Informatics 1B, 2008

School of Informatics, University of Edinburgh

Data and Analysis

Note 1

The Entity-relationship Data Model

Alex Simpson

Data and Analysis module: logistics

Lecturer: Alex Simpson `<Alex.Simpson@ed.ac.uk>`

Teaching Assistant: Laura Hutchins-Korte `<L.Korte@sms.ed.ac.uk>`

Weekly tutorials on Tuesdays/Wednesdays, weeks 3–11.

Weekly exercise sheets to be completed for tutorials.

Coursework assignment handed out Thurs week 8, due Fri week 9.

Assessment by means of 2-hour written exam during exam period

This exam contributes 50% of your Inf1B marks

Thanks Course structure and slides developed from material by Stratis Viglas, Frank Keller and Helen Pain.

Data and Analysis module: overview

The course concerns mechanisms for representing, manipulating and analysing different forms of data.

Course divided in three parts:

(I) Structured data (lectures 1–6, approx)

(II) Semi-structured data (lectures 7–11, approx)

(III) Unstructured data (lectures 12–15, approx)

Part I — Structured data

- For some application domains, data is *inherently structured*
  - For instance, all students share common information
- In such domains, it makes sense to organise the data in a way that directly maps to their *physical properties*, and to devise mechanisms to access and manipulate data
- We will deal with two main *data representation* models:
  - The *entity-relationship (ER)* model, and the *relational* model
- Finally, we will deal with data *manipulation* for the *relational model*, in particular:
  - *Relational algebra*, the *Tuple-relational calculus* and the query language *SQL*

Part I — Structured Data

Data Representation:

Note 1 The entity-relationship (ER) data model

Note 2 The relational model

Data Manipulation:

Note 3 Relational algebra

Note 4 Tuple relational calculus

Note 5 The SQL query language

Initial stages of database design

1. Requirements analysis.

   Understand what data is to be stored in the database and what operations are likely to be performed on it.

2. Conceptual design

   Develop a high-level description of data to be stored and constraints that hold over it.

   This description is often given using the ER data model.

3. Logical design

   Implement the conceptual design by mapping it to a *logical data representation*. The outcome is a *logical schema*.

   The implementation is often performed by translating the ER data model into a *relational database schema* (see Note 2).

The ER data model

- What is it used for?

  The ER model is a way to describe *entities* (for example, real-world entities) and the *relationships* between them

- Why is it useful?

  Because it maps to different *logical data models*, including the relational model

- How is it used?

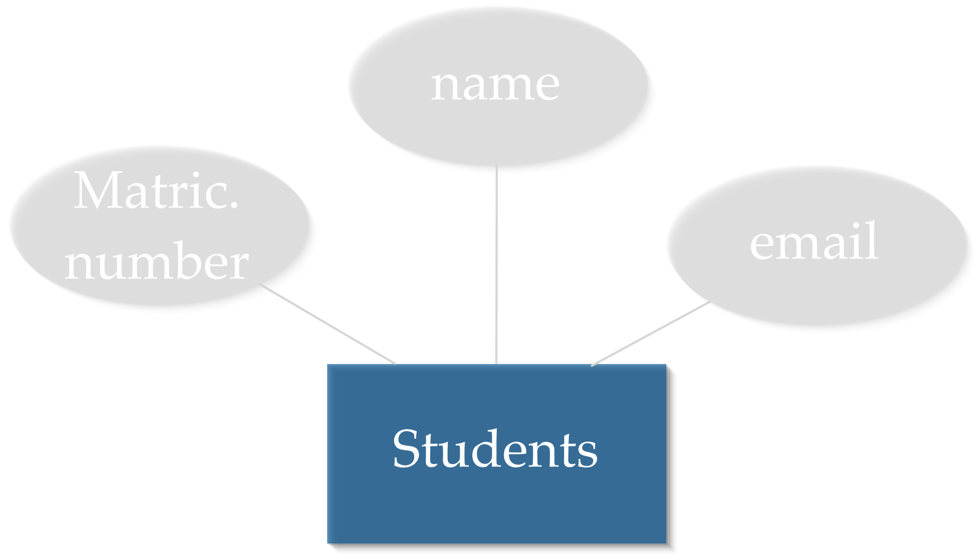
  It is essentially a way to visualise data and their dependencies

