Normalisation

G51DBS Database Systems

Dr Jason Atkin

This Lecture

- **VERY IMPORTANT LECTURES! Normalisation is a key topic in relational database design**

- Normalisation
  - Data Redundancy
  - Functional Dependencies
  - Normal Forms
  - First, Second and Third Normal Forms
- Further reading
  - The Manga Guide to Databases, Chapter 3
  - Database Systems, Chapter 14

Redundancy and Anomalies

- Redundant data and database anomalies
  - Some data could be repeated in the database
  - Some data can be determined from other data (e.g. sum, max, count)
  - There may be enforced/false relationships between data due to database structure.
  - These can all lead to various problems/anomalies
- INSERT Anomalies / Insertion Anomalies
  - May not be possible to store some data at all, due to relationships with other required data. I.e. it may not be possible to correctly insert some data
- UPDATE Anomalies
  - If data is in multiple places, all copies need to be updated at once. If not then the data will be inconsistent. I.e. some updates may result in anomalies
- DELETE Anomalies / Deletion Anomalies
  - Deleting some facts/data requires the deletion of other facts/data, which may still be relevant. i.e. some deletes may remove too much data

Normalisation

- Normalisation
  - Aims to reduce data redundancy
  - Redundancy is expressed in terms of functional dependencies
  - Normal forms are defined that don't contain specific types of functional dependency
- Normalisation is a formal process for something that you will often do naturally
  - When you create your tables, they'll often be normalised already
- E/R diagrams help produce normalised tables
- Despite being somewhat common sense, it's good to have a formal process we can use

First Normal Form

- In most definitions of the relational model:
  - All data values should be atomic
  - This means that table entries should be single values, not sets or composite objects
  - Simplifies queries and data comparisons
- A relation is said to be in first normal form (1NF) if all data values are atomic
- As well as the other required things, e.g.:
  - Row order cannot matter
  - Column order cannot matter

Normalisation to 1NF

- To convert any relation into 1NF, split any non-atomic values

Unnormalised

Module	Dept	Lecturer	Texts
M1	D1	L1	T1, T2
M2	D1	L1	T1, T3
M3	D2	L2	T4
M4	D2	L3	T1, T5
M5	D2	L4	T6

1NF

Module	Dept	Lecturer	Text
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6

But be careful with the key…

- Question: What are good primary keys for these tables? (Assuming it makes sense.)

Unnormalised

Module	Dept	Lecturer	Texts
M1	D1	L1	T1, T2
M2	D1	L1	T1, T3
M3	D2	L2	T4
M4	D2	L3	T1, T5
M5	D2	L4	T6

1NF

Module	Dept	Lecturer	Text
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6

But be careful with the key…

- Be careful with the primary key when you split the tables to convert to 1NF

Unnormalised

Module	Dept	Lecturer	Texts
M1	D1	L1	T1, T2
M2	D1	L1	T1, T3
M3	D1	L2	T4
M4	D2	L3	T1, T5
M5	D2	L4	T6

Module?

1NF

Module	Dept	Lecturer	Text
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6

Module + Text ?

Problems with 1NF

1NF

Module	Dept	Lecturer	Text
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6

- INSERT Anomalies
  - Can't add a module with no texts
- UPDATE Anomalies
  - To change the lecturer for M1, we will need to update two rows
- DELETE Anomalies
  - If we remove M3, we will remove L2 as well

Functional Dependencies

- Redundancy is often caused by a functional dependency
- A Functional Dependency (FD) is a link between two sets of attributes in a relation
- i.e. **if you know the value of one set of attributes, you also know the values of the others**
- We can normalise a relation by removing **undesirable** FDs

- A set of attributes, A, **functionally determine**s another set, B, if whenever two rows of the relation have the same values for all the attributes in A, then they also have the same values for all the attributes in B.
- In this case, we can say there exists **a functional dependency** between A and B ($A \rightarrow B$)

Example

- Consider this relation:

ID	First	Last	moduleCode	moduleName
111	Joe	Smith	G51PRG	Programming
222	Anne	Jones	G51DBS	Databases

- Three notable functional dependencies exist:
  - {ID} $\rightarrow$ {First, Last}
  - {moduleCode} $\rightarrow$ {moduleName}
  - {ID, moduleCode} $\rightarrow$ {First, Last, moduleName}

Properties of FDs

- In any relation
  - The primary key functionally determines **any** set of attributes in that relation

    $$K \rightarrow X$$

    - K is the primary key, X is any set of attributes
  - Same for candidate keys
    - Minimal set of cols which identifies the row – i.e. functionally determines it
  - Any set of attributes is FD on itself

    $$X \rightarrow X$$

- Rules for FDs
  - Subset:
    If B is a subset of A then

    $$A \rightarrow B$$

  - Augmentation:
    If $A \rightarrow B$ then

    $$A \cup C \rightarrow B \cup C$$

  - Transitivity:
    If $A \rightarrow B$ and $B \rightarrow C$ then

    $$A \rightarrow C$$

FDs and Normalisation

- We define a set of 'normal forms'
  - Each normal form has fewer FDs than the last
  - Since the FDs which we remove represent redundancy, each normal form has less redundancy than the last
- Not all FDs cause a problem
  - We identify various sorts of FD that do
  - Each normal form removes a type of FD that causes problems

FD Example

1NF

Module	Dept	Lecturer	Text
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6

- The Primary Key is {Module, Text} so
  - {Module, Text} → {Dept, Lecturer}
- 'Trivial' FDs, eg:
  - {Text, Dept} → {Text}
  - {Module} → {Module}
  - {Dept, Lecturer} → { }

FD Example

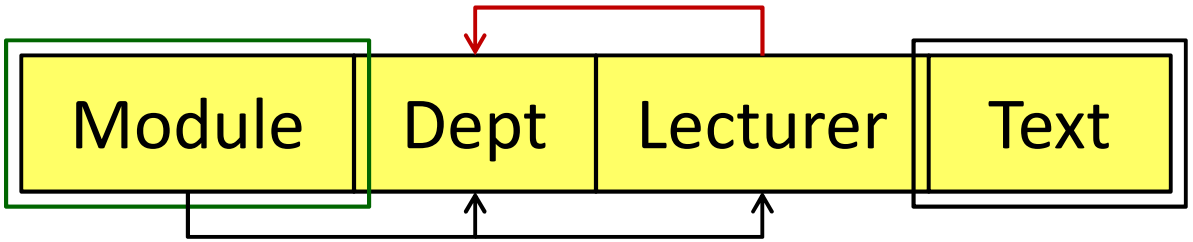
1NF

Module	Dept	Lecturer	Text
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6

- Other FDs are
  - {Module} $\rightarrow$ {Lecturer}
  - {Module} $\rightarrow$ {Dept}
  - {Lecturer} $\rightarrow$ {Dept}
  - These are non-trivial and the determinants (left hand side of the dependency) are not candidate keys.

FD Diagrams

- Rather than an entire table, FDs can be represented simply using the headings:



- {Module , Text} is a candidate key, so we put a double box around them
- {Lecturer} → {Dept}, so we have an arrow from Lecturer to Dept
- {Module} → {Dept} and {Module} → {Lecturer} , so we have
  {Module} → {Dept, Lecturer}

Note: Trivial FDs and FDs dependent on an entire candidate key are not included

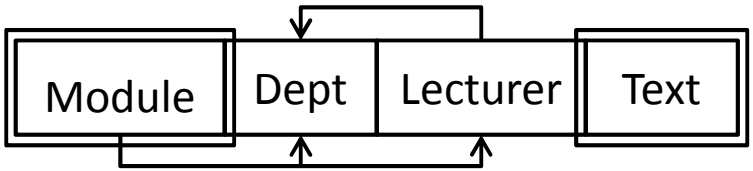
Second Normal Form

- **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 a candidate key**
  - In other words, no $C \rightarrow B$ where C is a **strict subset** of a candidate key and **B is a non-key attribute**
  - i.e. non-key B depends on ALL of the key
- **Partial FDs:**
  - A FD, $A \rightarrow B$ is a partial FD, if some attribute of A can be removed and the FD still holds
    - i.e. B depends on a part of A, rather than needing all of A
  - Formally, there is some proper subset of A, $C \subset A$, such that $C \rightarrow B$

Normalising to 2NF

Table1NF

Module	Dept	Lecturer	Text
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6



- 'Table1NF' is not in 2NF
  - We have the FD

{Module, Text} → {Lecturer, Dept}

  - But also

{Module} → {Lecturer, Dept}

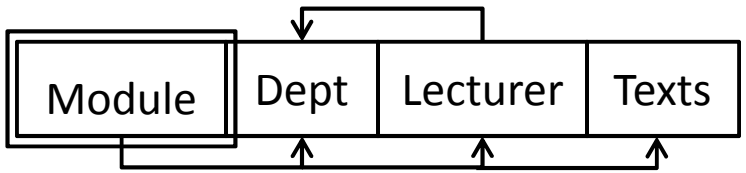
- And so Lecturer and Dept are **partially** dependent on the primary key

Observation about going to 1NF

- Going to 1NF meant that ‘Texts’/’Text’ was no longer dependent on Module

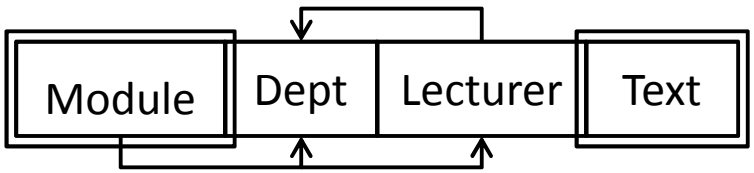
Unnormalised

Module	Dept	Lecturer	Texts
M1	D1	L1	T1, T2
M2	D1	L1	T1, T3
M3	D2	L2	T4
M4	D2	L3	T1, T5
M5	D2	L4	T6



1NF

Module	Dept	Lecturer	Text
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6



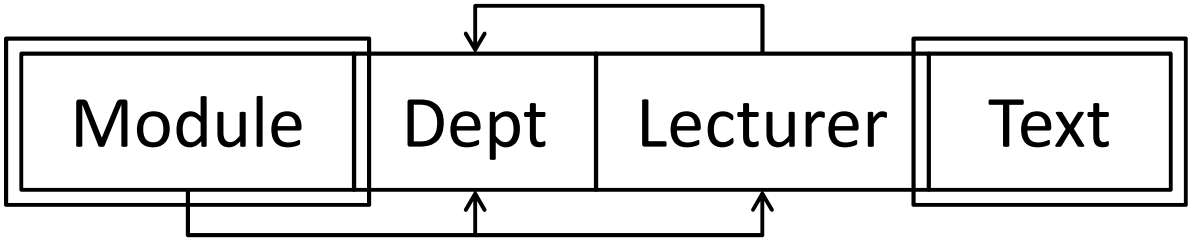
Normalising to 2NF

- Suppose we have a relation R with schema S and the FD $A \rightarrow B$ where $A \cap B = \{ \}$
- Let $C = S - (A \cup B)$
- In other words:
  - A – attributes on the left hand side of the FD
  - B – attributes on the right hand side of the FD
  - C – all other attributes
- Then we can split R into two parts:
  - R1, with schema $A \cup C$
  - R2, with schema $A \cup B$
- The original relation can be recovered as the natural join of R1 and R2:
  - R = R1 NATURAL JOIN R2

  (More next lecture on recovering the table)

