

# Datenbankbasierte Web-Anwendungen

## Entity Relationship Modellierung (ERM) & Structured Query Language (SQL)

Medieninformatik SoSe 2017

Renzo Kottmann



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

# Kontakt:

- mail
- linkedin:<http://www.linkedin.com/in/renzokottmann>
- twitter: @renzokott

# Schritte im ad-hoc Projekt

Datenverwaltungssystem: semistrukturierte Daten im Filesystem (CSV)

1. Problem in der realen Welt
  - Wer sind die Teilnehmerinnen in meinem Kurs?
2. Implizites Model (Tabelle)
3. Datenerfassung
  - Liste der Teilnehmerinnen

**Was sind die Nachteile von diesem System  
und dieser Vorgehensweise?**

**Was wäre besser?**

# Was wäre besser?

1. Hohe Datenintegrität bzw. Konsistenz
2. Mehrere Nutzer können gleichzeitig an denselben Daten arbeiten
3. Klare Trennung von Struktur und Inhalt bzw. Daten
4. Effiziente Suche in grossen Daten

# Für Datenintegrität und Strukturtrennung

- Strukturierte Daten
- (relationale) Datenbank (DB)
  - Schema („Modell“, „Metadaten“)
  - + Daten („Extension“, „Nutzdaten“)
  - Modell und Daten sind konzeptionell getrennt!

# Für Effizienz und Mehrbenutzer

- Datenbankmanagementsystem (DBMS)
- Softwaresystem
  - Ermöglicht Erstellung und Pflege vieler Datenbanken
  - Stellt Werkzeuge für *alle* Aspekte der Datenverwaltung bereit



