Spatial DBMS testing
with data from the Cadastre and TNO-NITG

drs. T.P.M. Tijssen, drs. C.W. Quak,
prof. dr.ir. P.J.M. van Oosterom

GISt Report No. 7
Delft, March 2001

Summary

This report describes the set up and results of the functionality and performance tests of Oracle 8i spatial DBMS (object model). Two data sets are used. The first data set is geological data delivered by TNO-NITG. The second data set is cadastral (geometric and thematic) data delivered by the Dutch Cadastre. The tests with the cadastral data have also been executed within Ingres II DBMS (with OME/SOL) in order to have some reference figures for the performance tests. Further, this also enables the comparison of query results in the different DBMSs as the answers should be independent on the type of DBMS.

It turned out that benchmarking was a very difficult process in which two factors are important. First, on a multi-user machine only the necessary processes should be running and no other large processes (e.g. DBMSs) should be started even if they are not actively running. Second, big differences can exist between 'hot' and 'cold' situations. In order to obtain repeatable results is was decided to reboot the system and restart the DBMS before every test run.

The tests indicate that the functionality of Oracle 8i Spatial is sufficient for both Cadastre and TNO to start developments. The implemented functionality, which is quite complete, is close to production quality. Problems encountered in the tests which were sent to Oracle could be resolved on short notice. With respect to performance, the Oracle patch 0115 (non-production) is a big improvement, especially for the cadastral queries. In general the patched Oracle spatial performance is still less than the Ingres performance and the storage requirements are higher (overall a factor 2 for both aspects with the exception of long polyline queries, which are treated more efficiently by Oracle). This seems acceptable if seen in the perspective of future developments to improve performance by Oracle. Spatial clustering (via index organized tables) is the most urgent enhancement required. Future functional improvements in the area of topology support are especially important for the Cadastre, but are not to be expected before the year 2003 ('Oracle 10'?).

It should be noted that during the test period from December 2000 to March 2001, six different versions of Oracle Spatial were tested. This is proof of the rapid on-going developments, but also an indication that this report is a snapshot of the situation as of March 2001.

This project was executed by the TU Delft as a joint assignment of the Netherlands' Kadaster (Cadastre and Public Registers Agency) and TNO-NITG (Netherlands Institute of Applied Geoscience TNO - National Geological Survey).

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

Contents

List of Figures

List of Tables

1 Introduction

This report describes the research performed on the storage and retrieval of spatial data in a standard DBMS. The basic research question is the suitability of the current Oracle spatial DBMS. Both functionality and performance are tested with Oracle 8i, using cadastral data (supplied by the Dutch Cadastre) and geological data (supplied by TNO-NITG). For reference and checks of correctness the Ingres DBMS is used (cadastral data only). The Cadastre and TNO also stated a number of specific questions which have to be answered. These questions are not explicitly included in the report, but answered (and implicitly referred to) in the sections about data loading (3), data querying (4) and the JDBC connection (5). By loading the data sets in the database the questions related to the storage of spatial data can be answered. Then the functionality and reliability of spatial data retrieval is tested by means of various queries (which are the 'translation' of a number of the questions asked). Finally performance is measured and reported to make it possible for TNO and the Cadastre to draw the conclusion whether the current DBMS is fast enough for their applications.

1.1 Test configuration

All testing has been done on the database server of the GDMC (Geo-Database Management Center) in Delft. The main characteristics of this server are:

- Sun Enterprise E3500 server with Sun Solaris 7;
- two 400 MHz UltraSPARC CPUs with each 8 Mb CPU cache;
- 2 Gb main memory;
- two mirrored internal disks of 18.2 Gb, fiber channel; mainly used for system directories and swap space (all disks are 10000 rpm);
- two internal RAID0 sets, fiber channel (3 * 18.2 Gb disks per RAID set, effectively 2 * 52211532 Kb, striping only): mainly used for the DBMS workspaces, sorting;
- four external, hardware controlled, RAID5 sets in two Sun Storedge A1000 boxes, 24 Mb cache each (6 * 18.2 Gb disks per RAID set, effectively 4 * 86883062 Kb, striping with distributed parity) on two ultra SCSI adapters; mainly used for the DBMS data.

The most important software used is of course the Oracle DBMS, but also other software was used:

- Oracle 8i Enterprise Edition 8.1.7.0.0 Production and various patches;
- Ingres II (2.0/9808, su4.us5/00);
- (k)sh, (n)awk, perl, g(un)zip;
- gcc (GNU C++);

- Geo++, FME (as viewers).

The Oracle version history during the test period:

- Oct 12 2000 Oracle 8i (8.1.7.0.0) Production downloaded and installed;
- Jan 08 2001 regular Oracle Spatial patch libordsdo_0106;
- Jan 23 2001 1st version of special Oracle Spatial performance patch libordsdo_0115;
- Feb 16 2001 2nd version of special Oracle Spatial performance patch libordsdo_0115;
- Feb 27 2001 new version of regular Oracle Spatial patch libordsdo_0106;
- Mar 21 2001 3rd version of special Oracle Spatial performance patch libordsdo_0115.

Regular patch libordsdo_0106 is part of the 'normal' sequence of patches issued regularly by Oracle, and is available to all Oracle users. This patch solved some of the software bugs encountered during testing. Special patch libordsdo_0115 is not available for general use, it was created specifically for the Spatial DBMS testing project. This patch improves the performance of most spatial queries dramatically, especially queries using relatively long, polyline query geometries. The performance improvements of this special patch are scheduled to be included in Oracle 9i production.

In the remainder of this report the Oracle version is indicated whenever Oracle timings are reported. The term 'production' refers to Oracle 8.1.7 with the latest production patch (libordsdo_0106 of Feb 27 2001). The alias 'patch 0115' refers to Oracle 8.1.7 with the latest performance patch (libordsd0_0115 of Mar 21 2001).

One standard production version of Ingres was used and this included the standard functionality of OME/SOL (Object management Extention/Spatial Object Library) [1]. OME/SOL offers (non OpenGIS compliant) 2D spatial data types (point, line, polygon, box), spatial functions (distance, overlap, area) and spatial indexing (r-tree). Note that OME/SOL offers only 2D spatial data types. It is possible to develop 3D data types on the basis of OME. However, this has not been used during the tests reported in this document.

1.2 Report overview

The data sets used in the tests are geological and cadastral data. Section 2 describes the data models and gives the total number of records per table. Database creation, definition of the base tables, loading of the data and indexing are described in Section 3. This section also includes the elapsed time measurements of the different load phases. The various types of queries and the timing results can be found in Section 4. All results are given for Oracle, for Ingres only the querying of cadastral data is reported. Section 5 describes how to make a JDBC connection to Oracle 8i. In this case an example with geological data is given. During the spatial DBMS testing period several limitations and bugs were discovered in the Oracle software, these are described in Section 6. Finally, the conclusions and recommendations can be found in Section 7.

2 Overview of data sets

This section describes the data sets that are used in the project. An overview of the data and the way it is organized is given.

2.1 Geological data

The geological data of TNO-NITG was deliverd as a zip archive containing some subdirectories. Every subdirectory contains a separate data set. These data sets are:

9 key horizons These are depth grids (1121 x 1321 nodes) with a resolution of 250 m. For every pixel in the grid one value (depth) is given. Many pixels in the raster have a NOVALUE entry, meaning the horizon does not exist at this position. The depth attribute can be seen as the Z-coordinate of the points, making this set a 3-dimensional data set.

Fault lines This set consists of 3000 2-dimensional polylines with about 10 intermediate points per polyline.

Subcrop lines Approximately 200 polylines with between 50 and 100 points. Also 2-dimensional.

Wells The depths of 8 key horizons in 1000 wells. This means about 8000 points, with 3 dimensions.

Seismic interpretation picks These are interpreted depths of the picks on the key horizons. There are about 300,000 3-dimensional picks. At TNO-NITG more seismic data is available. Because this data is confidential it is not included in the tests.

2.2 Cadastral data

The Dutch Cadastre has 15 offices, in the tests data of 3 offices was used: Arnhem office (province of Gelderland), Rotterdam office and Zoetermeer office (together the province of Zuid-Holland). The test data set contains all geometric (LKI) and administrative (AKR) data of these provinces.

The geometric cadastral data is maintained by the LKI system[1], which stores the data in an Ingres database using OME/SOL (Object Management Extension/Spatial Object Library)[1, 7]. The geometric model has been published before quite extensively [5, 6, 7]. Since 1997 the geometric database keeps track of all changes over time, that is, it is a spatio-temporal database. The metric attributes of type point, polyline and box, are stored in the relational database together with the other attributes describing the measurement (date, accuracy, etc.). Every object is extended with two additional attributes: `tmin` and `tmax`. The objects are valid from and including `tmin` and remain

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

valid until and excluding `tmax`. Current objects get a special `tmax` value, `TMAX_VALUE`, indicating they are valid now. The geometric data model for the cadastral *parcel* layer is based on winged-edge topology [2] as described in [7]. There are seven base tables, all prefixed with `xfio_`, see Table 1.

Table name	Description	# Objects
xfio_boundary	parcel boundary	10,044,511
xfio_gcpnt	geodetic control point	60,986
xfio_line	topographic line (building)	3,502,313
xfio_parcel	parcel	3,820,699
xfio_parcelover	overflow record (> 1 enclave)	42,852
xfio_sympnt	topographic symbol	2,054,463
xfio_text	text	1,640,122

Table 1: Tables in the geometric model

Legal and other administrative data related to parcels is maintained by the AKR[2] system, which stores the data in an IDMS database on an IBM mainframe. This database will be referred to as the *administrative* database in contrast to the *geometric* database. The administrative base tables are prefixed with `mo_`, see Table 2.

Table name	Short description	# Objects
mo_obj_belemmering	legal notification	621,602
mo_object	cadastral object	2,386,641
mo_objectadres	cadastral object address	3,185,769
mo_ontstaan_uit	created from	1,359,908
mo_overgegaan_in	transferred into	596,695
mo_onzelfstandig_deel	3D part above/below other object	2
mo_recht	legal right	3,197,039
mo_rechtbelemmering	legal right notification	281,558
mo_reg_9	parcel/object surveying task	60,876
mo_rente	interest due to land consolidation	134,604
mo_subject	subject	2,401,682
mo_subject	subject (without duplicates)	2,264,036
mo_groepsrelatie	group relationship	22,722
mo_huwelijksrelatie	marriage relationship	1,081,956
mo_subjectrelatie	subject relationship	133,016

Table 2: Tables in the administrative model

The administrative data model is based on a few key concepts: *object*, *subject*, and *right*. Objects (parcels) and subjects (persons) have an n-to-m relationship via rights: a subject can have rights related to several objects (e.g. a person owning three parcels) and an object can be related to multiple subjects. Two examples of the latter: an object is owned by two partners or an object is leased by one subject to another subject. There are two types of subjects: *natural persons* and *non-natural persons* (organizations), having

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

some attributes in common, but also each having their own specific attributes. In turn, the objects can be one of three basic types: complete *ground parcels*, *part-of-parcels*, or *apartments*. In the Netherlands a part of a parcel can be sold, as an object, before it has been measured by the surveyor. These part-of-parcels again can be sold in part. This results in an hierarchy which is represented by a tree structure with the root representing the ground parcel.

The key entries to the database are a region (usually a rectangle, sometimes just one point), parcel number, address and (subject) names. Whenever possible data is clustered based on spatial location. This is obvious for the geometric data using the Spatial Location Code SLC [9]. However, this is also applied to the administrative data by clustering on parcel number, which contains a municipality and section code, or on postal/zip code. This enables spatial range queries to perform well in all situations including the integrated views. The other entries are supported by secondary indices (b-tree [3] or r-tree [4]), because they usually return only one or just a few results.

3 Loading the data

This section describes the process of loading the data into the database. First the parameters that were used when setting up the database are given. Then details on loading and indexing the geological and cadastral data are given.

3.1 Oracle database

During the test period the Oracle database was (re)created at least a dozen times, and not once with exactly the same parameters. The difficult part in database set up is to reconcile the requirements for data loading with those of day to day production. Loading (and to some extent also using) spatial data is still 'special' in the Oracle database because a number of limitations exist for this type of data which do not exist for more traditional (alphanumeric) data (see Section 6).

The procedure to create and use the Oracle test database is the following:

1. Create new file systems;
2. Create empty database;
3. Load data;
4. Run queries.

1) File systems

The databases involved in the tests are spread across 3 file systems, each file system is located on a separate RAID set. To avoid file system fragmentation the database is created in newly created file systems, using 'cooked files'. Use of raw devices for the databases instead of file systems was not tested. In the test set up the stripe size of the RAID sets, the operating system allocation unit and the database block size are identical: 8 Kb. This 1:1:1 relationship provides consistent and predictable behavior, although not necessarily the most efficient one. It might be that, for example, a stripe size of 64 Kb is more efficient, but this was not tested.

Tablespaces for temporary data, rollback and indices are located on a 3-disk (software) RAID0 set (file system /r02). AKR, LKI and TNO data are stored in separate tablespaces on two 6-disk (hardware) RAID5 sets (/r52 and /r54 file systems). So, the complete test database is spread over 15 disks, the RAID sets are also on separate I/O controllers. The data files of the test database, as viewed from the operating system:

```
/r02/oradata/kadtest/:
total 25180736
-rw-r-----   1 oracle   dba      1286144 Mar 15 12:10 control01.ctl
-rw-r-----   1 oracle   dba   8589942784 Mar 15 11:39 indx01.dbf
-rw-r-----   1 oracle   dba   2147491840 Mar 15 12:08 rbs01.dbf
-rw-r-----   1 oracle   dba   2147491840 Mar 15 11:46 temp01.dbf
```

```
/r52/oradata/kadtest/:
total 20693584
-rw-r-----   1 oracle   dba   4294975488 Mar 15 11:39 akr01.dbf
-rw-r-----   1 oracle   dba      1286144 Mar 15 12:10 control02.ctl
-rw-r-----   1 oracle   dba    109060096 Mar 15 11:39 drsys01.dbf
-rw-r-----   1 oracle   dba    268435968 Mar 15 11:41 redo01.log
-rw-r-----   1 oracle   dba    268435968 Mar 15 12:08 redo02.log
-rw-r-----   1 oracle   dba    268435968 Mar 15 11:22 redo03.log
-rw-r-----   1 oracle   dba   1073750016 Mar 15 11:46 tno.dbf
-rw-r-----   1 oracle   dba     10493952 Mar 15 11:39 tools01.dbf
-rw-r-----   1 oracle   dba   4294975488 Mar 15 11:39 users01.dbf

/r54/oradata/kadtest/:
total 17320752
-rw-r-----   1 oracle   dba      1286144 Mar 15 12:10 control03.ctl
-rw-r-----   1 oracle   dba   8589942784 Mar 15 11:39 lki01.dbf
-rw-r-----   1 oracle   dba    272637952 Mar 15 11:46 system01.dbf
```

The test data sets are stored in tablespaces TNO (1 Gb), AKR (4 Gb), LKI (8 Gb) and INDX (8 Gb: AKR and LKI indices). After loading and index creation these are not completely filled as shown by the following overview, which lists the free space in all tablespaces:

```
SQL> SELECT tablespace_name "Tablespace",
  2    COUNT(*) "Pieces",
  3    MIN(blocks) "Minimum",
  4    MAX(blocks) "Maximum",
  5    AVG(blocks) "Average",
  6    SUM(blocks)*8/1024 "Total_Mb"
  7    FROM sys.dba_free_space
  8    GROUP BY tablespace_name;

Tablespace     Pieces    Minimum    Maximum     Average    Total_Mb
---------- ---------- ---------- ---------- ----------- -----------
AKR                 1     283647     283647      283647     2215.99
DRSYS               1      12783      12783       12783       99.87
INDX                4        256       4095        1280       39.99
LKI                 1      46079      46079       46079      359.99
RBS                15        127     195840       17553     2056.99
SYSTEM              3         72       3734        1306       30.61
TEMP               10       2944     182271       26099     2038.99
TNO               224        127       2560         327      572.99
TOOLS               1       1279       1279        1279        9.99
USERS              18        256     287231       24761     3481.99

10 rows selected.
```

The sizing of AKR (requires less than 2 Gb) and TNO (requires less than 0.5 Gb) could be more economical than is currently the case. Overall the 21.2 mln LKI records require approximately 13 Gb disk space (data and indices), the 17.7 mln AKR records require 5 Gb disk space.

2) Database creation

The very first set up for the Oracle test database was created using the graphical `dbassist` tool (Oracle Database Configuration Assistant). After specifying a number of parameters and tablespaces this tool offers the possibility to save the database set up in a number of scripts. The initial set up as stored in the scripts was refined and augmented by editing the scripts. Finally the database is created by running the database creation scripts. Subsequent versions of the database were created by running the scripts again, after some additional editing. The most relevant part of the scripts can be found in Appendix A. It takes approximately 1 hour 45 minutes to create the Oracle database as used in the tests. 45 minutes for the core database (including LKI, AKR and TNO tablespaces), the remaining time for loading the various packages (e.g. Java, Intermedia, Timeseries, Spatial). Initially the size of the tablespaces was estimated conservatively, but all tablespaces were configured to grow dynamically. After loading the data for the first time the required sizes were known and the tablespaces were given their proper size. Because Oracle allocates and formats tablespaces to the specified size at creation time all space in the tablespaces is guaranteed to be contiguous disk space (except after the very first version of the database).

3) Loading data (general)

Some general remarks about data loading are included here, the details are described in subsections 3.2 and 3.3. The biggest demand for temporary and rollback space is while loading the data, specifically during index creation and table analysis. Exactly how much space is needed depends very much on the way and order in which the index creation and analyze commands are specified. The maximum space required is governed by the biggest 'object' in the database, in the test data set this is the `xfio_boundary` table. For example, if all indices are created for the `xfio_boundary` table and then the following command is issued, this command will take 3.5 hours to complete and require 4.5 Gb temporary tablespace (approximately the size of the table: 4.7 Gb):

```
analyze table xfio_boundary compute statistics
for table for all columns for all indexes;
```

To avoid such excessive use of space and to speed up the loading process the following sequence of commands was adopted after some experimentation:

```
analyze table xfio_boundary compute statistics;
create index xfiox_boundary_4 on xfio_boundary (l_obj_id)
tablespace indx nologging compute statistics;
create index xfiox_boundary.....
```

The analyze statement is given before index creation so the table and all columns are analyzed but not the indices because these do not exist yet. This takes 1 hour and requires 650 Mb temporary tablespace. Then use the option to analyze the indices while creating them, this is more efficient than a separate analysis of the indices (but the exact 'cost' in terms of temporary tablespace and time can no longer be determined because the analyze operation is now part of the create statement).

The amount of temporary space required also depends on the amount of memory available for sorting (the major use of tempspace is sorting). The Oracle default of 1 Mb is sufficient

for the type of queries executed during the tests, but while loading the data much more is required. In the final database set up 1 Gb of memory was reserved for sorting during LKI post-processing (index creation and table analysis of all LKI tables). Even so, with this amount of memory available, still 347 (8.8 %) out of the 3949 sorting operations during LKI post-processing do not fit in main memory and must reside to disk (with a maximum use of 650 Mb disk space as indicated before).

Looking at 'regular' memory usage (not while mass loading data) Oracle can hardly be called 'lean' compared to Ingres. The Oracle server process (instance) resulting from the final database set up has a size of 450 Mb, of which 240 Mb is used for I/O buffering (the database parameters used can be found in Appendix A). This is quite a lot compared to the 30 Mb Ingres uses. Oracle client processes as used in the tests are normally the same size as the server process, except in the case of 'clipping'. During the set of 27 queries which make use of the `sdo_intersection` operator the memory usage grows to a maximum of 1.8 Gb. It is not clear whether this is normal behavior or evidence of a memory leak.

The size of the rollback tablespace must be configured to accommodate at least the biggest spatial index, in the case of the test data set this translates to a size of at least 700 Mb (for the spatial indices of the `xfio_boundary` table). This amount of space is required while building the index. The biggest demand for rollback space comes from updating the `mo_subject` table. Updating this table is necessary to remove the double subjects after loading more than one office. The rollback space used during this operation can grow to more than 2 Gb.

Although hungry for resources, the Oracle DBMS did prove to be a stable and reliable environment. Not once during the test period did the database server or a client crash unexpectedly, or were there any problems with inconsistent or lost data, locking problems, etc. Of course, this is exactly what a DBMS is supposed to do (at a certain cost: some [Ingres] or quite a lot [Oracle] overhead).

3.2 Loading geological data

The loading of the geological data is script driven. The main script is `convert_all.sh` which is presented in Appendix F. Most data is delivered in Arc/Info 'generate' format. This format is converted to a format that can be read by the Oracle tool `sqlldr`. This data is then loaded into the database.

For raster data there are two alternatives for modelling the data. First, data can be modelled as a geo-image using the Oracle GeoImage library. Second, the raster can be converted to a big table with point objects, which can then be loaded into the database. The first option results in a small and efficient structure inside the database, however it is impossible to clip the raster data with a polygon. With the second option the data will take much more storage space inside the database, but clipping operations can be performed. Because clipping is an important operation for TNO the data is currently loaded as point data. Below a short description is given of the procedure to load the data sets into Oracle.

Type	File name	Table name	# Objects
faultlines	12br_br.asc	faultlines_12br_br	205
faultlines	19no_br.asc	faultlines_19no_br	574
faultlines	24tx_br.asc	faultlines_24tx_br	409
faultlines	29rijn_br.asc	faultlines_29rijn_br	143
faultlines	30df_br.asc	faultlines_30df_br	105
faultlines	40al_br.asc	faultlines_40al_br	198
faultlines	49bo_br.asc	faultlines_49bo_br	357
faultlines	54ze_br.asc	faultlines_54ze_br	1124
faultlines	59ro_br.asc	faultlines_59ro_br	986
faultlines	tze_br.asc	faultlines_tze_br	714
subcrops250	19no_sc.asc	subcrops250_19no_sc	51
subcrops250	24tx_sc.asc	subcrops250_24tx_sc	26
subcrops250	29rijn_sc.asc	subcrops250_29rijn_sc	29
subcrops250	30df_sc.asc	subcrops250_30df_sc	36
subcrops250	40al_sc.asc	subcrops250_40al_sc	78
subcrops250	49bo_sc.asc	subcrops250_49bo_sc	62
subcrops250	54ze_sc.asc	subcrops250_54ze_sc	61
subcrops250	59ro_sc.asc	subcrops250_59ro_sc	20
subcrops250	tze_sc.asc	subcrops250_tze_sc	85

Table 3: Fault line and subcrop tables

Subcrops and fault lines

The subcrop and fault lines data consists of polyline data in Arc/Info 'generate' format. Apart from oids, there are no attributes for this data. Currently for every input file a seperate Oracle table is created which contains one record for each polyline. In Table 3 an overview of the data loaded is given. The SQL statement to create these tables is:

```
create table tablename
(
   oid number(11) not null,
   geometry mdsys.sdo_geometry
);
```

In this case the geometry attribute always contains a geometry of type polyline.

After these tables have been created and loaded a database view on them is created which presents the data of all these tables in one virtual table. This view is created with the following SQL statement:

```
create view subcrops
as
          select oid, geometry, '19no_sc'   name from subcrops250_19no_sc
union all select oid, geometry, '24tx_sc'   name from subcrops250_24tx_sc
union all select oid, geometry, '29rijn_sc' name from subcrops250_29rijn_sc
union all select oid, geometry, '30df_sc'   name from subcrops250_30df_sc
union all select oid, geometry, '40al_sc'   name from subcrops250_40al_sc
union all select oid, geometry, '49bo_sc'   name from subcrops250_49bo_sc
union all select oid, geometry, '54ze_sc'   name from subcrops250_54ze_sc
```

```
union all select oid, geometry, '59ro_sc'   name from subcrops250_59ro_sc
union all select oid, geometry, 'tze_sc'    name from subcrops250_tze_sc;
```

Horizons

There are 9 raster (1121 x 1321) data sets with horizons. This data is loaded into 9 different tables. Each raster is loaded into a different table. Table 4 gives the file names, the database table names and the number of records in each raster. Although all rasters contain a regular grid, the actual number of records in each table differs. This is because entries that contain a NODATA value in the grid are not inserted into the database. During the construction of the test data set several map sheets of the data were undergoing some processing at TNO and were not included. This results in NODATA values in the southwest of the Netherlands.

File name	Table name	# Objects
19b_tert250.asc	horizons_19b_tert250	334720
24b_ucret250.asc	horizons_24b_ucret250	281088
29b_lcret250.asc	horizons_29b_lcret250	291413
30b_ujura250.asc	horizons_30b_ujura250	59466
40b_ljura250.asc	horizons_40b_ljura250	62509
49b_ltrias250.asc	horizons_49b_ltrias250	210632
50t_zech250.asc	horizons_50t_zech250	288955
54b_zech250.asc	horizons_54b_zech250	288945
59b_rotl250.asc	horizons_59b_rotl250	272993

Table 4: Horizon data

In the input raster the location of the points is implicit. The points in the database contain explicit geometry. Therefore the geometry of each point is calculated. In order to be able to query all horizon data with one query we have created a database view that unites the data of all horizons. This is done with the following SQL statement:

```
create view horizons
as select location, '19b_tert250' name from horizons_19b_tert250
union all select location, '24b_ucret250' name from horizons_24b_ucret250
union all select location, '29b_lcret250' name from horizons_29b_lcret250
union all select location, '30b_ujura250' name from horizons_30b_ujura250
union all select location, '40b_ljura250' name from horizons_40b_ljura250
union all select location, '49b_ltrias250' name from horizons_49b_ltrias250
union all select location, '50t_zech250' name from horizons_50t_zech250
union all select location, '54b_zech250' name from horizons_54b_zech250
union all select location, '59b_rotl250' name from horizons_59b_rotl250;
```

Borders

The border data contain the names and geometries of the Dutch provinces, and 12 rectangular map sheet tiles covering the Netherlands . This data contains closed polygons. The province polygons contain relatively many points per polygon (see Table 8), whereas the map sheets contain very few points (< 20). All provinces are loaded into one database table called `arcdata`. This table is created with the following SQL command:

```
create table arcdata
(
   oid number(11) not null,
   geometry mdsys.sdo_geometry,
   name varchar(30)
);
```

This data is not geological, but is used to generate interesting queries.

Seismic data

The file `nav_public.dat` contains the seismic data. This data is loaded into the Oracle database table `navseis`. This table has the following definition:

```
create table navseis
(
   lijnnaam varchar2(30),
   spnummer number(11),
   location mdsys.sdo_geometry
);
```

Currently it contains only the public seismic data (i.e. 66,558 points). When also classified data is included the size of this table will grow considerably.

Well pick data

The file `wells.dat` contains the co-ordinates of the well picks. The Oracle table has the following definition:

```
create table wellpicks
(
   location mdsys.sdo_geometry,
   putnaam varchar2(23),
   formatie_cd varchar2(7),
   diepte float,
   dikte float
);
```

Both the `diepte` (depth) and `dikte` (thickness) attributes can have NULL values. This table contains 2,790 elements.

Indices on geological data

On all spatial attributes available in the geological data r-tree indices are created. Indices (b-trees) are also created on non-spatial attributes that have an attribute on which selections can be performed. An overview of the indices that are created is given in Table 5 (a complete overview of all tables and indices can be found at the end of Appendix A). An example of a SQL index creation statement is given below. Because in Oracle the name of an index has a maximum length the name of the index is a truncated version of the table name.

Table	Index name	Index
faultlines_*	faultlines_*_1	r-tree (geometry)
subcrops250_*	subcrops250_*_1	r-tree (geometry)
horizons_*	horizons_*_1	r-tree (location)
navseis	navseis_1	r-tree (location)
navseis	navseis_2	b-tree (lijnnaam)
navseis	navseis_3	b-tree (spnummer)
wellpicks	wellpicks_1	r-tree (location)
wellpicks	wellpicks_2	b-tree (formatie_cd)
arcdata	arcdata_idx	r-tree (geometry)

Table 5: Indices created on geological data

```
create index HORIZONS_19B_TER_1 on HORIZONS_19B_TERT250(LOCATION)
indextype is mdsys.spatial_index
parameters ('initial=4m next=4m pctincrease=0');
```

The loading of all geological data, creation of indices and statistics calculations take approximately 1 hour.

3.3 Cadastral data

The loading of data takes a number of steps. These steps depend on the source (format) of the data and the destination of the data. The source of the LKI data is a selection from the query tool database. The source of the AKR data is 'Mass Output' (MO), a standard product to deliver legal/administrative information to the municipalities. This MO municipality delivery is also causing the duplicate subjects, see Table 2. In general the following aspects can be discerned with respect to cadastral data loading:

1. Define tables (with, including or additionally, primary storage structures);
2. Prepare source data for loading;
3. Load data;
4. Define secondary indices (b-tree, r-tree, ..);
5. Define geom/admin views;

Appendix B contains samples of the table definitions, primary storage structures (if possible) and secondary indices for the cadastral data in Oracle. The same is done in Appendix C, but now for Ingres.

For many operations it proves to be very beneficial (in terms of query performance) to cluster the data on disk using an index. This means that database records that are close to each other according to the index are located in the same block (or adjacent blocks) on disk. In the project all data is clustered on a spatial attribute if possible. All objects that have a geometric attribute are clustered on this attribute. Objects without such an attribute (all administrative data) are clustered on attributes that have a spatial meaning,

i.e. persons are clustered on the zipcode of their address. In Oracle this clustering of data is called 'index organized tables'. In the current version of Oracle it is unfortunately not possible to use spatial indices for index organized tables.

3.3.1 Loading LKI data

Loading LKI data into Oracle is done in various steps. After creating the tables the original LKI ASCII files are converted by a Perl script into a format `sqlldr` can understand. The same Perl script also generates the `sqlldr` control file. Then these files (including geometry) are loaded into the Oracle database. This is done without the 'direct path' option, which is not possible in case the table contains (spatial) objects.

Loading LKI data is done with a shell script (`lki_load.sh`). The structure of this script is depicted in Figure 1. The load process can be subdivided in three steps:

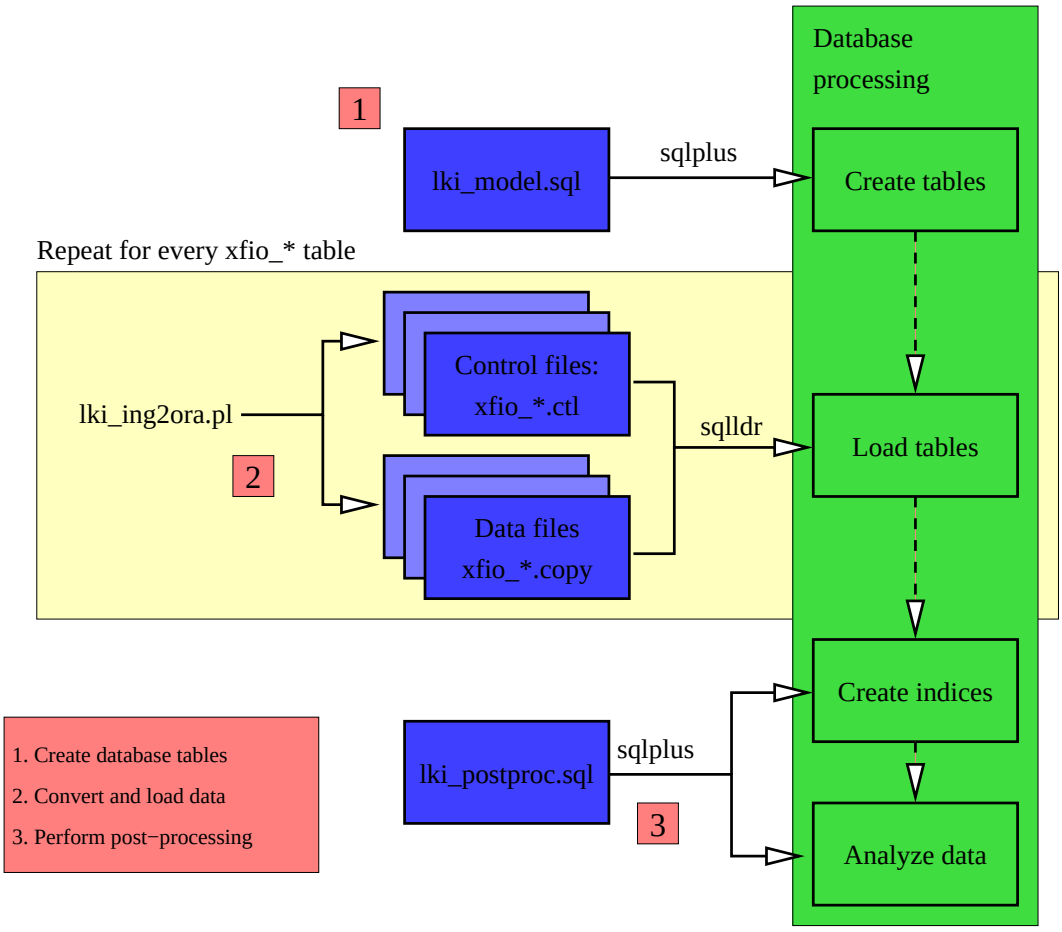

Figure 1: LKI load process (script `lki_load.sh`)

1. In the first step, the database tables are created. This is done by executing the SQL script `lki_model.sql`.

2. In the second step, which is performed separately for every database table, LKI data is converted to a file that can be loaded with the Oracle `sqlldr` tool. Also a

`sqlldr` control file is generated. If these two files are created, the `sqlldr` program is called and the data is loaded. The load time for the LKI tables shown in Table 6 is the elapsed time of this step.

3. When all data is loaded the SQL script `lki_postproc.sql` is called that creates indices on the tables and computes statistics for the query optimizer.

Table name	Oracle size Mb	Ingres size Mb	Oracle load time	Ingres load time
xfio_boundary	4770	2527	04:27	18:16
xfio_gcpnt	9	9	00:01	00:06
xfio_line	1332	532	01:13	05:04
xfio_parcel	1151	762	01:11	06:26
xfio_parcelover	10	5	00:01	00:04
xfio_sympnt	204	217	00:15	02:59
xfio_text	170	182	00:13	02:26
mo_obj_belemmering	69	119	00:01	00:04
mo_object	381	683	00:13	00:26
mo_objectadres	436	747	00:07	00:31
mo_ontstaan_uit	165	266	00:03	00:12
mo_overgegaan_in	72	130	00:01	00:05
mo_onzelfstandig_deel	0	0	00:00	00:00
mo_recht	507	847	00:07	00:34
mo_rechtbelemmering	33	54	00:01	00:02
mo_reg_9	6	11	00:00	00:00
mo_rente	14	26	00:00	00:01
mo_subject	397	659	00:11	02:38
mo_groepsrelatie	2	3	00:00	00:00
mo_huwelijksrelatie	81	140	00:01	00:19
mo_subjectrelatie	10	18	00:00	00:01

Table 6: Cadastral table sizes and load times (timings in hh:mm)

Loading LKI data into Ingres has been done with the very slow `copyrel` program, because the Ingres SQL `copy` command cannot be used to copy ASCII source data in spatial data types. The Cadastre uses a binary 'dump' of the Ingres data from the production databases (provincial offices) to load the query tool database in an efficient way. On machine `CSU004` at the Dutch cadastre it takes about 3 hours and 11 minutes to load the seven `xfio_` tables of the three cadastral offices. However, during this research we only used ASCII data to load the databases (both Oracle and Ingres).

It must be remarked that the table definitions in Oracle and Ingres are slightly different. Oracle has NULL values as the nodata value in the tables, these attributes in Ingres are filled with a default value. This might be the reason why the `mo_` tables in Ingres occupy more space than in Oracle, whereas it is the other way around with the `xfio_` tables. A complete overview of the sizes of all tables and indices in the Oracle database, with the commands to generate the listings, can be found in Appendix A.

The size of tables and indices in Ingres is determined with the command `help table` and/or `help index`, which gives the number of pages (and the page size). Note that all sizes in Tables 6 and 7 are given in Mb, that is 1024*1024 bytes. Note also that the Ingres indices on columns with (large) varchar attributes are defined with compression. Otherwise, a lot of disk space is waisted (which makes things slow).

Index name	Oracle size Mb	Ingres size MB	Oracle create time	Ingres create time
xfiox_parcel_0	262	75	01:25	00:07
xfiox_parcel_1b	75	47	00:04	00:04
xfiox_parcel_3	121	108	00:05	00:05
xfiox_boundary_0	688	197	04:18	00:16
xfiox_boundary_0b	688	197	04:30	00:19
xfiox_boundary_1b	197	123	00:12	00:12
xfiox_text_0	112	32	00:27	00:02
xfiox_text_2	28	41	00:02	00:02
idx_mo_object_1	135	67	00:03	00:04
idx_mo_subject_1	-	45	-	00:03
idx_mo_subject_2	99	125	00:07	00:07
idx_mo_subject_3	98	100	00:04	00:03
idx_mo_recht_1	115	90	00:04	00:05
idx_mo_recht_2	77	63	00:03	00:05

Table 7: Cadastral index sizes and creation times (timings in hh:mm)

Analyzing the LKI tables and indices in Ingres takes 5 hours 2 min. For Oracle a comparable figure is difficult to give because most analysis (all non-spatial indices) is done while creating the index. The remaining analyze statements (for tables and spatial indices) take approximately 2 hours.

3.3.2 Loading AKR data

The process of loading AKR data is shown in Figure 2. The seven steps of this load process are steered by the script `convert_akr.sh`. Below we shortly describe the steps:

1. In step 1, the C-program `readakr` is called with the option `-makescripts`. This program reads the `used_definitions.dat` file, which contains a description of the format in which the AKR data is delivered. This description is then converted into SQL script `model.sql`, which contains the data definitions for the database.

