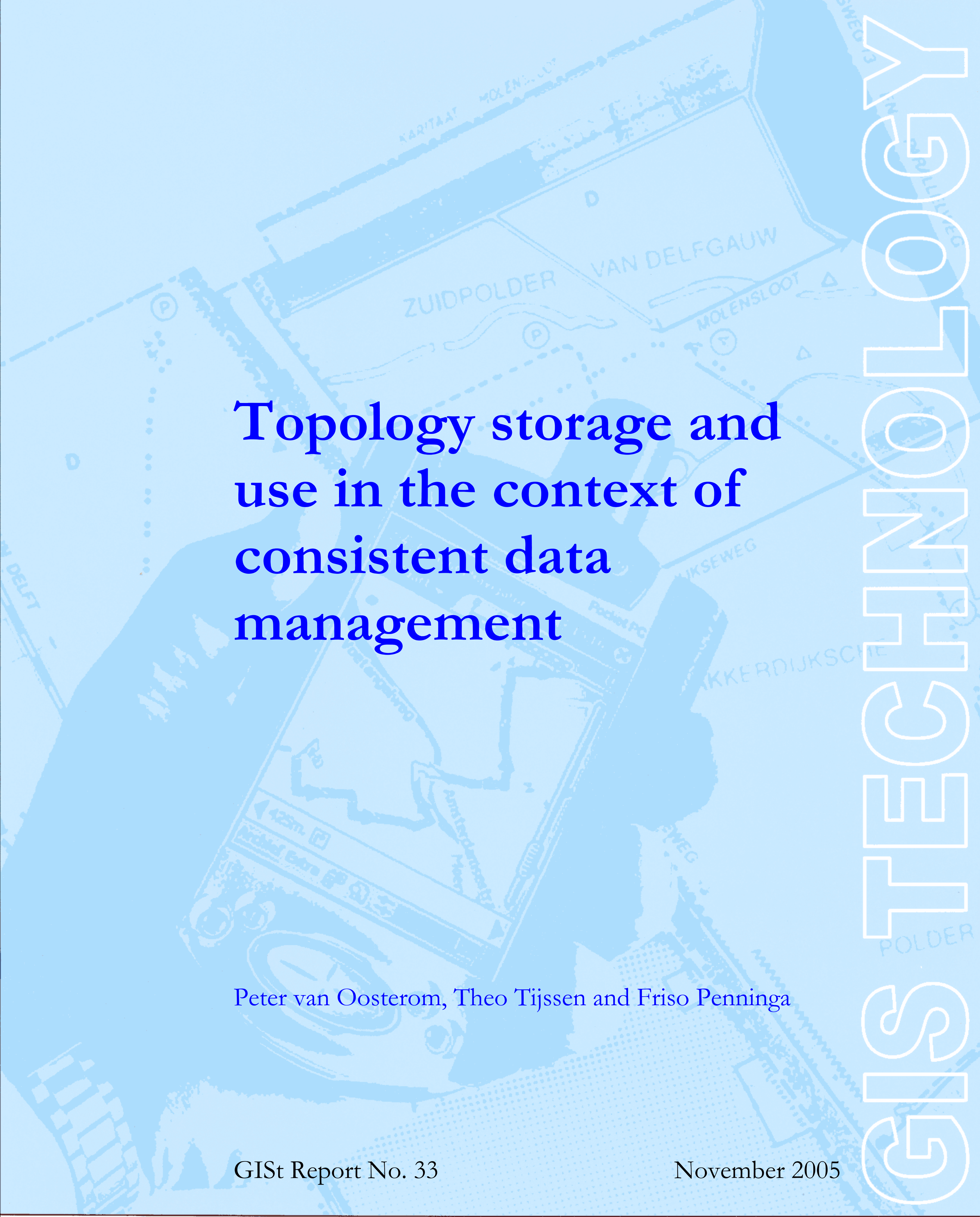


Topology storage and use in the context of consistent data management

Peter van Oosterom, Theo Tijssen and Friso Penninga

GISt Report No. 33

November 2005

GIS TECHNOLOGY

OTB Research Institute for Housing, Urban and Mobility Studies

Delft University of Technology

Topology storage and use in the context of consistent data management

Peter van Oosterom, Theo Tijssen and Friso Penninga

GISt Report No. 33 November 2005

Summary

This report describes a research to compare the use of topology within Oracle 10g Topology and ESRI Geodatabase. Cadastral data was selected as this data is already available in a topological winged-edge data structure. The purpose of this project is to investigate different approaches to manage topology related to the geo-information storage within a DBMS environment. Both the functional aspect and the performance aspect are investigated. A large number of consistency checks have be defined for the cadastral model. In total, over 50 constraints, of which many related to the topological structure, have been defined in a number of different categories; see Appendix 1. Based on the results of these checks one of the more clean and large cadastral offices (province of Gelderland) was further used for testing. The cadastral data set is considered very clean and is designed to avoid certain types of errors, such as overlapping parcels, by the nature of the structure (based on topological references). However, the constraints applied to production data of 1 sept'03 showed that certain types of errors have been introduced in the data (in small numbers). One important lesson is that one should never trust on front-end and/or middle ware alone for consistency checking, but implement this throughout the whole system and in any case within DBMS. Other important findings of this research are: Oracle Topology does not support circular arcs (these have to be stroked), the stroked coordinates need more than 32 bits per ordinate (this was not available in ESRI geodatabase v. 9.1, but is now available in v. 9.2) in order to avoid topology errors, Oracle topology needs explicit nodes and separation of the 'topology and geometry' level from the 'feature' level (can be automated via conversion script based on current topological winged-edge structure in LKI).

ISBN: 90-77029-09-5/978-90-77029-09-1
ISSN: 1569-0245
© 2005 Section GIS technology
OTB Research Institute for Housing, Urban and Mobility Studies
TU Delft
Jaffalaan 9, 2628 BX Delft, the Netherlands
Tel.: +31 (0)15 278 4548; Fax +31 (0)15-278 2745
Websites: http://www.otb.tudelft.nl
http://www.gdmc.nl

E-mail: Sectie-GIST@otb.tudelft.nl

This publication is the result of the research programme Sustainable Urban Areas, carried out by Delft University of Technology.

Contents

1 Introduction

The purpose of this project is to investigate different approaches to manage topology related to the geo-information storage within a DBMS environment. Both the functional aspect and the performance aspect are investigated.

The test environment consisted of a Sun E3500, Solaris 9, Oracle 10g Release 10.1.0.2.0 and later Release 10.1.0.4.0, ArcGIS/ ArcSDE/ Geodatabase 9 with test data consisting of cadastral parcels including update history (three separate data sets, each with a different number of parcels; data set 1: 50,000 parcels, data set 2: 300,000 parcels, and data set 3: 1,000,000 parcels). The data is loaded in different geo-DBMS platforms:

A. Oracle 10g with topology
B. ArcGIS Geodatabase

Case B will not use Oracle 10g topology. In all cases Oracle 10g will be used, but in different manners. In these test environments it was planned to carry out 500 single user test queries (random rectangle and polylines) and 500 single user updates (based on history in database of Cadastre). The research project is structured in the following phases:

- Loading: Making the load-scripts and loading the test data sets in the Oracle 10g and the Geodatabase environments.
- Queries: Making of 500 (random) rectangle and polygon queries and query-scripts, carry out the queries in the test environments.
- Updates: Making 500 updates with matching update scripts, carry out 500 updates in the test environments.

In chapter 2 an overview of the test environment is given (hardware). The next chapter describes the selection of a suitable test data set (selection of cadastral data of right size and quality). Some details with respect to loading this geometry and topology in Oracle 10g topology are given in chapter 4, while chapter 5 does the same for the ESRI Geodatabase environment. Due to the (unexpected) difficulties in selecting appropriate data and loading these in the different environments, no systematic comparison of performing the queries in these different environments could be made. The generation of updates for the test environment is discussed in chapter 6 (again the actual updates could not be executed and compared). Despite these difficulties, several important conclusions can be obtained from the research presented in this report. These are given in the last chapter together with suggestions for future research.

2 Test Environment

The system on which both the Oracle instance and the ArcSDE server are running during this project is a Sun Enterprise 3500 with the following vital characteristics:

- 2 UltraSPARC CPUs of 400 MHz;
- 2 GB main memory,
- 8 internal harddisks of 18 GB (10000 rpm, fiber channel attached, 2 controllers);
- 24 external harddisks of 18 GB in 2 Sun StorEDGE A1000 boxes offering hardware RAID support (10000 rpm, SCSI channel attached, 2 controllers);
- Solaris 9 operating system (64 bit)

The external disks are organized in 4 RAID5 disk sets with 6 disks in each set. All operating system patches mentioned in the Oracle 10 and ArcSDE 9 documentation are applied.

A 'standard' PC, running the ArcGIS software under Windows 2000, is used as client machine in the project: Pentium 4, 2.8 Ghz, 1 GB memory.

3 Cadastral Data Selection and Preparation

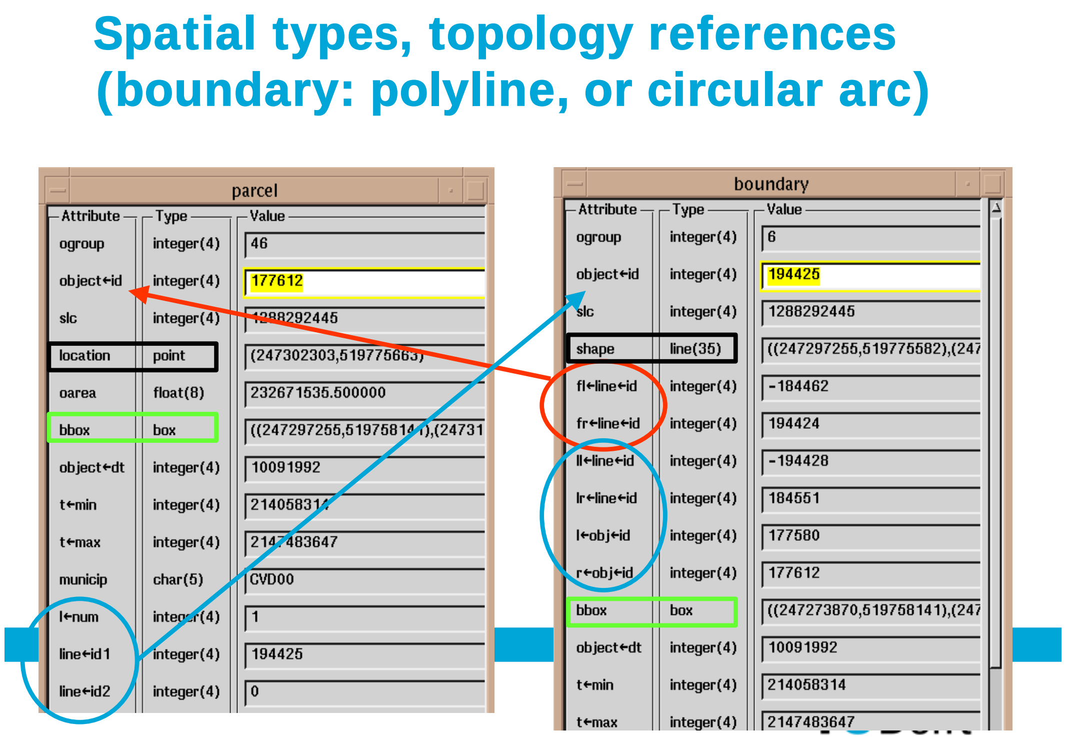
As topology structure in Oracle 10g requires a clean data set without topology errors, first the different cadastral offices were analyzed with respect to these errors. The cadastral office with the least errors will be used. After correcting, the hopefully few errors, a fair comparison can be made between the topology solutions of Oracle and ESRI. This analysis is based on a number of queries that check the integrity (correctness) of the data, these are called the integrity constraints. In section 3.1 a general introduction of the Netherlands cadastral data is given. The selection of the cadastral office is based on the least data errors on integrity constraints given in section 3.2. Section 3.3 gives an overview of how often a certain type of constraint is violated in the different cadastral offices. Based on these results the province of Gelderland was selected as test data set. However, some data errors had to be corrected first (see section 3.4). This chapter finally concludes with some general remarks with respect to managing data integrity constraints.

3.1 Netherlands cadastral data

The cadastral map is based on winged-edge topology structure stored in a DBMS (Van Oosterom and Lemmen, 2001), see Figure 3.1. An important characteristic of the system is that it is considered to be a topologically very clean data set. Further, the model contains redundancy in the topological references: both the (meaningless system) object_id reference to the left and right parcel and the (meaningful user) parcel_number references to the left and right parcels are stored and maintained. In the LKI production environment, the data quality (incl. topology) checks are hard coded and build in the editor and also in the check-in software at DBMS server side. However, the checks are currently not implemented within the DBMS itself (Ingres). The data set completely covers the Netherlands and contains history since 1997-today. For the test a copy of the data set with history a little after 1 Sept. ’03 was available. The total number of current boundaries (polylines) is about 22,000,000 and the number of current parcels (topological faces) is about 7,000,000. When all historic versions are also counted then the numbers are currently about four times higher. There is a separate, but linked, subsystem containing the legal and administrative data (AKR).

3.2 Cadastral data constraints

Due to some redundancy in the system and the fact that, in general, topology references can be derived from the metric information, a large number of consistency checks can be defined for the cadastral model. In total, over 50 constraints have been defined in a number of different categories; see Appendix 1. Note that these are not (all) used in the production environment, but now applied afterwards to select the best quality data (least number of constraint violations). In this section some examples of the different categories will be presented. The examples will be accompanied by SQL select statements, which in case of correct data should not find any objects. The following four categories of cadastral constraints will be discussed: metric constraints, topological reference existence constraints, topological reference correctness constraints, and (other) referential integrity constraints.



