Relational Design Theory

- Assess the quality of a schema
  - redundancy
  - integrity constraints
  - **Quality seal: normal forms (1-4, BCNF)**

- Improve the quality of a schema
  - **synthesis algorithm**
  - **decomposition algorithm**

- Construct a (high-quality) schema
  - start with universal relation
  - apply synthesis or decomposition algorithms

What is wrong with redundancy?

- Waste of storage space
  - importance is diminishing as storage gets cheaper
  - (disk density will even increase in the future)

- Additional work to keep multiple copies of data consistent
  - multiple updates in order to accomodate one event

- Additional code to keep multiple copies of data consistent
  - Somebody needs to implement the logic

Bad Schemas

ProfLecture						
PersNr	Name	Level	Room	Nr	Title	CP
2125	Sokrates	FP	226	5041	Ethik	4
2125	Sokrates	FP	226	5049	Mäeutik	2
2125	Sokrates	FP	226	4052	Logik	4
...	...	...	...	...	...	...
2132	Popper	AP	52	5259	Der Wiener Kreis	2
2137	Kant	FP	7	4630	Die 3 Kritiken	4

- Update-Anomaly
  - What happens when Sokrates moves to a different room?
- Insert-Anomaly
  - What happens if Roscoe is elected as a new professor?
- Delete-Anomaly
  - What happens if Popper does not teach this semester?

Multi-version Databases

- Storage becomes cheaper -> never throw anything away
  - It is more expensive to think about what to keep than simply to keep everything.

- Consequence 1: No delete
  - Instead, set a status flag to „deleted“
  - No delete anomalies (only wasted storage)
- Consequence 2: No update in place
  - Instead, create a new version of the tuple
  - No update anomalies (only wasted storage)
- Insert anomalies still exist, but not a big problem
  - Result in multiple NULL values, but no inconsistencies

- **NoSQL Movement: Denormalized data (XML is great!)**

Functional Dependencies

- Schema: $\mathcal{R} = \{A{:}D_A,\ B{:}D_B,\ C{:}D_C,\ D{:}D_D\}$
- Instance: R

- Let $\alpha \subseteq \mathcal{R},\ \ \beta \subseteq \mathcal{R}$
- $\alpha \rightarrow \beta$ iff $\forall r, s \in R$: $r.\alpha = s.\alpha \Rightarrow r.\beta = s.\beta$
- (There is a function f: X $D_\alpha \rightarrow$ X D_β)

R			
A	*B*	*C*	*D*
a4	b2	c4	d3
a1	b1	c1	d1
a1	b1	c1	d2
a2	b2	c3	d2
a3	b2	c4	d3

$\{A\} \rightarrow \{B\}$

$\{C, D\} \rightarrow \{B\}$

Not: $\{B\} \rightarrow \{C\}$

Convention:

$CD \rightarrow B$

Example

Family Tree				
Child	Father	Mother	Grandma	Grandpa
Sofie	Alfons	Sabine	Lothar	Linde
Sofie	Alfons	Sabine	Hubert	Lisa
Niklas	Alfons	Sabine	Lothar	Linde
Niklas	Alfons	Sabine	Hubert	Lisa
…	…	…	Lothar	Martha
...	...	...	...	...

Example

Family Tree				
Child	Father	Mother	Grandma	Grandpa
Sofie	Alfons	Sabine	Lothar	Linde
Sofie	Alfons	Sabine	Hubert	Lisa
Niklas	Alfons	Sabine	Lothar	Linde
Niklas	Alfons	Sabine	Hubert	Lisa
...	...	...	Lothar	Martha
...	...	...	...	...

- Child→ Father,Mother
- Child, Grandpa → Grandma
- Child, Grandma → Grandpa

Analogy to functions

- f1 : Child → Father
  - E.g., f1(Niklas) = Alfons
- f2: Child → Mother
  - E.g., f2(Niklas) = Sabine
- f3: Child x Grandpa → Grandma
- FD: Child → Father, Mother
  - represents two functions (f1, f2)
  - Komma on right side indicates multiple functions
- FD: Child, Grandpa → Grandma
  - Komma on the left side indicates Cartesian product

Keys

- $\alpha \subseteq \mathcal{R}$ is a superkey iff
  - $\alpha \rightarrow \mathcal{R}$
- $\alpha \rightarrow \beta$ is minimal iff
  - $\forall A \in \alpha: \neg((\alpha - \{A\}) \rightarrow \beta)$
- Notation for minimal functional dependencies: $\alpha \rightarrow^{\cdot} \beta$
- $\alpha \subseteq \mathcal{R}$ is a key (or candidate key) iff
  - $\alpha \rightarrow^{\cdot} \mathcal{R}$

Determining Keys

Town			
Name	Canton	AreaCode	Population
Buchs	AG	081	6500
Buchs	SG	071	8000
Zurich	ZH	044	300000
Lausanne	VD	021	60000
...	...	...	...

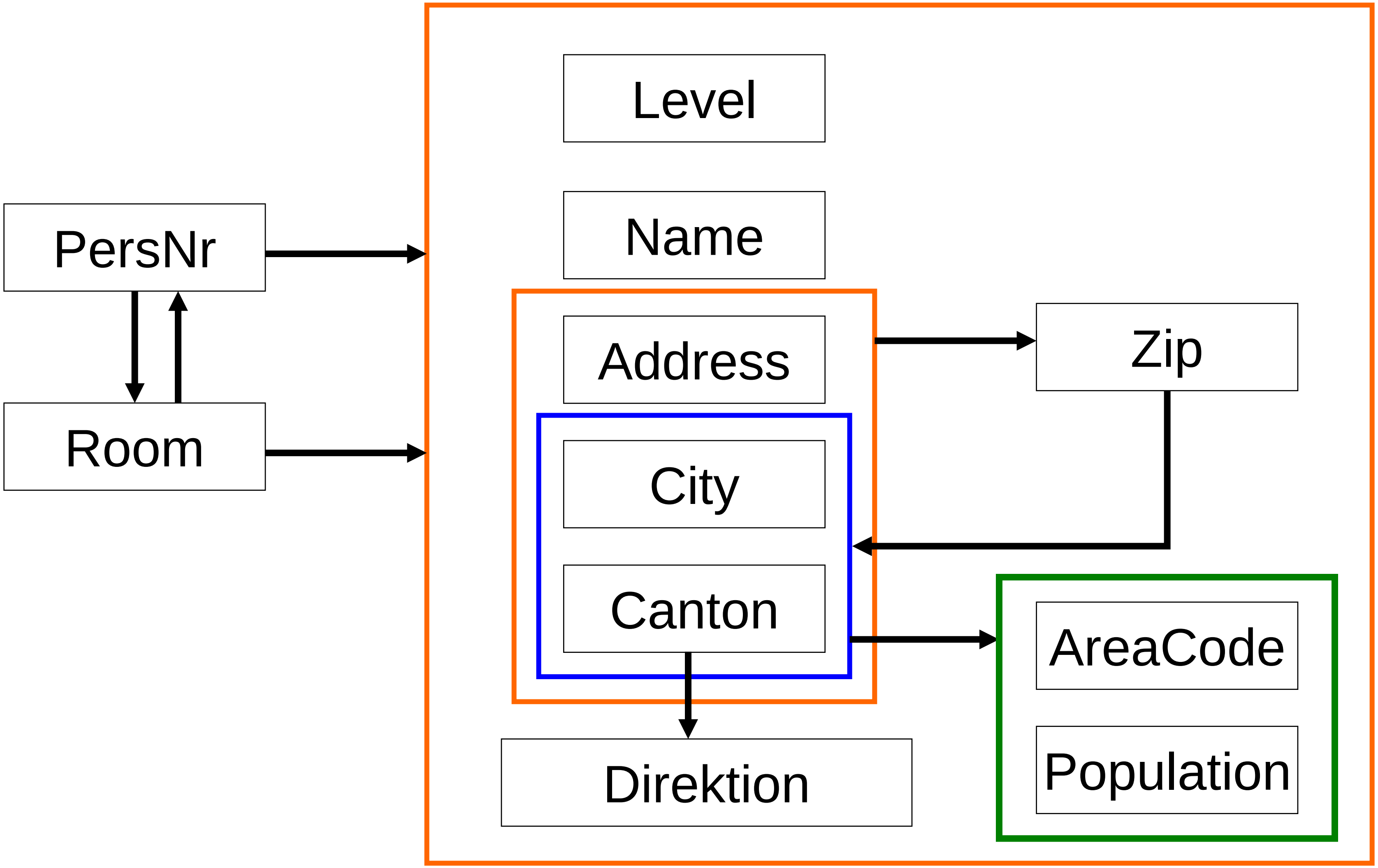
- Keys of Town
  - {Name, Canton}
  - {Name, AreaCode}

- N.B. Two small towns may have the same area code.

