

# The Relational Model

G51DBS Database Systems

Jason Atkin

Slides are on Moodle

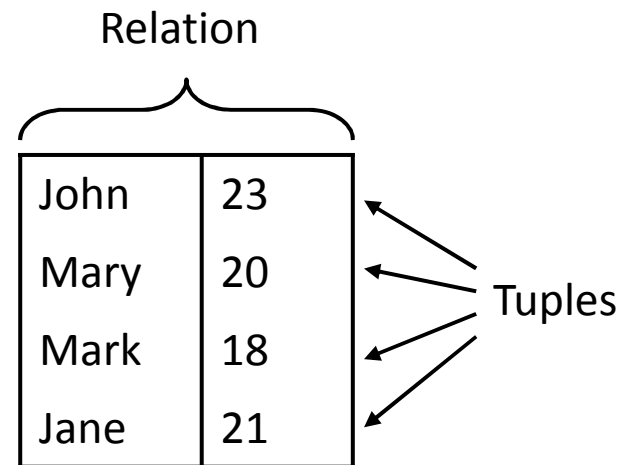
[jaa@cs.nott.ac.uk](mailto:jaa@cs.nott.ac.uk)

# This Lecture

- The Relational Model
  - More on Relations
  - Relational data integrity
- Further reading
  - Database Systems, Connolly & Begg, Chapter 4.2
  - The Manga Guide to Databases, Chapter 2

# Last Lecture

- Data is stored in *relations* (tables)
- Data takes the form of *tuples* (rows)
  - The order of tuples is not important
  - There must not be duplicate tuples



# Reminder: Example from Last Lecture

What is the result of the following operation

$\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4)}(R \times S))$ , where R and S are:

R

Anne	111111
Bob	222222

S

Chris	111111
Dan	222222

# Reminder: Example from Last Lecture

What is the result of the following operation

$\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4)}(\mathbf{R} \times \mathbf{S}))$ , where R and S are:

R x S

Anne	111111	Chris	111111
Anne	111111	Dan	222222
Bob	222222	Chris	111111
Bob	222222	Dan	222222

# Reminder: Example from Last Lecture

What is the result of the following operation

$\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4)}(R \times S))$ , where R and S are:

$\sigma_{\text{col}(2) = \text{col}(4)}(R \times S)$

Anne	111111	Chris	111111
Bob	222222	Dan	222222

# Reminder: Example from Last Lecture

What is the result of the following operation

$\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4)}(R \times S))$ , where R and S are:

$\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4)}(R \times S))$

Anne	Chris
Bob	Dan

# Another example

- What about a single table?
- Can we find a list of pairs of people who share a phone number?

R

Anne	111111
Chris	222222
Bob	333333
Dan	111111
Max	222222
Sam	444444
Joe	555555



# Another example

What about a single table? Can we find a list of pairs of people who share a phone number?

Answer:  $\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

R

Anne	111111
Chris	222222
Bob	333333
Dan	111111
Max	222222
Sam	444444
Joe	555555

# Another example

What about a single table? Can we find a list of pairs of people who share a phone number?

Answer:  $\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

(R x R)

Anne	111111	Anne	111111
Chris	222222	Anne	111111
Bob	333333	Anne	111111
Dan	111111	Anne	111111
Max	222222	Anne	111111
Sam	444444	Anne	111111
Joe	555555	Anne	111111
Anne	111111	Chris	222222
Chris	222222	Chris	222222
Bob	333333	Chris	222222
etc	etc	etc	etc

# Another example

What about a single table? Can we find a list of pairs of people who share a phone number?

Answer:  $\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

$\sigma_{\text{col}(1) \neq \text{col}(3)} (R \times R)$

Anne	111111	Anne	111111
Chris	222222	Anne	111111
Bob	333333	Anne	111111
Dan	111111	Anne	111111
Max	222222	Anne	111111
Sam	444444	Anne	111111
Joe	555555	Anne	111111
Anne	111111	Chris	222222
<del>Chris</del>	<del>222222</del>	<del>Chris</del>	<del>222222</del>
Bob	333333	Chris	222222
etc	etc	etc	etc

# Another example

What about a single table? Can we find a list of pairs of people who share a phone number?

Answer:  $\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

$\sigma_{\text{col}(1) \neq \text{col}(3)} (R \times R)$

Chris	222222	Anne	111111
Bob	333333	Anne	111111
Dan	111111	Anne	111111
Max	222222	Anne	111111
Sam	444444	Anne	111111
Joe	555555	Anne	111111
Anne	111111	Chris	222222
Bob	333333	Chris	222222
Dan	111111	Chris	222222
Max	222222	Chris	222222
etc	etc	etc	etc

# Another example

What about a single table? Can we find a list of pairs of people who share a phone number?

Answer:  $\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

$\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R)$

Chris	222222	Anne	111111
Bob	333333	Anne	111111
Dan	111111	Anne	111111
Max	222222	Anne	111111
Sam	444444	Anne	111111
Joe	555555	Anne	111111
Anne	111111	Chris	222222
Bob	333333	Chris	222222
Dan	111111	Chris	222222
Max	222222	Chris	222222
etc	etc	etc	etc

# Another example

What about a single table? Can we find a list of pairs of people who share a phone number?

Answer:  $\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

$\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R)$

Dan	111111	Anne	111111
Max	222222	Chris	222222
Anne	111111	Dan	111111
Chris	222222	Max	222222

# Another example

What about a single table? Can we find a list of pairs of people who share a phone number?

Answer:  $\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

$\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

Dan	111111	Anne	111111
Max	222222	Chris	222222
Anne	111111	Dan	111111
Chris	222222	Max	222222

# Another example

What about a single table? Can we find a list of pairs of people who share a phone number?

Answer:  $\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

$\pi_{1,3}(\sigma_{\text{col}(2) = \text{col}(4) \text{ and } \text{col}(1) \neq \text{col}(3)} (R \times R))$

Dan	Anne
Max	Chris
Anne	Dan
Chris	Max

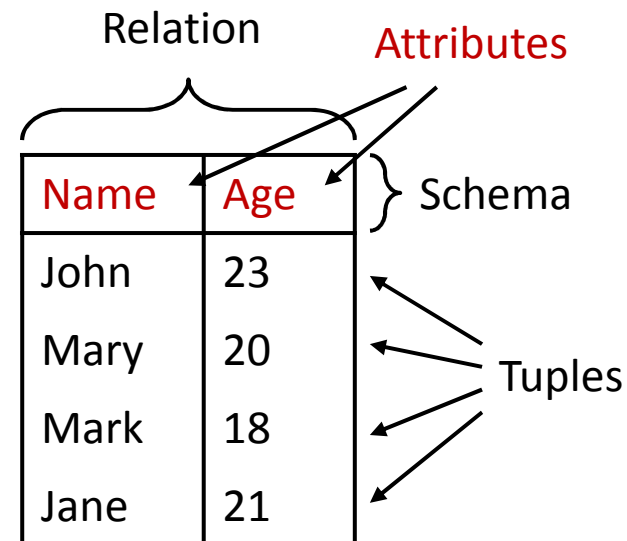


# Schemas and Attributes

- Previously, we referenced specific columns in a relation using numbers
  - E.g.  $\pi_{1,2}(R)$
- It is often helpful to reference columns using names, which we will have to provide
- **Attributes** are **named columns** in a relation
- A schema defines the attributes for a relation

# Relational Data Structure

- Each relation has a *schema* (sometimes called a scheme or heading)
- The schema defines the relation's *attributes* (columns).



# Named and Unnamed Tuples

- Tuples specify values for each attribute in a relation
- When writing tuples down, they can be named as sets of pairs, e.g.
  - $\{ (1, \text{John}), (2, 23) \}$  or  $\{ (2, 23), (1, \text{John}) \}$
  - $\{ (\text{Name}, \text{John}), (\text{Age}, 23) \}$
- Or unnamed, for convenience, e.g.
  - $(\text{John}, 23)$  (equivalent to the above)
- There is no real difference between named and unnamed tuples, but be careful with the ordering of unnamed tuples.

# Relational Data Structure

- More formally:
  - A schema is a set of attributes
  - A tuple assigns a value to each attribute in the schema
  - A relation is a set of tuples with the same schema

Name	Age
John	23
Mary	20
Mark	18
Jane	21

{ { (Name, John), (Age, 23) },  
{ (Name, Mary), (Age, 20) },  
{ (Name, Mark), (Age, 18) },  
{ (Name, Jane), (Age, 21) } }

# Example Relation

ID	Name	Salary	Department
M139	John Smith	18,000	Marketing
M140	Mary Jones	22,000	Marketing
A368	Jane Brown	22,000	Accounts
P222	Mark Brown	24,000	Personnel
A367	David Jones	20,000	Accounts

# Example Relation


ID	Name	Salary	Department
M139	John Smith	18,000	Marketing
M140	Mary Jones	22,000	Marketing
A368	Jane Brown	22,000	Accounts
P222	Mark Brown	24,000	Personnel
A367	David Jones	20,000	Accounts

} Schema is { ID, Name, Salary, Department }

# Example Relation

Attributes are ID, Name, Salary and Department

The degree of the relation is 4




ID	Name	Salary	Department
M139	John Smith	18,000	Marketing
M140	Mary Jones	22,000	Marketing
A368	Jane Brown	22,000	Accounts
P222	Mark Brown	24,000	Personnel
A367	David Jones	20,000	Accounts

} Schema is { ID, Name, Salary, Department }

# Example Relation

Attributes are ID, Name, Salary and Department

The degree of the relation is 4



ID	Name	Salary	Department
M139	John Smith	18,000	Marketing
M140	Mary Jones	22,000	Marketing
A368	Jane Brown	22,000	Accounts
P222	Mark Brown	24,000	Personnel
A367	David Jones	20,000	Accounts

} Schema is { ID, Name, Salary, Department }

Tuples, e.g.  
{ (ID, A368),  
(Name, Jane Brown),  
(Salary, 22,000),  
(Department, Accounts) }

The cardinality of the relation is 5



# Relational Data Integrity

- Data integrity controls what data can be in a relation
  - *Domains* restrict the possible values a tuple can assign to each attribute
  - *Candidate* and *Primary Keys* consist of an attribute, or set of attributes, that uniquely identify all tuples
  - *Foreign Keys* link relations to each other

# Attributes and Domains

- A *domain* is given for each attribute
- The domain lists possible values for the attribute
- Each tuple assigns a value to each attribute from *its domain*
- Examples
  - An 'age' might have to come from the set of integers between 0 and 150
  - A 'department' might come from a list of given strings
  - A 'notes' field may allow any string at all

# Candidate Keys

- A set of attributes in a relation is a candidate key if, and only if:
  - Every tuple has a unique value for that set of attributes: *uniqueness*
  - No proper subset of the set has the uniqueness property: *minimality*

ID	First	Last
S139	John	Smith
S140	Mary	Jones
S141	John	Brown
S142	Jane	Smith

**What are the candidate keys here?**

# Candidate Keys

- A set of attributes in a relation is a candidate key if, and only if:
  - Every tuple has a unique value for that set of attributes: *uniqueness*
  - No proper subset of the set has the uniqueness property: *minimality*

ID	First	Last
S139	John	Smith
S140	Mary	Jones
S141	John	Brown
S142	Jane	Smith

Candidate key is {ID};  
{First, Last} looks plausible, but  
people might have the same name

{ID, First}, {ID, Last} and  
{ID, First, Last} satisfy uniqueness,  
but are not minimal

{First} and {Last} do not give a  
unique identifier for each row

# Choosing Candidate Keys

- You can't necessarily infer the candidate keys based solely on the data in your table
  - An instance of a relation will only often hold only a small subset of all the possible values
- You must use knowledge of the real-world to help
  - i.e. will the candidate key ALWAYS be unique?

# Choosing Candidate Keys

What are the candidate keys of the following relation?

CompanyOffices

officeID	Name	Country	Postcode/Zip	Phone
O1001	Headquarters	England	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	England	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	England	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

# Choosing Candidate Keys

The candidate keys are {OfficeID} ...

CompanyOffices

officeID	Name	Country	Postcode/Zip	Phone
O1001	Headquarters	England	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	England	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	England	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

# Choosing Candidate Keys

The candidate keys are {OfficeID}, {Phone} ...

CompanyOffices

officeID	Name	Country	Postcode/Zip	Phone
O1001	Headquarters	England	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	England	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	England	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763



# Choosing Candidate Keys

The candidate keys are {OfficeID}, {Phone} and {Name, Postcode/Zip}

CompanyOffices

officeID	Name	Country	Postcode/Zip	Phone
O1001	Headquarters	England	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	England	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	England	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

# Choosing Candidate Keys

## NOT KEYS LIKE THIS:

CompanyOffices

officeID	Name	Country	Postcode/Zip	Phone
O1001	Headquarters	England	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	England	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	England	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

Keys like {Name, Country, Phone} satisfy uniqueness,  
**but not minimality**

# Primary Keys

- One candidate key is usually chosen to identify tuples in a relation
- This is called the *Primary Key*
- Often a special ID is used as the Primary Key

ID	First	Last
S139	John	Smith
S140	Mary	Jones
S141	John	Brown
S142	Jane	Smith

We might use either {ID} or {First,Last} as the primary key. ID is more convenient as we know it will always be unique. People could have the same name

# NULLs and Primary Keys

- Missing information can be represented using NULLs
- A NULL indicates a missing or unknown value
- This will be discussed in a later lecture
- *Entity integrity*  
Primary Keys cannot contain NULL values

# Foreign Keys

- Foreign Keys are used to link data in two relations. A set of attributes in the first (referencing) relation is a Foreign Key if its value **either**:
  - **Matches a Candidate Key** value in a second (referenced) relation
  - Is **wholly** NULL
- This is called *Referential Integrity*

# Foreign Keys Example

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

{DID} is a Candidate Key for Department – Each entry has a unique value for DID

Employee

EID	EName	DID
15	John Smith	13
16	Mary Jones	14
17	John Brown	13
18	Jane Smith	NULL

{DID} is a Foreign Key in Employee – each employee's DID value is either NULL, or matches an entry in the Department relation. This links each Employee to at most one Department

# Recursive Foreign Keys Example

Employee

ID	Name	Manager
E1496	John Smith	E1499
E1497	Mary Jones	E1498
E1498	John Brown	E1499
E1499	Jane Smith	NULL

{ID} is a Candidate Key for Employee, and {Manager} is a Foreign Key that refers to the same relation. Every tuple's Manager value must match an ID value, or be NULL

# Referential Integrity

- When relations are updated, referential integrity can be violated
- This usually occurs when a referenced tuple is updated or deleted
- There are a number of options when this occurs:
  - RESTRICT – stop the user from doing it
  - CASCADE – let the changes flow on
  - SET NULL – make referencing values null
  - SET DEFAULT – make referencing values the default for their column



# Referential Integrity Example

- What happens if
  - Marketing's DID is changed to 16 in Department?
  - The entry for Accounts is deleted from Department?

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Jones	14
17	John Brown	13
18	Jane Smith	NULL

# RESTRICT

- RESTRICT stops any action that violates integrity
  - You cannot update or delete Marketing or Accounts
  - You *can* change Personnel as it is not referenced

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Jones	14
17	John Brown	13
18	Jane Smith	NULL

# CASCADE

- CASCADE allows the changes made to flow through
  - If Marketing's DID is changed to 16 in Department, then ... ?
  - If Accounts is deleted then ... ?

Department

DID	DName
<del>13</del> 16	Marketing
<del>14</del>	<del>Accounts</del>
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Jones	14
17	John Brown	13
18	Jane Smith	NULL

# CASCADE

- CASCADE allows the changes made to flow through
  - If Marketing's DID is changed to 16 in Department, then the DIDs for John Smith and John Brown also change
  - If Accounts is deleted then so is Mary Jones

Department

DID	DName
<del>13</del> 16	Marketing
<del>14</del>	<del>Accounts</del>
15	Personnel

Employee

EID	EName	DID
15	John Smith	<del>13</del> 16
<del>16</del>	<del>Mary Jones</del>	<del>14</del>
17	John Brown	<del>13</del> 16
18	Jane Smith	NULL

# SET NULL

- What happens if :
  - Marketing's DID is changed to 16 in Department?

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Jones	14
17	John Brown	13
18	Jane Smith	NULL

# SET NULL

- What happens if :
  - Marketing's DID is changed to 16 in Department?
  - The entry for Accounts is deleted from Department

Department

DID	DName
<del>13</del> 16	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	<del>13</del> NULL
16	Mary Jones	14
17	John Brown	<del>13</del> NULL
18	Jane Smith	NULL

# Naming Conventions

- Naming conventions
  - A consistent naming convention can help to remind you of the structure
  - Assign each table a unique prefix. E.g. a student name may be stuName, and a module name modName
  - You may even wish to assign a project prefix to the tables you use
- Naming keys
  - Having a unique number as the primary key can be useful
  - If the table prefix is abc, call this abcID
  - A foreign key to this table is then also called abcID

# Naming Example

## Student

stuID	stuName
-------	---------

These attributes are  
clearly related to the  
student table

## Enrolment

stuID	modID
-------	-------

These attributes are  
foreign keys, related  
to other tables

## Module

modID	modName
-------	---------

These attributes are  
clearly related to the  
module table



# Example

Flight Booking

Booking Date	Travel Date	Flight Number	Passenger Name	Date of Birth	Email Address
12/12/2012	01/03/2013	BA649	Joe Smith	12/05/1970	jxs@cs.nott....
13/12/2012	02/03/2013	UA932	Bob Smith	04/01/1990	bxs@cs.nott....
14/12/2012	03/03/2013	BA281	Alice White	21/12/2012	axw@cs.nott....
15/12/2012	01/03/2013	BA912	Kate Green	29/02/1988	kxg@cs.nott....

Potential questions:

- What are the potential primary keys?
- Which fields do you think may be foreign keys?
  - Into what potential relations?
- Which fields do you think could validly be NULL?

# Further reading

- Database Systems, Connolly & Begg, Chapter 12
- The Manga Guide to Databases, Chapter 3

# Entity Relationship Modelling

G51DBS Database Systems

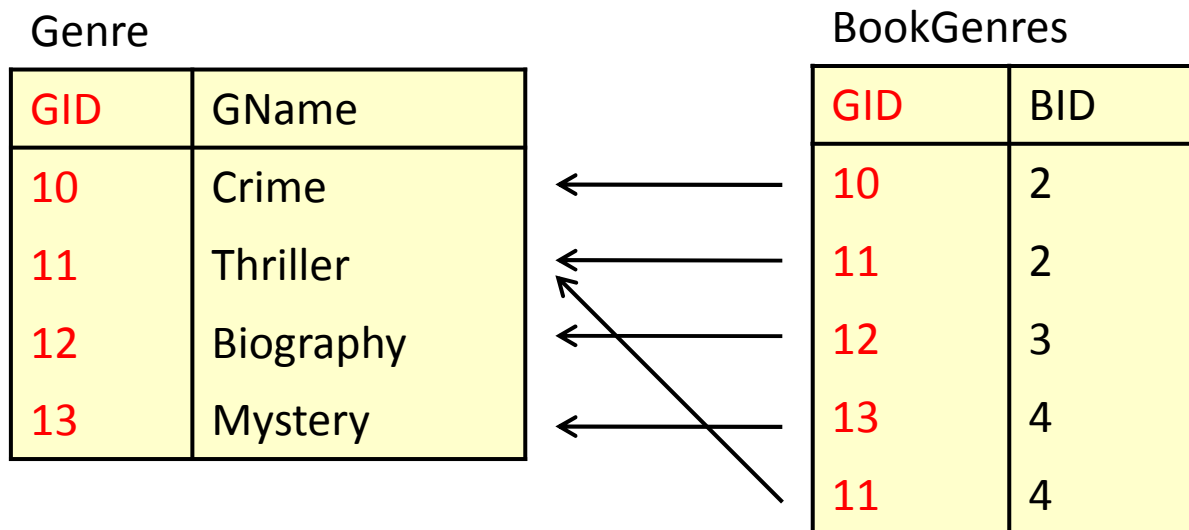
Jason Atkin

Slides are on Moodle

[jaa@cs.nott.ac.uk](mailto:jaa@cs.nott.ac.uk)

# Last Lecture

- Foreign Keys reference a Candidate Key in another relation.