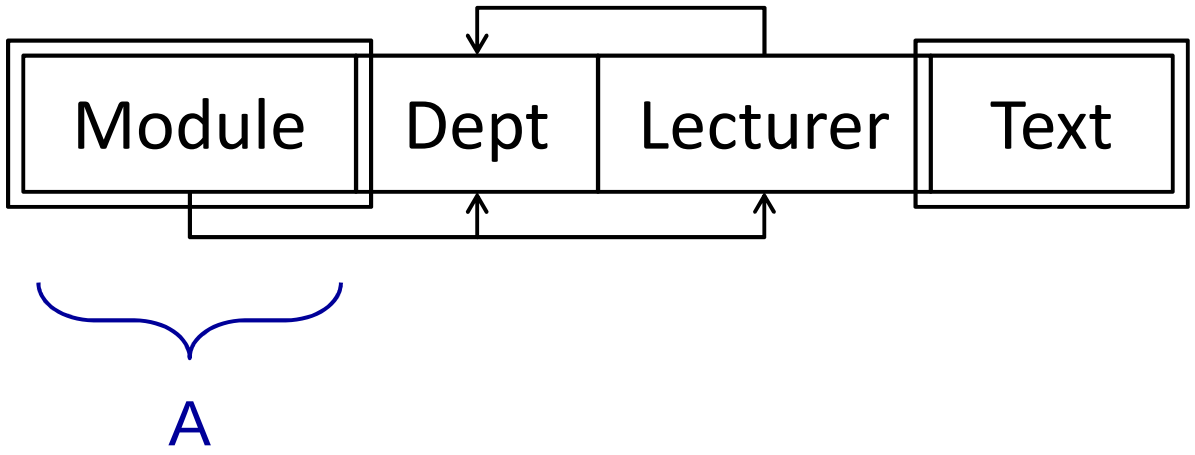
Normalising to 2NF

- We need to remove FD A $\rightarrow$ B in order to convert the relation to 2NF



Normalising to 2NF

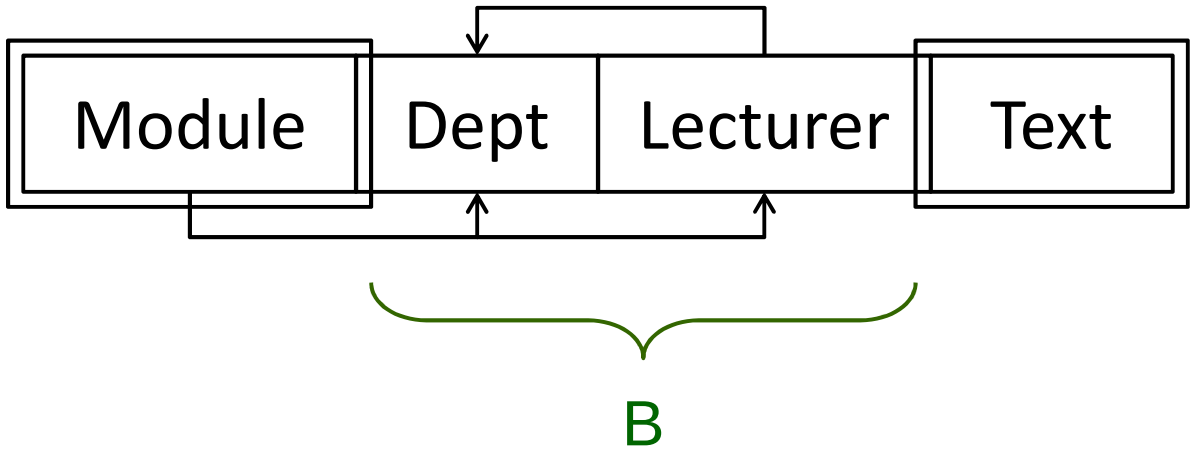
- We need to remove FD A $\rightarrow$ B in order to convert the relation to 2NF



A – The determinant of the functional dependency

Normalising to 2NF

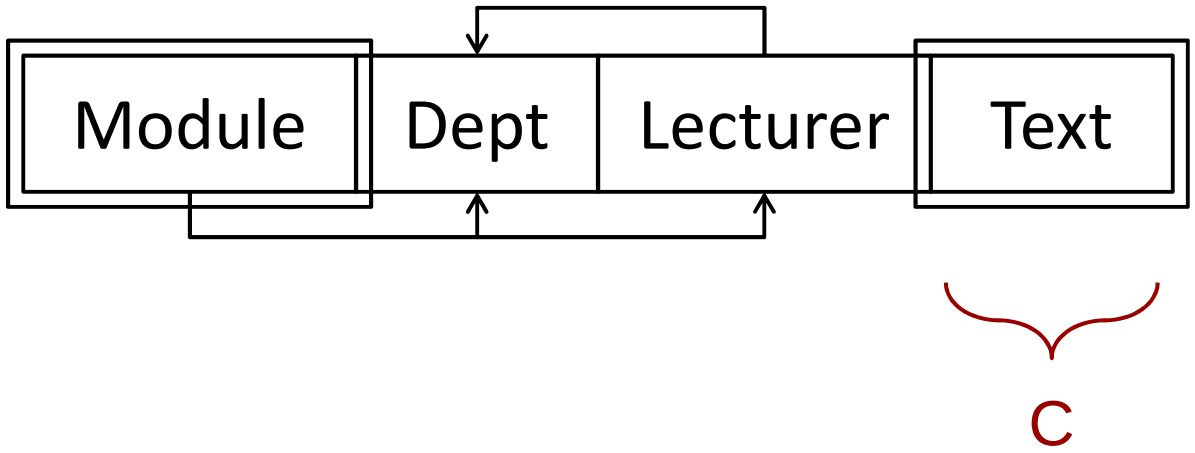
- We need to remove FD A $\rightarrow$ B in order to convert the relation to 2NF



B – The dependant attributes of the functional dependency

Normalising to 2NF

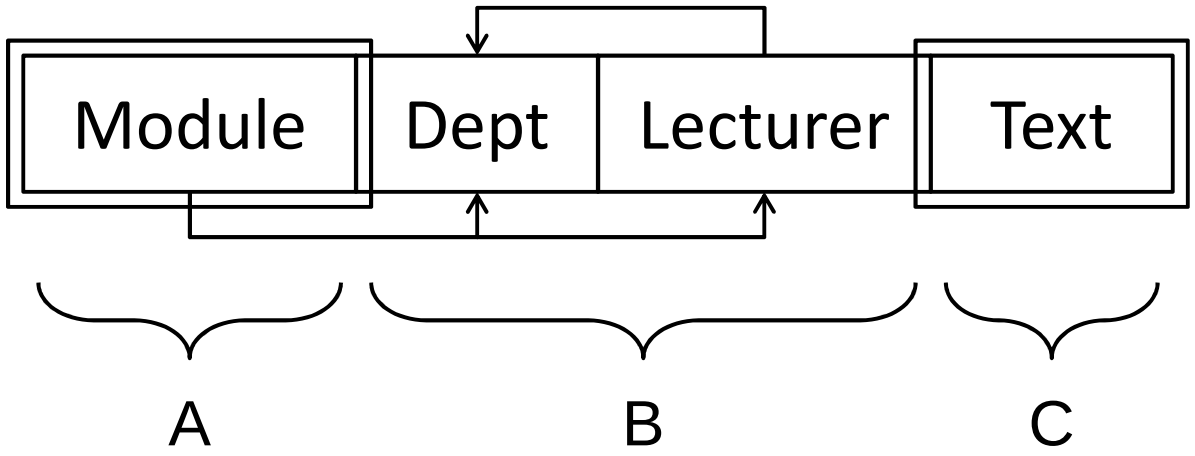
- We need to remove FD A $\rightarrow$ B in order to convert the relation to 2NF



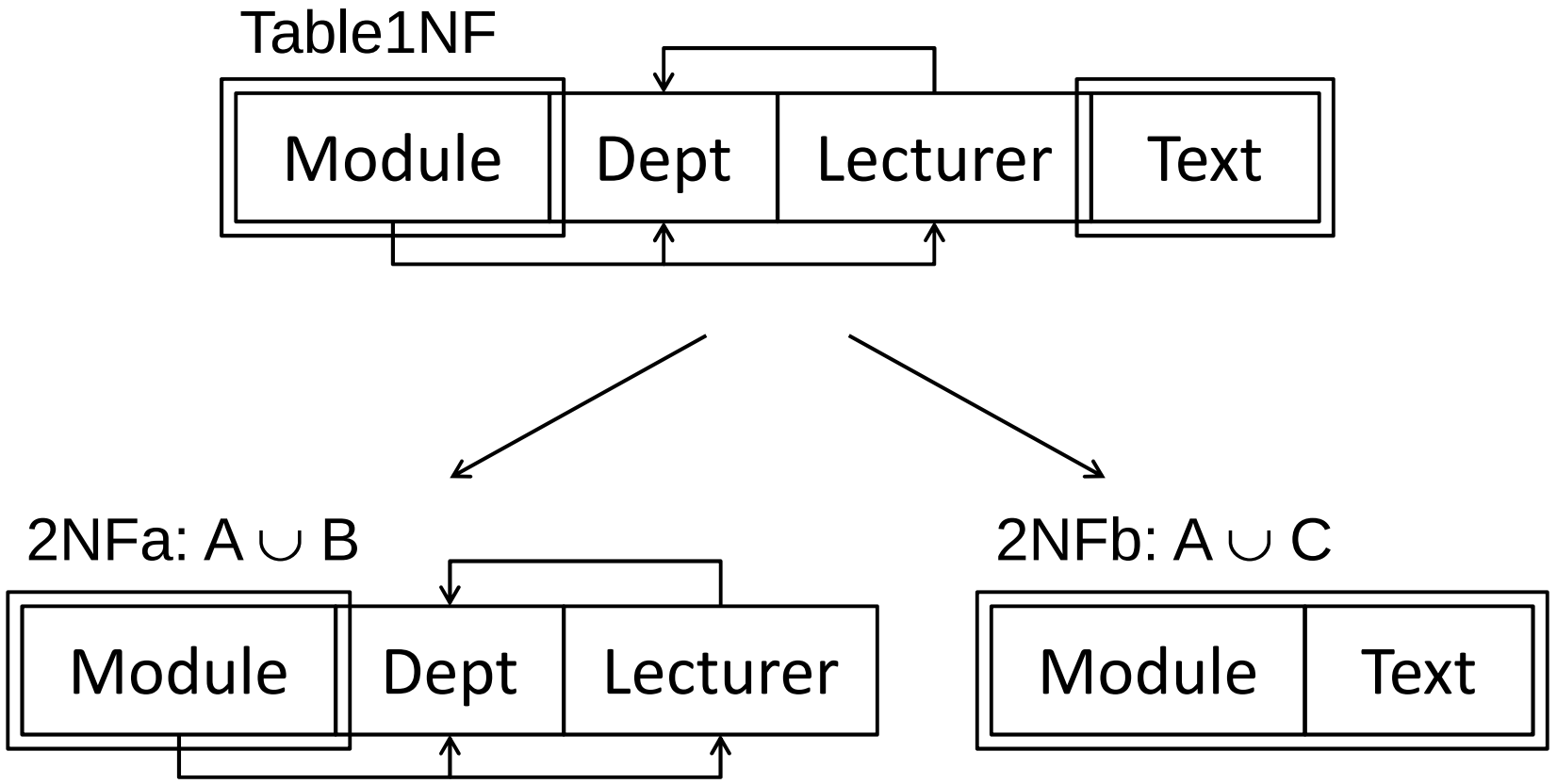
C – All remaining attributes in the relation

Normalising to 2NF

- To convert to 2NF, create two relations A ∪ B and A ∪ C



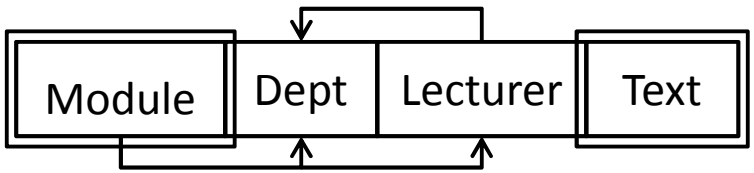
Normalising to 2NF



Normalising to 2NF

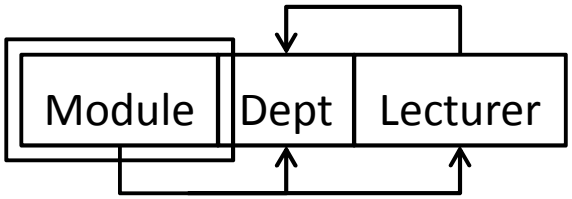
Table1NF

Module	Dept	Lecturer	Text
M1	**D1**	**L1**	T1
M1	**D1**	**L1**	T2
M2	**D1**	**L1**	T1
M2	**D1**	**L1**	T3
M3	D1	L2	T4
M4	**D2**	**L3**	T1
M4	**D2**	**L3**	T5
M5	D2	L4	T6



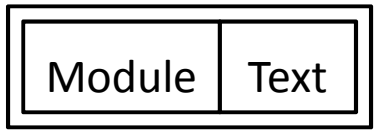
2NFa

Module	Dept	Lecturer
M1	D1	L1
M2	D1	L1
M3	D1	L2
M4	D2	L3
M5	D2	L4



2NFb

Module	Text
M1	T1
M1	T2
M2	T1
M2	T3
M3	T4
M4	T1
M4	T5
M5	T6



Problems Resolved in 2NF

- INSERT Anomalies
  - We can now add a module without texts
- UPDATE Anomalies
  - We only need to change a single row when changing a module lecturer

