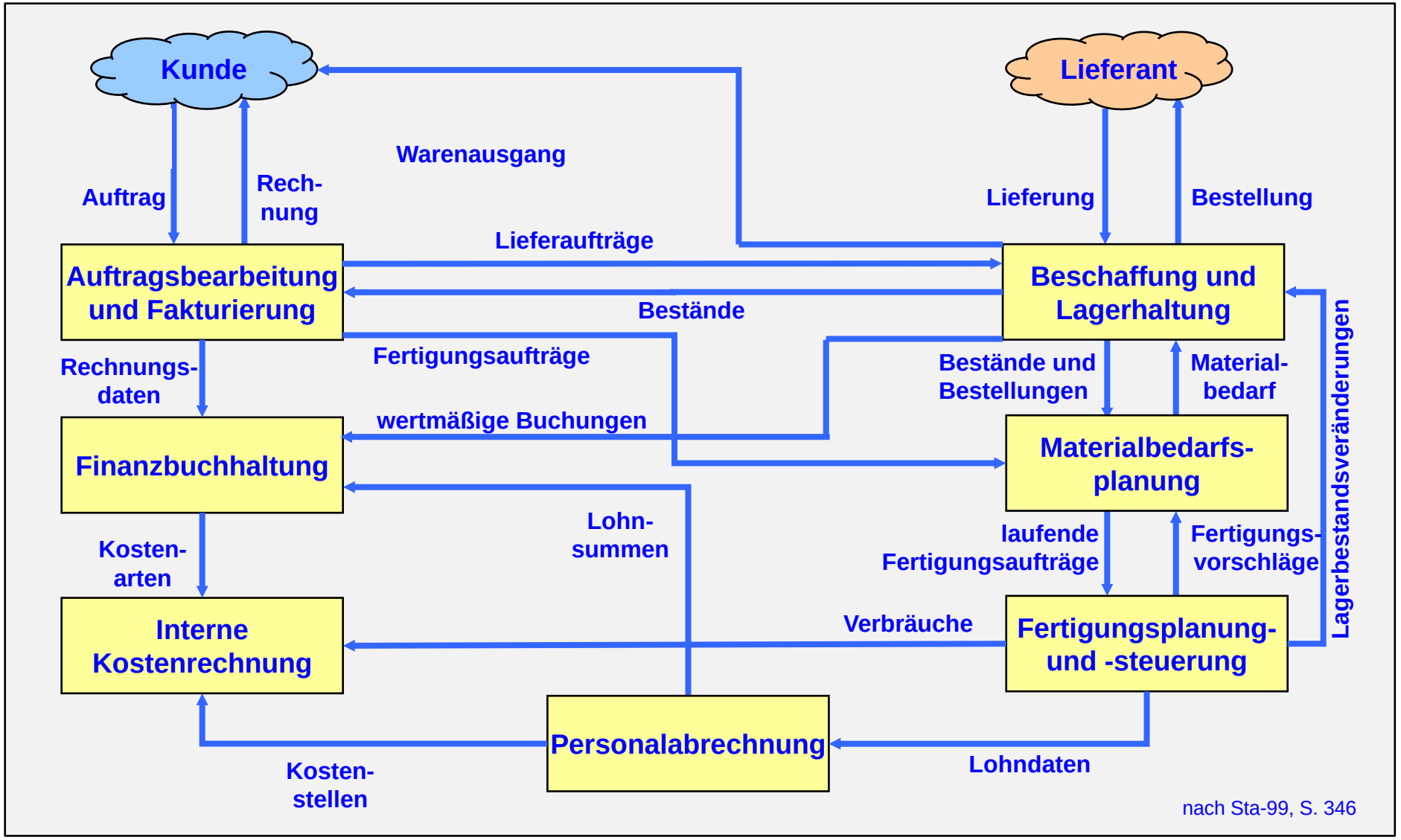
Kapitel 1: Inhaltsübersicht

- Im Anschluss an **grundlegende Begriffsdefinitionen** wird die Geschichte der Datenverwaltung grob skizziert, wobei **Notwendigkeit und Vorteile des datenbank-orientierten Konzeptes** im Vordergrund stehen.
- Im folgenden Abschnitt werden die wichtigsten kommerziell genutzten **Datenmodelle** kurz vorgestellt.
- Abschließend wird die sog. **3-Ebenen-Architektur** von Datenbanksystemen nach ANSI/SPARC behandelt.

Betriebliche Informationsflüsse im Überblick



nach Sta-99, S. 346

Klassifikation von Daten (1)

- **Stammdaten** sind Daten, die der Identifizierung, Klassifizierung und Charakterisierung von Sachverhalten dienen und die unverändert über einen längeren Zeitraum hinweg zur Verfügung stehen.
- **Änderungsdaten** sind abwicklungsorientierte Daten, die fallweise eine Änderung von Stammdaten auslösen.
- **Bestandsdaten** sind Daten, welche die betriebliche Mengen- und Wertestruktur kennzeichnen. Sie unterliegen durch das Betriebsgeschehen einer permanenten Änderung.
- **Bewegungsdaten** sind abwicklungsorientierte Daten, die immer wieder neu durch die betrieblichen Leistungsprozesse entstehen und dabei die Veränderung von Bestandsdaten bewirken.