# This Lecture

- Entity/Relationship models
  - Entities and Attributes
  - Relationships
  - E/R Diagrams
- Further Reading
  - Database Systems, Connolly & Begg, Chapter 12
  - The Manga Guide to Databases, Chapter 3

# Database Design

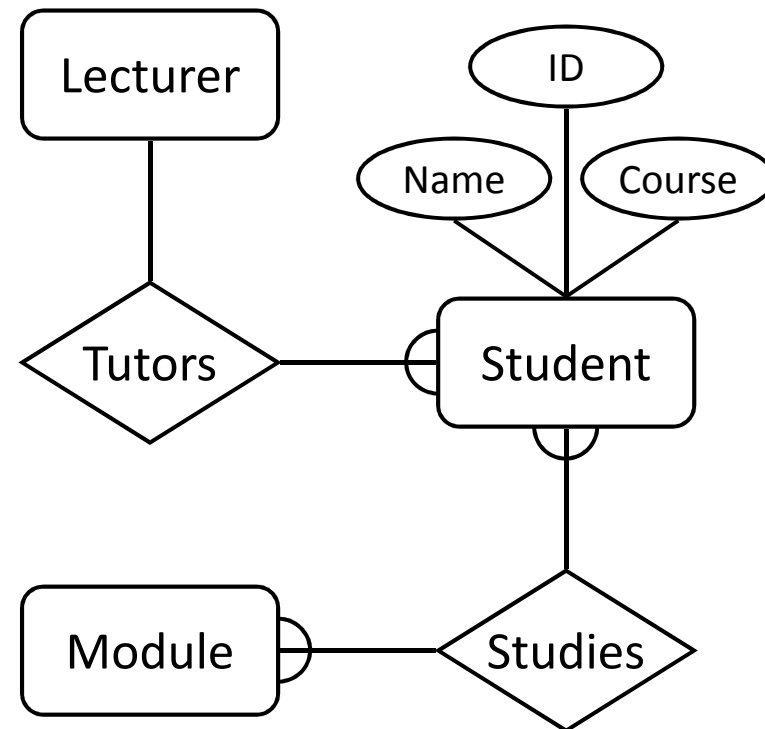
- Before looking at creating and using a database, we will look at how to design one
- Need to consider:
  - Tables?
    - Attributes?
    - Keys?
  - Relationships?
- **Designing** your database is **important**
- Create a database design that is **independent of DBMS**
- Often results in simpler and more efficient queries once database has been created

# Entity/Relationship Modelling

- E/R Modelling is used for **conceptual** design
  - Entities: objects or items of interest (Tables/relations)
  - Attributes: properties of an entity
  - Relationships: links between entities
- Example: University database
- Entities:
  - Students
    - With attributes such as ID, Name, and Course
  - Modules
  - Lecturers
- And relationships:
  - Between Students and Modules (enrolment)
  - Between Students and Lecturers (tutor/tutee)

# Entity/Relationship Diagrams

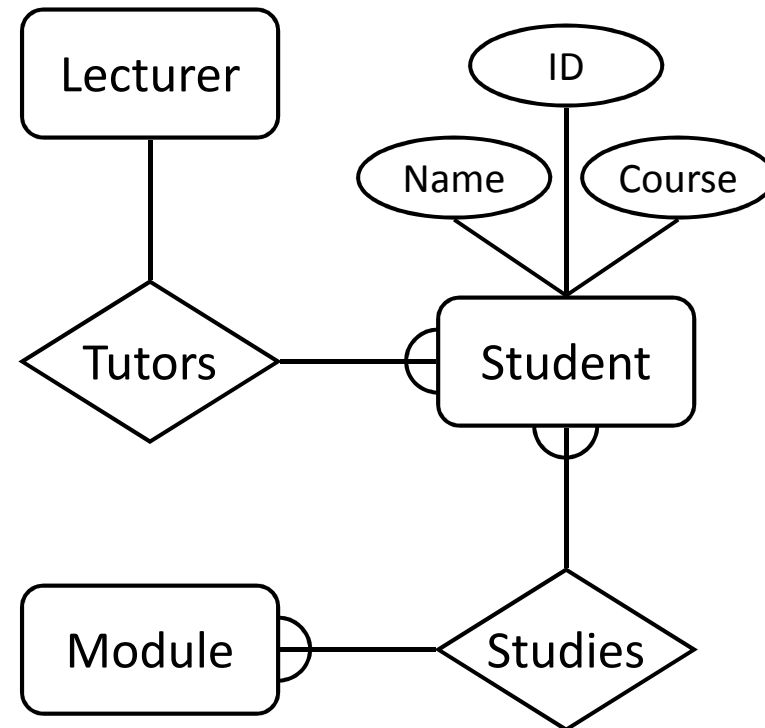
- E/R Models are often represented as E/R diagrams that
  - Give a conceptual view of the database
  - Are independent of the choice of DBMS
  - Can identify some problems in a design





# Diagram Conventions

- There are various notations for representing E/R diagrams
- These specify the shape of the various components, and the notation used to represent relationships
- **For this introductory module**, we will use simplified diagrams
- **Please ensure that you use these shapes for your coursework and any exam question**

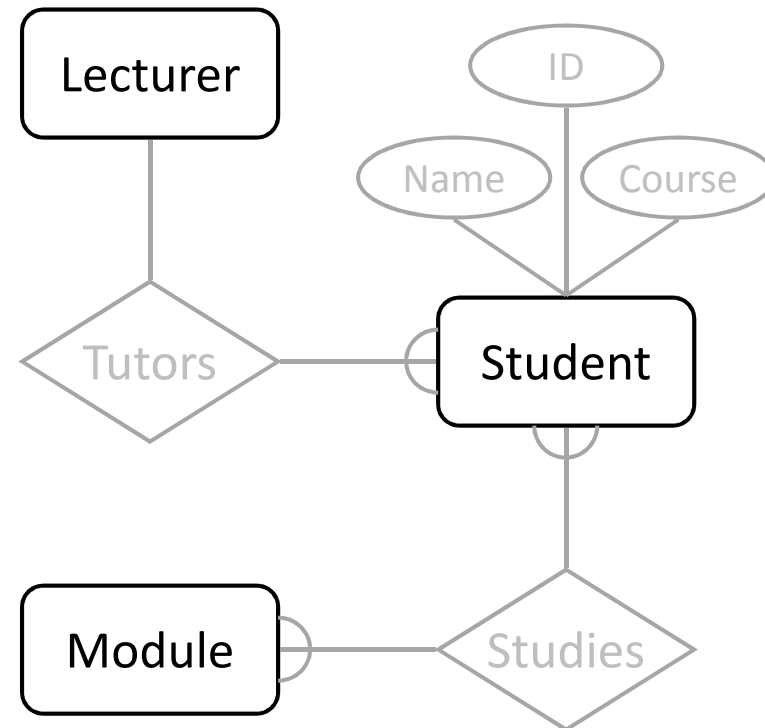


# Entities

- Entities represent objects or things of interest
  - Physical things like students, lecturers, employees, products
  - More abstract things like modules, orders, courses, projects
- Entities have
  - A general type or class, such as Lecturer or Module
  - Instances of that particular type.
    - E.g. Peter Blanchfield, Steven Bagley are instances of Lecturer
  - Attributes (such as name, email address)

# Diagramming Entities

- In E/R Diagrams, we will represent *Entities* as boxes with rounded corners
- The box is labelled with the *name* of the entity
  - The table name?
  - Name of class of objects represented by that entity
  - Name of real-world thing?

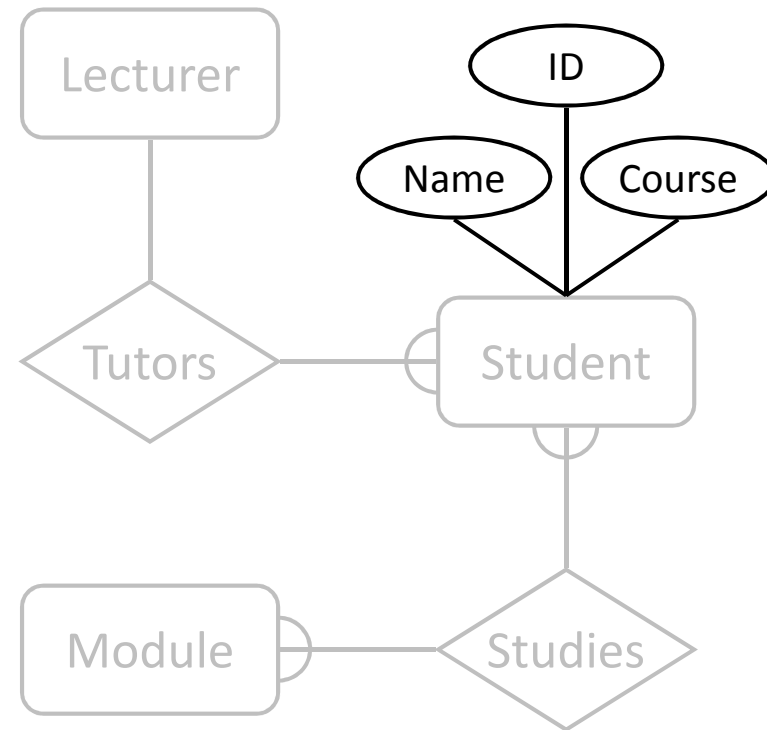


# Attributes

- Attributes are facts, aspects, properties, or details **about an entity**
  - Students have IDs, names, courses, addresses, ...
  - Modules have codes, titles, credit weights, levels, ...
- Attributes have
  - A name
  - An associated entity
  - Domains of possible values
    - E.g. 'integer', 'surnames', 'strings'?
  - For each *instance* of the associated entity, a *value* from the attributes domain

# Diagramming Attributes

- In an E/R Diagram *attributes* are drawn as ovals
- Each attribute is linked to its entity by a line
- The *name* of the attribute is written in the oval



# Relationships

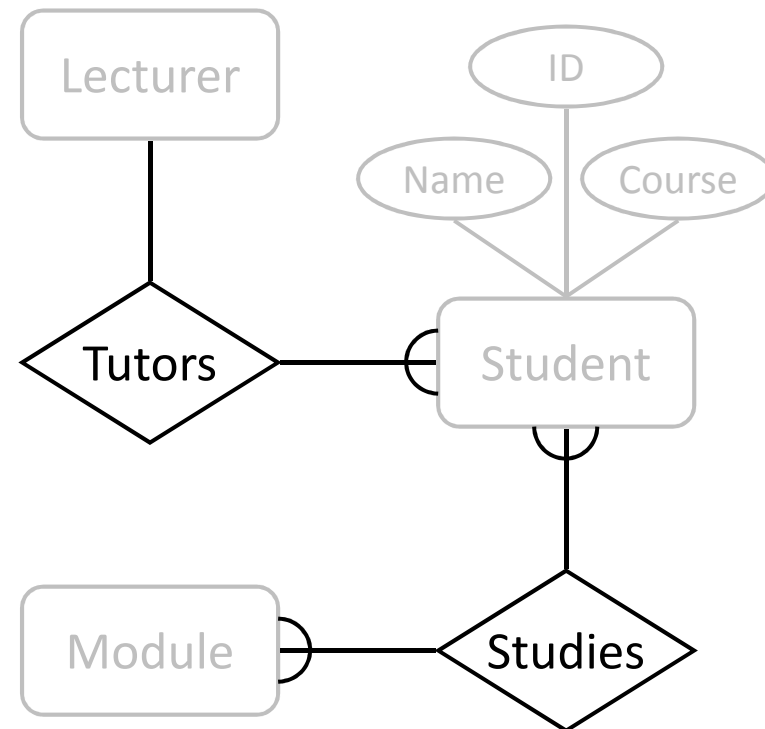
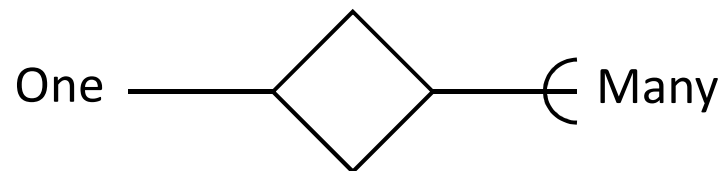
- *Relationships* are an *association* between two or more entities
  - Each Student takes several Modules
  - Each Module is taught by a Lecturer
  - Each Employee works for a single Department
- Relationships have
  - A name
  - A set of entities that participate in them
  - A degree
    - The number of entities that participate (most have degree 2)
  - A cardinality ratio
    - The number of each entity type

# Cardinality Ratios

- Each entity in a relationship can participate in *zero, one, or more than one* **instances** of that relationship
- We will ignore optional (zero instances) of relationships
  - Although you could label them with a \* or similar
- This leads to 3 types of relationship...
- One to one (1:1)
  - Each lecturer has a unique office
  - Each office has one lecturer
- One to many (1:M)
  - A lecturer may tutor **many** students
  - But each student has just **one** tutor
- Many to many (M:M)
  - Each student takes **several** modules
  - Each module is taken by **several** students

# Entity/Relationship Diagrams

- Relationships are shown as links between two entities
- The name is given in a diamond box
- The ends of the link show cardinality





# Making E/R Models

- To make an E/R model you need to identify
  - Entities
  - Attributes
  - Relationships
  - Cardinality ratios
- **We obtain these from a problem description**
- **General guidelines**
  - **Entities** are *things* or *objects*, so they are *often nouns* in the description
  - **Attributes** are *facts* or *properties*, and so are *often nouns* also
  - **Verbs** often describe relationships between entities

# Example

A university consists of a number of departments. Each department offers several courses. A number of modules make up each course. Students enrol in a particular course and take modules towards the completion of that course. Each module is taught by a lecturer from the appropriate department, and each lecturer tutors a group of students

**What are the entities?**

# Example - Entities

A university consists of a number of departments. Each **department** offers several **courses**. A number of **modules** make up each course. **Students** enrol in a particular course and take modules towards the completion of that course. Each module is taught by a **lecturer** from the appropriate department, and each lecturer tutors a group of students

- **Entities – Department, Course, Module, Student, Lecturer**

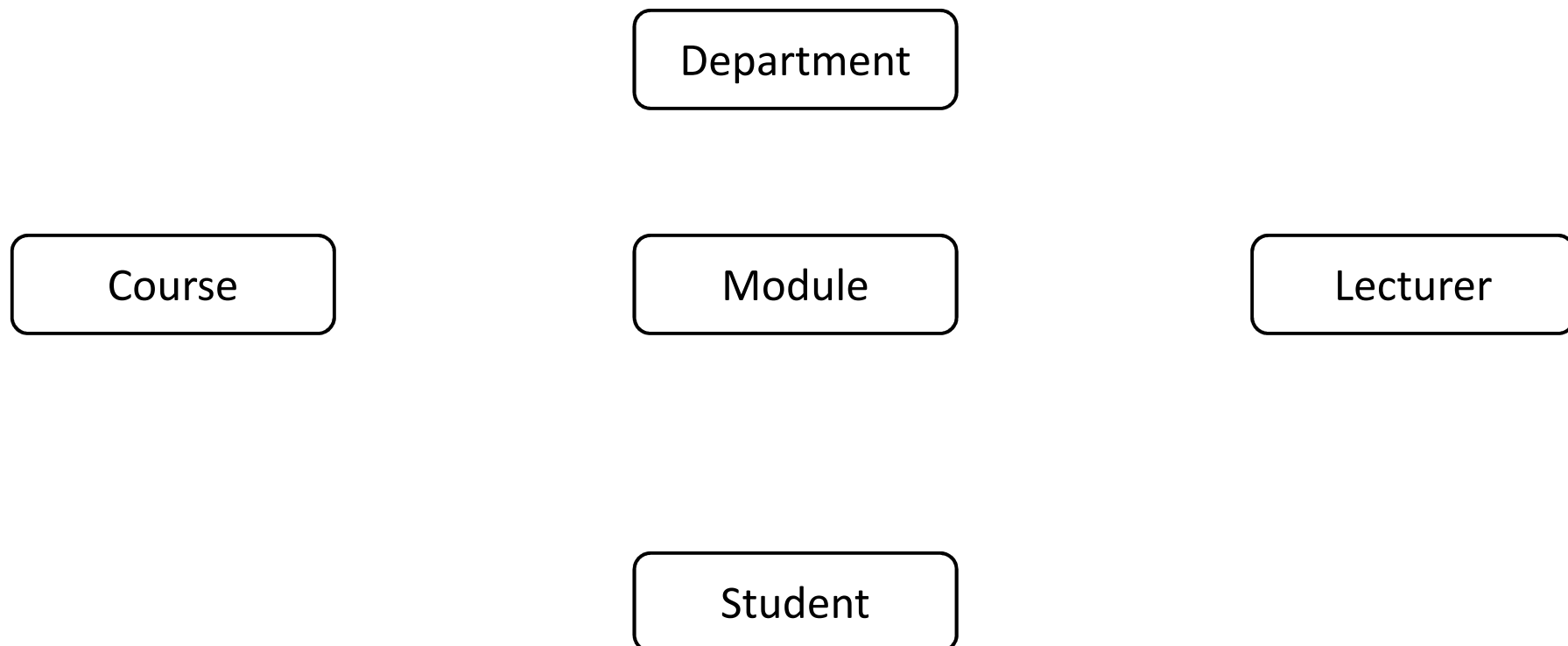
# Example - Relationships

A university consists of a number of departments. Each **department** **offers** several **courses**. A number of **modules** **make up** each course. **Students** **enrol** in a particular course and take modules towards the completion of that course. Each module is **taught by** a **lecturer from the** appropriate department, and each lecturer **tutors** a group of students

- **Entities – Department, Course, Module, Student, Lecturer**
- **Relationships – Offers, Make Up, Enrol, Taught By, From The, Tutors**

