Week 6 Workshop – Normalisation

http://en.wikipedia.org/wiki/Ursus_Wehrli

Housekeeping

- **A**

Th

Housekeeping

- **A**

  Th
- **A**

  https://cs.anu.edu.au/dab/bench/db-exercises/

Australian National University

Housekeeping

- **A**

  Th

- **A**

  https://cs.anu.edu.au/dab/bench/db-exercises/

- **An anonymous survey from our course repre**

  (till 4 Sep under Week 6)

  Help us to improve our planning and teaching

Decomposition vs Normalisation

Decomposing a relation schema can possibly create more problems than it solves!

Australian National University

Decomposition vs Normalisation

> Decomposing a relation schema can possibly create more problems than it solves!

- Th

1. **Do we need to decompose a relation?**

Decomposition vs Normalisation

Decomposing a relation schema can possibly create more problems than it solves!

- Th

1. **Do we need to decompose a relation?**
   - **Several normal forms**

     ↪ help us to decide whether or not to d

Decomposition vs Normalisation

Decomposing a relation schema can possibly create more problems than it solves!

- Th

1. **Do we need to decompose a relation?**
   - **Several normal forms**

     ↪ help us to decide whether or not to d

2. **What problem (if any) does a given decomposition cause?**

Decomposition vs Normalisation

Decomposing a relation schema can possibly create more problems than it solves!

- Th

1. **Do we need to decompose a relation?**
   - **Several normal forms**

     $\hookrightarrow$ help us to decide whether or not to d

2. **What problem (if any) does a given decomposition cause?**
   - **Two properties**

     $\hookrightarrow$ help us to decide how to decompose a relation

Two Properties

- In addition to **data redundancy**, we need to consider the following pr

Australian National University

Two Properties

- In addition to **data redundancy**, we need to consider the following pr

To disallow the possibility of generating sp
NATURAL JOIN operation is applied to the r

Two Properties

- In addition to **data redundancy**, we need to consider the following pr

1. To disallow the possibility of generating sp
NATURAL JOIN operation is applied to the r

2. **Dependency preservation** – "**cap**

   To ensure that each functional dependency can be inferred from functional dependencies after decomposition.

Lossless Join – Example

- **Lossless join** – "**capture the same data**"
  To disallow the possibility of generating spurious tuples when a NATURAL JOIN operation is applied to the relations after decomposition.

Lossless Join – Example

- **Lossless join** – "**capture the same data**"
  To disallow the possibility of generating spurious tuples when a NATURAL JOIN operation is applied to the relations after decomposition.

R_1	
Name	StudentID
Mike	123456
Mike	123458

St[illegible]	[illegible]
123458	25/01/1988

- **Example 1:** Does the decomposition of R into R_1 and R_2 has the lossless join property?

Lossless Join – Example

- **Lossless join** – "**capture the same data**"
  To disallow the possibility of generating spurious tuples when a NATURAL JOIN operation is applied to the relations after decomposition.

[illegible]		
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]

R_1	
Name	StudentID
Mike	123456
Mike	123458

[illegible]	
St[illegible]	[illegible]
[illegible]	[illegible]
123458	25/01/1988

- **Example 1:** Does the decomposition of R into R_1 and R_2 has the lossless join property?
  Yes, because the natural join of R_1 and R_2 yields R.

Lossless Join – Example

- **Lossless join** – "**capture the same data**".
  To disallow the possibility of generating spurious tuples when a NATURAL JOIN operation is applied to the relations after decomposition.

[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]

R_3

Name	StudentID
Mike	123456
Mike	123458

N	[illegible]
M	[illegible]
Mike	25/01/1988

- **Example 2:** Does the decomposition of R into R_3 and R_4 has the lossless join property?

Lossless Join – Example

- **Lossless join** – "**capture the same data**":
  To disallow the possibility of generating spurious tuples when a NATURAL JOIN operation is applied to the relations after decomposition.

R_3	
Name	StudentID
Mike	123456
Mike	123458

N	
M	
Mike	25/01/1988

- **Example 2:** Does the decomposition of R into R_3 and R_4 has the lossless join property?
  No, because the natural join of R_3 and R_4 generates spurious tuples.

Lossless Join – Example

- **Example 2:** The following decomposition from R into R_3 and R_4 doesn't have the lossless join property. **It generates spurious tuples.**

[illegible]	[illegible]	[illegible]
Mike	123456	20/09/1989
Mike	123458	25/01/1988

R_3 JOIN R_4		
[illegible]	[illegible]	DoB
[illegible]	[illegible]	/09/1989
Mike	123456	25/01/1988
[illegible]	[illegible]	989
[illegible]	[illegible]	988

R_3	
Name	StudentID
Mike	123456
Mike	123458

R_4	
Name	DoB
Mike	20/09/1989
Mike	25/01/1988

Lossless Join – Example

- Lossless join – “capture the same data”

R		
Name	StudentID	DoB

R_1	
Name	StudentID
Mike	123456
Mike	123458

R	
StudentID	
123456	
123458	

R_3	
Name	StudentID
Mike	123456
Mike	123458

R_4	
Name	DoB
Mike	20/09/1989
Mike	25/01/1988

Not lossless join

Dependency Preservation – Example

- **Dependency preservation**: To ensure that each functional dependency can be inferred from functional dependencies after decomposition.

Dependency Preservation – Example

- **Dependency preservation**: To ensure that each functional dependency can be inferred from functional dependencies after decomposition.
- **Example 1:** Given a FD StudentID Name defined on R

R_1	
Name	StudentID
Mike	123456
Mike	123458

Stud	
12	
123458	COMP2600

- Does the above decomposition preserves $\{\text{StudentID}\} \rightarrow \{\text{Name}\}$?

Australian National University

Dependency Preservation – Example

- **Dependency preservation**: To ensure that each functional dependency can be inferred from functional dependencies after decomposition.
- **Example 1:** Given a FD StudentID Name defined on *R*

R_1	
Name	StudentID
Mike	123456
Mike	123458

Stud	
12	
123458	COMP2600

- Does the above decomposition preserves {StudentID} $\rightarrow$ {Name}?
  Yes, because {StudentID} and {Name} are both in R_1 after decomposition and thus {StudentID} $\rightarrow$ {Name} is preserved in R_1.

Australian National University

Dependency Preservation – Example

- **Dependency preservation**: To ensure that each functional dependency can be inferred from functional dependencies after decomposition.
- **Example 2:** Given a FD StudentID Name defined on *R*

R_1	
Name	CourseNo
Mike	COMP2400
Mike	COMP2600

St	
1	
123458	COMP2600

- Does the above decomposition preserves {StudentID} $\rightarrow$ {Name}?

