NATIONAL RESEARCH UNIVERSITY

DATABASES
Fall 2021

Lecture 3. Relational Data Model

Alexander Breyman
Software Engineering Department
Computer Science Faculty

Relational Data Model

- ***Relation***:
  - $R \subseteq D_1$ x ... x D_n
  - D_1, D_2, ..., D_n are domains

  Example: AddressBook $\subseteq$ string x string x integer

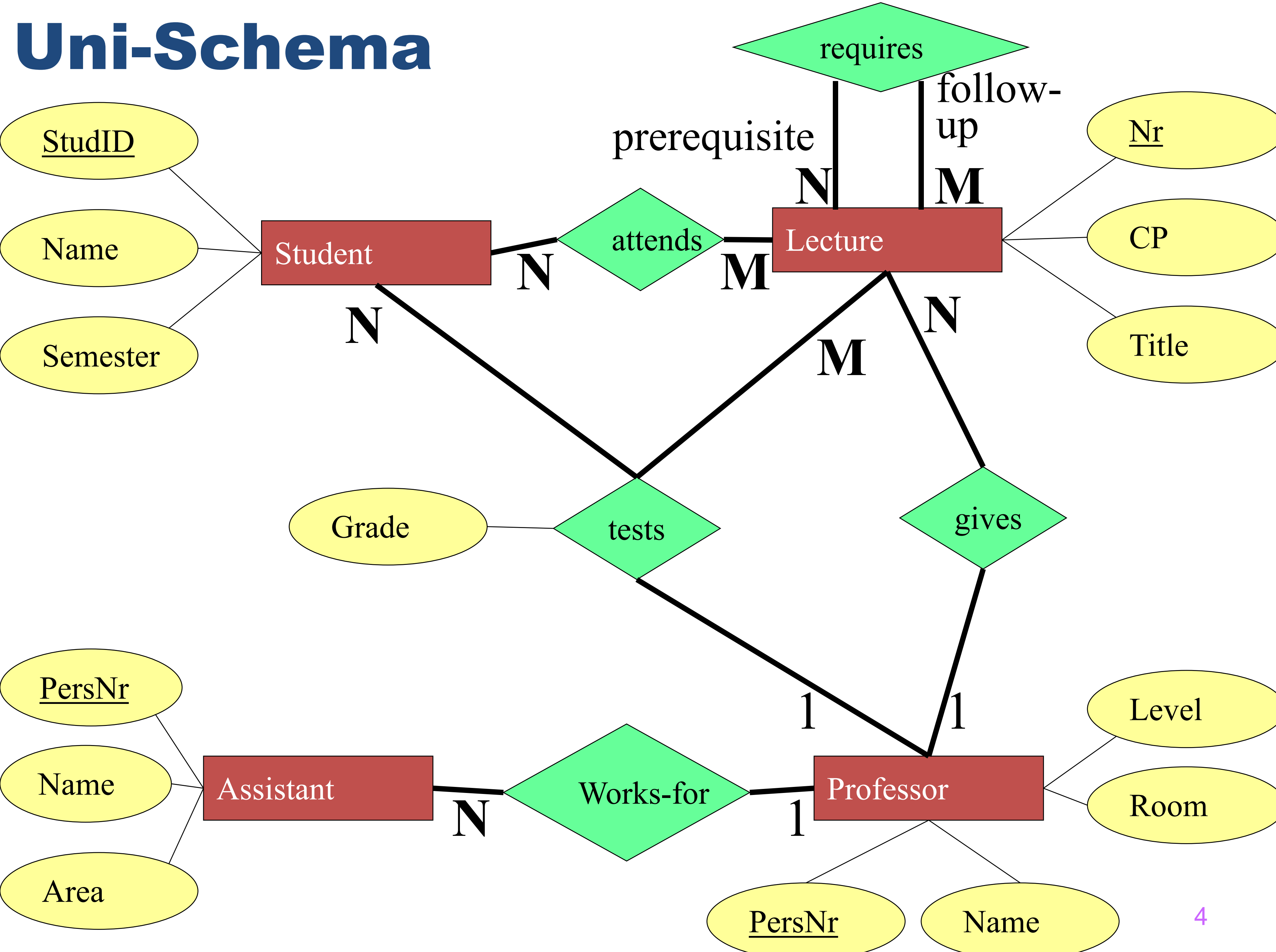
- ***Tuple***: $t \in R$

  Example: t = („Mickey Mouse“, „Main Street“, 4711)

AddrBook		
Name	**Street**	**Tel#**
Mickey Mouse	Main Street	4711
Minnie Mouse	Broadway	94725
Donald Duck	Broadway	95672
...	...	...

- ***Instance:*** the state of the database
- ***Key:*** minimal set of attributes that identify each tuple uniquely
    - E.g., {Tel#} or {Name, BirthDate}
- ***Primary Key:*** (marked in schema by underlining)
    - select one key
    - use primary key for references

Uni-Schema



requires
prerequisite
follow-up
N
M
StudID
Name
Semester
Student
N
attends
M
Lecture
Nr
CP
Title
N
M
N
Grade
tests
gives
1
1
PersNr
Name
Area
Assistant
N
Works-for
1
Professor
Level
Room
PersNr
Name

