

UNIT 0x05

RELATIONALES MODELL

RELATIONALE ALGEBRA I

Motivation/Erinnerung

Aufgabe

- Entwicklung einer 'guten' Datenbank zu einem realen Problem (z.B. Mini-Welt), d.h. u.a. vollständig, korrekt, minimal, aber auch redundanzfrei und effizient...

Vorgehen

- Konsolidierung und Konzeptueller Entwurf in Form eines ER-Modells ✓ (Beispiel)
- Überführung in eine 'gute' Datenbank... dazu benötigen wir:
 - Implementationsentwurf, z.B. mit Tabellen und Relationen
→ Relationales Modell
 - Mathematisches Modell der Datenbankoperationen, z.B. zur Anfrageoptimierung
→ Relationale Algebra
 - Gütemaß für den Implementationsentwurf
→ Normalformen

Wiederholung Modellierung

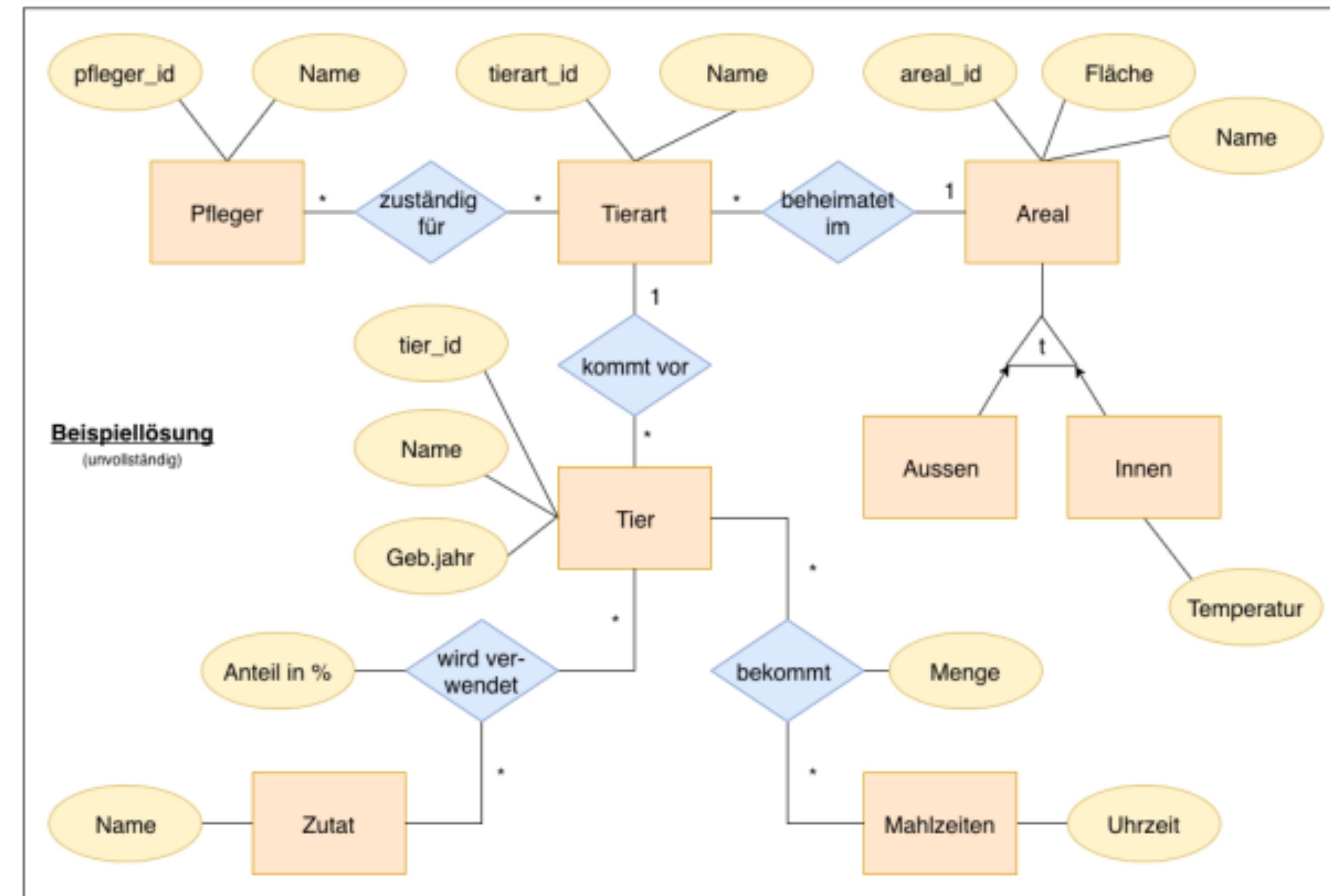
Aufgabe GaiaZOO

[...] Er beherbergt verschiedene Tierarten, die von Pflegern betreut werden. Alle Tiere einer Art befinden sich in genau einem ganz bestimmten Areal im Zoo. Dort können aber natürlich mehrere Tierarten leben. Aufgrund unterschiedlicher Allergien bekommt jedes Tier eine individuelle fest vorgegebene Futtermischung mit bestimmten Zutaten. Weiter gilt:

- Pfleger kümmern sich nicht um einzelne Tiere, sondern allgemein um alle Tiere einer Tierart. D.h. Ihnen sind bestimmte Tierarten zugeordnet. In der Regel sind das je Tierart mindestens zwei.
- Falls das letzte Tier einer Tierart sterben sollte, wird umgehend ein neues Tier dieser Art erworben. Dennoch kommt es vor, dass zeitweilig ein Gehege auch leer sein kann.
- Die Areale sind in genau zwei Kategorien eingeteilt: Innen- und Aussengehege. Für jedes Areal ist die Fläche in m² wichtig, aber die Innengehege besitzen ausserdem noch eine Temperaturvorgabe. Vereinzelt kommt es vor, dass Tiere innen und aussen leben können.
- Für die Zutaten einer individuellen Futtermischung eines Tieres ist gewünscht, sie in Prozent der Gesamtmenge einer Mahlzeit anzugeben. Jedes Tier bekommt verschiedene Mahlzeiten am Tag, wobei dann eine konkrete Mengenangabe in kg je Mahlzeit ausreicht, um die Mahlzeit zusammenzustellen. Mögliche Mahlzeiten sind Morgens 6 Uhr, Mittags 12 Uhr, Abends 18 Uhr und gegen Mitternacht 24 Uhr.

Wiederholung Modellierung

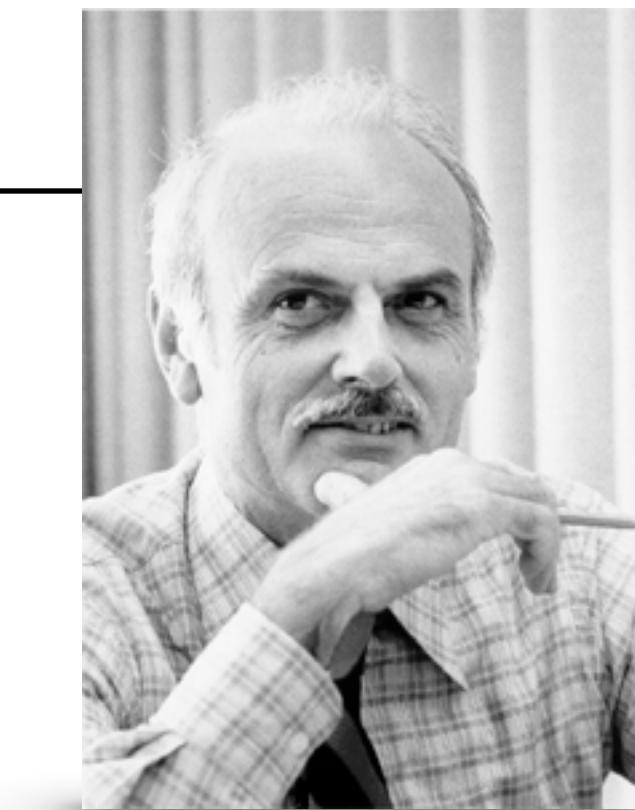
Lösungsvorschlag GaiaZoo



Historische Entwicklung

Entwickelt von Edgar (Ted) Codd, IBM

- Relational Model of Data for Large Shared Data Banks,
Comm. ACM, Juni 1970
- Formale Grundlage DBMS, Relationale Algebra, Normalform



Information Retrieval

A Relational Model of Data for Large Shared Data Banks

E. F. CODD

IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation

2.1.2. Projection. Suppose now we select certain columns of a relation (striking out the others) and then remove from the resulting array any duplication in the rows. The final array represents a relation which is said to be a *projection* of the given relation.

A selection operator π is used to obtain any desired permutation, projection, or combination of the two operations. Thus, if L is a list of k indices⁷ $L = i_1, i_2, \dots, i_k$ and R is an n -ary relation ($n \geq k$), then $\pi_L(R)$ is the k -ary relation whose j th column is column i_j of R ($j = 1, 2, \dots, k$) except that duplication in resulting rows is removed. Consider the relation *supply* of Figure 1. A permuted projection of this relation is exhibited in Figure 4. Note that, in this particular case, the projection has fewer n -tuples than the relation from which it is derived.

2.1.3. Join. Suppose we are given two binary relations, which have some domain in common. Under what circumstances can we combine these relations to form a

⁷ When dealing with relationships, we use domain names (role-qualified whenever necessary) instead of domain positions.

$R * S$	(supplier)	part	project
1	1	1	1
1	1	1	2
2	1	1	1
2	1	1	2
2	2	2	1

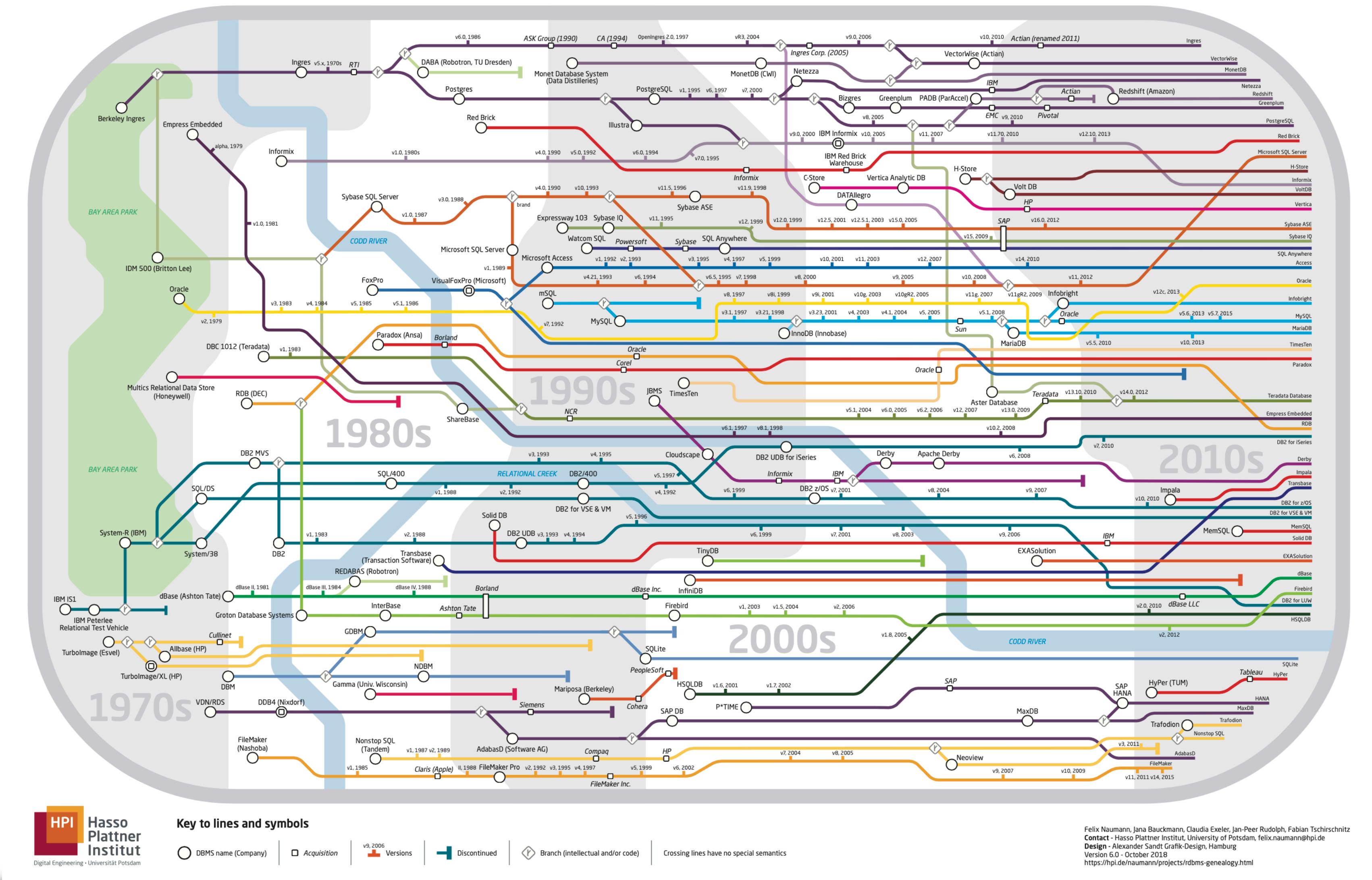
FIG. 6. The natural join of R with S (from Figure 5)

