Database Fundamentals

Lecture 9 (Relational Model & Normalisation)

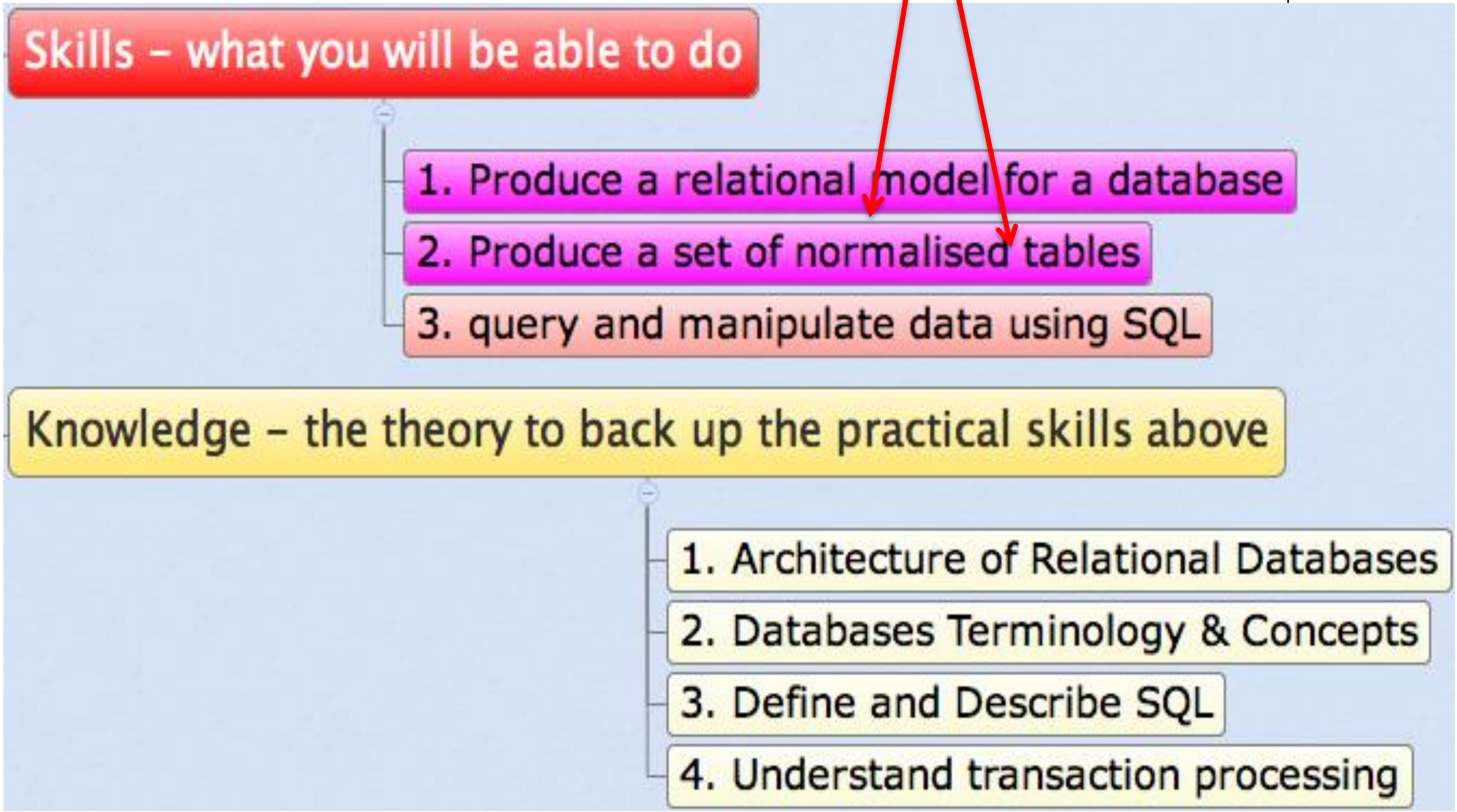
Lecturer : Dr Irene Murtagh

Room :A15

Email: irene.murtagh@tudublin.ie

Geraldine Gray, Maria Brennan, Irene Murtagh 2013

Learning outcomes



Skills – what you will be able to do

1. Produce a relational model for a database
2. Produce a set of normalised tables
3. query and manipulate data using SQL

Knowledge – the theory to back up the practical skills above

1. Architecture of Relational Databases
2. Databases Terminology & Concepts
3. Define and Describe SQL
4. Understand transaction processing

