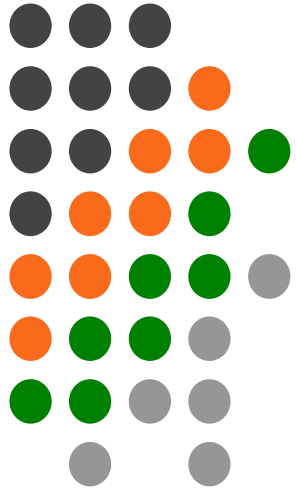
Database Fundamentals

Lecture 6 (Data Modelling Requirements)

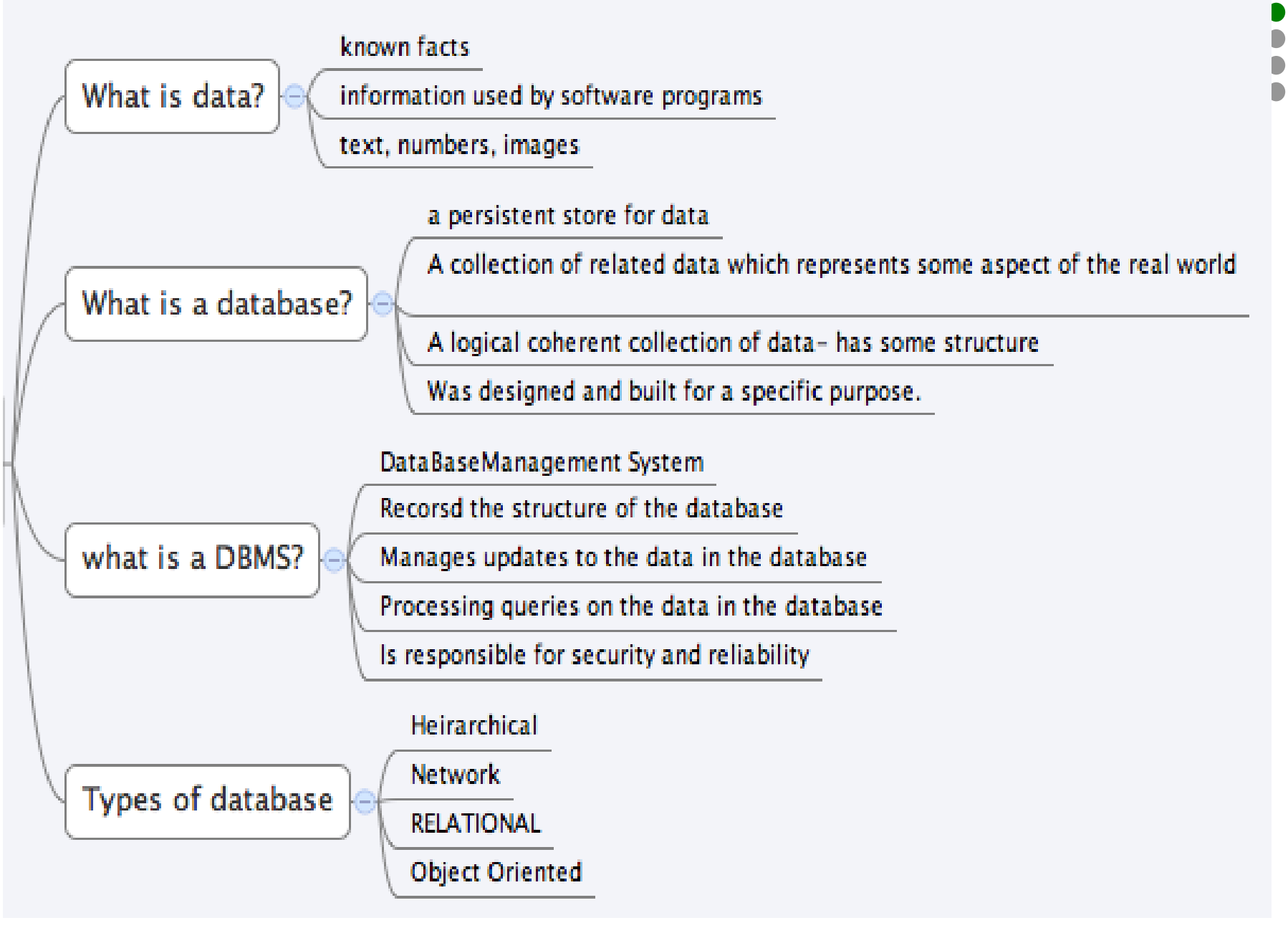


Lecturer : Irene Murtagh

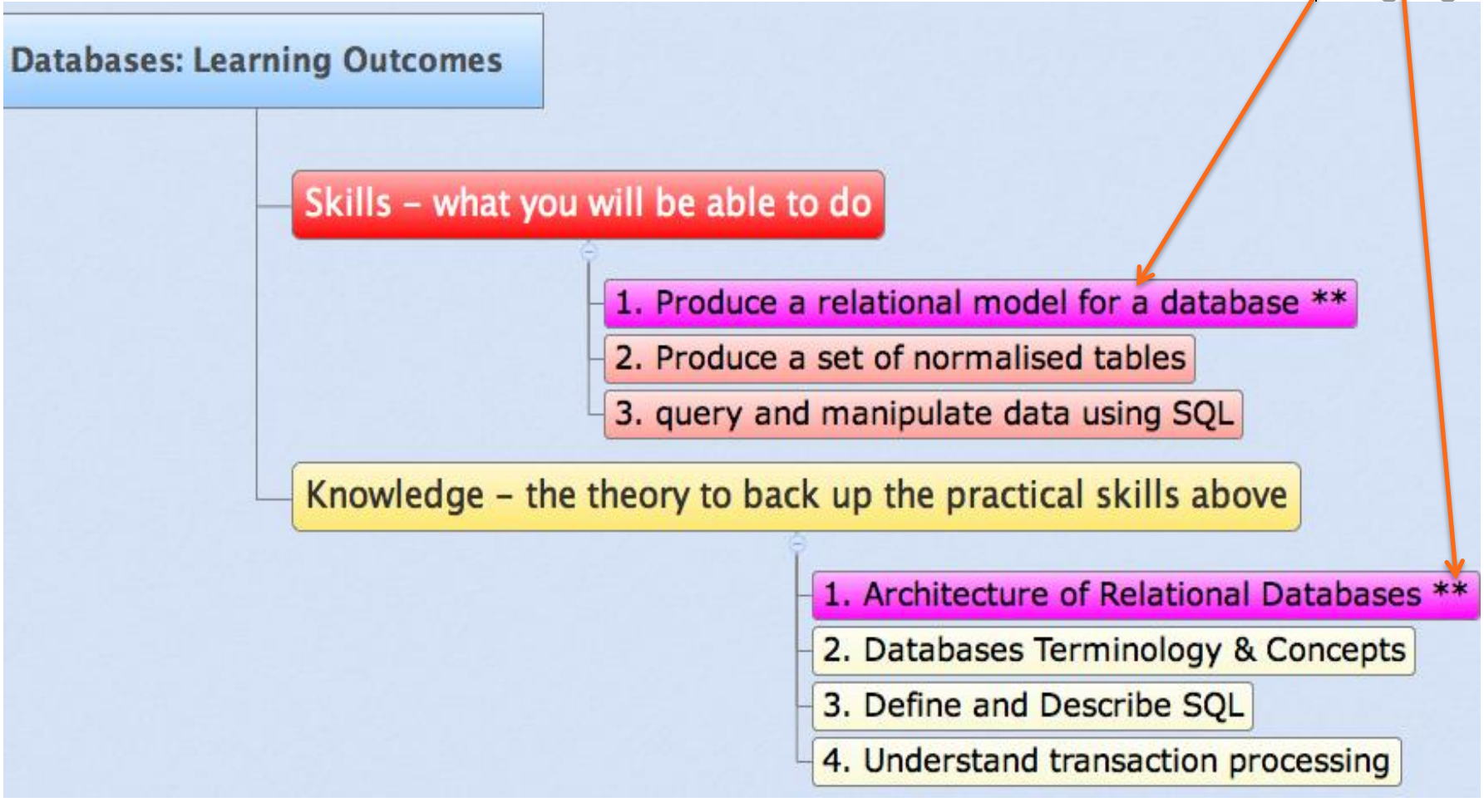
Room :A15

Email: irene.murtagh@tudublin.ie

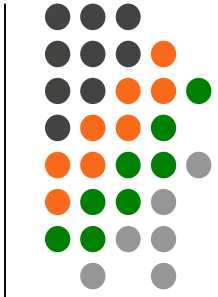
Recap on Terminology



Learning outcomes addressed today:



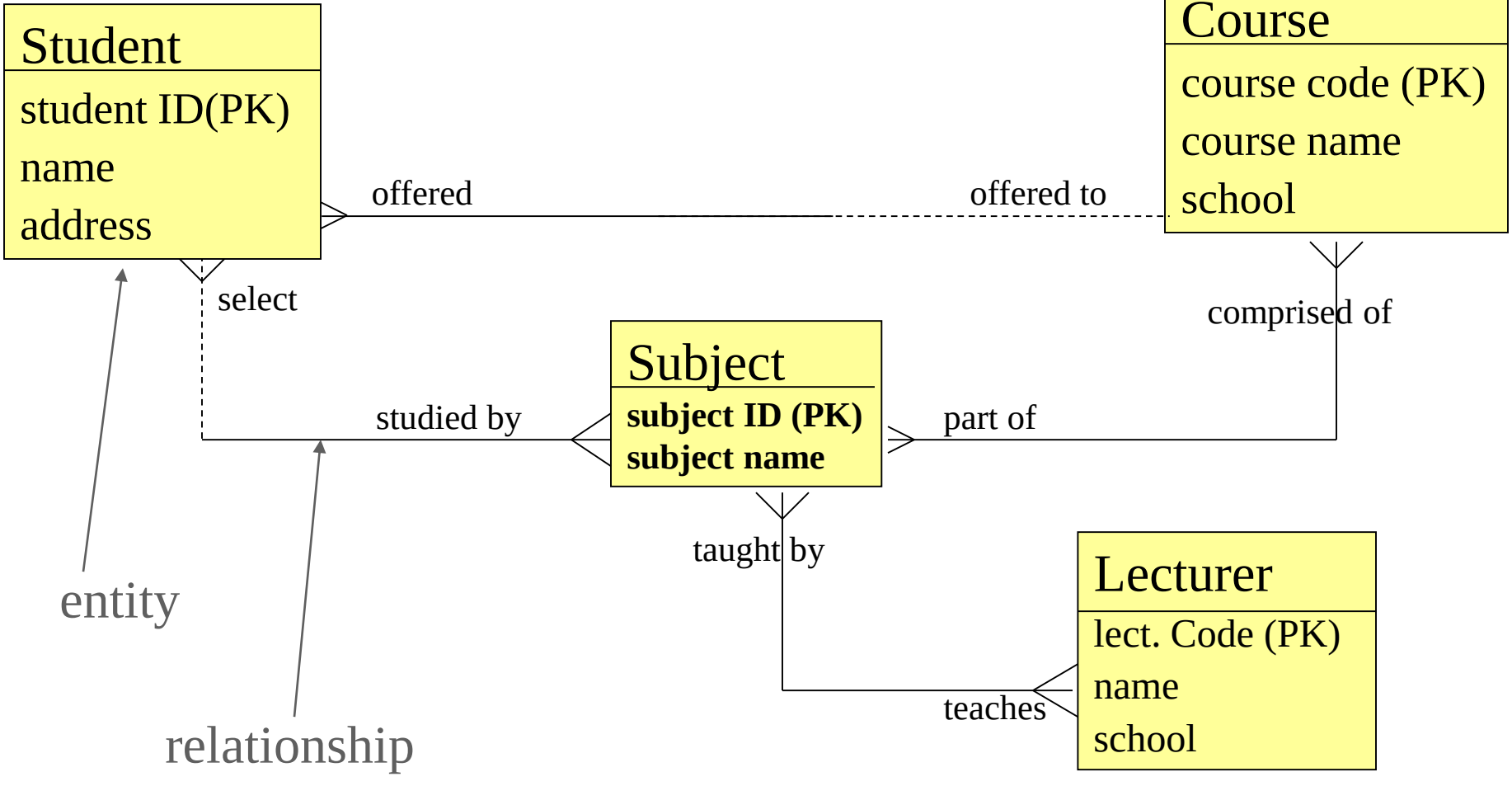
Want to go from this…



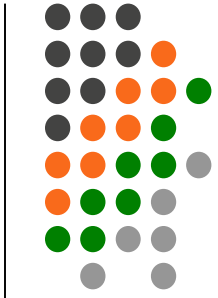
Data specification / Problem statement /

- *Students are offered a course which is comprised of a number of subjects, each taught by a lecturer. Students can select the subjects they want to do.*

To this – database schema with tables, attributes, relationships…



After completing this unit, you should know . .. .

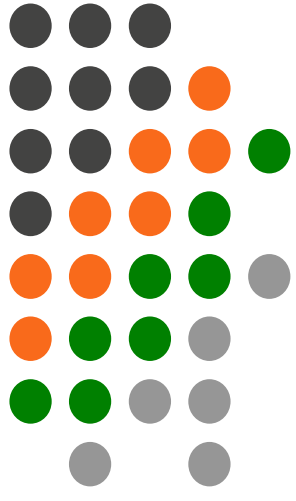


- What the ANSI/SPARC architecture for Database Management Systems is, also….

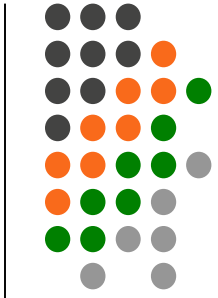
- How to model the data requirements of an application using an ERD (Entity-Relationship Diagram)

ANSI/SPARC Architecture

3 levels of a DBMS

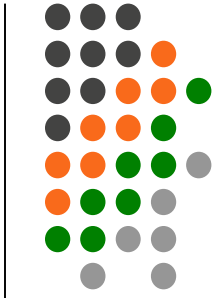


3 levels of a DBMS - ANSI/SPARC Architecture



- ANSI: *American National Standards Institute*
- *SPARC: Standards Planning And Requirements Committee*
- The ANSI/SPARC architecture is an abstract design standard for a Database Management Systems, first proposed in 1975, and **still** adhered to by most modern commercial DBMS today.

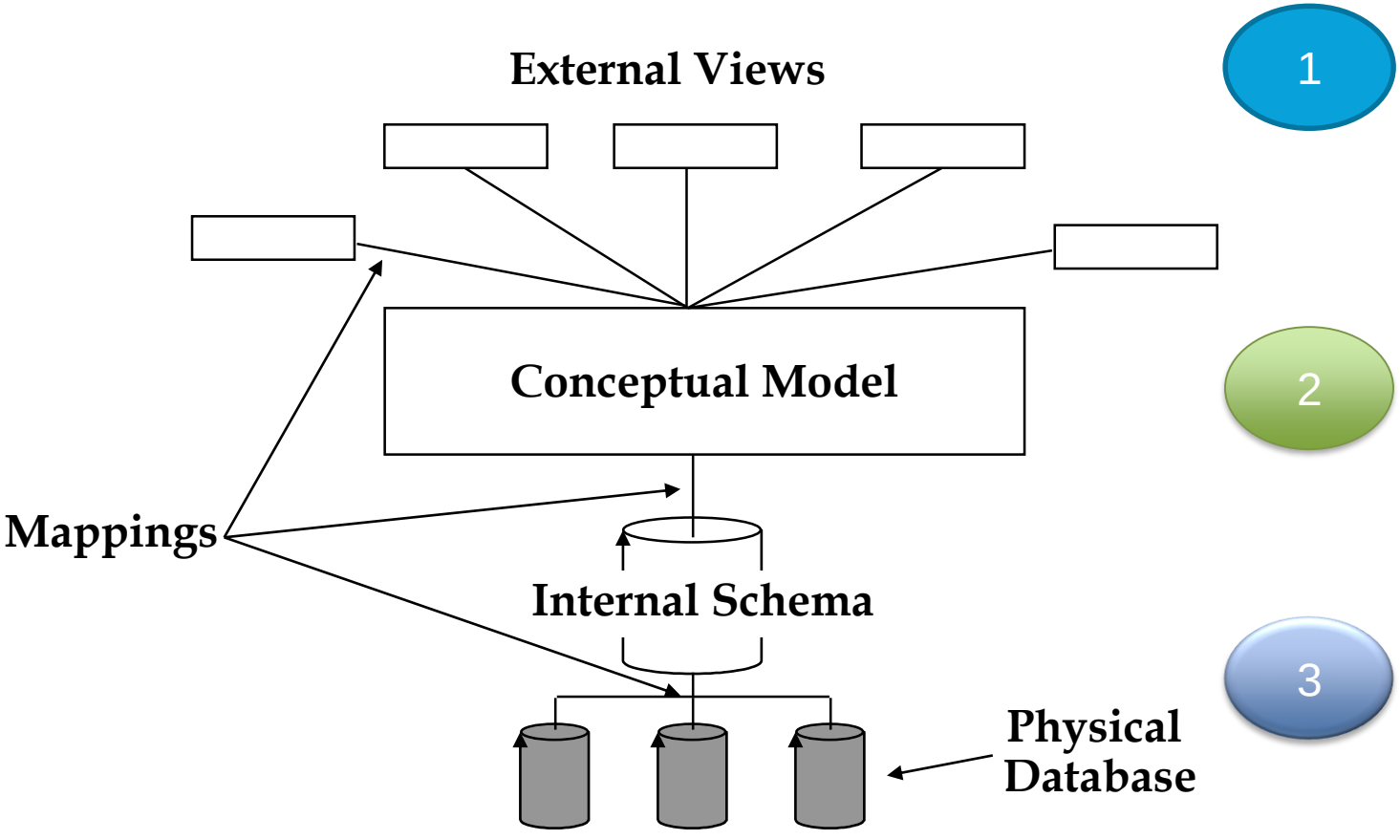
3 levels of a DBMS - ANSI/SPARC Architecture



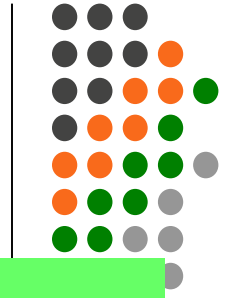
Objective: To separate the users' view(s) of the database from the way that it is physically represented.

- Each user can have a different view of the data
  - they see just what they need to see, but not all the data.
- Hide the physical storage details from users
  - Data could be moved to a different disc and users would not need to know.

3 levels of a DBMS - ANSI/SPARC Architecture



ANSI/SPARC Architecture



1. External Views:

A users view of the **section** of the database that is relevent to that user. For example:

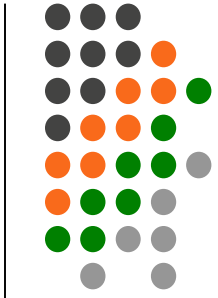
- **Lecturer view** would include student name, email, and results for their module only;
- **administration view** could include address, phone number, payment details, but not module results.

2. Conceptual Model:

A model of the entire database which bridges the layer between the internal schema and the external views.

