# Data Modelling/Data Base Systems

## VU 184.685/VU 184.686, WS 2020

## Relational Design Theory – Normalforms

Anela Lolić

Institute of Logic and Computation, TU Wien

# Acknowledgements

The slides are based on the slides (in German) of Sebastian Skritek.

The content is based on Chapter 6 of
(Kemper, Eickler: Datenbanksysteme – Eine Einführung).

For related literature in English see Chapter 19 of
(Ramakrishnan, Gehrke: Database Management Systems).

# Overview

- Aims
- Functional Dependencies
  - Definitions
  - Canonical Cover
- Design Theory and Decomposition
  - "Bad" Relational Schemata
  - Decomposition of Relational Schemata
  - Criteria for a "meaningful" Decomposition
- Normalforms (1., 2., 3., Boyce-Codd)
  - Normalization through Synthesis Algorithm
  - Normalization through Decomposition

# Design Theory and Decomposition

- "Bad" Relational Schemata
- Decomposition of Relational Schemata
- Criteria for a "meaningful" Decomposition
    - Lossless-Join Decomposition
    - Dependency-Preserving Decomposition

# "Bad" Relational Schemata

| **ProfLec** | | | | | | |
|---|---|---|---|---|---|---|
| persNr | name | rank | room | lecNr | title | SWS |
| 2125 | Sokrates | C4 | 226 | 5041 | Ethik | 4 |
| 2125 | Sokrates | C4 | 226 | 5049 | Mäeutik | 2 |
| 2125 | Sokrates | C4 | 226 | 4052 | Logik | 4 |
| ... | ... | ... | ... | ... | ... | ... |
| 2132 | Popper | C3 | 52 | 5295 | Der Wiener Kreis | 2 |
| 2137 | Kant | C4 | 7 | 4630 | Die drei Kritiken | 4 |

# "Bad" Relational Schemata

| ProfLec | | | | | | |
|---|---|---|---|---|---|---|
| persNr | name | rank | room | lecNr | title | SWS |
| 2125 | Sokrates | C4 | 226 | 5041 | Ethik | 4 |
| 2125 | Sokrates | C4 | 226 | 5049 | Mäeutik | 2 |
| 2125 | Sokrates | C4 | 226 | 4052 | Logik | 4 |
| ... | ... | ... | ... | ... | ... | ... |
| 2132 | Popper | C3 | 52 | 5295 | Der Wiener Kreis | 2 |
| 2137 | Kant | C4 | 7 | 4630 | Die drei Kritiken | 4 |

update anomaly: Sokrates moves

# "Bad" Relational Schemata

| **ProfLec** | | | | | | |
|---|---|---|---|---|---|---|
| persNr | name | rank | room | lecNr | title | SWS |
| 2125 | Sokrates | C4 | 226 | 5041 | Ethik | 4 |
| 2125 | Sokrates | C4 | 226 | 5049 | Mäeutik | 2 |
| 2125 | Sokrates | C4 | 226 | 4052 | Logik | 4 |
| ... | ... | ... | ... | ... | ... | ... |
| 2132 | Popper | C3 | 52 | 5295 | Der Wiener Kreis | 2 |
| 2137 | Kant | C4 | 7 | 4630 | Die drei Kritiken | 4 |

update anomaly: Sokrates moves

deletion anomaly: ex. "Die 3 Kritiken" does not take place

# “Bad” Relational Schemata

| **ProfLec** | | | | | | |
|---|---|---|---|---|---|---|
| persNr | name | rank | room | lecNr | title | SWS |
| 2125 | Sokrates | C4 | 226 | 5041 | Ethik | 4 |
| 2125 | Sokrates | C4 | 226 | 5049 | Mäeutik | 2 |
| 2125 | Sokrates | C4 | 226 | 4052 | Logik | 4 |
| ... | ... | ... | ... | ... | ... | ... |
| 2132 | Popper | C3 | 52 | 5295 | Der Wiener Kreis | 2 |
| 2137 | Kant | C4 | 7 | 4630 | Die drei Kritiken | 4 |

update anomaly: Sokrates moves

deletion anomaly: ex. “Die 3 Kritiken” does not take place

insertion anomaly: ex. Curie is new and does not give any lectures yet (has no key?!)

# Decomposition of Relational Schemata

- anomalies are based on the fact, that data is stored that does not correspond to each other

# Decomposition of Relational Schemata

- anomalies are based on the fact, that data is stored that does not correspond to each other
- intuitively:
  - all information to professors is stored in a relation,
  - all information to lectures is stored in a relation,
  - the information about the connection of the two relations is stored in a relation

# Decomposition of Relational Schemata

- anomalies are based on the fact, that data is stored that does not correspond to each other
- intuitively:
  - all information to professors is stored in a relation,
  - all information to lectures is stored in a relation,
  - the information about the connection of the two relations is stored in a relation
- solution: decomposition of the schema into sub schemata

# Decomposition of Relational Schemata

- anomalies are based on the fact, that data is stored that does not correspond to each other
- intuitively:
  - all information to professors is stored in a relation,
  - all information to lectures is stored in a relation,
  - the information about the connection of the two relations is stored in a relation
- solution: decomposition of the schema into sub schemata
- question: what is a “meaningful’ decomposition?

# Decomposition of Relational Schemata

## Example

decomposition of

profLec(persNr, name, rank, room, lecNr, title, SWS)

in sub schemata

# Decomposition of Relational Schemata

## Example

decomposition of

profLec(persNr, name, rank, room, lecNr, title, SWS)

in sub schemata

1 attempt: decomposition in:

- prof(persNr, name, rank, room)
- lecture(lecNr, title, SWS)

# Decomposition of Relational Schemata

## Example

decomposition of

profLec(persNr, name, rank, room, lecNr, title, SWS)

in sub schemata

1 attempt: decomposition in:

- prof(persNr, name, rank, room)
- lecture(lecNr, title, SWS)

problem: Who is teaching which lecture?

cause: sub schemata cannot be correctly connected any more

# Decomposition of Relational Schemata

## Example

decomposition of

profLec(persNr, name, rank, room, lecNr, title, SWS)

in sub schemata

# Decomposition of Relational Schemata

## Example

decomposition of

profLec(persNr, name, rank, room, lecNr, title, SWS)

in sub schemata

2 attempt: decomposition in:

- prof(persNr, name, room, lecNr, title, SWS )
- title(name, rank)

# Decomposition of Relational Schemata

## Example

decomposition of

profLec(persNr, name, rank, room, lecNr, title, SWS)

in sub schemata

2 attempt: decomposition in:

- prof(persNr, name, room, lecNr, title, SWS )
- title(name, rank)

problem: rank of professors with the same names

cause: the FD $\{\text{persNr}\} \rightarrow \{\text{rank}\}$ is lost

# Decomposition of Relational Schemata – Soundness

- a decomposition of a relational schema $\mathcal{R}$ is a set of schemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$ such that:

$$att(\mathcal{R}_1) \cup \cdots \cup att(\mathcal{R}_n) = att(\mathcal{R})$$

# Decomposition of Relational Schemata – Soundness

- a decomposition of a relational schema $\mathcal{R}$ is a set of schemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$ such that:

$$att(\mathcal{R}_1) \cup \cdots \cup att(\mathcal{R}_n) = att(\mathcal{R})$$

- there are two soundness criteria for the decomposition of relational schemata:

  Lossless-Join Decomposition (loss of information)
  Dependency-Preserving Decomposition (loss of meta information)

# Decomposition of Relational Schemata – Soundness

## Definition ( Lossless-Join Decomposition - intuitively)

Information occurring in the state $R$ of the schema $\mathcal{R}$ has to be reconstructible from the states $R_1, \ldots, R_n$ of the new schemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$.

# Decomposition of Relational Schemata – Soundness

## Definition ( Lossless-Join Decomposition - intuitively)

Information occurring in the state $R$ of the schema $\mathcal{R}$ has to be reconstructible from the states $R_1, \ldots, R_n$ of the new schemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$.

## Definition (Dependency-Preserving Decomposition - intuitively)

functional dependencies on $\mathcal{R}$ have to be transformable to the schemata $\mathcal{R}_1, \ldots, \mathcal{R}_n$.

