chef et sera remplacé par Monsieur le professeur Matthias. Je saisis l'occasion qui m'est offerte pour remercier vivement M. le prof. Conzett de sa collaboration efficiente et de l'excellent travail qu'il a fourni durant de nombreuses années. Je souhaite la bienvenue à M. le prof. Matthias dans sa nouvelle fonction et lui adresse ma gratitude pour avoir accepté cette nouvelle charge. Mes bons vœux et ma reconnaissance vont aussi aux rédacteurs qui restent à leur poste, ainsi qu'à Madame Wieser qui s'occupe avec dévouement et compétence du secrétariat de la rédaction.

Pour terminer, je salue au nom de la SSMAF toutes les associations partenaires dans l'édition de la revue. C'est en poursuivant notre collaboration harmonieuse que nous parviendrons à une cohésion efficace pour la promotion de tout notre secteur professionnel auprès d'un large public.

Je vous souhaite à tous, bien sincèrement et avec conviction, une «Bonne et heureuse année».

Le président central SSMAF:
A. Frossard

reicht, die Nachfolge übernimmt Herr Prof. Matthias. Ich benütze gerne die Gelegenheit, an dieser Stelle Herrn Prof. Conzett für seine während vielen Jahren geleistete ausgezeichnete Arbeit herzlich zu danken. Gleichzeitig begrüsse ich Herrn Prof. Matthias in der neuen Funktion als Chefredaktor und danke für seine Bereitschaft zur Übernahme dieses Amtes. Mein Dank und meine besten Wünsche richten sich ebenfalls an die bleibenden Redaktoren und ganz speziell auch an Frau Wieser, die sich in umsichtiger und kompetenter Weise den vielfältigen Arbeiten des Redaktionssekretariates annimmt.
Zum Schluss entbiete ich im Namen des SVVK allen mit uns in der Redaktion zusammengeschlossenen Fachvereinen die besten Glückwünsche. Mit der Fortführung einer harmonischen Zusammenarbeit zwischen unseren verschiedenen Fachvereinen erreichen wir gegen aussen jene Geschlossenheit, die zur Vertretung unserer gemeinsamen Anliegen in der Öffentlichkeit nötig ist.
Ich wünsche allen ein gutes und glückliches neues Jahr.

Der Zentralpräsident SVVK:
A. Frossard

che qui mi si è data, per ringraziare vivamente il Professor Conzett della sua preziosa e valida collaborazione e dell'eccellente lavoro svolto durante innumerevoli anni. Auguro il benvenuto al Professor Matthias nella sua nuova veste e gli porgo i sensi della nostra gratitudine per aver accettato questo nuovo mandato.

I miei migliori auguri e la mia riconoscenza vanno anche indirizzati ai redattori rimasti in carica, alla Signora Wieser, che si occupa con zelo e competenza del segretariato della redazione.

Per concludere, a nome della SSCGR, saluto tutte le associazioni partecipanti alla nostra rivista. Siamo convinti, che proseguendo in una collaborazione armoniosa, noi raggiungeremo una coesione efficace alfine di promuovere ogni settore della nostra professione e nello stesso tempo sensibilizzare i vari stratti dell'opinione pubblica.

A Voi tutti, i più sinceri e cordiali auguri di Anno Nuovo.

Il presidente centrale della SSCGR:
A. Frossard

Anforderungen an Datenbanksysteme zur Verwaltung grosser raumbezogener Datenbestände*

A. Frank

Einleitend werden die besonderen Anforderungen an die Datenverwaltung im Vermessungswesen hervorgehoben, bevor auf die zentrale Frage «warum Datenbanken?» eingegangen wird. Wichtige Methoden im Rahmen dieser Studie sind das sog. Schichtenmodell und das Transaktionenkonzept. Vorerst werden aber noch die der Arbeit zugrundegelegten Annahmen klargestellt: Technische Annahmen, Gliederung der Verarbeitung, Softwaretechnik, Anwendungsgebiete.
In den nachfolgenden Kapiteln werden die einzelnen Komponenten der raumbezogenen Datenbank behandelt: Datenspeicherung (physische Bündelung, Pufferung), Datensicherheit (Transaktionen, Mehrfachbenützer, Datenverlust, unberechtigte Benützer), Zugriffsmethoden (eindeutige Schlüssel, Hierachien, verallgemeinerter Zugriff, raumbezogener Zugriff), Datenmodelle, abstrakte Datentypen, geometrische Daten mit entsprechenden Konsistenzbedingungen, interaktive Abfragesprachen. Abschliessend werden die sich aufdrängenden Schlussfolgerungen gezogen.

En guise d'introduction cet article présente les exigences faites au traitement des données dans le domaine des mensurations, puis aborde la question centrale «pourquoi les banques de données?». Les soi-disants modèles conçus en couches successives et le concept de transaction représentent des méthodes importantes dans le cadre de cette étude. En premier lieu certains principes sont expliqués: hypothèses techniques, répartition du traitement, technique du logiciel d'application.

1. Einleitung

Informationssysteme, welche sich mit Daten befassen, die in der Realität einem bestimmten Ort zugeordnet werden können – hier als *raumbezogene Daten* bezeichnet – werden heute zunehmend diskutiert. Verwaltungstätigkeiten auf verschiedenen Ebenen wie auch Raumplanung oder Wissenschaft benötigen Fakten, die in Beziehung zum Raum stehen. Entsprechende Informationssysteme findet man unter verschiedenen Titeln, z. B. Geographische Informationssysteme, Landinformationssysteme, Mehrzweckkataster usw. Diese Abhandlung konzentriert sich auf allgemeine Aspekte von Systemen mit

*Vortrag vom 20. August 1984 am Internationalen Symposium für räumliche Datenverarbeitung in Zürich; Übersetzung Institut für Geodäsie und Photogrammetrie, ETH Zürich

Les chapitres suivants traitent des différents composants d'une banque de données à référence spatiale: stockage des données (liens physiques, mémoire-tampon), protection des données (transactions, utilisateurs simultanés, perte, emploi abusif), méthode d'accès (clé unique, hiérarchies, accès généralisé, accès spatial), modèle des données, types de données abstraits, données géométriques avec les conditions de consistance correspondantes, les languages interactifs d'interrogation. Pour conclure l'auteur tire les conclusions qui s'imposent.

raumbezogenen Daten. Sie werden als raumbezogene Informationssysteme bezeichnet, ohne dass Unterschiede zwischen Systemen mit verschiedenem Zweck in Betracht gezogen werden. Ein Versuch einer Klassierung ist andernorts vorgelegt worden (Frank 1980). Im Vordergrund stehen Systeme, welche Daten mit einem exakten Ortsbezug speichern und Geometrie mit Punkten und Vektoren beschreiben. Wir glauben, dass solche Vektorsysteme für viele Anwendungen vorteilhaft sind, besonders wenn man in Betracht zieht, wieviel Speicherplatz Rastersysteme benötigen. Das soll nicht andere System-Typen ausschliessen – für bestimmte Aufgaben sind Rasteroperationen offensichtlich vorteilhaft – aber Vektorsysteme scheinen sich von anderen in wesentlichen Punkten zu unterscheiden und rechtfertigen deshalb eine besondere Behandlung.

Raumbezogene Informationssysteme erfordern grosse Datensammlungen, welche permanent in Massenspeichern (meist Festplattenspeicher) gespeichert sind und greifen auf kleine Teilmengen dieser Sammlung zu, um damit einfache Operationen durchzuführen. Schon früh in der Geschichte der Datenverarbeitung wurde den Benützern aus verschiedenen Anwendungsgebieten bewusst, dass zwischen ihren Bedürfnissen, Daten zu speichern und für die spätere Verarbeitung auf diese Daten zuzugreifen, Ähnlichkeit besteht. Statt für solche Funktionen bei jeder Anwendung immer wieder ähnliche Prozeduren zu schreiben, wurde versucht, ein allgemein anwendbares Programm zu erstellen, das diesen Dienst zu versehen hätte. Die Idee der (allgemeinen) Datenverwaltungssysteme war damit geboren (CODASYL, 1962, 1971).

Diese Anstrengungen wurden von der kommerziellen Datenverarbeitung gefördert und waren entsprechend auf die Behandlung grosser kommerzieller Datenmengen ausgerichtet. In der Vergangenheit beschränkten sich Forschungsarbeiten auf kommerzielle Datensammlungen, und erst in der neuesten Literatur findet man Diskussionen über ‹nonstandard›-Datenbankanwendungen wie Bibliothek-Datenbanken, Datenbanken für Computer Aided Design, geographische und Landinformationssysteme, die hier abgekürzt nicht-kommerzielle Datenbankanwendungen genannt werden sollen (Meier, A., Zehnder, C. A., 1980), (Härder und Reuter, 1982), (Scheck und Lum, 1983). Einerseits hat die Forschung auf dem Gebiet der raumbezogenen Informationssysteme gezeigt, dass solche Systeme ähnliche Aufgaben wie die Standarddatenbankverwaltungssysteme erfüllen müssen. Es scheint, dass Techniken, wie sie in kommerziellen Datenbanksystemen angewendet werden, auch hier vorteilhaft sein können (Frank and Tamminen, 1982). Anderseits wurde die Forschung in der allgemeinen Datenbanktheorie angeregt, sich mit den Anforderungen von nicht-kommerziellen Anwendungen zu befassen. Nicht-kommerzielle Anwendungen zeigen neue Anforderungen an Datenbank-Funktionen, und dieser Aufsatz möchte diese besonderen Anforderungen von grossen raumbezogenen Datenbanken an generelle Datenbanksysteme untersuchen.

Typische Eigenschaften eines Datenbankverwaltungssystems werden von den Anforderungen der vorgesehenen Anwendungsgebiete bestimmt, und Systeme, die auf kommerzielle Anwendungen ausgerichtet sind, eignen sich deshalb nicht automatisch für nichtkommerzielle Anwendungen. Aus den Aussprachen an zwei kürzlich abgehaltenen Internationalen Paneldiskussionen über nicht-kommerzielle Datenbanksysteme ergibt sich, dass Datenbanksysteme als Werkzeugkasten zu bauen sind, aus denen für eine bestimmte Anwendung Bausteine ausgewählt werden können, die genau den jeweiligen Bedürfnissen entsprechen. Dies nicht nur, um die von der Anwendung benötigten Funktionen zusammenzustellen, sondern vor allem auch, um optimale Implementierungen dieser Funktionen anzubieten, und endlich, um unnötige Funktionen auszuschliessen, die nur die Komplexität und die Kosten der Anwendung erhöhen.

Als Rahmen, um Kompatibilität zwischen den Bausteinen sicherzustellen, verwenden wir ein hierarchisch geschichtetes Modell, ähnlich dem weitherum akzeptierten Modell, das die Kommunikation zwischen Computern verschiedener Hersteller beschreibt (ISO/TC97/SC16).

Dieser Aufsatz ist das Ergebnis der Arbeit mehrerer Jahre an der ETH Zürich und nun an der Universität von Maine in Orono, die der Anwendung des Datenbankkonzepts für raumbezogene Daten gewidmet war. Die Ideen, über die hier berichtet wird, basieren auf Erfahrungen mit dem PANDA-Datenbankverwaltungssystem, das wir aufbauten (Frank, 1982a), und beschreiben sowohl Methoden, die erfolgreich eingeführt wurden, als auch Kritik an Methoden, die sich nicht bewährten und in Zukunft ersetzt werden.