Concerned with entities, relationships, constraints, security (who has access to what data), integrity (student can only do a course that is listed in the course table)

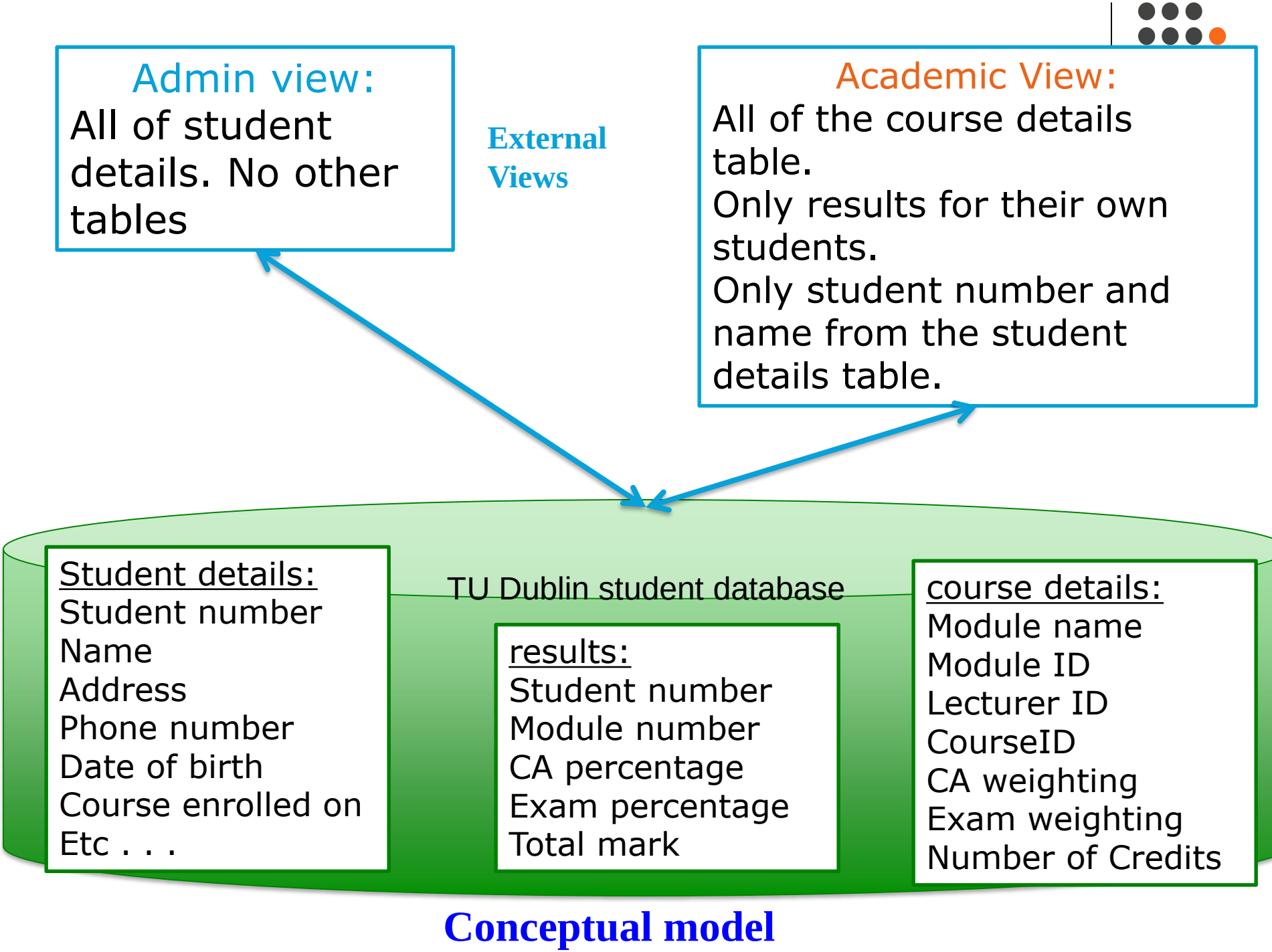
3 levels of a DBMS - ANSI/SPARC Architecture



3. Internal Schema:

The way data is stored on a physical database.

Concerned with disk storage, indexes, record description and placement, compression & encryption



Conceptual model

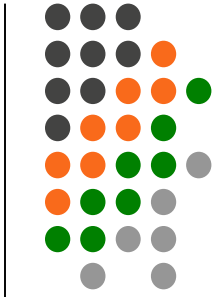
Internal schema . .

Student details:
Student number – data type: character (string), length 9.
Name – data type: character , length 50.
Address – data type: character , length 250
Phone number - data type: character, length 20
Date of birth – data type: datetime
Course enrolled on – data type: character, length 5
Etc . . .

results:
Student number – data type: character (string), length 9, linked to student number in student details
Module number -data type: character , length 5
CA percentage - data type: decimal. Valid range 0 to 100.
Exam percentage - data type: decimal. Valid range 0 to 100.
Total mark - data type: decimal. Valid range 0 to 100.

. .and similarly for the course table

Relevance of the ANSI/SPARC architecture to how we design a database?



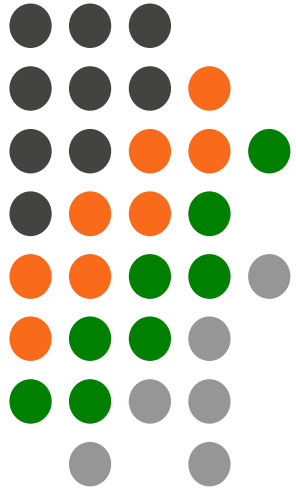
The Steps to design a database following the ANSI/SPARC architecture:

1. The data needs of different user groups will define what data needs to be included in the database.
2. Data needs are combined into an single **Conceptual Model** of all the data to be stored in the database.
3. This conceptual model is converted to a definition of database tables.

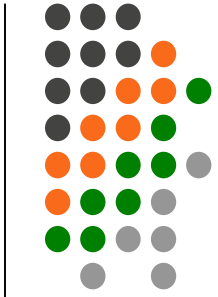
Marie Brennan

Conceptual Models

ERD – Entity Relationship Model



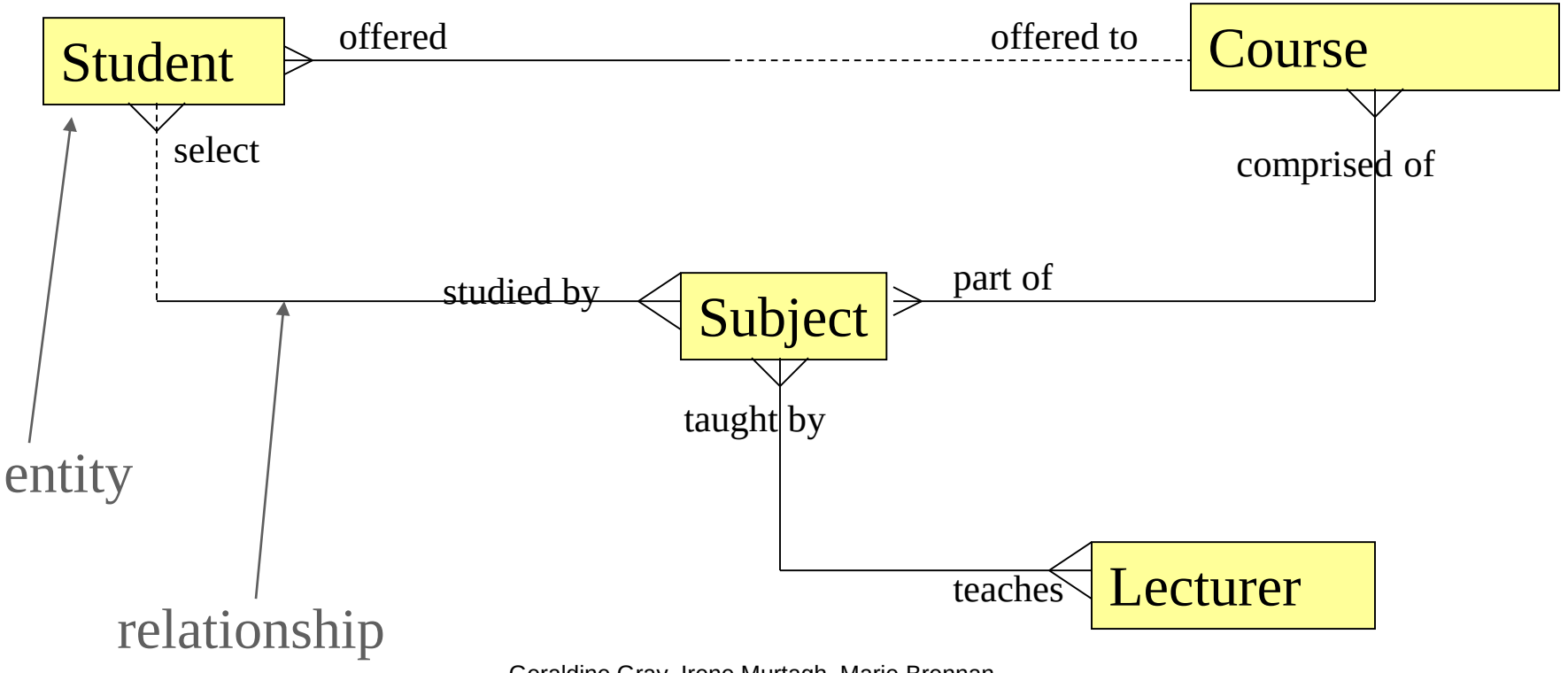
Entity Relationship Diagrams



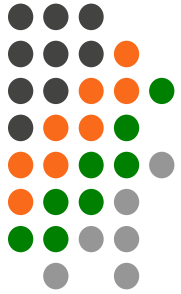
- The initial model for a database is a conceptual model to determine what **entities** a system needs to store data about.
- The most common conceptual model used is an **Entity Relationship Diagram**.
- It models Entities, and the relationships between entities.

Entity Relationship Diagrams (ERD)

Students are offered a course which is comprised of a number of subjects, each taught by a lecturer. Students can select the subjects they want to do.



So what is an Entity?



- General Definition: An Entity is something that exists as a particular and discrete unit.
- Definition with respect to ERD's: An **entity** is something in the real world about which we want to store data.

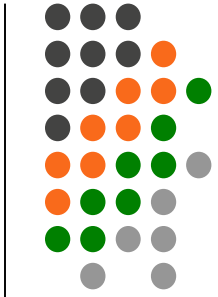
- An entity may be:
  - a **physical object** such as a **house** or a **car**,
  - an **event** such as a **house sale** or a **car service**,
  - A **business transaction** such as an **order**, a **bill**, an **invoice**

Entities

- In a description of computer applications, entities are the nouns in the description about which you want to store information. They generally fall into four categories:

1. **People**: customer, supplier, employee
2. **Products**: car, book, food item, clothing, etc.
3. **Services**: holiday booking, eating out, hair cut
4. **Recording transactions**: deposit, withdrawal, order, invoice, bill, receipt

From Entities to entity types

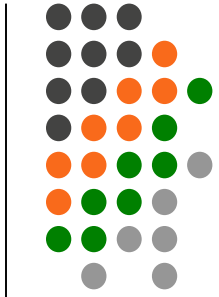


- ERD's model entity types, rather than entities.

- An **entity type** defines a set of entities that have the same attributes or properties.
  - For example, students *John Murphy, Mary O'Reilly, and Paul Ryan* are all entities. The entity type would be *STUDENT*.
  - The name given to an entity type is always singular (not plural) – e.g. student not students.

Entity types can be identified from **nouns** used to describe the system.

Exercise



Identify three entities in the following description:

A customer orders a product. Each product is manufactured from a number of parts.