# Decomposition of Relational Schemata – Lossless-Join Decomposition

## Definition ( Lossless-Join Decomposition)

Given a relation schema $(\mathcal{R}, F_{\mathcal{R}})$.
A decomposition $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$ of $\mathcal{R}$ is said to be a lossless-join decomposition, if for every instance $R$ of $\mathcal{R}$ that satisfies $F_{\mathcal{R}}$ it holds that:

$$R = \pi_{\mathcal{R}_1}(R) \bowtie \pi_{\mathcal{R}_2}(R) \bowtie \cdots \bowtie \pi_{\mathcal{R}_n}(R)$$

# Decomposition of Relational Schemata – Lossless-Join Decomposition

## Definition ( Lossless-Join Decomposition)

Given a relation schema $(\mathcal{R}, F_{\mathcal{R}})$.
A decomposition $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$ of $\mathcal{R}$ is said to be a lossless-join decomposition, if for every instance $R$ of $\mathcal{R}$ that satisfies $F_{\mathcal{R}}$ it holds that:

$$R = \pi_{\mathcal{R}_1}(R) \bowtie \pi_{\mathcal{R}_2}(R) \bowtie \cdots \bowtie \pi_{\mathcal{R}_n}(R)$$

## Theorem (sufficient condition for lossless-join decomposition)

*A decomposition of $\mathcal{R}$ in $\mathcal{R}_1$ and $\mathcal{R}_2$ is said to be a lossless-join decomposition, if the join attributes are (super-) keys in one of the sub relations.*

# Decomposition of Relational Schemata – Lossless-Join Decomposition

## Example (lossless-join decomposition 1)

decomposition of profLec(persNr, name, rank, lecNr, title, SWS) in several sub schemata:

| **profLec** | | | | | |
|---|---|---|---|---|---|
| persNr | name | rank | lecNr | title | SWS |
| 2125 | Sokrates | C4 | 5041 | Ethik | 4 |
| 2125 | Sokrates | C4 | 5049 | Mäeutik | 2 |
| 2125 | Sokrates | C4 | 4052 | Logik | 4 |
| ... | ... | ... | ... | ... | ... |
| 2132 | Popper | C3 | 5295 | Der Wiener Kreis | 2 |
| 2137 | Kant | C4 | 4630 | Die drei Kritiken | 4 |

# Decomposition of Relational Schemata – Lossless-Join Decomposition

## Example (lossless-join decomposition 1)

first attempt:

| **professors** | | |
|---|---|---|
| <u>persNr</u> | name | rank |
| 2125 | Sokrates | C4 |
| 2132 | Popper | C3 |
| 2137 | Kant | C4 |
| ... | ... | ... |

| **lectures** | | |
|---|---|---|
| <u>lecNr</u> | title | SWS |
| 5041 | Ethik | 4 |
| 5049 | Mäeutik | 2 |
| 4052 | Logik | 4 |
| 5295 | Der Wiener Kreis | 2 |
| 4630 | Die drei Kritiken | 4 |
| ... | ... | ... |

question: profLec = professors $\bowtie$ lectures ?

# Decomposition of Relational Schemata – Lossless-Join Decomposition

## Example (lossless-join decomposition 1)

first attempt:

**professors**

| persNr | name | rank |
|---|---|---|
| 2125 | Sokrates | C4 |
| 2132 | Popper | C3 |
| 2137 | Kant | C4 |
| ... | ... | ... |

**lectures**

| lecNr | title | SWS |
|---|---|---|
| 5041 | Ethik | 4 |
| 5049 | Mäeutik | 2 |
| 4052 | Logik | 4 |
| 5295 | Der Wiener Kreis | 2 |
| 4630 | Die drei Kritiken | 4 |
| ... | ... | ... |

question: profLec = professors $\bowtie$ lectures ?

no: the combination via join degenerates to cross product due to the lack of join attributes.

# Decomposition of Relational Schemata – lossless-Join decomposition

## Example (lossless-join decomposition 1)

second attempt:

**professors**

| persNr | name | rank |
|---|---|---|
| 2125 | Sokrates | C4 |
| 2132 | Popper | C3 |
| 2137 | Kant | C4 |
| ... | ... | ... |

**lectures**

| lecNr | title | SWS | persNr |
|---|---|---|---|
| 5041 | Ethik | 4 | 2125 |
| 5049 | Mäeutik | 2 | 2125 |
| 4052 | Logik | 4 | 2125 |
| 5295 | Der Wiener Kreis | 2 | 2132 |
| 4630 | Die drei Kritiken | 4 | 2137 |
| ... | ... | ... | ... |

question: profLec = professors $\bowtie$ lectures ?

# Decomposition of Relational Schemata – lossless-Join decomposition

## Example (lossless-join decomposition 1)

second attempt:

| professors | | |
|---|---|---|
| persNr | name | rank |
| 2125 | Sokrates | C4 |
| 2132 | Popper | C3 |
| 2137 | Kant | C4 |
| ... | ... | ... |

| lectures | | | |
|---|---|---|---|
| lecNr | title | SWS | persNr |
| 5041 | Ethik | 4 | 2125 |
| 5049 | Mäeutik | 2 | 2125 |
| 4052 | Logik | 4 | 2125 |
| 5295 | Der Wiener Kreis | 2 | 2132 |
| 4630 | Die drei Kritiken | 4 | 2137 |
| ... | ... | ... | ... |

question: profLec = professors $\bowtie$ lectures ?

yes: the join attribute persNr is key in professors $\Rightarrow$ sufficient condition is satisfied
$\Rightarrow$ lossless-join decomposition

# Decomposition of Relational Schemata – Lossless-Join Decomposition

## Example (lossless-join decomposition 2)

decompose relation parents(father, mother, <u>child</u>) in two sub relations:

$\mathcal{R}_1$ fathers(father, <u>child</u>)

$\mathcal{R}_2$ mothers(mother, <u>child</u>)

question: lossless-join decomposition?

# Decomposition of Relational Schemata – Lossless-Join Decomposition

## Example (lossless-join decomposition 2)

decompose relation parents(father, mother, <u>child</u>) in two sub relations:

$\mathcal{R}_1$ fathers(father, <u>child</u>)

$\mathcal{R}_2$ mothers(mother, <u>child</u>)

question: lossless-join decomposition?

yes: the join attribute of fathers and mothers is child
$\Rightarrow$ lossless-join decomposition guaranteed

# Decomposition of Relational Schemata – Lossless-Join Decomposition

## Example (lossless-join decomposition 2)

decompose relation parents(father, mother, child) in two sub relations:

$\mathcal{R}_1$ fathers(father, child)

$\mathcal{R}_2$ mothers(mother, child)

question: lossless-join decomposition?

yes: the join attribute of fathers and mothers is child
$\Rightarrow$ lossless-join decomposition guaranteed

remark: the decomposition of parents is a lossless-join decomposition but also not really needed, as the relation is in a very good state (normalform)

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

- let $F$ be a set of FDs over $\mathcal{R}$, and $\mathcal{R}' \subseteq \mathcal{R}$:

$$F[\mathcal{R}'] = \{\alpha \to \beta \in F \mid \alpha \cup \beta \subseteq \mathcal{R}'\}$$

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

- let $F$ be a set of FDs over $\mathcal{R}$, and $\mathcal{R}' \subseteq \mathcal{R}$:

$$F[\mathcal{R}'] = \{\alpha \to \beta \in F \mid \alpha \cup \beta \subseteq \mathcal{R}'\}$$

## Definition (dependency-preserving decomposition)

Given a relation schema $(\mathcal{R}, F)$.
A decomposition $\mathcal{R}_1, \ldots, \mathcal{R}_n$ of $\mathcal{R}$ is said to be a dependency-preserving decomposition, if

$$F \equiv (F^+[\mathcal{R}_1] \cup \cdots \cup F^+[\mathcal{R}_n]) \text{ i.e. } F^+ = (F^+[\mathcal{R}_1] \cup \cdots \cup F^+[\mathcal{R}_n])^+$$

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

- let $F$ be a set of FDs over $\mathcal{R}$, and $\mathcal{R}' \subseteq \mathcal{R}$:

$$F[\mathcal{R}'] = \{\alpha \to \beta \in F \mid \alpha \cup \beta \subseteq \mathcal{R}'\}$$

## Definition (dependency-preserving decomposition)

Given a relation schema $(\mathcal{R}, F)$.
A decomposition $\mathcal{R}_1, \ldots, \mathcal{R}_n$ of $\mathcal{R}$ is said to be a dependency-preserving decomposition, if

$$F \equiv (F^+[\mathcal{R}_1] \cup \cdots \cup F^+[\mathcal{R}_n]) \text{ i.e. } F^+ = (F^+[\mathcal{R}_1] \cup \cdots \cup F^+[\mathcal{R}_n])^+$$

alternatively: dependency-preserving decomposition if there is a set $G$ of FDs, such that $G \equiv F$ and $F \equiv (G[\mathcal{R}_1] \cup \cdots \cup G[\mathcal{R}_n])$

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example (no dependency-preserving decomposition)

Given a relation schema address(street, town, state, zipcode) and the following assumptions:

- towns are identified uniquely by name and state
- zipAreas remain within towns $\Rightarrow$ $\{\text{zipcode}\} \rightarrow \{\text{town, state}\}$
- zipcode does not change within a street $\Rightarrow$ $\{\text{street, town, state}\} \rightarrow \{\text{zipcode}\}$.