Entities and entity sets

Any distinguishable object (for example, in the real world) can be an *entity*

A collection of the same type of entities is an *entity set*

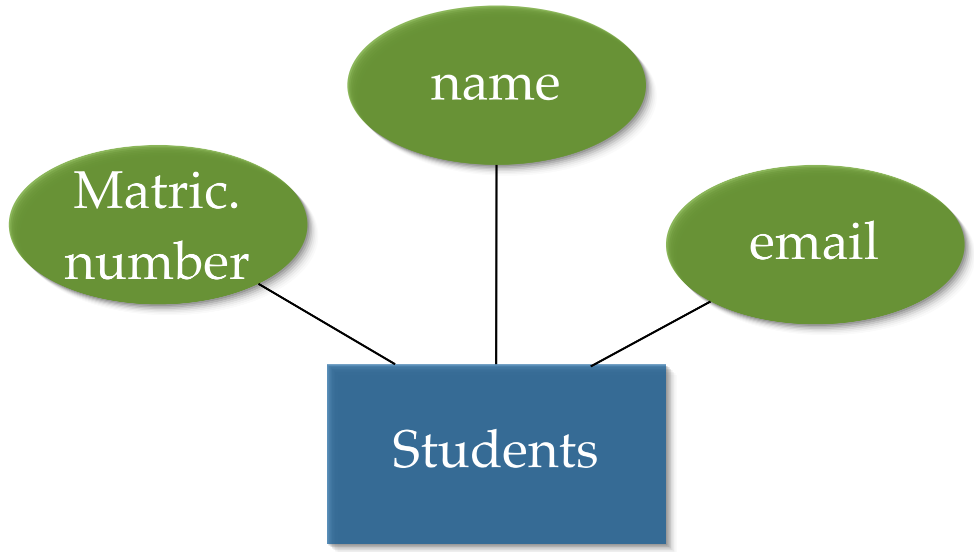
Entity sets are represented by *boxes*, labelled with the entity set's name



Attributes

Each entity of the same entity set has some characteristic *attributes*

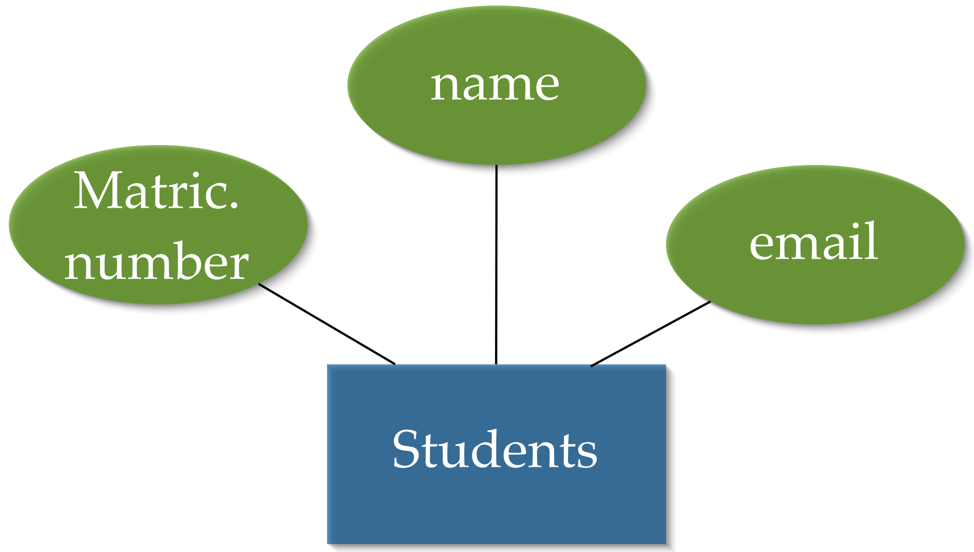
Attributes are represented by *ovals*, labelled with the attribute's name, connected to the entity set they belong to.



