Database Development: More RDB Modelling (normalization)

Lecturer: Dr. Richard Holden

Topics this week

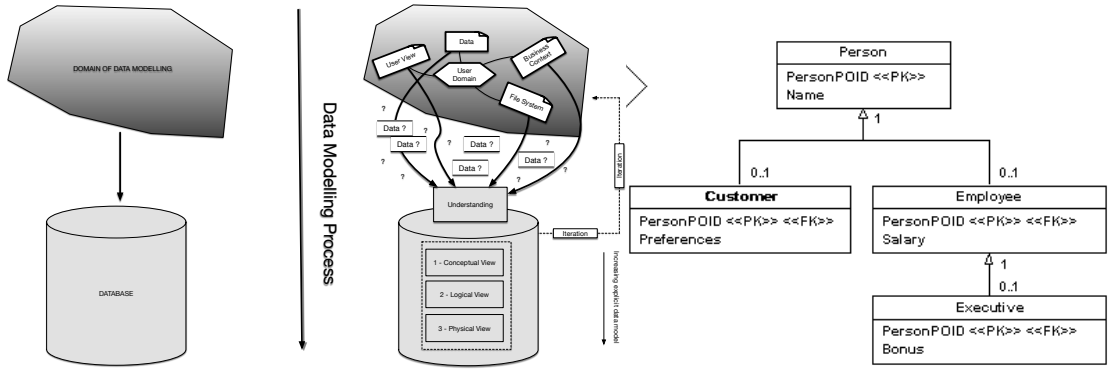
- **Top-down v bottom-up modelling**
  - ERD v normalization
- **Normal forms**
  - **1NF**
  - **2NF**
  - **3NF**

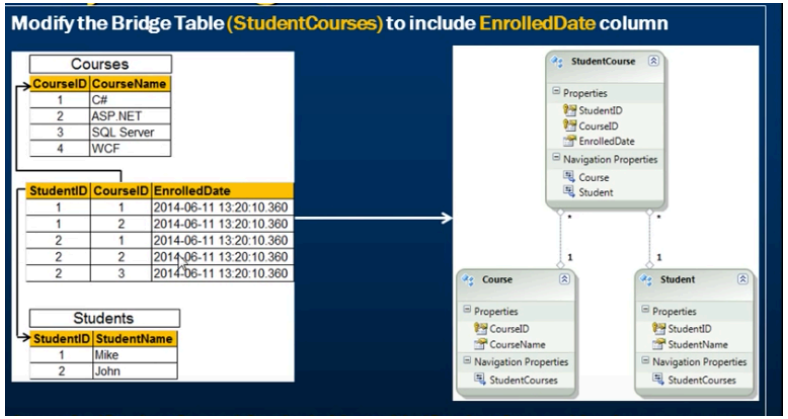
Bottom-Up Modelling

1. Start with the data itself rather than an idea of how you want things organised.
2. Simple, scenario (one table approach)… 0NF - > 1NF

Normalization in relational databases

- Relations are tables, right?
  - You learnt this already
- Problems related to a clear purpose
  - Representing entities in a supply chain etc.
  - A hotel booking scenario.
- Its easiest to write textbooks, based on such scenarios too.

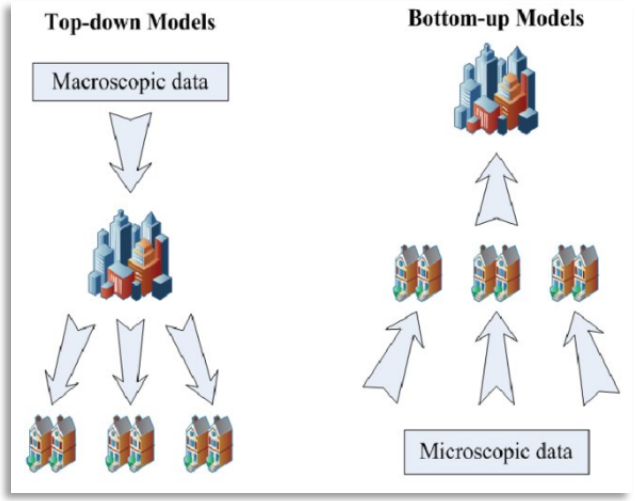




Normalization in relational databases

- Yet, arriving at well-defined relations from existing data isn't always so straight forwards.