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example (no dependency-preserving decomposition)

Given a relation schema address(street, town, state, zipcode) and the following assumptions:

- towns are identified uniquely by name and state
- zipAreas remain within towns $\Rightarrow$ $\{\text{zipcode}\} \rightarrow \{\text{town, state}\}$
- zipcode does not change within a street $\Rightarrow$ $\{\text{street, town, state}\} \rightarrow \{\text{zipcode}\}$.

decomposition in:

$\mathcal{R}_1$ streets($\underline{\text{zipcode, street}}$)

$\mathcal{R}_2$ towns($\underline{\text{zipcode}}$, town, state) with FD $\{\text{zipcode}\} \rightarrow \{\text{town, state}\}$

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example (no dependency-preserving decomposition)

Given a relation schema address(street, town, state, zipcode) and the following assumptions:

- towns are identified uniquely by name and state
- zipAreas remain within towns $\Rightarrow$ $\{\text{zipcode}\} \to \{\text{town, state}\}$
- zipcode does not change within a street $\Rightarrow$ $\{\text{street, town, state}\} \to \{\text{zipcode}\}$.

decomposition in:

$\mathcal{R}_1$ streets(<u>zipcode, street</u>)

$\mathcal{R}_2$ towns(<u>zipcode</u>, town, state) with FD $\{\text{zipcode}\} \to \{\text{town, state}\}$

BUT: $\{\text{street, town, state}\} \to \{\text{zipcode}\}$ is lost
(lossless-join decomposition, as zipcode is key in towns)

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example (no dependency-preserving decomposition)

| **address** | | | |
|---|---|---|---|
| town | state | street | zipcode |
| Frankfurt | Hessen | Goethestraße | 60313 |
| Frankfurt | Hessen | Schillerstraße | 60437 |
| Frankfurt | Brandenburg | Goethestraße | 15234 |

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example (no dependency-preserving decomposition)

**address**

| town | state | street | zipcode |
|---|---|---|---|
| Frankfurt | Hessen | Goethestraße | 60313 |
| Frankfurt | Hessen | Schillerstraße | 60437 |
| Frankfurt | Brandenburg | Goethestraße | 15234 |

**streets**

| zipcode | street |
|---|---|
| 60313 | Goethestraße |
| 60437 | Schillerstraße |
| 15234 | Goethestraße |
| 15235 | Goethestraße |

**towns**

| town | state | zipcode |
|---|---|---|
| Frankfurt | Hessen | 60313 |
| Frankfurt | Hessen | 60437 |
| Frankfurt | Brandenburg | 15234 |
| Frankfurt | Brandenburg | 15235 |

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example ( $F^+[\mathcal{R}_i]$?)

Given a relation schema $(\mathcal{R}, F)$ with

$$\begin{aligned} \mathcal{R} &= \text{ancestors(child, mother, grandmother) and} \\ F &= \\ &\{\{\text{child}\} \rightarrow \{\text{mother}\}, \{\text{child, mother}\} \rightarrow \{\text{grandmother}\}\} \end{aligned}$$

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example ( $F^+[\mathcal{R}_i]$?)

Given a relation schema $(\mathcal{R}, F)$ with

$$\begin{aligned}\mathcal{R} &= \text{ancestors(child, mother, grandmother) and} \\ F &= \\ &\{\{\text{child}\} \to \{\text{mother}\}, \{\text{child, mother}\} \to \{\text{grandmother}\}\}\end{aligned}$$

decomposition in

$\mathcal{R}_1$ mothers(child, mother),

$\mathcal{R}_2$ grandmothers(child, grandmother),

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example ( $F^+[\mathcal{R}_i]$?)

Given a relation schema $(\mathcal{R}, F)$ with

$$\begin{aligned} \mathcal{R} &= \text{ancestors(child, mother, grandmother) and} \\ F &= \\ &\{\{\text{child}\} \rightarrow \{\text{mother}\}, \{\text{child, mother}\} \rightarrow \{\text{grandmother}\}\} \end{aligned}$$

decomposition in

$\mathcal{R}_1$ mothers(child, mother), $F[\mathcal{R}_1] = \{\{\text{child}\} \rightarrow \{\text{mother}\}\}$

$\mathcal{R}_2$ grandmothers(child, grandmother), $F[\mathcal{R}_2] = \emptyset$

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example ( $F^+[\mathcal{R}_i]$?)

Given a relation schema $(\mathcal{R}, F)$ with

$$\mathcal{R} = \text{ancestors(child, mother, grandmother) and}$$
$$F = \{\{\text{child}\} \rightarrow \{\text{mother}\}, \{\text{child, mother}\} \rightarrow \{\text{grandmother}\}\}$$

decomposition in

$\mathcal{R}_1$ mothers(child, mother), $F[\mathcal{R}_1] = \{\{\text{child}\} \rightarrow \{\text{mother}\}\}$

$\mathcal{R}_2$ grandmothers(child, grandmother), $F[\mathcal{R}_2] = \emptyset$

$(F[\mathcal{R}_1] \cup F[\mathcal{R}_2])^+ = \{\{\text{child}\} \rightarrow \{\text{mother}\}, \ldots\} \neq F^+$

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example ( $F^+[\mathcal{R}_i]$?)

Given a relation schema $(\mathcal{R}, F)$ with

$$\mathcal{R} = \text{ancestors(child, mother, grandmother) and}$$
$$F = \{\{\text{child}\} \to \{\text{mother}\}, \{\text{child, mother}\} \to \{\text{grandmother}\}\}$$

decomposition in

$\mathcal{R}_1$ mothers(child, mother), $F[\mathcal{R}_1] = \{\{\text{child}\} \to \{\text{mother}\}\}$
$F^+[\mathcal{R}_1] = \{\{\text{child}\} \to \{\text{mother}\}, \dots\}$

$\mathcal{R}_2$ grandmothers(child, grandmother), $F[\mathcal{R}_2] = \emptyset$

$$(F[\mathcal{R}_1] \cup F[\mathcal{R}_2])^+ = \{\{\text{child}\} \to \{\text{mother}\}, \dots\} \neq F^+$$

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example ( $F^+[\mathcal{R}_i]$?)

Given a relation schema $(\mathcal{R}, F)$ with

$$\mathcal{R} = \text{ancestors(child, mother, grandmother) and}$$
$$F = \{\{\text{child}\} \rightarrow \{\text{mother}\}, \{\text{child, mother}\} \rightarrow \{\text{grandmother}\}\}$$

decomposition in

$\mathcal{R}_1$ mothers(child, mother), $F[\mathcal{R}_1] = \{\{\text{child}\} \rightarrow \{\text{mother}\}\}$
$F^+[\mathcal{R}_1] = \{\{\text{child}\} \rightarrow \{\text{mother}\}, \dots\}$

$\mathcal{R}_2$ grandmothers(child, grandmother), $F[\mathcal{R}_2] = \emptyset$
$F^+[\mathcal{R}_2] = \{\{\text{child}\} \rightarrow \{\text{grandmother}\}, \dots\}$

$$(F[\mathcal{R}_1] \cup F[\mathcal{R}_2])^+ = \{\{\text{child}\} \rightarrow \{\text{mother}\}, \dots\} \neq F^+$$

# Decomposition of Relational Schemata – Dependency-Preserving Decomposition

## Example ( $F^+[\mathcal{R}_i]$?)

Given a relation schema $(\mathcal{R}, F)$ with

$$\begin{aligned} \mathcal{R} &= \text{ancestors(child, mother, grandmother) and} \\ F &= \\ &\{\{\text{child}\} \rightarrow \{\text{mother}\}, \{\text{child, mother}\} \rightarrow \{\text{grandmother}\}\} \end{aligned}$$

decomposition in

- $\mathcal{R}_1$ mothers(child, mother), $F[\mathcal{R}_1] = \{\{\text{child}\} \rightarrow \{\text{mother}\}\}$
  $F^+[\mathcal{R}_1] = \{\{\text{child}\} \rightarrow \{\text{mother}\}, \dots\}$
- $\mathcal{R}_2$ grandmothers(child, grandmother), $F[\mathcal{R}_2] = \emptyset$
  $F^+[\mathcal{R}_2] = \{\{\text{child}\} \rightarrow \{\text{grandmother}\}, \dots\}$

$(F[\mathcal{R}_1] \cup F[\mathcal{R}_2])^+ = \{\{\text{child}\} \rightarrow \{\text{mother}\}, \dots\} \neq F^+$
$(F^+[\mathcal{R}_1] \cup F^+[\mathcal{R}_2])^+ = \{\{\text{child}\} \rightarrow \{\text{mother}\}, \{\text{child}\} \rightarrow \{\text{grandmother}\}, \{\text{child, mother}\} \rightarrow \{\text{grandmother}\}, \dots\} = F^+$

# Normalforms

- 1. Normalform
  - $NF^2$ Relations
- 2. Normalform
- 3. Normalform
  - Decomposition in 3NF
  - Synthesis Algorithm
- Boyce-Codd Normalform
  - Decomposition in BCNF
  - Decomposition Algorithm

# 1. Normalform

## Definition (1. normalform)

A relation schema $\mathcal{R}$ is in 1. normalform, if the domains of $\mathcal{R}$ are atomic.

