

Datenbanken

Informatik, ICS und als Wahlfach

1. Einführung & Datenbankgrundlagen

Prof. Dr. Markus Goldstein

SoSe 2022

1.1 Fallstudie

1.2 Grundlagen

1.2.1 ANSI SPARC

1.2.2 CASE Werkzeuge

1.2.3 Datenbank- vs. dateibasierte Anwendungssysteme

1.2.4 Phasenmodell für Datenbankentwurf

- Neben Ihrem Studium suchen Sie nach einer zusätzlichen Verdienstmöglichkeit
- Daher eröffnen Sie zusammen mit einem Kommilitonen einen Kiosk mit Bio- und Fairtrade-Produkten in Ulm (Marktnische).



- Da Sie möglichst wenig Geld investieren möchten, beschließen Sie, möglichst kostenlose Software einzusetzen.
- Auf der Suche nach einer Kassensoftware zur Verwaltung der Einkäufe finden Sie eine Excel bzw. LibreOffice-Datei. Da Sie beide gut mit Excel umgehen können, denken Sie, dass dies eine gute Lösung sein könnte. Also laden Sie die Datei herunter, um sich diese näher anzusehen.
- **Bearbeiten Sie die Aufgaben zum Einführungsbeispiel in Moodle in den nächsten 20 Minuten**

- Leere Felder (z.B. Spalte „Zeit“, „Preis“)
- Unterschiedliche Schreibweisen
 - bar/Bar; kosmetik/Kosmetik
 - Leerzeichen vor Strings
 - „Migros Zollikon“ 3x vorhanden
- Leer vs. „“ (vergl. Laden)
- Unterschiedliches Format (vergl. PLZ im Laden oder Kommentar)
- Filtern nach PLZ nicht möglich (nicht atomar)
 - Spalte Klasse zusammengesetzt
- Redundanzen: „Lebensmittel“ kommt mehrfach vor (Umbenennen schwierig)

- Datenbanken wollen Probleme „normaler Tabellen“ vermeiden
- Es geht um **Konsistenz** der Daten
 - Erfüllung von Bedingungen
 - Widerspruchsfreiheit
- Eine Datenbank hat eine bestimmte **Struktur** und enthält **Daten**
- Erreichung durch **praktische Umsetzung** von **theoretischen Grundlagen**

- Es gibt viele verschiedene **Modelle** im Zusammenhang mit Datenbanken
 - Externes, konzeptionelles/logisches und internes Modell (bzw. Schema)
 - Datenmodell (insbes. Relationales Modell)
 - (E)ER-Modell
- Zusammenhänge:
Konsistenzhöhung durch **Datenunabhängigkeit** aufgrund der ANSI-SPARC 3-Schichten Architektur

1.1 Fallstudie

1.2 Grundlagen

1.2.1 ANSI SPARC

1.2.2 CASE Werkzeuge

1.2.3 Datenbank- vs. dateibasierte Anwendungssysteme

1.2.4 Phasenmodell für Datenbankentwurf

- Probleme von dateibasierten „Datenbanken“:

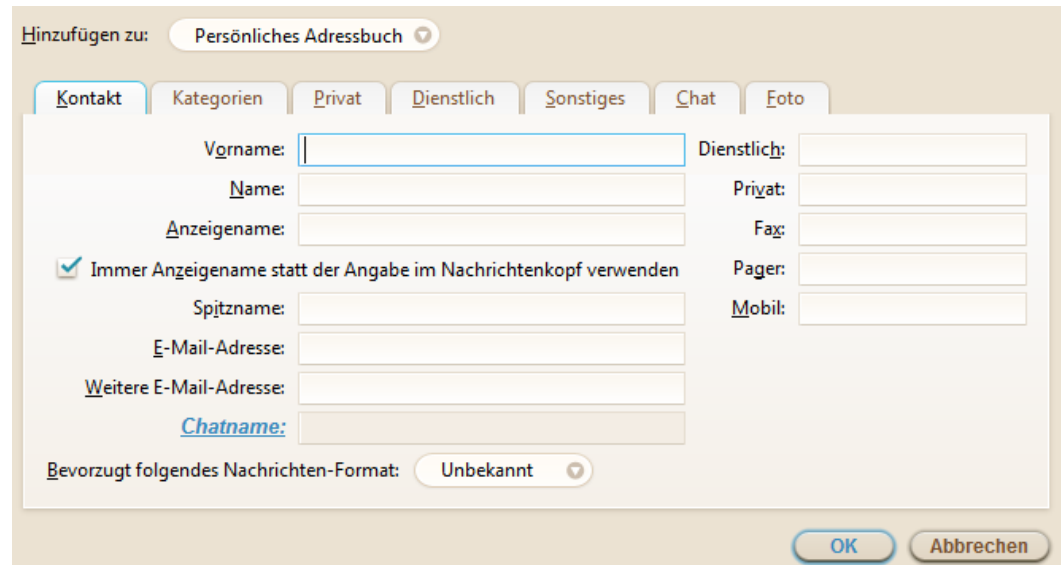


- Änderung der Anwendungslogik führt unmittelbar zur Änderung der Datenstruktur.
- Änderung der Datenstruktur bedingt Änderung der Zugriffslogik.
- **Starke Abhängigkeit/ Verzahnung von Applikation und Datenhaltung**

- **Lösung:** Einführung von Abstraktionsebenen zwischen Anwendung und Datenbank durch das American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC)

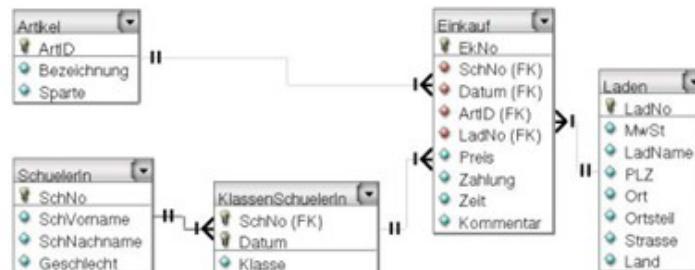
- **Abstraktionsebenen** der Referenzarchitektur
 - Externe Ebene
 - Konzeptionelle Ebene
 - Interne Ebene

- **Externe Ebene: Sichten**
- Nutzer benötigen typischerweise nur *Teilausschnitt* der Daten
 - Nutzer-/Programmsicht der Anwendungsdaten
 - Datenbankbeschreibung für einen bestimmten Nutzer bzw. ein Anwendungsprogramm
- Konkret: **Spezifikation** (textuell-graphisch) der notwendigen Datensicht
- Beispiele: Formulare, personalisierte Webseiten, Anwendungen (Adressbuch z.B. im E-Mail Programm)