2. The `model.sql` script is executed, this creates the tables needed for AKR.

3. Then the `readakr` program is called again, this time with the `-makedata` option. This time the MO-files, which contain the data, are fed to the program and are converted to copy files, which can be read by Oracle. This step is not shown in Table 6 (it is equal for Oracle and Ingres) and is done for all tables at the same time. This step takes about 3 hours and 8 min. Apart from the copy files, `readakr` also creates the necessary control files for Oracle.

4. In this step, the `model_load.sh` script is used to load the data. The loading itself is done with the Oracle `sqlldr` tool. Because AKR does not contain any spatial data, the 'direct path' loading option of Oracle is used, which results in faster loading. The Oracle load time for the AKR tables shown in Table 6 is the elapsed time of `sqlldr`.

5. After the data has been loaded, some post-processing needs to be done. This is done in two SQL scripts. `Delete_double_subject.sql` removes the duplicate subjects from the `mo_subjects` table. Then the `model_modify` script is used to create primary indices on the data.

6. After that, the generated (in step 1) script `model_index.sql` is called, which creates secondary indices on the tables.

7. Finally, statistics for the query optimizer are collected for all tables with `analyze table` commands in akr_postproc.sql.

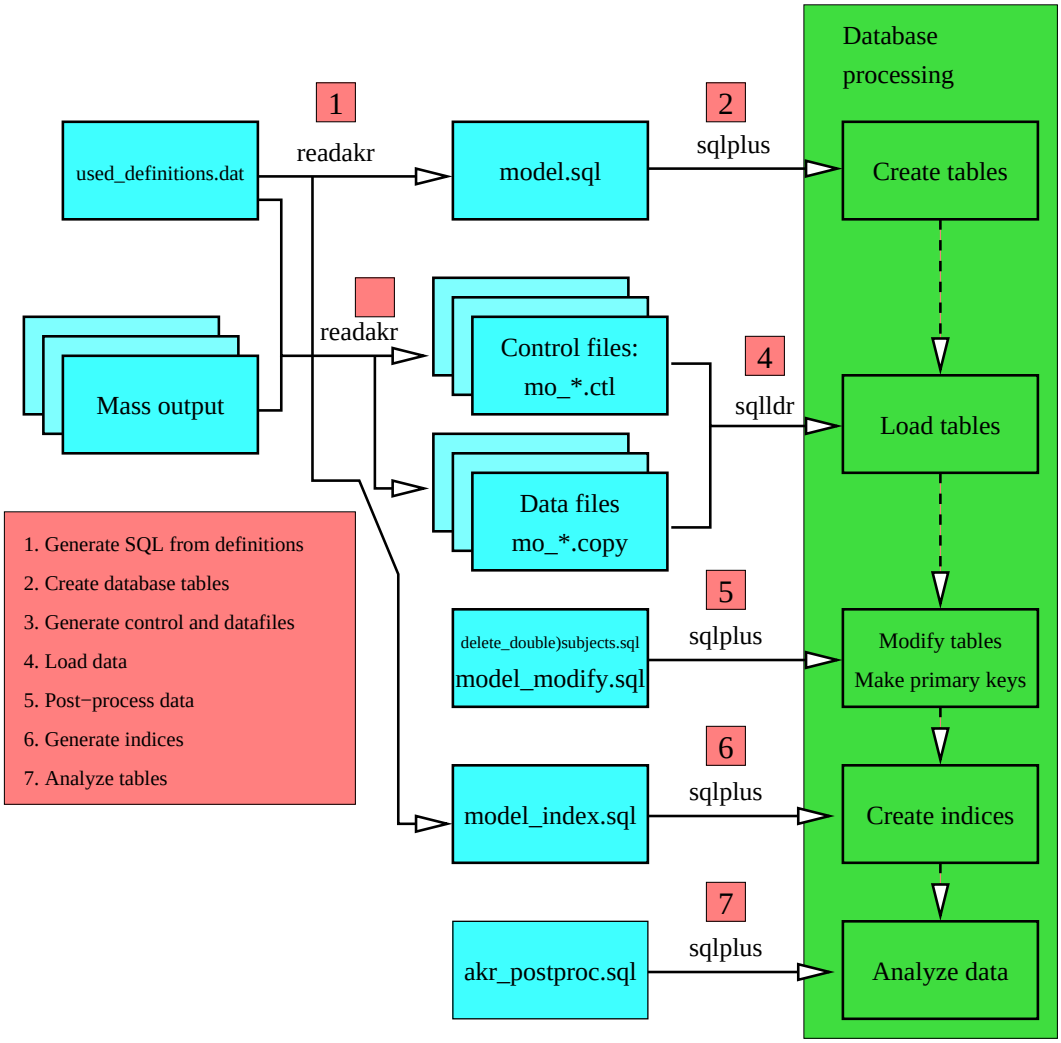

Figure 2: AKR load process (script `convert_akr.sh`)

Loading AKR data into Ingres has been done in three steps, first the MO source files are pre-processed by program `readakr` to prepare ASCII files ready to be loaded into the

database. The second step is using the Ingres SQL `copy` command to load the data into a temporary table (heap storage structure without data compression). The third step consists of inserting the data into the final tables (with b-tree storage structure and data compression). Table 6 does show the sum of the second and third step, including two count(*) queries per loaded table (temporary and final table).

In the query tool environment several special actions are performed besides loading and creating indices:

1. remove double subjects;
2. correct initial tmin creation time in LKI data;
3. set the abox for the `xfio_boundary` table.

The source of the AKR data is Mass Output for the municipalities. The result is that subjects having rights to objects in multiple municipalities are true duplicates in the query tool database and must be removed. In Oracle this is done with the following query:

```
delete from mo_subject m1
where m1.subject_id in
      (select m2.subject_id
       from mo_subject m2
       where m1.subject_id=m2.subject_id and m1.rowid<m2.rowid);
```

The second and third special actions are needed for the correct interactive geographic operation of the query tool. These steps are not part of the Oracle Spatial tests and are therefore not described in this report.

Some queries were performed to check if the cadastral data is correct and internally consistent. For example, do all subjects (`gerechtigde`) mentioned in the `mo_recht` table also occur in the `mo_subject` table:

```
select gerechtigde, x_akr_objectnummer, stuk_wijziging, soort_recht,aandeel
from mo_recht
where gerechtigde not in
   (select subject_id from mo_subject);\p\g

Executing . . .

(0 rows)
```

This is correct, all subjects can be found in the `mo_subject` table. A requirement for using an Oracle 'index organized table' is that the table does have a primary key (unique combination of attributes). It was not easy to find these keys. Below a query to check if it would be possible at all to find such a primary key in the `mo_right` table (note the relatively rare having-clause in the query):

```
select aantal=count(*),
klant_id, municip, osection, parcel, pp_i_ltr, pp_i_nr, ontvangstdatum,
volgnummer, was_wordt, stuk_wijziging, aanbieder, verlijdensdatum,
soort_stuk, gerechtigde, soort_recht, aandeel, stuk_vest_recht, ontv_datum,
ind_einde, aandeel_medeger_groep, ind_split
```

```
from mo_recht
group by
klant_id, municip, osection, parcel, pp_i_ltr, pp_i_nr, ontvangstdatum,
volgnummer, was_wordt, stuk_wijziging, aanbieder, verlijdensdatum,
soort_stuk, gerechtigde, soort_recht, aandeel, stuk_vest_recht, ontv_datum,
ind_einde, aandeel_medeger_groep, ind_split
having count(*) > 1;

....

(383 rows)
```

This is not correct as all attributes of the `mo_right` table are used in the query above. This table should not contain duplicates. It implies that a primary key does not exist and therefore it is not possible to store the table in Oracle as an 'index organized table'. In Ingres it is no problem to execute a `modify to btree` command for this table and achieve the preferred physical clustering.

3.3.3 Defining views

After the spatial (LKI) and thematic (AKR) data have been loaded into the database, the two separate data sets are combined with the use of views. Within the database model of the Cadastre, views are an important modelling concept. The ideas behind the views are published in [8], below we give two examples of views. The view definitions are created in the scripts `akr_model_view.sql` and `hist_view`. Because views do not manipulate real data, the creation of views virtually takes no time and timing results are not included in this report.

The first view definition creates a 'current' view on the data in `lki_parcel`, whereas the base tables also include historic parcels:

```
create view lki_parcel(
        ogroup, object_id, slc, classif, location, geo_color,
        z, d_location, rotangle, accu_cd, oarea, geo_bbox, object_dt,
        tmin, tmax, sel_cd, source, quality, vis_cd, akr_area,
        municip, osection, sheet, parcel, pp_i_ltr, pp_i_nr,
        l_num, line_id1, line_id2, x_akr_objectnummer
) as select
        ogroup, object_id, slc, classif, location,
        (3+mod((tmin+tmax),35)),
        z, d_location, rotangle, accu_cd, oarea, bbox, object_dt,
        tmin, tmax, sel_cd, source,
        quality, vis_cd, akr_area,
        municip, osection, sheet, parcel, pp_i_ltr, pp_i_nr,
        l_num, line_id1, line_id2, x_akr_objectnummer
from xfio_parcel
where
   ogroup=46 and
   (tmin <= 313372800) and ((313372800 < tmax) or (tmax=0));
```

The second view presented combines data from AKR (`mo_object`) and LKI (`lki_parcel`). These two tables are joined using an extra table (`object_parcel`) into one integrated view `akr_object`:

```
drop view akr_object;
create view akr_object as
select
   o.osection,o.parcel,o.municip,o.pp_i_ltr,o.pp_i_nr,o.klant_id,
   o.ontvangstdatum, o.volgnummer, o.was_wordt, o.stuk_wijziging,
   o.aanbieder, o.verlijdensdatum, o.soort_stuk, o.stuk_vestiging,
   o.ontdat_st_vest, o.grootte, o.indicatie_grootte_geschat, o.bladnr,
   o.bladvolgnr, o.ruitletter, o.ruitnummer, o.x, o.y, o.beb_code,
   o.soort_cult, o.ind_meer_cult, o.koopsom, o.koopjaar,
   o.ind_meer_kad, o.omschr_deelp,
   o.ind_verv_per, o.belast_plicht,
   p.slc,p.object_id,p.classif,p.location,p.d_location,
   p.rotangle,p.geo_bbox, 3 as geo_color,p.tmin,p.tmax,
   p.l_num,p.line_id1,p.line_id2, p.x_akr_objectnummer, p.ogroup
from
   mo_object o,
   lki_parcel p,
   object_parcel op
where
   o.x_akr_objectnummer = op.x_akr_objectnummer and
   op.g_akr_objectnummer = p.x_akr_objectnummer;
```

4 Querying the data

4.1 Geological data

TNO has specified 5 queries which will be treated in various subsections. After the 5 TNO queries some other interesting queries will be given. As described in Section 3, the amount of data is much less, compared to the cadastral data. The focus is on the functionality of the queries.

Before we go to the queries themselves, first the query regions are described. Two types of query polygons are given. The first set consists of the provinces of the Netherlands. The second set are the TNO-NITG map sheets. Tables 8 and 9 show some details of these query polygons obtained with the following query:

```
select name, SDO_GEOM.SDO_AREA(geometry, 0.5)/1000000 pol_area_km2,
SDO_GEOM.SDO_AREA(SDO_GEOM.SDO_CONVEXHULL(geometry,0.5),0.5)/1000000 con_area_km2,
SDO_GEOM.SDO_LENGTH(geometry, 0.5)/1000 length_km from arcdata q
where not name like 'kb%';
```

Name	**# Points**	**Polygon area** km^2	**Convex area** km^2	**Length** km
brab	3112	5084.79116	6599.88563	478.928363
drent	2571	2680.46740	3135.76174	277.889299
flevo	464	2412.29709	2781.24637	242.883851
fries	1165	6110.19406	7118.77525	384.639329
gelder	6004	5144.94571	6988.96215	559.695032
gron	948	3054.94484	4121.86328	308.014330
limb	1076	2208.31876	3759.81705	452.287599
noordh	1190	4435.24737	5521.19538	383.748061
overij	4490	3420.08511	4663.83646	419.321524
utr	3913	1434.25714	1889.27545	274.533136
zeel	913	3043.20128	3403.20445	260.617166
zuidh	1733	3712.59008	4436.74214	349.424243

Table 8: The polygon geometries of the query provinces

In summary, there are 24 query polygons. The total numbers of geological data records (see Section 3.2) are: 4,815 fault lines (polylines), 2,090,721 key horizon points (3D points), 66,558 seismic interpretation picks (2D points), 448 subcrop lines (2D polylines) and 2,790 well picks (2D points). Figure 3 shows the horizon data (note the missing map sheets) and the query polygons (both map sheets and provinces).

4.1.1 Horizon queries

The first query is to find all depth related coordinates of a specified key horizon within a given query polygon. This query was solved by an aggregate query to find these answers for all horizons and query polygons at once. This is based on a spatial join of the `horizons` view and the table with query geometries, `arcdata`.

Name	# Points	Polygon area km^2	Convex area km^2	Length km
kb1	16	4500	4500	270
kb2	16	4500	4500	270
kb3	16	4500	4500	270
kb4	16	4500	4500	270
kb5	16	4500	4500	270
kb6	16	4500	4500	270
kb7_8	23	9000	9000	390
kb9	16	4500	4500	270
kb10	15	4500	4500	270
kb11_12	20	7500	7500	350
kb13_14	24	9000	9000	390
kb15	12	3000	3000	220

Table 9: The polygon geometries of the query sheets

Initially the model for horizons was based on a single geometry attribute containing only one 3D point. On all horizon base tables an r-tree index was created without problems after the geometry metadata table was filled with an entry for each of the horizon base tables. The table below shows a small part of the contents with 3D points (note the value of the geometry type GTYPE=3001, indication a 3D point):

```
LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------
SDO_GEOMETRY(3001, NULL, SDO_POINT_TYPE(223875, 598875, -786.9034), NULL, NULL)
SDO_GEOMETRY(3001, NULL, SDO_POINT_TYPE(224125, 598875, -775.4023), NULL, NULL)
SDO_GEOMETRY(3001, NULL, SDO_POINT_TYPE(224375, 598875, -767.4771), NULL, NULL)
```

The query uses the `horizon` view (which 'contains' all horizon base tables) and performs a spatial join with the query table `arcdata`. The result should be the number of horizon points per key horizon and per query polygon.

```
select count(*), h.name, q.name
from horizons h, arcdata q
where mdsys.sdo_relate (h.location,q.geometry,
'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
group by h.name, q.name;

ERROR at line 1:
ORA-13226: interface not supported without a spatial index
ORA-06512: at "MDSYS.MD", line 1723
ORA-06512: at "MDSYS.MDERR", line 8
ORA-06512: at "MDSYS.SDO_3GL", line 57
ORA-06512: at line 1
```

The Oracle error message indicates that it did not recognize the spatial index on the horizon base tables. Probably because they are now accessed via views instead of direct access to the table. Therefore, the second attempt for this first query does not use the `horizon` view, but directly uses one of the horizon base tables:

```
select count(*), q.name
from horizons_19b_tert250 h, arcdata q
```

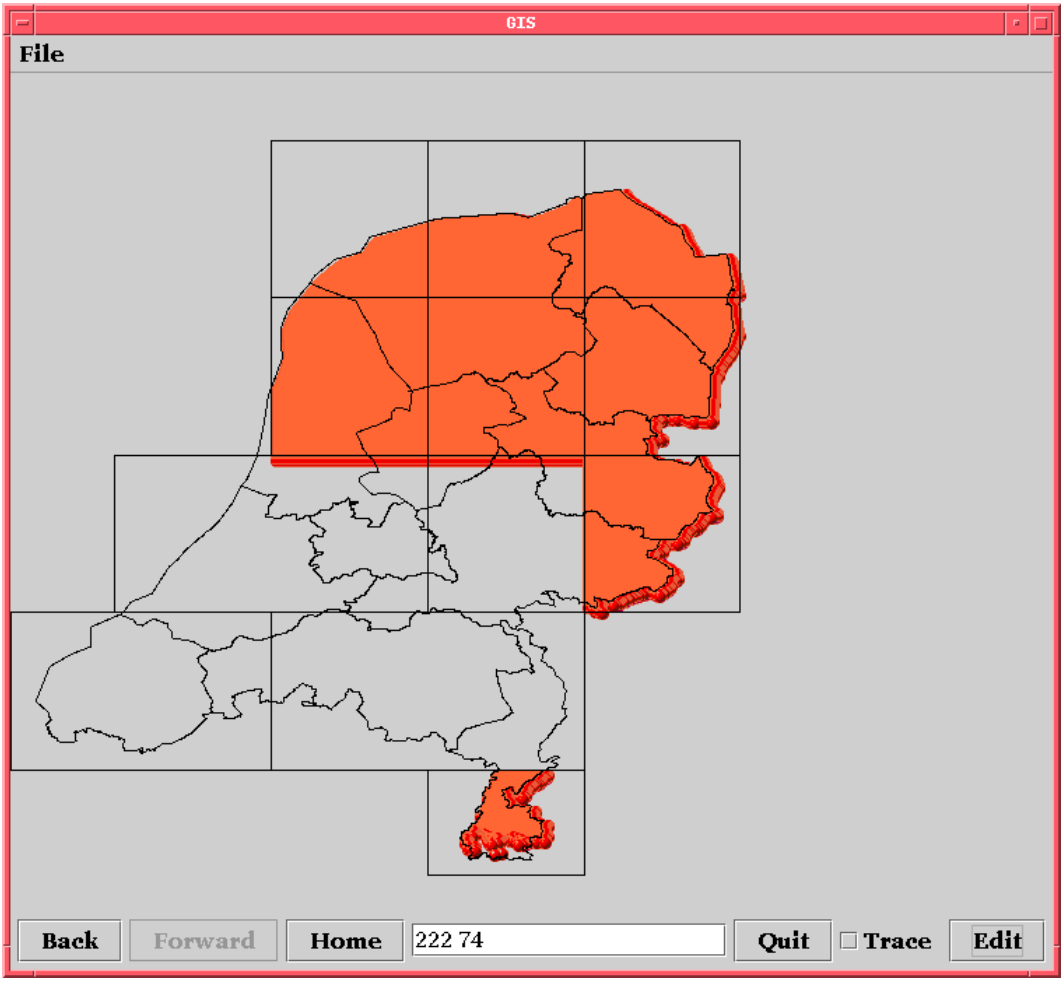

Figure 3: Overview of geological selection/query geometries

```
where mdsys.sdo_relate (h.location,q.geometry,
'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
group by q.name;

ERROR at line 1:
ORA-29902: error in executing ODCIIndexStart() routine
ORA-13207: incorrect use of the [More than 2D not supported] operator
ORA-06512: at "MDSYS.SDO_INDEX_METHOD", line 84
ORA-06512: at line 1
```

This Oracle message indicates that it does not like the mixing of 3D points and 2D polygons as operands for the `mdsys.sdo_relate` operator. An attempt was made to solve this by switching the order of the operands:

```
select count(*), q.name
from horizons_19b_tert250 h, arcdata q
where mdsys.sdo_relate (q.geometry,h.location,
'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
group by q.name;

ERROR at line 1:
ORA-13050: unable to construct spatial object
```

```
ORA-06512: at "MDSYS.SDO_3GL", line 41
ORA-06512: at "MDSYS.MD2", line 722
ORA-06512: at "MDSYS.SDO_3GL", line 105
ORA-06512: at line 1
```

The result was a different Oracle error message, so is was impossible to get results in this way. This can be considered a flaw in the Oracle operators (close to a bug). Instead of using operators an attempt was made to use a geometry function, because these were used before with mixed 2D and 3D operands:

```
select count(*), q.name
from horizons_19b_tert250 h, arcdata q
where sdo_geom.relate (h.location,'anyinteract',q.geometry,0.5) = 'TRUE'
group by q.name;

select count(*), h.name, q.name
from horizons h, arcdata q
where sdo_geom.relate (h.location,'anyinteract',q.geometry,0.5) = 'TRUE'
group by h.name, q.name;
```

Both queries (on horizon base table and on horizon view) are accepted by Oracle, but are very, very slow (too long to wait for the result). This can be explained by the fact that a geometry function never uses an index and is performing a point-in-polygon test for every combination of a key horizon point and query polygon. Also the query polygons of the provinces contain a lot of points in their definition (up to 6000).

Finally, the 2D/3D mix of operands problem is avoided by storing the key horizon points as GTYPE=2001 (indicating a 2D point) instead of GTYPE=3001 (a 3D point). This was also done in the geometry metadata table before the indices were recreated. This would then result in 2D r-tree indices for the key horizons. However, the actual geometry was filled with 3D points as can be seen below:

```
LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z), SDO_ELEM_INFO, SDO_ORDINATES)
--------------------------------------------------------------------------------
SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(228375, 623125, -613.5092), NULL, NULL)
SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(228625, 623125, -636.7278), NULL, NULL)
SDO_GEOMETRY(2001, NULL, SDO_POINT_TYPE(228875, 623125, -660.1608), NULL, NULL)

select count(*), q.name
from horizons_19b_tert250 h, arcdata q
where mdsys.sdo_relate (h.location,q.geometry,
'mask=ANYINTERACT querytype = WINDOW') = 'TRUE'
group by q.name;

  COUNT(*) NAME
---------- ------------------------------
     42846 drent
     25297 flevo
     97181 fries
     16196 gelder
     48596 gron
      8325 limb
     48747 noordh
     47312 overij
     17849 kb1
```

```
38441 kb2
32951 kb3
68885 kb4
72000 kb5
54786 kb6
  240 kb7_8
  240 kb9
40911 kb10
 8417 kb15
```

Note that not all query regions contain points from this key horizon. This indeed works, but still takes a relatively long time (about 35 min) taken into account that only 1 of the 9 horizons is used and that the operator is supported by an r-tree index to avoid unnecessary point-in-polygon computations. However, due to the large number of points per polygon this still takes quite some time.

In order to analyze the time spent a little better the query has been repeated on a query polygon per query polygon base (instead of all query polygons in one spatial join). The result is shown in Table 10. The columns labelled 'Prod 1' and 'Patch 1' show the the query timings on the production and patched versions of Oracle for one single horizon ('horizon_19b_tert250'). Rows with timing values 00:00 correspond to a query polygon without horizon data. It is remarkable that there is no siginificant time difference between the production version and the patched version of Oracle. The columns labelled 'Prod all' and 'Patch all' show the the query timings on the production and patched versions of Oracle for the table containing all horizons (`t_horizons`). In the production version of Oracle, querying all layers is hardly slower than querying a single layer. This is not true for the patched version which is much slower!

Due to the nature of the data and the query polygons (defined by many points) also some alternative index types are tested: fixed grid and hybrid indexing (quadtree-like). This has been done on the table containing all horizons 't_horizons', but only with one query polygon 'drent'. After some experiments a little better solution was found by creating a fixed grid index of level 10 or 11. Note that the size of the 'spatial indices' for the `arcdata` table grows fast in contrast to the 'spatial indices' size of the `horizon` table. For comparison reasons we also did use the r-tree and also analyzed this with the analyze commands similar to the other indices (as shown in the SQL script below). It is strange that the r-tree query now takes about 32 min as before this took less than 4 min in the Oracle production version. The difference is that in the later case more analyze commands were performed. So, we did expect either better or equal results, but we obtainted worse results with the r-tree after the analyze statements. Apparantly the analyze statements influence the query plan generated by the query optimizer with a much less efficient plan as a result.

Table 11 shows the results of the query timings in the last column (other columns contain index sizes and index creation times). A quadtree index ('fixed7' to 'fixed11', number indicating the level, and 'hybrid) has three parts: a 'spatial index table' with 'SDO_ROWID' (rowid of the geometry in the base table), 'SDO_CODE' (grid cell id), and 'SDO_STATUS' (status can be 'I' inside or 'B' boundary indicating if the grid cell is inside or on the boundary of the geometry) and two b-tree indices on the first two columns of this 'spatial index table'.

Name	Prod 1	Patch 1	1 horizon count	Prod all	Patch all	All horizons count
brab	00:02	00:01	0	00:07	00:01	0
drent	04:12	04:12	42846	03:47	32:51	330737
flevo	00:53	00:54	25297	00:41	04:35	120913
fries	05:20	05:13	97181	05:08	34:24	623712
gelder	05:04	05:04	16196	03:47	28:57	87359
gron	02:02	02:04	48596	01:49	15:54	343871
limb	00:19	00:19	8325	00:18	01:05	28434
noordh	02:27	02:24	48747	01:58	13:58	264759
overij	07:33	07:34	47312	08:58	50:39	289725
utr	00:00	00:00	0	00:10	00:00	0
zeel	00:00	00:00	0	00:03	00:00	0
zuidh	00:00	00:00	0	00:03	00:00	0
kb1	00:01	00:01	17849	00:22	00:05	115755
kb2	00:01	00:02	38441	00:49	00:11	257924
kb3	00:01	00:01	32951	00:38	00:10	227443
kb4	01:08	01:08	68885	01:05	06:36	382627
kb5	01:11	01:12	72000	01:20	07:22	423632
kb6	00:54	00:54	54786	01:10	07:19	419441
kb7_8	00:00	00:00	240	00:16	00:00	1707
kb9	00:00	00:00	240	00:11	00:00	1503
kb10	00:02	00:01	40911	00:33	00:09	231879
kb11_12	00:00	00:00	0	00:00	00:00	0
kb13_14	00:00	00:00	0	00:02	00:00	0
kb15	00:00	00:00	8417	00:04	00:01	28810

Table 10: Counting the number of points in a polygon (timings in mm:ss)

```
drop index arcdata_idx force;
create index arcdata_idx on arcdata(geometry)
  indextype is mdsys.spatial_index
  parameters ('sdo_level=10 initial=1m next=1m pctincrease=0');
analyze table arcdata compute statistics for table
  for all columns for all indexes;
```

Index type	Arcdata create	Index size Mb	Horizon create	Index size Mb	Query time
fixed7	0:00:47	0+0+0	0:12:23	47+61+50	0:06:25
fixed8	0:02:40	1+2+2	0:12:24	47+61+50	0:03:23
fixed9	0:11:08	6+7+6	0:12:35	49+64+50	0:02:09
fixed10	0:39:23	98+29+23	0:12:34	49+64+50	0:01:29
fixed11	2:48:11	90+116+91	0:12:45	49+64+50	0:01:13
hybrid	0:00:49	0+1+0	0:15:23	63+77+50	0:06:29
r-tree	0:00:01	0	0:32:57	146	0:32:35

Table 11: 'Drent' window query with different indices (timings in h:mm:ss)

```
analyze table arcdata_idx_fl10$ compute statistics for table
  for all columns for all indexes;
grant select on arcdata to public;

drop index t_horizons_idx force;
create index t_horizons_idx on t_horizons(LOCATION)
indextype is mdsys.spatial_index
parameters ('sdo_level=10 initial=1m next=1m pctincrease=0');
analyze table t_horizons compute statistics for table for all columns
  for all indexes;
analyze table t_horizons_idx_fl7$ compute statistics for table for all columns
  for all indexes;
GRANT SELECT on t_horizons to public;

select count(*) from t_horizons h, arcdata q
where mdsys.sdo_relate (h.location,q.geometry,
'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' and q.name = 'flevo';

describe arcdata_idx_fl7$
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SDO_CODE                                           RAW(12)
 SDO_ROWID                                          ROWID
 SDO_STATUS                                         VARCHAR2(1)

select * from arcdata_idx_fl7$

SDO_CODE                 SDO_ROWID          S
------------------------ ------------------ -
35F4                     AAAHlrAAJAAAZuDAAX I
35F8                     AAAHlrAAJAAAZuDAAX I
35FC                     AAAHlrAAJAAAZuDAAX I
3610                     AAAHlrAAJAAAZuDAAX B
3614                     AAAHlrAAJAAAZuDAAX B
361C                     AAAHlrAAJAAAZuDAAX B
3634                     AAAHlrAAJAAAZuDAAX B
3640                     AAAHlrAAJAAAZuDAAX I
3644                     AAAHlrAAJAAAZuDAAX I
..... many more rows
6320                     AAAHlrAAJAAAZuDAAX B

16379 rows selected.
```

An attempt was made to use the `horizon` view again (instead of a horizon base table), but this still gives the same problem as before. Not being able to use the view can be considered a bug in Oracle.

Of course, instead of counting, it is also easy to select the actual values. The query below does select the `SDO_POINT` (in this case a 3D point) instead of the whole `SDO_GEOMETRY` attribute from a key horizon table, grouped by query polygon:

```
select h.location.SDO_POINT, q.name
from horizons_19b_tert250 h, arcdata q
where mdsys.sdo_relate (h.location,q.geometry,
'mask=ANYINTERACT querytype = WINDOW') = 'TRUE'
```

```
group by q.name;
```

In another experiment we tested the scalability of database querying with respect to the size of the result set. In the experiment we queried the `t_horizons` table multiple times with a growing rectangle. This growing rectangle resulted in a growing number of points being selected. The result of this experiment is presented in Figure 4. From this figure it can be concluded that, for this type of query, the query time increases linearly with the number of selected records.

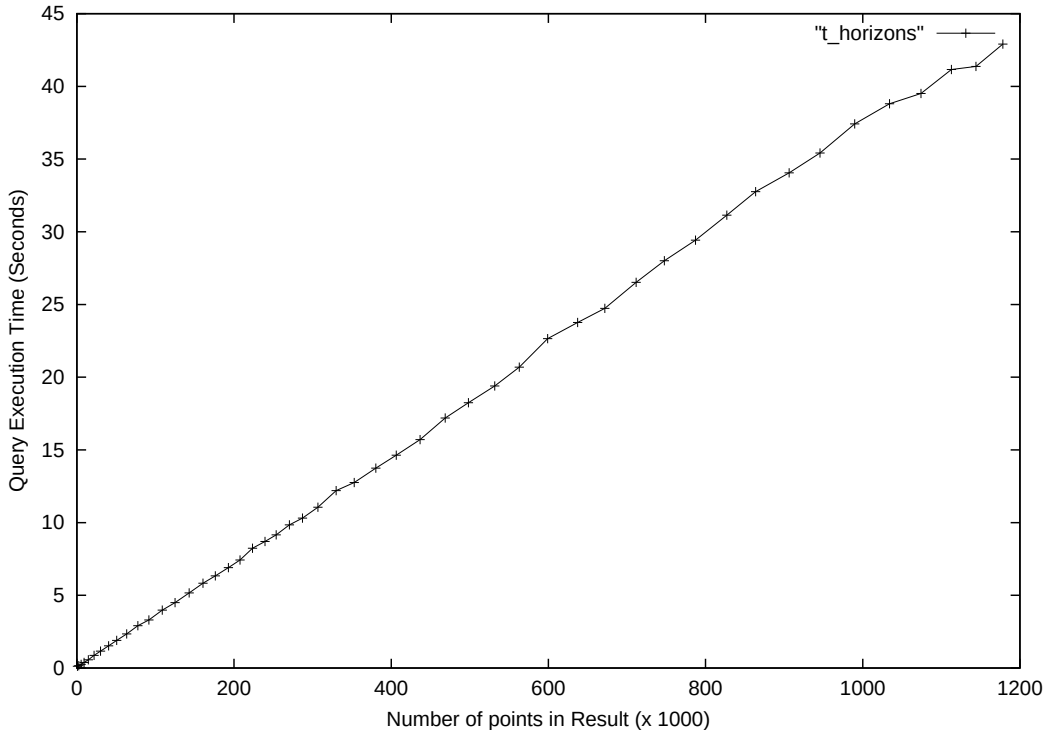

Figure 4: Query time related to number of selected points in `t_horizons`

4.1.2 Fault line clip queries

The second TNO-NITG question was to select the parts of the subcrop lines within a given query polygon. Again, this query could be posed per query polygon, but below it is treated as a spatial join against all query polygons at once. The total length of the clipped fault lines from a specific table (not the view with all fault lines) within a query polygon is summed and compared to the total of the same unclipped fault lines (this query took about 6 to 7 minutes in both Oracle versions):

```
select q.name,
sum(SDO_GEOM.SDO_LENGTH(f.geometry,0.5)/1000) orig_length_km,
sum(SDO_GEOM.SDO_LENGTH(sdo_geom.sdo_intersection
   (f.geometry,q.geometry,0.5),0.5)/1000) clip_length_km
from faultlines_12br_br f, arcdata q
where mdsys.sdo_relate (f.geometry, q.geometry,
'mask=ANYINTERACT querytype = WINDOW') = 'TRUE'
group by q.name;
```

```
NAME                           ORIG_LENGTH_KM CLIP_LENGTH_KM
------------------------------ -------------- --------------
drent                              214.155268     185.990048
fries                               22.839734       3.475886
gron                               209.685518     202.403702
limb                               175.234935     175.139566
overij                              54.354432      48.466407
kb3                                173.271555     150.679158
kb5                                 44.621834      10.024898
kb6                                283.336025     258.245082
kb9                                  5.404084       3.704949
kb10                                43.738479      43.738479
kb13_14                               .511500              0
kb15                               175.234935     175.234714
```

Again, it was not possible to use the view, in this case `faultlines`, in combination with an operator. The `sdo_intersection` function is not an operator, but a true geometry function and in principle the following query works (but it takes a very long time, because it does not use spatial indexing). Therefore, the `sdo_relate` operator is used in the where-clause. In this case the r-tree is used to select only pairs of fault lines and query geometries of which the bounding boxes overlap. Of course, instead of summing the total length, it is also easy to select the actual clipped lines and save the result in a new table `clip_faultlines`:

```
/* slow because no operator can be used on view faultlines */
create table clip_faultlines as
select q.name,
  sdo_geom.sdo_intersection(f.geometry, q.geometry,0.5) clip_geom
from faultlines f, arcdata q
```

One last remark: TNO asked for the begin and end points of the fault lines (clipped against the query geometries). It is not easy to obtain the first and last point from an Oracle geometry polyline using standard Oracle Spatial commands. It is possible to do this using stored PL/SQL procedures (see subsection 4.2.6). Of course, it is also easy to do this in an application which has received the clipped polyline from Oracle (via embedded SQL, ODBC/JDBC or OCI).

