COMP3311 Week 01 Lecture

COMP3311 Database Systems

Lecturer: ***Dr Raymond Wong***

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

Lecturer

2/62

Name:	Dr Raymond Wong
Office:	K17-213 (turn right from lift)
Phone:	9385 5932
Email:	wong@cse.unsw.edu.au
Consults:	Mon 9-10, Wed 9-10 in K17-213
Research:	Information Extraction Information Retrieval Data Mining Text Analytics

Why Study Databases?

3/62

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? What is a Database?

According to the Elmasri/Navathe textbook ...

- *Data* = known recorded facts, with implicit meaning
  - e.g. a student's name, a product id, a person's address or birthday
- *Database* = collection of related data, satisfying constraints
  - e.g. a student *is enrolled in* a course, a product *is sold at* a store
- *DBMS* = database management system
  - software to manage data, control access, enforce constraints
  - e.g. PostgreSQL, SQLite, Oracle, SQL Server, MySQL, ...

Studying 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 ...

- COMP9313: managing Big Data (but may not run in 2020)
- COMP9315: how to build relational DBMSs (write your own PostgreSQL)
- COMP9318: techniques for data mining (discovering patterns in DB)
- COMP9319: Web data compression and search (dealing with big text/Web data efficiently)
- COMP6714: information retrieval, web search (dealing with text data)

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 (Python, ORMs)

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

---

... Syllabus Overview

- Database management systems (DBMSs)
  - DBMS architecture: query processing, index structures
  - Transaction processing: transactions, concurrency control, recovery
- Future of Databases
  - Limitations of RDBMS's, potential future technologies

The green stuff is covered only briefly, and is not examinable

To learn more about the green stuff ... take other courses such as COMP9315

Your Background

We assume that you ...

- have experience with procedural programming
- have some background in data structures
- hopefully, have some knowledge of Python

You might have acquired this background in

- COMP1511, COMP1531, COMP2521

If you don't know Python, look at online tutorials ASAP.

Teaching/Learning

Stuff that is available for you:

- *Textbooks*: describe most syllabus topics in detail
- *Lectures*: cover all syllabus topics, with exercises
- *Theory exercises*: tutorial-type questions (+ solutions)
- *Prac exercises*: lab-like exercises
- *Assignments*: more detailed practical exercises
- *Tutorials and Labs*: provide assistance if you have questions in exercises or lecture materials

---

... Teaching/Learning

Scheduled classes?

- two **2-hour** lectures each week (Mon 10-12, Wed 10-12)

What to do if you have problems understanding stuff?

- ask a question during/after the lecture
- ask your tutor during the tutorial / lab
- come to a *consultation* (Mon 9-10, Wed 9-10)
- ask short questions on the *Forums* (under WebCMS3)
- send an email to the Lecturer for urgent matters

Forums and tute/lab are the best channels for asking short and long questions respectively.

... Teaching/Learning

12/62

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
- get your "short" questions answered (via the Forums)

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

Lectures

13/62

Lectures have three purposes:

- introduce content
- discuss practice
- work through some exercises (more so in the tutes)

Lectures are intended to be interactive, so ask questions

All lectures are recorded and available via Moodle

All exercise solutions are attached to the course website

Labs/Tutorials

14/62

Lectures/Tutorials have two purposes:

- offer assistance to issues that you have from the exercises / lecture materials
- work through exercises (mostly in the tutorials)

Labs run from Week 2 to 5 (4 labs).

Tutes run from Week 7 to 10 (4 tutes).

Labs and tutes are *not* compulsory

We expect that you will attempt the exercises (both Prac and Theory Exercises) in your own time. For example, if you do not have attempted the Prac Work and do not have problems setting up your database and complete the exercise, you do not need to attend the corresponding lab session.

Note: Week 9 Mon is a public holiday, so the last Monday tute will be on Week 11 Mon

---

Assignments

15/62

Two assignments, which are critical for *learning*

1. SQL and PLpgSQL, 20%, due end week 5
2. Python and SQL, 20%, due end week 9

Assignments are done *individually*, and ...

