

# Datenbanken

## Grundlagen

- Vorlesung und Seminar: Victor Şolea ([victor.solea@ubbcluj.ro](mailto:victor.solea@ubbcluj.ro))
- Labor: Victor Şolea ([victor.solea@ubbcluj.ro](mailto:victor.solea@ubbcluj.ro)) Gr. 721  
Cristian Matei ([cristian.matei@ubbcluj.ro](mailto:cristian.matei@ubbcluj.ro)) Gr. 722 und 723
- Fragen und Feedback sind immer erwünscht - Mail oder Teams
- Teams Code n82uqla
- Bitte folgt Nachrichten auf Teams für Updates
- Bitte regelmäßig Tabelle mit Anwesenheiten und Noten prüfen

# Struktur

- Vorlesung: jede Woche
  - Labor: jede Woche
  - Seminar: jede 2te Woche
- 
- Bei Labor und Seminar bitte mit der eigenen Gruppe kommen, außer ausnahmsweise bei Sonderfällen

# Anwesenheiten

- Um die Prüfung schreiben zu dürfen:
  - Wenigstens **12** Anwesenheiten bei dem Labor
  - Wenigstens **5** Anwesenheiten bei dem Seminar
- Bem. Das gilt auch für Studenten aus älteren Jahrgängen
- Bei entschuldigten Abwesenheiten (Arzttermin, Krankheit, dringender Familiennotfall, usw): Beweis max. 1 Woche später mitbringen
- Wer wegen Notfall mit einer anderen Gruppe kommt: **IM VORAUS MELDEN.**
- Wenn möglich, Alternativtermin für Labor bei derselben Lehrkraft finden

# Noten & Klausur

- Labor-Hausaufgaben – 25%
  - Bei den praktischen Aufgaben muss die **Mindestnote 5** erzielt werden, um an der praktischen Prüfung teilnehmen zu dürfen
- Praktischer Test – 25%
  - Bei dem praktischen Test muss die **Mindestnote 5** erzielt werden, um die schriftliche Prüfung schreiben zu dürfen
  - Der praktische Test findet in der letzten oder vorletzten Laborstunde statt
  - Kein Zweittermin
- Schriftliche Prüfung während der Prüfungszeit – 50%
  - Für das Bestehen der Prüfung: **Mindestnote 5** bei der schriftlichen Prüfung
  - Bei der Nachprüfung kann **nur** die schriftliche Prüfung nachgeholt werden

# Noten & Klausur

- Laboraufgaben:
  - Es ist möglich, die Hausaufgaben mit Verspätung abzugeben
  - Für die erste Verspätungswoche werden 2p abgezogen, danach je 1p pro Woche, d.h.
    - In der **ersten Verspätungswoche** kann man **höchstens Note 8** bekommen
    - In der **zweiten Verspätungswoche** kann man **höchstens Note 7** bekommen
    - In der **dritten Verspätungswoche** kann man **höchstens Note 6** bekommen
    - usw
  - Man kann nicht mehr als 2 Aufgaben in einer Laborstunde abgeben
- Um eine Aufgabe abzugeben muss man **die Lösung erklären!**

# Klausurdatum

JANUAR 2025						
Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag	Sonntag
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

2025 FEBRUAR						
Mon	Die	Mit	Don	Fre	Sam	Son
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	1	2

# Folien, Literatur

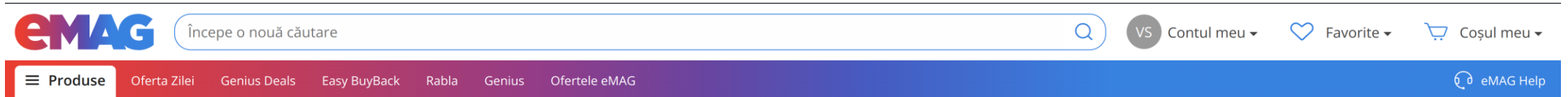
- Folien und andere Informationen zur Vorlesung, Seminar und Übungen werden auf Teams zur Verfügung gestellt
- Literatur:
  - **A. Kemper , A. Eickler. Datenbanksysteme – Eine Einführung. Oldenbourg Verlag, 2015. 10. Auflage.**
  - **A. Kemper, M. Wimmer. Übungsbuch Datenbanksysteme. Oldenbourg Verlag, 3. Auflage, 2012.**



Fragen?

