Advanced Data Management (CMM524)

Solution/Discussion to Laboratory #4: Normalisation

1. Aims

- To design a database that conforms to the third normal form (3NF).

2. Outcomes

In completing this exercise, you should be able to:

- Identify functional dependency among attributes in a table.
- Identify full and partial functional dependency.
- Identify transitive functional dependency.
- Apply normalisation techniques to a given relational database design up to 3NF.

3. The Problem Domain

In this domain, we are storing inspection information of properties of an estate agent.

- A property has a unique property number and an address.
- From time to time, staff have to inspect a property and record their findings as comment. A staff will take a company car to do the visit.
- Each staff has a unique staff number and a name. The name may not be unique.
- Each company car has a unique registration number.
- We need to know the date and time an inspection is done, the staff who did it, the company car used, and comments on the inspection.
- There may be multiple visits to the same property on the same date but the company makes sure that only one inspection is done to the same property at the same time.

3.1. The Original Table

The original table design is as followed:

propertyNo	pAddress	iDate	iTime	comments	staffNo	sName	carReg
PG4	6 Lawrence Street, Glasgow	18/10/2016	10:00	Need to replace crockery.	SG36	Ann Beech	M231JGR
		22/04/2017	09:00	In good order.	SG14	David Ford	M533HDR
		01/10/2017	12:00	Damp rot in bathroom.	SG14	David Ford	N721HFR
PG16	5 Novar Drive, Glasgow	22/04/2017	13:00	Replace living room carpet.	SG14	David Ford	M533HDR
		24/10/2017	14:00	Good condition.	SG36	Ann Beech	N721HFR

Notes:

- **DO NOT blindly copy this example to your coursework.**

- This table is an extreme case of poor design. We use it to illustrate different design flaws and how normalisation can fix the issues.
- In most cases, if you have done some sensible ER modelling, you should not end up with 1 single big table but multiple smaller tables.
- Your normalisation process should start from these multiple smaller tables rather 1 single big table.

- This table is not in 1NF.
  - How can you tell?
    - There are non-atomic attributes (i.e. have multiple values). This is not allowed in 1NF.

3.2. First Normal Form & Primary Key

The table is then revised:

propertyNo	pAddress	iDate	iTime	comments	staffNo	sName	carReg
PG4	6 Lawrence Street, Glasgow	18/10/2016	10:00	Need to replace crockery.	SG36	Ann Beech	M231JGR
PG4	6 Lawrence Street, Glasgow	22/04/2017	09:00	In good order.	SG14	David Ford	M533HDR
PG4	6 Lawrence Street, Glasgow	01/10/2017	12:00	Damp rot in bathroom.	SG14	David Ford	N721HFR
PG16	5 Novar Drive, Glasgow	22/04/2017	13:00	Replace living room carpet.	SG14	David Ford	M533HDR
PG16	5 Novar Drive, Glasgow	24/10/2017	14:00	Good condition.	SG36	Ann Beech	N721HFR

- Is this table in 1NF? How can you tell?
  - The table is now in 1NF as all attributes are atomic (i.e. has 1 value only).
- Choose a primary key for this table.
  - Note: The primary key can be a composite key comprises multiple attributes. It must be able to uniquely identify a row in the table.
    - There are different candidate keys for this table. As far as a combination of attributes can uniquely identify a row, it can be used as a key. However, it is good to choose a primary key which makes sense in the domain.
    - There are multiple candidates for the primary key:
      - The combination of `(propertyNo, iDate, iTime)` is enough to uniquely identify an inspection.
        - Note that just `(propertyNo, iDate)` is not enough as there may be different inspections done to the same property on the same date but different times.
      - Another candidate key is `(staffNo, iDate, iTime)` as a staff can only be at a single property on a certain date and at a certain time.

        - Similarly, (carReg, iDate, iTime) is also a candidate key as a car can only be at a single property on a certain day and at a certain time.
    - A good primary key is (propertyNo, iDate, iTime). if our focus is on properties, rather than car or staff.

