



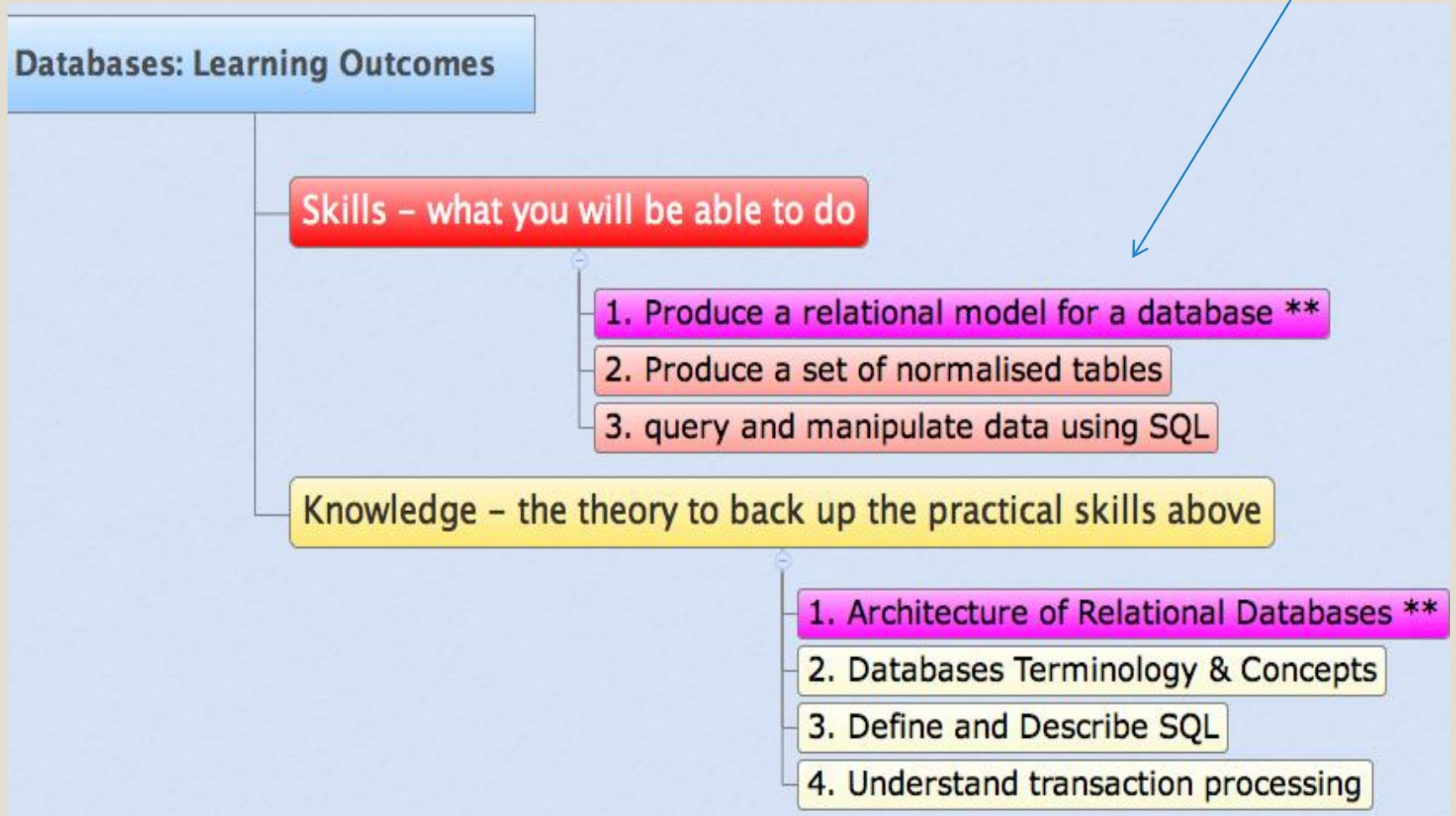
# DATABASE FUNDAMENTALS

**Lecture 7 (Data Modelling Cont)**

**Lecturer : Marie Brennan**

1

# Learning outcomes addressed today:



# Recap

## ANSI/SPARC architecture

Objective: separate 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 diagram 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, ternary

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

Derived

can have a NULL value

be a PRIMARY KEY

Relationships can also have attributes

# Recap on last week

- What is ANSI SPARC Architecture?

-Architecture which allows us to view a database system as a series of levels , External, conceptual and Internal level.

- What is an ERD?

Diagram used to model entities you want to store data about, including the relationships and attributes

- What is an entity/ entity type?

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

- What is a relationship/relationship type?

A relationship type defines the associations between entities.

Relationship Degree, Cardinality, Participation

# This Week .....

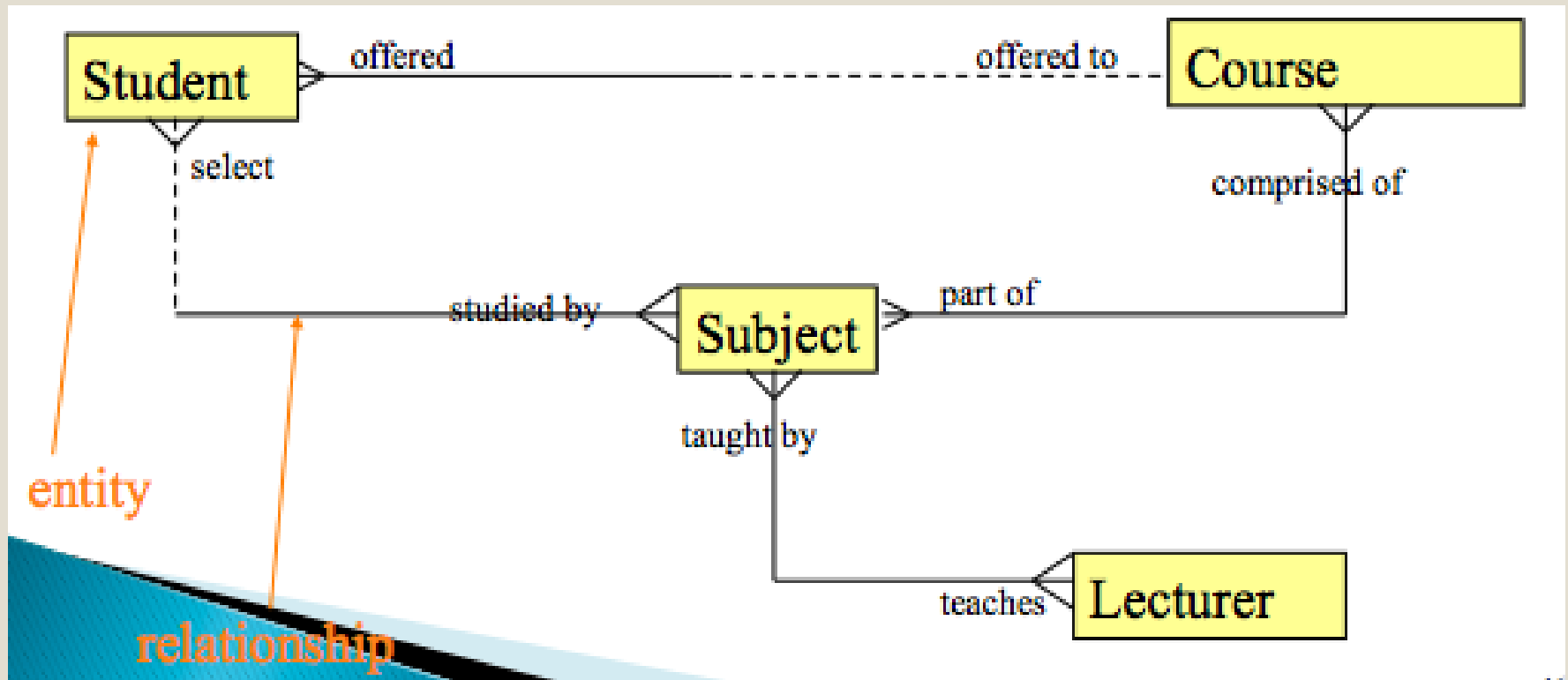
- Entities and Relationships continued
- Attributes
- The Relational Model

# RECAP

- Last week we covered:
  - Creating an entity relationship diagram
  - Relationship properties
    - Cardinality / Degree
      - (1:1; 1:m; m:n , unary, binary, ternary, multiple, exclusive)
    - Participation (mandatory, optional)
- Allocate attributes to each entity
- Attributes have characteristics (single multi-valued etc....)

# ERDs

- **Entities**: nouns in the text identifying what tables are needed in the database
- **Relationships**: verbs in the text identifying which tables will need to be joined using foreign keys.