U	(supplier)	part	project
1	1	1	2
2	1	1	1
2	2	2	1

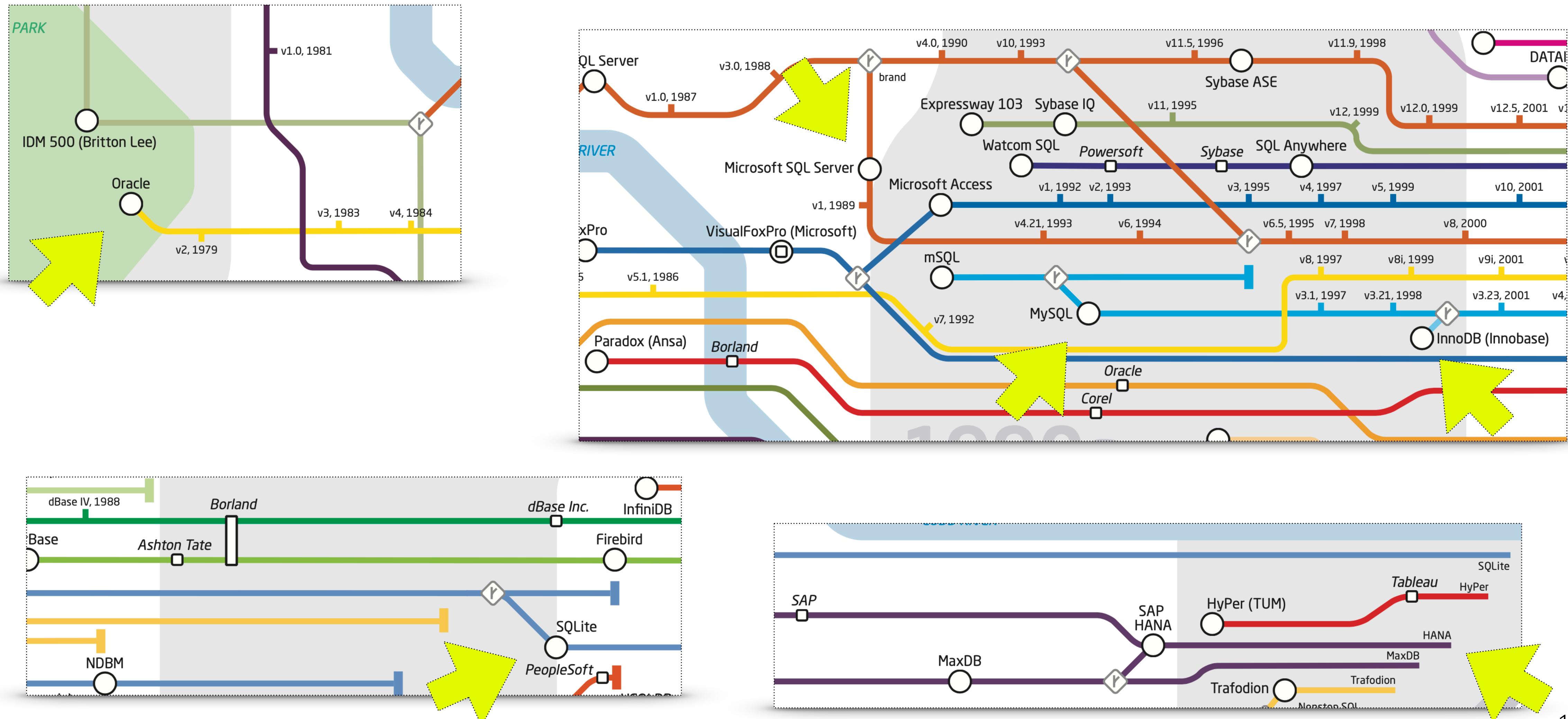
FIG. 7. Another join of R with S (from Figure 5)

Inspection of these relations reveals an element (element 1) of the domain *part* (the domain on which the join is to be made) with the property that it possesses more than one relative under R and also under S . It is this ele-

Historische Entwicklung - Genealogy of RDBMS



Historische Entwicklung - Genealogy of RDBMS



ER-Modell → Relationales Modell

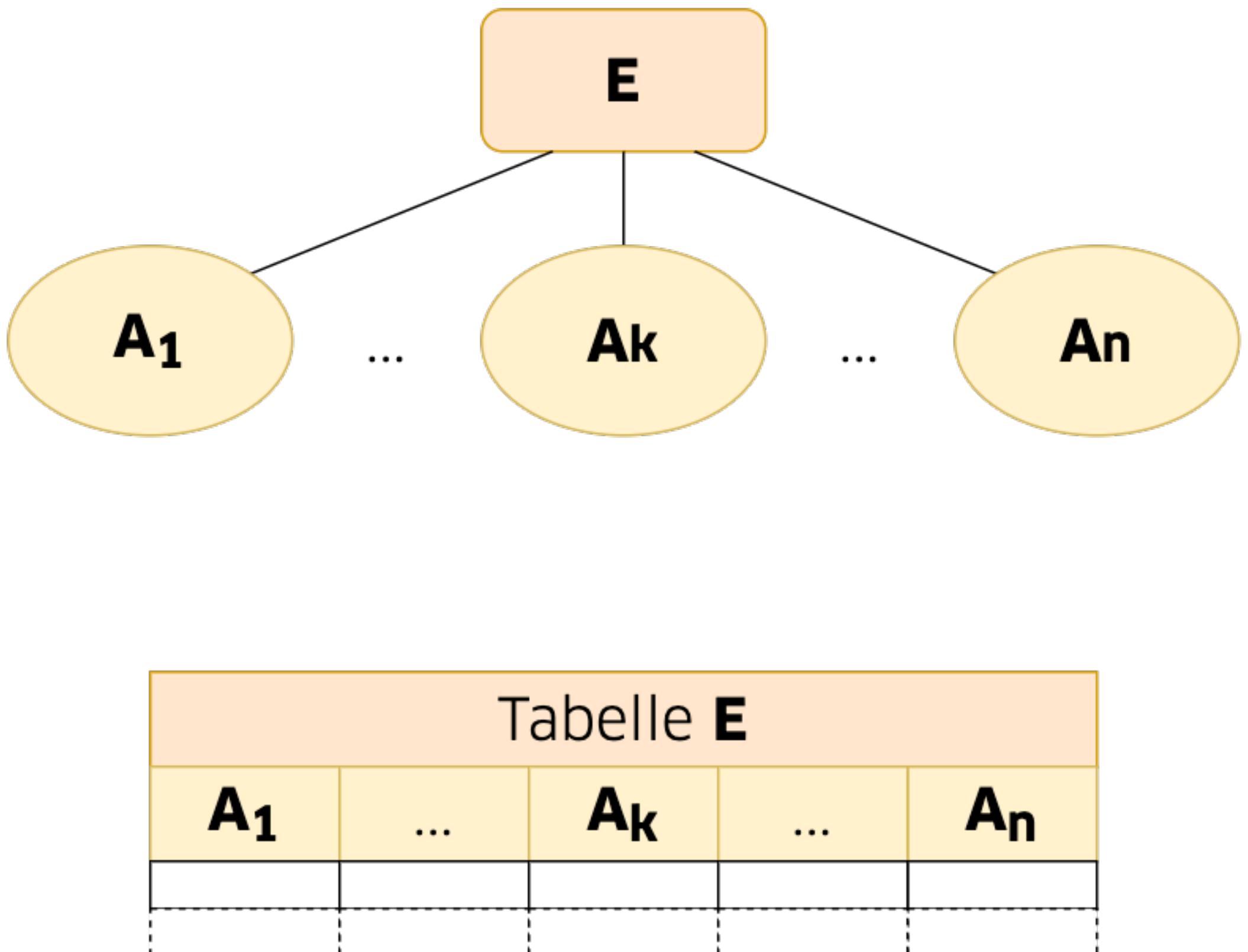
Überführung der Elemente im ER-Diagramm in Datenbankstrukturen

- Entitätstypen
- Attribute
 - zusammengesetzt
 - mehrwertig
- Relationen
 - 1:1
 - 1:N
 - N:M
- Hierarchien (später)

ER-Modell → Relationales Modell

Entitätstyp E: {[A₁:D₁,...,A_n:D_n]}

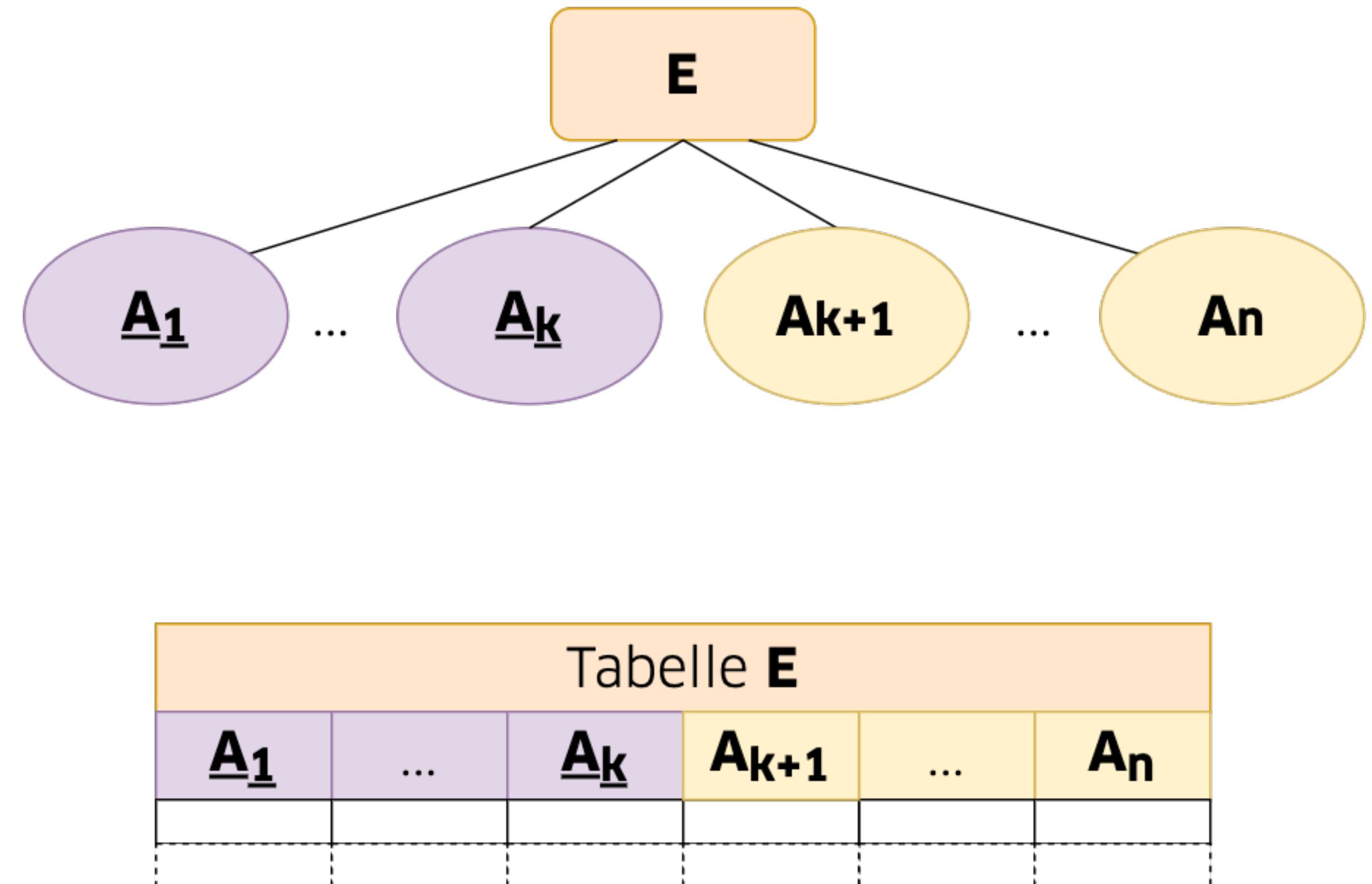
- E wird zur Tabelle E.
- Attribute A_k sind Spalten mit Datentyp D_k.
 - Alle Attribute sind *atomar*, also weder zusammengesetzt noch mehrwertig.
Letztere werden in eine andere Darstellung überführt (folgt).
 - Schlüsselattribute werden zu 'primary keys' (folgt).
- Die Reihenfolge der Zeilen und Spalten ist nicht relevant.
- Enthaltene Informationen werden nur durch Datenwerte ausgedrückt.



