Research issues in integrated querying of geometric and thematic cadastral information (1)

Peter van Oosterom

GISt Report No. 1
Delft, August 2000

Summary: *This document describes different types of research oriented applications with respect to querying cadastral information. Most applications require both the geometric and thematic/legal aspects of the cadastral information. The work described in this document has been done in the context of the 'Cadastral query tool' system in the first quarter of the year 2000. First a short summary of the architecture of the query tool system is given, followed by an overview of the cadastral data model. Then, the research aspects of the following queries will be described in more detail: find neighbors of parcels owned by the province of North-Holland, selecting parcels with monument, legal notifications due to NAM pipelines, and finding potential agricultural parcels suitable for exchange.*

Faculty of Civil Engineering and Geosciences
Department of Geodesy, Section GIS Technology
Thijsseweg 11, 2629 JA Delft, the Netherlands
oosterom@geo.tudelft.nl

Contents

List of Figures

1 Introduction

In this document the integration of cadastral geographic data sets and associated administrative legal data in one database is described. The relationship between the parcels on the cadastral map and the administrative data is realized through the nationwide unique parcel numbers. The cadastral maps are based on a topologically structured model. Manipulating area features in such a model requires navigation using the topology references to the boundaries. The topographic maps and the cadastral maps contain the full history since 1997. This is not (yet) the case for the administrative data. Section 2 introduces the basic data models for the geometric and administrative data.

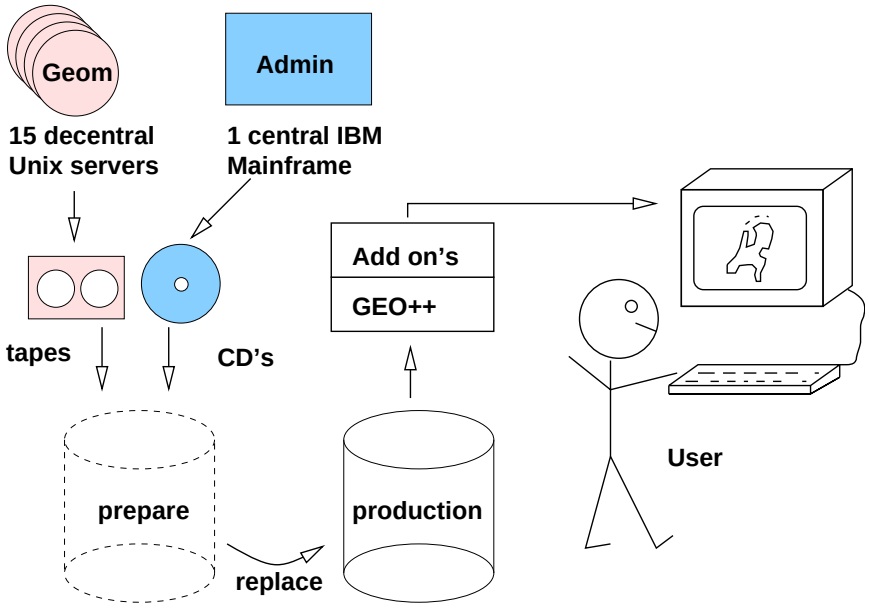

Figure 1: The overall architecture of the query tool.

An earlier paper [10] described a prototype version of the query tool. The current version is in production since August 1999. 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; see [12]. The views are used to:

- integrate data from different tables in one view,
- visualize historic data,
- different geometric visualization of the same table,
- derive default cartographic attributes, such as color, width, symbol type,
- derive thematic aggregates without storing the result, and
- present (encoded) attributes in a more clear way.

Examples of the later are time stamps which are encoded with integers but visualized as readable strings such as '22-04-1998 09:52:50' or legal right codes which are short coded with two characters which can be represented better with a full string. The overall system architecture is displayed in Figure 1. The query tool system is based on the Ingres DBMS [1, 9] and the GEO++ GIS package [5, 8, 14]. Specific add-on's to these basic components

of the architecture are described in [12]. The custom made add-on's are used for:

- *easier access to data:* just one button, instead of querying 4 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'.

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.

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. In this document a number of applications with research aspects will be described in more detail: finding neighbors of parcels owned by the province of North-Holland (Section 3), selecting parcels with monument (Section 4), legal notifications due to NAM pipelines (Section 5), and finding potential agricultural parcels suitable for exchange (Section 6).

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)[1, 9]. 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 model 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.

Subsection 2.1 gives an overview of the geometric model, Subsection 2.2 gives an overview of the administrative model, and Subsection 2.3 gives an impression of the size of the overall database.

2.1 Geometric model

The geometric model will be described very briefly as it has been published before quite extensively [6, 7, 9]. 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 [2] as described in [9]; see Figure 2. In addition to the topologically structured cadastral parcel layer, this model also includes *topographic* layers, which are not (yet) topologically structured.

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

[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'.

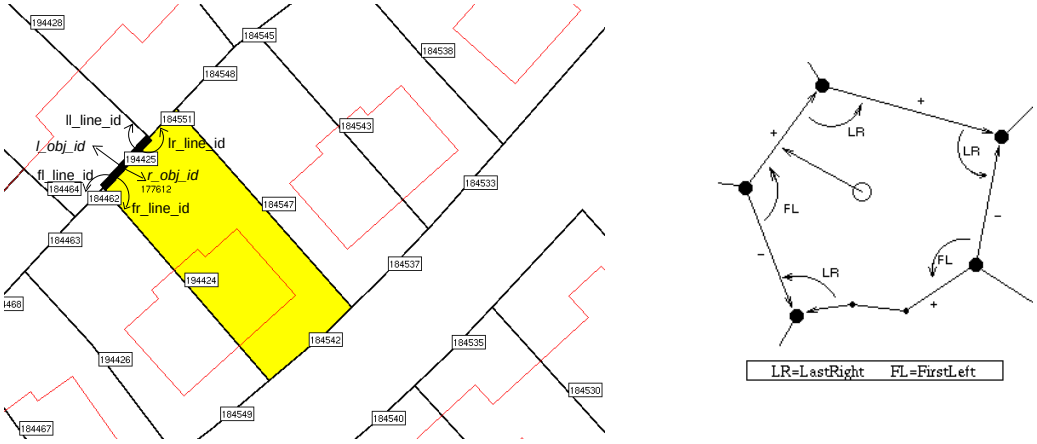

Figure 2: The topologically structured model.

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

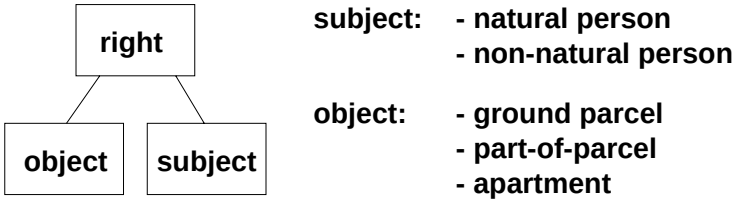

Figure 3: The core of the administrative data model: object, subject, and right.

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 looks like '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, that is an hierarchical structure. The apartment

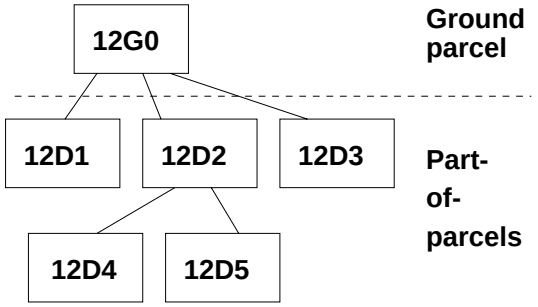

Figure 4: The tree structure representing the part-of-parcel hierarchy.

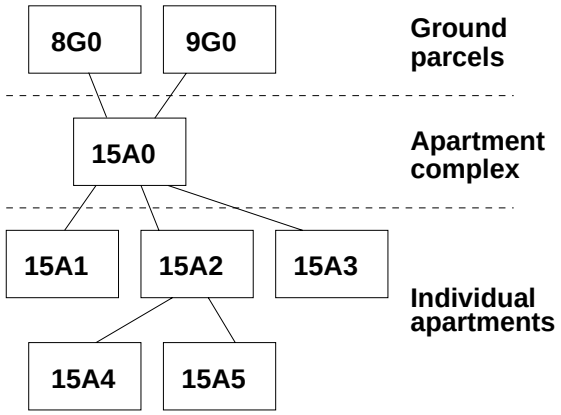

Figure 5: The structure relating ground parcels and apartments.

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.

2.3 Size of the database

A few numbers describe 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 (ground parcels, part-of-parcels, or apartments), 9,700,000 object addresses, 7,100,000 subjects (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. a 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 the 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)

[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.

names. Whenever possible data is clustered based on spatial location. This is obvious for the geometric data using the spatial location code SLC [13]. 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 [3] or rtree [4]), because they usually return one or a few results.

