Überlegungen zur Entwicklung von Datenbanksystemen

Horst Remus, IBM Palo Alto, Californien

Zusammenfassung

Bei der Entwicklung zur integrierten Datenverarbeitung sind zwei Schritte besonders bemerkenswert:

- Die Datenbank als Zentrale, wobei die Anwendungsprogramme im wesentlichen den Verkehr mit der Datenbank regeln (Abfrage oder Aufarbeitung).

- Das Datenfernverarbeitungsnetzwerk, das den gleichzeitigen Zugriff zu einem Programm oder einer Datenbank von mehreren Benutzerstationen aus gestattet.

Die Datenbankzentrale des Datenverarbeitungssystems im Gegensatz zu der Datei als Zugriffsdatei für ein bestimmtes Programm (mit OPEN und CLOSE der Datei von diesem einen Programm) erfordert bestimmte Überlegungen bezüglich ihrer Organisation. Ein weiterer Schritt ist die Einfügung genereller Datenbanksysteme mit der Idee der Datenunabhängigkeit. Andere Überlegungen haben mit der Beantwortungszeit ("performance"), Datenschutz und Datensicherung ("integrity" und "recovery") zu tun. Für den Benutzer stellt sich das System in zwei Teilen dar:

- Das Datenmodell
- Die Sprache mit der diese Daten manipuliert werden ("user interface").

Künftig zu lösende Probleme weisen in die Richtung von Datenbanken mit gleichzeitigem Zugriff von mehreren Systemen und in Netzwerken auf verschiedene Knotenpunkte verteilte Datenbanken.

1. ENTWICKLUNG ZUR DATENBANK

Wir betrachten Mengen, deren Elemente aus alphanumerischen Zeichen zusammengesetzte Daten oder Informationen sind. Für diese Mengen ergeben sich folgende Operationen:

a) Die *Abfrage*, d.h. die Herauskristallisierung gewisser Teilinformation aus der Gesamtmenge.

b) Die *Berichterstellung*, d.h. die (meist summarische) Zusammenfassung der Informationsmenge, oder Teilen daraus, nach gewissen nicht notwendig automatisch in der Mengenstruktur gegebenen Merkmalen.

c) Die *Aufarbeitung* der Informationsmenge, d.h. Hinzufügung, Ausstreichen oder Verändern von Teilen der Informationsmenge. (Eine spezielle Form der Aufarbeitung ist die Formatänderung, d.h. das Hinzufügen oder Fortlassen von Information relativ zu jeder vorhandenen Teilinformation.)

Historisch gesehen ergibt sich bezüglich der Struktur oder Organisationsform von Informationsmengen folgende Entwicklung (Abbildung 1 zeigt einen Versuch zur schematischen Darstellung):

Der erste Schritt zur Zusammenfassung von Information ist die *Liste*, wobei die einfachste Form die *fortlaufende Liste* ist. Als Datenträger in der ursprünglichen Form dienen Medien auf denen lesbar geschrieben werden konnte. Die Abfrage erfolgte manuell, die Liste wird nach dem infrage stehenden Eintrag (normalerweise startend am Anfang der Liste) durchsucht. Eine Berichterstellung ist in den meisten Fällen unmöglich, da Einzelabfragen sehr zeitraubend sind. Die Aufarbeitung erfolgt manuell durch Hinzufügung eines neuen Eintrags am Ende oder durch Streichung überflüssig gewordener Einträge. Eine Änderung im Listenformat (zusätzliche Information per Eintrag) führt normalerweise nicht zu Schwierigkeiten, da die zusätzliche Information ohnehin nur für die neu hinzugefügten Einträge verfügbar ist.

Der nächste Schritt ist die *geordnete Liste* mit den gleichen Medien als Datenträger. Eine geordnete Liste entsteht aus einer fortlaufenden Liste durch Sortierung nach einem Ordnungsbegriff. Es ist auch möglich, daß eine fortlaufende Liste automatisch geordnet ist, z.B. bei chronologischen Listen wie Kirchenbuchregistern.

Die Abfrage ist wesentlich vereinfacht und erleichtert damit die Berichterstellung. Bei der Aufarbeitung treten Probleme mit der Einschiebung von Einträgen auf. Jede Menge dafür vorgesehener Platz erschöpft sich. Das führt entweder zu einer Zerstörung der Ordnung oder es muss eine neue Liste erstellt werden. Ein gewisser Ausweg sind die Ergänzungslisten und Hinweise auf solche in der Basisliste (anstelle des Eintrags der Gesamtinformation). Derartige Verfahren führen jedoch schnell zur Unübersichtlichkeit, z.B. werden eröffnungstheoretische Werke für Schach immer wieder neu aufgelegt.

Der nächste Schritt wäre das Auseinanderbrechen der Liste in Einzeleinträge, die *Kartei*. Sie stellt gewisse spezielle Ansprüche an die Medien. Die Schwierigkeiten in der geordneten Liste bezüglich Hinzufügen von Einträgen sind beseitigt.

Die Erfindung der Lochkarte und die damit verbundene elektromechanische Behandlung von Information bedeutete die Möglichkeit, einzelne manuelle Verarbeitungsschritte zu automatisieren. Die semi-automatische Einzelabfrage ist jedoch im Normalfall zu zeitraubend. Die Berichterstellung kann weitgehend automatisch erfolgen, jedoch muß die *Lochkartenkartei* für das Programm, d.h. die Tabelliermaschinenschaltung, speziell vorbereitet werden (Sortieren, Mischen und andere spezielle Arbeitsgänge). Die Aufarbeitung erfolgt semi-automatisch. Problematisch wird die Formatänderung, die meist zur Erstellung einer neuen Kartei führt.

