Research issues in integrated querying of geometric and thematic cadastral information (2)

Peter van Oosterom

GISt Report No. 4
Delft, December 2000

Summary: *This report describes different types of research oriented applications with respect to querying cadastral information. Most applications require both geometric and thematic/legal aspects of cadastral information. The work described in this report has been carried out within the context of the 'Cadastral query tool' system. The research aspects of the following seven queries will be described in more detail: find all agricultural parcels, that is parcels outside the urban polygons of DLG, find all parcels within a large polygon representing a river given by RWS, find the total length of all parcel boundaries in the Netherlands, analyze the number and concentration of parcels to be surveyed and split ('register 9 akteposten'), check the quality of topology references in the cadastral map, check the quality of legal right notification (code OZ), and find neighbor parcels with similar rights.*

©TU Delft
Faculty of Civil Engineering and Geosciences
Department of Geodesy, Section GIS Technology
Thijsseweg 11, 2629 JA Delft, the Netherlands
http://www.gdmc.nl/

Contents

List of Figures

1 Introduction

In this report the research applications based on applying cadastral geographic data sets and associated administrative legal data in one database are described. The relationship between the parcels on the cadastral map and the administrative data is realized through the nation-wide 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.

Several papers and documents have introduced the query tool functionality and the basic data models for the geometric and administrative data [3, 5, 4, 1]. The query tool is nation-wide 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 data, in which the data can be analyzed, queried and filtered.

This report is the second report on the research aspects of ad hoc cadastral queries. In the previous report [2], four applications (ad hoc queries) were described in more detail: (1) neighbors of parcels owned by the province of North-Holland, (2) selecting parcel with monument, (3) legal notifications due to NAM pipelines, and (4) finding potential agricultural parcels suitable for exchange. In this report seven more applications with research aspects will be described in more detail:

- Find all agricultural parcels, that is parcels outside the urban polygons of DLG (Section 2),
- Find all parcels within a large polygon representing a river given by RWS (Section 3),
- Find the total length of all parcel boundaries in the Netherlands (Section 4),
- Analyze the number and concentration of parcels to be surveyed and split, that is, the 'register 9 akteposten' (Section 5),
- Check the quality of topology references in the cadastral map (Section 6),
- Check the quality of legal right notifications (Section 7), and
- Find neighbor parcels with similar rights (Section 8).

2 Find all agricultural parcels for DLG

Original question from DLG via Klaas van der Hoek (Cadastre, B&M) and later on an additional question via Ernst-Peter Oosterbroek (Cadastre, Land Consolidation):

q76, q99:
DLG (Dienst Landelijk Gebied) wil data van alle landelijke gebieden hebben. Om één of andere reden verloopt het aanleveren van deze hoeveelheid (niet of) moeizaam via de normale route. Daarom het verzoek van B&M: DLG levert polygonen van het interessegebied, kan dan uit de querytool een lijst van perceelnummers worden gehaald, die er binnen vallen?
Na de eerste beantwoording van de vraag, kwam een vervolg vraagbinnen: Er zijn nu nieuwe zeefpolygonen van DLG, daarom graag was/wordt leveren t.o.v. vorige selectie.

Translation: DLG (Ministry of Agriculture, Nature Management and Fisheries, Government Service for Land and Water Management) wants to obtain cadastral data related to all rural regions in the Netherlands. The normal procedure within the Cadastre does not work well to define the product(s) to be delivered. This is probably due to the huge number of involved parcels. Therefore the request from B&M: DLG supplies polygons of their regions of interest and the query tool is used to make a selection and to find all involved parcels.
After answering this query, a second request was made to deliver mutations (changes) to the initial delivery, because DLG had supplied new polygons.

In fact it turned out that DLG did not deliver polygons of their areas of interest, but that they deliver polygons of urban regions (areas not of their interest); see Figure 1. The task is to find all agricultural/rural parcels, that is, parcels outside the urban polygons of DLG. The polygons are delivered in Arc/Info ASCII export format. The polygon boundaries are first converted to a database ASCII table format with the program. `arc3pg.org`. Then this result is prepared for the Ingres database. Polylines are correctly split into polygons with 60 points or less with the program `split`. Splitting may result in non-integer coordinates, which cannot be imported into the standard Ingres type `ipolygon`. Therefore the table `nam_pgon_all` is based on the Kadaster variant of spatial data types, in this case the `kpolygon`. After loading the polygons from all provinces of the Netherlands with the program `copyrel`, the `bbox` of the split polygons is computed and set in the table `nam_pgon_all`. Finally, an rtree index is defined on the `bbox` attribute. The selection of parcels outside the DLG urban polygons was now possible in one step for the whole of the Netherlands. However, is was requested to give the result per province and/or per cadastral office. The conversion and loading has been implemented as described above and can be found in the script `convert.ing`:

```
#! /bin/sh

DB=${1:-iqt_prd1_0100}
PWD=‘pwd‘

# eerst converteren (op sun) van arc/info export file:
# 1. arc/info export files iets aanpassen voor arc3pg programma via sedfile:
#    sed -f sedfile < zeef_zh.e00 > zeef_zh3.e00
#
# 2. converteren van arc/info export files naar tabel file:
#    ../q04_dlg/arc3pg.org -aat < zeef_zh3.e00
```

