INT. J. GEOGRAPHICAL INFORMATION SCIENCE, 2002
VOL. 16, NO. 8, 713–748

Research Article

Generic query tool for spatio-temporal data

PETER VAN OOSTEROM†, BART MAESSEN‡ and
WILKO QUAK†

†Department of GIS Technology, Faculty of Civil Engineering and Geosciences, Delft University of Technology, P.O. Box 5030, 2600 GA, The Netherlands
‡Cadastre Netherlands, P.O. Box 9046, 7300 GH Apeldoorn, The Netherlands;
e-mail: oosterom@geo.tudelft.nl, bart.maessen@kadaster.nl, quak@geo.tudelft.nl

(*Received 3 March 2001; accepted 21 September 2001*)

Abstract. Geographical information systems are more and more based on a DBMS with spatial extensions, which is also the case for the system described in this paper. The design and implementation of a generic geographical query tool, a platform for querying multiple spatio-temporal data sets and associated thematic data, is presented. The system is designed to be generic, that is without one specific application in mind. It supports ad-hoc queries covering both the spatial and the thematic part of the data. The generic geographic query tool will be illustrated with spatial and thematic Cadastral data. Special attention will be given to the temporal aspects: a spatio-temporal data model will be described together with a set of views for easy querying. DBMS views play an important role in the architecture of the system: integration of models, aggregation of information, presentation of temporal data, and so on. The current production version of the geographic query tool within the Dutch Cadastre is based on Geo-ICT products with a relatively small market share (Ingres and GEO++). A new prototype version is being developed using mainstream Geo-ICT products (Oracle and MapInfo). First results and open issues with respect to this prototype are presented.

1. Introduction

Traditionally, most Geographical Information Systems (GIS) can only query and display information that is under the control of the specific system (e.g. stored in a proprietary file format). Spatial data that are created and managed outside a specific GIS cannot be queried in this GIS. Since more and more spatial functionality is available within DBMSs (Database Management Systems, outside a GIS), a more generic GIS approach is required, which can work with spatial data in a model not controlled by the GIS. In this paper a tool based on a generic GIS using a DBMS is described.

When trying to design a generic information system, one is first confronted with the fundamental question: is it possible at all to develop an information system without a specific application (data and functionality) in mind? It could be argued that it is impossible because it is not known which data sets are needed, which functions are needed, what the user interface should look like, and so on. However, the system presented in this paper is an attempt to develop such a generic system. Of course, the application domain of the system is not unlimited and could be characterized as a geographical query tool. It should be noted that even this focus is not strictly needed. The tool could also be used for non-geographical data and for data entry or manipulation. Once loaded with (different) data sets, the system could be described as a spatial data warehouse. In this paper the term geographical query tool is preferred.

The usefulness of the generic geographical query tool is tested in practice. The integration of multiple geographical data sets and associated thematic (legal) data in one DBMS is described within the context of the Dutch Cadastre. The data set is nationwide and available for analysing and performing consistency checks on the Cadastral data. The purpose is to create an environment with easy access to all the data. Therefore a user-friendly interface is needed on top of this DBMS. The Dutch Cadastre has the advantage over most other cadastral organisations that it maintains both the cadastral map and the (administrative legal/tax) registers unlike many other countries where these registrations are the responsibility of different organizations; e.g. in Italy (Arcieri *et al.* 1999) or France (Spéry *et al.* 2001).

Section 2 gives an overview of the generic geographical query tool system architecture. The system architecture of a generic geographical query tool is based on two components: the backend, a spatial DBMS, and the frontend, a (geo)graphical user interface for comfortable query formulation and showing the results. The frontend is generic in two ways. First, it supports any data model. Second, it uses the (spatial) functionality offered by the DBMS. The backend is generic in the sense that it should be able to handle the formulated queries efficiently. That is, the backend should have a good query optimizer taking into account both spatial and thematic aspects of the query. Section 3 introduces the basic Cadastral data models for the geometric and administrative data in our case study with emphasis on the temporal aspects of the geometric model. In traditional (non-spatial) information systems users interact with data by specifying a key, such as an identification number. An alternative is browsing or viewing large tables of information which are ordered; e.g. alphabetically. Using administrative and geometric data in an integrated manner gives a new entrance: the map.

In the DBMS of the generic geographical query tool, the original data models (base tables) are not changed. However, DBMS *views* are used to present the data in a more appropriate manner: integrated, aggregated, time specific and with cartographic attributes; see §4. Specific add-ons to the basic components of the generic architecture, extensible relational DBMS and generic GIS-frontend, are described in §5. The custom made add-ons are used for: *easier access to data*: select just one function, instead of querying several tables; *analysis not possible in a relational DBMS*: e.g. intersection of a topologically structured area feature with a polyline; and *introduction of new interface concepts*: e.g. the 'active set'. The usefulness of the tool is shown by a number of quite different cases from the real world, each illustrating different aspects of the system. These applications are based, among others, on the following generic concepts: spatial aggregates, historic spatial data, spatial join, integrated geometric and thematic data, SQL and shell scripts. A number of practical use cases are described in §6 with emphasis on temporal applications. Section 7 shows the first results of the recent investigations within the Dutch Cadastre to migrate the query tool environment towards mainstream Geo-ICT solutions, summarizing the obtained results and the remaining open issues. The paper concludes with a short list of other practical uses and future developments in §8. In summary, the generic geographical query tool can be used for easy querying, analysis and visualisation of any geographical and related thematic data needed during solving many types of different tasks.

2. System architecture

This section gives an overview of the architecture of the generic geographical query tool. First, some remarks with respect to developing generic (DBMS-based) information systems in general are made in §2.1. The geographical query tool system is based on the backend Ingres DBMS (ASK-OpenIngres 1994, van Oosterom 1997) and the frontend GEO++ GIS package (Professional Geo Systems (PGS) 1996, Vijlbrief and van Oosterom 1992). The DBMS backend will be described in §2.2 and the GIS frontend is described in §2.3.

2.1. *Generic information systems*

Designing and developing a *generic* information system, such as the query tool, is not trivial and requires specific techniques. The implementation cannot use hard-coded names of objects (tables), attributes, operators, etc. The foundation of most information systems is a DBMS (in contrast to traditional GIS, which are file based) and this is also the case for the generic geographic query tool. The DBMS-based architecture also gives the first indications of how to develop a generic application. One of the goals of the language SQL itself is to separate the high-level functionality (what is requested by the user) from the implementation (how is this obtained). Further, the current data model managed by the DBMS can be obtained by querying system catalogues with meta data; below the example in Ingres (meta data in system catalogue `iicolumns`) and Oracle (meta data in system catalogue `ALL_TAB_COLUMNS`) to obtain the data model. Note that these queries have to return the description of both (real) tables and views. Further note that though SQL itself is standardized, obtaining the same information from the system catalogues differs between DBMSs.

```
/* Ingres example */
SELECT table_owner, table_name, column_sequence,
   column_name, column_internal_datatype,
   column_internal_length, column_internal_ingtype
FROM iicolumns
ORDER BY table_owner, table_name, column_sequence

/* Oracle example */
SELECT OWNER, TABLE_NAME, COLUMN_ID,
   COLUMN_NAME, DATA_TYPE, DATA_LENGTH
   FROM ALL_TAB_COLUMNS
   ORDER BY OWNER, TABLE_NAME, COLUMN_ID
```

The generic geographical query tool uses this information, that is, the names of the tables and the names and data types of attributes, in the interaction with the user. Note that a data type can be a geometric data type. The way geometric data types are managed by the DBMS can be either by specific data types such as point, polyline and polygon or by one generic geometric type (which can internally represent any number and combination of basic types, such as points, lines, and polygons).

The advantage of specific data types is that they allow the designer of the data model to specify more accurately the geometric type of the attribute of the feature in the real world. The advantage of the generic geometric type is that it is closed under the intersection operation: e.g. the intersection of two generic geometric types is always a generic geometry type. This is not true for the specific geometric data types such as a polygon: the intersection of two polygons can be a collection of points, polylines and polygons. Note that this closure property is nice, but there are other well-accepted data types that are also not closed under certain operations; e.g. integer under division: 1 divided by 2 is 0.5 and not an integer.

Another advantage of specific geometric data types is that the generic geographical query tool can present the user with visualization options suitable for a specific type: e.g. cartographic symbolization for a point, line style and width for a polyline, fill pattern for a polygon, etc. It is possible to think of a, less obvious, solution to present visualization options for the generic geometric type. The OpenGIS consortium has chosen in its simple feature specification standard SFS/SQL (Buehler and McKee 1998, Open GIS Consortium, Inc. 1999) for specific geometric types. There are several DBMSs available implementing these specific geometric types and functions; e.g. Informix (2000) or IBM DB2 (2000). Oracle has chosen the generic geometric type approach (internally inspired by the OpenGIS SFS model). The Ingres geometry implementation dates back several years before the OpenGIS SFS/SQL standard, but is also based on specific geometric data types within OME/SOL (Object Management Extension/Spatial Object Library) (ASK-OpenIngres 1994, van Oosterom 1997).

All modern (object) relational DBMSs can be extended with new (spatial) data types and functions/operators. Therefore the frontend cannot assume a fixed number of functions offered by the DBMS. Similarly to obtaining the data model from system catalogues (meta data), it is possible to obtain information about the available data types and operators from the system catalogues. For an operator this information consists of at least the name of the operator, the number of operands, the data types of the operands and the data type of the return value. This information is used by the generic geographical query tool to assist the user in the formulation of correct queries based on the operations offered by the DBMS. Note that there is no real difference between spatial and non-spatial data types with respect to this generic functional approach.

2.2. *DBMS backend*

Besides the ‘standard’ DBMS functionality, the DBMS should also give access to the description of the current data model and available data types and operators; see §2.1. Unfortunately in the previous subsection it became clear that this part of the DBMS is less standardized than the query language SQL. Different DBMS vendors have chosen different solutions, that is, they offer different system catalogues. This makes porting the generic geographical query tool from one DBMS to another more difficult than just linking the query tool system code with a different DBMS interface library.

The geographical query tool DBMS has a data warehouse nature. Periodically large amounts of data are copied into the query tool DBMS, creating a huge data set. The advantage of having all data integrated in one DBMS should not be countered by degraded interactive response times. In the generic geographical query tool, DBMS performance on a large data set should be virtually the same as in a

DBMS containing only a small data set. This can be achieved by applying proper (spatial) data clustering and indexing techniques. Also integrated (geometric and administrative) views using data from different tables must be at the same speed as pure geometric views; e.g. when using some administrative information from a related table to colour code a geometric entry on the map. The same holds for the temporal or historic views, an historic map or change over a period must be at the same speed as retrieving and displaying current date/time.

The key entries to the generic geographical query tool DBMS are in general a region (usually a rectangle, sometimes just one point), or an administrative identifier; e.g. parcel number, address or owner name. Whenever possible data are organised based on spatial location, because a typical map contains thousands of objects and it would be inefficient to retrieve them from different physical locations on disk. This is obvious for the geometric data; for example by using the spatial location code SLC (van Oosterom and Vijlbrief 1996). However, this is also applied to administrative data which can be linked in one way or another to spatial data; e.g. the owners of a parcel could be clustered by postal code of their residence. This enables spatial range queries to perform well in all situations including the integrated geometric-administrative views. The other entries are supported by secondary indices (b-tree (Comer 1979) or r-tree (Guttman 1984)), because they usually return one or a few results.

2.3. *GIS frontend*

One important requirement is that the generic geographic query tool must be open, i.e. it may not be based on a GIS vendor specific data format. This is a similar motivation as the one driving the OpenGIS consortium (Buehler and McKee 1998, Open GIS Consortium, Inc. 1999). However, the generic geographical query tool developments within the Cadastre started about seven years ago, before the OpenGIS standards and products were available. However, it is based on the same principle and it is expected that migration to OpenGIS products should not be too difficult (§7).

At this moment, many GISs still do not support this open DBMS approach. The traditional approach is that geometric data have to be stored in the GIS vendor-specific format. Quite often, GIS have the capability to access an external DBMS. Usually this means that the geometric information still has to be stored locally, but with the help of a key, the corresponding administrative data can be found in the external DBMS. This is not good enough, since complete integration of geometric and administrative data is the most efficient and consistent data management architecture. A generic geographical query tool must function as a frontend to the DBMS backend. The generic geographical query tool assists the user in posing questions through a user friendly interface. The generic geographical query tool generates ANSI SQL dynamically and processes the responses of the DBMS. GEO++ is a GIS which is designed to deal with this. Therefore, GEO++ can be described as a generic query tool for relational DBMS with spatial extensions.

3. Cadastral data

The generic geographical query tool will be illustrated with examples from a Cadastral system. To understand these examples, this section introduces the data model of the Cadastre in The Netherlands. The geographical data sets consist of large-scale topographic data and the cadastral maps of all the provinces in The Netherlands. Associated with the cadastral maps are administrative data, which are also organised per province, but stored in a central mainframe. The relationships between the parcels on the cadastral map and the administrative data are through the nationwide unique parcel numbers. The cadastral maps are based on a topologically structured model and manipulating area features in such a model involves navigation using the topology references to the boundaries. It is interesting to note that the topology speeds up the visualization compared to a representation without topology, because in the latter case all coordinates are transferred twice from DBMS to the application: once for the polygonal face on the left side and once for the polygonal face on the right side. The topographic maps and the cadastral maps contain the full history since their introduction in 1997. This is not (yet) the case for the administrative data.