ER-Modell → Relationales Modell

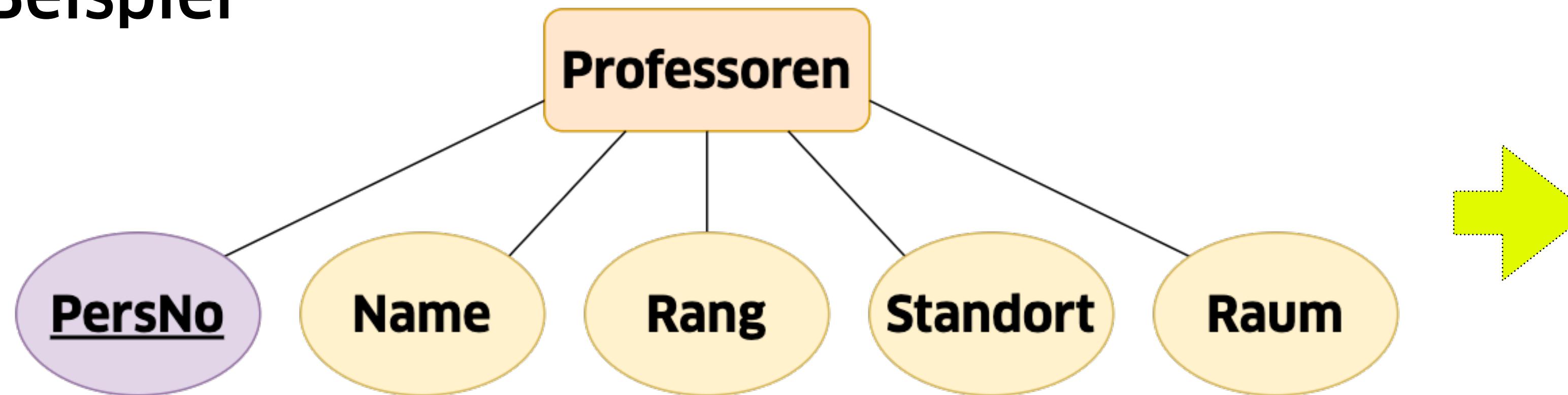
(Primär)Schlüsselattribute

- Zeile/Tuple/Entität in der Tabelle beschreibt ein eindeutiges Objekt!
 - Achtung: Technisch sind Tabellen ohne Schlüsselattribut und Einträge(Zeilen), die in allen Werten identisch sind, denkbar... aber
 - erst die Verwendung eindeutiger Schlüsselattribute garantiert die Identifikation unterschiedlicher Objekte, selbst bei gleichen Werten der restlichen Attribute!
- Schlüsselattribute werden 'primary keys'.



ER-Modell → Relationales Modell

Beispiel



Tabellenstruktur

Table:					Comment
Professoren					
Columns (5)	Keys (2)	Indices (1)	Foreign Keys		
PersNr int(11) - part of primary key					
Name varchar(30)					
Rang varchar(10)					
Standort varchar(20)					
Raum int(11)					

Tabellenstruktur
{ [PersNo:int,
Name:varchar,
Rang:varchar,
Standort:varchar,
Raum:varchar] }

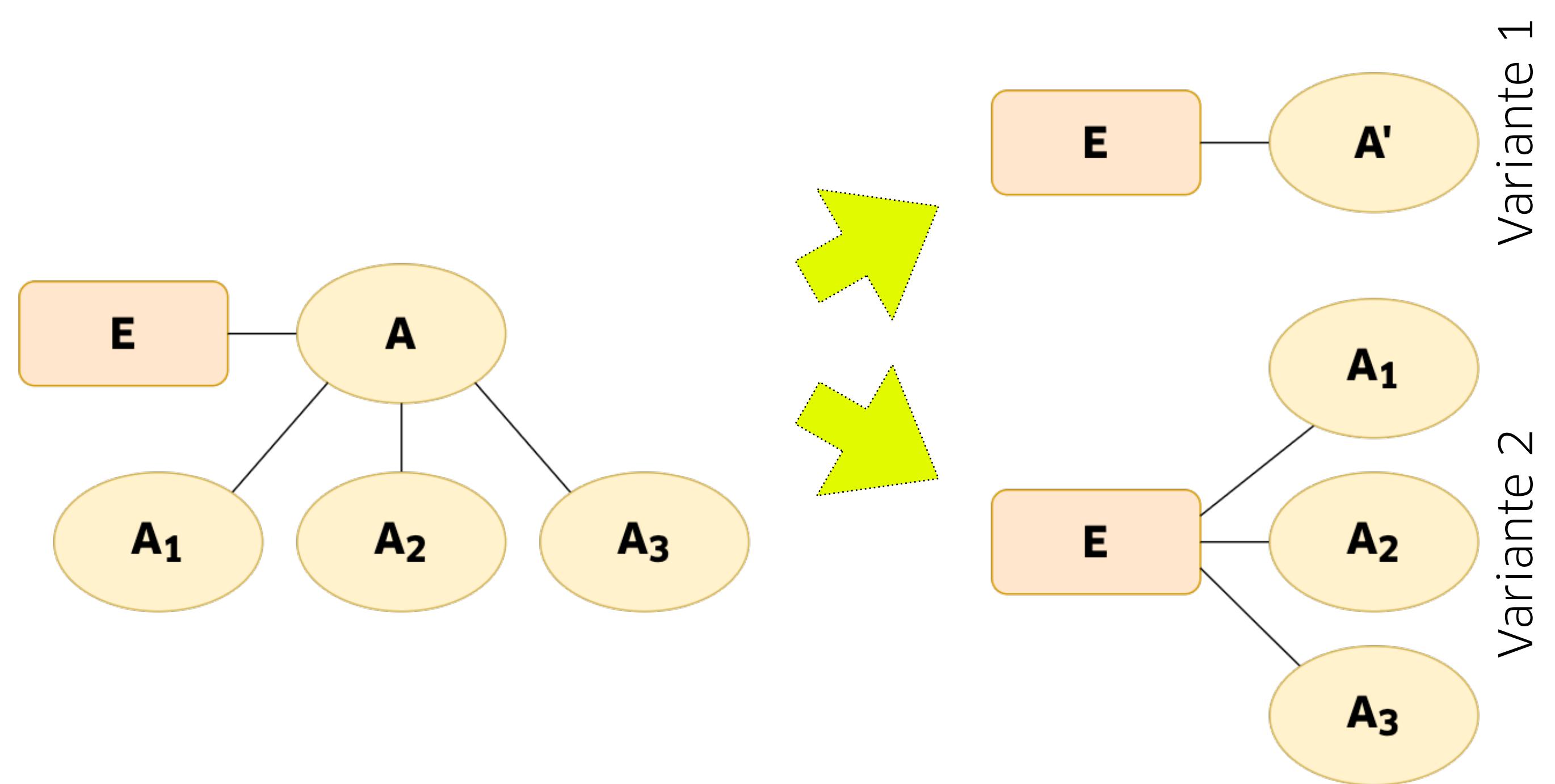
PersNr	Name	Rang	Standort	Raum
2125	Sokrates	C4	Jülich	226
2126	Russel	C4	Jülich	232
2127	Kopernikus	C3	Aachen	310
2133	Popper	C3	Aachen	52
2134	Augustinus	C3	Aachen	309
2136	Curie	C4	Jülich	36

Entitäten

ER-Modell → Relationales Modell

Zusammengesetzte Attribute

- Die Informationen von A liegen in den A_k , d.h. A enthält selber keine Informationen. Deswegen kann das Modell in eine der zwei Möglichkeiten mit nur *atomaren* Attributen überführt werden:
 - Zusammenfassung der A_k zu einem neuen gemeinsamen Attribut A' .
 - 'Umhängen' der Attribute A_k direkt an den Entitätstyp.



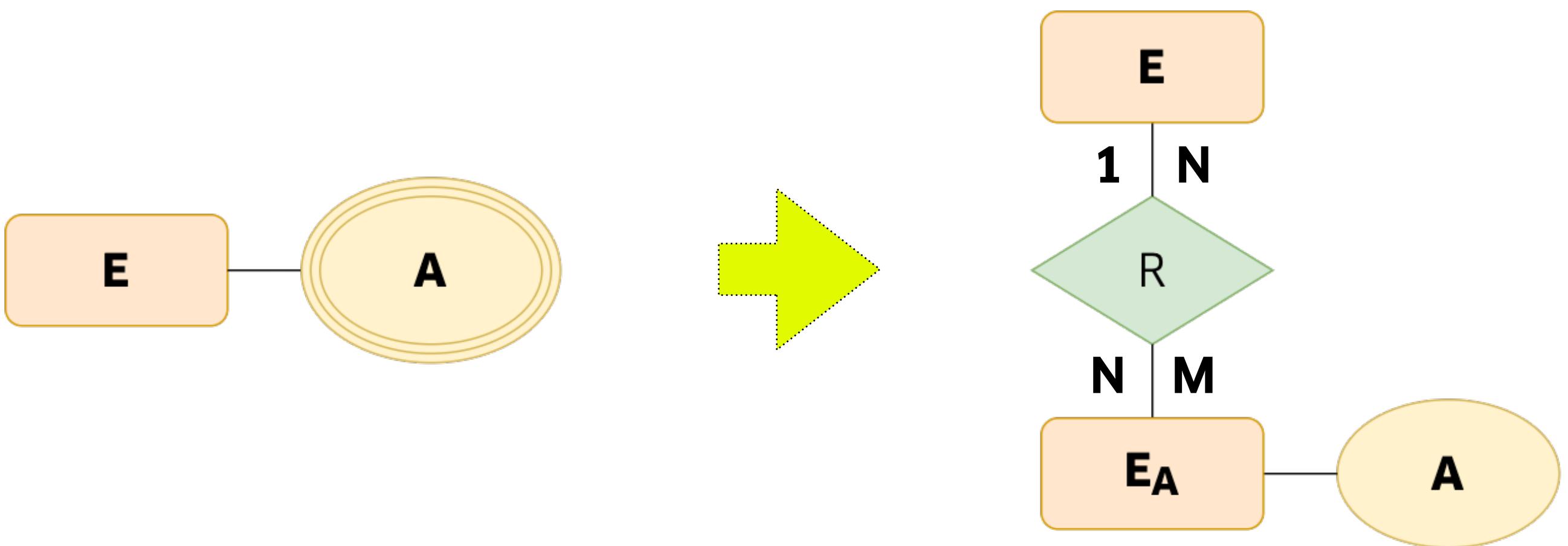
Q&A

- Pro/Con der Varianten?

ER-Modell → Relationales Modell

Mehrwertige Attribute

- Die verschiedenen Ausprägungen von A werden üblicherweise in eine 1:N-Relation mit einem weiteren Entitätstyp E_A abgebildet. Eine allgemeinere Rolle von E_A und eine N:M-Relation ist auch denkbar (siehe Beispiel).
- Schlüsselattribute von E können so auch zu Schlüsselattributen von E_A werden. Das hängt von der genauen Umsetzung der Relation ab (folgt).



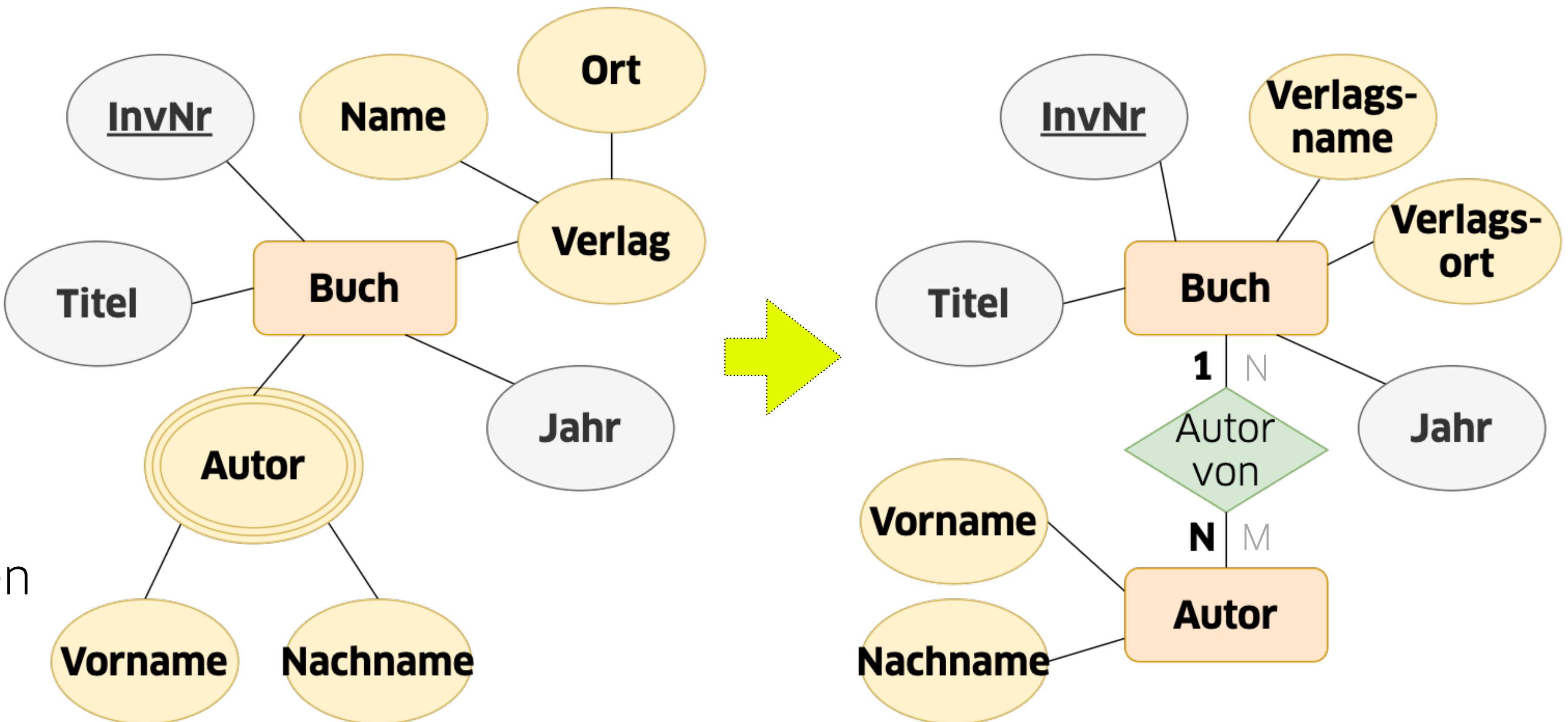
Q&A

- Pro/Con?