# 1. Normalform

## Definition (1. normalform)

A relation schema $\mathcal{R}$ is in 1. normalform, if the domains of $\mathcal{R}$ are atomic.

| parents | | |
|---|---|---|
| father | mother | children |
| Johann | Martha | {Else, Lucie} |
| Heinz | Martha | {Cleo} |

not in 1NF

| parents | | |
|---|---|---|
| father | mother | child |
| Johann | Martha | Else |
| Johann | Martha | Lucie |
| Heinz | Martha | Cleo |

in 1NF

# $NF^2$ Relations

- non-first normalform relations
- attribute of a relation can be a relation itself
- contained in SQL2003 standard

# NF$^2$ Relations

- non-first normalform relations
- attribute of a relation can be a relation itself
- contained in SQL2003 standard

| parents | | | |
|---|---|---|---|
| father | mother | children | |
| | | cName | cAge |
| Johann | Martha | Else | 5 |
| | | Lucie | 3 |
| Johann | Maria | Theo | 3 |
| | | Josef | 1 |
| Heinz | Martha | Cleo | 9 |

# 2. Normalform

- intuitively: relation schema $\mathcal{R}$ violates the 2. normalform, if information about more than one concept is modelled in the relation
- only of historical interest, as anomalies are still possible

# 3. Normalform

## Definition (3. normalform)

A relation schema $\mathcal{R}$ is in third normalform, if for every on $\mathcal{R}$ holding FD of the form $\alpha \rightarrow B$ with $\alpha \subseteq \mathcal{R}$ and $B \in \mathcal{R}$ at least one of the following three conditions hold:

1. $B \in \alpha$, i.e., the FD is trivial
2. $\alpha$ is super key of $\mathcal{R}$
3. the attribute $B$ is contained in one of the keys of $\mathcal{R}$

# 3. Normalform – “Alternative” Definition

original definition of Codd (1971):

A relation schema is in 3NF if it is in 2NF and if no non-key attribute transitively depends on a key.

# 3. Normalform – “Alternative” Definition

original definition of Codd (1971):

A relation schema is in 3NF if it is in 2NF and if no non-key attribute transitively depends on a key.

i.e. the following conditions have to be satisfied:

- 2NF
- there are no attribute sets $\alpha$ (keys), $\beta$ (non-keys) and attribute $A$ (not contained in a key), such that
  - $\alpha \rightarrow \beta$ and $\beta \rightarrow A$ holds and
  - $\beta \not\rightarrow \alpha$, $A \notin \alpha$, $A \notin \beta$

# 3. Normalform – "Alternative" Definition

original definition of Codd (1971):

A relation schema is in 3NF if it is in 2NF and if no non-key attribute transitively depends on a key.

i.e. the following conditions have to be satisfied:

- 2NF
- there are no attribute sets $\alpha$ (keys), $\beta$ (non-keys) and attribute $A$ (not contained in a key), such that
  - $\alpha \rightarrow \beta$ and $\beta \rightarrow A$ holds and
  - $\beta \not\rightarrow \alpha$, $A \notin \alpha$, $A \notin \beta$

*"Every non-key attribute must provide a fact about the key, the whole key, and nothing but the key. (So help me Codd)"*

# 3. Normalform

## Example

profLec(persNr, name, rank, room, lecNr, title, SWS)

$F_d$ = {persNr → name rank room,
lecNr → persNr name rank room lecNr title SWS}

**profLec**

| persNr | name | rank | room | lecNr | title | SWS |
|---|---|---|---|---|---|---|
| 2125 | Sokrates | C4 | 226 | 5041 | Ethik | 4 |
| 2125 | Sokrates | C4 | 226 | 5049 | Mäeutik | 2 |
| 2125 | Sokrates | C4 | 226 | 4052 | Logik | 4 |
| ... | ... | ... | ... | ... | ... | ... |
| 2132 | Popper | C3 | 52 | 5295 | Der Wiener Kreis | 2 |
| 2137 | Kant | C4 | 7 | 4630 | Die drei Kritiken | 4 |

not in 3. normalform

# 3. Normalform

## Example (check for 3NF 1)

$$\mathcal{R} = (ABCDEF),$$
$$F = \{C \to BDAE\} = \{C \to B, C \to D, C \to A, C \to E\}.$$

# 3. Normalform

## Example (check for 3NF 1)

$$\mathcal{R} = (ABCDEF),$$
$$F = \{C \to BDAE\} = \{C \to B, C \to D, C \to A, C \to E\}.$$

the only key of $\mathcal{R}$ is: $CF$

# 3. Normalform

## Example (check for 3NF 1)

$$\mathcal{R} = (ABCDEF),$$
$$F = \{C \to BDAE\} = \{C \to B, C \to D, C \to A, C \to E\}.$$

the only key of $\mathcal{R}$ is: $CF$

$(\mathcal{R}, F)$ in 3NF, if for every FD one of the NF-conditions holds:

1. Is the FD trivial? No, condition 1 does not hold for any FDs.

# 3. Normalform

## Example (check for 3NF 1)

$$\mathcal{R} = (ABCDEF),$$
$$F = \{C \to BDAE\} = \{C \to B, C \to D, C \to A, C \to E\}.$$

the only key of $\mathcal{R}$ is: $CF$

$(\mathcal{R}, F)$ in 3NF, if for every FD one of the NF-conditions holds:

1. Is the FD trivial? No, condition 1 does not hold for any FDs.
2. Is the attribute set on the left side of the FD a super key of $\mathcal{R}$? No, condition 2 does not hold for any FDs.

# 3. Normalform

## Example (check for 3NF 1)

$$\mathcal{R} = (ABCDEF),$$
$$F = \{C \to BDAE\} = \{C \to B, C \to D, C \to A, C \to E\}.$$

the only key of $\mathcal{R}$ is: $CF$

$(\mathcal{R}, F)$ in 3NF, if for every FD one of the NF-conditions holds:

1. Is the FD trivial? No, condition 1 does not hold for any FDs.
2. Is the attribute set on the left side of the FD a super key of $\mathcal{R}$? No, condition 2 does not hold for any FDs.
3. Is the attribute on the right side of the FD contained in a key of $\mathcal{R}$? No, condition 3 does not hold for any FDs.

# 3. Normalform

## Example (check for 3NF 1)

$$\mathcal{R} = (ABCDEF),$$
$$F = \{C \to BDAE\} = \{C \to B, C \to D, C \to A, C \to E\}.$$

the only key of $\mathcal{R}$ is: $CF$

$(\mathcal{R}, F)$ in 3NF, if for every FD one of the NF-conditions holds:

1. Is the FD trivial? No, condition 1 does not hold for any FDs.
2. Is the attribute set on the left side of the FD a super key of $\mathcal{R}$? No, condition 2 does not hold for any FDs.
3. Is the attribute on the right side of the FD contained in a key of $\mathcal{R}$? No, condition 3 does not hold for any FDs.

at least one FD violates the 3NF $\Rightarrow$ $\mathcal{R}$ not in 3NF.

# 3. Normalform

## Example (check for 3NF 2)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \rightarrow B, C \rightarrow D, D \rightarrow A, D \rightarrow E, E \rightarrow C, E \rightarrow F, F \rightarrow E\}$$

# 3. Normalform

## Example (check for 3NF 2)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E\}$$

the keys of $\mathcal{R}$ are: $C, E, F, D$

# 3. Normalform

## Example (check for 3NF 2)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \rightarrow B, C \rightarrow D, D \rightarrow A, D \rightarrow E, E \rightarrow C, E \rightarrow F, F \rightarrow E\}$$

the keys of $\mathcal{R}$ are: $C, E, F, D$

1. Is the FD trivial? No, condition 1 does not hold for any FDs.

# 3. Normalform

## Example (check for 3NF 2)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E\}$$

the keys of $\mathcal{R}$ are: $C, E, F, D$

1. Is the FD trivial? No, condition 1 does not hold for any FDs.
2. Is the attribute set on the left side of the FD a super key of $\mathcal{R}$? OK for $C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E$.

# 3. Normalform

## Example (check for 3NF 2)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E\}$$

the keys of $\mathcal{R}$ are: $C, E, F, D$

1. Is the FD trivial? No, condition 1 does not hold for any FDs.
2. Is the attribute set on the left side of the FD a super key of $\mathcal{R}$? OK for $C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E$.
3. Is the attribute on the right side of the FD contained in one of the keys of $\mathcal{R}$? No for $C \to B, D \to A$
   but OK for $C \to D, D \to E, E \to CF, F \to E$.

# 3. Normalform

## Example (check for 3NF 2)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E\}$$

the keys of $\mathcal{R}$ are: $C, E, F, D$

1. Is the FD trivial? No, condition 1 does not hold for any FDs.
2. Is the attribute set on the left side of the FD a super key of $\mathcal{R}$? OK for $C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E$.
3. Is the attribute on the right side of the FD contained in one of the keys of $\mathcal{R}$? No for $C \to B, D \to A$
   but OK for $C \to D, D \to E, E \to CF, F \to E$.

no FD violates the 3NF $\Rightarrow$ in 3NF
(In this case this is clear after step 2.)

# 3. Normalform

## Example

profLec(persNr, name, rank, room, lecNr, title, SWS)

$$F_d = \{\text{persNr} \rightarrow \text{name rank room}, \\ \text{lecNr} \rightarrow \text{persNr name rank room lecNr title SWS}\}$$

**profLec**

| persNr | name | rank | room | lecNr | title | SWS |
|---|---|---|---|---|---|---|
| 2125 | Sokrates | C4 | 226 | 5041 | Ethik | 4 |
| 2125 | Sokrates | C4 | 226 | 5049 | Mäeutik | 2 |
| 2125 | Sokrates | C4 | 226 | 4052 | Logik | 4 |
| ... | ... | ... | ... | ... | ... | ... |
| 2132 | Popper | C3 | 52 | 5295 | Der Wiener Kreis | 2 |
| 2137 | Kant | C4 | 7 | 4630 | Die drei Kritiken | 4 |

not in 3. normalform

# Decomposition in 3NF

## Theorem

*A relation schema $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ is in third normalform, if all $\mathcal{R}_i$ are in third normalform.*

# Decomposition in 3NF

## Theorem

*A relation schema $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ is in third normalform, if all $\mathcal{R}_i$ are in third normalform.*

- given a relation schema $\mathcal{R}$ with FDs $F$.
- find: decomposition in sub schemata $\mathcal{R}_1 \ldots \mathcal{R}_n$, such that:
  - lossless-join decomposition in $\mathcal{R}_1 \ldots \mathcal{R}_n$,
  - dependency-preserving decomposition in $\mathcal{R}_1 \ldots \mathcal{R}_n$,
  - all $\mathcal{R}_1 \ldots \mathcal{R}_n$ are in third normalform.