Determining Functional Dependencies

- Professor: {[PersNr, Name, Level, Room, City, Address, Zip, AreaCode, Canton, Population, Direktion]}
  - {PersNr} → {PersNr, Name, Level, Room, City, Address, Zip, AreaCode, Canton, Population, Direktion}
  - {City, Canton} → {Population, AreaCode}
  - {Zip} → {Canton, City, Population}
  - {Canton, City, Address} → {Zip}
  - {Canton} → {Direktion}
  - {Room} → {PersNr}
- Additional functional dependencies (inferred):
  - {Room} → {PersNr, Name, Level, Room, City, Address, Zip, AreaCode, Canton, Population, Direktion}
  - {Zip} → {Direktion}

Visualization of Funct. Dependencies



Armstrong Axioms: Inference of FDs

- Reflexivity
  - $(\beta \subseteq \alpha) \Rightarrow \alpha \rightarrow \beta$
  - Special case: $\alpha \rightarrow \alpha$

- Augmentation
  - $\alpha \rightarrow \beta \Rightarrow \alpha\gamma \rightarrow \beta\gamma$.
  - (Notation $\alpha\gamma := \alpha \cup \gamma$)

- Transitivity
  - $\alpha \rightarrow \beta \wedge \beta \rightarrow \gamma \Rightarrow \alpha \rightarrow \gamma$.

- These three axioms are complete. All possible other rules can be implied from these axioms.

Other rules

- Union of FDs:
  - $\alpha \to \beta \wedge \alpha \to \gamma \Rightarrow \alpha \to \beta\gamma$
- Decomposition:
  - $\alpha \to \beta\gamma \Rightarrow \alpha \to \beta \wedge \alpha \to \gamma$
- Pseudo transitivity:
  - $\alpha \to \beta \wedge \gamma\beta \to \delta \Rightarrow \alpha\gamma \to \delta$

Correctness of Union rule

- Premise: $\alpha \to \beta$ and $\alpha \to \gamma$
- Claim: $\alpha \to \beta\gamma$
- Proof:
  1. $\alpha \to \beta$ (Premise)
  2. $\alpha\gamma \to \beta\gamma$ (Augmentation)
  3. $\alpha \to \gamma$ (Premise)
  4. $\alpha \to \alpha\gamma$ (Augmentation)
  5. $\alpha \to \beta\gamma$ (Transitivity of (4) and (2)) qed

Closure of Attributes

- Input:
  - F: a set of FDs
  - α: a set of attributes
- Output: $\alpha+$ such that $\alpha \rightarrow \alpha+$

```
Closure(F, α)
   result := α                                          // Reflexivity
   while   (result has changed) do
        foreach FD: β → γ in F do                       // Transitivity
                if  β ⊆ result then result := result ∪ γ
   output(result)
```

- Exercise: Proof that Closure is deterministic and terminates.

Example: Closure of ZIP (Slide 8)

Minimal Basis

Fc is a minimal basis of F iff:

1. $Fc \equiv F$
   - The closure of all attribute set is the same in Fc and F

2. All functional dependencies in Fc are minimal:
   - $\forall A \in \alpha: (Fc - (\alpha \to \beta) \cup ((\alpha - \{A\}) \to \beta)) \not\equiv Fc$
   - $\forall B \in \beta: (Fc - (\alpha \to \beta) \cup (\alpha \to (\beta - \{B\}))) \not\equiv Fc$

3. In Fc, there are no two functional dependencies with the same left side.
   - Can be achieved by applying the Union rule.

Computing the Minimum Basis

1. Reduction of left sides of FDs. Let $\alpha \rightarrow \beta \in F$, $A \in \alpha$:

   if $\beta \subseteq$ Closure(F, α – A)

   then replace $\alpha \rightarrow \beta$ **with** (α - A) $\rightarrow \beta$ in F

2. Reduction of right sides of FDs. Let $\alpha \rightarrow \beta \in F$, $B \in \beta$:

   if B $\in$ Closure(F – ($\alpha \rightarrow \beta$) $\cup$ ($\alpha \rightarrow (\beta - B)$), α)

   then replace $\alpha \rightarrow \beta$ **with** $\alpha \rightarrow (\beta – B)$ in F

1. Remove FDs: $\alpha \rightarrow \varnothing$ (clean-up of Step 2)

2. Apply Union rule to FDs with the same left side.

Determining Functional Dependencies

- Professor: {[PersNr, Name, Level, Room, City, Address, Zip, AreaCode, Canton, Population, Direktion]}
  - {PersNr} → {PersNr, Name, Level, Room, City, Address, Zip, AreaCode, Canton, Population, Direktion}
  - {City, Canton} → {Population, AreaCode}
  - {Zip} → {Canton, City, Population}
  - {Canton, City, Address} → {Zip}
  - {Canton} → {Direktion}
  - {Room} → {PersNr}
- Additional functional dependencies (inferred):
  - {Room} → {PersNr, Name, Level, Room, City, Address, Zip, AreaCode, Canton, Population, Direktion}
  - {Zip} → {Direktion}

Correctness of the Algorithm (Left Reduction)

- Premise: $\beta \subseteq$ Closure(F, α - A)
- Claim: Closure(F-$\{\alpha \rightarrow \beta\} \cup \{ (\alpha - A) \rightarrow \beta \}$, α - A) $\subseteq$ Closure(F, α - A)
- Proof:

let $\gamma \in$ Closure(F-$\{\alpha \rightarrow \beta\} \cup \{ (\alpha - A) \rightarrow \beta \}$, α - A)
$\gamma \in$ Closure(F, α - A $\cup$ β) (Apply FD $(\alpha - A) \rightarrow \beta$)
$\gamma \in$ Closure(F, α - A) (Premise) qed

Bad Schemas

ProfLecture						
PersNr	Name	Level	Room	Nr	Title	CP
2125	Sokrates	FP	226	5041	Ethik	4
2125	Sokrates	FP	226	5049	Mäeutik	2
2125	Sokrates	FP	226	4052	Logik	4
...	...	...	...	...	...	...
2132	Popper	AP	52	5259	Der Wiener Kreis	2
2137	Kant	FP	7	4630	Die 3 Kritiken	4

- Update-Anomaly
  - What happens when Sokrates moves to a different room?
- Insert-Anomaly
  - What happens if Roscoe is elected as a new professor?
- Delete-Anomaly
  - What happens if Popper does not teach this semester?

Decomposition of Relations

- Bad relations combine several concepts
  - decompose them so that each concept in one relation
  - $\mathcal{R} \rightarrow \mathcal{R}_1, \ldots, \mathcal{R}_n$

1. Lossless Decomposition

$$\mathcal{R} = \mathcal{R}_1 \text{ A } \mathcal{R}_2 \text{ A } \ldots \text{ A } \mathcal{R}_n$$

2. Preservation of Dependencies

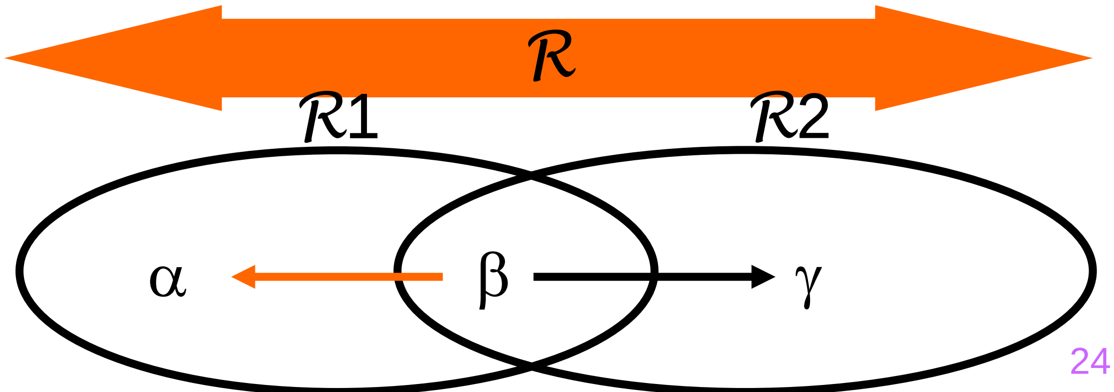
   - $FD(\mathcal{R})+ = (FD(\mathcal{R}_1) \cup \ldots \cup FD(\mathcal{R}_n))+$

When is a decomposition lossless?

