Homework from Tuesday

Viewpoint | Jeannette M. Wing

Computational Thinking
It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking breaks on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational thinking involves the ability to solve problems and design systems that are out of our reach. Computational thinking requires the ability to solve problems and design systems that are out of our reach. Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science. Having to solve a particular problem, we might ask: How difficult is it to solve? And what is the best way to solve it? Computer science relies on and builds on the core principles to answer such questions.

Computational thinking is thinking computationally. It is parallel processing. It is interpreting code as data and data as code. It is thinking (or the centralization of computational analysis. It is recognizing both the macro and the micro as algorithms, or getting someone or something more than one slide. It is recognizing the role and power of data and database processing and procedure calls. It is judging a program not just for correctness and efficiency but also for aesthetics, and its impact for simplicity and elegance.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science.

See also https://is.gd/channel9wing

Jeannette Wing

“Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science”
Course website and blog
https://blog.inf.ed.ac.uk/da17

Piazza discussion group Inf1-DA

Mailing list inf1-da-students@inf.ed.ac.uk
Posting is restricted to list members; send from your University email address.

Facebook group inf1da2017

Twitter @inf1da17
Tutorials and Exercise Sheets

Tutorials start in **Week 3** of semester and continue each week until the end of semester, except for Flexible Learning Week which falls between lecturing weeks 5 and 6.

Your *course tutor* leads your tutorial group; this is not the same as your *personal tutor*. Group membership, times and study rooms will be confirmed during Week 2: at the moment they are still changing as we complete the allocation.

If you wish to move to a different tutorial group then you will be able to do so using the Informatics student portal:

https://www.student.inf.ed.ac.uk

Studying for Inf-DA requires you to attend the weekly tutorials. If you are ill or otherwise unable to attend one week then email your course tutor, and if possible attend another tutorial group in the same week.
Structured Data

- e.g. University database of students, staff, courses, rooms, etc.
- Which students take Inf1-DA? How do we timetable exams?

Semistructured Data

- e.g. Tourist factbook about countries, regions, cities, ...
- e.g. BioModels repository of computational models for biochemical reactions and metabolic pathways.

Unstructured Data

- e.g. British National Corpus of spoken and written English.
Structured Data

Some application domains involve handling quantities of data that can be very strictly organised. For example:

- The University of Edinburgh records some standardized pieces of information about each of several thousand students.
- A supermarket chain will maintain information on tens of thousands of product lines, and the stock in each shop where they are sold.
- A web browser will keep details of passwords and preferences for all websites a user visits.

What’s central to this structure is that we are working with the same information about many different individuals. Even when there are different kinds of individual (product lines, shops, staff, ...) there are far more items of each kind than there are different kinds.
Structured Data

Some application domains involve handling quantities of data that can be very strictly organised. For example:

- The University of Edinburgh records some standardized pieces of information about each of several thousand students.
- A supermarket chain will maintain information on tens of thousands of product lines, and the stock in each shop where they are sold.
- A web browser will keep details of passwords and preferences for all websites a user visits.

As well as individuals or entities, it’s usually important to also work with the relationship between individuals: which students take which course, or which shop stocks which product.
Structured Data

Some application domains involve handling quantities of data that can be very strictly organised. For example:

- The University of Edinburgh records some standardized pieces of information about each of several thousand students.
- A supermarket chain will maintain information on tens of thousands of product lines, and the stock in each shop where they are sold.
- A web browser will keep details of passwords and preferences for all websites a user visits.

It turns out — perhaps unexpectedly — to be very effective to concentrate more on the relations between things than on the things themselves.

“The fundamental interconnectedness of all things”
Douglas Adams, Dirk Gently’s Holistic Detective Agency
Lecture Plan for Weeks 1–4

Data Representation

This first course section starts by presenting two common data representation models.

- The entity-relationship (ER) model
- The relational model

Note slightly different naming: -relationship vs. relational

Data Manipulation

This is followed by some methods for manipulating data in the relational model and using it to extract information.

- Relational algebra
- The tuple-relational calculus
- The query language SQL
1 Requirements analysis
Understand what data is to be stored in the database and what
operations on it are likely to be needed.

2 Conceptual design
Develop a high-level description of data to be stored, and the
constraints that apply to it.
This is the level where we might use an ER data model.

3 Logical design
Implement the conceptual design by mapping it to a specific data
representation. The outcome is a logical schema.
For example, implementation can be performed by translating the ER
data model into a relational data model.
The ER Data Model

- **What is it?**
  The ER model is a way to organise description of *entities* (individual things in the real world) and the *relationships* between them.

- **Why is it useful?**
  It readily maps into different *logical data models*, such as the relational model.

- **How is it used?**
  As a graphical notation for visualizing the structure of data, to clarify and communicate that structure.

---

*P. P. Chen*

The Entity-Relationship Model – Toward a Unified View of Data.

An *entity* is any kind of thing that we want to model in our database. For example, a university database might be designed to capture notions of *course* or *student*.

Within this, each individual item is an *entity instance*; and the collection of all such items is an *entity set*.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Course</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity instance</strong></td>
<td>Inf1-DA 2016/17</td>
<td>A. Lovelace</td>
</tr>
<tr>
<td><strong>Entity set</strong></td>
<td>Edinburgh courses</td>
<td>Edinburgh students</td>
</tr>
</tbody>
</table>
Entity-relationship modelling provides a graphical language for describing entities and the relationships between them in an *ER diagram*.

The ER diagram syntax for an entity is a rectangle, labelled with the kind of entity it represents.
Attributes

An entity is described by its characteristic *attributes*. These are the properties to be captured in the data model.