Benutzung anderer Medien wie Platte oder Band ermöglichen vollautomatische Verarbeitung und führen zur *Datei*. Normalerweise ist diese, ähnlich wie die Lochkartenkartei, relativ zu einer bestimmten Anwendung organisiert. Der Programmierer "öffnet" (OPEN) und "schließt" (CLOSE) die Datei, je nachdem ob die zugehörige Anwendung läuft oder nicht. Läuft die Anwendung nicht, wird die Datei unter Umständen sogar physikalisch vom System entfernt; jedenfalls ist sie normalerweise nicht für andere Anwendungen zugriffsbereit. Abfrage und Berichterstellung sind auch nur für bestimmte Anwendungsprogramme möglich. Die gleichzeitige Bearbeitung mehrerer Anwendungen von ein und derselben Datenstation oder von einer oder mehr Anwendungen von verschiedenen Datenstationen wird problematisch. Aufarbeitung und Formatänderung erfordern die automatische Erstellung einer neuen Datei.

Eine Vielzahl von Anwendungen und Benutzern für ein und dieselbe Datenmenge führt zur Datenbank. Ihre speziellen Erfordernisse werden im folgenden näher erläutert.

2. DATENBANKEN UND DATENBANKSYSTEME

Implizit enthalten in der Definition der Datenbank ist das Konzept der minimalen Redundanz und die Notwendigkeit einer für den Benutzer verständlichen Struktur, dem *Datenmodell*.

Zugriff zu einer Datenbank erfolgt normalerweise von einer Reihe von Benutzern mit verschiedenartigen Anwendungen gleichzeitig. Das erfordert eine fortlaufende Überwachung der Datenbank durch einen *Datenbankverwalter*. Neben der Erhaltung der Integrität der Datenbank streben diese Systemprogrammierer eine optimale Erzielung von Leistungsfaktoren wie Beantwortungszeit und Speicher an. Sie interessieren sich daher für die physikalische Organisation der Datenbank, einschließlich der Wirkungsweise von Indizes und Zeigern.

Die Anwendungsprogrammierer oder "Enduser" interessieren sich für das logische Datenmodell und für Wege zum Wiederauffinden und zur Aufarbeitung von Datenbankelementen.

Um zu verstehen, welche Forderungen beide, der Datenbankverwalter und der Anwendungsprogrammierer, an Datenbanksysteme haben, müssen die Anwendungen von Datenbanken näher untersucht werden.

Zunächst sei an den Unterschied von Stapelverarbeitung (batch processing) und Echtzeitverarbeitung (real time processing) erinnert (Abbildung 2). Bei der Stapelverarbeitung erfolgt die Verarbeitung bezüglich eines Merkmales oder Begriffs gruppenweise an bestimmten festgelegten Terminen oder nachdem eine bestimmte Menge zur Verarbeitung angesammelt ist. Bei der Echtzeitverarbeitung wird jeder Schritt sofort auf der gesamten Datenmenge ausgeführt.

Außerdem sind bei den Anwendungen zwei Parameter von besonderer Bedeutung:

. die Voraussehbarkeit
. die Häufigkeit gleichartiger Zugriffe (Repetivität).

Hierbei gibt es bezüglich beider Merkmale eine Reihe von Mischungen. Man weiß z.B. nicht im voraus, nach welchem Teil eines Lagerbestands ein Magazinverwalter fragt. Was er darüber wissen will, ist jedoch genauestens bekannt. Im allgemeinen kann man Datenbankoperationen in folgende verschiedenartige Operationen einteilen (Abbildung 3):

1. Wirkungsvolle Ausführung sich wiederholender Arbeiten (traditionelle Stapelverarbeitung).

2. Im voraus definierte Abfragen ("Wie groß ist der Lagerbestand an 2 Zoll Nägeln ?").

3. Zufällige, schlecht strukturierte und unvorhergesehene Abfragen ("Wieviele Ingenieure in Hamburg haben ein Monatseinkommen von mehr als DM 6000.-- ?").

Ein System, das Nr. 1 und 2 behandelt, wird *"Operational"* oder *"Supervisory System"* genannt, ein System, das Nr. 3 behandelt, ein *"Informations"* oder *"Executive System"*. Beispiele für beide Gruppen wären:

"Operational" Systeme: Bank mit Datenstationen an jedem Schalter, Flugreservierung, Flugsicherung.

Informationssysteme: Bücherei mit Aufsuchen von Information nach Kennwort, Marktinformation für Management, Datenbank mit Personaldaten.

Ein und dieselbe Datenbank sollte normalerweise die Anwendung beider Systeme erlauben.

3. SPEZIELLE ANFORDERUNGEN AN DATENBANKEN

Es wurde bereits auf die Forderung der *minimalen Redundanz* hingewiesen. Die meisten Band-Bibliotheken enthalten eine Fülle von redundanten Daten. Unkontrollierte Behandlung der Frage der Redundanz kann (wie z.B. bei vielen Büroablagesystemen) zu der Notwendigkeit häufiger Um- oder Neuordnung führen. Eine weitere Frage ist natürlich der Verbrauch an Speicherplatz und die damit verbundene Kostenfrage. Mehrfache Kopien derselben Daten können außerdem wegen eines möglicherweise verschiedenen Aufarbeitungsstandes zu verschiedener Information führen. Ziel einer Datenbankorganisation sollte es also sein, Redundanz zu vermeiden, wo es ökono-

misch richtig erscheint. Aus Gründen der Datensicherheit und zur möglichen Wiederherstellung fehlerhafter Daten kann jedoch einige Redundanz erforderlich sein.

Eine weitere Forderung ist die *Vielseitigkeit in der Darstellung* von Datenbeziehungen. Verschiedene Programmierer benutzen unterschiedliche logische Dateien, die jedoch alle auf derselben Datenbank beruhen.

