Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Datenbanken

Vorlesungsskript für das 3. Semester

Studiengang Internationale Medieninformatik

1. Einführung in Datenbanken und Datenbankmanagementsysteme

Dozent: M. Sc. Burak Boyaci

Version: 07. Oktober 2025

Wintersemester 25/26

Lizenz

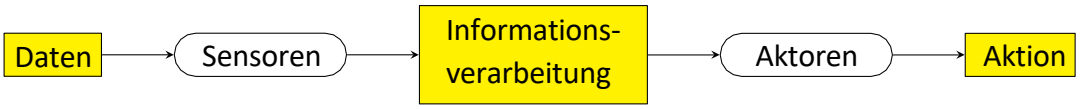
1

Datenbankmanagementsysteme (DBMS)

Kapitelübersicht

1.1 Die Hierarchie der Speicherkonzepte

Ein wichtiges Merkmal aller Lebewesen ist es, Daten aus der Umwelt als Information zu verarbeiten und daraus Aktionen abzuleiten.



Die Daten sind hierbei Reize der Umgebung, sei es in Form elektromagnetischer oder akustischer Wellen, chemischer Reaktionen oder mechanischen Drucks. In der Informatik wird eine Datenmenge in Bits gemessen, der Einheit des Informationsgehalts. Höhere Lebewesen filtern aus diesen Daten durch ihr zentrales Nervensystem Information. Beim Menschen geschieht dies zum allergrößten Teil unbewusst. Wie aus Tabelle 1.1 ersichtlich, werden von den etwa 11,2 Megabits, die unsere Sinnesorgane pro Sekunde empfangen können, nur etwa 77 Bits bewusst wahrgenommen, also etwa 0,01 ‰ = 10^{-5}. Zum Vergleich dazu hat ein Farbfernseher eine Datenrate von ungefähr 10 Mbit s, also etwa die Aufnahmekapazität unserer Augen, ein Radio hat eine Datenrate von ca. 10 kbit s, d.h. etwa 10 % der Kapazität unserer Ohren, und ein laut gelesener Text hat eine Datenrate von etwa 25 bit s.[1]
An einem Tag mit 16 Stunden Helligkeit nehmen also allein unsere Augen eine Datenmenge von etwa 72 GB auf, bewusst wahrgenommen werden davon lediglich 288 kB. Die Speicherkapazität eines menschlichen Gehirns wird auf etwa 60 TB geschätzt.[2]

Ganz ähnlich wie biologische Organismen verarbeiten auch unsere technischen Informationssysteme Daten zu Information und speichern sie. Abhängig von Menge und Komplexität

[1]vgl. Nørretranders (1997):S. 227.
[2]Nach Bartol Jr. et al. (2015) kann eine Synapse, also die Verbindung zwischen zwei Neuronen, Informationen

Sinnessystem	Datenrate [bit/s]	
	Kapazität	Bewusstsein
Augen	10 000 000	40
Ohren	100 000	30
Haut	1 000 000	5
Geschmack	1 000	1 (?)
Geruch	100 000	1 (?)
Gesamt	11 201 000	77

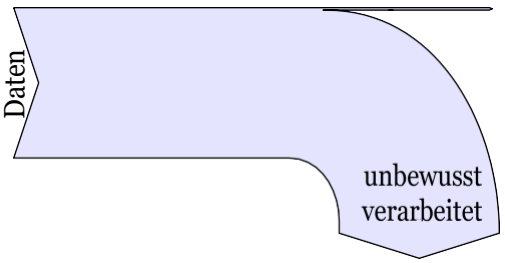


Tabelle 1.1: Informationsfluss der Sinnesorgane und des Bewusstseins. Schätzwerte sind mit "(?)" gekennzeichnet. Quelle: Zimmermann (1993)

verwendet man in der Informatik dazu verschiedene Speicherkonzepte, die eine Hierarchie hinsichtlich ihres Strukturierungsgrades bildet (Abbildung 1.1). Die einfachste Struktur sind die

