Testing current DBMS products with realistic spatial data

Wilko Quak and Peter van Oosterom and Theo Tijssen
Section GIS Technology, Department of Geodesy,
Faculty of Civil Engineering and Geosciences,
Delft University of Technology,
Thijsseweg 11, 2629 JA Delft, the Netherlands.
Email: {quak|oosterom|tijssen}@geo.tudelft.nl

Abstract. Support of spatial data types can be found in more and more commercial Database Management Systems (DBMSs). In our research we have tested three DBMSs with support for spatial data: Oracle, Informix and Ingres. Both the functionality, a predefined set of questions had to be translated into spatial SQL queries and the performance is evaluated. In order to test the performance a cadastral data set, including a boundary table with more than 10.000.000 variable length rows, has been used.

Introduction

More and more standard commercial DBMSs offer functionality to store and retrieve spatial data efficiently (ASK-OpenIngres 1994, Hebert & Murray 1999, IBM 2000, Inf 2000). Since these products are relatively new, there is not much known about their functionality and performance. This paper describes the experiments performed with these DBMSs. The basic research question is the suitability of current DBMS products for the storage and retrieval of large volumes of spatial data. The best known benchmark for spatial data, the Sequoia Benchmark (Stonebraker et al. 1993), does not answer our questions about the functionality of the DBMSs, because it contains only a few different queries. The functionality and performance of different systems are tested using cadastral data (supplied by the Dutch Cadastre) The Cadastre also stated a number of specific questions that were used as a basis for test queries on the data. This rest of the paper is structured as follows: In Section 2 a description of the test data and of the test configuration is given. Section 3 describes the loading of the data and the creation of index structures on the data. Section 4 describes the queries that were performed on the data and the results of different systems are compared. Finally in Section 5 we conclude.

	Loading Time	Database Size
Oracle	7h	7646Mb
Informix	12h	9272Mb
Ingres	35h	4234Mb

Table 1: Preliminary results of loading

Test configuration and test data

Test configuration

All experiments have been performed on a Sun Enterprise E3500 server at the GDMC (Geo-Database Management Center) in Delft. This computer has two 400 MHz UltraSPARC CPUs, 2 Gb of main memory, and 600Gb of disk space, on several RAID0 and RAID5 sets. Details can be found in (Tijssen et al. 2001). In the experiments the following DBMSs were tested: Oracle 8i, Ingres II and Informix Spatial 8.11. All of the databases were installed on the unix file system; no raw devices were used.

All DBMSs have an almost infinite amount of parameters that can be tuned for optimal performance. In the experiments the same amount of time has been spent on tuning the different systems.

Test data

In the experiments we used data from the Dutch Cadastre. The dataset consisted of the topographic- and administrative data of two provinces. Note: currently we are performing experiments with the data of all provinces; the final paper will describe these experiments.

The topographic data is stored in a topological model (van Oosterom 1997) and contains the geometry of almost 4 million parcels, which is stored in 10 million boundary lines. This database also contains historic data since 1997. This is implemented by adding a time stamp attribute to all spatial data.

The administrative data contains the ownership information of all current parcels.

Loading the data

This section describes the bulk-loading of the spatial data into the DBMS. The loading of the data consists of two parts; first the data is loaded into the system, then index structures are created on the data. The dataset consists of a set of base tables that contain the real data. On top of these tables many database views exist.

In the first stage, the loading of the data, we researched the time needed to load the data, and the size of the loaded data. In Table 1 the overall results of loading the data in the different systems is given. The poor performance of the Ingres database can be explained by the fact that the Ingres bulk loading tools cannot handle spatial data; this means that all spatial data needs to be inserted with separate 'insert' queries.

	Index Creation Time	Index Size
Oracle	12h 23m	2171 Mb
Informix	5h 59m	2523 Mb
Ingres	6h 9m	820 Mb

Table 2: Preliminary results on indexing the data

Creating spatial indices

After the data is loaded, index structures that accelerate query operations on the data are built. Different DBMSs use different structures to accelerate the queries. All tested DBMSs offer R-tree indexing (Guttman 1984) or variants on it. Oracle also provides a quad-tree like index (Samet 1990) and an index structure based on regular grids.