2. Gründe für die Anwendung von Datenbankverwaltungssystemen

Die allgemein zu beobachtende Tendenz, von der Stapelverarbeitung zu interaktiven, dialogorientierten Anwendungen überzugehen, ändert die Anforderungen für die Datenspeicherung: viele Anwenderprogramme greifen in unvorsehbarer Reihenfolge zum Teil auf die gleichen Dateien zu. Zunehmend verlangen Benützer unmittelbaren Zugriff zu grossen Datensammlungen.

Die in einer Datenbank aufbewahrten Daten sind kostbar, weil grosse Anstrengungen notwendig sind, um solche Daten zu sammeln, ins System einzufügen und sie nachzuführen. Diese Daten müssen für eine lange Zeitperiode und für verschiedene Anwendungen verfügbar sein, um diesen Aufwand zu rechtfertigen. Dazu kommt, dass während der Lebensdauer der gespeicherten Daten neue, nicht voraussehbare Änderungen ihrer Verwendung auftreten werden.

Unter diesen Umständen kann die einfache traditionelle Datei-Struktur, die für eine spezielle Anwendung entworfen wurde, nicht mehr genügen. Speicherstrukturen, die sich für den allenfalls gleichzeitigen Zugriff von mehreren Anwendungsprogrammen eignen, sind indessen umständlich zu programmieren, und es dürfte unzweckmässig sein, diese Arbeit mehrfach zu wiederholen.

Ein allgemein verwendbares Datenbankverwaltungssystem muss folgenden Anforderungen genügen:

- Speicherung und Abfrage von Daten; Auswahl von Daten nach verschiedenen und sich ändernden Schlüsselwörtern.
- Standardisierter Zugriff zu den Daten und damit Trennung der Datenspeicherungs- und -abfragefunktionen von den Programmen, die Daten auswerten. Diese Forderung macht die Datenbank und die Anwenderprogramme voneinander unabhängig; Änderungen der einen Komponente führen nicht automatisch zu Änderungen in der andern. Diese Unabhängigkeit ist wichtig, damit man die unvermeidlichen Änderungen während der Lebensdauer einer Datenbank mit wenig Aufwand vornehmen kann.
- Schnittstellen zwischen Datenbank und Anwenderprogrammen müssen auf einer logischen Beschreibung der Daten beruhen und verhindern, dass

Anwenderprogramme Eigenschaften der physischen Speicherstruktur ausnützen. Indem man die Zugriffsfunktionen für die Anwenderprogramme von der physischen Speicherstruktur unabhängig macht, wird erreicht, dass Anpassungen, die den Speicherbedarf vergrössern, die Anwenderprogramme nicht beeinflussen.
- Gleichzeitiger Zugriff von verschiedenen Personen zu den Daten muss erlaubt sein.
- Massnahmen, um Konsistenzbedingungen für die Daten einzuführen, werden automatisch unterstützt. Konsistenzbedingungen sind Regeln, welche von allen gespeicherten Daten eingehalten werden müssen. Sie sind eine ausgezeichnete Methode, um die Anzahl Fehler in grossen Datensammlungen zu vermindern.

Eine leicht abweichende, weiter ausgearbeitete Aufzählung über Datenbankfunktionen findet man in (Codd, 1982).
Der Zugriff zu den Daten sollte sowohl mit einer hochentwickelten Programmiersprache als auch mit einer benützerfreundlichen Abfragesprache möglich sein. Die Integration der Datenbankmanipuliersprache in eine hochentwikkelte Programmiersprache ist entscheidend für die bequeme Handhabung und kann helfen, schwer herauszufindende Fehler zu vermeiden. Eine eigenständige Abfragesprache hilft dem gelegentlichen Benützer, wenn er Daten aus der Datenbank absuchen und wenn er auf ‹ad hoc›-Fragen ohne jede formale Programmierung Antwort erhalten will. Dies erlaubt, die Datenbank für diese Art von Fragen leicht zugänglich zu machen.
Alle diese grundlegenden Anforderungen sind sowohl für die kommerzielle Datenverarbeitung als auch für raumbezogene Datensammlungen gültig. Allerdings sind Einzelheiten bei diesen Anforderungen, besonders für die Behandlung von geometrischen Objekten und für die Herstellung von graphischen Ausgaben, im Bereich der Datenbanken neu.

3. Methoden

Das Rahmenkonzept dieser Studie bildet eine Hierarchie von Moduln, wobei jedes Modul bestimmte Aufgaben für die nächste übergeordnete Schicht erfüllt. Die tiefste Schicht bezieht sich direkt auf die Funktionen, die das Betriebssystem zur Verfügung stellt, während die höchste Schicht ihre Dienstleistungen dem menschlichen Benützer für den Dialog anbietet.
Die tiefsten Schichten speichern Daten, indem sie Operationen des Betriebssystems einsetzen, um Zugriff zu den Dateien zu erhalten. Diese Schichten befassen sich hauptsächlich damit, die Zugriffsleistung durch Bündelungs- und Pufferungstechniken zu beschleunigen. Die angebotenen Dienstleistungen sind die Operationen ‹speichern› und ‹holen› für Datenelemente (records), wobei die Datenelemente durch interne Schlüssel bezeichnet werden.
Die nächsten Schichten erfüllen im wesentlichen die gleichen Aufgaben, aber sie sorgen für deren *sichere* Ausführung. Die Gliederung von zusammenhängenden Änderungen in der Datenbank in *Transaktionen* bedeutet, dass jede Änderungs-Transaktion durch einen bestandenen Konsistenztest erfolgreich abgeschlossen wird. Die Datenbank wird so gegen (Konsistenz-) Verlust und gegen Einwirkungen anderer Benützer abgesichert, indem nur Änderungen, die die Konsistenz der Datenbank nicht beeinträchtigen, ausgeführt werden können.
Die dritte Schicht fügt verschiedene Arten von Zugriffsmethoden hinzu, z. B. Methoden, um Daten anhand bestimmter Schlüsselwerte oder aufgrund der räumlichen Zuordnung herauszusuchen.
Die vierte Schicht bietet einen Mechanismus an, um Daten nach den Regeln des logischen Schemas zu strukturieren und diese Beziehungen zur Vereinfachung der Programmierung auszunützen. Diese Leistungen werden als Erweiterung der höheren Programmiersprache oder der unabhängigen Abfragesprache angeboten.

4. Annahmen

Diese Abhandlung baut – wie die meisten Abhandlungen – auf einer Reihe von Annahmen auf. Einige davon sollen hier ausdrücklich erwähnt werden, um Verwirrungen beim Leser vorzubeugen.

4.1 Technische Annahmen

Die Ausführungen in diesem Aufsatz beziehen sich auf die heute erhältliche Computertechnik, welche weiterentwikkelt ist, als sie sich in manchen installierten Systemen vorfindet. Hingegen schliessen wir hier experimentelle Konstruktionen, die vielleicht in wenigen Jahren erhältlich sein werden, aus.

Prozessoren: Wir unterstellen die Von Neumann Hardware Architektur und berücksichtigen den Einfluss künftiger Parallel-Multiprozess-Systeme nicht. Es scheint noch nicht Einigkeit darüber zu herrschen, wieweit eine solche Erweiterung für Datenbankoperationen vorteilhaft wäre (deWitt and Hawthorn, 1981).

Speichermedien: Wir unterstellen die Verwendung eines Hauptspeichers (früher ‹core›), auf den der Prozessor direkt zugreift und der vom langsameren Massenspeicher getrennt ist. Daten im Massenspeicher sind nur zugänglich, nachdem sie in den Hauptspeicher übertragen worden sind. Der Zugriff zu Daten im Massenspeicher ist viel langsamer als zu Daten, welche im Hauptspeicher sind (entsprechende Zugriffszeiten stehen typischerweise im Verhältnis 10 000 : 1). Der Zugriff geschieht in grösseren Paketen mit einer Zugriffszeit, die im wesentlichen unabhängig vom Umfang des Datenpaketes und konstant ist (typisch 30 msec pro Zugriff).

Gliederung der Verarbeitung: Daten, die in einem zentralen Computer gespeichert sind, können für die Benützung in eine ‹intelligente› Arbeitsstation mit eigenem Prozessor und eigenem Programm übertragen werden. Die Verteilung von Daten auf verschiedene zusammengeschlossene Computer soll aus dieser Diskussion ausgeschlossen werden; es scheint, als ob für die wichtigsten Probleme dieser Anordnung noch keine genügend allgemeine Lösung gefunden worden ist (Lampson et al., 1981).

Software-Technik: Die Betrachtungen, welche in dieser Abhandlung angestellt werden, beziehen sich auf allgemein anerkannte Programmiertechniken mit transportablen hochentwickelten Programmiersprachen. Unsere Implementierung ist in PASCAL geschrieben mit einem Precompiler, um die modulare Programmierung zu erleichtern und um den Code für verschiedene Hardware (IBM 370 VS/CMS, DECsystem-10, PERQ) verwendbar zu machen.

4.2 Zu den Anwendungsgebieten

Grösse der Datensammlung: Ein raumbezogenes Informationssystem enthält gewöhnlich eine sehr grosse Datensammlung. Insbesondere nehmen wir an, dass die Datensammlung zu gross ist, um im Hauptspeicher Platz zu finden und deshalb in einem Massenspeicher untergebracht werden muss.

Anzahl verschiedener Datentypen: Es wird angenommen, dass sich die Daten aus einer kleinen Anzahl (<1000) von verschiedenen Datentypen zusammensetzen, von diesen wenigen Datentypen aber sehr viele verschiedene Einzelexemplare vorkommen.

Viele Anwendungen: Wir nehmen weiter an, dass die gespeicherten Daten für eine grosse Anzahl verschiedener Anwendungen verwendet werden sollen.

Diese drei Annahmen sind typisch für die Anwendung von Datenbanksoftware. Unter anderen Umständen gibt es manchmal einfachere, billigere und angemessenere Lösungen.

aumbezogene Daten: Es ist typisch für ein raumbezogenes Informationssystem, dass es Daten speichert, die eine bekannte Beziehung zum Raum haben, und dass die Verarbeitung dieser Daten diese räumliche Beziehung ausnützt. Die Lage im Raum wird durch Koordinatenwerte in einem gegebenen Koordinatensystem ausgedrückt.

Geometrische Daten in Vektordarstellung: Teile der Daten in raumbezogenen Informationssystemen beschreiben die geometrische Form und Ausdehnung von Objekten im Raum. Die Auswertung untersucht geometrische Eigenschaften, und häufig werden die Resultate als Pläne herausgegeben. Geometrische Formen werden dargestellt, indem gegebene Punkte mit Vektoren verbunden werden.

nteraktives Abfragen für die Plandarstellung: Um einen Plan auf einem interaktiven Terminal-Bildschirm zu zeichnen, müssen sehr viele Datenelemente (records) herausgesucht werden. Schätzungen von Punktdichten anhand von erzeugten Auswertungen ergeben über 2000 Records, hauptsächlich Punkt- und Linien-Records, aber auch viele verschiedene Records, die sichtbare Einheiten (z. B. Häuser, Strassen und Dörfer) beschreiben. Das Heraussuchen von allen Daten, die für einen Plan notwendig sind, sollte innerhalb 20 Sekunden, wenn möglich aber schneller, abgeschlossen sein, damit wirklich interaktives Arbeiten ermöglicht wird.

