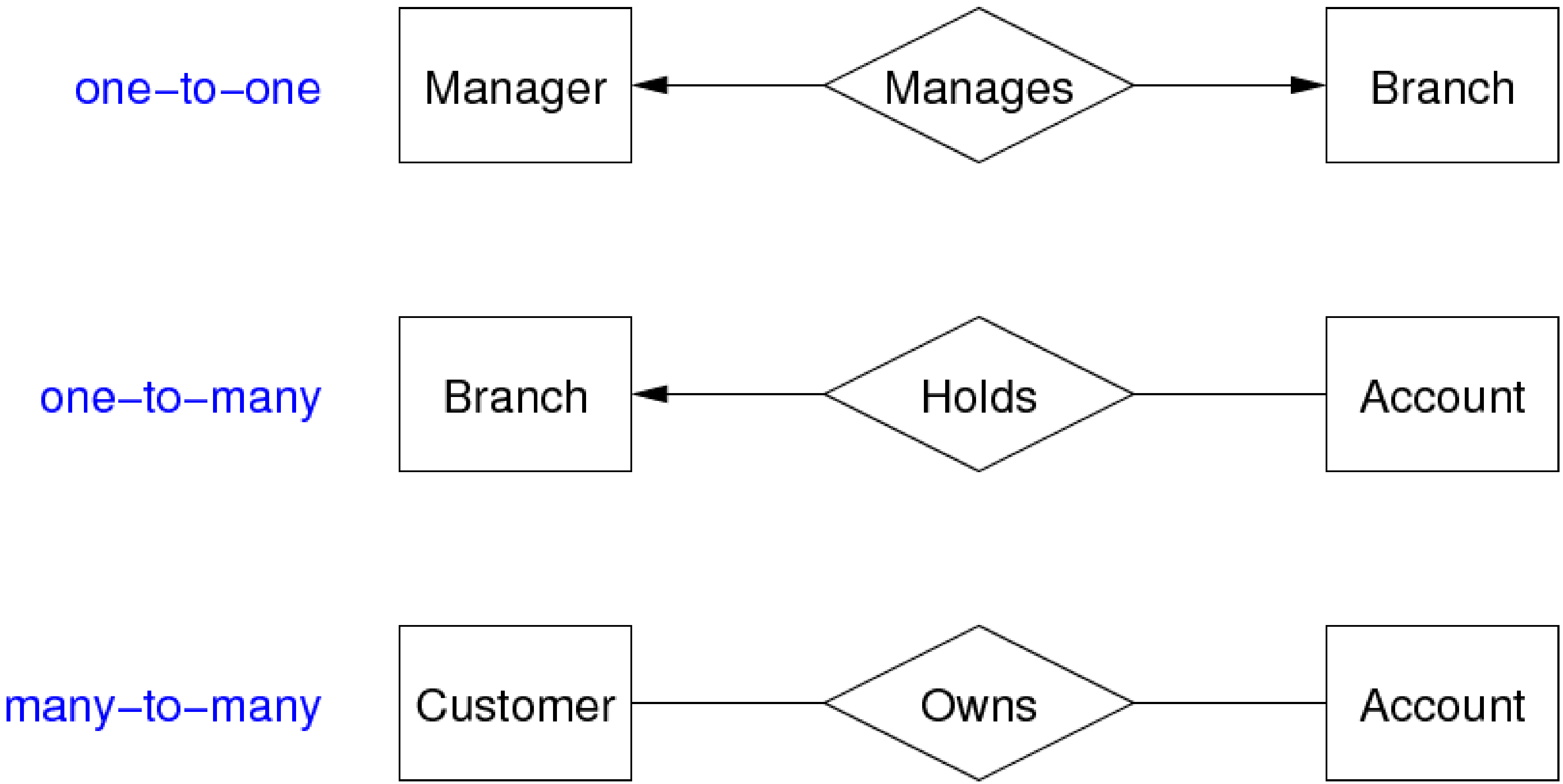
Participation = must every entity be in the relationship

Relationship Sets (cont'd)

Example: relationship participation

Relationship Sets (cont'd)

