Kurs
Datenbankgrundlagen und Modellierung

Sebastian Maneth, Universität Bremen
maneth@uni-bremen.de

Sommersemester 2023

5.6.2023
Vorlesung 6: Normal Forms

Kurs **Datenbankgrundlagen und Modellierung**

17.4. Vorlesung 1 — Intro
24.4. V2 — ER, SQL
1.5. keine Vorlesung
4.5. Fragestunden
8.5. V3 — SQL
10.5. Ü1
11.5. Fragestunden
15.5. V4 — SQL
17.5. Ü2
22.5. V5 — SQL & funct. dependencies
24.5. Ü3
~~29.5. V6 — funct. dependencies~~
31.5. Ü4
5.6. V6 — normal forms
7.6. Ü5
12.6. V7 — modelling Intro
14.6. Ü6
19.6. V8 — class diagrams
21.6. Ü7
26.6. V9 — state charts
28.6. Ü8
3.7. V10 — sequence diagrams
5.7. Ü9
10.7. V11 — Klausurvorbereitung

Agenda

1.) Recap: Functional Dependencies

2.) Information and Dependency Preservation

3.) Third Normal Form (3NF)

4.) Boyce-Codd Normal Form (BCNF)

5.) Fourth Normal Form (4NF)

2NF removed from curriculum!

1.) Recap Functional Dependencies

Functional Dependencies

Let X, Y be non-empty sets of attributes of a given table T.
The functional dependency (FD) X → Y means that for any two tuples t_1, t_2 in val(T):

$$\text{if } \pi_X(t_1) = \pi_X(t_2) \text{ then } \pi_Y(t_1) = \pi_Y(t_2).$$

"if two tuples agree on their X-values, then they also agree on their Y-values"

"the X-values determine the Y-values"

$\pi_{X,Y}(\text{val}(T))$ is a **function** from X to Y

Functional Dependencies $\leftrightarrow$ Keys

Functional dependencies are a generalization of **key constraints**:

$$A_1, \ldots, A_n \text{ is a set of identifying attributes}^{11} \text{ in relation } R(A_1, \ldots, A_n, B_1, \ldots, B_m).$$

"superkey"

$$\Leftrightarrow$$

$$A_1 \ldots A_n \rightarrow B_1 \ldots B_m \text{ holds.}$$

Conversely, functional dependencies can be explained with keys.

$$A_1 \ldots A_n \rightarrow B_1 \ldots B_m \text{ holds for } R.$$

$$\Leftrightarrow$$

$$A_1, \ldots, A_n \text{ is a set of identifying attributes in } \pi_{A_1,\ldots,A_n,B_1,\ldots B_m}(R).$$

→ Functional dependencies are **"partial keys"**.

→ A goal of this chapter is to turn FDs into **real keys**, because key constraints can easily be enforced by a DBMS.

[11] If the set is also minimal, $A_1, \ldots, A_n$ is a key (↗ slide 53).

Functional Dependencies ↔ Keys

Why do we want to do that?

→ Functional dependencies are **"partial keys"**.

→ A goal of this chapter is to turn FDs into **real keys**, because key constraints can easily be enforced by a DBMS.

[11] If the set is also minimal, $A_1, \ldots, A_n$ is a key (↗ slide 53).

Functional Dependencies ↔ Keys

Why do we want to do that?
— to **remove redundancy!**

→ Functional dependencies are **"partial keys"**.

→ A goal of this chapter is to turn FDs into **real keys**, because key constraints can easily be enforced by a DBMS.

[11] If the set is also minimal, $A_1, \ldots, A_n$ is a key (↗ slide 53).

Functional Dependencies

A functional dependency X → Y is trivial, if Y is a subset of X.

How many non-trivial FDs are there for a table with 3 columns?
30

A functional dependency X → Y is completely non-trivial,

if $X \cap Y = \emptyset$.

How many completely non-trivial FD for a table w. 3 columns?
12

ab → c	b → a
ac → b	b → c
bc → a	b → ac
a → b	c → b
a → c	c → a
a → bc	c → ab

Still: if a → b holds, then we **do not care** that also ac → b holds.

The FD a → b **implies** ac → b (and it is strictly smaller).

Functional Dependencies

sch(Teach) = (Course, Prof, Time)

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Important assumption: each course is taught by exaclty one Prof.

What are the (interesting) functional dependencies?

Course —> Prof

Prof, Time —> Course

Functional Dependencies

sch(Teach) = (Course, Prof, Time)

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Important assumption: each course is taught by exaclty one Prof.

What are ~~the (interesting)~~ functional dependencies of this relation?

all the completely non-trivial

Course —> Prof

Prof, Time —> Course

Functional Dependencies

sch(Teach) = (Course, Prof, Time)

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Important assumption: each course is taught by exaclty one Prof.

What are ~~the (interesting)~~ functional dependencies of this relation?

all the completely non-trivial

Course —> Prof
Course, Time —> Prof
Prof, Time —> Course

Functional Dependencies

sch(Teach) = (Course, Prof, Time)

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Important assumption: each course is taught by exaclty one Prof.

What are the (interesting) functional dependencies?

Course —> Prof
Course, Time —> Prof
Prof, Time —> Course

What are the candidate keys?

Functional Dependencies

sch(Teach) = (Course, Prof, Time)

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Important assumption: each course is taught by exaclty one Prof.

What are the (interesting) functional dependencies?

Course —> Prof
<u>Course, Time</u> —> Prof
<u>Prof, Time</u> —> Course

What are the candidate keys?

1.) Prof, Time
2.) Course, Time

Functional Dependencies

sch(Teach) = (Course, Prof, Time)

redundancy!!

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Important assumption: each course is taught by exaclty one Prof.

What are the (interesting) functional dependencies?

Course —> Prof
<u>Course, Time</u> —> Prof
<u>Prof, Time</u> —> Course

What are the candidate keys?

1.) Prof, Time
2.) Course, Time

Functional Dependencies

Consider a table T with three columns:
sch(T) = (A, B, C)

How many different functional dependencies exist for such a table T?

a.) 8
b.) 27
c.) 49 ← correct
d.) 64?

$(2^3 - 1) * (2^3 - 1) = 7*7 = 49$

number of non-empty sets with at most 3 elements

a
b
c
ab
ac
bc
abc

7 nonempty subset of { a, b, c}

A functional dependency X → Y is completely non-rivial,

if $X \cap Y = \emptyset$.

