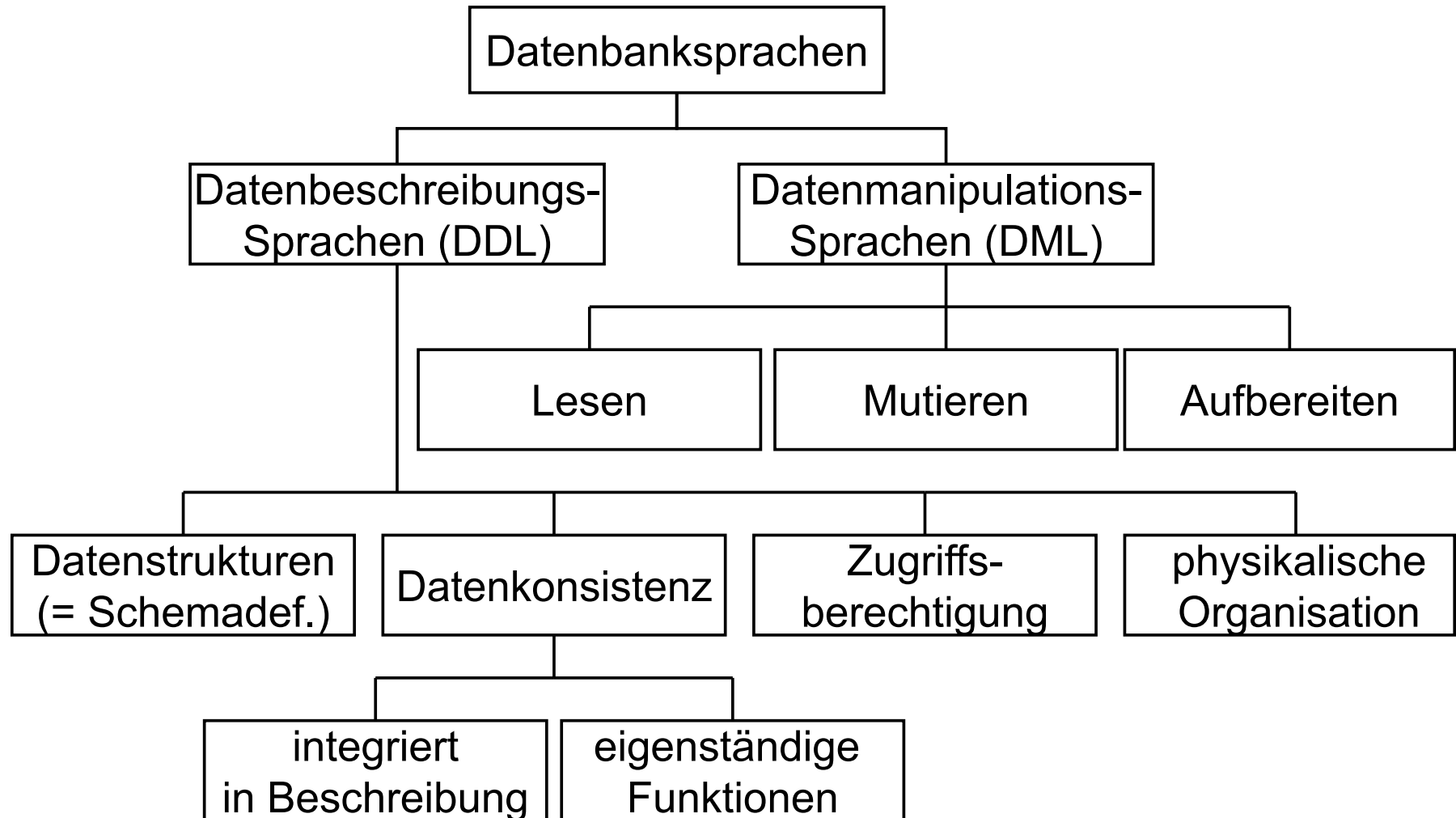


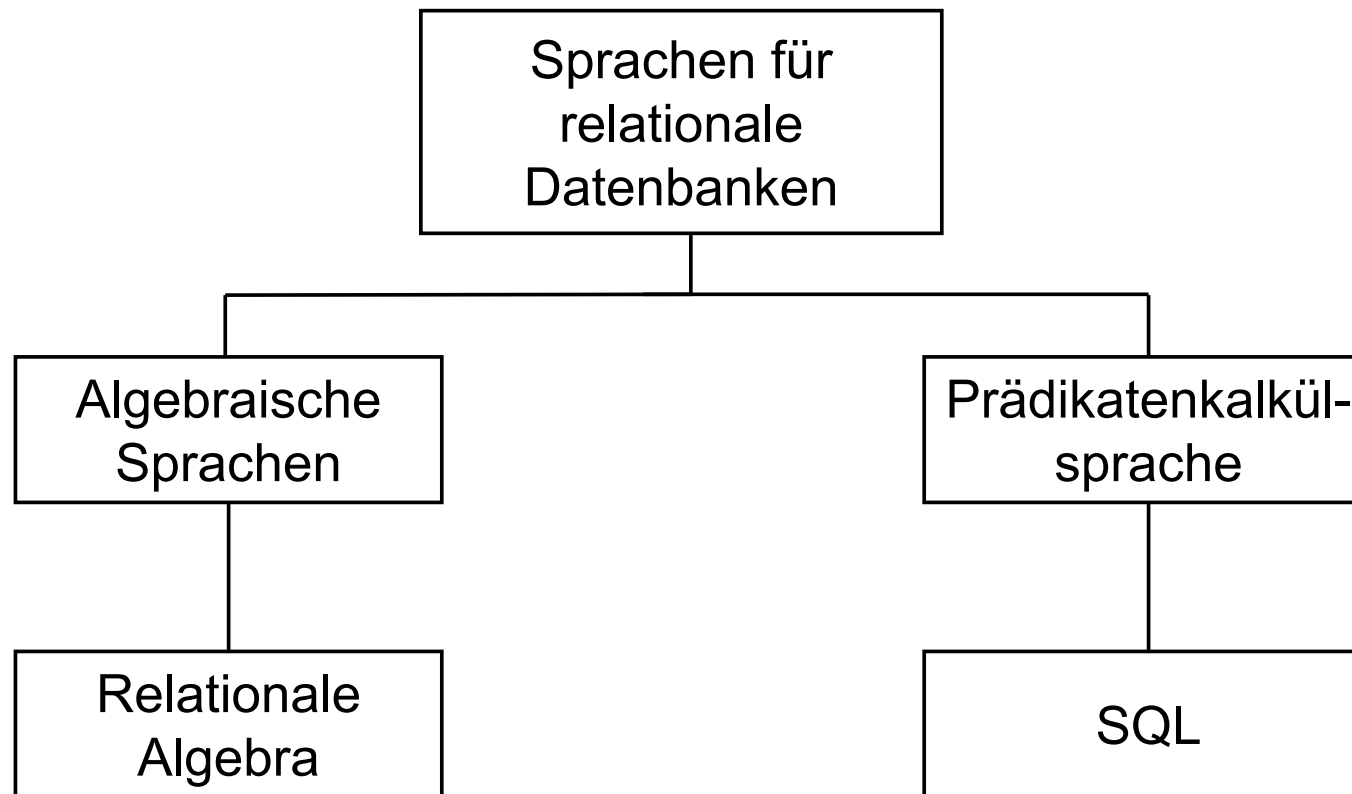
5 Datenbanken

1. Motivation
2. Datenorganisation und Datenbankkonzept
3. Semantische Datenmodellierung
4. Umsetzung in Datenbanken
- 5. Datenbanknutzung mit SQL**
6. Transaktionsmanagement
7. Datenbankentwicklung
8. Datenbanken und IT-Sicherheit
9. Systemarchitektur
10. Verteilte Datenbanken
11. NoSQL und Entwicklungstrends

5 Lernziele

- Sie verstehen die mathematischen Konzepte der Relationenalgebra, welche die Grundlage aller Datenbankoperationen darstellt.
- Sie kennen die SQL-Befehle für folgende Operationen:
 - Tabellen anlegen und verwalten
 - Datensätze anlegen und verwalten
 - Daten auslesen und auswerten







Eine **Relation** R ist eine Teilmenge des kartesischen Produktes der (nicht notwendigerweise disjunkten) Wertebereiche W_i von n Attributen A_i , $W_i = \text{dom}(A_i)$, $i = 1, \dots, n$:

$$R \subseteq W_1 \times W_2 \times \dots \times W_n$$

Die Relation R stellt somit eine Menge von n -Tupeln dar:

$$R = \{(w_1, w_2, \dots, w_n) \mid w_i \in W_i, i = 1, \dots, n\}$$

5 Relation – Beispiel

Geschlecht: $W_1 = \{m, w, d\}$

Familienstand: $W_2 = \{\text{ledig, verheiratet, geschieden, verwitwet}\}$

Mögliche Relation Mitarbeiter

Relationstyp: Mitarbeiter(Geschlecht, Familienstand)

$W_1 \times W_2 = \text{Geschlecht} \times \text{Familienstand} =$

$\{(m, \text{ledig}), (m, \text{verheiratet}), (m, \text{geschieden}), (m, \text{verwitwet}),$
 $(w, \text{ledig}), (w, \text{verheiratet}), (w, \text{geschieden}), (w, \text{verwitwet}),$
 $(d, \text{ledig}), (d, \text{verheiratet}), (d, \text{geschieden}), (d, \text{verwitwet})\}$

Mitarbeiter = $\{(m, \text{ledig}), (m, \text{verwitwet}),$
 $(w, \text{verheiratet}), (w, \text{verwitwet})\}$

Mitarbeiter	
Geschlecht	Familienstand
m	ledig
m	verwitwet
w	verheiratet
w	verwitwet

5 Relation

Attribute		Mitarbeiter		Tabellenkopf / Relationstyp
		Geschlecht	Familienstand	
Aus Domäne mit Wertebereich m, w, d		m	ledig	Tupel / Datensatz
		m	verwitwet	
		w	verheiratet	
		w	verwitwet	

Grad der Relation
= Anzahl der Attribute

Seien R und S zwei Relationen:

Klassische Operationen

- | | |
|---------------------------|--------------|
| 1. Vereinigung | $R \cup S$ |
| 2. Durchschnitt | $R \cap S$ |
| 3. Differenz | $R - S$ |
| 4. Symmetrische Differenz | R / S |
| 5. Kartesisches Produkt | $R \times S$ |

Relationenspezifische Operationen

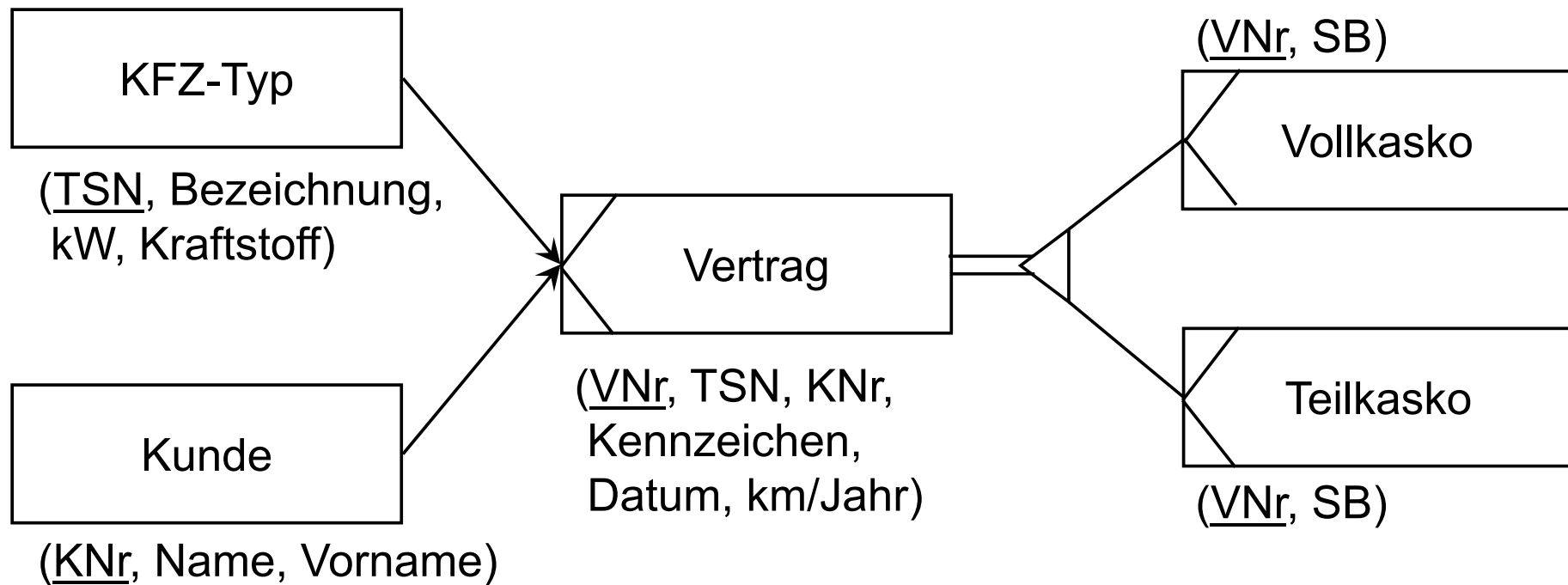
- | | |
|-------------------|------------------------|
| 6. Projektion | $R[a]$ |
| 7. Verbund (Join) | $R[op_1 \oplus op_2]S$ |
| 8. Restriktion | $R[op_1 \oplus op_2]$ |

Hierbei sind op_1 und op_2 Operanden (z.B. Attribute oder Strings) zur Operation \oplus .

Eine Datenbanksprache wird **relational vollständig** genannt, wenn sie mindestens den Umfang der relationalen Algebra umfasst.

- Mit Hilfe der relationalen Algebra lassen sich beliebige Teilmengen aus einem gegebenen Satz von Relationen extrahieren.

5 Fallbeispiel



5 Fallbeispiel

KFZ-Typ			
<u>TSN</u>	Bezeichnung	kW	Kraftstoff
773	Seat Leon 1,6 16 V	77	Benzin
851	Audi A4 2,5 TDI	120	Diesel
503	BMW 735i	200	Benzin
456	Mercedes Benz 220	110	Diesel
167	Mazda MX-5 1,9 l	107	Benzin

Kunde		
<u>KNr</u>	Name	Vorname
1	Bliemel	Harald
2	Schneider	Gertrud
3	Kawulske	Dieter
4	Heinrich	Klaus

5 Fallbeispiel

Vertrag					
<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH PE 156	03.05.14	20.000
2457	456	2	DO S 256	04.08.13	19.000
3549	167	3	HA B 9864	15.03.15	8.500
1756	851	1	EN AA 456	28.01.12	12.500

Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

Teilkasko	
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600



Zwei Relationen gleichen Grades heißen **vereinigungsverträglich**, wenn jedem Attribut der ersten Relation ein Attribut gleichen Datentyps der zweiten Relation zugeordnet werden kann.

Intuitiv: Zwei Tabellen können vereinigt werden, wenn beide Tabellen den gleichen Aufbau haben.

Es seien R und S zwei vereinigungsverträgliche Relationen. Dann besteht die **Vereinigung** $R \cup S$ in der Menge aller Datensätze, die in R oder in S bzw. in beiden Relationen enthalten sind:

$$R \cup S = \{t \mid t \in R \vee t \in S\}$$



Es seien R und S zwei vereinigungsverträgliche Relationen. Der **Durchschnitt** $R \cap S$ besteht aus der Menge aller Tupel, die sowohl in der Relation R als auch in der Relation S enthalten sind:

$$R \cap S = \{t \mid t \in R \wedge t \in S\}$$



Vollkasko

<u>VNr</u>	SB
1578	100
2457	0
3549	600

Teilkasko

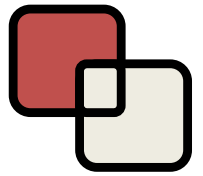
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600

Vollkasko \cup Teilkasko

<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600
1578	100

Vollkasko \cap Teilkasko

<u>VNr</u>	SB
2457	0
3549	600



Es seien R und S zwei vereinigungsverträgliche Relationen. Die **Differenz $R - S$** besteht aus der Menge aller Tupel, die in R aber nicht zugleich in S enthalten sind:

$$R - S = \{t \mid t \in R \wedge t \notin S\}$$

Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

Teilkasko	
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600

Teilkasko – Vollkasko

<u>VNr</u>	SB
1756	0
1578	150



Es seien R und S zwei vereinigungsverträgliche Relationen. Die **symmetrische Differenz** R/S besteht aus der Menge aller Tupel, die in R oder in S aber nicht zugleich in R und in S enthalten sind:

$$R/S = \{t \mid t \in R \text{ oder } t \in S \wedge t \notin R \cap S\}$$

$$\text{bzw. } R/S = (R - S) \cup (S - R)$$

Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

Teilkasko	
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600

Teilkasko / Vollkasko

<u>VNr</u>	SB
1578	100
1578	150
1756	0

Es seien R und S zwei Relationen beliebigen Grades. Dann besteht das **kartesische Produkt** $R \times S$ aus der Menge aller möglichen Verkettungen von Datensätzen aus R und aus S :

$$R \times S = \{r \sim s \mid r \in R \wedge s \in S\}$$



Vertrag \times Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
-----	-----	-----	-------------	-------	---------	-----	----

Vertrag					
<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH PE 156	03.05.14	20.000
2457	456	2	DO S 256	04.08.13	19.000
3549	167	3	HA B 9864	15.03.15	8.500
1756	851	1	EN AA 456	28.01.12	12.500

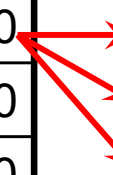
Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

Vertrag \times Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
-----	-----	-----	-------------	-------	---------	-----	----

Vertrag					
<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH PE 156	03.05.14	20.000
2457	456	2	DO S 256	04.08.13	19.000
3549	167	3	HA B 9864	15.03.15	8.500
1756	851	1	EN AA 456	28.01.12	12.500

Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600



Vertrag \times Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
1578	503	4	HH PE 157	03.05.14	20.000	1578	100
1578	503	4	HH PE 157	03.05.14	20.000	2457	0
1578	503	4	HH PE 157	03.05.14	20.000	3549	600

Vertrag					
<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH PE 156	03.05.14	20.000
2457	456	2	DO S 256	04.08.13	19.000
3549	167	3	HA B 9864	15.03.15	8.500
1756	851	1	EN AA 456	28.01.12	12.500

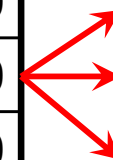
Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

Vertrag \times Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
1578	503	4	HH PE 157	03.05.14	20.000	1578	100
1578	503	4	HH PE 157	03.05.14	20.000	2457	0
1578	503	4	HH PE 157	03.05.14	20.000	3549	600

Vertrag					
<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH PE 156	03.05.14	20.000
2457	456	2	DO S 256	04.08.13	19.000
3549	167	3	HA B 9864	15.03.15	8.500
1756	851	1	EN AA 456	28.01.12	12.500

Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600



Vertrag \times Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
1578	503	4	HH PE 157	03.05.14	20.000	1578	100
1578	503	4	HH PE 157	03.05.14	20.000	2457	0
1578	503	4	HH PE 157	03.05.14	20.000	3549	600
2457	456	2	DO S 256	04.08.13	19.000	1578	100
2457	456	2	DO S 256	04.08.13	19.000	2457	0
2457	456	2	DO S 256	04.08.13	19.000	3549	600

Vertrag					
<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH PE 156	03.05.14	20.000
2457	456	2	DO S 256	04.08.13	19.000
3549	167	3	HA B 9864	15.03.15	8.500
1756	851	1	EN AA 456	28.01.12	12.500

Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

Vertrag \times Vollkasko

VNr	TSN	KNr	Kennzeichen	Datum	km/Jahr	VNr	SB
1578	503	4	HH PE 157	03.05.14	20.000	1578	100
1578	503	4	HH PE 157	03.05.14	20.000	2457	0
1578	503	4	HH PE 157	03.05.14	20.000	3549	600
2457	456	2	DO S 256	04.08.13	19.000	1578	100
2457	456	2	DO S 256	04.08.13	19.000	2457	0
2457	456	2	DO S 256	04.08.13	19.000	3549	600
3549	167	3	HA B 9864	15.03.15	8.500	1578	100
3549	167	3	HA B 9864	15.03.15	8.500	2457	0
3549	167	3	HA B 9864	15.03.15	8.500	3549	600
1756	851	1	EN AA 456	28.01.12	12.500	1578	100
1756	851	1	EN AA 456	28.01.12	12.500	2457	0
1756	851	1	EN AA 456	28.01.12	12.500	3549	600

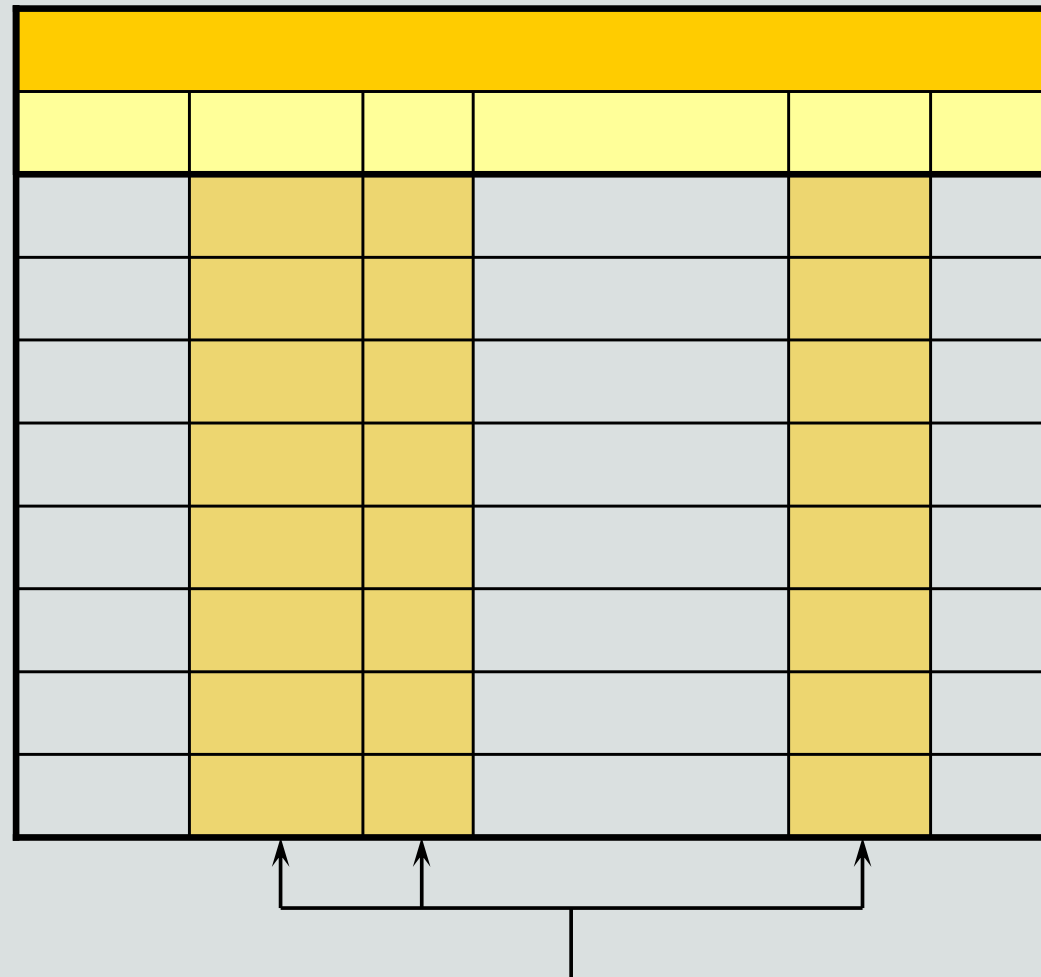


Sei R eine Relation mit den Attributen a_1, a_2, \dots, a_M und sei a eine in R vertretene Attributkombination. Dann ist die **Projektion** der Relation R auf die Attributkombination a definiert als:

$$R[a] = \{t[a] \mid t \in R \text{ ein beliebiges Tupel}\}$$

Alternative Schreibweise:

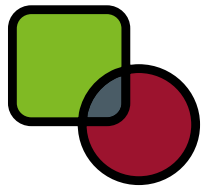
$$\pi_a(R) = \{t_a \mid t \in R\}$$



Kunde		
<u>KNr</u>	Name	Vorname
1	Bliemel	Harald
2	Schneider	Gertrud
3	Kawulske	Dieter
4	Heinrich	Klaus

Kunde[KNr, Name]

KNr	Name
1	Bliemel
2	Schneider
3	Kawulske
4	Heinrich



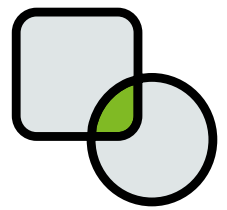
Es seien R und S zwei beliebige Relationen, wobei a und c vereinigungsverträgliche Attributkombinationen aus den Relationen R bzw. S sind. Dann ist der **Verbund** der Relation R und S über die Attributkombination a und c definiert als:

$$R[a \oplus c]S = \{r \sim s \mid r \in R, s \in S \wedge r[a] \oplus s[c]\}$$

Alternative Schreibweise:

$$R \bowtie_{\text{Ausdruck}} S = \{r \cup s \mid r \in R \wedge s \in S \wedge \text{Ausdruck}\}$$

Bemerkung: \oplus ist hier ein beliebiger Operator.



Der **Equi-Join** zweier Relationen R und S über die vereinigungsverträglichen Attributkombinationen a und c ist definiert als:

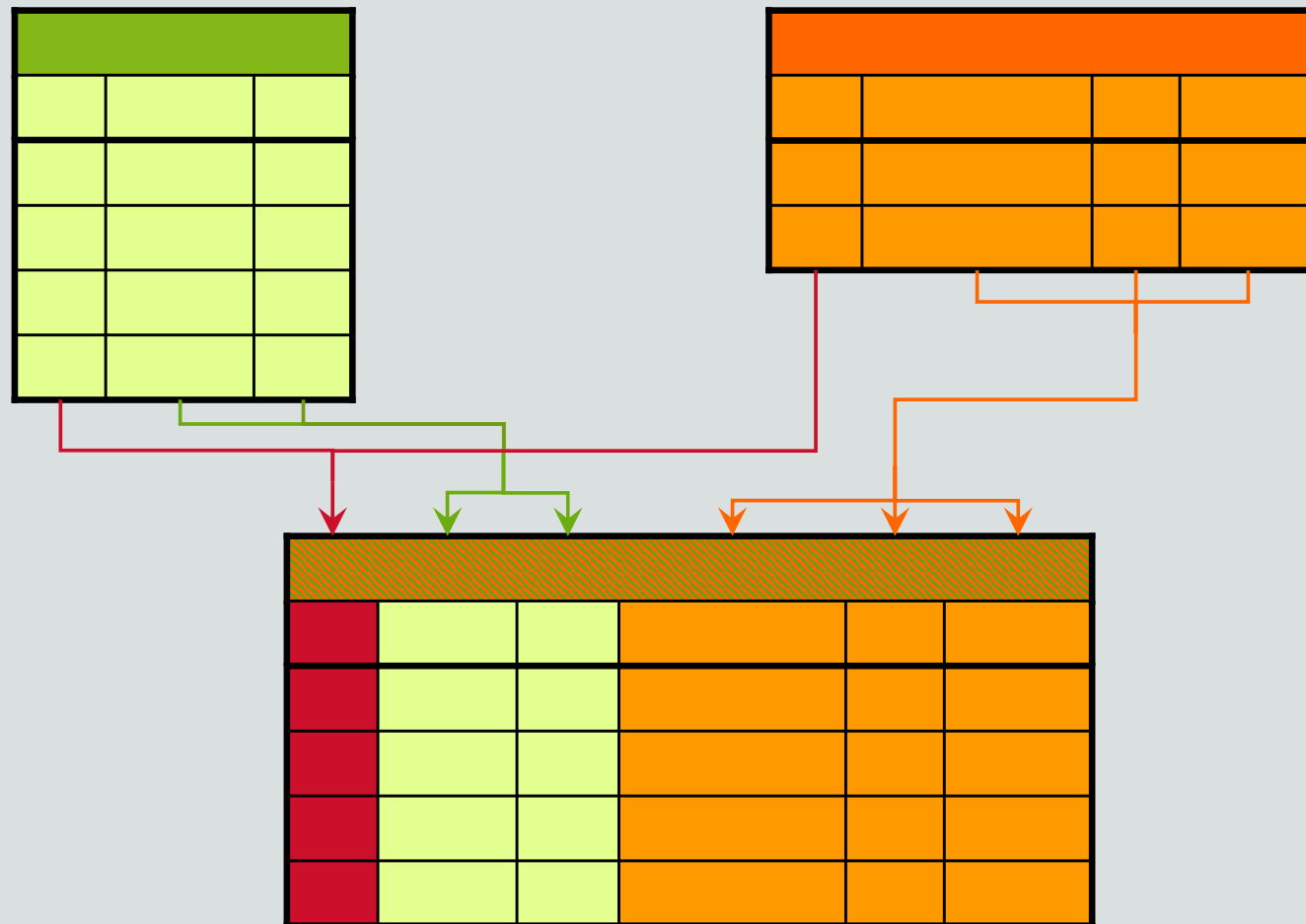
$$R[a=c]S = \{r \sim s - s[c] \mid r \in R, s \in S \wedge r[a] = s[c]\}$$

Alternative Schreibweise:

$$R \bowtie_{a=c} S = \{r \cup s \mid r \in R \wedge s \in S \wedge r[a] = s[c]\}$$

Bemerkung: Hier ist es immer der Operator $=$.

Die doppelte Spalte $s[c]$ wird im Join weggelassen.



Kunde			Vertrag					
<u>KNr</u>	Name	Vorname	<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1	Bliemel	Harald	1578	503	4	HH PE 156	03.05.14	20.000
2	Schneider	Gertrud	2457	456	2	DO S 256	04.08.13	19.000
3	Kawulske	Dieter	3549	167	3	HA B 9864	15.03.15	8.500
4	Heinrich	Klaus	1756	851	1	EN AA 456	28.01.12	12.500

P := Kunde [Kunde.KNr = Vertrag.KNr] Vertrag

KNr	Name	Vorname	VNr	TSN	Kennzeichen	Datum	km/Jahr
1	Bliemel	Harald	1756	851	EN AA 456	28.01.12	12.500
2	Schneider	Gertrud	2457	456	DO S 256	04.08.13	19.000
3	Kawulske	Dieter	3549	167	HA B 9864	15.03.15	8.500
4	Heinrich	Klaus	1578	503	HH PE 156	03.05.14	20.000



Der **Natural Join** ist ein Equi-Join, bei dem die vereinigungsverträglichen Attributkombinationen aus den Relationen R und S aus den gleich benannten Attributen in R und S bestehen.

Sei $R(A_1, \dots, A_m, B_1, \dots, B_n)$ und $S(B_1, \dots, B_n, C_1, \dots, C_m)$ zwei Relationen. Dann ist

$$R \bowtie S = \{r \cup s[C_1, \dots, C_m] \mid r \in R \wedge s \in S \wedge r[B_1, \dots, B_n] = s[B_1, \dots, B_n]\}$$

Gibt es keine gemeinsamen Attribute, so ist das Ergebnis des natürlichen Verbundes das kartesische Produkt.

KFZ-Typ			
<u>TSN</u>	Bezeichnung	kW	Kraftstoff
773	Seat Leon 1,6 16 V	77	Benzin
851	Audi A4 2,5 TDI	120	Diesel
503	BMW 735i	200	Benzin
456	Mercedes Benz 220	110	Diesel
167	Mazda MX-5 1,9 l	107	Benzin

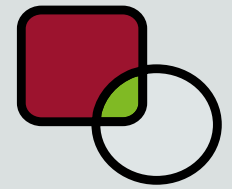
Kunde		
<u>KNr</u>	Name	Vorname
1	Bliemel	Harald
2	Schneider	Gertrud
3	Kawulske	Dieter
4	Heinrich	Klaus

5 Fallbeispiel

Vertrag					
<u>VNr</u>	TSN	KNr	Kennzeichen	Datum	km/Jahr
1578	503	4	HH PE 156	03.05.14	20.000
2457	456	2	DO S 256	04.08.13	19.000
3549	167	3	HA B 9864	15.03.15	8.500
1756	851	1	EN AA 456	28.01.12	12.500

Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

Teilkasko	
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600



Es seien R und S zwei beliebige Relationen, wobei a und c vereinigungsverträgliche Attributkombinationen aus den Relationen R bzw. S sind. Der **Left-Outer-Join** ist ein Spezialfall des Equi-Joins, bei dem die Tupel aus R , die keine Entsprechung in S haben, in die Ergebnismenge aufgenommen und mit leeren Werten (NULL) aufgefüllt werden.

$$R [a=c]_{LO} S = \{r \sim s - s[c] \mid r \in R, s \in S \wedge r[a] = s[c]\} \\ \cup \{r \sim \text{NULL}^{|s - s[c]|} \mid r \in R \wedge \nexists s[c]: r[a] = s[c]\}$$

Alternative Schreibweise:

$$R \bowtie_{a=c} S$$

5 Relationenalgebra – Left-Outer-Join

KFZ-Typ

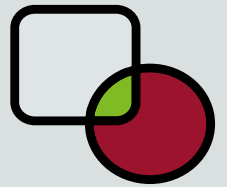
<u>TSN</u>	Bezeichnung	kW	Kraftstoff
773	Seat Leon 1,6 16 V	77	Benzin
851	Audi A4 2,5 TDI	120	Diesel
503	BMW 735i	200	Benzin
456	Mercedes Benz 220	110	Diesel
167	Mazda MX-5 1,9 l	107	Benzin

Vertrag

<u>VNr</u>	TSN	KNr	Kennz.	Datum	km/Jahr
1578	503	4	HH PE 156	03.05.14	20.000
2457	456	2	DO S 256	04.08.13	19.000
3549	167	3	HA B 9864	15.03.15	8.500
1756	851	1	EN AA 456	28.01.12	12.500

LOJ:= KFZ-Typ[KFZ-Typ.TSN = Vertrag.TSN]_{LO} Vertrag

TSN	Bezeichnung	kW	Kraftstoff	VNr	KNr	Kennz.	Datum	km/Jahr
773	Seat Leon 1,6 16 V	77	Benzin					
851	Audi A4 2,5 TDI	120	Diesel	1756	1	EN AA 456	28.01.12	12.500
503	BMW 735i	200	Benzin	1578	4	HH PE 156	03.05.14	20.000
456	Mercedes Benz 220	110	Diesel	2457	2	DO S 256	04.08.13	19.000
167	Mazda MX-5 1,9 l	107	Benzin	3549	3	HA B 9864	15.03.15	8.500



Es seien R und S zwei beliebige Relationen, wobei a und c vereinigungsverträgliche Attributkombinationen aus den Relationen R bzw. S sind. Der **Right-Outer-Join** ist ein Spezialfall des Equi-Joins, bei dem die Tupel aus S , die keine Entsprechung in R haben, in die Ergebnismenge aufgenommen und mit leeren Werten (NULL) aufgefüllt werden.

$$\begin{aligned} R[a=c]_{RO} S = \{ r \sim s - s[c] \mid r \in R, s \in S \wedge r[a] = s[c] \} \\ \cup \{ \text{NULL} \mid r - s[c] \sim s \mid s \in S \wedge \nexists r[a]: r[a] = s[c] \} \end{aligned}$$

Alternative Schreibweise:

$$R \bowtie_{a=c} S$$

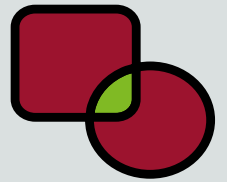
5 Relationenalgebra – Right-Outer-Join

Vollkasko	
<u>VNr</u>	SB
1578	100
2457	0
3549	600

Teilkasko	
<u>VNr</u>	SB
1756	0
1578	150
2457	0
3549	600

ROJ:= Vollkasko[Vollkasko.VNr = Teilkasko.VNr]_{RO} Teilkasko

VNr	Vollkasko.SB	Teilkasko.SB
1578	100	150
2457	0	0
3549	600	600
1756		0



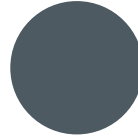
Es seien R und S zwei beliebige Relationen, wobei a und c vereinigungsverträgliche Attributkombinationen aus den Relationen R bzw. S sind. Der **Full-Outer-Join** ist ein Spezialfall des Equi-Joins, bei dem die Tupel, die keine Entsprechung in der jeweils anderen Relation haben, in die Ergebnismenge aufgenommen und mit leeren Werten (NULL) aufgefüllt werden.

$$\begin{aligned} R[a=c]_{FO} S = & \{r \sim s - s[c] \mid r \in R, s \in S \wedge r[a] = s[c]\} \\ & \cup \{r \sim \text{NULL} \mid s - s[c] \mid r \in R \wedge \nexists s[c]: r[a] = s[c]\} \\ & \cup \{\text{NULL} \mid r - s[c] \sim s \mid s \in S \wedge \nexists r[a]: r[a] = s[c]\} \end{aligned}$$

Alternative Schreibweise:

$$R \bowtie_{a=c} S$$

Datensatz der linken Tabelle:
Query liefert die roten Mengen.



Datensatz der rechten Tabelle:



■ Inner Join

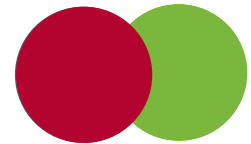
- Nur Datensätze anzeigen, welche direkt miteinander verbunden sind.
- Datensätze ohne Verknüpfung werden nicht angezeigt



■ Outer Join

- Datensätze mit direkter Verbindung werden angezeigt.
- Nicht-verknüpfte Datensätze werden ebenfalls angezeigt.

→ **Left** (linker Verbund) $R \bowtie_{a=c} S$



→ **Right** (rechter Verbund) $R \bowtie_{a=c} S$



→ **Full** (voller Verbund) $R \bowtie_{a=c} S$





Es seien a eine Attributkombinationen aus der Relation R und c eine mit a vereinigungsverträgliche Kombination aus konstanten Werten bzw. Attributen von R . Dann ist die **Restriktion / Selection** der Relation R bzgl. der Attributkombinationen a und c definiert als:

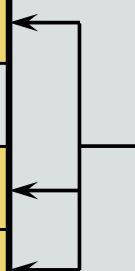
$$R[a \oplus c] = \{t \mid t \in R \wedge t[a] \oplus t[c]\}$$

Es sei R eine Relation und B eine Bedingung, die auf einer Attributkombination a aus R definiert ist. So ist die Restriktion/Selection der Relation R bzgl. B definiert als:

$$R_B = \{t \mid t \in R \wedge B(t) = \text{wahr}\}$$

Alternative Schreibweise:

$$\sigma_B(R) = \{t \mid t \in R \wedge t \text{ erfüllt } B\}$$



KFZ-Typ			
<u>TSN</u>	Bezeichnung	kW	Kraftstoff
773	Seat Leon 1,6 16 V	77	Benzin
851	Audi A4 2,5 TDI	120	Diesel
503	BMW 735i	200	Benzin
456	Mercedes Benz 220	110	Diesel
167	Mazda MX-5 1,9 l	107	Benzin

R := KFZ-Typ [kW>=110]

TSN	Bezeichnung	kW	Kraftstoff
851	Audi A4 2,5 TDI	120	Diesel
503	BMW 735i	200	Benzin
456	Mercedes Benz 220	110	Diesel

- Vereinigung

$$R \cup S = \{t \mid t \in R \text{ oder } t \in S\}$$

- Durchschnitt

$$R \cap S = \{t \mid t \in R \text{ und } t \in S\}$$

- Differenz

$$R - S = \{t \mid t \in R \text{ und } t \notin S\}$$

- Symmetrische Differenz

$$R/S = \{t \mid t \in R \text{ oder } t \in S \text{ und } t \notin R \cap S\}$$

- Kartesisches Produkt

$$R \times S = \{r \times s \mid r \in R \text{ und } s \in S\}$$

- Projektion

$$\pi_a(\mathbf{R}) = \{t_a \mid t \in \mathbf{R}\}$$

- Verbund-Join

$$\mathbf{R} \bowtie_{\text{Ausdruck}} \mathbf{S} = \{r \sim s \mid r \in \mathbf{R}, s \in \mathbf{S} \text{ und } r[a] \oplus s[c]\}$$

- Equi-Join

$$\mathbf{R} \bowtie_{a=c} \mathbf{S} = \{r[a] \sim s[c] \mid r \in \mathbf{R}, s \in \mathbf{S} \text{ und } r[a] = s[c]\}$$

- Restriktion/Selection

$$\sigma_B(\mathbf{R}) = \{t \mid t \in \mathbf{R} \wedge t \text{ erfüllt } B\}$$

5 Structured Query Language – SQL

Was ist SQL?

- Datenbanksprache
- Deskriptiv (das gewünschte Ergebnis beschreibend)
- Basis: Relationenalgebra bzw. Mengenlehre
- Aufgaben
 - Definition relationaler Datenbanken
 - Bearbeitung und Auswertung der Daten

Merkmale von SQL

- Plattformunabhängigkeit
- Normung
 - American National Standards Institute (ANSI)
 - International Standards Organisation (ISO)
- Große Verbreitung
- Kleiner, aber mächtiger Sprachumfang

- Data Definition Language (DDL)
 - Definition der Datenbankstruktur
- View Definition Language (VDL)
 - Definition von Sichten auf die Daten
- Data Manipulation Language (DML)
 - Manipulation und Retrieval von Datensätzen
- Data Control Language (DCL)
 - Definition von Zugriffsrechten
- Data Storage Definition Language (DSDL)
 - Definition der physischen Speicherstruktur

5 SQL Standard

- Trotz Standardisierung Unterschiede in den verschiedenen Implementierungen
- DDL, VDL und DML weitgehend konform
- Unterschiede in DCL und DSDL
 - ➔ Handbücher und Dokumentationen
- Es gibt Erweiterungen zum SQL
 - z.B. im Datawarehouse-Kontext (Cube/Rollup-Operator)

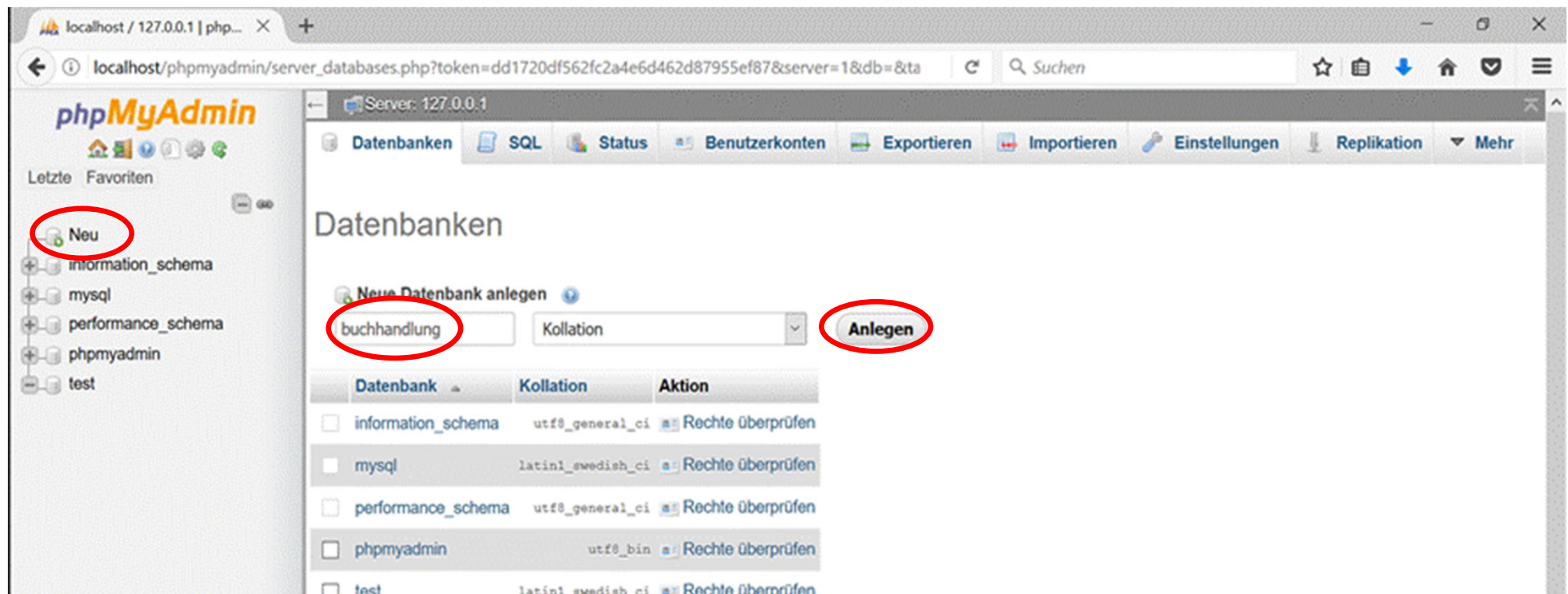
- Datenbankmanagementsysteme können mehrere Datenbanken parallel verwalten

```
CREATE DATABASE <dbname>;
```

```
USE <dbname>;
```

5 Erstellung einer Datenbank – phpMyAdmin

- Menü links: Neu → Namen eingeben → Anlegen
- SQL Befehl: **CREATE DATABASE** buchhandlung;



- Definition der Datenbankstruktur
 - Erzeugen
 - Ändern
 - Löschen
- von Datenbanktabellen

```

CREATE TABLE <dbname>.<tabellenname> (
    <spaltenname> DATENTYP [NOT NULL] [{AUTO_INCREMENT | DEFAULT WERT}],
    ...
    <spaltenname> DATENTYP [NOT NULL] [{AUTO_INCREMENT | DEFAULT WERT}],
    [[CONSTRAINT <constraintname>] PRIMARY KEY (<spaltenname> [,... <sp.>]),]
    [[CONSTRAINT <constraintname>] FOREIGN KEY (<spaltenname> [, ... <sp.>])
        REFERENCES <tabellenname> [(<spaltenname>, ... <spaltenname>)]
        [ON {DELETE | UPDATE}
            {RESTRICT | CASCADE | SET NULL | SET DEFAULT | NO ACTION}],]
    [[CONSTRAINT <constraintname>] CHECK <bedingung>],]
    [[CONSTRAINT <constraintname>] UNIQUE (<spaltenname>, ... <sp.>)]
    [COMMENT = '']
);

```

5 Datentypen

Datentyp	Inhalt	Beispiele
INTEGER	ganze Zahl	0 / -5 / 1500
REAL / DOUBLE	Fließkommazahlen	0.0 / 34.23 / -13.4534
NUMERIC(x,y)	Festkommazahlen	0.0 / 34.23 / -13.4534
TEXT	Text unbegrenzter Länge	„Lorem ipsum dolor...“
VARCHAR(max)	Zeichenkette variabler Länge	‘aA?’ / ‘Test’ / “
CHAR(fix)	Zeichenkette fixer Länge	‘aA?’ / ‘Test’ / “
DATE	Datumswert	2021-12-08
TIME	Tageszeit (ohne Datum)	08:47:51
TIMESTAMP	Datum mit Uhrzeit	2038-01-19 03:14:07
... WITH TIME ZONE	Zeitdaten mit Zeitzonendaten	08:47:51+01
BOOLEAN	Logischer Wert	TRUE / FALSE

<https://www.postgresql.org/docs/current/datatype.html>

5 Erzeugen von Tabellen durch Abfragen

```
CREATE TABLE <tabellenname> AS <select befehl>;
```

- „Kopieren“ der Struktur
- Automatisches Füllen mit Datensätzen

5 Ändern von Tabellen

- Nur Änderungen ohne Informationsverlust

```
ALTER TABLE <tabellenname> ADD COLUMN <spaltendefinition>;
```

```
ALTER TABLE <tabellenname> DROP <spaltenname>;
```

```
ALTER TABLE <tabellenname> ADD CONSTRAINT <constraintdefinition>;
```

```
ALTER TABLE <tabellenname> DROP CONSTRAINT <constraintname>;
```

```
ALTER TABLE <tabellenname> ALTER <spaltendefinition>;
```

```
ALTER TABLE <tabellenname> RENAME <spaltenname> TO <neuer_name>;
```

```
ALTER TABLE <tabellenname> ALTER <spaltenname> DROP DEFAULT;
```

```
ALTER TABLE <tabellenname> ALTER <spaltenname> SET NOT NULL;
```

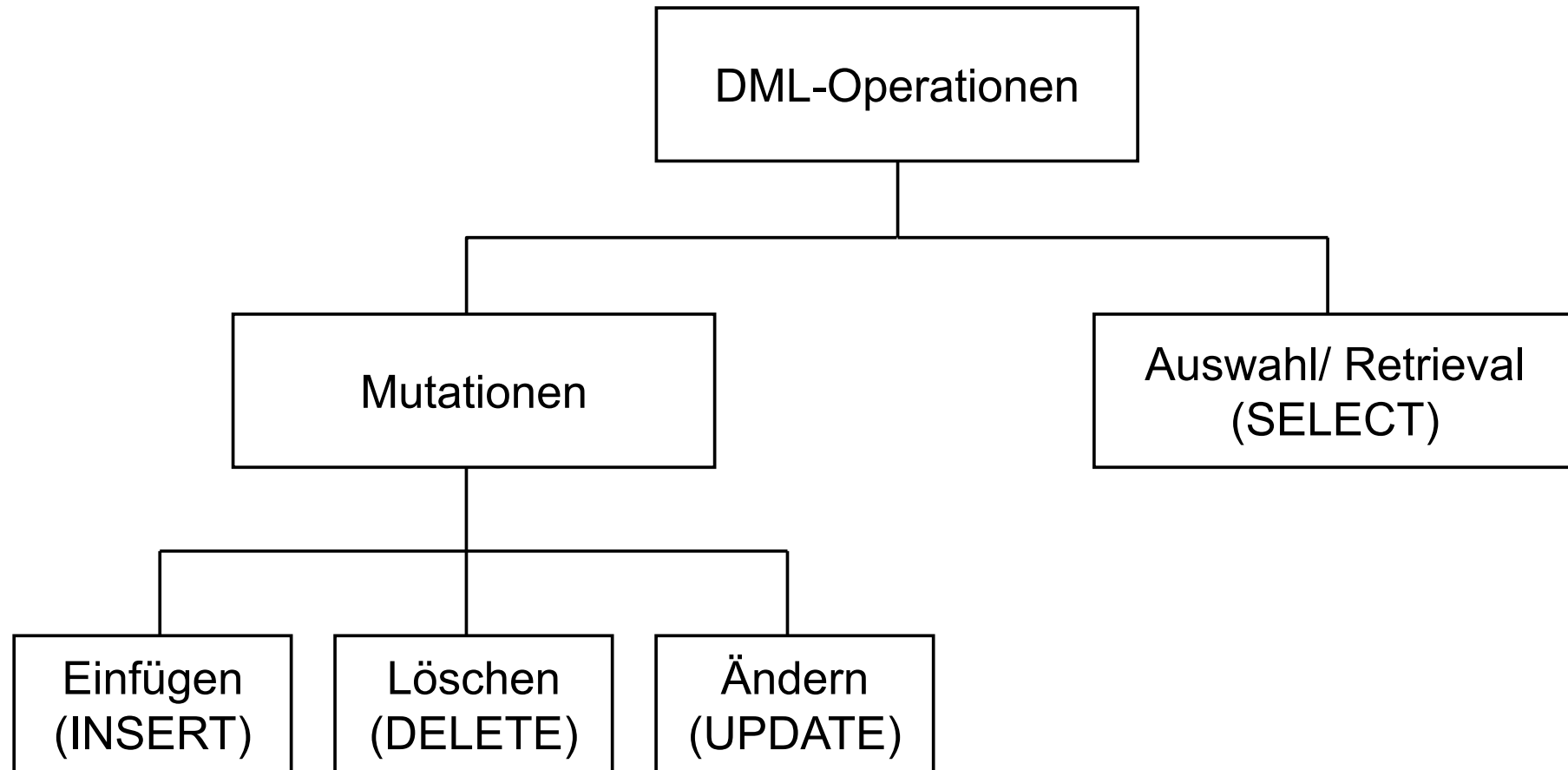
```
ALTER TABLE <tabellenname> ALTER <spaltenname> DROP NOT NULL;
```

<https://www.postgresql.org/docs/current/sql-altertable.html>

5 Löschen von Tabellen

```
DROP TABLE <tabellenname>;
```

- Berücksichtigung von Abhängigkeiten
- Sicherstellung der referentiellen Integrität



```
INSERT INTO <tabellenname> [(<spaltenname>, ...)]  
  VALUES (<werteliste>)[, (<werteliste>), ..., (<werteliste>)];
```

- Manche DBMS unterstützen nur ein Datensatz gleichzeitig
- Länge der Werteliste muss der Länge der Liste der Spaltennamen entsprechen

```
INSERT INTO <tabellenname> [(<spaltenname>, ...)]  
  AS <select befehl>;
```

5 Löschen von Datensätzen

```
DELETE FROM <tabellenname>  
[WHERE <bedingung>];
```

- Berücksichtigung von Abhängigkeiten
- Sicherstellung der referentiellen Integrität
- Potentiell kaskadiertes Löschen
⇒ ON DELETE CASCADE

- **RESTRICT**
 - ⇒ Kein Löschen / Ändern
- **CASCADE**
 - ⇒ Kaskadiertes Löschen / Ändern
- **SET NULL**
 - ⇒ Referenzen werden auf NULL gesetzt
- **SET DEFAULT**
 - ⇒ Referenzen werden auf DEFAULT-Wert gesetzt
- **NO ACTION**
 - ⇒ Keine Aktion (wie RESTRICT)

(postleitzahl PK, name)



(id, name, postleitzahl_FK)



■ ON DELETE CASCADE

<u>id</u>	name	postleitzahl_FK
1	Schmitt	10000
2	Müller	10000
3	Maier	79312

<u>postleitzahl</u>	name
10000	Musterhausen
79312	Freiburg

löschen

<u>id</u>	name	postleitzahl_FK
3	Maier	9312

<u>postleitzahl</u>	name
79312	Freiburg

automatische
Löschung in
Kindtabelle

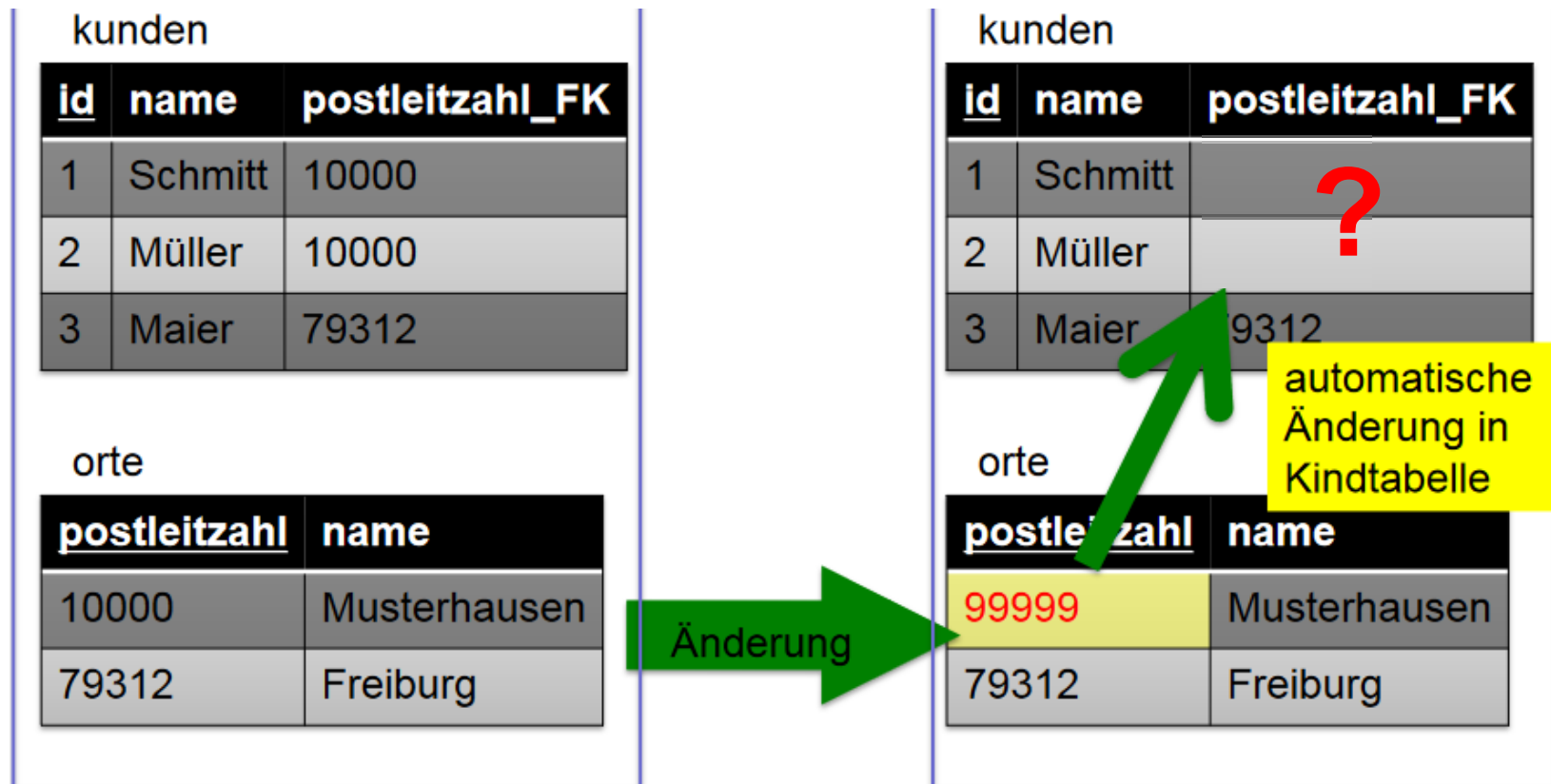
(postleitzahl PK, name)



(id, name, postleitzahl_FK)



■ ON UPDATE CASCADE



(postleitzahl_PK, name)



(id, name, postleitzahl_FK)



■ ON DELETE SET NULL

id	name	postleitzahl_FK
1	Schmitt	10000
2	Müller	10000
3	Maier	79312

postleitzahl	name
10000	Musterhausen
79312	Freiburg

löschen

id	name	postleitzahl_FK
1	Schmitt	NULL
2	Müller	NULL
3	Maier	9312

postleitzahl	name
79312	Freiburg

automatisch
NULL in
Kindtabelle

(postleitzahl_PK, name)



(id, name, postleitzahl_FK)



■ ON UPDATE SET NULL

id	name	postleitzahl_FK
1	Schmitt	10000
2	Müller	10000
3	Maier	79312

postleitzahl	name
10000	Musterhausen
79312	Freiburg

löschen

id	name	postleitzahl_FK
1	Schmitt	NULL
2	Müller	NULL
3	Maier	9312

postleitzahl	name
79312	Freiburg

automatisch
NULL in
Kindtabelle

5 Ändern von Datensätzen

```
UPDATE <tabellenname>
```

```
  SET <spaltenname> = <ausdruck> [, <spaltenname> = <ausdruck> ...]
```

```
  [WHERE <bedingung>];
```

- Berechnungen in Ausdrücken möglich
- Berücksichtigung von Abhängigkeiten
- Sicherstellung der referentiellen Integrität

```
SELECT [DISTINCT] { <tabellenname>.* |  
                    <tabellenname>.<attributname> [AS <alias>]|  
                    <ausdruck> [AS <alias>] }  
                    [, ...]  
  
FROM <tabellenname> [<alias>]  
      [LEFT JOIN | RIGHT JOIN | NATURALJOIN | JOIN] [USING | ON]  
      [, ...]  
  
[WHERE <bedingung>]  
  
[GROUP BY {<tabellenname>.<attributname> | <alias>} [, ...]]  
  
[HAVING <bedingung>]  
  
[UNION <select befehl>]  
  
[ORDER BY {<tabellenname>.<attributname> | <alias>} [ASC|DESC]  
            [, ...]]  
  
;
```

- **SELECT (Selection)**
Auswahl der Attribute/Tabellenspalten
- **DISTINCT**
Nur Anzeige unterschiedlicher Datensätze
- *****
Auswahl aller Attribute einer Tabelle
- **AS**
Umbenennung eines Attributes / eines Ausdrucks
- **<Ausdruck>**
Berechnungsvorschrift / Formel
- **FROM ... Kreuzprodukt / Join**
Auswahl der im Select-Befehl verwendeten Tabellen

- **WHERE (Restriktion)**
Definition der Bedingungen zur Auswahl der Datensätze
Ausdruck kann nur die Werte wahr oder falsch annehmen
- **GROUP BY**
Spalten, nach denen gruppiert werden soll
alle Spalten, die nicht in Aggregatfunktionen verwendet werden*
- **HAVING**
Überprüfung der Eigenschaften einer Gruppe
- **UNION**
Vereinigung der Ergebnisse zweier Select-Befehle
- **ORDER BY**
Definition der Sortierung der Ergebnisdatensätze

* Manche DBMS fordern dies nicht. ➔ Vorteile & Probleme

- >; >= ; =; =<; <; <>
- AND; OR; NOT
- IS (NOT) NULL: Überprüfung auf den Wert (NOT) NULL
- IN: Mengenoperator
- LIKE: Vergleichsoperator für Zeichenketten
 - _: Ein beliebiges Zeichen
 - %: beliebige Zeichenfolge
- BETWEEN ... AND : Intervallangabe

5 Verbund / Join

- Attributvergleich in WHERE-Bedingung
- Unterabfragen
- UNION
- Spezielle Joins
 - LEFT JOIN
 - RIGHT JOIN
 - NATURAL JOIN
- Join-Bedingung
 - ... JOIN ... USING (Attribut)
 - ... JOIN ... ON (Bedingung)

5 Aggregatfunktionen

- COUNT
- SUM
- MIN
- MAX
- AVG

(ANr, Bezeichnung,
Zusatzkosten)

Ausstattung

Normal

(HoNr, ZNr, ANr,
Anzahl)

(HoNr, HName, Ort,
Straße, HausNr, PLZ)

Hotel

(HoNr, ZNr, ZName,
Kapazität, Preis)

Zimmer

Zusatz

(RNr, RPos,
ANr, von, bis, Anzahl)

Gast

(GNr, Name, Vorname
Ort, PLZ, Straße, HausNr)

Reser-
vierung

(RNr, Datum,
HoNr, GNr)

RPosition

(RNr, RPos, von, bis,
HoNr, ZNr)

Feedbacksystem 

≡ SQL Playground 

Meine Kurse

Alle Kurse

Benutzer

Feedback App

SQL Playground

Analyseplattform

Ergebnisse

Ergebnis Nr. 1 



<https://feedback.mni.thm.de>

New_Query 

Steuerung

DB Vorlagen Do-Working

Ausgewählte DB: *Standard Datenbank*

Vorlagen
Hotel, Schema + Daten

Einfügen

Bearbeiten

DB Schema

Tabellen

Views

Stored Procedures

Triggers

5 SQL-Query im Browser ausführen

Feedbacksystem

Meine Kurse

Alle Kurse

Benutzer

Feedback App

SQL Playground

Analyseplattform

SQL Playground

Ergebnisse

Ergebnis Nr. 1

honr	hname	ort	straße
1036	Ringhotel	Luenen	Kurt-Schumacher
34	Haus Schmuelling	Bergkamen	Landwehrstrasse
135	City Hotel	Dortmund	Auf dem Wall
843	Maritim	Stralsund	Strandgasse
428	Riss	Chemnitz	An der Erzkuhle
522	Bohne	Kakau b. Bitterfeld	Zur Plantage
132	Gruene Bogenschuetze	Geseke	An der Abtei
625	Hotel Adlon	Dortmund	Lilienthalstrasse

9 of

New_Query

1 select * from hotel;

Steuerung

DB Vorlagen No-Working

Version: PostgreSQL 14

Select Database Standard Datenbank

Löschen Neue DB

Wechseln in den privaten Modus

DB Schema

Tabellen

ausstattung

gast

hotel

normal

- Alle Hotels mit allen Attributen

```
SELECT *  
FROM hotel;
```

- Alle Hotels mit Hotelnamen und Orten

```
SELECT hname, ort  
FROM hotel;
```

- Hotels aus Gießen

```
SELECT hname  
FROM hotel  
WHERE ort = 'Gießen';
```

- Alle Hotels mit ihren Zimmern

```
SELECT hotel.honr, hotel.hname, zimmer.znr
FROM hotel, zimmer
WHERE hotel.honr = zimmer.honr;
```

- Alle Hotels mit ihren Zimmern mit Alias-Verwendung

```
SELECT h.honr, h.hname, z.znr
FROM hotel h, zimmer z
WHERE h.honr = z.honr;
```

- Alle Ausstattungen, die in Hotels verwendet werden

```
SELECT h.honr, h.hname, n.anr, a.bezeichnung
FROM hotel h, normal n, ausstattung a
WHERE h.honr = n.honr and n.anr = a.anr;
```

Joins sind effizienter als ein Verbund mit where.

- Alle Hotels mit ihren Zimmern

```
SELECT h.honr, h.hname, z.znr
FROM hotel h JOIN zimmer z ON h.honr = z.honr;
```

```
SELECT h.honr, h.hname, z.znr
FROM hotel h JOIN zimmer z USING (honr);
```

- Alle Ausstattungen, die in Hotels verwendet werden

```
SELECT h.honr, h.hname, n.anr, a.bezeichnung
FROM hotel h JOIN normal n ON h.honr = n.honr
      JOIN ausstattung a ON n.anr = a.anr;
```

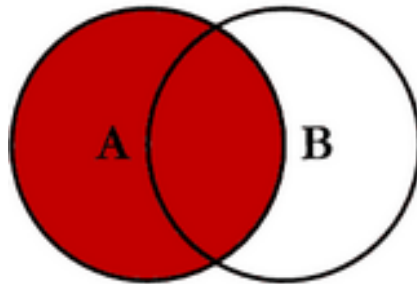
```
SELECT h.honr, h.hname, n.anr, a.bezeichnung
FROM hotel h JOIN normal n USING (honr)
      JOIN ausstattung a USING (anr);
```

Joins sind effizienter als ein Verbund mit where.

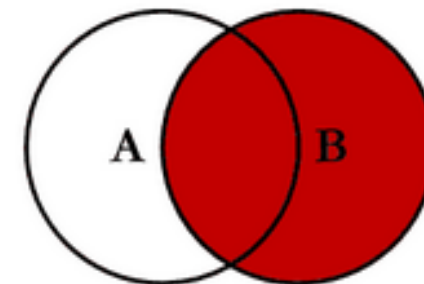
- Auflistung aller Gäste mit ihren Reservierungen, auch wenn diese keine Reservierungen haben

```
SELECT g.gnr, g.name, r.rnr  
FROM gast g LEFT JOIN reservierung r USING (gnr);
```

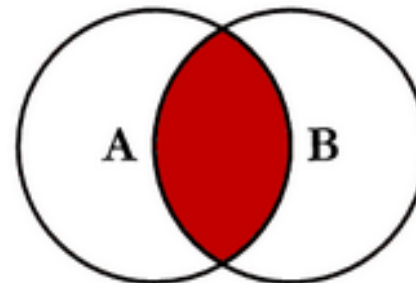

SQL JOINS



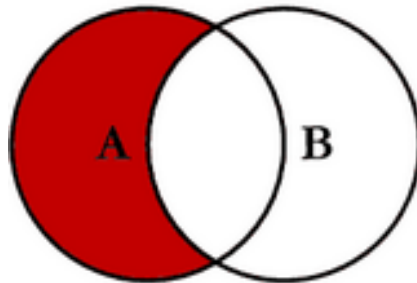
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



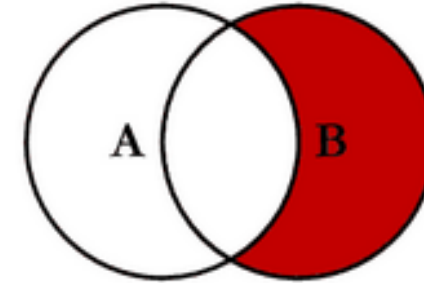
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



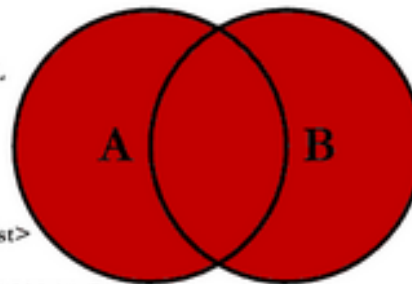
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



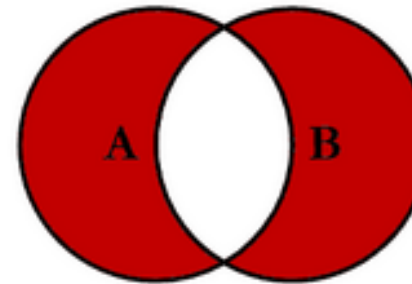
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

5 Unterdrückung gleicher Datensätze

- Alle verschiedenen Ausstattungen, die in Hotels verwendet werden

```
SELECT DISTINCT h.honr, h.hname, n.anr, a.bezeichnung  
FROM hotel h JOIN normal n USING (honr)  
            JOIN ausstattung a USING (anr);
```

Häufiger Anfängerfehler: Es wird **DISTINCT** an Stellen verwendet, wo **GROUP BY** verwendet werden sollte.

- Alle verschiedenen Ausstattungen, die in Hotels verwendet werden, aufsteigend sortiert nach honr

```
SELECT DISTINCT h.honr, h.hname, n.anr, a.bezeichnung
FROM hotel h JOIN normal n USING (honr)
             JOIN ausstattung a USING (anr)
ORDER by h.honr ASC;
```

- Alle Zimmer mit ihrer Normalausstattung sortiert,
 - erst aufsteigend nach Hotelnummer,
 - dann absteigend nach Zimmernummer

```
SELECT z.znr, z.zname, z.honr, n.anr
FROM zimmer z JOIN normal n USING (znr, honr)
ORDER BY z.honr ASC, z.znr DESC;
```

- Alle Zimmer mit ihrer Normalausstattung mit allen Attributen

```
SELECT *  
FROM zimmer z JOIN normal n USING (znr, honr);
```

- Alle Zimmer mit ihrer Normalausstattung nur mit allen Attributen aus Zimmer

```
SELECT zimmer.*  
FROM zimmer z JOIN normal n USING (znr, honr);
```

- Alle Hotels aus Gießen

```
SELECT hname
FROM hotel
WHERE ort = 'Gießen';
```

- Alle Hotels aus Städten mit G am Anfang

```
SELECT hname
FROM hotel
WHERE ort LIKE 'G%';
```

- Alle Hotels mit einem i an zweiter Stelle
(Gießen, Dillenburg, ...)

```
SELECT hname
FROM hotel
WHERE ort LIKE '_i%';
```

5 Unterabfragen

- Gäste, die für das Hotel Adlon reserviert haben

```
SELECT g.name, g.vorname
FROM gast g JOIN reservierung r USING (gnr)
           JOIN rposition rp USING (rnr)
           JOIN hotel h ON h.honr = rp.honr
WHERE h.hname='Hotel Adlon';
```

- Gäste, die noch nie für das Hotel Adlon reserviert haben

```
SELECT g.name, g.vorname
FROM gast g
WHERE g.gnr NOT IN (
    SELECT g.gnr
    FROM gast g JOIN reservierung r USING (gnr)
               JOIN rposition rp USING (rnr)
               JOIN hotel h ON h.honr = rp.honr
    WHERE h.hname='Hotel Adlon')
;
```

- Welche Hotels liegen im gleichen Ort?

```
SELECT a.hname, b.hname, a.ort  
FROM hotel a, hotel b  
WHERE a.ort=b.ort;
```

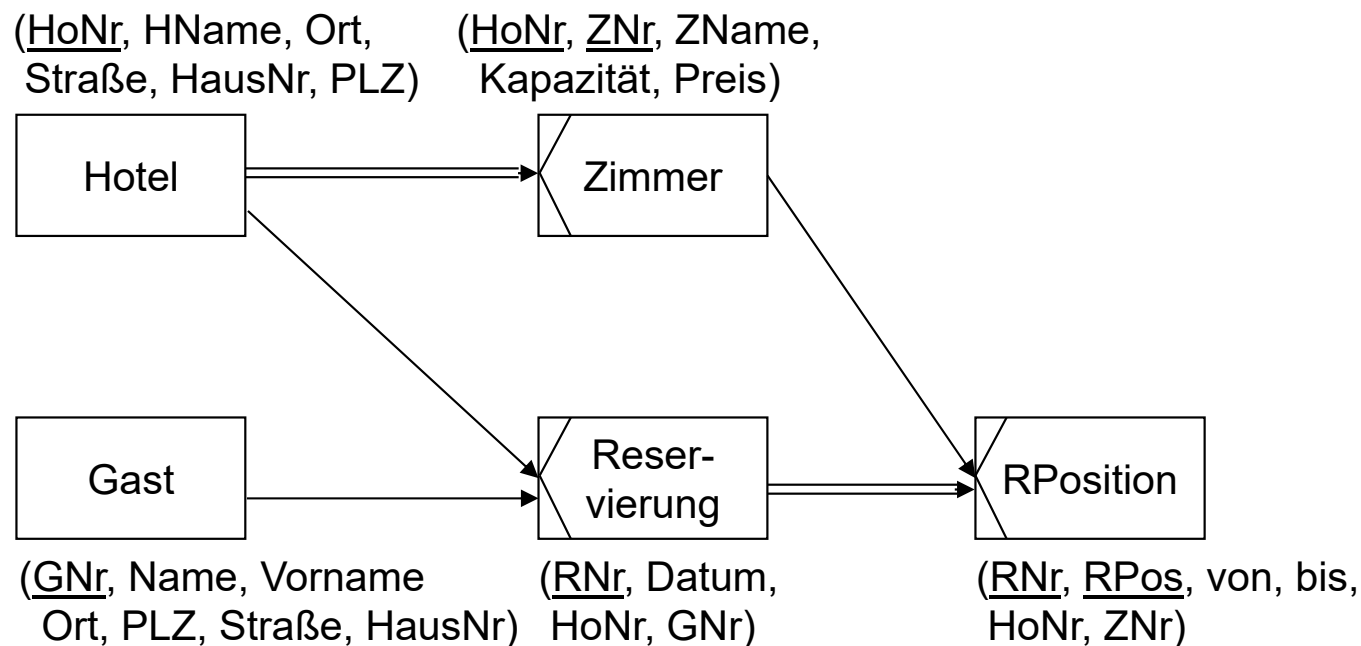
- Welche Hotels liegen im gleichen Ort (ohne gleiche Hotels)?

```
SELECT a.hname, b.hname, a.ort  
FROM hotel a, hotel b  
WHERE a.ort=b.ort AND a.honr<>b.honr;
```

5 Mehrfachverwendung einer Tabelle

- Welcher Gast hat in welchem Hotel (entgegennehmendes Hotel) für welches Hotel reserviert?

```
SELECT g.gnr, g.name, h.honr, h.hname, e.honr, e.hname
FROM gast g JOIN reservierung r USING (gnr)
      JOIN rposition rp USING (rnr)
      JOIN hotel e ON e.honr = r.honr
      JOIN hotel h ON h.honr = rp.honr;
```



5 Anwendungsbeispiel – Zur Erinnerung

(ANr, Bezeichnung,
Zusatzkosten)

Ausstattung

Normal

(HoNr, ZNr, ANr,
Anzahl)

(HoNr, HName, Ort,
Straße, HausNr, PLZ)

Hotel

(HoNr, ZNr, ZName,
Kapazität, Preis)

Zimmer

Zusatz

(RNr, RPos,
ANr, von, bis, Anzahl)

Gast

(GNr, Name, Vorname
Ort, PLZ, Straße, HausNr)

Reser-
vierung

(RNr, Datum,
HoNr, GNr)

RPosition

(RNr, RPos, von, bis,
HoNr, ZNr)

- Auflistung der Kosten aller Reservierungspositionen für jeden Kunden

```
SELECT g.gnr, g.name, z.preis*(rp.bis-rp.von)
FROM gast g JOIN reservierung r USING (gnr)
      JOIN rposition rp USING (rnr)
      JOIN zimmer z ON z.honr = rp.honr AND z.znr = rp.znr;
```

- Auflistung der Kosten aller Reservierungspositionen für jeden Kunden mit Spaltenbenennung

```
SELECT g.gnr, g.name, z.preis*(rp.bis-rp.von) AS kosten
FROM gast g JOIN reservierung r USING (gnr)
      JOIN rposition rp USING (rnr)
      JOIN zimmer z ON z.honr = rp.honr AND z.znr = rp.znr;
```

1) Bilden Sie das **Kartesisches Produkt** folgender Tabellen (Relationen):

Name	Ort
Meier	Künzelsau
Schmidt	Öhringen
Schulz	Heilbronn

Typ	Hersteller	Farbe
Corsa	Opel	grün
Golf	VW	rot

Gegeben sei folgendes Datenbankschema, in dem Informationen über Orte, Filme und das aktuelle Programm gespeichert sind:

Kino (**Orte**(Kino, Strasse, PLZ),
Filme(Titel, Regie, Schauspieler),
Programm(Kino, Titel, Zeit))

2.) Was ist das Ergebnis der folgenden Ausdrücke:

a.) $R_1 = \pi_{\text{Regie}}(\sigma_{\text{Schauspieler}=\text{"Johnny Depp"}}(\text{Filme}))$

b.) $R_2 = \pi_{\text{Schauspieler}}(\text{Filme} \bowtie \text{Programm})$

c.) $R_3 = \pi_{\text{Titel}}(\sigma_{\text{Schauspieler}=\text{"Johnny Depp"}}(\text{Filme}))$

d.) $R_4 = R_3 - \pi_{\text{Titel}}(\text{Programm})$

e.) $R_5 = \pi_{\text{Kino}, \text{PLZ}}(\text{Orte} \bowtie_{\text{Kino}=\text{Titel}} R_4)$

Gegeben sei folgendes Datenbankschema, in dem Informationen über Orte, Filme und das aktuelle Programm gespeichert sind:

Kino (**Orte**(Kino, Strasse, PLZ),
 Regisseure(Titel, Regie),
 Filme(Titel, Nr, Schauspieler),
 Programm(Kino, Titel, Zeit))

- 3.) Wie sehen die Anfragen in relationaler Algebra aus?
- a.) Welche Regisseure gibt es über alle Filme?
 - b.) Welche Filmtitel gibt es mit dem Schauspieler Jürgen Vogel?
 - c.) Welche Regisseure haben mit Jürgen Vogel gearbeitet?
 - d.) Liste der Kinos (Name + PLZ) mit deren Filmen?
 - e.) In welchen Kinos (Name + PLZ) laufen Filme mit Nora Tschirner?
 - f.) In welchen Filmen (Titel) spielt N.Tschirner und J.Vogel mit?