3.3. Second Normal Form

- Identify all functional dependency in the table. For example:

  propertyNo → pAddress

  - The sample data DO NOT show all possible cases. You should use your understanding of the domain to identify the functional dependencies.
  - List all function dependencies found using the above notation.
  - Assuming that you use (propertyNo, iDate, iTime) as the primary key, the non-prime attributes will be pAddress, comments, staffNo, carReg, sName.
  - As (propertyNo, iDate, iTime) is the primary key, it identifies a unique row in the table. So all each non-prime attribute is functionally dependent on the primary key (but may not be "***fully*** functionally dependent" until check each of them is not dependent on a subset of the primary key):

    - {propertyNo, iDate, iTime) → pAddress
    - {propertyNo, iDate, iTime} → comments
    - {propertyNo, iDate, iTime} → staffNo
    - {propertyNo, iDate, iTime} → carReg
    - {propertyNo, iDate, iTime) → sName

  - If we look at each non-prime attribute, we also get the following functional dependencies:

    - propertyNo → pAddress
    - staffNo → sName

Note: In identifying dependencies among the attributes, DO NOT just look at the sample data.

- The same data are here to give you an idea of what we are storing. However, in the future there will be more data stored in the table(s).
- Uniqueness in the sample data does not guarantee uniqueness in future data.
- You should use your understanding of the domain to see if a dependency is there, and makes sense.
  - e.g. In the sample data, once you know propertyNo & iDate, you can find a single row with no ambiguity.
    - However, in the actual domain, just knowing propertyNo and iDate is not enough to uniquely identify an inspection as you can have multiple inspections to the same property done on the same date but at different times.
    - It just happened that the sample data do not contain such an example.

- List all full functional dependencies on the primary key.
  - These are full functional dependencies on the primary key:
    - {propertyNo, iDate, iTime} → comments
    - {propertyNo, iDate, iTime} → staffNo
    - {propertyNo, iDate, iTime} → carReg
    - {propertyNo, iDate, iTime} → sName
- List all partial functional dependency on the primary key.
  - There are partial functional dependency on the primary key:
    - propertyNo → pAddress
  - Note that the following is not a partial dependency on the primary key {propertyNo, iDate, iTime} as staffNo is not part of the key:
    - staffNo → sName
    - As sName is still fully functionality dependent on the primary key, the above does not break 2NF.
- Is the table in 2NF? How can you tell?
  - It is not in 2NF as pAddress, as a non-prime attribute, is partially functionally dependent on the primary key.
- Using the functional dependency identified, transform this table into 2NF.
  - The technique is to take attributes involved in the partial functional dependency to another table.
    - Note: sName is indirectly, fully functionally dependent on {propertyNo, iDate, iTime} via staffNo. So it is also fully functionally dependent on the primary key. It should stay in the table when you perform the 2NF transformation.

propertyNo	iDate	iTime	comments	staffNo	sName	carReg
PG4	18/10/2016	10:00	Need to replace crockery.	SG36	Ann Beech	M231JGR
PG4	22/04/2017	09:00	In good order.	SG14	David Ford	M533HDR
PG4	01/10/2017	12:00	Damp rot in bathroom.	SG14	David Ford	N721HFR
PG16	22/04/2017	13:00	Replace living room carpet.	SG14	David Ford	M533HDR
PG16	24/10/2017	14:00	Good condition.	SG36	Ann Beech	N721HFR

propertyNo	pAddress
PG4	6 Lawrence Street, Glasgow
PG16	5 Novar Drive, Glasgow

- Update the primary keys in all tables if necessary.
  - We have a new table (property?). A good primary key for this table is propertyNo.

