Research on usability of Oracle Spatial within the RWS organisation

AGI-GAG-2003-21

Dr. dipl.-eng. S. Zlatanova, drs. T.P.M. Tijssen,
prof.dr.ir. P.J.M. van Oosterom and drs. C.W. Quak

GISt No. 21 Delft, April 2003

Summary:
The goal of this project is to investigate the functionality offered by Oracle Spatial 9i with respect to the needs of RWS (http://www.verkeerenwaterstaat.nl) towards an integrated maintenance and querying of objects (GIS-Nat) en their characteristics (Zlatanova 2002). This document reports the tests performed with the DTB-Nat, Beheerkaart-Nat and Regiokaart-Nat as they are organized in the currently available Oracle Spatial geometry types. The aim of the research is to propose an appropriate model for the maintenance of these objects that have different resolution and different description (corresponding to the needs of different applications).

This report has been prepared under the authority of Rijkswaterstaat – Meetkundige Dienst.

ISSN: 1569-0245

ISBN: 90-77029-07-9

AGI-GAG-2003-21

Rijkswaterstaat – Meetkundige Dienst
P.O. Box 5023, 2600 GA Delft, the Netherlands
Tel.: +31 (0)15-269 1111; Fax +31 (0)15-261 8962;
E-mail: md-info@mdi.rws.minvenw.nl

CONTENTS

1 INTRODUCTION

The goal of this project is to investigate the functionality offered by Oracle Spatial 9i with respect to the needs of RWS (http://www.verkeerenwaterstaat.nl) towards an integrated maintenance and querying of objects (GIS-Nat) and their characteristics (Zlatanova, 2002). This document reports the tests performed with the DTB-Nat, Beheerkaart-Nat and Regiokaart-Nat as they are organised in the currently available Oracle Spatial geometry types. The aim of the research is to propose an appropriate model for the maintenance of these objects that have different resolution and different description (corresponding to the needs of different applications).

Since ArcGIS is largely in use within the RWS, it is utilised as a basic visualisation tool to display results of queries. GeoGraphics (an extension of MicroStation) and in-house Java-based software that creates VRML files (to be visualised in VR browsers) are used to demonstrate the possibilities to access the Oracle Spatial objects from different front-ends.

Moreover, the report discusses the following issues considering the knowledge of section GISt (http://www.geo.tudelft.nl/gist/) and GDMC (http://www.gdmc.nl):

- Functionality offered by Oracle Spatial regarding data storage, conversion, querying, analysis and topology.
- Possibilities of Oracle Spatial for use in a distributed database environment.
- Built-in functionality of Oracle Spatial regarding 2½ en 3D data storage and processing.
- Compatibility / applicability of Oracle Spatial with new standards like GML and other OpenGIS standards.
- Possibilities and/or complications for the use of Oracle Spatial in combination with ESRI products ArcGIS and ArcSDE.
- Advantages of storage of spatial data in Oracle Spatial instead of in other databases like SQL server or Oracle in combination with ArcSDE (on the basis of already available knowledge and experience within TUD, and without performing tests).
- Utilisation of Oracle Spatial without ArcSDE as a database for ArcGIS or Microstation GeoGraphics without loss of functionality or threatening the database integrity.

Additional details on the testing:

- The tests are only related to the functionality of Oracle Spatial, i.e. only spatial data stored in SDO_GEOMETRY is considered.
- The tests are limited to a pure geo-database architecture, i.e. middleware architecture (e.g. using ArcSDE) is not tested.
- The data is accessed directly from different frond-ends (ArcGIS, MicroStation, Java programs).
- The tests are executed in a single-user environment.
- The elapsed times (when given) are measured at the client side, thus statistical information on the performance of different components (network, CPU, disk I/O) is not gathered.
- Tests are executed using PL/SQL scripts at a database level. Some of the scripts are organised as shell scripts.
- Spatial data that cannot be maintained by Oracle Spatial is either edited or removed from the data set.

The system on which Oracle Spatial is running is a Sun Enterprise 3500 with the following characteristics: 2 UltraSPARC CPUs of 400 MHz, 2 Gb main memory, 8 internal hard disks of 18 Gb (10000 rpm, fibre channel attached, 2 controllers), 24 external hard disks of 18 Gb in 2 Sun StorEDGE A1000 boxes offering hardware RAID support (10000 rpm, SCSI channel attached, 2 controllers), Solaris 7 operating system (64 bit). The tests are performed from a PC (Pentium II 400 Mhz, 364 Mb main memory). The Oracle server version used for the project is "Oracle 9i Enterprise Edition Release 9.2.0.2.0 (32 bit)".

The report is organised in the following seven general chapters:

- Study of the existing systems, formats and types of objects. The study was carried out in close cooperation with specialists from RWS.
- Short overview on Oracle Spatial 9i geometry model.
- Performing tests with existing models (i.e. DTB-Nat, Regiokaart-Nat, Beheerkaart-Nat).
- Discussion on the proposed integrated model (GIS-Nat) for maintenance of beheerobjects and DTB-Nat objects. The integrated model is designed taking into account the discussions with the users and the tests performed on the data sets. The model concentrates on a subset of the objects of interest for RWS that can be found within the provided test maps.
- Performing tests on the integrated model.
- Access, visualisation and editing of objects in different front-ends.
- Conclusions and recommendations.

2 STUDY OF EXISTING SYSTEMS, MEETINGS WITH PROJECT LEADERS

A study based on existing documentation and a series of meetings with project leaders of RWS and users of the data sets contributed to clarifying the objects and types of analysis (Asperen and Bannier 2002, Boeringa 2002, Brink et al 2002, Klatter and Padding 1998, Looman and Wouters 2002, Meijer et al 2001, Wybenga and Koster 2002, Zwijnenburg 2001). The following text summarises the findings.

2.1 Types of data used in the NAT sector

The DTB-Nat maintains the objects of interest along the rivers at scale 1:1000 (or 1:5000). The objects of this map are created from aerial photographs in the software *InfoCam* and are supposed to consist of closed non-overlapping polygons. The DTB-Nat objects are defined with respect to their physical appearance in reality (or in the aerial photographs). The rivers are subdivided into sections related to the stereo-models. *InfoCam* stores all the information in Oracle 7 in a topological structure, which is not directly available. From this model the objects can be exported to another system using ArcView shape files, Arc/Info coverages, DXF, etc. (currently shape files are used mostly).

The DTB-Nat is produced for and used by the Dienstkringen. However, update of information is not performed by the Dienstkringen. In case of changes, a new shape file is obtained from the DTB-Nat after photogrammetric revision. DTB-Nat is used mostly for visualisation and GIS-applications for many purposes. In this respect, the analysis performed can be classified as a semantic analysis, e.g. visualise all the objects with a given code (to check what kind of maintenance is required for the objects of interest). Hardly any spatial analysis is done. Some Dienstkringen (e.g. DK ZH) do not have Regiokaart-Nat, but only Beheerkaart-Nat.

The Beheerkaart-Nat contains the objects as they are defined and maintained by the RWS Dienstkringen. The object codes correspond to the classification in TISBO. The scale of this map is the same as the DTB, i.e. 1:1000 (1:5000 for regions where 1:1000 does not exist). The Beheerkaart-Nat is created in ArcView and Arc/Info, the boundaries of the objects are closed polygons (stored as shape files), which also coincide with the borders of the DTB-Nat objects. The objects of the Beheerkaart-Nat do not correspond to the objects of DTB-Nat due to three general reasons:

- Most of the Beheerkaart-Nat objects are compositions of DTB-Nat objects.
- The Beheerkaart-Nat contains objects that are not part of the DTB-Nat, e.g. riverbeds.
- Some objects that are not maintained by RWS are not included in the Beheerkaart-Nat and therefore some gaps (compared to the DTB-Nat) may appear.

The Regiokaart-Nat contains the GIS-Nat objects that are of interest for the RWS Regionale Directies. It is compiled from handmade drawings and digitised at a scale of 1:50000. This map is maintained in two parts: The "Beperkte Bopper Kaart (BBK)" and the "Uitgebreide Bopper Kaart (UBK)". The BBK contains water systems and water parts. The UBK contains the Beheerkaart-Nat objects (Note: UBK is the previous name of the Regiokaart-Nat, so UBK and Regiokaart-Nat are actually the same). The objects are closed polygons stored as ArcView shape files. The objects from the Regiokaart-Nat do not correspond to the Beheerkaart-Nat objects because of:

- Different boundaries (due to the generalisation, they fit on the TOP50raster).
- Some of the polygon objects in the Beheerkaart-Nat may be represented as lines in the Regiokaart-Nat.
- Some of the objects from the Beheerkaart-Nat could be missing from the Regiokaart-Nat (being a part of larger objects).

The following differences between Beheerkaart-Nat and Regiokaart-Nat are related to the data sets used in the tests:

- The first three symbols of the Regiokaart-Nat and Beheerkaart-Nat are different, i.e. 112B has to be 108C with respect to the new coding system. This is taken into account in the database. The other new codes are 117, 113 and NZ5C, but they do not exist in the area covered by the Beheerkaart-Nat. (the pilot area of the Beheerkaart-Nat is smaller then the coverage of the Regiokaart-Nat).
- There is no 1:1 correspondence between Regiokaart-Nat and Beheerkaart-Nat. This is further investigated using Oracle Spatial functions.
- The lines from the Regiokaart-Nat that do not have codes are to be deleted. They are used only as borders for existing polygons. They are still in the shape files because the databases are not cleaned recently.
- The columns Haringvlietoud, WSD (WaterSyteemDelen), WS (WaterSysteem), and HWS HoofdWatersysteemDelen etc, can be deleted since this information is incorporated in the codes. Only the Dienstkring is not included in the code, because the Beheerkaart is currently produced and maintained for/by the Dienstkring.
  HWS, WS and WSD:
  WSD: always within Dienstkring;
  WS: always within Regionale Directie;
  HWS: multiple RD's possible.
- The IDs in the Regiokaart-Nat (lines and polygons) are automatically generated by Arc/Info and can be deleted.
- There is a correspondence between the objects from the Beheerkaart-Nat and DTB-Nat that is further investigated with Oracle Spatial functions.

The definition of Regiokaart-Nat has been set in 1999. Since then only two RD's (Zuid-Holland and Noord-Brabant) ordered the Regiokaart-Nat. The definition of the Beheerkaart-Nat has been set in 2003. In 2002 several pilot data sets have been compiled and they have been used in this study. As a result, both products are at this moment hardly used, because of non-availability. This also relates to problems with the envisaged GIS-applications: they are hardly used or even not developed at this time. Many problems encountered in this research relate to the actual non-availability of fully developed and widely accepted products (only DTB-Nat is fully developed and widely accepted).

2.2 Description of data sets

Three data sets presenting objects from the DTB-Nat, Beheerkaart-Nat and Regiokaart-Nat were used for the tests.

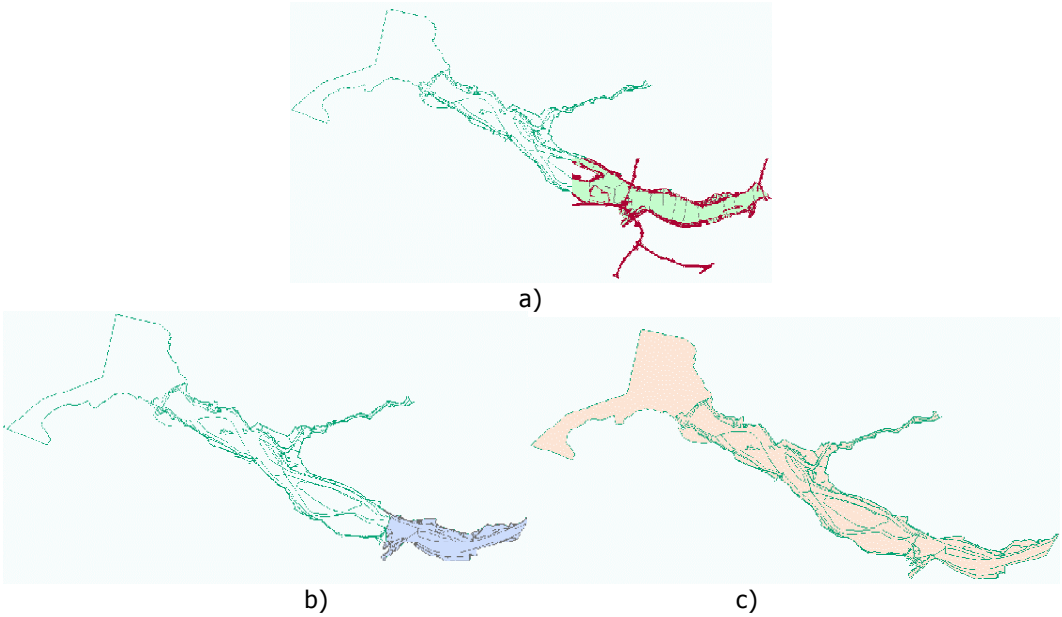


Figure 1: Data sets: a) DTB-Nat, b) Beheerkaart-Nat, c) Regiokaart-Nat.

The following text presents the contents of the shape files (columns and description), the columns as they are defined in the FME mapping files (see Section 4.1 and Appendix 1). In ArcGIS an ID (FID) is generated for all objects. This pseudo column is maintained internally, it is not recognised as a separate column by FME.

DTB_NAT of Hollandsch Diep and Volkeraksluizen, scale 1:1000, points, lines and polygons are given in three separate 3D shape files.

```
 SHAPE_DEF dtbnat_sym                    \ definition of DTB_Nat SYMBOLS
                                         \
    SHAPE_GEOMETRY       shape_pointz    \ 3D point
    CTE                  char(20)        \ Classificatie Topografishe Elementen
    DATUM                date            \ date
    DTM                  char(50)        \ J/N J-can be used to create TIN
    EIGNAM               char(50)        \ name of owner
    GRADENHOEK           number(16,5)    \ orientation of the symbol
    HECTO                char(50)        \ hectometre
    LAYER                char(50)        \ 1-4 for multilevel objects
    OMSCHR               char(50)        \ class (meaning)
    THEMA                char(50)        \ class
    THEMB                char(50)        \ text visualised near the object

SHAPE_DEF dtbnat_lin                    \ definition of DTB_Nat LINES
                                        \
    SHAPE_GEOMETRY       shape_polylinez \ 3D polyline
    CTE                  char(20)        \ see shape sym
    DATUM                date            \ see shape sym
    DTM                  char(50)        \ see shape sym
    LAYER                char(50)        \ see shape sym
    OMSCHR               char(50)        \ see shape sym
    PRJL                 char(50)        \
    THEMA                char(50)        \ see shape sym
    THEMB                char(50)        \ see shape sym

SHAPE_DEF dtbnat_reg                    \ definition of DTB_Nat POLYGONS
                                        \
    SHAPE_GEOMETRY       shape_polygonz  \ 3D polygon
    CTE                  char(20)        \ see shape sym
    DATUM                date            \ see shape sym
    DTM                  char(50)        \ see shape sym
    EIGNAM               char(50)        \ see shape sym
    LAYER                char(50)        \ see shape sym
    OMSCHR               char(50)        \ see shape sym
    THEMA                char(50)        \ see shape sym
    THEMB                char(50)        \ see shape sym
```

REGIOKAART-Nat of Haringvliet, Hollandsch Diep and Volkeraksluizen, scale 1:50 000, 2 separate 2D shape files with lines and polygons:

```
SHAPE_DEF haringvl_l2                   \ REGIOKAART-Nat LINES (after changes, see below)
                                        \
    SHAPE_GEOMETRY       shape_arc      \ 2D polyline
    LENGTH               number(16,3)   \ length
    FNODE                number(11,0)   \ info from Arc/Info coverage
    TNODE                number(11,0)   \ info from Arc/Info coverage
    LPOLY                number(11,0)   \ left (ID left polygon)
    RPOLY                number(11,0)   \ right (ID right polygon)
    HARINGOUD1           number(11,0)   \ sequential code (ID polygon)
    HARINGOUD2           number(12,0)   \ other code
    OBJECTCODE           char(20)       \ TISBO code 112B (108C in Beheerkaart-Nat)
    WSD                  char(40)       \ waterdsysteemdelen
    WS                   char(40)       \ watersystemen
    DKR                  char(40)       \ Dienstkringen
    HWS                  char(40)       \ HoofdWaterSysteemDelen
    DIR                  char(40)       \ Directie
    OBJCODE              char(20)       \ ObjectCode
```

The shape file structure was changed due to duplicated column names (FNODE, TNODE, LPOLY, RPOLY and HARINGOUD1) and content (HARINGOUD1, HARINGVL_L and HARINGVL_1). We looked at the data set using ArcGIS (ArcMAP) and the first impression was that all the names were unique. However, at a later stage it was detected that the shape file couldn't be imported in Oracle Spatial due to duplicate column names. We opened the same shape file with ArcView and then we realised that, indeed, the names of many columns are the same. Apparently ArcView does not have a problem with non-unique names (perhaps maintains internal numeration of the fields). ArcGIS internally resolves names and ensures that they are unique. We decided to delete duplicated

columns that have either "0 value everywhere" (i.e. columns FNODE, TNODE, LPOLY and RPOLY) or "duplicated content" (i.e. HARINGVL_L and HARINGVL_1 were exactly the same as the first HARINGOUD1), and rename the second column HARINGOUD1 to HARINGOUD2. The information in the columns FNODE, TNODE, LPOLY and RPOLY is related to the polygons in the polygon file (and comes from ArcInfo coverages), but no one could ensure that this information is still correct. Therefore these columns were also removed from the tables. Thus the original 21 columns were reduced to 15.

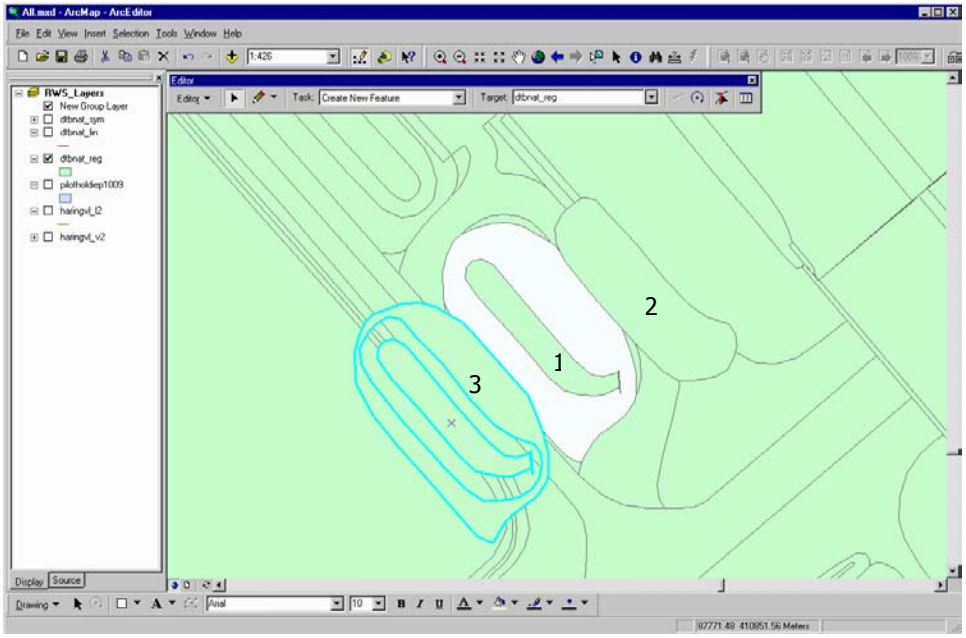
```
SHAPE_DEF haringvl_v2                    \ REGIOKAART-Nat POLYGONS (after changes)
                                         \
    SHAPE_GEOMETRY       shape_polygon   \ 2D polygon
    AREA                 number(16,3)    \ area
    PERIMETER            number(16,3)    \ polygon
    SUBCLASS             char(13)        \ ????? (TEMP)
    RINGS_OK             number(7,0)     \ 1 to 5
    RINGS_NOK            number(7,0)     \ 0
    HARINGOUD1           number(11,0)    \ sequential code (ID polygon)
    OBJECTCODE           char(20)        \ see shape lines
    WSD                  char(40)        \ see shape lines
    WS                   char(40)        \ see shape lines
    DKR                  char(40)        \ Dienstkring
    HWS                  char(40)        \ see shape lines
    DIR                  char(40)        \ see shape lines
    OBJCODE              char(20)        \ see shape lines
```

The shape file with the polygons had a similar problem as the lines. The initial number of columns was 19. The column's name SUBCLASS was two times in the list with different content (the second SUBCLASS basically the same as HARINGVL_1). HARINGVL_V, HARINGOUT1 and POLY had the same content, i.e. ID of polygons including the outer polygon with ID=1. Since the outer polygon (maintained in ArcInfo) is not converted in the shape file the polygon's IDs start with 2. HARINGVL_1 was identical with the second column HARINGOUD1. Columns HARINGVL_V, HARINGVL_1, POLY and one of the SUBCLASS and HARINGOUD1 were deleted from the table. These changes were not performed in ArcView but in Oracle Spatial, i.e. columns POLY_, SUBCLASS_ and HARINGVL_V were dropped from the table HARING_REG (see table names in Oracle).

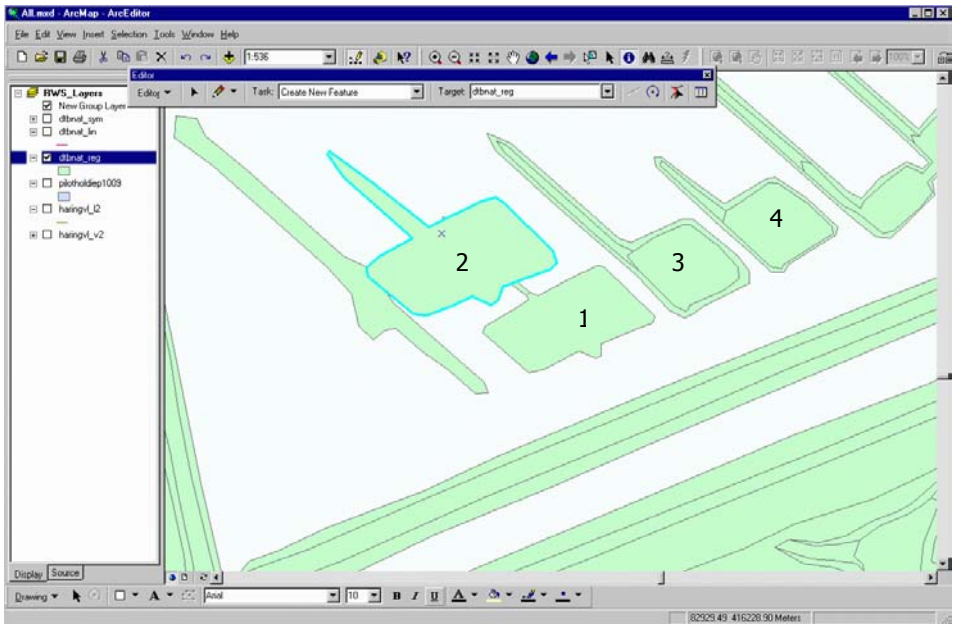
BEHEERKAART-Nat of Hollandsch Diep and Volkeraksluizen, scale 1:50 000, a 2D shape file with polygons. The Beheerkaart-Nat does not contain lines and polygons:

```
SHAPE_DEF pilotholdiep1009               \ RBEHEERKAART-Nat POLYGONS
                                         \
    SHAPE_GEOMETRY       shape_polygon   \ 2D polygon
    AREA                 number(16,3)    \ area
    PERIMETER            number(16,3)    \ perimeter
    HOLDIEP6_            number(11,0)    \ ID polygon +1
    HOLDIEP6_I           number(11,0)    \ ID polygon
    OBJECTCODE           char(20)        \ code BOPPER/TISBO
```

Note: These tables and columns were imported into Oracle Spatial. The description and content of the tables was further modified at database level after discussion with RWS specialists. Since the validation and self-overlapping tests revealed a large number of errors in representing polygons with holes, a new set of DTB-Nat shape files was obtained from RWS. The reported results in this research refer to the second data set. Figure 2, a) shows three objects that overlap. Two objects (3 and 2) are shifted left and right from their original positions to show the overlap. Object 1 is on its original position, object 2 has to be around it (donut) and object 3 around object 2. Figure 2, b) is another example. The large object around all small shapes is deleted for clarity and object 2 is shifted up-left from its position. It is visible that below it another smaller object appears. Actually, object 2 has to be a tiny "donut" object around object 1. Objects 3 and 4 have the same problem (not shifted from their original positions).



a)



b)

Figure 2: Problems with holes and donuts in the first data set of DTB-Nat.

3 ORACLE SPATIAL GEOMETRY MODEL

Oracle Spatial provides a SQL schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial features in an Oracle 9i database (Oracle 9i Spatial, 2003). Oracle Spatial consists of the following components: a schema (MDSYS) that prescribes the storage, syntax, and semantics of supported geometric data types; a spatial indexing mechanism; a set of operators and functions for performing area-of-interest queries, spatial join queries, and other spatial analysis operations and administrative utilities.

Object-relational model: Oracle Spatial supports an object-relational model for representing geometries. In this model, the geometric description of a spatial object is stored in a single row, in a single column of object type SDO_GEOMETRY in a user-defined table. The object-relational model corresponds to a "SQL with Geometry Types" implementation of spatial feature tables in the OpenGIS ODBC/SQL specification for geo-spatial features. Geometry types are used to represent a spatial object.

Geometry types: Geometry for line and area features is an ordered sequence of vertices that are connected by straight-line segments or circular arcs. The semantics of the geometry is determined by its type. Oracle Spatial supports several primitive types and geometries composed of collections of these types: points and point clusters, line strings, n-point polygons, arc line strings (all arcs are generated as circular arcs), arc polygons, compound polygons, compound line strings, circles, optimized rectangles (see Figure 3).

Oracle Spatial maintains different dimensions (2D, 3D and 4D, 4D is used for 3D spatial data with measures – linear references - attached). For example, three-dimensional points are elements composed of three coordinates, X, Y and Z. Line strings are composed of one or more pairs of points that define line segments. Polygons are composed of connected line strings that form a closed ring and the area of the polygon is implied. Self-crossing polygons are not supported, self-crossing line strings are allowed. If a line string crosses itself, it does not become a polygon. A self-crossing line string does not have an implied area.