# Entities

- Entities are the nouns in the text 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





# Relationships - verbs

- On each relationship, you need to decide: -----
  - Is the participation mandatory or optional
  - Is the cardinality 1:1, 1:m or m:n

The degree is the number of entities the relationship joins together, which is usually unary or binary



- Attributes: information you want to record about each entity. Attributes can be:
  - Simple or composite
  - Single values or multi-valued
  - Derived
  - The primary key
  - Allowed be NULL

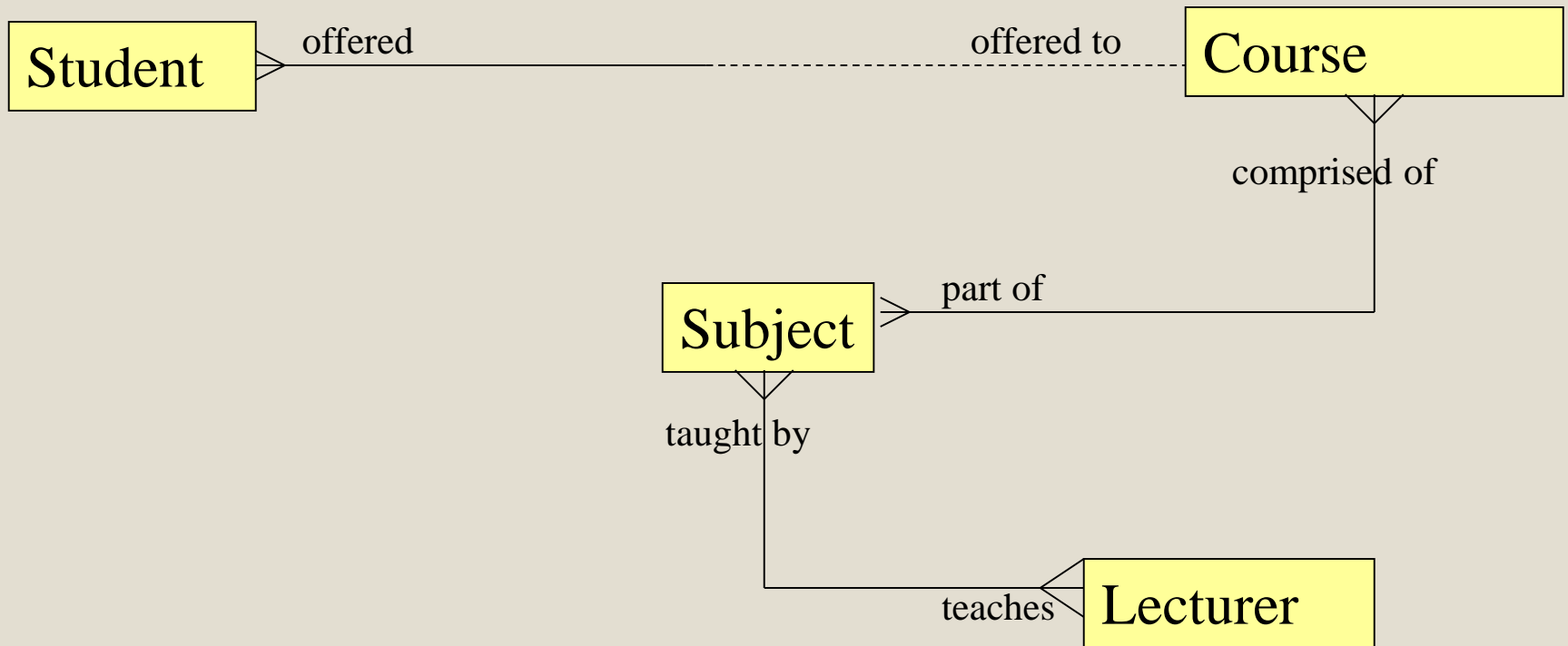
# Attributes

- Attributes are the information you want to record about each entity:
  - People, e.g. employee (name, address, phone number, dateOfBirth)
  - Products: e.g. spareParts( ID, description, quantityInStock, Price)
  - Service: e.g. car service(ID, description, price)
  - Transactions: e.g. quote(ID, date, time, cost, etc)

# RECAP

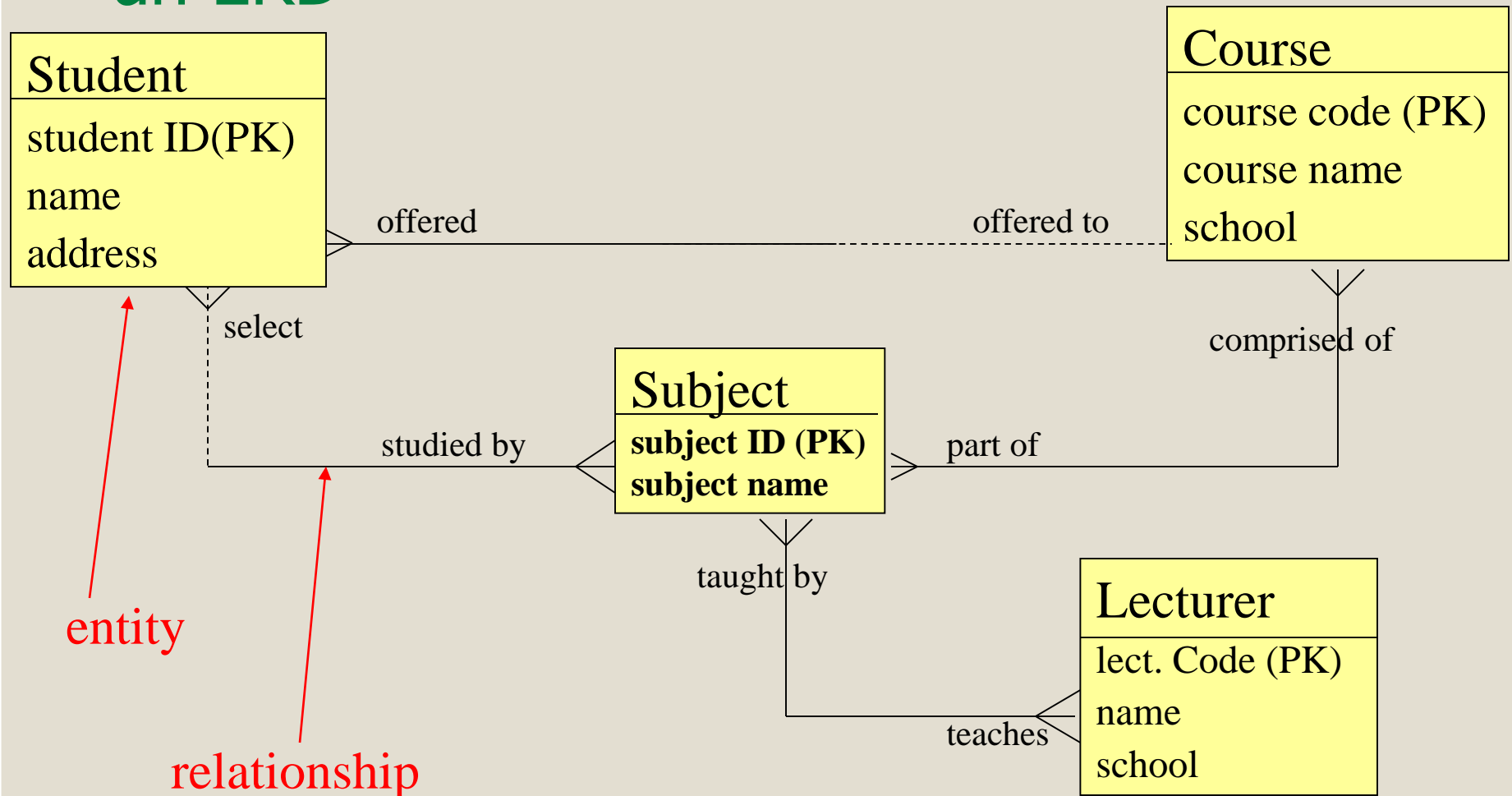
## We now know how to create an ERD

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



# RECAP

We now know how to add attributes to an ERD



Example 1: Each department employs one or more employees and each employee works for one department. Each of the employees may or may not have one or more dependants, and each dependant belongs to each employee. Each employee may or may not have an employment history.