# Decomposition in 3NF

## Theorem

*A relation schema $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ is in third normalform, if all $\mathcal{R}_i$ are in third normalform.*

- given a relation schema $\mathcal{R}$ with FDs $F$.
- find: decomposition in sub schemata $\mathcal{R}_1 \ldots \mathcal{R}_n$, such that:
  - lossless-join decomposition in $\mathcal{R}_1 \ldots \mathcal{R}_n$,
  - dependency-preserving decomposition in $\mathcal{R}_1 \ldots \mathcal{R}_n$,
  - all $\mathcal{R}_1 \ldots \mathcal{R}_n$ are in third normalform.
- solution: synthesis algorithm

# The Synthesis Algorithm

1. construct the canonical cover $F_c$ for $F$

# The Synthesis Algorithm

1. construct the canonical cover $F_c$ for $F$
2. for each FD $\alpha \rightarrow \beta \in F_c$:
    - construct a relation schema $\mathcal{R}_i := \alpha \cup \beta$
    - assign to $\mathcal{R}_\alpha$ the FDs $F_i := F_c[\mathcal{R}_i]$

      informally: from every FD one schema (condition for dependency-preserving decomposition is satisfied) - canonical cover important to ensure not creating too many or too big schemata.

# The Synthesis Algorithm

1. construct the canonical cover $F_c$ for $F$
2. for each FD $\alpha \rightarrow \beta \in F_c$:
   - construct a relation schema $\mathcal{R}_i := \alpha \cup \beta$
   - assign to $\mathcal{R}_\alpha$ the FDs $F_i := F_c[\mathcal{R}_i]$

   informally: from every FD one schema (condition for dependency-preserving decomposition is satisfied) - canonical cover important to ensure not creating too many or too big schemata.
3. does one of the sub schemata constructed in step 2 contain a key from $\mathcal{R}$ w.r.t. $F_c \Rightarrow$ step 4
   otherwise: choose one key $\kappa \in \mathcal{R}$ and define the following additional schema: $\mathcal{R}_\kappa := \kappa$ with $F_\kappa := \emptyset$

   informally: construct a schema to connect sub schemata (condition for lossless-join decomposition satisfied)

# The Synthesis Algorithm

1. construct the canonical cover $F_c$ for $F$
2. for each FD $\alpha \to \beta \in F_c$:
   - construct a relation schema $\mathcal{R}_i := \alpha \cup \beta$
   - assign to $\mathcal{R}_\alpha$ the FDs $F_i := F_c[\mathcal{R}_i]$

     informally: from every FD one schema (condition for dependency-preserving decomposition is satisfied) - canonical cover important to ensure not creating too many or too big schemata.
3. does one of the sub schemata constructed in step 2 contain a key from $\mathcal{R}$ w.r.t. $F_c \Rightarrow$ step 4

   otherwise: choose one key $\kappa \in \mathcal{R}$ and define the following additional schema: $\mathcal{R}_\kappa := \kappa$ with $F_\kappa := \emptyset$

   informally: construct a schema to connect sub schemata (condition for lossless-join decomposition satisfied)
4. eliminate the schemata $\mathcal{R}_i$ contained in another schema $\mathcal{R}_j$

   informally: shorten superfluous schemata

# The Synthesis Algorithm

## Example (synthesis algorithm 1)

$\mathcal{R} = \{persNr, name, rank, room, town, street, zipcode, Vorwahl, state, population, Landesregierung\} = \{P, N, R, Z, O, S, zip, V, B, E, L\}$ and the FDs from before

# The Synthesis Algorithm

## Example (synthesis algorithm 1)

$\mathcal{R} = \{persNr, name, rank, room, town, street, zipcode, Vorwahl, state, population, Landesregierung\} = \{P, N, R, Z, O, S, zip, V, B, E, L\}$ and the FDs from before

1. canonical cover:
   $P \rightarrow NRZOSB$, $Z \rightarrow P$, $SBO \rightarrow zip$, $OB \rightarrow EV$, $B \rightarrow L$, $zip \rightarrow BO$

# The Synthesis Algorithm

## Example (synthesis algorithm 1)

$\mathcal{R} = \{persNr, name, rank, room, town, street, zipcode, Vorwahl, state, population, Landesregierung\} = \{P, N, R, Z, O, S, zip, V, B, E, L\}$ and the FDs from before

1. canonical cover:
   $P \to NRZOSB$, $Z \to P$, $SBO \to zip$, $OB \to EV$, $B \to L$, $zip \to BO$
2. generate sub schemata and assign all FDs:

$$\begin{aligned}
\{PNRZOSB\} &: P \to NRZOSB \text{ and } Z \to P \\
\{ZP\} &: Z \to P \\
\{SBOzip\} &: SBO \to zip \text{ and } zip \to BO \\
\{OBEV\} &: OB \to EV \\
\{BL\} &: B \to L \\
\{zipBO\} &: zip \to BO
\end{aligned}$$

# The Synthesis Algorithm

## Example (synthesis algorithm 1)

3. Does any of the sub schemata contain a key of $\mathcal{R}$ w.r.t. $F$? Yes: $P$ was a key and is contained in $\{PNRZOSB\}$ $\Rightarrow$ done

# The Synthesis Algorithm

## Example (synthesis algorithm 1)

3. Does any of the sub schemata contain a key of $\mathcal{R}$ w.r.t. $F$? Yes: $P$ was a key and is contained in $\{PNRZOSB\} \Rightarrow$ done
4. schema elimination:

   $\{ZP\}$ is already contained in $\{PNRZOSB\} \Rightarrow$ shorten

   $\{zipBO\}$ is already contained in $\{SBOzip\} \Rightarrow$ shorten

# The Synthesis Algorithm

## Example (synthesis algorithm 1)

3. Does any of the sub schemata contain a key of $\mathcal{R}$ w.r.t. $F$? Yes: $P$ was a key and is contained in $\{PNRZOSB\} \Rightarrow$ done
4. schema elimination:

   $\{ZP\}$ is already contained in $\{PNRZOSB\} \Rightarrow$ shorten

   $\{zipBO\}$ is already contained in $\{SBOzip\} \Rightarrow$ shorten

5. result:

   professors: {[persNr, name, rank, room, town, street, state]}
   zipcodeList: {[street, town, state,zipcode]}
   cities: {[town, state, population, area code]}
   government: {[state, state government]}

# The Synthesis Algorithm

## Example (synthesis algorithm 2)

$\mathcal{R} = (ABCDEF), F = \{A \to EC, BC \to F, D \to B\}$
question: in 3 NF?

# The Synthesis Algorithm

## Example (synthesis algorithm 2)

$\mathcal{R} = (ABCDEF), F = \{A \to EC, BC \to F, D \to B\}$

question: in 3 NF? key: AD

# The Synthesis Algorithm

## Example (synthesis algorithm 2)

$\mathcal{R} = (ABCDEF), F = \{A \rightarrow EC, BC \rightarrow F, D \rightarrow B\}$
question: in 3 NF? no key: AD

# The Synthesis Algorithm

## Example (synthesis algorithm 2)

$\mathcal{R} = (ABCDEF), F = \{A \rightarrow EC, BC \rightarrow F, D \rightarrow B\}$
question: in 3 NF? no key: AD

1. is already canonical

# The Synthesis Algorithm

## Example (synthesis algorithm 2)

$\mathcal{R} = (ABCDEF), F = \{A \rightarrow EC, BC \rightarrow F, D \rightarrow B\}$
question: in 3 NF? no key: AD

1. is already canonical
2. $\mathcal{R}_1 = (AEC), \mathcal{R}_2 = (BCF), \mathcal{R}_3 = (DB)$
   no (non-trivial) assignment possible

# The Synthesis Algorithm

## Example (synthesis algorithm 2)

$\mathcal{R} = (ABCDEF), F = \{A \rightarrow EC, BC \rightarrow F, D \rightarrow B\}$
question: in 3 NF? no key: AD

1. is already canonical
2. $\mathcal{R}_1 = (AEC), \mathcal{R}_2 = (BCF), \mathcal{R}_3 = (DB)$
   no (non-trivial) assignment possible
3. key of $\mathcal{R}$: $AD$ add $\mathcal{R}_4 = (AD)$

# The Synthesis Algorithm

## Example (synthesis algorithm 2)

$\mathcal{R} = (ABCDEF), F = \{A \rightarrow EC, BC \rightarrow F, D \rightarrow B\}$
question: in 3 NF? no key: AD

