Datenbanken 1

Einführung

Nikolaus Augsten

nikolaus.augsten@sbg.ac.at
FB Computerwissenschaften
Universität Salzburg

http://dbresearch.uni-salzburg.at

Sommersemester 2019

Version 19. März 2019

Inhalt

Inhalt

Alle Infos zu Vorlesung und Proseminar:

http://dbresearch.uni-salzburg.at/teaching/2019ss/db1/

Inhaltsübersicht Datenbanksysteme/1

1. **Einführung in Datenbanksysteme**
   - Gebiet, Terminologie, Datenbanksysteme
   - Kapitel 1 in Kemper und Eickler
2. **Datenbankentwurf**
   - Datenbank Entwurf, ER Modell
   - Kapitel 2 in Kemper und Eickler
3. **Das relationale Modell**
   - Relationales Modell, relationale Algebra
   - Kapitel 3 in Kemper und Eickler

Inhaltsübersicht Datenbanksysteme/2

4. **SQL**
   - Schemadefinition, Datenmanipulation, Datenabfrage, JDBC
   - Kapitel 4 + 5 in Kemper und Eickler
5. **Relationale Entwurfstheorie**
   - Funktionale Abhängigkeit, Normalformen, Zerlegung von Relationen
   - Kapitel 6 in Kemper und Eickler
6. **Vertiefende Konzepte**
   - Indexstrukturen, Transaktionen
   - Auszüge aus Kapitel 7 und 9 in Kemper und Eickler

Inhalt

Daten sind wertvoll

- Unternehmen: Information über Kunden, Lieferanten, Waren, Bestellungen, Buchhaltung, Marktstudien, usw.
- Verwaltung: Meldedaten, Führerschein, Strafregister, Krankenversicherung, etc.
- Wissenschaft: Messdaten, fachspezifische Datensammlungen (z.B. Protein-Eigenschaften), Analyseergebnisse
- Privat: Telefonnummern, Email-Kontakte, Online-Zugänge, Familienfotos, MP3-Sammlung
- ...

Daten verwalten ist schwierig

Anforderungen an Datenverwaltung:

- Daten müssen (möglichst schnell) zugänglich sein
- Einzelne Fakten müssen verknüpft werden können
- Daten müssen geändert werden können
- Mehrere Benutzer sollen gleichzeitig lesen und ändern können
- Daten müssen konsistent bleiben
- Daten dürfen nicht verloren gehen
- Daten müssen vor unberechtigtem Zugriff geschützt sein

Ansatz ohne Datenbanksystem:

- Daten in isolierten Dateien speichern
- nach Bedarf Programme zum Einfügen, Auslesen und Ändern der Daten schreiben

Probleme der Datenverwaltung/1

- Redundanz und Inkonsistenz
  - Redundanz: ein Fakt ist mehrmals gespeichert
  - bei Änderungen müssen alle Kopien geändert werden
  - Inkonsistenz: nicht alle Kopien wurden geändert, d.h., es existieren widersprüchliche Daten
  - isolierte Dateien: habe ich alle relevanten Dateien geändert?
  - Bespiel: Adresse wurde nur im Fachbereich geändert, auf Universitätsebene hingegen nicht.
  - Ziel: Redundanz kontrollieren und Inkonsistenz vermeiden.
- Beschränkte Zugriffsmöglichkeiten
  - Verknüpfungen logisch verwandter Daten erzeugt deutlichen Mehrwert
  - isolierte Dateien: verschiedene Verwalter und Formate, eigenes Programm für jede Verknüpfung
  - Bespiel: freien Hörsaal für Datenbank-Vorlesung finden (Hörsäle mit Kapazität, Veranstaltungskalender, Teilnehmerzahl der Vorlesung)
  - Ziel: Alle Daten im System lassen sich flexibel miteinander verknüpfen.

