18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

Testing geospatial database implementations for water data

R. Power

Commonwealth Scientific and Industrial Research Organisation
Email: robert.power@csiro.au

Abstract: The Bureau of Meteorology is currently assembling seamless hydrological and associated time series datasets of the Australian continent for the purposes of national water reporting and analysis. These datasets are to be managed as a data warehouse using a geospatial database providing access to numerous stakeholders. The choice of geospatial database platform is critical and will be assessed in terms of performance, functionality, and cost. This paper presents a testing methodology to compare different geospatial database products.

Evaluating the performance characteristics of databases has many motivations: identify performance issues with typical queries, compare alternative schema structures, benchmark price/performance trade-offs for different hardware/ operating system/ database platform/ database versions. The process of conducting a performance evaluation can focus on numerous factors: input/output (I/O) activity, memory usage, CPU time, and elapsed time can each be measured. The testing methodology presented is based on a configurable test harness framework that measures performance of database queries under varying experimental conditions.

The work performed to date has used datasets for a small representative region of Australia in order to verify the testing methodology. The real benefit of this work will be realised when continuous hydrological datasets for the entire Australian continent become available. These datasets are currently being prepared by GeoScience Australia on behalf of the Bureau as part of the Australian Hydrological Geospatial Fabric (AHGF) Project (Atkinson et al, 2008). The AHGF will deliver a consistent, authoritative, topologically connected set of spatial features representing the Australian hydrological system.

The AHGF Project will provide a tested solution to support the maintenance, extension and delivery of the AHGF datasets. The project will manage a pathway from the initial implementation of the AHGF using available small-scale data to the incorporation of more detailed data, complex network models, updates, and dissemination of simplified derived data products.

Central to the success of the AHGF vision will be the ability to establish a continental scale geospatial database supporting the requirements of the Bureau and other stakeholders. Selection of the database platform will require research attention to solve novel aspects of the problem domain and engineering activities to gather experience and evaluate existing tools. It is in this context that initial experiences of profiling the performance of two geospatial database products, MySQL with spatial extensions and PostgreSQL/PostGIS, are reported. These systems were compared using a small sample collection of water datasets and associated time series measurements from the Gwydir region in northern NSW. Preliminary results comparing the performance of these systems are presented for database queries anticipated to support water resources applications. Comparative results for Oracle Spatial have been conducted, but they cannot be reported due to licensing constraints.

These early tests using a small collection of water datasets aimed to establish a methodology for comparing geospatial database systems. This exercise has also provided a foundation for exploring the features and capabilities of these systems. This initial examination does not aim to establish which system performs best, but rather provides a preliminary investigation for using them. It is expected that different users will have a preference for different products, and so our task is also to explore the issues of interoperability to support data exchange. Future work will continue the evaluation of other databases with geospatial capabilities. For example, ESRI's Spatial Database Engine (ArcSDE) incorporated into ArcGIS will also be investigated, as will the spatial extensions for Microsoft's SQL Server.

Keywords: *GIS, Geospatial Database, water data*

1. INTRODUCTION

A Relational Database Management System (RDBMS), also referred to simply as a database, is a complex suite of software applications that collectively controls the organisation, storage, and retrieval of data. Modern database systems provide support for data types other than numbers and strings. The type system may be extended allowing new types and operations to be introduced. All the major database vendors support extensions to the traditional type system to include spatial data types and associated functions.

The term *geospatial database* describes an RDBMS that supports geographic information in the same way as any other data in the database. Vector data types such as points, lines, and polygons can be used and this data may make use of a Spatial Reference System (SRS) describing the coordinate system used. Raster, or gridded data, may also be supported in the form of an array structure where the cells contain information being modelled. Examples are satellite imagery or Digital Elevation Models (DEMs). A geospatial database may be used as the data repository for a Geographical Information System (GIS). The GIS supports an advanced user interface and incorporates specialised tools for the processing and presentation of spatial data.

It is the purpose of this work to focus on the functionality and performance of the underlying geospatial databases when using water datasets. A testing methodology has been established to profile the performance of different systems so they can be compared. The testing methodology has been exercised using a small test collection of data and queries. It will be used on 'real' datasets as the AHGF Project progresses.

The rest of the paper is organised as follows. Section 2 describes the computing environment, how the databases are compared, the data used, and the database structures. Section 3 presents the first set of profiling results, comparing MySQL and PostgreSQL using 20 database queries. Section 4 details how the databases respond when stress tested using 1000 queries that retrieve all records within a specified bounding box placed randomly over the dataset region. The paper concludes with a discussion and outlines future work.