5. Anforderungen an die Datenspeicherungs-Schicht

Diese Schicht kommuniziert mit den vorhandenen Speicheroperationen, wie sie vom Betriebssystem zur Verfügung gestellt werden. Ihr Hauptzweck ist, die Geschwindigkeit von Speicherung und Abfrage zu verbessern. Die Anforderungen für diese Schicht ergeben sich aus der maximal zulässigen Wartezeit für die Zeichnung eines Planes auf dem Bildschirm. Einige Tests ergeben, dass etwa 2000 Datensätze für einen Planausschnitt nötig sind: um diese in 20 Sekunden herauszusuchen, muss die mittlere Zugriffszeit für einen Satz kleiner als 10 msec sein. Die Zugriffszeit bei physischer Speicherung auf Platten ist, wenn die Umtriebe des Betriebssystems eingeschlossen werden, in der Grössenordnung von 50–200 msec. Deshalb muss diese Schicht die Anzahl der physischen Zugriffe, die nötig sind, um die Datensätze für die Planzeichnung herauszusuchen, reduzieren. Dazu sind zwei technische Prinzipien bekannt:

- Datenbündelung (clustering)
- Pufferverwaltung.

Es ist wünschbar, dass die Schnittstelle zum Betriebssystem einfach ist und nur Standardfunktionen verwendet, die in vielen Betriebssystemen vorhanden sind (‹lesen› und ‹schreiben› in bzw. von ‹random access files›). Damit wird das Datenbankverwaltungssystem transportabel und seine Anwendung viel einfacher. Indessen muss angemerkt werden, dass einige Verfahren, die im Betriebssystem verwendet werden, um die Leistung der Zugriffsoperationen auf Plattenspeicher zu steigern, sich auf Datenbank-Zugriffszeiten nachteilig auswirken können, so dass es oft vorteilhafter ist, direkte Plattenzugriffsoperationen auf tieferem Niveau zu verwenden.

5.1 Physische Bündelung (clustering)

Ein physischer Zugriff auf den Plattenspeicher erfasst eine grössere Masse von Daten, die man meistens als ‹Seite› bezeichnet. Wenn wir unsere Datensätze auf der Platte so anordnen können, dass jede ‹Seite› mehrere Datensätze enthält, die für den Plan nötig sind, wird diese Methode die Anzahl der physischen Zugriffe auf die Platte um die mittlere Anzahl von verwendbaren Datensätzen pro ‹Seite› herabsetzen.

Das ist in allen Fällen möglich, wo eine vernünftige Vorhersage darüber möglich ist, welche Daten meistens gemeinsam verwendet werden. Diese Daten werden dann benachbart gespeichert und bilden zusammen ein *physisches Bündel* (cluster) (Salton und Wong, 1978). Für raumbezogene Zugriffe bei Planzeichnungen sind solche Voraussagen glücklicherweise leicht; sie beziehen sich auf die *Nachbarschaft* der Objekte. Für Pläne suchen wir Daten von Objekten heraus, welche sich in einem bestimmten Bereich befinden. Wenn wir ein Datenelement eines Objektes heraussuchen, ist die Chance gross, dass die Daten anderer benachbarter Objekte auch benötigt werden. Wenn solche Daten zusammengebündelt und mit *einem* physischen Zugriff herausgeholt werden, können wir erreichen, dass die Plandaten innert so kurzer Zeit gefunden werden, dass interaktive Arbeit möglich ist.

Ein Datenbanksystem für raumbezogene Daten muss daher auf alle Fälle die physische Bündelung von Daten einbeziehen (ein typischer Speicher-Befehl müsste etwa lauten: ‹speichern in der Nähe von x›, wobei x einen bereits gespeicherten Datensatz bezeichnet). Diese Anforderung schliesst die Implementierung von vielen einfacheren relationalen Datenbanken aus, bei denen die physische Speicherung durch einen Primärschlüssel gesteuert wird und durch den Benützer nicht beeinflusst werden kann. Es scheint auch vorteilhaft, wenn Daten verschiedener Art (z. B. Häuser, Strassen, Flüsse) auf der selben ‹Seite› gespeichert werden können und nicht verschiedene physische Zugriffe erfordern. Indessen ist es weder nötig noch wünschbar, dass die physische Bündelung von der Benützerschnittstelle aus ‹sichtbar› ist; vielmehr sollte sie als internes Verfahren für schnellen Zugriff verstanden werden, um das sich der Benützer nicht kümmern muss.

5.2 Pufferung

Viele Programme zeigen eine besondere ‹lokale› Zugriffscharakteristik, die darin besteht, dass die gleichen Datenelemente während einer kurzen Zeitperiode mehrfach verwendet und dann lange nicht mehr benötigt werden. Wenn diese Datenelemente in einem Puffer zur Verfügung stehen, kann die Anzahl der physischen Zugriffe zu einem langsameren Massenspeicher reduziert werden: für jedes Datenelement ist dann nur ein erster Zugriff zum langsamen Speicher nötig, und alle folgenden Abfragen können über den Puffer befriedigt werden. Diese Strategie wird in Computern allgemein verwendet (virtueller Speicher, cache memory etc.).

Eine Datenbank enthält gewöhnlich eine einzige Ebene, um ‹Seiten›, die vom Plattenspeicher eingelesen werden, zu puffern. Dies ist unerlässlich, um die physische Bündelung der Daten auf dem Massenspeicher auszuwerten.

Programme, welche raumbezogene Daten behandeln, zeigen bezeichnenderweise ausgeprägte lokale Zugriffscharakteristiken, verwenden aber eine grosse Menge an Datensätzen (alle Sätze für eine Planzeichnung). Dies ist besonders deutlich bei interaktiven Programmen. Die Benützer neigen dazu, in einem dargestellten Planausschnitt mehrere Interaktionen vorzunehmen, bevor sie zu einem anderen Planbereich übergehen. Sehr oft verlangen sie zuerst einen Übersichtsplan, ‹fokussieren› ihn dann auf eine interessierende Einzelheit und beginnen dort zu arbeiten.

In unserem Datenbankverwaltungssystem PANDA, mit dem wir experimentieren, ist eine zweite Pufferebene für einzelne Datenelemente eingebaut. Im allgemeinen wird die Grösse dieses Puffers auf rund 3000 Sätze festgelegt, was erlaubt, eine vollständige Planzeichnung im Puffer zu speichern. Eine erneute Aufzeichnung dieses Planausschnittes verlangt keinen zusätzlichen physischen Zugriff und ist deshalb sehr schnell.

Wir fanden, dass dieses Konzept wesentlichen Einfluss auf den Programmierstil eines Anwenderprogrammes hat: es wird darauf verzichtet, Daten,

die in der Datenbank gespeichert sind, als Programmvariable einzuführen, da ein erneuter Zugriff auf diese Daten immer sehr schnell ist. Zum Beispiel verwenden wir keine ‹linear display lists› zur Steuerung der Graphik, sondern zeichnen Pläne direkt von den Datenbanksätzen aus.
Ein Datenbankverwaltungssystem sollte auch Hilfsprogramme umfassen, die helfen, optimale Puffergrössen für bestimmte Aufgaben zu bestimmen.

6. Anforderungen an die Datensicherheit

Die Leistungen, die von kommerziellen Datenbankverwaltungssystemen angeboten werden, genügen im allgemeinen den Anforderungen an die Datensicherheit. Hingegen genügen viele Systeme für kleinere Mikro- oder Personalcomputer wie auch Systeme, die auf geometrische und geographische Daten ausgerichtet sind, sehr oft den Ansprüchen nicht, die gestellt werden müssen, um grosse Datensammlungen über längere Zeiträume zu verwalten.
Die Schicht, die in diesem Kapitel beschrieben wird, fügt keine neuen Funktionen ein, sie erhöht aber die Sicherheit der Operationen. Die meisten dieser Vorrichtungen werden die Leistung reduzieren; das ist der Preis, den wir für die gewonnene Sicherheit zu zahlen haben, den Fünfer und das ‹Weggli› gibt es auch hier nicht.

6.1 Transaktionen

Der Inhalt einer Datenbank muss einige Bedingungen, sog. Datenintegritätsbedingungen, erfüllen. Sie helfen, die langfristige Verwendbarkeit der Daten zu gewährleisten, indem sie fehlerhafte Daten identifizieren, bevor sie gespeichert werden. Der Benützer eröffnet eine Transaktion, wenn er Änderungen in eine Datenbank einführen will. Dann ändert er die Datensätze, welche durch diese Nachführung betroffen werden.
All diese Änderungen gelten so lange als Versuch, bis sie der Benützer als gültig bestätigt. Dann prüft das System, ob die neuen oder veränderten Daten die Integritätsbedingungen nicht verletzen. Erst nach diesen Prüfungen werden die Änderungen in die Datensammlung als dauernde Nachführung eingefügt. Jede Transaktion führt die Datenbank von einem konsistenten Anfangszustand in einen neuen, ebenfalls konsistenten Zustand über; dementsprechend ist es unmöglich, dass die Datenbank inkonsistent wird.
Das Transaktionenkonzept ist für den Entwurf von interaktiven Nachführungsprozeduren sehr geeignet; der Benützer kann alle seine beabsichtigten Änderungen entwerfen und hat dann immer noch die Wahl, ob er sie tatsächlich einführen will.

6.2 Mehrfach-Benützer

Grosse Datenmengen sind mächtige, aber teure Ressourcen, welche bestmöglichst genützt werden müssen. In vielen Fällen muss mehr als ein Benützer im gleichen Zeitpunkt auf die Datenbank zugreifen können. Das ist kein schwieriges Problem, solange alle Benützer nur Daten *abfragen*, es wird aber gefährlicher, wenn ein oder mehrere Benützer *Änderungen* in die Datenbank einführen wollen. Wenn keine besonderen Vorkehren getroffen werden, kann eine Änderung des einen Benützers eine unmittelbar vorher von einem anderen Benützer eingeführte Änderung löschen, oder Änderungen verschiedener Benützer kommen sich in die Quere und erzeugen ungewollte und nicht vorhersehbare Ergebnisse. Es ist ungenügend, sich auf manuelle Vorkehrungen oder auf Abmachungen, die nicht in das Datenbankverwaltungssystem eingeführt sind, zu verlassen, wie das in der Regel für graphische Datenbankverwaltungssysteme empfohlen wird. Automatische Kontrollen für Mehrfachbenützer von Datenbanken basieren auf dem Transaktionskonzept: die Wirkung einer Änderung wird andern Benützern erst nach Abschluss der Transaktion zugänglich gemacht. Mehrfachbenützer begegnen nie einer unvollständigen Transaktion, und der Konflikt zwischen zwei gleichzeitig einwirkenden Benützern ist im Transaktionsbereich gelöst (Kung und Robionson, 1981).