Probleme der Datenverwaltung/2

- Integritätsverletzung
  - Änderungen können zu unerlaubten Zuständen (aus der Sicht der Anwendung) führen
  - oft sind Verknüpfungen zwischen Daten erforderlich, um Integritätsverletzungen zu entdecken
  - Beispiel: Student schreibt sich in Bachelor-Projekt ein, bevor er genug Kreditpunkte gesammelt hat.
  - Ziel: Integritätsregeln formulieren und Verletzungen nicht zulassen.
- Sicherheitsprobleme
  - Nicht alle Benutzer sollen alle Daten sehen.
  - Nur ausgewählte Benutzer sollen bestimmte Daten ändern dürfen.
  - Granularität: Informationsteil, auf den sich der Zugang bezieht, z.B. ganzes Objekt, gewisse Eigenschaften des Objektes
  - Beispiel: Studenten dürfen ihre eigenen Noten sehen, aber nicht die anderer. Eigene Noten dürfen nicht verändert werden.
  - Ziel: Lese- und Schreibrechte flexibel und in feiner Granularität an Benutzer vergeben.

Probleme der Datenverwaltung/3

- Probleme des Mehrbenutzerbetriebs
  - Viele Anwender greifen zugleich auf Daten zu.
  - Beispiel: Flugreservierungssystem
  - Keine Kontrolle: Unerwünschte Anomalien, z.B. "lost updates" = meine Änderungen werden von einem anderen Benutzer überschrieben
  - Dateisysteme bieten nur sehr rudimentäre Kontrollmechanismen, z.B., Sperren auf Dateiebene
  - Rudimentäre Kontrolle: Ineffizient, ein einziger Benutzter kann Datei blockieren.
  - Ziel: Effizienter Mehrbenutzerbetrieb ohne Anomalien.
- Umgang mit Fehlern / Datenverlust
  - Verlust von Daten kann für Unternehmen existenzbedrohend sein.
  - Dateisysteme bieten Backups, aber alles nach Backup geht verloren.
  - isolierte Dateien: Konsistenz zwischen Dateien ist im Fehlerfall nicht garantiert, da sich Dateien unabhängig ändern können
  - Beispiel: Stromausfall oder Systemabsturz während Bankomatbehebung
  - Ziel: Garantien gegen Datenverlust auch im Fehlerfall

Probleme der Datenverwaltung/4

- Effizienz
  - Große Datenmengen erfordern effiziente Algorithmen für Suche, Verknüpfung und Änderung.
  - isolierte Dateien erfordern individuelle Programme für einzelne Anfragen und/oder Datentypen.
  - sehr aufwändig und möglicherweise ineffizient, da die Wahl der Algorithmen von den Daten abhängt, die sich ändern können
  - Ziel: Automatisch effiziente Algorithmen in Abhängigkeit von Anfrage und Daten wählen.
- Hohe Entwicklungskosten
  - Zumindest einem Teil oben genannter Probleme muss sich jeder Anwendungsentwickler stellen.
  - Rad ständig neu erfinden ist zeit- und kostenintensiv
  - Ziel: Komfortable Schnittstelle, die Datenverwaltungsprobleme transparent löst.

Warum Datenbankverwaltungssysteme?

- DBMS lösen Probleme der Datenverwaltung:
  - Unkontrollierte Redundanz wird vermieden.
  - Daten lassen sich flexibel miteinander verknüpfen.
  - Definierte Integritätsregeln können erzwungen werden.
  - Flexible Vergabe von Benutzerrechten.
  - Effiziente Mehrbenutzerkontrolle vermeidet Anomalien.
  - Ausgefeilte Recovery-Komponente schützt vor Datenverlust.
  - Anfrageoptimierung sorgt transparent für effiziente Ausführung.
- Fast alle Unternehmen verwenden Datenbanksysteme, weil es kaum eine Alternative gibt.

Datenbankanwendungen

- Traditionelle Anwendungen:
  - Datenbanken mit Zahlen und Wörtern
- Neuere Anwendungen:
  - Multimedia Datenbanken
  - Geographische Informationssysteme (GIS)
  - Data Warehouses
  - Echtzeit Datenbanken
  - Aktive Datenbanken
  - Viele andere Anwendungen
- Beispiele:
  - Banken (Konten)
  - Unternehmen (Lager, Verkauf)
  - Reservierungssysteme
  - Universität (Studenten, Vorlesungen, Räume)
  - Online Verkäufe (www.amazon.com)
  - Online Zeitungen (www.salzburg.com)

Wann braucht man kein DBMS?

