# Datenbanken
# Grundlagen

- Dozent: Diana Cristea
- E-mail: dianat [at] cs.ubbcluj.ro
- Website: www.cs.ubbcluj.ro/~dianat/

- Fragen und Feedback sind immer erwünscht: per e-mail oder per persönlichem Gespräch
- Anonymes Feedback möglich: auf meine Website
- Kursanforderungen - http://www.cs.ubbcluj.ro/~dianat/bd.php

# E-learning-Plattformen, Technologien

- Website: www.cs.ubbcluj.ro/~dianat/ – für Nachrichten, Kursanforderungen
- Microsoft Teams (neue Accounts) – für live Meetings
- Moodle: https://moodle.cs.ubbcluj.ro/ , Kurs “BD IG - Datenbanken” – für Nachrichten, Quiz, Hausaufgaben, Diskussionen Forum
- Bitte folge Nachrichten auf der Website und auf Moodle für Updates

# Moodle

- Aufpassen! Ab nächste Woche wird die **Anwesenheit auf Moodle** überprüft. Ihr müsst bis dann alle in dem Moodle BD Kurs angemeldet sein!

# Coursera

- https://coursera.org

# Struktur

- Vorlesung: jede Woche (2 Std)
- Praktische Übungen: jede Woche (2 Std)
- Seminar: jede 2te Woche (2 Std)

# Anwesenheiten

- Um die Prüfung schreiben zu dürfen:
  - Wenigstens 12 **aktive** Anwesenheiten bei den Übungen/Labor
  - Wenigstens 5 **aktive** Anwesenheiten bei dem Seminar

- Was heißt **aktive** Anwesenheit?
  - Eine aktive Anwesenheit wird durch unterschiedliche Aktivitäten bewiesen, wie z.B.:
    - Hausaufgabe
    - Quiz
    - u.a.
  - **Aufpassen!** Das einfache Teilnehmen in einem Call heißt **NICHT** active Anwesenheit!
- Während jeder Labor/Seminar-Stunde wird die Aktivität erklärt, die nötig ist für die Anwesenheit.
- Die Anwesenheit kann nur während der entsprechenden Stunde ausgeführt werden!
- **Bem**. Auch Studenten aus dem dritten Jahr (oder höher) brauchen aktive Anwesenheiten!

# Noten & Klausur

- Labor-Hausaufgaben – 25%
  - Bei den praktischen Aufgaben muss die **Mindestnote 5** erzielt werden um die Prüfung schreiben zu dürfen

- Praktisches Test: während der Prüfungszeit – 25%
  - Bei den praktischen Test muss die **Mindestnote 5** erzielt werden um die Prüfung schreiben zu dürfen

- Schriftliche Prüfung: während der Prüfungszeit – 50%
  - Für das Bestehen der Prüfung: **Mindestnote 5** bei der schriftlichen Prüfung!

# Noten & Klausur

- Laboraufgaben:
  - Nach dem Deadline gibt es zwei Wochen Verlängerungszeitraum, in welcher man die Aufgabe abgeben kann
  - Für jede Verspätungswoche werden 2p abgezogen, d.h.
    - In der **erste Verspätungswoche** kann man **höchstens Note 8** bekommen
    - In der **zweiten Verspätungswoche** kann man **höchstens Note 6** bekommen
    - **Ab der dritten Verspätungswoche** kann man die Aufgabe nicht mehr abgeben und die Note für die entsprechende Hausaufgabe ist **1**
  - Um ein Aufgabe abzugeben muss man **die Lösung erklären**!

# Folien, Literatur

- Folien und andere Informationen zur Vorlesung, Seminar und Übungen werden auf Teams oder Moodle zur Verfügung gestellt

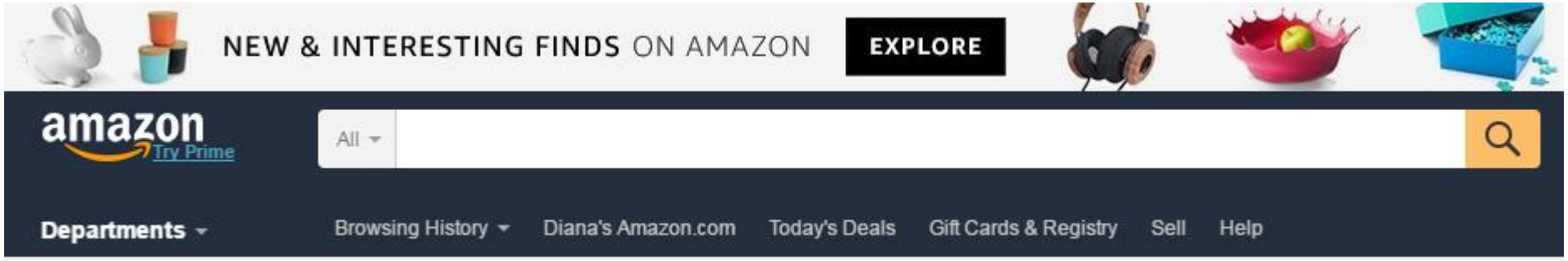
- Literatur:
  - **A. Kemper , A. Eickler. Datenbanksysteme – Eine Einführung. Oldenbourg Verlag, 2015. 10. Auflage.**
  - **A. Kemper, M. Wimmer. Übungsbuch Datenbanksysteme. Oldenbourg Verlag, 3. Auflage, 2012.**

