Frank, A. U. "Requirements for Database Mangement Systems for Large Spatial Databases." *Geol. Jb.* (1988): 75-96.

Geol. Jb.	A 104	75-96	6 Fig.	Hannover 1988

Requirements for Database Management Systems for Large Spatial Databases

ANDREW U. FRANK

Database, management system, data processing

Abstract: First the notion of 'spatial information systems' is introduced and the general trend in information processing from batch to interactive processing is mentioned. Then the database management system concept is explained and how it applies to the automatic processing and retrieval of geoscientific data. A list of functions a database management system should provide is included and some quantitative estimates given on the size of a typical graphical retrieval.

In order to establish a base for discussion, the present state and trends in computer hard- and software, as far as relevant for the issues, are reviewed.

In order to fulfill varying needs, a DBMS should be organized in layers with specific functions. This way parts may be tailored to suit specific requirements. The main part of the paper shows a possible layered architecture for a Geo DBMS:
- physical storage level with clustering and buffer management,
- layer to protect data,
- access methods using key values,
- spatial access (supported by physical clusters),
- data modelling concepts and corresponding operations,
- support for abstract data types and object-oriented programming,
- methods to deal with consistency constraints,
- query languages.

This architecture has been used for the construction of the PANDA database system, which is generally available for research applications. The paper concludes with a list of requirements for a DBMS usable for geoscientific applications.

[Anforderungen an Datenbank-Verwaltungs-Systeme für große räumliche Datenmengen]

Kurzfassung: Zuerst wird der Begriff des „räumlichen Informationssystems" eingeführt und der allgemeine Trend in der Datenverarbeitung, Stapelverarbeitung zu meiden und vermehrt interaktive Systeme anzuwenden, erwähnt. Das Datenbank-Konzept wird erklärt und seine Wichtigkeit für die Verarbeitung der Daten in Geowissenschaften wird erläutert. Eine Liste von Funktionen, die ein allgemeines Datenbanksystem erfüllen sollte, zusammen mit einer quantitativen Abschätzung der typischen graphischen Abfrage, wird angegeben.

Author's address: Prof. Dr. A.U. FRANK, University of Maine, Department of Civil Engineering, 103 Boardman Hall, Orono, ME 04469-0110, U.S.A.

Als Grundlage für die weitere Diskussion werden die relevanten Eigenschaften der gegenwärtig erhältlichen Computer Hard- und Software, sowie deren unmittelbar zu erwartende Entwicklung gestreift.

Um verschiedenen Bedürfnissen zu genügen, sollte Software für Datenbank-Verwaltungssysteme in Schichten organisiert sein, die jeweils bestimmte Aufgaben erfüllen. Module in einzelnen Schichten können ausgewechselt werden, sofern die Schnittstellen klar genug definiert sind. Wir stellen solche Schichtenorganisation für ein Datenbank-Verwaltungssystem vor:
- Physische Speicherung mit Clusterung und Pufferverwaltung,
- Datensicherung und Datenschutz,
- Zugriffsmethoden mit Hilfe von Schlüsseln,
- Zugriff auf Daten nach räumlicher Selektion,
- Methoden zur Datenmodellierung,
- Unterstützung für objekt-orientierte Programmierung,
- Hilfe bei der Formulierung von Konsistenzbedingungen,
- Abfragesprachen.

Diese Architektur wurde für das PANDA Datenverwaltungssystem verwendet, welches für Forschungsarbeiten erhältlich ist.

Die Arbeit schließt mit einer Liste von Anforderungen, die ein Datenbankverwaltungssystem für geowissenschaftliche Anwendungen generell erfüllen sollte.

Contents

1. Introduction

It is now becoming feasible to use computer systems to construct powerful information systems for the geosciences. This is an important step in the quest for better understanding of our environment, valuable for better management of the natural resources and contributes to the overall quality of life (PETERSOHN & MCLAUGHLIN 1986). Information systems for the geosciences must be comprehensive and broad and must include more than the data for one or a few single aspects of our environment. The combination of data on different aspects and their integration does yield more insight than the analysis of single components. Unfortunately, bringing data from different sources together and managing them to make them available for research and decision making is a major task: we have to solve hardware, software and organizational problems.

Hardware problems are easiest to solve – the components for storage and processing of even large amounts of data are available from various manufacturers. Prices are quite reasonable and the general trend is toward 'zero cost hardware' (DANGERMOND & MOREHOUSE 1987).

In recent years we have learned that generally software is much more difficult to build than previously thought (the famous 'software crisis') and that progress is usually much slower than one expects. The software system to manage geoscientific data must contain a database management system. This paper deals primarily with this software component and the requirements placed on it by applications from the geosciences. We use modern software engineering concepts to organize the discussion and are especially concerned with the integration of database management systems with other software which is specifically written for geoscientific data processing.

Before the more technical discussion starts, it must be noted that organizing the cooperation of different groups to collect data individually and share the results is a very difficult issue, for which few guidelines and rules are available. On the other hand, we can see that many projects fail not for technical reasons, but for lack of organizational arrangements, or due to a poor understanding of social or economic implications. I assume that progress on these topics will be substantially slower than on building the technical systems.

In this paper, we will first introduce some general trends in data processing which foster the use of database management systems. We will then introduce the database concept and how it applies to processing of geoscientific data.

Section 5 gives some assumptions about the present state and the probable future development of hard- and software systems, as relevant to the present topic. We also give a quantitative description of the typical graphical retrieval, which could be used as a benchmark.

The next section gives an overview of the layered architecture we propose for a database management system and sections 7 through 12 discuss the functionality and implementation aspects of each layer:
- physical storage, buffer management and clustering,
- protection of data and the transaction concept,
- access methods based on unique keys or spatial location of objects,
- data models and how they integrate with object-oriented programming,
- query languages.

The paper concludes with some recommendation for building database management systems for spatial information systems.

2. Spatial Information Systems

The use of computers for 'batch' processing, where all the input data are collected and an output with the result is delivered later, has been largely replaced by interactive information systems, where the system maintains a collection of data which are then interrogated by the users as they need the information.

In general terms, an information system contains an image or model of reality, which we can use to make decisions and need not investigate the facts each time anew. This is extremely important in all situations, where data collection is expensive, cumbersome or slow, which is generally true for geoscientific data.