- Hauptgründe gegen DBMS:
  - hohe Anfangsinvestition und möglicherweise zusätzlicher Hardware-Bedarf
  - Overhead für Allgemeinheit, Sicherheit, Mehrbenutzerkontrolle, Recovery, Integrationskontrolle
- DBMS möglicherweise nicht nötig, wenn:
  - einfache Datenbank und Anwendung, die klar definiert ist und sich voraussichtlich nicht ändern wird
  - kein Mehrbenutzerbetrieb
- DBMS nicht geeignet:
  - zwingende Echtzeitanforderungen, die DBMS nicht garantieren kann
  - Daten können aufgrund ihrer Komplexität nicht (nur schwer) modelliert werden
  - spezielle Operationen, die von DBMS nicht unterstützt werden

Datenbankforschung

- Konferenzen
  - SIGMOD – seit 1975
  - VLDB – seit 1975
  - ICDE – seit 1985
  - EDBT – seit 1988
- Zeitschriften
  - ACM Trans. on Database System (TODS) – seit 1976
  - The VLDB Journal (VLDBJ) – seit 1992
  - IEEE Trans. on Knowledge and Data Engineering (TKDE) – seit 1989
  - Information Systems (IS) – seit 1975
- DBLP Bibliographie (Michael Ley, Uni Trier, Germany)
  - ursprünglich für Datenbankforschung, jetzt allgemein Informatik
  - `http://dblp.uni-trier.de/db/`
- DBWorld Mailing Liste
  - `http://www.cs.wisc.edu/dbworld/`

Produkte

- Kommerzielle Produkte
  - Oracle
  - Microsoft SQL Server
  - IBM DB2
  - Teradata
  - Sybase Adaptive Server Enterprise
  - IBM Informix
  - PC "DBMSs": Access, dBase, ...
  - ...
- Open Source Produkte
  - PostgreSQL
  - MySQL
  - MonetDB
  - ...

Wir verwenden PostgreSQL für die Übungen.

Zusammenfassung

- Funktionierende Datenverwaltung ist "mission critical"
- Datenverwaltung wirft Probleme auf:
  - Konsistenz
  - effizienter und flexibler Zugriff
  - Integrität
  - Sicherheit
  - Mehrbenutzerbetrieb
  - Datenverlust
- Datenbanksysteme lösen Probleme transparent für Benutzer
- etabliertes und aktives Forschungsgebiet seit 40 Jahren
- Milliardenumsatz mit Datenbankprodukten

Inhalt

Grundlegende Definitionen/1

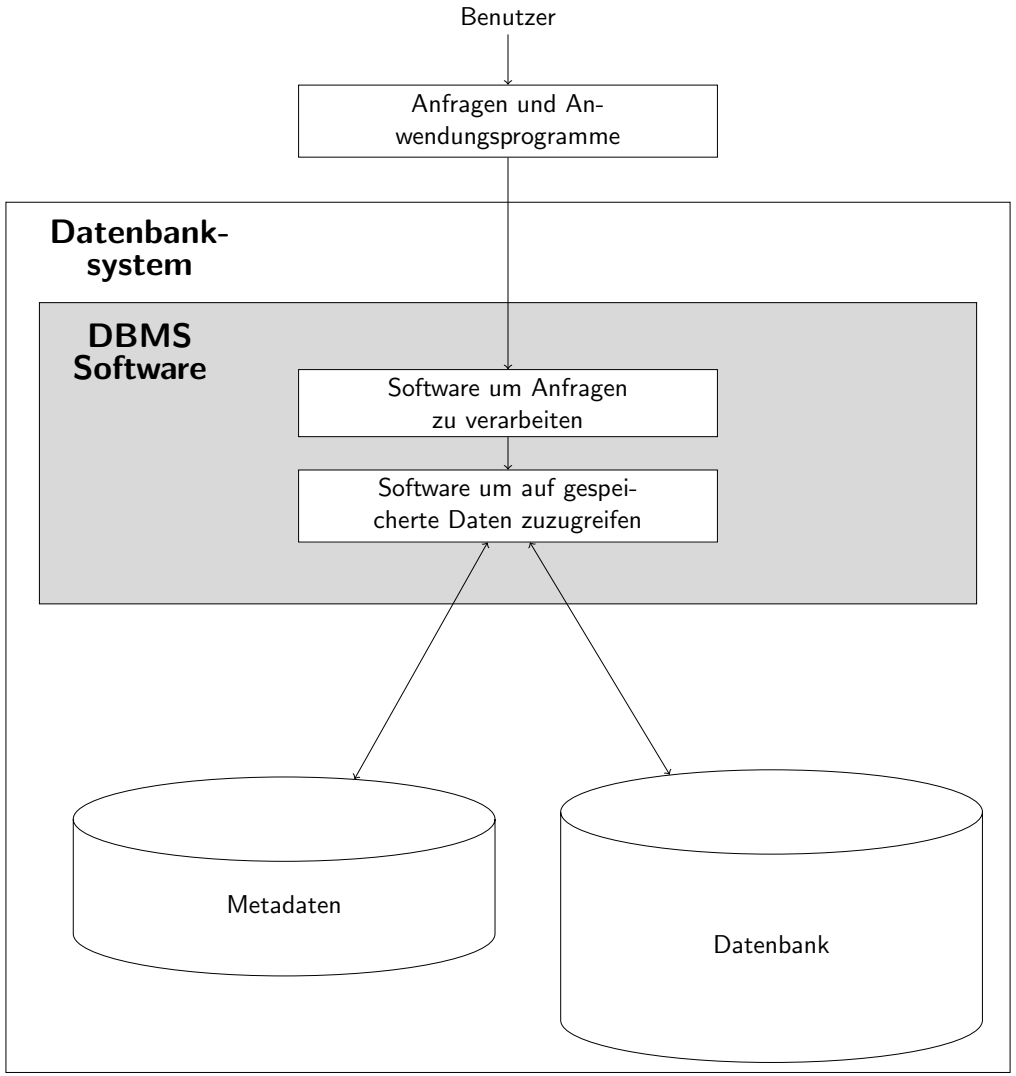
Über **Daten**, **Information** und **Wissen**:

- **Daten** sind Fakten die gespeichert werden können:
  - Buch(Herr der Ringe, 3, 10)
- **Information** = Daten + Bedeutung
  - Buch:
    - Titel = Herr der Ringe,
    - Band = 3,
    - Preis EUR = 10
- **Wissen** = Information + Anwendung

Grundlegende Definitionen/2

- **Mini-Welt**: Jener Teil der realen Welt der uns interessiert
- **Daten**: Bekannte Fakten über die Miniwelt die gespeichert werden können
- **Datenbank (DB)**: Eine Menge von Daten die miteinander in Beziehung stehen
- **Metadaten**: Information über die Struktur einer Datenbank (selbst als Datenbank organisiert)
- **Datenbankverwaltungssystem (DBMS, database management system)**: Ein Software Paket das die Erstellung, Pflege und Abfrage von Datenbanken unterstützt
- **Datenbanksystem (DBS, database system)**:
  DBMS + DB + Metadaten

Grundlegende Definitionen/3



Datenmodel

- "Infrastruktur" zur Modellierung der realen Welt:
  - **Datendefinitionssprache**: beschreibt Datenobjekte und Integritätsbedingungen
  - **Datenmanipulationssprache**: anwendbare Operatoren und deren Wirkung
- Analog zu Programmiersprache:
  - Typenkonstruktoren (Typendefinition)
  - Sprachkonstrukte (if-then, Schleifen, etc.)

DDL und Schema

- Datendefinitionssprache (DDL, data definition language) beschreibt:
  - **Schema**: Struktur der Datenobjekte (Typen, Gruppierung elementarer Typen) und Beziehung zwischen den Datenobjekten
  - **Integritätsbedingungen**: Einschränkung der zulässigen Daten; müssen zu jedem Zeitpunkt erfüllt sein
- Datenbankverzeichnis (Katalog): speichert Metadaten
  - Schema und Integritätsbedingungen
  - weitere Metadaten wie Zugriffsrechte, Statistiken über Datenverteilung

Haupteigenschaften des Datenbankansatzes/3

Beispiel eines Datenbankverzeichnisses (stark vereinfacht):

Tabellen

RelationName	NrOfColumns
Studenten	4
Vorlesungen	4
Module	5
Notenblätter	3
Voraussetzungen	2

Spalten

ColumnName	DataType	BelongsToRelation
Name	CHARACTER(30)	Studenten
StudentNr	CHARACTER(4)	Studenten
Class	INTEGER(1)	Studenten
...	...	...