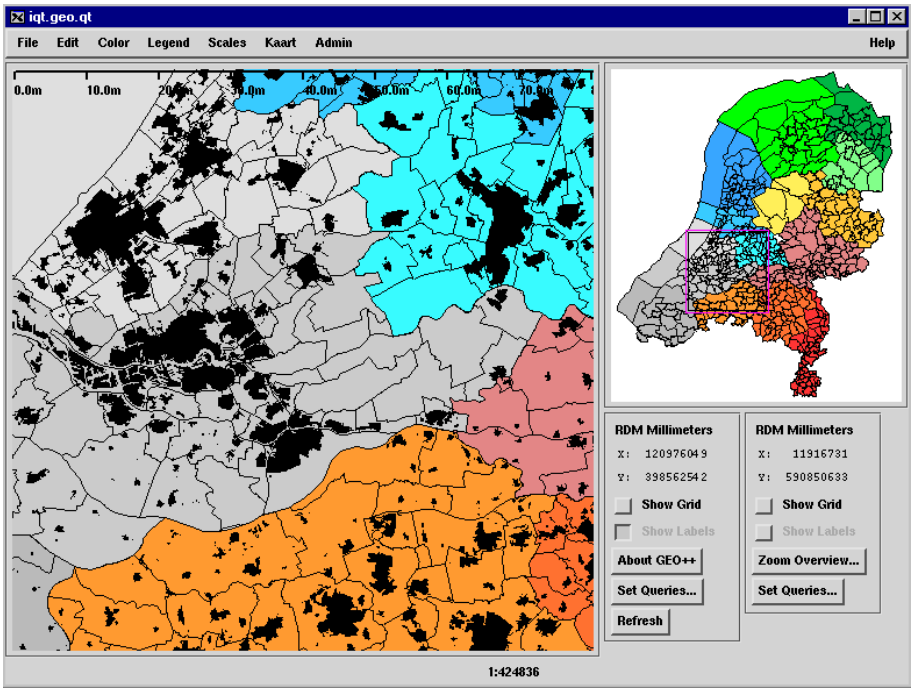

Figure 1: Overview of selection/filter polygons of urban areas.

```
#
# Op Sun of DigUnix:
#
# 3. topologie bouwen (eventueel bij split epslion toevoegen: -g 100):
#    cat pln.copy |\
#    $GEOLOCALHOME/analyses-d/split 1 1 50 |\
#    sed -f sedfile2 |\
#    nl |\
#    $GEOLOCALHOME/analyses-d/ing2geo 4 |\
#    $GEOLOCALHOME/analyses-d/topol -r -b

# first make one table with all polygons:
# (nam_ is used to get tables in extern group)
sql $DB << EOB
set nojournaling\g
/* tabel met zeef polygon maken */
drop table nam_pgon_all\p\g
create  table nam_pgon_all(
oid         object_key  with system_maintained,
junk        varchar(20),
tot_bbox    box,
area        float8,
shape       kpolygon(60),
bbox        ibox)\p\g

drop view nam_pgon_all_vw\p\g
create view nam_pgon_all_vw as
select oid, junk, geo_polygon=char(char(shape),1900), geo_bbox=bbox
from nam_pgon_all\p\g

drop view nam_pgon_all_lvw\p\g
create view nam_pgon_all_lvw as
```

```
select oid, junk, geo_polyline=char(char(shape),1900), geo_bbox=bbox
from nam_pgon_all\p\g
EOB

gunzip -c data/poly_ge.copy.gz data/poly_fr.copy.gz data/poly_nb.copy.gz data/poly_ut.copy.gz data/po
   $GEOLOCALHOME/analyses_d/split a 4 60 |\
   $GEOHOME/ingres/copyrel $DB nam_pgon_all -q junk -q tot_bbox -n area -q shape

sql $DB << EOB2
set nojournaling\g

update nam_pgon_all set bbox = ibox(char(kbbox(shape)))\p\g

drop index idx_nam_pgon_all;\p\g
create index idx_nam_pgon_all on nam_pgon_all(bbox)
with structure=rtree, range=((-25000000,275000000),(325000000,625000000));\p\g

EOB2

# now do processing per office:
./conv_base.ing $DB 230000000 260000000 ad # nh Amsterdam
./conv_base.ing $DB 140000000 200000000 ah # ge Arnhem
./conv_base.ing $DB 260000000 290000000 am # nh Alkmaar
./conv_base.ing $DB  60000000  90000000 as # dr Assen
./conv_base.ing $DB 390000000 420000000 bd # nb Breda
./conv_base.ing $DB 420000000 450000000 eh # nb Eindhoven
./conv_base.ing $DB         0  30000000 gn # gr Groningen
./conv_base.ing $DB 310000000 340000000 gv # zh Zoetermeer
./conv_base.ing $DB  30000000  60000000 la # fr Leeuwarden
./conv_base.ing $DB 120000000 140000000 ll # fl Lelystad
./conv_base.ing $DB 370000000 390000000 md # ze Middelburg
./conv_base.ing $DB 470000000 500000000 rm # li Roermond
./conv_base.ing $DB 340000000 370000000 rt # zh Rotterdam
./conv_base.ing $DB 200000000 230000000 ut # ut Utrecht
./conv_base.ing $DB  90000000 120000000 zl # ov Zwolle

exit 0
# now do processing per province:
./conv_base.ing $DB 60000000 90000000 dr
./conv_base.ing $DB 120000000 140000000 fl
./conv_base.ing $DB 30000000 60000000 fr
./conv_base.ing $DB 1 30000000 gr
./conv_base.ing $DB 470000000 500000000 li
./conv_base.ing $DB 390000000 450000000 nb
./conv_base.ing $DB 230000000 290000000 nh
./conv_base.ing $DB 90000000 120000000 ov
./conv_base.ing $DB 370000000 390000000 ze
./conv_base.ing $DB 200000000 230000000 ut
./conv_base.ing $DB 310000000 370000000 zh
./conv_base.ing $DB 140000000 200000000 ge
```

The last part of the script above calls the script `conv_base.ing`, which performs the actual selection of the parcel per province/office; see Figure 2. The input parameters of this script are the database name, the range of valid object id's of the requested parcels in LKI, and the name of the current province/office selection (indicated with `EXT`). The first steps in this script are identical to the nation-wide loading of DLG polygons. However, now loaded in a specific table for the province/office `nam_pgon_${EXT}`. The

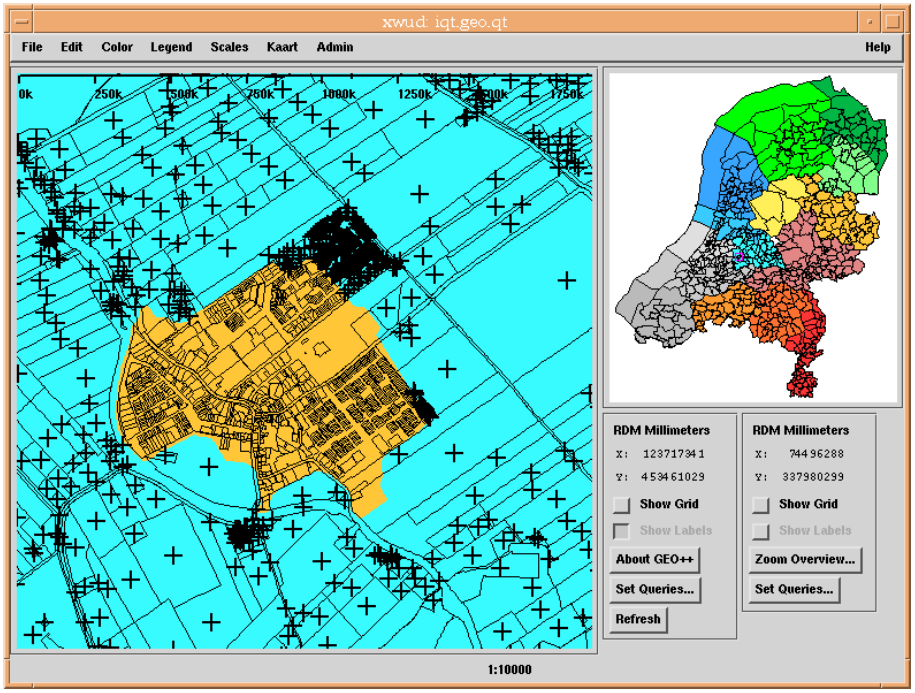

Figure 2: Some details of selected parcels, outside urban polygons.

next step is selection of all parcel reference points for the current province/office in the table `nam_pnt_${EXT}` based on the given range of object id's. In this step the points are converted from the standard `ipoint` into the Kadaster data type `kpoint` in order to be able to perform a point in polygon test later on with the `kpolygon` data. Again an rtree index is put on this table. The actual selection of the parcels is done in three steps:

1. select into table `nam_pntpgon_${EXT}` all parcels with reference point inside the urban polygons. Note that in addition to the box `overlaps` test, also the point in polygon `kinside` test is specified in the where-clause, in order to make an efficient query plan using the rtree index).
2. select into table `nam_pntpgon_${EXT}_zeef` all parcels in the province/office which are not in the previous selection as stored in the table `nam_pntpgon_${EXT}`. This is the requested result.
3. However it is possible, due to splitting large polygons into smaller ones, that a parcel reference point is selected twice. This is a suspicious case and has to be checked by hand.

The last step of the script `conv_base.ing`, which is given below, writes the result to an ASCII output file per province/office:

```
#! /bin/sh

# spliting and loading polygons.
# max number of points is 60 (check if split did work correct,
# no bigger polygons created).
# all tables and views start with nam_ (in order to get the tables in
# the extern group).

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

```
OID1=${2:-200000000}
OID2=${3:-230000000}
EXT=${4:-ut}
PWD=`pwd`

sql $DB << EOB
set nojournaling\g
/* tabel met zeef polygon maken */
drop table nam_pgon_${EXT}\p\g
create  table nam_pgon_${EXT}(
oid         object_key  with system_maintained,
junk        varchar(20),
tot_bbox    box,
area        float8,
shape       kpolygon(60),
bbox        ibox)\p\g

drop view nam_pgon_${EXT}_vw\p\g
create view nam_pgon_${EXT}_vw as
select oid, junk, geo_polygon=char(char(shape),1900), geo_bbox=bbox
from nam_pgon_${EXT}\p\g

drop view nam_pgon_${EXT}_lvw\p\g
create view nam_pgon_${EXT}_lvw as
select oid, junk, geo_polyline=char(char(shape),1900), geo_bbox=bbox
from nam_pgon_${EXT}\p\g
EOB

gunzip -c data/poly_${EXT}.copy |\
   $GEOLOCALHOME/analyses_d/split a 4 60 |\
   $GEOHOME/ingres/copyrel $DB nam_pgon_${EXT} -q junk -q tot_bbox -n area -q shape

sql $DB << EOB2
set nojournaling\g

update nam_pgon_${EXT} set bbox = ibox(char(kbbox(shape)))\p\g

drop index idx_nam_pgon_${EXT};\p\g
create index idx_nam_pgon_${EXT} on nam_pgon_${EXT}(bbox)
with structure=rtree, range=((-25000000,275000000),(325000000,625000000));\p\g