Domains

Each attribute has a *domain* from which allowable values are derived

E.g., Matric. number is an *integer*
name and email are *40-character strings*

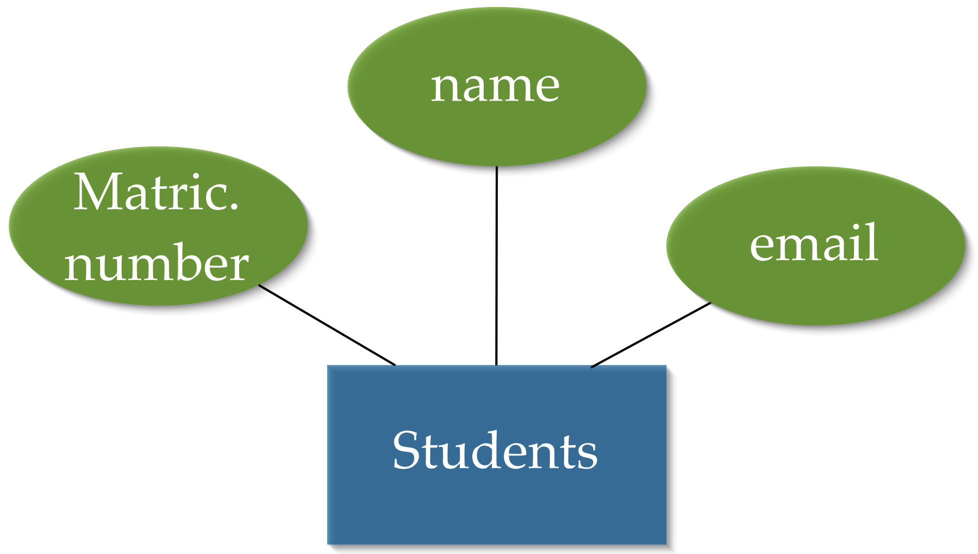


Keys

A *key* is a minimal set of attributes whose values allow us to uniquely identify an entity in an entity set

There may be more than one such minimal set, they are called *candidate keys*

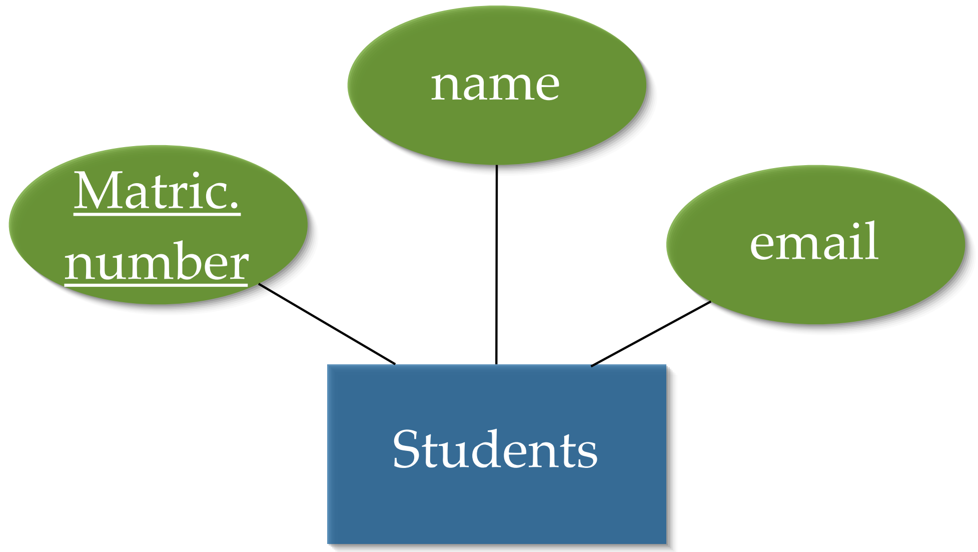
E.g., either Matric. number or email can act as keys.



Primary keys

If multiple candidate keys exist, we choose one and make it the *primary key*.

