

Abschlussveranstaltung Übung Datenbanken

Marcus Riemer, mri

GROUP BY

Dozent	Veranstaltung
gh	ABWL
gh	ABWL
gh	ProWi
eh	Analysis 1
dh	Analysis 1
dh	Analysis 1

Tabelle *Prüfung*

```
SELECT veranstaltung
FROM pruefung
GROUP BY dozent
```



Mehrdeutig und damit ungültig

```
SELECT dozent, veranstaltung
FROM pruefung
GROUP BY dozent, veranstaltung
```

Sofern eine Gruppierung vorliegt, muss jeder Ausdruck im SELECT aggregiert oder gruppiert werden.

Gruppiert wird, wenn jeder Wert in den aufgeführten Spalten identisch ist. Diese Regel gilt unabhängig von der Spaltenanzahl.

Unterabfragen

pin	name
1	gh
2	eh
3	dh

Tabelle *Prüfer*

vid	name	pin
1	ABWL	1
2	ProWi	1
3	DB 1	3

Tabelle *Veranstaltung*

```
SELECT p.*  
FROM pruefer p  
WHERE p . pin IN  
(SELECT pin  
FROM veranstaltung v  
WHERE v . vid > 1)
```



```
SELECT p.*  
FROM pruefer p  
WHERE p . pin IN (1,3)
```

① **Keine Korrelation**

② **Auswertung der Unterabfrage**

```
pin  
_____  
1  
  
3  
_____
```

③ **Einsetzen in die Oberabfrage**

Unterabfragen

2

pin	name
1	gh
2	eh
3	dh

Tabelle *Prüfer*

vid	name	pin
1	ABWL	1
2	ProWi	1
3	DB 1	3

Tabelle *Veranstaltung*

```
SELECT p.*  
FROM pruefer p  
WHERE EXISTS  
(SELECT pin  
FROM veranstaltung v  
WHERE v.pin = p.pin)
```

3

```
SELECT pin  
FROM veranstaltung v  
WHERE v.pin = 1
```

```
SELECT pin  
FROM veranstaltung v  
WHERE v.pin = 2
```

```
SELECT pin  
FROM veranstaltung v  
WHERE v.pin = 3
```

1 **Korrelation**



2 **Teilweise Auswertung der OA**

3 **Zeilenweise einsetzen in UA**

4 **Einsetzen in die Oberabfrage**

JOIN

pin	name
1	gh
2	eh
3	dh

Tabelle *Prüfer*

vid	name	pin
1	ABWL	1
2	ProWi	1
3	DB 1	3

Tabelle *Veranstaltung*

```
SELECT *  
FROM pruefer p JOIN veranstaltung v
```

p.pin	p.name	v.vid	v.name	v.pin
1	gh	1	ABWL	1
1	gh	2	ProWi	1
1	gh	3	DB 1	3
2	eh	1	ABWL	1
2	eh	2	ProWi	1
2	eh	3	DB 1	3
3	dh	1	ABWL	1
3	dh	2	ProWi	1
3	dh	3	DB 1	3

INNER JOIN

pin	name
1	gh
2	eh
3	dh

Tabelle *Prüfer*

vid	name	pin
1	ABWL	1
2	ProWi	1
3	DB 1	3

Tabelle *Veranstaltung*

```
SELECT *  
FROM pruefer p JOIN veranstaltung v  
WHERE p.pin = v.pin
```

p.pin	p.name	v.vid	v.name	v.pin
1	gh	1	ABWL	1
1	gh	2	ProWi	1
1	gh	3	DB 1	3
2	eh	1	ABWL	1
2	eh	2	ProWi	1
2	eh	3	DB 1	3
3	dh	1	ABWL	1
3	dh	2	ProWi	1
3	dh	3	DB 1	3

OUTER JOIN

pin	name
1	gh
2	eh
3	dh

Tabelle *Prüfer*

vid	name	pin
1	ABWL	1
2	ProWi	1
3	DB 1	3

Tabelle *Veranstaltung*

```
SELECT *
FROM pruefer p
LEFT OUTER JOIN veranstaltung v USING (pin)
```

p.pin	p.name	v.vid	v.name	v.pin
1	gh	1	ABWL	1
1	gh	2	ProWi	1
1	gh	3	DB 1	3
2	eh	1	ABWL	1
2	eh	2	ProWi	1
2	eh	3	DB 1	3
3	dh	1	ABWL	1
3	dh	2	ProWi	1
3	dh	3	DB 1	3
2	eh	NULL	NULL	NULL

Zusammenfassung JOINS

- 1. Jede Zeile mit jeder anderen Zeile kombinieren (Kreuzprodukt)**
- 2. Innere Verknüpfung bilden**
- 3. Künstliche Zeilen für komplett entfallene Datensätze hinzufügen**

- 1. JOIN**
- 2. INNER JOIN**
- 3. OUTER JOIN**

Der Unterschied bei den verschiedenen JOINS ist der Zeitpunkt des "Ausstiegs".