- Often data is collected without any well-defined purposes
  - Related to what should be recorded
  - Related to how it should be used

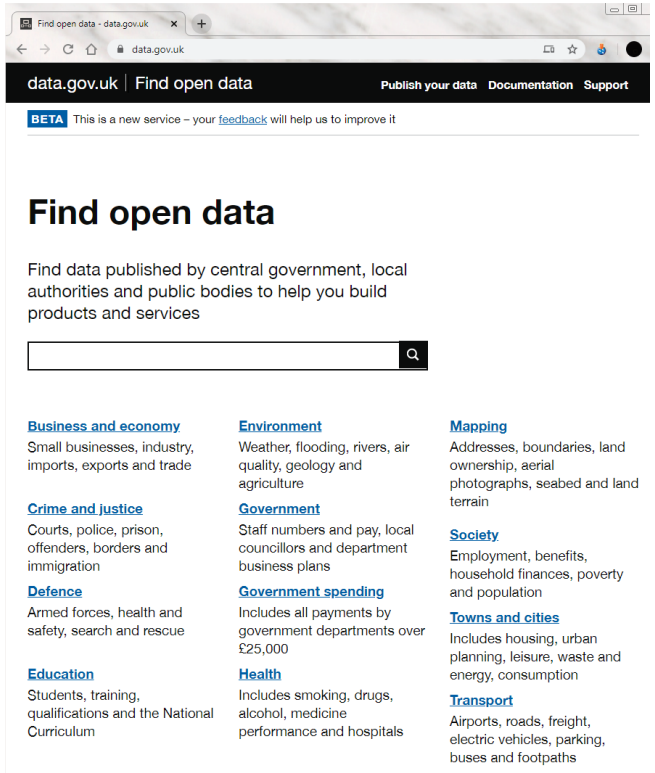


- Relations are tabular, but 'tables' re not relations.
- There are tables and there are *tables.*

Normalisation in relational databases

- Yet, arriving at well-defined relations from existing data isn't always so straight forwards.

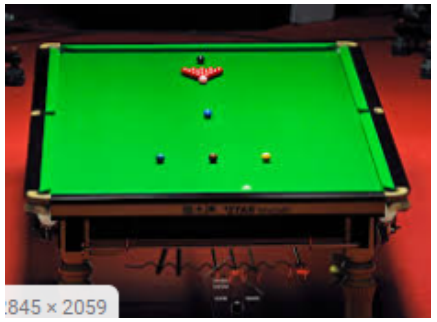
- Often data is collected without any very well-defined purposes
  - Related to what should be recorded
  - Related to how it should be used
  - How it should be stored etc



https://data.gov.uk/

Simple scenario: club membership

- Take a situation where two or more people are members of different (possibly overlapping) sports clubs.
  - Tracy attends a dance and swimming club
  - Richard attends a snooker club



- We can represent this in a tabular form……..

Simple scenario: club membership

- **First table**

- **This isn't so good.**
  - The number of clubs is unspecified
- **Therefore, the form of the table is unknown, given N data entries.**
- **Relational databases wont work with this structure**
  - Each entry inside a column must have a single datum inside it.
- Non-repeating groups is the aim
  - In this example the group Club is repeated against Tracy.

Attends	
Name	Club
Tracy	Swimming, dance
Richard	Snooker

"Unnormalized" relation

Simple scenario: club membership

- **Second table**

- **Exactly the same information in this as the first table**
  - The number of clubs is not repeated
- **This is the property we want!**
- **1NF: First Normal Form**
- **1NF defined:**
  - A relation is in 1NF if no entry has > 1 values
  - i.e non-repeating groups.