The attributes occurring in the primary key are *underlined* in the ER diagram



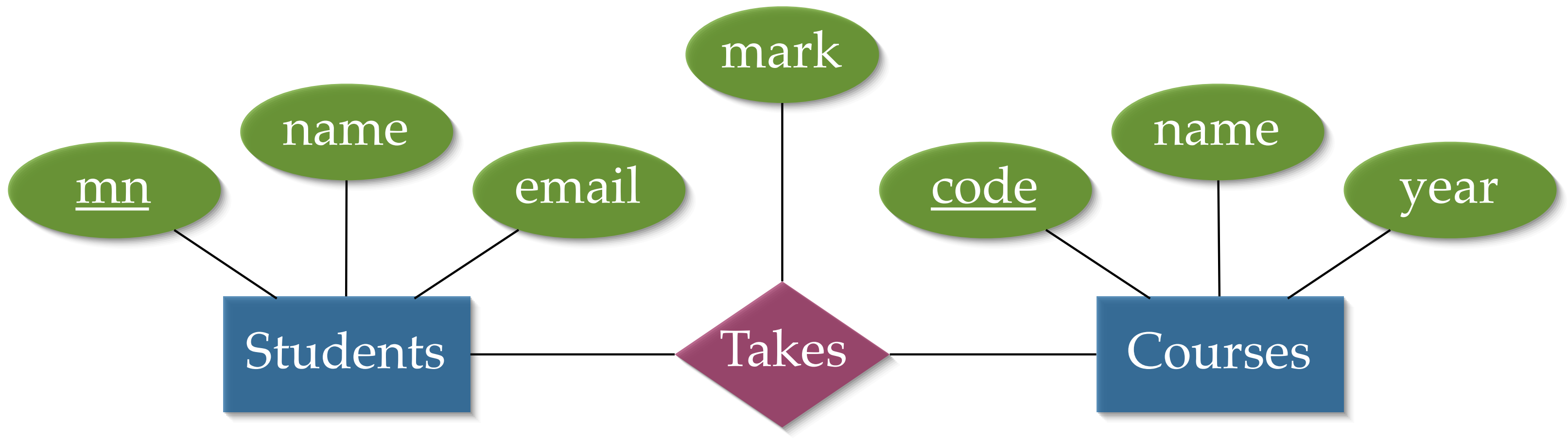
Relationships and relationship sets

Relationships model associations between entities

Relations are grouped into *relationship sets* of relationships between entities from specified entity sets.

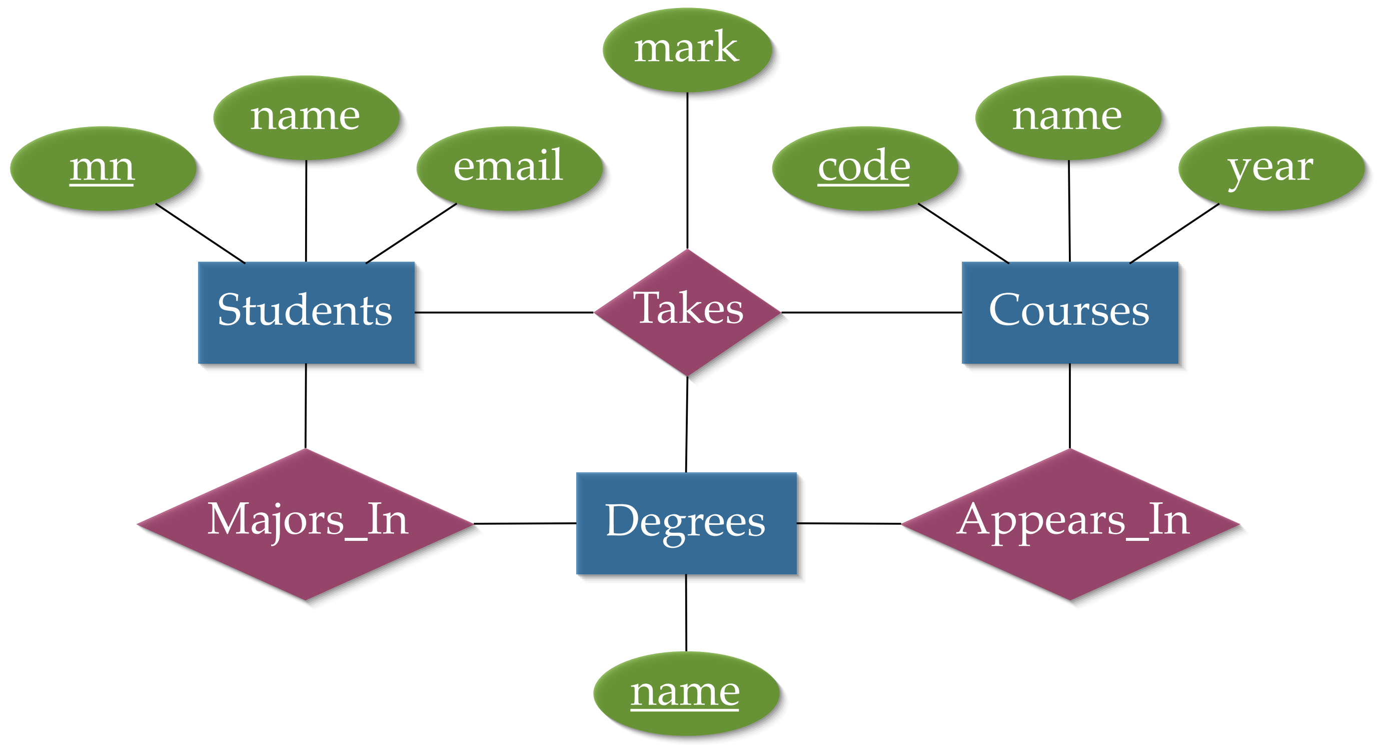
Relationship sets are represented as *diamonds* in ER diagrams