Klassifikation von Daten (2)

- **Unstrukturierte Daten** unterliegen keinem festen Format, so dass die enthaltenen Informationen nur mit „nicht unerheblicher“ Intelligenz extrahiert werden können.
  - Beispiel: _____
- **Semistrukturierte Daten** haben eine vorgegebene Anordnung, die jedoch nicht verbindlich erzwungen werden kann.
  - Beispiel: _____
- **Strukturierte Daten** haben eine fest vorgegebene Anordnung, deren Einhaltung systemseitig überwacht wird. Hierfür muss die Bedeutung der Daten (Semantik) beschrieben werden.
  - Beispiel: _____

Die Anfänge der Datenverwaltung

- **Frühgeschichte (bis ca. 1960)**
  - Vereinzelt Speicherung auf nichtmagnetischen Medien (insbesondere Lochkarten), was einen rein sequenziellen Zugriff erzwingt.
  - Ausschließlich programmgesteuerte Batchbearbeitung.
- **Traditionelle Datenverwaltung in Dateien (bis ca. 1965)**
  - Große Datenmengen werden auf Magnetband gespeichert, vereinzelt auch magnetische Platten- und Trommelspeicher im Einsatz.
  - Datensätze werden als anwendungsspezifische Dateien gespeichert.
  - Zugriffsmechanismen werden in die Anwendungen eingebettet.
  - Erster Dialogbetrieb für den Datenzugriff.

Folgen der Datenverwaltung mittels Dateien

- Jeder Programmierer baut seine Dateien selbst auf und zwar unabhängig – oft sogar ohne Kenntnis – von den Dateien anderer Programmierer.
- Der logische Dateiaufbau ist unmittelbar an die jeweilige Aufgabenstellung angepasst und in dieser Form auch als Datei physisch gespeichert.
- Jede betriebliche Anwendung greift auf ihren eigenen Datenbestand zu.
- Aus einer isolierten Sicht heraus betrachtet, führt dieser Ansatz zu einer Datenhaltung, die für die jeweilige Anwendung optimiert ist. Mögliche Synergieen werden aber nicht genutzt.

Die ersten Datenbanken

- **Trennung von Anwendungslogik und Datenverwaltung (bis ca. 1970)**
  - Befehle für den Datenzugriff werden aus den Anwendungsprogrammen entfernt und als Routinen von Dateisystemen oder datenorientierten Programmiersprachen wie Cobol angeboten.
  - Die reine Datenverwaltung gewinnt gegenüber der Datenverarbeitung zunehmend an Bedeutung.
- **Dateiverwaltung mittels Data Dictionary (bis ca. 1980)**
  - Zusätzlich wird ein Katalog mit Informationen über die in den Dateien gespeicherten Datenobjekte angelegt, d.h. Metadaten generiert.
  - Hierarchische und netzwerkorientierte Datenbanksysteme im Einsatz.

