



Datenbanken und Informationssysteme

Einführung

Prof. Dr. Stefan Decker

📍 RWTH Aachen

Das Problem: Wenn einfache Dateien scheitern

Die Anforderung



Szenario: Taylor Swift Konzert

 50.000 verfügbare Tickets

 100.000 User gleichzeitig online

Die naive Lösung

Ansatz: Alles in einer `tickets.csv` speichern.

- ✗ **Race Conditions:** 10 User kaufen dasselbe Ticket.
- ✗ **Inkonsistenz:** Server stürzt beim Schreiben ab.
- ✗ **Sicherheit:** Jeder sieht alle Käuferdaten.

 **Fazit:** Ein simples Dateisystem kann **Nebenläufigkeit und Datensicherheit** nicht garantieren.

Die Lösung: Das Datenbanksystem

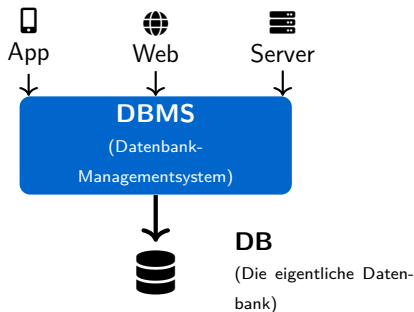
DBS = DBMS + DB

⚙️ Das Datenbank-Management-system (DBMS):

- ▶ Die intelligente Software-Schicht (Der „Türsteher“).
- ▶ Verhindert Race Conditions, sichert Integrität, prüft Rechte.

🗄️ Die Datenbank (DB):

- ▶ Die dauerhaft gespeicherte, strukturierte Datensammlung.



📌 Eiserne Regel: Anwendungsprogramme greifen **NIEMALS** direkt auf die Daten zu!

Ihr Paradigmenwechsel: Deklarativ statt Prozedural

Prozedural: Das „WIE“ (Bisher)

Java-Code:

```
for (Student s : studentenListe) {  
    if (s.getNote() < 2.0) {  
        ergebnis.add(s);  
    }  
}
```

- Sie beschreiben den genauen Lösungsweg
- Schritt-für-Schritt-Anweisungen (Schleifen, If/Else)

Deklarativ: Das „WAS“ (Neu)

SQL-Code:

```
SELECT * FROM Studierende  
WHERE Note < 2.0;
```

- Sie beschreiben nur das gewünschte Ergebnis
- Das System sucht selbst den besten Weg (z.B. via Index)

Persistenz (Dauerhaftigkeit)

Wir arbeiten nach dem Prinzip „Die Daten sind schon da“.
Der Zustand überlebt den Programmaufruf dauerhaft!

Abstraktion: Die Kunst des Weglassens

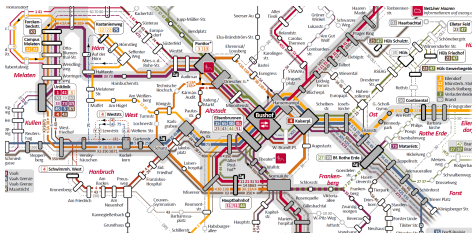
Die Realität



- ▶ Unendlich komplex und detailliert.
- ▶ Zu viel „Rauschen“ für die Navigation.

Abstraktion

Das Modell



- ▶ Fokus auf Struktur und Wesentliches
- ▶ Geografisch „falsch“, aber nützlich.

💡 **Definition: Abstraktion** ist das Ableiten des Wesentlichen für einen bestimmten Zweck (z.B. Wegfindung).

Zwei Ebenen: Bauplan vs. Bewohner (Schema vs. Zustand)

Der Bauplan (Schema / Intension)



- Beschreibt die möglichen Inhalte (Metadaten, Typen).
- Ändert sich extrem selten.
- Schemaevolution ist aufwendig.

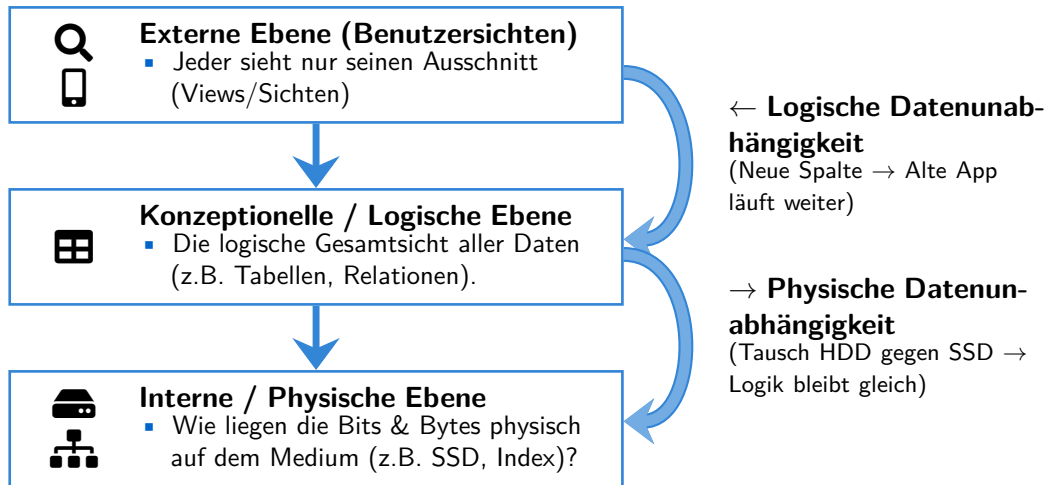
Die Bewohner (Zustand / Extension)