Currently, the large-scale topographic and cadastral data are maintained by the LKI system (*Landmeetkundig Kartografisch Informatiesyteem* (in Dutch): Information System for Surveying and Mapping), which stores the data in an Ingres DBMS using OME/SOL (Object Management Extension/Spatial Object Library) (ASK-OpenIngres 1994, van Oosterom 1997). Legal and other administrative data related to parcels are maintained by the AKR system (*Automatisering Kadastrale Registratie* (in Dutch): Automated Cadastral Registration), which stores the data in an IDMS DBMS on an IBM mainframe. This data set will be referred to as the *administrative* data in contrast to the *geometric* data.

The generic geographical query tool has its own DBMS, which contains a copy of all geometric and administrative data in their original data models. Therefore, a good understanding of the two data models, as a case study, is important. These data models contain structures, which can be found in many other applications; e.g. metric, topology and measurement information (date, accuracy, type of measurement) within the geometric data and hierarchies, n-to-m relationships, generalization/specialization structures within the administrative data. These structures and their semantics are relatively difficult to deal with in a generic geographical query tool environment. However, they have to be included in a generic geographic query tool which is acceptable for users familiar with this model.

In §3.1, the geometric model of the Cadastre will be described in more detail. Followed by a description of the administrative (legal) model in the next subsection. The last subsection gives some numbers indicating the size of the query tool data set. Also, the load process of the data into the query tool DBMS is described.

3.1. *Geometric model*

Since 1997 the geometric DBMS keeps track of all geometry changes over time, that is, it contains a spatio-temporal data set. The geometric attributes of type point, polyline and box are stored in the relational DBMS together with the other attributes describing the measurement (data, accuracy, etc.).

The geometric data model for the cadastral *parcel* layer is based on winged-edge topology (Baumgart 1975) as described in (Lemmen and van Oosterom 1995, van Oosterom 1997, van Oosterom and Lemmen 2001); references in this topology model are visualized in figure 1. In addition to the topologically structured cadastral parcel layer, this model also includes *topographic* layers, which are not (yet) topologically structured. Though some operations in a relational DBMS are impossible on a topologically structured area feature (e.g. compute area), this structure has many advantages:

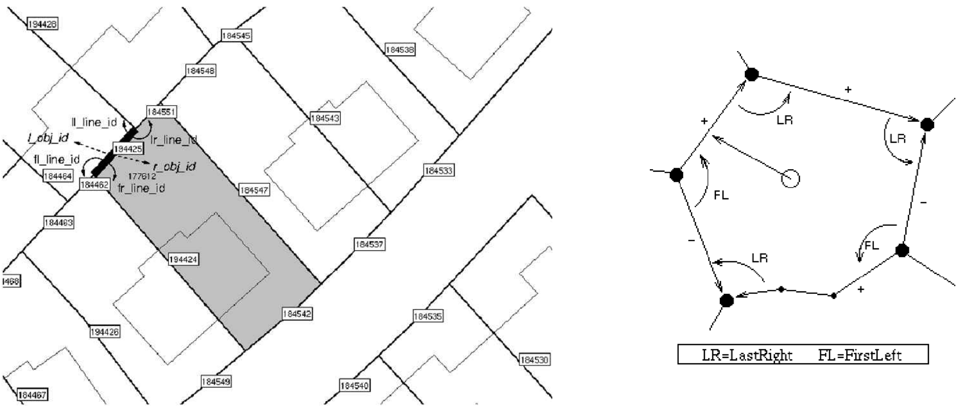


Figure 1. The topology references in the winged-edge structure of the geometric model.

- It avoids redundant storage of shared edges and vertices in a planar partition of space (and therefore is more compact than a full-polygon model).
- It is more efficient during the visualization of the data stored in the DBMS in some kinds of front-end, because less data has to be read from disk and transferred to the front-end.
- It is the natural data model for certain application; e.g. during surveying an edge is collected (together with non-geometric attributes belonging to a boundary) and not a polygon.
- The topology structure can be used efficiently in certain operations (e.g. find neighbour).

The next paragraphs elaborate the temporal aspects in the geometric model. First, the temporal model itself will be introduced, then some temporal examples will be given, and finally some open questions will be discussed.

3.1.1. *Spatio-temporal model*

In the last decade quite a lot of attention has been paid to methods of modelling, storing, indexing (clustering), editing (updating), manipulating, analysing, and visualizing, spatio-temporal data. Books or literature papers concentrate on one or more of these aspects. Good overviews of handling spatio-temporal data can be found in (Langran 1992, Al-Taha *et al.* 1994, Abraham and Roddick 1999, *CHOROCHRONOS: A Research Network for Spatiotemporal Database Systems* 1996–2000). Some papers emphasise modelling (Tryfona and Jensen 1999, Worboys 1994), some emphasise functionality or analysis (Peuquet and Wentz 1994, Raafat *et al.* 1994, Erwig *et al.* 1999, Kollios *et al.* 1999, Hornsby and Egenhofer 1998) and some access methods (Relly *et al.* 1999, Nascimento *et al.* 1999). Some papers specifically focus on temporal aspects in the cadastral domain: such as access and update in a distributed environment (Arcieri *et al.* 1999), model predecessor-successor relationships (Spéry *et al.* 2001), or event-based modelling (Chen and Jiang 1998).

The updates in the spatial database of the Cadastre are related to changes of a discrete type in contrast to more continuous changes in natural phenomena (Cheng and Molenaar 1998) or stock rates. The number of changes per year related to the total number of objects is relatively low. It was therefore decided to implement history at the tuple level, rather than at the attribute level, which requires specific database support or will complicate the data model significantly in a standard relational database. Note that instead of storing the old and new states, it is also possible to store the events only (Gold 1996, Claramunt and Theériault 1996). However, it is not easy to retrieve the situation at any given point in time. More information on event-based modelling is given in Chen and Jiang (1998).

Every object is extended with two additional attributes: `tmin` and `tmax`. This is similar to the Postgres model (Stonebraker and Rowe 1986). A temporal SQL extension is described by Snodgrass *et al.* (1994). In Voigtmann *et al.* (1996) a temporal object database query language for spatial data. The objects are valid from and including `tmin` and remain valid until and excluding `tmax`. Current objects get a special `tmax` value: `MAX_Time`, indicating they are valid now.

There is a difference between the *system (transaction)* time, when the recorded object changed in the database, and the *valid (user)* time, when the observed object changed in reality. In the data model `tmin/tmax` are system times. Further, the model includes the user time attribute `object_dt` (or `valid_tmin`) when the object was observed. Perhaps in the future the attributes `last_verification_dt` and `valid_tmax` could also be included, which would make it a *bitemporal* model.

When a new object is inserted, the current time is set as the value for `tmin`, and `tmax` gets a special value: `MAX_Time`. When an attribute of an existing object changes, this attribute is not updated, but the complete record, including the `oid`, is copied with the new attribute value. Current time is set as `tmax` in the old record and as `tmin` in the new record. This is necessary to be able to reconstruct the correct situation at any given point in history. The *unique identifier* (key) is the pair (`oid`, `tmax`) for every object version in space and time. Note that in theory it would have been better to use `tmin` in the key instead of `tmax`, because for a given object version `tmin` never changes and `tmax` does. In practice however, it does not make any difference.

For the topological references, only the `oid` is used to refer to another object and not `tmax`. In the situation that a referred object is updated and keeps its `oid`, then the reference (and therefore the current object) does not change. This avoids, in a topologically structured data set, the propagation of one changed object to all other objects as all objects are somehow connected to each other. In case the `oid` of a referred object has changed (becomes a different object), the referring object is also updated and a new version of the referring object is created.

3.1.2. *Some temporal examples*

Table 1 shows the contents of a database, which contained on 12 January one line with `oid` 1023. On 20 February this line was split into two parts: 1023 and 1268 (figure 2). Finally, the attribute quality of one of the lines was changed on 14 April.

Table 1. The contents of the spatio-temporal line table.

Oid	Shape	Quality	*t*min	*t*max
1023	(0,0), (4,0), (6,2)	1	12 January	20 February
1023	(0,0), (4,0)	1	20 February	14 April
1268	(4,0), (6,2)	1	20 February	MAX_Time
1023	(0,0), (4,0)	2	14 April	MAX_Time

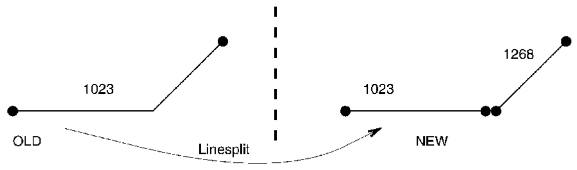


Figure 2. A 'line' split into 2 parts.

The SQL-queries in §6.3 show how easy it is to use this model and to produce the update files with new, changed, and deleted objects related to a specific time interval.

A query producing all historic versions of a given object only needs to specify the `oid` and leave out the time attributes. This does work for simple object changes, but does not work for splits, joins, or more complicated spatial editing. Therefore, explicit representation of predecessors and successors could be introduced by additional tables storing the many-to-many 'parent-child' relationships. Table 2 shows the history table `line_hist` of the previous line table example.

Additionally, the history table `parcel_hist` is shown with two non-simple edit events: three parcels (1234, 1235, and 1236) are created from two parent parcels (10 and 31) on 1 April further two parcels (2363 and 2364) are created from one parcel (77) on 10 June (figure 3; table 3). All parent parcels are deleted, that is, they get a `tmax` value and not a new successor record.

In order to avoid repeating parent and child oid in a 'cluster' edit operation in the `parcel_hist` table, an alternative history model could look like this:

```
parcel_hist1(cluster_nr, parent_oid)
parcel_hist2(cluster_nr, child_oid)
parcel_hist3(cluster_nr, time)
```

Both implementations correspond to the same conceptual view, a directed acyclic graph (DAG), explicitly representing the relationships between predecessors and

Table 2. The contents of the spatio-temporal `line_hist` table with predecessor and successor information.

parent_oid	child_oid	Time
1023	1023	20 February
1023	1268	20 February

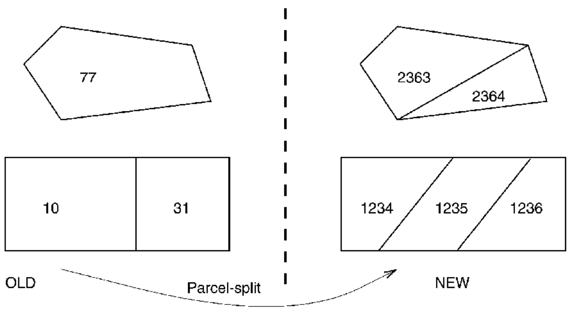


Figure 3. Some 'parcel' reorganizations.

Table 3. The contents of the spatio-temporal `parcel_hist` table with predecessor and successor information.

parent_oid	child_oid	time
10	1234	1 April
10	1235	1 April
10	1236	1 April
31	1234	1 April
31	1235	1 April
31	1236	1 April
77	2363	10 June
77	2364	10 June

successors. A prototype of such a system has been developed by Spéry *et al.* (2001). For the time being it has been decided *not* to maintain explicit history in `_hist` tables with predecessors and successors. However, this information can always be obtained by using spatial overlap queries with respect to the given object over time, that is, not specifying `tmin/tmax` restrictions.

3.1.3. *Some open questions*

Although many aspects of maintaining topology and time in a database have been described, there are still some open questions: 1. should we try to model the future?, and 2. how long should the history be kept inside the database tables? The current proposal is to keep the information in the database forever. Note that the historic data will not only increase the size of the database, but will also slow down the response times as the data has to be selected out of a larger data set. However, by using good access methods, the response times should be in the order of log(n) (where n is the total number of objects). In addition, the hardware is getting faster. An alternative could be to move historic data to another table (with the same attributes). This solution can be made transparent to the application by using rules/procedures and views. So, if it becomes necessary in the furture, this can be achieved without modifying any application.

Returning to the first question: in addition to the history we might also want to model the (plans for the) future. In contrast to the past where there is only one time 'line', the future might consist of alternative time 'lines', each related to a different plan. There is a different type of 'time topology' for these future time lines (Frank 1994). In this case multiple versions are needed (Easterfield *et al.* 1990). In The Netherlands there are detailed design plans called 'matenplannen'. These can be included in the database with `object_dt` set to some time stamp in the future, but they cannot be integrated in the topological model.

3.2. *Administrative model*

The administrative data model is based on a few key concepts: *object*, *subject*, and *right*. Objects (parcels) and subjects (persons) have an n-to-m relationship via rights (figure 4); a subject can have rights related to several objects (e.g. a person owning three parcels) and an object can be related to multiple subjects. Two examples of the latter: an object is owned by two partners or an object is leased by one subject to another subject. There are two types of subjects: *natural persons* and *non-natural*

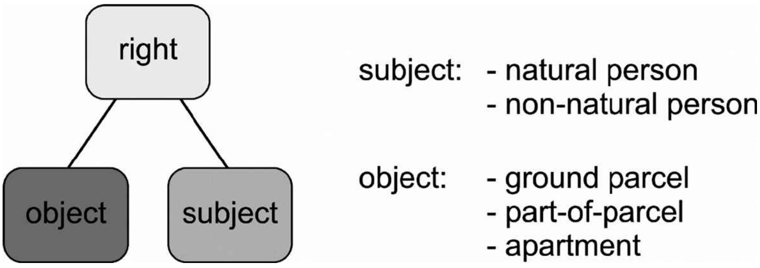


Figure 4. The core of the administrative data model: the n-to-m relationship between objects and subjects via rights.

persons (organizations), having some attributes in common, but also each having their own attributes.