ER-Modell → Relationales Modell

Beispiel

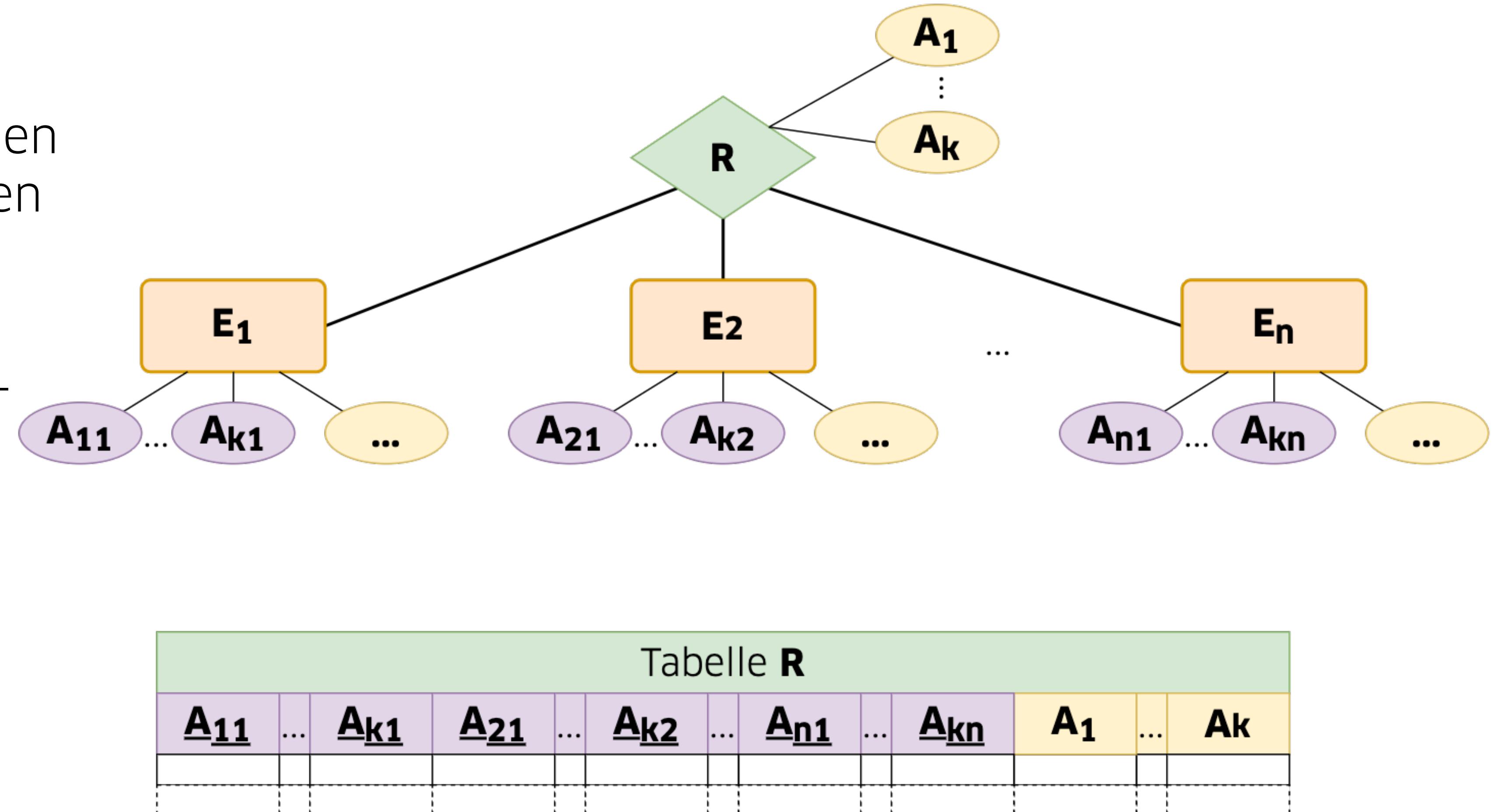
Hier ist auch eine allgemeinere Rolle von 'Autor', also eine N:M-Relation möglich, denn ein Autor kann viele Bücher schreiben. Das hätte in der Modellierung dann auch schon auffallen sollen...



ER-Modell → Relationales Modell

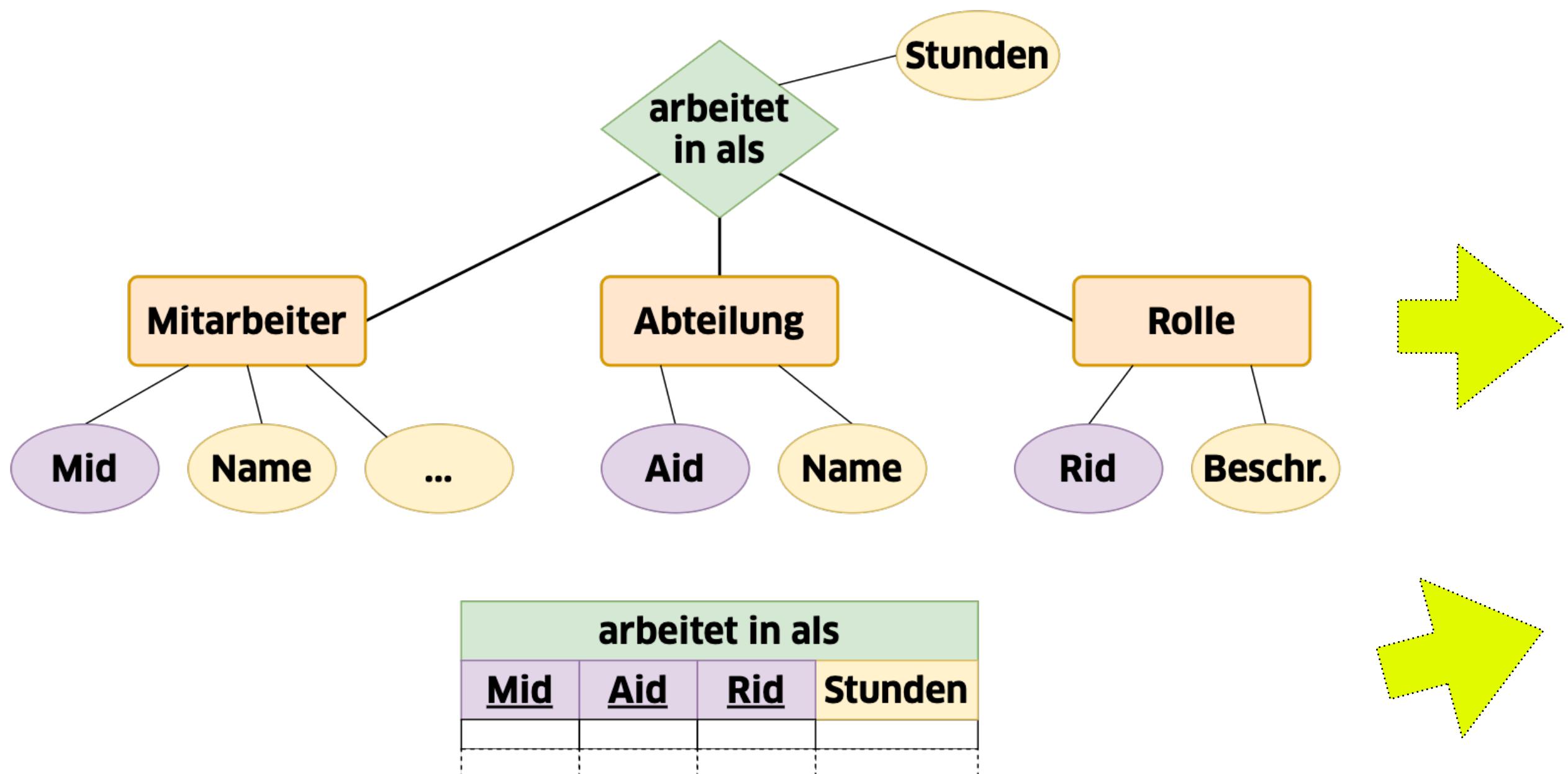
N:M-Relationen

- N:M-Relationen werden *immer* in einer eigenen Tabelle R realisiert.
- Diese Tabelle R enthält *alle* Schlüsselattribute der E_k , die vereint Schlüsselattribute von R sind ('Fremdschlüssel'),
- sowie die Attribute von R.



ER-Modell → Relationales Modell

Beispiel 3-stellige N:M-Relation



The relational schema consists of four tables:

- mitarbeiter**: Stores employee information. The primary key is id.
- abteilung**: Stores department information. The primary key is id.
- rolle**: Stores role information. The primary key is id.
- arbeitet_in_als**: The junction table for the many-to-many relationship. It has foreign keys to the primary keys of mitarbeiter, abteilung, and rolle.

Sample data for each table:

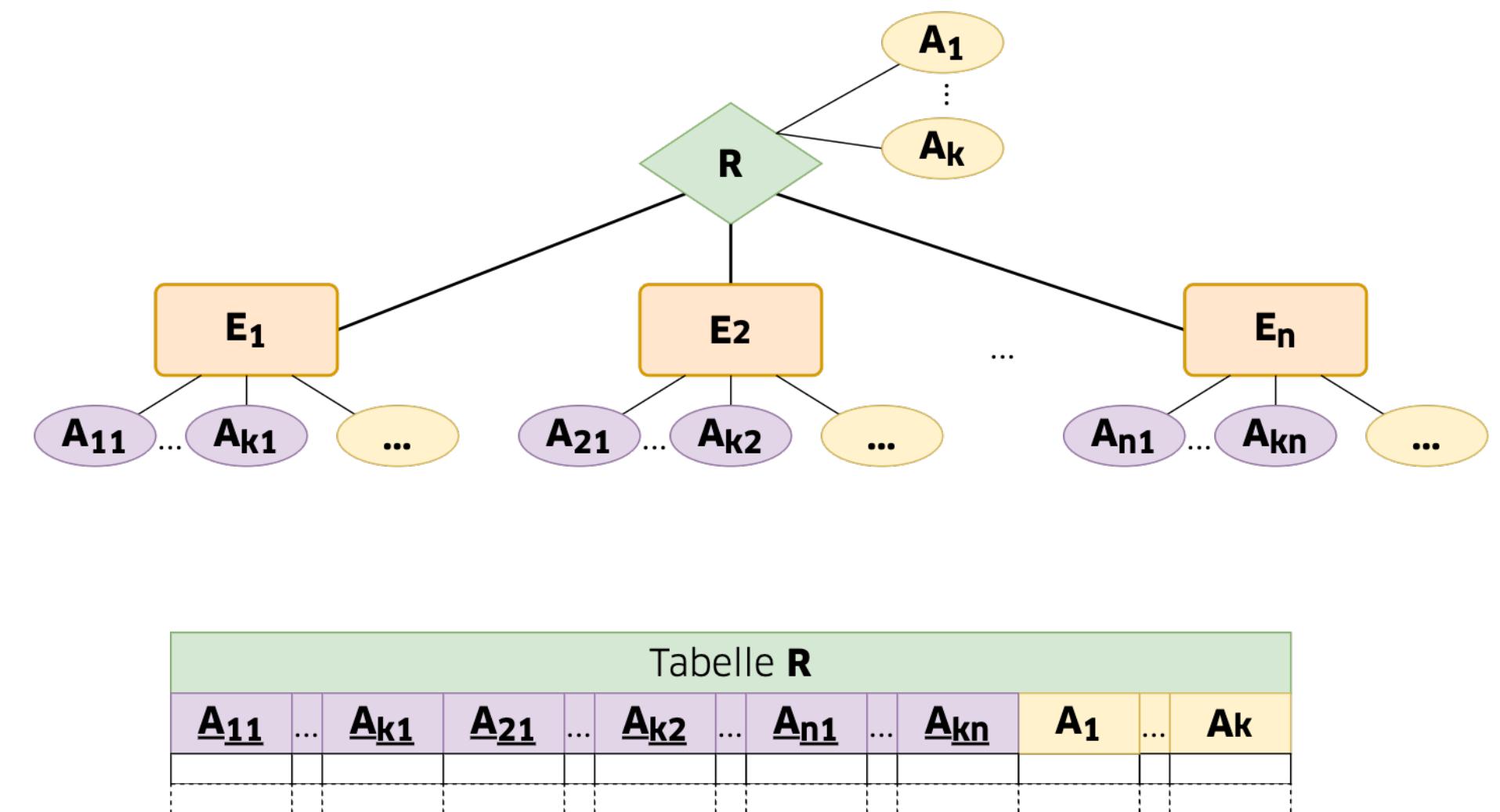
mitarbeiter	abteilung	rolle	arbeitet_in_als
id: 1, name: Mia, jahresgehalt: 110000.00	id: 1, name: Vorstand	id: 3, beschreibung: Leitung	mitarbeiter_id: 1, abteilung_id: 1, rolle_id: 3, Wochenstunden: 10.00
id: 2, name: Ben, jahresgehalt: 90000.00	id: 2, name: HR/Buchhaltung	id: 4, beschreibung: Mitarbeiter	mitarbeiter_id: 1, abteilung_id: 1, rolle_id: 4, Wochenstunden: 5.00
id: 3, name: Emma, jahresgehalt: 70000.00	id: 3, name: Vertrieb	id: 7, beschreibung: Student	mitarbeiter_id: 1, abteilung_id: 2, rolle_id: 3, Wochenstunden: 10.00
id: 4, name: Paul, jahresgehalt: 5000.00	id: 4, name: Marketing	id: 8, beschreibung: Auszubildender	mitarbeiter_id: 1, abteilung_id: 2, rolle_id: 4, Wochenstunden: 15.00
id: 5, name: Hannah, jahresgehalt: 5000.00	id: 5, name: Einkauf	id: 9, beschreibung: Schüler	mitarbeiter_id: 2, abteilung_id: 1, rolle_id: 4, Wochenstunden: 5.00

'Mia' (id=1) arbeitet im Vorstand (id=1) in der Leitung (id=3).

ER-Modell → Relationales Modell

Anmerkungen zu N:M-Relationen

- 1:1- und 1:N-Relationen können auch über eine eigene Tabelle realisiert werden, aber
- Relationen so abzubilden verursacht Kosten!
- Die Kombination der Schlüsselattribute ist natürlicherweise ein eindeutiger Schlüssel für R. Es ist aber auch möglich, ein künstliches Schlüsselattribut ('id') zu verwenden.



Q&A

- Pro/Con einer Relationen-Tabelle?
- Pro/Con eines künstlichen Schlüsselattributs?

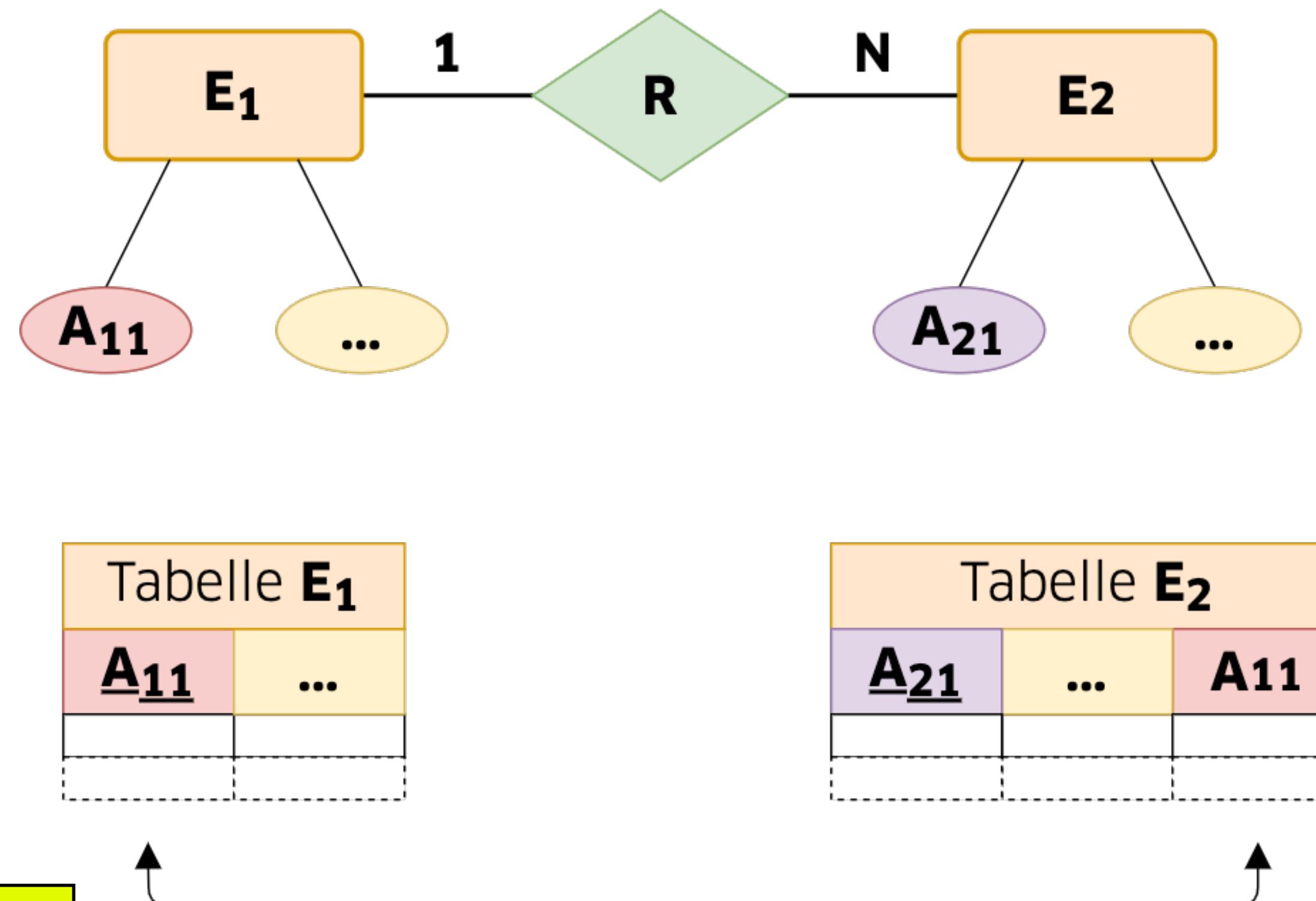
ER-Modell → Relationales Modell

1:N-Relationen

- Da wir eine partielle Funktion $f_2:E_2 \rightarrow E_1$ haben (Abb. auf 1), kann das Schlüsselattribut als sogenannter 'Fremdschlüssel' in E_2 als weiteres Attribut eingebettet werden.
- Dort ist es *kein* Schlüssel von E_2 , es referenziert nur die assoziierte Entität in E_1 .

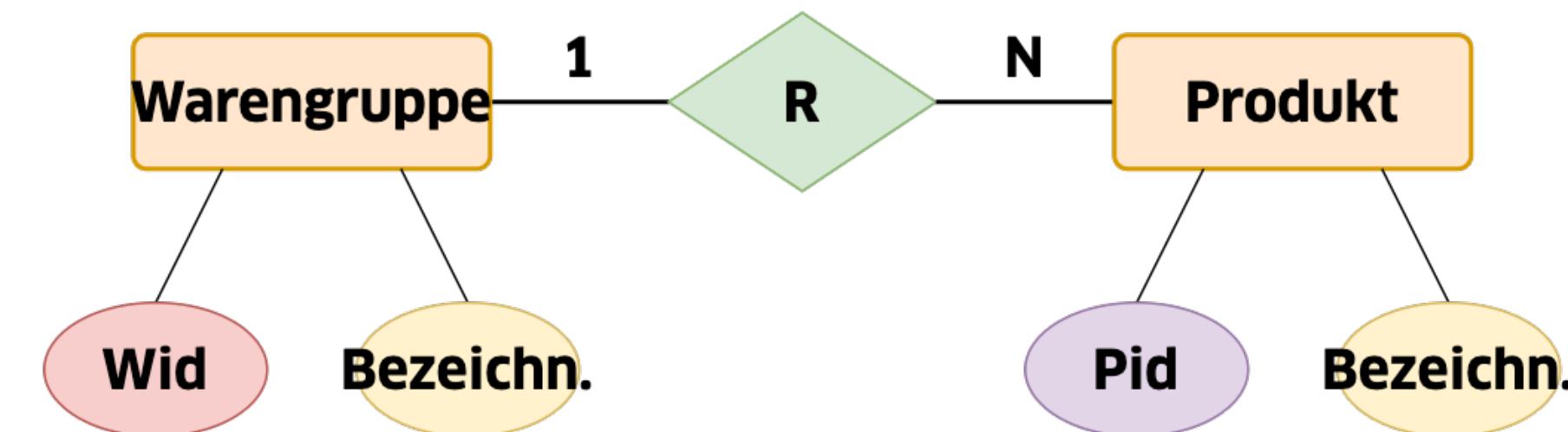


Es funktioniert nur mit Fremdschlüssel in E_2 !
Umgekehrt müsste man 'viele' Fremdschlüssel in E_1 ablegen... Widerspruch zur Atomarität!



ER-Modell → Relationales Modell

Beispiel 1:N-Relation



Warenguppe	
<u>Wid</u>	...

Produkt		
<u>Pid</u>	...	<u>Wid</u>

<u>id</u>	<u>bezeichnung</u>
1	TK
2	Obst, Gemüse
3	Wurst, Aufschnitt
4	Milchprodukte
5	Kaltgetränke

warengruppe

<u>id</u>	<u>bezeichnung</u>	<u>stueckpreis</u>	<u>warengruppe_id</u>
3	Spätzlepizza	2.27	1
4	Fischstäbchen	1.99	1
5	Nudelpfanne	3.29	1
6	Möhren	0.39	2
7	Zwiebeln	0.29	2

produkt

ER-Modell → Relationales Modell

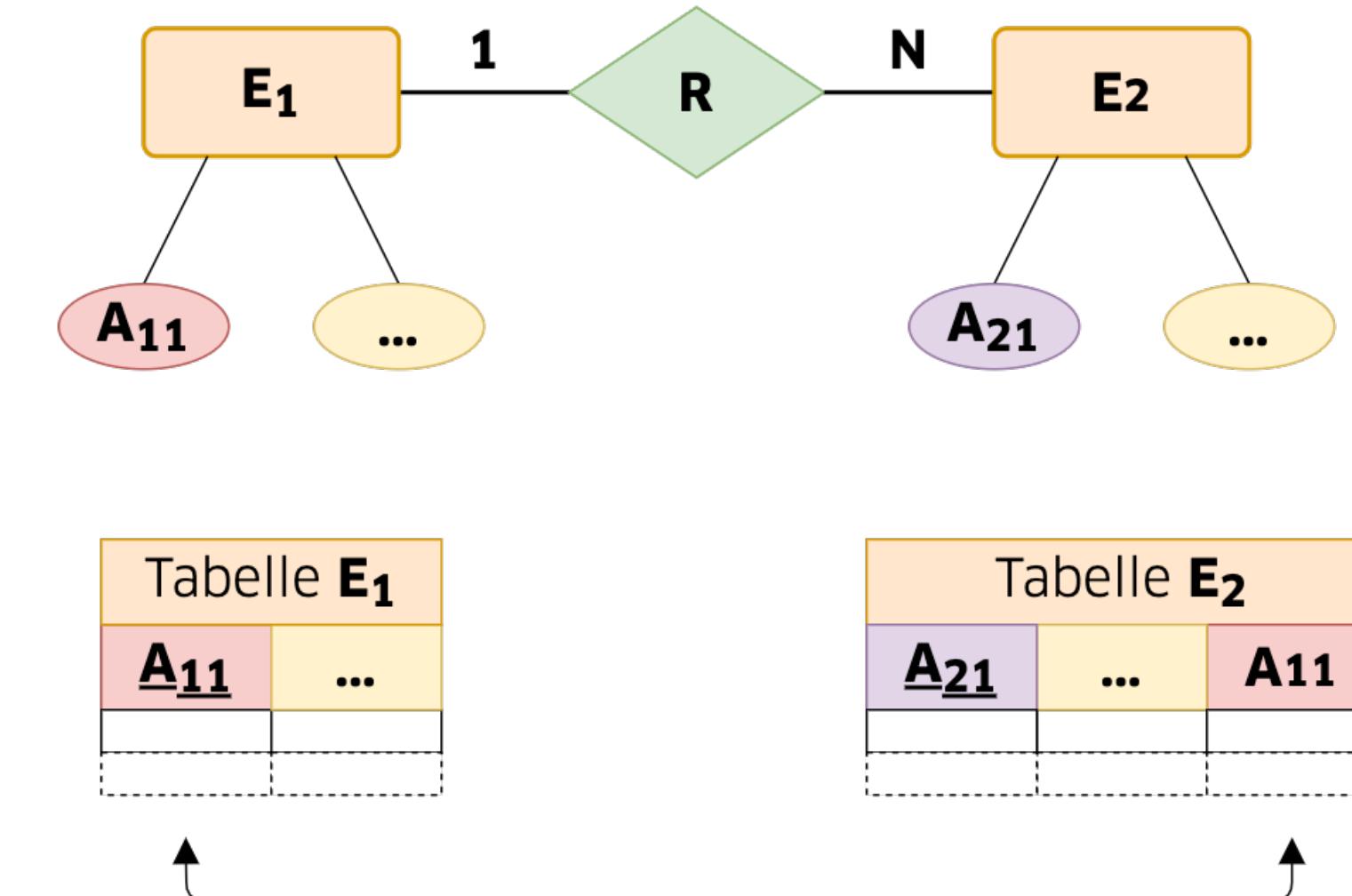
Anmerkungen zu 1:N-Relationen

- In dem Beispiel ist auf beiden Seiten jeweils nur ein Schlüsselattribut gezeigt.