Spatial types, topology references (boundary: polyline, or circular arc)

parcel

Attribute	Type	Value
ogroup	integer(4)	46
object←id	integer(4)	177612
slc	integer(4)	1288292445
location	point	(247302303,519775663)
oarea	float(8)	232671535.500000
bbox	box	((247297255,519758141),(24731
object←dt	integer(4)	10091992
t←min	integer(4)	214058314
t←max	integer(4)	2147483647
municip	char(5)	CVD00
l←num	integer(4)	1
line←id1	integer(4)	194425
line←id2	integer(4)	0

boundary

Attribute	Type	Value
ogroup	integer(4)	6
object←id	integer(4)	194425
slc	integer(4)	1288292445
shape	line(35)	((247297255,519775582),(247
fl←line←id	integer(4)	-184462
fr←line←id	integer(4)	194424
ll←line←id	integer(4)	-194428
lr←line←id	integer(4)	184551
l←obj←id	integer(4)	177580
r←obj←id	integer(4)	177612
bbox	box	((247273870,519758141),(247
object←dt	integer(4)	10091992
t←min	integer(4)	214058314
t←max	integer(4)	2147483647

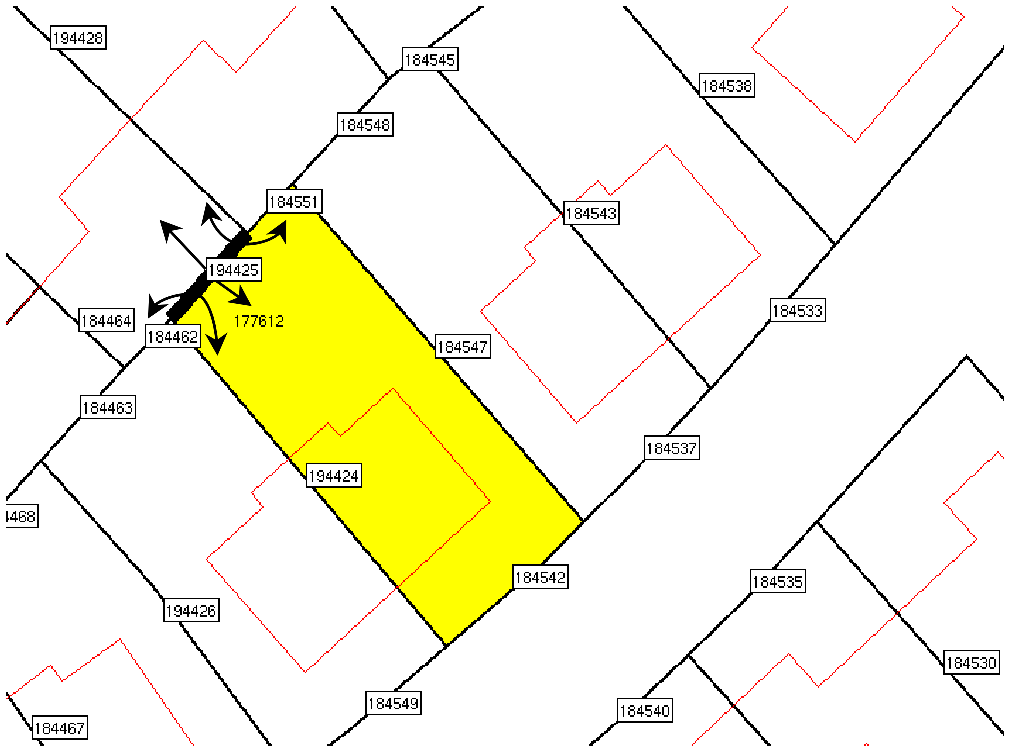
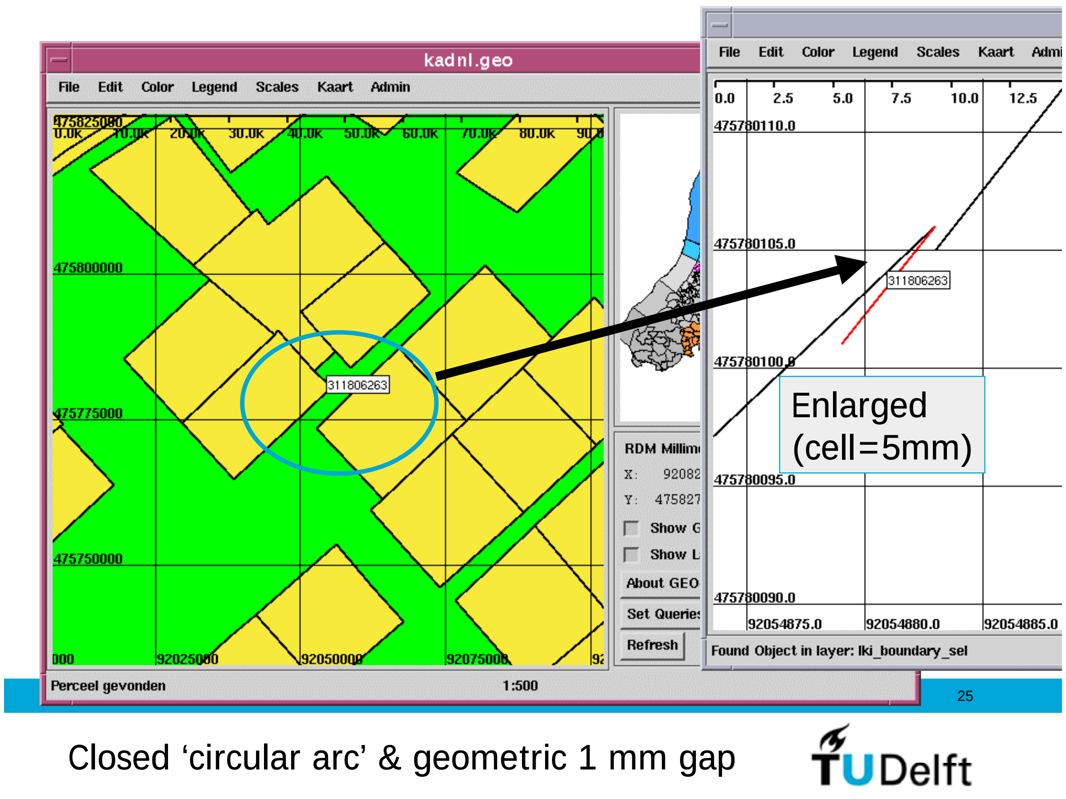


Figure 3.1 **Winged-edge topology structure of the cadastral map.**


kadnl.geo
File Edit Color Legend Scales Kaart Admin
311806263
Enlarged (cell=5mm)
Found Object in layer: lki_boundary_sel
Perceel gevonden
1:500
Refresh
25
Closed ‘circular arc’ & geometric 1 mm gap
TU Delft

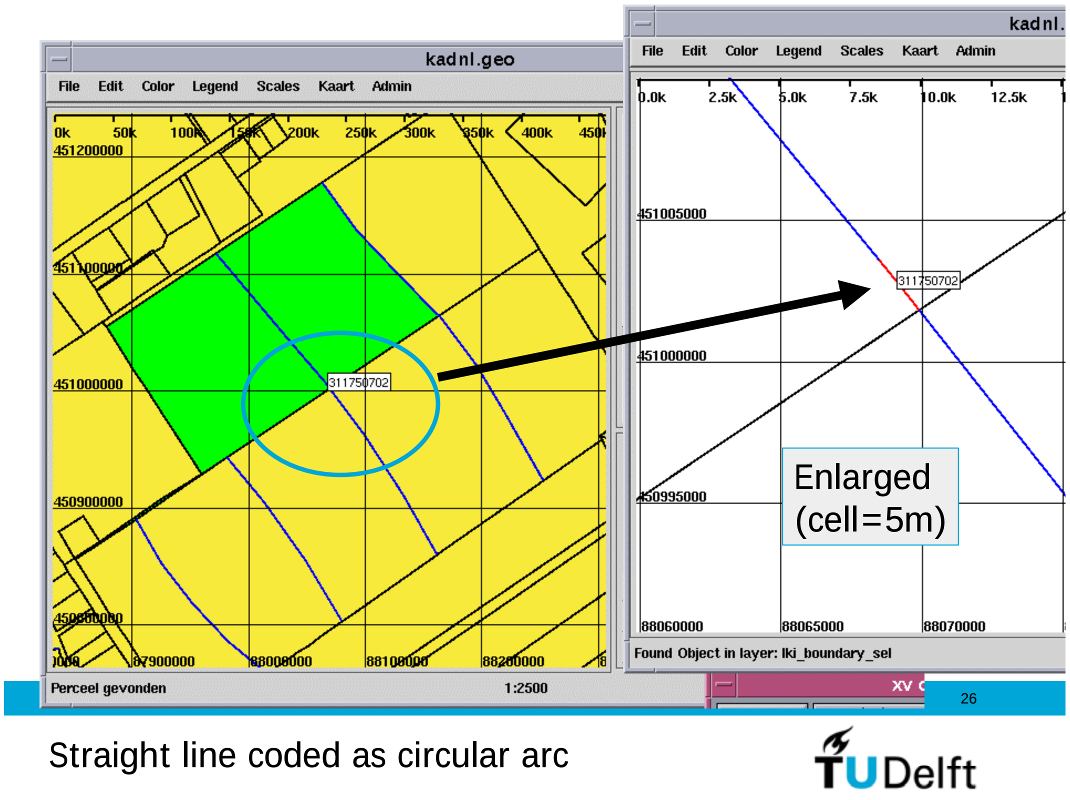

kadnl.geo
File Edit Color Legend Scales Kaart Admin
311750702
Enlarged (cell=5m)
Found Object in layer: lki_boundary_sel
Perceel gevonden
1:2500
26
Straight line coded as circular arc
TU Delft

Figure 3.2 Some metric errors in the cadastral data set (top: small gap between two boundaries, bottom: straight line encoded as circular arc).

The first category of constraints could be classified as *metric* checks. The first example finds closed 'arcs' (circles are not allowed as their orientation is not specified and therefore it is not possible to define what is the left and right side, which is very important in a topological structure), which can be detected by checking whether the first and last (third) point defining the arc are equal:

```
SELECT object_id, numpoints(shape),
  anypoint(shape, 1), anypoint(shape, 2), anypoint(shape, 3)
FROM xfio_boundary
WHERE numpoints(shape)=3 and interp_cd=3 and tmax = 0 and
  ogroup = 6 and (anypoint(shape, 1) =  anypoint(shape, 3));
```

In the same category is the constraint that disallows straight 'arcs'; see Figure 3.2. In the Ingres DBMS this is accepted, but this is not necessary the case in other systems; for example Oracle DBMS. In any case it is not correct, so this type of error should be avoided by defining a constraint that does not allow straight 'arcs'. Every parcel has a reference point, which should be within the area of the parcel. Therefore this reference point should also be in the bounding box of the parcel, which is easily checked with the following constraint:

```
SELECT object_id from lki_parcel
WHERE inside(location, geo_bbox) != 1;
```

The last example in this category is that two different boundaries (straight line or circular arc) should not intersect, but should be disjoint or touch at their end points.

The second category of constraints is based to the *existence of topological references*. This can be very well compared to the referential integrity checks well-known in administrative databases. A slight complication is that the topological references can be signed (+ or -) in order to indicate the proper orientation. The first constraint in this category checks whether the left (and right) parcel references from the boundaries do indeed exist:

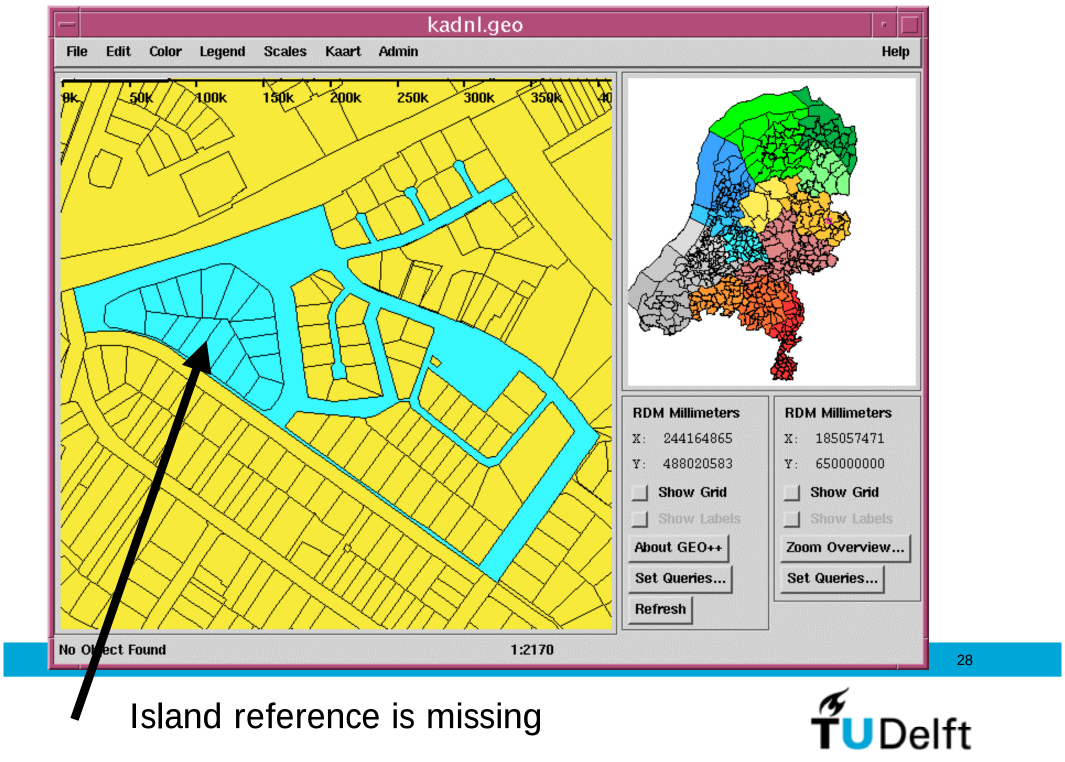
```
SELECT l_obj_id FROM lki_boundary
WHERE l_obj_id not in (select object_id from lki_parcel);
```

The next example in this category does check whether the winged-edge boundary-boundary reference (in this case the first left reference, similar constraints have to be specified for the other three references) does exist**:**

```
SELECT object_id, fl_line_id FROM lki_boundary
WHERE abs(fl_line_id) not in (select object_id from lki_boundary);
```

Now starting from the parcel also a number of topological reference existence checks can be imagined. As parcels can have island boundaries, also these references have to be correct. So, the reference from the parcel to the island boundary reference must exist. Further, a parcel can have any number of islands (and the number of islands is encoded as an explicit attribute). It has to be checked whether the correct number of parcel references are specified and if they all refer to existing boundaries. The final example in this constraint category checks whether the reference from the parcel to its outer boundary does exist:

```
SELECT object_id, line_id1 FROM lki_parcel
WHERE abs(line_id1) not in (select object_id from lki_boundary);
```



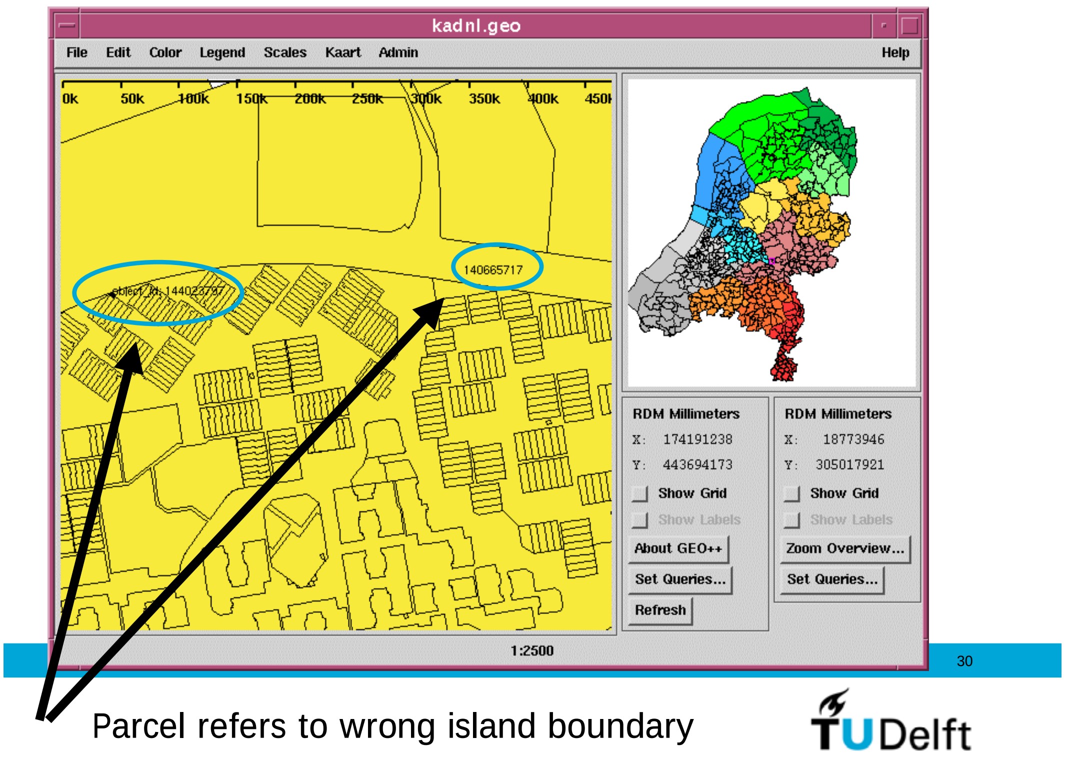


Figure 3.3 **Some topology reference errors in the cadastral data set (top: island reference is missing, bottom: parcel refers to wrong island).**

The third category of constraints goes a step further by checking the *correctness of a topological reference*, see Figure 3.3. A first example in this category is the check that two consecutive boundaries must have the same parcel at the side. In total there are 8 combinations which have to be checked as every of the four winged-edge boundary-boundary references is signed, that is, direction of next edge has to be reversed or not (and thereby switching left and right hand side). The following example checks the positive first left reference:

```
SELECT s.object_id, s.fl_line_id
FROM xfio_boundary s, xfio_boundary r
WHERE s.fl_line_id > 0 and s.fl_line_id=r.object_id and
  s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
  s.r_obj_id <> r.l_obj_id;
```

A similar constraint in this category is that the end point of one boundary is the start point of the next boundary (as in the cadastral database all boundaries are stored with start, end and if any, also intermediate points). Similar to the previous consistency check, there are again 8 combinations which have to be checked of which the following example shows the positive first left case again:

```
SELECT s.object_id, s.fl_line_id
FROM xfio_boundary s, xfio_boundary r
WHERE s.fl_line_id > 0 and s.fl_line_id=r.object_id and
  s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
  (anypoint(s.shape, 1) <>  anypoint(r.shape, 1));
```

Also in this category of constrains is the check whether the island boundary has parcels at the proper side. Another constraint would be the check if first coordinate of island boundary is indeed within bounding box of the parcel. These examples will not be illustrated here, but finally a check is given to see whether the outer boundary and parcel references back-and-forth are consistent:

```
SELECT s.object_id, s.line_id1
FROM xfio_parcel s, xfio_boundary r
WHERE s.line_id1 > 0 and s.line_id1=r.object_id and
  s.tmax=0 and s.ogroup=46 and r.tmax=0 and r.ogroup=6 and
  (s.object_id <>  r.r_obj_id);
```

Near trivial consistency check are queries to verify that the double left and right references, via (meaningless) object_id and (meaningful) parcel_number are consistent. That is, they should both point to the same parcel.