Attends	
Name	Club
Tracy	Swimming
Tracy	Dance
Richard	Snooker

1NF is the first requirement in 'designing' our database from the bottom-up

Problem: duplication in 1NF

- **Leisure chain owns clubs**
- **Chain-level membership type (Gold, Silver)**
- **Adds detail of diff clubs (type, clubId=cid, cost, date joined)**
  - **To club attendees table**

ClubAttendees

Id	Name	club	type	cid	cost	joined
1	Tracy	Swim	Gold	6	£10	3-2-16
2	Richard	Snooker	Silver	3	£20	5-6-17
1	Tracy	Dance	Gold	7	£15	1-1-2

Problem: duplication in 1NF

- There are no duplicate rows
- But there is duplicate information
- Update anomaly
  - Suppose Tracy downgrades to Silver
  - All records must be changed otherwise inconsistent data
  - Imagine millions of rows!
- So, the data is more structured than 0NF, but not structured enough (for Relational database purposes)!

ClubAttendees

Id	Name	club	type	cid	cost	joined
1	Tracy	Swim	Gold	6	£10	3-2-16
2	Richard	Snooker	Silver	3	£20	5-6-17
1	Tracy	Dance	Silver?	7	£15	1-1-2

Anomalies: insertion/deletion in 1NF

- **Suppose a new Gold+ type is introduced**
- **Violates entity integrity**
  - **maintenance of primary key**
- **A club attendee decides to not renew to any clubs one month, remaining a (Gold, Silver), but remains member of the leisure chain annually.**
  - **How to maintain this information?**
  - **Delete all instances?**
  - **Deletion anomalies!**

ClubAttendees

Id	Name	club	type	cid	cost	joined
1	Tracy	Swim	Gold	6	£10	3-2-16
2	Richard	Snooker	Silver	3	£20	5-6-17
NULL	NULL	NULL	Gold+	NULL	£40	NULL

Normalization scenario

Dental Company

From 0NF -> 3NF

Dental scenario

- A Dental company employs a number of Dental Professionals in several locations – each dental professional works in a single location.
- A dental professional has a unique identifier, a first name and surname and a specialism (dentist, hygienist etc.).
- Patients register with identified dental professionals with the date of registration noted.
  - e.g. a dentist, a hygienist, an orthodontist etc.
- A patient has a patient number, first name, last name and address.
- A location has a location code and address.

Dental company: UNF/0NF

Professional(ProfessionalId, FirstName, Surname, Specialism,
LocationCode, LocationAddress,
(PatientNo, PatientFirstName,PatientSurname,
PatientAddress, DateRegistered)
)

- Patients register with dental professional
- Note the repeating group (same patient info per professional ?)
- ProfessionalId has been identified as a relevant key

Dental company: UNF/0NF

Professional(ProfessionalId, FirstName, Surname, Specialism,
LocationCode, LocationAddress,
(PatientNo, PatientFirstName,PatientSurname,
PatientAddress, DateRegistered)
)

- Operations to perform on red table to get decomposed amber tables on next slide
  1. Remove repeating group to new relation
  2. Copy the existing key to new relation
  3. Examine relation to determine its key

Dental company: 0NF -> 1NF

Professional(ProfessionalId, FirstName, Surname, Specialism,
LocationCode, LocationAddress,
(PatientNo, PatientFirstName,PatientSurname,
PatientAddress, DateRegistered)
)

0NF

Professional(ProfessionalId, FirstName, Surname, Specialism,
LocationCode, LocationAddress)

1NF

PatientRegistration(ProfessionalId, PatientNo, PatientFirstName,
PatientSurname, PatientAddress, DateRegistered)

Dental company: 1NF -> 2NF

Professional(<u>ProfessionalId</u>, FirstName, Surname, Specialism, LocationCode, LocationAddress)