2NFa

Module	Dept	Lecturer
M1	D1	L1
M2	D1	L1
M3	D1	L2
M4	D2	L3
M5	D2	L4

Problems Remaining in 2NF

- INSERT Anomalies
  - We can't add lecturers who don't currently teach modules
- UPDATE Anomalies
  - To change the department for L1, we must change two rows
- DELETE Anomalies
  - To delete module M3, we must delete L2

2NFa

Module	Dept	Lecturer
M1	D1	L1
M2	D1	L1
M3	D1	L2
M4	D2	L3
M5	D2	L4

Transitive FDs and 3NF

- **Transitive FDs:**
  - A FD, $A \rightarrow C$ is a transitive FD, if there is some set B such that

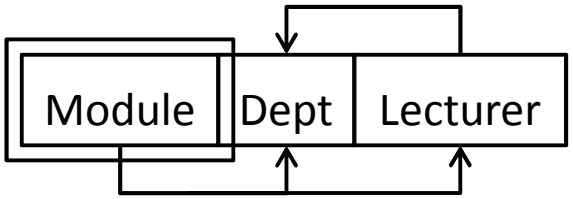
    $A \rightarrow B$ and $B \rightarrow C$ are non-trivial FDs
  - $A \rightarrow B$ non-trivial means: B is not a subset of A
  - Essentially

    $$A \rightarrow B \rightarrow C$$

- 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 a candidate key
  - This will never be the case if a non-key attribute is functionally dependent on other non-key attributes

Normalising to 3NF

Table2NFa



- **Table2NFa is not in 3NF**
  - There are FDs

    {Module} → {Lecturer}

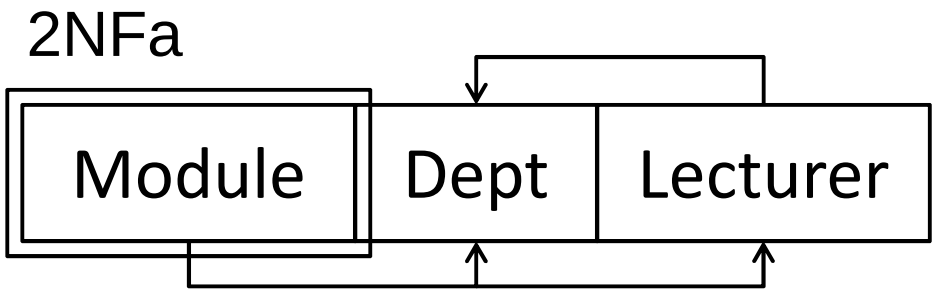
    {Lecturer} → {Dept}
  - So there is a **transitive** FD from Primary key {Module} to {Dept}

- **To move a relationship from 2NF to 3NF:**
- Given the transitive FD

  $A \rightarrow B \rightarrow C$
- We split the relation into two new relations
- The first contains all of the columns contained in B and C
- 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

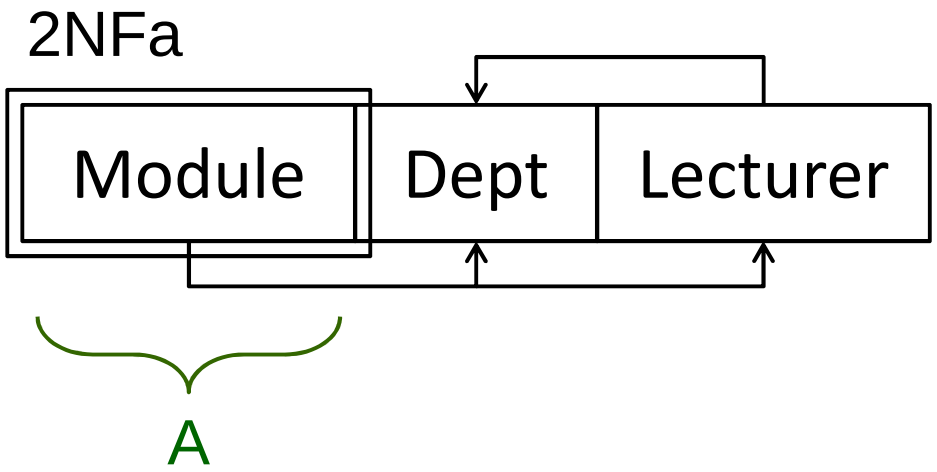
Normalising to 3NF

- We need to remove FD A $\rightarrow$ B $\rightarrow$ C in order to convert the relation to 3NF



Normalising to 3NF

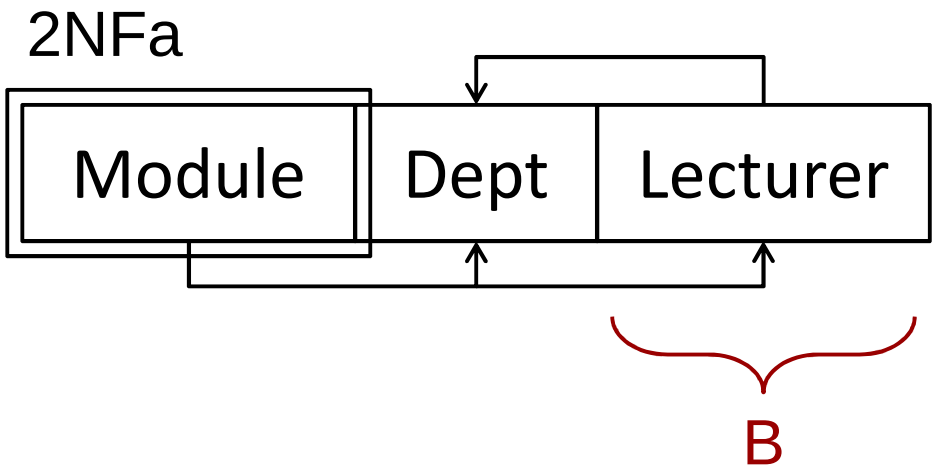
- We need to remove FD A $\rightarrow$ B $\rightarrow$ C in order to convert the relation to 3NF



A – The determinant of the functional dependency

Normalising to 3NF

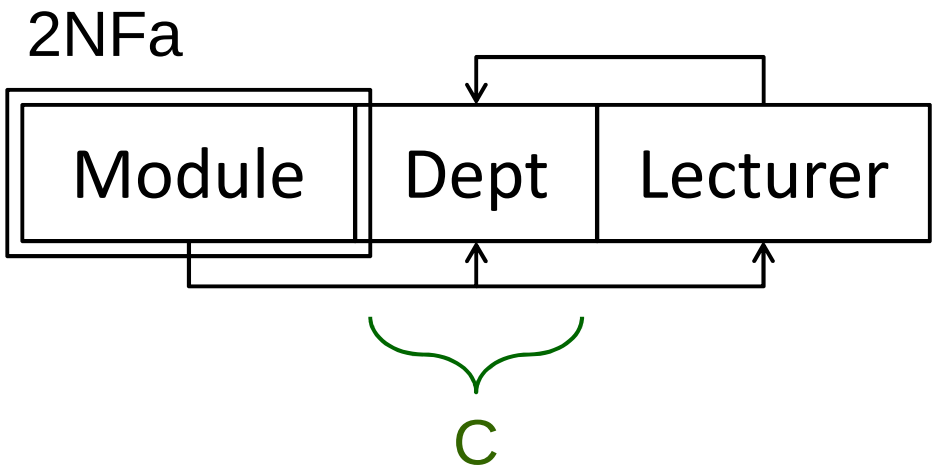
- We need to remove FD A $\rightarrow$ B $\rightarrow$ C in order to convert the relation to 3NF



B – The dependant attributes of the functional dependency

Normalising to 3NF

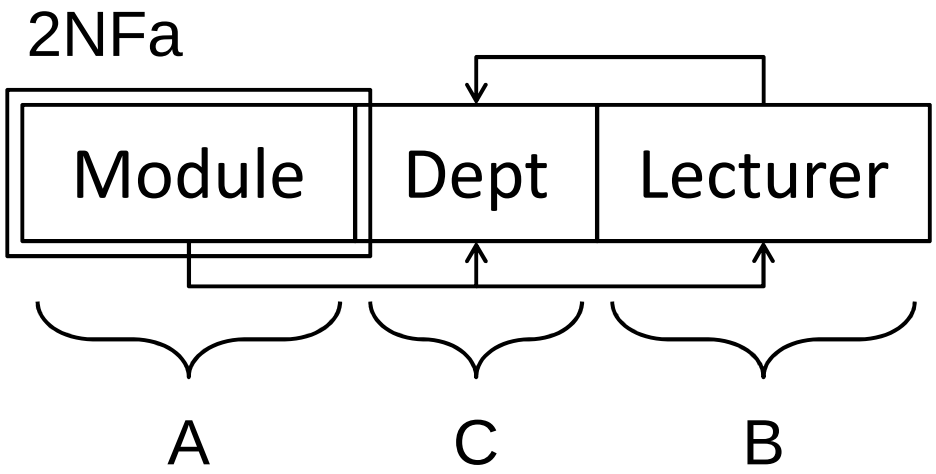
- We need to remove FD A $\rightarrow$ B $\rightarrow$ C in order to convert the relation to 3NF



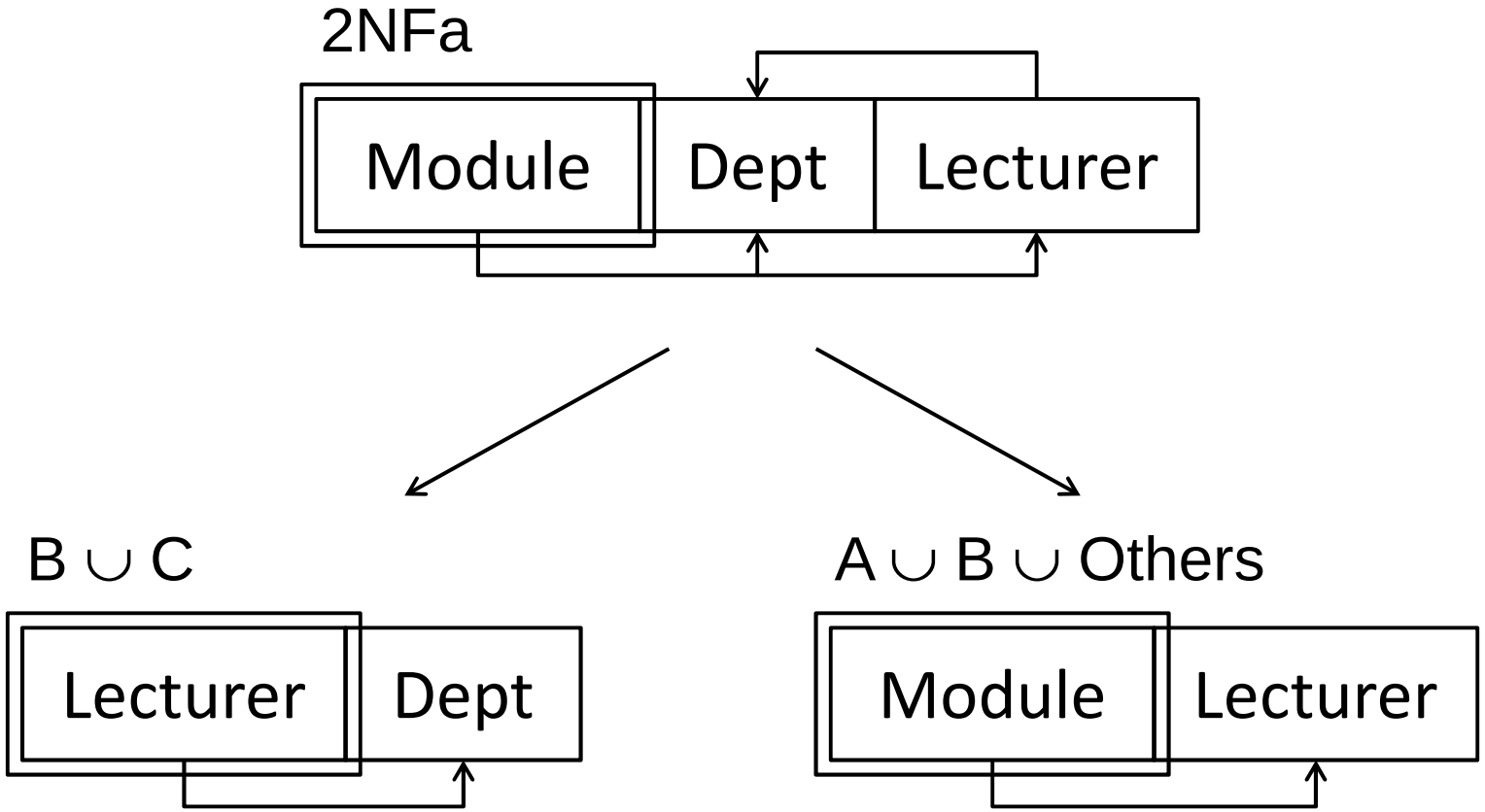
C – The transitively dependant attributes of the functional dependency