# Fragen?

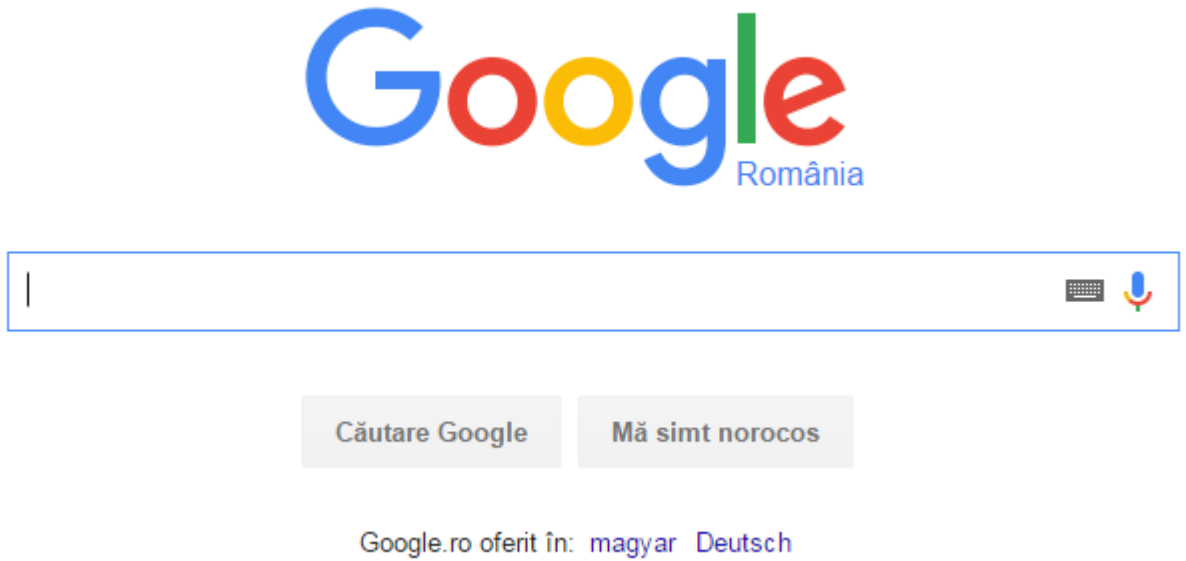
# 1. Einführung

Datenbanken Grundlagen

# Wo finden wir Datenbanken?







# Was sind Datenbanken/ Datenbankensysteme(DBS)?

- “A collection of related data items”  mit folgenden Eigenschaften:
  - Eine Datebank repräsentiert  einen bestimmten **Ausschnitt der realen Welt** durch einen **Datenmodell**
  - Eine Datenbank ist **logisch konsistent** und hat eine bestimmte Bedeutung
  - Eine Datenbank ist entworfen, aufgebaut und mit Daten gefüllt
  - Die Daten werden gespeichert für Aufzeichnungen (record-keeping) und Analyse