6.3 Sicherheit gegen Datenverlust

Grosse Sammlungen von geographischen oder administrativen Daten sind wertvoll und müssen vor Verlusten bewahrt werden. Selbst wenn die Information in anderer, traditioneller Art (Register, Pläne usw.) parallel instandgehalten wird, sind die Übertragungskosten solcher Informationen in maschinenlesbare Form beträchtlich. Um solche Kosten zu vermeiden, ist es nötig, die gespeicherten Daten vor Verlusten zu schützen, die von Fehlern in der Bedienung oder von Hardware- oder Softwaremängeln herrühren.
Um entsprechende Massnahmen festzulegen, müssen wir das Auftreten möglicher Schäden wegen Datenverlusten (Kosten, um die Daten neu einzuführen, Kosten im Zusammenhang mit Betriebsunterbrechungen) mit den Kosten vergleichen, die durch Massnahmen entstehen, die solche Verluste vermeiden sollen. Es wird im allgemeinen angenommen, dass bei den meisten kommerziellen Verarbeitungen der ununterbrochene Ablauf der Operationen wichtig ist, und um das zu gewährleisten, sind sehr aufwendige, aber sichere Methoden erfunden worden. In unserem Anwendungsgebiet mögen niedrigere Ansprüche an die Sicherheit insbesondere längere Unterbrüche, annehmbar sein, aber als minimale Anforderung muss ein Mechanismus, der Datenverluste verhindert, verlangt werden.
In all den Fällen, wo Änderungen nicht nur seltene Ausnahmen sind, sollte dieser Mechanismus die Nachführungen einbeziehen. Eine Änderung, die der Benützer eingegeben und für die er von der Datenbank eine Bestätigung erhalten hat, darf nicht verlorengehen, auch nicht bei Unterbrüchen wegen Hardware- oder Softwareproblemen. Die Transaktion dient hier als Kriterium: nicht bestätigte Änderungen mögen verlorengehen, aber bestätigte sind permanent zu erhalten.
Gewöhnlich unterscheidet man zwei Typen von Problemen:

- Unterbrüche des Datenbankverwaltungsprogrammes, die durch Operationsfehler verursacht sind, und Probleme im Betriebssystem oder bei der Hardware. Solche Unterbrüche treten bei den meisten Installationen ziemlich häufig auf (einmal im Tag .. einmal pro Woche). Während solchen Ereignissen geht der ganze Inhalt des Hauptspeichers verloren und deshalb ist es notwendig, die geänderten Daten in den permanenten Massenspeicher zu schreiben, bevor man eine Transaktion bestätigt.
- Verlust von Speichermedien, ebenfalls durch Operationsfehler oder Hardware-Mängel verursacht (sog. ‹head crashes›). Solche Probleme sind gewöhnlich sehr selten (einmal im Jahr), und aufwendigere Prozeduren, um die Daten wieder einzuführen, sind annehmbar. Die traditionelle Datenverarbeitung verwendet das ‹Zwei-Generationen-System›, wo die Kopien der Daten für mindestens zwei bestimmte Zustände mit allen dazwischenliegenden Nachführungen auf Magnetband aufbewahrt werden.
  Eine ähnliche Methode ist für Datenbankverwaltungen anwendbar: der Datenbestand wird in regelmässigen Intervallen auf Magnetbänder kopiert, und mit allen dazwischenliegenden Änderungen wird nicht nur die Datenbank nachgeführt, sondern die Änderungen werden auch in ein zusätzliches ‹Journal› (log file) geschrieben (tatsächlich werden sie *zuerst* ins ‹log file› geschrieben).

6.4 Schutz gegen unberechtigte Benützer

Einige Datenbanken enthalten vertrauliche oder geheime Information, und es ist nötig, diese gegen unberechtigte Benützer zu schützen. Die Anforderungen sind sehr unterschiedlich:

- Unberechtigten Benützern wird der Zugriff zur Datenbank verwehrt; dies sollte das Betriebssystem besorgen.
- Bestimmten Personen werden nur bestimmte Teile der Datenbank zugänglich gemacht; dazu sind verschiedene Methoden bekannt und in Anwendung, welche verschiedene Ebenen für Zugriffsselektionen und Sicherheit anbieten.
- Dem Benützer wird nur generalisierte, statistische Information zugänglich gemacht, während der direkte Zugriff auf individuelle Daten verwehrt wird: die meisten der bekannten Methoden haben sich als ungenügend erwiesen und sind äusserst aufwendig (Denning und Schlorer, 1980).

7. Zugriffsmethoden

Die bisher besprochenen Schichten, die für die Speicherung und die Abfrage von Daten sorgen, verwenden ein internes Kennzeichen (Datenbankschlüssel), welches einem Datensatz zugeordnet wird, wenn er zum erstenmal gespeichert wird. Der Datensatz selbst wird von diesen Methoden als eine nicht interpretierte konstante Gruppe von ‹bytes› betrachtet, die gespeichert und später reproduziert wird. Für nicht-kommerzielle Anwendungen ist es besonders wichtig, dass das Speichersystem nicht von der Interpretation der gespeicherten Daten abhängt, weil das die Flexibilität der Anordnung der Datenfelder im Satz beschränken und unerwünschte Abhängigkeiten zwischen den verschiedenen Schichten des Systems schaffen würde. Das Subsystem Datenbank einer grösseren Anwendung muss beliebig strukturierte Datensätze ohne Abhängigkeiten von ihrer internen Struktur behandeln können. Die oft angebotene Strukturierung der Datensätze in einer hierarchischen Schicht von Datenfeldern der Basistypen wie z. B. integers, reals und strings sind nicht einmal für kommerzielle Anwendungen immer genügend, wie dies aus den zahlreich angebotenen Erweiterungen (Datentypen für Geld, Datum usw.) hervorgeht. Die Zugriffsmethoden, die in diesem Abschnitt besprochen werden, verwenden Teile der in einem Satz gespeicherten Daten. Um Unabhängigkeit von der Datenbeschreibung zu erreichen, müssen die definierenden Moduln Zugriffsfunktionen an die massgebenden Datenelemente übertragen, welche dann für die Zugriffsmethoden verwendet werden.

Es muss betont werden, dass wir hier logisch definierte Methoden für den Datenzugriff beschreiben. Eine Implementierung wird Algorithmen und Speicherorganisationen umfassen, um diese Methoden zu realisieren und schnelle Antworten zu erreichen.

7.1 Zugriff durch eindeutigen Schlüssel

Benützer brauchen oft Zugriffe zu Datensätzen, welche auf dem Wert eines oder mehrerer Datenfelder (Schlüsselfelder) aufgebaut sind. Im einfachsten Fall erfordert dies, dass die gespeicherten Werte in diesen Feldern eindeutig seien, derart, dass die vorgegebenen Werte den gewünschten Datensatz eindeutig identifizieren (das bildet dann eine Konsistenzbedingung für die Datenbank). Im allgemeinen ist es notwendig, dass man auch mehr als ein Schlüsselwort für einen Satz festlegen kann, so dass der Satz durch verschiedene Werte gefunden werden kann (z. B. Nummer oder Name).

7.2 Zugriff über Hierarchien

Sehr oft wird eine Entität durch einen Wert innerhalb eines Gebietes gekennzeichnet, und der selbe Wert kann in einem andern Gebiet wieder auftreten. Das ist typisch bei Hierarchien wie Staat, Kanton, Stadt. Um Entitäten vollständig zu identifizieren, müssen alle Werte gegeben und nur ihre Kombination muss eindeutig sein.

7.3 Verallgemeinerter Zugriff

Wir fanden, dass der Mensch als Benützer zur Kennzeichnung sehr oft Namen verwendet, die ‹fast› eindeutig sind: Namen von Städten oder Namen von Kunden sind fast immer eindeutig, aber in einzelnen Fällen ist zusätzliche Information nötig, um die gewünschte Entität zu kennzeichnen (z. B. die Stadt mit dem Kanton, der Kunde mit der Adresse).

Aus ähnlichen Überlegungen erlauben wir auch mehr als ein Schlüsselwort, um die selbe Entität abzusuchen. Verschiedene Benützer verwenden oft unterschiedliche Namen zur Kennzeichnung gleicher Dinge. In unserem PANDA-Datenbankverwaltungssystem ist diese Zugriffsmethode implementiert, und sie scheint sehr hilfreich, um eine alltägliche Situation angemessen zu behandeln.

7.4 Raumbezogener Zugriff

Ein grosser Teil der Daten in einem raumbezogenen Informationssystem bezieht sich auf Objekte im Raum. Für diese Objekte sind Ort und Ausdehnung bekannt. Viele Anwendungen benötigen den Zugriff zu den Daten über den Raum (Burton, 1978). Das ist für alle Abfragen bei der Erstellung eines Planes offensichtlich: alle Objekte innerhalb des Planbereiches müssen herausgefunden und dann wiedergegeben werden. Ähnliche Zugriffe sind für interne Operationen nötig, etwa um geometrische Eingaben zu prüfen und um geometrische Umwandlungen von gespeicherten Daten vorzunehmen.

Raumbezogener Zugriff zu Daten geht von einer Abfrage in folgender Form aus:

Zeige alle <Objekt-Typen> innerhalb <Gebiet>, wobei <Gebiet> die Beschreibung des Interessensgebietes ist.

Um eine allgemein wirksame Funktion zu schaffen, scheint es angemessen, das <Gebiet> auf einen rechteckigen, zum Koordinatensystem parallelen Kasten zu beschränken. Detailliertere Abfragen werden dann in zwei Schritten behandelt: in einem ersten Schritt werden alle Daten in der rechteckigen Umhüllung des Gebietes herausgesucht, und der teurere, exakte Auswahlprozess wird auf einen zweiten Schritt verlagert, der sich auf die im ersten Auswahlprozess selektionierten Daten beschränken kann.

Das Ergebnis einer solchen Abfrage kann eine grosse Anzahl von Objekten umfassen. Eine Planzeichnung in der Grösse eines Bildschirms umfasst – wie früher erwähnt – nach unseren Schätzungen etwa 2000 ... 3000 Objekte. Für eine interaktive Anwendung müssen diese Daten in wenigen Sekunden gesucht und dargestellt werden. Das ist nur möglich, wenn vorher eine besondere Datenstruktur aufgebaut wurde; eine sequentielle Abfrage der vollständigen Datenbank dauerte viel zu lang. Zweitens müssen die notwendigen Daten auf dem Plattenspeicher physikalisch gebündelt und mit wenigen disk-Zugriffsoperationen erreichbar sein (wenn jedes Objekt einzeln abgerufen werden müsste, dauerte das ungefähr $2000 \cdot 50\ \text{msec} = 100$ Sekunden).

Mit wenigen Worten: es ist notwendig, eine Abbildung der realen Welt auf den linearen Speicherraum zu finden, der von der örtlichen Nachbarschaft Gebrauch macht.

Es sind nur wenige Datenstrukturen bekannt, welche bei diesem Typ von zweidimensional angeordneten Abfragen schnelle Antworten erlauben. Alle beruhen darauf, dass die Aufteilung der Daten selbstregulierend erfolgt und dass sie räumlich gebündelt auf eine Seite des Plattenspeichers gespeichert werden, derart, dass der Zugriff auf eine Seite viele Entitäten einbringt, die für den verlangten Plan wesentlich sind (Nievergelt et al., 1984) (Tamminen, 1982). Die Methode, die in (Nievergelt, 1984) beschrieben wird, ist eine Übernahme einer allgemeinen ‹hash›-Struktur für die Speicherung mehrdimensionaler Objekte. Es stellt sich heraus, dass sie sehr ähnlich ist zur Methode, die wir entwickelt (Frank, 1981) und in den letzten Jahren verfeinert haben (Frank, 1983).