Of special interest to us are systems dealing with data related to locations in real world space – here referred to a spatial data. Many operations of government at all levels, as well as planning and research, exploit data which has a spatial component. Such systems are referred to by various headings, e.g. geographic information system, land information system, multi-purpose cadastre, etc. This paper concentrates on general aspects of systems dealing with spatial data referred to as "spatial information systems", without consideration of differences between systems for differing purposes. An attempt to define a taxonomy is presented elsewhere (FRANK 1980).

This paper concentrates on systems which store data with an exact reference to location and describes geometry using points and vectors. We believe there are applications for such vector systems, especially if one considers the storage necessary for raster-based systems. This is not to exclude systems of other types – there are obvious advantages in the use of raster operations for certain tasks – but they seem to be substantially different and warrant a separate discussion.

3. General Trends in Data Processing of Geoscientific Data

3.1 Move towards Interactive Processing

Users are less and less willing to wait for the response produced by a central computing facility. They want to use collected data in an interactive, conversational mode and they know from the personal computer wave that this is technically feasible.

3.2 Storage of Data in Databases

As we move away from batch processing and towards interactive systems, the traditional file structures, which are optimized for a single (or few) programs accessing the data, become impractical. In an information system, many different programs want to use the data and the database must respond. The database management system is the technical answer making the data collection independent of any specific application and generally available.

3.3 Software Engineering

Research into the problem of writing the software needed for today's complex systems has resulted in some general rules and a shift of emphasis. In the past, it seemed that writing the code to achieve some desired action was the problem. Today we see information systems more as models of reality and the code as the exact description of the model. Thus the representation of knowledge and structuring of data have become central themes, and we use results from artificial intelligence research and methods used in the construction of expert systems.

A current trend in software engineering is that of building software as a collection of models of objects, such that each object has a specific set of operations applicable to it. These techniques are known under different names (with minor differences) such as "abstract data types" (PARNAS 1972, GUTTAG, 1977) or "object-oriented design" (DITTRICH 1986, MEYROWITZ 1986), and have been shown to be especially effective in building engineering or scientific systems.

4. Database Management Systems

Data collected in a database are valuable since much effort is necessary to collect and insert the data into the system and keep the data up to date. These data must be available for a long period of time to justify these expenses. Obviously, new, unforeseen changes will occur in the application during the lifetime of the data.

Under these circumstances the traditional simple file structure designed to facilitate a special application program is no longer adequate. Generic storage structures for multiple access, however, are complicated to program, and it seems inappropriate to repeat this effort over and over again.

A database management system should provide the following functionality:

- Storage and retrieval of data; selection of data using different and varying key fields;
- Standardizing access to data and separating the data storage and retrieval functions from the programs using the data. This makes database and application programs independent of each other, so that changes in the one do not necessarily lead to changes in the other. This independence is important in order to accommodate the changes during the lifetime of a database;
- The interface between database and application programs should be based on a logical description of the data, and not make details of the physical storage structure visible to the applications;
- Making access functions in applications independent of the physical storage structure, so adaptations to expanding storage needs do not influence the application programs;

- Allowing for access to the data by several users at a time;
- Providing for the definition of consistency constraints for the data which will then be automatically enforced. Consistency constraints are rules which must hold for all data stored, and are an excellent technique to reduce the number of errors in a large data collection.

For a slightly different, more elaborate list see CODD (1982).

Access to data should be possible both from a high level language and from a user-friendly query language. The integration of the database manipulation language with the programming language used is crucial to ease of use and can help to avoid hard-to-track bugs. A free-standing query language is helpful for casual users to retrieve data from the database to answer ad-hoc questions without any formal programming; this makes the database usable for one-of-a-kind questions, which are often posed in scientific research work in planning applications.

A database management system is thus a method of encapsulating the valuable data to make it available to a multitude of users programs while simultaneously protecting the data. A database is not just a collection of data: we should use the term only for a collection of data together with the corresponding management system (Fig. 1).

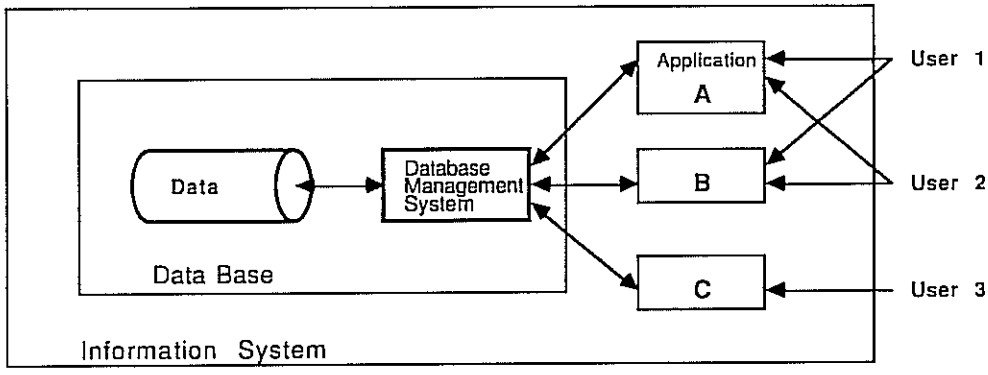


Fig. 1: Database management system concept.

Spatial information systems typically require large data collections, stored in a permanent manner on mass storage devices (disk) and accessing only small parts of these collections for the execution of a simple operation.

Early in the history of data processing, programmers became aware of the similarity in their needs to store data and retrieve them later for processing. In lieu of writing procedures for these functions for each application anew, an attempt was made to write a generally applicable program to provide these services. The idea of a (generalized) database management system was born (CODASYL 1962, 1971).

Several attempts have been made to use standard commercial database management systems for spatial information systems, but they generally did not succeed. The assumptions about typical utilization in commercial systems were quite different from usage patterns found in engineering and geoscientific databases (PLOUFFE et al. 1984).

On the one hand, research in the area of spatial information systems has shown that functionality similar to standard database management systems is required. It seems that techniques used in commercial database management could be beneficial (FRANK & TAMMINEN 1982). On the other hand, researchers in database theory have become interested in learning about requirements for non-standard applications.

Research in non-standard database management systems started in the early '80s (FRANK 1981, HÄRDER & REUTER 1982, SCHEK & LUM 1983) and continues to the present (MANOLA et al. 1987).

This paper is the result of several years' work dedicated to applying the database concept to spatial data handling at the Swiss Federal Institute of Technology Zürich and the University of Maine at Orono. The ideas reported here are based on experience with the PANDA database management system we built (FRANK 1982a, 1984, 1986, EGENHOFER & FRANK 1987). They identify methods successfully implemented, and include a critique of methods which did not work and will be replaced in the future.

