

Datenbank-Systeme

Technische und semantische Beziehungen modellieren
und implementieren

Internationaler Frauenstudiengang Informatik

Renzo Kottmann



This work is licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/).

Zusammenfassung der vorherigen Vorlesung

Constraints

- Datentypen
- Primary Keys
- NULL or NOT NULL Constraints
- DEFAULT VALUES
- CHECK Constraint
- UNIQUE Constraint
- CREATE DOMAIN

Vertiefung:

Einschränkung von Wertebereichen durch Aufzählungen

Arten von Daten

1. Zahlen (integer, numeric, double precision...)
2. Free Text (text, char)
3. Enumeration
4. Code-List
5. Komplexe Zusammensetzungen der oberen "einfachen Arten"
Z.B. Telefonnummern, Zeitstempel (Datum + Uhrzeit)...

Enumeration & Code List

1. **Enumeration:** Liste von Werten, die in Zukunft wenig bis gar nicht geändert wird
z.B. Geschlecht = {maennlich, weiblich}
2. **Code-List:** Liste von Werten, die in Zukunft häufig und zu jeder Zeit geändert wird
 - Und haeufig der eigentliche Wert codifiziert wird

Implementierungsvarianten

1. check constraint:

```
gender text check (gender in ['maennlich','weiblich'])
```

2. [Enumerated Types](#):

```
CREATE TYPE gender AS ENUM ('maennlich','weiblich');
```

3. Lookup-Tabelle

Implementierung durch Lookup-Tabelle

```
CREATE TABLE geschlecht (  
    name text PRIMARY KEY  
);  
INSERT INTO geschlecht  
VALUES ('maennlich'),('weiblich');  
  
CREATE TABLE teilnehmerin (  
    vorname person_name CHECK ( vorname != '' ),  
    nachname person_name,  
    matrikel_nr integer PRIMARY KEY,  
    email text NOT NULL CHECK ( email ~ '.*@.*' ) UNIQUE,  
    semester integer DEFAULT 3,  
    geschlecht text  
    REFERENCES geschlecht(name)  
);  
  
INSERT INTO teilnehmerin (geschlecht,vorname, nachname, matrikel_nr, email)  
VALUES ('maennlich','renzo','kottmann',007,'renzo@007.bond');
```

Foreign Key-Referenz auf Lookup-Tabelle

A [foreign key constraint](#) specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table.

We say this maintains the referential integrity between two related tables.

```
CREATE TABLE geschlecht (  
  name text PRIMARY KEY  
);  
  
CREATE TABLE teilnehmerin (  
  vorname person_name CHECK ( vorname != '' ),  
  nachname person_name,  
  matrikel_nr integer PRIMARY KEY,  
  email text NOT NULL CHECK ( email ~ '@' ),  
  semester integer DEFAULT 3,  
  geschlecht text  
  REFERENCES geschlecht(name)  
);
```


Foreign Key-Referenz auf Lookup-Tabelle

- Kann man erst Tabelle teilnehmerin und dann geschlecht anlegen?
- Was passiert, wenn in teilnehmerin geschlecht null ist?
- Was passiert, wenn in geschlecht die Zeile maennlich geloescht wird?
- Muss name in Tabelle geschlecht Ein-Eindeutig sein?

```
CREATE TABLE geschlecht (  
  name text PRIMARY KEY  
);  
  
CREATE TABLE teilnehmerin (  
  vorname person_name CHECK ( vorname != ''),  
  nachname person_name,  
  matrikel_nr integer PRIMARY KEY,  
  email text NOT NULL CHECK ( email ~ '@' ),  
  semester integer DEFAULT 3,  
  geschlecht text  
    REFERENCES geschlecht(name)  
);
```

Foreign Key Verhalten bei Datenänderung

- Was soll mit Einträgen in teilnehmerin passieren wenn ein FOREIGN KEY (also ein Eintrag in geschlecht geändert wird?)
 - Verhindern oder alle Einträge in teilnehmerin auch ändern?