PatientRegistration(<u>ProfessionalId</u>, <u>PatientNo</u>, PatientFirstName, PatientSurname, PatientAddress, DateRegistered)

- Operations to perform on amber tables to get to further decomposed yellow tables on following slide
  1. Identify compound key tables and examine to simplify
     a) Examine each non-key item to see if it depends on the whole key or only part of the key
     b) Move the partially dependent non-key items into a new relation along with a copy of the partial key

Dental company: 1NF -> 2NF

Professional(<u>ProfessionalId</u>, FirstName, Surname, Specialism, LocationCode, LocationAddress)

1NF

PatientRegistration(<u>ProfessionalId</u>, <u>PatientNo</u>, PatientFirstName, PatientSurname, PatientAddress, DateRegistered)

Professional(<u>ProfessionalId</u>, FirstName, Surname, Specialism, LocationCode, LocationAddress)

2NF

PatientRegistration(<u>ProfessionalId</u>, <u>PatientNo</u>, DateRegistered)

Patient(<u>PatientNo</u>, PatientFirstName, PatientSurname, PatientAddress)

Dental company: 2NF-> 3NF

Professional(<u>ProfessionalId</u>, FirstName, Surname, Specialism, LocationCode, LocationAddress)

2NF

PatientRegistration(<u>ProfessionalId</u>, <u>PatientNo</u>, DateRegistered)

Patient(<u>PatientNo</u>, PatientFirstName, PatientSurname, PatientAddress)

- Operations to perform on yellow tables to get to further decomposed green tables
  1. Consider dependencies among non-key items
  2. Move non-key items that depend on other non-key items to new relation
  3. Identify a key for the new relation
  4. Remove any items that can be calculated from existing data (redundancy)

Dental company: 2NF-> 3NF

Professional(<u>ProfessionalId</u>, FirstName, Surname, Specialism, LocationCode, LocationAddress)

2NF

PatientRegistration(<u>ProfessionalId</u>, <u>PatientNo</u>, DateRegistered)

Patient(<u>PatientNo</u>, PatientFirstName, PatientSurname, PatientAddress)

Professional(<u>ProfessionalId</u>, FirstName, Surname, Specialism, LocationCode)

