

Informatics 1: Data & Analysis

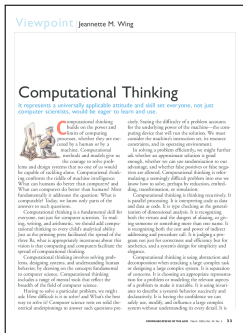
Lecture 2: Entities and Relationships

Ian Stark

School of Informatics
The University of Edinburgh

Thursday 17 January 2019
Semester 2 Week 1

Homework from Tuesday



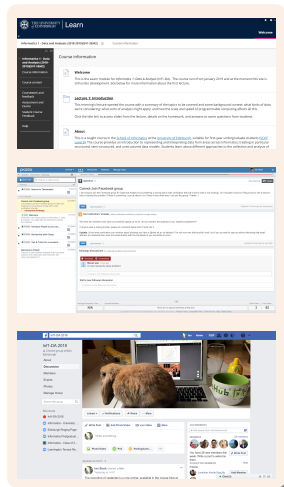
Picture credit: Carnegie Mellon University

Jeannette Wing

“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>

- Course website on Learn <https://course.inf.ed.ac.uk/inf1-da>
- 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 [inf1da2019](#)



This is not a textbook course, and there is no single compulsory book.

For certain parts of the course, however, I shall indicate one or more books which cover the current material — usually in much more depth and generality than required for this introductory course.

You can consult these books in the library, or borrow them, and you may find one or other helpful to you. Although the content is often similar, styles and tastes can differ significantly.

Occasionally I shall distribute PDF and photocopies of an individual textbook chapter when it is especially relevant to the course.

Tutorials start in **Week 3** of semester and continue each week until the end of semester, except for during the *Festival of Creative 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*.

You will be able to choose your own tutorial group during Week 2: all tutorials will be one-hour time-slots on Monday, Tuesday or Wednesday.

Studying for Inf-DA requires you to work through exercise sheets and then take part in your 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 may keep details of usernames, passwords, and preferences for the 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 may keep details of usernames, passwords, and preferences for the 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 may keep details of usernames, passwords, and preferences for the 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*

Database Design

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 any *constraints* that might apply to the data.

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. *ACM Transactions on Database Systems* 1(1):9–36.

Entities and Entity Sets

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*.

Entity	Student	Course
Entity instance	A. Lovelace	Inf1-DA 2018/19
Entity set	Edinburgh students	Edinburgh courses

Entity-Relationship Diagrams

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.



```
graph LR; Student[Student]; Course[Course];
```

Student

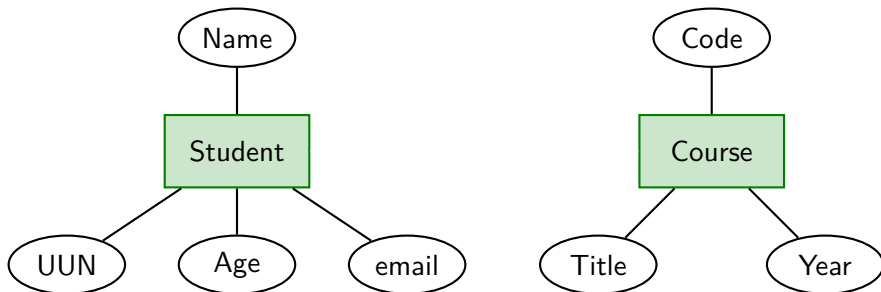
Course

Attributes

An entity is described by its characteristic *attributes*.

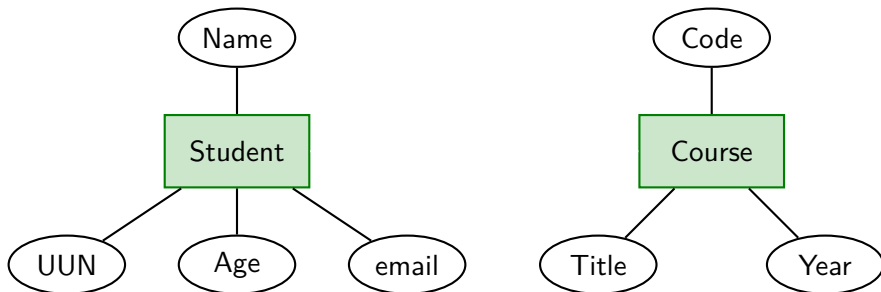
These are the properties to be captured in the data model.

An ER diagram shows attributes as ovals, labelled with the attribute's name, connected to the appropriate entity.



Domains

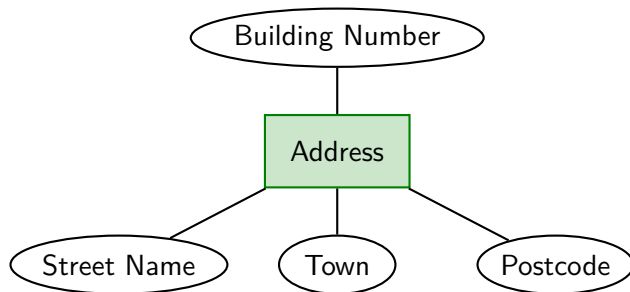
Each attribute has a *domain* of allowed values, similar to the use of types in Haskell or Java. For example, **Age** could have domain “positive integers”, while the domain for **email** might be “strings of up to 254 characters”.



Keys

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**.



Keys

A set of attributes is a *superkey* for an entity if those attributes, taken together, always uniquely identify every entity instance.

