

Informatics 1: Data & Analysis

Lecture 2: Entities and Relationships

Ian Stark

School of Informatics
The University of Edinburgh

Friday 15 January 2016
Semester 2 Week 1



THE UNIVERSITY
of EDINBURGH

<http://www.inf.ed.ac.uk/teaching/courses/inf1/da>

TopHat Question

Three Notable Computer Scientists



Karen Sparck-Jones



Jeannette Wing



Margaret Hamilton

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

Who wrote the article on this that was your Inf1-DA reading for today?

TopHat Question

Three Notable Computer Scientists



Karen Sparck-Jones



Jeannette Wing

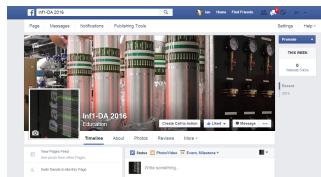


Margaret Hamilton

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

Who wrote the article on this that was your Inf1-DA reading for today?

- Course website and blog <https://blog.inf.ed.ac.uk/da16>
- Piazza discussion group **Inf1-DA**
- Mailing list inf1-da-students@inf.ed.ac.uk
Posting is restricted to list members; use your University email address.
- Facebook page **inf1da16**
- Facebook group **UofE Computer Science (Informatics) Class 2019/2020**
- Twitter **@inf1da16**



Homework

Read This

Sections 2.1–2.4 of the handout, which is from this textbook on databases.



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.

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**ship** 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 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.
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	Course	Student
Entity instance	Inf1-DA 2015/2016	A. Lovelace
Entity set	Edinburgh courses	Edinburgh students

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

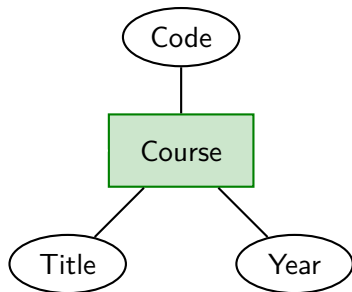
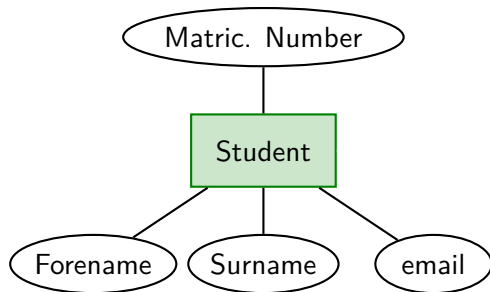
Course

Attributes

An entity is described by its characteristic *attributes*.

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

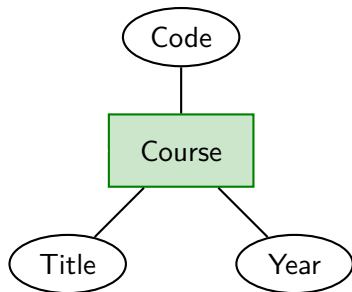
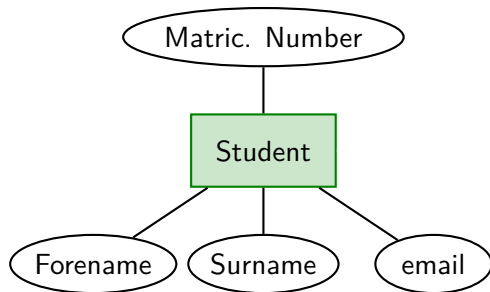
An ER diagram shows attributes as ovals, labelled with the attribute's name, connected to the appropriate entity.



Domains

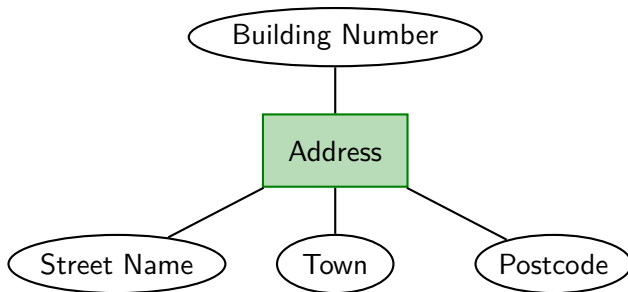
Each attribute has a *domain* of allowed values, similar to the use of types in Haskell or Java.

For example, **Matric. Number** has domain “positive integer”, 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
BN	SN	T	PC
BN	SN		PC
BN	SN	T	
BN			PC
BN	SN		
			PC

Appleton Tower:
11 Crichton Street
Edinburgh EH8 9LE

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

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		Key
Superkey	BN			PC	Key
	BN	SN			
				PC	

Appleton Tower:
11 Crichton Street
Edinburgh EH8 9LE

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		Key
Superkey	BN			PC	Key
Not a key	BN	SN			
Not a key				PC	

Appleton Tower:
11 Crichton Street
Edinburgh EH8 9LE

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
Superkey	BN			PC	Key
Not a key	BN	SN			
Not a key				PC	

Appleton Tower:
11 Crichton Street
Edinburgh EH8 9LE

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		
	BN	SN	T	PC		
Superkey	BN	SN		PC		
Superkey	BN	SN	T		Key	Candidate Key
Superkey	BN			PC	Key	Candidate Key
Not a key	BN	SN				
Not a key				PC		

Appleton Tower:
11 Crichton Street
Edinburgh EH8 9LE

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
Not a key	BN	SN					
Not a key				PC			

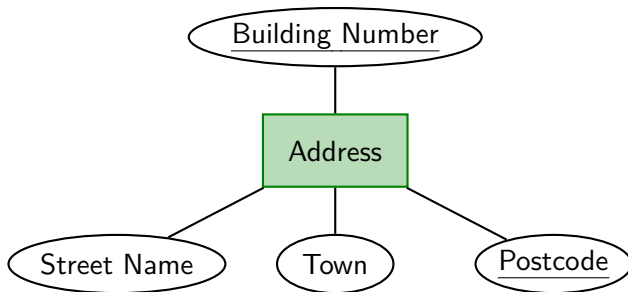
Appleton Tower:
11 Crichton Street
Edinburgh EH8 9LE

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 primary key.



Inspace Media Laboratory

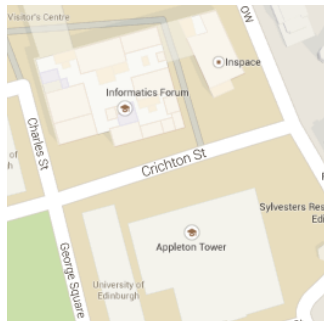
1 Crichton Street, Edinburgh

Informatics Forum

10 Crichton Street, Edinburgh

Appleton Tower

11 Crichton Street, Edinburgh



Picture credit: Google Maps

Bad News

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 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.

TopHat Questions

Title Informatics 1: Functional Programming

Code INFR08013

Year 2015/2016

Course Title	Course Code	Year
T	C	Y
T	C	
T		Y
	C	
	C	Y

TopHat Questions

Title Informatics 1: Functional Programming

Code INFR08013

Year 2015/2016

Course Title	Course Code	Year		
T	C	Y	Superkey	
T	C			
T		Y	Superkey	Key
	C			
	C	Y	Superkey	Key

Summary: Keys

- A *superkey* is a **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**.