The last and fourth category of constraints that will be mentioned and illustrated, does check whether two subsystems are consistent, that is, *referential integrity* (also in this category are checks to make sure that for every parcel there is at least one person with a certain type of legal right associated). The two subsystems are the geometric subsystem (LKI) and the administrative and legal subsystem (AKR). For this purpose every ground parcel in AKR should also be present in LKI:

```
SELECT count(*),municip FROM mo_object
WHERE pp_i_ltr='G' and x_akr_objectnummer not in
```

```
  (select x_akr_objectnummer from lki_parcel)
GROUP BY municip;
```

3.3 The results

The four tables below show the results of the quality analysis of the cadastral data (dated 1 September 2003). In all columns the number indicates the number of found errors (note that different types of errors can be related to one problem situation). The only exceptions are the columns in Table 3.2, showing the total numbers of boundaries and parcels (both current and historic). Again all queries can be found in Appendix 1.

vest	closed_circ	straight_arc	vest_grens_l	vest_grens_r	l_notexits	l_diff	l_diff2	r_notexits	r_diff	r_diff2
'GN'		4								
'LA'		12								
'AS'		2								
'ZL'		10		1					1	
'LL'		2								
'AH'		**11**	**1**				**1**			
'UT'	1	20	13	3	4	1	14	2	1	4
'AD'		1								
'AM'	2	3	7	1		2	5			1
'GV'	2	12	3			2	3			1
'RT'		18	10	74		6	5		2	72
'MD'		4								
'BD'		9								
'EH'		1							4	4
'RM'		26				44	44		41	41
	5	**135**	**34**	**79**	**4**	**55**	**72**	**2**	**49**	**123**

Table 3.1 Results of the first set of checking the integrity constraints (number of violations reported).

vest	fl	ll	fr	lr	bnd_notexits	wel_lki_niet_akr	niet_lki_wel_akr	bnd act	bnd hist	parc act	parc hist
'GN'						6	10	878786	1938483	281671	834681
'LA'						31		1434267	2511276	417936	955216
'AS'						3		921025	1877345	281609	1088828
'ZL'						1	1	1767124	9747939	526427	3565565
'LL'								392784	693723	122778	287805
'AH'						**13**	**11**	**3122320**	**11271772**	**881961**	**4271111**
'UT'						11	25	1524905	3624894	402668	1386004
'AD'						24	74	1407737	4256141	432650	1694700
'AM'						25	5	1290887	3627422	380983	1476231
'GV'						47	18	1401591	6546249	423593	2458029
'RT'						28	15	1962170	8100361	577540	2964697
'MD'						11		810230	1595640	238030	681151
'BD'								1526290	3313511	476081	2048831
'EH'							4	1898759	3678490	584445	2393360
'RM'	1		1			1	5	1938375	4713624	567021	2241493
	1	**0**	**1**	**0**	**0**	**201**	**168**	**22277250**	**67496870**	**6595393**	**28347702**

Table 3.2 Results of the second set of checking the integrity constraints (number of violations reported) with exception of the last four columns (they contain statistics).

vest	box err	fl+	fl-	fr+	fr-	Ll+	ll-	lr+	lr-	parbnd+	parbnd-	parbnd2+	parbnd2-
'GN'													
'LA'		1			1		1	1					
'AS'					1		1						
'ZL'											1		
'LL'													
'AH'				**1**			**1**						**1**
'UT'													
'AD'													
'AM'													
'GV'				2		2							
'RT'			1		1	2		2	1				
'MD'													
'BD'		1	1	1	1	2		1					
'EH'										1			
'RM'		8	6	6	19	13	12	10	7				
	0	**14**	**8**	**10**	**23**	**19**	**15**	**14**	**8**	**1**	**1**	**0**	**1**

Table 3.3 Results of the third set of checking the integrity constraints (number of violations reported).

vest	eq_fl+	eq_fl-	eq_fr+	eq_fr-	eq_ll+	eq_ll-	eq_lr+	eq_lr-	isl_cnt	isl_ref	isl+ref	isl-ref	isl-box
'GN'													
'LA'	1		1	2	1	1	1						
'AS'				1		1							
'ZL'	1	2					2			1			
'LL'													
'AH'			**1**			**1**							
'UT'	1	1	1	2	2		1				1	4	
'AD'													
'AM'	1		1			1		1		1			
'GV'			2	2	2								
'RT'	8	2	7			7	3	8					
'MD'													
'BD'	2	1	1	1	1		1						
'EH'	2	4	1			1	4	1					
'RM'	11	6	7	19	13	14	10	9					
	27	**16**	**22**	**27**	**19**	**26**	**22**	**19**	**0**	**2**	**1**	**4**	**0**

Table 3.4 **Results of the fourth set of checking the integrity constraints (number of violations reported).**

3.4 Selecting (province of Gelderland) and correcting cadastral data

From the Tables 3.1 to 3.4 it can be concluded that no cadastral office is without errors. The offices of Flevoland ('LL'), Groningen ('GN'), part of Noord-Holland ('AD') and Zeeland ('MB') are very clean, but they are also too small for the required large test data set (1,000,000 parcels). The Arnhem office is quite large (close to the 1,000,000 parcels, that is, 880,000 to be a bit more precise) and does not contain many errors. It was therefore decided to use the Arnhem office after correcting the few errors 'by hand'. That is, the errors were first visually inspected (see also Figures 3.2 and 3.3) and small SQL update scripts were created to remove the topology errors at the current data set time (1 Sept. '03). In the Arnhem data set all errors (except the 'straight arc' errors) could be related to 4 errors.

Correction of error 1:

```
delete from xfio_parcelover
  where object_id=140747901 and tmax = 0;
```

This parcel did not exist anymore in the xfio_parcel table, but it did exist wrongly in the xfio_parcelover table.

Correction of error 2:

```
update xfio_boundary set r_obj_id=140287656
  where ogroup = 6 and r_obj_id=0  and l=obj_id=141047977
    and tmax = 439295996;
update xfio_boundary set l_obj_id=140287656
  where ogroup = 6 and l_obj_id=0  and r=obj_id=141047977
    and tmax = 439295996;
```

This was the parcel with object_id 140287656, with a number of islands. However one of the islands did not point back to the parent parcel via the left/right references. This was corrected. Remarkable in this case was the fact that shortly after 1 Sept. '03 this parcel has been updated in LKI. However, we are looking at the historic situation at 1 Sept. '03, and that must be correct. Actually also the redundant left/right municip/section/parcel should be corrected, but this is not used in the Oracle scripts later on. So, this was not done.

Correction of error 3:
```
update xfio_parcel
  set l_num=2, line_id2=-143720040
  where ogroup = 46 and object_id = 140665717 and tmax = 0;
delete from xfio_parcelover
  where object_id = 140665717 and tmax=0;
```

This was parcel 140665717, which did actually have 1 island, but it was wrongly indicated that there were more islands, according the l_num and the record in the xfio_parcelover table.

Correction of error 4:
```
update xfio_boundary
  set fr_line_id=-144641971, ll_line_id=144641971
  where ogroup = 6 and object_id = 144641970 and tmax = 0;
```

Correcting one wrong boundary-boundary reference.

3.5 Some lessons, cadastral data

The cadastral data set is considered clean and is designed to avoid certain types of errors, such as overlapping parcels, by the nature of the structure (based on topological references). During the conversion from the old to the new version of the system all consistency errors were resolved and removed. Further, the cadastral data is delivered to many different customers and the cadastral data is loaded in different systems (each possibly sensitive for different types of errors). As the customers pay for this data, they are inclined to complain quite quickly in case of errors. However as the constraints discussed in the previous section and applied to production data of 2004 have clearly illustrated that certain types of errors have been (re)introduced in the data despite all the measures. One important lesson is that one should never trust on front-end and/or middle ware alone for consistency checking, but implement this throughout the whole system and in any case within DBMS (as this will contain what is considered valid data to be shared by multiple users). Further, it can be the case that even if the errors may not be noticed in ones own production environment, they may be harmful in the environments of others; e.g. straight 'circular arcs'.

Despite the thorough treatment of the different categories of constraints, still not all (types of) possible constraints are discussed. In the category of topological correctness it is not checked whether the complete domain is covered with parcels. This is a very important type of topological constraint (in many applications) as there should be no gaps (in the cadastral case this would be an area without the possibility to have an owner). However, it is the strong suspicion of the authors, that the earlier given topology constraints implicitly cover the total coverage constraint. Besides the well-known attribute value domain constrained (not illustrated) and the illustrated four other constraint categories (metric, topological reference existence, topological reference correctness, and (other) referential integrity), one last category will be mentioned: temporal constraints. Temporal constraints make sure that the time intervals of two consecutive versions of an object do touch (no gaps or overlap in the time dimension of an object).

4 Oracle Geometry and Topology

For more backgrounds on the Oracle Topology data model and the conversion from the Cadastral data model into the Oracle model we refer to the report 'Oracle10g Topology, Testing oracle 10g Topology using cadastral data' (GISt report No. 26) and the documentation 'Oracle Spatial Topology and Network Data Models'. Before the selected Dutch cadastral data (of the province of Gelderland) could be used with Oracle and ESRI topology some geometrical and topological preprocessing was required. This has to do with an unsupported feature (circular arcs and topology; see section 4.1), coordinate representation and tolerance (section 4.2) and the explicit presence of nodes in the Oracle topology model, in contrast to the current LKI cadastral model (section 4.3).

4.1 Circular arcs

Oracle Spatial supports more geometry types than Oracle Topology. In particular the circular arcs included in the Dutch cadastral boundaries are not allowed in Oracle Topology. The boundaries that are arcs must be converted to 'stroked' approximations consisting of straight-line segments. The function to stroke arcs in Oracle can only be applied to valid geometries (like all other spatial functions). But 'straight' arcs, a circular arc defined by three points on a straight line, are not valid in Oracle Spatial. So the first preprocessing step is to find the boundaries that are straight arcs and change their geometry type from arc to straight line (see Section 3.3). A few of the remaining arcs are not arc segments but complete circles: a circular arc defined by three points of which the first and the last point are identical. According to cadastral standards these, just like straight arcs, should not occur, but in fact they do occur in the current cadastral data set so they must be dealt with. The arcs-as-complete-circle (also invalid in Oracle because the points are on a straight line) are stroked separately (this did not occur in Gelderland). Some additional processing is required to establish the proper 'direction' as a closed circle does not have a direction. The direction of the resulting polyline must correspond to the topological (left and right) references of the cadastral data. Finally the remaining circular arcs are stroked 'as-is', with an accuracy (arc_tolerance=0.4 mm) that is sufficient to avoid collapses or unwanted intersections of geometries (because of the approximation).

A small example of the treatment of arcs is provided below. Initially all circular arcs in the data set are stored as 3-point polylines (SDO_GTYPE=2002) with arcs connecting the points (SDO_ELEM_INFO=1,2,2). The coordinates of one randomly selected polyline with arcs are:

```
1  213665592     460342565
2  213535280     460283709
3  213440355     460176779
```

After stroking the coordinates of the same polyline are (SDO_GTYPE still 2002, but SDO_ELEM_INFO now 1,2,1):

```
    1  213665592     460342565
    2  213664572.33  460342341.83
    3  213663553.34  460342115.56
    4  213662535.05  460341886.18
    5  213661517.47  460341653.7
   .        .             .
   .        .             .
   .        .             .
  273  213442436.11   460180398.57
  274  213441911.7    460179496.06
  275  213441390.04   460178591.96
  276  213440871.14   460177686.27
  277  213440355      460176779
```

In this case the resulting line contains 277 points, in the data set used the stroked boundaries contain up to 400 points. Oracle calculates many more digits after the decimal point, the coordinates are rounded to 2 digits after the point to avoid wasting unnecessary storage. The Oracle Spatial SDO_ARC_DENSIFY function, available in the more recent versions of Spatial, did prove to do be very useful for this purpose.

4.2 Coordinate representation and tolerance

Dutch cadastral boundary coordinates are stored in mm, as integers. The coordinate range for the whole of the Netherlands is approximately from –25,000,000 to 650,000,000. This fits nicely in a 32-bit integer. One complication in the whole process is that, because of the stroking of the arcs, coordinates with digits after the decimal point appear. Maybe one digit after the decimal point would suffice (this was not tested), but rounding to integers would certainly not work. If only integers were used in the stroked arcs the resulting geometries would collapse, intersect with other boundaries or end up on the 'wrong' side of other boundaries.

Another complication is the required tolerance for the current data set. Although, before stroking, only integer coordinates are used, this does not prevent parcel boundaries from passing very close to each other. In fact a tolerance of 0.002 mm is required to avoid problems (intersections) in the data set used for the research. Situations like this are not meaningful and should not occur. However, in reality they do occur in the current data set.

4.3 Preparation of Oracle topology

The Dutch cadastral data used as input for the project is already topologically structured, in principle it should not be too hard to convert this into Oracle (and ArcGIS) topology. In reality it proved to be a substantial effort. The main obstacle to overcome is the fact that the input (LKI) topology does not use nodes, it completely relies on winged-edge relations. During the conversion process from LKI to Oracle topology the nodes are derived from the start- and endpoints of boundaries. To make this work is not that difficult, but to make it work with a reasonable performance requires significant tuning and testing. Another difficulty is caused by 'correcting' the topological references at the 'perimeter' of data sets that consist of subsets of the complete data set. The 50,000 and 300,000 parcel subsets require complicated update statements to make the boundaries at the outside of the subset point to the 'exterior face' and not to parcels and boundaries that are outside the subset, as is the case initially. Appendix 3 contains an example of the scripts to create the topology and the parcel features.