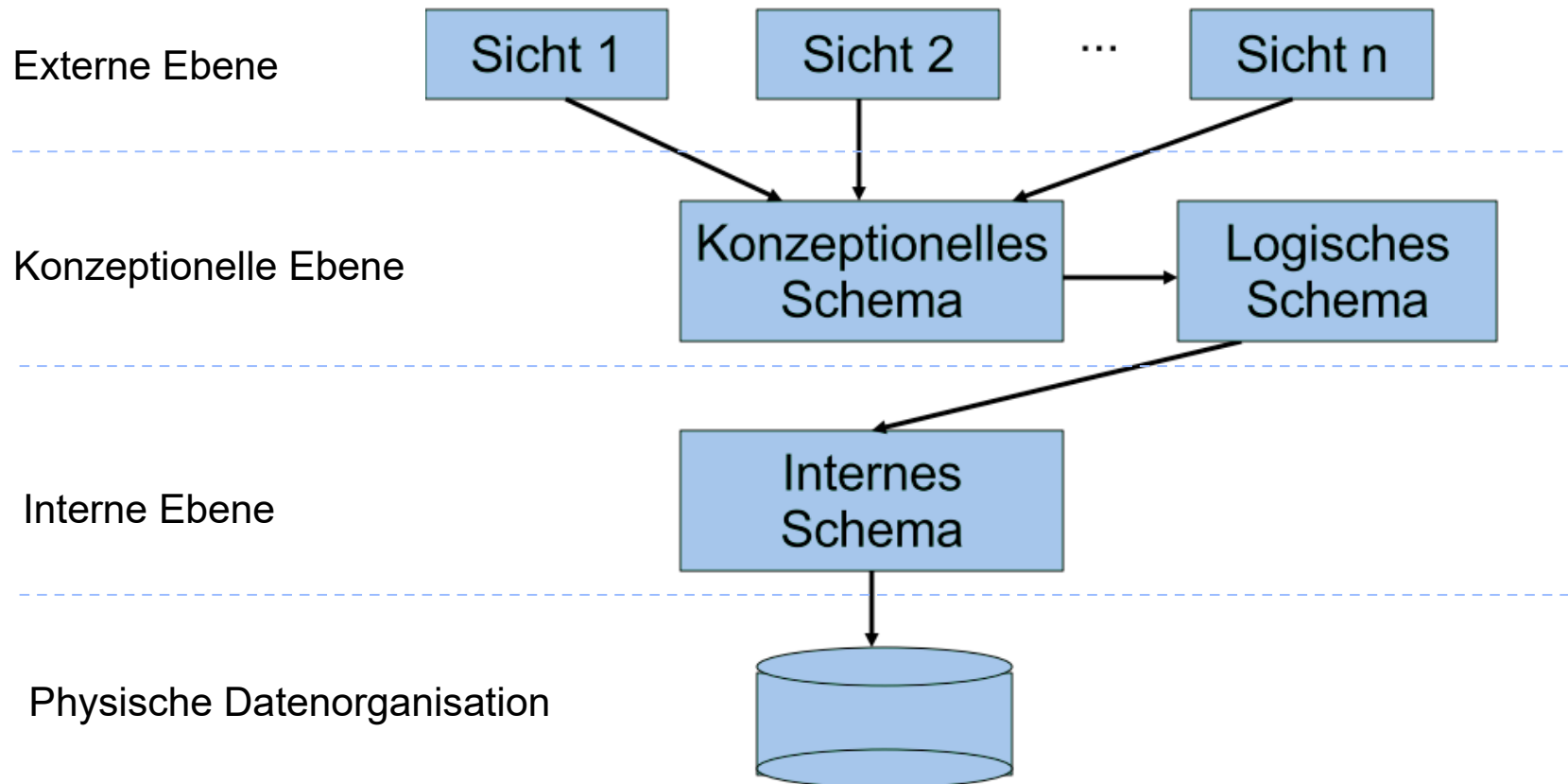
The screenshot shows a web form for adding a contact to a personal address book. The form has a title bar 'Hinzufügen zu: Persönliches Adressbuch' with a dropdown arrow. Below the title bar are several tabs: 'Kontakt' (selected), 'Kategorien', 'Privat', 'Dienstlich', 'Sonstiges', 'Chat', and 'Foto'. The form contains several input fields: 'Vorname:', 'Name:', 'Anzeigename:', 'Spitzname:', 'E-Mail-Adresse:', 'Weitere E-Mail-Adresse:', 'Chatname:', 'Dienstlich:', 'Privat:', 'Fax:', 'Pager:', and 'Mobil:'. There is a checkbox labeled 'Immer Anzeigename statt der Angabe im Nachrichtenkopf verwenden' which is checked. At the bottom, there is a label 'Bevorzugt folgendes Nachrichten-Format:' followed by a dropdown menu showing 'Unbekannt'. At the very bottom right are two buttons: 'OK' and 'Abbrechen'.

- **Konzeptionelle Ebene: Schema**
- Vereinheitlichung der Sichten der Externen Ebene
→ **Datenbankschema (gesamt)**
- Vollständige Beschreibung der für alle Anwendungen relevanten
 - Objekte (Entitäten)
 - und deren Beziehungen
- Konkrete Umsetzung: **(E)ER-Modell**



- **Interne Ebene: Schema**
- physikalische Darstellung der Datenbank im Computer
- Speicherstrukturen zur Ablage der Daten
 - Wie und wo werden die Daten binär gespeichert?
 - Ziel: Effizienz beim Zugriff auf die Daten
- Konkrete Umsetzung
 - **SQL-Skript** zur Definition von Tabellen (mit Datentypen)
 - **JSON-Datei** zur Definition eines Dokuments

- **Darstellung als 3-Schichten und physikalische Speicherung**



ANSI-SPARC – Eigenschaften

- Gleiche Daten für alle Nutzer
- Änderungen in Nutzersichten sind lokal für die Anwendung
- Datenspeicherung (wie und wo) für Nutzer unwichtig
 - ohne Einfluss auf Sicht
 - Änderungen bewirken keine Änderung der Sicht
 - **Abstraktion**
- Anwendungsstruktur für Datenbankaufbau unerheblich
- Änderung im Datenbankaufbau ohne Wirkung auf Nutzersicht

- Beispiel

Externe Ebene

Externe Sicht

SNo	Artikel	Sparte	Preis
-----	---------	--------	-------

Externe Sicht

Zahlung	Kaufdatum	Kaufzeit	Kommentar	LNo
---------	-----------	----------	-----------	-----

Konzeptionelle Ebene

Konzeptionelles/Logisches Schema

EKNo	Artikel	Sparte	Preis	Zahlung	Kaufdatum	Kaufzeit	Kommentar	SNo	LNo
------	---------	--------	-------	---------	-----------	----------	-----------	-----	-----