Rule #1: Implementation of Entities

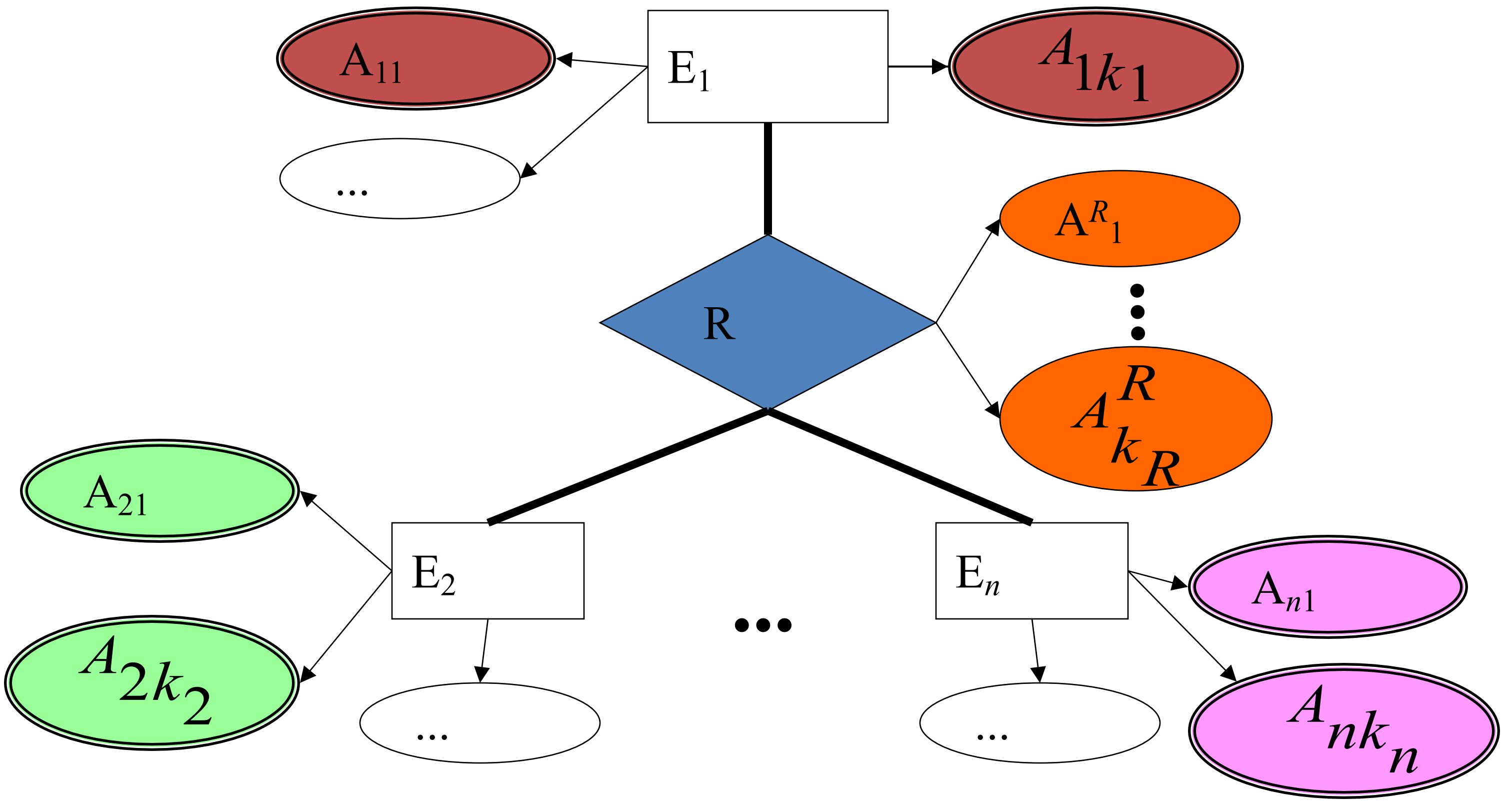
Student: {[StudID:integer, *Name*: *string, Semester*: *integer*]}

Lecture: {[Nr:integer, *Title*: *string, CP*: *integer*]}

Professor: {[PersNr:integer, *Name*: *string, Level*: *string, Room*: *integer*]}

Assistant: {[PersNr:integer, *Name*: *string, Area*: *string*]}

Rule #2: Relationships



$$R: \{[\underbrace{A_{11},\ldots, A_{1k_1}}_{\text{Key of } E_1}, \underbrace{A_{21},\ldots, A_{2k_2}}_{\text{Key of } E_2}, \ldots, \underbrace{A_{n1},\ldots, A_{nk_n}}_{\text{Key of } E_n}, \underbrace{A^R_1,\ldots, A^R_{k_R}}_{\text{Attributes of } R}]\}$$