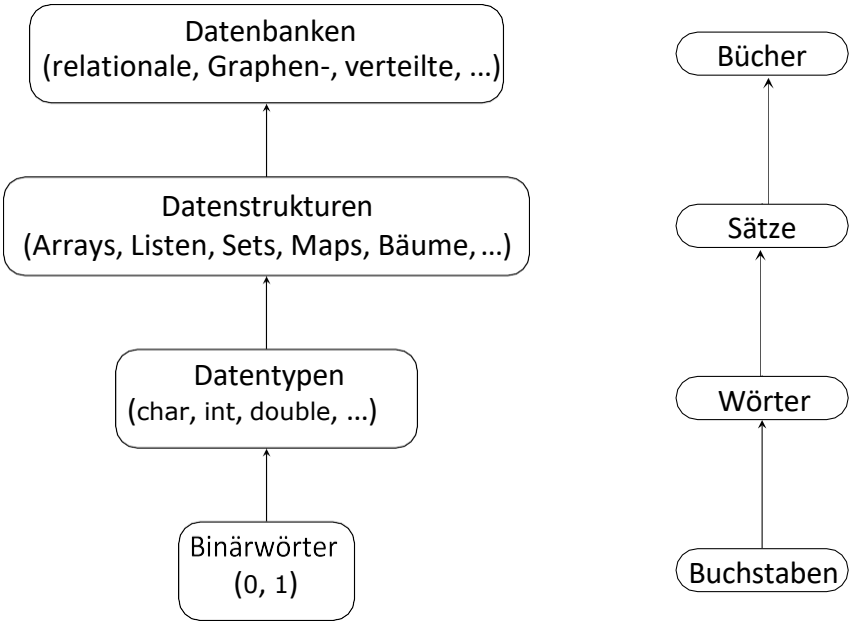


Abbildung 1.1: Die Hierarchie der Speicherkonzepte in der Informatik (links) und in der Literatur (rechts)

Binärwörter, die sich nur aus Nullen und Einsen zusammensetzen. Sie bilden die logische Grundlage aller Datenspeicher, d.h. alle Daten werden am Ende durch sie gespeichert. Das entspricht in etwa der Tatsache, dass geschriebene Texte als Basis mit Buchstaben geschrieben („gespeichert") sind. Für sich allein genommen haben Binärwörter zwar keine eigentliche Bedeutung, dafür können sie aber auch Daten aller Art speichern.

Die nächste Strukturierungsstufe sind die *Datentypen*, die Daten in Symbole, ganze Zahlen oder Kommazahlen unterscheiden und als Binärwörter fester Länge speichern. Datentypen sind die Basis für die Programmiersprachen und zum großen Teil standardisiert:
Symbole sind üblicherweise durch den Unicode UTF-8 der Längen 1 bis 4 Byte kodiert (früher ASCII mit 7 bits), ganze Zahlen als Integer der Größe 4 Byte und Kommazahlen als Gleitkommazahlen nach dem Format IEEE 754 mit 4 Byte (single oder float) oder 8 Byte (double). Analog einem geschriebenen Text bilden sie als Struktureinheiten Wörter, die aus den Buchstaben gebildet sind.

in 26 Zuständen digital speichern, d.h. sie hat eine Speicherkapazität von 4,7 = $\log_2 26$ bits. Bei etwa 100 Milliarden = 10^{11} Neuronen mit jeweils etwa 1000 Synapsen ergibt dies $470 \cdot 10^{12}$ bits oder 470/8 ≈ 60 TB.

Werte von Datentypen können zu Einheiten sogenannter *Datenstrukturen* zusammengefasst werden. Die häufigste Datenstruktur der Informatik ist das Array. Die APIs fast aller gängigen Programmiersprachen bieten darüber hinaus Listen und Zuordnungen („Maps“) an.
Zur strukturierten und vor allem persistenten – d.h. dauerhaften – Speicherung großer und komplexer Datenmengen eignen sich diese Datenstrukturen allerdings nicht. Oft sollen vorhandene Datenbestände außerdem noch für mehrere Nutzer gleichzeitig verfügbar sein, wobei die Lese- und Schreibrechte individuell festzulegen sind. Für diese Zwecke werden *Datenbanken* eingesetzt. Abhängig von der Struktur der Daten und den Zielsetzungen der Anwendung gibt es dazu verschiedene Datenbanktypen. Das Spektrum reicht zum Beispiel von relationalen Datenbanken auf zentralen Servern über verteilte Datenbanken, die auf mehreren Rechnern oder sogar auf Rechnerverbünden laufen, bis hin zu Graphendatenbanken, die sich für eine netzorientierte Datenorganisation eignen. Der für betriebswirtschaftliche Anwendungen seit den 1970er Jahren wichtigste Datenbanktyp jedoch sind die relationalen Datenbanken. Sie sind Hauptthema dieses Skriptes.

