Datenbank-Systeme - Speichern und Rückgewinnen von Daten

Autor(en): **Zehnder, C.A.**

Objekttyp: **Article**

Zeitschrift: **Nachrichten VSB/SVD = Nouvelles ABS/ASD = Notizie ABS/ASD**

Band (Jahr): **53 (1977)**

Heft 6

PDF erstellt am: **21.07.2024**

Persistenter Link: https://doi.org/10.5169/seals-771440

Datenbank-Systeme — Speichern und Rückgewinnen von Daten

C. A. Zehnder
Institut für Informatik, ETH-Zentrum, 8092 Zürich

Vorbemerkung

Der Computer und die elektronische Datenverarbeitung (EDV) haben heute eine wichtige und zunehmende Bedeutung bei der Lösung von Problemen der Informations-Speicherung und -Rückgewinnung. Bibliotheken und Dokumentationssysteme als klassische Informationsstellen bilden daher ein schönes Anwendungsgebiet der entsprechenden Wissenschaft, der Informatik. Auf der anderen Seite benötigen Dokumentalisten und Bibliothekare ihrerseits vermehrt Grundkenntnisse aus der Datenverarbeitungs-Welt, sei es, weil ihre Großbestände an (Daten-)Material den Computer als Arbeitsgerät brauchen, sei es, weil die Informatik die theoretischen Methoden anbietet, die den großen Datenmengen am ehesten gewachsen sind.

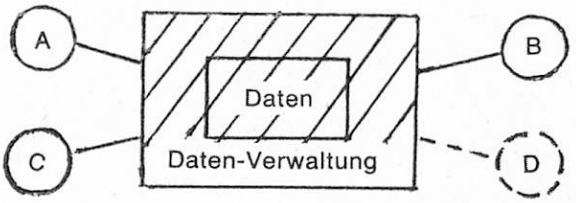
Der nachstehende Artikel beschäftigt sich daher konkret mit solchen Methoden, teilweise angewandt auf Bibliotheks- und Dokumentationsbeispiele. Einige Grundbegriffe der EDV sind unabhängig davon als Anhang beigefügt.

Die benützte Terminologie ist im methodischen Bereich die in der Informatik übliche, bei Anwendungen im Dokumentationsbereich ist deren Terminologie weitgehend berücksichtigt. Insbesondere verstehen wir unter Daten speicherungsfähige Grundelemente des Sachwissens, unter Information Antworten an einen fragenden Menschen.

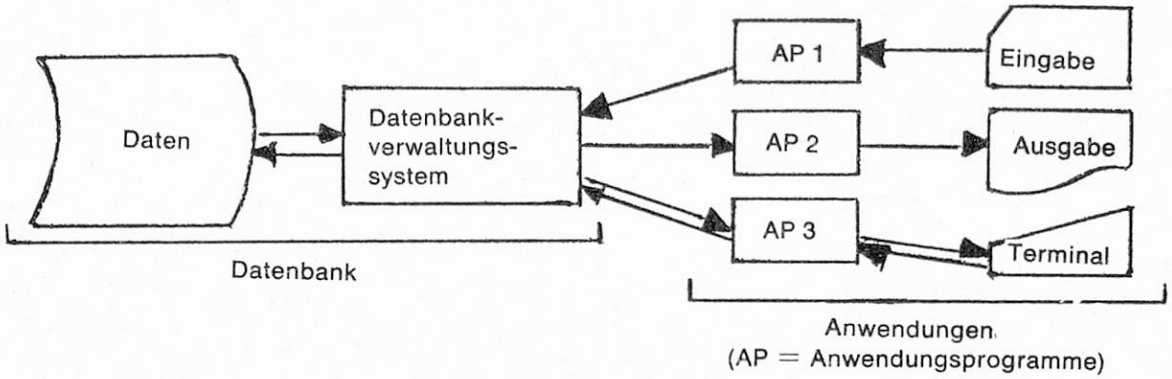
1. Das Datenbank-Konzept

Während in den Anfängen der EDV das programmgesteuerte *Bearbeiten* der Daten – also das Rechnen, Zählen, Schreiben – im Vordergrund stand, wird heute die Bedeutung der *permanenten Datenspeicherung* immer größer. Datenbestände existieren über eine längere Zeit, verschiedene Interessenten benützen sie gemeinsam und sind somit an einem geregelten Datenunterhalt, an einer zentralen Datenverwaltung, interessiert.

Wenn nun ein sogenanntes *Datenverwaltungssystem* einen auf Dauer angelegten *Datenbestand* organisiert, schützt und verschiedenen Benützern geeignet zugänglich macht, bilden diese zusammen (Datenverwaltung *und* Daten) eine *Datenbank* (Fig. 1, schraffierter Bereich) (1).



