

Introduction to the Pragmatic Teaching of Database Concepts

Arjen Lentz
Shaun Nykvist

Community Relations Manager, Trainer, MySQL AB (arjen@mysql.com)
Faculty of Education, Queensland University of Technology (s.nykvist@qut.edu.au)

ABSTRACT

Databases play a crucial role in today's connected world. Generally, they are invisible to a customer or employee, yet are 'under the hood' of nearly every web site and business application. Essentially a good understanding of databases allows us to effectively deal with data such that we have worthwhile information. This paper describes a practical approach to teaching essential database theory to computer studies students and undergraduate education students. It highlights a practical approach to illustrating set theory in terms of diagrammatic examples and relates this to junior mathematics concepts. While we discuss the importance of representing data in usable forms we will relate these principles to the SQL language and demonstrate how students can use this in both junior and senior computer studies courses for a greater concrete understanding of database concepts. Further to this, we describe a number of affordable and easy to use, yet powerful tools that are available to all students. We also make links between simple programming languages and back end databases.

DATABASES IN TODAY'S SOCIETY

Information available to people today has grown exponentially (Hilton 1992:x) and since a teacher or individual cannot possibly retain all the information available today, it is imperative that one develops skills in finding, interpreting and updating the required information (Sharp 1993:113). Databases no longer exist purely in the domain of information technology. Databases are used in many facets of society by an array of people, ranging from professionals to students to researchers to individual members of the public. Today, much of our business infrastructure relies on databases for storage, management and efficient retrieval of information. Whether dealing with customer details, sales records, library inquiries, or perhaps a cheap flight or holiday destination, databases are directly or indirectly involved. Essentially, a database is an all-encompassing term that describes anything from an 'address book, recipe box, dictionary or a filing cabinet to a set of computerised data files with sophisticated data relationships' (Sharp 1993:113).

The Internet and in particular the Wide World Web (WWW) has further changed the way in which we access various sources of information. Library catalogues, indexes and databases of websites, shopping stores, radio and music stations, timetables for movies, transport and other events, education training materials and a plethora of other information and services is all available on-line. Access to these resources has also increased with a 35 to 95 percent increase in access to the Internet in school classrooms between 1994 and 1998 (William 2000). In recent times most developed countries have gained access to the internet in their schools, however, according to Becker (1999), many teachers and students merely use this access as an information gathering tool. If this is indeed the case, then the need for understanding databases is at the forefront of this activity. Having a sound understanding of basic set logic would aid in the effectiveness of gathering or searching for information. For example, a simple search in a web browser such as Google with an "or" and "not" operator in addition to the default 'and' operator can make a large difference to a search query. Hence, basic understanding of set logic not only enables more effective searches, but it also allows one to manipulate and form more powerful actions on a database. Given that much of the information and many of the services available on the Internet involves some type of database management software, it is necessary for a large number of people to work with these databases. Computer studies students also need to be able to

interact with databases and have a sound knowledge in set theory and the relationship to the SQL language. In Queensland schools a large number of courses are incorporating the design of information systems and programming in such a manner that students produce interactive websites (QSITE 2006).

Often when people think of a database management system (DBMS) they think about products such as Oracle, Microsoft Access, or MySQL. These products are often identified as Relational Database Management Systems (RDBMS) and will form the focus of the remainder of this paper. While we have placed emphasis on the plethora of database driven web systems that exist, the following sections describe an approach used by students that is generic to most database management systems.

INTRODUCING RELATIONAL DATABASES

At a tertiary level, relational databases are introduced through the theoretical concepts they are built on, as first developed by E.F. Codd in the 1970s (Codd 1970). The SQL language is introduced thereafter and there are a number of standards associated with this, the most recent publication of these standards being in 2003. It must be understood that the implementations (i.e., actual relational database products) are not 'pure' relational systems, and the SQL language is in many ways imperfect from the theoretical perspective.

Evading this quagmire, it is our proposal to take a pragmatic approach and investigate how we can teach students to work with and understand today's actual products and the SQL language. The key aspects of relational theory can be mapped into very practical rules and guidelines, which are easily understood independently (i.e., without a background in either IT, or relational theory). These concepts are suitable to a range of learning abilities.

The approach that this course takes with students, tries to build on the familiarity of other common tools. Often students will usually be more familiar with spreadsheets, and this can be used as a general introduction to the topic of databases. In a spreadsheet, information is stored in cells, which are arranged in rows and columns. We have found that students with limited IT background are able to use this simply structured system in an effective manner.

