BibTeX meets relational databases

Dedicated to the memory of
Edgar Frank "Ted" Codd (1923–2003) and
James Nicholas "Jim" Gray (1944–2007)

Nelson H. F. Beebe

Abstract

After giving some background and comments on the BibTeX bibliographic database system, we discuss the problem of searching large collections of such data. We briefly describe how relational databases are structured and queried.

Portable new programs, bibtosql and bibsql, are introduced and their use is illustrated. The first handles the conversion of BibTeX data to input for three free, popular, and portable, database systems. The second provides a uniform and simple user interface for issuing search queries to any of the supported backend databases.

We finish with a discussion of the contributions of the two late computer scientists to whom this article is dedicated.

Contents

1 Introduction

Oren Patashnik's BibTeX has been in wide use in the TeX and LaTeX communities now for more than two decades after it was introduced in the *LaTeX User's Guide and Reference Manual* (Lamport 1985, Appendix B). The first mention of BibTeX in that book is at the bottom of page 72, where we find

> With LaTeX, you can either produce the list of sources yourself or else use a separate program called BibTeX to generate it from information contained in a bibliographic *database*.

The last word on that page has been emphasized here, and the current online edition of the *Oxford English Dictionary* defines it like this:

> **database**, n. ('deɪtəbeɪs) [f. DATA n. pl. + BASE n]
>
> 1. A structured collection of data held in computer storage; esp. one that incorporates software to make it accessible in a variety of ways; transf., any large collection of information.
> 2. Special Comb.: database management, the organization and manipulation of data in a database; database management system, a software package that provides all the functions required for database management; abbrev. DBMS s.v. D III.
> 3. database manager = database management system above; database system, a database together with a database management system.

The earliest mention recorded in the word history in that entry is from a technical memo of System Development Corporation, California in 1962 that describes it like this:

> A 'data base' is a collection of entries containing item information that can vary in its storage media and in the characteristics of its entries and items.

As often happens in English, increased use of a compound word has collapsed it into a single word.

2 An overview of BibTeX

BibTeX is a specialized database engine that takes from the `.aux` file a list of bibliography files, bibliographic style information, and a reference to a publication, all encoded in TeX commands like these:

```
\bibdata{bibsql,master}
\bibstyle{alpha}
\citation{Lamport:1985:LDP}
```

BibTeX locates bibliography files on a user-definable search path, and searches each file in turn for a BibTeX entry identified by the specified citation label, terminating the search at the first match. It also finds the first instance of the style file, here `alpha.bst`, in another user-specified search path. It then decodes the BibTeX entry to collect fields such as `author`, `title`, `year`, and so on, and then executes functions in the style file that format the publication data, usually with TeX markup, and write it to the `.bbl` file. When TeX or LaTeX next process the document, the expansion of the `\bibliography` macro includes commands to read the `.bbl` file and typeset the entry, as illustrated in the **References** section at the end of this article.

The key observation here is that by suitable *explicit* and *implicit* markup, computer software can attach meaning to strings of characters.

The BibTeX entry specifies the document authors as a list, with author names separated by the word *and* in the string value of the `author` key. In most cases, no additional markup is needed, because BibTeX contains hard-coded algorithms for parsing an author name into four parts: *first*, *von*, *last*, and *junior*. Thus, BibTeX recognizes the name *Ludwig von Beethoven, Sr.* if it is coded as **`von Beethoven, Sr., Ludwig`**, and BibTeX can convert it, if required, to *Beethoven, Ludwig von, Sr.*, and even abbreviate it to *Beethoven, L. von, Sr.* or *von Beethoven, L., Sr.*, depending on format settings in the style file. Name reordering is needed both for citation styles that require it, and also for sorting of reference list entries. Abbreviation of all but the family name is also common in some bibliography styles.

Embedded commas and *and* in author names are handled by minimal extra markup in the form of enclosing braces, such as in the corporate author **`{Chapman and Hall, Ltd.}`**. Some styles convert titles to lowercase letters after the first word, but braces prevent downcasing of proper names and/or German text in titles (recall that nouns in German, and pre-1948 Danish, are always capitalized). BibTeX interprets a macro name at the start of a braced group as a font change, rather than protection against lettercase conversion. An extra level of braces restores the protection, as in the name of a soil-living amoeba popular in biological research: **`{{\em Dictyostelium discoideum}}`**.

BibTeX's processing handles most personal names in Western Europe and the English-speaking world, but it does not deal with the case of Hungarian, and some Asian languages, where the family name comes first: **`Erd\H{o}s P\'al`** and **`K'ung Fu-Tzu`** are usually known as *Paul Erdős* and *Confucius* in English. Abbreviations of Spanish surnames that include both paternal and maternal family names, in that order, like **`Juan Vald{\'e}s Rodr{\'\i}guez`**, are also problematic, since that name can be shortened to *Juan Valdés R.*, or *J. Valdés R.*, or just *J. Valdés*. Some South Indians, and some people from a few other cultural groups, carry only a single name: *Sureshchander* and *Luqi* are examples found in computer-science bibliography articles. The current version of BibTeX, 0.99c, is dated 4 April 1988, but a final version of BibTeX is planned, and its author expects to incorporate additional markup to handle these difficult cases (Patashnik 2003).

Although standard BibTeX styles recognize a dozen or so standard key names, the system permits other keys to be used: their values are simply ignored if the style file does not reference them. Consequently, BibTeX markup has proved *extensible*, and this author has even employed it for representing equipment-inventory records and for editorial tracking of journal-article submissions.

The flexibility of BibTeX key/value markup, and its separation of most of the formatting job into (potentially) user-programmable style files, has proved of great value. The TeX Live 2008 software distribution includes 277 distinct BibTeX style files, and collections of millions of entries in BibTeX markup exist in some communities. The reference list in this article is typeset according to a new style, `acmtrans-v2.bst`, developed by the ACM for their publications in computer science, and as-yet absent from TeX Live. A few publisher Web sites, notably in the areas of computer science, electrical engineering, mathematics, and physics, can return dynamically-constructed BibTeX entries as search results.

3 BibTeX style-file language

BibTeX's style-file language is unique, and it uses reverse Polish notation, like some Hewlett–Packard calcula-

tors, the *Forth* programming language, and the *PostScript*™ page-description language. Such languages are easily parsed with a stack onto which operands are pushed by the lexical scanner, and then popped off the stack during the execution of operators encountered by the scanner. Parsing then requires no lookahead at all, but simply recognition of tokens, one at time, with the parser action either a push or an execute.

In particular, this language design means that functions are called with their arguments already in place on the stack, but the problem is, they could have been put there at any earlier time, and may not be visible at the location of the operator in the source code. This makes errors of incorrect type or number of arguments hard to diagnose, and even harder to find in the source code at some earlier level in the call tree.

While efficient to execute, such languages are horrid for humans to code in, except possibly for tiny programs. Consequently, some researchers have developed other markup schemes and bibliography-style descriptions. The *TUGboat* journal archives contain descriptions of work by James Alexander (Alexander 1986; Alexander 1987), Jean-Michel Hufflen (Hufflen 2003b; Hufflen 2003a), Tristan Miller (Miller 2005), and Robert Burgess (Burgess and Siren 2007). Patrick W. Daly's custom-bib system (Daly 1994; Viton 2000) in the TEX Live distribution generates dozens of BIBTEX style files from a generic file, and provides support for variants in many languages other than English.

Had BIBTEX been developed somewhat later after the importance of scripting languages like awk, JavaScript, lua, perl, php, python, and ruby was better appreciated, it could have been built on those software technologies. Hans Hagen and Taco Hoekwater are now doing this with ConTEXt and lua (Hoekwater 2007), creating a new TEX-like engine named LuaTEX whose internals are much more accessible to the author, document designer, and programmer.

4 BIBTEX grammar and software tools

Lamport's description of BIBTEX in his cited book is informal, but BIBTEX's markup language can be defined with a rigorous grammar, and tools have been developed to check for adherence to that grammar, for parsing BIBTEX files into easier-to-handle data streams, for prettyprinting and syntax checking BIBTEX files, and for converting Web pages from numerous library catalogs and publishers into BIBTEX entries (Beebe 1993a; Beebe 1993b; Beebe 2004a).