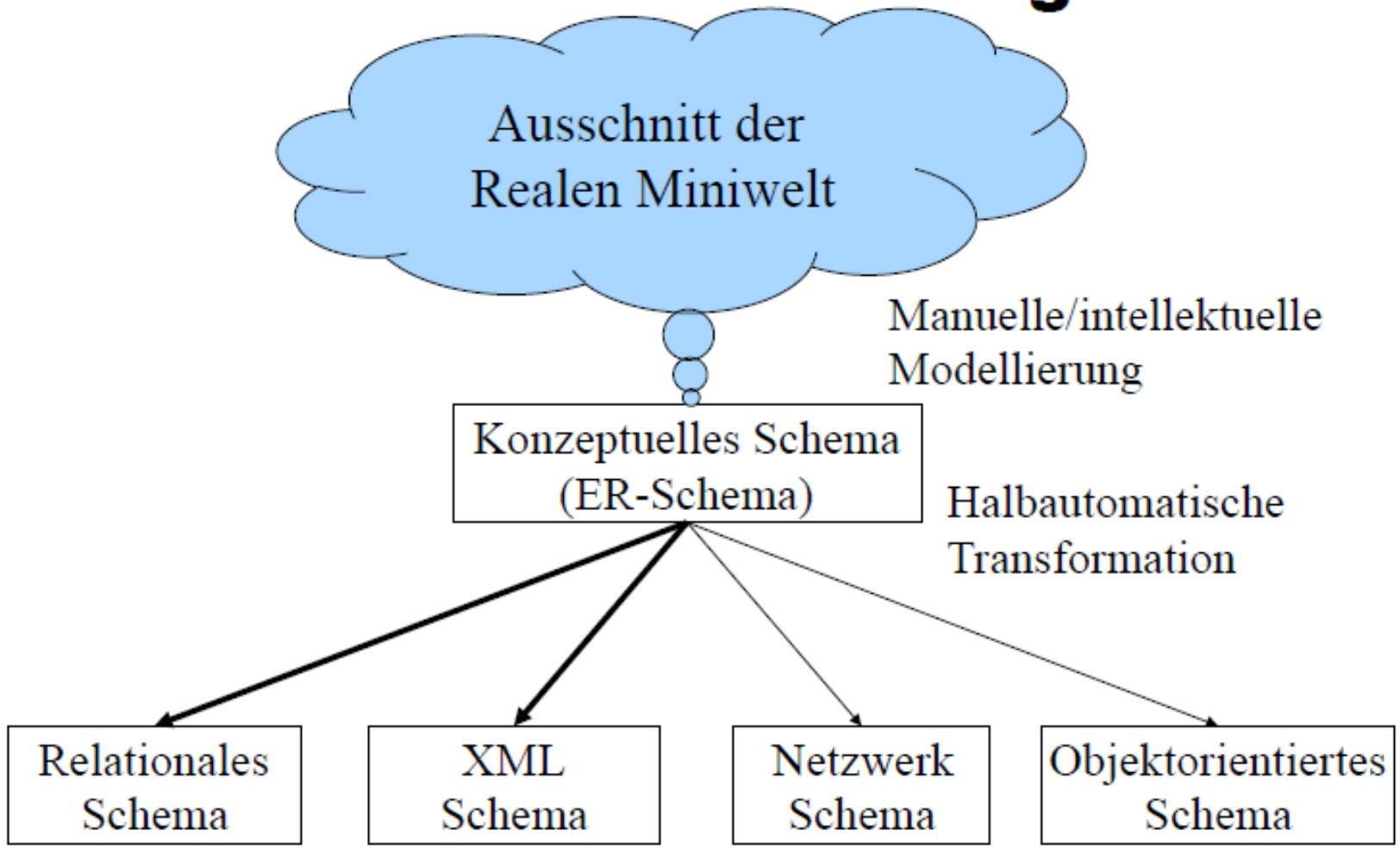
# Ziel und Zweck der Datenbanken

- Datenbanken werden benutzt für effiziente Speicherung, Wiederfindung und Analyse von Daten (store and manage data)
- Einsatzgebiete für Datenbanksysteme:
  - Kontoführungsdaten bei Banken
  - Verwaltung der Kundendaten bei Versicherung
  - E-learning Platforms
  - E-commerce Websites (Amazon, Emag, etc.)
  - Facebook
- Beispiele von non-computerized Datenbanken:
  - Telefonbuch
  - Wörterbuch

