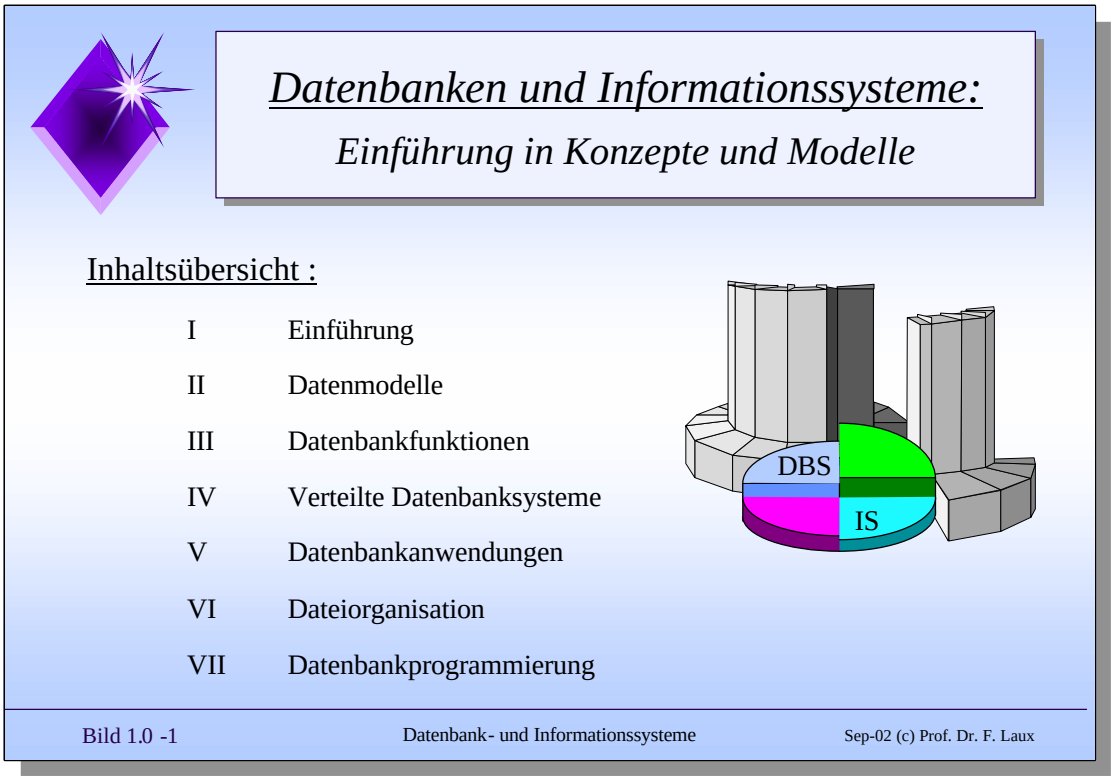


> It is evident that a course in database systems now plays a central role in the undergraduate and graduate programs in computer science.
>
> Jeffrey D. Ullman, Principles of database systems, 1988

Ziel der Vorlesung ist es, eine Einführung in Modelle und Konzepte von Datenbank- und Informationssystemen zu geben.

Die Teilnehmer erlernen methodisches Vorgehen bei der Datenmodellierung und üben den Datenbankentwurf an betrieblichen Aufgabenstellungen. Sie erwerben darüber hinaus solide Kenntnisse in allen grundlegenden Aspekten und Funktionen der Datenbank- und Informationssysteme.

Die Veranstaltung wird von einem Praktikum begleitet, in welchem die behandelten Theorien und Konzepte in die Praxis umgesetzt werden.

Inhalt

Nach einer kurzen Einführung und Motivation des Themas wird eine Rahmenarchitektur für Datenbanksysteme vorgestellt. Der erste Schritt eines Datenbankentwurfs ist die Datenmodellierung, dazu werden in Kap. 2 mehrere Modelle eingeführt und im Praktikum intensiv geübt. In Kap. 3 befassen wir uns mit der Datenbanksprache SQL und den Konzepten der Datenbankfunktionalität, die im folgenden Kapitel um spezifische Aspekte verteilter Datenbanken ergänzt werden. Danach werden die Einsatzgebiete von Datenbanken exemplarisch untersucht, wobei auch neuere Entwicklungen wie Multimediadatenbanken Berücksichtigung finden.

Abgerundet wird die Veranstaltung durch die Betrachtung von gängigen Dateistrukturen und -systemen, die auch als Beispiele für die physische Speicherorganisation bei Datenbanken dienen. Optional werden Konzepte und Beispiele der Datenbankprogrammierung (Embedded SQL, ODBC und JDBC) vorgestellt.

Es gibt eine Vielzahl von guten Büchern, welche Theorie und Anwendung von Datenbanken behandeln. Die o.g. stellen demzufolge nur eine kleine Auswahl dar.

Als Textbuch kann "Database System Concepts" von *Korth* und *Silberschatz* empfohlen werden, da es sich inhaltlich am besten mit der Vorlesung deckt.

Die anderen Literaturangaben sind als Ergänzungs- und Vertiefungslektüre gedacht; sie sind alle in unserer Bibliothek ausleihbar.

Datenmodellierung
Die genannten Bücher führen in die Datenmodellierung ein, sie behandeln jedoch nicht die neuere Entwicklung der objektorientierten Modellbildung (z.B. Unified Modeling Language, Buchempfehlung hierzu: M. Fowler: UML Distilled, 1997, Addison Wesley, auch in dt. Übers. erhältlich).