1. is already canonical
2. $\mathcal{R}_1 = (AEC), \mathcal{R}_2 = (BCF), \mathcal{R}_3 = (DB)$
   no (non-trivial) assignment possible
3. key of $\mathcal{R}$: $AD$ add $\mathcal{R}_4 = (AD)$
4. nothing to be eliminated

# The Synthesis Algorithm

## Example (synthesis algorithm 2)

$\mathcal{R} = (ABCDEF), F = \{A \to EC, BC \to F, D \to B\}$
question: in 3 NF? no key: AD

1. is already canonical
2. $\mathcal{R}_1 = (AEC), \mathcal{R}_2 = (BCF), \mathcal{R}_3 = (DB)$
   no (non-trivial) assignment possible
3. key of $\mathcal{R}$: $AD$ add $\mathcal{R}_4 = (AD)$
4. nothing to be eliminated

$$\mathcal{R} = (AEC) \cup (BCF) \cup (DB) \cup (AD)$$

# The Synthesis Algorithm

## Example

Why do we need a canonical cover?

profLec(persNr, name, rank, room, lecNr, title, SWS)

$F_1$ : {persNr $\rightarrow$ name rank room,
lecNr $\rightarrow$ persNr name rank room lecNr title SWS}

# The Synthesis Algorithm

## Example

Why do we need a canonical cover?

profLec(persNr, name, rank, room, lecNr, title, SWS)

$F_1$: {persNr → name rank room,
lecNr → persNr name rank room lecNr title SWS}

$F_2$: {persNr → name, persNr → rank, persNr → room,
lecNr → persNr, lecNr → name, lecNr → rank,
lecNr → room, lecNr → lecNr, lecNr → title,
lecNr → SWS}

# Is 3 NF enough?

## Example

$$\mathcal{R} = \text{cities}(\text{town}, \text{land}, \text{boss}, \text{population})$$

$$F = \{\{\text{town}, \text{land}\} \rightarrow \{\text{population}\}, \{\text{land}\} \rightarrow \{\text{boss}\}, \{\text{boss}\} \rightarrow \{\text{land}\}\}$$

key:

# Is 3 NF enough?

## Example

$$\mathcal{R} = \text{cities(town, land, boss, population)}$$

$$F = \{\{\text{town, land}\} \rightarrow \{\text{population}\}, \{\text{land}\} \rightarrow \{\text{boss}\}, \{\text{boss}\} \rightarrow \{\text{land}\}\}$$

key: {town, land}, {town, boss}

# Is 3 NF enough?

## Example

$$\mathcal{R} = \text{cities(town, land, boss, population)}$$
$$F = \{\{\text{town, land}\} \to \{\text{population}\}, \{\text{land}\} \to \{\text{boss}\}, \{\text{boss}\} \to \{\text{land}\}\}$$

key: $\{$town, land$\}$, $\{$town, boss$\}$ $\Rightarrow$ in 3NF

**towns**

| town | land | boss | population |
|---|---|---|---|
| Minas Tirith | Gondor | Denethor | 2002 |
| Osgiliath | Gondor | Denethor | 0 |
| Dol Amroth | Gondor | Denethor | 700 |
| Barad-dûr | Mordor | Sauron | 20 |

# Boyce-Codd Normalform

## Definition (Boyce-Codd normalform)

A relation schema $\mathcal{R}$ is in Boyce-Codd normalform, if for every on $\mathcal{R}$ holding FD of the form $\alpha \rightarrow B$ with $\alpha \subseteq \mathcal{R}$ and $B \in \mathcal{R}$ at least one of the following two conditions hold:

1. $B \in \alpha$, i.e., the FD is trivial
2. $\alpha$ is super key of $\mathcal{R}$.

# Boyce-Codd Normalform

## Definition (Boyce-Codd normalform)

A relation schema $\mathcal{R}$ is in Boyce-Codd normalform, if for every on $\mathcal{R}$ holding FD of the form $\alpha \rightarrow B$ with $\alpha \subseteq \mathcal{R}$ and $B \in \mathcal{R}$ at least one of the following two conditions hold:

1. $B \in \alpha$, i.e., the FD is trivial
2. $\alpha$ is super key of $\mathcal{R}$.

## Theorem (normalforms correlation)

*The normalforms are connected in the following way:*

$$BCNF \subseteq 3NF \subseteq 2NF \subseteq 1NF$$

# Boyce-Codd Normalform

## Example (check for BCNF 1)

$$\mathcal{R} = (ABCDEF)$$

$$F = \{C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E\}$$

# Boyce-Codd Normalform

## Example (check for BCNF 1)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E\}$$

the keys of $\mathcal{R}$ are: $C, E, D, F$

# Boyce-Codd Normalform

## Example (check for BCNF 1)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E\}$$

the keys of $\mathcal{R}$ are: $C, E, D, F$
$(\mathcal{R}, F)$ is in BCNF, if for every FD one of the two NF-conditions holds:

1. no FD is trivial: condition 1 does not hold

# Boyce-Codd Normalform

## Example (check for BCNF 1)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E\}$$

the keys of $\mathcal{R}$ are: $C, E, D, F$
$(\mathcal{R}, F)$ is in BCNF, if for every FD one of the two NF-conditions holds:

1. no FD is trivial: condition 1 does not hold
2. Is the attribute set on the left side of the FDs a super key of $\mathcal{R}$? Yes, condition 2 holds for all FDs.

# Boyce-Codd Normalform

## Example (check for BCNF 1)

$$\mathcal{R} = (ABCDEF)$$
$$F = \{C \to B, C \to D, D \to A, D \to E, E \to C, E \to F, F \to E\}$$

the keys of $\mathcal{R}$ are: $C, E, D, F$

$(\mathcal{R}, F)$ is in BCNF, if for every FD one of the two NF-conditions holds:

1. no FD is trivial: condition 1 does not hold
2. Is the attribute set on the left side of the FDs a super key of $\mathcal{R}$? Yes, condition 2 holds for all FDs.

no FD violates the BCNF $\Rightarrow$ in BCNF

# Boyce-Codd Normalform

## Example (check for BCNF 2)

$$\mathcal{R} = \text{cities}(\text{town}, \text{land}, \text{boss}, \text{population})$$

$$F = \{\{\text{town}, \text{land}\} \rightarrow \{\text{population}\}, \{\text{land}\} \rightarrow \{\text{boss}\}, \{\text{boss}\} \rightarrow \{\text{land}\}\}$$

# Boyce-Codd Normalform

## Example (check for BCNF 2)

$$\begin{aligned}\mathcal{R} &= \text{cities(town, land, boss, population)}\\ F &= \{\{\text{town, land}\} \to \{\text{population}\},\\ &\quad \{\text{land}\} \to \{\text{boss}\}, \{\text{boss}\} \to \{\text{land}\}\}\end{aligned}$$

key: {town, land} resp. {town, boss}

# Boyce-Codd Normalform

## Example (check for BCNF 2)

$$\mathcal{R} = \text{cities(town, land, boss, population)}$$
$$F = \{\{\text{town, land}\} \rightarrow \{\text{population}\}, \{\text{land}\} \rightarrow \{\text{boss}\}, \{\text{boss}\} \rightarrow \{\text{land}\}\}$$

key: {town, land} resp. {town, boss}

1. no FD is trivial

# Boyce-Codd Normalform

## Example (check for BCNF 2)

$$\mathcal{R} = \text{cities(town, land, boss, population)}$$

$$F = \{\{\text{town, land}\} \to \{\text{population}\}, \{\text{land}\} \to \{\text{boss}\}, \{\text{boss}\} \to \{\text{land}\}\}$$

key: $\{\text{town, land}\}$ resp. $\{\text{town, boss}\}$

1. no FD is trivial
2. Is the attribute set on the left side of the FDs a super key?No for land $\to$ boss, boss $\to$ land

# Boyce-Codd Normalform

## Example (check for BCNF 2)

$$\mathcal{R} = \text{cities(town, land, boss, population)}$$
$$F = \{\{\text{town, land}\} \rightarrow \{\text{population}\},\ \{\text{land}\} \rightarrow \{\text{boss}\}, \{\text{boss}\} \rightarrow \{\text{land}\}\}$$

key: $\{$town, land$\}$ resp. $\{$town, boss$\}$

1. no FD is trivial
2. Is the attribute set on the left side of the FDs a super key?No for land $\rightarrow$ boss, boss $\rightarrow$ land

$\Rightarrow \mathcal{R}$ not in BCNF

# Boyce-Codd Normalform

## Example (check for BCNF 2)

$\mathcal{R}$ = cities(town, land, boss, population)

F = {{town, land} → {population},
{land} → {boss}, {boss} → {land}}

key: {town, land} resp. {town, boss}

1. no FD is trivial
2. Is the attribute set on the left side of the FDs a super key?No for land → boss, boss → land

⇒ $\mathcal{R}$ not in BCNF
But: in 3NF, as condition 2 (3NF): on the right side is a key attribute is satisfied.