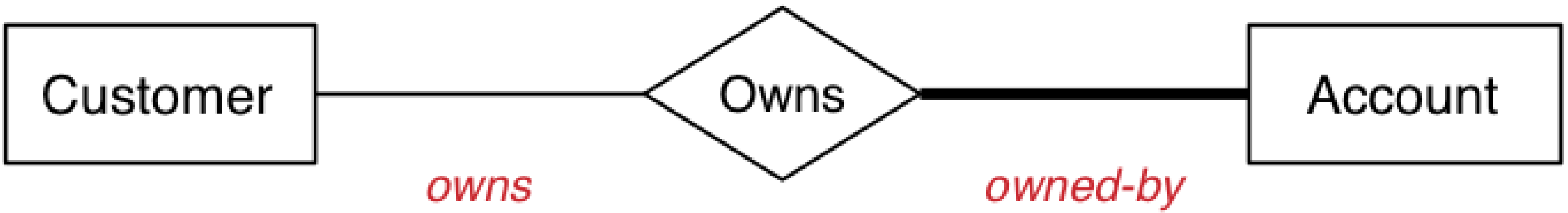
Example: relationship cardinality

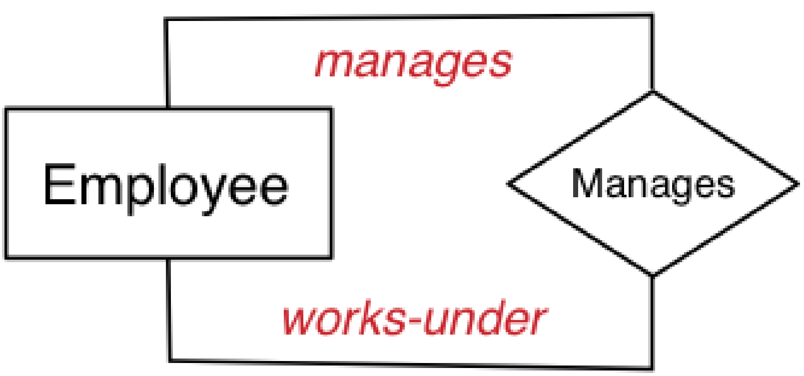


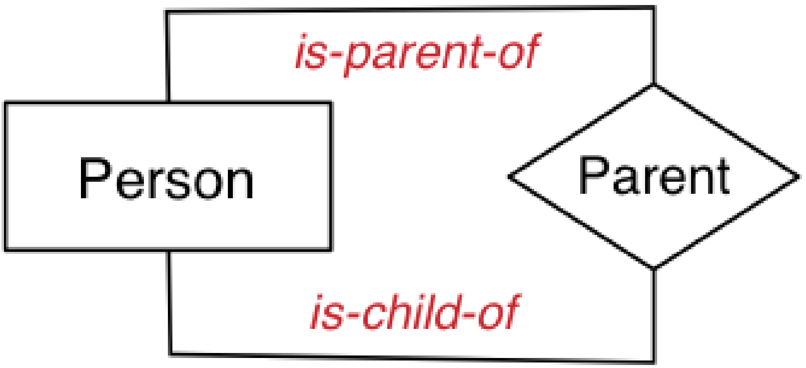
Relationship Sets (cont'd)

The role of each entity in a relationship is usually implicit.

If ambiguity arises, can explicitly name the role.