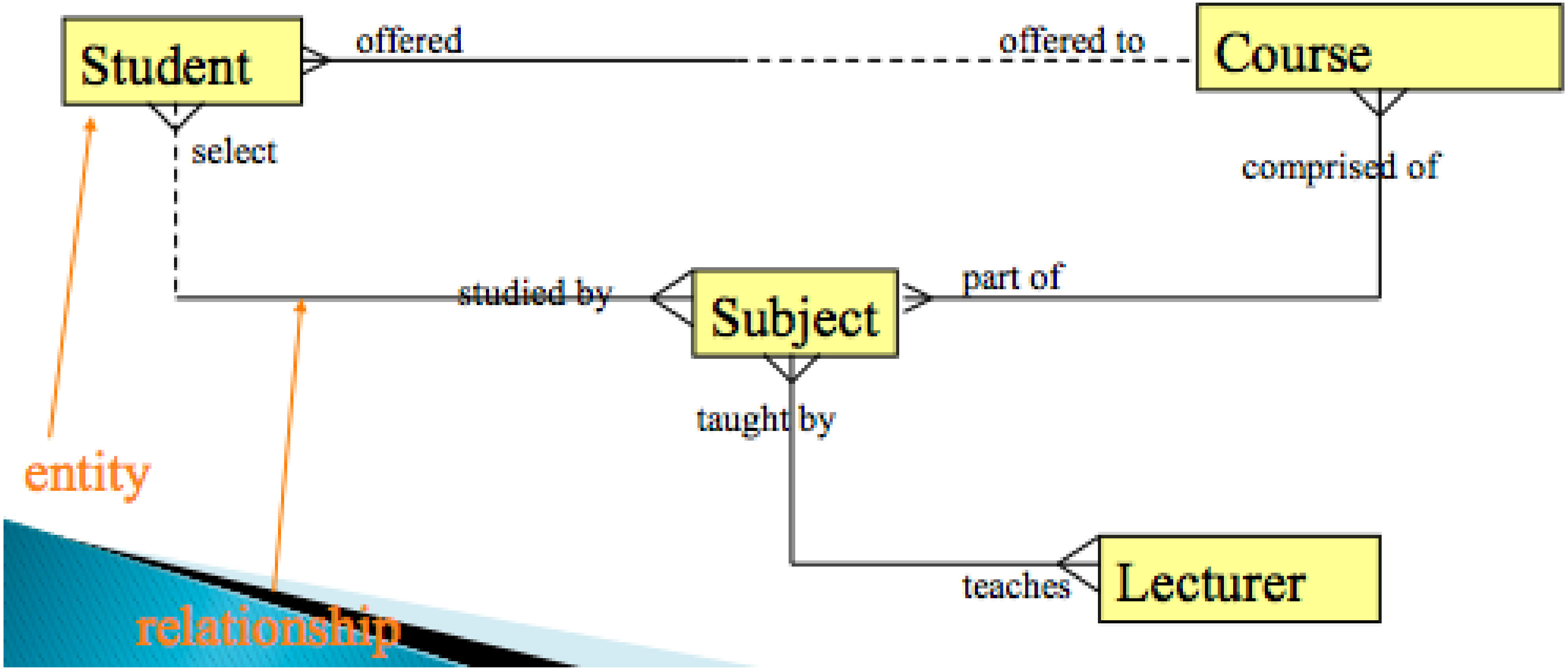
Objective

1. Recap on ERD’s
2. Convert an ERD to a relational model
3. Bring the relational model to 3rd normal form

Re-cap ERDs

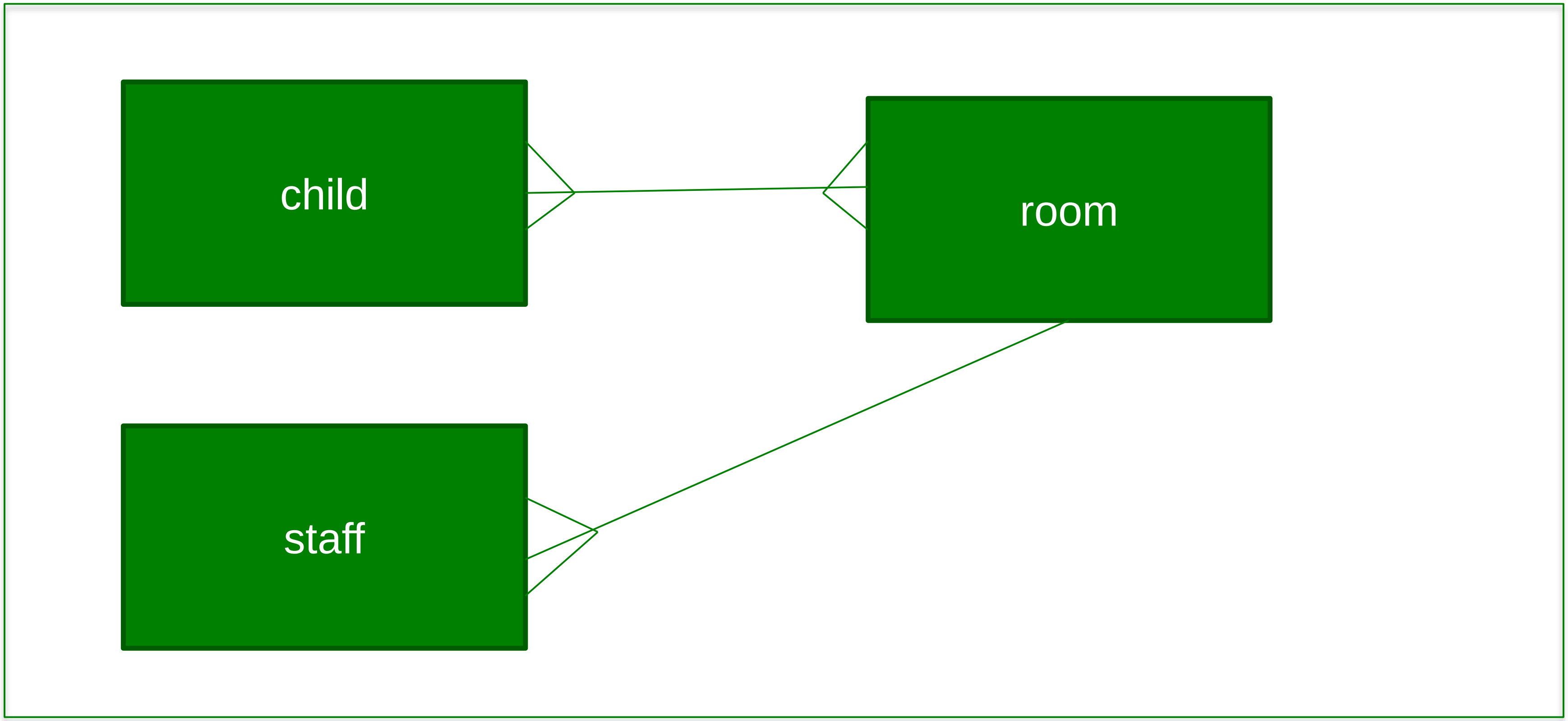
ERDs

- 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.



Draw an ERD for the following:

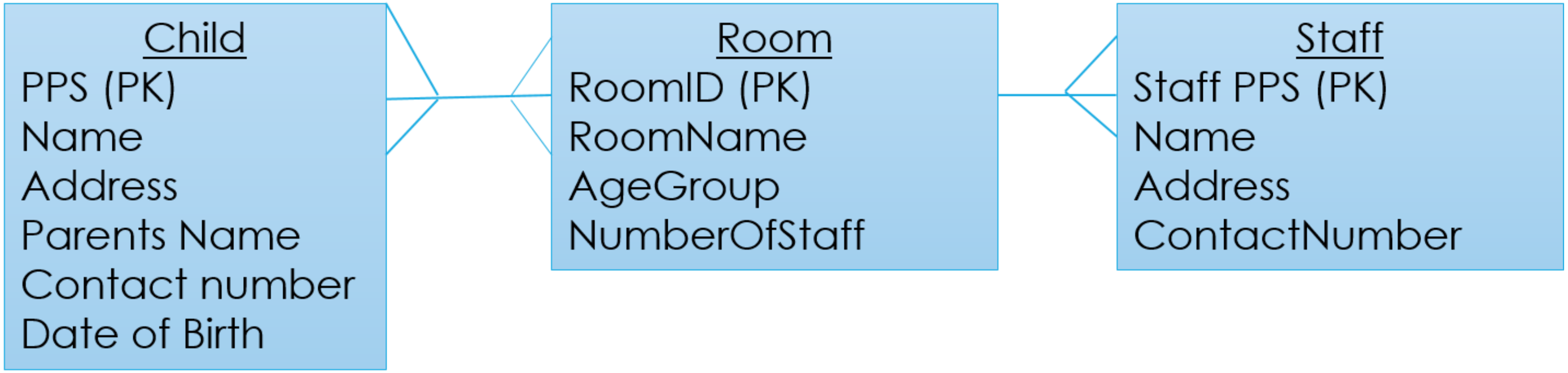
- A crèche needs to record information about the children they mind, the staff they employ, and the rooms in the crèche.
- Children are allocated to particular rooms.
- One or more members of staff are allocated to a particular room



. . . add the attributes

- For each child, the crèche needs to know their PPS number, name, address, parents name, contact phone number of the parent, and child’s date of birth.
- For each room, the crèche records the room name, age group, and number of staff members needed for the room
- For each member of staff, the crèche records their name, address, PPS number and phone number.

ERD with attributes



Child
PPS (PK)
Name
Address
Parents Name
Contact number
Date of Birth

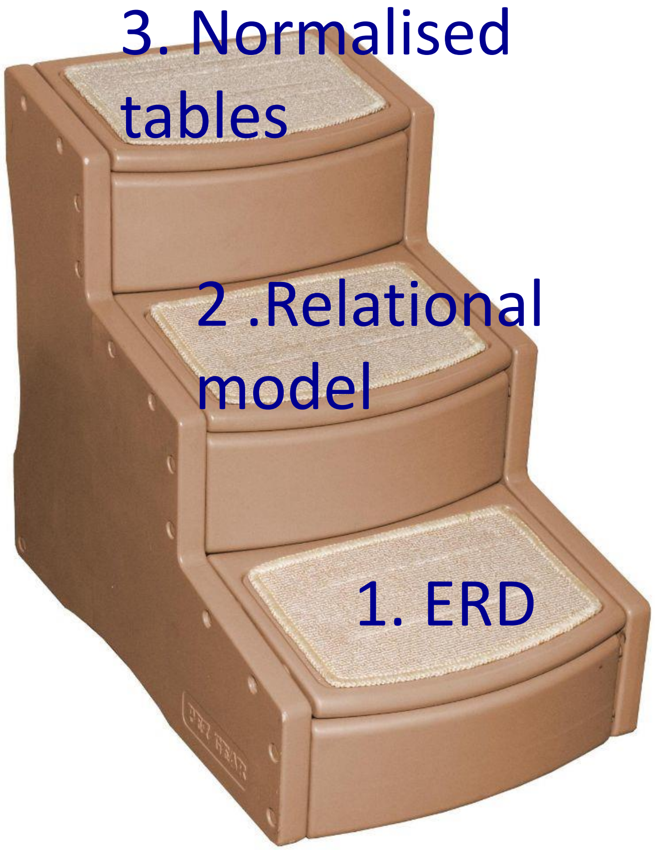
Room
RoomID (PK)
RoomName
AgeGroup
NumberOfStaff

