Normalization II

Gary KL Tam

Department of Computer Science
Swansea University

This Lecture

- Review of 1NF to 3NF
- Normalisation Example
- Lossless Deccomposition
- BCNF
- Denormalisation

Normal Forms (Review)

First Normal Form (1NF)

- A relation is in 1NF if all data values are atomic

Second Normal Form (2NF)

A relation is in 2NF if it is

- in 1NF and
- every non-key attribute is fully functionally dependent on the primary key (i.e. no partial dependency)

Third Normal Form (3NF)

A relation is in third normal form (3NF) if

- in 2NF and
- no transitive dependency - a functional dependency between two (or more) non-key attributes.

Review

Normalisation

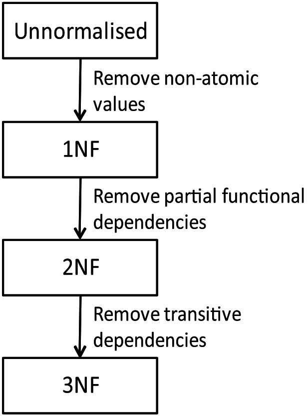
- Redundancy
- Functional Dependencies

review

$R\ (\underline{A,\ B},\ C,\ D)$

- Partial FD: $B \rightarrow C$
- Transitive FD: $C \rightarrow D$

- First, Second and Third Normal Forms
- C, D are non-key attributes



Another Example

Unnormalised

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...

Relation R - order_stock, Candidate key: {orderID}

orderID	orderDate	customerID	cAddress	stockNos	stockQuant	stockPrices

Another Example

Split any non-atomic values
1NF

orderID	orderDate	customerID	cAddress	stockNos	stockQuant	stockPrices
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...
100236	19-11-10	C1011	7 Be...	59	1	0.99
100236	19-11-10	C1011	7 Be...	13	1	3.99
100236	19-11-10	C1011	7 Be...	4	2	3.49

...

Relation R - order_stock, Candidate key: {orderID, stockNo}

<u>orderID</u>	orderDate	customerID	cAddress	<u>stockNos</u>	stockQuant	stockPrices

Another Example - 1NF

1NF

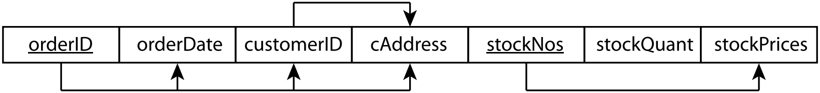
orderID	orderDate	customerID	cAddress	stockNos	stockQuant	stockPrices
100152	12-11-10	C1035	5 Ar...	10	1	9.99
100152	12-11-10	C1035	5 Ar...	98	10	4.99
100152	12-11-10	C1035	5 Ar...	14	2	6.99...
100236	19-11-10	C1011	7 Be...	59	1	0.99
100236	19-11-10	C1011	7 Be...	13	1	3.99
100236	19-11-10	C1011	7 Be...	4	2	3.49

. . .

Next:

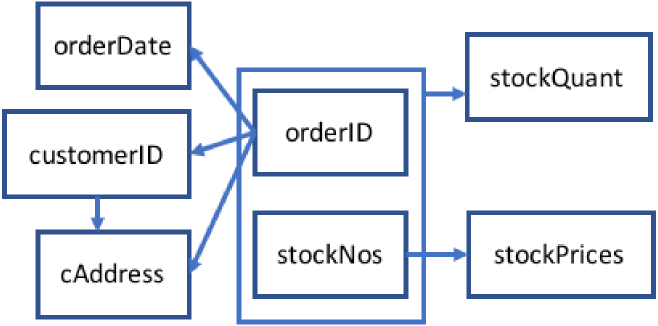
- Identify FDs (if not given), how?

Relation R - order_stock, Candidate key: {orderID, stockNo}

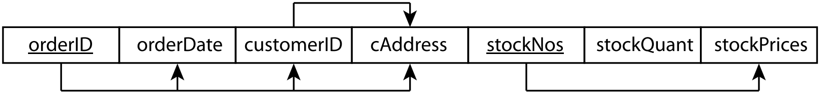


Functional Dependency Diagram

A reminder, a functional dependency diagram is drawn like this!



