Kapitel 2

Grundlagen von Datenbanksystemen

Inhalt

2.1 Paradigmen der Wissenschaft

Wissenschaft war **empirisch** (vor 1000 Jahren)

- Empirie beobachtet die Natur, um Daten zu erheben, die es erlauben, Annahmen über die Welt zu bestätigen oder zu widerlegen

Wissenschaft war **theoretisch** (die letzten Jahrhunderte)

- Theorie stellt überprüfbare Modelle der Realität auf, die die Zusammenhänge natürlicher Phänomene beschreiben

Wissenschaft war **computational** (die letzten Dekaden)

- In der Simulation werden Experimente an einem Modell durchgeführt, um Erkenntnisse über das reale System zu gewinnen

Wissenschaft ist **explorativ** (Heute: **eScience, Data Science**)

- Vereinigt Theorie, Experiment und Simulation
- Daten erfasst durch Instrumente oder generiert von Simulatoren
- Bearbeitet durch Software
- Information/Wissen administriert am Computer
- Wissenschaftler analysieren Modelle in der Datenbank mit Werkzeugen der Informatik, Statistik, Computational Intelligence, etc.

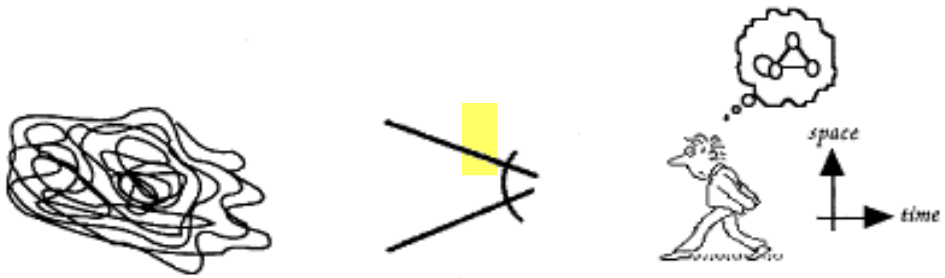
- Siehe *The Fourth Paradigm: Data-Intensive Scientific Discovery*, http://research.microsoft.com/en-us/collaboration/fourthparadigm

Modell

Ein Modell stellt einfach ausgedrückt eine **Abstraktion der Realität** dar, bei der auf die **essentiellen Charakteristiken** fokussiert und auf **nicht relevanten Eigenschaften** verzichtet wird

„Ein Modell ist ein System, das als Repräsentant eines komplizierten Originals auf Grund mit diesem gemeinsamer, für eine bestimmte Aufgabe wesentlicher, Eigenschaften von einem dritten System benutzt, ausgewählt oder geschaffen wird, um letzterem die Erfassung oder Beherrschung des Originals zu ermöglichen oder zu erleichtern, beziehungsweise um es zu ersetzen.“ (Klaus Dieter Wüsteneck, Philosoph).

In der Wissenschaft ist ein Modell eine **physikalische**, **mathematische** oder **logische Darstellung** eines Systems von Objekten, Phänomenen oder Prozessen.



Modellierungsarten in der Informatik

Bei der **datenorientierten Modellierung** werden die Objekte der Realität mit allen relevanten *Eigenschaften* und relevanten *Beziehungen* zwischen ihnen bestimmt

Die **prozessorientierte Modellierung** fokussiert auf die *dynamischen Abläufe* innerhalb der zu modellierenden Realität

In der **objektorientierten Modellierung** werden die Elemente der datenorientierten und prozessorientierten Modellierung zusammengefasst, um die zu verarbeitenden Daten anhand ihrer *Eigenschaften* und der möglichen *Operationen* zu beschreiben

Ziel der **semantischen Modellierung** ist über die rein syntaktische und strukturelle Beschreibung hinaus die *Bedeutung* der zu modellierenden Objekte zu erhalten und über geeignete Werkzeuge und Verfahren in maschinell verarbeitbare Daten zu verwandeln

2.2 Datenorientierte Modellierung

Grundlegende Struktur der in der Datenbank gespeicherten Daten ist das so genannte **Datenmodell**

Repräsentiert durch eine Umgebung von konzeptuellen Werkzeugen, die folgendes beschreiben:

- Daten,
- Beziehungen zwischen Daten,
- Bedeutung von Daten,
- Integritätsbedingungen und
- Operatoren zur Manipulation der Daten

Ziel ist, einen **Ausschnitt der Realität**, „the Universe of Discourse“ (oder Problembereich) mit den konzeptuellen Werkzeugen des Datenmodells zu repräsentieren

Datenmodelle

Datenmodell ist ein Notationswerkzeug für die Beschreibung von

- Daten
- Beziehung zwischen Daten
- Bedeutung von Daten
- Bedingungen auf Daten

Entity-Relationship Modell

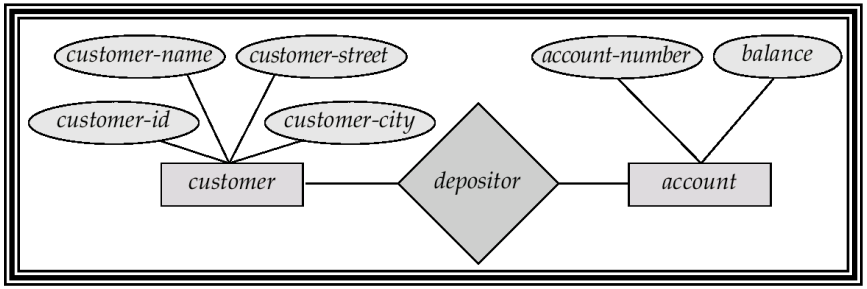
- Fokus semantische Modellierung
- ER, EER, …