1. Customer

2. Product

3. Part

Exercise

Identify two entities in the following description:

When a student registers at TU Dublin, their name and address are entered into the system, and they are allocated to a course.

1. Student	2. Course

Exercise

How many entities are in the following description?

To order a book from the website, add it to the shopping cart, and then enter delivery and payment details on the checkout page.

Remember: An **entity** is something in the real world about which we want to store data.

Answer: 2

Relationships

- Each entity is mapped to it's own table in the database.
- A **relationship** defines the associations between entities.

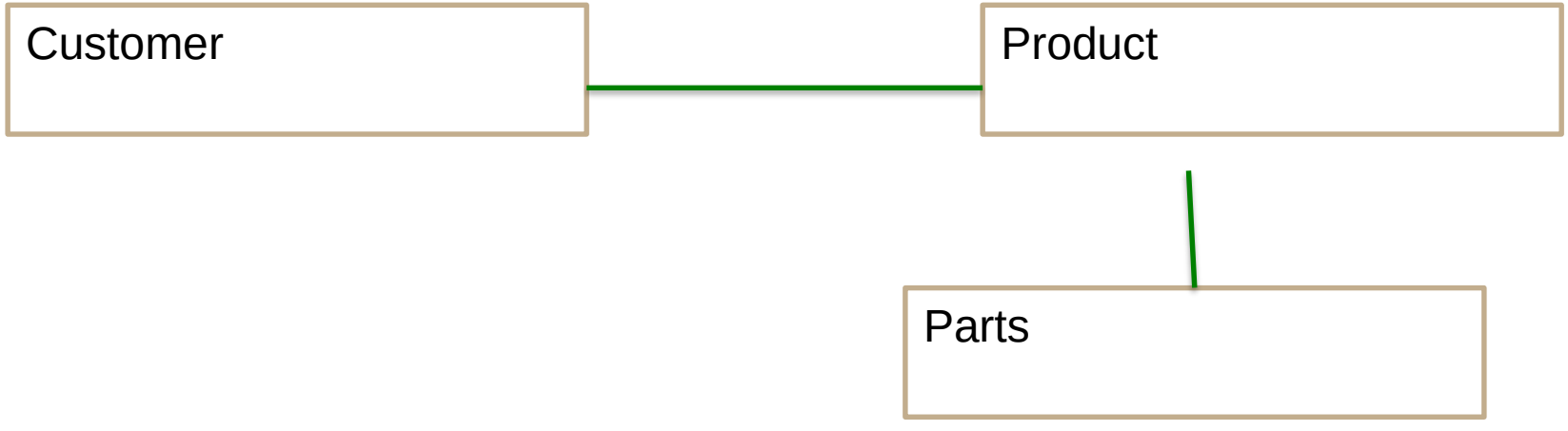
➢ Relationships indicate which entities (tables in the database) need to be linked using foreign keys.

Relationships can be identified from verbs used to describe the system.

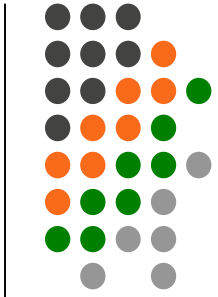
Exercise

What are the relationships in the following description? Link the entities that are related.

A customer orders a product. Each product is manufactured from a number of parts.



Drawing an ERD



- An Entity Type is drawn as a Rectangle with name of entity type in singular inside.
- A relationship is drawn as a line linking entity types.
- The relationship can have one or two labels explaining the relationship between the entities.
  - Only include a second label if doing so helps to explain the diagram.

Example

Students are offered a course

- Nouns/entity types: student, course
- Verb: is offered

- ERD:

More on relationships

A relationship is characterised by three properties:

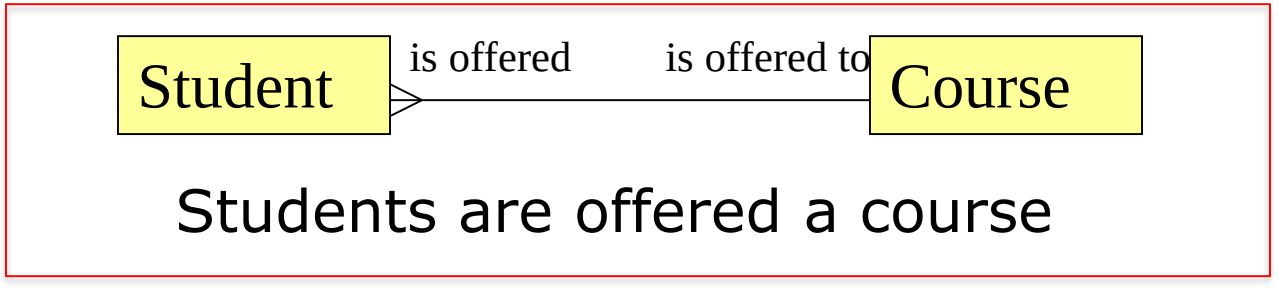
1. Degree: how many entity types are associated by the relationship
   - Unary (one), binary (two), ternary (three)

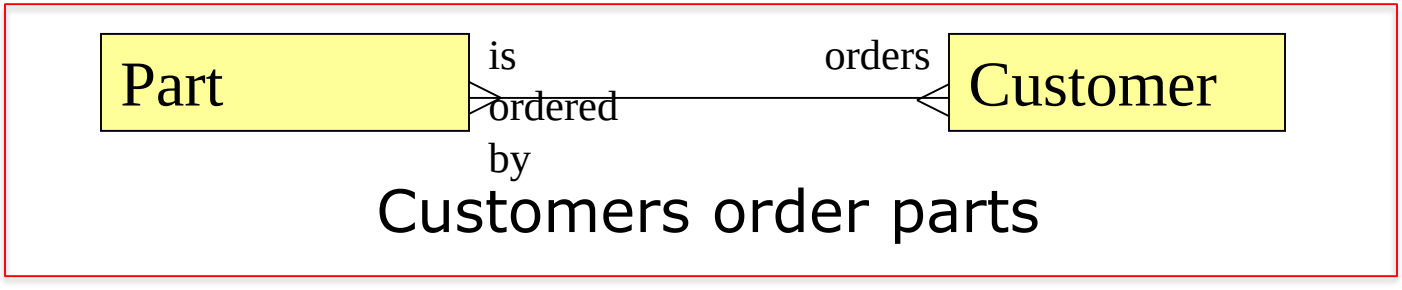
2. Cardinality: How many entities of entity type A can be associated with an entity of entity type B
   - An association can be:
     one to one - 1:1
     one to many - 1:m
     or many to many m:n

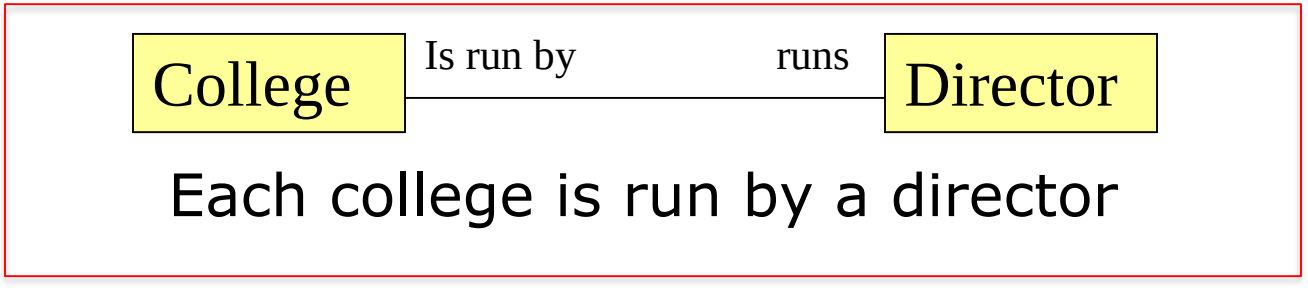
3. Participation
   - Mandatory or optional association

