Database Fundamentals

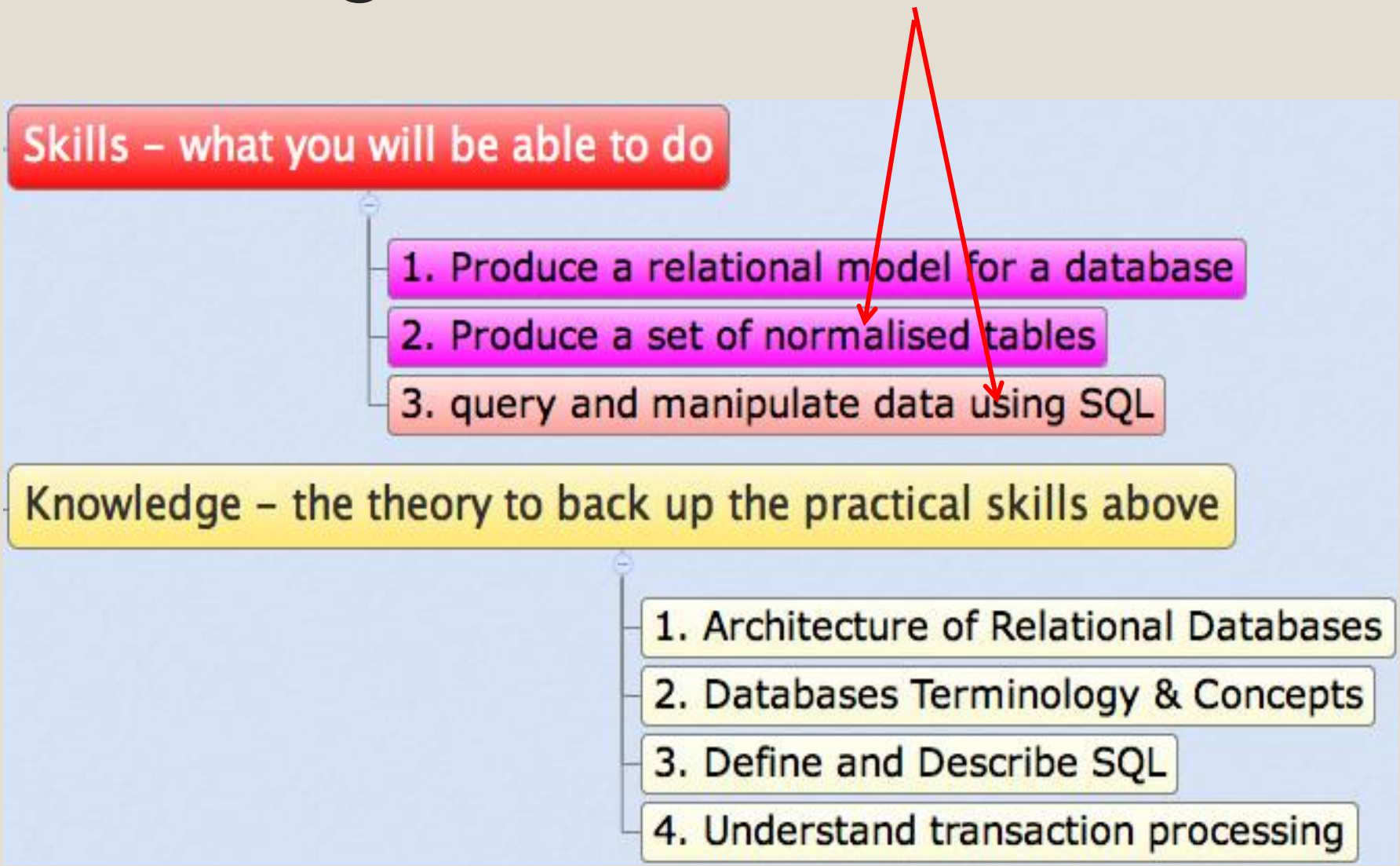
Lecture 9: Data Design part 2

ERDs

Relational model

Normalisation

Learning outcomes



Objective

Work through a number of examples of the steps in database design:

1. Produce and ERD
2. Convert it to a relational model
3. Normalise the relation model to produce a set of tables in 3rd normal form.

RECAP ON LECT 8 . . .