How many completely non-trivial FD for a table w. 3 columns?
12

ab → c	b → a
ac → b	b → c
bc → a	b → ac
a → b	c → b
a → c	c → a
a → bc	c → ab

Still: if a → b holds, then we **do not care** that also ac → b holds.

The FD a → b **implies** ac → b (and it is strictly smaller).

#possibleFD(n columns) = $(2^n — 1)^2$

Question:
#possibleCompletelyNon-TrivialFD(n columns) = **?**

also exponential in n?

yes!

Evgenii Pavlov, MaST Mathematics, University of Cambridge (2020)

Answered September 25, 2016

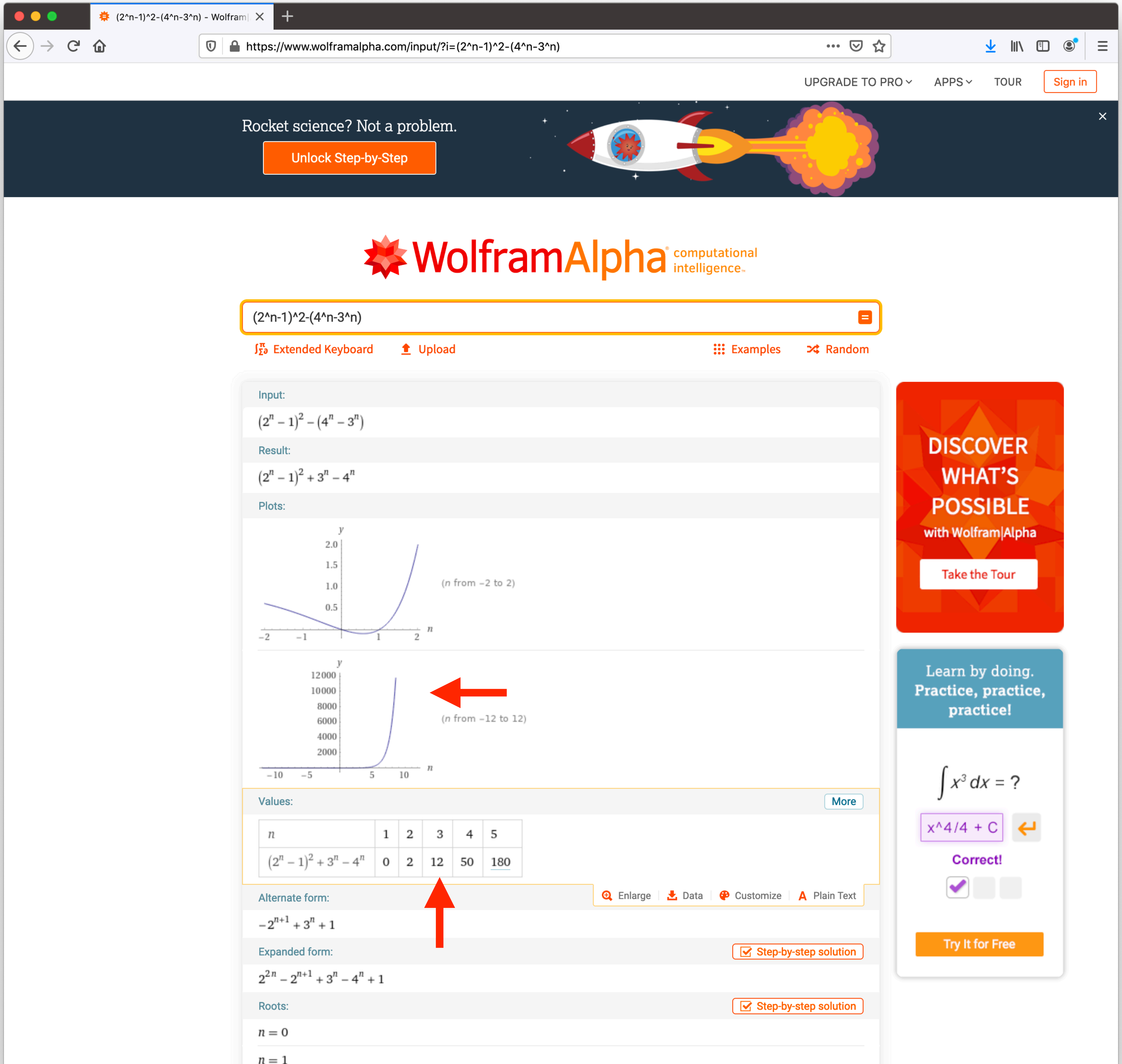
There are $2^n \times 2^n$ ways to choose two subsets without imposing any restrictions and 3^n ways to choose two non-intersecting subsets (for every element we choose whether it lies in the first set, the second set or in neither of the sets). This means that there are $4^n - 3^n$ ways to choose two intersecting subsets.

https://www.quora.com/In-how-many-ways-can-we-choose-two-subsets-from-an-n-elements-set-so-that-they-have-a-non-empty-intersection

#possibleFD(n columns) = $(2^n — 1)^2$

Question:
#possibleCompletelyNon-TrivialFD(n columns) = ?

also exponential in n?


(2^n-1)^2-(4^n-3^n) - Wolfram|
https://www.wolframalpha.com/input/?i=(2^n-1)^2-(4^n-3^n)
UPGRADE TO PRO
APPS
TOUR
Sign in
Rocket science? Not a problem.
Unlock Step-by-Step
WolframAlpha computational intelligence.
(2^n-1)^2-(4^n-3^n)
Extended Keyboard
Upload
Examples
Random
Input:
$(2^n - 1)^2 - (4^n - 3^n)$
Result:
$(2^n - 1)^2 + 3^n - 4^n$
Plots:
(n from −2 to 2)
(n from −12 to 12)
Values:
More
n | 1 | 2 | 3 | 4 | 5
$(2^n - 1)^2 + 3^n - 4^n$ | 0 | 2 | 12 | 50 | 180
Enlarge
Data
Customize
Plain Text
Alternate form:
$-2^{n+1} + 3^n + 1$
Expanded form:
Step-by-step solution
$2^{2n} - 2^{n+1} + 3^n - 4^n + 1$
Roots:
Step-by-step solution
$n = 0$
$n = 1$
DISCOVER WHAT'S POSSIBLE
with Wolfram|Alpha
Take the Tour
Learn by doing. Practice, practice, practice!
$\int x^3\,dx = ?$
x^4/4 + C
Correct!
Try It for Free

Armstrong Axioms

- **Reflexivity:** ("trivial functional dependencies")

  If $\beta \subseteq \alpha$ then $\alpha \rightarrow \beta$.