However, relational databases are definitely not spreadsheets and spreadsheets are, even more definitely, not relational databases and through the comparison of the two it is anticipated that students are more informed when it comes to choosing whether they need a spreadsheet or a database for a particular task. While a database has rows and columns, a number of additional rules are also applied. For instance, in a spreadsheet one can configure each individual cell to store a number, a string of text, or a date. A relational database consists of one or more tables, each containing zero or more rows with an equal and specified structure. Each row has the same number of columns, and each column is defined to have a specific type. So, this column will be of that type, regardless of which row we look at.

<i>Column name:</i>	<i>Firstname</i>	<i>Lastname</i>	<i>City</i>
<i>Row 1:</i>	David	Johnson	Brisbane
<i>Row 2:</i>	Mark	Andrews	Sydney
...
<i>Row N:</i>			

Table 1: a simple relational database table

In some respects, using the analogy of a filing cabinet (drawers, records, fields) appears more fitting on the conceptual level. However, since a traditional filing cabinet has even less enforceable structure

than a spreadsheet, it is our suggestion that an introduction incorporates the analogy of both spreadsheets and filing cabinets, for maximum effectiveness.

The SQL Language

Most commonly, SQL (Structured Query Language) is used to interact with relational databases. Basic SQL looks much like English, though advanced use exhibits many grammatical quirks.

Examples of basic SQL:

- `SELECT * FROM contacts;`
- `INSERT INTO contacts (firstname, lastname, city) VALUES ('David', 'Johnson', 'Brisbane');`
- `SELECT firstname, lastname FROM contacts WHERE city = 'Brisbane';`
- `UPDATE contacts SET city = 'Cairns'`
`WHERE firstname = 'David' AND lastname = 'Johnson';`
- `DELETE FROM contacts WHERE firstname = 'David' AND lastname = 'Johnson';`

SQL and Sets

Database tables, and results returned by the SQL `SELECT` command, are *sets*. These are basic mathematical concepts that many students are introduced to in primary and sometimes secondary school. In set theory students look at the relationships between sets through Venn diagrams, subsets and set operations. This is the practical basis of relational databases, and it would be entirely safe to regard them as one and the same. The term *relational* is a direct reference to the way this type of database system handles its data: in mathematical relations (sets). It is important to note that it is within this context that sets contain no duplicate elements and that ordering of elements is not significant.

Numbers = { 1,2,3,4 }
Fruit = { Apple, Carrot, Banana }

Table 1: Basic Sets

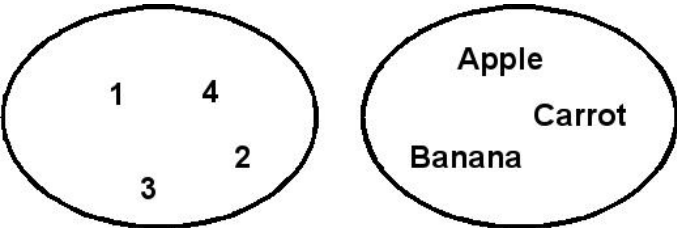


Diagram 1: Basic Venn diagrams

Binary (boolean) and Ternary Logic

As can be seen in the above examples, the SQL language utilizes binary logic (for the proposition P, P is true or P is false) . For example, we wanted to update the record where the firstname field equals 'David', *and* the lastname field equals 'Johnson'. Other records which may have a firstname 'David' or a lastname of 'Johnson', but not both, are not returned. Hence, the concept of binary logic (AND, OR, NOT, equal) and corresponding truth tables is illustrated.

In addition, SQL uses NULL to indicate where a value is *unknown*, making the logic ternary and a simple extension of the basic binary truth tables is all that is required (table 2). With sets, NULL is not a value and lies outside the domain (diagram 2). In SQL, NULL behaviour does not appear entirely logical in all cases, but the practical effects are fairly minimal.