Interne Ebene

Internes Schema

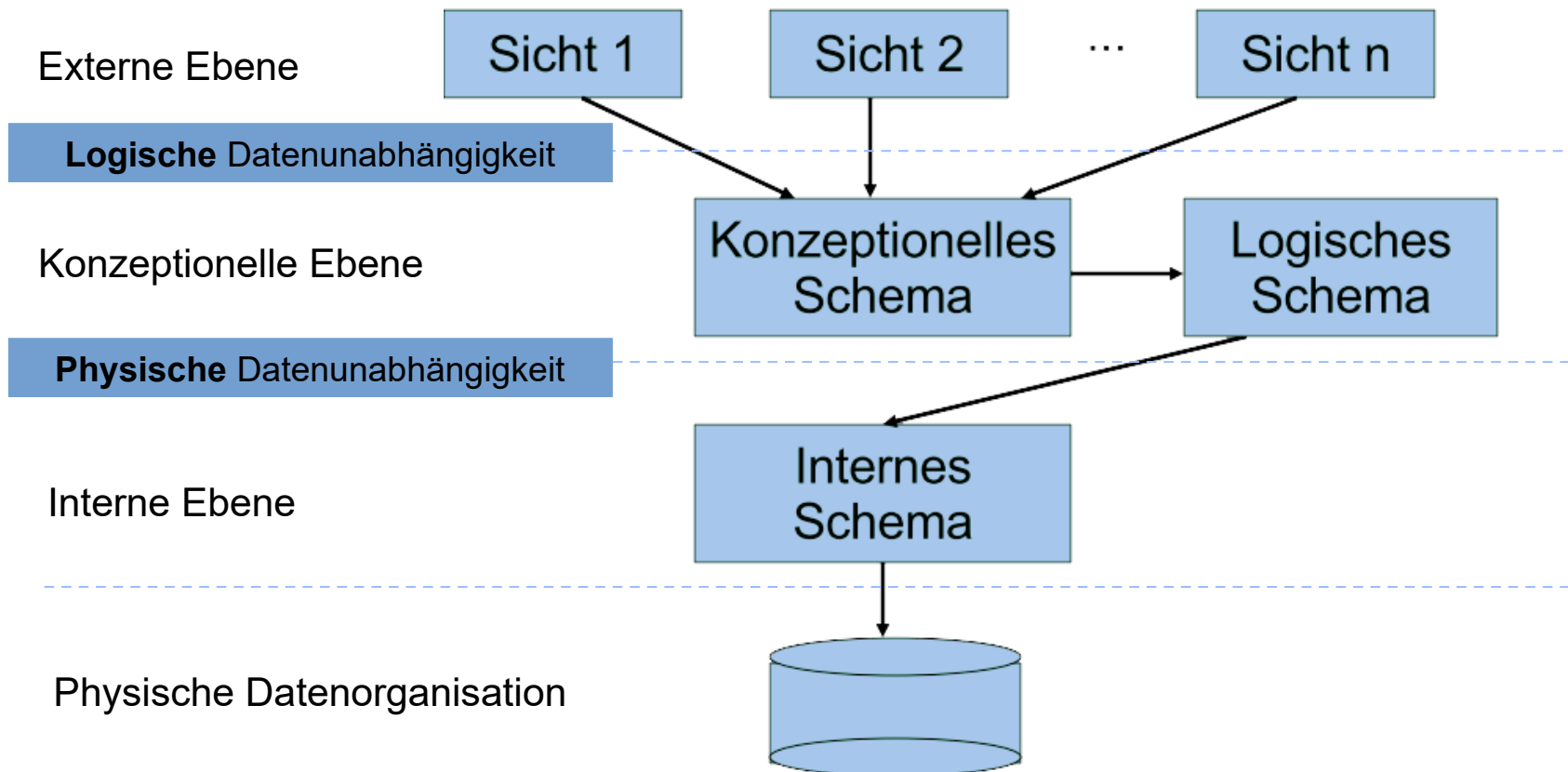
```
CREATE TABLE Einkauf(
  EKNo INT NOT NULL AUTO_INCREMENT,
  Artikel CHAR( ) NOT NULL,
  Sparte CHAR( ) NOT NULL,
  Preis DECIMAL( , ) NOT NULL,
  Zahlung CHAR( ),
  Kaufdatum DATE,
  Kaufzeit TIME,
  Kommentar CHAR( ),
  FOREIGN KEY (LadNo) REFERENCES Laden (LadNo),
  FOREIGN KEY (SchNo) REFERENCES SchuelerIn (SchNo),
  PRIMARY KEY (EKNo));
```

```
Struct EINKAUF{
  int EKNo;
  char Artikel[ ];
  char Sparte[ ];
  float Preis;
  char Zahlung[ ];
  struct date Kaufdatum;
  struct time Kaufzeit;
  char Kommentar[ ];
  struct EINKAUF *next;
}
index LadNo; index SchNo;
```


- **Logische Datenunabhängigkeit**
 - Idee: Externe Ebene getrennt von konzeptioneller Ebene
 - Anwendungen werden nicht beeinträchtigt, wenn Änderungen am Schema vorgenommen werden
 - Bei logischer Datenunabhängigkeit:
Keine Änderung an Spezifikationen bzw. Anwendungsprogrammen durch
 - Hinzufügen
 - Ändern
 - Löschen
- von Objekten (Entitäten)
- Nur **teilweise** gegeben in praktischen Datenbanksystemen

- **Physische Datenunabhängigkeit**
 - Idee: Konzeptionelle Ebene getrennt von interner Ebene
- Aus physischer Datenunabhängigkeit folgt:
Keine Änderung am konzeptionellen oder gar externen Schema durch Umstellung der
 - Dateistruktur
 - Speicherstruktur (z.B. „Hersteller“ der Datenbank)
 - Speichermedien
 - Anzahl der DB Server
- **Weitestgehend** gegeben in praktischen Datenbanksystemen

- **Darstellung als 3-Schichten und physikalische Speicherung**



- **(Enhanced) Entity-Relationship-Modellierung** zur Darstellung

- Konzeptioneller -
- und logischer Schemata



Gleiche Modellierung für zwei verschiedene Dinge

- **Entity:** Objekt (mit Attributen)
- **Relationship:** Beziehung zwischen den Objekten
- **ER-Diagramm** (oder auch ERD)
- Später mehr

