Exploiting Spatial Database Software for use in Teaching SQL

Simon Burt
UCOL
Palmerston North
s.burt@ucol.ac.nz

This quality assured paper appeared at the 2nd annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2011) incorporating the *24th Annual Conference of the National Advisory Committee on Computing Qualifications*, Rotorua, New Zealand, July 6-8. Samuel Mann and Michael Verhaart (Eds).

Abstract
This paper proposes the use of spatial data for teaching and assessment of SQL skills. The advantages over non-spatial data are described. Various database servers are evaluated and Microsoft SQL Server 2008 and PostGIS are recommended. Basic SQL commands were trialled against sample data based on countries of the world. The immediate visual feedback afforded by spatial data is presented as a great benefit to the student and teacher.

Keywords
Teaching SQL, spatial data, spatial database.

Introduction
The Structured Query Language (SQL – pronounced "sequel"), is a de facto standard language for retrieving data from a relational database. There are differences in implementation of SQL between databases, but many commonalities between them. SQL's widespread use in industry dictates that it is taught in ICT programmes. The teaching of SQL is well covered by the literature (Caldeira 2008), (Kearns et al. 1996), (Raadt et al. 2006), and the key learning outcomes will typically include inner joins, "where" clause, "group by", aggregates, having, insert, update, and delete capabilities of the language.

Visual aids to teach programming languages have a long history (Solomon & Papert 1976), and readers of this paper might remember one of the numerous incarnations of "turtle graphics"[1]. Visual aids to aid the teaching of SQL are more recent; for example Mitrovic (1998), Allenstein et al. (2008) and S. Sadiq et al. (2004) all introduce software that provides the student with a visual aid to understanding either the underlying database schema, or the SQL query process itself.

This paper demonstrates that spatial data can be used to aid students to learn the fundamentals of SQL. The major advantage in doing so is that the *results* of many spatial queries can be displayed visually. Visual results to SQL queries give feedback to the student in a form that they are likely to instantly comprehend. Another difference to this approach, compared to the other visual aids to teaching SQL mentioned above, is that no special software needs to be written. As a further consequence, students do not need to learn any new application, over and above what would have to be used anyway.

It is important to emphasise that spatial data can be introduced without students knowing anything about co-ordinate systems, or indeed, that there is anything "special" about spatial data per se.

Methodology

A list of requirements for the spatial database was produced as a basis for evaluation. Requirements included: affordability; compliance with standards; operating system restrictions; minimum hardware specifications; ease of installation; support tools; and multi-user support.

[1] This is not to imply that turtle graphics are not still being used to teach programming; a recent but somewhat esoteric example can be seen in Bush (2009).

Many different software packages were installed, as this was the quickest and simplest method of working out what a particular package was capable of doing, and whether or not it would meet the requirements. This author found merit in many packages, not just those recommended here. Since there was so many different options to choose from in the FOSS arena, software was selected for trial based upon perceived support from the online community, and the frequency (or lack thereof) of software updates. Since this paper is not a review or evaluation of spatial databases, only software that would meet the requirements was selected. No attempt to find the "best" software was made; if it could meet the requirements, then it was investigated further.

A spatial database needs spatial data. For reasons discussed later, a sample database was created based on countries of the world.

Once the dataset was complete, experimentation with the data enabled the creation of suggestions for how to teach the basics of SQL.

Results

Microsoft SQL Server 2008 and PostGIS were both identified as being fit for purpose. A summary of the reasoning now follows.

Affordability

The big names in spatial data such as Oracle, ESRI and Intergraph were not tested in any depth because they did not meet the affordability criteria. The School of Business and Computing at UCOL (herein the "School") is a member of Microsoft's Academic Alliance (MSDNAA) and as such, can use Microsoft SQL Server for no additional cost. Microsoft also provides SQL Server 2008 Express Edition, which is a limited version of the full database, but free of charge. The Express Edition can be used by students at home. PostGIS is FOSS and hence also met the affordability constraints.

Compliance with Standards

The Open Geospatial Consortium (OGC) is a standards body claiming to have over 300 active members including ESRI, Oracle, Microsoft and Intergraph (OGC n.d.). Using spatial data as an aid to teaching SQL does not demand OGC compliance, but where such a widely supported standard exists, it was an easy decision to specify that any spatial database must be OGC compliant. Since most spatial databases claimed at least *some* compliance with the OGC standards, there was little danger in forcing this constraint. An OGC compliant database supports *geometry* and *geography* data types, as well as numerous spatial functions that can be performed with these data types. Both recommended databases showed adequate compliance with the OGC standards.