## 2. Runde

Mit dem relationalem Ziel vor Auge

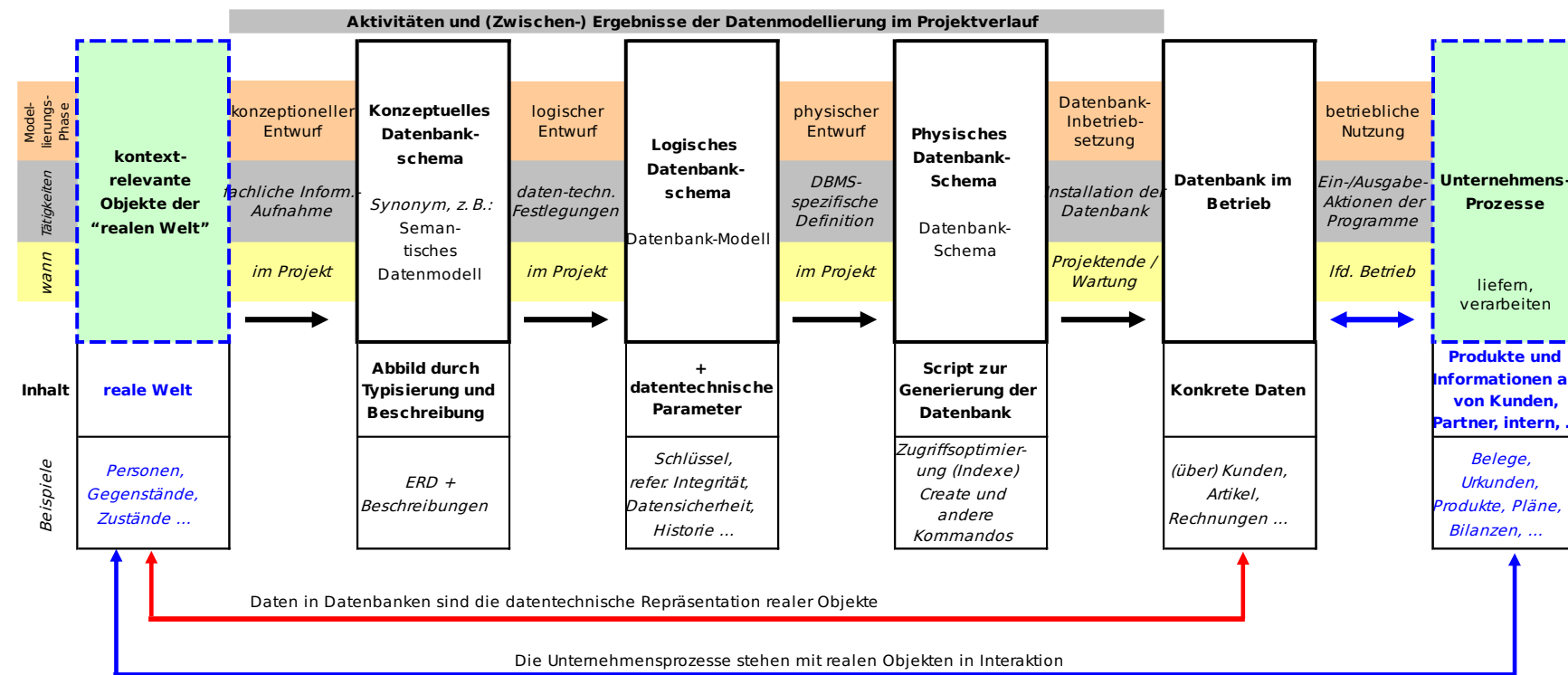
# Schritte im agilen Datenbank Projekt

Datenverwaltungssystem: Relationales Datenbankmanagementsystem

1. Anwendungsdefinition
  - Genaue textuelle Beschreibung des Problems in der realen Welt.
2. Konzeptioneller Entwurf / Datenbankmodellierung
  - Entity Relationship Modellierung (ERM)
3. Datenbank Realisierung
  - Skript(e) zur Implementierung des Datenmodells
  - Skript(e) zum laden/einfügen der Daten
4. Von Geschäftsanfragen zu Datenbank-Abfragen
  - Skript(e) für SQL 's SELECTs

# Alternativer Blick: Datenbankentwicklungszyklus

Modellieren: Entwicklung von der fachlichen, implementierungsunabhängigen Konzeption bis zur Datenbank



[DatMod v semMod zur DBK](#) von [VÖRBY](#), Konvertierung zu SVG [Perhelion](#) - eigene Erstellung, aus Wikipedia-Text abgeleitet. Lizenziert unter Gemeinfrei über [Wikimedia Commons](#).

# Anwendungsdefinition

In der Phase der Systemanalyse werden ausgehend von der Problemstellung die Anforderungen an die Lösung formuliert. Diese sollten möglichst vollständig und konsistent sein, d.h. alle (vollständigen) Anforderungen sollten widerspruchsfrei (konsistent) formuliert werden.

Ein guter Einstieg in die Anwendungsdefinition ist, sich zu verdeutlichen, welche genaueren Zwecksetzungen erfüllt werden sollen. D.h. was ist das Ziel des Projektes.

# Anwendungsdefinition

Welchen Zwecken soll die Erfassung aller Kursteilnehmer dienen?

# Anwendungsdefinition

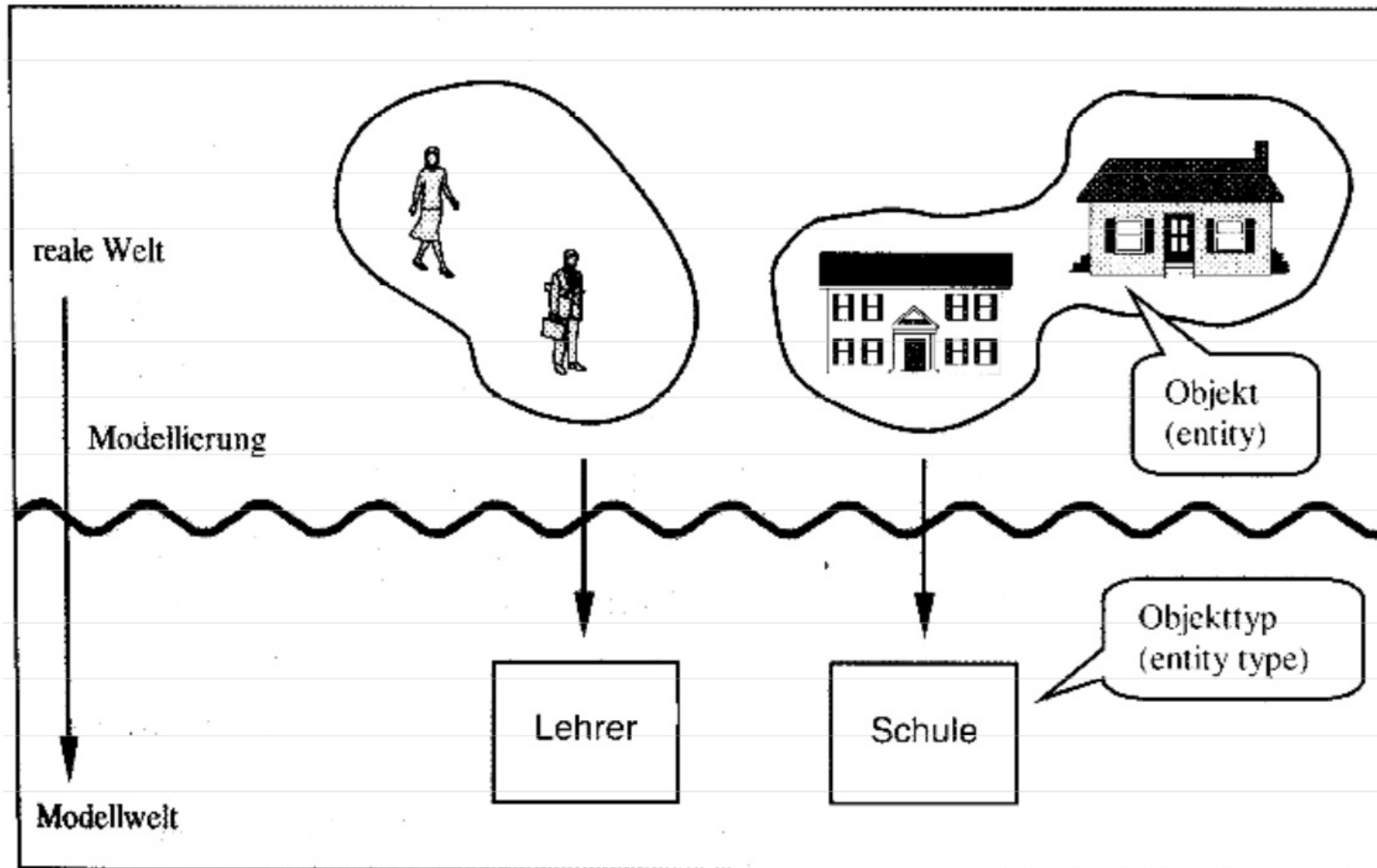
Welchen Zwecken soll die Erfassung aller Kursteilnehmer dienen?