- Let $\mathcal{R} = \mathcal{R}1 \cup \mathcal{R}2$
  - R1 := $\Pi_{\mathcal{R}1}$ (R)
  - R2 := $\Pi_{\mathcal{R}2}$ (R)

- Lemma: The decomposition is lossless if
  - $(\mathcal{R}1 \cap \mathcal{R}2) \rightarrow \mathcal{R}1$ or
  - $(\mathcal{R}1 \cap \mathcal{R}2) \rightarrow \mathcal{R}2$

- Exercise: Proof of this Lemma.



Example

Drinker		
Pub	*Guest*	*Beer*
Kowalski	Kemper	Pils
Kowalski	Eickler	Hefeweizen
Innsteg	Kemper	Hefeweizen

Lossy Decomposition

Drinker		
Pub	*Guest*	*Beer*
Kowalski	Kemper	Pils
Kowalski	Eickler	Hefeweizen
Innsteg	Kemper	Hefeweizen

$\Pi_{\text{Pub, Guest}}$

$\Pi_{\text{Guest, Beer}}$

Visitor	
Pub	*Guest*
Kowalski	Kemper
Kowalski	Eickler
Innsteg	Kemper

Drinks	
Guest	*Beer*
Kemper	Pils
Eickler	Hefeweizen
Kemper	Hefeweizen

Drinker		
Kneipe	*Gast*	*Bier*
Kowalski	Kemper	Pils
Kowalski	Eickler	Hefeweizen
Innsteg	Kemper	Hefeweizen

$\Pi_{....}$

Visitor	
Pub	*Guest*
Kowalski	Kemper
Kowalski	Eickler
Innsteg	Kemper

Drinks	
Guest	*Beer*
Kemper	Pils
Eickler	Hefeweizen
Kemper	Hefeweizen

A

$\neq$

Visitor A Drinks		
Pub	*Guest*	*Beer*
Kowalski	Kemper	Pils
Kowalski	Kemper	Hefeweizen
Kowalski	Eickler	Hefeweizen
Innsteg	Kemper	Pils
Innsteg	Kemper	Hefeweizen

Comments on the Example

- Drinker has one (non-trivial) functional dependency
  - $\{Pub, Guest\} \rightarrow \{Beer\}$
- But none of the criteria of the Lemma hold
  - $\{Guest\} \nrightarrow \{Beer\}$
  - $\{Guest\} \nrightarrow \{Pub\}$
- The problem is that Kemper likes different beer in different pubs.

Lossless Decomposition

Parents		
Father	*Mother*	*Child*
Johann	Martha	Else
Johann	Maria	Theo
Heinz	Martha	Cleo

$\Pi_{\text{Father, Child}}$

$\Pi_{\text{Mother, Child}}$

Father	
Father	*Child*
Johann	Else
Johann	Theo
Heinz	Cleo

Mother	
Mother	*Child*
Martha	Else
Maria	Theo
Martha	Cleo

Comments on Example

- Parents: {[Father, Mother, Child]}
- Father: {[Father, Child]}
- Mother: {[Mother, Child]}

- Actually, both criteria of the lemma are met:
  - {Child}$\rightarrow${Mother}
  - {Child}$\rightarrow${Father}

- {Child} is a key of all three relations
  - Wrt loss of info, it never hurts to decompose with a key
  - However, it is never beneficial either. Why?

Preservation of Dependencies

- Let $\mathcal{R}$ be decomposed into $\mathcal{R}1, \ldots, \mathcal{R}n$
- $F_{\mathcal{R}} = (F_{\mathcal{R}1} \cup \ldots \cup F_{\mathcal{R}n})$

- ZipCodes: {[Street, City, Canton, Zip]}

- Functional dependencies in ZipCodes
  - {Zip} → {City, Canton}
  - {Street, City, Canton} → {Zip}

- What about this decomposition?
  - Streets: {[Zip, Street]}
  - Cities: {[Zip, City, Canton]}

- Is it lossless? Does it preserve functional depend.?

Decomposition of ZipCodes

ZipCodes			
City	***Canton***	***Street***	*Zip*
Buchs	AG	Goethestr.	5033
Buchs	AG	Schillerstr.	5034
Buchs	SG	Goethestr.	8107

$\Pi_{\text{Zip,Street}}$

$\Pi_{\text{City,Canton,Zip}}$

Streets	
Zip	***Street***
8107	Goethestr.
5033	Goethestr.
5034	Schillerstr.

Cities		
City	*Canton*	***Zip***
Buchs	AG	5033
Buchs	AG	5034
Buchs	SG	8107

{Street, City, Canton} → {Zip} not checkable in decomp. schema

It is possible to insert inconsistent tuples

Violation of City,Canton,Street→Zip

ZipCodes			
City	***Canton***	***Street***	*Zip*
Buchs	AG	Goethestr.	5033
Buchs	AG	Schillerstr.	5034
Buchs	SG	Goethestr.	8107

$\Pi_{\text{Zip,Street}}$

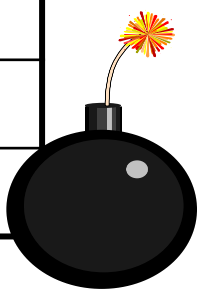
$\Pi_{\text{City,Canton,Zip}}$

Streets	
Zip	***Street***
8107	Goethestr.
5033	Goethestr.
5034	Schillerstr.
8108	Goethestr.

Cities		
City	*Canton*	***Zip***
Buchs	AG	5033
Buchs	AG	5034
Buchs	SG	8107
Buchs	SG	8108

Violation of City,Canton,Street→Zip

ZipCodes			
City	***Canton***	***Street***	*Zip*
Buchs	AG	Goethestr.	5033
Buchs	AG	Schillerstr.	5034
Buchs	SG	Goethestr.	8107
Buchs	SG	Goethestr.	8108



A

Streets	
Zip	***Street***
8107	Goethestr.
5033	Goethestr.
5034	Schillerstr.
8108	Goethestr.

Cities		
City	*Cantion*	***Zip***
Buchs	AG	5033
Buchs	AG	5034
Buchs	SG	8107
Buchs	SG	8108

First Normal Form

- Only atomic domains (as in SQL 92)

Parents		
Father	*Mother*	*Children*
Johann	Martha	{Else, Lucie}
Johann	Maria	{Theo, Josef}
Heinz	Martha	{Cleo}

vs.

Parents		
Father	*Mother*	*Child*
Johann	Martha	Else
Johann	Martha	Lucie
Johann	Maria	Theo
Johann	Maria	Josef
Heinz	Martha	Cleo

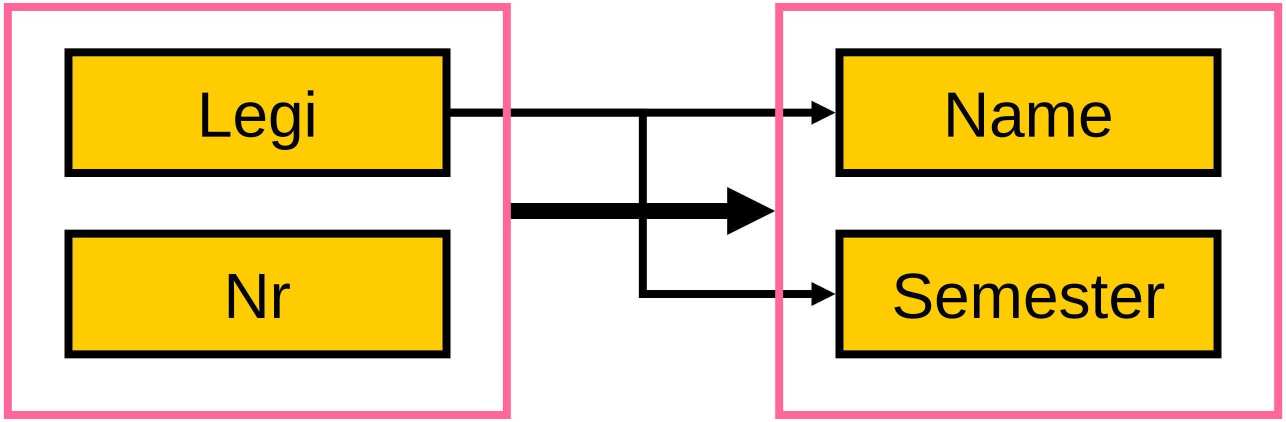
Second Normal Form

- $\mathcal{R}$ is in 2NF iff every non-key attribute is minimally dependent on every key.