Sehr bedeutend sind die Aspekte der *Leistungsfähigkeit* eines Datenbanksystems. Entscheidende Leistungsfaktoren sind die Antwortzeit für die Benutzer einer Datenstation und die Anzahl der Übertragungen je Zeiteinheit, die ein System bewältigen kann. Es gibt Systeme mit geringerem Verkehrsvolumen, bei denen die Anzahl der Übertragungen je Zeiteinheit (throughput) von geringer Bedeutung ist. Systeme mit hohem Verkehrsvolumen sind z.B. Flugreservierungssysteme und Großbanken. Es gibt heute bereits Anwendungen, die 10 und mehr Übertragungen in der Sekunde erfordern. Bei derartigen Anwendungen ist eine weitere rasche Steigerung zu erwarten (Hinzufügen von weiteren Bank-Zweigstellen etc.). Um die erforderliche Leistungssteigerung besser in den Griff zu bekommen, sind weitere Maßnahmen in Betracht zu ziehen, wie z.B. Aufspaltung der Datenbank in mehrere Einzeldatenbanken (Dezentralisierung) oder Zugriff zu einer Datenbank von mehreren Rechenanlagen aus. Für traditionelle Stapelverarbeitungssysteme ist die Antwortzeit ohne Bedeutung. Ihr Entwurfskriterium ist die Effektivität des "batch processing" (Stapelverarbeitung). Für gewisse Anwendungen ist ein Dialog mit einer Antwortzeit von 2 Sekunden oder weniger erforderlich. Natürlich ist die Leistungsfähigkeit der Recheneinheit von Einfluß auf die Leistungsfähigkeit des Datenbanksystems.

Es ist notwendig, daß Daten und ihre Beziehungen untereinander nicht durch Maschinenfehlverhalten oder andere "Unfälle" zerstört werden (D a t e n s i c h e r h e i t). Jedes System muß daher die Möglichkeit von Datensicherheitstests beinhalten.

In vielen Fällen müssen Daten vor dem Zugriff Unbefugter geschützt werden ("security and privacy" = *Datenschutz)*. Diese Forderung kann übertragen werden auf die Forderung, daß das System die Authorisation eines Benutzers und seiner Aktionen überprüft (z.B. durch ein Passwort). Die Kontrollen sollten so gestaltet sein, daß geschickte Programmierer sie nicht ohne weiteres umgehen können. Auch sollten die Aktionen überwacht und notiert werden, sodaß falscher Gebrauch nachträglich herausgefunden

werden kann. Ebenso ist es erforderlich, daß die Datenbank selbst laufend überprüft werden kann.

Außerdem tritt die Forderung auf, Anwendungsprogramme unabhängig von der Datenorganisation und Zugriffstechnik zu schreiben (*Datenunabhängigkeit*). Z.B. bietet IMS [3] einen gewissen Grad von Datenunabhängigkeit, indem neue Datensegmente an bestimmten Punkten der Hierarchie ohne Programmänderung hinzugefügt werden können, oder auch die Länge eines Datensatzes oder die Aufteilung der Datenbank in Datengruppen geändert werden kann.

4. DATENBANKSTRUKTUREN

Die Funktion einer Datenbank ist das Abspeichern der Daten und der Beziehungen zwischen den Daten.

Die logische Beschreibung einer Datenbank wird das Datenbankschema genannt. Ein Schema definiert also das Datenmodell für den Anwender. Ein Subschema ist die Aufgliederung der Datenbank für ein spezielles Anwendungsprogramm. Abbildung 4 zeigt das Zusammenwirken der verschiedenen Teile innerhalb eines Datenbanksystems und insbesondere die Bedeutung der Begriffe Schema und Subschema.

Abbildung 5 zeigt die Aufgliederung einer Datenbank zur Arbeitsplatzbeschaffung. Die Beziehungen zwischen den einzelnen Dateien sind klar ersichtlich. Die Arbeitgeberdatei gibt die Einzelheiten zu dem Feld "Arbeitgebernummer", die Talentdatei die Einzelheiten zu dem Feld "Gefordertes Talent" in der Arbeitsplatzliste. Hierbei zeigt sich eine Hauptform für Datenbankstrukturen: die hierarchische Gliederung. Die Dateien "Arbeitgebernummer" und "Talentgruppe" sind Untergliederungen der Datei "Arbeitsplatzliste" (Eltern-Kind-Beziehung).

Die Möglichkeit Beziehungen zwischen den einzelnen Datenfeldern in der Datenbankstruktur zum Ausdruck zu bringen, hat zu drei wesentlichen Datenbankorganisationsformen geführt:

1. Die *hierarchische Datenbankstruktur* (Abbildung 6). Hierbei hat der höchste Level einen und nur einen Knotenpunkt, die "Wurzel des Baumes". Jeder Knotenpunkt eines anderen Levels erhält genau einen Knotenpunkt in dem nächsthöheren Level zugeordnet.

Knuth [4] definiert einen Baum oder eine hierarchische Struktur entsprechend als "eine endliche Menge T von einem oder mehr Knotenpunkten mit

a. einem speziell ausgezeichneten Knotenpunkt, der Wurzel des Baumes und
b. $m \geq 0$ verbleibenden disjunkten (unverbundenen) Teilmengen $T_1 \ldots\ldots T_m$, wobei jede dieser Teilmengen ein Baum ist. Diese Teilmengen werden Teilbäume genannt."

IMS [3] verwendet die hierarchische Datenbankstruktur.

2. Falls ein Knotenpunkt auf mehr als einen Knotenpunkt einer höheren Ebene zurückgeführt werden soll, kann die Beschreibung nicht mehr durch einen Baum erfolgen. Die entstehende Struktur wird als "*Netzwerkstruktur*" bezeichnet. Wegen des vielseitigen Gebrauchs des Wortes Netzwerk in der Datenindustrie wird im angloamerikanischen Sprachbereich häufig die Bezeichnung "plex structures" verwendet. Abbildung 7 zeigt einige einfache Beispiele von Netzwerkstrukturen. Natürlich ist eine hierarchische oder Baumstruktur nur ein spezieller Fall derselben. Ein Beispiel einer einfachen Netzwerkstruktur ist ein Stammbaum. Komplexere Strukturen entstehen, wenn mehrfache, nicht algorithmisch bestimmbare Beziehungen zwischen den Elementen verschiedener Level existieren. Unter Einführung von Mehrfachindizes und Redundanz können Netzwerkstrukturen auf Baumstrukturen zurückgeführt werden. Die Ausarbeitungen der Codasylgruppe [1] führen zu einer Netzwerkstruktur.

