

A Performance Benchmark for Spatiotemporal Databases

Paul Werstein

Database Research Laboratory
Department of Computer Science
University of Otago
Dunedin, New Zealand
Email:werstein@cs.otago.ac.nz

*Presented at the 10th Colloquium of the Spatial Information Research Centre,
University of Otago, New Zealand, 16-19 November, 1998*

Abstract

In the course of developing a spatiotemporal database to serve as a research tool, it became apparent that existing benchmarks were inadequate for our needs. This paper describes the benchmark I created to fill the void. The benchmark is unique in two ways. First it requires significant temporal processing and storage capabilities. Second it has provisions for evaluating the ability of a spatiotemporal database to handle three dimensional data.

Keywords and phrases: spatiotemporal database, benchmark, performance.

1.0 Introduction

The University of Otago Department of Computer Science is developing a spatiotemporal database to serve as a testbed for several areas of research. This research includes a spatiotemporal query language, innovative ways of handling long transactions, and efficient operating system support for spatiotemporal databases. Our database is unique in providing three dimensional capabilities. We need a benchmark against which we can measure the query language developments and the research into operating system support.

Most recognised database benchmarks such as SPEC, BAPco, and TPC are not applicable to spatiotemporal data. There are some benchmarks aimed primarily at spatial databases such as the Sequoia 2000 benchmark (Stonebraker et al., 1993) and the Paradise Geo-Spatial DBMS benchmark (Patel et al., 1997). While good for spatial databases, they have a rather weak

temporal component and do not evaluate three dimensional capabilities.

Therefore I developed a benchmark that thoroughly tests the temporal and three dimensional capabilities as well as the spatial capabilities of a database. The benchmark expands on the Sequoia 2000 and Paradise benchmarks. It is described in detail in this article along with a justification for the queries.

In this paper, Section 2 gives a description of the standard benchmarks described in the literature. A complete review of these benchmarks is beyond the scope of this article; rather, a brief overview of the main issues and benchmarks is given. Then Section 3 reviews the Sequoia 2000 and Paradise benchmarks. Next Section 4 describes the benchmark developed for testing the University of Otago spatiotemporal database.

2.0 Recognised Database Benchmark Queries

Several groups have defined domain specific benchmarks that can be used to measure performance of computer systems (Gray, 1993). These include SPEC, BAPco, and TPC. SPEC (Standard Performance Evaluation Corporation) benchmarks are designed primarily to test performance in the scientific domain with emphasis on workstations and computational capability. The SPEC benchmarks require CPU intensive calculations, involving six integer programs and fourteen floating point programs. None of them involve large files or much disk I/O activity.



BAPco (Business Application Performance Consortium) benchmarks were created to determine the performance of personal computers, primarily running client-server applications in an office environment. The BAPco benchmarks require running a sample of word processors, spreadsheets, databases, email programs and graphics applications that are commonly found on office personal computers. The use of LAN based file and application servers are included in the benchmarks.

The SPEC and BAPco benchmarks are not applicable to large database environments. TPC (Transaction Processing Performance Council) developed a set of benchmarks for the evaluation of database systems. In the following subsections, I describe the TPC benchmarks as well as several other benchmarks designed to measure the performance of databases.

2.1 Transaction Processing Performance Council

The Transaction Processing Performance Council (TPC) is a consortium of companies interested in database benchmarking and had a membership of 41 in 1998 (TPC, 1998b). It was organised to bring some consistency to the performance results being advertised by vendors. At the time the TPC was formed, the reliability of vendor performance claims was dubious at best.

The TPC created an original set of two benchmark standards, called TPC-A and TPC-B. These standards were intended to set rules so that benchmark results could be considered to be reliable indicators of performance. The performance indicator in these benchmarks is the number of transactions per second (tps).

Both standards are intended to measure online transaction processing performance. TPC-A is meant to be a terminal emulation benchmark employing some "human think time" in the results. It includes the concept of response time. TPC-B permits a batch transaction generator to create incoming transactions. The response time of TPC-A is replaced by residence time in TPC-B as an indicator of how long a transaction is in the system before being completed. Both

TPC-A and TPC-B were oriented around a banking application.