Relationship may have *attributes* of their own.



There is no bound on the number of entities participating in a relationship.

Correspondingly, there is no bound on the number of relationships an entity can participate in



Instances

Entity instances and *relationship instances* are what we obtain after instantiating the attributes of an entity or a relationship

Examples

An entity instance from the Students entity set:

$$(\mathbf{123},\ \text{Natassa},\ \text{natassa@somewhere})$$

An entity instance from the Courses entity set:

$$(\text{inf1}, \text{Informatics 1},\ \mathbf{1})$$

A relationship instance from the Takes relationship set:

$$(\mathbf{123},\ \text{Natassa},\ \text{natassa@somewhere},\ \text{inf1}, \text{Informatics 1},\ \mathbf{1},\ \mathbf{88})$$

Key constraints

A *key constraint* captures identification connections between entities participating in a relationship

Definition. Suppose $\boldsymbol{R}$ is a relationship between $\boldsymbol{n}$ entity sets, $\boldsymbol{E_1, \ldots, E_n}$. There is a *key constraint* on one of the entities, $\boldsymbol{E_k}$, if, by instantiating the attributes of $\boldsymbol{E_k}$, we uniquely identify the relationship instance it participates in.

Example. Students, directors of studies (DoS), and the relationship between them (Directed-By)

- Given a Students instance, we can determine the Directed-By instance it appears in. That is, each student has a unique DoS.

One-to-many and many-to-many relationships

A *one-to-many* relationship $\boldsymbol{R}$ between entity sets $\boldsymbol{E_o}$ and $\boldsymbol{E_m}$ means that, for each instance $\boldsymbol{e_m} \in \boldsymbol{E_m}$, there is at most one instance $\boldsymbol{e_o} \in \boldsymbol{E_o}$ such that $\boldsymbol{e_o}$ and $\boldsymbol{e_m}$ appear together in some relationship instance $\boldsymbol{r} \in \boldsymbol{R}$.

More simply: each instance $\boldsymbol{e_o} \in \boldsymbol{E_o}$ may be associated (in $\boldsymbol{R}$) with many instances $\boldsymbol{e_m} \in \boldsymbol{E_m}$, but each instance $\boldsymbol{e_m} \in \boldsymbol{E_m}$ must be associated (in $\boldsymbol{R}$) with at most one instance $\boldsymbol{e_o} \in \boldsymbol{E_o}$.

If $\boldsymbol{R}$ is a binary relationship between $\boldsymbol{E_o}$ and $\boldsymbol{E_m}$, then being one-to-many is equivalent to there being a key contraint on $\boldsymbol{E_m}$.