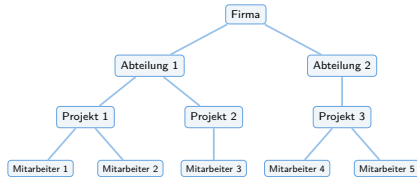
- Der tatsächliche, aktuelle Inhalt (viele Datensätze).
- Ändert sich permanent (viele Transaktionen/s).

 **Aufgabe des DBS: Schema (Bauplan)**
und **Zustand (Inhalte)** gemeinsam verwalten.

Architektur: Das 3-Ebenen Modell (ANSI/SPARC)



Ein kurzer Blick in die Geschichte: Von Hierarchien zu Relationen



Vor 1970:
Hierarchische
& Netzwerkmodelle

Die
Revolution

**1970: Edgar F. Codd &
Das Relationale Modell**

**Heute: Der
unangefochtene
Industriestandard**

- ▶ Starre Strukturen, komplexe Navigation („Wo ist mein Datensatz?“)
- ▶ Keine Datenunabhängigkeit (Änderungen brechen Programme!).

- ▶ Mathematisch fundiert (Mengenlehre)
- ▶ Trennung von logischer Sicht und physischer Speicherung

- ▶ Rückgrat der Weltwirtschaft (SQL).
- ▶ Verlässlich, ausgereift, mächtig für strukturierte Daten.

Big Data (Problem) & NoSQL (Lösung)

⚙️ Big Data (Problem)



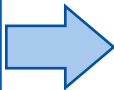
Volume: Petabytes an Sensordaten.



Velocity: Millionen Klicks pro Sekunde.



Variety: Unstrukturierte Daten (Texte, Bilder, Videos).



🗄️ NoSQL / Polyglot Persistence (Lösung)



Dokumenten-Stores: z.B. MongoDB für flexible Kataloge.

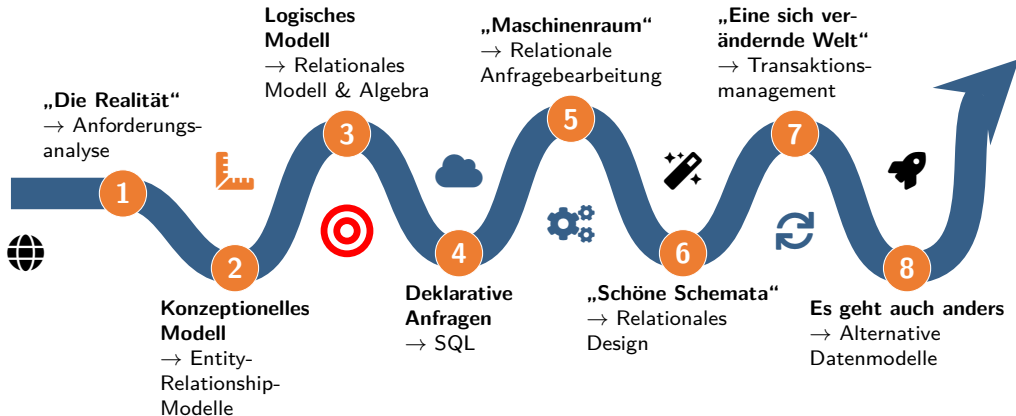


Graphdatenbanken: z.B. RDF für vernetztes Wissen.



Fazit: SQL bleibt für viele klassische Unternehmensprobleme ideal. Für stark vernetzte, sehr große, unstrukturierte oder verteilte Daten nutzen wir spezialisierte Modelle.

Ausblick: Unsere Reise in diesem Semester



-  **Unser roter Faden:** Von der chaotischen Realität bis in den strukturierten Maschinenraum der Datenbank.
-  **Der Fokus:** Das Relationale Modell steht als absoluter Industriestandard im Zentrum dieser Vorlesung.
-  **Das Ziel:** Am Ende des Semesters können Sie komplexe Informationssysteme sauber entwerfen und effizient abfragen.