A recent, and as-yet-undescribed, software tool, cattobib, builds on international conventions for library-catalog markup and network catalog services to convert catalog entries to rough BIBTEX format (Beebe 2004b). While the conversion is syntactically accurate and generally acceptable to BIBTEX, a certain amount of manual cleanup is almost always required because of irregular and sloppy library cataloging. cattobib searches one or more user-specified catalogs sequentially, and can communicate with large national libraries of several countries. An obvious, although as-yet-unimplemented, extension is to perform the lookups in parallel.

5 The search problem

As the collection of data in BIBTEX markup grows, it becomes important to be able to search that data efficiently. More than a decade ago, this author developed bibsearch (Beebe 1997), leveraging advanced search technology developed by an international team of researchers and described in their *Managing Gigabytes* books (Witten, Moffat, and Bell 1994; Witten, Moffat, and Bell 1999). Their mg search engine stores documents in a highly-compressed in-memory index, and matches a user-supplied list of words, possibly qualified with *and*, *or*, and *not* operators, against the documents. It then returns an ordered list of documents that most closely match the word list. The search is astonishingly fast, often under a millisecond on modern workstations, even for a collection of a half-million BIBTEX entries.

Unfortunately, as so often happens with university research projects, interests changed, students graduated, and software development ceased before mg acquired the necessary feature of subdocument searching. With bibsearch, it is not possible to specify that a search string must, or must not, lie in, say, the `author` field. Such a facility is necessary if one is to quickly identify articles about, say, the *Knuth–Bendix* theorem without also finding articles by those authors themselves. As a result, bibsearch often returns much more than one would like, forcing the human searcher to visually examine a large list of returned BIBTEX entries, or save the results for searching with a text editor, to find the one desired.

BIBTEX was developed about the time that computer networking was evolving from the original Arpanet to what became the Internet, and still several years before the World-Wide Web, and the search engines that began with Archie at McGill University (Emtage and Deutsch 1992), Gopher at the University of Minnesota (Lindner 1993; Anklesaria, McCahill, Lindner, Johnson, Torrey, and Albert 1993), and AltaVista at Digital Equipment Cor-

poration (Seltzer, Ray, and Ray 1997). The first of these is eponymous with a long-running American comic strip (Archie Comics 2008), and was followed later by support tools named after strip companions Veronica and Jughead (Jonzy's Universal Gopher Hierarchical Extracter And Digger, created by a friend and colleague at the University of Utah) (Harris 1993).

At the time of writing this, three large search engines at Google, Microsoft, and Yahoo! dominate the field. They are hosted on thousands of geographically-distributed computers running highly-customized standard operating systems with specialized filesystems. They run Web crawlers to collect files from Web and FTP servers and they handle search queries with massive parallelism and sophisticated search algorithms that combine the history of previous searches with information about the number, and trustworthiness, of links to Web pages from other Web pages.

The search engine companies make their money from targeted advertising, and from fees paid by Web sites to increase the frequency of crawler visits. Their historical stock market performance makes an interesting case study of the successful commercialization of software technology.

The Google search algorithm is called *PageRank*, and the TUG bibliographic archives record more than two dozen publications about it in journals in computer science, mathematics, and oriental languages. Here, we cite only the earliest and latest of them (Page, Brin, Motwani, and Winograd 1998; Brezinski and Redivo-Zaglia 2008). A key component of the PageRank algorithm is the mathematical algorithm known as the *Singular Value Decomposition* (SVD), originally developed in 1965 by the late Gene Golub (1932–2007) at Stanford University, near where Don Knuth would later invent TEX and METAFONT, and by William Kahan at the University of California, Berkeley (Golub and Kahan 1965; Chan, Greif, and O'Leary 2007). These two researchers have strongly influenced modern numerical mathematics, and Kahan is a key figure in the improvement of floating-point arithmetic on computers (Beebe 2007a; Beebe 2007b). Comprehensive BIBTEX bibliographies of their publications can be found in the BibNet Project archives at `http://www.math.utah.edu/pub/bibnet/`.

Many computer users today consider Internet access a critical component of their environment, and routinely use search engines, even for tasks as mundane as finding the nearest designer caffeine-delivery service or where a new movie is shown. While putting BIBTEX files on the Web makes them accessible to Internet search engines, that action does not resolve the issue of how to restrict searches to particular subfields of entries, not does it provide speedy access to new data without substantial cost. The remainder of this article describes a significant solution to the BIBTEX search problem.

6 Relational databases

In their early days, computers had just two main applications: numerical computation, and data processing and management. The human brain stores information, but it is unreliable, has comparatively slow retrieval times, cannot be backed up reliably or quickly to other brains, and eventually, expires. Computers can solve the storage, retrieval, and backup problems if their human companions keep them clean, cool, and supplied with energy, and occasionally migrate their data to newer storage technologies.

Database systems were originally developed in the 1960s, but the early ones were inflexible, and searching could require a sequential scan of the entire corpus of data. However, in 1969, IBM researcher E. F. Codd proposed a new organization, called the *relational model* (Codd 1969; Codd 1970), and that idea eventually revolutionized computer database software.

Codd's essential idea was to consider the data as a set of tables, where each table row is a single logical item. For example, in a parts catalog, one row might contain columns labeled *customer demand*, *description*, *part number*, *price*, *profit margin*, *quantity available*, *supplier*, and *total sales*. Software can then *create a view* of a subset of the data by choosing only certain columns for consideration, with rows possibly further selected by applying constraints like *supplier in or near Madrid* and *price under €100*. Data subsetting can separate public information, such as part number, description, and price that might be recorded in a catalog for customers, from proprietary information that an organization uses for competitive advantage.

If at least one column in each row holds a unique identifier that distinguishes that row from all other rows, then it is possible to *join* tables by creating a new table that appends columns to rows that share a common unique identifier. This operation can be generalized to allow *union* (include all columns), *intersection* (include only common columns), *difference* (include only columns present in the first table and absent from the second), and *symmetric difference* (include only columns present in a single table). These operations are fundamental in the mathematical field of *set theory*, once introduced in American schools as "New Math" (Kline 1973). That idea was delightfully parodied by Tom Lehrer in a song of that name (Lehrer 1965), and later brought to the London (1980)

and New York (1981) stages in the play *Tomfoolery* (Macintosh 1980). For those interested in the mathematical background of relational databases, there is a recent book on that subject (de Haan and Koppelaars 2007).

It is precisely the idea of a unique identifier that provides great power for the collection and management of huge sets of data. While part numbers are possibly innocuous, personal identifiers like biometrics (blood groups, DNA sequences, facial photographs, fingerprints, retinal scans, voice patterns, ...), taxpayer numbers, pension-plan numbers, credit-card numbers, vehicle-license numbers, telephone numbers, house numbers, e-mail addresses, highway toll e-passes, and so on have made it possible for governments and commercial organizations to amass and collate enormous amounts of information about most of the population of the developed world, with a complete loss of privacy (Rambam 2008). The database *join* operation allows this material to be collected and sifted so that, for example, a medical-insurance company might maximize profits by insuring only customers who are both healthy and wealthy. The dangers of collecting information about people are well-chronicled in the book *IBM and the Holocaust* (Black 2002), and any study of human history soon shows that power and victory often go to those with the best information technology.

7 Structured Query Language (SQL)

By the mid-1980s, it became clear that, at least for users, standardization of the search interface to database systems was desirable. Vendors eventually settled on what is now known as the *Structured Query Language*, usually abbreviated as SQL, and pronounced either by its letters (ess cue ell) or like the noun *sequel*.

There have been several national, government, international, and industry standards for SQL, including ANSI (X3.168-1989, X3.135-1992, 9579-2-1993, 9075-3-1995, and 9075-4-1996), FIPS (127:1990, 127-2:1993, 193:1995), ISO/IEC, (9075:1987, 9075:1989, 9075:1992, 9075:2003, and 13249:2007), and X/Open (CAE 1994) (X/Open 1994). Many 2008-vintage SQL systems claim conformance to most of the 1992 ISO Standard. There are more than 500 books about SQL, and BibTeX entries for many of them can be found at `http://www.math.utah.edu/pub/tex/bib/index-table-s.html#sqlbooks`. The prolific writer, Christopher J. Date, has produced dozens of books about databases, and four editions of one of them chronicle the development of SQL (Date 1987; Date 1989; Date and Darwen 1993; Date and Darwen 1997), and eight editions of another cover database systems in general (Date 2004). There are also very readable treatments of the use of SQL in the books by Judith Bowman and her co-authors (Emerson, Darnovsky, and Bowman 1989; Bowman, Emerson, and Darnovsky 1993; Bowman, Emerson, and Darnovsky 1996; Bowman, Emerson, and Darnovsky 2001; Bowman 2001). For deeper coverage of database theory and practice, see the O'Neils' treatise (O'Neil and O'Neil 2001).