Figur 1: Datenbank und Benützer



Figur 2: Datenbank-Konzept

Daraus und darin lassen sich *Datenbank-Grundsätze* erkennen:

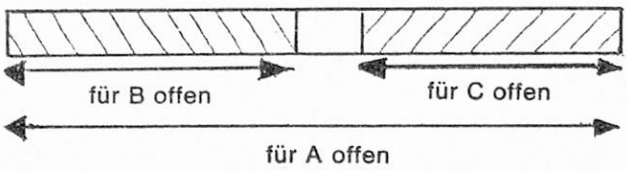
- *Trennung* der Daten und ihrer Organisation von den Anwendungen.
- *Datenunabhängigkeit:* Reorganisationen innerhalb des Datenbanksystems sollen die Anwendungsprogramme nicht tangieren.
- *Flexibilität:* Neue Datenbedürfnisse (z. B. Anwender D in Fig. 1) sollen auch nachträglich eingebaut werden können.
- *Datenintegrität:* «Datensicherung» gegen Verlust und Verfälschung, «Datenschutz» gegen mißbräuchliche Verwendung und Eingabekontrollen zur Vermeidung der Aufnahme falscher Daten.
- *Permanenz:* Die Daten sollen auf Dauer verfügbar sein.

In der Praxis sind heute noch selten alle diese anzustrebenden Grundsätze voll erfüllt. Das Postulat «Datenunabhängigkeit» z. B. geht nämlich von der Voraussetzung aus, daß die Anwendungsprogramme ihrerseits keine Rücksicht auf die interne Datenorganisation der Datenbank nehmen müssen. Und das ist heute aus Gründen der effizienten Verarbeitung bei *großen* Datenbeständen kaum möglich. Häufig muß man noch Kunstgriffe anwenden, um die Verarbeitung überhaupt mit tragbarem Aufwand durchführen zu können; die Verarbeitung ist dann von der Datenorganisation nicht unabhängig.

Die obigen Grundsätze sind *Zielvorstellungen*. Wir werden aber auch Datensammlungen, welche heute diese Ziele erst teilweise erreichen, als Datenbanken bezeichnen dürfen.

Verschiedene Gründe – vom Datenschutz (Zugriffsbeschränkung) bis zur angestrebten Flexibilität – machen es erwünscht, daß die gespeicherten Daten nicht für alle Anwender gleich zugänglich sein sollen. Wenn (in Fig. 3)

Anwender A den ganzen Datenbestand, B und C hingegen nur *Teilbereiche* (z.B. ohne vertrauliche Daten usw.) abfragen oder gar verändern können sollen, so muß auch diese *selektive* Zugänglichkeit durch das Datenverwaltungssystem sichergestellt werden. Wir sprechen dabei von Teilbereichen, Ausschnitten oder technisch von einem Subschema.

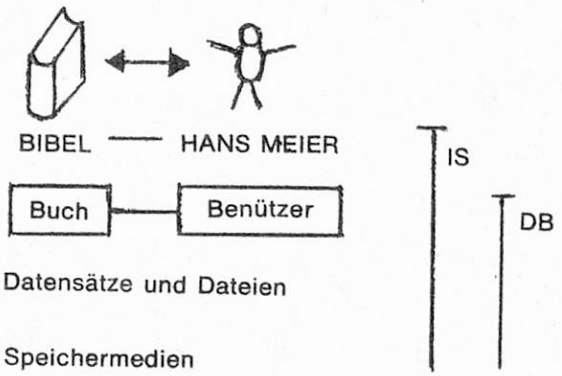


Figur 3: Zugänglichkeit von Teilbereichen

2. Betrachtungsebenen für Datensysteme

Der erste Abschnitt ging davon aus, daß irgendwelche Datenbestände existieren und nur vernünftig organisiert werden müssen. Um diese Organisation aber praktisch durchführen zu können, müssen wir genauer festhalten, *auf welcher Ebene* wir dies tun wollen. Wir unterscheiden dazu folgende Betrachtungsebenen (Fig. 4):

— *Reale Welt,* Teil der realen Welt
— *Informationen* über einen Teil der realen Welt, Modell
— *Logisches Datensystem*
— *Physisches Datensystem*
— *Computer* (Hardware und Software)



Figur 4: Betrachtungsebenen

Die Ebene der *realen Welt* steht in unserer Datenbetrachtung zuoberst; sie wird auf der nächsten Ebene durch *Informationen* beschrieben. Wir wollen annehmen, daß wir auf dieser Informationsebene bestimmte Fragen *über* die reale Welt beantworten wollen. Dabei benützen wir zur Umschreibung *Bezeichnungen* oder *Namen.* Vom Zweck her können wir so ein «Informationssystem» definieren:

> Ein *Informationssystem* soll Sachfragen in einem bestimmten Anwendungsgebiet innert nützlicher Zeit, mit vernünftigem Aufwand und mit genügender Genauigkeit beantworten.