5. Assumptions

This paper is - like most papers - based on a series of assumptions. Some of them should be made explicit in order to prevent the confusion of the reader. The discussion is based on today's technology, which may be more advanced than found in some installations. However, we exclude all experimental hardware.

Processor: We assume a VON NEUMANN hardware architecture and do not discuss future parallel multiprocessor hardware. The extent to which this can be beneficial to database operations seems to be in dispute (DEWITT & HAWTHORN 1980) and hardly any practical results are reported.

Storage media: We assume use of main storage (formerly "core") directly accessible by the processor, and separated slower mass storage (disk). Data in mass storage is only accessible after transfer into main storage. Access to data on mass storage is much slower than to data stored in main memory (access times in range of 10,000 : 1), and access is in larger chunks, with access time being essentially constant and independent of the amount of data accessed.

Distribution of processing: Data stored on a central computer may be transferred for use in an "intelligent" workstation having its own processor and program. Our assumption is that all the data are centrally stored and used by the workstations and do not treat the distribution of the storage of data on several cooperative computers. It seems that the basic problems of such an arrangement have not yet found a sufficiently general solution (LAMPSON et al. 1981).

On the side of the application, we assume that the amount of data to be stored is larger than can be fitted into the main memory of the computer used. Thus a subset of the data must be transferred from the permanent storage (disk) to the working memory.

The primary form of data usage is to draw maps on a display unit. We propose as a sort of benchmark the retrieval of the set of data necessary to construct such a display:

Estimates derived from counting objects on produced output screens show that useful screen drawings contain about 2000 to 5000 objects. Screens with substantially less data

seem empty and do not convey enough information about an area, whereas screens with more data are too crowded and difficult to read. From literature about human factors, it is known that response time is very important for user satisfaction. We therefore set as a goal the retrieval and display of "a screen full" of data within less than 20 seconds.

6. Layered Architecture

The framework for this study is a hierarchy of modules, each providing certain types of services or functions to the next upper layer. The lowest layer is directly related to the services provided by the operating system, whereas the top layer provides services to the human user (Fig. 2).

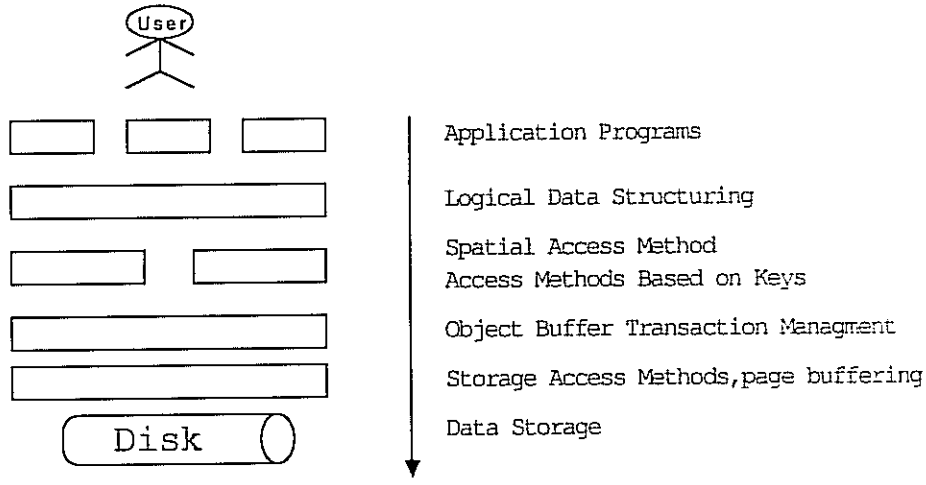


Fig. 2: The layers of a spatial database management system.

The bottom layer stores data, using the operating system to access the file system. These layers are mainly concerned with improving the performance of access to data by clustering techniques and buffer management. Services offered are "store" and "retrieve" operations for data elements (records) using internal keys.

The next layer will provide essentially the same operations, but will make them secure. Changes in the database are guaranteed against loss or interference with other users.

The third layer adds different types of access methods, e.g. access to data based on a key value or spatial location.

The fourth layer offers a logical structuring tool for the data and manipulations based on this logical schema. These services are then offered as an extension to a high level programming language or an independent query language.

7. Requirements for the Data Storage Layer

This bottom layer interfaces with the actual storage operations provided by the operating system. Its main purpose is to interface with the operating system and to improve

speed of storage and retrieval. Perfomance of a database management system is primarily perceived as response time to queries of general time to retrieve data stored in the system. Experience shows that retrieval time to data is determined by the number of physical disk accesses and that processing time of data once transferred to working memory is of minor influence (recall that access to data in working memory is 10,000 times faster than accessing data on the disk).

The requirements for this layer result from the maximum delay we allow for the drawing of a map on a screen. If each of the 2000 data records necessary for "one screen full of data" were fetched from the disk with an independent access (taking 50 to 200 msec.), the user would have to wait 2 to 4 minutes for the map to be drawn. Therefore, this layer must reduce the number of physical accesses necessary to retrieve the records for the map. Two basic techniques are known:
- clustering of data
- buffer management

and both of them will be used.

7.1 Interface to the Operating System

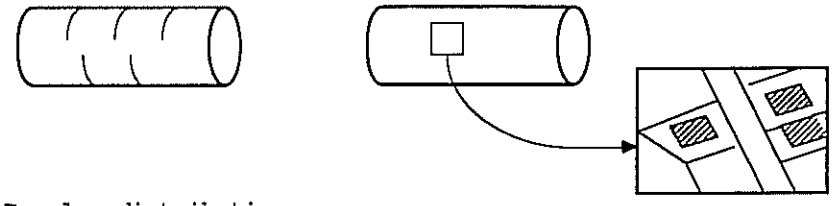
It is desirable that the interface to the operating system be simple and only use standard functions found on many operating systems (read and write to random access files). This will facilitate transportability of the database management system and the application using it from one computer to another. However, it must be noted that some of the methods used in operating systems to improve performance of disk operations can interfere with the methods a database uses, and it may be advantageous to use low level direct disk access operations.

7.2 Physical Clustering

A physical access to the disk brings in a larger chunk of data, usually called a disk page: it is 512 bytes to a few 1000 bytes large. If we can arrange our records on disk pages in such a way that each page contains several data records necessary for the map drawing, the number of time-consuming disk accesses to transfer the data to working memory will be greatly reduced. In 20 sec. we can possibly read about 200 pages; if each contains 25 records which are needed (and possibly some others), all the necessary data are read.