3.4. Third normal Form

- Identify and list all any transitive functional dependencies in your tables.
  - In the first table:

        - {propertyNo, iDate, iTime} → staffNo → sName
    - The second table has propertyNo as the primary key. No transitive functional dependency exist.
- Is your table now in 3NF? How can you tell?
    - The table is not in 3NF as there is a transitive functional dependency.
- Using the functional dependency identified, further transform this table into 3NF.
    - Again, the technique is to take attributes which are transitive dependent on the primary key and move them into a new table:

propertyNo	iDate	iTime	comments	staffNo	carReg
PG4	18/10/2016	10:00	Need to replace crockery.	SG36	M231JGR
PG4	22/04/2017	09:00	In good order.	SG14	M533HDR
PG4	01/10/2017	12:00	Damp rot in bathroom.	SG14	N721HFR
PG16	22/04/2017	13:00	Replace living room carpet.	SG14	M533HDR
PG16	24/10/2017	14:00	Good condition.	SG36	N721HFR

staffNo	sName
SG36	Ann Beech
SG14	David Ford

propertyNo	pAddress
PG4	6 Lawrence Street, Glasgow
PG16	5 Novar Drive, Glasgow

If you choose (staffNo, iDate, iTime) as the primary key, non prime attributes will be: propertyNo, pAddress, comments, staffNo, sName, carReg

The primary key (staffNo, iDate, iTime) uniquely identifies a row. So each non-prime attribute is functionally dependent on the primary key (but may not be "***fully*** functional dependent"):

- {staffNo, iDate, iTime} → propertyNo
- {staffNo, iDate, iTime) → pAddress
- {staffNo, iDate, iTime} → comments
- {staffNo, iDate, iTime} → sName
- {staffNo, iDate, iTime} → carReg

By systematically looking at each non-prime attribute, we also get the following functional dependency:

- propertyNo → pAddress
- staffNo → sName

The following is a partial functional dependency on the primary key, and it breaks 2NF as `sName` is not fully functionally dependent on the primary key:

- `staffNo → sName`

Note that the following is not breaking 2NF as `propertyNo` is not part of the primary key. The non-prime attribute `pAddress` is still fully functionally dependent on the primary key `(staffNo, iDate, iTime)`:

- `propertyNo → pAddress`

Removing the partial functional dependencies gives the following tables:

propertyNo	iDate	iTime	comments	staffNo	sName	carReg
PG4	18/10/2016	10:00	Need to replace crockery.	SG36	Ann Beech	M231JGR
PG4	22/04/2017	09:00	In good order.	SG14	David Ford	M533HDR
PG4	01/10/2017	12:00	Damp rot in bathroom.	SG14	David Ford	N721HFR
PG16	22/04/2017	13:00	Replace living room carpet.	SG14	David Ford	M533HDR
PG16	24/10/2017	14:00	Good condition.	SG36	Ann Beech	N721HFR

propertyNo	pAddress
PG4	6 Lawrence Street, Glasgow
PG16	5 Novar Drive, Glasgow

The tables are now in 2NF.

To convert these to 3NF, we further remove the transitive dependency identified earlier:

- `{staffNo, iDate, iTime} → staffNo → sName`

propertyNo	iDate	iTime	comments	staffNo	carReg
PG4	18/10/2016	10:00	Need to replace crockery.	SG36	M231JGR
PG4	22/04/2017	09:00	In good order.	SG14	M533HDR
PG4	01/10/2017	12:00	Damp rot in bathroom.	SG14	N721HFR
PG16	22/04/2017	13:00	Replace living room carpet.	SG14	M533HDR
PG16	24/10/2017	14:00	Good condition.	SG36	N721HFR

staffNo	sName
SG36	Ann Beech
SG14	David Ford

propertyNo	pAddress
PG4	6 Lawrence Street, Glasgow
PG16	5 Novar Drive, Glasgow

If you use `(carReg, iDate, iTime)` as the primary key, the normalisation process will be similar.

4. The Patient Record Domain

- In a hospital, a patient is admitted to a ward on the admission date. There will be a discharge date when the patient leaves.
- The patient will be give a bed on admission.
- Throughout his/her lifetime, a patient may be admitted several times (and potentially to different wards).
- Each patient has a patient number which uniquely identifies his/her records in the hospital.
- Each ward has a unique ward number, and a name.
- While the patient is in hospital, he/she may be prescribed with drugs on a certain date and time. Note that multiple different drugs can be prescribed at the same time.
- Each drug has a unique drug ID, and a name.
- Each prescription has a dosage.