Implementation of Relationships

attends : {[StudID: integer, Nr: integer]}

gives : {[PersNr: integer, Nr: integer]}

works-for : {[AssistantPersNr: integer, ProfPersNr: integer]}

requires: {[prerequisite: integer, follow-up: integer]}

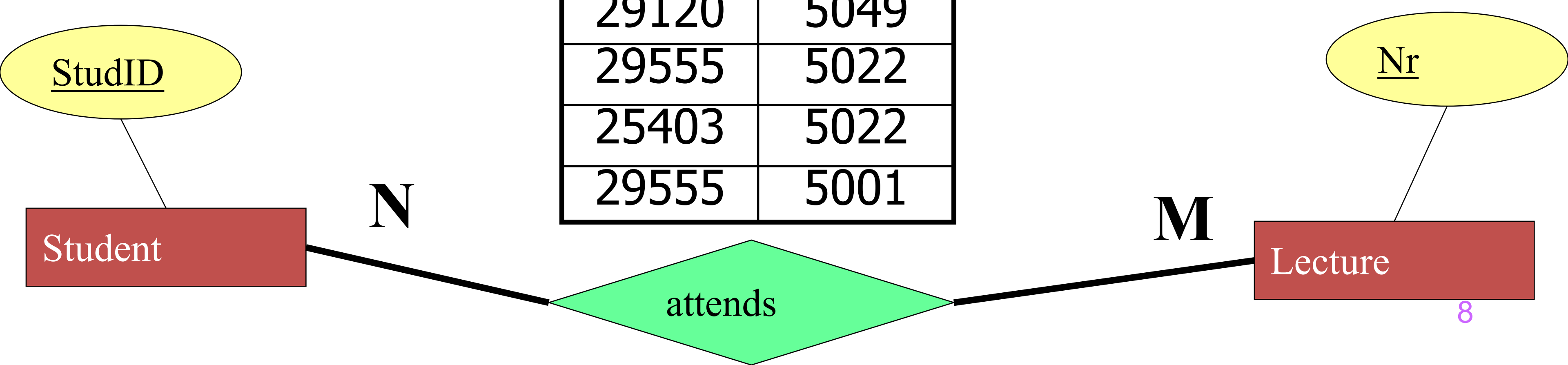
tests : {[StudID: integer, Nr: integer, PersNr: integer, Grade: decimal]}

Instance of *attends*

Student	
StudID	...
26120	...
27550	...
...	...

attends	
StudID	Nr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

Lecture	
Nr	...
5001	...
4052	...
...	...

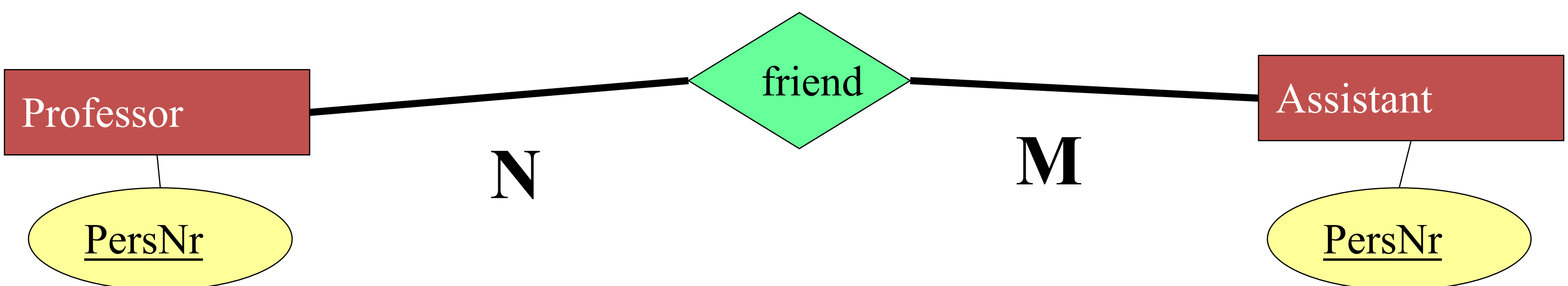


Rule 2: How to call the attributes

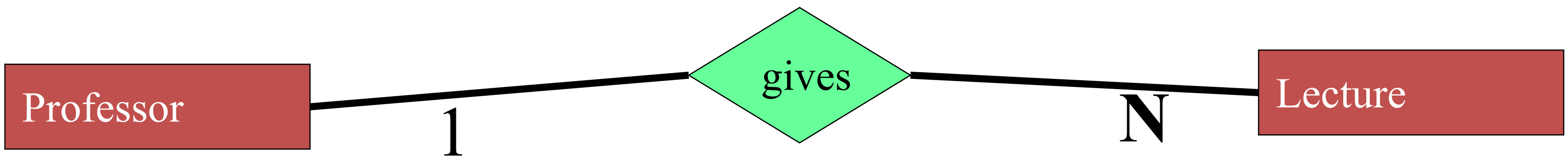
- If the ER specifies roles
  - use the names of the roles

- Otherwise
  - use the names of the key attributes in the entities
  - in case of ambiguity, invent new names

- Example: **friend** : {[<u>ProfNr: integer, AssiNr: integer</u>]}



Rule #3: Merge relations with the same key



- Implementation according to Rule #2

  Lecture : {[*<u>Nr</u>, Title, CP*]}
  Professor : {[*<u>PersNr</u>, Name, Level, Room*]}
  gives: {[*<u>Nr</u>, PersNr*]}

- Merge according to Rule #3

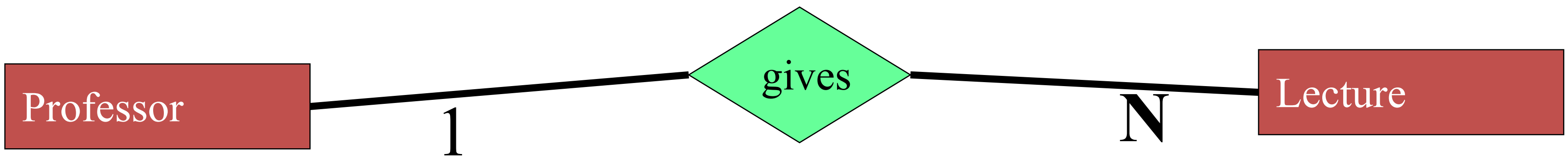
  Lecture : {[*<u>Nr</u>, Title, CP,* ***PersNr***]}
  Professor : {[*<u>PersNr</u>, Name, Level, Room*]} 1

Why is this better? When can this be done?