Forderung an eine effiziente Datenverwaltung

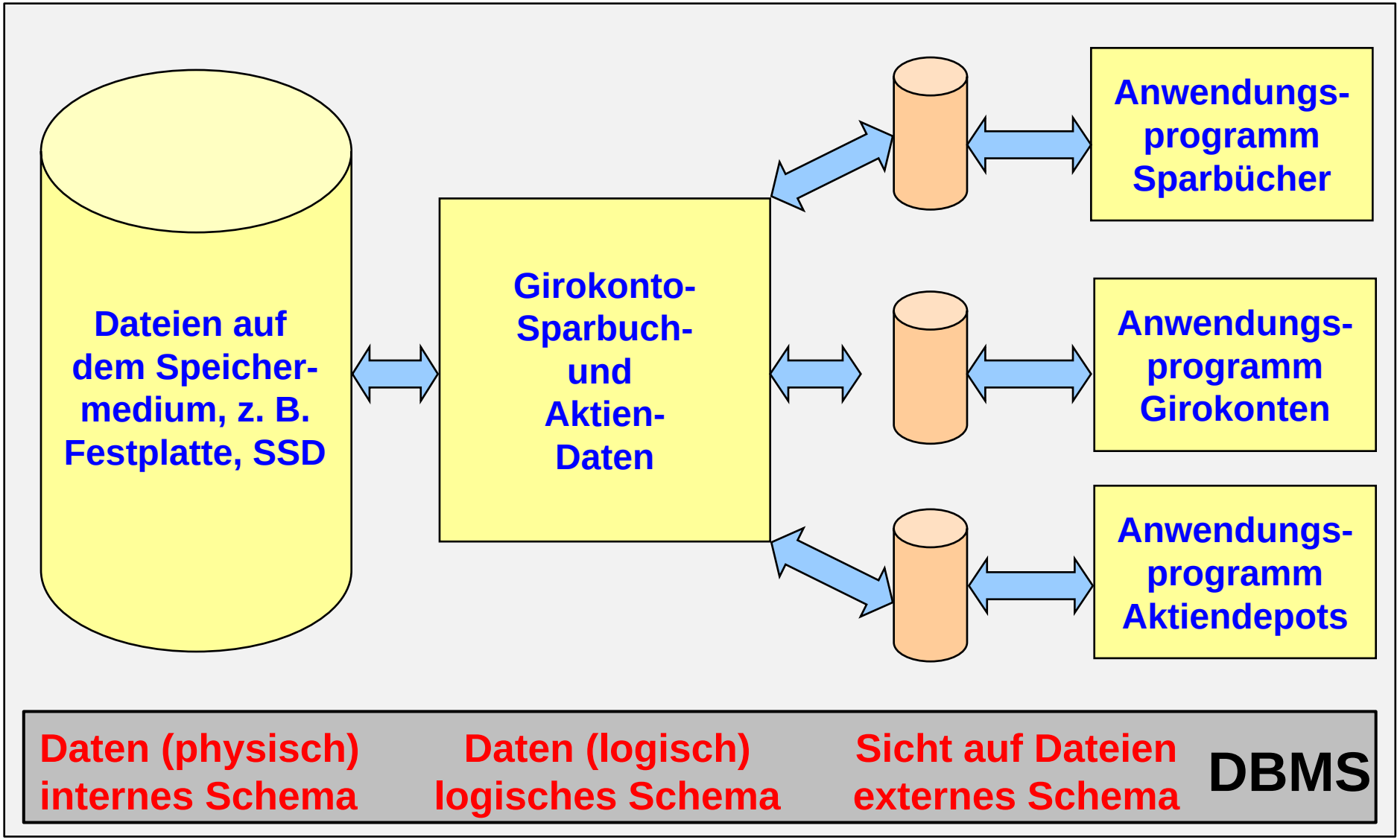
- **Physische Datenunabhängigkeit**

  Anwendungsprogramme sollen die physische Organisation der Daten „nicht sehen“, so dass Änderungen in der Datenspeicherung nicht zwangsläufig umfangreiche Programm-anpassungen nach sich ziehen.

- **Logische Datenunabhängigkeit**

  Die „logische Sicht“ eines Programms auf die von ihm benötigten Daten soll unabhängig von den Veränderungen im Datenbestand sein, die sich durch neue Anforderungen im Unternehmen ergeben.

Datenverwaltung durch ein DBMS



Datenbankkonzepte

- **Einsatz von Datenbankmanagementsystemen (ab 1980)**
  - Alle benötigten Daten werden in einer Datenbank zentral gespeichert.
  - Das Datenbankmanagementsystem (DBMS) trennt die physische von der logischen Ebene. D.h. unabhängig von der tatsächlichen Dateiorganisation auf dem physikalischen Speichermedium wird den Anwendungsprogrammen eine logische Sicht auf die Daten präsentiert.
  - Schnelle Verbreitung von relationalen Datenbanken; hierarchische und netzwerkorientierte Datenbanken verschwinden vom Markt.
- **Neuere Datenbankkonzepte (ab ca. 1990)**
  - Neuere Ansätze, wie z.B. objektorientierte und deduktive Datenbanken konnten die Dominanz der relationalen Datenbanken nicht verhindern.
  - Sukzessive wurden und werden diese Ansätze jedoch von den Anbietern relationaler DBMS integriert.

Vorteile des datenbankorientierten Konzeptes (1)

1. Vermeidung von Redundanz

- Informationen über ein Objekt (z.B. Mitarbeiter, Kunde) werden nur an einer einzigen Stelle gespeichert, wodurch Speicherplatz gespart wird.
- Änderungen müssen nur noch an einer Stelle durchgeführt werden.

2. Verhinderung von Inkonsistenz

- Es existieren keine Duplikate mehr, nur weil zwei Programme die gleichen Informationen jeweils anders strukturiert benötigen.
  Folge: Duplikate werden bei Änderungen nicht zufällig „vergessen“.

3. Datensicherheit

- Transaktionskonzept stellt sicher, dass eine Datenbank auch nach einem Systemabsturz wieder in einem konsistenten Zustand ist.
- Ausgereifte Backup-Verfahren verfügbar.

Vorteile des datenbankorientierten Konzeptes (2)

4. Integritätssicherung

- Oft bestehen zwischen Daten logische Zusammenhänge, die bei der traditionellen Dateiverarbeitung nicht berücksichtigt werden können.
  - Zu einem Lieferschein gibt es keine Rechnung.
  - Jeder Mitarbeiter muss genau einer Abteilung zugeordnet sein.
- Das DBMS kann durch zentrale Kontroll- und Prüfroutinen derartige Integritätsverletzungen, sofern überhaupt feststellbar, unterbinden.

5. Datenschutz, besonders seit der DSGVO im Fokus

- Kontrollierter Zugriff auf sensible Daten in einer Granularität, die weit über den Möglichkeiten eines Betriebssystems liegen.