Normalising to 3NF

- To convert to 3NF, create two relations B ∪ C and A ∪ B ∪ Other Columns



Normalising to 3NF



Normalising to 3NF

2NFa

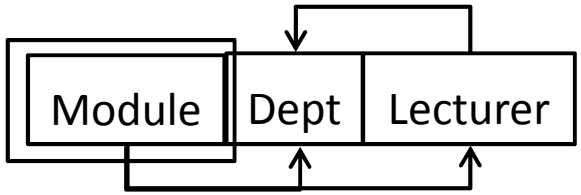
Module	Dept	Lecturer
M1	D1	L1
M2	D1	L1
M3	D1	L2
M4	D2	L3
M5	D2	L4

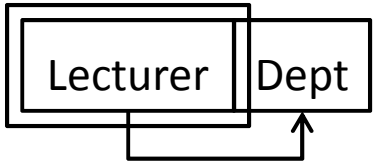
3NFa

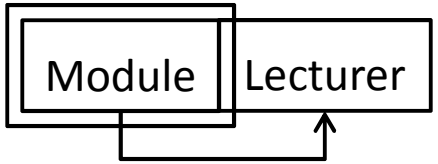
Lecturer	Dept
L1	D1
L2	D1
L3	D2
L4	D2

3NFb

Module	Lecturer
M1	L1
M2	L1
M3	L2
M4	L3
M5	L4







The 3NF tables we now have left

3NFa

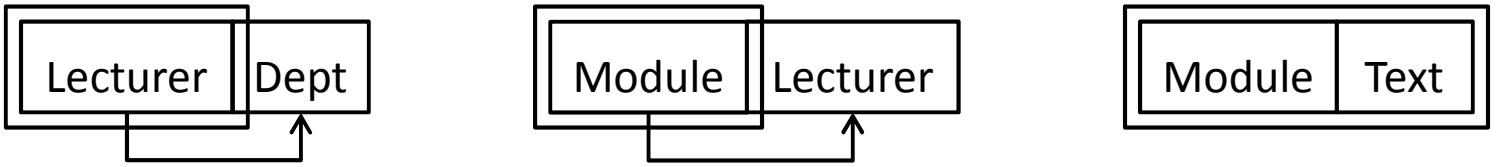
Lecturer	Dept
L1	D1
L2	D1
L3	D2
L4	D2

3NFb

Module	Lecturer
M1	L1
M2	L1
M3	L2
M4	L3
M5	L4

2NFb

Module	Text
M1	T1
M1	T2
M2	T1
M2	T3
M3	T4
M4	T1
M4	T5
M5	T6



Problems Resolved in 3NF

- Problems resolved in 3NF
  - INSERT – We can now add Lecturers who don't teach any modules
  - UPDATE – We need only change a single row to update the department for L1
  - DELETE – We can delete M3 while preserving L2

3NFa

Lecturer	Dept
L1	D1
L2	D1
L3	D2
L4	D2

3NFb

Module	Lecturer
M1	L1
M2	L1
M3	L2
M4	L3
M5	L4

Normalisation and Design

- Normalisation is related to Database design
  - A database should normally be in 3NF at least
  - If your design leads to a non-3NF database, then you might want to revise it
- When you find you have a non-3NF database
  - Identify the FDs that are causing a problem
  - Decide whether they will lead to any insert, update, or delete anomalies
  - Try to remove them

Informal Summary

- Informally (not formally) removing dependencies:

- 1NF: Ensure all data is atomic, by adding rows
  - Rows also must be unique and row and column order must not matter
- 2NF: Remove dependencies upon only part of a key
- 3NF: Remove transitive dependencies upon key data
  - Including upon non-key data
- BCNF: Remove dependencies for key data
  - Tomorrow

Coursework

- Coursework 1 is out
  - Your tutorial with your personal tutor should help you to do it
    - **I have gained permission to delay the submission date to 21st March for this reason – submit early if you can**
  - Muhammed Ahmed's ER-diagram tool should help
  - Use the labs to ask questions!
- Coursework 2 will come out soon
  - Deadline 2nd April – not long after CW1 now!
  - Select statements and PHP (next week's lectures)

Next Lecture

- More Normalisation
  - Another example
  - Lossless decomposition
  - BCNF

- Further Reading
  - The Manga Guide to Databases, Chapter 3
  - Database Systems, Chapters 14 and 15

Normalisation II

G51DBS Database Systems

Jason Atkin

Informally – to help remember ONLY

- **"[Every] non-key [attribute] must provide a fact about the key, the whole key, and nothing but the key.“ [Bill Kent, as quoted on Wikipedia ;)]**

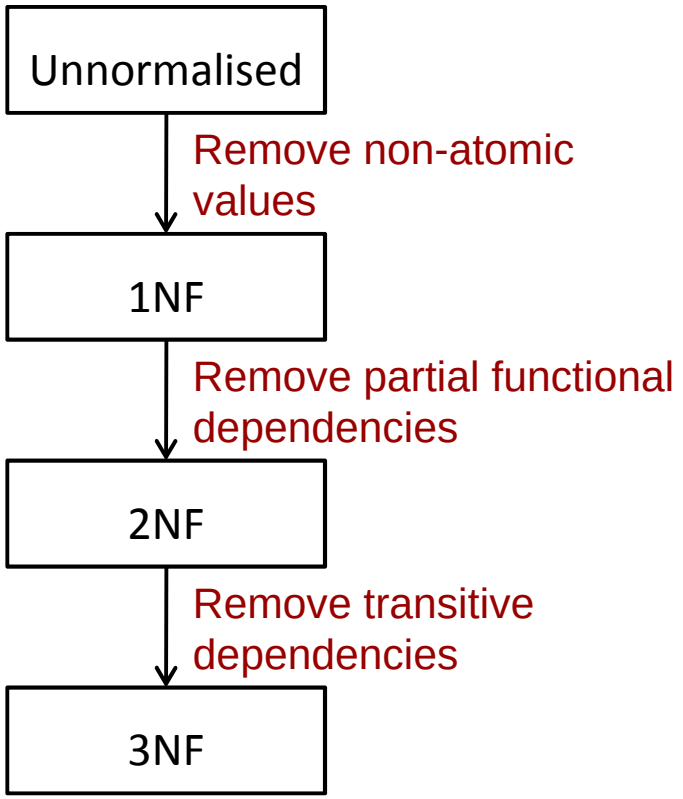
- 1NF: **A key exists**, cell contents atomic
- 2NF: Non-key attributes depend on the **whole** key
- 3NF: Non-key attributes depend on **nothing but** the key
- BCNF: **All attributes** depend on nothing but the key

This Lecture

- Review and example of 1NF to 3NF
- More Normalisation
  - Lossless decomposition
  - BCNF
- Denormalisation
- Further Reading
  - The Manga Guide to Databases, Chapter 3
  - Database Systems, Chapters 14 and 15

Last Lecture

- Normalisation
  - Data Redundancy
  - Functional Dependencies
  - Normal Forms
  - First, Second and Third Normal Forms
- Further Reading
  - The Manga Guide to Databases, Chapter 3
  - Database Systems, Chapter 14



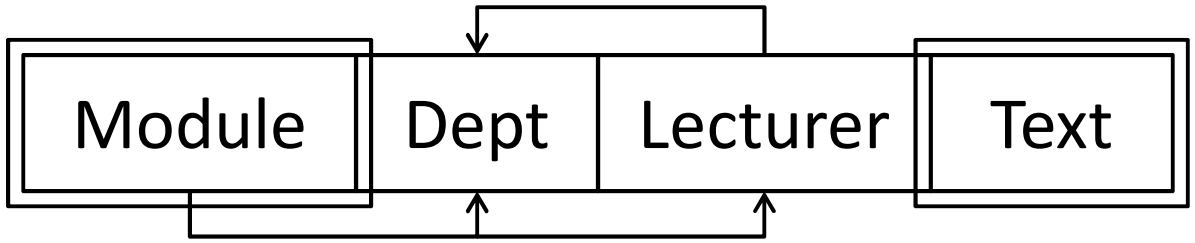
Last Lecture

- Partial FDs
  - Some set of **non-key** attributes B is **dependent on a subset** of a candidate key A

{Module} → {Dept, Lecturer}

- Transitive FDs
  - Some set of **non-key** attributes C is **transitively dependent** on a candidate key A

{Module} → {Lecturer}
→ {Dept}



Example

Unnormalised (not even 1NF)

orderID	orderDate	customerID	cAddress	stockNos	stockQuant	stockPrices
100152	12-11-10	C1035	5 Ar...	10,98,14	1,10,2	9.99,4.99...
100236	19-11-10	C1011	7 Be...	59,13,...	1,1,2,1,1,...	0.99,3.99...
101562	01-02-11	C2693	Flat 1a...	7,45,9,...	10,10,1,...	2.99,3.49...
102648	26-02-11	C1011	7 Be...	59,56,...	1,5,3,4,6,...	0.99,4,9...

orderID	orderDate	customerID	cAddress	stockNos	stockQuant	stockPrices

Currently the only candidate key is {orderID}

Example

1NF

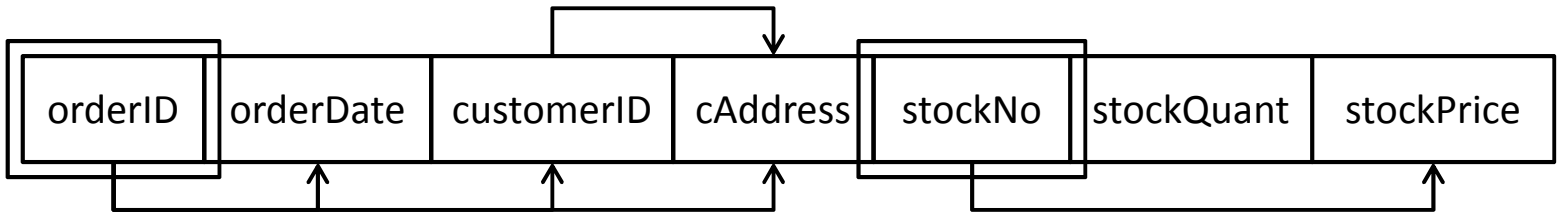
orderID	orderDate	customerID	cAddress	stockNo	stockQuant	stockPrice
100152	12-11-10	C1035	5 Ar...	10	1	9.99
100152	12-11-10	C1035	5 Ar...	98	10	4.99
100152	12-11-10	C1035	5 Ar...	14	2	6.99

The only candidate key is {orderID, stockNo}

Example

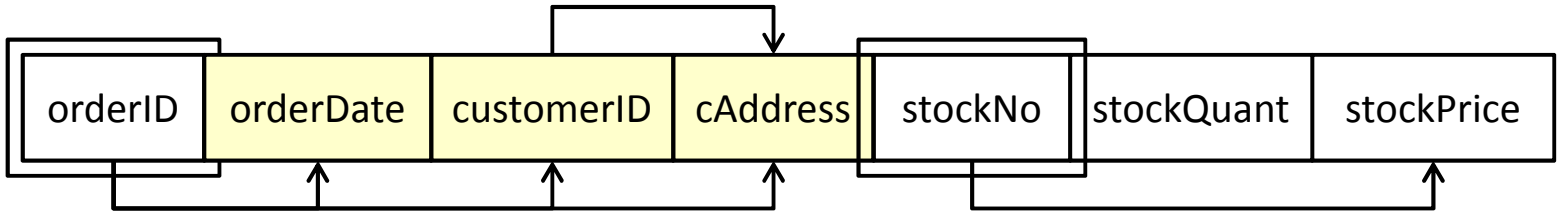
1NF

orderID	orderDate	customerID	cAddress	stockNo	stockQuant	stockPrice
100152	12-11-10	C1035	5 Ar...	10	1	9.99
100152	12-11-10	C1035	5 Ar...	98	10	4.99
100152	12-11-10	C1035	5 Ar...	14	2	6.99