3 Find neighbors of parcels

Original question from Ad Scholman (Cadastre region NW):

q93:
De provincie NH wil eenmalig de kadastrale info uit AKR geleverd krijgen van al haar eigendommen EN alle aangrenzende percelen. Hiervoor willen zij eerst een offerte ontvangen. In deze offerte dienen beide aantallen vermeld te zijn. Daarom het volgende verzoek: Kunnen op korte termijn (enkele dagen) de gevraagde aantallen geleverd worden. Kunnen op wat langere termijn de betrokken percelen geleverd worden (bestand).

Translation: the province of North Holland wants to have a single delivery of cadastral information from AKR related to all parcels owned (VE) by the province and the same for all the neighbor parcels. First, they want to receive a quotation based on the numbers of parcels (own parcels and neighbors). Therefore, the following request: please provide the mentioned numbers of parcels within a couple of days. Also, after that send two files with the involved parcels.

Later on a similar question but now with respect to legal notifications was posed by Ad Scholman:

q98:
Vind de percelen met BZ (objectbelemmeringen) rechten van de provincie Noord Holland (belemmeraar) en de buurpercelen hiervan (vwb Alkmaar is dit subjnr 1100075010 en vwb Amsterdam is dit subjnr 1603281134.

Translation: Find the parcels on which the province of North Holland has a legal notification of BZ(D) and also find the neighbor parcels. 'BZ' stand for legal notification described as (in Dutch) 'Zakelijk recht als bedoeld in art.5, lid 3, onder b, van de Belemmeringenwet Privaatrecht'. The province of North Holland is maintained by two cadastral offices, therefore the province is indicated by two different subject numbers in the AKR databases: 1100075010 and 1603281134.

The result was obtained by a mixture of interactive work with the query tool and a SQL script `prov_buren.sql`. In this section the second question (q98) will be described in more detail (q93 is very similar). First, with a SQL script the parcels on which the province has a legal notification of type BZ(D) is executed. This SQL query uses the geometric parcel `lki_parcel` table and the administrative table with legal notifications `mo_obj_belemmering`. These tables are joined on the parcel number `x_akr_objectnummer`. The other parts of the where clause select the proper type of legal notifications using the attribute `soortbelem` and only those for the province using the attribute `belemmeraar`.

```
insert into select_parcel
select lki_parcel.*, object_key('0000000000000000')
from lki_parcel, mo_obj_belemmering
where lki_parcel.x_akr_objectnummer=mo_obj_belemmering.x_akr_objectnummer
  and ((mo_obj_belemmering.soortbelem = 'BZ')
    or (mo_obj_belemmering.soortbelem = 'BZD'))
  and ((mo_obj_belemmering.belemmeraar = '1603281134')
    or (mo_obj_belemmering.belemmeraar = '1100075010'));\p\g
```

This SQL query results in 1699 selected ground parcels `select_parcel`. Using the 'Parcel-Manager' of the interactive part of the query tool interface double parcels are removed.

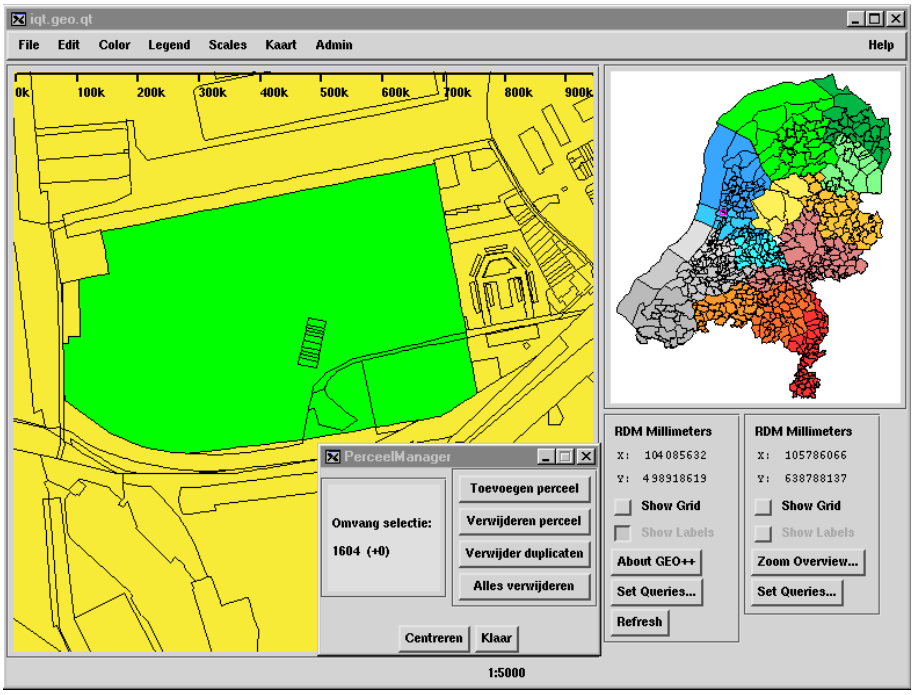

Figure 6: Some parcels for which there is a legal notification of type BZ(D) for the province of North Holland.

This results in 1604 unique parcels. Figure 6 shows some of the selected parcels. The result is copied in a file `provBZ.txt` with the following SQL statement:

```
copy select_parcel (x_akr_objectnummer=c0nl ) into 'provBZ.txt'\g
```

The graphic interface is used to find all neighbors of the 1604 selected parcels. This is the 'Selecteer buren' option. Now the total number of parcels is 4496 (without doubles); see Figure 7. Finally, the result is copied in a file `provBZbuur.txt` with the last SQL statement in the script:

```
copy select_parcel (x_akr_objectnummer=c0nl ) into 'provBZbuur.txt'\g
```

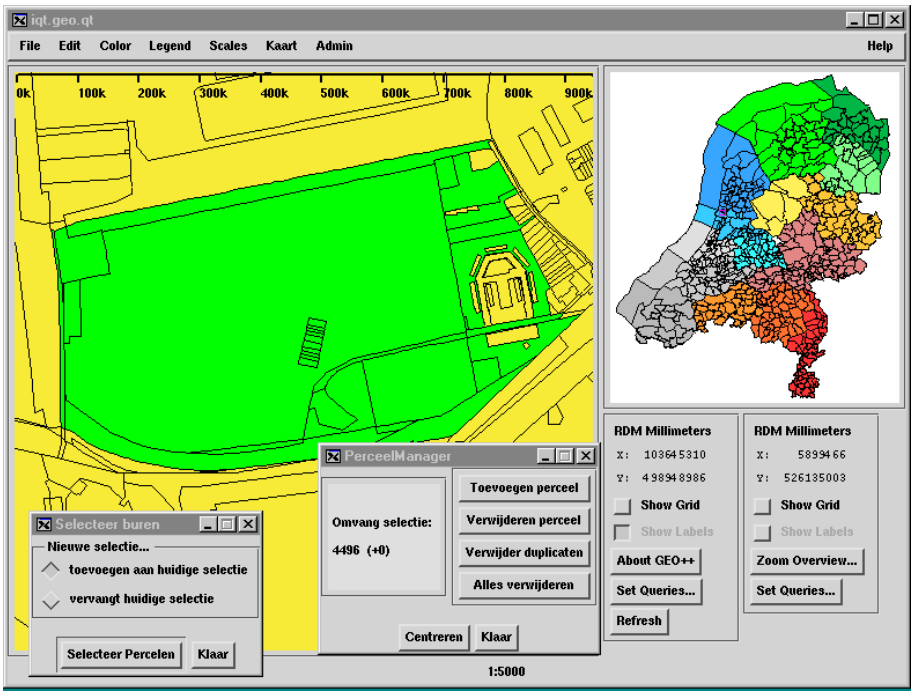

Figure 7: Neighbors of parcels for which there is a legal notification of type BZ(D) for the province of North Holland.

4 Selecting parcel with monument

Original question from Peter Nieland (Kadata):

q91
Willen jullie voor mij van de volgende gemeenten: 'Gennep, Grubbenvorst, Haelen, Horst, Kessel, Maasbree, Meerssen, Mook en Middelaar, Onderbanken, Roggel en Neer, Sevenum, Susteren, Venlo, Venray, Weert, Assen, Coevorden, Axel, Hontenisse, Hulst, Oostburg, Sas van Gent, Sluis-Aardenburg, Terneuzen en Gorinchem' de kadastrale percelen incl. bebouwing (LKI gegevens) leveren met daar aan gekoppeld de massale output gegevens (object, subject). De selectie is die percelen die voorzien zijn van MW, MWD, BS en BSD.