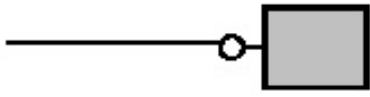
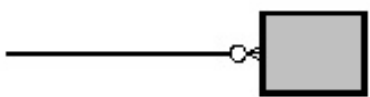
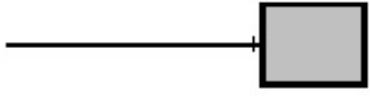
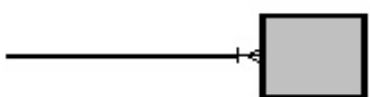
```
CREATE TABLE geschlecht (  
  name text PRIMARY KEY  
);  
  
CREATE TABLE teilnehmerin (  
  vorname person_name CHECK ( vorname != ''),  
  nachname person_name,  
  matrikel_nr integer PRIMARY KEY,  
  email text NOT NULL CHECK ( email ~ '@' ),  
  semester integer DEFAULT 3,  
  geschlecht text  
    REFERENCES geschlecht(name)  
);
```

Entity Relationships (Beziehungen) revisited

Verschiedene Entitäten können zueinander in Beziehung gesetzt werden.

- In jeder Beziehung haben Entitäten gewisse Rollen
- Beziehungen können Eigenschaften (Attribute) haben
- Beziehungen haben Kardinalitäten

Notationen

(1)	(2)	(3)	(4)
0..1		1	c
0..*		N	mc
1..1		1	1
1..*		N	m

1. UML
2. Crow Foot
3. Darstellung nach Chen-Notationen
4. Darstellung nach [Zehnder](#)
aus [Matthiesen et al.](#)

Agile änderung

Wie ändert sich das ERM und die implementierung wenn folgende Anforderung hinzukommt:

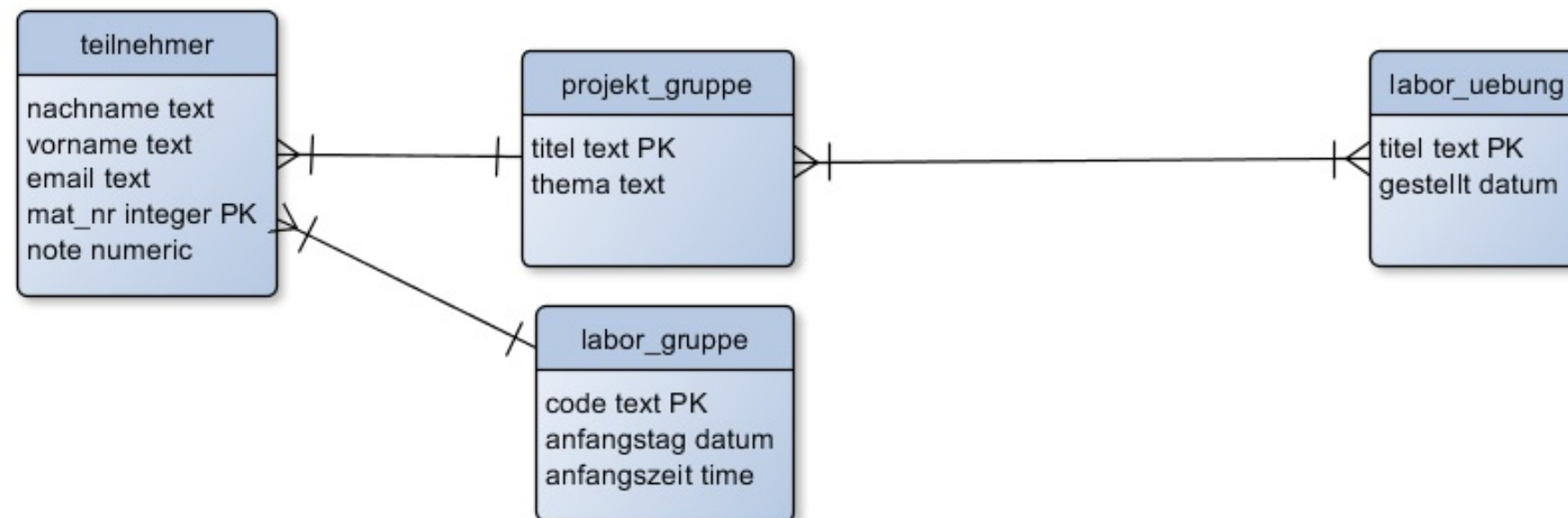
- Die Datenbank soll für alle vergangenen und zukünftigen Datenbankkurse informationen speichern können

