Efficient Access to a Very Large Spatial Database

Peter J.M. van Oosterom and Christiaan H.J. Lemmen
Cadastre Netherlands, Company Staff,
P.O. Box 9046, 7300 GH Apeldoorn, The Netherlands.
{P.J.M.vanOosterom|C.H.J.Lemmen}@ap.kadaster.nl.{net}

Summary: *This paper describes the results of the benchmark of a very large database performed during the months July/August 1995 with the test data set 'Zeeland'. Some important aspects of the tests are using: OME/SOL (Object Management Extension/ Spatial Object Library), SLC (Spatial Location Code), and data compression. The results have been compared to the current situation of our main database in Zeeland. In general it can be concluded that the benchmarks requirements are met; on the average the check-out times are 10 times faster compared to the current main database partly due to faster hardware, check-in times are speed-up even more. The SLC technique has proven to be a good alternative for spatial indexing within the database, as representative spatial queries are speed up between 100 and 200 times. The SQL (Structured Query Language) queries show the flexibility of a relational database; especially valuable are the SQL queries using spatial operators of OME/SOL.*

1 Introduction

Our very large spatial database contains both the Cadastral map and the large scale topographic map of the Netherlands. After more than 10 years, this data set is now digital for, by far, the largest parts; about 20 Gbyte data. The increasing user requirements and uses are the main motivations for a technical renewal: more and more users have taken a subscription for periodic updates, users expect efficient and flexible geographic data browsers, 'Clearinghouse' with specific requirements, etc.

Therefore, a more flexible software architecture is needed. The basis is an extended (object) relational database; CA-OpenIngres [2]. An important aspect is the use of the OME/SOL (Object Management Extension/ Spatial Object Library) with spatial data types such as point, box, line, and polygon. In addition to the use of these data types, some other important new capabilities in the data model are storing topology and historic information. Furthermore, unique object identifiers have been introduced for

all geographic objects, e.g., boundaries, houses, symbols. The data model is summarized in Section 2. The GIS frontend and spatial data edit program is X-Fingis [10], which uses a check-out/check-in mechanism to deal with the long transactions. Though the data is maintained by X-Fingis, other GIS frontends, e.g. GEO++ [17, 18] can also access the data in the RDBMS (Relational Data Base Management System).

Section 3 gives the sizes of the tables and indices, and the compressed table sizes of the test data set 'Zeeland'. A benchmark has been executed in order to determine whether all this functionality can be obtained with good performance. It is impossible to obtain the required interactive responses without a spatial index. The SLC-technique (Spatial Location Code) [16] has been designed for efficiently supporting spatial clustering and range queries; see Section 4.

The hardware of test environment is described in Section 5. The actual performance measurements are given in Sections 6 (check-out/check-in) and 7 (Structured Query Language, SQL queries). Finally, the conclusions can be found in Section 8.

2 Overview of the data model

This section presents an overview of the data model which has been described in [11] in more detail. It is based on using OME/SOL spatial data types such as point, line, and box. The most important tables are xfio_line (a.o. boundaries), xfio_texpgn (a.o. parcels), xfio_sympnt (cartographic symbols), and xfio_gcpnt (geodetic control points). The spatial extend in the tables xfio_line and xfio_texpgn is indicated with a minimal bounding rectangle of type `box`. There is no need for a type `polygon`, because the area features (parcels) are stored topologically in xfio_texpgn using the CHAIN-method: the edges in xfio_line table contain references to other edges according to the 'winged edge structure' [3], which are used to form the complete boundary chains. The text/label location in the parcel table (xfio_texpgn) is represented with the type `point`.