1.2 Die ANSI-SPARC-Architektur

Interessanterweise folgte die historische Entwicklung der Datenspeicherung im Wesentlichen genau der Speicherkonzepthierarchie in Abbildung 1.1, von unten nach oben. In den 1960er Jahren wurde Datenspeicherung allgemein auf Basis von Datenstrukturen wie Arrays, Listen und Maps realisiert. Durch die zunehmende Komplexität der anfallenden Daten und sich oft erweiternde Anforderungen an sie kam es zu immer ernsteren Problemen, zu deren Lösung schließlich 1975 die dreischichtige Referenzarchitektur des ANSI-SPARC eingeführt wurde. Dieses Modell bildete den abstrakten Entwurf der in der Folgezeit realisierten Datenbanken. Es ist in Abbildung 1.2 skizziert. Es ist ein Schichtenmodell, in dem jede Schicht ihre speziellen Aufgaben

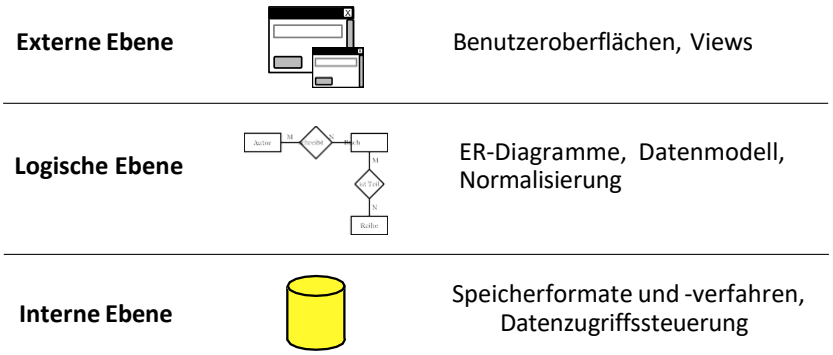


Abbildung 1.2: Die drei Schichten der ANSI-SPARC-Architektur

hat und mit den angrenzenden Schichten über festgelegte Protokollschnittstellen kommuniziert. Dadurch ist es problemlos möglich, in jeder Schicht separat Änderungen vorzunehmen, solange sie die Schnittstellen korrekt bedienen. Ein solches Schichtenmodell ist beispielsweise auch die TCP/IP-Architektur von 1973, die später die Grundlage des Internets wurde.

Die ANSI-SPARC-Architektur fordert von einer allgemeinen Datenbank, dass sie aus drei Ebenen besteht: Die externe Ebene ist die Schnittstelle zu den Anwendern und stellt die individuellen Benutzersichten bereit, beispielsweise Formulare, Eingabemasken oder Ausgabelisten für die verschiedenen Nutzergruppen. In der logischen oder konzeptionellen Ebene werden die zu speichernden Daten beschrieben und mit dem Ziel modelliert, sie effizient verwaltbar und zugreifbar zu machen und gleichzeitig den Anwenderanforderungen zu genügen. Die interne Schicht schließlich stellt die physikalische Sicht auf das Datenbanksystem dar und beschreibt,

in welchen Dateien welche Daten genau gespeichert werden, wie bei Mehrbenutzersystemen die lesenden und schreibenden Zugriffe geregelt werden usw.
Die ANSI-SPARC-Architektur wurde zwar nie ein offizieller Standard. Auch kann bei der Implementierung eine klare Trennung zwischen den Ebenen kaum gelingen; insbesondere sind die externe und die logische Ebene eng verzahnt, so dass eine ideale logische Datenunabhängigkeit unrealistisch ist. Dennoch gilt ANSI-SPARC noch heute als das ideale Entwurfsmodell für ein Datenbanksystem[3]. Wir definieren daher den exakten Begriff Datenbanksystem wie folgt:

Definition 1.1. Ein *Datenbanksystem* ist ein Softwaresystem zur Speicherung und Verwaltung strukturierter Daten, das die drei Ebenen der ANSI-SPARC-Architektur realisiert. Die Menge der gespeicherten Daten heißt *Datenbank* oder auch *Datenbasis*, die Menge an Programmen zum Zugriff auf die Daten heißt *Datenbankmanagementsystem (DBMS)*.[4]

1.3 Eigenschaften eines Datenbankmanagementsystems (DBMS)

Ein Datenbankmanagementsystem hat die Aufgabe alle Datenbanken im System zu verwalten und dem Benutzer der Datenbank den Zugriff zur Datenbank zu gewähren. Dazu muss das DBMS sicherstellen, dass

- Daten *persistent*, d.h. dauerhaft gespeichert werden
- Daten *konsistent* und korrekt innerhalb der Datenbank bleiben
- Daten *resistent* gegenüber Soft- und Hardwarefehlern sind
- der Zugriff effizient ist

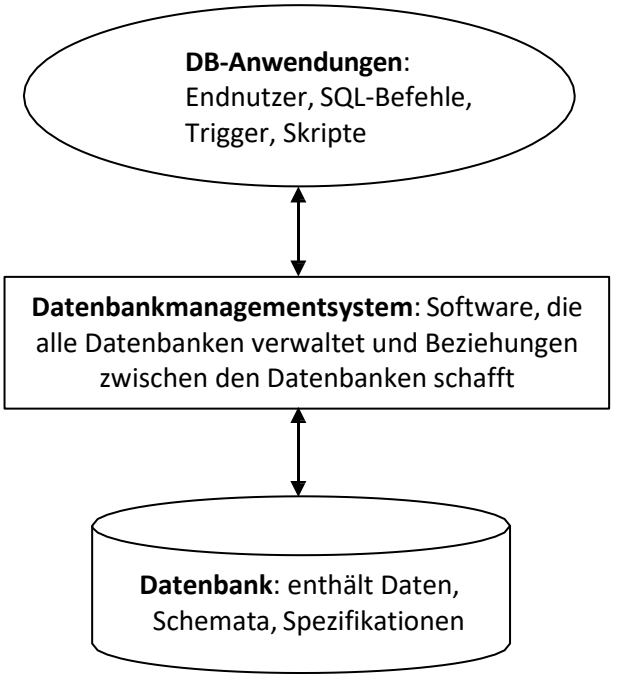


Abbildung 1.3: Komponenten eines Datenbanksystems

[3]Piepmeyer (2011):S. 16.
[4]vgl. Krcmar (2015):§6.1.3.1.

Die Vorteile von DBMS liegen in der einfachen Speicherung von Daten in einer zentralen Sammelstelle, so dass Endbenutzer in der Lage sind auf Daten zuzugreifen, diese zu ändern oder bei Bedarf zu löschen. Außerdem können durch DBMS automatisierte Rollbacks durchgeführt und LOGs gespeichert werden, um jegliche Änderung innerhalb der Datenbanken zu kontrollieren (Datensicherheit und Datenintegrität). Außerdem können innerhalb eines DBMS mithilfe eines Zugriffskonzepts Rechte auf bestimmte Aktionen wie zum Beispiel Schreib- oder Leserechte innerhalb des DBMS vergeben und bei Bedarf auch wieder entzogen werden.

1.4 Arten von DBMS

Datenbankmanagementsysteme werden generell in zwei Haupttypen unterteilt:

a) Relationale Datenbankmanagementsysteme

b) Nichtrelationale Datenbankmanagementsysteme (NoSQL- Datenbankmanagementsysteme)

➔ Zu diesen zählen die hierarchischen, netzwerkartigen, dokumentorientierten und objektorientierten Datenbankmanagementsysteme

In der Praxis sind relationale Datenbankmanagementsysteme die mit Abstand am häufigsten auftretenden DBMS, so dass wir uns im Modul Datenbanken auf diese konzentrieren werden. Die Gründe hierfür werden wir im laufenden Semester kennenlernen.