- PostgreSQL 9.2: 72 Objekte im Datenbankkatalog
- Oracle: über 1000 Objekte im Datenbankkatalog

Datenmanipulationssprache

- Die Datenmanipulationssprache (DML, data manipulation language) besteht aus:
  - **Anfragesprache**: beantwortet Anfragen, lässt Daten unverändert
  - **eigentliche Manipulationssprache**: einfügen, löschen, ändern von Daten

Anfragesprachen

- Sprache um Information aus der Datenbank zu holen
- Kategorien von Sprachen:
  - **Imperativ**[1]: spezifiziert **wie** etwas gemacht wird; kann als Grundlage für die Anfrageoptimierung verwendet werden (weil das Vorgehen bzw. die Reihenfolge angegeben wird)
  - **Deklarativ**: spezifiziert **was** gemacht wird; nicht geeignet für die Anfrageoptimierung
- Reine Sprachen:
  - Relationale Algebra (imperativ)
  - Tupelkalkül (deklarativ)
  - Domänenkalkül (deklarativ)
- Reine Sprachen sind die Basis für Sprachen, die in der Praxis verwendet werden.

[1] "imperativ" und "prozedural" werden manchmal synonym verwendet

Integrierte Übung 1.1

- Geben Sie Beispiele aus der realen Welt für imperative bzw. deklarative Vorgehensweisen.

SQL

- Die Standardsprache von Datenbanksystemen ist SQL (Structured Query Language); “Intergalactic data speak” [Michael Stonebraker].
- SQL beinhaltet sowohl eine DDL als auch eine DML.
- SQL ist eine deklarative Sprache und wurde von IBM als praktische Anfragesprache zur relationalen Algebra vorgeschlagen.

Schema vs. Instanz/1

- Datenbankschema:
  - Die Beschreibung einer Datenbank.
  - Beinhaltet die Beschreibung der Datenbankstruktur, der Datentypen, und der Integritätsbedingungen auf der Datenbank.
- Das Datenbankschema ändert sich selten.
- Das Datenbankschema wird auch als intensionale Ebene bezeichnet.

Schema vs. Instanz/2

Beispiel: Datenbankschema

Studenten

Name	StudNr	Hauptfach

Vorlesungen

VorlesungsName	VorlesungsNr	ECTS	Institut

Voraussetzungen

VorlesungsNr	VoraussetzungsNr

Module

ModulNr	VorlesungsNr	Semester	Jahr	Dozent

Notenblätter

StudNr	ModulNr	Note

Schema vs. Instanz/3

- Datenbankinstanz:
  - Daten die zu einem gegebenen Zeitpunkt in der Datenbank gespeichert sind
  - auch Datenbankausprägung, Datenbankzustand oder extensionale Ebene genannt
  - Der Begriff "Instanz" wird auch für einzelne Komponenten verwendet (Instanz eines Tupels, Instanz einer Tabelle)
- Gültige Datenbankinstanz: Eine Instanz die sämtliche Strukturen und Integritätsbedingungen erfüllt.
- Eine Datenbankinstanz ändert sich jedesmal wenn die Datenbank geändert wird.

Schema vs. Instanz/4

Beispiel: Datenbankinstanz

Vorlesungen

VorlesungsName	VorlesungsNr	ECTS	Institut
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Databases	CS3360	3	CS

Module

ModulNr	VorlesungsNr	Semester	Jahr	Dozent
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

Voraussetzungen

VorlesungsNr	VoraussetzungsNr
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Notenblätter

StudNr	ModulNr	Note
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

Einordnung der Datenmodelle

- Konzeptionelle Datenmodelle (high-level)
  - Konzepte möglichst nahe an der Benutzersicht
  - keine Datenmainpulationssprache, da nur Schema beschrieben wird, keine Instanzen
  - Beispiele: Entity-Relationship-Modell (ER), Unified Modeling Language (UML)
- Logische Datenmodelle
  - konzentriert sich auf Darstellung der Instanzen
  - geeignet zur Implementierung der Datenbank
  - Beispiele: relationales Modell, objektorientiertes Modell
- Physische Datenmodelle (low-level)
  - Konzepte möglichst nahe an internen Datenstrukturen
  - abhängig von internem Design der Datenbank
  - systemspezifisch, in Handbuch beschrieben

