Normalisation II

Database Systems

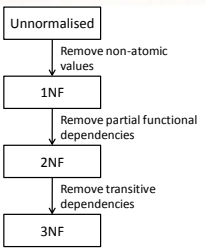
Michael Pound

This Lecture

- Review 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

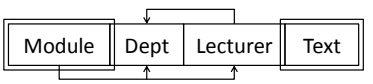
Last Lecture

- 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 non-key set of attributes B is dependent on a subset of a candidate key A
  - {Module} → {Dept, Lecturer}
- Transitive FDs
  - Some non-key set of attributes C is transitively dependent on a candidate key A
  - {Module} → {Lecturer} → {Dept}



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...

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
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

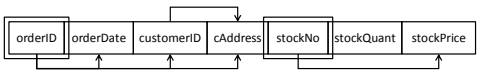
orderID	orderDate	customerID	cAddress	stockNo	stockQuant	stockPrice

The only candidate key is {orderID, stockNo}

Example

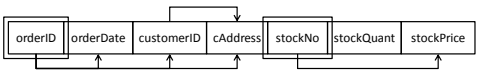
1NF

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
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



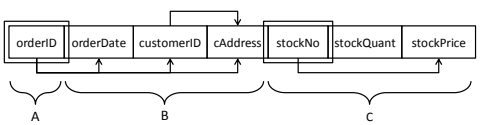
Example

- This database does not adhere to 2NF
  - There are non-key attributes partially dependent on a candidate key

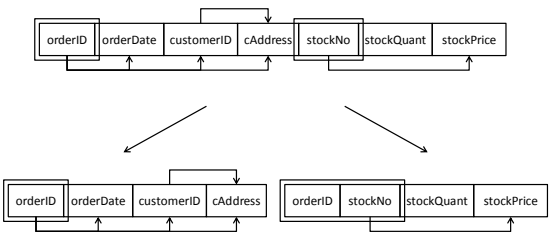


Example

- To remove the FD A → B, where C is all other attributes
  - Create two new relations A ∪ B and A ∪ C

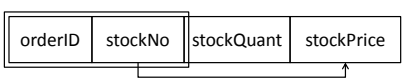


Example



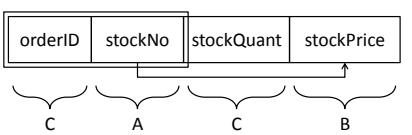
Example

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

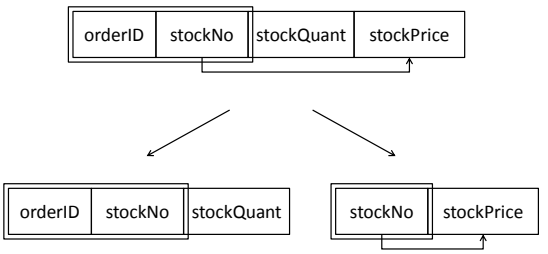


Example

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

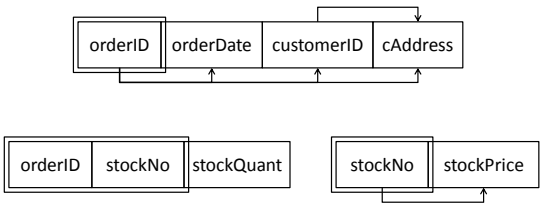


Example



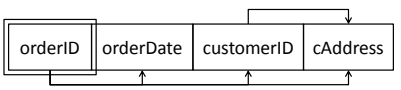
Example

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



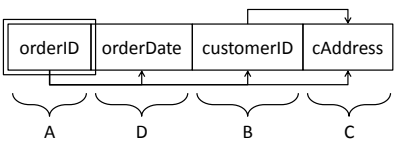
Example

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

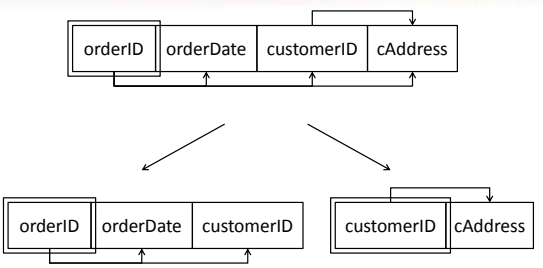


Example

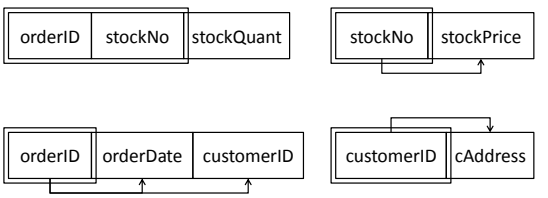
- To remove the Transitive FD $A \rightarrow B \rightarrow C$, where D is all other attributes
  - Create two new relations $A \cup B \cup D$ and $B \cup C$



