Grundlagen der Datenbanksysteme I

WiSe 2018/2019

Prof. Dott. Ing. Roberto V. Zicari

Goethe Universität Frankfurt am Main

Fachbereich 12 - Informatik und Mathematik

Goethe Universität
Frankfurt am Main

Institut für Informatik
im Fachbereich Informatik und Mathematik (12)

Grundlagen der Datenbanksysteme I

Veranstalter:

Prof. Dott. Ing. Roberto V. Zicari

Zeit und Ort:

Mi, 10 bis 12 Uhr im Hörsaal HV,
Do, 14 bis 16 Uhr im Hörsaal HVI,

Übung:

Fr, 14 bis 16 Uhr im Hörsaal HV

Achtung: Auf etwaige Änderungen wird jeweils in der Vorlesung und auf den Web Seiten hingewiesen.
http://www.bigdata.uni-frankfurt.de/db1_2018/

Mitarbeiter:

Dr. Karsten Tolle

Todor Ivanov

(Robert-Mayer-Str. 10, 5. Stock)

DBIS Homepage:

http://www.bigdata.uni-frankfurt.de/
http://www.dbis.cs.uni-frankfurt.de

E-Mail:

db@dbis.cs.uni-frankfurt.de

Scheinerwerb durch schriftliche Prüfung (es wird eine zweite Klausur/Nachklausur im März/April geben)

Bearbeitungszeit 180 Minuten

Termin Erstklausur (noch unter Vorbehalt):

Datum: Fr. 15.02.

Raum: Hörsaaltrakt Bockenheim - H VI

Zeit: 13:00 – 16:00

Bonuspunkte:

Es können Bonuspunkte durch Übungsabgaben vor der Klausur gesammelt werden bis zu 10% der Klausur. Mehr dazu am Freitag in der Übung.

Termin Zweitklausur (noch unter Vorbehalt):

Datum: **03**.04.2019

Raum: Hörsaaltrakt Bockenheim - H VI

Zeit: 10:00 – 13:00

Vorwissen:

Modellierung/Design und ER-Modelle

Die Lerninhalte von PRG-2 siehe (http://www-stud.rbi.informatik.uni-frankfurt.de/~prg2/)

Frühere Folien aus alten DB1 Veranstaltungen auf der Webseite!

- Konzeptionelles Design
- Entity-Relationship Modell
- Methoden des DB-Designs 1
- Methoden des DB-Designs 2

Literatur

Theorie:

J. Ullman, *‘Principles of Database Systems’*, 2d ed., Computer Science Press, 1982
ISBN 0-7167-8069-0

Transaktionen:

Bernstein, Hadzilacos, Goodman, *‘Concurrency Control and Recovery in Database Systems’*, Addison-Wesley, 1987
ISBN 0-201-10715-5

Online frei erhältlich unter:

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/05/ccontrol.zip

Weitere empfehlenswerte Literatur

A. Kemper, A. Eickler: *'Datenbanksysteme - Eine Einführung*', 8. Auflage Oldenburg Verlag, 2011
ISBN 978-3-486-598334-6

Ramez Elmasri und Shamkant B. Navathe, 'Grundlagen von Datenbanksystemen', Pearson Studium, 2009

Weitere Literaturempfehlungen befinden sich auf der Web Seite der Veranstaltung.

Inhalt

- Einführung
- Relationales Datenmodell
- Relationale Algebra
- SQL
- Logische Optimierung
- Normalformen
- Transaktionen

Einführung

Kernaufgaben von Datenbanksystemen ist die Speicherung und Verwaltung von großen Datenbeständen.

Wichtig für diese Kernaufgabe sind:

- **Angebotene Benutzerschnittstellen**
  Wie greife ich auf die gespeicherten Daten zu?
  Wie kann ich gespeicherte Daten ändern?

- **Organisation der Daten**
  Wie effizient ist die Speicherorganisation der Daten?

Beispiel 1: Flug-Reservierungssystem

Daten die gespeichert werden müssen:

- Reservierungen einzelner Kunden auf einzelnen Flügen, inklusive Sitzplatzvorlieben und Essenswünsche
- Informationen über die Flüge (Start- und Zielflughäfen, Abflugzeiten, Ankunftszeiten, Flugzeugtyp, Fluggesellschaft, ...)
- Informationen über Ticketpreise, Voraussetzungen und Verfügbarkeit

Anforderungen (Flug-Reservierungssystem)

- Anfragen nach Flügen zu gewissen Zeiten zwischen gegebenen Flughäfen
- Die Buchung einzelner Flüge, festlegen von Sitzplätzen und Vormerkung von Essenswünschen
- Gleichzeitiger Zugang und Bearbeitung der aktuellen Daten durch verschiedene Reisebüros

Beispiel 2: Bank

Eine Bank benötigt Daten über:

- **Die Kunden**
  (Namen, Anschrift, Alter, ...)
- **Das Personal**
  (Name, Anschrift, Tätigkeit, Gehalt, ...)
- **Die Konten**
  (Kontonummer, Buchungen, Stand, ...)
- **Die Zuordnung** von Kunden und Konten

... und vieles mehr!