Staff
Staff PPS (PK)
Name
Address
ContactNumber

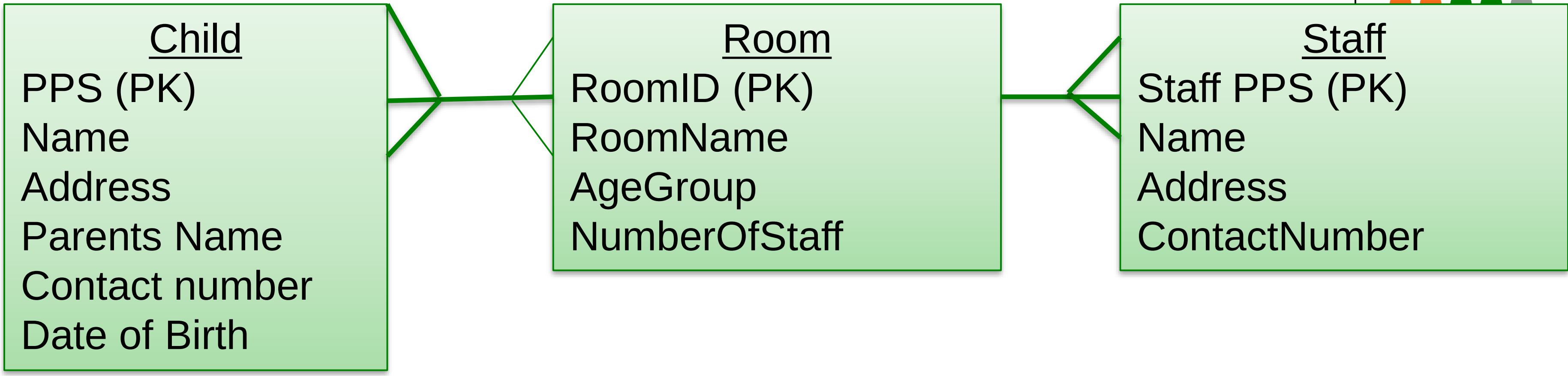
Converting and ERD to a Relational model

Represent relationships as foreign keys

- ERD’s is the first step in data design
  - i.e. identifying the entities which the database needs to record data on to support an application
- The next step is to rewrite the the ERD as a RELATIONAL MODEL, with replaces relationships lines with foreign keys.



Relational model



Child

PPS (PK)
Name
Address
Parents Name
Contact number
Date of Birth

Room

RoomID (PK)
RoomName
AgeGroup
NumberOfStaff

Staff

Staff PPS (PK)
Name
Address
ContactNumber

The Relational model for the ERD above is written as:

- Child(PPS(PK), Name, Address, ParentsName, ContactNumber, DateOfBirth)

- Room(RoomID(PK), RoomName, AgeGroup, NumberOfStaff,

where PK means primary key, and FK means foreign key

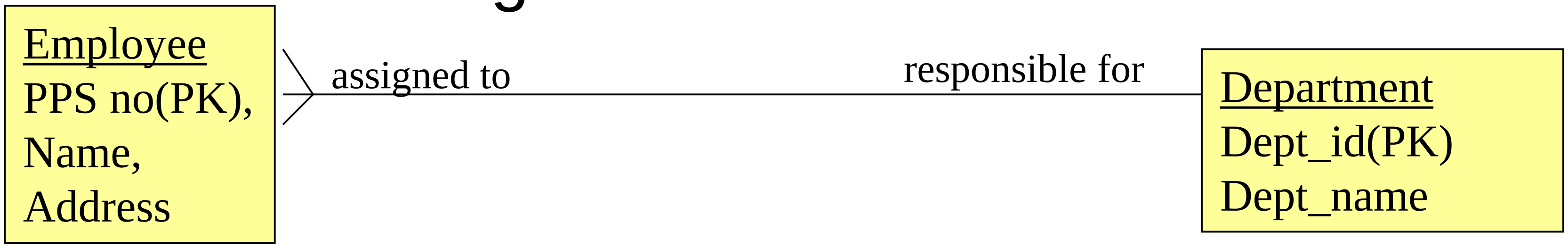
- Staff (staffPPS(pk), Name, Address, ContactNumber, RoomID(FK))
- **Child_Room**(RoomID(PK,FK), PPS(PK, FK))

Converting an ERD to a Relational model

1. Each entity type in an ERD becomes a relation in the relational model.
2. Each attribute in an ERD becomes an attribute in the relational model.
3. Relationships in an ERD are represented as Foreign Keys in the relational model

Another example . . . .

- Take the following ERD:

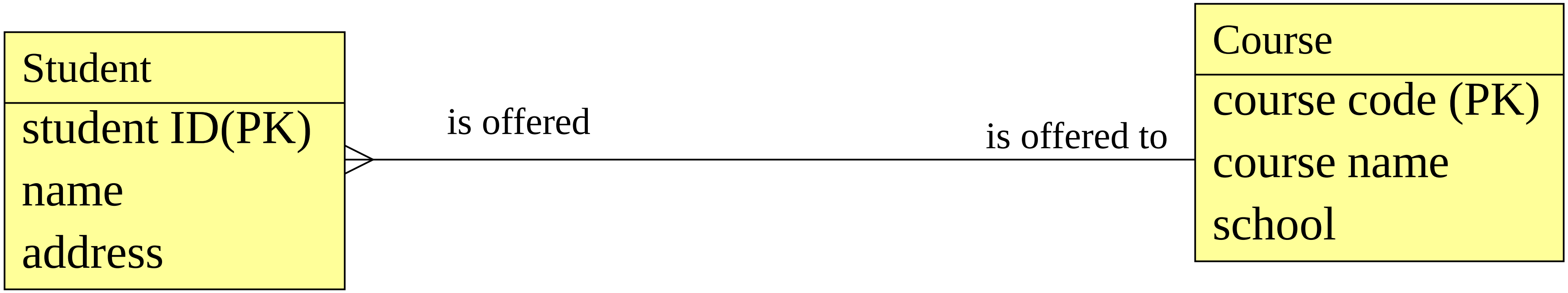


Relational Model

- Relations are written as follows:
  - Employee(PPS_no(PK), name, address)
  - Department(dept_id(PK), name)
- The relations are then linked by adding dept_id as a **foreign key** to the employee table
  - Employee(PPS_no(PK), name, address, dept_id(FK))
  - Department(dept_id(PK), name)

Which table do you add the foreign key to?

- Every relationship in an ERD must be represented using foreign keys
- 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



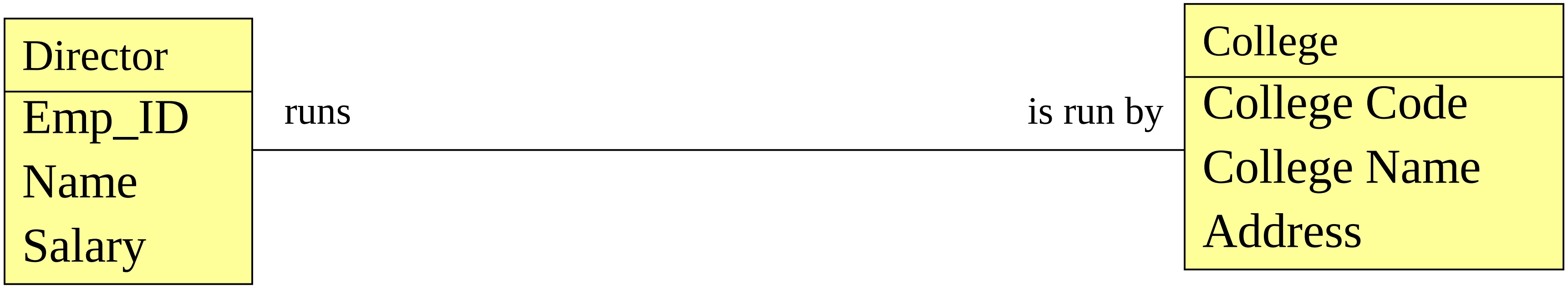
- course code is added as a foreign to the student relation giving:

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

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

Which table to you add the foreign key to?