TPC-C is a follow-on to TPC-A/B and is more complex, requiring multiple transaction types on a more extensive database. This benchmark is oriented to an order-entry application (Raab, 1993) and is composed of ten different transactions. TPC-C requires scaling the number of terminals and the size of the database to the processing power of the system being benchmarked.

TPC-D is oriented toward a decision support environment, requiring complex ad-hoc business queries on a large database. The queries access significant parts of the database, requiring multiple table joins, sorting, grouping and aggregation, and sequential scans (TPC, 1998a). The queries posed by decision support systems typically are long, complex, read-only queries. Updates are performed relatively infrequently.

2.2 The Wisconsin Benchmark

The Wisconsin benchmark was developed at the University of Wisconsin as a method of evaluating the performance of relational database systems. This effort arose from the work on the DIRECT database machine (De Witt, 1993). The benchmark was designed to test the performance of the major components of a relational database system. Its major shortcoming is the fact that it is a single user benchmark that does not require the use of concurrency controls.

This benchmark determines the performance of basic relational operations. Included are: selection with various criteria, projections, single and multiple joins, aggregate functions, and append, delete, and modify operations. The benchmark has also been used to evaluate parallel processor database systems.

2.3 Other Recognised Database Benchmarks

The ANSI SQL Standard Scalable and Portable Benchmark (AS³AP) is a benchmark containing a mixed workload of transactions, relational queries, and utility functions (Turbyfill, Orji and Britton, 1993). It arose from the desire to compare relational

databases on very different architectures with a variety of workloads.

It is divided into two sections. The first section is for single user tests and evaluates utilities for structuring and loading a database. It also tests access methods and query optimisation capabilities by performing selections, joins, projections, and updates. The second section tests the multiple user capabilities. It evaluates online transaction processing workloads, information retrieval, and mixed loads of short and long transactions, report generation, and table scans.

The SetQuery benchmark has more complex queries and report requirements than AS³AP (O'Neil, 1993). It is primarily designed to evaluate systems known mainly as decision support systems. These systems require queries that return multiple rows of a table at once whereas the TPC benchmarks deal with row-at-a-time queries.

Jensen et al. (1993) created a test suite for temporal database queries. The intention is to test the expressive powers of temporal data models and query languages. It is not intended as a performance benchmark.

Other benchmarks exist but are not covered here. The next section discusses some benchmarks that are explicitly designed to test the capabilities of spatial and spatiotemporal databases.

3.0 Review of Spatiotemporal Queries

Initially, testing of spatial database systems was based on specific applications. The results were not easily applied to other application domains. Marble (1986) produced one of the first attempts to define a benchmark for spatial databases. While this effort was helpful, it only applied to raster based spatial database systems with no temporal component.

Langran (1992) proposed a set of four spatiotemporal queries. They are:

- a simple temporal query (What is the state of a feature at a given time?)
- a temporal range query (What happens to a feature over a given time period?)

- a simple spatiotemporal query (What is the state of a region at a given time?)
- a spatiotemporal range query (What happens to a region over a period of time?)

Langran's queries were posed in the context of evaluating disk access requirements for various indexing and partitioning schemes. These queries provide a starting point but were used only in a theoretical analysis of performance. A more complete performance evaluation of spatial databases is required. I now discuss two recently proposed spatiotemporal database benchmarks which will set the stage for describing our proposed benchmark in section 4.

3.1 Sequoia 2000 Benchmark

Stonebraker et al. (1993) propose a set of 11 queries to evaluate the performance of the Sequoia 2000 project. This project is described in Stonebraker, Frew and Dozier (1993). Each of these queries is listed below along with some comments describing them. The letter, S, is used to denote that these queries came from the subject article.

Query S-1: Create and load the data base and build any necessary indexes.

The time for this query assumes the raw data exists on disk and does not include time to move data from tape if it originates on tape.

Queries S-2 through S-4 are raster queries. Query S-2 involves a temporal range query. Queries S-3 and S-4 have a fixed time.