Gängige **relationale** Datenbankmanagementsysteme sind u.a.

- PostgreSQL
- MySQL
- Oracle Database
- IBM Db2
- MS Access
- MariaDB
- Exasol

1.5 Relationen

Ende der 1960er Jahre entwickelte der IBM-Mitarbeiter Edgar F. „Ted“ Codd ein Datenbankmodell, das zur Strukturierung von Daten Tabellen nutzt. Er realisierte damit das in der Mathematik des 19. und 20. Jahrhunderts intensiv untersuchte Konzept der *Relationen*. Eine Relation R ist mathematisch eine Teilmenge des kartesischen Produkts gegebener Mengen $A_1, ..., A_n$,

$$R \subseteq A_1 \times \ldots \times A_n. \tag{1.1}$$

Formal lassen sich die Elemente der Relation als n-Tupel schreiben:

$$(x_1, \ldots, x_n) \quad \text{mit } x_j \in A_j \text{ für } j = 1, \ldots, n. \tag{1.2}$$

Beispiel 1.2. ‘ Betrachten wir die folgenden zwei Mengen, die jeweils eine der Eigenschaften A_1 = „Farbe“ und A_2 = „Karte“ einer Spielkarte beschreiben.

A_1 = {Kreuz, Pik, Herz, Karo}, A_2 = {2, 3, 4, 5, 6, 7, 8, 9, 10, Bube, Dame, König, Ass}.

Das kartesische Produkt A_1 x A_2 beider Mengen ist dann die Menge aller 4 x 13 = 52 Kombinationen der beiden Eigenschaften, also mit anderen Worten ein komplettes Pokerblatt. Dann ist das folgende „Full House“ R eine Relation dieser beiden Mengen:

$$R = \{(\text{Pik, Ass}), (\text{Kreuz, Ass}), (\text{Herz, Ass}), (\text{Herz, König}), (\text{Karo, König})\}, \tag{1.3}$$

denn $R \subset A_1 \times A_2$ ist eine Teilmenge des kartesischen Produkts $A_1 \times A_2$. Sie kann äquivalent auch durch eine Tabelle beschrieben werden:

R

A_1	A_2
Pik	Ass
Kreuz	Ass
Herz	Ass
Herz	König
Karo	König

oder eben:

Full House

Farbe	Karte
Pik	Ass
Kreuz	Ass
Herz	Ass
Herz	König
Karo	König

Die Menge *aller* Full Houses ist ebenfalls eine Relation von $A_1 \times A_2$.

Bemerkung 1.3. Eine Relation von (endlichen) Mengen mit m Elementen lässt sich wie in obigem Beispiel 1.2 immer als eine Tabelle darstellen, deren Spalten die n Mengen des kartesischen Produkts beschreiben und deren Zeilen die Kombinationen der Werte sind. Umgekehrt ist jede Tabelle eine endliche Relation, wenn man vereinbart, dass nur Tabellen ohne doppelte Zeilen betrachtet werden und zudem zwei Tabellen als gleich angesehen werden, wenn sie sich nur durch Zeilenvertauschungen unterscheiden: [5]

‘nach Piepmeyer (2011):S. 36f.

[5]vgl. Piepmeyer (2011):S. 37.

$$R = \{(x_{i1}, \ldots, x_{in}) : i = 1, \ldots, m \text{ und } x_{ij} \in A_j \text{ für } j = 1, \ldots, n\} \quad \Longleftrightarrow \quad \begin{array}{|cccc|} \hline A_1 & A_2 & \cdots & A_n \\ x_{11} & x_{12} & \cdots & x_{1n} \\ \vdots & \vdots & & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{in} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \\ \hline \end{array} \tag{1.4}$$

In der Informatik werden die Spalten solcher Tabellen auch als *Felder* oder *Attribute* bezeichnet, eine Zeile als *Datensatz*. In der folgenden Übersicht sind diese Grundbegriffe für Daten einer relationalen Datenbank tabellarisch gegenübergestellt und mit dem Objektbegriff der objektorientierten Programmierung verglichen.

