NAME

bibtosql – convert BibTeX database files to SQL (Structured Query Language) database files

SYNOPSIS

bibtosql [**– –author**]
[**– –create**]
[**– –database** *dbname*]
[**– –help**]
[**– –server** *[MySQL | psql | PostgreSQL | SQLite]*]
[**– –version**]
< *infile* or *bibfile1 bibfile2 bibfile3 . . .*
> *outfile*

DESCRIPTION

bibtosql converts BIBTEX database files to a format suitable for populating an SQL (Structured Query Language) relational database where most of the common bibliography entry keys (`author`, `title`, `publisher`, `year`, etc.) are SQL table columns. The original BIBTEX entries are also recorded in the database tables as the `entry` column.

The acronym SQL is commonly pronounced either as it is spelled (*ess cue ell*), or like the name *sequel*.

SQL is defined in several national, government, international, and industry standards, 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). Many 2008-vintage SQL systems claim conformance to most of the 1992 ISO Standard.

The data conversion provided by **bibtosql** gives access to the fast and powerful search and display facilities of modern relational database technology without giving up the convenience of simple bibliographic text files that can be prepared manually or with the help of other bibliographic software.

In addition, since SQL databases are in wide use, and most have powerful data-conversion facilities, having bibliographic data accessible through an SQL interface may facilitate its conversion to other bibliographic formats, without the need to handle BIBTEX files directly.

The companion **bibsql**(1) utility provides a convenient interface to the SQL database, hiding the unimportant details of how the SQL client program is started, and where its data are located in the filesystem or elsewhere on the network.

OPTIONS

Command-line options may be abbreviated to a unique leading prefix, and lettercase is *not* significant.

The leading hyphen that distinguishes an option from a filename may be doubled, for compatibility with GNU and POSIX conventions. Thus, **–a**, **–author** and **– –author** are equivalent.

– –author Display an author credit on *stdout*, and then terminate with a success return code.

– –create Output leading SQL commands to delete any existing data, and create a new BIBTEX table for key/value searching. Without this option, output consists only of commands for adding new records to an existing database.

– –database *name of BIBTEX database*
Specify the name to be used for the database in the SQL system. The default name is *bibtex*.

This option exists primarily for experimentation, and should rarely be needed.

– –help Display a help message on *stdout*, giving a usage description, similar to this section of the manual pages, and then terminate with a success return code.

– –server *[MySQL | psql | PostgreSQL | SQLite]*
Select the database server type for which input is to be prepared (default: *SQLite*).

The name *psql* is an alias for *PostgreSQL*, since the former is the name of the client program on many systems.

The name may be abbreviated to any unique leading prefix, and its lettercase is *not* significant.

Although the input formats of the supported SQL systems are similar, there are important small differences that make it imperative to identify the target system.

−−version Display the program version number and release date on *stdout* and then terminate with a success return code.

−− Terminate options. All remaining arguments are treated as filenames.

DETAILED DESCRIPTION

In order to keep the data-conversion task as simple as possible, **bibtosql** assumes that the BIBTEX input files have been standardized, either manually, or more easily, with the help of **bibclean**(1). In particular, **bibtosql** assumes the following conventions:

- Comment lines begin with a percent in the first column, and can be discarded for the purposes of the database conversion.
- BIBTEX entries begin with an @ character in column 1, immediately followed by a document type, an open brace, a citation label, and a comma.
- Key/value pairs start on separate lines, and the value is a quoted string followed by a comma. The value string may span multiple lines, but it must start on the same line as the key. The quoted string may be replaced by a single name of a BIBTEX `@String{...}` definition.
- BIBTEX entries end with a single close brace at the beginning of a line.

Any BIBTEX line may optionally have trailing whitespace, which is ignored.

These conventions are most easily illustrated by two simple examples, one fictional, and one real:

```
@Book{Jones:2008:GGE,
  author =       "Terry Jones",
  title =        "Gnat and Gnu Entozoology",
  publisher =    "The National Zoo Press",
  address =      "Washington, DC, USA",
  pages =        "xx + 723",
  year =         "2008",
  ISBN =         "0-9999999-9-0",
  ISBN-13 =      "978-0-9999999-9-8",
}

@Article{Boyer:2002:MEW,
  author =       "R. S. Boyer and W. Feijen and D. Gries and
                 C. A. R. Hoare and J. Misra and J. Moore
                 and H. Richards",
  title =        "In memoriam: {Edsger W. Dijkstra}
                 1930--2002",
  journal =      j-CACM,
  volume =       "45",
  number =       "10",
  pages =        "21--22",
  month =        oct,
  year =         "2002",
  CODEN =        "CACMA2",
  ISSN =         "0001-0782",
  bibdate =      "Wed Sep 3 17:06:29 MDT 2003",
  bibsource =    "http://www.acm.org/pubs/contents/journals/cacm/",
  acknowledgement = ack-nhfb,
```

```
}
```

BIBTEX can process a compressed entry of the form

```
@Book{Jones:2008:GGE,author={Terry Jones},title=
  {Gnat and Gnu Entozoology},publisher={The Zoo Press},
  address={Washington,DC,USA},pages={xx + 723},
  year={2008},ISBN={0-9999999-9-0},ISBN-13={978-0-9999999-9-8}}
```

but **bibtosql** and most other simple BIBTEX tools cannot.