Gibt es mehrere Schlüsselattribute in E_1 , so müssen sie offenbar *alle* in E_2 auftauchen, denn nur gemeinsam referenzieren sie die assozierte Entität in E_1 .

- Mögliche Attribute der Relation R können ebenfalls in E_2 eingebettet werden. Hier ist aber drauf zu achten, insbesondere bei mehreren 1:N-Relationen, ob E_2 immer noch *genau einen* Aspekt modelliert...
- Manchmal reicht eine kleine Änderung in den Anforderungen und aus einer 1:N-Relation R wird eine N:M-Relation. Bei einer eigenen Tabelle für R ist das kein Problem, ansonsten (wie hier) ändert sich das Schema.

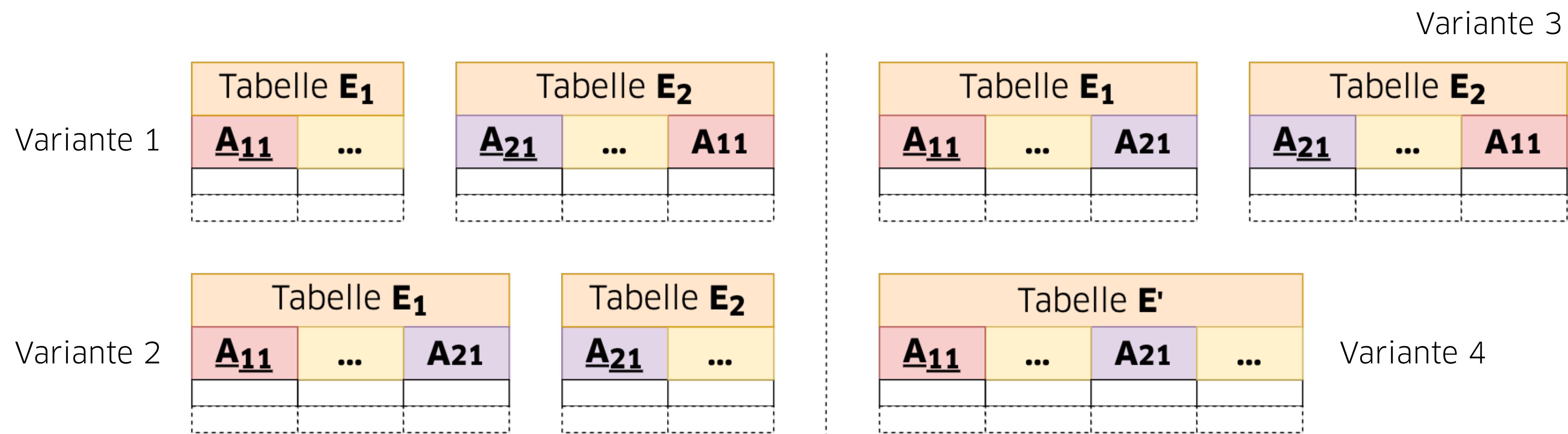
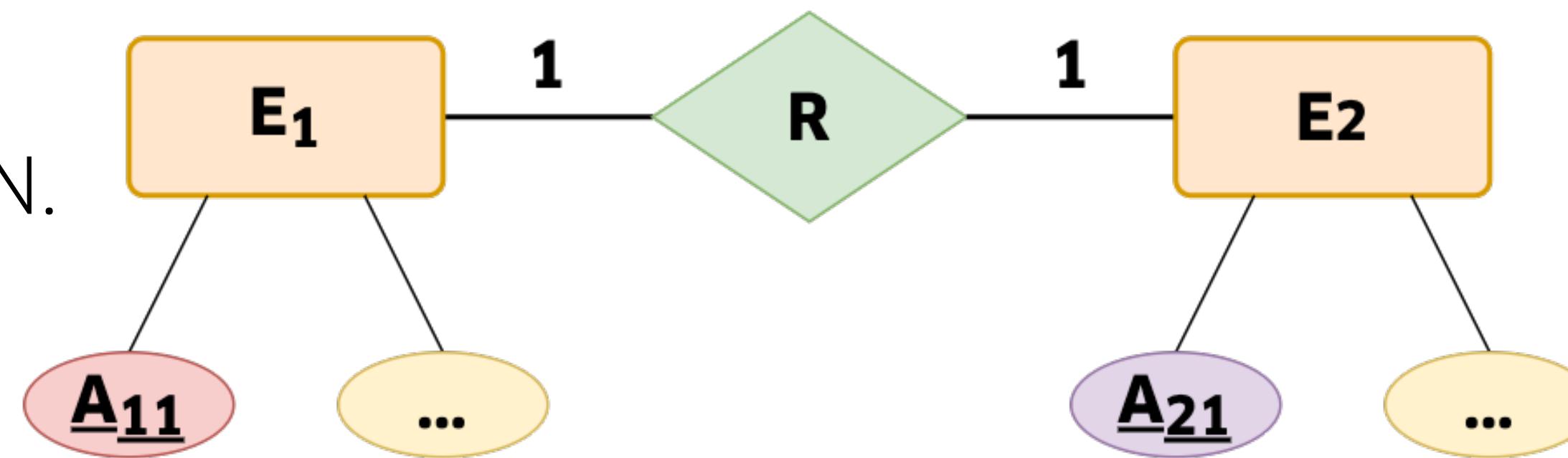
Stabilität der Anforderungen!



ER-Modell → Relationales Modell

1:1-Relationen

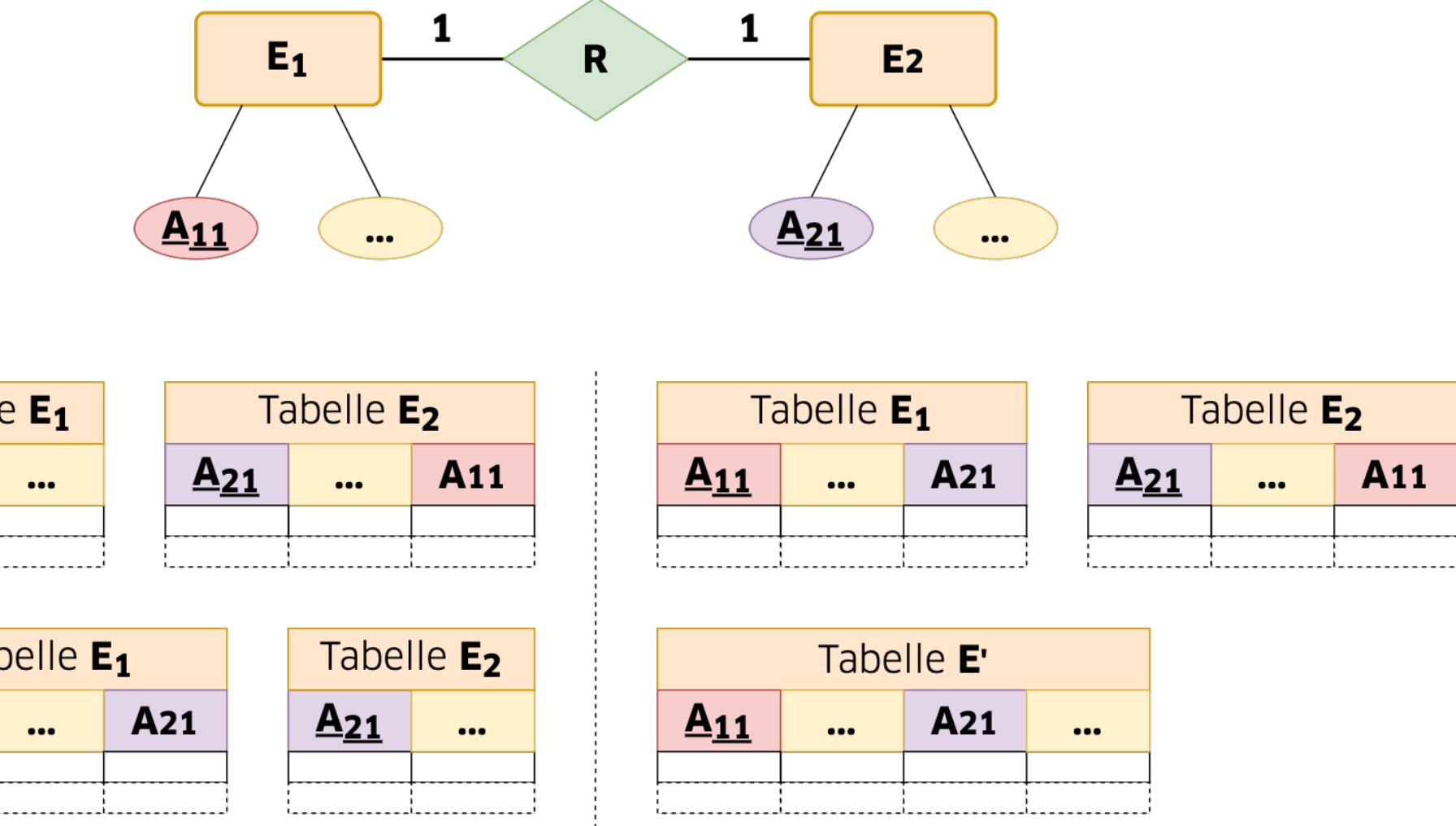
- Variante 1-3: Idee wie 1:N.
- Variante 4: Tabelle E'.
- Variante 5: Tabelle R.



ER-Modell → Relationales Modell

Anmerkungen zu 1:1-Relationen

- Welche der Möglichkeiten am geeignetsten ist, ist am Ende von den Anforderungen und der Verwendung der Relation in der Praxis abhängig:
 - Daten aus E_1 und E_2 werden häufig gemeinsam benötigt und man sucht ein e_2 zu einem e_1 - oder umgekehrt.
 - Auch wenn eine gemeinsame Tabelle E' die Trennung der modellierten Aspekte möglicherweise aufhebt, kann diese Lösung sehr effizient sein.
 - Auch wenn eine eigene Relationetabelle R die Datenbankoperationen komplexer macht, ist diese Lösung stabil gegenüber Änderungen zu einer 1:N- oder N:M-Rel.
 - ... Kein triviales Problem!



Zwischenstand

Erinnerung Vorgehen [...]

- Überführung in eine 'gute' Datenbank... dazu benötigen wir:
 - Implementationsentwurf, z.B. mit Tabellen und Relationen
→ Relationales Modell ✓
 - Mathematisches Modell der Datenbankoperationen, z.B. zur Anfrageoptimierung
→ Relationale Algebra (folgt)

Zusammenfassung

- Sie sind jetzt in der Lage, ein Problem zu modellieren und das resultierende ER-Diagramm (ohne Spezialisierungen) in einen Datenbankentwurf zu überführen!



Das ist doch schonmal etwas 👍

Motivation

Warum ein formaler (mathematischer) Ansatz?

- In den matse_mhist Produkten befinden sich Pizzen und wir fragen, ob welche weniger als 2.30€ kosten.
- Sind nebenstehende SQL-Befehle äquivalent?
Alle liefern

■ pizza	■ price
Spinatpizza	2.29
- Kann z.B. das DBMS entscheiden, ob es eine Anfrage (select) 'umformen' kann? Z.B. weil es dann effizienter oder mit weniger Zwischendaten arbeiten kann?
- Dafür brauchen wir ein formales Gerüst → Relationale Algebra!