- **Step 1**: List all the entities (nouns) NOT including the company name.
  - Department, employee, dependants, employment history
- Are there any duplicate entities?
  - Is employee different from employment history? (both store information about the employee)
  - Is employee and dependant the same entity? (both are people)

department

employee

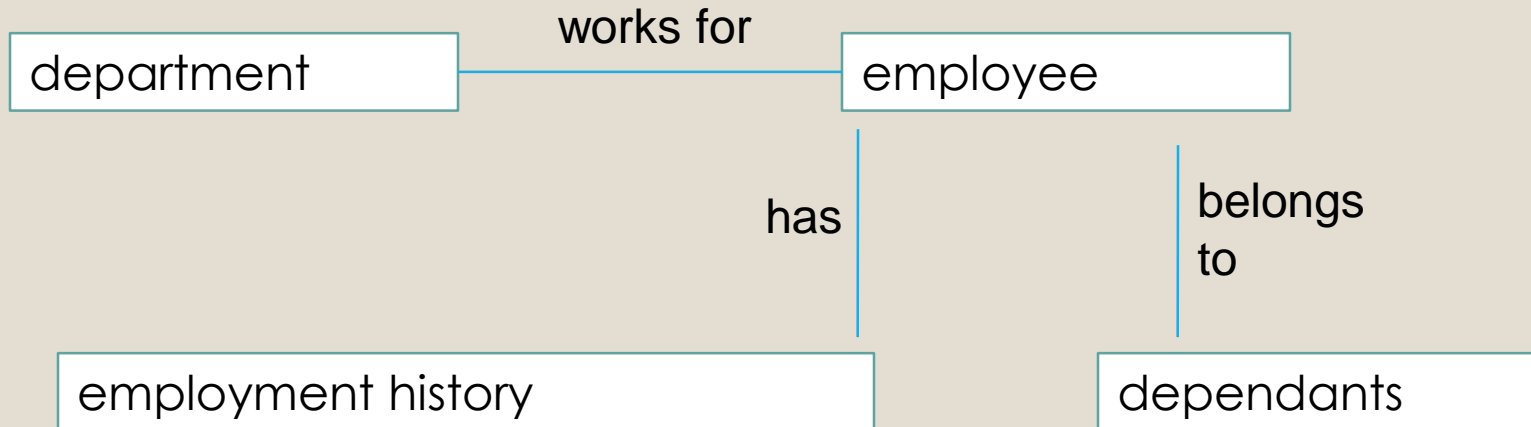
dependants

employment history

Example 1: Each department employs one or more employees and each employee works for one department. Each of the employees may or may not have one or more dependants, and each dependant belongs to one employee. Each employee may or may not have an employment history.

- Step 2: Find the **relationships** i.e. verbs linking the entities.
  - Employs; works for; have; belongs; have
- Step 3
  - Draw a **draft ERD**, showing entities connected by relationships based ONLY on verbs given.

### Step 3. Connect Entities to Relationships

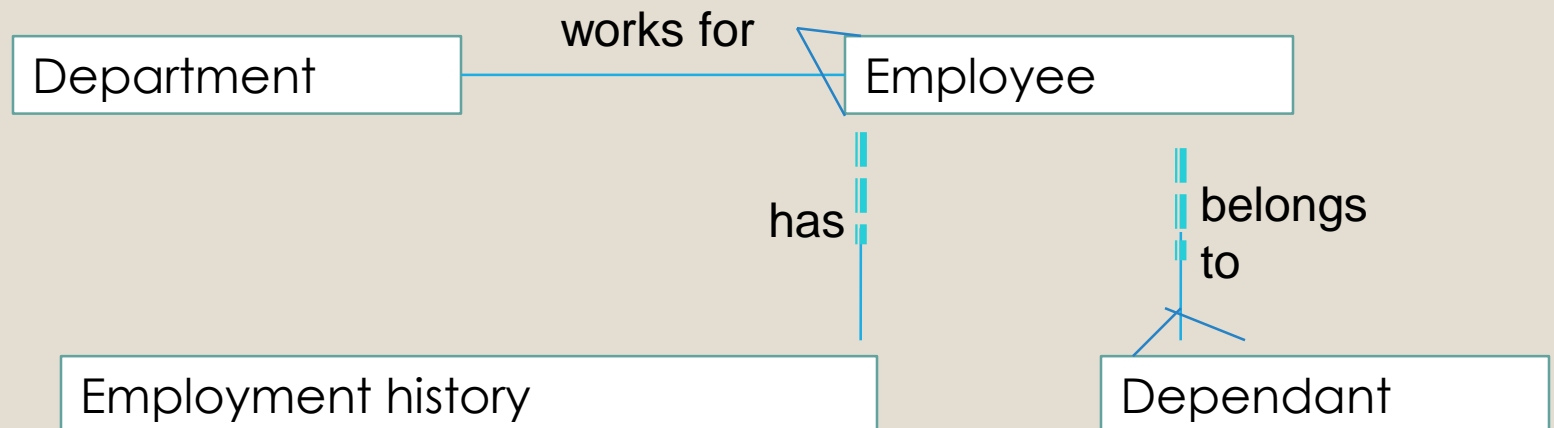


#### ◦ Step 4

- Assign **a name** to each relationship (using a verb or verb phrase).

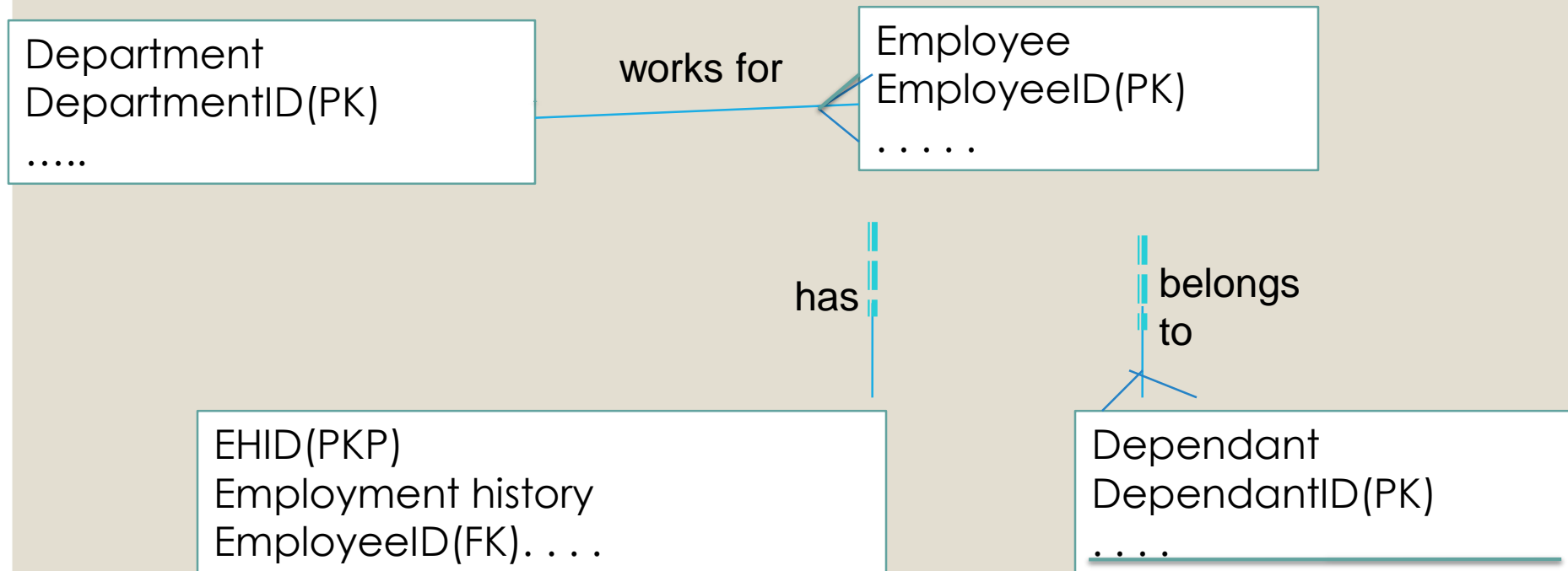
- **Step 5**

- Indicate the **cardinality** of the relationship
- 1:1, 1:m, m:n
- Indicate in the ERD whether a relationship is **optional or mandatory**





- **Step 6**
  - **Resolve** many:many (m:n) relationships
  - (there are none in this example)
- **Step 7**
  - Add attributes
  - Highlight the **Primary Key** in the entities



- **Example 2** : A cooking club organizes several dinners for it's members. A database is used to track which members attends each dinner, the menus used at each dinner, and which member designed the menu.
- Dinners include one starter, one main course, one dessert, and recommended wine.
- Invitations are issued for each dinner, giving the time and place of the dinner. This invitation is sent to each member.

- **Step 1:** List all the entities (nouns) NOT including the company/organization itself.

◦

---

---

---

---

- **Are there any duplicate entities?**
  - If you think of the attributes you would include for each entity, is there a similar list for any two entities?

- Step 2: Find the **relationships** i.e. verbs linking the entities.