/* percelen in ${EXT} selecteren uit landelijke database m.b.v. object_id */
drop table nam_pnt_${EXT}\p\g
create table nam_pnt_${EXT} as
select geo_loc= kpoint(char(location)),geo_bbox, x_akr_objectnummer
from lki_parcel
where object_id >= ${OID1} and  object_id < ${OID2} \p\g

modify nam_pnt_${EXT} to btree on x_akr_objectnummer;\p\g

drop index idx_nam_pnt_${EXT};\p\g
create index idx_nam_pnt_${EXT} on nam_pnt_${EXT}(geo_bbox)
with structure=rtree, range=((-25000000,275000000),(325000000,625000000));\p\g

EOB2

sql $DB << EOB3
```

```
set nojournaling\g

/* perceelnummers selecteren die in zeef polygonen vallen */
drop table nam_pntpgon_${EXT}\p\g
create table nam_pntpgon_${EXT} as
select geo_loc,x_akr_objectnummer
from nam_pnt_${EXT}, nam_pgon_${EXT}
where overlaps(nam_pnt_${EXT}.geo_bbox, nam_pgon_${EXT}.bbox)=1 and
      kinside(nam_pnt_${EXT}.geo_loc,nam_pgon_${EXT}.shape) > 0\p\g

modify nam_pntpgon_${EXT} to btree on x_akr_objectnummer;\p\g

/* perceelnummers selecteren die buiten polygonen vallen */
drop table nam_pntpgon_${EXT}_zeef\p\g
create table nam_pntpgon_${EXT}_zeef as
select geo_loc,geo_bbox,oid=x_akr_objectnummer
from nam_pnt_${EXT}
where x_akr_objectnummer not in
      (select x_akr_objectnummer from nam_pntpgon_${EXT})\p\g

modify nam_pntpgon_${EXT}_zeef to btree on oid;\p\g

/* perceelnummers selecteren die dubbel in polygonen vallen */
/* deze zijn verdacht (split) en moeten gecontroleerd worden */
drop table nam_pntpgon_${EXT}_zeef2\p\g
create table nam_pntpgon_${EXT}_zeef2 as
select p1.geo_loc,oid=p1.x_akr_objectnummer
from nam_pntpgon_${EXT} p1
where p1.x_akr_objectnummer in
      (select p2.x_akr_objectnummer
       from nam_pntpgon_${EXT} p2
       where p1.x_akr_objectnummer=p2x_akr_objectnummer
         and p1.tid < p2.tid)\p\g

modify nam_pntpgon_${EXT}_zeef to btree on oid;\p\g

copy nam_pntpgon_${EXT}_zeef (oid=c0nl ) into 'zeef_${EXT}.txt';\p\g
EOB3
```

After the initial delivery of parcel numbers based on the selection with the DLG polygons, a new, more accurate set of DLG polygons was obtained. Basically, the old selection was saved in a subdirectory `old_result` and the same work was done again with the new polygons. In other to find the mutations (new or deleted parcel numbers), the differences per cadastral office between the old and the new selections are obtained with the script `get_diffs.sh`:

```
#! /bin/sh

get_diffs_base.sh ad
get_diffs_base.sh ah
get_diffs_base.sh am
get_diffs_base.sh as
get_diffs_base.sh bd
get_diffs_base.sh eh
get_diffs_base.sh gn
get_diffs_base.sh gv
```

```
get_diffs_base.sh la
get_diffs_base.sh ll
get_diffs_base.sh md
get_diffs_base.sh rm
get_diffs_base.sh rt
get_diffs_base.sh ut
get_diffs_base.sh zl
```

The real work, finding the differences and writing the result into two sets of files `old` and `new`, is done by the script `get_diffs_base.sh`:

```
#! /bin/sh

NAME=${1:-ut}

diff zeef_${NAME}.txt old_result/ > ${NAME}.diff
grep '>'  ${NAME}.diff | sed 's/> //' >  ${NAME}_old.txt
grep '<'  ${NAME}.diff | sed 's/< //' >  ${NAME}_new.txt
```

The DLG polygons are supplied per province and every province is indicated with a two letter abbreviation. There is sometimes small difference between the province and the cadastral office. In most cases this is identical, but in some cases there are two cadastral offices in one province (Noord-Brabant `nb`, Zuid-Holland `zh`, and Noord-Holland `nh`). Also the cadastral offices are indicated with a two letter abbreviation. Logical links from the data files (per province) are made to every cadastral office. Normally one, except for Noord-Brabant, Zuid-Holland, and Noord-Holland (they all have two links) and for Utrecht (zero links, because province name and cadastral office name are the same). The links are created in data subdirectory with the script `make_links`:

```
ln -s poly_nh.copy.gz poly_ad.copy.gz
ln -s poly_nh.copy.gz poly_am.copy.gz
ln -s poly_ge.copy.gz poly_ah.copy.gz
ln -s poly_dr.copy.gz poly_as.copy.gz
ln -s poly_nb.copy.gz poly_bd.copy.gz
ln -s poly_nb.copy.gz poly_eh.copy.gz
ln -s poly_gr.copy.gz poly_gn.copy.gz
ln -s poly_fr.copy.gz poly_la.copy.gz
ln -s poly_fl.copy.gz poly_ll.copy.gz
ln -s poly_ze.copy.gz poly_md.copy.gz
ln -s poly_li.copy.gz poly_rm.copy.gz
ln -s poly_zh.copy.gz poly_rt.copy.gz
ln -s poly_zh.copy.gz poly_gv.copy.gz
ln -s poly_ov.copy.gz poly_zl.copy.gz
```

3 Parcels within a river given by RWS

Original question from Klaas van der Hoek (Cadastre, B&M):

q04: Er is een offerte-aanvraag van RWS (Rijkswaterstaat) bij B&M binnengekomen voor de levering van alle percelen binnen een bepaald rivierengebied. Dit gebied loopt door meerdere provincies en is gespecificeerd door middel van een polygon. Er moet een schatting worden gemaakt hoeveel percelen dit nu betreft langs dit rivieren/uiterwaarden-gebied om een goede offerte te kunnen maken.

Translation: RWS (Ministry of Transport, Public Works and Water Management, Directorate-General of Public Works and Water Management) requested B&M a quotation for a cadastral data delivery for all parcels inside a certain river region; see Figure 3. This river covers several provinces and is specified as one large polygon. An estimation has to be made how many parcels are involved in this river region in order to make a reasonable quotation.

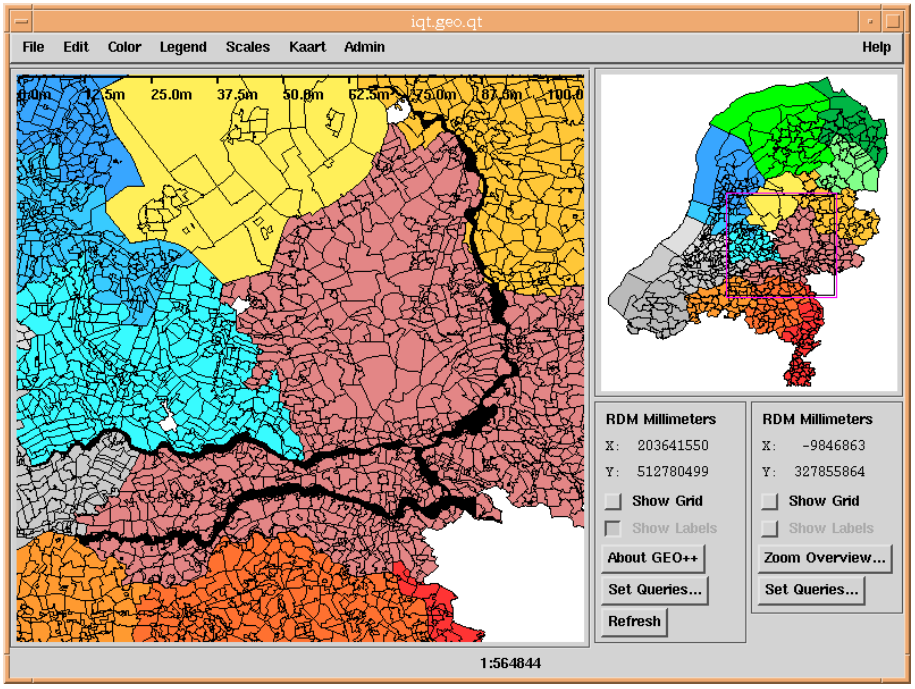

Figure 3: Overview of river polygon (black) defined by over 10,000 points.

As in the previous application (see Section 2) the data was delivered in an Arc/Info ASCII export file. Again, more or less the same steps have been applied to obtain the result: `arc3pg` to convert the data into a database ASCII format, `split` to make the large polygon better manageable and produce smaller polygons, and also the Kadaster spatial data types `kpoint` and `kpolygon` are used as input for the point in polygon processing in the Ingres database. Figure 4 shows the result in the south-east of the province of Utrecht and near to the border with the province of Gelderland. The 'river' parcels are now shown in light gray colors. The whole procedure has been implemented in the script `convert.ing`:

```
#! /bin/sh
```

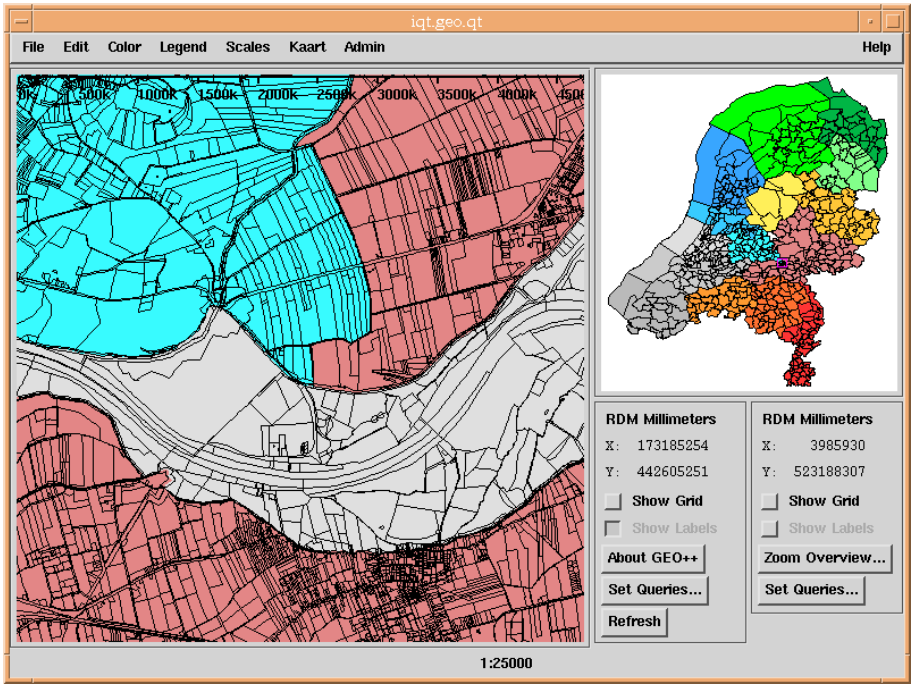

Figure 4: Parcel details near river polygon (light gray) in the south-east of the province of Utrecht (blue) and near to the border with the province of Gelderland (red).

```
DB=${1:-lki_akr}
PWD=‘pwd‘
EPS=1000.0

echo "Current directory is " $PWD

arc3pg < dongrens.e00
rm AI.pq
cat pln.copy |\
   $GEOLOCALHOME/analyses_d/split l 1 50 -g $EPS |\
   sed -f sedfile |\
   nl |\
   $GEOLOCALHOME/analyses_d/ing2geo 4 |\
   $GEOLOCALHOME/analyses_d/topol -r -b

sql $DB << EOB
drop table pgon\p\g
create  table pgon(
oid         object_key  with system_maintained,
junk        varchar(20),
tot_bbox    box,
area        float8,
shape       kpolygon(50),
bbox        box)\p\g

drop view pgon_vw\p\g
create view pgon_vw as
select oid, junk, geo_polygon=char(char(shape),1500), geo_bbox=bbox
from pgon\p\g

drop view pgon_lvw\p\g
```

```
create view pgon_lvw as
select oid, junk, geo_polyline=char(char(shape),1500), geo_bbox=bbox
from pgon\p\g
EOB

cat poly.copy |\
   $GEOLOCALHOME/analyses_d/split a 4 50 -g $EPS |\
   $GEOHOME/ingres/copyrel $DB pgon -q junk -q tot_bbox -n area -q shape

sql $DB << EOB2
set nologging\p\g
set nojournaling\g

update pgon set bbox = box(char(kbbox(shape)))\p\g

drop table pnt\p\g
create table pnt as
select oid,geo_loc=kpoint(char(location)),akr_area,oarea
from hist_parcel_beg\p\g

EOB2

sql $DB << EOB3
drop table pntpgon_tv\p\g
create table pntpgon_tv as
select geo_loc=point(char(pnt.geo_loc)), pnt.akr_area,
       pnt.oarea,kin=kinside(pnt.geo_loc,pgon.shape)
from pnt, pgon
where kinside(pnt.geo_loc,pgon.shape) > 0\p\g

delete from geo_dyninfo where relname = 'pntpgon_tv' \p\g
insert into geo_dyninfo (relname, relattr, bboxattr, the_oid,
  dynfunc, dynfile, is_bin, info, cleanupfunc)
  values('pntpgon_tv', 'geo_loc', '', 'geo_loc',
  'make_pnt2_shape', '$GEOHOME/dynamic/Geo2Shapes.o', 'f',
  '','')
 ) \g

drop index _pntpgon_tv;\p\g
create index _pntpgon_tv
on pntpgon_tv(geo_loc)
with structure=rtree,
     range=((-25000000,275000000),(325000000,625000000));\p\g

select aantal=count(*), opp_akr=sum(akr_area), opp_lki=sum(oarea), kin
from pntpgon_tv
where kin >0
group by kin\p\g

select distinct p1.geo_loc
from pntpgon_tv p1, pntpgon_tv p2
where p1.geo_loc = p2.geo_loc\g
EOB3

# rm pgn.copy face.copy edge.copy
```

4 Total length of all parcel boundaries

q109:
Bepaal de totale lengte van alle perceelgrenzen in Nederland. Deze vraag is bij het Kadaster binnengekomen in het kader van een promotie-onderzoek. Naast totale lengte, ook de aantallen grenzen bepalen en dit alles uitsplitsten per type grens (perceel, sectie, gemeente, vestiging, land).

Translation: Determine the total length of all parcel boundaries in the Netherlands. This question was posed to the Cadastre in the context of a PhD research. Besides the total length, also determine the number of parcel boundaries and separate this per type of boundary: parcel, section, municipality, cadastral office (province) and country.

This is a relatively straight forward question for the nation-wide query tool database. The view `lki_boundary` is used, because it selects exactly one moment in time (which was in this case 1 Oct. 1999). Further, the unit length in LKI is millimeters. Therefore the `linelen` is divided by 1,000,000 to get kilometer units in the view `gr_m2`. To check if this division did not cause numerical problems (rounding off to zero), also a view `gr_m` is defined, which is based on meters (division by 1,000). However, the results were identical. This small application was implemented with the script `lengte.sql`:

```
drop view gr_m\g
create view gr_m as select len=float8(linelen)/1000 from lki_boundary\g
select sum(len) from gr_m\g
select count(*) from gr_m\g

drop view gr_m2\g
create view gr_m2 as select len=float8(linelen)/1000000 from lki_boundary\g
select sum(len) from gr_m2\g
select count(*) from gr_m2\g