Relationales Modell

- Fokus datenorientiert Modellierung

Andere Modelle

- Objektorientiertes Modell
- Semistrukturiertes Modell
- Alte Modelle: Netzwerk und hierarchisches Modell



ER Modell

customer-id	customer-name	customer-street	customer-city
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	account-number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Relationales Modell

Abstraktion als Element der Modellierung

Vorgang der Modellierung mit (mindestens) drei generischen Schritten

1. **Selektion** auf die relevanten Daten
2. **Strukturierung** und
3. **Benennung** der Daten

Zentrum aller Überlegungen das Prinzip der **Abstraktion**

„The principle of ignoring those aspects of a subject that are not relevant to the current purpose in order to concentrate more fully on those that are“
(Dictionary of Computing, Oxford University Press, 1986)

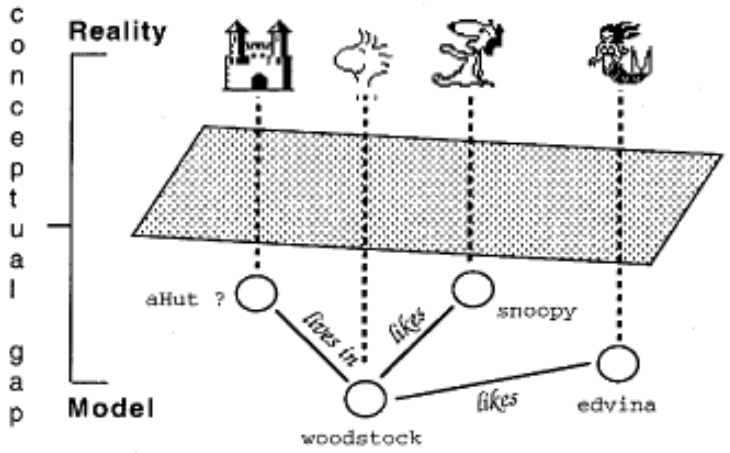
Abstraktion ist der Prozess auf die **essentiellen Charakteristiken** eines Objektes der Realität zu fokussieren, sodass es sich von allen anderen Objekten unterscheidet

Pragmatische Gründe für die Abstraktion

Vereinfachte Nachbildung
der Realität und Situation der Realität,

Verringerte Komplexität,
dadurch weniger Aufwand für die Behandlung realer Sachverhalte und

Kommunikationsbasis
zwischen Benutzer und Entwickler.



2.3 Drei-Ebenen-Architektur

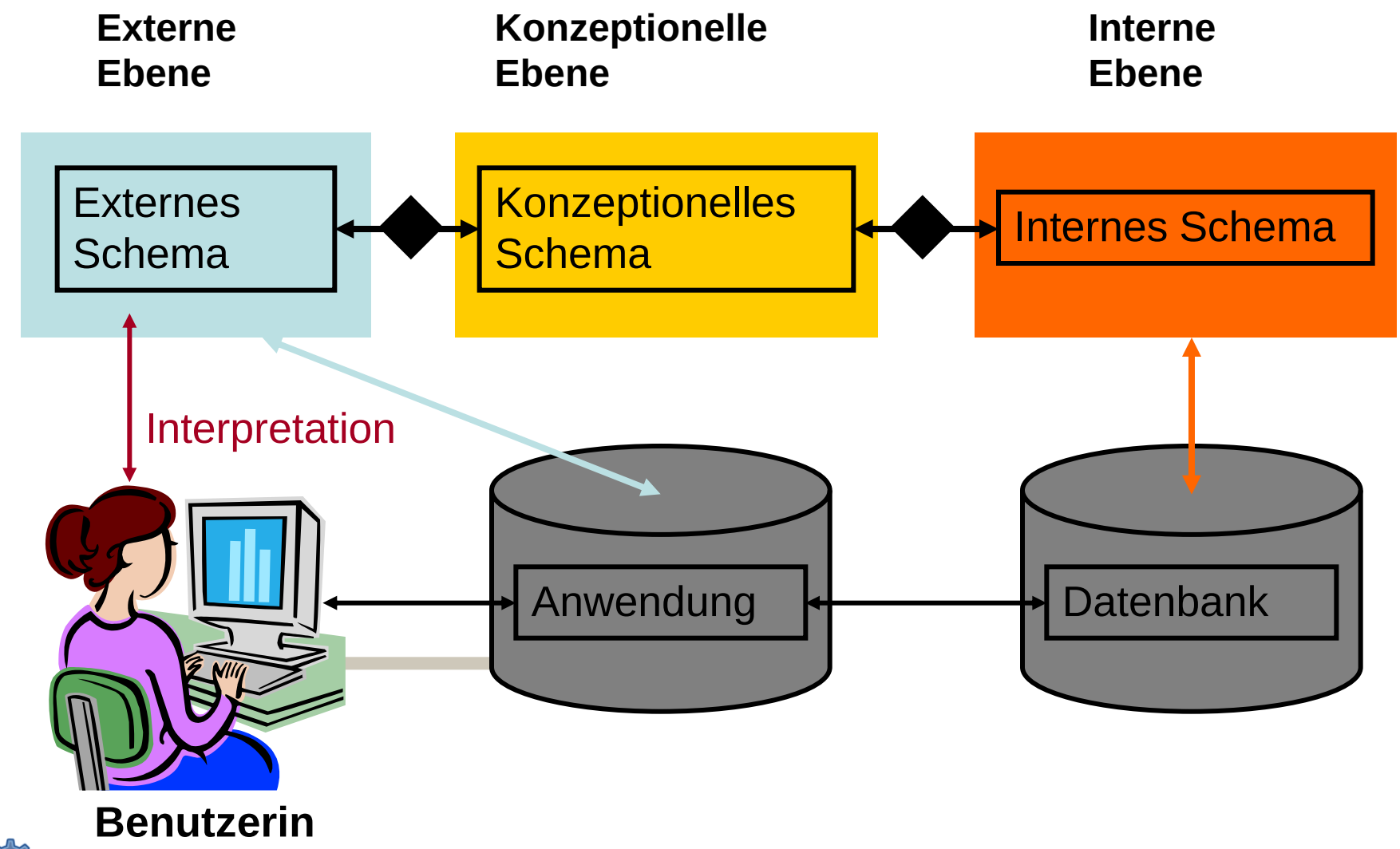
Ein Hauptzweck von DBMS ist, dem Benutzer eine **abstrakte Betrachtungsweise der Daten** zu liefern