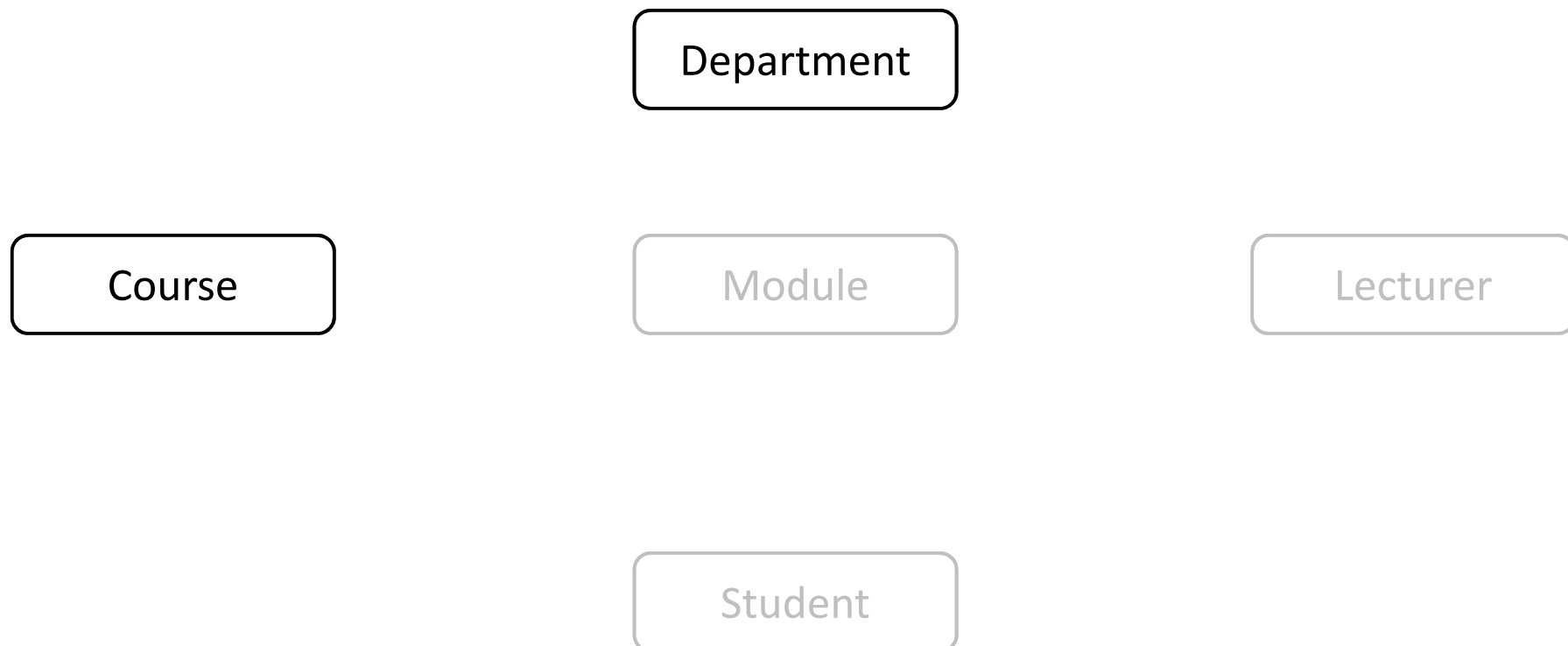
# Example – E/R Diagram

Entities: Department, Course, Module, Lecturer,  
Student



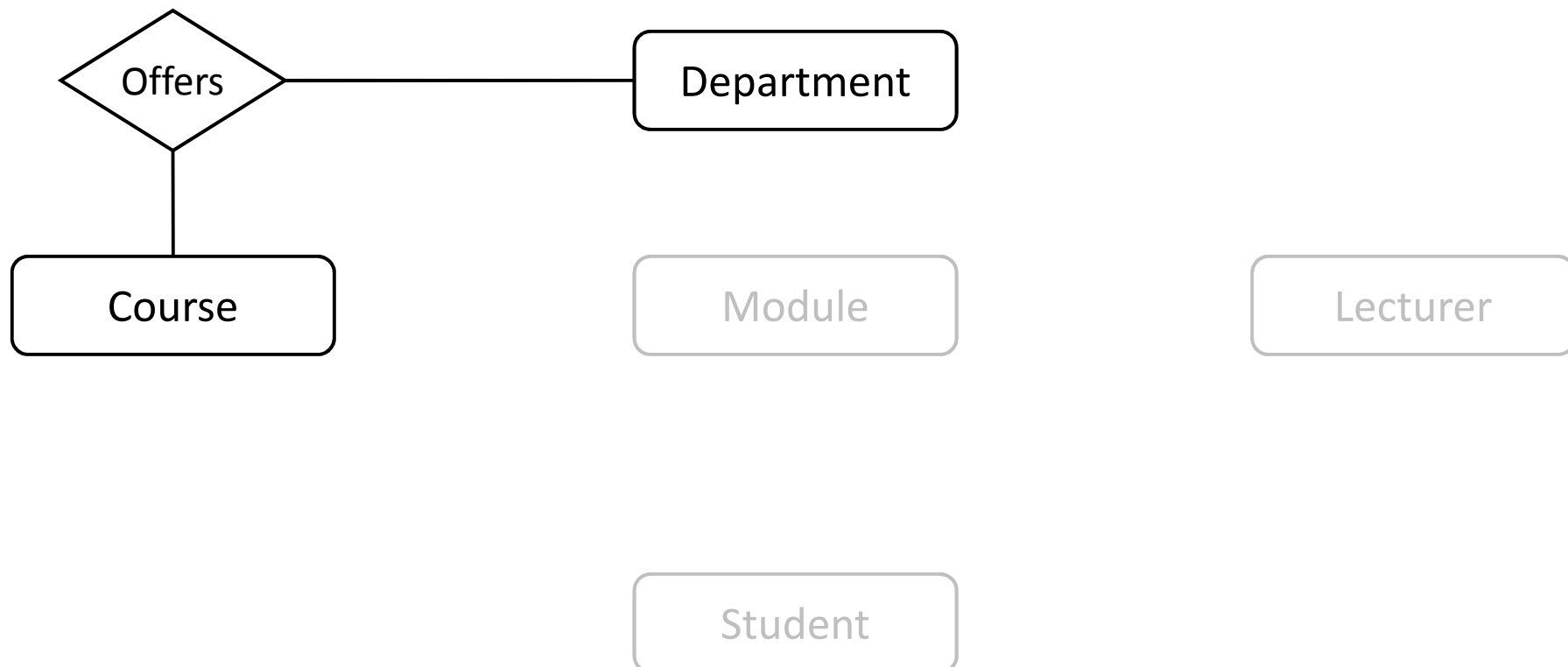
# Example – E/R Diagram

Each Department **offers** several Courses



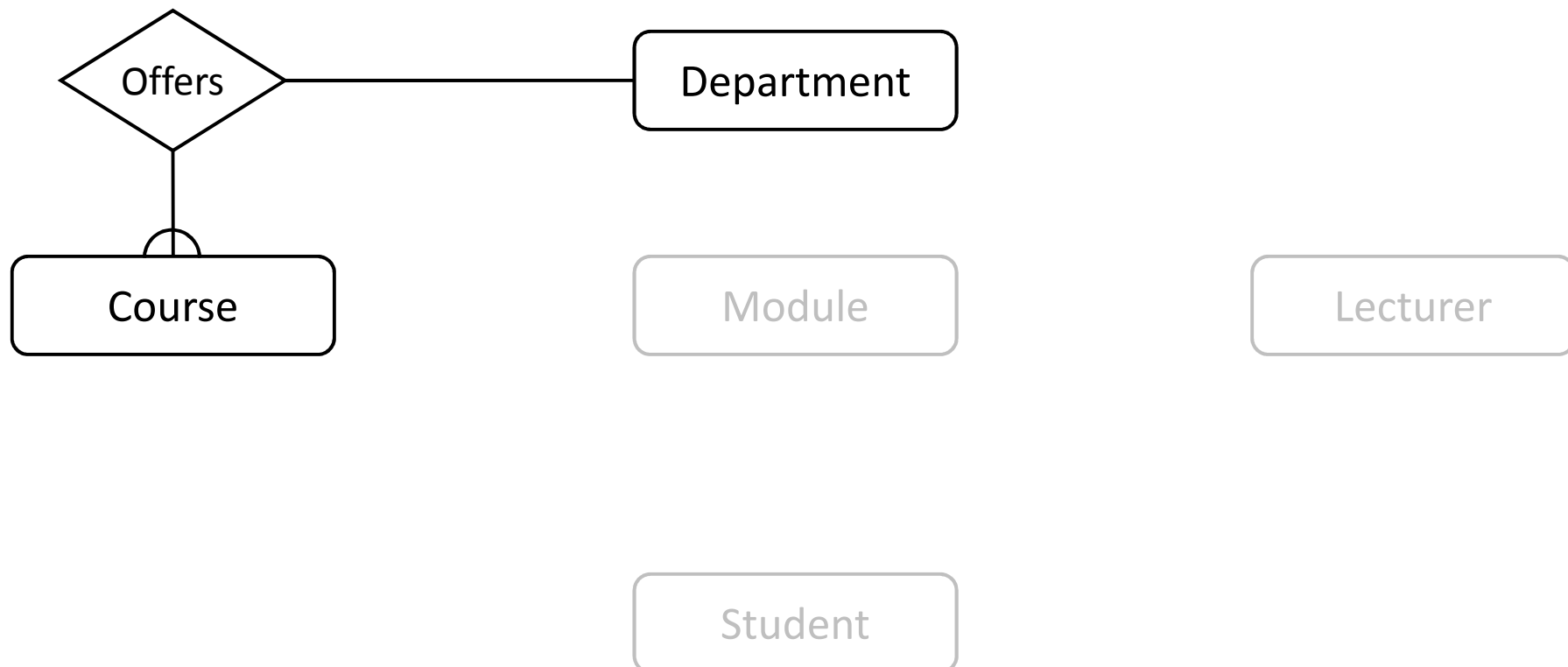
# Example – E/R Diagram

Each Department offers **several** Courses



# Example – E/R Diagram

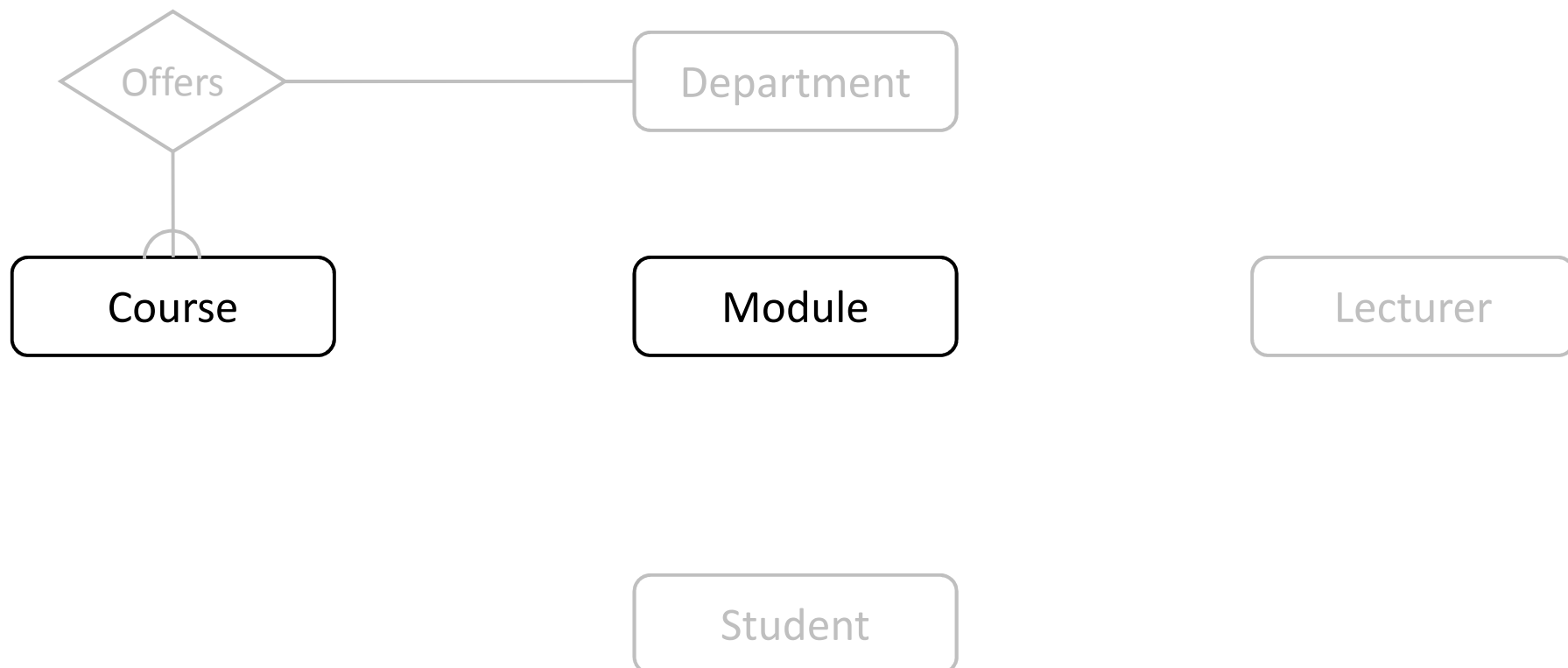
Each Department offers several Courses





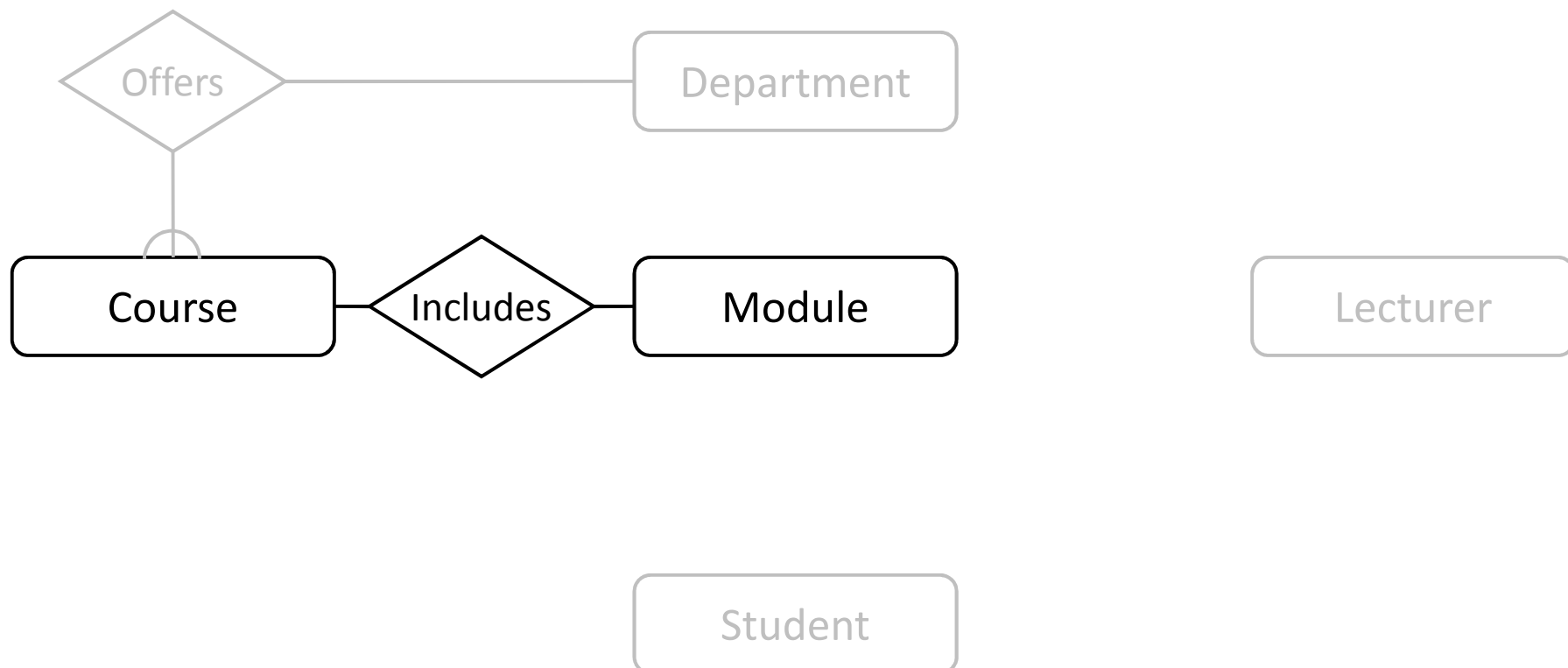
# Example – E/R Diagram

A number of modules make up each Course



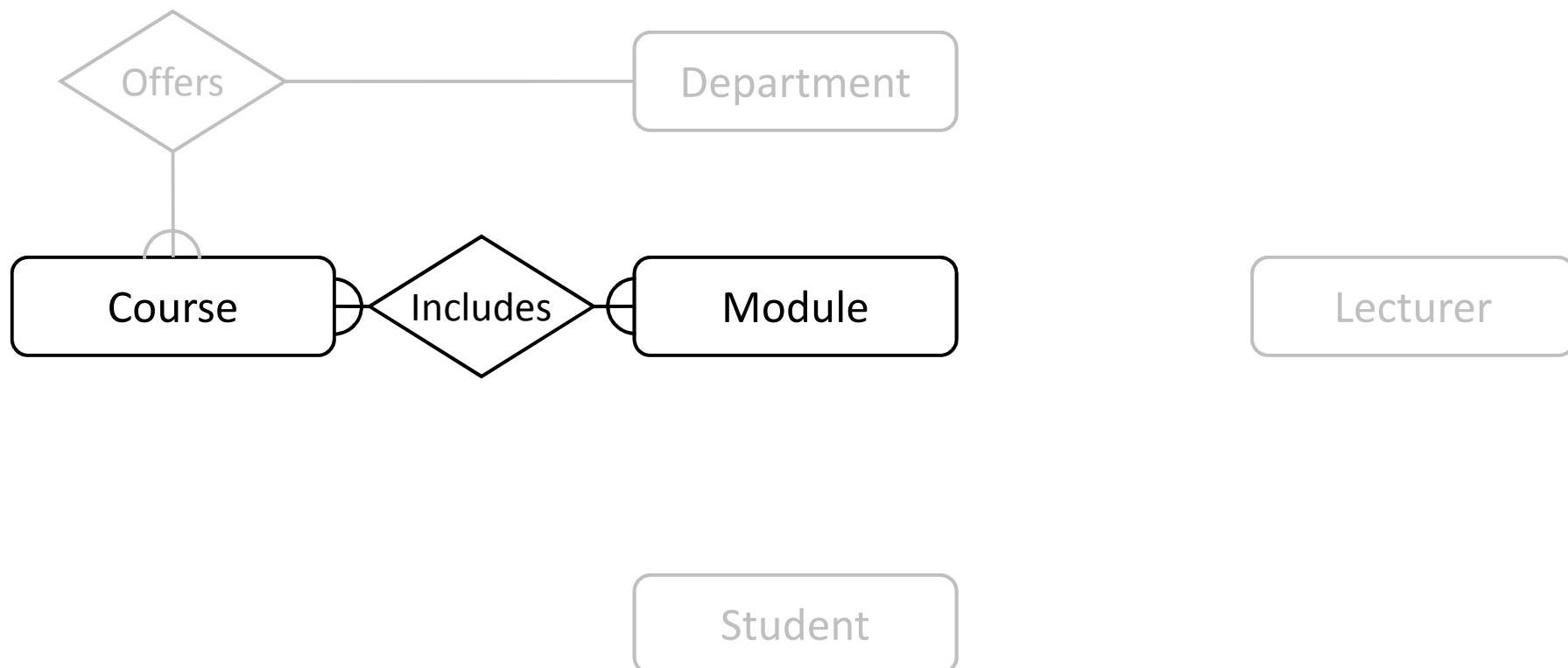
# Example – E/R Diagram

A number of modules make up each Course



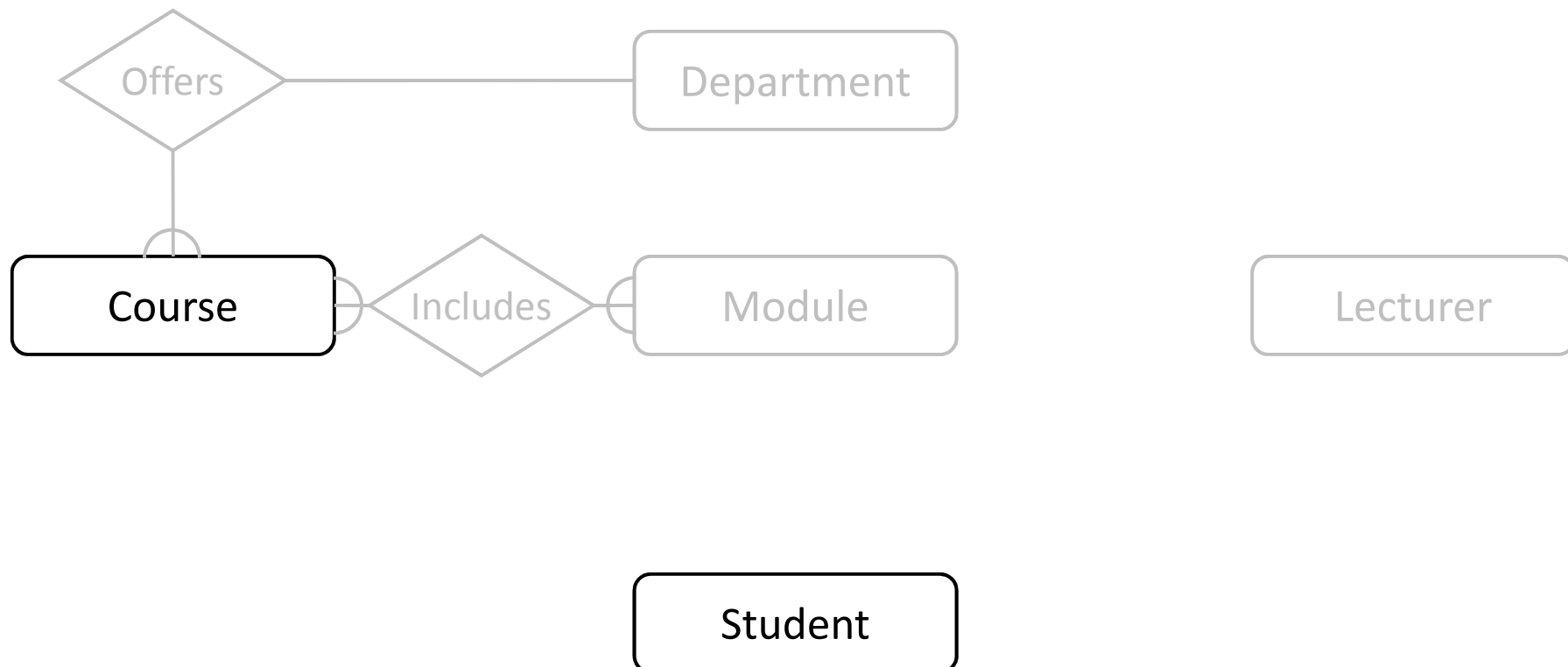
# Example – E/R Diagram

A number of modules make up each Course



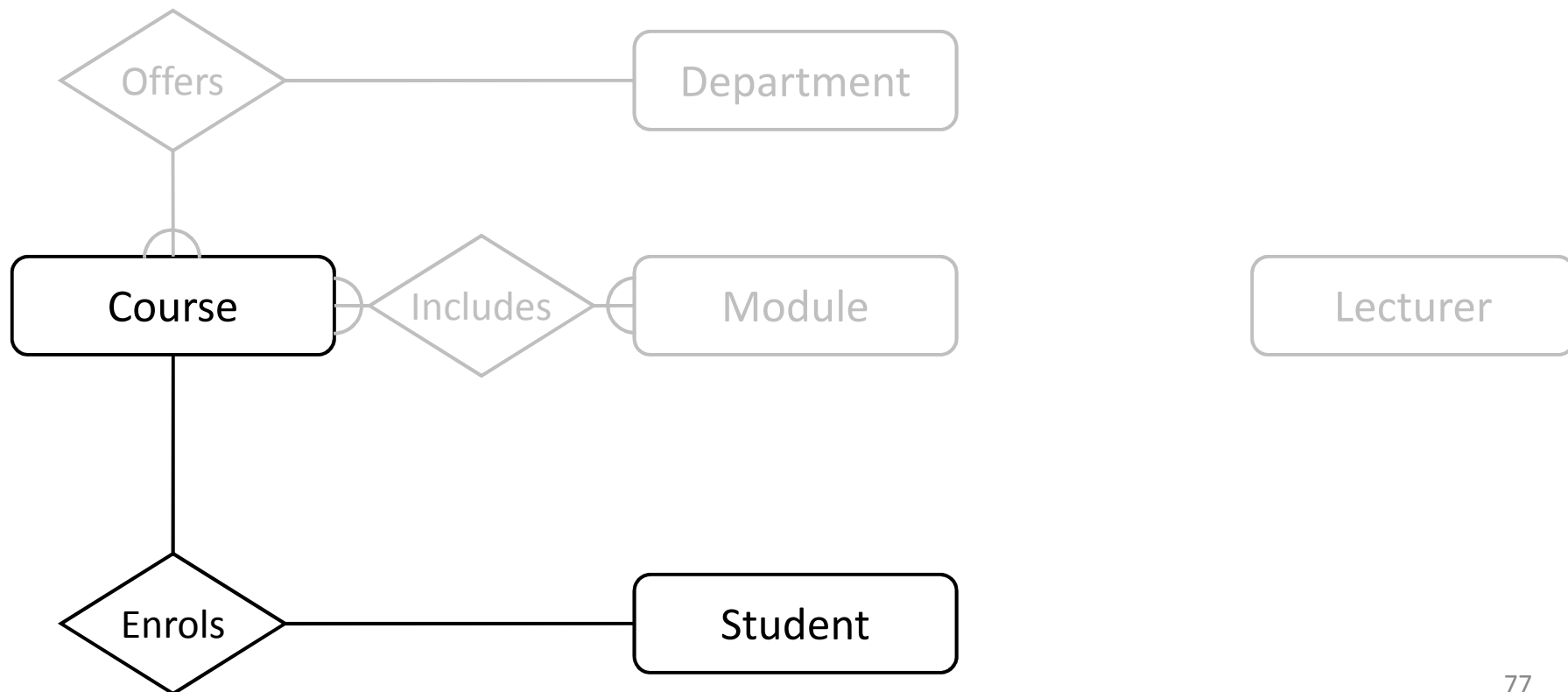
# Example – E/R Diagram

Students enrol in a particular course



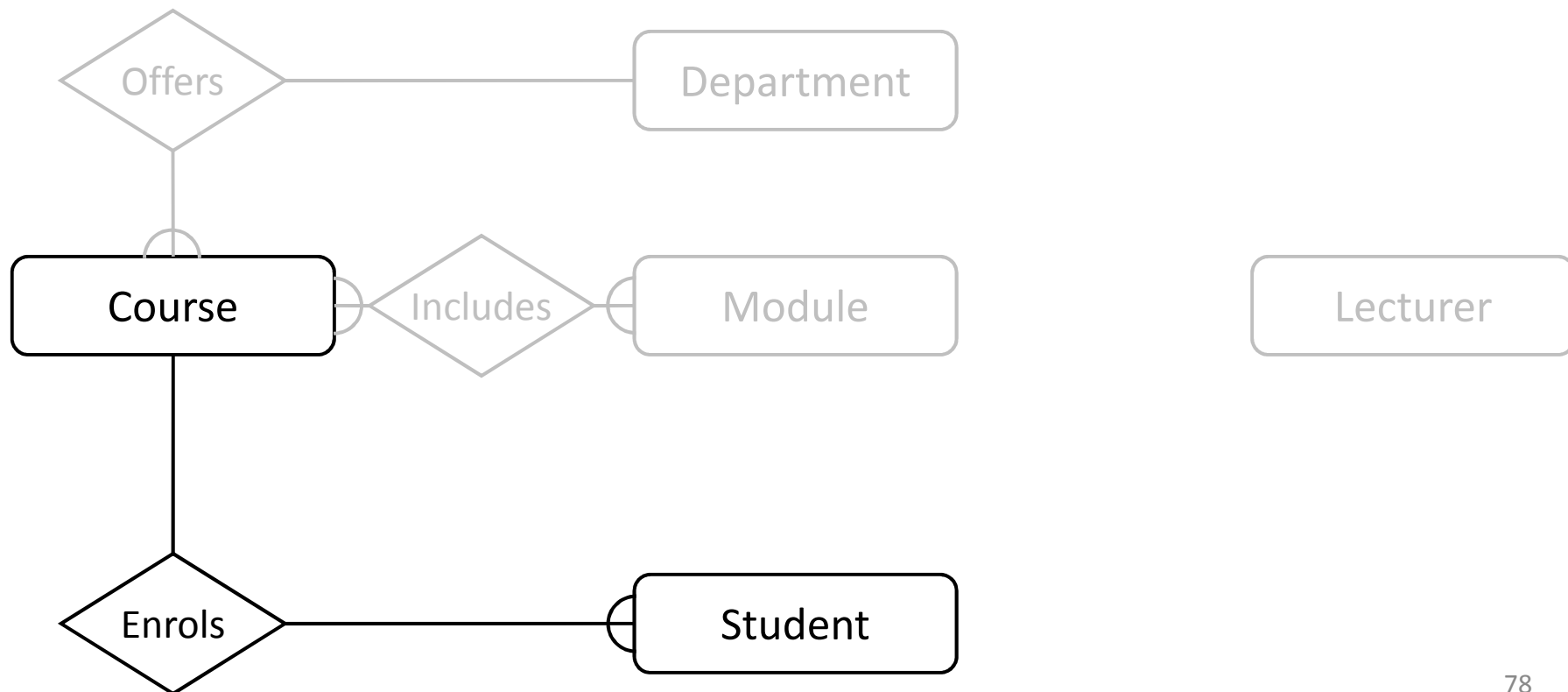