The language SQL offers much to criticize, and Ted Codd was never happy with it. One of members of the ISO SQL Standard Committee wrote this about it (Date and Darwen 1993, p. 374):

> I lay most of the blame for the following problems firmly on SQL itself. If the world needs a better database standard, it should prepare itself to move to a better database *language*.

Everyone who has searched an online bookstore or library catalog has almost certainly used SQL, but that use is generally hidden behind a fill-in-the-box screen interface that destroys much of the true power of the language.

Three observations about SQL are relevant for our discussion here:

- The **S** in **SQL** is for *Structured*, not *Standard*. Life with SQL is very different from life with conventional programming languages, like Fortran, C, C++, C#, and Java, for which highly-portable subsets and international standards exist.

  With SQL, there are far too many gratuitous and irritating differences between database systems that serve no purposes but customer lock-in, database programmer job security, and user confusion.
- SQL is *not* a programming language: there are no variables, no loops, no conditionals, and no user-defined functions or procedures, although particular databases may offer some of these extended features. When programmability is required, it is conventional to embed calls to an SQL interface library, either in a high-level compiled language like Ada, C, C++, C#, COBOL, or Java, or in a scripting language, such as JavaScript, perl, php, python, or ruby, all of which have database module interfaces.
- SQL is complex, large, and powerful, but it is not difficult to learn a useful tiny subset: indeed, in Section 10, we concentrate primarily on just *one* statement in the language.

8 BibTeX database tables

The first step in making BibTeX entries amenable to storage in a relational database is to identify column headings that are likely to be useful. All that we need to do is to mentally rotate the entry by 90 degrees, so that the entry keys

become column names, their values become table-cell data, and the citation label becomes a (possibly-)unique identifier characterizing the table row. Because a BibTeX file often contains string definitions for use in entries, they too are fodder for another table.

At the time of writing this, we have implemented three distinct tables for BibTeX data:

`bibtab` with data from BibTeX entries under these columns:

`authorcount`, `editorcount`, `pagecount`, `bibtype`, `filename`, `label`, `author`, `editor`, `booktitle`, `title`, `crossref`, `journal`, `volume`, `type`, `number`, `institution`, `organization`, `publisher`, `school`, `address`, `edition`, `pages`, `day`, `month`, `monthnumber`, `year`, `CODEN`, `DOI`, `ISBN`, `ISBN13`, `ISSN`, `LCCN`, `MRclass`, `MRnumber`, `MRreviewer`, `bibdate`, `bibsource`, `bibtimestamp`, `note`, `series`, `URL`, `abstract`, `keywords`, `remark`, `subject`, `TOC`, `ZMnumber`, `entry`.

`strtab` a table of BibTeX string definitions with three columns:

`key`, `value`, `entry`.

`namtab` a table of personal names used in `author` and `editor` fields, with just two columns:

`name`, `count`.

Cells in the `*count` columns have integer values, and the cells in all of the remaining columns contain variable-length character strings, where the permitted string lengths are well beyond what is needed for typical BibTeX entries.

The `entry` column in the first two tables contains the *exact* original text from the BibTeX file.

It is conventional in modern database systems to use a special value, *NULL*, to represent indeterminate, unknown, or unset values. NULL is distinguishable from both numeric zeros and empty strings, and most database operators, and functions that act on column data, ignore rows with NULL values. Thus, if you take the sum or average of a numerical column, only rows with numbers in them participate. Similarly, if you select data with an expression **`year > '2005'`**, rows with a NULL year do not match that expression.

The query facilities described in Section 10 allow testing for the presence or absence of NULL values with expressions like ***`column`*** **`IS NULL`** and ***`column`*** **`IS NOT NULL`**.

While most of the key names in the `bibtab` table are standard ones in BibTeX, a few are not. These include at least these keys:

`CODEN` Chemical Abstracts *CODE Name* for serial publications.

`DOI` *Digital Object Identifier*, a *unique and permanent* name for the master copy of an electronic document. If it does not look like an Internet URL, convert it to one by prefixing it with `http://dx.doi.org/`.

`ISSN` *International Standard Serial Number*: eight digits, optionally written as two groups of four digits separated by a hyphen.

`ISBN` old-style *International Standard Book Number*: 10 digits, the last of which may be *X* or *x*. Optional hyphens separate it into four digit groups that define the country or language, publisher, book number, and a final check digit for error detection.

`ISBN13` new-style (since 2007) *International Standard Book Number*: 13 digits, the last of which may be *X* or *x*. This contains the old ten-digit ISBN, with a new prefix of `978-`, and a revised check digit. When the supply of unused ISBNs is exhausted, a new prefix `979-` will be assigned, and such numbers will then not be convertible to the old ten-digit form.

`LCCN` US *Library of Congress Catalog Number*, used for book identification at many US libraries, and at some libraries in other countries.

`MRclass` list of American Mathematical Society *Math Reviews* five-character subject classification codes.

`MRnumber` *Math Reviews* database number.

`MRreviewer` *Math Reviews* reviewer name.

`bibdate` Unix date string in the formats `Sat Oct 25 20:15:00 MDT 2008` or `Sat Oct 25 20:15:00 2008`; it records a timestamp of the last significant change to the BibTeX entry data.

`bibsource` arbitrary text indicating source(s) of bibliographic data, often as a list of Internet URLs or Z3950 library-catalog addresses.

`bibtimestamp` alternate representation of `bibdate` value in the form `2008.10.25 20:15:00 MDT`, for which lexicographic sort order is also time order.

URL	Internet *Uniform Resource Locator(s)* for retrieval of an electronic form of the document.
remark	additional commentary about the entry that is not intended to be included in a typeset reference list.
subject	subject classification phrases.
TOC	table of contents of the publication.
ZMnumber	European Mathematical Society *Zeitschrift für Mathematik* database number.

9 BibTeX/SQL software tools

This author has created two new programs, bibtosql and bibsql, to get BibTeX files into relational databases with an SQL interface and make that data searchable. The programs are freely available in various distribution formats at `http://www.math.utah.edu/pub/bibsql`. Both are written as portable shell scripts, and the first of them uses another program in the awk scripting language, which is standardly available on all Unix platforms, and versions can be found for Microsoft Windows as well. The awk program needs no changes whatever to move between systems, and at a given site, it is likely that the shell scripts, once configured, are identical across local systems.

The first program, bibtosql, handles the conversion of BibTeX files to input for one of three commonly-used, and freely-available, SQL database systems: *MySQL*, *PostgreSQL*, and *SQLite*. All three are reasonably portable, and they can be installed from source code, or from binary distributions, on all common desktop platforms. In 2007, Sun Microsystems purchased MySQL, but the source code is expected to remain available. We also investigated use of the *IBM DB2*, *Ingres*, and *Microsoft SQL Express* databases, but abandoned further work with them because their limits on the sizes of strings or table rows make them unusable for our needs, and they are not available on the broad range of platforms that the others are. Also, their SQL input syntax is somewhat different from the three that we support, and that would introduce unnecessary confusion for users of bibsql.

MySQL and PostgreSQL are large and complex systems that implement a client/server model of database access. A single server can typically handle simultaneous requests from many client programs elsewhere on the network. Large sites may already have one or more instances of them installed and operational, making it easy to create yet another database for them to serve.

The third is much simpler, and recommended for single-user systems, and small groups. An SQLite database consists of a *single file* that is independent of operating system, CPU architecture, and storage byte order. Once created, an SQLite database file can be used on any system where SQLite can be compiled, and is ideally suited for distribution on read-only media such as CD-ROMs or DVDs, or via the Web.

The SQLite software is *public domain* and may be used for any purpose, and redistributed, without restriction. This makes it of interest, for example, for future TeX Live distributions.