2. EXPERIMENTAL METHODOLOGY

2.1. Computing Environment

The machine used was a Dell Optiplex 755 with Core2 Duo CPU, 2 GB main memory and 226 GB of disk, running OpenSUSE Linux 11.0. The database systems tested are MySQL 5.0.51a and PostgreSQL 8.3.1 using PostGIS 1.3.4. The databases were installed 'out of the box' with no special tuning performed. The detailed machine specifications are not relevant since all tests are performed on the same machine.

2.2. Profiling Process

The central component of the testing methodology is the *test harness*. This is a collection of Java software and configuration scripts combined to provide an automated evaluation of the databases. The objective is to keep everything the same except for the databases: use the same data, same database structures, same queries, same test harness code, running on the same machine. The aim is to compare 'apples to apples'.

Two categories of queries expected from a data warehouse were used: ad-hoc queries and batch processing (Gray 1993). Ad-hoc queries simulate user requests for data that are not planned for in advance. Batch processing corresponds to the regular procedures performed in an operational system. Both categories involve the sequential testing of a series of SQL queries and measuring the elapsed time taken. SQL is a standardized language for requesting information from a database. Elapsed time is used as this is the duration that will be experienced by a user or application in response to a query. CPU time could be recorded, however this may distort the measurements due to the presence of two CPUs: the test harness would be waiting (not accruing CPU time) while the database gathers results using the other CPU.

The overhead of establishing a connection to the database is not recorded. The database and test harness are both running on the same machine, with no network connections used. When a query is performed, the test harness processes the results returned from the database by iterating over the result set and converting the contents to a String. This simple processing ensures that all data returned are accessed and not discarded.

Prior to starting the performance profiling the database and I/O sub-system data caches are flushed to eliminate any cache effects. This is achieved by shutting down the database and restarting it, then copying a large file (1 Gigabyte) from one location to another. These operations are repeated prior to running each ad-hoc query, but only before the first of the battery of 1000 queries. Also, the sequence of ad-hoc queries are repeated a number of times and the average elapsed time recorded. This aims to reduce anomalous fluctuations in performance due to other system processes not under the control of the testing environment.

2.3. Data

Catchment boundaries of a portion of the Gwydir region were created from a 30 metre resolution Digital Elevation Model (DEM) and 1:250000 scale stream network using the Arc Hydro toolset (Stenson et al, 2008). The 87 Gwydir catchments were derived using the eight-direction pour point model creating a dendritic drainage system. The resulting drainage lines, points, and areas are shown in Figure 1. Each drainage area (catchment) has a single drainage line, draining to a single drainage point for the region. The drainage lines form a collection of nine stream systems that flow out of the Gwydir in various directions.

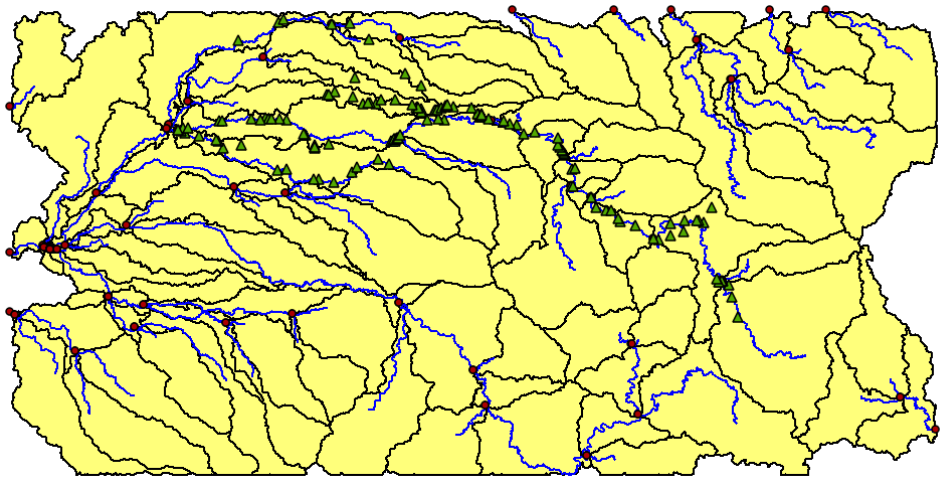

Figure 1. Gwydir drainage lines, points and areas with monitoring locations.