6. Synchronisation im Mehrbenutzerbetrieb

- Datenbank garantiert, dass parallel ausgeführte Benutzeraktivitäten isoliert und konsistent abgearbeitet werden.

Übersicht verschiedener Datenmodelle

- **Hierarchisches** Datenmodell (veraltet)
- **Netzwerkorientiertes** Datenmodell (veraltet)
- **Relationales** Datenmodell (de-facto-Standard)
- **Objektorientiertes** Datenmodell
- **Objektrelationales** Datenmodell (für Spezialanwendungen)
- **Deduktives** Datenmodell (nur im wissenschaftlichen Umfeld)
- **Semantische** Datenmodelle (zur Veranschaulichung)

Hierarchisches Datenmodell (1)

- Wurde bereits 1969 von IBM eingeführt und war das kommerziell erfolgreichste Datenbankmodell der ersten Generation.
- Es entstand aus dem konventionellen Dateisystem zur Speicherung von Dateien mit Sätzen von variabler Länge.
- „Wiederholungsgruppen“ wurden aus den Sätzen herausgelöst und als separate sequenzielle Datei neu gespeichert. Durch diese sog. untergeordneten Satztypen entstand eine Dateienhierarchie.
- In vielen Anwendungen reichten diese sog. 1:n-Beziehungen aus, z.B. Kunde – Auftrag, Auftrag – Position, Flugzeug – Passagier.
- Vorteil des hierarchischen Datenmodells ist die sehr effiziente Programmierung durch die Anlehnung an das Filesystem, Nachteil jedoch die fehlende Möglichkeit, n:m-Beziehungen abzubilden.

Hierarchisches Datenmodell (2)

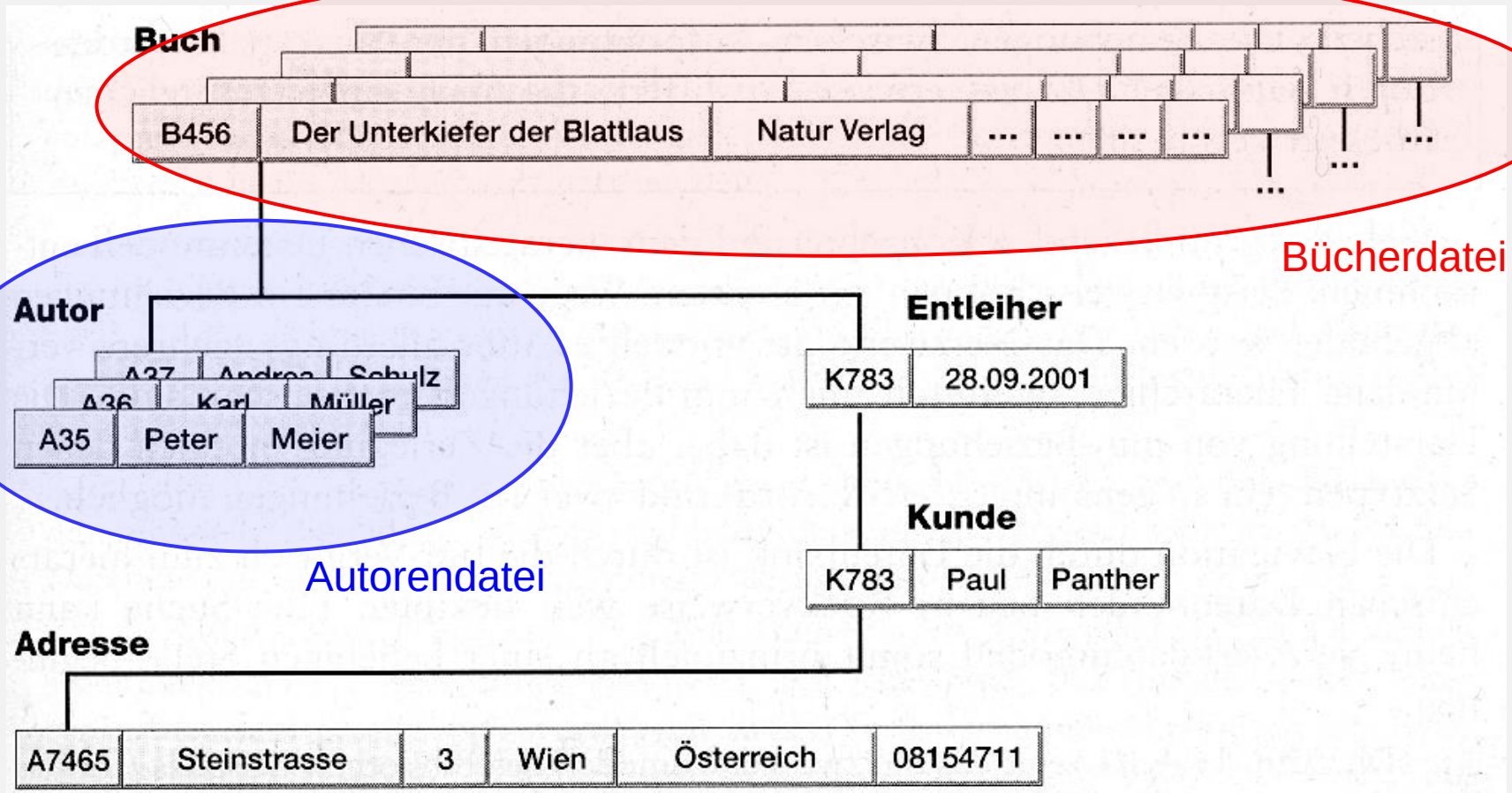


Abb. 11.4.2/2: Beispielausprägung des hierarchischen Datenmodells für die Bibliotheksdatenbank