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

	Building Number	Street Name	Town	Postcode
Superkey	BN	SN	T	PC
Superkey	BN	SN		PC
Superkey	BN	SN	T	
Superkey	BN			PC
	BN	SN		
				PC

Appleton Tower: 11 Crichton Street
EDINBURGH
EH8 9LE

Key
Key

Primary Keys

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.

	Building Number	Street Name	Town	Postcode			
Superkey	BN	SN	T	PC			
Superkey	BN	SN		PC			
Superkey	BN	SN	T		Key	Candidate Key	
Superkey	BN			PC	Key	Candidate Key	Primary Key
	BN	SN					
				PC			

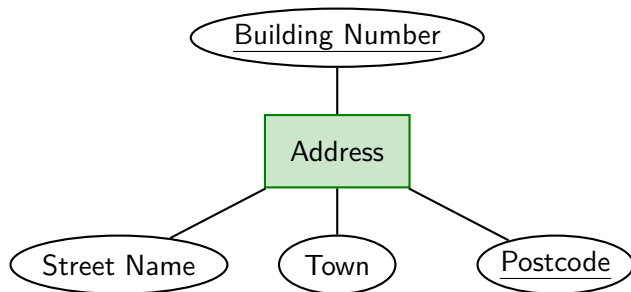
Appleton Tower: 11 Crichton Street
EDINBURGH
EH8 9LE

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 chosen primary key.



Summary: Keys

- 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**.

Relationships

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*.

Relationship

Takes

Relationship instance

((s0456782), (INF08013, 2016))

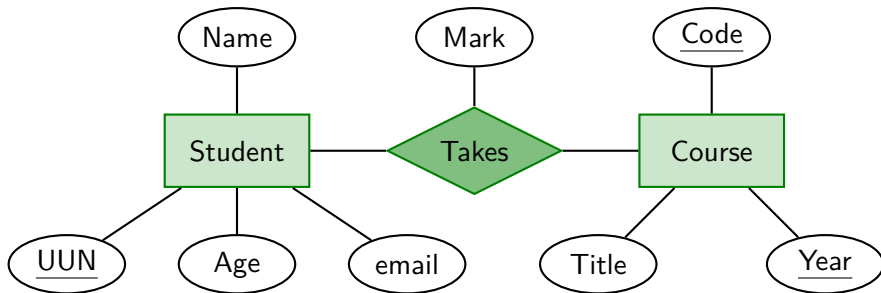
Relationship set

Edinburgh course registrations

Relationships

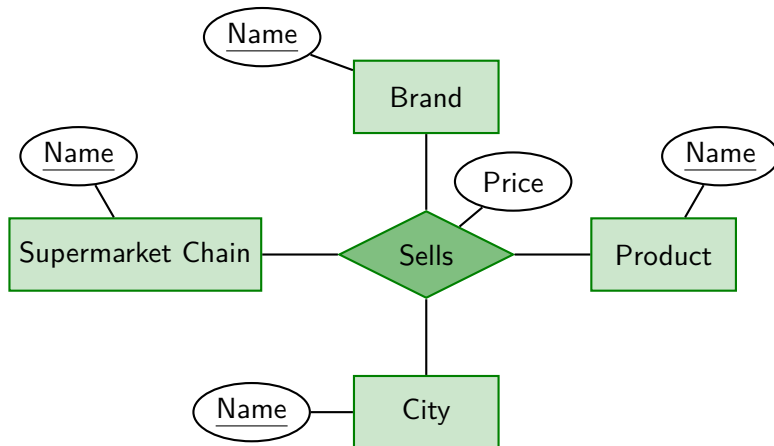
ER diagrams show relationships as diamonds, labelled with the name of the relationship and connected to all the participating entities.

A relationship may also have its own attributes.



Relationships

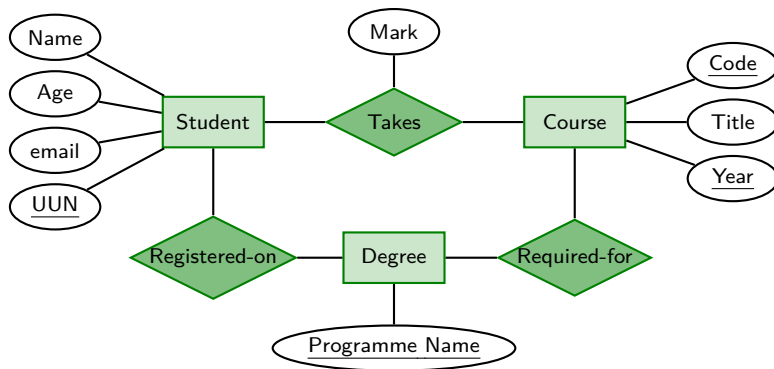
Relationships can be between two entities (“**binary**”), three (“**ternary**”) or more (“**n-ary**”).



Relationship instance (ASDA, Heinz, Ketchup, Edinburgh, £2)

Relationships

There is no bound on the number of entities participating in a relationship.
An entity may be involved in any number of different relationships.



Summary

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**;
- For each kind of entity there are **entity instances** that together make up an **entity set**;
- A **key** as a minimal set of attributes to identify and distinguish entity instances;
- **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: you will need to log in to *Learn* to access this chapter as a PDF.



R. Ramakrishnan and J. Gehrke.
Database Management Systems.
McGraw-Hill, third edition, 2003.

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.

Finally: Find the video of *this lecture* online and watch the first minute. If you have any problems then ask on Piazza/Facebook.