# Normalisation

Database Systems Lectures 11-12

Natasha Alechina

# In This Lecture

- Idea of normalisation
  - Functional dependencies
  - Normal forms
  - Decompositions
- 2NF, 3NF, BCNF

# 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
- We can normalise a relation by removing undesirable FDs

- A set of attributes, A, functionally determines another set, B, or: there exists a functional dependency between A and B ($A \rightarrow B$), 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.

# Example

- {ID, modCode} $\rightarrow$ {First, Last, modName}
- {modCode} $\rightarrow$ {modName}
- {ID} $\rightarrow$ {First, Last}

| ID | First | Last | modCode | modName |
|---|---|---|---|---|
| 111 | Joe | Bloggs | G51PRG | Programming |
| 222 | Anne | Smith | G51DBS | Databases |

# FDs and Normalisation

- We define a set of 'normal forms'
  - Each normal form has fewer FDs than the last
  - Since FDs 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 is a problem
  - We will also need a way to remove FDs

# Key attributes and superkeys

- We call an attribute a *key attribute* if this attribute is part of some candidate key. Alternative terminology is `prime’ attribute.
- We call a set of attributes a *superkey* if it includes a candidate key (or is a candidate key).

# Partial FDs and 2NF

- Partial FDs:
  - A FD, $A \rightarrow B$ is a partial FD, if some attribute of $A$ can be removed and the FD still holds
  - Formally, there is some proper subset of $A$,

  $C \subset A$, such that $C \rightarrow B$
- Let us call attributes which are part of some candidate key, key attributes, and the rest non-key attributes.

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.

# Second Normal Form

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 |

- 1NF 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

# Removing FDs

- Suppose we have a relation R with scheme 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
- It turns out that we can split R into two parts:
- R1, with scheme $C \cup A$
- R2, with scheme $A \cup B$
- The original relation can be recovered as the natural join of R1 and R2:
- R = R1 NATURAL JOIN R2

# 1NF to 2NF – 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 |

A $\underbrace{\qquad\qquad}_{B}$ C

2NFa

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

A, B where A $\rightarrow$ B is the `bad' dependency – violating 2NF

2NFb

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

A, C

# Problems Resolved in 2NF

- Problems in 1NF
  - INSERT – Can't add a module with no texts
  - UPDATE – To change lecturer for M1, we have to change two rows
  - DELETE – If we remove M3, we remove L2 as well
- In 2NF the first two are resolved, but not the third one

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
  - Can't add lecturers who teach no modules
- UPDATE anomalies
  - To change the department for L1 we must alter two rows
- DELETE anomalies
  - If we delete M3 we delete L2 as well

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, $A \to C$ is a transitive FD, if there is some set $B$ such that $A \to B$ and $B \to C$ are non-trivial FDs
  - $A \to B$ non-trivial means: $B$ is not a subset of $A$
  - We have

$$A \to B \to C$$

- 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
  - Alternative (simpler) definition: a relation is in 3NF if in every non-trivial fd A $\to$ B either B is a key attribute or A is a superkey.

# Third Normal Form

2NFa

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

- 2NFa is not in 3NF
  - We have the FDs

    {Module} $\rightarrow$ {Lecturer}

    {Lecturer} $\rightarrow$ {Dept}
  - So there is a transitive FD from the primary key {Module} to {Dept}

# 2NF to 3NF – Example

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 |

# Problems Resolved in 3NF

- Problems in 2NF
  - INSERT – Can't add lecturers who teach no modules
  - UPDATE – To change the department for L1 we must alter two rows
  - DELETE – If we delete M3 we delete L2 as well

- In 3NF all of these are resolved (for this relation – but 3NF can still have anomalies!)

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 so Far

- First normal form
  - All data values are atomic
- Second normal form
  - In 1NF plus no non-key attribute is partially dependent on a candidate key
- Third normal form
  - In 2NF plus no non-key attribute depends transitively on a candidate key (or, no dependencies of non-key on non-superkey)

# The Stream Relation

- Consider a relation, Stream, which stores information about times for various streams of courses
- For example: labs for first years
- Each course has several streams
- Only one stream (of any course at all) takes place at any given time
- Each student taking a course is assigned to a single stream for it

# The Stream Relation

| Student | Course | Time |
|---|---|---|
| John | Databases | 12:00 |
| Mary | Databases | 12:00 |
| Richard | Databases | 15:00 |
| Richard | Programming | 10:00 |
| Mary | Programming | 10:00 |
| Rebecca | Programming | 13:00 |