Verschiedene Anwendungen müssen auf die Bank-Daten zugreifen und sie ändern können:

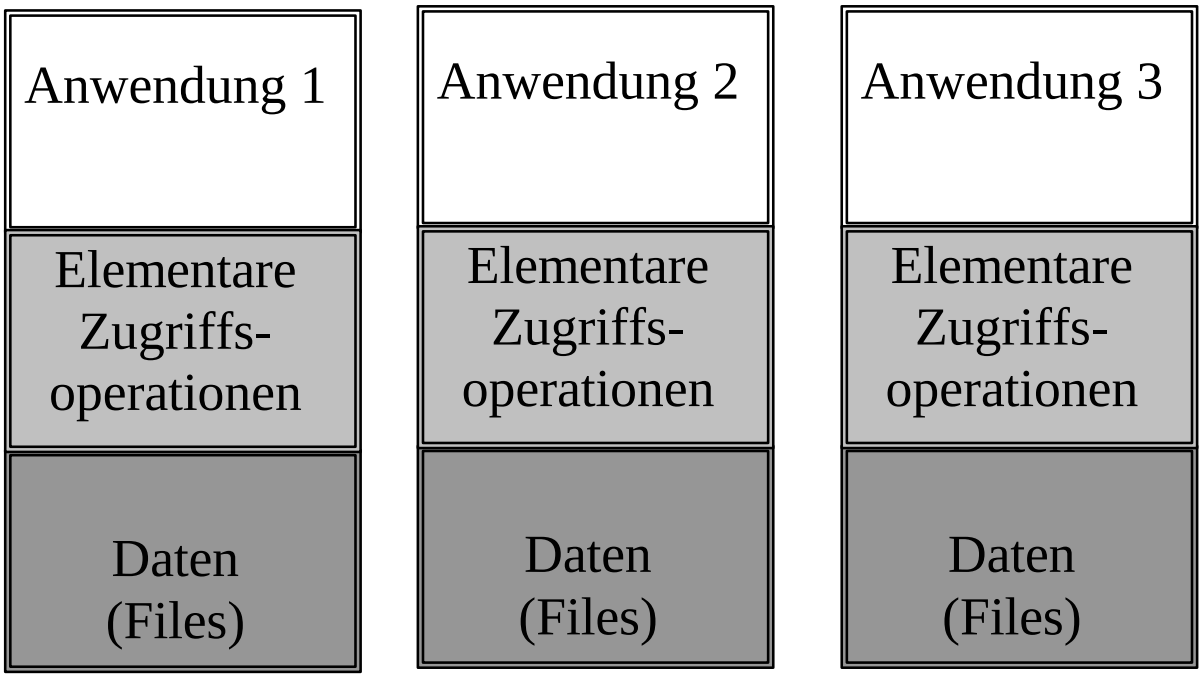
1. Ein Geldautomat muss von Konten Abbuchungen vornehmen.
2. Ein Überweisungsterminal muss Abbuchungen und auch Zugänge verarbeiten können.
3. Ein Bankangestellter muss neue Kunden und Konten anlegen können.
4. Ein Kontoauszugsdrucker muss den Kontostand und die Buchungen auslesen können.

Anforderungen an einen Geldautomaten:

Wird an einem Automaten Geld ausbezahlt, muss dies unverzüglich vermerkt werden. Auch nach einem sofortigen Stromausfall darf eine solche Information nicht verloren gehen. Auf der anderen Seite darf diese Information nur dann vermerkt werden, wenn das Geld auch wirklich ausbezahlt wurde.

Warum ein Datenbanksystem?

Als es noch kein Datenbanksystem gab wurden die Daten für jede Anwendung in separaten Files permanent gespeichert (Einzellösung).



Nachteile und Probleme dieser Einzellösung (1)

- **Redundanz und Inkonsistenz**
  Daten müssen mehrfach gespeichert werden.

Beispiel:
Die Adresse und Telefonnummer eines Kunden ist in einem File, welches zum Speichern von Kontobewegungen dient, als auch in einem (anderen) File, welches zur Überprüfung von Kontobewegungen dient, gespeichert.

Diese Daten sind also mehrfach gespeichert (Redundanz), was einen höheren Bedarf an Speicher bedeutet. Zusätzlich birgt dies immer die Gefahr der Inkonsistenz, falls Daten nur an einer Stelle geändert werden.

Nachteile und Probleme dieser Einzellösung (2)

- **Beschränkte Zugriffsmöglichkeiten**
  Keine Ad-hoc-Abfrage, keine Abfragesprache.

Beispiel:

Angenommen ein Bankangestellter benötigt die Namen aller Kunden, die in Frankfurt wohnen. Der Bankangestellte kann nun entweder manuell versuchen die Namen aus dem entsprechenden File herauszusuchen oder einen Programmierer beauftragen ein entsprechendes Programm zu schreiben.