The creation of index structures is not yet standardised in SQL as can be seen from the statements that are needed for the different DBMSs below.

```
/* Oracle 8i */
create index xfiox_boundary_0 on xfio_boundary (bbox)
indextype is mdsys.spatial_index
parameters ('sdo_fanout=64 sdo_rtr_pctfree=10;
/* Informix */
create index xfiox_boundary_0 on xfio_boundary (bbox st_geometry_ops)
using rtree (NO_SORT='FALSE', FILL_FACTOR='90', BOUNDING_BOX_INDEX='NO');
/* Ingres */
create index xfiox_boundary_0
on xfio_boundary(bbox)
with structure=rtree,
range=((-25000000,275000000),(325000000,625000000));
```

Table 2 contains a short overview of the resources needed for the spatial indices on the data. Because of the different architectures of the DBMSs it is sometimes hard to give good comparison. In Table 2 the time for index creation and analysis are added up.

Spatial Clustering of the data

The way the spatial data is physically distributed on disk appears to be very important for response time of queries on this data. This means that the data on the disk is organised in such a way, that objects that are spatially close to each other are also close together on disk. In all DBMSs we tried to cluster the data on a spatial attribute. Depending on the DBMS there are three different ways of achieving this disk clustering, which are handled in the next three paragraphs:

Clustering using a spatial index Most system can use a primary index to cluster the data. The following Ingres command would put all parcels that are spatially close together in the same disk block:

```
modify parcels to rtree on shape;
```

However currently this option is not available in any of the databases for the rtree index. Only standard index structures (like btrees) can be used for clustering.

Clustering using a non-spatial attribute Some systems do not allow clustering on a spatial attribute, and some tables do not have a spatial attribute, but are strongly related to location. In this case it is useful to cluster on a non-spatial attribute that has a clustering effect in space. For example, the parcels table in the system can be clustered on ZIP-code with the following command:

```
modify parcels to btree on zipcode;
```

Clustering at Load Time If the DBMS does not offer support for the clustering of spatial data, in some systems the data can still be clustered by controlling the order in which the objects are loaded into the data base. Although it is not not documented, and not guaranteed by the system, most systems insert the objects on disk in the order in which they are loaded. So by ordering the data before it is loaded into the system a very good clustering can sometimes be achieved.

Querying the data

The Cadastre provided us with sample questions from cases they encountered regularly which they want to be solved by the database. These questions were translated into SQL queries, which were tested on all databases. All operations in the experiments focus on querying the data; this means update operations are not considered. Some of the queries in the testset are:

- Find and display all objects within a certain rectangular area of the map. Normally this very common query is solved in the database by just looking at the minimum bounding rectangles (MBR) of all the objects ignoring the actual shape of the objects.
- Find all objects that overlap with a given polygon. In this case the polygon can be irregular. This means that the DBMS has to perform an intersection test with all objects in the database. If the query polygon has many points, this operation can become very expensive.
- Find and clip all objects with a given search polygon. Now the objects have to be clipped as well.
- Find all objects that intersect with a (possibly very long) query polyline. Although the number of objects returned can be small, the standard filtering on MBR will not work, because the filter will return far too many objects. During the testing the environment was kept as stable as possible.

Basic windowing queries

Basic windowing queries are performed well by all DBMSs. This query selects all objects that overlap with a given rectangle. It is executed when you draw a map on the screen. We tested the DBMSs by selecting rectangles with different sizes (ranging from a few thousand square meters to an area of 45km x 45km). Some of these rectangles are in densely populated areas and some not. Table 3 shows the result of these queries. As a global result it can be concluded that the DBMSs are fast enough for interactive mapping.

extent	# polylines	oracle	ingres	informix
100 x 125 m	423	3s	0s	2s
200 x 250 m	271	1s	1s	0s
400 x 400 m	2118	2s	1s	0s
1000 x 1250 m	784	1s	1s	1s
5 x 6.25 km	17250	11s	6s	5s
10 x 12.5 km	142405	68s	62s	39s

Table 3: Select all objects within a rectangle

Database	Execution Time
Informix	2s
Oracle	18 m 33s
Ingres	2m 10s
Oracle (patch)	37s