Einführung

# Wo finden wir Datenbanken?



# Was sind Datenbanken / Datenbankensysteme (DBS)?

- “A collection of related data items” mit folgenden Eigenschaften:
  - Eine Datenbank 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 für Aufzeichnungen (record-keeping) und Analyse gespeichert

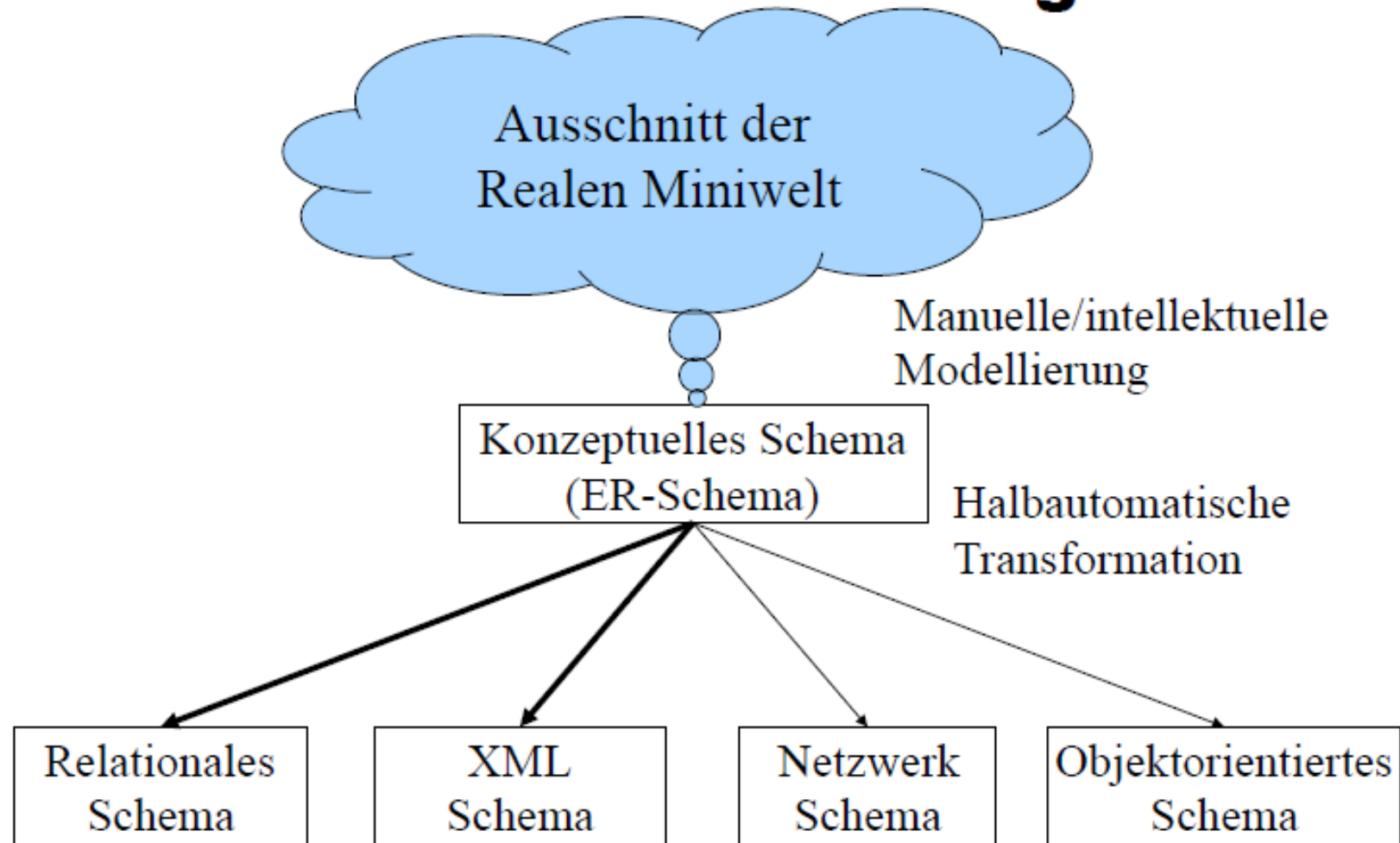
# Ziel und Zweck der Datenbanken

- Datenbanken werden für effizientes Speichern, Wiederfinden und Analysieren von Daten (store and manage data) benutzt
- Einsatzgebiete für Datenbanksysteme:
  - Kontoführungsdaten bei Banken
  - Verwaltung der Kundendaten bei Versicherung
  - E-learning Plattformen
  - E-commerce Websites (Amazon, Emag, etc.)
  - Soziale Netzwerke (Instagram, Reddit, etc.)
- Beispiele von non-computerized Datenbanken:
  - Telefonbuch
  - Wörterbuch