Quelle: Han-01, S. 1063

Netzwerkdatenmodell (1)

- Das Netzwerkmodell ist eine Weiterentwicklung des hierarchischen Datenmodells und wurde 1971 von dem Normungsausschuss CODASYL definiert.
- Datenbanksysteme nach dem Netzwerkmodell stellen gerichtete Graphen dar, wobei die Knoten des Graphen den Satztypen (Records) entsprechen und die Pfeile die Beziehungen zwischen den Records darstellen.
- Eine Suche muss im Gegensatz zum hierarchischen Modell nicht zwangsläufig an der Wurzel starten.
- Da jeder Knoten prinzipiell mit mehreren anderen Knoten in Beziehung stehen kann, können beim Netzwerkmodell auch n:m-Beziehungen abgebildet werden.

Netzwerkdatenmodell (2)

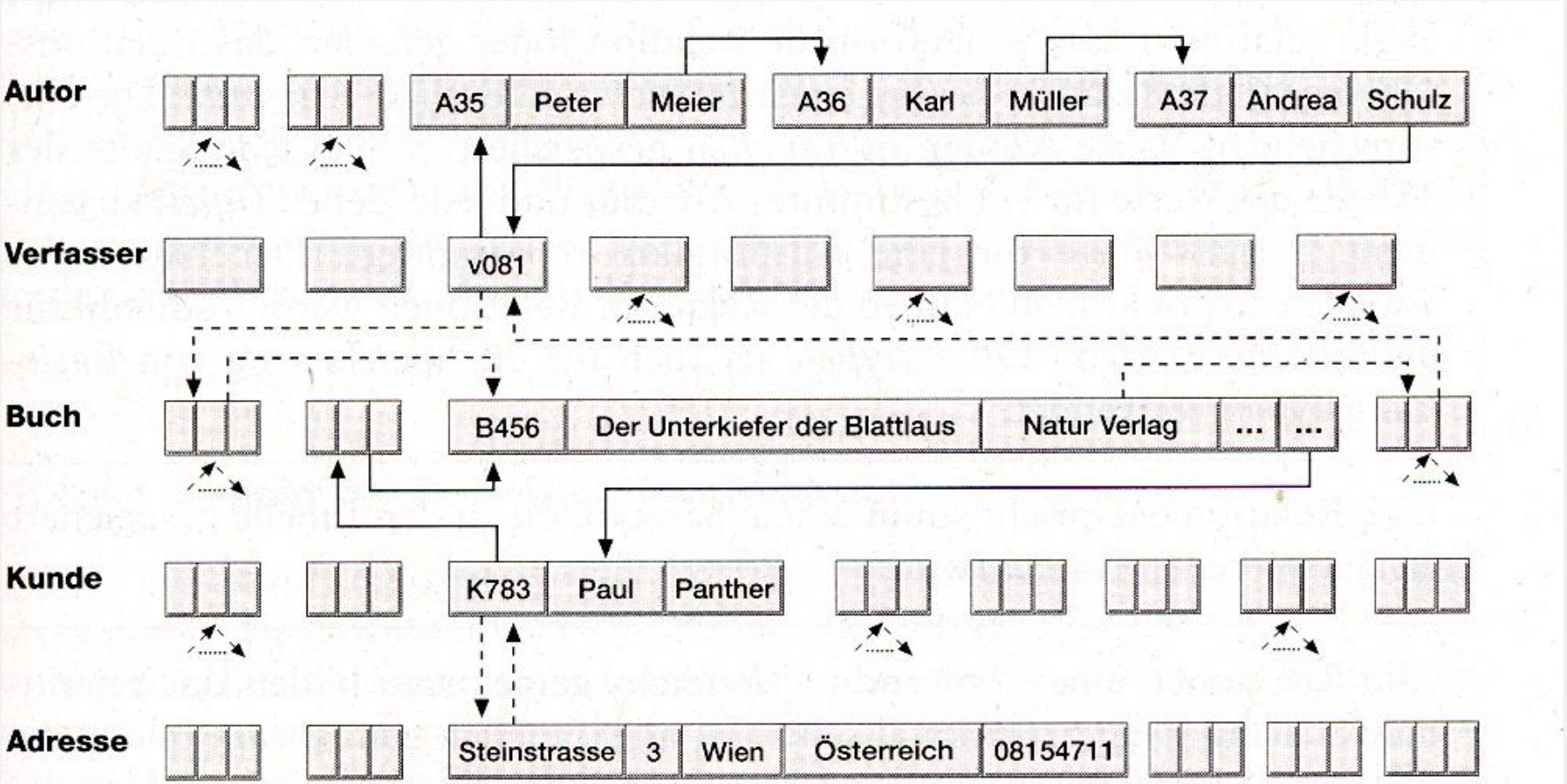


Abb. 11.4.3/2: Beispielausprägung des Netzwerkdatenmodells für die Bibliothek

Quelle: Han-01, S. 1065

Relationales Datenmodell (1)