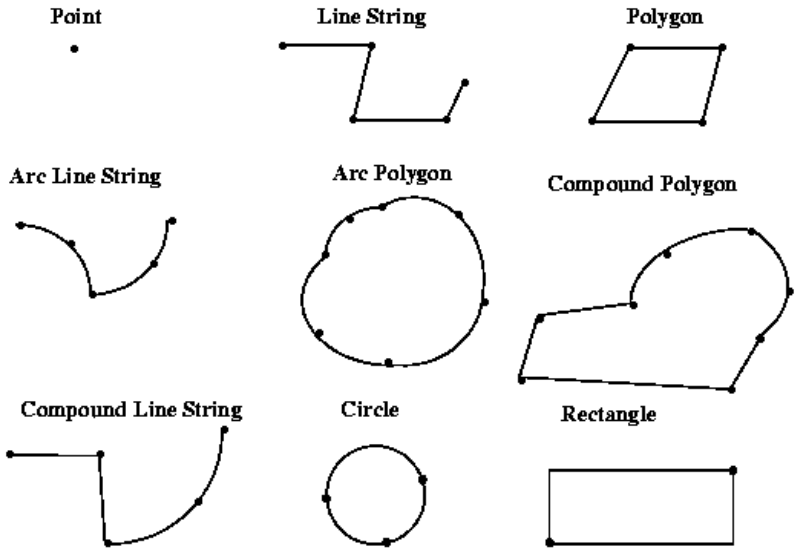
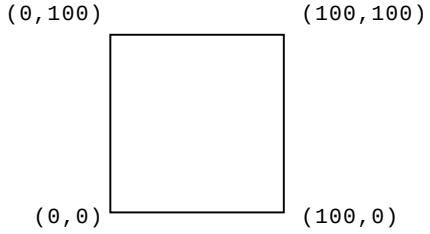


Figure 3: Geometry types supported by Oracle Spatial (Oracle 9i Spatial, 2003).

Oracle Spatial defines the object type SDO_GEOMETRY as:

```
Create type SDO_GEOMETRY as object (
    SDO_GTYPE          NUMBER,
    SDO_SRID           NUMBER,
    SDO_POINT          SDO_POINT_TYPE,
    SDO_ELEM_INFO      SDO_ELEM_INFO_ARRAY,
    SDO_ORDINATES      SDO_ORDINATE_ARRAY);
```

Simple polygon: Considering the model shown above, a simple polygon (the box below) can be represented as follows:




```
SDO_GEOMETRY Column = (
 SDO_GTYPE = 2003
 SDO_SRID = NULL
 SDO_POINT = NULL
 SDO_ELEM_INFO= (1,1003,1)
 SDO_ORDINATES =(0,0, 100,0, 100,100, 0,100, 0,0))
```

In SDO_GTYPE=2003, the first position indicates the dimension (2D in this case), the last position indicates the element type (3 indicates a polygon). In SDO_ELEM_INFO, the final 1 in '1,1003,1' indicates that this is a polygon, 1003 means an exterior polygon ring. The first position ('1') indicates that the first (and only) element starts at offset 1 in the coordinate list.

SDO_SRID can be used to identify a coordinate system (spatial reference system) to be associated with the geometry. If SDO_SRID is null, no coordinate system is associated with the geometry. If SDO_SRID is not null, it must contain a value from the SRID column of the MDSYS.CS_SRS table and this value must be inserted into the SRID column of the USER_SDO_GEOM_METADATA view.

The next SQL statements illustrate how this box is stored as a geometry type in Oracle (sdo_geometry type) in the 'GEOM2D' table. The same box with height 50 is stored in the 'GEOM3D' table. The coordinates have to be listed in an anti-clockwise direction looking from above or from outside (if it is a closed 3D object). This requirement is derived from standards in rendering (and 3D computer graphics), i.e. polygons are rendered (visualised on the screen) if their normal vector points toward the user. The first coordinate is always repeated as a last coordinate in the SDO_ORDINATES list to indicate that the polygon is closed

```
/* creation of the tables */
create table geom2d (shape mdsys.sdo_geometry not null, ID number(11) not null);
create table geom3d (shape mdsys.sdo_geometry not null, ID number(11) not null);

/* inserting the data */
/* a 2D box */
insert into geom2d (shape,ID) values (
   mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
   mdsys.SDO_ORDINATE_ARRAY(0,0, 100,0, 100,100, 0,100, 0,0)), 8);

/* a 3D box */
insert into geom3d (shape, ID) values (
   mdsys.SDO_GEOMETRY(3003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
   mdsys.SDO_ORDINATE_ARRAY(0,0,50, 100,0,50, 100,100,50, 0,100,50, 0,0,50)), 9);
```

Polygon with a hole: Figure 4 illustrates a polygon consisting of two elements: an exterior polygon ring and an interior polygon ring. The inner element in this example is treated as a void (a hole). The order of the coordinates in the inner ring has to be oriented clockwise. In the SDO_GEOMETRY definition the geometry is:

SDO_GTYPE = 2003. The 2 indicates two-dimensional, and the 3 indicates a polygon.
SDO_SRID = NULL.
SDO_POINT = NULL.
SDO_ELEM_INFO = (1,1003,1, 19,2003,1). There are two triplet elements: 1,1003,1 and 19,2003,1. 1003 indicates that the element is an exterior polygon ring; 2003 indicates that the element is an interior polygon ring. 19 indicates that the second element (the interior polygon ring) ordinate specification starts at the 19th number in the SDO_ORDINATES array (that is, 7, meaning that the first point is 7,5).
SDO_ORDINATES = (2,4, 4,3, 10,3, 13,5, 13,9, 11,13, 5,13, 2,11, 2,4, 7,5, 7,10, 10,10, 10,5, 7,5).

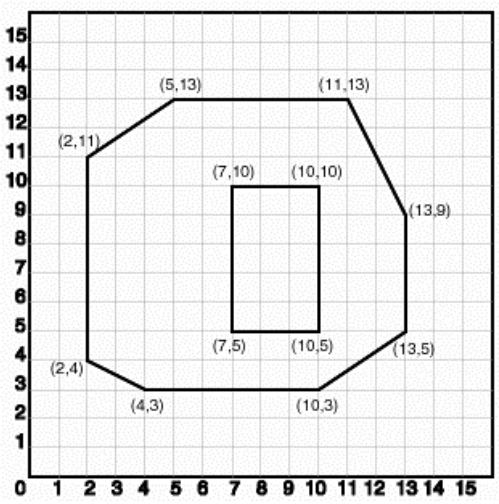


Figure 4: Polygon with a hole (Oracle 9i Spatial, 2003).

This polygon is inserted in the GEOM2D table created above, using the following command:

```
INSERT INTO GEOM2D (shape, ID) VALUES(
    MDSYS.SDO_GEOMETRY(2003, NULL, NULL,
    MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1, 19,2003,1),
    MDSYS.SDO_ORDINATE_ARRAY(2,4, 4,3, 10,3, 13,5, 13,9, 11,13, 5,13, 2,11, 2,4,
        7,5, 7,10, 10,10, 10,5, 7,5)), 10);
```

Multi geometries, collections, lines, points: A multipolygon (sdo_gtype = 2007 or 3007) is a polygon with more than one exterior boundary (e.g. to model an area feature like the province of Friesland with its islands as a single object). In a geometry collection (sdo_gtype = 2004 or 3004) any combination of simple and multi-versions of point, line and polygon geometries can be stored.

Representation of (multi)line and multipoint is similar to polygons in the sense that gtype, element info and ordinate arrays are used to define the geometry (with relevant values for sdo_gtype and sdo_elem_info of course). For (single) points a choice exists between storage like lines and polygons (using element info and ordinate arrays) and "optimized" storage using the SDO_POINT element of the SDO_GEOMETRY type. Simple polygons, polygons with holes, lines and points are geometry types that are used to model the objects from the Beheerkaart-Nat, Regiokaart-Nat and DTB-Nat. While processing geometries (overlapping, matching) sometimes multipolygons were created.

Spatial metadata: Data about the geo-data in the database must be provided in a metadata table. Information from the metadata table is used by Oracle Spatial for validation and index creation, and by clients. For example, the creation of spatial indexes is impossible without registering the geometry column in the user metadata table. The metadata describes the dimensions, lower and upper bounds, and tolerance in each dimension. The USER_SDO_GEOM_METADATA view contains metadata information for all spatial tables owned by the user (schema). The users are responsible for populating this view.

The user metadata view has the following definition:

```
TABLE_NAME    VARCHAR2(32)
COLUMN_NAME   VARCHAR2(32)
DIMINFO       SDO_DIM_ARRAY
SRID          NUMBER
```

DIMINFO contains for dimension the name (e.g. 'X'. 'Y', 'Z'), the extent of the geometry column (minimum and maximum coordinates) and a tolerance (geometric accuracy).

Spatial indexing: Once data has been loaded into the spatial tables through either bulk or transactional loading, a spatial index must be created on the tables for efficient access to the data. Each spatial index can be an R-tree index or a Quad-tree (hierarchical grid) index. Each index type is appropriate in different situations. One even can maintain both an R-tree and a Quad-tree index on the same geometry column. Some of the properties of R-tree indexing are: index creation and tuning is easy, relatively less storage space is required, the indexing can be up to 4 dimensions.

However the index cannot be fine-tuned. In case of heavy update of the geometry column, the R-tree is not a good choice. Usage of Quad-tree allows fine-tuning of index geometries and updates do not affect the performance of the Quad-tree. However, the performance of some operators is worse (e.g. nearest neighbour), storage space is larger and tuning is more complex. To select the appropriate indexing, analysis of the data and the operations on them has to be made in advance.

If a spatial index is created without specifying any Quad-tree-specific parameters, an R-tree index is created. For example, the following statement creates a spatial R-tree index named territory_idx (on column territory_geom in table territories) using default values for parameters that apply to R-tree indexes:

```
CREATE INDEX territory_idx ON territories (territory_geom)
    INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

The default number of dimensions for an R-tree index is the number of dimensions specified in the USER_SDO_GEOM_METADATA view. But it is possible to create a 2D R-tree index on 3D data. However, if a spatial index has been built on more than two dimensions of a layer, the only spatial operator that can be used against that layer is SDO_FILTER (the primary filter or index-only query), which considers all dimensions. The other operators (SDO_RELATE, SDO_NN, SDO_WITHIN_DISTANCE) are disabled if the index has been built on more than two dimensions. The indexes used in our tests are two-dimensional R-trees.

Validation of geometry in Oracle Spatial: An Oracle Spatial function, VALIDATE_GEOMETRY, exists that checks the stored geometries against the following list of constraints (Oracle 9i Spatial, 2003):

- Polygons have at least four points, which includes the point that closes the polygon. (The last point is the same as the first.)
- Polygons are not self-crossing.
- No two vertices on a line or polygon are the same.
- Polygons are oriented correctly. (Exterior ring boundaries must be oriented counter clockwise, and interior ring boundaries must be oriented clockwise.)
- An interior polygon ring touches the exterior polygon ring at no more than one point.
- If two or more interior polygon rings are in an exterior polygon ring, the interior polygon rings touch at no more than one point.
- Line strings have at least two points.
- 1-digit and 4-digit SDO_ETYPE values are not mixed (that is, both used) in defining polygon ring elements.
- Points on an arc are not collinear (that is, are not on a straight line) and are not the same point.
- Geometries are within the specified bounds of the applicable DIMINFO column value (from the USER_SDO_GEOM_METADATA view).
- Linear Referencing Systems (a means to associate attributes or events to locations or portions of a linear feature, used for highways, railroads, transit routes) geometries have 3 or 4 dimensions and a valid measure dimension position (3 or 4, depending on the number of dimensions).
- Geometries are within the extent of the coordinate system.

Detecting relationships between objects: Although no topological data structures can be maintained by the current version of Oracle Spatial, topological relationships can be detected. The SDO_RELATE operator implements a 9-intersection model for categorizing binary topological relations between points, lines, and polygons. According to the 9i-framework, each spatial object has an interior, a boundary, and an exterior. The boundary consists of points or lines that separate the interior from the exterior. The boundary of a line consists of its end points. The boundary of a polygon is the line that describes its perimeter. The interior consists of points that are in the object

but not on its boundary, and the exterior consists of those points that are not in the object. Given that an object A has 3 components (a boundary Ab, an interior Ai, and an exterior Ae), any pair of objects has 9 possible interactions between their components. Pairs of components have an empty (0) or a non-empty (1) set intersection. The set of interactions between 2 geometries is represented by a 9-intersection matrix that specifies which pairs of components intersect and which do not. Figure 5 shows the 9-intersection matrix for 2 polygons that are adjacent to one another.

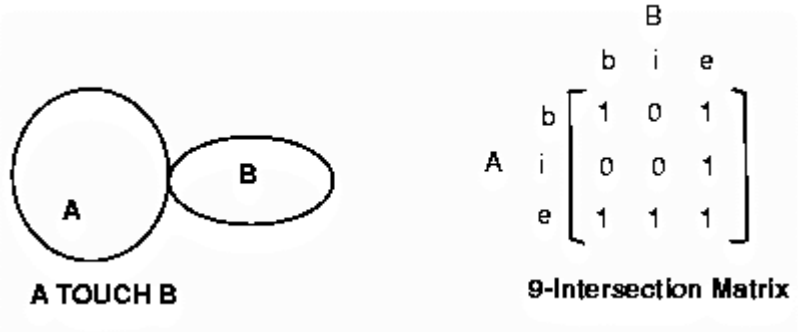


Figure 5: Touching objects and the 9-intersection matrix (Oracle 9i Spatial, 2003).

This matrix yields the following bit mask, generated in row-major form: "101001111". Some of the topological relationships identified are given the names used by Max Egenhofer (Figure 6). Oracle Spatial uses the following names:

- DISJOINT -- The boundaries and interiors do not intersect.
- TOUCH -- The boundaries intersect but the interiors do not intersect.
- OVERLAPBDYDISJOINT -- The interior of one object intersects the boundary and interior of the other object, but the two boundaries do not intersect. This relationship occurs, for example, when a line originates outside a polygon and ends inside that polygon.
- OVERLAPBDYINTERSECT -- The boundaries and interiors of the two objects intersect.
- EQUAL -- The two objects have the same boundary and interior.
- CONTAINS -- The interior and boundary of one object is completely contained in the interior of the other object.
- COVERS -- The interior of one object is completely contained in the interior of the other object and their boundaries intersect.
- INSIDE -- The opposite of CONTAINS. A INSIDE B implies B CONTAINS A.
- COVEREDBY -- The opposite of COVERS. A COVEREDBY B implies B COVERS A.
- ON -- The interior and boundary of one object is on the boundary of the other object (and the second object covers the first object). This relationship occurs, for example, when a line is on the boundary of a polygon.
- ANYINTERACT -- The objects are non-disjoint.

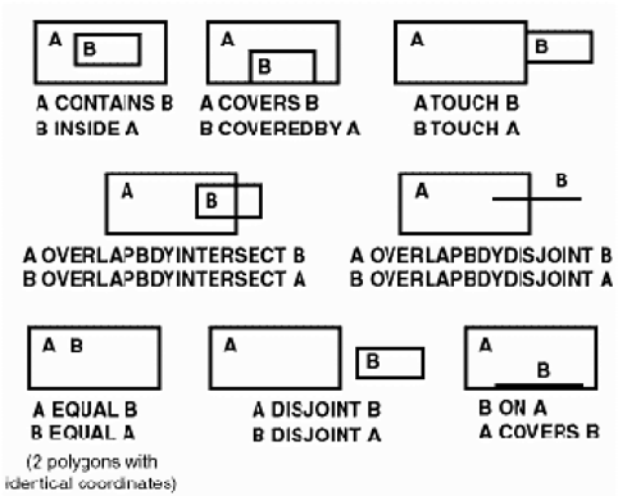


Figure 6: Spatial relationships supported by Oracle Spatial (Oracle 9i Spatial, 2003).

4 TESTS WITH EXISTING DATA

This section presents operations related to the loading of the objects in an Oracle database, validation of objects and consistency checks.

4.1 Loading shape files in Oracle Spatial 9i

The shape files were loaded in Oracle Spatial using the Feature Manipulation Engine (FME) (http://www.safe.com/, FME Oracle Suite 2003, 14 days evaluation trial). The operation was done in two steps: first, mapping files are prepared and second, the data is loaded according to the translation schema. For each shape file a separate mapping file is created. FME reads the shape file and translates the ESRI schema into an Oracle (object-relational) schema. The mapping file for the Beheerkaart-Nat is given in Appendix 1. The files are automatically generated by FME but we changed some of the parameters as follows:

- LOG_FILENAME holdarea2ora.log (a specific log file name is given for each mapping file).
- ORACLE_Dimension 3 (the dimension is changed from 2 to 3 for all the 6 maps). The idea was to use the same dimension for all the maps since the operators may have problems with different dimensions.
- ORACLE_GeometryColumn SHAPE.
- All the FLOAT data types were replaced with the more general Oracle NUMBER data type.
- The names of the Oracle tables are changed to short, easy to use, names.

The names of Oracle tables are:

DTBNAT_SYM	DTB-Nat symbols
DTBNAT_LIN	DTB-Nat lines
DTBNAT_REG	DTB-Nat polygons
HARING_LIN	Regiokaart-Nat lines
HARING_REG	Regiokaart-Nat polygons
HOLDIEP	Beheerkaart-Nat polygons

After the mapping files were created and edited, they were executed within FME. The min-max boundaries of the model have to be given for each mapping file. We have used the following min-max values:

```
XMIN =  45000 m     XMAX = 104000 m
YMIN = 404000 m     YMAX = 437000 m
ZMIN =    -10 m     ZMAX =   1000 m
```

The ZMAX was set to 1000 due to a strange value of 999 in DTB-Nat symbols. It is not clear whether it is a special code or a mistake. The loading was successfully performed with only two problems related to the field of type DATE, i.e. in two records the date was wrong and one record was empty and these were omitted from the data set. The final result was:

- creating the tables in Oracle Spatial,
- populating the tables with data and
- registering the column with geometry in the USER_SDO_GEOM_METADA view.

All the columns containing geometry have the name SHAPE and the unique ID is MSLINK. An example of the creation of a spatial index is given in Appendix 2.

Some statistics:

DTBNAT_SYM has 23083 features/23083 coordinates imported in 49.3 sec
DTBNAT_LIN has 39950 features/417422 coordinates imported in 162.7 sec
DTBNAT_REG has 21556 features/694927 coordinates imported in 163.3 sec
HARING_LIN has 734 features/6041 coordinates imported in 4.7 sec.
HARING_REG has 170 features/9343 coordinates imported in 4.1 sec
HOLDIEP_REG has 76 features/29783 coordinates imported in 5.1 sec

Note: Oracle offers a mechanism (i.e. using SQLLDR) to load data without using FME software. The steps to be followed are a bit different. The user has to create the tables and restructure the original files according the Oracle rules for import. Extensive discussion on this approach can be found in Tijssen et al 2001 or Oosterom et al 2001.

4.2 Results of validation and overlap tests

The validation and consistency checks are executed using functions and operators described in Chapter 3. The tolerance specified for all data sets is 0.000001 m. The geometries were checked with the function SDO_VALIDATE against all the rules for correct geometry. An overlap test was performed to investigate the relationships between the polygons in one layer (assuming polygons should not overlap). All the data sets were checked (with the operator SDO_RELATE) for the relationship ANYINTERACT. If the operator detects interaction, the geometry of the overlapping polygon is calculated using the function SDO_INTERSECTION. The function returns a geometry object that is the intersection (AND operation) of two geometry objects. Finally, the area of the intersecting polygon is computed by means of the SDO_AREA function. The computed area gives an indication over the type of overlap. A procedure in PL/SQL is written, which processes the polygons of the three maps (DTB-Nat, Regiokaart-Nat and Beheerkaart-Nat) in the way described above. The script is given in Appendix 3.

Beheerkaart-Nat and Regiokaart-Nat: These maps revealed no invalid geometries or unwanted overlaps (overlap between polygons in the same layer).

DTB-Nat: Several types of problems can be identified with polygons in DTB-Nat:

- The most common error is invalid attribute values. Many objects should have a LAYER value different than the basic layer 1. We assume that the objects in layer 1 should be non-overlapping (form a planar partition).
- Invalid geometries: three objects (MSLINK=1511, 17087, 20443) have geometries that do not follow the rules for allowed geometries, as two rules were violated: no self-intersection and no duplicate points in the list of coordinates. Since these are rules adopted by virtually every system capable of processing geometry, errors like these can only occur if geometry is created without being properly checked.
- Incorrect geometries: many objects have missing or superfluous "holes".
- Inaccurate geometries: many objects have small corners or "slivers", which overlap with neighbouring objects. It looks as if these objects should have identical boundaries (which can be enforced best if a planar partition with topology is used).
- Incomplete coverage: sometimes "gaps" exist between objects (see Figure 17, right).

We have cleaned up only some of the errors. The 3 invalid geometries were removed but all the incorrect and inaccurate objects and errors due to wrong attribute values are still in the data sets used in the tests.

The result of the second test (overlapping objects in a layer) is presented in Appendix 4. The procedure creates a table to contain information about overlapping objects. The result table has the following columns:

OID_1	NUM_1	OID_2	NUM_2	AREA_1 (m^2)	AREA_2 (m^2)	AREA_12 (m^2)
2091	2	20443	16	193.8428	131393.3416	7.7223
2102	2	20443	16	602.3114	131393.3416	25.0793
2109	2	20443	16	53.4034	131393.3416	0.0096
2192	2	20475	2	88.9262	6380337.9336	88.9262
				.		
				.		

Each line of this table identifies two objects that overlap. The objects involved are in the OID_1 and OID_2 columns, their areas are in AREA_1 and AREA_2, the area of the overlap is in AREA_12. For example, line 1 reveals that a small part of object 2091 is overlapping with object 20443, line 4 shows that object 2192 is completely inside object 20475. The NUM_1 and NUM_2 columns indicate the number of times an object is "involved" in an overlap with other objects. As each object is compared to all other objects, an overlap between two objects is detected twice (once while checking A against B and once while checking B against A). To get the proper number of overlaps one has to divide NUM_1 and NUM_2 by 2. For example, object 20443 has an overlap with 8 other objects. Given this logic it should not be possible for NUM_1 and NUM_2 to have uneven values, but it still does occur. This is the result of an unknown problem Oracle has with some of the objects in the data set. The "problem" objects have a tiny overlap with another object and while working on them the SDO_INTERSECTION function crashes with an 'internal error, contact Oracle' message. The two objects involved are removed from the final data set. Data-related bugs like these occur occasionally in Oracle Spatial, the number of them decreases with every new version of the software.

4.3 Establishing a link between Regiokaart-Nat and Beheerkaart-Nat

By definition the content of Beheerkaart- and Regiokaart-Nat are the same, the only difference is with visualisation. The test here shows the capabilities of Oracle Spatial with comparing independently derived datasets. The datasets used in this research are independently developed at different times, most likely introducing inconsistencies.

The correspondence between the objects in the two data sets (the data sets with the polygons were used) was checked for the relationships 1:1, m:1 and m:n (Beheerkaart-Nat and Regiokaart_Nat). This was done in two steps by comparing overlapping areas and comparing codes of objects.

Furthermore, the objects from Beheerkaart-Nat and Regiokaart-Nat lines that do not have object codes were deleted (agreed with Paul van Asperen). All the objects of the Beheerkaart-Nat must have codes. In this respect it is not clear why 8 objects from Beheerkaart-Nat are without codes (i.e. MSLINK= 8,17, 22, 23, 31, 38, 45, 52). Some of these objects (see Figure 7 and Figure 8) are very small (do not exist in the Regiokaart-Nat) and most probably belong to a larger object (e.g. MSLINK=21, *Oevers*). Their code cannot be obtained by overlapping with Regiokaart-Nat objects since they would be overlapped by the wrong objects (*bodems*, instead of *oevers*). Object 38 from Beheerkaart-Nat is also not present in the Regiokaart-Nat and the code should not be related to *bodems* (Figure 9). Object 45 is at the edge of the overlapping area and somehow generalised (Figure 10). Object 52 does not have a code in both maps (Figure 11).

All the codes from the Regiokaart-Nat were changed according to the new coding system as it is in the Beheerkaart-Nat. That is, "112B" from the Regiokaart-Nat was changed to "108C".

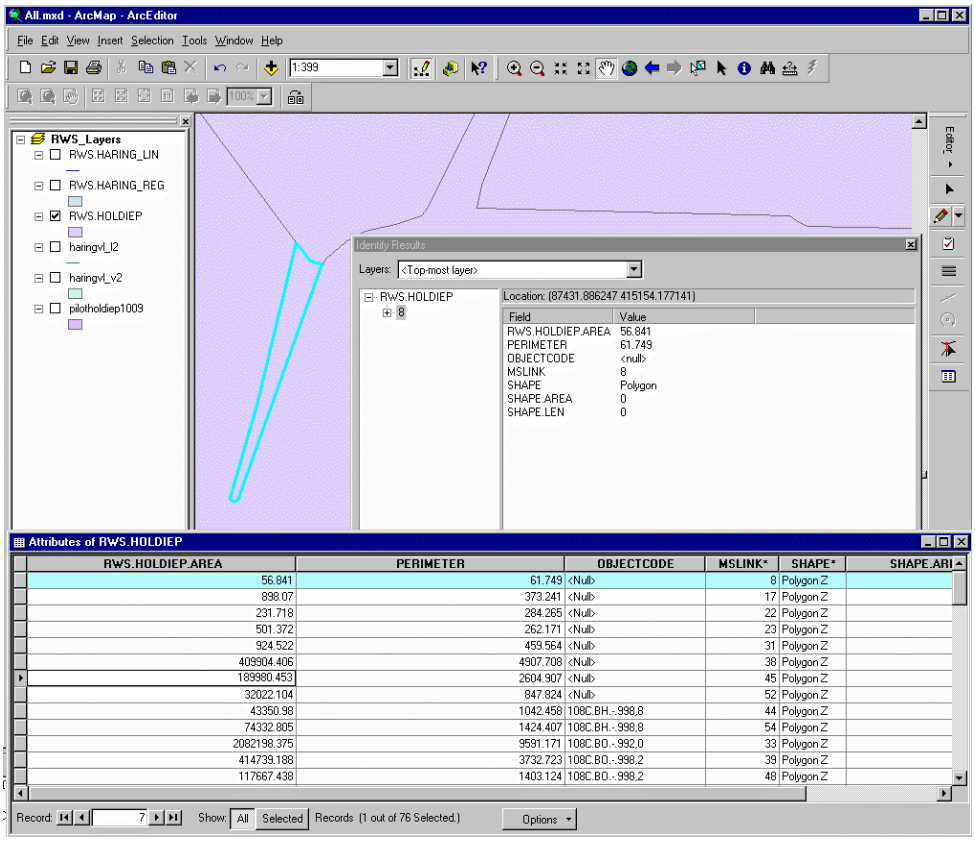


Figure 7: Object of Beheerkaart-Nat with MSLINK=8.

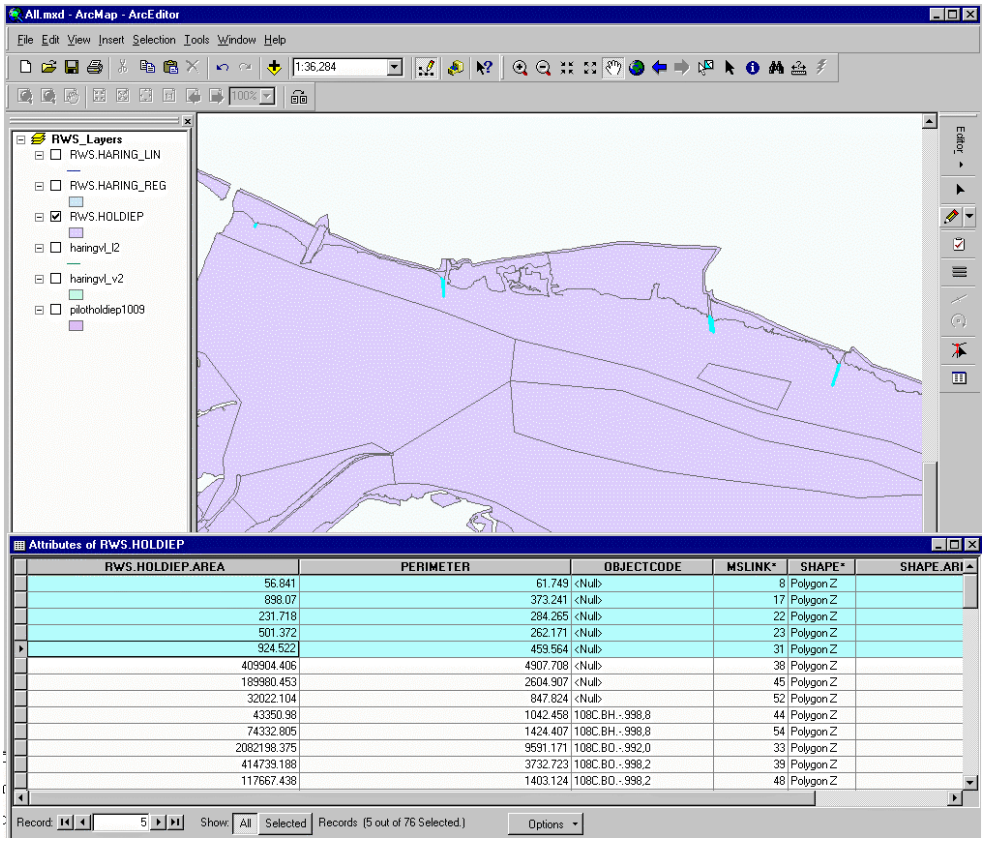


Figure 8: All the small objects from the Beheerkaart-Nat without codes.

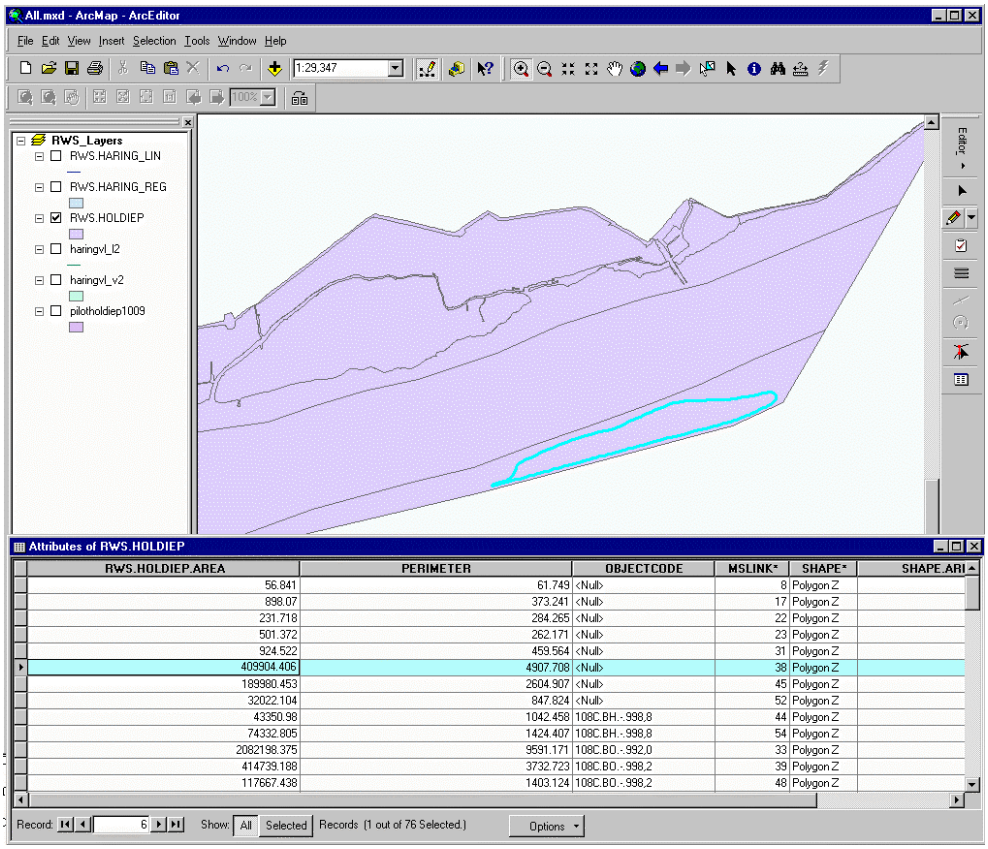


Figure 9: Object with MSLINK=38 from the Beheerkaart-Nat that does not exist in the Regiokaart-Nat.

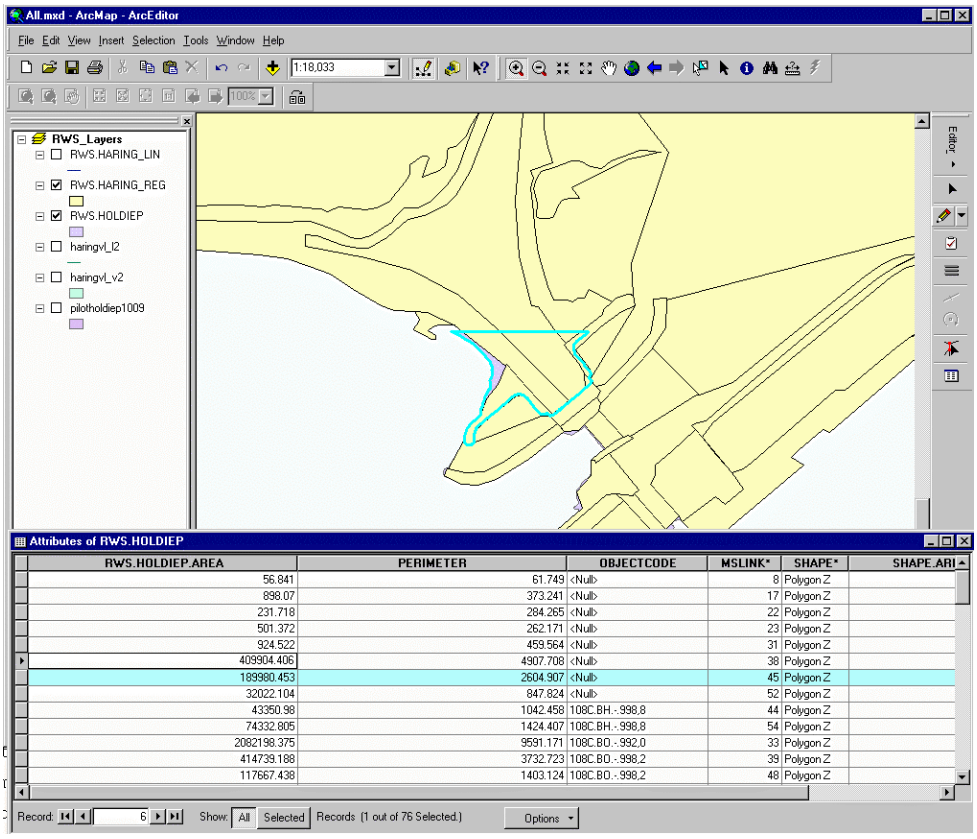


Figure 10: Object with MSLINK=45 from the Beheerkaart-Nat.

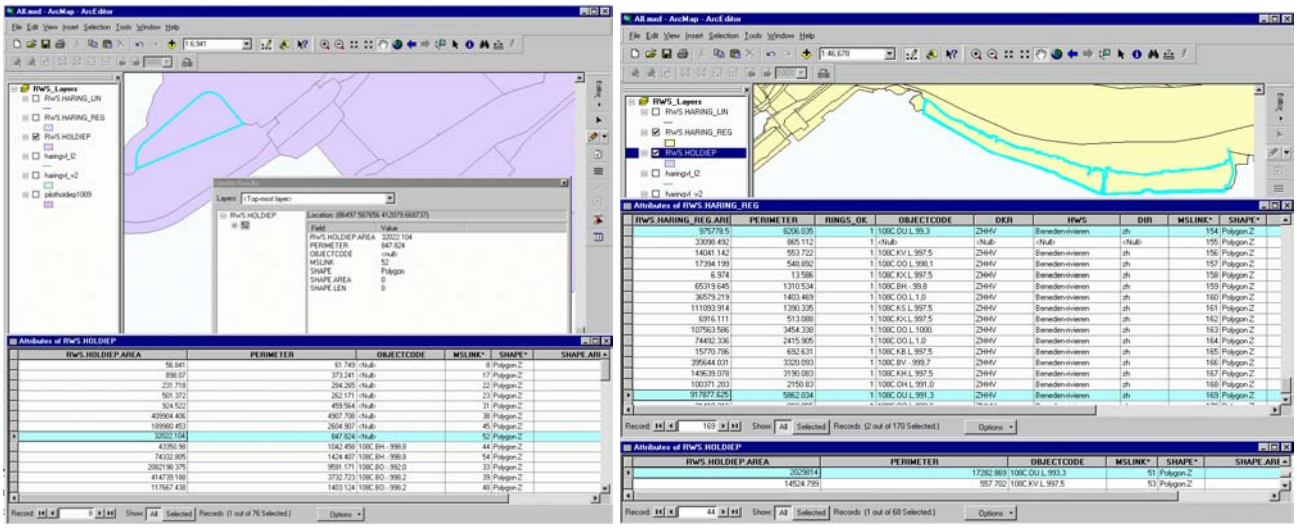


Figure 11: Object 52 (Beheerkaart-Nat) without code in both maps (left) and Object 51 (Beheerkaart-Nat) matched with two objects from the Regiokaart-Nat (right).

Comparing overlapping areas: This operation was performed following a procedure similar to the one described above for overlapping objects within the same map. Each object of one of the maps is checked against ANYINTERACT with all the objects from the second map. Then the geometry of the overlapping parts is calculated and their area is computed. These areas are further compared with the areas of the objects and the percentage is computed. The results are stored in a separate Oracle table (part of the content is given in Appendix 5). The description of the result table created is:

```
Name                 Type
-------------------  ------------------
IN_OID               NUMBER(12)
IN_AREA              NUMBER
IN_PCT_OVERLAP       NUMBER
MT_OID               NUMBER(12)
MT_AREA              NUMBER
MT_PCT_OVERLAP       NUMBER
OVERLAP_GEOM         MDSYS.SDO_GEOMETRY
```

The IN.... columns refer to the input layer (the first table specified in the script, Beheerkaart-Nat in this case), the MT.... columns refer to the layer matched against (the second table specified, Regiokaart-Nat in this example).

Our experiments have shown that objects with overlapping areas larger than 60% can be considered as one object (see Appendix 6). All the 68 objects (originally 76, 8 deleted without code) from the Beheerkaart-Nat were tested and the following results were obtained: 51 objects from the Beheerkaart-Nat are matched with only one object from the Regiokaart-Nat (84%). 2 objects (MSLINK=5, 51) from the Beheerkaart-Nat are matched with more than two objects from the Regiokaart-Nat (Figure 11, right). 4 objects (MSLINK= 100, 104, 113, 141) from the Regiokaart-Nat are matched with more than one object from the Beheerkaart-Nat. One object from the Beheerkaart-Nat (MSLINK=10) was not matched with any object. One object (MSLINK=10) from the Beheerkaart-Nat is not matched with objects from Regiokaart-Nat (overlapping area is smaller than 60%). How many objects from the Regiokaart-Nat are not matched with any objects is difficult to check, since the area covered by the Regiokaart-Nat is much larger.

Comparing codes: The second step, aimed at resolving multiple matches between the objects in the two maps, is comparing the codes. The code is a composition of numbers and letters separated by dots. The first group of three numbers and a letter (e.g. 108C) is related to the Watersysteemdelen in a Dienstkring. The second group of two letters (e.g. OO) corresponds to the classification of objects in the TISBO system. The third group of one letter is a sub-classification, the last group is a number giving the river-kilometres markers. Since the objects without codes in the Regiokaart-Nat were not deleted, those objects were matched only on the basis of the area

overlap. The assumption was that they could get the code of the objects in the Beheerkaart-Nat. The results of the test are given in Appendix 7. 43 objects of the Beheerkaart-Nat are matched with 38 objects from the Regiokaart-Nat (55%). The multiple relations (m:1) left are only two: objects 3,7,11 (Beheerkaart-Nat) are now clearly part of object 104 (Regiokaart-Nat) and objects 2,4 (Beheerkaart-Nat) are a part of object 100 (Regiokaart-Nat). However, a closer look at all the matches revealed that this is the result of different coding of the objects (13 cases) and wrong codes (11 cases). A large number of the errors are apparently typing errors in the kilometre field. For example:

BEH_ID	REG_ID	BEH_CODE	REG_CODE
44	146	108C.BH.-.998,8	108C.BH.-.99,8
51	154	108C.OU.L.993,3	108C.OU.L.99,3
54	159	108C.BH.-.998,8	108C.BH.-.99,8
56	160	108C.OO.L.1000,0	108C.OO.L.1,0
59	163	108C.OO.L.1000,0	108C.OO.L.1000.
60	141	108C.KB.L.997,5	108C.FT.L.997,5
63	164	108C.OO.L.1000.0	108C.OO.L.1,0

Therefore, a new match was attempted using only the first 10 characters of the code. The number of matched objects increased to 54 (70%) (see Appendix 8). Still many objects were not matched but this is mostly because they are different objects. For example, object 141 from Regiokaart-Nat (code: 108C.FT.L.) covers 9 smaller objects from Beheerkaart-Nat (different codes). This case still can be resolved by checking the overlapping area. If there is a 100% overlap then the smaller objects are inside the larger object and it is probably acceptable to have different codes.

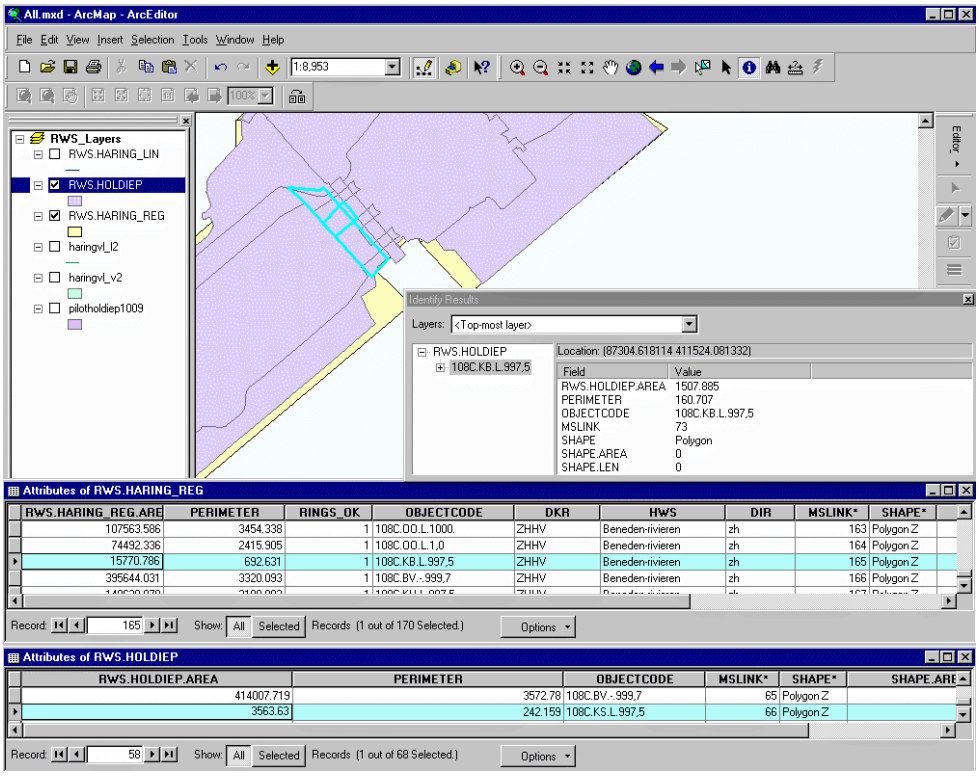


Figure 12: Wrongly determined object in the Regiokaart-Nat: the object coded as bridge.

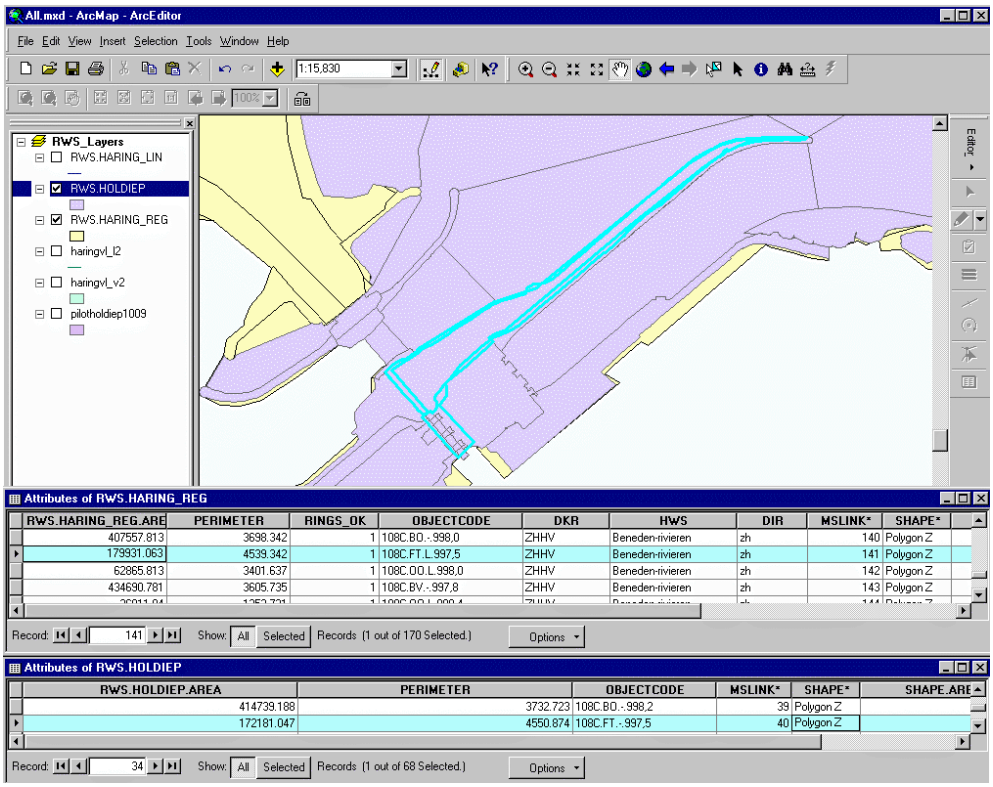


Figure 13: The real bridge incorporated in an object of type facilitair *(FT).*

After applying this restriction, two interesting cases were discovered: the matches of objects with MSLINK = 66, 70 (Beheerkaart-Nat) and the corresponding objects with MSLINK = 165, 141 (Regiokaart-Nat). Figure 12 is an illustration of match 66/165. The object from the Regiokaart-Nat is given with a code for *beweegbare brug* (KB). Object 66 matches very well with object 165 where the area overlap is concerned, but failed when the codes were compared. Something similar happens with objects 70 (Beheerkaart-Nat) and 141 (Regiokaart-Nat).

4.4 Establishing a link between Beheerkaart-Nat and DTB-Nat

In principle, the link between Beheerkaart-Nat and DTB-Nat can be established in two ways:

- Applying the same straightforward approach as for the objects between Beheerkaart-Nat and Regiokaart-Nat.
- Modifying the DTB-Nat objects by uniting objects that are artificially separated because they are created from two different stereo-pairs and then applying the procedure for overlap.

We have completed tests applying both approaches. For this purpose appropriate scripts were written. In the first case, the script simply loops through all geometries of a layer and calculates all possible intersections with the geometries of another layer. The result is written to a log file and to a table. The first records of the corresponding result table are:

```
select IN_OID,IN_AREA,IN_PCT_OVERLAP,MT_OID,MT_AREA,MT_PCT_OVERLAP
       from DTBNAT_REG_HOLDIEP_MATCH;

    IN_OID    IN_AREA IN_PCT_OVERLAP     MT_OID    MT_AREA MT_PCT_OVERLAP
---------- ---------- --------------   -------- ---------- --------------
        55 59157.2308             100       32 220667.824     26.8082721
        56 12471.3637             100       32 220667.824     5.65164574
        58 105.499966      19.0248235        3 385629.347     .005204786
        59 2368.51568      25.7240072        3 385629.347     .157995534
        63 2449.55804             100       35 313500.213     .781357695
        74 3626.56375             100       13 756923.473     .479118943
        83  85756.046             100       13 756923.473      11.329553
```

Clearly, when a DTB-Nat object has 100% overlap with Beheerkaart-Nat object then the object is completely inside the Beheerkaart-Nat object and can be linked to it. Probably it has to be a multi-step process in which first objects that have a 100% match can be linked, then link "obvious" objects (those which are e.g. 60% or 70% or more inside a matching object). The remaining objects will require some additional "rules" (e.g. assign the object to the matching object with the largest overlap if this overlap is more than 50% of the input object). Probably some objects will remain which cannot be matched. At a certain stage the match between the objects has to be handled in a different way. We have matched all the objects that have an overlap of 100% (58%), larger than 80% (70%) and larger than 60% (74%). The ID of the Beheerkaart-Nat object is stored in an additional column in the DTB-Nat table. Also in this procedure some data-related Oracle Spatial errors occurred, for 5 objects the geometric intersection could not be calculated.

The second approach (aggregating DTB-Nat objects first where possible) has the advantage of significantly reducing the number of objects in the DTB-Nat. The apparent question is what should be a criterion for aggregating the objects. Bearing in mind the possible operations in Oracle Spatial, it is possible to compute the geometry of an object that is a union of two objects. Using this function, all the objects with the same OMSCHR that have common boundaries can be recursively aggregated in one large object. However, in case of rivers this process can result in rather large objects (e.g. the river will not stop at a bridge, since the bridge is in another layer). Having already the objects of Beheerkaart-Nat, it is appropriate to limit such objects to the limits of the Beheerkaart-Nat objects. Since the Beheerkaart-Nat is created per Dienstkring, the parts of the river, for example, can be related to the objects Watersysteemdelen (WSD).

These experiments where completed in two steps:

Beheerkaart-Nat: Creating new objects out of the Beheerkaart-Nat objects that can be considered a part of a river. The process is carried out by a script that extracts all objects that have touching boundaries and have codes corresponding to the object category *Bodems*, i.e. BV, BH, BO and creates a new object out of them. For the purpose the standard Oracle Spatial function SDO_UNION can be used:

```
newshape:= SDO_GEOM.SDO_UNION (newshape, shape2(j), TOL);
```

Figure 14 illustrates one of the new objects. The total number of objects obtained after this procedure is 6 (stored in a separate table called 'BEHEER_RIVER_UNION').

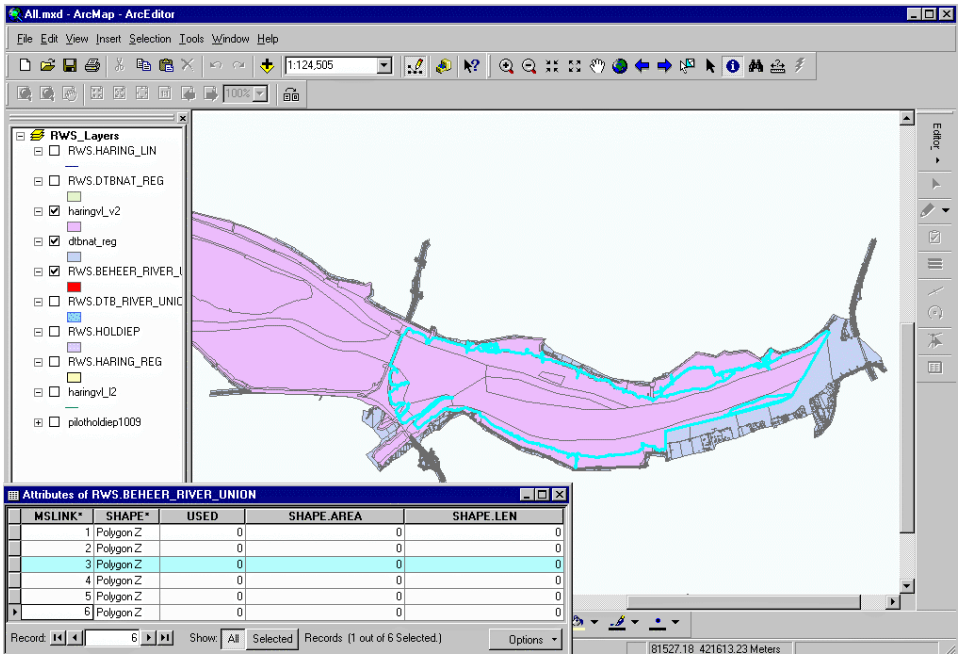


Figure 14: A new object of the Regiokaart-Nat, created as union from all the objects that have common boundaries and code Bodems.

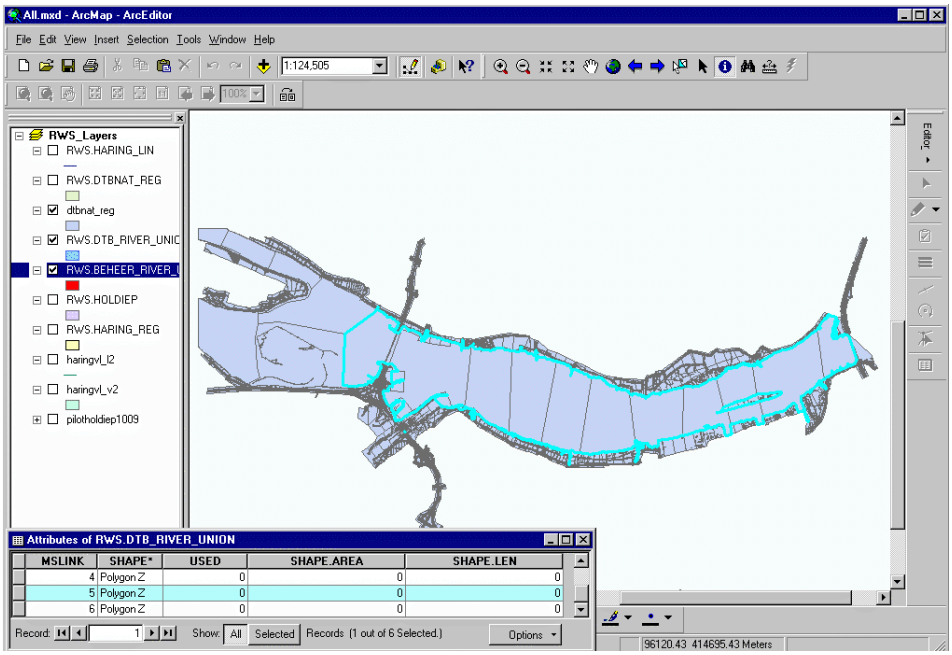


Figure 15: The original DTB-Nat objects and the new object obtained by SDO_UNION.

DTB-Nat: The union of the DTB-Nat objects river is completed with the help of the "river"-objects created at the previous step. The DTB-objects that have OMSCHR ='rivier' are exported in a separate table. The "river"-objects from the Beheerkaart-Nat are matched with the DTB-Nat objects and those which interact (using the Oracle standard function MDSYS.SDO_RELATE) are united (using the same Oracle Spatial function SDO_GEOM.SDO_UNION). Figure 15 shows one of the new objects. The new object is a result of 25 original objects from DTB-Nat. As can be noticed, the object extends under the bridge (Figure 16). However, the user can decide which boundary should be considered as leading. In our case, we have preserved the outer boundaries of the DTB-Nat objects. It is possible however to create a new object (from the DTB-objects) of which the boundary coincides with the boundary of the BeheerObject.

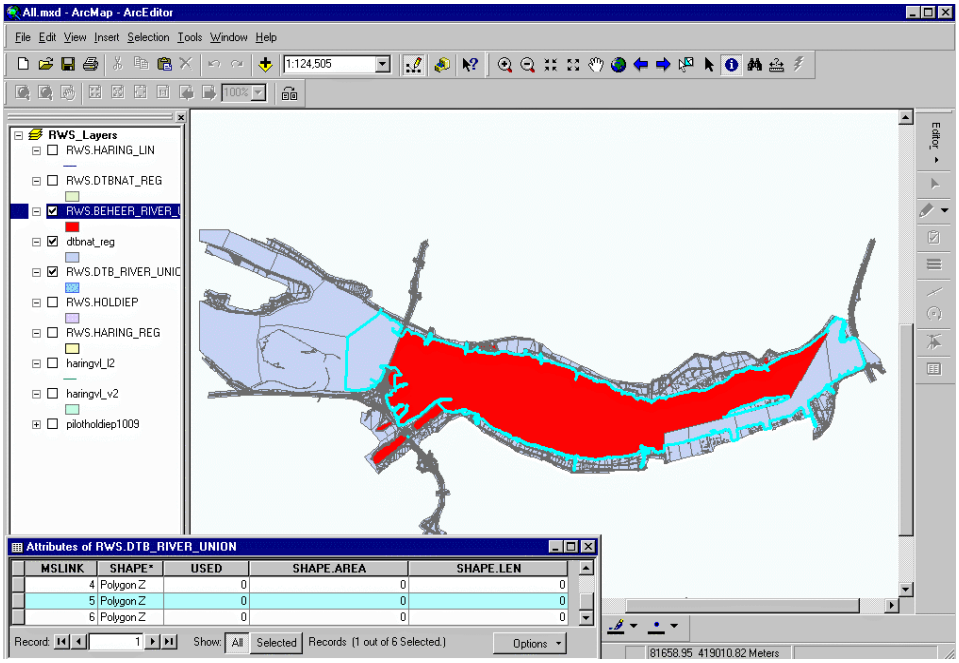


Figure 16: The new object (union of DTB-Nat objects) superimposed with the corresponding object from the Beheerkaart-Nat.

While applying this procedure, another type of error in the DTB-Nat data set was discovered, i.e. a lack of complete coverage of objects. Object 1 and 2 are both rivers but could not be aggregated into one object, although the object from the Beheerkaart-Nat is only one. Figure 17 shows the part of the river that is still a separate object (left) and the gap between the objects that "prevents" the aggregation of the object.

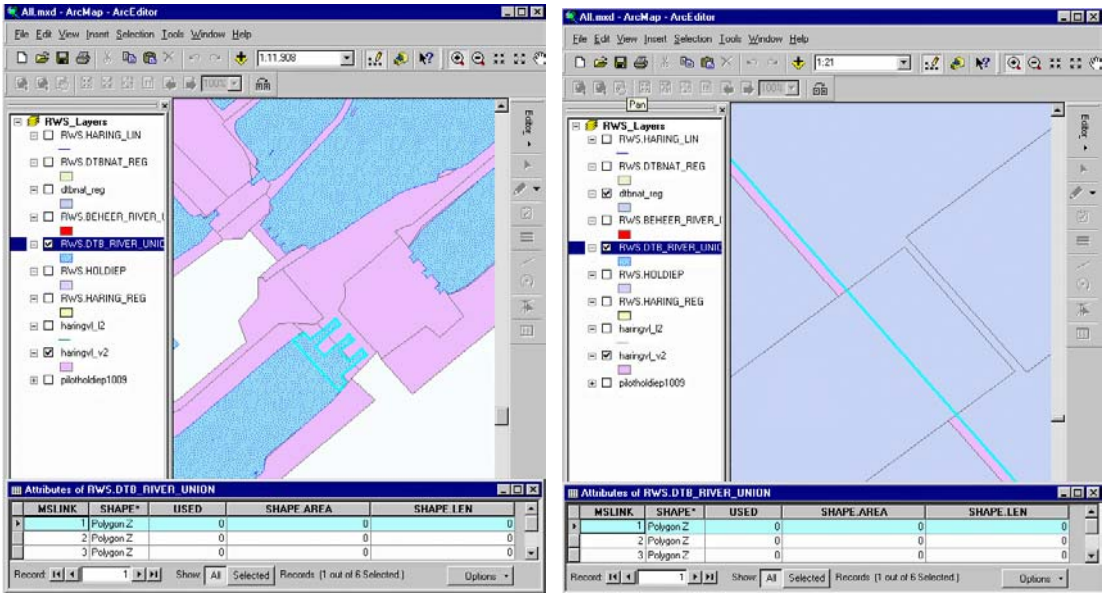


Figure 17: Lack of complete coverage:

object 1 cannot be aggregated with object 2 (left) and the gap between them (right).

4.5 Creating Beheerkaart-Nat from Regiokaart-Nat and DTB-Nat

The last test was performed to demonstrate that the functionality offered by Oracle Spatial can be used to facilitate the creation of the Beheerkaart-Nat in areas where both DTB-Nat and Regiokaart exist. The hypothesis was: if the RegiokaartObjects differ from the BeheerObjects only in the level of detail and the BeheerObject follows the boundaries of the DTBObjects, then we have to be able to obtain the boundary of the BeheerObject by aggregating all the DTBObjects that are inside the RegiokaartObject. Due to time constraints, we did not prepare scripts for all objects from the Regiokaart-Nat and DTB-Nat. We tested the hypothesis for two objects, i.e. RegiokaartObject 128 that corresponds to BeheerObject 28 (Figure 18) and RegiokaartObject 19 (Figure 19) that is outside the area covered by the Beheerkaart-Nat. Similar to the previous tests several standard Oracle functions and operators were used as well as the description of the DTBObjects. For example, the DTBObjects were restricted to be NOT *rivier* in both cases.

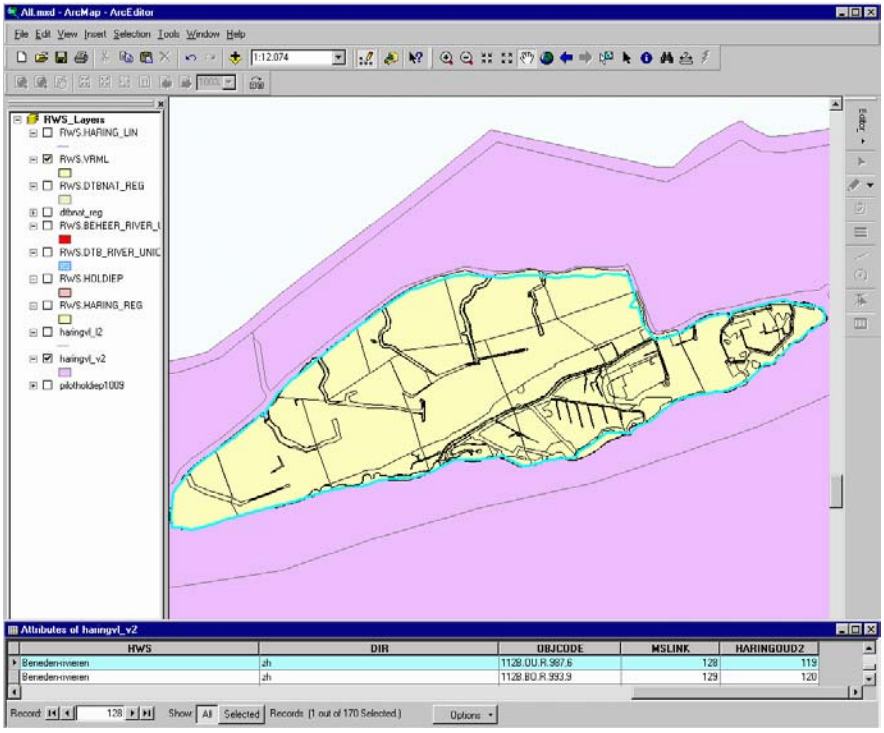


Figure 18: Object 128 of the Regiokaart-Nat (the blue line) used to obtain a new object (aggregation of DTBObjects) that corresponds exactly to BeheerObject 28.

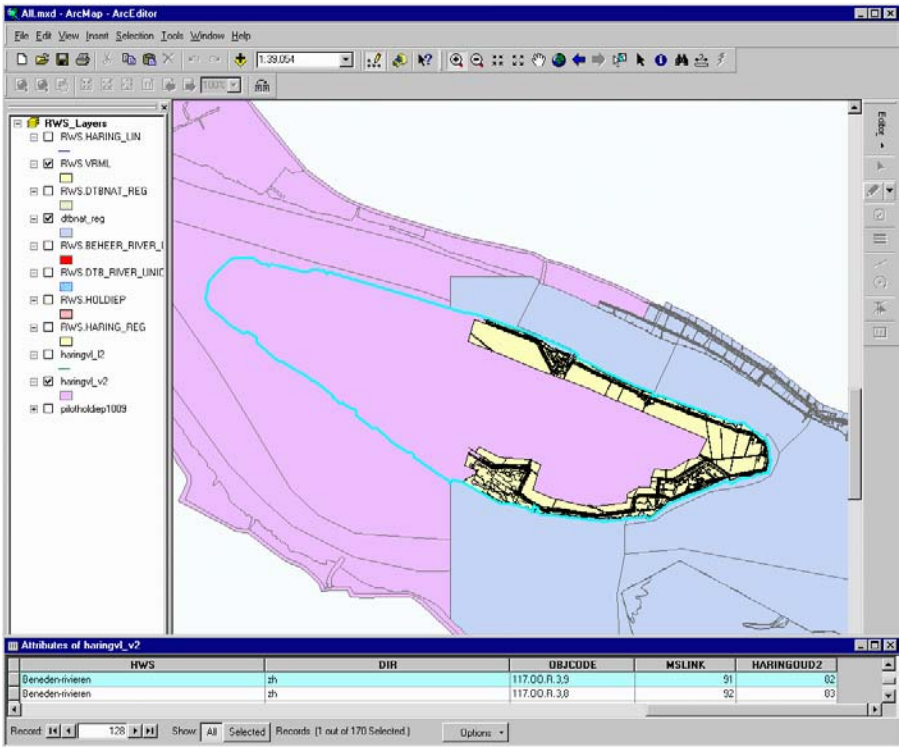


Figure 19: Object 19 from the Regiokaart-Nat and corresponding BeheerObject (it is outside the area covered by the existing Beheerkaart-Nat area).

5 PROPOSED MODEL

The tests performed on the three data sets have shown that a link between the objects can be established using standard functions of Oracle Spatial. Although the process is completely automatic, user interaction would be still necessary to resolve doubtful cases or mistakes.

The tests confirmed the opinion that maintenance of the Regiokaart-Nat can be omitted. Firstly, the maintenance of two maps with the same objects increases the possibilities for introducing errors. Secondly, the objects from the Beheerkaart-Nat and Regiokaart-Nat are practically the same and can be easily identified as such. Approximately 80% of the objects from the two maps can be matched uniquely with respect to the geometry (position and the shape of the objects). About 15% of the failures to match objects is a result of wrong object codes in the Regiokaart-Nat. Most of the errors in the codes are in the part containing numbers. A split of the object code into two parts, i.e. a string and a number, will reduce significantly the possibility for introducing errors because of redundant storage (e.g. typing a coma instead of a dot).

Most of the experiments were performed on the Regiokaart-Nat polygons. The lines of the Regiokaart-Nat (after deleting the lines that are part of Regiokaart-Nat objects, i.e. without codes) are not matched with Beheerkaart-Nat objects, since they (only four lines lie in the area covered by Beheerkaart-Nat polygons) appear to be borders of Beheerkaart-Nat objects.

Bearing in mind these considerations, the Regiokaart-Nat is not considered any further.

The functionality currently offered by Oracle Spatial allows establishing a link between the Beheerkaart-Nat and DTB-Nat. The process, as mentioned above, is expected to be more complex and requiring several iterations compared to the link between the Regiokaart-Nat and the Beheerkaart-Nat. The number of objects in the DTB-Nat is rather large and there are still quite a lot of errors (e.g. self-overlapping polygons, gaps, wrong layers, etc). As it was shown, many DTB-Nat objects (parts of rivers, or other objects) can be merged first (using appropriate conditions) and after that linked to the Beheerkaart-Nat objects. The link can be further controlled with respect to the classification of the two types of objects (CTE code and TISBO code). All the tests were completed on the tables with the polygons, since they were considered to represent the more general and complex case. Point objects can be easily located inside Beheerkaart-Nat objects or DTB-Nat polygons. The lines of DTB-Nat were not investigated.

Consultation with the RWS specialists has contributed to the cleaning up of the data stored in the Oracle tables. The description of the tables DTB-Nat (polygons) and Beheerkaart-Nat is given below:

DTB-Nat (table name DTB_NAT):

Name	Type
CTE	VARCHAR2(8)
DATUM	DATE
DTM	VARCHAR2(1)
EIGNAM	VARCHAR2(20)
LAYER	VARCHAR2(1)
OMSCHR	VARCHAR2(37)
THEMA	VARCHAR2(20)
THEMB	VARCHAR2(17)
MSLINK	NUMBER(38)
SHAPE	MDSYS.SDO_GEOMETRY

Beheerkaart-Nat (table name HOLDIEP):

Name	Type
AREA	NUMBER
PERIMETER	NUMBER
OBJECTCODE	VARCHAR2(20)
MSLINK	NUMBER(38)
SHAPE	MDSYS.SDO_GEOMETRY

Regiokaart-Nat (table-name HARING_REG):

Name	Type
AREA	NUMBER(16,3)
PERIMETER	NUMBER(16,3)
RINGS_OK	NUMBER(12)
OBJECTCODE	VARCHAR2(20)
DKR	VARCHAR2(40)
HWS	VARCHAR2(40)
DIR	VARCHAR2(40)
MSLINK	NUMBER(38)
SHAPE	MDSYS.SDO_GEOMETRY

This content corresponds in large extent to the DTB2000_INFO model and TISBO classification of objects. It should be noticed that the Beheerkaart-Nat does not hold information about *WaterSysteemdelen*, and *HoofdWatersysteemDelen*. However, this information can be obtained from the Regiokaart when the link is established. The DTB2000_INFO maintains 16 types of data to be considered for describing DTB-Nat and DTB-Droog objects (Gebruikershandleiding DTB2000, 2000). In a way it is metadata about DTB-Nat and DTB-Droog and is created to facilitate the user:

CTE-code (Classificatie Topografische Elementen)
- B bebouwing
- F samengestelde
- H hoogte
- L leiding
- MD MD-elementen die niet in CTE zijn opgenomen
- N grond
- Q objecten wegen en verkeer
- R objecten
- T terreinafscheiding
- V verharding
- W water
- ZH constructielijnen hoogte

Naam topografisch element
Vorm (punt, lijn of vlak)

Klasse precisie (2,3 of 4):	xy	z
Niveau 1 (terrestrisch)	4	4
Niveau 2 (fotogr. hard)	5	9
Niveau 3 (fotogr. middel)	8.5	10
Niveau 4 (fotogr. zacht)	15.5	12,5

Element komt voor in DTB-Droog
Element komt voor in DTB-Nat
Thema, gebruikte attribuut
Themb, gebruikte attribuut indien multicodering is gebruikt
Layer, layer in welke element voor kan komen (1,2,3,4)
DTM (j,n) kan of wordt element gebruikt voor maaiveld DTM
Naam AutoCad-Layer
Naam Arc/Info coverage of ArcView 3D shape-file
Naar Arc/Info coverage KernGis
TIN-code, code voor opbouw van een TIN, in geval element het maaiveld beschrijft

punt	1
hard breakline	3
hard erase polygon	9

MX-label, gebruikte label voor MX
Overlappend vlak in maatwerk shape-file punt, lijn en vlak

The information is important because it gives an indication what kind of data is interesting for the user. The classification of the Beheerobjects is (van de Brink et al, 2002):

Kunstwerk	Aanleginrichting KA	aantal stuks steigers, meerpalen
	Beweegbare brug KB	aantal stuks brug te A
	Gemaal KG	aantal stuks gemaal X
	Hoogwaterkering KH	aantal stuks stormvloedkering
	Kunstwerken t.b.v. natuur KN	aantal stuks vistrap
	Aquaducten KQ	aantal stuks aquaduct te B
	Waterreguleringswerk KR	aantal stuks overlaat
	Schutsluis KS	aantal stuks grote schutsluis Y
	Tunnel KT	aantal stuks tunnel te A
	Spui/uitwateringsluis KU	aantal stuks spui bij kunstwerk X
	Vaste brug KV	aantal stuks brug Z

	Stuw KW	aantal stuks stuw X
	Sifon/duiker/hevel KZ	aantal stuks grondduiker
Bodems	Vaargeulbodem (vak) BV	lengte km kanaalpand
	Havenbodem BH	oppervlakte ha overnachtingshaven
	Bodems (overig) BO	oppervlakte ha tussen kribben
Oevers	Strekdam OD	lengte km strekdam Y
	Oever haven OH	lengte km kade
	Kribben OK	aantal stuks kribben te A
	Leidam/leikade OL	lengte km leidam X
	Oevers/dijken OO	lengte km kustlijn, linkeroever
	Strandhoofd OS	lengte km strandhoofd Vlieland
	Uiterwaarden OU	oppervlakte ha uiterwaarden te B
	Kribvakken OV	lengte km Tiel 1
	Waddenzeekwelders OW	oppervlakte ha kwelderareaal te A
Water		Water QQ oppervlakte km2 watervak
Facilitair	Gebouwen FG	aantal stuks dienstkringkantoor, werkplaats
	Scheepvaartbegeleiding FS	havenlichten, verkeerspost
	Terreinen FT	oppervlakte ha opslagterrein, zanddepot
	Vaartuigen FV	aantal stuks patrouillevaartuig, ponton
	Algemeen FX	aantal stuks dienstauto's

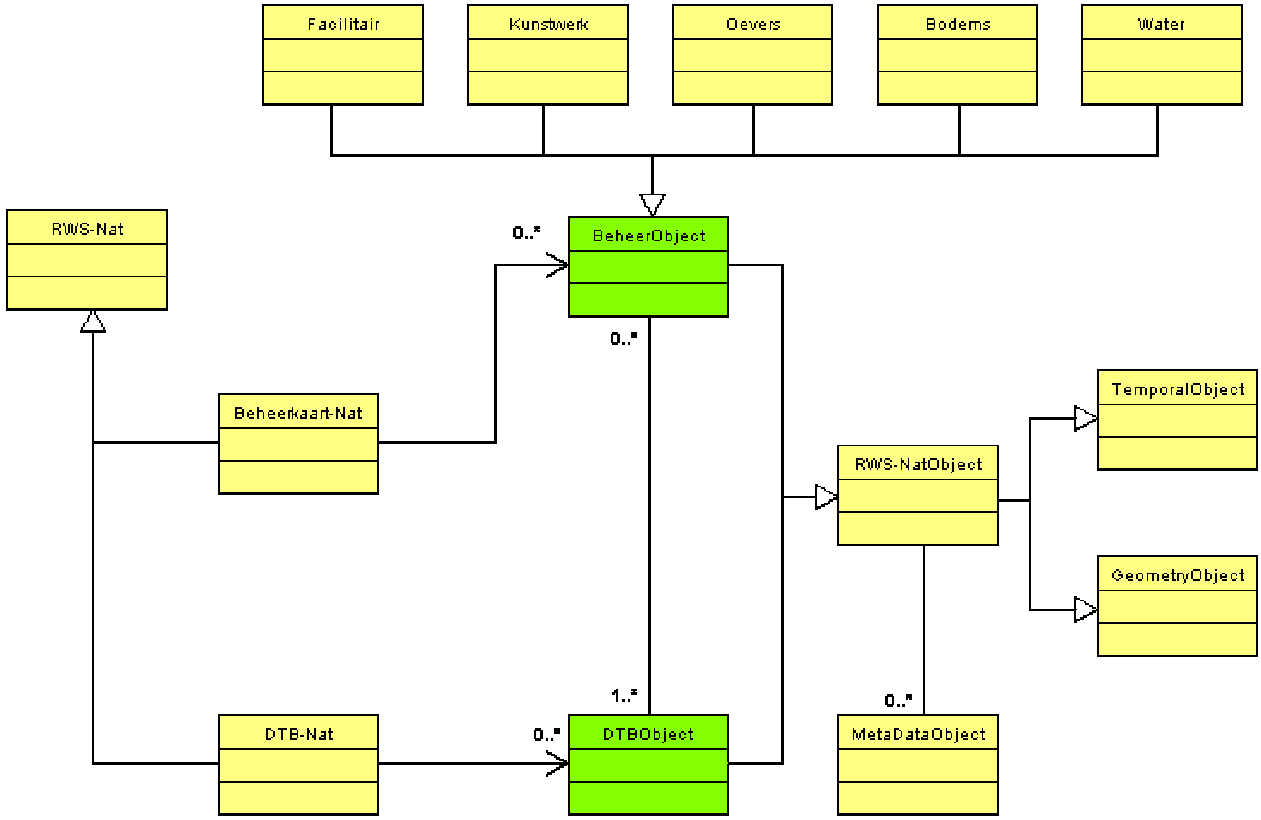


Figure 20: UML representation of the proposed model.

Bearing in mind the functionality of Oracle Spatial (and the tests performed) and the existing classification and specification of DTB-Nat and Beheerkaart-Nat we propose an integrated maintenance of DTB-Nat and Beheerobjects as is shown in Figure 20. The major object classes according to the TISBO categories are included in the schema. The BeheerObject and the DTBObject are RWS-NatObjects to be maintained in the model with geometry (GeometryObject), temporal aspects (TemporalObject: date of creation, updates and "dead") and metadata (MetaDataObject). The BeheerObject is associated with the Beheerkaart-Nat super class and all the objects form a planar partition. Similarly the DTBObject is part of the DTB-Nat super class. These two super-classes are part of RWS-Nat. The link between BeheerObject and DTBObject is given as an association (*Note: we recommend to the reader the GISt report on the second GML prototype of the TOP10vector object model for more information on this type of modelling; Vries et al, 2002).*

The object associations can be further specialised with respect to the object categories and the types of objects in DTB-Nat. For example, the objects of class *Water* (DTB-Nat) can be associated only with objects from the categories *Water* or *Bodems* (Beheerkaart-Nat). Further investigations are needed to check the semantic relations between the objects of DTB-Nat and Beheerkaart-Nat. The establishment of a semantic link between the objects will further contribute to the control and consistency maintenance. Further details on semantic integration can be found in Uitermark, 2001.

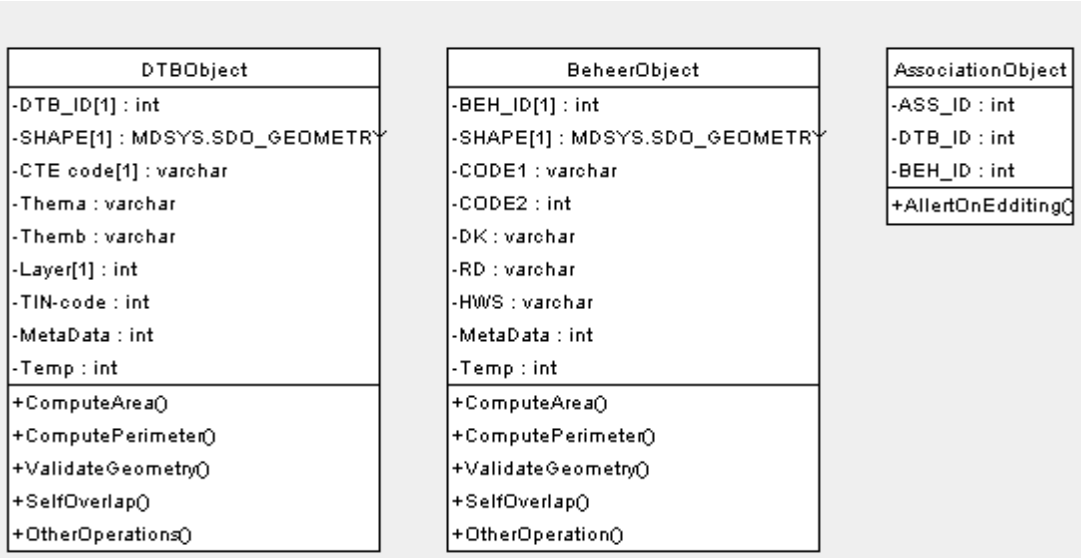


Figure 21: Type of information maintained for the objects.

Figure 21 illustrates the information that is proposed for maintenance in the database. The information about geometry is organised in the object tables (BeheerObject and DTBObject) together with the other attributes specific for the two types of objects. The metadata information and the temporal aspect can be stored in separate tables with only a link to the corresponding records in the object tables. The association between the two objects will be given in a separate table AssociationObject that will ensure the maintenance of multiple relationships.

DTB-Object: In the proposed model, each object has a unique ID (DTB_ID), which is assigned automatically by the DBMS when the object is created. The geometry of the object is stored in a column named SHAPE. Note, the type of geometry (point, line, polygon, collection, etc.) can be obtained from the geometry and therefore there is no need to store the type explicitly. It should not be forgotten that ArcGIS cannot work with tables with mixed geometries. Eventually, views can be created that can refer to only one geometry type (we did not test this possibility). A number of fields are the same as in DTB2000_INFO, i.e. indication whether it is Thema, Themb, Layer or TIN. According to us, the information in field DTM can be obtained from the information stored in TIN and Layer and therefore this field is also omitted. Oracle Spatial offers functions to compute area and perimeter. The MetaData attribute will provide a link to a MetaData table containing more information about the origin of the object. Furthermore, Temp will link temporal information for every object to the table with temporal data.

BeheerObject: The attributes of the BeheerObject do not differ significantly from the existing situation. As discussed before, the TISBO code is split into two parts (string and number). The information about the Dienstkringen and Regionale Directies is added as well as a link to a metadata table and a table maintaining the temporal aspect.

ObjectAssociation. The table maintains the link between the objects, as multiple references can exist between objects. The columns of the table will be unique combinations of the link ASS_ID and the IDs of the corresponding objects from Beheerkaart-Nat and DTB-Nat.

Since topology is not maintained, editing of the objects has to be strictly controlled either at a front-end level or using the topological operators offered by Oracle Spatial.

6 USING DIFFERENT FRONT-ENDS

Visualisation of Oracle Spatial data in ArcGIS and GeoGraphics is quite straightforward. Suppose the columns with the geometries are registered in the spatial metadata table in Oracle and the index is built, the geometries can be accessed and visualised in both software products.

6.1 ArcGIS (www.esri.nl)

ArcGIS (direct connect): ArcGIS requires the column that contains the unique ID (in our case MSLINK) to be declared as NUMBER(38) to avoid problems. The next step should be the registration of the table containing geometry in the metadata tables of ArcSDE. The registration can be done by means of the 'auto-register' feature of direct connect, or by means of ArcSDE commands. A command example is:

```
sdelayer -o register -e a3 -l holdiep,shape -c mslink -C USER -k SDO_GEOMETRY
         -u <username> -p <password>
```

An explanation of some of the parameters: a3 means that the geometries are 3D areas, MSLINK is the user-maintained ID column, HOLDIEP is the name of the table and SHAPE is the column that contains the geometry. The next step is defining a connection to Oracle from ArcCatalog (Database connections, Add Spatial Database Connection):

```
Server:   <blank>
Service:  sde:oracle
Database: <blank>
User:     <user_name>
Password: <password>@<database_name>
```

.

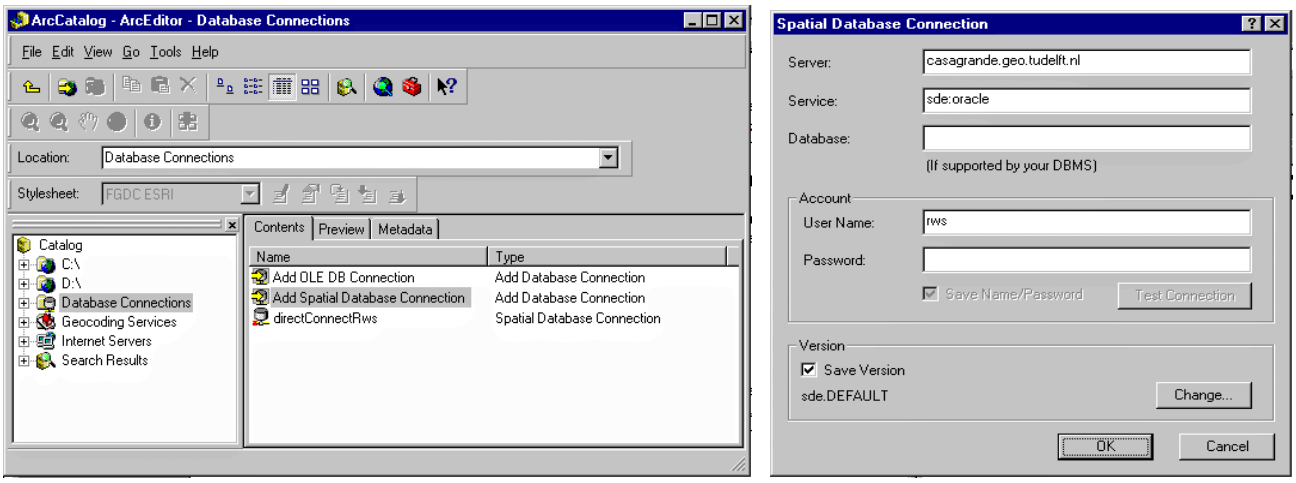


Figure 22: Add connection in ArcCatalog.

If the connection works the data can be visualized in all ArcGIS modules. Currently all ESRI products can access data stored in Oracle Spatial using direct connect. "Direct connect" makes a direct connection to the DBMS without using the ArcSDE application server. The user requires Oracle Net client software, which is not the case for connect via the ArcSDE service. The ESRI manual states that direct connect is easier to install and to maintain. Another advantage of direct connect is that an ArcSDE licence is not required for read-only access to the data (e.g. visualisation). However, if the user wants to edit the data, he/she will need a licence.

ArcGIS (connect via the ArcSDE service): The only differences to make a connection are in the parameters that must be specified for the connection: the server on which ArcSDE is running must be supplied, and the port number (or alias) the service is listening on. For both type of connections the ArcSDE metadata tables (that contain information about the geometry tables: dimension, spatial

data types, geometry column) must be available. If a layer is "unregistered" sometimes the Oracle metadata entries (in the USER_SDO_GEOM_METADATA view) are removed. This should not happen. The influence of a spatial index is also not clear. Without a spatial index the layer can be registered, however it cannot be visualised. A table cannot contain more than one geometry column as well as only one geometry type per column (views might be a solution). Invalid geometries, "vertical" polygons and multipolygon geometries cannot be visualised in ArcGIS. The Z coordinate is recognised by ArcGIS. The requirement that the ID column should be of type NUMBER(38) to avoid visualisation problems is difficult to find in the manuals.

6.2 MicroStation GeoGraphics (www.bentley.nl)

The first step is creating a project and categories. The project is created by specifying the name of the parent directory and the name of the project from Project->Setup. At least one category should be created. The following text and figures describe how this can be done. In the Project->Setup dialog box, all the text fields have to be filled in as they apply to the particular database and the database access string. The project has to be created and then the project has to be Open. In the Tables menu -> Feature Setup, a category has to be created and the corresponding fields (name, IndexFile, Level, Extension, category type) have to be filled in (Figure 23):

Name: Descriptive category name.
IndexFile: Name of the index file (index)
Level: Defines the level on which the index shape for the category resides (1)
Extension: Sets the extension for design files (idx)
Category Type: Spatial

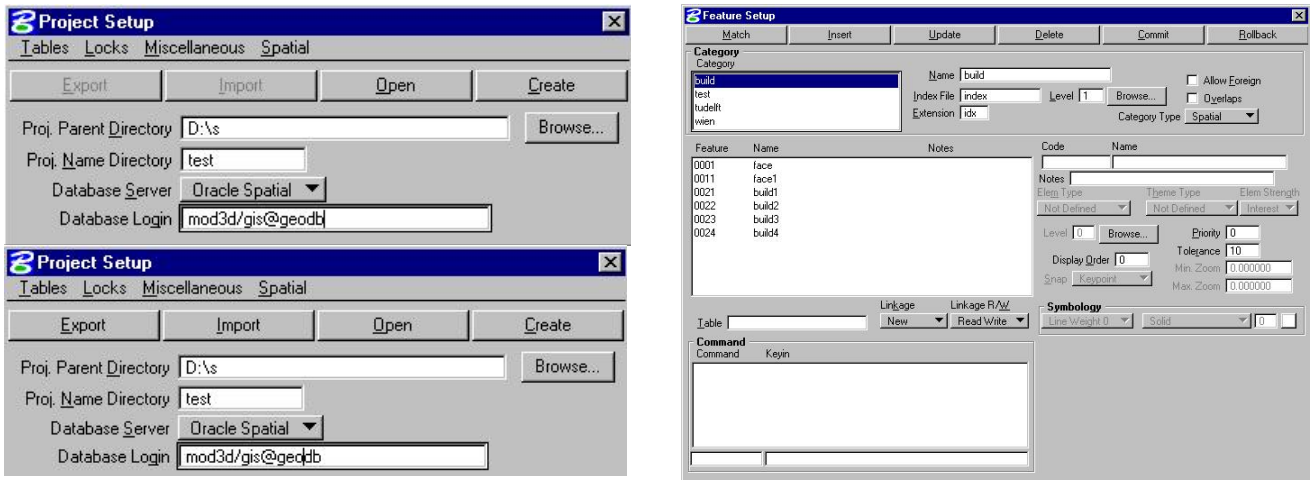


Figure 23: Creating a project in GeoGraphics.

The second step is to open the project (Project->Open) and query the data. In the Query tool, a click on the icon with a globe will activate the Project explorer. In the Project explorer all the layers, categories and features are visualised. The layer for visualisation has to be dragged to the Query tool window (Figure 24).

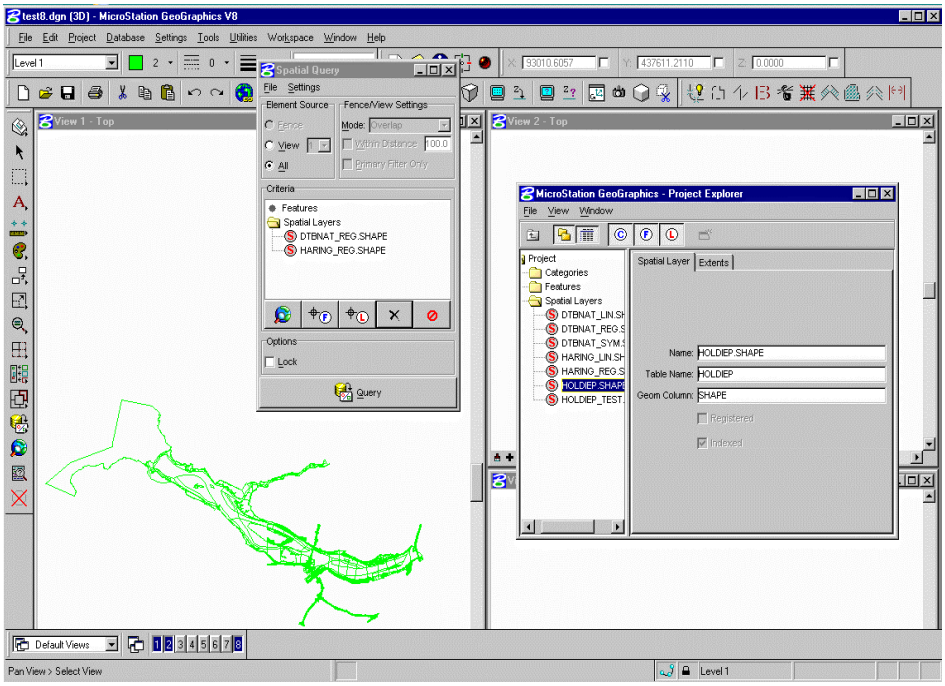


Figure 24: Querying and visualising data in GeoGraphics.

6.3 Virtual Reality Modelling Language (www.web3d.org/fs_technicalinfo.htm)

In the previous two sections, GIS and CAD software visualise the objects stored in Oracle Spatial. In this section, we show how to visualise the objects in a VRML file. VRML is a language to describe 3D models and to make them accessible on the Internet. Interaction and visualisation can be done by plug-ins or stand-alone programs (e.g. Cosmoplayer, Cortona). Since VRML is an open standard and can be used without licenses, it is quite appropriate for visualising of Oracle Spatial data by users that do not have a commercial front-end. There are two approaches that can be followed. It is possible to create only one VRML file, which contains both attributes and geometries, or to create one VRML file for geometries and one HTML for the attributes (e.g. codes, descriptions of objects, etc.). In the first case, the attributes become visible (as texts in the VRML browser) on "mouse-click" or "mouse-on" the object of interest. Figure 25 (left) shows an example of viewing the attributes of an object (a building, represented as a cube). The text becomes visible when the user places the cursor on the building. The disadvantage of this approach is that the file can become rather large (since the click operations have to be described inside the VRML file). This problem can be overcome by using separate HTML files for the attribute data. A VRML file is generated containing the geometries in the Oracle table. Every object in the VRML file is represented as an IndexedFaceSet and an *anchor* is attached to every object. An anchor is used to link an URL to an object. The anchor contains fields specifying the object. For every object, a single HTML file is generated containing the attributes of the object that are stored in the Oracle table. The VRML and HTML files are created on the fly when requested by the user. If one clicks on the object the corresponding URL (which indicates the specific HTML file) is opened in a frame defined in the parameter field of the anchor with the keyword (Figure 25, right).

Figure 25: Visualisation of Oracle Spatial data: one VRML file (left) and VRML and HTML files (right).

Since the Java program creating these files works with a topological representation of the objects, this program was not directly used for the data sets of RWS. We have adopted a simpler Java program that reads only the geometry columns. The Java program accesses a particular table (VRML) that contains only two columns MSLINK and SHAPE. Different queries can be executed at a database level and the resulting set of geometries can be stored in the table VRML. The Java program reads the contents of the VRML table and converts it to VRML. Any user can access the VRML file via the Internet (using a freeware VR Browser). For example, the following lines are part of a PL/SQL script that extracts all objects that are classified as *Oevers (*Figure 26*).*

```
...
select mslink, shape bulk collect into beh_objects,
shape1 from holdiep where substr(objectcode, 6, 1) = 'O';
select count(*) into numi from holdiep where
substr (objectcode, 6, 1) = 'O';
delete from vrml where mslink>0;
FOR I in 1..numi LOOP
  Insert into VRML (mslink,  shape) values (beh_objects(i), shape1(i));
END LOOP;
...
```

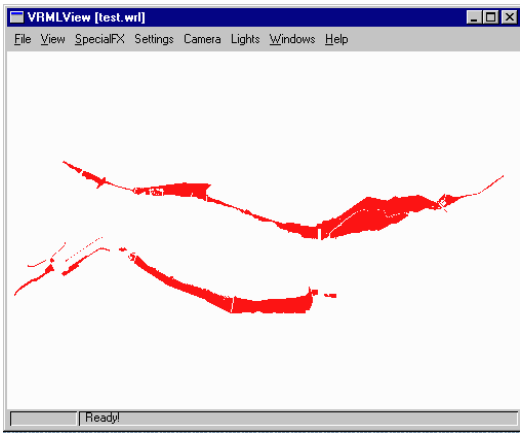


Figure 26: Visualising data in VRML: a part of the script (left) and the view (right).

More information about visualising objects stored in Oracle Spatial with the three front-ends can be found in Stoter and Zlatanova 2003, Zlatanova et al 2002, Arens 2003.

6.4 Selecting and editing data

ArcGIS: Editing of data using ArcGIS is possible but the layer has to be registered in a different way, this was not tested.

GeoGraphics: To be able to perform editing, settings as used for visualisation are not sufficient. Each geometric object has to be linked to a feature. The procedure is a bit complicated: the table with geometries has to be registered as a spatial layer in GeoGraphics. The features that are related to geometries from this layer have to be created and linked to the layer. In addition to these operations, which can be executed within GeoGraphics, there are some other settings that have to be completed in Oracle Spatial. Further details can be found in Zlatanova et al 2002.

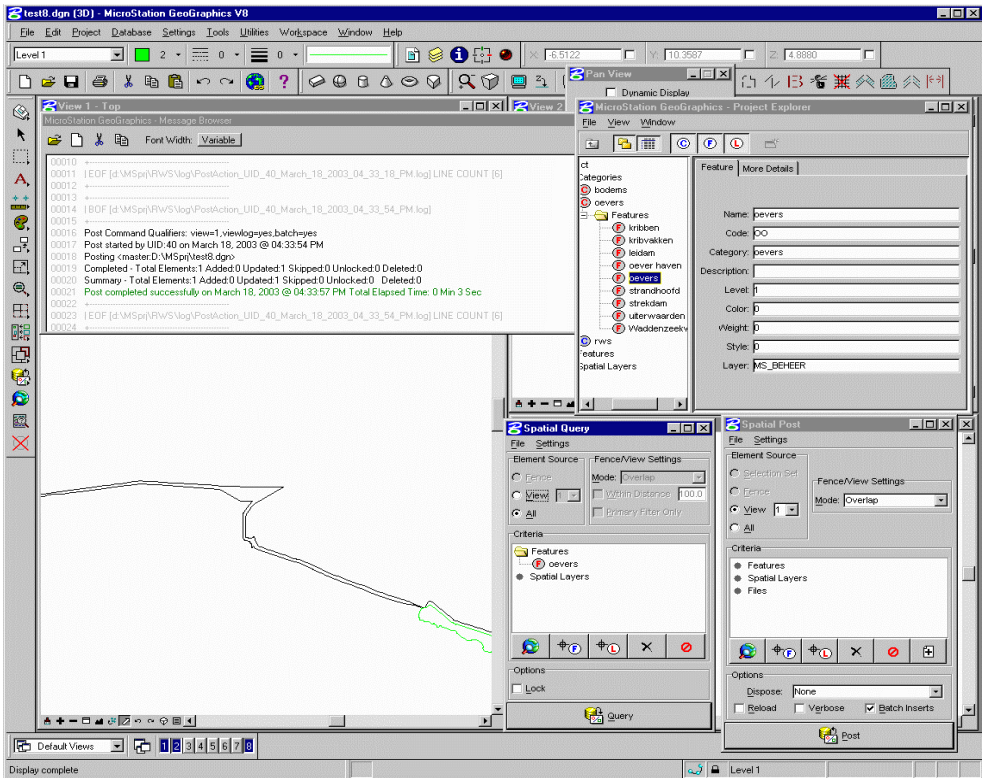


Figure 27: Querying, editing and posting objects in GeoGraphics.

A rather awkward problem surfaced when spatial data is first registered in ArcGIS and then in GeoGraphics. GeoGraphics remains working fine but ArcGIS reports an error and cannot visualise the geometries anymore. The problem is probably related to some settings or a conflicting configuration of all software packages involved. According to Bentley, this should not happen, i.e. it should be possible to use the same Oracle Spatial tables with different front-ends. The versions and the patches have to be carefully checked. Kees van Prooijen (Bentley NL) reports:

> *"We have successfully tested iSpatial interoperability by creating/posting elements (geometries) into Oracle Spatial and reading/analysing the data with GeoMedia. I am also aware that others have successfully tested interoperability with other packages such as MapInfo and Esri's ArcView and SDE. The spatial layers that are enabled for GeoGraphics can also be accessed properly by Oracle's spatial viewer"*

Since RWS does not use MicroStation and GeoGraphics, more experiments to localise the problem were not performed. In order to demonstrate how the objects can be edited and posted in Oracle Spatial, the content of the HOLDIEP table was copied in a new table MS_BEHEER and all the settings were performed on this table. Two object categories were introduced (*Bodems* and *Oevers*), and the corresponding sub-categories were defined as features in GeoGraphics. Figure 247 is an illustration how an object can be queried (with respect to the categories and the features), edited and posted back to the database. It should be noted that GeoGraphics adds new columns to the table that is registered as a spatial layer. For example, the MS_BEHEER table has the following structure (all the columns starting with MS are created by GeoGraphics):

```
Name                 Type
-------------------  --------------------
AREA                 NUMBER
PERIMETER            NUMBER
OBJECTCODE           VARCHAR2(20)
MSLINK               NUMBER(38)
SHAPE                MDSYS.SDO_GEOMETRY
MS_BEHEER_DFLAG      NUMBER(10)
MS_BEHEER_UDL        RAW(200)
MS_BEHEER_LOCK       NUMBER(10)
MS_BEHEER_FID        FCODE_LIST
```

```
MS_BEHEER_CREATED      DATE
MS_BEHEER_REVD         DATE
MS_BEHEER_RETIRED      DATE
MS_BEHEER_XML          VARCHAR2(1024)
MS_BEHEER_TXT          VARCHAR2(1024)
MS_BEHEER_STYLE        UG_STYLE
```

In the RWS data sets most of the objects are represented as simple 3D polygons that correspond to 3D "loose" polygons in MicroStation. The editing operations are thus restricted to the defined objects (in this case polygons and their vertices). For example, a shift of one vertex will change the vertex of the selected polygon. However, our experiments with other data sets (Stoter and Zlatanova 2003) have shown that the Oracle Spatial geometry type *collection* is interpreted by GeoGraphics as a group of polygons. To be able to edit such a group, it has to be "dropped" into the composing polygons. Before posting the object back to the database, the polygons have to be "grouped" again. Otherwise several new polygons will be posted and recorded as new objects in the database.

VRML: In general, VRML is a language for visualisation of and interaction with 3D models and therefore means for editing are not provided. However, in combination with HTML and Java or ECMA (JavaScript) scripts, some sort of editing can be simulated. For example, in HTML a list of coordinates can be delivered to the user together with the VRML file. This list can be edited and sent back to the database (Zlatanova 2000). However, such a graphics user interface requires certain programming efforts.

ArcGIS – GeoGraphics comparison: The advantage of ArcGIS compared to GeoGraphics is that the queries performed on the attributes of objects result in highlighting the corresponding geometries. GeoGraphics offers a SQL builder but the results of the query are in text form. For example, the result of the query "select shape from HOLDIEP where MSLINK = 100" will be a list of objects (the coordinates are not shown). Furthermore ArcGIS does not change the user tables. However, a table cannot contain more than one geometry column and only one geometry type per column is possible (eventually a solution for this is using different views). ArcGIS has severe problems with invalid geometries (geometries that do not validate in Oracle Spatial). It can happen that none of the geometries in the table can be visualised. In ArcGIS the ID column should be of type number(38) to avoid visualisation problems. GeoGraphics requires the column that will be referred as geometry ID to be named MSLINK. To be able to extract only a limited set of data (e.g. only *oevers* instead of the entire layer with all the Beheerobjects), the user has to create category/feature links to the geometry in GeoGrahpics. In ArcGIS such a limited set of data can be extracted by simple SQL statements.

7 OTHER ORACLE SPATIAL ISSUES

Oracle Spatial functionality: Chapter 3 has already discussed some issues related to the Oracle Spatial geometry model that is used for storing spatial data and the spatial functions that can be applied to them. Most of the functions correspond to the ones in the OpenGIS specifications (OGC 2003, ISO TC211 2003) although the names used in Oracle Spatial differ (Figure 28).

Topological operations

OGC	Oracle
Equals	EQUAL
Disjoint	DISJOINT
Intersects	ANYINTERACT
Touches	TOUCH
Crosses	OVERLAPBDYDISJOINT
Within	INSIDE
Contains	CONTAINS
Overlaps	OVERLAPBDYINTERSECT

In Oracle these relationships are specified by means of the SDO_RELATE spatial operator

Other operations

OGC	Oracle
Distance	SDO_DISTANCE
Convexhull	SDO_CONVEXHULL
Intersection	SDO_INTERSECTION
Union	SDO_UNION
Difference	SDO_DIFFERENCE
Symdifference	SDO_XOR
Area	SDO_AREA
Buffer	SDO_BUFFER
Centroid	SDO_GEOMCENTROID
Length	SDO_LENGTH
Boundary	SDO_MBR
Within distance	SDO_WITHIN_DISTANCE

Figure 28: Comparison between Oracle Spatial functions and the OGC specification.

In Oracle Spatial a distinction is made between operators (require a spatial index) and functions (do not require an index). Some of the operators/functions are briefly explained.

Spatial Operators

Spatial operators require and use the spatial index for efficient operation, an operator can be applied to a complete layer (column).

SDO_FILTER	detects whether objects are likely to interact
SDO_NN	identifies nearest neighbour
SDO_NN_DISTANCE	distance to the nearest neighbour
SDO_RELATE	relationships according to the 9-intersection model
SDO_WITHIN_DISTANCE	all the objects within a given distance

Geometry Functions

Geometry functions do not use the index, they operate on individual (pairs of) geometries.

SDO_GEOM.RELATE	relationships according to the 9-intersection model
SDO_GEOM.SDO_ARC_DENSIFY	returns circular arcs converted into lines
SDO_GEOM.SDO_AREA	returns area
SDO_GEOM.SDO_BUFFER	returns buffer
SDO_GEOM.SDO_CENTROID	
SDO_GEOM.SDO_CONVEXHULL	
SDO_GEOM.SDO_DIFFERENCE	
SDO_GEOM.SDO_DISTANCE	
SDO_GEOM.SDO_INTERSECTION	returns intersection of two areas
SDO_GEOM.SDO_LENGTH	returns length of linestring or polygon boundary
SDO_GEOM.SDO_MAX_MBR_ORDINATE	
SDO_GEOM.SDO_MBR	returns minimum bounding rectangle
SDO_GEOM.SDO_MIN_MBR_ORDINATE	
SDO_GEOM.SDO_POINTONSURFACE	returns a point that is guaranteed to lie on the surface
SDO_GEOM.SDO_UNION	
SDO_GEOM.SDO_XOR	returns a polygon that is the symmetric difference of the geometries
SDO_GEOM.VALIDATE_GEOMETRY	
SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT	validates geometries and indicates where problem is
SDO_GEOM.VALIDATE_LAYER	
SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT	
SDO_GEOM.WITHIN_DISTANCE	

Aggregate functions

Another class of functions returns an aggregate of a collection of geometries. The following aggregate functions are available in Oracle:

SDO_AGGR_CENTROID	returns a geometry that is the "center of gravity"
SDO_AGGR_CONVEXHULL	returns a geometry object that is the convex hull
SDO_AGGR_LRS_CONCAT	
SDO_AGGR_MBR	returns minimum bounding rectangle
SDO_AGGR_UNION	returns geometry that is the topological union (OR operation)

Coordinate System Transformation Functions

SDO_CS.TRANSFORM	transforms geometry representation
SDO_CS.TRANSFORM_LAYER	transforms entire column with geometries
SDO_CS.VIEWPORT_TRANSFORM	transforms a rectangle into a polygon

Information about all the operators and functions supported by Oracle spatial can be found on websites with online Oracle 9i Documentation, e.g:

http://download-east.oracle.com/docs/cd/B10500_01/appdev.920/a96630/toc.htm

The available operations can be used in any regular SQL command such as "create table", "select" etc. The following SQL statement creates a table named new_geometry in which information is stored about parcels (from a table 'parcel') that intersect with a line (abstraction of a tunnel) stored in the table 'tunnel'. In this particular example the function SDO_INTERSECTION is used.