Candidate keys: {Student, Course} and {Student, Time}

# FDs in the Stream Relation

- Stream has the following non-trivial FDs
- {Student, Course} $\rightarrow$ {Time}
- {Time} $\rightarrow$ {Course}
- Since all attributes are key attributes, Stream is in 3NF

# Anomalies in Stream

- INSERT anomalies
  - You can’t add an empty stream
- UPDATE anomalies
  - Moving the 12:00 class to 9:00 means changing two rows
- DELETE anomalies
  - Deleting Rebecca removes a stream

| Student | Course | Time |
|---|---|---|
| John | Databases | 12:00 |
| Mary | Databases | 12:00 |
| Richard | Databases | 15:00 |
| Richard | Programming | 10:00 |
| Mary | Programming | 10:00 |
| Rebecca | Programming | 13:00 |

# Boyce-Codd Normal Form

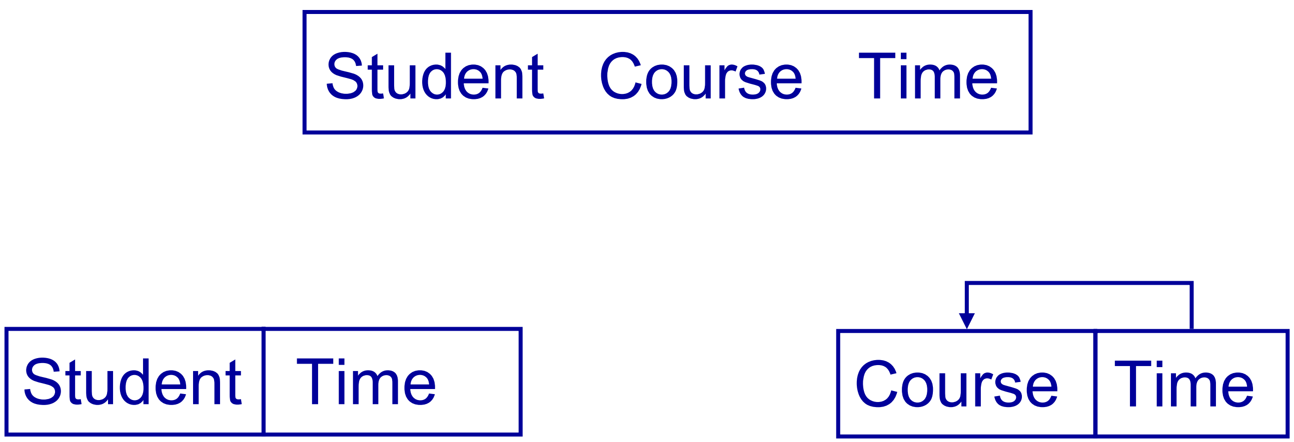
- A relation is in Boyce-Codd normal form (BCNF) if for every FD $A \rightarrow B$ either
  - B is contained in A (the FD is trivial), or
  - A contains a candidate key of the relation,
- In other words: every determinant in a non-trivial dependency is a (super) key.
- The same as 3NF except in 3NF we only worry about non-key Bs
- If there is only one candidate key then 3NF and BCNF are the same

# Stream and BCNF

- Stream is not in BCNF as the FD {Time} $\rightarrow$ {Course} is non-trivial and {Time} does not contain a candidate key

| Student | Course | Time |
|---|---|---|
| John | Databases | 12:00 |
| Mary | Databases | 12:00 |
| Richard | Databases | 15:00 |
| Richard | Programming | 10:00 |
| Mary | Programming | 10:00 |
| Rebecca | Programming | 13:00 |