A *many-to-many* relationship $\boldsymbol{R}$ between entity sets $\boldsymbol{E_o}$ and $\boldsymbol{E_m}$ means that there are no constraints on the number of times entity instances $\boldsymbol{e_o} \in \boldsymbol{E_o}$ and $\boldsymbol{e_m} \in \boldsymbol{E_m}$ may appear in relationship instances $\boldsymbol{r} \in \boldsymbol{R}$.

Examples

The Directed_By relationship between the Students and DoS entity sets is a many-to-one relationship.

- Each student has a single DoS, but
- each DoS may have many students

The Takes relationship between Students and Courses is a many-to-many relationship

- Each student takes many different courses;
- Each course may be taken by many different students

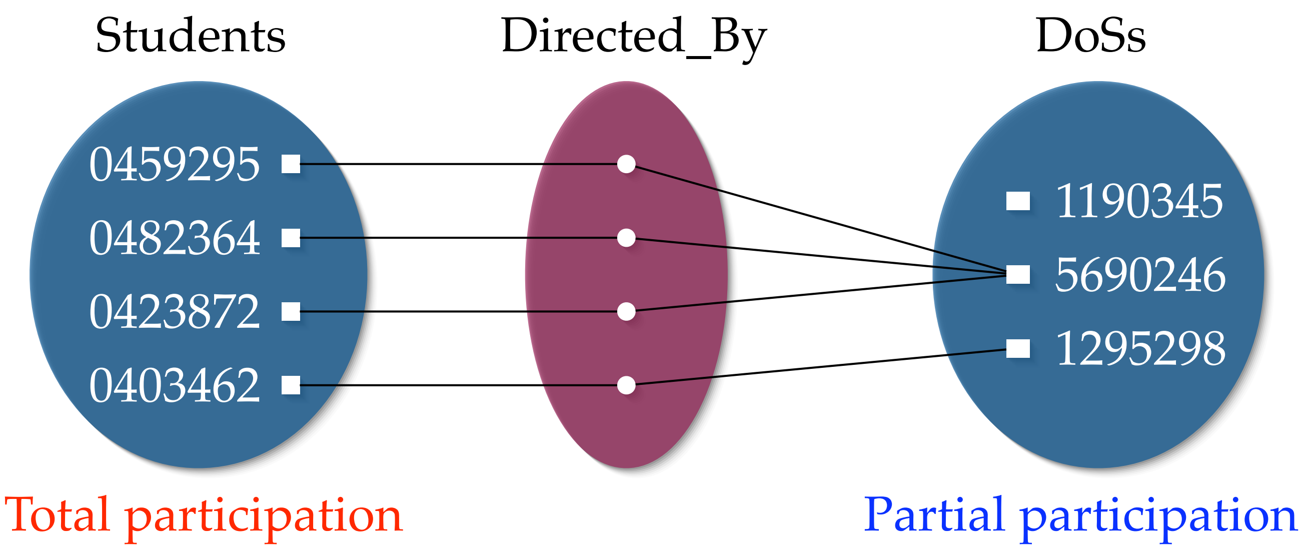
Participation constraints

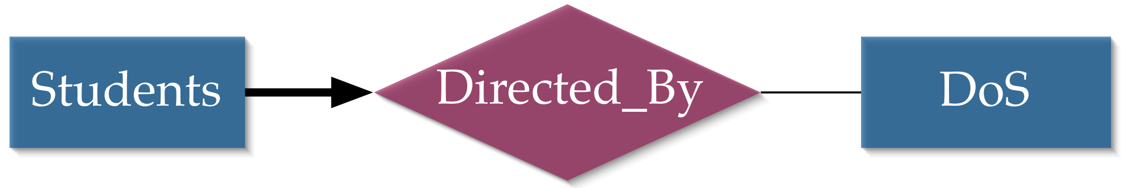
Participation constraints capture the mode in which an entity participates in a relationship.

Total participation on entity set $\boldsymbol{E}$ for relationship $\boldsymbol{R}$ is declared when every entity instance $\boldsymbol{e} \in \boldsymbol{E}$ appears in at least one relationship instance of $\boldsymbol{R}$.