Sollte sich die Fragestellung nur leicht ändern, z.B. nur die Namen mit einem Kontostand über 10.000 €, besteht das gleich Problem jedoch wieder.

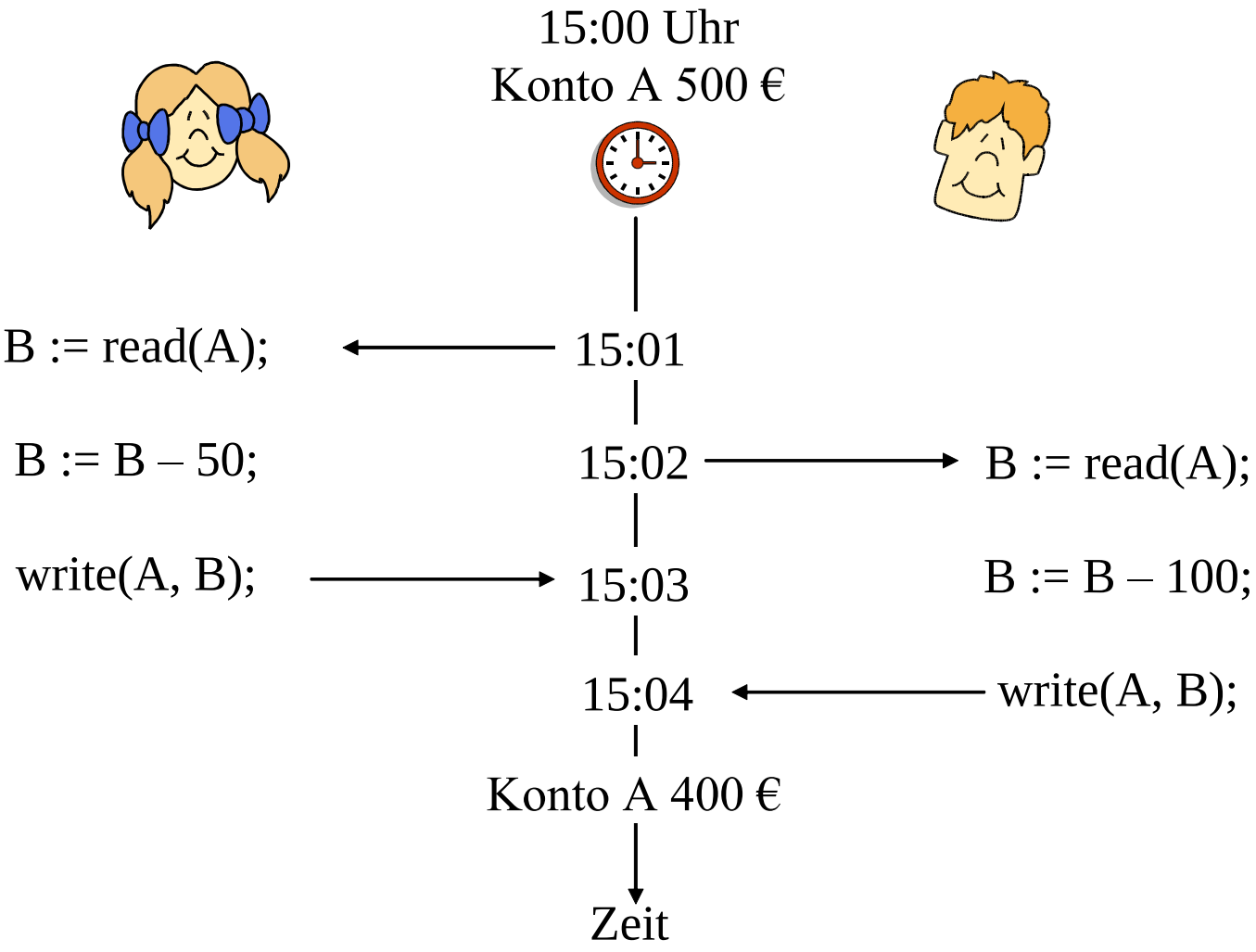
Nachteile und Probleme dieser Einzellösung (3)

- **Probleme des Mehrbenutzerbetriebs (Concurrency)**
  Keine Überwachung gleichzeitiger Zugriffe.

Beispiel:

Um die Geschwindigkeit zu steigern, erlauben viele Systeme, insbesondere Banksysteme mit vielen Benutzern, den Mehrbenutzerbetrieb. Mehrere Benutzer arbeiten also gleichzeitig auf und mit den Daten.

Falls gleichzeitig zwei Benutzer von dem gleichen Konto (A mit 500 €) abheben möchten (z.B. 100 € und 50 €), so kann es sein, dass der Kontostand hinterher nicht stimmt. In diesem Fall könnte Konto A hinterher 400 € oder 450 € aufweisen, anstelle der korrekten 350 €.