# Conversion to BCNF



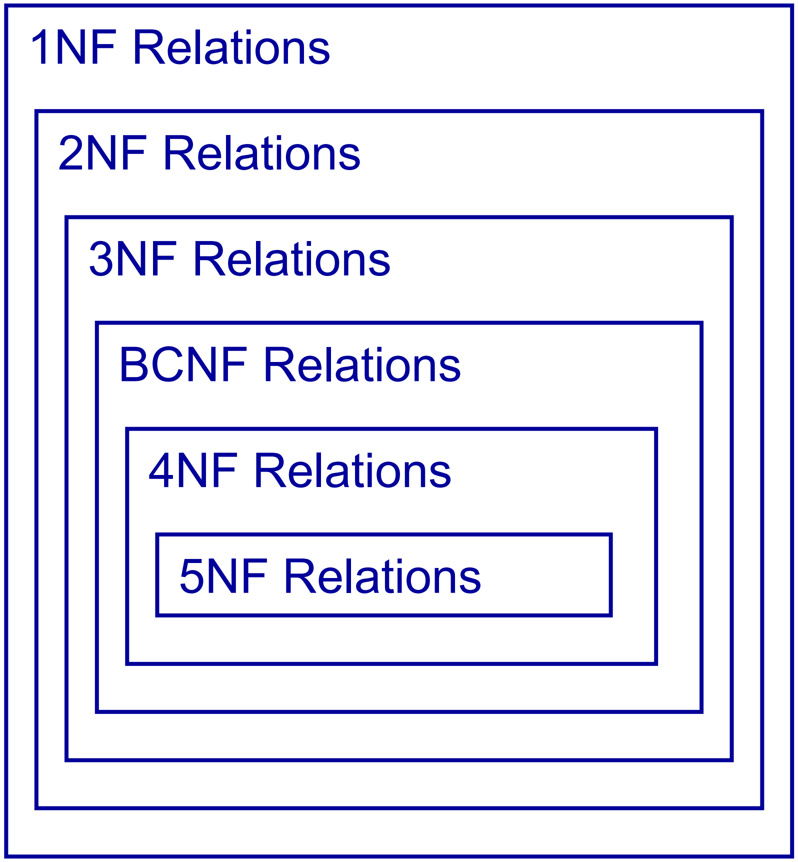

Stream has been put into BCNF but we have lost the FD
$\{Student, Course\} \rightarrow \{Time\}$

# 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} $\rightarrow$ {City}

Address1

| Number | Street | Postcode |
|---|---|---|

Address2

| Postcode | City |
|---|---|

# Lossless decomposition

- To normalise a relation, we used projections
- If R(A,B,C) satisfies $A \rightarrow B$ then we can project it on A,B and A,C without losing information
- Lossless decomposition:

$R = \pi_{AB}(R) \bowtie \pi_{AC}(R)$

where $\pi_{AB}(R)$ is projection of R on AB and $\bowtie$ is natural join.

- Reminder of projection:

R

| A | B | C |
|---|---|---|
| | | |

$\pi_{AB}(R)$

| A | B |
|---|---|
| | |

# Relational algebra reminder: selection

R

| A | B | C | D |
|---|---|---|---|
| 1 | x | c | c |
| 2 | y | d | e |
| 3 | z | a | a |
| 4 | u | b | c |
| 5 | w | c | d |

$\sigma_{C=D}(R)$

| A | B | C | D |
|---|---|---|---|
| 1 | x | c | c |
| 3 | z | a | a |

# Connection to SQL

SELECT A,B

FROM R1, R2, R3

WHERE (some property $\alpha$ holds)

translates into relational algebra

$\pi_{A,B}\, \sigma_{\alpha}\, (R1 \times R2 \times R3)$

# Relational algebra reminder: product

R1

| A | B |
|---|---|
| 1 | x |
| 2 | y |

R2

| A | C |
|---|---|
| 1 | w |
| 2 | v |
| 3 | u |

R1×R2

| A | B | A | C |
|---|---|---|---|
| 1 | x | 1 | w |
| 1 | x | 2 | v |
| 1 | x | 3 | u |
| 2 | y | 1 | w |
| 2 | y | 2 | v |
| 2 | y | 3 | u |

# Relational algebra: natural join

$$R1 \bowtie R2 = \pi_{R1.A,B,C}\ \sigma_{R1.A = R2.A}\ (R1 \times R2)$$

R1

| A | B |
|---|---|
| 1 | x |
| 2 | y |

R2

| A | C |
|---|---|
| 1 | w |
| 2 | v |
| 3 | u |

$R1 \bowtie R2$

| A | B | C |
|---|---|---|
| 1 | x | w |
| 2 | y | v |

# When is decomposition lossless: Module $\rightarrow$ Lecturer

R

| Module | Lecturer | Text |
|---|---|---|
| DBS | nza | CB |
| DBS | nza | UW |
| RDB | nza | UW |
| APS | rcb | B |