- Notenvergabe
- Ermittlung des Wissensstands
- Wer macht welches Datenbankprojekt?
- Wer ist in welcher Laborgruppe?
- Wer hat welche Laboruebung, wann fertig gemacht?

# Mini-Welt

Da wir wissen was wir wollen, können wir folgende Fragen beantworten:

- Welchen Ausschnitt der realen Welt brauchen wir?
- Welche Aspekte müssen wir Berücksichtigen?





# Entity Relationship Modellierung (ERM)

- Semantischer Datenbankentwurf
  - unabhängig von konkreten Datenbank-spezifischen Modellen
- Graphisch
- Idee simpel
  - Leider sehr viele inkonsistente Varianten

# Entity (Entität)

- Ein Entity-Relationship-Model (ERM) geht von Entitäten ( $\sim$ = Objekten) aus.  
"Eine Entität ist eine eigenständige Einheit, die im Rahmen des betrachteten Modells **eindeutig identifiziert** werden kann."
- Ein Entitätstyp wird durch Attribute genauer beschrieben und stellt somit eine abstrakte Beschreibung oder Charakterisierung von Entitäten da.
- Beispiel:  
Lehrer = Entitätstyp  
Renzo  $\sim$ = Entität (ein spezieller Lehrender)

# Attribute

- Eigenschaften von Entitäten werden durch Attribute beschrieben
- Attribute haben einen Namen und eine Domaine (= Bestimmung der Wertmenge).

# Keys (Schlüssel)

Da die Definition einer Entität beinhaltet, dass diese zumindest im Rahmen eines Modells eindeutig identifiziert werden kann, braucht jeder Entitätstyp eine Menge von Attributen als Schlüssel.

Die Auswahl eines oder mehrerer Attribute als Schlüssel legt fest, dass es keine zwei Entitäten eines Entitätstyp geben kann die identische Attributwerte haben.

- Wichtige Eigenschaften:
  - Eindeutigkeit
  - Zuteilbarkeit

# Relationship (Beziehung)

Verschiedene Entitäten können zueinander in Beziehung gesetzt werden.


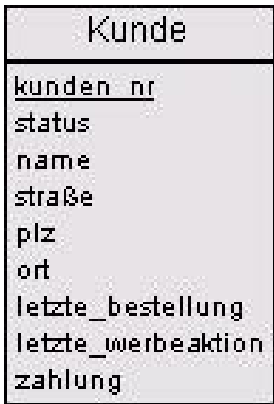
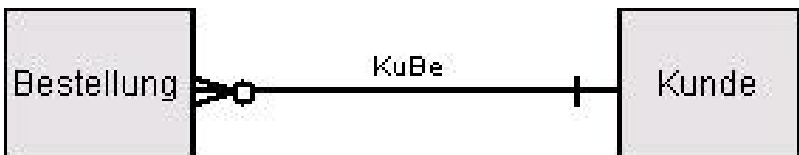
- In jeder Beziehung haben Entitäten gewisse Rollen
- Beziehungen können Eigenschaften (Attribute) haben
- Beziehungen haben Kardinalitäten

# Notationen

Es gibt verschiedene Formen ERM zu notieren (textuell und/oder graphisch):

- [Chen Notation](#)
- [Crow Foot's](#)
- [Unified Modelling Language](#) (UML)