1.1 Fallstudie

1.2 Grundlagen

1.2.1 ANSI SPARC

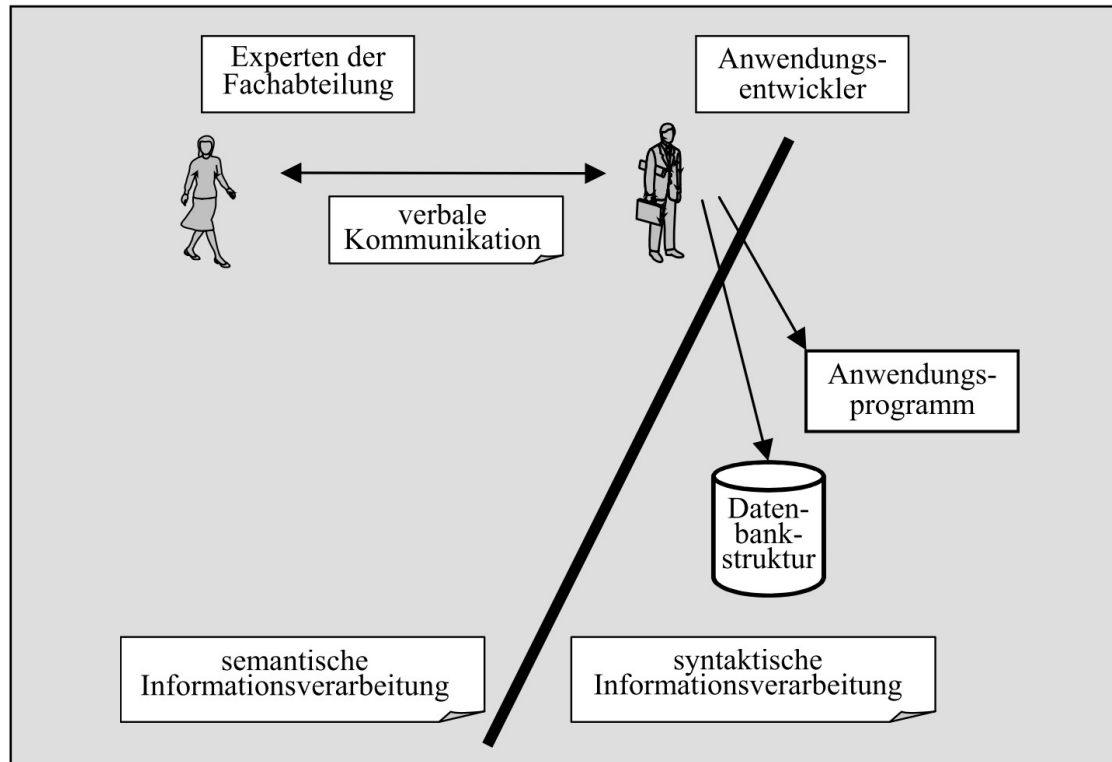
1.2.2 CASE Werkzeuge

1.2.3 Datenbank- vs. dateibasierte Anwendungssysteme

1.2.4 Phasenmodell für Datenbankentwurf

Traditionelle Anwendungsentwicklung

- Durch „verbale“ Kommunikation mit natürlicher Sprache

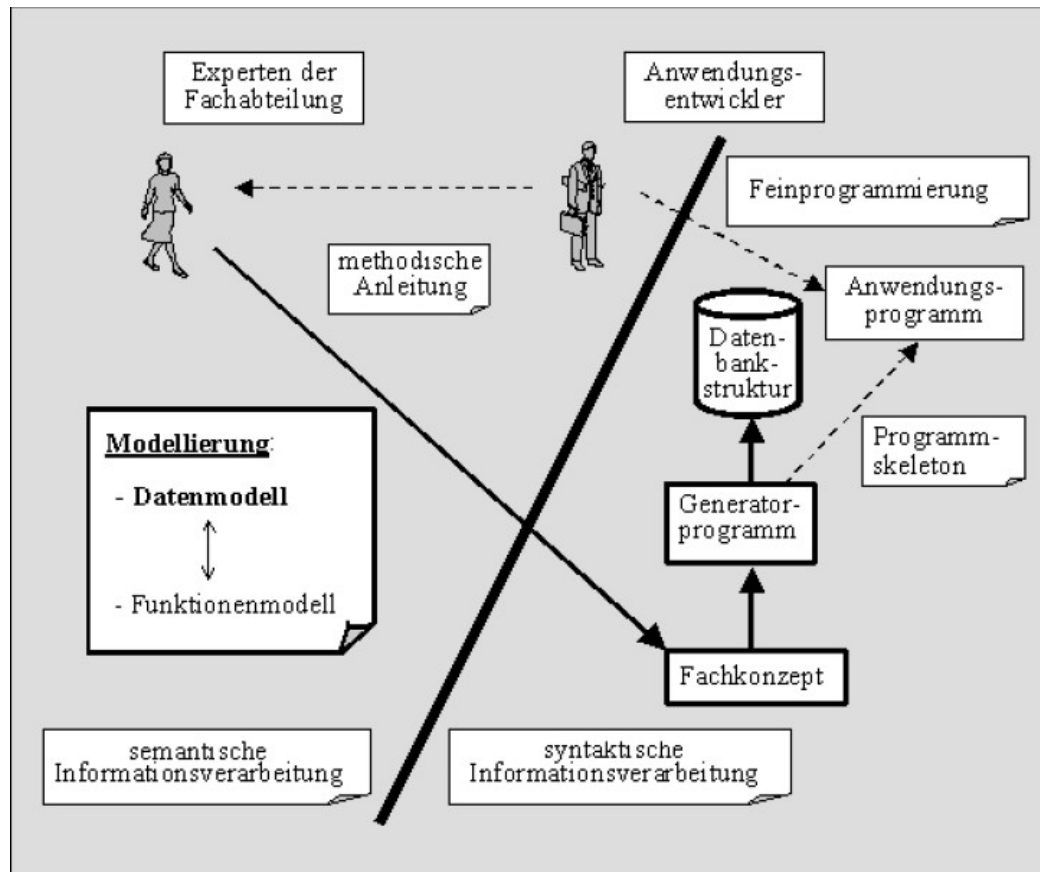


Traditionelle Anwendungsentwicklung

- Durch „verbale“ Kommunikation mit natürlicher Sprache
- Mögliche Nachteile
 - Natürliche Sprache sehr ungenau
 - Verschiedene Fachrichtungen und Terminologien
 - Fachkenntnisse lassen sich nicht einfach übertragen
 - Fachwissen hängt an Person

Veränderte Anwendungsentwicklung

- Mit Hilfe eines Fachkonzepts (ER-Modell)