An ER diagram shows attributes as ovals, labelled with the attribute’s name, connected to the appropriate entity.
Each attribute has a *domain* of allowed values, similar to the use of types in Haskell or Java.

For example, *Age* could have domain “positive integer”, while the domain for *email* might be “strings of up to 254 characters”.

```
Student
UUN   Age   email

Course
Code
Title  Year
```
Attributes list the information recorded in a model for each entity instance. Attributes are also used to identify entity instances and distinguish between them. This is the role of a key: a chosen set of attributes.
A set of attributes is a \textit{superkey} for an entity if those attributes, taken together, always uniquely identify every entity instance.

A set of attributes is a \textit{key} if it is a minimal set of identifying attributes — removing any one attribute would make it no longer uniquely identifying.

<table>
<thead>
<tr>
<th>Building Number</th>
<th>Street Name</th>
<th>Town</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Superkey</strong></td>
<td>BN</td>
<td>SN</td>
<td>T</td>
</tr>
<tr>
<td><strong>Superkey</strong></td>
<td>BN</td>
<td>SN</td>
<td></td>
</tr>
<tr>
<td><strong>Superkey</strong></td>
<td>BN</td>
<td>SN</td>
<td>T</td>
</tr>
<tr>
<td><strong>Superkey</strong></td>
<td>BN</td>
<td></td>
<td>PC</td>
</tr>
<tr>
<td></td>
<td>BN</td>
<td>SN</td>
<td></td>
</tr>
</tbody>
</table>

Appleton Tower:
11 Crichton Street
Edinburgh EH8 9LE
Again: a key is a **minimal set of attributes** whose values uniquely identify each entity instance in an entity set.

Where there is more than one such attribute set, each is a *candidate key*.

From all the candidates keys we choose a *primary key* to be used as the unique identifier for the entity.

<table>
<thead>
<tr>
<th>Building Number</th>
<th>Street Name</th>
<th>Town</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superkey BN</td>
<td>SN</td>
<td>T</td>
<td>PC</td>
</tr>
<tr>
<td>Superkey BN</td>
<td>SN</td>
<td></td>
<td>PC</td>
</tr>
<tr>
<td>Superkey BN</td>
<td>SN</td>
<td>T</td>
<td>Key</td>
</tr>
<tr>
<td>Superkey BN</td>
<td>PC</td>
<td>Key</td>
<td>Candidate Key</td>
</tr>
<tr>
<td>Superkey BN</td>
<td>PC</td>
<td>Key</td>
<td>Candidate Key</td>
</tr>
<tr>
<td>Superkey BN</td>
<td>SN</td>
<td>PC</td>
<td>Primary Key</td>
</tr>
</tbody>
</table>

Appleton Tower:
11 Crichton Street
Edinburgh EH8 9LE
Primary Keys

Any key is a set of attributes: it may contain one, two, or more attributes. A key made of more than one attribute is a composite key. The ER diagram syntax underlines each attribute that is part of the primary key.
Addresses

Inspace Media Laboratory
1 Crichton Street, Edinburgh

Informatics Forum
10 Crichton Street, Edinburgh

Appleton Tower
11 Crichton Street, Edinburgh

Picture credit: Google Maps
Email Addresses

Bad News

The string s0412375@ed.ac.uk is not an email address. Really, it’s not.
It’s a username to log in to Office365.
Yes, it does look very like an email address, and within Office365 it may even work as an address.
But, it doesn’t work as an email address out there on the internet.

s0412375@sms.ed.ac.uk is an email address (sms = Student Mail Service)

Good News

You have another email address at the University, it’s based on your name, and it’s almost certainly more readable than s0412375@sms.ed.ac.uk

To find it, follow “Find Students/Staff” on the MyEd Dashboard, select “student”, and enter your student ID.
A **superkey** is any set of attributes whose values uniquely identify each entity instance in an entity set.

A **key** is a minimal set of attributes whose values uniquely identify each entity instance in an entity set.

Where there is more than one such set, each forms a **candidate key**.

Any key with more than one attribute is a **composite key**.

One of the candidate keys is selected as the **primary key**.

In an ER diagram each attribute in the primary key is **underlined**.
A *relationship* is an association between entities. For example, the *takes* relationship between students and courses.

Each individual occurrence of the relationship is a *relationship instance*, and the collection of all such is a *relationship set*.

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Takes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Relationship instance</strong></td>
<td>((s0456782), (INF08013, 2016))</td>
</tr>
<tr>
<td><strong>Relationship set</strong></td>
<td>Edinburgh course registrations</td>
</tr>
</tbody>
</table>
ER diagrams show relationships as diamonds, labelled with the name of the relationship and connected to all the participating entities.

A relationship may also have its own attributes.
Relationships

Relationships can be between two entities ("binary"), three ("ternary") or more ("n-ary").

Relationship instance  (ASDA, Heinz, Ketchup, Edinburgh, £1.50)
There is no bound on the number of entities participating in a relationship. An entity may be involved in any number of different relationships.
The **entity-relationship** (ER) model is a way to organise the description of *entities* (individual things) and the *relationships* between them.

So far we have seen the following elements of ER modelling:

- **Entities**, all with characteristic **attributes**;
- A set of attributes making a **key** to identify and distinguish entities;
- **Relationships** between entities.

Entity-relationship modelling provides a *graphical language* for describing this structure in an **ER diagram**.
Homework

Read This

Sections 2.1–2.4 of this textbook on databases: PDF circulated by email; paper copy available from me or outside ITO.


This is the recommended textbook for the third-year course *Database Systems*. It’s a large book, with thorough and extensive material on a wide range of database topics.

It is *not* necessary to buy this book for Inf1-DA. Instead:

Do This

Step inside the library, locate the HUB and find this book. Have a look at the other textbooks there, too, and compare style and content.