Translation: please find for the following municipalities '...', the cadastral parcels with legal notification MW, MWD, BS or BSD (indicating the presence of a monument on the parcel). Besides the parcels, also include the buildings from LKI and the legal/administrative information from AKR (object, subject).

The result was obtained by interactive inspection in the query tool and two shell/SQL scripts (`convert_mw.sh` and `base_mw.sh`) to do the actual selection. As there are many municipalities involved it was decided to organize the output per cadastral office. The main script `convert_mw.sh` specifies the requested municipalities per office by listing the relevant cadastral municipality codes (`gennep_new` is just a small test case).

```
#!/bin/sh

DB=${1:-iqt_prd1_0100}
```

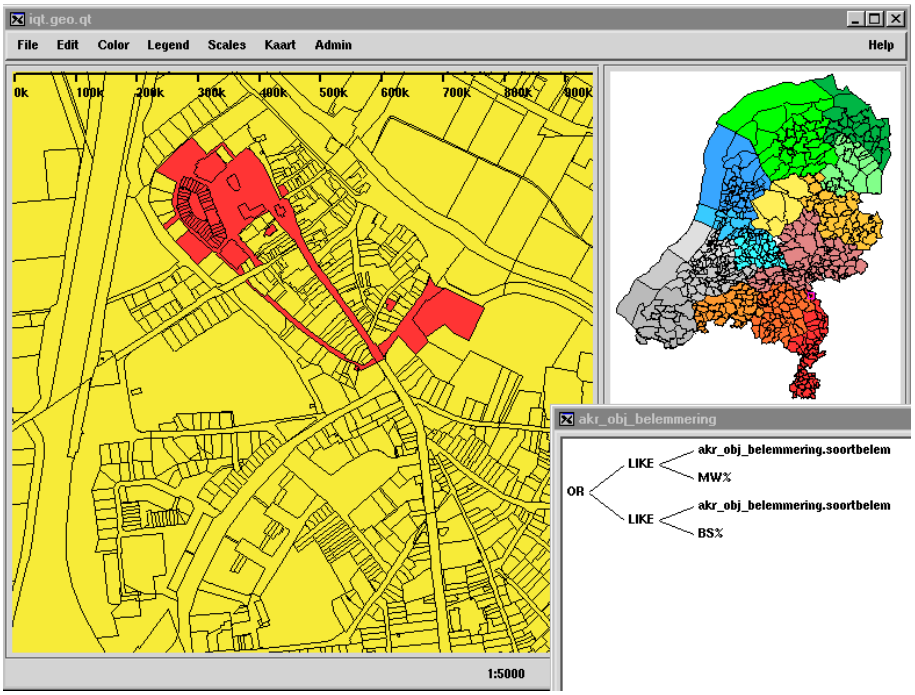

Figure 8: Parcels with legal notification of type BZ(D) or BS(D) in the center of Gennep.

```
./base_mw.sh $DB "'GNP00'" gennep_new

./base_mw.sh $DB "'GNP00','OTS00','GBV00','BGN04','HLN02','HOR02',
  'NHM00','HOR01','KSL00','MBE00','BDE00','GLE02','MSN01','ULS00',
  'M0000','BGR02','JBK00','MKB00','SVD00','NEE00','RGL00','SVN01',
  'NST02','RTR00','STR01','VL000','VRY00','SRY00','WEE01'" limburg

./base_mw.sh $DB "'ASN00','CVD00','DLN00','OTH02','SLE00','ZL000'" drente

./base_mw.sh $DB "'AEL00','HTN01','HUL00','JSN00','OBG00','PLP00',
  'SAS00','WDP00','ADB00','SLU00','HOE02','TNZ00','ZSG00'" zeeland

./base_mw.sh $DB "'GRC00'" zuidholland
```

Figure 8 gives an impression of parcels selected based on the legal notification of type BZ(D) or BS(D) in the center of Gennep. The 'real' work is done by the script `base_mw.sh`, which takes three input parameters: 1. database, 2. list of cadastral municipality codes, and 3. name of region (cadastral office). Below the parts of the contents of the script `base_mw.sh` is given interleaved with explanations:

```
#!/bin/sh

DB=${1:-iqt_prd1_0100}
# Limburg:
MUNICIP=${2:-"'GNP00'"}
NAME=${3:-gennep}
TMPDIR=/usr/data/iqtdata/tmp/mw/$NAME
DBUSER=oostep

mkdir $TMPDIR
sql -u$DBUSER -f8G15.6 $DB << EOF
```

```
/************** FIRST PART ****************/
drop table sel_p\p\g
create table sel_p as
select distinct
   obj_id=x.object_id, box=x.geo_bbox, x_akr_obj=x.x_akr_objectnummer
from mo_obj_belemmering o, lki_parcel x
where (o.soortbelem = 'BSD' or o.soortbelem = 'BS'
   or o.soortbelem = 'MWD' or o.soortbelem = 'MW')
   and (o.municip = x.municip and
        o.osection = x.osection and
        o.parcel = x.parcel)
   and o.municip in ($MUNICIP)
   with nojournaling;\p\g

grant select on sel_p to public;\p\g
EOF
```

The SQL query selects the parcels based on a join of the parcel `lki_parcel` and legal notification table `mo_obj_belemmering`. Of course, only the proper types of legal notifications should be selected `soortbelem` (MW,...) and in the specified municipalities `municip in ($MUNICIP)`. The selected parcels are saved in the temporary table `sel_p`. The keyword `distinct` is used to avoid double selection of the same parcel. This could occur in case there are multiple legal notifications.

```
/************** SECOND PART ****************/

drop table seltmp_line\p\g
create table seltmp_line(
        ogroup, object_id, slc, classif, interp_cd,
        geo_polyline, height, accu_cd,
        geo_bbox, linelen, object_dt,
        tmin, tmax, sel_cd, source, quality, vis_cd
) as select distinct
        ogroup, object_id, slc, classif, interp_cd,
        char(char(geo_polyline),1200), height, accu_cd,
        char(geo_bbox), linelen, object_dt,
        lkidate2int(tmin), lkidate2int(tmax), sel_cd, source, quality, vis_cd
from lki_line, sel_p where overlaps(geo_bbox,box)=1
     and sel_cd LIKE '%B%'
with nojournaling; \p\g
EOF
sql -u$DBUSER -f8N15.5 $DB << EOF

grant select on seltmp_line to public;\p\g
copy seltmp_line(
        ogroup= c0tab, object_id= c0tab, slc= c0tab, classif= c0tab, interp_cd= c0tab,
        geo_polyline= c0tab, height= c0tab, accu_cd= c0tab,
        geo_bbox= c0tab, linelen= c0tab, object_dt= c0tab,
        tmin= c0tab, tmax= c0tab,
        sel_cd= c0tab, source= c0tab, quality= c0tab, vis_cd= c0nl
) into '$TMPDIR/lki_line.oos'\p\g

EOF
cat $TMPDIR/lki_line.oos | sed 's/ * / /g'\
                         | sed 's/ *$//'  > $TMPDIR/lki_line.dat
```

The second part of the base_mw.sh script copies the LKI information per table to an ASCII file base on overlap between the sel_p objects and the objects in the table; for example: create table seltmp_line and copy seltmp_line.... The temporary table seltmp_line is used to first convert the spatial Ingres into character strings. This is not possible in the Ingres copy statement. Note that the view lki_line is used and not the base table xfio_line. This implies that only data at the specified time (date of AKR data) are exported without historic information. The time stamps tmin and tmax are converted back to integer values using the function lkidate2int. After copying the data in the file some redundancy spaces are removed with a sed script.