Instance of *Professor* and *Lecture*

Professor			
PersNr	Name	Level	Room
2125	Sokrates	FP	226
2126	Russel	FP	232
2127	Kopernikus	AP	310
2133	Popper	AP	52
2134	Augustinus	AP	309
2136	Curie	FP	36
2137	Kant	FP	7

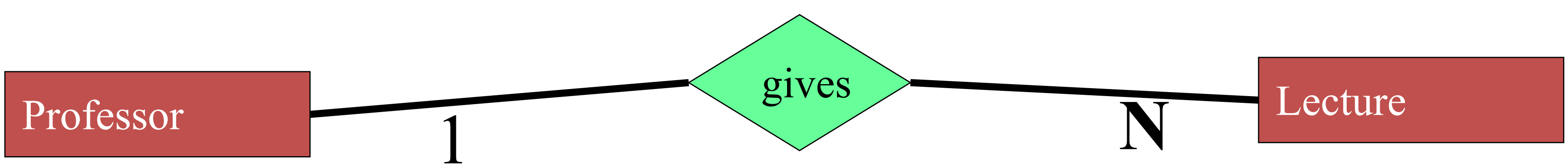
Lecture			
Nr	Title	CP	PersNr
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137



This will NOT work

Professor				
PersNr	Name	Level	Room	gives
2125	Sokrates	FP	226	5041
2125	Sokrates	FP	226	5049
2125	Sokrates	FP	226	4052
...	...	...	...	...
2134	Augustinus	AP	309	5022
2136	Curie	FP	36	??

Lecture		
Nr	Title	CP
5001	Grundzüge	4
5041	Ethik	4
5043	Erkenntnistheorie	3
5049	Mäeutik	2
4052	Logik	4
5052	Wissenschaftstheorie	3
5216	Bioethik	2
5259	Der Wiener Kreis	2
5022	Glaube und Wissen	2
4630	Die 3 Kritiken	4



This will NOT work

Professor				
PersNr	Name	Level	Room	gives
2125	Sokrates	FP	226	5041
2125	Sokrates	FP	226	5049
2125	Sokrates	FP	226	4052
...	...	...	...	...
2134	Augustinus	AP	309	5022
2136	Curie	FP	36	??

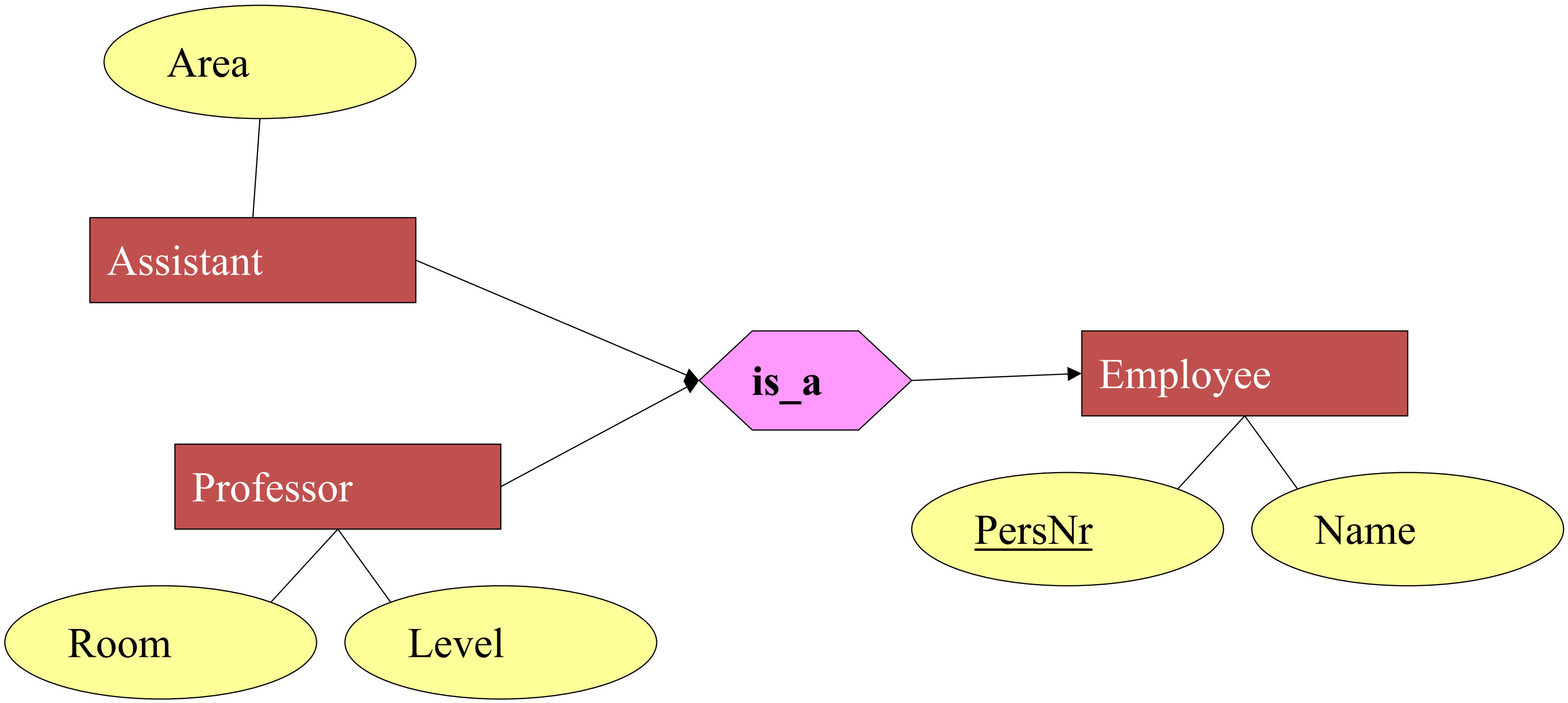
Lecture		
Nr	Title	CP
5001	Grundzüge	4
5041	Ethik	4
5043	Erkenntnistheorie	3
5049	Mäeutik	2
4052	Logik	4
5052	Wissenschaftstheorie	3
5216	Bioethik	2
5259	Der Wiener Kreis	2
5022	Glaube und Wissen	2
4630	Die 3 Kritiken	4