- 

---

---

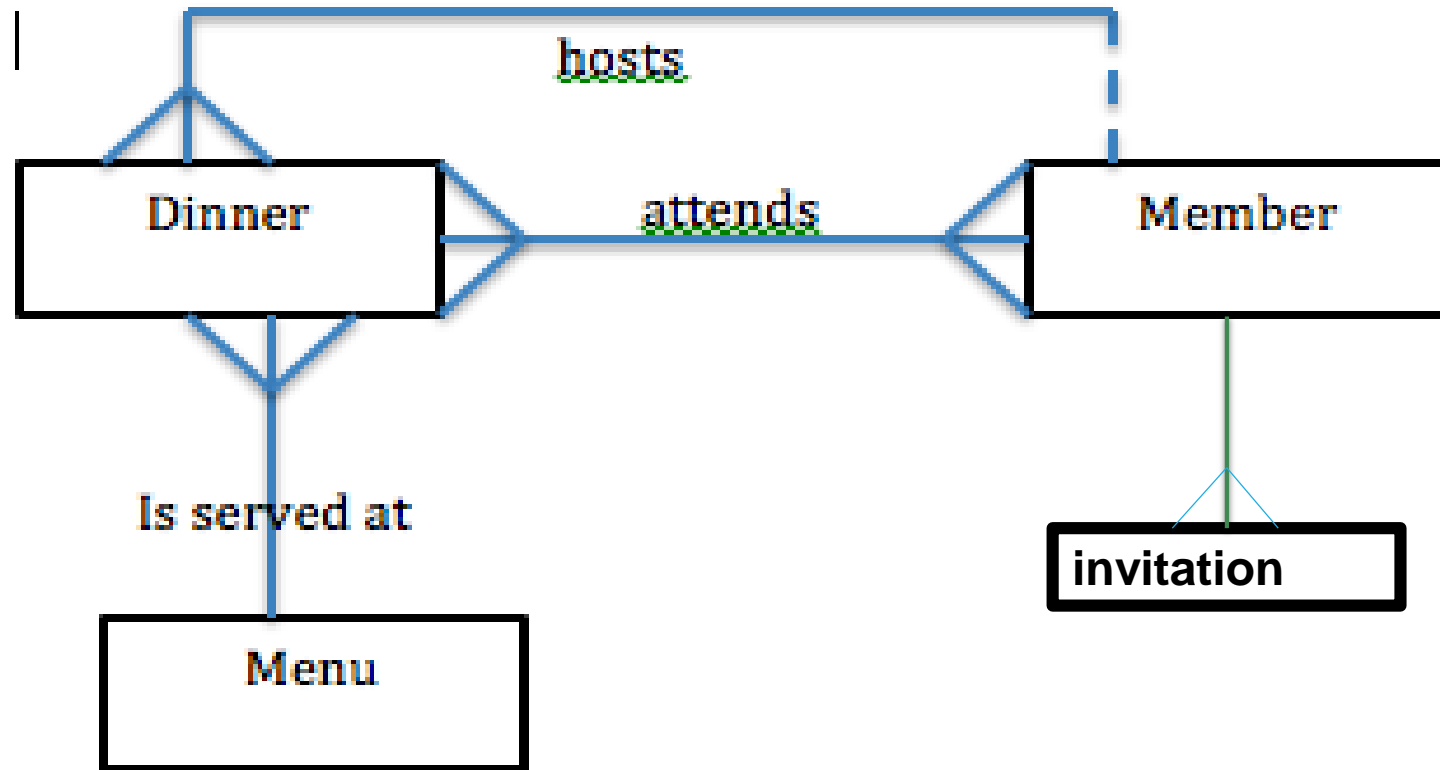
---

---

- **Step 3**

- Draw a **draft ERD**, showing entities connected by relationships based **ONLY** on verbs given.

### Step 3. Draft ERD



#### Step 4

- Assign **a name** to each relationship (using a verb or verb phrase).

- **Step 5**

- Indicate the **cardinality** of the relationship
- 1:1, 1:m, m:n
- Indicate in the ERD whether a relationship is **optional or mandatory**

- **Step 6**

- **Resolve** many:many (m:n) relationships by adding a link entity.
- Is this link entity storing the same information as an existing entity? If so remove the duplicate entity.

- **Step 7**

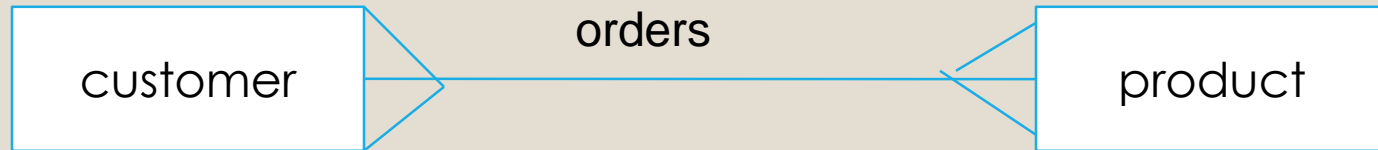
- Add attributes
- Highlight the **Primary Key** in the entities

# Point to note on ERD's

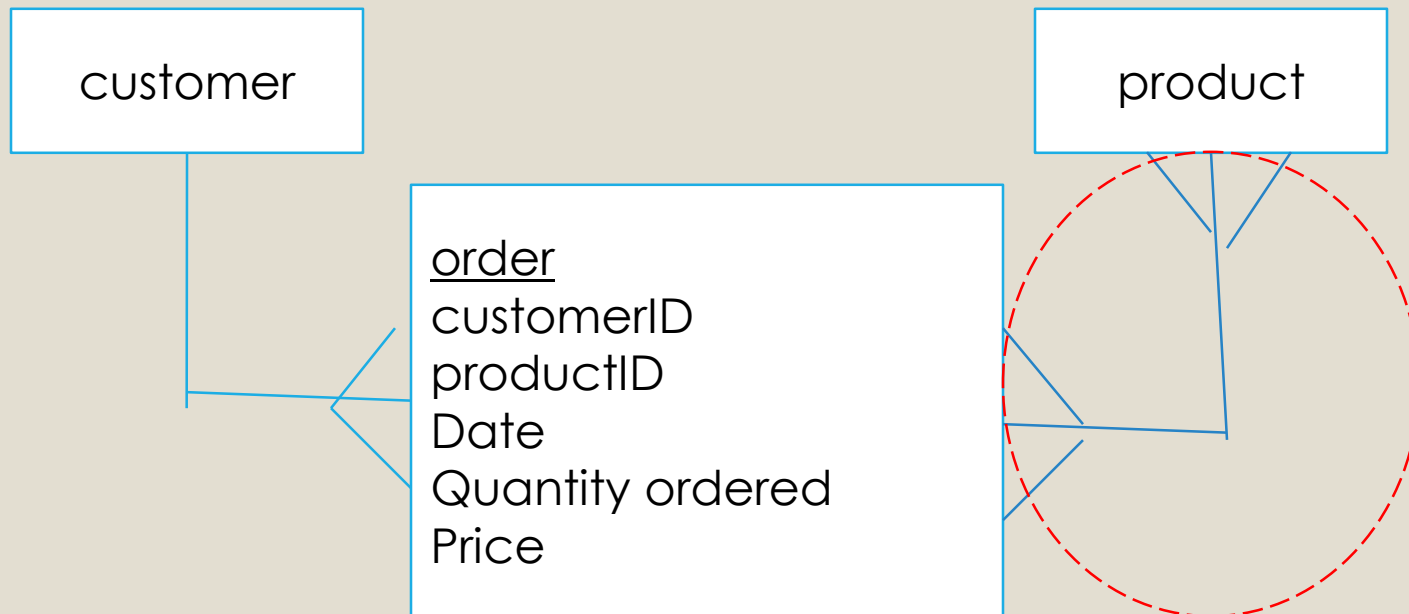
- Recap: Four types of Entities:
  1. **People**: customer, supplier, employee
  2. **Products**: car, book, food item, clothing, etc.
  3. **Services**: holiday booking, eating out, hair cut
  4. **Recording transactions**: deposit, withdrawal, order, invoice, bill, receipt
- **People**, **Products** and **Services** are generally **EASY** to model. **Confusion** & mistakes often arises when recording details of a business transaction, such as an invoice, an order, billing details etc.
- Such transactions often arise as a link entity between two entities, arising from a m:n relationship.