Dies beschleunigt die Abfrage von Plandaten – alle Objekte eines Planes sind nahe beieinander – und ist für manche andere Formen geometrischer

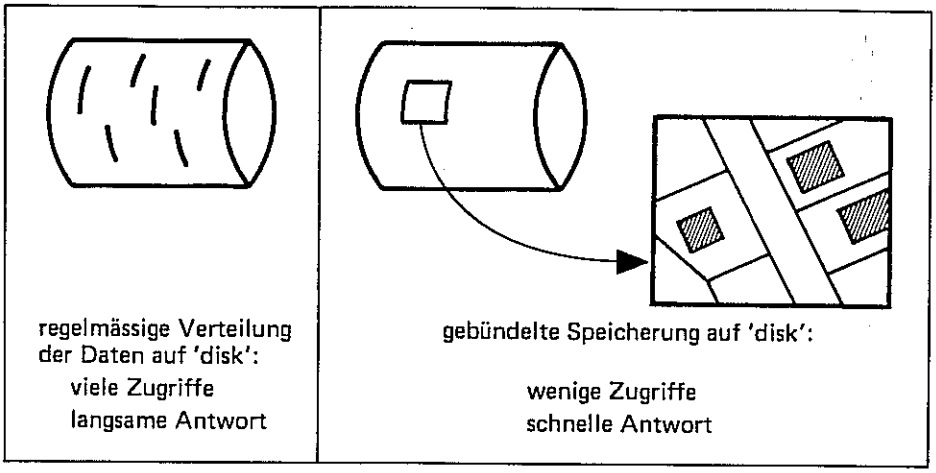

Abb. 1

Datenverarbeitung vorteilhaft. So können wir die speziellen örtlich-geometrischen Zugriffscharakteristiken, wie sie in den meisten geometrischen Algorithmen vorkommen, ausnutzen (Dutton, 1978).

In der FIELD TREE-Methode (Frank, 1983b) wird bei der Bündelung nicht nur der Ort und die Ausdehnung der Objekte berücksichtigt, sie enthält auch eine Komponente, die das Bedeutungsniveau eines Objektes einschliesst. Dies beschleunigt Antworten mit Übersichtscharakter: grosses Gebiet – nur die wichtigen Objekte. Die Antwortzeit für Abfragen ist linear abhängig von der Zahl der gesuchten Objekte. Weitere Einflüsse sind die Grösse des abgefragten Gebietes und die Zahl der in diesem Gebiet gespeicherten Objekte. Die Gesamtzahl der im System gespeicherten Objekte hat keinen messbaren Einfluss auf die Antwortzeit.

Die traditionelle Lösung, wie man sie in manchen marktgängigen Systemen findet, besteht darin, das Anwendungsgebiet in Planblätter (manchmal Fazetten genannt) aufzuteilen und die Daten dieser Planblätter als individuelle Dateien zu speichern. Die Datenmenge in einer solchen Datei ist dann klein genug, um den Einsatz linearer Suchmethoden zu erlauben. Das ist unserer Meinung nach aus folgenden Gründen ungenügend:

- das Auflösungsvermögen des Zugriffs ist festgelegt und gibt nur schnelle Zugriffe, wenn das Abfragegebiet mit der Plangrösse vergleichbar ist; für Abfragen über weit grössere Gebiete ist die Antwort langsam, weil mehrere Dateien geöffnet und gelesen werden müssen;
- der Zugriff auf mehr als ein Planblatt erfordert, dass Objekte an den Planrändern, die durch die Planteilung zerschnitten werden, wieder zusammenzusetzen sind. Das ist eine umständliche und zeitaufwendige Operation, und sie ist nur möglich, wenn das Programm die innere Struktur der Darstellung der Objekte versteht. Deshalb sind die Typen der behandelten geometrischen Objekte in der Speicherschicht definiert, und es ist für einen Benützer schwierig, neue geometrische Objektdefinitionen anzufügen;
- die meisten Systeme verbergen dem Benützer weder die Notwendigkeit, eine Planeinteilung vornehmen zu müssen, noch stellen sie ihm eine Abfragesprache zur Verfügung, welche automatisch die Blattgrenzen überquert. Es scheint zwar akzeptabel, von einem Zeichner zu verlangen, dass er weiss, welches Planblatt er nachführt; es ist hingegen völlig unangemessen, von einem gelegentlichen Benützer, der eine thematische Karte benötigt, die entsprechende Kenntnis zu erwarten.

8. Datenmodelle

Das Datenmodell stellt die Methoden zur Verfügung, die erlauben, die Daten eines Anwendungsgebietes zu beschreiben. Es muss ‹Werkzeuge› anbieten, um die Entitäten und die Beziehungen zwischen ihnen in einer strukturierten (halb-)formalen Art zu beschreiben.

Oft werden die Benützer gezwungen, zwei verschiedene Datenmodelle zu verwenden:

- ein Modell auf hohem Niveau, das Werkzeuge enthält, um die Bedeutung der Daten im Datenbankentwurf zu integrieren,
- ein Modell auf tieferem Niveau, das für die Datenbankverwaltung benötigt wird.

In diesem Abschnitt werden wir kurz die Anforderungen an das Datenmodell besprechen, wie sie sich für den Datenbankentwurf ergeben. Es ist dann wichtig zu überprüfen, ob das gewählte Datenbankverwaltungssystem die mit dem Datenmodell beschriebene Situation angemessen wiedergeben ka und ob Regeln aufgestellt werd können, mit denen man Konstruktion vom Modell ins Datenverwaltungs stem übertragen kann.

Ein Datenmodell muss Strukturen bieten, um Situationen, wie wir sie der realen Welt finden, so genau möglich zu beschreiben. Mindestens der Phase des Datenbankentwurfs s ten wir nicht Gesichtspunkte, die Verarbeitung beschleunigen, in Betra ziehen, sondern wir müssen nach ei möglichst korrekten Abbildung realen Welt streben.

Die Aufgabe hat Ähnlichkeit mit d Problemen, welche im Bereich ‹künstlichen Intelligenz› behandelt u dort mit ‹Organisation des Wisse bezeichnet werden. Drei fundament konzeptionelle Werkzeuge zur Abstra tion sind dabei herausgearbeitet w den, nämlich:

- *Klassifizierung:* bilden von Klass von Entitäten mit den gleichen Eige schaften
- *Aggregation:* verbinden mehre unterschiedlicher Klassen zu ei neuen Klasse, und
- *Generalisierung:* aus mehreren unt schiedlichen Klassen eine allgem nere Klasse entwickeln (Smith, 197 (Mylopoulos, 1980).

Einzelheiten dazu sind in (Frank, 1983 S. 69) erläutert.

Die meisten Datenmodelle enthalt Anweisungen für Klassifikation u Aggregation, Generalisierung aber oft nicht vorhanden.

Wir verwenden zur Zeit ein System, d sich auf das Entitäten-Relationen-M dell (Chen, 1976) bezieht, in de graphisch-visuelle Mittel verwend werden, um die Datenstruktur g sichtbar zu machen. Unsere Metho baut auf Entitäten, Attributen und Bez hungen (sets) zwischen den Entität auf und ist vergleichbar mit manch anderen Datenmodellen, die in d Literatur vorgeschlagen wurden.

Wir haben das Entitäten-Relationen Modell um die Generalisierung erw tert, damit wir Beziehungen wie etw zwischen ‹gerade Linie› und ‹Kreisb gen› mit dem Konzept ‹Linie› angeme sen ausdrücken können. Dies hat si als sehr wichtiges und häufig benöti tes Konzept erwiesen und sollte in jed Verfahren beim Softwareentwurf eing baut werden (Borgida, 1984).

In Abb. 2 zeigen wir ein einfach Beispiel eines Schemas, das uns Entwurfsverfahren verwendet (Fran 1983a).

In der PANDA-Datenbank verwend wir das gleiche Datenmodell, und n kleine Änderungen sind notwendig, u das Entwurfsschema in das Impleme tierungsschema überzuführen.

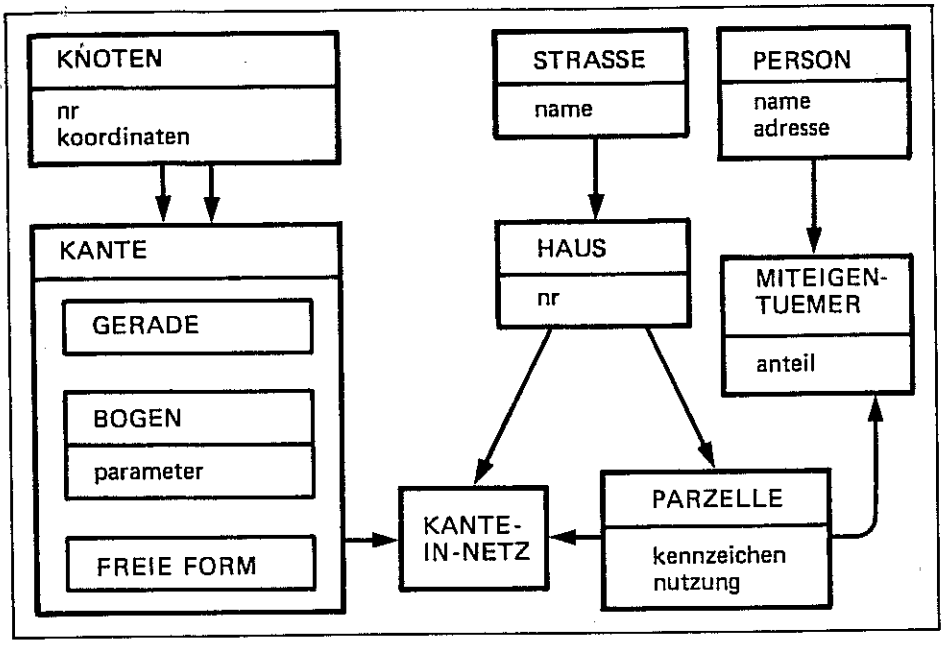

Abb. 2

Es ist möglich, Generalisierungen in Datenmodelle zu übertragen, welche dieses Konzept nicht besitzen, aber das führt zu komplizierteren Schemata und Programmen, welche die Intensionen nicht auf natürliche Weise ausdrücken. Es scheint wünschbar, hoch entwickelte Programmiersprachen zu verwenden, welche die Generalisierung unterstützen, z. B. PASCAL, das Datensätze mit variablen Teilen verwenden kann, oder Sprachen, die auf Generalisierung und Vererbung von Eigenschaften (inheritance) aufbauen (SIMULA, Smalltalk [Goldberg, 1983]).

9. Abstrakte Datentypen

Einige Datenmodelle sind nur mit einem Aggregationsniveau (record) und mit den Grunddatentypen ausgestattet, wie man sie in den meisten Programmiersprachen findet (z. B. integer, real, string of characters). Das ist für nicht-kommerzielle Anwendungen, wie etwa für raumbezogene Informationssysteme, ungenügend. Charakteristische Datentypen sind etwa:

- Werte mit zugeordneten Fehlermassen
- Punkte, welche durch mehrere Koordinatenwerttripel beschrieben sind
- Linien, welche durch eine Folge von Punkten beschrieben sind,

und es ist wünschbar, dass wir das Konzept ‹abstrakte Datentypen› verwenden können, um unsere Software zu entwerfen (Parnas, 1972) (Guttag, 1977). Ein abstrakter Datentyp ist ein Paket, welches eine Datenstruktur und Operationen auf dieser Struktur enthält. Ein solches Paket kann abstrakt definiert werden, ohne dass man seine Implementierung diskutiert; man legt ausschliesslich die Ergebnisse fest, die sich ergeben, wenn man die Operationen anwendet. Anwenderprogrammen ist nur erlaubt, die entsprechenden Daten mit diesen definierten Operationen zu manipulieren; sie dürfen nie von speziellen, nicht definierten Eigenschaften Gebrauch machen, weder von solchen, die durch die Implementierung entstehen könnten, noch durch direkte Veränderung von Daten, die im abstrakten Datentyp eingeschlossen sind.