3. Die Forderung ohne Redundanz auszukommen und die Beziehungen zwischen den Datenbankelementen als algebraischen Kalkül darstellen zu können, führt zu der "relational data base" nach Codd (siehe ausführliche Beschreibung in [2]).

   Die Grundoperationen zur Formung neuer Datensätze sind Vereinigung und Durchschnitt. Die Sprache erscheint vom mathematischen Standpunkt aus sehr elegant, doch haben sich Implementierungen aus Gründen der Leistungsfähigkeit bisher wenig durchgesetzt. Die Vorteile von Datei mit Datensätzen auf dem gleichen Level gliedern sich um Übersichtlichkeit des Datenmodells und Einfachheit der Sprache mit denen Beziehungen manipuliert werden können. Darstellungen in "relational data base"-Form können durch Verwendung von Mehrfachindizes und Redundanz auf

obige Formen der hierarchischen oder Netzwerkstrukturen zurückgeführt werden.

Im Zusammenhang mit Datenbankstrukturen wird häufig von Listen und Ringen gesprochen (chains or lists, rings). Diese Strukturen beziehen sich jedoch auf die Art, in der Datensätze innerhalb einer Datei untereinander verbunden sind. Sie beschreiben daher Techniken, wie logische Strukturen aus physikalischen erreicht werden, während die unter 1-3 beschriebenen Strukturen spezielle Formen logischer Strukturen darstellen. Ein entscheidendes Element für beide, die Listen- als auch die Ringstruktur, sind die Zeiger (pointer), die von einem auf den folgenden Datensatz weisen. Bei der Ringstruktur sind dabei normalerweise zweiseitige Zeiger gebräuchlich.

5. DATENBESCHREIBUNGSSPRACHEN

Eine Sprache, die die logische Datenstruktur beschreibt, sollte die folgenden Forderungen erfüllen:

Die Gliederung in Datenmengen wie Dateien, Sätze, Segmente, Datenelemente, sollte klar beschreibbar sein.

Jeder Typ einer solchen Mengeneinheit sollte spezifisch bezeichnet sein (z.B. sollten 2 verschiedene Satztypen verschiedene Bezeichnungen haben).

Die Untergliederung einer bestimmten Datenmenge in bestimmte Untermengen sollte klar erkennbar sein (welche Datenelemente in einer bestimmten Datengruppierung enthalten sind etc.). Die Aufeinanderfolge muß spezifiziert und Wiederholungen sollten aufgezeigt sein.

Die Sprache sollte ausdrücken, welche Datenelemente als Indizes benutzt werden.

Beziehungen zwischen Satztypen, Segmenttypen etc., die die Grundlage der Datenstruktur bilden, müssen spezifiziert und klar bezeichnet werden.

Nach J. Martin [5] ergeben sich je nach dem Gesichtspunkt des Benutzers verschiedene Level der Datenbeschreibungssprachen (Abbildung 8):

1. Die Sprache für den Anwendungsprogrammierer, die das Datenbanksubschema beschreibt (z.B. die Datendivision in COBOL oder die PSBs in DL/1 (PSB = program specification block)).

2. Die generelle Beschreibung des Schemas der Datenbank, die vom Datenbankverwalter angewandt wird (z.B.: DL/1 logical data base description). Die COBOL Datendivision erlaubt z.B. nicht, die Beziehungen in einem Schema zu beschreiben. Sie kann daher hier nicht verwendet werden.

3. Die physikalische Datenbeschreibung (z.B.: DL/1 physical data base description). Im Gegensatz zur logischen Datenbeschreibung, die völlig losgelöst ist von Hardware- und Speicherüberlegungen, sind diese jedoch für Leistungsoptimierung sehr interessant.

Außer DL/1 ist wahrscheinlich CODASYLs data description language DDL die bekannteste Datenbankbeschreibungssprache.

6. ÜBERLEGUNGEN BEI DER HARDWARE

Es sind Datenbanken von der Größenordnung von mehr als 4 Milliarden Bytes bekannt. Das entspricht 40-50 Platteneinheiten IBM 3330. Es ist denkbar, eine Platteneinheit durch eine größere Speichereinheit mit längerer Zugriffszeit zu unterstützen, ähnlich wie beim virtuellen Speicherkonzept zwischen Kernspeicher und Platte. Die vor etwa einem Jahr angekündigte IBM 3850 liefert z.B. 10^3 bis 10^4 mehr Speicherraum mit einer um den Faktor 10^2 verlängerten Zugriffszeit. Der Benutzer sieht das System als ein einziges Plattensystem, für Leistungsfähigkeitsbetrachtungen sind die Hardware-Parameter jedoch von größter Bedeutung. Zum Beispiel bestehen strenge Abhängigkeiten zwischen Antwortzeit, Übertragungsrate und Direktspeichergröße, oder Speicherverfügbarkeit in der niedrigsten Stufe der Speicherhierarchie. Die Antwortzeit wächst mit der Übertragungsrate und fällt mit mehr Direktspeicherverfügbarkeit (weniger paging). Die Übertragungsrate kann mit mehr Direktspeicher gesteigert werden.

Andere Hardware-Parameter sind natürlich die Geschwindigkeit des Computers, der Aufbau und die Komponenten des Nachrichtennetzes.

7. AUSBLICK

Die zusätzlichen Anforderungen für Erweiterungen bestehender oder Entwicklung zukünftiger Datenbanksysteme gliedern sich um die folgenden Aspekte:

a) *Steigerung der Leistungsfähigkeit*. Wachstum der Datenbank und der Anzahl der Datenbankbenutzer erfordern höhere Übertragungsraten und kürzere Antwortzeiten. Die Antwort liegt in geeigneteren Datenbankorganisationen und einer Minimisierung von Verwaltungsfunktionen. Gewisse Hilfsmittel der Hersteller ermöglichen ein "tuning" der Datenbank, dazu ergeben sich Anwender-beeinflußte Verbesserungsmöglichkeiten. Gewisse Verbesserungen sind durch geeignetere Verwendung von Hardware erzielbar (multiprocessing oder ähnliche Verfahren).

b) *Fortlaufende Operation*. Die Forderung einer 24-stündigen Zugriffsmöglichkeit zur Datenbank führt zu gewissen Konsequenzen bei der Implementierung.