Other tables are converted in the same way as the lki_line table: xfio_gcpnt, xfio_sympnt, xfio_text, xfio_boundary, xfio_parcel, and xfio_parcelover. This code is not shown is this document again. There are two small differences in this code. First, in case of xfio_boundary the records are selected by using the l_obj_id and r_obj_id attributes instead of the overlap test with the geo_bbox. Similarly, in case of xfio_parcel and xfio_parcelover the records are selected based on the object id's from the sel_p table. Some further post-processing is required to convert the LKI ASCII file selections into other file formats (e.g. MapInfo).

```
/************** THIRD PART *****************/

sql $DB << EOF
set nojournaling\g

drop table seltmp_object;\p\g
create table seltmp_object as select *
from mo_object
where x_akr_objectnummer in (select x_akr_obj from sel_p)
with nojournaling;\p\g
modify seltmp_object to btree on x_akr_objectnummer;\p\g
grant select on seltmp_object to public;\p\g

drop table seltmp_adrobject;\p\g
create table seltmp_adrobject as select *
from mo_objectadres
where x_akr_objectnummer in (select x_akr_obj from sel_p)
with nojournaling;\p\g
modify seltmp_adrobject to btree on x_akr_objectnummer;\p\g
grant select on seltmp_adrobject to public;\p\g

drop table seltmp_recht;\p\g
create table seltmp_recht as select *
from mo_recht
where x_akr_objectnummer in (select x_akr_obj from sel_p)
with nojournaling;\p\g
modify seltmp_recht to btree on x_akr_objectnummer;\p\g
grant select on seltmp_recht to public;\p\g

drop table seltmp_subject;\p\g
create table seltmp_subject as select *
from mo_subject
where subject_id in (select gerechtigde from seltmp_recht)
with nojournaling;\p\g
grant select on seltmp_subject to public;\p\g
```

```
EOF

# Note that the extension of the generated file names is
# derived from the user name (e.g. oos from oostep)
# by the copydb statement
copydb -c $DB seltmp_object seltmp_adrobject seltmp_recht seltmp_subject
cat copy.out |\
  sed 's/varchar(0)/c0/' |\
  sed 's/oid= c0tab,//' |\
  sed 's/nl= d1//'  |\
  sed 's/c0nl,/c0nl/' > copy.out2

sql $DB < copy.out2

cat seltmp_o.oos | sed 's/ * / /g' \
                 | sed 's/ *$//' > $TMPDIR/akr_object.dat
cat seltmp_a.oos | sed 's/ * / /g' \
                 | sed 's/ *$//' > $TMPDIR/akr_adrobj.dat
cat seltmp_r.oos | sed 's/ * / /g' \
                 | sed 's/ *$//' > $TMPDIR/akr_recht.dat
cat seltmp_s.oos | sed 's/ * / /g' \
                 | sed 's/ *$//' > $TMPDIR/akr_subject.dat

rm seltmp_o.oos seltmp_a.oos seltmp_r.oos seltmp_s.oos
rm $TMPDIR/lki_*.oos
```

The third part of the script selects the administrative information related to the parcels in `sel_p`; for example `create table seltmp_object` using the subselect in the where clause. This is done for the object, the object addresses, the rights and the subjects. Again the `sel_p` table is used in the where clause to specify the required selection. Please pay attention for the subtle difference in the where clause of the creation of the table `seltmp_subject`. After filling the temporary table, the Ingres tool `copydb` is used to copy the selected information into files and again `sed` is used to do some textual post processing.

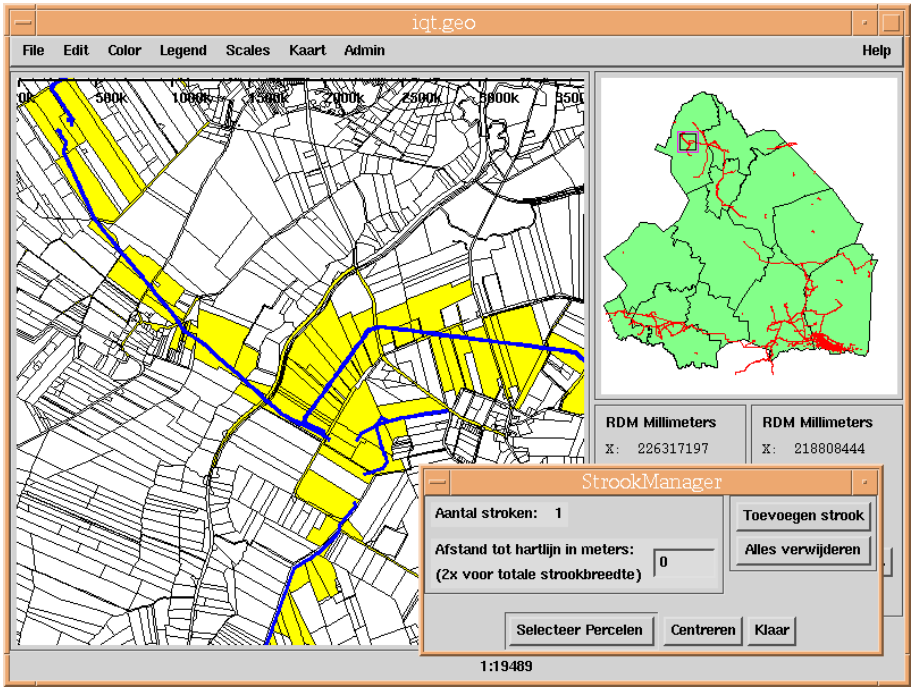

Figure 9: Pipelines of the NAM and the parcels they cross.

5 Legal notifications due to NAM pipelines

Original question posed by Gerard Hesselink (ORK):

q33:
Inbrengen leidingen van de NAM voor opschonen belemmeringen. Mogelijke pilot. (Generiek aspect: hoe omgaan met data van externen.)

Translation: import external data from the NAM in order to improve the quality of the registration of legal notifications due to NAM pipelines. This question contains a generic aspect: how to deal with data from external parties.

Later on this pilot project did get a follow up and quite a lot of requests followed, among others the question by Henk Averink (working on the 'BP project' in Zwolle):

q95:
Maak plots van NAM cq. BP projecten. De plots zijn nodig omdat binnenkort voorlichting wordt gegeven aan het KNB (notarissen) over voornoemde projecten.

Translation: make plots from the NAM and BP projects. These plots are needed to inform the KNB (notary) about these projects.

First some background information (taken from [11]). 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 due to some reason. An important type of legal notification is related to pipelines usually below the surface; see Figure 9.

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

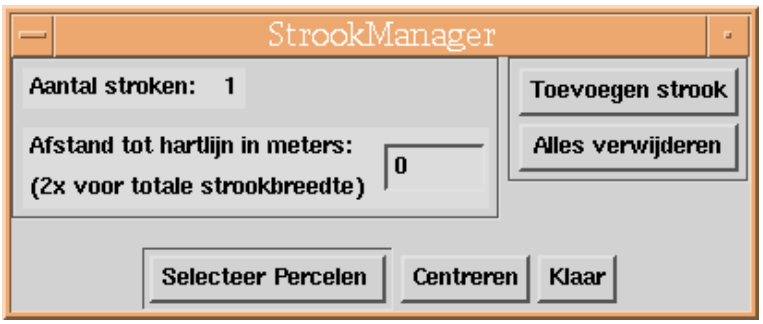

Figure 10: Starting the selection of parcels crossed by a pipeline in the front-end part of the query tool.

registration. It has to be done in an official manner described by law: a dead 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 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 very easy to forget a few parcels when trying 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 very 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 because they are 'anonymous'. It could very well be the case that another utility company has a pipeline crossing the same parcels.

In order 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 query tool was used to select the parcels, which have these 'anonymous' legal notifications (these are of types BP, BG or OG). Using the list of selected parcels the paper deeds are now 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. 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. The NAM delivered a digital version of their pipelines to the Cadastre, which were

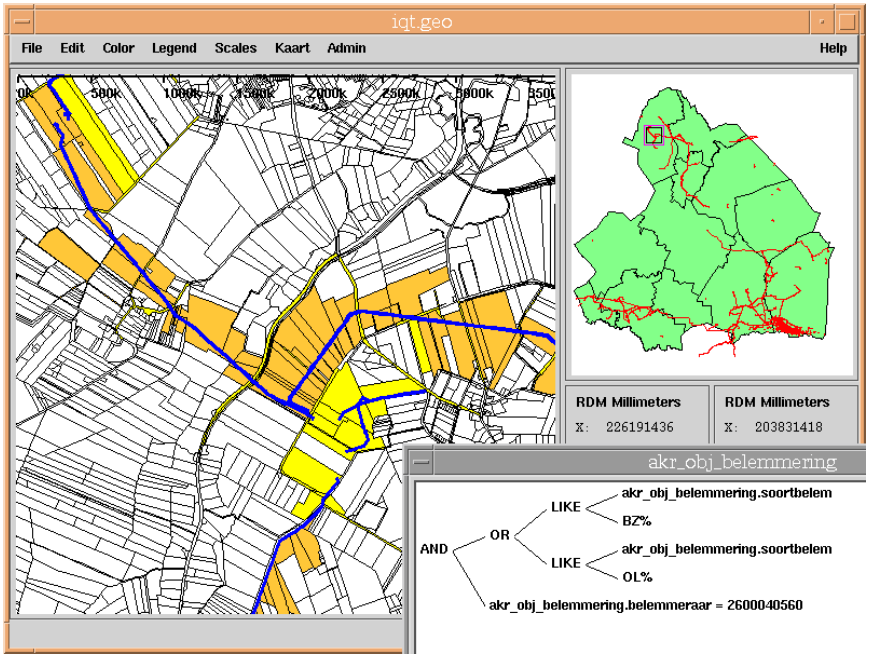

Figure 11: Parcels with a NAM legal notification of type OL or BZ.

then entered into the query tool database and confronted with the parcels; see Figure 9. This was not a simple query in the database, because the geographic data model of the Cadastre is based on topology [9]. Within a relational database an overlap or cross operation based on parcels modelled with topology is impossible. Therefor this operation was implemented in the interface (front-end) part of the query tool; see the inset window of Figure 10.