StundentAttends			
Legi	**Nr**	Name	Semester
26120	5001	Fichte	10
27550	5001	Schopenhauer	6
27550	4052	Schopenhauer	6
28106	5041	Carnap	3
28106	5052	Carnap	3
28106	5216	Carnap	3
28106	5259	Carnap	3
...	...	...	...

- StudentAttends is not in 2NF!!!
  - {Legi} → {Name, Semester}

Second Normal Form



- Insert Anomaly: What about students who attend no lecture?
- Update Anomaly: Promotion of Carnab to the 4th semester.
- Delete Anomaly: Fichte drops his last course?

- Solution: Decompose into two relations
  - attends: {[Legi, Nr]}
  - Student: {[Legi, Name, Semester]}

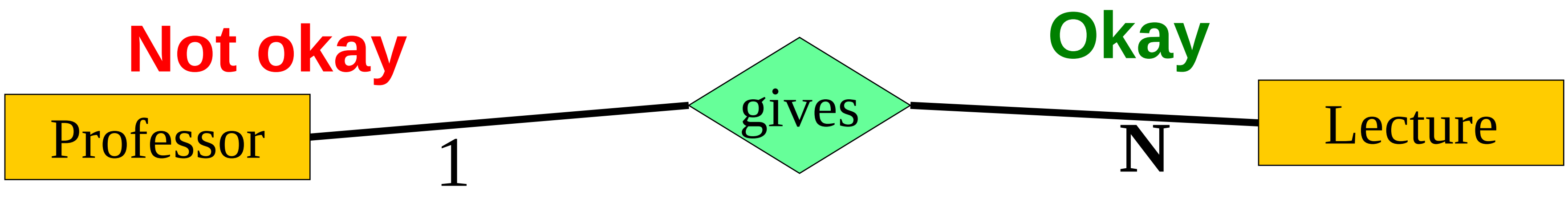
- Student, attends are in 2NF. The decompostion is lossless and preserves dependencies.

2NF and ER Modelling

- Violation of 2NF
  - mixing an entity with an N:M (or 1:N) relationship
  - E.g., mixing Student (entity) with attends (N:M)

- Solution
  - Separate: entity and relationship
  - i.e., implement entity and relationship in separate relations

- However, okay to mix entity and 1:1/N:1 relationship

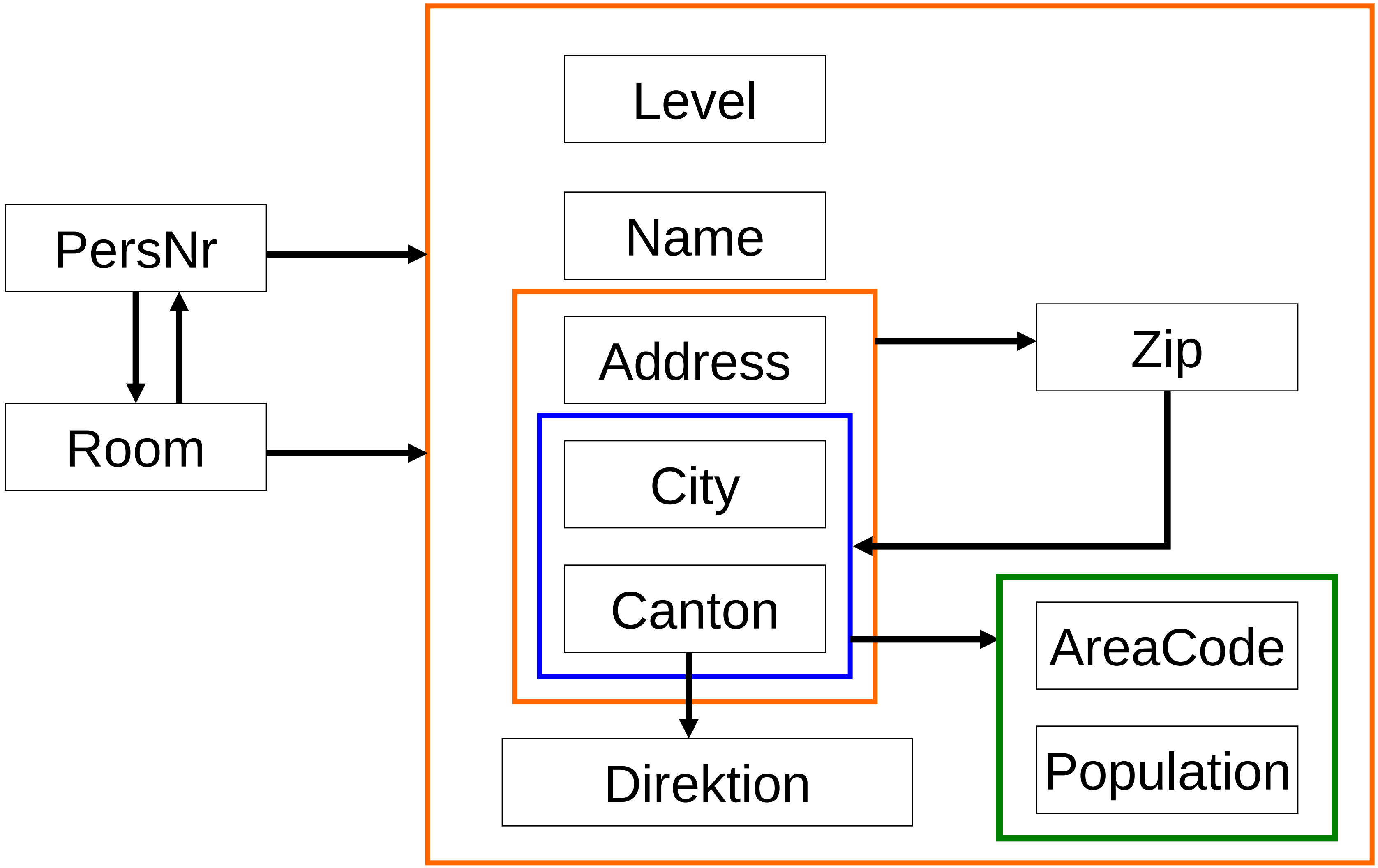


Third Normal Form

- $\mathcal{R}$ is in 3NF iff for all $\alpha \rightarrow B$ in $\mathcal{R}$ at least one condition holds:
  - B $\in \alpha$ (i.e., $\alpha \rightarrow B$ is trivial)
  - B is an attribute of at least one key
  - α is a superkey of $\mathcal{R}$

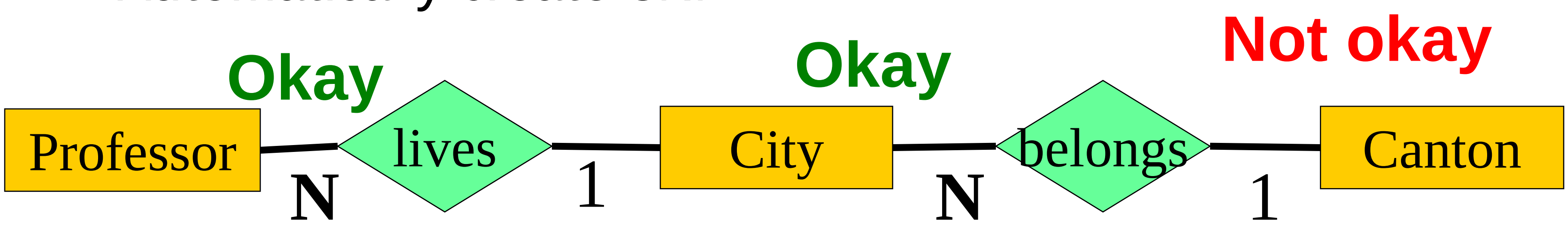
- If $\alpha \rightarrow B$ does not fulfill any of these conditions
  - α is a concept in its own right.