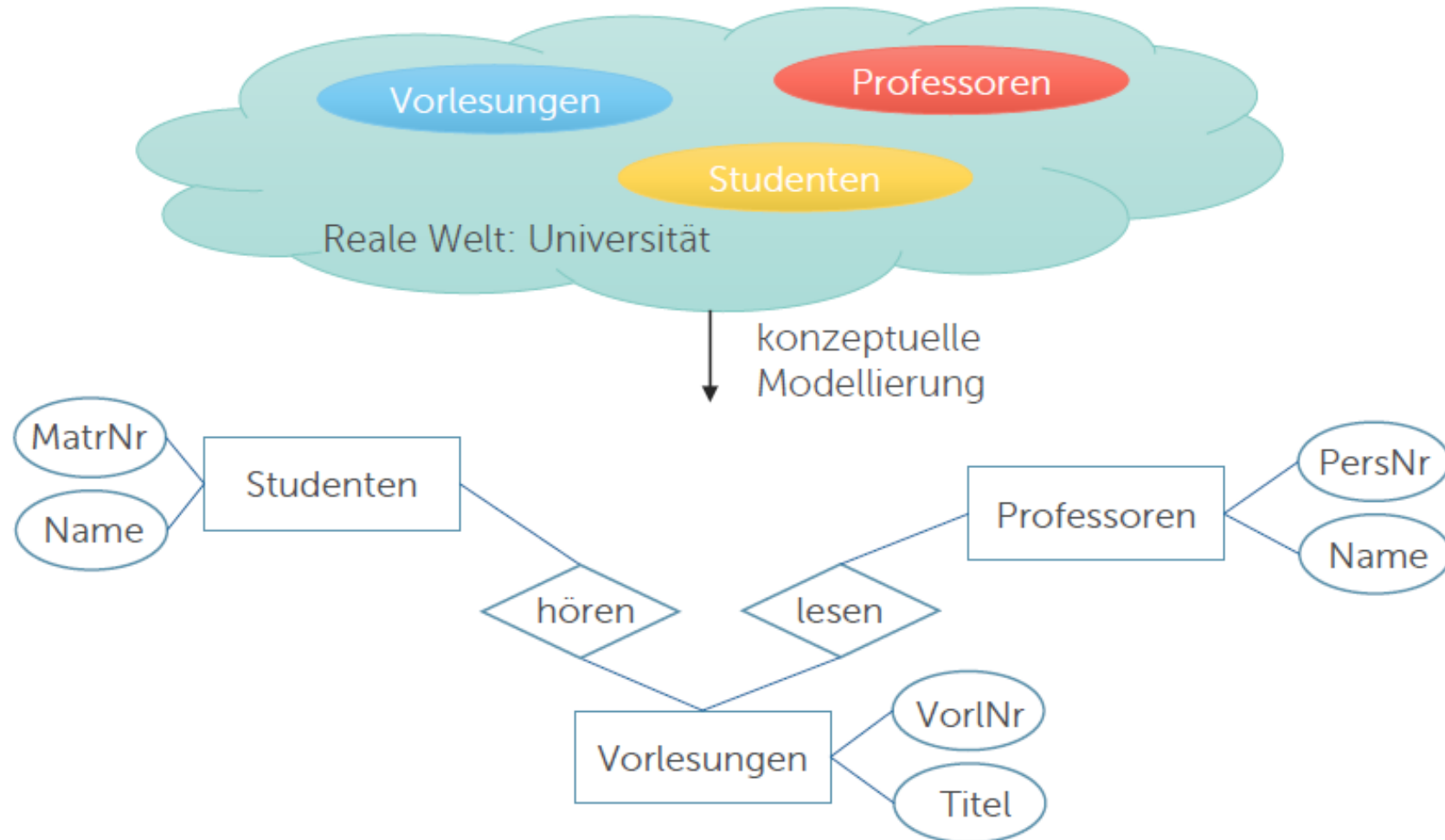
# Datenmodell

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

# Datenmodellierung

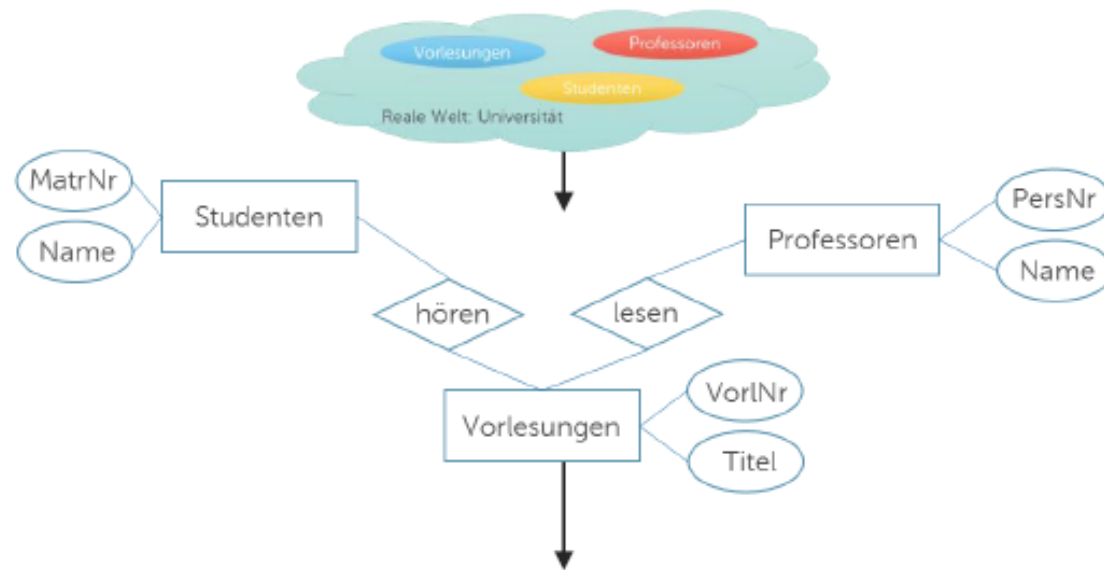


# Modellierungsbeispiel





# Modellierungsbeispiel



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

hören	
MatrNr	VorlNr
292305	24
224833	24
...	...

