COMP3311 Week 01 Lecture

COMP3311 Database Systems

COMP3311 Database Systems

Lecturer: ***Dr Rachid Hamadi***

Web Site: http://www.cse.unsw.edu.au/~cs3311/

Introduction

Lecturer

Name: Dr Rachid Hamadi

Email: rhamadi@cse.unsw.edu.au

Why Study Databases?

Every significant modern computer application has Large Data.

This needs to be:

- stored (typically on a disk device)
- manipulated (efficiently, usefully)
- shared (by many users, concurrently)
- transmitted (all around the Internet)

Red stuff handled by databases; brown by networks.

Challenges in building effective databases: efficiency, security, scalability, maintainability, availability, integration, new media types (e.g., music), ...

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 implicit meaning
- *Example* - a student records database
  - Contains information identifying students, courses they are enrolled in, results from past courses, ...

What is 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 Database Management System?

Elmasri/Navathe

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

Databases in CSE

COMP3311 introduces foundations & technology of databases

- skills: how to build database-backed applications
- theory: how do you know that what you built was good

After COMP3311 you can go on to study ...

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

---

Syllabus Overview

- Data modelling and database design
  - ER model, ODL, ER-to-relational
  - Relational model (design theory, algebra)
- Database application development
  - SQL, views, stored procedures, triggers, aggregates
  - SQLite: `sqlite3` (an SQL shell)
  - PostgreSQL: `psql` (an SQL shell), `PLpgSQL` (procedural),
  - Programming language access to databases (PHP, ORMs)

The brown stuff is not covered in lectures and is not examinable

---

... Syllabus Overview

- Database management systems (DBMSs)
  - DB Administration: catalogs, access control, performance
  - DBMS architecture: client/server, file system, relational engine
  - Storage and indexing, data access operations
  - Query processing: translation, optimisation, evaluation
  - Transaction processing: transactions, concurrency control, recovery
- Future of Databases
  - Limitations of RDBMS's, potential future technologies

The green stuff is covered only briefly; details are in other courses such as COMP9315

Your Background

Required: programming (essential) and data-structures (helpful)

- The official pre-requisite for this course is that students must have taken either COMP1531 Software Engineering Fundamentals or COMP1927 Computing 2
- Whatever the formal pre-requisites, we assume primarily that students have some experience with procedural programming and some knowledge of elementary data structures

---

Teaching/Learning

Stuff that is available for you:

- *Texts*: describe most syllabus topics in detail
- *Lectures*: describe all syllabus topics in some detail, with exercises

Things that you need to **do**:

- Theory *exercises*: tutorial-type questions
- *Prac* work: lab-like exercises
- *Assignments*: extended practical exercises
- *Quizzes*: periodic progress check

... Teaching/Learning

Scheduled classes?

- two **2-hour** lectures each week (Tue 1-3, Thu 9-11)
- there are **no** tute classes or lab classes

What to do if you have problems understanding stuff?

- ask a question during/after the lecture
- come to a *consultation* (Tue 3-4, Thu 11-12)
- ask about it on the *Forums* (under WebCMS3)
- send an email to the Lecturer in Charge or Course Admin

Since no tutes/labs *you* need to push yourself to keep up-to-date.

Distance (Web) Mode

COMP3311 is offered in *regular* and *Web* mode

- all material is the same (incl. assignments and exam)
- videos of all lectures will be available
- all other material is online

Only difference: indicate that you won't be attending lectures

Allows to organize a better-sized lecture theatre

... Distance (Web) Mode

On the course website, you can:

- find out the latest course news/announcements
- view lecture slides/videos and collect all-in-one lecture notes
- get all of the information about theory/prac exercises
- get assignment specs/material
- do the quizzes
- get your questions answered (via the Forums)

URL: http://www.cse.unsw.edu.au/~cs3311/

Lectures

Lectures have two purposes:

- introduce content
- work through exercises

Lectures are intended to be interactive, so ask questions

All lectures are recorded and available via Moodle (link is added in the course website)

All exercise solutions are attached to the course website

Assignments

Two assignments, which are critical for *learning*

1. Queries/Procedures, SQL/PLpgSQL, due end week 6
2. SQL/PLpgSQL/PHP, due end week 10

All assignments are done *individually*, and ...