Whiteboard

ERD und Beziehungen von Teilnehmerin Datenbank

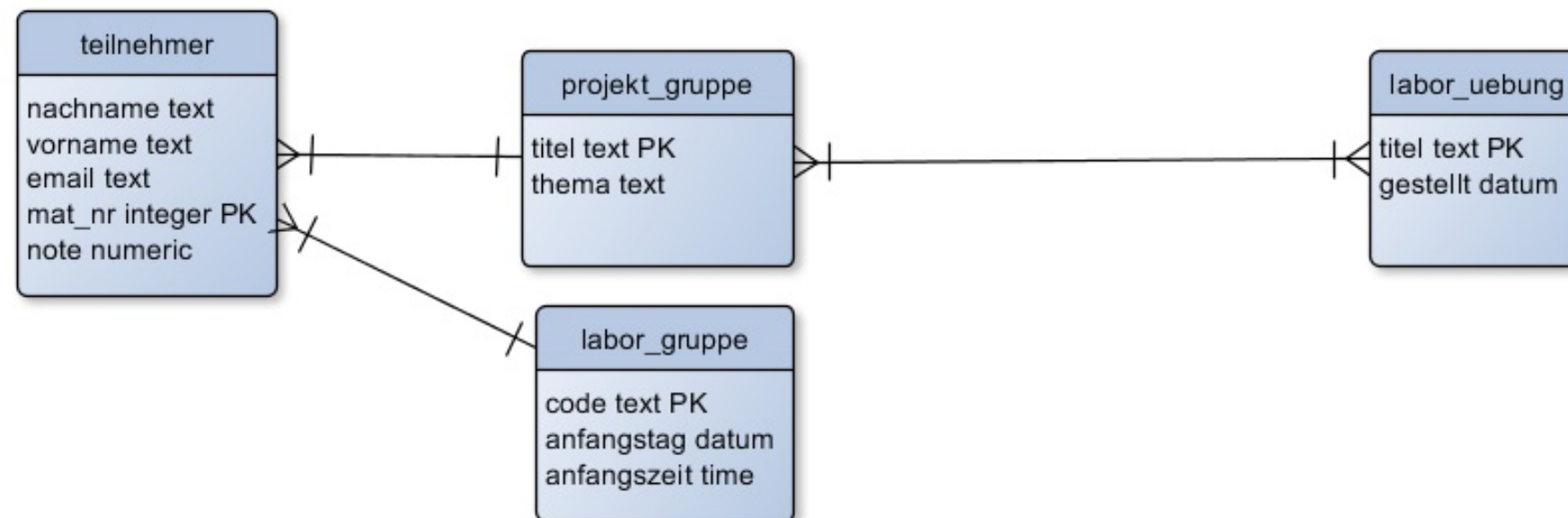
Beziehungen

- Teilnehmer ist in einer Projektgruppe und einer Laborgruppe
- Jede Projektgruppe bearbeitet mehrere Laboruebungen



Beziehungen mit Foreign Keys

- Bestimmen die
 - Attribute welche die Beziehungen identifizieren
 - Kardinalitäten



Foreign Keys (one to many)