- **Augmentation:**

  If $\alpha \rightarrow \beta$ then $\alpha\gamma \rightarrow \beta\gamma$.

- **Transitivity:**

  If $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$ then $\alpha \rightarrow \gamma$.

Exercise: Show that $A_1 \dots A_n \to B_1 \dots B_m$ (#)

is **equivalent** to the m functional dependencies

$$A_1 \dots A_n \to B_1 \quad (1)$$
$$\vdots \qquad \vdots$$
$$A_1 \dots A_n \to B_m \quad (m)$$

using the Armstrong Axioms.

- **Reflexivity:** ("trivial functional dependencies")
  If $\beta \subseteq \alpha$ then $\alpha \to \beta$.
- **Augmentation:**
  If $\alpha \to \beta$ then $\alpha\gamma \to \beta\gamma$.
- **Transitivity:**
  If $\alpha \to \beta$ and $\beta \to \gamma$ then $\alpha \to \gamma$.

First, show that (#) implies (1)

{ B1 } $\subseteq$ { B1, … Bm }
B1, … Bm —> B1

Now use (#) and **Transitivity**!

Attribute Closure

Attribute Closure

The attribute closure $\alpha_{\mathcal{F}}^{+}$ can be computed as follows:

Algorithm: AttributeClosure

Input : α (a set of attributes); $\mathcal{F}$ (a set of FDs $\alpha_i \rightarrow \beta_i$)

Output: $\alpha_{\mathcal{F}}^{+}$ (all attributes functionally determined by α in $\mathcal{F}^{+}$)

```
x ← α;
repeat
    x' ← x;
    foreach αi → βi ∈ F do
        if αi ⊆ x then
            x ← x ∪ βi;
until x' = x;
return x;
```

Example

Given

$$\mathcal{F} = \{AB \rightarrow C, D \rightarrow E, AE \rightarrow G, GD \rightarrow H, ID \rightarrow J\}$$

for a relation $R, \text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \rightarrow GH$ **entailed by** $\mathcal{F}$**?**
- ✎ $ABD \rightarrow HJ$ **entailed by** $\mathcal{F}$**?**

Example

Given

$$\mathcal{F} = \{AB \rightarrow C, D \rightarrow E, AE \rightarrow G, GD \rightarrow H, ID \rightarrow J\}$$

for a relation $R, \text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \rightarrow GH$ **entailed by $\mathcal{F}$?**
- ✎ $ABD \rightarrow HJ$ **entailed by $\mathcal{F}$?**

— simply compute the attribute closure of ABD
— check if G,H is contained
— check if HJ is contaned

Example

Given

$$\mathcal{F} = \{AB \rightarrow C, D \rightarrow E, AE \rightarrow G, GD \rightarrow H, ID \rightarrow J\}$$

for a relation $R, \text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \rightarrow GH$ **entailed by $\mathcal{F}$?**
- ✎ $ABD \rightarrow HJ$ **entailed by $\mathcal{F}$?**

$$\{A, B, D\}^+_{\mathcal{F}} \Rightarrow$$

Example

Given

A,B "give us" C

$$\mathcal{F} = \{AB \to C, D \to E, AE \to G, GD \to H, ID \to J\}$$

for a relation R, $\text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \to GH$ **entailed by $\mathcal{F}$?**
- ✎ $ABD \to HJ$ **entailed by $\mathcal{F}$?**

$$\{A, B, D\}^+_{\mathcal{F}} \Rightarrow$$

Example

Given

$$\mathcal{F} = \{AB \rightarrow C, D \rightarrow E, AE \rightarrow G, GD \rightarrow H, ID \rightarrow J\}$$

for a relation R, $\text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \rightarrow GH$ **entailed by $\mathcal{F}$?**
- ✎ $ABD \rightarrow HJ$ **entailed by $\mathcal{F}$?**

$$\{A, B, D\}^+_{\mathcal{F}} \Rightarrow$$ { A, B, D, C }

Example

Given

D "gives us" E

$$\mathcal{F} = \{AB \rightarrow C, D \rightarrow E, AE \rightarrow G, GD \rightarrow H, ID \rightarrow J\}$$

for a relation R, $\text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \rightarrow GH$ **entailed by $\mathcal{F}$?**
- ✎ $ABD \rightarrow HJ$ **entailed by $\mathcal{F}$?**

$$\{A, B, D\}^+_{\mathcal{F}} \Rightarrow$$ { A, B, D, C }

Example

D "gives us" E

Given

$$\mathcal{F} = \{AB \to C, D \to E, AE \to G, GD \to H, ID \to J\}$$

for a relation R, $\text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \to GH$ **entailed by** $\mathcal{F}$**?**
- ✎ $ABD \to HJ$ **entailed by** $\mathcal{F}$**?**

$\{A, B, D\}^+_{\mathcal{F}} \Rightarrow$ { A, B, D, C }

==> { A, B, C, D, E }

Example

Given

A,E "give us" G

$$\mathcal{F} = \{AB \rightarrow C, D \rightarrow E, AE \rightarrow G, GD \rightarrow H, ID \rightarrow J\}$$

for a relation R, $\text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \rightarrow GH$ **entailed by** $\mathcal{F}$**?**
- ✎ $ABD \rightarrow HJ$ **entailed by** $\mathcal{F}$**?**

$\{A, B, D\}^+_{\mathcal{F}} \Rightarrow$ { A, B, D, C }

==> { A, B, C, D, E }

Example

Given

A,E “give us” G

$$\mathcal{F} = \{AB \to C, D \to E, AE \to G, GD \to H, ID \to J\}$$

for a relation R, $\text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \to GH$ **entailed by** $\mathcal{F}$**?**
- ✎ $ABD \to HJ$ **entailed by** $\mathcal{F}$**?**

$\{A, B, D\}^+_{\mathcal{F}} \Rightarrow$ { A, B, D, C }

==> { A, B, C, D, E }

==> { A, B, C, D, E, G }

Example

Given

G,D "give us" H

$$\mathcal{F} = \{AB \rightarrow C, D \rightarrow E, AE \rightarrow G, GD \rightarrow H, ID \rightarrow J\}$$

for a relation R, $\text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \rightarrow GH$ **entailed by** $\mathcal{F}$**?**
- ✎ $ABD \rightarrow HJ$ **entailed by** $\mathcal{F}$**?**

$\{A, B, D\}^+_{\mathcal{F}} \Rightarrow$ { A, B, D, C }

==> { A, B, C, D, E }

==> { A, B, C, D, E, G }

==> { A, B, C, D, E, G, H }

Example

Given

$$\mathcal{F} = \{AB \rightarrow C, D \rightarrow E, AE \rightarrow G, GD \rightarrow H, ID \rightarrow J\}$$

for a relation R, $\text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \rightarrow GH$ **entailed by** $\mathcal{F}$**?** Yes!
- ✎ $ABD \rightarrow HJ$ **entailed by** $\mathcal{F}$**?**

$\{A, B, D\}^+_\mathcal{F} \Rightarrow$ { A, B, D, C }

==> { A, B, C, D, E }

==> { A, B, C, D, E, G }

==> { A, B, C, D, E, G, H }

Example

Given

$$\mathcal{F} = \{AB \to C, D \to E, AE \to G, GD \to H, ID \to J\}$$

for a relation R, $\text{sch}(R) = ABCDEFGHIJ$.