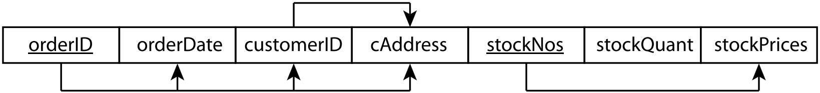
Another Example - 2NF



Think?

Is this relation in 2NF?

Another Example - 2NF



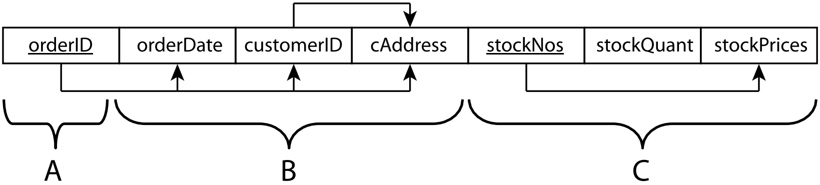
Think?

Is this relation in 2NF?

- ✗, this database does not adhere to 2NF
- There are non-key attributes partially dependent on a candidate key
- violating FD: orderID $\rightarrow$ orderDate, customerID, cAddress

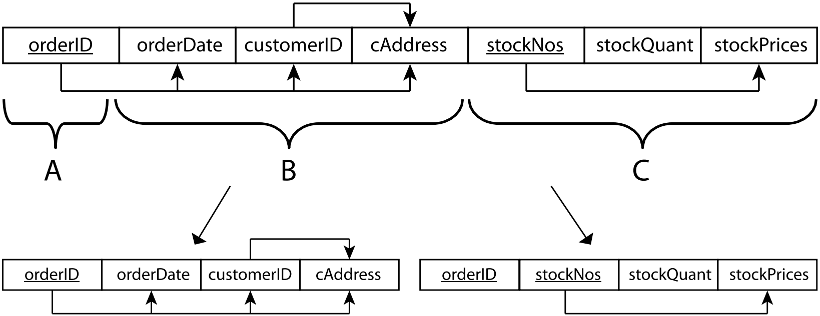
Another Example - 2NF

- To remove the violating FD: A $\rightarrow$ B, where C is all other attributes (i.e. C = R - A - B)
- Create two new relations A $\cup$ B and A $\cup$ C



Another Example - 2NF

- To remove the violating FD: $A \rightarrow B$, where C is all other attributes (i.e. $C = R - A - B$)
- Create two new relations $A \cup B$ and $A \cup C$



Think

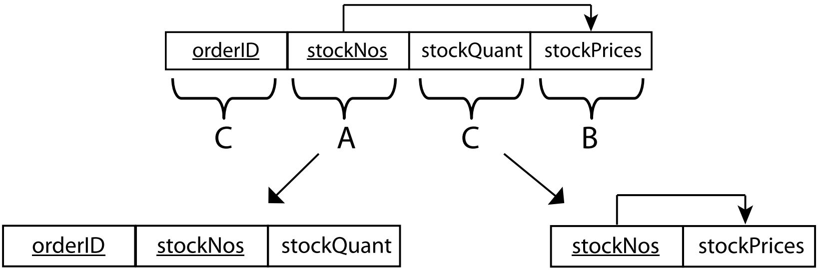
Are all tables in 2NF?

Another Example - 2NF

- Candidate key (orderID, stockNos)
- One of the relations is still not in 2NF
- There is still a partial dependency, which FD?