4.1.3 Subcrop line clip queries

The subcrop line queries are very similar to fault line queries. For well-known reasons by now, we will not use the view, but one of the subcrop line base tables. The query computes the total length of the clipped subcrop lines, and the total length of the unclipped subcrop lines having overlap with the query polygons (query takes a little over 1 min in both Oracle versions):

```
select q.name,
sum(SDO_GEOM.SDO_LENGTH(f.geometry, 0.5)/1000) orig_length_km,
sum(SDO_GEOM.SDO_LENGTH(sdo_geom.sdo_intersection(f.geometry, q.geometry,0.5),  0.5)/1000) clip_le
from subcrops250_19no_sc f, arcdata q
where mdsys.sdo_relate (f.geometry, q.geometry,
'mask=ANYINTERACT querytype = WINDOW') = 'TRUE'
group by q.name;
```

```
NAME                           ORIG_LENGTH_KM CLIP_LENGTH_KM
------------------------------ -------------- --------------
kb15                               107.383567     107.382856
limb                               107.383567     107.302832
```

4.1.4 Wellpick queries

The following query counts the number of wellpicks per key horizon (`formatie_cd`) and per query geometry (this query takes a little less than 20 seconds in both versions):

```
select count(*), w.formatie_cd, q.name
from wellpicks w, arcdata q
where mdsys.sdo_relate (w.location,q.geometry,
'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
group by w.formatie_cd, q.name;

  COUNT(*) FORMATI NAME
---------- ------- ------------------------------
        45 12BR    drent
         1 12BR    fries
         9 12BR    gelder
        11 12BR    gron
        28 12BR    kb10
... many rows skipped ....
        45 TZE     noordh
        83 TZE     overij

148 rows selected.
```

Selecting the ids `putnaam` within a given key horizon (24TX) and query polygon can be done withe the following query (in less than 1 sec in both versions):

```
select putnaam, w.formatie_cd, q.name
from wellpicks w, arcdata q
where mdsys.sdo_relate (w.location,q.geometry,
'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
and q.name='flevo' and formatie_cd='24TX';

PUTNAAM              FORMATI NAME
-------------------- ------- ------------------------------
NAG-01               24TX    flevo
EMO-01               24TX    flevo
OFL-01               24TX    flevo
```

4.1.5 Seismic interpretation pick queries

Selecting seismic interpretation picks very much resembles querying the well picks. The query below counts the number of seismic interpretation picks per line `lijnnaam` and per query polygon (takes about 6 to 7 minutes in both versions):

```
select count(*), n.lijnnaam, q.name
from navseis n, arcdata q
where mdsys.sdo_relate (n.location,q.geometry,
'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
```

```
group by n.lijnnaam, q.name;

  COUNT(*) LIJNNAAM                       NAME
---------- ------------------------------ ------------------------------
...many other rows...
        14 Y-547                          flevo
        12 Y-547                          fries
         8 Y-547                          kb4
        18 Y-547                          kb5
        18 Y-548                          fries
         5 Y-548                          kb4
        13 Y-548                          kb5

3395 rows selected.
```

4.1.6 Some other queries

Below some other queries with geological data showing a mix of standard SQL functionality and Oracle spatial functionality are listed, e.g. functions `sdo_length`, `sdo_distance` and `sdo_centroid`.

```
select prov2.name,
  SDO_GEOM.SDO_DISTANCE(prov1.geometry,prov2.geometry,0.5)/1000 true_dist_km,
  SDO_GEOM.SDO_DISTANCE(
    SDO_GEOM.SDO_CENTROID(prov1.geometry,0.5),
    SDO_GEOM.SDO_CENTROID(prov2.geometry,0.5),0.5)/1000 centr_dist_km
from arcdata prov1, arcdata prov2
where prov1.name = 'utr' and not prov2.name like 'kb%';
```

The result of this first query is:

```
NAME                           TRUE_DIST_KM CENTR_DIST_KM
------------------------------ ------------ -------------
brab                                      0     60.171245
drent                                     0    129.626678
flevo                                     0     55.006424
fries                                     0    123.669776
gelder                                    0    52.6959426
gron                                      0      167.0096
limb                                      0    112.607683
noordh                                    0    66.5603543
overij                                    0    94.7192287
utr                                       0             0
zeel                                      0    116.581318
zuidh                                     0    52.3406209
```

Note that the polygon-polygon distance is always 0, this is an Oracle bug because not all provinces are neighbors of Utrecht. The query takes about 2 min in both versions.

```
select oid, geometry, name from arcdata
where name like 'kb%';

select max(SDO_GEOM.SDO_LENGTH(geometry, 0.5)), name
from faultlines
group by name;
```

```
select oid from faultlines_19no_br;
select distinct oid from faultlines_19no_br;
select * from faultlines_19no_br where oid='1';

select count(*) aantal,
 max(SDO_GEOM.SDO_LENGTH(geometry, 0.5))/1000 langste_km,
 avg(SDO_GEOM.SDO_LENGTH(geometry, 0.5))/1000 gem_km,
 min(SDO_GEOM.SDO_LENGTH(geometry, 0.5))/1000 kortste_km, name
from subcrops
group by name;

select count(*),min(diepte),max(diepte),min(dikte),max(dikte) from wellpicks;
select count(*) from wellpicks where diepte =1.000E+30;
select count(*) from wellpicks where dikte is NULL;

select * from wellpicks where putnaam = 'OMM-03';
select count(*),formatie_cd from wellpicks group by formatie_cd;

create table polygon_intersect as
select g1.name kb_name, g2.name prov_name,
sdo_geom.sdo_intersection(g1.geometry, g2.geometry,0.5) int_geom
from arcdata g1, arcdata g2
where mdsys.sdo_relate (g1.geometry, g2.geometry,
  'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
  and g1.name like 'kb%' and g2.name not like 'kb%';

select sum(SDO_GEOM.SDO_area(int_geom) orig_area, kb_name name
from polygon_intersect
group by kb_name
union all
select sum(SDO_GEOM.SDO_area(int_geom) orig_area, prov_name name
from polygon_intersect
group by prov_name;

 ORIG_AREA NAME
---------- ------------------------------
1135.47587 kb1
2422.08434 kb2
2072.58794 kb3
4315.52092 kb4
4500.00779 kb5
3430.02160 kb6
6528.43053 kb7_8
4361.76924 kb9
2562.88974 kb10
4404.96893 kb11_12
6119.81383 kb13_14
 808.85769 kb15
5084.79116 brab
2680.46740 drent
2412.29709 flevo
6110.19406 fries
5144.94571 gelder
3054.94484 gron
2208.31876 limb
```

```
4356.33575 noordh
3420.08511 overij
1434.25714 utr
3043.20128 zeel
3712.59008 zuidh
```

The polygon-polygon intersection (with result in the table `polygon_intersect` takes 33 minutes in the production version (and 37 seconds in the patched version). To verify the result, the last query again aggregates the areas of the fragments. The result of the last query is shown. Note that a part of 'noordh' is missing after the overlay (and many parts of map sheets are missing after the overlay, but this is correct). The tolerance specified was always 0.5.

4.2 Cadastral data

The querying of the data has two purposes. First to check if the spatial predicates are functioning correctly. Second, to obtain an impression of the performance of these queries. There are several, more or less independent, options when querying the DBMS with cadastral data. In subsection 4.2.1 some of the options are mentioned,

4.2.1 Cadastral query set up

1. Type of spatial query:
   - window, that is the records selected from the tables based on a given spatial query geometry (constant). Note that several window types are possible: area (polygon, rectangle,...), line, point;
   - join, that is two tables are joined based on a predicate in the where-clause which involves a spatial operator (and operands from both tables).
2. Type of spatial table or view involved:
   - pure spatial tables (such as `xfio_boundary` or `xfio_parcel`) or
   - 'geom/admin' views (such as `akr_objectSOM` or `akr_recht_subject`).
3. Query type:
   - 'count(*)' (aggregate query with possibly only access to index) or
   - 'avg(value)' (aggregate query based on a non-indexed attribute) or
   - 'create table as select' (move data within the DBMS) or
   - frontend application via 'embedded SQL' or 'ODBC/JDBC' or 'OCI' (move data from DBMS to application).
4. Selection based on which type of spatial attribute from table/view:
   - bbox (rectangle) or
   - shape (polyline tested for `xfio_boundary` only) or
   - location (point).

5. Test temperature: 'hot' or 'cold'.

It is not feasible to test all combinations of these options. First of all it was decided to perform mostly 'cold' tests by stopping and starting the DBMS server just before a test is run. 'Hot' is the situation where some or all data has been retrieved before and is possibly present in one of the caches. Note that this occurs at several levels: the DBMS level, the Operating System level and the RAID (disks) level. For some of the tests not much difference exists between 'hot' and 'cold': many query scripts consist of a series of different query geometries which are run against the same table. Basically these are 'cold' queries (different geometries) on 'hot' tables (after the first query in a set most of the table index and some of the data will be in memory). Where a marked difference exists between 'hot' and 'cold' (e.g. pure admin queries) the results are presented.

The Cadastre supplied the query geometries, which can be grouped into two categories: area and line geometries. In Table 13 and Figure 13 (Appendix D) an overview of the 16 area query geometries is given. It should be noted that in Oracle the first point in a polygon must be repeated as the last point. In Table 14 and Figure 14 an overview of the 11 (long) line query geometries is given. Figure 5 gives a visual impression of the location and of the query geometries. Figure 6 shows some query shape details in one region.

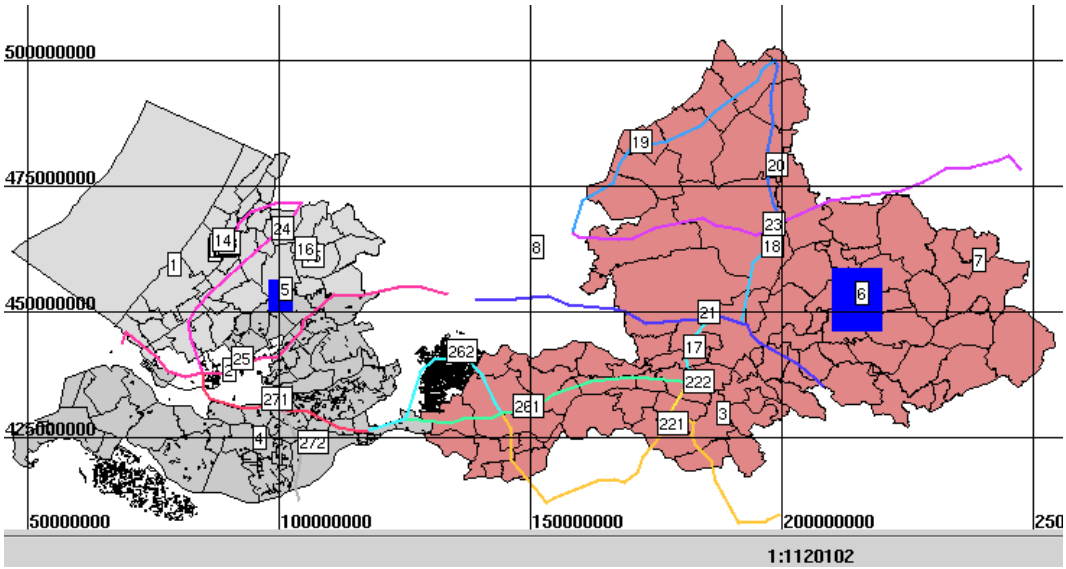

Figure 5: Overview of cadastral selection/query geometries

The area and the length in Tables 13 and 14 are obtained with the following Oracle query (after inserting the query geometries in the table `query_geom`):

```
select TAG,SDO_GEOM.SDO_AREA(q.shape, 0.5)/1000000,
       SDO_GEOM.SDO_LENGTH(q.shape, 0.5)/1000
from query_geom q;
```

The query geometries (and the same is true for the spatial data in the LKI tables) are specified in Oracle in the following way:

- rectangle:
  SDO_GTYPE=2003 (2D polygon),

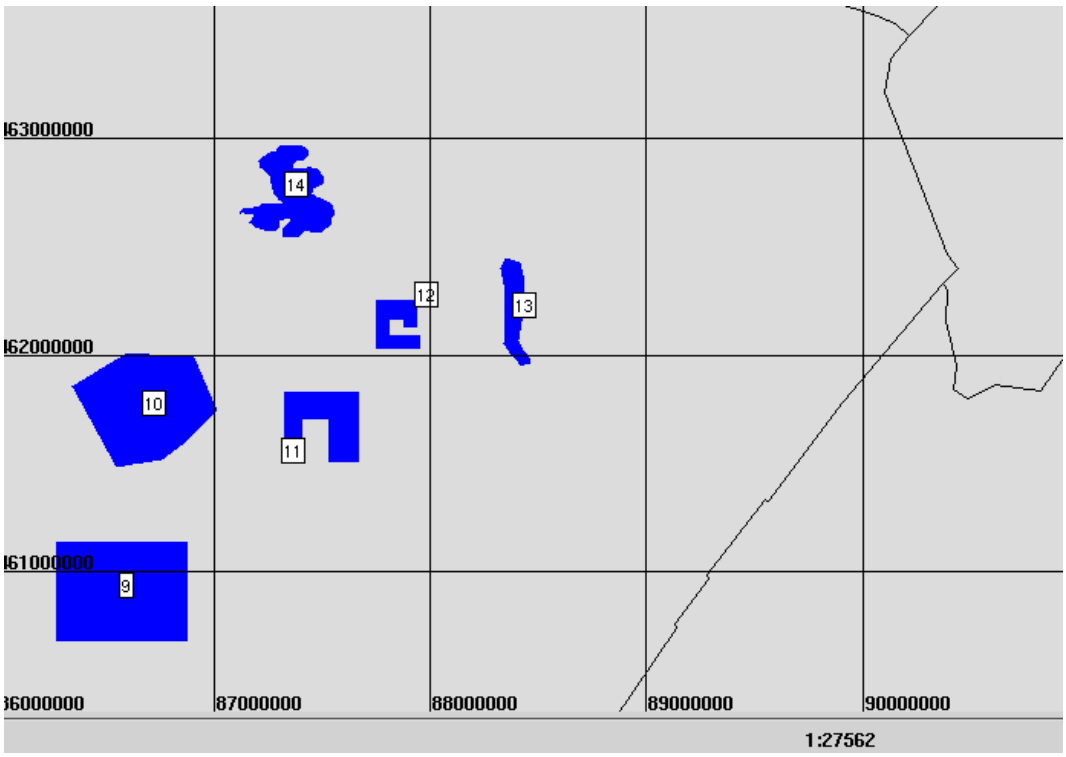

Figure 6: Detail of some selection/query geometries

SDO_ETYPE=3 (rectangle),
SDO_INTERPRETATION=3 (rectangle specified by lower-left and upper-right point)

- polygon:
  SDO_GTYPE=2003 (2D polygon),
  SDO_ETYPE=1003 (exterior polygon ring, counter-clockwise),
  SDO_INTERPRETATION=1 (series of vertices connected by straight lines)

- polygon with hole:
  SDO_GTYPE=2003 (2D polygon),
  SDO_ETYPE=1003 (exterior polygon ring, counter-clockwise),
  SDO_INTERPRETATION=1 (series of vertices connected by straight lines),
  followed by
  SDO_ETYPE=2003 (interior polygon ring, clockwise order),
  SDO_INTERPRETATION=1 (series of vertices connected by straight lines)

- polyline:
  SDO_GTYPE=2002 (2D polyline),
  SDO_ETYPE=2 (line),
  SDO_INTERPRETATION=1 (series of vertices connected by straight lines)

- multi polyline:
  SDO_GTYPE=2006 (2D multi polyline),
  SDO_ETYPE=2 (line),
  SDO_INTERPRETATION=1 (series of vertices connected by straight lines),
  followed by

SDO_ETYPE=2 (line),
SDO_INTERPRETATION=1 (series of vertices connected by straight lines)

Note that not all geometry types do exist in Ingres (polygon with holes, multi polyline). Therefore, these queries are stated with simple geometric primitives (normal polygons and polylines) with AND/OR in the where-clause to specify the proper query. Appendix E contains some examples of Ingres query scripts. Appendix D contains an overview of the query geometries and the scripts for Oracle.

Query performance is measured on a multi-user system, on which several tasks are running. However, during testing as few as possible other tasks were active. Different hardware configurations for the Oracle database were tested. The presented results reflect the situation as described in Section 3, that is, the use of three RAID sets for data and indices. Also the effect on the query times on the order in which the data was loaded, original clustering or random clustering, was tested. For Oracle significant differences in query times occur depending on clustering in contrast to Ingres (no differences). This can be explained by the fact that Ingres supports spatial clustering ('index organized tables' in Oracle terms). Two hardware configurations for the Ingres database were tested. Benchmarks indicate that the query performance is the same whether one RAID set is used or three RAID sets. From this it can be concluded that the query process is not I/O bound. This may be a little surprising, but is an indication that the spatial indexing and clustering of Ingres is working well. Also there is no significant influence on the query times (and results) when comparing a database loaded with clustered data to a database loaded with randomly sorted data. Of course, after loading randomly sorted data, the data must be reorganized in Ingres (with the `modify` statement) to obtain good spatial clustering again.

In Oracle some bugs were discovered, see Section 6. No bugs in Ingres were discovered. Some of the bugs are solved in (production and 'beta') patches and also some performance gain is obtained in patches, see Section 1 for an overview of the different Oracle versions. Therefore, the results of both the production version of Oracle and the patched version of Oracle are included. In Appendix G the results of selected combinations of query options are presented for Oracle and Ingres.

4.2.2 Basic overlap queries

Several possibilities exist to use Oracle spatial functions and/or operators, which are close to the requested functionality: 'find all records in the table which do (partly) overlap with the given search geometry'. Below three examples are given of basically the same query, but with different operators/functions.

```
select count(*) from xfio_boundary where MDSYS.SDO_FILTER (shape,
  mdsys.sdo_geometry(2003, null, null, mdsys.sdo_elem_info_array(1, 1003, 3),
  mdsys.sdo_ordinate_array(78550000,457900000,78650000,458025000)),
  'querytype=window') = 'true';

select count(*) from xfio_boundary where MDSYS.SDO_RELATE (shape,
  mdsys.sdo_geometry(2003, null, null, mdsys.sdo_elem_info_array(1, 1003, 3),
  mdsys.sdo_ordinate_array(78550000,457900000,78650000,458025000)),
  'mask=anyinteract querytype=window') = 'true';
```

```
select count(*) from xfio_boundary where MDSYS.SDO_GEOM.RELATE (shape,'anyinteract',
  mdsys.sdo_geometry(2003, null, null, mdsys.sdo_elem_info_array(1, 1003, 3),
  mdsys.sdo_ordinate_array(78550000,457900000,78650000,458025000)),0.0) = 'true';
```

The geometry function `sdo_geom.relate` is very slow as it does not use a spatial index. The speed of the spatial operators `sdo_filter` and `sdo_relate` is much better because the index is used. However, `sdo_filter` returns all potential candidates (based on a bounding box selection), which is more than wanted. Therefore, the operator `sdo_relate` is used in most of the queries. Note that in order to function at all both `sdo_filter` and `sdo_relate` need a spatial index. The advantage of `sdo_geom.relate` is that it can function without an index.

The overlap queries, based on the operator `sdo_relate`, are applied to several pure spatial tables (`xfio_`) and geom-admin views (`akr_`). Tables 16 to 19 in Appendix G show the timing results of the pure geometry queries: three Oracle variants (production original clustering, patch original clustering, patch random clustering) and one Ingres version. The tables are splitted into area queries and line queries. The largest table (`xfio_boundary`, indicated as 'bnd' in the tables) is best used for comparing the results. Table 15 in Appendix G shows the number of selected records per cadastral geometry query. These numbers are identical for Oracle and Ingres. Oracle offers the possibility to specify a tolerance for geometry operations. Increasing the tolerance has the expected effect of returning more selected records than the same query with a smaller tolerance. All cadastral queries in Oracle were run with a tolerance of 0.

In the analysis in this paragraph the focus is on the counting of records based on overlap with the `shape` attribute. Assuming that the index contains only (r-tree) rectangles, both the index and the base table must therefore be used in answering this query, labelled with 'bnd/shp count'. Note that this is the third (in the case of Oracle) or the second (in the case of Ingres) series of queries in the set of geometry queries. So, the table is relatively hot. However the other series have similar results. The total times (in format h:mm:ss) of the area queries (line queries within parenthesis) for Oracle production and Ingres are 0:01:29 (4:12:48) and 0:00:32 (0:49:47). This indicates that Oracle is much slower than Ingres. The Oracle patch 0115 improves things quite a lot, overhead is reduced and a smarter use of the r-tree index for long polylines is realized. The results are for area queries 0:00:45 and line queries 0:10:27, the latter is faster than Ingres. The performance of multiple polyline queries (line queries 22, 26 and 27) is relatively poor in Ingres. Oracle implicitly uses the Ingres clustering, as the data is originally unloaded from Ingres in a sorted manner. Removing this Ingres 'clustering' and loading randomly sorted data results in 0:24:21 for area queries and 1:20:51 for line queries (both only for the Oracle fast patch 0115). This proves that Oracle really needs some kind of 'index organized' table for spatial data as the area queries are more than 30 times slower (using same data, same index).

Table 20 in Appendix G shows the results of timing the geom-admin queries (with a spatial entrance). It contains both the production and patch 0115 version of Oracle and one Ingres version. The overall conclusion is that the Oracle patch version is a little slower with the area queries compared to Ingres, but more than two times faster with the long polyline queries. The performance of the Oracle production version is one order of magnitude slower for both area and line queries (compared to the Oracle patch version

or to Ingres). The last three columns of Table 15 show the number of selected records per cadastral geom-admin query and the average 'objectSOM' obtained. All results are the same for Oracle and Ingres, which gives confidence in the correctness.

4.2.3 Spatial join

It was decided to focus on 'window' queries. However, the 27 query geometries were also inserted in a query table `query_geom` and in addition to the individual queries (per query geometry), the same queries are also posed based on one spatial join[3] (only for the `xfio_sympnt` table and the `xfio_boundary` table). A typical spatial join query is shown below, the results are aggregated and presented per query geometry (labelled with a `TAG`):

```
select count(*), q.TAG
from xfio_boundary t, query_geom q
where mdsys.sdo_relate (t.shape,q.shape,
  'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
group by q.TAG;
```

The results of this operation can be found in Tables 21 and 22, columns 'sym/loc join' and 'bnd/shp join'. The query times look reasonable (but cannot be compared to something else) with the exception of the join query on the `xfio_boundary` table in the Oracle production version (took more than 20 hours). In the column with the spatial join only the total time can be reported. Note that the summed time for all 'bnd/shp count' queries for Oracle patch is 0:10:27 (Table 17), which is a little faster than the single join query of 0:13:34 (Table 22).

4.2.4 Queries not based on overlap operator

Besides the overlap query functionality, that is using the operator `sdo_relate`, also operators `sdo_filter` (find overlap only based on bounding boxes), `sdo_within_distance` (find objects within a given distance from the query geometry), and `sdo_nn` (find the nearest neighbors from the query geometry) are tested. Again, note that spatial operators need a spatial index in order to work. Objects (partially) overlapping the query geometry are treated as 'perfect' nearest neighbors, see Figure 7, which shows the result of selecting the 10 nearest neighbors produced with the following query:

```
select ogroup, object_id, tmax, shape
from xfio_boundary where mdsys.sdo_nn (shape,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(78550000,457900000,78650000,458025000)),
  'sdo_num_res=10') = 'TRUE' ;
```

Results for operator `sdo_filter` can be found in Tables 16 to 18, column 'bnd/shp filter'. In contrast to operator `sdo_relate` the `sdo_filter` operator does not benefit from Oracle patch 0115 (and is really slow with random data). In Tables 21 and 22 the timings for the `sdo_within_distance` operator (columns 'sym/loc distance' and 'bnd/shp distance') and the `sdo_nn` operator (columns 'sym/loc nn1000' and 'bnd/shp nn1000') are presented. As the name of the column for `sdo_nn` implies, the 1000 nearest neighbors

[3]Spatial join queries only tested with the Oracle DBMS.

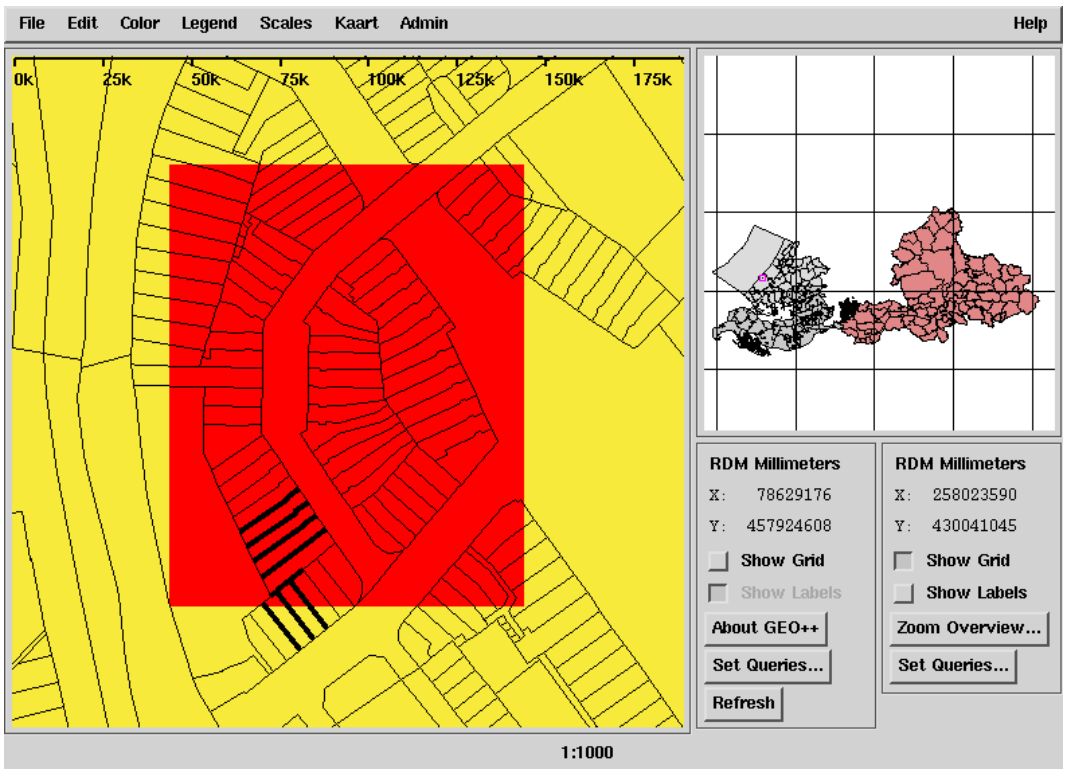

Figure 7: Selecting the 10 nearest neighbors of `xfio_boundary`

of the query geometries were selected. Again these operators do not benefit from patch 0115, so consequently they suffer from the long polyline queries.

4.2.5 Other Oracle geometry functions

Oracle offers several interesting geometry functions (not needing a spatial index) of which several were tested: `sdo_area` and `sdo_length` (to obtain characteristics from the query geometries), `sdo_buffer` (to compute a buffer around a query geometry, tested in combination with overlap and results compared to the `sdo_within_distance` queries) and `sdo_intersection` (to clip spatial data from the source tables with the query geometries). The last two functions are used in combination with a spatial operator in the where-clause in order to allow efficient execution. This works especially well for the `sdo_intersection` function as is clearly visible in Tables 21 and 22 (columns 'sym/loc clip' and 'bnd/shp clip'). The results of clipping have been checked visually and no errors were found in this way. Also the number of objects clipped for each query geometry corresponds with the number found with the `sdo_relate` operator. The result of the `sdo_intersection` function can be seen in Figures 8 and 9. Below an example of how the clip function (`sdo_intersection`) can be used in Oracle, in this case with query geometry 9:

```
create table q9 as select
sdo_geom.sdo_intersection(shape,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(86275806,460673681,86275806,461137450,
  86882416,461137450,86882416,460673681,86275806,460673681)),0.5) clip_geom
```

```
from xfio_boundary where mdsys.sdo_relate (shape,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(86275806,460673681,86275806,461137450,
  86882416,461137450,86882416,460673681,86275806,460673681)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;
```

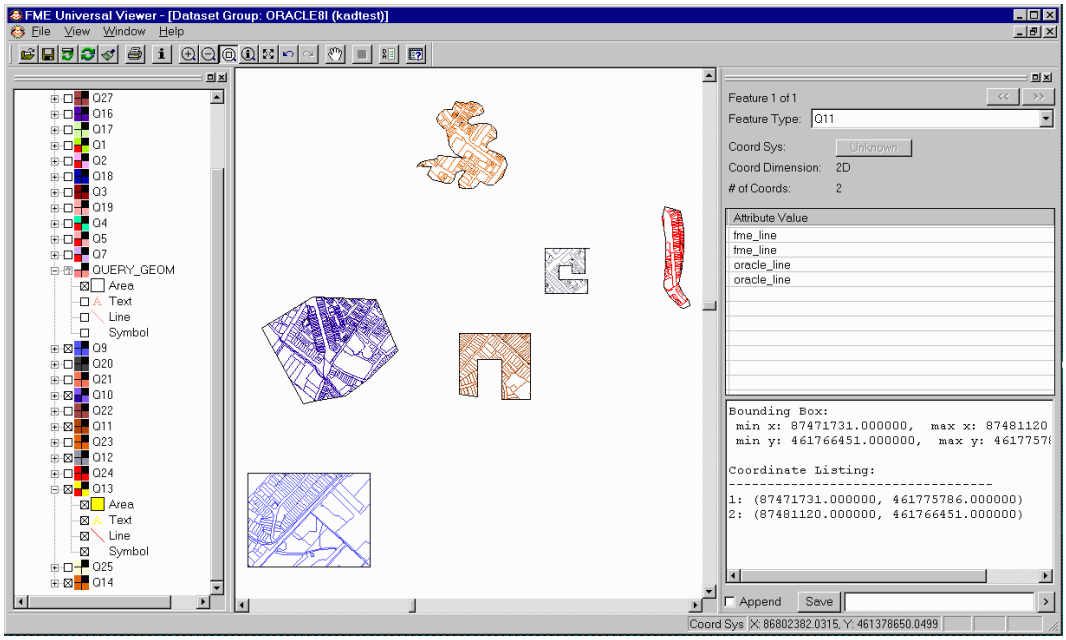

Figure 8: The result of clipping `xfio_boundary` with query geometries 9-14

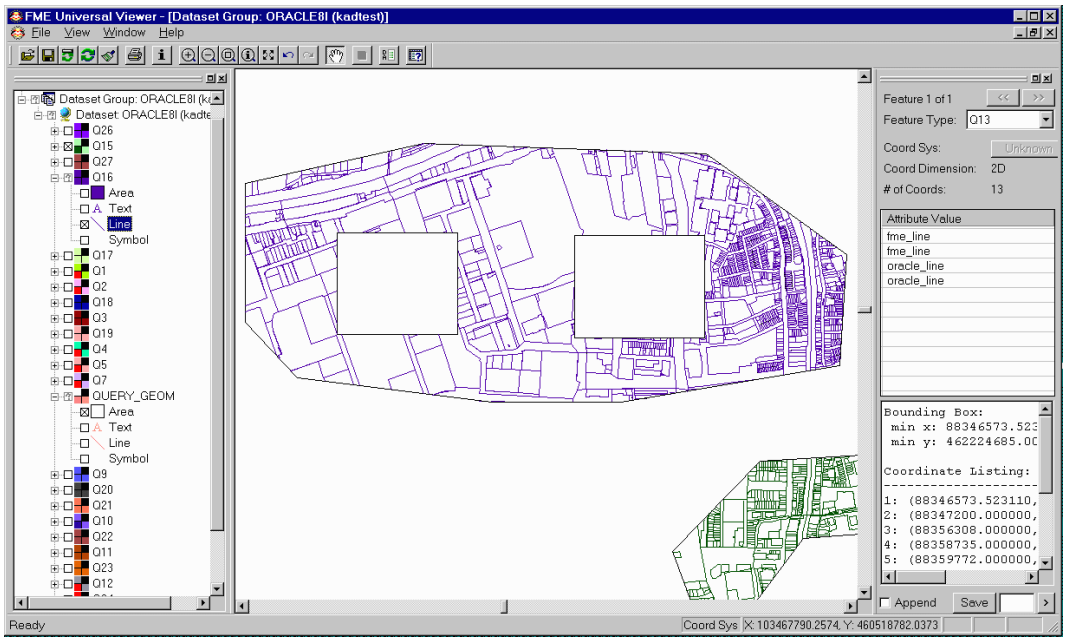

Figure 9: The result of clipping `xfio_boundary` with query geometry 16

The number of objects included in buffered geometries is equal to the number of objects selected by the `sdo_within_distance` operator (using the same distance).

4.2.6 Additonal functions

Oracle Spatial is missing a number of useful functions which are part of the OpenGIS standard. It is possible to define these functions and an (unsupported) package `sdo_util` containing this functionality does indeed exist. It provides functions such as:

- `n = get_num_points(geometry)`
- `n = get_num_ordinates(geometry)`
- `geometry = mbr(geometry)`
- `geometry = get_element(geometry,n)`
- `string = get_element_type(geometry)`

After installing the package (the first two lines in the example), it is possible to use the defined functions at SQL level as indicated in the example below:

```
sqlplus kadtest/kadtest @sdoutil
sqlplus kadtest/kadtest @sdoutil_body
sqlplus kadtest/kadtest
SQL> select sdo_util.get_num_points(shape) nr_point,
  2    sdo_util.get_num_elements(shape) nr_element,
  3    sdo_util.get_num_ordinates(shape) nr_ord,
  4    sdo_util.get_element_type(shape) elem_type, tag from query_geom;
```

In most cases the functions work fine, in certain situations it can be noted that the package was developed against an older Oracle version; e.g. the `get_element_type` function does not recognize the new ETYPEs.

To check the correctness of geometries the functions `sdo_geom.validate_geometry` and `sdo_geom.validate_layer` are available as standard functions in Oracle Spatial. While loading spatial data Oracle only checks the syntax of the statements used, the 'content' of the data is not checked. So it is possible to store invalid geometries in a database (e.g. polygons which are not closed, have the wrong direction or are self-intersecting). The validate functions can be used to identify geometries which do not conform to the Oracle spatial object types. In all cases where we used the validate functions they returned the proper result (`TRUE` for valid geometries, an error code for invalid geometries).

4.2.7 From admin to geom

The query types described in previous subsections all start with spatial query objects as an entrance. An alternative scenario is starting with an administrative entrance and obtaining the related spatial information. There is a great difference in the number of rights (and objects/parcels) related to a single subject. To test performance, subjects in several categories were selected (from different cadastral offices):

- related to 1 right;
- related to 10 rights;

- related to 100-105 rights;
- related to 500-530 rights;
- related to over 4,000 rights.

In total 20 queries are executed (all with the same structure, that is using three tables in the queries):

```
create table tmp as
select *
from lki_parcel
where x_akr_objectnummer in
  (select g_akr_objectnummer
   from object_parcel op,mo_recht r
   where op.x_akr_objectnummer=r.x_akr_objectnummer
     and r.gerechtigde='0123456789');
```

The last table in Appendix G, Table 23, shows both the timings and the number of selected records for the administrative queries, that is queries with a subject-id as entrance and selecting parcels. Oracle and Ingres find the same number of parcels (last column) and are equally fast. Differences in timings that do occur are the result of rounding errors. These queries have been repeated a second time, resulting in 'hot' query times. The results are approximately 3 times faster than the 'cold' queries. Note that these queries do not use a spatial index, but the clustering is important.

4.3 Query types not performed

No topology queries on parcels as implemented in the query tool are possible inside the Ingres and/or Oracle DBMSs (e.g. find all parcels within a certain distance of a trace).

All tests are done with straight line segments and not with circular arcs. This is true for both the data stored in the tables in the DBMS and the specified query region/line.

As indicated in the introduction of Subsection 4.2 no data transfer between client and server has been tested. An exception is the JDBC program of Section 5. However, this section focuses on functionality and not on performance.

No linear referencing functions are tested. These functions are especially useful when modelling and analyzing linear networks, such as roads or pipelines.

3D data storage is supported by Oracle. However, the third dimension cannot be used in indexing and in the operators/functions. The best one can expect now that the third dimension remains in the geometry but is ignored by functions/operators. In the test no 3D queries are posed.

No other coordinate system than the RD (Rijksdriehoekscoordinaten, Cartesian coordinates) has been used. Therefore no coordinate transformation functions are tested. For larger regions (than the Netherlands) geographic coordinates (lat/long on some kind of ellipsoid) can be useful. Also, functions on the Earth's surface (intersect, area, distance) are expected in Oracle 9i.

5 JDBC connection to Oracle 8i Spatial

Once data has been loaded into an Oracle database, it can be accessed from any Java application across the internet via JDBC. Below an example of a Java program can be found that calculates the minimum bounding rectangle for a spatial column of a table. This is done by retrieving the spatial data from the database and expanding an initially empty box with all geometries encountered.

In order to run this program there should be a JDBC connection, and the Oracle Java library `sdoapi.zip` should be installed.

5.1 JDBC connection

To establish a JDBC connection in a Java program the following steps should be taken:

- Add the Oracle JDBC driver to your Java CLASSPATH enviroment variable. In our case this is done as follows:

  ```
  setenv CLASSPATH ${CLASSPATH}:${ORACLE_HOME}/jdbc/lib/classes12.zip
  ```

- Register the Oracle driver to the DriverManager. This can be done by adding the driver class to the `java.lang.System` property `jdbc.drivers`. This is a (colon separated) list of driver class names. We have done this by aliasing the Java command:

  ```
  alias java java -Djdbc.drivers=oracle.jdbc.driver.OracleDriver
  ```

  When Java is started the JDBC DriverManager automaticly registers all drivers that are listed in `jdbc.drivers`.

- Determine the parameters for the getConnection call. If a database with name `igis` is running on computer `www.gdmc.nl` listening to port 1521 (default) the connection is established with the following command:

  ```
  Connection conn =
  DriverManager.getConnection(
     "jdbc:oracle:thin:@www.gdmc.nl:1521:igis", // Connect String
     "user",                                    // Username
     "password");                               // Password
  ```

  In this case the *thin* JDBC driver is used. This is the appropriate driver in case the client application is an applet or is run via the internet. If you are writing a client application for an Oracle client environment, then the JDBC OCI driver will give better performance. In this case the connect string would be:

  ```
  "jdbc:oracle:oci8:@kadtest"
  ```

  Here kadtest is an entry in the `tnsnames.ora` file.

5.2 Install `sdoapi.zip`

The Oracle Spatial Java Library simplifies and standardizes the work of spatial service providers and application developers. It allows access to and processing of geometry objects. The file `sdoapi.zip` can be downloaded from Oracle Technet:

`http://technet.oracle.com/software/products/spatial/htdocs/listing.htm`

This should be all that needs to be done to run the following JDBC program.

5.3 Example JDBC program

```
import java.io.*;
import java.sql.*;
import oracle.sql.STRUCT;
import oracle.sdoapi.OraSpatialManager;
import oracle.sdoapi.adapter.*;
import oracle.sdoapi.geom.*;

/**
 * This sample program reads all geometries from the spatial column
 * of a database table and calculates the minimum bounding rectangle.
 */
public class JdbcEnvelope
{
  public static void main(String args[]) throws Exception
  {
    //
    // The program has two parameters: 1 the name of the table
    // 2. the name of the spatial attribute.
    //
    if (args.length != 2)
    {
      System.out.println("Parameter:");
      System.out.println("<Table name>: Table name of an input table");
      System.out.println("<Attribute name>: Attribute name in an input table");
      return;
    }

    //
    // The database only likes uppercase names.
    //
    String tablename = args[0].toUpperCase();
    String attributename = args[1].toUpperCase();

    //
    // Connect to the database
    //
    Connection conn = DriverManager.getConnection(
       "jdbc:oracle:thin:@www.gdmc.nl:1521:kadtest", // Identification of database
       "tno",                                        // username.
       "nitg");                                      // password.

    //
```

```
    // Make Geometry Adapter.
    //
    GeometryAdapter sdoAdapter =
      OraSpatialManager.getGeometryAdapter("SDO", "8.1.6",
                                           STRUCT.class, null, null, conn);

    //
    // Create and Execute SQL statement.
    //
    Statement stmt = conn.createStatement();
    ResultSet result =  stmt.executeQuery("SELECT " + attributename + " FROM " + tablename);

    //
    // Find out the column number of the result attribute.
    //
    int column = result.findColumn(attributename);

    //
    // We start with an empty envelope.
    //
    Envelope env = new EnvelopeImpl();

    //
    // Walk Through Resultset.
    //
    while (result.next())
    {
      //
      // Read the Object from the resultset into an Oracle STRUCT.
      //
      STRUCT dbObject = (STRUCT) result.getObject(column);

      //
      // Convert the STRUCT to an oracle Java geometry type
      // using the input adapter
      //
      Geometry geom = sdoAdapter.importGeometry(dbObject);

      //
      // Expand the current bounding box with the object
      // we just read.
      //
      env.expand(geom.getEnvelope());
    }

    //
    // We are ready. Print the envelope.
    //
    System.out.println("The envelope is: " + env);

    stmt.close();
    conn.close();
  }
}
```

6 Software limitations and bugs

In this section most 'defects' of the Oracle DBMS related to the use of spatial data are collected. Various inherent (as for now) limitations, bugs, performance problems and ways to circumvent these are described. First an overview of Oracle Spatial limitations and bugs, as encountered in this project, is presented in a table. In the subsections following, the details are explained.