Das System versteckt gewisse Teile der Daten in Datenbanksystemen

Bei Datenbanksystemen hat sich hierzu eine **Drei-Ebenen-Architektur** durchgesetzt:

- Die **externe Ebene** zeigt die Beschreibung der Daten und ihrer Beziehungen aus Sicht der einzelnen **Anwender** bzw. Anwendungen (externe Schema).
- Die **konzeptuelle Ebene** beschreibt alle relevanten Objekte und ihre Beziehungen im Sinne einer logischen **Gesamtbeschreibung** (konzeptuelles Schema).
- In der **internen Ebene** wird die **physische Datenorganisation** mit Hilfe einer Speicherbeschreibungs-sprache festgelegt (physisches Schema).

Verwendung einer Datenbank



Instanzen und Schemas

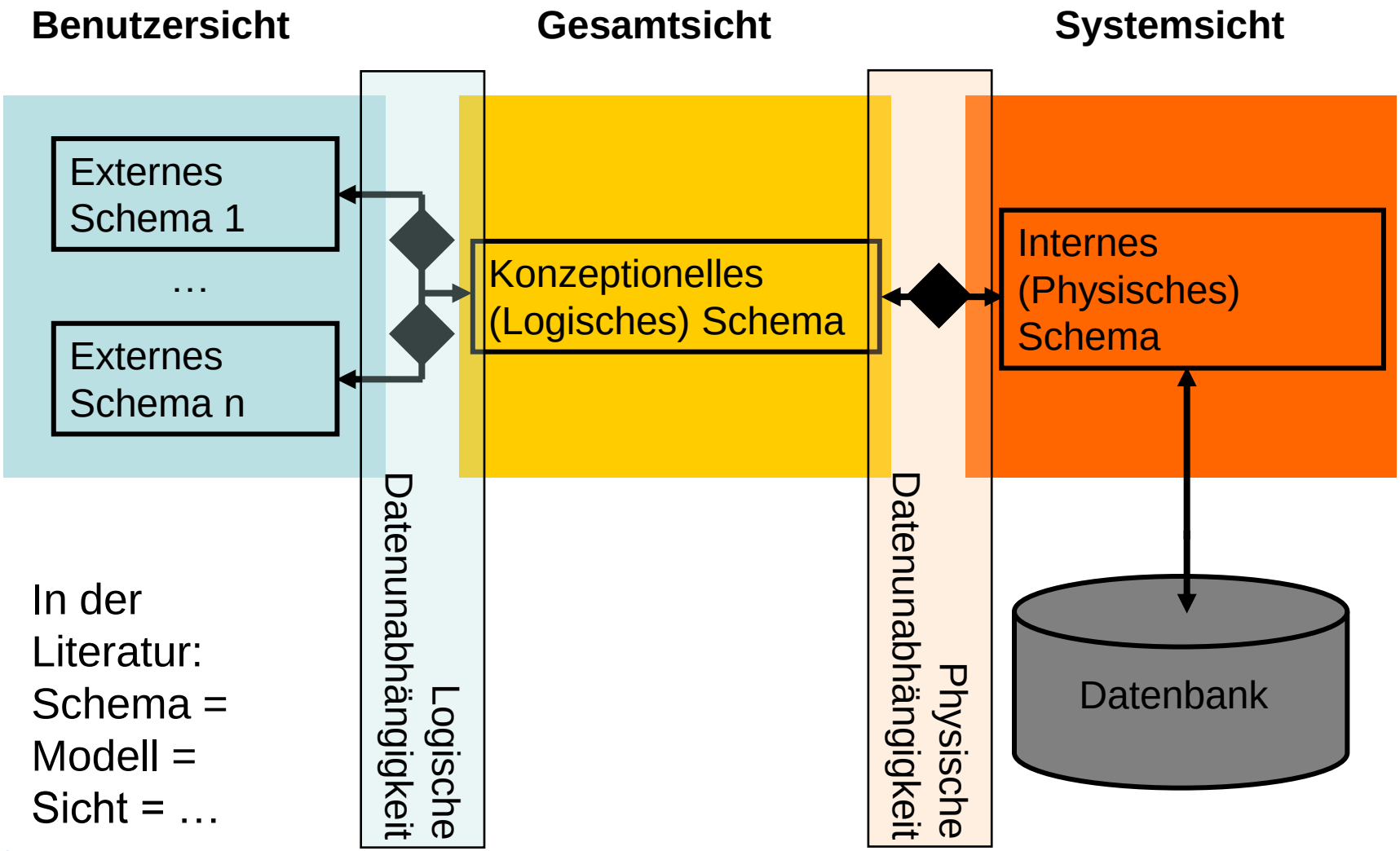
Die logische Struktur der Information in einer Datenbank nennt man **Schema**