M.Brennan

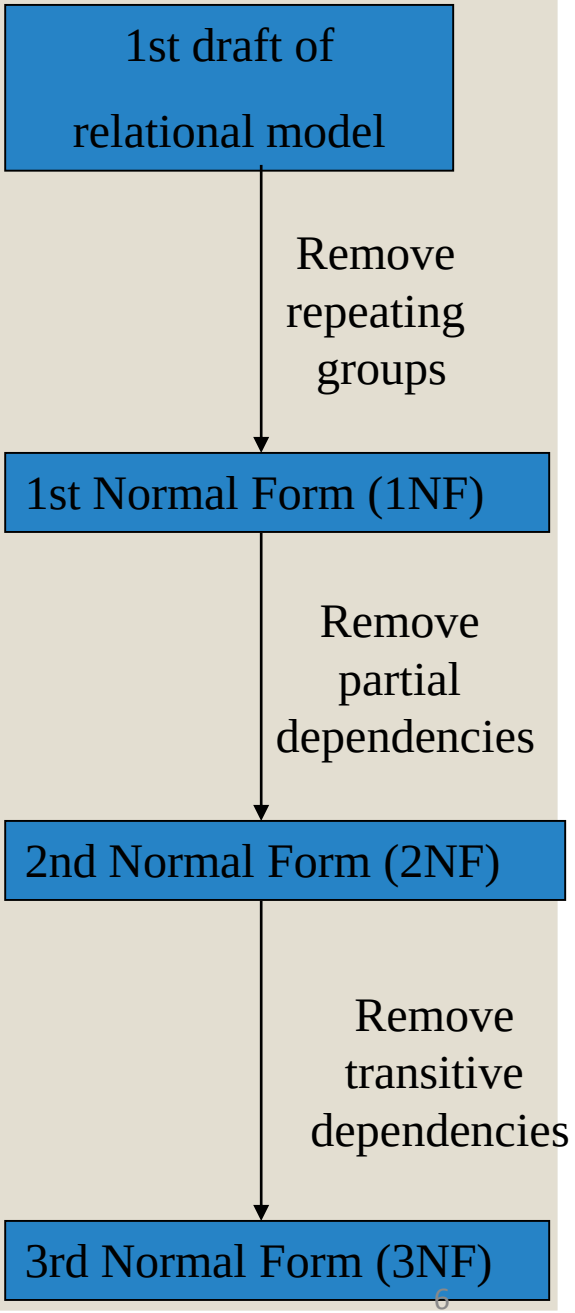
2 Converting from ERD to relational model

Relationship		Decompose	Rules for Foreign Key
One:one	1:1		**Post the Primary Key from one of the entities into the other, as a Foreign Key.** **(It does not matter which one is posted).**
One:many	1:*		**Post the Primary Key from the One side into the Many side as a Foreign Key.**
Many:many	*:*	Create a **Link Entity** between the two original entities.	**Post the Primary Key from each of the original entities into the Link Entity as both Primary Keys and Foreign Keys.**
N-ary e.g. Ternary		Create a **Link Entity** between the three original entities.	**Post the Primary Key from each of the original entities into the Link Entity as both Primary Keys and Foreign Keys.**
Multi-valued attribute		Create a **new entity** to represent the multivalued attribute.	**Post the Primary Key from the One side into the Many side as a Foreign Key.**

3 Normalisation

Purpose:

1. Ensure all attributes are FUNCTIONALLY DEPENDENT on their primary key
2. Ensure NO data is duplicated.



SOME EXAMPLES.

Example 1

- There are a number of sale Representatives covering the sales for each depot. Each depot stores a wide range of products.
- Each depot records the quantity of product in stock.
- The location of each depot is recorded, as is the description of each product

Person?

Product?

Service?