Query S-2: Select AVHRR (Advanced Very High Resolution Radiometer) data for a given wavelength band and rectangular region, order by ascending time.

The benchmark AVHRR data contains 26 different time sequences. At the given resolution and rectangular clipping area, each image consists of 80K bytes. The time includes the time required to return the data to the application, but not the time required to display the data.



Query S-3: Select AVHRR data for a given time and geographic rectangle and calculate an arithmetic function of the five wavelength bands for each cell in the rectangle.

The benchmark AVHRR images contain 5 different wavelength bands. The suggested arithmetic function is a weighted average of cell values. The computation must be done at run-time; it is not permissible to compute and store the computed values during data load.

Earth scientists often run several types of weighted queries on a given study area. This test gives a measure on how well this activity is supported by the database.

Query S-4: Select AVHRR data for a given time, wavelength band, and geographic rectangle. Lower the resolution of the image by a factor of 64 to a cell size of 4 km x 4 km and store it as a new DBM object.

The resolution of the original AVHRR data set is 0.5 km x 0.5 km. This query simulates the abstracting of data to allow easier browsing of massive amounts of data. The abstracted data is stored for later retrieval.

Queries S-5 through S-7 are point and polygon queries.

Query S-5: Find the point record that has a specific name.

This query is a non-spatial retrieval of a specific object. An index is required on the name in order to perform the retrieval in a reasonable amount of time. All data associated with the object are returned.

Query S-6: Find all polygons that intersect a specific rectangle and store them in the DBMS.

This query is a spatial range query and requires a spatial index. It also requires the ability to create new database objects “on-the-fly” as in query S-4.

Query S-7: Find all polygons that are more than a specific size and within a specific circle.

This query combines spatial and non-spatial restrictions in the selection.

Queries S-8 and S-10 are spatial join queries.

Query S-8: Show the landuse/landcover in a 50 km quadrangle surrounding a given point.

In executing this query, a rectangle is constructed around the given point. This rectangle is joined with the polygons in the database to find the polygons that intersect the rectangle.

Query S-9: Find the raster data for a given landuse type in a study rectangle for a given wavelength band and time.

This query performs a join between raster and polygon data.

Query S-10: Find the names of all points within polygons of a specific vegetation type and create the result as a new DBMS object.

A join between point and polygon data is required to perform this query. New DBMS objects are created and stored.




Some types of database objects such as drainage objects form a data network (examples are rivers and streams). Retrieving the drainage object from a given point requires a restricted recursive retrieval. Query S-11 is such a query.

Query S-11: Find all segments of any waterway that are within 20 km downstream of a specific geographic point.

It should be noted that this query does not simply retrieve a sequence of ever larger objects such as stream to river to lake. For example, in the case of irrigation systems, a river may feed several irrigation intake points which in turn feed many irrigation channels. Thus the retrieved results may resemble a large tree structure.

3.2 Paradise Geo-Spatial DBMS Benchmark

Patel et al. (1997) present some techniques for parallelising a spatial database system. The project uses the Paradise object-relational database system.



While the Sequoia 2000 project alludes to large databases, only a regional database of about 1 GB of data was available at the time of the Patel article. A large database of 120 GB of global data was created by using NASA satellite data for the Paradise project. This large data set allows tests for scalability which was part of their goal.

The article also asserts that the spatial queries in the Sequoia 2000 benchmark are relatively simple. Therefore they created an alternative benchmark of fourteen queries. The first nine queries correspond to the first nine queries in the Sequoia 2000 benchmark. They are not repeated in this section. The remaining five queries are new. They will be preceded with the letter, P, and will be numbered beginning with 10.

Query P-10: Select all rasters whose average pixel value over a particular geographical region (polygon) is greater than a fixed constant.

This query requires the creation of the clipped region and calculation of a value over the clipped data. This clipping and calculation must be done for all rasters that cover the given region.

Query P-11: Find the closest road of each type to a given point.

This query requires the use of the spatial aggregate, closest. Spatial databases should be able to handle spatial aggregates as well as relational aggregates.