# Point to note on ERD's

- For example:



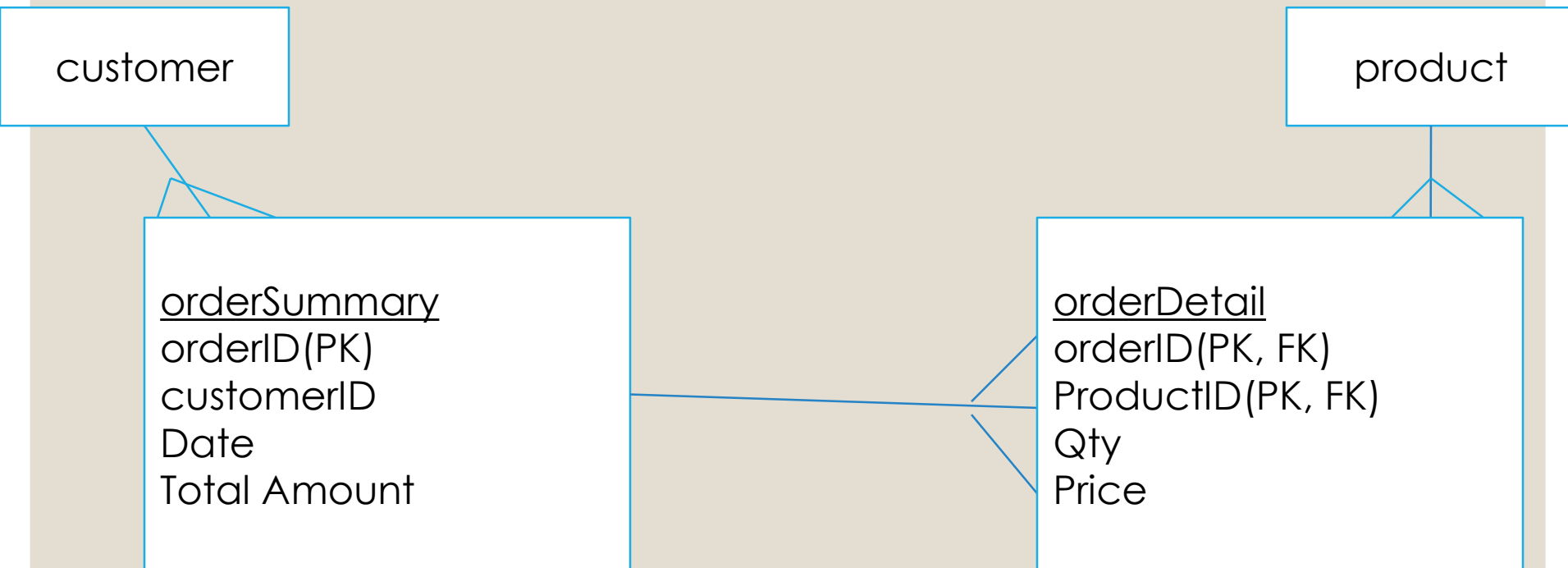
If you want to store information (attributes) about the order, this BECOMES .  
..





# Point to note on ERD's

- However, while an order will only be for one customer, an order can include more than one product, so there is still a M:N relationship!
- In practice, such transaction details are stored across two tables, one with the order summary information, and the other with the order details as follows:



# Point to note on ERD:

- In a database table, this gives the following:

## Full Order

OrderID	CustomerID	Date	ProductID	Qty	Price
1001	C111	10Feb2010	P654	2	20.00
1001	C111	10Feb2010	P991	5	15.00
1001	C111	10Feb2010	P292	2	60.00

## BECOMES

### Order Summary

OrderID	CustomerID	Date
1001	C111	10Feb2010

### Order Detail

OrderID	ProductID	Qty	Price
1001	P654	2	20.00
1001	P991	5	15.00
1001	P292	2	60.00

Objective: Ensure information (like order date) does not have to be entered more than once

# Most business transactions follow the same pattern on an ERD:

## Person entity:

e.g. the Customer who places an order or receives a bill / invoice

## Transaction

e.g. details that appear once on the transaction such as customer ID, transaction date,,

## Product or service entity:

e.g. the products or services listed on the order or invoice

## Transaction details:

e.g. attributes that can have more than one value:  
Product ID, quantity, price

# Examples . . .

## Customer:

Customer ID (PK)  
Customer Name  
Customer Address  
Customer Phone number

## Order

Order ID (PK)  
Order Date  
Customer ID (FK)

## Product:

Product ID (PK)  
Product Description  
Price  
Quantity in Stock

## OrderDetails:

OrderID (PK)  
ProductID (FK)  
QuantityOrdered  
Price



# EXTENDED ERDS

EER

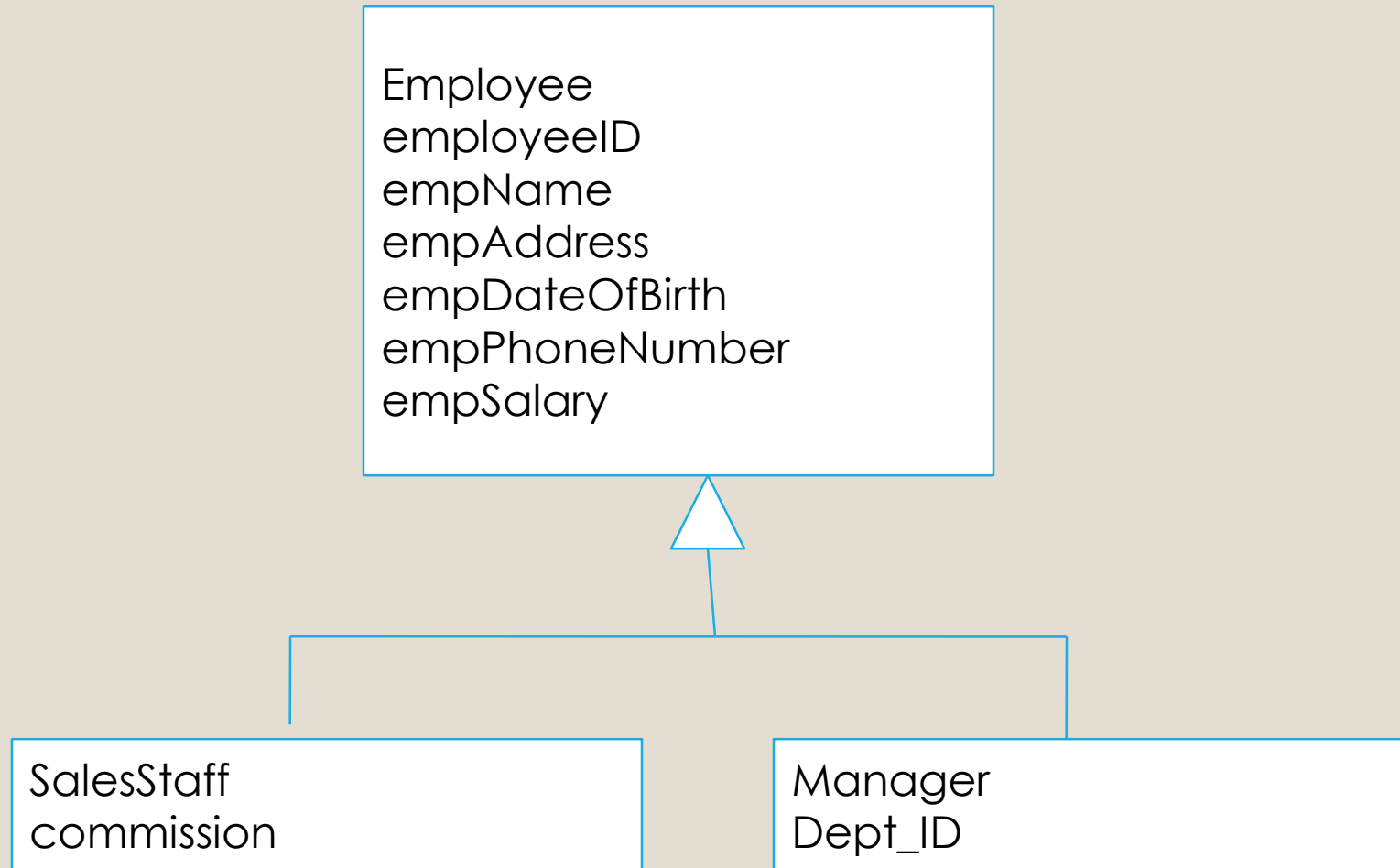
# Enhanced ERD's

- For next weeks lab exercises, you will be using MySQL Workbench to draw Entity Relationship Diagrams.
- MySQL Workbench supports Enhanced Entity Relationship diagrams (EER's).
- EER's model a system in the same way as ERD's with the addition that they support **INHERITANCE**.