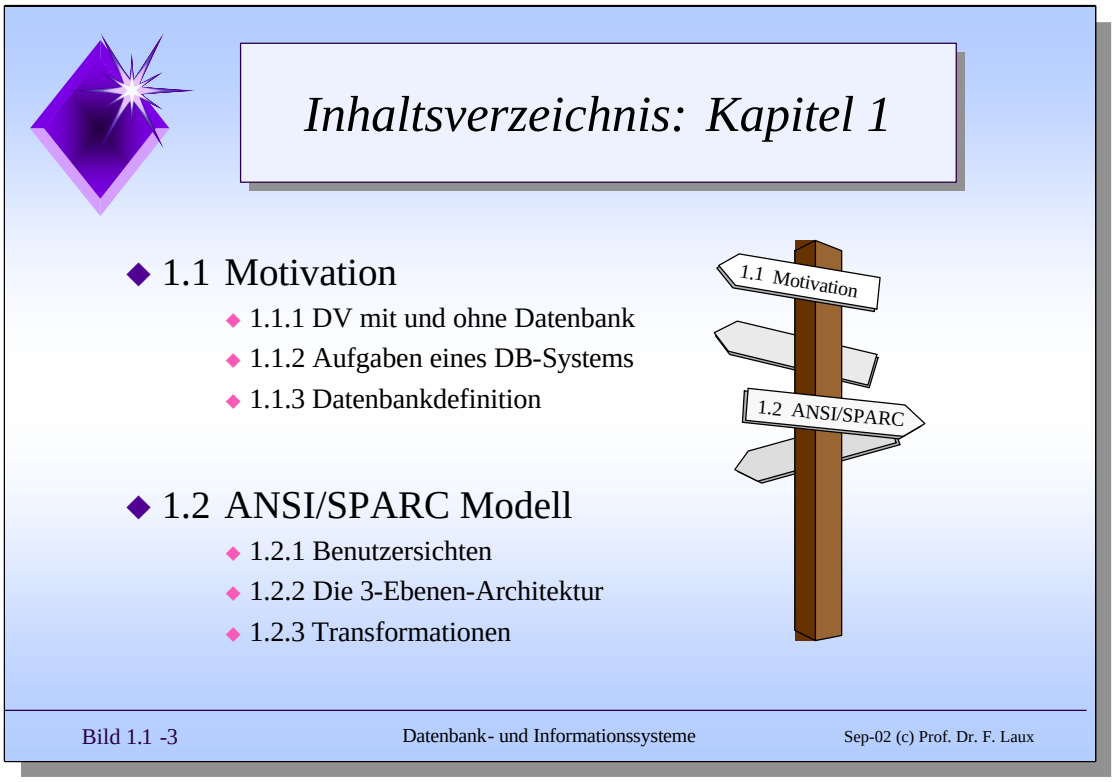
Datenbanksysteme
Die beiden Bände von *Date* sind sehr ausführlich und zählen zu den Klassikern. Das Buch von *Zehnder* ist schon etwas betagt, aber leicht zu lesen und in ca. 15 Exemplaren in der FH-Bibliothek vorhanden.

Verteilte Datenbanken
Die beiden Bücher über verteilte Datenbanken gehen weit über das hinaus, was in der Vorlesung behandelt werden kann.

Anwendungen
Das Buch von *Scheer* über die "Architektur integrierter Informationssysteme" ist für Wirtschaftsinformatiker und im Hinblick auf unseren SAP R/3 Schwerpunkt jedem zu empfehlen.

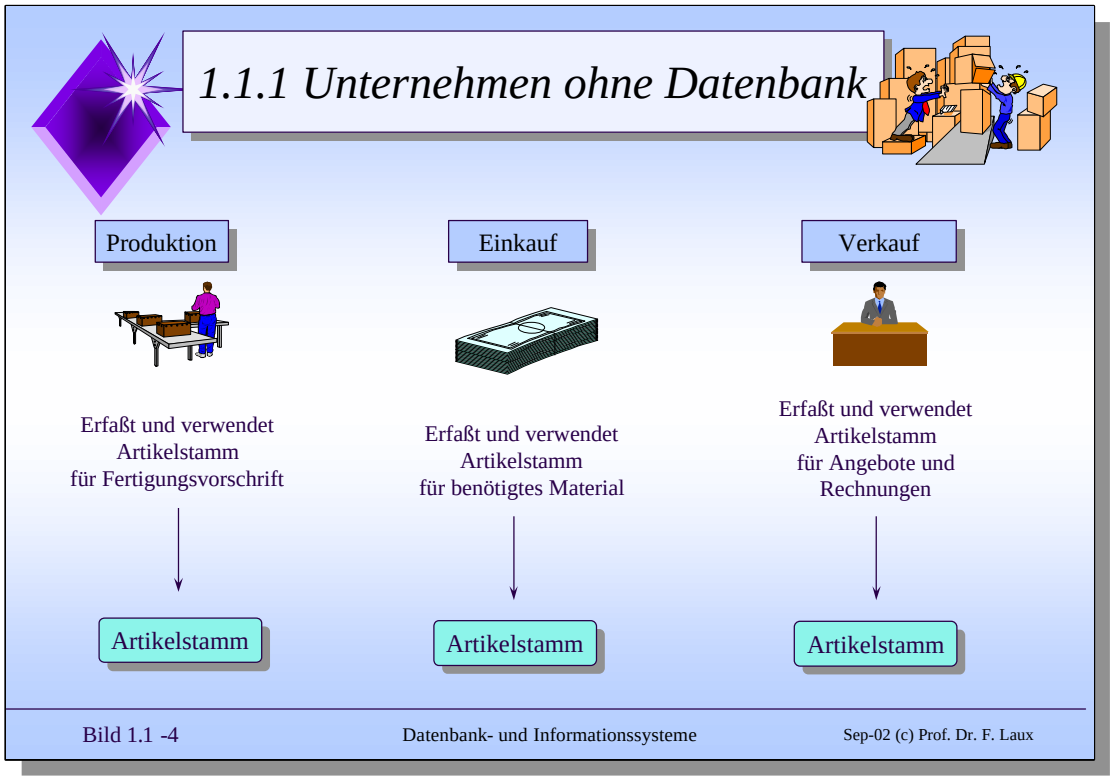
Dateiorganisation
Das Buch von *Wiederhold* vertieft die Themen von Kapitel 6.

DB-Programmierung
Die definitive Referenz über JDBC-Programmierung stammt von den Architekten der JDBC-Schnittstelle selbst.



Zunächst motivieren wir das Vorlesungsthema am Beispiel einer betrieblichen Datenverarbeitung mit und ohne Datenbanksystem (DBS) aus ökonomischer Sicht. Daraus werden technische Eigenschaften und Aufgaben eines DBS abgeleitet. Zum Schluß definieren wir den Begriff DBS mit Hilfe seiner Elemente.

Eine wesentliche Eigenschaft stellt die Datenunabhängigkeit dar. Wir illustrieren dies an einigen Beispielen, die uns zum ANSI/SPARC Modell führen. Dieses richtungsweisende Modell beschreibt eine Architektur zur Realisierung der Datenunabhängigkeit.