Example: 2NF but not 3NF



3NF and ER Modelling

- Violation of 3NF
  - mixing several entities (maybe connected by relationships)
  - e.g., Professor, City, Canton
- Solution
  - implement each entity in a separate relation
  - (implement N:M relationships in separate relation)
- ER Modelling and Rules of ER -> relational
  - Automatically create 3NF



3NF implies 2NF

- Premise: $\mathcal{R}$ is in 3NF
- Claim: $\mathcal{R}$ is in 2NF
- Proof:
  - assume $\mathcal{R}$ is not in 2NF
  - By definition of 2NF: exists $\alpha \to \mathrm{B}$ such that
    - (1) B is not part of any key
    - (2) $\alpha \subseteq \kappa$, κ is a key
  - $\alpha \to \mathrm{B}$ is evil
    - it is not trivial (otherwise B would be part of a key)
    - B is not part of any key (1)
    - α is not a superkey (2)
  - $\mathcal{R}$ is not in 3NF. qed

Synthesis Algorithm

- Input: Relation $\mathcal{R}$, FDs F
- Output: $\mathcal{R}$1, …, $\mathcal{R}$n such that
  - $\mathcal{R}$1, …, $\mathcal{R}$n is a lossless decomposition of $\mathcal{R}$.
  - $\mathcal{R}$1, …, $\mathcal{R}$n preserves dependencies.
  - All $\mathcal{R}$1, …, $\mathcal{R}$n are in 3NF.

Synthesis Algorithm

1. Compute the minimal basis Fc of F.

2. For all $\alpha \to \beta \in$ Fc create:
   - $\mathcal{R}\alpha := \alpha \cup \beta$

3. If exists $\kappa \subseteq \mathcal{R}$ such that κ is a key of $\mathcal{R}$ create:
   - $\mathcal{R}\kappa := \kappa$
   - (N.B.: $\mathcal{R}\kappa$ has no non-trivial functional dependencies.)

4. Eliminate $\mathcal{R}\alpha$ if exists $\mathcal{R}\alpha$` such that:
   - $\mathcal{R}\alpha \subseteq \mathcal{R}\alpha$`

Example: Synthesis Algorithm

- Professor: {[PersNr, Name, Level, Room, City, Street, Zip, AreaCode, Canton, Population, Direktion]}
  1. {PersNr} → {Name, Level, Room, Canton, Street, Canton}
  2. {Room} → {PersNr}
  3. {Street, Canton, City} → {Zip}
  4. {City, Canton} → {Population, AreaCode}
  5. {Canton} → {Direktion}
  6. {Zip} → {Canton, City}

- Professor: {[PersNr, Name, Level, Room, City, Street, Canton]}
- ZipCodes: {[Street, Canton, City, Zip]}
- Cities: {[City, Canton, Population, AreaCode]}
- Administration: {[Canton, Direktion]}

Example why Step 3 is needed

- StudentAttends(**Legi, Nr,** Name, Semester)

- Minimum Basis (Step 1)
  - {Legi} $\rightarrow$ {Name, Semester}

- Relation generated from minimum basis (Step 2)
  - Student(**Legi**, Name, Semester)

- Relation generated from Step 3
  - attends(**Legi, Nr**)

- The attends relation is needed!

Corner Case: Step 3

- R(A, B, C, D)
  - B -> C, D
  - D -> B
- Keys of R
  - A, B
  - A, D
- Decomposition into 3NF (Synthesis Algorithm)
  - R1(B, C, D)
  - R2(A, B)
- N.B. R3(A,D) is not needed!!!
  - Needs to be cleaned up in Step 4!

ZipCodes(Street, Canton, City, Zip)

- Is ZipCodes in 3NF?
  - Keys: {Street,Canton,City}, {Zip,Street}
  - All attributes are part of keys. There are no evil FDs!

- Does the decomposition preserve dependencies?
  - Yes!

- Is the decomposition lossless?
  - Professor $\cap$ ZipCodes = {Street,Canton,City}
  - {Street,Canton,City} $\rightarrow$ ZipCodes
  - Criterion of Lemma is fullfilled!

- Is ZipCode free of redundancy?

Exercises

- Proof for the following lemmas:
  - The synthesis algorithm preserves dependencies.
  - The synthesis algorithm creates lossless decompositions.
  - The synthesis algorithm creates relations in 3NF only.
  - The synthesis algorithm creates relations in 2NF only.

Synthesis Algo produces 3NF only

- Let $\mathcal{R}_i$ be a relation created by the Synthesis Algo
- Case 1: $\mathcal{R}_i$ was created in Step 3 of the algo
  - $\mathcal{R}_i$ contains a key of $\mathcal{R}$
  - there are no non-trivial FDs in $\mathcal{R}_i$
  - $\mathcal{R}_i$ is in 3NF
- Case 2: $\mathcal{R}_i$ was created in Step 2 by an FD: $\alpha \rightarrow \beta$
  - (1) $\mathcal{R}_i := \alpha \cup \beta$
  - (2) α is a key of $\mathcal{R}_i$
    - α is minimal because of left reduction of minimal basis
    - $\alpha \rightarrow \mathcal{R}_i$ by construction of $\mathcal{R}_i$
  - (3) $\alpha \rightarrow \beta$ is not evil because α is a superkey of $\mathcal{R}_i$
  - (4) Let $\gamma \rightarrow \delta$ be any <u>other</u> non-trivial FD ($\gamma \rightarrow \delta \neq \alpha \rightarrow \beta$)
    - $\delta \subseteq \alpha$ because of right reduction in minimal basis and because $\alpha \rightarrow \gamma$
    - δ contains only attributes of a key; $\gamma \rightarrow \delta$ is not evil qed

Boyce-Codd-Normal Form (BCNF)

- $\mathcal{R}$ is in BCNF iff for all $\alpha \rightarrow \mathrm{B}$ in $\mathcal{R}$ at least one condition holds:
  - B $\in \alpha$ (i.e., $\alpha \rightarrow \mathrm{B}$ is trivial)
  - α is a superkey of $\mathcal{R}$

- $\mathcal{R}$ in BCNF implies $\mathcal{R}$ in 3NF
  - Proof trivial from definition

- Result
  - any schema can be decomposed losslessly into BCNF
  - but, preservation of dependencies cannot be guaranteed
  - need to trade „correctness“ for „efficiency“
  - that is why 3NF is so important in practice

ZipCodes(Street, Canton, City, Zip)

- ZipCodes is not in BCNF
  - {Zip} → {Canton, City} // evil
  - {Street, Canton, City} → {Zip} // okay
- Redundancy in ZipCodes
  - (Rämistr., Zürich, Zürich, 8006)
  - (Universitätsstr., Zürich, Zürich, 8006)
  - (Schmid-Str., Zürich, Zürich, 8006)
  - stores several times that 8006 belongs to Zürich
- Exercise: How would you model ZipCodes in ER?
  - What would the relational schema look like?

Decomposition Algorithm (BCNF)

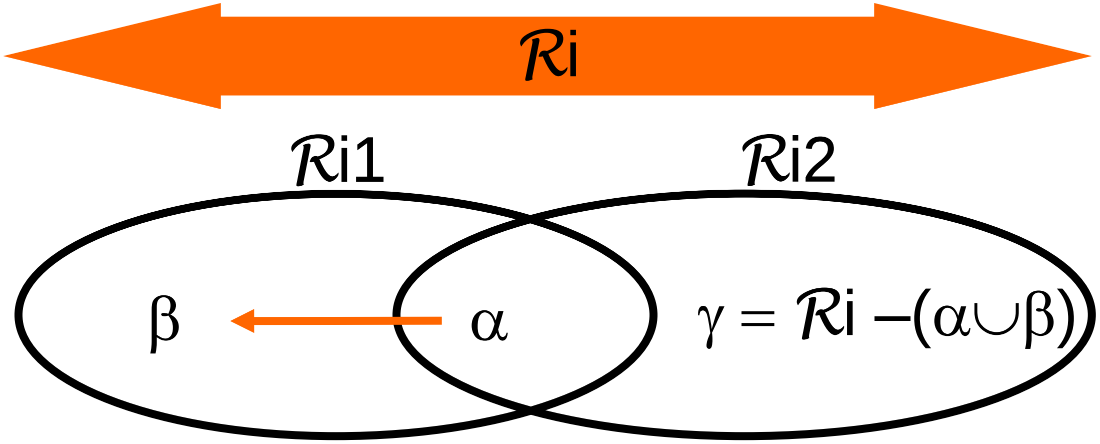
- Input: $\mathcal{R}$
- Output: $\mathcal{R}1, \ldots, \mathcal{R}n$ such that
  - $\mathcal{R}1, \ldots, \mathcal{R}n$ is a lossless decomposition of $\mathcal{R}$.
  - $\mathcal{R}1, \ldots, \mathcal{R}n$ are in BCNF.
  - (Preservation of dependencies is not guaranteed.)