- Jeder INNER JOIN bildet zunächst ein Kreuzprodukt
- Jeder OUTER JOIN bildet zunächst einen INNER JOIN

JOIN Schreibweisen

```
SELECT *  
FROM person p  
  INNER JOIN pruefung pr  
    ON (p.pin = pr.mit_pin)
```



```
SELECT *  
FROM person p, pruefung pr  
WHERE (p.pin = pr.mit_pin)
```

```
SELECT *  
FROM person  
  INNER JOIN pruefung USING (pin)
```



```
SELECT *  
FROM person p, pruefung pr  
WHERE (p.pin = pr.pin)
```

```
SELECT *  
FROM person  
  NATURAL JOIN veranstaltung  
  INNER JOIN veranstaltung USING (pin, name)
```



```
SELECT *  
FROM person p, veranstaltung v  
WHERE p.pin = v.pin  
  AND p.name = v.name
```

Person
pin INT
name VARCHAR(45)
gebdat DATE

Pruefung
pin INT
mit_pin INT
vid INT

Veranstaltung
vid INT
name VARCHAR(45)
pin INT

Reihenfolge bei JOINS

Tabelle *Krankenkasse*

kkid	name
1	A
2	B
3	C

Tabelle *Person*

pin	name	kkid
1	X	1
2	Y	1
3	Z	3
4	M	1
5	U	<i>NULL</i>

Tabelle *Student*

pin
1
2
3
5

```
SELECT *  
FROM krankenkasse k  
INNER JOIN person p USING (kkid)
```

k.kkid	k.name	p.pin	p.name	p.kkid
1	A	1	X	1
1	A	2	Y	1
1	A	4	M	1
3	C	3	Z	3

Reihenfolge bei JOINS

Tabelle *Krankenkasse*

kkid	name
1	A
2	B
3	C

Tabelle *Person*

pin	name	kkid
1	X	1
2	Y	1
3	Z	3
4	M	1
5	U	<i>NULL</i>

Tabelle *Student*

pin
1
2
3
5

```
SELECT *  
FROM krankenkasse k  
LEFT OUTER JOIN person p USING (kkid)
```

k.kkid	k.name	p.pin	p.name	p.kkid
1	A	1	X	1
1	A	2	Y	1
1	A	4	M	1
3	C	3	Z	3
2	B	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>

Reihenfolge bei JOINS

Tabelle *Krankenkasse*

kkid	name
1	A
2	B
3	C

Tabelle *Person*

pin	name	kkid
1	X	1
2	Y	1
3	Z	3
4	M	1
5	U	NULL

Tabelle *Student*

pin
1
2
3
5

```

SELECT *
FROM krankenkasse k
LEFT OUTER JOIN person p USING (kkid)
WHERE p.name LIKE "X%"
    
```

k.kkid	k.name	p.pin	p.name	p.kkid
1	A	1	X	1
1	A	2	Y	1
1	A	4	M	1
3	C	3	Z	3
2	B	NULL	NULL	NULL

Reihenfolge bei JOINS

Tabelle *Krankenkasse*

kkid	name
1	A
2	B
3	C

Tabelle *Person*

pin	name	kkid
1	X	1
2	Y	1
3	Z	3
4	M	1
5	U	<i>NULL</i>

Tabelle *Student*

pin
1
2
3
5

SELECT *

FROM krankenkasse k

LEFT OUTER JOIN person p **USING** (kkid)

INNER JOIN student s **USING** (pin)

k.kkid	k.name	p.pin	p.name	p.kkid	s.pin
1	A	1	X	1	1
1	A	2	Y	1	2
1	A	4	M	1	
3	C	3	Z	3	3
2	B	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>	

Reihenfolge bei JOINS

Tabelle *Krankenkasse*

kkid	name
1	A
2	B
3	C

Tabelle *Person*

pin	name	kkid
1	X	1
2	Y	1
3	Z	3
4	M	1
5	U	<i>NULL</i>

Tabelle *Student*

pin
1
2
3
5

```

SELECT *
FROM krankenkasse k
LEFT OUTER JOIN person p USING (kkid)
LEFT OUTER JOIN student s USING (pin)
    
```

k.kkid	k.name	p.pin	p.name	p.kkid	s.pin
1	A	1	X	1	1
1	A	2	Y	1	2
1	A	4	M	1	NULL
3	C	3	Z	3	3
2	B	NULL	NULL	NULL	NULL

Reihenfolge bei JOINS

Tabelle *Vorige_Folie*

k.kkid	k.name	p.pin	p.name	p.kkid	s.pin
1	A	1	X	1	1
1	A	2	Y	1	2
1	A	4	M	1	NULL
3	C	3	Z	3	3
2	B	NULL	NULL	NULL	NULL

```
SELECT  
  k.kkid AS „kkid“,  
  COUNT(p.pin) AS „anz_pers“,  
  COUNT(s.pin) AS „anz_stud“  
FROM Vorige_Folie  
GROUP BY k.kkid
```

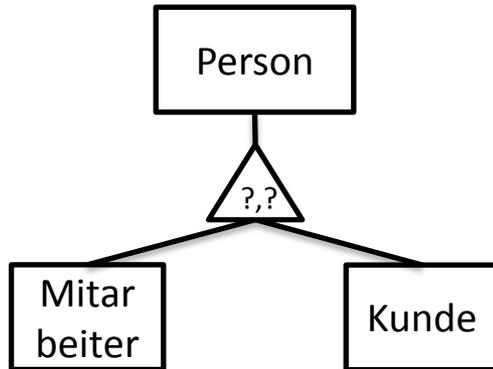


kkid	anz_pers	anz_stud
1	3	2
2	1	1
3	0	0

“Typische” Fehler in Queries

- Ausdrücke im SELECT die bei Verwendung von GROUP BY weder aggregiert noch gruppiert wurden.
- Unterabfragen, bei denen die Struktur der Unterabfrage nicht zur Operation passt.
 - Bei EXISTS: Beliebige Anzahl Zellen
 - Bei Vergleichen: Eine Zeile, eine Spalte
 - Bei skalaren Werten: Eine Zeile, eine Spalte
 - Bei IN: Beliebige viele Zeilen, eine Spalte

IS-A



- Partiell := $Mitarbeiter \cup Kunde \subseteq Person$
Total := $Mitarbeiter \cup Kunde = Person$
Disjunkt := $Mitarbeiter \cap Kunde = \emptyset$
-Disjunkt := $Mitarbeiter \cap Kunde \subset Person$

Partiell

Total

Disjunkt

Jede Person ist **entweder** Kunde **oder** Mitarbeiter (disjunkt).
Es **gibt** Personen, die weder Kunde noch Mitarbeiter sind (partiell).