Um den Einsatz von Datenbanksystemen besser motivieren zu können, soll die Situation an einem Unternehmen ohne Datenbank untersucht werden.

Betriebliche Informationen unterschiedlichster Art werden in Form von Daten gespeichert:

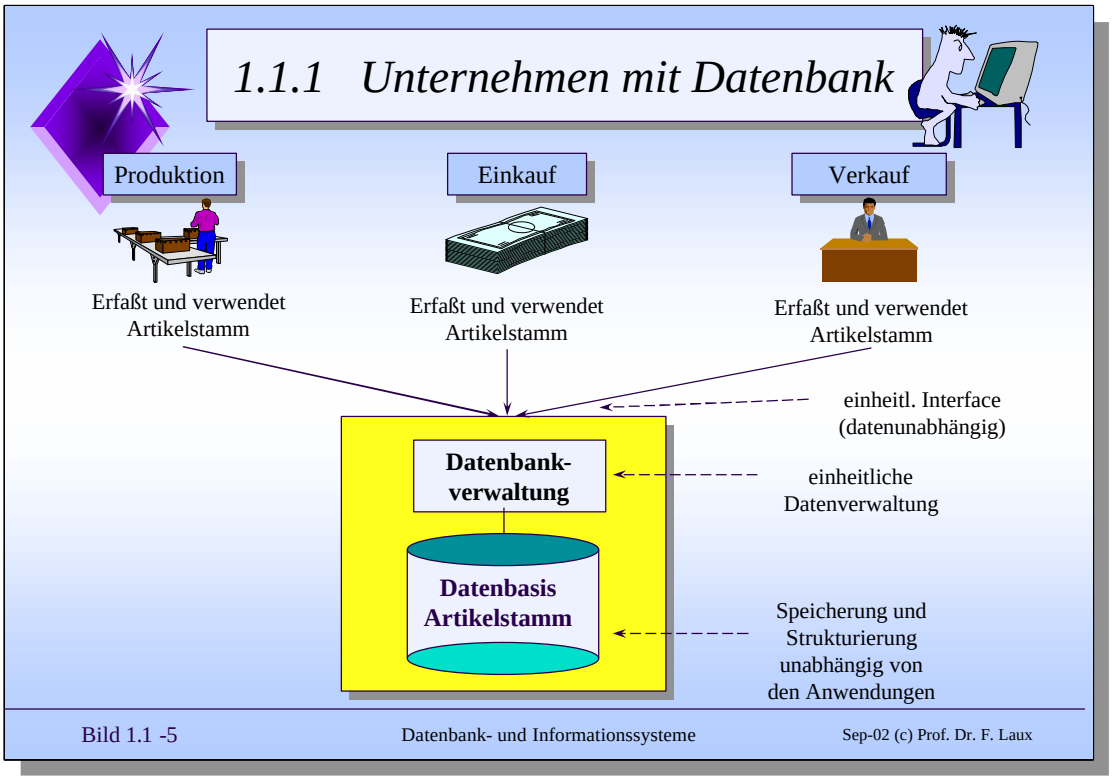
In unserer Folie erfaßt, verwaltet und benützt jeder Bereich einer Unternehmung seine eigenen Artikeldaten. D.h. die Daten sind in mehreren Abteilungen vorhanden, ohne daß bereits erfaßte Daten von anderen Abteilungen genutzt werden können.

Die Folgen der Mehrfacherfassung sind:

- in vielen Dateien sind die selben Informationen gespeichert (-> hohe *Redundanz*)
- eine Aktualisierung oder Änderung der Daten wird möglicherweise nicht in allen Kopien gleichzeitig durchgeführt, d.h. die fachliche (logische) Übereinstimmung der Daten ist nicht gewährleistet (-> *Inkonsistenz* der Daten)
- die Daten sind nur für eine Anforderung (z.B. Abfrage der Artikel) optimal gespeichert, d.h. der Dateiaufbau ist den aktuellen Bedürfnissen angepaßt. Daraus folgt, daß eine Änderung im Dateiaufbau (z.B. ein neues Feld einführen) auch eine Änderung im Programm zur Folge hat (*Datenabhängigkeit* der Anwendung). Das System wird dadurch inflexibel.

Wünschenswert wäre eine *redundanzfreie* und *anwendungsunabhängige* Speicherung der Daten. Der Datenbestand sollte so *strukturiert* sein, daß er *effizient* und *gleichzeitig* von mehreren Anwendern mit unterschiedlichen Anforderungen an die Daten genutzt werden kann.

In der nächsten Folie wird gezeigt, wie durch ein Datenbanksystem die o.g. Forderungen erfüllt und die genannten Nachteile vermieden werden können.



Anstatt die Daten mehrfach zu erfassen und an verschiedenen Orten zu speichern, wird nur einmal in einer zentralen Datenbank (*Datenbasis*) erfaßt. Dies kann durch eine beliebige Abteilung oder auch gemeinsam durchgeführt werden. Trotzdem können die Daten von allen Anwendungsprogrammen gleichzeitig benützt werden.

Die gemeinsame und gleichzeitige Nutzung wird durch eine *Datenbankverwaltung* ermöglicht. Dies ist ein Programm, das die Daten unabhängig von den Anwendungen verwaltet, einen einheitlichen Zugang (*Schnittstelle, Interface*) zu den Daten bereitstellt und konkurrierende Zugriffe regelt.

Der Zugriff erfolgt über eine genormte, anwendungsneutrale Schnittstelle obwohl die Bereitstellung der Daten in einer anwendungsspezifischen Art (Sicht) erfolgt.

Durch die redundanzfreie Speicherung ist sichergestellt, daß alle Anwendungen die aktuellen Daten erhalten.

Die Datenbankverwaltung realisiert gewissermaßen einen komplexen, abstrakten Datentyp: sie schirmt die Daten und ihre Speicherstruktur vor dem direkten Zugriff ab.