The following attributes are included in the data model for all spatial features: *id* unique feature id, *sel_code* indicates to which map type a geographic object belongs, *source* of data, *quality* data accuracy, method of measurements, *vis_code* visibility code used in photogrammetic data collection, and *akr_area* official area; only for xfio_texpgn. A part of the data model:

```
create table xfio_texpgn(  // parcel
       pid        integer4,  // Parcel identifier
       tmin       integer4   // insertion time
       tmax       integer4   // deletion time
```

```
        slc         integer4   // spatial location code
        parea       float,     // Area of related polygon
        bid(s)      intlist,   // Boundary Ids (CHAIN)
        bbox        box,       // Bounding box of polygon
        text        char(80),  // Text
        sel_code    char(6),   // Belongs to map: cad/topo
        source      char(5),   // Source of data
        quality     char(1),   // Data quality: method/acc.
        vis_code    char(1),   // Visibility code
        akr_area    integer4,  // Official AKR area parcel
);
create table xfio_line(// line object
        bid         integer4,  // boundary identifier
        tmin        integer4   // insertion time
        tmax        integer4   // deletion time
        slc         integer4   // spatial location code
        beg_l_bid   integer4,  // Line Id left, begin pnt
        beg_r_bid   integer4,  // Line Id right, begin pnt
        end_l_bid   integer4,  // Line Id left, end pnt
        end_r_bid   integer4,  // Line Id right, end pnt
        l_pid       integer4,  // Parcel Id left side
        r_pid       integer4,  // Parcel Id right side
        bbox        box,       // Bounding box
        sel_code    char(6),   // Belongs to map: cad/topo
        source      char(5),   // Source of data
        quality     char(1),   // Data quality: method/acc.
        vis_code    char(1),   // Visibility code
);
```

Recently, quite a lot of attention has been paid to methods of storing and manipulation spatial-temporal data; e.g. in [1, 6, 13, 19]. Though very complex solutions have been described, our solution is based on a simple extension with 2 attributes per record: tmin and tmax. When a new entity is inserted, it gets the current time as value for tmin during check-in, and tmax remains unset; gets a special value. When an attribute of an existing entity changes, it is not updated, but the complete record is copied with the new attribute value. The old version gets current check-in time for its tmax value and the new version (record) gets this time value for tmin. Using this technique it is possible to retrieve the the data from any given time in history including correct topology references and it is also possible to produce the changes a given period efficiently.

3 The test data set

The complete province of 'Zeeland' is used as a test data set, including both the cadastral map and the large scale topographic map, containing about

Table 1: Tables in the test database

table (btree)	record-size bytes	#objects	table-size uncompr	table-size compr	secondary index
xfio_line	716	1398014	1440 Mb	673 Mb	47 Mb
xfio_texpgn	955	356895	367 Mb	173 Mb	12 Mb
xfio_gcpnt	140	113398	22 Mb	-	4 Mb
xfio_sympnt	74	218273	27 Mb	-	7 Mb

2.000.000 records of which most have a variable length attribute; e.g. a line with upto 35 points. The total database size without compression is about 1.9 Gb. When the tables xfio_line and xfio_texpgn are compressed, then the overall database size is about 1.0 Gb. Note that the size of the current network (production) database is about 420 Mb. More details about the relational table and index sizes in the database can be found in Table 1.

4 Spatial location code

In this section a short description of the Spatial Location Code (SLC) technique is given, more details can be found in [16]. The SLC technique is used for indexing and clustering geographic objects in a database. It combines the strong aspects of several known spatial access methods (Quadtree [14, 15], Morton code [12]; Fig. 1, and Field-tree [7, 9, 8]; Fig. 2)) into one SLC value per object.

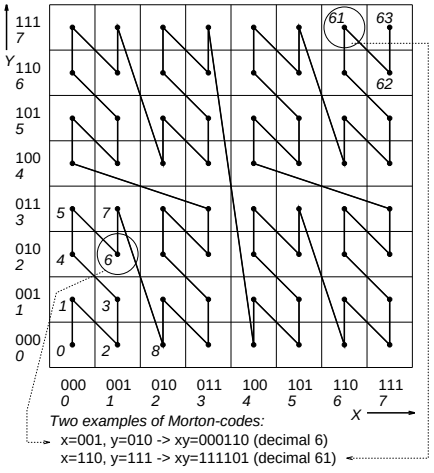

Fig. 1: The Morton-codes (shown on a 8*8 grid)

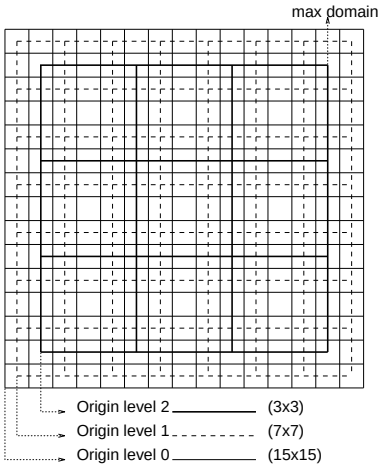

Fig. 2: The Field-tree (shown with 3 levels)