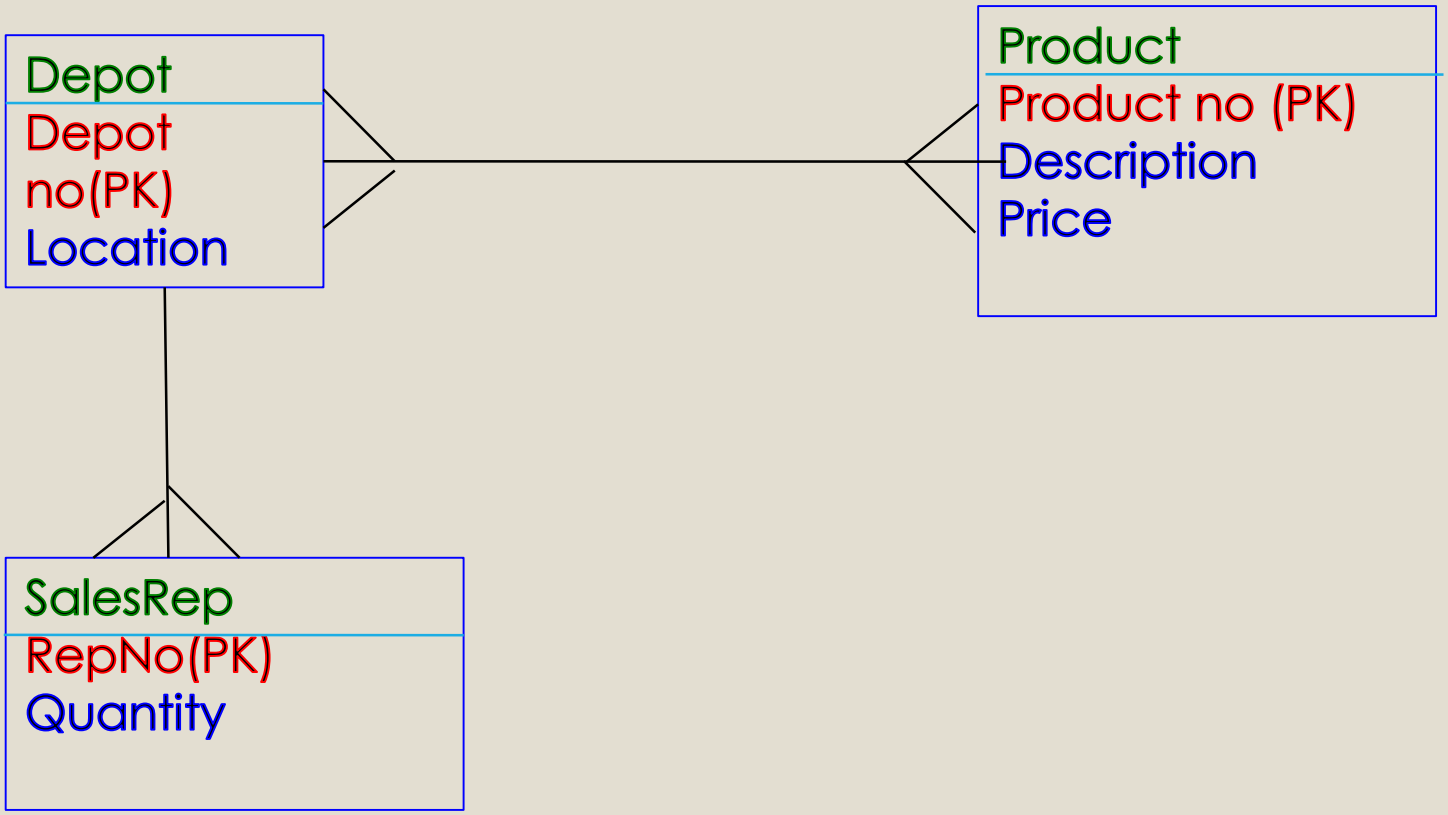
Recording a business event?

There are a number of sale Representatives *working* in each depot. Each depot *stores* a wide range of products. Products can be stored at more than one depot.
Each depot records the quantity of product in stock.
The location of each depot is recorded, as is the description of each product

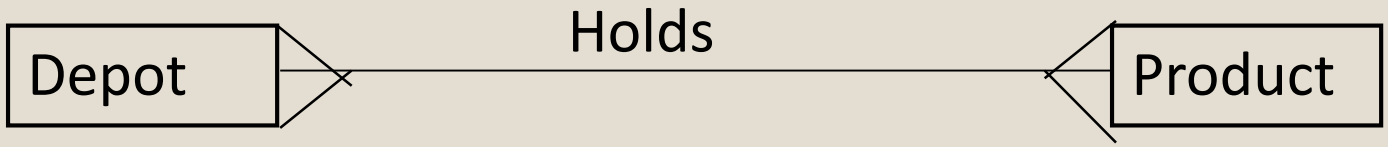
Questions:

- Is the organisation an entity?
- What are the 'straight forward' entities of interest to the organisation (person, service, product)?
- What entities are needed to record transactions?
- What is the relationship between each entity?
- What is the cardinality of the relationship between each
- entity?
- Is each relationship mandatory or optional?