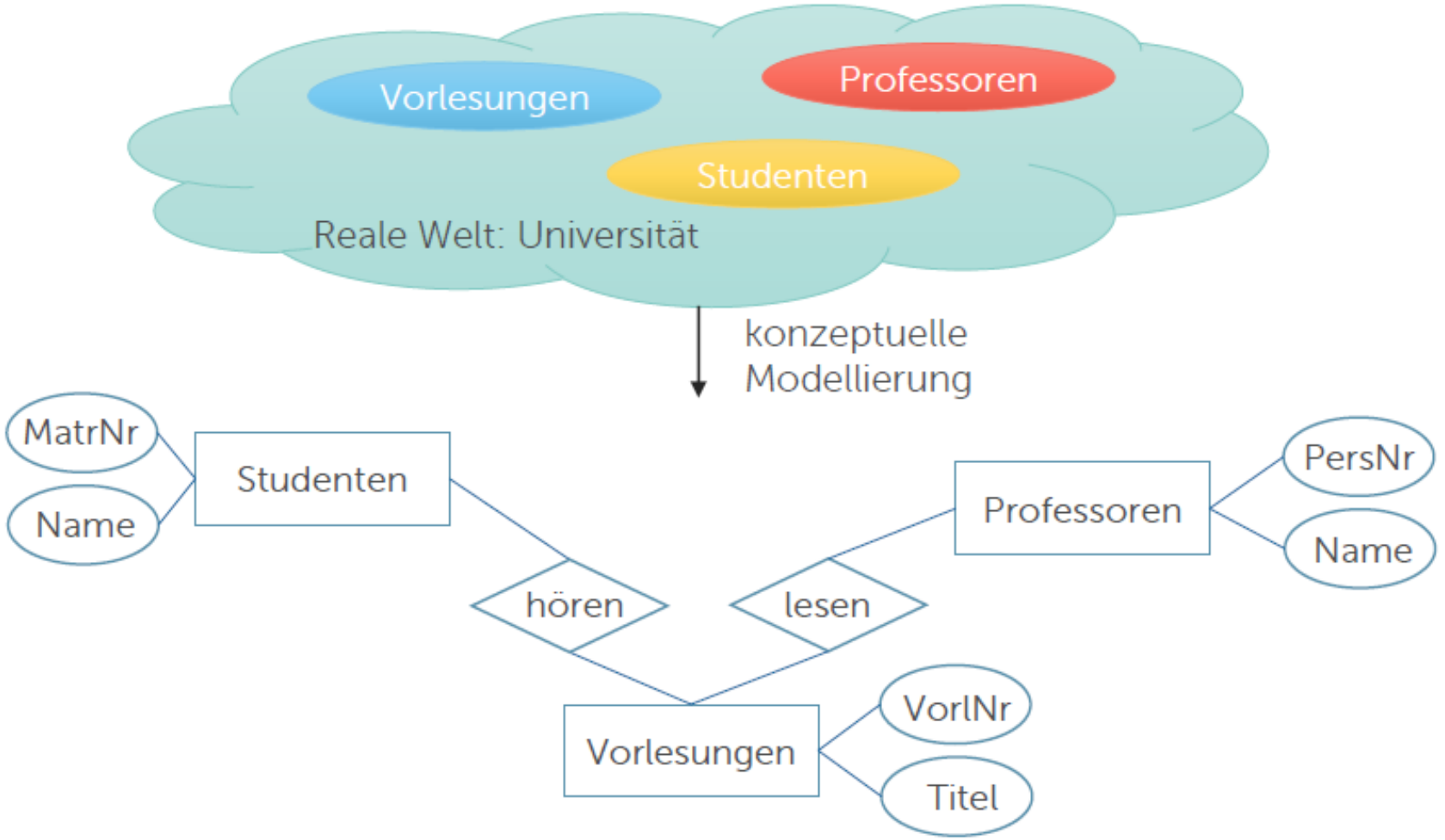
# Datenmodell

- Datenmodell
  - legt fest, welche Konstrukte zum Beschreibung der Daten existieren
- Schema
  - Eine konkrete Beschreibung einer bestimmten Datensammlung, unter Verwendung eines Datenmodells

# Datenmodellierung



# Modellierungsbeispiel



# Modellierungsbeispiel


| Studenten | |
|---|---|
| MatrNr | Name |
| 293948 | Schlegel |
| 292305 | Strufe |
| ... | ... |

| hören | |
|---|---|
| MatrNr | VorlNr |
| 292305 | 24 |
| 224833 | 24 |
| ... | ... |

| Vorlesung | |
|---|---|
| VorlNr | Titel |
| 24 | DB Grundlagen |
| 41 | Betriebssysteme |
| ... | ... |