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

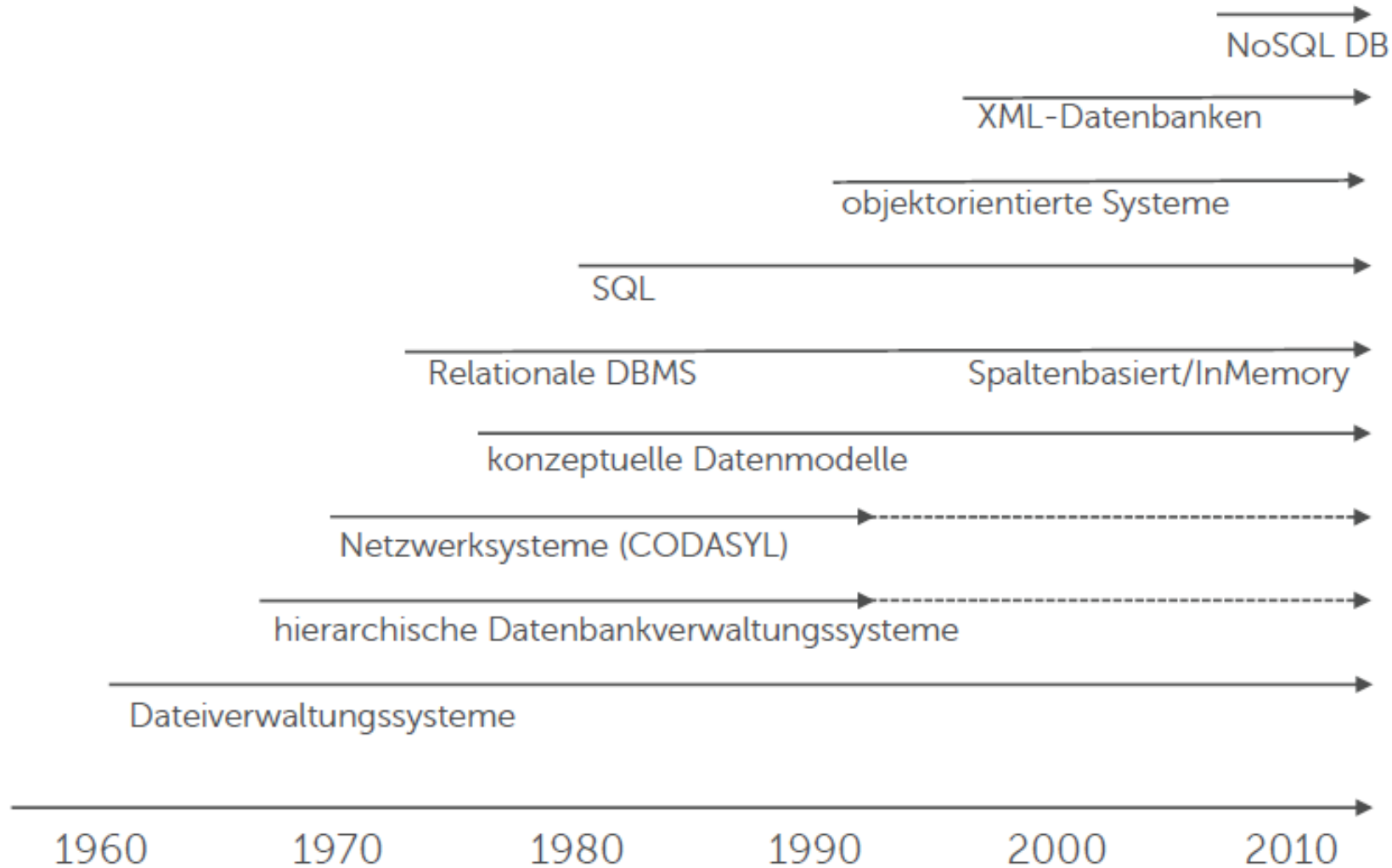
# Konzeptuelle Modelle

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

# Logische Modelle

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