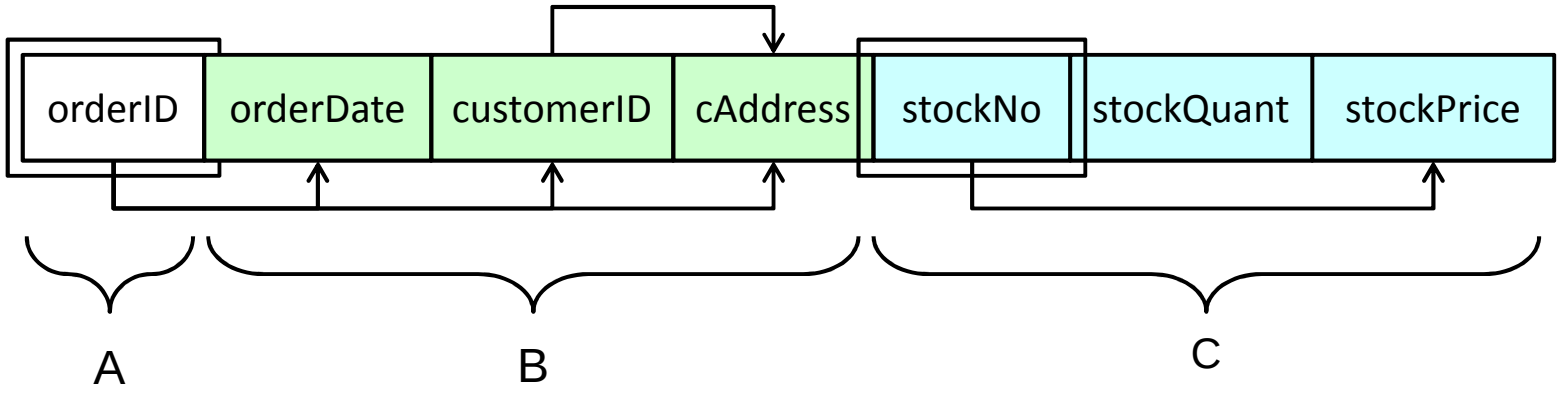
Example

- This database does not adhere to 2NF
  - There are non-key attributes partially dependent on a candidate key (i.e. dependent on partial key)

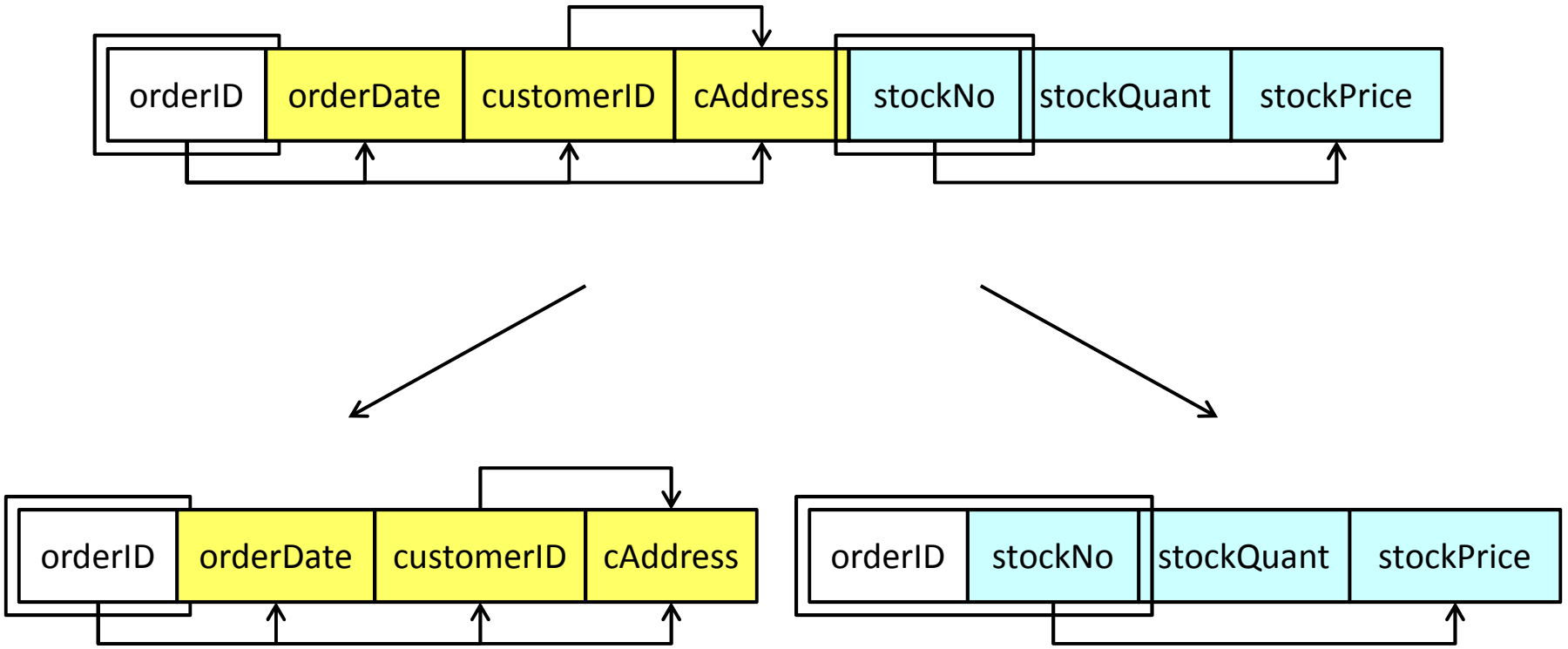


Example

- To remove the FD A $\rightarrow$ B, where C is all other attributes
  - Create two new relations A $\cup$ B and A $\cup$ C

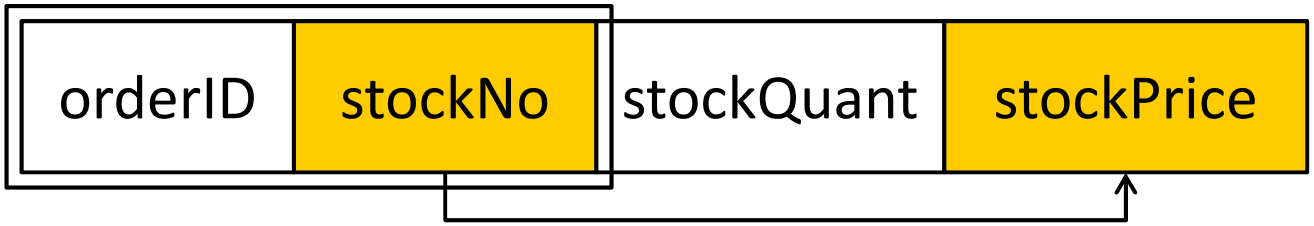


Example


orderID
orderDate
customerID
cAddress
stockNo
stockQuant
stockPrice
orderID
orderDate
customerID
cAddress
orderID
stockNo
stockQuant
stockPrice

Example

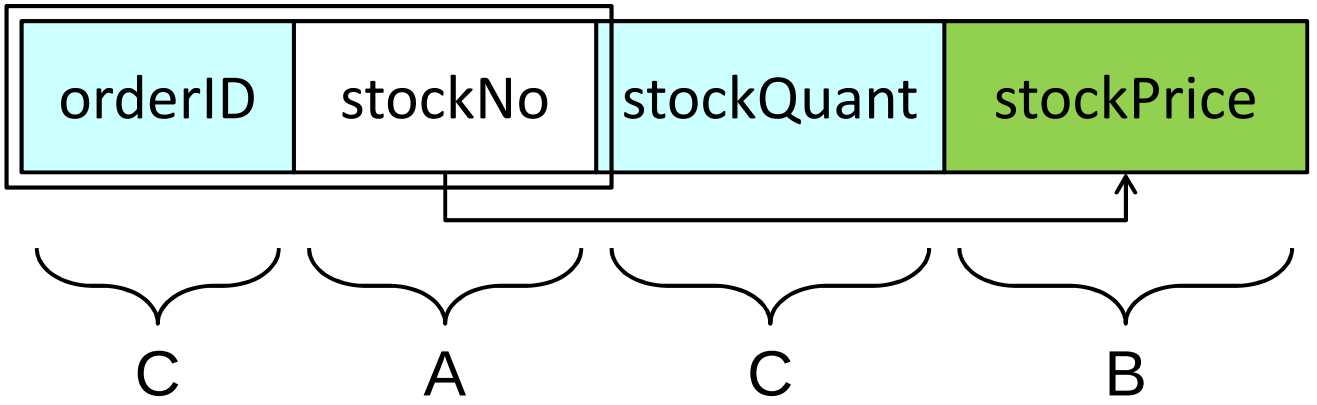
- One of the relations is still not in 2NF
  - {stockPrice} is partially dependent on {orderID, stockNo}



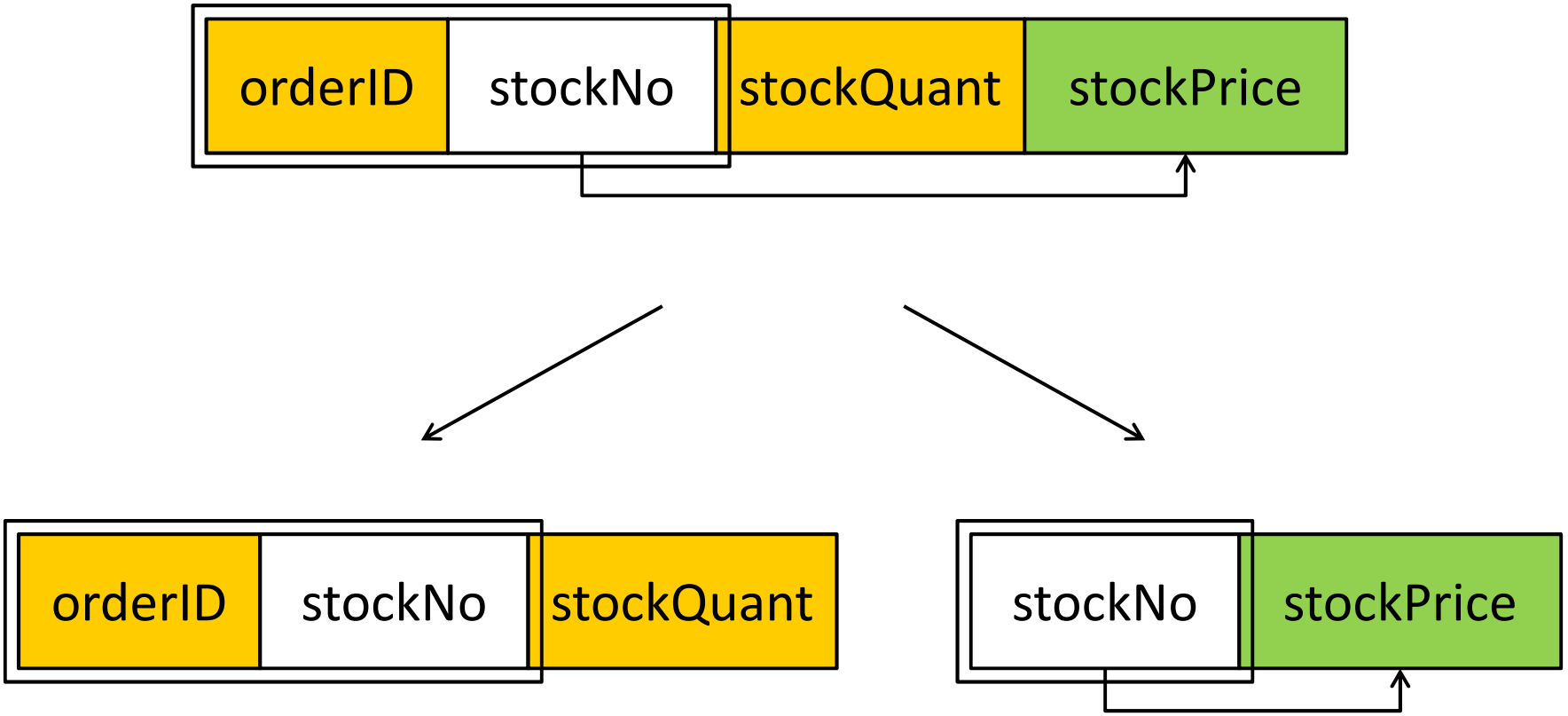
Example

- One of the relations is still not in 2NF
  - As before, we need to create two new relations

