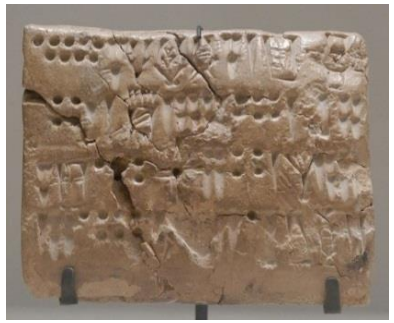
1. Einführung

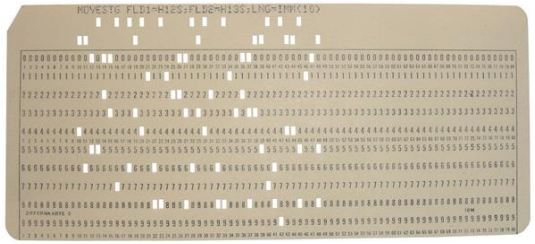
1. **Historische Aspekte und Grundbegriffe**
2. Abgrenzung zu Dateisystemen
3. Inhalte von Datenbanken
4. Architektur von Datenbanksystemen

Motivation von Datenbanken als Buchführungsinstrument

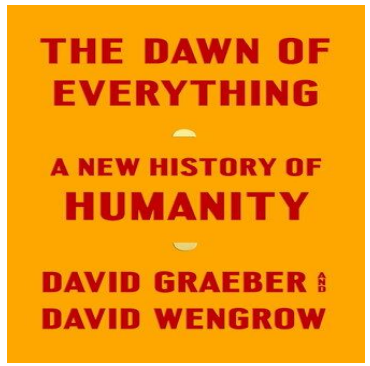
- Tontafeln von Uruk (3.200 v.Chr.)
- Druckerpresse (Gutenberg, ca. 1450)
- Lochkarten (1890)
- PC (1984)
- Verbreitetes Internet/WWW (1991)

exponentielle Tempoverschärfung





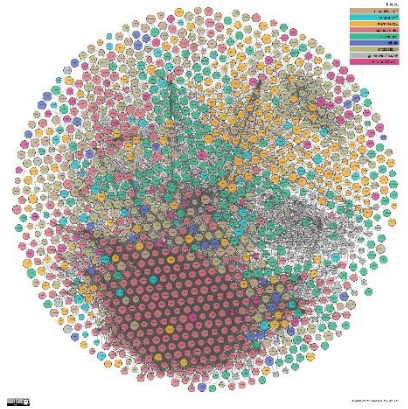
http://datentraeger-museum.de



Variety

Thema der Vorlesung

- Unstrukturierte Daten
  - Daten folgen keinem Format, Schema oder Grammatik
  - Text, Bitströme auf Speichern, rohe Video- & Bilddaten, …
- Semi-Strukturierte Daten
  - Daten folgen einem flexiblen Format, optionale Felder, graphbasierte Daten
  - Webseiten, XML documente, JSON, RDF, …
- Strukturierte Daten
  - Daten folgen einem striktem Schema
  - relationale Datenbanken, Tabellen, Sensordaten, …



Relation "Mitarbeiter"

Personal-nummer	Name	Gehalt
1427	Meier	3217,33
8219	Schmidt	1425,87
2624	Müller	2438,21
⋮	⋮	⋮

Relation "Mitarbeiter in Abteilung"

Personalnummer	Abteilung
1427	3-1
8219	2-2
2624	3-1
⋮	⋮

Einführung

- *Datenbanksysteme* sind zentrale Komponenten großer *Informationssysteme* in vielen Anwendungsbereichen:
  - administrativ-betriebswirtschaftlich: Abrechnung, Buchung, Planung usw.
  - technisch-wissenschaftlich: CAD/CAM/CIM, Geographie, Medizin, Biologie, usw

- Die primäre Aufgabe von Datenbanksystemen umfasst
  - die Beschreibung,
  - die Speicherung und Pflege sowie
  - die Wiedergewinnung

  von (umfangreichen) Datenmengen, die von verschiedenen Anwendungsprogrammen benutzt werden.

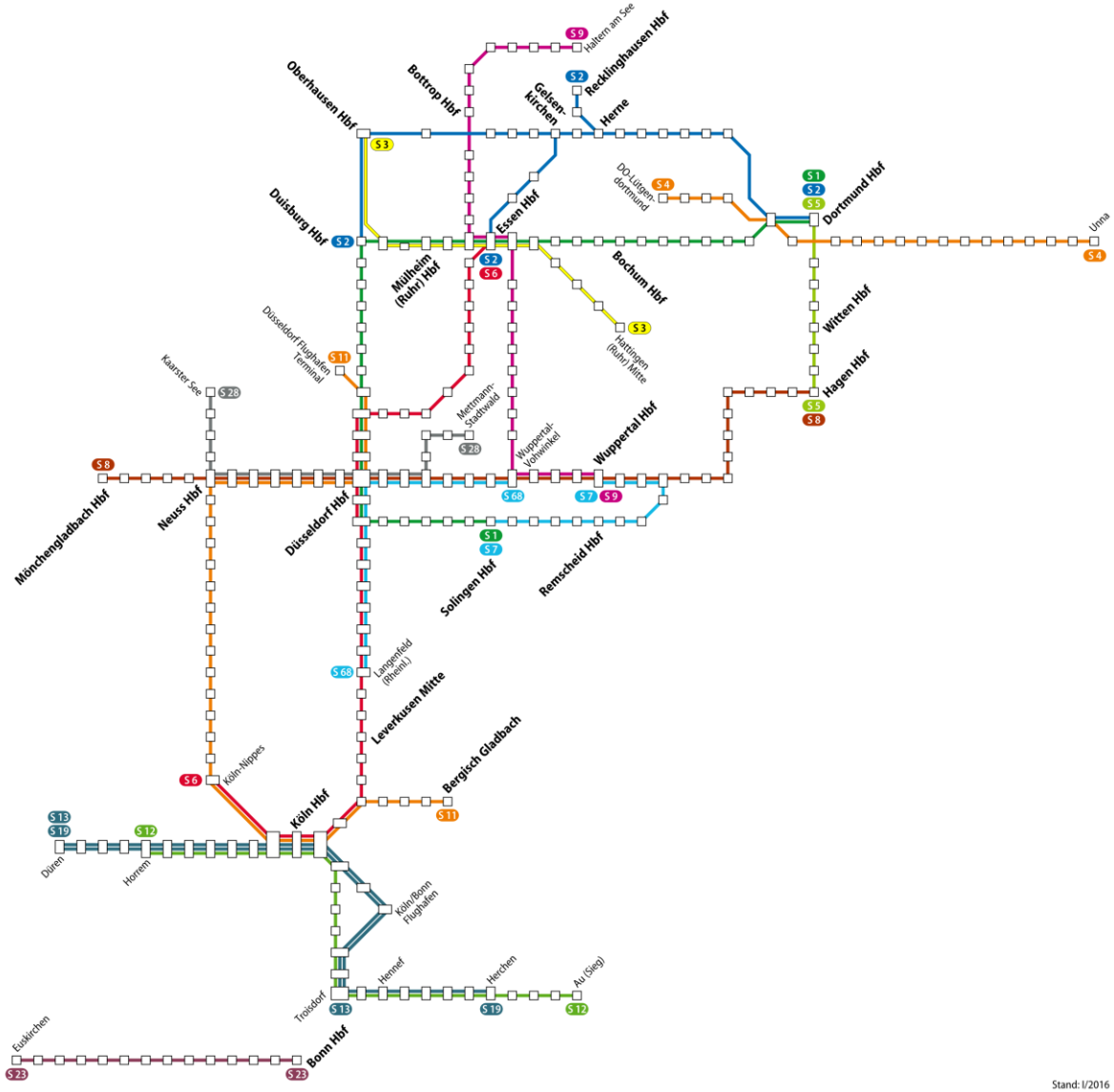
Ausrichtung dieser Vorlesung

- Datenbanken aus Benutzersicht
  - Beschreibung von Daten
    - Datenmodelle, Datenbankschema, Entwurfstheorie
  - Arbeiten mit Datenbanken
    - Anfragesprachen, Transaktionen