The *unique* aspect of the SLC is that both location and extent of possibly non-zero-sized objects are approximated by this *single* value. These SLC values can then be used in combination with traditional access methods, such as the b-tree [5], available in every database. The typical query response time for spatial objects is reduced by orders of magnitude for a very large spatial data set. In this paper the SLC is used in two-dimensional space, but the SLC is quite general and can be applied in higher dimensions. In the practice the SLC technique is used in the following manner:

- Add one 'spatial location code' SLC attribute to every table in the database which has a spatial (point, line, polygon, or box) attribute. The SLC is a one-dimensional code and every object gets exactly one code. It is possible that two different objects are approximated by the same SLC, but these objects will have about the same size and location.
- Modify the table structure to b-tree according to the SLC attribute, that is, the objects are more or less stored on disk in the order defined by the b-tree. This primary index is responsible for the spatial clustering.
- Define two functions:
  1. `Compute_SLC`: computes the SLC of a two-dimensional box. First, the bounding box (bbox) of an object is computed, then the SLC for this box is computed. So, only one `Compute_SLC` is needed to compute a correct SCL value for all spatial objects types with bbox stored in the database;
  2. `Overlap_SLC`: determines the ranges of SLCs which overlap the given two-dimensional search rectangle (query box). This function is needed to translate the query box to a set of corresponding ranges in the where-clause of a spatial SQL query.

There are 2 important 'tuning parameters' for the SQL technique: 1. how many grid levels are used in the Field-tree (e.g. 5 or 7 grids of different resolution), and 2. how many ranges may be used at most in the where-clause of a query. Note: less SLC ranges in a where-clause means that more overhead data will be retrieved.

5 Test environment

The tests are executed on a DEC Alpha 2100 4-275 (19Gb disk and 512Mb main memory) with OpenVMS 6.1. The tests are described in the test cases document [4]. The main tests are check-out and check-in of 33 typical work rectangles with predefined time limits. Further, a set of 'ad hoc' queries executed within the CA-OpenIngres `sql` tool. The tests are started from

command-files and the reported times are *elapsed* seconds on the DEC Alpha with no other users.

In order to be able to compare the benchmark figures, also some measurements with the current system LKI 3.21, based on the network database VAX DBMS 6.2A[1]), are performed. The platform is a DEC VAX 4100 (128 Mb main memory) under VAX/VMS 5.5. The 'overall' performance of the VAX 4100 is about 50% of the Alpha 2100. The benchmarks are performed in a situation whit no other users on the system.

6 Performance measurements

This section contains a summary of the results of the performance measurements. As indicated before all timings are *elapsed seconds* on a machine with no other users. Besides the measured and required check-out times, also the amount of data involved is recorded (per feature type). The number and type of modifications (insert, delete, update) during check-in are also recorded. The type of selection (check-out) is classified by data density and size of the area. Three types of data density are distinguished: High (cities) Middle (villages) and Low (country side).

The results of the performance measurements are shown in Figure 3. The column labeled with 'OME5' represents the database with a 5 level SLC. The database is also tested with a 7 level SLC and using the new, more efficient, ASCII to binary conversion in check-out (X-Fingis); see column 'OME7'. The required check-out times are represented by the column 'required'. In order to compare the new results to the current situation LKI 3.21, the measurements performed in Zeeland are given in column 'LKI3.21'. It can be observed that the required time limits are obtained with the 'OME7' solution and that this is about 10 times faster than the current situation. The check-in times are much faster than the required times: 3 to 10 times (not in the figure). The explanation is that only new, changed, and deleted objects are registered in the database.

To get a feeling for the response time when the system has an heavy load, 35 interactive users where simulated during check-out with the tool UETP on the VAX/VMS environment UETP. The resulting check-out times are then 2 to 3 times slower, which is quite good. Another type of heavy load is simulated by selecting the 10 most difficult check-out regions, which are then all started within 30 seconds. So, a lot of parallel check-outs are created. In this case the resulting check-out times are about 6 to 7 times slower, which is still very acceptable.

In order to establish the effectiveness of the SLC value in practice, 7 repre-

[1] The network database VAX DBMS is now an Oracle product.