Another Example - 2NF

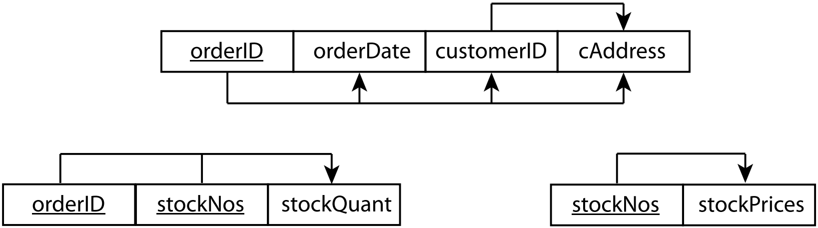
- Candidate key (orderID, stockNos)
- One of the relations is still not in 2NF
- There is still a partial dependency, which FD?
- violating FD: stockNo $\rightarrow$ stockPrice
- To remove violating FD: A $\rightarrow$ B, we create two new relations A $\cup$ B and A $\cup$ C again



Another Example - 3NF

Think

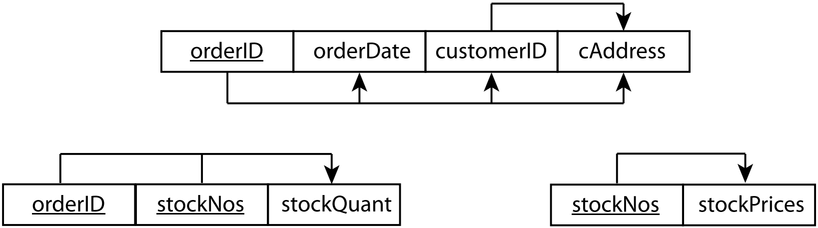
- These relations are now in 2NF.



Another Example - 3NF

Think

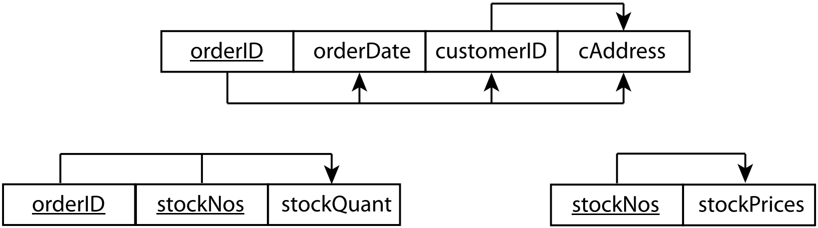
- These relations are now in 2NF.
- Are these relations in 3NF?



Another Example - 3NF

Think

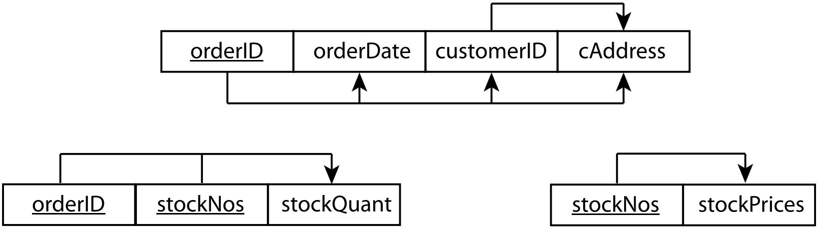
- These relations are now in 2NF.
- Are these relations in 3NF?
- Are there transitive functional dependencies between non-key attributes?



Another Example - 3NF

Think

- These relations are now in 2NF.
- Are these relations in 3NF?
- Are there transitive functional dependencies between non-key attributes?

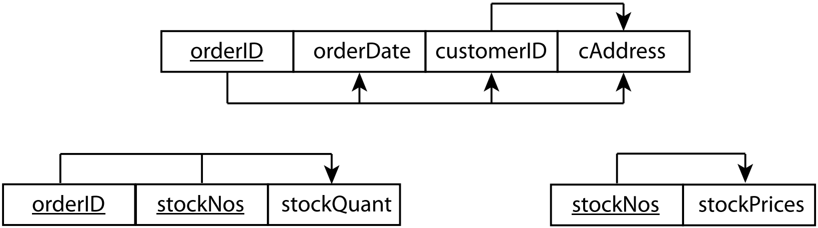


- ✗, one of the relations is NOT in 3NF. which one?