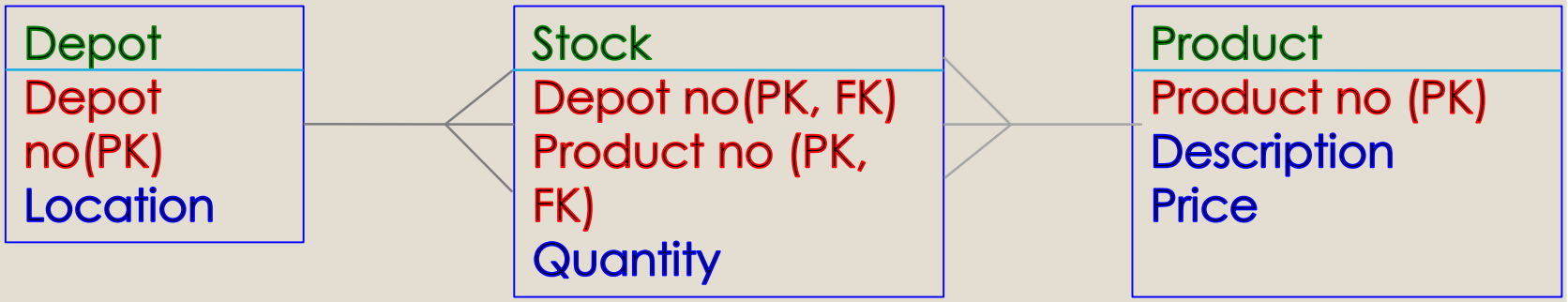
Step 1: ERD


Depot
Depot no(PK)
Location
Product
Product no (PK)
Description
Price
SalesRep
RepNo(PK)
Quantity

Lets look first at the m:n relationship . . . .