1.1.2 Warum Datenbanksysteme ?

- Einheitliche Datenverwaltung
  - Datenintegrität
  - Wiederverwendbarkeit
- Datenunabhängige Programmschnittstelle
  - Unabhängigkeit der Anwendung von der Speicherung
  - Entkopplung der Anwendung von techn. Funktionen (Speicherverwaltung, Mehrbenutzerbetrieb, Zugriffsschutz)

⇩ Kostensenkung u. Qualitätssteigerung bei Progr., Verbesserung d. Flexibilität u. Aktualität d. Daten

Bild 1.1 -6 Datenbank- und Informationssysteme Sep-02 (c) Prof. Dr. F. Laux

Die positiven Eigenschaften eines Datenbanksystems bewirken vor allem eine

- Kostensenkung
- Qualitätsteigerung
- höhere Flexibilität und
- bessere Aktualität der Daten.

Dadurch resultiert insgesamt eine größere Wirtschaftlichkeit der DV im Unternehmen.

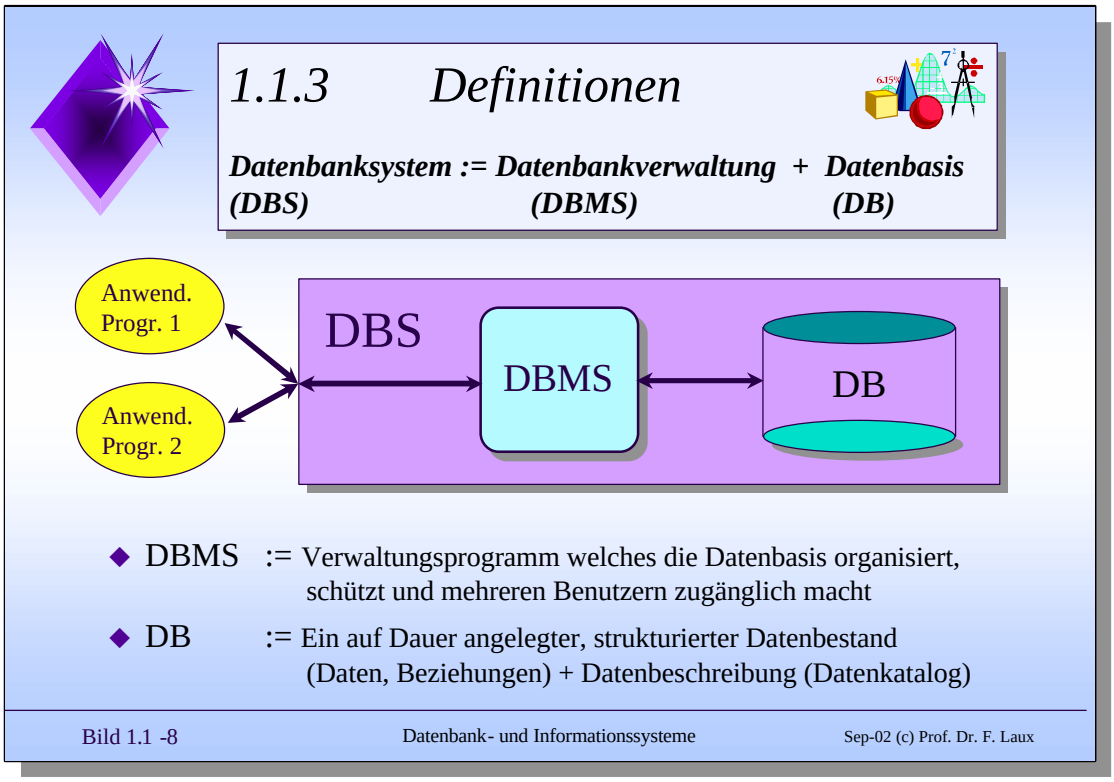
Die Zielsetzung eines DB-Systems läßt sich aus technischer Sicht wie folgt zusammenfassen:

1. höheres Abstraktionsniveau der Daten zu erzeugen (z.B. unabhängig von der Speicher- oder Dateistrukturen)
2. Darstellung der Daten in adäquater (angemessener) und realitätsnaher Form (komplexe, anwendungsspezifische Datentypen).

Eine höhere Abstraktion wird durch das Datenmodell erreicht, welches dem DBS zu Grunde liegt und die möglichen Datenstrukturen festlegt. Man erreicht eine Unabhängigkeit von der physischen Speicherung, indem die Daten für die Präsentation geeignet (realitätsnah) transformiert werden. Die vom Datenmodell vorgegeben Strukturierungsmöglichkeiten unterstützen die Datenkonsistenz indem z.B. Anzahl und Typ einer Beziehung zwischen den Objekten (Bsp. Kunde und Adressen) festgelegt werden.

Konsistenzbedingungen (z.B. der Auftragswert muß größer 0 sein) und Transaktions-mechanismus (z.B. eine Transaktion wird vollständig oder überhaupt nicht ausgeführt) sichern die korrekte Durchführung von Transaktionen (besteht aus einer Folge von Operationen auf einer Datenbank, die als Einheit betrachtet werden).

Datensicherungsverfahren garantieren einen verlustfreien Betrieb des Datenbanksystems.



Definitionen

Datenbasis:

Eine Menge *strukturierter Daten*, ihre *Beziehungen* zueinander und eine *Datenbeschreibung* (Datenkatalog), die unabhängig von einer Anwendung gespeichert sind, heißt ***Datenbasis (DB)***.

Datenkatalog:

Eine *Beschreibung der* in einer Datenbasis speicherbaren *Daten*, ihrer *Beziehungen* und ihre *Verwendung* (Programme) heißt ***Datenkatalog*** (engl. , Data Dictionary, **DD**).