The green triangles in Figure 1 are the locations of monitoring instruments. Associated with these are synthetic time series data generated from hydrological models. There are 144 monitoring locations with a total of 1,705,075 measurement values spanning the time period January 1990 to June 2006.

2.4. Database Schema

The Gwydir data was exported from Arc Hydro as a collection of shapefiles and imported into PostgreSQL/PostGIS using the *shp2pgsql* tool. This process highlighted a discrepancy between the syntax used to represent polygons in shapefiles and that expected by PostgreSQL. Fortunately, the geometries were easily 'fixed', as reported in (Power, 2009). There is no corresponding shapefile import tool for MySQL: the translated data was exported as text from PostgreSQL and loaded into MySQL as SQL insert statements.

Once the data was loaded, referential integrity constraints were established by defining primary key and foreign key relationships. The primary key uniquely identifies each record in the table. A foreign key is a cross reference from one table that matches the primary key column of another table. Referential integrity is where the database enforces the foreign key relationships during inserts, updates, and deletes of data. Note that MySQL does not enforce foreign key referential integrity constraints.

Indexes were created where it was anticipated they would be needed. An index speeds up data retrieval for queries that make use of indexed columns. Each table was then 'analysed' using a database provided tool which collects internal statistics about the table and its indexes to aid the database in formulating efficient query plans: the access methods to process data in response to queries.

The resulting drainage line table for PostgreSQL is shown in Figure 2, the result of 'describing' the table using `psql`, the PostgreSQL interactive application interface. Note there are some extra columns (`tree_num`, `lvl`, `lft`, `rgt`) in the drainage line table not originally present in the shapefile. This data is generated to support network tracing up and downstream, as described for general 'tree' SQL data structures by Joe Celko[1] and adapted for water datasets (Power, 2009).

Figure 2 highlights a number of important aspects of the database structures:

- The name of the primary key column (`hydroid`) derives from the use of the Arc Hydro toolset.

[1] Celko, J. Trees in SQL. Viewed 12 Feb 2009 http://www.intelligententerprise.com/001020/celko.jhtml.

- The spatial data for each drainage line feature is stored in the `geom` column which is of type `geometry`. The specific spatial characteristics are then checked using constraints: the `geom` column is either 'null' (an empty or missing value) or has a geometry type of 'LINESTRING'.

```
    Table "public.drainageline"
   Column    |    Type    | Modifiers
-------------+------------+-----------
 hydroid     | integer    | not null
 drainid     | integer    | not null
 nextdownid  | integer    |
 from_node   | integer    | not null
 to_node     | integer    | not null
 tree_num    | integer    | not null
 lvl         | integer    | not null
 lft         | integer    | not null
 rgt         | integer    | not null
 geom        | geometry   |

Indexes:
    "drainageline_pkey" PRIMARY KEY,
        btree (hydroid)
    "drainageline_did" btree (drainid)
    "drainageline_from" btree (from_node)
    "drainageline_next" btree (nextdownid)
    "drainageline_nsml"
        btree (tree_num, lft)
    "drainageline_nsmr" btree
        (tree_num, rgt)
    "drainageline_spind" gist (geom)
    "drainageline_to" btree (to_node)

Check constraints:
    "enforce_dims_geom"
        CHECK (ndims(geom) = 2)
    "enforce_geotype_geom"
        CHECK (geometrytype(geom) =
               'LINESTRING'::text
               OR geom IS NULL)
    "enforce_srid_geom"
        CHECK (srid(geom) = 28355)

Foreign-key constraints:
    "drainageline_drainid_fkey"
        FOREIGN KEY (drainid)
        REFERENCES catchment(hydroid)
    "drainageline_nextdownid_fkey"
        FOREIGN KEY (nextdownid)
        REFERENCES drainageline(hydroid)
```

Figure 2. PostgreSQL drainage line table.

- The number of dimensions for the vertices in the geometry object is constrained to be 2.
- Indexes are present for a number of columns. An index may involve the combination of columns, for example `(tree_num, lft)`. Most indexes are described as `btree`, used for text and numeric data.
- The `geom` column has a spatial index defined, indicated as `gist`, implemented as an R-tree index.
- The geometry uses the Spatial Reference System identified as '28355', corresponding to Map Grid of Australia (MGA) zone 55.
- The drainage line table links to its catchment using the `drainid` as a foreign key.
- The `nextdownid` column is a foreign key to the next downstream drainage line, identified by its `hydroid`. The `nextdownid` column is allowed to be 'null' since on the Gwydir boundary there are no next downstream drainage lines: the data does not extend beyond this border.
- The `fromnode` and `tonode` columns correspond to the start and end vertices of the drainage lines. This defines a topologically connected structure.