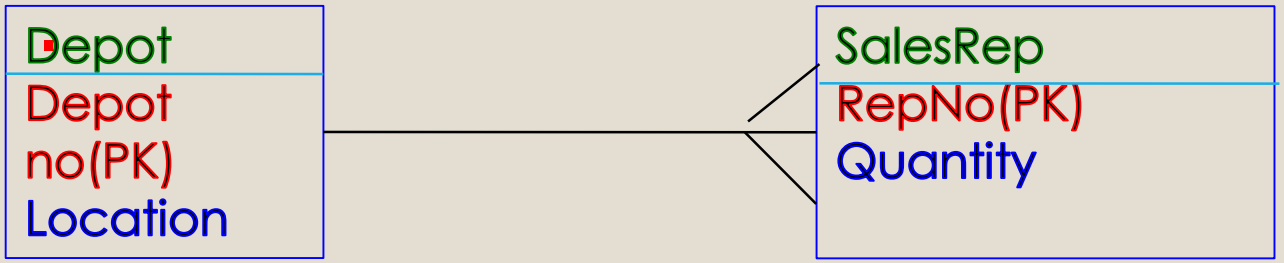


Decompose the n:m relationship
Use a LINK ENTITY

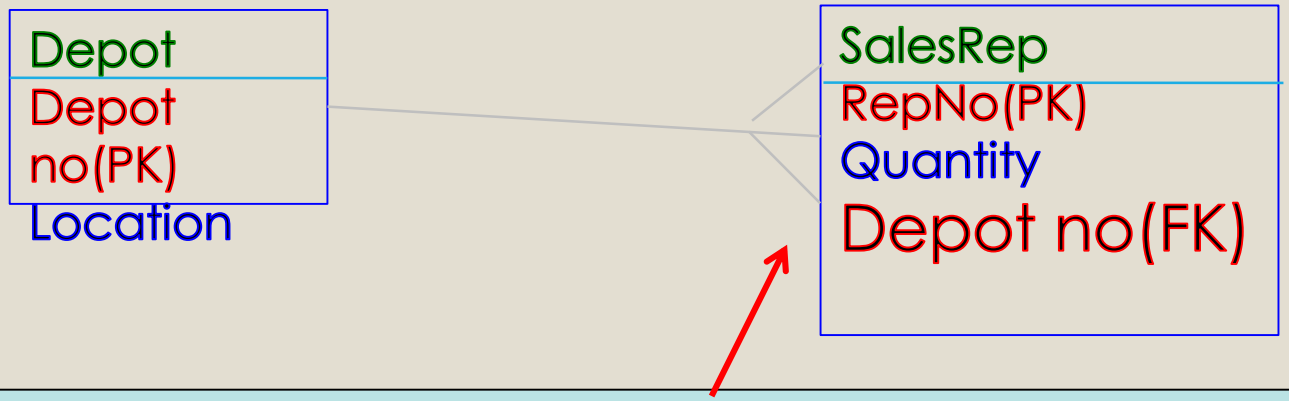


Put the Primary Key from each of the adjoining entities into the Link Entity as a Primary Key and Foreign Key.

Now lets look at the 1:n relationship . . .



Represent the 1:n relationship
Use a FOREIGN KEY



Put the Primary Key from the ONE side as a Foreign Key on the MANY side.

There are no composite attributes, or multi-valued attributes, to the Relational Model is . . .

Depot (Depot no(PK), Location)

Stock (Depot no(PK, FK), Product no (PK, FK) Quantity)

SalesRep (RepNo(PK), Quantity, Depot no(FK))

Product (Product no (PK), Description, Price)

Is each relation in 3rd normal form?

Example 2

- A clinic that performs minor procedures records details on GPs for both out-patients and in-patients.
- For out-patients, it records details on appointments that have been issued and the clinical doctor who attended the patient.
- For in-patients, it records details on procedures that they have undergone and the doctors who performed those prodecures.

Person? Service? Product? Recording a business event?

Hospital Information System

Entities

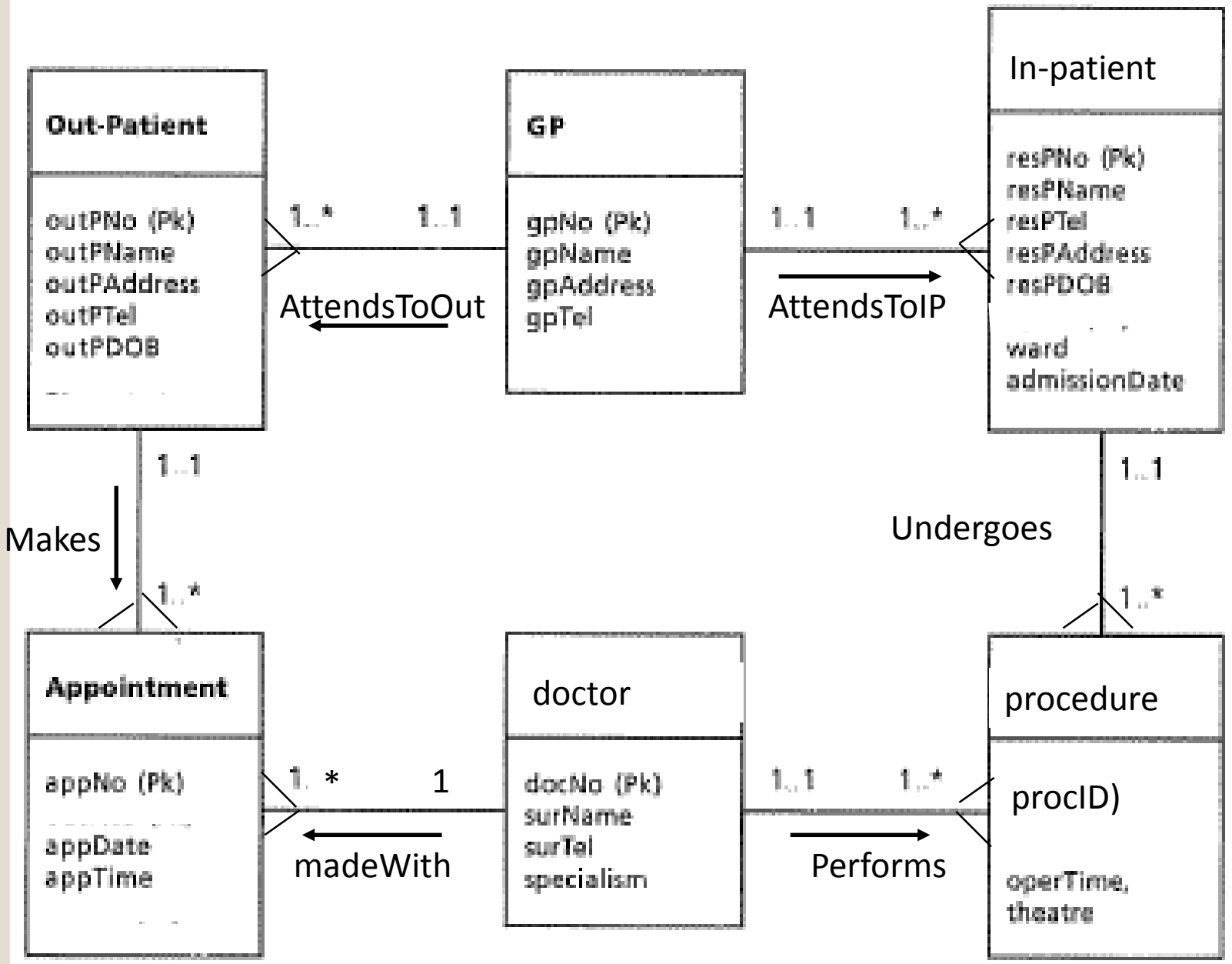
- A clinic records details on GPs for both out-patients and in- patients.
- For in-patients, it records details on appointments that have been issued and the doctor who attended the patient.
- For out-patients, it records details on procedures that they have undergone and the doctors who performed those procedures.