- Was ist charakteristisch für Datenbanken (im Gegensatz zu prozeduralen Programmen)?
  - Datenorientierte Programmierung
    - Beschreibe Daten, nicht Abläufe
    - Deklarativer Zugriff: Auf **was** möchte ich zugreifen (nicht: **wie**)?
  - Persistenz von Daten
    - Persistenz: Daten überleben Prozesse
    - „Daten sind schon da“-Programmierung
    - Dadurch Legacy-Problematik: Portierung ist schwieriger

Einschub: Abstraktion in der Informatik [1]

- **Abstraktion** (abstraction) – Das Ableiten oder Herausheben des unter einem bestimmten Gesichtspunkt Wesentlichen/Charakteristischen/Gesetzmäßigen aus einer Menge von Individuen (Dingen, Beobachtungen, ...)
- Abstraktionen sind ein zentrales Mittel für das Erstellen und Verstehen von Systemen und Modellen.



[1] Abschnitt basierend auf: Martin Glinz: Abstraktion. Vorlesung Informatik II: Modellierung, Kap. 12: Abstraktion. Universität Zürich https://files.ifi.uzh.ch/rerg/arvo/ftp/inf_II/inf_II_kapitel_12.pdf

Karte: NordNordWest, Lizenz: Creative Commons by-sa-3.0 de

Abstraktion beim Erstellen von Modellen

Abstraktion ist erforderlich:

- Zur Umsetzung der Pragmatik: Was ist unter welchem Gesichtspunkt wichtig / hervorzuheben?
- Beim Verkürzen: welche Eigenschaften des Originals werden nicht modelliert?

Abstraktion beim Verstehen von Modellen

- Abstraktion ist notwendig zur Beherrschung der Komplexität großer Modelle
- Große Modelle sind ohne einen klar strukturierten Aufbau nicht verstehbar.
- Abstraktion ermöglicht
  - das Sichtbarmachen großer Zusammenhänge unter Weglassung der Details
  - die Darstellung eines Details unter Weglassung/ starker Vergröberung des Rests
  - die Herstellung eines systematischen Zusammenhangs zwischen Übersichten und Detailsichten.

Abstraktionsarten

Die wichtigen Abstraktionsarten für Modelle sind:

- **Klassifizierung:** Charakterisierung der Gemeinsamkeiten von Individuen durch Typen oder Klassen
- **Komposition:** Zusammensetzen einer Menge zusammenhängender Individuen zu einem Ganzen
- **Generalisierung:** Verallgemeinerung der Merkmale einer Menge ähnlicher Typen
- **Benutzung:** Nutzung von Leistungen Dritter durch ein Individuum zwecks Erbringung eigener, höherwertiger Leistungen

- Alle vier Abstraktionsarten sind unabhängig voneinander

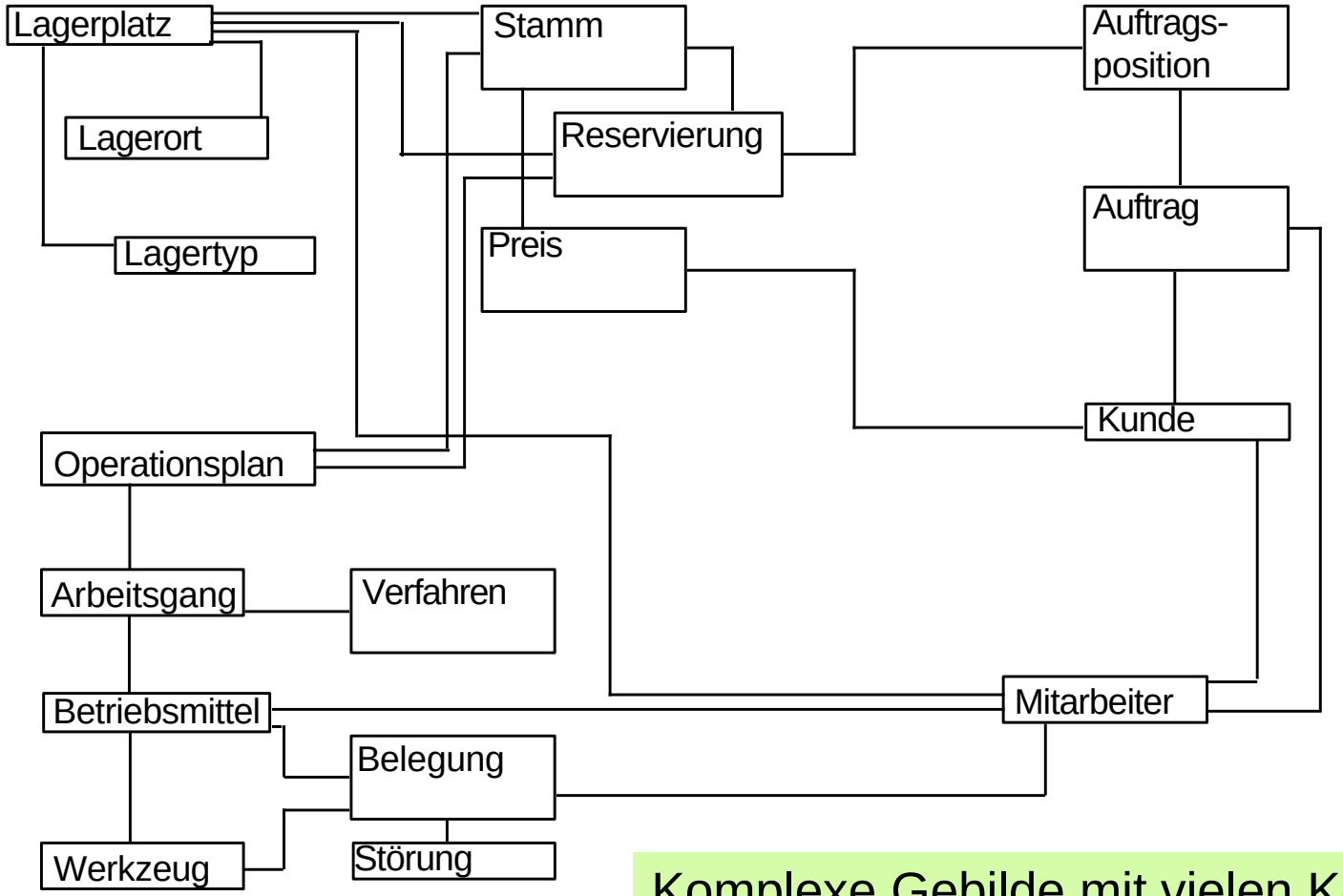
Klassifizierung

Der Typ bzw. die Klasse

- beschreibt die Attribute der Exemplare
- beschreibt den zulässigen Wertebereich zu jedem Attribut
- lässt alle Information zu individuellen Attributausprägungen der Exemplare weg (diese wird zu Wertebereichen abstrahiert)

