Normalisation II

G51DBI – Databases and Interfaces

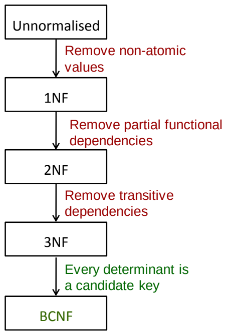
Yorgos Tzimiropoulos

yorgos.tzimiropoulos@nottingham.ac.uk

Overview of Normalisation

- Normalisation
  - Data Redundancy
  - Functional Dependencies
  - Normal Forms
  - First, Second and Third Normal Forms
- This Lecture

  Boyce-Codd Normal Form


Unnormalised
Remove non-atomic values
1NF
Remove partial functional dependencies
2NF
Remove transitive dependencies
3NF
Every determinant is a candidate key
BCNF

Second Normal Form

Second normal form:

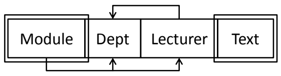
A relation is in second normal form (2NF) if it is in 1NF and **no non-key attribute is *partially* dependent on the primary key**

Normalising to 2NF

Suppose we have a relation R and the partial FD

A, C → B

A → B



- **Table is not in 2NF**
- There are FDs

{Module, Text} → {Dept, Lecturer}

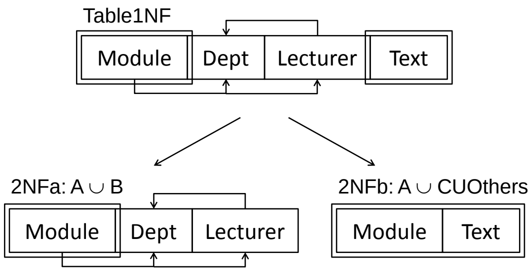
{Module} → {Dept, Lecturer}

We split the relation into two new relations

The first contains all of the columns contained in A and B: **AUB**

The second contains the columns contained in A and C and all of the (other) columns which are not contained in A, B or C: **AUCUOthers**

Normalising to 2NF



Transitive FDs and 3NF

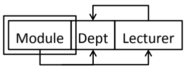
Third normal form:

A relation is in third normal form (3NF) if **it is in 2NF** and **no non-key** attribute is **transitively dependent** on the primary key

Normalising to 3NF

Suppose we have a relation and the transitive FD

$A \rightarrow B \rightarrow C$



- **Table2NFa is not in 3NF**
- There are FDs
  - {Module} → {Lecturer}
  - {Lecturer} → {Dept}

We split the relation into two new relations

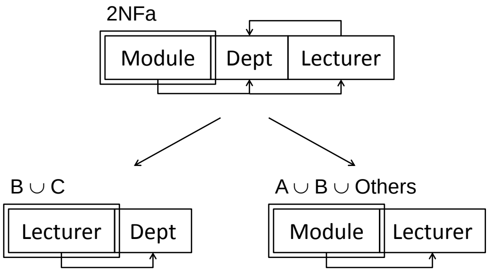
The first contains all of the columns contained in B and C:

BUC

The second contains the columns contained in A and B and all of the (other) columns which are not contained in A, B or C:

AUBUOthers

Normalising to 3NF



General definitions of 2NF and 3NF

- So far we have selected a primary key and proceeded with normalising the table to 2NF and then to 3NF
- However this process does not take into account other candidate keys of a relation

General definitions

- 2NF: A relation that is in 1NF and every non-candidate key attribute is fully (i.e. non partially) dependent on any candidate key
- 3NF: A relation that is in 1NF and 2NF and in which no non-candidate key attribute is transitively dependent on any candidate key

Informally – to help remember only

- **[Every] non-key [attribute] must provide a fact about the key, the whole key, and nothing but the key.**

- 1NF: **A key exists**, cell contents atomic
- 2NF: Non-key attributes depend on the **whole** key
- 3NF: Non-key attributes depend on **nothing but** the key
- BCNF: **All attributes** depend on nothing but the key
  - This lecture