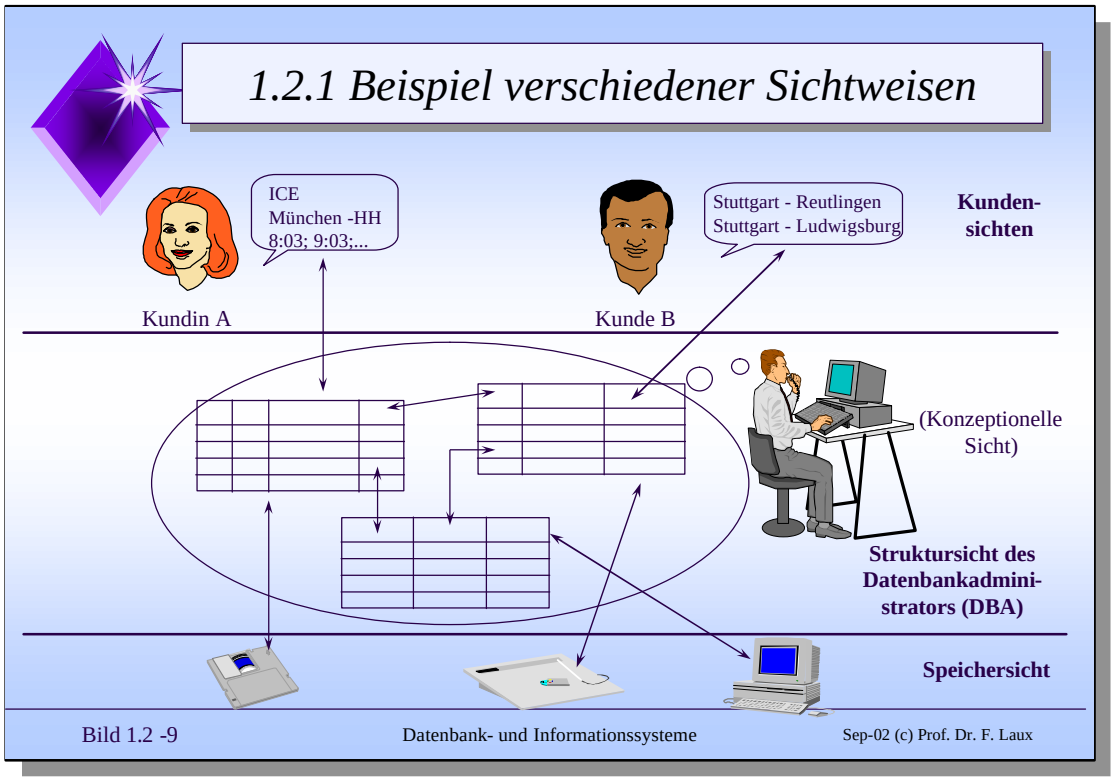
Er stellt damit eine Metadatenbank dar, d.h. eine Informationsquelle über die Datenstrukturen und -formate einer Datenbasis.

Datenbankverwaltung:

Ein Programm zur einheitlichen *Speicherung*, *Verwaltung*, *Recherche* und *Schutz* einer Datenbasis heißt ***Datenbankverwaltung***. (engl. Database Management System, **DBMS**)

Datenbanksystem:

Ein auf Dauer angelegter, strukturierter ***Datenbestand*** (Datenbasis, DB) und ein ***Verwaltungsprogramm*** (Datenbankverwaltungssystem, DBMS (Database Management System), welches diese Daten organisiert, schützt und mehreren Benutzern oder Anwendungsprogrammen zugänglich macht, bilden ein ***Datenbanksystem (DBS)***.



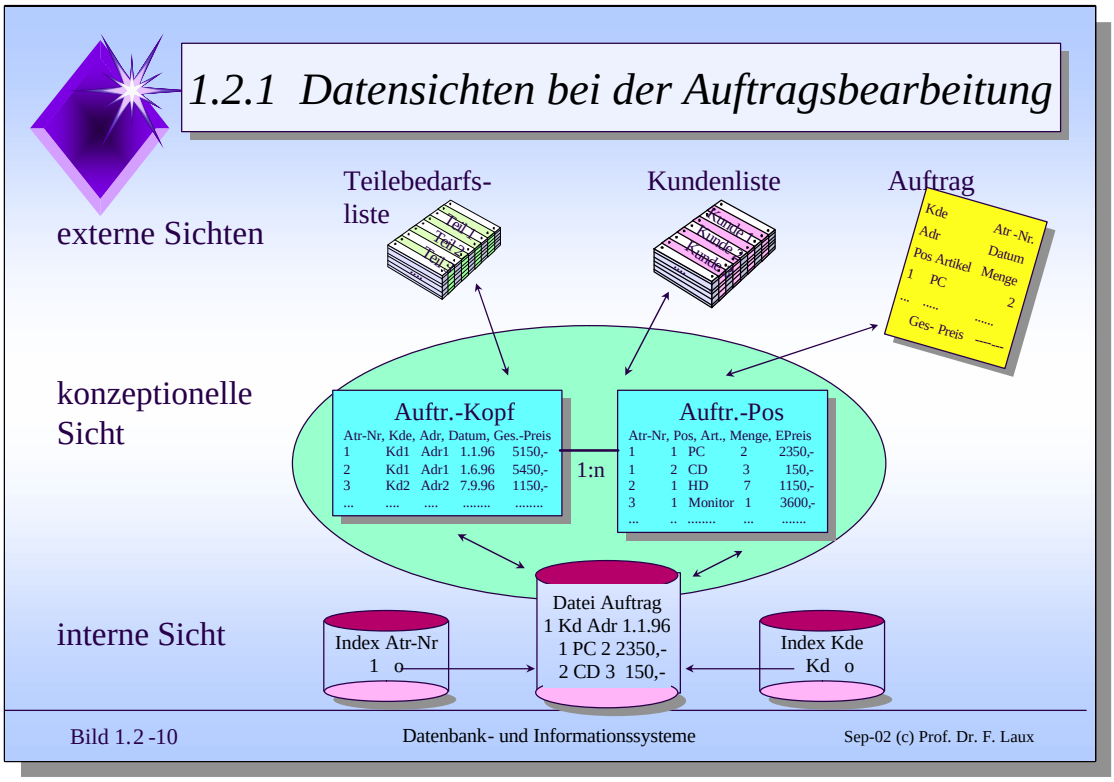
Menschen haben die Fähigkeit, sich je nach Interessenlage oder Aufgabe mit verschiedenen Facetten (Sichten) einer Sache zu beschäftigen. Dies ist an folgendem Beispiel erkennbar:

Ein Mechaniker betrachtet ein Auto aus einem technischen Blickwinkel. Er interessiert sich beispielsweise für die 'Innereien' eines Motors. Ein Fahrer richtet sein Augenmerk auf die Bedienungselemente des Fahrzeugs, während der Halter des Pkws sich eher für die wirtschaftlichen Aspekte interessiert.

In ähnlicher Weise lassen sich betriebliche Vorgänge, Funktionen und Dienste in verschiedener Weise betrachten. Soll nun ein Datenbanksystem als Informationsbasis für einen Betrieb oder einen Teil davon dienen, so muß es diesen unterschiedlichen Anforderungen (Sichten) gerecht werden.