- 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

Name

Salary

runs

is run by

College

College Code

College Name

Address

Director (Emp_ID(PK), Name, Salary, College_Code(FK))

College (College_code (PK), College_Name, Address)

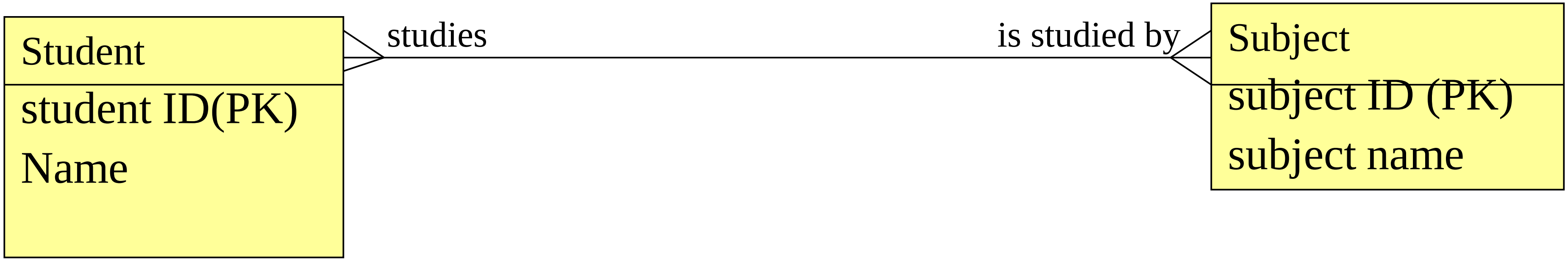
- **OR**

Director (Emp_ID(PK), Name, Salary)

College (College code (PK), College_Name, Address, Emp_ID(FK))

Which table to you add the foreign key to?

- For a m:n relationship, you must create a new relation, which includes the primary key from each of the original entity types.
- So the following ERD becomes:



Student

student ID(PK)

Name

studies

is studied by

Subject

subject ID (PK)

subject name

Student (student ID(PK), student name)

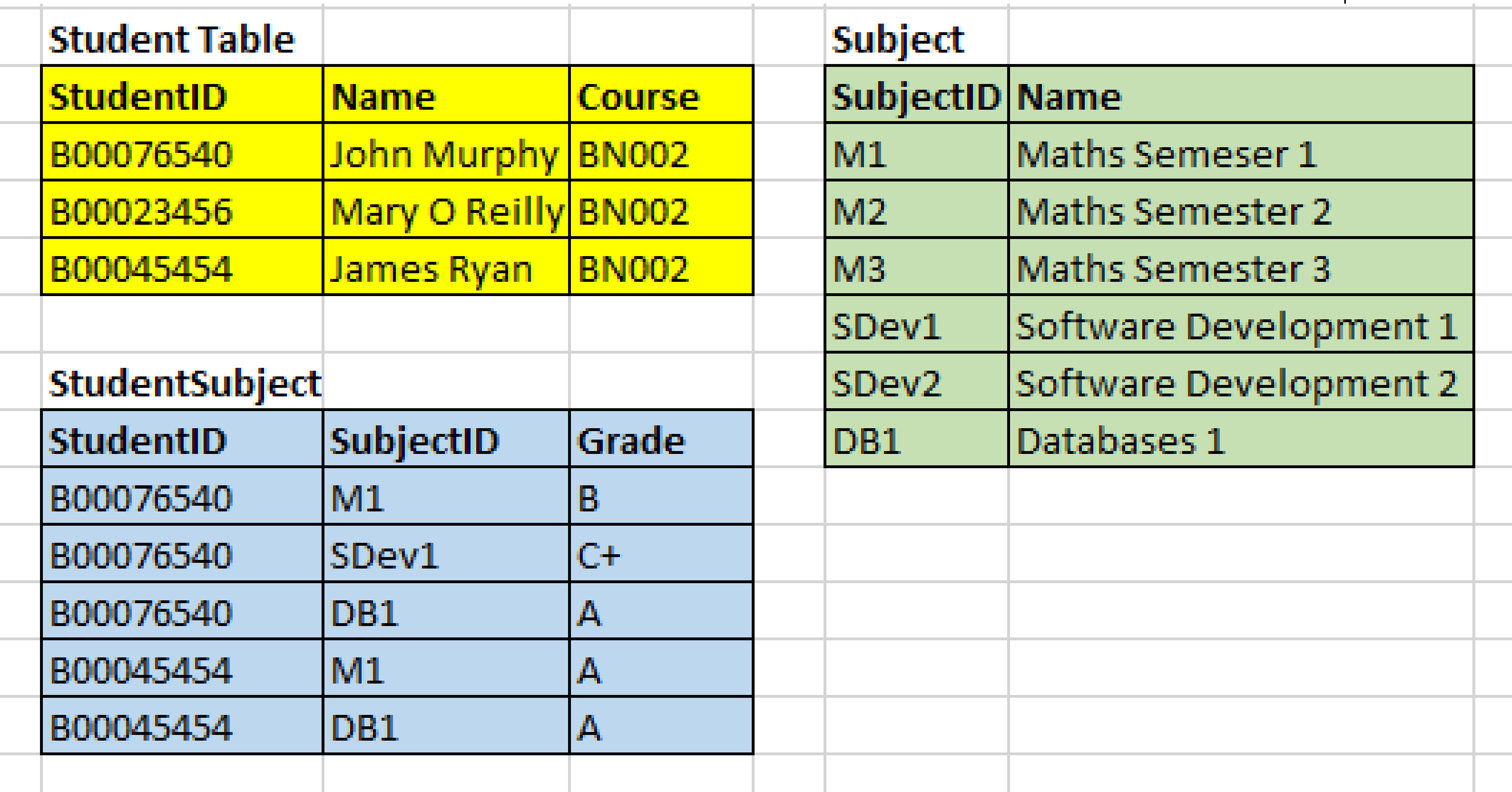
Subject (subject ID(PK), subject name)

Student_Subject (student ID(PK, FK), subject ID(PK, FK))

- Note: If the relationship has attributes, these would be added to the new relation, e.g.

Student_Subject (student ID(PK, FK), subject ID(PK, FK), grade)

Example of data:



Student Table

StudentID	Name	Course
B00076540	John Murphy	BN002
B00023456	Mary O Reilly	BN002
B00045454	James Ryan	BN002

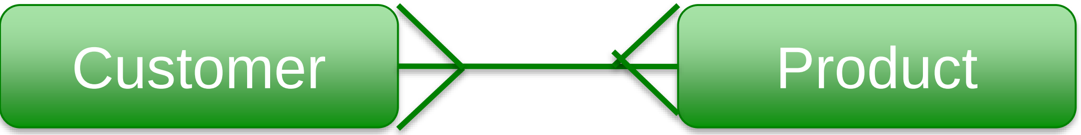
StudentSubject

StudentID	SubjectID	Grade
B00076540	M1	B
B00076540	SDev1	C+
B00076540	DB1	A
B00045454	M1	A
B00045454	DB1	A

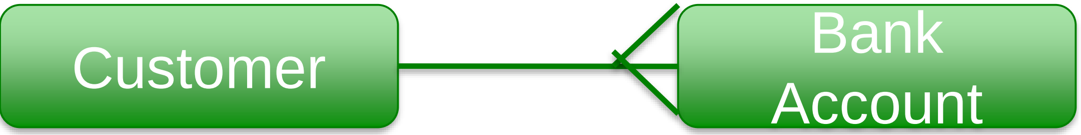
Subject

SubjectID	Name
M1	Maths Semeser 1
M2	Maths Semester 2
M3	Maths Semester 3
SDev1	Software Development 1
SDev2	Software Development 2
DB1	Databases 1

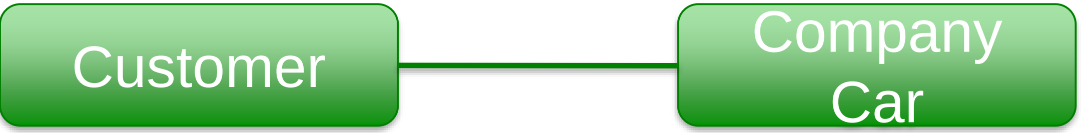
Where would the foreign key go in each of the following?