$\pi_{\text{Module,Lecturer}} R$

| Module | Lecturer |
|---|---|
| DBS | nza |
| RDB | nza |
| APS | rcb |

$\pi_{\text{Module,Text}} R$

| Module | Text |
|---|---|
| DBS | CB |
| DBS | UW |
| RDB | UW |
| APS | B |

# When is decomposition is not lossless: no fd

S

| First | Last | Age |
|---|---|---|
| John | Smith | 20 |
| John | Brown | 30 |
| Mary | Smith | 20 |
| Tom | Brown | 10 |

$\pi_{\text{First,Last}} S$

| First | Last |
|---|---|
| John | Smith |
| John | Brown |
| Mary | Smith |
| Tom | Brown |

$\pi_{\text{First,Age}} S$

| First | Age |
|---|---|
| John | 20 |
| John | 30 |
| Mary | 20 |
| Tom | 10 |

# When is decomposition is not lossless: no fd

$\pi_{\text{First,Last}}\, S \bowtie \pi_{\text{First,Last}}\, S$

| First | Last | Age |
|---|---|---|
| John | Smith | 20 |
| *John* | *Smith* | *30* |
| *John* | *Brown* | *20* |
| John | Brown | 30 |
| Mary | Smith | 20 |
| Tom | Brown | 10 |

$\pi_{\text{First,Last}} S$

| First | Last |
|---|---|
| John | Smith |
| John | Brown |
| Mary | Smith |
| Tom | Brown |

$\pi_{\text{First,Age}} S$

| First | Age |
|---|---|
| John | 20 |
| John | 30 |
| Mary | 20 |
| Tom | 10 |

# Heath's theorem

- A relation R(A,B,C) that satisfies a functional dependency $A \rightarrow B$ can always be non-loss decomposed into its projections $R1=\pi_{AB}(R)$ and $R2=\pi_{AC}(R)$.

Proof.

- First we show that $R \subseteq \pi_{AB}(R) \bowtie_{A} \pi_{AC}(R)$. This actually holds for any relation, does not have to satisfy $A \rightarrow B$.
- Assume $r \in R$. We need to show $r \in \pi_{AB}(R) \bowtie_{A} \pi_{AC}(R)$. Since $r \in R$, $r(A,B) \in \pi_{AB}(R)$ and $r(A,C) \in \pi_{AC}(R)$. Since r(A,B) and r(A,C) have the same value for A, their join $r(A,B,C) = r$ is in $\pi_{AB}(R) \bowtie_{A} \pi_{AC}(R)$.

# Heath's theorem

- Now we show that $\pi_{AB}(R) \bowtie_A \pi_{AC}(R) \subseteq R$. This only holds if R satisfies $A \rightarrow B$.
- Assume $r \in \pi_{AB}(R) \bowtie_A \pi_{AC}(R)$.
- So, $r(A,B) \in \pi_{AB}(R)$ and $r(A,C) \in \pi_{AC}(R)$.
- By the definition of projection, if $r(A,B) \in \pi_{AB}(R)$, then there is a tuple $s_1 \in R$ such that $s_1(A,B) = r(A,B)$. Similarly, since $r(A,C) \in \pi_{AC}(R)$, there is $s_2 \in R$ such that $s_2(A,C) = r(A,C)$.
- Since $s_1(A,B) = r(A,B)$ and $s_2(A,C) = r(A,C)$, $s_1(A) = s_2(A)$. So because of $A \rightarrow B$, $s_1(B) = s_2(B)$. This means that $s_1(A,B,C) = s_2(A,B,C) = r$ and $r \in R$.

# Normalisation in exams

- Consider a relation Book with attributes Author, Title, Publisher, City, Country, Year, ISBN. There are two candidate keys: ISBN and (Author, Title, Publisher, Year). City is the place where the book is published, and there are functional dependencies Publisher $\rightarrow$ City and City $\rightarrow$ Country. Is this relation in 2NF? Explain your answer. (4 marks)
- Is this relation in 3NF? Explain your answer. (5 marks)
- Is the relation above in BCNF? If not, decompose it to BCNF and explain why the resulting tables are in BCNF. (5 marks).

# Next Lecture

- Physical DB Issues
  - RAID arrays for recovery and speed
  - Indexes and query efficiency
- Query optimisation
  - Query trees
- For more information
  - Connolly and Begg chapter 21 and appendix C.5, Ullman and Widom 5.2.8

# Next Lecture

- More normalisation
  - Lossless decomposition; why our reduction to 2NF and 3NF is lossless
  - Boyce-Codd normal form (BCNF)
  - Higher normal forms
  - Denormalisation
- For more information
  - Connolly and Begg chapter 14
  - Ullman and Widom chapter 3.6