Partial participation on entity set $\boldsymbol{E}$ for relationship $\boldsymbol{R}$ is declared when there exist entities $\boldsymbol{e} \in \boldsymbol{E}$ that do not appear in instances of $\boldsymbol{R}$.

Example



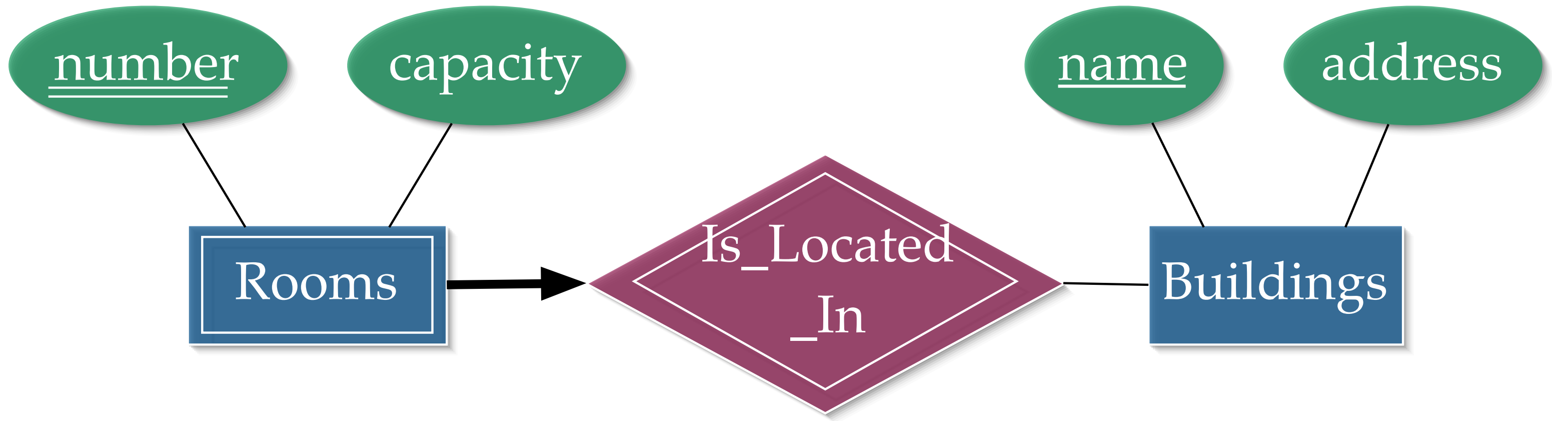


Notation. A *thick arrow* from an entity to a relationship represents that the entity both totally participates in the relationship and also satisfies a key constraint.

Weak entity sets

In certain cases, it is impossible to designate a primary key for entities of an entity set.

Instead, the only way in which set participation can be declared is by “borrowing” the key of another entity set