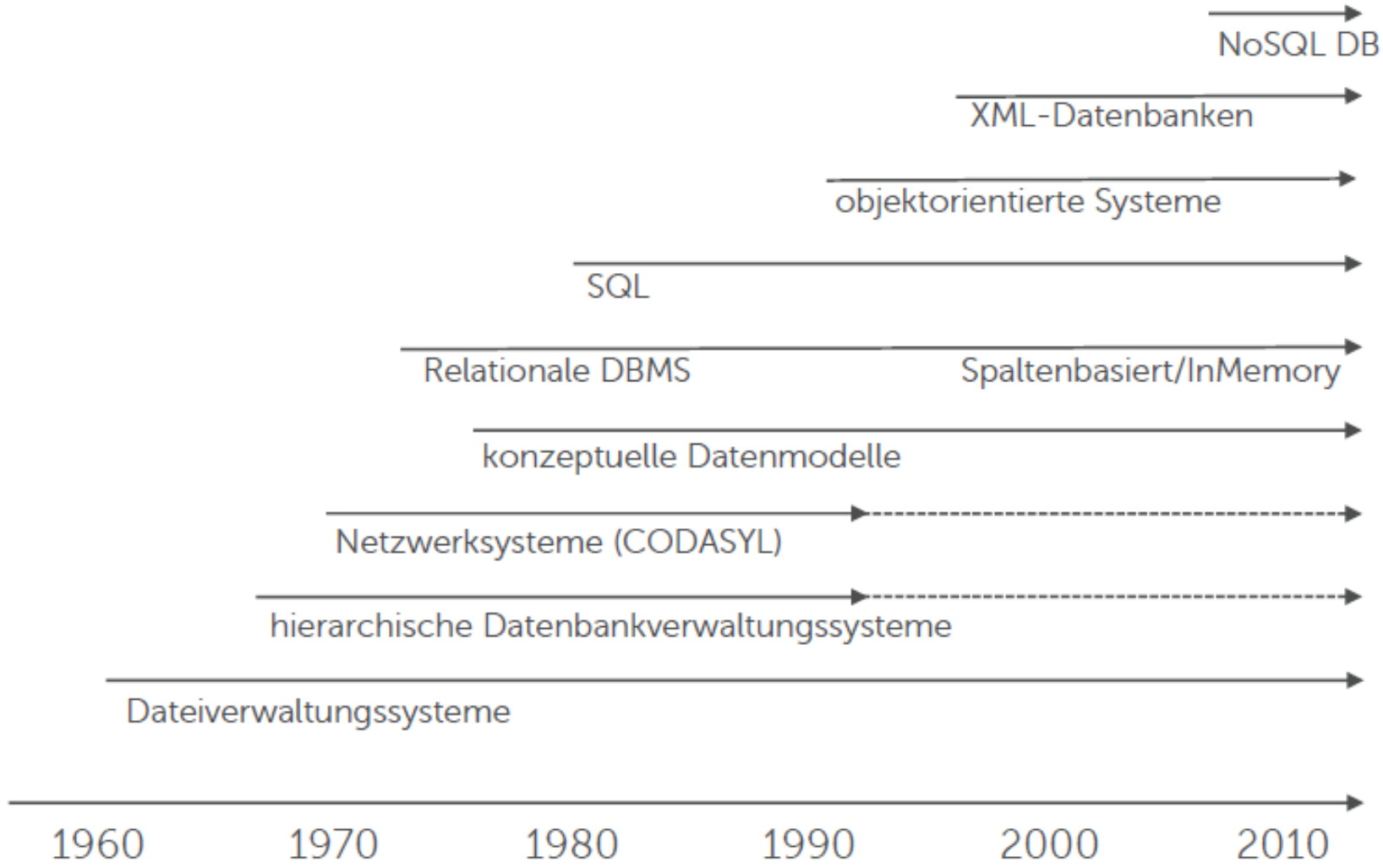
# Konzeptuelle Modelle

- Entity-Relationship-Modell (ER-Modell)
- Unified Modeling Language (UML)

# Logische Modelle

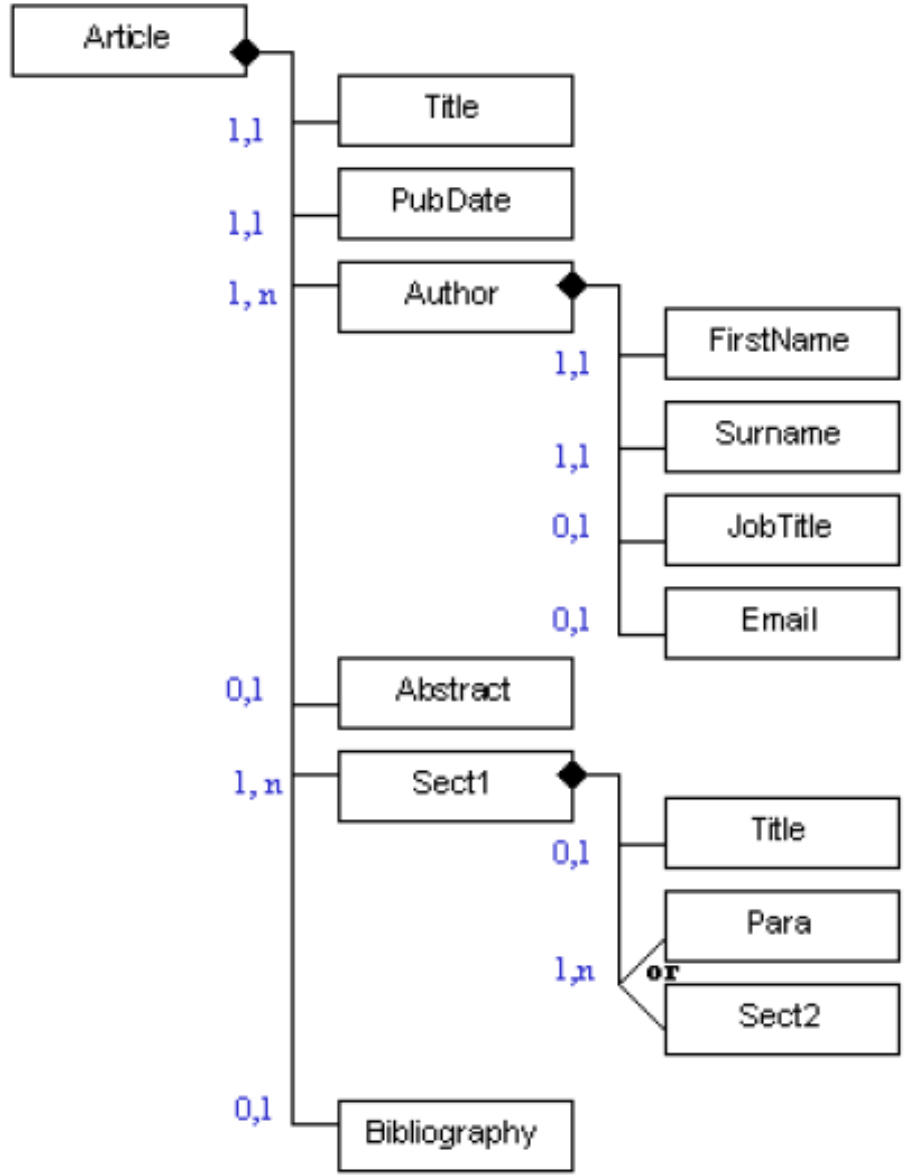
- Hierarchisches Datenmodell
- Netzwerkmodell
- Relationales Datenmodell
- Deduktives Datenmodell
- Objektorientiertes Datenmodell
- XML Schema