The status registered in Table 12 is the situation as per March 2001. The severity attributed to a defect depends on the nature of the problem (real bug or 'only' a performance issue), how easy it is to use a work-around, how difficult (we expect) will be the solution and in the case of performance issues the impact the problem has on performance. In the 'Fixed' column the version of Oracle is indicated in which the problem will be fixed (if known to us).

Description	**Status**	**Severity**	**Fixed**
Switch off logging for spatial data	open	medium	??
Use direct path load for spatial data	open	medium	9i, rel 2
Parallellize spatial statements/queries	open	low	9i, rel 2
Use partitioned spatial tables	open	low	9i, rel 1
Use physically clustered spatial tables	open	high	??
Direct path loading error	open	low	9i
Owner missing from metadata table	open	low	9i
Parse overhead of spatial queries	open	medium	9i
Inefficient storage of r-tree tables	open	low	9i, rel 2
R-tree fanout problem	closed	-	8.1.7
Analyze of r-tree table not performed	open	low	9i
Within_distance misses one side of line	closed	-	8.1.7
Inconsistent behavior of tolerance	open	medium	9i
Valid geometry not accepted	open	medium	??
Spatial index and 'union all' view	open	high	9i, rel 2
Problems with mixing 2D/3D operands	open	medium	9i, rel 2
Wrong polygon-polygon distance computed	open	medium	9i, rel 2
Unexpected failure of `sdo_intersection`	open	high	9i, rel 1

Table 12: Status of software limitations and problems

6.1 Oracle Spatial limitations

The most optimal situation for a DBMS is the case where all available functionality can be applied equally to all data in the database, whatever the type of data (an 'orthogonal' DBMS design). This is not yet true for spatial data in Oracle 8i. The situation is better than in previous releases (e.g. `sqlldr` can now be used to load spatial data), but is still not quite where it should be. The following limitations exist for tables which contain 'object' columns (spatial is one such data type):

- It is not possible to switch off logging. This means that logging is always active

during loading, index creation or updating (options to disable logging are 'silently' ignored by the database). Especially while mass loading data or during index creation this slows down the process.

- Direct path load and direct insert cannot be used, again this affects performance.
- No parallellization of DDL/DML statements or queries is possible.
- Partitioned and clustered tables (in the specific Oracle sense) are not an option for spatial data.
- No useful form of index-organized tables can be applied. It is possible to create an index-organized table (clustered in the 'normal' sense) which contains a spatial column. But this column cannot be used as the primary key, nor is it possible to use secondary spatial indices once a table is index-organized.

Together these limitations make the handling of spatial data in the Oracle DBMS slower than the handling of alphanumeric data. Some of the limitations will be removed in Oracle 9i. At this moment it is not exactly clear which ones that will be.

Note that this section not imply anything with respect to Ingres. Some limitations may also be present in Ingres, others not.

6.2 Bugs and performance issues

It is not surprising that extensive use of a complex piece of software like a modern DBMS brings to the surface a number of bugs. For the problems encountered and sent to Oracle patches were available in a short time (but sometimes not yet included in generally available patches). The problems are described in the order in which they were discovered, some of them are not really bugs but 'flaws' with a performance penalty. If not indicated otherwise all problems/bugs occur in all versions of Oracle Spatial 8.1.7 (production and patched).

It has to be remarked that, apart from the work-arounds and fixes mentioned in this section, also other performance improving measures were adopted. One of these is the incorporation of hints for the query optimizer. Adding e.g. the `/*+ ORDERED */` hint to queries or DML statements can sometimes make a tremendous difference in performance. It can also work counter productive, so care must be taken where and when to use optimizer hints. No hints were used in query results reported, but a number of queries will be (much) faster if hints are used. Over time the query optimizer will improve, although it will remain doubtful if (without hints) always the most optimal query plan will be made.

Direct path loading error

The fast 'direct path loading' option (only available for non-spatial data, so used for AKR) does not work if at the same time the database parameter `session_cached_cursors` is set to a nonzero value. Setting e.g. `session_cached_cursors=300` keeps the specified number of cursors in memory. This improves the performance of spatial queries (related

to 'parse overhead' problem). It is a documented bug, scheduled to be solved in Oracle 9i production. The work-around is to make sure that this parameter is 0 while using direct path load, and setting it to a higher value before spatial queries are run.

Owner missing from metadata table

In the `user_sdo_geom_metadata` table some metadata (dimension, extent, tolerance) about all spatial columns is stored. In the version distributed with Oracle 8.1.7 the 'owner' attribute is missing from this table. This does not lead to erroneous results, but performance is affected. Problem is scheduled to be solved in Oracle 9i production. A work-around is to add the owner attribute to the table before the metadata table is used. A script to perform this action automatically was added to the database creation scripts of the test database.

Parse overhead of spatial queries

Another well documented problem is the very heavy parse overhead of most spatial queries. Again this is a performance problem and scheduled to be solved in Oracle 9i production. No work-around is available for this, although the special patch libordsdo_0115 did include some improvements related to this. Also the database parameters `session_cached_cursors` (mentioned before) and `cursor_space_for_time` if set to 'true' alleviate this problem somewhat.

Inefficient storage of r-tree tables

One of the parameters which can be specified for an r-tree spatial index is the fanout value: the number of nodes which 'fanout' from all nodes in the tree. By playing with this value (an r-tree should not have too many levels, so for spatial tables with many rows a higher fanout value than the default of 35 is better). We discovered that some fanout values can lead to rather inefficient storage of the r-tree table. Using the default fanout of 35 the r-tree tables look like:

TSpace	TABLE_NAME	Size_Mb	Blocks	Empty	Num_rows	Chained	AvRowLen	FrSpace
INDX	XFIOX_BOUNDARY_0B_RT$	843.82	108009	22	324019	108006	2076	1857
INDX	XFIOX_BOUNDARY_0_RT$	843.82	108009	22	324019	108006	2076	1857
INDX	XFIOX_GCPNT_0_RT$	5.16	661	362	1969	656	2075	1904
INDX	XFIOX_LINE_0_RT$	294.26	37665	222	112981	37660	2076	1858
INDX	XFIOX_PARCELOVER_0_RT$	3.63	465	46	1385	461	2075	1905
INDX	XFIOX_PARCEL_0_RT$	320.97	41084	387	123251	41083	2076	1857
INDX	XFIOX_SYMPNT_0_RT$	172.62	22095	432	66275	22091	2076	1858
INDX	XFIOX_TEXT_0_RT$	137.79	17637	282	52910	17636	2076	1858

Ugly and a waste of space. Using a fanout of 64 they look normal, as you expect:

TSpace	TABLE_NAME	Size_Mb	Blocks	Empty	Num_rows	Chained	AvRowLen	FrSpace
INDX	XFIOX_BOUNDARY_0B_RT$	688.38	88113	462	176221	0	3698	702
INDX	XFIOX_BOUNDARY_0_RT$	688.38	88113	462	176221	0	3698	702
INDX	XFIOX_GCPNT_0_RT$	4.19	536	487	1072	0	3697	704

INDX	XFIOX_LINE_0_RT$	240.03	30724	507	61446	0	3698	703
INDX	XFIOX_PARCELOVER_0_RT$	2.97	380	131	753	0	3697	772
INDX	XFIOX_PARCEL_0_RT$	261.87	33519	272	67032	0	3698	703
INDX	XFIOX_SYMPNT_0_RT$	140.81	18024	407	36045	0	3698	703
INDX	XFIOX_TEXT_0_RT$	112.42	14390	457	28776	0	3698	704

This problem is not yet officially reported to Oracle, status is unknown. The work-around is to find a decently working fanout value by trial and error.

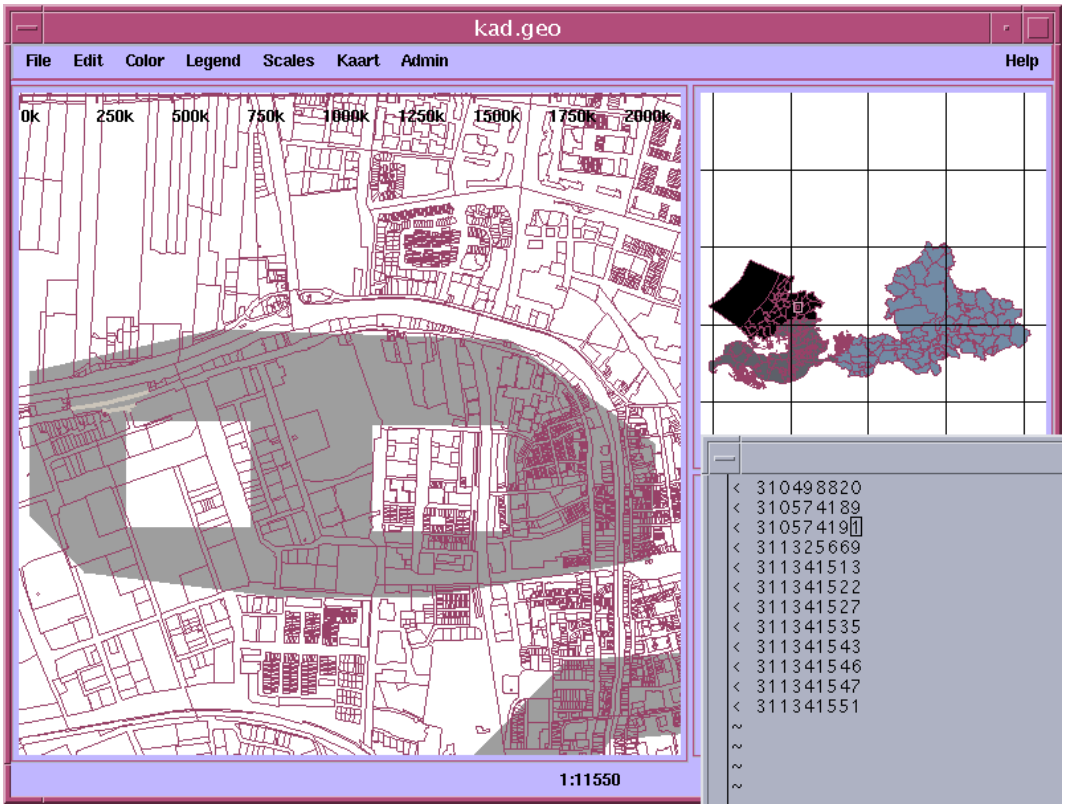

Figure 10: Objects (highlighted) missing from result due to r-tree fanout problem

R-tree fanout problem

In exceptional cases it is possible at higher fanout values that not all objects indicated by the r-tree index are returned to the application. The name of this problem suggests otherwise, but this proved to be a database caching problem. It had nothing to do with the r-tree code. Figure 10 illustrates an example of this problem as it occurrred in the tests. The problem was solved by the latest version of both patch libordsdo_0106 and patch libordsdo_0115.

Analyze of r-tree table not performed

The r-tree index is stored in a 'normal' table, e.g. XFIOX_BOUNDARY_0_RT$ in the listing above. As is the case with all tables, before the query optimizer can determine the best query plan it needs some information about the tables involved in a query. This

information should be generated automatically for the r-tree tables because one can hardly expect this from a user (this table is not documented in the 'standard' documentation). Because of a bug this is not done automatically, the obvious work-around is to do it explicitly (included in the scripts which create the test database). It is not exactly known when this is scheduled to be fixed, Oracle 9i production at the last.

Within_distance operator misses all objects on one side of line

This error was introduced by the first version of performance patch libordsdo_0115. If applied to line geometries the `sdo_within_distance` operator missed roughly half of all objects it should find (illustrations in Figures 11 and 12). The second and later versions of patch libordsdo_0115 solved the problem.

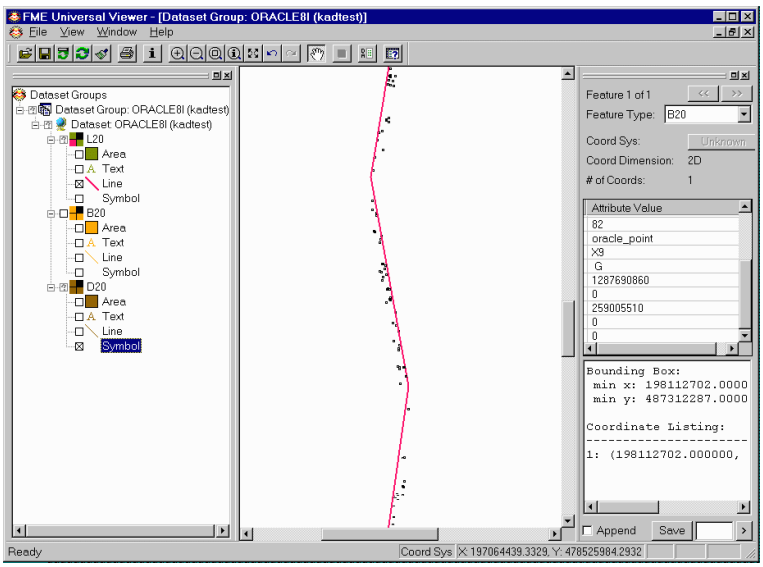

Figure 11: Within_distance operator missing alternating sides of query polyline

Inconsistent behavior of tolerance

The accuracy of geometry operations while using a tolerance > 0 is questionable, e.g. specifying a tolerance of 0.5 also returns objects at a distance > 0.5. The Oracle `within_distance` operator with 0 tolerance does return accurate results (if it works, see next bug). In a communication with the developers we learned that this behavior results from the fact that at the moment the tolerance is not consistently applied in the DBMS. Sometimes geometry calculations are performed with maximum accuracy and then the tolerance is applied to the result. In other cases the tolerance is taken into account during geometry calculations. For version 9i of Oracle Spatial an 'overhaul' is planned of the way the tolerance is treated inside the database.

Valid geometry not accepted by within_distance operator

This is another example of a problem with the `sdo_within_distance` operator. At a specific combination of tolerance (0), distance (100000) and query geometry (coordinates

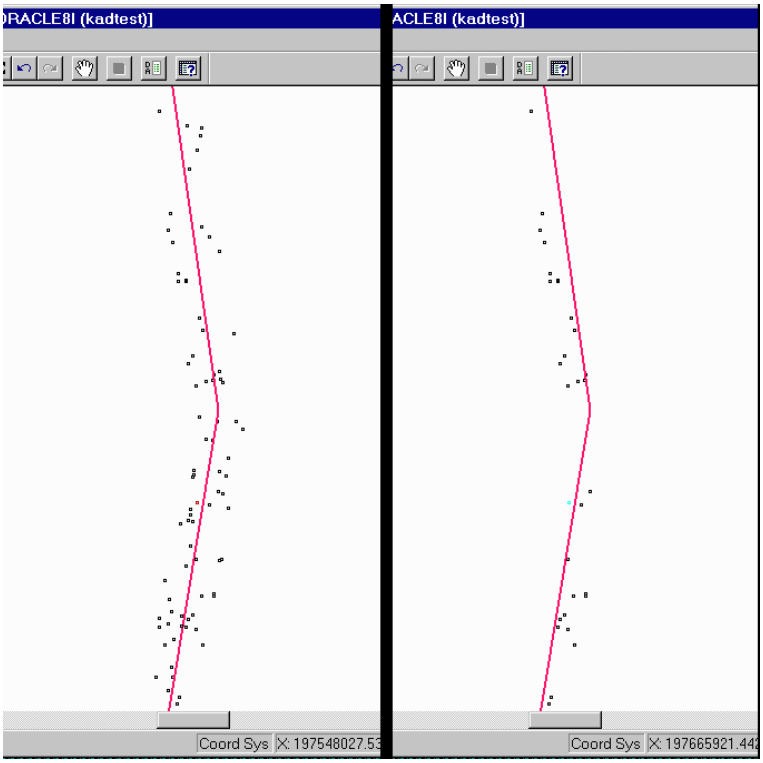

Figure 12: Detail of the results obtained with the buffer overlap query (left) compared to the results obtained with the within_distance operator (right)

78550000,457900000,78650000,458025000) the operator did not accept an otherwise perfectly valid query:

```
SQL> select count(*) from bug_test where mdsys.sdo_within_distance (location,
  2    mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  3    mdsys.SDO_ORDINATE_ARRAY(78550000,457900000,78650000,458025000)),
  4    'distance = 100000') = 'TRUE'
  5  /
select count(*) from bug_test where mdsys.sdo_within_distance (location,
*
ERROR at line 1:
ORA-29902: error in executing ODCIIndexStart() routine
ORA-13052: unsupported geometric type for geometry WINDOW_OBJECT.
ORA-06512: at "MDSYS.SDO_INDEX_METHOD", line 84
ORA-06512: at line 1
```

This problem is registered as an official bug. It is unknown when a patch will be forthcoming, probably not before Oracle 9i. Various work-arounds exist, specifying a tolerance > 0 (e.g. 0.0001) avoids the error, so does a slightly changed distance (e.g. 99999.99).

Spatial index and a 'union all' view

A view using the 'union all' construction on two (or more) tables with a spatial column cannot be used in a spatial join query. In the example below the table `TAB_A` and `TAB_B` contain both one 2D point (and have r-tree indices), `TAB_Q` contains one rectangle (and also has an r-tree index), and `VIEW_AB` is the view on `TAB_A` and `TAB_B`. The last query in

the script, using the operator `mdsys.sdo_relate`, gives the following error message:

```
ERROR at line 1:
ORA-13226: interface not supported without a spatial index
ORA-06512: at "MDSYS.MD", line 1723
ORA-06512: at "MDSYS.MDERR", line 8
ORA-06512: at "MDSYS.SDO_3GL", line 57
ORA-06512: at line 1
```

The querytype specified (WINDOW or JOIN) does not matter. This error message is not generated in the situation that `TAB_A` (or `TAB_B`) is used instead of `VIEW_AB`. The same is true if a 'constant' value is used as second operand instead of `TAB_Q`. This problem has been reported in the last week of testing. No response from Oracle has been received. Below the script:

```
drop table TAB_A;
create table TAB_A (LOCATION mdsys.sdo_geometry);
insert into TAB_A (LOCATION) values (
  mdsys.SDO_GEOMETRY(2001,NULL,
  mdsys.SDO_POINT_TYPE(223875,598875,NULL),NULL,NULL));
delete from user_sdo_geom_metadata
  where table_name = 'TAB_A' and column_name = 'LOCATION' ;
insert into user_sdo_geom_metadata
  values ('TAB_A','LOCATION', mdsys.sdo_dim_array(
    mdsys.sdo_dim_element('X',0.0,325000.0,0.5),
    mdsys.sdo_dim_element('Y',275000.0,625000.0,0.5)),NULL);
drop index TAB_A_1 force;
create index TAB_A_1 on TAB_A(LOCATION)
  indextype is mdsys.spatial_index
  parameters ('initial=1m next=1m pctincrease=0 sdo_fanout=32');
analyze table TAB_A_1_rt$ compute statistics;
analyze table TAB_A compute statistics;

drop table TAB_B;
create table TAB_B (LOCATION mdsys.sdo_geometry);
insert into TAB_B (LOCATION) values (
  mdsys.SDO_GEOMETRY(2001,NULL,
  mdsys.SDO_POINT_TYPE(223877,598877,NULL),NULL,NULL));
delete from user_sdo_geom_metadata
  where table_name = 'TAB_B' and column_name = 'LOCATION' ;
insert into user_sdo_geom_metadata
  values ('TAB_B','LOCATION', mdsys.sdo_dim_array(
    mdsys.sdo_dim_element('X',0.0,325000.0,0.5),
    mdsys.sdo_dim_element('Y',275000.0,625000.0,0.5)),NULL);
drop index TAB_B_1 force;
create index TAB_B_1 on TAB_B(LOCATION)
  indextype is mdsys.spatial_index
  parameters ('initial=1m next=1m pctincrease=0 sdo_fanout=32');
analyze table TAB_B_1_rt$ compute statistics;
analyze table TAB_B compute statistics;

drop table TAB_Q;
create table TAB_Q (GEOMETRY mdsys.sdo_geometry);
insert into TAB_Q (GEOMETRY) values (
  mdsys.SDO_GEOMETRY(2003,NULL,NULL,mdsys.SDO_ELEM_INFO_ARRAY(1,1003,3),
  mdsys.SDO_ORDINATE_ARRAY(78550,457900,78650,458025)));
```

```
delete from user_sdo_geom_metadata
  where table_name = 'TAB_Q' and column_name = 'GEOMETRY' ;
insert into user_sdo_geom_metadata
  values ('TAB_Q','GEOMETRY', mdsys.sdo_dim_array(
    mdsys.sdo_dim_element('X',0.0,325000.0,0.5),
    mdsys.sdo_dim_element('Y',275000.0,625000.0,0.5)),NULL);
drop index TAB_Q_1 force;
create index TAB_Q_1 on TAB_Q(GEOMETRY)
  indextype is mdsys.spatial_index
  parameters ('initial=1m next=1m pctincrease=0 sdo_fanout=32');
analyze table TAB_Q_1_rt$ compute statistics;
analyze table TAB_Q compute statistics;

drop view VIEW_AB;
create view VIEW_AB as
  select location, 'A' name from TAB_A
  union all select location, 'B' name from TAB_B;

select * from VIEW_AB h, TAB_Q q
where mdsys.sdo_relate(h.LOCATION, q.geometry,
  'mask=ANYINTERACT querytype = JOIN') = 'TRUE' ;
```

Problems with mixing 2D/3D operands

In case a spatial join is performed with data from a table `TAB_A` with 3D point data (with r-tree) and a table `TAB_Q` with 2D rectangle data using the operator `mdsys.sdo_relate` the following error message is given:

```
ERROR at line 1:
ORA-13050: unable to construct spatial object
ORA-06512: at "MDSYS.SDO_3GL", line 41
ORA-06512: at "MDSYS.MD2", line 722
ORA-06512: at "MDSYS.SDO_3GL", line 105
ORA-06512: at line 1
```

Below the script to set up the test environment:

```
drop table TAB_A;
create table TAB_A (LOCATION mdsys.sdo_geometry);
insert into TAB_A (LOCATION) values (
  mdsys.SDO_GEOMETRY(3001,NULL,
  mdsys.SDO_POINT_TYPE(223875,598875,123),NULL,NULL));
delete from user_sdo_geom_metadata
  where table_name = 'TAB_A' and column_name = 'LOCATION' ;
insert into user_sdo_geom_metadata
  values ('TAB_A','LOCATION', mdsys.sdo_dim_array(
    mdsys.sdo_dim_element('X',0.0,325000.0,0.5),
    mdsys.sdo_dim_element('Y',275000.0,625000.0,0.5),
    mdsys.sdo_dim_element('Z',-1000.0,1000.0,0.5)),NULL);
drop index TAB_A_1 force;
create index TAB_A_1 on TAB_A(LOCATION)
  indextype is mdsys.spatial_index
  parameters ('initial=1m next=1m pctincrease=0 sdo_fanout=32');
analyze table TAB_A_1_rt$ compute statistics;
analyze table TAB_A compute statistics;
```

```
drop table TAB_Q;
create table TAB_Q (GEOMETRY mdsys.sdo_geometry);
insert into TAB_Q (GEOMETRY) values (
  mdsys.SDO_GEOMETRY(2003,NULL,NULL,mdsys.SDO_ELEM_INFO_ARRAY(1,1003,3),
  mdsys.SDO_ORDINATE_ARRAY(78550,457900,300650,6008025)));
delete from user_sdo_geom_metadata
  where table_name = 'TAB_Q' and column_name = 'GEOMETRY' ;
insert into user_sdo_geom_metadata
  values ('TAB_Q','GEOMETRY', mdsys.sdo_dim_array(
    mdsys.sdo_dim_element('X',0.0,325000.0,0.5),
    mdsys.sdo_dim_element('Y',275000.0,625000.0,0.5)),NULL);
drop index TAB_Q_1 force;
create index TAB_Q_1 on TAB_Q(GEOMETRY)
  indextype is mdsys.spatial_index
  parameters ('initial=1m next=1m pctincrease=0 sdo_fanout=32');
analyze table TAB_Q_1_rt$ compute statistics;
analyze table TAB_Q compute statistics;

select * from TAB_A h, TAB_Q q
where mdsys.sdo_relate(h.LOCATION, q.geometry,
  'mask=ANYINTERACT querytype = JOIN') = 'TRUE' ;
```

Instead of using an operator (`mdsys.sdo_relate`) it is also possible to use a geometry function (`sdo_geom.relate`) with the same semantics. This query does not give an error message, but the result is also not correct as no rows are selected.

```
select * from TAB_A h, TAB_Q q
where sdo_geom.relate(h.LOCATION, 'querytype', q.geometry,0.5)='TRUE';
```

This problem has been reported in the last week of testing. No response from Oracle has been received.

Erroneous polygon-polygon distance computed

The distance between two polygons in not computed correctly. In the script below two polygons are defined (`triang` and `square`), both in counter-clockwise order and checked with the validate function (with result `TRUE`). However the last query of the script shows an incorrect polygon-polygon distance:

```
NAME       NAME       TRUE_DIST_M CENTR_DIST_M
---------- ---------- ----------- ------------
square     triang               0   .002173067
triang     square               0   .002173067
```

The script used to create test environment:

```
drop table TAB_Q;
create table TAB_Q (name varchar2(10), GEOMETRY mdsys.sdo_geometry);
insert into TAB_Q (name,GEOMETRY) values ('triang',
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(0,0, 1,0, 0,1, 0,0)));
insert into TAB_Q (name,GEOMETRY) values ('square',
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(2,0, 3,0, 3,1, 2,1, 2,0)));
```

```
select name, mdsys.SDO_GEOM.VALIDATE_GEOMETRY(geometry,0.5)
from TAB_Q;

select  q1.name, q2.name,
  SDO_GEOM.SDO_DISTANCE(q1.geometry,q2.geometry,0.5)/1000 true_dist_m,
  SDO_GEOM.SDO_DISTANCE(
  SDO_GEOM.SDO_CENTROID(q1.geometry,0.5),
  SDO_GEOM.SDO_CENTROID(q2.geometry,0.5),0.5)/1000 centr_dist_m
from TAB_Q q1, TAB_Q q2
where q1.rowid != q2.rowid;
```

It was also tried to use some clockwise data, but with the same result (except that the validate function does not return TRUE as result):

```
insert into TAB_Q (name,GEOMETRY) values ('triang-c',
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(0,0, 0,1, 1,0, 0,0)));
insert into TAB_Q (name,GEOMETRY) values ('square-c',
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(2,0, 2,1, 3,1, 3,0, 2,0)));
```

This problem has been reported in the last week of testing. No response from Oracle has been received.

Unexpected failure of sdo_intersection

A query used to compute the intersection of the province polygons (defined with between 1000 and 6000 points per polygon) does not return a proper result. The result should have been a set of polylines defining the boundary between the provinces (or sliver polygons in case the province definitions do overlap). Instead an error message is given after 40 minutes of computing:

```
SQL> create table polygon_intersect as
  2  select g1.name prov1_name, g2.name prov2_name,
  3  sdo_geom.sdo_intersection(g1.geometry, g2.geometry,0.5) int_geom
  4  from arcdata g1, arcdata g2
  5  where mdsys.sdo_relate (g1.geometry, g2.geometry,
  6    'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
  7       and g1.name not like 'kb%'
  8       and g2.name not like 'kb%'
  9       and g1.name <> g2.name;
create table polygon_intersect as
           *
ERROR at line 1:
ORA-03113: end-of-file on communication channel
```

This problem has been reported in the last week of testing. No response from Oracle has been received.

7 Conclusions and recommendations

7.1 Conclusions

Loading (spatial) data into Oracle is a smooth procedure which is performed at acceptable speed for spatial data and at high speed for non-spatial data. Oracle spatial tables require more storage than in Ingres: LKI tables require about 2 times more storage overall (and especially those containing lines or polygons). The storage requirements for spatial indices (r-tree) are 3 to 4 times larger in Oracle. Spatial index creation in Oracle (for tables with millions of rows) is very slow compared to Ingres.

Loading TNO data poses no special problems, the third dimension can be stored in Oracle Spatial objects (using that dimension in queries is not possible). The geological data sets are relatively small, only the horizons data takes a noticable amount of time to load.

Spatial queries return the same number of records as the Ingres queries do. During testing several bugs surfaced but it seems that these bugs could be corrected very well (and some have already been corrected in patches received from Oracle). There are many configuration (and loading and querying) parameters which can be, and sometimes have to be, specified. Changing parameters may result in new bugs, some of the bugs were discovered in this way. The problems encountered are summarized in Table 12 (Section 6) together with their current status and severity.

The performance (speed) of spatial queries, based on r-tree indices, is roughly the same for Oracle and Ingres (although it has to be said that during the test period benchmarking once again proved to be a business fraught with many pitfalls). In general the area queries of Ingres are about 2 times faster than the Oracle area queries due to less overhead within Ingres. On the other hand the Oracle polyline queries are 2 times faster than the ones in Ingres due to the fact that Oracle has very smart line-segment based r-tree indexing in performance patch 0115. Without this patch Oracle polyline queries are about 5 times slower than the ones in Ingres. Oracle patch 0115 (non-production) is a big improvement. Oracle performance appears acceptable for cadastral queries if seen from the perspective of future developments to improve the performance.

The performance of some of the queries requested by TNO-NITG is disappointing (using 'standard' Oracle Spatial functionality). It was demonstrated that improvements are possible (lack of time prevented finding an optimal solution) but this requires a substantial effort and expertise from the database user. By using the Oracle Spatial Java Library it is easy to create a JDBC connection to an Oracle database from a Java program.

To achieve speed Oracle needs clustering of spatial data. Having data stored in a random order on disk results in poor performance even in case a spatial index is used. Properly clustered data compared to randomly stored data results in a performance improvement of over a factor 30 for area range queries (for the same data model and same indices). Some limitations in the object-relational model of Oracle prohibit that the DBMS can guarantee spatial clustering at this moment. It relies on 'external' clustering and a loading process which preserves the clustering (so with updating the clustering will degrade over time). If Oracle could enforce proper clustering, e.g. by offering the possibility of index organized tables in combination with spatial indices, this would be a major improvement. This technique has also been applied to administrative data with dramatic effect (one or

two orders of magnitude faster geom-admin queries).

7.2 Recommendations

The functionality of Oracle 8i Spatial is sufficient for both Cadastre and TNO to start developments. The functionality is quite complete. Future functional improvements in the area of topology support are especially important for the Cadastre.

It is advised to the Cadastre to start the development of Oracle Spatial based systems as soon as possible, with production setting for the parameters (which should not be changed without serious testing). It is further advised that cadastral data becomes part of the regression tests at Oracle. So, whenever a new version is released cadastral data is used to check for correct behavior.

It is advised to TNO to investigate if the query performance, especially of the horizon data against the polygons defined with 1000 to 6000 points, is sufficient. If not, alternative indexing techniques must be investigated in more detail. In general the spatial functionality offered seems to be sufficient. However, besides performance also robustness of the functionality must be improved. For this purpose some TNO data should be given to Oracle together with the bug reports.

Both Oracle and Ingres are not OpenGIS compliant (SQL simple feature specification, types and functions). It is advised to both Oracle and CA to take the standards seriously as compliant alternatives are available, e.g. Informix. In general Oracle spatial functionality is richer than Ingres functionality. Further more active developments are going on in this area within Oracle. Some (simple) functions which are used in the current Ingres query tool environment are not yet available as standard in Oracle Spatial, e.g. counting the number of points in a polyline, selecting a point from a polyline and union of two bounding boxes. It is possible to define these functions but it is advised to ask Oracle to include them in Oracle Spatial. The most urgent performance improvement is spatial clustering (via index organized spatial tables).