The output of **bibtosql** consists of optional SQL commands to create a database, followed by a transaction block that contains zero or more SQL commands to insert data into the database in either of two tables, one for document entries, and the other for BIBTEX `@String{key = value}` definitions. The data conversion provided by **bibtosql** is much faster than the loading of the converted data into the SQL database system, but each task needs to be done only once, as long as the database input format is not altered. Subsequent changes to the BIBTEX data can be converted by **bibtosql** and entered into an SQL database previously created with input from that tool, without having to recreate the entire database.

Once the SQL database is available, starting the SQL program and issuing queries is relatively quick, and search responses can be produced in a fraction of a second on modern personal and office computers, even for a database with a million or so entries.

The database creation is initiated by an input block that might look something like this:

```
CREATE TABLE bibtab (
        authorcount  INTEGER,
        editorcount  INTEGER,
        pagecount    INTEGER,
        bibtype      TEXT,
        filename     TEXT,
        label        TEXT,
        author       TEXT,
        editor       TEXT,
        booktitle    TEXT,
        title        TEXT,
        crossref     TEXT,
        chapter      TEXT,
        journal      TEXT,
        volume       TEXT,
        type         TEXT,
        number       TEXT,
        institution  TEXT,
        organization TEXT,
        publisher    TEXT,
        school       TEXT,
        address      TEXT,
        edition      TEXT,
        pages        TEXT,
        day          TEXT,
        month        TEXT,
        monthnumber  TEXT,
        year         TEXT,
        CODEN        TEXT,
        DOI          TEXT,
        ISBN         TEXT,
        ISBN13       TEXT,
        ISSN         TEXT,
        LCCN         TEXT,
```

```
        MRclass      TEXT,
        MRnumber     TEXT,
        MRreviewer   TEXT,
        bibdate      TEXT,
        bibsource    TEXT,
        bibtimestamp TEXT,
        note         TEXT,
        series       TEXT,
        URL          TEXT,
        abstract     TEXT,
        keywords     TEXT,
        remark       TEXT,
        subject      TEXT,
        TOC          TEXT,
        ZMnumber     TEXT,
        entry        TEXT NOT NULL UNIQUE
);

CREATE TABLE namtab (
        name         TEXT NOT NULL UNIQUE,
        count        INTEGER
);

CREATE TABLE strtab (
        key          TEXT,
        value        TEXT,
        entry        TEXT NOT NULL UNIQUE
);
```

The key values are mostly character strings, and the key names are available for use in subsequent searches in the database. The lettercase of key names is not significant. The values are of type `TEXT`, a character string type that holds at least 65,535 characters on all supported SQL databases. As described later, MySQL requires a following parenthesized length field.

The key names `author` through `ZMnumber` are given in the `CREATE` command in the order chosen by **biborder**(1), which is similar to that of the field order in conventional publication and reference lists. That order has no significance for either searching or database efficiency.

While most of the key names in the `bibtab` table are standard ones in BibTeX, a few are not. They include

- `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).
- `bibdate` (Unix **date**(1) 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 fuer Mathematik* database number).

In **bibtosql**, we follow the common SQL practice that indeterminate, unknown, or unset values are output as NULL, so as to facilitate their exclusion in later searches.

The `NOT NULL` attribute on the `entry` key requires that column to be supplied in each row: it is the only mandatory row entry.

The `UNIQUE` attribute on the `entry` key ensures that only one distinct copy of each BIBTEX entry is stored in the database, even when identical copies are present in the input stream. That situation is common, since BIBTEX entries are often copied between files and shared between users.

For PostgreSQL and SQLite, the `TEXT` type can represent a string of any size. For MySQL, that type must be followed by a parenthesized typical length: we use (32767). If that length is too small, long strings are arbitrarily, and silently, truncated.

The data-insertion block looks like this:

```
BEGIN TRANSACTION;
... INSERT commands go here ...
COMMIT;
```

This allows the database system to delay committing the input to permanent database storage until all of the input has been processed successfully. Any error since the start of the transaction causes the entire batch of updates to be discarded.

A single insertion command might look like this:

```
INSERT INTO bibtab
        (authorcount, editorcount, pagecount, bibtype, filename, label,
        author, editor, booktitle, title, crossref, chapter, 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)
        VALUES (
        NULL,
        5,
        549,
```

```
        'book',
        'master.bib',
        'Goossens:2008:LGC',
        NULL,
        'Michel Goossens and Frank Mittelbach and Sebastian Rahtz and \
Denis Roegel and Herbert Voss',
        NULL,
        'The LaTeX Graphics Companion',
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        'pub-AW',
        NULL,
        'pub-AW:adr',
        'Second',
        'xxi + 528',
        NULL,
        NULL,
        NULL,
        '2008',
        NULL,
        NULL,
        '0-321-50892-0 OR 0321508920',
        '978-0-321-50892-8 OR 9780321508928',
        NULL,
        'Z253.4.L38G663 2008',
        NULL,
        NULL,
        NULL,
        'Thu Sep 20 12:02:17 MDT 2007',
        'z3950.bibsys.no:2100/BIBSYS',
        '2007.09.20 12:02:17 MDT',
        NULL,
        'Addison-Wesley series on tools and techniques for computer \
typesetting',
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        '
@Book{Goossens:2008:LGC,
  editor =       "Michel Goossens and Frank Mittelbach and Sebastian
                 Rahtz and Denis Roegel and Herbert Vo{\ss}",
  title =        "The {\LaTeX} Graphics Companion",
  publisher =    pub-AW,
```

```
  address =      pub-AW:adr,
  edition =      "Second",
  year =         "2008",
  ISBN =         "0-321-50892-0",
  ISBN-13 =      "978-0-321-50892-8",
  LCCN =         "Z253.4.L38G663 2008",
  bibdate =      "Thu Sep 20 12:02:17 MDT 2007",
  bibsource =    "z3950.bibsys.no:2100/BIBSYS",
  series =       "Addison-Wesley series on tools and techniques for
                 computer typesetting",
  acknowledgement = ack-nhfb,
}'
);
```

Long lines have been wrapped with a backslash-newline for display purposes here; such wrapping is absent from the program output.