Diese Technik bringt für den Softwareentwurf und die Programmierung beträchtliche Vorteile, da kleine, in sich geschlossene und leicht zu prüfende Programme geschrieben werden können, die je eine einzige Idee umfassen. Sie reduziert die Abhängigkeiten zwischen den verschiedenen Programmteilen auf ein Mindestmass und vereinfacht dadurch den Softwareunterhalt und Softwareerneuerungen.

Ein solcher abstrakter Datentyp kann z. B. für eine als Kette verschlüsselte Linie definiert und implementiert werden (Freeman, 1974) und Operationen enthalten wie:

- Entschlüsselung der Ketten zu Vektoren und umgekehrt
- Darstellungsroutinen für verschiedene Ausgabemedien
- Schnittpunktberechnungen zwischen solchen Linien usw.

Wenn die Datenbank solche abstrakten Datentypen verarbeiten kann, können diese verschlüsselten Linien gespeichert, abgefragt und mit anderen Entitäten in der Datenbank verbunden werden ohne jegliche spezielle Programmierung.

Dieses Beispiel zeigt die Vorteile der Generalisierung, indem solche verketteten Linien als eine spezielle Form von Linien betrachtet werden (andere Formen von Linien sind z. B. der gerade Vektor, der Kreisbogen); es ist einfacher, wenn wir auf einer höheren Programmebene für alle Linienarten die gleichen Linienoperationen verwenden können, ohne Rücksicht auf besondere Art der Speicherung nehmen zu müssen. Ein verallgemeinerter abstrakter Datentyp ‹Linie› kann also die Operationen anbieten:

- zeichne ‹Linie›
- berechne Schnittpunkt von ‹Linie› mit ‹Linie›.

Intern wird der korrekte Algorithmus ohne spezielle Programmierung ausgewählt.

10. Spezielle Behandlung geometrischer Daten und deren Konsistenzbedingungen

Ein Beispiel, wie abstrakte Datentypen helfen, geometrische Daten zu behandeln, wurde im vorausgehenden Abschnitt gegeben. Wenn man abstrakte Datentypen und Generalisierung anwendet, ist es möglich, ein vollständiges Paket von Routinen zusammenzustellen, womit alle Arten von geometrischen Daten behandelt werden können. In (Cox und Rind, 1980) (Burton, 1979) sind Beispiele von geometrischen Datentypen, welche als abstrakte Datentypen definiert sind, beschrieben. Diese

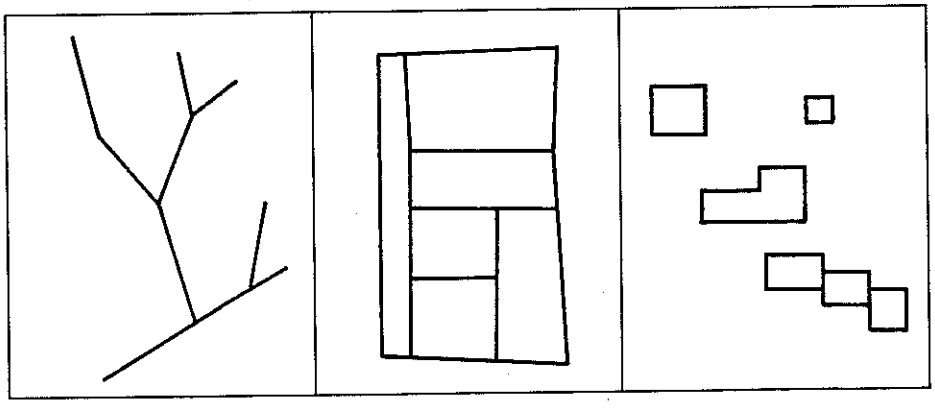
Abb. 3

Vorschläge müssen sorgfältig überprüft werden, um festzustellen, ob sie auf ein bestimmtes Problemgebiet angewendet werden können. Besondere Aufmerksamkeit sollte der Behandlung der unvermeidlichen, mit jeder Messung verbundenen Fehler geschenkt werden.
Wenn wir diese grundlegenden Bausteine verwenden, können komplexere Strukturen gebildet werden, die Polygon-, Netzwerk- und Teilungsstrukturen einschliessen (Abb. 3).
Diese Strukturen enthalten nicht nur *metrische* Informationen über die Lage der Punkte und die Form der Linien, sondern auch *topologische* Information über die Verbindungen zwischen den Punkten und die Zuordnung der Flächen. Einige Anwendungstypen machen ausdrücklich von topologischen Strukturen Gebrauch (z. B. die Bestimmung des kürzesten Weges zwischen zwei Punkten): neue abstrakte Datentypen müssen gebildet werden, um solche Objekte zu behandeln.
In diesem Rahmen sind zwei Aspekte in die Definition der abstrakten Datentypen einzubeziehen, welche in anderen Anwendungsgebieten so nicht in Erscheinung treten.

Graphische Wiedergabe:
Für die meisten Objekte, die in einer raumbezogenen Datenbank behandelt werden, muss die graphische Wiedergabe einbezogen werden. Vorerst hat dies zur Folge, dass das Datenbankverwaltungssystem mit dem graphischen Ausgabesystem verbunden werden muss. Unsere Untersuchungen der vorhandenen graphischen Systempakete und die damit angebotenen Operationen decken zwei Probleme auf:

- Sie bieten einige verhältnismässig primitive Datenstrukturwerkzeuge an (z. B. Segmente [GKS 1984]), welche nicht genügen, um topologische oder andere typische Informationsstrukturen darzustellen; deshalb sind sie umständlich und kostspielig im Gebrauch;
- sie bieten nur wenige mögliche Funktionen für kartographische Anwendungen an, und die meisten Operationen, die notwendig sind, um Pläne herzustellen, fehlen oder haben nicht die erforderliche Qualität (z. B. Linienarten).

Eine gründliche Erforschung der graphischen Operationen, die für kartographische Zwecke nötig sind, scheint unausweichlich. Wir kamen zum Schluss, dass wir besser eine eigene Bibliothek von graphischen Prozeduren aufbauen und ohne zusätzliche Zwischenspeicherung direkt die Daten aus der Datenbank verwenden. Indessen können graphische Ausgabeoperationen nicht in die Operationen eines abstrakten Datentyps für ein gespeichertes Objekt (z. B. ein Haus oder ein Bezirk) eingeschlossen werden, weil die graphische Wiedergabe der Geometrie des Objekts je nach Planart oder Planmassstab Änderungen unterworfen ist. Eine zusätzliche Behandlung der geometrischen Objekte hinsichtlich ihrer graphischen Wiedergabe ist notwendig.

Konsistenzbedingungen:
Für jede grosse Datensammlung, welche auf langfristige Verwendung abzielt, müssen Kontrollen eingebaut werden, die alle Eingabedaten überprüfen. Ohne solche Kontrollen werden sich mit der Zeit unvermeidbare kleine Fehler einschleichen und die Datensammlung wertlos machen. Es ist einleuchtend, dass kein computerisiertes System überprüfen kann, ob die eingegebenen Daten korrekt sind (d. h. ob sie der äusseren Wirklichkeit entsprechen) oder nicht, aber wir können uns mindestens versichern, dass die neuen Daten einigen vorgängig aufgestellten Regeln entsprechen und nicht den bereits gespeicherten Daten widersprechen. Interne Widersprüche in den Daten tragen nicht dazu bei, das Vertrauen der Benützer in die Antworten, die sie vom System erhalten, zu stärken. Alle Inkonsistenzen können Probleme verursachen, wenn etwa Algorithmen davon ausgehen, dass gewisse Bedingungen in den Eingabedaten erfüllt seien; sie können sich unerwartet verhalten, wenn dann die Eingabedaten nicht den erwähnten Regeln entsprechen (z. B. beim Programmausgang, bei Schleifen usw.). Algorithmen müssen natürlich so geschrieben sein, dass sie gegen die am meisten störenden Folgen solcher Inkonsistenzen geschützt sind, aber im allgemeinen ist es nicht möglich, ein konkretes Ergebnis zu erhalten, wenn Inkonsistenzen vorhanden sind (Beispiel: wie gross ist die Fläche in Abb. 4?).

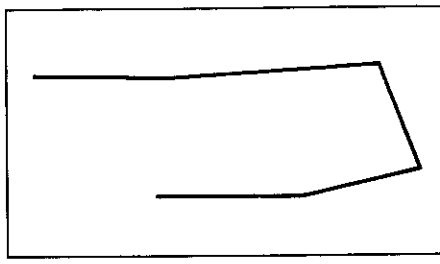

Abb. 4

Die Regeln, welche die von den Daten zu erfüllenden Bedingungen genau umschreiben, heissen Integritäts- oder Konsistenzbedingungen. Sie treten in verschiedenen Formen auf, z. B. indem sie einen Bereich für einen Wert angeben, z. B. ‹das Alter einer Person liegt zwischen 0 und 120 Jahren› oder komplizierter, ‹das Grenzpolygon eines Bezirks ist geschlossen›.
Einerseits sind Integritätsbedingungen für geometrische Beschreibungen meistens verwickelt und nicht leicht zu formulieren. Anderseits kann aber deren Verletzung durch die meisten Auswerteprogramme – besonders jene der topologischen Bedingungen – nicht verkraftet werden. Das U.S. Bureau of Census, welches wahrscheinlich die grösste Sammlung von topologisch strukturierten, raumbezogenen Daten unterhält, hat entschieden, dass ein beträchtlicher Aufwand gerechtfertigt ist, um die Dateien nach Änderungen zu überprüfen, damit spätere Schwierig-