15:00 Uhr
Konto A 500 €
B := read(A);
15:01
B := B – 50;
15:02
B := read(A);
write(A, B);
15:03
B := B – 100;
15:04
write(A, B);
Konto A 400 €
Zeit

Nachteile und Probleme dieser Einzellösung (4)

- **Integritätsverletzungen**
  Erzwingen von Integritätsbedingungen ist schwierig.

Beispiel:

Bei Konten möchte man den Kreditrahmen gerne begrenzen. Eine Bedingung wie ‚*Kontostand darf nicht negativ werden*' ist mit Files nur sehr schwierig bzw. aufwendig zu realisieren.

Nachteile und Probleme dieser Einzellösung (5)

- **Sicherheitsprobleme**
  Nicht alle Benutzer sollten die gleichen Zugriffsmöglichkeiten auf die gespeicherten Daten haben.

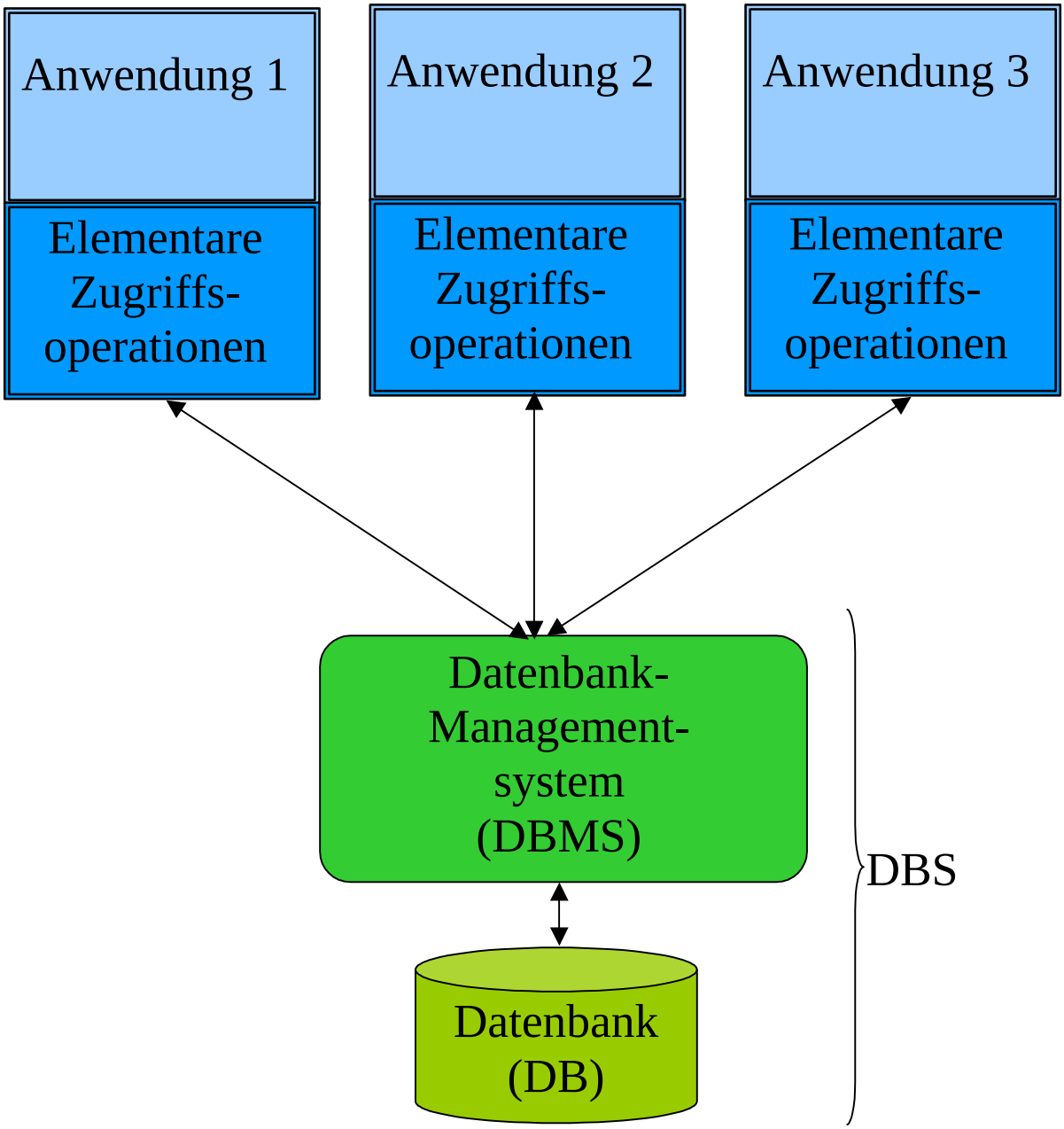
Beispiel:

Ein Bankangestellter zuständig für die Gehaltsabrechnungen sollte keine Einsicht in die Kundenkonten und Kundendaten haben. Bei der Einbindung neuer Anwendungen müssen diese Sicherheitsüberprüfungen jeweils wieder überprüft und entsprechend implementiert werden.