Dependency Preservation – Example

- **Dependency preservation**: To ensure that each functional dependency can be inferred from functional dependencies after decomposition.
- **Example 2:** Given a FD StudentID Name defined on R

R_1	
Name	CourseNo
Mike	COMP2400
Mike	COMP2600

St	
1	
123458	COMP2600

- Does the above decomposition preserves $\{\text{StudentID}\} \rightarrow \{\text{Name}\}$?
  No, because $\{\text{StudentID}\}$ and $\{\text{Name}\}$ are not in a same relation after decomposition.

Dependency Preservation – Example

- **Dependency preservation**: To ensure that each functional dependency can be inferred from functional dependencies after decomposition.
- **Example 3:** Given a set of FDs { {StudentID} → {Email}, {Email} → {

Tom	123123	123123@anu.edu.au

R_1

Name	Email
Mike	123456@anu.edu.au
Tom	123123@anu.edu.au

StudentI	
123456	123456@anu.edu.au
123123	123123@anu.edu.au

Dependency Preservation – Example

- **Dependency preservation**: To ensure that each functional dependency can be inferred from functional dependencies after decomposition.
- **Example 3:** Given a set of FDs { {StudentID} → {Email}, {Email} → {

Tom	123123	123123@anu.edu.au

R_1	
Name	Email
Mike	123456@anu.edu.au
Tom	123123@anu.edu.au

StudentI	
123456	123456@anu.edu.au
123123	123123@anu.edu.au

- Does the above decomposition preserves {StudentID} → {Name}?

Dependency Preservation – Example

- **Dependency preservation**: To ensure that each functional dependency can be inferred from functional dependencies after decomposition.
- **Example 3:** Given a set of FDs { {StudentID} $\rightarrow$ {Email}, {Email} $\rightarrow$ {

Tom	123123	123123@anu.edu.au

R_1	
Name	Email
Mike	123456@anu.edu.au
Tom	123123@anu.edu.au

StudentI	
123456	123456@anu.edu.au
123123	123123@anu.edu.au

- Does the above decomposition preserves {StudentID} $\rightarrow$ {Name}? **Yes**, because {StudentID} $\rightarrow$ {Name} can be inferred by {StudentID} $\rightarrow$ {Email} (preserved in R_2) and {Email} $\rightarrow$ {Name} (preserved in R_1).

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,

Australian National University

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

Australian National University

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R={A, B, C} with the set of FDs }.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

Australian National University

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - [illegible] lds.

- Consider R={A, B, C} with the set of FDs [illegible] }.
  Does the decompostion of R into $R_1 = \{$ [illegible]
  lossless join and **dependency preservin**[illegible]
  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_2 = \{A \rightarrow C\}$

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R=$\{$A, B, C$\}$ with the set of FDs $\}$.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_2 = \{A \rightarrow C\}$
  - **Lossless join?**

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R={A, B, C} with the set of FDs }.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_2 = \{A \rightarrow C\}$
  - **Lossless join?** Yes because A is a superkey for R_1.

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R={A, B, C} with the set of FDs }.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_2 = \{A \rightarrow C\}$
  - **Lossless join?** Yes because A is a superkey for R_1.
  - **Dependency preserving?**

Australian National University

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R={A, B, C} with the set of FDs }.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_2 = \{A \rightarrow C\}$
  - **Lossless join?** Yes because A is a superkey for R_1.
  - **Dependency preserving?** No because $(\Sigma_1 \cup \Sigma_2)^* \neq \Sigma^*$ from the fact that $\{A \rightarrow B, A \rightarrow C\} \not\models B \rightarrow C$.

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R=$\{$A, B, C$\}$ with the set of FDs $\}$.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R={A, B, C} with the set of FDs }.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_3 = \{B \rightarrow C\}$

Australian National University

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R=$\{$A, B, C$\}$ with the set of FDs $\}$.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**
  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_3 = \{B \rightarrow C\}$
  - **Lossless join?**

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R={A, B, C} with the set of FDs }.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_3 = \{B \rightarrow C\}$
  - **Lossless join?** Yes because B is a superkey for R_3.

Discussion

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R={A, B, C} with the set of FDs }.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_3 = \{B \rightarrow C\}$
  - **Lossless join?** Yes because B is a superkey for R_3.
  - **Dependency preserving?**

Discussion

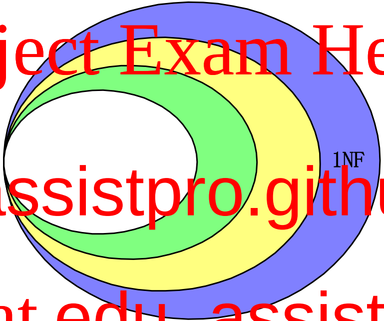
- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a
  - lds.

- Consider R={A, B, C} with the set of FDs }.
  Does the decompostion of R into $R_1 = \{$
  lossless join and **dependency preservin**

  - $\Sigma_1 = \{A \rightarrow B\}$ and $\Sigma_3 = \{B \rightarrow C\}$
  - **Lossless join?** Yes because B is a superkey for R_3.
  - **Dependency preserving?** Yes because $(\Sigma_1 \cup \Sigma_3)^* = \Sigma^*$ from the fact that $\{A \rightarrow B, B \rightarrow C\} \models A \rightarrow C$.

Normal Forms

Normal forms	Test criteria
1NF	weak
⇓	
BCNF	strong
...	



- **Note that:**
  - 1NF is independent of keys and functional dependencies.
  - 2NF, 3NF and BCNF are based on keys and functional dependencies.
  - 4NF and 5NF are based on other dependencies (will not be covered in this course).

BCNF

D

BCNF - Definition

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey** .

Australian National University

BCNF - Definition

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey**.
- W
fu

BCNF - Definition

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey**.

- W
  fu

  - non-trivial FD $X \rightarrow Y$, there exists a rela
    distinct tuples t_1 and t_2 with $t_1[XY] =$

{CourseName} $\rightarrow$ {Instructor}

StudentID		ructor
u123456	Operating Systems	Hegel
u234566	Relational Databases	Yu
u234567	Relational Databases	Yu

Normalisation to BCNF

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey**.

- Consider the relation schema TEACH with the following FD:
  -

u123456	Operating Systems	
u234566	Relational Databases	
u234567	Relational Databases	

- Is TEACH in BCNF?

Normalisation to BCNF

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then **X is a superkey**.
- Consider the relation schema TEACH with the following FD:
  -

u123456	Operating Systems	
u234566	Relational Databases	
u234567	Relational Databases	

- Is TEACH in BCNF?

  Not in BCNF because {CourseName} is not a superkey.

Australian National University

Normalisation to BCNF

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then **X is a superkey** .
- Consider the relation schema TEACH with the following FD:
  -

u123456	Operating Systems	
u234566	Relational Databases	
u234567	Relational Databases	

- Is TEACH in BCNF?

  Not in BCNF because {CourseName} is not a superkey.

- Did we represent the same fact twice (or more times)?

Normalisation to BCNF

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then **X is a superkey**.
- Consider the relation schema TEACH with the following FD:
  -

u123456	Operating Systems	
u234566	Relational Databases	
u234567	Relational Databases	

- Is TEACH in BCNF?