Minimum Hardware Specifications

The hardware used for testing the software was not fast by modern standards. However, performance was never an issue with any of the tested software, and would not be expected to be so even in a multi-user environment. A 2GHz Intel CPU with 512MB RAM was sufficient for all single-user instances of the software. More RAM would be advisable for multi-user Windows environments.

Ease of Installation

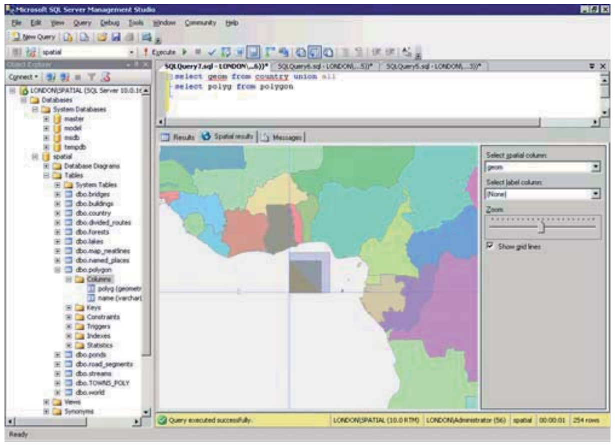
The School uses Windows 7 as the desktop operating system, and Ubuntu Linux and Windows Server 2003/2008 on the servers. Consideration was also made for students working at home, who for the most part, are exclusively using Microsoft operating systems. The prerequisites for running either edition of Microsoft SQL Sever 2008 are quite onerous: dot net framework 3.5 SP1; Windows installer 4.5; and Windows Powershell 1.0. A number of non-default options were required for a correct installation, which used over 1GB of disk space and took over 30 minutes to complete. To ease expected difficulties in this area, an installation guide was written. No issues arose when installing PostGIS in either a Windows or Linux environment so no installation guide would be required.

Support Tools

The primary support tool for SQL Server is the Microsoft SQL Server Management Studio (SSMS). Students of the School are normally expected to gain familiarity with SSMS, therefore there would be no extra burden in demanding its use. The killer advantage to using MSSMS is that spatial data can be graphically displayed, simply by clicking on the "Spatial results" tab that appears whenever a query returns spatial data, as shown in Figure 1.

The primary support tool for PostGIS is pgAdminIII and this tool contains no specific spatial extensions, indeed some problems were found with pgAdminIII's ability to display the wide columns of data prevalent in spatial data. In order to display PostGIS spatial data an extra tool is therefore required. Of the large number of choices available, this author chose OpenJump, as it seemed to be the preferred choice of the PostGIS developers (BostonGIS n.d.). Figure 2 gives an example of two SQL queries displayed in OpenJump which are produced in layers, a concept that users familiar with GIS systems will recognise. The auto-colouring is also not as refined as with SSMS, although it can be manually changed.

Microsoft's SQL Server had some export/import issues when spatial data was involved. A guide to solving some of the problems encountered was written. The backup and restore utilities, available from pgAdminIII, worked without issue with spatial data.



```
select geom from country union all
select polyg from polygon
```

Figure 1. The Spatial results tab of SSMS displays spatial data in a visual format; here showing a union of simple shapes and countries. The colours are auto-generated.

Multi-user support.

Multi-user support is essential for the classroom environment where students' work needs to be centrally backed-up and administrated. SQL Server's authentication can be integrated with the operating system authentication, providing seamless logins and some institutes may find this beneficial. A default install of PostGIS requires the students to provide a username/password which necessitates some administrative effort from the supervisor.

Spatial Datasets

The spatial dataset used for testing and development were freely available, but came from numerous sources. They often required some cleansing or reformatting to ensure suitability. For example, some sources of data differed with their country code names. The final data sets however do not require any further modification.

Teaching SQL

Experimentation with the dataset quickly led to some basic ideas on how to teach SQL and examples are given here.

The Select Statement

Instead of a basic select statement such as this:

```
select item_id, item_name from order;
```

a query of similar complexity using spatial data can be used:

```
select cntry_name, geom from country;
```

The first SQL statement requires extra cognitive effort from the student – ie. students will need some understanding of "an order", "an item" and "an item id". Since the field item_id has no real-world equivalent, the lecturer can expect to spend some time explaining these terms. Students entering the correct query will be presented either with a lot of text, or a table containing a lot of text. There is nothing about the result that tells the student that they have got the correct answer, or how many rows it should return. Indeed, in this author's experience, some students who enter a text-based query (or command), will accept any output as valid, including error messages in red text, and no output at all.