____



____



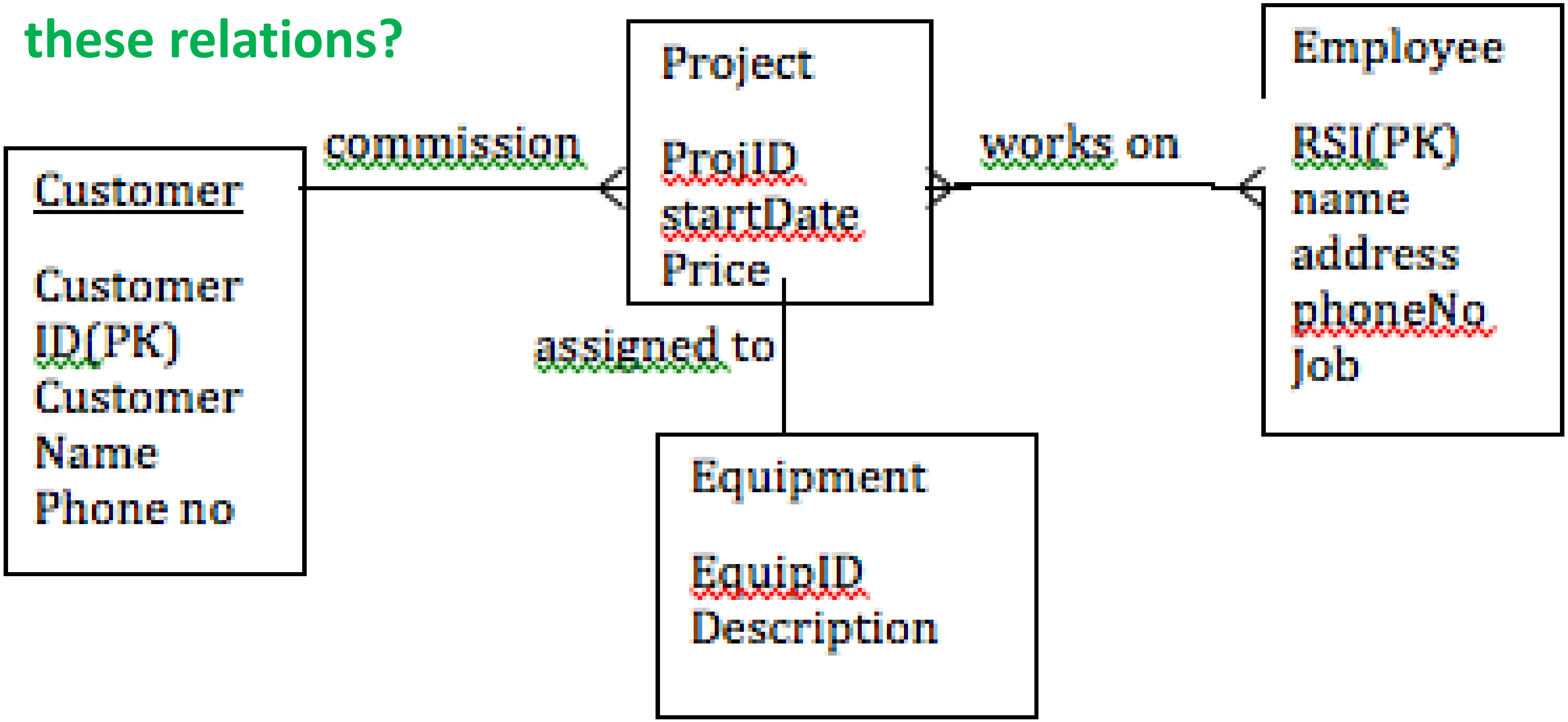
____

Finishing the relational model . . .

- You would also check at this stage:
  - Does every relation have a primary key?
  - Are there any composite attributes?
  - Are there any multi-valued attributes?
    - If so, create a new relational for the multivalued 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 . .

Exercise: Produce a relational model for the following ERD:

Where does the foreign key go in each of these relations?



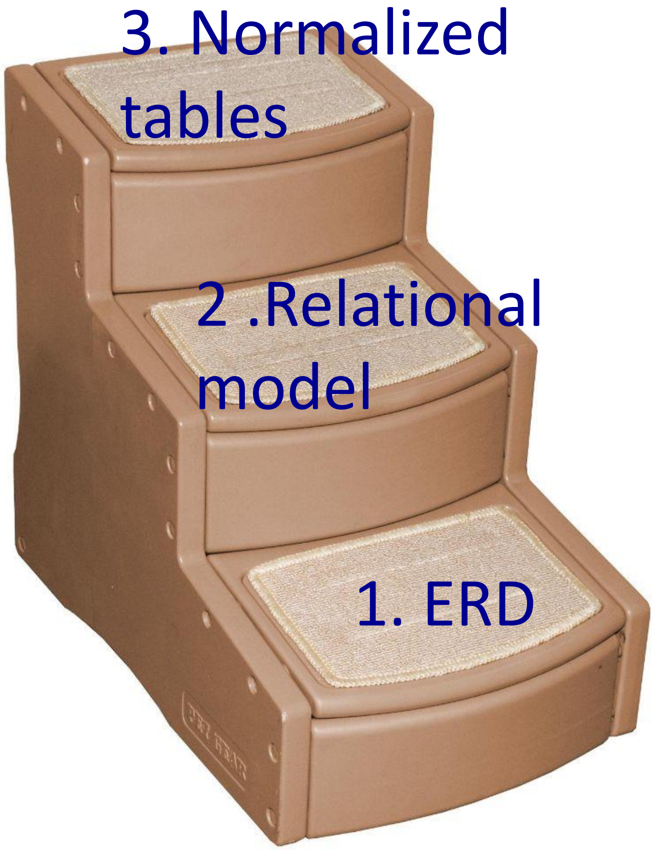
Solution

- **Customer**(**CustomerID(PK)**, CustomerName,Phone No)
- **Project**(**ProjectID(PK),** startdate, price, CustomerID(FK))
-
- **Equipment** (**EquipmentID(PK),** Description, **ProjectID(FK))**
- **Employee (** **RSI(PK),** name, address, phone, job)
- **Employee_Project**(**RSI(PK, FK), ProjectID(PK, FK)**))

Composite Attributes – Name? Address? …

Next step . . .

- Once the ERD is mapped to a relational model, the 3rd and final step is to bring these relations to Third Normal Form.
- Why?
  - To ensure no data is duplicated.



Why avoid duplicate data? To avoid the following three anomalies . . .

Student ID	Student name	Course	Subject	Lecturer
99143757	John Murphy	BN002	Maths	Susan
99143757	John Murphy	BN002	French	Ruth
99143757	John Murphy	BN002	S. Dev	Brian
99143757	John Murphy	BN002	Databases	Geraldine
99123456	Mary O'Reilly	BN002	Maths	Susan
99123456	Mary O'Reilly	BN002	S.Dev	Brian
99123456	Mary O'Reilly	BN002	Multimedia	Hugh
99123456	Mary O'Reilly	BN002	Databases	Geraldine
99454545	Paul Ryan	BE002	Multimedia	Hugh

POOR DATA DESIGN

Why avoid duplicate data? To avoid the following three anomalies . . .

- Update anomaly – suppose the maths lecturer changes from Susan to Colm. How many places would you need to make the change?
- Delete anomaly – if John Murphy leaves the course, there will be no record of who teaches French
- Insertion anomaly – Suppose you want to add a new subject called “Human Language technology”, but there is no student registered for the subject yet. How do you add it the table above?

Well Structured Relations

- Once the relational model is created, the final stage in database design is to **NORMALISE** the data, also called producing a well structured relation.

- A relation is well-structured if all the attributes in the relation are functionally dependent on the primary key.
    - i.e. the attribute has one unique value that can be determined from the primary key.

Example 1

- Student (Student ID(PK), student name, lecturer name, course description)

- Does a student ID identify a specific student's name?
- Does a student ID identify a specific lecturer's name?
- Does a student ID identify a specific course description?

Only student name is functionally dependent on Student ID. The other attributes are in the wrong table.

Example 2

Student_Subject(Student ID(PK), subject ID(PK), grade, student name, subject name)

- Do you need both the student ID and the subject ID to find the grade a student got in a particular subject?
- Do you need both the student ID and the subject ID to get a student's name?
- Do you need both the student ID and the subject ID to get the subject's name?