The user interface to bibtosql looks like this:

```
% bibtosql --help
Usage: /usr/local/bin/bibtosql
                [ --author ]
                [ --create ]
                [ --database dbname ]
                [ --help ]
                [ --version ]
                [ --server ( MySQL | psql | PostgreSQL | SQLite ) ]
                [ -- ]
                BibTeXfiles or <infile
                >outfile

Options may begin with either one or two dashes, and lettercase in
options and database names is ignored.

Examples:
        bibtosql --create *.bib                       | sqlite3 bibtex.db
        bibtosql --create --server sqlite *.bib       | sqlite3 bibtex.db
        bibtosql --create --server psql *.bib         | psql ...
        bibtosql --create --server mysql *.bib        | mysql ...
```

```
bibtosql -c *.bib                              | sqlite3 bibtex.db
bibtosql -c -s p *.bib                         | psql ...
bibtosql -c -s m *.bib                         | mysql ...

bibtosql newstuff.bib                          | sqlite3 bibtex.db

bibclean *.bib | biborder | bibtosql --create | sqlite3 bibtex.db
```

The manual pages for bibtosql contain extensive details about setting up the supported databases, and they need not be repeated here. SQLite is by far the easiest to use, since a single command of the type shown in the first example does the job.

The second program, bibsql, provides the user interface to the SQL database system, and hides the messy details of where the data are located, and how the SQL client program communicates with the server. Its help option reports:

```
% bibsql --help
Usage: /usr/local/bin/bibsql
                [ --author ]
                [ --command ' command1; command2; ... ' ]
                [ --database dbname ]
                [ --help ]
                [ --options ' ... server options ...' ]
                [ --server ( MySQL | psql | PostgreSQL | SQLite ) ]
                [ --user dbuser ]
                [ --version ]

Options may begin with either one or two dashes, and lettercase in
options and database names is ignored.

Examples:
        bibsql
        sqlite> ... user input here ...

        bibsql -s psql
        psql>  ... user input here ...
```

The default database is currently called **bibtex**, but the option is there to allow access to others, should that prove useful.

At this author's site, SQL clients for all three database systems are operational on about 25 flavors of Unix, including Apple Mac OS X, covering all of the common CPU architectures of the last decade. This demonstrates the impressive portability of these database systems. The Cygwin environment on Microsoft Windows XP and Vista systems provides clients for PostgreSQL and SQLite; see `http://www.cygwin.com/`. MySQL for Windows can be downloaded from its vendor Web site, and its client program runs nicely with Cygwin. All three servers supply essentially the same data, but they vary in performance and also in search and display facilities. The default server is always SQLite, but users of bibsql should have few surprises if they select a different database server with a command-line option.

The bibsql distribution includes sample C programs that can communicate with one of each of the supported SQL databases, providing a useful coding template for those who wish to incorporate bibliographic-database access directly into their own programs.

10 Using SQL with bibsql

It is now time to look at exactly how searches are done in SQL. The language has an imperative English-like grammar that is nevertheless precise enough to allow unambiguous interpretation by computer software. Once bibsql has started the SQL client program, all further user interaction is with that program: there is no input filtering whatever.

For user convenience, lettercase is ignored in SQL commands. Lettercase is ignored in search strings as well, except with PostgreSQL, where nonstandard special operators are required.

Character strings in SQL are written with surrounding single quotes, like this: **'SQL'**. Some SQL clients also support quotation-mark delimiters, **"SQL"**.

To represent a single quote inside a string, double it: **'O''Neil'** is the name *O'Neil*. In MySQL, use a backslash escape instead: **'O\'Neil'**.

Long strings can be split into separate strings that are joined with the double-bar concatenation operator. For example, these two expressions evaluate to the same string:

```
'Aloisius Baldwin Chadwick, IV'
'Aloisius ' || 'Baldwin ' || 'Chadwick, ' || 'IV'
```

Comments in SQL take two forms. An Ada-style double hyphen starts a remark that continues to end of line or end of file, whichever comes first. MySQL additionally requires that the double hyphen be followed by at least one space to eliminate an ambiguity in the SQL expression grammar. Otherwise, C-style `/* ... */` comments can span one or more lines. Comments cannot be nested, and can appear inside commands anywhere that whitespace can (except inside character strings).

The format of the output depends on the database system. For SQLite, it can be set by a dotted command described in the help system like this:

```
.mode MODE ?TABLE?  Set output mode where MODE is one of:
                      csv    Comma-separated values
                      column Left-aligned columns.  (See .width)
                      html   HTML <table> code
                      insert \SQL\ insert statements for TABLE
                      line   One value per line
                      list   Values delimited by .separator string
                      tabs   Tab-separated values
                      tcl    TCL list elements

.nullvalue STRING   Print STRING in place of NULL values

.output FILENAME    Send output to FILENAME

.output stdout      Send output to the screen

.separator STRING   Change separator used by output mode and .import

.width NUM NUM ...  Set column widths for "column" mode
```

The default is to separate cell values by vertical bars, with each table row output on a single line (unless the string data contain linebreaks). Examples are given later in this section. However, other output styles chosen by the SQLite `.mode` command make it easy to output the data in formats suitable for input to other databases, spreadsheets, and Web pages.

The simplest query asks for a return of all records, where the asterisk means all data, and a final semicolon is required to terminate the command:

```
select * from bibtab;
1||9|article|acm-turing-awards.bib|Perlis:1967:SAS|
Alan J. Perlis|||The Synthesis of Algorithmic Systems||
j-J-ACM|14||1|||||||1--9||jan|1|1967|JACOAH|
http://doi.acm.org/10.1145/321371.321372|||0004-5411 OR 00045411|
||||Mon Dec 05 19:37:58 1994||1994.12.05 19:37:58 ???|
|||||This is the 1966 ACM Turing Award Lecture, and the first award.||||
@Article{Perlis:1967:SAS,
  author =       "Alan J. Perlis",
  title =        "The Synthesis of Algorithmic Systems",
  journal =      j-J-ACM,
```

```
  volume =       "14",
  number =       "1",
  pages =        "1--9",
  month =        jan,
  year =         "1967",
  CODEN =        "JACOAH",
  DOI =          "http://doi.acm.org/10.1145/321371.321372",
  ISSN =         "0004-5411",
  bibdate =      "Mon Dec 05 19:37:58 1994",
  acknowledgement = ack-nhfb,
  remark =       "This is the 1966 ACM Turing Award Lecture, and the
                 first award.",
}
...
```

Here, we wrapped the long first line for readability, and showed only the first record returned. The order of records depends on the database creation and update history, and is unpredictable without further specifications.

Next, we limit the output to just three specified columns, and further limit the selection with a `WHERE` clause:

```
select year, author, title from bibtab
    where author like '%Perlis%' and year = '1967';
1967|Alan J. Perlis|The Synthesis of Algorithmic Systems
1967|B. A. Galler and A. J. Perlis|A proposal for definitions in ALGOL
```

Because the command is long, we wrote it on separate lines.

The `LIKE` operator is followed by a string pattern wherein *percent* represents zero or more characters, *underscore* a single character, and, in MySQL and SQLite, lettercase is *ignored*. Use `NOT LIKE` to negate the comparison. These matches are against the full text of the string, so patterns usually need to begin and end with a percent.

To search for a literal percent or underscore, double them in the search pattern. With PostgreSQL, use the nonstandard operator `ILIKE`, or its synonym `~~*`, for case-insensitive searches.

To make string comparisons case sensitive in SQLite, set a library option like this:

```
pragma case_sensitive_like = on;
```

Set it to `off` to restore the default behavior. SQLite recognizes synonyms `true`, `yes`, and `1` for `on`, and `false`, `no`, and `0` for `off`.

SQLite also has a `GLOB` operator that uses Unix pathname matching, where *asterisk* matches zero or more characters, *question mark* matches a single character, and lettercase is always *significant*. Unfortunately, there is no standard support in SQLite for regular-expression matching like that provided by many other Unix tools and scripting languages, and some other SQL systems.

We can also use string-equality tests, but then the match must be exact, including lettercase:

```
select year, author, title from bibtab
    where author = 'Alan J. Perlis'
    order by year;
1958|Alan J. Perlis|Announcement
1963|Alan J. Perlis|Computation's development critical to our society
1967|Alan J. Perlis|The Synthesis of Algorithmic Systems
1969|Alan J. Perlis|Introduction to extensible languages
1978|Alan J. Perlis|The American side of the development of Algol
1986|Alan J. Perlis|Two Thousand Words and Two Thousand Ideas --- The 650 at Carnegie
```

In the `ORDER BY` clause, the operand can be a comma-separated list of column names or ordinal numbers: **`ORDER BY 1, 3`** sorts by the first column, and when that column has the same values, by the third column.

