PostgreSQL Administrator’s Guide

The PostgreSQL Development Team

Edited by

Thomas Lockhart

PostgreSQL Administrator’s Guide

by The PostgreSQL Development Team

Edited by Thomas Lockhart

Table of Contents

List of Tables

List of Figures

List of Examples

Summary

Postgres, developed originally in the UC Berkeley Computer Science Department, pioneered many of the object-relational concepts now becoming available in some commercial databases. It provides SQL92/SQL3 language support, transaction integrity, and type extensibility. PostgreSQL is an open-source descendant of this original Berkeley code.

Chapter 1. Introduction

This document is the Administrator's Manual for the PostgreSQL (http://postgresql.org/) database management system, originally developed at the University of California at Berkeley. PostgreSQL is based on Postgres release 4.2 (http://s2k-ftp.CS.Berkeley.EDU:8000/postgres/postgres.html). The Postgres project, led by Professor Michael Stonebraker, was sponsored by the Defense Advanced Research Projects Agency (DARPA), the Army Research Office (ARO), the National Science Foundation (NSF), and ESL, Inc.

Resources

This manual set is organized into several parts:

Tutorial

An introduction for new users. Does not cover advanced features.

User's Guide

General information for users, including available commands and data types.

Programmer's Guide

Advanced information for application programmers. Topics include type and function extensibility, library interfaces, and application design issues.

Administrator's Guide

Installation and management information. List of supported machines.

Developer's Guide

Information for Postgres developers. This is intended for those who are contributing to the Postgres project; application development information should appear in the *Programmer's Guide*. Currently included in the *Programmer's Guide*.

Reference Manual

Detailed reference information on command syntax. Currently included in the *User's Guide*.

In addition to this manual set, there are other resources to help you with Postgres installation and use:

man pages

The man pages have general information on command syntax.

FAQs

The Frequently Asked Questions (FAQ) documents address both general issues and some platform-specific issues.

READMEs

README files are available for some contributed packages.

Web Site

The Postgres (postgresql.org) web site might have some information not appearing in the distribution. There is a mhonarc catalog of mailing list traffic which is a rich resource for many topics.

Mailing Lists

The pgsql-general (mailto:pgsql-general@postgresql.org) (archive (http://www.PostgreSQL.ORG/mhonarc/pgsql-general/)) mailing list is a good place to have user questions answered. Other mailing lists are available; consult the Info Central section of the PostgreSQL web site for details.

Yourself!

Postgres is an open source product. As such, it depends on the user community for ongoing support. As you begin to use Postgres, you will rely on others for help, either through the documentation or through the mailing lists. Consider contributing your knowledge back. If you learn something which is not in the documentation, write it up and contribute it. If you add features to the code, contribute it.

Even those without a lot of experience can provide corrections and minor changes in the documentation, and that is a good way to start. The pgsql-docs (mailto:pgsql-docs@postgresql.org) (archive (http://www.PostgreSQL.ORG/mhonarc/pgsql-docs/)) mailing list is the place to get going.

Terminology

In the following documentation, *site* may be interpreted as the host machine on which Postgres is installed. Since it is possible to install more than one set of Postgres databases on a single host, this term more precisely denotes any particular set of installed Postgres binaries and databases.

The Postgres *superuser* is the user named *`postgres`* who owns the Postgres binaries and database files. As the database superuser, all protection mechanisms may be bypassed and any data accessed arbitrarily. In addition, the Postgres superuser is allowed to execute some support programs which are generally not available to all users. Note that the Postgres superuser is *not* the same as the Unix superuser (which will be referred to as *root*). The superuser should have a non-zero user identifier (*UID*) for security reasons.

The *database administrator* or DBA, is the person who is responsible for installing Postgres with mechanisms to enforce a security policy for a site. The DBA can add new users by the method described below and maintain a set of template databases for use by createdb.

The postmaster is the process that acts as a clearing-house for requests to the Postgres system. Frontend applications connect to the postmaster, which keeps tracks of any system errors and communication between the backend processes. The postmaster can take several command-line arguments to tune its behavior. However, supplying arguments is necessary only if you intend to run multiple sites or a non-default site.

The Postgres backend (the actual executable program postgres) may be executed directly from the user shell by the Postgres super-user (with the database name as an argument).

However, doing this bypasses the shared buffer pool and lock table associated with a postmaster/site, therefore this is not recommended in a multiuser site.

Notation

`"..."` or `/usr/local/pgsql/` at the front of a file name is used to represent the path to the Postgres superuser's home directory.