- Das derzeit in der Praxis mit weitem Abstand vorherrschende Datenmodell ist sowohl für betriebswirtschaftliche als auch administrative Anwendungen optimal geeignet.
- Bekannte „Vertreter“ dieser Gattung sind z.B.
  - ORACLE
  - DB2
  - MS SQL
  - INFOMIX
  - MySQL
  - PostgreSQL
  - Progress
  - (MS-Access)
- Mit SQL (Structured Query Language) gibt es eine weit verbreitete und standardisierte Datendefinitions- und Datenmanipulationssprache.

Relationales Datenmodell (2)

- Die Daten werden in zweidimensionalen Tabellen gespeichert.
- Die Zeilen einer Tabelle repräsentieren untereinander gleichartige Informationseinheiten (Datensätze) und entsprechen (häufig) konkreten Objekten der realen Welt.
- Die Spalten einer Tabelle beschreiben die Eigenschaften bzw. Merkmale dieser Objekte.
- Die Anzahl der Datensätze in einer Tabelle ist beliebig und deren Reihenfolge ohne Bedeutung. Die Reihenfolge der Spalten sollte „sinnvoll“ sein.
- Weniger geeignet ist das relationale Datenmodell für
  - sehr große Objekte, z.B. Multimediaanwendungen.
  - Ingenieurwissenschaftliche Anwendungen mit komplexen Objekten und umfangreichen Operationen auf diesen, z.B. CAD-Lösungen.

Objektorientierte Datenmodelle

- Teilweise wurden die Konzepte der objektorientierten Programmierung auf Datenbanken übertragen, beispielsweise
  - Komplexere Objekte als in zweidimensionalen Tabellen
  - Vererbungsmechanismus mit Generalisierung bzw. Spezialisierung
  - Verhalten von Objekten (Methoden) als integraler Bestandteil der DB
- Die Object Database Management Group, eine Vereinigung von Herstellern objektorientierter Datenbanken, hat mit ODMG-93 (letztes Release 3.0 im Jahr 1999) einen Standard und eine an SQL angelehnte Abfragesprache definiert.
- Einsatz heute in Nischenmärkten für vorwiegend ingenieurwissenschaftliche Anwendungen und im XML-Umfeld.

Objektrelationale Datenbanksysteme

- Im Gegensatz zu den objektorientierten DBS wird diese Entwicklung von den großen DB-Herstellern vorangetrieben und hat den SQL3-Standard erheblich beeinflusst.
- Alle führenden Datenbankhersteller haben in ihren aktuellen Versionen die grundlegenden Konzepte von SQL3 realisiert.
- Prinzip ist eine funktionale Erweiterung des relationalen Datenmodells um Konzepte aus der objektorientierten Datenmodellierung. Hierbei handelt es sich z.B. um
  - Geschachtelte Relationen, d.h. erlaubt sind auch Attribute, die selbst wiederum Relationen sind.
  - Typendeklaration, d.h. Unterstützung der Definition von anwendungsspezifischen Typen (UDTs = user defined types).

Semantische Datenmodelle

- Die bisher dargestellten Datenmodelle haben den Nachteil, dass das „Wissen“ über die reale Welt nicht vollständig darstellbar ist.
  Beispielsweise „kennt“ das relationale Datenmodell nur ein einziges Konstrukt und zwar die zweidimensionale Tabelle.

- Daher wurden **semantische Datenmodelle** mit dem Ziel entwickelt, möglichst viel „Semantik“ anschaulich und dennoch formal – also nicht nur umgangssprachlich – darzustellen.