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 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 crossed by a pipeline, without a legal notification and which are clearly not road parcels (so also without a 'permit'). 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 13. In the project of quality improvement of legal notifications and the pilot project with the NAM, the query tool turned out to be very useful. As described, the 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 indeed the results are correct; e.g. there are no more anonymous legal notifications. One final remark: there are many more types

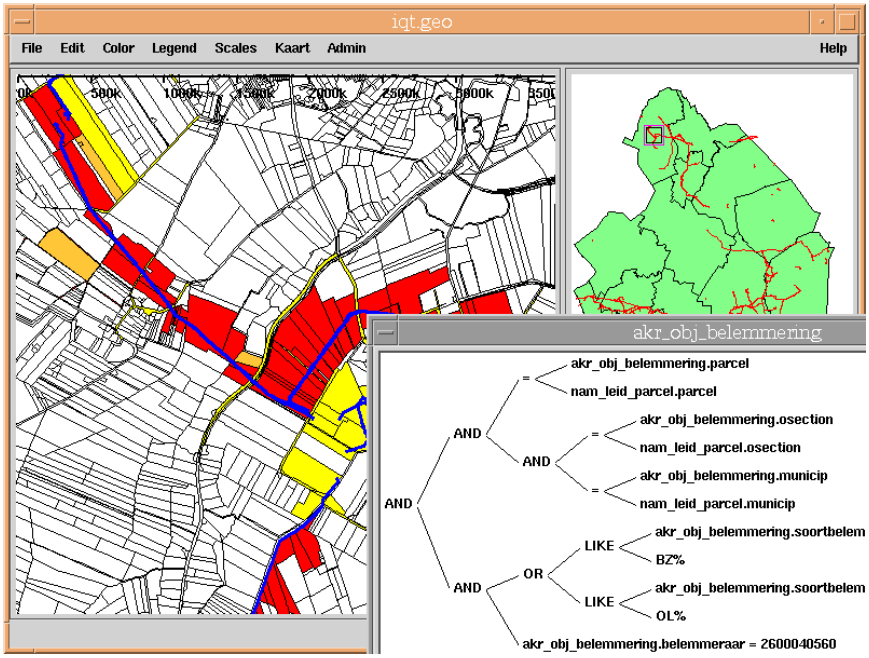

Figure 12: Parcels, crossed by a NAM pipeline and also with a NAM legal notification of type OL or BZ.

of legal notifications than the ones mentioned in this section. These were also quality improved, but are not discussed in this section.

The research tools applied in the work described above did consist of a mixture of interactive query tool usage, shell/SQL scripts, and in this case an external tool to convert data. The main script is `convert.ing`, which converts the three input data sets from the

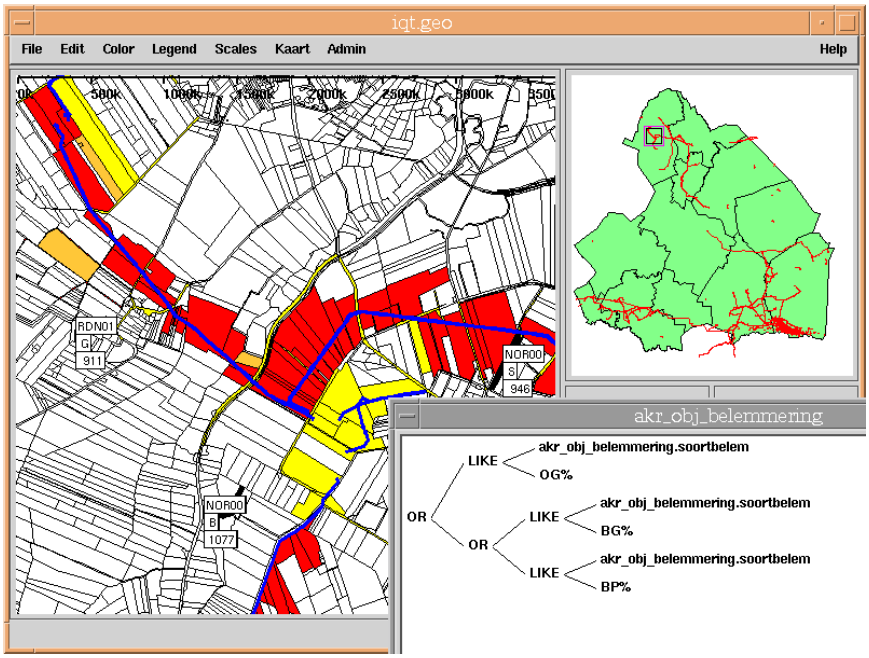

Figure 13: Parcels with legal notification of type BP, BG or OG are marked with a label.

NAM: pipelines (into the table nam_leid), locations (in table nam_lokatie), and access roads (in table nam_toeg_weg). Only the pipelines (nam_leid) will be discussed in the remainder of this section. The other tables are treated in the same way.

In the appendix A it is indicated, which problems did occur during the conversion of the NAM data sets (in Dutch). The MapInfo files (mif/mid) did give the best results during the conversion with the PGS tool mif2ic (performed on a Sun workstation). Besides the mif2ic, two other PGS tools are used to process the NAM data before and during loading the database: split and copyrel. Two scripts are created convert.ing and convert_post.ing. Between these two scripts some interactive work with the query tool has to be done.

```
#! /bin/sh

DB=${1:-assen}

sql $DB << EOB

drop table nam_leid\p\g
create  table nam_leid(
oid object_key with system_maintained,
geo_polyline iline(50),
str_geo_bbox varchar(60),
geo_bbox ibox,
TAG varchar(60))\p\g

EOB

# after correcting the mif files:

$GEOLOCALHOME/convtools/bin/mif2ic -p50 -i1000 data/pipmif
$GEOLOCALHOME/analyses-d/split -l 1 50 -s 1000 < data/pipmif.icd |\
  $GEOHOME/ingres/copyrel $DB nam_leid \
  -q geo_polyline -q str_geo_bbox -q TAG

sql $DB << EOB2

update nam_leid set geo_bbox=bbox(geo_polyline);\p\g
modify nam_leid to heap with compression = (nokey, data);\p\g

drop index nam_leid_idx;\p\g
create index nam_leid_idx
on nam_leid(geo_bbox) with structure=rtree,
range=((-25000000,275000000),(325000000,625000000));\p\g

delete from select_stroken;\p\g
insert into select_stroken (geo_polyline)
select geo_polyline FROM nam_leid;\p\g
EOB2
```

The script first creates a table nam_leid to store the pipelines from the NAM with at most 50 points per polyline. The mif/mid files are converted (mif2ic), split to 50 point and scaled from meter to millimeters (split) and finally copied in the Ingres database (copyrel). In the database, the geo_bbox is set based on the value of the geo_polyline. The data in the table is compressed in order to store the variable length polyline efficiently. An spatial index nam_leid_idx is created on the geo_bbox attribute of the nam_leid table.

The `select_stroken` table of the query tool interface is filled with the pipelines of the NAM with the `insert` SQL statement. Using the 'StrookManager' of the query tool interface, the parcels crossed by a pipeline are selected. Note that this has to be done in the query tool interface outside the database, because SQL cannot form the polygonal loops for the parcels. The result is stored in the query tool interface table `select_parcel` and the script `convert_post.ing` is used to save this into a special table: `nam_leid_parcel` for the pipelines (`nam_toeg_weg_parcel` and `nam_lokatie_parcel` for the other data).

```
#! /bin/sh

DB=${1:-assen}
NAME=${2:-nam_leid}

sql $DB << EOB

drop table ${NAME}_parcel \p\g
create table ${NAME}_parcel as
select distinct ogroup,object_id,slc,classif,location,geo_color,z,
d_location,rotangle,accu_cd,oarea,geo_bbox,object_dt,tmin,tmax,sel_cd,
source,quality,vis_cd,akr_area,municip,osection,sheet,parcel,pp_i_ltr,
pp_i_nr,l_num,line_id1,line_id2,x_akr_objectnummer FROM select_parcel\p\g

delete from geo_dyninfo where relname = '${NAME}_parcel' \p\g
insert into geo_dyninfo (relname, relattr, bboxattr, the_oid,
    dynfunc, dynfile, is_bin, info, cleanupfunc)
values('${NAME}_parcel',
'object_id,l_num,line_id1,line_id2,geo_bbox,"lki_boundary","lki_parcelover","0"',
'geo_bbox', '', 'make_tpgn2_shape', '$GEOHOME/dynamic/NewTopolShapes.o', 'f',
'lki_boundary,object_id,geo_polyline,geo_bbox,fr_line_id,lr_line_id,2,interp_cd',
't3pgn2_clean')\p\g

delete from geo_oid where relname = '${NAME}_parcel' \p\g
insert into geo_oid (relname, relkey) values
('${NAME}_parcel', 'ogroup,object_id,tmax') \p\g

EOB
```

The `geo_oid` and the `geo_dyninfo` table are filled to make the new table easy accessible in the query tool interface.