Problem: Redundancy and Anomalies

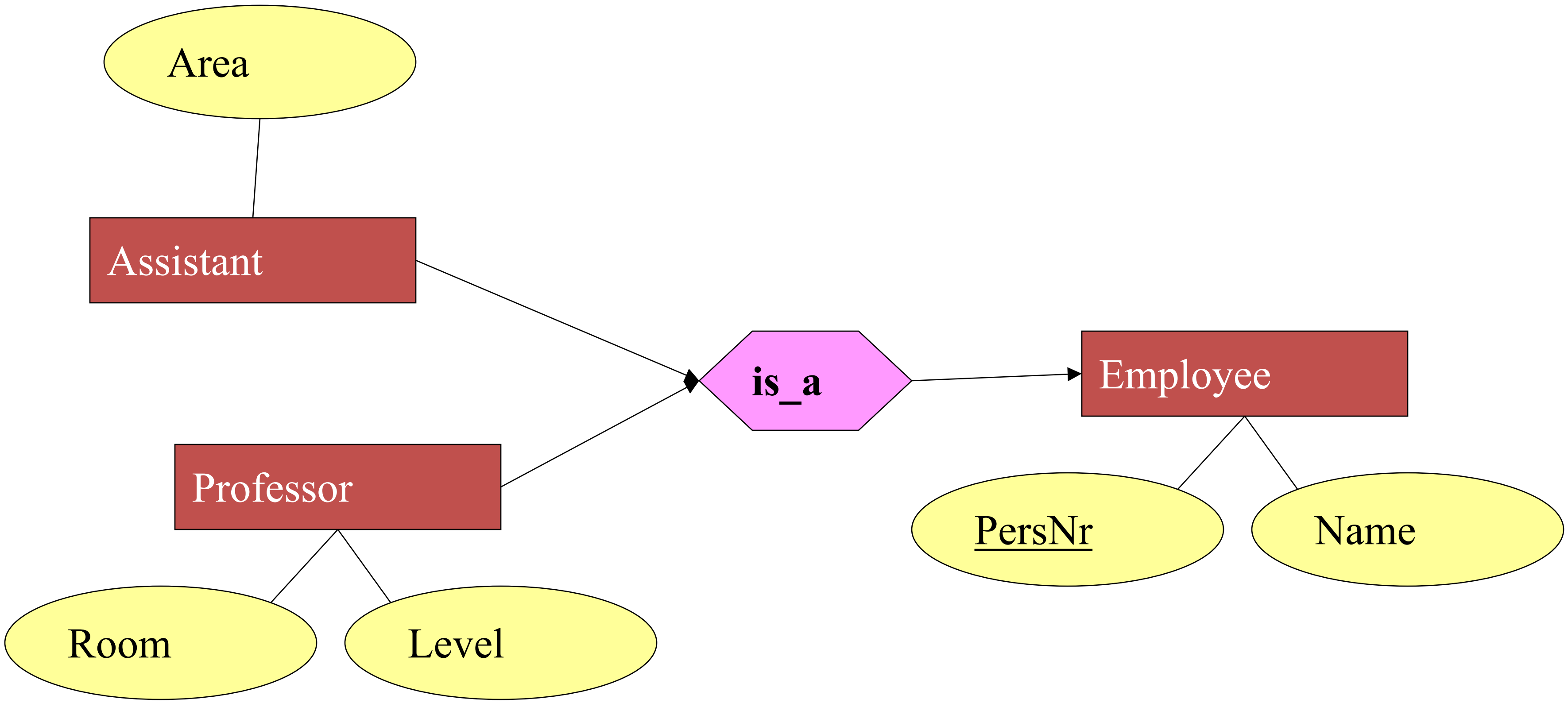
PersNr is no longer key of Professor

(issue will be revisited when we talk about normal forms)