Das Informationssystem (IS) verbindet somit die Informationsebene mit den gespeicherten Grunddaten (Fig. 4).

Von *unten* (von der technischen Hardware) her sind wir gewohnt, Daten viel computermäßiger zu sehen. Wir sprachen von Magnetbändern und

Plattenspeichern und ihrer Verarbeitung. Eine erste Abstraktion erlaubt uns den Schritt auf die Ebene der *physischen Datenorganisation,* wo wir uns immerhin etwas allgemeiner mit (sequentiellen) Dateien, invertierten Dateien, Tabellen usw. beschäftigen. Aber auch das ist das Tummelfeld des technischen Informatikers, für den Anwender kaum verständlich und uninteressant. Dennoch sind heute die meisten funktionierenden Datenbanksysteme auf dieser Ebene konzipiert und organisiert.

Auf der *logischen Datenebene* sollten sich nun Benützer und Informatiker halbwegs treffen können. Wir ersetzen auf dieser Ebene die technischen Datenbegriffe durch Merkmalsbezeichnungen («Buch», «Standortnummer» usw.), die dem Benützer nahe stehen, verlangen aber vom Benützer sehr präzise Aussagen über die von ihm verwendeten Begriffe (siehe Abschnitt 3). Zukünftige Datenbanken (DB in Fig. 4) sollten auf dieser logischen Ebene definiert werden können. Darum sei diese Idealstufe etwas näher betrachtet.

3. Logische Datenmodelle

Grundbaustein («Atom») für ein Datenmodell ist ein *Datenelement,* bestehend aus einem *Wert* eines *Merkmals,* wobei dieser Wert aus einem vorgegebenen («erlaubten») Wertevorrat oder *Wertebereich* stammt:

Datenelement: Farbe: mit Wertebereich: ROT, BLAU, . . .

SCHWARZ

Wenn wir vom Merkmal «Farbe» dieses Datenelements sprechen, stellen wir dies gelegentlich ebenfalls als Kästchen dar:

Merkmal:

Farbe

Dieses Kästchen steht damit für alle möglichen Datenelemente des Merkmals «Farbe». (Wir wollen im folgenden Merkmale *klein,* Merkmalswerte *GROSS* schreiben.) Der Wert «SCHWARZ» allein stellt übrigens *kein* Datenelement dar, da er ebensogut auch «Familienname» oder «Parteifarbe» sein könnte.

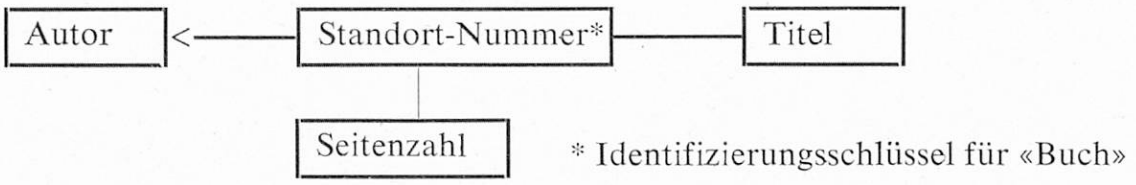
Datenelemente wie «Farbe: SCHWARZ» haben isoliert keinen Informationsgehalt. Erst die Kombination zweier oder mehrerer Datenelemente bringt eine Aussage:

«Herr Meier ist blond» könnte so aussehen:

Name:		Haarfarbe:
MEIER	——	BLOND

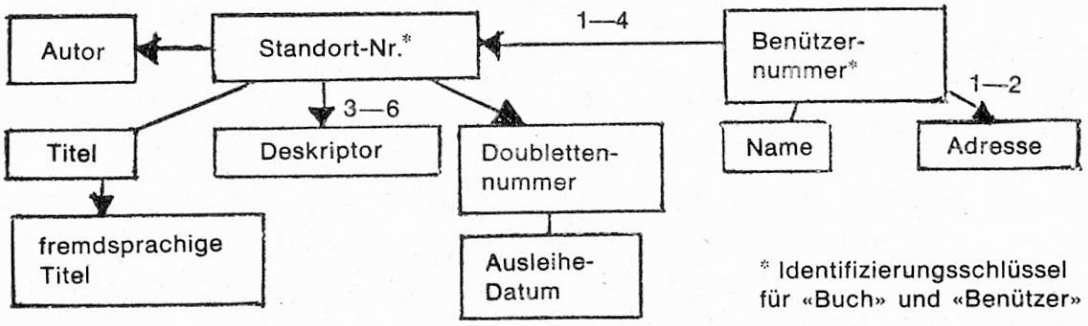
In den meisten praktischen Anwendungen ist es dabei bequem, Merkmale und Merkmalsträger auseinanderzuhalten. Der *Merkmalsträger* (allgemein:

Entität, entity, bei Personen auch «Individuum», bei Sachen «Gegenstand») hat dabei verschiedene Eigenschaften, die durch *Merkmale* beschrieben werden. So wird der Merkmalsträger «Buch» durch Merkmale wie «Titel», «Standort-Nr.», «Seitenzahl» beschrieben. Da der Merkmalsträger bequemerweise meist durch ein *Identifizierungsmerkmal* (auch Identifizierungs*schlüssel* genannt) *eindeutig* bezeichnet wird (ein «Buch» z. B. durch die «Standort-Nummer»), können wir auch die Merkmale einander direkt zuordnen, der Merkmalsträger steht dann nur noch als Gedanke im Hintergrund (Fig. 5).

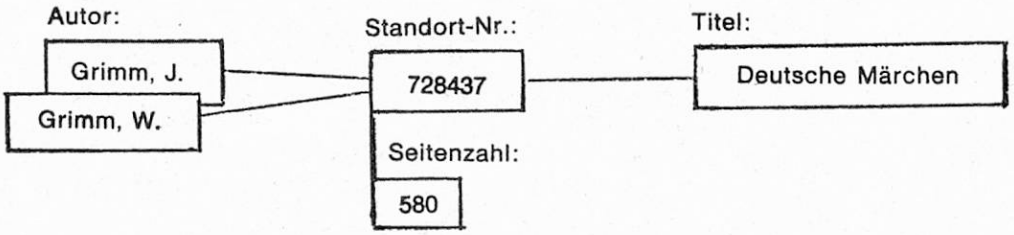


Figur 5: Logische Datenstruktur «Buch»

In Fig. 5 sehen wir einerseits Merkmale, die zu jedem Buch genau einmal vorkommen, anderseits aber bei «Autor» eine kompliziertere Beziehung. Ein Buch kann 1, 2, 3. . . oder sogar keinen Autor haben. Wir wollen diesen Zusammenhang zwischen «Standort-Nr.» und «Autor» eine «Vater-Kind-Beziehung» oder auch eine «1-m-Beziehung» (m für mehrere) nennen. Damit können wir eine bestimmte Datenbeziehung, die wir in einer Datenbank speichern möchten, anschaulich darstellen; wir nennen sie eine *Datenstruktur* (Fig. 6):

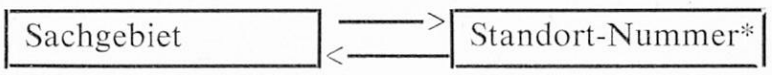


Die Datenstruktur bildet somit den Aufbauplan der Datenbank, hingegen bildet erst das *Vorkommen* der Daten gemäß dieser Struktur den eigentlichen Datenbestand. Das Beispiel aus Fig. 5 könnte dann so aussehen:



Figur 7: Datenstruktur-*Vorkommen* der Struktur aus Figur 5

Bisher haben wir bei der Verknüpfung von Daten auf der logischen Ebene keine Einschränkungen gemacht, sondern allein die *Bedürfnisse der Anwendung* berücksichtigt. Anderseits sind gewisse Verknüpfungen *technisch* nur sehr aufwendig darstellbar, z. B. würde dies gelten für:



* Identifikations-Schlüssel für «Dokument»

Figur 8: m-m-Beziehung

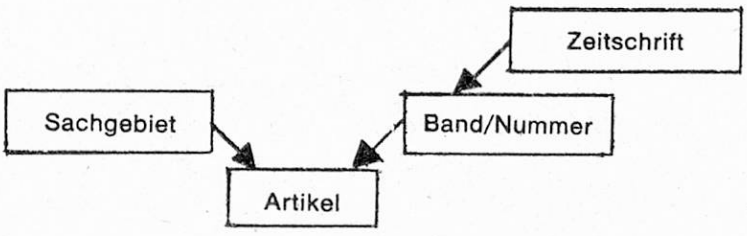
Denn jedes Dokument gehört vermutlich mehreren Sachgebieten an und jedes Sachgebiet sollte mehrere Dokumente umfassen. Wollten wir jede dieser *möglichen* Beziehungen (Vorkommen!) unterstützen, so ergeben sich theoretisch bei einigen tausend Sachgebieten und einer Million Dokumente einige Milliarden mögliche Verbindungen, wovon allerdings in der Praxis der weitaus größte Teil verschwindet. (Die Logarithmentafel hat nichts mit Philologie zu tun.)