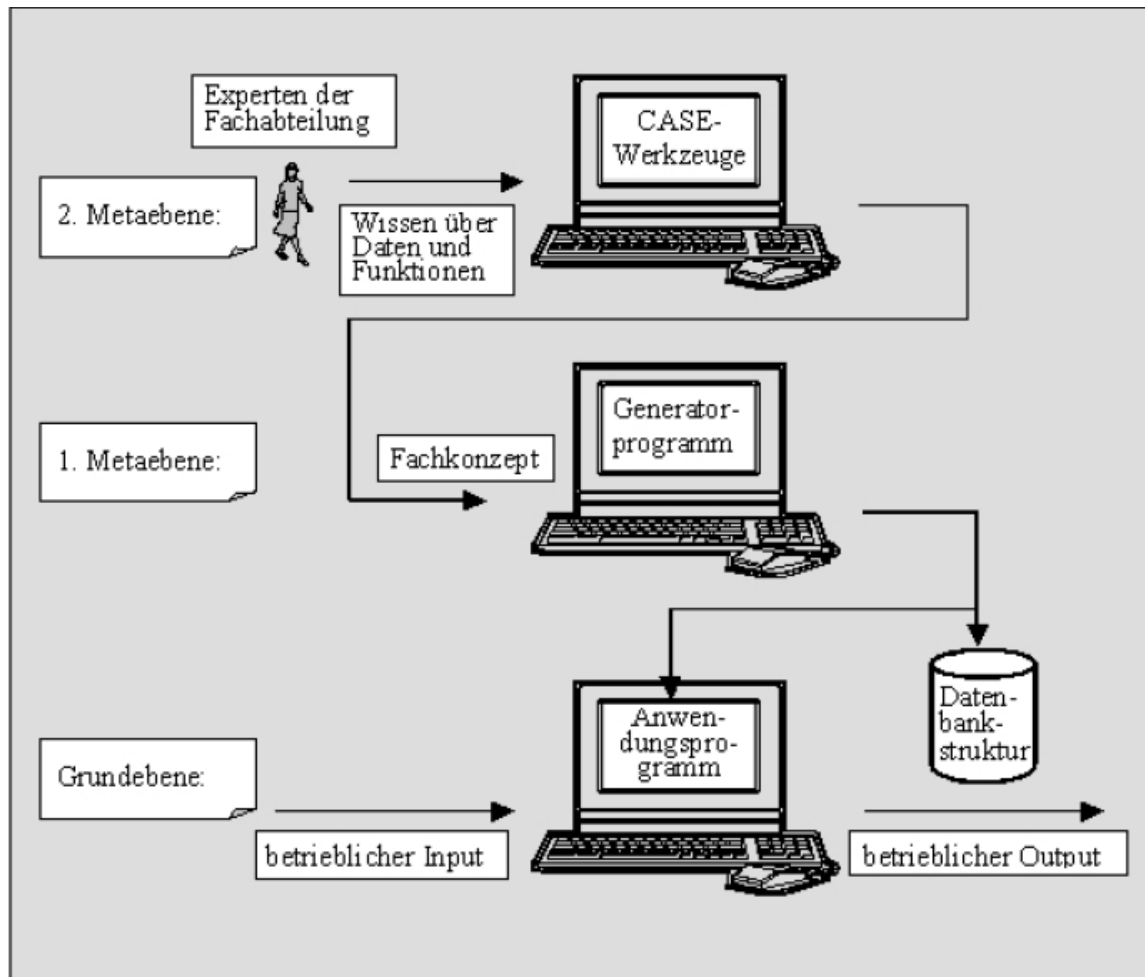
Veränderte Anwendungsentwicklung

- Durch **CASE**
 - Computer
 - Aided
 - Software
 - Engineering
- Dt.: Rechnergestützte Softwareentwicklung
- Entwicklung von Software nach ingenieur-wissenschaftlichen Methoden unter Verwendung eines Computers
- **Ziel:** Erstellung von Software möglichst **automatisch** aus dem Fachkonzept

- Oftmals graphische Notation des Fachkonzepts
- CASE-Werkzeuge (Tools)
 - Planung
 - Entwurf
 - Dokumentation
 - Ggf. Generierung von Quelltext
- Können in die IDE integriert sein
- Beispiele: Tools für UML oder ER-Diagramme