- Schemas werden selten (wenn überhaupt) geändert
  - Analogie Programmiersprache: Variablenvereinbarung
- Generell unterscheiden wir
- **Logisches Schema (Externes / Konzeptuelles Schema)**
- **Physisches Schema (Internes Schema)**
- Was wiederum führt zu
- **Physische Datenunabhängigkeit**
- **Logische Datenunabhängigkeit**

Die Sammlung der Information in einem DBMS zu einem Zeitpunkt nennt man **Instanz**

- Schema/Instanz Analogie in Programmiersprache: Typ/Wert der Variablen

ANSI-SPARC-Architektur: 3-Schema-Konzept



In der Literatur:
Schema =
Modell =
Sicht = …

Physische Datenunabhängigkeit

Änderungen können im internen Schema vorgenommen werden, ohne das konzeptionelle (externe) Schema zu beeinflussen

Wird durch die Trennung zwischen interner und konzeptioneller Schicht erreicht, führt zu

- Geräteunabhängigkeit:

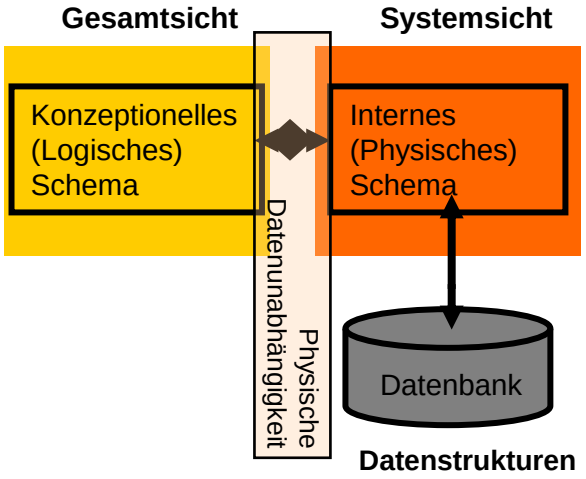
  Die Anwendungsprogramme sind unabhängig von der Hardware, die zur Speicherung der Daten benötigt wird

- Softwareunabhängigkeit:

  Die Anwendungsprogramme sind unabhängig von der Software, die die Daten verwaltet, d.h. Datenbanksystem

- Speicherungsstrukturunabhängigkeit:

  Die Anwendungsprogramme sind unabhängig von den internen Speicherungsstrukturen



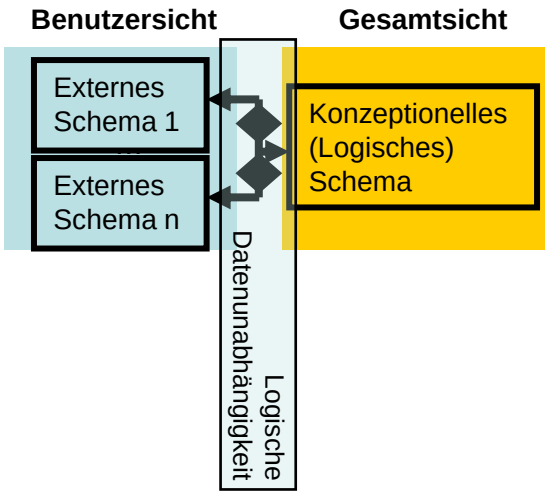
Logische Datenunabhängigkeit

Das konzeptionelle Schema kann geändert werden, ohne die externen Schemata ändern zu müssen

- Die logische Struktur der Datenbank ist unabhängig von den Schnittstellen, über die Anwendungsprogramme darauf zugreifen können

wird durch die Trennung von externer und konzeptioneller Schicht erreicht

- Applikationen werden nicht beeinflusst von Änderungen der Datenbasis, wie
  - Schema-Erweiterungen
  - Schema-Modifikationen