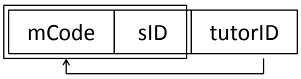
Boyce-Codd Normal Form

- Let’s consider extending our Grade table from the University Database example
  - Each student will be assigned a (PhD student) tutor for each module they are on
  - Tutors can have many students, **but can only help with one module** (i.e. if you know the tutor, then you can work out the module)
  - A module can have many tutors assigned to it

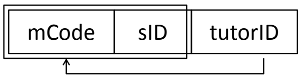
mCode	sID	tutorID

Boyce-Codd Normal Form

- Let’s consider extending our Grade table from the University Database example
  - Each student will be assigned a (PhD student) tutor for each module they are on
  - Tutors can have many students, **but can only help with one module** (i.e. if you know the tutor, then you can work out the module)
  - A module can have many tutors assigned to it



Boyce-Codd Normal Form



Functional Dependencies

- mCode,sID -> tutorID
- tutorID->mCode
- Table is already in the 3rd Normal Form

Problems with 3NF

Grade

mCode	studentID	tutorID
G51DBI	109684	T001
G51PRG	108348	T002
G51IAI	110798	T003
G51DBI	112943	T001
G51OOP	107749	T016
G51PRG	109684	T002
G51OOP	110798	T015

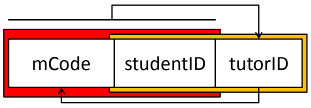
- INSERT Anomalies
  - Can't add a tutor who isn't currently tutoring anyone
- UPDATE Anomalies
  - Changing the module a tutor teaches is complicated and involves multiple rows
- DELETE Anomalies
  - If we remove student 110798, we no longer know that T003 is tutoring in G51IAI

Boyce-Codd Normal Form

- A relation is in Boyce-Codd normal form (BCNF) if for every FD $A \rightarrow B$ either
  - B is contained in A (the FD is trivial), or
  - A contains a candidate key of the relation
- In other words: every determinant in a non-trivial dependency is a key.

Example

- The Grade table is in 3NF but not BCNF
  - {tutorID} is not a candidate key, however the FD {tutorID} → {mCode} exists
    - Remember: tutor is on only one module
  - {mCode, studentID} → {tutorID} is ok because {mCode, studentID} is a key



Normalising to BCNF

- Suppose we have a relation R with schema S and the FD $A \rightarrow B$ that violates BCNF

  $A \cap B = \{\}$
- Let Others = S - $(A \cup B)$
- In other words:
  - A – attributes on the left hand side of the FD
  - B – attributes on the right hand side of the FD
  - Others – all other attributes

- To normalise to BCNF we create two new relations

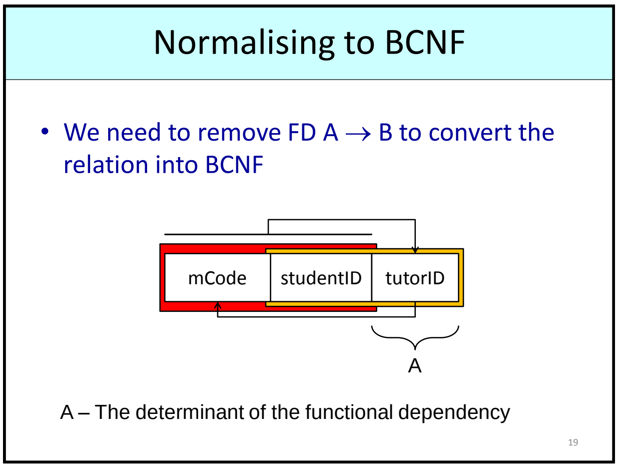
  ??

Normalising to BCNF

- Suppose we have a relation R with schema S and the FD $A \rightarrow B$ that violates BCNF

  $A \cap B = \{\}$
- Let Others = S - $(A \cup B)$
- In other words:
  - A – attributes on the left hand side of the FD
  - B – attributes on the right hand side of the FD
  - Others – all other attributes

- To normalise to BCNF we create two new relations
  - **A ∪ Others**
  - **A ∪ B**