This is feasible in all cases where a reasonable prediction of what data will be used together is possible. These data are then stored together and form a physical cluster (SALTON & WONG 1978). Fortunately for spatial retrieval for map drawings such predictions are easy, based on the neighborhood relation of objects. We retrieve data for a map from objects situated within a certain area. If we retrieve a data element of an object, chances are good that data from other objects in the vicinity are needed next. If such data are clustered together and retrieved with one physical access, we can achieve the goal of retrieving a map within a short time span, permitting interactive work (Fig. 3).

In simple terms: it is necessary to find a mapping from real worlds space to the linear storage space which preserves neighborhood relationships.



Regular distribution
of data on disk
→ many accesses
→ slow response

Clustered storage on disk
→ few accesses
→ fast response

Fig. 3: Spatial clustering.

A database system for spatial data must at least provide for physical clustering of data (typically a command like "STORE NEAR X" where x is an already stored record). This requirement excludes many of the simpler relational database implementations, where the physical storage is directed by the primary key and cannot be influenced by the user. It seems also advantageous if data of different types (e.g. houses, roads and rivers) can be stored on the same page and do not necessarily require different physical accesses.

However, it is neither necessary nor desirable that physical clustering be visible on the level of the user interface; it should rather be used internally for fast spatial access and be of no concern to the user.

7.3 Buffering

Many programs show a locality of access, using the same data elements repeatedly over a short period of time. If these data elements can be kept in a buffer, the number of physical accesses to a slower mass storage device can be reduced: for each data element only the first access uses the slow device, and all following requests are satisfied using the buffered data. This strategy is generally employed in computers (virtual memory, cache etc.). A database usually contains a single level of buffering for pages brought in from the disk. This is indispensable in order to exploit a physical clustering of data on the mass storage device.

Programs dealing with spatial data typically show much locality of access to data but use a large working set (all the records for a map drawing). This is especially important for interactive programs. Users tend to work in a geographical area for several interactions before they request another base map. Very often they ask for an overview map first, then zoom in on a detail of interest and start to work on this.

In our experimental database management system PANDA we included a second level of buffering for single data elements. We currently set the size of the buffer to 5000 records, which allows us to keep a complete map drawing in buffer. Redrawing a map or zooming in does not require any additional physical accesses and is consequently very fast.

We found that this concept has considerable influence on the programming style of an application: no attempts are made to keep database data stored in program variables since a second access to data is always very fast. In our computer graphics programs, for

example, we do not use linear display lists but produce graphics directly from the database records.

8. Protection of Data

Large collections of geographic or administrative data are, as other data collections, valuable assets and must be guarded against loss. Even if the information is maintained in parallel in other, traditional forms (registers, maps, etc.), the cost of transferring this information into machine readable form is considerable.

But it is not only the cost of reestablishing a machine-readable data collection that makes us protect data. Information that may be deduced from the data may be of enormous economic value for someone who knows how to take advantage of it: information is power. Even if this aspect is probably less pronounced for a spatial database than for most databases in commercial companies, the data collected must be guarded against unauthorized use. For example, most national statistical bureaus collect data which must only be disclosed as statistical aggregates which do not permit tracking down values for individuals.

In order to avoid the costs of reestablishing a data collection and to prevent unauthorized use, it is necessary to protect the stored data. Looking at the technical problems, it is customary to differentiate the following four classes of threats:
- Loss due to errors in operating or malfunctioning of hard-, software or erroneous manipulation by authorized users;
- Destruction and access by unauthorized users;
- Introduction of false data by authorized personnel using correct procedures;
- Corruption of data by multiple users at the same time (concurrent updates).

Unlike other resources, data are more complex to guard and organizations typically have much less experience in managing data resources. First, human beings cannot sense the presence (or absence) of data directly: one needs computer equipment and programs to assess a data collection. Second, data collections cannot easily be measured, as the presence of data does not yet guarantee that the data is useful, correct, updated and complete.

The layer described in this paragraph will not add substantial new functionality but will increase security of operations. Most of the functions in this layer will reduce performance – that is the price we have to pay for the security gained ("there ain't no such thing as a free lunch!").

In order to determine the appropriate measures, we must assess possible damages resulting from loss of data (cost of reentering data, cost associated with interruption of operations etc.) and balance them against the cost of operations to prevent such losses. It is generally assumed that most commercial operations place more importance on interrupted operations and confidentiality. In consequence, very involved but secure schemes have been devised. In our area of application, lower levels of security may be acceptable.

8.1 Protection against Loss

Protection of the data against loss by malfunctioning hardware or software is absolutely necessary. In all situations in which more than just infrequent changes are made,

the protection mechanism should include the updates of the database. A change the user affected and the confirmation received from the database must not be lost any time later, independent of any interruption by hard- or software problems.

It is customary to distinguish two types of problems:

Interruption of the database management program due to operating errors, problems in the operating system or failure of the hardware. Such interruptions occur in most installations quite frequently (one per day..one per week). During such events all contents of main memory are lost, and it is therefore necessary to write changed data to permanent mass storage before confirmation of an update.

Loss of the storage media, again due to errors in operations or hardware defects (the so-called "head crashes"). Such problems are usually rare (once a year) and slower recovery procedures are acceptable. Traditional data processing used the "two generation" system, keeping the copies of data on magnetic tape for at least two previous defined states and for all updates in between. A similar method is applicable to database management: the database is copied to magnetic tapes at regular intervals, and all changes between the copies are not only used to change the database but also written to an additional log file (in fact they should be written to the log file first).

The services offered in commercial database management systems are generally sufficient. However, many of the systems for use on smaller micro- and personal computers, as well as the systems specialized in geometric and geographic data, very often do not provide the functionality necessary to manage large data collections over long periods of use.

8.2 Secure Data from Unauthorized Usage

Some databases will contain confidential or secret information, and it is necessary to protect it against unauthorized use. This means first, not to grant access to the database to people who must not have access to any of its contents. But it also includes measures to exclude certain, otherwise authorized users from access to certain classes of data. For example personnel in the town planning office need not know the amount of mortgages on a property. Finally, not only must unauthorized access be prevented but changes to the database must also be controlled. A data collection is easily destroyed by adding wrong data. Clear regulation of responsibility for data quality is required and only the groups responsible can be allowed to finalize changes.