The **degree** of a relationship

1. A Binary relationship links two entity types

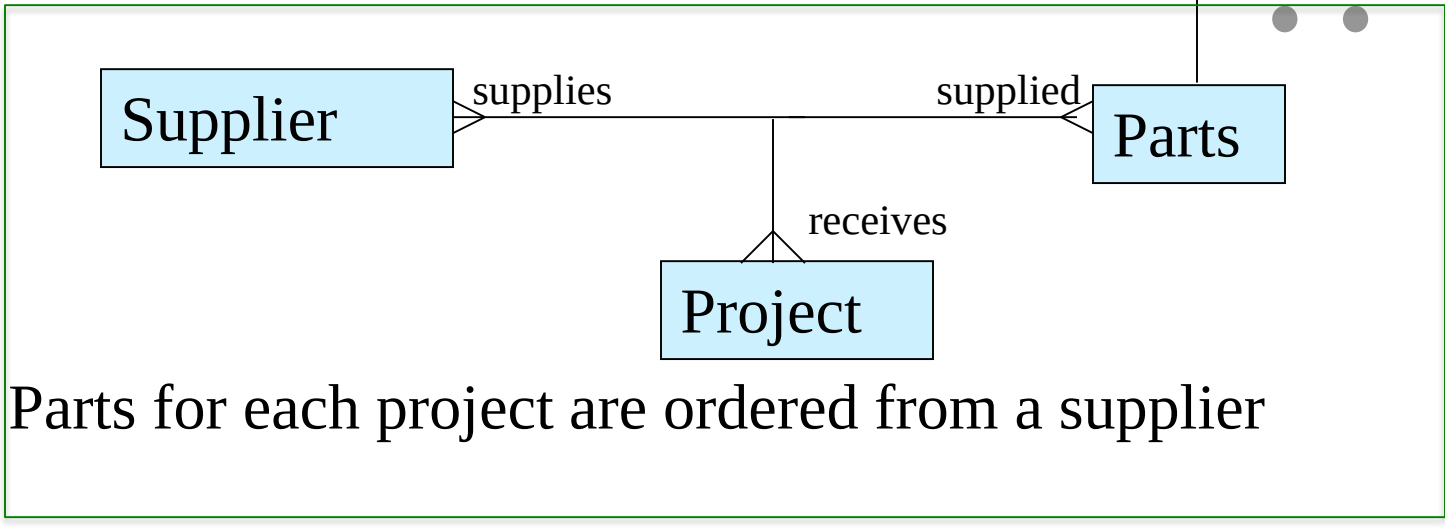




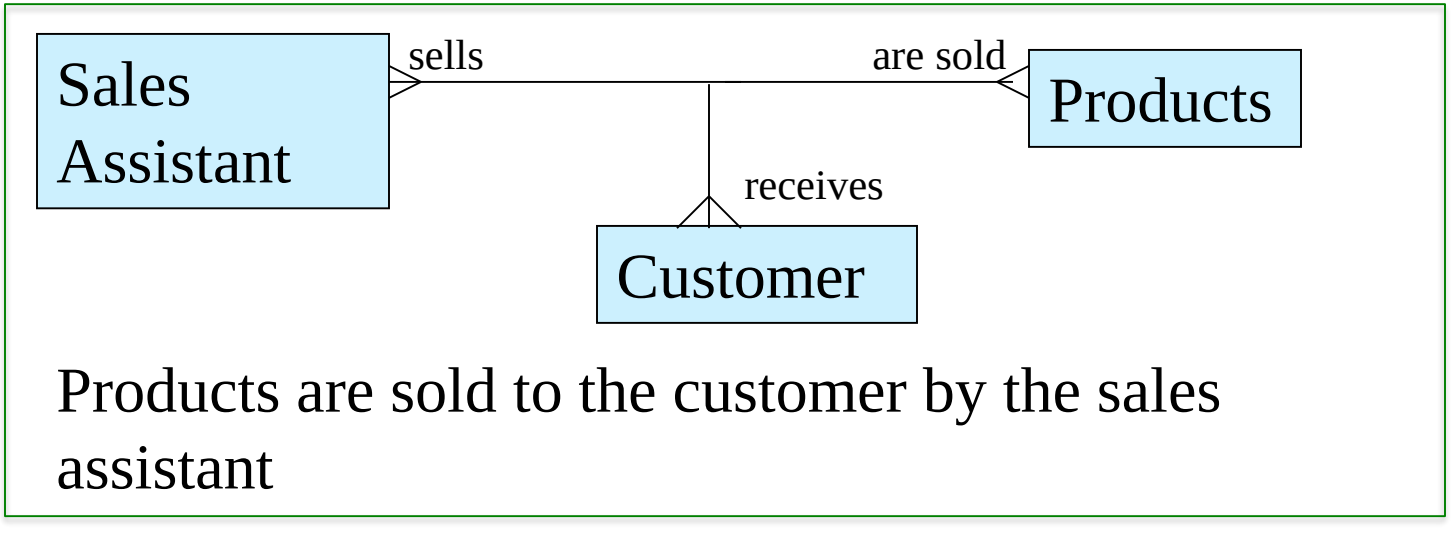


The Degree of a Relationship

2. A Ternary relationship links three entity types.

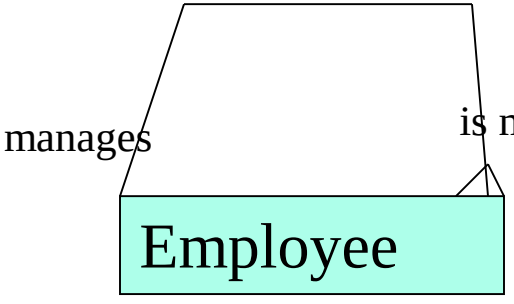


Relationship labels should make the diagram easier to understand.

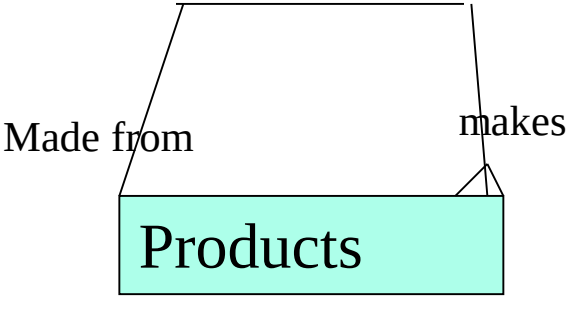


The degree of a relationship

3. A Unary relationship links an entity type to itself

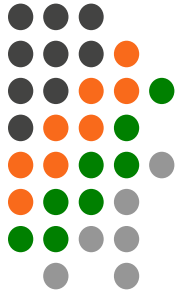


Managers manage employees



Some products are made from other products

Cardinality of a relationship



- How many entities of entity type A can be associated with an entity of entity type B?

Students are offered a course

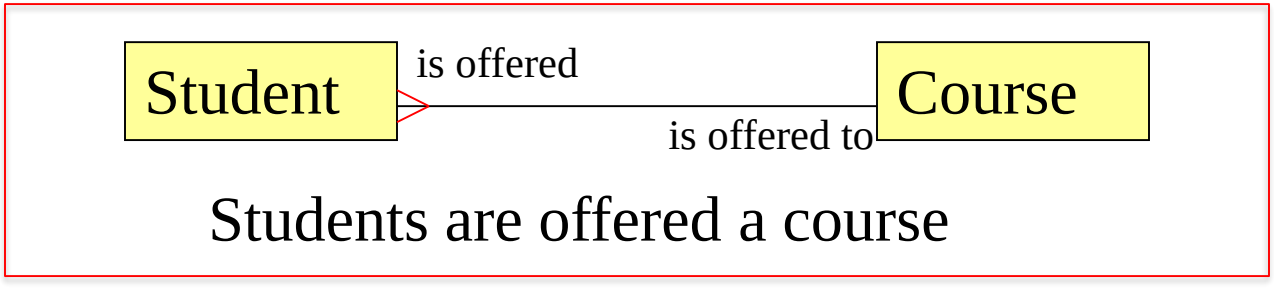
1. How many students can be offered the same course?
   - Is it 1 or more than 1?
   - Ans: more than one

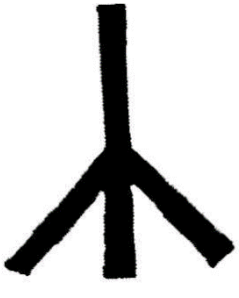
This is a 1:m relationship – read as 'one-to-many'

2. How many courses can a student be offered?
   - Is it one or more than 1?
   - Ans: at most one

Cardinality of a relationship

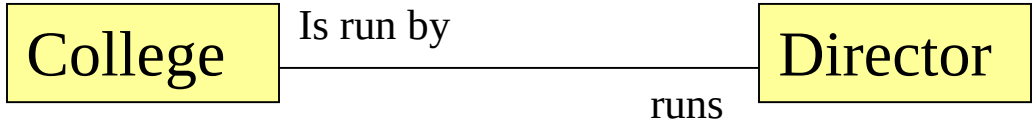
- A 1:m relationship is represented as a crows foot on the many side:





Cardinality of a relationship

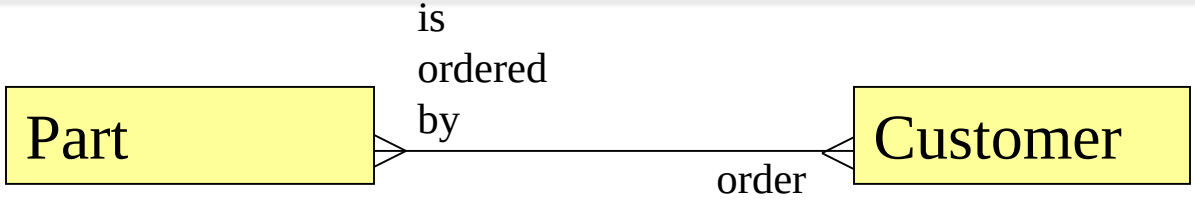
- There are two other cardinalities:
  - 1:1 – one to one



A college can only have one director

A director can run only one college

No crows feet needed



Customers can order more than one part

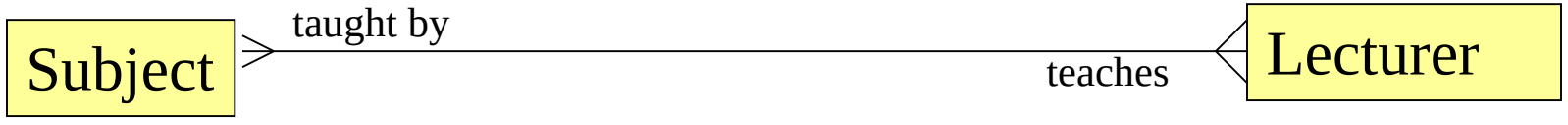
A part can be ordered by more than one customer

Represented as a crows foot on both end

  - M:n – many to many

Participation - Mandatory or Optional relationship

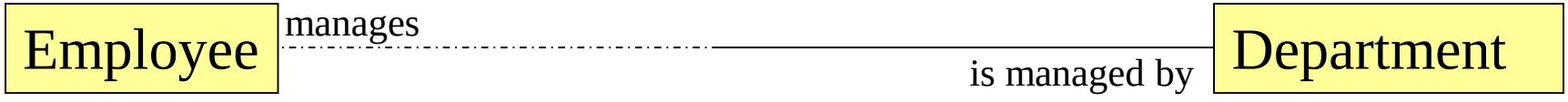
If two entity types are related, must each entity in one entity type be linked to at least one entity in the other entity type?



- Every subject must be taught by at least 1 lecturer, so each subject in the subject table must be linked to at least one lecturer
- Similarly, every lecturer must teach at least one subject, so every lecturer listed in the lecturer table must be linked to at least one subject
- Both sides of this relationship is mandatory (or total)

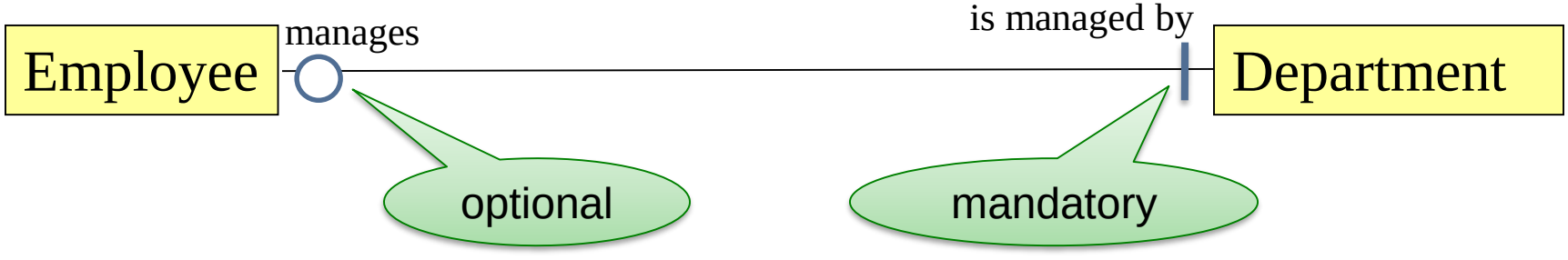
Participation

- Other examples . . .