Rule #4: Generalization



Rule #4: Generalization

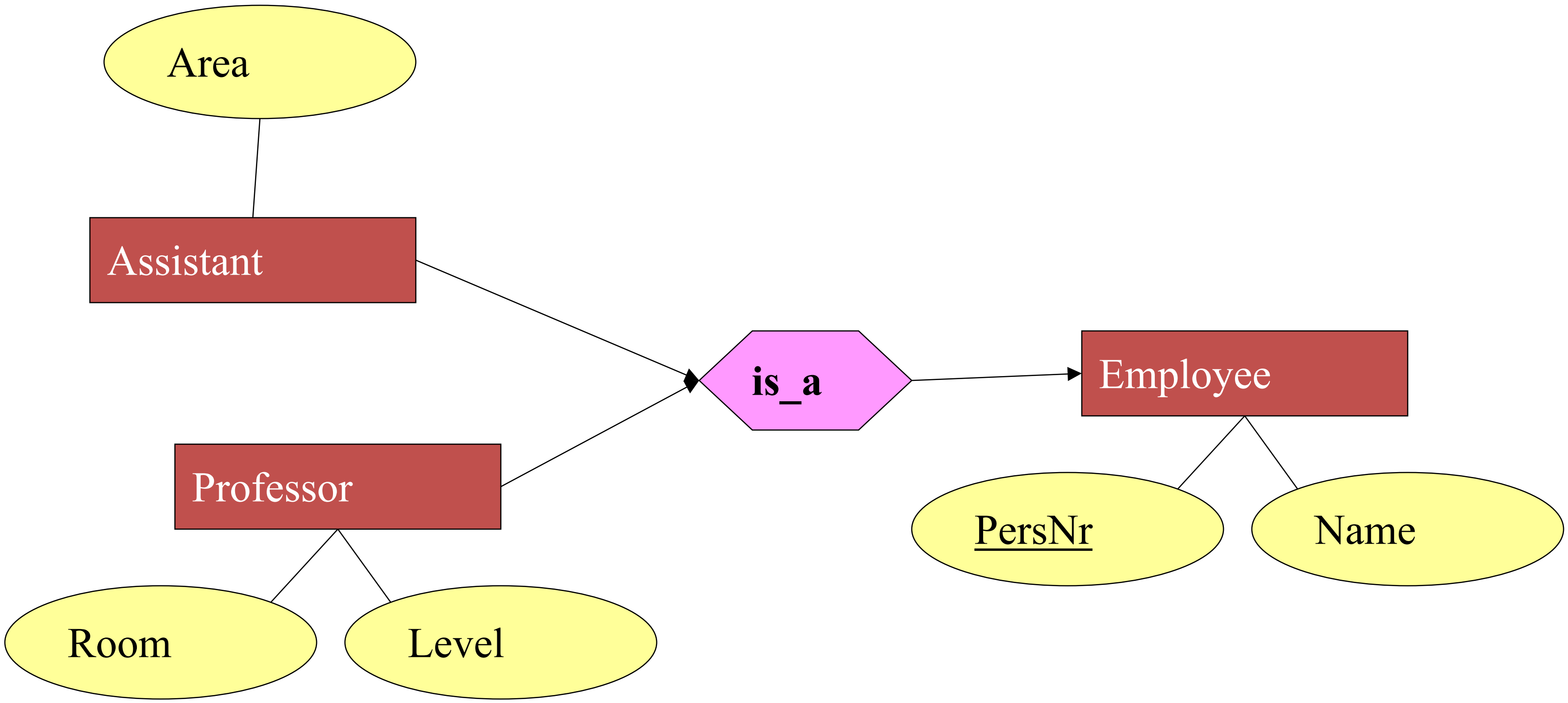


Employee: {[*<u>PersNr</u>, Name*]}

Professor: {[*<u>PersNr</u>, Level, Room*]}

Assistant: {[*<u>PersNr</u>, Area*]}

Rule #4: Generalization (alternative)



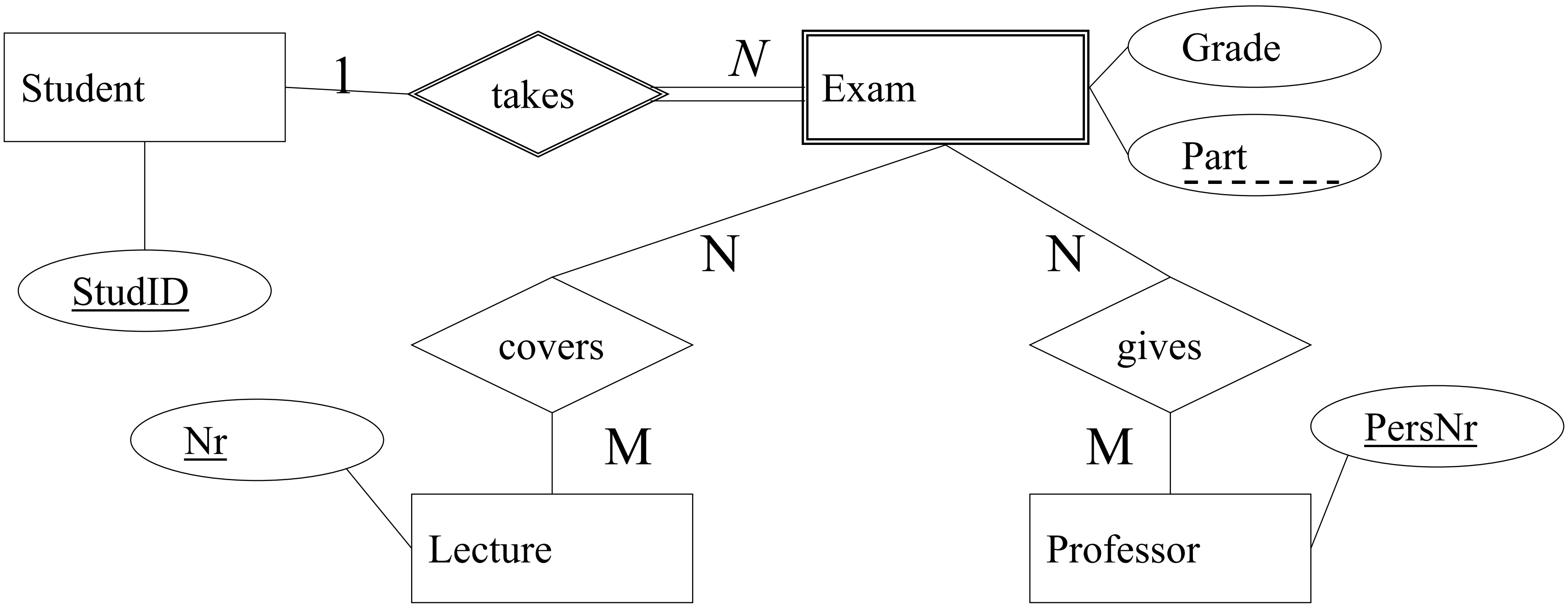
What is better?