■ id	■ bezeichnung	■ stueckpreis
1	Spinat	1.99
2	Vier Käse Pizza	2.39
3	Spinatpizza	2.29
4	Fischstäbchen	1.99

```

SELECT P.bezeichnung as 'pizza',P.stueckpreis as 'price'
FROM produkt P
WHERE P.bezeichnung like '%pizza%' and P.stueckpreis<2.30;

SELECT Q.*
FROM (
  SELECT P.bezeichnung as 'pizza',P.stueckpreis as 'price'
  FROM produkt P WHERE P.bezeichnung like '%pizza%'
) Q WHERE Q.price<2.30;

SELECT Q.*
FROM (
  SELECT P.bezeichnung as 'pizza',P.stueckpreis as 'price'
  FROM produkt P WHERE P.stueckpreis<2.30
) Q WHERE Q.pizza like '%pizza%';

```

Das Ergebnis eines 'Subselects' (Q) verhält sich wie eine eigene Tabelle.

Definitionen

Begriffe der relationalen Algebra

- Im mathematischen Kontext der Relationalen Algebra verstehen wir, im Einklang mit Unit 0x03, unter einer Relation R primär eine konkrete *Tupelmenge* und weniger die Beziehung zwischen Entitätstypen:
 - Ein Tupel ist eine geordnete Menge von Attributwerten, wobei D den Wertebereich eines Attributs $a:T$ mit Datentyp T beschreibt (E.F.Codd nutzt später ebenfalls Attribute statt der Ordnung des Tupels).
 - Eine Relation $R \subseteq D_1 \times \dots \times D_n$ ist eine Menge von Tupeln $t = [a_1, \dots, a_n] \in R$, $a_k \in D_k$.
- Eine Tabelle ist wiederum eine *visuelle Repräsentation* einer Relation und *eine Zeile* in einer Tabelle repräsentiert ein Tuple. Steht die Relation für einen Entitättyp, so sind die Tupel konkrete Ausprägungen bzw. Entitäten.

Definitionen

Begriffe der relationalen Algebra

- Die nachfolgend diskutierten Operationen der relationalen Algebra, etwa *Selektion*, *Projektion* und *Join*, sind mathematisch Mengenoperationen auf Relationen, die als Ergebnis ebenfalls Relationen liefern. Daher bezeichnet man die Relationenalgebra als abgeschlossen.
- Achtung Praxis...
 - SQL-Kommandos sind so gesehen Umsetzungen der Operationen auf Tabellen.
 - Je nach Kommando/Operation im DBMS können *doppelte Zeilen* entstehen – das Ergebnis ist mathematisch gesehen keine Menge mehr. Praktisch gesehen können Duplikate entfernt bzw. die Operationen direkt geeignet ausgeführt werden.
 - Die mathematische Behandlung des speziellen Attributwertes `NULL`, z.B. in einer dreiwertigen Logik, lassen wir hier aussen vor. Es ist aber wichtig, sich bei jeder Operation klar zu machen, wie diese mit `NULL`-Werten umgeht!

Definitionen

Begriffe der relationalen Algebra

- Ist man statt an einer Relation R , also der konkreten Tupelmenge, nur am Aufbau der Relation, d.h. den Attributen und Datentypen, interessiert, helfen diese Funktionen:
 - ♦ $\text{ident}(R) = \{a_i\}_i$,
 - ♦ $\text{schema}(R) = \{[a_1:T_1, \dots, a_n:T_n]\}$.
- Ein Schlüsselkandidat K mit $K \subseteq \text{ident}(R)$, ist eine Menge von Attributen, deren Werte jeweils alle Tupel der Relation eindeutig identifizieren ('identifier').
 - So ist implizit garantiert, dass es keine zwei gleichen Tupel in der Relation gibt.
 - Man kann auch sagen, dass aus der Kenntnis eines Schlüsselkandidaten, genauer den entsprechenden Werten, die anderen Attributwerte des Tupels in der Relation folgen. Diese Sichtweise werden wir im Kontext von 'funktionalen Abhängigkeiten' und 'Normalformen' vertiefen.

Definitionen

Begriffe der relationalen Algebra

- Grundsätzlich kann es mehrere Schlüsselkandidaten geben, aus denen dann *der Schlüssel* ('primary key') ausgewählt wird. Im Sprechgebrauch wird oftmals der konkrete Wert eines Schlüssels, der 'identifier', gemeint.
 - Kurz: Kennt man den primary key, kennt man die Entität.
- Falls geboten, kann man den Entitätstyp $E = \langle R, K \rangle$ als Kombination von Relation R und Schlüssel K angeben. So haben wir es in Unit 0x03 vorbereitet und das entspricht auch dem, was man in DBMS bei der Definition einer Tabelle angibt.
- Achtung Praxis...
 - Wenn ein Schlüsselkandidat aus mehreren Attributen besteht, so ist diese Situation im konkreten DBMS hinsichtlich Effizienz und Speicherbedarf zu prüfen. Ggf. kann die Einführung eines neuen künstlichen Schlüssels, bestehend aus nur *einem* Attribut, Sinn machen.

Operationen der relationalen Algebra

Mengenoperationen

- Um Mengenoperationen auf Relationen durchzuführen, müssen diese kompatibel sein. Das bedeutet, Attributanzahl und Wertebereiche müssen übereinstimmen – auch *Vereinigungsverträglichkeit* oder *Typkompatibilität* genannt.
- Für zwei Relationen S und T könnte man auch etwas unscharf fordern, dass $\text{schema}(S) = \text{schema}(T)$, gilt, ohne dass die Attribute formal gleich heißen müssen. Allerdings wäre eine verträgliche Bedeutung (Semantik) schon sinnvoll...
- Es existiert eine minimale Menge von (Grund)Operationen, die mindestens notwendig ist, um alle Ausdrücke der relationalen Algebra bilden zu können. Alle anderen Operationen lassen sich dadurch nachbilden.

Operationen der relationalen Algebra

Grundoperationen

- Vereinigung
- Differenz
- Kartesisches Produkt
- Selektion
- Projektion
- Umbenennung

Wir betrachten im Folgenden
ausgewählte Operationen mit
Bezug zu den entsprechenden
SQL-Kommandos.



Keine Grundoperationen

- Schnittmenge
- Symmetrische Differenz
- Join
- Equi-Join
- Natural Join
- Semi Join
- Outer Join
- Division

Operationen der relationalen Algebra

Klassische Mengenoperationen

- Seien S und T zwei kompatible Relationen. Dann sind definiert

$$\star \quad S \cup T = \{ r \mid r \in S \vee r \in T \}, \quad \text{SQL: union}$$

- $S - T = S \setminus T = \{ r \mid r \in S \wedge r \notin T \}$ SQL: minus ...

$$\star \ S \cap T = \{ r \mid r \in S \wedge r \in T \} \quad \text{SQL: intersect ...}$$

- Leider ist es so, dass nicht alle DBMS alle SQL-Befehle unterstützen. MySQL etwa hat keine Implementierung von `minus` und `intersect` und man muss diese anders umsetzen.



Soviel zum Thema 'Standard'.

Operationen der relationalen Algebra

Beispiele Klassische Mengenoperationen

$S \subseteq \text{produkt}$, $T \subseteq \text{produkt}$

- $S \cup T$:

```
✓ ┌─ SELECT P.*  
   ├─ FROM produkt P WHERE P.stueckpreis<0.5  
   └─ UNION  
      ┌─ SELECT P.*  
      ├─ FROM produkt P WHERE P.stueckpreis>5.99;
```

id	bezeichnung	stueckpreis
7	Zwiebeln	0.29
6	Möhren	0.39
43	GameStar	6.50

- aber auch
 $S \cup T$:

```
✓ ┌─ SELECT P.einheit  
   ├─ FROM produkt P WHERE P.stueckpreis<0.5  
   └─ UNION ALL  
      ┌─ SELECT P.einheit  
      ├─ FROM produkt P WHERE P.stueckpreis>5.99;
```

einheit
KG
KG
ST
1x
nx

Operationen der relationalen Algebra

Kartesisches Produkt

- Für zwei Relationen S und T mit $\text{ident}(S) \cap \text{ident}(T) = \emptyset$ ist das kartesische Produkt (auch Mengen- oder Kreuzprodukt) $S \times T$ definiert durch
 - ♦
$$S \times T = \bigcup_{(s_1, \dots, s_n) \in S} \left[\bigcup_{(t_1, \dots, t_k) \in T} \{(s_1, \dots, s_n, t_1, \dots, t_k)\} \right]$$
- Achtung: Im üblichen kartesischen Produkt entstehen Paare (s, t) mit $s \in S$ und $t \in T$. Hier hingegen entstehen $|S \times T| = |S| \cdot |T|$ Tupel, bestehend aus allen Attributen einer Entität $s \in S$ verbunden mit den Attributen einer Entität $t \in T$.
- Weiter: Im Fall $\text{ident}(S) \cap \text{ident}(T) \neq \emptyset$ würde obige Definition 'doppelte' Attribute erzeugen, was mathematisch wiederum ein Problem ist, aber in der Praxis durch 'Umbenennung' der Attribute gelöst werden kann.
- Merkregel: Jedes Element von S mit jedem von T.

Operationen der relationalen Algebra

Beispiel Kartesisches Produkt

$S \subseteq \text{abteilung}$, $T \subseteq \text{standort}$, $S \times T$:

Umbenannt
zu $S.id$, $T.id$

SELECT S.id, S.name FROM abteilung S;	
id	name
1	Vorstand
2	HR/Buchhaltung
3	Vertrieb
4	Marketing
5	Einkauf
6	F&E
7	AR

SELECT T.id, T.ort FROM standort T;	
id	ort
1	Aachen
2	Jülich
3	Köln
4	Berlin

Jedes Element von S
mit jedem von T

SELECT S.id, S.name, T.id, T.ort FROM abteilung S, standort T;			
S.id	name	T.id	ort
1	Vorstand	1	Aachen
1	Vorstand	2	Jülich
1	Vorstand	3	Köln
1	Vorstand	4	Berlin
2	HR/Buchhaltung	1	Aachen
2	HR/Buchhaltung	2	Jülich
2	HR/Buchhaltung	3	Köln
2	HR/Buchhaltung	4	Berlin
3	Vertrieb	1	Aachen
3	Vertrieb	2	Jülich
3	Vertrieb	3	Köln
3	Vertrieb	4	Berlin

Operationen der relationalen Algebra

Selektion

- Für eine Relation S und eine logische Bedingung Θ ist die Selektion definiert durch
 - ♦ $\sigma_\Theta(S) = \{ s \in S \mid s \text{ erfüllt Bedingung } \Theta \}$
- Selektionsbedingungen sind häufig Vergleichsoperationen ($=, \neq, \leq, <, >, \geq$) auf den Attributen und logische Verknüpfungen (\wedge, \vee, \neg). Auf eine formale Definition verzichten wir hier.
- Die Selektion wirkt wie ein Filter/Auswahlkriterium auf der Relation S , wo sie auf jedes Element angewandt wird (siehe Definition).
- Es ist leicht zu sehen, dass der SQL-Befehl `select` mit einer Bedingung Θ genau die Operation σ_Θ abbildet und die Frage nach der günstige Pizza aus der Motivation darauf hinausläuft, ob $\sigma_{\Theta_1 \wedge \Theta_2}(S) = \sigma_{\Theta_1}(\sigma_{\Theta_2}(S)) = \sigma_{\Theta_2}(\sigma_{\Theta_1}(S))$ gilt.

Operationen der relationalen Algebra

Beispiele Selektion

- $\sigma_{\text{bez. like 'getränke'}}(\text{warengruppe})$

```
SELECT * FROM warengruppe
WHERE bezeichnung like '%getränke';
```

Output matse_mhist.warengruppe

id	bezeichnung
5	Kaltgetränke
8	Heissgetränke

- $\sigma_{\text{gehalt}>50000}(\text{mitarbeiter})$

```
SELECT * FROM mitarbeiter
WHERE jahresgehalt>50000;
```

Output matse_mhist.mitarbeiter

id	name	jahresgehalt
1	Mia	110000.00
2	Ben	90000.00
3	Emma	70000.00
12	Leon	70000.00
14	Luis	70000.00
16	Lukas	70000.00
20	Leonie	70000.00

Operationen der relationalen Algebra

Beispiele Selektion

- $\sigma_{(\text{bez. like 'pizza'}) \wedge (\text{preis} < 2.30)}(\text{produkt})$
- $\sigma_{\text{preis} < 2.30}(\sigma_{\text{bez. like 'pizza'}}(\text{produkt}))$
- $\sigma_{\text{bez. like 'pizza'}}(\sigma_{\text{preis} < 2.30}(\text{produkt}))$

```

SELECT P.bezeichnung as 'pizza', P.stueckpreis as 'price'
FROM produkt P
WHERE P.bezeichnung like '%pizza%' and P.stueckpreis<2.30;

SELECT Q.*
FROM (
    SELECT P.bezeichnung as 'pizza', P.stueckpreis as 'price'
    FROM produkt P WHERE P.bezeichnung like '%pizza%'
) Q WHERE Q.price<2.30;

SELECT Q.*
FROM (
    SELECT P.bezeichnung as 'pizza', P.stueckpreis as 'price'
    FROM produkt P WHERE P.stueckpreis<2.30
) Q WHERE Q.pizza like '%pizza%';

```

▪ pizza	▪ price ▾
Spinatpizza	2.29

Wir lassen hier die Benennung der Attribute und die Projektion auf bezeichnung und stueckpreis zunächst aussen vor.