	<i>False</i>	<i>True</i>	<i>Null</i>
<i>False</i>	False	False	False
<i>True</i>	False	True	Null
<i>Null</i>	False	Null	Null

Table 2: Ternary truth table for AND operations

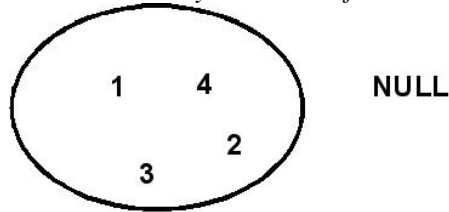


Diagram 2: NULL outside value domain

Set Operations (Joins)

Commonly, a database consists of multiple tables, and these tables are used together inside SQL commands. Relational designs should **always** avoid duplication of stored 'facts' or propositions, which is another key difference from spreadsheets. If we want to set up a database with Australian states, cities, we would not want to keep repeating the name of the state for each city.

<i>StateID</i>	<i>StateName</i>
1	Queensland
2	New South Wales
3	Victoria
4	Tasmania
...	

Table 4: States table

<i>CityName</i>	<i>StateID</i>
Brisbane	1
Cairns	1
Sydney	2
Melbourne	3
...	

Table 5: Cities table

We add a so-called *surrogate key* (an artificial primary key) for each state, and each city has a reference (foreign key) to the appropriate state. We can now ask our database system to tell us which cities belong to which state, and other similar queries, using a combination of the two tables. In SQL, such operations are called joins. Two types of joins exist, inner and outer. Using Venn diagrams, joins are easily explained in a visual way. The basis of joins is dependent on set multiplication, often referred to as the Cartesian product. Essentially a join is an operation which combines elements of two or more datasets by specifying which combinations are valid. *Inner joins* are introduced first:

```
SELECT StateName, CityName
FROM States s JOIN Cities c ON s.StateID = c.StateID
WHERE StateName = 'Queensland';
```

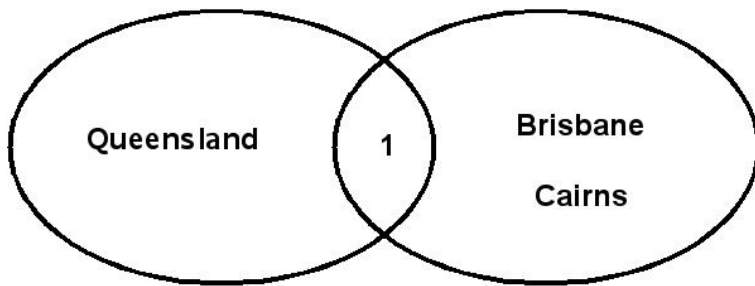


Diagram 3: Inner Join of States and Cities tables

Outer joins are typically used to find mismatches. We could, for instance, find out which customers have *not* bought a particular product. Or, re-using our data set from above, we could find out that we haven't actually added Hobart to the Cities table and thus Tasmania does not have any Cities associated with it. In an SQL query, the CityName column would become NULL on the row for Tasmania, and this fact can be used to find out exactly we want about our data set.

```
SELECT StateName, CityName
FROM States s LEFT JOIN Cities c ON s.StateID = c.StateID
WHERE c.StateID IS NULL;
```

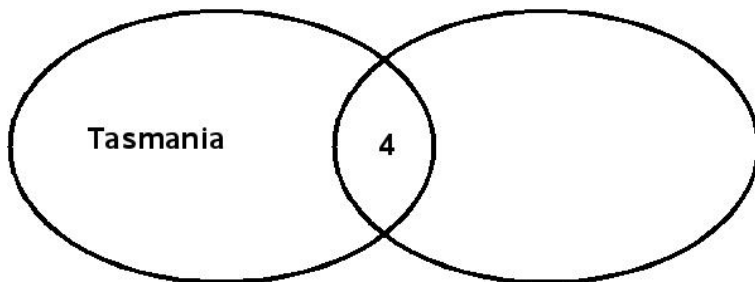


Diagram 4: Outer Join of States and Cities tables

The concepts associated with the previous SQL queries are often difficult for a large number of students and it is our intention to give students a more visual approach to this topic from a set theory perspective as illustrated above. While set theory is essential to performing effective and efficient SQL queries on data, it is important to note that the applicability of set theory to relational databases extends to other operations like UNION, INTERSECTION, DIFFERENCE, DIVIDE-BY and sub-setting operations. Further, we also note the importance of constructing well designed relational databases, however these topics are beyond the realms of this paper.

TOOLS

Naturally, at some point practical application of the set theory is necessary, and appropriate computer tools are needed. There are a number of database tools that exist, some of which are available to both students and staff at no cost, others come with a fee. Given that we live in a networked society and many students have access to a computer and the internet it is essential that they can use the same tools at home to complete assessment tasks. There are a number of open source software solutions which can assist students in their understanding and development of databases and SQL. Many of these solutions are simple to use yet quite powerful, with many of them being used in enterprise situations.