- An employee may manage a department (optional or partial participation), but a department must be managed by an employee (mandatory participation)

Alternative Notation



Participation (Recap.)

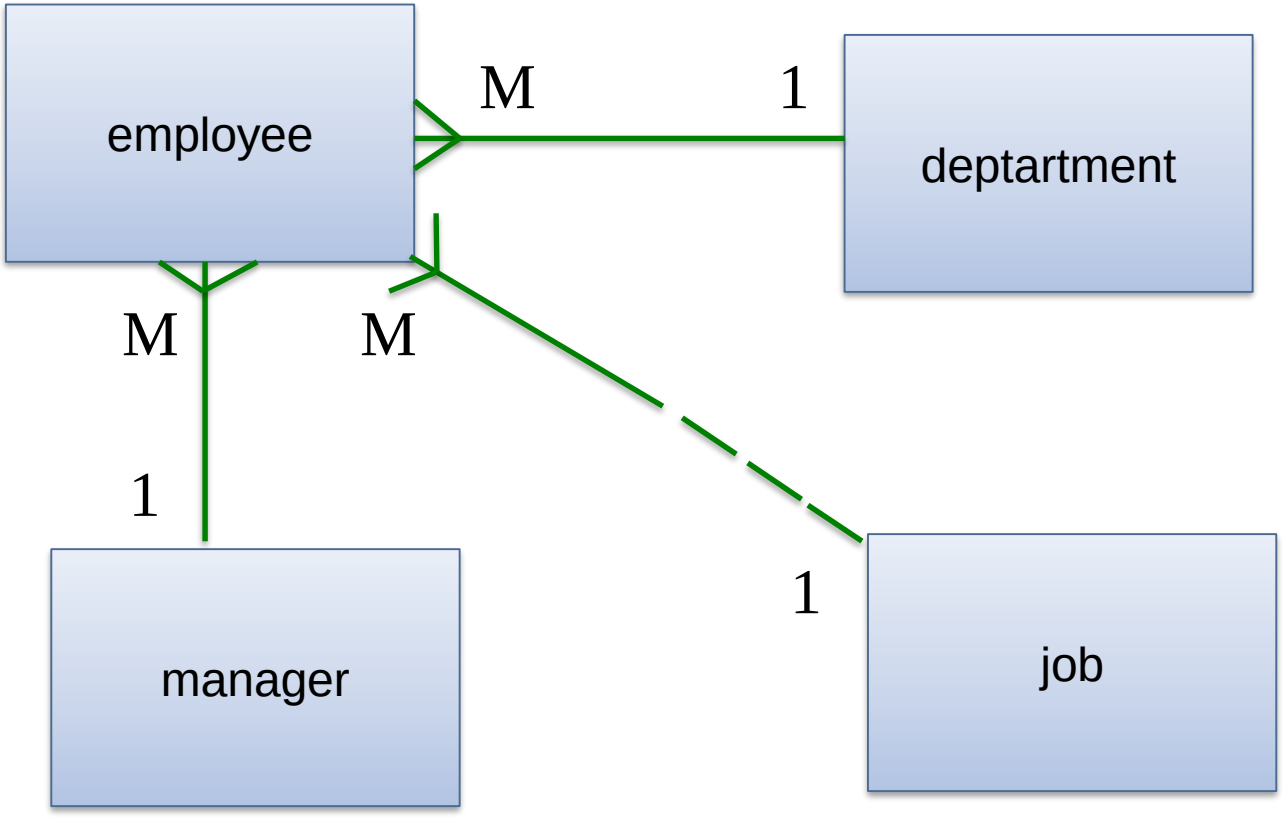
- Mandatory relationship

- Optional relationship

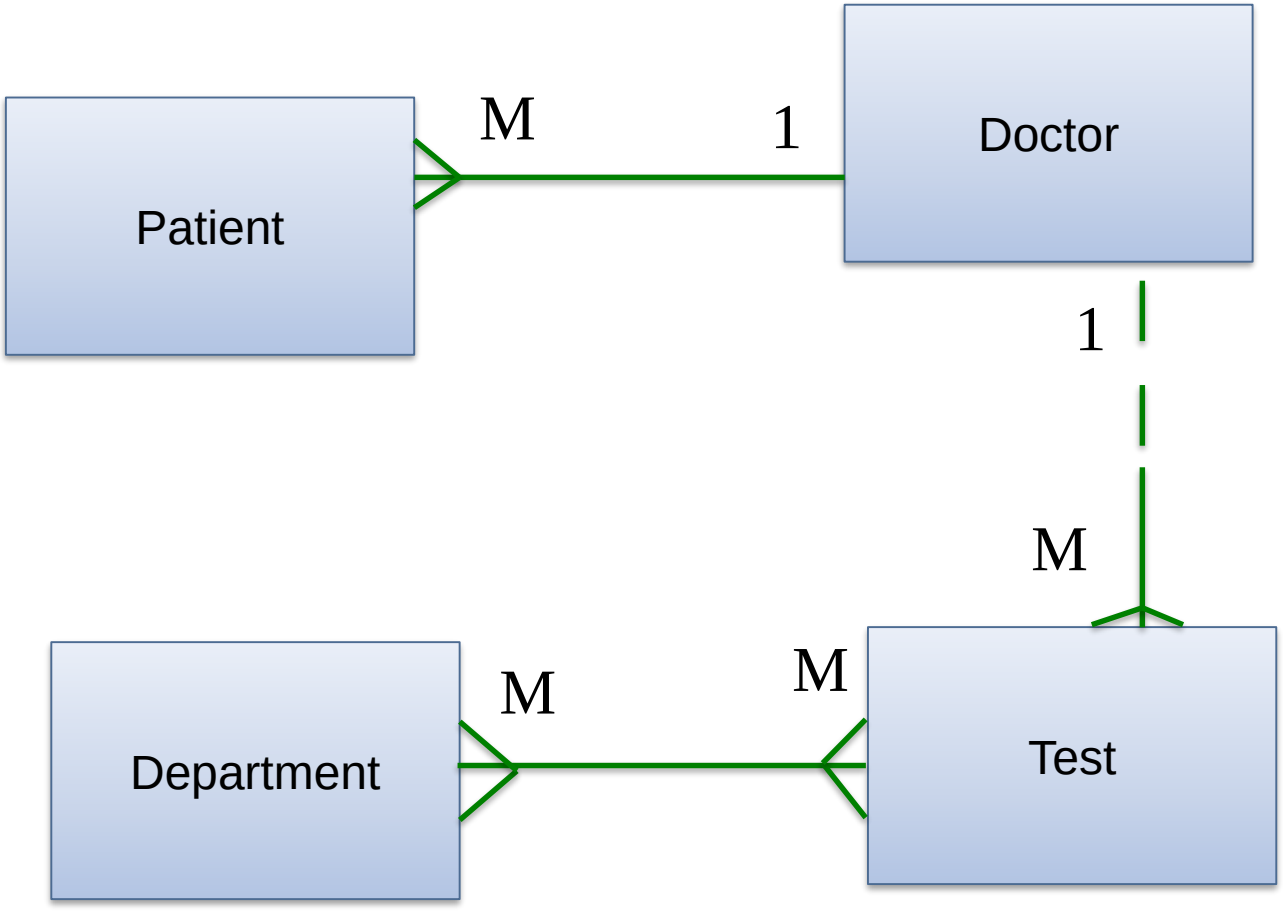
Exercises

- Represent each of the following as an entity relationship diagram:
  1. Every employee is assigned to a department. Each employee reports to a manager, and is allocated to exactly one job. A job can have one or more employees allocated to it, but may have none.
  2. Every patient is examined by one doctor. The doctor may order a number of tests for each patient. Each test is carried out by a department. Each department is responsible for a number of tests.

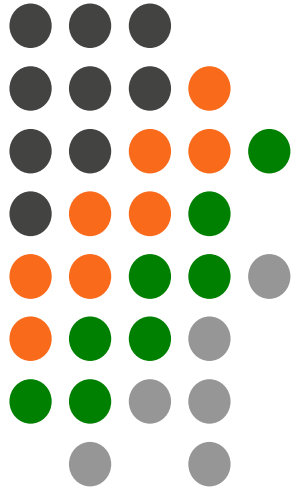
Solution to Exercise 1. . . .



Solution to Exercise 2. . . .



Adding attributes to entities



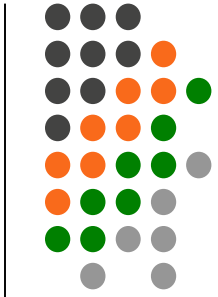
Re-cap

- To date we have covered:
  - Creating an entity relationship diagram
  - Relationship properties
    - Cardinality / Degree

      (1:1; 1:m; m:n , unary, binary, ternary)
    - Participation (mandatory, optional)

- Next stage: allocate attributes to each entity

Attributes



- An Entity-Relationship Diagram can also include attributes.
- Attributes are the information we want to store about each entity.
  - e.g. Attributes of the entity type **CUSTOMER** might be ***Customer name, Customer address, Telephone number, Contact name,*** ***etc***.

- Each attribute has a value
  - ***Customer name***: IBM
  - ***Customer address:*** Ballycoolin Ind. Estate, D 15
  - ***Telephone number***: 8871000
  - ***Contact name:*** Sadhbh Quinn

Attributes

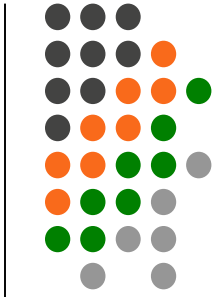
- Like entities, attributes can be identified as the NOUNS in a system description.
  - For most NOUNS, its easy to tell if its an attribute or an entity, but its not always obvious.

What are the attributes in the following?

When a student registers at TU Dublin, their name and address are entered into the system, and they are allocated to a course.