In turn, the objects can be one of three basic types: *complete ground parcels*, *part-of-parcels*, or *apartments*. In The Netherlands a part of a parcel can be sold, as an objects, before it has been measured by the surveyor. These part-of-parcels again can be sold in part. This results in an hierarchy which is represented by a tree structure with the root representing the ground parcel (figure 5). The rights are only related to the leaves in the tree, that is, part-of-parcels not being subdivided any further. The base parcel numbers of the identifiers of a ground parcel and a part-of-parcel are the same (number 12 in figure 5). The difference can be found in the, so-called, *index* part of the identifier. For ground parcels this is always 'G0', for part-of-parcels this is of the form 'D1', 'D2', and so on. The link between the geometric model and the administrative model is based on the ground parcel (number), which is present in both models. Once the part-of-parcels have been surveyed, they become new complete ground parcels, with their own new base parcel number. The new base parcel number is no longer related to its original parent. Actually, the base part also includes the municipality and section codes in addition to the parcel number. An example of the complete identification of an object is 'WDB02B 02762G0000', a ground parcel in the municipality with code 'WDB02', section 'B' and number '02762'. The municipality and section code are not shown in the figures for readability.

The apartment objects are related to an apartment complex in the same manner as part-of-parcels are related to ground parcels: in an hierarchical structure. The apartment complex itself can be composed of multiple (disconnected) ground parcels.

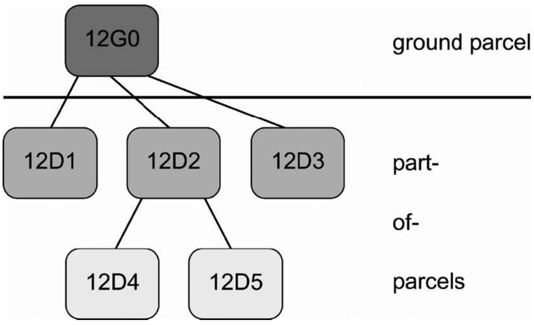


Figure 5. The tree structure representing the part-of-parcel hierarchy.

It is not possible for an apartment complex to be based on a part-of-parcel. This could theoretically occur if one tries to sell a part of the ground parcel, defining an apartment complex, before it is measured. However, in this case the ground parcel has to be surveyed first. Then it will be split into new ground parcels and not via part-of-parcels. After that the parcel can be removed from the apartment complex and can be sold. Note that the base parcel number of the apartment complex (number 15 in figure 6) is different from the base parcel number of the related ground parcels (numbers 8 and 9 in figure 6). The index part of the apartment complex identifier is always 'A0', the individual apartment objects have the same base parcel number and index parts which look like 'A1', 'A2', etc. (figure 6).

3.3. *Loading and size of the data set*

In this subsection the status of 1 September 2000 (and compared to the situation of 1 October 1999) is given. In the meantime the data sets have been growing. This is a result of the ever-increasing complexity of the real-estate situation and also as a result of keeping history in the geometric data set. A few numbers describing the size of the geometric data including history are given in the upper half of table 4.

Due to geometric quality improvement (adjustment of Cadastral map and large-

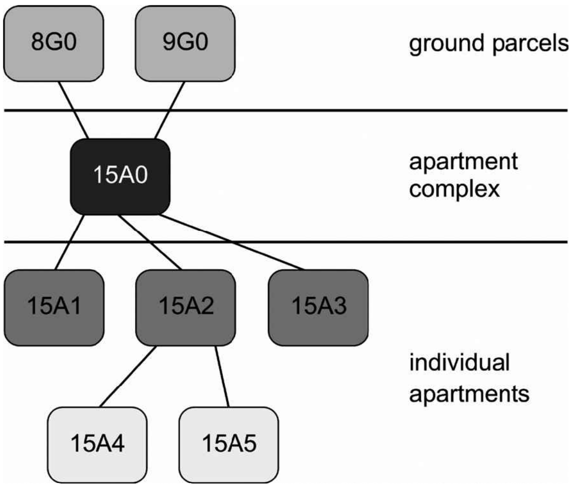


Figure 6. The structure relating ground parcels and apartments.

Table 4. Number of records per table in the query tool DBMS.

Table	1 September 2000	1 October 1999
parcel	15 700 000	9 300 000
boundary	41 100 000	25 200 000
topographic line	51 900 000	31 200 000
symbol	7 200 000	5 100 000
text label	7 400 000	5 200 000
object	7 700 000	7 500 000
object address	9 900 000	9 700 000
subject	7 300 000	7 100 000
right record	10 600 000	10 100 000
object limitation	2 200 000	1 900 000

scale topographic map) and the maintenance of history, the growth of the geometric data set is larger than one would normally expect. The total number of different line segments in the data set is over 400 000 000 (guessed status 2000, computed status 1999: 250 000 000). The administrative (legal) data sets contain the amount of data (without history) as given in the lower half of table 4. Note that (1) an object is either a parcel or a part-of-parcel or an apartment, (2) a subject is either natural or non-natural, (3) a right record is a relationship between an object and a subject, and (4) an object limitation is also called a legal notification, restricting the use of the object for some reason.

The integration of the administrative data and geometric data models is realized through views as described in §4. The query tool DBMS is periodically filled with complete copies of the 15 decentralized spatial data sets from LKI and 15 centralized administrative data sets from AKR (figure 7). Currently, the data are loaded four times per year into a single Ingres DBMS. Loading the data includes defining indices, computing geometric aggregates, collecting statistics (the basis for producing good query plans by the query optimizer) and making checkpoints. The whole process takes between three and four days on a (Compaq) AlphaServer 4100 with 1 CPU (598 MHz), 2 Gb main memory and about 500 Gb disks in the form of RAID5 (for the software) and RAID0 + 1 (for the data storage using striping and mirroring). The result is a 80 Gb DBMS including the index structures (status 1 September 2000). During loading, the previous version of the query tool DBMS remains available to the users.

Note that instead of copying all production data sets into one query tool DBMS, it would also be possible to define the tables and the views in a distributed DBMS. For applications, which do need up-to-date information, but 'move' smaller amounts of data, this may be the optimal solution. Within the Dutch Cadastre this

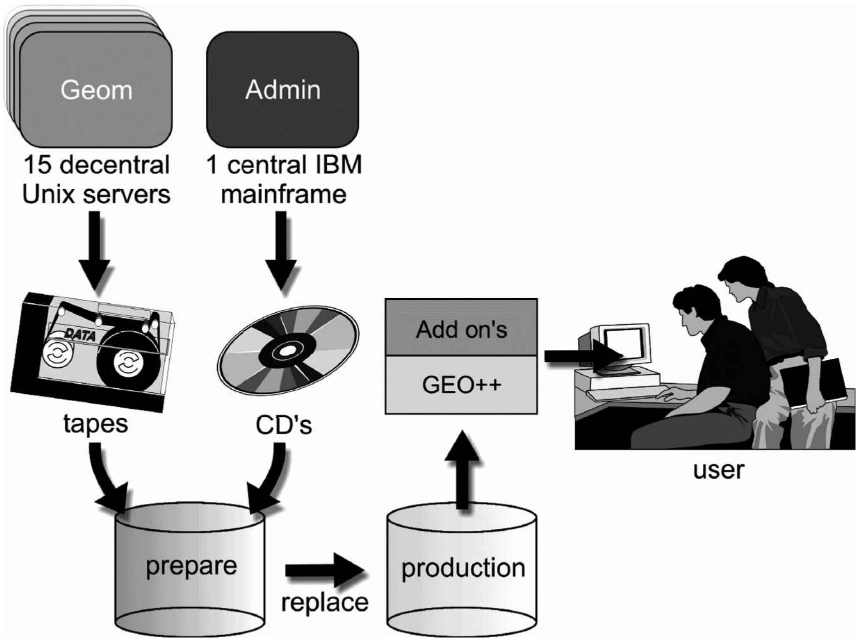


Figure 7. Overview of loading and using the query tool DBMS.

approach is used for legal information and small maps ('kadaster netwerk') and for certain types of object notifications maintained by the municipalities. A distributed approach is also described in the context of the Italian Cadastre (Arcieri *et al.* 1999). In the case of the Ingres DBMS, this could involve using the Net and Star products. This should also work in combination with other relational DBMSs using 'gateways'. However, the query tool DBMS is large and its applications often require the combination of large amounts of data. Organizing the data (clustering and indexing) is important for performance in these applications. This may not always be possible in the original production systems, especially in the administrative system. Therefore, the data is periodically copied into the query tool DBMS.

4. DBMS views

DBMS views play a key role in the architecture of the generic geographic query tool. The query tool DBMS contains tables with the unaltered copies of the data from the production DBMSs. The views are a generic concept and in the generic geographic query tool used to:

- integrated data from different tables in one view (§4.1).
- visualize historic data (§4.2).
- visualize same table using different geometric primitives (§4.3).
- derive cartographic attributes, such as colour, width, symbol type, from other attributes (§4.3).
- derive thematic aggregates without storing the result (§4.4).
- present (encoded) attributes in a clear way.

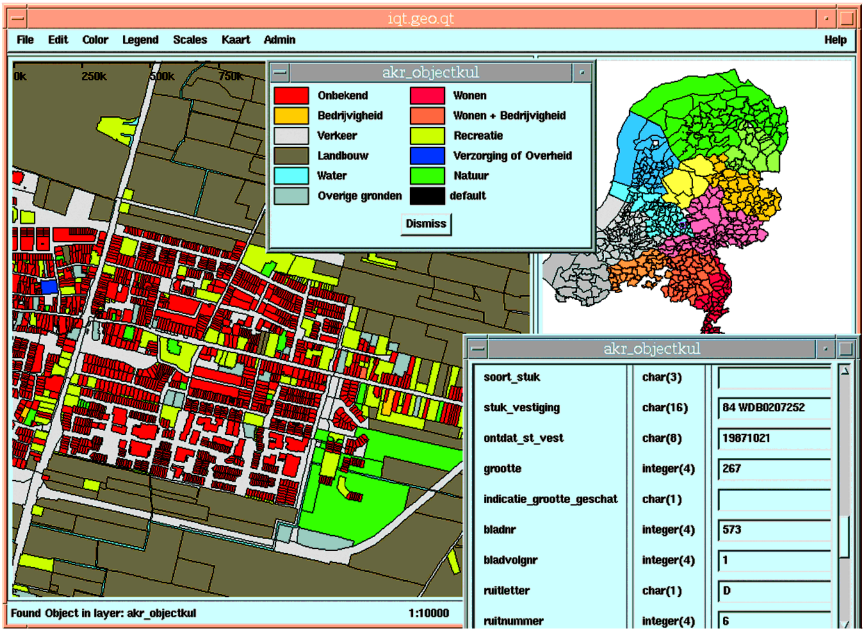
Examples of the last are time stamps encoded with integers but visualized as readable strings such as '22-04-1998 09:52:50' or legal right codes short-coded with two characters which can be represented better with a full string.

4.1. *Thematic views*

Integration of data in a relational DBMS environment involves integration of data models. Each data model represents a part of the real world, and its content is maintained by a specific application. Therefore, the data models cannot be changed. Integration is realized by defining DBMS views using the relational *join* mechanism. The left-hand side of figure 8 colours the land use of parcels, based on the attribute `culture_code`. The parcels themselves are stored in the geometric data model. The `culture_code` is stored in the administrative data model together with other administrative information such as prices and owners. The integrated model can be realized with the following DBMS view:

```
create view parcel_culture as
   select ap.culture_code, gp.location, ...
   from   parcel gp,  /* geometric parcel */
          object ap  /* administrative parcel */
   where  gp.municip=ap.municip and
          gp.osection=ap.osection and
          gp.parcel=ap.parcel
```

In the introduction of this paper it was stated that the map is a new entry to the data. Based on this principle, simply clicking on a parcel gives the corresponding administrative and geometric properties; see right hand side of figure 8. In the



🟥	Onbekend	🟥	Wonen
🟧	Bedrijvigheid	🟧	Wonen + Bedrijvigheid
⬜	Verkeer	🟨	Recreatie
🟫	Landbouw	🟦	Verzorging of Overheid
🟦	Water	🟩	Natuur
⬜	Overige gronden	⬛	default

soort_stuk	char(3)	
stuk_vestiging	char(16)	84 WDB0207252
ontdat_st_vest	char(8)	19871021
grootte	integer(4)	267
indicatie_grootte_geschat	char(1)	
bladnr	integer(4)	573
bladvolgnr	integer(4)	1
ruitletter	char(1)	D
ruitnummer	integer(4)	6

Figure 8. Land use of parcels.

parcel_culture view you can see that municip(ality), (o)section and parcel (number) are part of the key on which the geometric and administrative tables are joined. The cadastral map in The Netherlands is divided into municipalities. These municipalities consist of sections. In turn these sections are divided into parcels, which form the basic elements of the cadastral map. Therefore, it is implicitly stored to which municipality a parcel belongs.

A difficulty is that the administrative data model contains tree-like structures of unknown depth (§3.2). These are used to represent the part-of-parcel and apartment hierarchical structures and the relationship to the ground parcel. It was decided to flatten these structures, that is, only the information related to the top nodes, the leaf nodes and their (derived) relationships are stored.

4.2. *Temporal views*

To visualize spatio-temporal data (Langran 1992, Armenakis 1996, Kraak and MacEachren 1994) specific techniques are required in the generic geographical query tool. In the generic geographical query tool five sets of views are available for the manipulation of the spatio-temporal data, which is a generic solution. These are available for all spatial tables. The first set of views can be used to produce a ‘snap-shot’ view with the moment in time being the same fixed date/time of the administrative data. The parcel used in §4.1 was not a base table, but was the following view (assume $AKR_T holds the administrative date/time):

```
create view parcel as
 select gp.location, ...
 from   temporal_parcel gp,  /* geometric-temporal parcel base table */
 where  gp.tmin<=$AKR_T and $AKR_T<gp.tmax;
```

The next two sets of temporal views are similar to the 'snap-shot' view, but the query tool users can specify their own dates/times for two moments in time: `$BEG_T` and `$END_T`. In the case of the parcel table these views are called `parcel_beg` and `parcel_end`. With these views the users can display two maps of the same region, but related to two different moments in time.

The last two sets of temporal views are used to visualize the changes in the form of the new and deleted entities over a specified time interval (`$BEG_T`, `$END_T`]. In case of the parcel table these views are called `parcel_new` and `parcel_del`. These views can be displayed on top of a reference snap-shot map.

```
create view parcel_new as
   select gp.location, ...
   from   temporal_parcel gp,  /* geometric-temporal parcel table */
   where  $BEG_T<=gp.tmin and gp.tmin<$END_T;
```

The production of update files for customers of cadastral data is based on the same type of queries (§6.3). More advanced visualization techniques for showing spatial temporal data may be implemented in the future; e.g. time animation or time as a third dimension.

4.3. *Spatial views*

Some tables may have multiple geometric attributes and in these situations, these entities can be visualized on the map in multiple ways. For example, a parcel has a point attribute, it has a minimal bounding box, and it has a topologically structured area. Different view names are used to make these different visualization methods available to the user; e.g. in the case of a parcel: `parcel_label` (centroid or point location suitable for label placement), `parcel_box` (minimal bounding box of a parcel), and `parcel` (topologically structured area).

Cartographic attributes, such as colour, width, symbol type, etc., can be derived from other attributes and specified in the form of a view. The example below will visualize the difference between the official legal area and the measured area by computing a colour value code `geo_colour` in the range 15–20 in case the official area is larger than the measured area and 20–25 otherwise:

```
   create view parcel_area as
     select     gp.location,
                geo_colour=20+integer  (5*(gp.measured_a  -  gp.le-
                gal_a)/
                gp.measured_a),
                ...  /* other attributes */
     from       parcel gp;  /* geometric parcel */
```

4.4. *Aggregate views*

Overview or summary maps can be based on spatial aggregation of detail information. If the data are aggregated in terms of relational queries, then this will easily translate into a map. Figure 9 shows the average price of the parcels per municipality created by the view below:

```
   create view avg_price_municip as
     select     avg(ap.price), ap.municip
     from       object ap  /* administrative_parcel */
     group by   ap.municip
```

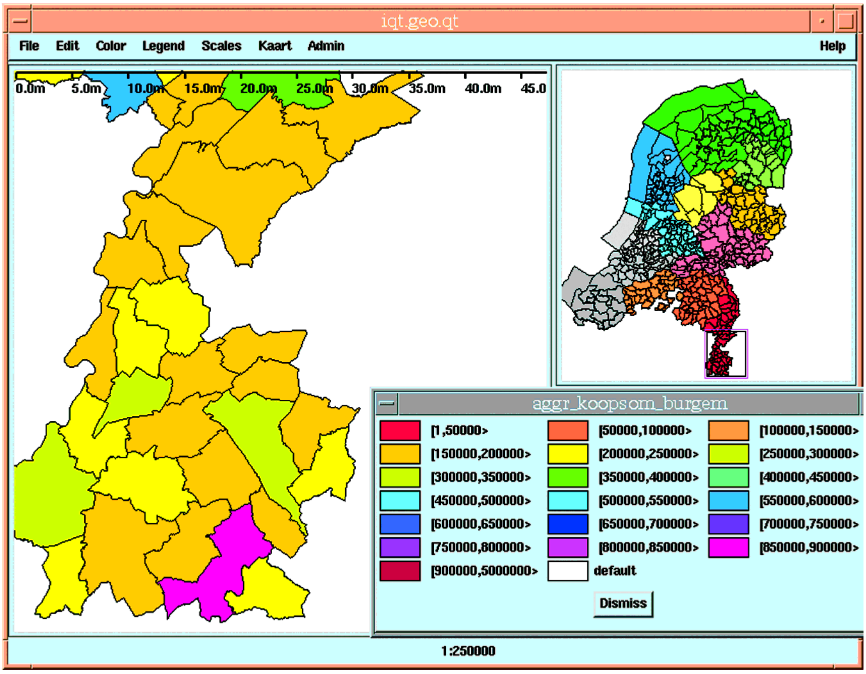


Figure 9. The average parcel price per municipality.

In this case we were able to aggregate the average price of a parcel and group this by using an attribute within the same table; municip, the municipality code. However, it is also relatively easy to aggregate to regions specified by another organization; e.g. census regions of the CBS, Central Bureau for Statistics.

```
create table census_region (name text(20), region long polygon);

create view avg_price_census as
  select      avg(ap.price), cr.name, cr.region
  from        object ap,     /* administrative_parcel */
              census_region cr
  where       inside (ap.location, cr.region)=1
  group by  cr.name, cr.region;
```

In this case, a spatial join is used, using the `inside` operator to relate the parcels to the census regions. Spatial indexing and clustering are very important for efficient manipulation of the view. One thing which was not solved by views was spatial aggregations; that is, deriving the larger spatial units from the smaller ones. These are computed outside the DBMS once and stored with topology in the DBMS; e.g. Cadastral sections or municipalities derived from parcels. The spatial aggregates are used as a basis for visualizing aggregated thematic data and for orientation purposes. The definition of the thematic aggregates in a generic manner still has to be improved in the generic geographical query tool. This is quite a challenge as there are many degrees of freedom when specifying an aggregation: spatial unit (section, municipality, province); temporal unit (one moment in time or even a period); aggregate function (sum, min, max, avg); thematic attribute; and additional constraints to the selection. How can these degrees of freedom be offered in an elegant way to the users?

5. Cadastral add-ons

With the generic geographical query tool based on standard Ingres and GEO++, it is possible to query and visualize the base tables and views in a user-friendly manner. It is also easy to specify a selection condition for a table or view by graphically creating the where-clause of the SQL statement. However, without reducing the value of the generic geographical query tool concept, this section presents a number of Cadastral specific add-ons for more easy access to the data. The *first* type of extensions developed by the Cadastre run inside the DBMS. Using Ingres OME (Object Management Extensions) types, both spatial and traditional types, are extended with new functions. Added, for example, are: the union function of two boxes or the translation function of an integer encoded date/time into a readable date/time string. These additional functions are used in the view definitions, but can also be used to specify a selection condition in the where-clause.

The selection results can be displayed in tabular format. In case the selection contains an attribute of type point, polyline or polygon, then GEO++ can also visualize this in a map format and other attributes can optionally be shown as text labels. The *second* type of cadastral extensions include additional visualization functions within the GIS frontend GEO++ for: a generic line (either a circular arc or polyline), cartographic text (scaling and rotating), and area based on topology (possibly including circular arcs in the boundary).

Both Ingres and GEO++ (data model driven using the DBMS system catalogue information) have no knowledge of the semantics of the data models being manipulated. This means that these tools are general and can be used in many situations. For the generic geographical query tool system it was decided to extend GEO++ with a few add-ons based on cadastral data model semantics. These add-ons form the *third* type of extensions and they are implemented using GEO++'s object-oriented scripting language Builder (Vijlbrief and van Oosterom 1992). To the user of the generic geographical query tool, these add-ons are available via two additional pull-down menus: one for the 'map entrance' to the data (figure 10) and one for the 'administrative entrance' to the data (figure 13).

These custom made add-ons in the generic geographical query tool are used for easier access to data, that is, the user has to select just one function from a pull-down menu, instead of querying several tables to obtain a specific result (§5.3). Also, the add-ons are used when the analysis is not possible within a relational DBMS. For example, the selection based on intersection of a topologically structured area feature with an input polyline (§5.2). Further, a new interface concept has been introduced in GEO++ by means of the add-ons, the *active set* (§5.1). The add-ons run on top of GEO++ and will be described in the subsequent sections.

5.1. *The active set concept*

During the design and development of the generic geographical query tool it became clear that there was a need for selecting objects (parcels) and saving them for further processing. For this purpose the *active set* concept was introduced, which is used throughout the interface of the add-ons. Parcels (objects) can be either selected geographically or administratively and be added to the active set. One can pan and zoom to the selected parcels in the active set on the map. One can also obtain administrative information related to the parcels in the active set. The active set concept itself is generic and could and is also used in other generic (geographical) information systems; e.g. the 'power selector' in MicroStation (Bentley MicroStation (2001).

5.2. *The geographical interaction extensions*

In the map 'kaart' pull-down menu, the extensions with mainly geographic interaction can be found; see figure 10. Translated from this figure the options are: (1) ParcelManager, (2) Find parcel by number, (3) Find parcel by coordinate, (4) Select neighbours, (5) Select parcels by cartographic text, (6) Select parcels within distance, (7) PointManager, (8) LineManager, (9) AreaManager, (10) Map history, (11) Additional map window, and (12) Print parcel. Each of these options will give the user a dialogue form controlling the (parameters of the) requested actions. It goes too far to describe all dialogue forms in detail and most operations should be clear by their name.

A few examples will be highlighted. Find parcel by coordinate 'jumps' to a specific coordinate and selects the parcel, that is, adds it to the active set. Find parcel by number jumps to a specific parcel and selects this parcel. The LineManager can be used for selecting all parcels, which are crossed by a polyline; e.g. planned road. Due to the topology model these queries cannot be posed as straightforward SQL selections. In the GIS frontend GEO++, the polygons of the relevant parcels are formed using the topology information from the DBMS. Then the actual test, e.g. polyline-crosses-polygon, is done in the frontend. In an object oriented DBMS, these queries could be completely executed within the DBMS, but this is not the case in a relational DBMS. Figure 11 shows the Amsterdam Subway route and the corresponding parcels that are crossed. The LineManager dialogue form is shown in figure 12 and enables the user to create new lines, specify the width of buffer zones, select the parcels crossed, etc.

5.3. *The administrative interaction extensions*

In the 'admin' pull-down menu, the extensions with mainly administrative interaction can be found (figure 13). Translated from the pull-down menu in this figure

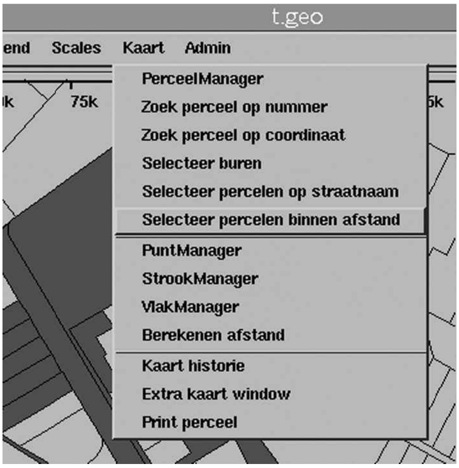


Figure 10. The map 'kaart' pull-down menu (Dutch interface).

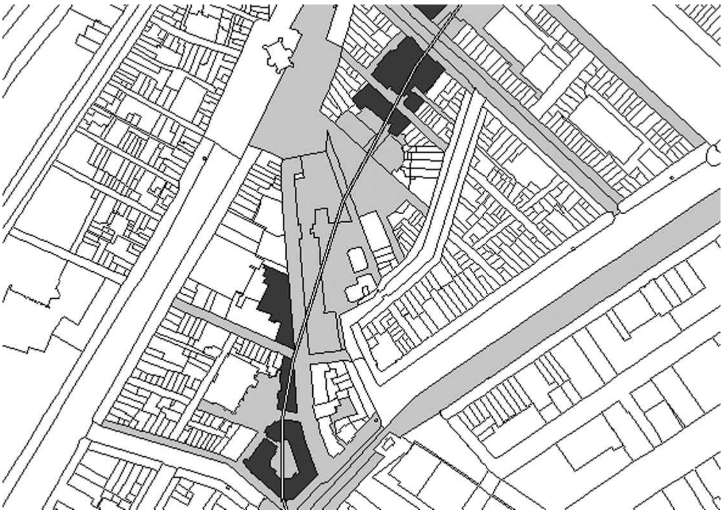


Figure 11. Parcels crossed by the Amsterdam Subway route.

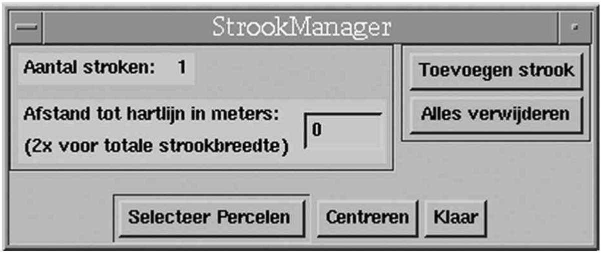


Figure 12. The dialogue form LineManager (StrookManager in Dutch) to manipulate line selection.

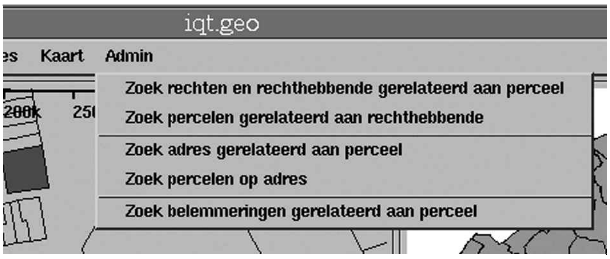
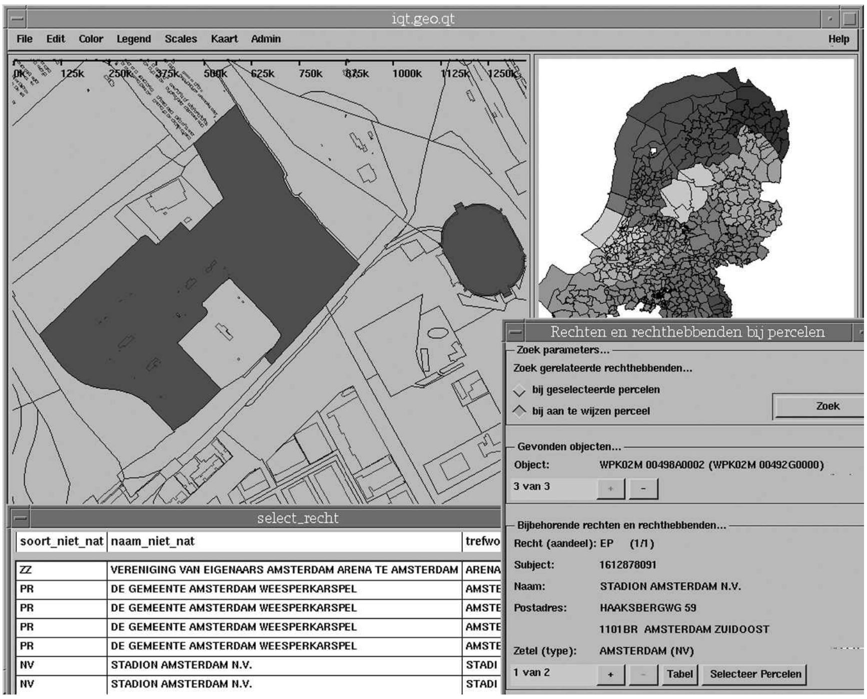

iqt.geo
es Kaart Admin
Zoek rechten en rechthebbende gerelateerd aan perceel
Zoek percelen gerelateerd aan rechthebbende
Zoek adres gerelateerd aan perceel
Zoek percelen op adres
Zoek belemmeringen gerelateerd aan perceel

Figure 13. The admin pull-down menu (Dutch interface).

the operations are: (1) Find rights and subjects related to parcels, (2) Find parcels related to subject, (3) Find addresses related to parcels, (4) Find parcels related to address, and (5) Find legal notifications related to parcels.

Filling in forms is a more traditional method for searching data. If this is combined with a map interface, then even better browsing possibilities are obtained. Options 2 and 4 can be used to add parcels to the active set (§5.1). The other options are used to easily obtain administrative information related to the parcels in the active set. Figure 14 shows in the lower right part, the form of option (1) Find rights and subjects related to parcel. In this case the active set contains the parcels of the


iqt.geo.qt
File Edit Color Legend Scales Kaart Admin
Help
0k 125k 250k 375k 500k 625k 750k 875k 1000k 1125k 1250k
Rechten en rechthebbenden bij percelen
select_recht
Zoek
+
-
Tabel
Selecteer Percelen

Zoek parameters...

Zoek gerelateerde rechthebbenden...

◇ bij geselecteerde percelen

◆ bij aan te wijzen perceel

Gevonden objecten...

Object:	WPK02M 00498A0002 (WPK02M 00492G0000)
3 van 3	

Bijbehorende rechten en rechthebbenden...

Recht (aandeel):	EP (1/1)
Subject:	1612878091
Naam:	STADION AMSTERDAM N.V.
Postadres:	HAAKSBERGWG 59
	1101BR AMSTERDAM ZUIDOOST
Zetel (type):	AMSTERDAM (NV)
1 van 2	

soort_niet_nat	naam_niet_nat	trefwo
ZZ	VERENIGING VAN EIGENAARS AMSTERDAM ARENA TE AMSTERDAM	ARENA
PR	DE GEMEENTE AMSTERDAM WEESPERKARSPEL	AMSTE
PR	DE GEMEENTE AMSTERDAM WEESPERKARSPEL	AMSTE
PR	DE GEMEENTE AMSTERDAM WEESPERKARSPEL	AMSTE
PR	DE GEMEENTE AMSTERDAM WEESPERKARSPEL	AMSTE
NV	STADION AMSTERDAM N.V.	STADI
NV	STADION AMSTERDAM N.V.	STADI

Figure 14. Searching for parcels by owner, Amsterdam Arena.

Amsterdam Arena, the soccer stadium of Ajax Amsterdam. Without this form, the user must transverse the following four tables/views: `parcel`, `object`, `right`, and finally `subject`. Also within the object table the user has to deal with the hierarchical structures related to part-of-parcels and apartment complexes (§3).

6. Practical use cases

In this section a number of practical use cases or 'applications' are described in more detail. These use cases illustrate the capabilities of the generic geographical query tool. However, these use cases also show the limitations of the generic geographical query tool and solutions 'outside' (on top of) the generic geographical query tool are presented. It should be noted that the presented use cases are relatively complex compared to the over one hundred different ad-hoc queries, which were posed to the generic geographical query tool. Earlier papers (IJsselstein and Kap 1995, van Oosterom and Maessen 1997) described a prototype version of the generic geographic query tool. The current version has been in production since August 1999 and is based on a large nationwide DBMS, which contains probably one of the largest vector data sets in the world.

The first application 'collecting statistical information on changes in the topographic map' (§6.1) uses spatial aggregates (municipalities), historic data and the spatial join. The next application, described in §6.2, 'quality improvement of the registration of legal notifications' integrates geometric and thematic data (legal notifications related to parcel), but also uses a spatial join between topologically structured parcels and linear pipe lines (data from a third party). As the relational DBMS cannot execute the spatial join with topologically structured data, this is implemented in the (geo)graphical user interface: the GIS frontend GEO++. The last application, the production of 'update files' for customers of the cadastral data, heavily depends on the spatio-temporal data model (§6.3).

Besides these example applications and many small *ad hoc* queries, the generic geographical query tool has been used for several other projects. A few will be mentioned here:

- Collecting statistics with respect to 'akte posten' (that is, parcels which have to be surveyed because of changes such as splitting or reorganizing).
- Deriving zip-code map from cadastral data, which aggregates geometric data (parcels to zip-code regions) based thematic data (object addresses).
- Finding potential parcels owned by farmers which may be used for land exchange (lots at a large distance from the farm).
- Finding parcels which may be merged because they have equal legal status (e.g. owner and so on).
- Finding all parcels of interest to the Ministry of Agriculture which have to be outside given built-up area polygons.
- Finding all parcels and their owners on which a protected monument is located.
- Deriving the type of house (free-standing house, corner house, middle house in a row, two under one roof house, apartments) by overlaying the topographic buildings (which are not classified in the way described above) and the cadastral map (Rengelink *et al.* 2000).
- Aggregating thematic information (e.g. average price) and visualizing the result on a geometric aggregation of the same level (e.g. municipality).

6.1. *Collecting statistical information on the changes in the large-scale topographic map*

The organization of the creation and maintenance of the large-scale topographic map in The Netherlands is on a regional level. In each region there is a local institution, which is responsible for the maintenance of the large-scale topographic map. These institutions are founded by several participants, who have a common interest in maintaining such a topographic map. Usually the participants are utility companies, water boards, municipalities, the Cadastre, etc. This case is applicable to the province of 'Zuid-Holland'. The institution maintains the map per municipality. For each municipality the map is updated on a yearly basis by a selected partner. The partner is paid by the number of mutations in a certain period. The following kinds of mutations are distinguished:

Deletion. A deletion of an element;
Semantic Mutation. A change in a text string and symbols;
Building. A new small building or changes to an existing building with the maximum of 8 coordinate points;
New hard topographic element. New visible topographic element with a maximum of 10 coordinate points;
Main building. New main building with a maximum of 10 coordinate points;
Big Building. Change or a new complex building;
'Soft' topographic, concentration. Mutations in so-called 'soft' topographic elements with a maximum of 10 coordinates, in case they can be measured in combination with other mutations;
'Soft' topographic, no concentration. Mutations in so called 'soft' topographic elements with a maximum of 10 coordinates, in case they cannot be measured in combination with other mutations.

In case a topographic element has more than 10 coordinates it counts as more than one element. The Cadastre maintains the large-scale topographic map in the LKI system. For such entity, like a topographic line, or text element, two time stamps are maintained. First, we have `tmin` which is the time of creation. Second, we have `tmax` which is the time of deletion (§3.1). If an entity is current, then `tmax=MAX_Time`. Based on these time stamps it is easy to find the changes in the topographic map in a certain period. Further, we need to know which topographic lines are located within which municipality. There are two tables `topographic_line` and `municipality` and their attributes are (for readability only the relevant attributes are shown):

```
create table topographic_line (
   object_id integer,        - the unique ID of the line
   line iline(50),           - the coordinates which define the position
   tmin integer,             - time of creation
   tmax integer);            - time of deletion (if current tmax=0)

create table municipality (
   code char(5),             - The code of the municipality
   pgon long polygon);       - The coordinates which define the boundary
                             of the municipality.
```

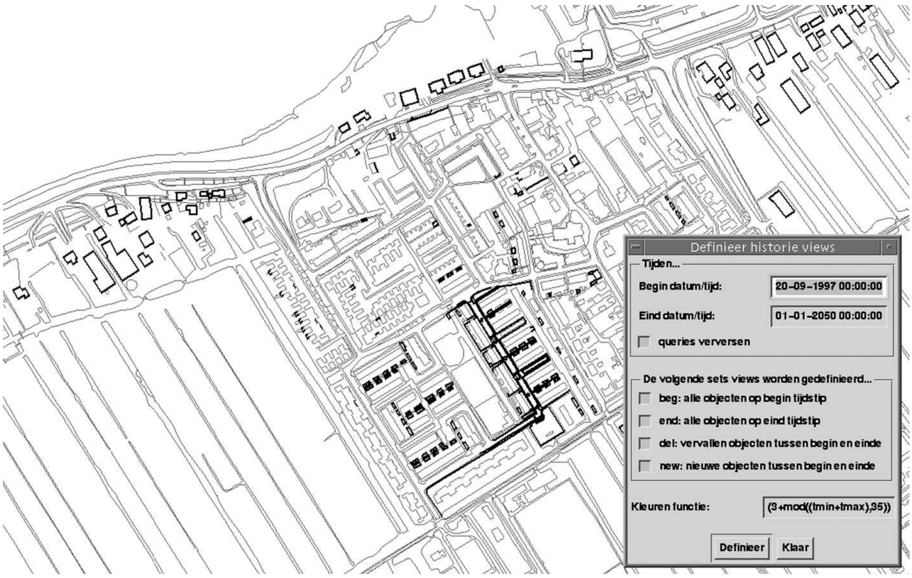
With the following SQL view definition we can determine which topographic lines have been deleted per municipality in the time frame from 30 June 1998 up to and including 30 June 1999 (for readability the SQL-commands have been simplified):

```
   create view deleted_topographic_line as
      select l.object_id, municip=m.code, l.line,
        month=period(tmax), num_points=numpoints(l.line)
      from topographic_line l, municipality m
      where inside (first_point(l.line), m.pgon)=1 and
        (l.tmax>lkidate2int(`30-06-1998 23:59:00')) and
        (l.tmax<=lkidate2int(`30-06-1999 23:59:00'))
```

If we also define SQL views for the new topographic lines and the current lines, then we can create maps where all lines that are changed in a certain time frame are highlighted, see figure 15. The bold lines are the lines that have changed. The following view counts the total number of deleted lines per municipality per month:

```
   create view deleted_per_municip_month as
      select municip, month, number=count(*)
      from deleted_topographic_line
      group by municip, month
```

These changes, as shown in the map of figure 15, are not equal to the mutuations as described earlier. Usually, a group of changes is equal to a mutation. The actual changes in the DBMS depend quite a lot on the manner in which an operator has edited the topographic map. The map with changes is merely used as a means to find and recognize the mutations. Further, a list of changes is produced from the DBMS, which can be check marked, to make sure none of the mutations have been skipped. The plots of maps with changes and the list form the appendix of the bill for the maintenance of the topographic map in a certain municipality.



Tijden...

Begin datum/tijd: 20-09-1997 00:00:00

Eind datum/tijd: 01-01-2050 00:00:00

queries verversen

De volgende sets views worden gedefinieerd...

beg: alle objecten op begin tijdstip

end: alle objecten op eind tijdstip

del: vervallen objecten tussen begin en einde

new: nieuwe objecten tussen begin en einde

Kleuren functie: (3+mod((tmin+tmax),35))

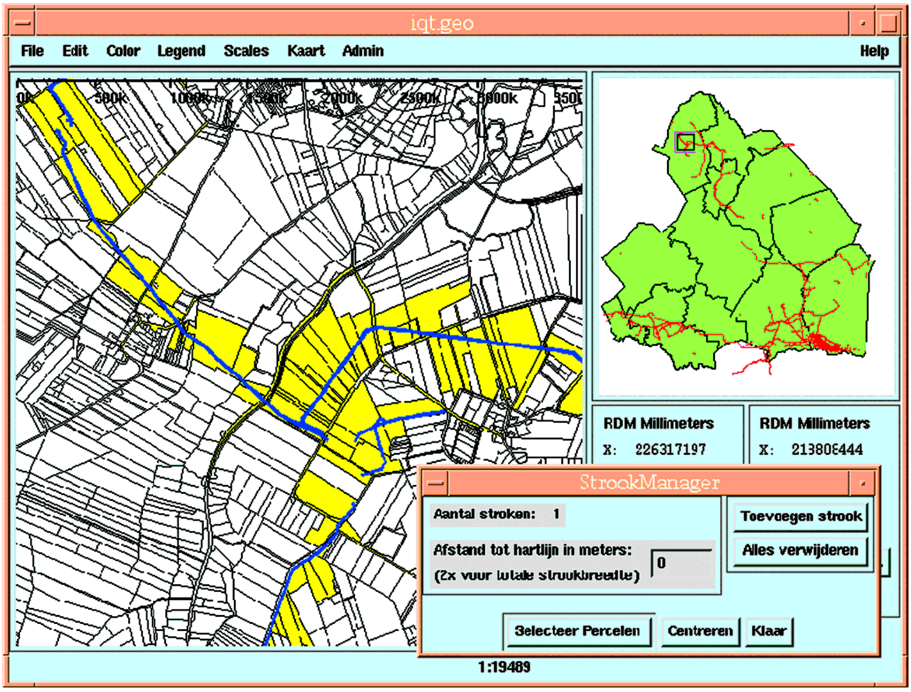
Figure 15. All changed objects.

6.2. *Quality improvement of the registration legal notifications*

In addition to the registration of the basic rights, such as ownership, related to parcels (cadastral objects), the Cadastre also registers many types of legal notifications. These legal notifications restrict the use of a parcel by the owner. An important type of legal notification is related to pipelines, usually below the surface (figure 16).

In order to protect these pipelines, the parcels crossed by a pipeline get a legal notification of the proper type. This is only done in the administrative part of the Cadastral registration. It has to be done in an official manner described by law: a deed has to be drawn up by the notary and submitted to the Cadastre for registration. The pipelines are not available in the geographic part of the cadastral registration. Several types of problems became more and more visible in the last couple of years:

1. Whenever a parcel is split, all new parts inherit the legal notification. This is because the pipelines themselves are not registered at the Cadastre, so it is impossible to determine which new parts are crossed by the pipeline. In order to be safe all new parts inherit the legal notification. This means that too many parcels have these legal notifications, which implies unnecessary costs for the owner of the pipeline.
2. It is easy to forget a few parcels when tying to register the complete trace of a pipeline without the exact geometry of the parcels and the pipeline. This results in parcels without a legal notification. This is a dangerous (legal) situation as the pipeline crosses these parcels, but without the proper status.
3. The registration of basic rights always stores who (which subject) has a certain type of right on which parcel (object). In the early registration of legal notifications it was not registered who caused the specified type of legal notification. Only the fact that there were one or more (types of) legal notifications was associated with the parcel. This makes the maintenance of this registration difficult. Imagine that for some reason a pipeline does not need the legal protection anymore, then it is dangerous to remove all the legal notifications



RDM Millimeters X: 226317197

RDM Millimeters X: 213808444

Aantal stroken: 1

Afstand tot hartlijn in meters: 0
(2x voor totale strookbreedte)

1:19489

Figure 16. Pipelines of the NAM and the parcels they cross.

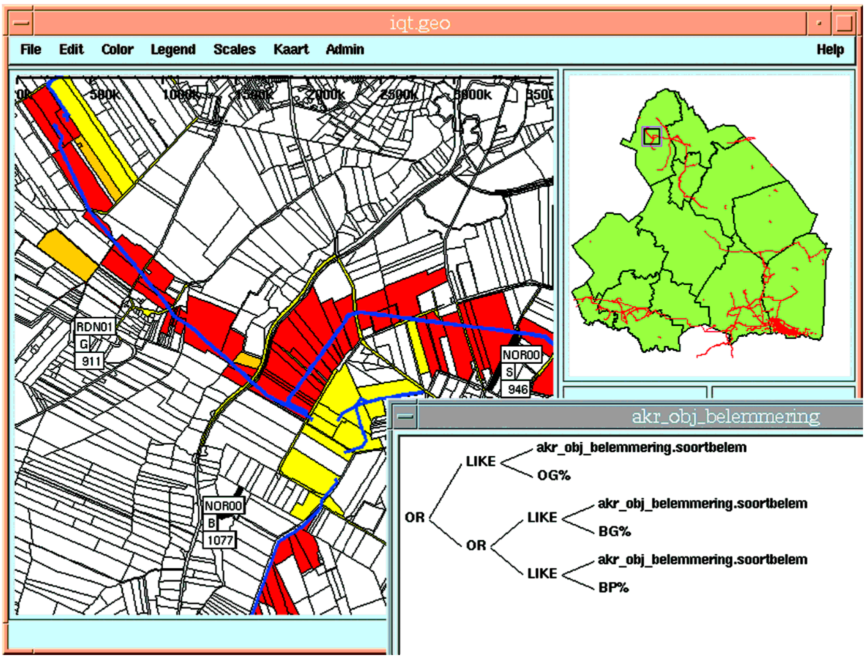
because they are 'anonymous'. It could be the case that another utility company has a pipeline crossing the same parcel(s).

To solve the problems mentioned above, it was decided to start a quality improvement process. Going back to all the paper deeds is just too much work, so the generic geographical query tool was used to select the parcels, which have these 'anonymous' legal notifications (these are of types BP, BG or OG, which stand in Dutch for respectively 'Belemmering privaatrecht', 'BP-gedoogplicht', and 'Opstal olie/gas'). Using the list of selected parcels the paper deeds are now physically retrieved and the legal notification is associated with the proper organization ('owner' of the pipeline) and also the type of legal notification is changed to OL or BZ. 'OL' stands for a legal notification described as (in Dutch) 'Recht van opstal m.b.t. het leggen en houden van leidingen in, op of boven een onr. zaak' (right to construct and maintain pipelines/cables in, on or above a property based on private law). 'BZ' stands for a legal notification described as (in Dutch) 'Zakelijk recht als bedoeld in art.5, lid 3, onder b, van de Belemmeringenwet Privaatrecht' (real right which limits private property right in general interest based on public law). This solves the third problem mentioned above. However, it does not solve the first two problems.

A pilot project was started with an important owner of pipelines in The Netherlands; the NAM, Nederlandse Aardolie Maatschappij, a company equally owned by Shell and Esso (Exxon). The NAM delivered a digital version of their pipelines to the Cadastre, which were then entered into the query tool DBMS and confronted with the parcels (figure 16). This was not a simple query in the DBMS, because the geographical data model of the Cadastre is based on topology. Within a relational DBMS an overlap or cross-operation based on parcels modelled with topology is impossible. Therefore this operation was implemented in the interface (front-end) part of the generic geographical query tool; see the inset window of figure 16.

After the quality improvement of the legal notifications, the parcels with a legal notification of OL or BZ associated with the NAM can be displayed on top of the parcels crossed by a pipeline of the NAM. A few things can then be observed. First, not all parcels crossed by a pipeline have the legal notification. This can be correct in case the parcel is owned by the government; e.g. roads, in this situation a 'permit' is sufficient, but this is not registered at the Cadastre. However, there are several parcels without a legal notification and these are clearly not road parcels, which are crossed by a pipeline. This could be an old pipeline and has to be checked by the NAM. Second, there are parcels with a NAM legal notification which are not crossed by a pipeline. Again, this has to be checked by the NAM. It could be correct; e.g. the parcel might contain a NAM access road or some type of NAM location.

Finally, it is interesting to check if all 'anonymous' legal notifications of type BP, BG or OG are resolved by the quality improvement process. Therefore all legal notifications of these types are selected and displayed with a label, indicating the parcel number, in figure 17. In the project of quality improvement of legal notifications and the pilot project with the NAM, the generic geographical query tool turned out to be useful. As described, the generic geographical query tool is used during several stages: before the process to inspect the situation and select the 'problem' parcels. During quality improvement to find the parcels crossed by pipelines, after quality improvement to check if the result is correct; e.g. there are no more anonymous legal notifications. One final remark: there are many more types of legal notifications than the ones mentioned in this section. These were also quality improved, but not discussed in this section.



OR
- LIKE
  - akr_obj_belemmering.soortbelem
  - OG%
- OR
  - LIKE
    - akr_obj_belemmering.soortbelem
    - BG%
  - LIKE
    - akr_obj_belemmering.soortbelem
    - BP%

Figure 17. Parcels with legal notification of type BP, BG or OG are marked with a label.

6.3. *Production of update files*

After an initial full delivery of the data set, the customers receive periodic update files, which contain the differences with respect to the previous delivery (Lemmen and Keizer 1993). The time interval for a typical update file starts at the begin point in time `t_beg` and stops at the end point in time `t_end`. The update files are composed of two parts: `OLD` (*in Dutch* `WAS`): deleted objects and old versions of changed objects; `NEW` (*in Dutch* `WORDT`): new objects and new versions of changed objects.

Besides selecting these data from the database (using SQL queries with time stamps), the production of update files at least has to include reformatting the database output in the national data transfer standard (NEN-1878 1993) or some other desired data transfer format. The object changes might occur in attributes, such as topological references, which the customer does not receive. These invisible changes can be either filtered out (*signif-changes*) or may be left in the update file (*all-changes*). There are two ways of interpreting the begin (`t_beg`) and end (`t_end`) time related to an update file: as a complete time *interval* or as two individual *points* (*instants*) in time. In the second case, the customer is not interested in temporary versions of the objects between the two points in time `t_beg` and `t_end`. This results in four different types of update files:

1. *interval_all_changes:* all changes over time interval (`t_beg`, `t_end`] *including* `t_end`*, with delivery of all temporary object versions.*

```
/* deleted/updated objects */
select * from line 1 where
  t_beg<1.tmax and 1.tmax<=t_end;

/* new/updated objects */
select * from line 1 where
  t_beg < 1.tmin and 1.tmin <= t_end;
```

In case an object is updated two times, two versions of old objects (`OLD`: `x,t1` and `x,t2`) and two versions of new objects (`NEW`: `x,t2` and `x,MAX_Time`) will be included in the update file (figure 18).

2. *points_all_changes:* only changes comparing the two points in time `t_beg` and `t_end`, excluding all temporary versions, have to be delivered. This means that the

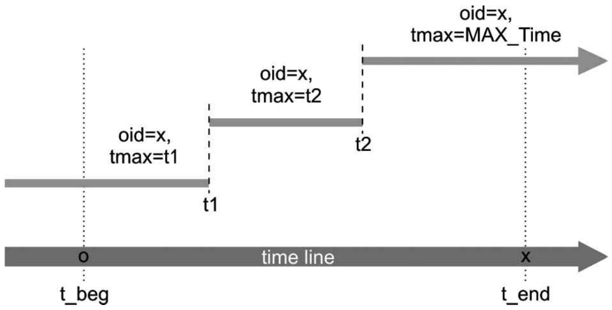


Figure 18. Updating an object twice in the specified time interval (`t_beg`, `t_end`].

object versions have to overlap in time either t_beg (deleted/updated objects) or t_end (new/updated objects).

```
/* deleted/updated objects */
select * from line l where
  t_beg<l.tmax and l.tmax<=t_end
                and l.tmin<=t_beg;

/* new/updated objects */
select * from line l where
  t_beg<l.tmin and l.tmin<=t_end
                and t_end<l.tmax;
```

In the example of figure 18 this will produce only one version of the old object (OLD: x,t1) and only one version of the new object (NEW: x,MAX_Time). This approach will also filter out the temporary versions when an object is really deleted (but also changes one or more times in the time interval) or when an object is really new (figure 19).

3. *interval_signif_changes:* all changes over time interval (t_beg, t_end] with respect to the delivered attributes (A1, A2, ..., An) are included in the update file. Ai can be a geometric data type. As the data has to be reformatted anyhow by the front-end application in order to produce the standard transfer format NEN-1878 (1993), it is easy to include the filter for significant changes in this application (especially if the input data is sorted on oid):

```
select l.oid, l.tmax, l.A1, l.A2, . . .
from line l
where  /* deleted/updated */
  t_beg<l.tmax and l.tmax<=t_end
  or  /* new/updated */
  t_beg<l.tmin and l.tmin<=t_end
sort by l.oid , l.tmax;
```

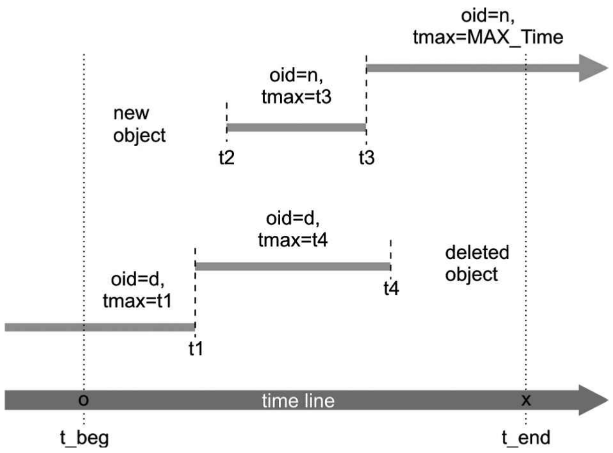


Figure 19. Removing temporary version of new and deleted objects in the specified time interval (t_beg, t_end].

4. *points_signif_changes:* all changes comparing the two points in time `t_beg` and `t_end` with respect to the delivered attributes (`A1`, `A2`, ..., `An`) are included in the update file. It is now not true anymore that the reported object versions have to overlap in time either `t_beg` (deleted/updated objects) or `t_end` (new/updated objects), because they can be related to insignificant changes. It could be that a significant change occurs somewhere in the middle (figure 20).

In general, many insignificant versions of an object, with respect to the attributes for a customer, may precede and/or follow a version with a significant change. These should be temporarily glued together with versions related to insignificant changes; not in the database itself. This can be included easily in the application program in two steps: first 'glue', then filter out glued object versions, which do not overlap the two points in time: `t_beg` and `t_end`.