Logische Datenmodelle

- Satzorientierte Datenmodelle: Netzwerkmodell, hierarchisches Modell
  - hauptsächlich historische Bedeutung
  - interessant für Legacy-Systeme (z.B. hierarchisch: IMS von IBM, Netzwerk: UDS von Siemens)
- Relationales Modell:
  - speichert Daten in Tabellen
  - elegantes mathematisches Modell
  - deklarative und imperative Abfragesprachen
- Objektorientiertes und objekt-relationales Modell:
  - Antwort auf Anwendungen mit komplexen Datentypen und Objektorientierung der Programmiersprachen
  - objektorientierte Datenbanken weniger verbreitet, aber Aspekte leben in objekt-relationalen Datenbanken weiter (z.B. PostgreSQL)

Datenbankbenutzer/1

Datenbank Benuzer haben unterschiedliche Aufgaben:

- Abfrage und Änderung des Datenbankinhaltes
- Entwurf der Datenbank
- Entwicklung und Unterhalt von Datenbankanwendungen
- Verwaltung des Datenbanksystems

Wir unterscheiden:

- Endbenutzer
- Datenbankdesigner
- Anwendungsprogrammierer
- Datenbankadministratoren

Datenbankbenutzer/2

- Endbenutzer: Verwenden die Datenbank für Anfragen, Berichte, und Änderungen.
- Endbenutzer können wie folgt kategorisiert werden:
  - **naive Benutzer**: umfasst den Grossteil der Endbenutzer
    - Verwenden genau definierte Funtionen in der Form von vorgefertigten Transaktionen auf der Datenbank
    - Beispiele: Bankomaten, Reservierungssyteme, Webformulare
  - **fortgeschrittene Benutzer**:
    - Analysten, Wissenschaftler und Ingenieure die vertraut mit den Fähigkeiten des Systems sind
    - Schreiben keine Programme, formulieren jedoch Anfragen anhand einer Anfragesprache

Datenbankbenutzer/3

- Anwendungsprogrammierer: Betten die Anfragesprache in eine Programmiersprache ein und stellen Endbenutzern einfach zu bedienende Programme zur Verfügung, welche komplexe Anfragen bewältigen.
  - erstellen von Webanwendungen
  - erstellen von Anwendungssoftware mit Datenbankzugriff
- Datenbankdesigner:
  - Verantwortlich für den Inhalt, die Strukturen, die Integritätsbedingungen, die Funktionen und Transaktionen. Datenbankdesigner müssen mit Endbenutzern kommunizieren und deren Bedürfnisse kennen.
- Datenbankadministratoren:
  - Verantwortlich für die Autorisierung des Datenbankzugriffs, der Koordination und Überwachung der Benutzung, der Beschaffung von Soft- und Hardware, Backup, Kontrolle der Effizienz der Operationen

Die ANSI/SPARC Drei-Ebenen Architektur/1

- Die ANSI/SPARC Architektur wurde vorgeschlagen um folgende Charakteristiken einer Datenbank zu unterstützen:
  - Unterschiedliche Sichten auf die Daten
  - Datenunabhängigkeit
- Definiert ein Datenbankschema auf drei Ebenen:
  - **Physische Ebene**: beschreibt die physischen Speicherstrukturen (z.B. Tabellen) und Zugriffspfade (z.B. Indizes).
    - verwendet typischerweise ein physisches Datenmodell
  - **Logische Ebene**: beschreibt die Strukturen und Integritätsbedingungen für die gesamte Datenbank und deren Benutzer
    - verwendet ein konzeptionelles oder logisches Datenmodell
  - **Externe Sicht**: beschreibt unterschiedliche Sichten (views) auf die Datenbank.
    - verwendet oft das gleiche Datenmodell wie die logische Ebene

Die ANSI/SPARC Drei-Ebenen Architektur/2

- Abbildungen zwischen den verschiedenen Ebenen sind notwendig um Anfragen und Daten transformieren zu können.
  - Anwendungen beziehen sich auf die externe Sicht und werden durch das Datenbanksystem auf die logische und physische Ebene abgebildet um ausgewertet zu werden.
  - Daten die aus der physischen/logischen Ebene kommen werden umformatiert, damit sie der externen Sicht des Benutzers entsprechen.

