Relationale Datenbanken

- ▷ Relationen
- ▷ Datenbankspezifikation
- ▷ Datenerfassung
- ▷ Einfache Anfragen
- ▷ Mehrtabellenanfragen
- ▷ Datenaktualisierung
- ▷ Datenmodellierung
- ▷ Relationenalgebra

Relationen

Relationen sind Beziehungen zwischen Objekten, Ereignissen und abstrakten Ideen und Begriffen.

- Objekte und Begriffe treten niemals isoliert auf.
- Vollständige Erfassung (Verstehen) erst durch Berücksichtigung der Gesamtheit seiner Relationen zu anderen Objekten und Begriffen.
- In realen Welten: *Weitgehende* Erfassung ... Berücksichtigung der *wesentlichen* Relationen ...

Beispiele für zweistellige Relationen:

- A "ist schwerer als" B
- A "ist kleiner als" B
- A "passierte zeitlich nach B"
- A "ist Voraussetzung für" B

Allgemeiner Fall: n-stellige Relationen ($n \geq 1$)

- A "liegt zwischen" B "und" C
- A "transportiert" B "von" C "nach" D

Spezialfall: einstellige Relationen (Eigenschaften)

- A "ist groß"
- A "ist blau"
- A "ist ein Haus"

Relationen

Relation in M: Teilmenge des Kreuzproduktes der Menge M mit sich selbst.

$$R^{(2)} \subseteq M \times M = M^2$$

Kreuzprodukt (Cartesisches Produkt) zweier Mengen A und B: Menge aller geordneten Paare (2-Tupel), die sich aus den Elementen von A und B bilden lassen:

$$A \times B = \{(x, y) | x \in A, y \in B\}$$

Domäne einer Relation: Menge M, über der die Relation definiert ist.

Beispiele:

$$A = \{\circ, \bigcirc, \large\bigcirc\}$$

$$\begin{array}{lccc} A \times A = \{ & (\circ,\circ), & (\bigcirc,\circ), & (\large\bigcirc,\circ), \\ & (\circ,\bigcirc), & (\bigcirc,\bigcirc), & (\large\bigcirc,\bigcirc), \\ & (\circ,\large\bigcirc), & (\bigcirc,\large\bigcirc), & (\large\bigcirc,\large\bigcirc)\} \end{array}$$

"ist gleich groß" (in A) $= \{(\circ,\circ), (\bigcirc,\bigcirc), (\large\bigcirc,\large\bigcirc)\} \subseteq A^2$

"ist kleiner als" (in A) $= \{(\circ,\bigcirc), (\circ,\large\bigcirc), (\bigcirc,\large\bigcirc)\} \subseteq A^2$

Relationen

Beispiele:

$$\text{"ist größer oder gleich" (in } A) = \{(\circ,\circ), (\bigcirc,\circ), ({\Large\bigcirc},\circ),$$
$$(\bigcirc,\bigcirc), ({\Large\bigcirc},\bigcirc), ({\Large\bigcirc},{\Large\bigcirc})\}$$
$$\subseteq A^2$$

$P = \{\text{Anna,Hanna,Minna}\}$

$$\text{"ist Mutter von" (in } P) = \{(\text{Hanna,Anna}), (\text{Minna,Hanna})\} \subseteq P^2$$

Tabellendarstellung:

ist Mutter von

Mutter	Tochter
Hanna	Anna
Minna	Hanna

Relationen

Verallgemeinerung (1): Relationen über zwei verschiedenen Domänen

$$R^{(2)} \subseteq A \times B, \quad A \neq B$$

Beispiele:

$B = \{\bigcirc, \square, \triangle\}, \quad C = \{\bullet, \blacksquare, \blacktriangle\}$

$$B \times C = \{ \begin{array}{lll} (\bigcirc, \bullet), & (\square, \bullet), & (\triangle, \bullet), \\ (\bigcirc, \blacksquare), & (\square, \blacksquare) & (\triangle, \blacksquare), \\ (\bigcirc, \blacktriangle), & (\square, \blacktriangle), & (\triangle, \blacktriangle)\} \end{array}$$

"ist kongruent" (über B und C) $= \{(\bigcirc, \bullet), (\square, \blacksquare)(\triangle, \blacktriangle)\} \subseteq B \times C$

$P = \{$Anna,Hanna,Minna$\}, \quad N = \{0, 1, 2, \ldots\}$

"Alter" (über P und N) $=$ {(Anna,12),
(Hanna,35)
(Minna,63)} $\subseteq P \times N$

Alter

Person	Alter
Anna	12
Hanna	35
Minna	63

Relationen

Verallgemeinerung (2): Relationen über mehr als zwei (verschiedenen) Domänen

$$R^{(n)} \subseteq \underbrace{A \times \ldots \times Z}_{n \text{ Faktoren}}$$

Beispiele:

$P = \{$Anna,Hanna,Minna$\}$, $N = \{0, 1, 2, \ldots\}$
$F = \{$ledig, verheiratet, geschieden, verwitwet$\}$

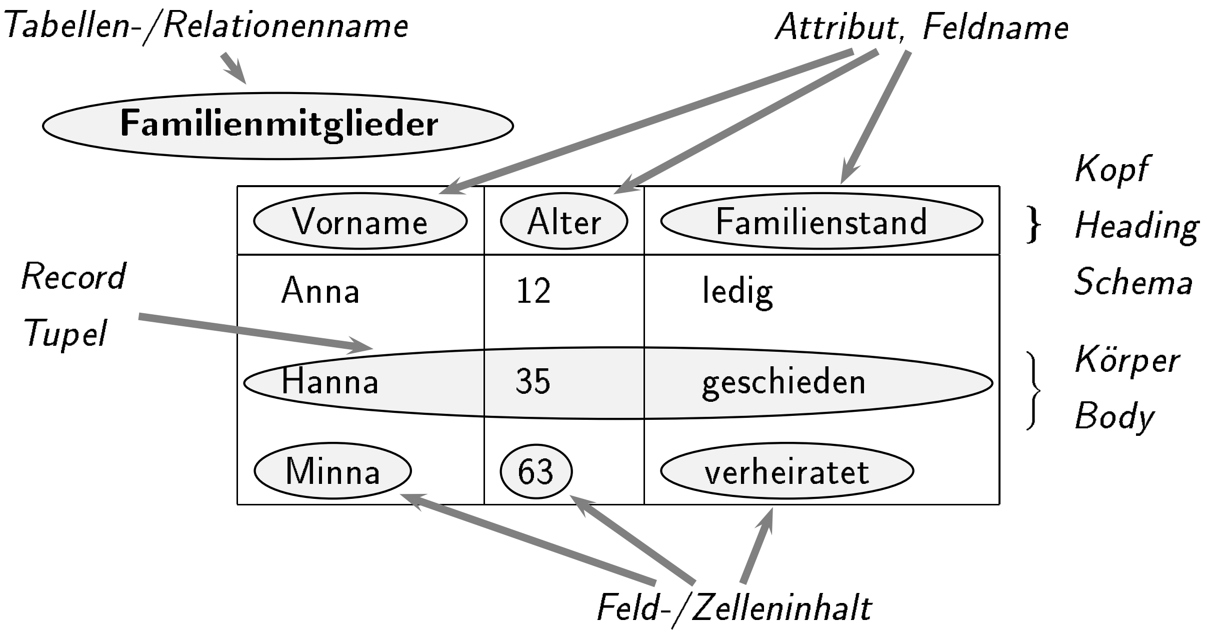
"Familienmitglieder" (über P, N und F) $=$ $\{$(Anna,12,ledig),
(Hanna,35,geschieden)
(Minna,63,verheiratet)$\}$
$\subseteq P \times N \times F$

Familienmitglieder

Vorname	Alter	Familienstand
Anna	12	ledig
Hanna	35	geschieden
Minna	63	verheiratet

Relationen

Relationen als Datenbanktabellen:



- Anzahl der Zeilen ist endlich aber beliebig
- Anordnung der Zeilen ist beliebig
  - Relation ist eine Menge!
- Anordnung der Spalten ist beliebig
  - Interpretation ist durch den Tabellenkopf gegeben
  - Unterschied zum mathematischen Tupelbegriff

Datenbankspezifikation

Relationale Datenbank ist eine Menge von Relationen (Tabellen).

- Identifikation der Tabellen durch Tabellennamen
- Manipulation der Tabellen durch SQL-Klauseln (Structured Query Language)

Tabellenstruktur wird durch den Tabellenkopf beschrieben.

Stelligkeit: Anzahl der Spalten

Attribute: Identifikation des Spalteninhalts

Domänenrestriktionen: Beschränkungen für zulässige Spalteninhalte
$\rightarrow$ Datentypen

`create table` $\langle$Tabellenname$\rangle$ $\langle$Tabellenkopf$\rangle$

$\langle$Tabellenkopf$\rangle$ $\rightarrow$ ($\langle$Spaltenbeschreibung$\rangle$ $\{$, $\langle$Spaltenbeschreibung$\rangle$ $\}$)
$\langle$Spaltenbeschreibung$\rangle$ $\rightarrow$ $\langle$Spaltenname$\rangle$ $\langle$Datentyp$\rangle$ [$\langle$Parameter$\rangle$]
$\langle$Spaltenname$\rangle$ $\rightarrow$ $\langle$String$\rangle$

`create table` $\langle$Tabellenname$\rangle$
($\langle$Spaltenname$\rangle$ $\langle$Datentyp$\rangle$ [$\langle$Parameter$\rangle$]
$\{$, $\langle$Spaltenname$\rangle$ $\langle$Datentyp$\rangle$ [$\langle$Parameter$\rangle$] $\}$)

Datentypen

Zeichenketten fester Länge: `char(`⟨Länge⟩`)`

Zeichenketten variabler Länge: `text(`⟨Länge⟩`)`

⟨Länge⟩ ist eine natürliche Zahl, die die normale Länge einer typischen Datenbankeintragung angibt.

Längere Ketten werden durch einen Überlaufpuffer verwaltet.

Integer (ganze Zahl): `int`

Gleitkommazahl: `real`

Beispiele:

```
create table familienmitglieder
    ( vorname       char(20),
      alter         int,
      familienstand char(11)  )

+---------------------------+-------+--------------------+
| vorname                   | alter | familienstand      |
+---------------------------+-------+--------------------+
|                           |       |                    |
|                           |       |                    |
+---------------------------+-------+--------------------+

create table obj
    ( id    int,
      type  char(10),
      str   char(20),
      no    int,
      jahr  int       )

+------+------------+----------------------+------+------+
| id   | type       | str                  | no   | jahr |
+------+------------+----------------------+------+------+
|      |            |                      |      |      |
|      |            |                      |      |      |
+------+------------+----------------------+------+------+
```

Datenerfassung

Einfügen neuer Tupel in eine Tabelle

$\rightarrow$ Spezifikation von Feldinhalten

Konstanten:

Zeichenketten: '⟨Zeichen⟩ {⟨Zeichen⟩}'

```
'Maier'
'Otto Raffke GmbH'
```

Ganze Zahlen: [⟨Vorzeichen⟩] ⟨Ziffer⟩ {⟨Ziffer⟩}

```
123
+5
-9999999
```

Gleitkommazahlen: ⟨Mantisse⟩ [⟨Exponent⟩]

Mantisse: [⟨Vorzeichen⟩] (⟨Ziffer⟩ {⟨Ziffer⟩} ['.' {⟨Ziffer⟩}]
| '.' ⟨Ziffer⟩ {⟨Ziffer⟩})

Exponent: ('E' | 'e') ⟨Vorzeichen⟩ ⟨Ziffer⟩ {⟨Ziffer⟩}

```
1536
-123.359
2357.
-.017
135.4E-23
3.7935e+6
```

Datenerfassung

INSERT-Klausel:

`insert` ⟨wo⟩ ⟨was⟩

Einfügen kompletter Tupel

`insert into` ⟨Tabellenname⟩
    `values` (⟨Wert⟩ {, ⟨Wert⟩})

- Anzahl und Reihenfolge der spezifizierten Werte muß genau der Tabellenstruktur entsprechen.

Beispiel:

```
insert into obj
    values (1,'EFH','Gaertnerstr',15,1965)

+------+-----------+-------------------+-----+------+
| id   | type      | str               | no  | jahr |
+------+-----------+-------------------+-----+------+
| 1    | EFH       | Gaertnerstr       | 15  | 1965 |
|      |           |                   |     |      |
+------+-----------+-------------------+-----+------+
```

Datenerfassung

Einfügen unvollständiger Tupel

`insert into` ⟨Tabellenname⟩ (⟨Spaltenname⟩ {, ⟨Spaltenname⟩ })
`values` (⟨Wert⟩ {, ⟨Wert⟩})

- Die Anzahl der Spaltennamen muß gleich der Anzahl der spezifizierten Werte sein.

```
insert into obj (id,type,jahr)
   values (2,'EFH',1965)
```

```
+------+------------+----------------------+------+------+
| id   | type       | str                  | no   | jahr |
+------+------------+----------------------+------+------+
| 1    | EFH        | Gaertnerstr          | 15   | 1965 |
| 2    | EFH        | NULL                 | NULL | 1943 |
+------+------------+----------------------+------+------+
```

- Jedes `insert`-Kommando fügt eine neue Zeile in die Tabelle ein!

Einfache Anfragen

SELECT-Klausel:

select ⟨welche Spalte⟩ ⟨wo⟩ ⟨welche Zeile⟩ ⟨wie⟩

Ausgabe einer kompletten Tabelle

select * from ⟨Tabellenname⟩

Beispiel:

```
select * from obj

+------+-----------+--------------------+------+------+
| id   | type      | str                | no   | jahr |
+------+-----------+--------------------+------+------+
| 1    | EFH       | Gaertnerstr        | 15   | 1965 |
| 2    | EFH       | Bahnhofsstr        | 27   | 1943 |
| 3    | EFH       | Bahnhofsstr        | 29   | 1955 |
| 4    | MFH       | Bahnhofsstr        | 28   | 1991 |
| 5    | Bahnhof   | Bahnhofsstr        | 30   | 1901 |
| 6    | Kaufhaus  | Bahnhofsstr        | 26   | 1997 |
| 7    | EFH       | Gaertnerstr        | 17   | 1982 |
+------+-----------+--------------------+------+------+
```

Das Ergebnis der Anwendung einer SELECT-Klausel auf eine Tabelle ist wieder eine Tabelle:

$$\text{SELECT: } T \rightarrow T$$

Einfache Anfragen

Auswahl bestimmter Spalten

select ⟨Feldname⟩, ..., ⟨Feldname⟩ from ⟨Tabellenname⟩

Beispiel:

```
select type,str,jahr from obj

+------------+----------------------+------+
| type       | str                  | jahr |
+------------+----------------------+------+
| EFH        | Gaertnerstr          | 1965 |
| EFH        | Bahnhofsstr          | 1943 |
| EFH        | Bahnhofsstr          | 1955 |
| MFH        | Bahnhofsstr          | 1991 |
| Bahnhof    | Bahnhofsstr          | 1901 |
| Kaufhaus   | Bahnhofsstr          | 1997 |
| EFH        | Gaertnerstr          | 1982 |
+------------+----------------------+------+
```

```
select str,no from obj

+----------------------+------+
| str                  | no   |
+----------------------+------+
| Gaertnerstr          | 15   |
| Bahnhofsstr          | 27   |
| Bahnhofsstr          | 29   |
| Bahnhofsstr          | 28   |
| Bahnhofsstr          | 30   |
| Bahnhofstr           | 26   |
| Gaertnerstr          | 17   |
+----------------------+------+
```

Einfache Anfragen

Eliminieren von Mehrfacheintragungen

select distinct ⟨Feldname⟩, ..., ⟨Feldname⟩ from ⟨Tabellenname⟩

Beispiel:

```
select type,str from obj

+------------+----------------------+
| type       | str                  |
+------------+----------------------+
| EFH        | Gaertnerstr          |
| EFH        | Bahnhofsstr          |
| EFH        | Bahnhofsstr          |
| MFH        | Bahnhofsstr          |
| Bahnhof    | Bahnhofsstr          |
| Kaufhaus   | Bahnhofsstr          |
| EFH        | Gaertnerstr          |
+------------+----------------------+
```

Beispiel:

```
select distinct type,str from obj

+------------+----------------------+
| type       | str                  |
+------------+----------------------+
| EFH        | Gaertnerstr          |
| EFH        | Bahnhofsstr          |
| MFH        | Bahnhofsstr          |
| Bahnhof    | Bahnhofsstr          |
| Kaufhaus   | Bahnhofsstr          |
+------------+----------------------+
```

Einfache Anfragen

Sortieren der Resultate

select ⟨Feldname⟩, ..., ⟨Feldname⟩ from ⟨Tabellenname⟩
        order by ⟨feldname⟩ ⟨Richtung⟩

⟨Richtung⟩ → asc|desc

asc ist Standard

Beispiel:

```
select type,str,jahr from obj order by type

+------------+---------------------+------+
| type       | str                 | jahr |
+------------+---------------------+------+
| Bahnhof    | Bahnhofsstr         | 1901 |
| EFH        | Bahnhofsstr         | 1943 |
| EFH        | Bahnhofsstr         | 1955 |
| EFH        | Gaertnerstr         | 1965 |
| EFH        | Gaertnerstr         | 1982 |
| Kaufhaus   | Bahnhofsstr         | 1997 |
| MFH        | Bahnhofsstr         | 1991 |
+------------+---------------------+------+

select type,str,jahr from obj order by jahr desc

+------------+---------------------+------+
| type       | str                 | jahr |
+------------+---------------------+------+
| Kaufhaus   | Bahnhofsstr         | 1997 |
| MFH        | Bahnhofsstr         | 1991 |
| EFH        | Gaertnerstr         | 1982 |
| EFH        | Gaertnerstr         | 1965 |
| EFH        | Bahnhofsstr         | 1955 |
| EFH        | Bahnhofsstr         | 1943 |
| Bahnhof    | Bahnhofsstr         | 1901 |
+------------+---------------------+------+
```

Einfache Anfragen

Mehrere Sortierkriterien:

`select` ⟨Feldname⟩, ..., ⟨Feldname⟩ `from` ⟨Tabellenname⟩
`order by` ⟨feldname⟩ ⟨Richtung⟩, ..., ⟨feldname⟩ ⟨Richtung⟩

Beispiel:

```
select type,str,jahr from obj order by type, jahr desc
```

```
+------------+----------------------+------+
| type       | str                  | jahr |
+------------+----------------------+------+
| Bahnhof    | Bahnhofsstr          | 1901 |
| EFH        | Gaertnerstr          | 1982 |
| EFH        | Gaertnerstr          | 1965 |
| EFH        | Bahnhofsstr          | 1955 |
| EFH        | Bahnhofsstr          | 1943 |
| Kaufhaus   | Bahnhofsstr          | 1997 |
| MFH        | Bahnhofsstr          | 1991 |
+------------+----------------------+------+
```

Einfache Anfragen

Auswahl von Zeilen:

`select` ⟨Feldname⟩, ..., ⟨Feldname⟩ `from` ⟨Tabellenname⟩
`where` ⟨Bedingung⟩

Bedingung ist ein logischer Ausdruck, der einen Wahrheitswert liefert

Vergleichsprädikate: ⟨Feldname⟩ ⟨Operator⟩ ⟨Wert⟩

Operatoren:

Operator	Bedeutung
<	kleiner als
>	größer als
=	gleich
<=	kleiner oder gleich
>=	größer oder gleich
<>	ungleich
`like`	Zeichenkettenvergleich mit Platzhaltern
`clike`	wie `like` aber ohne groß/klein Unterscheidung
`rlike`	Zeichenkettenvergleich mit regulären Ausdrücken

Werte: Konstanten (vgl. Datenerfassung)

Einfache Anfragen

Platzhalter:

Platzhalter	Bedeutung
_	Einzelzeichen
%	Zeichensequenz
\	Escape-Zeichen

Reguläre Ausdrücke:

Zeichen	Bedeutung
.	universelles Zeichen
^	linke Verankerung
$	rechte Verankerung
[...]	Zeichenauswahl
*	Zeichenwiederholung

Logische Konnektoren:

and	Konjunktion
or	Disjunktion
(...)	Klammerung

Einfache Anfragen

Beispiele:

```
select * from obj where type='EFH'
```

id	type	str	no	jahr
1	EFH	Gaertnerstr	15	1965
2	EFH	Bahnhofsstr	27	1943
3	EFH	Bahnhofsstr	29	1955
7	EFH	Gaertnerstr	17	1982

```
select * from obj where type='EFH' and jahr>1970
```

id	type	str	no	jahr
7	EFH	Gaertnerstr	17	1982

```
select * from obj where str>'Einbahnstrasse'
```

id	type	str	no	jahr
1	EFH	Gaertnerstr	15	1965
7	EFH	Gaertnerstr	17	1982

Einfache Anfragen

```
select * from obj where str like 'Bahnhofs%'
```

id	type	str	no	jahr
2	EFH	Bahnhofsstr	27	1943
3	EFH	Bahnhofsstr	29	1955
4	MFH	Bahnhofsstr	28	1991
5	Bahnhof	Bahnhofsstr	30	1901
6	Kaufhaus	Bahnhofsstr	26	1997

```
select * from obj where str clike 'bahnhofs%'
```

id	type	str	no	jahr
2	EFH	Bahnhofsstr	27	1943
3	EFH	Bahnhofsstr	29	1955
4	MFH	Bahnhofsstr	28	1991
5	Bahnhof	Bahnhofsstr	30	1901
6	Kaufhaus	Bahnhofsstr	26	1997

```
select * from obj where type rlike '[EM]FH'
```

id	type	str	no	jahr
1	EFH	Gaertnerstr	15	1965
2	EFH	Bahnhofsstr	27	1943
3	EFH	Bahnhofsstr	29	1955
4	MFH	Bahnhofsstr	28	1991
7	EFH	Gaertnerstr	17	1982

Einfache Anfragen

Auswertungsfunktionen:

`select sum(`⟨Feldname⟩`) from` ⟨Tabellenname⟩

`select count(*) from` ⟨Tabellenname⟩ `where` ⟨Bedingung⟩

Funktion	Bedeutung	Bemerkung
`count`	Anzahl der Werte in einer Spalte	`distinct` ist obligat
`count(*)`	Anzahl aller Zeilen	einschl. mögl. Duplikate
`sum`	Summe der Werte in einer Spalte	
`avg`	Mittelwert der Werte einer Spalte	
`max`	Maximum der Werte einer Spalte	
`min`	Minimum der Werte einer Spalte	

in mSQL nicht unterstützt

Mehrtabellenanfragen

```
select * from obj
```

id	type	str	no	jahr
1	EFH	Gaertnerstr	15	1965
2	EFH	Bahnhofsstr	27	1943
3	EFH	Bahnhofsstr	29	1955
4	MFH	Bahnhofsstr	28	1991
5	Bahnhof	Bahnhofsstr	30	1901
6	Kaufhaus	Bahnhofsstr	26	1997
7	EFH	Gaertnerstr	17	1982

```
create table bew
    ( id      int,
      obj_id  int,
      verk    char(15),
      kauf    char(15),
      preis   real,
      datum   char(8)    )
insert into bew values (1,1,'Mueller','Meier',450000,'97.01.01')
...
select * from bew
```

id	obj_id	verk	kauf	preis	datum
1	1	Mueller	Meier	450000	97.01.01
2	3	Schulze	Schneider	560000	88.12.13
3	3	Schneider	Mueller	615000	96.12.01
4	5	Bund	Piepenbrink	3500000	91.06.01

Mehrtabellenanfragen

Verknüpfung mehrerer Tabellen durch **Links**

- Ein Link verknüpft genau zwei verschiedene Spalten.
- Der Link wird durch gleiche Werte in den ausgewählten Spalten etabliert.

Wer ist der Käufer des Hauses Gärtnerstraße 15?

→ Die Spalten mit den Identifikationsnummern werden verknüpft.
→ Der Link wird durch identische Objektnummern hergestellt

id	type	str	no	jahr
1	EFH	Gaertnerstr	15	1965
2	EFH	Bahnhofsstr	27	1943
...	...	...	...	...

id	obj_id	verk	kauf	preis	datum
1	1	Mueller	Meier	450000	97.01.01
2	3	Schulze	Schneider	560000	88.12.13
...	...	...	...	...	...

Mehrtabellenanfragen

Join zweier durch einen Link verbundener Tabellen ist eine neue Tabelle, die aus den Spalten der beiden Ursprungstabellen gebildet wird:

1. Ermittle alle Paare von Tupeln, die in den verbundenen Spalten identische Werte besitzen
2. Bilde alle Paare von Tupeln mit identischen Werten in den verbundenen Spalten

Das Resultat eines Joins zweier Tabellen ist wieder eine Tabelle:

$$\text{JOIN: } \boldsymbol{T} \times \boldsymbol{T} \rightarrow \boldsymbol{T}$$

Mehrtabellenanfragen

Join:

1	A	
2	B	
3	C	
4	B	
5	F	
6	B	
7	A	
8	G	

m

1		B	
2		A	
3		E	
4		B	
5		D	
6		F	
7		A	

n

1	A		2		A	
1	A		7		A	
2	B		1		B	
2	B		4		B	
4	B		1		B	
4	B		4		B	
5	F		6		F	
6	B		1		B	
6	B		4		B	
7	A		2		A	
7	A		7		A	

$m + n$

Join: Welche Immobilien haben den Besitzer gewechselt?

id	type	str	no	jahr	id	obj_id	verk	kauf	preis	datum
1	EFH	Gaertnerstr	15	1965	1	1	Mueller	Meier	450000	97.01.01
3	EFH	Bahnhofsstr	29	1955	2	3	Schulze	Schneider	560000	88.12.13
3	EFH	Bahnhofsstr	29	1955	3	3	Schneider	Mueller	615000	96.12.01
5	Bahnhof	Bahnhofsstr	30	1901	4	5	Bund	Piepenbrink	3500000	91.06.01

```
select ⟨Tabellenname⟩.⟨Feldname⟩, ..., ⟨Tabellenname⟩.⟨Feldname⟩
       from ⟨Tabellenname1⟩, ⟨Tabellenname2⟩
       where ⟨Tabellenname1⟩.⟨Feldname⟩ = ⟨Tabellenname2⟩.⟨Feldname⟩
```

```
select * from obj, bew where obj.id = bew.obj_id
```

Mehrtabellenanfragen

Beispiel: Wer hat welche Häuser gekauft?

```
select bew.kauf,obj.str,obj.no from bew,obj
      where bew.obj_id = obj.id
```

```
+-----------------+----------------------+------+
| kauf            | str                  | no   |
+-----------------+----------------------+------+
| Meier           | Gaertnerstr          | 15   |
| Schneider       | Bahnhofsstr          | 29   |
| Mueller         | Bahnhofsstr          | 29   |
| Piepenbrink     | Bahnhofsstr          | 30   |
+-----------------+----------------------+------+
```

Beispiel: Welche Häuser, die älter als 40 Jahre sind wurden verkauft?

```
select obj.str,obj.no,obj.jahr from bew,obj
      where bew.obj_id = obj.id and obj.jahr < 1957
```

```
+----------------------+------+------+
| str                  | no   | jahr |
+----------------------+------+------+
| Bahnhofsstr          | 29   | 1955 |
| Bahnhofsstr          | 29   | 1955 |
| Bahnhofsstr          | 30   | 1901 |
+----------------------+------+------+
```

Beispiel: Wer hat das Haus Gärtnerstraße 15 gekauft?

```
select bew.kauf from bew,obj
      where bew.obj_id = obj.id
            and obj.str rlike 'G' and obj.no = 15
```

```
+-----------------+
| kauf            |
+-----------------+
| Meier           |
+-----------------+
```