- ✎ $ABD \to GH$ **entailed by $\mathcal{F}$?** Yes!
- ✎ $ABD \to HJ$ **entailed by $\mathcal{F}$?** No!

$\{A, B, D\}^+_{\mathcal{F}} \Rightarrow$ { A, B, D, C }

==> { A, B, C, D, E }

==> { A, B, C, D, E, G }

==> { A, B, C, D, E, G, H }

Minimal Cover

$\mathcal{F}^+$ is the **maximal cover** for $\mathcal{F}$.

→ $\mathcal{F}^+$ can be large and contain many redundant FDs. This makes $\mathcal{F}^+$ a poor basis to study a relational schema.

Thus: Construct a **minimal cover** $\mathcal{F}^-$ such that

1. $\mathcal{F}^- \equiv \mathcal{F}$, *i.e.*, $\left(\mathcal{F}^-\right)^+ = \mathcal{F}^+$.
2. All functional dependencies in $\mathcal{F}^-$ have the form $\alpha \rightarrow X$ (*i.e.*, the right side is a single attribute).
3. In $\alpha \rightarrow X \in \mathcal{F}^-$, no attributes in α are redundant:

$$\forall A \in \alpha : \left(\mathcal{F}^- - \{\alpha \rightarrow X\} \cup \{(\alpha - A) \rightarrow X\}\right) \not\equiv \mathcal{F}^- \quad .$$

4. No rule $\alpha \rightarrow X$ is redundant in $\mathcal{F}^-$:

$$\forall \alpha \rightarrow X \in \mathcal{F}^- : \left(\mathcal{F}^- - \{\alpha \rightarrow X\}\right) \not\equiv \mathcal{F}^- \quad .$$

Constructing a Minimal Cover

To construct the minimal cover $\mathcal{F}^-$:

1. $\mathcal{F}^- \leftarrow \mathcal{F}$ where all functional dependencies are converted to have only **one attribute on the right side**.
2. **Remove redundant attributes** from the left-hand sides of functional dependencies in $\mathcal{F}^-$:

    foreach $\alpha \rightarrow X \in \mathcal{F}^-$ **do**
        foreach $A \in \alpha$ **do**
            if $X \in (\alpha - A)^+_{\mathcal{F}^-}$ **then** *A redundant in α? Remove it.*
                $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \rightarrow X\} \cup \{(\alpha - A) \rightarrow X\}$;

3. **Remove redundant functional dependencies** from $\mathcal{F}^-$:

    foreach $\alpha \rightarrow X \in \mathcal{F}^-$ **do**
        if $(\mathcal{F}^- - \{\alpha \rightarrow X\}) \equiv \mathcal{F}^-$ **then**
            $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \rightarrow X\}$;

Constructing a Minimal Cover

A —> C
B —> C
A —> B
B —> A

$\mathcal{F}^-$

1.) check if first FD can be removed

$$(\mathcal{F}^- - \{A \rightarrow C\}) \stackrel{?}{\equiv} \mathcal{F}^-$$

how can we check this?

— any ideas?

3 Remove redundant functional dependencies from $\mathcal{F}^-$:

foreach $\alpha \rightarrow X \in \mathcal{F}^-$ **do**
  if $(\mathcal{F}^- - \{\alpha \rightarrow X\}) \equiv \mathcal{F}^-$ **then**
    $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \rightarrow X\}$;

Constructing a Minimal Cover

A —> C
B —> C
A —> B
B —> A

$\mathcal{F}^-$

1.) check if first FD can be removed

$$(\mathcal{F}^- - \{A \to C\}) \stackrel{?}{\equiv} \mathcal{F}^-$$

how can we check this?

— check if C is in the attribute closure of A under the reduced set of FDs!

$$C \stackrel{?}{\in} \{A\}^+_{\mathcal{F}^- - \{A \to C\}}$$

3 **Remove redundant functional dependencies** from $\mathcal{F}^-$:

1 **foreach** $\alpha \to X \in \mathcal{F}^-$ **do**
2 **if** $(\mathcal{F}^- - \{\alpha \to X\}) \equiv \mathcal{F}^-$ **then**
3 $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \to X\}$;

~~A —> C~~
B —> C
1 A —> B
B —> A

$(\mathcal{F}^- - \{A \rightarrow C\})$

A ==> AB
1

1.) check if first FD can be removed

$$(\mathcal{F}^- - \{A \rightarrow C\}) \overset{?}{\equiv} \mathcal{F}^-$$

how can we check this?

— check if C is in the attribute closure of A under the reduced set of FDs!

$$C \overset{?}{\in} \{A\}^+_{\mathcal{F}^- - \{A \rightarrow C\}}$$

3 **Remove redundant functional dependencies** from $\mathcal{F}^-$:

foreach $\alpha \rightarrow X \in \mathcal{F}^-$ **do**
  if $(\mathcal{F}^- - \{\alpha \rightarrow X\}) \equiv \mathcal{F}^-$ **then**
    $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \rightarrow X\}$;

Constructing a Minimal Cover

~~A —> C~~
2 B —> C
1 A —> B
B —> A

$(\mathcal{F}^- - \{A \to C\})$

A ==> AB ==> ABC
1 2

1.) check if first FD can be removed

$$(\mathcal{F}^- - \{A \to C\}) \stackrel{?}{\equiv} \mathcal{F}^-$$

how can we check this?