Such a search is likely to miss many entries belonging to alternate spellings of the selected author, such as:

```
select year, author, title from bibtab
    where author = 'A. J. Perlis'
    order by year;
1964|A. J. Perlis|A format language
```

```
1964|A. J. Perlis|Programming of digital computers
1964|A. J. Perlis|How should ACM publish computer research?
1966|A. J. Perlis|A Forum on Algorithms: A new policy for algorithms?
1975|A. J. Perlis|Introduction to Computer Science
1981|A. J. Perlis|The American side of the development of ALGOL
```

It also misses entries where Perlis is one of multiple authors, since SQL string-equality tests always compare against the full string values.

Use the namtab table to find the frequencies and variations of an author or editor name in the database:

```
select count, name from namtab
    where name like '%Steele%'
    order by 1 desc;
```

```
15|Guy L. Steele Jr.
3|Guy L. Steele
2|Guy L. Steele, Jr.
1|G. L. Steele, Jr.
1|G. Steele
```

A more complex query requests unique output from a range of years, sorts the data in descending-year order, and limits the number of records returned by the command to just five:

```
select distinct year, author, title from bibtab
    where author like '%D%Knuth'
    and '1955' < year
    and year < '1970'
    order by year desc
    limit 5;
```

```
1969|Donald E. Knuth|Seminumerical Algorithms
1968|Donald E. Knuth|Very magic squares
1967|Donald E. Knuth|The Remaining Trouble Spots in ALGOL 60
1966|Donald E. Knuth|Errata: ‘‘Additional comments on a problem in concurrent ...’’
1966|Donald E. Knuth|Letter to the Editor: Additional comments on a concurrent ...
```

Some SQL systems permit the quotes around the year values to be omitted, but strictly, they are strings, not integers, since they occasionally contain a list or range of years.

The expression

```
'1955' < year and year < '1970'
```

can also be written as

```
year between '1956' and '1969'
```

The endpoints of the BETWEEN operator are included in the range test.

When multiple logical operators are used in an expression without disambiguating parentheses, AND is evaluated before OR. Thus, **a and b or c and d** is treated as if it were written **(a and b) or (c and d)**. When in doubt about the meaning of a complex expression, parenthesize!

The SELECT command can be used for rudimentary expression evaluation in SQL, simply by omitting cell selections, as shown in Table 1. Notice that character-string collation differs among the three systems. Numerical expressions are evaluated in floating-point arithmetic if at least one of the operands contains a decimal point, except in MySQL. Although the underlying arithmetic is IEEE 754, the treatment of NaN, Infinity, subnormal numbers, and zero divides is irregular and unreliable in the three systems. Also, large numbers are not handled sensibly. In the view of this author, floating-point arithmetic is not to be trusted in these systems.

The larger SQL systems offer a wide range of numerical and string functions, similar to those of many programming languages. However, SQLite has only a minimal repertoire of built-in functions. Here are examples of some of the SQL functions that can operate on the returned results, including ones that just report counts, averages, extrema, and sums. We pose them as questions in prose, then express them in a SELECT statement to retrieve an answer.

Table 1: Expression evaluation in SQL systems. SQLite lacks the illustrated functions from the square root to end of table.

SQL statement	Value	Remark
`select 'ABC' > 'DEF';`	0	
`select 'ABC' < 'ABCDEF';`	1	
`select 'a' < 'A', 'a' = 'A', 'a' > 'A';`	0,1,0	MySQL
`select 'a' < 'A', 'a' = 'A', 'a' > 'A';`	t, f, f	PostgreSQL
`select 'a' < 'A', 'a' = 'A', 'a' > 'A';`	0,0,1	SQLite
`select length('The Lion of TeX');`	15	
`select substr('The Lion of TeX', 13, 3);`	'TeX'	
`select 1.0 / 3.0;`	0.333333333333333	
`select 1.0 / 3;`	0.333333333333333	
`select 1 / 3;`	0	
`select 1 / 3;`	0.3333	in MySQL
`select 1/3 - 0.33333333333333333;`	−0.00000000033333333	in MySQL
`select 1.0/3.0 - 0.33333333333333333;`	0	in MySQL
`select null IS NULL;`	1	
`select max(1,2,3);`	3	SQLite only
`select min(1,2,3);`	1	SQLite only
`select (1 + sqrt(5))/2;`	1.6180339887499	
`select sin(22);`	−0.00885130929040388	
`select pi();`	3.14159265358979	
`select rand();`	0.54459485182967	in [0,1] in MySQL
`select random();`	0.751422385685146	in [0,1] in PostgreSQL
`select random();`	−5624361857185364588	in $[-2^{63}, 2^{63}-1]$ in SQLite
`select log(100);`	4.6051701859881	natural logarithm in MySQL
`select log10(1000);`	3	base-10 logarithm in MySQL
`select ln(10);`	4.60517018598809	natural logarithm in PostgreSQL
`select log(100);`	2	base-10 logarithm in PostgreSQL

How many BibTeX entries are in the database?

```
select count(*) from bibtab;
417349
```

How many titles are longer than 250 characters?

```
select count(length(title)) from bibtab where length(title) > 250;
772
```

What fraction of the entries have titles longer than 250 characters?

```
select '1 / ' || round(417349 / 772);
1 / 540.0
```

What is the average length of nonempty titles?

```
select avg(length(title)) from bibtab where length(title) > 0;
61.3055823817683
```

How long is the longest title in the database?

```
select max(length(title)) from bibtab;
1746
```

What is the average length in pages, rounded to the nearest integer, of the documents with known page counts?

```
select round(avg(pagecount)) from bibtab where pagecount > 0;
46.0
```

What is the largest page count in the database?

```
select max(pagecount) from bibtab;
4412
```

What is the largest number of editors in any document?

```
select max(editorcount) from bibtab;
18
```

What is the largest number of authors in any document?

```
select max(authorcount) from bibtab;
115
```

That number is unexpectedly large. *What is the citation label and title of that document?*

```
select label,title from bibtab where authorcount = 115;
Adiga:2002:OBS|An Overview of the BlueGene/L Supercomputer
```

Evidently, it takes a lot of people to build IBM's largest computer.

What is the largest number of authors of any book?

```
select max(authorcount) from bibtab where bibtype = 'book';
23
```

How small is the shortest BibTeX entry?

```
select min(length(entry)) from bibtab;
101
```

How big is the longest BibTeX entry?

```
select max(length(entry)) from bibtab;
19269
```

What is the average size of a BibTeX entry in the database?

```
select avg(length(entry)) from bibtab;
770.447505564887
```

How many pages are in the documents that have known page counts?

```
select sum(pagecount) from bibtab where pagecount > 0;
13415587
```

What are the numbers of documents in each month of publication?

```
select distinct count(month), lower(month) from bibtab
       where length(month) = 3
       group by lower(month)
       order by cast(monthnumber as number);
19111|jan
```

```
15411|feb
20423|mar
19398|apr
17489|may
21585|jun
19537|jul
16577|aug
20039|sep
18896|oct
16859|nov
20924|dec
```

This example shows how strings can be converted to lowercase with the **lower()** function (there is also an **upper()** function), and to numbers with the **cast()** function. It may also suggest that publisher staff prefer February, August, and November for their vacation time.

How many documents are there for each of the top ten publishers, ordering the results by descending counts?

```
select count(publisher), publisher from bibtab
    where length(publisher) > 0
    group by publisher
    order by count(publisher) desc
    limit 10;
```

```
5679|pub-SV
2154|pub-ORA
1820|pub-PROJECT-GUTENBERG
1593|pub-IEEE
1307|pub-AW
1076|pub-PH
891|pub-WILEY
838|pub-ACM
524|pub-PHPTR
425|pub-MCGRAW-HILL
```

How many documents are there for each of the top ten journals, ordering the results by descending counts?

```
select count(journal), journal from bibtab
    where length(journal) > 0
    group by journal
    order by count(journal) desc
    limit 10;
```

```
60788|j-LECT-NOTES-COMP-SCI
14540|j-J-MATH-PHYS
9988|j-CACM
8594|j-APPL-MATH-COMP
7896|j-SIGPLAN
7253|j-LINEAR-ALGEBRA-APPL
6435|j-THEOR-COMP-SCI
5344|j-COMPUTER
5335|j-MATH-COMPUT
4974|j-INFO-PROC-LETT
```

How many documents have a nonzero author count? (Remember that some document entries have only editors.)

```
select count(*) from bibtab where authorcount > 0;
```

```
406451
```

What percentage of the documents have one, two, . . . , five authors?

```
select round(100 * count(authorcount) / 406451) || '%', authorcount from bibtab
    where authorcount > 0
    group by authorcount
    order by count(authorcount) desc
    limit 5;
```

```
48.0%|1
28.0%|2
13.0%|3
5.0%|4
1.0%|5
```

The rule that NULL values are excluded from consideration in expressions does not apply to the special case of the **count(*)** expression, because it just counts rows, and no row is completely NULL.