```
create table new_geometry as select
    object_id,
    parcel_number,
    sdo_geom.sdo_intersection (t.shape, p.shape, 10) shape
 from parcel p, tunnel t;
```

Consequently the functions can be run from any front-end that has a facility for specifying queries (e.g. Select by Attribute in ArcMap or SQL builder in GeoGraphics). Figure 29 illustrates utilisation of such a function (e.g. SDO_AREA) in ArcMAP.

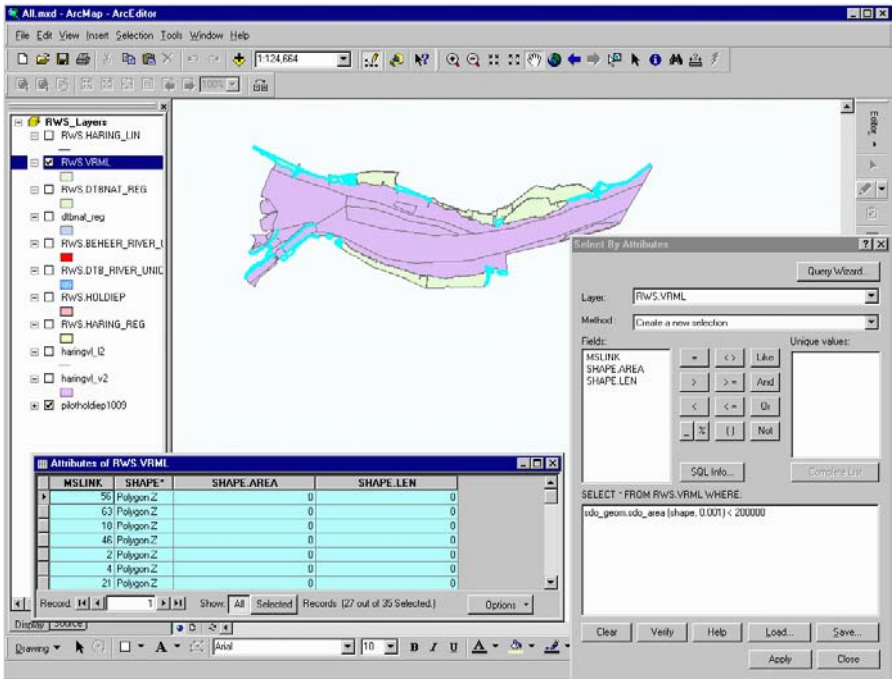


Figure 29: Selection of objects with a particular area.

As mentioned before, GeoGraphics currently does not show the geometries but only text, i.e. IDs of objects and other text data if available. If more elaborate operations are needed, which cannot be performed with a single SQL statement (e.g. using iterations, Figure 26 and Appendix 3), a programming language is required (PL/SQL, Java, C).

3D spatial data: Oracle Spatial geometry can contain 3D coordinates. The Z value is maintained together with the X,Y values, i.e. it is not an attribute. This is to say that maintenance of 2.5D and 3D objects is possible but the analysis is only 2D:

- Functions and operators do not use the Z coordinate. For example, the area of a polygon with coordinates (0,0,0; 100,0,0; 100,100,100; 0,100,100) will be computed as 0.
- 3D volume objects can be represented, as a set of 3D polygons (multipolygon or collection), but the validity (closed volume, non-intersecting faces, correct orientation, etc.) of such objects cannot be checked. Moreover in case of a complex 3D volume object, the size of the SDO_ORDINATES can grow significantly. Note that each triplet of coordinates is repeated at least three times. Stoter and Oosterom 2001 propose new values of SDO_GEOMETRY elements to describe 3D objects (e.g. tetrahedron, polyhedron, polyhedron with holes, etc.). The SDO_ORDINATES array is suggested to have two sections, i.e. a list of coordinates and references to the list. This approach will reduce the size of the array considerably, which is a critical consideration in maintaining 3D data. Arens 2003 develops a validation function and a number of operators to compute area, distance, overlapping, etc.

All these issues are under consideration and hopefully they will be implemented one day.

GML representation of spatial data: With the ever increasing use and possibilities of the Internet some reflections on the distribution and exchange of spatial data are in order. In the current situation GML is a logical candidate for any type of spatial data exchange: it is flexible, powerful, open, non-proprietary, extensible, etc. Many of its positive characteristics result from the fact that GML derives from XML. And the flexibility and extensibility of XML are nice from a modelling perspective, but it does provide problems for application software. GML can be applied in three different ways:

- as an application-specific format;
- based on a limited subset of all available GML elements and XML functionality;
- as a completely flexible and generic format for spatial data.

Accompanying a GML document there should be an XML Schema document that describes the structure of the data (hierarchy, dependencies), defined feature classes with attributes, data types, values allowed, etc. Reading or writing GML as a generic format implies interpretation and understanding of the underlying schema, quite a daunting task if you look at the extensive and complex GML 3 specification. But even the much more limited GML 2 specification is difficult to support if the full flexibility inherent in XML is allowed (not impossible, XML parsers can be used to 'understand' Schema documents). GIS vendors are currently starting to support GML 2, as an application-specific format or based on a limited subset ('profile') of GML 2.

The Oracle DBMS has built-in support for XML. An XML data type exists and XML documents can be read, parsed, stored and written. Just a few lines of specifications will suffice to transform a table containing a geometry column into an XML document. This will not be a valid GML document, the structure and the 'tags' will differ from the GML specification (because in this transformation the GML Schema documents, that define GML, are not used). It is also not difficult to create a program to produce an application-specific GML document from a table in a DBMS: the GML formatting and application schema will be 'hard-coded' in the program. But more generic solutions, which always means understanding Schema documents and applying the definitions, rules, restrictions, etc. specified in those schemas, will require more than a straightforward program that can be produced in a few days.

Oracle compared to other DBMSs: We have not used SQL server in our research but we have already performed a number of tests with several databases. The comparison between different DBMSs shows benefits for different systems with respect to different criteria. For example, regarding 3D functions, PostGIS (PostgreSQL) and the Mapinfo Spatialware Datablade (IBM/Informix) do support geometry calculations such as length and perimeter in 3D. PostGIS also has a box3D function that gives the maximum extents in 3D as a result. Many other mainstream DBMSs (like Informix, IBM DB2) support a set of geometry operators similar to Oracle Spatial, but most of them also skip the Z coordinate in the operations provided.

Performance is another critical issue for real data sets. The section GISt has tested three different DBMSs, Oracle, Informix and Ingres, with respect to loading and spatial queries. The Dutch cadastre selected sample questions from their experience. These questions, translated into SQL queries, were used to test all databases. All operations in the experiments focused on data query, i.e. update operations are not considered. Some of the queries in the test set are:

- Find and display all objects within a certain rectangular area of the map. This a common query that is completed by looking at the Minimum Bounding Rectangles (MBR) of all the objects and thus ignoring the actual shape of the objects.
- Find all objects that overlap with a given polygon (the polygon can be irregular). This is a typical overlay query between the polygon and all the objects in the database. If the query polygon consists of many points, the operation can become time-consuming.
- Find and clip all objects within a given search polygon. In contrast to the previous query, this one requires clipping of objects.

A substantial amount of spatial data (more than 121,000,000 records) was loaded into the three DBMS products without encountering any real problems. Loading spatial data is still more elaborate than loading alphanumeric data. This is because the loading tools are sometimes not able to work with the spatial types, or the creation of indexes needs manual adjusting of parameters. The performance of windowing queries is satisfactory for all DBMSs. The biggest difference in response time was observed in the overlap queries (especially overlaps with a relatively long line). Informix performed best, with Oracle second best and Ingres lagging behind somewhat. Further details related to the data, queries, and performance can be found in Quak et al 2002. Shortly, the performance of the geo-DBMS still requires improvements. Looking back in history it can be observed that performance has improved with every new release.

Separate from the choice of the DBMS the possibility exists to use middleware to take care of the spatial data. In that case everything is still stored in the DBMS, but the spatial data can only be accessed through the middleware because it is stored in a middleware specific format (e.g. ArcSDE SDELOB). Normally this provides a certain advantage in speed, but using the 'native' spatial data types of the DBMS brings advantages in several directions:

- Interoperability, i.e. possibility to access the data by different front-ends.
- If the DBMS takes care of consistency checks, data reliability will be increased when using different front-ends.
- Strengthening of multi-user access and editing of data.
- Possibilities for distributed maintenance and management.

Topology structure management is not supported by the current version of Oracle Spatial. However, some support for topology structure management has been announced. We have performed a lot of experiments with topological models in user-defined relational tables. We have used two types of representations, (1) with coordinates stored in user-defined nodes and edges, and (2) utilising the Oracle Spatial geometry model. Obviously the general drawback of the first case is that all the functions and operators on the topologies have to be developed by the user. Furthermore different topologies can be used in case of 2D or 3D types of data. As part of the research in the section GISt we have experimented with several different data structures, i.e. 2D (winged-edge, wheel-chain) and 3D (Simplified Spatial Structure and modifications of it). More details on pure topological models implemented in Oracle relational tables can be found in

Zlatanova and Verbree 2000. Oosterom et al 2002 suggests maintaining different types of topological structures in one database by specifying the characteristics of each topology in a metadata relational table and providing functions that can ensure consistent conversion between the topologies and the geometry. The second approach uses the geometry types of Oracle Spatial but additional information about the topology is also stored. For example, the 3D cadastral objects in Figure 25 can be represented as a list of polygons. Thus a 3D object is a list of polygon IDs. The polygons are represented by the geometry types of Oracle Spatial (3D polygon). Using such an approach, neighbourhood operations in 3D can be easily performed by comparing common faces. Furthermore, all the indexing techniques and functions of Oracle Spatial can be utilised.

Currently, a relatively mature product to manage topology in Oracle is Radius Topology (Laser-Scan). Another solution is waiting for the topology management probably offered by Oracle in the next release.

Distributed databases: The text below is compiled from Oracle specifications.

In principle, a distributed database is a network of databases managed by multiple database servers that are used together. They are usually seen as a single logical database. The data of all databases in the distributed database can be simultaneously accessed and modified. The primary benefit of a distributed database is that the data of physically separate databases can be logically combined and potentially made accessible to all users on a network.

A distributed database enables increased access to a large amount of data across a network and at the same time hides the location of the data and the complexity of accessing it across the network. The distributed database management system should also preserve the advantages of administrating each local database as though it was not distributed.

Two types of distributed systems are available in Oracle, i.e. homogeneous and heterogeneous. A homogenous distributed database system is a network of two or more Oracle databases that reside on one or more machines. An application can simultaneously access or modify the data in several databases in a single distributed environment.

In a heterogeneous distributed database system, at least one of the databases is a non-Oracle system. To the application, the heterogeneous distributed database system appears as a single, local, Oracle database. The local Oracle database server hides the distribution and heterogeneity of the data. The Oracle database server accesses the non-Oracle system using Oracle Heterogeneous Services in conjunction with an agent. If one accesses the non-Oracle data store using an Oracle Transparent Gateway, then the agent is a system-specific application. For example, if one includes a Sybase database in an Oracle distributed system, and then one needs to obtain a Sybase-specific transparent gateway so that the Oracle databases in the system can communicate with it.

8 CONCLUSIONS AND RECOMMENDATIONS

We have designed, implemented and executed a number of Oracle Spatial tests with data sets provided by RWS, i.e. DTB-Nat, Regiokaart-Nat and Beheerkaart-Nat. The tests have demonstrated that these data sets can be stored in Oracle DBMS using the currently supported geometry types. The functionality provided by Oracle Spatial allows a large number of operations to be performed on the spatial data. In this respect, the major goal of this research can be considered achieved. The finding of the tests can be summarised as follows.

There are tools (e.g. FME) that allow automatic conversion of shape files into Oracle Spatial tables (including the definition of tables). Oracle Spatial also offers a loader, which can import data in predefined tables. All the data of the six data sets were loaded into the Oracle Spatial model without significant problems. Oracle Spatial maintains spatial indexing for accelerating the operations on spatial objects. In the tests we have used 2D R-tree indexes.

The geometry types available in Oracle Spatial are sufficient for the representation of all types of spatial objects of interest to RWS (3D points, 3D lines and 3D polygons). A real 3D object can be represented as well. However, the validity of 3D objects cannot be checked with standard functions of Oracle Spatial. 2D geometry can be checked for validity and consistency without problems. The tests revealed a number of objects (polygons) in DTB-Nat with invalid geometries or overlap where this should not occur.

Once stored in the same database and the same geometry model, the objects of the three maps can be checked and compared against a variety of geometric and semantic conditions. We have performed overlap tests between the three maps with polygons to investigate whether it is possible to match the objects. On the basis of these tests we can make the following recommendations:

- The maintenance of Regiokaart-Nat as a separate data set should be avoided. If the Beheerkaart-Nat exists for the whole territory of the Netherlands, the Regiokaart-Nat can be derived from it (applying appropriate generalisation on the fly). The tests have also shown that the match between Regiokaart-Nat and Beheerkaart-Nat is more than 80% (the remaining 20% are mostly coding errors). This is a clear indication that the contents and use of the Regiokaart-Nat has to be reconsidered.
- A link between the geometries of the Beheerkaart-Nat and DTB-Nat can be established as well. Such a link can be organised in a separate table, which can maintain also a variety of rules for object editing. For example, one cannot change the geometry of a DTB-Nat object without changing the geometry of the same object in the Beheerkaart-Nat.
- The Beheerkaart-Nat eventually can be obtained from the objects from the DTB-Nat for the areas where the Regiokaart-Nat is available. Although this is not a lot, it might facilitate the creation of the Beheerkaart-Nat.
- Many DTB-Nat objects that are in reality one object (but digitised from different stereo-pairs in InfoCam) can be automatically aggregated utilising the established link between Beheerobjects and DTBobjects (e.g. on the basis of 'WaterSysteemDeel').

We have proposed an integrated model that consists of only the Beheerkaart-Nat and DTB-Nat. The model is built up based on the CTE categories and the classification introduced in TISBO. The integrated model consists of three basic tables, DTB-Nat, Beheerkaart-Nat and a table maintaining the link between the two. In Oracle Spatial it is not necessary to store different types of geometries in different tables. The link established between the objects (in this research) is only based on geometry. Considering the classification of the objects, it is wise to establish a link (and restrict) the objects with respect to their semantics.

The integrated model makes spatial cross-queries between the two maps relatively easy and straightforward. For example, "find all the DTBObjects in a given BeheerObject" or "find DTBObjects that correspond to a given TISBO code within a distance of 50 km from a supplied location", etc.

We have demonstrated that the objects stored in Oracle Spatial can be accessed from different front-ends. The problems encountered are mostly results of version incompatibility, which will be resolved in new software releases.

Topology is important to maintain a high level of data quality. Many of the errors discovered in the data sets (invalid geometries, overlapping objects, non-complete coverage) can be easily avoided. Topologically structured objects have many advantages:

- it avoids redundant storage (more compact than storing full geometry).
- it is easier to maintain consistency of the data after editing.
- it is the natural data model for certain applications (e.g. during surveying boundaries are measured, together with relevant attributes).
- it is efficient for certain visualisations or query operations (e.g. find neighbours).

Therefore, we firmly recommend organising the RWS data in a topological structure.

The tests and discussions with consultants from RWS have revealed that further research is needed in several directions:

- Tests related to access, visualisation and editing of data in the proposed model in combination with the AutoDesk products used in RWS.
- The current research did not investigate the organisation of points (and eventually lines, if RWS concludes that they are still needed). In general storage efficiency and query performance of points is relatively better in Oracle Spatial compared to lines and areas (and other systems).
- Further tests are needed with editing data in ArcGIS (direct connect and via ArcSDE) and posting the changes to Oracle Spatial.
- Automatic generalisation of the Regiokaart-Nat on the basis of the DTB-Nat. A number of issues have to be investigated: selection of objects, removal of points, joining of points, etc.
- Investigation of an appropriate topological structure for RWS, e.g. Laser-Scan Radius Topology vs. Oracle Spatial Topology Manager (release 10) vs. ESRI topology.
- Maintenance, visualisation and editing of 3D objects (e.g. buildings).

References

Asperen, P.C.M. van and P.R. Bannier, 2002, Ontwikkeling van de Beheer-en Regiokaart-Nat, GeoNieuws, Nr. 3, 2002, pp. 9-11.

Arens, C.A. 2003, Maintaining Reality, Modelling 3D spatial objects in a Geo-DBMS using a 3D primitive, MSc thesis, TUDelft, 65 p.

Boeringa, M., 2002, Handleiding DTB legenda's voor ArcView 3.x en ArcGIS, Ministerie van Verkeer en Waterstaat, MD, Afdeling TGFS, 31 p.

Brink, M.B van de, M.J. Dijkman-den Hollander, W.P Hoogenboom and M.D Taal, 2002, Wegwijzer Beheerplan –Nat, DWW - 2002 – 019, ISBN 90-369-3787-6, pp. 154.

Gebruikershandleiding DTB2000, DTB-NAT & DTB-DROOG, CONCEPT, pp. 41.

ISO TC211, 2003, Geometry model, available at: http://www.isotc211.org/.

Klatter, H.E and P. Padding, 1998, Pilot Uitgebreide BOPPER-Kaart (Pilot UBK), Projectrapportage,MD, RWS, MDGAG-9836, 17 p.

Looman, P.A.M. and P.C.A. Wouters, 2002, Productspecificatie DTB-Nat, MD< RWS, MD-TGM-2002-25, 21 p.

Meijer, P., J. Wäkerlin and R. Dood, 2001, Advies GIS-component doorontwikkeling TISBO, RWS, MD, GAG 2001.02.

OpenGIS Consortium, 2003, OpenGIS Implementation Specifications, available at: http://www.opengis.org/techno/implementation.htm.

Oosterom, P.J.M. van, C.W. Quak and T.P.M. Tijssen, 2001, Testing current DBMS products with real spatial data, In: E.M. Fendel, K. Jones, R. Laurini and M. Rumor (eds.), Proceedings of UDMS '02 23rd Urban Data Management Symposium, '30 Years of UDMS, Looking Back, Looking Forward (Prague, Czech Republic, 1-4 October 2002), Prague 2002, p. VII.1-VII.18.

Oosterom, P.J.M. van, J. Stoter, C.W. Quak and S. Zlatanova, 2002, The balance between geometry and topology, in: Advances in Spatial Data Handling, 10th International Symposium on Spatial Data Handling, D.Richardson and P.van Oosterom (Eds.), Springer-Verlag, Berlin, pp. 209-224.

Oracle 9i Spatial, 2003, Oracle Online Documentation, Spatial User's Guide and References available at: http://otn.oracle.com/documentation/content.html.

Radius Topology, Laser Scan, 2003, available at: http://www.radius.laser-scan.com/.

Stoter, J.E. and P.J.M. van Oosterom, 2001, Incorporating 3D geo-objects into an existing 2D geo-database: an efficient use of geo-data, In: Proceedings Geoinformatics and DMGIS' 2001, the 3rd ISPRS Workshop on Dynamic & Multi-dimensional GIS, the 10th Annual Conference of CPGIS on Geoinformatics, Asian Institute of Technology, Bangkok, Thailand, 23-25 May 2001.

Stoter, J. and S. Zlatanova, 2003, Visualisation and editing of 3D objects organised in a DBMS, Proceedings of the EuroSDR Com 5. Workshop on Visualisation and Rendering, 22-24 January, Enschede, the Netherlands, 16 p.

Wybenga, B.I. and E. Koster, 2002, Informatie-analyse BNP-proces: Op weg naar uniformiteit, Ministerie van Verkeer en Waterstaat, Directoraat-Generaal Rijkswaterstaat, MD, 63 p.

Tjissen, T.P., C. W. Quak and P.J.M. van Oosterom, 2001, Spatial DBMS testing with data from the Cadastre and TNO-NITG, GISt Report 7, available at: http://www.geo.tudelft.nl/gist/ (Publications/reports, 2001).

Uitermark, H., 2001, Ontology-based geographic data set integration, PhD thesis Enschede, ISBN 90-365-1617-X

Vries, M.E. de, T.P.M. Tijssen, J.E. Stoter, C.W. Quak and P.J.M. van Oosterom, 2002, The second GML prototype of the new TOP10vector object model, TU Delft/Geodesie GISt report no. 12.

Zlatanova, S. 2000, 3D GIS for urban development, PhD thesis, 2002, Graz University of Technology, ITC publication number 69, 222 p.

Zlatanova, S., 2002, Detailed Project Plan Research Project on the usability of Oracle Spatial within the RWS Organisation (MD-Nr. 3215), Rapport aan: MD, Uitgave TU/Geodesie, Delft, 13 p. ISSN 1569-0245, Rapportnummer: GISt no. 15.

Zlatanova, S. A.A. Rahman and M.Pilouk, 2002, 3D GIS: current status and perspectives, In: Proceedings of the Joint Conference on Geo-spatial theory, Processing and Applications, 8-12 July, Ottawa, Canada, 6 p. on CDROM.

Zlatanova S. and E. Verbree, 2000, A 3D topological structure management for augmented reality, in: Proceedings of the Second International Symposium on Mobile Multimedia Systems & Applications, 9-10 November, Delft, the Netherlands, pp. 19-26.

Zwijnenburg, J. 2001, Handleiding TISBO-Nat, Handleiding Nat release 4.2, RWS, 80 p.

Appendix 1: FME mapping file of Beheerkaart-Nat

```
# ===================================================================
#
# This mapping file was generated by the FME
# on 01/20/03 14:32:05 for translation between SHAPE and ORACLE8I
#
# This mapping file was generated with FME build 895
#
# You may edit this mapping file to customize its operation.  Comments are
# placed throughout to assist you.
#
# Modification History:
#
#     Name              Date     Description
#     ================ ======== ============================================
#
#
# ===================================================================

# ===================================================================
# The following line defines the title presented to the user when this
# mapping file is run through the FME GUI.  You may modify this
# if a more meaningful title would be appropriate.

GUI TITLE SHAPE to ORACLE9I Translation

# ===================================================================
# The following line names the log file to which useful statistics about
# the translation will be written.  This line can be uncommented and
# updated if you do wish to keep these statistics.

LOG_FILENAME holdarea2ora.log
# LOG_APPEND NO

# ===================================================================
# The following line instructs the FME to log any features that do not
# match any of the source feature patterns listed further down in
# this file.  If you are modifying this mapping file, this will be
# useful to describe to you exactly which features you are losing
# during translation, if the statistics indicate that features are
# not being correlated or grouped.  Uncorrelated features do not
# match any source specification, ungrouped features do not have
# any corresponding _DEF line.

# FME_DEBUG UNGROUPED UNCORRELATED

# ===================================================================
# The following two lines define the type of reader and writer to be
# used for this translation.  If you want to translate your data
# back into its original format, you may make a copy of this file
# and switch the reader and writer types.  If you rerun the FME, you
# will get your original data back again (together with any modifications
# you made in the meantime).  Note that several formats are NOT
# bi-directional (for example, GIF can only be used as a WRITER)
# so a reverse translation may not always be possible.

READER_TYPE SHAPE

WRITER_TYPE ORACLE8I

# ===================================================================
# The following GUI line prompts for a directory to be used as the
# source of the ESRI SHAPE files.
# The user input is stored in a macro, which is then used to define
# the dataset to be read.

# The dataset this mapping file was generated from was:
DEFAULT_MACRO SourceDataset D:\Data\Rws\pilotholdiep

GUI DIRNAME SourceDataset Original Shape File Directory:

SHAPE_DATASET "$(SourceDataset)"

# ===================================================================
# The following GUI line prompts for the name of the Oracle Service
# to which data will be written.

GUI TEXT DestDataset Destination Oracle Service:

ORACLE8I_TRANSACTION 0
```

```
# ============================================================================
# These next prompts ask for various layer creation parameters. Normally
# these will be hardcoded on a layer by layer basis
#

DEFAULT_MACRO _ORACLE_Dimension 3
DEFAULT_MACRO _ORACLE_Minz 0
DEFAULT_MACRO _ORACLE_Maxz 0
DEFAULT_MACRO _ORACLE_Tolx 0.0000000005
DEFAULT_MACRO _ORACLE_Toly 0.0000000005
DEFAULT_MACRO _ORACLE_Tolz 0.0000000005
DEFAULT_MACRO _ORACLE_CreationParams ""
DEFAULT_MACRO _ORACLE_SRID ""

GUI OPTIONAL TEXT _ORACLE_CreationParams Additional table creation parameters:

GUI INTEGER _ORACLE_Dimension Geometric dimension:

GUI OPTIONAL INTEGER _ORACLE_SRID Spatial referencing (SRID) value:

GUI FLOAT _ORACLE_Minx Minimum X Coordinate:

GUI FLOAT _ORACLE_Miny Minimum Y Coordinate:

GUI FLOAT _ORACLE_Minz Minimum Z Coordinate:

GUI FLOAT _ORACLE_Maxx Maximum X Coordinate:

GUI FLOAT _ORACLE_Maxy Maximum Y Coordinate:

GUI FLOAT _ORACLE_Maxz Maximum Z Coordinate:

GUI INTEGER _ORACLE_Levels Tesselation Levels:

GUI FLOAT _ORACLE_Tolx Comparison tolerance for X coordinate:

GUI FLOAT _ORACLE_Toly Comparison tolerance for Y coordinate:

GUI FLOAT _ORACLE_Tolz Comparison tolerance for Z coordinate:

GUI INTEGER _ORACLE_Levels Tesselation Levels:

GUI CHOICE _ORACLE_Indicies Yes%No Create Indices:

DEFAULT_MACRO _ORACLE_IndexCommitInterval 50
# GUI INTEGER _ORACLE_IndexCommitInterval Incremental index commit interval:

DEFAULT_MACRO _ORACLE_NumTiles 8
GUI INTEGER _ORACLE_NumTiles Hybrid index numtiles (0forfixedindex):

DEFAULT_MACRO _ORACLE_GeometryColumn SHAPE

ORACLE8I_SERVER_TYPE ORACLE8i

# ============================================================================
# The following GUI lines prompt for the user name and password to use for
# accessing Oracle Spatial

GUI TEXT _ORACLE_UserName Oracle User ID:

ORACLE8I_USER_NAME "$(_ORACLE_UserName)"

GUI PASSWORD _ORACLE_Password Oracle Password:

ORACLE8I_PASSWORD "$(_ORACLE_Password)"


ORACLE8I_DATASET "$(DestDataset)"

# ============================================================================
# The main body of the mapping file starts here.  Each of the
# _DEF lines describes the data model of the particular feature
# type, and the correlation lines describe how the feature is
# transformed from the source type to the destination type.
# You may edit the following lines to add or remove attributes, change
# attribute definitions, or invoke other FME functions as the
# features are translated.
# ============================================================================

# ============================================================================
# Source feature type definition section
```

```
# ========================================================================

SHAPE_DEF pilotholdiep1009                      \
    SHAPE_GEOMETRY       shape_polygon          \
    AREA                 number(16,3)           \
    PERIMETER            number(16,3)           \
    HOLDIEP6_            number(11,0)           \
    HOLDIEP6_I           number(11,0)           \
    OBJECTCODE           char(20)               \
    MSLINK               number(20,0)

# ========================================================================
# Transformation section
# ========================================================================

SHAPE pilotholdiep1009                          \
    AREA                 %AREA                  \
    PERIMETER            %PERIMETER             \
    HOLDIEP6_            %HOLDIEP6_             \
    HOLDIEP6_I           %HOLDIEP6_I            \
    OBJECTCODE           %OBJECTCODE            \
    MSLINK               %MSLINK