Wenn wir beispielsweise das Kursbuch der Deutschen Bahn AG in einer Datenbank abbilden wollen, müssen ganz unterschiedliche Kundenwünsche gleichzeitig berücksichtigt werden. Außerdem sollten auch zukünftige Anforderungen befriedigt werden können. Um dies zu erreichen, ist eine abstraktere Sichtweise (konzeptionelle Sicht) erforderlich, als die der Benutzer. Die optimale physische Organisation der Datenspeicher hingegen ist von der Häufigkeit der Zugriffe abhängig, kann also sowohl von der konzeptionellen wie auch der Benutzersicht abweichen.
Diese Überlegungen führen uns zu drei Sichtweisen: den Benutzersichten, der konzeptionellen Sicht (Struktursicht des Datenbankadministrators) und der Speichersicht.

In der nächsten Folie vertiefen wir diese Erkenntnis aus Sicht der Informatik.



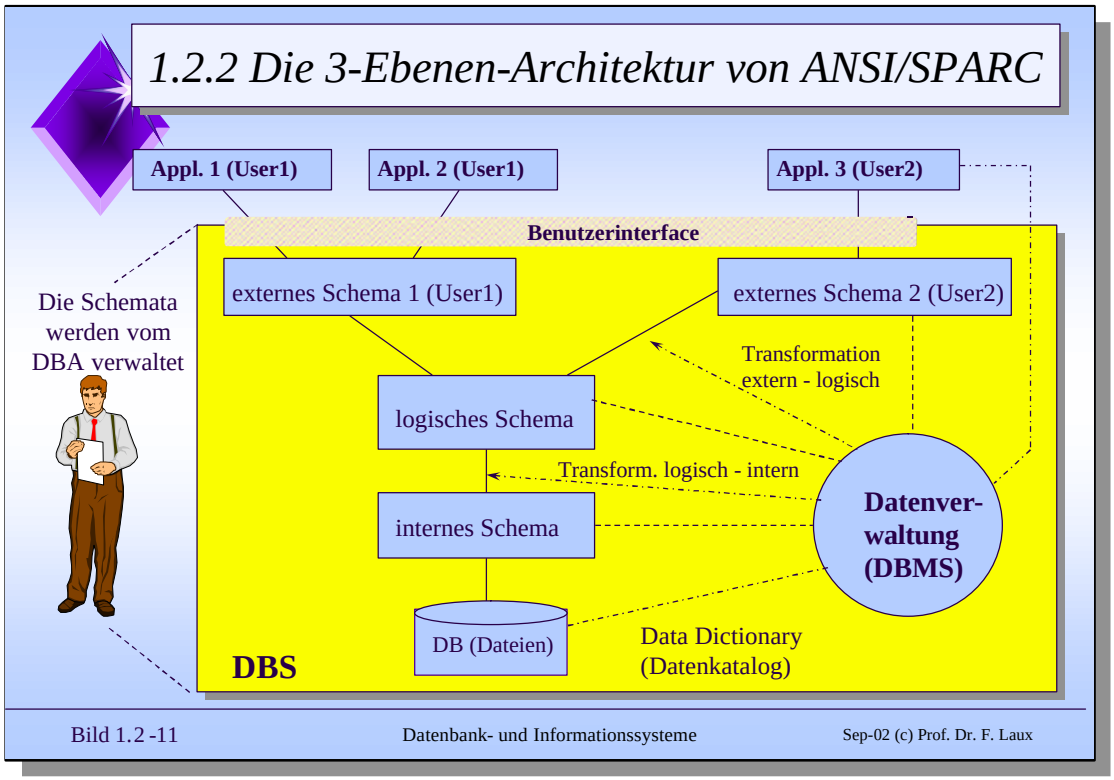
Betrachten wir das Standardbeispiel 'Auftragsbearbeitung'.

Wir nehmen an, daß unsere Datenbasis für die Auftragsbearbeitung tabellarisch strukturiert ist. Dies ist eine mögliche Abstraktion (konzeptionelle [auch konzeptuelle] Sicht) der in einem Auftrag enthalten Information. Sie bedeutet, daß wir wenigstens zwei Tabellenstrukturen benötigen, da Auftragskopf und Auftragspositionen verschieden strukturiert sind. Beide Tabellen stehen durch den Zusammenhang zwischen Auftragskopf und seinen Auftragspositionen (i.a. mehrere Positionen pro Auftragskopf) in Beziehung miteinander.

Neben den Auftragspapieren werden aus diesen Daten auch Kunden- und Teilebedarfslisten erstellt. Diese sind offensichtlich anders strukturiert als unsere konzeptionelle Sicht, obwohl sie sich aus unserer Datenbasis ableiten lassen.

Da der häufigste Zugriff in Form des Auftrags erfolgt, sollen diese Daten auch zusammen gespeichert werden. Um einen effizienten Zugriff auf den Datenbestand nicht nur in zeitlicher Reihenfolge der Erfassung zu ermöglichen, werden Indizes für die Kunden- und Auftragsnummer mitgeführt.

An diesem Beispiel sehen wir, daß die Strukturen der 3 Sichten erheblich voneinander abweichen können. Es kann offenbar nicht das Datenmodell geben, welches alle möglichen Anforderungen optimal erfüllt. Umso wichtiger ist es deshalb, daß ein DBS verschiedene Sichten (Abstraktionsebenen) realisiert, um die notwendige Flexibilität für unterschiedliche Anforderungen zu erhalten.



Wenn wir das Beispiel 1.2-10 generalisieren, so erhalten wir eine 3-Ebenen-Architektur welche schon 1975 im ANSI/SPARC Modell seinen Ausdruck gefunden hat.

Das **Modell von ANSI/SPARC** (American National Standards Institute / Standards Planning And Requirements Committee) definiert 3 Funktionsschichten für ein DBS. Diese Schichten stellen **Abstraktionsebenen** zwischen dem Speichermedium und dem Benutzer (oder Anwendungsprogramm) dar.