In general we may assume that the operating system contains methods to screen out unauthorized users as well as to identify authorized users. However, the database management system can make only certain parts of the database accessible to certain users. Different methods are known and used, providing varying levels of selectivity in access and security.

A special problem in statistical databases (and many spatial databases will be used as such) is that of restricting authorized users to generalized information only and preventing direct access to individual data. Most of the known systems have proved insufficient, and the proposed methods are extremely involved (DENNING & SCHLORER 1980).

8.3 Transaction Management

Transaction management deals with all safeguards to secure the data against accidental loss by authorized users. We especially have to deal with user input errors, but must also consider all sorts of hardware malfunctioning. Despite power-loss, errors in programs, multiple simultaneous users etc., the overall goal is to prevent loss of data due to all sorts of errors by authorized users, including the introduction of new, false data.

This control of authorized users is primarily concerned with changes inserted into the data collection, assuming that read-only accesses do not change the database and thus cannot introduce errors.

It is customary to group together a number of logically connected changes to the database to form a transaction. The transaction management in a database management system is concerned with
- Atomicity of the transactions,
- Concurrency of transactions by multiple users,
- Integrity of the database after the transaction, and
- Durability of the transactions.

8.3.1 Atomicity of Transactions

From the point of view of the database, all the changes included in a transaction are logically connected – that is the idea expressed in bundling them together as a transaction – and thus either all of them should be executed or none. A database should never contain partial results of a transaction, even if a transaction is stopped by a sudden hardware failure (or loss of power).

Atomicity allows programs to be restarted and continue their work after a sudden stop, as the database is always in a defined state (namely at the end of the last executed transaction) and the program needs never deal with the partial execution of a transaction.

8.3.2 Concurrency

Several users may access and change the same piece of data at the same time. Such concurrent actions may conflict and must be synchronized. In a database management system one generally requires that concurrent users see only the effects of completed transactions and cannot observe any partial results of transactions run by other users. Further, the database management system must check that the actions of one user do not invalidate changes another user has applied.

Commercially available systems for spatial data management typically do not contain automatic prevention of conflicts between users and rely on organizational mechanisms that only one user is working on a file at a time. This places a restriction on the organization of work and limits the availability of the data. As technical solutions are known they should be applied.

8.3.3 Integrity

A collection of data can easily be destroyed by adding 'wrong' data. Users will not use a database from which they often get erroneous or contradictory information. Unfortu-

nately, there are no easy ways to have a program check that entered data (or changes) are correct, meaning that the data describe reality correctly. Computers cannot go out and check! However, we can have the program check that a new (or changed) data item is not in contradiction to other facts already stored in the database. We say that a database is consistent if it is free of such contradictions. Each transaction must be checked to ensure that it does not violate a consistency constraint and introduce inconsistent data into the database.

8.3.4 Durability

A transaction which is once confirmed should never be lost, independent of malfunctioning hardware etc. A database management system should include some mechanism for maintenance of a journal of all changes posted such that an archival copy, which is made regularly, can be updated to the newest version if the current data storage is lost.

8.4 Cost of Data Protection

Methods for transaction management are well-known, but are unfortunately expensive to use. They add considerably to the processing of transactions and can decrease performance roughly by half. This is the reason they are often not included in the commercially available spatial information systems.

In comparison of systems (in benchmarking for example) it must always be specified what type of transaction management is used and what levels of protection of data against inconsistencies and loss are taken.

8.5 Using the Object Buffer for Transaction Management

In the experimental PANDA database management system we have included a large object buffer, which can hold all the data objects a user accesses during a transaction. With this buffer, we can prepare all changes to the database in the buffer and not affect the data files. Once the transaction is completed, all the changed objects are written out to disk at once. This is a relatively simple scheme which can be expanded to include transaction logging, as a journal of all changes can be (must be) written before any updates of database files are done.

8.6 Long Transactions

The use of the transaction management to achieve several competing goals restricts the flexibility of what can be defined as a transactions. If transactions serve as units for durability they must be short, because it is not acceptable to lose much work. Similarly, transactions used to control concurrency must be quick, because barring other users from changing data for longer periods of time cannot be tolerated.

On the other hand, certain operations are much too complex to be accomplished in a short transaction. In some applications final values only become available long after the original measurements were taken. We must be able to record the location of the measurements before the results are available, but consistency demands that ultimately the values are entered. Such situations are generally called 'long transactions' and no detailed concepts for treatment are available or implemented (BORGIDA 1984).

9. Access Methods

The layers discussed so far provide storage and retrieval of data using an internal identifier (database key) which is assigned to a data record when it is first stored. The record itself is considered by these methods as an uninterpreted fixed number of bytes which are stored and later reproduced. For nonstandard applications it is especially important that the storage system does not depend on interpretation of the data stored because this would limit the flexibility of arranging information in a record and produce undesirable dependencies between the different layers of the system. The database subsystem of a larger application must handle arbitrarily structured data records without dependency on their internal structure. The record data structure composed as a hierarchy of the basic data types, i.e. integers, reals, and strings, is not even sufficient for commercial applications, as exemplified by numerous extensions offered (data types for money, date etc.).

The access methods discussed in this section use parts of the data stored in a record. In order to achieve independence of the data description, the defining modules must export access functions to the relevant data elements which are then used for the access methods.

It should be pointed out that we are describing logically defined methods for accessing data. An implementation must include algorithms and storage organizations in order to realize these methods and achieve fast responses.

9.1 Access by Unique Key

Users often need access to data records based on the value of one or several data fields (key fields). This requires in the simplest case that the stored values for these fields be unique, so that given values unequivocally identify the desired record. (This will be a consistency constraint of the database). It is generally necessary to be able to define more than one key for a record, so the record can be found using either value (e.g. number or name).

9.2 Access Using Constructions of a Data Model

The data model permits the grouping of objects to units of a higher order. Such combinations create access paths, as users must be able to go from an object to the groups it is contained in or to the objects it consists of.

9.3 Spatial Access

A great deal of data in a spatial information system is related to objects in space. Location and extension are known for these objects. Many applications need access to data based on location (BURTON 1978). This is obvious for all retrievals associated with map production: all objects within the map frame must be retrieved and then graphically rendered. Similar accesses are necessary for internal operations to check incoming geometric data and for geometric manipulations of stored data. Spatial access to data is based on a query of the form:

Retrieve all (object-type) within (area), where (area) is the description of the area of interest (Fig. 4).