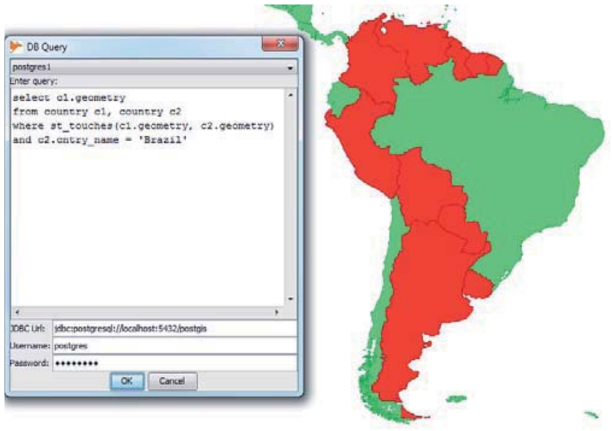
With the spatial query, the geom field contains the spatial data, and the result of this second query will be a map of the world. Students who enter this query correctly, are "rewarded" with a picture which they will intrinsically recognise as correct, especially if they perform a quick comparison with a colleague's screen. An instructor will also be able to quickly ascertain which students are having difficulty by asking to see a student's map. A text-based table is also produced by this query which lists the names of the countries.

Where Clauses and Table Joins
The "where" clause can be introduced with:

```
select cntry_name, geom from country
where cntry_name = 'New Zealand'
```

This approach has a clear visual appeal, and in most cases, students will be able to ascertain the correctness of their answer, without recourse to a lecturer. This will include ESL students

Table joins clearly require more than one table in the dataset. A region table containing country to continent mappings allows the teacher to instruct the students on how to produce a map of Africa or South America. Students will be able to recognise for themselves whether or not they have the correct answer.



```
select c1.geometry
from country c1, country c2
where st_touches(c1.geometry, c2.geometry)
and c2.cntry_name = 'Brazil'
```

JDBC Url: jdbc:postgresql://localhost:5432/postgis
Username: postgres
Password: ••••••••

Figure 2. An example of spatial data in a PostGIS database displayed using OpenJump. Two layers are shown here, the lower is the world map in green (lighter colour), the uppermost in red (darker colour) is a result of the query shown

Aggregate Functions and Group by
Standard SQL features such as sum(), count(), and order by, are all easily accommodated with the country and region tables. For example, the use of sum() can be introduced to answer a question such as "what is the population of Africa", and the count() function introduced for "how many countries are there in Africa". The teacher can have some degree of confidence that students will understand questions of this nature, and that the answer to these questions should be a number of a reasonable magnitude. An equivalent non-visual question such as "what is the total value of the order" or "how many orders has a customer made", may need some further explanation, and further abstraction by the student. The student will have no way of checking that the number produced by their query is of the

correct order of magnitude, let alone the correct answer.

The “group by” clause can be similarly introduced with “list all the continents and their populations”, and here again the students can be expected to judge the correctness of their answer for themselves.

Insertion and Modification of Data

Once students have confidence in querying data, the next stage will be inserting and modifying data using SQL “insert into” and “update” commands. If spatial data is to be included in the exercises, then clearly the raw geometry data type (which is a large hexadecimal number) presents a difficulty, as does the earth’s coordinate system, which most students cannot be expected to know, and it may be outside the scope of the paper for inclusion. Rather than plunge students into complex data types and coordinate systems, simple shapes on a Cartesian coordinate system can be used. Students can be asked to create a table that will contain some shapes.

```
create table shape (shape geometry,
"name" character varying);
```

Students can then be asked to insert shapes into this table starting with a square:

```
insert into shape(shape, "name")
values ( GeomFromText('POLYGON((1 0,1
3,3 3,3 0,1 0))', -1), 'square');
```

Perhaps the lecturer will hand out a sheet of squared paper to aid in this regard, or give the students a grid with some shapes on it and ask the students to create an entry in their table for each shape. A simple query, with its graphical output, will confirm whether or not the students have completed this task correctly. Students can be introduced to points, lines and polygons in this way. To introduce the use of functions, the students could be asked to create their own shapes, and given the challenge of creating a circle of radius 3 say. The difficulty of creating the circle will naturally lead to defining a circle as a point with a “buffer” around it, leading to the use of the STBuffer() function.

Working with the world map again, students could be asked to create a table called ‘capital’ that has an entry for each capital city. Using the Web, students can quickly ascertain the location of a capital city, and then insert it into a table:

```
insert into capital values
('Wellington', 'POINT(174.7762 -
41.2865)');
```

The lecturer may not need to introduce coordinate systems to the students. At this stage it does not matter (to the student) that the Cartesian units used here are degrees, and not a *direct* measure of distance such as kilometres. Points can be difficult to spot on a global map, so this gives another opportunity to demonstrate STBuffer().

Some students are likely to get the eastings/northings reversed since it is not obvious which way they should be inserted. However, since the results are visual, it can be left to the student to correct any errors; ie the student can be expected to know that the capital of Australia should appear on the map somewhere inside Australia.