# Decomposition in BCNF

**Theorem**

*A relation schema* $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ *is in Boyce-Codd normalform, if all* $\mathcal{R}_i$ *are in Boyce-Codd normalform.*

# Decomposition in BCNF

## Theorem

*A relation schema $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ is in Boyce-Codd normalform, if all $\mathcal{R}_i$ are in Boyce-Codd normalform.*

- given a relation schema $\mathcal{R}$ with FDs $F$
- find: decomposition in sub schemata $\mathcal{R}_1 \ldots \mathcal{R}_n$, for which it holds that:
  - lossless-join decomposition in $\mathcal{R}_1 \ldots \mathcal{R}_n$ ,
  - dependency-preserving decomposition $\mathcal{R}_1 \ldots \mathcal{R}_n$,
  - all $\mathcal{R}_1 \ldots \mathcal{R}_n$ are in Boyce-Codd normalform.

# Decomposition in BCNF

## Theorem

*A relation schema $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ is in Boyce-Codd normalform, if all $\mathcal{R}_i$ are in Boyce-Codd normalform.*

- given a relation schema $\mathcal{R}$ with FDs $F$
- find: decomposition in sub schemata $\mathcal{R}_1 \ldots \mathcal{R}_n$, for which it holds that:
  - lossless-join decomposition in $\mathcal{R}_1 \ldots \mathcal{R}_n$ ,
  - dependency-preserving decomposition $\mathcal{R}_1 \ldots \mathcal{R}_n$,
  - all $\mathcal{R}_1 \ldots \mathcal{R}_n$ are in Boyce-Codd normalform.
- only the BCNF guarantees to be anomaly-free

# Decomposition in BCNF

## Theorem

*A relation schema $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ is in Boyce-Codd normalform, if all $\mathcal{R}_i$ are in Boyce-Codd normalform.*

- given a relation schema $\mathcal{R}$ with FDs $F$
- find: decomposition in sub schemata $\mathcal{R}_1 \ldots \mathcal{R}_n$, for which it holds that:
  - lossless-join decomposition in $\mathcal{R}_1 \ldots \mathcal{R}_n$ ,
  - dependency-preserving decomposition $\mathcal{R}_1 \ldots \mathcal{R}_n$,
  - all $\mathcal{R}_1 \ldots \mathcal{R}_n$ are in Boyce-Codd normalform.
- only the BCNF guarantees to be anomaly-free
- problem: a lossless-join decomposition in BCNF is always possible, but dependency-preserving decomposition cannot always be reached

# Decomposition in BCNF

## Theorem

*A relation schema $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ is in Boyce-Codd normalform, if all $\mathcal{R}_i$ are in Boyce-Codd normalform.*

- given a relation schema $\mathcal{R}$ with FDs $F$
- find: decomposition in sub schemata $\mathcal{R}_1 \ldots \mathcal{R}_n$, for which it holds that:
  - lossless-join decomposition in $\mathcal{R}_1 \ldots \mathcal{R}_n$ ,
  - dependency-preserving decomposition $\mathcal{R}_1 \ldots \mathcal{R}_n$,
  - all $\mathcal{R}_1 \ldots \mathcal{R}_n$ are in Boyce-Codd normalform.
- only the BCNF guarantees to be anomaly-free
- problem: a lossless-join decomposition in BCNF is always possible, but dependency-preserving decomposition cannot always be reached
- decomposition algorithm (without a dependency-preserving decomposition)

# decomposition algorithm

- start with $Z = \{(\mathcal{R}, F)\}$

# decomposition algorithm

- start with $Z = \{(\mathcal{R}, F)\}$
- as long as there is a relation schema $(\mathcal{R}_i, F_i)$ in $Z$, which violates the BCNF i.e. there is a FD $\alpha \rightarrow \beta$ in $\mathcal{R}_i$ with $\beta \not\subseteq \alpha$ and $\neg(\alpha \rightarrow \mathcal{R}_i)$:

# decomposition algorithm

- start with $Z = \{(\mathcal{R}, F)\}$
- as long as there is a relation schema $(\mathcal{R}_i, F_i)$ in $Z$, which violates the BCNF i.e. there is a FD $\alpha \rightarrow \beta$ in $\mathcal{R}_i$ with $\beta \not\subseteq \alpha$ and $\neg(\alpha \rightarrow \mathcal{R}_i)$: pick one such FD $\alpha \rightarrow \beta$ and decompose as follows:
  - $\mathcal{R}_{i_1} := (\alpha \cup \beta)$, $F_{i_1} := F_i^+[\mathcal{R}_{i_1}]$
  - $\mathcal{R}_{i_2} := \mathcal{R}_i - (\beta - \alpha)$, $F_{i_2} := F_i^+[\mathcal{R}_{i_2}]$

# decomposition algorithm

- start with $Z = \{(\mathcal{R}, F)\}$
- as long as there is a relation schema $(\mathcal{R}_i, F_i)$ in $Z$, which violates the BCNF i.e. there is a FD $\alpha \to \beta$ in $\mathcal{R}_i$ with $\beta \not\subseteq \alpha$ and $\neg(\alpha \to \mathcal{R}_i)$: pick one such FD $\alpha \to \beta$ and decompose as follows:
  - $\mathcal{R}_{i_1} := (\alpha \cup \beta)$, $F_{i_1} := F_i^+[\mathcal{R}_{i_1}]$
  - $\mathcal{R}_{i_2} := \mathcal{R}_i - (\beta - \alpha)$, $F_{i_2} := F_i^+[\mathcal{R}_{i_2}]$

  remove $(\mathcal{R}_i, F_i)$ from $Z$ and insert $(\mathcal{R}_{i_1}, F_{i_1})$ and $(\mathcal{R}_{i_2}, F_{i_2})$:

$$Z := (Z - (\{\mathcal{R}_i, F_i)\}) \cup \{(\mathcal{R}_{i_1}, F_{i_1})\} \cup \{(\mathcal{R}_{i_2}, F_{i_2})\}$$

# decomposition algorithm

- start with $Z = \{(\mathcal{R}, F)\}$
- as long as there is a relation schema $(\mathcal{R}_i, F_i)$ in $Z$, which violates the BCNF i.e. there is a FD $\alpha \to \beta$ in $\mathcal{R}_i$ with $\beta \not\subseteq \alpha$ and $\neg(\alpha \to \mathcal{R}_i)$:
  pick one such FD $\alpha \to \beta$ and decompose as follows:
  - $\mathcal{R}_{i_1} := (\alpha \cup \beta)$, $F_{i_1} := F_i^+[\mathcal{R}_{i_1}]$
  - $\mathcal{R}_{i_2} := \mathcal{R}_i - (\beta - \alpha)$, $F_{i_2} := F_i^+[\mathcal{R}_{i_2}]$

  remove $(\mathcal{R}_i, F_i)$ from $Z$ and insert $(\mathcal{R}_{i_1}, F_{i_1})$ and $(\mathcal{R}_{i_2}, F_{i_2})$:

$$Z := (Z - (\{\mathcal{R}_i, F_i)\}) \cup \{(\mathcal{R}_{i_1}, F_{i_1})\} \cup \{(\mathcal{R}_{i_2}, F_{i_2})\}$$

$$\begin{array}{ccc} & \mathcal{R}_i & \\ & \alpha \to \beta \text{ and other FDs } F_i & \\ \swarrow & & \searrow \\ \mathcal{R}_{i_1} = (\alpha \cup \beta) & & \mathcal{R}_{i_2} = (\mathcal{R}_i - (\beta - \alpha)) \\ F_{i_1} = F_i^+[\mathcal{R}_{i_1}] & & F_{i_2} = F_i^+[\mathcal{R}_{i_2}] \end{array}$$

# decomposition algorithm

## Example (decomposition algorithm 1)

$$\mathcal{R} = \text{cities}(\text{town}, \text{land}, \text{boss}, \text{population})$$

$$F = \{\{\text{town}, \text{land}\} \to \{\text{population}\}, \{\text{land}\} \to \{\text{boss}\}, \{\text{boss}\} \to \{\text{land}\}\}$$

# decomposition algorithm

## Example (decomposition algorithm 1)

$$\begin{aligned}\mathcal{R} &= \text{cities(town, land, boss, population)}\\ F &= \{\{\text{town, land}\} \rightarrow \{\text{population}\},\\ &\quad \{\text{land}\} \rightarrow \{\text{boss}\}, \{\text{boss}\} \rightarrow \{\text{land}\}\}\end{aligned}$$

key: $\{$town, land$\}$, $\{$town, boss$\}$

# decomposition algorithm

## Example (decomposition algorithm 1)

$$\mathcal{R} = \text{cities(town, land, boss, population)}$$
$$F = \{\{\text{town, land}\} \to \{\text{population}\}, \{\text{land}\} \to \{\text{boss}\}, \{\text{boss}\} \to \{\text{land}\}\}$$

key: {town, land}, {town, boss}

- $Z := \mathcal{R}$
- {land} $\to$ {boss} violates the BCNF

# decomposition algorithm

## Example (decomposition algorithm 1)

$\mathcal{R}$ = cities(town, land, boss, population)

F = {{town, land} → {population},
{land} → {boss}, {boss} → {land}}

key: {town, land}, {town, boss}

- $Z := \mathcal{R}$
- {land} → {boss} violates the BCNF
  - $\mathcal{R}_1 :=$ (land, boss) and
    $F_1 := \{\{land\} \rightarrow \{boss\}, \{boss\} \rightarrow \{land\}, \ldots\}$
  - $\mathcal{R}_2 :=$ (town, land, population) and
    $F_2 := \{\{town, land\} \rightarrow \{population\}, \ldots\}$