In a command synopsis, brackets (`"["` and `"]"`) indicate an optional phrase or keyword. Anything in braces (`"{"` and `"}"`) and containing vertical bars (`"|"`) indicates that you must choose one.

In examples, parentheses (`"("` and `")"`) are used to group boolean expressions. `"|"` is the boolean operator OR.

Examples will show commands executed from various accounts and programs. Commands executed from the root account will be preceeded with ">". Commands executed from the Postgres superuser account will be preceeded with "%", while commands executed from an unprivileged user's account will be preceeded with "$". SQL commands will be preceeded with "=>" or will have no leading prompt, depending on the context.

> **Note:** At the time of writing (Postgres v7.0) the notation for flagging commands is not universally consistant throughout the documentation set. Please report problems to the Documentation Mailing List (mailto:docs@postgresql.org).

Problem Reporting Guidelines

When you encounter a problem in PostgreSQL we want to hear about it. Your bug reports are an important part in making PostgreSQL more reliable because even the utmost care cannot guarantee that every part of PostgreSQL will work on every platform under every circumstance.

The following suggestions are intended to assist you in forming bug reports that can be handled in an effective fashion. No one is required to follow them but it tends to be to everyone's advantage.

We cannot promise to fix every bug right away. If the bug is obvious, critical, or affects a lot of users, chances are good that someone will look into it. It could also happen that we tell you to update to a newer version to see if the bug happens there. Or we might decide that the bug cannot be fixed before some major rewrite we might be planning is done. Or perhaps it's simply too hard and there are more important things on the agenda. If you need help immediately, consider obtaining a commercial support contract.

Identifying Bugs

Before you ask "Is this a bug?", please read and re-read the documentation to verify that you can really do whatever it is you are trying. If it is not clear from the documentation whether you can do something or not, please report that too; it's a bug in the documentation. If it turns out that the program does something different from what the documentation says, that's a bug. That might include, but is not limited to, the following circumstances:

- A program terminates with a fatal signal or an operating system error message that would point to a problem in the program (a counterexample might be a "disk full" message, since that must be fixed outside of Postgres).

- A program produces the wrong output for any given input.
- A program refuses to accept valid input.
- A program accepts invalid input without a notice or error message.
- PostgreSQL fails to compile, build, or install according to the instructions on supported platforms.

Here "`program`" refers to any executable, not only the backend server.

Being slow or resource-hogging is not necessarily a bug. Read the documentation or ask on one of the mailing lists for help in tuning your applications. Failing to comply to SQL is not a bug unless compliance for the specific feature is explicitly claimed.

Before you continue, check on the TODO list and in the FAQ to see if your bug is already known. If you can't decode the information on the TODO list, report your problem. The least we can do is make the TODO list clearer.

What to report

The most important thing to remember about bug reporting is to state all the facts and only facts. Do not speculate what you think went wrong, what "it seemed to do", or which part of the program has a fault. If you are not familiar with the implementation you would probably guess wrong and not help us a bit. And even if you are, educated explanations are a great supplement to but no substitute for facts. If we are going to fix the bug we still have to see it happen for ourselves first. Reporting the bare facts is relatively straightforward (you can probably copy and paste them from the screen) but all too often important details are left out because someone thought it doesn't matter or the report would be understood anyway.

The following items should be contained in every bug report:

- The exact sequence of steps *from program startup* necessary to reproduce the problem. This should be self-contained; it is not enough to send in a bare select statement without the preceeding create table and insert statements, if the output should depend on the data in the tables. We do not have the time to decode your database schema, and if we are supposed to make up our own data we would probably miss the problem. The best format for a test case for query-language related problems is a file that can be run through the psql frontend that shows the problem. (Be sure to not have anything in your `~/.psqlrc` startup file.) You are encouraged to minimize the size of your example, but this is not absolutely necessary. If the bug is reproduceable, we'll find it either way.

  If your application uses some other client interface, such as PHP, then please try to isolate the offending queries. We probably won't set up a web server to reproduce your problem. In any case remember to provide the exact input files, do not guess that the problem happens for "large files" or "mid-size databases", etc. since this information is too inexact to be of use.

- The output you got. Please do not say that it "didn't work" or "failed". If there is an error message, show it, even if you don't understand it. If the program terminates with an operating system error, say which. If nothing at all happens, say so. Even if the result of your test case is a program crash or otherwise obvious it might not happen on our platform. The easiest thing is to copy the output from the terminal, if possible.