— check if C is in the attribute closure of A under the reduced set of FDs!

$$C \stackrel{?}{\in} \{A\}^+_{\mathcal{F}^- - \{A \to C\}}$$

3 **Remove redundant functional dependencies** from $\mathcal{F}^-$:

foreach $\alpha \to X \in \mathcal{F}^-$ **do**
  if $(\mathcal{F}^- - \{\alpha \to X\}) \equiv \mathcal{F}^-$ **then**
    $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \to X\}$;

~~A —> C~~
(2) B —> C
(1) A —> B
B —> A

$(\mathcal{F}^- - \{A \to C\})$

1.) check if first FD can be removed

$$(\mathcal{F}^- - \{A \to C\}) \stackrel{?}{\equiv} \mathcal{F}^-$$

how can we check this?

— check if C is in the attribute closure of A under the reduced set of FDs!

A ==> AB ==> ABC
(1) (2)

Yes!

$$C \stackrel{?}{\in} \{A\}^+_{\mathcal{F}^- - \{A \to C\}}$$

3 **Remove redundant functional dependencies** from $\mathcal{F}^-$:

foreach $\alpha \to X \in \mathcal{F}^-$ **do**
  if $(\mathcal{F}^- - \{\alpha \to X\}) \equiv \mathcal{F}^-$ **then**
    $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \to X\}$;

Constructing a Minimal Cover

A —> C
B —> C
A —> B
B —> A

$\equiv$

B —> C
A —> B
B —> A

3 **Remove redundant functional dependencies** from $\mathcal{F}^-$:

foreach $\alpha \rightarrow X \in \mathcal{F}^-$ **do**
  if $(\mathcal{F}^- - \{\alpha \rightarrow X\}) \equiv \mathcal{F}^-$ **then**
    $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \rightarrow X\}$;

Constructing a Minimal Cover

(1) A —> C	
(2) B —> C	B —> C
A —> B	A —> B
B —> A	B —> A

$\equiv$ (left set equivalent to right set)

|||

(2) B —> C
(1) A —> C
A —> B
B —> A

first two rules interchanged

3. **Remove redundant functional dependencies** from $\mathcal{F}^-$:

foreach $\alpha \to X \in \mathcal{F}^-$ **do**
  if $(\mathcal{F}^- - \{\alpha \to X\}) \equiv \mathcal{F}^-$ **then**
    $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \to X\}$;

Constructing a Minimal Cover

1	A —> C		
2	B —> C		B —> C
	A —> B	≡	A —> B
	B —> A		B —> A

$|||$ $\qquad$ $|||$

2	B —> C		A —> C
1	A —> C	≡	A —> B
	A —> B		B —> A
	B —> A		

first two rules interchanged

3. **Remove redundant functional dependencies** from $\mathcal{F}^-$:

foreach $\alpha \rightarrow X \in \mathcal{F}^-$ **do**
    if $(\mathcal{F}^- - \{\alpha \rightarrow X\}) \equiv \mathcal{F}^-$ **then**
        $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \rightarrow X\}$;

Constructing a Minimal Cover

(1) A —> C (2) B —> C A —> B B —> A	≡ B —> C A —> B B —> A
≡	≡
(2) B —> C (1) A —> C A —> B B —> A	≡ A —> C A —> B B —> A

first two rules interchanged

Both are Minimal Covers!

3. **Remove redundant functional dependencies** from $\mathcal{F}^-$:

foreach $\alpha \rightarrow X \in \mathcal{F}^-$ **do**
  if $(\mathcal{F}^- - \{\alpha \rightarrow X\}) \equiv \mathcal{F}^-$ **then**
    $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \rightarrow X\}$;

Constructing a Minimal Cover

① A —> C ② B —> C A —> B B —> A	≡	B —> C A —> B B —> A	
≡		≡	
② B —> C ① A —> C A —> B B —> A	≡	A —> C A —> B B —> A	

Both are
Minimal Covers!

==> Minimal Cover is **not unique**!

(depends on the visit order of the FDs)

first two rules interchanged

3. **Remove redundant functional dependencies** from $\mathcal{F}^-$:

```
foreach α → X ∈ F⁻ do
    if (F⁻ − {α → X}) ≡ F⁻ then
        F⁻ ← F⁻ − {α → X} ;
```

Constructing a Minimal Cover

To construct the minimal cover $\mathcal{F}^-$:

1. $\mathcal{F}^- \leftarrow \mathcal{F}$ where all functional dependencies are converted to have only **one attribute on the right side**.
2. **Remove redundant attributes** from the left-hand sides of functional dependencies in $\mathcal{F}^-$:

   foreach $\alpha \rightarrow X \in \mathcal{F}^-$ **do**
       foreach $A \in \alpha$ **do**
           if $X \in (\alpha - A)^+_{\mathcal{F}^-}$ **then** *A redundant in α? Remove it.*
               $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \rightarrow X\} \cup \{(\alpha - A) \rightarrow X\}$;

   Order is not specified.
   Not a deterministic algorithm!
   Order is up to you.

3. **Remove redundant functional dependencies** from $\mathcal{F}^-$:

   foreach $\alpha \rightarrow X \in \mathcal{F}^-$ **do**
       if $(\mathcal{F}^- - \{\alpha \rightarrow X\}) \equiv \mathcal{F}^-$ **then**
           $\mathcal{F}^- \leftarrow \mathcal{F}^- - \{\alpha \rightarrow X\}$;

   Note: for Steps 2 and 3, all you need is the **attribute closure**!

Constructing a Minimal Cover

✎ **Minimal cover for the following FDs?**

$$ABH \rightarrow C \qquad F \rightarrow AD \qquad C \rightarrow E \qquad E \rightarrow F$$
$$A \rightarrow D \qquad BGH \rightarrow F \qquad BH \rightarrow E$$

Homework

—> try to solve this on your own!

Will be discussed on Wednesday (Übung / Exercises).

2.) Information and Dependency Preservation

2.) Information and Dependency Preservation

ISBN	**Title**	**Author**
1-55860-570-3	Managing Gigabytes	Witten
1-55860-570-3	Managing Gigabytes	Moffat
1-55860-570-3	Managing Gigabytes	Bell
0-387-98210-8	Graph Theory	Diestel

fact stored more than once

We will remove redundancy, by **decomposing** a table into several new tables.