# decomposition algorithm

## Example (decomposition algorithm 1)

$$\mathcal{R} = \text{cities(town, land, boss, population)}$$
$$F = \{\{\text{town, land}\} \to \{\text{population}\}, \{\text{land}\} \to \{\text{boss}\}, \{\text{boss}\} \to \{\text{land}\}\}$$

key: {town, land}, {town, boss}

- $Z := \mathcal{R}$
- {land} → {boss} violates the BCNF
  - $\mathcal{R}_1 :=$ (land, boss) and $F_1 := \{\{land\} \to \{boss\}, \{boss\} \to \{land\}, \dots\}$
  - $\mathcal{R}_2 :=$ (town, land, population) and $F_2 := \{\{town, land\} \to \{population\}, \dots\}$
  - $Z := \{(\mathcal{R}_1, F_1), (\mathcal{R}_2, F_2)\}$

# decomposition algorithm

## Example (decomposition algorithm 1)

$\mathcal{R}$ = cities(town, land, boss, population)

$F = \{\{\text{town, land}\} \rightarrow \{\text{population}\},$
$\{\text{land}\} \rightarrow \{\text{boss}\}, \{\text{boss}\} \rightarrow \{\text{land}\}\}$

key: {town, land}, {town, boss}

- $Z := \mathcal{R}$
- $\{\text{land}\} \rightarrow \{\text{boss}\}$ violates the BCNF
  - $\mathcal{R}_1 :=$ (land, boss) and
    $F_1 := \{\{land\} \rightarrow \{boss\}, \{boss\} \rightarrow \{land\}, \ldots\}$
  - $\mathcal{R}_2 :=$ (town, land, population) and
    $F_2 := \{\{town, land\} \rightarrow \{population\}, \ldots\}$
  - $Z := \{(\mathcal{R}_1, F_1), (\mathcal{R}_2, F_2)\}$
- lossless-join decomposition and dependency-preserving decomposition

FAKULTÄT FÜR INFORMATIK
Faculty of Informatics

# decomposition algorithm

## Example (decomposition algorithm 2)

$\mathcal{R}$ = zipcodeList(street, town, state, zipcode)

F = {{zipcode}→{town, state}, {street, town, state}→{zipcode}}

# decomposition algorithm

## Example (decomposition algorithm 2)

$\mathcal{R}$ = zipcodeList(street, town, state, zipcode)

F = {{zipcode}→{town, state}, {street, town, state}→{zipcode}}

key:

# decomposition algorithm

## Example (decomposition algorithm 2)

$\mathcal{R}$ = zipcodeList(street, town, state, zipcode)

F = {{zipcode}→{town, state}, {street, town, state}→{zipcode}}

key: {street, town, state}, {zipcode, street},

- $Z := (\mathcal{R}, F)$

# decomposition algorithm

## Example (decomposition algorithm 2)

$\mathcal{R}$ = zipcodeList(street, town, state, zipcode)

F = {{zipcode}→{town, state}, {street, town, state}→{zipcode}}

key: {street, town, state}, {zipcode, street},

- $Z := (\mathcal{R}, F)$
- {{zipcode} → {town, state}} violates the BCNF

# decomposition algorithm

## Example (decomposition algorithm 2)

$$\mathcal{R} = \text{zipcodeList(street, town, state, zipcode)}$$
$$\text{F} = \{\{\text{zipcode}\}\rightarrow\{\text{town, state}\}, \{\text{street, town, state}\}\rightarrow\{\text{zipcode}\}\}$$

key: {street, town, state}, {zipcode, street},

- $Z := (\mathcal{R}, F)$
- {{zipcode} → {town, state}} violates the BCNF
  - $\mathcal{R}_1 :=$ (zipcode, town, state) and $F_1 := \{\{zipcode\} \rightarrow \{town, state\}, \dots\}$
  - $\mathcal{R}_2 :=$ (zipcode, street) and $F_2$ contains only trivial FDs

# decomposition algorithm

## Example (decomposition algorithm 2)

$\mathcal{R}$ = zipcodeList(street, town, state, zipcode)

F = {{zipcode}→{town, state}, {street, town, state}→{zipcode}}

key: {street, town, state}, {zipcode, street},

- $Z := (\mathcal{R}, F)$
- {{zipcode} → {town, state}} violates the BCNF
  - $\mathcal{R}_1$ := (zipcode, town, state) and $F_1 := \{\{zipcode\} \to \{town, state\}, \ldots\}$
  - $\mathcal{R}_2$ := (zipcode, street) and $F_2$ contains only trivial FDs
  - $Z := \{(\mathcal{R}_1, F_1), (\mathcal{R}_2, F_2)\}$

# decomposition algorithm

## Example (decomposition algorithm 2)

$\mathcal{R}$ = zipcodeList(street, town, state, zipcode)

F = {{zipcode}→{town, state}, {street, town, state}→{zipcode}}

key: {street, town, state}, {zipcode, street},

- $Z := (\mathcal{R}, F)$
- {{zipcode} → {town, state}} violates the BCNF
  - $\mathcal{R}_1 :=$ (zipcode, town, state) and $F_1 := \{\{zipcode\} \rightarrow \{town, state\}, \dots\}$
  - $\mathcal{R}_2 :=$ (zipcode, street) and $F_2$ contains only trivial FDs
  - $Z := \{(\mathcal{R}_1, F_1), (\mathcal{R}_2, F_2)\}$
- lossless-join decomposition but the dependency {{street, town, state} → {zipcode}} is lost

# decomposition algorithm

## Example (decomposition algorithm 3)

$$\mathcal{R} = ABCDE$$
$$F = \{A \to B, C \to B, BE \to D\}$$
key: $ACE$

# decomposition algorithm

## Example (decomposition algorithm 3)

$$\mathcal{R} = ABCDE$$
$$F = \{A \to B, C \to B, BE \to D\}$$
key: $ACE$

$$\mathcal{R}_1 = AB$$
$$F_1 = \{A \to B, \dots\}$$
key: $A$

# decomposition algorithm

## Example (decomposition algorithm 3)

$$\mathcal{R} = ABCDE$$
$$F = \{A \to B, C \to B, BE \to D\}$$
key: $ACE$

$$\mathcal{R}_1 = AB$$
$$F_1 = \{A \to B, \dots\}$$
key: $A$

$$\mathcal{R}_2 = ACED$$
$$F_2 = \{AE \to D, CE \to D, ACE \to D, \dots\}$$
key: $ACE$

# decomposition algorithm

## Example (decomposition algorithm 3)

$$\mathcal{R} = ABCDE$$
$$F = \{A \to B, C \to B, BE \to D\}$$
key: $ACE$

$$\mathcal{R}_1 = AB$$
$$F_1 = \{A \to B, \dots\}$$
key: $A$

$$\mathcal{R}_2 = ACED$$
$$F_2 = \{AE \to D, CE \to D, ACE \to D, \dots\}$$
key: $ACE$

$$\mathcal{R}_{2,1} = AED$$
$$F_{2,1} = \{AE \to D, \dots\}$$
key: $AE$

# decomposition algorithm

## Example (decomposition algorithm 3)

$$\mathcal{R} = ABCDE$$
$$F = \{A \rightarrow B, C \rightarrow B, BE \rightarrow D\}$$
key: $ACE$

$$\mathcal{R}_1 = AB$$
$$F_1 = \{A \rightarrow B, \ldots\}$$
key: $A$

$$\mathcal{R}_2 = ACED$$
$$F_2 = \{AE \rightarrow D, CE \rightarrow D, ACE \rightarrow D, \ldots\}$$
key: $ACE$

$$\mathcal{R}_{2,1} = AED$$
$$F_{2,1} = \{AE \rightarrow D, \ldots\}$$
key: $AE$

$$\mathcal{R}_{2,2} = ACE$$
$$F_{2,2} = \{\ldots\}$$
key: $ACE$

# decomposition algorithm

## Example (decomposition algorithm 3)

$$\mathcal{R} = ABCDE$$
$$F = \{A \to B, C \to B, BE \to D\}$$
$$\text{key: } ACE$$

$$\mathcal{R}_1 = AB \qquad\qquad \mathcal{R}_2 = ACED$$
$$F_1 = \{A \to B, \ldots\} \qquad F_2 = \{AE \to D, CE \to D, ACE \to D, \ldots\}$$
$$\text{key: } A \qquad\qquad \text{key: } ACE$$

$$\mathcal{R}_{2,1} = AED \qquad\qquad \mathcal{R}_{2,2} = ACE$$
$$F_{2,1} = \{AE \to D, \ldots\} \qquad F_{2,2} = \{\ldots\}$$
$$\text{key: } AE \qquad\qquad \text{key: } ACE$$

lossless-join decomposition, but not a dependency-preserving decomposition:
$\{C \to B, BE \to D\} \in F$, z.B. $\{CE \to D, \ldots\} \in F^+$

# Learning Objectives

- What kinds of anomalies are there?
- What is a lossless-join decomposition and what is a dependency-preserving decomposition?
- Which normalforms are there?
    - When are they satisfied and how are the computed?