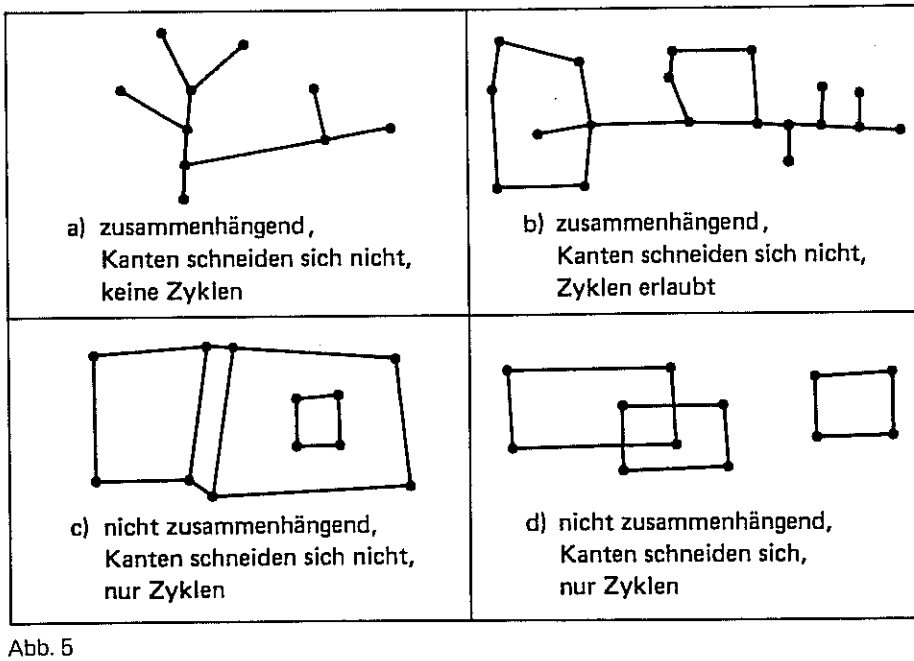

Abb. 5

keiten in der Verarbeitung vermieden werden (Corbett und Farnswork, 1972) (Cranford, 1978) (White, 1984).
Wir verallgemeinerten diesen topologischen Ansatz auf verschiedene geometrische Strukturen (einige Beispiele sind in Abb. 5 gegeben) und zeigen, dass sie durch wenige topologische Bedingungen gekennzeichnet werden können (Frank, 1983).
Abb. 5a bildet einen «Baum»: es gibt keine Zyklen, aber alle Kanten hängen zusammen. Demgegenüber sind in Abb. 5c alle Kanten Teil eines Zyklus, aber der Graph ist nicht zusammenhängend.
Versuchsweise haben wir drei Bedingungen verwendet:

- zusammenhängend/nicht zusammenhängend
- nur Zyklen/einige Zyklen/keine Zyklen
- planarer Graph/nicht planarer Graph (d.h. Kanten dürfen sich nicht schneiden/Kanten schneiden sich).

Diese Bedingungen dürften mindestens eine grosse Klasse von Anwendungen abdecken.
In einer interaktiven Umgebung ist es nötig, diese Bedingungen nach jeder Änderung, die die topologische Struktur beeinflusst, mindestens für die betroffenen Entitäten zu prüfen. Zur Zeit arbeiten wir daran, diese Prüfungen zu beschleunigen.

11. Interaktive Abfragesprache

Die bisherige Diskussion hat Entwurfskriterien beschrieben, welche in erster Linie für den Ersteller von Anwenderprogrammen wichtig sind und den Aufwand und die Kosten bestimmen, die mit dem Entwerfen, Programmieren, Dokumentieren und Unterhalten von Anwendungsprogrammen mit raumbezogenen Daten verbunden sind.
Oft benötigen Benützer Antworten auf verhältnismässig einfache Sachverhalte und haben keine Zeit oder nicht die notwendigen Kenntnisse, um besondere Programme in einer höheren Programmiersprache zu schreiben. In solchen Fällen kann eine interaktive Abfragesprache als Methode dienen, mit der Informationen in einer Sprache abgefragt werden können, die leicht zu lernen ist und die eine unmittelbare Antwort erzeugt. Interaktive Abfragesprachen sind für den gelegentlichen Benützer der Schlüssel zu den Reichtümern, die in einem Informationssystem gespeichert sind!
Interaktive Abfragesprachen sind für die meisten kommerziellen Datenbankverwaltungssysteme erhältlich. Sie werden weitherum verwendet, und es wurden verschiedene Studien gemacht, um ihre Benützerfreundlichkeit zu bestimmen (Reisner et al., 1975).

```
SELECT   StrassenName, HausNr
FROM     Haus
WHERE    HausNr = 20
AND      StrassenName =
         SELECT   StrassenName
         FROM     Strasse
         WHERE    StrassenName = 'Hauptstrasse'
```

Abb. 6

Sie sind meistens auf dem relationalen Datenmodell aufgebaut und umfassen Methoden, um die Daten, an denen der Benützer interessiert ist, in Form des Prädikaten-Kalküls auszuwählen. In Abb. 6 wird zur Veranschaulichung eine typische Formulierung in SQL (Chamberlin et al., 1976) dargestellt.
Weiter enthalten sie einen Mechanismus, um die Form, in der die Antwort gewünscht wird, zu beschreiben. Dieser Teil der Abfrageumschreibung wird durch den Benützer meistens nicht vorgegeben; deshalb werden sog. default-Werte verwendet (d.h. Parameterwerte, die den Normalfall auslösen). Die Ausgabe besteht in allen Systemen aus einer Liste; einige fortgeschrittene Programme bieten auch einfache Formen von Graphik an.
Kommerziell orientierte Systeme enthalten keine Methoden, die Daten suchen, um sie in einem Plan darzustellen. Ausnahmsweise könnten sie dazu dienen, um raumbezogene Daten in alphanumerischer Form abzufragen, aber sie vermögen die Anforderungen des Benützers an ein raumbezogenes Informationssystem nicht zu erfüllen.
Im wesentlichen muss eine interaktive Abfragesprache für raumbezogene Informationssysteme analoge Funktionen wie bei kommerziellen Systemen anbieten, Ein- und Ausgabe müssen jedoch in Form von Planzeichnungen möglich sein. In (Frank, 1982b) haben wir den Entwurf von MAPQUERY gezeigt, einer Sprache, um raumbezogene Daten abzufragen. In der Zwischenzeit ist sie implementiert worden (Egenhofer, 1984).
Zusätzlich zu den Problemen in alphanumerischen Abfragesprachen sind die folgenden für die graphische Ausgabe typischen Probleme zu lösen:

- Datenauswahl: es genügt in den meisten Fällen nicht, nur gerade das Datenelement, nach dem der Benützer fragt, zu suchen, sondern es müssen genügend weitere Daten herausgesucht und wiedergegeben werden, um ein Umfeld zu bilden, in dem die vom Benützer gewünschten Daten interpretiert werden können. Dieses Umfeld soll entweder vom Benützer ausdrücklich gewählt werden können, oder es kann ein bezeichnetes Standard-Umfeld sein.
- Auswahl eines Planausschnittes: es ist notwendig, dass man ein Gebiet vorgeben kann, das auf dem Bildschirm darzustellen ist. Dieses Gebiet kann entweder durch den Benützer ausdrücklich bezeichnet werden, indem er Koordinaten eingibt, vorzugsweise interaktiv mit der Bezeichnung von zwei Ecken in einem vorgängig gezeichneten Plan, oder es wird vom System bestimmt, damit es das vom Benützer ausgewählte Objekt umfasst (z.B. ein Bezirk, ein Haus, eine Strasse).
- Bezeichnung der Plansymbole, welche zur Darstellung der Daten benützt werden sollen: gewöhnlich wird sich der Benützer mit einer Standardliste von konventionellen Signaturen zufriedengeben, aber in gewissen Fällen wird eine graphische Differenzierung aufgrund der gewählten Attribute und Werte nötig sein (z.B. zeige alle Häuser einer Stadt mit verschiedenartigen, vom Alter abhängigen Schraffierungen).
- Interaktive Eingabe von Auswahlkriterien durch Zeigen auf einem Plan, der von einer vorangegangenen Abfrage stammt.

Der implementierte Prototyp bestätigte, dass diese Idee realisierbar ist. Die Durchsicht des Programm-Kodes – ungefähr 5000 Zeilen Pascal – zeigt einige Punkte, die vor der nächsten Implementierung einer genaueren Überprüfung bedürfen:

- die Information über die Datenstruktur in der Datenbank muss über die Sprache des Abfrageprogrammes zugänglich sein (es ist vorteilhaft, sie allgemein zugänglich zu machen);
- in der Datenbank muss mehr beschreibende Information über die Datenstruktur, einschliesslich geometrische Eigenschaften usw., vorhanden sein;
- der Benützer benötigt zur Formulierung einer Abfrage ein vereinfachtes Teilschema (Benützer-Übersicht) des

konzeptionellen Schemas, wie es für den Datenbankentwurf entwickelt wurde; dazu sind verbesserte Formen der Darstellung nötig. Solche Darstellungen müssen Vereinfachungen der intern gebrauchten geometrischen Strukturen einbeziehen, derart, dass sie in der Sicht des Benützers als einfaches Attribut ‹Geometrie› erscheinen.

12. Folgerungen

Datenbankverwaltungssysteme bieten gegenüber traditionellen Datei-Strukturen manche vorteilhaften Eigenschaften an:

- verbesserte Modellierungsfähigkeiten; damit können die Datenstrukturen besser den Anwendungen angepasst werden;
- die Daten können von verschiedenen Benützern gleichzeitig und für sehr verschiedene Anwendungen verwendet werden;
- der Unterhalt der Programme wird erleichtert; sie können veränderten Anforderungen schneller angepasst werden.

Um komplexere raumbezogene Mehrzweck-Datenbanken zu verwirklichen, ist das Datenbankkonzept auf raumbezogene Datensammlungen anzuwenden. Manche der vorhandenen Datenbankverwaltungssysteme sind demgegenüber für kommerzielle Anwendungen entworfen worden und eignen sich nicht gut für die Verwaltung von raumbezogenen Daten.

Diese Abhandlung zeichnet einen Diskussionsrahmen für die Implementierung eines Datenbankverwaltungssystems und zählt wichtige Merkmale für raumbezogene Datenbanken auf:

- um schnelle Antwortzeiten zu erreichen, muss der physischen Bündelung der Daten im Speichermedium Aufmerksamkeit geschenkt werden;
- die Puffervorrichtung muss erlauben, alle für eine Planzeichnung notwendigen Daten zwischenzuspeichern, um wiederholte Zugriffe auf den Massenspeicher zu vermeiden;
- Konsistenzbedingungen sind Regeln, welche Eigenschaften der Daten beschreiben, die immer eingehalten und die automatisch erzwungen werden müssen. Konsistenzbedingungen müssen Beschreibungen der geometrischen Eigenschaften der Daten einbeziehen. So werden Fehler, die sich in die Datensammlung einschleichen können, reduziert, was die Datensammlung langfristig nutzbar macht.

Es wurde ein Datenbankverwaltungssystem, welches sich besonders für raumbezogene Datenverarbeitung eignet, aufgebaut, um zu zeigen, dass die vorgeschlagenen Lösungen realisierbar sind. Eine Datenbank für Kanalisationssysteme mit interaktiver Graphik auf einem Personal Computer (PERQ-Arbeitsplatz) aufzubauen, war recht einfach. Zur Zeit arbeiten wir daran, die Modellierung von geometrischen Datenstrukturen und die Operationen für graphische Wiedergabe unabhängig vom Anwendungsgebiet zu verbessern. Die meisten Datenbankverwaltungssysteme bieten eine interaktive Abfragesprache an, um ohne formale Programmierung Daten aus der Datenbank abzufragen. Es hat sich gezeigt, dass eine entsprechende Einrichtung für raumbezogene Datenbanken realisierbar ist. Solche Abfragesprachen sind unschätzbare Hilfsmittel für gelegentliche Benützer, um den Reichtum der in der Datenbank gesammelten Information auszubeuten.

Literatur

Burton, W., 1978: Efficient Retrieval of Geographical Information on the Basis of Location. In: Dutton, 1978.

Burton, W., 1979: Logical and Physical Data Types in Geographical Information Systems. Geo-Processing. Vol. 1, p. 167.