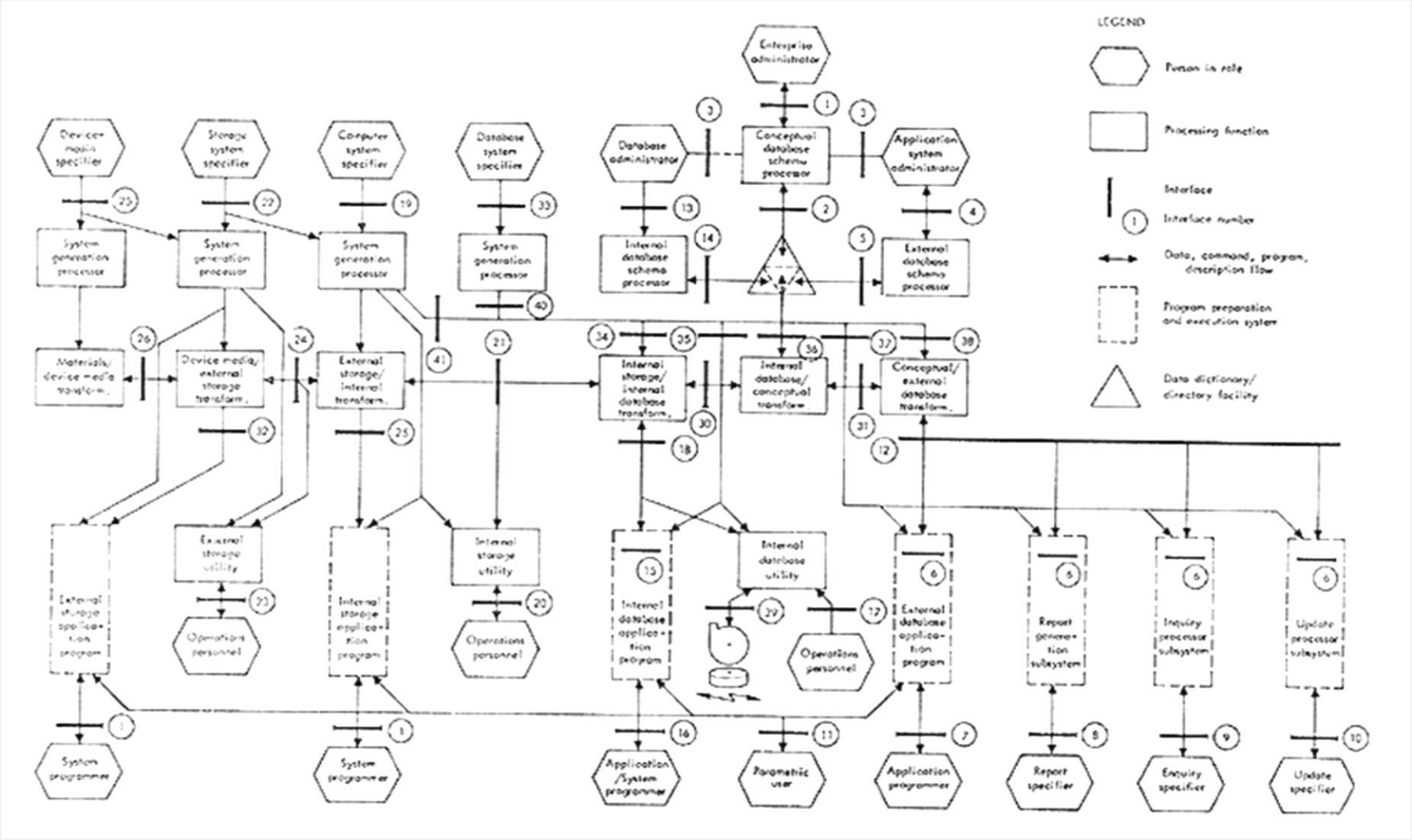
- Typisches Beispiel ist das **Entity-Relationship-Modell**, das im Datenbankentwurf sehr weit verbreitet ist und im Verlauf der Vorlesung noch ausführlicher behandelt wird.