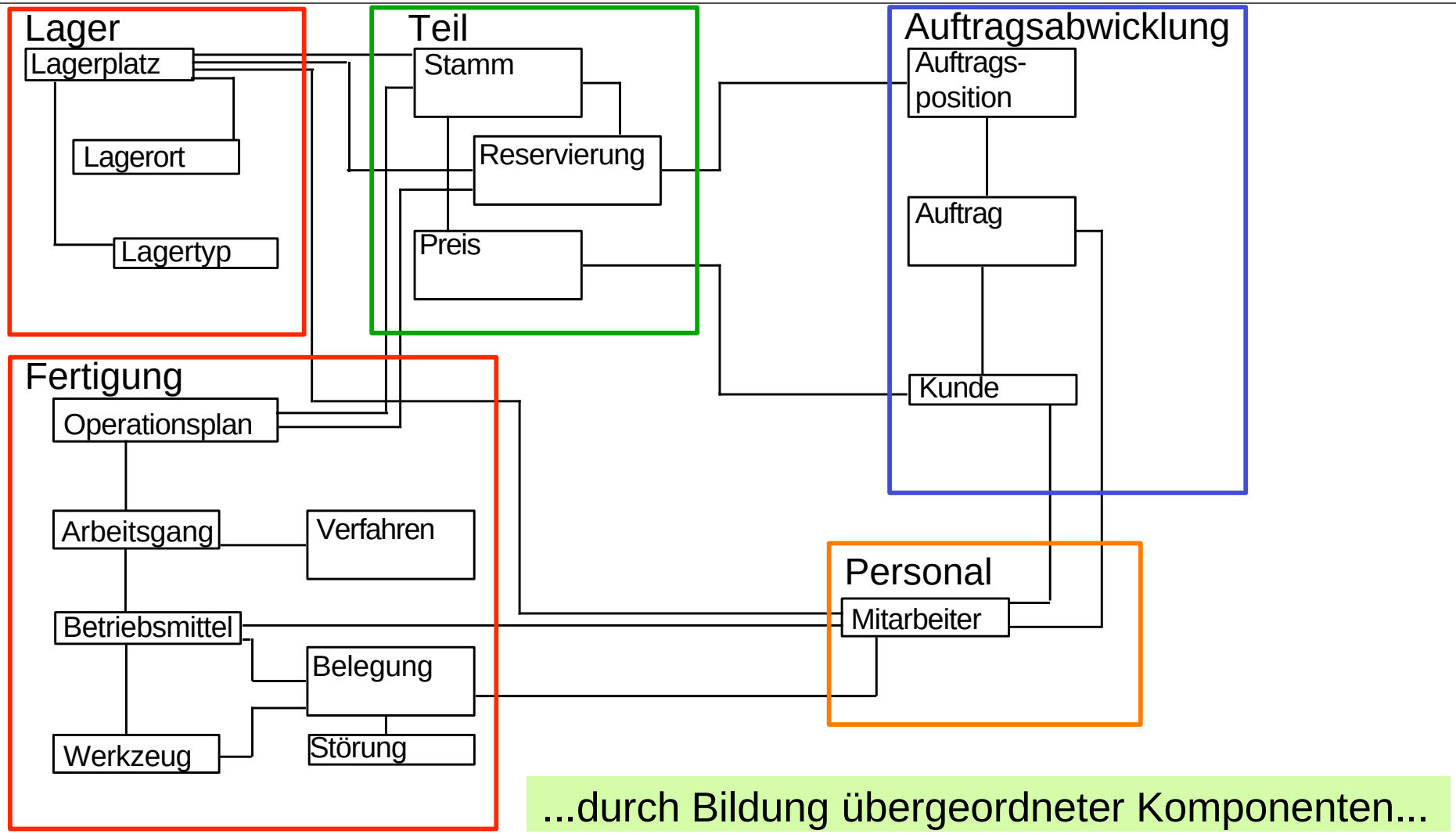
⇨ Zentrale Abstraktion bei jeder Modellbildung

Komposition



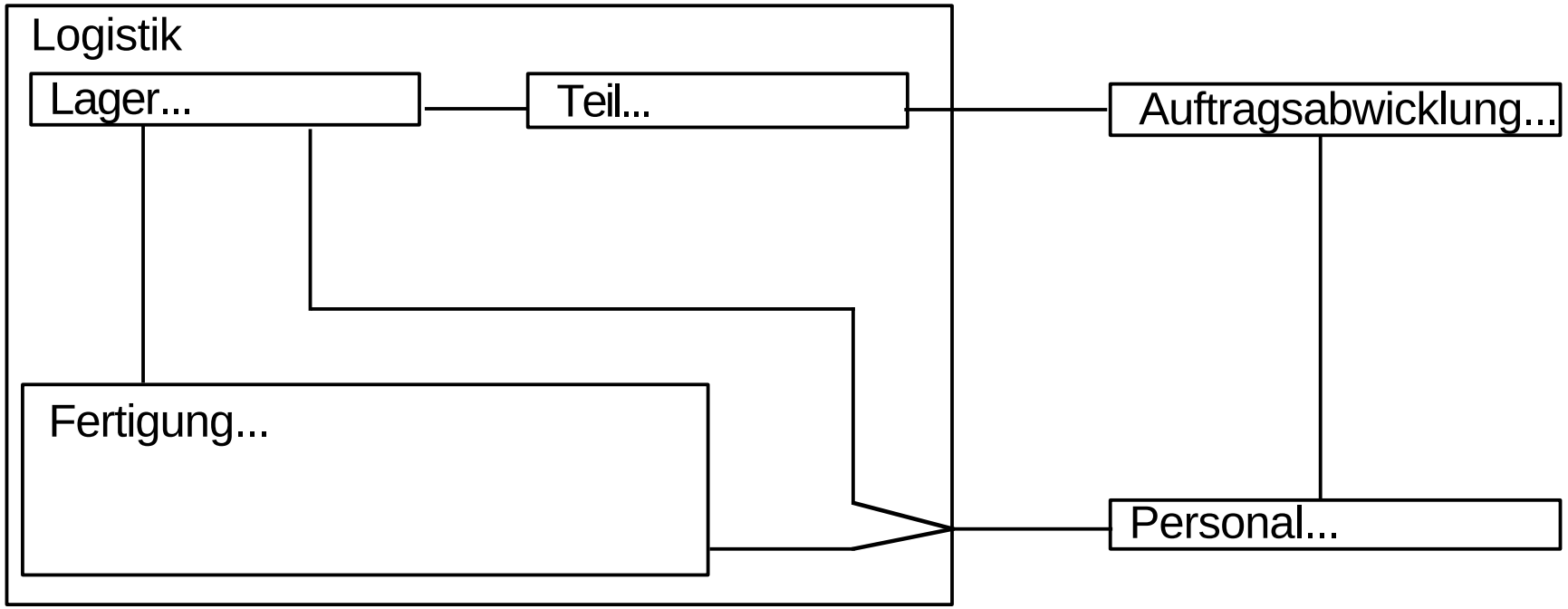
Komplexe Gebilde mit vielen Komponenten...

Komposition



...durch Bildung übergeordneter Komponenten...