Another Example - 3NF

Think

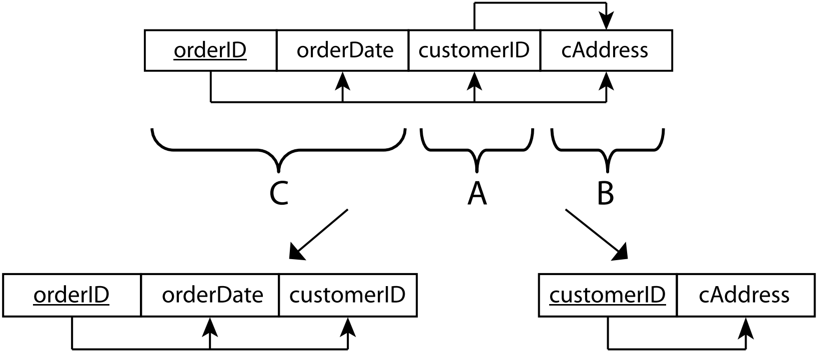
- These relations are now in 2NF.
- Are these relations in 3NF?
- Are there transitive functional dependencies between non-key attributes?



- ✗, one of the relations is NOT in 3NF. which one?
- There is a violating FD: customerID $\rightarrow$ cAddress

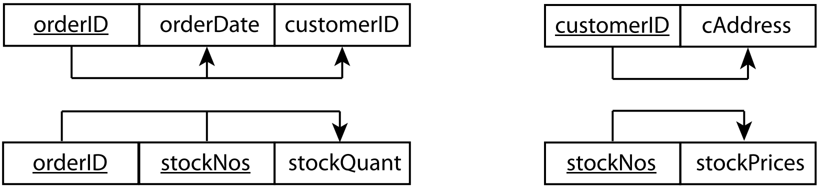
Another Example - 3NF

- To remove violating FD: A $\rightarrow$ B, we create two new relations A $\cup$ B and A $\cup$ C again



Another Example

Final 3NF Database



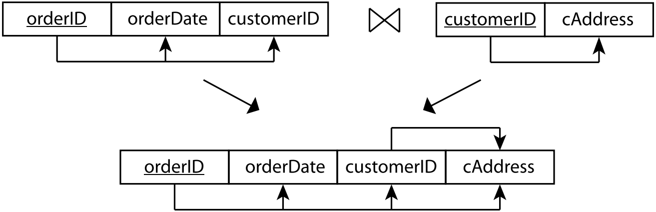
- Normalisation is related to Database Design
- A database should normally be in 3NF at least
- If your design leads to a database with lower NF than 3NF, then you might want to revise it.

Lossless Decomposition

- Decompsition of relations is lossless if we can recover the original relation through a join
- A natural join is the most convenient way to do so
- Lossless decomposition ensures that we haven't remove any data from our database
- All data can be retrieved again using joins if required

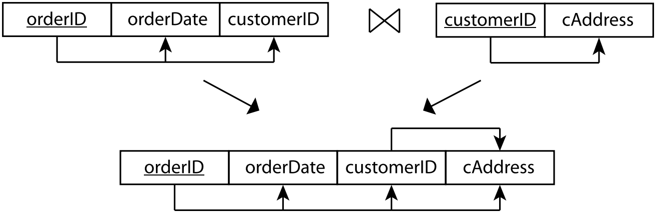
Lossless Decomposition

First 2 tables:

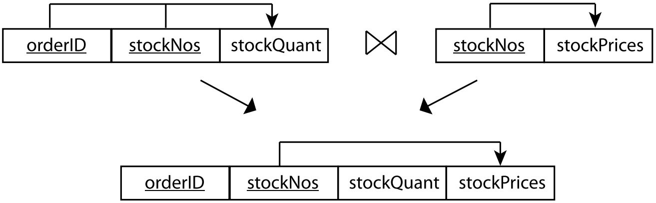


Lossless Decomposition

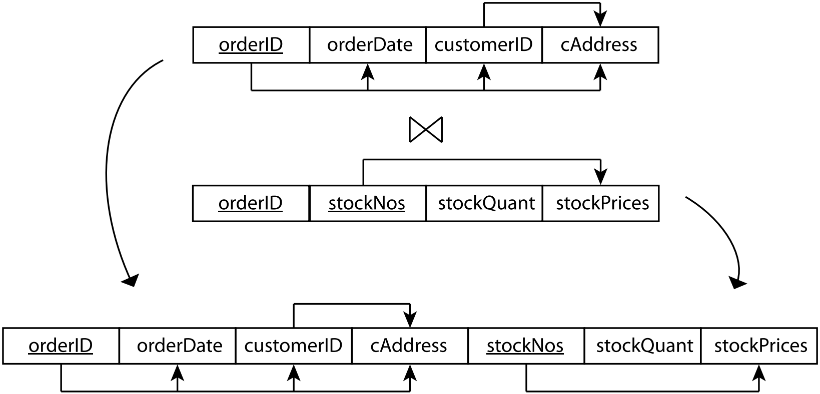
First 2 tables:



Next 2 tables:



Lossless Decomposition



"Lossy" Decomposition - an example

ID	**Name**	**Addr**
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak

↙ ↘

ID	**Name**
11	Pat
12	Jen
13	Jen

$\bowtie$

Name	**Addr**
Pat	1 Main
Jen	2 Pine
Jen	3 Oak

↘ ↙

"Lossy" Decomposition - an example

ID	**Name**	**Addr**
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak

ID	**Name**
11	Pat
12	Jen
13	Jen

$\bowtie$

Name	**Addr**
Pat	1 Main
Jen	2 Pine
Jen	3 Oak

ID	**Name**	**Addr**
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak
12	Jen	3 Oak
13	Jen	2 Pine

"Lossy" Decomposition - an example

ID	Name	Addr
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak

ID	Name
11	Pat
12	Jen
13	Jen

$\bowtie$

Name	Addr
Pat	1 Main
Jen	2 Pine
Jen	3 Oak

ID	Name	Addr
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak
12	Jen	3 Oak
13	Jen	2 Pine

- We lose 2 facts:
  - 12 lives ONLY at 2 Pine, and
  - 13 lives ONLY at 3 Oak
- Lossy decompositions yield more tuples

"Lossy" Decomposition - an example

ID	Name	Addr
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak

ID	Name
11	Pat
12	Jen
13	Jen

$\bowtie$

Name	Addr
Pat	1 Main
Jen	2 Pine
Jen	3 Oak

ID	Name	Addr
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak
12	Jen	3 Oak
13	Jen	2 Pine

- We lose 2 facts:
  - 12 lives ONLY at 2 Pine, and
  - 13 lives ONLY at 3 Oak
- Lossy decompositions yield more tuples
- Tuples aren't lost; information is.

"Lossy" Decomposition - an example

ID	Name	Addr
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak

ID	Name
11	Pat
12	Jen
13	Jen

$\bowtie$

Name	Addr
Pat	1 Main
Jen	2 Pine
Jen	3 Oak

ID	Name	Addr
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak
12	Jen	3 Oak
13	Jen	2 Pine

- We lose 2 facts:
  - 12 lives ONLY at 2 Pine, and
  - 13 lives ONLY at 3 Oak
- Lossy decompositions yield more tuples
- Tuples aren't lost; information is.

Think in Normalisation?

When we split the table, what have we assumed?

"Lossy" Decomposition - an example

ID	Name	Addr
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak

ID	Name
11	Pat
12	Jen
13	Jen

$\bowtie$

Name	Addr
Pat	1 Main
Jen	2 Pine
Jen	3 Oak

ID	Name	Addr
11	Pat	1 Main
12	Jen	2 Pine
13	Jen	3 Oak
12	Jen	3 Oak
13	Jen	2 Pine

- We lose 2 facts:
  - 12 lives ONLY at 2 Pine, and
  - 13 lives ONLY at 3 Oak
- Lossy decompositions yield more tuples
- Tuples aren't lost; information is.

Think in Normalisation?

When we split the table, what have we assumed?

does *Name* $\rightarrow$ *Addr* hold?

Lossless Decomposition - a definition

Decomposing R into R1, R2 is lossless (i.e., losing no information) if and only if X contains a candidate key of either R1 or R2, where X is the set of common attributes of R1 and R2.