- submitted via `give` or Webcms3
- automarked (so you must follow specifcation exactly)
- plagiarism-checked (copying solutions ⇒ **0** mark for course)

- rent-a-coder monitored (buying solutions ⇒ **exclusion**)

... Assignments

16/62

Assignment 1: answer queries on a database

- we supply a PG database
- we give you a bunch of questions to solve
- you write SQL queries and PLpgSQL functions
- possibly include triggers
- auto-marking using supplied and new DB instances

... Assignments

17/62

Assignment 2: answer queries and manipulate data on a database

- we supply a PG or SQLitedatabase
- we give you a bunch of operations to build
- you write Python/psycopg2/SQL to solve
- auto-marking using supplied and new DB instances

Exam

18/62

3-hour **lab-based** exam during exam period

Comprising a mixture of

- SQL, PLpgSQL, Python, design exercises, analyses, theories

Prac part: SQL using SQLite (or maybe PostgreSQL)

All questions: typed in and submitted online

PG/SQL/Python documentation is accessible during exam

Sample exams will be available on the course website

Supplementary Assessment Policy

19/62

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 ... just before T2 (TBC).

Assessment Summary

20/62

Your final mark/grade will be determined as follows:

```
ass1    = mark for assignment 1      (out of 20)
ass2    = mark for assignment 2      (out of 20)
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 + exam

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

Textbook (options)

21/62

- 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 (7th ed, 2019)
- Kifer, Bernstein, Lewis
  Database Systems: Application-Oriented Approach (2nd ed, 2006)