Decomposition Algorithm

- Input: $\mathcal{R}$
- Output: $\mathcal{R}1, \ldots, \mathcal{R}n$

result = $\{\mathcal{R}\}$
while ($\exists$ $\mathcal{R}$i $\in$ Z: $\mathcal{R}$i is not in BCNF))
    let $\alpha \rightarrow \beta$ be evil in $\mathcal{R}$i
    $\mathcal{R}$i1 = $\alpha \cup \beta$
    $\mathcal{R}$i2 = $\mathcal{R}$i - β
    result = (result – $\{\mathcal{R}$i$\}$) $\cup$ $\{\mathcal{R}$i1$\}$ $\cup$ $\{\mathcal{R}$i2$\}$
output(result)

Visualization of Decomposition Algo



Decomposition of ZipCodes

- ZipCodes: {[Street, City, Canton, Zip]}
  - {Zip} → {City, Canton}   // evil
  - {Street, City, Canton} → {Zip}   // okay

- Applying the decomposition algorithm…
  - Street: {[Zip, Street]}
  - Cities: {[Zip, City, Canton]}

- Assessment
  - decomposition is lossless
  - decomposition does not preserve dependencies

Cities is not in BCNF

- Cities: {[City, Canton, Direktion, Population]}
- FDs of Cities:
  - {City, Canton} → {Population}
  - {Canton} → {Direktion}
  - {Direktion} → {Canton}
- Keys:
  - {City, Canton}

Köln	NRW	Rütgers	1mio
Bonn	NRW	Rütgers	200K
Aachen	NRW	Rütgers	200K

- In which highest NF is Cities?
  - N.B. decomposition algo can also be applied to non 3NF!

Decomposition of Cities

- Cities: {[City, Canton, Direktion, Population]}
  - {Canton} → {Direktion} // evil
  - {Direktion} → {Canton} // evil
  - {City, Canton} → {Population} // okay

- $\mathcal{R}$i1:
  - Administration: {[Canton, Direktion]}

- $\mathcal{R}$i2:
  - Cities: {[City, Canton, Population]}

- Is this decomposition lossless? Preserves depend.?

Fourth Normal Form

Skills		
PersNr	Language	Programming
3002	Greek	C
3002	Latin	Pascal
3002	Greek	Pascal
3002	Latin	C
3005	German	Ada

- Give FDs, keys. Which highest normal form?
- Does „Skills“ have redundancy?
  - What happens if 3002 learns a new language?
- How would you model Skills in ER? How translated?

NFNF

Skills		
PersNr	Language	Programming
3002	{Greek, Latin}	{C, Pascal}
3005	{German}	{Ada}

- This model would work: no redundancy, no anomolies
- But, unfortunately, not implementable in SQL 2
- It is implementable in SQL 3 and XML
  - That is why design theory needs to be adapted to XML

What is wrong with this?

Skills		
PersNr	Language	Programming
3002	Greek	C
3002	Latin	Pascal
3005	German	Ada

- Who knows Greek and Pascal?
- Anomolies?
  - What happens if 3002 learns a new language?

Multi-value Dependencies

R		
A	B	C
a	b	c
a	bb	cc
a	bb	c
a	b	cc

- A →→ B
- A →→ C

Multi-value Dependencies (MVD)

R		
$\overbrace{A1 \ldots Ai}^{\alpha}$	$\overbrace{Ai{+}1 \ldots Aj}^{\beta}$	$\overbrace{Aj{+}1 \ldots An}^{\gamma}$
a1 … ai	ai+1 … aj	aj+1 … an
a1 … ai	bi+1 … bj	bj+1 … bn
a1 … ai	bi+1 … bj	aj+1 … an
a1 … ai	ai+1 … aj	bj+1 … bn

- $\alpha \rightarrow\rightarrow \beta$ iff
  - $\forall$ t1, t2 $\in$ R: t1.α = t2.α $\Rightarrow$ $\exists$ t3, t4 $\in$ R:
    - t3.α = t4.α = t1.α = t2.α
    - t3.β = t1.β, t4.β = t2.β
    - t3.γ = t2.γ , t4.γ = t1.γ

MVD: Example

Skills		
PersNr	**Language**	**Programming**
3002	Greek	C
3002	Latin	Pascal
3002	Greek	Pascal
3002	Latin	C
3005	German	Ada

- MVDs of Skills
  - {PersNr}→→{Language}
  - {PersNr}→→{Programming}
  - {Language} {PersNr, Programming} (???)
- MVDs can result in anomalies and redundancy

Are MVDs symmetric?

- NO!
  - NOT {Language}$\rightarrow\rightarrow${PersNr}

Skills		
PersNr	Language	Programming
3002	Greek	C
3002	Latin	Pascal
3002	Greek	Pascal
3002	Latin	C
3005	German	Ada
3007	Greek	XQuery

- Exercise: Find examples for symmetric MVDs!

MVD: Example

Skills		
PersNr	Language	Programming
3002	Greek	C
3002	Latin	Pascal
3002	Greek	Pascal
3002	Latin	C
3005	German	Ada

$\Pi_{\text{PersNr, Language}}$

$\Pi_{\text{PersNr, Programming}}$

Language	
PersNr	Language
3002	Greek
3002	Latin
3005	German

Programming	
PersNr	Programming
3002	C
3002	Pascal
3005	Ada

MVD: Example

Skills		
PersNr	Language	Programming
3002	Greek	C
3002	Latin	Pascal
3002	Greek	Pascal
3002	Latin	C
3005	German	Ada

Join

Language	
PersNr	Language
3002	Greek
3002	Latin
3005	German

Programming	
PersNr	Programming
3002	C
3002	Pascal
3005	Ada

Example: Drinkers

- drinkers: {[name, addr, phones, beersLiked]}
  - some people have several phones
  - some people like several kind of beers

- FDs and MVDs
  - name $\rightarrow$ addr
  - name $\rightarrow\rightarrow$ phones
  - name $\rightarrow\rightarrow$ beersLiked

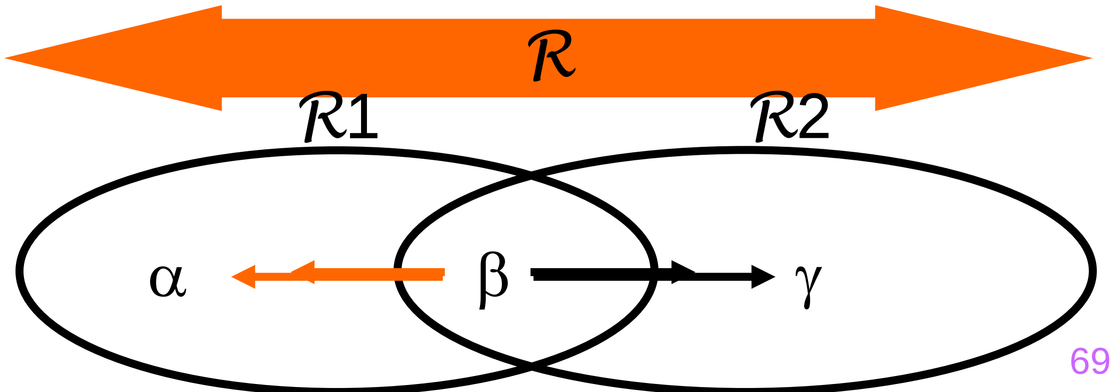
- Again, MVDs indicate redundancy
  - phones and beersLiked are independent concepts
  - (Would work if you could store them as sets: NFNF.)

Lossless Decompositions with MVDs

- Let $\mathcal{R} = \mathcal{R}1 \cup \mathcal{R}2$
  - R1 := $\Pi_{\mathcal{R}1}$ (R)
  - R2 := $\Pi_{\mathcal{R}2}$ (R)

- Lemma: The decomposition is lossless **iff**
  - ($\mathcal{R}1 \cap \mathcal{R}2$) $\rightarrow\rightarrow$ $\mathcal{R}1$ or
  - ($\mathcal{R}1 \cap \mathcal{R}2$) $\rightarrow\rightarrow$ $\mathcal{R}2$

- Exercise: Proof of this Lemma.
  Which direction is easier?