  Not in BCNF because {CourseName} is not a superkey.
- Did we represent the same fact twice (or more times)?

  Yes, the Instructor of Relational Databases is Yu.

Normalisation to BCNF

- **Algorithm** for a BCNF-decomposition

  Input: a relation schema R' and a set Σ of FDs on R'.

  O

Australian National University

Normalisation to BCNF

- **Algorithm** for a BCNF-decomposition

  Input: a relation schema R' and a set Σ of FDs on R'.

  O

  -

Normalisation to BCNF

- **Algorithm** for a BCNF-decomposition

  Input: a relation schema R' and a set Σ of FDs on R'.

  O

  -
  - $\mathcal{S}$:

    - $\rightarrow$ f any;
    - Replace R in $\mathcal{S}$ by two relation sc nd project the FDs to these two relation sc

Australian National University

Normalisation to BCNF

- **Algorithm** for a BCNF-decomposition

  Input: a relation schema R' and a set Σ of FDs on R'.

  O

  -
  - $\mathcal{S}$:
    - $\rightarrow$ f any;
    - Replace R in $\mathcal{S}$ by two relation sc nd project the FDs to these two relation sc

- Does the above **Algorithm** always produc

Australian National University

Normalisation to BCNF

- **Algorithm** for a BCNF-decomposition

  Input: a relation schema R' and a set Σ of FDs on R'.

  O

  -

  - $\mathcal{S}$:
    - $\rightarrow$ f any;
    - Replace R in $\mathcal{S}$ by two relation sc nd project the FDs to these two relation sc

- Does the above **Algorithm** always produc

  If R with a set Σ of FDs is decomposed i $_1$ Σ_1 $_2$ ith Σ_2, this decomposition is **lossless join** if and only if the common attributes of R_1 and R_2 are a superkey for R_1 or R_2.

Normalisation to BCNF

- **Algorithm** for a BCNF-decomposition

  Input: a relation schema R' and a set Σ of FDs on R'.

  O

  -

  - $\mathcal{S}$:

    - $\rightarrow$ f any;
    - Replace R in $\mathcal{S}$ by two relation sc nd project the FDs to these two relation sc

- Does the above **Algorithm** always produc

  If R with a set Σ of FDs is decomposed i $_1$ Σ_1 $_2$ ith Σ_2, this decomposition is **lossless join** if and only if the common attributes of R_1 and R_2 are a superkey for R_1 or R_2.

- **Yes because X is a superkey for XY.**

Normalisation to BCNF

Australian National University

Normalisation to BCNF

Normalisation to BCNF

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey** .
- Consider the relation schema TEACH with the following FD:

{CourseN

		ructor
		gel
		Yu
u234567	Re	

Normalisation to BCNF

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey**.
- Consider the relation schema TEACH with the following FD:

{CourseN

		ructor
		gel
		Yu
u234567	Re	

- Can we normalise TEACH into BCNF?

Normalisation to BCNF

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then **X is a superkey** .
- Consider the relation schema TEACH with the following FD:

{CourseN

		ructor
		gel
		Yu
u234567	Re	

- Can we normalise TEACH into BCNF? Ye

R_1

CourseName	Instructor
Operating Systems	Hegel
Relational Databases	Yu

Stude	
u123456	Operating Systems
u234566	Relational Databases
u234567	Relational Databases

Normalisation to BCNF

- A relation schema R is in **BCNF** if whenever a non-trivial FD $X \to A$ holds in R, then **X is a superkey**.
- Consider the relation schema TEACH with the following FD:

{CourseN

		ructor
		gel
		Yu
u234567	Re	

- Can we normalise TEACH into BCNF? Ye

R_1

CourseName	Instructor
Operating Systems	Hegel
Relational Databases	Yu

Stude	
u123456	Operating Systems
u234566	Relational Databases
u234567	Relational Databases

- **Do not represent the same fact twice (within a relation)!**

BCNF - Exercise

- Consider INTERVIEW={OfficerID, CustomerID, Date, Time, Room} with the following FDs:

- Is INTERVIEW in BCNF? If not, normalize I

BCNF - Exercise

- Consider INTERVIEW={OfficerID, CustomerID, Date, Time, Room} with the following FDs:
  -
  -
  -
  -

- Is INTERVIEW in BCNF? If not, normalize I

  - {CustomerID, Date}, {OfficerID, Dat
    Room} are the keys.

Australian National University

BCNF - Exercise

- Consider INTERVIEW={OfficerID, CustomerID, Date, Time, Room} with the following FDs:
  -
  -
  -
  -

- Is INTERVIEW in BCNF? If not, normalize I

  - {CustomerID, Date}, {OfficerID, Dat Room} are the keys.
  - Any superkey must contain one of of these keys as a subset.

Australian National University

BCNF - Exercise

- Consider INTERVIEW={OfficerID, CustomerID, Date, Time, Room} with the following FDs:

- Is INTERVIEW in BCNF? If not, normalize I

  - {CustomerID, Date}, {OfficerID, Dat
    Room} are the keys.
  - Any superkey must contain one of of these keys as a subset.
- **INTERVIEW is not in BCNF because {OfficerID, Date} $\rightarrow$ {Room} and {OfficerID, Date} is not a superkey.**

BCNF - Exercise

- We decompose INTERVIEW along the FD: {OfficerID, Date} $\rightarrow$ {Room}

INTERVIEW				
S1024	P107	14/11/2013	14:00	R10

INTERVIEW1		
OfficerID	Date	Room
S1011	12/11/2013	R15
S1024	14/11/2013	R10

OfficerID			Time
S1011			10:00
S1011	P105	12/11/2013	12:00
S1024	P108	14/11/2013	14:00
S1024	P107	14/11/2013	14:00

Australian National University

BCNF - Exercise

- We decompose INTERVIEW along the FD: {OfficerID, Date} → {Room}

INTERVIEW				
S1024	P107	14/11/2013	14:00	R10

INTERVIEW1		
OfficerID	Date	Room
S1011	12/11/2013	R15
S1024	14/11/2013	R10

OfficerID			Time
S1011			10:00
S1011	P105	12/11/2013	12:00
S1024	P108	14/11/2013	14:00
S1024	P107	14/11/2013	14:00

- **Do not represent the same fact twice (within a relation)!**

Australian National University

BCNF - Exercise

- Consider INTERVIEW={OfficerID, CustomerID, Date, Time, Room} with the following FDs:
  - {OfficerID, Date} → {Room}
  - CustomerID, Date OfficerID, Time

INTERVIEW1		
OfficerID	Date	Room
S1011	12/11/2013	R15
S1024	14/11/2013	R10

OfficerID			Time
S1011			10:00
S1011			12:00
S1024			14:00
S1024			14:00

- Project FDs on two new relation schemas.