- Können nur eins zu viele Beziehungen umsetzen
- Werden immer von viele zu eins gesetzt

```
CREATE TABLE labor_gruppe (  
  code text  
    PRIMARY KEY  
    CHECK ( code IN ('w','x','y','z') ) ,  
  -- Attribute fehlen  
  anfangszeit time  
    NOT NULL  
);  
  
CREATE TABLE teilnehmer (  
  vorname text  
    CHECK ( vorname != '' ),  
  nachname text,  
    CHECK ( vorname != '' ),  
  matrikel_nr integer  
    PRIMARY KEY,  
  -- Attribute fehlen  
  labor text  
    REFERENCES labor_gruppe(code),  
  geschlecht text  
    REFERENCES geschlecht(name)  
    ON UPDATE CASCADE  
);
```


Foreign Keys (one to many)

Frage

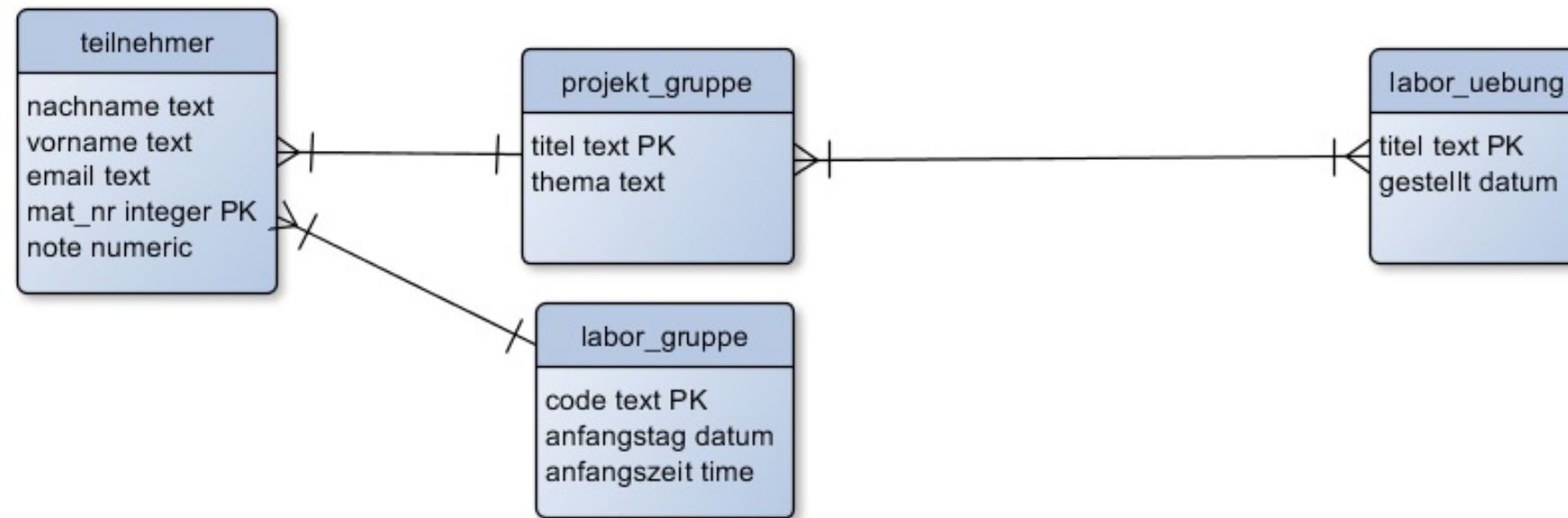
Wird hier 0..1 zu N

oder

1 zu N umgesetzt?