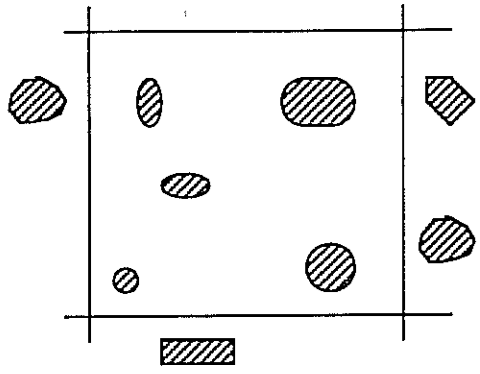


Fig. 4: Query window.

In order to provide a general efficient function, it seems appropriate to reduce (area) to a rectangular box parallel to the coordinate system. Similarly for each object the location and extension is recorded as a rectangular box (minimal bounding rectangle) (Fig. 5). This seems to be a general and computationally simple solution.

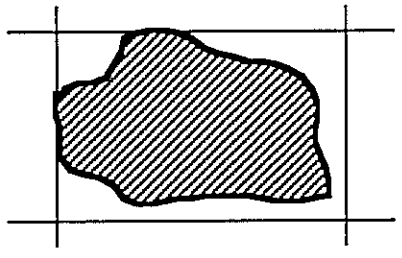


Fig. 5: Object with box (minimal bounding rectangle).

More specific requests are then treated in two steps. In a first step, all data within the rectangular envelope of the area are retrieved (based on a comparison of the query rectangle and the object box). The more expensive, exact selection process is performed in a second step only on the data passing the first test.

The result of such a retrieval may be a large number of objects; a map drawing of the size of a CRT is, in our estimation, composed of 2000 to 5000 objects. We have seen that this is only possible if a special data structure has been built and the data is physically clustered – a sequential scan of the complete, large database takes much too long.

The access method should be divided into (1) a method of physical clustering to achieve a minimal number of physical disk accesses, and (2) a logical data structure, which permits spatial access and guarantees its correctness, preferably even in the presence of some shortcomings in the physical clustering.

The physical clustering will speed up the retrieval of map output – all objects on a map are close to each other – and it benefits many other forms of geometric data processing. We can thus exploit the specific geometric locality of access observed in most algorithms of computational geometry (DUTTON 1978).

The traditional approach found in many of the systems on the market is to divide the world space into map sheets (sometimes called 'facettes'), and to store the data of these map sheets as individual files. The amount of data within such a file is then small enough to permit the use of linear search methods. This is in our opinion insufficient for the following reasons:

- The granularity of access is fixed and provides rapid access only if the query area is comparable to the sheet size. For queries about much larger areas response is slow since many different files must be opened and read in;
- Access to more than one sheet requires recombination of objects at the sheet borders. This is a complicated and time consuming operation and only possible with an understanding of the internal structure of the representation of the objects. Thus the types of geometric objects treated are defined in the storage layer, and it is difficult for a user to add new geometric object definitions;
- Most systems do not hide the sheets from the user, nor do they provide a query language working automatically across sheet boundaries. It seems acceptable to ask draftsmen to know which map sheet they update; it is however clearly inappropriate to require this knowledge of a casual user needing a thematic map.

Only a few logical data structures are known to permit fast response to this type of 2-dimensional range query. They are all based on self-adjusting partition of space and clustering data of one partition in a disk page, so that access to one page brings in many entities useful for producing the requested map (NIEVERGELT et al. 1984, TAMMINEN 1982, GUTTMAN 1984, SAMET 1984). The method described in NIEVERGELT et al. (1984) is an adaption of a more general hash based structure for storage of multidimensional objects. It turns out to be very similar to the FIELD TREE method we have developed (FRANK 1981) and refined during the last few years (FRANK 1983b).

In the FIELD TREE method, clustering is not only guided by location and extension of objects but includes also a component of level of importance of an object. This speeds up responses to "overview" requests (large area, few objects) and for detail maps (small area, all objects). Response time for queries is linearly dependent on the number of objects retrieved. Other influences are the size of the area of the query and the number of objects stored for this area. The total number of objects stored in the system has virtually no influence on response time.

10. Data Models

A data model provides a method to logically describe the data necessary for an application domain. It must contain tools to describe the entities and the relationships between them in a structured, (semi-) formal way.

Often users will be forced to use two different data models:

- A very high level model, which contains tools to integrate many of the semantics of the data for the design of the database, and
- a lower level model, used with the database management system.

In this section we will briefly discuss requirements for the data model used for design. It is then important to check that the data model used with the selected database management system can render the situation expressed in the design data model adequately, and rules for translating constructs from the one to the other must be established.

As was stated before, a data model must provide tools to describe the situation as accurately as possible, following the circumstances we find in the real world. At least during the design of the database we should not consider aspects to facilitate data processing, but strive to picture the real world as closely as possible.

The task is similar to problems treated in Artificial Intelligence, there known as 'kowledge organization'. Three fundamental conceptual tools for abstraction have been identified, namely:

- Classification: forming a group of entities with the same properties;
- Aggregation: combining several different classes to a new one; and
- Generalization: extracting from several different classes a more general one (SMITH 1977, MYLOPOULOS 1980).

Most data models provide tools for classification and aggregation, but generalization is often lacking. These structuring methods can be used to access data, as there are operations fo find all the components of an aggregation or to go from a component to the aggregation it belongs to, etc. We also include access methods using keys within an aggregation, where the key values must only be unique within the aggregation. This reduces the need for artificially constructed, globally unique keys to identify objects.

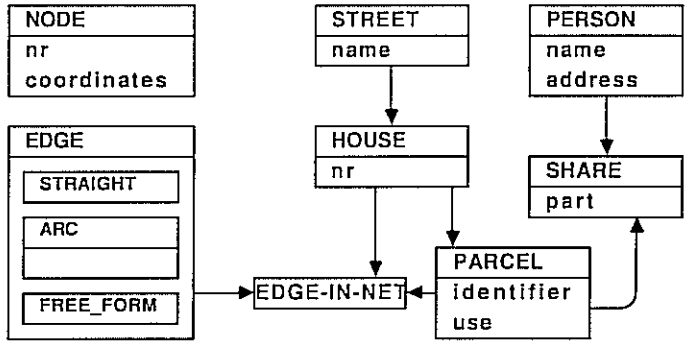


Fig. 6: Simple conceptual database schema.