Query P-12: Find the closest drainage feature (i.e. lake, river) to every large city.

This query is designed to evaluate the declustering and parallel nature of the Paradise database. It requires the evaluation of a spatial aggregate on a cross product of two relations. The Paradise database divides the drainage into 10,000 tiles and distributes the tiles to the nodes in the system. The point data is similarly distributed. To answer the query, a spatial index on the declustered drainage shape attribute is built during the query. The index is local to each node and only covers the local data.

Query P-13: Find all drainage features (lakes, rivers, etc.) which cross a road.

This query requires a join on two large spatial relations.

Query P-14: Select all rasters for a particular year and channel and clip those rasters by all oil fields.

To answer this query a large number of rasters must be joined with a large number of polygons.

3.3 Critique of the Spatial Benchmarks

The Sequoia 2000 and Paradise Geo-Spatial DBMS benchmarks are adequate to evaluate the characteristics of a spatial database when used in the environment required for earth scientists. They do not adequately evaluate the queries required in environments such as utility work or roading. They are certainly deficient when used to evaluate a spatiotemporal database. The main deficiency is the failure to evaluate thoroughly the temporal capabilities. Only queries, S-2, S-3, S-4, and P-14, have a temporal component. Only S-2 requires the ability to handle temporal ranges.

In addition the only temporal data is a set of 26 AVHRR images evenly spaced throughout a year. The benchmark for Paradise expands this range to 10 years, but the data is still neatly spaced. This even spacing makes it easy to locate the proper files or data for a given date. The point, line and polygon data are current data only and have no historical characteristics.

In order to evaluate the temporal capabilities of a spatiotemporal database, we need data that has an arbitrarily distributed time component. The point, line, and polygon data also must have a temporal component in order to simulate the real world. For example, buildings are built and destroyed; roads are built, rerouted, and abandoned. The spatiotemporal database should support historical queries on these data as well as fixed interval images. Raster data should also be randomly distributed in time.

The next section describes the proposed benchmark. It expands the realm of benchmarks for spatiotemporal databases into 3 dimensions. Most current systems and research revolve around 2



dimensional data. Many application areas require full 3 dimensional capabilities. For example, pollution clouds are 3 dimensional, geological studies require depth as well as areal studies, and marine studies involve water depth as well as locational data.

4.0 Proposed 3-Dimensional Spatiotemporal Benchmark

The queries for the proposed benchmark have the following goals:

- Evaluate the full spatiotemporal capabilities of the database.
- Evaluate the 3 dimensional capabilities of the database.
- Evaluate the support provided by operating system enhancement research.

The last point may need additional explanation. The Database Research Laboratory is researching enhancements to the Linux operating system to better support spatiotemporal databases. The areas of active investigation include file systems, virtual memory, and process scheduling. We need a method to measure the effectiveness of changes to the operating system. This benchmark provides that metric as well as being able to evaluate spatiotemporal databases in general. It can also be used to evaluate 2D spatiotemporal database systems by omitting the 3D queries.

When the queries below match a Sequoia 2000 query or a Paradise query, the corresponding query is given in brackets beside the query number. For example, (S-2) means this query corresponds to the second query of the Sequoia 2000 benchmark.

Query 1: (S-1) Create and load the database and build indices.

The raw data for this query shall be on disk or other high speed media before starting the test so the data transfer rate of a slow device such as a tape is not the limiting factor.

Query 2: (S-2) Select AVHRR data for a given wavelength band, rectangular region, and time period, order by ascending time.

Query 3: Select 3D matrix data for a given 3D region and time period, ordered by ascending time.

Query 2 is a raster query involving a temporal range. Query 3 is the 3D equivalent. (Note: Matrix data is the 3D analogue of raster data.) The time period will be selected to return 10 data sets. For Query 2 each data set is about 80 kilobytes. For Query 3 each data set is about 500 kilobytes. This query tests the ability to locate data by date and to reconstruct original data from deltas.

Query 4: (S-3) Select AVHRR data for a given time and geographic rectangle and calculate an arithmetic function of the five wavelength bands for each cell in the rectangle.