CASE Ablauf: Forward Engineering

- Wissen → Fachkonzept → Datenbank



1.1 Fallstudie

1.2 Grundlagen

1.2.1 ANSI SPARC

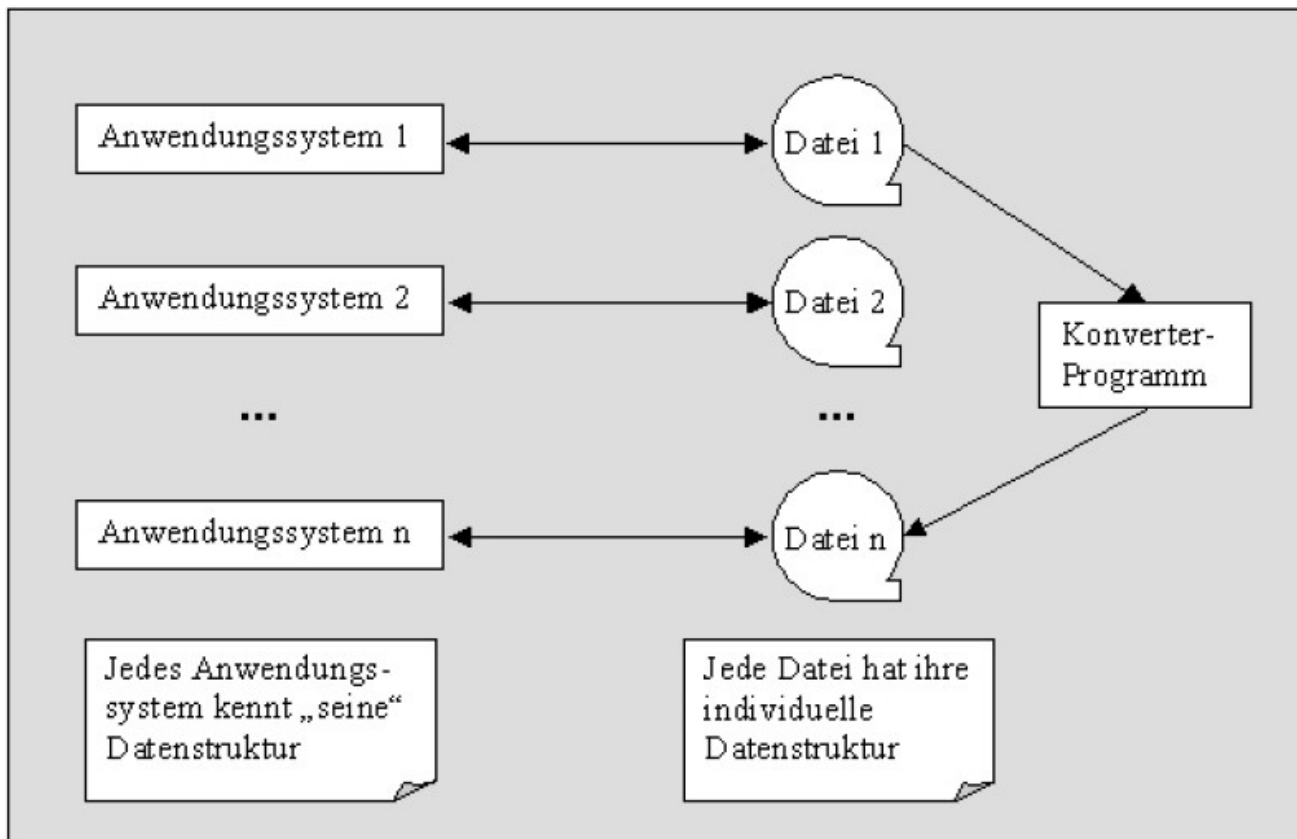
1.2.2 CASE Werkzeuge

1.2.3 Datenbank- vs. dateibasierte Anwendungssysteme

1.2.4 Phasenmodell für Datenbankentwurf

Architektur Datei-basierter Anwendungen

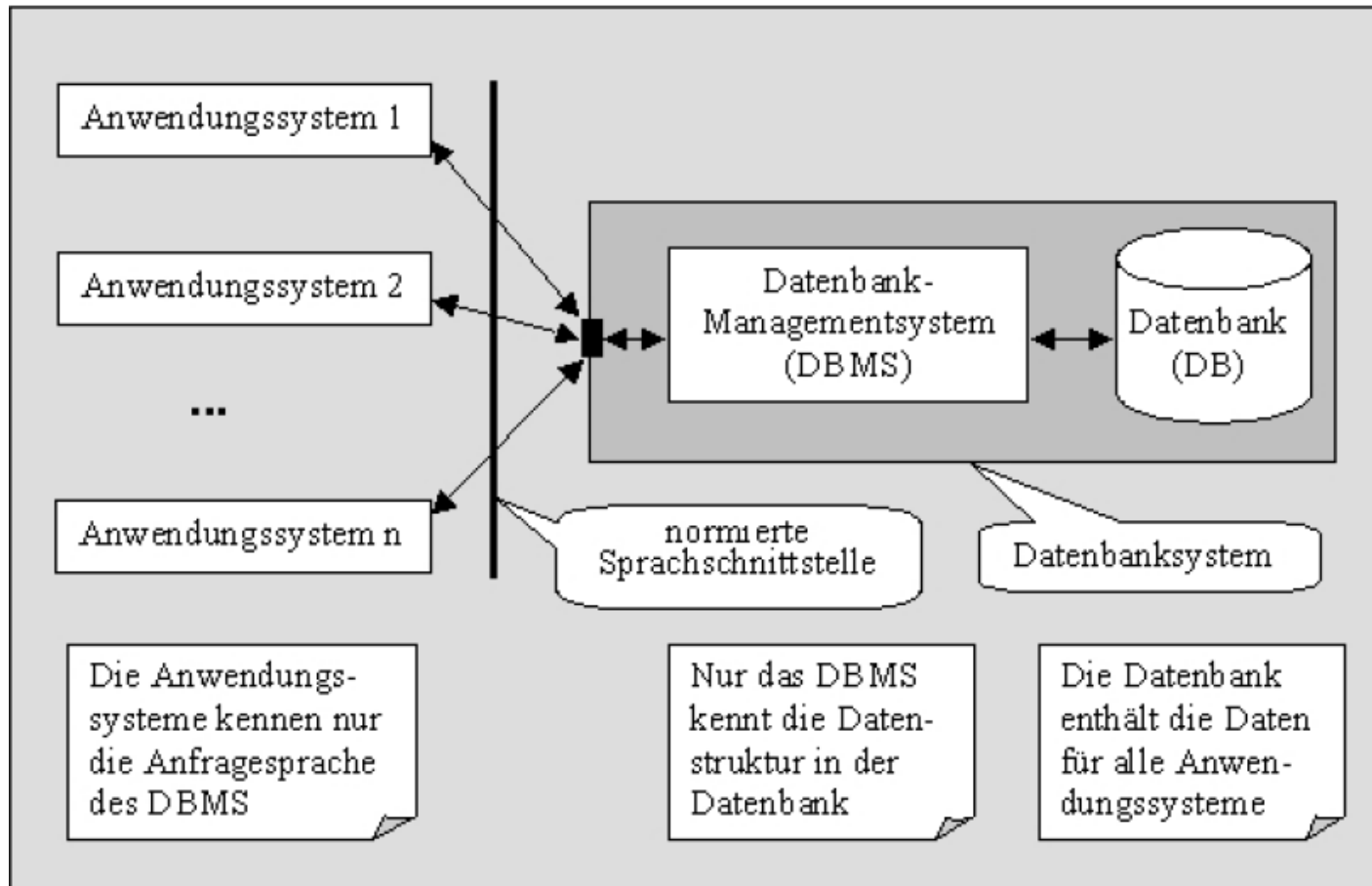
- Jede Anwendung hat ihre „eigene“ Datei



Zahlreiche Nachteile

- Redundanz
- Gemeinsamer Zugriff mittels Konvertern
- Datenstrukturänderung bedingt Umprogrammieren
- Keine parallelen Zugriffe möglich
- Keine Sicherungsmechanismen
 - Konsistenz (z.B. „gleichzeitigen“ Updates),
 - Zugriffsrechte und
 - Wiederherstellung

- Gemeinsame Datenbasis



DBMS vs. DB

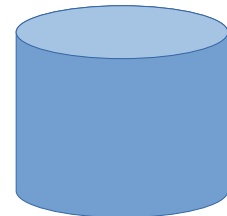
Datenbank (engl. database – DB)

- Sammlung strukturierter Daten
- sachlogische Zusammenhänge untereinander



Datenbankmanagementsystem (engl. Database Management System – DBMS)

- Programmsystem (Software als Prozess oder Bibliothek)
- Systemsoftware für alle Aspekte der Daten**verwaltung**
- Beinhaltet oft eine oder mehrere DBen