Laws of MVDs

- Trivial MVDs: $\alpha \twoheadrightarrow \mathcal{R}$
  - Check criterion of MVDs ($\beta = \mathcal{R}$, $\gamma = \emptyset$)
    - $\forall$ t1, t2 $\in$ R: t1.α = t2.$\alpha \Rightarrow \exists$ t3, t4 $\in$ R:
      - t3.α = t4.α = t1.α = t2.α
      - t3.β = t1.β, t4.β = t2.β
      - t3.γ = t2.γ , t4.γ = t1.γ
  - **let** t1, t2 $\in$ R: t1.α = t2.α
  - **set** t3=t1, t4=t2
    - t3.α = t4.α = t1.α = t2.α (by def. of t3, t4)
    - t3.β = t1.β (by def. of t3)
    - t4.β = t2.β (by def. of t4)
    - t3.γ = t2.γ ($\gamma = \emptyset$)
    - t4.γ = t1.γ ($\gamma = \emptyset$) qed
- Adapt proof for: $\alpha \twoheadrightarrow \mathcal{R} - \alpha$

Laws of MVDs

- Promotion: $\alpha \rightarrow \beta \Rightarrow \alpha \rightarrow\rightarrow \beta$
  - **let** t1, t2 $\in$ R: t1.α = t2.α
    - (1) t1.β = t2.β ($\alpha \rightarrow \beta$)
  - **set** t3=t2, t4=t1
    - t3.α = t4.α = t1.α = t2.α (by def. of t3, t4)
    - t3.β = t1.β (1)
    - t4.β = t2.β (1)
    - t3.γ = t2.γ (by def. of t3)
    - t4.γ = t1.γ (by def. of t4) qed
- $\alpha \rightarrow\rightarrow \beta \Rightarrow \alpha \rightarrow \beta$ does not always hold!
  - e.g., PersNr $\rightarrow\rightarrow$ Language, but PersNr $\not\rightarrow$ Language

Laws of MVDs

- Reflexivity: $(\beta \subseteq \alpha) \Rightarrow \alpha \rightarrow \beta$
- Augmentation: $\alpha \rightarrow \beta \Rightarrow \alpha\gamma \rightarrow \beta\gamma$
- Transitivity: $\alpha \rightarrow \beta \wedge \beta \rightarrow \gamma \Rightarrow \alpha \rightarrow \gamma$
- Complement: $\alpha \rightarrow\rightarrow \beta \Rightarrow \alpha \rightarrow\rightarrow \mathcal{R} - \beta - \alpha$
- Multi-value Augmentation: $\alpha \rightarrow\rightarrow \beta \wedge (\delta \subseteq \gamma) \Rightarrow \alpha\gamma \rightarrow\rightarrow \beta\delta$
- Multi-value Transitivity: $\alpha \rightarrow\rightarrow \beta \wedge \beta \rightarrow\rightarrow \gamma \Rightarrow \alpha \rightarrow\rightarrow \gamma$
- Generalization (Promotion): $\alpha \rightarrow \beta \Rightarrow \alpha \rightarrow\rightarrow \beta$

Laws of MVDs (ctd.)

- Coalesce: $\alpha \rightarrow\rightarrow \beta \wedge (\gamma \subseteq \beta) \wedge (\delta \cap \beta = \emptyset) \wedge \delta \rightarrow \gamma \Rightarrow \alpha \rightarrow \gamma$
- Multi-value Union: $\alpha \rightarrow\rightarrow \beta \wedge \alpha \rightarrow\rightarrow \gamma \Rightarrow \alpha \rightarrow\rightarrow \beta\gamma$
- Intersection: $\alpha \rightarrow\rightarrow \beta \wedge \alpha \rightarrow\rightarrow \gamma \Rightarrow \alpha \rightarrow\rightarrow (\beta \cap \gamma)$
- Minus: $\alpha \rightarrow\rightarrow \beta \wedge \alpha \rightarrow\rightarrow \gamma \Rightarrow \alpha \rightarrow\rightarrow (\beta - \gamma) \wedge \alpha \rightarrow\rightarrow (\gamma - \beta)$
- Warning: **NOT** $(\alpha \rightarrow\rightarrow \beta\gamma \Rightarrow \alpha \rightarrow\rightarrow \beta \wedge \alpha \rightarrow\rightarrow \gamma)$
  - Not all rules of FDs apply to MVDs!
  - Exercise: find an example for which this law does not hold

WARNING

- **NOT** $(\alpha \to\to \beta\gamma \Rightarrow \alpha \to\to \beta \wedge \alpha \to\to \gamma)$
  - Not all rules of FDs apply to MVDs!
  - E.g., {Language} $\to\to$ {PersNr, Programming}
  - But, NOT {Language} $\to\to$ {PersNr}
  - And NOT {Language} $\to\to$ {Programming}

- Be careful with MVDs
  - Not a totally intuitive concept

Trivial MVDs

- $\alpha \to\to \beta$ is trivial iff
  - $\beta \subseteq \alpha$ or
  - $\beta = \mathcal{R} - \alpha$
- Proof: a trivial MVD holds for any relation.

Fourth Normal Form (4NF)

- $\mathcal{R}$ is in 4NF iff for all $\alpha \twoheadrightarrow \beta$ at least one condition holds:
  - $\alpha \twoheadrightarrow \beta$ is trivial
  - α is a superkey of $\mathcal{R}$

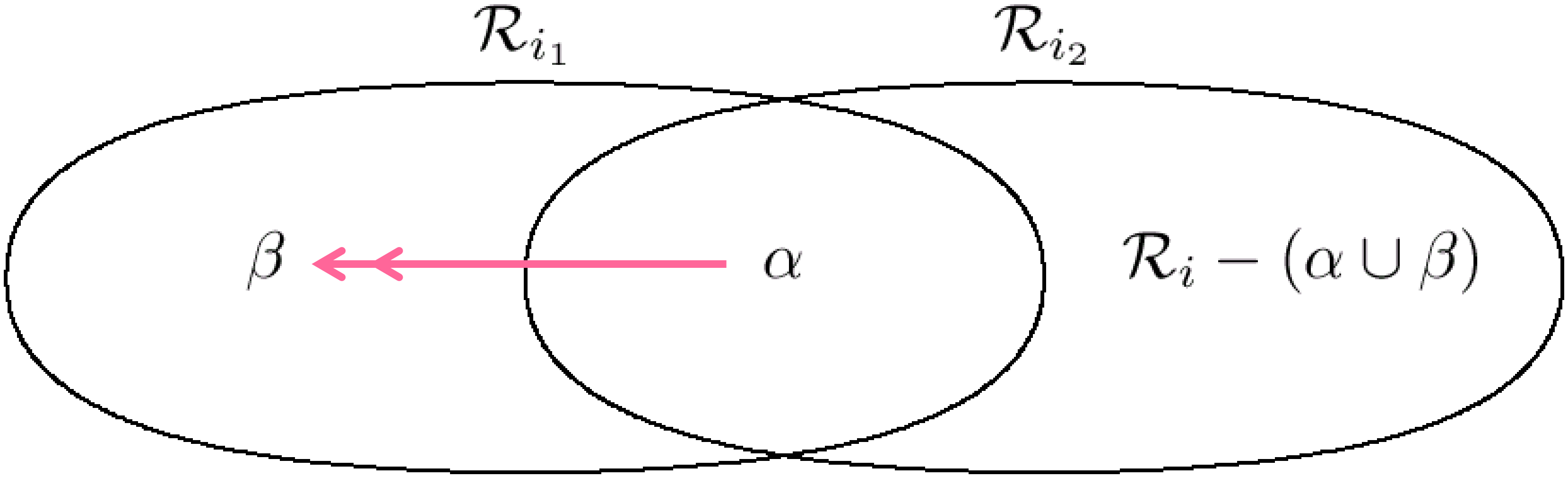
- $\mathcal{R}$ in 4NF implies $\mathcal{R}$ in BCNF
  - Proof is based on $\alpha \rightarrow \beta \Rightarrow \alpha \twoheadrightarrow \beta$.
  - (Can you prove that?)