Komposition-3



...abstrahieren und übersichtlicher machen

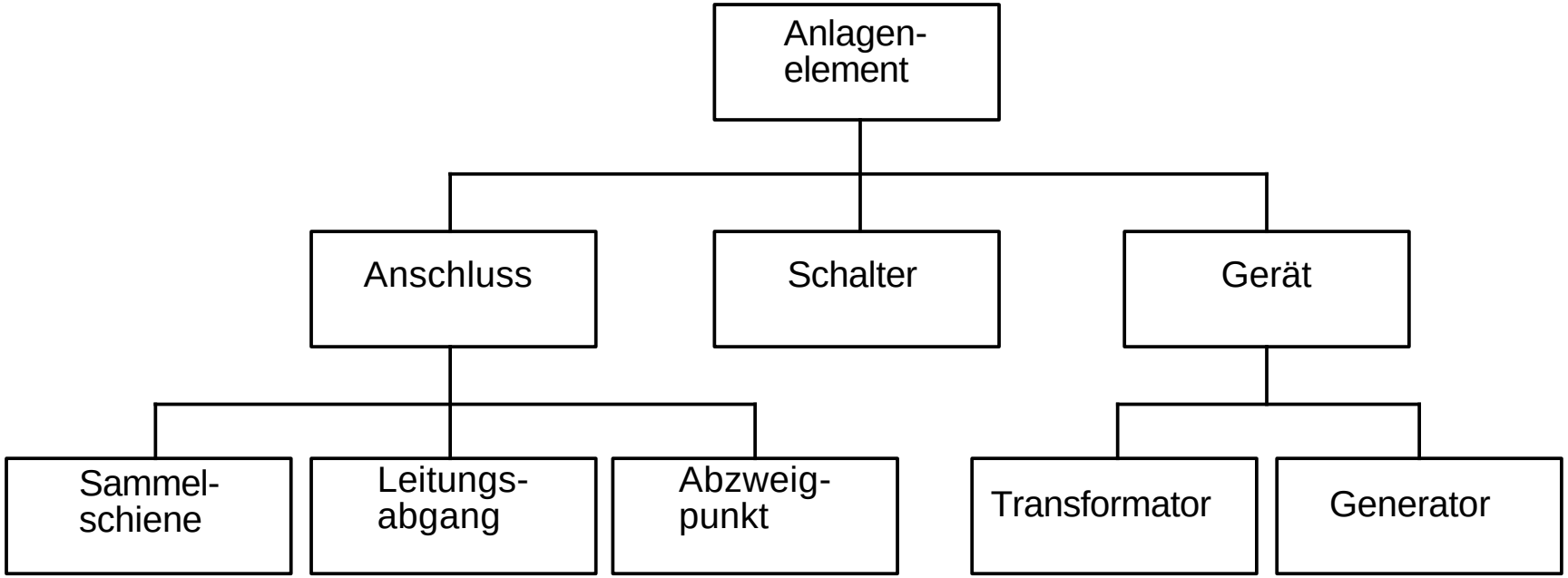
Komposition-4

- **Komposition** (composition) – Zusammenfassung einer Menge von Individuen mit teilweise gemeinsamen Merkmalen zu einem neuen Individuum mit neuen Merkmalen, welches die Gesamtheit der zusammengefassten Merkmale repräsentiert
- Die Komposition
  - fasst eine Menge von Einzelkomponenten (Teilen) unter Weglassung von Details zu einer übergeordneten Komponente (einem Ganzen) zusammen
  - fasst nicht irgendetwas irgendwie zusammen, sondern nur logisch zusammengehörende Komponenten
- Die übergeordnete Komponente (das Ganze) ist ein in sich möglichst geschlossenes Teilmodell

Komposition-5

- Zusammenfassung über mehrere Stufen
  ⇨Kompositionshierarchie (Teil-Ganzes-Hierarchie)
- Kompositionshierarchien bilden in umgekehrter Richtung eine hierarchische Zerlegung eines Modells
- Zentrales Mittel zur Beherrschung der Komplexität von Modellen mit vielen Einzelkomponenten

Generalisierung



Beispiel einer Generalisierungsabstraktion – Modellierung der Begriffswelt eines Problembereichs (Engineering von Unterstationen in Stromverteilungsnetzen)

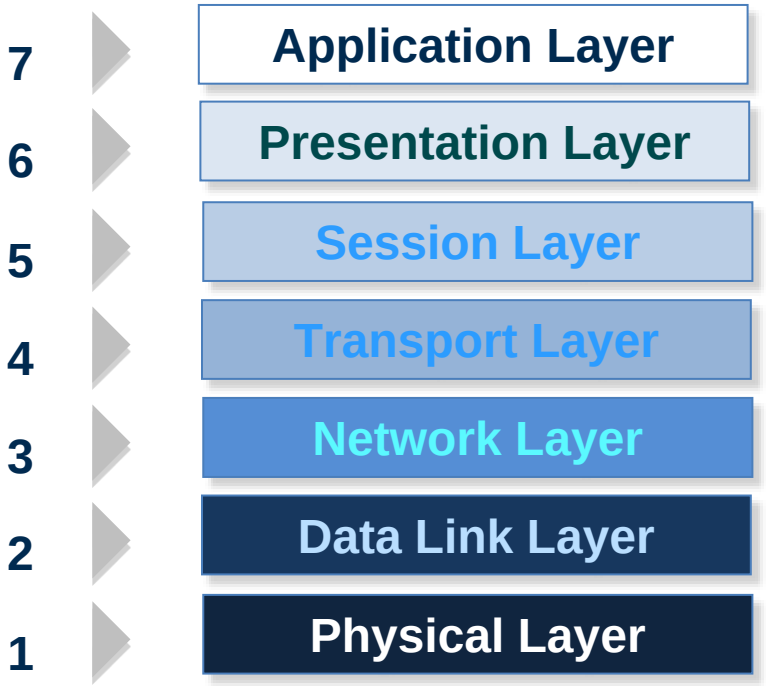
Generalisierung-2

- **Generalisierung** (generalization) – Zusammenfassung einer Menge von Individuen mit teilweise gemeinsamen Attributen durch ein übergeordnetes Individuum, welches nur die gemeinsamen Attribute aufweist.

- Die Bildung von Oberbegriffen zu Begriffen im menschlichen Denken ist eine Generalisierungsabstraktion
- Die Generalisierung erlaubt die systematische Modellierung ähnlicher Dinge (solcher mit teilweise gemeinsamen und teilweise unterschiedlichen Merkmalen)
- Die Bildung von Klassenhierarchien in Klassen- und Objektmodellen basiert auf der Generalisierung
- Generalisierung ist auch in Datenmodellen möglich

Benutzung und Schichtenbildung

Komplexe Gebilde durch Delegieren von Aufgaben und Anordnung der Aufgaben in Schichten übersichtlich und anschaulich modellieren