Example 1, Suppose ID→Name, Addr

Decomposing R(ID, Name, Addr) into
R1 (ID, Name) and R2 (ID, Addr)
Lossless: $X = \{ID\}$ contains a candidate key of R1 (or R2).

Example 2, Suppose ID→Name, Addr

Decomposing R(ID, Name, Addr) into
R1 (ID, Name) and R2 (Name, Addr)
Lossy: $X = \{Name\}$ is neither a candidate key of R1 nor of R2.
This decomposition, obviously, must not be performed!

Normal Forms (Review)

First Normal Form (1NF)

- A relation is in 1NF if all data values are atomic

Second Normal Form (2NF)

A relation is in 2NF if it is

- in 1NF and
- every non-key attribute is fully functionally dependent on the primary key (i.e. no partial dependency)

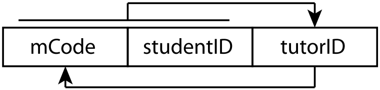
Third Normal Form (3NF)

A relation is in third normal form (3NF) if

- in 2NF and
- no transitive dependency - a functional dependency between two (or more) non-key attributes.

Another Example

- Let's consider extending a University Enrolment table
  - Each student will be assigned a PhD tutor for each module they are on
  - Tutors can have many students, but only help with one module
  - A module can have many tutors assigned to it



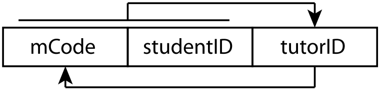
mCode, studentID $\rightarrow$ tutorID
tutorID $\rightarrow$ mCode

Candidate keys: (mCode, studentID) or (tutorID, studentID)

Think: Is this in 3NF?

Another Example

- Let's consider extending a University Enrolment table
  - Each student will be assigned a PhD tutor for each module they are on
  - Tutors can have many students, but only help with one module
  - A module can have many tutors assigned to it



mCode, studentID $\rightarrow$ tutorID
tutorID $\rightarrow$ mCode

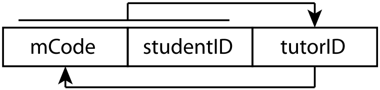
Candidate keys: (mCode, studentID) or (tutorID, studentID)

Think: Is this in 3NF?

✓, It is in 3NF. No partial dependencies. No transitive dependencies between non-key attributes

Another Example

- Let's consider extending a University Enrolment table
  - Each student will be assigned a PhD tutor for each module they are on
  - Tutors can have many students, but only help with one module
  - A module can have many tutors assigned to it



mCode, studentID $\rightarrow$ tutorID
tutorID $\rightarrow$ mCode

Candidate keys: (mCode, studentID) or (tutorID, studentID)

Think: Is this in 3NF?

✓, It is in 3NF. No partial dependencies. No transitive dependencies between non-key attributes
Note: tutorID $\rightarrow$ mCode is NOT partial FD. mCode is not a non-key attribute.

Still, there are problems with 3NF

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 Problem
  Can't add a tutor who isn't currently tutoring anyone
- UPDATE Problem
  Changing the module a tutor teaches involves multiple rows (e.g. T001)
- DELETE Problem
  If we remove student 110798, we no longer know that T003 is tutoring in G51IAI

Boyce-Codd Normal Form

Boyce-Codd Normal Form (BCNF)

- A relation is said to be in 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.
- **determinant**: the left hand side **LHS** of a FD : **A** $\rightarrow$ B.

Boyce-Codd Normal Form

Boyce-Codd Normal Form (BCNF)

- A relation is said to be in 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

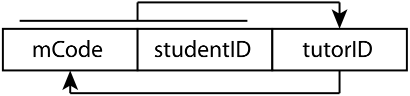
- In other words: every determinant in a non-trivial dependency is a (super) key.
- **determinant**: the left hand side **LHS** of a FD : **A** $\rightarrow$ B.

Examples of these terminologies

R(X,Y,Z) with candidate keys (X,Y), (Y,Z)

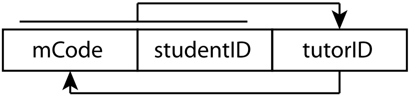
- $XY \rightarrow X$ trivial: RHS X is contained in LHS XY
- $XY \rightarrow Z$ LHS: XY contains a candidate key, ✓
- $Z \rightarrow X$ LHS: Z does NOT contains a candidate key, ✗
  Note: $Z \rightarrow X$ is not a partial FD!
  X is NOT a non-key attribute.