# Inheritance

- Inheritance is used when you have two entity types which have almost the same attributes, but one or two attributes are different.
  - For example: if you are modeling sales staff and managers, most of the details will be the same (employeeID, name, phone number, start date, date of birth etc.) but a manager can have an additional attribute of 'deptID' showing the department he/she manages. This is illustrated as follows:

# Inheritance





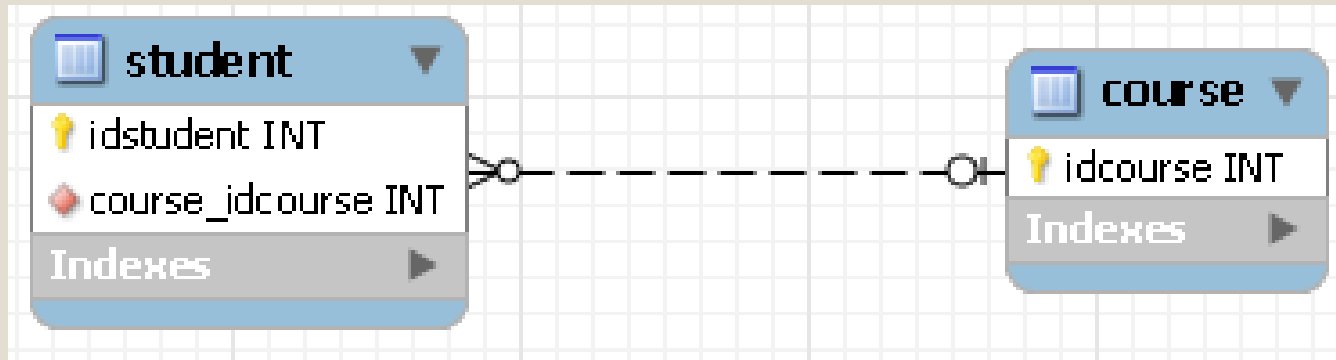
# Inheritance

- Sales staff inherit all the attributes of employee, and have an additional attribute of 'commision'. So the full list of attributes for SalesStaff is: **employeeID, empName, empAddress, empDateOfBirth, empPhoneNumber, empSalary, commission**
- Manager inherits all the attributes of employee as well, and has an additional attribute of 'dept\_id'. So the full list of attributes for manager is: **employeeID, empName, empAddress, empDateOfBirth, empPhoneNumber, empSalary,dept\_id**

# MySQLWorkbench notation

- Unfortunately, the notation used in MySQLWorkbench is a little different to the notation we have been using so far....
- All relationships are shown as a broken line:
- Mandatory and optional are shown as a 0 or | at the end of the relationship

# MySQLWorkbench notation



1:m relationship that is optional at both ends

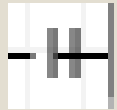
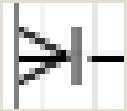
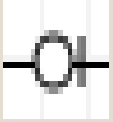



1:m relationship that is mandatory at both ends

# MySQLWorkbench notation

Notation	Lecture notes	MySQLWorkbench
Optional relationship	Dotted line	Dotted line with a circle at the end
Mandatory relationship	Solid line	Dotted line with vertical line at the end.
The 'many' side of a relationship	Crows foot	Crows foot
The 'one' side of a relationship	Single line	Single line with a vertical line at the end

# MySQLWorkbench notation

Symbol	Meaning
	The 'one' side of a mandatory relationship
	The many side of a mandatory relationship
	The 'one' side of an optional relationship
	The many side of an optional relationship

# Exam question 1 (2006)

(a) An engineering company, ENGCO, wishes to create a database to cater for its training program as follows:

ENGCO has 12 instructors and can handle up to 30 trainees per class. It offers six “advanced technology” courses, each of which may generate several classes. If a class has fewer than 10 trainees in it, it may be cancelled. Each class is taught by one instructor. Each instructor may teach up to two classes or may be assigned to do research only. Each trainee may take up to two classes per year.

Design and draw an Entity-Relationship Diagram that captures this information. Indicate clearly any assumptions that you make.

**(12 marks)**

# Exam question 2 (2006 repeat)

(a) FOOD4U is a catering company which provides a dining services to major companies as follow:

The company caters for major events organized by their clients. It can offer clients a number of set menus for these events, each with a wide-variety of dishes. The more popular dishes appear on a number of menus. Staff from the catering company work at the events and one member of staff acts as a catering supervisor.

Design and draw an Entity-Relationship Diagram that captures this information. Indicate clearly any assumptions that you make.

**(12 marks)**

# Stage 2: Relational Model

The next stage in the database **table design** is to convert the Entity Relationship Diagram into a **Relational Model**.



# Relational Data Model

## ADVANTAGES

- Highly standardised (Codd) with SQL
- Widely used
- Well supported commercially - tools available
- Requires less system development & maintenance time.

## DISADVANTAGES

- Resource hungry
- Some lack of acceptance for larger (transaction-based) systems
- Inefficiencies when analysing/reporting large volumes of data (key and index tables drastically slow down applications)
- Difficult to construct models at early stages

# Creating a relational model

- Each **entity** type in an ERD becomes a **relation** in the relational model.
- Each **attribute** in an ERD becomes an **attribute** in the relational model.
- Relationships** in an ERD are represented as **Foreign Keys** in the relational Model
- Domain** refers to valid values for each **attribute**
- Tuples** are the individual **entities**, i.e. the rows of data that will be in your table

# Producing a Relational Model:

## Representing entities as relations

Each entity is written as follows:

Student
student ID(PK)
name
address

becomes:

***Student (Student ID(PK), name, address)***

# Producing a Relational Model: Foreign Keys

- A foreign key is an attribute in a table which is the primary key of another table
- e.g. CourseID in the student table below is a foreign key

**Student (Student ID(PK), name, address, Course ID(FK))**

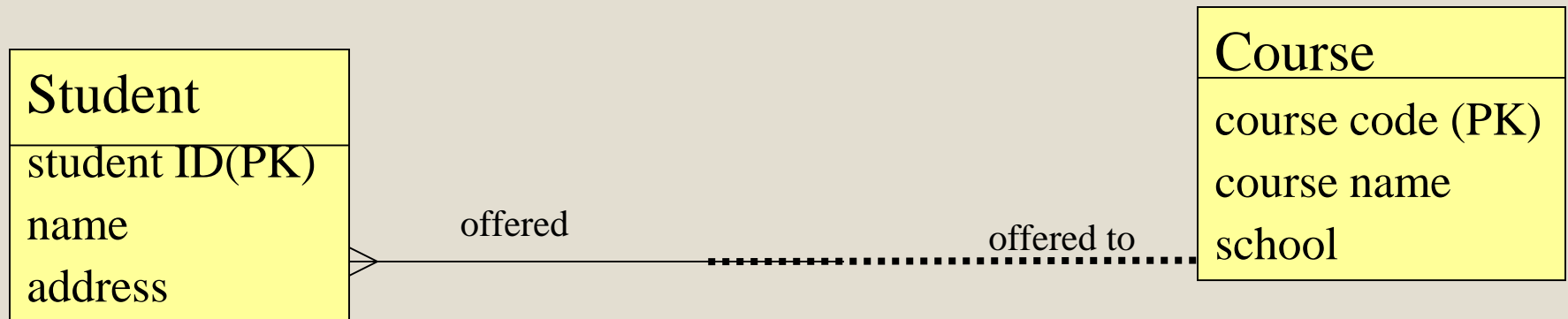
**Course (Course ID(PK), Course name, School)**

Student ID	Name	Address	Course ID
99123456	P. Hardy	Dublin 12	BN001
99456123	J.King	Dublin 15	BN002
99452112	S. O'Neill	Dublin 13	BN001
9945885	D. Casey	Dublin 15	BN001
99754412	F. Cashman	Dublin 11	BN002

Course ID	Course Name	School
BN001	Cert. in Engineering	I & E
BN002	Cart. in IT	I & E

# Producing a Relational Model/ Table: Representing relationships as foreign keys 1:M

- There are three ways of doing this, depending on the cardinality of the relationship, i.e. 1:1, 1:M, M:N
- For 1:M relationships, the primary key of the one side is added as a foreign key to the many side



- course code is added as a foreign key to the student relation giving:

**Course** (course code(PK), course name, school)

**Student** (student ID(PK), name, address, course code(FK))

## Producing a Relational Model/ Table: Representing relationships as foreign keys 1:M

- Add the Primary Key from the **One (1)** side of the relation to attributes of the **Many side**.
- It is known as **Foreign key** on the **Many (m)** side.
- Foreign key acts as a **Link** between the entities



## Producing a Relational Model/ Table: Representing relationships as foreign keys 1:1

- For a **1:1 relationship**, the **primary key** of one side is added as a **foreign key** to the other side, but it does not matter which side the **foreign key** is added to
- So the following ER diagram can become:



*Director (Emp ID(PK), Name, Salary, College Code(FK))*

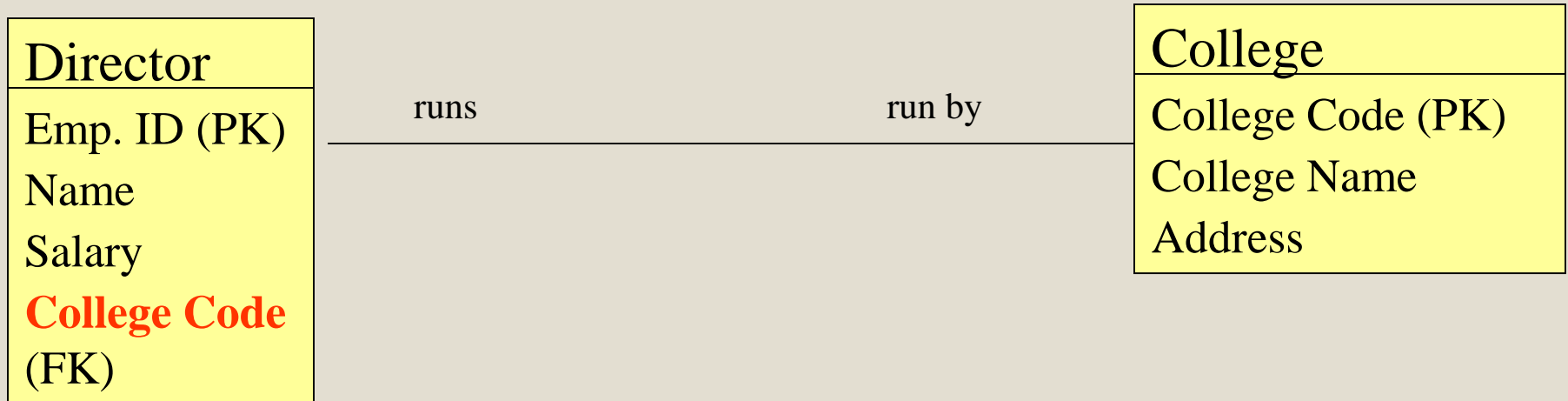
*College (College code (PK), College Name, Address)*

OR

*Director (Emp ID(PK), Name, Salary)*

*College (College code (PK), College Name, Address, Emp ID(FK))*

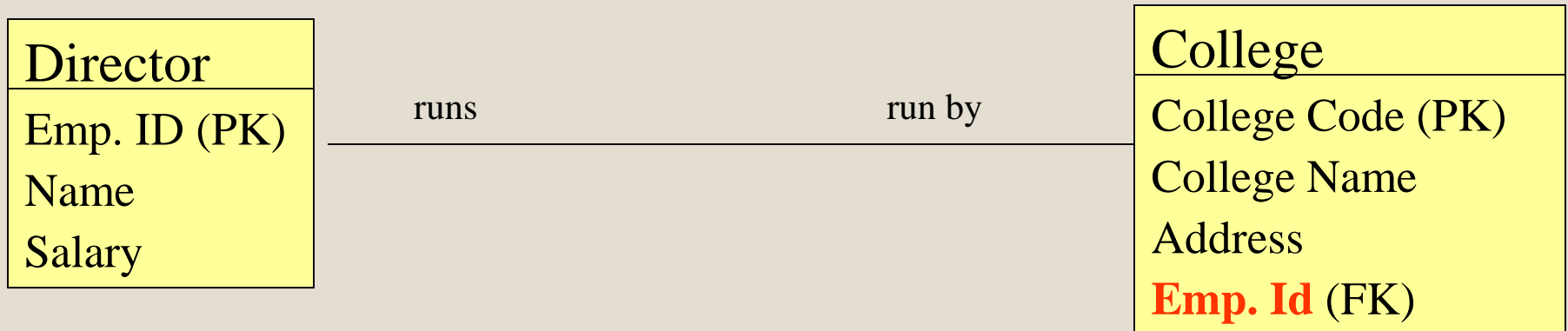
## Producing a Relational Model/ Table: Representing relationships as foreign keys 1:1





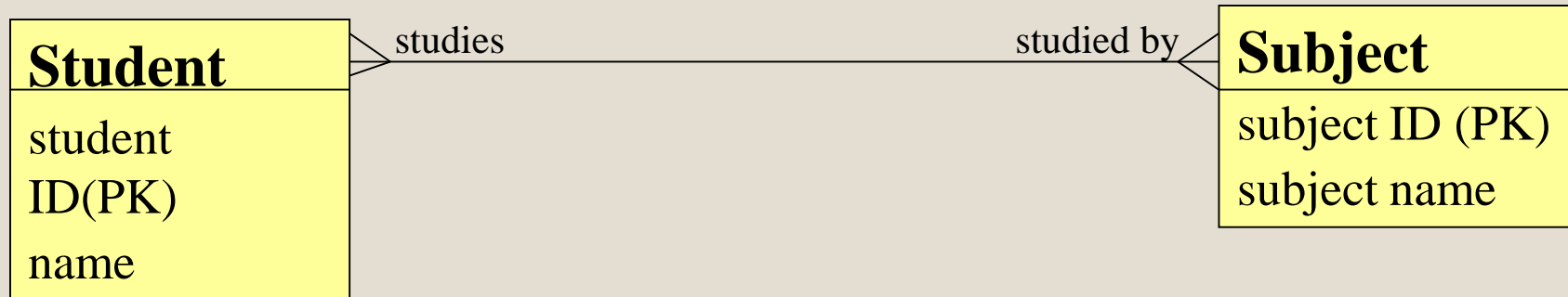
# Producing a Relational Model/ Table: Representing relationships as foreign keys 1:1

or



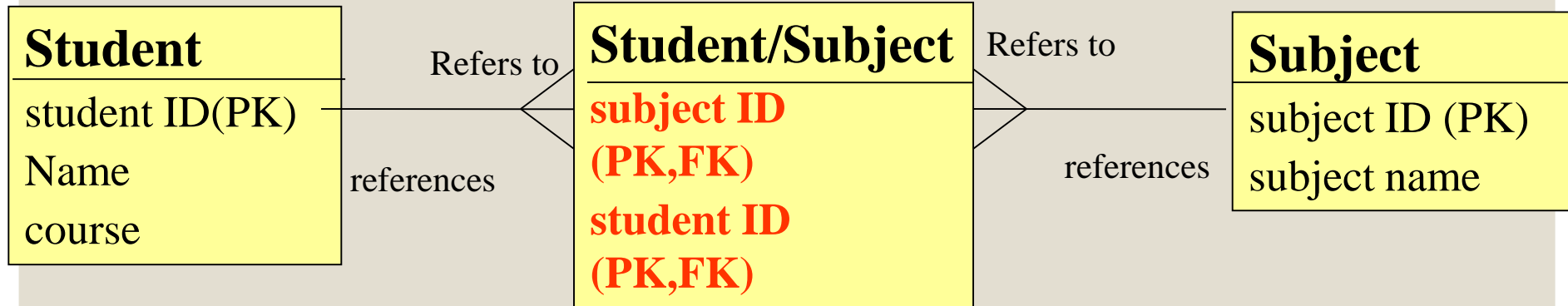
## Producing a Relational Model/ Table: Representing relationships as foreign keys M:N

- For a **m:n** relationship, you must create a new relation, with **two attributes**, the **primary key** from each of the original entity types.



- Create a **LINK ENTITY**
- So the ER diagram becomes:

## Producing a Relational Model/ Table: Representing relationships as foreign keys M:N



***Student (student ID(PK), name, course)***

***Subject (subject ID(PK), subject name)***

***Student\_Subject (student ID(PK, FK), subject ID(PK, FK))***

# Where would the foreign key go in each of the following?

Customer

Product

Customer

Bank  
Account

Customer

Company  
Car

# Finishing the relational model .

- You would also check at this stage:
  - Does every relation have a primary key?
  - Is there any composite attributes?
  - Is there any multi-value attributes?
    - If so, create a new relation for the composite attribute with the same primary key as the original attribute
  - Are there duplicate relations – i.e. do two relations have the same (or similar) attributes and can they be merged?
    - e.g. customer and client, or employee and manager . .