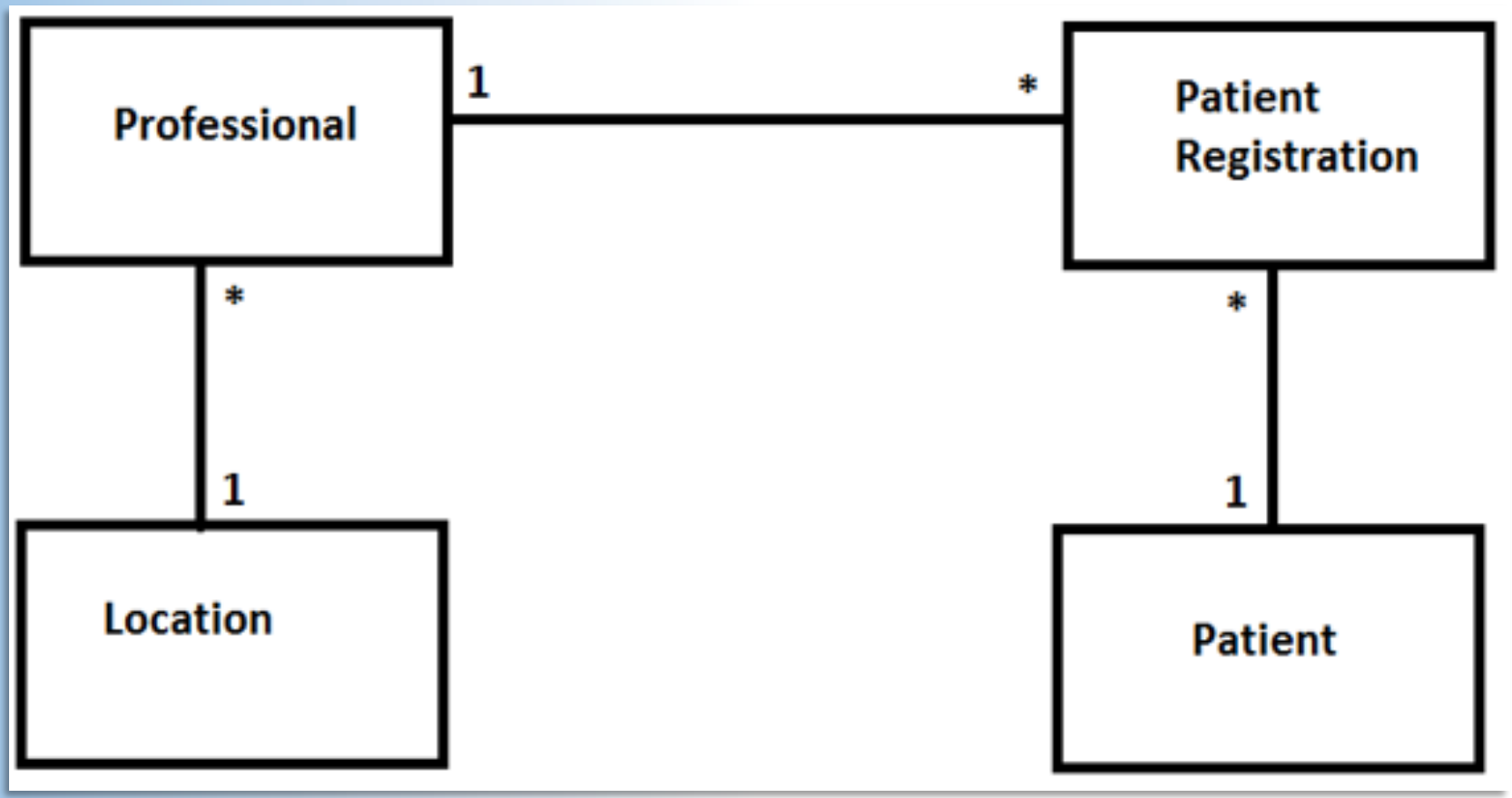
PatientRegistration(<u>ProfessionalId</u>, <u>PatientNo</u>, DateRegistered)

3NF

Patient(<u>PatientNo</u>, PatientFirstName, PatientSurname, PatientAddress)

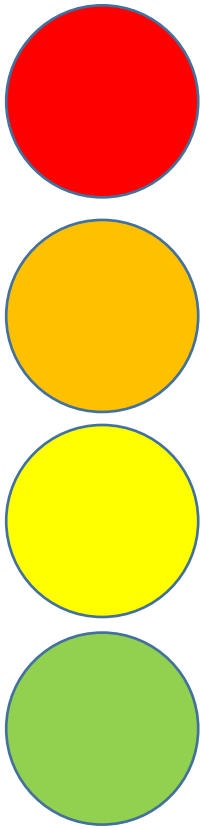
Location(<u>LocationCode</u>, LocationAddress)

Dental company model



Normalisation in relational databases

- For more unusual relations higher-level normal forms exist but…
- Often with 3NF your’re good to go