  The followings attributes/columns are identified:

  patientNo, pName, wardNo, wName, bedNo, admissionDate, dischargeDate, drugID, dName, dosage, pDate, pTime

- Identify a primary key (which may comprises multiple attributes) for the table.
  - A possible primary key is:
    - `(patientNo, admissionDate, pDate, pTime, drugID)`
    - This combination of attributes allows you to identify a prescription in the patient's record.
  - By `(patientNo, admissionDate)`, you can locate a particular admission instance of a patient, but you cannot locate a specific prescription.
  - By `(patientNo, admissionDate, pDate, pTime)`, you can locate a specific prescription at a certain time. However, as multiple drugs can be prescribed at a time, you still don't know which drug.

- List all functional dependencies among the attributes.
  - The primary key is `(patientNo, admissionDate, pDate, pTime, drugID)`.
  - Non-prime attributes are: `pName, wardNo, wName, bedNo, dischargeDate, dName, dosage`.
  - The primary key uniquely identifies a row. So we have the following dependencies (but may not be "***fully*** functionally dependent"):
    - `{patientNo, admissionDate, pDate, pTime, drugNo} → pName`
    - `{patientNo, admissionDate, pDate, pTime, drugNo} → wardNo`
    - `{patientNo, admissionDate, pDate, pTime, drugNo} → wName`
    - `{patientNo, admissionDate, pDate, pTime, drugNo} → bedNo`

- {patientNo, admissionDate, pDate, pTime, drugNo} → dischargeDate
- {patientNo, admissionDate, pDate, pTime, drugNo} → dName
- {patientNo, admissionDate, pDate, pTime, drugNo} → dosage

Then we can work through the non-prime attributes one by one:

- patientNo → pName
- {patientNo, admissionDate} → wardNo
- wardNo → wName
- {patientNo, admissionDate} → bedNo
- {patientNo, admissionDate} → dischargeDate
- drugID → dName

The following partial functional dependencies on the primary key are identified, which break 2NF. So non-prime attributes pName, wardNo, bedNo, dischargeDate and dName are not ***fully*** functionally dependent on the primary key:

- patientNo → pName
- {patientNo, admissionDate} → wardNo
- {patientNo, admissionDate} → bedNo
- {patientNo, admissionDate} → dischargeDate
- drugID → dName

The following does not break 2NF as wardNo is not part of the primary key:

- wardNo → wName

Transform the table to 2NF by removing the partial functional dependencies on the primary key. We have 4 tables:

(patientNo, admissionDate, drugID, pDate, pTime), dosage

patientNo, pName

drugID, dName

(patientNo, admissionDate), wardNo, wName, bedNo, dischargeDate

First 3 tables have only 1 non-prime attribute. So there is no room for breaking 3NF (which requires at least 2 non-prime attributes for transitive dependency to occur).

In the 4th table, we have the following dependencies on the primary key:

{patientNo, admissionDate} → wardNo
{patientNo, admissionDate} → wName
{patientNo, admissionDate} → bedNo
{patientNo, admissionDate} → dischargeDate

Again,

Again, go through the non-prime attributes to identify the following dependency:
wardNo → wName

So we have a transitive dependency on the primary key, which breaks 3NF:
{patientNo, admissionDate} → wardNo → wName

Transform to 3NF by removing transitive functional dependencies. We gave 5 tables:

(patientNo, admissionDate, drugID, pDate, pTime), dosage

patientNo, pName

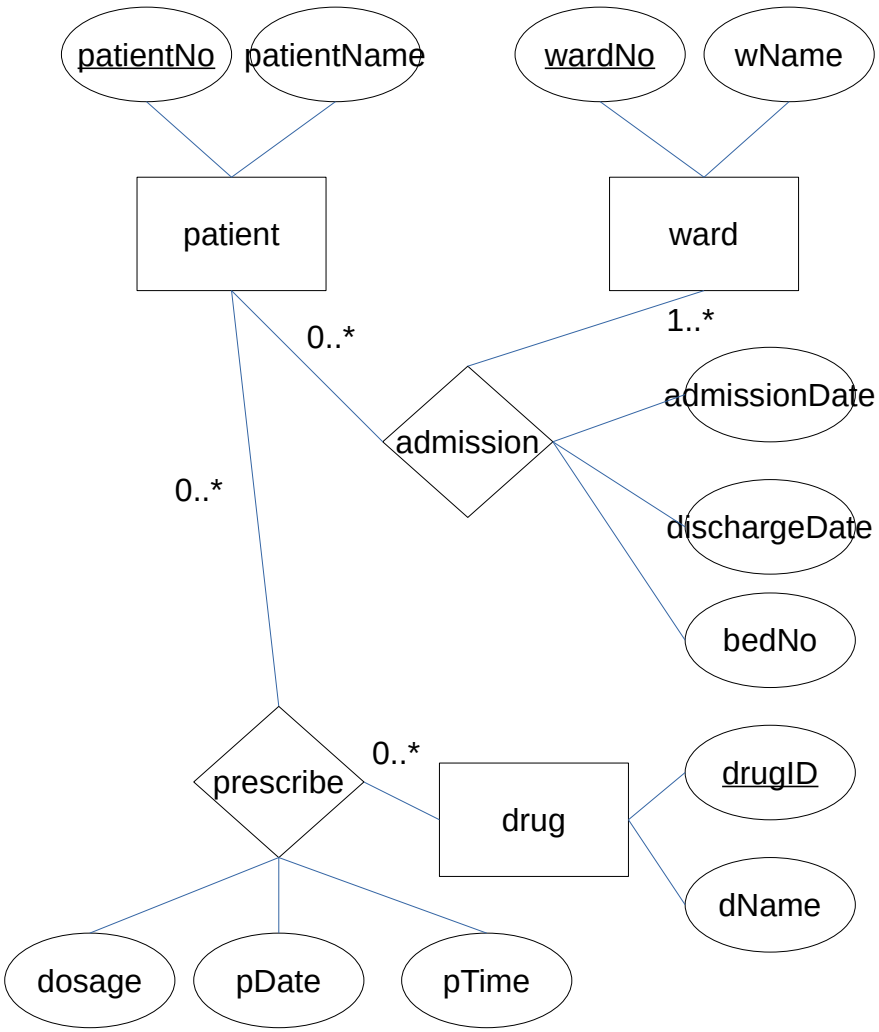
drugID, dName

(patientNo, admissionDate), wardNo, bedNo, dischargeDate

wardNo, wName

Approach#2

In this second approach, I start by doing ER modelling of the domain. The ER model is then mapped into tables and finally normalised.



Notes:

- I limit my modelling to the 10 attributes stated in the question. Some of you may want to include "doctor" as an entity. This is completely fine if you want to know who is the doctor prescribing a drug. But as the question does not state this, I did not include "doctor".
- There is no foreign key in an ER model. I have said this many times. See Lecture 01, slide 32.
- See how "`admission`" is modelled as a relationship between "`patient`" and "`ward`". This is based on my understanding. Yours can be different.
  - "`admissionDate`", "`dischargeDate`" and "`bedNo`" are all attributes of the "`admission`" relationship as they are attached to an

instance of "admission". Try to attach them to either "patient" or "ward". They won't fit as every time a patient is admitted to a ward, these attribute values may change.
    - See that 1 patient can be in "1 or many" wards. It cannot be 0 ward. It can be multiple wards as the patient may have been admitted to this hospital before, in different wards.
    - A ward can have "0 or many" patients.
- Similarly, I model "prescribe" (or "prescription") as a relationship between "patient" and "drug". You can model "prescription" as an entity if you wish. We will then see how the final schema may differ.
    - Again, see how I model "dosage", "pDate", and "pTime" as attributes of the "prescribe" relationship.
    - As mentioned above, I didn't model "doctor". If you do have a "doctor", it can be an attribute of this "prescribe" relationship, or the "doctor" entity can take part in this relationship (It will then be a N-ary relationship).
    - A patient can be related to "0 or many drugs".
    - A drug can be prescribed to "0 or many patients".