Similar table columns and indexes are present in the corresponding MySQL table, but the 'check' and 'foreign key' constraints are not.

3. SAMPLE QUERIES

Database performance was initially assessed using a collection of 20 queries (Power, 2009) with the Gwydir dataset. The first 9 are general spatial queries, finding the areas of catchments, the length of drainage lines, and so on. The next 5 perform hierarchical queries to navigate upstream or downstream to identify connected catchments or drainage lines. Then 4 queries explore the time series data, aggregating the results into yearly or monthly totals. There is one example, query 19, to link the time series data to its spatial location using the monitoring table. The final query combines spatial, monitoring locations, and time series data. A summary of the results including a brief description for each query is shown in Table 1 and is discussed in Section 3.2.

3.1. SQL Examples

The original aim was to use the same SQL queries when comparing systems. This was not possible since the MySQL and PostgreSQL geospatial implementations are different. This means most of the queries used to test both systems are slightly different, as can be seen in Figures 3 and 4. This shows the SQL for query 3 to find the length of each drainage line, sorting the results into ascending order. Of the 20 queries tested, only five are the same.

```
select hydroid, glength(geom)/1000
from   DrainageLine
order by 2;
```

Figure 3. Query 3 for MySQL.

```
select hydroid, st_length(geom)/1000
from   DrainageLine
order by 2;
```

Figure 4. Query 3 for PostgreSQL.

Figures 3 and 4 show the name of the function to calculate the length of a spatial feature is different in MySQL and PostgreSQL. While this is a trivial distinction, it is cumbersome when trying to establish a single collection of SQL queries to test both systems.

There are more significant differences: MySQL cannot perform coordinate transformations, linear referencing, 'true' spatial relationship operations, nor calculate distances. Spatial relationship operations, such as 'contains' and 'intersects', only use the Minimum Bounding Box (MBB): the smallest rectangle that contains the feature. These differences required the initial collection of test queries to be revised, accommodating the reduced capabilities of MySQL. Also, the original data was provided using a geographic datum Spatial Reference System (SRS) and when calculating area or length, a coordinate transformation is required to reproject the data into a distance preserving coordinate system. Since MySQL does not support such operations, PostgreSQL was used to convert the data into the correct SRS prior to loading into MySQL.

3.2. Query Discussion

Table 1 records the average elapsed time after repeating the 20 queries ten times, a total of 200 tests. The standard deviation (σ) is included to indicate the variance in response times. While the SQL used to obtain the results of Table 1 were different for MySQL and PostgreSQL, the queries mostly perform the same task and return the same results. Queries 7 and 19 find different result sets, indicated by the different number of records retrieved for each. Also, query 20 reports different aggregate results. These differences are due to the MySQL spatial 'contains' operation using the MBB for the spatial feature, whereas PostgreSQL uses the feature's spatial description. Query 9 can not be performed by either system – it is retained to highlight that it is possible using Oracle with functions specifically provided to perform this operation.

	Query Description	# records retrieved	MySQL		PostgreSQL	
			time	σ	time	σ
1	Catchment area, in ascending sequence	87	0.03	0.001	0.09	0.069
2	Find the largest Catchment	1	0.01	0.001	0.06	0.001
3	DrainageLine length, in ascending sequence	87	0.02	0.001	0.08	0.008
4	Find the longest DrainageLine	1	0.01	0.001	0.03	0.001
5	Total area of all Catchments	1	0.01	0.001	0.06	0.006
6	Total length of all DrainageLines	1	0.01	0.001	0.03	0.001
7	Distance from point to nearest Catchment DP	2 | 1	0.01	0.001	0.05	0.015
8	Distance from point to nearest DrainagePoint	1	0.01	0.001	0.03	0.001
9	Nearest point on DrainageLine	1	n/a	n/a	n/a	n/a
10	'Source' DrainageLines (a starting DL)	48	0.01	0.001	0.02	0.001
11	Upstream Catchments from Catchment '25'	15	0.01	0.006	0.03	0.007
12	Downstream Catchments from Catchment '25'	6	0.01	0.001	0.03	0.006
13	Length of longest stream (connected DL's)	1	0.04	0.001	0.21	0.007
14	Number of DrainageLines in longest stream	1	0.08	0.001	0.36	0.003
15	Aggregate measurements by node, year	4811	3.55	0.018	46.95	0.332
16	Aggregate measurements by node for 1995.	283	0.60	0.011	0.41	0.019
17	Aggregate measurements by node, year, month	56034	4.76	0.102	51.75	0.121
18	Aggregate measurements by node, month for 1995	3396	0.91	0.017	2.47	0.078
19	Closest upstream monitoring point for each DL	30 | 19	1.43	0.006	0.95	0.034
20	Aggregate measurements for upstream nodes	84	253.76	1.363	0.52	0.042

Table 1: Database Performance with average time in seconds.