Only grade is functionally dependent on Student ID **AND** subject ID. The other attributes are in the wrong table.

Example 3

Student ID(PK),	student name	subject name,	grade
99123456	Kelly	Databases	C
99123456	Kelly	Software Dev	B
99123456	Kelly	Networking	C+

- Does a student ID identify a specific student's name? Subject Name? Grade?

Only student name is functionally dependent on Student ID. The other attributes are in the wrong table.

Recap – Functionally dependent means the attribute is a unique value that can be determined from the key field.

Example….

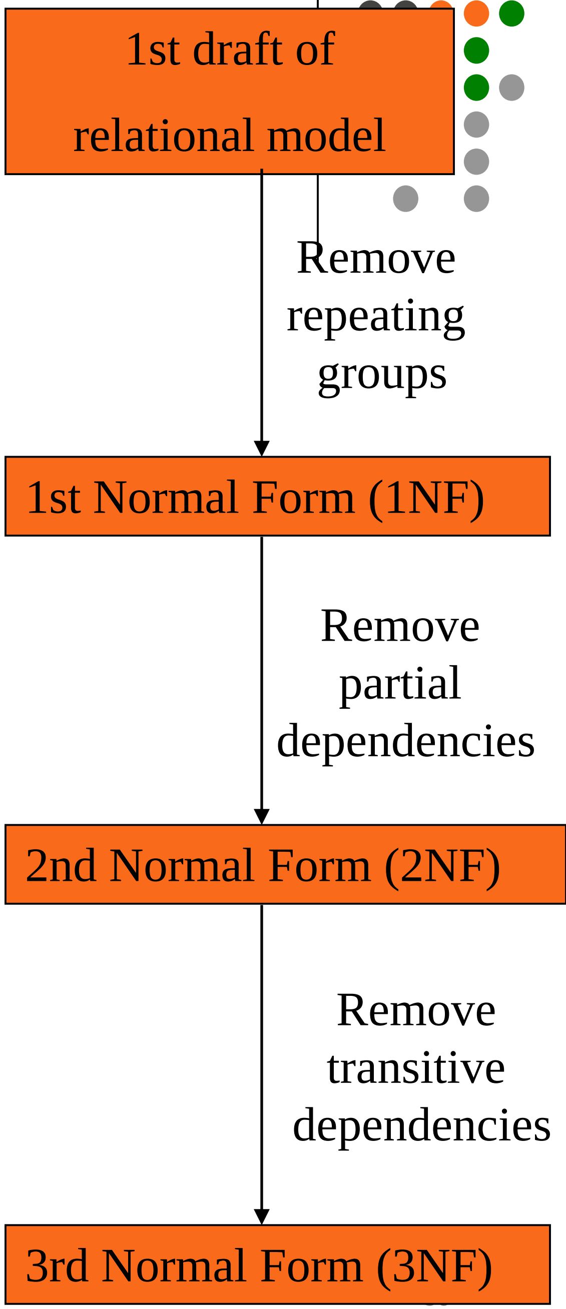
- Lets separate the lists of attributes below into well-formed relations:

1. Product (product ID, description, quantity in stock, supplier name, supplier address, contact name)

2. Order (Order number, order date, customer name, customer address, product id, product description, quantity ordered)

Normalisation

- There are three ways in which an attribute is NOT functionally dependent on the primary key, as illustrated in the three examples done previously.
- Identifying these scenarios is done by following the three steps of Normalisation:
  1. Bring to 1st normal form – remove repeating groups, i.e Example 3 above.
  2. Bring to 2nd normal form – remove partial dependencies, i.e. Example 2 above
  3. Bring to 3rd normal form – remove transitive dependencies, i.e. Example 1 above
- Once in third normal form (3NF), the tables are well-structured



1st draft of relational model

Remove repeating groups

1st Normal Form (1NF)

Remove partial dependencies

2nd Normal Form (2NF)

Remove transitive dependencies

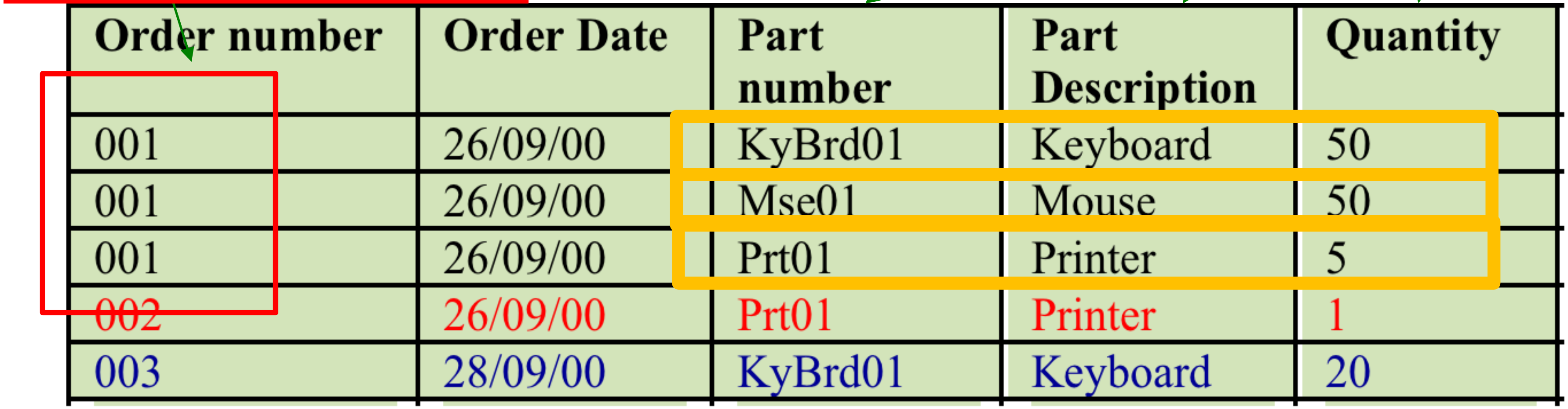
3rd Normal Form (3NF)

1st step - remove Repeating Groups

Each attribute has more than one value (3 values here) for the same instance of the primary key (001)

- A repeating group is a group of attributes which have more than one value for each instance of the primary key
  - order (order number(PK), order date, part number, part description, quantity)

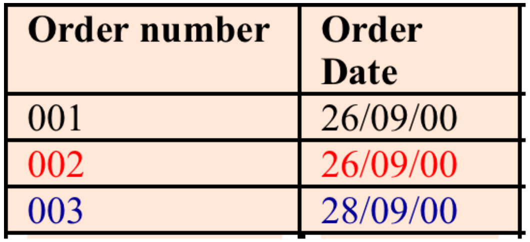
Same instance of the primary key



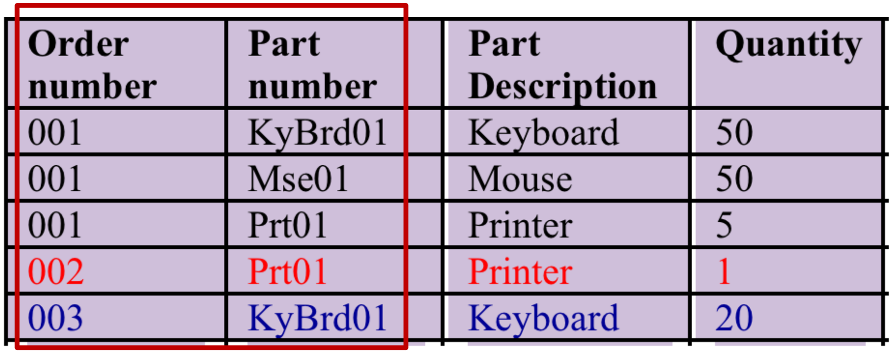
Order number	Order Date	Part number	Part Description	Quantity
001	26/09/00	KyBrd01	Keyboard	50
001	26/09/00	Mse01	Mouse	50
001	26/09/00	Prt01	Printer	5
002	26/09/00	Prt01	Printer	1
003	28/09/00	KyBrd01	Keyboard	20

Removing Repeating Groups

- The attributes in a repeating group are moved to a new table. The **original** **primary** **key** is also added to the new table to link it back to the original table.
- The new tables will have a composite primary key
  - order (order number(PK), order date)
  - order_details (order number(PK, FK), part number(PK), part description, quantity)



Order number	Order Date
001	26/09/00
002	26/09/00
003	28/09/00



Order number	Part number	Part Description	Quantity
001	KyBrd01	Keyboard	50
001	Mse01	Mouse	50
001	Prt01	Printer	5
002	Prt01	Printer	1
003	KyBrd01	Keyboard	20

What is the repeating group in the following table?