Hospital Information System

Relationships:

- A GP **attends one or more** patients (in-patient or out-patient).
- An out-patient **attends for one or more** appointments.
- An in-patient **undergoes one or more** operations.
- A doctor **attends one or more** appointments.
- A doctor **performs one or more** operations.

ERD, showing relationships both as crows feet AND as foreign keys



So the relationship model is:

- Outpatient(OutPNo(PK), OutPName, OutPAddress, OutPDOB, gpNo(FK))
- GP(gpNo(PK), gpName, gpAddress, gpTel)
- In_Patient(resPNo(PK), resName, resTel, resAddress, resDOB, gpNo(FK), ward, admissionDate)
- Appointment(appNo(PK), OutPNo(FK) , appDate, appTime, docNo(FK))
- Doctor (docNo(PK), surName, surTel, specialisation)
- procedure(procedure ID(PK),resPNo(FK), docNo(FK), oppDate, oppTime, Theatre)

Questions:
Why was the attribute appNo created in the Appointment table?
Are these tables in 3rd Normal Form?

ALL PREVIOUS EXAMPLES PRODUCED RELATIONS IN 3RD NORMAL FORM, SO NORMALIZATION WAS NOT NECESSARY.

THE FOLLOWING EXERCISES FOCUS ON **NORMALISATION**.

Normalisation Exercise – car service

- The following file keeps details of all cars, and the services they have had with the garage. During a service, a number of tests are carried out, and the details are kept of results, and costs. Services are distinguished by the date on which they were carried out.

Service_record(Car_Reg(PK), Service_Date, Owner name, Miles on clock, Full/Basic_Service, Test ID, Test name, Test result, Test's Labour costs, Test's Materials costs)

Steps in normalization

Step 1: Remove repeat groups

Step 2: Remove partial dependencies

Step 3: Remove transitive dependencies

Step 1: Remove repeat groups

A repeating group is a group of attributes which have more than one value for each instance of the primary key

- Service_record(Car_Reg(PK), Service_Date, Owner name, Miles on clock, Full/Basic_Service, Test_ID(FK))
- Test(Car_Reg(PK, FK), Test ID(Pk, FK), Test name, Test result, Test's Labour costs, Test's Materials costs)

Step 2: Remove partial dependencies

- A partial dependency can **only** occur if you have a composite primary key.

