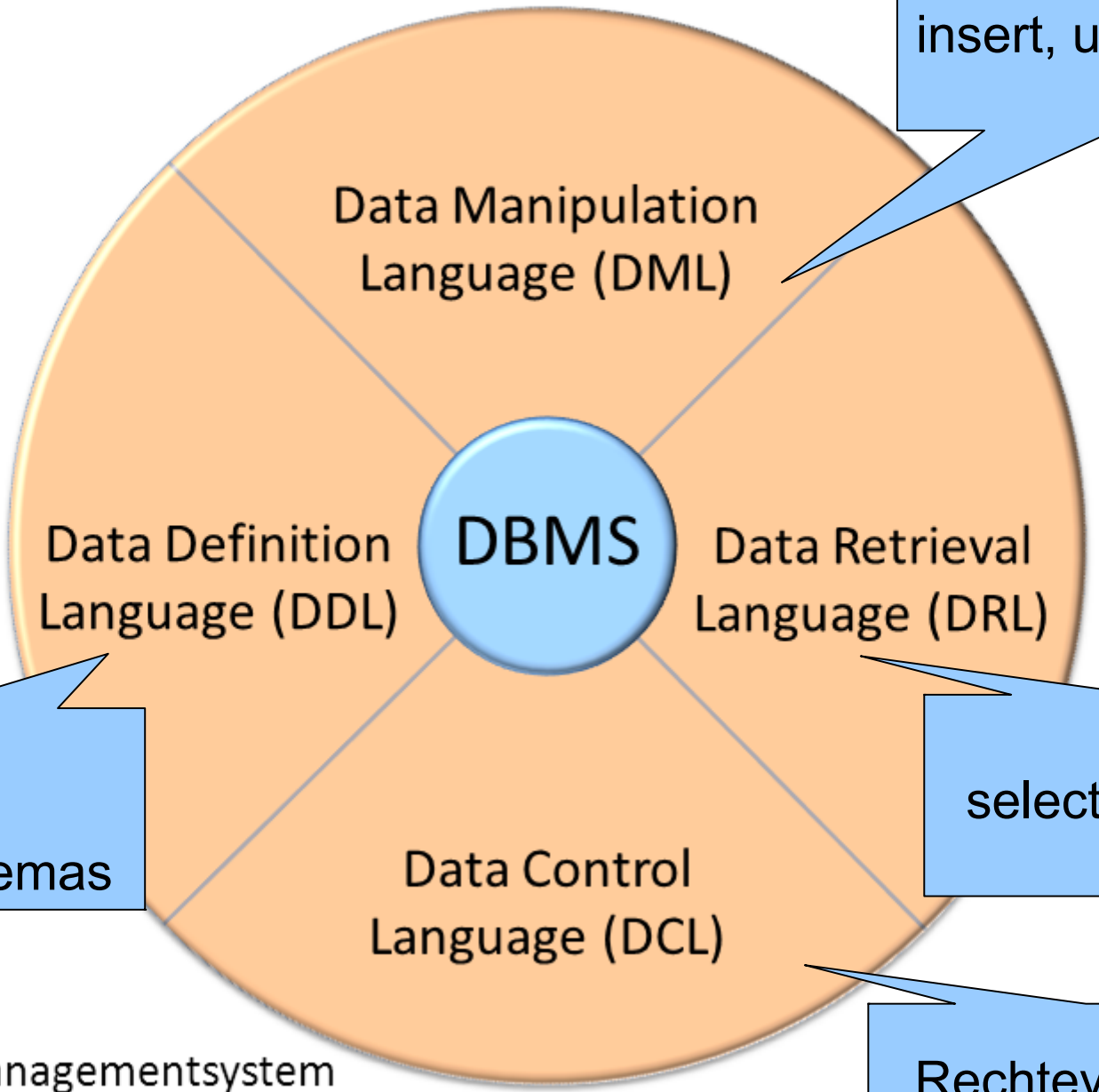
Einführung in SQL

Structured Query Language (SQL)



insert, update, delete

Definition des Datenbankschemas

select, from, where

Rechteverwaltung, Transaktionskontrolle

DBMS = Datenbankmanagementsystem

SQL zur Kommunikation mit dem DBMS

*SQL ist eine Datenbanksprache zur **Definition**, **Abfrage** und **Manipulation** von Daten in relationalen Datenbanken.*

```
SELECT [DISTINCT] Auswahlliste [AS Alias]
FROM Tabelle [AS Alias]
[WHERE Where-Klausel]
[GROUP BY (Group-by-Attribut)+
[HAVING Having-Klausel]]
[ORDER BY (Sortierungsattribut [ASC|DESC])+];
```

Beispiele (Select) (1)

Gesucht ist die gesamte Tabelle namens „*Protokoll*“.

```
SELECT * FROM Protokoll;
```

id	Name	Vorname	Datum	Stunden	entschuldigt
6	Einstein	Albert	2015-09-02	2	0
9	Fischer	Gottfried	2015-09-02	2	1
13	Kant	Imanuel	2015-09-02	2	0
1	Einstein	Albert	2015-09-01	2	0
2	Descartes	Rene	2015-09-01	2	1
4	Merkel	Angelika	2015-09-01	2	1
12	Kant	Imanuel	2015-09-01	2	0
3	Rosenthal	Hans	2015-08-28	2	0
5	Fischer	Gottfried	2015-08-28	1	0
10	Reagan	Ronald	2015-08-28	2	0
14	Kant	Imanuel	2015-08-28	5	1
7	Becker	Boris	2015-08-27	2	0
11	Reagan	Ronald	2015-08-27	1	0
8	Einstein	Albert	2015-08-26	2	1
15	Becker	Boris	2015-08-26	3	1

0 = false
1 = true

Beispiele (Select) (2)

Gesucht sind die Namen aller Teilnehmer, die Fehlstunden haben.

```
SELECT Name,Vorname FROM Protokoll;
```

Name	Vorname
Einstein	Albert
Descartes	Rene
Rosenthal	Hans
Merkel	Angelika
Fischer	Gottfried
Einstein	Albert
Becker	Boris
Einstein	Albert
Fischer	Gottfried
Reagan	Ronald
Reagan	Ronald
Kant	Imanuel
Kant	Imanuel
Kant	Imanuel
Becker	Boris

Problem:
Viele Namen sind hier doppelt!

Beispiele (Select) (3)

Gesucht sind die Namen aller Teilnehmer die Fehlstunden haben.

```
SELECT DISTINCT Name,Vorname FROM Protokoll;
```

Name	Vorname
Einstein	Albert
Descartes	Rene
Rosenthal	Hans
Merkel	Angelika
Fischer	Gottfried
Becker	Boris
Reagan	Ronald
Kant	Imanuel

Beispiele (Select) (4)

Gesucht ist nur die Person mit id 6.

```
SELECT * FROM Protokoll WHERE id = "6";
```

id	Name	Vorname	Datum	Stunden	entschuldigt
6	Einstein	Albert	2015-09-02	2	0

Vergleichsoperatoren

Allgemein:

=, <, >, <=, >=, <>

Weitere Vergleichsoperatoren:

LIKE: Ähnlichkeit mit Zeichenkette (Platzhalter: %)

BETWEEN: Werte zwischen zwei Grenzen

IN: Inhalt einer Spalte in angegebener Liste vorhanden

IS NULL: Null-Wert prüfen

NULL

- bezeichnet „unbestimmten Wert“ bzw. „leere Tabellenzelle“
- um mit null zu vergleichen werden die Befehl „IS NULL“ und „IS NOT NULL“ verwendet

```
SELECT Spaltenname
FROM Tabelle
WHERE Spaltenname IS NOT NULL
```

Logische Verknüpfungen

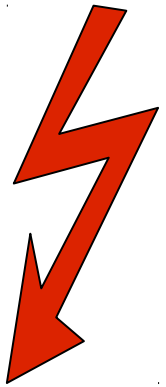
Verknüpfung	**Operator**
AND	&&
OR	||
NOT	!

Beispiele (Select) (5)

Gesucht sind die Fehlstunden jeder Person.

```
SELECT Name,Vorname,SUM(Stunden)
FROM Protokoll;
```

Die Spalten *Name* und *Vorname* enthalten mehrere Einträge.

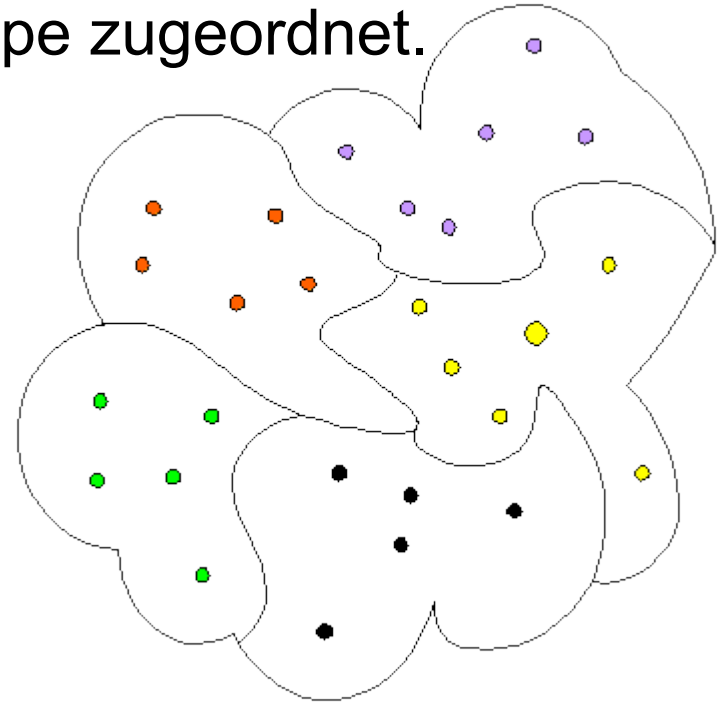


Die Aggregatfunktion *SUM* liefert aber nur einen Wert.

Name	Vorname	SUM(Stunden)
Einstein	Albert	32

GROUP BY: Datensätze in Gruppen einteilen

Jeder Datensatz wird genau einer Gruppe zugeordnet.



Gesamtwolke = die Menge aller Datensätze. Jeder Punkt symbolisiert einen Datensatz. Diese Menge ist mittels `GROUP BY farbe` in fünf Gruppen eingeteilt.

→ man kann nur noch Angaben über die Gruppen machen, nicht mehr über einzelne Datensätze.

Man kann also Informationen über die Gruppe der gelben Datensätze ausgeben, aber keine Detailangabe über den Datensatz mit dem großen gelben Punkt.

Beispiele (Select) (6)

Gesucht sind die Fehlstunden jeder Person.

```
SELECT Name, Vorname, SUM(Stunden) AS Gesamt
FROM Protokoll
GROUP BY Name, Vorname;
```

Name	Vorname	Gesamt
Becker	Boris	5
Descartes	Rene	2
Einstein	Albert	6
Fischer	Gottfried	3
Kant	Imanuel	9
Merkel	Angelika	2
Reagan	Ronald	3
Rosenthal	Hans	2

Beispiele (Select) (7)

Gesucht sind die unentschuldigten Fehlstunden jeder Person – absteigend sortiert.

```
SELECT Name,Vorname,SUM(Stunden) AS unentschuldigt
FROM Protokoll
WHERE entschuldigt = 0
GROUP BY Name, Vorname
ORDER BY SUM(Stunden) DESC;
```

WHERE schränkt auf die unentschuldigten Stunden ein.

ORDER BY sortiert das Ergebnis mit **DESC** absteigend.

Name	Vorname	unentschuldigt
Einstein	Albert	4
Kant	Imanuel	4
Reagan	Ronald	3
Rosenthal	Hans	2
Becker	Boris	2
Fischer	Gottfried	1

Typischer Fehler bei GROUP BY

Der typische Fehler im Zusammenhang mit der Gruppierung besteht darin, dass nach der Gruppenbildung noch versucht wird, Informationen von einzelnen Datensätzen auszugeben. Zum Beispiel:

```
SELECT Name, Datum, SUM(Stunden) AS Gesamt
FROM Protokoll
GROUP BY Name, Vorname;
```

Man kann zwar für jede Gruppe den Namen und die Summe der Stunden angeben, aber nicht das Datum, denn jeder Name hat ja mehrere unterschiedliche Fehltage.

Typischer Fehler bei GROUP BY (2)

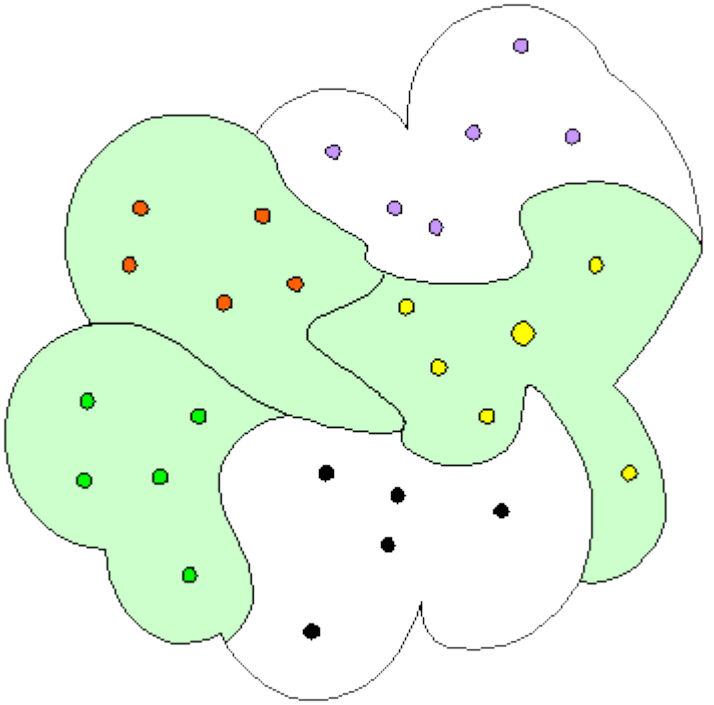
meisten DBMS → Fehlermeldung („Attribut Name nicht Bestandteil der GROUP BY-Klausel“).
MySQL liefert:

Name	Datum	Gesamt
Becker	2015-08-27	5
Descartes	2015-09-01	2
Einstein	2015-09-01	6
Fischer	2015-08-28	3
Kant	2015-09-01	9
Merkel	2015-09-01	2
Reagan	2015-08-28	3
Rosenthal	2015-08-28	2

Ergebnis fragwürdig!

→ in einer SELECT-Anweisung mit GROUP BY dürfen nur die Gruppierungsattribute und die Aggregatfunktionen vorkommen!

Gruppenwahl - die HAVING-Klausel



In Frage kommenden Gruppen auswählen

Kriterien zur Auswahl?

- Gruppeneigenschaft, also entweder danach, wie die Gruppen gebildet wurden
- Aggregatfunktion, die eine Gruppeneigenschaft bestimmt

→ HAVING-Klausel in aller Regel mit einer Aggregatfunktion benutzt,

(in einer WHERE-Klausel ist keine Aggregatfunktion möglich)

Gruppenwahl - die HAVING-Klausel (2)

Es sollen die Personen angezeigt werden, die mehr als 5 Fehlstunden haben

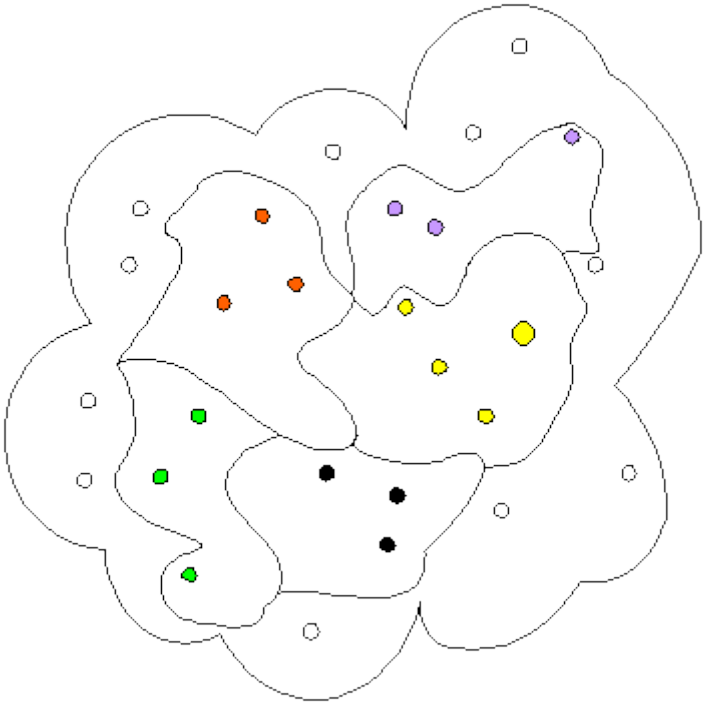
```
SELECT Vorname, Name, SUM(Stunden) AS Gesamt
FROM Protokoll
GROUP BY Name, Vorname
HAVING SUM(Stunden) >= 5;
```

Vorname	Name	Gesamt
Boris	Becker	5
Albert	Einstein	6
Imanuel	Kant	9

WHERE und GROUP BY

Auch eine SELECT-Anweisung mit GROUP-BY-Klausel kann eine WHERE-Bedingung enthalten.
Die WHERE-Bedingung wählt aus der gesamten Datenmenge die Datensätze aus, die anschließend gruppiert werden.

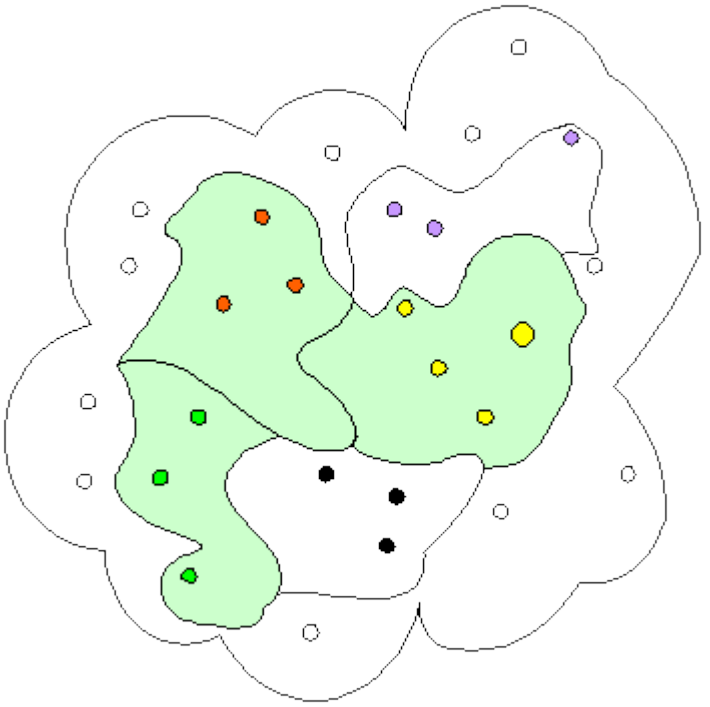
Die WHERE-Bedingung hat die farbigen Datensätze ausgewählt, das anschließende GROUP BY hat die farbigen Datensätze nach der Farbe gruppiert.



Und jetzt alles zusammen! - WHERE, GROUP BY und HAVING

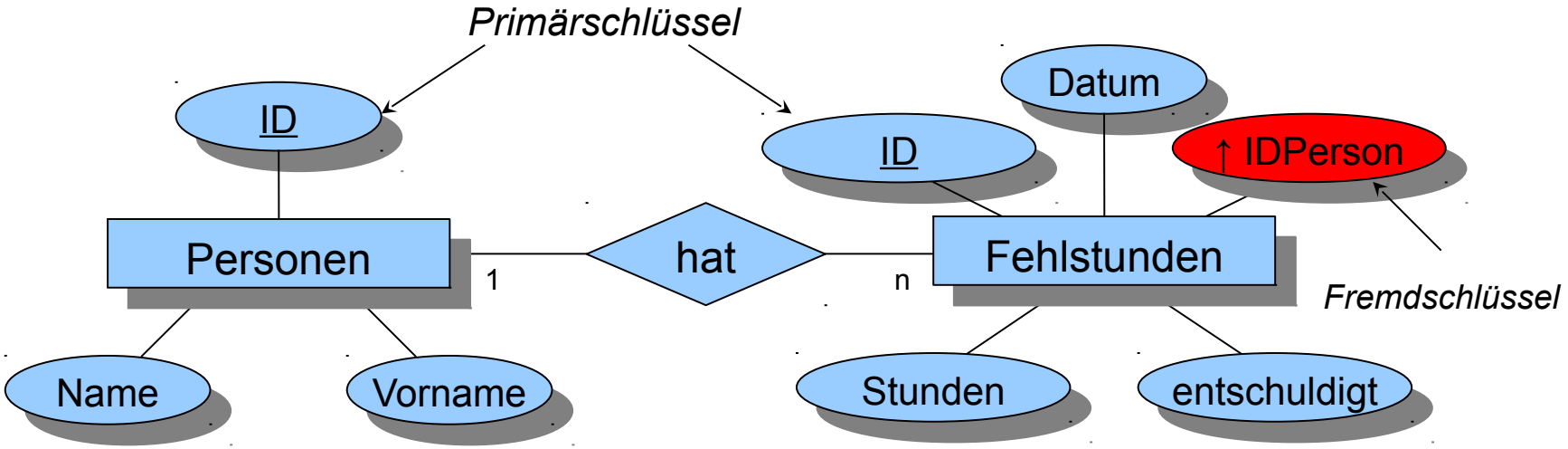
Reihenfolge:
WHERE, GROUP BY dann HAVING.

Zuerst werden die zu betrachtenden Datensätze mittels WHERE selektiert, dann werden diese mit GROUP BY gruppiert und zum Schluss werden mit HAVING die gewünschten Gruppen ausgewählt



Die WHERE-Klausel wählt also die farbigen Datensätze aus, die GROUP BY-Klausel gruppiert nach Farben und die HAVING-Klausel wählt die rote, grüne und gelbe Gruppe aus.

Datenbanken mit mehreren Tabellen (1)



Die Daten werden in **zwei Tabellen** aufgeteilt. So können Kursmitglieder (in die Personentabelle) eingefügt werden, ohne dass sie zwangsläufig Fehlstunden haben müssen.

Die Verbindung zwischen beiden Tabellen wird mit **Primär-** bzw. **Fremdschlüsseln** realisiert.

Datenbanken mit mehreren Tabellen (2)

Tabelle: *Fehlstunden*

id	Datum	Stunden	entschuldigt	IDPerson ▲ 1
1	2015-09-01	2	0	1
12	2015-09-01	2	0	1
8	2015-08-26	2	1	1
4	2015-09-01	2	1	2
14	2015-08-28	5	1	2
13	2015-09-02	2	0	2
11	2015-08-27	1	0	2
3	2015-08-28	2	0	3
9	2015-09-02	2	1	3
6	2015-09-02	2	0	4
2	2015-09-01	2	1	4
10	2015-08-28	2	0	5
7	2015-08-27	2	0	5
5	2015-08-28	1	0	5
15	2015-08-26	3	1	6

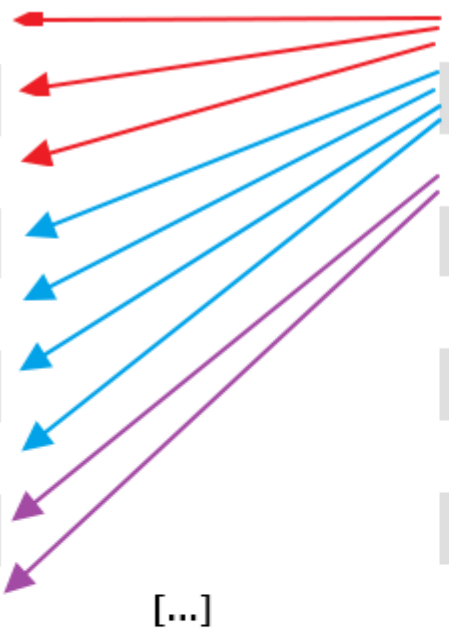


Tabelle: *Personen*

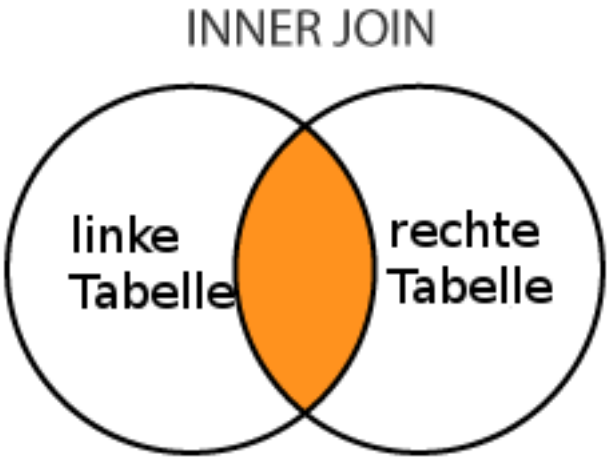
id	Name	Vorname
1	Einstein	Albert
2	Descartes	Rene
3	Rosenthal	Hans
4	Merkel	Angelika
5	Fischer	Gottfried
7	Becker	Boris
10	Reagan	Ronald
12	Kant	Imanuel

Datenbanken mit mehreren Tabellen

Joins

Joins

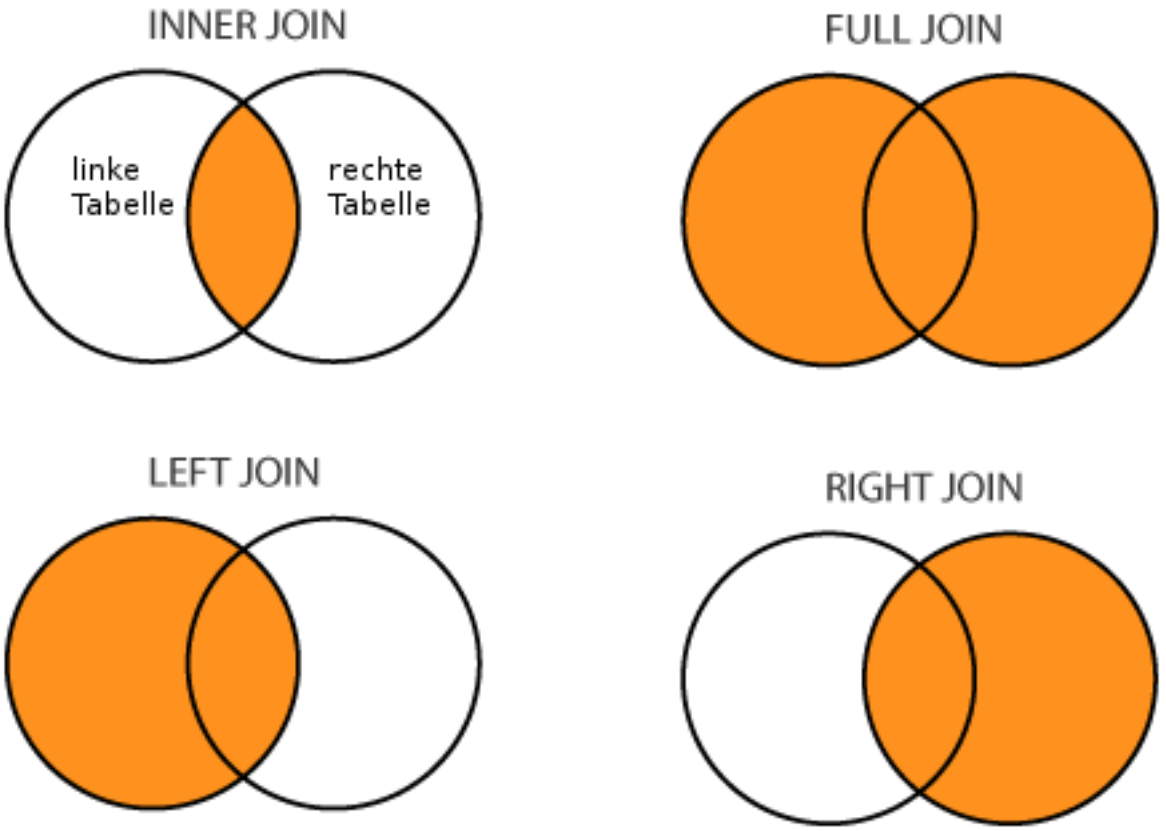
- Ein SQL JOIN kombiniert Datensätze aus zwei Tabellen
- JOIN findet entsprechende Spaltenwerte in zwei Tabellen
- Eine SQL-Anweisung kann keine, eine oder mehrere JOIN Anweisungen haben
- INNER JOIN ist dasselbe wie JOIN; das Schlüsselwort INNER ist optional



Verschiedene Arten von Joins

- **(INNER) JOIN**: Wählt Datensätze aus, die gleiche Werte in beiden Tabellen haben
- **LEFT (OUTER) JOIN**: Wählt alle Datensätze von der linken (left-most) Tabelle mit entsprechend gleichen Werte in der rechten Tabelle
- **RIGHT (OUTER) JOIN**: Wählt alle Datensätze von der rechten/zweiten (right-most) Tabelle mit entsprechend gleichen Werte in der linke Tabelle
- **FULL (OUTER) JOIN**: Wählt alle Datensätze beider Tabellen aus

Verschiedene Arten von Joins



Schlüsselworte INNER und OUTER sind optional

inner join (natürlichen Verbund)

- die Werte der gleichbenannten Attribute müssen übereinstimmen
  - so breit wie alle Attribute der Tabellen
  - Tupel ohne Join-Partner gehen verloren

Join (rette partnerlose Tupel)

- **Left join**
  Es werden auch die Zeilen aus der ersten Tabelle aufgeführt, die keinen Partner in der zweiten Tabelle haben. Es wird mit NULL aufgefüllt.
  Ziel: Tabelle erhalten, semantische Zusatzwerte daran kleben
- **Right join**
  Die Zeilen aus der zweiten Tabelle aufgeführt, die keinen Partner in der zweiten Tabelle haben. Es wird ebenso mit NULL aufgefüllt
- **Full join**
  Tupel der beiden Argumentrelationen bleiben erhalten

Beispiel: Relationen L und R

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R		
C	D	E
c_1	d_1	e_1
c_3	d_3	e_3

inner join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R		
C	D	E
c_1	d_1	e_1
c_3	d_3	e_3

inner join				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1

left join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R		
C	D	E
c_1	d_1	e_1
c_3	d_3	e_3

left join				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	***null***	***null***

Bsp.: Studenten, die Vorlesungen belegt haben.
Studenten ohne Vorlesung sollen auch angezeigt werden.

right join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R		
C	D	E
c_1	d_1	e_1
c_3	d_3	e_3

right join				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
null	*null*	c_3	d_3	e_3

full join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R		
C	D	E
c_1	d_1	e_1
c_3	d_3	e_3

full join				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	*null*	*null*
null	*null*	c_3	d_3	e_3

Allgemeine Syntax Join

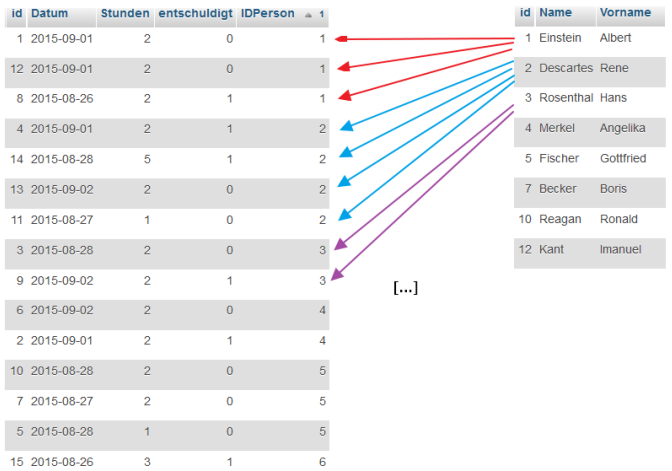
```
SELECT Spaltenname(n)
FROM Tabelle1 JOIN Tabelle2
ON   Spaltenname1 = Spaltenname2
WHERE Bedingung
```

```
SELECT Spaltenname(n)
FROM Tabelle1 JOIN Tabelle2
ON   Tabelle1.Spaltenname = Tabelle2.Spaltenname
WHERE Bedingung
```

Select mit Inner-Join (1)

Gesucht sind die unentschuldigten Fehlstunden jeder Person – absteigend sortiert.

```
SELECT P.Name, P.Vorname, SUM(F.Stunden) AS ue
FROM Personen AS P, Fehlstunden AS F
WHERE    F.entschuldigt = 0 AND
         P.ID = F.IDPerson
GROUP BY P.Name, P.Vorname
ORDER BY SUM(F.Stunden) DESC;
```



id	Datum	Stunden	entschuldigt	IDPerson ▲ 1
1	2015-09-01	2	0	1
12	2015-09-01	2	0	1
8	2015-08-26	2	1	1
4	2015-09-01	2	1	2
14	2015-08-28	5	1	2
13	2015-09-02	2	0	2
11	2015-08-27	1	0	2
3	2015-08-28	2	0	3
9	2015-09-02	2	1	3
6	2015-09-02	2	0	4
2	2015-09-01	2	1	4
10	2015-08-28	2	0	5
7	2015-08-27	2	0	5
5	2015-08-28	1	0	5
15	2015-08-26	3	1	6

[...]

id	Name	Vorname
1	Einstein	Albert
2	Descartes	Rene
3	Rosenthal	Hans
4	Merkel	Angelika
5	Fischer	Gottfried
7	Becker	Boris
10	Reagan	Ronald
12	Kant	Imanuel

Name	Vorname	ue
Fischer	Gottfried	5
Einstein	Albert	4
Descartes	Rene	3
Rosenthal	Hans	2
Merkel	Angelika	2

Select mit Inner-Join (2)

Gesucht sind die unentschuldigten Fehlstunden jeder Person – absteigend sortiert.

```
SELECT P.Name, P.Vorname, SUM(F.Stunden) AS ue
FROM Personen AS P INNER JOIN Fehlstunden AS F
        ON P.ID = F.IDPerson
WHERE   F.entschuldigt = 0
GROUP BY P.Name, P.Vorname
ORDER BY SUM(F.Stunden) DESC;
```

INNER JOIN
LEFT JOIN
RIGHT JOIN
OUTER JOIN
[...]

Name	Vorname	ue
Fischer	Gottfried	5
Einstein	Albert	4
Descartes	Rene	3
Rosenthal	Hans	2
Merkel	Angelika	2

SELECT mit Inner/Left Join

```
SELECT P.Name, P.Vorname, SUM(F.Stunden) AS FS
FROM   Personen AS P {INNER | LEFT} JOIN
Fehlstunden AS F ON P.ID = F.IDPerson
GROUP BY P.Name, P.Vorname;
```

INNER

Name	Vorname	FS
Descartes	Rene	10
Einstein	Albert	6
Fischer	Gottfried	5
Merkel	Angelika	4
Rosenthal	Hans	4

LEFT

Name	Vorname	FS
Becker	Boris	*NULL*
Descartes	Rene	10
Einstein	Albert	6
Fischer	Gottfried	5
Kant	Imanuel	*NULL*
Merkel	Angelika	4
Reagan	Ronald	*NULL*
Rosenthal	Hans	4

UNION

```
SELECT Spalte1 FROM tabelle1
UNION Spalte2 FROM  tabelle2
```

- Datentypen der zusammenzufassenden Spalten müssen sich gleichen
- beide SELECT-Anweisungen müssen die gleiche Anzahl an Spalten zurückgeben
- Inhalte der zurückgegebenen Tabelle/Spalte sind einzigartig

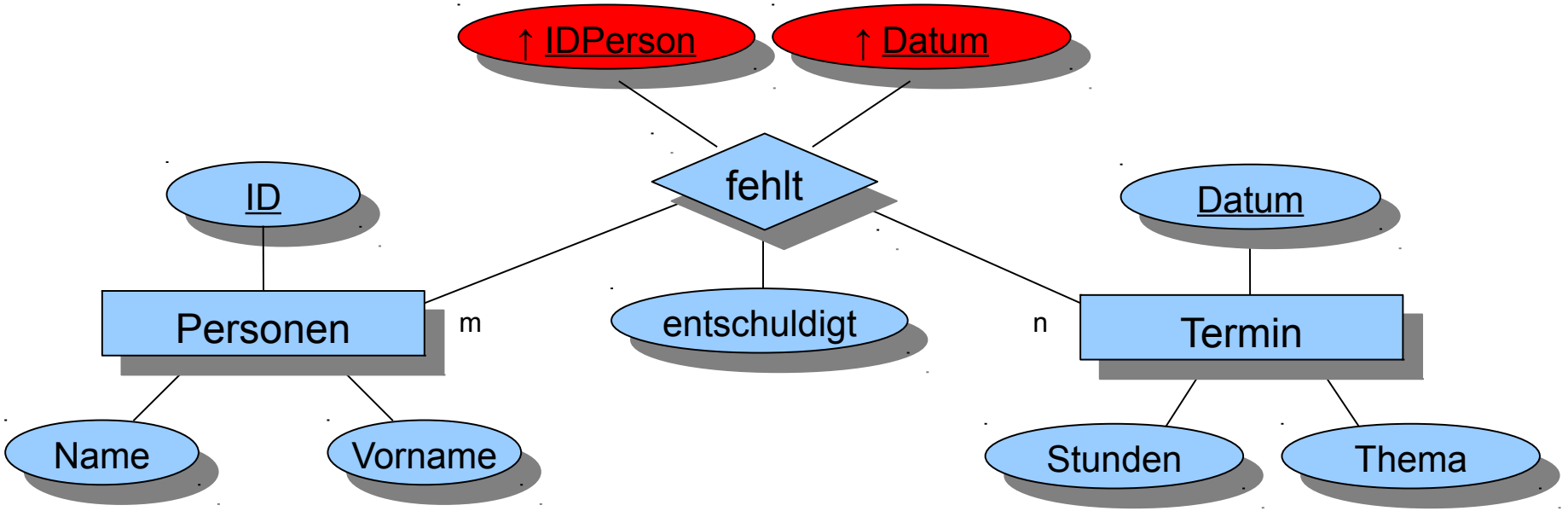
Union (2)

Name
Herbert
Nelli

Name
Otto
Angelika
Hugo

Name
Herbert
Nelli
Otto
Angelika
Hugo

Verknüpfungstabellen (1)



Personen und *Termine* werden mit Hilfe einer **Verknüpfungstabelle** in Beziehung zueinander gesetzt. (n:m-Beziehung)

Das Attribut *entschuldigt* wird in der Beziehungstabelle eingefügt.

Verknüpfungstabelle (2)

Termin

Thema	Stunden	datum
Java	2	2015-09-02
Datenbanken	2	2015-09-01
Implementierung	2	2015-08-28
UML	2	2015-08-27
OOP	2	2015-08-26

fehlt

Datum ▾ 1	entschuldigt	IDPerson
2015-09-02	0	2
2015-09-02	1	3
2015-09-02	0	4
2015-09-01	0	1
2015-09-01	0	1
2015-09-01	1	2
2015-09-01	1	4
2015-08-28	0	5
2015-08-28	1	2
2015-08-28	0	3
2015-08-28	0	5
2015-08-27	0	2
2015-08-27	0	5
2015-08-26	1	1
2015-08-26	1	6

Person

id	Name	Vorname
1	Einstein	Albert
2	Descartes	Rene
3	Rosenthal	Hans
4	Merkel	Angelika
5	Fischer	Gottfried
7	Becker	Boris
10	Reagan	Ronald
12	Kant	Imanuel

Select mit Verknüfungstabelle

Gesucht sind die Termine, an denen alle Teilnehmer anwesend waren.

```
SELECT Termin.Datum, Termin.Thema
FROM Termin
WHERE Termin.Datum NOT IN
     (    SELECT DISTINCT Termin.Datum
          FROM  Personen, fehlt, Termin
          WHERE Personen.ID = fehlt.IDPerson AND
                fehlt.Datum = Termin.Datum
     );
```

Sprachelemente

Sprachelemente

Folgende SQL-Sprachelemente werden vorausgesetzt:

```
SELECT (DISTINCT) … FROM
WHERE
GROUP BY
ORDER BY
ASC, DESC
(LEFT / RIGHT) JOIN … ON
UNION
AS
NULL
Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL
Arithmetische Operatoren: +, -, *, /, (…)
Logische Verknüpfungen: AND, OR, NOT
Funktionen: COUNT, SUM, MAX, MIN
```

Zusätzlich: AVG

Es werden SQL-Abfragen über eine und mehrere verknüpfte Tabellen vorausgesetzt.
Es können auch verschachtelte SQL-Ausdrücke vorkommen.

Quellen

- Wikipedia
- Präsentation von Volker Quade
- http://www.schulserver.hessen.de/darmstadt/lichtenberg/SQLTutorial/
- http://www.dofactory.com