6 Exchange of agricultural parcels

Question from Guus van der Brink (Region East, via Annemarie van Gelder, ITS):

q43:
In de provincie Overijssel is een Stichting Stimuland die grond wil ruilen van eigenaren van agrarische bedrijven, zodat de bedrijfseigenaren hun grond zo dicht mogelijk bij huis hebben liggen. Concreet is hun vraag: "Kan het Kadaster 1000 adressen leveren van agrarische bedrijven die grond op enige afstand van het bedrijf hebben liggen?"

Translation: In the province of Overijssel an organization Stimuland wants to find potential parcels owned by farmers which may be used for land exchange (lots at a large distance from the farm). The specific question is: 'Can the Cadastre provide 1000 addresses of farms, which own land at a certain distance?'

Again, the research involved interactive use of the query tool and the development of a SQL/shell script `boeren_gr.ing`. Below the script is shown and explained. One of the tricky parts is that not only farmers subjects are considered but also married couples and groups of subjects. That is, if one partner has a suitable piece of land then this may be exchanged (even if the actual owner of the farm is somebody else). Further, a member of a group can also be married. These subjects are also taken into account. In theory the married partner could also be a member of another group, these group members are the 'last' to be considered as relevant.

```
#! /bin/sh
set -v

DB=${1:-iqt_prd1_0100}

sql -s $DB << EOF
set nojournaling;\p\g
set autocommit on;\p\g
```

First select the distinct ground parcels are related subjects in the province Overijssel from the nationwide database based on the range of subject id's in the attribute `gerechtigde` into the table `st_sel`:

```
drop table st_sel;\p\g
create table st_sel as
select distinct x_akr_objectnummer,gerechtigde,soort_recht,aandeel
from mo_recht
where gerechtigde>='7100000000' and gerechtigde<'7600000000' and pp_i_ltr='G'
with nojournaling;\p\g
```

In AKR a group can have rights. Using the table `mo_groepsrelatie`, where the `subject_id` is the id of the group, these rights are 'translated' to the members of the group. The group members are indicated in the table `mo_groepsrelatie` with the strange attribute name `groep`. Using a join between the tables `st_sel` and `mo_groepsrealatie`, those group subjects (`gerechtigde`) are replaced by the members of that group (`groep`) in the table `st_sel_groep`:

```
drop table st_sel_groep;\p\g
create table st_sel_groep as
select distinct x_akr_objectnummer,groep,soort_recht,st_sel.aandeel
```

```
from st_sel,mo_groepsrelatie
where st_sel.gerechtigde=mo_groepsrelatie.subject_id
with nojournaling;\p\g
```

The join in the previous query is only successful in case the `gerechtigde` is a group subject. Therefore, subjects id's which are not representing a group do not participate in the join and will not be in the table `st_sel_groep`. These are directly obtained from the original table `st_sel` and using an SQL `union` statement these two sets are combined into `st_sel2`:

```
drop table st_sel2;\p\g
create table st_sel2 as
select * from st_sel
union all
select * from st_sel_groep
with nojournaling;\p\g
```

In a similar two-step way the partners of married subjects are also added. For this purpose the table `mo_huwelijksrelatie` is joined with the obtained result until now in the table `st_sel2` into the table `st_sel_huw`:

```
drop table st_sel_huw;\p\g
create table st_sel_huw as
select distinct x_akr_objectnummer,huwelijksrelatie,soort_recht,aandeel
from st_sel2,mo_huwelijksrelatie
where st_sel2.gerechtigde=mo_huwelijksrelatie.subject_id
with nojournaling;\p\g
```

Again, these partners (stored in `st_sel_huw`) are added to the subjects (original and 'translated' groups in table `st_sel2`) using the SQL `union` command. The result is stored in the table `st_sel3`:

```
drop table st_sel3;\p\g
create table st_sel3 as
select * from st_sel2
union all
select * from st_sel_huw
with nojournaling;\p\g
```

It is possible that a certain combination of objects, subjects, type of right (`soort_recht`) and share (`aandeel`) occurs more than one time in the table. Therefore a `select distinct` is done to removes these doubles and the result is stored into the table `st_sel`:

```
drop table st_sel;\p\g
create table st_sel as
select distinct x_akr_objectnummer,gerechtigde,soort_recht,aandeel
from st_sel3
with nojournaling;\p\g
```

The number of rights (and related objects) is counted per subject via the `group by gerechtigde` using an aggregate query to create the table `st_sel_aantal`:

```
drop table st_sel_aantal;\p\g
create table st_sel_aantal as
select gerechtigde, aantal=count(*)
from st_sel
group by gerechtigde
with nojournaling;\p\g
```

Only subjects with more than one right (object) are candidates for exchanging land, so the others are dropped. The subset of `st_sel` is stored into the table `st_sel_meer`:

```
drop table st_sel_meer;\p\g
create table st_sel_meer as
select x_akr_objectnummer,st_sel.gerechtigde,soort_recht,aandeel,aantal
from st_sel, st_sel_aantal
where st_sel.gerechtigde=st_sel_aantal.gerechtigde and aantal > 1
with nojournaling, structure=btree, key=(x_akr_objectnummer);\p\g
```

A join on the attribute `x_akr_objectnumber` with the table `mo_object` extends the table `st_sel_meer` with a location attribute. This location is stored in millimeters and the function `ipoint` is used to obtain the proper type. The results are stored in `st_loc`:

```
drop table st_loc;\p\g
create table st_loc as
select st_sel_meer.x_akr_objectnummer,
       gerechtigde,soort_recht,aandeel,aantal,
       loc_akr=ipoint(x*1000,y*1000),soort_cult,grootte
from st_sel_meer, mo_object
where st_sel_meer.x_akr_objectnummer=mo_object.x_akr_objectnummer
with nojournaling, structure=btree, key=(x_akr_objectnummer);\p\g
```

In a similar way the *object* address is added from the table `mo_objectadres` to the table `st_loc`. This results in a table called `st_o`:

```
drop table st_o;\p\g
create table st_o as
select st_loc.x_akr_objectnummer,
       gerechtigde,soort_recht,aandeel,aantal,loc_akr,soort_cult,grootte,
       o_woonplaats=woonplaats,o_straatnaam=straatnaam,
       o_aan_huis_nr=aan_huis_nr,o_huisnummer=huisnummer,
       o_huisletter=huisletter,o_huisnr_toev=huisnr_toev
from st_loc, mo_objectadres
where st_loc.x_akr_objectnummer=mo_objectadres.x_akr_objectnummer
with nojournaling, structure=btree, key=(x_akr_objectnummer);\p\g
```

The subject address is added by joining on the attribute `subject_id` (`gerechtigde`) the tables `st_o` and `mo_subject`. Note that the addresses for natural and non natural persons are obtained in a different way. For non natural person two alternatives are possible: normal address or post address (depending on which one is filled). The result is stored in the table `st_os`:

```
drop table st_os;\p\g
create table st_os as
select st_o.x_akr_objectnummer,gerechtigde,soort_recht,aandeel,aantal,
       loc_akr,soort_cult,grootte,o_woonplaats,o_straatnaam,
       o_aan_huis_nr,o_huisnummer, o_huisletter,o_huisnr_toev,
       s_woonplaats=woonplaats,s_straatnaam=straatnaam,
       s_aan_huis_nr=aanduiding_huisnummer,s_huisnummer=huisnummer,
       s_huisletter=huisletter,s_huisnr_toev=huisnummertoevoeging
from st_o, mo_subject
where st_o.gerechtigde=mo_subject.subject_id and nat_pers_code <> ' '
union
select st_o.x_akr_objectnummer,gerechtigde,soort_recht,aandeel,aantal,
       loc_akr,soort_cult,grootte,o_woonplaats,o_straatnaam,
       o_aan_huis_nr,o_huisnummer,o_huisletter,o_huisnr_toev,
```

```
       s_woonplaats=post_woonplaats,s_straatnaam=post_straatnaam,
       s_aan_huis_nr=post_aanduiding_huisnummer,s_huisnummer=post_huisnummer,
       s_huisletter=post_huisletter,s_huisnr_toev=post_huisnummertoevoeging
from st_o, mo_subject
where st_o.gerechtigde=mo_subject.subject_id and nat_pers_code =' '
       and post_straatnaam<> '                          '
union
select st_o.x_akr_objectnummer,gerechtigde,soort_recht,aandeel,aantal,
       loc_akr,soort_cult,grootte,o_woonplaats,o_straatnaam,
       o_aan_huis_nr,o_huisnummer,o_huisletter,o_huisnr_toev,
       s_woonplaats=woonplaats,s_straatnaam=straatnaam,
       s_aan_huis_nr=aanduiding_huisnummer,s_huisnummer=huisnummer,
       s_huisletter=huisletter,s_huisnr_toev=huisnummertoevoeging
from st_o, mo_subject
where st_o.gerechtigde=mo_subject.subject_id and nat_pers_code =' '
       and post_straatnaam= '                           '
with nojournaling, structure=btree, key=(x_akr_objectnummer);\p\g
```