We currently use a system related to the entity-relationship model (CHEN 1976), stressing graphical, visual aids to make the data structure easy to understand (Fig. 6). Our method is based on entities, attributes, and relationships (sets) between the entities, comparable to many other data models proposed in the literature. However, we are presently reducing the role of attributes to gain more flexibility in the allowed data types and to observe the rules of 'information hiding' (PARNAS 1972) in software engineering.

We extended this model with generalization to be able to adequately express the relationship between, for example, a 'straight line', an 'arc of cycle' and the general concept of 'line'. This has proved to be a very important and often used concept, and should be included in any software design method (BORGIDA 1984).

This schema can be translated either into the database description language used for a CODASYL type database (network) or into a relational database description. The encoding for the PANDA database management system is straightforward, as PANDA offers all three constructs for abstraction.

11. Abstract Data Types or Object-oriented Design

Combining the data base concept with the requirements of object-oriented software design is not easy. A number of standard assumptions in most regular data models contradict the methods of data encapsulation used in an object-oriented design. Further, the database seems to be an implementation of a generic abstract data type, which offers a number of standard operations on database objects (e.g. store or retrieve using an access path) but this is orthogonal to an independent definition of the same objects with specific operations (e.g. for a parcel operation such as change owner or subdivide) (TROYER et al. 1986).

We have found, however, that object-oriented design methods are crucial for modelling real world objects in a spatial information system. This is clearly the case to solve the difficult problems of geometric modelling in a generic way, independent of specific properties of individual geometric objects and object classes (FRANK & KUHN 1986).

In practice, an object-oriented software engineering method is based on the construction of software modules which encapsulate data objects modelling a single type of object in reality together with the pertinent operations. Higher level objects are then implemented as combinations of such lower-level objects, and higher-level operations are constructed from the operations on the lower-level objects. This method helps to reduce the complexity of designing software modules to a level which can be easily managed. Nevertheless powerful systems can be built from the combination of these simple parts.

It is difficult to achieve these goals using commercially available databases, as they typically do not include generalization in their data model and the number of data types available to build data records is also limited.

11.1 Consistency Constraints in an Object-oriented Design

An object-oriented approach provides a powerful solution to the formulation of consistency constraints. Database management systems today attempt to provide tools to express restrictions on the data which exclude contradictory data from being entered in the data collection. The languages to express such restrictions are limited to a few constructs, generally useful in a commercial environment, but by far not sufficient to express the complex consistency constraints necessary to describe, for example, the constraints of an arc-node topological data structure. We have concluded that the full power of a programming language is necessary to deal with the consistency constraints often found in spatial or engineering databases. The object-oriented paradigm of the encapsulation of an object with the appropriate operations shows a different solution. For each legal operation on an object type we write a procedure or function which tests all the consistency constraints (the preconditions for this operation) before any change to the database is executed. As these procedures are written in our regular programming language, every computable condition can be included and tested.

As a top level for the application programmer, we make all these operations available, but do not permit the general use of the generic database operations – thus this layer of object definitions including all legal operations guarantees consistency of the stored data. There are just no operations which can perform changes in the database which would violate the consistency constraints.

This method only covers the regular (short) transactions: it is not usable for the so-called 'long transactions'. It is presently a topic for research to see how to extend these concepts to handle scripts which prescribe and enforces legal succession of operations.

12. Query Language

Query languages are very important for the retrieval of data to satisfy certain immediate needs, which were not planned for and for which no programmed access procedures are available. Interactive languages for ad-hoc queries are usually included in commercial database management systems. It appears today that the SQL language originally developed by IBM (ASTRAHAN et al. 1976) is becoming the accepted standard, and a respective proposal is currently being dealt with in the American National Standards Institute. A number of groups are presently working on extensions of SQL to make it useful for spatial data bases. This raises, however, a number of questions, as a spatial application not only poses special requirements to data retrieval but specifications for the graphical rendering of the data must also be included in the query (FRANK 1982b).

13. Conclusions

We conclude with a number of recommendations:

- Spatial data collections should be built using the database management system concept. This concept is crucial for interactive information systems which can serve multiple users and multiple requirements;
- Spatial data collections pose some special requirements, which renders the commercially available database management systems unsuitable. They generally lack the special provision to achieve physical clustering necessary for fast access to map data, and they are optimized for data with quite different characteristics. Their performance is generally observed to be insufficient;
- Spatial database management systems must include many of the standard features found in commercial systems, especially data protection and transaction management to preseve loss of data due to malfunctioning hardware and to permit concurrent users.

We recommend a layered architecture of the spatial database management system, with a database kernel providing generally required services, and additional modules to adapt the database management system to the special needs of an application area (HÄRDER 1986).

Finally, we feel that an increase in the modelling power of database management systems is most important. They should include some of the best features of knowledge representation systems from Artificial Intelligence research. The methods of Abstract Data Types or of object-oriented software design should be employed to build applications in geosciences, and therefore the database management system must support such techniques.

14. References

ASTRAHAN, M.M. & CHAMBERLIN, H. (1976): Sequel 2: A Unified Approach to Data Definitions, Manipulations and Control. – IBM Journ. Research and Development, 20: 560.

BORGIDA, A., MYLOPOULOS, T. & WONG, H. (1984): Generalization as a Basis for Software Specification. – In: BRODIE, M. (Ed.): On Conceptual Modelling – Prespective from Artificial Intelligence, Databases, and Programming Languages. – New York (Springer).

BURTON, W. (1978): Efficient Retrieval of Geographical Information on the Basis of Location. - In: DUTTON, G. (Ed.): First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems. - Harvard papers on Geographic Information Systems; Cambridge, Massachusetts (Harvard Univ.).

CHEN, P. (1976): The Entity-Relationship Model: Towards a Unified View of Data. - ACM Transactions on Database Systems, 1, 1: 9-36; New York.

CODASYL, (1962): An Information Algebra. - Phase I Report, Commun. ACM, 5: 190-204; New York.

- - (1971): Data Base Task Group (DBTG) Report.

CODD, E.F. (1982): Relational Database: A Practical Foundation for Productivity. - Commun. ACM, 25: 109; New York.

DALTON, G. & WONG, A. (1978): Generation and Search of Clustered Files. - ACM Transactions on Database Systems, 3; New York.

DANGERMOND, J. & MOREHOUSE, S. (1987): Trend in Hardware for Geographic Information Systems. - In: CHRISMAN, N. (Ed.): Proc. 8th int. Symposium on Computer-Assisted Cartography; Baltimore.

DENNING, D.E. & SCHLOERER, U. (1980): A Fast Procedure for Finding a Tracker in a Statistical Database. - ACM Transactions on Database Systems: 5; New York.