Student ID	Student name	Qualification	Date of Graduation
B00098765	John	Cert in Computing	10 Nov 2005
B00098765	John	Degree in Computing	9 Nov 2006
B00098765	John	Hons Degree in Computing	7 Nov 2007
B00098765	John	MSc in Computing	8 Nov 2009
B0002376	Alice	Degree in Digital Media	8 Nov 2009

Recommend a well structured relational model to store the data above:

Give qualification a suitable PK

student(student ID(PK), studentname)

student_award(student ID(PK,FK), QualID(PK),qualification, Date of Graduation)

What is the repeating group in the following table?

ISBN	Book Title	Date	Author
12365458532	Rework	2009	Jason Fried
12365458532	Rework	2009	David Hansson
56733451123	A Patriots History of the United States	2008	Larry Schweikart
56733451123	A Patriots History of the United States	2008	Michael Allen

Recommend a well structured relational model to store the data above:

Book(ISBN(PK), book title, date)

Book__details(ISBN(PK,FK), Author(PK) , author)

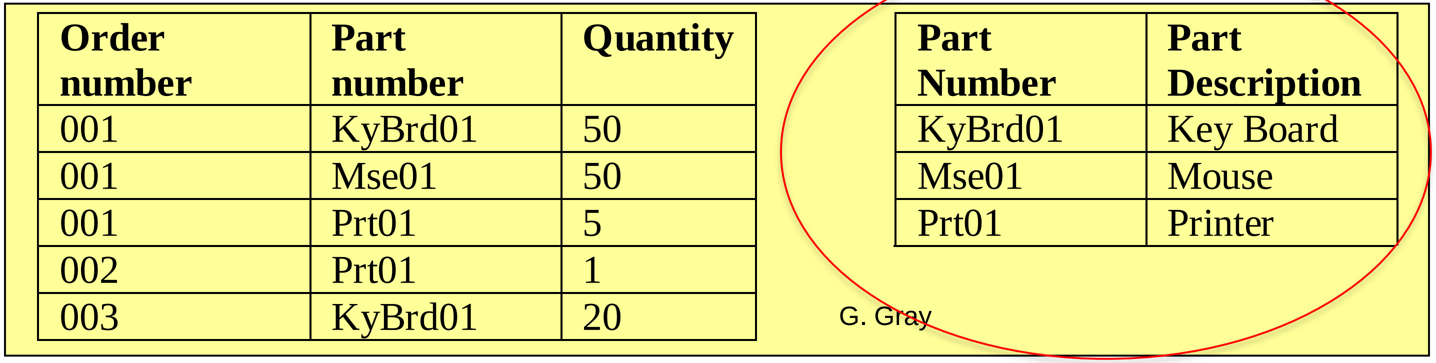
2nd Step - remove partial dependencies

- A partial dependency can **only** occur if you have a composite primary key.
- An attribute is partially dependent on the primary key if it is functionally dependent on only part of the primary key and not the full key.
  - order_details (order number(PK, FK), part number(PK, FK), part description, quantity)

Order number	Part number	Part Description	Quantity
001	KyBrd01	Key Board	50
001	Mse01	Mouse	50
001	Prt01	Printer	5
002	Prt01	Printer	1
003	KyBrd01	Key Board	20

Removing partial dependencies

- Attributes that are partially dependent on the primary key are moved to a new table. The primary key of the new table is the part of the original composite key which the attribute was dependent on. This original key now becomes a primary key and a foreign key
  - order_details (order number(PK, FK), part number(PK, FK), quantity)
  - part (part number(PK), part description)



Order number	Part number	Quantity
001	KyBrd01	50
001	Mse01	50
001	Prt01	5
002	Prt01	1
003	KyBrd01	20

Part Number	Part Description
KyBrd01	Key Board
Mse01	Mouse
Prt01	Printer

Re-Cap

Original Table

Order number	Order Date	Part number	Part description	Quantity
001	26/09/00	KyBrd01	Key Board	50
001	26/09/00	Mse01	Mouse	50
001	26/09/00	Prt01	Printer	5
002	26/09/00	Prt01	Printer	1
003	28/09/00	KyBrd01	Key Board	20

New Tables

Order number	Order Date
001	26/09/00
002	26/09/00
003	28/09/00

Order number	Part number	Quantity
001	KyBrd01	50
001	Mse01	50
001	Prt01	5
002	Prt01	1
003	KyBrd	20

Part Number	Part Description
KyBrd01	Key Board
Mse01	Mouse
Prt01	Printer

Identify the partial dependency in the following table:

Car Reg (PK)	Service ID (PK)	Service Description	Date of Service
03-D-123	Ser1	Full Service	01/03/2010
06-C-5643	Ser1	Full Service	03/03/2010
02-MH-3214	Ser2	Part Service	04/03/2010

Recommend a well structured relational model to store the data above:

car(car reg(pk,fk), serviceID(pk,fk), Date of service)

service(service ID(pk)), service description)

3rd Step - Remove Transitive Dependency

- A transitive dependency is an attribute which is functionally dependent on some **other** attribute that is not the primary key.
  - employee (RSI number(PK) ,name, address, department ID, department name)

RSI number	name	address	department ID	department name
7455122	Gleeson	Dublin 3	D001	Sales
9562214	Burke	Dublin 7	D001	Sales
5412332	Griffin	Dublin 15	D002	Purchasing
4112512	Lucey	Dublin 11	D003	Warehouse

OR

- employee (RSI number(PK) ,name, address, department name)

Removing transitive dependencies

- As for partial dependencies, move the attributes to a new table. Select a primary key for the new table. Add a foreign key to the original table to link to this new table.
  - employee (RSI number(PK), name, address, department ID(FK))
  - department (department ID(PK), department name)

RSI number	name	address	department ID
7455122	Gleeson	Dublin 3	D001
9562214	Burke	Dublin 7	D001
5412332	Griffin	Dublin 15	D002
4112512	Lucey	Dublin 11	D003

department ID	department name
D001	Sales
D002	Purchasing
D003	Warehouse

Identify the transitive dependency in the following table:

MemberID	MemberName	PhoneNumber	Book title
2010GF	Gary Field	01-8223456	Rework
2009SD	Sinead Dempsey	085-1234567	Patriot Games

Recommend a well structured relational model to store the data above:

Identify the transitive dependency in the following table:

ClubCardID	Name	Points	Store Name
3275432	Gary Field	500	Tesco Roselawn
675637	Sinead Dempsey	850	Tesco Clare Hall

Recommend a well structured relational model to store the data above:

Putting it all together – step 1

- Convert the following list of attributes to a set of relations in 3rd normal form (3NF):

Product (product ID(PK), description, quantity in stock, supplier name, supplier address, contact name)

1st NF: Are there any repeating groups?

Does any attribute have more than one value for a given value of the primary key?

Putting it all together – step 2

Product (product ID(PK), description, quantity in stock, supplier name, supplier address, contact name)

2nd NF: Are there partial dependencies?

- Does it have a composite primary key?
- If so, are there attributes that are functionally dependent on just PART of the primary key?

Putting it all together – step 3

- Product (product ID(PK), description, quantity in stock, supplier name, supplier address, contact name)

- 3rdNF – are there any transitive depdencies
  - Are any attributes in the wrong table? i.e. not functionally dependent on PK productID as they do not describe a product.
  - Yes: supplier name, supplier address and contact name describe a supplier rather than a product, and so should be in table with supplierID as the primary key.