  Note: In case of fatal errors, the error message provided by the client might not contain all the information available. In that case, also look at the output of the database server. If you do not keep your server output, this would be a good time to start doing so.

The output you expected is very important to state. If you just write "This command gives me that output." or "This is not what I expected.", we might run it ourselves, scan the output, and think it looks okay and is exactly what we expected. We shouldn’t have to spend the time to decode the exact semantics behind your commands. Especially refrain from merely saying that "This is not what SQL says/Oracle does." Digging out the correct behavior from SQL is not a fun undertaking, nor do we all know how all the other relational databases out there behave. (If your problem is a program crash you can obviously omit this item.)

Any command line options and other startup options, including concerned environment variables or configuration files that you changed from the default. Again, be exact. If you are using a pre-packaged distribution that starts the database server at boot time, you should try to find out how that is done.

Anything you did at all differently from the installation instructions.

The PostgreSQL version. You can run the command `SELECT version();` to find out what version you are currently running. If this function does not exist, say so, then we know that your version is old enough. If you can’t start up the server or a client, look into the README file in the source directory or at the name of your distribution file or package name. If your version is older than 7.0 we will almost certainly tell you to upgrade. There are tons of bug fixes in each new version, that’s why we write them.

If you run a pre-packaged version, such as RPMs, say so, including any subversion the package may have. If you are talking about a CVS snapshot, mention that, including its date and time.

Platform information. This includes the kernel name and version, C library, processor, memory information. In most cases it is sufficient to report the vendor and version, but do not assume everyone knows what exactly "Debian" contains or that everyone runs on Pentiums. If you have installation problems information about compilers, make, etc. is also necessary.

Do not be afraid if your bug report becomes rather lengthy. That is a fact of life. It’s better to report everything the first time than us having to squeeze the facts out of you. On the other hand, if your input files are huge, it is fair to ask first whether somebody is interested in looking into it.

Do not spend all your time to figure out which changes in the input make the problem go away. This will probably not help solving it. If it turns out that the bug can’t be fixed right away, you will still have time to find and share your work around. Also, once again, do not waste your time guessing why the bug exists. We’ll find that out soon enough.

When writing a bug report, please choose non-confusing terminology. The software package as such is called "PostgreSQL", sometimes "Postgres" for short. (Sometimes the abbreviation "Pgsql" is used but don’t do that.) When you are specifically talking about the backend server, mention that, don’t just say "Postgres crashes". The interactive frontend is called "psql" and is for all intends and purposes completely separate from the backend.

Where to report bugs

In general, send bug reports to the bug report mailing list (mailto:pgsql-bugs@postgresql.org). You are invited to find a descriptive subject for your email message, perhaps parts of the error message.

Do not send bug reports to any of the user mailing lists, such as the SQL language mailing list (mailto:pgsql-sql@postgresql.org) or the general topics mailing list (mailto:pgsql-general@postgresql.org). These mailing lists are for answering user questions and their subscribers normally do not wish to receive bug reports. More importantly, they are unlikely to fix them.

Also, please do *not* send reports to the developers' mailing list (mailto:pgsql-hackers@postgresql.org). This list is for discussing the development of PostgreSQL and it would be nice if we could keep the bug reports separate. We might choose to take up a discussion about your bug report on it, if the bug needs more review.

If you have a problem with the documentation, send email to the documentation mailing list (mailto:pgsql-docs@postgresql.org). Mention the document, chapter, and sections in your problem report.

If your bug is a portability problem on a non-supported platform, send mail to the porting issues mail list (mailto:pgsql-ports@postgresql.org), so we (and you) can work on porting PostgreSQL to your platform.

> **Note:** Due to the unfortunate amount of spam going around, all of the above email addresses are closed mailing lists. That is, you need to be subscribed to them in order to be allowed to post. If you simply want to send mail but do not want to receive list traffic, you can subscribe to the special pgsql-loophole mailing list, which allows you to post to all PostgreSQL mailing lists without receiving any messages. Send email to pgsql-loophole-request@postgresql.org (mailto:pgsql-loophole-request@postgresql.org) to subscribe.

Y2K Statement

> **Author:** Written by Thomas Lockhart (mailto:lockhart@alumni.caltech.edu) on 1998-10-22. Updated 2000-03-31.