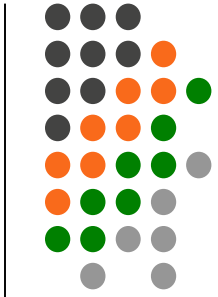
Never add your own attributes, only use the attributes identified by the user/customer in the system description (unless asked to do so in an exam question!)

Properties of attributes



- Attributes can have a number of properties:
  - Single-valued or multi-valued
  - Simple or composite
  - Derived
  - Null or not null
  - Key attribute

Attribute Characteristics

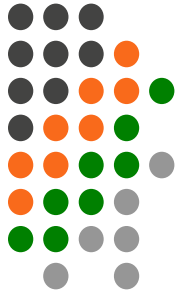


single valued & multi-valued attributes:

- Most attributes will only have one value for each entity, and so are single-valued attributes
  - e.g. a person can only have one DOB/age, one gross salary, one home address etc.

- Some attributes can have more than one value for each entity, and so are multi-valued,
  - e.g. qualifications (cert, degree, masters, PhD etc.)

Attribute Characteristics



Simple & Composite attributes:

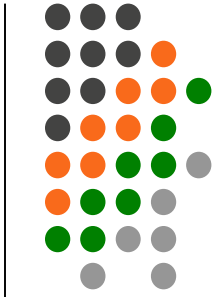
- A composite attribute is one which can be divided into simple attributes - each which has its own meaning
  - e.g. address can be split into:

    street name, town, city, country, area code

    or address line 1, address line 2, address line 3, address line 4, area code
- A simple attribute can not be further sub-divided
  - e.g. surname

Attribute Characteristics:

Derived Attributes



- The value of some attributes can be derived from other attributes and so do not need to be stored in the database
  - e.g. a persons age can be derived from their date of birth and so you would not need to store both age and date of birth in the table

Note: You may decide to store an attribute that could be derived from other values so that select queries run more efficiently.

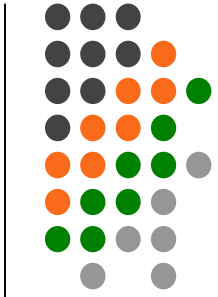
Attribute Characteristics

null value

- Not all attributes will have a value for each entity,
  - e.g. employee entity type could have an attribute car registration number, but not all employees would have a car.
  - This attribute is allowed to have a value of **NULL.**

- Note: if a large number of entities have null for a particular attribute, that attribute should be removed to another table. e.g. if 10% of employees have offices, then rather than having '**office_number**' as an attribute of '**employee**', set up a entity **emp_offices** with attributes **employee_number, office_number**.

Key attributes

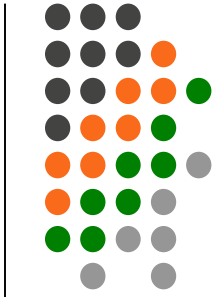


- An entity type should have some attribute, or set of attributes, which uniquely identifies each entity in the set e.g.
  - Student (student ID, student name, student address, phone number, course)
  - Customer (customer name, customer address, telephone number, contact) cust_id
  - Employee (employee name, employee address, date of birth, phone number, salary) emp_id

- This key attribute is called the **PRIMARY KEY**.

- A primary key, which is made up of more than one attribute is called a composite primary key e.g. (song, album)

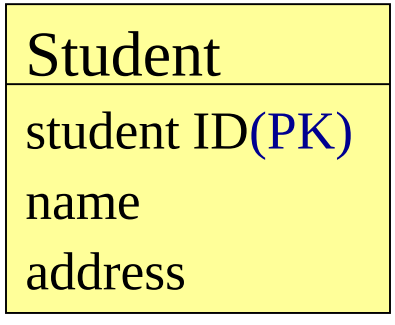
Primary Key



- Every entity type must have a primary key, which is an attribute or set of attributes that uniquely identifies each entity. This key must:
  - not change its value during the life of the entity
  - always have a value for each entity
  - replace large composite keys with a **surrogate key** - i.e. an attribute added to the entity type for the sole purpose of being the primary key.

Adding attributes to an ERD

- Attributes are added to an entity in an ERD as shown below. (PK) after an attribute indicates it's the primary key



Exercise

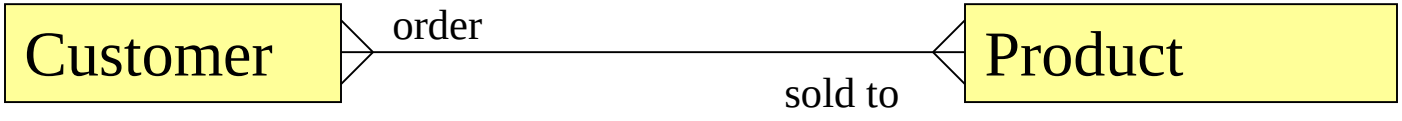
For each of the following attributes, state whether they are:

1. Single or multi-valued (S/M)
2. Simple or composite (S/C)
3. Would it make sense for this attribute to be NULL? (Y/N)
4. Could the attribute be used as a primary key? (Y/N)

- Product ID (in a product table). Ans:_____
- Product description (in a product table) Ans:_____
- Full name (in a customer table) Ans:_____
- Contact names (in a customer table) Ans:_____
- Address (in a customer table) Ans:_____
- Web site address (in a customer table) Ans:_____

Relationship Attributes

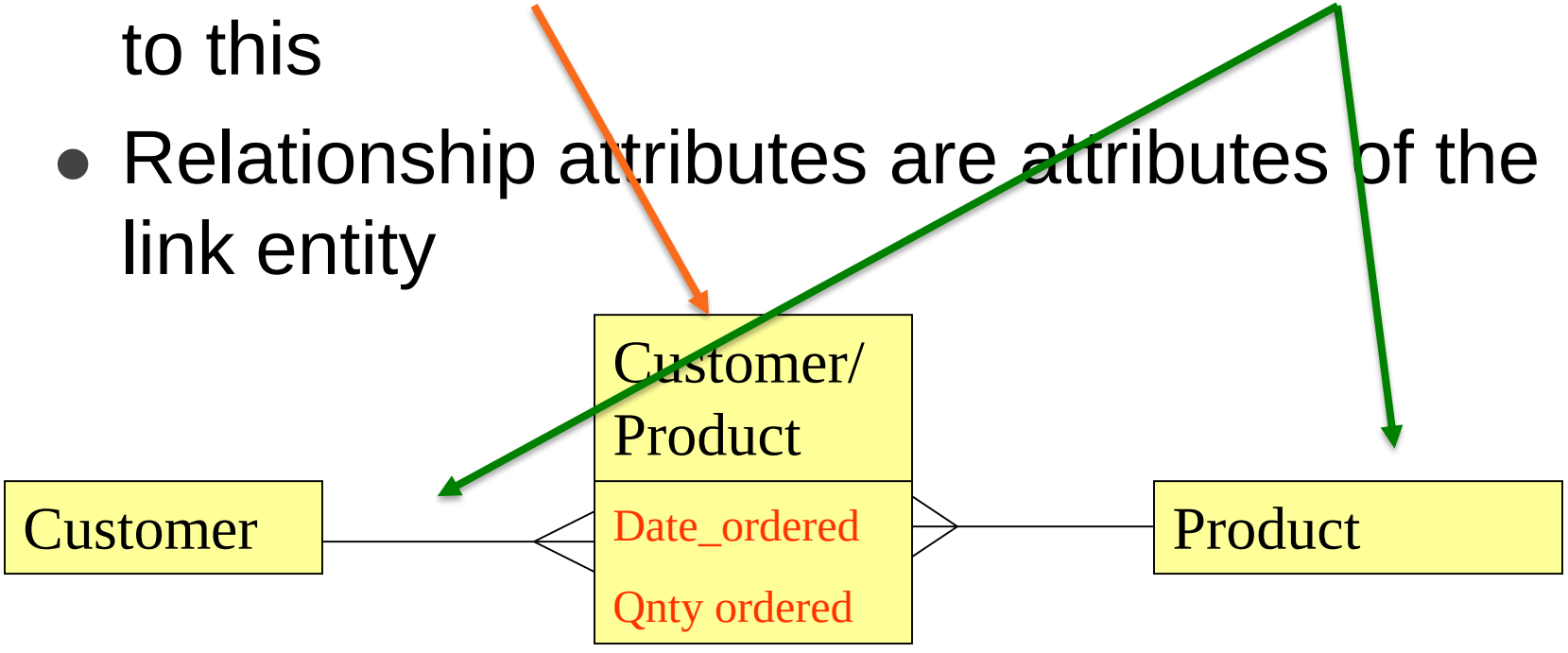
- A relationship can also have attributes associated with it:



- what happens details about the order such as:
  - date ordered, quantity ordered?
- This generally applies to m:n relationships only.

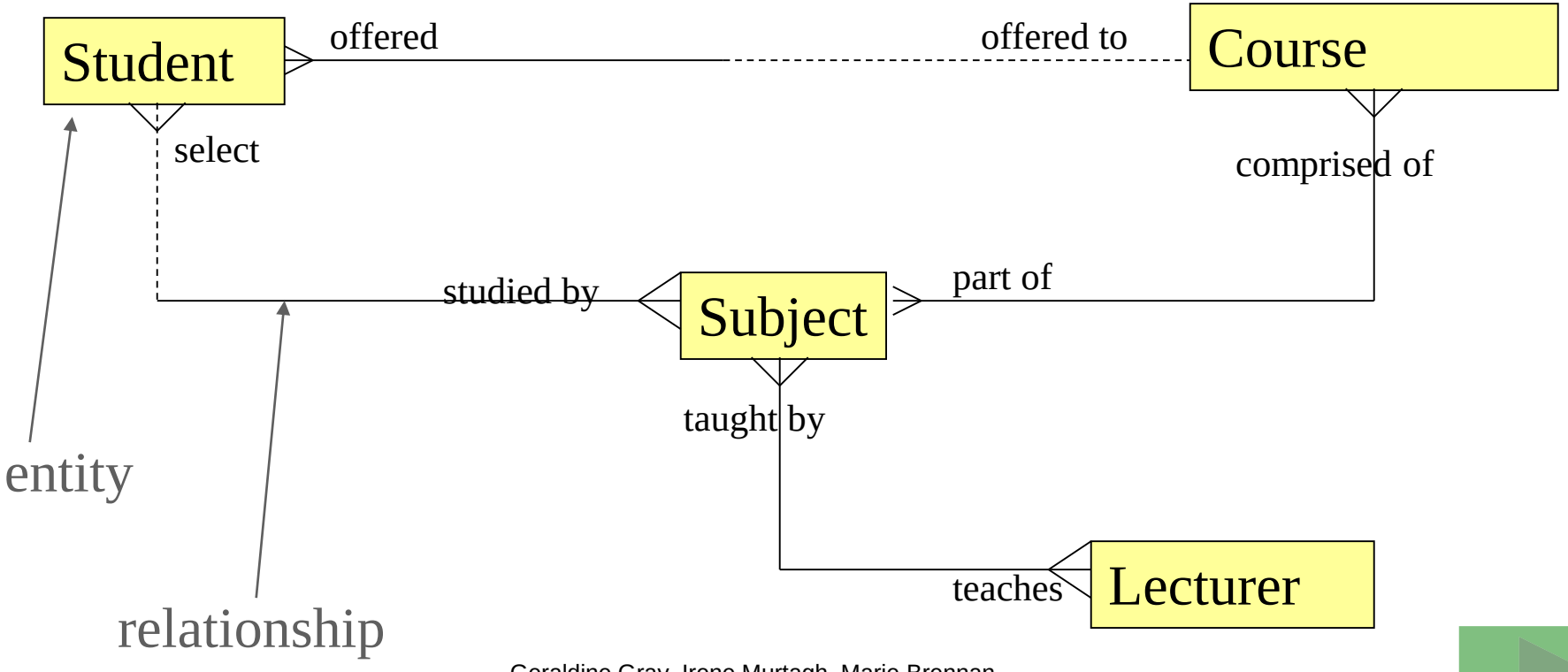
Relationship Attributes (cont)