Earlier Oracle 10g versions produced in certain cases geometries (based on correct topology) with some errors. The latest version, Oracle 10.1.0.4, produced completely correct topology and geometry based on the Dutch cadastral data set.

5 ESRI Geodatabase

The ESRI Geodatabase does not store explicit topology structure, but polygons. For the cadastral data set these polygons first have to be created (e.g. with the available function in Oracle) and these polygons can be loaded in the ESRI Geodatabase. The coordinates of the (shared) boundaries are now stored in two polygons (left and right). A set of topology rules makes sure that there are no errors introduced (overlapping parcels/polygons, or gaps within the domain).

The ArcGIS Interoperability Extension was used to export cadastral (parcels could be created in Oracle using the functions for the topology structure) boundaries and related information from the Oracle tables to an ArcSDE geodatabase. At least that was the intention, but because of the issues regarding coordinate representation and tolerance the resulting output geodatabase (version 9.1) is not really comparable to the input Oracle data. This problem does no longer exist with ArcGIS Geodatabases (version 9.2), which can use more than 32 bits to store coordinates and related tolerances.

An initial attempt to export a data set failed, nearly all numerical attributes came out complete garbled at the geodatabase side. After consulting ESRI this problem was solved: the cause was the fact that in the Oracle database all numerical attributes were initially stored with an 'unspecified' NUMBER type, defining these with specific lengths in the Oracle tables (e.g. NUMBER(12) or NUMBER(15,3)) solved the problem.

For more backgrounds on topology in ESRI Geodatabase we refer to 'ArcGIS: Working with Geodatabase Topology', an ESRI White Paper.

6 Generating Updates

In order to simulate interactive editing the history in the LKI database can be used to obtain the changes over a certain period. Every edit (set of deleted and new object versions all with the same time stamp) should bring the database from one consistent state into the next consistent state. The LKI copy did contain also some information after 1 Sept. ’03 (the current time) and this was the source of the edit actions. The script in Appendix 2 counts the number of new parcels, deleted parcels, new boundaries and deleted boundaries (after 1 Sept. ’03) and groups them on the edit/mutation time. Table 6.1 gives the result (only the first rows).

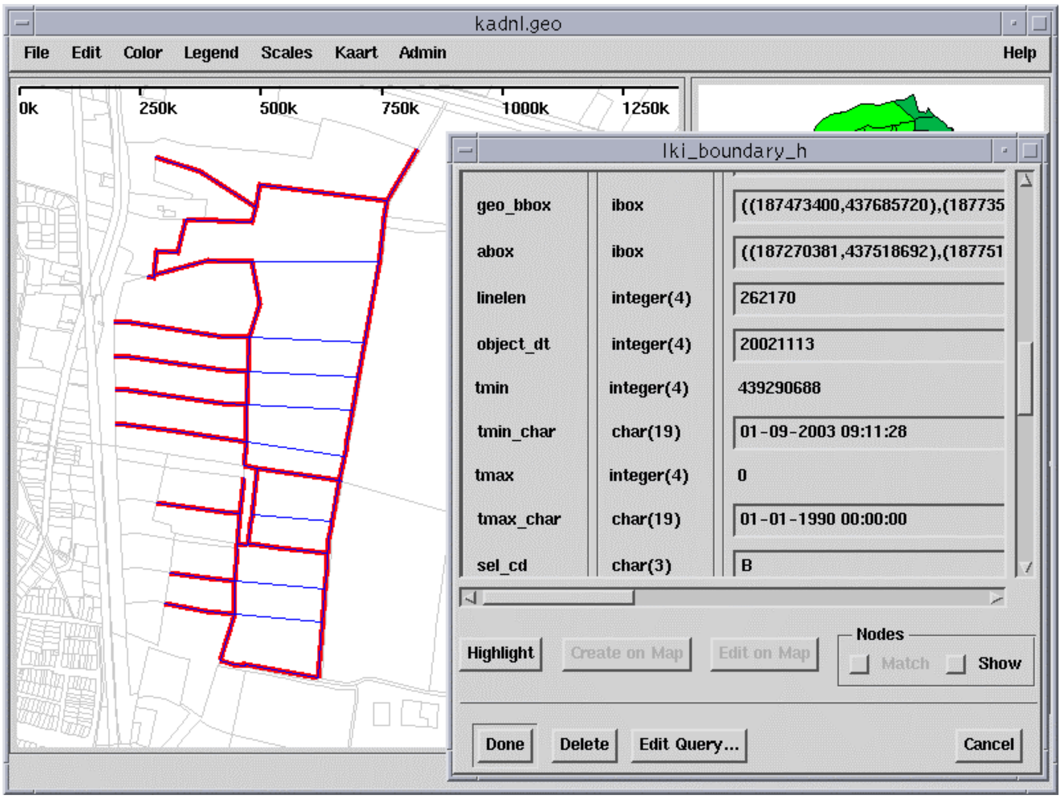
```
pnew.tmin    lkiint2date(pnew.tmin)  new_parc old_parc new_bnd old_bnd
439290688       01-09-2003 09:11:28        31       23      58      40
439293140       01-09-2003 09:52:20       295      295     632     593
439295996       01-09-2003 10:39:56         3        3      20      34
439297186       01-09-2003 10:59:46        58       58      91      91
439297439       01-09-2003 11:03:59         2        2       2       5
439297593       01-09-2003 11:06:33        15       12      44      20
439302720       01-09-2003 12:32:00        35       32     149     129
439311433       01-09-2003 14:57:13        39       37      84      68

439311799       01-09-2003 15:03:19        61       58      93      87
439312132       01-09-2003 15:08:52        18       18      38      37
439312261       01-09-2003 15:11:01         8        7      14      13
439312528       01-09-2003 15:15:28         9        8      15      14
439315533       01-09-2003 16:05:33       309      309    1256     886
439373634       02-09-2003 08:13:54       816      816    2191    2006
439377697       02-09-2003 09:21:37        11       10      15      12
439379264       02-09-2003 09:47:44       314      313     579     576
439379536       02-09-2003 09:52:16         8        6      14       9
439383940       02-09-2003 11:05:40        54       53      76      72
439384448       02-09-2003 11:14:08        13       12      19      17
..
439397689       02-09-2003 14:54:49        45       38     131     109
..
439641120       05-09-2003 10:32:00         5        5      13       9
..
439906946       08-09-2003 12:22:26        11       14      14      19
..
440598911       16-09-2003 12:35:11       373      380     439     445
..
441988830       01-10-2003 14:40:30        14       17      19      22
```

Table 6.1. **The first updates on the Arnhem (Gelderland) database after 1 Sept. ’03 (and a number of selected updates as depicted in the figures on following pages).**

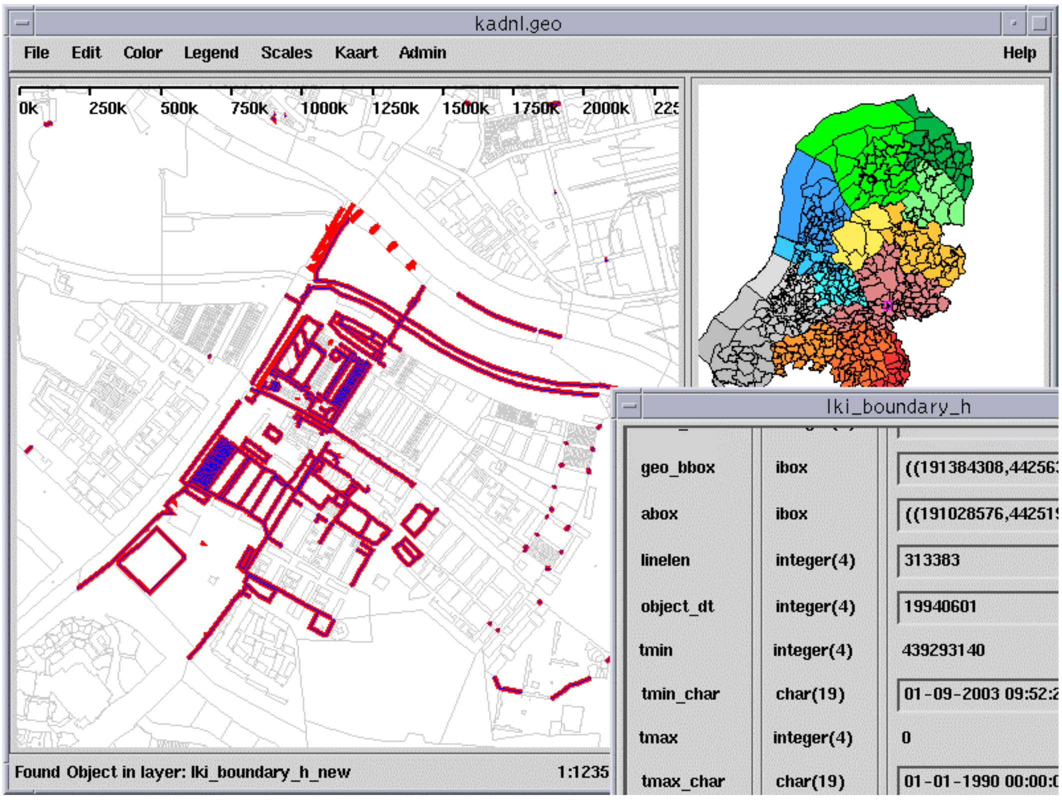
The integer time id (first column in Table 6.1) can be used as a unique identifier for the update (as all check-ins are sequential and have a different time stamp). It can be observed that the number of involved parcels and boundaries does vary quite a lot.

Usually the number of objects does grow a little bit, but there are also examples were the number of objects is shrinking. Figures 6.1 until 6.8 show a number of selected edit actions (corresponding to the red rows in Table 6.1).



geo_bbox	ibox	((187473400,437685720),(187735
abox	ibox	((187270381,437518692),(187751
linelen	integer(4)	262170
object_dt	integer(4)	20021113
tmin	integer(4)	439290688
tmin_char	char(19)	01-09-2003 09:11:28
tmax	integer(4)	0
tmax_char	char(19)	01-01-1990 00:00:00
sel_cd	char(3)	B

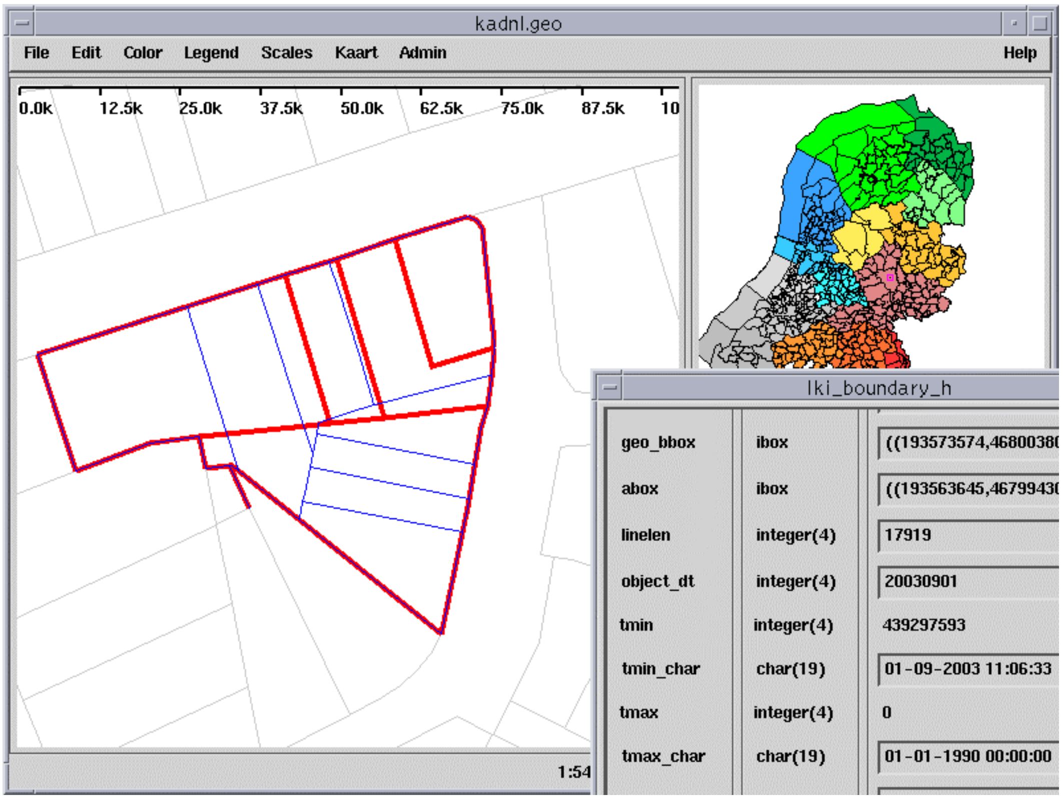
Figure 6.1 **Edit '439290688' with an increased number of parcels (note red boundaries are deleted and blue boundaries are new).**



Found Object in layer: lki_boundary_h_new 1:1235

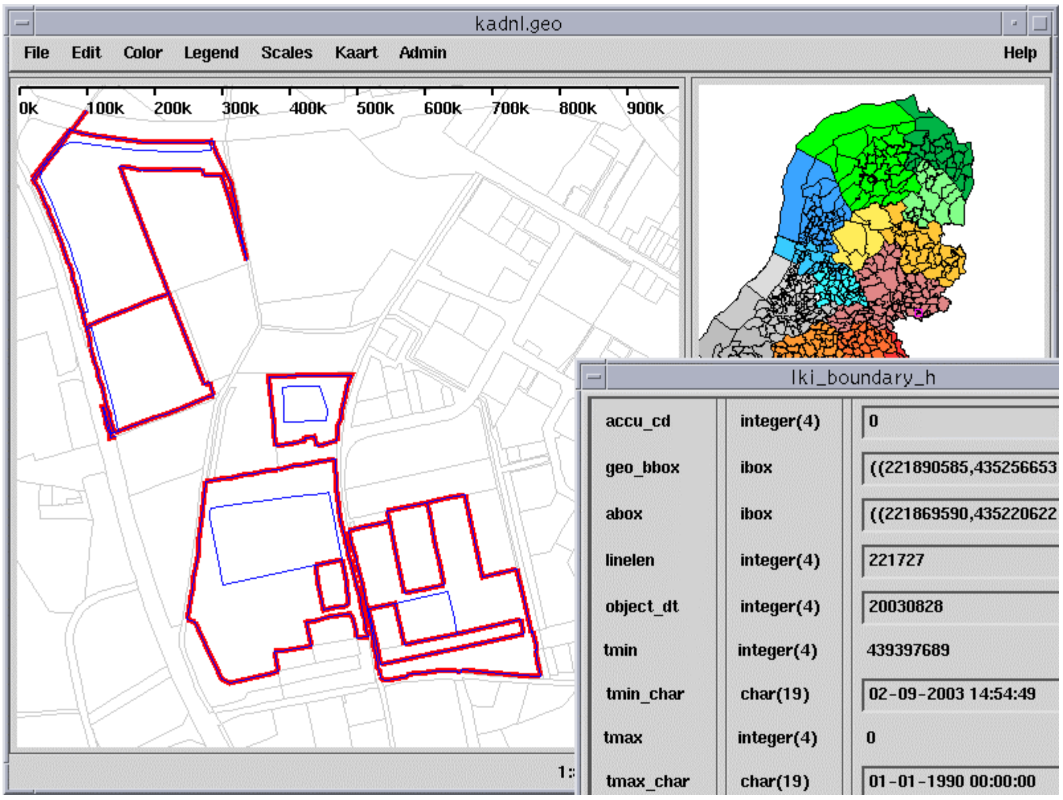
geo_bbox	ibox	((191384308,44256
abox	ibox	((191028576,44251
linelen	integer(4)	313383
object_dt	integer(4)	19940601
tmin	integer(4)	439293140
tmin_char	char(19)	01-09-2003 09:52:
tmax	integer(4)	0
tmax_char	char(19)	01-01-1990 00:00:

Figure 6.2 **Edit '439293140' with an equal number of parcels, but with a reduced number of boundaries as they have been joined (note red is deleted and blue is new).**



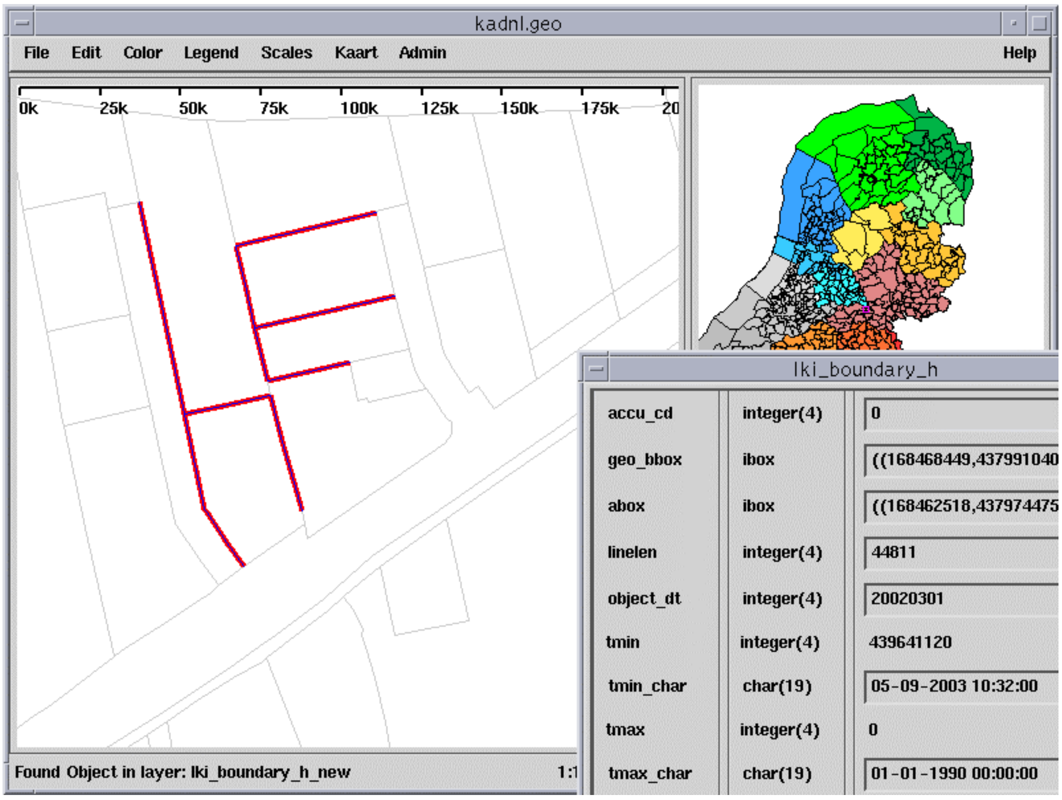
geo_bbox	ibox	((193573574,4680038
abox	ibox	((193563645,4679943
linelen	integer(4)	17919
object_dt	integer(4)	20030901
tmin	integer(4)	439297593
tmin_char	char(19)	01-09-2003 11:06:33
tmax	integer(4)	0
tmax_char	char(19)	01-01-1990 00:00:00

Figure 6.3 Edit '439297593' with an increased number of parcels (note red boundaries are deleted and blue boundaries are new).



accu_cd	integer(4)	0
geo_bbox	ibox	((221890585,435256653
abox	ibox	((221869590,435220622
linelen	integer(4)	221727
object_dt	integer(4)	20030828
tmin	integer(4)	439397689
tmin_char	char(19)	02-09-2003 14:54:49
tmax	integer(4)	0
tmax_char	char(19)	01-01-1990 00:00:00

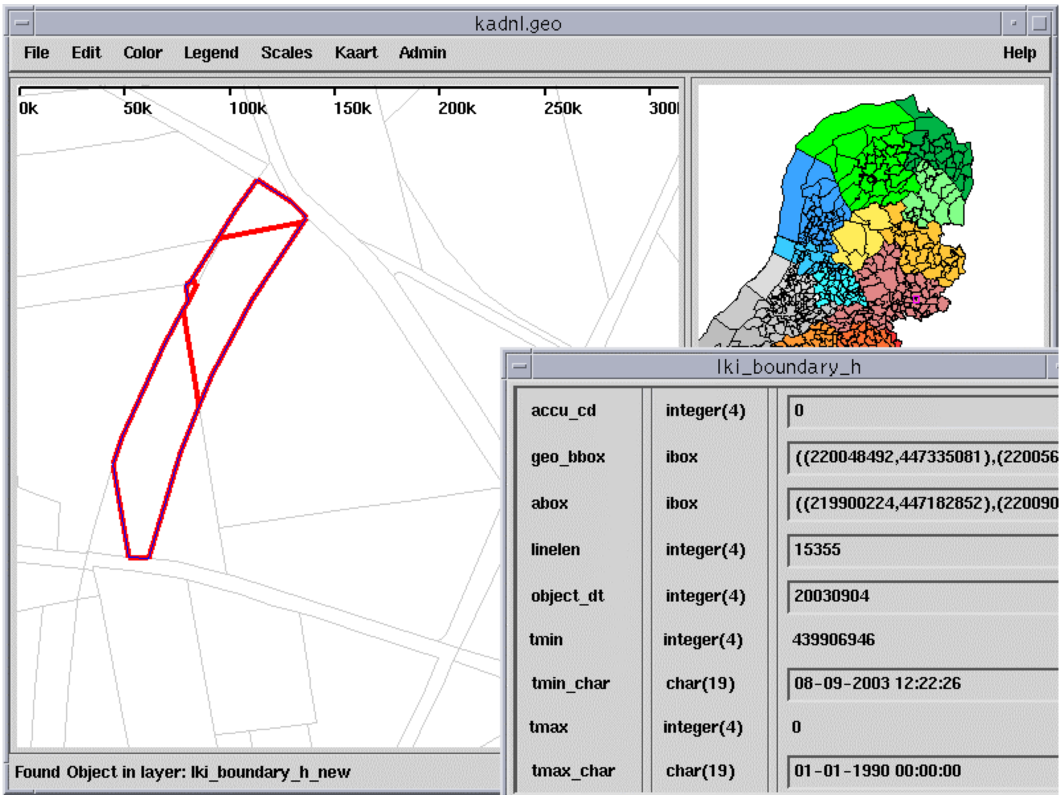
Figure 6.4 Edit '439397689' again with an increased number of parcels (note red boundaries are deleted and blue boundaries are new).



Found Object in layer: lki_boundary_h_new

accu_cd	integer(4)	0
geo_bbox	ibox	((168468449,437991040
abox	ibox	((168462518,437974475
linelen	integer(4)	44811
object_dt	integer(4)	20020301
tmin	integer(4)	439641120
tmin_char	char(19)	05-09-2003 10:32:00
tmax	integer(4)	0
tmax_char	char(19)	01-01-1990 00:00:00

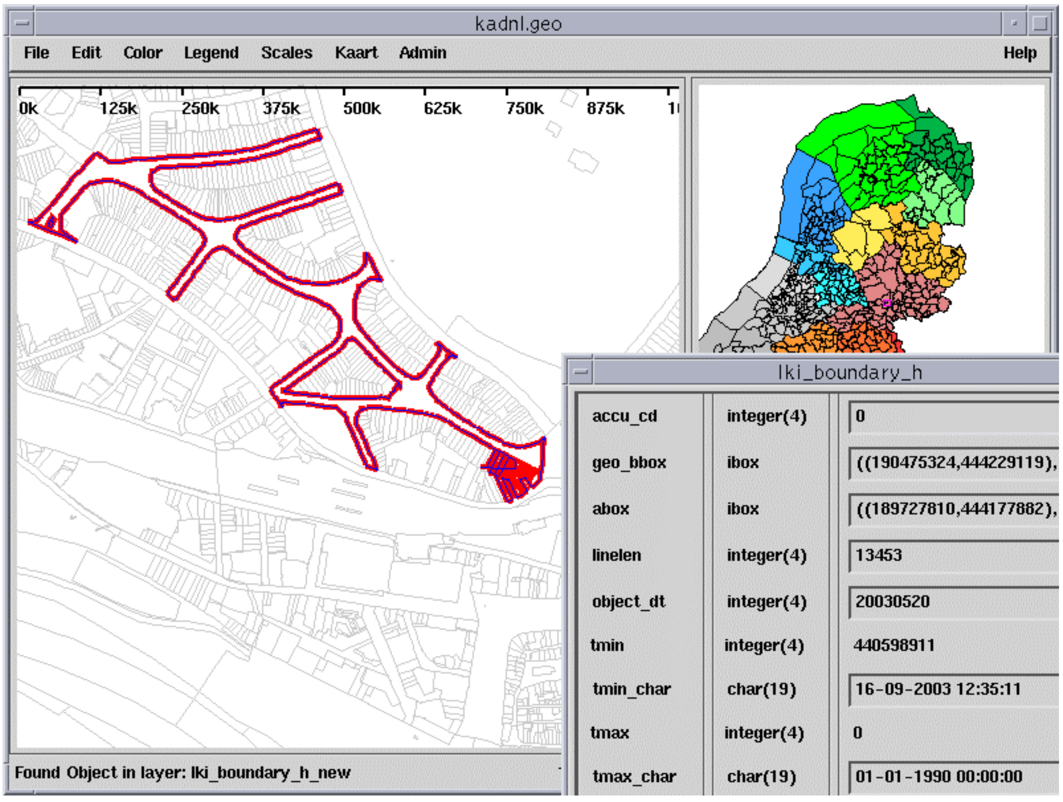
Figure 6.5 Edit '439641120' with an equal number of parcels, but with a reduced number of boundaries as they have been joined (note red boundaries are deleted).



Found Object in layer: lki_boundary_h_new

accu_cd	integer(4)	0
geo_bbox	ibox	((220048492,447335081),(220056
abox	ibox	((219900224,447182852),(220090
linelen	integer(4)	15355
object_dt	integer(4)	20030904
tmin	integer(4)	439906946
tmin_char	char(19)	08-09-2003 12:22:26
tmax	integer(4)	0
tmax_char	char(19)	01-01-1990 00:00:00

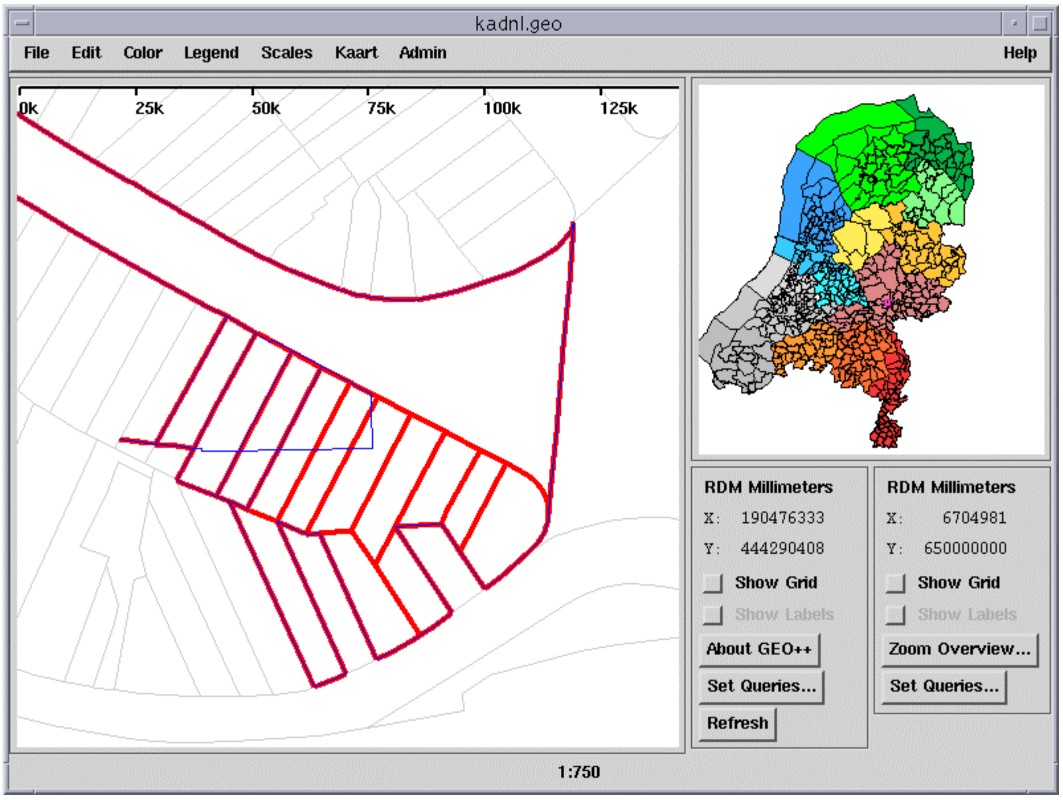
Figure 6.6 Edit '439906946' with a lower number of parcels, as they have been merged (note red boundaries are deleted).



lki_boundary_h

accu_cd	integer(4)	0
geo_bbox	ibox	((190475324,444229119),
abox	ibox	((189727810,444177882),
linelen	integer(4)	13453
object_dt	integer(4)	20030520
tmin	integer(4)	440598911
tmin_char	char(19)	16-09-2003 12:35:11
tmax	integer(4)	0
tmax_char	char(19)	01-01-1990 00:00:00

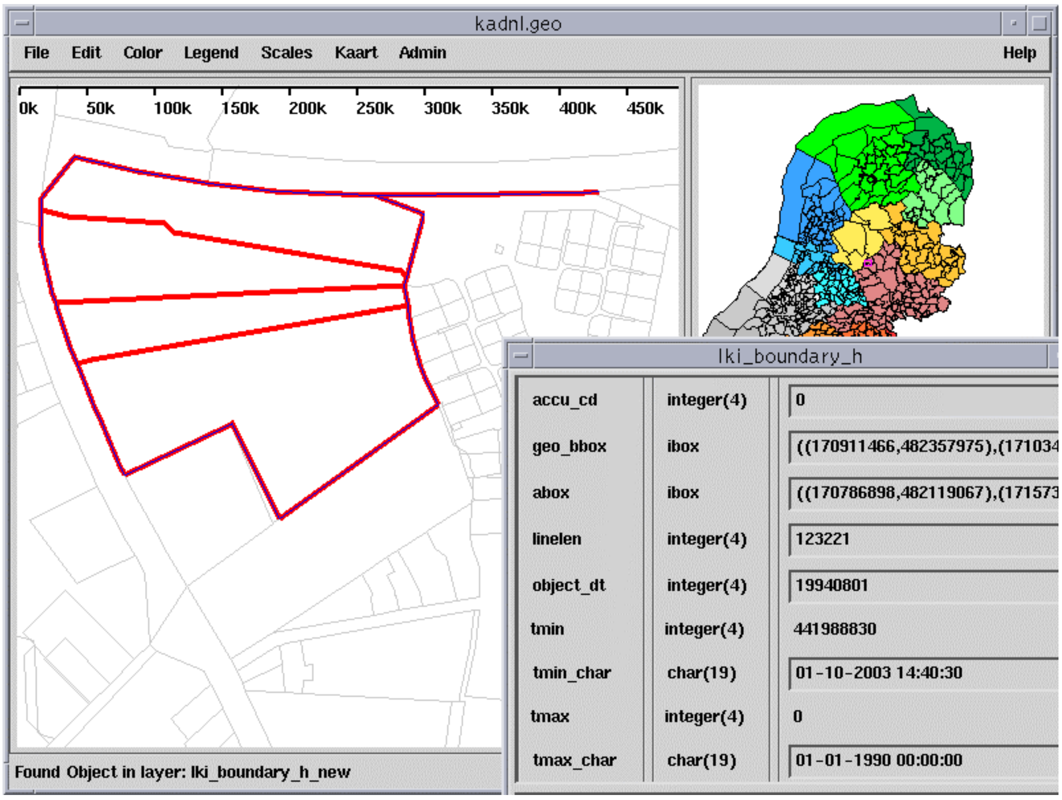
Found Object in layer: lki_boundary_h_new



RDM Millimeters	RDM Millimeters
X: 190476333	X: 6704981
Y: 444290408	Y: 650000000

1:750

Figure 6.7 **Edit '440598911' with a lower number of parcels, as they have been reorganized and merged, especially in the lower-right corner (note red boundaries are deleted and blue boundaries are new). Top: overview, Bottom: detail.**



Found Object in layer: lki_boundary_h_new

accu_cd	integer(4)	0
geo_bbox	ibox	((170911466,482357975),(171034
abox	ibox	((170786898,482119067),(171573
linelen	integer(4)	123221
object_dt	integer(4)	19940801
tmin	integer(4)	441988830
tmin_char	char(19)	01-10-2003 14:40:30
tmax	integer(4)	0
tmax_char	char(19)	01-01-1990 00:00:00

Figure 6.8 Edit ‘441988830’ with again a lower number of parcels, as they have been merged (note red boundaries are deleted).