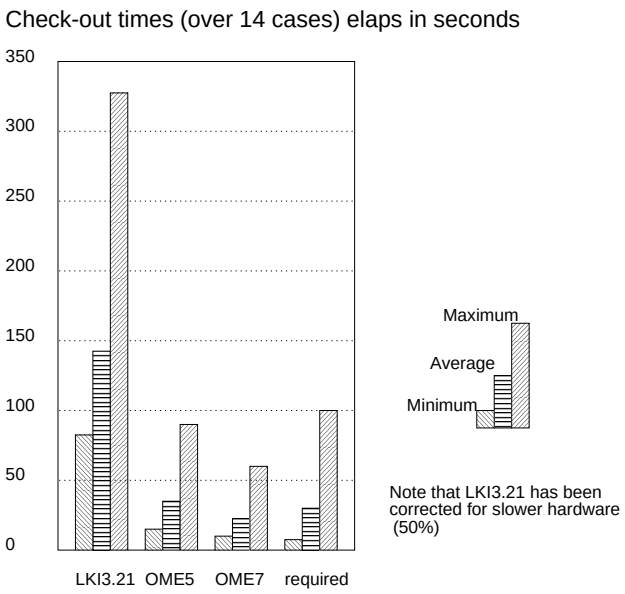

Fig. 3: Overview of check-out times

sentative check-out regions are used in SQL parcel boundary count queries on the xfio_line table. Three types of queries are created: 1. using both the bbox test and SLC value; 2. using only bbox test; and 3. using only SLC value.

From Table 2 it can be observed that using the SLC value responses are between 100 and 200 times faster. Note that the representative queries in the large xfio_line table (about 1,400,000 records) are quite restrictive (about 0.1% of the data) and only count data. If data is also transferred and/or the queries are less restrictive, the gain will be less.

Looking at the ratio between the number of counted objects based on the SLC value only and the number of counted objects based on bbox is reasonably low. Indicating that the SLC value is quite restrictive and well tuned (7 levels, smallest grid 100 m, and 29 ranges in the query where-clause). The highest ratios are in the small (test 1) and large area (test 34). In case of the small area the 29 ranges are more than sufficient to exactly specify the required cell. However, these cells also contain some objects outside the requested region. In case of the large area, the 29 ranges are probably too few and too many unwanted cells have to be included.

Table 2: Results of some representative spatial selections (OME/SOL, compressed, 7 level SLC, max 29 ranges)

	SLC & bbox		bbox only		SLC only		
test	#obj	sec	#obj	sec	#obj	sec	ratio
1	165	1	165	496	602	1	3.66
7	750	3	750	474	1718	3	2.29
13	2080	3	2080	475	2992	4	1.44
22	169	1	169	474	445	2	2.63
30	4178	5	4178	476	4730	5	1.13
31	0	0	0	474	72	0	-
34	1423	6	1423	475	5346	6	3.76
average		2.7		477.7		3.0	

Table 3: Results of the 11 SQL queries (OME/SOL, compressed, 5 level SLC and max 29 ranges)

query#	table	#counted	no slc sec	with slc sec
1	xfio_texpgn	2938	273	22
2	xfio_line	9203	254	40
3	xfio_line	451	451	
4	xfio_gcpnt	83	6	2
5	xfio_gcpnt	32	6	2
6	xfio_gcpnt	0	21	
7	xfio_line	0	0	
8	xfio_texpgn	5008	273	11
9	xfio_line	5024	453	
10	xfio_gcpnt	45	6	1
11	xfio_line	28131	468	45

7 SQL queries

The 11 test SQL queries are also specified in the test cases document [4]. Note that the OME/SOL spatial operators, such as `distance`, `inside`, and `intersect` are used in these SQL queries. Some SQL queries will show the flexibility of the new extended relational database: e.g. test query 1. find all parcels larger than 10.000 ca. in a rectangular region

```
SELECT * FROM xfio_texpgn
WHERE oarea>10000 AND inside(bbox,'((12000,363000),(40000,382000))')=1
```

and test query 10. find all geodetic control points with quality code 'T1' within a given circle with radius of 2 km:

```
SELECT * FROM xfio_gcp /* Geodetic Control Point */
WHERE quality='T1' AND distance(location,'(45000,393000)') < 2000
```

Without using the SLC value, the DBMS has to perform sequential table scans. When the SLC is used, the btree primary index can be used and the queries are speed-up. The difference in response time can be seen in Table 3.