Putting it all together – final tables in 3NF

Product (product ID(PK), description, quantity in stock, supplier name, supplier address, contact name)

- becomes

Product (product ID(PK), description, quantity in stock, supplierID(FK))

Supplier (supplierID(PK), supplier name, supplier address, contact name)

Converting to 3rd Normal Form in a nutshell

Marie Brennan

Convert the following list of attributes to a set of relations in 3rd normal form (3NF):

Order Number(PK)	Order Date	Customer Name	Customer Address	Product ID	Product Desc	Quantity Ordered
Order001	21April2010	Dunnes	Dublin 15	P445	Socks	500
Order001	21April2010	Dunnes	Dublin 15	P467	Slippers	250
Order001	21April2010	Dunnes	Dublin 15	P872	Shoes	300
Order002	21April2010	M&S	Dublin 15	P445	Socks	240

Order (Order number(PK), order date, customer name, customerAddress, product id, product description, quantity ordered)

- **Identify repeating groups (groups of attributes that have more than one value for each instance of the primary key)**

Repeating Group

order

Order Number(PK)	Order Date	Customer Name	Customer Address	Product ID	Product Desc	Quantity Ordered
Order001	21April2010	Dunnes	Dublin 15	P445	Socks	500
Order001	21April2010	Dunnes	Dublin 15	P467	Slippers	250
Order001	21April2010	Dunnes	Dublin 15	P872	Shoes	300
Order002	21April2010	M&S	Dublin 15	P445	Socks	240

Order Number(PK)	Product ID	Product Desc	Quantity Ordered
Order001			
Order001			
Order001			
Order002			

Move to a new table, along with the primary key of the original table.

This new table will have a **composite** primary key, the PK from the original table, and a key value for the repeating group:

This means more than one column is defined as part of the primary key.

Original table

order

Order Number(PK)	Order Date	Customer Name	Customer Address
Order001	21April2010	Dunnes	Dublin 15
Order002	21April2010	M&S	Dublin 15

New table

Order_details

Order Number(PK)	Product ID (PK)	Product Desc	Quantity Ordered
Order001	P445	Socks	500
Order001	P467	Slippers	250
Order001	P872	Shoes	300
Order002	P445	Socks	240

Identify **partial dependencies** (an attribute is only dependent on ONE part of the primary key.

- The new table includes product description, which is only functionally dependent on productID, and not order number.

- It should therefore be in a product table ONLY, and not included in the order table.

A new table

Order Number(PK, FK)	Product ID (PK, FK)	Product Desc	Quantity Ordered
Order001	P445	Socks	500
Order001	P467	Slippers	250
Order001	P872	Shoes	300
Order002	P445	Socks	240

New Tables

product

Product ID (PK)	Product Desc
P445	Socks
P467	Slippers
P872	Shoes

order

Order Number(PK)	Order Date	Customer Name	Customer Address
Order001	21April2010	Dunnes	Dublin 15
Order002	21April2010	M&S	Dublin 15

Order_details

Order Number(PK)	Product ID (PK)	Quantity Ordered
Order001	P445	500
Order001	P467	250
Order001	P872	300

Transitive Dependencies

- **Identify transitive dependencies (are there any other attributes not functionally dependent on the primary key?).**

Product ID (PK)	Product Desc
P445	Socks
P467	Slippers
P872	Shoes

Order Number(PK)	Order Date	Customer Name	Customer Address
Order001	21April2010	Dunnes	Dublin 15
Order002	21April2010	M&S	Dublin 15

Order Number(PK)	Product ID (PK)	Quantity Ordered
Order001	P445	500
Order001	P467	250
Order001	P872	300

What do you suggest?

Transitive Dependencies

- Customer name and address are not functionally depdendent on Order Number, and should be in a table where customerID is the primary key.

- These should be moved to a customer table, leaving a foreign key of CustomerID in the orders table to link the order to the customer.

CustomerID (PK)	Customer Name	Customer Address
Cust211	Dunnes	Dublin 15
Cust212	M&S	Dublin 15

All tables are now in 3rd normal form – every attribute is functionally dependent on its primary key.

Order Number(PK)	Order Date	Customer ID(FK)
Order001	21April2010	Cust211
Order002	21April2010	Cust212

Product ID (PK)	Product Desc
P445	Socks
P467	Slippers
P872	Shoes

Order Number(PK)	Product ID (FK)	Quantity Ordered
Order001	P445	500
Order001	P467	250
Order001	P872	300

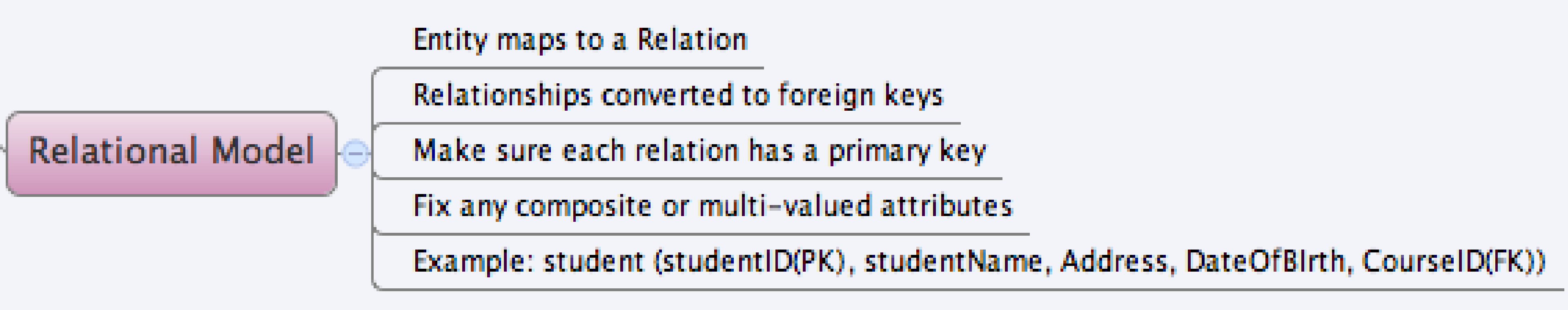
CustomerID (PK)	Customer Name	Customer Address
Cust211	Dunnes	Dublin 15
Cust212	M&S	Dublin 15

Steps in normalization of tables

1. Remove Repeating Groups
2. Remove Functional dependencies
3. Remove Transitive Dependencies

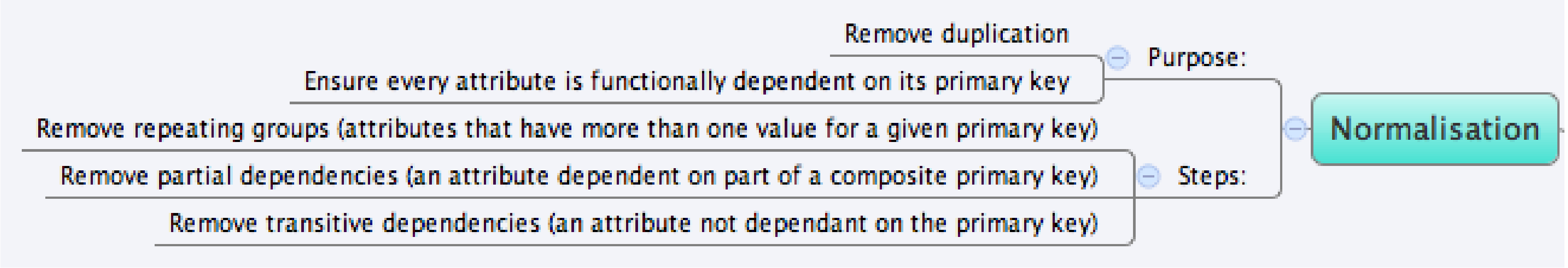
The aim is to produce a set of well structured database tables before you start to create the tables or enter data.

Summary



Relational Model

- Entity maps to a Relation
- Relationships converted to foreign keys
- Make sure each relation has a primary key
- Fix any composite or multi-valued attributes
- Example: student (studentID(PK), studentName, Address, DateOfBIrth, CourseID(FK))



Normalisation

- Purpose:
  - Remove duplication
  - Ensure every attribute is functionally dependent on its primary key
- Steps:
  - Remove repeating groups (attributes that have more than one value for a given primary key)
  - Remove partial dependencies (an attribute dependent on part of a composite primary key)
  - Remove transitive dependencies (an attribute not dependant on the primary key)