# Example – E/R Diagram

Students enrol in a particular course



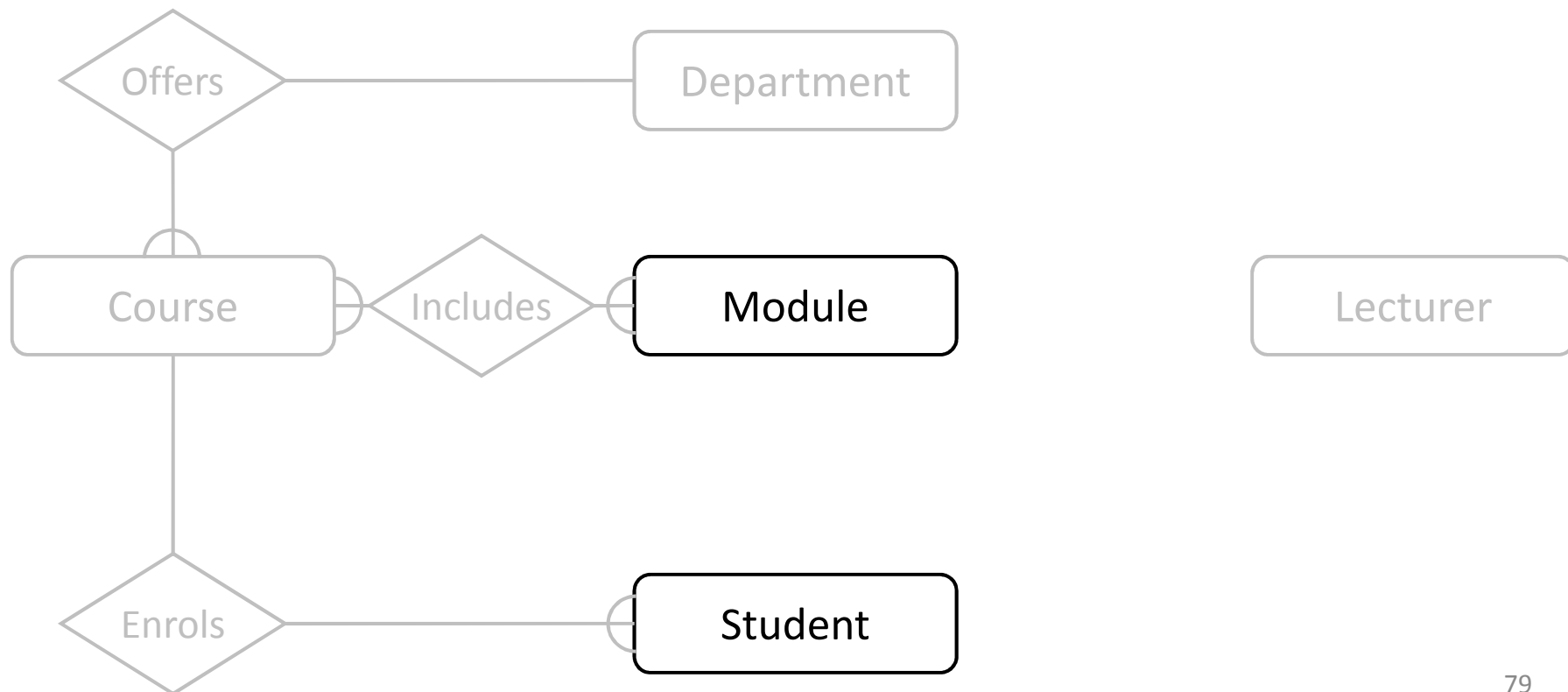
# Example – E/R Diagram

Students enrol in a particular course



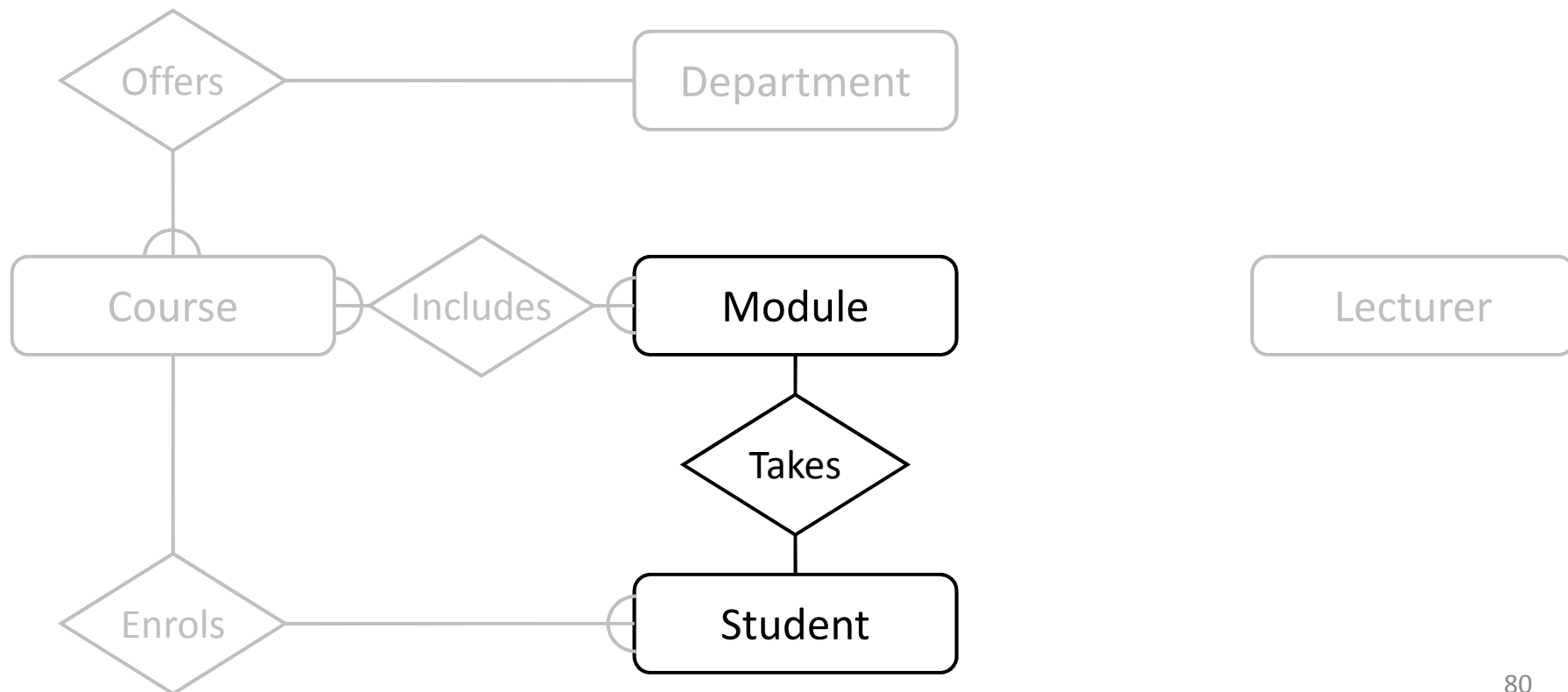
# Example – E/R Diagram

Students take several modules



# Example – E/R Diagram

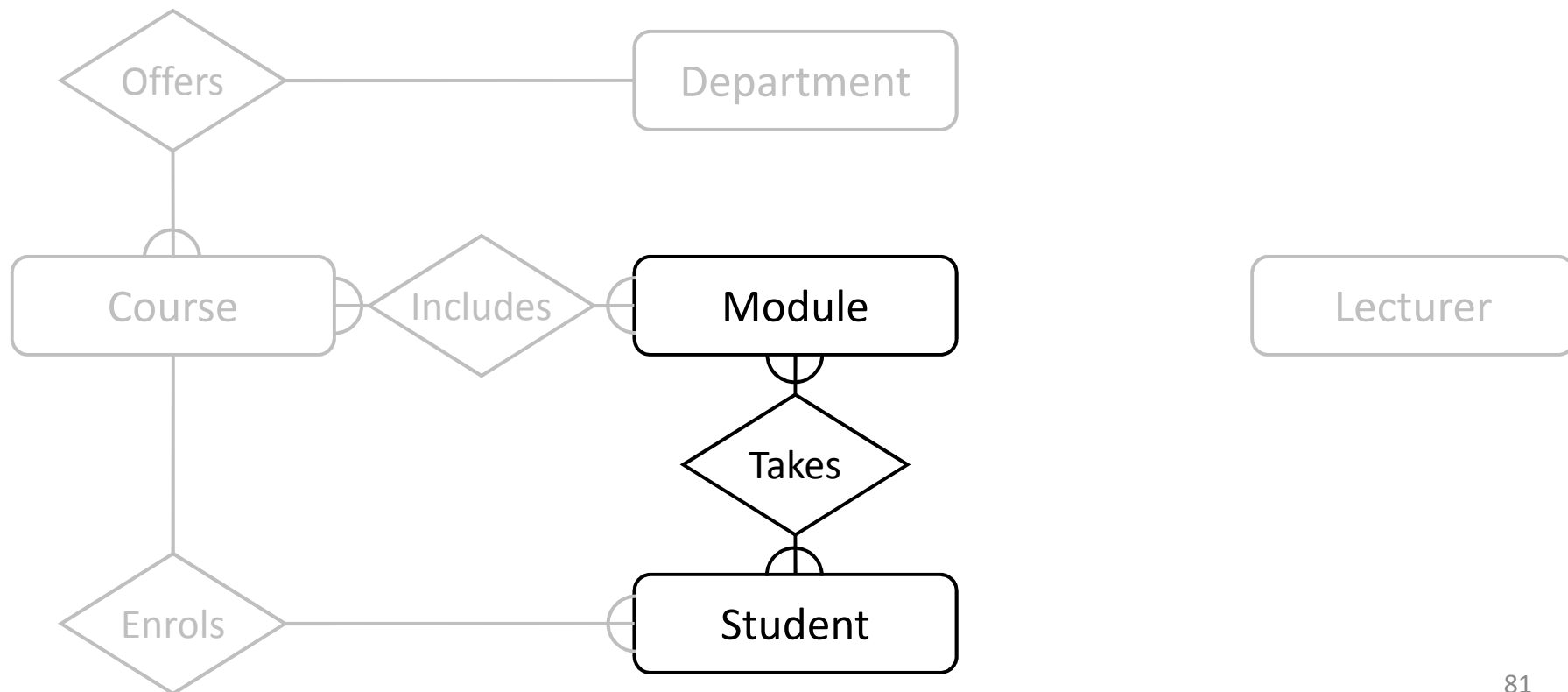
Students take several modules





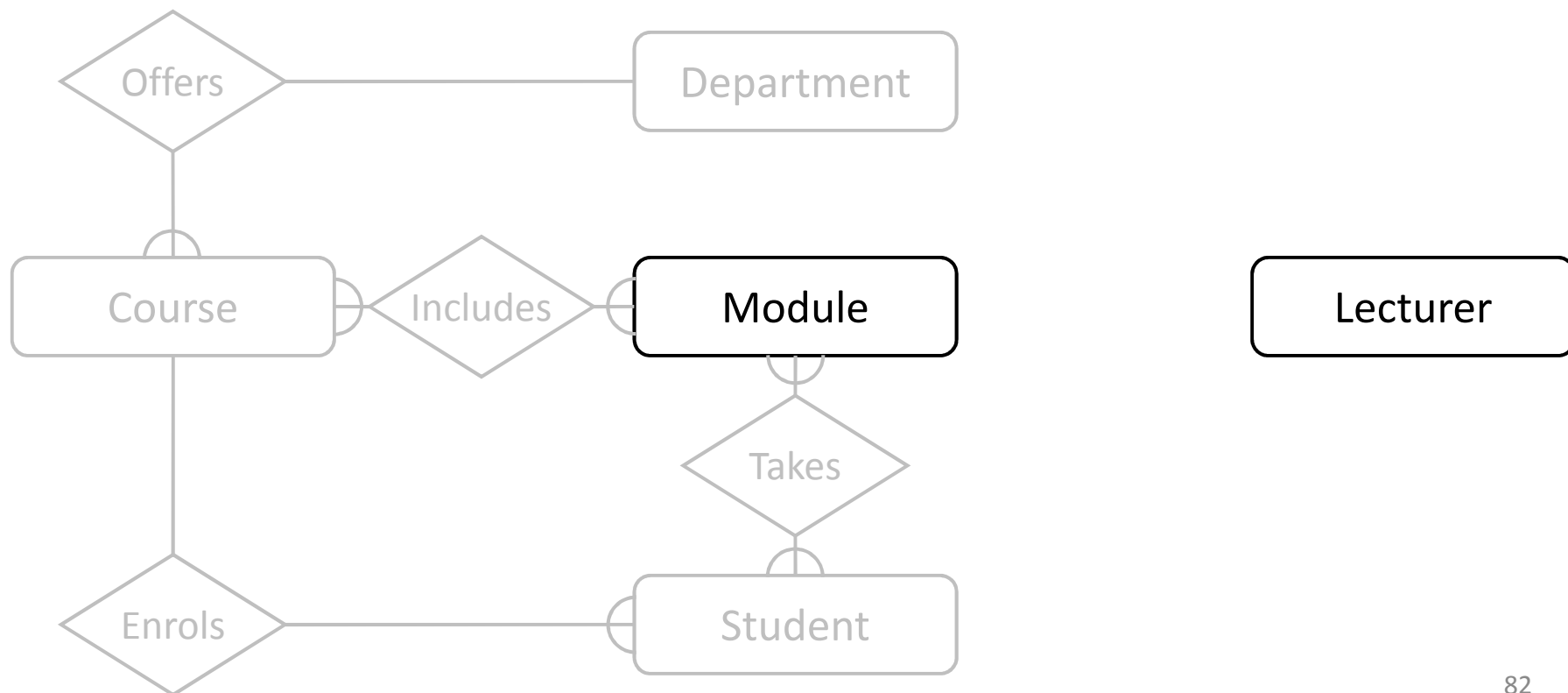
# Example – E/R Diagram

Students take several modules



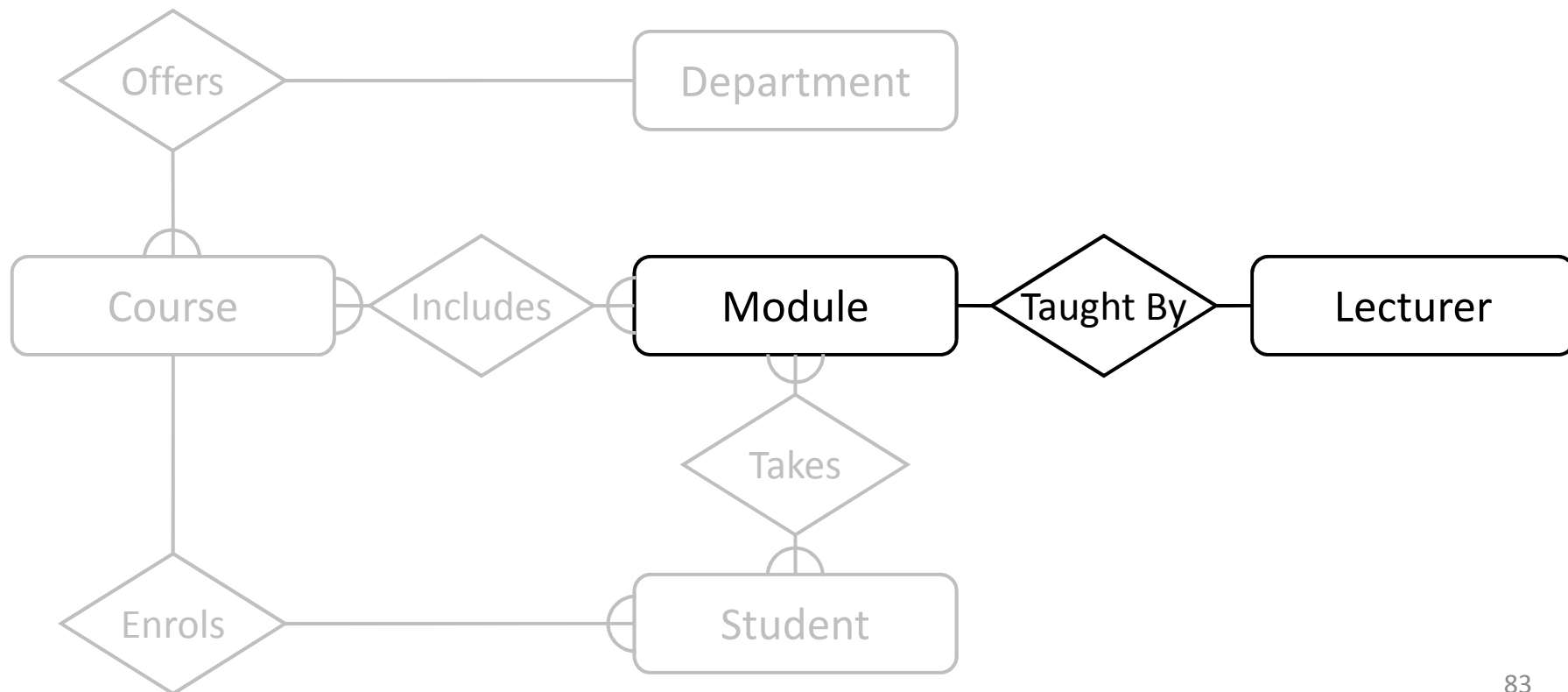
# Example – E/R Diagram

Each Module is taught by a Lecturer



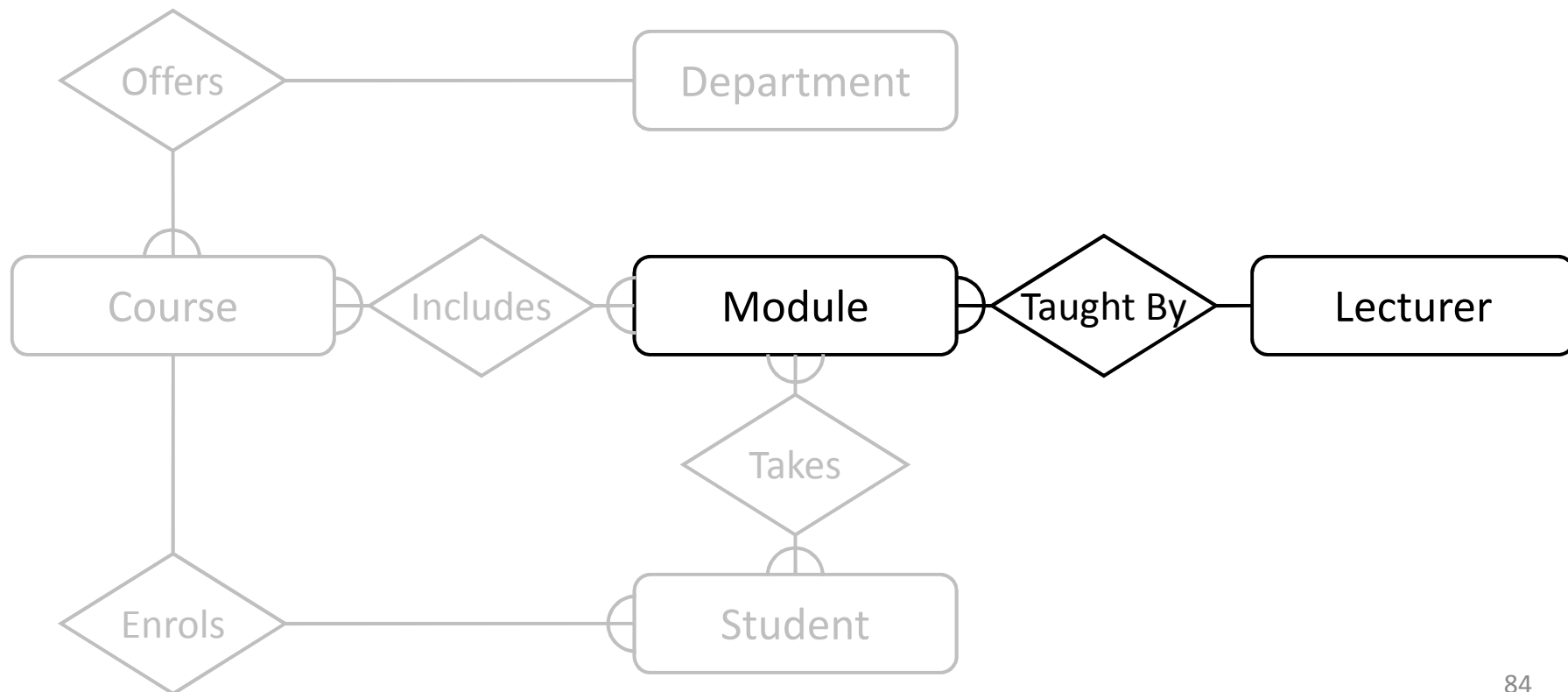
# Example – E/R Diagram

Each Module is taught by a Lecturer



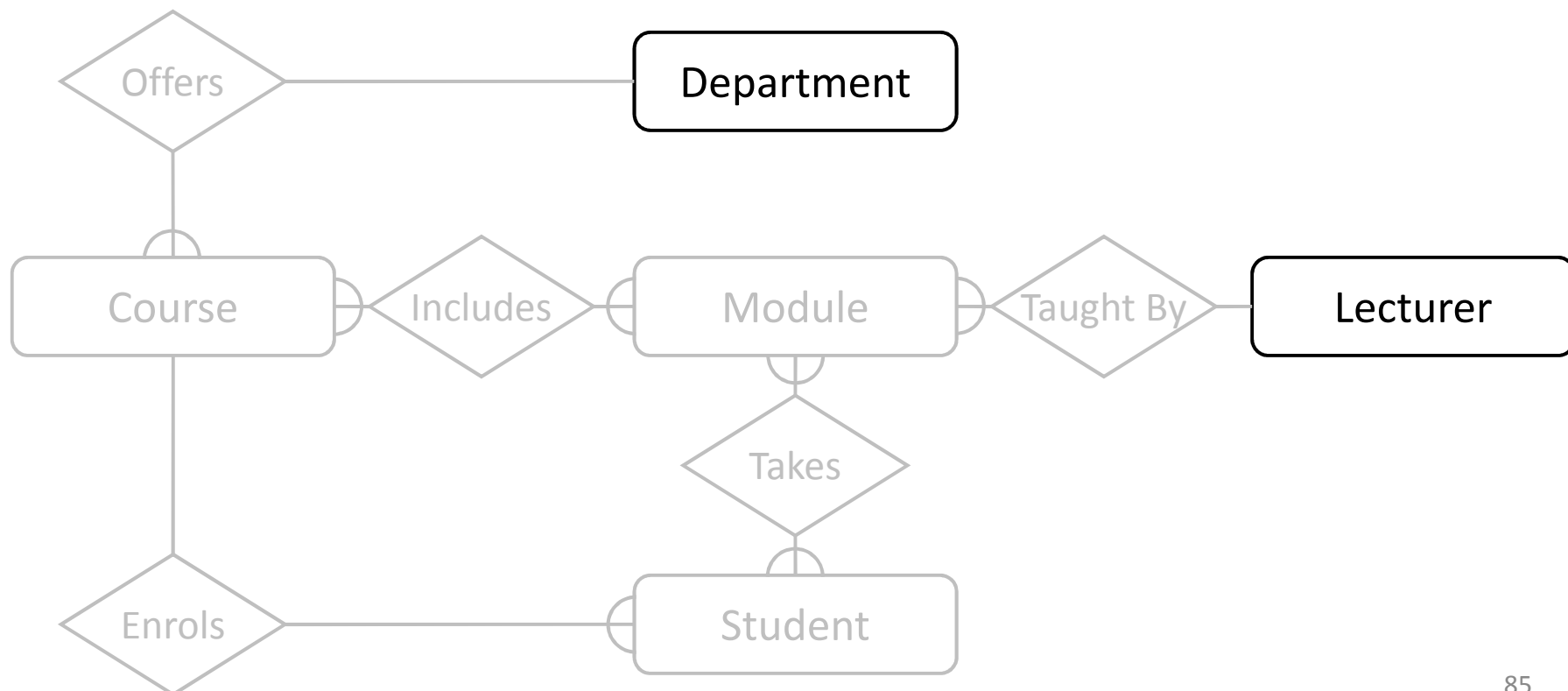
# Example – E/R Diagram

Each Module is taught by a Lecturer



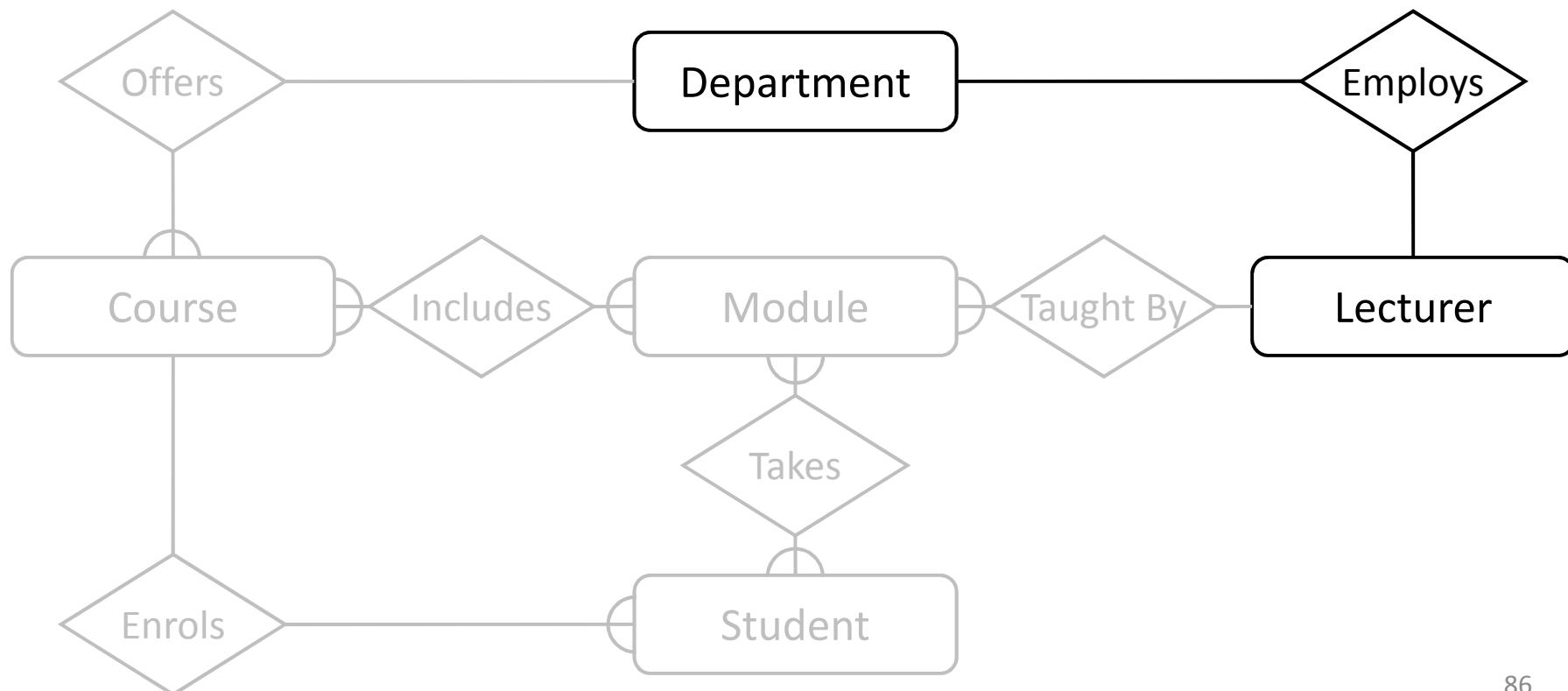
# Example – E/R Diagram

Each department employs a number of lecturers



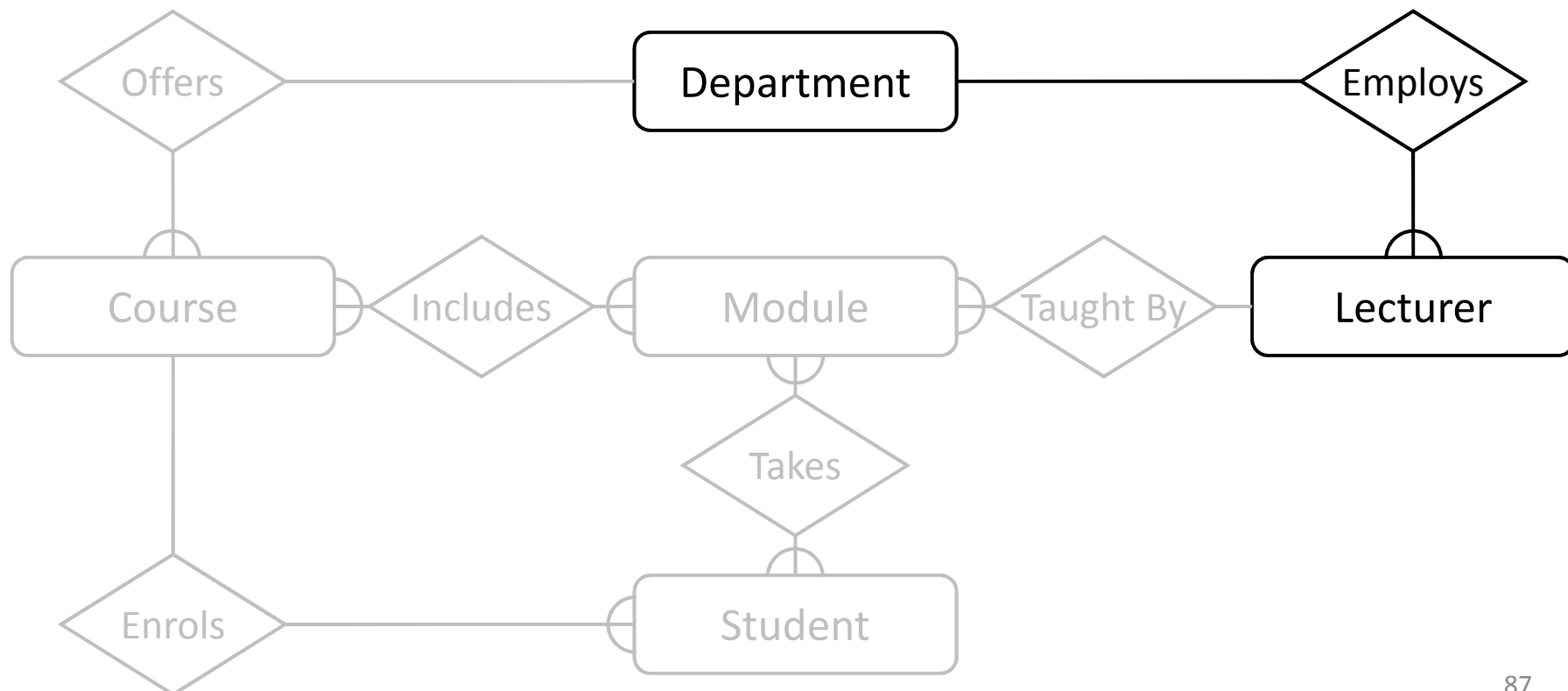
# Example – E/R Diagram

Each department employs a number of lecturers