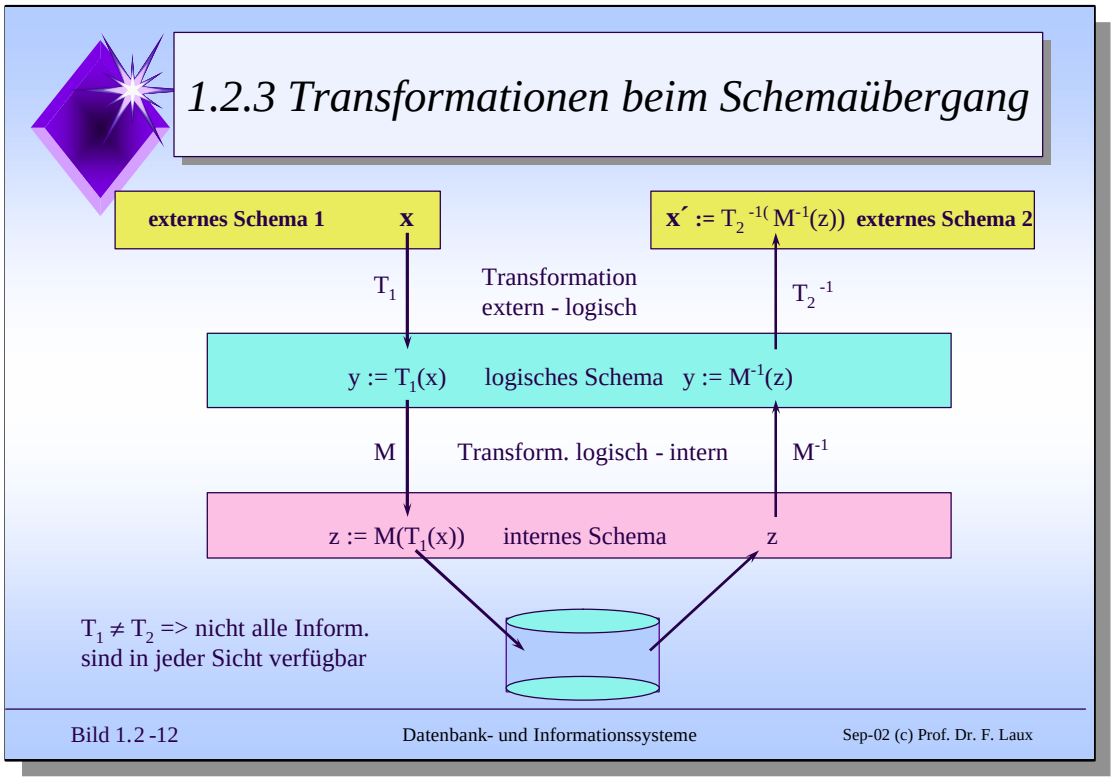
Die externe Ebene präsentiert die Daten dem Benutzer in der gewünschten Form. Im Extremfall besitzt jede Anwendung und jeder Benutzer eine individuelle Sicht auf die Daten. Diese Sichten werden durch **externe Schemata** definiert.

Die konzeptionelle Gesamtsicht der Datenbank, d.h. die gesamte Datenstruktur gemäß einem vorgegebenen Datenmodell, wird im **logischen Schema** durch den ***Datenbankadministrator (DBA)*** festgelegt. Physische Speichereigenschaften oder individuelle Sichten (Anwendungsstrukturen) werden nicht hier berücksichtigt.

Das interne Schema schließlich beschreibt die physischen Speicherstrukturen auf dem Massenspeicher (Festplatte o.ä.).

Mit Hilfe von Transformationen werden die Sichten von einer Ebene in eine andere überführt. Dies bedeutet, daß bei jedem Datenzugriff zwei Transformationen durchlaufen werden(vgl. Bild 1.2-12). Damit kann sowohl die **physische Datenunabhängigkeit** (Unabhängigkeit der externen und logischen Datenstrukturen von der Speicherung) als auch die **logische Datenunabhängigkeit** (Unabhängigkeit der externen Datenstrukturen von der logischen Struktur) realisiert werden. Dies bedeutet, daß sich Schemaänderungen einer Ebene nicht auf eine andere auswirken.

Das Architekturmodell von ANSI/SPARC liefert gewissermaßen den Bauplan, um die Datenunabhängigkeit zu realisieren.



Am Beispiel eines Datenelements x wird gezeigt, welche Umwandlungen die Transformationen bewirken, bis es als Wert z auf dem physischen Speicher der Datenbasis abgelegt werden kann. Die Transformationen Ti sind für die Umsetzung zwischen externem und logischem Schema zuständig; M formt zwischen logischem und internem Schema um. Wird dieses Element x von einer anderen externen Datensicht (Schema 2) gelesen, erscheint es als x' dem Benutzer.

Die inverse Transformation von T wird als T^{-1} bezeichnet. Wenn T1 ungleich T2 ist, dann gibt es für ein Datum x unterschiedliche externe Darstellungen.