BCNF - Exercise

- Consider INTERVIEW={OfficerID, CustomerID, Date, Time, Room} with the following FDs:
  - {OfficerID, Date} → {Room}
  - CustomerID, Date OfficerID, Time
  -
  -

INTERVIEW1		
OfficerID	Date	Room
S1011	12/11/2013	R15
S1024	14/11/2013	R10

OfficerID			Time
S1011			10:00
S1011			12:00
S1024			14:00
S1024			14:00

- Project FDs on two new relation schemas.

  INTERVIEW1: {OfficerID, Date} → {Room}

Australian National University

BCNF - Exercise

- Consider INTERVIEW={OfficerID, CustomerID, Date, Time, Room} with the following FDs:
  - {OfficerID, Date} → {Room}
  - CustomerID, Date OfficerID, Time
  -
  -

INTERVIEW1		
OfficerID	Date	Room
S1011	12/11/2013	R15
S1024	14/11/2013	R10

OfficerID			Time
S1011			10:00
S1011			12:00
S1024			14:00
S1024			14:00

- Project FDs on two new relation schemas.

  INTERVIEW1: {OfficerID, Date} → {Room}

  INTERVIEW2: {CustomerID, Date} → {OfficerID, Time}, {OfficerID, Date, Time} → {CustomerID}.

BCNF - Exercise

- Consider INTERVIEW={OfficerID, CustomerID, Date, Time, Room} with the following FDs:
  - {OfficerID, Date} $\rightarrow$ {Room}
  -
  -
  -

INTERVIEW1		
OfficerID	Date	Room
S1011	12/11/2013	R15
S1024	14/11/2013	R10

INTERVIEW2			
OfficerID			Time
S1011			10:00
S1011			12:00
S1024			14:00
S1024	P107	14/11/2013	14:00

- Is this decomposition dependency-preservation?

BCNF - Exercise

- Consider INTERVIEW={OfficerID, CustomerID, Date, Time, Room} with the following FDs:
  - {OfficerID, Date} → {Room}
  -
  -
  -

INTERVIEW1		
OfficerID	Date	Room
S1011	12/11/2013	R15
S1024	14/11/2013	R10

INTERVIEW2			
OfficerID			Time
S1011			10:00
S1011			12:00
S1024			14:00
S1024	P107	14/11/2013	14:00

- Is this decomposition dependency-preservation?
  No because {Date, Time, Room} → {CustomerID} is lost (and cannot be recovered)!

BCNF - Order Does Matter

- When applying BCNF decomposition, the order in which the FDs are ap

BCNF - Order Does Matter

- When applying BCNF decomposition, the order in which the FDs are ap
- E

BCNF - Order Does Matter

- When applying BCNF decomposition, the order in which the FDs are ap
- E
  -

BCNF - Order Does Matter

- When applying BCNF decomposition, the order in which the FDs are ap
- E
  -

$$R_1 = \{B, C\}, \Sigma_1 = \{B \rightarrow C, C \rightarrow B \}$$

Australian National University

BCNF - Order Does Matter

- When applying BCNF decomposition, the order in which the FDs are ap

- **E**
  -

    $R_1 = \{B, C\}, \Sigma_1 = \{B \rightarrow C.C \rightarrow B$ $\}$

  - **Case 2:** (Using $B \rightarrow C$ first)

Australian National University

BCNF - Order Does Matter

- When applying BCNF decomposition, the order in which the FDs are ap

- E
  -

    $R_1 = \{B, C\}, \Sigma_1 = \{B \to C.C \to B \quad \}$

  - **Case 2:** (Using $B \to C$ first)

    $R_1' = \{B, C\}, \Sigma_1' = \{B \to C, C \to B\}; R_2' = \{A, B\}, \Sigma_2' = \{A \to B\};$

Australian National University

Lossless Join & Dependency Preservation

- S be
- Is there **a less restrictive normal form** s dependency-preserving decomposition can a

Lossless Join & Dependency Preservation

- S
  be
- Is there **a less restrictive normal form** s
  dependency-preserving decomposition can a
  Yes, refer to 3NF.

3NF - Definition

- A r *A* holds in
  R, t

- Q

Australian National University

3NF - Definition

- A r A holds in
  R, t

- Q

  Yes

3NF - Definition

- A r *A* holds in R, t

- Q

  Yes

- 3NF **preserves all the functional dependen ing some data redundancy**.

Normalisation to 3NF

- Consider the following FDs of ENROL:
  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName};
  - {ConfirmedBy_ID} → {StaffName}.

				ffName
				ane
				inda
123458	COMP2600	2008 S2		

Normalisation to 3NF

- Consider the following FDs of ENROL:
  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName};
  - {ConfirmedBy_ID} → {StaffName}.

[illegible]	[illegible]	[illegible]	[illegible]	[illegible] ffName
[illegible]	[illegible]	[illegible]	[illegible]	[illegible] ane
[illegible]	[illegible]	[illegible]	[illegible]	inda
123458	COMP2600	2008 S2	[illegible]	[illegible]

- **Is ENROL in 3NF?**

Normalisation to 3NF

- Consider the following FDs of ENROL:
  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName};
  - {ConfirmedBy_ID} → {StaffName}.

				...ffName
				...ane
				inda
123458	COMP2600	2008 S2		

- **Is ENROL in 3NF?**
  - {StudentID, CourseNo, Semester} i

Normalisation to 3NF

- Consider the following FDs of ENROL:
  - {StudentID, CourseNo, Semester} $\rightarrow$ {ConfirmedBy_ID, StaffName};
  - {ConfirmedBy_ID} $\rightarrow$ {StaffName}.

[illegible]	[illegible]	[illegible]	[illegible]	ffName
[illegible]	[illegible]	[illegible]	[illegible]	ane
				inda
123458	COMP2600	2008 S2		

- **Is ENROL in 3NF?**
  - {StudentID, CourseNo, Semester} i

    A relation schema R is in **3NF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey** or A is a **prime attribute**.

Normalisation to 3NF

- Consider the following FDs of ENROL:
  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName};
  - {ConfirmedBy_ID} → {StaffName}.

[illegible]	[illegible]	[illegible]	[illegible]	ffName
[illegible]	[illegible]	[illegible]	[illegible]	ane
				inda
123458	COMP2600	2008 S2		

- **Is ENROL in 3NF?**
  - {StudentID, CourseNo, Semester} i

    A relation schema R is in **3NF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then **X is a superkey** or **A is a prime attribute**.
  - **Not in 3NF,** because of {ConfirmedBy_ID} → {StaffName}: {ConfirmedBy_ID} is NOT a **superkey** and {StaffName} is NOT a **prime attribute**.