The `ISBN` and `ISBN13` values are augmented with hyphen-free book numbers. Although the hyphen positions are well defined, they are usually unpredictable to humans, and may be incorrect in hand-produced bibliographic data. The hyphen-free form provides another way to find them in searches.

The `INSERT` statement is clearly verbose, and somewhat awkward and error-prone because of the separation of keys from their values, but that matters little for a computer program. When all fields are provided in the expected order, the parenthesized list of key names can be omitted. However, **bibtosql** intentionally includes the key names to ensure compatibility of its output file with databases created by future versions of the program that might define additional columns.

Manual insertions later can provide data for subsets of keys; any that are omitted simply acquire NULL values.

Except in the value of the `entry` key, consecutive whitespace is reduced to a single space, TEX macros are simplified or removed, and TEX accents and braces are stripped from key values. That convention makes specification of search strings for key values easier. Thus, to search for `Paul Erd{\H{o}}s`, use the pattern `%P%Erdos%` in the `author` or `editor` columns. To find documents with `{\TeX}book` or `\TeX{}book` or `\TeX book` in their titles, use the pattern `%texbook%`. Search patterns are described in the **SQL SEARCH TUTORIAL** section of the manual pages for **bibsql**(1).

SQLite SERVER SETUP

SQLite is the simplest of the databases supported by **bibtosql** and **bibsql**(1). The same program, **sqlite3**(1), acts as both a client and a server, and thus, the database must reside on the same system as the client. There is no data protection other than that provided by the host filesystem. In contrast to the complex access controls of client/server database systems, with SQLite, no special user privileges are needed to create or use an SQLite database.

SQLite is an excellent system for users new to databases to learn about, and experiment with, SQL.

The SQLite software is *public domain* and may be used for any purpose, and redistributed, without restriction. The software is fast, portable, and reliable. It can be built easily on almost any modern desktop operating system, including all current Unix, Mac OS X, and Windows systems. Package downloads and documentation are available at the developer site, `http://www.sqlite.org/`.

An interesting feature of SQLite is that there is an amalgamated distribution where the many ordinarily-separate source code files are collected into just two large files, enabling advanced compilers to do interprocedural optimizations and procedure inlining, further enhancing database performance.

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.

To create an SQLite database, name it on the command line, and then input a file of SQL commands to populate the database:

% bibtosql ∗.bib | sqlite3 bibtex.db

To search the database, just run the program again with the same filename argument:

% sqlite3 bibtex.db
`sqlite>` **. . . your input here . . .**

If a database filename is not provided, then SQLite creates an in-memory database that is lost when the program terminates. This is useful for testing new data and new table formats, as well as for fast searching of small private collections of BIBTEX data. Here is a sample in-memory session:

% sqlite3
`sqlite>` **.read testdata.sql**
`sqlite>` **. . . your input here . . .**

For use with **bibsql**(1), the SQLite database file should be created and copied to the installation directory, and then set read-only to prevent unintentional or malicious modification. Assuming the default prefix of `/usr/local`, the job can be done like this:

% cp bibtex.db /usr/local/share/lib/bibsql/bibsql-x.yz/
% chmod a-wx,a+r /usr/local/share/lib/bibsql/bibsql-x.yz/bibtex.db

If the database file is large, searches may be made faster by creating indexes of important parts of the data:

`sqlite>` **create index bibidx on bibtab (author, title, label);**
`sqlite>` **create index namidx on namtab (name);**
`sqlite>` **create index stridx on strtab (key, value);**

MySQL SERVER SETUP

Setting up the server side of a MySQL connection is complex, and these step-by-step notes may be a helpful guide. It is a good thing to log what you are doing, either with the log facility provided in an **xterm**(1) window (usually on a menu bound to Ctl-Button-1), or else with the **script**(1) command.

We assume that the MySQL software has been installed in the directory `/usr/local/bin`, and its databases are to be created in subdirectories under the directory `/var/lib/mysql`, named by the database. For clarity, we use three different prompt strings: `#` for the `root` user, `$` for the MySQL superuser, and `%` for an unprivileged ordinary user.

- If you have not yet done so, create a Unix account for the MySQL superuser:
  # adduser mysql

  This account has no special Unix privileges, and exists only to restrict access to database files, and to provide an account under which the MySQL server daemon runs.

  In addition, it has no password set, so direct login with this account is impossible. Instead, you must first become the `root` user, and then use the **su**(1) or **sudo**(1) commands to change to the `mysql` account.