From the ER model, we can mapped it into tables.

First we map entities into tables:

Table: patient	
patientNo	varchar(16)
patientName	varchar(255)

Table: ward	
wardNo	int
wName	varchar(32)

Table: drug	
drugID	varchar(32)
dName	varchar(64)

Notes:

- Your chosen data types for the attributes may differ. e.g. I assume patientNo is a max-16-character string. You may want it to be an int, or a string of different length.

Next we map the "admission" relationship to a table:

Table: admission	
patientNo	varchar(16)
wardNo	int
admissionDate	datetime
dischargeDate	datetime

bedNo	int

Notes:

- This table has a composite primary key `(patientNo,admissionDate)`.
  - Knowing the patientNo+wardNo is not enough to know which admission instance it is, as a patient can be admitted to the same ward on different occasions (i.e. dates)
  - We assume a patient cannot be admitted to the same ward more than once on 1 day. If this is not true, you will need to modify the primary key.
  - We also assume a patient cannot be admitted to different wards on the same day. Otherwise you will need to include "wardNo" in the key.

Similarly, we map the "`prescribe`" relationship into a table:

Table: prescribe	
patientNo	varchar(16)
drugID	varchar(32)
pDate	datetime
pTime	datetime
dosage	int

Notes:

- This table has a composite primary key `(patientNo,drugID,pDate,pTime)`.
  - If we assume a patient can be prescribed the same drug multiple times on the same day, we need to know the different occasions. That's why "`pTime`" is part of the key.
  - "`dosage`" can be a `float`. It is a numeric value, although we did not specify the unit.

Normalisation

With the tables, we are ready to perform normalisation.

First normal form

- No attribute in any table is multiple-valued. All tables are in 1NF.

Second normal form

- A table may only violate 2NF it has a composite primary key. (See Lecture 03, slide 35.)
- Tables "`patient`", "`ward`", and "`drug`" do not have composite primary key. So they are already in 2NF.
- For the other 2 tables, we need to look at each table systematically.
- For each table, we can systematically look at the non-prime attributes:
- For table "`admission`", the following functional dependencies are identified:
- There are 3 non-prime attributes "`wardNo`", "`dischargeDate`" and "`bedNo`".
  - The following functional dependencies are identified:
    - `{patientID,admissionDate} → wardNo`

        - {patientID,admissionDate} → dischargeDate
        - {patientID,admissionDate} → bedNo
    - There is no partial dependency of non-prime attributes on the primary key. All non-prime attributes are fully functionally dependent on the primary key.
    - Table "admission" is in 2NF.
- For table "prescribe":
    - There is only 1 non-prime attribute "dosage".
    - The following functional dependencies are identified:
    - {patientID,drugID,pDate,pTime}→ dosage
    - Again, there is no partial dependency of non-prime attribute on the primar key. All non-prime attributes are fully functionally dependent on the primary key
    - Table "prescribe" is in 2NF.
- There is no change to our tables after normalisation to 2NF.

Third normal form

- Again, we need to look at each table systematically to see if there is any transitive dependency which will break 3NF.
- For table "admission", the following functional dependencies are identified:
    - {patientID,admissionDate} → wardNo
    - {patientID,admissionDate} → dischargeDate
    - {patientID,admissionDate} → bedNo
    - There is no transitive dependency.
    - Table "admission" is in 3NF.
- For table "prescribe", the following functional dependencies are identified:
    - {patientID,drugID,pDate,pTime}→ dosage
    - There is no transitive dependency.
    - Table "prescribe" is in 3NF.

All tables are now in 3NF.