Daher unterstützen und beeinflussen technisch orientierte Datenbanksysteme den Entwurf von Datenstrukturen derart, daß die unerwünschten „<——>" oder m-m-Beziehungen umgangen werden (was technisch sehr gut möglich ist). Wir wollen auf diese logischen Datenmodelle nicht im einzelnen eintreten, aber doch Hinweise machen:

- *Hierarchische Modelle:* erlaubt sind mehrstufige «Vater-Kind» Beziehungen, kein Element darf jedoch mehr als einen Vater haben (aber sehr wohl einen Großvater). Beispiel:

  Zeitschrift ——> Band ——> Artikel in Fig. 9.

- *Netzwerkartige Modelle:* wie hierarchisch, aber mehrere Väter sind erlaubt, z. B. Sachgebiet *und* Zeitschrift für den «Artikel» in Fig. 9.



Achtung: Das Netzwerk tritt erst beim Struktur-*Vorkommen* auf

Figur 9: Netzwerk-Struktur

- *Relationenmodelle:* Es gibt keine «Vater-Kind»-Beziehungen, sondern nur 1-1-Beziehungen. Was in einer 1-1-Beziehung zusammenhängt, beschreibt meist Entitäten und wird als Tabelle (= «Relation») dargestellt. Die Datenstruktur und die Daten selber bestehen aus *mehreren* solchen Tabellen; eine Verknüpfung der verschiedenen Daten erfolgt nur indirekt über den *Wert* der Daten:

Standort-Nr.	Titel
728437	Deutsche Märchen

Standort-Nr.	Autor:
728437	Grimm Jacob
728437	Grimm Wilhelm

Die Speicherung im Relationenmodell ist damit sehr einfach, die Abfrage eher aufwendig.

Für Einzelheiten dieser und anderer Datenmodelle siehe (2).

Nebst der Bedeutung als Aufbauplan ist die Datenstruktur aber auch *zentraler Träger* der *Datenintegritätskontrollen:*

- Dateneingabeprüfung: Die definierten Wertbereiche für jedes Merkmal und für jede Verbindung (Bsp.: 1 bis 2 Adressen pro Benützer in Fig. 6) erlauben zentrale Prüfungen.
- Datenschutz: Der Zugriff kann für bestimmte Merkmale, Merkmalsgruppen (Entitäten) oder gar Merkmalswerte beschränkt werden.
- Datensicherheit: Da diese vor allem auch auf intelligenten Doppelabspeicherungen (Redundanz) beruht, kann und soll sie auf dieser Ebene konzipiert werden.

Selbstverständlich sind damit längst nicht *alle* Integritätsmaßnahmen definiert und abgesteckt, da auf allen Ebenen Gefahren für die Datenintegrität lauern. Jene Maßnahmen aber, die die Struktur der Datenbank berücksichtigen, gehören in die logische Datenstruktur-Definition.

4. Physische Datenmodelle, Effizienzbetrachtungen

Mehrfach haben wir bisher die logische Ebene als Mittelebene im Entwurf von Datensystemen bezeichnet: Sie beschreibt Anwendungen, ist aber auch noch genügend datentechnisch orientiert, um entsprechend den «Modellen» in Abschnitt 3 auf einem Computer realisiert zu werden. Wie soll diese Realisierung oder *Implementierung* geschehen?

Dazu benötigen wir Konstruktionselemente der physischen Stufe. Vorstellungsmäßig stehen diese recht nahe bei herkömmlichen Zettelkatalogen, allerdings auf elektronischer Basis, was rasche automatische Suchprozesse, Quervergleiche und -abfragen erlaubt. Der Bücherkatalog als *Beispiel* (man beachte dabei die veränderte Terminologie gegenüber der logischen Ebene):

- Die zu einem Merkmalsträger («Buch») gehörigen *Daten* werden nach Möglichkeit in einen *Datensatz* vereinigt:

728437	DEUTSCHE MAERCHEN	GRIMM J., GRIMM W.
Standort-Nr.	Titel	Autoren

- Alle Datensätze vom gleichen Typ (also z. B. zum Typ «Buch») werden zu einer *Datei* aufgereiht und dabei nach einem *Primärschlüssel* geordnet (z.B. nach «Standort-Nr.»).