drop view gr_m3\g
create view gr_m3 as
  select classif,len=float8(linelen)/1000000 from lki_boundary\g
select sum(len) from gr_m3 group by classif\g
select classif,count(*) from gr_m3 group by classif\g
```

Total length of all parcel boundaries in the Netherlands (on 1 Oct. 1999) was 236,033.040 km (the number of boundaries was 19,664,704), based on querying views `gr_m` and `gr_m2`. View `gr_m3` also selects the boundary classification (type). The results of the aggregation queries `sum` and `count` on view `gr_m3` grouped by `classif` were:

tot.length(km)	classif	number	type
1.405	0	9	?
218623.599	31	18991509	parcel
8315.104	32	471621	section
2000.153	33	84270	cad muncip
4086.230	34	92683	muncip
2714.986	35	16981	cad province
287.156	36	7557	country
3.119	255	68	?
0.077	332	4	?
0.676	334	1	?
0.534	335	1	?

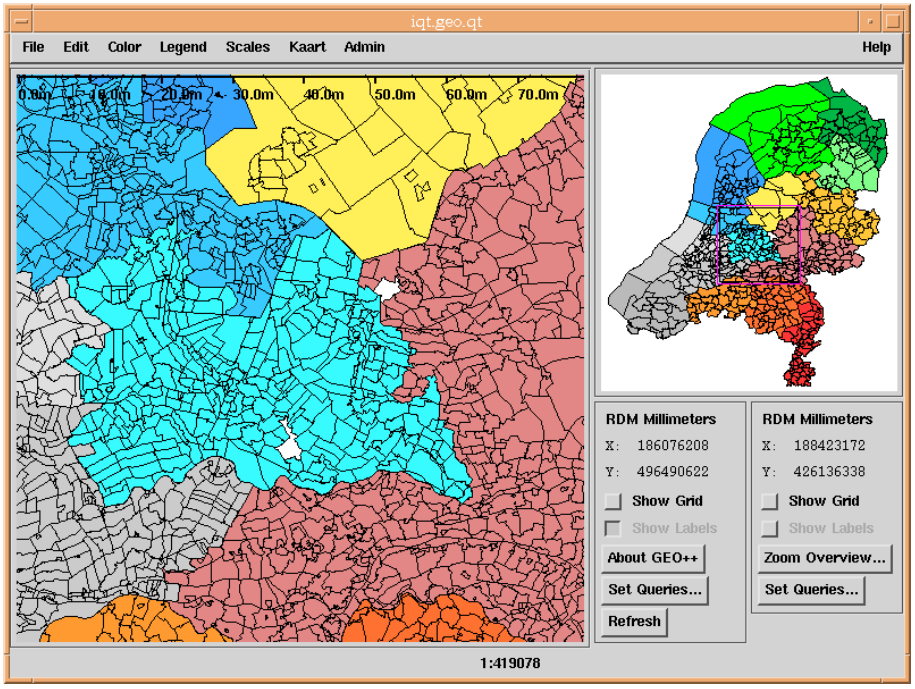

Figure 5: Cadastral sections in detail window and municipalities in overview window.

Strange are the classification codes 0, 255, 332, 334 en 335 given the fact that there has been a selection for the valid topology layer `ogroup=6` and proper date based on a where-clause with `tmin/tmax` specifying '1 Oct. 1999'. Additional investigations turned out that in the production environment everything that has not the value `tmax=0` is considered past history and is therefore not valid anymore. Sometimes this is done by hand to remove certain types of errors, such as correction of classification codes for the whole cadastral office with a few SQL statements. This is quite dangerous as with the query tool, one can look in the past (features that have `tmax != 0`). Figure 5 shows in the detail window the cadastral sections (boundary classification 32) and in the overview window the municipalities (boundary classification 34). The areas are colored with cadastral office both in the detail and overview window.

5 Analyze parcels to be surveyed and split

Original question from Auke Hoekstra (Cadastre, B&M):

q80, q84:
Is het mogelijk het aantal openstaande akteposten (deelpercelen) per perceel te visualiseren (hoe meer hoe donkerder de kleur). Hierdoor worden de urgente gevallen met veel akteposten duidelijk en ook wordt de ruimtelijke verspreiding hiervan in kaart gebracht, zodat er geplanned kan worden om later efficiënt te meten in een bepaald gebied.
Een vervolgvraag hierop: Maak een overzicht van de aantallen register 9 posten per sectie en per kadastrale gemeente (dit zowel in tabelvorm als in kaartvorm). Merk op dat een object vele register 9 posten kan betreffen en dat een register 9 post vele objecten kan betreffen (n:m relatie). Alleen het echte aantal verschillende register 9 posten moet geteld worden. Verder zou het mooi zijn als ook de datum (hoe oud een register 9 post is) meegenomen kan worden.

Translation: Is it possible to visualize the number of planned future surveys and splits per parcel (in Dutch 'register 9 akteposten')? Urgent cases will become clear and also the spatial distribution can be mapped. This will support planning of the actual surveying per region.

The additional question was related to aggregating this information to the cadastral section and municipality level (output requested in both map and tabular form). Note that one object (parcel) can have many 'register 9 akteposten' and that one 'register 9 aktepost' can involve many objects (parcels), that is a n-to-m relationship. Only the real number of 'register 9 akteposten' should be counted. In addition it would be useful if also the date (how old is a certain 'register 9 aktepost') could be involved in this analysis.

The answers to the questions above are both simple and difficult. In the analysis two groupings have been defined. The first one is related to the age (date) of a 'register 9 aktepost'. This definition is stored in the table `date_def`: $<= 6, 7-12, 13-18, 19-24, 25-30, 31-36, >= 37$ months. The second one is related to the number of 'register 9 akteposten' per parcel, or in other words the concentration of 'register 9 akteposten'. This is stored in the table `concen_def`: $1-4, 5-8, >= 9$ 'register 9 akteposten' per parcel. Figure 6 shows the concentration of 'register 9 akteposten' per parcel (based on the table `reg9_aantal`).

The administrative information contains one table `mo_reg_9`, which holds the 'register 9 akteposten'. However, the first thing which is not so easy to determine is the unique identifier of a 'register 9 aktepost'. It turns out that the 'register 9 akteposten' are numbered per cadastral section. Therefore the unique identifier is the combination `munucip`, `osection`, and `reg_9_nr`. This has been checked with the help tables `t1` and `t2`. The script below further shows the SQL statements in the refined analysis `reg9_6.sql`:

```
drop table date_def;\p\g
create table date_def( period char(12), start integer4, last integer4);\p\g
insert into date_def values ('1:   <=6 mnd',199904,199909);\p\g
insert into date_def values ('2:  7-12 mnd',199810,199903);\p\g
insert into date_def values ('3: 13-18 mnd',199804,199809);\p\g
insert into date_def values ('4: 19-24 mnd',199710,199803);\p\g
insert into date_def values ('5: 25-30 mnd',199704,199709);\p\g
insert into date_def values ('6: 31-36 mnd',199610,199703);\p\g
```

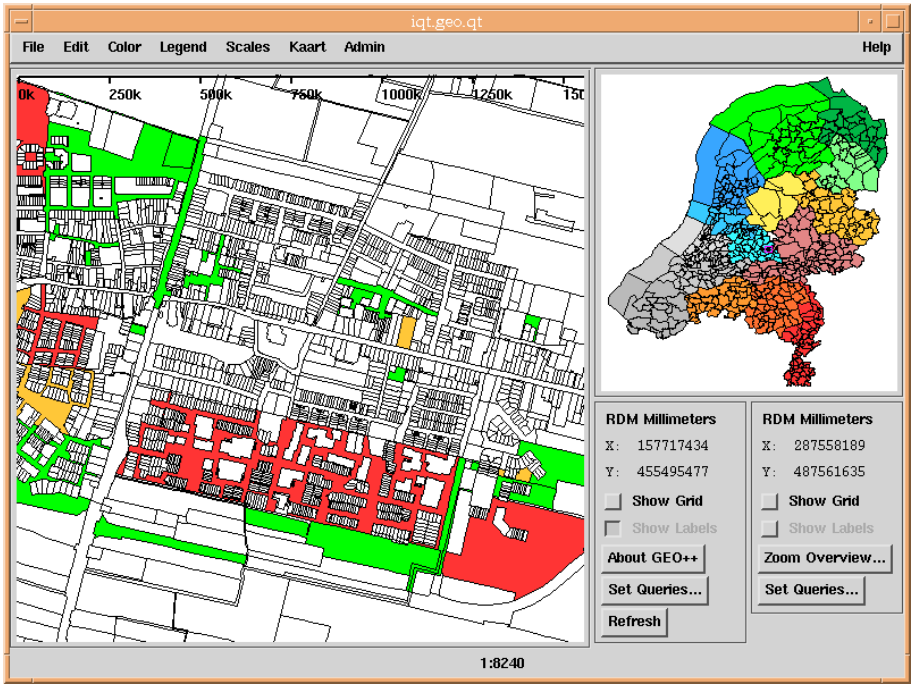

Figure 6: Parcels with more than 0/4/8 'register 9 akteposten' in green/orange/red.