Database Server

MySQL is probably the most well known *open source* database server, and the most wide spread

(Garry 2003). The available versions are fully featured, contrary to other 'free' offerings this is not cripple-ware. MySQL version 5.0 features a full array of common features such as transactions, update-able views, sub-queries, foreign key constraints, SQL standard stored procedures, triggers, and even advanced clustering (should you need it in a school situation). Version 5.1 also offers partitioning. The same software that can freely used for teaching is used in the real world, from small personal web sites to big data-warehouses. Also, if students choose to study computer science later, the full source code is indeed available for in-depth exploration.

MySQL is available for Windows, Mac OS X, Novell NetWare and every Unix-like system including Linux. Once installed, MySQL installations behave identically regardless of the operating system used. MySQL actually comes pre-installed on any Linux, NetWare and OS X server. Students can set up MySQL on a laptop or their computer at home, no matter what type it may be. Web hosts generally offer access to a MySQL database also, this can be useful for projects.

Last but not least, while the nature of open source software and its licensing falls outside the scope of this paper, with MySQL there is neither a financial obstacle, nor a build-in reciprocal requirement that burdens either the school or the student. Full educational independence can be maintained, without the need for 'gifts' or other corporate sponsorship.

Client Tools

We strongly recommend against introducing this matter using visual tools. In the field, we have found that students tend to acquire bad habits as the actual database and query structure remains hidden. Naturally, once the fundamentals are understood, visual tools can be an excellent development tool and would speed up the process, however it is important to understand the basics first, especially if you are going to use a web programming language such as PHP to connect to a database. Note: many students may in fact prefer to keep using the direct approach as well. This is an excellent skill to have in the market place.

We suggest using either the command line interface at first, requiring students to type queries directly, but web-based and graphical tools that offer similar functionality (and much more) are also available. The MySQL server software could be installed in one central location, with all students connecting to it from the appropriate tool such as putty on a windows machine or terminal on a Macintosh or Linux system. Another easy way of accessing the database is with a web browser and using tools such as PHPmyAdmin.

CONCLUSION

We have introduced relational database systems and the SQL language entirely through the relatively straightforward application of mathematics: sets, binary logic and Venn diagrams. While this paper has reinforced the relationship between set theory mathematics and SQL database operations, it is important to note that the introduction to databases from this perspective should initially occur without the use of a computer and then be reinforced through the use of a simple command line interface application that reinforces the underlying mathematical concepts. What is important to note here is the pedagogical issues associated with the introduction of databases in the curriculum, and it is this area that will form the emphasis of further research.

Through this method of teaching, students acquire generic skills that apply to any relational database management system. While we recommend using a particular database system to teach, this choice does not limit students for the future. We explicitly focus on general skills rather than using proprietary aspects of a specific product or user interface. With such a solid foundation, students will be well equipped to understand and utilize current and future products effectively, applying their skills in the real world.

References

- Becker, H. J. (1999). *Internet use by teachers*. Retrieved October 10, 2002, from <http://www.crito.uci.edu/TLC/FINDINGS/internet-use/>.
- Codd, E.F. (1970), A Relational Model of Data for Large Shared Data Banks, *Communications of the ACM*, 13(6), p377–387. Eprint <http://www.acm.org/classics/nov95/toc.html>.
- Garry, C. (2003), *MySQL: Open Source and the Commodity Effectively*, Meta Group.
- Hilton, T. L. (1992). *Using National Databases in Educational Research*. Lawrence Erlbaum Associates, New Jersey.
- National Academy of Sciences (1999), The Rise of Relational Databases, Chapter 6 in *Funding a Revolution: Government Support for Computing Research*, National Academy Press, Washington DC. Eprint <http://www.nap.edu/readingroom/books/far>.
- QSITE (2006). QSITE IPT mailing list. Accessed at qsite-ipt@qsite.edu.au.
- Sharp, V. (1993). *Computer Education for Teachers*. Brown and Benchmark Publishers: Australia.
- Williams, C. (2000). *Internet Access in Public Schools and Classrooms: 1994-99* (NCES 2000-086). U.S. Department of Education, National Center for Education Statistics. Washington, DC: U.S. Government Printing Office.
- Yuhanna, N. (2004), *Open Source Databases Come of Age*, Forrester Research.