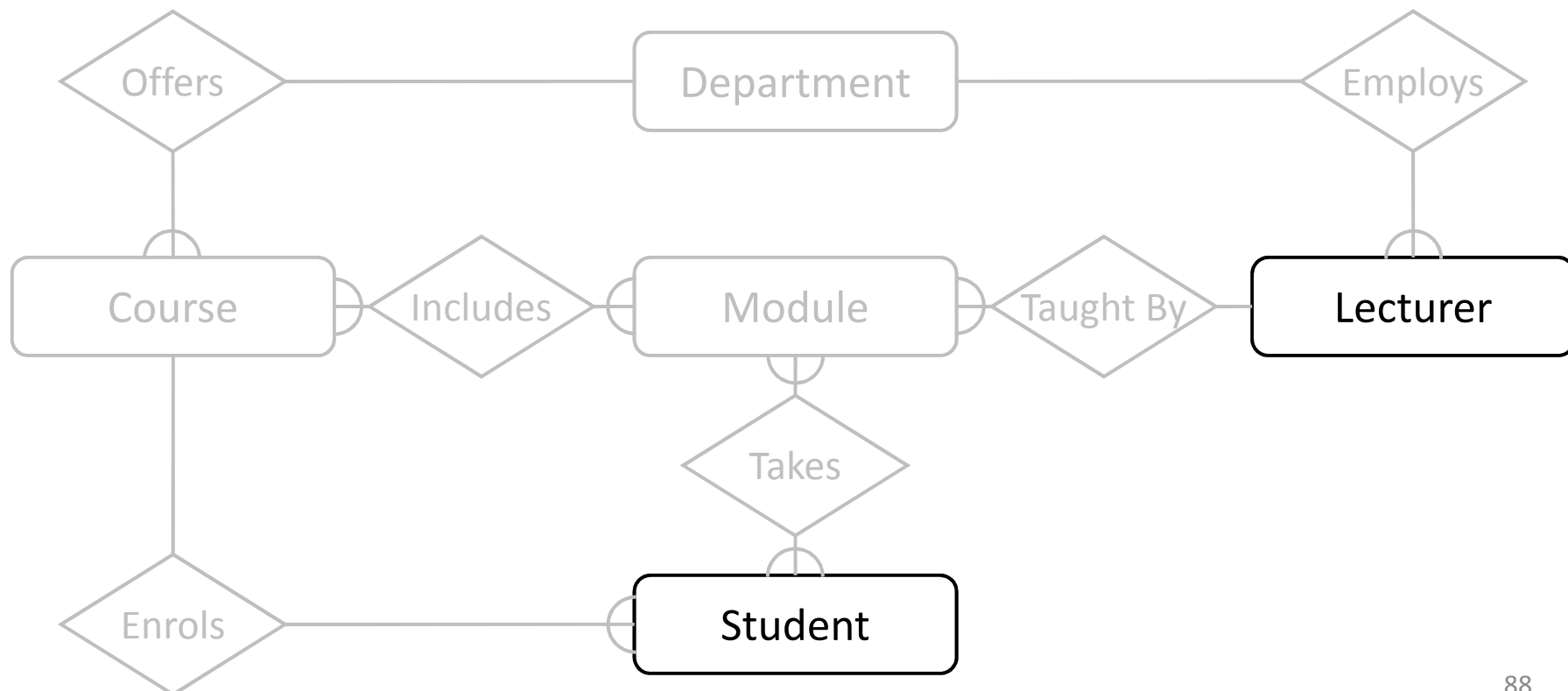
# Example – E/R Diagram

Each department employs a number of lecturers



# Example – E/R Diagram

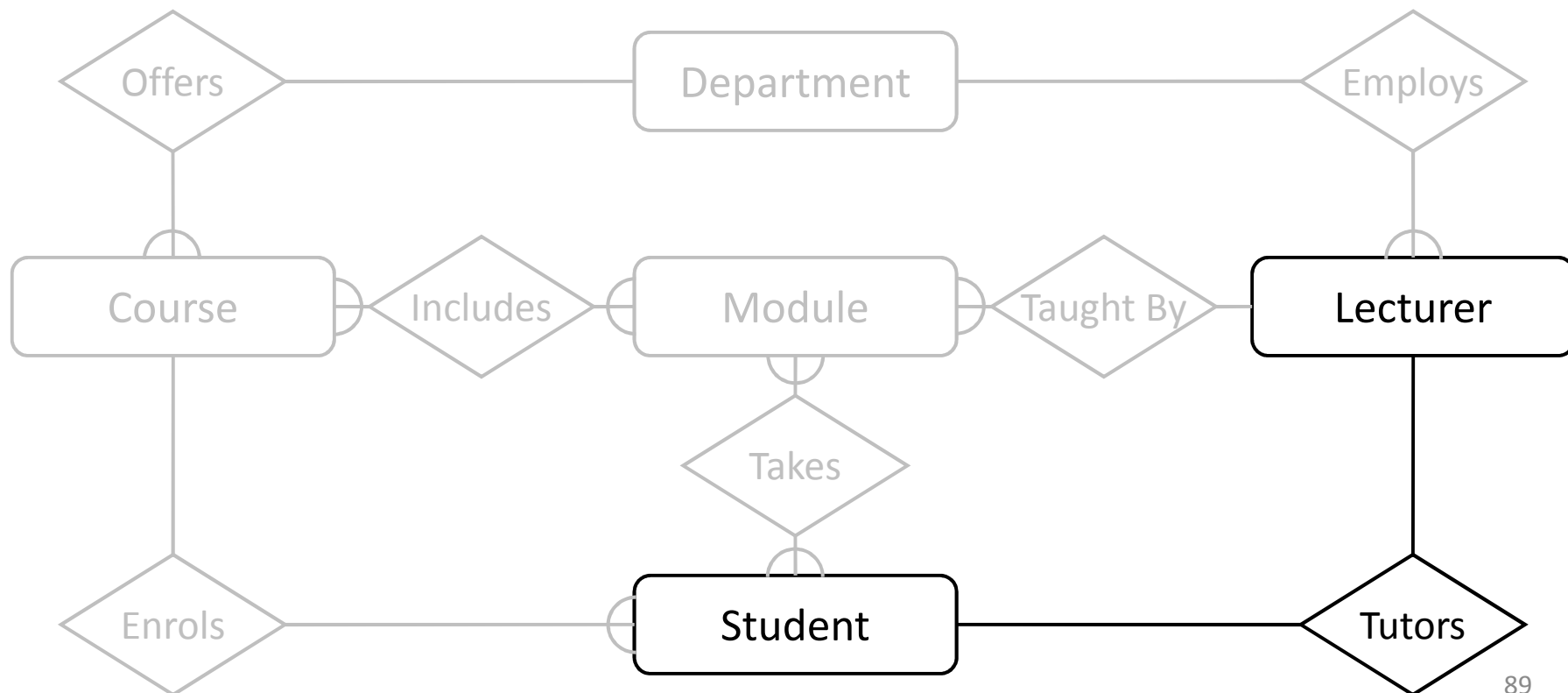
Each Lecturer tutors a number of Students





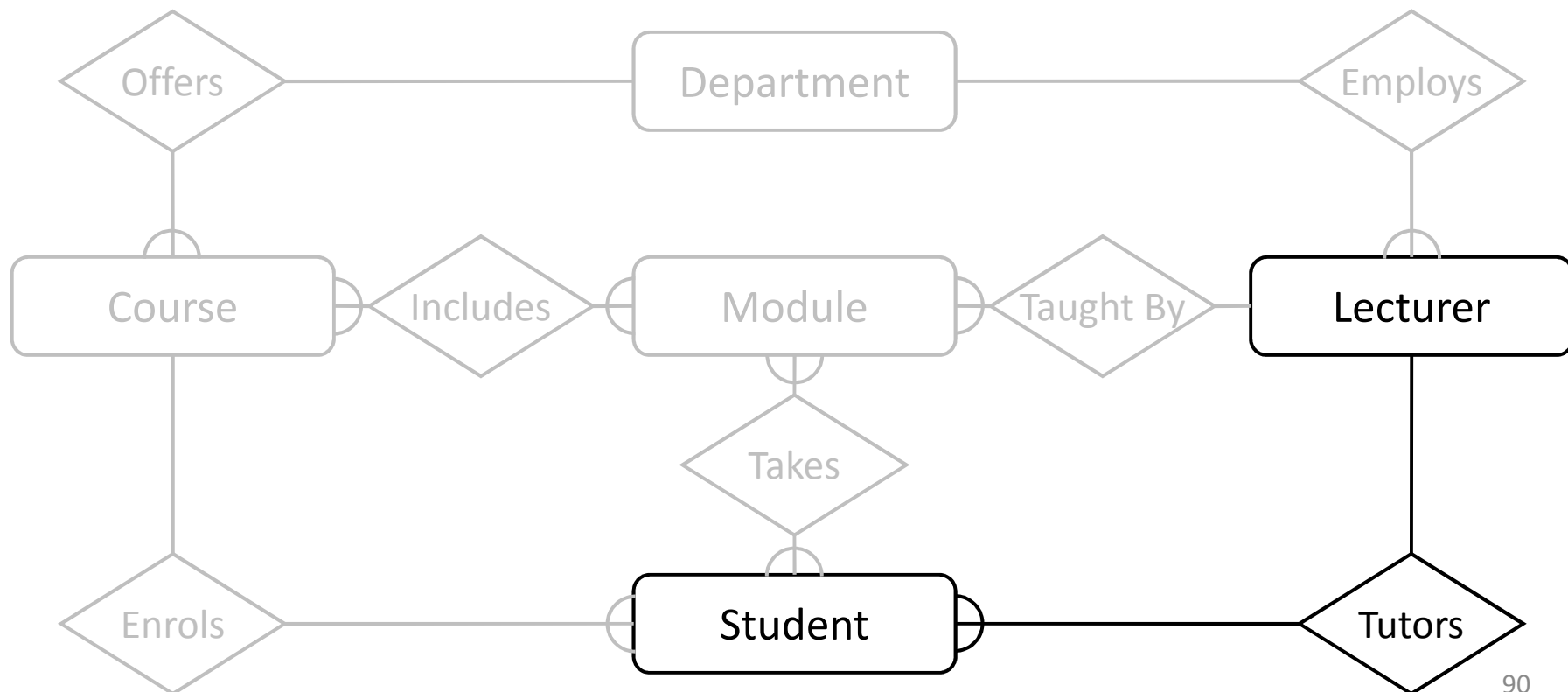
# Example – E/R Diagram

Each Lecturer tutors a number of Students



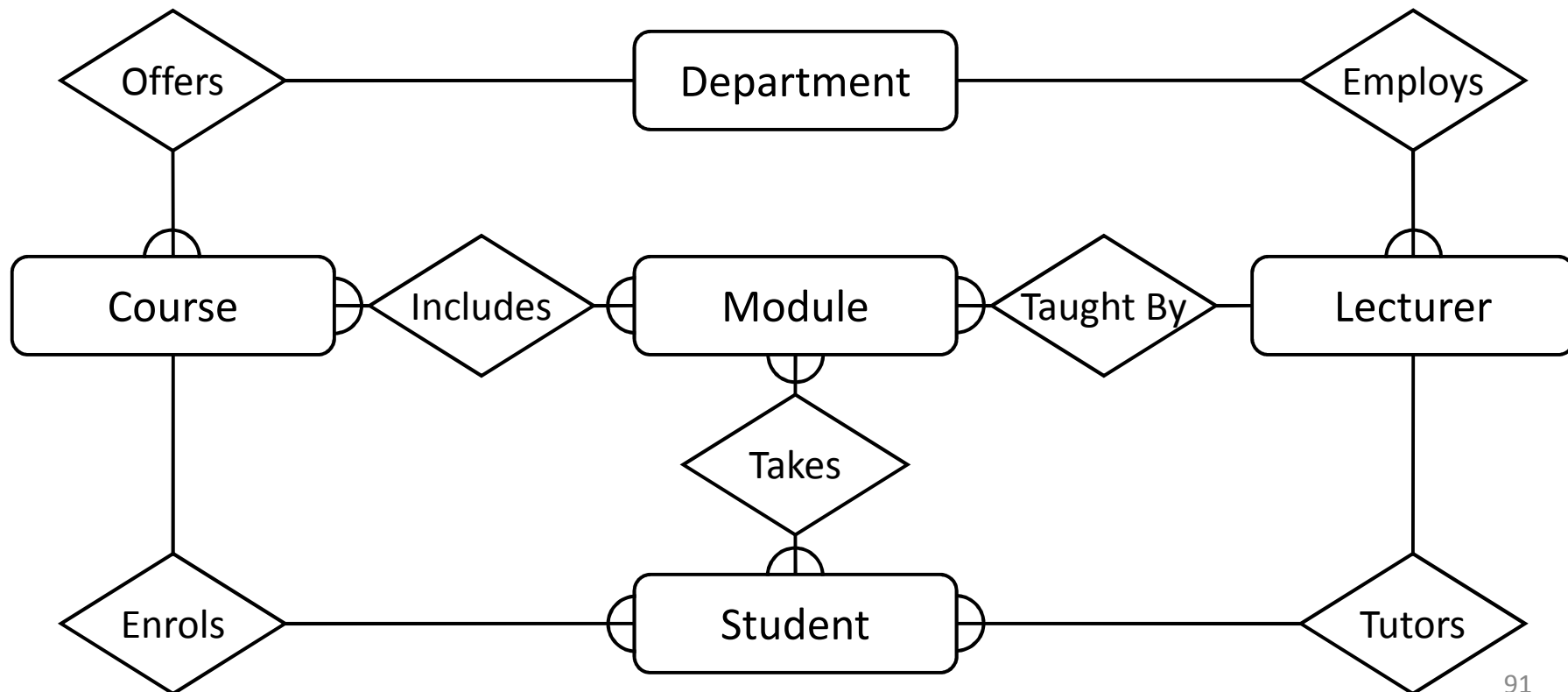
# Example – E/R Diagram

Each Lecturer tutors a number of Students



# Example – E/R Diagram

The completed diagram. All that remains is to remove M:M relationships



# Removing M:M Relationships

- Many to many relationships are difficult to represent in a database:

Student

SID	SName	SMods
1001	Jack Smith	DBS, PRG, IAI
1002	Anne Jones	PRG, IAI, VIS

Student

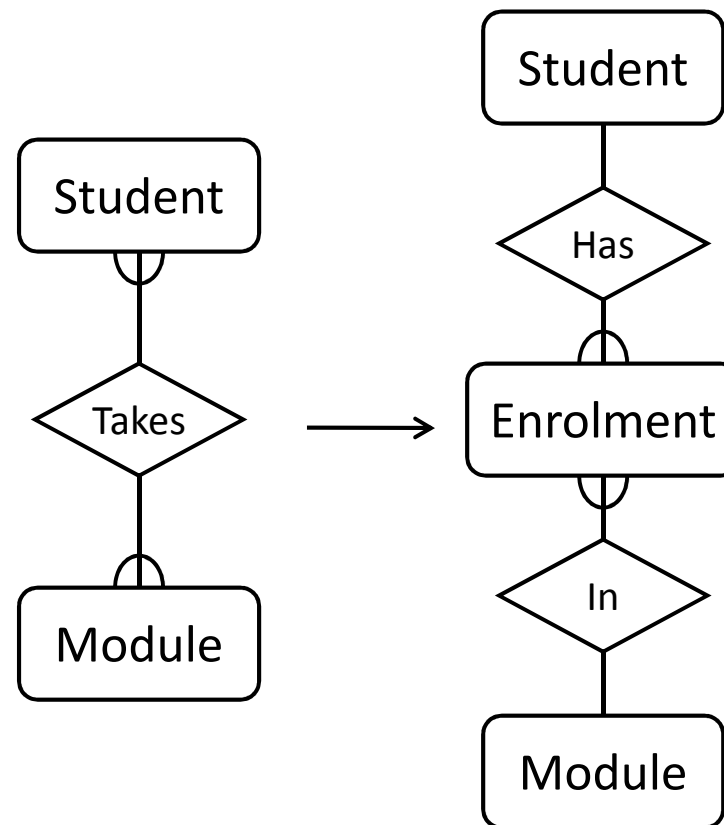
SID	SName	SMod
1001	Jack Smith	DBS
1001	Jack Smith	PRG
1001	Jack Smith	IAI
1002	Anne Jones	PRG
1002	Anne Jones	IAI
1002	Anne Jones	VIS

Module

MID	MName
DBS	Database Systems
PRG	Programming
IAI	AI
VIS	Computer Vision

# Removing M:M Relationships

- Many to many relationships are difficult to represent in a database
- We can split a many to many relationship into two one to many relationships
- An additional entity is created to represent the M:M relationship



# Entities and Attributes

- Sometimes it is hard to tell if something should be an entity or an attribute
  - They both represent objects or facts about the world
  - They are both often represented by nouns in descriptions
- General guidelines
  - Entities can have attributes but attributes have no smaller parts
  - Entities can have relationships between them, but an attribute belongs to a single entity

# Example

We want to represent information about products in a database. Each product has a description, a price and a supplier. Suppliers have addresses, phone numbers, and names. Each address is made up of a street address, a city, and a postcode.