# Example of data:

Student Table

Student	Name	Course
99143757	John Murphy	BN002
99123456	Mary O'Reilly	BN002
99454545	Paul Ryan	BE002

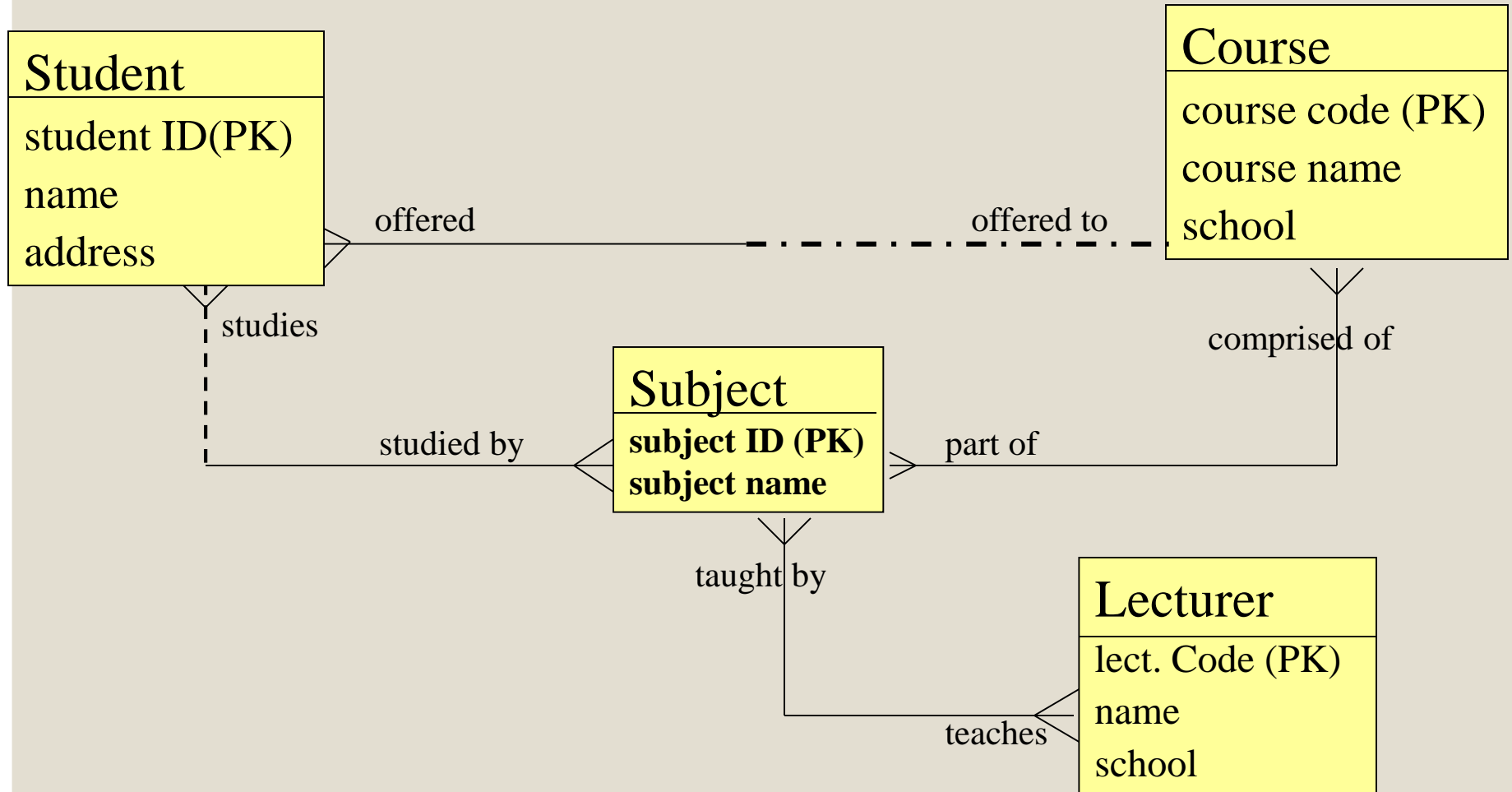
Student / Subject

Student ID	Subject ID
99143757	M1
99143757	F1
99143757	Sdev1
99143757	DB1
99123456	M1
99123456	Sdev1
99123456	MM1
99123456	DB
99454545	MM1

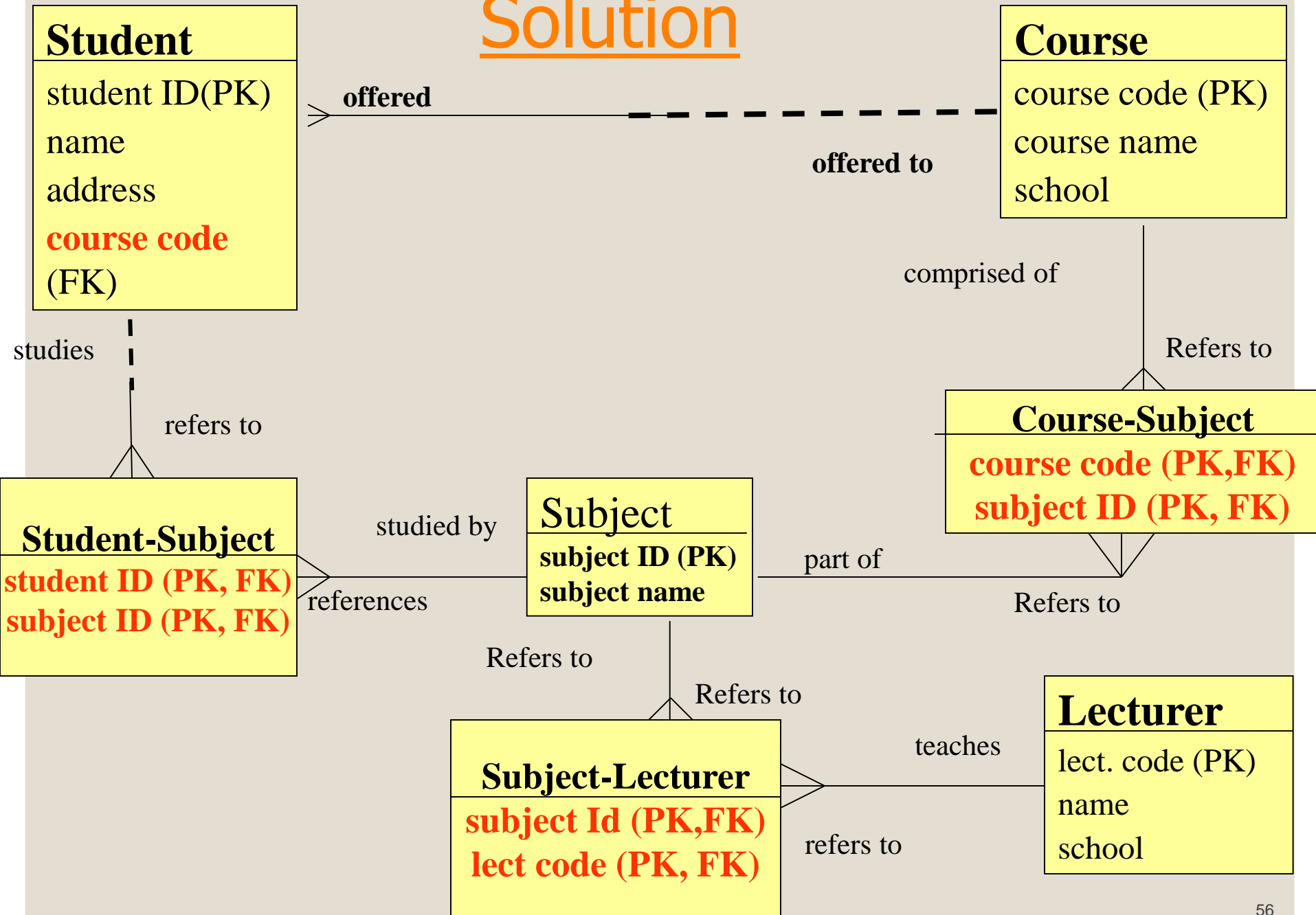
Subject Table

Subject ID	Subject Name
M1	Maths Semester 1
M2	Maths Semester 2
M2	Maths Semester 3
Sdev 1	Software Development Semester 1
DB1	Database Technology 1

# Exercise: Produce a relational model for the following:



# Solution





# Relational Model/Table

Student (Student ID (PK), name, address, course code (FK))

Course (Course code (PK), course name, school)

Course-Subject (Course code (PK, FK), Subject ID (PK, FK))

Subject (Subject ID (PK), subject name)

Student-Subject (Student ID (PK,FK), Subject ID (PK,FK))

Subject-Lecturer (Subject ID (PK, FK), Lect Code (PK, FK))

Lecturer (Lect. Code (PK), name, school)

# Steps for Creating a Relational Model from ERDs

- Starting point is drawing an Entity-Relationship Diagram (ERD)
  - Underline **nouns** in the text
  - Identify what **entities** you need to store data about
  - Underline **verbs** - Identify the relationships between entities
  - Draw a 1<sup>st</sup> draft version of the ERD
  - Draw a 2<sup>nd</sup> version of the ERD and:
    - create **LINK ENTITY** where there are **M:N** relationships
    - label each **relationship line** with a name
    - show whether **relationship** lines are **mandatory** or **optional**

# Steps for Creating a Relational Model from ERDs

- For each entity identify the Primary Key and Foreign Key (if it exists)
  - Identify other attributes for each entity
- 
- Create the tables or relational model of the ERD
  - Use the normalisation process (next) to check that tables are correct.