Test(Car_Reg(PK, FK), Test ID(Pk, FK), Test name, Test result, Test's Labour costs, Test's Materials costs)

Car(Car_Reg(PK), Owner name)

Service_record(recordID, Car_Reg(FK), Service_Date, Miles on clock, Full/Basic_Service)

Test(Test ID(Pk), Test name, result, costs, materials)

Normal Forms

- A set of conditions on table structure that improves maintenance.
- Normalization removes processing anomalies:
  - Update
  - Inconsistent Data
  - Addition
  - Deletion

Normal Forms

All attributes depend on the key, the whole key and nothing but the key.

1NF Keys and no repeating groups

2NF No partial dependencies

3NF All transitive dependencies

1st Normal Form

- Table has a primary key
- Table has no repeating groups

A multivalued attribute is an attribute that may have several values for one record (e.g. a book in a book table with two authors)

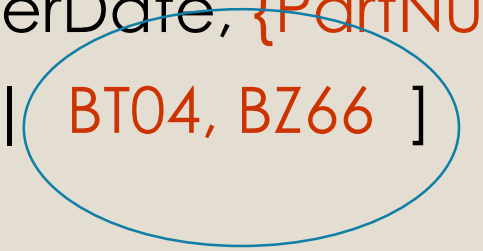
A repeating group is a set of one or more multivalued attributes that are related

Example

- Multivalued attribute:

Orders(OrderNumber, OrderDate, {PartNumber})

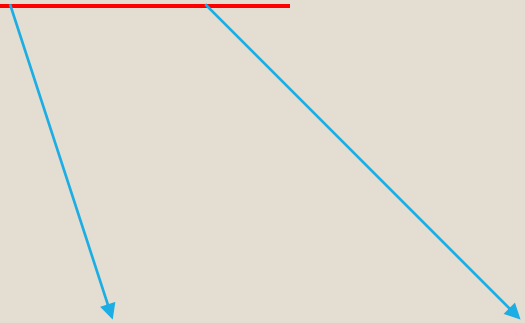
[12491 | 9/02/2001 | BT04, BZ66]



example

- Repeating group:
- Orders(OrderNumber, OrderDate, {PartNumber, NumberOrdered})

- A repeating group is a set of one or more multivalued attributes that are related



- VALUES [12491 | 9/02/2001 | (BT04, 1), (BZ66) 1]

Normalization: 1NF

- Every repeating group becomes a new table with the appropriate foreign key relationships preserved.
- Remove nested repeating groups from the outside in

Order(OrderNumber, OrderDate, {PartNumber, {Supplier}})

Example: 1NF

Breaking down the tables as much as possible to make all attributes completely dependent on the PK

Order(OrderNumber, OrderDate, {PartNumber, {Supplier}})

Order(OrderNumber, OrderDate)

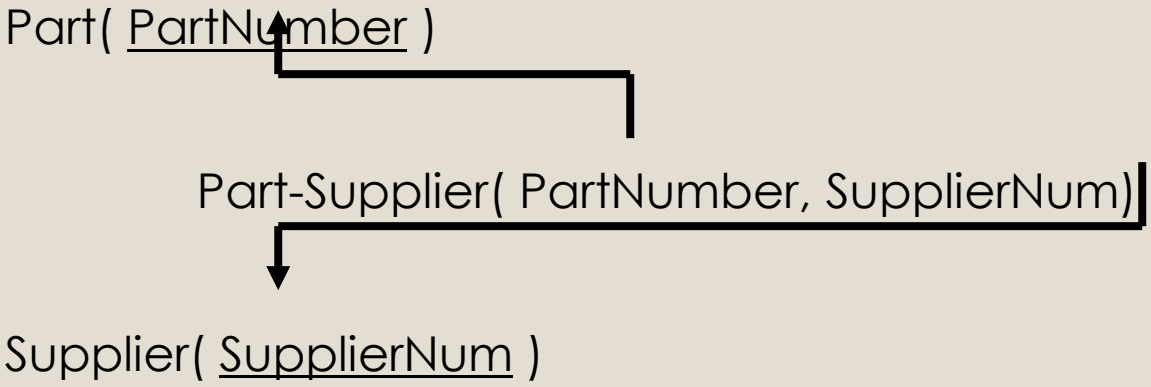
Order-Part(OrderNumber, PartNumber)

Part(PartNumber, {Supplier})

Breaking down the tables as much as possible to make all attributes completely dependent on the PK

Example: 1NF (cont.)