```
insert into date_def values ('7:  >=37 mnd',     0,199609);\p\g

drop table concen_def;\p\g
create table concen_def( concen_groep char(10), start integer4, last integer4);\p\g
insert into concen_def values ('A: 1-4',1,4);\p\g
insert into concen_def values ('B: 5-8',5,8);\p\g
insert into concen_def values ('C: >=9',9,100000);\p\g

/* tellen aantal akteposten:
aangenomen dat unique id aktepost is municip,osection,reg_9_nr */
create table t1 as
select municip, osection, reg_9_nr, aantal=count(*)
from mo_reg_9
group by municip, osection,reg_9_nr
with nojournaling;\p\g
/* antwoord: 177350 */

/* tellen aantal akteposten:
aangenomen dat unique id aktepost is municip,reg_9_nr */
create table t2 as
select municip, reg_9_nr, aantal=count(*)
from mo_reg_9
group by municip, reg_9_nr
with nojournaling;\p\g
/* antwoord: 177329 (dit is te weinig, dus osection hoort bij id aktepost */

/* alleen op 'G' objecten reg9 posten, anders klopt script niet, check */
SELECT pp_i_ltr,count(*) from mo_reg_9 group by pp_i_ltr;\p\g

/* tel per perceel het aantal akteposten */
drop table reg9_aantal;\p\g
create table reg9_aantal as
```

```
select municip, osection, parcel, aantal_post=count(*)
from mo_reg_9
group by municip, osection, parcel
with nojournaling;\p\g
modify reg9_aantal to btree on municip, osection, parcel;\p\g
```

6 Quality check topology cadastral map

Original question from Theo Eskens (Cadastre, ITS/APS):

q83:
In de LKI produktie omgeving zijn topologische problemen geconstateerd. Het blijk soms dat bij een perceelgrens (xfio_boundary) de verwijzingen via de object_id's links en rechts naar hetzelfde perceel wijzen. Vraag komt dit vaker voor? En hoe zit dit met de verwijzingen via perceelnummer links en rechts (i.p.v. object_id)?

Translation: In the LKI production environment some topology reference errors have been observed. It turns out that at a parcel boundary `xfio_boundary` references to the object_is's left and right are the same. Question is how often does this occur? A similar question is posed to find out how often does this occur in the case of the left/right references based on parcel numbers (instead of object_id's)?

The script `lr_error.sql` contains the SQL statements for the requested analysis. Note that always `tmax=0` is specified in order to be sure that the query only affects 'current' data. Further, only the specific groups, which contain the valid parcel topology layer, are used: `ogroup=6` in case of the `xfio_boundary` table and `ogroup=46` in case of the `xfio_parcel` table. Besides counting how often equal left/right references occur, also reported is the last date/time such a new boundary with the same left/right references has been created. This is an error in LKI and should have been corrected. However, there may be some 'old' data, still valid, created with a previous version of LKI. In order to be able to check if the new version of LKI does not introduce new topology reference errors, the last date/time is included. The first two queries check for the same left/right information based on object_id and parcel number respectively. The last four queries check if the left (or right) references based on object_id and parcel number to the parcel table are consistent with each other. Again, reporting the last date/time an error was introduced. This analysis was done on the 1 March 1999 database.

```
/* same parcel to left/right according to parcel number references */
select count(tmin), max(tmin), lkiint2date(max(tmin)) from xfio_boundary
where r_municip=l_municip and r_section=l_section and r_parcel=l_parcel
  and ogroup=6 and tmax=0;\p\g

+------------+------------+------------------+
|col1        |col2        |col3              |
+------------+------------+------------------+
|          12|   300455978|06-05-1999 11:59:38|
+------------+------------+------------------+
(1 row)

/* same parcel to left/right according to object_id references */
select count(tmin), max(tmin), lkiint2date(max(tmin)) from xfio_boundary
where r_obj_id=l_obj_id
  and ogroup=6 and tmax=0;\p\g

+------------+------------+------------------+
|col1        |col2        |col3              |
+------------+------------+------------------+
|         130|   300455978|06-05-1999 11:59:38|
+------------+------------+------------------+
(1 row)
```

```
/* consistency check right parcel reference: the referred parcel based
   on parcel number has a different object_id */
select count(xfio_boundary.tmin),
  max(xfio_boundary.tmin), lkiint2date(max(xfio_boundary.tmin))
from xfio_boundary, xfio_parcel
where xfio_boundary.r_municip=xfio_parcel.municip
  and xfio_boundary.r_section=xfio_parcel.osection
  and xfio_boundary.r_parcel=xfio_parcel.parcel
  and xfio_boundary.r_obj_id<>xfio_parcel.object_id
  and xfio_parcel.ogroup=46 and xfio_parcel.tmax=0
  and xfio_boundary.ogroup=6 and xfio_boundary.tmax=0;\p\g

+-------------+-------------+-------------------+
|col1         |col2         |col3               |
+-------------+-------------+-------------------+
|         1831|    297431265|02-04-1999 11:47:45|
+-------------+-------------+-------------------+
(1 row)

/* consistency check left parcel reference: the referred parcel based
   on parcel number has a different object_id */
select count(xfio_boundary.tmin),
  max(xfio_boundary.tmin), lkiint2date(max(xfio_boundary.tmin))
from xfio_boundary, xfio_parcel
where xfio_boundary.l_municip=xfio_parcel.municip
  and xfio_boundary.l_section=xfio_parcel.osection
  and xfio_boundary.l_parcel=xfio_parcel.parcel
  and xfio_boundary.l_obj_id<>xfio_parcel.object_id
  and xfio_parcel.ogroup=46 and xfio_parcel.tmax=0
  and xfio_boundary.ogroup=6 and xfio_boundary.tmax=0;\p\g

+-------------+-------------+-------------------+
|col1         |col2         |col3               |
+-------------+-------------+-------------------+
|         2054|    297431265|02-04-1999 11:47:45|
+-------------+-------------+-------------------+
(1 row)

/* consistency check right parcel reference: the referred parcel based
   on object_id has a different parcel number */
select count(xfio_boundary.tmin),
  max(xfio_boundary.tmin), lkiint2date(max(xfio_boundary.tmin))
from xfio_boundary, xfio_parcel
where not(xfio_boundary.r_municip=xfio_parcel.municip
  and xfio_boundary.r_section=xfio_parcel.osection
  and xfio_boundary.r_parcel=xfio_parcel.parcel)
  and xfio_boundary.r_obj_id=xfio_parcel.object_id
  and xfio_parcel.ogroup=46 and xfio_parcel.tmax=0
  and xfio_boundary.ogroup=6 and xfio_boundary.tmax=0;\p\g

+-------------+-------------+-------------------+
|col1         |col2         |col3               |
+-------------+-------------+-------------------+
|          327|    297427727|02-04-1999 10:48:47|
+-------------+-------------+-------------------+
(1 row)
```

```
/* consistency check left parcel reference: the referred parcel based
   on object_id has a different parcel number */
select count(xfio_boundary.tmin),
  max(xfio_boundary.tmin), lkiint2date(max(xfio_boundary.tmin))
from xfio_boundary, xfio_parcel
where not(xfio_boundary.l_municip=xfio_parcel.municip
  and xfio_boundary.l_section=xfio_parcel.osection
  and xfio_boundary.l_parcel=xfio_parcel.parcel)
  and xfio_boundary.l_obj_id=xfio_parcel.object_id
  and xfio_parcel.ogroup=46 and xfio_parcel.tmax=0
  and xfio_boundary.ogroup=6 and xfio_boundary.tmax=0;\p\g

+-------------+-------------+-------------------+
|col1         |col2         |col3               |
+-------------+-------------+-------------------+
|          348|    297427727|02-04-1999 10:48:47|
+-------------+-------------+-------------------+
(1 row)
```

It can be concluded that there still was an error in the LKI software on 1 March 1999. Note that the dates are even after 1 March 1999. This is because the actual copy of LKI data can be any date after 1 March 1999 in the query tool database (for AKR the date is exactly 1 March 1999)

7 Quality check legal right notification

Original question from Jans Vrieling (Cadastre, Region North):

q103:
Zoek percelen met zakelijk recht OS en met rechtbelemmering OZ. (Dit had OL moeten zijn, maar is in het verleden fout gegegaan in AKR).

Translation: Find parcels with legal right code 'OS' and legal right notification 'OZ' (this should have been 'OL', but indicates a data error from the past in AKR).

The first selection tries to find all objects (parcels) for which both conditions (legal right code 'OS' and legal right notification 'OZ') are true. This has to be done by joining the tables `akr_recht` (which contains legal right code) and `akr_rechtbelemmering` (which contains legal right notification code). The result is stored in the table `sel_osrecht2` and is valid for the whole of the Netherlands.

The second approach is limited to only the province of Groningen and is based on a two step approach: first a selection on right code 'OS' and the province of Groningen (based on the range of valid object_id in Groningen) from the table `akr_recht` is done and the result is stored in the table `sel_osrecht_la`. Then this table is joined with the table `akr_rechtbelemmering` based on the `x_akr_objectnummer` and the additional condition legal right notification code 'OZ'.

The contents of script `convert.sh`