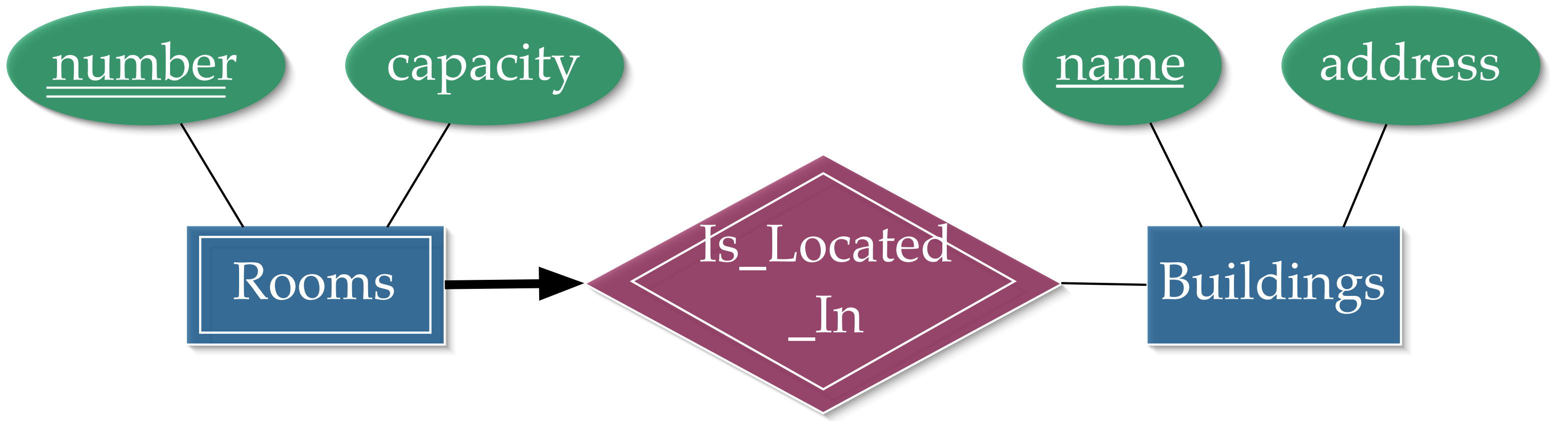


Notation

Double lines for weak entity and identifying relationship

Doubly underlined attributes of the weak entity set participating in the composite key

The *identifying relationship* is many-to-one and total.



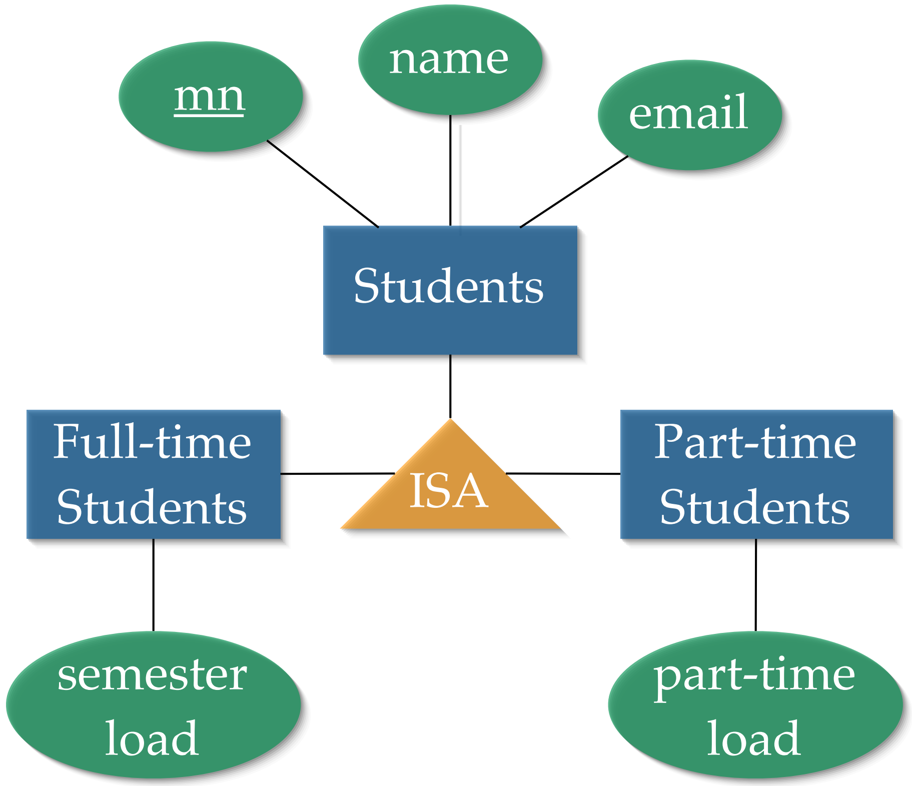
Weak entity set: Definition

- A *weak entity set* is an entity set for which a primary key consisting only of its own attributes cannot be identified
- The *key* is formed by a combination of its own attributes and the key attributes from another entity set with which it has a relationship
- The entity set from which attributes are borrowed is called the *identifying owner*
- The relationship between the weak entity set and its identifying owner is called an *identifying relationship*.
- The identifying relationship must be many-to-one and total.

Hierarchical entities and inheritance

Subclasses (Full-time Students, Part-time Students) *specialise* a *superclass* (Students) by *inheriting* attributes from the superclass.

Subclasses also have additional attributes of their own.


mn
name
email
Students
Full-time Students
ISA
Part-time Students
semester load
part-time load