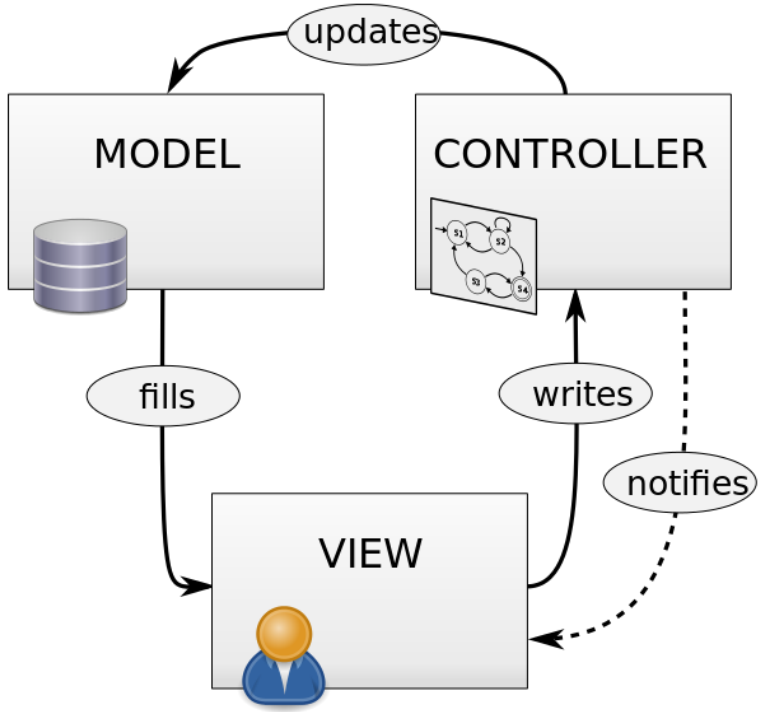


ISO/OSI Schichtenarchitektur

Präsentation

„Business“ Logik

Daten



https://commons.wikimedia.org/wiki/File:MVC_Diagram_%28Model-View-Controller%29.svg

Benutzung und Schichtenbildung-2

- **Benutzung** (delegation, usage) – Verwendung von Leistungen eines Individuums (oder einer Menge von Individuen) durch ein anderes Individuum zum Zweck der Erbringung eigener Leistungen.

- Bündelung von Leistungen zu höherwertigen, komplexen Leistungen; Verwender der komplexen Leistung sehen/kennen die dabei benutzten Leistungen nicht mehr
- Erlaubt die Bildung von Schichten, deren Komponenten
  - sich einerseits auf Leistungen tieferer Schichten abstützen,
  - andererseits Leistungen für höhere Schichten anbieten
  - Metapher der virtuellen Maschinen
  - In der Informatik von Dijkstra (1968) erstmals systematisch verwendet

Benutzung und Schichtenbildung-3

- Benutzer der Leistungen einer Schicht müssen nur wissen, was diese Leistungen sind, nicht aber, wie das Modell im Inneren der leistungserbringenden Schicht und in allen darunterliegenden Schichten aussieht
- Geheimnisprinzip (information hiding, Parnas 1972)
- Verstehen eines Modells, ohne alle Details zu kennen
- Trennung der Belange („Separation of concerns“). Entwickler separater Schichten können sich auf die bestmögliche Ausführung konzentrieren.

Abstraktion in der Informatik

Beispiele für die Anwendung von diesen Abstraktionsprinzipien in dieser Vorlesung:

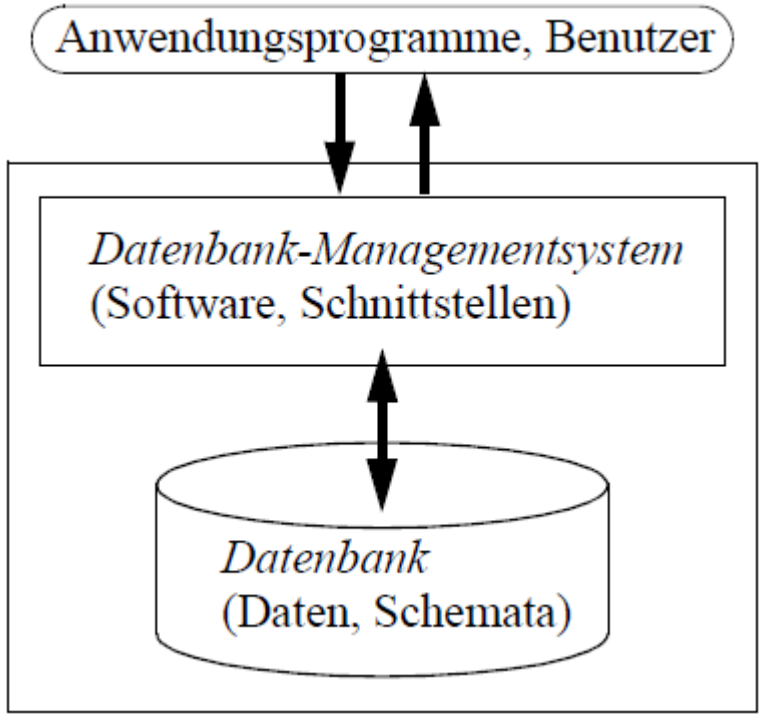
- Architektur von Datenbanksystemen
- Entity Relationship Modellierung
- Relationale Algebra

Literatur

- Dijkstra E.W. (1968). The Structure of the THE multiprogramming System. *Communications of the ACM* **11**, 5 (May 1968). 341-346.
- Ghezzi, C., M. Jazayeri, D. Mandrioli (1991). *Fundamentals of Software Engineering*. Englewood Cliffs: Prentice-Hall.
- Joos, S., S. Berner, M. Arnold, M. Glinz (1997). Hierarchische Zerlegung in objektorientierten Spezifikationsmodellen. *Softwaretechnik-Trends* **17**, 1 (Feb 1997). 29-37.
- Parnas, D.L. (1972). On the Criteria To Be Used in Decomposing Systems into Modules. *Communications of the ACM* **15**, 12 (Dec. 1972). 1053-1058.

Grundbegriffe

- Ein **Datenbanksystem** (DBS) besteht aus:
  - **Datenbank-Managementsystem** (DBMS)
    - Programmsystem zur Verwaltung der Datenbank (Änderungen, Zugriffskontrolle)
  - **Datenbank** (DB)
    - Dauerhaft gespeicherte Sammlung aller Daten und der zugehörigen Beschreibungen



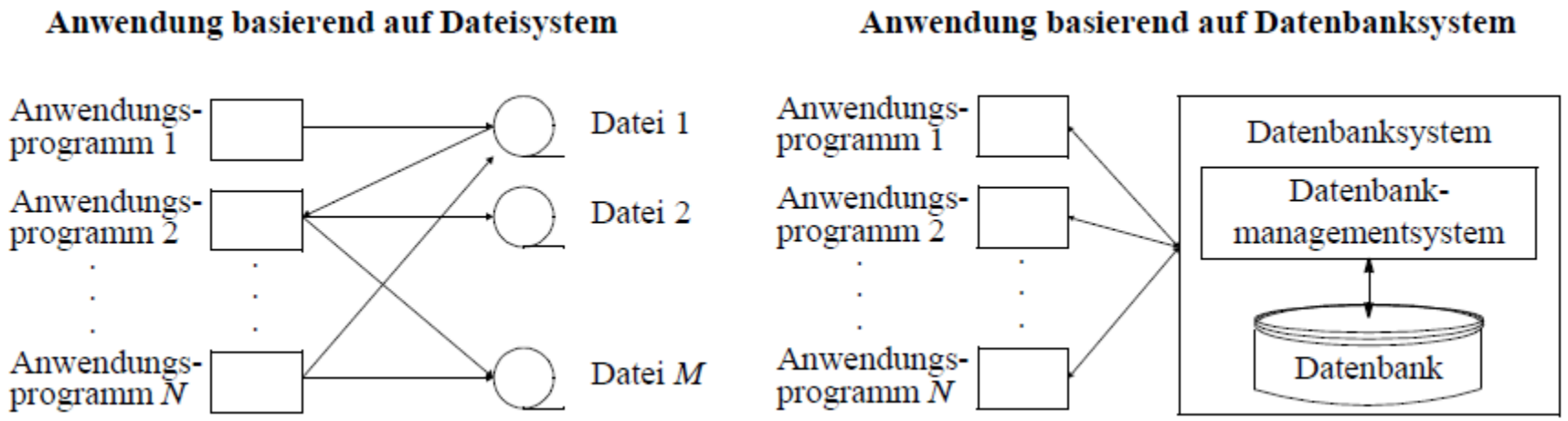
1. Einführung

1. Historische Aspekte und Grundbegriffe
2. **Abgrenzung zu Dateisystemen**
3. Inhalte von Datenbanken
4. Architektur von Datenbanksystemen