Normalisation to 3NF

- **Algorithm** for a dependency-preserving and lossless 3NF-decomposition

In

O

Normalisation to 3NF

- **Algorithm** for a dependency-preserving and lossless 3NF-decomposition

In

O

-

Normalisation to 3NF

- **Algorithm** for a dependency-preserving and lossless 3NF-decomposition

In

O

- Group FDs in Σ' by their left-hand-side attribue sets

Normalisation to 3NF

- **Algorithm** for a dependency-preserving and lossless 3NF-decomposition

In

O

- Group FDs in Σ' by their left-hand-side attribue sets
- For each distinct left-hand-side X_i o
  $X_i \rightarrow A_1, X_i \rightarrow A_2, \ldots, X_i \rightarrow A_k$
  - Add $R_i = X_i \cup \{A_1\} \cup \{A_2\} \cdots \cup \{$

Normalisation to 3NF

- **Algorithm** for a dependency-preserving and lossless 3NF-decomposition

In

O

- Group FDs in Σ' by their left-hand-side attribue sets
- For each distinct left-hand-side X_i o

  $X_i \to A_1, X_i \to A_2, \ldots, X_i \to A_k$

  - Add $R_i = X_i \cup \{A_1\} \cup \{A_2\} \cdots \cup \{$
- Remove all redundant ones from $\mathcal{S}$ (i.e., remove R_i if $R_i \subseteq R_j$)

Normalisation to 3NF

- **Algorithm** for a dependency-preserving and lossless 3NF-decomposition

In

O

-
  - Group FDs in Σ' by their left-hand-side attribue sets
  - For each distinct left-hand-side X_i o
    $X_i \to A_1, X_i \to A_2, \ldots, X_i \to A_k$
    - Add $R_i = X_i \cup \{A_1\} \cup \{A_2\} \cdots \cup \{$

  - Remove all redundant ones from $\mathcal{S}$ (i.e., remove R_i if $R_i \subseteq R_j$)
  - if $\mathcal{S}$ does not contain a superkey of R, add a key of R as R_0 into $\mathcal{S}$.

Normalisation to 3NF

- **Algorithm** for a dependency-preserving and lossless 3NF-decomposition

In

O

- Group FDs in Σ' by their left-hand-side attribue sets
- For each distinct left-hand-side X_i o
  $X_i \to A_1, X_i \to A_2, \ldots, X_i \to A_k$
  - Add $R_i = X_i \cup \{A_1\} \cup \{A_2\} \cdots \cup \{$

- Remove all redundant ones from $\mathcal{S}$ (i.e., remove R_i if $R_i \subseteq R_j$)
- if $\mathcal{S}$ does not contain a superkey of R, add a key of R as R_0 into $\mathcal{S}$.
- Project the FDs in Σ' onto each relation schema in $\mathcal{S}$

Normalisation to 3NF

Australian National University

Normalisation to 3NF

Normalisation to 3NF

Australian National University

Equivalence of Functional Dependencies

- Σ_1 and Σ_2 are **equivalent** if $\Sigma_1^* = \Sigma_2^*$.
- $\Sigma_1^* = \Sigma_2^*$ if $\Sigma_1 \models \Sigma_2$ and $\Sigma_2 \models \Sigma_1$.

Equivalence of Functional Dependencies

- Σ_1 and Σ_2 are **equivalent** if $\Sigma_1^* = \Sigma_2^*$.
- $\Sigma_1^* = \Sigma_2^*$ iff $\Sigma_1 \models \Sigma_2$ and $\Sigma_2 \models \Sigma_1$.

- **Example 1:**
  $\Sigma_1 = \{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$ and Σ_2
  If $\Sigma_1^* = \Sigma_2^*$,

Australian National University

Equivalence of Functional Dependencies

- Σ_1 and Σ_2 are **equivalent** if $\Sigma_1^* = \Sigma_2^*$.
- $\Sigma_1^* = \Sigma_2^*$ iff $\Sigma_1 \models \Sigma_2$ and $\Sigma_2 \models \Sigma_1$.

- **Example 1:**
  $\Sigma_1 = \{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$ and Σ_2
  If $\Sigma_1^* = \Sigma_2^*$, then Σ_1 is not **minimal**

Equivalence of Functional Dependencies

- Σ_1 and Σ_2 are **equivalent** if $\Sigma_1^* = \Sigma_2^*$.
- $\Sigma_1^* = \Sigma_2^*$ if $\Sigma_1 \models \Sigma_2$ and $\Sigma_2 \models \Sigma_1$.

- **Example 1:**
  $\Sigma_1 = \{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$ and Σ_2
  If $\Sigma_1^* = \Sigma_2^*$, then Σ_1 is not **minimal**
- **Example 2:**
  $\Sigma_1 = \{X \rightarrow Y, XY \rightarrow Z\}$ and $\Sigma_2 = \{X \rightarrow Y, X \rightarrow Z\}$
  If $\Sigma_1^* = \Sigma_2^*$,

Equivalence of Functional Dependencies

- Σ_1 and Σ_2 are **equivalent** if $\Sigma_1^* = \Sigma_2^*$.
- $\Sigma_1^* = \Sigma_2^*$ iff $\Sigma_1 \models \Sigma_2$ and $\Sigma_2 \models \Sigma_1$.

- **Example 1:**
  $\Sigma_1 = \{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$ and Σ_2
  If $\Sigma_1^* = \Sigma_2^*$, then Σ_1 is not **minimal**
- **Example 2:**
  $\Sigma_1 = \{X \rightarrow Y, XY \rightarrow Z\}$ and $\Sigma_2 = \{X \rightarrow Y, X \rightarrow Z\}$
  If $\Sigma_1^* = \Sigma_2^*$, then Σ_1 is not **minimal**

Equivalence of Functional Dependencies

- Σ_1 and Σ_2 are **equivalent** if $\Sigma_1^* = \Sigma_2^*$.
- $\Sigma_1^* = \Sigma_2^*$ if $\Sigma_1 \models \Sigma_2$ and $\Sigma_2 \models \Sigma_1$.

- **Example 1:**
  $\Sigma_1 = \{X \to Y, Y \to Z, X \to Z\}$ and Σ_2
  If $\Sigma_1^* = \Sigma_2^*$, then Σ_1 is not **minimal**
- **Example 2:**
  $\Sigma_1 = \{X \to Y, XY \to Z\}$ and $\Sigma_2 = \{X \to Y, X \to Z\}$
  If $\Sigma_1^* = \Sigma_2^*$, then Σ_1 is not **minimal**
- **Questions:** Can we find the **minimal** one among equivalent sets of FDs?

Minimal Cover – The Hard Part!

Minimal Cover – The Hard Part!

- Let Σ be a set of FDs. A **minimal cover** Σ_m of Σ is a set of FDs such that

Minimal Cover – The Hard Part!

- Let Σ be a set of FDs. A **minimal cover** Σ_m of Σ is a set of FDs such that

$_{1}$ $_{k}$ $_{m}$ ith

$X \rightarrow A_1, \ldots, X \rightarrow A_k$;

Australian National University

Minimal Cover – The Hard Part!