References

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

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

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

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

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

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

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

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

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

Appendix A: Oracle database

This Appendix contains detailed information about the configuration of the Oracle database and the storage sizes of cadastral and geological tables and indices

Database creation

In this subsection the most relevant tablespace creation commands are listed. All through the testing period database parameters have been changing, these examples should not be interpreted as the most optimal database configuration for geological or cadastral data.

```
CREATE DATABASE "kadtest"
   maxdatafiles 62 maxinstances 2 maxlogfiles 8
DATAFILE '/r54/oradata/kadtest/system01.dbf' SIZE 260M AUTOEXTEND ON NEXT 8M
logfile '/r52/oradata/kadtest/redo01.log' SIZE 256M,
        '/r52/oradata/kadtest/redo02.log' SIZE 256M,
        '/r52/oradata/kadtest/redo03.log' SIZE 256M;

REM ********** TABLESPACE FOR ROLLBACK **********  2 Gb
CREATE TABLESPACE RBS DATAFILE '/r02/oradata/kadtest/rbs01.dbf' SIZE 2048M REUSE
 AUTOEXTEND ON NEXT 1M MINIMUM EXTENT 1M
DEFAULT STORAGE ( INITIAL 1M NEXT 1M MINEXTENTS 4 MAXEXTENTS UNLIMITED);

REM ********** TABLESPACE FOR TEMPORARY **********  2 Gb
CREATE TABLESPACE TEMP DATAFILE '/r02/oradata/kadtest/temp01.dbf' SIZE 2048M REUSE
 AUTOEXTEND ON NEXT 8M MINIMUM EXTENT 1M
DEFAULT STORAGE ( INITIAL 1M NEXT 1M MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0)
 TEMPORARY;

REM ********** TABLESPACE FOR USERS **********  4 Gb
CREATE TABLESPACE USERS DATAFILE '/r52/oradata/kadtest/users01.dbf' SIZE 4096M REUSE
 AUTOEXTEND ON NEXT 16M MINIMUM EXTENT 2M NOLOGGING
DEFAULT STORAGE ( INITIAL 2M NEXT 2M MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

REM ********** TABLESPACE FOR AKR tables **********  4 Gb
CREATE TABLESPACE AKR DATAFILE '/r52/oradata/kadtest/akr01.dbf' SIZE 4096M REUSE
 AUTOEXTEND ON NEXT 16M MINIMUM EXTENT 2M NOLOGGING
DEFAULT STORAGE ( INITIAL 2M NEXT 2M MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

REM ********** TABLESPACE FOR LKI tables **********  8 Gb
CREATE TABLESPACE LKI DATAFILE '/r54/oradata/kadtest/lki01.dbf' SIZE 8192M REUSE
 AUTOEXTEND ON NEXT 16M MINIMUM EXTENT 2M NOLOGGING
DEFAULT STORAGE ( INITIAL 2M NEXT 2M MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

REM ********** TABLESPACE FOR INDEXES **********  8 Gb
CREATE TABLESPACE INDX DATAFILE '/r02/oradata/kadtest/indx01.dbf' SIZE 8192M REUSE
 AUTOEXTEND ON NEXT 16M MINIMUM EXTENT 2M NOLOGGING
DEFAULT STORAGE ( INITIAL 2M NEXT 2M MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);

REM ********** TABLESPACE FOR TNO data **********  1 Gb
CREATE TABLESPACE TNO DATAFILE '/r52/oradata/kadtest/tno.dbf' SIZE 1024M REUSE
 AUTOEXTEND ON NEXT 8M MINIMUM EXTENT 1M NOLOGGING
DEFAULT STORAGE ( INITIAL 1M NEXT 1M MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);
```

Database parameters

The following initialization file was used for the test database:

```
# initkadtest.ora 28-01-2001

db_name =      "kadtest"
instance_name = kadtest
service_names = kadtest,kadtest.geo.tudelft.nl

control_files = ("/r02/oradata/kadtest/control01.ctl",
                 "/r52/oradata/kadtest/control02.ctl",
                 "/r54/oradata/kadtest/control03.ctl")
processes = 30
max_enabled_roles = 30

shared_pool_size =      142606336     # 136 Mb
log_buffer =              1048576     #   1 Mb
large_pool_size =         1048576     #   1 Mb
java_pool_size =         16777216     #  16 Mb
sort_area_size =          8388608     #   8 Mb
sort_area_retained_size = 1048576     #   1 Mb

db_block_size =     8192              #   8 Kb
db_block_buffers = 30720              # 240 Mb
#dbwr_io_slaves =       4

open_cursors =           300
#session_cached_cursors = 300      # for improved R-tree performance;
# does not work in combination with direct path loading (bug 931820),
# so switched off for now: set explicitly in query scripts
cursor_space_for_time = true      # for improved R-tree performance

# The following parameters are appropriate for mass loading
# spatial data, should be changed for production
fast_start_io_target =        0
log_checkpoint_timeout =      0
log_checkpoint_interval =     0
log_checkpoints_to_alert = true

# define directories to store trace and alert files
background_dump_dest = /opt/oracle/admin/kadtest/bdump
core_dump_dest =       /opt/oracle/admin/kadtest/cdump
user_dump_dest =       /opt/oracle/admin/kadtest/udump

max_dump_file_size = 10M     # limit trace file size to 10 Mb each

remote_login_passwordfile = exclusive
os_authent_prefix = ""

compatible = "8.1.0"

# Uncomment the following line for a multi-threaded server
#mts_dispatchers = "(PROTOCOL=TCP)(PRE=oracle.aurora.server.SGiopServer)"

# end-of-initfile
```

Sizes of cadastral tables and indices

In the following overviews formatting commands are not included.

```
SQL> select tablespace_name "TSpace", table_name, (blocks*8/1024) "Size_Mb",
  2       blocks "Blocks", empty_blocks "Empty", Num_rows "Num_rows",
  3       chain_cnt "Chained",avg_row_len "AvRowLen",avg_space "FrSpace"
  4  from user_tables
  5  order by table_name
  6  /
```

TSpace	TABLE_NAME	Size_Mb	Blocks	Empty	Num_rows	Chained	AvRowLen	FrSpace
AKR	MO_GROEPSRELATIE	2.07	265	246	22722	0	82	900
AKR	MO_HUWELIJKSRELATIE	81.28	10404	347	1081956	0	68	822
	MO_OBJECT				2386641	0	159	0
AKR	MO_OBJECTADRES	436.03	55812	507	3185769	0	125	873
AKR	MO_OBJECTMUTATIE	.00	0	511	0	0	0	0
AKR	MO_OBJ_BELEMMERING	68.81	8808	407	621602	0	101	859
AKR	MO_ONTSTAAN_UIT	164.97	21116	387	1359908	0	111	845
AKR	MO_ONZELFSTANDIG_DEEL	.01	1	510	2	0	115	7868
AKR	MO_OVERGEGAAN_IN	71.74	9183	32	596695	0	110	824
AKR	MO_RECHT	506.88	64881	142	3197039	0	144	891
AKR	MO_RECHTBELEMMERING	33.36	4270	337	281558	0	107	915
AKR	MO_REG_9	6.10	781	242	60876	0	91	890
AKR	MO_REG_9_TEKST	.00	0	511	0	0	0	0
AKR	MO_RENTE	13.95	1785	262	134604	0	94	876
	MO_SUBJECT				2264017	0	176	0
AKR	MO_SUBJECTMUTATIES	.00	0	511	0	0	0	0
AKR	MO_SUBJECTRELATIE	10.22	1308	227	133016	0	69	882
	OBJECT_PARCEL				2684254	0	38	0
AKR	SYS_IOT_OVER_17894	.00	0	511	0	0	0	0
AKR	SYS_IOT_OVER_17950	.00	0	511	0	0	0	0
INDX	XFIOX_BOUNDARY_0A_RT$	.01	1	510	1	0	3696	4404
INDX	XFIOX_BOUNDARY_0B_RT$	688.38	88113	462	176221	0	3698	702
INDX	XFIOX_BOUNDARY_0_RT$	688.38	88113	462	176221	0	3698	702
INDX	XFIOX_GCPNT_0_RT$	4.19	536	487	1072	0	3697	704
INDX	XFIOX_LINE_0_RT$	240.03	30724	507	61446	0	3698	703
INDX	XFIOX_PARCELOVER_0_RT$	2.97	380	131	753	0	3697	772
INDX	XFIOX_PARCEL_0_RT$	261.87	33519	272	67032	0	3698	703
INDX	XFIOX_SYMPNT_0_RT$	140.81	18024	407	36045	0	3698	703
INDX	XFIOX_TEXT_0_RT$	112.42	14390	457	28776	0	3698	704
LKI	XFIO_BOUNDARY	4769.20	610457	869	10044511	0	429	1009
LKI	XFIO_GCPNT	9.44	1208	839	60986	0	141	881
LKI	XFIO_LINE	1332.45	170553	454	3502313	0	344	991
LKI	XFIO_PARCEL	1151.41	147381	74	3820699	0	277	867
LKI	XFIO_PARCELOVER	9.59	1228	819	42852	0	203	935
LKI	XFIO_SYMPNT	203.94	26104	519	2054463	0	90	845
LKI	XFIO_TEXT	170.06	21768	759	1640122	0	94	859

```
36 rows selected.
```

```
SQL> select index_type, index_name, leaf_blocks*8/1024 "Mb_in_Leaf_blk",
  2       table_name, tablespace_name "Tablespace"
  3  from user_indexes
  4  where (index_type != 'LOB')
  5  order by table_name, index_name
  6  /

INDEX_TYPE INDEX_NAME                     Mb_in_Leaf_blk TABLE_NAME            Tablespace
---------- ------------------------------ -------------- ---------------------- ----------
NORMAL     IDX_MO_GROEPSRELATIE_MAIN                .55 MO_GROEPSRELATIE       INDX
NORMAL     IDX_MO_HUWELIJKSRELATIE_MAIN           25.93 MO_HUWELIJKSRELATIE    INDX
NORMAL     IDX_MO_OBJECT_1                       135.12 MO_OBJECT              INDX
IOT - TOP  SYS_IOT_TOP_17875                     380.80 MO_OBJECT              AKR
NORMAL     IDX_MO_OBJECTADRES_1                  114.70 MO_OBJECTADRES         INDX
NORMAL     IDX_MO_OBJECTADRES_2                   48.38 MO_OBJECTADRES         INDX
NORMAL     IDX_MO_OBJECTADRES_3                   73.64 MO_OBJECTADRES         INDX
NORMAL     IDX_MO_OBJECTADRES_4                   77.38 MO_OBJECTADRES         INDX
NORMAL     IDX_MO_OBJECTADRES_MAIN               100.77 MO_OBJECTADRES         INDX
NORMAL     IDX_MO_OBJECTMUTATIE_1                   .00 MO_OBJECTMUTATIE       INDX
NORMAL     IDX_MO_OBJECTMUTATIE_MAIN                .00 MO_OBJECTMUTATIE       INDX
NORMAL     IDX_MO_OBJ_BELEMMERING_1               22.38 MO_OBJ_BELEMMERING     INDX
NORMAL     IDX_MO_OBJ_BELEMMERING_MAIN            19.66 MO_OBJ_BELEMMERING     INDX
NORMAL     IDX_MO_ONTSTAAN_UIT_1                  48.96 MO_ONTSTAAN_UIT        INDX
NORMAL     IDX_MO_ONTSTAAN_UIT_2                  43.02 MO_ONTSTAAN_UIT        INDX
NORMAL     IDX_MO_ONTSTAAN_UIT_MAIN               38.16 MO_ONTSTAAN_UIT        INDX
NORMAL     IDX_MO_ONZELFSTANDIG_DEEL_1              .01 MO_ONZELFSTANDIG_DEEL  INDX
NORMAL     IDX_MO_ONZELFSTANDIG_DEEL_MAIN           .01 MO_ONZELFSTANDIG_DEEL  INDX
NORMAL     IDX_MO_OVERGEGAAN_IN_1                 21.48 MO_OVERGEGAAN_IN       INDX
NORMAL     IDX_MO_OVERGEGAAN_IN_MAIN              18.88 MO_OVERGEGAAN_IN       INDX
NORMAL     IDX_MO_RECHT_1                        115.10 MO_RECHT               INDX
NORMAL     IDX_MO_RECHT_2                         76.62 MO_RECHT               INDX
NORMAL     IDX_MO_RECHT_MAIN                     101.13 MO_RECHT               INDX
NORMAL     IDX_MO_RECHTBELEMMERING_1              10.14 MO_RECHTBELEMMERING    INDX
NORMAL     IDX_MO_RECHTBELEMMERING_MAIN            8.91 MO_RECHTBELEMMERING    INDX
NORMAL     IDX_MO_REG_9_1                          2.20 MO_REG_9               INDX
NORMAL     IDX_MO_REG_9_MAIN                       1.93 MO_REG_9               INDX
NORMAL     IDX_MO_REG_9_TEKST_1                     .00 MO_REG_9_TEKST         INDX
NORMAL     IDX_MO_REG_9_TEKST_MAIN                  .00 MO_REG_9_TEKST         INDX
NORMAL     IDX_MO_RENTE_1                          4.85 MO_RENTE               INDX
NORMAL     IDX_MO_RENTE_MAIN                       4.26 MO_RENTE               INDX
NORMAL     IDX_MO_SUBJECT_10                       1.60 MO_SUBJECT             INDX
NORMAL     IDX_MO_SUBJECT_2                       98.84 MO_SUBJECT             INDX
NORMAL     IDX_MO_SUBJECT_3                       97.84 MO_SUBJECT             INDX
NORMAL     IDX_MO_SUBJECT_4                       78.85 MO_SUBJECT             INDX
NORMAL     IDX_MO_SUBJECT_5                       10.89 MO_SUBJECT             INDX
NORMAL     IDX_MO_SUBJECT_6                        4.75 MO_SUBJECT             INDX
NORMAL     IDX_MO_SUBJECT_7                        4.04 MO_SUBJECT             INDX
NORMAL     IDX_MO_SUBJECT_8                        3.37 MO_SUBJECT             INDX
NORMAL     IDX_MO_SUBJECT_9                        2.69 MO_SUBJECT             INDX
IOT - TOP  SYS_IOT_TOP_17894                     397.20 MO_SUBJECT             INDX
NORMAL     IDX_MO_SUBJECTMUTATIES_MAIN              .00 MO_SUBJECTMUTATIES     INDX
NORMAL     IDX_MO_SUBJECTRELATIE_MAIN              3.20 MO_SUBJECTRELATIE      INDX
NORMAL     IDX_OBJECT_PARCEL_0                   194.18 OBJECT_PARCEL          INDX
IOT - TOP  PK                                    174.43 OBJECT_PARCEL          INDX
DOMAIN     XFIOX_BOUNDARY_0                             XFIO_BOUNDARY         SYSTEM
DOMAIN     XFIOX_BOUNDARY_0A                            XFIO_BOUNDARY         SYSTEM
```

```
INDEX_TYPE INDEX_NAME                     Mb_in_Leaf_blk TABLE_NAME            Tablespace
---------- ------------------------------ -------------- ---------------------- ----------
DOMAIN     XFIOX_BOUNDARY_0B                             XFIO_BOUNDARY          SYSTEM
NORMAL     XFIOX_BOUNDARY_10                     417.58 XFIO_BOUNDARY          INDX
NORMAL     XFIOX_BOUNDARY_1B                     196.70 XFIO_BOUNDARY          INDX
NORMAL     XFIOX_BOUNDARY_2                      337.43 XFIO_BOUNDARY          INDX
NORMAL     XFIOX_BOUNDARY_3                      337.41 XFIO_BOUNDARY          INDX
NORMAL     XFIOX_BOUNDARY_4                      195.66 XFIO_BOUNDARY          INDX
NORMAL     XFIOX_BOUNDARY_5                      195.66 XFIO_BOUNDARY          INDX
NORMAL     XFIOX_BOUNDARY_6                      271.05 XFIO_BOUNDARY          INDX
NORMAL     XFIOX_BOUNDARY_7                      153.27 XFIO_BOUNDARY          INDX
NORMAL     XFIOX_BOUNDARY_8                      273.27 XFIO_BOUNDARY          INDX
NORMAL     XFIOX_BOUNDARY_9                      273.28 XFIO_BOUNDARY          INDX
DOMAIN     XFIOX_GCPNT_0                                XFIO_GCPNT             SYSTEM
NORMAL     XFIOX_GCPNT_1                           1.47 XFIO_GCPNT             INDX
DOMAIN     XFIOX_LINE_0                                 XFIO_LINE              SYSTEM
NORMAL     XFIOX_LINE_1                           80.02 XFIO_LINE              INDX
DOMAIN     XFIOX_PARCEL_0                               XFIO_PARCEL            SYSTEM
NORMAL     XFIOX_PARCEL_1B                        74.83 XFIO_PARCEL            INDX
NORMAL     XFIOX_PARCEL_2                        137.55 XFIO_PARCEL            INDX
NORMAL     XFIOX_PARCEL_3                        120.85 XFIO_PARCEL            INDX
DOMAIN     XFIOX_PARCELOVER_0                           XFIO_PARCELOVER        SYSTEM
NORMAL     XFIOX_PARCELOVER_1                       .98 XFIO_PARCELOVER        INDX
DOMAIN     XFIOX_SYMPNT_0                               XFIO_SYMPNT            SYSTEM
NORMAL     XFIOX_SYMPNT_1                         46.94 XFIO_SYMPNT            INDX
DOMAIN     XFIOX_TEXT_0                                 XFIO_TEXT              SYSTEM
NORMAL     XFIOX_TEXT_1                           37.47 XFIO_TEXT              INDX
NORMAL     XFIOX_TEXT_2                           27.57 XFIO_TEXT              INDX

73 rows selected.

SQL> select segment_name, COUNT(*) extents, sum(bytes/(1024*1024)) size_in_mb,
  2       sum(blocks) "Blocks", tablespace_name "Tablespace"
  3  from user_extents
  4  where (segment_name not LIKE 'SYS_IL000%' and segment_name not LIKE 'SYS_LOB000%')
  5  GROUP BY SEGMENT_NAME, tablespace_name
  6  order BY SEGMENT_NAME
  7  /

SEGMENT_NAME                      EXTENTS SIZE_IN_MB  Blocks Tablespace
------------------------------ ---------- ---------- ------- ----------
IDX_MO_GROEPSRELATIE_MAIN               1          4     512 INDX
IDX_MO_HUWELIJKSRELATIE_MAIN            7         28    3584 INDX
IDX_MO_OBJECTADRES_1                   29        116   14848 INDX
IDX_MO_OBJECTADRES_2                   13         52    6656 INDX
IDX_MO_OBJECTADRES_3                   19         76    9728 INDX
IDX_MO_OBJECTADRES_4                   20         80   10240 INDX
IDX_MO_OBJECTADRES_MAIN                26        104   13312 INDX
IDX_MO_OBJECTMUTATIE_1                  1          4     512 INDX
IDX_MO_OBJECTMUTATIE_MAIN               1          4     512 INDX
IDX_MO_OBJECT_1                        34        136   17408 INDX
IDX_MO_OBJ_BELEMMERING_1                6         24    3072 INDX
IDX_MO_OBJ_BELEMMERING_MAIN             5         20    2560 INDX
IDX_MO_ONTSTAAN_UIT_1                  13         52    6656 INDX
IDX_MO_ONTSTAAN_UIT_2                  11         44    5632 INDX
IDX_MO_ONTSTAAN_UIT_MAIN               10         40    5120 INDX
```

SEGMENT_NAME	EXTENTS	SIZE_IN_MB	Blocks	Tablespace
IDX_MO_ONZELFSTANDIG_DEEL_1	1	4	512	INDX
IDX_MO_ONZELFSTANDIG_DEEL_MAIN	1	4	512	INDX
IDX_MO_OVERGEGAAN_IN_1	6	24	3072	INDX
IDX_MO_OVERGEGAAN_IN_MAIN	5	20	2560	INDX
IDX_MO_RECHTBELEMMERING_1	3	12	1536	INDX
IDX_MO_RECHTBELEMMERING_MAIN	3	12	1536	INDX
IDX_MO_RECHT_1	29	116	14848	INDX
IDX_MO_RECHT_2	20	80	10240	INDX
IDX_MO_RECHT_MAIN	26	104	13312	INDX
IDX_MO_REG_9_1	1	4	512	INDX
IDX_MO_REG_9_MAIN	1	4	512	INDX
IDX_MO_REG_9_TEKST_1	1	4	512	INDX
IDX_MO_REG_9_TEKST_MAIN	1	4	512	INDX
IDX_MO_RENTE_1	2	8	1024	INDX
IDX_MO_RENTE_MAIN	2	8	1024	INDX
IDX_MO_SUBJECTMUTATIES_MAIN	1	4	512	INDX
IDX_MO_SUBJECTRELATIE_MAIN	1	4	512	INDX
IDX_MO_SUBJECT_10	1	4	512	INDX
IDX_MO_SUBJECT_2	25	100	12800	INDX
IDX_MO_SUBJECT_3	25	100	12800	INDX
IDX_MO_SUBJECT_4	20	80	10240	INDX
IDX_MO_SUBJECT_5	3	12	1536	INDX
IDX_MO_SUBJECT_6	2	8	1024	INDX
IDX_MO_SUBJECT_7	2	8	1024	INDX
IDX_MO_SUBJECT_8	1	4	512	INDX
IDX_MO_SUBJECT_9	1	4	512	INDX
IDX_OBJECT_PARCEL_0	49	196	25088	INDX
MO_GROEPSRELATIE	1	4	512	AKR
MO_HUWELIJKSRELATIE	21	84	10752	AKR
MO_OBJECTADRES	110	440	56320	AKR
MO_OBJECTMUTATIE	1	4	512	AKR
MO_OBJ_BELEMMERING	18	72	9216	AKR
MO_ONTSTAAN_UIT	42	168	21504	AKR
MO_ONZELFSTANDIG_DEEL	1	4	512	AKR
MO_OVERGEGAAN_IN	18	72	9216	AKR
MO_RECHT	127	508	65024	AKR
MO_RECHTBELEMMERING	9	36	4608	AKR
MO_REG_9	2	8	1024	AKR
MO_REG_9_TEKST	1	4	512	AKR
MO_RENTE	4	16	2048	AKR
MO_SUBJECTMUTATIES	1	4	512	AKR
MO_SUBJECTRELATIE	3	12	1536	AKR
PK	44	176	22528	INDX
SYS_IOT_OVER_17894	1	4	512	AKR
SYS_IOT_OVER_17950	1	4	512	AKR
SYS_IOT_TOP_17875	96	384	49152	AKR
SYS_IOT_TOP_17894	100	400	51200	INDX
XFIOX_BOUNDARY_0A_RT$	1	4	512	INDX
XFIOX_BOUNDARY_0B_RT$	173	692	88576	INDX
XFIOX_BOUNDARY_0_RT$	173	692	88576	INDX
XFIOX_BOUNDARY_10	105	420	53760	INDX
XFIOX_BOUNDARY_1B	50	200	25600	INDX
XFIOX_BOUNDARY_2	85	340	43520	INDX
XFIOX_BOUNDARY_3	85	340	43520	INDX

```
SEGMENT_NAME                    EXTENTS SIZE_IN_MB  Blocks Tablespace
------------------------------ ---------- ---------- ------- ----------
XFIOX_BOUNDARY_4                    50        200   25600 INDX
XFIOX_BOUNDARY_5                    50        200   25600 INDX
XFIOX_BOUNDARY_6                    68        272   34816 INDX
XFIOX_BOUNDARY_7                    39        156   19968 INDX
XFIOX_BOUNDARY_8                    69        276   35328 INDX
XFIOX_BOUNDARY_9                    69        276   35328 INDX
XFIOX_GCPNT_0_RT$                    2          8    1024 INDX
XFIOX_GCPNT_1                        1          4     512 INDX
XFIOX_LINE_0_RT$                    61        244   31232 INDX
XFIOX_LINE_1                        21         84   10752 INDX
XFIOX_PARCELOVER_0_RT$               1          4     512 INDX
XFIOX_PARCELOVER_1                   1          4     512 INDX
XFIOX_PARCEL_0_RT$                  66        264   33792 INDX
XFIOX_PARCEL_1B                     19         76    9728 INDX
XFIOX_PARCEL_2                      35        140   17920 INDX
XFIOX_PARCEL_3                      31        124   15872 INDX
XFIOX_SYMPNT_0_RT$                  36        144   18432 INDX
XFIOX_SYMPNT_1                      12         48    6144 INDX
XFIOX_TEXT_0_RT$                    29        116   14848 INDX
XFIOX_TEXT_1                        10         40    5120 INDX
XFIOX_TEXT_2                         7         28    3584 INDX
XFIO_BOUNDARY                      597       4776  611328 LKI
XFIO_GCPNT                           2         16    2048 LKI
XFIO_LINE                          167       1336  171008 LKI
XFIO_PARCEL                        144       1152  147456 LKI
XFIO_PARCELOVER                      2         16    2048 LKI
XFIO_SYMPNT                         26        208   26624 LKI
XFIO_TEXT                           22        176   22528 LKI

97 rows selected.
```

Sizes of geological tables and indices

```
SQL> select tablespace_name "TSpace", table_name, (blocks*8/1024) "Size_Mb",
  2       blocks "Blocks", empty_blocks "Empty", Num_rows "Num_rows",
  3       chain_cnt "Chained",avg_row_len "AvRowLen",avg_space "FrSpace"
  4  from user_tables order by table_name;

TSpace TABLE_NAME               Size_Mb  Blocks Empty  Num_rows Chained AvRowLen FrSpace
------ ------------------------ -------- ------- ----- --------- ------- -------- -------
TNO    ARCDATA                       .01       1   126        24       0      169    4006
TNO    ARCDATA_IDX_RT$               .01       1   126         1       0     2072    6028
TNO    FAULTLINES_12BR_BR            .12      15   112       205       0      381    2864
TNO    FAULTLINES_12BR__1_RT$        .02       3   124         9       0     1904    2384
TNO    FAULTLINES_19NO_BR            .23      30    97       574       0      347    1416
TNO    FAULTLINES_19NO__1_RT$        .08      10   117        21       0     1904    4101
TNO    FAULTLINES_24TX_BR            .20      25   102       409       0      384    1787
TNO    FAULTLINES_24TX__1_RT$        .03       4   123        16       0     1904     478
TNO    FAULTLINES_29RIJN_BR          .12      15   112       143       0      569    2656
TNO    FAULTLINES_29RIJ_1_RT$        .02       2   125         6       0     1904    2384
TNO    FAULTLINES_30DF_BR            .08      10   117       105       0      589    1895
TNO    FAULTLINES_30DF__1_RT$        .02       2   125         5       0     1904    3337
```

TSpace	TABLE_NAME	Size_Mb	Blocks	Empty	Num_rows	Chained	AvRowLen	FrSpace
TNO	FAULTLINES_40AL_BR	.16	20	107	198	0	642	1727
TNO	FAULTLINES_40AL__1_RT$	.02	2	125	8	0	1904	478
TNO	FAULTLINES_49BO_BR	.27	35	92	357	0	622	1733
TNO	FAULTLINES_49BO__1_RT$	.03	4	123	14	0	1904	1431
TNO	FAULTLINES_54ZE_BR	.74	95	32	1124	0	567	1375
TNO	FAULTLINES_54ZE__1_RT$	.12	15	112	42	0	1904	2766
TNO	FAULTLINES_59RO_BR	.63	80	47	986	0	557	1215
TNO	FAULTLINES_59RO__1_RT$	.08	10	117	37	0	1904	1050
TNO	FAULTLINES_TZE_BR	.31	40	87	714	0	386	1185
TNO	FAULTLINES_TZE_B_1_RT$	.08	10	117	26	0	1904	3148
TNO	HORIZONS_19B_TERT250	10.42	1334	73	334720	0	27	844
TNO	HORIZONS_19B_TER_1_RT$	23.38	2993	78	11957	0	1905	483
TNO	HORIZONS_24B_UCRET250	8.73	1118	33	281088	0	27	838
TNO	HORIZONS_24B_UCR_1_RT$	19.62	2511	48	10041	0	1905	476
TNO	HORIZONS_29B_LCRET250	9.07	1161	118	291413	0	27	849
TNO	HORIZONS_29B_LCR_1_RT$	20.34	2604	83	10409	0	1905	479
TNO	HORIZONS_30B_UJURA250	1.85	237	18	59466	0	27	850
TNO	HORIZONS_30B_UJU_1_RT$	4.19	536	103	2126	0	1905	538
TNO	HORIZONS_40B_LJURA250	1.97	252	3	62509	0	27	933
TNO	HORIZONS_40B_LJU_1_RT$	4.38	561	78	2235	0	1905	505
TNO	HORIZONS_49B_LTRIAS250	6.54	837	58	210632	0	27	829
TNO	HORIZONS_49B_LTR_1_RT$	14.73	1886	33	7525	0	1905	493
TNO	HORIZONS_50T_ZECH250	8.97	1148	3	288955	0	27	828
TNO	HORIZONS_50T_ZEC_1_RT$	20.19	2584	103	10321	0	1905	485
TNO	HORIZONS_54B_ZECH250	8.97	1148	3	288945	0	27	831
TNO	HORIZONS_54B_ZEC_1_RT$	20.19	2584	103	10321	0	1905	485
TNO	HORIZONS_59B_ROTL250	8.50	1088	63	272993	0	27	854
TNO	HORIZONS_59B_ROT_1_RT$	19.07	2441	118	9752	0	1905	483
TNO	NAVSEIS	2.34	300	83	66558	0	30	895
TNO	NAVSEIS_1_RT$	4.66	596	43	2380	0	1905	487
TNO	POLYGON_INTERSECT							
TNO	SUBCROPS250_19NO_1_RT$	.01	1	126	3	0	1904	2384
TNO	SUBCROPS250_19NO_SC	.08	10	117	51	0	1088	2543
TNO	SUBCROPS250_24TX_1_RT$	.01	1	126	1	0	1904	6196
TNO	SUBCROPS250_24TX_SC	.04	5	122	26	0	1068	2536
TNO	SUBCROPS250_29RIJN_SC	.08	10	117	29	0	1177	4686
TNO	SUBCROPS250_29RI_1_RT$	.01	1	126	1	0	1904	6196
TNO	SUBCROPS250_30DF_1_RT$	.01	1	126	3	0	1904	2384
TNO	SUBCROPS250_30DF_SC	.08	10	117	36	0	1030	4388
TNO	SUBCROPS250_40AL_1_RT$	.01	1	126	4	0	1904	478
TNO	SUBCROPS250_40AL_SC	.12	15	112	78	0	961	3095
TNO	SUBCROPS250_49BO_1_RT$	.01	1	126	4	0	1904	478
TNO	SUBCROPS250_49BO_SC	.08	10	117	62	0	949	2205
TNO	SUBCROPS250_54ZE_1_RT$	.01	1	126	4	0	1904	478
TNO	SUBCROPS250_54ZE_SC	.03	4	123	61	0	365	2509
TNO	SUBCROPS250_59RO_1_RT$	.01	1	126	1	0	1904	6196
TNO	SUBCROPS250_59RO_SC	.03	4	123	20	0	1043	2875
TNO	SUBCROPS250_TZE_SC	.08	10	117	85	0	621	2809
TNO	SUBCROPS250_TZE__1_RT$	.01	1	126	4	0	1904	478
TNO	T_FAULTLINES	2.78	356	27	4815	0	504	1259
TNO	T_HORIZONS	97.73	12509	34	2090721	0	41	843
TNO	T_HORIZONS_IDX_RT$	145.85	18669	18	74670	0	1906	471
TNO	WELLPICKS	.16	20	107	2790	0	41	2158
TNO	WELLPICKS_1_RT$	.23	30	97	102	0	1904	1622

```
66 rows selected.

SQL> select index_type, index_name, leaf_blocks*8/1024 "Mb_in_Leaf_blk",
  2       table_name, tablespace_name "Tablespace"
  3  from user_indexes
  4  where (index_type != 'DOMAIN' and index_type != 'LOB')
  5  order by table_name, index_name;

INDEX_TYPE INDEX_NAME                     Mb_in_Leaf_blk TABLE_NAME            Tablespace
---------- ------------------------------ -------------- ----------------------- ----------
NORMAL     NAVSEIS_2                                1.28 NAVSEIS                 TNO
NORMAL     NAVSEIS_3                                1.09 NAVSEIS                 TNO
NORMAL     WELLPICKS_2                               .05 WELLPICKS               TNO
```

Appendix B: Oracle cadastral data loading

LKI table definitions

```
create table xfio_boundary
(
 ogroup      number(11)         not null,
 object_id   number(11)         not null,
 slc         number(11)         not null,
 classif     number(11)         not null,
 interp_cd   number(4)          not null,
 shape       mdsys.sdo_geometry not null,  -- type 2 met interpretatie 1 (max 50 XY)
 height      number(11)         not null,
 node_cd     number(4)          not null,
 status_cd   number(11)         not null,
 fl_line_id  number(11)         not null,
 fr_line_id  number(11)         not null,
 ll_line_id  number(11)         not null,
 lr_line_id  number(11)         not null,
 l_obj_id    number(11)         not null,
 r_obj_id    number(11)         not null,
 accu_cd     number(11)         not null,
 bbox        mdsys.sdo_geometry         ,  -- type 3 met interpretatie 3 (lo+rb)
 abox        mdsys.sdo_geometry         ,  -- type 3 met interpretatie 3 (lo+rb)
 linelen     number(11)         not null,
 object_dt   number(11)         not null,
 tmin        number(11)         not null,
 tmax        number(11)         not null,
 sel_cd      varchar2(3)                ,
 source      varchar2(5)                ,
 quality     varchar2(2)                ,
 vis_cd      varchar2(1)        not null,
 l_municip   varchar2(5)                ,
 l_section   varchar2(2)                ,
 l_sheet     varchar2(4)                ,
 l_parcel    varchar2(5)                ,
 l_pp_i_ltr  varchar2(1)                ,
 l_pp_i_nr   varchar2(4)                ,
 r_municip   varchar2(5)                ,
 r_section   varchar2(2)                ,
 r_sheet     varchar2(4)                ,
 r_parcel    varchar2(5)                ,
 r_pp_i_ltr  varchar2(1)                ,
 r_pp_i_nr   varchar2(6)
)
 storage (initial 8m next 8m pctincrease 0)
 tablespace lki pctfree 10 nologging
;

create table xfio_parcel
(
 ogroup             number(11)         not null,
 object_id          number(11)         not null,
 slc                number(11)         not null,
 classif            number(11)         not null,
```

```
 location           mdsys.sdo_geometry not null,  -- type 1 point
 z                  number(11)         not null,
 d_location         mdsys.sdo_geometry not null,  -- type 1 point
 rotangle           number(11)         not null,
 accu_cd            number(11)         not null,
 oarea              float              not null,
 bbox               mdsys.sdo_geometry not null,  -- type 3 box
 object_dt          number(11)         not null,
 tmin               number(11)         not null,
 tmax               number(11)         not null,
 sel_cd             varchar2(3)                ,
 source             varchar2(5)                ,
 quality            varchar2(2)                ,
 vis_cd             varchar2(1)                ,
 akr_area           float              not null,
 municip            varchar2(5)        not null,
 osection           varchar2(2)        not null,
 sheet              varchar2(4)        not null,
 parcel             varchar2(5)        not null,
 pp_i_ltr           varchar2(1)        not null,
 pp_i_nr            varchar2(4)        not null,
 l_num              number(11)         not null,
 line_id1           number(11)         not null,
 line_id2           number(11)         not null,
 x_akr_objectnummer varchar2(17)       not null
)
 storage (initial 8m next 8m pctincrease 0)
 tablespace lki pctfree 10 nologging
;

create table xfio_text
(
 ogroup     number(11)         not null,  -- Group Id (KEY.1) ->CLASS.1
 object_id  number(11)         not null,  -- text/polygon object Id (KEY.2)
 slc        number(11)         not null,  -- slc code
 classif    number(11)         not null,  -- Object class code ->CLASS.2
 location   mdsys.sdo_geometry not null,  -- x and y-coordinate (on map) (point)
 z          number(11)         not null,  -- z-coordinate
 d_location mdsys.sdo_geometry not null,  -- Delta x and y of pointnum (point)
 rotangle   number(11)         not null,  -- Rotation angle
 accu_cd    number(11)         not null,  -- Accuracy code
 textlen    number(4)          not null,  -- text length (num of chs)
 otext      varchar2(80)                ,  -- object text
 object_dt  number(11)         not null,  -- Date (of measurement; user time)
 tmin       number(11)         not null,  -- Time created/last updated
 tmax       number(11)         not null,  -- Time deleted/last updated
 sel_cd     varchar2(3)        not null,  -- Belongs to map: cadast, GKBN...
 source     varchar2(5)                ,  -- Source of data
 quality    varchar2(2)                ,  -- Data quality: method, accuracy
 vis_cd     varchar2(1)                   -- Visibility code
)
 storage (initial 8m next 8m pctincrease 0)
 tablespace lki pctfree 10 nologging
;
```

Secondary indices

Before spatial indices can be created metadata has to be entered in the `user_sdo_geom_metadata` table for all spatial columns (as explained in Section 6 no useful primary storage structure can be specified for spatial data in Oracle). For the example tables the commands are:

```
insert into user_sdo_geom_metadata
  values ('xfio_boundary','shape',
  mdsys.sdo_dim_array(mdsys.sdo_dim_element('X',-25000000,325000000,0.5),
                      mdsys.sdo_dim_element('Y',275000000,625000000,0.5)),NULL);

insert into user_sdo_geom_metadata
  values ('xfio_boundary','bbox',
  mdsys.sdo_dim_array(mdsys.sdo_dim_element('X',-25000000,325000000,0.5),
                      mdsys.sdo_dim_element('Y',275000000,625000000,0.5)),NULL);

insert into user_sdo_geom_metadata
  values ('xfio_boundary','abox',
  mdsys.sdo_dim_array(mdsys.sdo_dim_element('X',-25000000,325000000,0.5),
                      mdsys.sdo_dim_element('Y',275000000,625000000,0.5)),NULL);

insert into user_sdo_geom_metadata
values ('xfio_parcel','location',
  mdsys.sdo_dim_array(mdsys.sdo_dim_element('X',-25000000,325000000,0.5),
                      mdsys.sdo_dim_element('Y',275000000,625000000,0.5)),NULL);
        .
        .
        .
insert into user_sdo_geom_metadata
  values ('xfio_text','d_location',
  mdsys.sdo_dim_array(mdsys.sdo_dim_element('X',-25000000,325000000,0.5),
                      mdsys.sdo_dim_element('Y',275000000,625000000,0.5)),NULL);
```

After this preparation, and loading of the data, spatial indices can be created. The default spatial index is r-tree, as an alternative quadtree indices are available. For 'normal' indices the default type is b-tree.

```
analyze table xfio_boundary compute statistics;

create index xfiox_boundary_0 on xfio_boundary (bbox)
indextype is mdsys.spatial_index
parameters ('sdo_fanout=64 sdo_rtr_pctfree=10
tablespace=indx initial=4m next=4m pctincrease=0');

create index xfiox_boundary_0a on xfio_boundary (abox)
indextype is mdsys.spatial_index
parameters ('sdo_fanout=64 sdo_rtr_pctfree=10
tablespace=indx initial=4m next=4m pctincrease=0');

create index xfiox_boundary_0b on xfio_boundary (shape)
indextype is mdsys.spatial_index
parameters ('sdo_fanout=64 sdo_rtr_pctfree=10
tablespace=indx initial=4m next=4m pctincrease=0');

create index xfiox_boundary_1b on xfio_boundary (object_id)
storage (initial 4m next 4m pctincrease 0)
```

```
tablespace indx nologging compute statistics;
        .
        .
        .
create index xfiox_boundary_10 on xfio_boundary (l_municip,
    l_section,l_sheet,r_municip,r_section,r_sheet)
storage (initial 4m next 4m pctincrease 0)
tablespace indx nologging compute statistics;

-- Analyze "_rt$" tables (to avoid bug in Spatial,
-- should be done automatically)

analyze table xfiox_boundary_0_rt$  compute statistics;
analyze table xfiox_boundary_0a_rt$ compute statistics;
analyze table xfiox_boundary_0b_rt$ compute statistics;

analyze table xfio_parcel compute statistics;

create index xfiox_parcel_0 on xfio_parcel (bbox)
indextype is mdsys.spatial_index
parameters ('sdo_fanout=64 sdo_rtr_pctfree=10
tablespace=indx initial=4m next=4m pctincrease=0');

create index xfiox_parcel_1b on xfio_parcel (object_id)
storage (initial 4m next 4m pctincrease 0)
tablespace indx nologging compute statistics;
        .
        .
        .
create index xfiox_text_2 on xfio_text (otext)
storage (initial 4m next 4m pctincrease 0)
tablespace indx nologging compute statistics;

analyze table xfiox_text_0_rt$ compute statistics;
```