Abgrenzung zu Dateisystemen

- Datenbanksysteme ermöglichen die Verwaltung großer Datenbestände in einem einheitlichen System anstatt in einer Vielzahl konventionellen Dateien

- Einsatz von Dateiverwaltungssystemen und Datenbanksystemen:



Charakteristika von Datenbanken: Datenunabhängigkeit

- Datenbank-Managementsystem entkoppelt Daten von Speicher- und Zugriffsstrategien

- **Logische Datenunabhängigkeit**
  - Die logische Unabhängigkeit von Programmen und Daten bewirkt eine Immunität von Anwendungsprogrammen bzgl. Änderung der logischen Gesamtsicht der Daten, wie sie z.B. bei der Einführung neuer Attribute in Datensätzen und neuer Beziehungen zwischen Objekten auftreten.

- **Physische Datenunabhängigkeit**
  - Die physische Unabhängigkeit von Programmen und Daten führt zu einer Immunität von Anwendungsprogrammen bzgl. Änderung der Datendarstellung und Datenspeicherung, wie sie z.B. beim Einsatz effizienter Indexstrukturen auftreten.

Weitere Charakteristika von Datenbanken

- **Verminderte Redundanz**
  - Jedes Datum wird nur einmal in der Datenbank gespeichert. Kopien der Daten werden nur aus internen Sicherheits- oder Effizienzgründen gehalten (*kontrollierte Redundanz*). Dies spart Speicherplatz und erleichtert die Beachtung von *Integritätsbedingungen* in der Datenbank.

- **Einhaltung der Datenintegrität**
  - Die zentralisierte Kontrolle der Daten erlaubt eine einfachere Überprüfung der Daten auf Korrektheit und Vollständigkeit (*Datenintegrität*).

- **Verbesserter Schutz der Daten**
  - Der einheitliche und kontrollierte Zugang aller Anwender zur Datenbank erleichtert die Durchführung von Datenschutz- und Datensicherheitsmaßnahmen.

- **Erleichterung von Standardisierungen**
  - Ein klar organisierter Zugriff auf die Daten erleichtert die Durchsetzung von internationalen Standards bei der Datenmodellierung und bei den Anfragesprachen.

1. Einführung

1. Historische Aspekte und Grundbegriffe
2. Abgrenzung zu Dateisystemen
3. **Inhalte von Datenbanken**
4. Architektur von Datenbanksystemen

Inhalte von Datenbanken: Zwei Ebenen

- **Datenbankschema** (intensionale Ebene)
  - beschreibt *möglichen* Inhalt einer Datenbank
  - Struktur- und Typeninformation über die Daten (“Metadaten”)
  - Art der Beschreibung wird durch das *Datenmodell* vorgegeben
  - Änderungen sind typischerweise eher selten (“Schemaevolution”); einschneidende Schemaänderungen ziehen trotz logischer Datenunabhängigkeit aufwendige Änderungen des Datenbankinhalts nach sich (“Migration”)

- **Ausprägung einer Datenbank** (extensionale Ebene)
  - *tatsächlicher* Inhalt der Datenbank (Datenbankzustand)
  - Objektinformationen, Attributwerte
  - Struktur der Daten ist durch das Datenbankschema vorgegeben
  - Änderungen können sehr häufig auftreten (z.B. Flugbuchungssystem, Kontoführung, …)

- Eine Datenbank muss sowohl Informationen zum Datenbankzustand (Extension) als auch zum Schema (Metadaten) verwalten

Datenmodelle

- **Datenmodelle** sind Formalismen zur Beschreibung aller in der Datenbank enthaltenen Objekte und ihrer Beziehungen untereinander (Datenbankschema), wie sie auf der konzeptionellen und externen Ebene benötigt werden.

- Datenmodelle decken folgende Kernaspekte in unterschiedlicher Weise ab:

  - **Strukturen**
    - Objekttypen und Beziehungen zwischen Objekten

  - **Operationen**
    - Extraktion und Verknüpfung von Daten
    - Anfragesprachen: SQL, OQL, XQuery, SPARQL

  - **Integritätsbedingungen**
    - Modellinhärente: Schlüsseleigenschaften, Typsicherheit, etc.
    - Benutzerdefinierte: Beschränkte Wertebereiche, etc.

Relationales Daten(bank)modell

- Das **relationale Modell** basiert auf dem Strukturierungsprinzip “*Mengen*” (Tabellen, Relationen)
- Vorgeschlagen von Edgar F. Codd (1970)
- IBM System R: erste kommerzielle Implementierung (1980)
- Sehr hohe praktische Bedeutung, sehr viele Implementierungen, Industriestandard
- Steht in dieser Vorlesung im Vordergrund

Relation “Mitarbeiter”

Personal-nummer	Name	Gehalt
1427	Meier	3217,33
8219	Schmidt	1425,87
2624	Müller	2438,21
⋮	⋮	⋮

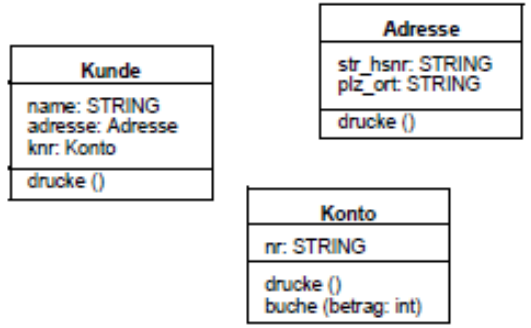
Relation “Mitarbeiter in Abteilung”

Personalnummer	Abteilung
1427	3-1
8219	2-2
2624	3-1
⋮	⋮

Objektorientiertes Daten(bank)modell

- Im **objektorientierten Modell** wird das Konzept “allgemeiner Graph” aufgegriffen
- Zu den zentralen Konzepten *Objekte*, *Attribute* und *Beziehungen* gibt es im objektorientierten Modell noch *Methoden* zur Beschreibung des Verhaltens von Objekten:
  - *Objekttypen* vollständige Beschreibungen gleichartiger Objekte
  - *Objektidentität* dient zur Unterscheidung einzelner Objekte (vs. Werteidentität)
  - *Attribute* speichern den Zustand eines Objektes (statischer Aspekt)
  - *Methoden* beschreiben das Verhalten von Objekten (dynamischer Aspekt)
  - *Beziehungen* werden im OO-Modell weiter unterschieden

- Gute Anbindung an die objektorientierte Programmierung
  - Modellierung durch UML-Klassendiagramme
- Hat sich dennoch in der Praxis nicht stark durchgesetzt
  - Schwierige Optimierbarkeit wg. prozedural geprägter Elemente