- Das *Suchen nach dem Primärschlüssel* (z. B. «Standort-Nr. = 843729?») ist sehr effizient möglich (da mit dem Verfahren der Eingabelung nur $\log_2(n)$ Zugriffe bei n Datensätzen nötig sind).
- Soll auch nach anderen Suchbegriffen (sog. Sekundärschlüsseln, z.B. «Autor») abgefragt werden können, hat man zwei Möglichkeiten:
  - *langsam:* sequentielles Durchsuchen der ganzen Datei.
  - *schnell:* Benutzung einer *invertierten Datei,* welche nach dem Sekundärschlüssel sortiert ist. In der invertierten Datei (= «Hilfskatalog») sucht man den gewünschten Primärschlüssel, womit die Rückkehr in die Hauptdatei möglich ist.
    (Invertierte Dateien erlauben durch vermehrten *Speicher*aufwand eine gewaltige Reduktion des *Rechen*aufwandes!)
- *Große Dateien* müssen durch Gliederungsmaßnahmen (z. B. durch Aufteilung in Teildateien, welche ihrerseits über einen Index, ein Inhaltsverzeichnis, zugänglich sind) handlicher gemacht werden. (Indexsequentielle Datei-Organisation, Typus «Telefonbuch»).

Bei der Implementation eines großen Datenbanksystems müssen mehrere solche Datei-Organisationen miteinander geeignet gekoppelt werden. Dabei müssen Primär- und Sekundärschlüssel, Sortierreihenfolge und Suchstrategien, sowie Speichergrößen und noch maschinennähere Parameter festgelegt werden, was anderseits die Leistungsfähigkeit, aber auch den Aufwand für ein Datenbanksystem weitgehend vorbestimmt. Da Zugriffe auf «Nachbardaten» je nach Speichermedien vom Mikrosekundenbereich (im Zentralspeicher) bis zum Zehntelsekundenbereich (auf großen Magnetplatten) variieren, ist für solche Entwurfsfragen eine gute Kenntnis des Computersystems unumgänglich. In Zukunft erhofft man sich sogar Optimierungshilfe vom Computersystem selber, das die Zugriffe statistisch auswerten und die Datenorganisation dementsprechend intern anpassen können sollte (Datenunabhängigkeit!).

Aus diesen Ausführungen sollen nun folgende *Überlegungen* folgen:

- Die EDV-Methoden sind den manuellen Katalog-Techniken sehr verwandt.
- Die EDV-Methoden unterstützen vor allem die systematische Bearbeitung *großer* Datenmengen.
- Große Datenmengen führen rasch an die Grenze des wirtschaftlich Machbaren (man denke an die Zehntelssekunden-Zugriffe bei falscher Reihenfolge für Hunderttausende von Datensätzen!).
- Der EDV-Spezialist muß mit dem Anwendungsorganisator zusammenarbeiten.

5. Abfragesprachen

Welcher Art sind die Fragen, die an eine Datenbank gestellt werden? Wir wollen mehrere Fälle unterscheiden:

– *Präzise und unpräzise Fragen:*
In der Ausleihe werden meist *präzise* Fragen gestellt, etwa «Ist das Buch mit Standort-Nr. = 728437 ausgeliehen?»
Ein Datenbanksystem muß nun imstande sein, über einen Suchprozeß auf den physischen Datensatz mit der genannten Standort-Nummer zuzugreifen (ein solcher Prozeß heißt auch «Zugriffspfad») und die Daten zum Merkmal «Ausleihe-Datum» zu lesen; ein leeres Ausleihe-Datum würde die Antwort «NEIN» bedeuten. In der Dokumentation hingegen steht mindestens amAnfang meist eine *unpräzise Frage,* wie etwa «Welche Literatur soll ich noch beziehen für mein aktuelles Forschungsprojekt?» Diese unpräzise Frage wird dann – oft unter Beizug von Spezialisten – umgeformt in eine Folge von Hilfsfragen, die dabei immer präziser werden und am Ende sogar in eine ganz bestimmte Bestell-Frage münden. (Zu diesem Dokumentations-Dialog siehe die Ausführungen von H. P. Frei in diesem Seminar (3) sowie insbesondere (4).)

– *Fragen und Mutieren*
Datenbanken werden im Normalfall weit häufiger abgefragt als in ihrem Datenbestand geändert, mutiert. Die Abfrage ist bezüglich *Suchaufwand* zwar ähnlich wie die Mutation, dennoch sind Mutationen *viel aufwendiger.* Hauptgründe dafür sind:
  – *Datensicherheit:* Beim Schreiben könnten vorhandene Daten zerstört oder beschädigt werden, was verhindert werden muß.
  – *Gleichzeitiger Zugriff:* Während einer Mutation (was auch im Computer einige Millisekunden dauert) darf kein anderer Benützer an die betroffenen Datenteile heran.
  – *Speicherorganisation:* Gewisse Mutationen erfordern anschließend kleinere oder größere Umspeichermaßnahmen.