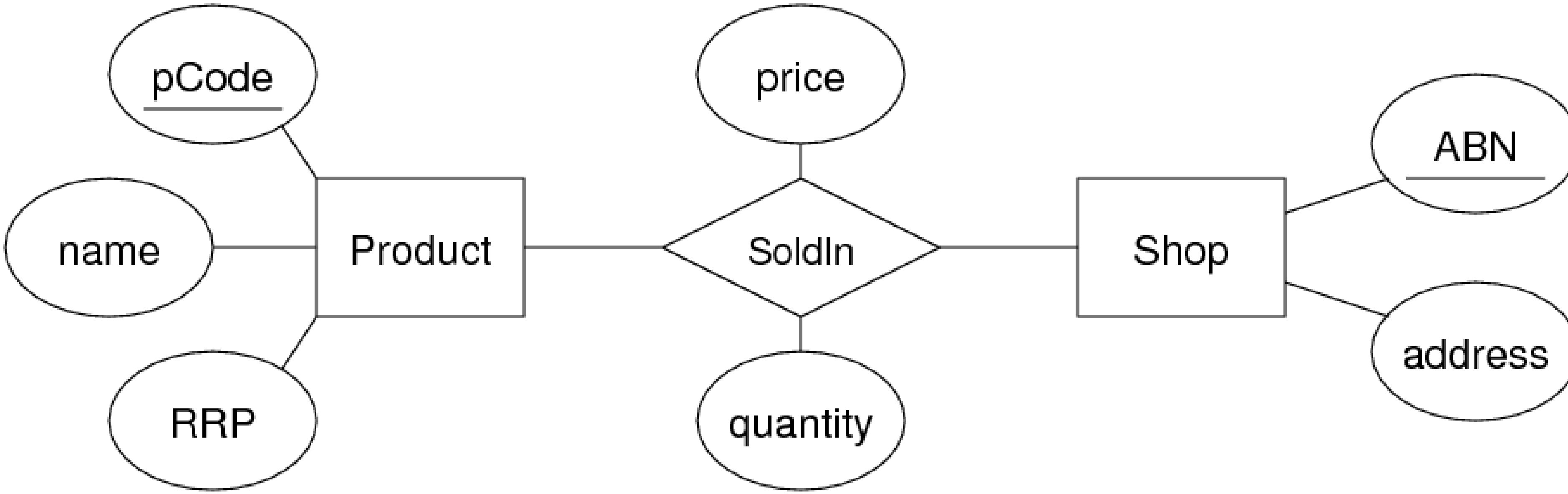


Role names become more important when developing SQL schemas

Relationship Sets (cont’d)

In some cases, a relationship needs associated attributes.

Example:



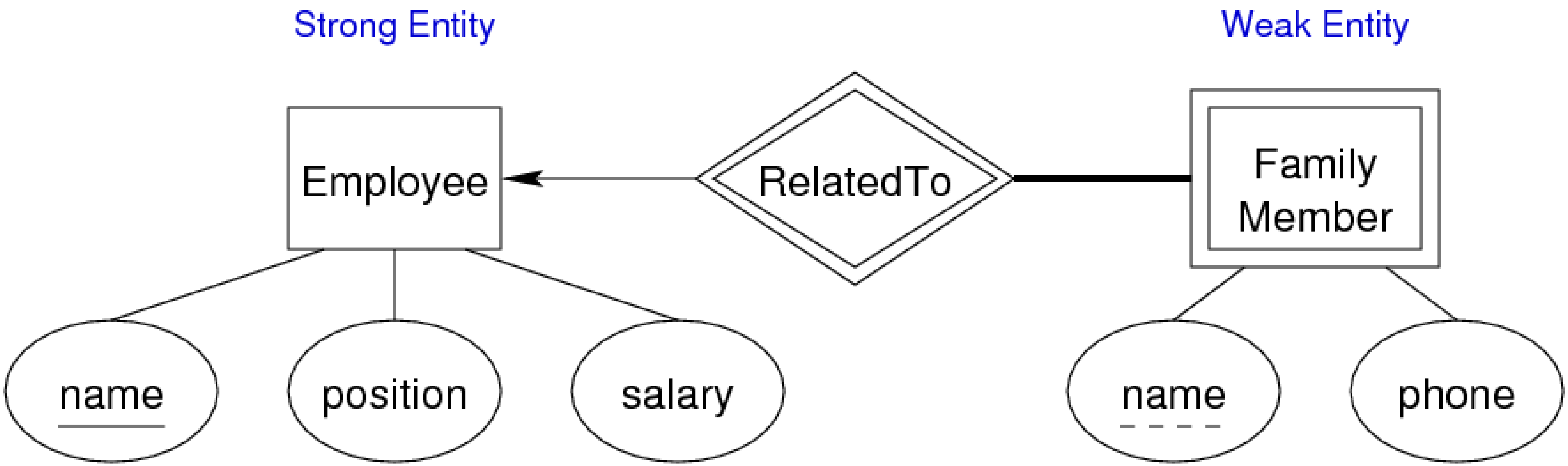
(Price and quantity are related to products in a particular shop)

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

Properties of subclasses:

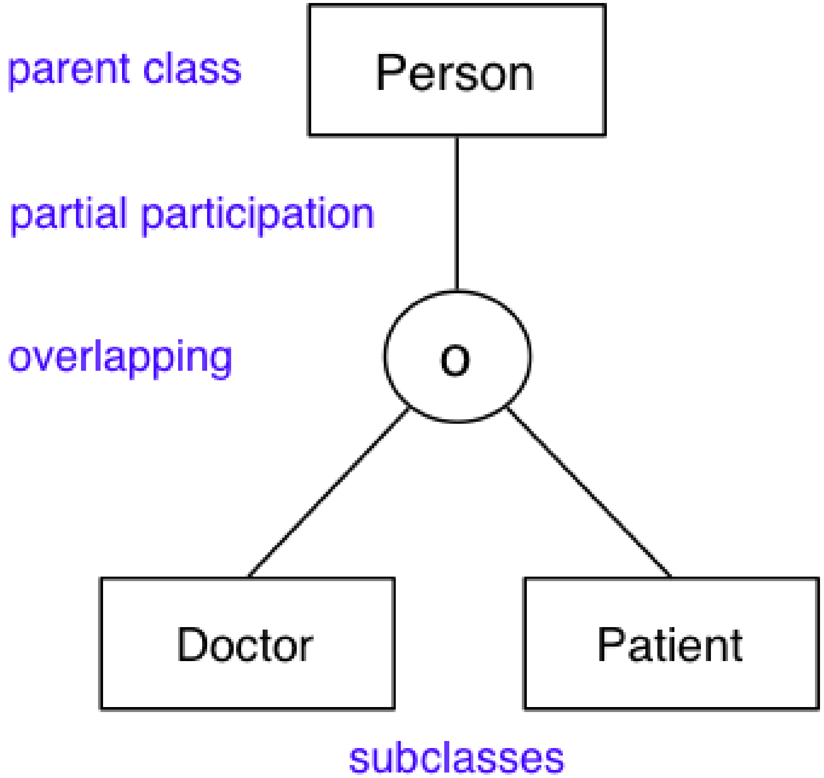
- **overlapping** or **disjoint** (can an entity be in multiple subclasses?)
- **total** or **partial** (does every entity have to also be in a subclass?)

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

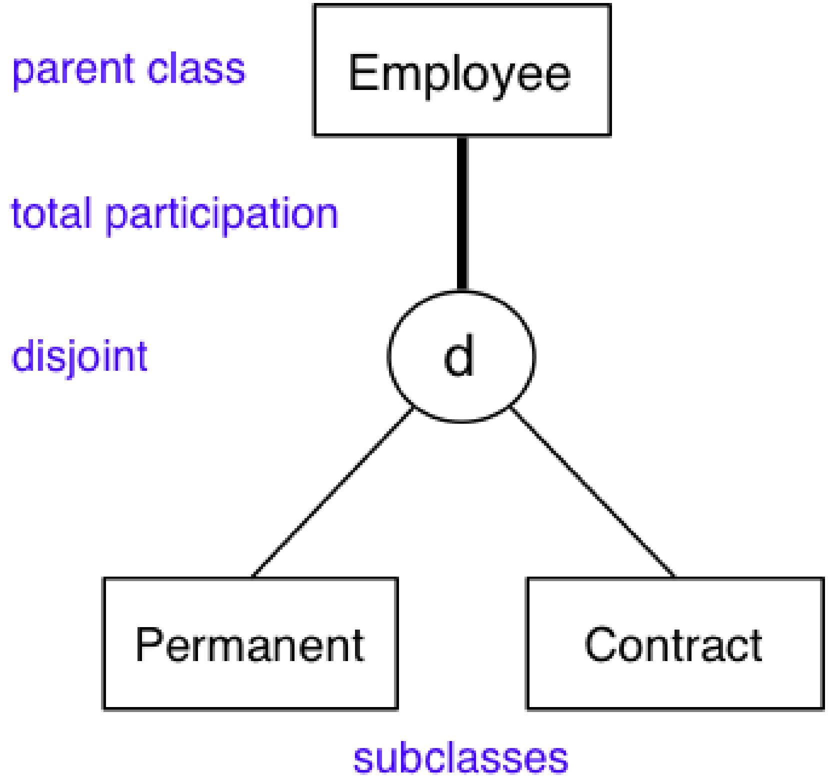
Subclasses and Inheritance (cont'd)

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 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

Design Using the ER Model (cont'd)

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 entity attributes)
- if very large design, may use several linked diagrams

Exercise: Medical Information

- 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.