Earlier editions of texts are ok

Database Management Systems

22/62

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

** Unless it seriously violates SQL standards ... I mean you, MySQL

Further Reading Material

23/62

The on-line documentation and manuals provided with:

- SQLite are reasonably good
- PostgreSQL are very good

- Python are similarly comprehensive

Some comments on technology 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

24/62

Software versions that we'll be running this semester (TBC):

- PostgreSQL 11.3, SQLite 3.8, Python 3.7, psycopg2 2.8

If you install them at home:

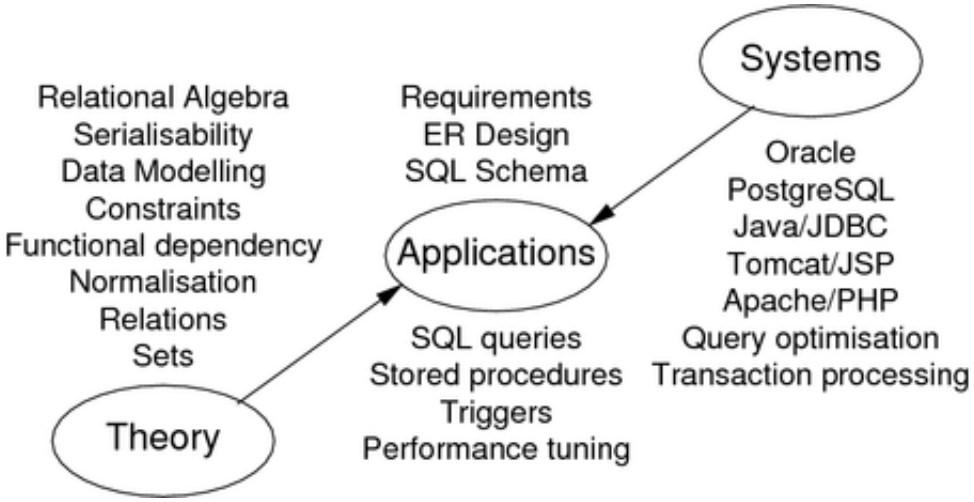
- get versions "close to" these
- test all work at CSE before submitting

Alternative to installing at home:

- run them on the CSE servers (grieg) as you would in labs
- use `vlab` to log in to a CSE server from home

Overview of the Databases Field

25/62



Database Application Development

26/62

A variation on standard software engineering process:

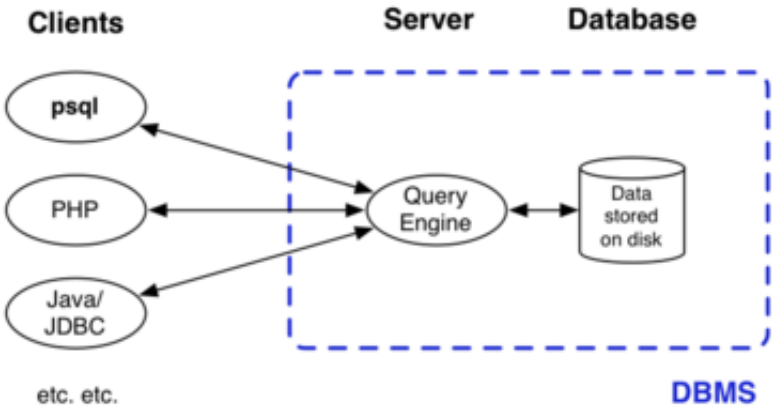
1. analyse application requirements
2. develop a data model to meet these requirements
3. define operations (transactions) on this model
4. implement the data model as relational schema
5. implement transactions via SQL and procedural PLs
6. construct an interface to these transactions

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

Database System Architecture

27/62

The typical environment for a modern DBMS is:



SQL queries and results travel along the client↔server links

Data Modelling

Data Modelling

29/62

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, Tom's account is held at Coogee)
- describe *constraints* on data