The last four searches are complex queries that use the SQL GROUP BY feature to simultaneously compute counts of publishers and journals, producing a list of the top ten that occur most frequently in the database, and do a similar computation to find the percentage of publications with one to five authors.

To avoid the need to first compute the count of documents with authors, the last pair of queries can be rephrased as a single command with an embedded SELECT command, but we now make a further restriction.

What is the percentage of journal articles that have each of one to five authors?

```
select round(100 * count(authorcount) /
             (select count(*) from bibtab
                     where authorcount > 0 and
                           bibtype = 'article')) || '%',
           authorcount
    from bibtab
    where authorcount > 0 and
          bibtype = 'article'
    group by authorcount
    order by count(authorcount) desc
    limit 5;
```

```
47.0%|1
29.0%|2
14.0%|3
5.0%|4
1.0%|5
```

BIBTEX string abbreviations are commonly used for data that are repeated in many entries, particularly for journals, institutions, organizations, publishers, and addresses. The abbreviations can be retrieved from the strtab table with searches like these:

Which publishers are in the Wiley family?

```
select entry from strtab
    where key like 'pub-WILEY%'
      and key not like '%adr%';
```

```
@String{pub-WILEY               = "Wiley"}
@String{pub-WILEY-INTERSCIENCE  = "Wiley-In{\e-}ter{\e-}sci{\e-}ence"}
```

Which publishers have offices in the city of Boston?

```
select entry from strtab where value like '%Boston%' order by entry;
```

```
@String{pub-ALLYN-BACON:adr     = "Boston, MA, USA"}
```

```
@String{pub-AP-PROFESSIONAL:adr = "Boston, MA, USA"}
...
@String{pub-LITTLE-BROWN:adr    = "Boston, Toronto, London"}
@String{pub-MORGAN-KAUFMANN:adrbo = "Boston, MA, USA"}
```

How many of the document publishers are in the city of New York?

```
select count(value) from strtab where value like '%New York%';
59
```

What are the string names and addresses for publishers in the Birkhäuser family?

```
select distinct key from strtab
    where key like '%BIRK%'
    order by key;
pub-BIRKHAUSER
pub-BIRKHAUSER-BOSTON
pub-BIRKHAUSER-BOSTON:adr
pub-BIRKHAUSER:adr
```

It is possible, although complex, to combine searches in multiple tables to collect output data from all of them. Here is just one example, which also introduces another SQL feature of providing short aliases for database and table names within a single query.

Can we find the strings and BibTeX entry for the Classic Shell Scripting *book?*

```
.separator "\n"
select s.entry, t.entry, b.entry
       from bibtab b, strtab s, strtab t
       where b.author like '%Robbins%'
       and b.title like '%classic%'
       and b.publisher = s.key
       and b.address = t.key;
@String{pub-ORA-MEDIA           = "O'Reilly Media, Inc."}
@String{pub-ORA-MEDIA:adr       = "1005 Gravenstein Highway North,
                                   Sebastopol, CA 95472, USA"}
@Book{Robbins:2005:CSS,
  author =       "Arnold Robbins and Nelson H. F. Beebe",
  title =        "Classic Shell Scripting",
  publisher =    pub-ORA-MEDIA,
  address =      pub-ORA-MEDIA:adr,
  pages =        "xxii + 534",
  year =         "2005",
  ISBN =         "0-596-00595-4",
  ISBN-13 =      "978-0-596-00595-5",
  LCCN =         "QA76.76.063 R633 2005",
  bibdate =      "Tue Jul 12 16:13:16 2005",
  URL =          "http://www.oreilly.com/catalog/shellsrptg/",
  acknowledgement = ack-nhfb,
}
```

Changing the output separator to a newline suppresses unwanted additional decoration, producing output that can be copied directly into a BibTeX file.

To list the column fields in the SQLite database, and to get further help, use the commands

```
.schema bibtab
.help
```

The output of `.schema` is similar to the `CREATE TABLE` command shown in the manual pages for bibtosql.

For more on the command syntax of SQLite, consult its Web site documentation collection: `http://www.sqlite.org/docs.html`.

More details on additional features of MySQL and PostgreSQL searching and output formatting are provided in the manual pages for bibsql. Here, we only note that both provide for regular-expression pattern matching, and allow lettercase to be significant, or to be ignored. Because key is a built-in function in MySQL, when used as a column name in MySQL search expressions, it must be protected with back quotes, like this: `` `key` ``.

All three client programs record a command-history file in the user's home directory, and support the GNU readline library for command recall and editing. This feature is of great convenience, especially because it is common to repeat search commands after minor edits.

11 The impact of Ted Codd and Jim Gray

Since 1966, the *Association for Computing Machinery* (ACM) has awarded an annual prize for significant work that furthers the young field of computer science. This is known as the *Turing Award*, and it has come to be regarded as a kind of Nobel Prize for computer scientists. It is named after the renowned English mathematician Alan Mathison Turing, who produced early important work on artificial intelligence, complexity theory, the now-famous *Turing Machine* test, computer design, cryptography, and floating-point arithmetic, as well as in pure mathematics and mathematical biology. BibTeX biographies of Turing's complete works, and of the ACM Turing Awards, are available in the previously-cited BibNet Project and in the TUG bibliography archives at `http://www.math.utah.edu/pub/tex/bib/index-table-a.html#acm-turing-awards`.

The first award went to Alan Perlis (1922–1990) of Yale University for his important work on compiler construction and programming languages, notably, Algol. Some of his papers are mentioned in the search examples in Section 10.

The 1981 recipient was the English computer scientist Edgar Frank "Ted" Codd (1923–2003), who revolutionized computing with the idea of relational databases. He spent much of his career at IBM research laboratories in the USA, where he was influential in the development of IBM's *System R* database. One of the spinoffs of that work was the founding by Larry Ellison of Oracle Corporation, today the largest database vendor, and a fierce competitor of others in that field, particularly Microsoft. There is a tribute to Ted Codd in one of the chapters of a recent memorial volume (Date 2008), which notes in its first paragraph: *The entire relational database industry, now worth many billions of dollars a year, owes its existence to Ted's original work.* An earlier version appeared in a journal article (Date 2003), and there is a Web site containing pointers to his publications (McJones 2003).

The 1998 award went to James Nicholas "Jim" Gray (1944–2007), who worked in academia, and in industry at IBM, Tandem, DEC, and Microsoft. He was the first recipient of a Ph.D. from the Department of Computer Science at the University of California, Berkeley, a department that is one of three great leaders in the field (the others are MIT and Stanford), and the one that brought Unix and VAXes into academia. Jim was tragically lost without a trace on January 28, 2007, during a solo sailing trip in clear and calm weather to the Farallon Islands offshore west of San Francisco, California. A large satellite and sea rescue effort failed to find either him, or his boat, or any evidence whatever of what happened; see `http://openphi.net/tenacious/`.

In the view of many people in the field, Jim was the person who made relational databases really work. He made important contributions by applying database technology in the Microsoft Virtual Earth Project, and the SkyServer database for the Sloan Digital Sky Survey (SDSS) (see `http://www.sdss.org/`, `http://skyserver.sdss.org/`, and `http://doi.acm.org/10.1145/1400214.1400231`). SkyServer has revolutionized the field of astronomy, because it makes the large and expensive-to-create astronomical data archives freely available to all of the world's astronomers and astrophysicists, even those who are amateurs or hobbyists.

A conference in memory of Jim Gray was held at Berkeley on 31 May 2008 (see `http://www.eecs.berkeley.edu/IPRO/JimGrayTribute/`). Its proceedings are recorded in the June 2008 issue of ACM *SIGMOD Record*; see `http://www.math.utah.edu/pub/tex/bib/index-table-s.html#sigmod`. The November 2008 issue of *Communications of the ACM* has a charcoal-gray portrait of Jim on its cover, and articles about his work; see `http://portalparts.acm.org/1410000/1400214/fm/frontmatter.pdf`.