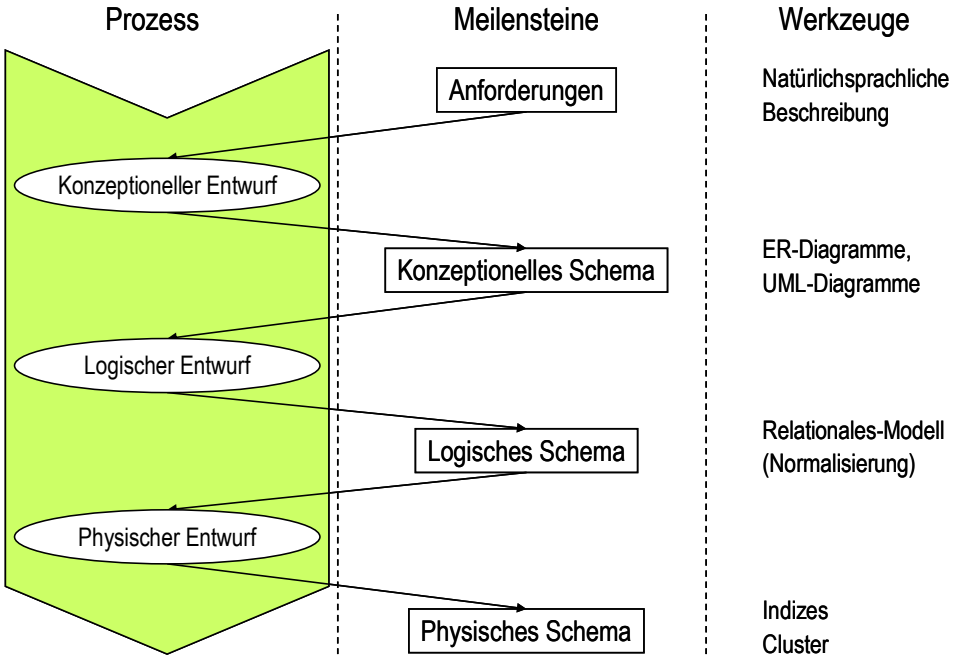


Datenbankentwurfsprozess

Vereinfacht ein Ansatz in drei Phasen

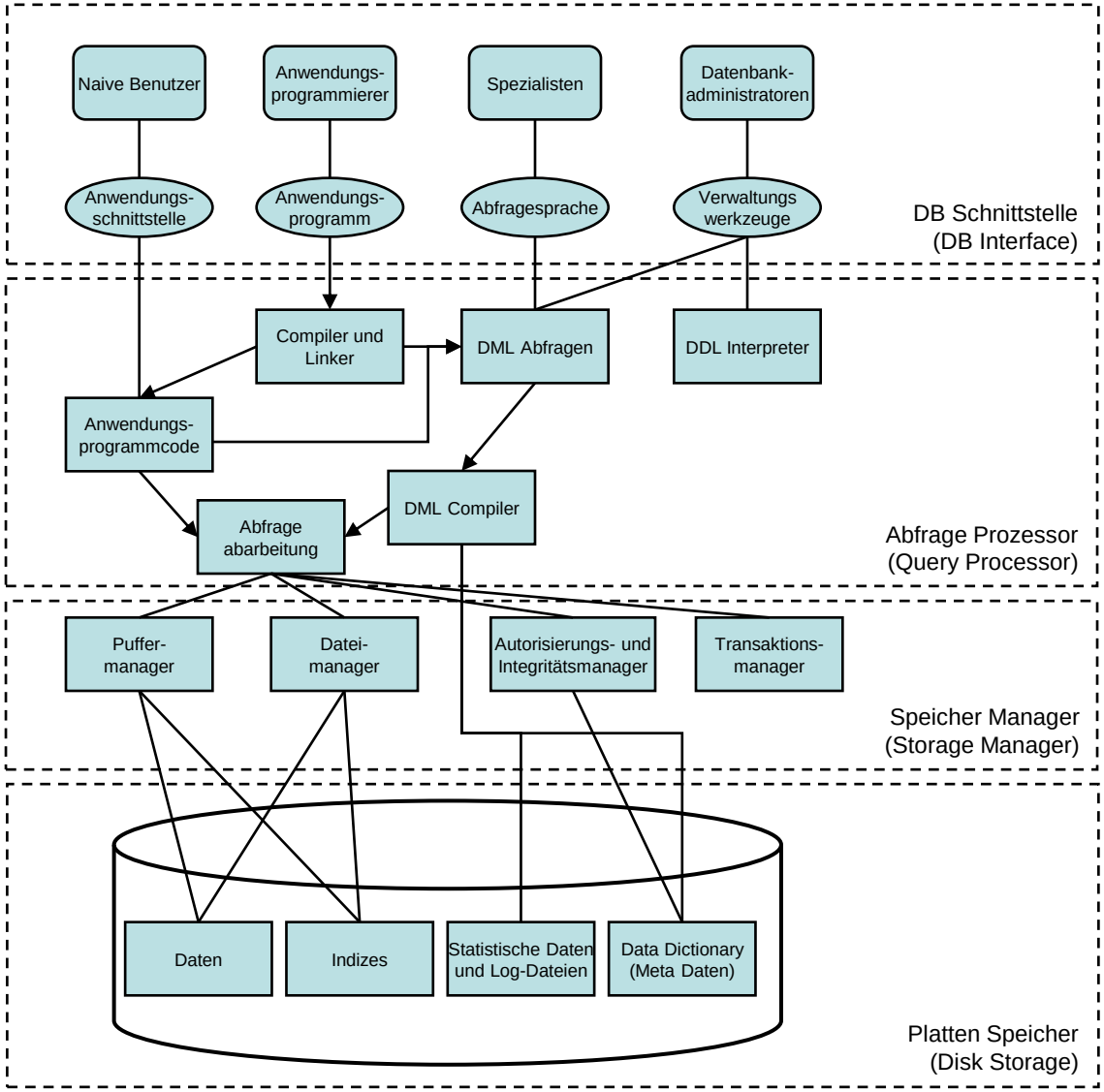
Konzeptioneller Entwurf - ausgehend von den Anforderungen wird ein konzeptionelles Schema entworfen, das Objekte des Ausschnitts der Realität mit ihren Eigenschaften und Beziehungen zu anderen Objekten in einem konzeptionellen Datenmodell erfasst.