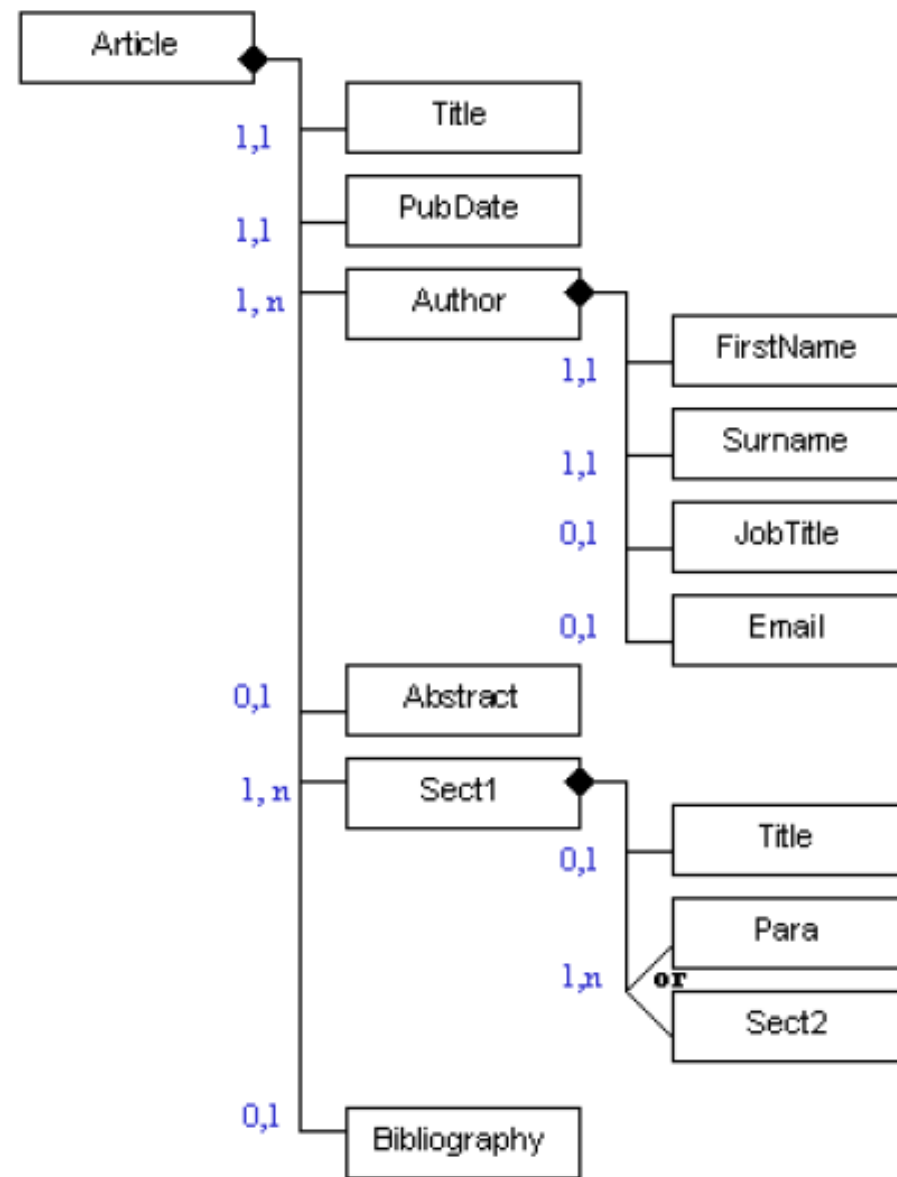
# Historische Entwicklung von DBMS



# Hierarchisches Datenmodell

- Wurde in den 1960er definiert
- Stellt die Daten in einer hierarchischen Baumstruktur dar
- Vorher:

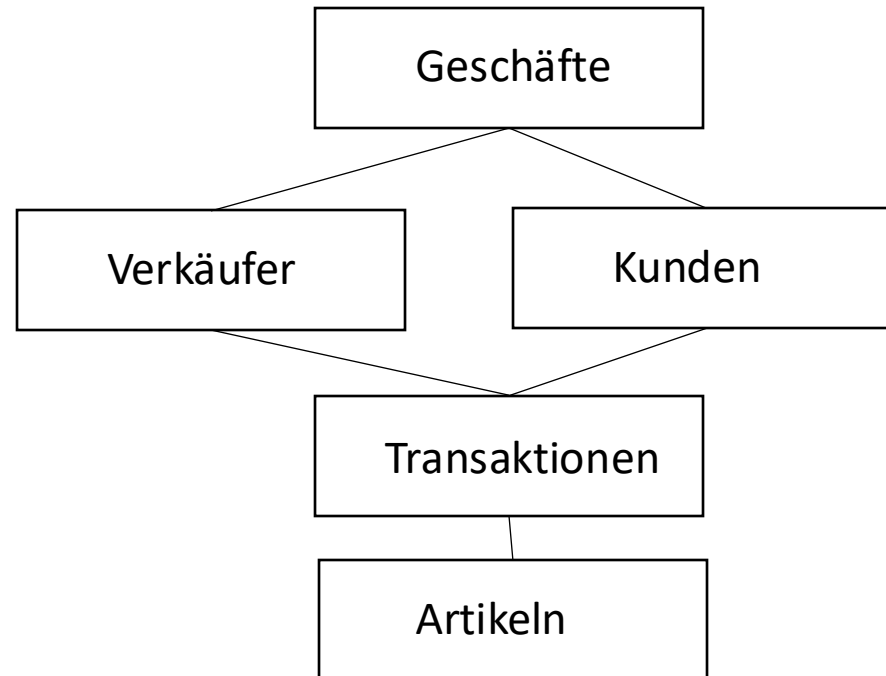




Entität *Article* – Hierarchisches Datenmodell

# Netzwerkmodell

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

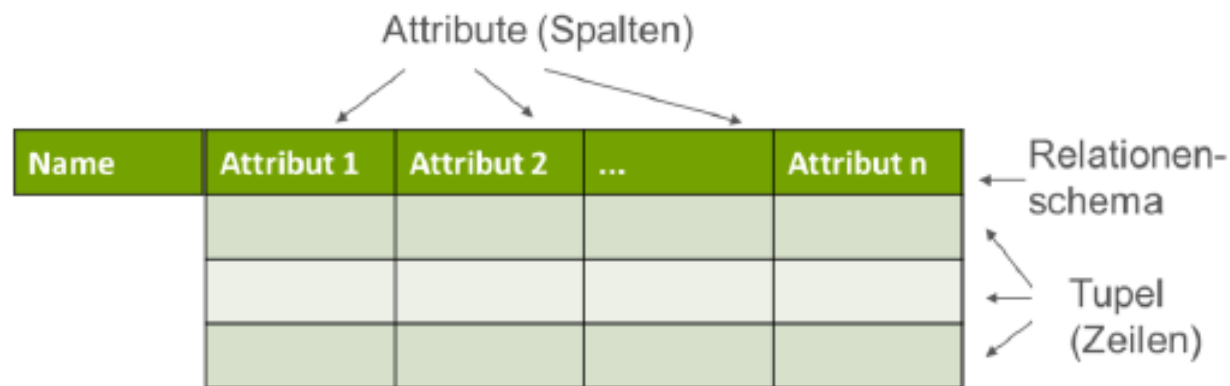


# Objektorientiertes Datenmodell

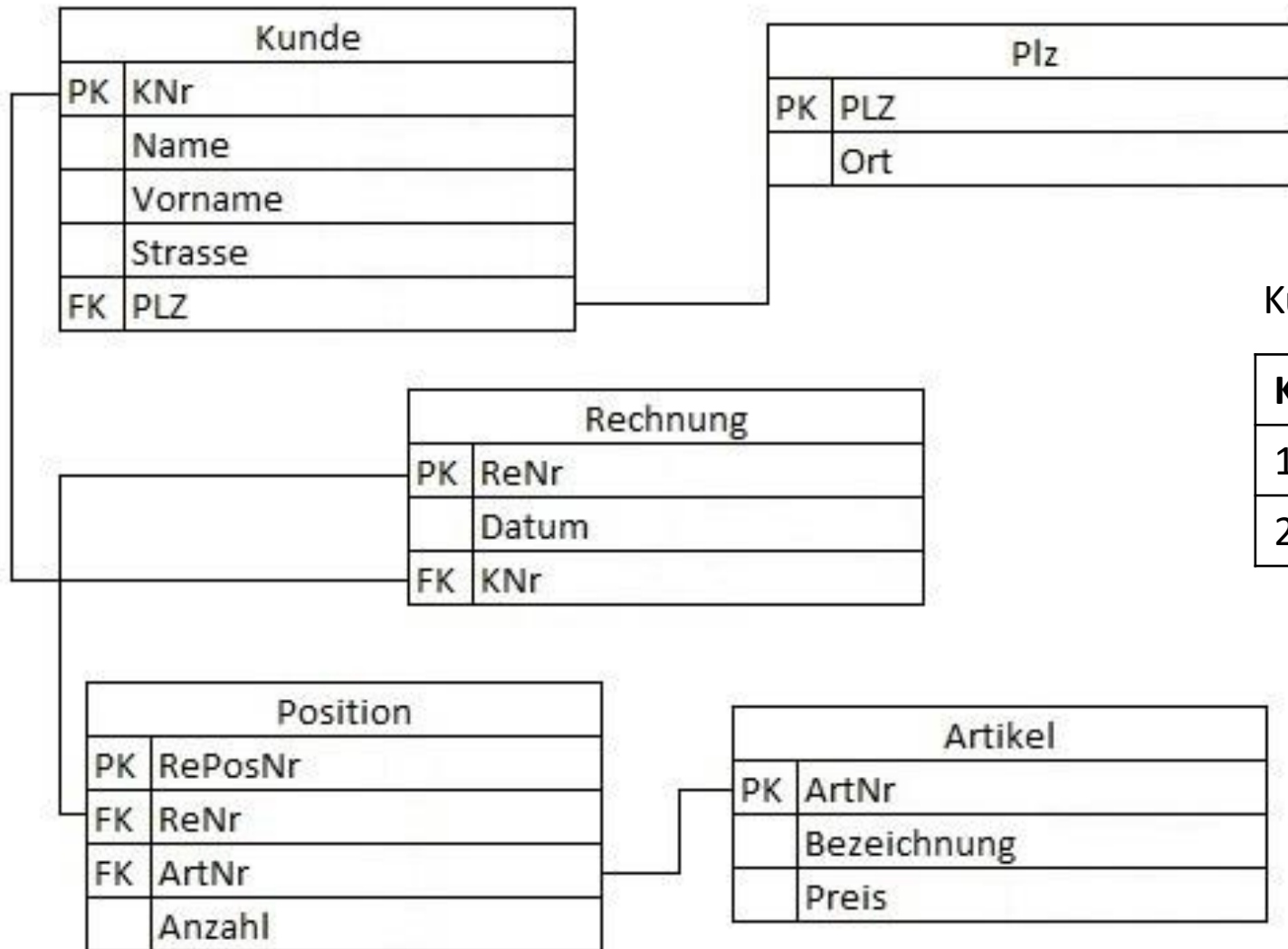
- Konzepte: Klasse, Attribute, Methoden
- Relationen zwischen den Klassen: Assoziation, Aggregation, Vererbung
- Wird als Modell für Programmiersprachen benutzt
- In Datenbanken, aus Effizienzgründen nicht so viel benutzt