Part(PartNumber, {Supplier})



2nd Normal Form

- No **partial** dependencies

No attribute depends on only some of the attributes of a concatenated key. E.g

Order-Part

[OrderNumber | PartNumber | PartDescription]

Create a new table with PartNumber key.

3rd Normal Form /

Breaking down the tables as much as possible to make all attributes completely dependent on the PK

- 3rd Normal Form: no transitive dependencies

Transitive dependency means that a non-key attribute depends on another non-key attribute(s).

*Book(isbn, title, authorID, **publisher**)*

Book(isbn, title, authorID, pubID(FK))

publisher(pubID(PK), name)

Normalisation

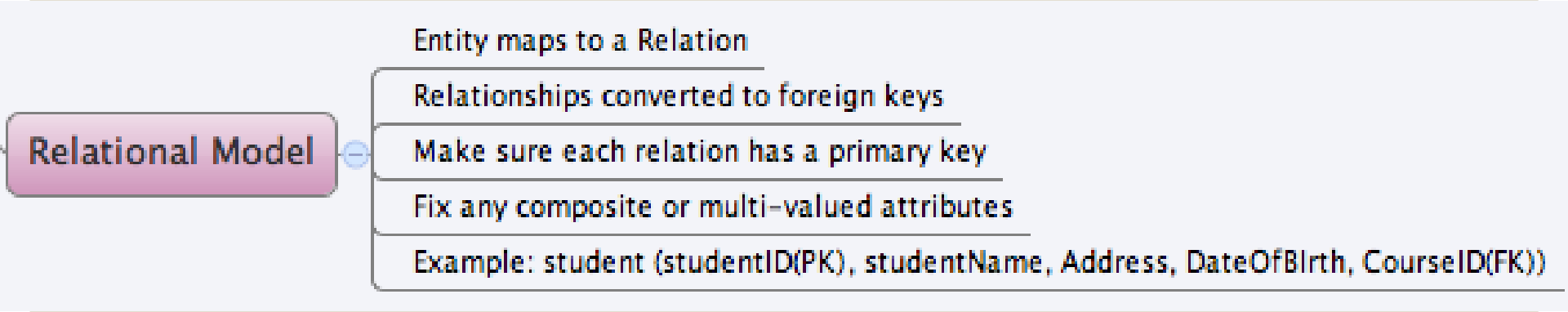
- There are three ways in which an attribute is NOT functionally dependent on the primary key
- Identifying these scenarios is done by following the three steps of Normalisation:
  1. Bring to 1st normal form – remove repeating group
  2. Bring to 2nd normal form – remove partial dependencies,
  3. Bring to 3rd normal form – remove transitive dependencies

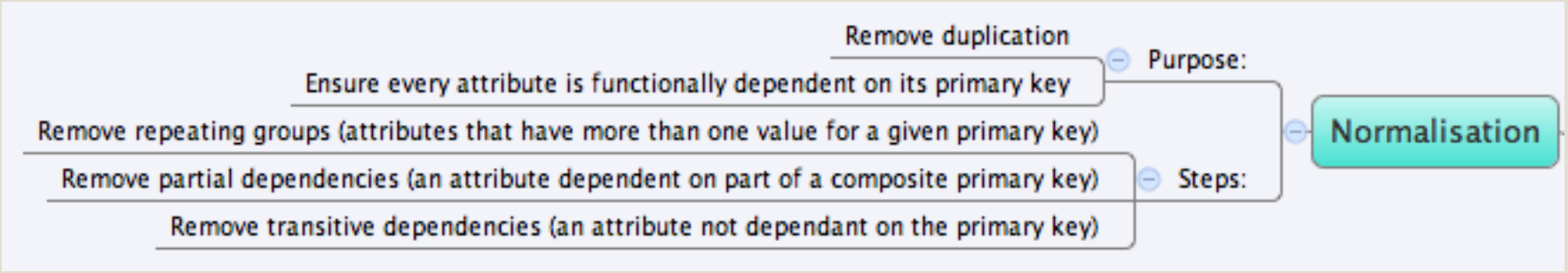
- Once in third normal form (3NF), the tables are well-structured
- This is what you need to concentrate on!!

PAST EXAM QUESTIONS ON DATABASE DESIGN . . .

See handouts

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)