    $A \cup B$ and $A \cup C$

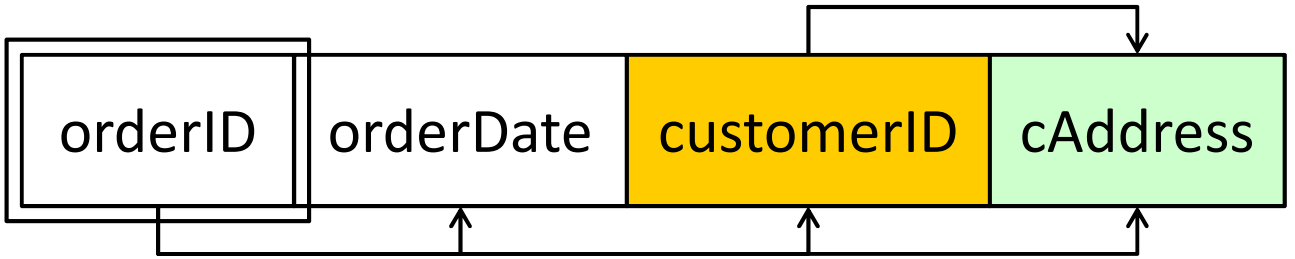


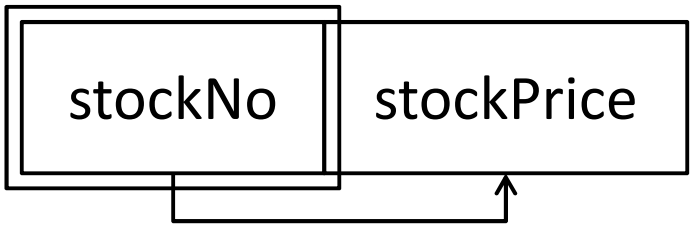
Example


orderID
stockNo
stockQuant
stockPrice
orderID
stockNo
stockQuant
stockNo
stockPrice

Example

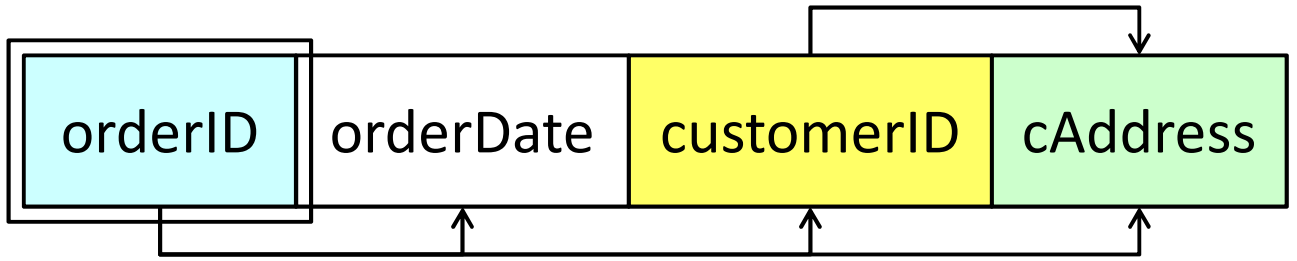
- This database is now in 2NF, but it isn't in 3NF
  - A transitive functional dependency exists…





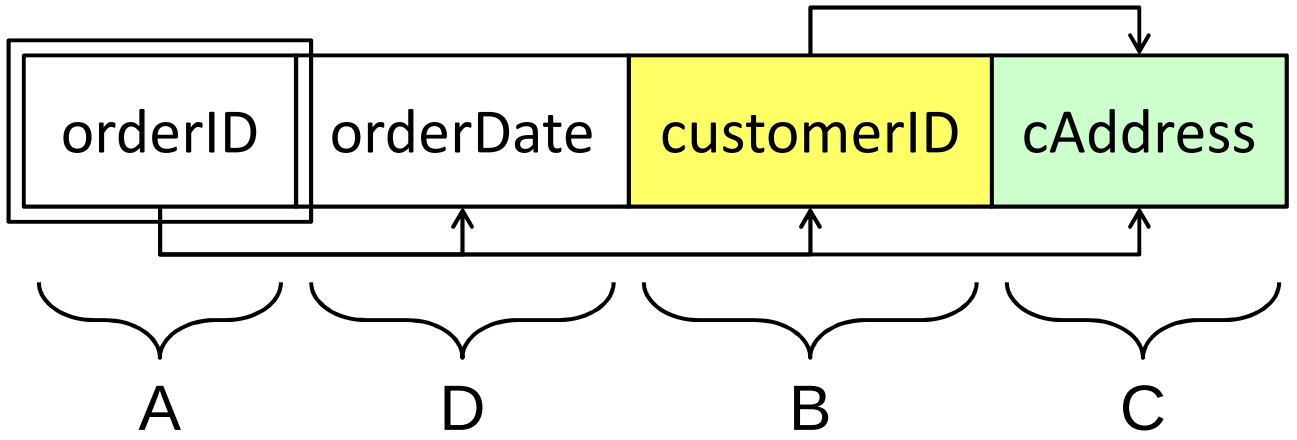
Example

- This relation is not in 3NF
  - {cAddress} is transitively dependent on {orderID} via {customerID}

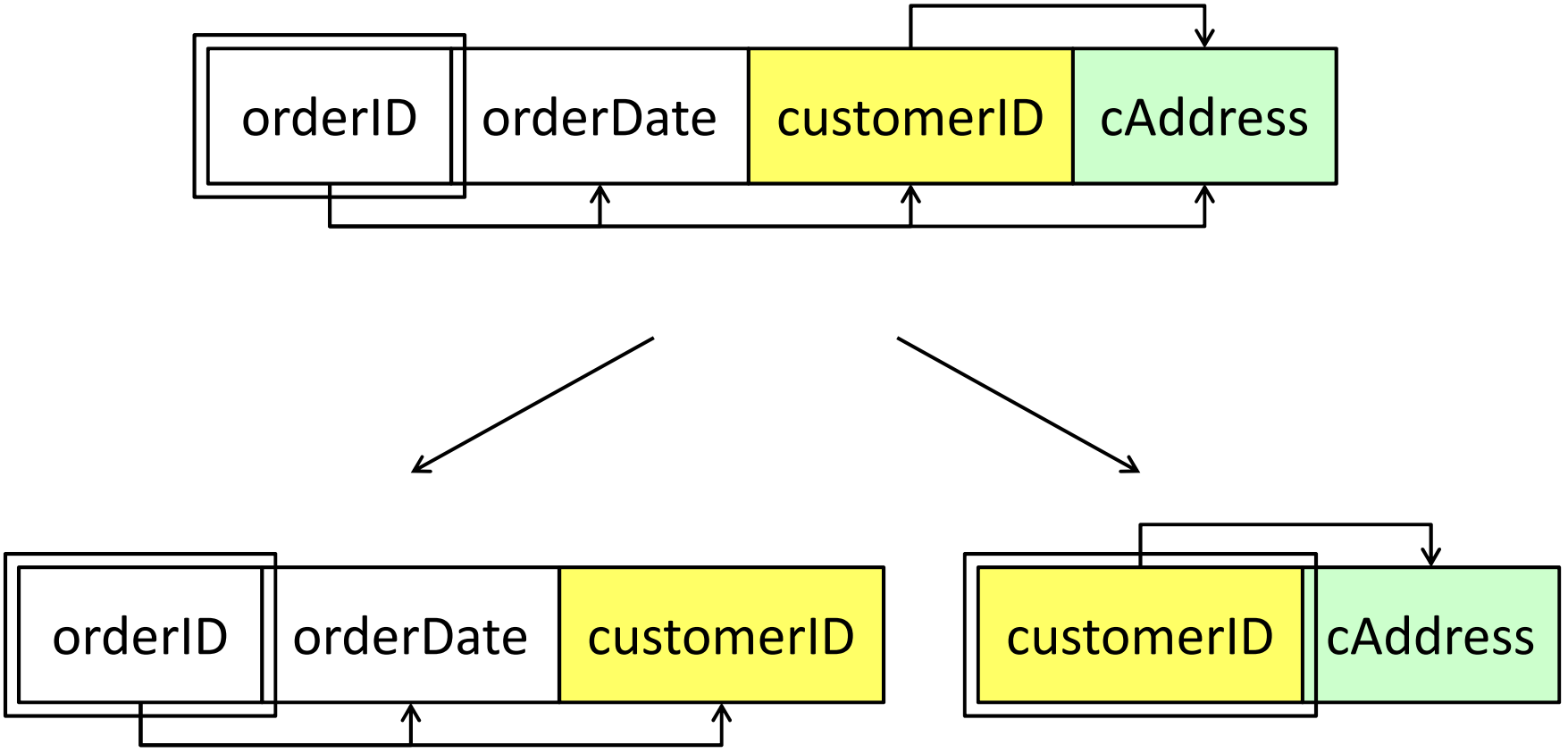


Example

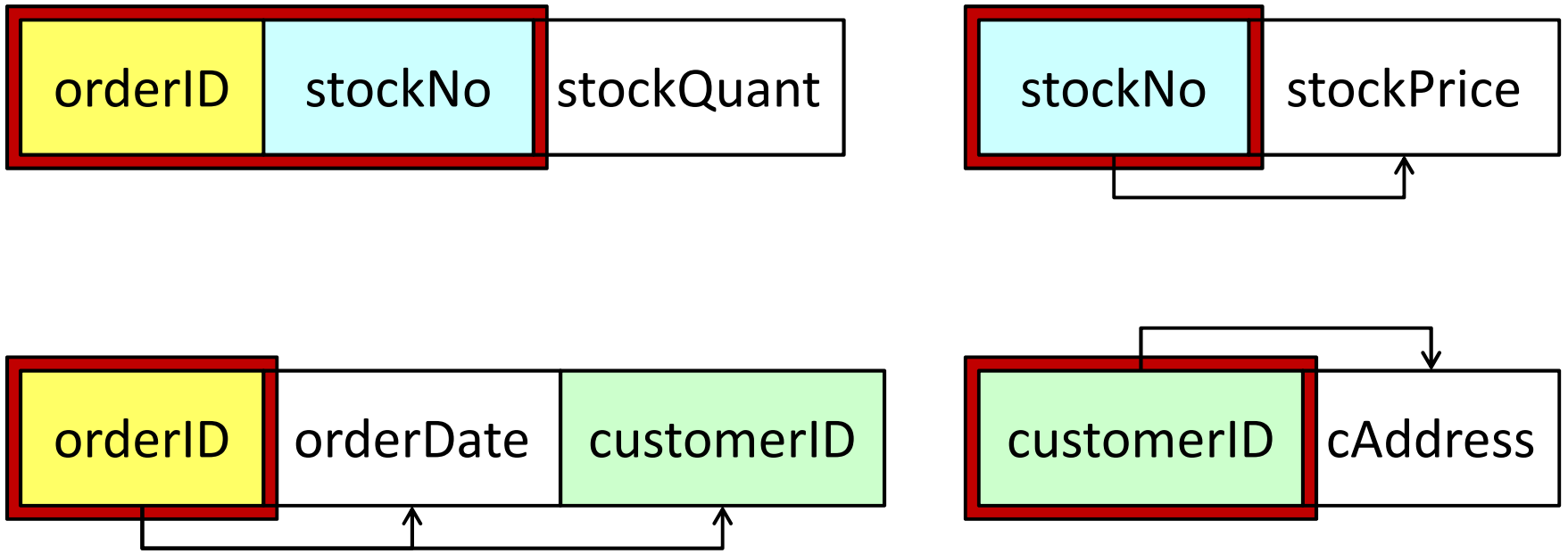
- To remove the Transitive FD A $\rightarrow$ B $\rightarrow$ C, where D is all other attributes
  - Create two new relations A $\cup$ B $\cup$ D and B $\cup$ C



Example


orderID
orderDate
customerID
cAddress
orderID
orderDate
customerID
customerID
cAddress