AKR table definitions

For administrative data index-organized tables are a good solution for the primary storage structure. However these can only be used on primary keys. This means that index organization must be set on unique attribute(s). This makes `mo_subject` and `mo_object` good candidates for index organization. For `mo_subject` first the duplicate subjects have to be removed, then it is clustered on `subject_id`. The table `mo_object`, which contains parcels, is clustered on the attribute `xfio_objectnummer`. This clustering will result in a semi-spatial clustering.

```
/*-------------------------------------------------*/
/* NOTE: This script has been manually edited after   */
/*       being generated                           */
/*-------------------------------------------------*/
/* NOTE: do not edit by hand. This file is generated! */
/*-------------------------------------------------*/
set echo on;
set timing on;
```

```
drop table mo_object;
create table mo_object(
   x_akr_objectnummer varchar(17)   primary key,
   klant_id varchar(5),
   municip varchar(5),
   osection varchar(2),
   parcel varchar(5),
   pp_i_ltr varchar(1),
   pp_i_nr varchar(4),
   ontvangstdatum varchar(8),
   volgnummer number(11),
   was_wordt varchar(1),
   stuk_wijziging varchar(16),
   aanbieder varchar(10),
   verlijdensdatum varchar(8),
   soort_stuk varchar(3),
   stuk_vestiging varchar(16),
   ontdat_st_vest varchar(8),
   grootte number(11),
   indicatie_grootte_geschat varchar(1),
   bladnr number(11),
   bladvolgnr number(11),
   ruitletter varchar(1),
   ruitnummer number(11),
   x number(11),
   y number(11),
   beb_code varchar(1),
   soort_cult varchar(2),
   ind_meer_cult varchar(1),
   koopsom number(11),
   koopjaar number(11),
   ind_meer_kad varchar(1),
   omschr_deelp varchar(20),
   ind_verv_per varchar(1),
   belast_plicht varchar(10)
)
organization index
tablespace akr
pctthreshold 10
storage ( initial 4m next  4m pctincrease  0);

drop table mo_subject;
create table mo_subject(
   klant_id varchar(5),
   subject_id varchar(10),
   ontvangstdatum varchar(8),
   volgnummer number(11),
   was_wordt varchar(1),
   stuk_wijziging varchar(16),
   aanbieder varchar(10),
   verlijdensdatum varchar(8),
   soort_stuk varchar(3),
   indicatie_diacritisch varchar(1),
   nat_pers_code varchar(1),
   soort_niet_nat varchar(2),
```

```
   naam_niet_nat varchar(512),
   trefwoord1 varchar(5),
   trefwoord2 varchar(5),
   trefwoord3 varchar(5),
   zetel varchar(24),
   gesl_naam varchar(128),
   voornamen varchar(128),
   voorletters varchar(10),
   voorvoegsel varchar(10),
   ad_titel varchar(11),
   geb_datum number(11),
   geb_plaats varchar(24),
   ind_overleden varchar(1),
   overl_datum number(11),
   landc_wadres varchar(2),
   buitl_adr1 varchar(39),
   buitl_adr2 varchar(39),
   buitl_adr3 varchar(39),
   postcode varchar(6),
   woonplaats varchar(24),
   straatnaam varchar(24),
   aanduiding_huisnummer varchar(2),
   huisnummer varchar(5),
   huisletter varchar(1),
   huisnummertoevoeging varchar(4),
   locatiebeschrijving varchar(40),
   landcode_postadres varchar(2),
   post_buitl_adr1 varchar(39),
   post_buitl_adr2 varchar(39),
   post_buitl_adr3 varchar(39),
   post_postcode varchar(6),
   post_woonplaats varchar(24),
   post_straatnaam varchar(24),
   post_aanduiding_huisnummer varchar(2),
   post_huisnummer varchar(5),
   post_huisletter varchar(1),
   post_huisnummertoevoeging varchar(4),
   post_locatiebeschrijving varchar(40),
   post_postbusnr varchar(5)
)
storage
(
   initial 4m
   next  4m
   pctincrease  0
) tablespace akr;

drop index idx_mo_subject_tmp;
create index idx_mo_subject_tmp on mo_subject ( subject_id)
   storage (initial 4m next 4m pctincrease 0)
   tablespace indx nologging compute statistics;

delete from mo_subject m1
where m1.subject_id in
      (select m2.subject_id
       from mo_subject m2
```

```
       where m1.subject_id=m2.subject_id and m1.rowid<m2.rowid);

--
-- Make a temporary table which contains all
-- subjects with doubles. This table is index-organized.
--
drop table mo_subject_tmp;
create table mo_subject_tmp(
   klant_id varchar(5),
   subject_id varchar(10) primary key,
   ontvangstdatum varchar(8),
   volgnummer number(11),
   was_wordt varchar(1),
   stuk_wijziging varchar(16),
   aanbieder varchar(10),
   verlijdensdatum varchar(8),
   soort_stuk varchar(3),
   indicatie_diacritisch varchar(1),
   nat_pers_code varchar(1),
   soort_niet_nat varchar(2),
   naam_niet_nat varchar(512),
   trefwoord1 varchar(5),
   trefwoord2 varchar(5),
   trefwoord3 varchar(5),
   zetel varchar(24),
   gesl_naam varchar(128),
   voornamen varchar(128),
   voorletters varchar(10),
   voorvoegsel varchar(10),
   ad_titel varchar(11),
   geb_datum number(11),
   geb_plaats varchar(24),
   ind_overleden varchar(1),
   overl_datum number(11),
   landc_wadres varchar(2),
   buitl_adr1 varchar(39),
   buitl_adr2 varchar(39),
   buitl_adr3 varchar(39),
   postcode varchar(6),
   woonplaats varchar(24),
   straatnaam varchar(24),
   aanduiding_huisnummer varchar(2),
   huisnummer varchar(5),
   huisletter varchar(1),
   huisnummertoevoeging varchar(4),
   locatiebeschrijving varchar(40),
   landcode_postadres varchar(2),
   post_buitl_adr1 varchar(39),
   post_buitl_adr2 varchar(39),
   post_buitl_adr3 varchar(39),
   post_postcode varchar(6),
   post_woonplaats varchar(24),
   post_straatnaam varchar(24),
   post_aanduiding_huisnummer varchar(2),
   post_huisnummer varchar(5),
   post_huisletter varchar(1),
```

```
   post_huisnummertoevoeging varchar(4),
   post_locatiebeschrijving varchar(40),
   post_postbusnr varchar(5)
)
organization index
tablespace indx
storage ( initial 4m next  4m pctincrease  0)
pctthreshold 10
overflow tablespace akr
storage ( initial 4m next  4m pctincrease  0);

--
-- Insert all subjects into this table.
--
insert into mo_subject_tmp
   select * from mo_subject
   order by subject_id;

drop index idx_mo_subject_tmp;
drop table mo_subject;

alter table mo_subject_tmp rename to mo_subject;
```

AKR table analyze

```
AKR:
analyze table mo_object compute statistics;
analyze table mo_objectadres compute statistics;
analyze table mo_rente compute statistics;
analyze table mo_onzelfstandig_deel compute statistics;
analyze table mo_obj_belemmering compute statistics;
analyze table mo_overgegaan_in compute statistics;
analyze table mo_ontstaan_uit compute statistics;
analyze table mo_recht compute statistics;
analyze table mo_rechtbelemmering compute statistics;
analyze table mo_objectmutatie compute statistics;
analyze table mo_reg_9_tekst compute statistics;
analyze table mo_reg_9 compute statistics;
analyze table mo_subject compute statistics;
analyze table mo_huwelijksrelatie compute statistics;
analyze table mo_groepsrelatie compute statistics;
analyze table mo_subjectrelatie compute statistics;
analyze table mo_subjectmutaties compute statistics;

VIEWS:
analyze table object_parcel compute statistics;
```

Views

Views are an important database concept, and the view is widely used in this project. Below some code fragments of the creation of views are given:

```
SQL> create view lki_boundary(
```

```
  2          ogroup, object_id, old_slc, classif, interp_cd, geo_color,
  3          geo_polyline, height, node_cd, status_cd,
  4          fl_line_id, fr_line_id, ll_line_id, lr_line_id, l_obj_id, r_obj_id,
  5          accu_cd, geo_bbox, abox, linelen, object_dt,
  6          tmin, tmax, sel_cd, source, quality, vis_cd,
  7          l_municip, l_section, l_sheet, l_parcel, l_pp_i_ltr, l_pp_i_nr,
  8          r_municip, r_section, r_sheet, r_parcel, r_pp_i_ltr, r_pp_i_nr
  9  ) as select
 10          ogroup, object_id, slc, classif, interp_cd, (3+mod((tmin+tmax),35)),
 11          shape, height, node_cd, status_cd,
 12          fl_line_id, fr_line_id, ll_line_id, lr_line_id, l_obj_id, r_obj_id,
 13          accu_cd, bbox, abox, linelen, object_dt,
 14          tmin, tmax, sel_cd, source, quality, vis_cd,
 15          l_municip, l_section, l_sheet, l_parcel, l_pp_i_ltr, l_pp_i_nr,
 16          r_municip, r_section, r_sheet, r_parcel, r_pp_i_ltr, r_pp_i_nr
 17  from xfio_boundary
 18  where ogroup=6 and (tmin <= 313372800) and ((313372800 < tmax) or (tmax=0));

SQL> create view akr_objectSOM as
  2      select x.ogroup,x.object_id,x.slc,x.classif,x.location,x.d_location,
  3      x.rotangle,x.geo_bbox,x.tmin,x.tmax,o.municip,o.osection,o.parcel,
  4      o.pp_i_ltr,o.pp_i_nr,x.l_num,x.line_id1,x.line_id2,
  5      o.klant_id, o.ontvangstdatum, o.volgnummer , o.was_wordt, o.stuk_wijziging,
  6      o.aanbieder , o.verlijdensdatum, o.soort_stuk, o.stuk_vestiging,
  7      o.ontdat_st_vest, o.grootte , o.indicatie_grootte_geschat, o.bladnr ,
  8      o.bladvolgnr, o.ruitletter, o.ruitnummer,
  9      o.x , o.y , o.beb_code, o.soort_cult,
 10      o.ind_meer_cult, o.koopsom , o.koopjaar , o.ind_meer_kad, o.omschr_deelp,
 11      o.ind_verv_per, o.belast_plicht
 12      from   mo_object o, lki_parcel x
 13      where o.municip = x.municip and
 14            o.osection = x.osection and
 15            o.parcel = x.parcel /*and
 16            o.pp_i_ltr = x.pp_i_ltr and
 17            o.pp_i_nr = x.pp_i_nr*/;

SQL> create view akr_recht_subject as
  2      select x.ogroup,x.object_id,x.slc,x.classif,x.location,x.d_location,
  3      x.rotangle,x.geo_bbox,x.tmin,x.tmax,o.municip,o.osection,o.parcel,
  4      o.pp_i_ltr,o.pp_i_nr,x.l_num,x.line_id1,x.line_id2, op.x_akr_objectnummer,
  5      op.g_akr_objectnummer,
  6      /* o.klant_id,  o.ontvangstdatum, o.volgnummer ,  o.was_wordt, o.stuk_wijziging,
  7      o.aanbieder, o.verlijdensdatum,  o.soort_stuk, */ o.gerechtigde, o.soort_recht,
  8      o.aandeel, o.stuk_vest_recht, o.ontv_datum, o.ind_einde,
  9      o.aandeel_medeger_groep , o.ind_split, s.*
 10      from   mo_recht o, lki_parcel x, mo_subject s , object_parcel op
 11      where o.x_akr_objectnummer = op.x_akr_objectnummer and
 12            op.g_akr_objectnummer = x.x_akr_objectnummer and
 13            o.gerechtigde = s.subject_id ;
```

`Object_parcel` table

The `object_parcel` table plays a key role in making the connection between LKI and AKR. The data in this table is extracted from many different tables and then inserted into

the `object_parcel` table. The table is primarily index-organized on `x_akr_objectnummer`.

```
SQL> create table object_parcel
  2  (
  3     x_akr_objectnummer varchar(17),
  4     g_akr_objectnummer varchar(17),
  5     constraint pk primary key(x_akr_objectnummer,g_akr_objectnummer)
  6  )
  7  organization index
  8  tablespace indx
  9  storage ( initial 4m next 4m pctincrease 0)
 10  pctthreshold 10
 11  overflow tablespace akr
 12  storage ( initial 4m next 4m pctincrease 0);

SQL> insert into object_parcel
  2   select  x_akr_objectnummer, g_akr_objectnummer
  3   from  akr_object_g
  4   order by x_akr_objectnummer,g_akr_objectnummer;

SQL> insert into object_parcel
  2   select  x_akr_objectnummer, g_akr_objectnummer
  3   from  akr_object_a
  4   order by x_akr_objectnummer,g_akr_objectnummer;

SQL> insert into object_parcel
  2   select /*+ ORDERED */  x_akr_objectnummer, g_akr_objectnummer
  3   from  akr_object_d
  4   order by x_akr_objectnummer,g_akr_objectnummer;
```

LKI load script

To explain the LKI load process the first part of the `lki_load.sh` script is listed. This script controls the loading, some other (Perl and SQL) scripts are called to perform various tasks.

```
#!/bin/ksh
# lki_load.sh  Oracle version  12-03-2001  TT
#
################################################################################
# Section 0.0) Explanation
################################################################################
#
# Script to load LKI data from Ingres dump files into Oracle 8i database:
#
#     - create tables (by lki_model.sql script)
#     - reformat Ingres dump files to Sqlldr format (by ing2ora perl script)
#     - load data into database (section 3 of this script)
#     - create indexes, analyze tables/indexes, alter properties,
#       grant privileges (by lki_postproc.sql script)
#
# Usage: lki_load.sh <load_option> <office>
#
#            load_option: defines mode of operation
```

```
#            office:      indicates office(s)/region(s) to load
#                         (one or more subdirectories in the filesystem)
#
# "office" may contain wildcards to indicate more offices/regions, but because
# this script expects exactly 2 arguments wildcards must be protected from
# expansion by the shell (by escaping or quoting the office argument).
#
# Currently 4 "modes of operation" are recognized:
#
#     all:      do everything in one run: create tables, load data, create
#               indexes, analyze tables/indexes, alter properties and
#               grant privileges for office(s) specified
#     create:   create tables and load data into database for office(s)
#     append:   add data to database for office(s)
#     appindex: add data to database for office(s), create indexes,
#               analyze tables/indexes, alter properties and grant privileges
#
# This script consists of 5 sections:
#
#     0) explanation, user configurable options, functions
#     1) program initialization (argument check, availability scripts/data)
#     2) table creation section
#     3) data loading section
#     4) postprocessing section
#
# The various sections are executed based on the load_option specified:
#     all:      1, 2, 3, 4
#     create:   1, 2, 3
#     append:   1, 3
#     appindex: 1, 3, 4
#
.....
```

The first part of the `lki_ing2ora.pl` script (called by `lki_load.sh` for each LKI table) explains the function of this script in the load process:

```
#!/usr/local/bin/perl
# lki_ing2ora.pl  Oracle version  14-03-2001
# History:
#
#  1-Dec-2000: quak@geo.tudelft.nl
#     Creation based on awk script by Sander Alten & Peter
#     van Oosterom & Theo Tijssen.
# 28-Dec-2000: TT
#     option action added
#     additional char types added
#     explicit size added to datatypes in control file
#
# This script helps loading files in Ingres export format into
# an Oracle database. Syntax:
#
#  lki_ing2ora.pl <action> <tablename> [-<datatype> <attributename>]*
#
# Recognized actions:
#    all  control  convert
#
```

```
# Recognized datatypes:
#    -integer -real -char8 -char32 -char128 -point -polyline -box
#
# The program performs two functions:
# 1. A file tablename.ctl is generated (action=all | control).
#    This file controls the loading into the Oracle database.
# 2. The standard input, which is supposed to be the data in Ingres
#    export format, is converted to Oracle sqlldr format and printed
#    to standard output (action=all | convert).
#
.....
```

SQLLDR control file

As part of the load process (both LKI and AKR) `sqlldr` control files are generated automatically by Perl scripts and then used to control data loading. As an example the `xfio_boundary` control file is included.

```
load data
append into table xfio_boundary
fields terminated by ','
trailing nullcols
(
 ogroup integer external(12),
 object_id integer external(12),
 slc integer external(12),
 classif integer external(12),
 interp_cd integer external(12),
 shape column object
 (
   sdo_gtype integer external(4),
   sdo_elem_info varray terminated by '|'
    (till_element_info integer external(4)),
     sdo_ordinates varray terminated by '|'
    (after_element_info integer external(10))
 ),
 height integer external(12),
 node_cd integer external(12),
 status_cd integer external(12),
 fl_line_id integer external(12),
 fr_line_id integer external(12),
 ll_line_id integer external(12),
 lr_line_id integer external(12),
 l_obj_id integer external(12),
 r_obj_id integer external(12),
 accu_cd integer external(12),
 bbox column object
 (
   sdo_gtype integer external(4),
   sdo_elem_info varray terminated by '|'
    (till_element_info integer external(4)),
     sdo_ordinates varray terminated by '|'
    (after_element_info integer external(10))
 ),
```

```
 linelen integer external(12),
 object_dt integer external(12),
 tmin integer external(12),
 tmax integer external(12),
 sel_cd char(8) terminated by ',' enclosed by '<' and '>',
 source char(8) terminated by ',' enclosed by '<' and '>',
 quality char(8) terminated by ',' enclosed by '<' and '>',
 vis_cd char(8) terminated by ',' enclosed by '<' and '>',
 l_municip char(8) terminated by ',' enclosed by '<' and '>',
 l_section char(8) terminated by ',' enclosed by '<' and '>',
 l_sheet char(8) terminated by ',' enclosed by '<' and '>',
 l_parcel char(8) terminated by ',' enclosed by '<' and '>',
 l_pp_i_ltr char(8) terminated by ',' enclosed by '<' and '>',
 l_pp_i_nr char(8) terminated by ',' enclosed by '<' and '>',
 r_municip char(8) terminated by ',' enclosed by '<' and '>',
 r_section char(8) terminated by ',' enclosed by '<' and '>',
 r_sheet char(8) terminated by ',' enclosed by '<' and '>',
 r_parcel char(8) terminated by ',' enclosed by '<' and '>',
 r_pp_i_ltr char(8) terminated by ',' enclosed by '<' and '>',
 r_pp_i_nr char(8) terminated by ',' enclosed by '<' and '>'
)
```

AKR load script

The following script is used to control the loading of AKR data. The script uses other scripts and software, some of which are described in more detail below.

```
#! /bin/sh
#
# Master script to create a akr database from scratch.
# (ingres version)
#
# no date known: Original version by Bart Maessen?
# 20-Mar-2000  : Ported to postgres by Wilko Quak (quak@geo.tudelft.nl)
# 30-Nov-2000  : Ported to Oracle by Wilko Quak (quak@geo.tudelft.nl)
#
set -v

#
# Include Parameters.
#
. params.sh

CWD=`pwd`
DEF=$CWD/used_definition.dat

#
# make sure 'readakr' program is up to date.
#
(cd src ; make)

#
# make sure the tmp directory exists.
#
```

```
if [ ! -d $TMPDIR ]; then
   mkdir $TMPDIR
fi

#
# Collect and load data and create control files.
#
(cd $TMPDIR ;$CWD/src/readakr -oracle -makescripts -makedata $DEF /dev/null )
if [ ! -s $TMPDIR/model.sql ]; then
    echo "FATAL: missing '$TMPDIR/model.sql'"
    exit
fi

#
# Build database-structure
#
# Normally this file is generated by a script. Because we modified
# the generated file, we use the edited copy. (quak@geo.tudelft.nl)
#
#sqlplus  $DATABASE @$TMPDIR/model
sqlplus  $DATABASE @model

#
# For all departments that need to be loaded
#
for i in arnhem rotterdam zoetermeer
do
    DATADIR=/r01/home/kadtest/kad/data/akr_$i
    mkdir $TMPDIR/$i

    #
    # Read all data and use readakr to convert it to oracle format.
    #
    gunzip -c $DATADIR/obj/*.gz $DATADIR/sub/*.gz |  \
        (cd $TMPDIR/$i ;$CWD/src/readakr -oracle -makescripts -makedata $DEF - )

    #
    # Load the data into oracle.
    #
    cd $TMPDIR/$i; sh $TMPDIR/model_load.sh $DATABASE ROWS=10000 DIRECT=TRUE
    cat $TMPDIR/$i/*.log
done

cd $CWD

#
# Remove double subjects.
#
sqlplus  $DATABASE @delete_double_subject

#
# Make primary keys. We should look at clustering.
#
sqlplus  $DATABASE @model_modify

#
```

```
# Build database-index/views
#
sqlplus $DATABASE @$TMPDIR/model_index

#
# Compute statistics on tables etc.
#
sqlplus $DATABASE @akr_postproc

#
# Ready.
#
```

`Delete_double_subject.sql` script

The following script deletes the double subjects from the `mo_subject` table. Interesting detail is that the zip code ('postcode') is used as the beginning of the primary (cluster) key, so that the data is clustered spatially. Because the zip code attribute is sometimes NULL (which is not allowed for a primary key) first all zip codes with value NULL are set to valu '0000':

```
--
-- delete_double_subject.sql
--
set timing on;
set echo on;

--
-- Create temporary index to speed up the deletion of
-- double subjects.
--
drop index idx_mo_subject_tmp;
create index idx_mo_subject_tmp on mo_subject ( subject_id)
   storage (initial 4m next 4m pctincrease 0)
   nologging compute statistics;

delete from mo_subject m1
where m1.subject_id in
      (select m2.subject_id
       from mo_subject m2
       where m1.subject_id=m2.subject_id and m1.rowid<m2.rowid);

--
-- Delete nulls from postcode attribute, so we can use it for
-- a primary index.
--
update mo_subject
set postcode = '0000'
where postcode is null;

drop table mo_subject_tmp;
create table mo_subject_tmp(
   klant_id varchar(5),
   subject_id varchar(10),
```

```
   ontvangstdatum varchar(8),
   volgnummer number(11),
   was_wordt varchar(1),
   stuk_wijziging varchar(16),
   aanbieder varchar(10),
   verlijdensdatum varchar(8),
   soort_stuk varchar(3),
   indicatie_diacritisch varchar(1),
   nat_pers_code varchar(1),
   soort_niet_nat varchar(2),
   naam_niet_nat varchar(512),
   trefwoord1 varchar(5),
   trefwoord2 varchar(5),
   trefwoord3 varchar(5),
   zetel varchar(24),
   gesl_naam varchar(128),
   voornamen varchar(128),
   voorletters varchar(10),
   voorvoegsel varchar(10),
   ad_titel varchar(11),
   geb_datum number(11),
   geb_plaats varchar(24),
   ind_overleden varchar(1),
   overl_datum number(11),
   landc_wadres varchar(2),
   buitl_adr1 varchar(39),
   buitl_adr2 varchar(39),
   buitl_adr3 varchar(39),
   postcode varchar(6),
   woonplaats varchar(24),
   straatnaam varchar(24),
   aanduiding_huisnummer varchar(2),
   huisnummer varchar(5),
   huisletter varchar(1),
   huisnummertoevoeging varchar(4),
   locatiebeschrijving varchar(40),
   landcode_postadres varchar(2),
   post_buitl_adr1 varchar(39),
   post_buitl_adr2 varchar(39),
   post_buitl_adr3 varchar(39),
   post_postcode varchar(6),
   post_woonplaats varchar(24),
   post_straatnaam varchar(24),
   post_aanduiding_huisnummer varchar(2),
   post_huisnummer varchar(5),
   post_huisletter varchar(1),
   post_huisnummertoevoeging varchar(4),
   post_locatiebeschrijving varchar(40),
   post_postbusnr varchar(5),
   constraint mo_subject_pk primary key(postcode,subject_id)
)
organization index compress 1 -- first 2 col's are prefix-part and compressed
tablespace indx nologging
storage ( initial 4m next  4m pctincrease  0)
pctthreshold 10
overflow tablespace akr nologging
```

```
storage ( initial 4m next  4m pctincrease  0);

insert into mo_subject_tmp
   select * from mo_subject
   order by subject_id;

drop index idx_mo_subject_tmp;
drop table mo_subject;

alter table mo_subject_tmp rename to mo_subject;

exit;
```

Appendix C: Ingres cadastral data loading

Table definitions

```
create  table xfio_parcel(
ogroup      integer4    not null not default,   /* Group Id (KEY.1) ->CLASS.1 */
object_id   integer4    not null not default,   /* text/polygon object Id (KEY.2) */
slc         integer4    not null not default,   /* slc code */
classif     integer4    not null with default,  /* Object class code ->CLASS.2 */
location    ipoint      not null with default,  /* x&y-coordinate (on map) */
z           integer4    not null with default,  /* z-coordinate */
d_location  ipoint      not null with default,  /* Delta x&y of pointnum */
rotangle    integer4    not null with default,  /* Rotation angle */
accu_cd     integer4    not null with default,  /* Accuracy code */
oarea       float8      not null with default,  /* terrain area of related \p\gn */
bbox        ibox        not null with default,  /* bounding box of polygon */
object_dt   integer4    not null with default,  /* Date (of measurement; user time) */
tmin        integer4    not null with default,  /* Time created/last updated */
tmax        integer4    not null with default,  /* Time deleted/last updated */
sel_cd      char(3)     not null with default,  /* Belongs to map: cadast, GKBN... */
source      char(5)     not null with default,  /* Source of data */
quality     char(2)     not null with default,  /* Data quality: method, accuracy */
vis_cd      char(1)     not null with default,  /* Visibility code */
akr_area    float8      not null with default,  /* Official AKR area of \p\gn */
municip     char(5)     not null with default,  /* Municipality code (ALT-KEY.1) */
osection    char(2)     not null with default,  /* Section code (ALT-KEY.2) */
sheet       char(4)     not null with default,  /* Sheet code (ALT-KEY.3) */
parcel      char(5)     not null with default,  /* Parcel code (ALT-KEY.4) */
pp_i_ltr    char(1)     not null with default,  /* Part-parcel LNNNN (ALT-KEY.5) */
pp_i_nr     char(4)     not null with default,
l_num       integer4    not null with default,  /* number of line references) */
line_id1    integer4    not null with default,  /* 1st line ref, outer boundary */
line_id2    integer4    not null with default,  /* 2nd line ref, 1st hole */
x_akr_objectnummer char(17) not null with default) /* EXTRA: voor koppeling met IDB */
with    duplicates;\p\g

create  table xfio_boundary(
ogroup      integer4    not null not default,   /* Group Id (KEY.1) ->CLASS.1 */
object_id   integer4    not null not default,   /* Line object Id (KEY.2) */
slc         integer4    not null not default,   /* slc code */
classif     integer4    not null with default,  /* Object class code ->CLASS.2 */
interp_cd   integer1    not null with default,  /* Line interpolation */
shape       iline(50)   not null with default,  /* Line coordinates (2D) (compr.) */
height      integer4    not null with default,
node_cd     integer1    not null with default,  /* Node code */
status_cd   integer4    not null with default,
fl_line_id  integer4    not null with default,  /* Line Id left side 1st p.->LINE */
fr_line_id  integer4    not null with default,  /* Line Id right side 1st p.->LINE */
ll_line_id  integer4    not null with default,  /* Line Id L side last pnt->LINE */
lr_line_id  integer4    not null with default,  /* Line Id R side last pnt->LINE */
l_obj_id    integer4    not null with default,
r_obj_id    integer4    not null with default,
accu_cd     integer4    not null with default,  /* Accuracy code */
bbox        ibox        not null with default,  /* Bounding box */
abox        ibox        not null with default,  /* Area box */
```

```
linelen     integer4    not null with default,  /* Length of line */
object_dt   integer4    not null with default,  /* Date (of measurement; user time) */
tmin        integer4    not null with default,  /* Time created/last updated */
tmax        integer4    not null with default,  /* Time deleted/last updated */
sel_cd      char(3)     not null with default,  /* Belongs to map: cadast, GBKN... */
source      char(5)     not null with default,  /* Source of data */
quality     char(2)     not null with default,  /* Data quality: method, accuracy */
vis_cd      char(1)     not null with default,  /* Visibility code */
l_municip   char(5)     not null with default,  /* Left Mun. code ->ALT-TEXPGN.1 */
l_section   char(2)     not null with default,  /* Left Section code ->ALT-TEXPGN.2 */
l_sheet     char(4)     not null with default,  /* Left Sheet code ->ALT-TEXPGN.3 */
l_parcel    char(5)     not null with default,  /* Left Parcel code ->ALT-TEXPGN.4 */
l_pp_i_ltr  char(1)     not null with default,
l_pp_i_nr   char(4)     not null with default,
r_municip   char(5)     not null with default,  /* Right Mun. code ->ALT-TEXPGN.1 */
r_section   char(2)     not null with default,  /* Right Section code ->ALT-TEXPGN.2 */
r_sheet     char(4)     not null with default,  /* Right Sheet code ->ALT-TEXPGN.3 */
r_parcel    char(5)     not null with default,  /* Right Parcel code ->ALT-TEXPGN.4 */
r_pp_i_ltr  char(1)     not null with default,
r_pp_i_nr   char(4)     not null with default)
/* with page_size=2048 */
with location=(ii_database,data2);\p\g

/*-----------------------------------------------------*/
/* NOTE: do not edit by hand. This file is generated! */
/*-----------------------------------------------------*/
drop table mo_object \p\g
create table mo_object(
oid object_key with system_maintained,
x_akr_objectnummer char(17) not null with default '',
klant_id char(5) not null with default '',
municip char(5) not null with default '',
osection char(2) not null with default '',
parcel char(5) not null with default '',
pp_i_ltr char(1) not null with default '',
pp_i_nr char(4) not null with default '',
ontvangstdatum char(8) not null with default '',
volgnummer integer4 not null with default 0,
was_wordt char(1) not null with default '',
stuk_wijziging char(16) not null with default '',
aanbieder char(10) not null with default '',
verlijdensdatum char(8) not null with default '',
soort_stuk char(3) not null with default '',
stuk_vestiging char(16) not null with default '',
ontdat_st_vest char(8) not null with default '',
grootte integer4 not null with default 0,
indicatie_grootte_geschat char(1) not null with default '',
bladnr integer4 not null with default 0,
bladvolgnr integer4 not null with default 0,
ruitletter char(1) not null with default '',
ruitnummer integer4 not null with default 0,
x integer4 not null with default 0,
y integer4 not null with default 0,
beb_code char(1) not null with default '',
soort_cult char(2) not null with default '',
ind_meer_cult char(1) not null with default '',
```

```
koopsom integer4 not null with default 0,
koopjaar integer4 not null with default 0,
ind_meer_kad char(1) not null with default '',
omschr_deelp char(20) not null with default '',
ind_verv_per char(1) not null with default '',
belast_plicht char(10) not null with default '') with duplicates \p\g

create table mo_subject(
oid object_key with system_maintained,
klant_id char(5) not null with default '',
subject_id char(10) not null with default '',
ontvangstdatum char(8) not null with default '',
volgnummer integer4 not null with default 0,
was_wordt char(1) not null with default '',
stuk_wijziging char(16) not null with default '',
aanbieder char(10) not null with default '',
verlijdensdatum char(8) not null with default '',
soort_stuk char(3) not null with default '',
indicatie_diacritisch char(1) not null with default '',
nat_pers_code char(1) not null with default '',
soort_niet_nat char(2) not null with default '',
naam_niet_nat varchar(512) not null with default '',
trefwoord1 char(5) not null with default '',
trefwoord2 char(5) not null with default '',
trefwoord3 char(5) not null with default '',
zetel char(24) not null with default '',
gesl_naam varchar(128) not null with default '',
voornamen varchar(128) not null with default '',
voorletters char(10) not null with default '',
voorvoegsel char(10) not null with default '',
ad_titel char(11) not null with default '',
geb_datum integer4 not null with default 0,
geb_plaats char(24) not null with default '',
ind_overleden char(1) not null with default '',
overl_datum integer4 not null with default 0,
landc_wadres char(2) not null with default '',
buitl_adr1 varchar(39) not null with default '',
buitl_adr2 varchar(39) not null with default '',
buitl_adr3 varchar(39) not null with default '',
postcode char(6) not null with default '',
woonplaats char(24) not null with default '',
straatnaam char(24) not null with default '',
aanduiding_huisnummer char(2) not null with default '',
huisnummer char(5) not null with default '',
huisletter char(1) not null with default '',
huisnummertoevoeging char(4) not null with default '',
locatiebeschrijving varchar(40) not null with default '',
landcode_postadres char(2) not null with default '',
post_buitl_adr1 varchar(39) not null with default '',
post_buitl_adr2 varchar(39) not null with default '',
post_buitl_adr3 varchar(39) not null with default '',
post_postcode char(6) not null with default '',
post_woonplaats char(24) not null with default '',
post_straatnaam char(24) not null with default '',
post_aanduiding_huisnummer char(2) not null with default '',
post_huisnummer char(5) not null with default '',
```

```
post_huisletter char(1) not null with default '',
post_huisnummertoevoeging char(4) not null with default '',
post_locatiebeschrijving varchar(40) not null with default '',
post_postbusnr char(5) not null with default '') with duplicates \p\g

create table mo_recht(
oid object_key with system_maintained,
x_akr_objectnummer char(17) not null with default '',
klant_id char(5) not null with default '',
municip char(5) not null with default '',
osection char(2) not null with default '',
parcel char(5) not null with default '',
pp_i_ltr char(1) not null with default '',
pp_i_nr char(4) not null with default '',
ontvangstdatum char(8) not null with default '',
volgnummer integer4 not null with default 0,
was_wordt char(1) not null with default '',
stuk_wijziging char(16) not null with default '',
aanbieder char(10) not null with default '',
verlijdensdatum char(8) not null with default '',
soort_stuk char(3) not null with default '',
gerechtigde char(10) not null with default '',
soort_recht char(6) not null with default '',
aandeel char(16) not null with default '',
stuk_vest_recht char(16) not null with default '',
ontv_datum char(8) not null with default '',
ind_einde char(1) not null with default '',
aandeel_medeger_groep integer4 not null with default 0,
ind_split char(1) not null with default '') with duplicates \p\g
```

Note that the table `xfio_boundary` is stored in two locations: `ii_database` (the default location) and `data2`. The reason for this is that in Ingres there is a limitation of 2 Gb per table per location. In order to be able to do fair comparison with the Oracle test, both locations are on the same physical RAID5 disk set. The same was true for the table `tmp_mo_subject`, used during loading the `mo_subject` table. The table `mo_subject` is less than 2 Gb (773 Mb). However data is first loaded in the `tmp_mo_subject` table. This table has a heap storage structure without data compression and there are many varchar attributes. Therefore, more locations (`ii_database`, `data2`, `data3`, `data4`) are required during loading in the `tmp_mo_subject` table. All locations are again on the same physical RAID5 disk set.