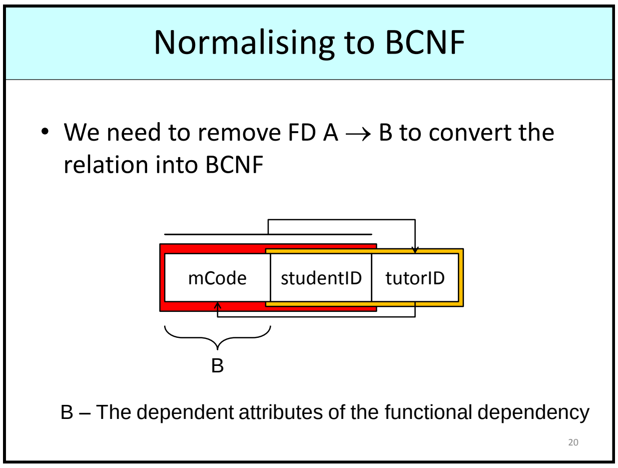
Normalising to BCNF

- We need to remove FD A → B to convert the relation into BCNF

mCode studentID tutorID

A

A – The determinant of the functional dependency



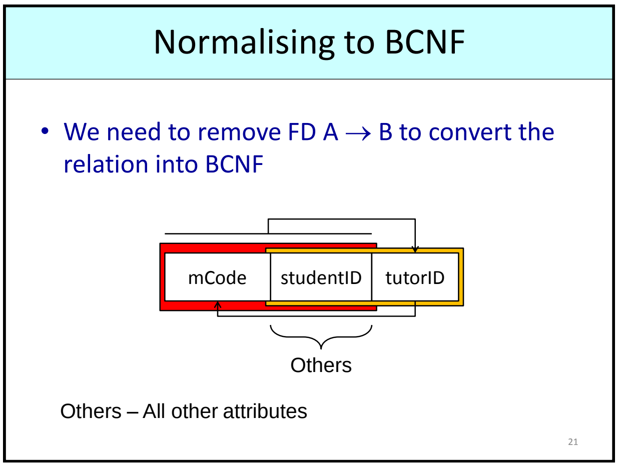
Normalising to BCNF

- We need to remove FD A → B to convert the relation into BCNF

mCode studentID tutorID

B

B – The dependent attributes of the functional dependency



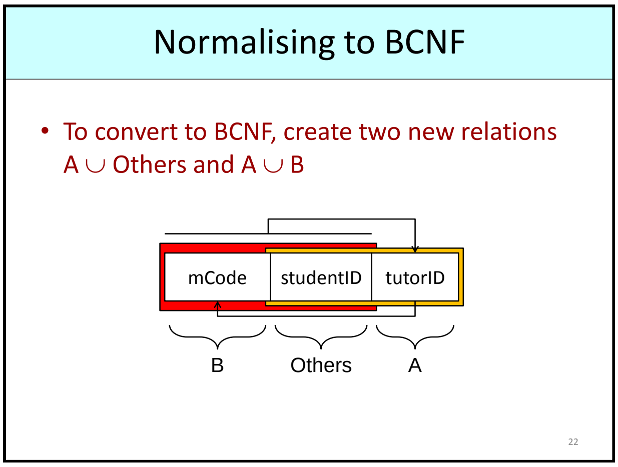
Normalising to BCNF

- We need to remove FD A → B to convert the relation into BCNF

mCode studentID tutorID

Others

Others – All other attributes

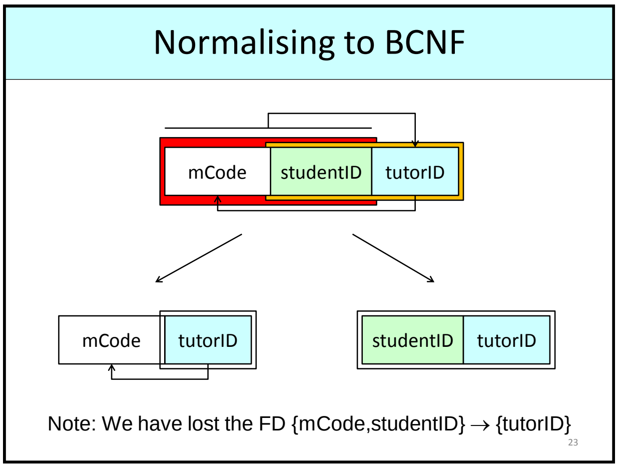


Normalising to BCNF

- To convert to BCNF, create two new relations A ∪ Others and A ∪ B

mCode studentID tutorID

B Others A



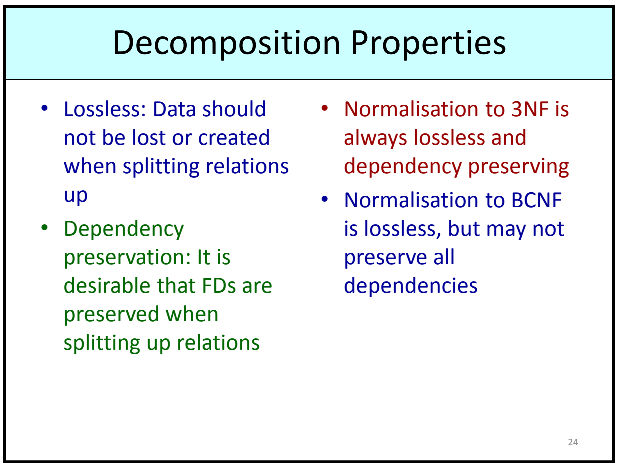
Normalising to BCNF

mCode studentID tutorID

mCode tutorID

studentID tutorID

Note: We have lost the FD {mCode,studentID} → {tutorID}



Decomposition Properties

- Lossless: Data should not be lost or created when splitting relations up
- Dependency preservation: It is desirable that FDs are preserved when splitting up relations

- Normalisation to 3NF is always lossless and dependency preserving
- Normalisation to BCNF is lossless, but may not preserve all dependencies

Example

ClientInterview

clientNo	interviewDate	interviewTime	staffNo	roomNo
CR76	13/05/16	10:30	SG5	G101
CR56	13/05/16	12:00	SG5	G101
CR74	13/05/16	12:00	SG37	G102
CR56	01/07/16	10:30	SG5	G102

Functional dependencies:

Example

ClientInterview