Query 5: Select 3D matrix data for a given time and 3D region and calculate an arithmetic function for each cell in the matrix.

The arithmetic function is a weighted average of the selected cell with the surrounding cell values. The computation is done at run time.

Query 6: (S-4) Select AVHRR data for a given time, wavelength band, and geographic rectangle. Lower the resolution of the image by a factor of 64 to a cell size of 4 km x 4 km and store it as a new DBMS object.

Query 7: Select 3D matrix data for a given time and 3D region. Lower the resolution by a factor of 512 to a cell size of 4 km x 4 km x 4 km and store it as a new DBMS object.

This query simulates the abstracting of data to allow easier browsing. The abstracted data is stored for later retrieval.

Query 8: (S-5) Find the 2D point object that has a specific name.

Query 9: Find the 3D point object that has a specific name.

This query is a non-spatial retrieval of a specific object. An index is required on the name in order to



perform the retrieval in a realistic amount of time. All data associated with the object is returned.

Query 10: (S-6) Find all polygons that intersect a specific rectangle and store them in the DBMS.

This query is a spatial range query. It requires a spatial index. It also requires the ability to create new database objects “on-the-fly” as in Queries 6 and 7.

Query 11: Find all polyhedrons that intersect a specific region of space and store them in the DBMS.

This query is a 3D spatial range query. It requires a 3D spatial index. It also requires the ability to create new database objects “on-the-fly.”

Query 12: (S-7) Find all polygons that are more than a specific size and within a specific bounding circle.

Query 13: Find all polyhedrons that are more than a specific size and within a specific radius of a given point.

These two queries combine spatial and non-spatial restrictions in the selection.

Query 14: (S-8) Retrieve the landuse/landcover polygons that lie fully or partially within a square (50 km on a side) centered at a given point.

This query is a spatial join. In executing this query, a square is constructed around the given point. This square is joined with the polygons in the database to find any polygons that intersect the rectangle.

Query 15: Retrieve any landuse/landcover polygons that have intersected a square (50 km on a side) centered at a given point, including those polygons in the historical part of the database. In cases where the current polygon has a history, retrieve only the current polygon.

This query is similar to query 14, but requires a search through history to retrieve any polygons that have intersected the square but have been “deleted.”

This query tests the ability of the database to track “deleted” objects.

Query 16: (S-9) Find the raster data for a given landuse type (polygon) in a study rectangle for a given wavelength band and time.

This query performs a join between raster and polygon data.

Query 17: Find the 3D matrix data for a given polyhedron in a study box for a given wavelength band and time.

This query is the 3D equivalent of Query 16, requiring a join between matrix and polyhedron data.

Query 18: (S-10) Find the names of all points within polygons of a specific vegetation type and create the result as a new DBMS object.

A join between point and polygon data is required to perform this query. New DBMS objects are created and stored.

Query 19: Find the names of all points within polyhedrons of a specific type and create the result as a new DBMS object.

This query is the 3D equivalent of query 18. A join between point and polyhedron data is required to perform this query. New DBMS objects are created and stored.

Query 20: Find the names of all points within polygons of a specific vegetation type at a time in history.

A spatiotemporal join between point and polygon data is required to perform Query 20. This query also requires the use of historical data.

Query 21: Find the names of all points within polyhedrons of a specific type at a time in history.

This query is the 3D equivalent of query 20. A spatiotemporal join between point and polyhedron data is required to perform this query, along with the use of historical data.



Query 22: (S-11) Find all segments of any waterway that are within 20 km downstream of a specific geographic point.

Query 23: Find all segments of any waterway that are within 20 km downstream of a specific geographic point throughout time and order them by time.

This query is the spatiotemporal equivalent of Query 22. It requires a search through history.

Query 24: (P-10) Select all rasters whose average pixel value over a particular geographical region (polygon) is greater than a fixed constant.

Query 25: (P-11) Find the closest road of each type to a given point.

Query 26: (P-12) Find the closest drainage feature (i.e. lake, river) to every large city.

Query 27: (P-13) Find all drainage features (lakes, rivers, etc.) which cross a road.