Primary storage structure

```
modify  xfio_parcel to btree on slc, ogroup
with    nonleaffill = 80,
        leaffill = 70,
        fillfactor = 100,
        compression = (nokey, data);\p\g

modify  xfio_boundary to btree on slc, ogroup
with    nonleaffill = 80,
        leaffill = 70,
        fillfactor = 100,
```

```
        compression = (nokey, data);\p\g

modify  xfio_text to btree on slc, ogroup
with    nonleaffill = 80,
        leaffill = 70,
        fillfactor = 100,
        compression = (nokey, data);\p\g

modify mo_object to btree on x_akr_objectnummer
with compression = (nokey, data) \p\g

modify mo_subject to btree on subject_id,oid
with compression = (nokey, data) \p\g

modify mo_recht to btree on x_akr_objectnummer
with compression = (nokey, data) \p\g
```

Secondary indices

```
create          index xfiox_parcel_0
        on      xfio_parcel(bbox)
        with    structure=rtree,
                range=((-25000000,275000000),(325000000,625000000));\p\g

create          index xfiox_parcel_1b
        on      xfio_parcel (object_id)
        with    structure = isam,
                fillfactor = 80;\p\g

create          index xfiox_parcel_3
        on      xfio_parcel (x_akr_objectnummer)
        with    structure = isam,
                fillfactor = 80;\p\g

create          index xfiox_boundary_0
        on      xfio_boundary(bbox)
        with    structure=rtree,
                range=((-25000000,275000000),(325000000,625000000));\p\g

create          index xfiox_boundary_0b
        on      xfio_boundary(shape)
        with    structure=rtree,
                range=((-25000000,275000000),(325000000,625000000));\p\g

create          index xfiox_boundary_1b
        on      xfio_boundary (object_id)
        with    structure = isam,
                fillfactor = 80;\p\g

create          index xfiox_text_0
        on      xfio_text(location)
        with    structure=rtree,
                range=((-25000000,275000000),(325000000,625000000));\p\g
```

```
create          index xfiox_text_2
        on      xfio_text (otext)
        with    compression,
                structure = btree;\p\g

/* NOTE: do not edit by hand. This file is generated! */
/* Remark when no index structure is specified the default is isam */

create index idx_mo_object_1 on mo_object (
municip,osection,parcel,pp_i_ltr,pp_i_nr) \p\g

create index idx_mo_subject_1 on mo_subject (
subject_id) \p\g

create index idx_mo_subject_2 on mo_subject (
gesl_naam,woonplaats) with compression \p\g

create index idx_mo_subject_3 on mo_subject (
postcode,woonplaats) with compression \p\g

create index idx_mo_recht_1 on mo_recht (
municip,osection,parcel,pp_i_ltr,pp_i_nr) \p\g

create index idx_mo_recht_2 on mo_recht (
gerechtigde) \p\g
```

Views

```
create view  lki_boundary(
  ogroup, object_id, old_slc, classif,
  interp_cd, geo_color, geo_polyline, height, node_cd, status_cd,
  fl_line_id, fr_line_id, ll_line_id, lr_line_id, l_obj_id, r_obj_id,
  accu_cd, geo_bbox, abox, linelen, object_dt, tmin, tmax, sel_cd,
  source, quality, vis_cd, l_municip, l_section, l_sheet, l_parcel,
  l_pp_i_ltr, l_pp_i_nr, r_municip, r_section, r_sheet, r_parcel,
  r_pp_i_ltr, r_pp_i_nr ) as
select ogroup, object_id, slc, classif,
  interp_cd, (3+mod((tmin+tmax),35)), char(char(shape),1150), height,
  node_cd, status_cd, fl_line_id, fr_line_id, ll_line_id, lr_line_id,
  l_obj_id, r_obj_id, accu_cd, bbox, abox, linelen, object_dt,
  lkiint2date(tmin), lkiint2date(tmax), sel_cd, source, quality, vis_cd,
  l_municip, l_section, l_sheet, l_parcel, l_pp_i_ltr, l_pp_i_nr,
  r_municip, r_section, r_sheet, r_parcel, r_pp_i_ltr, r_pp_i_nr
from xfio_boundary
where ogroup=6 and (tmin <= lkidate2int('01-10-1999 00:00:00')) and
  ((lkidate2int('01-10-1999 00:00:00') < tmax) or (tmax=0));\p\g

create view akr_objectSOM as
    select o.oid,x.ogroup,x.object_id,x.slc,x.classif,x.location,x.d_location,
    x.rotangle,x.geo_bbox,x.tmin,x.tmax,o.municip,o.osection,o.parcel,
    o.pp_i_ltr,o.pp_i_nr,x.l_num,x.line_id1,x.line_id2,
    o.klant_id, o.ontvangstdatum, o.volgnummer , o.was_wordt, o.stuk_wijziging,
    o.aanbieder , o.verlijdensdatum, o.soort_stuk, o.stuk_vestiging,
    o.ontdat_st_vest, o.grootte , o.indicatie_grootte_geschat, o.bladnr ,
```

```
    o.bladvolgnr, o.ruitletter, o.ruitnummer,
    o.x , o.y , o.beb_code, o.soort_cult,
    o.ind_meer_cult, o.koopsom , o.koopjaar , o.ind_meer_kad, o.omschr_deelp,
    o.ind_verv_per, o.belast_plicht
    from   mo_object o, lki_parcel x
    where  o.municip = x.municip and
           o.osection = x.osection and
           o.parcel = x.parcel /*and
           o.pp_i_ltr = x.pp_i_ltr and
           o.pp_i_nr = x.pp_i_nr*/\p\g

create view akr_recht_subject as
    select /* o.oid, */  x.ogroup,x.object_id,x.slc,x.classif,x.location,x.d_location,
    x.rotangle,x.geo_bbox,x.tmin,x.tmax,o.municip,o.osection,o.parcel,
    o.pp_i_ltr,o.pp_i_nr,x.l_num,x.line_id1,x.line_id2, op.x_akr_objectnummer,
    op.g_akr_objectnummer,
    /* o.klant_id,  o.ontvangstdatum, o.volgnummer ,  o.was_wordt, o.stuk_wijziging,
    o.aanbieder, o.verlijdensdatum,  o.soort_stuk, */ o.gerechtigde, o.soort_recht,
    o.aandeel, o.stuk_vest_recht, o.ontv_datum, o.ind_einde,
    o.aandeel_medeger_groep , o.ind_split, s.*
    from   mo_recht o, lki_parcel x, mo_subject s , object_parcel op
    where o.x_akr_objectnummer = op.x_akr_objectnummer and
          op.g_akr_objectnummer = x.x_akr_objectnummer and
          o.gerechtigde = s.subject_id \p \g
```

Appendix D: Oracle cadastral data querying

Query geometries

Nr	Geometry type	Short description	Area m^2	Length m
1	rectangle	Scheveningen	12500	450
2	rectangle	Rotterdam Spangen	50000	900
3	rectangle	Nijmegen Keizer Karelplein	200000	1800
4	rectangle	Binnenbedijkte Maas	1250000	4500
5	rectangle	Hazerswoude	31250000	22500
6	rectangle	Achterhoek -west	125000000	45000
7	rectangle	in parcel EBG02R00364G0000	30000	800
8	rectangle	Mil terrain Soesterberg	15000	500
9	polygon	rectangle	281327	2141
10	polygon	polygon with 8 points	239130	1865
11	polygon	N shape polygon	88993	1709
12	polygon (counter clockwize)	C shape polygon	38351	1253
13	polygon	Linear shape polygon	41228	1149
14	polygon	Irregular polygon with 50+ points	101931	2137
15	polygon (with one hole)	donut	680715	5018
16	polygon (with two holes)	glasses	1268459	8329

Table 13: Cadastral area queries characteristics

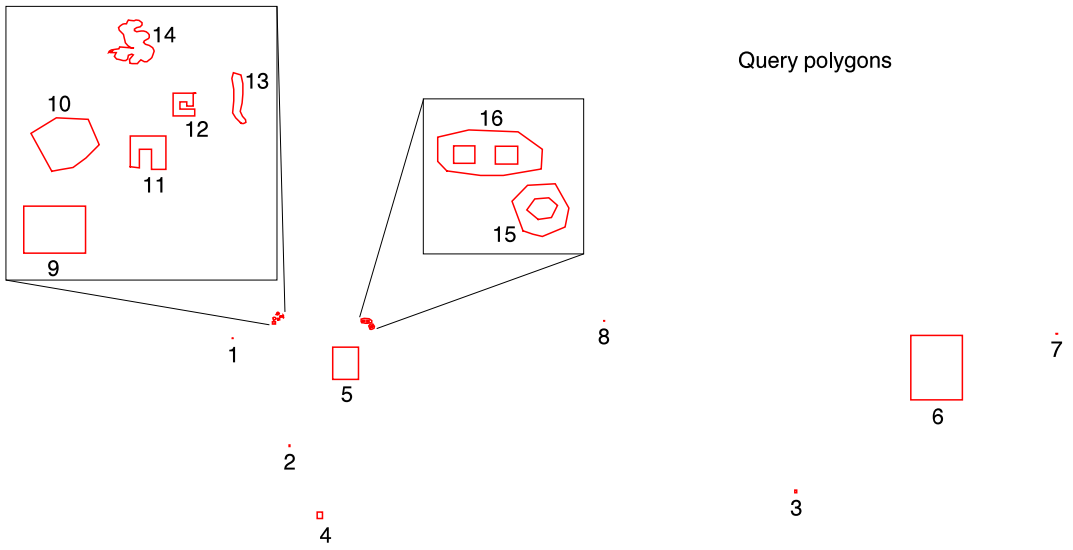

Figure 13: Cadastral area queries geometries

Nr	Geometry type	Short description	Area m^2	Length m
17	polyline	Motorway A50	0	14631
18	polyline	Motorway A50	0	21592
19	polyline	Motorway	0	55981
20	polyline	Motorway A50	0	34146
21	polyline	Motorway A12 oost	0	75415
22	multi polyline	Motorway	0	97356
23	polyline	Motorway A1	0	95344
24	polyline	Motorway	0	63542
25	polyline	Important Route	0	77035
26	multi polyline	Motorway	0	100966
27	multi polyline	Motorway	0	58435

Table 14: Cadastral line queries characteristics

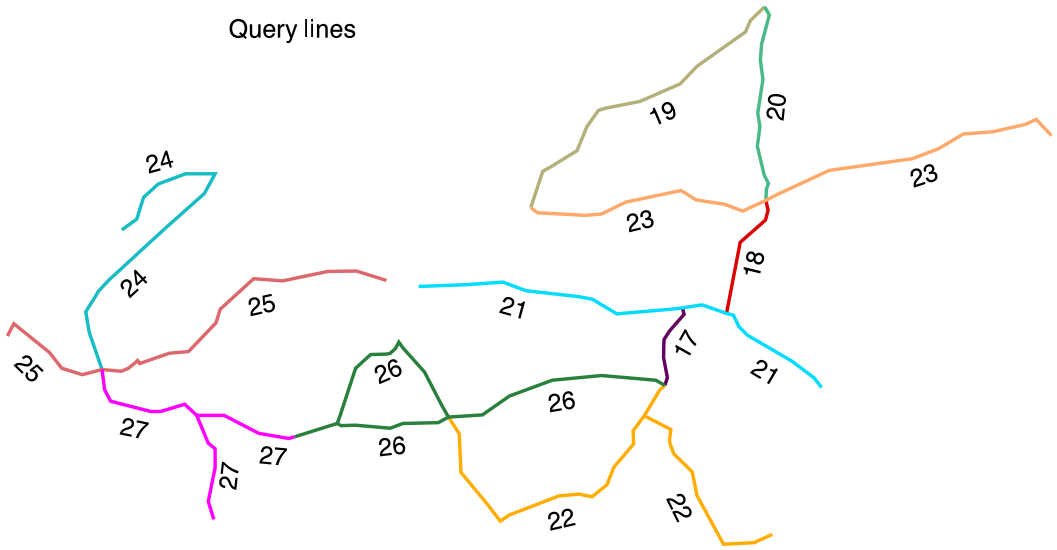

Figure 14: Cadastral line queries geometries

Oracle query scripts

The overall script `query_all.ora` to run the queries against the Oracle database:

```
#!/bin/sh
#               query_all.ora  06-03-2001
#
# Overall script to query the Oracle database (all queries)

USR_PW=${1:-'kadtest/kadtest'}
set -x

# admin queries
/bin/time query_adm.ora $USR_PW

# geom filter queries
/bin/time query_base_filter.ora $USR_PW shape xfio_boundary 'select count(*)'

# geom overlap query
/bin/time query_base.ora $USR_PW bbox     xfio_boundary 'select count(*)'
/bin/time query_base.ora $USR_PW shape    xfio_boundary 'select count(*)'
/bin/time query_base.ora $USR_PW bbox     xfio_boundary 'drop table t;
    create table t as select *'
/bin/time query_base.ora $USR_PW bbox     xfio_line     'select count(*)'
/bin/time query_base.ora $USR_PW location xfio_text     'select count(*)'
/bin/time query_base.ora $USR_PW location xfio_gcpnt    'select count(*)'
/bin/time query_base.ora $USR_PW location xfio_sympnt   'select count(*)'
/bin/time query_base.ora $USR_PW bbox     xfio_parcel   'select count(*)'

# geom-admin overlap queries
/bin/time query_base.ora $USR_PW geo_bbox akr_objectSOM     'select count(*)'
/bin/time query_base.ora $USR_PW geo_bbox akr_objectSOM     'select avg(koopsom)'
/bin/time query_base.ora $USR_PW geo_bbox akr_recht_subject 'select count(*)'
/bin/time query_base.ora $USR_PW geo_bbox akr_recht_subject 'drop table t;
    create table t as select *'

# geom other queries
/bin/time query_base_dist.ora   $USR_PW location xfio_sympnt   'select count(*)'
/bin/time query_base_buf.ora    $USR_PW location xfio_sympnt   'select count(*)'
/bin/time query_base_join.ora   $USR_PW location xfio_sympnt   'select count(*)'
/bin/time query_base_clip.ora   $USR_PW location xfio_sympnt
/bin/time query_base_nn1000.ora $USR_PW location xfio_sympnt

/bin/time query_base_dist.ora   $USR_PW shape    xfio_boundary 'select count(*)'
/bin/time query_base_buf.ora    $USR_PW shape    xfio_boundary 'select count(*)'
/bin/time query_base_join.ora   $USR_PW shape    xfio_boundary 'select count(*)'
/bin/time query_base_clip.ora   $USR_PW shape    xfio_boundary
/bin/time query_base_nn1000.ora $USR_PW shape    xfio_boundary

exit
```

The basic script `query_base.ora` which contains all the query geometries:

```
#!/bin/sh
#               query_base.ora  20-01-2001
# Basic script to query the Oracle database (using mdsys.sdo_relate)

ORACLE_SID=kadtest
export ORACLE_SID

USR_PW=${1:-'/'}
ATTR=${2:-shape}
TABLE=${3:-xfio_boundary}
QUERY_TYPE=${4:-'select /*+ ORDERED */ count(*)'}

sqlplus $USR_PW << EOB

column avg(koopsom) format 9999990.99
-- Increase cursor caching for improved R-tree performance
alter session set session_cached_cursors=300;
set timing on

prompt >> Query 1 box Scheveningen
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(78550000,457900000,78650000,458025000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 2 box Rotterdam Spangen
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(89500000,437000000,89700000,437250000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 3 Nijmegen Keizer Karelplein
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(187500000,428000000,187900000,428500000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 4 Binnenbedijkte Maas
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(95000000,423000000,96000000,424250000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 5 Hazerswoude
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(98000000,450000000,103000000,456250000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 6 Achterhoek West
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(210000000,446000000,220000000,458500000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;
```

```
prompt >> Query 7 in perceel EBG02R00364
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(238100000,458800000,238400000,458900000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 8 Mil terrein Soesterberg
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(150500000,461300000,150650000,461400000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 9 rechthoek
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(86275806,460673681,86275806,461137450,
  86882416,461137450,86882416,460673681,86275806,460673681)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 10 polygon met 8 punten
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(86553033,461481233,86345860,461854954,
  86597716,462009319,86910505,461993069,87020183,461741215,86886133,461611222,
  86760205,461517794,86630213,461497483,86553033,461481233)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 11 N vorming polygon
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(87325629,461525642,87325629,461828507,
  87675279,461828507,87675279,461501020,87534926,461501020,87534926,461698004,
  87414273,461698004,87414273,461515794,87325629,461525642)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 12 C vormig polygon, tegen de klok in
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(87968294,462249562,87749147,462249562,
  87749147,462025492,87960906,462025492,87960906,462091974,87815630,462091974,
  87815630,462163381,87882111,462163381,87882111,462123984,87946132,462123984,
  87946132,462247100,87968294,462249562)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 13 strookvormig polygon
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(88412072,461959592,88364878,462009206,
  88335837,462056398,88345517,462177406,88346727,462288732,88327366,462397639,
  88343097,462453302,88418121,462431521,88436273,462343186,88436273,462243960,
  88426592,462153204,88410862,462070919,88431432,462023727,88462895,461988634,
  88464104,461964433,88441113,461953542,88425383,461949912,88412072,461959592)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 14 grillig polygon met 50+ punten
```

```
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1),
  mdsys.SDO_ORDINATE_ARRAY(87353257,462695317,87331476,462700157,
  87302434,462717098,87275813,462748560,87270973,462767922,87266132,462796962,
  87258872,462830846,87239511,462850207,87215309,462869567,87210469,462896189,
  87227410,462915550,87249191,462932491,87266132,462934911,87287914,462934911,
  87295174,462951852,87307274,462971213,87326636,462968794,87362938,462963953,
  87379879,462971213,87411341,462968794,87440383,462947012,87442803,462922810,
  87421022,462905870,87387140,462898609,87367778,462884088,87367778,462867147,
  87387140,462857467,87406500,462859886,87433122,462864728,87454903,462864728,
  87483945,462857467,87500886,462840525,87510566,462826004,87510566,462801804,
  87505726,462787283,87488786,462782441,87464584,462767922,87457323,462750980,
  87474265,462734039,87510566,462721938,87546869,462700157,87556549,462678376,
  87558969,462666275,87554129,462642074,87549289,462632393,87539609,462600931,
  87522668,462586411,87512987,462579150,87491206,462562209,87464584,462562209,
  87428282,462579150,87408920,462557369,87394400,462542848,87343577,462542848,
  87321796,462547688,87319375,462583990,87355677,462629973,87341157,462634813,
  87307274,462622712,87300014,462586411,87280653,462571890,87254031,462571890,
  87212889,462583990,87191108,462588830,87166906,462613032,87186267,462651754,
  87137864,462649334,87147545,462661435,87113663,462651754,87120924,462663855,
  87137864,462675956,87183847,462685636,87212889,462688056,87220150,462707418,
  87353257,462695317)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 15 donut
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL,
  mdsys.SDO_ELEM_INFO_ARRAY(1,1003,1,19,2003,1),
  mdsys.SDO_ORDINATE_ARRAY(105162551,459792109,104945729,460372674,
  105247507,460684152,105799724,460716011,106065213,460238131,105990876,459866447,
  105544856,459675294,105353703,459717772,105162551,459792109,
  105458979,460013950,105232708,460201590,105396182,460418664,105672291,460439903,
  105839778,460289891,105723883,460052582,105458979,460013950)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 16 bril (exterior met verkeerde richting)
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL,
  mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 1, 21, 2003,3, 25, 2003,3),
  mdsys.SDO_ORDINATE_ARRAY(103654574,460970880,103474041,461162032,
  103474041,461639912,104089976,461777967,105061624,461741343,105544856,461395663,
  105523616,461013359,104769627,460885924,104323607,460885924,103654574,460970880,
  104610334,461108935,105056356,461459380,
  103792627,461119555,104206790,461470000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 17 Motorway A50
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 2, 1),
  mdsys.SDO_ORDINATE_ARRAY(181363000,435240000,181790000,436533000,
  181118000,440000000,181205000,443059999,182124999,444543999,184649000,447451000,
  184367999,448494999)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 18 Motorway A50
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
```

```
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 2, 1),
  mdsys.SDO_ORDINATE_ARRAY(191924000,447577000,194299000,459754999,
  198593000,463596000,199023000,465268000,198700000,466934000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 19 Motorway
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 2, 1),
  mdsys.SDO_ORDINATE_ARRAY(158382000,465719999,160393000,471918000,
  161921000,472748000,166302000,475450000,167973000,479624000,169952000,482374999,
  171115000,482710999,177073000,483902000,184019000,486930000,186840000,489929000,
  195163000,495831000,196071000,497841000,198463000,500119999)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 20 Motorway A50
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 2, 1),
  mdsys.SDO_ORDINATE_ARRAY(198700000,466934000,198740000,468692000,
  198738000,468848000,199139000,469852999,198409000,471304000,197219000,476245000,
  197689000,479586000,197259000,481941000,198198000,487657000,197719000,491045999,
  197968000,493773999,199249000,498780999,198463000,500119999)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 21 Motorway A12 oost
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 2, 1),
  mdsys.SDO_ORDINATE_ARRAY(139088000,452172000,147112000,452509000,
  153591000,452926000,157569000,451469000,166686000,450436000,168957999,449997000,
  173144999,447474000,183374000,448334000,184367999,448494999,187740999,449050000,
  191924000,447577000,193086000,447252000,194084000,445257999,195516999,443823000,
  203087999,439418000, 206934000,436530000,208258000,434875000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 22 Motorway, multiple polyline
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1,2,1,49,2,1),
  mdsys.SDO_ORDINATE_ARRAY(144286000,429821000,146072000,426902999,
  146405000,420338999,150935000,414793000,153130000,411899999,154651000,412934000,
  163215999,416278000,166648000,416585000,168845000,416090000,171428000,418156000,
  172591000,421196999,176018000,425158999,175947000,427406000,177795999,429971999,
  182389000,427659000,182135999,425878000,182229000,425265999,182873999,423490000,
  186030999,420415999,186871000,416360999,191420000,407993000,196848999,408258000,
  197021000,408426000,199791000,409650999,
  181363000,435240000,180452999,434378999,177795999,429971999)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 23 Motorway A1
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 2, 1),
  mdsys.SDO_ORDINATE_ARRAY(158382000,465719999,159511000,464796999,
  167618000,464400000,170516999,464597000,174647999,466669999,184147000,468607999,
  186720000,467031999,191560000,466264999,194729000,465099000,198700000,466934000,
  200558000,467929999,209536000,472103999,223751999,474060000,228200000,475770999,
  232562000,478340000,237579000,478688000,243117999,479929999,245011000,480832000,
  247709999,478032999)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;
```

```
prompt >> Query 24 Motorway
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 2, 1),
  mdsys.SDO_ORDINATE_ARRAY(88223000,461906000,90745000,463742000,
  91922000,467498000,94430999,469725999,99188000,471539000,104242000,471474000,
  102361999,468110999,95825999,462310000,85947000,453236000,84204999,451301999,
  83770000,450651999,82048000,447840000,82615000,444327000,84807000,437953999)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 25 Important Route
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 2, 1),
  mdsys.SDO_ORDINATE_ARRAY(133571999,453215999,128432000,454798000,
  123538000,454771000,115848999,453179000,110860000,453477000,105068999,448301999,
  104393999,445972999,99635000,441017000,96380000,440698000,91320000,438969000,
  90947999,439484999,90565000,439122999,90009000,438663000,89417999,438170000,
  88173000,437651999,84807000,437953999,81423000,437079000,77918000,438189000,
  75773000,440870000,69669999,445811000,68623000,443658999)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 26 Motorway, multiple polylines
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1,2,1,27,2,1),
  mdsys.SDO_ORDINATE_ARRAY(181363000,435240000,179831000,436133999,
  170473000,436930000,162041000,436120999,154687000,433401999,150038000,430166999,
  144286000,429821000,142526999,428857000,136430000,428697999,134235000,427872000,
  128290999,428356000,125817000,428314000,125129999,428674000,
  117876440,426434111,124198000,428380000,125129999,428674000,125563999,429873000,
  128342999,438190000,130918999,440491000,133621000,440566999,134233000,440732000,
  135130000,441562000,135725000,442639000,136491000,441578999,140175999,437507000,
  143235999,431525000,144286000,429821000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

prompt >> Query 27 Motorway, multiple polylines
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2002, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1,2,1,25,2,1),
  mdsys.SDO_ORDINATE_ARRAY(84807000,437953999,85239999,434499000,
  86280000,432527000,93212000,430722999,94727999,430761000,95141999,430777999,
  98984999,432012999,101029999,430098999,105748000,430072999,111708000,426970000,
  116842999,426116000,117876440,426434111,
  101029999,430098999,103081000,425294000,104237000,424313999,104246000,421143999,
  103081999,415281999,104031999,412246000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;

exit
EOB
```

Fragments from the script `query_base_join.ora`:

```
#! /bin/sh
#             query_base_join.ora  02-02-2001
#
# Basic script to insert query geometries and query the Oracle database
# with spatial join
```

```
sqlplus $USR_PW << EOB
set timing on
set echo on

drop table query_geom;
create table query_geom (
 shape mdsys.sdo_geometry not null,
 TAG number(11) not null)
 storage (initial 8m next 8m pctincrease 0) pctfree 10 nologging ;

/* query 1 box scheveningen */
insert into query_geom (shape,TAG) values (
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(78550000,457900000,78650000,458025000)),
1);
      .
      .
      .
insert into user_sdo_geom_metadata
  values ('query_geom','shape',
  mdsys.sdo_dim_array(mdsys.sdo_dim_element('X',-25000000,325000000,0.5),
                      mdsys.sdo_dim_element('Y',275000000,625000000,0.5)),NULL);

create index query_geom_idx on query_geom (shape)
indextype is mdsys.spatial_index;

select TAG,SDO_GEOM.SDO_AREA(q.shape, 0.5)/1000000,
       SDO_GEOM.SDO_LENGTH(q.shape, 0.5)/1000
from query_geom q;

$QUERY_TYPE,q.TAG from $TABLE t, query_geom q
where mdsys.sdo_relate (t.$ATTR,q.shape,
  'mask=ANYINTERACT querytype = JOIN') = 'TRUE'
group by q.TAG;

$QUERY_TYPE,q.TAG from $TABLE t, query_geom q
where mdsys.sdo_relate (t.$ATTR,q.shape,
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE'
group by q.TAG;

exit
EOB
```

Fragment from the start of the script `query_base_dist.ora`:

```
#!/bin/sh
#               query_base_dist.ora  02-02-2001
#
# Basic script to query the Oracle database (using mdsys.sdo_within_distance)

sqlplus $USR_PW << EOB
set timing on

prompt >> Query 1 box Scheveningen
```

```
$QUERY_TYPE from $TABLE where mdsys.sdo_within_distance ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(78550000,457900000,78650000,458025000)),
  'distance = 100000') = 'TRUE' ;

prompt >> Query 2 box Rotterdam Spangen
$QUERY_TYPE from $TABLE where mdsys.sdo_within_distance ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(89500000,437000000,89700000,437250000)),
  'distance = 100000') = 'TRUE' ;
      .
      .
      .
```

Fragment from the start of the script `query_base_nn1000.ora`:

```
#!/bin/sh
#               query_base_nn1000.ora  02-02-2001
#
# Basic script to find 1000 nearest neighbors to given query
# geometries in the Oracle database

sqlplus $USR_PW << EOB
set timing on

prompt >> Query 1 box Scheveningen
drop table nn1;
create table nn1 as select ogroup, object_id, tmax, $ATTR
from $TABLE where mdsys.sdo_nn ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(78550000,457900000,78650000,458025000)),
  'sdo_num_res=1000') = 'TRUE' ;
select count (*) from nn1;
      .
      .
      .
```

Fragment from the start of the script `query_base_buf.ora`:

```
#!/bin/sh
#               query_base_buf.ora  02-02-2001
#
# Basic script to query the Oracle database (using sdo_geom.sdo_buffer)

sqlplus $USR_PW << EOB
set timing on

prompt >> Query 1 box Scheveningen
$QUERY_TYPE from $TABLE where mdsys.sdo_relate ($ATTR,sdo_geom.sdo_buffer(
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(78550000,457900000,78650000,458025000)),100000,0.5),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;
      .
      .
      .
```

Fragment from the start of the script `query_base_clip.ora`:

```
#!/bin/sh
#               query_base_clip.ora  02-02-2001
#
# Basic script to clip geometries in the Oracle database

sqlplus $USR_PW << EOB
set timing on

prompt >> Query 1 box Scheveningen
drop table q1;
create table q1 as select
sdo_geom.sdo_intersection($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(78550000,457900000,78650000,458025000)),0.5) clip_geom
from $TABLE where mdsys.sdo_relate ($ATTR,
  mdsys.SDO_GEOMETRY(2003, NULL, NULL, mdsys.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
  mdsys.SDO_ORDINATE_ARRAY(78550000,457900000,78650000,458025000)),
  'mask=ANYINTERACT querytype = WINDOW') = 'TRUE' ;
select count (*) from q1;
      .
      .
      .
```

Fragment from the start of the script `query_adm.ora`:

```
#!/bin/sh
#                   query_adm.ora  28-03-2001
# Basic script to query the Oracle database, administrative queries

sqlplus $USR_PW << EOB
set timing on

/* aantal rechten 1 */
drop table tmp;
create table tmp as select * from lki_parcel where x_akr_objectnummer in
(select g_akr_objectnummer from object_parcel op,mo_recht r where
op.x_akr_objectnummer=r.x_akr_objectnummer and gerechtigde='0123456789');
select count(*) from tmp;
      .
      .
      .
```

Appendix E: Ingres cadastral data querying

The overall script to query the Ingres database `query_all.ing`:

```
#! /bin/sh
# Overall script to query the Ingres database

DB=${1:-kadtest}

query_adm.ing $DB

query_base.ing $DB bbox xfio_boundary 'select count(*)'
query_base.ing $DB shape xfio_boundary 'select count(*)'
query_base.ing $DB bbox xfio_boundary 'drop table t;\p\g \date;
    create table t as select *'
query_base.ing $DB bbox xfio_line 'select count(*)'
query_base.ing $DB location xfio_text 'select count(*)'
query_base.ing $DB location xfio_gcpnt 'select count(*)'
query_base.ing $DB location xfio_sympnt 'select count(*)'
query_base.ing $DB bbox xfio_parcel 'select count(*)'

query_base.ing $DB geo_bbox akr_recht_subject 'select count(*)'
query_base.ing $DB geo_bbox akr_recht_subject 'drop table t;\p\g \date;
    create table t as select *'
query_base.ing $DB geo_bbox akr_objectSOM 'select count(*)'
query_base.ing $DB geo_bbox akr_objectSOM 'select avg(koopsom)'
exit
```

The basic geometry query script `query_base.ing`:

```
#! /bin/sh
# Basic script to query the Ingres database

DB=${1:-kad}
ATTR=${2:-shape}
TABLE=${3:-xfio_boundary}
QUERY_TYPE=${4:-'select count(*)'}

sql $DB << EOB
set nologging;\p\g

/* query 1 */
\date
$QUERY_TYPE from $TABLE where overlaps
(ibox('((78550000,457900000),(78650000,458025000))'),$ATTR)=1;\p\g

/* query 2 */
\date
$QUERY_TYPE from $TABLE where overlaps
(ibox('((89500000,437000000),(89700000,437250000))'),$ATTR)=1;\p\g

/* query 3 */
\date
$QUERY_TYPE from $TABLE where overlaps
(ibox('((187500000,428000000),(187900000,428500000))'),$ATTR)=1;\p\g
```

```
/* query 4 */
\date
$QUERY_TYPE from $TABLE where overlaps
(ibox('((95000000,423000000),(96000000,424250000))'),$ATTR)=1;\p\g
.
.
.
/* query 15 */
\date
$QUERY_TYPE from $TABLE where overlaps
(ipolygon('((105162551,459792109),(104945729,460372674),(105247507,460684152),
(105799724,460716011),(106065213,460238131),(105990876,459866447),
(105544856,459675294),(105353703,459717772))'),$ATTR)=1
and not inside
($ATTR,ipolygon('((105458979,460013950),(105232708,460201590),
(105396182,460418664),(105672291,460439903),(105839778,460289891),
(105723883,460052582))'))=1;\p\g

/* query 16 */
\date
$QUERY_TYPE from $TABLE where overlaps
(ipolygon('((103654574,460970880),(103474041,461162032),(103474041,461639912),
(104089976,461777967),(105061624,461741343),(105544856,461395663),
(105523616,461013359),(104769627,460885924),(104323607,460885924))'),$ATTR)=1
and not inside
($ATTR,ibox('((104610334,461108935),(105056356,461459380))'))=1
and not inside
($ATTR,ibox('((103792627,461119555),(104206790,461470000))'))=1;\p\g

/* query 17 MotorwayA50 */
\date
$QUERY_TYPE from $TABLE where intersects
(iline('((181363000,435240000),(181790000,436533000),(181118000,440000000),
(181205000,443059999),(182124999,444543999),(184649000,447451000),
(184367999,448494999))'),$ATTR)=1;\p\g

/* query 18 Motorway A50 */
\date
$QUERY_TYPE from $TABLE where intersects
(iline('((191924000,447577000),(194299000,459754999),(198593000,463596000),
(199023000,465268000),(198700000,466934000))'),$ATTR)=1;\p\g
.
.
.
/* query 27 Motorway, multiple polylines */
\date
$QUERY_TYPE from $TABLE where intersects
(iline('((84807000,437953999),(85239999,434499000),(86280000,432527000),
(93212000,430722999),(94727999,430761000),(95141999,430777999),
(98984999,432012999),(101029999,430098999),(105748000,430072999),
(111708000,426970000),(116842999,426116000),(117876440,426434111))'),$ATTR)=1
or intersects
(iline('((101029999,430098999),(103081000,425294000),(104237000,424313999),
(104246000,421143999),(103081999,415281999),
(104031999,412246000))'),$ATTR)=1;\p\g
```

```
EOB
exit
```

The top of the basic admin query script `query_adm.ing`:

```
#! /bin/sh
# Basic script to query the Ingres database

DB=${1:-kadtest}

echo $DB

sql $DB << EOB
set nologging;\p\g

/* aantal rechten 1 */
\date
drop table tmp;\p\g
create table tmp as select * from lki_parcel where x_akr_objectnummer in
(select g_akr_objectnummer from object_parcel op,mo_recht r where
op.x_akr_objectnummer=r.x_akr_objectnummer and gerechtigde='2100192169');\p\g
select count(*) from tmp;\p\g
\date
drop table tmp;\p\g
create table tmp as select * from lki_parcel where x_akr_objectnummer in
(select g_akr_objectnummer from object_parcel op,mo_recht r where
op.x_akr_objectnummer=r.x_akr_objectnummer and gerechtigde='2559439701');\p\g
select count(*) from tmp;\p\g
\date
      .
      .
      .
```

Appendix F: Oracle geological data loading

This appendix contains some scripts that are used to load the geological data into Oracle. The first subsection contains the master script `convert_all.sh` that loads all data. Furthermore one of the scripts that are called by `convert_all.sh` is included. The script `load_wellpicks.sh` loads the wellpicks data. Other scripts are very similar to these scripts and are not included.

`Convert_all.sh` script

```
#!/bin/sh
#
# file: convert_all.sh
#   Loads all TNO-NITG Geological data into an Oracle8i database.
#
# History:
# 20-Dec-2000: creation (quak@geo.tudelft.nl)
# 14-Mar-2001: some additions (oosterom@geo.tudelft.nl)

set -v

#
# Load parameter file.
#
. params.sh

#
# Load all faultlines.
#
cat $DATADIR/faultlines/12br_br.asc | arc2ora.perl $DATABASE faultlines_12br_br
cat $DATADIR/faultlines/19no_br.asc | arc2ora.perl $DATABASE faultlines_19no_br
cat $DATADIR/faultlines/24tx_br.asc | arc2ora.perl $DATABASE faultlines_24tx_br
cat $DATADIR/faultlines/29rijn_br.asc | arc2ora.perl $DATABASE faultlines_29rijn_br
cat $DATADIR/faultlines/30df_br.asc | arc2ora.perl $DATABASE faultlines_30df_br
cat $DATADIR/faultlines/40al_br.asc | arc2ora.perl $DATABASE faultlines_40al_br
cat $DATADIR/faultlines/49bo_br.asc | arc2ora.perl $DATABASE faultlines_49bo_br
cat $DATADIR/faultlines/54ze_br.asc | arc2ora.perl $DATABASE faultlines_54ze_br
cat $DATADIR/faultlines/59ro_br.asc | arc2ora.perl $DATABASE faultlines_59ro_br
cat $DATADIR/faultlines/tze_br.asc | arc2ora.perl $DATABASE faultlines_tze_br

#
# Create view that contains union of all these sets.
#
sqlplus $DATABASE @faultlines.sql

#
# Load subcrops
#
cat $DATADIR/subcrops250/19no_sc.asc | arc2ora.perl $DATABASE subcrops250_19no_sc
cat $DATADIR/subcrops250/24tx_sc.asc | arc2ora.perl $DATABASE subcrops250_24tx_sc
cat $DATADIR/subcrops250/29rijn_sc.asc | arc2ora.perl $DATABASE subcrops250_29rijn_sc
cat $DATADIR/subcrops250/30df_sc.asc | arc2ora.perl $DATABASE subcrops250_30df_sc
cat $DATADIR/subcrops250/40al_sc.asc | arc2ora.perl $DATABASE subcrops250_40al_sc
```

```
cat $DATADIR/subcrops250/49bo_sc.asc | arc2ora.perl $DATABASE subcrops250_49bo_sc
cat $DATADIR/subcrops250/54ze_sc.asc | arc2ora.perl $DATABASE subcrops250_54ze_sc
cat $DATADIR/subcrops250/59ro_sc.asc | arc2ora.perl $DATABASE subcrops250_59ro_sc
cat $DATADIR/subcrops250/tze_sc.asc | arc2ora.perl $DATABASE subcrops250_tze_sc

sqlplus $DATABASE @subcrops.sql

#
# Load all horizons
#
DDD=$DATADIR/horizons250
cat $DDD/19b_tert250.asc | raster2ora.perl $DATABASE horizons_19b_tert250
cat $DDD/24b_ucret250.asc | raster2ora.perl $DATABASE horizons_24b_ucret250
cat $DDD/29b_lcret250.asc | raster2ora.perl $DATABASE horizons_29b_lcret250
cat $DDD/30b_ujura250.asc | raster2ora.perl $DATABASE horizons_30b_ujura250
cat $DDD/40b_ljura250.asc | raster2ora.perl $DATABASE horizons_40b_ljura250
cat $DDD/49b_ltrias250.asc | raster2ora.perl $DATABASE horizons_49b_ltrias250
cat $DDD/50t_zech250.asc | raster2ora.perl $DATABASE horizons_50t_zech250
cat $DDD/54b_zech250.asc | raster2ora.perl $DATABASE horizons_54b_zech250
cat $DDD/59b_rotl250.asc | raster2ora.perl $DATABASE horizons_59b_rotl250

sqlplus $DATABASE @horizons.sql

#
# Load all borders into table arcdata
#
sqlplus $DATABASE @arcdata.sql

for i in brab drent flevo fries gelder gron kb1 kb10 kb11_12 kb13_14 kb15 \
   kb2 kb3 kb4 kb5 kb6 kb7_8 kb9 limb noordh overij utr zeel zuidh
do
   cat $DATADIR/borders/$i.asc | arc2ora1.perl $DATABASE $i
done

insert_into_user_sdo_geom_metadata.sql $DATABASE ARCDATA GEOMETRY 0 325000 275000 625000 0.5

sqlplus $DATABASE @arcindex.sql

#
# Load all navseis
#
load_navseis.sh

#
# Load all wellpicks
#
load_wellpicks.sh

#
# End of convert_all.sh
#
exit
```

Load_wellpicks.sh script

```
#!/bin/sh
#
# This file loads TNO-NITG wellpicks data into an Oracle8i database.
#
# History:
# 15-Mar-2001: Extracted from 'convert_all.sh'. (quak@geo.tudelft.nl)
#

set -v

#
# Load parameter file.
#
. params.sh

#
# Convert data to Oracle sqlldr format.
#
cat $DATADIR/wellpicks/wells.dat | wellpicks2ora.perl > tmp_wellpicks.dat

#
# Create database table wellpicks.
#
sqlplus $DATABASE << EOF
set echo on;

drop table wellpicks;
create table wellpicks
(
   location mdsys.sdo_geometry,
   putnaam varchar2(23),
   formatie_cd varchar2(7),
   diepte float,
   dikte float
);
EOF

#
# Generate sqlldr controlfile.
#
cat > wellpicks.ctl << EOF
load data
infile 'tmp_wellpicks.dat' "str"
into table wellpicks
fields terminated by ','
trailing nullcols (
   location column object
   (
      sdo_gtype integer external,
      sdo_point column object
         (x float external,
          y float external)
   ),
```

```
   putnaam char terminated by ',' enclosed by '<' and '>',
   formatie_cd char terminated by ',' enclosed by '<' and '>',
   diepte float external,
   dikte float external
)
EOF

#
# Call sqlldr to load data into database.
#
sqlldr $DATABASE CONTROL=wellpicks.ctl READSIZE=15000000 BINDSIZE=15000000 ROWS=10000

#
# Insert Metadata into database.
#
insert_into_user_sdo_geom_metadata.sql $DATABASE WELLpicks LOCATION 0 325000 275000 625000 0.5

#
# Create indices.
#
sqlplus $DATABASE << EOF
create index wellpicks_1 on wellpicks(location)
indextype is mdsys.spatial_index
parameters ('initial=4m next=4m pctincrease=0');

drop index wellpicks_2;
create index wellpicks_2 on wellpicks(formatie_cd)
storage( initial 4m next 4m pctincrease 0);

analyze table wellpicks compute statistics;
analyze table wellpicks_1_rt$ compute statistics;
EOF

#
# Clean up.
#
rm -f tmp_wellpicks.dat
rm -f wellpicks.ctl
rm -f wellpicks.log
exit
```

Appendix G: Cadastral query results

This appendix contains detailed information on the cadastral query results. Most of the results are for the Oracle database, sometimes Ingres results are included for comparisons. In order to obtain fair and reproducable results, the server was rebooted before each benchmark (and only relevant processes started).