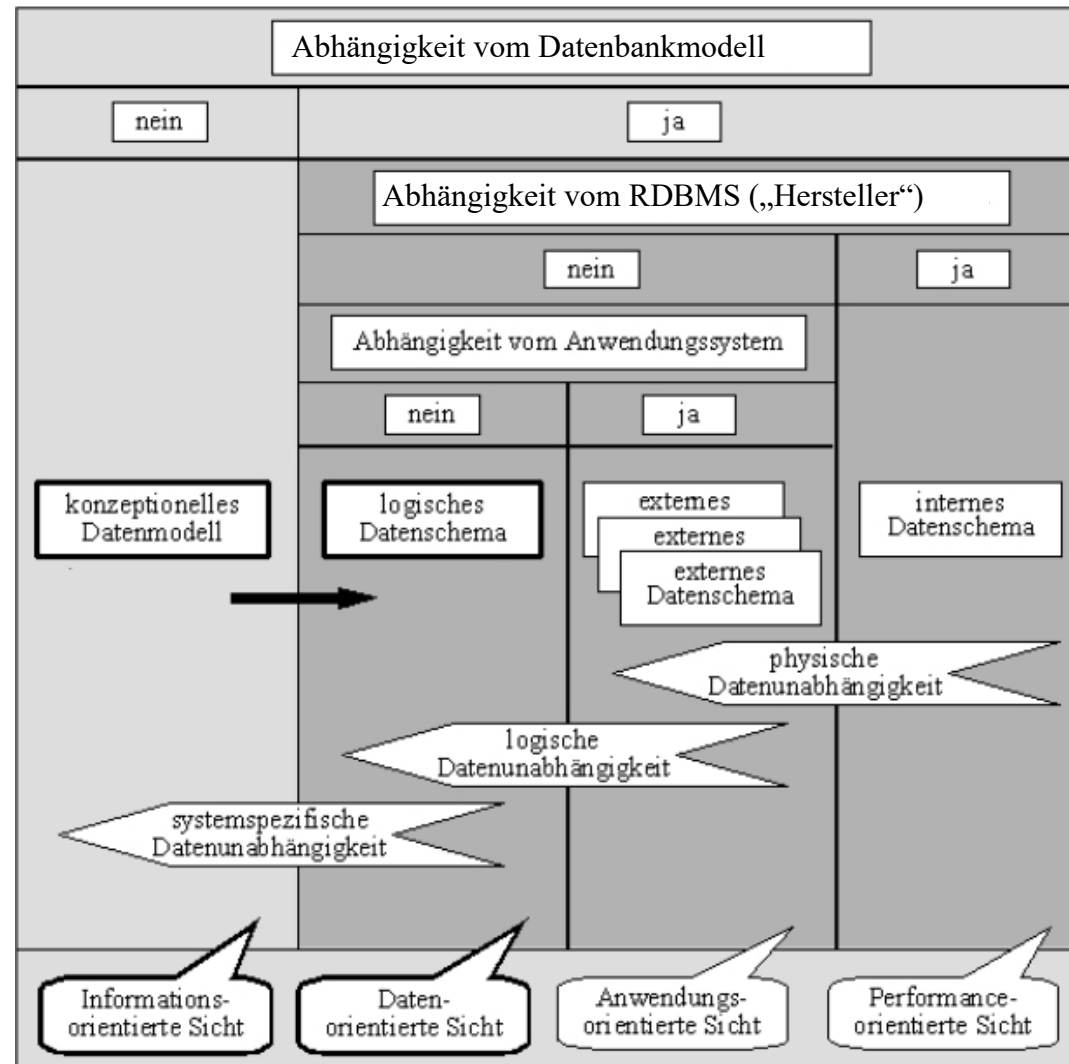
RDBMS: Relationales DBMS (später mehr)

Zahlreiche Vorteile

- Redundanzfrei
- Logische Datenunabhängigkeit
- Physische Datenunabhängigkeit
- Mehrbenutzerbetrieb mit Rechteverwaltung
- Normierte Schnittstelle (für Sichten)
- Effiziente Verwaltung (Daten schnell finden)
- Integritätssicherungsmechanismen

- Datenintegrität
 - Vollständigkeit
 - Widerspruchsfreiheit
 - Korrektheit
- Zugriffsschutz
- Datenwiederherstellung nach Störungen
 - Transaktionen (später)
- Persistenz

- Abhängigkeiten



1.1 Fallstudie

1.2 Grundlagen

1.2.1 ANSI SPARC

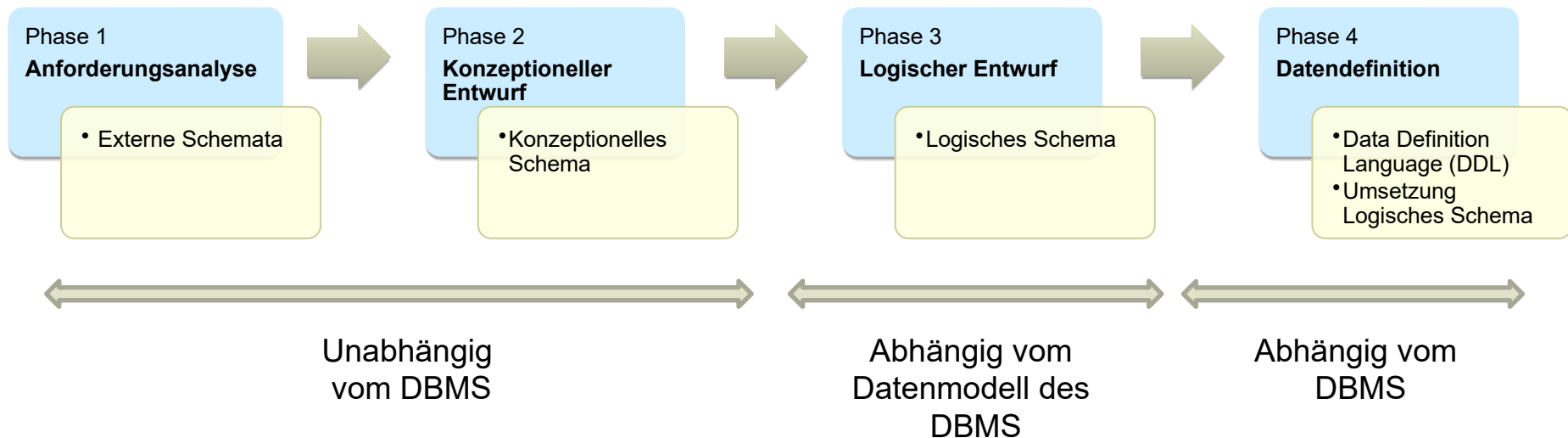
1.2.2 CASE Werkzeuge

1.2.3 Datenbank- vs. dateibasierte Anwendungssysteme

1.2.4 Phasenmodell für Datenbankentwurf

Phasenmodell für Datenbankentwurf

- Prinzipielles Vorgehen beim Erstellen einer Datenbank



(angelehnt an Thomas Kudraß, Taschenbuch Datenbanken, 2. Auflage, Hanser Verlag 2015, S. 45 ff)

1. Anforderungsanalyse

Anforderungen der potentiellen Benutzer werden erfasst

- Informelle Beschreibung (Tabellen, Texte, etc)
- Unterscheidung in **Informations-** und **Bearbeitungsanforderungen**
 - Informationsanforderungen → Datenmodell
 - Bearbeitungsanforderungen → Funktionenmodell
- Funktionenmodell: **Create**, **Read**, **Update**, **Delete** (CRUD)
 - Die CRUD-Matrix beschreibt die Beziehungen zwischen Datenmodell und Funktionenmodell

1. Anforderungsanalyse

Beispiel einer CRUD-Matrix

Funktionen	Kunden verwalte n	Auftrag abwickel n	Auftrag abrech nen	Lieferan ten verwalt en	Bestell ung erstelle n	Bestell ung abrech nen	Warenei ngang überprüf en	...
Objekte	Kundenaufträge			Beschaffung				...
Kunde	CRUD	R	R					...
Auftrag		CRUD	R					...
Auftragsposition		CRUD	R					...
Kundenrechnung			CRU					...
Kundenrechnungsposition			CRU					...
Artikel		R	R		R	R	R	...
...