Decomposition Algorithm for 4NF

- Input: $\mathcal{R}$
- Output: $\mathcal{R}1, \ldots, \mathcal{R}n$

result = {$\mathcal{R}$}
while ($\exists\ \mathcal{R}i \in Z$: $\mathcal{R}i$ is not in 4NF))
    let $\alpha \twoheadrightarrow \beta$ be evil in $\mathcal{R}i$
    $\mathcal{R}i1 = \alpha \cup \beta$
    $\mathcal{R}i2 = \mathcal{R}i - \beta$
    result = (result – {$\mathcal{R}i$}) $\cup$ {$\mathcal{R}i1$} $\cup$ {$\mathcal{R}i2$}
output(result)

Decomposition into 4 NF



Example

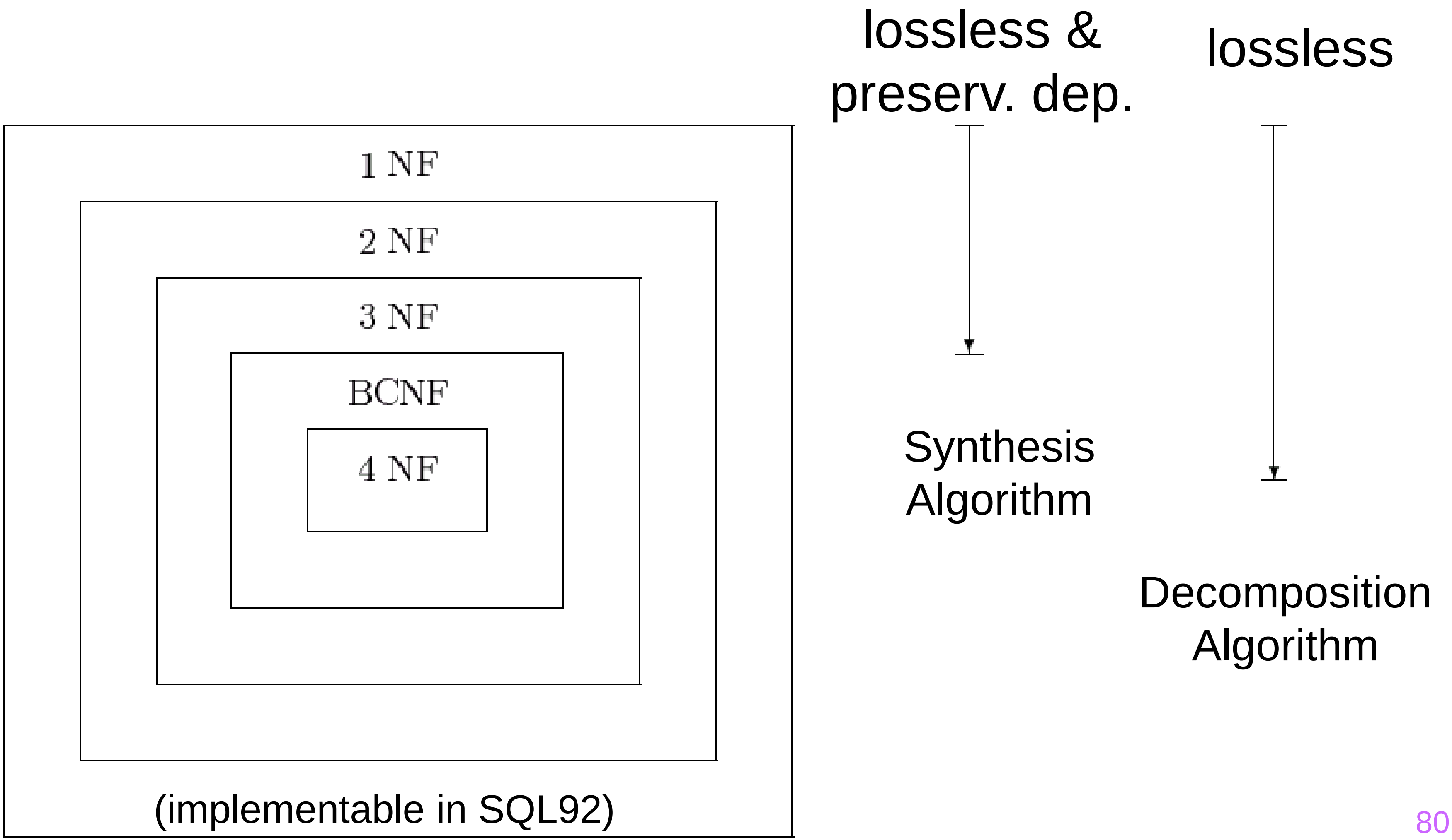
- Assistant: {[PersNr, Name, Area, Boss, Language, Progr.]}

- Synthesis Algorithm (3NF)
  - Assistant: {[PersNr, Name, Area, Boss]}
  - Skills: {[PersNr, Language, Programming]}

- Decomposition Algorithm (4NF)
  - Assistant: {[PersNr, Name, Area, Boss]}
  - Languages: {[PersNr, Language]}
  - Programming: {[PersNr, Programming]}

Summary

- Lossless decomposition up to 4NF
- Preserving dependencies up to 3NF

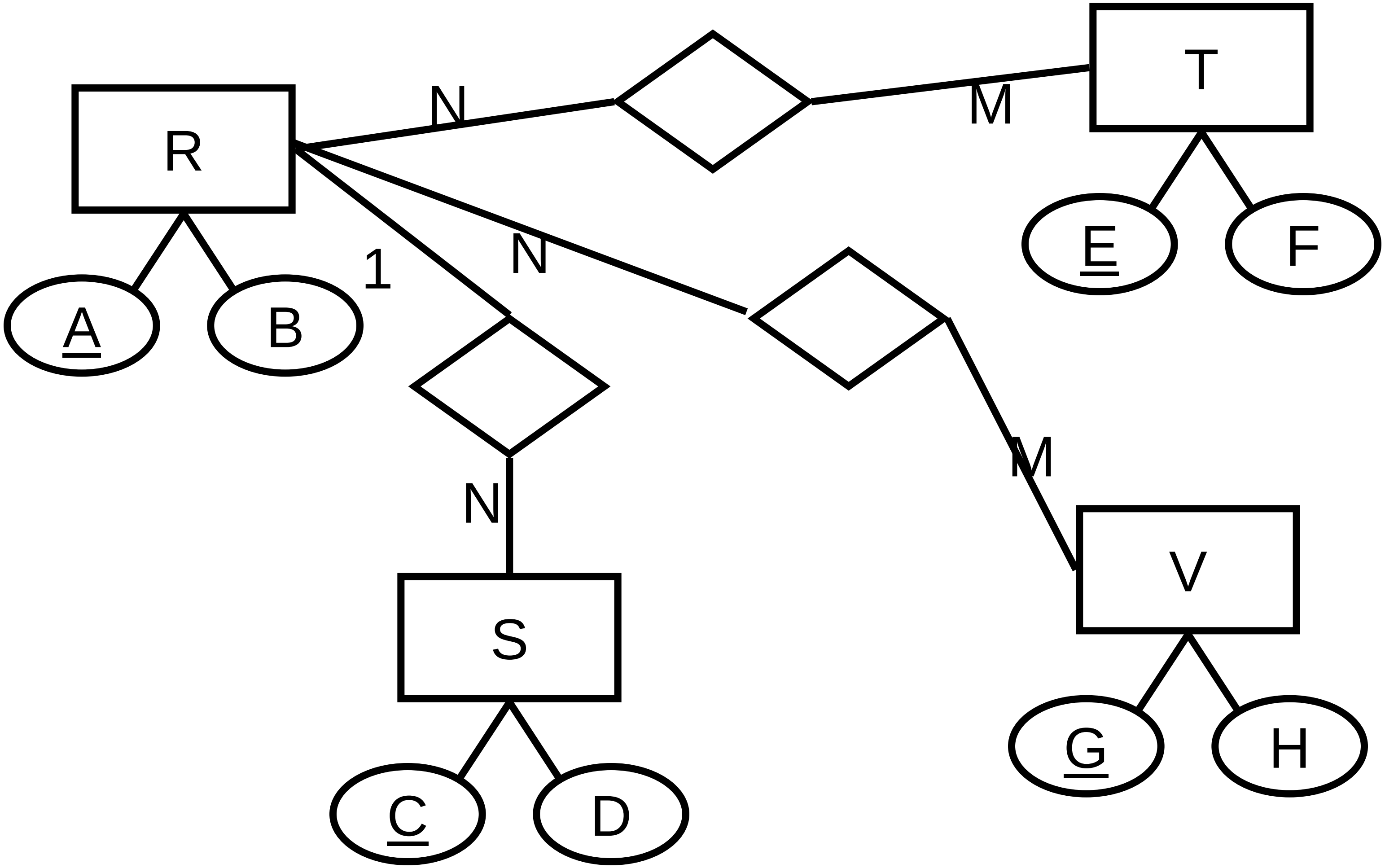


Exercise

Family Tree				
Child	Father	Mother	Grandpa	Grandma
Sofie	**Alfons**	**Sabine**	**Lothar**	**Linde**
Sofie	**Alfons**	**Sabine**	**Hubert**	**Lisa**
Niklas	**Alfons**	**Sabine**	**Lothar**	**Linde**
Niklas	**Alfons**	**Sabine**	**Hubert**	**Lisa**
Tobias	Leo	Bertha	Hubert	Martha
...	...	...	...	...

- Find FDs, MVDs, keys
- Decompose into 3NF, BCNF, 4NF

Exercise: 1:N & N:M Relationships



UR: {[A , B , C , D , E , F , G , H]}