clientNo	interviewDate	interviewTime	staffNo	roomNo
CR76	13/05/16	10:30	SG5	G101
CR56	13/05/16	12:00	SG5	G101
CR74	13/05/16	12:00	SG37	G102
CR56	01/07/16	10:30	SG5	G102

Functional dependencies:

1. {clientNo, interviewDate} -> {interviewTime, staffNo,roomNo}
2. {staffNo, interviewDate, interviewTime} -> clientNo
3. {roomNo, interviewDate, interviewTime } -> {staffNo,clientNo}
4. {staffNo, interviewDate} -> roomNo

Example

ClientInterview

clientNo	interviewDate	interviewTime	staffNo	roomNo
CR76	13/05/16	10:30	SG5	G101
CR56	13/05/16	12:00	SG5	G101
CR74	13/05/16	12:00	SG37	G102
CR56	01/07/16	10:30	SG5	G102

- staffNo, interviewDate is not a candidate key
- Hence the fd

{staffNo, interviewDate} -> roomNo

violates the BCNF

Example

ClientInterview

clientNo	interviewDate	interviewTime	staffNo	roomNo
CR76	13/05/16	10:30	SG5	G101
CR56	13/05/16	12:00	SG5	G101
CR74	13/05/16	12:00	SG37	G102
CR56	01/07/16	10:30	SG5	G102

- To convert to BCNF, we split the original relation to 2 tables
- One contains the fd that violates BCNF

  staffNo, interviewDate, roomNo
- The other contains the left hand side of the fd violating the BCNF and all other attributes

  staffNo, interviewDate, interviewTime, clientNo

Example

Interview

clientNo	interviewDate	interviewTime	staffNo
CR76	13/05/16	10:30	SG5
CR56	13/05/16	12:00	SG5
CR74	13/05/16	12:00	SG37
CR56	01/07/16	10:30	SG5

StaffRoom

interviewDate	staffNo	roomNo
13/05/16	SG5	G101
13/05/16	SG5	G101
13/05/16	SG37	G102
01/07/16	SG5	G102

{roomNo, interviewDate, interviewTime}-> {staffNo,clientNo} is **lost**!