Example



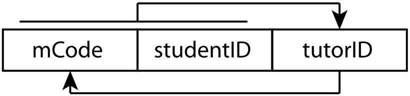
- Candidate keys: (mCode, studentID) or (tutorID, studentID)
- It is in 3NF, but not BCNF:

Example



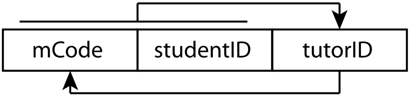
- Candidate keys: (mCode, studentID) or (tutorID, studentID)
- It is in 3NF, but not BCNF:
  - FD: mCode, studentID $\rightarrow$ tutorID

Example



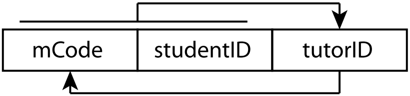
- Candidate keys: (mCode, studentID) or (tutorID, studentID)
- It is in 3NF, but not BCNF:
  - FD: mCode, studentID $\rightarrow$ tutorID ✓
    (mCode, studentID) contains a candidate key.

Example



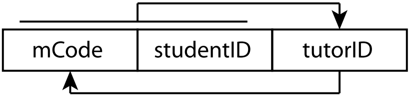
- Candidate keys: (mCode, studentID) or (tutorID, studentID)
- It is in 3NF, but not BCNF:
  - FD: mCode, studentID $\rightarrow$ tutorID ✓
    (mCode, studentID) contains a candidate key.
  - FD: tutorID $\rightarrow$ mCode

Example



- Candidate keys: (mCode, studentID) or (tutorID, studentID)
- It is in 3NF, but not BCNF:
  - FD: mCode, studentID $\rightarrow$ tutorID ✓
    (mCode, studentID) contains a candidate key.
  - FD: tutorID $\rightarrow$ mCode ✗a violating FD
    tutorID is not a candidate key

Example



- Candidate keys: (mCode, studentID) or (tutorID, studentID)
- It is in 3NF, but not BCNF:
  - FD: mCode, studentID $\rightarrow$ tutorID ✓
    (mCode, studentID) contains a candidate key.
  - FD: tutorID $\rightarrow$ mCode ✗a violating FD
    tutorID is not a candidate key
- Problems occur in 3NF relations that have two or more overlapping candidate keys.
- In this case, both share studentID

Muddiest Points / FAQ / Reminders

Boyce-Codd Normal Form (BCNF)

- A relation is said to be in BCNF if for every FD:**A** → **B** either
  - B is contained in A (the FD is trivial), or
  - A contains a candidate key of the relation

- In 2/3NF, we concern non-key **B**s.
- In BCNF, we concern determinant **A**s.
- For BCNF, watch out for overlapping candidate keys.
- AND, If there is only **one** candidate key, 3NF and BCNF **are the same**. BCNF another name: 3.5NF

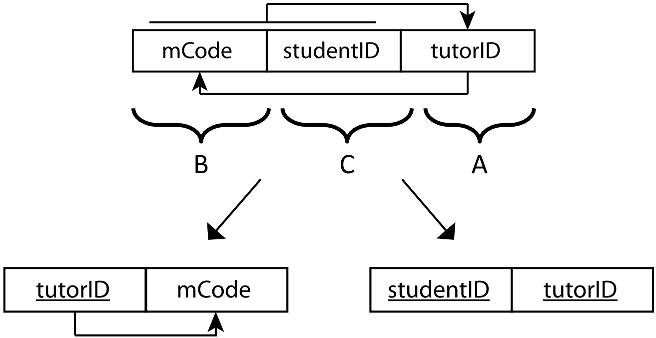
Quick reminders

R (A, B, C, D)

- Partial FD: B → C
- Transitive FD: C → D
- C, D are non-key attributes

3NF to BCNF

- To remove violating FD: A $\rightarrow$ B, we create two new relations A $\cup$ B and A $\cup$ C



- Note, FD: mCode,studentID $\rightarrow$ tutorID is NOT preserved.
- FD cannot be easily check in one table.