Datenbanksysteme

Datenbanksysteme (DBS) wurden entwickelt (seit ca. 1975), um diese Probleme zu überwinden. Ein Datenbanksystem sollte daher folgende Funktionalitäten bieten:

- **Dauerhafte Speicherung** von großen Datenbeständen
- **Zugriffsmöglichkeit** durch verschiedene Benutzer und Anwendungen, **ohne inkonsistente** Zustände zu erhalten
- Bereitstellung einer **Anfragesprache** zum einfachen Umgang mit der Datenbank
- Überwachung von **Integritätsbedingungen**
- **Sicherheit** gegenüber Hard- und Software ausfällen (backup and recovery)
- Sicherheit gegenüber nicht autorisierten **Datenzugriffen (views)**
- ... und natürlich **effizient**, möglichst **schnell** unter Benutzung weniger Ressourcen



Kernidee: eine zentrale Datenverwaltung / Datenhaltung

Folgerung

Diese Kernidee des DBS bedarf einer neuen Vorgehensweise:

- Erst Design der benötigten Datenstrukturen
- Danach können die Anwendungen auf den geteilten Daten (shared data) entwickelt und angewendet werden

---

Gegenüber vorher (Einzellösungen):

- Entwicklung von Anwendung 1 mit benötigten Daten
- Entwicklung von Anwendung 2 mit benötigten Daten

  ...

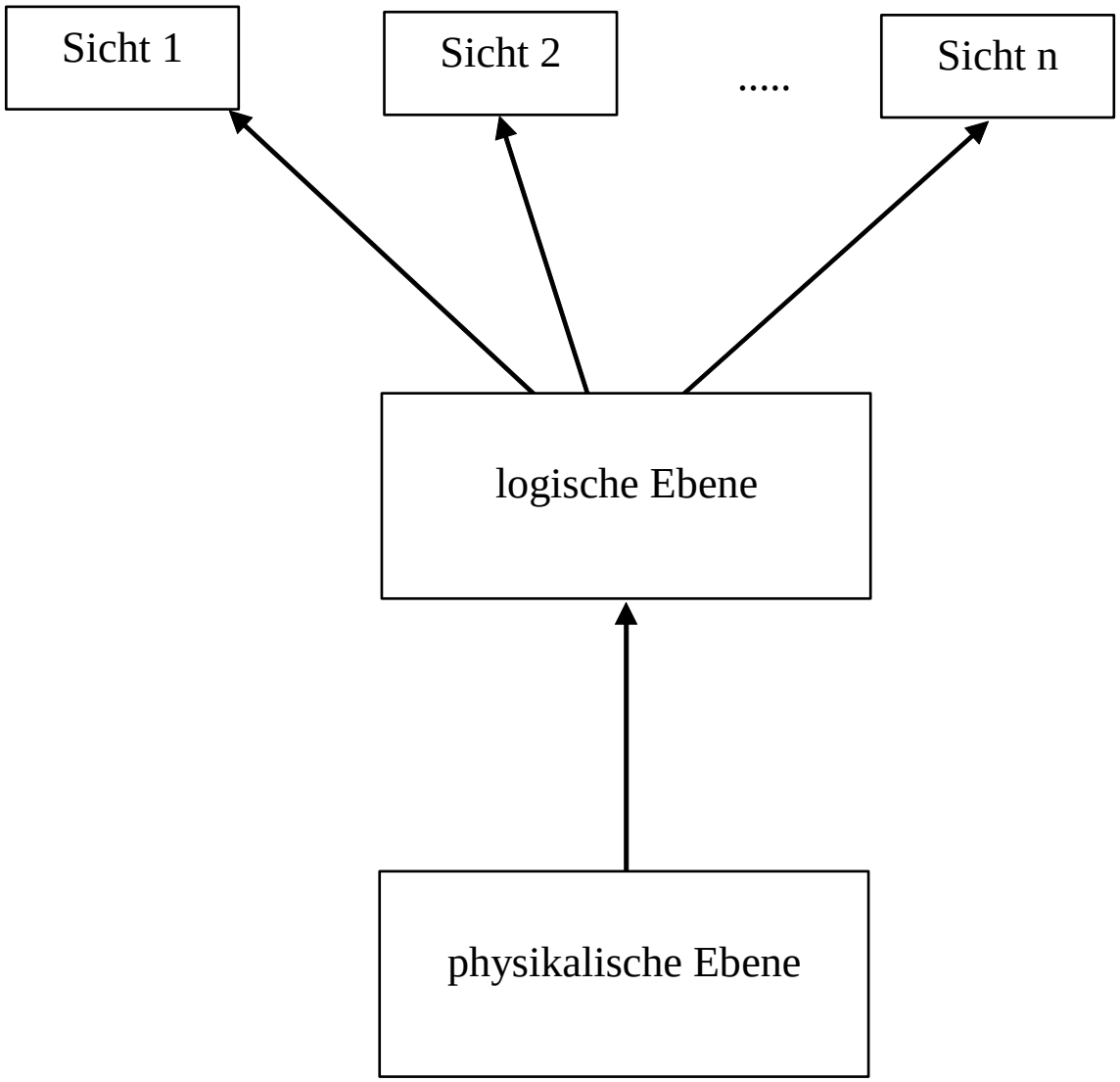
Datenabstraktion (Data Abstraction)

Sehr grob unterscheidet man drei Abstraktionsebenen in einem Datenbanksystem:

- Die **physische Ebene** (physical level)
  Auf dieser Ebene ist beschrieben, wie die Daten auf dem Sekundärspeicher abgelegt sind.