- Create a working directory and a log file owned by the `mysql` account:
  # mkdir /var/run/mysqld/
  # chown mysql.mysql /var/run/mysqld
  # touch /var/log/mysqld.log
  # chown mysql.mysql /var/log/mysqld.log

  Consult the log file to help diagnose database errors and connection problems.

- Become the MySQL superuser and start the MySQL server:
  # su – mysql
  $ cd /usr/local/mysql
  $ mysqld_safe &

```
nohup: ignoring input and redirecting stderr to stdout
Starting mysqld daemon with databases from /var/lib/mysql
```

  This program is a wrapper around the actual daemon, **mysql**(1), that handles incoming SQL

connections. The wrapper monitors the daemon process, and automatically restarts it, should it fail. We show later how to get it started each time the server system reboots.

As long as there were no errors, the MySQL daemon is almost ready to handle both local and network connections from MySQL clients, as soon as we create a database and MySQL user names.

- Check the running processes to find where various things are located:
  $ ps auxww | grep –i sql

```
. . .
mysql 11795 0.0 0.0  52776 1132  pts/6 S  21:58 0:00 \
    /bin/sh /usr/local/bin/mysqld_safe
mysql 11821 0.2 0.5 131092 17032 pts/6 Sl 21:58 0:00 \
    /usr/local/libexec/mysqld \
    --basedir=/usr/local \
    --datadir=/var/lib/mysql \
    --pid-file=/var/run/mysqld/mysqld.pid \
    --skip-external-locking \
    --socket=/var/lib/mysql/mysql.sock
. . .
```

  We wrapped the long process reports with backslash-newline to make them fit here, and also to highlight the options passed by the wrapper to the daemon. Notice, in particular, the socket setting, because we need it later. A Unix *socket* is the local equivalent of an external network connection: the socket allows interprocess communication on the same machine. When we start a MySQL client later, we have to tell it what socket to use to reach the MySQL server.

  If you record the socket pathname in an environment variable, like this,
  $ MYSQL_UNIX_PORT=/var/lib/mysql/mysql.sock
  $ export MYSQL_UNIX_PORT

  then it need not be supplied in later commands.

- Check that the server is listening for client connections:
  % netstat –an | egrep –i '1186|2273|3306|mysql'

```
tcp  0 0 0.0.0.0:3306        0.0.0.0:*             LISTEN
tcp  0 0 155.101.96.167:3306 155.101.96.19:47297  ESTABLISHED
tcp  0 0 155.101.96.167:3306 155.101.96.132:27429 ESTABLISHED
unix 2 [ ACC ] STREAM LISTENING 3885498 /var/lib/mysql/mysql.sock
```

  That command is another way to find out the socket path. The numbers in the search pattern are standard network ports reserved for MySQL and recorded in the file `/etc/services`.

- Initialize the database directory tree with template files:
  $ mysql_install_db – –user=mysql

  That command produces a lengthy report that you should record. It gives instructions for setting up MySQL user accounts and creating databases, and points you to free online documentation and commercial support.

- Create three MySQL user names (unrelated to Unix accounts) to be assigned access privileges shortly:
  $ mysqladmin – –socket=/var/lib/mysql/mysql.sock –u root password '/s/e/c/r/e/t/'
  $ mysqladmin – –socket=/var/lib/mysql/mysql.sock –u bibtex password '/t/e/r/c/e/s/'
  $ mysqladmin – –socket=/var/lib/mysql/mysql.sock –u reader

  You should, of course, use different, and harder-to-guess, passwords than shown here.

  If `MYSQL_UNIX_PORT` is not set, then we have to supply the socket location. Without it, MySQL programs assume that the socket resides in the `/tmp` directory, and quit when they fail to find it.

  WARNING WARNING WARNING: There is a clear security hole here: anyone else on the system

who is running a **ps**(1) command with options for verbose output can see the entire command lines, including the passwords. For that, and other security reasons, database server machines should have extremely restricted access, with only system-manager login accounts.

When you run a MySQL client with the password-protected user names, the client demands `Enter password:` before proceeding. The password that you supply at that point is program input, and is neither echoed as you type, nor visible in **ps**(1) command output.

The chapter *MySQL Server Administration* in the *MySQL Reference Manual* contains sections entitled *Assigning Account Passwords* and *Keeping Your Password Secure* that describe other, possibly more secure, ways to supply passwords.

- Create an empty database for the bibliographic data:
  $ mysqladmin –u root –p create bibtex – –socket=/var/lib/mysql/mysql.sock
  `Enter password:` **/s/e/c/r/e/t/**

  The **–p** option requests a password prompt, as shown. The alternative is to supply it on the command line as the option **– –password='/s/e/c/r/e/t/'**, but that is insecure and inadvisable. It may, however, be necessary if the command is run in batch mode where a prompt is not possible.

  There is no screen output from this command, but it creates a directory `/var/lib/mysql/bibtex` containing a single file, `db.opt`, that specifies the default character set and sorting order.