Employee: {[*<u>PersNr</u>, Name*]}

Professor: {[*<u>PersNr</u>, Name, Level, Room*]}

Assistant: {[*<u>PersNr</u>, Name, Area*]}

Rule #5: Weak Entities



Exam: {[StudID: integer, Part: string, Grade: integer]}

covers: {[StudID : integer, Part: string, Nr: integer]}

gives: {[StudID : integer, Part: string, PersNr: integer]}

Food for Thought: OO vs Relations

- How do Java and C++ implement ER?
  - Are they a better match than the relational model?
- Specifically, how do Java and C++ implement Generalization?
  - Is it good or bad to have several possible ways?
- Concept of Reference: Compare Java and Relational Model
  - Which one is better?
- Life-time of objects: Compare Java and Relational Model
  - Why different?

Relational Model of Uni-DB

Professor

PersNr	Name	Level	Room
2125	Sokrates	FP	226
2126	Russel	FP	232
2127	Kopernikus	AP	310
2133	Popper	AP	52
2134	Augustinus	AP	309
2136	Curie	FP	36
2137	Kant	FP	7

Student

Legi	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Lecture

Nr	Title	CP	PersNr
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

requires

Prerequisite	Follow-up
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

attends

Legi	Nr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

Assistant

PerslNr	Name	Area	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

tests

Legi	Nr	PersNr	Grade
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Relational Algebra

- σ Selection
- π Projection
- X Cartesian Product
- A Join
- ρ Rename
- – Set Minus
- ÷ Relational Division
- ∪ Union
- ∩ Intersection
- F Semi-Join (left)
- E Semi-Join (right)
- C left outer Join
- D right outer Join

Formal Definition of Rel. Algebra

Atoms (basic expressions)

- A relation in the database
- A constant relation

Operators (composite expressions)

- Selection: $\sigma_p (E_1)$
- Projection: $\Pi_S (E_1)$
- Cartesian Product: $E_1 \times E_2$
- Rename: $\rho_V (E_1)$, $\rho_{A \leftarrow B} (E_1)$
- Union: $E_1 \cup E_2$
- Minus: $E_1 - E_2$

Selection and Projection

Selection

$\sigma_{Semester > 10}$ (Student)

$\sigma_{Semester > 10}$ (Student)		
Legi	Name	Semester
24002	Xenokrates	18
25403	Jonas	12

Projection

Π_{Level}(Professor)

Π_{Rang}(Professor)
Level
FP
AP

Cartesian Product

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

x

R	
D	E
d_1	e_1
d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2

Cartesian Product (ctd.)

Professor x attends

Professor				attends	
PersNr	Name	Level	Raum	Legi	Nr
2125	Sokrates	FP	226	26120	5001
...	...	...	...	...	...
2125	Sokrates	FP	226	29555	5001
...	...	...	...	...	...
2137	Kant	FP	7	29555	5001

- Huge result set (n * m)
- Usually only useful in combination with a selection (-> Join)

Natural Join

Two relations:

- $R(A_1,\ldots, A_m, B_1,\ldots, B_k)$
- $S(B_1,\ldots, B_k, C_1,\ldots, C_n)$

$$R \text{ A } S = \Pi_{A1,\ldots,\, Am,\, R.B1,\ldots,\, R.Bk,\, C1,\ldots,\, Cn}(\sigma_{R.B1=S.\,B1\ \wedge\ldots\wedge\ R.Bk\,=\,S.Bk}(RxS))$$

R A S											
$R - S$				$R \cap S$				$S - R$			
A_1	A_2	...	A_m	B_1	B_2	...	B_k	C_1	C_2	...	C_n
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Three-way natural Join

(Student A attends) A Lecture

(Student A attends) A Lecture						
Legi	Name	Semester	Nr	Title	CP	PersNr
26120	Fichte	10	5001	Grundzüge	4	2137
27550	Jonas	12	5022	Glaube und Wissen	2	2134
28106	Carnap	3	4052	Wissenschftstheorie	3	2126
...	...	...	...	...	...	...

Theta-Join

Two Relations:

– R(A1, ..., An)

$R \ A_{\theta} \ S = \sigma_{\theta}(R \text{ x } S)$

$R \ A_{\theta} \ S$

$R \ A_{\theta} \ S$							
R				S			
A_1	A_2	…	A_n	B_1	B_2	…	B_m
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Join Variants

- natural join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

A

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1

- left outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

C

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-

Join Variants

- right outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

D

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
-	-	c_3	d_2	e_2

Join Variants

- (full) outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

B

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-
-	-	c_3	d_2	e_2

- left semi join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

E

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat		
A	B	C
a_1	b_1	c_1

Join Variants

- right semi join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

F

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat		
C	D	E
c_1	d_1	e_1

Rename Operator

Rename operator: ρ

- Renaming of relation names
  - Needed to process self-joins and recursive relationships
  - E.g., two-level dependencies of lectures („grandparents“)

$$\Pi_{\text{L1.Prerequisite}}(\sigma_{\text{L2. Follow-up}=5216 \wedge \text{L1.Follow-up} = \text{L2.Prerequisite}}$$
$$(\rho_{L1}(\text{requires}) \times \rho_{L2}(\text{requires})))$$

- Renaming of attribute names

$$\rho_{\text{Requirement} \leftarrow \text{Prerequisite}}(\text{requires})$$

Intersection

$\Pi_{\text{PersNr}}(\text{Lecture}) \cap \Pi_{\text{PersNr}}(\sigma_{\text{Level=FP}}(\text{Professor}))$