0

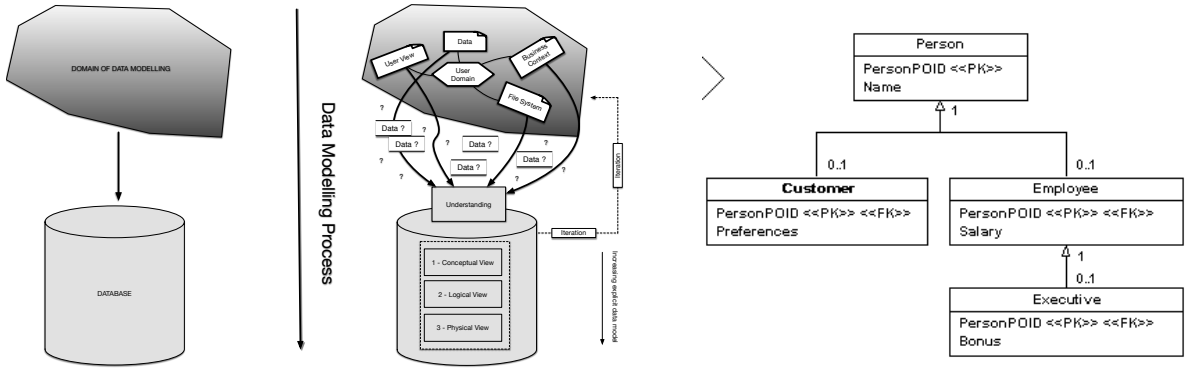
1

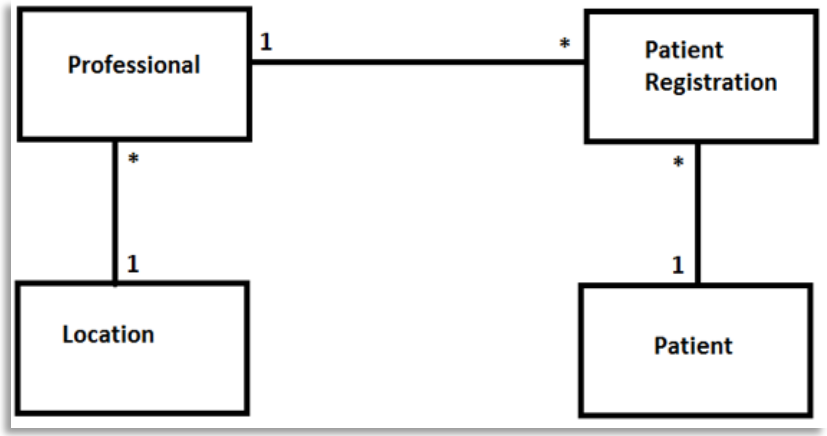
2

3

Dental company model

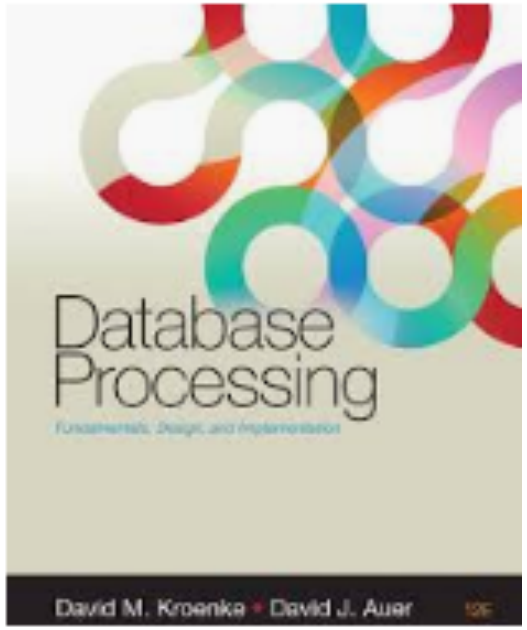
- Note what we did
- Start with data
- Word towards a model, from the bottom up
- So, different to top-down approach





Dental company model

- Whichever approach is used:
- “I swear to construct my tables so that all non-key columns are dependent on the **key, the whole key and nothing but the key**. So help me Codd”
- ….Functional dependency is a key idea behind the methods that drive normalisation from 0NF to 3NF



Functional dependency

- If a relation has two attributes A & B
- B is said to be **functionally dependent** on A if each value of A is associated with exactly one value of B at any instant in time
  - A is said to *determine* B while B *depends* on A
  - A B
- The value of attribute A determines the value of attribute B
  - A is called the **determinant**
- Both A & B can be single attributes or combinations of attributes
  - (A, C) (B, D, E)
    - (A, C) is said to be a **composite determinant**