Betrachtungsebenen beim Datenbankentwurf

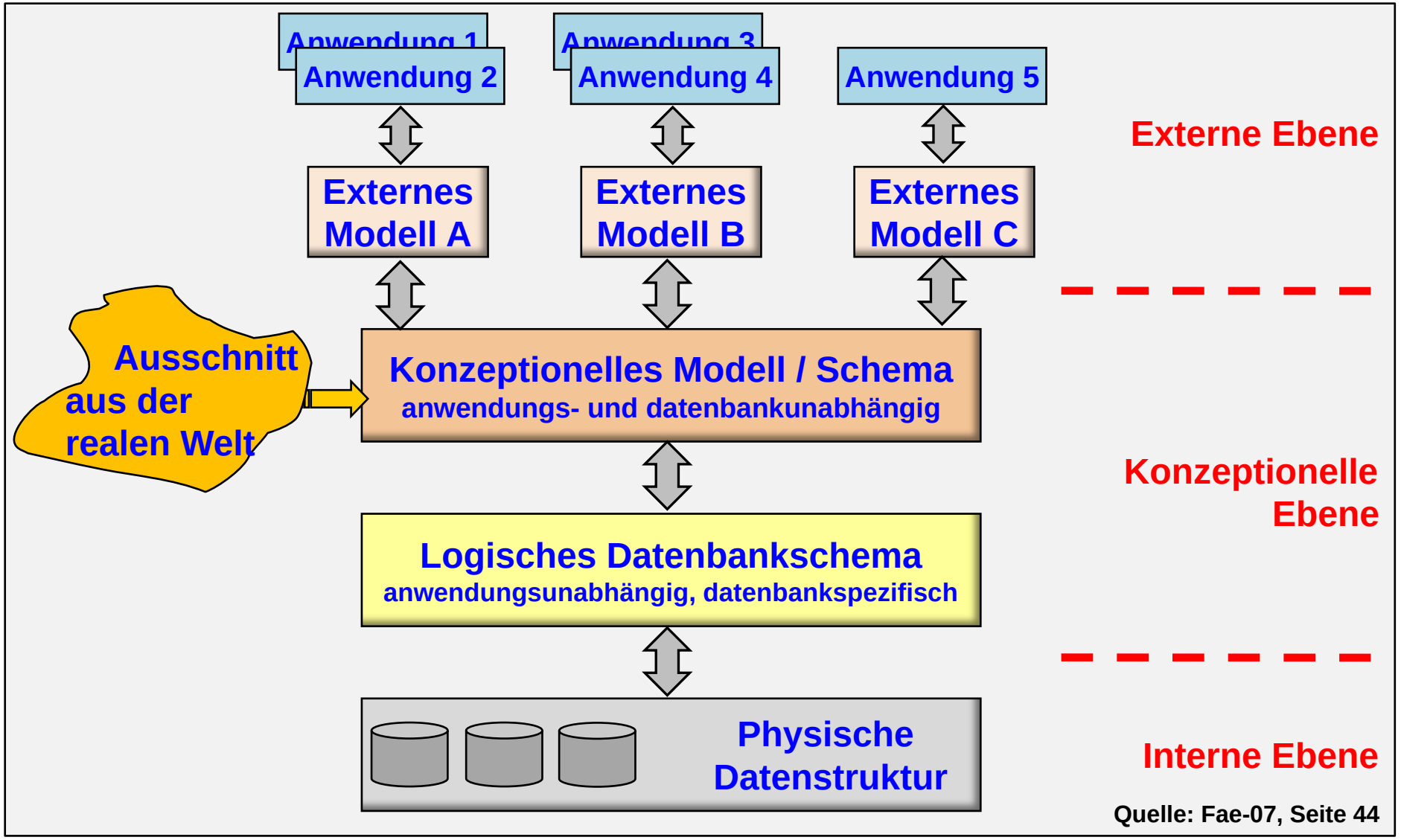
- Für ein systematisches Vorgehen beim Datenbankentwurf ist es essenziell, die Architektur eines Datenbanksystems näher zu betrachten.
- In Anlehnung an die in der Einleitung besprochenen Schemata unterscheidet man **drei Betrachtungsebenen**:
  - **externe Ebene** führt zu den externen Schemata
  - **konzeptionelle Ebene** führt zum konzeptionellen Modell, aus welchem dann das logische Datenbankschema abgeleitet wird
  - **interne (physische) Ebene** führt zum internen Schema
- Bereits im Jahr 1975 verabschiedete ANSI/SPARC* einen Standard für die Architektur von Datenbanksystemen.

* American National Standards Institute bzw. Standards Planning and Requirements Committee

B. Yormark "The ANSI/X3/SPARC/SGDBMS Architecture" in "The ANSI/SPARC DBMS Model", D.A. Jardine (Ed.), North Holland, 1977.



3-Ebenen-Architektur nach ANSI/SPARC



Quelle: Fae-07, Seite 44

Konzeptionelle Ebene

- Das konzeptionelle Modell entsteht aus einer Analyse der „realen Welt“ oder eines Ausschnitts davon durch Abstraktion und Klassenbildung.
- Das konzeptionelle Modell ist unabhängig von einer konkreten DV-Struktur und wird (normalerweise) auch nicht von Datenbankprogrammierern erstellt.
- Es stellt einen langfristigen und stabilen Bezugspunkt darstellt, während relativ häufig neue Anwendungsprogramme hinzukommen oder bestehende Programme modifiziert werden.
- Im weiteren Verlauf des Datenbankentwurfs wird aus dem konzeptionellen Modell das logische Datenbankschema abgeleitet.

Externe Ebene

- Das externe Modell stellt sicher, dass jeder Benutzer bzw. jede Benutzergruppe und jedes Anwendungsprogramm nur die Daten „sieht“, die für die Aufgabenstellung erforderlich sind.
- Es wird eine individuelle Sicht auf die Unternehmensdaten erzeugt, so dass
  - Daten, die nicht gesehen werden sollen und
  - Daten, die nicht gesehen werden wollen,

  automatisch ausgeblendet werden.
- Für die verschiedenen Benutzergruppen werden auf dieser Ebene geeignete Hilfsmittel zur Datenmanipulation bereitgestellt, z.B. SQL für versierte Anwender.

Interne (physische) Ebene

- Im internen Modell wird festgelegt, in welcher Form die Daten im Speicher abgelegt werden und welche Zugriffsmöglichkeiten bestehen. Dies beinhaltet beispielsweise
  - Codierung der Felder (feste oder variable Länge)
  - physische Folge der Sätze und Felder auf dem Speichermedium
  - spezifische Zugriffspfade, z.B. B-Bäume für Sekundärindexe
- Die Beschreibung des internen Schemas erfolgt durch eine DSDL (Data Storage Description Language)
- Grundlage hierfür bilden u.a. statistische Informationen über die Häufigkeit der Zugriffe auf bestimme Objekte, die Reihenfolge von Zugriffen, die Nutzung von Suchroutinen, etc.