2.) Information and Dependency Preservation

ISBN	Title	Author
1-55860-570-3	Managing Gigabytes	Witten
1-55860-570-3	Managing Gigabytes	Moffat
1-55860-570-3	Managing Gigabytes	Bell
0-387-98210-8	Graph Theory	Diestel

decompose

ISBN	Title
1-55860-570-3	Managing Gigabytes

ISBN	Author
1-55860-570-3	Witten
1-55860-570-3	Moffat
1-55860-570-3	Bell

Schema Decomposition

As illustrated by example on the previous slide redundancy can be eliminated by **decomposing** a schema into a collection of schemas:

$$\big(\text{sch}(R), \mathcal{F}\big) \rightsquigarrow \big(\text{sch}(R_1), \mathcal{F}_1\big), \dots, \big(\text{sch}(R_n), \mathcal{F}_n\big) \enspace .$$

The corresponding relations can be obtained by **projecting** on columns of the original relation:

$$R_i = \pi_{\text{sch}(R_i)} R \enspace .$$

While decomposing a schema, we do **not** want to **lose information**.

Lossless and Lossy Decompositions

A decomposition is **lossless** if the original relation can be **reconstructed** from the decomposed tables via natural joins:

$$R = R_1 \bowtie \cdots \bowtie R_n \quad .$$

decomposition has lossless-join property

2.) Information and Dependency Preservation

ISBN	**Title**	**Author**
1-55860-570-3	Managing Gigabytes	Witten
1-55860-570-3	Managing Gigabytes	Moffat
1-55860-570-3	Managing Gigabytes	Bell
0-387-98210-8	Graph Theory	Diestel

NATURAL JOIN

<u>ISBN</u>	Title
1-55860-570-3	Managing Gigabytes
0-387-98210-8	Graph Theory

<u>ISBN</u>	Author
1-55860-570-3	Witten
1-55860-570-3	Moffat
1-55860-570-3	Bell
0-387-98210-8	Diestel

Dependency-Preserving Decompositions

For a lossless decomposition of R, it would always be possible to **re-construct** R and check the original set of FDs $\mathcal{F}$ over the re-constructed table.

→ But re-construction is **expensive**.

→ We'd rather like to guarantee that FDs $\mathcal{F}_1, \ldots, \mathcal{F}_n$ over decomposed tables $R_1, \ldots, R_n$ **entail all** FDs in $\mathcal{F}$.

A decomposition is **dependency-preserving** if

$$\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_n \equiv \mathcal{F} \quad .$$

Example

sch(Teach) = (Course, Prof, Time)

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Course —> Prof

Prof, Time —> Course

decompose

Course	Prof

Course	Time

Example

sch(Teach) = (Course, Prof, Time)

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Course —> Prof

Prof, Time —> Course

decompose

Course	Prof

Course	Time

— decomposition is **not** dependency preserving!
— the dependency (Prof, Time —> Course) is **lost**!

Decomposing A Schema

When decomposing a schema, we obtain schemas by **projecting** on columns of the original relation (↗ slide 237):

$$R_i = \pi_{\mathrm{sch}(R_i)} R \quad .$$

How do we obtain the corresponding functional dependencies?

$$\mathcal{F}_i := \pi_{\mathrm{sch}(R_i)} \mathcal{F} := \left\{ \alpha \to \beta \mid \alpha \to \beta \in \mathcal{F}^+ \text{ and } \alpha\beta \subseteq \mathrm{sch}(R_i) \right\}$$

→ We call this the **projection** of the set $\mathcal{F}$ of functional dependencies on the set of attributes $\mathrm{sch}(R_i)$.

Lemma

Let I be an instance over U satisfying $X \to Y$. Then $I = \pi_{XY}(I) \bowtie \pi_{XZ}(I)$ with $Z = U - XY$.

X	Y	Z

$=$

X	Y

$\bowtie$

X	Z

U = { X, Y, Z }

Redundancy

Relation Schema R with functional dependency $X \rightarrow A$
has **fd-redundancy** (with respect to $X \rightarrow A$) if

(1) there exists a db instance D over R that satisfies $X \rightarrow A$

(2) there exist two distinct tuples in D that have equal (X, A)-values.

3.) Third Normal Form

Third Normal Form (3NF)

Definition of 3NF

Whenever $X \rightarrow A$ is a nontrivial FD that holds, then either

— X is a superkey or

— A is a prime attribute.

Third Normal Form (3NF)

Definition of 3NF

Whenever X→A is a nontrivial FD that holds, then either

— X is a superkey or

— A is a prime attribute.

Candidate Key: { BuildingID }

Example (Not in 3NF)

Schema → {BuildingID, Contractor, Fee}

1. BuildingID → Contractor
2. Contractor → Fee
3. BuildingID → Fee

BuildingID	Contractor	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

Third Normal Form (3NF)

Definition of 3NF

Whenever X→A is a nontrivial FD that holds, then either

— X is a superkey or

— A is a prime attribute.

BuildingID	Contractor	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

Example (Not in 3NF)

Schema → {BuildingID, Contractor, Fee}

1. BuildingID → Contractor
2. Contractor → Fee
3. BuildingID → Fee
4. Both Contractor and Fee depend on the entire key hence 2NF

violation of 3NF!

Decomposition into 3NF

Definition of 3NF

Whenever X→A is a nontrivial FD that holds, then either

— X is a superkey or

— A is a prime attribute.

---

(1) compute a minimal cover **C** of the set **F** of FDs that hold

(2) for each FD X→A in **C**, create a new table
and choose X as primary key

(3) if none of the new tables contains any candidate key K of the original table, then add a new table with exactly the columns of K

(4) remove redundant tables (that are contained in others)

— **all FDs** in **C** are preserved in the new tables
(and the lossless join property holds)!!

Third Normal Form (3NF)

(1) compute a minimal cover **C** of the set **F** FDs that hold
(2) for each FD X→A in **C**, create a new table
and choose X as primary key

BuildingID	Contractor	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

→

BuildingID	Contractor
100	Randolph
150	Ingersoll
200	Randolph
250	Pitkin
300	Randolph

Contractor	Fee
Randolph	1200
Ingersoll	1100
Pitkin	1100

Third Normal Form (3NF)

Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

→ do you see any redundancy?

Definition of 3NF

Whenever X —> A is a nontrivial FD that holds, then either

— X is a superkey or

— A is a prime attribute.

Third Normal Form (3NF)

Tournament Winners

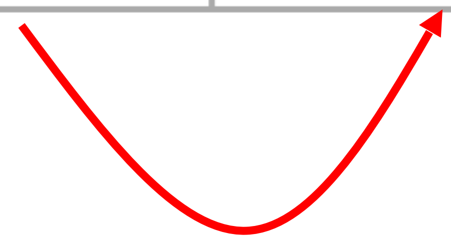
Tournament	**Year**	**Winner**	**Winner Date of Birth**
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

→ do you see any redundancy?