Borgida, A., Mylopoulos, T., Wong H., 1984: Generalization as a bases for software specification. In: M. Brodie et al. (Eds.). Perspectives on Conceptual Modeling. Berlin Springer.

Chamberlin, D. D., et al., 1976: Sequel 2: a unified approach to data definitions, manipulations and control. IBM Journal Research and Development Vol. 20, p. 560.

Chen P., 1976: The entity-relationship model: toward a unified view of data. ACM Transactions on Database Systems. Vol. 1, p. 9.

CODASYL, 1962: An Information Algebra, Phase I report. Commun ACM Vol. 5, p. 190–204.

CODASYL, 1971: Data Base Task Group (DBTG) Report, 1971.

Codd, E. F., 1982: Relational Database: A Practical Foundation for Productivity. Commun. ACM. Vol. 25, p. 109.

Corbett, T., Fransworth G., 1972: Theoretical Bases of Dual Independent Map Encoding (DIME). International DIME Colloquium Conference Proceedings. Census Bureau, Washington, D.C.

Cox, N. J., Aldred B. K., Rhind, D. W., 1980: A relational data base system and a proposal for a geographical data type. Geo-Processing Vol. 1, p. 217.

Cranford, G. F., 1978: Editing and updating Geographic Base Files – A Discussion of Practical Processing Alternatives. In: Dutton, 1978.

Denning, D. E., Schlörer, U., 1980: A Fast Procedure for Finding a Tracker in a Statistical Database. ACM Transaction on Database Systems. Vol. 5.

DeWitt, D. J. and Hawthorn, P. B., 1981: A Performance Evaluation of Database Machine Architectures. In: C. Zaniolo and C. Delobel (Eds.). Proceedings VII International Conference on Very Large Database. Cannes p. 199.

Dutton, G.: Navigating ODYSSEY. In: Dutton, 1978.

Dutton, G. (Ed), 1978: First international advanced study symposium on topological data structures for geographic information systems. Harvard Papers on Geographi Information Systems. Harvard University Cambridge, Massachusetts

Egenhofer, M., 1984: MAPQUERY. Abfrage sprache für Landinformationssysteme Inst tut für Geodäsie und Photogrammetrie. ETH Zürich.

Frank, André, 1980: Landinformationssyste me – ein Definitionsversuch: Nachrichte aus dem Karten- und Vermessungswesen Reihe 1, Heft 81, Frankfurt 1980.

Frank, A., 1981: Applications of DBMS t Land Information Systems. In: C Zaniolo an C. Delobel (Eds.) Proceedings VII Internatio nal Conference on Very Large Data Base Cannes p. 448.

Frank, A., 1982a: PANDA: PASCAL Netzwer Datenbankverwaltungssystem. Berich Nr. 62. Institut für Geodäsie und Photogram metrie, ETH, Zürich.

Frank, A., 1982b: MAPQUERY: Data Bas Query Language for Retrieval of Geometr Datea and their Graphical Representatio SIGGRAPH '82 Conference Proceeding Computer Graphics. Vol. 16, p. 199.

Frank, A. and Tamminen, M., 1982: Manag ment of spatially reference data. In: Leick, (Ed.) Proceedings International Symposiu Land Information at the Local Level. Unive sity of Maine at Orono, p. 330.

Frank, A., 1983a: Datenstrukturen für Landi formationssysteme – semantische, topolog sche und räumliche Beziehungen in Dat der Geo-Wissenschaften. Dissertation. ET Zürich.

Frank, A., 1983b: Storage Methods f Space Related Data: The FIELD TREE. M. Barr (Ed.) Spatial Algorithms for Proce sing Land Data with a Microcomput Boston, Lincoln Institute of Land Policy.

Freeman, H.: Computer processing of lin drawing images. Computing Surveys Vol. 1974.

GKS. Graphical Kernel System Americ National Standard Draft Proposal X 124–198x. Computer Graphics Vol. 16.

Goldberg, A. and Robson, D., 1983: Smallt – 80: The Language and its Implementatio Addison Wesley.

Guttag, J., 1977: Abstract Data Types and t Development of Data Structures. Commu ACM, Vol. 20, p. 396.

Härder, Th. and Reuter, A.: Database S stems for Non-Standard Applications. R port 54/82, Fachbereich Information. Univ sität Kaiserslautern.

ISO/TC97/SC16. ISO Reference Model Open System Interconnections. Version 4 of June 1979.

Kung, H. T. and Robinson, J. T., 1981: Optimistic Methods for Concurrency Co trol. ACM Transactions on Database Syster Vol. 6, p. 213.

Lampson, B. W. et al., 1981: Distribut Systems – Architecture and Implementati Lecture Notes in Computer Science 1 Berlin, Springer.

Meier, A., Zehnder, C. A.: Flächenmod Register; die Strukturen wichtiger geograp scher Datensammlungen der Schweiz; E Zürich, Inst. für Informatik. Bericht 39, 198

Mylopoulos, T., 1980: An Overview of Kno ledge Representation. Proceedings of Workshop on Data Abstraction, Databas and Conceptutal Modeling Pingree Pa Colorado. SIGMOD Record Vol. 11, p. 5.

Nievergelt, J. et al., 1984: The Grid File + Adaptable Symmetric Multikey File Struc re. ACM Transaction on Database Syste Vol. 9.

Parnas, D.L., 1972: On the Criteria to be Used in Decomposing Systems into Modules. Commun. ACM. Vol. 15, p. 1053.

Reisner, P., Boyce, R.F., Chamberlin, D.D., 1975: Human Factors Evaluation of Two Data Base Query Languages – Square and Sequel. AFIPS Conference Proceedings, 1975 National Computer Conference, Vol. 44, AFIPS Press, 1975.

Salton, G. and Wong, A., 1978: Generation and Search of Clustered Files. ACM Transaction on Database Systems. Vol. 3.

Scheck, H.J. and Lum, V., 1983: Complex Data Objects: Text, Voice, Images: Can DBMS Manage them? In: M. Schkolnick, C. Thanos (Eds.). Proceedings 9th International Conference on Very Large Databases. Florence.

Smith, J.M. and Smith, D.C.P., 1977: Database abstraction: Aggregation and Generalization. ACM Transactions on Database Systems. Vol. 2, p. 105.

Tamminen, M., 1982: Efficient Spatial Access to a Database. Proceedings ACM SIGMOD Conference.

White, M.S., 1984: Technical Requirements and Standards for a Multipurpose Data System. The American Cartographer. Vol. 11, p. 15.

Adresse des Verfassers:
Dr. André U. Frank
University of Maine at Orono
Department of Civil Engineering
103 Boardman Hall
Orono ME 04469, USA

Unsere Zukunft – Chancen und Risiken

Betrachtung des Kulturingenieurmarktes

Arbeitsgruppe Kultur-Ingenieure Zürich (AKIZ)

Es wird versucht, die im Kulturingenieurmarkt wirkenden Kräfte zu skizzieren und daraus Chancen und Risiken für uns als Dienstleistungsanbieter abzuleiten. Dieser Artikel ist als Denk- und Diskussionsanstoss zu verstehen und will einen kleinen Beitrag in Richtung Zukunftsbewältigung bzw. Zukunftssicherung liefern.

Les auteurs de cet article sont de l'avis que les services offerts et la poussée innovatrice correspondante d'un secteur de production peuvent être conservés et même augmentés le mieux par un marché le plus libre possible.
Ils essayent d'évaluer nos forces et possibilités dans le marché du génie rural et d'en déduire les chances et les risques de notre profession. Cet article veut initier les discussions et les réflexions afin de mieux maîtriser et sécuriser notre avenir.

Mit Nachdruck wies der zurückgetretene SVVK-Zentralpräsident J. Hippenmeyer an der Hauptversammlung 1984 auf Risiken im Kulturingenieurmarkt hin [2]:
‹Wir sind uns gewohnt, dass die Auftragserteilung dank unserer Sonderstellung oft ohne grosse Anstrengung unsererseits erfolgt. Diese Situation ist einerseits sehr angenehm, andererseits aber sehr gefährlich. Sie kann dazu führen, dass dort, wo keine Sonderstellung besteht, weniger qualifizierte Leute in unser Berufsgebiet vorstossen, weil sie eben gelernt haben, sich um Aufträge zu bemühen und ihr Angebot zu verkaufen.›

Was ist mit dem Kulturingenieurmarkt los? Wie funktioniert er? Wo krankt er? Was können wir tun?

Der Markt heute ...

Betrachten wir die vorhandenen Kräfte im Kulturingenieurmarkt, die Dienstleistungen, die Kunden, die Marktform.

Der Markt heute lässt sich in folgende *Dienstleistungsgruppen* gliedern:
- Vermessungswesen
- weitere Ingenieur-Tätigkeiten (Kulturtechnik, Ver-/Entsorgung, Tiefbau, Statik, Bodenbewertung)
- Raumplanung (inkl. Baupolizei, Quartierplanung)
- anderes.

Eine *Marktsegmentierung nach Kundengruppen* ergibt folgendes Bild:
- Private
- öffentlich-rechtliche Körperschaften
- öffentliche Hand.

Die *Ausprägungen der Marktform* lassen sich wie folgt umschreiben:

Viele kleine Anbieter – wenige grosse Nachfrager
Den vielen kleinen Kulturingenieurbüros als Anbieter stehen als Hauptauftraggeber Gemeinden und Kantone bzw. von diesen subventionierte Körperschaften gegenüber.

Anbieter bestimmen Preis
Die Anbieter bestimmen die Preise massgeblich über Kalkulation und Tarifabsprache (kartellähnliche Organisation).

Beschränkte Markttransparenz
Da die Konkurrenz kaum öffentlich anbietet, ist eine Markttransparenz nur beschränkt vorhanden. Somit kennen die Nachfrager von Kulturingenieurleistungen im allgemeinen nur einen beschränkten Kreis von Anbietern und sind mit Qualitätsstandards und Preisen wenig vertraut: Ein Wettbewerb für Ingenieurleistungen ist weitgehend unmöglich.

Gebietshoheiten
In einzelnen Hauptdienstleistungsgruppen unterbinden formelle Gebietshoheiten (z.B. Grundbuchvermessung), aber auch informelle (z.B. Kulturtechnik, Leitungsbau, -Kataster) einen eigentlichen Wettbewerb zwischen Anbietern.

Der Markt morgen ...

Unter der Annahme, dass sich in Zukunft (für die Zeitspanne bis ca. 1995) bezüglich *Marktform* keine einschneidenden Änderungen einstellen, kann für den Kulturingenieur als Dienstleistungsanbieter im künftigen Markt folgendes geschlossen werden:

Verknappung des Auftragsbestandes
Durch den Abschluss verschiedenster integraler Werke im Mittelland und die nur zögernde Inangriffnahme notwendiger Meliorationen im Berggebiet stellt sich eine Verknappung im Auftragsbestand in den angestammten Dienstleistungsgruppen ein, nicht zuletzt, weil die Stimmbürger als Entscheidungsträger über die Freigabe finanzieller Mittel unserer wichtigsten Auftraggeber gegenüber bautechnischen Vorlagen grösste Zurückhaltung ausüben (Betontechnik-Feindlichkeit).

Personalabbau
Beschränkter Auftragsbestand führt zu verschärfter Konkurrenz und zwingt zur Kostenreduktion. In Form von gezielten Investitionen (z.B. Zeichencomputer/CAD) wird dies früher oder später zu Personalabbau führen.