Query 28: (P-14) Select all rasters for a particular year and channel and clip those rasters by all oil fields.

The queries below evaluate the ability of a spatiotemporal database to retrieve and to create historical data. A true spatiotemporal database does not merely update data. Rather, the updated data becomes the current data, and the previous current data becomes part of the database history.

Query 28: Update a given 2D raster image, recreating the previous image as history and store the changes in the database.

Query 30: Update a given 3D matrix, recreating the previous matrix as history and store the changes in the database.

Queries 29 and 30 update raster and matrix data.

Query 31: Update a polygon, recreating the previous polygon as history and store the changes in the database.

Query 32: Update a polyhedron, recreating the previous polyhedron as history and store the changes in the database.

Queries 31 and 32 update point, line, polygon, and polyhedron data.

Query 33: Retrieve all objects in a given area throughout the history of the area.

Query 34: Trace the changes to a road system throughout its history.

Query 35: Trace the changes to a polygon throughout its history.

Query 36: Trace the changes to a polyhedron throughout its history.

Queries 33-36 perform a “walk through time.” They require retrieving all versions of their objects from the database.

5.0 Conclusions

The benchmark presented in this paper was developed for testing the capabilities and performance of spatiotemporal databases. Two features set this benchmark apart from other benchmarks. First there is significant temporal processing required. Both the Sequoia 2000 and Paradise benchmarks have relatively simple temporal requirements. These benchmarks have a single temporal range query and three queries that fix time. The proposed benchmark adds complete “walks-through-time” for objects as well as temporal updates.

The second new feature of the proposed benchmark is the requirement for three dimensional support. Most of the two dimensional queries of the Sequoia 2000 and Paradise benchmarks have a three dimensional equivalent in the new benchmark. This extension recognises that spatial data need not be limited to two dimensions.

References

Abel, D. And Ooi, B. C. (eds) (1993) *Advances in Spatial Databases (Third International Symposium, SSD '93)*, Vol. 692 of *Lecture Notes*



- in *Computer Science*, Springer-Verlag, Berlin.
- DeWitt, D. J. (1993) *The Wisconsin Benchmark: Past, Present, and Future*, in Gray (1993), chapter 4, pp. 269-315.
- Gray, J. (ed.) (1993) *The Benchmark Handbook for Database and Transaction Processing Systems*, second edn, Morgan Kaufmann Publishers, San Mateo.
- Jensen, C. S. et al. (1993) *A Consensus Test Suite of Temporal Database Queries*, Aalborg University, Denmark.
- Langran, G. (1992) *Time in Geographic Information Systems*, Taylor & Francis, London.
- Marble, D. F. (1986) The development of standardized benchmarks for spatial database systems, *Proceedings of the Second International Symposium on Spatial Data Handling*, Seattle, Washington, USA, pp. 488-496.
- O'Neil, P. E. (1993) *The Set Query Benchmark*, in Gray (1993), chapter 6, pp. 359-395.
- Patel, J. et al., (1997) Building a scalable geo-spatial DBMS: technology, implementation, and evaluation, *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, USA, pp. 336-347.
- Raab, F. (1993) *Overview of the TPC Benchmark C: A Complex OLTP Benchmark*, in Gray (1993), chapter 3, pp. 131-267.
- Stonebraker, M., Frew, J. and Dozier, J. (1993) *The SEQUOIA 2000 Project*, Vol. 692 of Abel and Ooi (1993), pp. 397-412.
- Stonebraker, M. et al. (1993) The Sequoia 2000 storage benchmark, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC, USA, pp. 2-11.
- TPC (1998a) *TPC Benchmark™ D (Decision Support) Standard Specification*, Transaction Processing Performance Council.
- TPC (1998b) Transaction Processing Performance Council. url = <http://www.tpc.org/>, webmaster@tpc.com.
- Turbyfill, C., Orji, C. and Bitton, D. (1993) *AS³AP: An ANSI SQL standard scalable and portable benchmark for relational database systems*, in Gray (1993), chapter 5, pp. 317-357.