# Relationales Datenmodell

- Wurde 1969 von Edgar F. 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) bei IBM entwickelt

# 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

# Database management systems (DBMS)/ Datenbankverwaltungssysteme (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 (Tupel-basierte) Datenmodelle:
  - Relationales Datenmodell (MySQL, **MS SQL Server**, Oracle, PostgreSQL, DB2, SQLite, usw)
  - 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
  - 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 Attributen und Tupeln
- Relation R hat ein Relationenschema RS und eine Ausprägung
  - **Relationenschema** RS: legt die Struktur der gespeicherten Daten fest
    - Menge von Attributen  $\{A_1, \dots, A_k\}$
    - Attribut  $A_j$  : Wertebereiche  $D_j = \text{dom}(A_j)$
    - **Ausprägung**: der aktuelle Zustand der Datenbasis
      - Teilmenge des kartesischen Produkt der Wertebereiche,  $\text{val}(R) \subseteq D_1 \times D_2 \times \dots \times D_k, k \geq 1$
- **Datenbankschema** – Menge der Relationenschemata;
- **Datenbank** – Menge der aktuellen Relationen

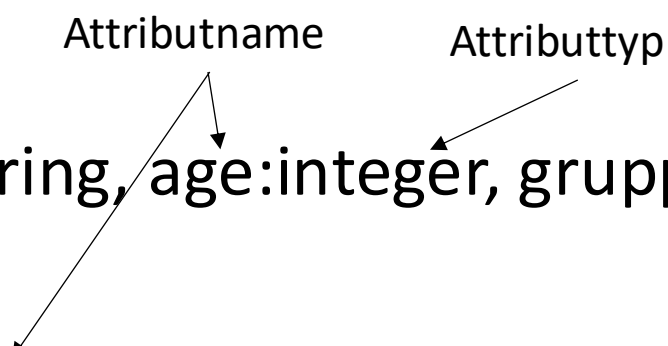
# Terminologie

- 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;

$$R \subseteq D_1 \times D_2 \times \dots \times D_k, k \geq 1$$

# 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):  $18 \leq \text{age} \leq 70$

# Primärschlüssel

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

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

- Definition. Eine Menge von Attributen  $X \subseteq RS$  wird als **Schlüsselkandidat** bezeichnet, wenn folgende Bedingungen 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 (Schlüsselkandidaten) sind minimale Mengen von Attributen, deren Werte ein Tupel eindeutig identifizieren

# Primärschlüssel

- Intuitiv:
  - *Eindeutigkeit + Definiertheit = man kann 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, Kandidatschlüssel*



# 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*  $\forall t_i \in R_1: (\pi_Y(t_i) = NULL \vee \exists t_j \in R_2: \pi_Y(t_i) = \pi_X(t_j))$
  - *Minimalität*  $\nexists Z \subset Y, Z \neq Y$ , so dass die vorige Bedingung erfüllt ist
- Intuitiv: ein Fremdschlüssel verweist auf einen Primärschlüssel (oder Kandidatschlüssel) 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