  (e.g., 7-digit IDs, students can enrol in no more than 3 courses per term)

Data modelling is a *design* process

- converts requirements into a data model

... Data Modelling

30/62

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

31/62

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

32/62

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

Quality of Designs

33/62

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

35/62

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

36/62

ER diagrams are a graphical tool for data modelling.

An ER diagram consists of:

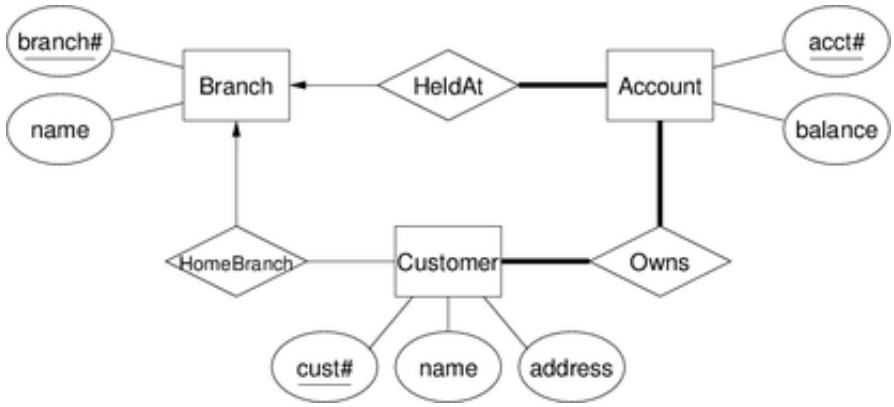
- a collection of *entity set* definitions
- a collection of *relationship set* definitions
- *attributes* associated with entity and relationship sets
- connections between entity and relationship sets

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

... Entity-Relationship (ER) Diagrams

37/62

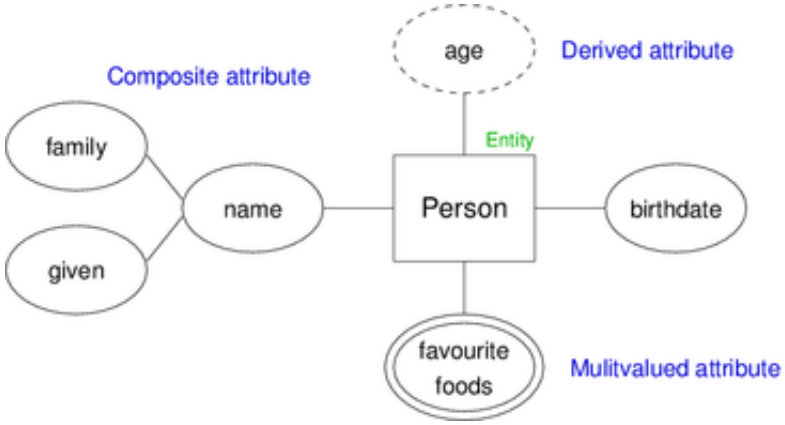
Example ER diagram:



... Entity-Relationship (ER) Diagrams

38/62

Example of attribute notations:



Entity Sets

39/62

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)

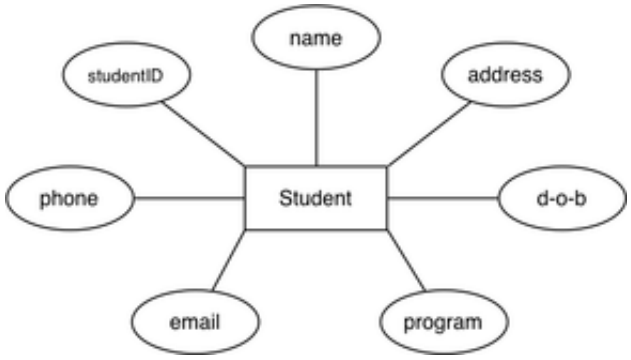
Primary key = candidate key chosen by DB designer

Keys are indicated in ER diagrams by underlining

Exercise 2: Keys #1

40/62

Identify candidate keys in the following ER diagram

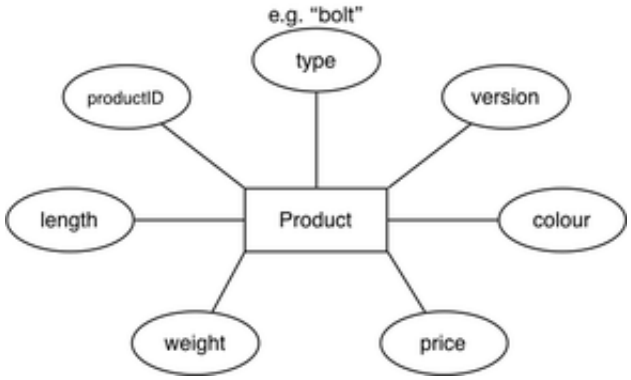


State any assumptions about the attribute types

Exercise 3: Keys #2

41/62

Identify candidate keys in the following ER diagram



State any assumptions about the attribute types

Relationship Sets

42/62

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

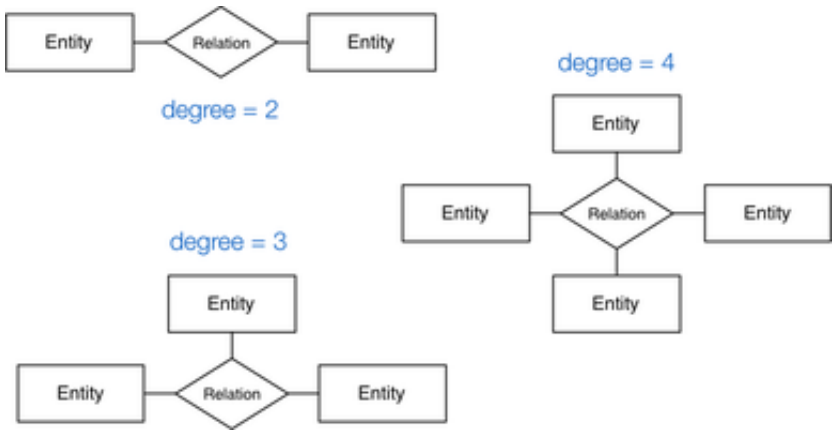