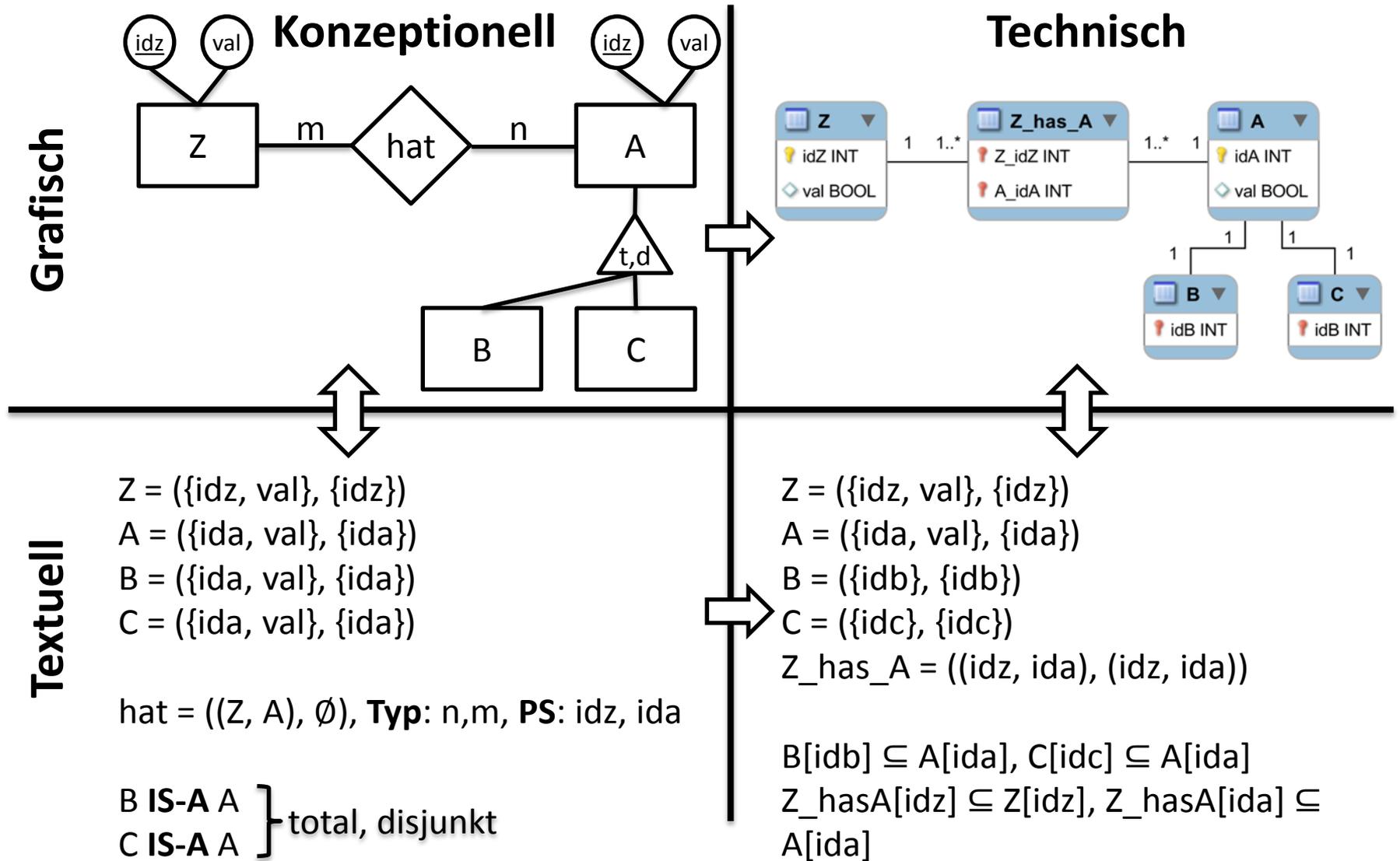
Jede Person ist **entweder** Kunde **oder** Mitarbeiter (disjunkt).
Es **gibt keine** Personen, die weder Kunde noch Mitarbeiter sind (total).