- M:n relationships are changed as follows:
- Create a LINK entity and have two 1:M links to this
- Relationship attributes are attributes of the link entity

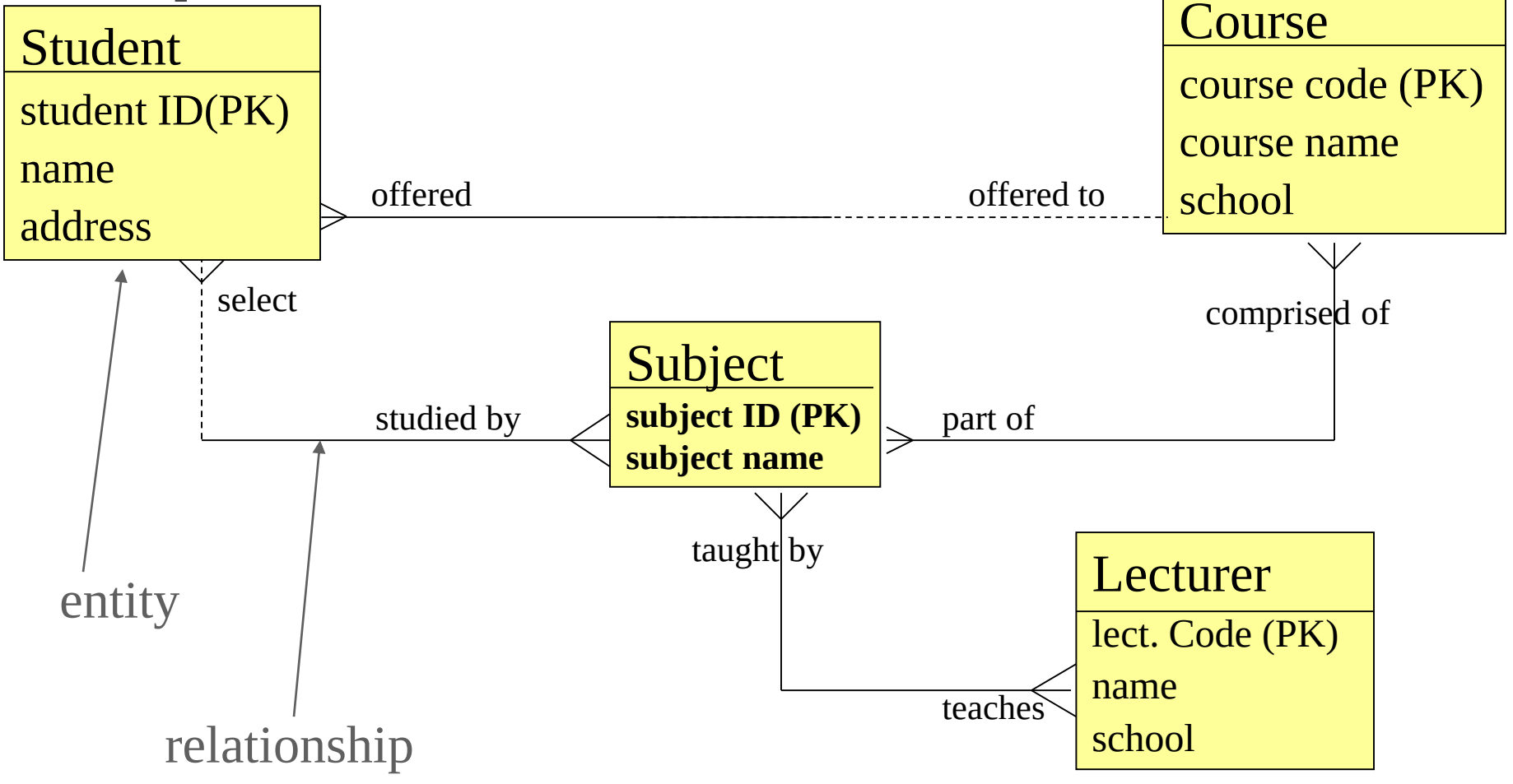


Step 1 - Create ERD

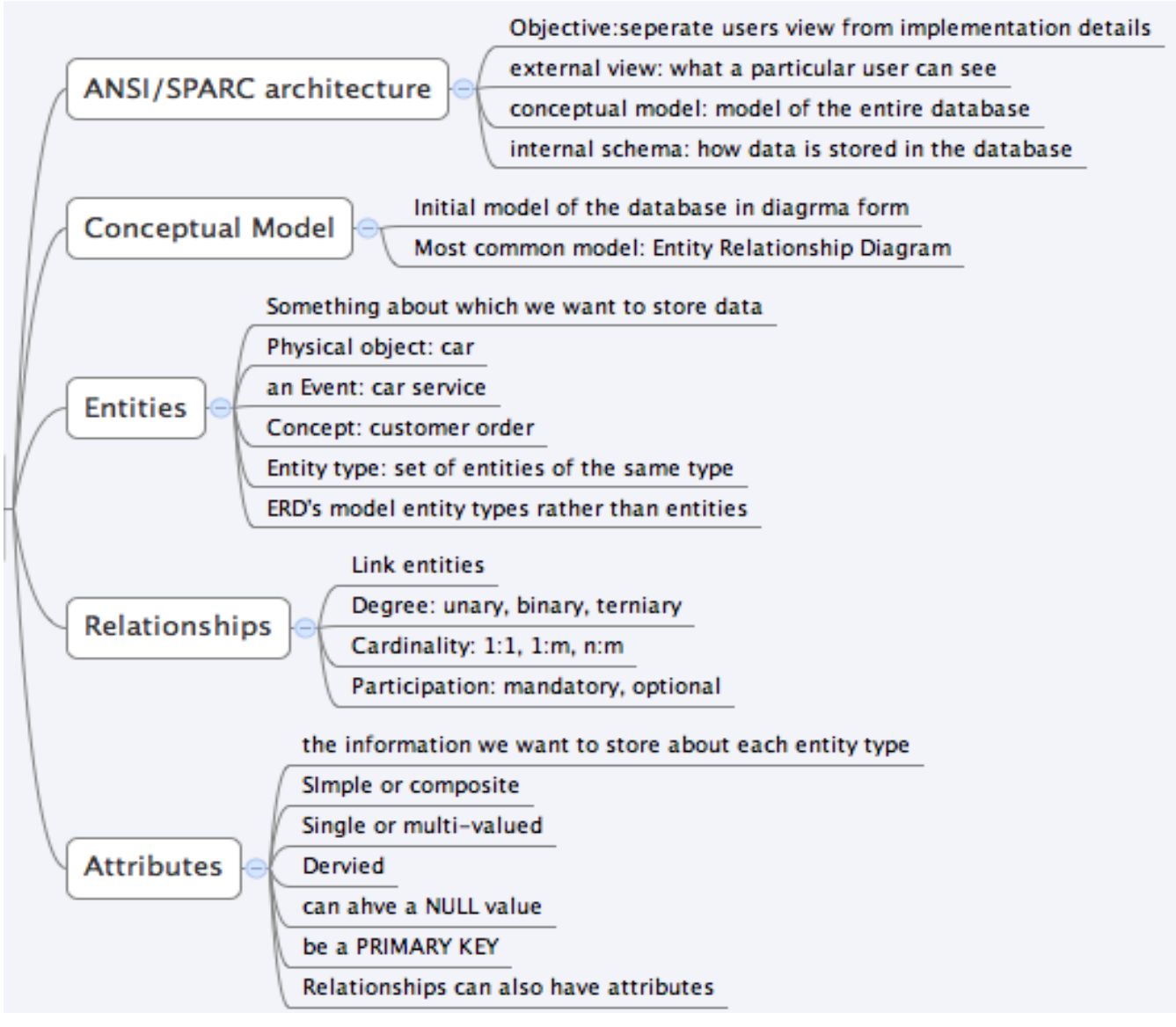
Students are offered a course which is comprised of a number of subjects, each taught by a lecturer. Students can select the subjects they want to do.



Step 2 - Add attributes to ERD



Summray


ANSI/SPARC architecture
Objective:seperate users view from implementation details
external view: what a particular user can see
conceptual model: model of the entire database
internal schema: how data is stored in the database
Conceptual Model
Initial model of the database in diagrma form
Most common model: Entity Relationship Diagram
Entities
Something about which we want to store data
Physical object: car
an Event: car service
Concept: customer order
Entity type: set of entities of the same type
ERD's model entity types rather than entities
Relationships
Link entities
Degree: unary, binary, terniary
Cardinality: 1:1, 1:m, n:m
Participation: mandatory, optional
Attributes
the information we want to store about each entity type
SImple or composite
Single or multi-valued
Dervied
can ahve a NULL value
be a PRIMARY KEY
Relationships can also have attributes