Participation = must every entity be in the relationship

Example: relationship participation

... Relationship Sets

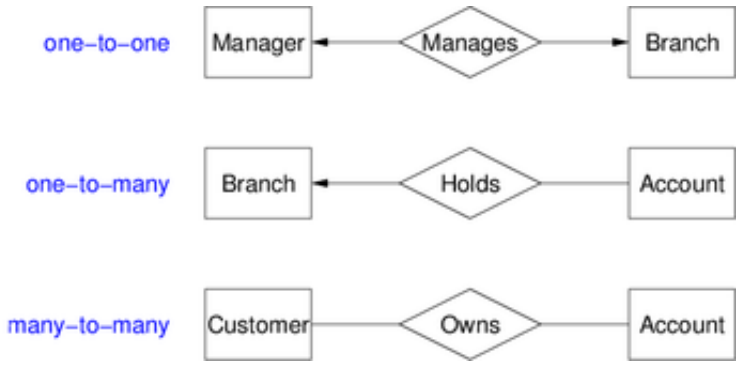
43/62

Examples: relationship degree



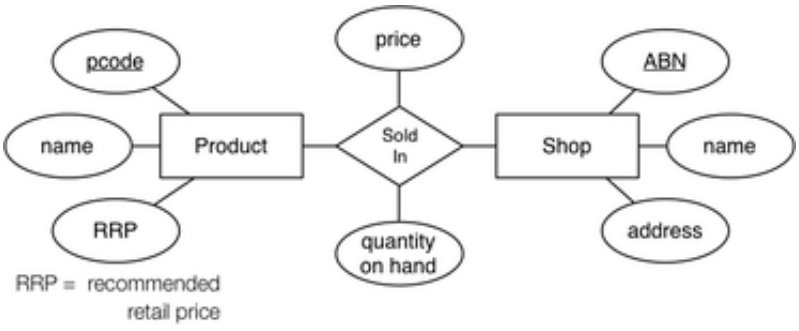
... Relationship Sets

Examples: relationship cardinality



... Relationship Sets

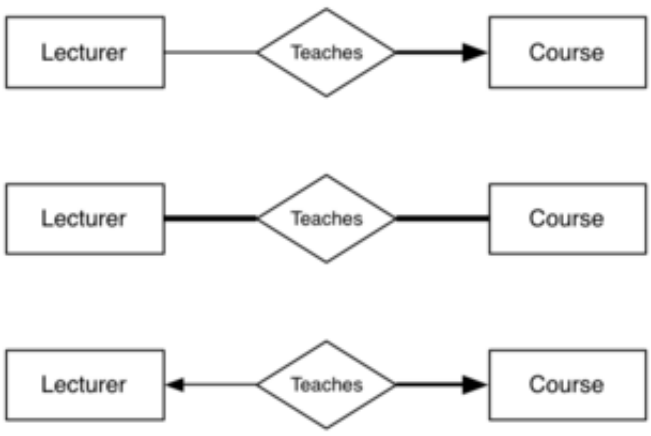
In some cases, a relationship needs associated attributes.



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

Exercise 4: Relationship Semantics

Describe precisely the semantics of the following relationships:

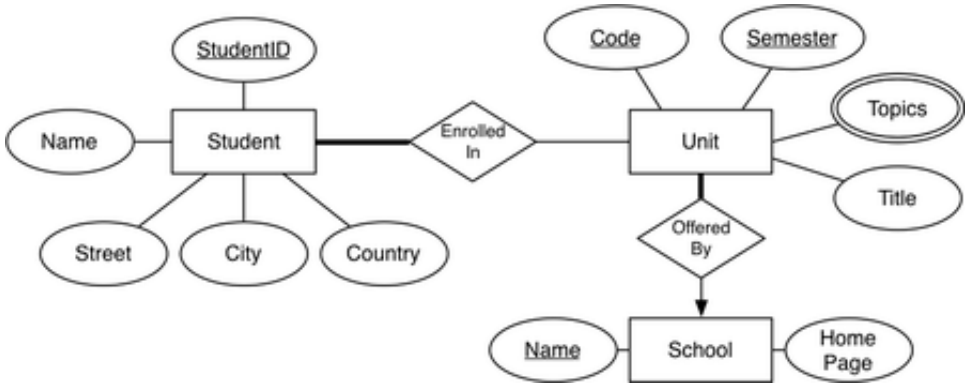


Exercise 5: ER Diagram

47/62

Using the ER diagram below

- give examples of entity values
- describe the semantics of the relationships



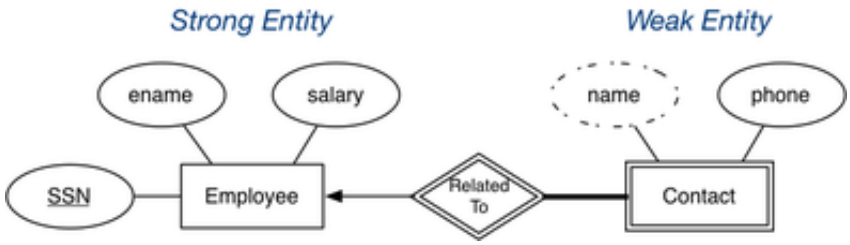
Weak Entity Sets

48/62

Weak entities

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

Example:



Subclasses and Inheritance

49/62

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

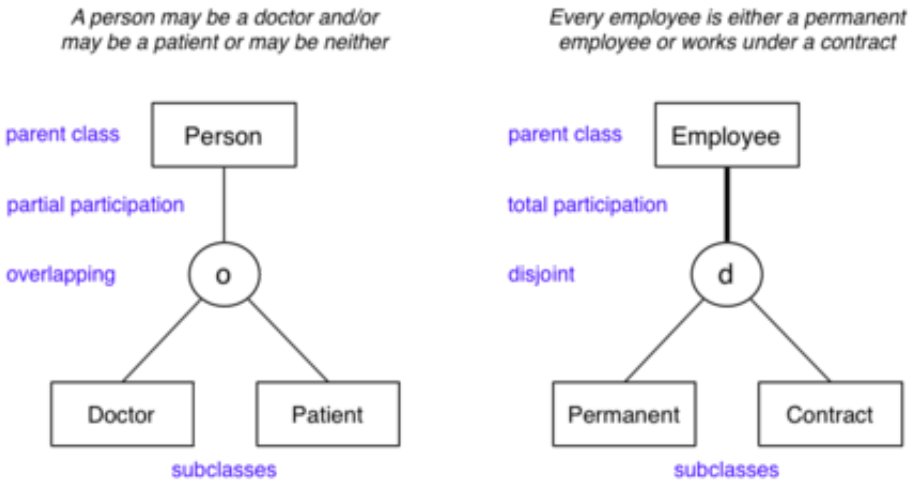
Properties of subclasses:

- *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 6: 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 7: 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.

Exercise 8: Book Publishing

Develop an ER design for the following scenario:

- for each person, we need to record their tax file number (TFN), their real name, and their address
- authors write books, and may publish books using a "pen-name" (a name, different to their real name, which they use as author of books)
- editors ensure that books are written in a manner that is suitable for publication
- every editor works for just one publisher