Weitere Daten(bank)modelle

- **Objektrelationales Datenmodell**
  - Erweiterung des relationalen Datenmodells um objektorientierte Konzepte
  - *Vorteil*: Investitionsschutz für bestehende relationale Systeme (bei Hersteller und Kunden)
  - *Nachteil*: Komplexität, zum Teil noch Performanzprobleme

- **eXtended Markup Language (XML)**
  - Ursprünglich als reines Daten- und Dokumentaustauschformat zwischen Anwendungen gedacht
  - Für Datenbanken um Anfragesprachen erweitert (XPath, XQuery, XSLT)

- **NoSQL oder Graph-Datenmodelle**
  - Keine festen Schemainformationen
  - Key-Value-Stores, Graphdatenbanken
  - Beispiele: JSON, CouchDB, MongoDB, TripleStores
  - Vgl. Vorlesung „Semantic Web“ (Decker)

Datenmodelle: Allgemeine Unterscheidung

- **Strukturierte Datenmodelle**
  - Relationales Modell: starre Tabellenstruktur mit Zeilen (= Tupel) und Spalten (=Attribute)
  - Objektorientiertes Modell: Objekte mit Attributen und Methoden

- **Semistrukturierte Datenmodelle**
  - XML, etc.: Strukturelemente (Tags, z.T. mit Attributen) in zum Teil beliebig freier Zusammensetzung, auch unstrukturierte Inhalte (zwischen den Tags)
  - Resource Description Framework (RDF): gelabelte Knoten und Kanten. Dabei sind Labels entweder URIs oder Literale (Text).

- **Unstrukturierte Datenmodelle**
  - BLOBs (binary large objects) in relationalen und objektrelationalen Datenbanken:
    - interne Struktur für Datenbanksystem nicht erkennbar
    - definiert durch Anwendungen (Bsp. JPEG)
  - Key-Value-Stores, NoSQL: Beliebig zusammengesetzte Datenelemente

1. Einführung

1. Historische Aspekte und Grundbegriffe
2. Abgrenzung zu Dateisystemen
3. Inhalte von Datenbanken
4. Architektur von Datenbanksystemen

1. Einführung

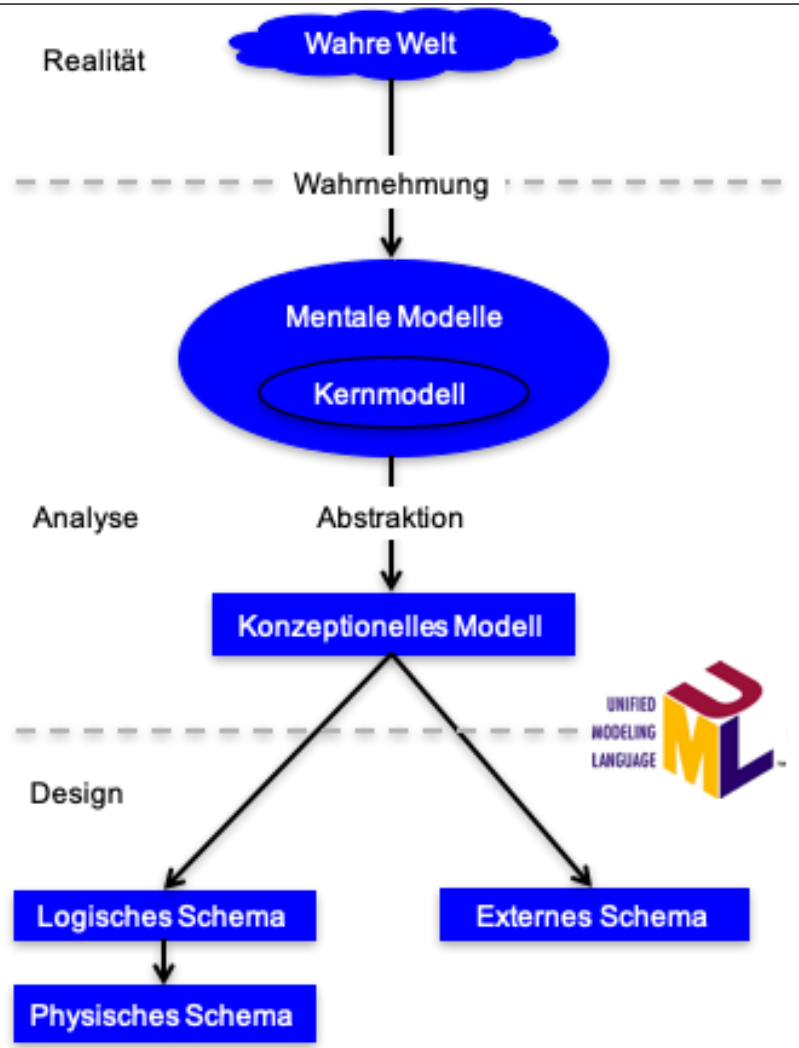
1. Historische Aspekte und Grundbegriffe
2. Abgrenzung zu Dateisystemen
3. Inhalte von Datenbanken
4. **Architektur von Datenbanksystemen**

Von der Realität zu Design-Schemata

Die Erstellung eines konzeptionellen Modells erfordert einen Einigungsprozess der Beteiligten Stakeholder auf Basis einer gemeinsamen Vorstellung der Realität

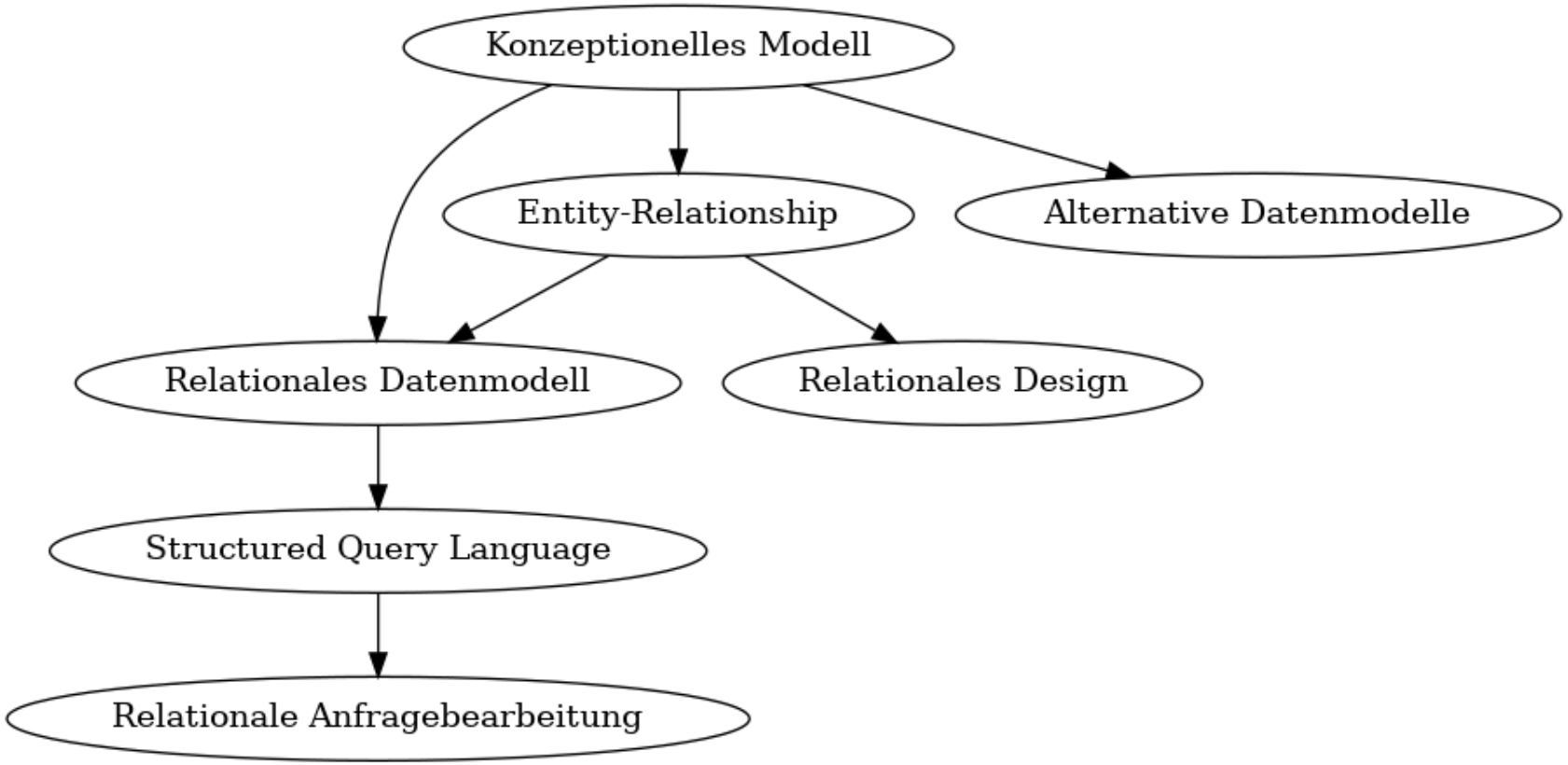
Die Gestaltung der Schemata erfolgt auf Basis funktionaler und nicht-funktionaler Anforderungen sowie von Randbedingungen.