The measurements are collected and stored in an Excel spreadsheet format. All timings are in the h:mm:ss format, so all measurements are rounded to whole seconds. For query sequences with many short queries (less than 1 sec) this can result in some rounding error. The following tables are included in this appendix:

- Table 15: Number of selected records geom and geom-admin queries;
- Table 16: Oracle geom filter/overlap queries (production);
- Table 17: Oracle geom filter/overlap queries (patch);
- Table 18: Oracle geom filter/overlap queries (patch/random);
- Table 19: Ingres geom overlap queries;
- Table 20: Geom-admin combination queries Oracle and Ingres;
- Table 21: Oracle distance, buffer, join, clip, nearest neighbor queries (production);
- Table 22: Oracle distance, buffer, join, clip, nearest neighbor queries (patch);
- Table 23: Administrative entrance queries Oracle and Ingres.

# Oracle filter	# bnd/box count	# bnd/shp count	# ln/box count	# txt/loc count	# gcp/loc count	# sym/loc count	# par/box count	**totals count**	# akr rechtsubj	# akr objectsom	avg value objectsom
423	423	415	58	78	0	0	231	1205	297	114	50210.09
271	271	267	274	71	0	0	115	998	250	106	111601.26
2118	2118	2118	1890	389	0	169	623	7307	867	513	232298.36
784	784	782	409	40	0	473	301	2789	163	119	55338.72
17250	17250	17241	4181	1400	56	6053	7594	53775	3748	3079	314187.98
142405	*142405*	*142405*	*8656*	*9186*	*51*	*13398*	*51951*	368052	*21615*	*17447*	145713.29
1	1	1	1	0	0	1	10	14	3	3	0.00
0	0	0	0	0	0	0	0	0	0	0	
508	508	507	187	126	1	129	198	1656	336	196	335736.63
2114	1692	1680	401	227	103	171	649	4923	800	551	525694.70
440	396	392	103	89	27	44	198	1249	245	182	370951.10
378	300	293	113	46	0	22	137	911	534	114	83692.98
919	462	442	57	39	12	42	227	1281	311	197	121858.88
1082	573	549	260	125	2	93	257	1859	844	225	333121.00
8472	5189	5159	900	892	8	0	2291	14439	3242	1543	202004.03
13699	10065	10035	864	673	0	0	3891	25528	1895	1127	438568.46
190864	**182437**	**182286**	**18354**	**13381**	**260**	**20595**	**68673**	**485986**	**35150**	**25516**	
27911	340	267	14	0	0	0	488	1109	175	158	96933.78
50488	242	125	5	0	0	0	248	620	260	239	36055.49
814919	1647	1133	159	0	0	0	2176	5115	1944	1607	1289714.92
94963	799	670	47	0	0	0	878	2394	527	467	73860.83
956608	1411	931	206	0	0	0	1899	4447	924	824	525277.80
895891	*1247*	*884*	*164*	*0*	*0*	*0*	*1677*	3972	*1221*	*1047*	66686.86
735999	1011	792	89	0	0	0	1425	3317	812	711	177272.32
1932126	2851	1753	314	0	0	0	3507	8425	2523	1499	260322.07
2027970	2216	1530	740	0	0	0	2914	7400	2645	1500	209554.18
1028776	*3305*	*2385*	*297*	*0*	*0*	*0*	*3880*	9867	*2081*	*1980*	176762.11
1211695	*1739*	*1426*	*634*	*0*	*0*	*0*	*2290*	6089	*1352*	*1305*	385622.48
9777346	**16808**	**11896**	**2669**	**0**	**0**	**0**	**21382**	**52755**	**14464**	**11337**	
9968210	**199245**	**194182**	**21023**	**13381**	**260**	**20595**	**90055**	**538741**	**49614**	**36853**	

Table 15: Number of selected records (#) geom and geom-admin queries

Oracle software version: production
No special clustering of LKI data

	bnd/shp ***filter***	bnd/box count	bnd/shp count	bnd/box create	line/box count	text/loc count	gcpnt/loc count	sym/loc count	par/box count	**totals** (no filter)
1	0:00:02	0:00:03	0:00:04	0:00:03	0:00:01	0:00:01	0:00:00	0:00:00	0:00:01	0:00:13
2	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
3	0:00:01	0:00:01	0:00:00	0:00:02	0:00:01	0:00:00	0:00:00	0:00:00	0:00:01	0:00:05
4	0:00:01	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
5	0:00:07	0:00:08	0:00:08	0:00:11	0:00:02	0:00:01	0:00:00	0:00:03	0:00:04	0:00:37
6	*0:00:43*	*0:00:44*	*0:00:44*	*0:01:08*	*0:00:03*	*0:00:02*	*0:00:00*	*0:00:03*	*0:00:15*	*0:02:59* *large area*
7	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
8	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
9	0:00:00	0:00:01	0:00:01	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:03
10	0:00:01	0:00:01	0:00:01	0:00:02	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	0:00:05
11	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
12	0:00:00	0:00:01	0:00:01	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:03
13	0:00:00	0:00:01	0:00:01	0:00:02	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	0:00:05
14	0:00:00	0:00:02	0:00:02	0:00:02	0:00:01	0:00:00	0:00:00	0:00:00	0:00:01	0:00:08
15	0:00:03	0:00:11	0:00:10	0:00:12	0:00:02	0:00:01	0:00:00	0:00:00	0:00:05	0:00:41
16	0:00:06	0:00:17	0:00:17	0:00:20	0:00:02	0:00:01	0:00:00	0:00:00	0:00:06	0:01:03
Atot	**0:01:04**	**0:01:30**	**0:01:29**	**0:02:09**	**0:00:12**	**0:00:06**	**0:00:00**	**0:00:06**	**0:00:35**	**0:06:07**
17	0:00:10	0:00:34	0:00:34	0:00:36	0:00:06	0:00:06	0:00:00	0:00:06	0:00:14	0:02:16
18	0:00:18	0:01:01	0:01:01	0:01:03	0:00:37	0:00:09	0:00:00	0:00:06	0:00:22	0:04:19
19	0:03:48	0:17:53	0:18:33	0:18:20	0:05:05	0:01:43	0:00:03	0:02:47	0:06:03	1:10:27
20	0:00:12	0:01:40	0:01:39	0:01:41	0:00:28	0:00:07	0:00:00	0:00:06	0:00:36	0:06:17
21	0:04:22	0:22:01	0:22:38	0:21:55	0:04:24	0:02:30	0:00:03	0:03:36	0:07:16	1:24:23
22	0:04:03	0:19:59	0:20:28	0:20:35	0:05:13	0:01:47	0:00:03	0:03:22	0:06:42	1:18:09
23	0:02:51	0:14:24	0:14:55	0:16:05	0:04:21	0:01:29	0:00:03	0:02:15	0:05:01	0:58:33
24	0:08:38	0:58:49	0:58:45	0:57:44	0:09:54	0:03:04	0:00:14	0:03:49	0:15:40	3:27:59
25	0:05:42	1:00:34	1:00:10	0:59:25	0:15:58	0:04:46	0:00:11	0:04:14	0:16:32	3:41:50
26	0:04:46	0:25:00	0:24:42	0:26:07	0:02:56	0:01:27	0:00:02	0:04:00	0:07:06	1:31:20
27	0:05:01	0:29:36	0:29:23	0:29:14	0:14:43	0:03:26	0:00:10	0:03:17	0:08:52	1:58:41
Ltot	**0:39:51**	**4:11:31**	**4:12:48**	**4:12:45**	**1:03:45**	**0:20:34**	**0:00:49**	**0:27:38**	**1:14:24**	**15:44:14**
Tot	**0:40:55**	**4:13:01**	**4:14:17**	**4:14:54**	**1:03:57**	**0:20:40**	**0:00:49**	**0:27:44**	**1:14:59**	**15:50:21**

Table 16: Oracle geom filter/overlap queries (production)

Oracle software version: patch 0115
No special clustering of LKI data

	bnd/shp ***filter***	bnd/box count	bnd/shp count	bnd/box create	line/box count	text/loc count	gcpnt/loc count	sym/loc count	par/box count	**totals** (no filter)
1	0:00:02	0:00:04	0:00:00	0:00:02	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:07
2	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
3	0:00:01	0:00:01	0:00:00	0:00:01	0:00:01	0:00:00	0:00:00	0:00:00	0:00:01	0:00:04
4	0:00:01	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
5	0:00:07	0:00:08	0:00:03	0:00:05	0:00:02	0:00:01	0:00:00	0:00:03	0:00:04	0:00:26
6	*0:00:43*	0:00:45	*0:00:16*	*0:00:36*	*0:00:03*	0:00:02	0:00:00	0:00:03	*0:00:16*	*0:02:01* *large area*
7	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
8	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
9	0:00:00	0:00:01	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:02
10	0:00:01	0:00:01	0:00:01	0:00:02	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	0:00:05
11	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
12	0:00:00	0:00:01	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:02
13	0:00:01	0:00:01	0:00:01	0:00:02	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	0:00:05
14	0:00:00	0:00:02	0:00:01	0:00:02	0:00:01	0:00:00	0:00:00	0:00:00	0:00:01	0:00:07
15	0:00:03	0:00:11	0:00:09	0:00:10	0:00:02	0:00:01	0:00:00	0:00:00	0:00:05	0:00:38
16	0:00:05	0:00:17	0:00:14	0:00:16	0:00:02	0:00:01	0:00:00	0:00:00	0:00:06	0:00:56
Atot	**0:01:04**	**0:01:32**	**0:00:45**	**0:01:23**	**0:00:12**	**0:00:05**	**0:00:00**	**0:00:06**	**0:00:35**	**0:04:38**
17	0:00:10	0:00:04	0:00:03	0:00:03	0:00:01	0:00:01	0:00:00	0:00:01	0:00:02	0:00:15
18	0:00:18	0:00:06	0:00:04	0:00:05	0:00:01	0:00:02	0:00:00	0:00:02	0:00:03	0:00:23
19	0:03:48	0:00:50	0:00:37	0:00:36	0:00:08	0:00:07	0:00:00	0:00:12	0:00:23	0:02:53
20	0:00:12	0:00:11	0:00:07	0:00:07	0:00:02	0:00:01	0:00:00	0:00:01	0:00:05	0:00:34
21	0:04:22	0:01:24	0:01:06	0:01:07	0:00:17	0:00:10	0:00:00	0:00:15	0:00:36	0:04:55
22	0:04:03	0:00:25	0:00:20	0:00:21	0:00:13	0:00:04	0:00:00	0:00:08	0:00:14	0:01:45
23	0:02:56	0:00:30	0:00:21	0:00:21	0:00:06	0:00:04	0:00:01	0:00:10	0:00:14	0:01:47
24	0:08:40	0:03:02	0:02:25	0:02:24	0:01:13	0:00:19	0:00:02	0:00:28	0:01:12	0:11:05
25	0:05:44	0:02:22	0:01:50	0:01:52	0:02:10	0:00:29	0:00:02	0:00:24	0:00:55	0:10:04
26	0:04:47	0:01:53	0:01:44	0:01:48	0:00:20	0:00:12	0:00:00	0:00:35	0:00:57	0:07:29
27	0:05:02	0:01:10	0:01:05	0:01:08	0:02:06	0:00:27	0:00:00	0:00:18	0:00:34	0:06:48
Ltot	**0:40:02**	**0:11:57**	**0:09:42**	**0:09:52**	**0:06:37**	**0:01:56**	**0:00:05**	**0:02:34**	**0:05:15**	**0:47:58**
Tot	**0:41:06**	**0:13:29**	**0:10:27**	**0:11:15**	**0:06:49**	**0:02:01**	**0:00:05**	**0:02:40**	**0:05:50**	**0:52:36**

Table 17: Oracle geom filter/overlap queries (patch)

Oracle software version: patch 0115
LKI data loaded (= clustered) randomly

	bnd/shp ***filter***	bnd/box count	bnd/shp count	bnd/box create	line/box count	text/loc count	gcpnt/loc count	sym/loc count	par/box count	**totals** (no filter)
1	0:00:08	0:00:06	0:00:04	0:00:05	0:00:01	0:00:02	0:00:00	0:00:00	0:00:03	0:00:21
2	0:00:03	0:00:02	0:00:02	0:00:03	0:00:03	0:00:01	0:00:00	0:00:00	0:00:01	0:00:12
3	0:00:02	0:00:18	0:00:16	0:00:22	0:00:19	0:00:03	0:00:00	0:00:01	0:00:06	0:01:25
4	0:00:08	0:00:07	0:00:06	0:00:09	0:00:00	0:00:00	0:00:00	0:00:04	0:00:03	0:00:29
5	0:02:45	0:02:28	0:02:10	0:02:50	0:00:41	0:00:11	0:00:01	0:00:38	0:01:11	0:10:10
6	*0:20:05*	*0:28:20*	*0:17:48*	*0:20:33*	*0:01:20*	*0:00:53*	*0:00:00*	*0:00:58*	*0:06:34*	*1:16:26* *large area*
7	0:00:04	0:00:00	0:00:04	0:00:04	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	0:00:09
8	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
9	0:00:04	0:00:10	0:00:04	0:00:05	0:00:02	0:00:01	0:00:00	0:00:01	0:00:01	0:00:24
10	0:00:04	0:00:42	0:00:17	0:00:18	0:00:06	0:00:02	0:00:01	0:00:01	0:00:05	0:01:32
11	0:00:03	0:00:09	0:00:04	0:00:04	0:00:01	0:00:01	0:00:00	0:00:00	0:00:02	0:00:21
12	0:00:03	0:00:08	0:00:03	0:00:04	0:00:01	0:00:00	0:00:00	0:00:00	0:00:01	0:00:17
13	0:00:07	0:00:19	0:00:08	0:00:09	0:00:01	0:00:00	0:00:00	0:00:00	0:00:03	0:00:40
14	0:00:08	0:00:24	0:00:09	0:00:10	0:00:05	0:00:01	0:00:00	0:00:01	0:00:03	0:00:53
15	0:01:05	0:02:58	0:01:11	0:01:15	0:00:15	0:00:07	0:00:00	0:00:00	0:00:25	0:06:11
16	0:01:43	0:04:42	0:01:55	0:02:00	0:00:11	0:00:05	0:00:00	0:00:00	0:00:36	0:09:29
Atot	**0:26:32**	**0:40:53**	**0:24:21**	**0:28:12**	**0:03:06**	**0:01:27**	**0:00:02**	**0:01:44**	**0:09:15**	**1:49:00**
17	0:03:35	0:00:47	0:00:20	0:00:20	0:00:05	0:00:03	0:00:00	0:00:02	0:00:09	0:01:46
18	0:06:22	0:01:26	0:00:35	0:00:36	0:00:03	0:00:03	0:00:00	0:00:04	0:00:11	0:02:58
19	1:48:26	0:12:11	0:04:59	0:05:00	0:00:49	0:00:22	0:00:01	0:00:33	0:01:39	0:25:34
20	0:14:42	0:02:17	0:00:56	0:00:57	0:00:11	0:00:02	0:00:01	0:00:04	0:00:18	0:04:46
21	2:20:59	0:23:21	0:09:13	0:09:19	0:01:56	0:00:32	0:00:01	0:00:36	0:02:40	0:47:38
22	1:58:15	0:07:00	0:02:46	0:02:48	0:01:18	0:00:10	0:00:00	0:00:15	0:00:45	0:15:02
23	1:32:19	0:07:05	0:02:47	0:02:49	0:00:27	0:00:08	0:00:02	0:00:17	0:00:44	0:14:19
24	4:00:08	0:42:04	0:20:10	0:20:14	0:07:10	0:00:37	0:00:04	0:00:37	0:03:59	1:34:55
25	4:10:42	0:15:22	0:15:14	0:15:21	0:08:40	0:00:38	0:00:02	0:00:26	0:02:22	0:58:05
26	2:08:36	0:14:27	0:14:26	0:14:36	0:00:50	0:00:13	0:00:00	0:00:35	0:01:58	0:47:05
27	2:40:28	0:09:26	0:09:25	0:09:28	0:05:33	0:00:28	0:00:00	0:00:19	0:01:05	0:35:44
Ltot	**21:04:32**	**2:15:26**	**1:20:51**	**1:21:28**	**0:27:02**	**0:03:16**	**0:00:11**	**0:03:48**	**0:15:50**	**5:47:52**
Tot	**21:31:04**	**2:56:19**	**1:45:12**	**1:49:40**	**0:30:08**	**0:04:43**	**0:00:13**	**0:05:32**	**0:25:05**	**7:36:52**

Table 18: Oracle geom filter/overlap queries (patch/random)

	no filter in Ingres	bnd/box count	bnd/shp count	bnd/box create	line/box count	text/loc count	gcpnt/loc count	sym/loc count	par/box count	**totals**	
1		0:00:01	0:00:01	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:01	0:00:04	
2		0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	
3		0:00:01	0:00:01	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:03	
4		0:00:01	0:00:00	0:00:01	0:00:01	0:00:00	0:00:00	0:00:00	0:00:01	0:00:04	
5		0:00:05	0:00:03	0:00:06	0:00:01	0:00:00	0:00:01	0:00:01	0:00:02	0:00:19	
6		*0:00:26*	*0:00:22*	*0:01:02*	*0:00:02*	*0:00:01*	*0:00:00*	*0:00:02*	*0:00:09*	*0:02:04*	*large area*
7		0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	
8		0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	
9		0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:01	
10		0:00:00	0:00:01	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	0:00:03	
11		0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	
12		0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	
13		0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	
14		0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	
15		0:00:02	0:00:02	0:00:03	0:00:01	0:00:00	0:00:00	0:00:00	0:00:01	0:00:09	
16		0:00:03	0:00:02	0:00:05	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01	0:00:11	
Atot		**0:00:40**	**0:00:32**	**0:01:22**	**0:00:06**	**0:00:02**	**0:00:01**	**0:00:03**	**0:00:16**	**0:03:02**	
17		0:00:07	0:00:06	0:00:05	0:00:01	0:00:01	0:00:00	0:00:01	0:00:03	0:00:24	
18		0:00:10	0:00:08	0:00:09	0:00:04	0:00:01	0:00:00	0:00:01	0:00:04	0:00:37	
19		0:02:28	0:02:10	0:01:49	0:00:32	0:00:10	0:00:00	0:00:15	0:00:48	0:08:12	
20		0:00:12	0:00:11	0:00:11	0:00:03	0:00:00	0:00:00	0:00:00	0:00:04	0:00:41	
21		0:02:56	0:02:54	0:02:22	0:00:31	0:00:15	0:00:01	0:00:21	0:00:59	0:10:19	
22		*0:09:54*	*0:11:46*	*0:11:04*	*0:02:24*	*0:00:54*	*0:00:02*	*0:01:05*	*0:03:10*	*0:40:19*	*multi polyline*
23		0:02:12	0:01:42	0:01:42	0:00:23	0:00:07	0:00:00	0:00:10	0:00:38	0:06:54	
24		0:04:54	0:03:33	0:04:13	0:01:10	0:00:20	0:00:01	0:00:25	0:01:54	0:16:30	
25		0:04:48	0:03:57	0:04:35	0:01:32	0:00:27	0:00:01	0:00:25	0:01:45	0:17:30	
26		*0:11:23*	*0:11:49*	*0:11:01*	*0:02:33*	*0:00:59*	*0:00:03*	*0:01:09*	*0:03:24*	*0:42:21*	*multi polyline*
27		*0:11:03*	*0:11:31*	*0:10:51*	*0:02:36*	*0:00:57*	*0:00:02*	*0:01:07*	*0:02:39*	*0:40:46*	*multi polyline*
Ltot		**0:50:07**	**0:49:47**	**0:48:02**	**0:11:49**	**0:04:11**	**0:00:10**	**0:04:59**	**0:15:28**	**3:04:33**	
Tot		**0:50:47**	**0:50:19**	**0:49:24**	**0:11:55**	**0:04:13**	**0:00:11**	**0:05:02**	**0:15:44**	**3:07:35**	

Table 19: Ingres geom overlap queries

	Oracle (software version: production) LKI: no special clustering AKR: mo_object, mo_subject index-organized					Oracle (software version: patch 0115) LKI: no special clustering AKR: only mo_object index-organized					Ingres				
	count akr rechtsubj	create akr rechtsubj	count akr objectsom	avg akr objectsom	**totals**	count akr rechtsubj	create akr rechtsubj	count akr objectsom	avg akr objectsom	**totals**	count akr rechtsubj	create akr rechtsubj	count akr objectsom	avg akr objectsom
1	0:00:53	0:01:25	0:00:11	0:00:33	0:03:02	0:00:02	0:00:01	0:00:00	0:00:00	0:00:03	0:00:03	0:00:01	0:00:00	0:00:01
2	0:00:50	0:01:21	0:00:10	0:00:12	0:02:33	0:00:01	0:00:01	0:00:00	0:00:00	0:00:02	0:00:01	0:00:01	0:00:00	0:00:00
3	0:00:52	0:01:31	0:00:10	0:00:12	0:02:45	0:00:06	0:00:01	0:00:00	0:00:01	0:00:08	0:00:03	0:00:04	0:00:01	0:00:01
4	0:00:49	0:01:37	0:00:10	0:00:12	0:02:48	0:00:01	0:00:01	0:00:00	0:00:00	0:00:02	0:00:01	0:00:01	0:00:00	0:00:00
5	0:00:56	0:01:38	0:00:11	0:00:16	0:03:01	0:00:18	0:00:04	0:00:03	0:00:02	0:00:27	0:00:10	0:00:08	0:00:02	0:00:02
6	*0:01:02*	*0:01:58*	*0:00:16*	*0:00:28*	*0:03:44*	*0:00:45*	*0:00:16*	*0:00:11*	*0:00:07*	*0:01:19*	*0:00:25*	*0:00:39*	*0:00:08*	*0:00:12*
7	0:00:51	0:01:24	0:00:09	0:00:12	0:02:36	0:00:00	0:00:01	0:00:00	0:00:00	0:00:01	0:00:00	0:00:01	0:00:00	0:00:00
8	0:00:00	0:00:01	0:00:00	0:00:00	0:00:01	0:00:00	0:00:01	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00
9	0:00:52	0:01:27	0:00:10	0:00:12	0:02:41	0:00:02	0:00:01	0:00:00	0:00:00	0:00:03	0:00:01	0:00:01	0:00:00	0:00:00
10	0:00:52	0:01:25	0:00:11	0:00:13	0:02:41	0:00:05	0:00:01	0:00:01	0:00:01	0:00:08	0:00:01	0:00:03	0:00:01	0:00:01
11	0:00:52	0:01:22	0:00:10	0:00:12	0:02:36	0:00:02	0:00:01	0:00:00	0:00:00	0:00:03	0:00:01	0:00:01	0:00:00	0:00:00
12	0:00:50	0:01:27	0:00:11	0:00:12	0:02:40	0:00:02	0:00:01	0:00:00	0:00:00	0:00:03	0:00:01	0:00:01	0:00:00	0:00:00
13	0:00:52	0:01:25	0:00:11	0:00:13	0:02:41	0:00:02	0:00:01	0:00:01	0:00:01	0:00:05	0:00:01	0:00:01	0:00:00	0:00:00
14	0:00:51	0:01:27	0:00:11	0:00:13	0:02:42	0:00:04	0:00:02	0:00:01	0:00:01	0:00:08	0:00:00	0:00:02	0:00:00	0:00:00
15	0:00:55	0:01:28	0:00:14	0:00:17	0:02:54	0:00:19	0:00:06	0:00:04	0:00:04	0:00:33	0:00:05	0:00:07	0:00:01	0:00:02
16	0:01:00	0:01:32	0:00:14	0:00:19	0:03:05	0:00:13	0:00:07	0:00:06	0:00:05	0:00:31	0:00:02	0:00:05	0:00:01	0:00:01
Atot	**0:13:17**	**0:22:28**	**0:02:49**	**0:03:56**	**0:42:30**	**0:02:02**	**0:00:46**	**0:00:27**	**0:00:22**	**0:03:37**	**0:00:55**	**0:01:16**	**0:00:14**	**0:00:20**
17	0:01:03	0:01:39	0:00:20	0:00:26	0:03:28	0:00:03	0:00:02	0:00:02	0:00:02	0:00:09	0:00:03	0:00:01	0:00:01	0:00:01
18	0:01:13	0:01:43	0:00:28	0:00:34	0:03:58	0:00:03	0:00:02	0:00:02	0:00:02	0:00:09	0:00:02	0:00:03	0:00:01	0:00:02
19	0:06:27	0:07:14	0:05:23	0:06:12	0:25:16	0:00:26	0:00:18	0:00:17	0:00:17	0:01:18	0:00:29	0:00:30	0:00:23	0:00:25
20	0:01:27	0:01:58	0:00:45	0:00:48	0:04:58	0:00:06	0:00:04	0:00:04	0:00:04	0:00:18	0:00:04	0:00:04	0:00:03	0:00:03
21	0:07:10	0:08:22	0:06:27	0:07:20	0:29:19	0:00:34	0:00:30	0:00:31	0:00:30	0:02:05	0:00:32	0:00:31	0:00:26	0:00:27
22	*0:07:06*	*0:07:44*	*0:05:58*	*0:06:46*	*0:27:34*	*0:00:15*	*0:00:11*	*0:00:11*	*0:00:10*	*0:00:47*	*0:02:25*	*0:02:24*	*0:02:22*	*0:02:22*
23	0:05:47	0:06:27	0:04:59	0:05:03	0:22:16	0:00:14	0:00:11	0:00:11	0:00:10	0:00:46	0:00:25	0:00:25	0:00:21	0:00:23
24	0:15:25	0:16:27	0:14:16	0:15:39	1:01:47	0:01:14	0:01:05	0:01:06	0:01:00	0:04:25	0:00:58	0:00:57	0:00:48	0:00:49
25	0:16:21	0:18:02	0:15:18	0:16:16	1:05:57	0:01:02	0:00:48	0:00:49	0:00:46	0:03:25	0:01:05	0:01:06	0:00:56	0:01:00
26	*0:07:44*	*0:08:35*	*0:06:54*	*0:07:01*	*0:30:14*	*0:00:58*	*0:00:49*	*0:00:49*	*0:00:47*	*0:03:23*	*0:02:25*	*0:02:28*	*0:02:20*	*0:02:24*
27	*0:08:38*	*0:09:29*	*0:07:59*	*0:08:53*	*0:34:59*	*0:00:37*	*0:00:30*	*0:00:30*	*0:00:29*	*0:02:06*	*0:02:24*	*0:02:25*	*0:02:19*	*0:02:24*
Ltot	**1:18:21**	**1:27:40**	**1:08:47**	**1:14:58**	**5:09:46**	**0:05:32**	**0:04:30**	**0:04:32**	**0:04:17**	**0:18:51**	**0:10:52**	**0:10:54**	**0:10:00**	**0:10:20**
Tot	**1:31:38**	**1:50:08**	**1:11:36**	**1:18:54**	**5:52:16**	**0:07:34**	**0:05:16**	**0:04:59**	**0:04:39**	**0:22:28**	**0:11:47**	**0:12:10**	**0:10:14**	**0:10:40**

Table 20: Geom-admin combination queries Oracle and Ingres

Oracle software version: production
No special clustering of LKI data

Note: join operation has no timings for separate queries

	sym/loc distance	sym/loc buffer	sym/loc join	sym/loc clip	sym/loc nn1000	bnd/shp distance	bnd/shp buffer	bnd/shp join	bnd/shp clip	bnd/shp nn1000
1	0:00:03	0:00:01		0:00:00	0:00:13	0:00:02	0:00:07		0:00:13	0:00:01
2	0:00:01	0:00:00		0:00:00	0:00:02	0:00:02	0:00:01		0:00:07	0:00:03
3	0:00:01	0:00:00		0:00:04	0:00:15	0:00:06	0:00:06		0:00:51	0:00:03
4	0:00:01	0:00:01		0:00:10	0:00:08	0:00:02	0:00:01		0:00:20	0:00:05
5	0:00:08	0:00:06		0:02:12	0:00:10	0:00:26	0:00:26		0:07:28	0:00:04
6	*0:00:14*	*0:00:12*		*0:04:56*	*0:00:07*	*0:03:03*	*0:03:04*		*2:36:27*	*0:00:03*
7	0:00:00	0:00:00		0:00:00	0:00:16	0:00:00	0:00:00		0:00:07	0:00:04
8	0:00:00	0:00:00		0:00:00	0:00:15	0:00:00	0:00:00		0:00:00	0:00:01
9	0:00:00	0:00:00		0:00:03	0:00:14	0:00:01	0:00:01		0:00:13	0:00:02
10	0:00:00	0:00:00		0:00:04	0:00:12	0:00:03	0:00:04		0:00:45	0:00:07
11	0:00:00	0:00:00		0:00:01	0:00:13	0:00:01	0:00:02		0:00:11	0:00:03
12	0:00:00	0:00:00		0:00:01	0:00:14	0:00:01	0:00:02		0:00:09	0:00:03
13	0:00:00	0:00:00		0:00:01	0:00:15	0:00:03	0:00:04		0:00:16	0:00:07
14	0:00:00	0:00:01		0:00:07	0:00:15	0:00:02	0:00:03		0:00:42	0:00:05
15	0:00:00	0:00:00		0:00:00	0:00:07	0:00:10	0:00:14		0:03:42	0:00:03
16	0:00:00	0:00:00		0:00:00	0:00:05	0:00:17	0:00:24		0:09:32	0:00:06
Atot	**0:00:28**	**0:00:21**		**0:07:39**	**0:03:01**	**0:04:19**	**0:04:39**		**3:01:03**	**0:01:00**
17	0:00:05	0:00:06		0:00:05	0:00:35	0:00:28	0:00:44		0:00:57	0:00:49
18	0:00:05	0:00:06		0:00:05	0:00:24	0:00:52	0:01:20		0:01:07	0:01:25
19	0:01:55	0:03:03		0:02:36	0:02:16	0:14:25	0:25:28		1:34:03	0:13:31
20	0:00:04	0:00:07		0:00:06	0:00:17	0:01:26	0:02:46		0:02:41	0:01:42
21	0:02:31	0:04:03		0:03:19	0:02:55	0:17:54	0:32:53		1:34:26	0:18:09
22	0:02:19	0:04:08		0:03:12	0:02:55	0:16:26	0:35:04		1:32:37	0:17:49
23	0:01:32	0:02:42		0:02:10	0:02:28	0:11:43	0:24:20		0:20:29	0:15:00
24	0:02:41	0:04:11		0:03:32	0:03:29	0:50:13	1:14:46		7:34:34	0:19:43
25	0:02:57	0:05:08		0:04:06	0:03:28	0:52:07	1:25:03		1:33:58	0:16:43
26	0:02:45	0:04:44		0:03:57	0:03:38	0:21:00	0:39:28		2:56:47	0:07:31
27	0:02:16	0:03:43		0:03:05	0:02:44	0:24:16	0:42:22		3:13:52	0:10:24
Ltot	**0:19:10**	**0:32:01**		**0:26:13**	**0:25:09**	**3:30:50**	**6:04:14**		**20:25:31**	**2:02:46**
Tot	**0:19:38**	**0:32:22**	**0:48:55**	**0:33:52**	**0:28:10**	**3:35:09**	**6:08:53**	**20:23:21**	**23:26:34**	**2:03:46**

Table 21: Oracle distance, buffer, join, clip, nearest neighbor queries (production)

Oracle software version: patch 0115 Note: join operation has no timings for separate queries
No special clustering of LKI data

	sym/loc distance	sym/loc buffer	sym/loc join	sym/loc clip	sym/loc nn1000	bnd/shp distance	bnd/shp buffer	bnd/shp join	bnd/shp clip	bnd/shp nn1000
1	0:00:00	0:00:01		0:00:00	0:00:13	0:00:03	0:00:04		0:00:14	0:00:07
2	0:00:00	0:00:00		0:00:00	0:00:02	0:00:02	0:00:01		0:00:07	0:00:03
3	0:00:00	0:00:00		0:00:04	0:00:16	0:00:08	0:00:06		0:00:52	0:00:03
4	0:00:01	0:00:01		0:00:10	0:00:08	0:00:02	0:00:01		0:00:21	0:00:04
5	0:00:06	0:00:06		0:02:12	0:00:10	0:00:37	0:00:26		0:07:34	0:00:05
6	*0:00:13*	*0:00:12*		*0:04:53*	*0:00:08*	*0:04:21*	*0:03:07*		*2:43:05*	*0:00:04* *large area*
7	0:00:00	0:00:00		0:00:00	0:00:16	0:00:00	0:00:00		0:00:14	0:00:04
8	0:00:00	0:00:00		0:00:00	0:00:15	0:00:00	0:00:00		0:00:00	0:00:01
9	0:00:00	0:00:00		0:00:03	0:00:14	0:00:01	0:00:01		0:00:13	0:00:03
10	0:00:00	0:00:00		0:00:04	0:00:12	0:00:03	0:00:04		0:00:46	0:00:08
11	0:00:00	0:00:00		0:00:01	0:00:13	0:00:01	0:00:02		0:00:11	0:00:03
12	0:00:00	0:00:00		0:00:01	0:00:14	0:00:01	0:00:02		0:00:09	0:00:03
13	0:00:00	0:00:00		0:00:01	0:00:15	0:00:02	0:00:04		0:00:16	0:00:07
14	0:00:00	0:00:01		0:00:07	0:00:15	0:00:02	0:00:03		0:00:43	0:00:06
15	0:00:00	0:00:00		0:00:00	0:00:07	0:00:09	0:00:15		0:03:29	0:00:03
16	0:00:00	0:00:00		0:00:00	0:00:05	0:00:14	0:00:24		0:08:06	0:00:06
Atot	**0:00:20**	**0:00:21**		**0:07:36**	**0:03:03**	**0:05:46**	**0:04:40**		**3:06:20**	**0:01:10**
17	0:00:04	0:00:06		0:00:01	0:00:35	0:00:26	0:00:44		0:00:12	0:00:49
18	0:00:04	0:00:06		0:00:01	0:00:24	0:00:50	0:01:19		0:00:10	0:01:31
19	0:01:58	0:03:04		0:00:07	0:02:17	0:14:24	0:25:26		0:02:44	0:15:04
20	0:00:04	0:00:07		0:00:01	0:00:17	0:01:28	0:02:49		0:00:32	0:01:47
21	0:02:31	0:04:06		0:00:11	0:03:00	0:17:54	0:32:43		0:02:31	0:18:33
22	0:02:24	0:04:13		0:00:06	0:02:59	0:16:25	0:35:50		0:01:36	0:17:50
23	0:01:36	0:02:46		0:00:07	0:02:30	0:11:36	0:24:52		0:00:58	0:13:32
24	0:02:41	0:04:16		0:00:21	0:03:31	0:50:28	1:16:49		0:04:54	0:19:40
25	0:03:03	0:05:11		0:00:19	0:03:31	0:52:19	1:27:03		0:04:02	0:16:17
26	0:02:52	0:04:50		0:00:28	0:03:40	0:21:01	0:40:37		0:11:25	0:07:29
27	0:02:20	0:03:47		0:00:16	0:02:44	0:24:22	0:43:54		0:05:13	0:10:23
Ltot	**0:19:37**	**0:32:32**		**0:01:58**	**0:25:28**	**3:31:13**	**6:12:06**		**0:34:17**	**2:02:55**
Tot	**0:19:57**	**0:32:53**	**0:02:16**	**0:09:34**	**0:28:31**	**3:36:59**	**6:16:46**	**0:13:34**	**3:40:37**	**2:04:05**

Table 22: Oracle distance, buffer, join, clip, nearest neighbor queries (patch)

Oracle software version: production
Oracle: index-organization of mo_object (x_akr_objectnummer)
and mo_subject (postcode, subject_id)

	Oracle cold	Ingres cold	Oracle hot	Ingres hot	# parcels
1	0:00:02	0:00:01	0:00:01	0:00:00	1
2	0:00:01	0:00:00	0:00:01	0:00:00	1
3	0:00:01	0:00:00	0:00:01	0:00:00	1
4	0:00:01	0:00:01	0:00:01	0:00:00	1
5	0:00:01	0:00:00	0:00:01	0:00:00	10
6	0:00:01	0:00:01	0:00:01	0:00:00	10
7	0:00:01	0:00:00	0:00:01	0:00:00	10
8	0:00:01	0:00:01	0:00:01	0:00:01	9
9	0:00:01	0:00:00	0:00:01	0:00:00	105
10	0:00:01	0:00:01	0:00:01	0:00:00	7
11	0:00:02	0:00:02	0:00:01	0:00:00	77
12	0:00:01	0:00:01	0:00:01	0:00:01	96
13	0:00:02	0:00:02	0:00:01	0:00:00	498
14	0:00:02	0:00:02	0:00:01	0:00:01	53
15	0:00:03	0:00:04	0:00:01	0:00:01	180
16	0:00:01	0:00:01	0:00:01	0:00:00	4
17	0:00:19	0:00:17	0:00:03	0:00:07	5132
18	0:00:26	0:00:28	0:00:03	0:00:08	6977
19	0:00:10	0:00:18	0:00:03	0:00:14	5757
20	0:00:03	0:00:06	0:00:01	0:00:03	180
Tot	**0:01:20**	**0:01:26**	**0:00:26**	**0:00:36**	**19109**

Table 23: Administrative entrance queries Oracle and Ingres