Nicht Disjunkt

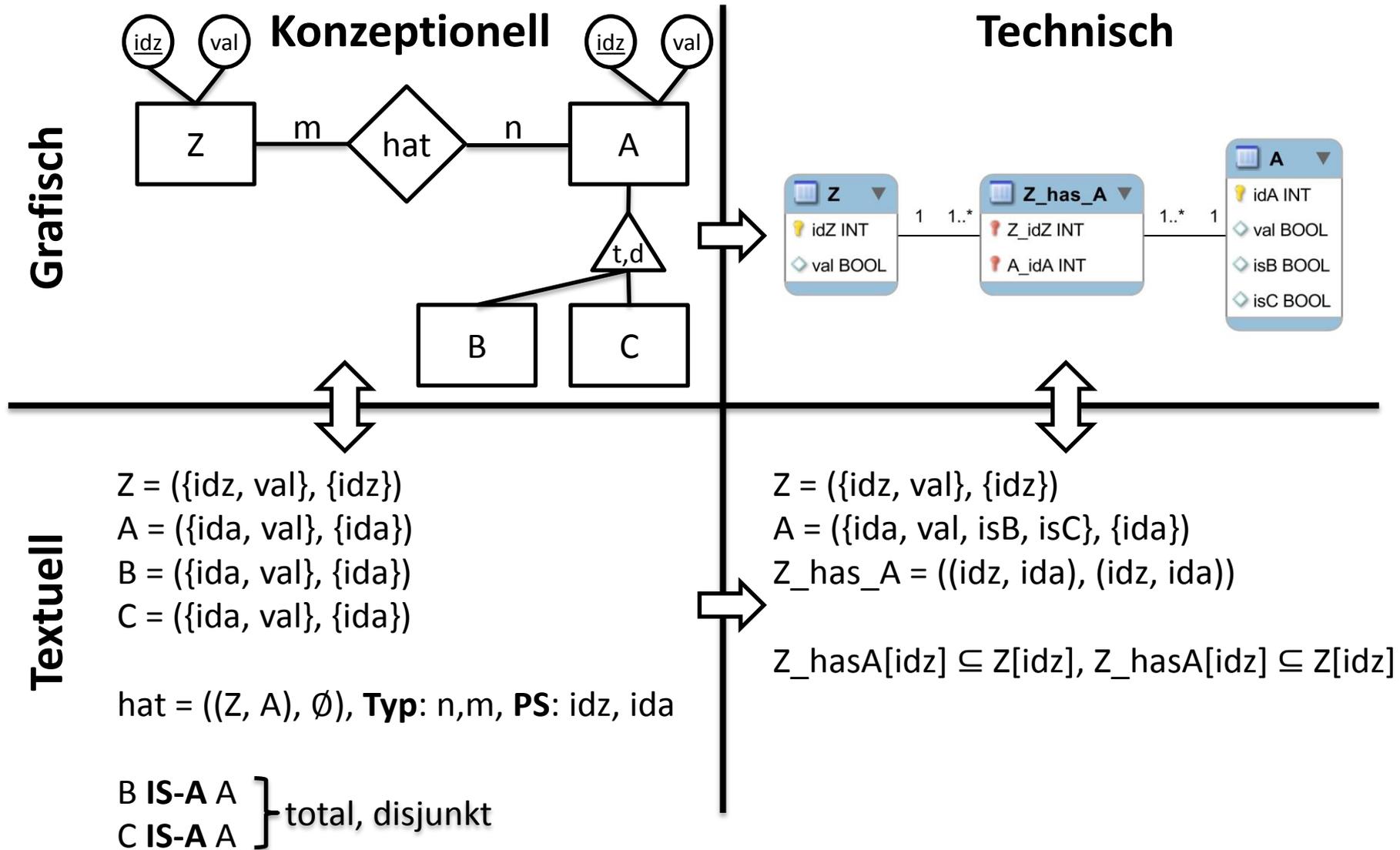
Eine Person **kann** Kunde **und** Mitarbeiter sein (nicht disjunkt).
Es **gibt** Personen, die weder Kunde noch Mitarbeiter sind (partiell).

Eine Person **kann** Kunde **und** Mitarbeiter sein (nicht disjunkt).
Es **gibt keine** Personen, die weder Kunde noch Mitarbeiter sind (total).