The farm parcel has the same address for both the object and subject address. This subset of the table `st_os` is called `st_bedrijfsperceel`:

```
drop table st_bedrijfsperceel;\p\g
create table st_bedrijfsperceel as
select x_akr_objectnummer,gerechtigde,soort_recht,aandeel,aantal,
       loc_akr,soort_cult,grootte,s_woonplaats,s_straatnaam,
       s_aan_huis_nr,s_huisnummer,s_huisletter,s_huisnr_toev
from st_os
where  o_woonplaats=s_woonplaats and o_straatnaam=s_straatnaam and
       o_aan_huis_nr=s_aan_huis_nr and
       char(int4(o_huisnummer))=s_huisnummer and
       o_huisletter=s_huisletter and o_huisnr_toev=s_huisnr_toev
with nojournaling, structure=btree, key=(x_akr_objectnummer);\p\g
```

The distances between the farm parcel and the other parcels of the same subject (`gerechtigde`) are computed (join of tables `st_bedrijfsperceel` and `st_loc`). Also a (poly)line is created between the farm and the other parcel (`geo_polyline`) and a fake bounding box of this polyline is computed (`geo_bbox`). Whenever a farm is visible in the query tool the line to the other parcel can be drawn. This results in the table `st_afst_bedrijf`:

```
drop table st_afst_bedrijf;\p\g
create table st_afst_bedrijf as
select
       b_x_akr_objectnummer=b.x_akr_objectnummer,b_gerechtigde=b.gerechtigde,
       b_soort_recht=b.soort_recht,b_aandeel=b.aandeel,b_aantal=b.aantal,
       b_loc_akr=b.loc_akr,b_soort_cult=b.soort_cult,b_grootte=b.grootte,
       s_woonplaats,s_straatnaam,
       s_aan_huis_nr,s_huisnummer,s_huisletter,s_huisnr_toev,
       a_x_akr_objectnummer=a.x_akr_objectnummer,
       a_soort_recht=a.soort_recht,a_aandeel=a.aandeel,a_aantal=a.aantal,
       a_loc_akr=a.loc_akr, a_soort_cult=a.soort_cult,a_grootte=a.grootte,
       afstand=distance(b.loc_akr,a.loc_akr),
       geo_polyline=iline('('+char(b.loc_akr)+','+char(a.loc_akr)+')'),
       geo_bbox=ibox('('+char(b.loc_akr)+',('+
             char(point_x(b.loc_akr))+','+char(point_y(b.loc_akr)+1)+'))')
from st_bedrijfsperceel b, st_loc a
where  b.gerechtigde = a.gerechtigde and
       b.x_akr_objectnummer <> a.x_akr_objectnummer
```

```
with nojournaling, structure=btree, key=(b_x_akr_objectnummer);\p\g
```

A character variant `st_afst_bedrijf_char` for the export of spatial types:

```
drop table st_afst_bedrijf_char;\p\g
create table st_afst_bedrijf_char as
select
       b_x_akr_objectnummer,b_gerechtigde,
       b_soort_recht,b_aandeel,b_aantal,
       char(char(b_loc_akr),24),b_soort_cult,b_grootte,
       s_woonplaats,s_straatnaam,
       s_aan_huis_nr,s_huisnummer,s_huisletter,s_huisnr_toev,
       a_x_akr_objectnummer,
       a_soort_recht,a_aandeel,a_aantal,
       char(char(a_loc_akr),24), a_soort_cult,a_grootte,
       afstand
from st_afst_bedrijf
with nojournaling, structure=btree, key=(b_x_akr_objectnummer);\p\g
```

Define a spatial index `xst_afst_bedrijf_0` for efficient access and some views for visualizing the farms (`st_afst_bedrijf_col_vw`) and links to distant parcels (`st_afst_bedrijf_loc_vw`). Further, views for obtaining only farms at distances larger than respectively 5 and 10 km are defined below:

```
drop index xst_afst_bedrijf_0;\p\g
create index xst_afst_bedrijf_0 on st_afst_bedrijf(geo_bbox)
with structure=rtree,range=((-25000000,275000000),(325000000,625000000));\p\g

drop view st_afst_bedrijf_col_vw;\p\g
create view st_afst_bedrijf_col_vw as
select oid=tid,b_x_akr_objectnummer, b_gerechtigde,
       b_soort_recht, b_aandeel, b_aantal, b_loc_akr, b_soort_cult,
       b_grootte, s_woonplaats, s_straatnaam, s_aan_huis_nr, s_huisnummer,
       s_huisletter, s_huisnr_toev, a_x_akr_objectnummer, a_soort_recht,
       a_aandeel, a_aantal, a_loc_akr, a_soort_cult, a_grootte,
       afstand, geo_polyline, geo_bbox,
       geo_color=4+mod(int4(right(b_gerechtigde,9)),32)
from st_afst_bedrijf;\p\g

drop view st_afst_bedrijf_loc_vw;\p\g
create view st_afst_bedrijf_loc_vw as
select oid=tid,b_x_akr_objectnummer, b_gerechtigde,
       b_soort_recht, b_aandeel, b_aantal, geo_loc=b_loc_akr, b_soort_cult,
       b_grootte, s_woonplaats, s_straatnaam, s_aan_huis_nr, s_huisnummer,
       s_huisletter, s_huisnr_toev, a_x_akr_objectnummer, a_soort_recht,
       a_aandeel, a_aantal, a_loc_akr, a_soort_cult, a_grootte,
       afstand, line=geo_polyline, geo_bbox,
       geo_color=4+mod(int4(right(b_gerechtigde,9)),32)
from st_afst_bedrijf;\p\g

drop view st_afst_bedrijf_10vw;\p\g
create view st_afst_bedrijf_10vw as
select distinct b_x_akr_objectnummer, b_gerechtigde
from st_afst_bedrijf_col_vw
where ((st_afst_bedrijf_col_vw.a_grootte > 10000)
   and (st_afst_bedrijf_col_vw.afstand > 10000000));\p\g
select count(*) from st_afst_bedrijf_10vw;\p\g
```

```
drop view st_afst_veld_10vw;\p\g
create view st_afst_veld_10vw as
select distinct b_x_akr_objectnummer, b_gerechtigde, a_x_akr_objectnummer
from st_afst_bedrijf_col_vw
where ((st_afst_bedrijf_col_vw.a_grootte > 10000)
   and (st_afst_bedrijf_col_vw.afstand > 10000000));\p\g
select count(*) from st_afst_veld_10vw;\p\g

drop table st_afst_veld_10infovw;\p\g
create table st_afst_veld_10infovw as
select distinct b_gerechtigde, b_x_akr_objectnummer,
       voorletters, voorvoegsel, gesl_naam, nat_pers_code,
       postcode,s_woonplaats,s_straatnaam,
       s_aan_huis_nr,s_huisnummer,s_huisletter,s_huisnr_toev,
       a_x_akr_objectnummer, a_loc_akr, a_grootte
from st_afst_bedrijf_col_vw, mo_subject
where ((st_afst_bedrijf_col_vw.a_grootte > 10000)
   and (st_afst_bedrijf_col_vw.afstand > 10000000))
   and b_gerechtigde=subject_id and nat_pers_code <> ' ';\p\g

drop table st_afst_veld_10npinfovw;\p\g
create table st_afst_veld_10npinfovw as
select distinct b_gerechtigde, b_x_akr_objectnummer,
       naam_niet_nat, zetel, soort_niet_nat, nat_pers_code,
       postcode,s_woonplaats,s_straatnaam,
       s_aan_huis_nr,s_huisnummer,s_huisletter,s_huisnr_toev,
       a_x_akr_objectnummer, a_loc_akr, a_grootte
from st_afst_bedrijf_col_vw, mo_subject
where ((st_afst_bedrijf_col_vw.a_grootte > 10000)
   and (st_afst_bedrijf_col_vw.afstand > 10000000))
   and b_gerechtigde=subject_id and nat_pers_code = ' ';\p\g

drop view st_afst_bedrijf_5vw;\p\g
create view st_afst_bedrijf_5vw as
select distinct b_x_akr_objectnummer, b_gerechtigde
from st_afst_bedrijf_col_vw
where ((st_afst_bedrijf_col_vw.a_grootte > 10000)
   and (st_afst_bedrijf_col_vw.afstand > 5000000));\p\g
select count(*) from st_afst_bedrijf_5vw;\p\g

drop view st_afst_veld_5vw;\p\g
create view st_afst_veld_5vw as
select distinct b_x_akr_objectnummer, b_gerechtigde, a_x_akr_objectnummer
from st_afst_bedrijf_col_vw
where ((st_afst_bedrijf_col_vw.a_grootte > 10000)
   and (st_afst_bedrijf_col_vw.afstand > 5000000));\p\g
select count(*) from st_afst_veld_5vw;\p\g

EOF
```