- If the database directory is on a filesystem that is accessible via network mounts to clients, for security and performance reasons, consider moving it now to another filesystem that is strictly local. If you make such a move, you must then leave behind a symbolic link:
  $ mv /var/lib/mysql/bibtex /local/mysql/
  $ ln –s /local/mysql/bibtex /var/lib/mysql/

- Assign database-access privileges to the newly-created accounts:
  $ mysql –p – –socket=/var/lib/mysql/mysql.sock –u root
  `Enter password:` **/s/e/c/r/e/t/**
  ...
  `mysql>` **GRANT ALL PRIVILEGES ON bibtex.* TO 'bibtex'@'localhost';**
  `Query OK, 0 rows affected (0.00 sec)`
  `mysql>` **GRANT SELECT ON bibtex.* TO 'reader'@'%';**
  `Query OK, 0 rows affected (0.00 sec)`

  Now the user `bibtex` can create and populate tables in the `bibtex` database, but only from the local host where the server is running. For security reasons, it is not advisable to allow changes to the database from hosts other than the MySQL server system.

  The user `reader` can read tables in the `bibtex` database from any host on the Internet. Further restrictions may be advisable. If so, instead of a percent meaning all host names, remove all access from that user, then limit access to hosts in your domain:
  `mysql>` **REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'reader'@'%';**
  `Query OK, 0 rows affected (0.00 sec)`
  `mysql>` **GRANT SELECT ON bibtex.* TO 'reader'@'%.example.com';**
  `Query OK, 0 rows affected (0.00 sec)`

  You can also use a numeric IP address to permit a connection from a specific host:
  `mysql>` **grant select on bibtex.* to 'reader'@'192.168.1.103';**

- Find the accounts in database tables and check their access privileges:
  `mysql>` **show databases;**

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bibtex             |
```

```
| mysql              |
| test               |
+--------------------+
mysql> use information_schema;
mysql> show tables;
+---------------------------------------+
| Tables_in_information_schema          |
+---------------------------------------+
| CHARACTER_SETS                        |
| COLLATIONS                            |
...
| USER_PRIVILEGES                       |
| VIEWS                                 |
+---------------------------------------+
mysql> select * from USER_PRIVILEGES;
+--------------------------+---------------+----------------+--------------+
| GRANTEE                  | TABLE_CATALOG | PRIVILEGE_TYPE | IS_GRANTABLE |
+--------------------------+---------------+----------------+--------------+
| 'root'@'localhost'       | NULL          | SELECT         | YES          |
...
| 'root'@'127.0.0.1'       | NULL          | CREATE USER    | YES          |
| 'bibtex'@'localhost'     | NULL          | USAGE          | NO           |
| 'reader'@'%.example.com' | NULL          | USAGE          | NO           |
+--------------------------+---------------+----------------+--------------+
```

- Now populate the database with suitable data produced by **bibtosql**:
  **$ mysql –p –u bibtex –D bibtex – –socket=/var/lib/mysql/mysql.sock **
  **< */path/to/data/*test-mysql.sql **
  > logs/bibtex-‘hostname‘.log 2>&1
  `Enter password:` **/s/e/c/r/e/t/**

  This step can take a long time if the input file is large.

  After a test with a file of about 11 400 BibTeX entries, the database directory looks something like this:
  $ ls –lFd /var/lib/mysql/bibtex
  `drwx------  2 mysql mysql 4096 2008-11-01 17:31 /var/lib/mysql/bibtex/`
  $ ls –l /var/lib/mysql/bibtex
  ```
  total 16104
  -rw-rw---- 1 mysql mysql    10218 2008-11-01 23:49 bibtab.frm
  -rw-rw---- 1 mysql mysql 15976280 2008-11-01 23:49 bibtab.MYD
  -rw-rw---- 1 mysql mysql     1024 2008-11-01 23:49 bibtab.MYI
  -rw-rw---- 1 mysql mysql       65 2008-11-01 23:49 db.opt
  -rw-rw---- 1 mysql mysql     8592 2008-11-01 23:49 namtab.frm
  -rw-rw---- 1 mysql mysql   299268 2008-11-01 23:49 namtab.MYD
  -rw-rw---- 1 mysql mysql     1024 2008-11-01 23:49 namtab.MYI
  -rw-rw---- 1 mysql mysql     8622 2008-11-01 23:49 strtab.frm
  -rw-rw---- 1 mysql mysql   124408 2008-11-01 23:49 strtab.MYD
  -rw-rw---- 1 mysql mysql     1024 2008-11-01 23:49 strtab.MYI
  ```
  $ du –h /var/lib/mysql/bibtex
  `16M     /var/lib/mysql/bibtex`

  Notice that the directory and file permissions allow access only by the `mysql` user.

  A simple calculation shows that the database consumes about 1400 bytes of filesystem space for an average BibTeX entry. We can compare this with a report from MySQL itself:
  `mysql>` **select avg(length(entry)) from bibtab;**

```
+--------------------+
| avg(length(entry)) |
+--------------------+
|           860.1307 |
+--------------------+
1 row in set (0.32 sec)
```

The count of 860 bytes represents just the `entry` string, and does not include the space needed for the other four dozen or so columns in each row of the `bibtex` database.

- You can get lengthy reports of MySQL default variable settings, and a list of all variables, like this:
  $ mysqladmin −−socket=/var/lib/mysql/mysql.sock −−print-defaults
  $ mysqladmin −−socket=/var/lib/mysql/mysql.sock variables
- The BIBTEX database is now ready for use. Test it like this from another user account:
  % mysql −−socket=/var/lib/mysql/mysql.sock –u reader –D bibtex
  `mysql>` **select count(*) from bibtab;**