DEWITT, D.E. & HAWTHORN, P.B. (1980): A Performance Evaluation of Database Machine Architectures. - In: ZANIOLO, C. & DELOBEL, C. (Eds.): Proc. VII int. Conf. on Very Large Databases: 199; Cannes.

DITTRICH, K.R. (1986): Object-Oriented Database Systems: The Notion and the Issues. - Proc. 1986 int. Workshop on Object Systems; Pacific Grove, Calif.

DUTTON, G. (1978): Navigating ODYSSEY. - In: DUTTON, D. (Ed.): First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems. - Harvard papers on Geographic Information Systems; Cambridge, Massachusetts (Harvard Univ.).

EGENHOFER, M. (1984): Implementation von MAPQUERY. - Rep. 79, Inst. f. Geodäsie und Photogrammetrie, ETH Zürich; Zürich.

- & FRANK, A. (1987): PANDA: An Object-Oriented Database Based on User-Defined Abstract Data Types. - Rep. 62, Surveying Engineering, Univ. Maine; Orono, ME.

FRANK, A. (1980): Landinformationssysteme - Ein Versuch zu einer Abgrenzung. Nachr. aus dem Karten- u. Vermessungswesen, 1, 81; Frankfurt a.M.

- (1981): Applications of DBMS to Land Information Systems. - In: ZANIOLO, C. & DELOBEL, C. (Eds.); Proc. VII int. Conf. on Very Large Databases: 488; Cannes.

- (1982a): PANDA: Pascal Netzwerk Datenbank. - Rep. 62, Inst. f. Geodäsie u. Photogrammetrie, ETH Zürich; Zürich.

- (1982b): MAPQUERY: Data Base Query Language for Retrieval of Geometric Data and their Graphical Representation. - SIGGRAPH '82, Conf.Proc., Computer Graphics, 16: 199; San Francisco.

- (1983a): Datenstrukturen für Landinformationssysteme - semantische, topologische und räumliche Beziehungen in Daten der Geo-Wissenschaften. - Diss. ETH Zürich; Zürich.

- (1983b): Storage Methods for Space Related Data: The FIELD TREE. - In: Barr, M. (Ed.): Spatial Algorithms for Processing Land Data with a Microcomputer, Boston (Lincoln Inst. of Land Policy).

- (1984): Extending a Network Database with Prolog. - In: KERSCHBERG, L. (Ed.): Expert Database Systems. - Proc. 1st. Workshop on Expert Database Systems; Kiawah Island, South Carolina.

- (1985a): Distributed Database for Surveying. - ACSM Journ. Surveying Engineering, 111, 1; New York.

- (1985b): Integrating Mechanisms for Storage and Retrieval of Data. - Proc. on the Workshop on Fundamental Research Needs in Surveying, Mapping and Land Information Systems, V.P.I. and S.U.; Blacksburg, Virginia.

- (1986a): PANDA: An Object Oriented Pascal Network Database Management System. - Rep. 57, Surveying Engineering, Univ. Maine; Orono, ME.

- & KUHN, W. (1986): Cell Graph: A Provable Correct Method for the Treatment of Geometry. - Proc.int.Symp.onSpat Data Handling, Seattle: 411-436; Williamsville.

- & TAMMINEN, M. (1982): Management of Spatially References Data. - In: LEICK, A. (Ed.): Proc.int.Symp.Land Information on the Local Level, Univ. of Maine; 330; Orono, ME.

GUTTAG, A. (1984): New Features for a Relational Database System to Support Computer Aided Design. - Mem. UBC/ERL M84/52, Electronic Research Laboratory, College of Engineering, Univ. California; Berkley.

HÄRDER, T. (1986): New Approaches to Object Processing in Engineering Database. - Proc. 1986 int.WorkshoponObject Systems; Pacific Grove, Calif.

- & REUTER, A. (1982): Database Systems for Non-Standard Applications. - Report 54/82, Fachber. Informatik, Univ. Kaiserslautern; Kaiserslautern.

LAMPSON, B.W. (1981): Distributed Systems - Architecture and Implementation. - Lecture Notes in Computer Science, 105; Berlin (Springer).

MANOLA, F., ORENSTEIN, J. & DAYAL, U. (1987): Geographic Information Processing in the Probe Database System. - In: CHRISMAN, N. (Ed.): Proc. 8th int. Symp. on Computer-Assisted Cartography; Baltimore.

MEYROWITZ, N. (1986): Object Oriented Programming Systems, Languages and Applications (OOPSLA). - Conf.Proc., ACM SIGPLAN Notices, 21, 11; New York.

MYLOPOULOS, T. (1980): An Overview of Knowledge Representation. - Proc. of the Workshop on Data Abstraction, Databases and Conceptual Modeling, SIGMOD Record, 11: 5; Pingree Park, Colorado.

NIEVERGELT, J. (1984): The Grid File: An Adaptable Symmetric Multikey File Structure. - ACM Transactions on Database Systems, 9; New York.

PARNAS, D.L. (1972): On the Criteria to be Used in Decomposing Systems into Modules. - Commun. ACM, 15; New York.

PETERSON, F.K. & MCLAUGHLIN, J. (1986): Multi-Purpose Land Information Systems A must to Improve the Quality of Life. - Professional Surveyor, 6, 4: 21-24; Fallchurch.

PLOUFFE, W. (1984): A Database System for Engineering Design. - Quat.Bull. of the IEEE Computer Society Technical Committee on Database Engineering, 7, 2.

SAMET, H. (1984): The Quadtree and Related Hierarchical Data Structures. - ACM Computing Surveys, 16, 2: 187-260; New York.

SCHEK, H.-J. & LUM, V. (1983): Complex Data Objects: Text, Voice, Images: Can DBMS Manage them?. - In: SCHKOLNICK, M. & THANOS, C. (Eds.): Proc. 9th int. Conf. on Very Large Databases; Florence.

SMITH, J.M. & SMITH, D.C.P. (1977): Database Abstration: Aggregation and Generalization. - ACM Transaction on Database Systems, 2; New York.

TAMMINEN, M. (1982): Efficient Spatial Access to a Database. - Proc. SIGMOD Conf.; San Francisco.

TROYER, O.D., KEUSTERMANS, J. & MEERSMAN, R. (1986): How Helpful is an Object-Oriented Language for an Object-Oriented Database Model? - Proc. 1986 int. Workshop on Object Systems; Pacific Grove, Calif.