12 Conclusion

Connecting BibTeX to SQL databases has been a long-term goal of mine, because the output of bibsearch is often overwhelming due to its inability to narrow the search. However, bibsearch is much faster than any SQL database, and my concern was that an SQL interface to bibliographic data might be too sluggish to be useful. Fortunately,

advances in computers have been dramatic in the years since BIBTEX was first developed. With the database server running on a fast modern multicore or multiprocessor system, and particularly with SQLite and MySQL, search speeds with bibsql are acceptably fast, and often under a second.

The ability to restrict searches to subfields of BIBTEX entries is liberating. It facilitates asking questions that could not be answered with previous tools, as we illustrated in some of the search examples in Section 10. It also makes it much easier, and more reliable, to collect new entries for the many subject-specific bibliographies in the TUG archive, since the `WHERE` clause can use expressions that test the `bibtimestamp` and `filename` columns to select entries newer than a given date, while ignoring entries already in the subject bibliography file.

Exploratory searches of our large body of bibliographic data have also shown that it is quite easy to detect, report, and repair irregularities in the data, even down to such simple things as a missing period following an author/editor initial.

My hope is that future editions of the TEX Live software distributions will include bibsql and bibtosql, along with binaries for SQLite for all supported platforms, accompanied by URLs from which to retrieve current collections of bibliographic data in SQLite format. That way, TEX users the world over can enjoy the benefits of fast access, via SQL searches, to bibliographic data without the trouble of producing and maintaining that data themselves. The SQL learning experience with bibsql may then encourage them to think of other applications of SQL databases for their own interests and problems, and appreciate the lessons from Ted Codd and Jim Gray that relational databases are *really useful*.

The BIBTEX entries for the publications cited in the **References** section that follows were collected almost entirely with the help of bibsql, bibsearch, and cattobib.

References

James Alexander. 1986. Tlb: a reference setting package for TEX. *TUGboat* 7, 3 (Oct.), 138–140. ISSN 0896-3207.

James Alexander. 1987. Tlb: a reference setting package, update. *TUGboat* 8, 2 (July), 102–102. ISSN 0896-3207.

F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. Torrey, and B. Albert. 1993. RFC 1436: The Internet Gopher Protocol (a distributed document search and retrieval protocol). (Mar.). See `ftp://ftp.internic.net/rfc/rfc1436.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc1436.txt`.

Archie Comics. 1941–2008. Archie Archive. Web site. See `http://www.archiecomics.com/`.

Nelson H. F. Beebe. 1993a. Bibliography prettyprinting and syntax checking. *TUGboat* 14, 3 (Oct.), 222–222. ISSN 0896-3207.

Nelson H. F. Beebe. 1993b. Bibliography prettyprinting and syntax checking. *TUGboat* 14, 4 (Dec.), 395–419. ISSN 0896-3207.

Nelson H. F. Beebe. 1997. `bibsearch`: search BIBTEX bibliography files. (May). See `http://www.math.utah.edu/pub/bibsearch`. Unpublished, but software released.

Nelson H. F. Beebe. 2004a. A bibliographer's toolbox. *TUGboat* 25, 1, 89–104. ISSN 0896-3207.

Nelson H. F. Beebe. 2004b. `cattobib`: convert Z39.50 library catalog server data to BIBTEX markup. (15 Nov.). See `http://www.math.utah.edu/pub/cattobib`. Unpublished, but software released.

Nelson H. F. Beebe. 2007a. Extending TEX and METAFONT with floating-point arithmetic. *TUGboat* 28, 3, 319–328. ISSN 0896-3207.

Nelson H. F. Beebe. 2007b. New directions in floating-point arithmetic. In *Computation in Modern Science and Engineering: Proceedings of the [Fifth] International Conference on Computational Methods in Science and Engineering 2007 (ICCMSE 2007), Corfu, Greece, 25–30 September 2007 (AIP Conference Proceedings (#963))*. Theodore E. Simos and George Maroulis (Eds.), Vol. 2A. American Institute of Physics, Woodbury, NY, USA, xxvi + 730 + 10 (vol. 2A) book pages, 155–158. ISBN 0-7354-0476-3 (set), 0-7354-0477-1 (vol. 1), 0-7354-0478-X (vol. 2). ISBN-13 978-0-7354-0476-2 (set), 978-0-7354-0477-9 (vol. 1), 978-0-7354-0478-6 (vol. 2). ISSN 0094-243X. LCCN Q183.9 .I524 2007. See `http://www.springer.com/physics/atoms/book/978-0-7354-0478-6`.

Edwin Black. 2002. *IBM and the Holocaust: the strategic alliance between Nazi Germany and America's most powerful corporation*. Three Rivers Press, New York, NY, USA. 551 pages. ISBN 0-609-80899-0 (paperback). ISBN-13 978-0-609-80899-3 (paperback). LCCN HD9696.2.U64 I253 2002.

Judith S. Bowman. 2001. *Practical SQL: the sequel*. Addison-Wesley, Reading, MA, USA. xv + 329 pages. ISBN 0-201-61638-6. ISBN-13 978-0-201-61638-5. LCCN QA76.73.S67 B695 2001.

Judith S. Bowman, Sandra L. Emerson, and Marcy Darnovsky. 1993. *The practical SQL handbook: using Structured Query Language*, (second ed.). Addison-Wesley, Reading, MA, USA. xvii + 453 pages. ISBN 0-201-62623-3. ISBN-13 978-0-201-62623-0. LCCN QA76.73.S67 E54 1993.

Judith S. Bowman, Sandra L. Emerson, and Marcy Darnovsky. 1996. *The practical SQL handbook: using Structured Query Language*, (third ed.). Addison-Wesley Developers Press, Reading, MA, USA. xxvi + 454 pages. ISBN 0-201-44787-8. ISBN-13 978-0-201-44787-3. LCCN QA76.73.S67 B69 1996.

Judith S. Bowman, Sandra L. Emerson, and Marcy Darnovsky. 2001. *The practical SQL handbook: using SQL variants*, (fourth ed.). Addison-Wesley, Reading, MA, USA. xxxvi + 469 pages. ISBN 0-201-70309-2. ISBN-13 978-0-201-70309-2. LCCN QA76.73.S67 B688 2001.

C. Brezinski and M. Redivo-Zaglia. 2008. Rational extrapolation for the PageRank vector. *Math. Comp.* 77, 263 (July), 1585–1598. CODEN MCMPAF. ISSN 0025-5718 (paper), 1088-6842 (electronic). See http://www.ams.org/mcom/2008-77-263/S0025-5718-08-02086-3/home.html.

Robert Burgess and Emin Gün Siren. 2007. CrossTEX: A modern bibliography management tool. *TUGboat* 28, 3, 342–349. ISSN 0896-3207.

Raymond H. Chan, Chen Greif, and Dianne P. O'Leary (Eds.). 2007. *Milestones in matrix computation: the selected works of Gene H. Golub with commentaries*. Oxford University Press, Walton Street, Oxford OX2 6DP, UK. xi + 565 + 3 pages. ISBN 0-19-920681-3. ISBN-13 978-0-19-920681-0. LCCN QA188 .G67 2007. See http://www.loc.gov/catdir/enhancements/fy0737/2007276086-d.html.

E. F. Codd. 1969. *Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks*. Research Report RJ599, IBM Corporation, San Jose, CA, USA. (19 Aug.).

E. F. Codd. 1970. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* 13, 6 (June), 377–387. DOI http://doi.acm.org/10.1145/362384.362685. CODEN CACMA2. ISSN 0001-0782. http://dblp.uni-trier.de/db/journals/cacm/Codd70.html. Reprinted in (Stonebraker 1988, pp. 5–15).

Patrick W. Daly. 1994. Customised BIBTEX styles. Web-accessible software. See http://www.ctan.org/tex-archive/help/Catalogue/entries/custom-bib.html.

C. J. Date. 1987. *A guide to the SQL standard: a user's guide to the standard relational language SQL*. Addison-Wesley, Reading, MA, USA. xiv + 205 pages. ISBN 0-201-05777-8 (paperback). ISBN-13 978-0-201-05777-5 (paperback). LCCN QA76.9.D3 D36951 1987.