# Historische Entwicklung von DBMS



# Hierarchisches Datenmodell

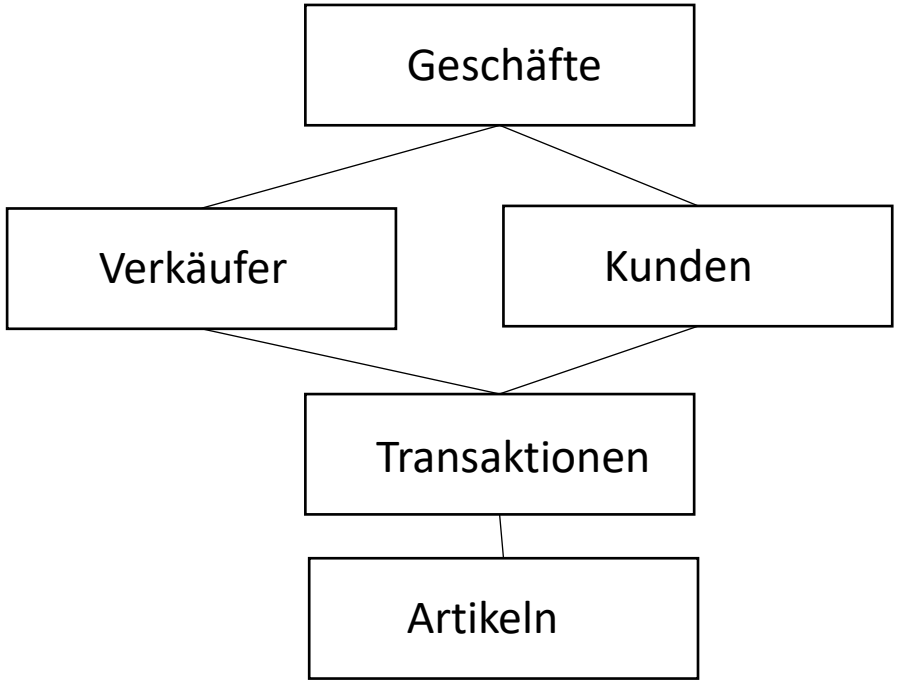
- Wurde in den 60er definiert
- Stellt die Daten in einer hierarchischen Baumstruktur dar




Entität *Article* – Hierarchisches Datenmodell

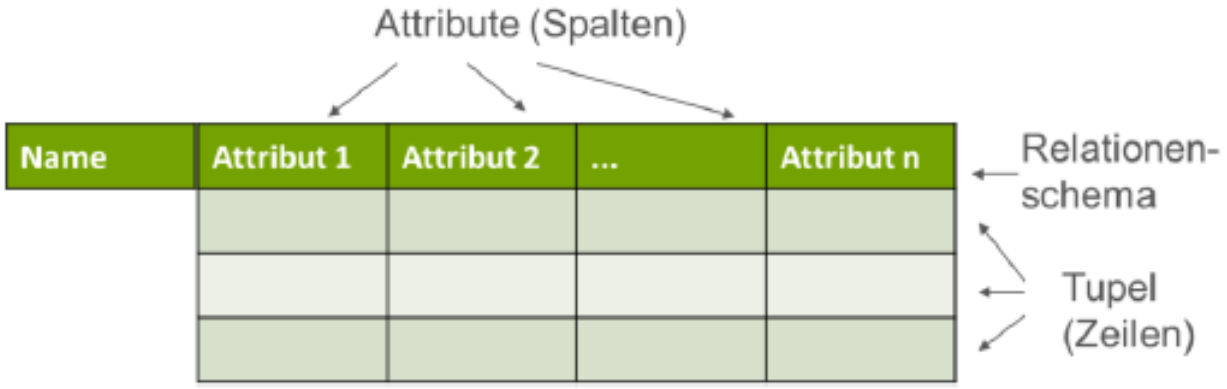
# Netzwerkmodell

- Eine Erweiterung von dem Hierarchisches Datenmodell
- Stellt die Daten in Form eines Graphs dar



# Relationales Datenmodell

- Wurde Anfang 70er von Ted Codd von IBM erfunden (1981 Turing Award)
- Am meisten benutztes Datenmodell (wird in den nächsten Vorlesungen ausführlich beschrieben)
- Relation als eigene Datenstruktur




Kunde

| KNr | Name | Vorname | Strasse | PLZ |
|---|---|---|---|---|
| 1 | Pop | Marcel | Abator | 534123 |
| 2 | Martin | Alex | Teiului | 324123 |

Relationales Datenmodell

# SQL

- Ende 70er wurde die Brauchbarkeit des relationalen Modells bewiesen
- SQL (Structured Query Language) entwickelt

# Objektorientiertes Datenmodell

- Konzepte: Klasse, Attribute, Methoden
- Relationen zwischen den Klassen: Assoziation, Aggregation, Vererbung
- Wird als Modell für Programmiersprachen benutzt
- In Datenbanken, aus Effizienz  Gründe, nicht so viel benutzt