```
+----------+
| count(*) |
+----------+
|    11431 |
+----------+
1 row in set (0.00 sec)
```

- The last step is to ensure that the MySQL server terminates safely when the system shuts down, and starts when the system reboots.

  The details depend on the operating system, but the MySQL software installation may contain a sample startup script in the file `/usr/local/share/mysql/mysql.server`. Your own system may already have some scripts that can be used after minor edits: look for files `/etc/init.d/*sql*`, `/etc/rc*.d/*sql*`, and `/etc/mysql/*`.

  The subdirectory `scripts` in the **bibsql**(1) software distribution contains copies of such scripts used at the author's site.

  On a Red Hat GNU/Linux system, only two simple changes to set location variables are needed:
  % diff /usr/local/share/mysql/mysql.server /etc/init.d/mysql

```
46,47c46,47
< basedir=
< datadir=
---
> basedir=/usr/local
> datadir=/var/lib/mysql
```

  Here is how to start, query, and stop the server:
  # /etc/init.d/mysql start

```
Starting MySQL.                           [  OK  ]
```

  # /etc/init.d/mysql status

```
MySQL running (8805)                      [  OK  ]
```

  # /etc/init.d/mysql stop

```
Shutting down MySQL..                     [  OK  ]
```

  Symbolic links from the controlling script `/etc/init.d/mysql` to two others make it accessible to the Unix startup and shutdown procedures:
  # cd /etc/init.d
  # ln –s ../init.d/mysql ../rc3.d/S99mysql
  # ln –s ../init.d/mysql ../rc0.d/K01mysql

  Should the BIBTEX database and the MySQL server no longer be needed, then those two links must be

removed. The database itself, and its filesystem directory, can be removed with a single command in a privileged client session:
$ mysql –p – –socket=/var/lib/mysql/mysql.sock –u root
`Enter password:` **/s/e/c/r/e/t/**
`...`
`mysql>` **DROP DATABASE bibtex;**
`Query OK, 3 rows affected (0.04 sec)`

There is no prompt to ask whether you really want to remove the database. To get it back, you either need to recover it from filesystem backups, or recreate it with the help of data from **bibtosql** and the two **mysqladmin**(1) and **mysql**(1) steps shown earlier in this section.

If the database file is large, searches may be made faster by creating indexes of important parts of the data:

`mysql>` **alter table bibtab add index (author(50),title(100),label(50));**
`mysql>` **alter table strtab add index (‘key‘(50), value(50));**
`mysql>` **alter table namtab add index (name(50));**

At the end of the next session, we discuss how to automatically update the database by regularly-scheduled jobs that find and insert new BIBTEX data.

PostgreSQL SERVER SETUP

Setting up the server side of a PostgreSQL connection is a daunting process if you have never done it. These step-by-step notes may therefore be helpful. It is a good thing to log what you are doing, either with the log facility provided in an **xterm**(1) window (usually on a menu bound to Ctl-Button-1), or else with the **script**(1) command.

We assume that the PostgreSQL software has been installed in the tree `/usr/local/pgsql`, and its databases are to be created under the subdirectory `data`. For clarity, we use three different prompt strings: `#` for the `root` user, `$` for the PostgreSQL superuser, and `%` for an unprivileged ordinary user.

- If you have not yet done so, create a Unix account for the PostgreSQL superuser:
  # adduser postgres

  This account has no special Unix privileges, and exists only to restrict access to database files, and to provide an account under which the PostgreSQL server daemon runs.

  In addition, it has no password set, so direct login with this account is impossible. Instead, you must first become the `root` user, and then use the **su**(1) or **sudo**(1) commands to change to the `postgres` account.

- As the `root` user, reset the ownership of two subdirectories in the installation directory:
  # cd /usr/local/pgsql
  # mkdir data
  # mkdir logs
  # chown –R postgres.postgres data logs

  For security and performance reasons, consider making those two directories symbolic links to a separate file tree, preferably on a local, rather than network-mounted, disk.

  All other files in the directory tree should be owned by some other user, preferably an account without login privileges, such as `bin`, so that a compromised `postgres` account cannot be used to change those files.

- Become the `postgres` user, set the search path to include PostgreSQL utilities, and initialize the database directory tree with template files:
  # su – postgres
  $ PATH=$PATH:/usr/local/pgsql/bin
  $ export PATH
  $ cd /usr/local/pgsql
  $ initdb –D /usr/local/pgsql/data/bibtex

- Create two PostgreSQL user names (unrelated to Unix accounts, and often called *roles* instead), one with permissions to create databases, and another with permissions to just read the bibliographic database:
  $ createuser bibtex
  ```
  Shall the new role be a superuser? (y/n) n
  Shall the new role be allowed to create databases? (y/n) y
  Shall the new role be allowed to create more new roles? (y/n) n
  ```
  $ createuser reader
  ```
  Shall the new role be a superuser? (y/n) n
  Shall the new role be allowed to create databases? (y/n) n
  Shall the new role be allowed to create more new roles? (y/n) n
  ```
- Start the PostgreSQL server in its default mode where connections are only accepted from clients on the same host:
  $ pg_ctl –D /usr/local/pgsql/data/bibtex –l logs/server-‘hostname‘.log start
- Create the database `bibtex`:
  $ createdb /usr/local/pgsql/data/bibtex
- Populate the newly-created database with suitable data produced by **bibtosql**, using the newly-created PostgreSQL `bibtex` role:
  **$ psql –d /usr/local/pgsql/data/bibtex –U bibtex **
  **< *</path/to/data/*test-psql.sql **
  > logs/bibtex-‘hostname‘.log 2>&1

  In the input redirection, supply the correct directory path to the SQL input file.
- Start the client program, check the status of the two roles, assign access permissions to them, and then check their permissions:
  $ psql –d /usr/local/pgsql/data/bibtex –U postgres
  ```
  postgres=# \du
                                 List of roles
   Role name | Superuser | Create role | Create DB | Connections | Member of
  -----------+-----------+-------------+-----------+-------------+-----------
   reader    | no        | no          | no        | no limit    | {}
   bibtex    | no        | no          | yes       | no limit    | {}
   postgres  | yes       | yes         | yes       | no limit    | {}
  postgres=# grant all on bibtab, namtab, strtab to bibtex;
  postgres=# grant select on bibtab, namtab, strtab to reader;
  postgres=# \z
    Access privileges for database "/usr/local/pgsql/data/bibtex"
   Schema | Name   | Type  |           Access privileges
  --------+-------+-------+---------------------------------------
   public | bibtab | table | {bibtex=arwdxt/bibtex,reader=r/bibtex}
   public | namtab | table | {bibtex=arwdxt/bibtex,reader=r/bibtex}
   public | strtab | table | {bibtex=arwdxt/bibtex,reader=r/bibtex}
  ```
- As an ordinary user, start the SQL client program on the current host with the `reader` role, verify that it can see the newly-created database and its newly-added contents, and try to alter the database, an action that should be denied:
  % psql –d /usr/local/pgsql/data/bibtex –U reader
  ```
  /usr/local/pgsql/data/bibtex=> \l
                   List of databases
              Name               |  Owner   | Encoding
  -------------------------------+----------+----------
   /usr/local/pgsql/data/bibtex | postgres | UTF8
   postgres                      | postgres | UTF8
  ```