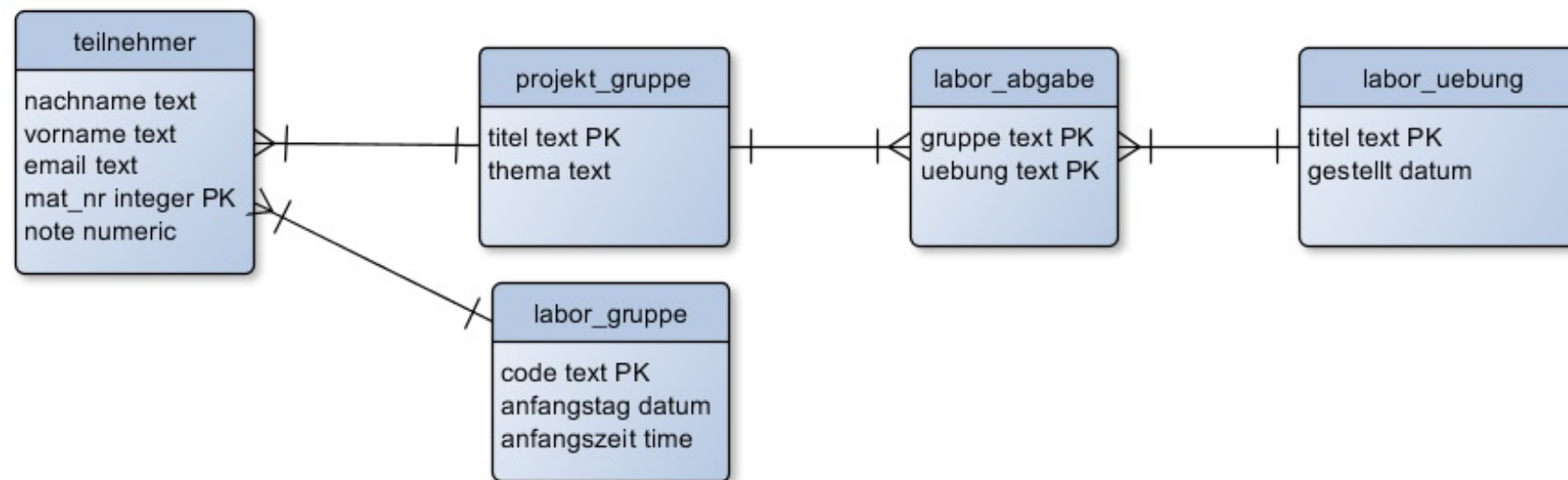
```
CREATE TABLE labor_gruppe (  
  code text  
    PRIMARY KEY  
    CHECK ( code IN ('w','x','y','z') ) ,  
  -- Attribute fehlen  
  anfangszeit time  
    NOT NULL  
);  
  
CREATE TABLE teilnehmer (  
  vorname text  
    CHECK ( vorname != '' ),  
  nachname text,  
    CHECK ( vorname != '' ),  
  matrikel_nr integer  
    PRIMARY KEY,  
  -- Attribute fehlen  
  labor text  
    REFERENCES labor_gruppe(code),  
  geschlecht text  
    REFERENCES geschlecht(name)  
    ON UPDATE CASCADE  
);
```

Foreign Keys (many to many)



Foreign Keys (many to many)

- Umsetzung durch neue "Beziehungs"-Relation



Foreign Keys (many to many)

- Neue Tabelle, die auf die beiden existierenden referenziert
 - Primary Key der neuen Tabelle ist Kombination der PKs der existierenden Tabellen

```
CREATE TABLE projekt_gruppe (  
  titel text  
    PRIMARY KEY  
    CHECK ( titel != '' ) ,  
  thema text  
    NOT NULL  
    DEFAULT ''  
);  
  
CREATE TABLE labor_uebung (  
  titel text  
    PRIMARY KEY,  
  gestellt date  
    NOT NULL  
    UNIQUE  
);  
  
CREATE TABLE labor_abgabe (  
  gruppe text  
    REFERENCES projekt_gruppe(titel),  
  uebung text  
    REFERENCES labor_uebung(titel),  
  
  PRIMARY KEY (gruppe,uebung)  
);
```

Danke für die Zusammenarbeit