# Schema vs. Data

- Datenbank Schema - Intension
  - beschreibt die Struktur der Datenbank (MetaDaten)
  - Zeitunabhängig (wird selten geändert)
- Ausprägung/Datenbankinstanz - Extension
  - Der Datenbankzustand zu einem bestimmten Zeitpunkt (snapshot), gegeben durch die aktuell existierenden Inhalte und Beziehungen und deren Attribute, wird **Datenbankinstanz** genannt
  - Die eigentlichen Daten einer Datenbank verändern sich im Laufe der Zeit häufig.
  - DBMS versichert, dass die Datenbank immer in einem validen Zustand ist

# Schema vs. Data

- Traditionales Data Management und Analyse
  - We never deduce from the extensions to the intension
  - But, by applying new intensional knowledge (via SQL) we are able to define intensions not covered by the original model (ex. average )

- Given Big Data (billions of extensions) it’s getting possible to deduce the intension, at least in a probabilistic sense

# Datenbankmanagementsystem (DBMS)/ Datenbankverwaltungssystem (DBVS)

- Eine Datenbank wird von einem laufenden DBMS verwaltet und für Anwendungssysteme und Benutzer unsichtbar auf nichtflüchtigen Speichermedien (damit die Daten nicht verloren gehen) abgelegt.
- DBMS ist eine Software, die für das Datenbanksystem installiert und konfiguriert wird
- Das DBMS legt das Datenbankmodell fest
- Bietet Tools für die bequeme, mühelose Verwaltung von Daten (ohne low-level Details)

# Beispiele von DBMS

- Record-based (Tuple-basierte) Datenmodelle:
  - Relationales Datenmodell (MySQL, MS SQL Server, Oracle, DB2, Informix, MS Access,  FoxBase, Paradox)
  - Hierarchisches Datenmodell (IBM’s DBMS)
  - Netzwerkmodell (wird in IDMS benutzt)

- Objekt-basierte Datenmodelle
  - Objektorientiertes Datenmodell (Objectstore, Versant)
  - Objektrelationales Datenmodell (Illustra, O2, UniSQL)

# Schwerpunkt der Vorlesung

- Relationale Datenbanken und DBMS:
  - Etablierter Stand der Technik und bestens erforscht
  - Flexibel und universell einsetzbar
  - In allen Größen und zu allen Preisen verfügbar
  - Von vielen Tools unterstützt

# Gründe für DBS-Einsatz

- Strukturierte Daten
- Effizienz und Skalierbarkeit (große Datenmengen)
- Integrität, Fehlerbehandlung und Fehlertoleranz
- Persistenz der Daten (nicht unkontrolliert verändern)
- Mehrbenutzersynchronisation
- Datenintegrität
- Deklarative Anfragesprachen: Benutzer sagt DBS was für Daten geholt werden sollen und nicht wie
- Datenunabhängigkeit: abstrakte Schichtenarchitektur

# Das Relationale Datenmodell

- Verwendet einfache Datenstrukturen: **Tabellen**
  - Einfach zu verstehen
  - Mengenorientiert
  - Nützliche Datenstruktur (passend für viele Situationen)
  - Hat eine nicht zu komplizierte Abfragesprache
- Grundlage des Konzeptes: **Relation**
  - Eine mathematische Beschreibung einer Tabelle
  - Führt zu formellen Abfragesprachen

# Terminologie

- **Domänen/Wertebereiche** – Integer, String, Datum, …
- **Relation** – besteht aus Attribute und Tupeln
- Relation R hat ein Relationsschema RS und eine Ausprägung
  - **Relationenschema** RS: legt die Struktur der gespeicherten Daten fest
    - Menge von Attributen $\{A_1, \ldots, A_k\}$
  - Attribute $A_j$ : Wertebereiche $D_j = dom(A_j)$
  - **Ausprägung**: der aktuelle Zustand der Datenbasis
    - Teilmenge des kartesischen Produkt der Wertebereiche, $val(R) \subseteq D_1 \times D_2 \times \ldots \times D_k$, $k \geq 1$
- **Datenbankschema** – Menge der Relationenschemata;
- **Datenbank** – Menge der aktuellen Relationen

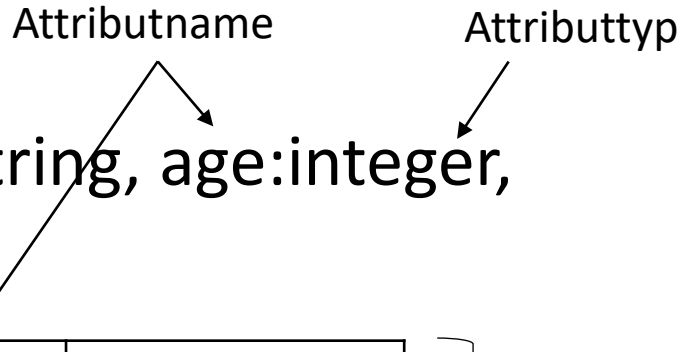
# Terminologie

- Ein Attribut beschreibt den Typ eines möglichen Attributwertes und bezeichnet ihn mit einem Attributnamen
- **Tupel**/Datensatz – Element einer Relation (eine konkrete Kombination von Attributwerten)
- Alle Tupel in der Relation sind verschieden
- **Kardinalität** – Anzahl der Tupel in einer Relation
- **Grad k einer Relation** (Degree) – Anzahl von Attributen in der Relationsschema;

$$R \subseteq D_1 \times D_2 \times \ldots \times D_k, \text{k} \geq 1$$

# Relation - Beispiel

- **Studenten**(sid:string, name:string, email:string, age:integer, gruppe:integer)




| **sid** | **Name** | **Email** | **Age** | **gruppe** |
|---|---|---|---|---|
| 2831 | Anne | anne@scs.ubbcluj.ro | 20 | 231 |
| 2532 | Silvia | silvia@scs.ubbcluj.ro | 19 | 233 |
| 2754 | Hannes | hannes@scs.ubbcluj.ro | 21 | 231 |