- submitted via WebCMS3
- automarked (so you must follow specifcation exactly)
- plagiarism-checked (copying solutions ⇒ **0** mark for course)
- rent-a-coder monitored (buying solutions ⇒ **exclusion**)

Quizzes

Four quizzes, each worth 2.5 marks

- cover material in previous few weeks lectures
- aim to check your understanding of recent material
- done via WebCMS3 in your own time
- mainly multiple-choice, but some drawing/text
- held in weeks 3, 5, 7, 9

Exam

3-hour **online** exam during exam period

Comprising a mixture of

- SQL, PLpgSQL, PHP, design exercises, analyses

Prac part: SQL using SQLite

All questions: typed in and submitted online

Only PG/SQ/PHP documentation is accessible during exam

Sample exams will be available in the course website

Supplementary Assessment Policy

Everyone gets **exactly one chance** to pass the Exam

If you attend the Exam

- I assume that you are fit/healthy enough to take it
- no 2nd chance exams, even with a medical certificate

All Special Consideration requests:

- must *document* how *you* were affected
- must be submitted to UNSW (useful to email lecturer as well)

Supplementary Exams are held shortly after the exam period (end June/early July); don't leave town

---

Assessment Summary

Your final mark/grade will be determined as follows:

```
ass1    = mark for assignment 1      (out of 15)
ass2    = mark for assignment 2      (out of 15)
quizzes = mark for on-line quizzes   (out of 10)
examP   = mark for exam (practical)  (out of 30)
examW   = mark for exam (written)    (out of 30)
```

```
exam   = examP + examW              (out of 60)
okExam = examP > 12 && examW > 12   (after scaling)

mark   = ass1 + ass2 + quizzes + exam

grade  = HD|DN|CR|PS  if mark >= 50 && okExam
       = FL           if mark <  50 && okExam
       = UF           if !okExam
```

Textbook (options)

- Elmasri, Navathe
  Fundamentals of Database Systems (7th ed, 2016)
- Garcia-Molina, Ullman, Widom
  Database Systems: The Complete Book (2nd ed, 2008)
- Ramakrishan, Gehrke
  Database Management Systems (3rd ed, 2003)
- Silberschatz, Korth, Sudarshan
  Database System Concepts (6th ed, 2010)
- Kifer, Bernstein, Lewis
  Database Systems: Application-Oriented Approach (2nd ed, 2006)

Earlier editions of texts are ok

Database Management Systems

Two example DBMSs for prac work:

- SQLite (open-source, free, no server needed)
- PostgreSQL (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
- PG docs describe all deviations from standard

... Database Management Systems

Comments on PostgreSQL vs Oracle:

- Oracle is resource-intensive (>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

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

---

Further Reading Material

The on-line documentation and manuals provided with:

- SQLite are reasonably good
- PostgreSQL are very good
- PHP are similarly comprehensive

Some comments on these technologies books:

- tend to be expensive and short-lived
- many provide just the manual, plus some examples
- generally, anything published by O'Reilly is useful

Aside: once you understand the concepts, the manual is sufficient

---

Home Computing

Software versions that we'll be running this semester:

- PostgreSQL 9.4, SQLite 3.8, PHP 5.4

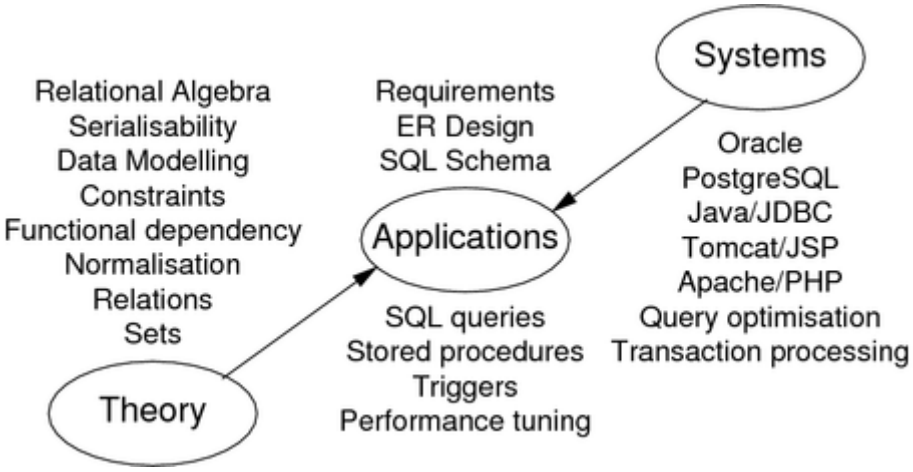
If you install them at home:

- get versions "close to" these
- test all work at CSE before submitting

Alternative to installing at home:

- run them on the CSE servers (grieg) as you would 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 Databases Field



Database Application Development

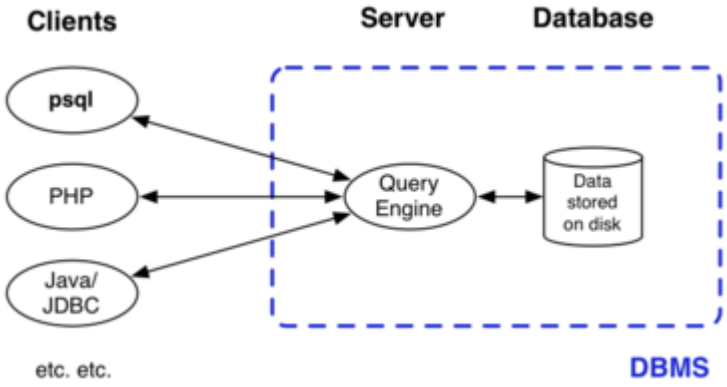
A variation on standard software engineering process:

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 procedural PLs
6. construct an interface to these transactions

At some point, populate the database (may be via interface)

Database System Architecture

The typical environment for a modern DBMS is:



SQL queries and results travel along the client↔server links

Data Modelling

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

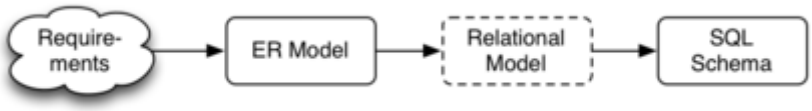
- converts requirements into a data model

... Data Modelling

Kinds of data models:

- *logical*: abstract, for conceptual design, e.g., ER, ODL
- *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.

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

[Solution]

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 (⇒ inconsistency)
- leads to an inefficient implementation
- violates syntactic or semantic rules of data model

Entity-Relationship (ER) 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 notation from SKS and GUW books (simple)

Entity-Relationship (ER) Diagrams

ER diagrams are a graphical tool for data modelling.

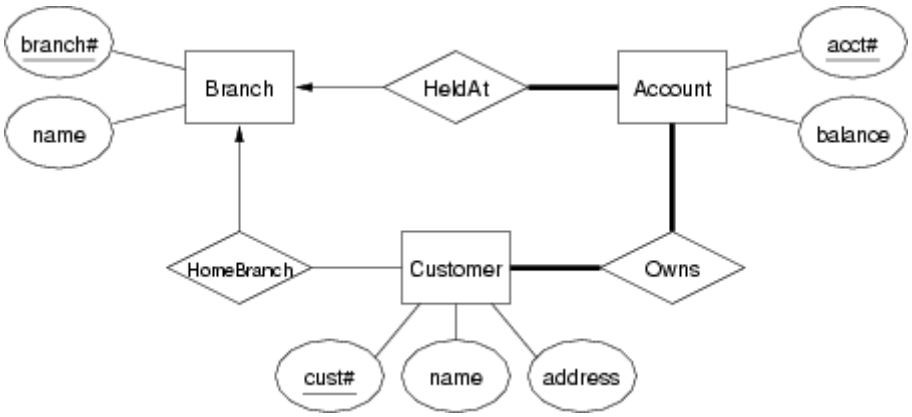
An ER diagram consists of:

- 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

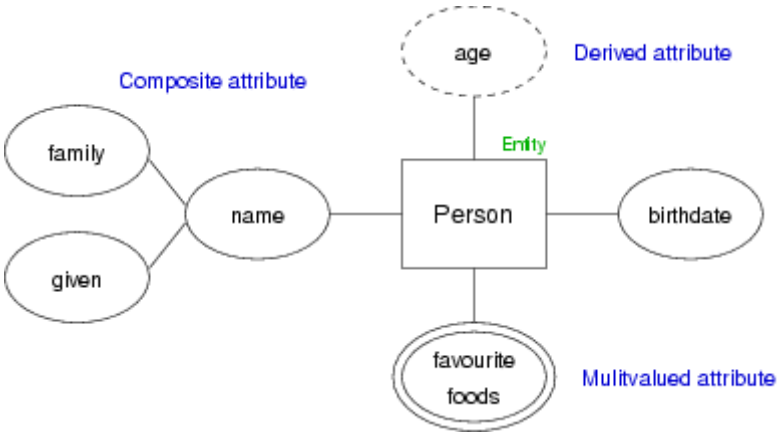
Terminology: when discussing "entity sets", we frequently say just "entity"

... Entity-Relationship (ER) Diagrams

Example ER diagram:



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)

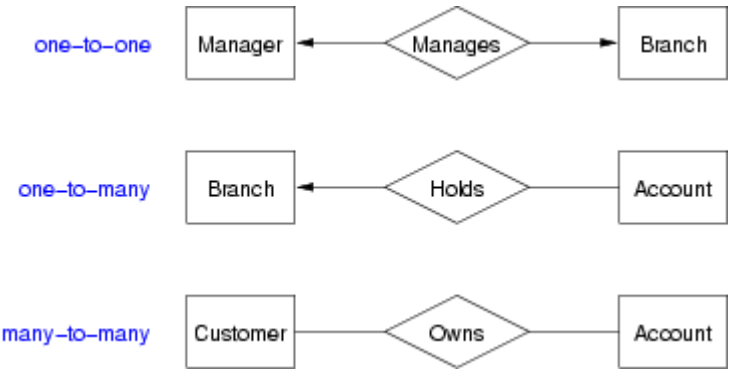
Cardinality = # associated entities on each side of reln

Participation = must every entity be in the relationship

Example: relationship participation

... Relationship Sets

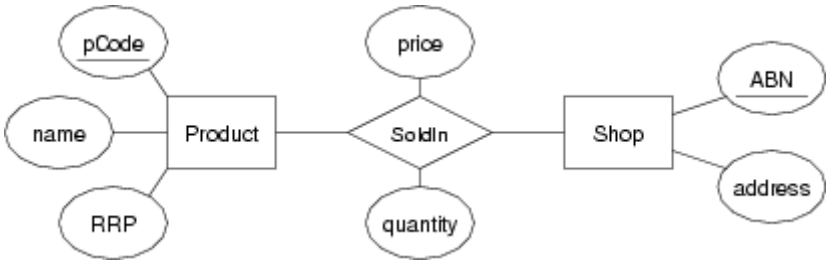
Examples: relationship cardinality



... Relationship Sets

In some cases, a relationship needs associated attributes.