The table `st_afst_bedrijf_char` contains the result with all needed information (without filtering). This table is exported to an ASCII file using the Ingres tool `copydb` and the Unix tool `sed` for some postprocessing:

```
copydb -c $DB st_afst_bedrijf_char
cat copy.out |\
  sed 's/varchar(0)/c0/' |\
  sed 's/nl= d1//'  |\
```

```
  sed 's/c0nl,/c0nl/' |\
  sql $DB

cat st_afst_.oos | sed 's/ * / /g' \
 | sed 's/ *$//' > st_afst.dat

exit
```

7 Appendix A: converting NAM data

This appendix is in Dutch (only).

```
# MapInfo Conversie NAM leidingen (pipmif.mid/pipmif.mif):
#
# 1. Er zijn geen zinnige attributen meer in de pipmif.mid
#    file (alleen attribuut 'tag' is aanwezig).
#    Terwijl de vorige keer er iets van 20 attributen waren.
#
# 2. Daarna loopt de conversie niet goed door de lege regel na de
#    beschrijving en sleutelwoord 'Data' en voor het eerste Pline-record.
#    De melding was 'Unknown line 9:'
#    Daarom deze lege regel verwijderd uit mif file.
#
# 3. Het blijkt dat in somminge leidingen er coordinaten dubbel zitten:
#
# (228606390,569100690) is dubbel! -> knip 1
# (255703250,520140610) is dubbel! -> dubbel achter elkaar, verwijderd
# (264127740,519195130) is dubbel! -> knip 2
# (251858470,527026070) en (251858600,527026560) zijn dubbel! -> knip 3
# (256625320,521615590) en (256627430,521619210) zijn dubbel. -> knip 4
# In keten zitten vele dubbelen: (255962150,522351010), (255965630,522350970),
# (255967440,522350600), (255968360,522350110), (255968470,522350110),
# (255968790,522349310), (255969140,522348130), (255969770,522346610),
# (255971040,522345390), (255972450,522345650), (255975680,522345570),
# (255979460,522345370), (255980050,522344840), (255982250,522344400),
# (255984340,522344480), (255987300,522344270) zijn dubbel.   -> knip 5
#    Vragen aan de NAM dit goed aan te leveren.
#    Voorlopig dus 5 lijnen genipt (deze achter aan de mif file gezet)
#    en in mid file 5 regels toegevoegd met labels 'kinp-1' tot 'knip-5'
#
# 4. En er lijkt een lijn met foute structuur te zijn:
#    Maar dit kan door conversie komen (mif2ic)
#    Bij nadere analyse blijkt dit een los punt in de file te zijn:
#       Point 257981.7090000000 521001.8000000000
#    Deze ook eruit gegooit!
#
# MapInfo Conversie NAM lokaties (lokmif.mid/lokmif.mif):
#
# 1. Ook hier is er maar een attribuut ('tag') in de mid file.
#
# 2. Daarna loopt de conversie niet goed door de lege regel na de
#    beschrijving en sleutelwoord 'Data' en voor het eerste Pline-record.
#    De melding was 'Unknown line 9:'
#    Daarom deze lege regel verwijderd uit mif file.
#
# 3. Daarna loopt de conversie mis met de melding:
#      Unknown line 16:     Center 226213.8300000000 565591.3500000000
#    Kennelijk betreft dit een bestand met polygonen en centroids.
#    De regels met 'Center' weggegooit (met de hand, wellicht een keer
#    door sed halen: sed -f sedfile lokmif.mif_org).
#
# 4. Aan de output te zien, lijkt het alsof mif2ic voor polygonen
#    (Region ipv Pline) nu wel split (op 50 punten aanroept).
#    Zelfs deisland lijkt erin te zitten!
#    Opletten dat dit later goed door split heengaat.
```

```
#
# 5. Van een polygon (Region) worden eerste en laatste coordinaat herhaald.
#    Dit mag niet in Ingres, daarom in ingres het type kpolygon(50) gebruiken.
#    Door een split van een polygon kunnen er ook niet integer-coord ontstaan.
#
# 6. Gedoe, gedoe: toch maar weer de lokaties zien als iline(50) in Ingres.
#    Daarom uit de lokmif.mif wel de ene enclave verwijderd door het
#    label Region 2 in Region 1 te veranderen van de volgende keten en
#    de tweede keten in volgorde om te draaien en als laatste in de
#    file te zetten (zodat de conversie het niet als enclave ziet).
#    Verder aan de mid-file een extra regel toegevoegd met als label 'PETER'.
#    (eerste coords: (25709510,522298620), (257070630,522324540), ...
#
# MapInfo Conversie NAM toegangsweg (lokwmif.mid/lokwmif.mif):
#
# 1. Geen attributen behalve 'tag'
#
# 2. lege regel eruit
#
# Verder zijn de coordinaten in meters (ipv mm) gegeven. Maar omdat de
# polylines toch nog van 'ic' naar Ingres converteerd moeten worden,
# kan 'split' nog extra misbruikt worden. Een nieuwe optie is aan split
# toegevoegd om te schalen, zie help 'split -h':
# split {l|a|n} polygon_field_nr max_nr_of_points [-g eps] [-s scale] <in >out
#
# Ook een optie om regels in de invoer die met '#' beginnen te skippen
# is handig voor split:
```

References

[1] ASK-OpenIngres. INGRES/Object Magemenent Extention User's Guide, Release 6.5. Technical report, 1994.

[2] Bruce G. Baumgart. A polyhedron representation for computer vision. In *National Computer Conference*, pages 589–596, 1975.

[3] Douglas Comer. The ubiquitous B-Tree. *ACM Computing Surveys*, 11(2):121–137, June 1979.

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

[5] J. A. IJsselstein and A. P. Kap. Het kadastraal perceel: een stevig fundament! *Nederlands Geodetisch Tijdschrift Geodesia*, 37(7/8):343–349, 1995. (In Dutch).

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

[7] Christiaan H.J. Lemmen and Peter J.M. van Oosterom. Efficient and automatic production of periodic updates of cadastral maps. In *JEC'95, Joint European Conference and Exhibition on Geographical Information, The Hague, The Netherlands*, pages 137–142, March 1995.

[8] Professional Geo Systems (PGS). The GEO++ system, version 2.80, Reference manual. Technical report, September 1996.

[9] Peter van Oosterom. Maintaining consistent topology including historical data in a large spatial database. In *Auto-Carto 13*, pages 327–336, April 1997.

[10] Peter van Oosterom and Bart Maessen. Geographic query tool. In *JEC-GI'97, Joint European Conference and Exhibition on Geographical Information, Vienna, Austria*, volume 1, pages 177–186, April 1997.

[11] Peter van Oosterom, Bart Maessen, and Wilko Quak. Querytool: design, implementation and applications. In *ISPRS2000, Amsterdam*, page ?, July 2000.

[12] Peter van Oosterom, Bart Maessen, and Wilko Quak. Spatial, thematic, and temporal views. In *SDH2000, 9th International Symposium on Spatial Data Handling*, page ?, August 2000.

[13] Peter van Oosterom and Tom Vijlbrief. The spatial location code. In *Proceedings of the 7th International Symposium on Spatial Data Handling, Delft, The Netherlands*, August 1996.

[14] T. Vijlbrief and P. van Oosterom. The GEO++ system: An extensible GIS. In *5th SDH*, pages 40–50, August 1992.