  Die Mutationen sind daher besonders zurückhaltend zu erlauben und im interaktiven Betrieb auf *Schalterbedürfnisse* und bestimmte komplizierte Datenerfassungsprozesse zu beschränken.

Für *Abfrage-Systeme und -Sprachen* existieren heute schon viele Beispiele und Erfahrungen (2). Ein Dialog zwischen Maschine und Benützer soll insbesondere erlauben, daß der Mensch jeweils aus einem gewissen Angebot von Antworten dank seiner Problemkenntnis rasch eine wählen kann, worauf die Maschine ihrerseits weitere, feinere Varianten offeriert. Der Mensch will eine ausgewogene Mischung zwischen Lesen und Schreiben.

6. *Offene Entwicklungen und Probleme*

Die bisherigen Überlegungen haben gezeigt, daß die heutige Datenbankmethodik und -technik umfangreiche Datenprobleme systematisch behandeln kann. Erst in den Anfängen stehen allerdings Entwicklungen, die in Zukunft wesentlichen Einfluß auf Datensysteme haben dürften:

- *Dezentralisierte Datenbanken:* Zusammen mit den Entwicklungen in der «Datenkommunikation» sollen Datenbestände, die an einer Stelle vorhanden sind, über Übertragungsnetze von anderen EDV-Netzen aus zugänglich sein. Dabei spielen die Regelung der Verantwortung für die Daten, die Messung der Datenqualität und andere generelle Aspekte eine wachsende Rolle.
- *Automatische Anpassung der Datenbankorganisation an die zeitliche Entwicklung:* Der Gebrauch einer Datenbank bringt nicht nur meist eine Zunahme der Daten, sondern die Art der Daten und des Gebrauchs selber ändert sich mit der Zeit. Die – aus Effizienzgründen – zweckmäßigen Anpassungen der Datenorganisation sollen bei Wahrung der Datenunabhängigkeit der Programme automatisch erfolgen.
- *Technische Entwicklungen in Hard- und Software:* Diese Entwicklungen werden wesentliche Leistungssteigerungen ermöglichen, wobei gerade für die Abfrage noch neue Konzepte (z. B. «Assoziativspeicher») zu erwarten sind. Gleichzeitig ist aber für herkömmliche Anlagen die «Maschinenunabhängigkeit» der Anwendungen von vermehrter Bedeutung, damit ein Wechsel des Computersystems die Permanenz der Datenbestände nicht mehr derart gefährdet, wie dies heute noch der Fall ist.

Auf jeden Fall ist anzustreben, daß der Benützer – also z. B. der am Datenbestand interessierte und arbeitende Bibliothekar oder Dokumentalist – ein natürlicheres Verhältnis zu diesen technischen Mitteln erhält. Datenbanksysteme sollen in ihrem Konzept durchaus überblickbar, verständlich und damit als Diener des Benutzers geeignet sein.

Literatur

a) *Zitate*

(1) C. A. Zehnder: «Datenbanksysteme oder traditionelle Datenverarbeitung?» in: Management-Zeitschrift Industrielle Organisation, Zürich, Band 45, Nr. 10,1976.

(2) H. Wedekind: «Datenbanksysteme», Band I, Mannheim, B-I-Wissenschaftsverlag, 1974.

(3) H. P. Frei: «Automatische Klassifikationsmethoden», vorn S. 308—314.

(4) G. Salton: «Dynamic Information and Library Processing», Prentice-Hall, 1975.

b) Lehrbücher zum behandelten Stoff:

H. Wedekind: «Systemanalyse», München, Hanser, 1973.

H. Wedekind: «Datenbanksysteme», Band I und II, Mannheim, B-I-Wissenschaftsverlag, 1974 und 1976.

K. Bauknecht, C. A. Zehnder: «Technik der Datenverarbeitung», Stuttgart, Teubner, in Vorbereitung.

Anhang

«Einführung in die EDV»

Einige Grundbegriffe, wie sie den Stoff eines anwendungsbezogenen Einführungskurses in dieses Gebiet abstecken können.