7 Conclusions

In this research we defined many cadastral data integrity checks (of which the majority does have a topological nature). To our surprise these checks did indeed find errors in the cadastral production data (date 1 Sept. '03), indicating that these constraints should not only be implemented in the front-end and middleware (check-in), but also (and foremost) in the DBMS. The large cadastral office (province of Gelderland), did turn out to have a low number of errors (only 4 topology errors, which were corrected by hand and few errors with circular arcs, which were in fact straight lines).

Loading the data in the Oracle 10g topology and ESRI Geodatabase did reveal a number of interesting issues:

- Circular arcs are not supported in Oracle Topology and must therefore be stroked to polylines (an ESRI Geodatabase with topology does support circular arcs);
- Oracle 10g Topology needs explicit nodes and associated topology relationships (and these have to be derived by scripts as the cadastral LKI model does not use nodes);
- Extracting a topological subset from a larger topological structure requires adjusting the references at the boundary;
- Though the original LKI data is based on (32 bits) integer coordinates, due to stroking, more digits are needed and the result is that these do not fit into a 32 bit integer (able to address the whole domain of the Netherlands). Solutions are: using only a subset of the whole domain (e.g. fit tightly around province of Gelderland) or the new version of ESRI Geodatabase (able representing more accurate that 32 bit coordinates).
- Validation functions in Oracle 10g Topology are not suitable for cleaning data, but are very useful in maintaining a valid topology. ESRI Geodatabase works with more (user defined) rules and is more suitable for cleaning data.

Future work, of course, includes executing the selection queries and update statements in the different storage environments. And to make things even more interesting also parallel queries and updates should be tested.

References

- *ArcGIS: Working with Geodatabase Topology, An ESRI White Paper*, May 2003. Available at: http://www.esri.com/library/whitepapers/pdfs/geodatabase-topology.pdf.

- Oosterom, P. van and C. Lemmen (2001). '*Spatial Data-management on a very large cadastral Database*' Computers, Environmnet and Urban Systems, Volume 25, Nr. 4-5, pp. 509-528.

- *Oracle Spatial Topology and Network Data Models, 10g Release 1 (10.1)*, Oracle Corp. 2003. Available at http://otn.oracle.com.

- Penninga, F., *Oracle 10g Topology; Testing Oracle 10g Topology using cadastral data*, GISt Report No. 26, Delft, 2004, 48 p.

- Tijssen, T.P.M., M.E. de Vries and P.J.M. van Oosterom, *Comparing the storage of Shell data in Oracle SDO_GEOMETRY and ArcSDE SDEBINARY formats*, Delft University of Technology, GISt Report No.13, Report to Shell International Exploration and Production B.V., Delft, 2002, 77 p.

The papers and reports from the authors are available at
http://www.gdmc.nl/publications

Appendix 1 Consistency checks (topological data quality)

```
# Script to analyse the quality of topology references in LKI


drop table vest;
create table vest(min_id integer, max_id integer, kode char(4));
insert into vest values(         1, 30000000,'GN');
insert into vest values(  30000001, 60000000,'LA');
insert into vest values(  60000001, 90000000,'AS');
insert into vest values(  90000001,120000000,'ZL');
insert into vest values( 120000001,140000000,'LL');
insert into vest values( 140000001,200000000,'AH');
insert into vest values( 200000001,230000000,'UT');
insert into vest values( 230000001,260000000,'AD');
insert into vest values( 260000001,290000000,'AM');
insert into vest values( 290000001,310000000,'AMg');
insert into vest values( 310000001,340000000,'GV');
insert into vest values( 340000001,370000000,'RT');
insert into vest values( 370000001,390000000,'MD');
insert into vest values( 390000001,420000000,'BD');
insert into vest values( 420000001,450000000,'EH');
insert into vest values( 450000001,470000000,'EHg');
insert into vest values( 470000001,500000000,'RM');

/* object_id = 201775411 and straight */
/* object_id = 311829573 and closed */
/* find closesd arcs */
drop table closed_circ;
create table closed_circ as
select object_id, numpoints(shape),
anypoint(shape, 1), anypoint(shape, 2), anypoint(shape, 3)
FROM xfio_boundary WHERE
numpoints(shape)=3 and interp_cd=3 and tmax = 0 and ogroup = 6 and
(anypoint(shape, 1) =  anypoint(shape, 3));

/* find straight 'arcs' */
drop table straight_arc;
create table straight_arc as
select object_id, numpoints(shape),
anypoint(shape, 1), anypoint(shape, 2), anypoint(shape, 3)
FROM xfio_boundary WHERE
numpoints(shape)=3 and interp_cd=3 and tmax = 0 and ogroup = 6 and
(float8(Point_x(anypoint(shape, 2)))-float8(Point_x(anypoint(shape, 1))))*
(float8(Point_y(anypoint(shape, 3)))-float8(Point_y(anypoint(shape, 2))))=
(float8(Point_x(anypoint(shape, 3)))-float8(Point_x(anypoint(shape, 2))))*
(float8(Point_y(anypoint(shape, 2)))-float8(Point_y(anypoint(shape, 1))));
```

```
/* investigate the quality of the empty references at the vest. grens */
/* two options municip is '     ' or object_id=0 */

select count(*) from lki_boundary where l_municip='     ';
select count(*) from lki_boundary where l_obj_id=0;
select count(*) from lki_boundary where l_obj_id is null;
select count(*) from lki_boundary where l_municip='     ' and l_obj_id=0;
drop table vestgrens_l;
create table vestgrens_l as
select object_id from lki_boundary
where (l_municip='     ' and l_obj_id<>0) or
      (l_municip<>'     ' and l_obj_id=0);

select count(*) from lki_boundary where r_municip='     ';
select count(*) from lki_boundary where r_obj_id=0;
select count(*) from lki_boundary where r_obj_id is null;
select count(*) from lki_boundary where r_municip='     ' and r_obj_id=0;
drop table vestgrens_r;
create table vestgrens_r as
select object_id from lki_boundary
where (r_municip='     ' and r_obj_id<>0) or
      (r_municip<>'     ' and r_obj_id=0);

select count(*), classif from lki_boundary group by classif;

drop table l_notexits;
create table l_notexits as select l_obj_id from lki_boundary
  where l_obj_id not in (select object_id from lki_parcel);

drop table l_diff;
create table l_diff as
select p.object_id, b.l_obj_id
from lki_boundary b, lki_parcel p
where
b.l_municip=p.municip and b.l_section=p.osection and b.l_parcel=p.parcel
and b.l_obj_id<>p.object_id;

drop table l_diff2;
create table l_diff2 as
select p.object_id, b.l_obj_id
from lki_boundary b, lki_parcel p
where
not(b.l_municip=p.municip and b.l_section=p.osection and
b.l_parcel=p.parcel)
and b.l_obj_id=p.object_id;

drop table r_notexits;
create table r_notexits
as select r_obj_id
from lki_boundary
where r_obj_id not in (select object_id from lki_parcel);
```

```
drop table r_diff;
create table r_diff as
select p.object_id, b.r_obj_id
from lki_boundary b, lki_parcel p
where
b.r_municip=p.municip and b.r_section=p.osection and b.r_parcel=p.parcel and
b.r_obj_id<>p.object_id;

drop table r_diff2;
create table r_diff2 as
select p.object_id, b.r_obj_id
from lki_boundary b, lki_parcel p
where
not(b.r_municip=p.municip and b.r_section=p.osection and
b.r_parcel=p.parcel)
and b.r_obj_id=p.object_id;

/* check boundary-boundary references fl, ll, fr, lr */
drop table fl_notexits;
create table fl_notexits
as select object_id, fl_line_id
from lki_boundary
where abs(fl_line_id) not in (select object_id from lki_boundary);

drop table ll_notexits;
create table ll_notexits
as select object_id, ll_line_id
from lki_boundary
where abs(ll_line_id) not in (select object_id from lki_boundary);

drop table fr_notexits;
create table fr_notexits
as select object_id, fr_line_id
from lki_boundary
where abs(fr_line_id) not in (select object_id from lki_boundary);

drop table lr_notexits;
create table lr_notexits
as select object_id, lr_line_id
from lki_boundary
where abs(lr_line_id) not in (select object_id from lki_boundary);

drop table bnd_notexits;
create table bnd_notexits
as select object_id, line_id1
from lki_parcel
where abs(line_id1) not in (select object_id from lki_boundary);

drop table akr_niet_lki_wel;
create table akr_niet_lki_wel as
select count(*),municip from lki_parcel
```

```
where x_akr_objectnummer not in
  (select x_akr_objectnummer from mo_object)
group by municip;
select * from akr_niet_lki_wel;

drop table akr_wel_lki_niet;
create table akr_wel_lki_niet as
select count(*),municip from mo_object
where pp_i_ltr='G' and x_akr_objectnummer not in
  (select x_akr_objectnummer from lki_parcel)
group by municip;
select * from akr_wel_lki_niet;

drop table akr_niet_lki_wel_vest;
create table akr_niet_lki_wel_vest as
select sum(col1), vestigings_code
from akr_niet_lki_wel, kad_burg_gemeente_cat_iqt
where municip=kadastrale_gem_code
group by vestigings_code;
select * from akr_niet_lki_wel_vest;

drop table akr_wel_lki_niet_vest;
create table akr_wel_lki_niet_vest as
select sum(col1), vestigings_code
from akr_wel_lki_niet, kad_burg_gemeente_cat_iqt
where municip=kadastrale_gem_code
group by vestigings_code;
select * from akr_wel_lki_niet_vest;

drop table box_location;
create table box_location as
select object_id from lki_parcel
where inside(location, geo_bbox) != 1;

drop table kadgem_aant;
create table kadgem_aant as
select count(*), municip
from lki_parcel
group by municip;
select * from kadgem_aant;

drop table burggem_aant;
create table burggem_aant as
select count(*), burg_gemeente_code
from lki_parcel, kad_burg_gemeente_cat_iqt
where municip=kadastrale_gem_code
group by burg_gemeente_code;
select * from burggem_aant;

/* check boundary-boundary references fl, ll, fr, lr (geometry connected) */
drop table fl_notcor_pos;
create table fl_notcor_pos
```

```
as select s.object_id, s.fl_line_id
from xfio_boundary s, xfio_boundary r
where s.fl_line_id > 0 and s.fl_line_id=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(anypoint(s.shape, 1) <>  anypoint(r.shape, 1));

drop table fl_notcor_neg;
create table fl_notcor_neg
as select s.object_id, s.fl_line_id
from xfio_boundary s, xfio_boundary r
where s.fl_line_id < 0 and (-1 * s.fl_line_id)=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(anypoint(s.shape, 1) <>  anypoint(r.shape, numpoints(r.shape)));

drop table fr_notcor_pos;
create table fr_notcor_pos
as select s.object_id, s.fr_line_id
from xfio_boundary s, xfio_boundary r
where s.fr_line_id > 0 and s.fr_line_id=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(anypoint(s.shape, 1) <>
 anypoint(r.shape, 1));

drop table fr_notcor_neg;
create table fr_notcor_neg
as select s.object_id, s.fr_line_id
from xfio_boundary s, xfio_boundary r
where s.fr_line_id < 0 and (-1 * s.fr_line_id)=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(anypoint(s.shape, 1) <>
 anypoint(r.shape, numpoints(r.shape)));

drop table ll_notcor_pos;
create table ll_notcor_pos
as select s.object_id, s.ll_line_id
from xfio_boundary s, xfio_boundary r
where s.ll_line_id > 0 and s.ll_line_id=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(anypoint(s.shape, numpoints(s.shape)) <>
 anypoint(r.shape, 1));

drop table ll_notcor_neg;
create table ll_notcor_neg
as select s.object_id, s.ll_line_id
from xfio_boundary s, xfio_boundary r
where s.ll_line_id < 0 and (-1 * s.ll_line_id)=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(anypoint(s.shape, numpoints(s.shape)) <>
 anypoint(r.shape, numpoints(r.shape)));

drop table lr_notcor_pos;
create table lr_notcor_pos
```

```
as select s.object_id, s.lr_line_id
from xfio_boundary s, xfio_boundary r
where s.lr_line_id > 0 and s.lr_line_id=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(anypoint(s.shape, numpoints(s.shape)) <>
 anypoint(r.shape, 1));

drop table lr_notcor_neg;
create table lr_notcor_neg
as select s.object_id, s.lr_line_id
from xfio_boundary s, xfio_boundary r
where s.lr_line_id < 0 and (-1 * s.lr_line_id)=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(anypoint(s.shape, numpoints(s.shape)) <>
 anypoint(r.shape, numpoints(r.shape)));

drop table parbnd_notcor_pos;
create table parbnd_notcor_pos
as select s.object_id, s.line_id1
from xfio_parcel s, xfio_boundary r
where s.line_id1 > 0 and s.line_id1=r.object_id and
s.tmax=0 and s.ogroup=46 and r.tmax=0 and r.ogroup=6 and
(s.object_id <>  r.r_obj_id);

drop table parbnd_notcor_neg;
create table parbnd_notcor_neg
as select s.object_id, s.line_id1
from xfio_parcel s, xfio_boundary r
where s.line_id1 < 0 and -1*s.line_id1=r.object_id and
s.tmax=0 and s.ogroup=46 and r.tmax=0 and r.ogroup=6 and
(s.object_id <>  r.l_obj_id);

drop table parbnd2_notcor_pos;
create table parbnd2_notcor_pos
as select s.object_id, s.line_id2
from xfio_parcel s, xfio_boundary r
where s.line_id2 > 0 and s.line_id2=r.object_id and
s.tmax=0 and s.ogroup=46 and r.tmax=0 and r.ogroup=6 and
(s.object_id <>  r.r_obj_id);

drop table parbnd2_notcor_neg;
create table parbnd2_notcor_neg
as select s.object_id, s.line_id2
from xfio_parcel s, xfio_boundary r
where s.line_id2 < 0 and -1*s.line_id2=r.object_id and
s.tmax=0 and s.ogroup=46 and r.tmax=0 and r.ogroup=6 and
(s.object_id <>  r.l_obj_id);

drop table fl_notsame_pos;
create table fl_notsame_pos
as select s.object_id, s.fl_line_id
from xfio_boundary s, xfio_boundary r
```

```
where s.fl_line_id > 0 and s.fl_line_id=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
s.r_obj_id <> r.l_obj_id;

drop table fl_notsame_neg;
create table fl_notsame_neg
as select s.object_id, s.fl_line_id
from xfio_boundary s, xfio_boundary r
where s.fl_line_id < 0 and (-1 * s.fl_line_id)=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(s.r_obj_id <>  r.r_obj_id);

drop table fr_notsame_pos;
create table fr_notsame_pos
as select s.object_id, s.fr_line_id
from xfio_boundary s, xfio_boundary r
where s.fr_line_id > 0 and s.fr_line_id=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(s.l_obj_id <>  r.r_obj_id);

drop table fr_notsame_neg;
create table fr_notsame_neg
as select s.object_id, s.fr_line_id
from xfio_boundary s, xfio_boundary r
where s.fr_line_id < 0 and (-1 * s.fr_line_id)=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(s.l_obj_id <>  r.l_obj_id);

drop table ll_notsame_pos;
create table ll_notsame_pos
as select s.object_id, s.ll_line_id
from xfio_boundary s, xfio_boundary r
where s.ll_line_id > 0 and s.ll_line_id=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(s.l_obj_id <>  r.l_obj_id);

drop table ll_notsame_neg;
create table ll_notsame_neg
as select s.object_id, s.ll_line_id
from xfio_boundary s, xfio_boundary r
where s.ll_line_id < 0 and (-1 * s.ll_line_id)=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(s.l_obj_id <>  r.r_obj_id);

drop table lr_notsame_pos;
create table lr_notsame_pos
as select s.object_id, s.lr_line_id
from xfio_boundary s, xfio_boundary r
where s.lr_line_id > 0 and s.lr_line_id=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(s.r_obj_id <>  r.r_obj_id);
```

```
drop table lr_notsame_neg;
create table lr_notsame_neg
as select s.object_id, s.lr_line_id
from xfio_boundary s, xfio_boundary r
where s.lr_line_id < 0 and (-1 * s.lr_line_id)=r.object_id and
s.tmax=0 and s.ogroup=6 and r.tmax=0 and r.ogroup=6 and
(s.r_obj_id <>  r.l_obj_id);

/* check island refs. first collect in one table all island ref's */

drop table island_refs;
create table island_refs as
select object_id, line_id2 as bnd_ref
from xfio_parcel
where tmax=0 and ogroup=46 and l_num>1;

insert into island_refs select object_id, line_id1
from xfio_parcelover where tmax=0 and ogroup=46 and line_id1 <>0;
insert into island_refs select object_id, line_id2
from xfio_parcelover where tmax=0 and ogroup=46 and line_id2 <>0;
insert into island_refs select object_id, line_id3
from xfio_parcelover where tmax=0 and ogroup=46 and line_id3 <>0;
insert into island_refs select object_id, line_id4
from xfio_parcelover where tmax=0 and ogroup=46 and line_id4 <>0;
insert into island_refs select object_id, line_id5
from xfio_parcelover where tmax=0 and ogroup=46 and line_id5 <>0;
insert into island_refs select object_id, line_id6
from xfio_parcelover where tmax=0 and ogroup=46 and line_id6 <>0;
insert into island_refs select object_id, line_id7
from xfio_parcelover where tmax=0 and ogroup=46 and line_id7 <>0;
insert into island_refs select object_id, line_id8
from xfio_parcelover where tmax=0 and ogroup=46 and line_id8 <>0;
insert into island_refs select object_id, line_id9
from xfio_parcelover where tmax=0 and ogroup=46 and line_id9 <>0;
insert into island_refs select object_id, line_id10
from xfio_parcelover where tmax=0 and ogroup=46 and line_id10 <>0;

drop table island_refs_cnt;
create table island_refs_cnt as
select object_id, count(*) as isl_num
from island_refs
group by object_id;

/* check if the number of island references is correct comapred to l_num */
drop table island_cnt_err;
create table island_cnt_err as
select p.object_id, l_num, isl_num
from xfio_parcel p, island_refs_cnt i
where p.object_id=i.object_id and p.tmax=0 and p.ogroup=46 and
  l_num <> (isl_num+1);

/* check if all references to island boundaries do exist */
```

```
drop table isl_notexits;
create table isl_notexits
as select object_id, bnd_ref
from island_refs
where abs(bnd_ref) not in
  (select object_id from xfio_boundary where ogroup=6 and tmax=0);

/* check if island has proper parcel at right side in case of + reference */
drop table islbnd_notcor_pos;
create table islbnd_notcor_pos
as select s.object_id, s.bnd_ref
from island_refs s, xfio_boundary r
where s.bnd_ref > 0 and s.bnd_ref=r.object_id and r.tmax=0 and r.ogroup=6
  and (s.object_id <>  r.r_obj_id);

/* check if island has proper parcel at left side in case of - reference */
drop table islbnd_notcor_neg;
create table islbnd_notcor_neg
as select s.object_id, s.bnd_ref
from island_refs s, xfio_boundary r
where s.bnd_ref < 0 and -1*s.bnd_ref=r.object_id and r.tmax=0 and r.ogroup=6
  and (s.object_id <>  r.l_obj_id);

/* check if first coord of island is within bbox of parcel */
drop table isl_loc_not_inparcel;
create table isl_loc_not_inparcel
as select s.object_id, s.bnd_ref
from island_refs s, xfio_parcel r, xfio_boundary b
where s.object_id = r.object_id and s.bnd_ref=b.object_id
  and r.tmax=0 and r.ogroup=46 and b.tmax=0 and b.ogroup=6
  and inside(anypoint(b.shape, 1), r.bbox) != 1;

EOF
```