   Zunächst wird bei Unterbrechung durch Fehlverhalten eine schnelle Wiederherstellung der Datenbank und kurzfristige Wiederaufnahme der Operationen notwendig. Das erfordert die Führung eines schnell zugriffsbereiten "Journals". Außerdem sollte an den besten Techniken zur Fehlerverhütung, -auffindung und -korrektur gearbeitet werden.

   Eine weitere Forderung ist, die Datenbank - bei gleichzeitiger Fortführung des Routinebetriebs - zu reorganisieren. Ein Dictionary [7] kann dabei als wesentliche Hilfe zum Management der Datenbanken dienen.

c) *Einfachheit der Installierung und Benutzung*. Die Parameter, die zur optimalen Organisation einer Datenbank führen, sind sehr komplex. Systemhersteller helfen allgemein mit automatischen Organisationshilfen oder Hinweisen in der Dokumentation.

   Die Frage der Installierbarkeit ist weitgehend identisch mit der Möglichkeit, die physikalische Repräsentation der Datenbank zu verstehen. Wiederum kann ein Dictionary [7] nützlich sein.

Einfachheit der Benutzung hängt wesentlich mit der Beschaffenheit der Sprachen zur Datenmanipulierung und -beschreibung und dem "interface" zu den Programmierungssprachen ab.

Weitere Funktionen, die zur Vereinfachung der Benutzung führen, haben mit der automatischen Regelung des Informationsflusses zu tun. Wesentlich ist hierbei die Handhabung der Kontrollinformation (Kontrollblöcke), wie sie z.B. bei der standard network architecture erfolgt.

Um die spätere Benutzung zu vereinfachen, müssen Datenbanken und zugehörige Systeme auf die Möglichkeit zur späteren Veränderung bzw. Erweiterung ausgelegt sein.

Literatur

[1] CODASYL, "1974 Status Report on Data Base Activities"

[2] Date, C.J., "An Introduction to Database Systems". Addison-Wesley, Reading, Mass. 1975

[3] Information Management System, "System/Application Design Guide" IBM Form No. SH 20-9025

[4] Knuth, D.E., "The Art of Computer Programming; Vol. 1, Fundamental Algorithms". Addison-Wesley, Reading, Mass., 1968

[5] Martin, J., "Computer Data Base Organization", Prentice-Hall, Englewood Cliffs, N.J., 1975

[6] Senko, M.E., Altman, E.B., Astrahan, M.M and Fehder, P.L., "Data Structures and Accessing in Data-Base Systems". IBM Systems Journal 12, 30-93 (1973)

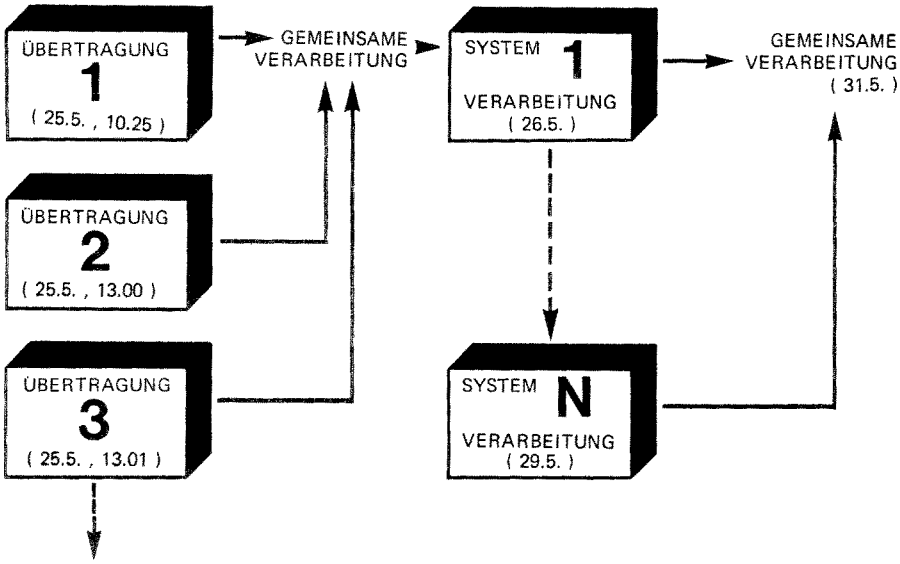
[7] Uhrowczik, P.P., "Data Dictionary/Directories". IBM Systems Journal 12, 332-350 (1973)

Datendarstellung	Datenträger	Abfrage	Berichterstellung	Aufarbeitung	Formatanderung
Fortlaufende Liste	Medium, das menschliches Schreiben und Lesen erlaubt.	Manuelles Durchsuchen (generell: Start am Anfang)	Manuell, bestimmt durch zeitraubende Einzelabfragen	Manuell, Zufügung neuer Einträge am Ende	Kein Problem, neues Format bleibt auf neue Einträge beschränkt.
Geordnete Liste	– " –	Manuell, unter Benutzung des Ordnungsbegriffs	Manuell	Häufige Neuerstellung wegen Ausschöpfung des Platzes für Zufügungen	– " –
Kartei	Medium,separierbar je Eintrag	– " –	– " –	Manuell, unbegrenztes Hinzufügen möglich	– " –
Lochkartenkartei	Lochkarte	Manuell oder semiautomatisch (sehr zeitraubend)	Semiautomatisch, die Kartei wird für das entsprechende Programm vorbereitet	Semiautomatisch	Erfordert normalerweise Neuerstellung der Kartei
Datei	Band, Platte	Automatisch, beschränkt auf die zu dieser Datei gehörende Anwendung	Automatisch, die Datei wird für das entsprechende Programm vorbereitet	Automatisch, mit häufiger Neuerstellung	Erfordert normalerweise Neuerstellung der Datei
Datenbank	– " –	Automatisch unbegrenzt	Automatisch soweit Information vorhanden unbegrenzt	Automatisch unbegrenzt	Automatisch unbegrenzt