Logischer Entwurf - das konzeptionelle Schema wird beim logischen Entwurf in ein logisches Schema übergeführt, das das Implementierungsmodell des spezifischen Datenbankmanagement-systems verwendet.



Physischer Entwurf - der physische Entwurf erstellt ein dem logischen Schema entsprechendes physisches Schema, wobei die Performanz des zu erwartenden Laufzeitverhaltens optimiert wird. Hierzu werden Indizes eingeführt und die Daten in Cluster (ähnliche Daten werden räumlich nahe gespeichert) angelegt.

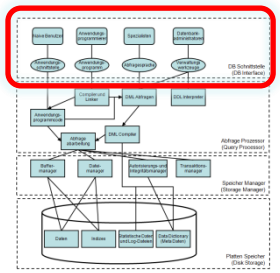
2.4 Komponenten eines DBMS



DB Schnittstelle

Auf der DB Schnittstellen Ebene unterscheiden wir eine Reihe unterschiedlicher Datenbank Benutzer

- **Naive Benutzer** – DB Zugriff über standardisierte Schnittstelle (z.B. Formular)
- **Spezialist** – Query Language Benutzer (z.B. SQL)
- **Anwendungsprogrammierer** – DB Anwendungsentwickler, Entwickler spezialisierter DB Anwendungen (z.B. JDBC, Programmiersprache)
- **Datenbank Administrator**
  - Besitzt zentrale Kontrolle über das DBMS
  - Aufgaben
    - Definition und Änderung des Schemas, Speicherstrukturen und Zugriffsmethoden
    - Gewährung von Zugriffsrechten
    - Spezifikation von Integritätsbedingungen
    - Überwachung der Leistungsfähigkeit
    - Ansprechpartner für den User
    - …



Abfrageprozessor

Der **Abfrageprozessor** besteht aus den folgenden wichtigen Komponenten

DDL Interpreter – die **Data Definition / Description Language** (**DDL**) dient der Beschreibung der logischen Datenstrukturen einer Datenbank (Datenbankschema).

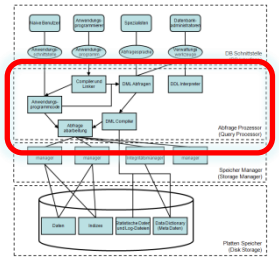
Der DDL Interpreter interpretiert alle DDL Anweisungen und vermerkt die Definitionen im **Data Dictionary**, das die **Metadaten**, die „Daten über Daten“, enthält.

DML Compiler – die **Data Manipulation Language** (DML) ermöglicht den Zugriff auf und die Manipulation der Daten der Datenbanken, wobei der Teil der DML, der die Daten abfragt, **Query Language** heißt.

Der DML Compiler übersetzt die DML Anweisungen in einen **Ausführungsplan**, der aus Basisoperationen besteht, die die Anfragebearbeitung ausführen.

Bei dieser Übersetzung wird auch eine **Optimierung** der Zugriffe unter Ausnützung möglicher Indexstrukturen oder Daten-Cluster durchgeführt, um die Performanz der Datenzugriffe zu steigern.

Abfrageabarbeitung (Query Evaluation Engine) führt die Basisoperationen durch.



Datenmanipulationssprache

Zwei Klassen von DML Sprachen

Prozedural – Benutzer spezifiziert, welche Daten benötigt werden und gibt an, wie man auf sie zugreift

Nicht-prozedural - Benutzer spezifiziert, welche Daten benötigt werden, ohne zu spezifizieren, wie man auf sie zugreift

Wichtigster Vertreter für **nicht-prozedurale** DMLs ist **SQL**

Structured Query Language

Wird jetzt wieder als „Sequel“ oder „Seekel“ ausgesprochen

Beispiel: Finde die Namen aller Studenten mit Studienkennzahl 926

```
select student.student_name
from   student
where  student.strkz = 926
```

Interaktives Abfragewerkzeug

Applikationsprogramme greifen auf DB zu über

Programmierspracherweiterungen die embedded SQL ermöglichen

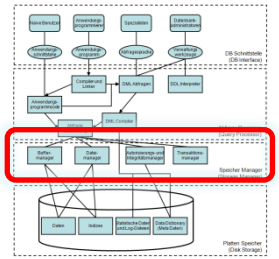
APIs (z.B. ODBC / JDBC) die es erlauben SQL Abfragen abzusetzen