- Only works if both relations have the same schema
  - Same attribute names and attribute domains
- Intersection can be simulated with minus:

$$R \cap S = R - (R - S)$$

Relational Division

Find students who attended **all** lectures with 4CP.

$$\text{attends} \div \Pi_{Nr}(\sigma_{CP=4}(\text{Lecture}))$$

Definition of Division

- $t \in R \div S$, iff for each $ts \in S$ exists a $tr \in R$ such that:
  - tr.S = ts.S
  - tr.(R-S) = t

R	
M	V
m_1	v_1
m_1	v_2
m_1	v_3
m_2	v_2
m_2	v_3

$\div$

S
V
v_1
v_2

=

$R \div S$
M
m_1

- Division can be simulated with other operators:

$$R \div S = \Pi_{(R-S)}(R) - \Pi_{(R-S)}((\Pi_{(R-S)}(R) \times S) - R)$$

Division: Example

$R \div S = \Pi_{(R-S)}(R) - \Pi_{(R-S)}((\Pi_{(R-S)}(R) \times S) - R)$

- R = attends; S = Lecture
- **Π_{Legi}(attends)**
  All students (who attend at least one lecture)
- **Π_{Legi}(attends) x Lecture**
  All students attend all lectures
- **(Π_{Legi}(attends) x Lecture) – attends**
  Lectures a student does not attend
- **Π_{Legi}((Π_{Legi}(attends) x Lecture) – attends):**
  Students who miss at least one lecture
- Π_{Legi}(hören) - Π_{Legi}((Π_{Legi}(attends) x Lecture) – attends)
  Students who attend all lectures
  What happens if there are no lectures or no attendance?

Relational Calculus

Queries have the following form:

$\{t \mid P(t)\}$

with t a variable, P(t) a predicate.

Examples:

- All full professors

  $\{p \mid p \in \text{Professor} \wedge p.\text{Level} = \text{,FP'}\}$

- Students who attend at least one lecture of Curie

  $$\begin{aligned}\{s \mid\ & s \in \text{Student} \\ & \wedge \exists a \in \text{attends}(s.\text{Legi}=a.\text{Legi} \\ & \wedge \exists l \in \text{Lecture}(a.\text{Nr}=l.\text{Nr} \\ & \wedge \exists p \in \text{Professor}(p.\text{PersNr}=l.\text{PersNr} \\ & \wedge p.\text{Name} = \text{'Curie'})))\}\end{aligned}$$

- Who attends all lectures with 4 CP?

$$\{s \mid s \in Student \wedge \forall l \in Lecture\ (l.CP=4 \Rightarrow \exists a \in attends(a.Nr=l.Nr \wedge a.Legi= s.Legi))\}$$

- There are two variants of relational calculus:
  - tuple relational calculus (as in examples above, tuple vars)
  - domain relational calculus (variables iterate over domains)

Tuple Relational Calculus

Atoms

- $s \mid R$
  - s is a tuple variable, R is a name of a relation
- $s.A \ \phi \ t.B$ or $s.A \ \phi \ c$
  - s and t tuple variables, A and B attribute names
    ϕ a comparison (i.e., $=$, $\neq$, $\leq$, ...)
  - c is a constant (i.e., 25)

Formulas

- All atoms are legal formulas
- If P is a formula, then $\neg P$ and (P) are also formulas
- If P_1 and P_2 are formulas, then $P_1 \wedge P_2$, $P_1 \vee P_2$ and $P_1 \Rightarrow P_2$
- If $P(t)$ is a formula with a free variable t, then
  $\forall t \in R(P(t))$ and $\exists\, t \in R(P(t))$

Safety

- Restrict formulas to queries with finite answers
  - Semantic not syntactic property!

- Example: The following expression is not safe

$$\{n \mid \neg (n \in \text{Professor})\}$$

- Definition of safety
  - result must be subset of the „domain of the formula“
  - „domain of the formula“
    - All constants used in the formula
    - All domains of relations used in the formula

Domain Relational Calculus

An expression has the following form

$\{[v_1, v_2, ..., v_n] \mid P(v_1, ..., v_n)\}$

each $v_1, ..., v_n$ is either a domain variable or a constant.

P is a formula.

Example: Legi and Name of all students tested by Curie:

$\{[l, n] \mid \exists s\,([l, n, s] \in \text{Student} \wedge \exists v, p, g\,([l, v, p, g] \in \text{tests} \wedge$
$\exists a, r, b([p, a, r, b] \in \text{Professor} \wedge a = \text{'Curie'})))\}$

Safety in the DRC

- Defined in same way as for tuple relational calculus

- Example: The following expression is not safe

$$\{[p,n,r,o] \mid \neg ([p,n,r,o] \in \text{Professoren})\}$$

Codd`s Theorem

The three languages

1. relational algebra,
2. tuple relational calculus (safe expressions only)
3. domain relational calculus (safe expressions only)

are **equivalent**

Impact of Codd`s theorem

- SQL is based on the relational calculus
- SQL implementation is based on relational algebra
- Codd`s theorem shows that SQL implementation is correct and complete.

Homework Assignment 3

1. Why any relation in a relational schema has at least one key?
2. What happens when one implements the ER diagram from previous homework (Homework #2, Library system) with a single relation (instead of creating distinct relations for entities and relationships)?
3. Translate all ER diagrams from previous homework into relational schemas.

NATIONAL RESEARCH
UNIVERSITY

Thank you
for your attention!

20, Myasnitskaya str., Moscow, Russia, 101000
Tel.: +7 (495) 628-8829, Fax: +7 (495) 628-7931
www.hse.ru