ORACLE8I HOLDIEP                                \
    oracle_type          oracle_area            \
    AREA                 %AREA                  \
    PERIMETER            %PERIMETER             \
    HOLDIEP6_            %HOLDIEP6_             \
    HOLDIEP6_I           %HOLDIEP6_I            \
    OBJECTCODE           %OBJECTCODE            \
    MSLINK               %MSLINK

# ========================================================================
# Destination feature type definition section
# ========================================================================

ORACLE8I_DEF HOLDIEP                            \
    oracle_model         object                 \
    oracle_create_indices $(_ORACLE_Indicies)   \
    oracle_params        $(_ORACLE_CreationParams) \
    oracle_srid          $(_ORACLE_SRID)        \
    oracle_levels        $(_ORACLE_Levels)      \
    oracle_numtiles      $(_ORACLE_NumTiles)    \
    oracle_min_x         $(_ORACLE_Minx)        \
    oracle_min_y         $(_ORACLE_Miny)        \
    oracle_min_z         $(_ORACLE_Minz)        \
    oracle_max_x         $(_ORACLE_Maxx)        \
    oracle_max_y         $(_ORACLE_Maxy)        \
    oracle_max_z         $(_ORACLE_Maxz)        \
    oracle_x_tol         $(_ORACLE_Tolx)        \
    oracle_y_tol         $(_ORACLE_Toly)        \
    oracle_z_tol         $(_ORACLE_Tolz)        \
    oracle_dim           $(_ORACLE_Dimension)   \
    oracle_default_geom_column $(_ORACLE_GeometryColumn) \
    AREA                 number                 \
    PERIMETER            number                 \
    HOLDIEP6_            number(12)             \
    HOLDIEP6_I           number(12)             \
    OBJECTCODE           varchar2(20)           \
    MSLINK               integer
```

Appendix 2: Spatial index creation example (DTBNAT_REG)

```
drop index IDTBNAT_REG force;
delete from user_sdo_geom_metadata where table_name = 'DTBNAT_REG';

insert into user_sdo_geom_metadata values(
    'DTBNAT_REG',
    'SHAPE',
     mdsys.sdo_dim_array(
         mdsys.sdo_dim_element('X', 44000, 104000, .0001),
         mdsys.sdo_dim_element('Y',404000, 437000, .0001),
         mdsys.sdo_dim_element('Z',   -10,   1000, .0001)),
     NULL);

create index IDTBNAT_REG on DTBNAT_REG (shape)
    indextype is mdsys.spatial_index
    parameters ('sdo_fanout=35 sdo_indx_dims=2 tablespace=indx');

call analyze_rtree ('IDTBNAT_REG');
execute sdo_tune.analyze_rtree ('RWS','IDTBNAT_SYM');
```

Appendix 3: PL/SQL Script to test for polygon overlap in DTB-Nat

```
#!/bin/sh
#          selfovrlap.sh  12-02-2003  TT
#
# Script to test if geometries in a layer overlap

BASETABLE=${1:-'dtbnat_reg'}
  ORA_USR=${2:-'rws'}

  ORA_PWD=${ORA_USR}
GEOM_ATTR='shape'
 OID_ATTR='mslink'
TOLERANCE='0.000001'

cat > selfovrlap.sql << EOB
--============================================================================
set echo on
set lines 120
set pages 500
set trimspool on
set serveroutput on
col area_1  format 99999990.99999
col num_1   format 9999
col area_2  format 99999990.99999
col num_2   format 9999
col area_12 format 99999990.99999
execute dbms_output.enable (2097152);
spool selfovrlap.log

drop table ${BASETABLE}_ovrlap;
create table ${BASETABLE}_ovrlap (
    oid_1   number(12),
    num_1   number(12),
    oid_2   number(12),
    num_2   number(12),
    area_1  number,
    area_2  number,
    area_12 number);
select count(*) from ${BASETABLE};
set timing on

DECLARE

  tmp_geom             ${BASETABLE}.${GEOM_ATTR}%TYPE;
  ovr_geom             ${BASETABLE}.${GEOM_ATTR}%TYPE;
  int_geom             ${BASETABLE}.${GEOM_ATTR}%TYPE;
  ${OID_ATTR}          ${BASETABLE}.${OID_ATTR}%TYPE;
  c_layer              ${BASETABLE}.layer%TYPE;
--cursor base_table is select * from ${BASETABLE} order by ${OID_ATTR};
  cursor base_table is select * from ${BASETABLE};
  type object_list is  table of ${BASETABLE}.${OID_ATTR}%TYPE;
  tmp_objects          object_list;
  nbase                number(12);
  ndistinct            number(12);
  nchk                 number(12) := 0;
  nn                   number(12) := 0;
  interact             number(12);
  int_area             number;
  tot_interact         number(12) := 0;
  logstr               varchar2(255);
  logfile              UTL_FILE.FILE_TYPE;
  oid_not_unique       EXCEPTION;

BEGIN

  logfile := utl_file.fopen('/var/tmp','selfovrlap.log','a');
  select count(*) into nbase from ${BASETABLE};
  logstr := 'Base table (${ORA_USR}.${BASETABLE}) rows: '||nbase;
  utl_file.put_line(logfile,logstr);
  select count(distinct ${OID_ATTR}) into ndistinct from ${BASETABLE};
  if nbase <> ndistinct then
      dbms_output.put_line('                          ');
      dbms_output.put_line('ERROR: Base table (${BASETABLE}) ${OID_ATTR} not unique.');
      raise oid_not_unique;
  end if;
  logstr := 'Potential checks: '||nbase*nbase;
  utl_file.put_line(logfile,logstr);
  utl_file.fflush(logfile);
```

```
  for base_object in base_table loop
      nn := nn + 1;
   if not nn in (12266,18127) then
--    if mod(nn,50) = 0 then
          logstr := '   check object '||nn||' of '||nbase||
                    '(oid='||base_object.${OID_ATTR}||')';
          utl_file.put_line(logfile,logstr);
          utl_file.fflush(logfile);
--    end if;

      tmp_geom := base_object.${GEOM_ATTR};

      select count(*) into interact from ${BASETABLE} where mdsys.sdo_relate
          (${GEOM_ATTR},tmp_geom,'mask=anyinteract querytype=window') = 'TRUE';
      nchk := nchk + nbase;

      if interact > 1 then

          select ${OID_ATTR} bulk collect into tmp_objects from ${BASETABLE} where
              mdsys.sdo_relate(${GEOM_ATTR},tmp_geom,
                  'mask=anyinteract querytype=window') = 'TRUE';

          for n in tmp_objects.first..tmp_objects.last loop

              select layer into c_layer from ${BASETABLE} where
                     ${OID_ATTR} = tmp_objects(n);

              if tmp_objects(n) <> base_object.${OID_ATTR} and
                 c_layer = base_object.layer then

                  select ${GEOM_ATTR} into ovr_geom from ${BASETABLE} where
                      ${OID_ATTR} = tmp_objects(n);
                  int_geom := sdo_geom.sdo_intersection(tmp_geom,ovr_geom,${TOLERANCE});

                  if int_geom is NULL then
                      int_area := 0;
                  else
                      int_area := sdo_geom.sdo_area(int_geom,${TOLERANCE});
                  end if;

                  if int_area > 0 then
                      if base_object.${OID_ATTR} <= tmp_objects(n) then
                          insert into ${BASETABLE}_ovrlap values (
                              base_object.${OID_ATTR},
                              0,
                              tmp_objects(n),
                              0,
                              sdo_geom.sdo_area(tmp_geom,${TOLERANCE}),
                              sdo_geom.sdo_area(ovr_geom,${TOLERANCE}),
                              int_area);
                      else
                          insert into ${BASETABLE}_ovrlap values (
                              tmp_objects(n),
                              0,
                              base_object.${OID_ATTR},
                              0,
                              sdo_geom.sdo_area(ovr_geom,${TOLERANCE}),
                              sdo_geom.sdo_area(tmp_geom,${TOLERANCE}),
                              int_area);
                      end if;

                      tot_interact := tot_interact + 1;
                      logstr := '${BASETABLE} objects '||base_object.${OID_ATTR}||' and '
                                ||tmp_objects(n)||' intersect or overlap';
                      utl_file.put_line(logfile,logstr);
                      utl_file.fflush(logfile);
                  end if;
              end if;
          end loop;
      end if;
   else
          logstr := '   skipping object '||nn||' of '||nbase||
                    ' (oid='||base_object.${OID_ATTR}||')';
          utl_file.put_line(logfile,logstr);
          utl_file.fflush(logfile);
   end if;

  end loop;
```

```
  if tot_interact > 0 then

      select distinct oid_1 bulk collect into tmp_objects from ${BASETABLE}_ovrlap;
      for n in tmp_objects.first..tmp_objects.last loop
          update ${BASETABLE}_ovrlap set num_1 =
              (select count(oid_1) from ${BASETABLE}_ovrlap where oid_1 = tmp_objects(n))
              where oid_1 = tmp_objects(n);
      end loop;

      select distinct oid_2 bulk collect into tmp_objects from ${BASETABLE}_ovrlap;
      for n in tmp_objects.first..tmp_objects.last loop
          update ${BASETABLE}_ovrlap set num_2 =
              (select count(oid_2) from ${BASETABLE}_ovrlap where oid_2 = tmp_objects(n))
              where oid_2 = tmp_objects(n);
      end loop;
  end if;

  logstr := 'Testing ready, '||tot_interact||' interactions of potential of '||nchk||'.';
  utl_file.put_line(logfile,logstr);
  utl_file.fclose(logfile);
  dbms_output.put_line('      ');
  dbms_output.put_line(logstr);

EXCEPTION
  when oid_not_unique then
      return;
END;
/

set timing off
commit;
select distinct * from ${BASETABLE}_ovrlap order by oid_1,oid_2;
spool off
quit
--============================================================================
EOB

sqlplus ${ORA_USR}/${ORA_PWD} @selfovrlap.sql
rm -f selfovrlap.sql
exit
```

Appendix 4: DTB-Nat polygons that overlap with other DTB-Nat polygons

OID_1	NUM_1	OID_2	NUM_2	AREA_1	AREA_2	AREA_12
2091	2	20443	16	193.8428	131393.3416	7.7223
2102	2	20443	16	602.3114	131393.3416	25.0793
2109	2	20443	16	53.4034	131393.3416	0.0096
2192	2	20475	2	88.9262	6380337.9336	88.9262
2775	2	15724	6	207.6360	639.1124	9.8383
2776	2	15725	8	161.2488	467.8823	7.2397
2778	2	15727	22	209.7772	816.0974	24.7726
3223	2	15727	22	10.3296	816.0974	10.3296
3224	2	15727	22	9.1917	816.0974	9.1917
3225	2	15727	22	9.2609	816.0974	9.2609
3226	2	15727	22	9.1350	816.0974	9.1350
3227	2	15727	22	9.8464	816.0974	9.8464
3228	2	15727	22	8.6249	816.0974	8.6249
3229	2	15727	22	8.8773	816.0974	8.8773
3230	2	15727	22	8.8058	816.0974	8.8058
3231	2	15727	22	9.0764	816.0974	9.0764
3300	2	15702	4	25.7615	86.2459	2.6905
3424	2	15702	4	5.6524	86.2459	0.3989
3438	2	20443	16	1.7992	131393.3416	0.2934
3440	2	20443	16	1.4094	131393.3416	0.2276
3441	2	20443	16	2.9417	131393.3416	0.4719
3566	2	17600	2	4.7970	995.1091	4.7970
4408	2	20451	4	8.5164	13291.2397	8.5164
5399	2	12221	2	117.6303	10821.8042	117.6303
8344	2	8380	2	234.5783	3803.8471	0.0925
8962	2	13847	2	56.3566	56.3566	56.3566
8963	2	13851	2	72.0553	72.0553	72.0553
8964	2	13850	2	146.7944	146.7944	146.7944
8969	2	13852	2	7.2208	7.2208	7.2208
11816	2	15724	6	397.1525	639.1124	7.1633
12694	2	15725	8	2357.6557	467.8823	9.0421
13775	2	15727	22	625.3206	816.0974	4.9431
14636	2	15744	2	24.8932	11.3817	11.3806
15092	2	15725	8	192.7565	467.8823	3.5057
15094	2	15725	8	28.4205	467.8823	1.0738
15372	2	15724	6	479.3068	639.1124	8.9425
15702	2	17793	2	86.2459	251.4221	83.1566
15724	10	17050	2	639.1124	59.5566	0.6117
15724	10	17055	2	639.1124	81.1904	0.2953
15724	10	18108	2	639.1124	2083.1058	599.4446
15724	10	18702	2	639.1124	18.8894	10.4684
15724	10	18704	2	639.1124	80.8525	2.3482
15725	6	16756	2	467.8823	2.3206	1.1120
15725	6	18211	10	467.8823	2048.4759	429.0928
15725	6	18776	2	467.8823	104.4514	16.8162
15727	6	16305	2	816.0974	686.4936	628.0237
15727	6	18300	2	816.0974	1437.0548	56.9918
15727	6	18806	2	816.0974	167.0453	18.2182
15733	2	20443	16	48.1906	131393.3416	1.7265
15742	4	20444	2	81.6752	67162.0185	32.6924
15742	4	20451	4	81.6752	13291.2397	48.9828
15744	2	17412	2	11.3817	6469.3665	0.0010
16755	2	18211	10	3.6546	2048.4759	3.6546
16757	2	18211	10	2.2958	2048.4759	2.2958
16758	2	18211	10	3.7108	2048.4759	3.7108
16759	2	18211	10	3.3736	2048.4759	3.3736
16865	2	18933	2	2575.4212	236.2247	236.2247
16866	4	18878	2	7348.4778	42.5193	42.5193
16866	4	18934	2	7348.4778	624.7658	624.7658
16872	2	18879	2	6575.8863	86.4958	86.4958
16879	2	18935	2	2021.4926	187.1378	187.1378
16881	2	18881	2	10092.3475	70.1526	70.1526
16885	2	18880	2	4579.4794	29.4282	29.4282
17047	12	18885	2	15470.6471	2.7315	2.7315
17047	12	18886	2	15470.6471	9.7979	9.7979
17047	12	18887	2	15470.6471	13.7820	13.7820
17047	12	18888	2	15470.6471	171.7916	171.7916
17047	12	18889	2	15470.6471	8.1720	8.1720
17047	12	18892	2	15470.6471	72.4679	72.4679
17049	2	18883	2	11857.2711	64.9606	64.9606
17051	2	18884	2	13041.5378	138.4355	138.4355
17057	4	18890	2	7520.1725	8.7727	8.7727
17057	4	18891	2	7520.1725	33.7184	33.7184
17094	4	18893	2	5869.2399	80.0129	80.0129
17094	4	18936	2	5869.2399	419.5361	419.5361

OID_1	NUM_1	OID_2	NUM_2	AREA_1	AREA_2	AREA_12
17095	4	18894	2	13175.2799	53.4300	53.4300
17095	4	18895	2	13175.2799	62.1225	62.1225
17101	4	18897	2	4390.5386	76.7548	76.7548
17101	4	18937	2	4390.5386	15.9562	15.9562
17104	2	18896	2	4411.0190	145.2082	145.2082
17107	4	18898	2	7747.8879	185.3691	185.3691
17107	4	18938	2	7747.8879	183.9000	183.9000
17120	2	18939	2	2621.9132	195.2125	195.2125
17129	2	18899	2	5062.4827	127.7113	127.7113
17130	2	18940	2	3137.1999	855.4416	855.4416
17133	2	18900	2	5603.6801	196.0172	196.0172
17138	2	18901	2	5784.0629	200.0760	200.0760
17173	1	18870	1	2715.9425	37.2918	37.2918
17183	2	18902	2	5344.6125	151.5230	151.5230
17193	2	18903	2	1581.2857	4.3856	4.3856
17212	2	18904	2	5054.9091	79.9238	79.9238
17218	14	18871	2	2325.5367	13.1207	13.1207
17218	14	18876	2	2325.5367	10.8133	10.8133
17218	14	18905	2	2325.5367	16.3770	16.3770
17218	14	18906	2	2325.5367	10.2555	10.2555
17218	14	18907	2	2325.5367	3.2506	3.2506
17218	14	18908	2	2325.5367	13.8958	13.8958
17218	14	18909	2	2325.5367	4.4463	4.4463
17225	2	18911	2	6969.8969	100.1981	100.1981
17226	2	18910	2	7363.7045	135.6940	135.6940
17227	2	18872	2	3285.8750	7.6359	7.6359
17231	8	18873	2	10196.3807	15.2985	15.2985
17231	8	18874	2	10196.3807	21.2961	21.2961
17231	8	18875	2	10196.3807	38.1960	38.1960
17231	8	18877	2	10196.3807	26.3509	26.3509
17234	2	18912	2	6430.7064	147.8755	147.8755
17235	2	18913	2	6516.2778	106.4127	106.4127
17268	2	18942	2	5566.0850	465.7358	465.7358
17270	2	18944	4	2910.0453	125.5978	81.4547
17272	8	18941	2	13431.9520	254.4832	254.4832
17272	8	18943	2	13431.9520	674.4276	674.4276
17272	8	18944	4	13431.9520	125.5978	44.1431
17272	8	18945	2	13431.9520	200.8326	200.8326
17288	2	18914	2	8339.2931	138.7907	138.7907
17289	2	18918	2	1276.4893	6.2247	6.2247
17299	2	18916	2	7824.8165	67.1100	67.1100
17305	2	18917	2	2051.4102	4.4184	4.4184
17318	2	18919	2	5928.8315	54.8062	54.8062
17320	6	18920	2	26184.5470	81.7465	81.7465
17320	6	18921	2	26184.5470	63.5193	63.5193
17320	6	18922	2	26184.5470	36.7186	36.7186
17336	2	18923	2	4743.9206	26.8678	26.8678
17348	2	18924	2	3379.0767	114.4194	114.4194
17349	2	18925	2	2935.8479	117.9606	117.9606
17351	2	18926	2	6612.3777	193.5491	193.5491
17353	2	18928	2	6271.7568	127.9583	127.9583
17356	2	18927	2	8104.7998	99.9890	99.9890
17363	2	18946	2	3617.4348	582.9358	582.9358
17366	2	18929	2	8783.6182	269.9716	269.9716
17369	2	18930	2	5679.4710	20.0315	20.0315
17374	2	18931	2	5986.0090	114.2341	114.2341
17421	2	18947	2	1836.9203	468.2907	468.2907
17424	4	18932	2	2523.6307	83.1343	83.1343
17424	4	18948	2	2523.6307	10.6379	10.6379
17805	2	18543	2	1819.4905	48.2440	48.2440
18299	2	18915	2	1724.5358	10.3704	10.3704
20411	2	20443	16	110534.9799	131393.3416	18.5353
20565	2	20573	2	32.1957	31037.4552	3.9173
20567	2	20925	2	31.7385	28.2206	28.2206
20573	2	20928	2	31037.4552	27900.3426	27900.3426
20574	2	20929	2	8099.1952	4433.3932	4433.3932
20575	2	20930	2	916.9915	487.8037	487.8037
20576	8	20931	2	19455.3621	10232.7483	10232.7483
20576	8	20932	2	19455.3621	7534.1747	7534.1747
20576	8	20955	2	19455.3621	414.3703	414.3703
20576	8	20958	4	19455.3621	94.4219	94.4219
20592	2	20951	2	1856.7080	1668.3796	1668.3796
20593	2	20952	2	1858.6094	1674.1507	1674.1507
20594	2	20954	2	44168.4991	43912.9795	43912.9795
20595	2	20707	2	12.2823	179.9436	0.0000
20686	2	20823	2	26.1139	48.4807	0.0088
20739	2	20790	2	47.8186	402.9236	0.0049
20902	2	20924	2	412.6107	272.7980	272.7980
20903	2	20926	2	460.5042	304.8227	304.8227

OID_1	NUM_1	OID_2	NUM_2	AREA_1	AREA_2	AREA_12
20904	2	20927	2	531.8468	282.9354	282.9354
20905	4	20935	2	2359.4163	1600.1504	1600.1504
20905	4	20936	2	2359.4163	455.9160	455.9160
20906	2	20938	2	644.1877	571.4400	571.4400
20907	2	20939	2	982.1488	887.9823	887.9823
20908	2	20940	2	927.4921	880.8812	880.7985
20909	2	20941	2	926.9134	886.7279	886.7279
20910	2	20942	2	768.2716	496.4026	496.4026
20911	2	20943	2	1039.4576	851.2788	851.2788
20912	4	20944	2	1331.7872	612.7987	612.7987
20912	4	20945	2	1331.7872	608.5777	608.5777
20913	2	20946	2	751.3078	502.2870	502.2870
20914	6	20947	2	766.9648	451.5287	451.5287
20914	6	20956	2	766.9648	122.9743	122.9743
20914	6	20957	2	766.9648	116.6908	116.6908
20915	2	20948	2	1635.3519	1567.1013	1567.1013
20916	2	20949	2	726.3433	490.7543	490.7543
20917	2	20950	2	912.8666	597.4627	597.4627
20918	4	20919	2	1215.0331	31.2993	31.2993
20918	4	20953	2	1215.0331	1137.9950	1137.9950
20931	2	20958	4	10232.7483	94.4219	94.4219

175 rows selected.

Appendix 5: Results of the overlap tests with Beheerkaart-Nat and Regiokaart-Nat

The IN.... columns refer to the input layer (the first table specified in the script, Beheerkaart-Nat in this case), the MT.... columns refer to the layer matched against (the second table specified, Regiokaart-Nat in this example).