Third Normal Form (3NF)

Tournament Winners

Tournament	**Year**	**Winner**	**Winner Date of Birth**
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977



Tournament Winners

Tournament	**Year**	**Winner**	**Winner Date of Birth**
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

Tournament Winners

Tournament	**Year**	**Winner**
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Winner Dates of Birth

Winner	**Date of Birth**
Chip Masterson	14 March 1977
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968

Third Normal Form (3NF)

Question: can there still be redundancy wrt a FD
in a table that is in 3NF?

Third Normal Form (3NF)

Question: can there still be redundancy wrt a FD in a table that is in 3NF?

Course	Prof	Time
cs101	**Knuth**	Mo, 9-11
cs101	**Knuth**	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Course —> Prof
Course, Time —> Prof
Prof, Time —> Course

Is this relation in **3NF**?

Definition of 3NF

Whenever $X \rightarrow A$ is a nontrivial FD that holds, then either

— X is a superkey or

— A is a prime attribute.

Third Normal Form (3NF)

Question: can there still be redundancy wrt a FD in a table that is in 3NF?

Course	Prof	Time
cs101	**Knuth**	Mo, 9-11
cs101	**Knuth**	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

3NF may still contain redundancy

Course —> Prof
Course, Time —> Prof
Prof, Time —> Course

Is this relation in **3NF**?
Yes, it is in 3NF!

Definition of 3NF
Whenever $X \rightarrow A$ is a nontrivial FD that holds, then either
— X is a superkey or
— A is a prime attribute.

4. Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

→ BCNF does **not** allow dependencies between prime attributes!

BCNF = "3NF + no dependencies between (distinct) prime attributes"

4. Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

→ BCNF does **not** allow dependencies between prime attributes!

BCNF = "3NF + no dependencies between (distinct) prime attributes"

"… the key, the whole key, and nothing but the key, so help me Codd."

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

BCNF = "3NF + no dependencies between (distinct) prime attributes"

Example (Not in BCNF)

Schema { ISBN, Title, Author }

— FD: ISBN → Title

— ISBN is not super key

ISBN	Title	Author
1-55860-570-3	Managing Gigabytes	Witten
1-55860-570-3	Managing Gigabytes	Moffat
1-55860-570-3	Managing Gigabytes	Bell
0-387-98210-8	Graph Theory	Diestel

Boyce-Codd Normal Form (BCNF)

Bring table R into BCNF:

→ Place two candidate primary keys into separate tables
→ Place items in either of the tables, according to their dependencies on the keys

---

→ show how BCNF removes redundancy!

ISBN	Title	Author
1-55860-570-3	Managing Gigabytes	Witten
1-55860-570-3	Managing Gigabytes	Moffat
1-55860-570-3	Managing Gigabytes	Bell
0-387-98210-8	Graph Theory	Diestel

ISBN	Title
1-55860-570-3	Managing Gigabytes
0-387-98210-8	Graph Theory

ISBN	Author
1-55860-570-3	Witten
1-55860-570-3	Moffat
1-55860-570-3	Bell
0-387-98210-8	Diestel

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Is it in BCNF?

Course —> Prof

Prof, Time —> Course

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Is it in BCNF?

No!

Course —> Prof

Prof, Time —> Course

Algorithm for BCNF Decomposition

BCNF can be obtained by repeatedly **decomposing** a table **along an FD that violates BCNF**:

Algorithm: BCNFDecomposition
Input : $(\text{sch}(R), \mathcal{F})$
Output: Schema $\{(\text{sch}(R_1), \mathcal{F}_1), \ldots, (\text{sch}(R_n), \mathcal{F}_n)\}$ in BCNF
Decomposed $\leftarrow \{(\text{sch}(R), \mathcal{F})\}$;
while $\exists\,(\text{sch}(S), \mathcal{F}_S) \in$ *Decomposed* that is not in BCNF **do**
  Let $\alpha \rightarrow \beta$ be an FD in $\mathcal{F}_S$ that violates BCNF;
  Decompose S into $S_1(\alpha\beta)$ and $S_2((S - \beta) \cup \alpha)$;
return *Decomposed*;

Algorithm for BCNF Decomposition

BCNF can be obtained by repeatedly **decomposing** a table **along an FD that violates BCNF**:

Algorithm: BCNFDecomposition

Input : $(\text{sch}(R), \mathcal{F})$

Output: Schema $\{(\text{sch}(R_1), \mathcal{F}_1), \ldots, (\text{sch}(R_n), \mathcal{F}_n)\}$ in BCNF

$\mathit{Decomposed} \leftarrow \{(\text{sch}(R), \mathcal{F})\}$;

while $\exists (\text{sch}(S), \mathcal{F}_S) \in \mathit{Decomposed}$ that is not in BCNF **do**

  Let $\alpha \rightarrow \beta$ be an FD in $\mathcal{F}_S$ that violates BCNF;

  Decompose S into $S_1(\alpha\beta)$ and $S_2((S - \beta) \cup \alpha)$;

return $\mathit{Decomposed}$;

Notes: (1) This decomposition has the lossless-join property (= information preserving)
(2) But, it **may not** be dependency preserving!

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Course —> Prof

violates BCNF

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

<u>Course</u> —> Prof

violates BCNF

Course	Prof
cs101	Knuth
cs311	Knuth
cs477	Smith

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

Prof, Time —> Course

is **lost** in new tables!!

— cannot easily enforce this dependency anymore :-((

Course —> Prof

violates BCNF

In the old table, Prof is removed!

Course	Prof
cs101	Knuth
cs311	Knuth
cs477	Smith

Course	Time
cs101	Mo, 9-11
cs101	Fr, 14-16
cs311	Th, 8-10
cs477	Mo, 9-11

Course	Prof	Time
cs101	Knuth	Mo, 9-11
cs101	Knuth	Fr, 14-16
cs311	Knuth	Th, 8-10
cs477	Smith	Mo, 9-11

<u>Prof, Time</u> —> Course

is **lost** in new tables!!

— cannot easily enforce this dependency anymore :-((

<u>Course</u> —> Prof

violates BCNF

In the old table, Prof is removed!

<u>Course</u>	Prof
cs101	Knuth
cs311	Knuth
cs477	Smith

<u>Course</u>	<u>Time</u>
cs101	Mo, 9-11
cs101	Fr, 14-16
cs311	Th, 8-10
cs477	Mo, 9-11

Question: in MySQL/PostgreSQL, how to enforce (prof,time —> course) on the tables?