- editors and authors have quite different skills; someone who is an editor cannot be an author, and vice versa
- a book may have several authors, just one author, or no authors (published anonymously)
- every book has one editor assigned to it, who liaises with the author(s) in getting the book ready for publication
- each book has a title, and an edition number (e.g. 1st, 2nd, 3rd)
- each published book is assigned a unique 13-digit number (its ISBN); different editions of the same book will have different ISBNs
- publishers are companies that publish (market/distribute) books
- each publisher is required to have a unique Australian business number (ABN)
- a publisher also has a name and address that need to be recorded
- a particular edition of a book is published by exactly one publisher

---

Summary of ER

ER model is popular for doing conceptual design

- high-level, models relatively easy to understand
- good expressive power, can capture many details

Basic constructs: *entities, relationships, attributes*

Relationship constraints: *total / partial, n:m / 1:n / 1:1*

Other constructs: *inheritance hierarchies, weak entities*

Many notational variants of ER exist
(especially in the expression of constraints on relationships)

Relational Data Model

Relational Data Model

The *relational data model* describes the world as

- a collection of inter-connected *relations* (or *tables*)

Goal of relational model:

- a simple, general data modelling *formalism*
- which maps easily to file structures (i.e. *implementable*)

Relational model has two styles of terminology:

- mathematical: relation, tuple, attribute, ...
- data-oriented: table, record, field/column, ...