# Example

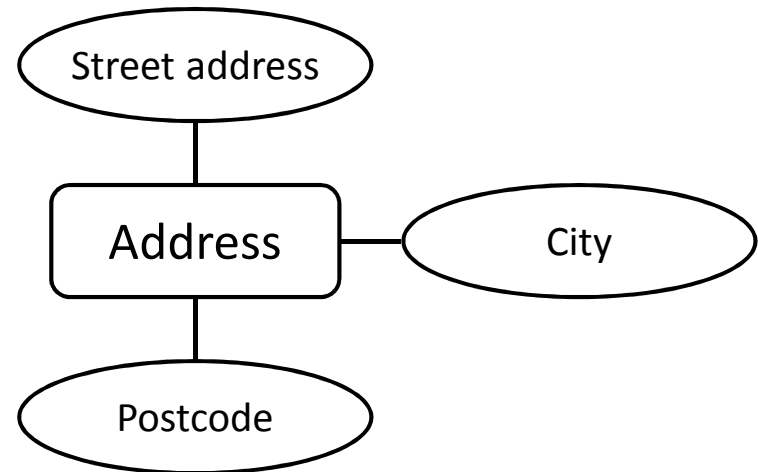
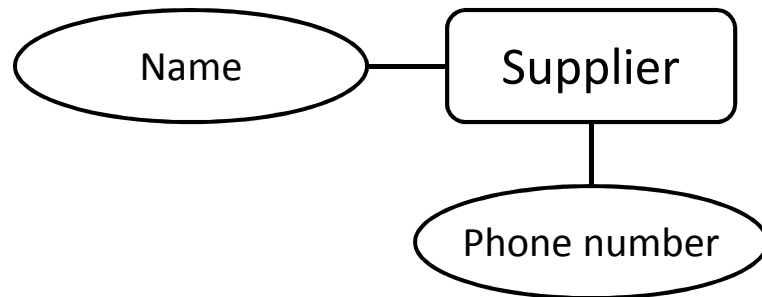
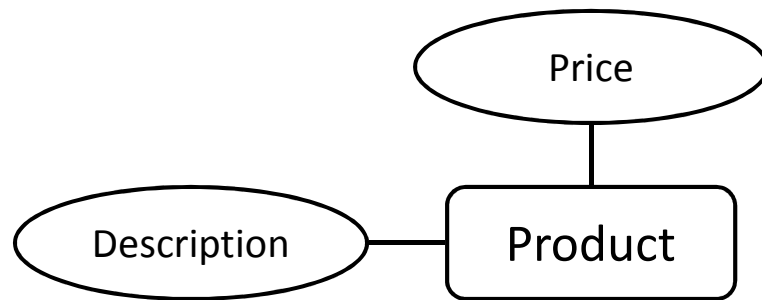
We want to represent information about products in a database. Each **product** has a **description**, a **price** and a **supplier**. **Suppliers** have **addresses**, **phone numbers**, and **names**. Each **address** is made up of a **street address**, a **city**, and a **postcode**.



# Example - Entities/Attributes

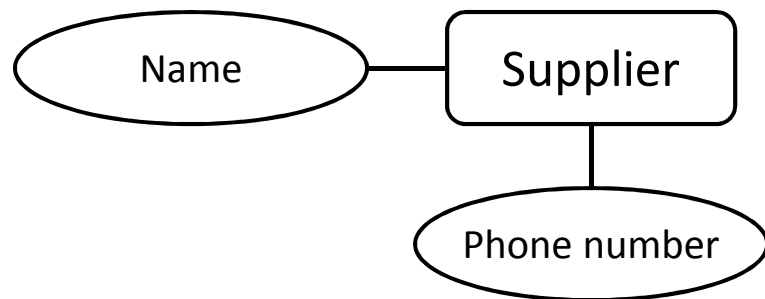
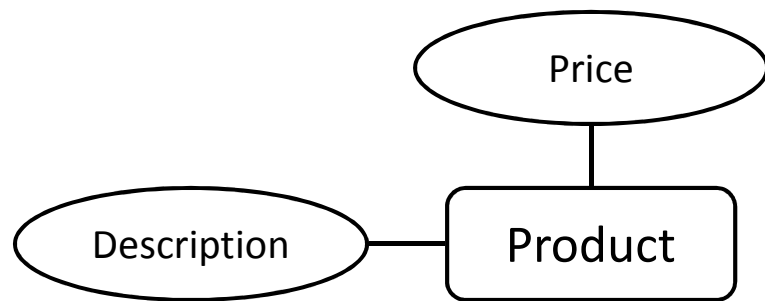
- Entities or attributes:
  - **product**
  - description
  - price
  - **supplier**
  - **address**
  - phone number
  - name
  - street address
  - city
  - postcode
- Products, suppliers, and addresses all have smaller parts so we make them entities
- The others have no smaller parts and belong to a single entity

# Example - E/R Diagram

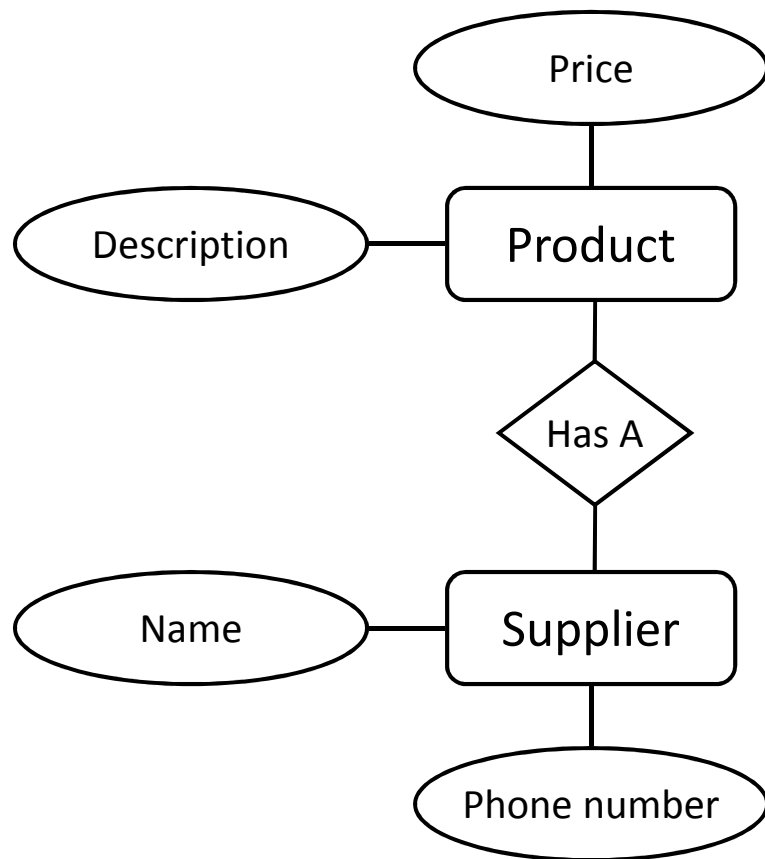


# Example - Relationships

- Each product has a supplier

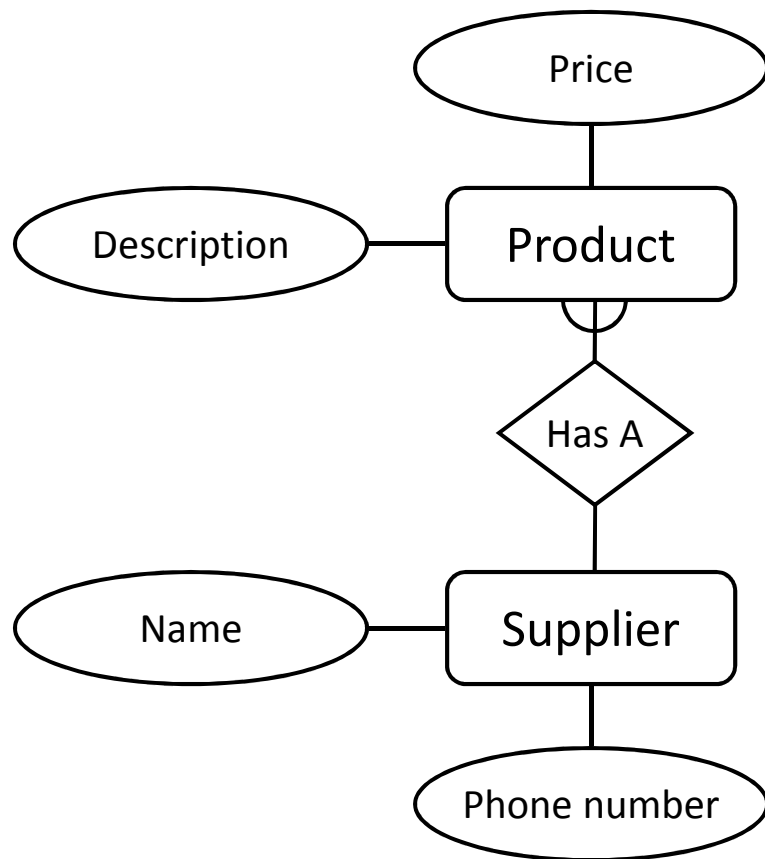


# Example - Relationships



- Each product has a supplier
  - Each product has a single supplier
  - but there is nothing to stop a supplier supplying many products
- A many to one relationship

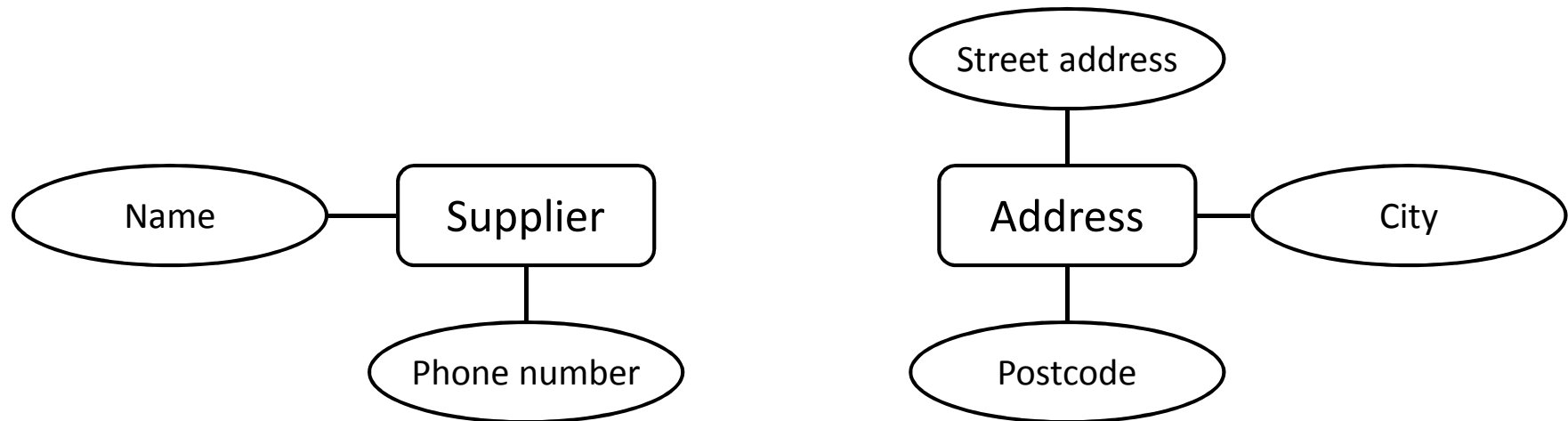
# Example - Relationships



- Each product has a supplier
  - Each product has a single supplier
  - but there is nothing to stop a supplier supplying many products
- A many to one relationship

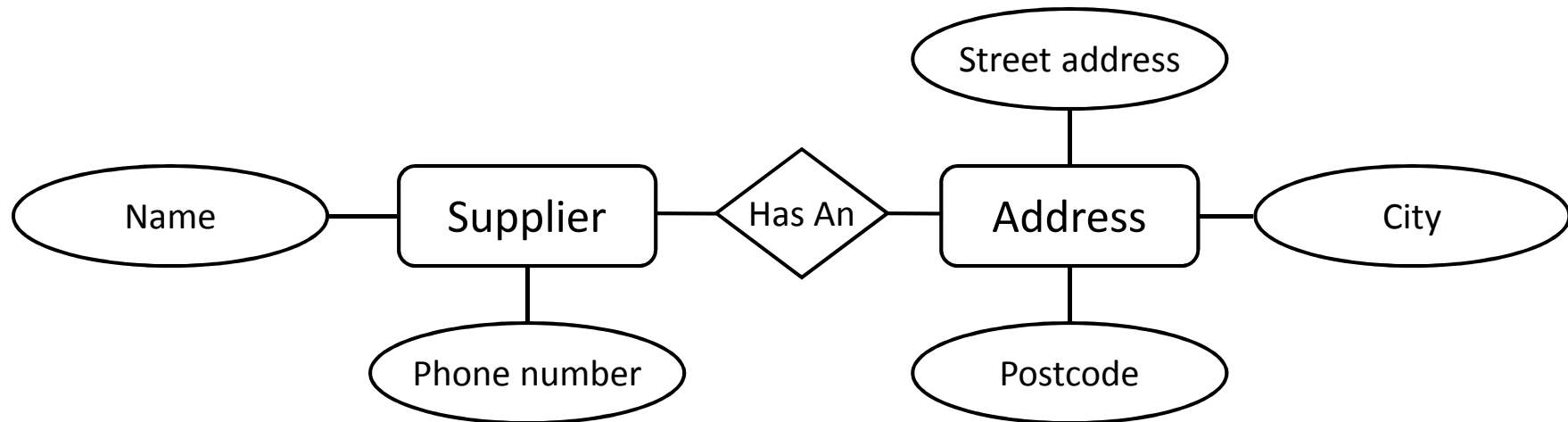
# Example - Relationships

- Each supplier has an address

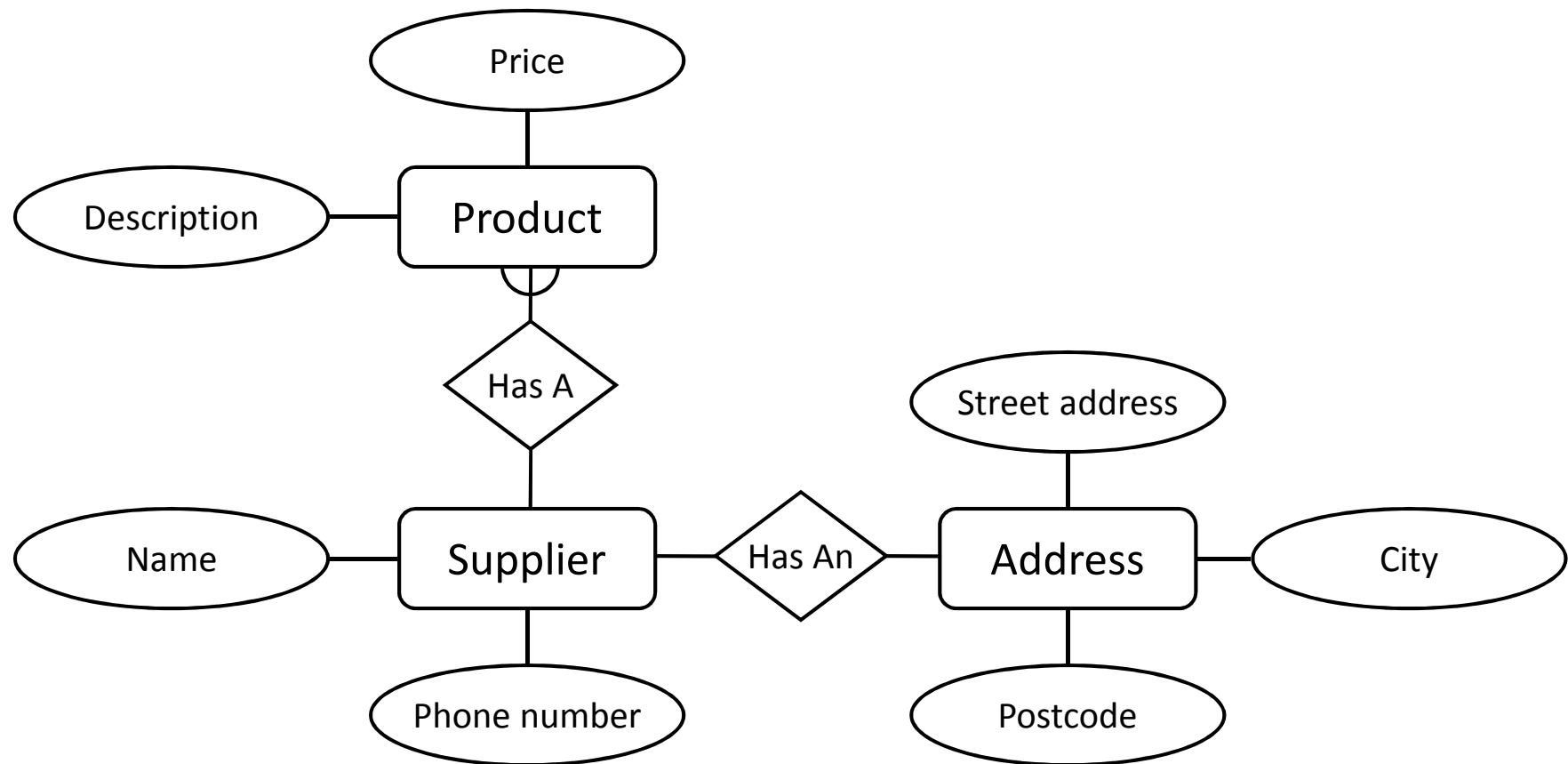


# Example - Relationships

- Each supplier has an address
  - A supplier has a single address
  - It does not seem sensible for two different suppliers to have the same address
  - A one to one relationship



# Example - E/R Diagram



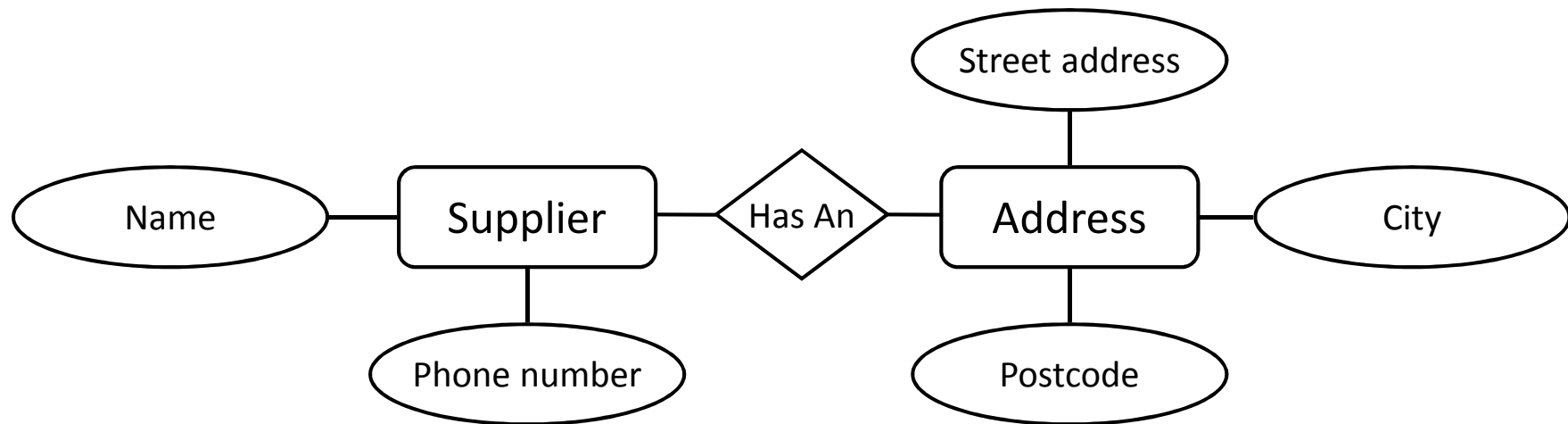


# One to One Relationships

- *Some* relationships between entities, A and B, *might* be redundant if:
  - It is a 1:1 relationship between A and B
  - Every A is related to a B and every B is related to an A