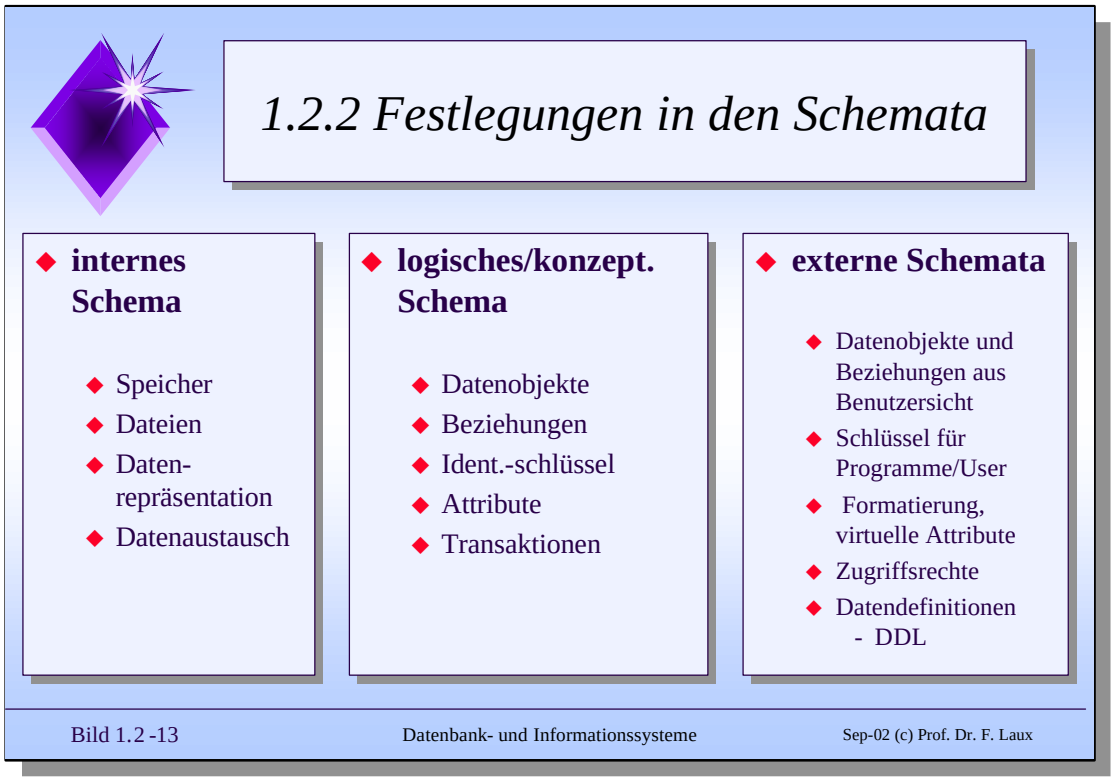
Beispiel: x := ‚gelb‘,

y := $T_1(x)$ = RGB-Farbcode (0,1,1)

z := M(y) = 24 Bit (00FFFF)

x‘ := $T_2^{-1}(y)$ = ‚jaune‘

x‘‘ := $T_3^{-1}(y)$ =



Die folgenden Festlegungen, die in den 3 Ebenen getroffen werden, sollen nach ANSI/SPARC jeweils in einer eigenen **Datendefinitionssprache** (**DDL**) formuliert werden. Bei SQL-Datenbanken (-> siehe Kap. 2.3 und 3.1) erfolgt die Datendefinition für alle Bereiche in SQL; bei Netzwerkdatenbanken nach dem CODASYL-Standard (Conference On Data Systems Languages) wird für jede Ebene eine eigene Sprache verwendet.

Das **interne Schema** definiert die physischen Aspekte und die Speicherung der Daten:

- Speicher: Speichermedium, -bereich, -partition, -reservierung
- Dateien: Dateiorganisation (index-sequentiell, gestreut (hash), relativ) -> siehe Kap. 6, Blockgröße, Ladefaktor, Hashfunktion, Sekundärorganisation/Zugriffshilfen (Indizes, invertierte Dateien, Multiliststrukturen)
- Datenrepräsentation:
  Satzdefinition: Struktur, Datentypen
  Felddefinition: Genauigkeit (Vor-, Nachkommastellen), Code, Kompression, Chiffrierung

Datenaustausch: Transfergrößen zwischen Haupt- und Massenspeicher, Speicherverwaltung (LRU=Least Recently Used, zyklisch, feste Reservierung)

Das **logische Schema** definiert die globale, fachlich abstrakte Datenstruktur:

- Datenobjekte: Typ der Entitäten/Objekte
- Beziehungen: Beziehungen zwischen Entitäten/Objekten (1:n, n:m, Aggregat/Komponenten, Generalisierung/Spezialisierung, etc.)

- Schlüssel: Damit jedes Objekt oder jede Entität eindeutig identifizierbar ist, wird eine Objekt-ID oder eine eindeutige Attributkombination festgelegt. Gegebenenfalls muß ein künstlicher Schlüssel/Attribut (Surrogat) eingeführt werden.
- Attribute: Festlegung der Objekteigenschaften durch Attribute (Datentypen), ihr möglicher Wertebereich (Domain), Genauigkeit und Anzahl.
- Modellanpassungen: Anpassungen der Objekte und Entitäten an das verwendete Datenmodell, Reduktion von Redundanzen (Normalisierung).
- Transaktionen: Die im Kontext der Anwendung zulässigen Datenmanipulationen (Operationen, welche die Datenbank konsistent erhalten) werden als unteilbare Vorgänge (Transaktionen) definiert.

Die **externen Schemata** definieren die Benutzersichten. Jedes Schema legt fest:

- Objekte und Beziehungen aus Benutzersicht:

  Objekt- und Beziehungstypen aus Benutzersicht, die aus dem logischen Schema ableitbar sind.
- Schlüssel für Benutzer und Anwendungen:

  Zugriffsschlüssel (Suchschlüssel, alternative Zugriffsarten) der Benutzer.
- Attribute: Vom Anwender definierte Attribute; sie können auch virtuell (z.B. berechnet oder abgeleitet) sein.
- Formatierung: Externe Darstellung der Attributwerte (Daten); z.B. Umsetzung sprachneutraler Attribute in anwendungsspezifische Präsentationen (1 = rot, 2 = gelb, 3 = grün)
- Zugriffsrechte: Lese-, Schreib- und Änderungsrechte für Objekte, Attribute etc.
- Datendefinitionen: Generierung von Datendefinitionen für verschiedene Hostsprachen.

Literatur zu Kapitel 1

- ANSI/X3/SPARC
  "Interim Report from the Study Group DBMS"
  FDT (Bulletin of the ACM SIGMOD*) Vol. 7, No 2 (1975)
- C. J. Date
  "An Introduction to Database Systems", Vol. 1, 6th Edition
  Addison-Wesley (1995)
- Nahezu jedes Lehrbuch über Datenbanksysteme behandelt das ANSI/SPARC-Modell

*)Association for Computing Machinery, Special Interest Group on Management Of Data

Bild 1.2 -15 Datenbank- und Informationssysteme Sep-02 (c) Prof. Dr. F. Laux

Der Zwischenbericht des ANSI/SPARC spiegelt den damaligen Erkenntnisstand wider. Die Arbeitsgruppe (Task Force), welche die Studie durchführte, zeigt ihre Herkunft aus der Zeit der COBOL-Standardisierung (Conference on Data System Languages, CODASYL) dadurch, daß sie auch detaillierte Vorschläge für die Datendefinitionssprachen macht.

Die Monographie von C. J. Date erklärt sehr anschaulich und mit viel Detailwissen alle wichtigen Aspekte von Datenbanken. Es ist vermutlich das erfolgreichste Lehrbuch über Datenbanken.