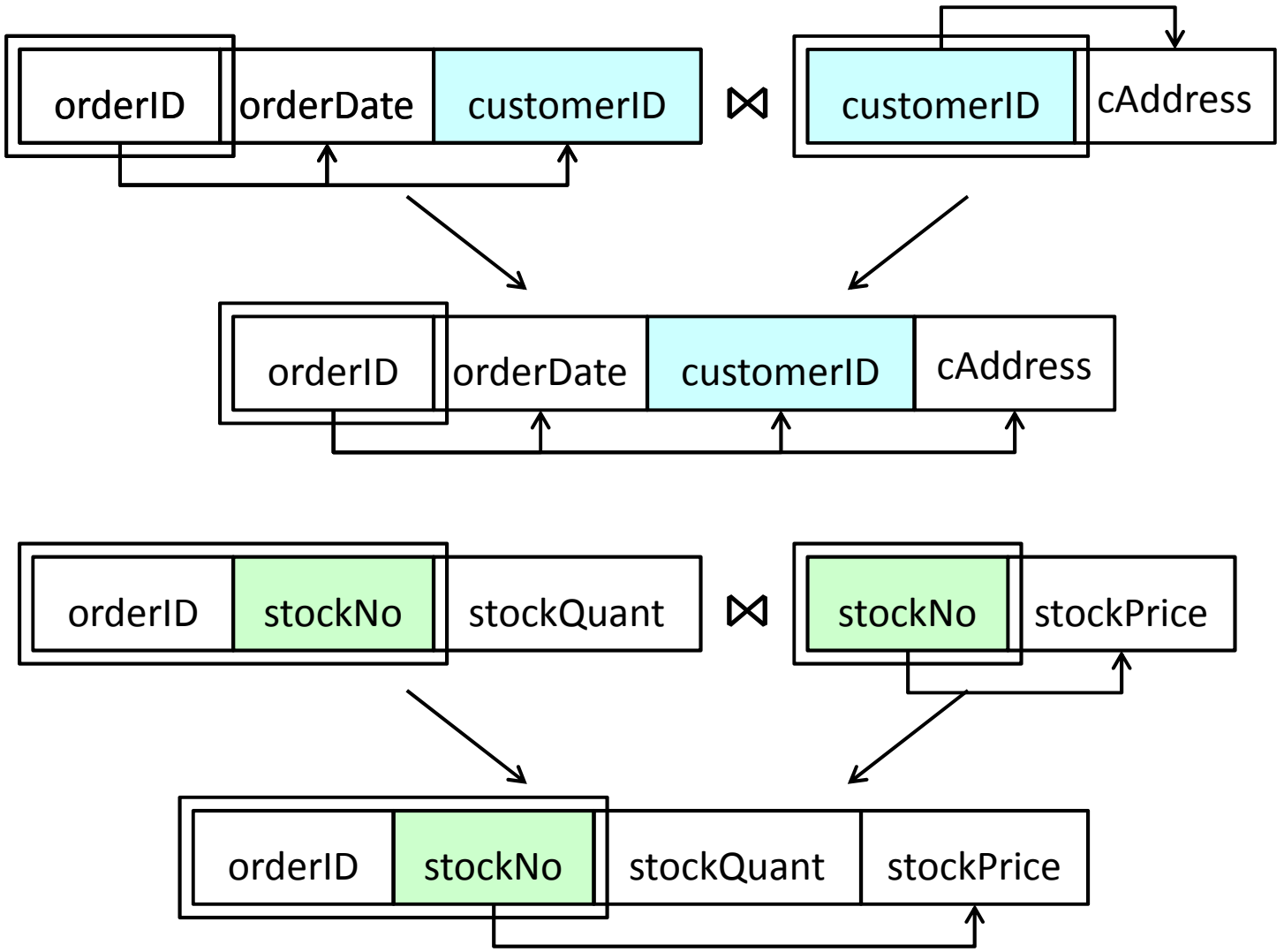
3NF Database



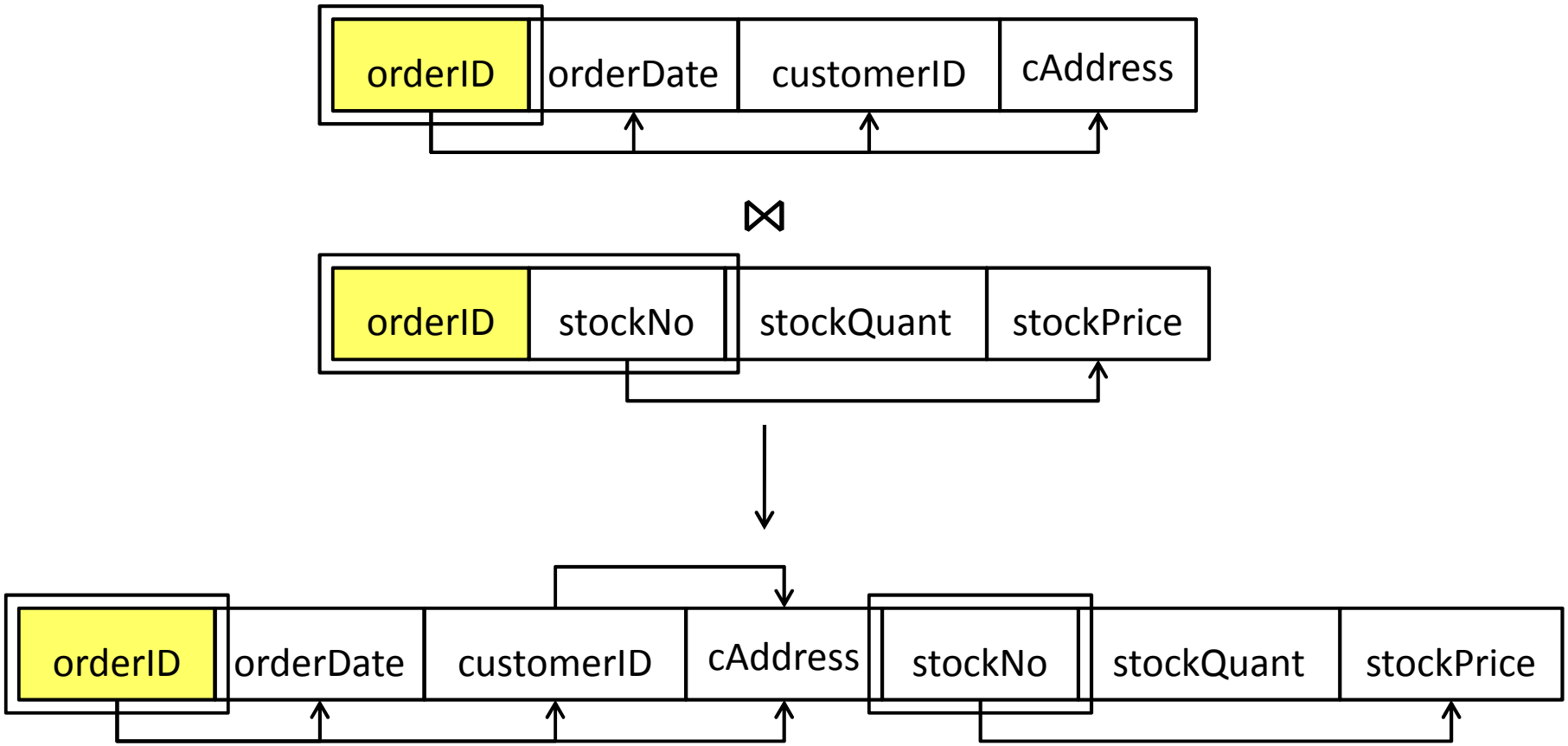
Lossless Decomposition

- Decomposition of tables is lossless if we can recover the original relation through a join
- A natural join is the most convenient way to do this, although most joins will work
- Lossless decomposition ensures that we haven't removed any data from our database
- All data can be retrieved again using joins if required

Lossless Decomposition

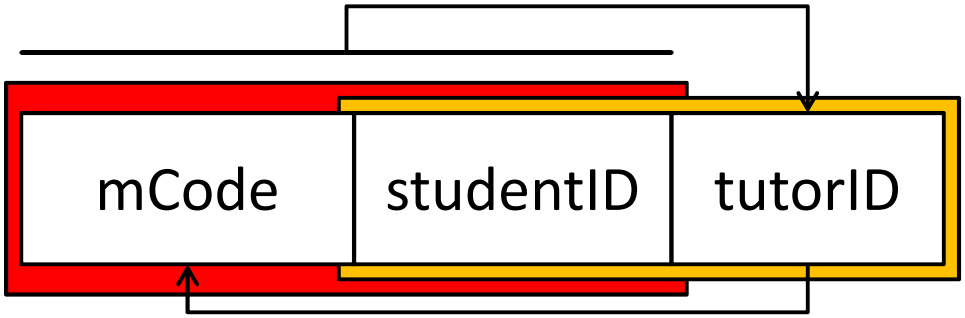


Lossless Decomposition



Boyce-Codd Normal Form

- Let's consider extending our Enrolment 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



Problems with 3NF

Enrolment

mCode	studentID	tutorID
G51DBS	109684	T001
G51PRG	108348	T002
G51IAI	110798	T003
G51DBS	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

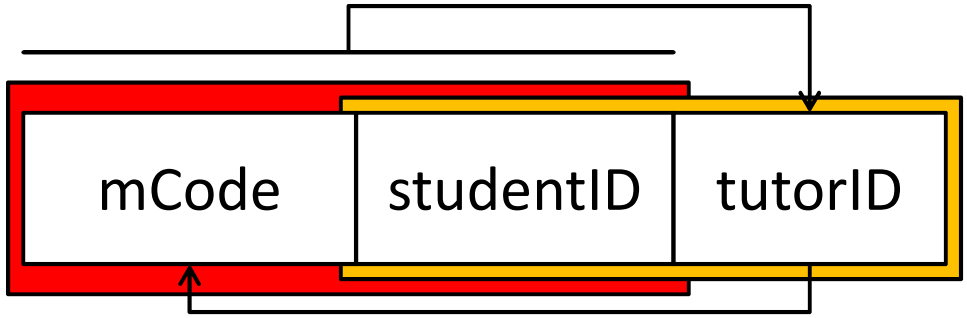
Boyce-Codd Normal Form

- 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 (super) key.

- The same as 3NF except in 3NF we only worry about non-key Bs
- i.e. with BCNF, any **key** B can only be determined by an entire (super) key
  - 3NF ignored key cols
- If there is only one candidate key then 3NF and BCNF are the same

Example

- The enrolment table is in 3NF but not BCNF
  - {tutorID} is not a candidate key, however the FD {tutorID} $\rightarrow$ {mCode} exists
    - Remember: tutor is on only one module
  - {mCode, studentID} $\rightarrow$ {tutorID} is ok because {mCode, studentID} is a super-key (contains a candidate key)



Normalising to BCNF

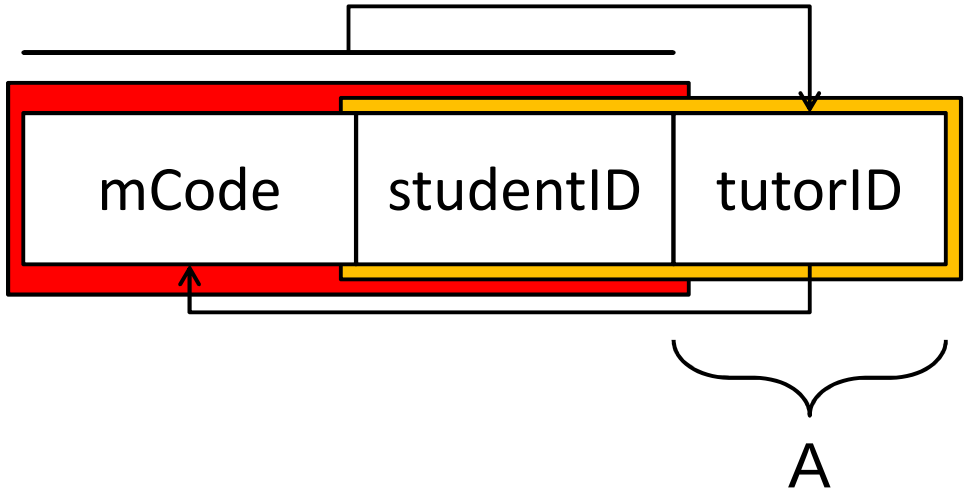
- Suppose we have a relation R with schema S and the FD $A \rightarrow B$ that violates BCNF

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

- To normalise to BCNF we create two new relations
  - **$A \cup C$**
  - **$A \cup B$**