- Die **logische/konzeptionelle Ebene** (conceptual level)
  Auf dieser Ebene wird mittels eines Datenbankschemas festgelegt, welche Daten abgespeichert sind.

- Die **Sichten** (view level)
  In den Sichten werden Teilmengen der Daten dargestellt. Die Sichten sind auf die Bedürfnisse der Benutzer zugeschnitten.



Die drei Abstraktionsebenen eines Datenbanksystems.

Die verschiedenen Ebenen können analog zu den Konzepten in Programmiersprachen verdeutlicht werden. Ein Kunde könnte in *C* oder *Modula 2* z.B. wie folgt deklariert werden:

```
TYPE Kunde = RECORD
    Name: String;
    Strasse: String;
    Ort: String;
END;
```

Der Record Kunde hat hier drei Felder. Jedem Feld werden ein Name und ein Type zugewiesen.

Auf der ***physikalischen Ebene*** kann die Speicherung eines Kunden als Block (von Words oder Bytes) im Speicher beschrieben werden.

Die ***konzeptionelle Ebene*** beschreibt die Type Deklaration wie oben beschrieben und die Beziehungen zwischen einzelnen Typen.

Die einzelnen ***Sichten*** schließlich zeigen nur Teile der vorhandenen Daten. Ein Kassierer am Bankschalter zum Beispiel sieht nur die für ihn relevanten Daten und nicht auch Gehaltsinformationen von Kollegen.

Datenunabhängigkeit (Data Independence)

Die Möglichkeit eine dieser Ebenen zu ändern, ohne dabei Auswirkungen auf die anderen Ebenen zu haben wird als *Datenunabhängigkeit* bezeichnet. Es wird dabei wie folgt unterschieden:

- ***Physische Datenunabhängigkeit*** = Stabilität der Benutzerschnittstelle gegen Änderungen der Datenorganisation (z.B. Datendarstellung, Feldlänge, Satzformate, Zugriffspfade), d.h. nach Änderungen auf der physikalischen Ebene müssen die Anwendungen nicht umgeschrieben werden.

- ***Logische Datenunabhängigkeit*** = Stabilität gegen Änderungen in der logischen Ebene (z.B. die Einführung neuer oder abgeleiteter Merkmale wie Ausbildung bzw. Alter für Kunden)

Bemerkungen:

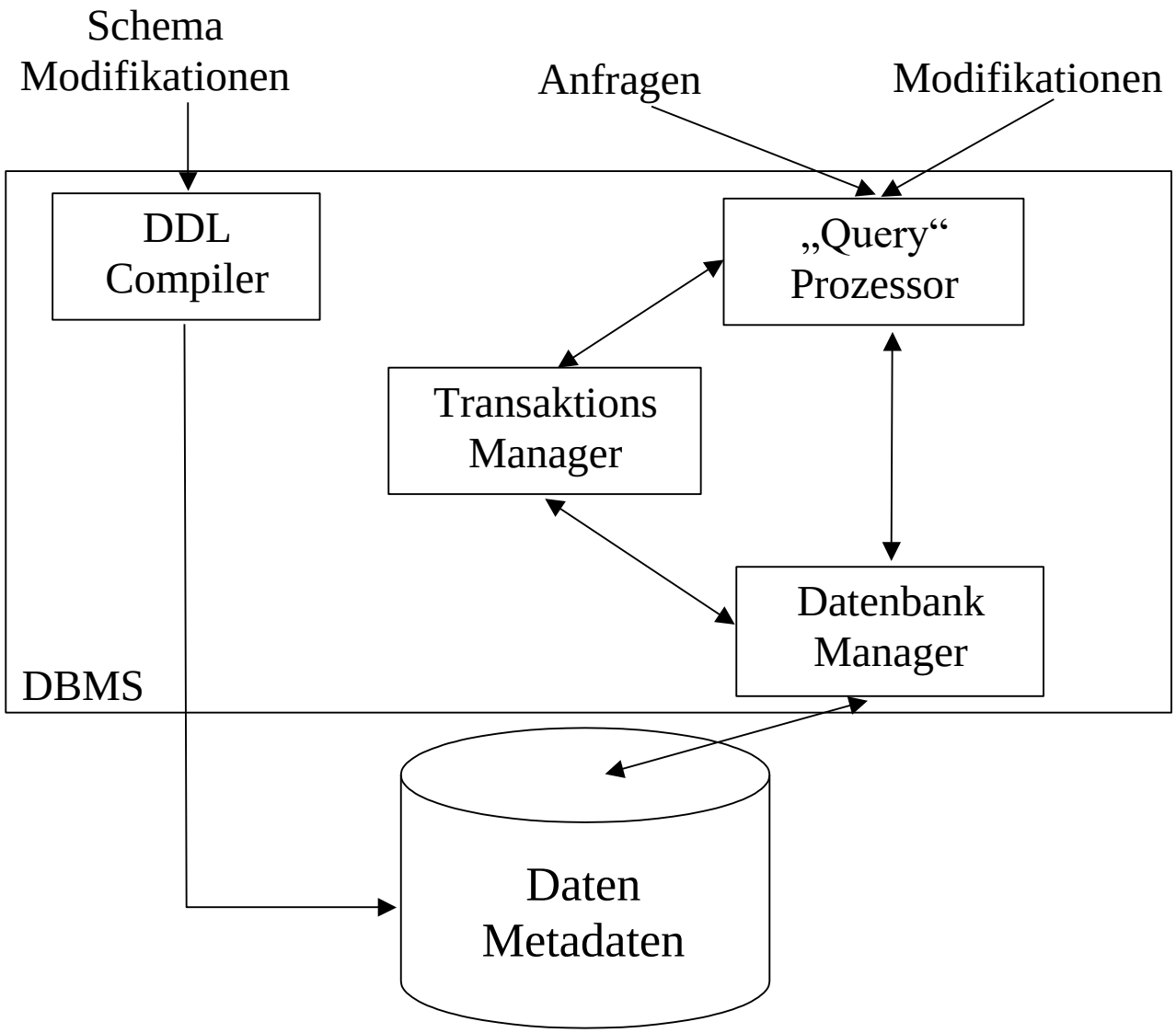
Das Erreichen der logischen Datenunabhängigkeit ist deutlich schwerer, da Anwendungen stark auf den logischen Strukturen der benötigten Daten aufbauen.