Assessment
In order to assess students' abilities to formulate SQL queries, a question using non-spatial data such as this has been proposed:

"*For each director, list the director's number and the total number of awards won by comedies he/she directed if that number is greater than 1*." (Mitrovic 1998)

There are some significant problems with this question. Firstly the "director's number" has no real-life analogy. All students will need to spend some time examining the tables to work out exactly which field is required here, and if a table contains a field containing the director's telephone number, misinterpretations, and not just from ESL students, can be expected.

Secondly, the sentence seems overly contrived, presumably because the author wanted to avoid the use of the words "where" and "having" in the question. An ESL student can be expected to struggle with the meaning of this question, and even native English speakers would need to read this at least twice to gain the correct meaning. The question could be better phrased as:

"*For each director, list the director's full name, and the total number of awards that have been won by comedies he or she directed. Only include director's that have won two or more awards for comedies*."

But even this seems contrived and will require significant effort in interpretation by the students. This author would argue that it is the student's SQL skills that need to be assessed, not their English language interpretation skills.

The question above is clearly trying to assess the students on their ability to create a select query, with a *where* clause, a *group by* statement, and a *having* clause. The equivalent assessment using spatial data looks like this:

"*Display the continents that have a total population less than 500 million.*"

This sentence is clear and concise, and has little room for misinterpretation. As a further advantage, students with only a rudimentary knowledge of the world will realise that their query is incorrect if their result includes Asia. The lecturer could also show the visual output in this case, without giving away any details about the SQL that produced it. The lecturer can also quickly ascertain whether a student has the correct SQL using a quick visual inspection of the output.

Conclusion

Microsoft SQL Server and PostGIS are viable options to use in the teaching of basic SQL. Their relative merits mean that one or both will meet the requirements of most teaching institutes. SQL Server requires more initial effort to get it going, and keep it working, than does PostGIS. The Management Studio for SQL Server is an excellent tool for displaying spatial data.

Visual-based queries using data based on countries, as demonstrated in this report, enables the lecturer to phrase questions in a manner that non-native English speakers can be expected to understand. For example, the meaning of "how many countries are there in Europe?" is likely to be understood immediately, so that the students can concentrate on the SQL, not on the meaning or interpretation of the question. This is especially important for assessments, as the students may have limited access to the lecturer to question the meaning of a question during an exam.

Students can be expected to have some notion of the order of magnitude of spatial results, and therefore should not demand as much feedback from the lecturer as to whether or not some answer is correct. If a student runs a query to answer a question about the number of countries in Europe, then a student may work out for themselves that an answer outside the range of 10-50 is probably wrong, and they need to look at their SQL more closely.

Further Work

Using spatial data on real students has been scheduled for semester 2, 2011. Before this can take place, formal lesson plans will be developed, expanding on the ideas given here.

References and Citations

Allenstein, B. et al., (2008). A query simulation system to illustrate database query execution. *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. Portland, OR, USA: ACM, pp. 493-497.

BostonGIS, Boston GIS: Tutorial Booklet. Available at: http://www.bostongis.com/TutBook.aspx#205 [Accessed October 11, 2009].

Bush, B.J., (2009). Solving the eltrut problem with hybrid evolutionary algorithms. *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. Montreal, Québec, Canada: ACM, pp. 2701-2704.

Caldeira, C.P., (2008). Teaching SQL: a case study. *Proceedings of the 13th annual conference on Innovation and technology in computer science education*. Madrid, Spain: ACM, pp. 340-340.

Kearns, R., Shead, S. & Fekete, A., (1996). A teaching system for SQL. *Proceedings of the 2nd Australasian conference on Computer science education*. The Univ. of Melbourne, Australia: ACM, pp. 224-231.

Mitrovic, A., (1998). A Knowledge-Based Teaching System for SQL. *IN ED-MEDIA 98*, 1027--1032.

OGC, OGC Members | OGC®. Available at: http://www.opengeospatial.org/ogc/members [Accessed October 10, 2009].

Raadt, M.D., Dekeyser, S. & Lee, T.Y., (2006). Do students <i>SQLify</i>? improving learning outcomes with peer review and enhanced computer assisted assessment of querying skills. *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*. Uppsala, Sweden: ACM, pp. 101-108.

Sadiq, S. et al., (2004). SQLator: an online SQL learning workbench. *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*. Leeds, United Kingdom: ACM, pp. 223-227.

Solomon, C.J. & Papert, S., 1976. A case study of a young child doing turtle graphics in LOGO. *Proceedings of the June 7-10, 1976, national computer conference and exposition*. New York, New York: ACM, pp. 1049-1056.