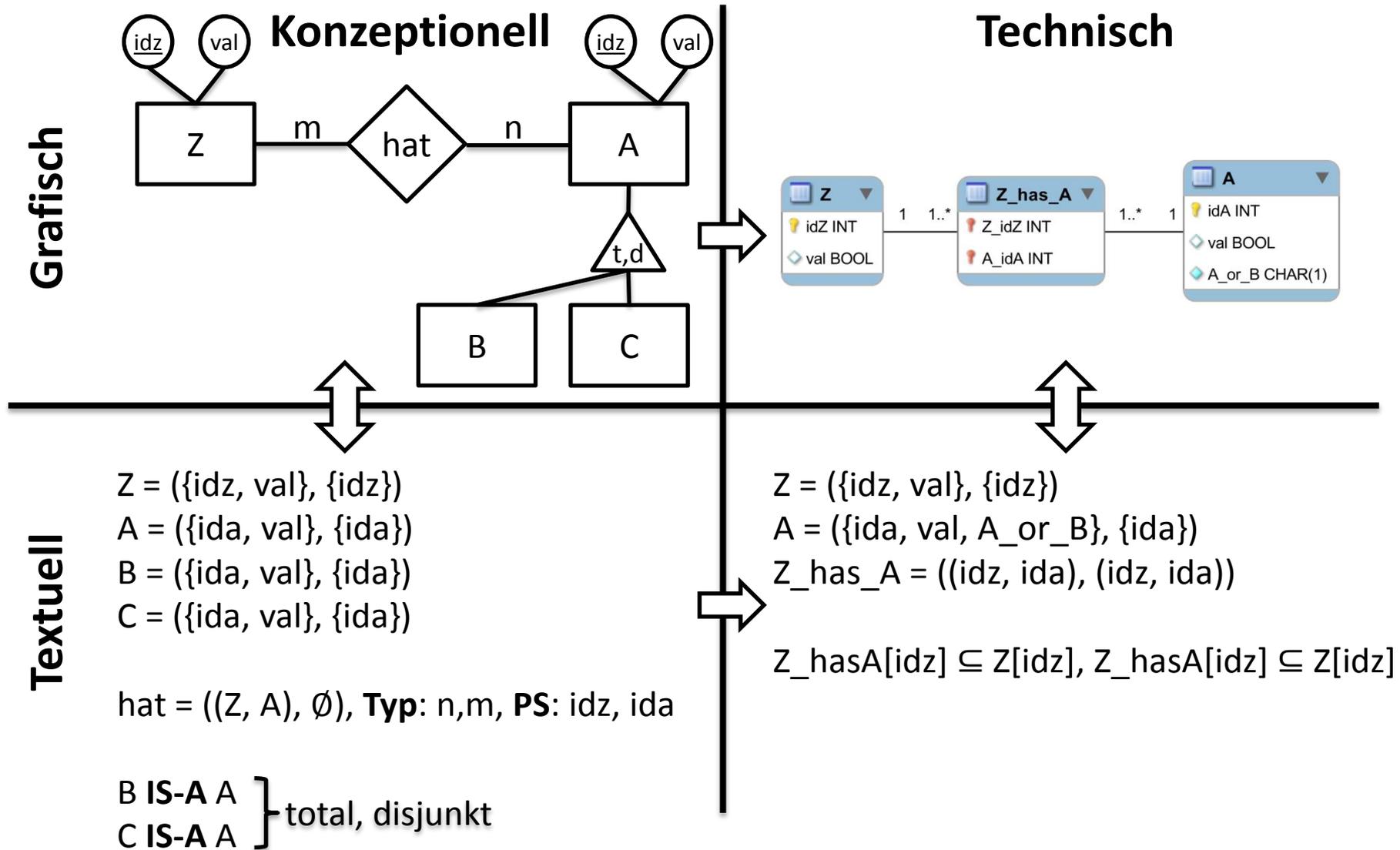
ER Modell, ER Diagramm, R Schema



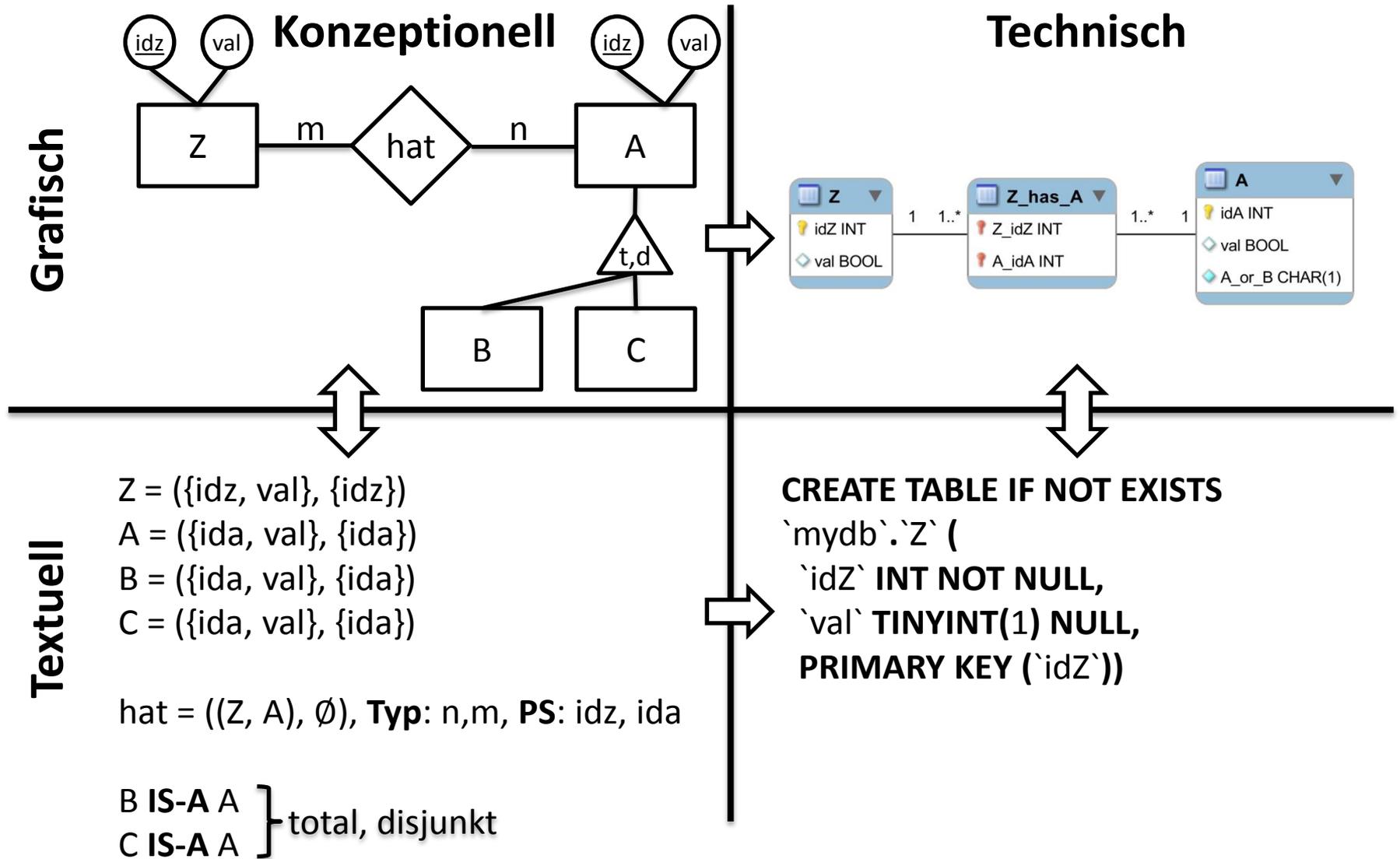
ER Modell, ER Diagramm, R Schema



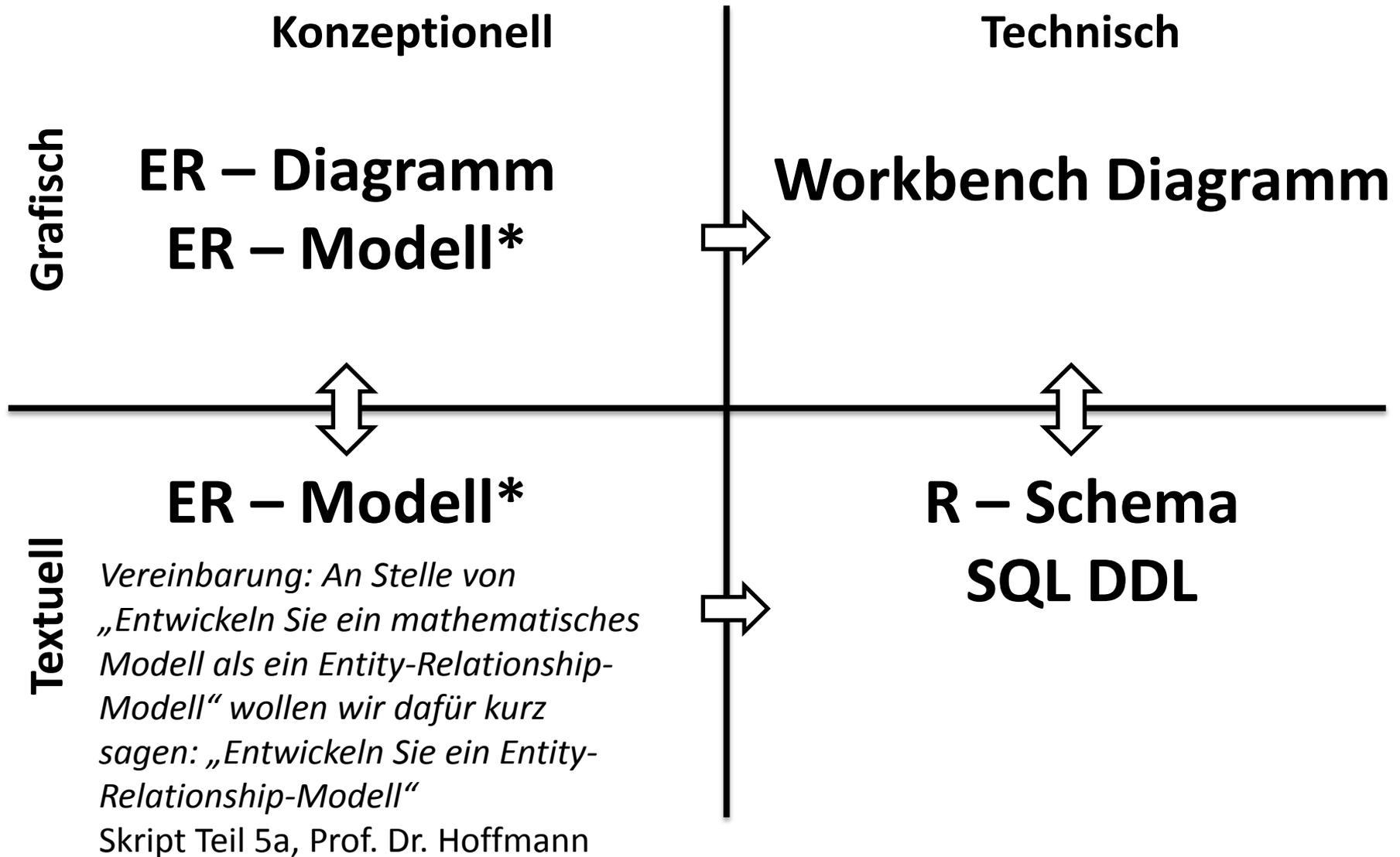
ER Modell, ER Diagramm, R Schema



ER Modell, ER Diagramm, R Schema



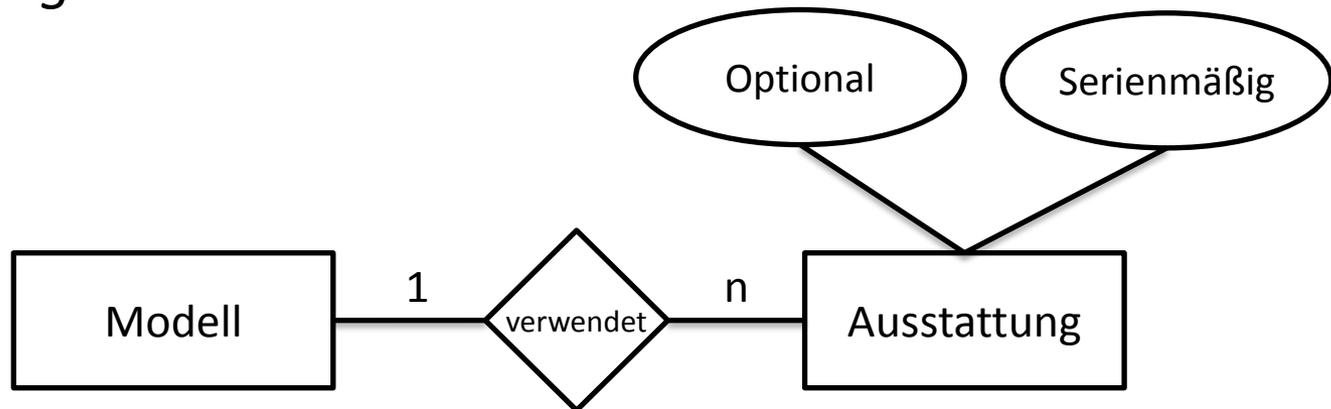
ER Modell, ER Diagramm, R Schema



Beziehungen mit Attributen

Es wird eine Menge von Ausstattungen verwaltet, wobei eine Ausstattung jeweils von mehreren Modellen genutzt werden kann. [...]

Zu jedem Modell wird eine Menge von serienmäßigen Ausstattungen festgelegt, ebenso eine Menge von optionalen Ausstattungen.