```
SQL> select in_oid, in_area, in_pct_overlap, mt_oid, mt_area, mt_pct_overlap from
     holdiep_haring_reg_match order by in_oid;
```

IN_OID	IN_AREA	IN_PCT_OVERLAP	MT_OID	MT_AREA	MT_PCT_OVERLAP
1	18781.4504	.040255232	77	3717185.79	.000203394
1	18781.4504	98.4226828	99	135590.665	13.6330973
2	80427.2441	1.23139947	77	3717185.79	.026643292
2	80427.2441	.035516158	102	82381.3064	.034673724
2	80427.2441	80.3547263	100	258275.227	25.022567
2	80427.2441	18.3783581	99	135590.665	10.9013455
3	385629.347	11.9644845	99	135590.665	34.0278318
3	385629.347	.253988719	100	258275.227	.379229185
3	385629.347	74.2698204	104	716144.378	39.9928048
3	385629.347	2.08681193	105	51938.1563	15.494118
3	385629.347	.059065097	106	5792004.17	.003932531
3	385629.347	.052679894	112	777499.689	.026128516
3	385629.347	.00002946	113	217914.607	.000052133
3	385629.347	.000370657	115	52341.4763	.002730842
3	385629.347	3.47882028	117	1311413.95	1.02296852
3	385629.347	1.20502025	118	169960.376	2.73411476
3	385629.347	.044330055	127	294504.679	.058046515
3	385629.347	1.01405533	130	220229.045	1.77564906
3	385629.347	.596679458	135	326166.805	.705458394
4	162563.803	7.16660075	99	135590.665	8.59225721
4	162563.803	91.1722553	100	258275.227	57.3857149
4	162563.803	.898677527	106	5792004.17	.025223124
4	162563.803	.762466433	102	82381.3064	1.50458215
5	6126369.39	.03179684	99	135590.665	1.43667106
5	6126369.39	.292104692	100	258275.227	6.92881489
5	6126369.39	.298976382	102	82381.3064	22.2336818
5	6126369.39	.114567143	104	716144.378	.980082591
5	6126369.39	.046169269	105	51938.1563	5.44589987
5	6126369.39	92.7778455	106	5792004.17	98.1337955
5	6126369.39	.53234314	108	68668.5782	47.4937854
5	6126369.39	.471472292	111	49366.7134	58.5093319
5	6126369.39	.751258702	112	777499.689	5.91960149
5	6126369.39	.242752596	113	217914.607	6.82465527
5	6126369.39	.047613906	114	112027.79	2.60382159
5	6126369.39	.649387995	115	52341.4763	76.0083784
5	6126369.39	.220228777	117	1311413.95	1.02881537
5	6126369.39	1.06757155	118	169960.376	38.4815436
5	6126369.39	.072696541	119	4667.35681	95.4214299
5	6126369.39	.021669023	120	1327.52441	100
5	6126369.39	.380812091	123	63791.3297	36.5722983
5	6126369.39	1.33534524	127	294504.679	27.7782282
5	6126369.39	.05879193	128	1185685.07	.303774662
5	6126369.39	.084492071	130	220229.045	2.35041494
5	6126369.39	.372511913	135	326166.805	6.99686647
5	6126369.39	.100658855	139	53133.9384	11.6060158
6	56856.63	33.5325497	104	716144.378	2.66223939
6	56856.63	.494910358	106	5792004.17	.004858238
6	56856.63	63.3958311	108	68668.5782	52.4908686
6	56856.63	.625624332	114	112027.79	.317518458
6	56856.63	.086319008	119	4667.35681	1.05151762
7	1604.60791	99.2366017	104	716144.378	.222351583
8	56.8411865	75.235709	100	258275.227	.016557867
8	56.8411865	24.764291	106	5792004.17	.00024303
9	33628.3734	8.53752236	100	258275.227	1.11161644
9	33628.3734	89.9888167	105	51938.1563	58.2650165
9	33628.3734	1.47366095	106	5792004.17	.008556075
10	47471.752	34.2725366	99	135590.665	11.9991841
10	47471.752	22.5547434	105	51938.1563	20.6151559
10	47471.752	.337751221	106	5792004.17	.002768237
10	47471.752	42.7822824	111	49366.7134	41.140067
11	51674.889	96.3855183	104	716144.378	6.9548978
11	51674.889	.020392777	108	68668.5782	.015346094
11	51674.889	.38493968	114	112027.79	.177560543
11	51674.889	2.90177755	115	52341.4763	2.86482238
12	4738473.81	.002916944	102	82381.3064	.16777914
12	4738473.81	.022122498	153	20138.0815	5.20540525
12	4738473.81	.007065113	146	31932.8333	1.0483834
12	4738473.81	98.6366386	110	4678610.97	99.8986945

IN_OID	IN_AREA	IN_PCT_OVERLAP	MT_OID	MT_AREA	MT_PCT_OVERLAP
12	4738473.81	7.2138E-06	126	523237.833	.000065329
12	4738473.81	.000187238	140	407557.801	.002176924
12	4738473.81	.180478047	145	33745.3375	25.3424788
12	4738473.81	.005894662	144	26811.8389	1.0417674
12	4738473.81	1.14457973	109	643424.471	8.42921166
12	4738473.81	.000109986	106	5792004.17	.00008998
13	756923.473	2.84638378	104	716144.378	3.00846416
13	756923.473	.642942102	106	5792004.17	.084022379
13	756923.473	96.5106741	112	777499.689	93.9565581
14	158398.459	8.01465181	104	716144.378	1.7726991
14	158398.459	2.93660822	106	5792004.17	.080309717
14	158398.459	.479619307	112	777499.689	.097711884
14	158398.459	88.3336457	113	217914.607	64.2082398
15	60073.411	11.5498318	106	5792004.17	.11979235
15	60073.411	88.4501682	113	217914.607	24.3834196
16	12422.2582	19.5926696	106	5792004.17	.042020895
16	12422.2582	80.4073304	113	217914.607	4.58363315
17	898.069946	80.7275368	106	5792004.17	.012517079
17	898.069946	19.2724632	111	49366.7134	.350601019
18	117335.325	2.64944011	104	716144.378	.434092518
18	117335.325	2.87534154	115	52341.4763	6.4457321
18	117335.325	.140298864	119	4667.35681	3.52705258
18	117335.325	92.5178843	114	112027.79	96.9010994
18	117335.325	1.81703518	106	5792004.17	.036809782
19	11568746.8	.013648262	110	4678610.97	.033747899
19	11568746.8	99.9861804	116	11568396.8	99.9892059
19	11568746.8	.000171366	142	62865.8105	.031535286
20	1303850.15	1.23242476	104	716144.378	2.24381738
20	1303850.15	.735019042	106	5792004.17	.16546167
20	1303850.15	.236908444	115	52341.4763	5.90149783
20	1303850.15	97.7956478	117	1311413.95	97.2315946
21	105176.87	4.24851122	104	716144.378	.623959532
21	105176.87	.777087059	106	5792004.17	.014111106
21	105176.87	94.9744017	118	169960.376	58.773171
22	231.718262	100	106	5792004.17	.004000658
23	501.371948	99.4538918	106	5792004.17	.008609005
23	501.371948	.546108195	118	169960.376	.001610983
24	62354.3334	12.729876	106	5792004.17	.137044606
24	62354.3334	7.36728801	115	52341.4763	8.77664072
24	62354.3334	15.0134015	117	1311413.95	.713848318
24	62354.3334	64.8894345	123	63791.3297	63.4277017
25	1340983.82	1.1709E-06	116	11568396.8	1.3573E-07
25	1340983.82	99.9995722	124	1340978.08	100
25	1340983.82	.000426617	131	2081444	.000274851
26	378.624512	100	106	5792004.17	.006537021
27	574540.961	9.17661344	125	192219.649	27.4287272
27	574540.961	88.6678444	126	523237.833	97.3616687
27	574540.961	1.98381594	136	58795.2283	19.3856466
27	574540.961	.171726191	144	26811.8389	3.67985691
28	1232453.82	4.08584271	106	5792004.17	.869407599
28	1232453.82	.001171108	117	1311413.95	.001100595
28	1232453.82	95.9129862	128	1185685.07	99.6962253
29	220741.587	2.72344549	104	716144.378	.839464354
29	220741.587	.318467206	106	5792004.17	.012137242
29	220741.587	96.0563848	127	294504.679	71.9976294
29	220741.587	.901702555	135	326166.805	.610249878
30	210277.833	100	129	210277.833	100
31	924.521606	98.2425942	106	5792004.17	.015681515
31	924.521606	1.75740584	118	169960.376	.009559638
32	220667.824	3.12921751	104	716144.378	.964215653
32	220667.824	.137817529	106	5792004.17	.005250669
32	220667.824	.009940184	117	1311413.95	.001672606
32	220667.824	94.8106756	130	220229.045	94.9995741
32	220667.824	1.91234918	139	53133.9384	7.94207894
33	2082198.43	.036507396	116	11568396.8	.006570975
33	2082198.43	99.9634926	131	2081444	99.9997251
34	3817120.99	98.2606226	133	4217312.26	88.9364274
34	3817120.99	3.5967E-06	143	434690.789	.000031584
34	3817120.99	.002067458	148	48269.4996	.16349321
34	3817120.99	.05511628	149	217518.216	.967208696
34	3817120.99	.027500065	151	59919.257	1.75187545
34	3817120.99	1.36873846	154	975778.524	5.35433009
34	3817120.99	.007100781	168	100371.206	.270042986
34	3817120.99	.012468095	169	917877.636	.051850297
34	3817120.99	.048000495	170	31419.317	5.83156205
35	313500.213	2.47960256	104	716144.378	1.08547376
35	313500.213	1.48669055	106	5792004.17	.080469176
35	313500.213	.156031838	127	294504.679	.166095882
35	313500.213	.043304968	130	220229.045	.061645441
35	313500.213	95.3558368	135	326166.805	91.652721

IN_OID	IN_AREA	IN_PCT_OVERLAP	MT_OID	MT_AREA	MT_PCT_OVERLAP
35	313500.213	.478533323	138	40803.7196	3.67663292
36	45047.0118	8.29185579	106	5792004.17	.064489478
36	45047.0118	3.96460504	130	220229.045	.810944852
36	45047.0118	.251279048	135	326166.805	.034704238
36	45047.0118	86.8569884	138	40803.7196	95.8894881
36	45047.0118	.635271682	139	53133.9384	.538584036
37	46225.0488	7.75123819	106	5792004.17	.061861379
37	46225.0488	.00844042	130	220229.045	.001771605
37	46225.0488	.382993113	138	40803.7196	.433878958
37	46225.0488	91.8573283	139	53133.9384	79.9133212
38	409904.395	100	133	4217312.26	9.71956475
39	414739.199	.154536043	110	4678610.97	.013698971
39	414739.199	97.6575556	140	407557.801	99.3783368
39	414739.199	1.49800358	141	179931.068	3.45288234
39	414739.199	.003256096	142	62865.8105	.021481162
39	414739.199	.667524796	145	33745.3375	8.2040578
39	414739.199	.019123845	150	109105.952	.072694552
40	172181.04	.011084518	110	4678610.97	.00040793
40	172181.04	1.3967864	140	407557.801	.590100678
40	172181.04	82.9273859	141	179931.068	79.3555206
40	172181.04	11.0479201	142	62865.8105	30.2587744
40	172181.04	.010034332	161	111093.91	.015551904
40	172181.04	1.05612823	162	6916.11121	26.292992
40	172181.04	3.55066053	163	107563.587	5.68367455
41	31710.0071	.058662452	110	4678610.97	.000397594
41	31710.0071	1.53974965	116	11568396.8	.004220591
41	31710.0071	.00895543	140	407557.801	.000696777
41	31710.0071	.447613867	141	179931.068	.07888487
41	31710.0071	93.3615908	142	62865.8105	47.0923174
41	31710.0071	4.48037839	143	434690.789	.326836533
41	31710.0071	.103049458	161	111093.91	.029413845
42	455225.72	.000062112	116	11568396.8	2.4442E-06
42	455225.72	.001496577	133	4217312.26	.000161544
42	455225.72	3.12033913	142	62865.8105	22.5950897
42	455225.72	94.8234814	143	434690.789	99.3029729
42	455225.72	1.71442162	149	217518.216	3.58796993
42	455225.72	.340199198	161	111093.91	1.39402263
43	18310.6813	.500389744	110	4678610.97	.001958376
43	18310.6813	97.5205814	145	33745.3375	52.9160001
43	18310.6813	.236811678	153	20138.0815	.215322555
43	18310.6813	1.74221715	156	14041.1412	2.27197935
44	43350.9786	3.69383562	110	4678610.97	.034226267
44	43350.9786	23.6560814	144	26811.8389	38.248562
44	43350.9786	72.650083	146	31932.8333	98.6273961
45	189980.457	30.7167529	83	1483918.6	3.93254908
45	189980.457	36.3918031	109	643424.471	10.7452105
45	189980.457	.022973577	110	4678610.97	.000932869
45	189980.457	1.99203924	125	192219.649	1.96883372
45	189980.457	3.58863742	144	26811.8389	25.4279827
45	189980.457	.05449654	146	31932.8333	.324220451
45	189980.457	19.2890967	152	39319.9961	93.1981628
45	189980.457	1.04173664	153	20138.0815	9.82762939
45	189980.457	.02461228	155	33098.4941	.141270844
45	189980.457	.003670775	158	6.97375488	100
45	189980.457	.015206048	159	65319.6434	.044226389
46	26694.9857	.4601237	133	4217312.26	.002912517
46	26694.9857	99.5398763	148	48269.4996	55.0495778
47	159110.97	.136059472	133	4217312.26	.005133259
47	159110.97	.905492583	143	434690.789	.331439743
47	159110.97	97.1227815	149	217518.216	71.043705
47	159110.97	1.81435308	161	111093.91	2.59855359
48	117667.44	.099367751	140	407557.801	.028688811
48	117667.44	3.30460225	141	179931.068	2.16107252
48	117667.44	3.87616568	145	33745.3375	13.5158966
48	117667.44	92.4020465	150	109105.952	99.6527873
48	117667.44	.059032381	156	14041.1412	.494702603
48	117667.44	.258785434	162	6916.11121	4.40285279
49	37368.8789	1.77492189	133	4217312.26	.015727278
49	37368.8789	92.7190771	151	59919.257	57.824615
50	17709.1971	.02792177	110	4678610.97	.000105688
50	17709.1971	9.07514295	156	14041.1412	11.4458998
50	17709.1971	.726964295	160	36579.2174	.351947224
50	17709.1971	.042943225	159	65319.6434	.011642593
50	17709.1971	90.0859319	153	20138.0815	79.2205318
50	17709.1971	.041095845	145	33745.3375	.021566666
51	2029813.97	2.33292205	133	4217312.26	1.12284732
51	2029813.97	4.3411E-06	151	59919.257	.000147057
51	2029813.97	45.3683782	154	975778.524	94.3752762
51	2029813.97	.299761069	168	100371.206	6.06208918

IN_OID	IN_AREA	IN_PCT_OVERLAP	MT_OID	MT_AREA	MT_PCT_OVERLAP
51	2029813.97	45.1827455	169	917877.636	99.9180766
52	32022.104	6.43341626	152	39319.9961	5.23935771
52	32022.104	93.5665838	155	33098.4941	90.5237218
53	14524.7992	1.5686443	150	109105.952	.208826768
53	14524.7992	82.9308027	156	14041.1412	85.7874183
53	14524.7992	6.73649653	157	17394.1995	5.62522348
53	14524.7992	1.64493205	160	36579.2174	.653166179
54	74332.8069	.61292238	152	39319.9961	1.15870411
54	74332.8069	1.49847651	153	20138.0815	5.53111105
54	74332.8069	4.15663956	155	33098.4941	9.33500734
54	74332.8069	87.3185804	159	65319.6434	99.3672781
54	74332.8069	6.41338111	160	36579.2174	13.0326632
55	16728.0223	.136715074	150	109105.952	.020961027
55	16728.0223	95.1762789	157	17394.1995	91.5311408
56	23702.7426	.244968942	159	65319.6434	.088892643
56	23702.7426	97.7420434	160	36579.2174	63.3352667
57	125064.279	12.2735297	141	179931.068	8.53093446
57	125064.279	.134578113	143	434690.789	.038719281
57	125064.279	2.14180178	149	217518.216	1.23145041
57	125064.279	85.2054709	161	111093.91	95.9202962
57	125064.279	.000403162	142	62865.8105	.000802044
58	4677.40222	1.04338796	150	109105.952	.044730329
58	4677.40222	10.5748373	157	17394.1995	2.84363575
58	4677.40222	88.0005098	162	6916.11121	59.5152055
58	4677.40222	.30378364	163	107563.587	.013210031
59	94556.9781	.329676867	141	179931.068	.17325106
59	94556.9781	.412653799	162	6916.11121	5.64179712
59	94556.9781	89.1630431	163	107563.587	78.3814312
59	94556.9781	6.50304563	164	74492.3363	8.25465242
59	94556.9781	1.27460627	165	15770.7859	7.64216305
59	94556.9781	.028017769	166	395644.029	.006696109
60	907.706421	100	141	179931.068	.504474537
61	467.507813	100	141	179931.068	.259826065
62	766.883545	100	141	179931.068	.426209633
63	59521.5739	.298298176	141	179931.068	.098677661
63	59521.5739	1.83251665	163	107563.587	1.0140446
63	59521.5739	3.60059233	165	15770.7859	13.5892354
63	59521.5739	.42143868	166	395644.029	.063402179
63	59521.5739	93.8471541	164	74492.3363	74.9866442
64	1601.59802	100	141	179931.068	.890117555
65	414007.721	.213897258	141	179931.068	.492161344
65	414007.721	.760189362	163	107563.587	2.92593687
65	414007.721	3.01539055	164	74492.3363	16.7587034
65	414007.721	93.841204	166	395644.029	98.1968139
65	414007.721	.381426215	167	149639.072	1.05529522
65	414007.721	1.7878926	165	15770.7859	46.9349686
66	3563.62988	18.592956	141	179931.068	.368243319
66	3563.62988	81.0722567	165	15770.7859	18.3194115
66	3563.62988	.334787189	166	395644.029	.003015482
67	464.577271	100	141	179931.068	.258197362
68	756.375488	100	141	179931.068	.420369588
69	1517.9043	100	141	179931.068	.843603228
70	841.233398	63.6782507	141	179931.068	.297715519
70	841.233398	36.3217493	165	15770.7859	1.93744743
71	464.440552	100	141	179931.068	.258121378
72	755.332642	100	141	179931.068	.419790007
73	1507.88538	52.3337059	141	179931.068	.438574787
74	125027.794	.196568421	141	179931.068	.136588507
74	125027.794	1.36104153	165	15770.7859	10.7900787
74	125027.794	3.72957053	166	395644.029	1.17858464
74	125027.794	94.5087578	167	149639.072	78.9648138
75	64387.8397	3.93871701	133	4217312.26	.060134385
75	64387.8397	95.6331484	168	100371.206	61.3483892
75	64387.8397	.349153739	169	917877.636	.024492649
76	35653.7737	1.9504305	133	4217312.26	.016489224
76	35653.7737	72.3902099	170	31419.317	82.146412

293 rows selected.

Appendix 6: Matched objects from Beheerkaart-Nat and Regiokaart-Nat that have an overlap of more than 60%

```
SQL> select * from beheer_regio_link_code order by beh_id;
```

BEH_ID	REG_ID	BEH_CODE	REG_CODE
1	99	108C.OO.R.985,0	
2	100	108C.OU.R.998,1	108C.OU.R.998,1
3	104	108C.OO.R.985,0	
4	100	108C.OU.R.998,1	108C.OU.R.998,1
5	106	108C.BO.R.985,0	108C.BO.R.985,0
5	115	108C.BO.R.985,0	108C.OH.R.986,4
5	119	108C.BO.R.985,0	108C.OU.R.985,6
5	120	108C.BO.R.985,0	108C.OU.R.985,6
6	108	108C.OO.R.984,0	108C.OO.R.984,0
7	104	108C.OO.R.985,0	
9	105	108C.OH.R.999,1	108C.OH.R.999,1
11	104	108C.OO.R.985,0	
12	110	108C.BV.-.997,6	108C.BV.-.997,6
13	112	108C.OU.R.996,6	108C.OU.R.99,6
14	113	108C.OU.R.997,6	108C.OU.R.997,6
15	113	108C.OU.R.997,6	108C.OU.R.997,6
16	113	108C.OU.R.997,6	108C.OU.R.997,6
18	114	108C.OU.R.985,6	108C.OU.R.985,6
19	116	108C.BV.-.985,0	108C.BV.-.985,0
20	117	108C.OX.R.990,5	108C.OX.R.990,5
21	118	108C.OU.R.996,1	108C.OU.R.996,1
24	123	108C.OU.R.986,7	108C.OU.R.98,6
25	124	108C.BV.-.991,7	108C.BV.-.991,7
26	106	108C.OD.L.986,6	108C.BO.R.985,0
27	126	108C.BO.-.999,1	108C.BO.-.999,1
28	128	108C.OU.R.987,6	108C.OU.R.987,6
29	127	108C.OO.R.991,8	108C.OO.R.991,8
30	129	108C.BO.R.993,9	108C.BO.R.993,9
32	130	108C.OX.R.990,5	108C.OX.R.990.50
33	131	108C.BO.-.992,0	108C.BO.-.992,0
34	133	108C.BO.L.985,0	108C.BO.L.985,0
35	135	108C.OU.R.990,5	108C.OU.R.990,5
36	138	108C.KT.R.990,5	108C.KT.R.990,5
37	139	108C.OU.R.990,5	108C.OU.R.990,5
39	140	108C.BO.-.998,2	108C.BO.-.998,0
40	141	108C.FT.-.997,5	108C.FT.L.997,5
41	142	108C.OO.L.998,0	108C.OO.L.998,0
42	143	108C.BV.-.997,8	108C.BV.-.997,8
43	145	108C.OO.L.999,3	108C.OO.L.999,3
44	146	108C.BH.-.998,8	108C.BH.-.99,8
46	148	108C.OH.L.996,9	108C.OH.L.996,9
47	149	108C.OO.L.997,5	108C.OO.L.997,5
48	150	108C.BO.-.998,2	108C.BO.-.998,2
49	151	108C.OH.L.996,7	108C.OH.L.996,7
50	153	108C.KS.L.997,5	108C.KS.L.997,5
51	154	108C.OU.L.993,3	108C.OU.L.99,3
51	169	108C.OU.L.993,3	108C.OU.L.991,3
53	156	108C.KV.L.997,5	108C.KV.L.997,5
54	159	108C.BH.-.998,8	108C.BH.-.99,8
55	157	108C.OO.L.998,1	108C.OO.L.998,1
56	160	108C.OO.L.1000,0	108C.OO.L.1,0
57	161	108C.KS.L.997,5	108C.KS.L.997,5
58	162	108C.KX.L.997,5	108C.KX.L.997,5
59	163	108C.OO.L.1000,0	108C.OO.L.1000.
60	141	108C.KB.L.997,5	108C.FT.L.997,5
61	141	108C.BV.-.999,7	108C.FT.L.997,5
62	141	108C.KB.L.997,5	108C.FT.L.997,5
63	164	108C.OO.L.1000.0	108C.OO.L.1,0
64	141	108C.KB.L.997,5	108C.FT.L.997,5
65	166	108C.BV.-.999,7	108C.BV.-.999,7
66	165	108C.KS.L.997,5	108C.KB.L.997,5
67	141	108C.BV.-.999,7	108C.FT.L.997,5
68	141	108C.KB.L.997,5	108C.FT.L.997,5
69	141	108C.KB.L.997,5	108C.FT.L.997,5
70	141	108C.KS.L.997,5	108C.FT.L.997,5
71	141	108C.BV.-.999,7	108C.FT.L.997,5
72	141	108C.KB.L.997,5	108C.FT.L.997,5
74	167	108C.KH.L.997,5	108C.KH.L.997,5
75	168	108C.OH.L.991,0	108C.OH.L.991,0
76	170	108C.OO.L.990,6	108C.OO.L.990,6

```
70 rows selected.
```

Appendix 7: Matched objects resulting from overlap (more than 60%) and entire code

```
SQL> select * from beheer_regio_link_code order by beh_id;
```

BEH_ID	REG_ID	BEH_CODE	REG_CODE
1	99	108C.OO.R.985,0	
2	100	108C.OU.R.998,1	108C.OU.R.998,1
3	104	108C.OO.R.985,0	
4	100	108C.OU.R.998,1	108C.OU.R.998,1
5	106	108C.BO.R.985,0	108C.BO.R.985,0
6	108	108C.OO.R.984,0	108C.OO.R.984,0
7	104	108C.OO.R.985,0	
9	105	108C.OH.R.999,1	108C.OH.R.999,1
11	104	108C.OO.R.985,0	
12	110	108C.BV.-.997,6	108C.BV.-.997,6
14	113	108C.OU.R.997,6	108C.OU.R.997,6
15	113	108C.OU.R.997,6	108C.OU.R.997,6
16	113	108C.OU.R.997,6	108C.OU.R.997,6
18	114	108C.OU.R.985,6	108C.OU.R.985,6
19	116	108C.BV.-.985,0	108C.BV.-.985,0
20	117	108C.OX.R.990,5	108C.OX.R.990,5
21	118	108C.OU.R.996,1	108C.OU.R.996,1
25	124	108C.BV.-.991,7	108C.BV.-.991,7
27	126	108C.BO.-.999,1	108C.BO.-.999,1
28	128	108C.OU.R.987,6	108C.OU.R.987,6
29	127	108C.OO.R.991,8	108C.OO.R.991,8
30	129	108C.BO.R.993,9	108C.BO.R.993,9
33	131	108C.BO.-.992,0	108C.BO.-.992,0
34	133	108C.BO.L.985,0	108C.BO.L.985,0
35	135	108C.OU.R.990,5	108C.OU.R.990,5
36	138	108C.KT.R.990,5	108C.KT.R.990,5
37	139	108C.OU.R.990,5	108C.OU.R.990,5
41	142	108C.OO.L.998,0	108C.OO.L.998,0
42	143	108C.BV.-.997,8	108C.BV.-.997,8
43	145	108C.OO.L.999,3	108C.OO.L.999,3
46	148	108C.OH.L.996,9	108C.OH.L.996,9
47	149	108C.OO.L.997,5	108C.OO.L.997,5
48	150	108C.BO.-.998,2	108C.BO.-.998,2
49	151	108C.OH.L.996,7	108C.OH.L.996,7
50	153	108C.KS.L.997,5	108C.KS.L.997,5
53	156	108C.KV.L.997,5	108C.KV.L.997,5
55	157	108C.OO.L.998,1	108C.OO.L.998,1
57	161	108C.KS.L.997,5	108C.KS.L.997,5
58	162	108C.KX.L.997,5	108C.KX.L.997,5
65	166	108C.BV.-.999,7	108C.BV.-.999,7
74	167	108C.KH.L.997,5	108C.KH.L.997,5
75	168	108C.OH.L.991,0	108C.OH.L.991,0
76	170	108C.OO.L.990,6	108C.OO.L.990,6

```
43 rows selected.
```

Appendix 8: Matched objects resulting from overlap (more than 60%) and part of the code (only the first 10 characters)

```
SQL> select * from beheer_regio_link_code order by beh_id;
```

BEH_ID	REG_ID	BEH_CODE	REG_CODE
1	99	108C.OO.R.985,0	
2	100	108C.OU.R.998,1	108C.OU.R.998,1
3	104	108C.OO.R.985,0	
4	100	108C.OU.R.998,1	108C.OU.R.998,1
5	106	108C.BO.R.985,0	108C.BO.R.985,0
6	108	108C.OO.R.984,0	108C.OO.R.984,0
7	104	108C.OO.R.985,0	
9	105	108C.OH.R.999,1	108C.OH.R.999,1
11	104	108C.OO.R.985,0	
12	110	108C.BV.-.997,6	108C.BV.-.997,6
13	112	108C.OU.R.996,6	108C.OU.R.99,6
14	113	108C.OU.R.997,6	108C.OU.R.997,6
15	113	108C.OU.R.997,6	108C.OU.R.997,6
16	113	108C.OU.R.997,6	108C.OU.R.997,6
18	114	108C.OU.R.985,6	108C.OU.R.985,6
19	116	108C.BV.-.985,0	108C.BV.-.985,0
20	117	108C.OX.R.990,5	108C.OX.R.990,5
21	118	108C.OU.R.996,1	108C.OU.R.996,1
24	123	108C.OU.R.986,7	108C.OU.R.98,6
25	124	108C.BV.-.991,7	108C.BV.-.991,7
27	126	108C.BO.-.999,1	108C.BO.-.999,1
28	128	108C.OU.R.987,6	108C.OU.R.987,6
29	127	108C.OO.R.991,8	108C.OO.R.991,8
30	129	108C.BO.R.993,9	108C.BO.R.993,9
32	130	108C.OX.R.990,5	108C.OX.R.990.50
33	131	108C.BO.-.992,0	108C.BO.-.992,0
34	133	108C.BO.L.985,0	108C.BO.L.985,0
35	135	108C.OU.R.990,5	108C.OU.R.990,5
36	138	108C.KT.R.990,5	108C.KT.R.990,5
37	139	108C.OU.R.990,5	108C.OU.R.990,5
39	140	108C.BO.-.998,2	108C.BO.-.998,0
41	142	108C.OO.L.998,0	108C.OO.L.998,0
42	143	108C.BV.-.997,8	108C.BV.-.997,8
43	145	108C.OO.L.999,3	108C.OO.L.999,3
44	146	108C.BH.-.998,8	108C.BH.-.99,8
46	148	108C.OH.L.996,9	108C.OH.L.996,9
47	149	108C.OO.L.997,5	108C.OO.L.997,5
48	150	108C.BO.-.998,2	108C.BO.-.998,2
49	151	108C.OH.L.996,7	108C.OH.L.996,7
50	153	108C.KS.L.997,5	108C.KS.L.997,5
51	154	108C.OU.L.993,3	108C.OU.L.99,3
51	169	108C.OU.L.993,3	108C.OU.L.991,3
53	156	108C.KV.L.997,5	108C.KV.L.997,5
54	159	108C.BH.-.998,8	108C.BH.-.99,8
55	157	108C.OO.L.998,1	108C.OO.L.998,1
56	160	108C.OO.L.1000,0	108C.OO.L.1,0
57	161	108C.KS.L.997,5	108C.KS.L.997,5
58	162	108C.KX.L.997,5	108C.KX.L.997,5
59	163	108C.OO.L.1000,0	108C.OO.L.1000.
63	164	108C.OO.L.1000.0	108C.OO.L.1,0
65	166	108C.BV.-.999,7	108C.BV.-.999,7
74	167	108C.KH.L.997,5	108C.KH.L.997,5
75	168	108C.OH.L.991,0	108C.OH.L.991,0
76	170	108C.OO.L.990,6	108C.OO.L.990,6

```
54 rows selected.
```

Reports published before in this series:

- 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.
- GISt Report No. 2, Stoter, J.E., Considerations for a 3D Cadastre, Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 30.p.
- 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.
- 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.
- 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.
- GISt Report No. 6, Quak, C.W., An implementation of a classification algorithm for houses, Rapport aan Concernstaf Kadaster, Delft 2001, 13.p.
- 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.
- GISt Report No. 8, Vries, M.E. de en E. Verbree, Internet GIS met ArcIMS, Delft 2001, 38 p.
- 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.
- 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.
- GISt Report No. 11, Geo DBMS, De basis van GIS-toepassingen, KvAG/AGGN Themamiddag, 14 november 2001, J. Flim (eds.), Delft 2001, 37 p.
- 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 TOP10NL object model, Delft, 2002, 68 p.
- 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)
- GISt Report No. 14, Stoter, J.E., 3D Cadastre, Progress Report, Report to Concernstaf Kadaster, Delft 2002, 16 p.
- 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.
- 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.
- GISt Report No. 17, Zlatanova, S. Augmented Reality Technology, Report to SURFnet bv, Delft 2002, 72 p
- GISt Report No. 18, Vries, M.E. de, Ontsluiting van Geo-informatie via netwerken, Plan van aanpak, Delft, 2002, 17 p.
- GISt Report No. 19, Tijssen, T.P.M., Testing Informix DBMS with spatial data from the cadastre, Delft 2002, 62 p.
- 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.