7. Towards mainstream Geo-ICT solutions

Although both GEO++ (Professional Geo Systems (PGS) 1996, Vijlbrief and van Oosterom 1992) and Ingres (ASK-OpenIngres 1994, van Oosterom 1997) with OME/SOL are commercially available software packages, they hardly qualify as mainstream Geo-ICT solutions. Both systems have a background in research and were even refined during the different projects at the Dutch Cadastre. The RDBMS-based cadastral data production system (LKI) has been in production since 1997 and the cadastral query tool system has been in production since 1999. The first prototype of the latter system dates back to 1995. In the meantime the mainstream Geo-ICT (both Geo-DBMS and GIS packages) have also developed towards the integrated architecture similar to GEO++ and Ingres OME/SOL. These developments are in a large part due to the work of the OpenGIS Consortium (Buehler and McKee 1998, Open GIS Consortium, Inc. 1999) and hopefully also in (a small) part due to the papers mentioned earlier by the authors and their colleagues.

Due to a small marketshare (especially GEO++) the current solution cannot be characterized as mainstream Geo-ICT solution. Therefore, the Dutch Cadastre is looking for alternative implementations of the query tool (to start with). Currently, the Dutch Cadastre is developing a prototype based on Oracle (Oracle Inc. 2001, Hebert and Murray 1999) and MapInfo (MapInfo Professional v6.5 2001). The TU Delft performed some experiments with two other recent versions of GIS packages:

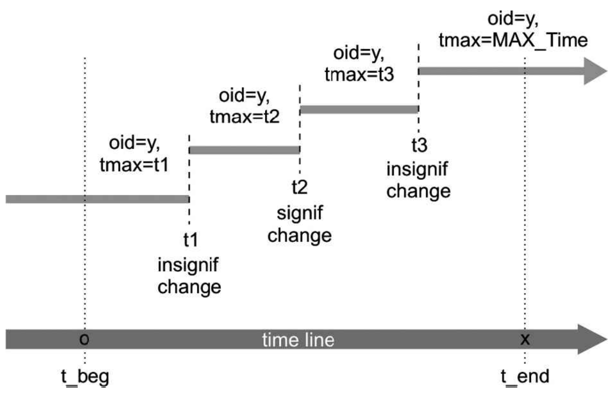
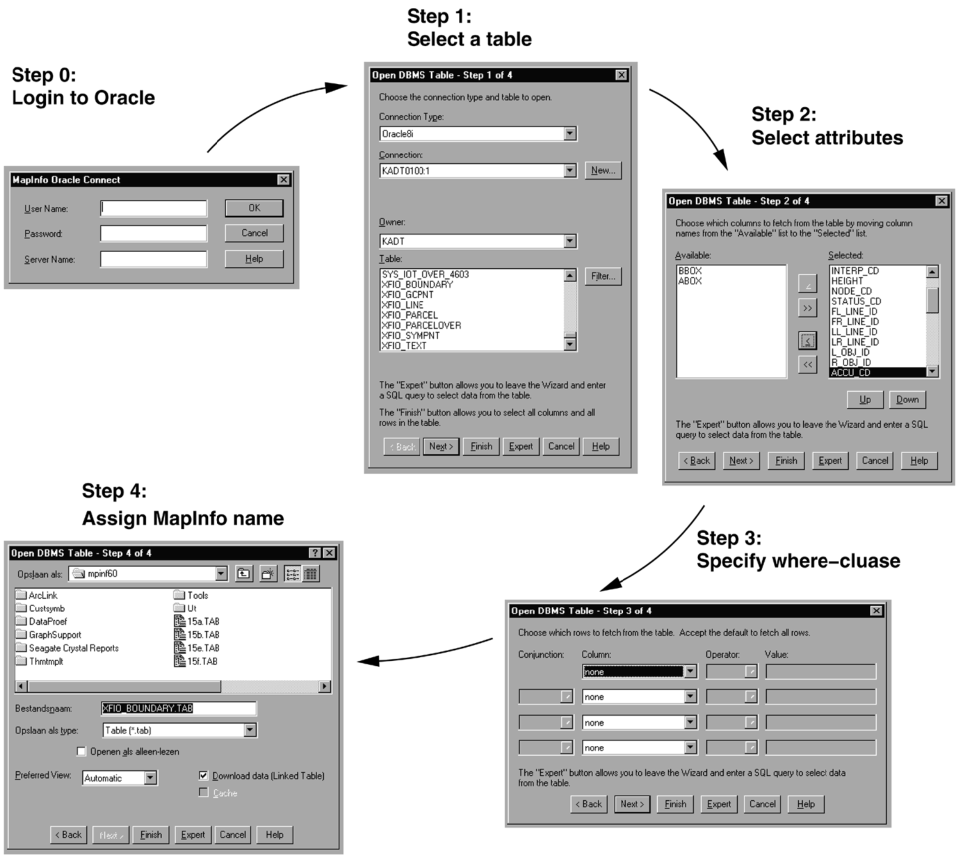


Figure 20. Detecting significant changes in temporary version of an object in the specified time interval (`t_beg`, `t_end`).

from Intergraph (Intergraph Inc. 2001) and from Bentley (Bentley MicroStation 2001). The MapInfo and the Intergraph software are official production releases, the Bentley software has beta status at this moment. In the remainder of this section our experiences with MapInfo will be described. Although every package has his own advantages and disadvantages, the results with respect to querying the data in Oracle spatial were more or less similar.

Figure 21 shows an overview of the steps to access Oracle spatial data in MapInfo. Of course, this starts with logging on to the Oracle database server (step 0). The list of available tables is then presented and the user can select a table (step 1). Note that this is data model driven, because when the DBMS is populated with more tables, these will (automatically) be shown to the user. The next two steps are also based on the data model driven principle. First in step 2, the user can indicate which attributes must be visible in MapInfo (do not select more than 1 spatial attribute). Then in step 3 it is possible to specify selection criteria on one or more (administrative) attributes, that is the SQL where-clause. Finally, in step 4 the user has to specify the (file) name associated with this query. If the 'download' checkbox is pressed, then the selected data will be actually stored in the MapInfo files. Otherwise, a 'live link' is used and the MapInfo file only contains some 'meta data'; e.g. visualization



Choose the connection type and table to open.

Connection Type: Oracle8i

Connection: KADT0100:1

Owner: KADT

Table: SYS_IOT_OVER_4603, XFIO_BOUNDARY, XFIO_GCPNT, XFIO_LINE, XFIO_PARCEL, XFIO_PARCELOVER, XFIO_SYMPNT, XFIO_TEXT

The "Expert" button allows you to leave the Wizard and enter a SQL query to select data from the table.

The "Finish" button allows you to select all columns and all rows in the table.

Choose which columns to fetch from the table by moving column names from the "Available" list to the "Selected" list.

Available: BBOX, ABOX

Selected: INTERP_CD, HEIGHT, NODE_CD, STATUS_CD, FL_LINE_ID, FR_LINE_ID, LL_LINE_ID, LR_LINE_ID, L_OBJ_ID, R_OBJ_ID, ACCU_CD

The "Expert" button allows you to leave the Wizard and enter a SQL query to select data from the table.

Choose which rows to fetch from the table. Accept the default to fetch all rows.

Conjunction:	Column:	Operator:	Value:
	none		
	none		
	none		
	none		

The "Expert" button allows you to leave the Wizard and enter a SQL query to select data from the table.

Opslaan als: mpinf60

ArcLink, Custsymb, DataProef, GraphSupport, Seagate Crystal Reports, Thmtmplt, Tools, Ut, 15a.TAB, 15b.TAB, 15e.TAB, 15f.TAB

Bestandsnaam: XFIO_BOUNDARY.TAB

Opslaan als type: Table (*.tab)

☐ Openen als alleen-lezen

Preferred View: Automatic

☑ Download data (Linked Table)

Figure 21. The steps to access an Oracle spatial table with MapInfo.

parameters. After selecting data from a number of different tables, the map shown in figure 22 is obtained.

Some problems remain and have to be solved in the near future. A short list of points, which require some further investigations are:

- The geometry model in Oracle Spatial is still not according to the OpenGIS standard for simple features, types and functions (Open GIS Consortium, Inc. 1999), though internally the requested types are available and grouped in a general geometry type SDO_GEOMETRY. Not being based on the OpenGIS standard makes integration with other software more difficult.
- Tables with multiple spatial attributes must be handled with care; e.g. a parcel with a polygon, reference point and a bounding box. These attributes cannot be obtained and visualized in MapInfo with one query.
- The parcels are based on a topological model. However, both Oracle and MapInfo do not support topology at this moment. This is also true for the OpenGIS simple feature specification. There is no implementation specification for topology or complex features. At the abstract specification level topology is included in the new ISO TC 211 geometry model (ISO TC 211/WG 2 1999), which is also the basis of the future OpenGIS geometry model. It may be difficult (or impossible) however to make a relational DBMS implementation of this specification due to the impossibility to navigate within the RDBMS. A future solution might be (a mix with) object DBMS technology.
- At the moment we do not use the temporal aspect of the data model. It should be possible to implement access to the temporal aspect in a similar way as in

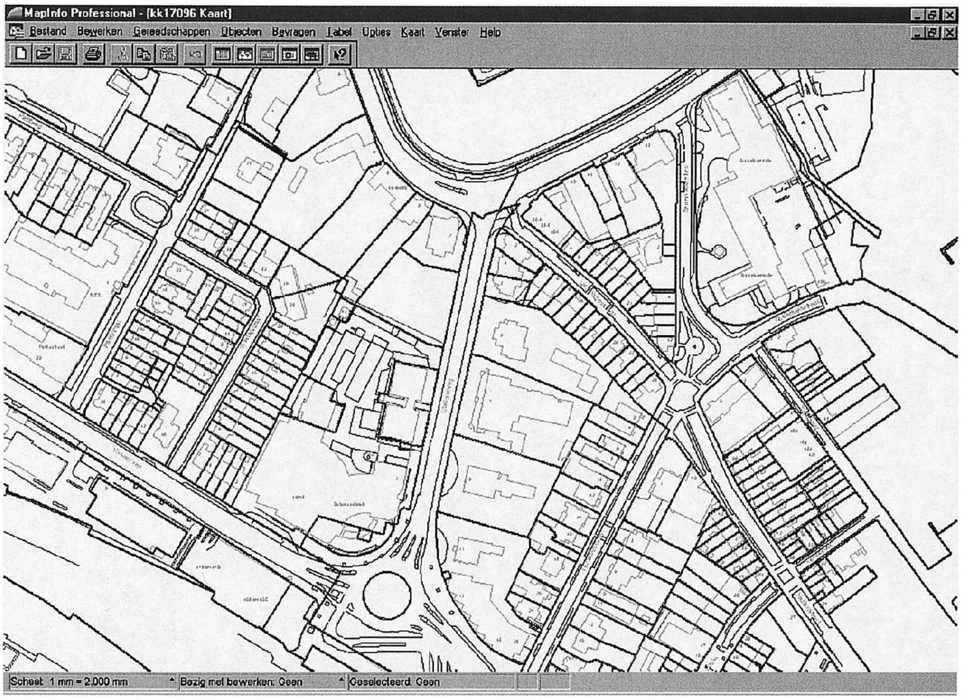

MapInfo Professional - [kk17096 Kaart]
Bestand Bewerken Gereedschappen Objecten Bevragen Label Opties Kaart Venster Help
Schaal: 1 mm = 2.000 mm
Bezig met bewerken: Geen
Geselecteerd: Geen

Figure 22. The cadastral data as selected from Oracle spatial shown in MapInfo.

the GEO++/Ingres environment. That is based on sets of views: snapshot views at specified moment in time and delta views showing the differences over a certain specified period.

- Although Oracle Spatial and MapInfo both support circular arcs, this does not work properly when selecting data from Oracle spatial and showing them in MapInfo. This can probably be solved with a minor patch (of MapInfo).
- In order to be a true *generic* query tool environment, the system should not only be data model driven (using table definitions and data types from the DBMS), but it should also be DBMS function driven. When a new function is added to an abstract data type (ADT or object class method definition), this should also be available to the front-end applications. So the DBMS has to have some system catalogues describing the available functions dynamically and the frontends must use this information and be flexible enough to present these DBMS functions to the user when formulating a query. This will probably require more developments on both the DBMS (backend) and the GIS (frontend) side.

8. Conclusions

In this paper, the design, implementation and application of a generic geographical query tool has been presented. Though one might be tempted to think that it is impossible to develop such a system, we think that the described generic geographical query tool has demonstrated itse usefulness in many unanticipated use cases (§6). Of course the users also have new functional wishes after using the system. The first extension is specifying his/her own views, based on joins, and having attributes from multiple tables in the result. The second extension is more import and export functionality. The third extension is to have more up to date data in the query tool DBMS (now updated four times per year). Therefore, instead of loading full data sets four times per year to the query tool DBMS, in the future, the data will have to be replicated more frequently from the geometric and administrative 'production' DBMSs. Instead of using full data set copies, this can be done more efficiently by only transferring the changes. These mutation files are standard products in the source systems and can be obtained every month.

Further, external users should be able to query the data over the Internet. Limited functionality of the generic geographical query tool is implemented in Java (Arnold and Gosling 1996) as a prototype for the digital Geoshop (van den Berg *et al.* 1997). This must be coupled to the NCGI (National Clearinghouse Geoinformation) (de Gunst and van Oosterom 1997, Jacobi and Lind 1997, van de Kieft and Kok 1997) and based on OpenGIS standards (Buehler and McKee 1998), specifically the standards currently being developed in the web-mapping testbed. More research is needed in the area of generic information systems (query tools) in the context of distributed setting, both with respect to the data and to the processing (functions/operands).

In this paper it was also demonstrated that the concept of views based on relational joins forms a powerful mechanism to integrate data models from different sources. If one of these data models stores geometric information, then it is easy to create thematic maps based on the values of the attributes stored in the other model. Aggregation in a relational DBMS environment is a step closer to deriving new map information and this can also be implemented via the concept of a relational view.

Searching and querying the DBMS can be done with the help of regular search forms, by way of the map, using the position of objects or a combination of them. Visualizing historic data requires specific techniques such as presenting specific moments in time or displaying the changes over a certain period of time (on top of a specific moment in time display). As demonstrated, this can be realized very well using views with time selection in the where-clause predicate. Impossible to implement using views are the spatial aggregates, traversing hierarchical structures and traversing topology structures. Note that these are general flaws of the relational model, which are all solved by the object-oriented model.

The generic geographical query tool turned out to be a useful tool within the Cadastral organization. Further enhancements and migration towards mainstream Geo-ICT solutions will improve the usefulness and effectiveness of the generic geographic query tool even more. The first impression of applying Oracle and MapInfo is promising, however there are some open issues to be solved, e.g. handling topology, circular arcs, temporal data and the support of the OpenGIS simple feature standard.

Acknowledgments

We would like to thank the persons involved in the implementation of the generic geographical query tool within the Dutch Cadastre. In total 30 to 40 persons have been involved with this project. These are just too many to mention. However, we will make a few exceptions. First, Herman Welleweerd, the project leader, who had the difficult task to implement and introduce a new system in a complex organization. Second, Bertus Padberg, the DBA who was responsible for loading the huge data sets in the DBMS. Finally, Joep Mathijssen, who developed large parts of the cadastral add-ons and also introduced the *active set* concept; see §5. Axel Smits improved many of the original figures. We owe our thanks to Elfriede Fendel, Jantien Stoter, Paul Strooper, Edward Verbree, and the two anonymous IJGIS reviewers for their valuable remarks.

References

Abraham, T., and Roddick, J. F., 1999, Survey of spatio-temporal databases. *GeoInformatica*, **3**, 61–99.

Al-Taha, K. K., Snodgrass, R. T., and Soo, M. D., 1994, Bibliography on spatiotemporal databases. *International Journal of Geographical Information Systems*, **8**, 95–103.

Arcieri, F., Cammino, C., Nardelli, E., Talamo, M., and Venza, A., 1999, The Italian cadastral information system: a real-life spatio-temporal DBMS. In *STDBM '99, workshop on Spatio-Temporal Database Management, Edinburgh, Scotland*, edited by M. H. Böhlen, C. S. Jensen and M. O. Scholl, number 1678 in *Lecture Notes in Computer Science (LNCS)* (Berlin: Springer-Verlag), pp. 79–99.

Armenakis, C., 1996, Mapping of spatio-temporal data in an active cartographic environment. *Geomatica*, **50**, 401–413.

Arnold, K., and Gosling, J., 1996, *The Java Programming Language* (Reading, Massachusetts: Addison-Wesley Publishing Company, Inc.).

ASK-OpenIngres, 1994, INGRES/Object Management Extention User's Guide, Release 6.5, Technical Report.

Baumgart, B. G., 1975, A polyhedron representation for computer vision. *National Computer Conference*, 589–596.

Bentley MicroStation, 2001, *GeoGraphics ISpatial edition (J 7.2.x)*, URL: http://www.bentley.com/products/geographics/faq.htm.

Buehler, K., and McKee, L., 1998, The OpenGIS Guide—Introduction to interoperable geoprocessing. Technical Report Third edition, The Open GIS Consortium, Inc.

Chen, J., and Jiang, J., 1998, An event-based approach to spatio-temporal data modelling in land subdivision systems. *GeoInformatica*, **2**, 387–402.

Cheng, T., and Molenaar, M., 1998, A process-oriented spatio-temporal data model to support physical environmental modelling. In *Proceedings of the 8th International Symposium on Spatial Data Handling*, edited by T. K. Poker and N. Chrisman (Burnaby: International Geographical Union), pp. 418–430.

CHOROCHRONOS: A Research Network for Spatiotemporal Database Systems, 1996–2000, URL: http://www.dbnet.ece.ntua.gr/~choros

Claramunt, C., and Theèriault, M., 1996, Towards semantics for modelling spatio-temporal processes within GIS. In *Advances on GIS II: Proceedings of the 7th International Symposium on Spatial Data Handling*, edited by M. J. Kraak and M. Molenaar (London: Taylor & Francis), pp. 2.27–2.43.

Comer, D., 1979, The ubiquitous B-Tree. *ACM Computing Surveys*, **11**, 121–137.

de Gunst, M., and van Oosterom, P., 1997, Network computers at the Dutch cadastre. In *Proceedings of the 46th Photogrammetric Week, 1997*, pp. 000–000.

Easterfield, M. E., Newell, R. G., and Theriault, D. G., 1990, Version management in GIS—applications and techniques. In *Proceedings of the European Geographical Information*, **90**, pp. 288–297.

Erwig, M., Gueting, R. H., Schneider, M., and Vazirgiannis, M., 1999, Spatio-temporal data types: an approach to modelling and querying moving objects in databases. *GeoInformatica*, **3**, 269–296.

Frank, A. U., 1994, Qualitative temporal reasoning in GIS—ordered time scales. In *Proceedings of the 6th SDH*, pp. 410–430.

Gold, C. M., 1996, An event-driven approach to spatio-temporal mapping. *Geomatica*, **50**, 415–424.

Guttman, A., 1984, R-trees: A dynamic index structure for spatial searching. *ACM SIGMOD*, **13**, 47–57.

Hebert, J., and Murray, C., 1999, *Oracle Spatial User's Guide and Reference* (Redwood City, CA: Oracle Corporation), Part No. A77132-01.

Hornsby, K., and Egenhofer, M. J., 1998, Identity-based change operations for composite objects. In *Proceedings of the 8th International Symposium on Spatial Data Handling*, edited by T. K. Poiker and N. Chrisman (Burnaby: International Geographical Union), pp. 202–213.

IBM, 2000, *IBM DB2 Spatial Extender User's Guide and Reference*. Special web release edition.

IJsselstein, J. A., and Kap, A. P., 1995, Het kadastraal perceel: een stevig fundament!, *Nederlands Geodetisch Tijdschrift Geodesia*, **37**, 343–349 (in Dutch).

Informix, 2000, *Informix Spatial DataBlade Module User's Guide*. Part no. 000-6868.

Intergraph, 2001, Geomedia Professional 4.0. URL: http://www.intergraph.com/gis/gmpro/

ISO TC 211/WG 2, 1999, Geographic information—Spatial schema. Technical Report second draft of ISO 19107 (15046-7), International Organization for Standardization.

Jacobi, O., and Lind, M., 1997, Metadata: From European standard to user service. In *Proceedings of Third Joint European Conference and Exhibition on Geographical Information (JEC-GI'97)*, **2**, pp. 1155–1164.

Kollios, G., Gunopulos, D., and Tsotras, V. J., 1999, Nearest neighbor queries in a mobile environment. In *STDBM'99, Workshop on Spatio-Temporal Database Management*, edited by M. H. Böhlen, C. S., Jensen and M. O. Scholl, number 1678 in *Lecture Notes in Computer Science (LNCS)* (Berlin: Springer-Verlag), pp. 119–134.

Kraak, M. J., and MacEachren, A. M., 1994, Visualization of the temporal component of spatial data. In *Advances in GIS Proceedings of the 6th International Symposium on Spatial Data Handling*, edited by T. Waugh and R. Healey (London: Taylor & Francis), pp. 391–409.

Langran, G., 1992, *Time in Geographic Information Systems* (London: Taylor & Francis).

Lemmen, C. H. J., and Keizer, B., 1993, Levering van mutaties uit de LKI-gegevensbank. *Geodesia*, **35**, 265–269 (in Dutch).

Lemmen, C. H., and van Oosterom, P. J., 1995, Efficient and automatic production of periodic updates of cadastral maps. In *Proceedings of JEC'95, Joint-European Conference and Exhibition on Geographical Information, The Hague, The Netherlands*, pp. 137–142.

MAPINFO INC., 2001, MapInfo Professional v6.5, URL: http://dynamo.mapinfo.com/products/index.cfm.

NASCIMENTO, M. A., SILVA, J. R. O., and THEODORIDIS, Y., 1999, Evaluation of access structures for discretely moving points, In *STDBM'99, workshop on Spatio-Temporal Database Management*, edited by M. H. Böhlen, C. S. Jensen and M. O. Scholl, number 1678 in *Lecture Notes in Computer Science (LNCS)* (Berlin: Springer-Verlag), pp. 171–188.

NEN-1878, 1993, Automatische gegevensverwerking—Uitwisselingsformaat voor gegevens over de aan het aardoppervlak gerelateerde ruimtelijke objecten, Technical report, Nederlands Normalisatie-instituut (in Dutch).

OPEN GIS CONSORTIUM INC., 1999, OpenGIS simple features specification for SQL. Technical Report Revision 1.1, OGC.

ORACLE INC., 2001, Oracle 9i Spatial, URL: http://otn.oracle.com/products/spatial/

PEQUET, D. J., and WENTZ, E., 1994, An approach for time-based analysis of spatio-temporal data. In *Advances in GIS: Proceedings of the 6th International Symposium on Spatial Data Handling*, edited by T. Waugh and R. Healey (London: Taylor & Francis), pp. 489–504.

PROFESSIONAL GEO SYSTEMS (PGS), 1996, The GEO++ system, version 2.80, Reference manual, Technial report.

RAAFAT, H., YANG, Z., and GAUTHIER, D., 1994, Relational spatial topologies for historical geographical information. *International Journal of Geographical Information Systems*, **8**, 163–173.

RELLY, L., KUCKELBERG, A., and SCHEK, H., 1999, A framework of a generic index for spatio-temporal data in CONCERT. In *STDBM'99, Workshop on Spatio-Temporal Database Management*, edited by M. H. Böhlen, C. S. Jensen and M. O. Scholl, number 1678 in *Lecture Notes in Computer Science* (Berlin: Springer-Verlag), pp. 135–151.

RENGELINK, M., VAN OOSTEROM, P., QUAK, W., and VERBREE, E., 2000, Automatic derivation and classification of houses on a cadastral map. In *Proceedings of UDMS 2000: 22nd Urban and Regional Data Management Symposium: Urban and Rural Data Management Common Problems—Common Solutions/Seminar Land Markets and Land Consolidation in Central Europe, Delft, The Netherlands*, edited by E. Fendel, pp. III.33–III.42.

SNODGRASS, R. T., AHN, I., and ARIAV, G., 1994, Tsq12 language specification. *SIGMOD Record*, **23**, 65–86.

SPÈRY, L., CLARAMUNT, C., and LIBOUREL, T., 2001, A spatio-temporal model for the manipulation of lineage metadata. *GeoInformatica*, **5**, 51–70.

STONEBRAKER, M., and ROWE, L. A., 1986, The design of Postgres. *ACM SIGMOD*, **15**, 340–355.

TRYFONA, N., and JENSEN, C. S., 1999, Conceptual data modelling for spatiotemporal applications. *GeoInformatica*, **3**, 245–268.

VAN DE KIEFT, I. A., and KOK, B., 1997, The development of a geo metadata service for The Netherlands. In *Proceedings of the JEC-'97, Joint European Conference and Exhibition on Geographical Information* (Utrecht: EGIS), pp. 1165–1176.

VAN DEN BERG, C., TUIJNMAN, F., VIJLBRIEF, T., MEIJER, C., UITERMARK, H., and VAN OOSTEROM, P., 1997, Multi-server internet gis: Standardization and practical experiences. In *Proceedings of the International Conference and Workshop on Interoperating Geographic Information Systems, Santa Barbara, California, USA, interop'97* (Boston: Kluwer Academic Publishers), pp. 365–378.

VAN OOSTEROM, P., 1997, Maintaining consistent topology including historical data in a large spatial database. In *Proceedings of Auto-Carto 13* (Bethesda: American Congress on Surveying and Mapping), pp. 327–336.

VAN OOSTEROM, P., and LEMMEN, C., 2001, Spatial data management on a very large cadastral database. *Computers, Environment and Urban Systems*, **25**, 509–528.

VAN OOSTEROM, P., and MAESSEN, B., 1997, Geographic query tool. In *Proceedings of JEC-GI'97 Joint European Conference and Exhibition on Geographical Information* (Utrecht: EGIS), pp. 177–186.

VAN OOSTEROM, P., and VIJLBRIEF, T., 1996, The spatial location code. In *Advances in GIS II: International Symposium on Spatial Data Handling*, edited by M. J. Kraak and M. Molenaar (London: Taylor & Francis), pp. 3B.1–3B.17.

VIJLBRIEF, T., and VAN OOSTEROM, P., 1992, The GEO++ system: an extensible GIS. In

Proceedings of the 5th International Symposium on Spatial Data Handling, edited by M. J. Kraak and M. Molenaar (London: Taylor & Francis), pp. 40–50.

Voigtmann, A., Becker, L., and Hinrichs, K. H., 1996, Temporal extensions for an object-orientated geodata model. In *Advances in GIS II: Proceedings of the 7th International Symposium on Spatial Data Handling*, edited by M. J. Kraak and M. Molenaar (London: Taylor & Francis), pp. 11A.25–11A.41.

Worboys, M. F., 1994, Unifying the spatial and temporal components of geographical information. In *Advances in GIS: Proceedings of the 6th Symposium on Spatial Data Handling*, edited by T. Waugh and R. Healey (London: Taylor & Francis), pp. 505–517.