Speicher Manager

Der **Speicher Manager** (Storage Manager) ist die Schnittstelle zwischen den in der DB gespeicherten („low-level“) Daten und dem Abfrage Prozessor und realisiert die Abfragen an das System

Er ist verantwortlich für

- Interaktion mit dem Platten Speicher
- Effizientes Speichern, Zugreifen und Ändern von Daten



Er besteht aus folgenden Software Komponenten:

- **Autorisierungs- und Integritätsmanager** – dieser überprüft Integritätsbedingungen und den Benutzerzugang.
- **Dateimanager** – er ist zuständig für die Zuweisung von Speicherplatz auf den Platten und den Aufbau von Datenstrukturen, die die gespeicherte Information repräsentieren.
- **Puffermanager** – er ist verantwortlich für den Transfer der Daten auf der Platte in den Hauptspeicher, und zurück.
- **Transaktionsmanager** (siehe nächste Folie)

Transaktions Manager

Der **Transaktions Manager** garantiert, dass die Datenbank in einem konsistenten (korrekten) Zustand bleibt, trotz Systemabstürzen oder Transaktionsfehlern

Weiters kontrolliert er die Interaktion zwischen gleichzeitig ablaufenden Transaktionen um die DB-Konsistenz zu erhalten.

Eine **Transaktion** ist eine Sammlung von Operationen die als einzelne logische Aktion in einer Datenbankapplikation ausgeführt wird

d.h. vollständig oder gar nicht

Transaktionen garantieren eine Reihe von Eigenschaften, die unter dem Akronym „**ACID**“ zusammengefasst sind.

ACID Eigenschaften

ACID steht für folgende vier Eigenschaften:

Atomicity (Atomarität): Eine Transaktion wird entweder vollständig oder gar nicht ausgeführt.

Consistency (Konsistenz): Das DBMS garantiert die Integrität der Datenbank auf Basis einer Menge von Integritätsregeln, die vom Benutzer definiert werden können. Die Ausführung einer isolierten Transaktion erhält die Konsistenz der Datenbank.

Isolation: Wenn mehrere Transaktionen gleichzeitig durchgeführt werden, darf keine Transaktion etwas von den anderen, gleichzeitig exekutierenden, bemerken.

Das heißt, für jedes Paar von Transaktionen T_i, T_j, scheint es, als ob entweder T_j vor dem Start von T_i beendet wurde, oder T_j gestartet wurde, nachdem T_i beendet wurde.

Durability (Dauerhaftigkeit): Wird eine Transaktion erfolgreich abgeschlossen, bleiben die Datenbankänderungen persistent (dauerhaft) erhalten, auch wenn danach ein Systemfehler auftritt.

Beispiel ACID

Es sei eine DB Transaktion gegeben, die 100 Euro von Konto A nach Konto B überweist

1. `read(A)`
2. `A := A - 100`
3. `write(A)`
4. `read(B)`
5. `B := B + 100`
6. `write(B)`

Atomicity: falls die Transaktion nach Schritt 3 und vor Schritt 6 scheitert, sollte das System sicherstellen, dass die Änderungen auf A nicht in der Datenbank durchgeführt werden

Andernfalls ergäbe sich eine Inkonsistenz, und zwar eine Abbuchung von A und KEINE Aufbuchung auf B, d.h. die 100 Euro sind verschwunden!

Consistency: die Summe von A und B wird durch die Ausführung der Transaktion nicht verändert.

Isolation: falls zwischen den Schritten 3 und 6 eine andere Transaktion Zugriff auf die teilweise veränderte Datenbank erhält, sieht sie eine inkonsistente Datenbank, d.h. die Summe A + B wäre kleiner als sie sein sollte.

Diese Eigenschaft kann am einfachsten sichergestellt werden, indem Transaktionen sequentiell ausgeführt werden, also eine nach der anderen. Das gleichzeitige Ausführen mehrerer Transaktionen hat aber entscheidende Vorteile.

Durability: sobald der Benutzer verständigt wurde, dass die Transaktion durchgeführt wurde (d.h., die Überweisung von 100 Euro hat stattgefunden), müssen die Änderungen in der Datenbank dauerhaft erhalten bleiben, unabhängig von eventuell später auftretenden Fehlern.

Platten Speicher

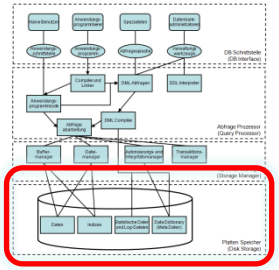
Der **Platten Speicher** (Disk Storage) verwaltet eine Reihe von Datenstrukturen, die die physische Implementierung der Datenbank repräsentieren:

Daten, die die Datenbank darstellen,

Indizes, die den effizienten Zugriff auf die gespeicherten Daten erlauben,

Data Dictionary, das die Metadaten über die gespeicherten Datenstrukturen, das Datenbankschema, enthält, und

Statistische Daten und Log Dateien, die Kennzahlen über die gespeicherten Daten enthalten (wie Anzahl, Verteilung, Selektivitätsfaktoren)