2. Konzeptioneller Entwurf

Ziel: Erste formale Darstellung erstellen

- i.d.R. konzeptionelles Schema ((E)ER-Modelle)
- und externe Schemata (Sichten)

- Datenmodelle sind abstrakte Darstellung der Wirklichkeit
 - Starke Vereinfachung
 - Stellen nur Ausschnitt der Realität dar und wird auch Universe of Discourse (UoD) genannt

2. Konzeptioneller Entwurf

Ansätze zur Erstellung eines konzeptionellen Entwurfs

- **Top-Down-Ansatz**

- Modellierung des konzeptionellen Schemas und Ableitung der nicht notwendigerweise disjunkten externen Schemata

- **Bottom-Up-Ansatz**

- Modellierung der externen Schemata und anschließende Integration der externen Schemata zu einem konzeptionellen Schema.
- Hierbei müssen i.d.R. Widersprüche und Konflikte zwischen einzelnen externen Sichten aufgelöst werden.

- Intuitiver und (automatisch) öfter verwendet: Top-Down-Ansatz

3. Logischer Entwurf

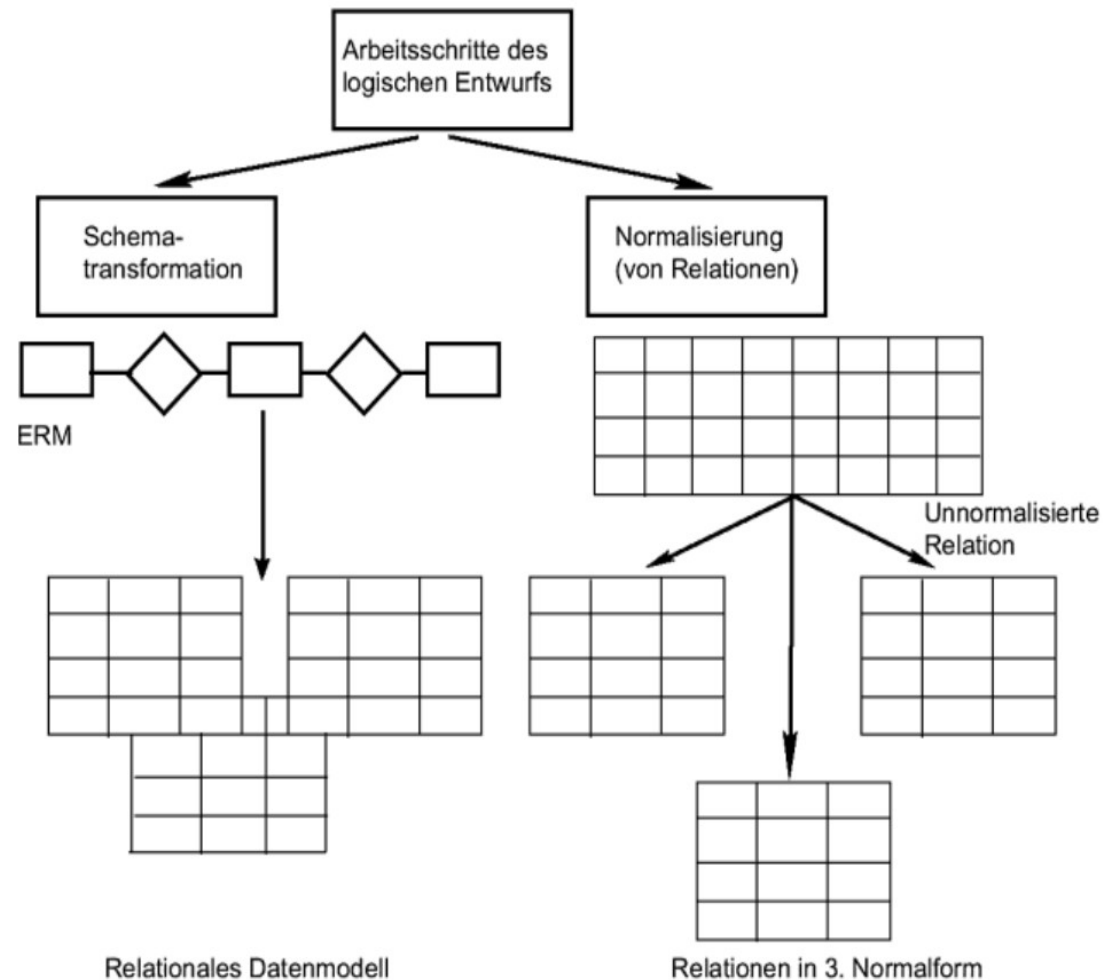
Das logische Schema beschreibt die Datenstrukturen des konzeptionellen Modells in **Abhängigkeit vom Datenbankmodell** des DBMS

Teilschritte

- Entscheidung für verwendetes DBMS oder mindestens für ein Datenbankmodell
- Transformation des konzeptionellen Modells in Abhängigkeit der Anforderungen des Datenbankmodells
- Optimierung des Modells durch Vermeidung von Redundanzen im Rahmen der **Normalisierung**
 - Evtl. Zielkonflikt mit Zugriffsgeschwindigkeit

3. Logischer Entwurf

Arbeitsschritte graphisch (hier: Relationales Modell)



4. Datendefinition

Umsetzung des logischen Schemas und der externen Schemata mit Hilfe der Datenbanksprache.

- Logisches Modell wird mit **Data Definition Language (DDL)** definiert
 - Festlegung von Attributen und Wertebereichen
 - Festlegung von Integritätsbedingungen z.B. durch Primär- und Fremdschlüssel
- Externe Schemata werden mit **View Definition Language (VDL)** definiert
 - Manchmal auch Data Manipulation Language (DML) genannt
 - Anzeigen, Ändern, Löschen von Daten

4. Datendefinition

Umsetzung des logischen Schemas und der externen Schemata mit Hilfe der **Datenbanksprache**.

- Diese Datenbanksprache ist i.d.R. SQL (bei relationalen DB)
- **Structured Query Language (SQL)**
 - Beinhaltet DDL (z.B. Definition von Tabellen)
 - Beinhaltet auch VDL (z.B. Abfragen)
- SQL ist standardisiert, allerdings gibt es Dialekte je nach DBMS
- Datendefinition ist vom DBMS abhängig

Aufgaben

Bitte bearbeiten Sie jetzt die Aufgaben in Moodle zum Kapitel 1.