Relationsschema

Ausprägung-/Instanz der Relation

Tupel

Kardinalität = 3
Grad/Degree = 5

# Grundregeln

- Jedes Tupel (Zeile) ist eindeutig und beschreibt ein Objekt
- Die Ordnung der Zeilen und Spalten ist ohne Bedeutung
- Jeder Datenwert innerhalb einer Relation ist ein atomares Datenelement (integer, string, date)

# Integritätsregeln (Integrity Constraints)

- Regeln, die für jede Instanz der Datenbank erfüllt werden sollen
- Integritätsregeln werden beim Erstellen des Schemas festgelegt
- Fehlerhafte Datensätze werden nicht angenommen
- Beispiel von Integritätsregeln:

  **Studenten**(sid:string, name:string, email:string, age:integer, gruppe:integer)
  - Domäne-Constraints: gruppe:integer
  - Wertebereich-Constraints (Range constraints): $18 \leq age \leq 70$

# Primärschlüssel

- Notation: R – Relation, X⊆R (X ist eine Menge von Attributen aus der Relation R)

$$\pi_X(t) = \text{Tupel t eingeschränkt auf die Attribute X}$$

- Definition. Eine Menge von Attributen X⊆RS wird als **Schlüsselkanditat** bezeichnet, wenn folgende Bedingunge erfüllt sind:
  - *Eindeutigkeit*: für alle Relationen R des Schemas RS gilt

$$\forall t_i, t_j \in R, \pi_X(t_i) = \pi_X(t_j) \Rightarrow i = j$$

  - *Definiertheit* $\forall\, t_i \in R, \pi_X(t_i) \neq NULL$
  - *Minimalität* $\nexists Z \subset X, Z \neq X$, so dass die vorigen Bedingungen erfüllt sind
- Intuitiv: Schlüssel (Schlüsselkandidate) sind minimale Mengen von Attributen, deren Werte ein Tupel eindeutig idenifizieren

# Primärschlüssel

- Intuitiv:
  - *Eindeutigkeit+Definiertheit = jedes Tupel eindeutig identifizieren*
  - *Minimal = bei Weglassen eines einzelnen Attributs geht die Eindeutigkeit verloren*
- **Primärschlüssel** = minimale Menge von identifizierenden Attributen
- Wenn eine Relation mehrere Schlüsselkandidaten besitzt, wird einer davon als *Primärschlüssel* ausgewählt
- Die anderen: *alternative Schlüssel*

# Fremdschlüssel

- Notation: Relation $R_1$, Relation $R_2$ und $X \subseteq R_2$ Primärschlüssel
- Definition. $Y \subseteq R_1$ als **Fremdschlüssel** für $R_1$ bezüglich der Relation $R_2$ bezeichnet, wenn folgende Bedingungen erfüllt sind:
  - *Definiertheit* $\forall\, t_i \in R_1 : (\pi_Y(ti) = NULL \vee \exists t_j \in R_2 : \pi_Y(ti) = \pi_X(tj))$
  - *Minimalität* $\nexists Z \subset Y, Z \neq Y$, so dass die vorige Bedingung erfüllt ist
- Intuitiv: ein Fremdschlüssel verweist auf einen Primärschlüssel einer anderen Relation

# Fremdschlüssel - Beispiel

- **Studenten**(sid:string, name:string, email:string, age:integer, gruppe:integer)
- **Vorlesung**(vid:string, vname:string, ects:integer)
- **Klausur**(sid:string,vid:string,note:integer)
- Klausur:
  - sid – Fremdschlüssel, verweist auf Studenten
  - vid – Fremdschlüssel, verweist auf Vorlesung

# Referenz-Integritätsregel

- Eine relationale Datenbank enthält keinen Fremdschlüssel (ungleich NULL), der auf einen nichtexistenten Primärschlüssel verweist.
- *Bemerkung*. Ein neuer Datensatz mit einem Fremdschlüssel kann nur dann in einer Tabelle eingefügt werden, wenn in der referenzierten Tabelle ein Datensatz mit entsprechendem Wert im Primärschlüssel existiert.
- *Problem*: ein Tupel mit Primärschlüssel auf den Fremdschlüssel verweisen kann nicht einfach gelöscht werden.
- Mögliche *Lösungen*:
  - Löschen/Ändern nicht durchführen
  - Löschen /Ändern rekursiv aller darauf verweisender Tupel
  - Nullsetzen aller darauf verweisender Fremdschlüssel