ENTWICKLUNG ZUR DATENBANK

Abbildung 1

STAPELVERARBEITUNG (BATCH PROCESSING)



ECHTZEITVERARBEITUNG (REAL TIME PROCESSING)

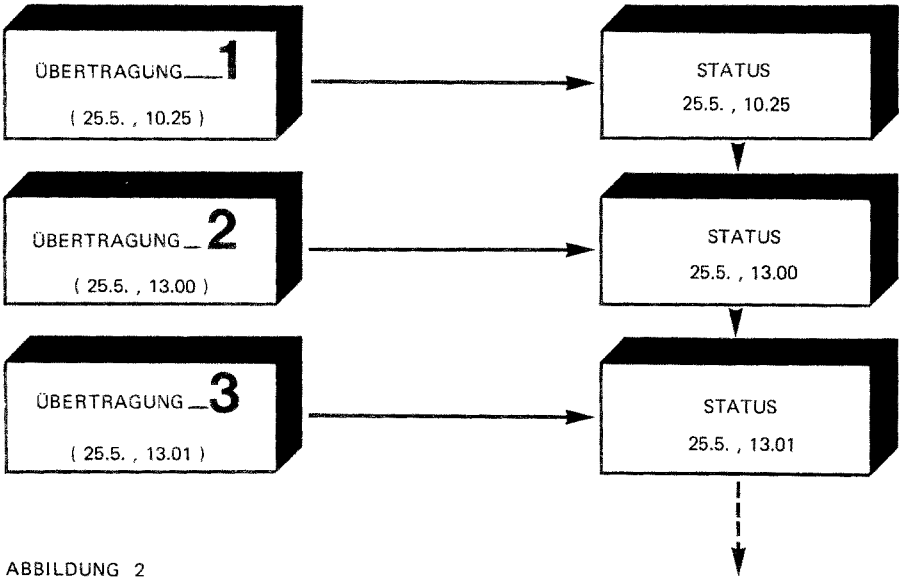
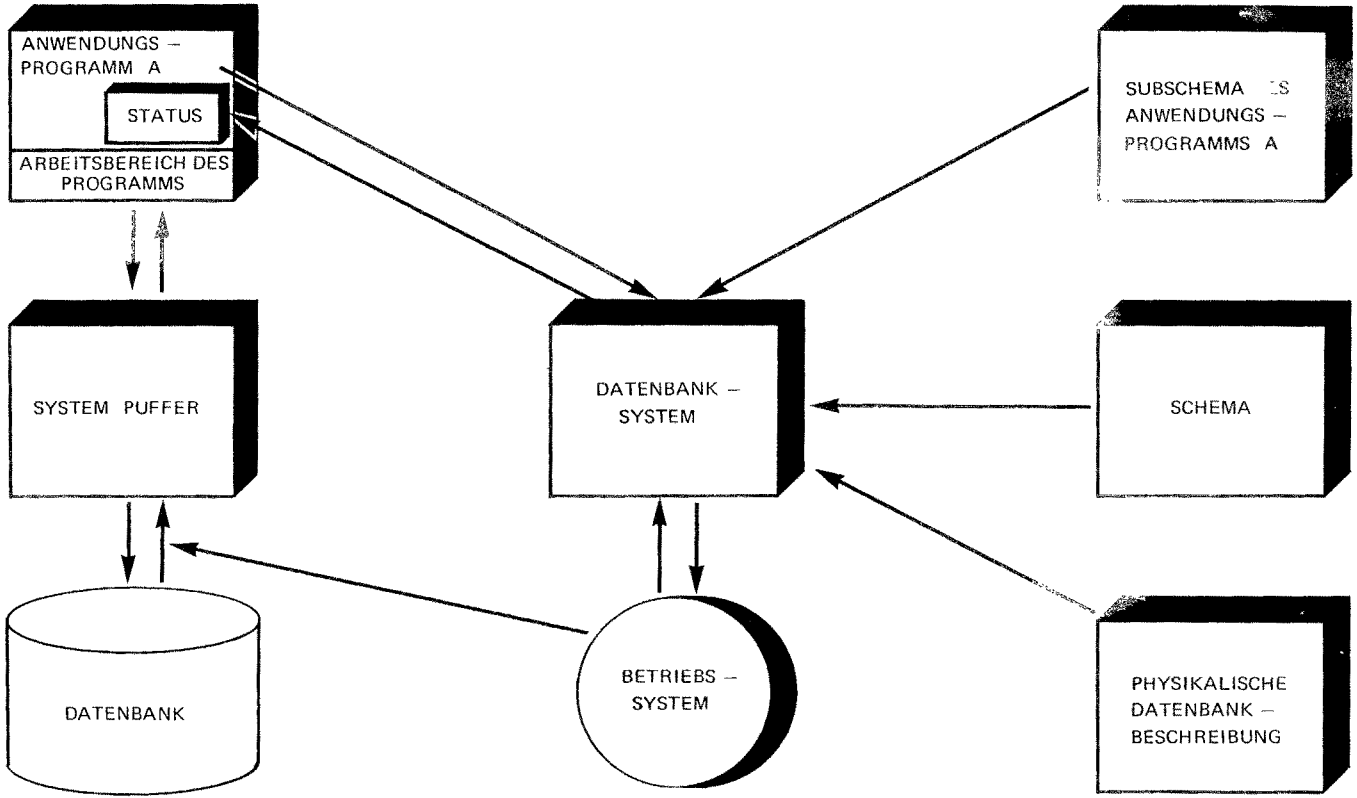