- Let Σ be a set of FDs. A **minimal cover** Σ_m of Σ is a set of FDs such that

1 k m ith

$X \to A_1, \ldots, X \to A_k$;

3. **Determinant:** each FD has as few attribu possible, i.e., for each FD $X \to A$ in f X to see if we can replace $X \to A$ with $(X - B) \to A$ in Σ_m;

Minimal Cover – The Hard Part!

- Let Σ be a set of FDs. A **minimal cover** Σ_m of Σ is a set of FDs such that

1.

2. $_1$ $_k$ $_m$ ith
   $X \to A_1, \ldots, X \to A_k$;

3. **Determinant:** each FD has as few attribu
   possible, i.e., for each FD $X \to A$ in f X to
   see if we can replace $X \to A$ with $(X - B) \to A$ in Σ_m;

4. Remove a FD from Σ_m if it is redundant.

Minimal Cover - Examples

- Given the set of FDs Σ
  $\{StudentID, CourseNo, Semester\} \rightarrow \{ConfirmedBy\_ID, StaffName\}$
  $\{ConfirmedBy\_ID\} \rightarrow \{StaffName\}$
- w

Minimal Cover - Examples

- Given the set of FDs Σ
  $\{StudentID, CourseNo, Semester\} \rightarrow \{ConfirmedBy\_ID, StaffName\}$
  $\{ConfirmedBy\_ID\} \rightarrow \{StaffName\}$
- w
  -

Minimal Cover - Examples

- Given the set of FDs Σ
  $\{StudentID, CourseNo, Semester\} \rightarrow \{ConfirmedBy\_ID, StaffName\}$
  $\{ConfirmedBy\_ID\} \rightarrow \{StaffName\}$
- w

Minimal Cover - Examples

- Given the set of FDs Σ
  $\{StudentID, CourseNo, Semester\} \rightarrow \{ConfirmedBy\_ID, StaffName\}$
  $\{ConfirmedBy\_ID\} \rightarrow \{StaffName\}$
- w

{ } _ affName }

Australian National University

Minimal Cover - Examples

- Given the set of FDs Σ

  $\{StudentID, CourseNo, Semester\} \rightarrow \{ConfirmedBy\_ID, StaffName\}$

  $\{ConfirmedBy\_ID\} \rightarrow \{StaffName\}$

- w

  $\{$ $\}$ _ affName $\}$

  can be replaced by

  $\{StudentID, CourseNo, Semester\}$ _

  $\{StudentID, CourseNo, Semester\}$

Minimal Cover - Examples

- Given the set of FDs Σ
  {StudentID, CourseNo, Semester} → {ConfirmedBy_ID}
  {StudentID, CourseNo, Semester} → {StaffName}
  {
- w
  -
  -
    hand side;

Australian National University

Minimal Cover - Examples

- Given the set of FDs Σ

  $\{$StudentID, CourseNo, Semester$\} \rightarrow \{$ConfirmedBy_ID$\}$

  $\{$StudentID, CourseNo, Semester$\} \rightarrow \{$StaffName$\}$

  $\{$

- w

  3.
  3. hand side;
  3. check whether all the FDs in Σ hav

     hand side;

Minimal Cover - Examples

- Given the set of FDs Σ

  {StudentID, CourseNo, Semester} → {ConfirmedBy_ID}

  {StudentID, CourseNo, Semester} → {StaffName}

  {

- w

  hand side;

  3. check whether all the FDs in Σ hav

     hand side;

     check if { StudentID, CourseNo, Sem } is
     minimal with respect to the left hand side

     check if { StudentID, CourseNo, Semester } → {StaffName} is
     minimal with respect to the left hand side

Minimal Cover - Examples

- Given the set of FDs Σ
  $\{$StudentID, CourseNo, Semester$\} \rightarrow \{$ConfirmedByID$\}$
  $\{$StudentID, CourseNo, Semester$\} \rightarrow \{$StaffName$\}$
  $\{$
- w

hand side;

3. check whether all the FDs in Σ hav
   hand side;
   check if $\{$ StudentID, CourseNo, Sem $\}$ is
   minimal with respect to the left hand side
   check if $\{$ StudentID, CourseNo, Semester $\} \rightarrow \{$StaffName$\}$ is
   minimal with respect to the left hand side
   All look good!

Minimal Cover - Examples

- Given the set of FDs Σ

  {StudentID, CourseNo, Semester} → {ConfirmedBy_ID}

  {StudentID, CourseNo, Semester} → {StaffName}

  { _

- w

  1.
  2. hand side;
  3. check whether all the FDs in Σ hav hand side;

Minimal Cover - Examples

- Given the set of FDs Σ

  $\{StudentID, CourseNo, Semester\} \rightarrow \{ConfirmedBy\_ID\}$

  $\{StudentID, CourseNo, Semester\} \rightarrow \{StaffName\}$

  $\{$

- w

  hand side;

  3. check whether all the FDs in Σ hav
  hand side;
  4. look for a redundant FD in $\{\{$Stude
  $\{ConfirmedBy\_ID\}$, $\{$StudentID, Cou
  $\{StaffName\}$, $\{ConfirmedBy\_ID\} \rightarrow \{StaffName\}$ $\}$

Minimal Cover - Examples

- Given the set of FDs Σ

  {StudentID, CourseNo, Semester} → {ConfirmedBy_ID}

  {StudentID, CourseNo, Semester} → {StaffName}

  {

- W

  1.
  2. hand side;
  3. check whether all the FDs in Σ hav hand side;
  4. look for a redundant FD in { {Stude {ConfirmedBy_ID}, {StudentID, Cou {StaffName}, {ConfirmedBy_ID} → {StaffName} }
     - {StudentID, CourseNo, Semester} → {StaffName} is redundant and thus is removed

Minimal Cover - Examples

- Given the set of FDs Σ
  {StudentID, CourseNo, Semester} → {ConfirmedBy_ID}
  {StudentID, CourseNo, Semester} → {StaffName}
  {
- W

  1.
  2. hand side;
  3. check whether all the FDs in Σ hav
     hand side;

  4. look for a redundant FD in { {Stude
     {ConfirmedBy_ID}, {StudentID, Cou
     {StaffName}, {ConfirmedBy_ID} → {StaffName} }
     - {StudentID, CourseNo, Semester} → {StaffName} is redundant and thus is removed
  5. Therefore, the minial cover of Σ is { {StudentID, CourseNo, Semester} → {ConfirmedBy_ID}, {ConfirmedBy_ID} → {StaffName}}

Normalisation to 3NF – Example

- Consider ENROL again:
  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName}
  - ConfirmedBy_ID → StaffName

				Name
				...

- **C**
  preserving decomposition?

Normalisation to 3NF – Example

- Consider ENROL again:
  - {StudentID, CourseNo, Semester} $\rightarrow$ {ConfirmedBy_ID, StaffName}
  - {ConfirmedBy_ID} $\rightarrow$ {StaffName}

				Name
				...