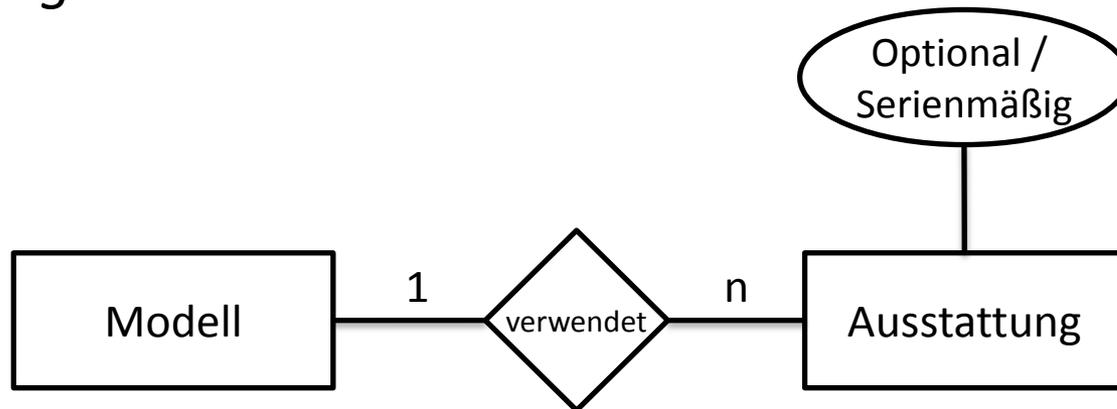


- Ausstattungen müssen für jedes Modell neu erfasst werden.
- Eine Ausstattung kann gleichzeitig serienmäßig und optional sein.

Beziehungen mit Attributen

Es wird eine Menge von Ausstattungen verwaltet, wobei eine Ausstattung jeweils von mehreren Modellen genutzt werden kann. [...]

Zu jedem Modell wird eine Menge von serienmäßigen Ausstattungen festgelegt, ebenso eine Menge von optionalen Ausstattungen.

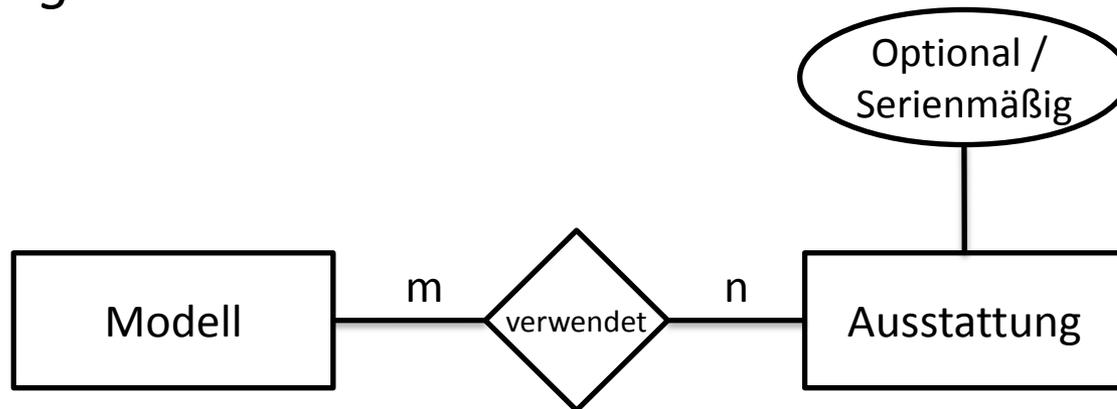


- Ausstattungen müssen für jedes Modell neu erfasst werden.
- Eine Ausstattung kann gleichzeitig serienmäßig und optional sein.

Beziehungen mit Attributen

Es wird eine Menge von Ausstattungen verwaltet, wobei eine Ausstattung jeweils von mehreren Modellen genutzt werden kann. [...]

Zu jedem Modell wird eine Menge von serienmäßigen Ausstattungen festgelegt, ebenso eine Menge von optionalen Ausstattungen.

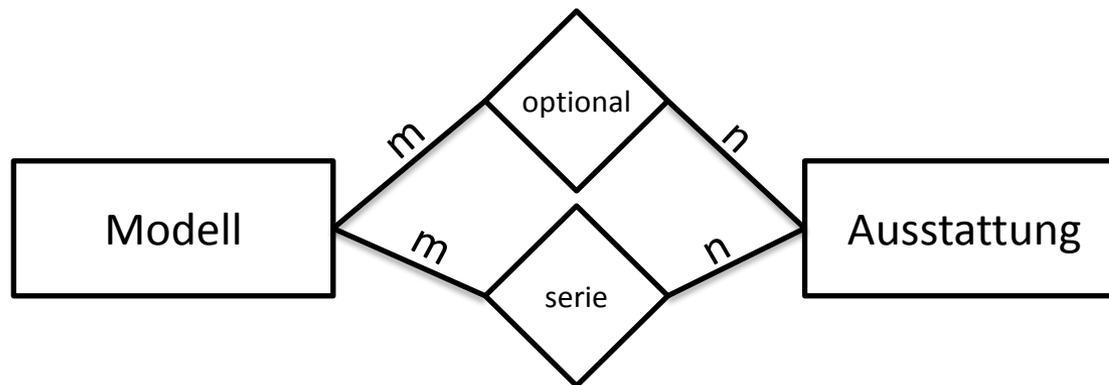


- ~~Ausstattungen müssen für jedes Modell neu erfasst werden.~~
- ~~Eine Ausstattung kann gleichzeitig serienmäßig und optional sein.~~
- ~~Ausstattungen müssen immer doppelt erfasst werden.~~

Beziehungen mit Attributen

Es wird eine Menge von Ausstattungen verwaltet, wobei eine Ausstattung jeweils von mehreren Modellen genutzt werden kann. [...]

Zu jedem Modell wird eine Menge von serienmäßigen Ausstattungen festgelegt, ebenso eine Menge von optionalen Ausstattungen.