1. Daten, Datengruppierungen

(Bit = Binärstelle, Informationselement, Ja/Nein-Wert)

Zeichen (character), gelegentlich auch der dafür nötige Speicherplatz = Byte.
Datenfeld, Wort, Zahl; auf der logischen Ebene oft «Merkmal», «Attribut».
Datensatz (record): repetierbare, oft einem Begriff der realen Welt entsprechende Gruppe von Daten, die zusammengehören und diesen Begriff beschreiben (z. B. Personen- oder Artikel-Datensatz).

(Block = technisch notwendige Gruppierung von Daten zu größeren Einheiten)

Datei (file,fichier) Sammlung zusammengehöriger Datensätze, meist sequentiell organisiert, mit erkennbarem Anfang und Ende.

Begriffe im Zusammenhang mit obiger Hierachie

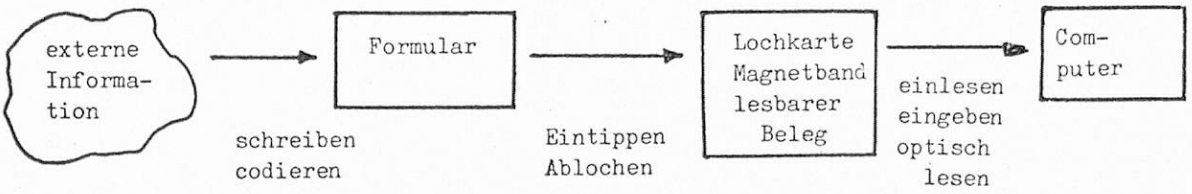
Zeichensatz: z. B. «64-Zeichen-ASCII-Code»: A . . . Z ∅ 1 . . . 9+ — . . .
Groß- und Kleinschrift erst ab 88 oder 96 Zeichen möglich
technische Darstellung: spezielle Druckerketten und Kugelköpfe

formatierte und unformatierte Daten: (Beispiele)
formatiert: AHV-Nummer, Postleitzahl, Name mit max. 30 Zeichen
unformatiert: «groß, blauäugig, mit Narbe über rechtem Auge»
codiert: 1,2,3,4 für «ledig, verheiratet, verwitwet, geschieden». Codiedierung erlaubt viel dichtere interne Speicherung, benötigt aber zum voraus Übersicht über alle möglichen «Werte» oder «Ausprägungen», die ein Merkmal annehmen kann.

Schlüssel: Die einfachste Beschreibung einer Realität geschieht durch eine Datei von Datensätzen (z. B. Katalog von Bücherdatensätzen). Ein *Identifikationsschlüssel* erlaubt die genaue Bestimmung eines bestimmten Datensatzes, er ist Basis für einen Suchprozeß.
Ein *Sortierschlüssel* erlaubt die Festlegung einer gewissen Sortierfolge innerhalb einer Datei, z. B. für eine Verarbeitung oder für eine Sucharbeit. Ist eine Datei in falscher Sortierreihenfolge gespeichert, so gibt es auf jedem Computer Sortierprogramme, um die gewünschte Sortierfolge zu erzeugen. Sortierschlüssel müssen nicht eindeutig sein (z. B. Stichworte).

2. Verarbeitung der Daten

Dateneingabe (input)



Bei gewissen Verfahren (z. B. optische Leser) entfallen einzelne Schritte.

Datenverarbeitung

— Normalfall: Große Arbeiten, große gleichartige Datenmengen
nicht außerordentlich terminorientiert («es wartet niemand»)
→ Stapelverarbeitung (batch processing)

— Sonderfall: Schaltersituation (es braucht jemand sofort eine Antwort)
Datenprüfung (bei komplizierter Dateneingabe)
→ Interaktive Verarbeitung (konversationell; ähnlich auch «real time» (Echtzeit), «time sharing» [= technische Lösung bei Großsystemen])

Hauptarbeiten

— Datenprüfung, Nachführung von Datenbeständen, Datenpflege
Wegen der großen Kontrolle bei möglichen Fehlern macht dies oft einen großen Teil der Programmierarbeit, aber nicht der Rechenleistung aus.

— Rechnen: ursprünglich eher bei technisch-wissenschaftlichen Anwendungen, heute überall: Optimieren, Suchen, Simulieren usw.

— Umorganisieren: Sortieren, Mischen von Dateien

— für die Ausgabe vorbereiten: Suchen, Auswählen

Datenausgabe (output)

— Listen von gespeicherten oder berechneten Daten

— Zählwerte (Statistiken) über gespeicherte Daten

— Einzelauskünfte

3. Aufwand-Überlegungen

— Entwicklung und Betrieb Personal — Computer — andere

— Einzel- oder Massenarbeiten

— Datenschutz und Datensicherheit

— Daten-Abfrage oder Daten-Mutation besonders in interaktiven Systemen.