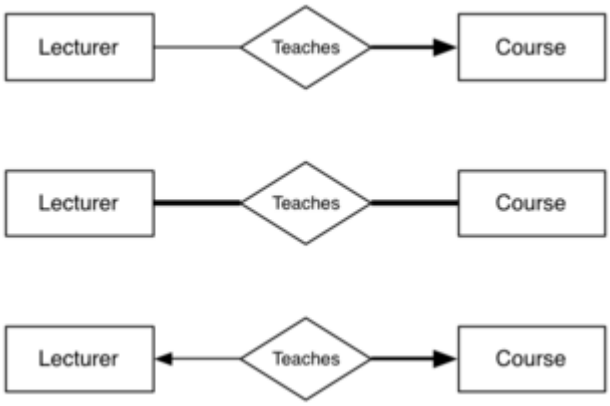
Example:



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

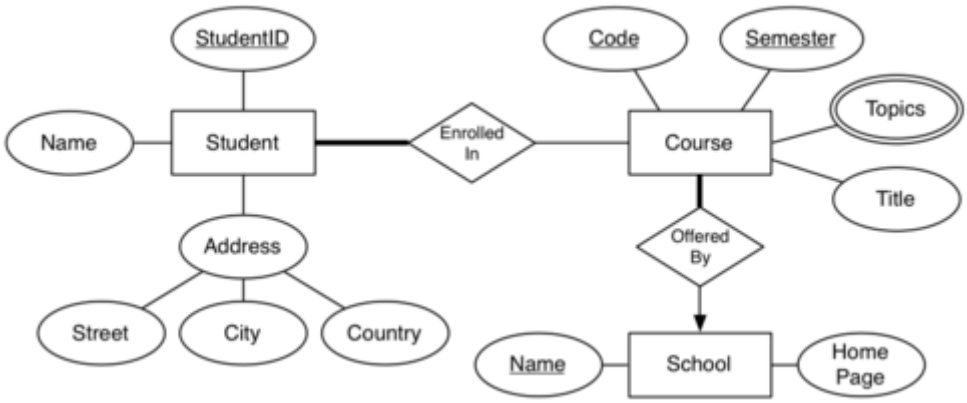
Exercise 2: Relationship Semantics

Describe precisely the scenarios implied by the following relationships:



ER: the story so far

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

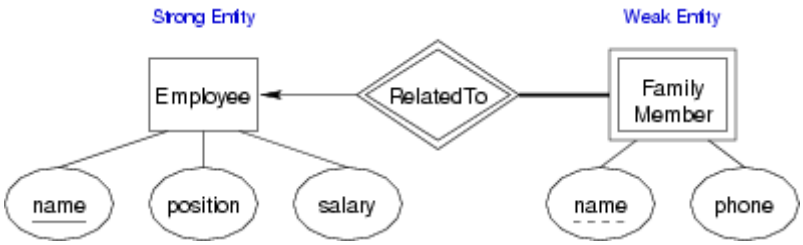


Weak Entity Sets

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

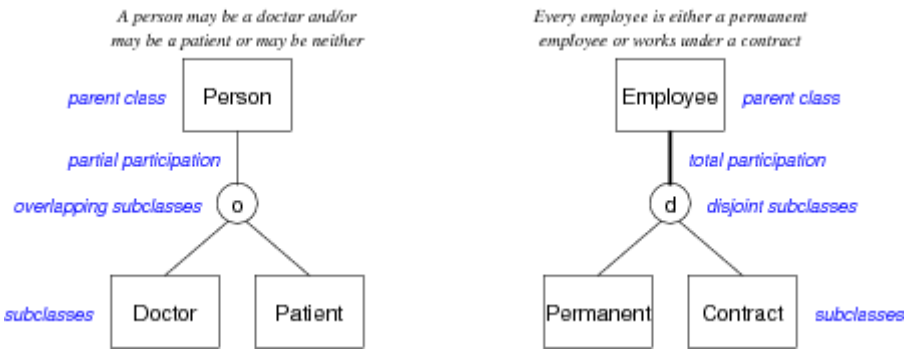
Properties of subclasses:

- *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 entitities 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.
  [Solution]