Appendix 2 Finding the updates after 1 Sept. ’03

```
# Script to analyse the updates (after 1 sept '03)

LKIDATETIME=${2:-'01-09-2003 00:00:00'}

drop table parcel_upd_new;
create table parcel_upd_new as
select count(*) as num_parc, tmin
from xfio_parcel
where tmin > lkidate2int('${LKIDATETIME}') and
object_id > 140000001 and object_id < 200000000 and ogroup=46
group by tmin;
create index pnew_idx on parcel_upd_new(tmin);

drop table parcel_upd_old;
create table parcel_upd_old as
select count(*) as num_parc, tmax
from xfio_parcel
where tmax > lkidate2int('${LKIDATETIME}') and
object_id > 140000001 and object_id < 200000000 and ogroup=46
group by tmax;
create index pold_idx on parcel_upd_old(tmax);

drop table boundary_upd_new;
create table boundary_upd_new as
select count(*) as num_bnd, tmin
from xfio_boundary
where tmin > lkidate2int('${LKIDATETIME}') and
object_id > 140000001 and object_id < 200000000 and ogroup=46
group by tmin;
create index bnew_idx on boundary_upd_new(tmin);

drop table boundary_upd_old;
create table boundary_upd_old as
select count(*) as num_bnd, tmax
from xfio_boundary
where tmax > lkidate2int('${LKIDATETIME}') and
object_id > 140000001 and object_id < 200000000 and ogroup=46
group by tmax;
create index bold_idx on boundary_upd_old(tmax);

drop table mutations;
create table mutations as
select pnew.tmin, lkiint2date(pnew.tmin),
  pnew.num_parc as new_parc, pold.num_parc as old_parc,
  bnew.num_bnd as new bnd, bold.num_bnd as old_bnd
from parcel_upd_new pnew, parcel_upd_old pold,
  boundary_upd_new as bnew, boundary_upd_old as bold
where pnew.tmin=pold.tmax and pnew.tmin=bnew.tmin and
pnew.tmin=bold.tmax;

EOF
```

Appendix 3 LKI to Oracle topology conversion

Topology creation script

```
-- Script to retrieve Dutch cadastral (LKI) parcel topology and
-- transform it into Oracle topology. This version is for the
-- 300K subset.
--
-- The input data set consists of three views and a table holding
-- (a subset of) the LKI cadastral parcels, these are:
--      name_BOUNDARY: LKI boundaries          (view on xfio_boundary)
--        name_PARCEL: LKI parcel information (view on xfio_parcel)
--    name_PARCELOVER: island references      (view on xfio_parcelover)
--       name_MUNICIP: table containing names of cadastral
--                     municipalities in dataset
--
-- Note that it is assumed that the dataset consists of a single,
-- contiguous area (so no holes and no disconnected areas) and that
-- the input topological references are correct and complete.

spool topol_SET300K.log
set pages 500
set lines 120
set trim on
set trimspool on
set serveroutput on
set echo on

--
-- Make sure we have a clean start
--
execute sdo_topo.delete_topo_geometry_layer
  ('KADASTER_SET300K','PARCELS_SET300K','FEATURE');
execute sdo_topo.drop_topology ('KADASTER_SET300K');

drop table TMP_VERTS_SET300K purge;
drop table TMP_VERTICES_SET300K purge;
drop table TMP_BOUNDARY_SET300K purge;

delete from user_sdo_geom_metadata where
  table_name='TMP_VERTICES_SET300K';
commit;
select * from user_sdo_topo_info;
select * from user_sdo_topo_metadata;

set timing on

-- Create temp boundary table from view
-- (must be updated in case dataset is subset of complete office)
--
create table TMP_BOUNDARY_SET300K as select * from SET300K_BOUNDARY;

-- Set references at perimeter of dataset to exterior face
-- (only for subset)
--
update TMP_BOUNDARY_SET300K
  set l_obj_id = -1
  where l_municip not in (select municip from SET300K_MUNICIP);
commit;
update TMP_BOUNDARY_SET300K
  set r_obj_id = -1
  where r_municip not in (select municip from SET300K_MUNICIP);
commit;

-- Create new topology
--
execute sdo_topo.create_topology ('KADASTER_SET300K',0.002);

--
-- Load edges
--

-- EDGE$: EDGE_ID, START_NODE_ID, END_NODE_ID, NEXT_LEFT_EDGE_ID,
-- PREV_LEFT_EDGE_ID, NEXT_RIGHT_EDGE_ID, PREV_RIGHT_EDGE_ID,
-- LEFT_FACE_ID, RIGHT_FACE_ID, GEOMETRY
--
-- Start_node and end_node are updated later
--
insert into KADASTER_SET300K_EDGE$
  select object_id, null, null, ll_line_id, -fr_line_id, fl_line_id,
         -lr_line_id, l_obj_id, r_obj_id, geo_polyline
  from TMP_BOUNDARY_SET300K;
commit;

--
-- Load nodes
--

-- LKI does not use nodes, the procedure to derive the nodes from LKI
-- edges is relatively complex (and can probably be done more
-- efficient)

-- First create a temp table with all start- and endpoints of edges
--
create table TMP_VERTICES_SET300K
(
  id       number,
  newid    number,
  geometry mdsys.sdo_geometry,
  fromedge number,
  isstart  number
);

declare
  rownmbr    number := 1;
  start_node mdsys.sdo_geometry :=
mdsys.sdo_geometry(2001,null,mdsys.sdo_point_type(0,0,null),null,null);
  end_node   mdsys.sdo_geometry :=
mdsys.sdo_geometry(2001,null,mdsys.sdo_point_type(0,0,null),null,null);
begin
  for e in
    (select geometry, edge_id from KADASTER_SET300K_EDGE$)
  loop
    start_node.sdo_point.x := e.geometry.sdo_ordinates(1);
    start_node.sdo_point.y := e.geometry.sdo_ordinates(2);
    end_node.sdo_point.x   :=
      e.geometry.sdo_ordinates(e.geometry.sdo_ordinates.last - 1);
    end_node.sdo_point.y   :=
      e.geometry.sdo_ordinates(e.geometry.sdo_ordinates.last);
    insert into TMP_VERTICES_SET300K values
      (rownmbr,-1,start_node,e.edge_id,0);
    rownmbr := rownmbr + 1;
    insert into TMP_VERTICES_SET300K values
      (rownmbr,-1,end_node,e.edge_id,1);
    rownmbr := rownmbr + 1;
  end loop;
end;
/
commit;
insert into user_sdo_geom_metadata values
('TMP_VERTICES_SET300K','GEOMETRY',
  mdsys.sdo_dim_array(
    mdsys.sdo_dim_element('X',-25000000,325000000,0.002),
    mdsys.sdo_dim_element('Y',275000000,650000000,0.002)), NULL);
commit;
analyze table TMP_VERTICES_SET300K compute statistics;
create index TMP_VERTICES_INDEX1_SET300K on TMP_VERTICES_SET300K
  (geometry) indextype is mdsys.spatial_index
  parameters ('sdo_fanout=46 layer_gtype=point tablespace=indx');
create index TMP_VERTICES_INDEX2_SET300K on TMP_VERTICES_SET300K (id)
  tablespace indx compute statistics;
create index TMP_VERTICES_INDEX3_SET300K on TMP_VERTICES_SET300K
  (fromedge) tablespace indx compute statistics;
call analyze_rtree ('TMP_VERTICES_INDEX1_SET300K');

-- Create another temp table for nodes and join all edge
-- start/endpoints at the same point into a single node
--
create table TMP_VERTS_SET300K
  as select max(b.id) id, a.id id2
  from TMP_VERTICES_SET300K a, TMP_VERTICES_SET300K b,
   table (sdo_join('TMP_VERTICES_SET300K','GEOMETRY',
'TMP_VERTICES_SET300K','GEOMETRY','mask=anyinteract')) c
  where c.rowid1 = a.rowid and c.rowid2 = b.rowid
  group by a.id
;
analyze table TMP_VERTS_SET300K compute statistics;
create index TMP_VERTS_INDEX1_SET300K on TMP_VERTS_SET300K (id2)
  tablespace indx compute statistics;

update TMP_VERTICES_SET300K v
  set newid = (select vt.id from TMP_VERTS_SET300K vt
    where vt.id2 = v.id);
commit;

-- Finally insert nodes in node table (edge_id will be updated later)
-- NODE$: NODE_ID, EDGE_ID, FACE_ID, GEOMETRY
--
insert into KADASTER_SET300K_NODE$
  select id, null, null, geometry
  from TMP_VERTICES_SET300K
  where id = newid;
commit;

-- Set start_node and end_node for edges
--
update KADASTER_SET300K_EDGE$ e set
  start_node_id =
    (select newid from TMP_VERTICES_SET300K v
       where v.fromedge=e.edge_id and v.isstart=0),
  end_node_id =
    (select newid from TMP_VERTICES_SET300K v
       where v.fromedge=e.edge_id and v.isstart=1);
commit;

create index TMP_SNODE_SET300K_IDX on KADASTER_SET300K_EDGE$
  (start_node_id) tablespace indx compute statistics;
create index TMP_ENODE_SET300K_IDX on KADASTER_SET300K_EDGE$
  (end_node_id) tablespace indx compute statistics;

-- Set edge references in node table
--
update KADASTER_SET300K_NODE$ n
  set edge_id =
    (select min(edge_id) from KADASTER_SET300K_EDGE$ e
      where e.start_node_id = n.node_id)
;
commit;
update KADASTER_SET300K_NODE$ n
  set edge_id =
    (select -min(edge_id) from KADASTER_SET300K_EDGE$ e
       where e.end_node_id = n.node_id)
  where edge_id is null;
```

```
commit;
drop index TMP_SNODE_SET300K_IDX;
drop index TMP_ENODE_SET300K_IDX;

--
-- Load faces
--

-- First add the exterior face
--
insert into KADASTER_SET300K_FACE$ values (-1,null,null,null,null);

-- Link the exterior face to an edge on the perimeter of the dataset
--
update KADASTER_SET300K_FACE$
  set island_edge_id_list = (
    select sdo_list_type(max(object_id)) from TMP_BOUNDARY_SET300K
    where l_municip not in (select municip from SET300K_MUNICIP)
  )
  where face_id = -1;
commit;

-- Add the other faces (island edge lists are updated later)
-- FACE$: FACE_ID, BOUNDARY_EDGE_ID, ISLAND_EDGE_ID_LIST,
--         ISLAND_NODE_ID_LIST, MBR_GEOMETRY
--
insert into KADASTER_SET300K_FACE$
  select object_id, -line_id1, null, null, geo_bbox
  from SET300K_PARCEL;
commit;

-- Function to retrieve LKI island references
--
create or replace function find_islands (parcel_id in number)
  return sdo_list_type
is
  island1 number;
  rval    sdo_list_type := sdo_list_type();
begin
  select line_id2 into island1 from SET300K_PARCEL
    where object_id = parcel_id;
  if (island1 <> 0) then
    rval.extend;
    rval(rval.last) := -island1;

    for lref in
      (select line_id1,line_id2,line_id3,line_id4,line_id5,line_id6,
              line_id7,line_id8,line_id9,line_id10
         from SET300K_PARCELOVER where object_id = parcel_id)
    loop
      if (lref.line_id1 <> 0) then
        rval.extend;
        rval(rval.last) := -lref.line_id1;
        if (lref.line_id2 <> 0) then
          rval.extend;
          rval(rval.last) := -lref.line_id2;
          if (lref.line_id3 <> 0) then
            rval.extend;
            rval(rval.last) := -lref.line_id3;
            if (lref.line_id4 <> 0) then
              rval.extend;
              rval(rval.last) := -lref.line_id4;
              if (lref.line_id5 <> 0) then
                rval.extend;
                rval(rval.last) := -lref.line_id5;
                if (lref.line_id6 <> 0) then
                  rval.extend;
                  rval(rval.last) := -lref.line_id6;
                  if (lref.line_id7 <> 0) then
                    rval.extend;
                    rval(rval.last) := -lref.line_id7;
                    if (lref.line_id8 <> 0) then
                      rval.extend;
                      rval(rval.last) := -lref.line_id8;
                      if (lref.line_id9 <> 0) then
                        rval.extend;
                        rval(rval.last) := -lref.line_id9;
                        if (lref.line_id10 <> 0) then
                          rval.extend;
                          rval(rval.last) := -lref.line_id10;
                        end if;
                      end if;
                    end if;
                  end if;
                end if;
              end if;
            end if;
          end if;
        end if;
      end if;
    end loop;
  end if;
  return rval;
end find_islands;
/
show errors

-- Update island edge lists
--
update KADASTER_SET300K_FACE$
  set island_edge_id_list = find_islands(face_id) where face_id <> -1;
commit;

-- Initialize topology metadata (sequences, indexes)
```

```
--
execute sdo_topo.initialize_metadata ('KADASTER_SET300K');

-- Update edge references at perimeter of dataset (only for subsets)
--
update KADASTER_SET300K_EDGE$ e
  set next_left_edge_id = (
    select n.edge_id from KADASTER_SET300K_EDGE$ n
    where n.left_face_id = -1 and n.start_node_id = e.end_node_id)
  where e.left_face_id = -1 and (e.next_left_edge_id is null or
    abs(next_left_edge_id) not in (select edge_id from
KADASTER_SET300K_EDGE$));
commit;
update KADASTER_SET300K_EDGE$ e
  set next_left_edge_id = (
    select -n.edge_id from KADASTER_SET300K_EDGE$ n
    where n.right_face_id = -1 and n.end_node_id = e.end_node_id)
  where e.left_face_id = -1 and (e.next_left_edge_id is null or
    abs(next_left_edge_id) not in (select edge_id from
KADASTER_SET300K_EDGE$));
commit;

update KADASTER_SET300K_EDGE$ e
  set prev_right_edge_id = (
    select -n.edge_id from KADASTER_SET300K_EDGE$ n
    where n.right_face_id = -1 and n.start_node_id = e.end_node_id)
  where e.right_face_id = -1 and (e.prev_right_edge_id is null or
    abs(prev_right_edge_id) not in (select edge_id from
KADASTER_SET300K_EDGE$));
commit;
update KADASTER_SET300K_EDGE$ e
  set prev_right_edge_id = (
    select n.edge_id from KADASTER_SET300K_EDGE$ n
    where n.left_face_id = -1 and n.end_node_id = e.end_node_id)
  where e.right_face_id = -1 and (e.prev_right_edge_id is null or
    abs(prev_right_edge_id) not in (select edge_id from
KADASTER_SET300K_EDGE$));
commit;

update KADASTER_SET300K_EDGE$ e
  set prev_left_edge_id = (
    select -n.edge_id from KADASTER_SET300K_EDGE$ n
    where n.right_face_id = -1 and n.start_node_id = e.start_node_id)
  where e.left_face_id = -1 and (e.prev_left_edge_id is null or
    abs(prev_left_edge_id) not in (select edge_id from
KADASTER_SET300K_EDGE$));
commit;
update KADASTER_SET300K_EDGE$ e
  set prev_left_edge_id = (
    select n.edge_id from KADASTER_SET300K_EDGE$ n
    where n.left_face_id = -1 and n.end_node_id = e.start_node_id)
  where e.left_face_id = -1 and (e.prev_left_edge_id is null or
    abs(prev_left_edge_id) not in (select edge_id from
KADASTER_SET300K_EDGE$));
commit;

update KADASTER_SET300K_EDGE$ e
  set next_right_edge_id = (
    select n.edge_id from KADASTER_SET300K_EDGE$ n
    where n.left_face_id = -1 and n.start_node_id = e.start_node_id)
  where e.right_face_id = -1 and (e.next_right_edge_id is null or
    abs(next_right_edge_id) not in (select edge_id from
KADASTER_SET300K_EDGE$));
commit;
update KADASTER_SET300K_EDGE$ e
  set next_right_edge_id = (
    select -n.edge_id from KADASTER_SET300K_EDGE$ n
    where n.right_face_id = -1 and n.end_node_id = e.start_node_id)
  where e.right_face_id = -1 and (e.next_right_edge_id is null or
    abs(next_right_edge_id) not in (select edge_id from
KADASTER_SET300K_EDGE$));
commit;

-- Clean up temp stuff
--
set timing off
drop table TMP_VERTS_SET300K purge;
drop table TMP_VERTICES_SET300K purge;
drop table TMP_BOUNDARY_SET300K purge;

spool off
exit;
```