Normalising to BCNF

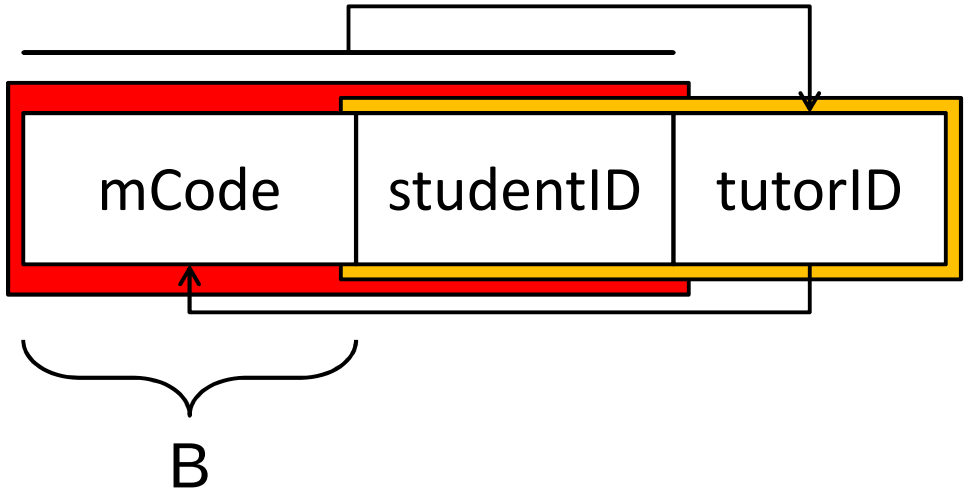
- We need to remove FD A $\rightarrow$ B to convert the relation into BCNF



A – The determinant of the functional dependency

Normalising to BCNF

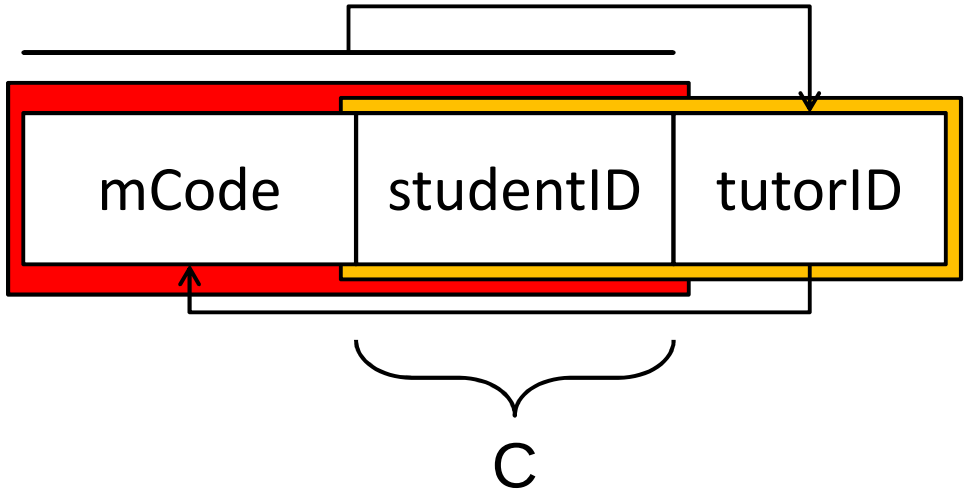
- We need to remove FD A $\rightarrow$ B to convert the relation into BCNF



B – The dependent attributes of the functional dependency

Normalising to BCNF

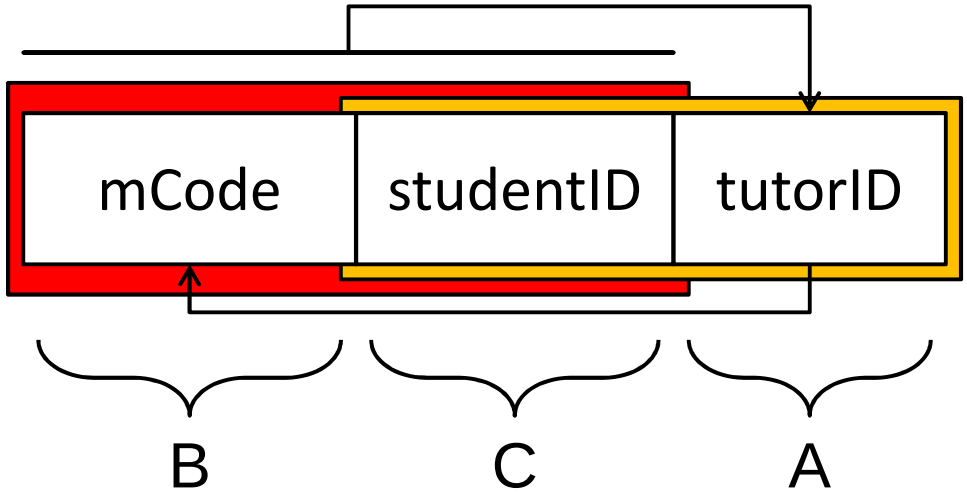
- We need to remove FD A $\rightarrow$ B to convert the relation into BCNF



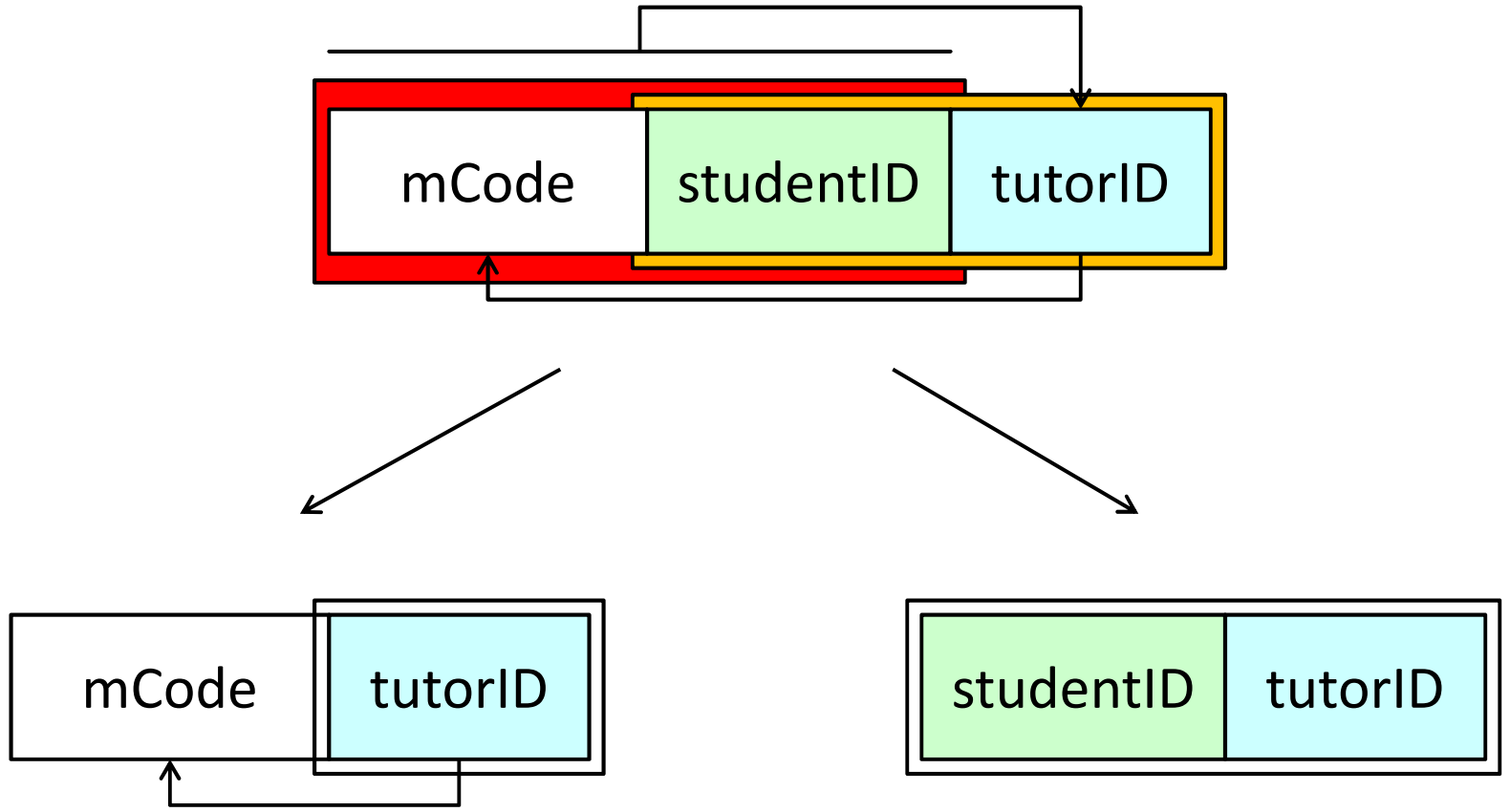
C – All other attributes

Normalising to BCNF

- To convert to BCNF, create two new relations $A \cup C$ and $A \cup B$



Normalising to BCNF



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

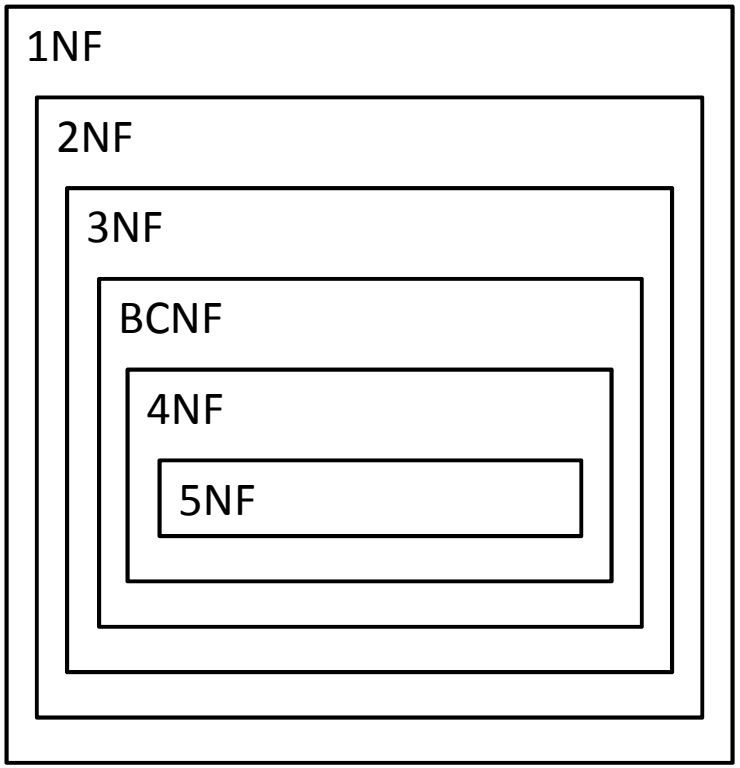
Decomposition Properties

- 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

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?) it is 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

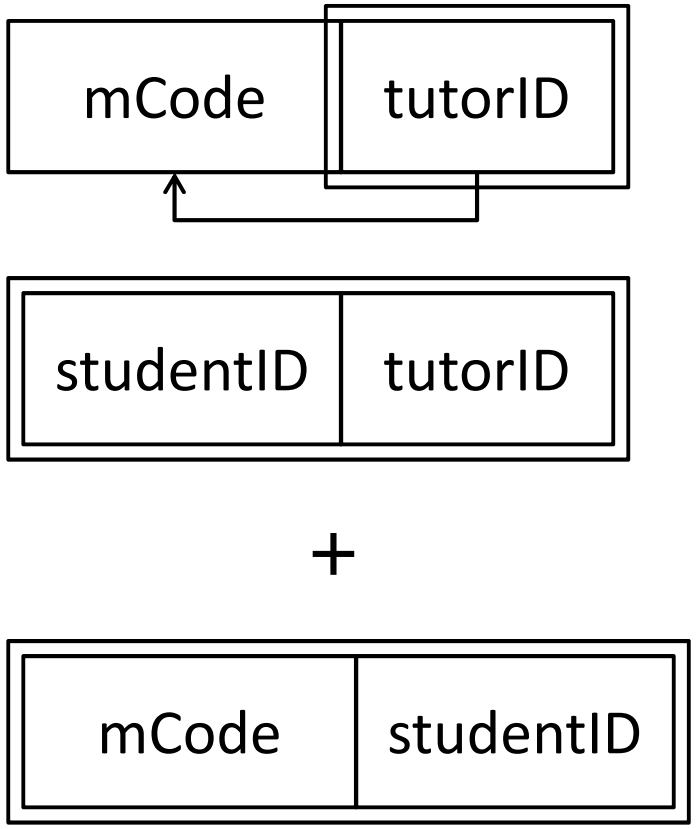
Number	Street	Postcode

Address2

PostCode	City

Denormalisation

- Sometimes creating redundant data makes INSERTs, UPDATEs and DELETEs more difficult, but avoids joins
- E.g. Realistically in our Enrolment table, we are probably going to often search for student "Enrolments"



Next two lectures

- PHP
  - Variables
  - Arrays
  - IF...ELSE statements
  - Loops
  - Connecting to MySQL
  - Formatting select results
- Further reading
  - W3Schools online tutorials at http://www.w3schools.com/php/