Die Idee der Datenunabhängigkeit ist vergleichbar mit der Idee der abstrakten Datentypen in modernen Programmiersprachen. Mit beiden Ideen soll ein *Information Hiding* erreicht werden, damit Anwender sich mehr auf das eigentliche Problem konzentrieren können, anstatt sich mit den genauen Implementierungsdetails zu beschäftigen.

Die Architektur des DBMS

Ein Datenbanksystem ist unterteilt in Module, die einzelne Aufgaben des Systems übernehmen. Zu diesen Modulen gehören:

- **Datenbank Manager** – Interface zwischen Low-Level Daten und den Programmen und Anfragen der Benutzer
- **Anfrage Prozessor –** übersetzt die Ausdrücke der Anfragesprache in Low-Level Anweisungen, die der Datenbank Manager versteht; führt Optimierung von Anfragen durch
- **Transaktions-Manager** – überwacht gleichzeitigen Zugriff auf Daten
- **Data Definition Language (DDL) Compiler –** übersetzt die DDL Ausdrücke in entsprechende Metadaten („Daten über Daten“)


Schema
Modifikationen
Anfragen
Modifikationen
DDL
Compiler
„Query“
Prozessor
Transaktions
Manager
Datenbank
Manager
DBMS
Daten
Metadaten

Data Definition Language (DDL)

Ein Datenbankschema wird mit Hilfe einer speziellen Definitionssprache spezifiziert, der *Data Definition Language (DDL)*.

Die Schemainformationen nach der Übersetzung von DDL-Aussagen werden in einem besonderen File, dem ***Data Dictionary*** gespeichert. Die Daten im Data Dictionary beschreiben also nicht einen Kunden und dessen Eigenschaften, sondern die Struktur welche Informationen zu einem Kunden gespeichert werden. Diese ‚Daten über Daten' werden als ***Metadaten*** bezeichnet.

Data Manipulation Language (DML)

Eine *Data Manipulation Language (DML)* ist eine Sprache, die es Benutzern erlaubt, auf Daten zuzugreifen (Anfragen) und diese zu modifizieren (Einfügen, Löschen, Ändern).

Der Datenbank Manager (database manager)

Der Datenbank Manager ist ein Modul, welches die Schnittstellen zwischen den Low-Level Daten auf dem Sekundärspeicher und den Programmen und Abfragen von Benutzern zur Verfügung stellt. Der Datenbank Manager muss dabei folgende Aufgaben erfüllen:

- Interaktion mit dem File manager des OS
- Zusicherung von Integritätsbedingungen
- Sicherheitskontrollen
- Backup und Recovery

Transaktions-Manager

- Nebenläufigkeitskontrolle (concurrency control)

Datenbank-Administrator

Einer der Hauptgründe für eine Datenbank, ist es, eine zentrale Kontrolle über die Daten und Programme zu bekommen.