Example

- {roomNo, interviewDate, interviewTime} -> {staffNo,clientNo} is **lost**!
- If members of staff conduct numerous interviews per day then the presence of fd4 in the original table (ClientInterview) will cause redundancy
- In this case, normalisation of this relation to BCNF is recommended
- If the lost fd is too important though then one might stop at 3NF

Example 2

StaffPropertyInspection

propNo	iDate	iTime	propAddress	comment	staffNo	staffName	carReg
pNo	iD	iT	pA	c	sNo	sName	cR

Functional dependencies:

pNo, iD -> iT, pA, c, sNo, sName, cR

pNo -> pA

sNo -> sName

iD, sNo -> cR

iD, iT, cR -> pNo, pA, c, sNo, sName

iD, iT, sNo -> pNo, pA, c

Example 2

AUB fd: pNo->pA

pNo	pA

AUCUOthers

pNo	iD	iT	c	sNo	sName	cR

After removing partial dependencies and splitting:

pNo, iD -> iT, c, sNo, sName, cR

sNo -> sName

iD, sNo -> cR

iD, iT, cR -> pNo, c, sNo, sName

iD, iT, sNo -> pNo, c

Example 2

fd: pNo->pA

pNo	pA

BUC fd: sNo -> sName

sNo	sName

AUBUOthers

pNo	iD	iT	c	sNo	cR

After removing transitive dependency and splitting:

pNo, iD -> iT, c, sNo, cR

iD, sNo -> cR

iD, iT, cR -> pNo, c, sNo

iD, iT, sNo -> pNo, c

Example 2

fd: pNo->pA

pNo	pA

fd: sNo -> sName

sNo	sName

AUB fd: iD, sNo -> cR

iD	sNo	cR

AUOthers

pNo	iD	iT	c	sNo

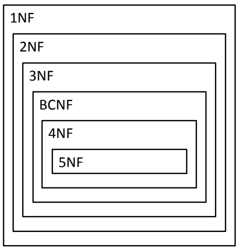
Ensuring all determinants are Candidate Keys:

pNo, iD -> iT, c, sNo

iD, iT, sNo -> pNo, c

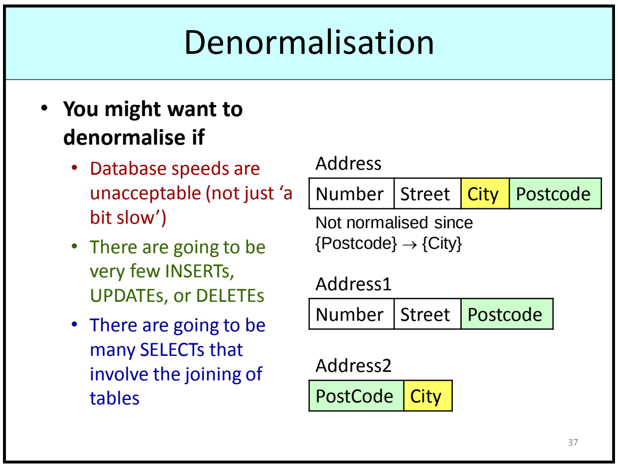
Higher Normal Forms

- **BCNF is as far as we can go with FDs**
  - Higher normal forms are based on other sorts of dependency
  - Fourth normal form removes multi-valued dependencies
  - Fifth normal form removes join dependencies



Denormalisation

- **Normalisation**
  - Removes data redundancy
  - Solves INSERT, UPDATE, and DELETE anomalies
  - This makes it easier to maintain the information in the database in a consistent state
- **However**
  - It leads to more tables in the database
  - Often these need to be joined back together, which is expensive to do
  - So sometimes (not often?) is it worth 'denormalising' ?



Denormalisation

- **You might want to denormalise if**
  - Database speeds are unacceptable (not just 'a bit slow')
  - There are going to be very few INSERTs, UPDATEs, or DELETEs
  - There are going to be many SELECTs that involve the joining of tables

Address

Number	Street	City	Postcode

Not normalised since {Postcode} → {City}

Address1

Number	Street	Postcode

Address2

PostCode	City

37

Thanks for your attention

- Any questions?

38