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF, if for any dependency X → Y at least one of the following holds

→ (X → Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R.

(by Boyce and Codd 1974)

Good News

Lemma If R is a relation schema in BCNF
then there are **<u>no fd-redundancies</u>** in R.

5. Fourth Normal Form (4NF)

5. Fourth Normal Form (4NF)

A table R is in 4NF, if for every multi-valued dependency (mvd) X -->> Y,

→ (X -->> Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R

[Fagin,1977]

R has multi-valued dependency (mvd) X -->> Y

If two tuples agree on all attributes in X, then their Y-values may be swapped, and the resulting two tuples must in R as well.

Note $X \rightarrow Y$ implies X -->> Y.

Fourth Normal Form (4NF)

A table R is in 4NF, if for every multi-valued dependency (mvd) X -->> Y,

→ (X -->> Y) is trivial (i.e., Y is a subset of X)
→ X is a superkey for R

[Fagin,1977]

Example (Not in 4NF)

Schema → {Movie, ScreeningCity, Genre)

Primary Key: (Movie, ScreeningCity, Genre)

1. All columns are a part of the only candidate key, hence BCNF
2. Many Movies can have the same Genre
3. Many Cities can have the same movie
4. A Movie can have several Genres
4. Violates 4NF

Movie	ScreeningCity	Genre
Hard Code	Los Angles	Comedy
Hard Code	New York	Comedy
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	Drama
The Code Warrier	New York	Horror
The Code Warrier	New York	Sci-Fi

Movie -->> ScreeningCity
Movie -->> Genre

Fourth Normal Form (4NF)

Example 2 (Not in 4NF)

Schema → {Manager, Child, Employee}

1. Primary Key → {Manager, Child, Employee}
2. Each manager can have more than one child
3. Each manager can supervise more than one employee
4. 4NF Violated

Manager	Child	Employee
Jim	Beth	Alice
Mary	Bob	Jane
Mary	Bob	Adam

Manager -->> Child
Manager -->> Employee

Example 3 (Not in 4NF)

Schema → {Employee, Skill, ForeignLanguage}

1. Primary Key → {Employee, Skill, Language }
2. Each employee can speak multiple languages
3. Each employee can have multiple skills
4. Thus violates 4NF

Employee	Skill	Language
1234	Cooking	French
1234	Cooking	German
1453	Carpentry	Spanish
1453	Cooking	Spanish
2345	Cooking	Spanish

Fourth Normal Form (4NF)

Bring a BCNF table into 4NF:
→ Move the two multi-valued sub-relations into separate tables
→ Identify primary keys for each new table.

Example 1 (Convert to 4NF)

Old Schema → {MovieName, ScreeningCity, Genre}

New Schema → {MovieName, ScreeningCity}

New Schema → {MovieName, Genre}

Movie	ScreeningCity	Genre
Hard Code	Los Angles	Comedy
Hard Code	New York	Comedy
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	Drama
The Code Warrier	New York	Horror
The Code Warrier	New York	Sci-Fi

Movie	Genre
Hard Code	Comedy
Bill Durham	Drama
The Code Warrier	Horror
The Code Warrier	Sci-Fi

Movie	ScreeningCity
Hard Code	Los Angles
Hard Code	New York
Bill Durham	Santa Cruz
Bill Durham	Durham
The Code Warrier	New York

Example 2 (Convert to 4NF)

Old Schema → {Manager, Child, Employee}

New Schema → {Manager, Child}

New Schema → {Manager, Employee}

Manager	Child
Jim	Beth
Mary	Bob

Manager	Employee
Jim	Alice
Mary	Jane
Mary	Adam

Example 3 (Convert to 4NF)

Old Schema → {Employee, Skill, ForeignLanguage}

New Schema → {Employee, Skill}

New Schema → {Employee, ForeignLanguage}

Employee	Skill
1234	Cooking
1453	Carpentry
1453	Cooking
2345	Cooking

Employee	Language
1234	French
1234	German
1453	Spanish
2345	Spanish

Fourth Normal Form (4NF)

Do not underestimate importance of 4NF:

→ [Wu 1992] of real word databases, 20% were **NOT** in 4NF!

(all of them were in 3NF)

The Practical Need for Fourth Normal Form

Margaret S. Wu
425 Beldon Ave
Iowa City, Iowa 52246
Phone: (319) 335-0846

ABSTRACT

Many practitioners and academicians believe that data violating fourth normal form is rarely encountered. We report upon a study of forty organizational databases; nine of them contained data violating fourth normal form. Consequently, the need to understand and use fourth normal form is more important than previously believed.

INTRODUCTION

A paramount issue in the design of any database is what data fields should be grouped together into records. In the relational model, the data fields are grouped into logical structures called relations. The determination of which data fields are placed together in a relation is based upon the concept of normal forms; the process is known as normalization. The set of data fields comprising the database is progressively organized into relations in first

that the relation has no modification anomalies.

There is some evidence that academicians view fourth normal form (4NF) as unimportant and thus may neglect the topic in database management courses in MIS. Stamper and Price in [10] state that "fourth and fifth normal forms are so rarely encountered in business applications as to be almost obscure; hence, they are not described in this book." Mittra [8] states that "Although

Author: Margaret S. Wu 425 Beldon Ave, Iowa City, Iowa

Published in:

- Proceeding
  SIGCSE '92 Proceedings of the twenty-third SIGCSE technical symposium on Computer science education
  Pages 19-23

  Kansas City, Missouri, USA — March 05 - 06, 1992
  ACM New York, NY, USA ©1992
  table of contents ISBN:0-89791-468-6 doi>10.1145/134510.134515

- Newsletter
  ACM SIGCSE Bulletin Homepage
  Volume 24 Issue 1, March 1992
  Pages 19-23
  ACM New York, NY, USA
  table of contents doi>10.1145/135250.134515

Redundancy

Relation Schema R with multi-valued dependency X -->> A
has **mvd-redundancy** (with respect to X -->> A) if

(1) there exists a db instance D over R that satisfies X -->> A

(2) there exist two distinct tuples in D that have equal (X, A)-values.

Good News

Lemma If R is a relation schema in 4NF,
then there are no mvd-redundancies in R

Kurs

Datenbankgrund… und Modell…

End of this lecture

Sebast… …rsität Bremen
…remen.de

…hersemester 2023

5.6.2023
Vorlesung 6: Normal Forms