Table 4: Select all objects overlapping a very long line

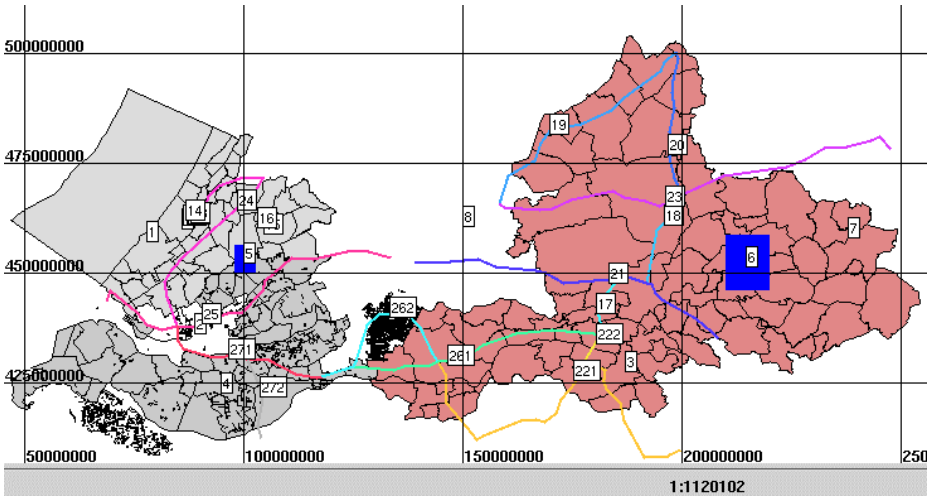

Figure 1: Overview of cadastral selection/query goemtries

Queries with long linear objects

One of the queries in the experiments that showed the biggest difference in response time for the different databases was the following: Find all parcels that overlap with a very long linear object (for example a power cable). The line (labelled '19' in Figure 1) intersects with 1133 parcels in the database. As can be seen from the test results in Table 4 the difference in response time between the different DBMSs is huge, especially Oracle was very slow. After reporting the results of this query to Oracle they responded with sending a patch. After installing the patch, performance for this query increased remendously. Apparently the different DBMSs have very different query plans for this query.

Conclusions

In this research we successfully loaded huge amounts of spatial (vector) data into 'off the shelf' DBMS products (Informix, Ingres, Oracle) without encountering any real problems. Loading spatial data is still more elaborate than loading alphanumeric data. This is because the loading tools are sometimes not able to work with the spatial types, or the creation of indices needs extra handwork.

During the experiments it was found that the way the data is clustered on disk is almost as important for response time as having the right index structures on the data. This remains true even if a table is so small that it easily fits in the core memory of the computer. All in all it seems that spatial DBMSs are growing up, but it will still take some time before they are fully mature.

In the future we plan to extend the experiments on the different DBMSs, we wish to look at the implementation of complex functionality (such as clipping or buffering) and spatial joins, because this is where the DBMSs differ most. Also the development of a benchmark for spatial DBMSs is considered.

References

ASK-OpenIngres (1994), INGRES/object management extention user's guide, release 6.5, Technical report, ASK-OpenIngres.

Guttman, A. (1984), 'R-trees: A dynamic index structure for spatial searching', *ACM SIGMOD* **1**3, 47–57.

Hebert, J. & Murray, C. (1999), *Oracle Spatial User's Guide and Referenc*e, Oracle Corp., Redwood City, CA, USA. Part No. A77132-01.

IBM (2000), *IBM DB2 Spatial Extender User's Guide and Referenc*e, special web release edn.

Inf (2000), *Informix Spatial DataBlade Module User's Guid*e. Part No. 000-6868.

Samet, H. (1990), *The Design and Analysis of Spatial Data Structure*s, Addison Wesley.

Stonebraker, M., Frew, J., Gardels, K. & Meredith, J. (1993), The Sequoia 2000 storage benchmark, *in* '19 ACM SIGMOD Conf. on the Management of Data, Washington, DC'.

Tijssen, D. T., drs. C.W. Quak & prof. dr.ir. P.J.M. van Oosterom (2001), Spatial DBMS testing with data from the Cadastre and TNO-NITG, Technical report, TU Delft, Faculty CiTG, Department of Geodesy. GISt No. 7.

van Oosterom, P. (1997), Maintaining consistent topology including historical data in a large spatial database, *in* 'Auto-Carto 13', pp. 327–336.