8 Conclusion

The specified benchmark requirements are met: check-out times are 30% faster on the average and check-in times are 3 to 10 times faster than required. Additional analyses showed that the SLC technique was an important factor in this success as it did speed-up a spatial range count query, for the representative selections, between 100 and 200 times. Compression saves about 50% of disk space. The SQL queries show the flexibility of a relational database; especially valuable are the SQL queries using spatial operators of OME/SOL. Besides the data model described in this paper using OME/SOL, we also implemented a data model without OME/SOL[2]. However, there was no significant difference in performance.

After this successful benchmark, it has been decided to implement the renewed very large spatial database. This means that we now get a fully integrated database with all data (geometry, topology, history, and attributes) in one relational database. Due to the open architecture, it is also possible to integrate this spatial database with other geographic data sets.

Acknowledgments

The benchmarks are performed by the following group of people: Tapio Keisteri, John Smedley, Gerard Wijmenga, Johan Smit, Louis de Bruijne, Chrit Lemmen, Marcel van de Lustgraaf, and Peter van Oosterom.

References

[1] Khaled K. Al-Taha, Richard T. Snodgrass, and Michael D. Soo. Bibliography on spatiotemporal databases. *International Journal of Geographical Information Systems*, 8(1):95–103, 1994.

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

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

[4] Cadastre. Testgevallen lki 3.26. Technical report, Kadaster, July 1995. Internal report in Dutch.

[2] The model without OME/SOL does not use spatial data types, but uses the standard data types for representing spatial data (integer, varchar). Of course, the database has no understanding of spatial data: queries with spatial operators are not possible and perhaps even more serious, it is possible to store illegal spatial data values.

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

[6] A. U. Frank. Qualitative temporal reasoning in GIS – ordered time scales. In *Proceedings of the 6th International Symposium on Spatial Data Handling, Edinburgh, Scotland*, pages 410–430, September 1994.

[7] André Frank. Storage methods for space related data: The Field-tree. Technical Report Bericht no. 71, Eidgenössische Technische Hochschule Zürich, June 1983.

[8] Andrew U. Frank and Renato Barrera. The Field-tree: A data structure for Geographic Information System. In *Symposium on the Design and Implementation of Large Spatial Databases, Santa Barbara, California*, pages 29–44, Berlin, July 1989. Springer-Verlag.

[9] Andreas Kleiner and Kurt E. Brassel. Hierarchical grid structures for static geographic data bases. In *Auto-Carto London*, pages 485–496, London, 1986. Auto Carto.

[10] KT-Datacenter Ltd., Riihimäki, Finland. X-Fingis Software V1.1, INGRES version. Technical report, October 1994.

[11] 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.

[12] Jack A. Orenstein and Frank A. Manola. PROBE spatial data modeling and query processing in an image database application. *IEEE Transactions on Software Engineering*, 14(5):611–629, May 1988.

[13] D. J. Peuquet and E. Wentz. An approach for time-based analysis of spatio-temporal data. In *Proceedings of the 6th International Symposium on Spatial Data Handling, Edinburgh, Scotland*, pages 489–504, September 1994.

[14] Hanan Samet. The quadtree and related hierarchical data structures. *Computing Surveys*, 16(2):187–260, June 1984.

[15] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, Mass., 1989.

[16] 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. Submitted.

[17] T. Vijlbrief and P. van Oosterom. GEO++ systeem: een uitbreidbaar GIS met consistentieregels. *Nederlands Geodetisch Tijdschrift Geodesia*, 35(6):274–281, September 1993.

[18] Tom Vijlbrief and Peter van Oosterom. The GEO++ system: An extensible GIS. In *Proceedings of the 5th International Symposium on Spatial Data Handling, Charleston, South Carolina*, pages 40–50, Columbus, OH, August 1992. International Geographical Union IGU.

[19] M. F. Worboys. Unifying the spatial and temporal components of geographical information. In *Proceedings of the 6th International Symposium on Spatial Data Handling, Edinburgh, Scotland*, pages 505–517, September 1994.