The results from Table1 provide some insight into the relative performance of the two systems. Specifically, MySQL performs best in the majority of cases, except for 16, 19, and 20. Query 20 is slow for MySQL and is the most complicated query tested, performing spatial joins, network navigation, and aggregation.

4. BOUNDING BOX QUERIES

A typical spatial query is to find all the objects that lie within a rectangular area, referred to as the Bounding Box (BB). The bounding box is defined as a subquery in SQL which requires an alias ('w'). MySQL defines the BB as a polygon, listing the corners in sequence, returning to the first point. Only the first point is shown in Figure 5 to conserve space. The MySQL 'intersects' only performs an MBB intersection and so the corresponding PostgreSQL operation is used: '&&'. The examples below return three columns: the hydroid, the number of vertices in the line, and the MBB (or 'envelope') of the matching drainage line features.

```
select hydroid, NumPoints(geom) as nverts, AsText(Envelope(geom)) as mbb
from   DrainageLine,
       (select GeomFromText('POLYGON((652954.661214154 6694795.33778429,…))') as bb) w
where  intersects(geom, bb)
order by glength(geom);
```

Figure 5. Example MySQL Bounding Box Query.

The BB could be defined in PostgreSQL using the same method as MySQL, but the 'BOX3D' function to define a 2D query region is preferred, see Figure 6 below. This defines the lower left and upper right corners of the BB. Note the use of the 'setsrid' to define the SRS as 28355. This is not needed in MySQL.

```
select hydroid, st_npoints(geom), AsText(st_envelope(geom)) as mbb
from   DrainageLine dl,
       (select setsrid('BOX3D(652954.661214154 6694795.33778429,
                              702954.661214154 6744795.33778429)'::box3d, 28355) as bb) w
where  geom && bb
order by st_length(geom);
```

Figure 6. Example PostgreSQL Bounding Box Query.

Three test collections of 1000 SQL queries were generated using a BB randomly located within the Gwydir region, but each selecting only one of the results in the select list shown above. This allows the performance of queries with varying spatial complexity to be compared.

The tests in Table 1 aimed to have the database in the same state prior to performing a query. This time, the database will make use of caching and other optimisation strategies opaque to external applications. The average elapsed times in milliseconds for the three sets of 1000 queries are shown in Table 2. The 'count' column is the total number of records retrieved for all the Set 1 queries combined, and the total number of points for Set 2.

	count	MySQL	PostgreSQL
Set 1	5250	0.835	1.935
Set 2	5082941	0.856	2.095
Set 3	n/a	1.034	2.355

Table 2: Database Performance with average times in milliseconds.

Table 2 highlights that these queries return the same information. However, as stated previously, MySQL has fewer spatial capabilities compared to PostgreSQL. These bounding box queries are fast since they make use of an approximation of the spatial description of the feature. Performing a 'true' intersects with PostgreSQL is a more computationally expensive operation and takes around 10 milliseconds for each of the query sets. For the Set 1 results in this case, the 'count' value is 4791. When using MySQL, if these extra features are not required they would have to be filtered out using some other means.

Figure 7 plots the Set 1 results for MySQL and PostgreSQL. Many of these results are most likely answered directly from the cache. Also, the databases pre-calculate common attributes, such as the MBB and the number of points in a spatial feature. The database can then simply look up these values, and not calculate them on the fly. It was hoped that more interesting spatial queries could be tested, but the limited capabilities of MySQL made it difficult to define spatially complex and motivating examples.

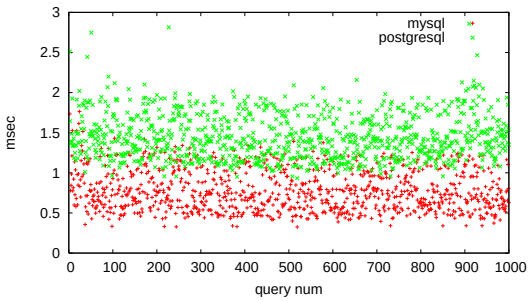

Figure 7. Set 1 results.