```
#!/bin/sh

DB=${1:-iqt_prd1_0100}

sql $DB << EOF

drop table sel_osrecht2\p\g
create table sel_osrecht2 as
select distinct
   x_akr_obj=r.x_akr_objectnummer, r.stuk_vest_recht
from akr_recht r, mo_rechtbelemmering b
where (r.soort_recht = 'OS' and b.soort_rechtbelemmering = 'OZ' and
   r.x_akr_objectnummer = b.x_akr_objectnummer)
   with nojournaling;\p\g

grant select on sel_osrecht to public;\p\g

drop table sel_osrecht_la\p\g
create table sel_osrecht_la as
select distinct
   x_akr_obj=r.x_akr_objectnummer, r.stuk_vest_recht
from akr_recht r
where (r.soort_recht = 'OS'
   and r.object_id >= 30000000 and r.object_id < 60000000 )
   with nojournaling;\p\g

drop table sel_osozrecht_la\p\g
create table sel_osozrecht_la as
select distinct
   x_akr_obj=r.x_akr_obj, r.stuk_vest_recht
```

```
from sel_osrecht_la r, mo_rechtbelemmering b
where b.soort_rechtbelemmering = 'OZ' and
   r.x_akr_obj = b.x_akr_objectnummer
   with nojournaling;\p\g

grant select on sel_osrecht_la to public;\p\g
grant select on sel_osozrecht_la to public;\p\g
EOF
```

The result was 910 objects (parcels) in Groningen. This information was sent back to Groningen together with the identification of the deed which was the basis for establishing this right.

8 Find neighbor parcels with similar rights

Original question from Jans Vrieling (Cadastre Regio, North):

q96:
In de provincie Groningen zijn in het verleden de zogenaamde stadsmeierrechten afgekocht. Deze rechten hadden in het verleden tot gevolg dat er veel versnippering in de perceelsvorming optrad, waardoor veel kleine percelen zijn ontstaan. Hierdoor komen thans vaak fouten in aktes voor, omdat kleine perceeltjes vergeten worden te noemen. Het idee is nu om dit soort percelen zo veel mogelijk te gaan verenigen. Hierbij moet dan aan de verenigingsvoorwaarden worden voldaan.

Selecteer binnen een gemeente de subjecten die in meer dan 1 perceel recht hebben. Selecteer vervolgens de percelen van een subject, waarop dezelfde rechten rusten en die aan elkaar grenzen (buren zijn). Het betreft de volgende kadastrale gemeenten: HGZ00 (Hoogezand), HRN01 (Haren), OWD00 (Onstwedde), PKL01 (Nieuwe Pekela), PKL02 (Oude Pekela), VDM00 (Veendam), VWD02 (Vlagtwedde) en WDV02 (Wildervank), WDV03 (Wildervank DE).

Translation: In the province of Groningen, due to some reasons related to a certain type of right 'stadsmeierrecht', fragmentation has occurred in the past, which resulted in many small parcels. Currently, the notary often forget to mention these small parcels in the deeds. This is an error which may be avoided if the Cadastre re-unites neighbor parcels with the same owner (subject). The area of interest is limited to the following municipalities: HGZ00 (Hoogezand), HRN01 (Haren), OWD00 (Onstwedde), PKL01 (Nieuwe Pekela), PKL02 (Oude Pekela), VDM00 (Veendam), VWD02 (Vlagtwedde) en WDV02 (Wildervank), WDV03 (Wildervank DE).

First, the objects (parcels) of the requested municipalities are selected from the table `mo_object` and saved into `sel_o`. The next query counts the number of objects an owner (subject) possesses in the municipalities of interest and subsequently only owners with more than one object are saved into `sel_s2`. For every owner from `sel_s2`, his/her/its objects (parcels) are selected and indicated with `x_akr_objectnummer` (and also with `object_id`). This result is stored in `sel_s2_obj_id`. The table `sel_b` contains a selection of the boundaries in the requested municipalities. Combining the last two tables in one query (actually a join of three tables of which `sel_s2_obj_id` is used twice: left and right) gives the result: per owner one or more pairs of neighbor parcels are stored in the table `buur`. The SQL details can be found in the script `q96_buurruil2.sql`:

```
/* selecteer objecten in betreffende secties */
drop table sel_o;\p\g
create table sel_o as
select * from mo_object
where municip in ('HGZ01','HRN01','PKL01','PKL02','WDV02','WDV03','OWD00','VWD02','VDM00')
with nojournaling;\p\g

/* tel aantal 'SM' rechten per subject */
drop table sel_s;\p\g
create table sel_s as
select gerechtigde, aantal=count(*)
from mo_recht
where soort_recht = 'VE' and x_akr_objectnummer in
  (select x_akr_objectnummer from sel_o)
```

```
group by gerechtigde
with nojournaling;\p\g

/* selecteer hieruit de subjecten met meer dan 1 'VE' recht */
drop table sel_s2;\p\g
create table sel_s2 as
select *
from sel_s
where aantal > 1;\p\g

/* zoek hierbij de AKR objectnummer */
drop table sel_s2_obj;\p\g
create table sel_s2_obj as
select s.gerechtigde, s.aantal, r.x_akr_objectnummer
from sel_s2 s, mo_recht r
where s.gerechtigde = r.gerechtigde
  and r.soort_recht='VE';\p\g

/* zoek hierbij de LKI object_id's, later nodig voor buur links/rechts */
drop table sel_s2_obj_id;\p\g
create table sel_s2_obj_id as
select s.gerechtigde, s.aantal, s.x_akr_objectnummer, x.object_id
from sel_s2_obj s, lki_parcel x
where s.x_akr_objectnummer=x.x_akr_objectnummer;\p\g

create table sel_b
as select l_obj_id, r_obj_id
from lki_boundary
where l_municip in ('PKL01','PKL01','WDV02','WDV03','OWD00','VWD02')
with nojournaling;\p\g

modify sel_s2_obj_id to btree on object_id;\p\g

/* zoek nu op waar links en rechts dezelfde subject_id een SM recht heeft */
drop table buur;\p\g
create table buur as
select sl.gerechtigde, sl.aantal,
       object_l=sl.x_akr_objectnummer,
       object_r=sr.x_akr_objectnummer
from sel_s2_obj_id sl, sel_s2_obj_id sr, sel_b x
where x.l_obj_id=sl.object_id and
      x.r_obj_id=sr.object_id and
      sl.gerechtigde=sr.gerechtigde
      with nojournaling;\p\g

/* sorteer op subject_id */
modify buur to btree on gerechtigde;\p\g
```

The contents of table `buur` is copied to a ASCII file `buur.dat` using the standard Ingres program `copydb` and the Unix command `sed` as described in the script `copy_buur.sh`:

```
#!/bin/sh
# make ascii dump of 'buur' table

DB=${1:-iqt_prd1_0100}

# by the copydb statement
copydb -c $DB buur
```

```
cat copy.out |\
  sed 's/varchar(0)/c0/' |\
  sed 's/nl= d1//'  |\
  sed 's/c0nl,/c0nl/' > copy.out2

sql $DB < copy.out2

cat buur.oos | sed 's/ *        /       /g' \
                 | sed 's/ *$//' > buur.dat
```

The result were 18320 pairs of objects (neighbors), sometimes involved in a large complex of neighbor parcels all related to same subject.

References

[1] PGS and Kadaster. Gebruikershandleiding IQT: Integrated query tool, voor administratieve en geografische kadastrale gegevens. Technical report, Kadaster, September 1999. (in dutch).

[2] Peter van Oosterom. Research issues in integrated querying of geometric and thematic cadastral information (1). Technical report, TU Delft, Faculty CiTG, Department of Geodesy, August 2000. GISt No. 1.

[3] 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.

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

[5] Peter van Oosterom, Bart Maessen, and Wilko Quak. Spatial, thematic, and temporal views. In *SDH2000, 9th International Symposium on Spatial Data Handling*, pages 8b. 37–52, August 2000.

Appendix A

This appendix contains some more details with respect to aggregating 'register 9 aktepostenʼ based on three different groupings: (1) concentration group, (2) age, and (3) cadastral office (see Section 5. First the SQL statements are given, this is the sequel of the script `reg9_6.sql` from Section 5:

```
/* bepaal per aktepost perceel met hoogste aantal aktepostenen */
/* unique id aktepost is municip,osection,reg_9_nr */
drop table reg9_maxa;\p\g
create table reg9_maxa as
select m.municip, m.osection, m.reg_9_nr, max_aant_post=max(a.aantal_post)
from mo_reg_9 m, reg9_aantal a
where m.municip=a.municip and m.osection=a.osection and m.parcel=a.parcel
group by m.municip, m.osection, m.reg_9_nr
with nojournaling;\p\g

/* herschrijf max aantal aktepostenen naar concentratiepost groepen */
drop table reg9_maxa2;\p\g
create table reg9_maxa2 as
select municip, osection, reg_9_nr, max_aant_post, concen_groep
from reg9_maxa, concen_def
where start <= max_aant_post and max_aant_post <= last
with nojournaling;\p\g

/* herschrijf aantal aktepostenen naar concentratiepost groepen */
drop table reg9_aantal2;\p\g
create table reg9_aantal2 as
select municip, osection, parcel, aantal_post, concen_groep
from reg9_aantal, concen_def
where start <= aantal_post and aantal_post <= last
with nojournaling;\p\g

/* koppel datum aan reg 9 posten via object attribuut ontdat_st_vest */
/* en herschrijf datum naar maand, dwz afronden op hele maanden */
drop table reg9_maand;\p\g
create table reg9_maand as
SELECT r.municip,r.osection,r.parcel,r.pp_i_ltr,r.pp_i_nr,r.reg_9_nr,
maand=char(o.ontdat_st_vest,6)
FROM mo_reg_9 r, mo_object o
where r.municip=o.municip and r.osection=o.osection and r.parcel=o.parcel and
      r.pp_i_ltr=o.pp_i_ltr and r.pp_i_nr=o.pp_i_nr
with nojournaling;\p\g

drop table reg9_maand2;\p\g
create table reg9_maand2 as
SELECT r.municip,r.osection,r.parcel,r.pp_i_ltr,r.pp_i_nr,r.reg_9_nr,
r.maand, o.period
FROM reg9_maand r, date_def o
where o.start <= r.maand and r.maand <= o.last
with nojournaling;\p\g

/* koppel vestiging aan reg9_maand2 tabel via gemeente,dwz municp */
drop table reg9_maand_vest;\p\g
create table reg9_maand_vest as
SELECT vestigings_code,municip,osection,parcel,pp_i_ltr,pp_i_nr,reg_9_nr,
maand, period
```

```
FROM (reg9_maand2 left join kad_burg_gemeente_cat_iqt
      on kadastrale_gem_code=municip and
      not indicatie_vervallen_kadgem='J')
with nojournaling;\p\g

/* koppel concentratie groep aan reg9_maand_vest via hulptabel reg9_aantal2 */
drop table reg9_maand_vest_con;\p\g
create table reg9_maand_vest_con as
SELECT r.vestigings_code,r.municip,r.osection,r.parcel,r.pp_i_ltr,r.pp_i_nr,r.reg_9_nr,
r.maand, r.period, o.concen_groep
FROM reg9_maand_vest r, reg9_aantal2 o
where r.municip=o.municip and r.osection=o.osection and r.parcel=o.parcel
with nojournaling;\p\g

/* product 1, tussenstap a: betrokken percelen per vest/per concen_groep */
drop table reg9_p1a;\p\g
create table reg9_p1a as
select distinct vestigings_code,concen_groep,municip,osection,parcel
from reg9_maand_vest_con
with nojournaling;\p\g

/* product 1: aantal betrokken percelen per vest/per concen_groep */
drop table reg9_p1;\p\g
create table reg9_p1 as
select vestigings_code,concen_groep,aantal_perceel=count(*)
from reg9_p1a
group by vestigings_code,concen_groep
with nojournaling;\p\g

select * from reg9_p1;\p\g

/* vest_con_maand: tel per perceel per periode het aantal akteposten */
drop table reg9_maand_vest_con_aantal;\p\g
create table reg9_maand_vest_con_aantal as
select vestigings_code, concen_groep, period,
       municip, osection, parcel, aantal_post=count(*)
from reg9_maand_vest_con
group by vestigings_code, concen_groep, period, municip, osection, parcel
with nojournaling;\p\g

/* product 2: per vest, per concen_group, per periode de aantallen posten */
/* vest_con_maand: tel nu hoe vaak een bepaald aantal akteposten per perceel
 voorkomt */
drop table reg9_p2;\p\g
create table reg9_p2 as
select vestigings_code, concen_groep, period, aantal=count(*)
from reg9_maand_vest_con_aantal
group by vestigings_code, concen_groep, period
with nojournaling;\p\g

select * from reg9_p2;\p\g

/* product 3: per vest, per max_concen_group, per periode */
drop table reg9_maxa3a;\p\g
create table reg9_maxa3a as
select r.vestigings_code,
       m.municip, m.osection, m.reg_9_nr, m.max_aant_post, m.concen_groep,
```

```
       r.maand, r.period
from reg9_maxa2 m, reg9_maand_vest r
where m.municip=r.municip and m.osection=r.osection and m.reg_9_nr=r.reg_9_nr
with nojournaling;\p\g

/* tel per aktepost het aantal percelen */
drop table reg9_maxa3b;\p\g
create table reg9_maxa3b as
select vestigings_code, concen_groep, period,
       municip, osection, reg_9_nr, aantal_perc=count(*)
from reg9_maxa3a
group by vestigings_code, concen_groep, period, municip, osection, reg_9_nr
with nojournaling;\p\g

drop table reg9_p3;\p\g
create table reg9_p3 as
select vestigings_code, concen_groep, period, aantal=count(*)
from  reg9_maxa3b
group by vestigings_code, concen_groep, period
with nojournaling;\p\g
```

```
select * from reg9_p1;

+------+----------+-------------+
|vestig|concen_gro|aantal_percee|
+------+----------+-------------+
|AD    |A: 1-4    |         2741|
|AD    |B: 5-8    |          122|
|AD    |C: >=9    |          157|
|AH    |A: 1-4    |        10791|
|AH    |B: 5-8    |          504|
|AH    |C: >=9    |          531|
|AM    |A: 1-4    |         2986|
|AM    |B: 5-8    |          157|
|AM    |C: >=9    |          222|
|AS    |A: 1-4    |         2619|
|AS    |B: 5-8    |          115|
|AS    |C: >=9    |          127|
|BD    |A: 1-4    |         3285|
|BD    |B: 5-8    |          169|
|BD    |C: >=9    |          328|
|EH    |A: 1-4    |         5196|
|EH    |B: 5-8    |          239|
|EH    |C: >=9    |          347|
|GN    |A: 1-4    |         2350|
|GN    |B: 5-8    |          122|
|GN    |C: >=9    |          140|
|GV    |A: 1-4    |         2588|
|GV    |B: 5-8    |          153|
|GV    |C: >=9    |          274|
|LA    |A: 1-4    |         3566|
|LA    |B: 5-8    |          162|
|LA    |C: >=9    |          126|
|LL    |A: 1-4    |         1317|
|LL    |B: 5-8    |           72|
|LL    |C: >=9    |          186|
|MD    |A: 1-4    |         2027|
|MD    |B: 5-8    |           83|
```

```
|MD    |C: >=9    |          112|
|RM    |A: 1-4    |         6594|
|RM    |B: 5-8    |          388|
|RM    |C: >=9    |          311|
|RT    |A: 1-4    |         5023|
|RT    |B: 5-8    |          271|
|RT    |C: >=9    |          346|
|UT    |A: 1-4    |         2970|
|UT    |B: 5-8    |          179|
|UT    |C: >=9    |          317|
|ZL    |A: 1-4    |         4482|
|ZL    |B: 5-8    |          259|
|ZL    |C: >=9    |          315|
|      |A: 1-4    |           72|
|      |B: 5-8    |            3|
|      |C: >=9    |            5|
+------+----------+-------------+
(48 rows)
```