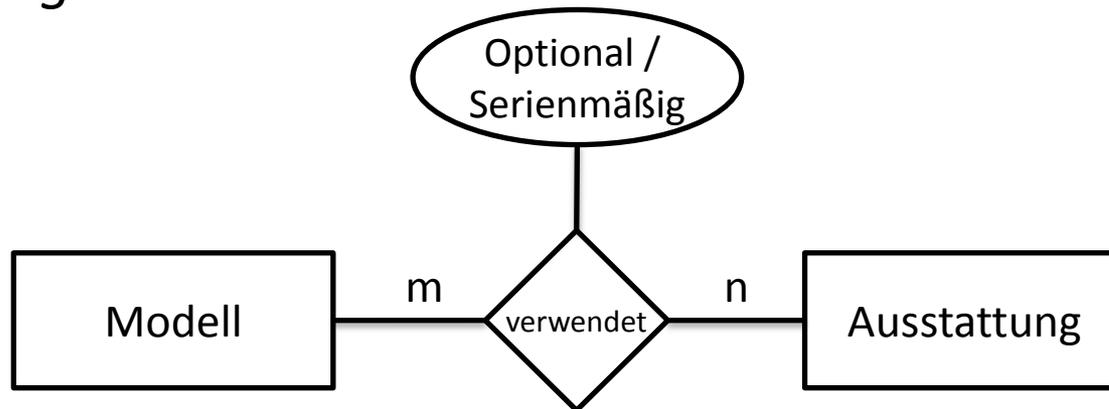


- ~~Ausstattungen müssen für jedes Modell neu erfasst werden.~~
- Eine Ausstattung kann gleichzeitig serienmäßig und optional sein.
- ~~Ausstattungen müssen immer doppelt erfasst werden.~~

Beziehungen mit Attributen

Es wird eine Menge von Ausstattungen verwaltet, wobei eine Ausstattung jeweils von mehreren Modellen genutzt werden kann. [...]

Zu jedem Modell wird eine Menge von serienmäßigen Ausstattungen festgelegt, ebenso eine Menge von optionalen Ausstattungen.



- ~~Ausstattungen müssen für jedes Modell neu erfasst werden.~~
- ~~Eine Ausstattung kann gleichzeitig serienmäßig und optional sein.~~
- ~~Ausstattungen müssen immer doppelt erfasst werden.~~

Rekursive Beziehungen

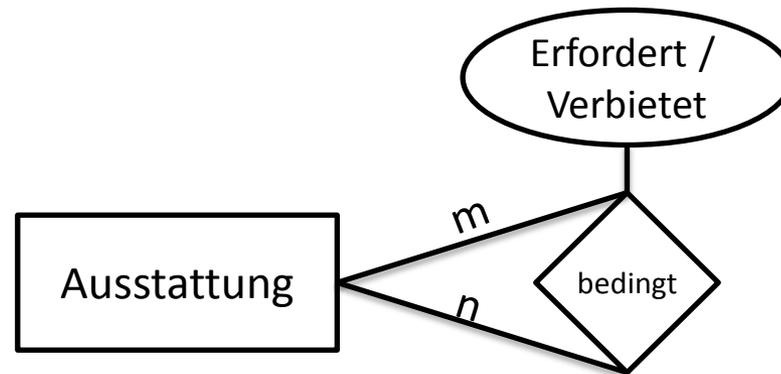
Darüber hinaus verwaltet das System zu jeder Ausstattung modellunabhängig, welche anderen Ausstattungen dafür vorausgesetzt werden [...] und welche Ausstattungen dadurch ausgeschlossen werden [...]. Im Modell müssen daher zwar die Vorbedingungen und Ausschlüsse abgebildet werden, jedoch nicht etwaige Konsistenzbedingungen.



Eine Ausstattung kann eine andere gleichzeitig erfordern und verbieten.

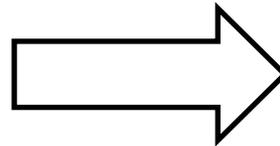
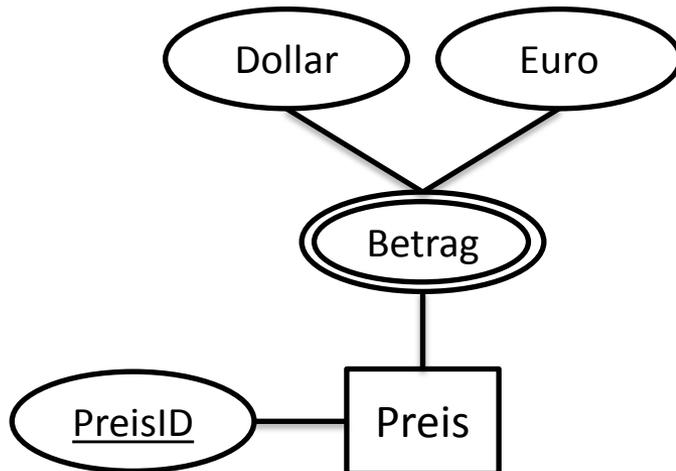
Rekursive Beziehungen

Darüber hinaus verwaltet das System zu jeder Ausstattung modellunabhängig, welche anderen Ausstattungen dafür vorausgesetzt werden [...] und welche Ausstattungen dadurch ausgeschlossen werden [...]. Im Modell müssen daher zwar die Vorbedingungen und Ausschlüsse abgebildet werden, jedoch nicht etwaige Konsistenzbedingungen.



Eine Ausstattung kann eine andere gleichzeitig erfordern und verbieten.

Zusammengesetzte Attribute



Preis	
PreisID	INT
Betrag_Euro	DECIMAL
Betrag_Dollar	DECIMAL

Der Name eines zusammengesetzten Attributes fungiert z.B. als Präfix für die Spaltennamen, taucht jedoch nicht selbst als Spalte in der Tabelle auf.