C. J. Date. 1989. *A guide to the SQL standard: a user's guide to the standard relational language SQL*, (second ed.). Addison-Wesley, Reading, MA, USA. xii + 228 pages. ISBN 0-201-50209-7. ISBN-13 978-0-201-50209-1. LCCN QA76.9.D3 D24 1989.

C. J. Date. 2003. Edgar F. Codd: a tribute and personal memoir. *SIGMOD Record (ACM Special Interest Group on Management of Data)* 32, 4 (Dec.), 4–13. DOI http://doi.acm.org/10.1145/959060.959061. CODEN SRECD8. ISSN 0163-5808.

C. J. Date. 2004. *An introduction to database systems*, (eighth ed.). Pearson/Addison Wesley, Boston, MA, USA. xxvii + 983 + 22 pages. ISBN 0-321-19784-4. ISBN-13 978-0-321-19784-9. LCCN QA76.9.D3 D3659 2003.

C. J. Date. 2008. Edgar F. Codd. In *Memorial Tributes*. Vol. 12. National Academy of Engineering, Washington, DC, USA, 80–87 (of xiv + 362). ISBN 0-309-12639-8. ISBN-13 978-0-309-12639-7. See http://books.nap.edu/openbook.php?record_id=12473&page=80.

C. J. Date and Hugh Darwen. 1993. *A guide to the SQL Standard: a user's guide to the standard relational language SQL*, (third ed.). Addison-Wesley, Reading, MA, USA. xvii + 414 pages. ISBN 0-201-55822-X. ISBN-13 978-0-201-55822-7. LCCN QA76.9.D3 D3695 1993.

C. J. Date and Hugh Darwen. 1997. *A guide to the SQL standard: a user's guide to the standard database language SQL*, (fourth ed.). Addison-Wesley, Reading, MA, USA. xxii + 522 pages. ISBN 0-201-96426-0. ISBN-13 978-0-201-96426-4. LCCN QA76.73.S67 D38 1997.

Lex de Haan and Toon Koppelaars. 2007. *Applied mathematics for database professionals*. Apress, Berkeley, CA, USA. xxviii + 376 pages. ISBN 1-59059-745-1. ISBN-13 9781590597453. LCCN QA76.9.D3 H239 2007. See http://www.loc.gov/catdir/toc/fy0804/2008299427.html. This book provides the mathematical background of logic and set theory for relational databases.

Sandra L. Emerson, Marcy Darnovsky, and Judith S. Bowman. 1989. *The practical SQL handbook: using Structured Query Language*. Addison-Wesley, Reading, MA, USA. xiii + 393 pages. ISBN 0-201-51738-8. ISBN-13 978-0-201-51738-5. LCCN QA76.73.S67 E54 1989.

Alan Emtage and Peter Deutsch. 1992. archie — *An Electronic Directory Service for the Internet*. Technical report, McGill University, Montréal, Québec, Canada.

G. H. Golub and W. Kahan. 1965. Calculating the Singular Values and Pseudo-Inverse of a Matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis* 2, 2, 205–224. ISSN 0887-459X. Reprinted in (Chan, Greif, and O'Leary 2007).

Judi Harris. 1993. Mining the Internet: Networked Information Location Tools: Gophers, Veronica, Archie, and Jughead. *Computing Teacher* 21, 1 (1 Aug.), 16–19. ISSN 0278-9175.

Taco Hoekwater. 2007. LuaTEX. *TUGboat* 28, 3, 312–313.

Jean-Michel Hufflen. 2003a. European bibliography styles and MlBIBTEX. *TUGboat* 24, 3, 489–498. ISSN 0896-3207.

Jean-Michel Hufflen. 2003b. MlBIBTEX's Version 1.3. *TUGboat* 24, 2, 249–262. ISSN 0896-3207.

Morris Kline. 1973. *Why Johnny can't add: the failure of the New Math*. Vintage Books, New York, NY, USA. xi + 208 pages. ISBN 0-394-71981-6. ISBN-13 978-0-394-71981-8. LCCN QA13 .K62 1974.

Leslie Lamport. 1985. *LaTeX—A Document Preparation System—User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA. xiv + 242 pages. ISBN 0-201-15790-X. ISBN-13 978-0-201-15790-1. LCCN Z253.4.L38 L35 1986.

Tom Lehrer. 1965. New Math. In *That Was the Year That Was*. Vol. R/RS 6179. Reprise Records, Burbank, CA, USA. See http://dmdb.org/discographies/lehrer.disco.html,http://www.stlyrics.com/songs/t/tomlehrer3903/newmath185502.html. Audio recording.

P. (Paul) Lindner. 1993. *Internet Gopher User's Guide*. Tech. rep., University of Minnesota, Minneapolis, MN, USA. vi + 103 pages. See also RFC 1436.

Cameron Macintosh. 1980. Tomfoolery. Web site and stage play. See http://www.tomlehrer.org/tomlehrer/enter.html.

Paul R. McJones. 2003. Collected Works of E. F. Codd. Web site. See http://www.informatik.uni-trier.de/~ley/db/about/codd.html.

Tristan Miller. 2005. Biblet: A portable BibTeX bibliography style for generating highly customizable XHTML. *TUGboat* 26, 1, 85–96. ISSN 0896-3207.

Patrick O'Neil and Elizabeth O'Neil. 2001. *Database—principles, programming, and performance*, (second ed.). Morgan Kaufmann Publishers, Los Altos, CA 94022, USA. xxiv + 870 pages. ISBN 1-55860-580-0 (paperback), 1-55860-438-3 (hardcover). ISBN-13 978-1-55860-580-0 (paperback), 978-1-55860-438-4 (hardcover). LCCN QA76.9.D3 O489 2001. See http://www.loc.gov/catdir/toc/els033/99089041.html.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. *The PageRank Citation Ranking: Bringing Order to the Web*. Tech. rep., Stanford Digital Library Technologies Project, Stanford University, Stanford, CA, USA. 17 pages. (29 Jan.). See http://dbpubs.stanford.edu/pub/1999-66;http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf.

Oren Patashnik. 2003. BibTeX yesterday, today, and tomorrow. *TUGboat* 24, 1, 25–30. ISSN 0896-3207.

Steven Rambam. 2008. The Grill: Privacy and Databases. *ComputerWorld* 42, 41 (13 Oct.), 18, 20. See http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=326821.

Richard Seltzer, Deborah S. Ray, and Eric J. Ray. 1997. *The AltaVista Search Revolution: How to find anything on the Internet*. Osborne/McGraw-Hill, Berkeley, CA, USA. xxii + 274 pages. ISBN 0-07-882235-1. ISBN-13 978-0-07-882235-3. LCCN TK5105.875 .I57 S44 1997.

Michael Stonebraker (Ed.). 1988. *Readings in Database Systems*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA. xii + 644 pages. ISBN 0-934613-65-6. ISBN-13 978-0-934613-65-1. LCCN QA76.9.D3 R4 1988.

Philip A. Viton. 2000. Getting Started with custom-bib: An introduction for SWP users. Web tutorial. (25 Aug.). See http://facweb.knowlton.ohio-state.edu/pviton/support/custom-bib.html.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. 1994. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, New York, NY, USA. xiv + 429 pages. ISBN 0-442-01863-0. ISBN-13 978-0-442-01863-4. LCCN TA1637 .W58 1994.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*, (second ed.). Morgan Kaufmann Publishers, Los Altos, CA 94022, USA. xxxi + 519 pages. ISBN 1-55860-570-3. ISBN-13 978-1-55860-570-1. LCCN TA1637 .W58 1994. See http://www.math.utah.edu/pub/mg/.

X/Open (Ed.). 1994. *X/Open CAE specification*. X/Open Co., Reading, Berkshire, UK. xvi + 156, x + 66 pages. ISBN 1-872630-58-8, 1-872630-98-7, 0-13-353558-4 (Prentice Hall). ISBN-13 978-1-872630-58-8, 978-1-872630-98-4, 978-0-13-353558-7 (Prentice Hall). LCCN QA76.73.S67 X18 1994.

Mini-colophon: The body of this article is typeset with the fourier font package and the luximono typewriter font package, augmented with the eurosym and tipa packages for additional special characters. The bibliography style is acmtrans-v2.

◇ Nelson H. F. Beebe
University of Utah
Department of Mathematics, 110 LCB
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA
WWW URL: http://www.math.utah.edu/~beebe
Telephone: +1 801 581 5254
FAX: +1 801 581 4148
Internet: beebe@math.utah.edu, beebe@acm.org, beebe@computer.org