Operationen der relationalen Algebra

Projektion

- Die Projektion $\Pi_{a_{i1} \dots a_{ik}}$ wählt aus einer Relation S mit $\text{ident}(S)=\{a_1, \dots, a_n\}$ die Attribute a_{i1} bis a_{ik} aus:
 - ♦ $\Pi_{a_{i1} \dots a_{ik}}(S) = \{ (a_{i1}, \dots, a_{ik}) \mid a \in S \}$
- Achtung: Die mathematische Menge eliminiert Duplikate, die Ergebnismenge in SQL aber nicht.
- Der SQL-Befehl select, verbunden mit der Angabe der Attribute einer Relation, realisiert die obige Projektion. Das Kommando ist bekanntlich mächtiger.

Operationen der relationalen Algebra

Beispiele Projektion

- $\Pi_{\text{id}, \text{name}, \text{jahresgehalt}}(\text{mitarbeiter})$
- $\Pi_{\text{stueckpreis}, \text{umsatzsteuer}}(\text{produkt})$

```
SELECT id, name, jahresgehalt  
FROM mitarbeiter;
```

Output matse_mhist.mitarbeiter

id	name	jahresgehalt
1	Mia	110000.00
2	Ben	90000.00
3	Emma	70000.00
4	Paul	5000.00

```
SELECT stueckpreis, umsatzsteuer  
FROM produkt order by stueckpreis;
```

Output matse_mhist.produkt

stueckpreis	umsatzsteuer
0.29	0.07
0.39	0.07
0.79	0.07
0.90	0.07
0.98	0.07
0.98	0.07
0.99	0.07

Operationen der relationalen Algebra

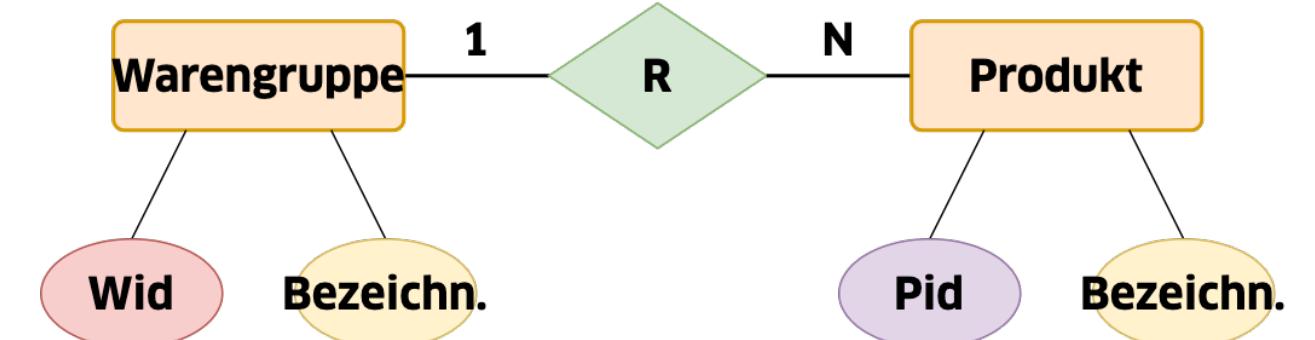
Theta-Verbund/-Join \bowtie_Θ

- Ganz zu Beginn haben wir in der Diskussion der Artikelsammlung die große Tabelle in mehrere Entitätstypen, Beziehungen und Fremdschlüsselbeziehungen aufgeteilt, um z.B. Redundanzen und Anomalien zu eliminieren. Diese so verteilten Daten wollen wir wieder zusammen führen und das leistet der sog. Theta-Verbund bzw. Theta-Join, oder kurz (und unpräzise) Join.
- Der Theta-Verbund $S \bowtie_\Theta T$ für zwei Relationen S und T und Selektionsbedingung Θ ist definiert durch:
 - ♦ $S \bowtie_\Theta T = \sigma_\Theta(S \times T)$
- Achtung: Hier entstehen theoretisch zunächst enorm große Datenmengen durch das kartesische Produkt, die dann durch die Selektion wieder reduziert werden. In der Praxis kann ein DBMS diese sehr häufigen Join-Operationen effizient durchführen.
- Achtung: Es gibt u.a. Inner, Outer, Cross, Self, Natural, Semi-, und Anti-Semi-Joins.

Operationen der relationalen Algebra

Beispiel Theta-Verbund

- Zur Motivation der Definition gucken wir zunächst auf die 1:N-Relation zwischen Warengruppe W und Produkt P, die über Fremdschlüssel realisiert ist.
- Das kartesische Produkt kombiniert zunächst jedes Produkt mit jeder Warengruppe ($\text{produkt} \times \text{warengruppe}$).
- Wenn wir genau die Zeilen selektieren, in denen $P.\text{warengruppe_id}$ mit $W.\text{id}$ übereinstimmt (Θ), haben wir zu jedem Produkt die jeweils assozierte Warengruppe → Voilà, die Definition $\sigma_{\Theta}(P \times W)$.



produkt

P.id	P.bezeichnung	warengruppe_id
4	Fischstäbchen	1
5	Nudelpfanne	1
6	Möhren	2

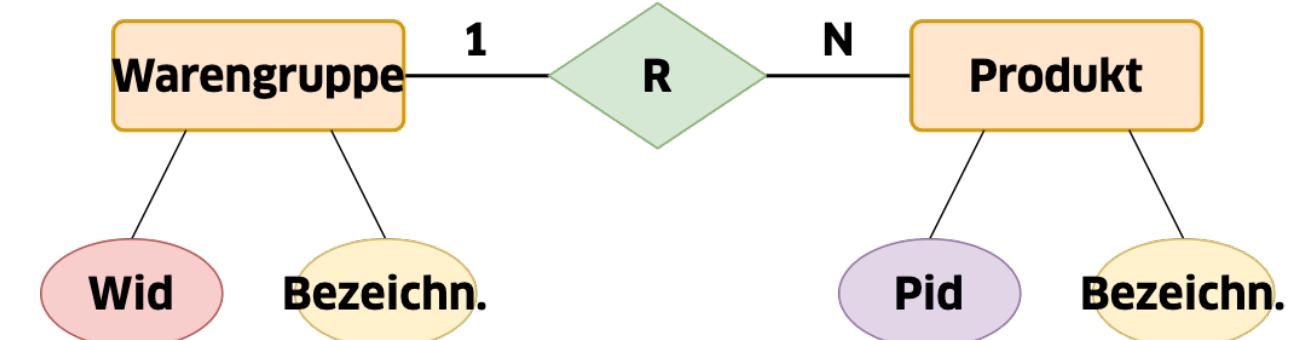
warengruppe

W.id	W.bezeichnung
1	TK
2	Obst, Gemüse
3	Wurst, Aufschnitt

produkt × warengruppe

P.id	P.bezeichnung	warengruppe_id	W.id	W.bezeichnung
9	5 Nudelpfanne	1	1	TK
0	5 Nudelpfanne	1	2	Obst, Gemüse
1	5 Nudelpfanne	1	3	Wurst, Aufschnitt
2	5 Nudelpfanne	1	4	Milchprodukte
3	5 Nudelpfanne	1	5	Kaltgetränke
4	5 Nudelpfanne	1	6	Grillgut
5	5 Nudelpfanne	1	7	Snacks, Süßwaren
6	5 Nudelpfanne	1	8	Heissgetränke
7	5 Nudelpfanne	1	9	Fertiggerichte
8	5 Nudelpfanne	1	10	Brot, Backwaren
9	5 Nudelpfanne	1	11	Zeitschriften
0	5 Nudelpfanne	1	12	Dienstleistung
1	6 Möhren	2	1	TK
2	6 Möhren	2	2	Obst, Gemüse

Operationen der relationalen Algebra



Beispiel Theta-Verbund

- 1:N-Relation
- P=produkt, W=warengruppe,
 $\Theta=P.\text{warengruppe_id}=W.\text{id}$
- $P \bowtie_{S.\text{warengruppe_id}=T.\text{id}} W = \sigma_\Theta(P \times W)$

```

SELECT P.id, P.bezeichnung, P.warengruppe_id, W.id, W.bezeichnung
FROM produkt P, warengruppe W
WHERE P.warengruppe_id=W.id;
  
```

produkt \times warengr.
 $\sigma_{S.\text{warengruppe_id}=T.\text{id}}$

P.id	P.bezeichnung	warengruppe_id	W.id	W.bezeichnung
1	Spinat	1	1	TK
2	Vier Käse Pizza	1	1	TK
3	Spinatpizza	1	1	TK
4	Fischstäbchen	1	1	TK
5	Nudelpfanne	1	1	TK
6	Möhren	2	2	Obst, Gemüse
7	Zwiebeln	2	2	Obst, Gemüse
8	Bananen	2	2	Obst, Gemüse
9	Knoblauch	2	2	Obst, Gemüse
10	Fleischwurst	3	3	Wurst, Aufschnitt
11	Frikadellen	3	3	Wurst, Aufschnitt



Dieser (Inner) Join ist ein zentrales Ergebnis!
So bekommen wir Daten aus 1:1-, 1:N- und N:M-Relationen wieder zusammen.

Operationen der relationalen Algebra

Beispiel Theta-Verbund

- N:M-Relation
- A=abteilung, R=sitzt_am, S=standort
- $A \bowtie_{A.id=R.abteilung_id} R \bowtie_{R.standort_id=S.id} S$

```
SELECT A.name, A.id, R.abteilung_id, R.standort_id, S.id, S.ort
FROM abteilung A, sitzt_am R, standort S
WHERE A.id=R.abteilung_id and R.standort_id=S.id;
```

name	A.id	abteilung_id	standort_id	S.id	ort
Vorstand	1		1	1	1 Aachen
HR/Buchhaltung	2	2		1	1 Aachen
Vertrieb	3	3		1	1 Aachen
F&E	6	6	1	1	Aachen
Vertrieb	3	3	2	2	Jülich
F&E	6	6	2	2	Jülich
Vertrieb	3	3	3	3	Köln
Einkauf	5	5	3	3	Köln
Marketing	4	4	3	3	Köln
Vertrieb	3	3	4	4	Berlin

Q&A

- Was passiert, wenn es keine assoziierten Entitäten gibt?

Operationen der relationalen Algebra

Beispiel Theta-Verbund

- N:1-Relation, Tabelle tier hat 3 Entitäten
- T=tier, M=mitarbeiter
- $T \bowtie_{T.\text{mitarbeiter_id}=M.\text{id}} M$
- Achtung: Der fehlende Eintrag (NULL) bei 'Rosa' führt zu einer kleineren Ergebnismenge, da der Verbund nur assoziierte Entitäten enthält.
- Möchte man *alle* Tiere im Join beachten, so führt das auf den Begriff des *Outer Joins*, hier konkret des *Left Outer Joins*. Dort werden etwaige NULL-Referenzen explizit mit berücksichtigt (praktische Schritte im SQL-Praktikum, Definitionen folgen).

name	mitarbeiter_id
Petra	4
Mini	18
Rosa	<null>

tier

```
SELECT T.name, T.mitarbeiter_id, M.id, M.name
FROM tier T, mitarbeiter M
WHERE T.mitarbeiter_id=M.id;
```

T.name	mitarbeiter_id	id	M.name
Petra	4	4	Paul
Mini	18	18	Max

Operationen der relationalen Algebra

Anmerkungen Theta-Verbund bzw. Join

- Die bisher gezeigten Joins sind sog. *Inner Joins* und SQL hat eigene (äquivalente und optimierte) Befehle dafür (`join`).
- Dies, und auch einen Teil der anderen *Joins*, besprechen wir schon einmal im SQL-Praktikum, damit man in der SQL-Praxis weiter kommt. Die jeweiligen formalen Definitionen folgen dann in Teil 2 der relationalen Algebra.

Zwischenstand Operationen der relationalen Algebra

Grundoperationen

- Vereinigung ✓
- Differenz (✓)
- Kartesisches Produkt ✓
- Selektion ✓
- Projektion ✓
- Umbenennung

Keine Grundoperationen

- Schnittmenge (✓)
- Symmetrische Differenz
- Inner Join ✓
- Equi-Join
- Natural Join
- Semi Join
- Outer Join
- Division