Feature creation script

```
--
-- Script to create parcel features and retrieve the geometries
-- from them.
--
spool feat_SET300K.log
set pages 500
set lines 120
set trim on
set trimspool on
set serveroutput on
set echo on

execute sdo_topo.delete_topo_geometry_layer
  ('KADASTER_SET300K','PARCELS_SET300K','FEATURE');

drop table PARCELS_SET300K purge;
drop table PARCELASGEOM_SET300K purge;
delete from user_sdo_geom_metadata where
table_name='PARCELASGEOM_SET300K';
commit;
set timing on

--
-- Create the parcel features
--

create table PARCELS_SET300K
(
  object_id number,
  feature   sdo_topo_geometry
);

execute sdo_topo.add_topo_geometry_layer
  ('KADASTER_SET300K','PARCELS_SET300K','FEATURE','POLYGON');
set timing on

declare
  tli number;
begin
  select tg_layer_id into tli from user_sdo_topo_info
    where topology='KADASTER_SET300K' and table_name='PARCELS_SET300K';
  for a in
    (select face_id from KADASTER_SET300K_FACE$ where face_id <> -1)
  loop
    insert into PARCELS_SET300K values (
      a.face_id,
      sdo_topo_geometry('KADASTER_SET300K',3,tli,
            sdo_topo_object_array(sdo_topo_object(a.face_id,3)))
    );
  end loop;
end;
/
commit;

--
-- Retrieve the parcel geometries from the features (using topology)
--

create table PARCELASGEOM_SET300K
  as select p.object_id, p.feature.get_geometry() as geom
  from PARCELS_SET300K p;

insert into user_sdo_geom_metadata values
('PARCELASGEOM_SET300K','GEOM',
  mdsys.sdo_dim_array(
    mdsys.sdo_dim_element('X',-25000000,325000000,0.002),
    mdsys.sdo_dim_element('Y',275000000,650000000,0.002)), NULL);
commit;
create index INDEX_SET300K on PARCELASGEOM_SET300K (geom)
  indextype is mdsys.spatial_index
  parameters('sdo_fanout=46 tablespace=indx');
call analyze_rtree ('INDEX_SET300K');

spool off
exit;
```

Reports published before in this series:

1. GISt Report No. 1, Oosterom, P.J. van, Research issues in integrated querying of geometric and thematic cadastral information (1), Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 29 p.p.
2. GISt Report No. 2, Stoter, J.E., Considerations for a 3D Cadastre, Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 30.p.
3. GISt Report No. 3, Fendel, E.M. en A.B. Smits (eds.), Java GIS Seminar, Opening GDMC, Delft 15 November 2000, Delft University of Technology, GISt. No. 3, 25 p.p.
4. GISt Report No. 4, Oosterom, P.J.M. van, Research issues in integrated querying of geometric and thematic cadastral information (2), Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 29 p.p.
5. GISt Report No. 5, Oosterom, P.J.M. van, C.W. Quak, J.E. Stoter, T.P.M. Tijssen en M.E. de Vries, Objectgerichtheid TOP10vector: Achtergrond en commentaar op de gebruikersspecificaties en het conceptuele gegevensmodel, Rapport aan Topografische Dienst Nederland, E.M. Fendel (eds.), Delft University of Technology, Delft 2000, 18 p.p.
6. GISt Report No. 6, Quak, C.W., An implementation of a classification algorithm for houses, Rapport aan Concernstaf Kadaster, Delft 2001, 13.p.
7. GISt Report No. 7, Tijssen, T.P.M., C.W. Quak and P.J.M. van Oosterom, Spatial DBMS testing with data from the Cadastre and TNO NITG, Delft 2001, 119 p.
8. GISt Report No. 8, Vries, M.E. de en E. Verbree, Internet GIS met ArcIMS, Delft 2001, 38 p.
9. GISt Report No. 9, Vries, M.E. de, T.P.M. Tijssen, J.E. Stoter, C.W. Quak and P.J.M. van Oosterom, The GML prototype of the new TOP10vector object model, Report for the Topographic Service, Delft 2001, 132 p.
10. GISt Report No. 10, Stoter, J.E., Nauwkeurig bepalen van grondverzet op basis van CAD ontgravingsprofielen en GIS, een haalbaarheidsstudie, Rapport aan de Bouwdienst van Rijkswaterstaat, Delft 2001, 23 p.
11. GISt Report No. 11, Geo DBMS, De basis van GIS-toepassingen, KvAG/AGGN Themamiddag, 14 november 2001, J. Flim (eds.), Delft 2001, 37 p.
12. GISt Report No. 12, Vries, M.E. de, T.P.M. Tijssen, J.E. Stoter, C.W. Quak and P.J.M. van Oosterom, The second GML prototype of the new TOP10vector object model, Report for the Topographic Service, Delft 2002, Part 1, Main text, 63 p. and Part 2, Appendices B and C, 85 p.
13. GISt Report No. 13, Vries, M.E. de, T.P.M. Tijssen en P.J.M. van Oosterom, Comparing the storage of Shell data in Oracle spatial and in Oracle/ArcSDE compressed binary format, Delft 2002, .72 p. (Confidential)
14. GISt Report No. 14, Stoter, J.E., 3D Cadastre, Progress Report, Report to Concernstaf Kadaster, Delft 2002, 16 p.
15. GISt Report No. 15, Zlatanova, S., Research Project on the Usability of Oracle Spatial within the RWS Organisation, Detailed Project Plan (MD-NR. 3215), Report to Meetkundige Dienst – Rijkswaterstaat, Delft 2002, 13 p.
16. GISt Report No. 16, Verbree, E., Driedimensionale Topografische Terreinmodellering op basis van Tetraëder Netwerken: Top10-3D, Report aan Topografische Dienst Nederland, Delft 2002, 15 p.
17. GISt Report No. 17, Zlatanova, S. Augmented Reality Technology, Report to SURFnet bv, Delft 2002, 72 p.
18. GISt Report No. 18, Vries, M.E. de, Ontsluiting van Geo-informatie via netwerken, Plan van aanpak, Delft 2002, 17p.
19. GISt Report No. 19, Tijssen, T.P.M., Testing Informix DBMS with spatial data from the cadastre, Delft 2002, 62 p.
20. GISt Report No. 20, Oosterom, P.J.M. van, Vision for the next decade of GIS technology, A research agenda for the TU Delft the Netherlands, Delft 2003, 55 p.
21. GISt Report No. 21, Zlatanova, S., T.P.M. Tijssen, P.J.M. van Oosterom and C.W. Quak, Research on usability of Oracle Spatial within the RWS organisation, (AGI-GAG-2003-21), Report to Meetkundige Dienst – Rijkswaterstaat, Delft 2003, 74 p.
22. GISt Report No. 22, Verbree, E., Kartografische hoogtevoorstelling TOP10vector, Report aan Topografische Dienst Nederland, Delft 2003, 28 p.
23. GISt Report No. 23, Tijssen, T.P.M., M.E. de Vries and P.J.M. van Oosterom, Comparing the storage of Shell data in Oracle SDO_Geometry version 9i and version 10g Beta 2 (in the context of ArcGIS 8.3), Delft 2003, 20 p. (Confidential)
24. GISt Report No. 24, Stoter, J.E., 3D aspects of property transactions: Comparison of registration of 3D properties in the Netherlands and Denmark, Report on the short-term scientific mission in the CIST – G9 framework at the Department of Development and Planning, Center of 3D geo-information, Aalborg, Denmark, Delft 2003, 22 p.
25. GISt Report No. 25, Verbree, E., Comparison Gridding with ArcGIS 8.2 versus CPS/3, Report to Shell International Exploration and Production B.V., Delft 2004, 14 p. (confidential).
26. GISt Report No. 26, Penninga, F., Oracle 10g Topology, Testing Oracle 10g Topology with cadastral data, Delft 2004, 48 p.
27. GISt Report No. 27, Penninga, F., 3D Topography, Realization of a three dimensional topographic terrain representation in a feature-based integrated TIN/TEN model, Delft 2004, 27 p.
28. GISt Report No. 28, Penninga, F., Kartografische hoogtevoorstelling binnen TOP10NL, Inventarisatie mogelijkheden op basis van TOP10NL uitgebreid met een Digitaal Hoogtemodel, Delft 2004, 29 p.

29. GISt Report No. 29, Verbree, E. en S.Zlatanova, 3D-Modeling with respect to boundary representations within geo-DBMS, Delft 2004, 30 p.
30. GISt Report No. 30, Penninga, F., Introductie van de 3e dimensie in de TOP10NL; Voorstel voor een onderzoekstraject naar het stapsgewijs introduceren van 3D data in de TOP10NL, Delft 2005, 25 p.
31. GISt Report No. 31, P. van Asperen, M. Grothe, S. Zlatanova, M. de Vries, T. Tijssen, P. van Oosterom and A. Kabamba, Specificatie datamodel Beheerkaart Nat, RWS-AGI report/GIST Report, Delft, 2005, 130 p.
32. GISt Report No. 32, E.M. Fendel, Looking back at Gi4DM, Delft 2005, 22 p.

ZUIDPOLDER
VAN DELFGAUW
MOLENSLOOT
PARALLELWEG
AKKERDIJKSCHE
POLDER
Pocket PC