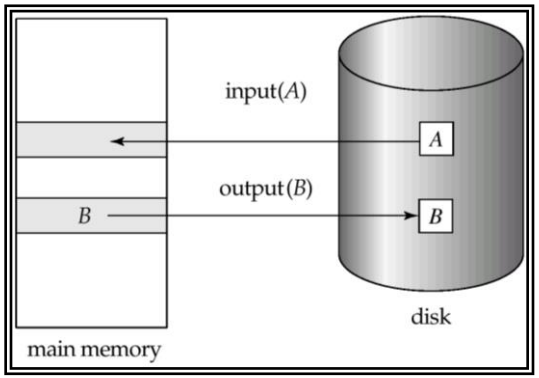
Datentransfer

Eine Datenbankdatei wird aufgeteilt in Speichereinheiten fester Größe, genannt **Blöcke** (blocks)

Blöcke sind sowohl die Einheit für die Speicherallokation (**Bucket**) und den Datentransfer (**Seite**, **page**)

Ein DBMS versucht die Anzahl der Blocktransfers zwischen Platte und Hauptspeicher zu minimieren

Ziel ist, soviel Blöcke wie möglich im Hauptspeicher zu halten



Puffer – Teil des Hauptspeichers um Kopien der Platten Blöcke zu speichern

Puffermanager – Teilsystem verantwortlich für die Allokation von Pufferbereichen im Hauptspeicher

Puffermanager

Programme kontaktieren den **Puffermanager**, wenn sie einen Block von der Platte benötigen

- Falls der Block im Puffer ist, wird dem Programm die Adresse des Blocks im Hauptspeicher übergeben
- Falls der Block nicht im Puffer ist:
  1. Der Puffermanager **alloziert Speicher** im Puffer für den Block, indem er, falls notwendig andere Blöcke entfernt um Platz zu schaffen
  2. Der zu **entfernende Block** wird, falls er verändert wurde, auf die Platte **geschrieben**,
  3. Nachdem Speicherplatz alloziert wurde, **liest** der Puffermanager **den Block** von der Platte und übergibt die Adresse dem anfordernden Programm

Puffer-Ersetzungsstrategien (Cache-Algorithmen)

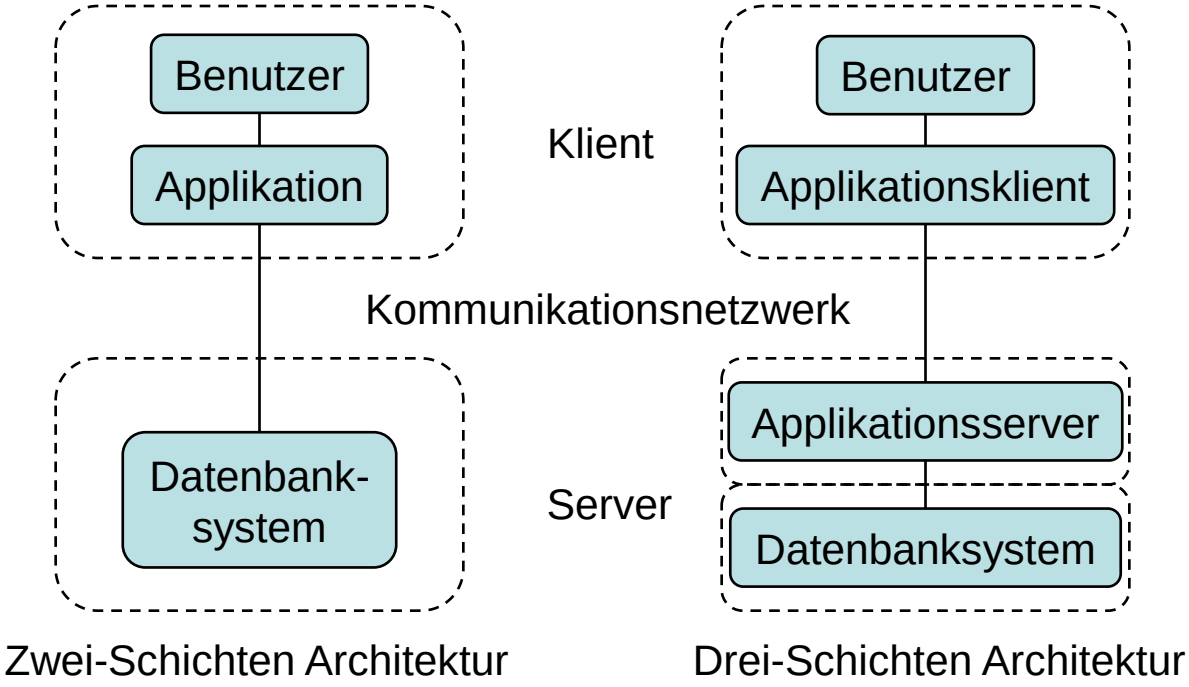
- **LRU**, Least Recently Used („Am längsten nicht verwendet“): ersetze den Block, dessen letzte Referenzierung zeitlich am längsten zurückliegt
- **MRU**, Most Recently Used („zuletzt verwendet“): ersetze den Block, dessen Referenzierung zeitlich am kürzesten zurückliegt
- …

Zwei-Schichten (Two-Tier) Architektur

Klientenprogramme kommunizieren über ODBC/JDBC mit dem DBMS

Drei-Schichten (Three-Tier) Architektur

Klienten benutzen „Middleware“ oder das Web



SOA / Grid / Cloud Architektur (klassisch)