- A
  {ConfirmedBy_ID}, {ConfirmedBy_ID} $\rightarrow$ {
- Hence, we have:

Normalisation to 3NF – Example

- Consider ENROL again:
  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName}
  - {ConfirmedBy_ID} → {StaffName}

				Name
				...

- A
  {ConfirmedBy_ID}, {ConfirmedBy_ID} → {
- Hence, we have:
  - R_1 ={StudentID, CourseNo, Semester,
    {StudentID, CourseNo, Semester}

Normalisation to 3NF – Example

- Consider ENROL again:

  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName}
  - {ConfirmedBy_ID} → {StaffName}

			_	Name
				...

- A
{ConfirmedBy_ID}, {ConfirmedBy_ID} → {
- Hence, we have:

  - R_1={StudentID, CourseNo, Semester,
{StudentID, CourseNo, Semester} _
  - R_2={ConfirmedBy_ID, StaffName} with
{ConfirmedBy_ID} → {StaffName}

Normalisation to 3NF – Example

- Consider ENROL again:
  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName}
  - {ConfirmedBy_ID} → {StaffName}

				Name
				...

- A
  {ConfirmedBy_ID}, {ConfirmedBy_ID} → {
- Hence, we have:
  - R_1={StudentID, CourseNo, Semester,
    {StudentID, CourseNo, Semester}
  - R_2={ConfirmedBy_ID, StaffName} with
    {ConfirmedBy_ID} → {StaffName}
  - Omit R_0 because R_1 is a superkey of ENROL.

Normalisation to 3NF – Example

- Consider ENROL again:

  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName}
  - {ConfirmedBy_ID} → {StaffName}

				Name
				...

- A {ConfirmedBy_ID}, {ConfirmedBy_ID} → {
- Hence, we have:

  - R_1={StudentID, CourseNo, Semester, {StudentID, CourseNo, Semester}
  - R_2={ConfirmedBy_ID, StaffName} with {ConfirmedBy_ID} → {StaffName}
  - Omit R_0 because R_1 is a superkey of ENROL.
- Is {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName} preserved?

Normalisation to 3NF – Example

- Consider ENROL again:
  - {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName}
  - {ConfirmedBy_ID} → {StaffName}

				Name
				...

- A
  {ConfirmedBy_ID}, {ConfirmedBy_ID} → {
- Hence, we have:
  - R_1={StudentID, CourseNo, Semester,
    {StudentID, CourseNo, Semester}
  - R_2={ConfirmedBy_ID, StaffName} with
    {ConfirmedBy_ID} → {StaffName}
  - Omit R_0 because R_1 is a superkey of ENROL.
- Is {StudentID, CourseNo, Semester} → {ConfirmedBy_ID, StaffName} preserved? **Yes**.

Normalisation to 3NF – Example

- Consider INTERVIEW:

INTERVIEW				
OfficerID	CustomerID	Date	Time	Room

3. {OfficerID, Date, Time} → {Custom
4. f Date, Time, Room} → {CustomerI

- **Is INTERVIEW in 3NF? If not, normalise dependency preserving 3NF.**

Australian National University

Normalisation to 3NF – Example

- Consider INTERVIEW:

INTERVIEW				
OfficerID	CustomerID	Date	Time	Room

3. $\{$OfficerID, Date, Time$\} \rightarrow \{$Custom
4. f Date, Time, Room$\} \rightarrow \{$CustomerI

- **Is INTERVIEW in 3NF? If not, normalise dependency preserving 3NF.**
  - A relation schema R is in **3NF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey** or A is a **prime attribute**.

Normalisation to 3NF – Example

- Consider INTERVIEW:

INTERVIEW				
OfficerID	CustomerID	Date	Time	Room

3. {OfficerID, Date, Time} $\rightarrow$ {Custom
4. f Date, Time, Room} $\rightarrow$ {CustomerI

- **Is INTERVIEW in 3NF? If not, normalise it into a dependency preserving 3NF.**
  - A relation schema R is in **3NF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey** or A is a **prime attribute**.
  - We know that {CustomerID, Date}, {OfficerID, Date, Time}, and {Date, Time, Room} are the keys.

Normalisation to 3NF – Example

- Consider INTERVIEW:

INTERVIEW				
OfficerID	CustomerID	Date	Time	Room

1.
2.
3. {OfficerID, Date, Time} → {Custom
4. {Date, Time, Room} → {CustomerI

- **Is INTERVIEW in 3NF? If not, normalise it into a dependency preserving 3NF.**
  - A relation schema R is in **3NF** if whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey** or A is a **prime attribute**.
  - We know that {CustomerID, Date}, {OfficerID, Date, Time}, and {Date, Time, Room} are the keys.

INTERVIEW is in 3NF because all the attributes are prime attributes.

The Minimal Cover – More Example

- Let us consider a relation schema LOTS(PropertyID, County, Lot, Area) with the following FDS:
  -
  -
  -

The Minimal Cover – More Example

- Let us consider a relation schema LOTS(PropertyID, County, Lot, Area) with the following FDS:
  -
  -
  -

- Let us abbriviate attributes of LOTS with first letter of represent our set of dependencies as F: }

- The minimal cover of a set of functional dependencies always exists but is not necessarily unique.

The Minimal Cover – More Example

- (Case X) Find a minimal cover of $F = \{P \to LCA, LC \to AP, A \to C\}$

Australian National University

The Minimal Cover – More Example

- (Case X) Find a minimal cover of $F = \{P \rightarrow LCA, LC \rightarrow AP, A \rightarrow C\}$
  1. **Initialise:** P LCA, LC AP, A C

Australian National University

The Minimal Cover – More Example

- (Case X) Find a minimal cover of $F = \{P \rightarrow LCA, LC \rightarrow AP, A \rightarrow C\}$
  1. **Initialise:** P LCA, LC AP, A C
  2. C}.

The Minimal Cover – More Example

- (Case X) Find a minimal cover of F = {P → LCA, LC → AP, A → C}
  1. **Initialise:** P LCA, LC AP, A C
  2. C}.
  3. → C}.

The Minimal Cover – More Example

- (Case X) Find a minimal cover of $F = \{P \rightarrow LCA, LC \rightarrow AP, A \rightarrow C\}$

1. **Initialise:** P LCA, LC AP, A C
2. C}.
3. → C}.
4.

Australian National University

The Minimal Cover – More Example

- (Case X) Find a minimal cover of F = {P → LCA, LC → AP, A → C}
  1. **Initialise:** P LCA, LC AP, A C
  2. C}.
  3. → C}.
  4.
  5.

The Minimal Cover – More Example

- (Case X) Find a minimal cover of F = {P → LCA, LC → AP, A → C}
  1. **Initialise:** P LCA, LC AP, A C

C}.

→ C}.

- (Case Y) Find a minimal cover of F = {P

Australian National University

The Minimal Cover – More Example

- (Case X) Find a minimal cover of F = $\{P \to LCA, LC \to AP, A \to C\}$
  1. **Initialise:** P LCA, LC AP, A C
  2. C}.
  3. $\to$ C}.
  4.
  5.