ABBILDUNG 2

	Operational Systeme	Informations-Systeme
Zugriff	geplant oder vorausprogrammiert	spontan, nicht vorausprogrammiert
Typische Beispiele	Bankschalter Flugreservierung	Verkaufsanalyse, Personalinformation
Typische Benutzer	Bankschalterbeamte, Vorarbeiter, Unteres Management	Informationsstab, Mittleres Management, Assistenten des höheren Management
Normalzweck	Unterstützung von Routine Operationen	Unterstützung von Planung und dringenden Informationsbedürfnissen
Antwortzeit	Sekunden	Minuten oder Stunden
Implementierer der Anwendung	Programmierer	Informationsspezialist
Implementierungs-zeit	Wochen oder Monate	Stunden
Typische Sprachen	COBOL, FORTRAN, PL/I	IQF, GIS

MERKMALE FÜR DATENBANKSYSTEME (nach James Martin)

Abbildung 3



WIRKUNGSWEISE EINES DATENBANKSYSTEMS

ABBILDUNG 4

ARBEITSPLATZLISTE

REFERENZ – NUMMER	ARBEITGEBER – NUMMER	GEFORDERTES TALENT	ANFANG	ENDE	BEZAHLUNG	STATUS

ARBEITGEBERDATEI

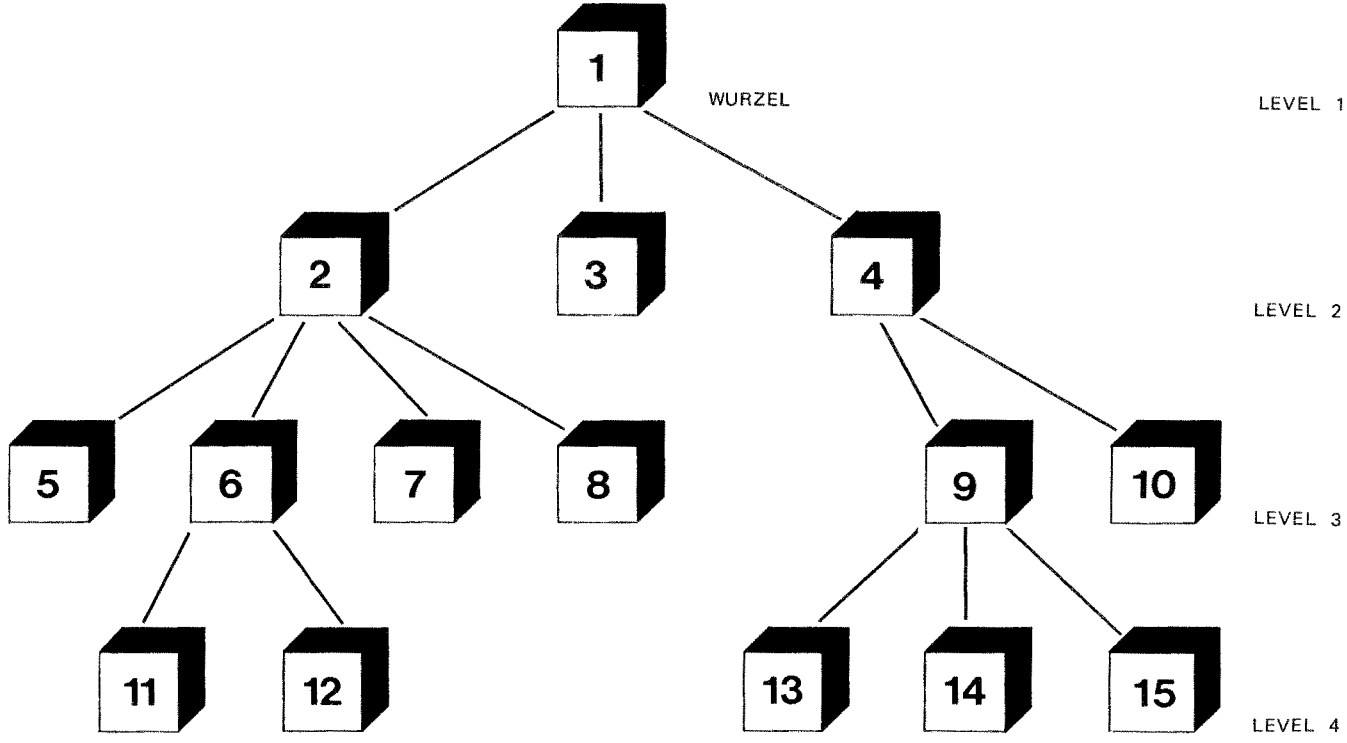
ARBEITGEBER – NUMMER	NAME	ADRESSE	ARBEITSKLIMA	SOZIALE LEISTUNGEN

TALENT DATEI

TALENT GRUPPE	NAME	ADRESSE	VERFÜGBARKEIT	AUSBILDUNG	ERFAHRUNG	PERSONAL – DATEN	GEHALT

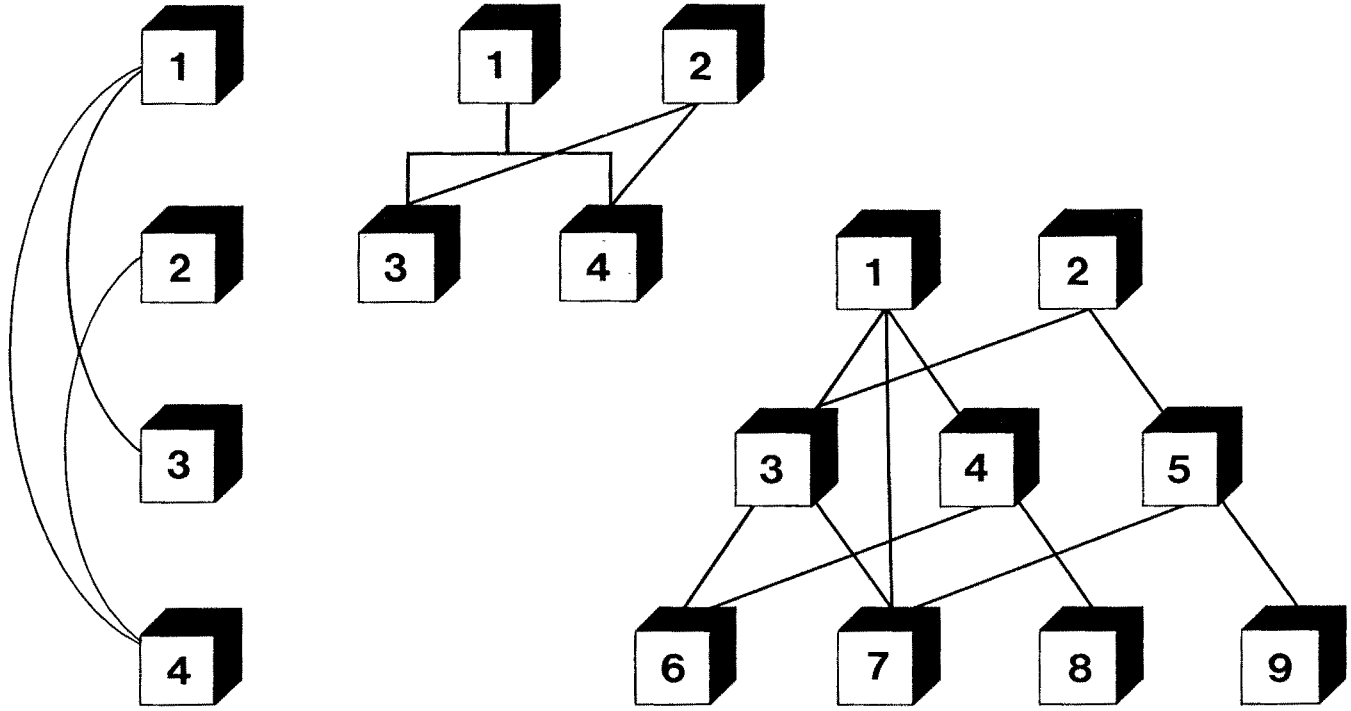
AUFGLIEDERUNG EINER DATENBANK ARBEITSPLATZBESCHAFFUNG

ABBILDUNG 5



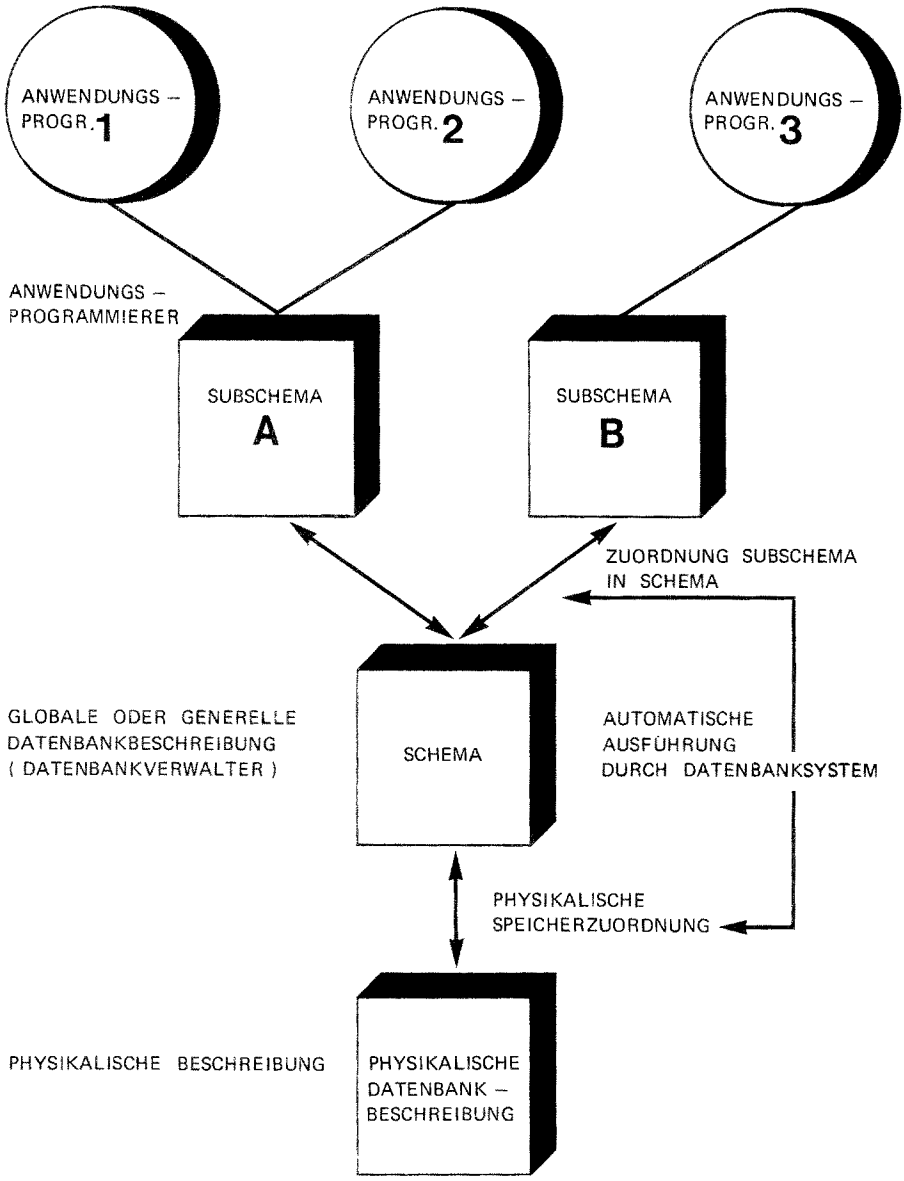
HIERARCHISCHE DATENBANKSTRUKTUR

ABBILDUNG 6



DATENBANKNETZWERKSTRUKTUREN

ABBILDUNG 7



LEVEL DER DATENBESCHREIBUNGEN

ABBILDUNG 8