Angestellter

AngNr	Name	Wohnort	Beruf	AbtNr
112	Müller	Erfurt	Ingenieur	3
205	Winter	Zwickau	Programmierer	3
117	Rüllich	Weimar	Hundezüchter	5
198	Schumann	Jena	Kaufmann	4

Projekt

ProNr	Titel	Inhalt	Leiter
27	Pkw2000	blabla...	205
16	Wankel99	blabla...	117
84	Trabi00	blabla...	117

Mitarbeit

ProNr	AngNr	Prozent
27	112	100
27	198	70
16	198	20

- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit))
- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{PJ}_{(AngNr)}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit)))
- $\mathsf{PJ}_{(Name)}$($\mathsf{SL}_{ProNr=27}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))
- $\mathsf{SL}_{ProNr=27}$($\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))

Vorsicht! Letzter Ausdruck syntaktisch falsch.

SQL — Teil 2

SELECT
Projektion
Selektion
Vereinigung, Schnitt, Differenz
Verbund
Komplexer SELECT-Ausdruck

Fahren fort mit SQL Befehlen. Bilden Relationenalgebra auf SQL ab.

So Umsetzung von Anfragen an die DB (bzw. Tabellen) möglich.

SELECT

Hatten schon

```
SELECT * FROM table_name;
```

Wählt *alle* Datensätze der `table_name` aus.

Allgemeiner (in SQLite)

```
SELECT [DISTINCT] select_heading
  FROM source_tables
  WEHRE filter_expression
  GROUP BY grouping_expression
    HAVING filter_expression
  ORDER BY ordering_expression
  LIMIT count
    OFFSET count;
```

Und wir erkennen, dass das ein wenig komplizierter wird . . .

Spielen die Relationenalgebra Operationen durch. Beispieltabellen dazu

Verkäufer1

Verkäufer	Produkt	Käufer
Meier	Hose	Schmidt
Müller	Rock	Schmidt
Meier	Hose	Schulz

Verkäufer2

Verkäufer	Produkt	Käufer
Müller	Hemd	Schmidt
Müller	Rock	Schmidt
Meier	Rock	Schulz

Produkte

Produkt	Preis	Klasse
Hose	100	B
Rock	200	A

Keine guten Tabellen, da bspw. Schlüssel unklar. Aber zum Üben ok. Und es waren die Beispiele der Relationenalgebra.

Projektion

Betrachten zuerst die Projektion einer Tabelle auf einen Satz von Spalten:

$$\text{Proj}(R, [B_1, \ldots, B_k]) \quad , \quad \pi_{B_1,\ldots,B_k} R \quad , \quad \text{PJ}_{(B_1,\ldots,B_k)} R$$

```
SELECT b1,b2,b3 FROM r;
```

Bsp.:

```
SELECT kaeufer FROM verkaeufer1;
SELECT kaeufer, produkt FROM verkaeufer1;
```

Erste Abfrage liefert Tabelle mit doppelten Einträgen (zweimal ‚Schmidt'). Manchmal unerwünscht.

```
SELECT DISTINCT kaeufer FROM verkaeufer1;
```

Benötigt etwas länger.

Ergebnisse *nicht* sortiert.

```
SELECT DISTINCT verkaeufer FROM verkaeufer2;
```

Sortierung festlegen über `ORDER BY` Klausel.

```
SELECT DISTINCT verkaeufer FROM verkaeufer2
  ORDER BY verkaeufer ASC;
SELECT DISTINCT verkaeufer FROM verkaeufer2
  ORDER BY verkaeufer;  -- ASC default
SELECT DISTINCT verkaeufer FROM verkaeufer2
  ORDER BY verkaeufer DESC;
```

Auch über mehrere Spalten hinweg möglich.

```
SELECT kaeufer, produkt FROM verkaeufer1
  ORDER BY kaeufer DESC, product ASC;
```

Selektion

Nun die Auswahl bestimmter Datensätze einer Tabelle:

$$\mathsf{Sel}(R, Bed) \quad , \quad \sigma_{Bed} R \quad , \quad \mathsf{SL}_{Bed} R$$

```
SELECT * FROM r WHERE bed;
```

Bsp.:

```
SELECT * FROM verkaeufer1
    WHERE verkaeufer=='Meier';

SELECT * FROM verkaeufer1
    WHERE verkaeufer==produkt;
```

Beantwortet für Tabelle `verkaeufer1` die Fragen

- Welche Daten haben wir über den Verkäufer ‚Meier'?
- Wo stimmen Verkäufer-Namen und Produkt-Name überein?

Direkt kombinierbar mit Projektion.

- Wer hat bei ‚Meier' gekauft?

```
SELECT kaeufer
    FROM verkaeufer1
    WHERE verkaeufer=='Meier';
```

- Was hat Käufer ‚Schmidt' bei ‚Meier' gekauft?

```
SELECT product
    FROM verkaeufer1
    WHERE (verkaeufer=='Meier') AND (kaeufer=='Schmidt');
```

Vergleiche bei Strings.

`verkaeufer=='Meier'` ist äquivalent mit `verkaeufer LIKE 'Meier'`

Auch mit Wildcards % und _ kombinierbar. Etwa

- ▷ `v LIKE 'M%'` , alles, was mit 'M' anfängt.
- ▷ `v LIKE '%er'` , alles, was mit 'er' aufhört.
- ▷ `v LIKE '%ei%'` , alles, was ein 'ei' enthält.

Und

- ▷ `v LIKE 'M____'` , alles, was mit 'M' anfängt und vier Zeichen danach besitzt.

Vor String-Vergleichen noch `UPPER()` oder `LOWER()` anwendbar. Gegen Gross-Klein-Schreibfehlern in der Tabelle.

```
SELECT kauefer
    FROM verkauefer1
    WHERE UPPER(verkaeufer)=='MEIER';
```


Vorsicht beim Vergleich und Rechnen mit `NULL`. Dreiwertige Logik. Und

```
v == NULL
```

liefert *immer* `NULL`. Auch wenn `v` tatsächlich `NULL` ist.

Korrekter Test auf `NULL`.

```
v IS NULL
```

Vereinigung, Schnitt, Differenz

Bisher nur auf einer Tabelle gearbeitet. Jetzt zwei (oder mehr). Aber vereinigungsverträglich.

Vereinigung zweier Tabellen:

$$R \cup S \quad , \quad R \text{ UN } S$$

```
SELECT * FROM r
UNION [ALL]
SELECT * FROM s;
```

Mit `ALL` werden auch doppelte Einträge angezeigt.

Bsp.:

```
SELECT * FROM verkaeufer1
UNION
SELECT * FROM verkaeufer2;
```


Schnitt zweier Tabellen:

$$R \cap S \quad , \quad R \text{ IN } S$$

```
SELECT * FROM r
INTERSECT
SELECT * FROM s;
```

Bsp.:

```
SELECT * FROM verkaeufer1
INTERSECT
SELECT * FROM verkaeufer2;
```

Differenz zweier Tabellen:

$$R \setminus S \quad , \quad R \text{ DF } S$$

```
SELECT * FROM r
EXCEPT
SELECT * FROM s;
```

Bsp.:

```
SELECT * FROM verkaeufer1
EXCEPT
SELECT * FROM verkaeufer2;
```


Verbund

Verknüpfen strukturungleiche Tabellen miteinander. Beginnen mit dem Kreuzprodukt:

$$R \times S$$

```
SELECT * FROM r, s;
```

oder

```
SELECT * FROM r CROSS JOIN s;
```

Bsp.:

```
SELECT * FROM verkaeufer1, produkte;
```

Liefert schnell sehr grosse Tabellen.

Join von zwei Tabellen:

$$\text{Join}(R, S, Bed) \quad , \quad R \bowtie_{Bed} S \quad , \quad R \text{ JN}_{Bed} S$$

```
SELECT * FROM r, s
  WHERE Bed;
```

Bsp.:

```
SELECT * FROM verkaeufer1, produkte
  WHERE verkaeufer1.verkaeufer==produkte.produkt;

SELECT * FROM verkaeufer1, produkte
  WHERE verkaeufer1.produkt==produkte.produkt;

SELECT * FROM verkaeufer1 NATURAL JOIN produkte;
```


Komplexer SELECT-Ausdruck

Tabellen:

Angestellter

AngNr	Name	Wohnort	Beruf	AbtNr
112	Müller	Erfurt	Ingenieur	3
205	Winter	Zwickau	Programmierer	3
117	Rüllich	Weimar	Hundezüchter	5
198	Schumann	Jena	Kaufmann	4

Projekt

ProNr	Titel	Inhalt	Leiter
27	Pkw2000	blabla...	205
16	Wankel99	blabla...	117
84	Trabi00	blabla...	117

Mitarbeit

ProNr	AngNr	Prozent
27	112	100
27	198	70
16	198	20

Anfrage: „Finde alle Namen der Angestellten, die im Projekt 27 mitarbeiten“

Anfrage: „Finde alle Namen der Angestellten, die im Projekt 27 mitarbeiten“

- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit))
- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{PJ}_{(AngNr)}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit)))
- $\mathsf{PJ}_{(Name)}$($\mathsf{SL}_{ProNr=27}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))
- $\mathsf{SL}_{ProNr=27}$($\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))

```
SELECT DISTINCT Name
  FROM Angestellter,
       (SELECT * FROM Mitarbeit WHERE ProNr==27) AS Temp
  WHERE Angestellter.AngNr==Temp.AngNr;
```


SQL — Teil 2 Komplexer SELECT-Ausdruck

Anfrage: „Finde alle Namen der Angestellten, die im Projekt 27 mitarbeiten“

- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit))
- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{PJ}_{(AngNr)}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit)))
- $\mathsf{PJ}_{(Name)}$($\mathsf{SL}_{ProNr=27}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))
- $\mathsf{SL}_{ProNr=27}$($\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))

```
SELECT DISTINCT Name
  FROM Angestellter,
       (SELECT AngNr FROM Mitarbeit WHERE ProNr==27) AS Temp
  WHERE Angestellter.AngNr==Temp.AngNr;
```

Anfrage: „Finde alle Namen der Angestellten, die im Projekt 27 mitarbeiten“

- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit))
- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{PJ}_{(AngNr)}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit)))
- $\mathsf{PJ}_{(Name)}$($\mathsf{SL}_{ProNr=27}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))
- $\mathsf{SL}_{ProNr=27}$($\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))

```
SELECT DISTINCT Name
  FROM Angestellter JOIN Mitarbeit
  WHERE Angestellter.AngNr==Mitarbeit.AngNr
        AND
        Mitarbeit.ProNr==27;
```

```
SELECT Name
  FROM Angestellter NATURAL JOIN Mitarbeit
  WHERE Mitarbeit.ProNr==27;
```


SQL — Teil 2 Komplexer SELECT-Ausdruck

Anfrage: „Finde alle Namen der Angestellten, die im Projekt 27 mitarbeiten“

- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit))
- $\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ ($\mathsf{PJ}_{(AngNr)}$ ($\mathsf{SL}_{ProNr=27}$ Mitarbeit)))
- $\mathsf{PJ}_{(Name)}$($\mathsf{SL}_{ProNr=27}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))
- $\mathsf{SL}_{ProNr=27}$($\mathsf{PJ}_{(Name)}$(Angestellter $\mathsf{JN}_{AngNr=AngNr}$ Mitarbeit))

```
SELECT *
  FROM (SELECT Name
          FROM Angestellter NATURAL JOIN Mitarbeit) AS Temp
  WHERE Temp.ProNr==27;
```

Geht schief!!


THE END