Example



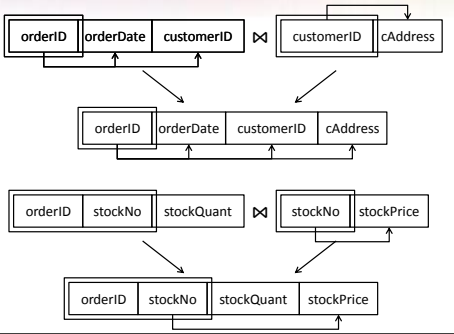
3NF Database



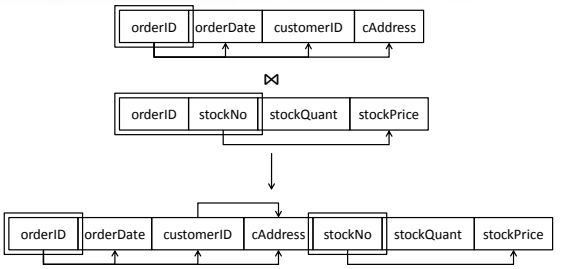
Lossless Decomposition

- 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

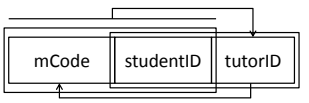


Lossless Decomposition



Boyce-Codd Normal Form

- Let's consider extending our Enrolment table from the University Database example
  - Each student will be assigned a PhD 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



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

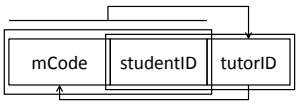
- INSERT Anomalies
  - Can't add a tutor who isn't currently tutoring anyone
- UPDATE Anomalies
  - Changing the module a tutor teaches is complicated and involves multiple rows
- DELETE Anomalies
  - If we remove student 110798, we no longer know that T003 is tutoring in G51IAI

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
- 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} → {mCode} exists
  - {mCode, studentID} → {tutorID} is ok because {mCode, studentID} is a super-key (contains a candidate key)



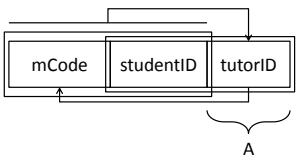
Normalising to BCNF

- Suppose we have a relation R with scheme S and the FD A → B that violates BCNF

  $A \cap B = \{\}$
- Let C = S - (A ∪ 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 ∪ C
  - A ∪ B

Normalising to BCNF

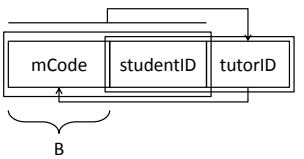
- We need to remove FD A → B to convert the relation into BCNF



A – The determinant of the functional dependency

Normalising to BCNF

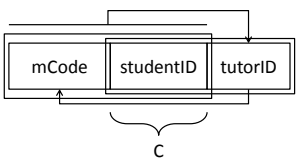
- We need to remove FD A → B to convert the relation into BCNF



B – The dependent attributes of the functional dependency

Normalising to BCNF

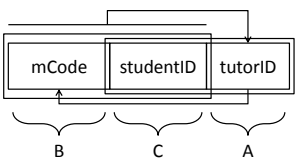
- We need to remove FD A → B to convert the relation into BCNF



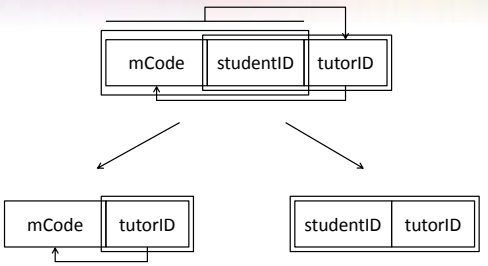
C – All other attributes

Normalising to BCNF

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



Normalising to BCNF



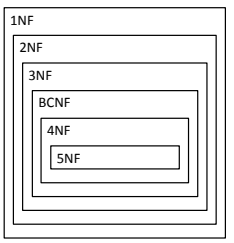
Note: We have lost the FD {mCode,studentID} → {tutorID}

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 relations up
- 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 lots of SELECTs that involve the joining of tables

Address

Number	Street	City	Postcode

Not normalised since {Postcode} → {City}

Address1

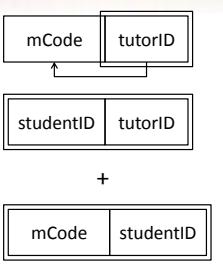
Number	Street	Postcode

Address2

PostCode	City

Denormalisation

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



Next Lecture

- Transactions
  - ACID Properties
  - COMMIT and ROLLBACK
- Recovery
  - System and Media Failures
- Concurrency
- Further reading
  - The Manga Guide to Databases, Chapter 5
  - Database Systems, Chapter 22