```
 template0                    | postgres | UTF8
 template1                    | postgres | UTF8
/usr/local/pgsql/data/bibtex=> select count(*) from strtab;
 count
-------
   920
/usr/local/pgsql/data/bibtex=> insert into bibtab (author, title) values('cat', 'dog');
ERROR:  permission denied for relation bibtab
/usr/local/pgsql/data/bibtex=> insert into strtab values('larry', 'curly', 'moe');
ERROR:  permission denied for relation strtab
/usr/local/pgsql/data/bibtex=> insert into namtab values ('Joe Blow', 0);
ERROR:  permission denied for relation namtab
```

At this point, users on the server host can run the client program and successfully access the database, and the `reader` role cannot modify the database.

- Enable network access to the server by editing the PostgreSQL installation file `data/bibtex/pg_hba.conf` to specify the Internet numeric addresses *from which* client connections are to be accepted. For example, we could add these lines at the end of the file:

```
host /usr/local/pgsql/data/bibtex bibtex 192.168.32.0/24 password
host /usr/local/pgsql/data/bibtex bibtex 192.168.96.0/26 password
host /usr/local/pgsql/data/bibtex reader 192.168.32.0/24 trust
host /usr/local/pgsql/data/bibtex reader 192.168.96.0/26 trust
```

  On the first two lines, the `password` keyword says that user `bibtex` must supply a password (set later) to access the database. We allow connections to the `bibtex` database for two PostgreSQL roles from 256 addresses beginning `192.168.32`, and 64 addresses beginning `192.168.96`. The trailing `/24` means that the leading 24 bits of the incoming address must match those of the pattern, so the last 8 bits can represent any of 256 hosts. A trailing `/32` restricts connections to a single host.

- Edit the file `data/postgresql.conf` to have a line like this:
  listen_addresses = '*'

  This tells the PostgreSQL server to listen on all local network connections for incoming client requests.

  If you have multiple networks, one of which reaches the outside world, and the others are "inside the fence", then you can replace the asterisk by a comma-separated list of inside addresses of the server, like this:
  listen_addresses = '192.168.96.10, 192.168.96.11, 192.168.96.12'

  The server can then respond to clients inside your organization, but cannot offer database access to clients elsewhere on the Internet.

- As the `postgres` user, restart the SQL server so that it rereads the changed configuration file that enables listening for a network connection from a client. Then check that the PostgreSQL port 5432 is being listened to:
  $ pg_ctl –D /usr/local/pgsql/data/bibtex –o '–d 5' –l logs/server-'hostname'.log restart
  $ netstat –an | grep 5432

