SPATIAL, THEMATIC, AND TEMPORAL VIEWS

Peter van Oosterom [(1)], Bart Maessen [(2)], Wilko Quak [(1)]

(1) Department of GIS Technology, Faculty of Civil Engineering and Geosciences, Delft University of Technology, The Netherlands. oosterom@geo.tudelft.nl, quak@geo.tudelft.nl
(2) Cadastre Netherlands P.O. Box 9046, 7300 GH Apeldoorn, The Netherlands. maessen@kadaster.nl

ABSTRACT

This paper explores the use of database views in the context of a GIS specialized as a geographic query tool for an object-relational database. Data from cadastral systems are used in case study examples. These data include the cadastral map (geometric) data and the land registration (administrative) data. The integration of geometric and administrative data is realized by using views based on joins without changing the original data models. Thematic overview maps are created by aggregate views on the administrative data. Finally, views are used to deal with the time aspect of the spatial-temporal data. This paper shows the power of the relational database view concept, but also indicates some problem areas: manipulating with topologically structured data, traversing hierarchical structures, and handling spatial aggregates.

keywords: view, DBMS, cadastre, query tool, GIS

1 Introduction

In this paper, the integration of multiple geographic data sets and associated administrative legal data in one database is described. The geographic data sets consist of large-scale topographic data and the cadastral maps of the provinces in The Netherlands. Associated with the cadastral maps is administrative data, which is also organized per province, but stored in a central mainframe. The relationship between the parcels on the cadastral map and the administrative data is 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. The topographic maps and the cadastral maps contain the full history since the introduction in 1997. This is not (yet) the case for the administrative data.

Section 2 introduces the basic data models for the geometric and administrative data in our case study. In traditional (non-spatial) information systems access to data is obtained 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 query tool database the original data models (base tables) are not changed. However, database *views* are used to present the data in a more appropriate manner: integrated, aggregated, time specific and with cartographic attributes; see Section 3. The overall system architecture is displayed in Figure 1 and is described in more detail in Section 4. Specific add-ons to the basic components of the architecture, object-relational DBMS Ingres and general GIS-front-end GEO++ are described in Section 5.

An earlier paper (van Oosterom & Maessen 1997) did describe a prototype version of the query tool. The current version is in production since August 1999 and is based on a huge nationwide database, which is probably one of the largest vector databases in the world. This paper focuses on the use of database views for several purposes and on changes and new developments since the earlier paper describing the prototype. Conclusions and future work can be found in the last section.

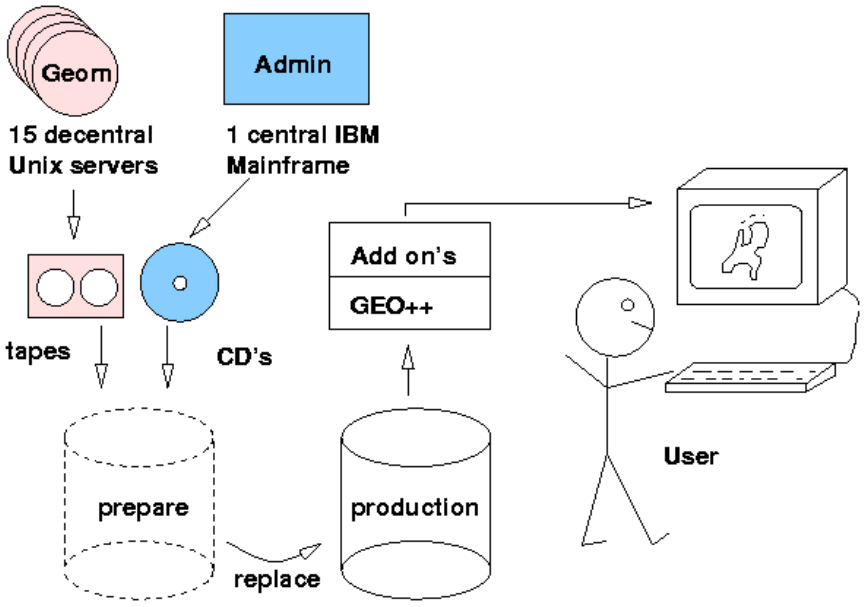

Figure 1: The overall architecture of the query tool

2 Data models

Currently, the large-scale topographic and cadastral data are maintained by the LKI system[1], which stores the data in an Ingres database 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[2] system, which stores the data in an IDMS database on an IBM mainframe. This database will be referred to as the *administrative* database in contrast to the *geometric* database.

The query tool has it own database, 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 query tool environment. However, they have to be included in a query tool which is acceptable for users familiar with this model.

2.1 Geometric model

The geometric model will be described briefly as it has been published before (Lemmen et al. 1998, Lemmen & van Oosterom 1995, van Oosterom 1997). Since 1997 the geometric database keeps track of all changes over time, that is, it is a spatio-temporal database. The metric attributes of type point, polyline and box, are stored in the relational database together with the other attributes describing the measurement (date, accuracy, etc.). Every object is extended with two additional attributes: `tmin` and `tmax`. The objects are valid from and including `tmin` and remain valid until and excluding `tmax`. Current objects get a special `tmax` value: `TMAX_VALUE`, indicating they are valid now.

The geometric data model for the cadastral *parcel* layer is based on winged-edge topology (Baumgart 1975) as described in (van Oosterom 1997); see Figure 2. In addition to the topologically structured cadastral parcel layer, this model also includes *topographic* layers, which are not (yet) topologically structured.

---

[1] LKI stands for *Landmeetkundig Kartografisch Informatiesyteem* (in Dutch): 'Information System for Surveying and Mapping'.

[2] AKR stands for *Automatisering Kadastrale Registratie* (in Dutch): 'Automated Cadastral Registration'.

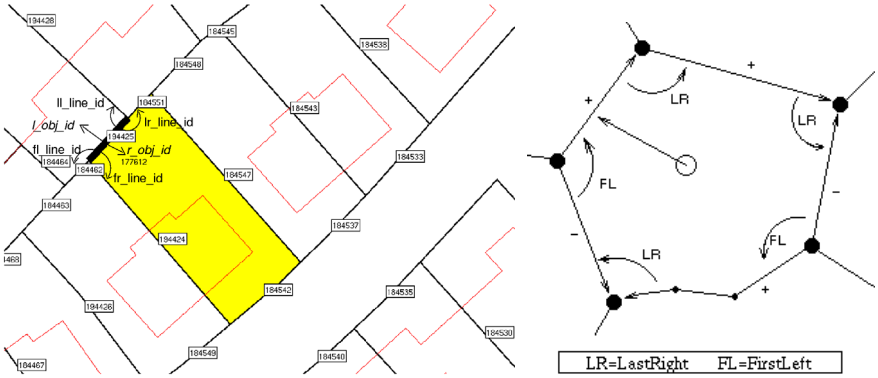

Figure 2: The topologically structured model

2.2 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; see Figure 3: 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 last 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 persons* (organizations), having some attributes in common, but also each having their own attributes.

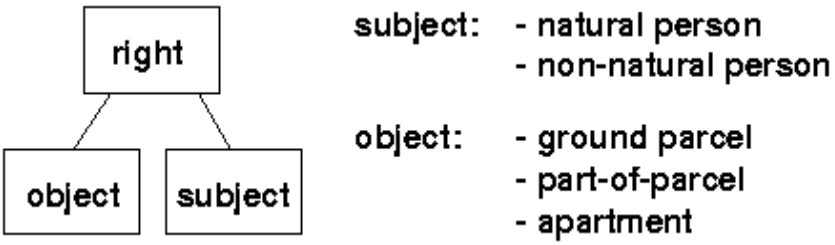

Figure 3: The core of the administrative data model: object, subject and right

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 object, 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; see Figure 4. 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 4). 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.

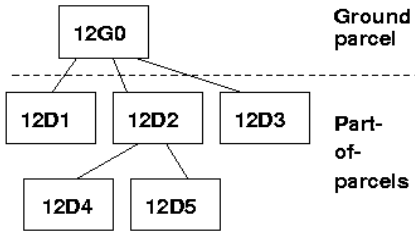

Figure 4: The tree structure representing the part-of-parcel hierarchy

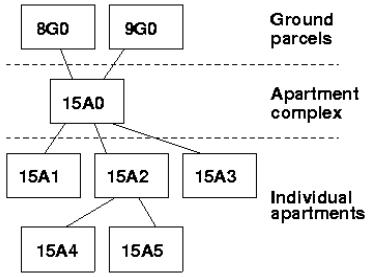

Figure 5: The structure relating ground parcels and apartments

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[3]. Note that the base parcel number of the apartment complex (number 15 in Figure 5) is different from the base parcel number of the related ground parcels (numbers 8 and 9 in Figure 5). 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; see Figure 5.

As already mentioned in the introduction, the administrative model does not (yet) include full history as implemented in the geometric model.

[3] It is not possible that an apartment complex is 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.

3 Database views

Database views play a key role in the architecture of the query tool system. The query tool database contains tables with the unaltered copies of the data from the production databases. The views are used to: integrate data from different tables in one view, see Section 3.1; visualize historic data, see Section 3.2; different geometric visualizations of the same table, see Section 3.3; derive default cartographic attributes, such as color, width, symbol type, from other attributes, also see Section 3.3; derive thematic aggregates without storing the result, see Section 3.4; and present (encoded) attributes in a more clear way. Examples of the later are time stamps are encoded with integers but visualized as readable strings such as '22-04-1998 09:52:50' or legal right codes are short-coded with two characters which can be represented better with a full string.

3.1 Thematic views

Integration of data in a relational database system involves integration of data models. Each data model represents a part of the real world, and is maintained by a specific application. Therefore, the data models cannot be changed. Integration is realized by defining database views using the relational *join* mechanism. The left hand side of Figure 6 colors the land use of parcels, based on the attribute `culture`. The parcels themselves are stored in the geometric data model. The `culture` 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 database view:

```
create view parcel_culture as
  select ap.culture, 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 principal, simply clicking on a parcel gives the corresponding administrative and geometric properties; see right hand side of Figure 6. In the `parcel_culture` view you can see that municip(ality), (o)section and parcel (number) are part of the key[4] on which the geometric and administrative tables are joined.

[4] The cadastral map in the Netherlands is divided into municipalities. These municipalities consist of sections. In turn these sections are divided into parcels, which

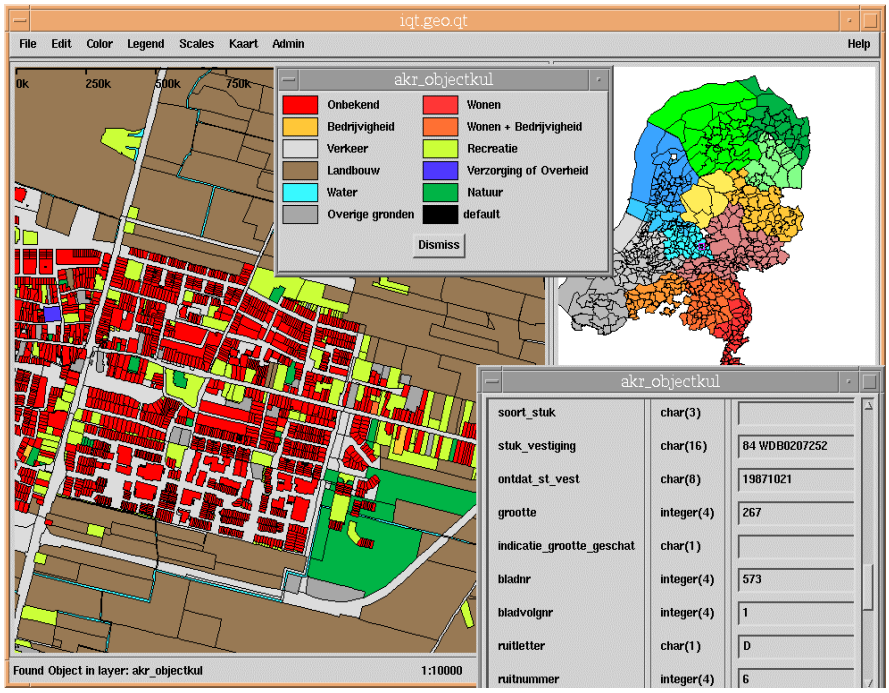

Figure 6: Land use of parcels

A difficulty is that the administrative data model does contain tree-like structures of unknown depth; see Section 2.2. 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 trees. The most interesting information is related to the top and the leaves of the tree; again see Section 2.2.

3.2 Temporal views

To visualize spatio-temporal data (Armenakis 1996, Kraak & MacEachren 1994, Langran 1992) specific techniques are required in the geographic query tool. In the query tool five sets of views are available for the manipulation of the spatio-temporal data. 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 fixed data/time of the administrative data. So, actually `parcel` used in Section 3.1 was

---

form the basic elements of the cadastral map. Therefore, it is implicitly stored to which municipality a parcel belongs.

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 user can specify its 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 user 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]; it is assumed that $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;
```

More advanced visualization techniques for showing spatial temporal data may be implemented in the future; e.g. time animation or time as third dimension.

3.3 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 as 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 (bounding box of a parcel), and parcel (topologically structured area).

Default cartographic attributes, such as color, 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 color value code geo_color 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_color =
   20 + integer(5*(gp.measured_a -gp.legal_a)/gp.measured_a),
         ....          /* other attributes */
  from   parcel gp;    /* geometric parcel */
```

3.4 Aggregate views

Overview or summary maps can be based on spatial aggregation of detail information. If the data is aggregated in terms of relational queries, then this will easily translate into a map. Figure 7 shows the average price of the parcels per municipality created by the view below:

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

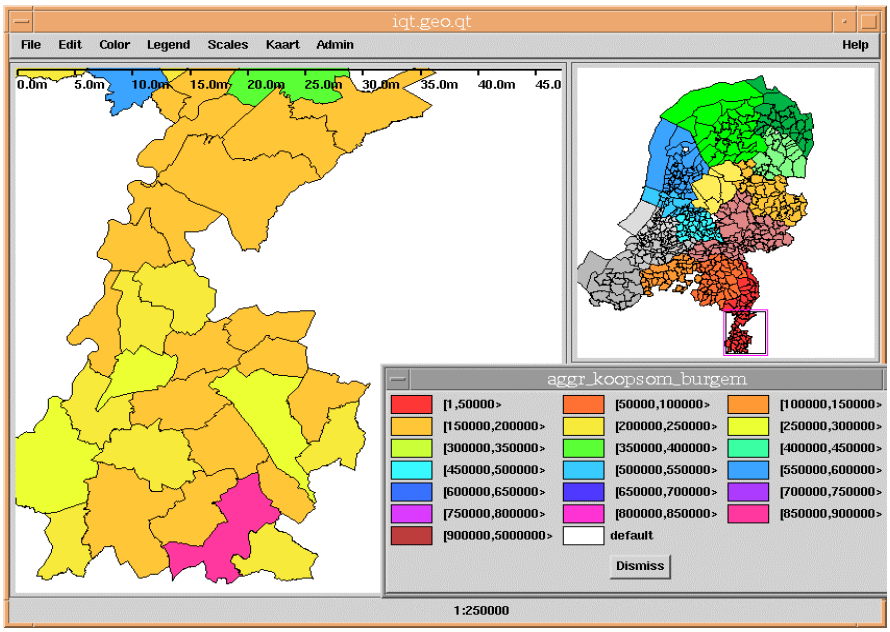

Figure 7: The average parcel price per municipality

In this case we were lucky that the attribute on which we grouped (`municip`) resided in the same table as the attribute on which we wanted to aggregate

(price). 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   average(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, the larger spatial units derived from the parcels (sheet, sections, municipalities, and provinces). These are computed outside the database once and stored (redundantly in the database). They are used as a basis for visualizing aggregated thematic data and for orientation purposes.

Defining aggregates still has to be improved in the query tool. The problem is that there many degrees of freedom: spatial unit (municipality, province), temporal unit (one moment in time or 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 end users?

4 System architecture

The query tool system is based on the Ingres DBMS (ASK-OpenIngres 1994, van Oosterom 1997) and the GEO++ GIS package (IJsselstein & Kap 1995, Professional Geo Systems (PGS) 1996, Vijlbrief & van Oosterom 1992). The administrative data and geometric data models have been discussed in Section 2. The integration of these models is realized through views as described in Section 3. The query tool database is periodically filled with complete copies of the 15 decentral spatial databases from LKI and 15 centralized administrative databases from AKR; see Figure 1. Currently, the data is loaded two times per year into a single Ingres DBMS. Loading the data includes defining indices, computing geometric aggregates, collecting statistics (the basis to produce 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 60 Gb database including the index structures. During

loading, the previous version of the query tool database remains available to the users.

4.1 GIS front-end

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

At this moment, most GISs do not support this open database approach. The traditional approach is that geometric data has to be read into the GIS vendor specific format. Quite often, GISs have the capability to access an external database system. 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 database system. This is still not satisfactory, since complete integration of geometric and administrative data is the most efficient and consistent data management architecture. A geographic query tool must function as a front-end to the database backend. The geographic query tool assists the user in posing questions through a friendly user interface. The query tool generates ANSI SQL dynamically and processes the responses of the database system. GEO++ is a GIS which is designed to deal with this. Therefore, GEO++ can be described as a generic query tool for relational databases with spatial extensions, similar to OpenGIS SFS/SQL (Open GIS Consortium, Inc. 1998).

The query tool is nationwide and available for analyzing and performing consistency checks on the cadastral data. One of the main goals is to improve the quality of the source data. The purpose is to create an environment with easy access to all the data, in which the data can be analyzed, queried and filtered. Therefore a user-friendly interface is needed on top of the database. This was realized by using standard GEO++ together with custom made add-ons, which will be described in Section 5.

4.2 Database aspects

A few numbers describing the size the geometric data including history in the nationwide database: 9.300.000 parcels, 25.200.000 boundaries, 31.200.000 topographic lines, 5.100.000 symbols and 5.200.000 text labels. The boundary and line entities are based on polylines and circular arcs. The total number of different line segments in the database is over 250.000.000. The administrative databases contain the following amount of data (without history): 7.500.000 objects (that is, ground parcels, part-of-parcels, or apartments), 9.700.000 object addresses,

7.100.000 subjects (that is, persons or organizations including their subject address), 10.100.000 right records (relationship between object and subject), 1.900.000 object limitations (legal notifications, restricting the use of the object due to some reason), etc.

When selecting a certain area the query tool displays the result with the same interactive speed for this nationwide database as for a database with only a smaller region; e.g. province. Also integrated views (e.g. showing the prices of the objects color coded on the parcel) are at the same speed as pure geometric views. The same is true for the historic views. It is interesting to note that the topology speeds up the visualization compared to a representation without topology, because in this case all coordinates are transferred twice from database to the application. The key entries to the database are a region (usually a rectangle, sometimes just one point), parcel number, address and (subject) names. Whenever possible data is clustered based on spatial location. This is obvious for the geometric data using the spatial location code SLC (van Oosterom & Vijlbrief 1996). However, this is also applied to the administrative data by clustering on parcel number, which contains a municipality and section code, or on postal/zip code. This enables spatial range queries to perform well in all situations including the integrated views. The other entries are supported by secondary indices (btree (Comer 1979) or rtree (Guttman 1984)), because they usually return one or a few results.

Note that instead of copying all production databases into one query database, it would also be possible to define the tables and the views in a sense of a distributed database. In the case of the Ingres DBMS using the Net and Star products. This should also work in combination with other relational databases using 'gateways'. However, the query database is very large, probably one of the largest vector databases in the world, and organizing the data (clustering and indexing) is very important for performance. This may not always be possible in the original production systems, especially in the administrative system.

5 Cadastral Add-ons

With standard Ingres and GEO++ it is possible to query and visualize the base tables and views in a user-friendly manner. It is also possible to specify a selection condition for a table or view by graphically creating the where-clause of the SQL statement. The *first* extensions developed by the Cadastre, run inside the database. Using Ingres OME (Object Management Extensions) types, both spatial and traditional types, are extended with new functions. For example added are: the union function of two boxes or the translation function of an integer encoded date/time into a readable data/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 tables. The *second* type of cadastral extensions include additional visualization functions 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).

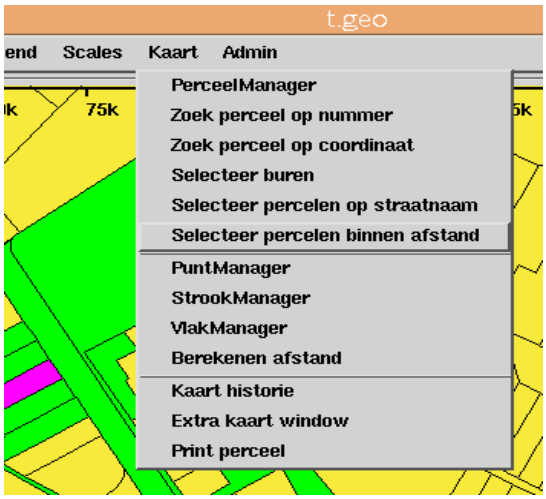

Figure 8: The map 'kaart' pull-down menu (Dutch Interface)

Both Ingres and GEO++ (data model driven using the DBMS system catalog information) have no knowledge of the semantics of the data models being manipulated. This means that these tools are very general and can be used in many situations. For the 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 & van Oosterom 1992). To the user of the query tool, these add-ons are available via two additional pull-down menus: one for the 'map entrance' to the data (see Figure 8) and one for the 'administrative entrance' to the data (see Figure 9).

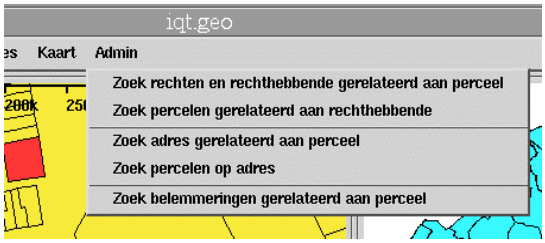

Figure 9: The admin pull-down menu (Dutch interface)

These custom made add-ons in the query tool are used for easier access to data, that is, the user can hit just one button, instead of querying several tables to obtain a specific result; see Section 5.3. Also, the add-ons are used when the analysis is not possible in a relational database. For example, the selection based on intersection of a topologically structured area feature with an input polyline; see Section 5.2. Finally, a new interface concept has been introduced in the add-ons, the *active set*; see Section 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 query tool it became clear that there was a need for selecting objects (parcels) and saving them for further processing. For this 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.

Figure 10: Parcels crossed by the Amsterdam Subway route

5.2 The geographic interaction extensions

In the map 'kaart' pull-down menu, the extensions with mainly geographic interaction can be found; see Figure 8. Translated from this figure the options are: ParcelManager, Find parcel by number, Find parcel by coordinate, Select neighbors, Select parcels by cartographic text, Select parcels within distance,

PointManager, LineManager, AreaManager, Map history, Additional map window, and Print parcel. Each of these options will give the user a dialog form controlling the (parameters of the) requested actions. It goes too far to describe all dialog forms in detail and most operations should be clear by their name.

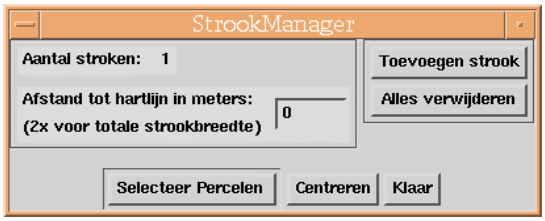

Figure 11: The dialog form LineManager (StrookManager in Dutch) to manipulate line selecion

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 front-end, the polygons of the relevant parcels are formed using the topology information from the database. Then the actual test, e.g. point-in-polygon, is done in the front-end. In an objected-oriented database, these queries could be completely executed within the database, but this is not the case in a relational database. Figure 10 shows the Amsterdam Subway route and the corresponding parcels that are crossed. The LineManager dialog form is shown in Figure 11 and enables the user to create new lines, specify the width of buffer zones, select the crossed parcels, etc.

5.3 The administrative interaction extensions

In the admin pull-down menu the extensions with mainly administrative interaction can be found; see Figure 9. Translated from the pull-down menu in this figure 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; see Section 5.1. The other options are used to easily obtain administrative information related to the parcels in the active set. Figure 12 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 Amsterdam Arena, the soccer stadium of Ajax

Amsterdam. Without this form, the user must traverse the following 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; see Section 2.

6 Conclusion

In this paper it was demonstrated that the concepts of views and relational joins form 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 database 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 database can be done with 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 moments in time display). As demonstrated in the paper 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.

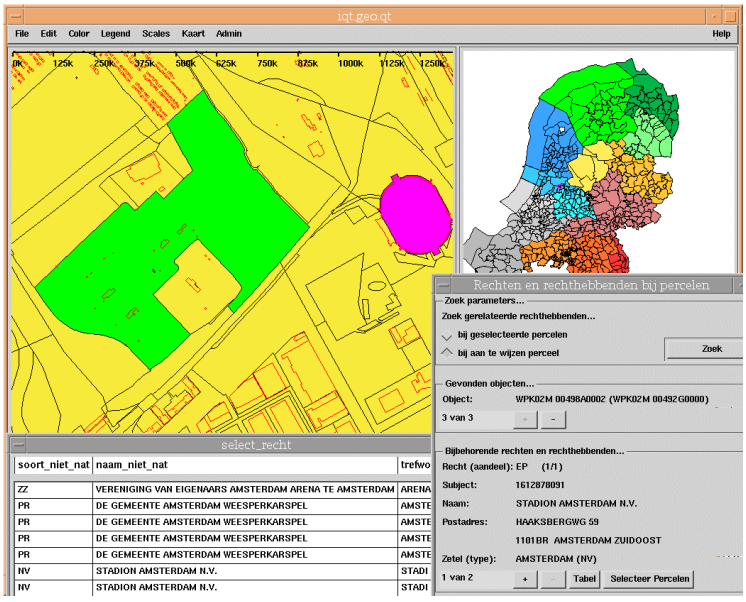

Figure 12: Searching for parcels by owner, Amsterdam Arena

Several types of tasks from the real world (applications) have been supported by the query tool. To give an impression of its possibilities, a few tasks assisted since the querytool went into production (August 1999) are mentioned here:

- in cooperation with a utility company: investigate if all parcels crossed by their cable have the proper legal status;
- for a geo-marketing company: derive a postal (zip) code map from the cadastral map and aggregate parcels with the same postal code;
- for an agricultural land exchange plan, find parcels owned by farmers which are a long distance from their farm;
- in order to produce bills for the maintenance of the topographic map, find the number of changed objects per municipality per month;
- aggregate thematic information (e.g. average price) and visualize the result on a geometric aggregation of the same level (e.g. section).

Of course the first users also did have some new functional wishes after using the system. The first wish, is specifying his/her own views, based on joins, and have attributes from multiple tables in the result. The second wish is to have more import and export functionality. The third wish is to have a more up to data query tool database. Therefore, instead of loading full data sets two times per year to the query tool database, in the future the data will have to be replicated periodically more frequently from our geometric and administrative 'production' databases. 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 monthly.
Further, external users should be able to query the data over the internet. Limited functionality of the geographic query tool is implemented in Java (Arnold & Gosling 1996) as a prototype for the digital Geoshop (van den Berg et al. 1997). This must be coupled to the NCGI (National Clearinghouse Geo-information) (de Gunst & van Oosterom 1997, Intergraph 1978, Jacobi & Lind 1997, Radwan et al. 1997, van de Kieft & Kok 1997) and based on OpenGIS standards (Buehler & McKee 1998), specifically the standards currently being developed in the web-mapping testbed.

Acknowledgments

We would like to thank the persons involved in the implementation of the query tool system. 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 database. Finally, Joep Mathijssen, who developed large parts of the cadastral add-ons and also introduced the *active set* concept; see Section 5.

References

C. Armenakis (1996). *Mapping of spatio-temporal data in an active cartographic environment*, Geomatica 50 (1996), no. 4, 401-413.

Ken Arnold and James Gosling (1996). *The java programming language*, Addison-Wesley Publishing Comapny, Inc., Reading, Massachusetts.

ASK-OpenIngres (1994). *INGRES/Object Magemenent Extention User's Guide, Release 6.5*, Tech. Report.

Bruce G. Baumgart (1975). *A polyhedron representation for computer vision*, National Computer Conference, pp. 589-596.

Kurt Buehler and Lance McKee (1998). *The opengis guide - introduction to interoperable geoprocessing*, Tech. Report Third edition, The Open GIS Consortium, Inc.

Douglas Comer (1979). *The ubiquitous B-Tree*, ACM Computing Surveys 11, no. 2, 121-137.

Marlies de Gunst and Peter van Oosterom (1997). *Network computers at the dutch cadastre*, 46th Photogrammetric Week, Stuttgart, September 22-26, 1997.

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

J. A. IJsselstein and A. P. Kap (1995). *Het kadastraal perceel: een stevig fundament!*, Nederlands Geodetisch Tijdschrift Geodesia 37, no. 7/8, 343-349, (In Dutch).

Intergraph (1978). *Introduction to data management and retrieval system (DMRS)*, Tech. Report 78-057, M&S Europe B. V., Netherlands.

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

M. J. Kraak and A. M. MacEachren (1994) *Visualization of the temporal component of spatial data*, 6th SDH, September 1994, pp. 391-409.

G. Langran (1992). *Time in geographic information systems*, Taylor & Francis, London.

Christiaan H.J. Lemmen, Ernst-Peter Oosterbroek, and Peter J.M.van Oosterom (1998). *New spatial data management developments in the netherlands cadastre*, proceedings of the FIG XXI interenational congres, Brighton UK, commission 3, Land Information Systems, July 1998, pp. 398-409.

Christiaan H.J. Lemmen and Peter J.M. van Oosterom (1995). *Efficient and automatic production of periodic updates of cadastral maps.*, JEC'95, Joint European Conference and Exhibition on Geographical Information, The Hague, The Netherlands, March 1995, pp. 137-142.

Open GIS Consortium, Inc. (1998). *Opengis simple features specification for sql*, Tech. Report Revision 1.0, OGC, March 1998.

Professional Geo Systems (PGS) (1996). *The GEO++ system, version 2.80, Reference manual*, Tech. Report.

M.M. Radwan, Y. Bishr, N. Friha, O. Driza, and J. Pandya (1997). *Guidelines for the development of a geospatial clearing house in a heterogeneous environment*, Third Joint European Conference & Exhibition on Geographical Information (JEC-GI'97), vol. 1, April 1997, pp. 277-287.

I.A. van de Kieft and B. Kok (1997). *The development of a geo metadata service for the netherlands*, Third Joint European Conference & Exhibition on Geographical Information (JEC-GI'97), April 1997, pp. 1165-1176.

Carel van den Berg, Frank Tuijnman, Tom Vijlbrief, Co Meijer, Harry Uitermark, and Peter van Oosterom, *Multi-server internet gis: Standardization and practical experiences*, International Conference and Workshop on Interoperating Geographic Information Systems, Santa Barbara, California, USA, December 3-4 and 5-6, 1997, interop'97 (Boston), Kluwer Academic Publishers, 1997, pp. 365-378.

Peter van Oosterom (1997). *Maintaining consistent topology including historical data in a large spatial database*, Auto-Carto 13, April 1997, pp. 327-336.

Peter van Oosterom and Bart Maessen (1997) *Geographic query tool*, JEC-GI'97, Joint European Conference and Exhibition on Geographical Information, Vienna, Austria, vol. 1, April 1997, pp. 177-186.

Peter van Oosterom and Tom Vijlbrief (1996). *The spatial location code*, Proceedings of the 7th International Symposium on Spatial Data Handling, Delft, The Netherlands.

T. Vijlbrief and P. van Oosterom (1992), *The GEO++ system: An extensible GIS*, 5th SDH, pp. 40-50.