Die ANSI/SPARC Drei-Ebenen Architektur/3

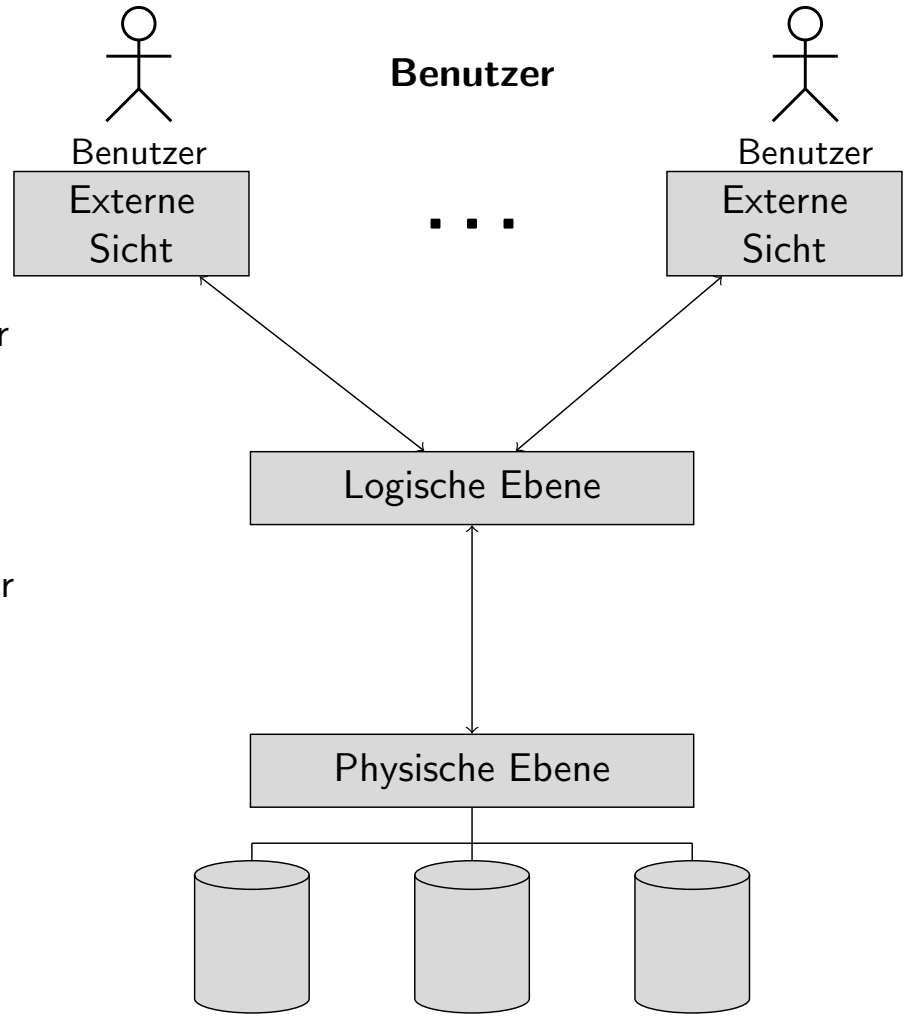
Externe Sicht

Abbildung zwischen externer Sicht und logischer Ebene

Logische Ebene

Abbildung zwischen logischer und physischer Ebene

Physische Ebene



Gespeicherte Datenbank

Datenunabhängigkeit

- Logische Datenunabhängigkeit:
  - Die Möglichkeit das logische Schema zu ändern ohne die externen Sichten und zugehörigen Anwendungen ändern zu müssen.
  - Beispiel: Objekte und deren Eigenschaften umbenennen, neue Eigenschaften hinzufügen
- Physische Datenunahängigkeit:
  - Die Möglichkeit die physische Ebene zu ändern, ohne die logische Ebene ändern zu müssen.
  - Beispiel: Speicherstruktur ändern oder neue Indices erstellen um die Effizienz zu verbessern.
- Vorteile der Datenunabhängigkeit:
  - nach der Änderung einer tieferen Ebene müssen nur die Beziehungen zwischen dieser und der darüberliegenden Ebene nachgeführt werden
  - die weiter darüberliegenden Ebenen werden nicht geändert
  - Anwendungsprogramme müssen nicht geändert werden, da sie auf die oberste Ebene zugreifen

Zusammenfassung

- Datenmodelle: Modellierung der Miniwelt
  - DDL: Data Description Language
  - DML: Data Modification Language
- Anfragesprachen (Teil von DML):
  - imperativ / deklarativ
  - reine Sprachen / praktische Sprachen
- SQL ist Standardsprache: DDL und DML
- Drei-Ebenen Architektur
  - externe, logische und interne Ebene