As a final test, the Set 1 queries were profiled using the previous process: the databases restarted and the disk cache cleared between performing each of the 1000 queries. MySQL reports an average elapsed time of 0.07 seconds (σ of 0.02); PostgreSQL 0.63 seconds (σ of 0.05). Both systems have a delay for the first query run after a restart and this overhead has been included in the results of Table 1.

5. DISCUSSION AND CONCLUSIONS

A general methodology for profiling different database systems has been described. This was used to compare MySQL and PostgreSQL/PostGIS using a small test dataset and contrived queries. The testing methodology strove to compare 'apples with apples' by using the same test harness code, sample data, and queries and only changing the database under evaluation. However, it was not possible to use the exact same queries due to spatial implementation differences.

A geospatial database provides a convenient means of maintaining water datasets. The support for spatial data types, functions, and operators built into SQL, enables complex spatial queries and analysis to be performed. PostgreSQL/PostGIS includes a rich collection of geospatial capabilities similar to a GIS, whereas MySQL only provides storage and efficient retrieval of spatial data. For these reasons, MySQL can be considered a spatial database while PostgreSQL/PostGIS a geospatial database.

The reported results should be treated with caution: the dataset is almost trivial in size and the queries tested are only simple spatial examples. However, it can be asserted that MySQL performs better than PostgreSQL for the sample dataset and queries. Note that no attention was given to optimising the database environments: the defaults were used, installing each database 'out of the box' and loading the sample data 'as is'.

The methodology of testing ad-hoc queries (database restart and clearing the disk cache) should be reserved for comparing performance of queries for the same database. To make comparisons between systems, large numbers of sample queries should be generated, for example generating the test SQL queries using a 'template', and then profiling performed and averages calculated.

There is still a large parameter space to explore: testing database specific JDBC driver options; different database schemas; variations to the table and index structures, optimising the SQL queries, and simulating multiple users. The impact of these variables is difficult to estimate: databases are complex systems and optimal performance is often only achieved after careful attention is paid to the various aspects of tuning.

The AHGF Project will deliver hydrological data products and describe stakeholder user requirements. This data and the queries can then be used to exercise the presented methodology to compare different systems. Performance results including Oracle Spatial have been conducted, but they cannot be reported due to licensing constraints. These tests explored a wider range of geospatial capabilities including functions expected from hydrological applications. ESRI's ArcSDE and Microsoft's SQL Server will also be tested. An advantage of ArcSDE is that it can be deployed on top of various databases allowing applications to be insulated from different geospatial implementations. This is appealing given our experience in the lack of compatibility of geospatial extensions. There will be other tradeoffs, and these will need to be examined.

These results can be considered a baseline performance against which improvements can be sought. The test harness code and sample data used are available upon request and others are encouraged to verify the reported findings. Given the differences in available geospatial databases and computing platforms, your mileage may vary.

ACKNOWLEDGMENTS

The AHGF Project is part of a five-year water information research and development alliance between the CSIRO Water for a Healthy Country Flagship and the Bureau of Meteorology. Jenet Austin and Matt Stenson provided the Gwydir data and Linda Merrin explained the need for coordinate transformations. David Lemon and Rob Atkinson improved the paper by clarifying the original aims and ensuring the various technical terms were defined. Michael Kearney, Amit Parashar, Ross Ackland and Peter Thew reviewed earlier versions. Thanks to Bureau of Meteorology colleagues Elizabeth and Dovey Dee for supporting this work.

REFERENCES

Atkinson, R., Power, R., Lemon, D. and O'Hagan, R. (2008). The Australian Hydrological Geospatial Fabric – Development Methodology and Conceptual Architecture. CSIRO Water for a Healthy Country National Research Flagship, Canberra, Australia, 57p.

Gray, J. (editor) (1993). The Benchmark Handbook for Database and Transaction Systems (2nd Edition). Morgan Kaufmann.

Power, R. (2009). Profiling Spatial Databases: a Gwydir Case Study. Technical report, CSIRO ICT Centre, GPO Box 664 Canberra ACT 2601, March 2009. 09/161.

Stenson, M., Ryan, C., Reed, A., Austin, J., Lemon, D. (2008). Australian Hydrologic Geospatial Fabric Project - Arc Hydro Pilot. CSIRO Water for a Healthy Country Flagship, Canberra, Australia, 56p.