```
tcp  0 0 0.0.0.0:5432        0.0.0.0:*           LISTEN
tcp  0 0 192.168.96.157:5432 192.168.96.19:36214 ESTABLISHED
tcp  0 0 :::5432             :::*                LISTEN
unix 2 [ ACC ]     STREAM   LISTENING     1119331 /tmp/.s.PGSQL.5432
unix 3 [ ]         STREAM   CONNECTED     1124578 /tmp/.s.PGSQL.5432
```

  The `–d 5` option requests maximal debug information in the log file, which is a good thing to do until you are sure the system is operating properly. The **pg_ctl**(1) program passes that option to **postmaster**(1), the PostgreSQL program that handles the startup of an SQL server process to respond to the client requests. That way, a single server system can provide simultaneous SQL access for multiple databases.

The `tcp` lines verify that the network port is being monitored, and the `unix` lines show that the socket port for local connections is also active.

- On another host, as an ordinary user, check whether the server is reachable from the client (change the sample host name):
  % psql –d /usr/local/pgsql/data/bibtex –U reader –h server.example.com
  `/usr/local/pgsql/data/bibtex=>` **select count(*) from strtab;**
  ```
   count
  -------
     920
  ```
  You can make the same check of the network connection from the server host by changing the hostname to `localhost`, and in another shell session on that host, requesting a network status report:
  % psql –d /usr/local/pgsql/data/bibtex –U reader –h localhost
  % netstat –an | grep 5432
  ```
  tcp  0  0 127.0.0.1:5432  0.0.0.0:*         LISTEN
  tcp  0  0 127.0.0.1:5432  127.0.0.1:33126  ESTABLISHED
  tcp  0  0 127.0.0.1:33126 127.0.0.1:5432   ESTABLISHED
  tcp  0  0 ::1:5432  :::*                    LISTEN
  unix 2  [ ACC ]  STREAM  LISTENING  15814  /tmp/.s.PGSQL.5432
  ```
  The address `127.0.0.1` is that of the *loopback* interface, which provides a way for client and server to be on the same system, while still using normal TCP/IP network system calls, instead of sockets, for communication.
- Our PostgreSQL client/server setup is now operational, but we need to tighten security further. We should restrict the `postgres` and `bibtex` roles so that they can be used only on the server system, and so that at least the first of them requires a database password.
  $ psql –d /usr/local/pgsql/data/bibtex –U postgres
  `/usr/local/pgsql/data/bibtex=#` **alter user postgres password 's/e/c/r/e/t'**
  `/usr/local/pgsql/data/bibtex=#` **alter user bibtex password 't/e/r/c/e/s'**
- In the `/etc` directory tree, install a script that is to be executed when the server machine starts up and shuts down. The script should start and stop the database. The details vary substantially across different flavors of Unix, so we leave that job to you, the local software installer.

  PostgreSQL software distributions contain a few sample scripts in the directory tree `postgresql-x.y.z/contrib/start-scripts`. Your own system may already have some that can be used after minor edits: look for files `/etc/init.d/*sql*`, `/etc/rc.d/*sql*`, `/etc/pgsql/*`, and `/etc/postgresql/*`.

  The subdirectory `scripts` in the **bibsql**(1) software distribution contains copies of such scripts used at the author's site.

  The `postgres` login directory needs a shell startup file that sets the `PATH` variable to include the directory where PostgreSQL programs are found:
  # ypmatch postgres passwd
  `postgres:*:14255:14255:Database PostgreSQL:/var/pgsql:/bin/sh`
  # cat >> /var/pgsql/.profile
  PATH=$PATH:/usr/local/pgsql/bin
  export PATH
  ^d
- If your BibTeX files change, the database should be updated. It is then useful to set up a **cron**(8) job to be run at suitable intervals on the database server under the `postgres` user account. The job reads a list of directories containing BibTeX files, uses **find**(1) to identify the files that have changed since the last update, and then runs **bibtosql** to convert them into an input stream for **psql**(1) under the role

`bibtex`. Entries that are unchanged from the last update are ignored because of the `UNIQUE` attribute on the `entry` column, but any that have been changed or added are inserted into the database. This is easier than it sounds:

```
$ find ‘cat BIBTEXFILEPATHS‘ –name ’∗.bib’ \
    –a –type f \
    –a –perm –444 \
    –a –newer LAST_TIMESTAMP | \
  xargs bibtosql –s psql | \
  psql –d /usr/local/pgsql/data/bibtex –U bibtex
$ touch LAST_TIMESTAMP
```

To avoid entering erroneously-formatted entries into the database, use **bibparse**(1) to validate each BIBTEX file before permitting it to be seen by **bibtosql**.

If the database file is large, searches may be made faster by creating indexes of important parts of the data:

```
psql> create index bibidx on bibtab (author,title,label);
psql> create index namidx on namtab (name);
psql> create index stridx on strtab (key, value);
```

AUTHOR

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
Tel: +1 801 581 5254
FAX: +1 801 581 4148
Email: `beebe@math.utah.edu`, `beebe@acm.org`, `beebe@computer.org` (Internet)
URL: `http://www.math.utah.edu/~beebe`

The master archive for the **bibsql**(1) and **bibtosql** software is at these equivalent locations:

```
ftp://ftp.math.utah.edu/pub/tex/bibsql
http://www.math.utah.edu/pub/tex/bibsql
```

SEE ALSO

bib2xml(1), **bib2xtx**(1), **bibcheck**(1), **bibclean**(1), **bibdestringify**(1), **bibdup**(1), **bibextract**(1), **bibindex**(1), **bibjoin**(1), **biblabel**(1), **biblex**(1), **biblook**(1), **biborder**(1), **bibparse**(1), **bibsearch**(1), **bibsort**(1), **bibsplit**(1), **bibsql**(1), **bibtex**(1), **bibunlex**(1), **cattobib**(1), **citefind**(1), **citesub**(1), **citetags**(1), **cron**(8), **date**(1), **find**(1) **latex**(1), **mysql**(1), **pg_ctl**(1), **postmaster**(1), **psql**(1), **ref2bib**(1), **scribe**(1), **script**(1), **sqlite3**(1), **tex**(1), **xml2bib**(1), **xml2wordbib**(1), **xterm**(1), **xtx2bib**(1).