- (Case Y) Find a minimal cover of F = {P
  1. **Initialise:** $\{P \to LCA, LC \to AP, A$

Australian National University

The Minimal Cover – More Example

- (Case X) Find a minimal cover of F = {P → LCA, LC → AP, A → C}
  1. **Initialise:** P LCA, LC AP, A C
  2. C}.
  3. → C}.
  4.
  5.

- (Case Y) Find a minimal cover of F = {P
  1. **Initialise:** {P → LCA, LC → AP, A
  2. **Dependent:** {P → L, P → C, P →

The Minimal Cover – More Example

- (Case X) Find a minimal cover of F = {P → LCA, LC → AP, A → C}
  1. **Initialise:** P LCA, LC AP, A C
  2. C}.
  3. → C}.
  4.
  5.

- (Case Y) Find a minimal cover of F = {P
  1. **Initialise:** {P → LCA, LC → AP, A
  2. **Dependent:** {P → L, P → C, P →
  3. **Determinant:** {P → L, P → C, P → }.

The Minimal Cover – More Example

- (Case X) Find a minimal cover of F = $\{P \to LCA, LC \to AP, A \to C\}$
  1. **Initialise:** P LCA, LC AP, A C
  2. C}.
  3. $\to$ C}.
  4.
  5.

- (Case Y) Find a minimal cover of F = {P
  1. **Initialise:** $\{P \to LCA, LC \to AP, A$
  2. **Dependent:** $\{P \to L, P \to C, P \to$
  3. **Determinant:** $\{P \to L, P \to C, P \to$ }.
  4. **Remove redundant FD:** $\{LC \to P, P \to A\} \models LC \to A$. $\{P \to A, A \to C\} \models P \to C$.

The Minimal Cover – More Example

- (Case X) Find a minimal cover of F = $\{P \rightarrow LCA, LC \rightarrow AP, A \rightarrow C\}$
  1. **Initialise:** P LCA, LC AP, A C
  2. C}.
  3. $\rightarrow$ C}.
  4.
  5.

- (Case Y) Find a minimal cover of F = {P
  1. **Initialise:** $\{P \rightarrow LCA, LC \rightarrow AP, A$
  2. **Dependent:** $\{P \rightarrow L, P \rightarrow C, P \rightarrow$
  3. **Determinant:** $\{P \rightarrow L, P \rightarrow C, P \rightarrow$ }.
  4. **Remove redundant FD:** $\{LC \rightarrow P, P \rightarrow A\} \models LC \rightarrow A$. $\{P \rightarrow A, A \rightarrow C\} \models P \rightarrow C$.
  5. Thus a minimal cover is $\{P \rightarrow LA, LC \rightarrow P, A \rightarrow C\}$.

Normal Forms

- **BCNF**: Whenever a non-trivial FD $X \to A$ holds in R, then X is a **superkey** .

Normal Forms

- **BCNF**: Whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey**.

Normal Forms

- **BCNF**: Whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey**.

- **3NF**: Whenever a non-trivial FD $X \rightarrow A$ h **key** or A is a **prime attribute**.

Normal Forms

- **BCNF**: Whenever a non-trivial FD $X \rightarrow A$ holds in R, then X is a **superkey**.

- **3NF**: Whenever a non-trivial FD $X \rightarrow A$ h **key** or A is a **prime attribute**.

Do not abandon any FDs, even if some facts have to be represented more than once within a relation!

Normalisation Algorithms

BCNF-decomposition

- Repeat until no changes

  - F

  - S

  a

Normalisation Algorithms

BCNF-decomposition

- Repeat until no changes

  - F

  - S

  a

3NF-decomposition

- Find a minimal cover
- er
-
-
-

Normalisation Algorithms

BCNF-decomposition

- Repeat until no changes

  - F

  - S

  a

3NF-decomposition

- Find a minimal cover
- er
-
-
-

What properties do these algorit

Australian National University

Normalisation Algorithms

BCNF-decomposition

- Repeat until no changes

  - F

  - S
  a

3NF-decomposition

- Find a minimal cover
- er
-
-
-

What properties do these algorit

$\Downarrow$	$\Downarrow$
Lossless join	Lossless join + dependency preservation

Normalisation Algorithms

BCNF-decomposition

- Repeat until no changes

  - F

  - S

  a

3NF-decomposition

- Find a minimal cover
- ver
-
-
- Project FDs

What do you need to compute usi

Normalisation Algorithms

BCNF-decomposition

- Repeat until no changes

  - F

  - S

  a

3NF-decomposition

- Find a minimal cover
- ver
-
-
- Project FDs

What do you need to compute usi

$\Downarrow$

SOME superkeys (check)

$\Downarrow$

SOME superkeys (check)
ALL candidate keys
ONE minimal cover

Denormalisation

- Do we need to normalize relation schemas in all cases when designing a relational database?
- **Denormalisation** is a **design process** that
    -
    - queries.
- We need to distinguish:

    - **Unnormalised** – there is no systematic d
    - **Normalised** – redundancy is reduced after a systematic design (to minimise data inconsistencies).
    - **Denormalised** – redundancy is introduced after analysing the normalised design (to improve efficiency of queries)

Trade-offs – Data Redundancy vs. Query Efficiency

- Normalisation: **No Data Redundancy but No Efficient Query Processing**
- Data redundancies are eliminated in the following relations.

Unit

Trade-offs – Data Redundancy vs. Query Efficiency

- Normalisation: **No Data Redundancy but No Efficient Query Processing**
- Data redundancies are eliminated in the following relations.

	Unit

- However, the query for “list the names of students w with 6 units” requires 2 join operations.


```
SELECT Name, CourseNo
FROM ENROL e, COURSE c, STUDENT s
WHERE e.StudentID=s.StudentID AND e.CourseNo=c.CourseNo
AND c.Unit=6;
```

Trade-offs – Data Redundancy vs. Query Efficiency

- Denormalisation: Data Redundancy but Efficient Query Processing

- If a s
  ne

					Unit
Tom	123456	25/01/1988			
Tom	123456	25/01/1988			
Michael	123458	21/04/1985			

Trade-offs – Data Redundancy vs. Query Efficiency

- Denormalisation: **Data Redundancy but Efficient Query Processing**
- If a s
  ne

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Unit
Tom	123456	25/01/1988			
Tom	123456	25/01/1988			
Michael	[illegible]	21/04/1985			

- The query for "list the names of students who enrolle
  can be processed efficiently (no join needed).

```
SELECT Name, CourseNo FROM ENROLMENT WHERE Unit=6;
```

(credit cookie) Raymond F. Boyce (1947-1974)

"SEQUEL: A Structured English Query Language",
D.D. Chamberlin and R.F. Boyce,
Proc. ACM SIGMOD Workshop on Data Description, Access and Control,
Ann Arbor, Michigan (May 1974)

http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf