Grundlagen von SQL

Informatik 2, FS18

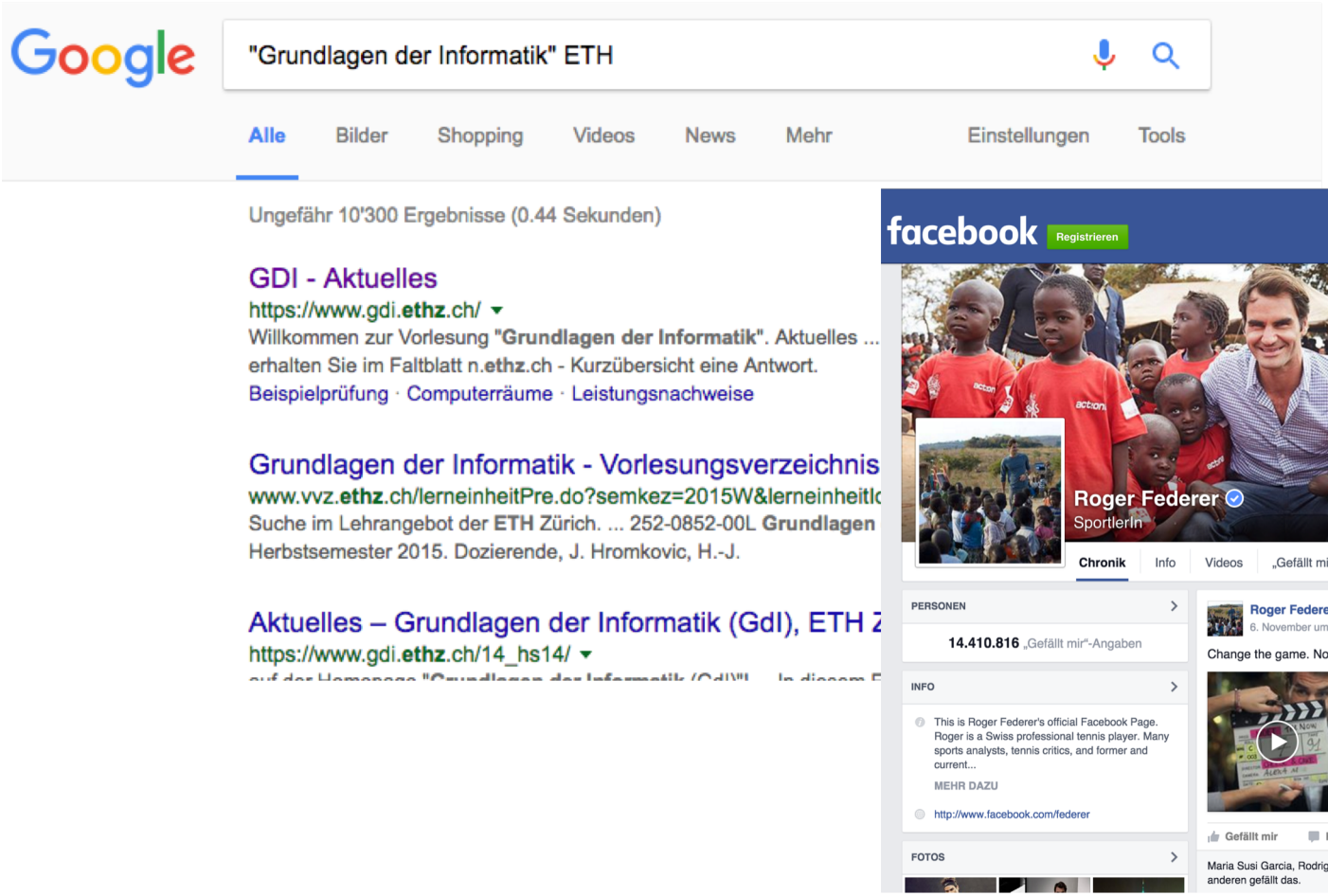
Dr. Hermann Lehner (Material von Dr. Markus Dahinden)
Departement Informatik, ETH Zürich

Grundlagen von SQL

(Structured Query Language)

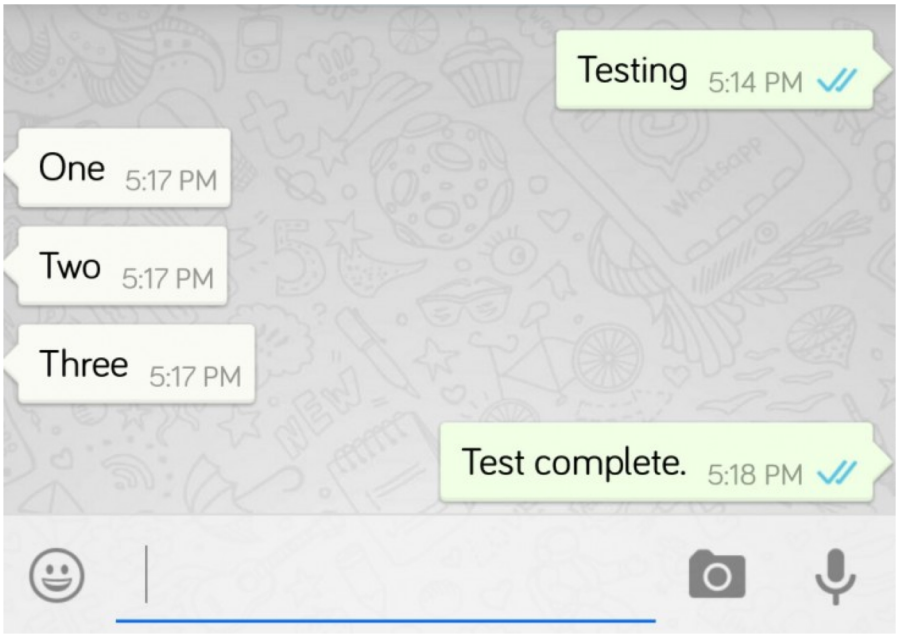
- Datenbanksprache
- Befehle
  - Datenbanken und Tabellen erstellen/verändern
  - Daten manipulieren (eingeben, ändern, löschen)
  - Datenbank durchsuchen (Queries erstellen)
- Die meisten Datenbanksysteme unterstützen SQL

Google, Facebook und Co. setzen auf SQL



goo.gl/R5sbo8 (Google) goo.gl/pduZgl (Facebook)

Whatsapp



wa.db

- Tables (3)
  - android_metadata
    - locale
  - sqlite_sequence
    - name
    - seq
  - wa_contacts
    - id
    - jid
    - is_whatsapp_user
    - is_iphone
    - status
    - number
    - raw_contact_id
    - display_name
    - phone_type
    - phone_label
    - unseen_msg_count
    - photo_ts

msgstore.db

- Tables (3)
  - chat_list
    - _id
    - key_remote_jid
    - message_table_id
  - messages
    - _id
    - key_remote_jid
    - key_from_me
    - key_id
    - status
    - needs_push
    - data
    - timestamp
    - media_url
    - media_mime_type
    - media_wa_type
    - media_size
    - media_name
    - latitude
    - longitude
    - thumb_image
    - remote_resource
    - received_timestamp
    - send_timestamp
    - receipt_server_timestamp
    - receipt_device_timestamp
  - sqlite_sequence
    - name
    - seq

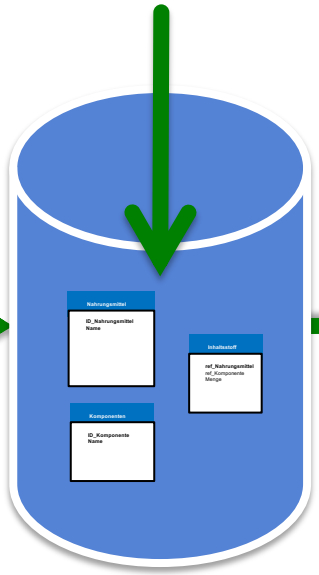
Pro Tag werden 44 Mia Nachrichten verschickt.
Für eine Nachricht sind 4 SQL Statements nötig.
Das sind 2x176 Milliarden SQL Statements pro Tag!

goo.gl/fj7d9u

Liste >SQL> relationale Datenbank >SQL> Liste

Create Table

Insert/
Update



Select

McDonald's Nährwerttabelle

PRODUKT	Brennwert (kcal)	Brennwert (kJ)	Eiweiß (g)	Kohlenhydrate (g)	Fett (g)
	Portion	Portion	Portion	Portion	Portion
BURGER & CO.					
Big Mäc™	503,0	2.107,3	26,2	43,5	25,3
Cheeseburger	312,1	1.309,0	16,4	32,8	13.0
Chicken McNuggets 6 Stück	236,0	995,9	19.3	11,8	13,1
Chicken McNuggets 9 Stück	357,1	1.493,9	29,0	17,6	19,6
Chicken McNuggets 20 Stück	793,5	3.319,7	64,4	39,2	43,6
Fischmäc	408,1	1.710,5	14,7	40,7	20,9
Gemüse Mäc	476,5	1.998.2	9,5	53,2	25,0
Hamburger	258,5	1.084,8	13,1	31,5	8,9
Hamburger Royal	513,5	2.151,6	31,8	36,7	27,3
Hamburger Royal Bacon	515	2240	32,1	36,7	28,1
Hamburger Royal TS	565,9	2.368,1	28,5	35,3	34,8
McChicken™	474,0	1.966,2	18,7	43,6	25,2
McRib™	485,3	2.034,1	26,7	45.8	22,1
Grilled Chicken Caprese	458,0	1.918,3	28.9	36,0	22,5
Pommes Frites klein	234,1	982,1	3,3	28,5	11,8
Pommes Frites mittel	332,6	1396,6	4,6	40,5	16,7
Pommes Frites groß	466,2	1.964,1	6,5	57,0	23,6
SALATE					
Grilled Chicken Caesar Salad (ohne Dressing)	205,3	856,6	29,4	5,6	7,9
Crispy Chicken Caesar Salad (ohne Dressing)	295,5	1.236,3	26,9	13,3	15.6
Grilled Chicken Ranch Salad (ohne Dressing)	290,5	1.214,2	33,8	5,9	15,5
Crispy Chicken Ranch Salad (ohne Dressing)	378,1	1.580,8	31,1	13,7	22,9
Garten Salat (ohne Dressing)	10,9	45,6	1,3	1,4	-
SNACKS					
Apfeltasche	234,7	964,4	2,4	27,4	12.8
Schoko Muffin	485,3	2.034,2	5,0	51,7	28.6
Blaubeer Muffin	430,5	1.805,1	4,4	48.4	24.2
Fruit &Yogurt	152,3	640,8	4,8	26,5	2,9
Frucht Tüte	47,7	201,0	0,3	10,6	0,3
McFlurry™ mit Bounty	352,1	1.476,0	7,7	36,9	19,3
McFlurry™ mit KITKAT	321,6	1.349,7	8,8	42,1	13,0
McFlurry™ mit Milka Erdbeergeschmack	373,5	1.568,3	8.4	52,8	14,1
McFlurry™ mit Smarties	348,7	1.463,6	8,0	47,5	13,9
McSundae™ (Eis ohne Waffeltüte)	110,5	463,6	3,9	13,6	4,5
McSundae™ Eisbecher mit Karamellsauce	285,3	1.199,0	8,0	44.8	8,0
McSundae™ Eisbecher mit Schokosauce	285,3	1.197,7	7,0	38,4	11,4
Schoko Milchshake 0,3l	296,7	1.255,4	10,1	46,3	7,9
Schoko Milchshake 0,5l	497,8	2.092,4	16,8	77,2	13,2
Vanille Milchshake 0,3l	295,7	1.243,1	9,6	46,6	7,7
Vanille Milchshake 0,5l	492,9	2.071,8	16,0	77,6	12,8
Erdbeer Milchshake 0,3l	299,7	1.259,7	9,6	47,5	7,7
Erdbeer Milchshake 0,5 l	499,4	2.099,4	16,0	79.2	12,8

Datenbanksystem

Befehle zur Daten-Definition	Befehle zur Daten-Manipulation/Abfrage	Alias und Funktionen
CREATE SCHEMA	INSERT INTO .. VALUES ..	AS
CREATE TABLE	SELECT	COUNT()
NOT NULL	FROM	AVG()
UNIQUE	WHERE	MAX()
PRIMARY KEY	GROUP BY	MIN()
FOREIGN KEY	ORDER BY	ROUND()
REFERENCES	UPDATE	SUBSTRING()
CHECK	DELETE	
CREATE INDEX	COMMIT	
CREATE VIEW	ROLLBACK	
ALTER TABLE	= | > | < | <= | >= | <> | LIKE	
CREATE TABLE AS	AND | OR | NOT	
DROP TABLE	ASC | DESC	
DROP VIEW	JOIN	

Datenbasis für nachfolgende Übungen

Studierende

Vorname	Nachname	Legi	Wohnort
Heinz	Meier	12-333-333	Zürich
Anne	Meier	13-123-456	Luzern
Michaela	Zbinden	13-222-222	Bern
Karl-Heinz	Meierhans	14-444-444	Zürich

Faecher

ID	Bezeichnung	Departement
1	Botanik	D-BIOL
2	Statistik	D-MATH
3	Informatik	D-INFK
4	Lin. Algebra	D-MATH

Noten

ID	Legi	FachID	Note
1	12-333-333	1	6
2	13-222-222	3	5
4	13-222-222	1	5,5
5	14-444-444	2	3,5

Tabelle erstellen

Studierende

Vorname	Nachname	Legi	Wohnort

```
CREATE TABLE Studierende
( Vorname TEXT NOT NULL,
  Name TEXT NOT NULL,
  Legi TEXT NOT NULL,
  Wohnort TEXT NOT NULL,
  PRIMARY KEY(Legi)
);
```

Tabelle erstellen

Noten

ID	Legi	FachID	Note

```
CREATE TABLE Noten
( ID INTEGER NOT NULL CHECK(typeof(Id) = 'integer'),
  Legi TEXT NOT NULL,
  FachID INTEGER NOT NULL CHECK(typeof(Id) = 'integer'),
  Note REAL NOT NULL,
  PRIMARY KEY(ID),
  FOREIGN KEY(Legi) REFERENCES Studierende(Legi)
);
```

```
FOREIGN KEY(Legi) REFERENCES Studierende(Legi)
```

Studierende

Vorna	ame	Legi	Wohnort
H		12-333-333	Zürich
An		13-123-456	Luzern
Mic		13-222-222	Bern
Karl	ernans	14-444-444	Zürich

erlaubt die Kontrolle der **referentiellen Integrität**!

1

n

Noten

ID	Legi	FachID	Note
1	12-333-333	1	6
2	13-222-222	3	5
4	13-222-222	1	5,5
5	14-444-444	2	3,5

Datensatz in eine Tabelle einfügen

Studierende

Vorname	Nachname	Legi	Wohnort
Heinz	Meier	12-333-333	Zürich
Anne	Meier	13-123-456	Luzern
Michaela	Zbinden	13-222-222	Bern
Karl-Heinz	Meierhans	14-444-444	Zürich

```
INSERT INTO Studierende (Vorname, Nachname, Legi, Wohnort)
VALUES ('Heinz', 'Meier', '12-333-333', 'Zürich');
```

- Bei **Zahlen** sind keine Gänsefüsschen erlaubt!
  `INSERT INTO Noten (ID) VALUES (1)`

Einfachste Version einer SQL-Abfrage

```
SELECT *
FROM Studierende;
```

→ liefert alle Einträge der Tabelle "Studierende"

- Wildcard-Zeichen (*) bedeutet:
  "alle Attribute dieser Tabelle"
- Semikolon (;) schliesst eine Abfrage ab

Wahl der anzuzeigenden Attribute

```
SELECT Vorname, Nachname
FROM Studierende;
```

→ liefert von allen Tuples der Tabelle "Studierende" die Vor- und Nachnamen

Einschränken der Tuples mit WHERE

```
SELECT Vorname, Nachname
FROM Studierende
WHERE Nachname = 'Meier';
```

→ liefert nur jene Einträge der Tabelle "Studierende", die den Nachnamen 'Meier' tragen.

- WHERE erlaubt logische Operatoren z.B. AND/OR

Festlegen der Sortierreihenfolge

```
SELECT Vorname, Nachname
FROM Studierende
WHERE Nachname = 'Meier'
ORDER BY Vorname ASC;
```

→ liefert das Ergebnis aufsteigend sortiert nach Vornamen

Übung: einfache Abfragen I

Welche Resultate liefern folgende SQL-Abfragen?

```
SELECT *
FROM Faecher;
```

```
SELECT ID, Bezeichnung
FROM Faecher;
```

```
SELECT Bezeichnung;
FROM Faecher
ORDER BY Departement DESC;
```

Übung: einfache Abfragen I (Lösungen)

Welche Resultate liefern folgende SQL-Abfragen?

```
SELECT *
FROM Faecher;
```

ID	Bezeichnung	Departement
1	Botanik	D-BIOL
2	Statistik	D-MATH
3	Informatik	D-INFK
4	Lin. Algebra	D-MATH

```
SELECT ID, Bezeichnung
FROM Faecher;
```

ID	Bezeichnung
1	Botanik
2	Statistik
3	Informatik
4	Lin. Algebra

```
SELECT Bezeichnung
FROM Faecher
ORDER BY Departement DESC;
```

Bezeichnung
Lin. Algebra
Statistik
Informatik
Botanik

Übung: einfache Abfragen II

Welche Resultate liefern folgende SQL-Abfragen?

```
SELECT Legi, Vorname, Nachname
FROM Studierende
WHERE Nachname = 'Brunner';
```

```
SELECT count(Legi) AS Anzahl
FROM Studierende
WHERE Wohnort = 'Zürich';
```

Übung: einfache Abfragen II (Lösungen)

Welche Resultate liefern folgende SQL-Abfragen?

```
SELECT Legi, Vorname, Nachname
FROM Studierende
WHERE Nachname = 'Brunner';
```

Legi	Vorname	Nachname

```
SELECT count(Legi) AS Anzahl
FROM Studierende
WHERE Wohnort = 'Zürich';
```

Anzahl
2

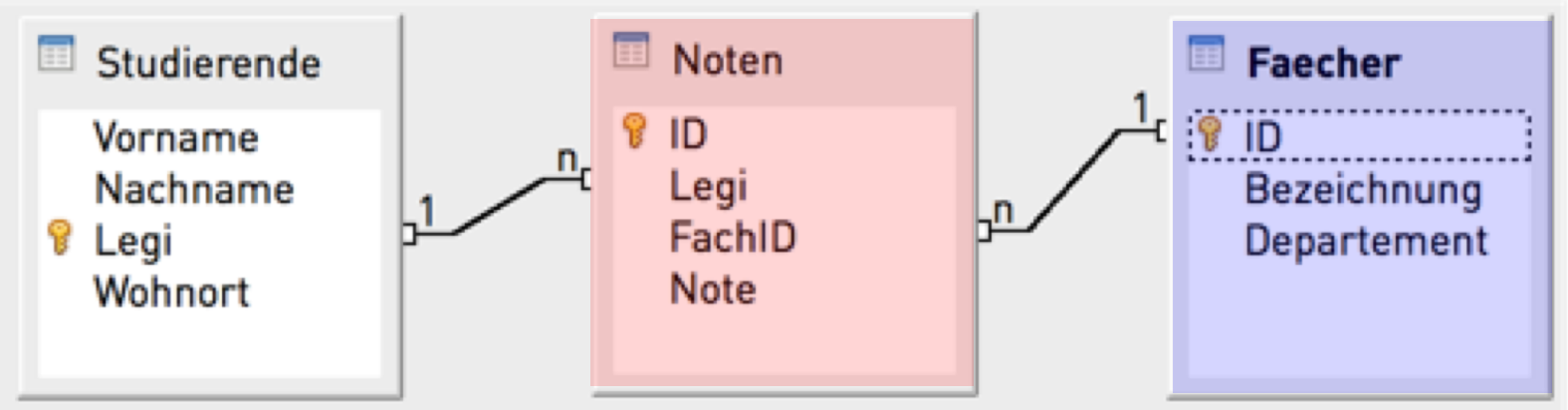
SQL: einfache Abfragen III (Übung)

Schreiben Sie folgende SQL-Query:

Liste mit allen Studierenden (Legi, Vorname, Nachname), die in Bern beziehungsweise in Luzern wohnen, sortiert nach Nachnamen von A bis Z.

Legi	Vorname	Nachname
13-123-456	Anne	Meier
13-222-222	Michaela	Zbinden

Abfrage über zwei Tabellen hinweg




```
SELECT *
FROM Noten, Faecher;
```

ID	Legi	FachID	Note	ID	Bezeichnung	Departement
1	12-333-333	1	6	1	Botanik	D-BIOL
2	13-222-222	3	5	3	Informatik	D-INFK
4	13-222-222	1	5,5	1	Botanik	D-BIOL
5	14-444-444	2	3,5	2	Statistik	D-MATH

erwartetes Resultat

ID	Legi	FachID	Note	ID	Bezeichnung	Departement
1	12-333-333	1	6	1	Botanik	D-BIOL
2	13-222-222	3	5	3	Informatik	D-INFK
4	13-222-222	1	5,5	1	Botanik	D-BIOL
5	14-444-444	2	3,5	2	Statistik	D-MATH

erhaltenes Resultat

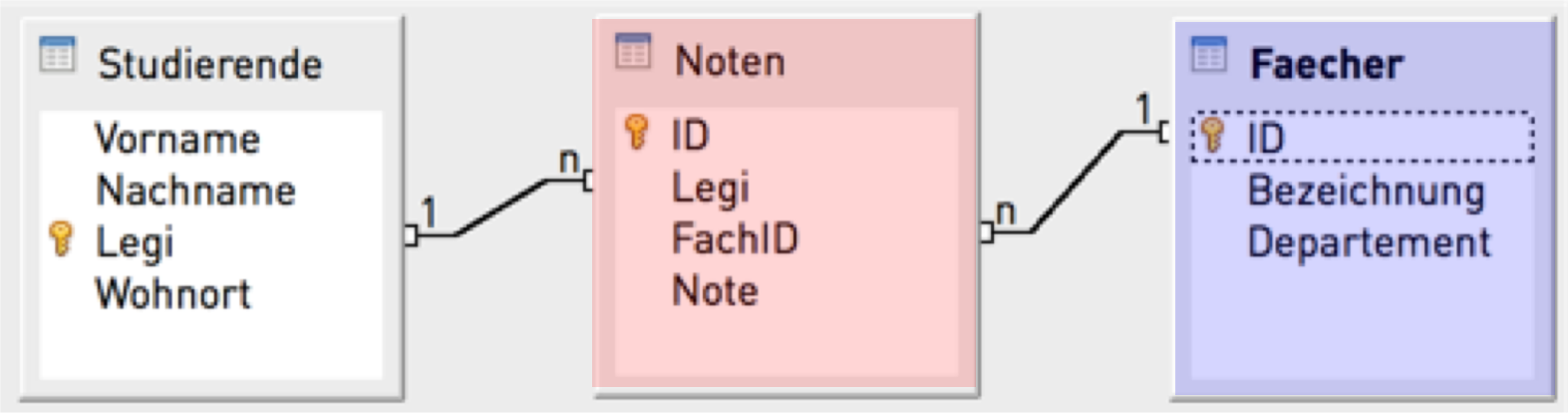
ID	Legi	FachID	Note	ID	Bezeichnung	Departement
1	12-333-333	1	6	1	Botanik	D-BIOL
1	12-333-333	1	6	2	Statistik	D-MATH
1	12-333-333	1	6	3	Informatik	D-INFK
1	12-333-333	1	6	4	Lin. Algebra	D-MATH
2	13-222-222	3	5	1	Botanik	D-BIOL
2	13-222-222	3				D-MATH
2	13-222-222	3				D-INFK
2	13-222-222	3				D-MATH
4	13-222-222	1	5,5	1	Botanik	D-BIOL
4	13-222-222	1	5,5	2	Statistik	D-MATH
4	13-222-222	1	5,5	3	Informatik	D-INFK
4	13-222-222	1	5,5	4	Lin. Algebra	D-MATH
5	14-444-444	2	3,5	1	Botanik	D-BIOL
5	14-444-444	2	3,5	2	Statistik	D-MATH
5	14-444-444	2	3,5	3	Informatik	D-INFK
5	14-444-444	2	3,5	4	Lin. Algebra	D-MATH

Kreuzprodukt

Aufteilen der Daten auf zwei Tabellen

ID	Legi	FachID
1	12-333-333	1
2	13-222-222	3
4	13-222-222	1
5	14-444-444	2

ID	Bezeichnung	Departement
1	Botanik	D-BIOL
3	Statistik	D-MATH
1	Informatik	D-INFK
2	Lin. Algebra	D-MATH

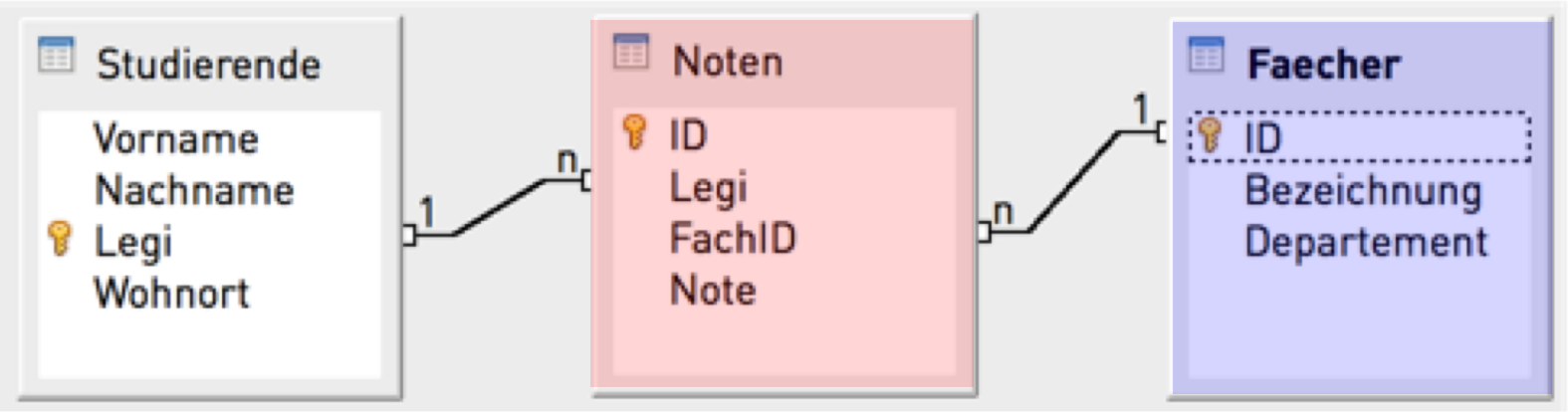


Beziehungen zwischen Tabellen berücksichtigen

ID	Legi	FachID	Note	ID	Bezeichnung	Departement
1	12-333-333	1	6	1	Botanik	D-BIOL
1	12-333-333	1	6	2	Statistik	D-MATH
1	12-333-333	1	6	3	Informatik	D-INFK
1	12-333-333	1	6	4	Lin. Algebra	D-MATH
2	13-222-222	3	5	1	Botanik	D-BIOL
2	13-222-222	3	5	2	Statistik	D-MATH
2	13-222-222	3	5	3	Informatik	D-INFK
2	13-222-222	3	5	4	Lin. Algebra	D-MATH
4	13-222-222	1	5,5	1	Botanik	D-BIOL
4	13-222-222	1	5,5	2	Statistik	D-MATH
4	13-222-222	1	5,5	3	Informatik	D-INFK
4	13-222-222	1	5,5	4	Lin. Algebra	D-MATH
5	14-444-444	2	3,5	1	Botanik	D-BIOL
5	14-444-444	2	3,5	2	Statistik	D-MATH
5	14-444-444	2	3,5	3	Informatik	D-INFK
5	14-444-444	2	3,5	4	Lin. Algebra	D-MATH

FachID = ID

Korrekte Verknüpfung von *Noten* und *Faecher*




```
SELECT *
FROM Noten JOIN Faecher
     ON Noten.FachID = Faecher.ID;
```

Übung: komplexere Abfragen I

Schreiben sie die entsprechende SQL Abfrage

Die Nachnamen aller Studierenden, welche eine ungenügende Note erzielt haben.

Die Nachnamen und die Noten aller Studierenden mit der Note 6 oder mit dem Nachnamen 'Zbinden'.

Übung: komplexere Abfragen I (Lösungen)

Welche Resultate liefern folgende SQL-Abfragen?

Die Nachnamen aller Studierenden, welche eine ungenügende Note erzielt haben.

```
SELECT Studierende.Nachname
FROM Studierende JOIN Noten
   ON Studierende.Legi = Noten.Legi
WHERE Noten.Note < 4;
```

Übung: komplexere Abfragen I (Lösungen)

Welche Resultate liefern folgende SQL-Abfragen?

Die Nachnamen und die Noten aller Studierenden mit der Note 6 oder mit dem Nachnamen 'Zbinden'.

```
SELECT Studierende.Nachname, Noten.Note
FROM Studierende JOIN Noten
   ON Studierende.Legi = Noten.Legi
WHERE Noten.Note = 6
   OR Studierende.Nachname = 'Zbinden'
```

Kürzel für Tabellen verwenden

Welche Resultate liefern folgende SQL-Abfragen?

Die Nachnamen und die Noten aller Studierenden mit der Note 6 oder mit dem Nachnamen 'Zbinden'.

```
SELECT s.Nachname, n.Note
FROM Studierende s JOIN Noten n
    ON s.Legi = n.Legi
WHERE n.Note = 6
    OR s.Nachname = 'Zbinden'
```

Verknüpfen mit JOIN...ON statt WHERE

```
SELECT *
FROM Noten JOIN Faecher
   ON Noten.FachID = Faecher.ID;
```

```
SELECT *
FROM Noten, Faecher
WHERE Noten.FachID = Faecher.ID;
```

Abfrage ohne JOIN (dafür mit WHERE)

```
SELECT Studierende.Nachname, Noten.Note
FROM Studierende JOIN Noten
   ON Studierende.Legi = Noten.Legi
WHERE Noten.Note = 6
   OR Studierende.Nachname = 'Zbinden';
```

```
SELECT Studierende.Nachname, Noten.Note
FROM Studierende, Noten
WHERE Studierende.Legi = Noten.Legi
  AND (    Noten.Note = 6
        OR Studierende.Nachname = 'Zbinden');
```

Welches Resultat erhalten Sie, wenn Sie die Klammer weglassen?

```
SELECT Studierende.Nachname, Noten.Note
FROM Studierende, Noten
WHERE Studierende.Legi = Noten.Legi
  AND (    Noten.Note = 6
        OR Studierende.Nachname = 'Zbinden');
```

Rangfolge unterschiedlicher Operatoren

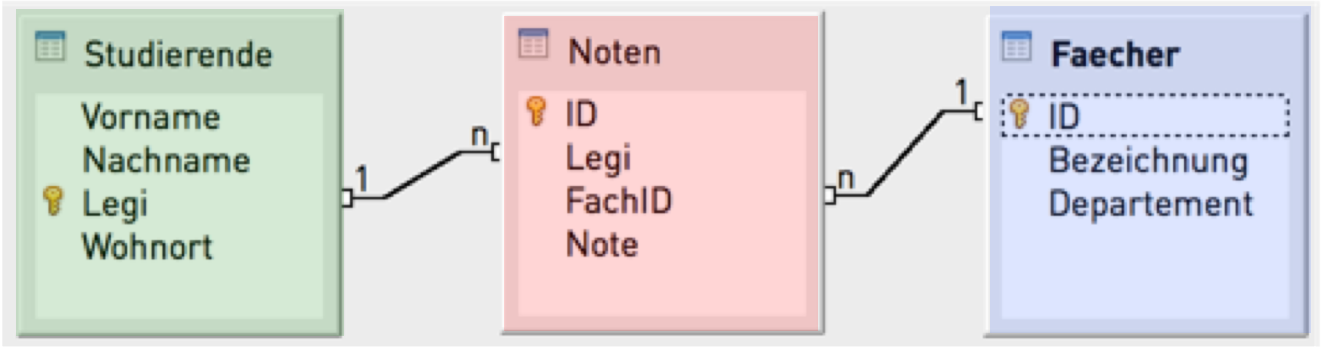
1. Arithmetische Operatoren (Punkt vor Strich)
2. Relationale Operatoren (Vergleichsoperatoren)
3. Logische Operatoren:
    1. Negation (NOT)
    2. Konjunktion (AND)
    3. Disjunktion (OR)

Das OR macht die ganze Arbeit kaputt!

ID	Legi	FachID	Note	Vorname	Nachname	Legi	Wohnort
1	12-333-333	1	6	Heinz	Meier	12-333-333	Zürich
1	12-333-333	1	6	Anne	Meier	13-123-456	Luzern
1	12-333-333	1	6	Michaela	Zbinden	13-222-222	Bern
1	12-333-333	1	6	Karl-Heinz	Meierhans	14-444-444	Zürich
2	13-222-222	3	5	Heinz	Meier	12-333-333	Zürich
2	13-222-222	3	5	Anne	Meier	13-123-456	Luzern
2	13-222-222	3	5	Michaela	Zbinden	13-222-222	Bern
2	13-222-222	3	5	Karl-Heinz	Meierhans	14-444-444	Zürich
4	13-222-222	1	5.5	Heinz	Meier	12-333-333	Zürich
4	13-222-222	1	5.5	Anne	Meier	13-123-456	Luzern
4	13-222-222	1	5.5	Michaela	Zbinden	13-222-222	Bern
4	13-222-222	1	5.5	Karl-Heinz	Meierhans	14-444-444	Zürich
5	14-444-444	2	3.5	Heinz	Meier	12-333-333	Zürich
5	14-444-444	2	3.5	Anne	Meier	13-123-456	Luzern
5	14-444-444	2	3.5	Michaela	Zbinden	13-222-222	Bern
5	14-444-444	2	3.5	Karl-Heinz	Meierhans	14-444-444	Zürich

Übung: Abfrage über drei Tabellen

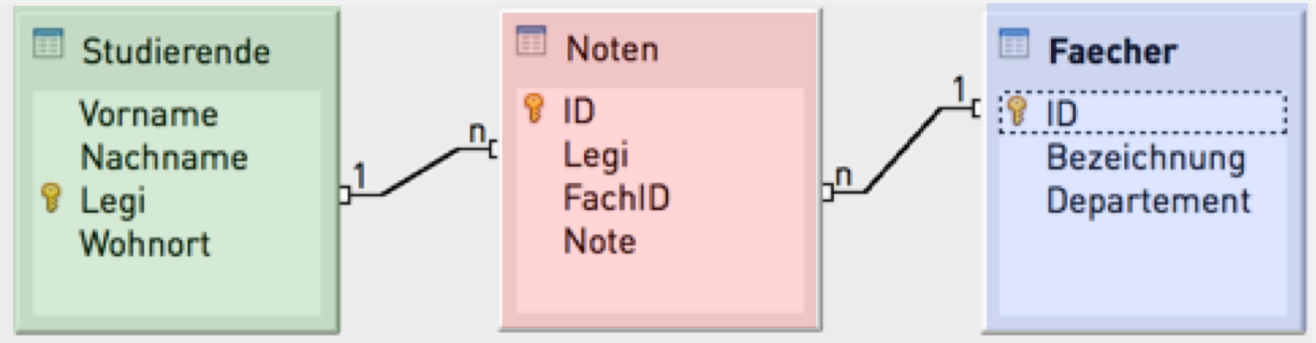
Legi	Note	Bezeichnung
12-333-333	4,5	Botanik
14-444-444	3,5	Statistik



„Meier“ im Namen

```
SELECT Studierende.Nachname, Noten.Note,
Faecher.Bezeichnung
FROM Studierende JOIN Noten
    ON Studierende.Legi = Noten.Legi
    JOIN Faecher
    ON Noten.FachID = Faecher.ID
WHERE Studierende.Nachname LIKE 'Meier%';
```

Übung: Umformen in ein WHERE-Statement




```
SELECT Studierende.Legi, Noten.Note,
Faecher.Bezeichnung
FROM Studierende, Noten, Faecher
WHERE Studierende.Legi = Noten.Legi
  AND Noten.FachID = Faecher.ID
  AND Studierende.Nachname LIKE 'Meier%';
```

Referentielle Integrität von Datenbanken

Die Integrität einer Datenbank ist dann sichergestellt, wenn:

- Alle Primärschlüssel eindeutig sind
- Alle Fremdschlüssel auf einen Primärschüssel verweisen

Wenn die Integrität verletzt ist, sind die Daten in der Datenbank fehlerhaft!

Welche Tuples können gelöscht werden, ohne die referentielle Integrität zu verletzen?

Studierende

Vorname	Nachname	Legi	Wohnort
Heinz	Meier	12-333-333	Zürich
~~Anne~~	~~Meier~~	~~13-123-456~~	~~Luzern~~
Michaela	Zbinden	13-222-222	Bern
Karl-Heinz	Meierhans	14-444-444	Zürich

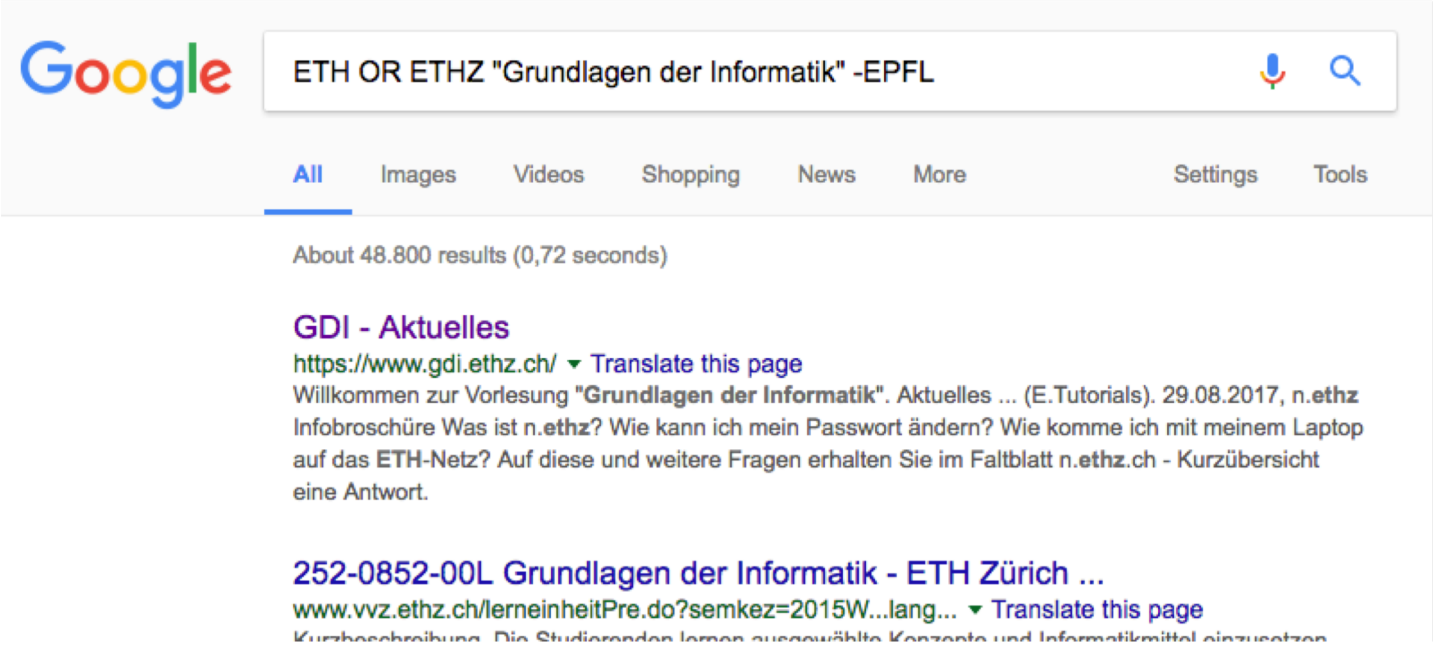
Faecher

ID	Bezeichnung	Departement
1	Botanik	D-BIOL
2	Statistik	D-MATH
3	Informatik	D-INFK
~~4~~	~~Lin. Algebra~~	~~D-MATH~~

Noten

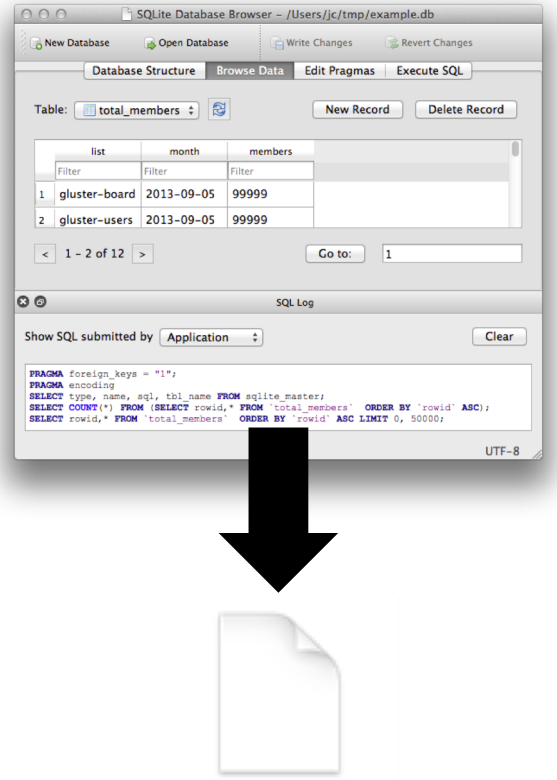
ID	Legi	FachID	Note
~~1~~	~~12-333-333~~	~~1~~	~~6~~
~~2~~	~~13-222-222~~	~~3~~	~~5~~
~~4~~	~~13-222-222~~	~~1~~	~~5,5~~
~~5~~	~~14-444-444~~	~~2~~	~~3,5~~

Google




```
SELECT URL, Seitentitel, substring(Seitentext,0,150)
FROM AlleWebseitenWeltweit
WHERE Seitentext like '%Grundlagen der Informatik%';
      AND Seitentext like '%ETH%'
      AND NOT Seitentext = 'EPFL';
```

Probleme aus dem Übungsbetrieb (Teil I)



sqlite-test.db

```
CREATE TABLE Noten
( ID INTEGER NOT NULL
  CHECK(typeof(Id) = 'Integer'),
  PRIMARY KEY(ID),
);
```

Probleme aus dem Übungsbetrieb (Teil II)

```
CREATE TABLE Studierende
( Name TEXT NOT NULL,
  Legi TEXT NOT NULL,
  PRIMARY KEY(Legi)
);
```

```
CREATE TABLE Noten
( ref_Studi TEXT NOT NULL,
  Note REAL NOT NULL,
  FOREIGN KEY(ref_Studi)
  REFERENCES Studierende(Legi),
  PRIMARY KEY(ref_Studi,Note)
);
```

```
INSERT INTO Noten(ref_Studi,Note) VALUES ('11-111-111',3.5)
   >> Query successful
INSERT INTO Noten(ref_Studi,Note) VALUES ('33-333-333',5.5)
   >> Foreign Key Constraint!
INSERT INTO Noten(ref_Studi,Note) VALUES ('11-111-111',3.5)
   >> Unique Key Constraint!
```

Finde die Fehler

```
select kunden.id
from kunden
where kunden.name = "Meier%"
```

1 Fehler

```
select kunden.name
from kunden, einkauf
where artikel = "Milch"
```

2 Fehler

```
select count(kunden.name) as Anzahl
from kunden inner join einkauf on
     kunden.id = einkauf.kunden_id
where einkauf.artikel like "Milch" OR "Reis"
```

1(+1) Fehler

Zusammenfassung

- CREATE TABLE legt in der DB eine neue Tabelle an
- Mit INSERT INTO können Datensätze in eine Tabelle eingefügt werden
- Mit SELECT können Sie Daten aus einer oder mehreren Tabellen abrufen
- JOIN…ON resp. FROM...WHERE führt Datensätze aus mehreren Tabellen zusammen
- Fehlende Beziehungen zwischen Tabellen liefern deren Kreuzprodukt
- Wenn die Referentielle Integrität einer Datenbank verletzt ist, kann das Resultat einer Abfrage falsch sein.