Warning: textbooks alternate between the two; treat them as synonyms.

... Relational Data Model

The relational model has one structuring mechanism ...

- a *relation* corresponds to a mathematical "relation"
- a *relation* can also be viewed as a "table"

Each *relation* (denoted *R,S,T,...*) has:

- a *name* (unique within a given database)
- a set of *attributes* (which can be viewed as column headings)

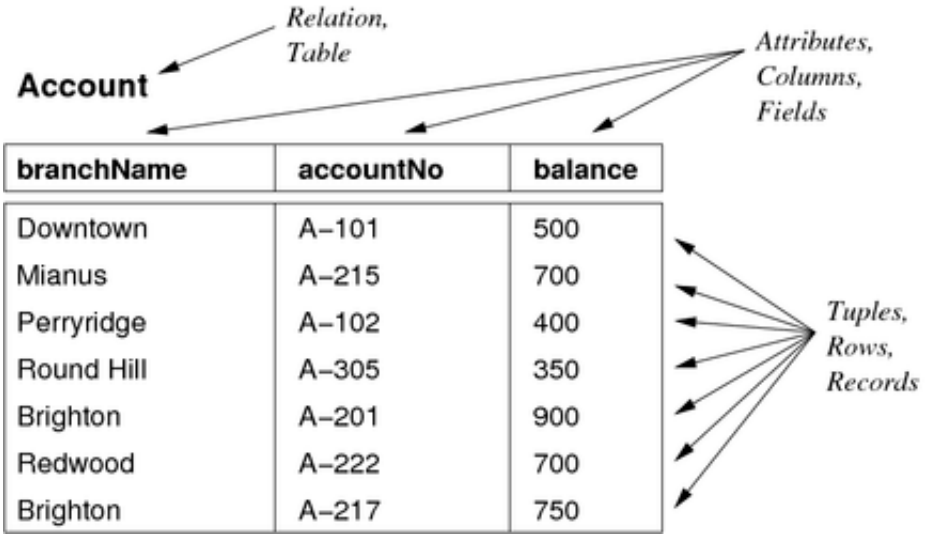
Each *attribute* (denoted *A,B,...* or $a_1,a_2,...$) has:

- a *name* (unique within a given relation)
- an associated *domain* (set of allowed values)

DB definition also uses *constraints* (logic expressions)

... Relational Data Model

Example relation (bank accounts):



branchName	accountNo	balance
Downtown	A–101	500
Mianus	A–215	700
Perryridge	A–102	400
Round Hill	A–305	350
Brighton	A–201	900
Redwood	A–222	700
Brighton	A–217	750

... Relational Data Model

61/62

Points to note ...

Attribute values are *atomic* (no composite or multi-valued attributes).

A distinguished value `NULL` belongs to all domains.

- `NULL` has several interpretations: none, don't know, irrelevant

Each relation has a *key* (subset of attributes unique for each tuple)

... Relational Data Model

62/62

Consider relation *R* with attributes $a_1, a_2, \ldots a_n$

Relation schema of *R* : $\mathbf{R}(a_1{:}D_1, a_2{:}D_2, \ldots a_n{:}D_n)$

Tuple of *R* : an element of $D_1 \times D_2 \times \ldots \times D_n$ (i.e. list of values)

Instance of *R* : subset of $D_1 \times D_2 \times \ldots \times D_n$ (i.e. set of tuples)

Database schema : a collection of relation schemas.

Database (instance) : a collection of relation instances.

Produced: 16 Feb 2020