3NF vs BCNF - difference

3NF

- No redundancy problem ✗
- Lossless Join ✓
- Dependency Preservation ✓

BCNF

- No redundancy problem ✓
- Lossless Join ✓
- Dependency Preservation ✗

3NF vs BCNF - difference

3NF

- No redundancy problem ✗
- Lossless Join ✓
- Dependency Preservation ✓

BCNF

- No redundancy problem ✓
- Lossless Join ✓
- Dependency Preservation ✗

- Dependency preservation: It is desirable that FDs are preserved when splitting relations.

3NF vs BCNF - difference

3NF	BCNF
No redundancy problem ✗	No redundancy problem ✓
Lossless Join ✓	Lossless Join ✓
Dependency Preservation ✓	Dependency Preservation ✗

- Dependency preservation: It is desirable that FDs are preserved when splitting relations.
- Neither BCNF nor 3NF can guarantee all three properties.
  - If you decompose too far, can't easily check/enforce all FDs.
  - If you don't go far enough, can have redundancy.

3NF vs BCNF - difference

3NF

- No redundancy problem ✗
- Lossless Join ✓
- Dependency Preservation ✓

BCNF

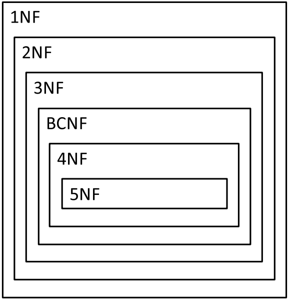
- No redundancy problem ✓
- Lossless Join ✓
- Dependency Preservation ✗

- Dependency preservation: It is desirable that FDs are preserved when splitting relations.
- Neither BCNF nor 3NF can guarantee all three properties.
  - If you decompose too far, can't easily check/enforce all FDs.
  - If you don't go far enough, can have redundancy.
- We must be satisfied with 2 out of 3.
- That is the reason why Industry often requires 3NF only.

Higher Normal Form

BCNF is as far as we 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
- Most textbooks copy Date's example, or do not give an example of 5NF because it is so rare and pathological.



Denormalisation

Normalisation

- Removes data redundancy
- Solves INSERT, UPDATE, and DELETE problems
- 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
- Occasionally it may be worth "considering denormaliisation" (**i.e. not often!**)

Very Caution!

- You should carry out normalisation before considering denormalisation!!! Do not skip the normalisation step!!!

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 lots of SELECTs that involve the joining of tables

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 lots of SELECTs that involve the joining of tables

Example: Address - application form

number	street	city	postcode

not normalised : postcode $\rightarrow$ city (SA2 $=$ Swansea)

number	street	postcode

postcode	city

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 lots of SELECTs that involve the joining of tables

Example: Address - application form

number	street	city	postcode

not normalised : postcode $\rightarrow$ city (SA2 = Swansea)

number	street	postcode

postcode	city

(how often is address updated? normalisation not preferred.)

Denormalisation

You might want to denormalise if

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

Example

studentID	tutorID

tutorID	mCode

+

mCode	studentID

(add Enrolments table)

12 Days of Denormalisation - Be Warned

1 st day: a really fast-running query.
2 nd day: 2 less tables.
3 rd day: 3 more indexes.
4 th day: 4 larger disks.
5 th day: 5 brand new reqs.
6 th day: 6 times the locking.
7 th day: 7 longer updates.
8 th day: 8 more requirements.
9 th day: 9 invalid rows.
10 th day: 10 delays deleting.
11 th day: 11 questionable queries.
12 th day: 12 lessons learned.

http://www.orafaq.com/wiki/Fun_stuff

Summary

- Normal forms provide a procedural mechanism for producing simple, stable relations.
- Normal forms give a good set of rules that can be followed, rather than designing relations without realising the problems that can occur.
- We are essentially trying to reduce relations to a very simple form.
- Often corresponds to our understanding of the data in the real world.