Der Lebenszyklus dieser Schemata ist oft langlebig und nicht an einzelne Softwareentwicklungsprojekte gebunden. Standardisierungsbemühungen können sehr aufwändig sein und hohe Kosten verursachen - Beispiel: ERP Systeme (SAP).



Vom Konzeptuellem Modell zum Relationalem Modell

Verweise auf Kapitel dieser Vorlesung



Datenmodelle, Datenbankmodelle, Datenstrukturen

- **Konzeptuelle Modelle**

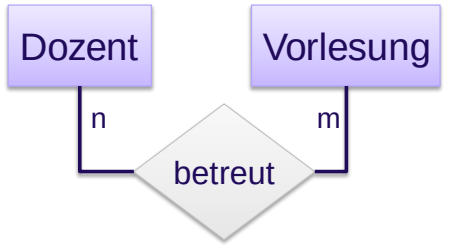
  Für Entwurfsphase
  - Entity-Relationship-Modell
  - Unified Modeling Language (UML)
  - „Ontologien“ (Wissensrepräsentation)

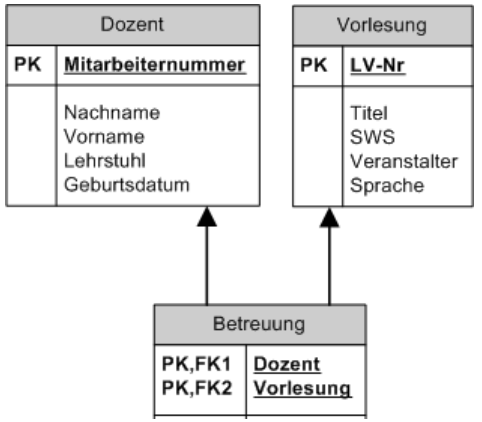
- **Logische Datenbankschema**

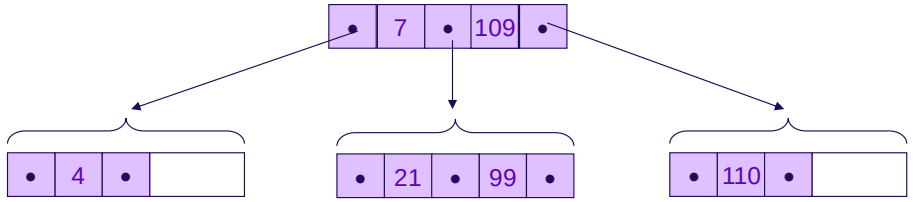
  Für Gesamtsicht und Benutzersichten
  - Relationales Datenbankmodell
  - Objektorientiertes Datenbankmodell
  - Semi-strukturiertes XML-Datenmodell

- **Physische Datenstrukturen**

  Für interne Datenspeicherung
  - B*-Bäume, Hashing, verkettete Listen, …



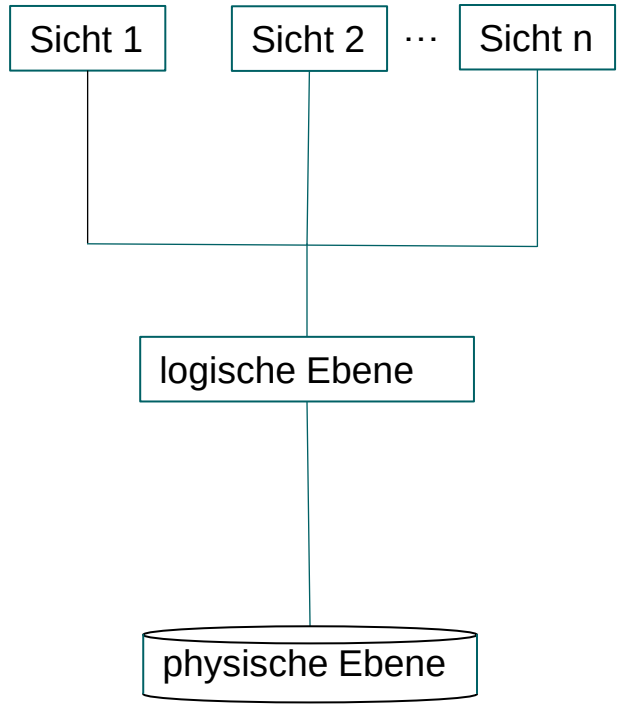




Abstraktionsebenen eines Datenbanksystems: 3-Ebenen Modell nach ANSI/SPARC (1975)

Zur Realisierung der physischen und logischen Datenunabhängigkeit sind Datenbanksysteme typischerweise in drei verschiedenen Abstraktionsebenen aufgebaut

- **Externe Ebene, Benutzersichten**
  - Unterschiedliche Sichten von Benutzern / Benutzergruppen auf den Datenbestand
  - Betrachtung einer Teilmenge der Daten, die für eine Anwendung benötigt wird (z.B. alle Studierenden eines Studiengangs)
- **Konzeptionelle Ebene, Logische Ebene**
  - Logische Gesamtsicht aller Daten; Abstraktion zwischen externer und interner Ebene
  - Logische Organisation der Daten, z.B. in Tabellen
- **Interne Ebene, Physische Ebene**
  - Festlegungen zur physischen Datenorganisation; Zugriffsalgorithmen
  - Ablegen, Verwalten und Finden der Daten auf einem Speichermedium (meist: Festplatte)



Konzeptuelle Ebene funktioniert als Abstraktion zwischen externer und interner Ebene