# Notation

Entity	
Attribute	
Beziehungen/Relationship	

# Von ERM zur ersten Implementation



# Structured Query Language (SQL)

SQL ist eine Datenbanksprache

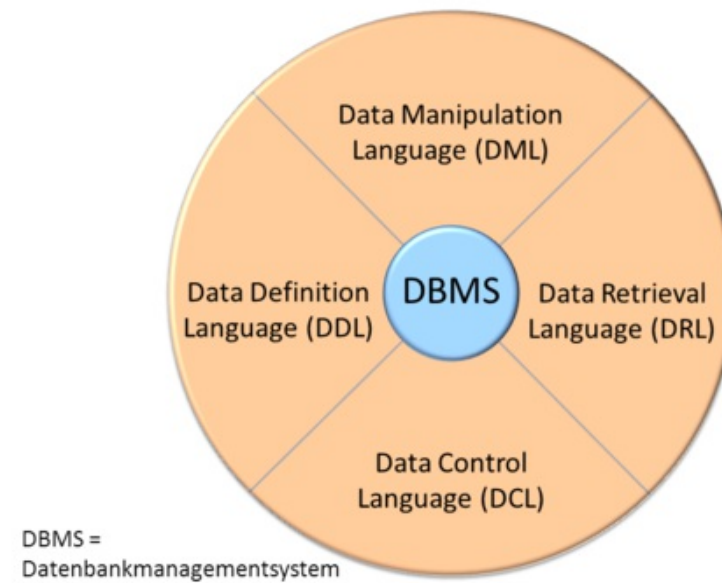
1. zur Definition von Datenstrukturen/Modellen
2. zum Bearbeiten (Einfügen, Verändern, Löschen)
3. zum Abfragen von darauf basierenden Datenbeständen
4. zur Rechtevergabe

# SQL Eigenschaften

- basiert auf relationaler Algebra
- an English angelehnt
- Deklarativ und funktional
- Fast alle Datenbanken verstehen SQL
- Standardisiert
  - PostgreSQL hat einer der besten Umsetzungen

# SQL Überblick

## Structured Query Language (SQL)



- DDL = Data Defintion Language:

Definition des Datenbankschemas

- DML = Data Manipulation Language:

Ändern, Einfügen, Löschen und lesender Zugriff

- DRL = Data Retrievel Language

Nicht standadisierte Bezeichnung des SELECT aus DML

- DCL = Data Control Language:

Rechteverwaltung und Transaktionskontrolle

# DDL: Create Table

Teilnehmerin
Vorname
Nachname
Matrikel Nummer
email
Semester

```
CREATE TABLE teilnehmer (  
  --Spalten Name Datentyp,  
  vorname text,  
  nachname text,  
  matrikel_nr integer,  
  email text,  
  semester integer  
);
```

s. <http://www.postgresql.org/docs/9.3/interactive/ddl-basics.html> und

<http://www.postgresql.org/docs/9.3/interactive/sql-createtable.html>

# DDL: Create Table Primary Key

Teilnehmerin
Vorname
Nachname
Matrikel Nummer
email
Semester

```
CREATE TABLE teilnehmer (  
  --Spalten Name Datentyp,  
  vorname text,  
  nachname text,  
  matrikel_nr integer,  
  email text,  
  semester integer  
);
```

# DDL: Create Table Primary Key

Teilnehmerin
Vorname
Nachname
Matrikel Nummer
email
Semester

```
CREATE TABLE teilnehmer (  
  --Spalten Name Datentyp,  
  vorname text,  
  nachname text,  
  -- Simpler (nicht bester Primary Key)  
  matrikel_nr integer PRIMARY KEY,  
  email text,  
  semester integer  
);
```

# DML: Daten Einfügen

Teilnehmerin
Vorname
Nachname
Matrikel Nummer
email
Semester

```
INSERT INTO teilnehmer  
  (vorname, nachname, matrikel_nr, email, semester)  
VALUES  
  ('renzo', 'kottmann', 007, 'renzo@007.bond', 0);
```

s. <http://www.postgresql.org/docs/9.6/interactive/dml-insert.html> und  
<http://www.postgresql.org/docs/9.3/interactive/sql-insert.html>

# Weiterfuehrende Fragen:

1. Wie ändert sich das ERM und die implementierung wenn folgende Anforderng hinzukommt:
  - Die Datenbank soll für alle vergangenen und zukünftigen Datenbankkurse informationen speichern können
2. Welche Datentypen gibt es schon in PostgreSQL?
3. Kann man eigene Datentypen definieren?
  - Wenn ja, welche Möglichkeiten gibt es?
4. Welche weiteren SQL-Befehle für Datenmodell-Management (DDL) gibt es noch?



**Danke fuer die Zusammenarbeit**