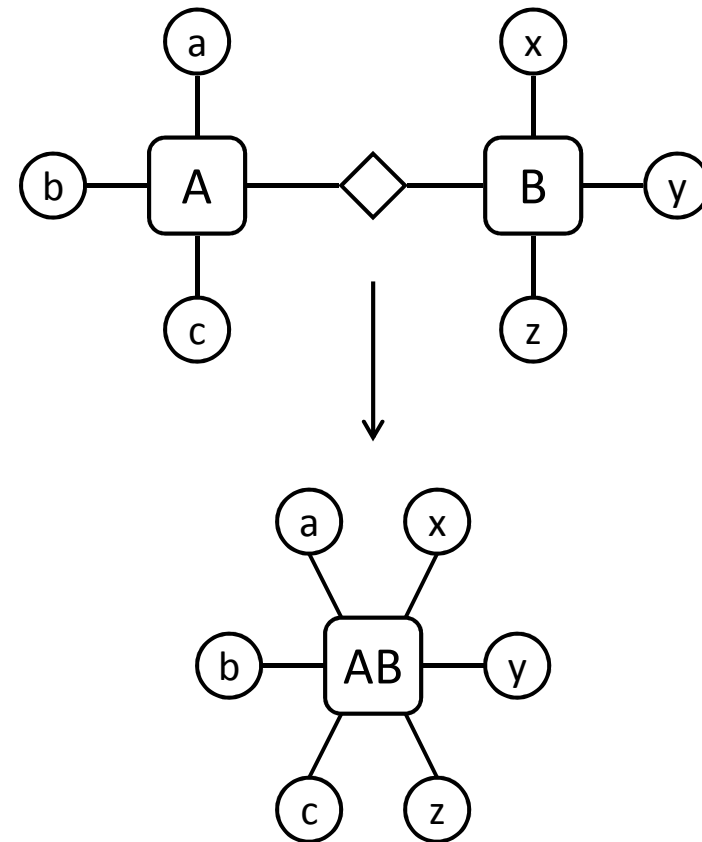
# Example One to One Relationship

- Supplier-address relationship
  - Is one to one
  - Every supplier has an address
  - We don't need addresses that are not related to a supplier

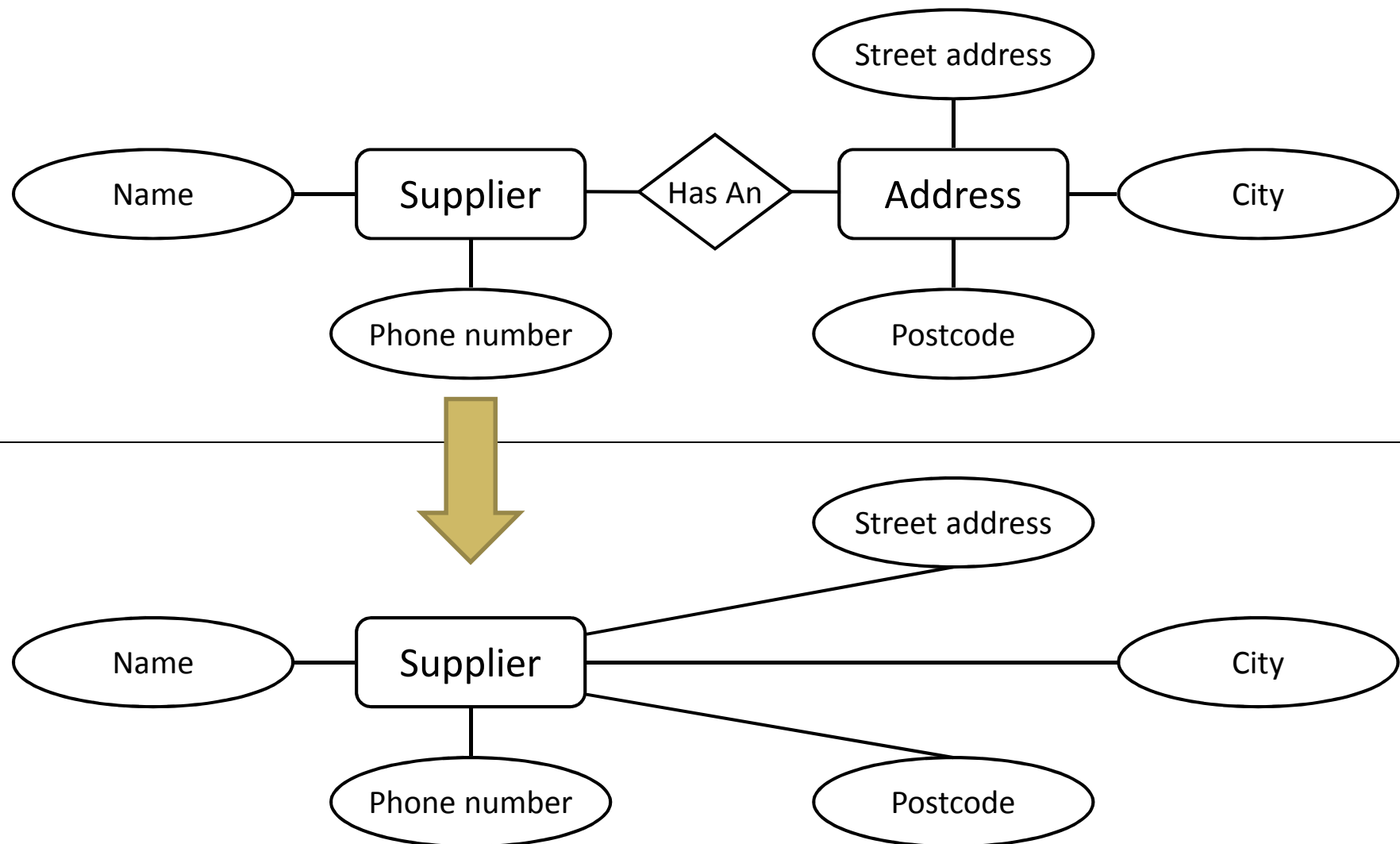


# Redundant Relationships

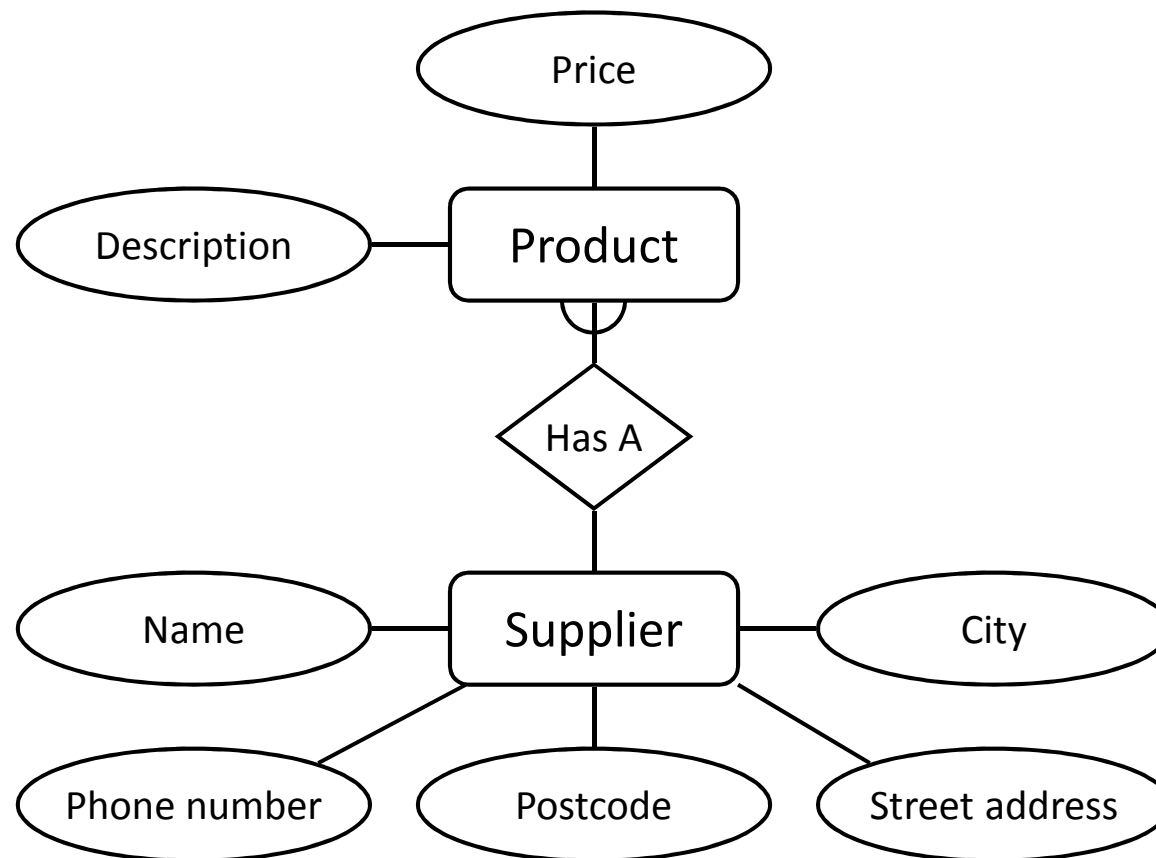
- We can merge the two entities that take part in a redundant relationship together
  - They become a single entity
  - The new entity has all the attributes of the old ones



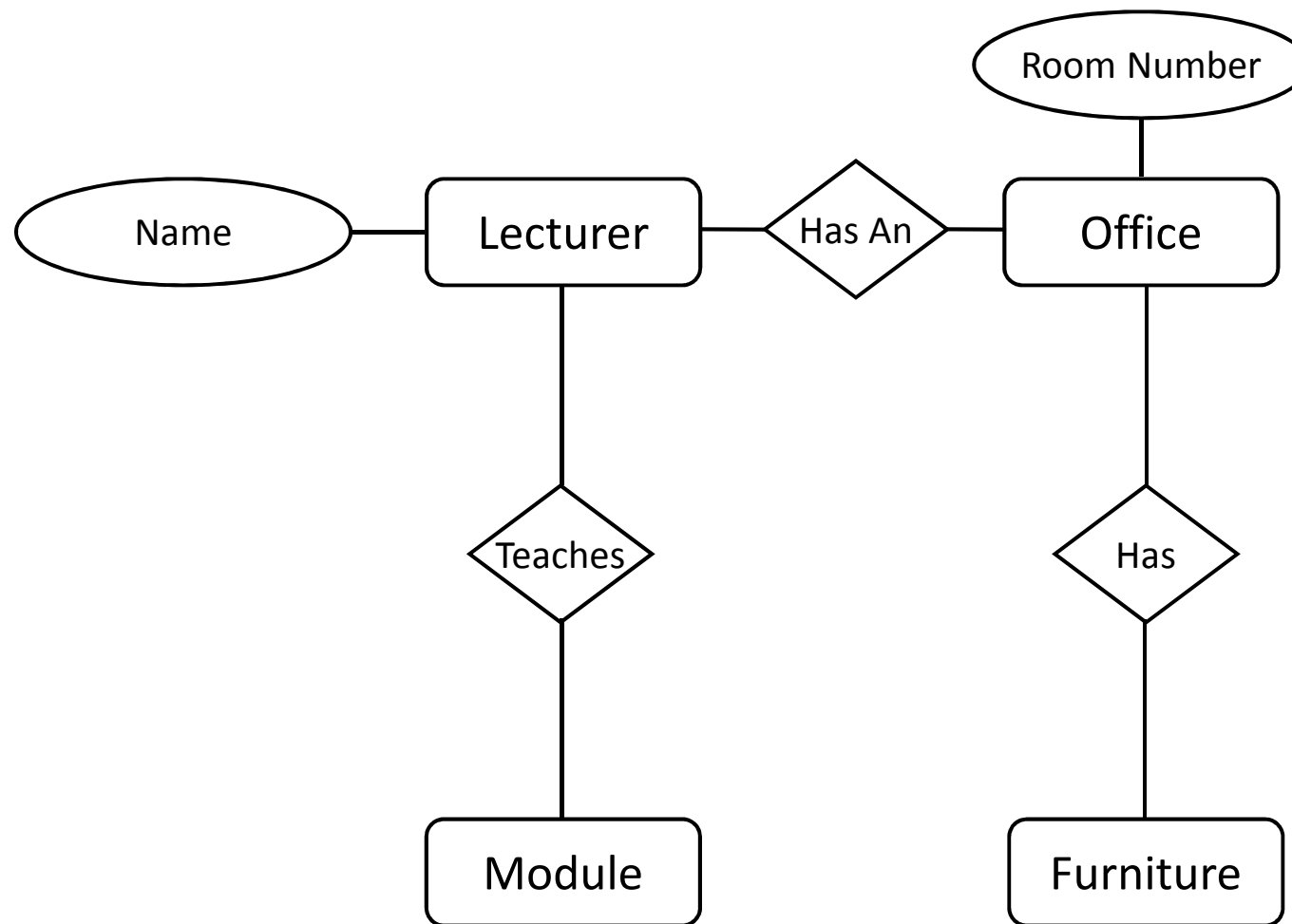
# Removing the 1:1 Relationship



# Example - E/R Diagram



# Keeping some 1:1 relationships?



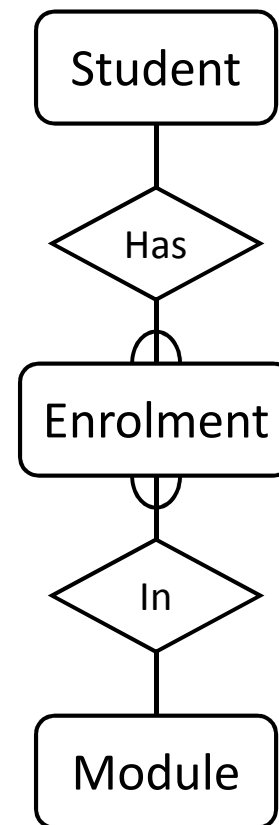
**Coursework:**  
If you keep 1:1 relationships, ensure that you have a reason **and explain it**  
**(Same applies to the exam)**

# Making E/R Diagrams

- From a description of the requirements identify:
  - Entities
  - Attributes
  - Relationships
  - Cardinality ratios of the relationships
- Draw the E/R diagram
- Look at one to one relationships as they **might** be redundant
- Look at many to many relationships as they will often need to be split into two one to many links, using an intermediate entity

# Debugging Designs

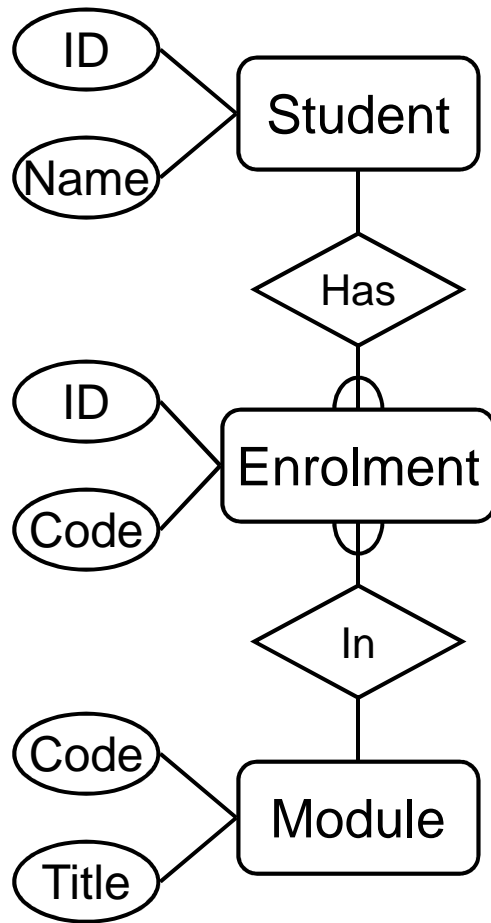
- With a bit of practice E/R diagrams can be used to plan queries
  - You can look at the diagram and figure out how to find useful information
  - If you can't find the information you need, you may need to change the design



How can you find a list of students who are enrolled in Database systems?

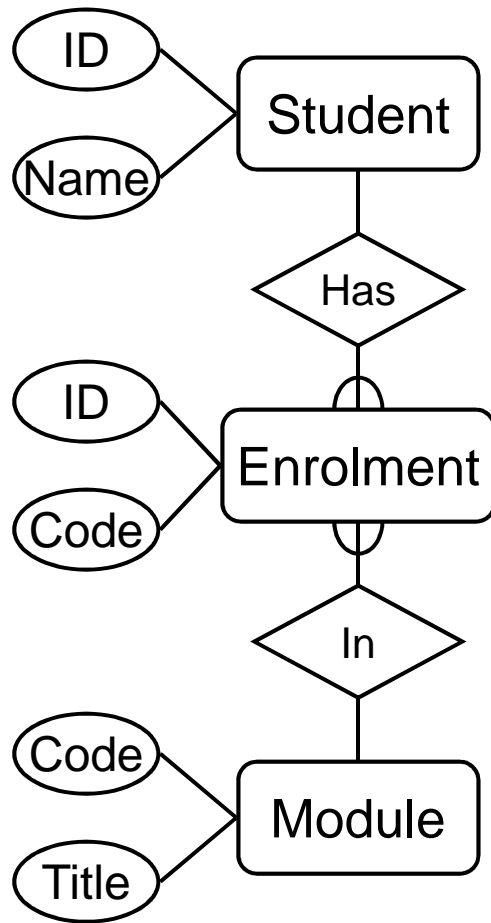


# Debugging Designs



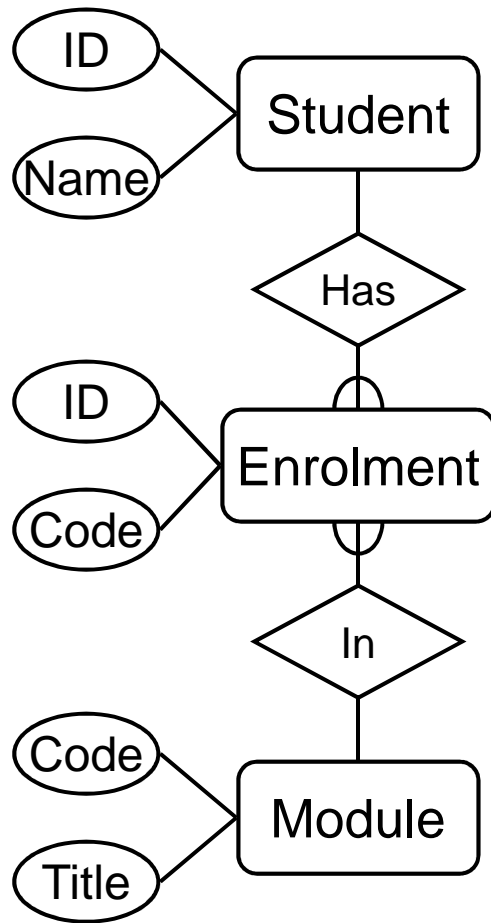
1. Find the instance of Module with the title 'Database Systems'

# Debugging Designs



1. Find the instance of Module with the title 'Database Systems'
2. Find instances of the Enrolment entity with the same Code as the result of (1)

# Debugging Designs



1. Find the instance of Module with the title 'Database Systems'
2. Find instances of the Enrolment entity with the same Code as the result of (1)
3. For each instance of Enrolment in the result of (2) find the corresponding student

# This Lecture in Exams and Coursework

- Identify the *entities, attributes, relationships, and cardinality ratios* from the description below.
- Draw an entity-relationship diagram showing the items you identified.
- Many-to-many relationships are hard to represent in database tables. Explain the nature of these problems, and describe how they may be overcome.

“A database will be created to store information about patients in a hospital. On arrival, each patient’s personal details (name, address, and telephone number) are recorded where possible, and the patient is given an admission number. They are then assigned to a particular ward (Accident and Emergency, Cardiology, Oncology, etc.). In each ward there are a number of doctors and nurses. A patient will be treated by one doctor and several nurses over the course of their stay, and each doctor and nurse may be involved with several patients at any given time.”

# Next Lecture

- SQL
  - The SQL language
  - SQL, the relational model, and E/R diagrams
  - CREATE TABLE
    - Columns
    - Primary Keys
    - Foreign Keys
- Further Reading
  - Database Systems, Connolly & Begg, Chapter 7.3
  - The Manga Guide to Databases, Chapter 4