Relationales Modell (Mathematik)	**Tabellenmodell** (Angewandte Informatik)	**Objektmodell** (Programmierung)
Relation *(relation)*	Tabelle *(table)*	Klasse *(class)*
Attribut *(attribute)*	Spalte *(column)* / Feld *(field)*	Attribut *(attribute)*
Tupel *(tuple)*	Zeile *(row)* / Datensatz *(record set)*	Objekt *(object)*

Das Objektmodell wird uns in diesem Skript allerdings nicht weiter beschäftigen. Es ist hier erwähnt, um die enge Verwandtschaft der Begriffe Relation, Tabelle und Objekt aus den Bereichen Mathematik, Angewandte Informatik und Programmierung zu zeigen. Da ferner die Begriffe in der Fachliteratur häufig geeignet ausgetauscht werden – z.B. wird oft gesagt, eine Tabelle habe

Attribute – haben wir hier eine Art „Vokabelliste“.

Definition 1.4. Der *Relationstyp* ist definiert als die Kombination des Namens R der Relation und der Auflistung der Mengen $(A_1, \ldots, A_n)$ und wird

$$R(A_1, A_2, \ldots, A_n) \tag{1.5}$$

geschrieben. Ähnlich wie der Datentyp in der Datenhierarchie Abbildung 1.1 ist der Relationstyp also das „Format“ einer Relation. Im Zusammenhang mit Datenbanken spricht man oft auch von dem *Schema* der Relation, manchmal spricht man auch von der *Struktur der Tabelle* oder der *Tabellenstruktur*. Die Notation (1.5) wird später bei der Programmierung von Tabellen noch eine wichtige Rolle spielen.

Beispiel 1.5. Der Relationstyp der Relation „Full House“ aus Beispiel 1.2 lautet

$$\text{Full House (Farbe, Karte).} \tag{1.6}$$

An dieser Notation kann man ablesen, dass die Relation ein 2-Tupel (x_1, x_2) ist, dessen erster Wert x_1 eine Farbe (A_1) und dessen zweiter, x_2, eine Karte (A_2) ist.

In einem später berühmt gewordenen Fachartikel[6] veröffentlichte Codd 1970 seine Gedanken zu Relationen als Grundlage für ein Datenbankmodell. Das Modell fand schnell weite Beachtung. Heute werden auf Tabellen basierende Datenbanken relational genannt, und das Verwaltungssystem entsprechend RDBMS (für relationales Datenbankmanagementsystem).

[6] Codd (1970).

Die Begriffe der Relationen und der Tabellen beziehen sich dabei genau genommen auf die logische Ebene der ANSI-SPARC-Architektur in Abbildung 1.2.

1.6 Anforderungen an Datenbanksysteme nach Edgar F. Codd

Nach Edgar F. Codd müssen Datenbanksysteme bestimmte Voraussetzungen erfüllen, damit grundlegende Datenbankfunktionen effizient ausgeführt werden können. Folgende Anforderungen muss ein Datenbanksystem demnach abbilden können:

• Data Dictionary
Ein Katalog, der die Metadaten des Datenbankmodells enthält

- Integration

Einheitliche Verwaltung aller von Anwendungen benötigten Daten.
Redundanzfreie Datenhaltung des gesamten Datenbestandes

• Operationen
Operationen zur Speicherung und Manipulation der Daten müssen vorhanden sein

• Konsistenzüberwachung
Das DBMS überwacht die Korrektheit der Daten bei Änderungen

- Zugriffskontrolle

Zugriffskontrolle gewährleistet Ausschluss unautorisierter Zugriffe

• Transaktionen
Zusammenfassung einer Folge von Änderungsoperationen zu einer Einheit, deren Effekt bei Erfolg permanent in der Datenbank gespeichert wird

• Synchronisation
Arbeiten mehrere Benutzer gleichzeitig mit der Datenbank dann vermeidet das DBMS unbeabsichtigte gegenseitige Beeinflussungen oder Schreibkonflikte

• Datensicherung
Nach Systemfehlern wie Abstürzen wird die Wiederherstellung ermöglicht

• Benutzersichten
Für unterschiedliche Anwendungen sind unterschiedliche Sichten auf den Datenbestand von Nöten