Die Person mit diesen Kontrollmöglichkeiten nennt man Datenbank-Administrator. Seine Aufgaben sind:

- Definieren von Schemata
- Festlegen von Speicherstrukturen und Zugriffsmethoden
- Vergabe von Rechten
- Spezifikation von Integritätsbedingungen

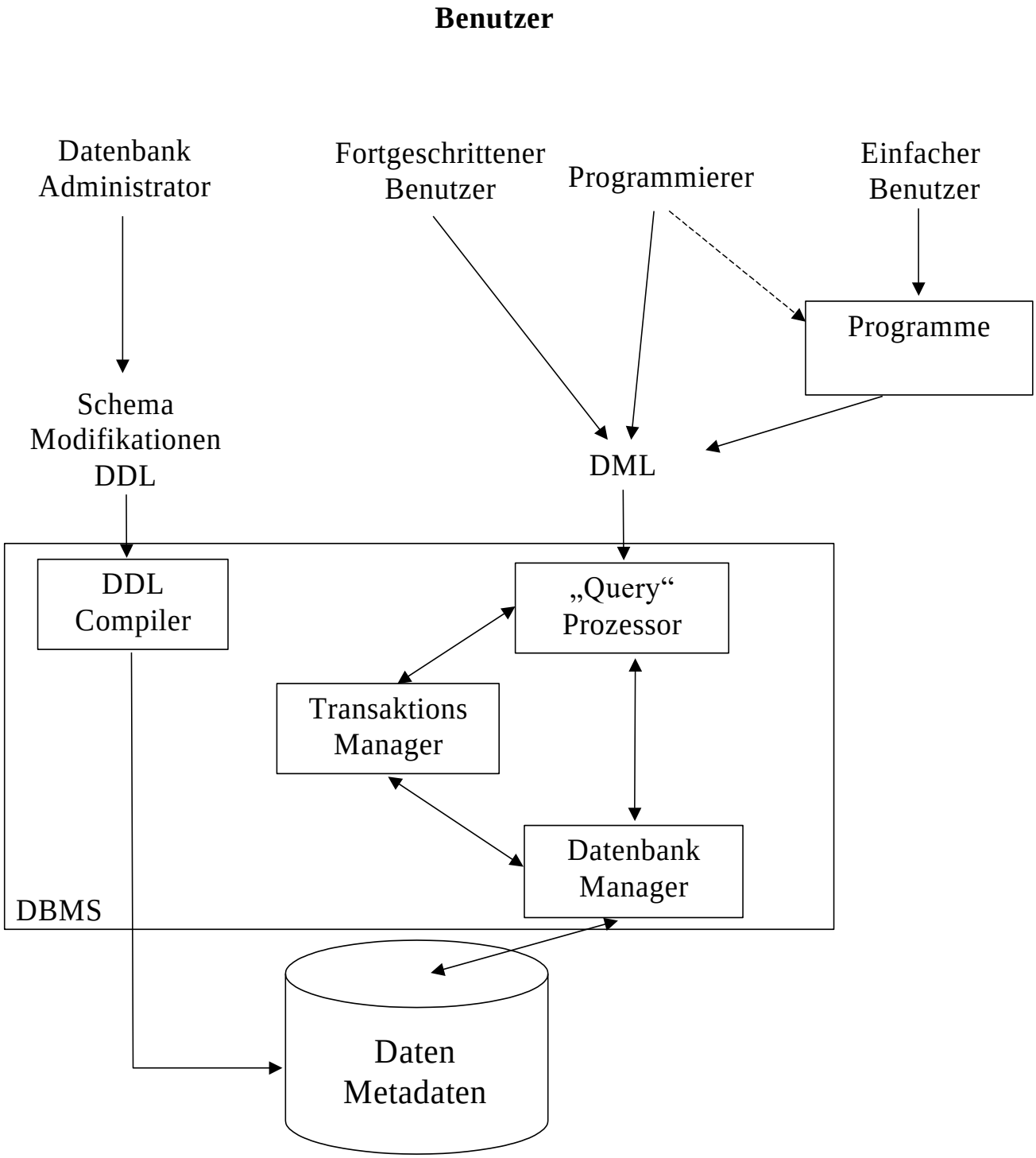
Datenbank Benutzer

Ziel einer Datenbank ist es, die Informationen den Benutzern zur Verfügung zu stellen. Es werden drei Typen von Benutzern unterschieden:

- **Programmierer**
  Computer-Experten, die mit der Datenbank durch DML-Aufrufe eingebettet in Programmiersprachen kommunizieren (Anwendungsprogrammierer).

- **Fortgeschrittene Benutzer**
  Benutzen keine fertigen Programme, sondern erstellen selbst ihre Anfragen in einer Anfragesprache.

- **Einfacher Benutzer**
  Interagiert mit der Datenbank durch fertige Programme.


Benutzer
Datenbank Administrator
Fortgeschrittener Benutzer
Programmierer
Einfacher Benutzer
Programme
Schema Modifikationen DDL
DML
DDL Compiler
„Query“ Prozessor
Transaktions Manager
Datenbank Manager
DBMS
Daten Metadaten