The PostgreSQL Global Development Team provides the Postgres software code tree as a public service, without warranty and without liability for it's behavior or performance. However, at the time of writing:

- The author of this statement, a volunteer on the Postgres support team since November, 1996, is not aware of any problems in the Postgres code base related to time transitions around Jan 1, 2000 (Y2K).
- The author of this statement is not aware of any reports of Y2K problems uncovered in regression testing or in other field use of recent or current versions of Postgres. We might have expected to hear about problems if they existed, given the installed base and the active participation of users on the support mailing lists.
- To the best of the author's knowledge, the assumptions Postgres makes about dates specified with a two-digit year are documented in the current User's Guide (http://www.postgresql.org/docs/user/datatype.htm) in the chapter on data types. For two-digit years, the significant transition year is 1970, not 2000; e.g. "`70-01-01`" is interpreted as 1970-01-01, whereas "`69-01-01`" is interpreted as 2069-01-01.
- Any Y2K problems in the underlying OS related to obtaining "the current time" may propagate into apparent Y2K problems in Postgres.

Refer to The Gnu Project (http://www.gnu.org/software/year2000.html) and The Perl Institute (http://language.perl.com/news/y2k.html) for further discussion of Y2K issues, particularly as it relates to open source, no fee software.

Copyrights and Trademarks

Chapter 2. Ports

This manual describes version 7.0 of Postgres. The Postgres developer community has compiled and tested Postgres on a number of platforms. Check the web site (http://www.postgresql.org/docs/admin/ports.htm) for the latest information.

Currently Supported Platforms

At the time of publication, the following platforms have been tested:

Table 2-1. Supported Platforms

OS	Processor	Version	Reported	Remarks
AIX 4.3.2	RS6000	v7.0	2000-04-05	Andreas Zeugswetter (mailto:Andreas.Zeugswetter@telecom.at)
BSDI 4.01	x86	v7.0	2000-04-04	Bruce Momjian (mailto:maillist@candle.pha.pa.us)
Compaq Tru64 5.0	Alpha	v7.0	2000-04-11	Andrew McMurry (mailto:andrew.mcmurry@astro.uio.no)
FreeBSD 4.0	x86	v7.0	2000-04-04	Marc Fournier (mailto:scrappy@hub.org)
HPUX	PA-RISC	v7.0	2000-04-12	Both 9.0x and 10.20. Tom Lane (mailto:tgl@sss.pgh.pa.us)
IRIX 6.5.6f	MIPS	v6.5.3	2000-02-18	MIPSPro 7.3.1.1m; full N32 build. Kevin Wheatley (hxpro@cinesite.co.uk)
Linux 2.0.x	Alpha	v7.0	2000-04-05	With patches; Ryan Kirkpatrick (mailto:pgsql@rkirkpat.net)
Linux 2.2.x	armv4l	v7.0	2000-04-17	Regression test needs work. Mark Knox (mailto:segfault@hardline.org)
Linux 2.2.x	x86	v7.0	2000-03-26	Lamar Owens (mailto:lamar.owen@wgcr.org)
Linux 2.0.x	MIPS	v7.0	2000-04-13	Cobalt Qube. Tatsuo Ishii (mailto:t-ishii@sra.co.jp)
Linux 2.2.5	Sparc	v7.0	2000-04-02	Tom Szybist (mailto:szybist@boxhill.com)
LinuxPPC R4	PPC603e	v7.0	2000-04-13	Tatsuo Ishii (mailto:t-ishii@sra.co.jp)
mklinux	PPC750	v7.0	2000-04-13	Tatsuo Ishii (mailto:t-ishii@sra.co.jp)
NetBSD 1.4	arm32	v7.0	2000-04-08	Patrick Welche (mailto:prlw1@newn.cam.ac.uk)
NetBSD 1.4U	x86	v7.0	2000-03-26	Patrick Welche (mailto:prlw1@newn.cam.ac.uk)

OS	Processor	Version	Reported	Remarks
NetBSD	m68k	v7.0	2000-04-10	Mac 8xx. Henry B. Hotz (mailto:hotz@jpl.nasa.gov)
NetBSD/sparc	Sparc	v7.0	2000-04-13	Tom I Helbekkmo (mailto:tih@kpnQwest.no)
QNX 4.25	x86	v7.0	2000-04-01	Dr. Andreas Kardos (mailto:kardos@repas-aeg.de)
SCO OpenServer 5	x86	v6.5	1999-05-25	Andrew Merrill (mailto:andrew@compclass.com)
SCO UnixWare 7	x86	v7.0	2000-04-18	See FAQ; needs patch for compiler bug. Billy G. Allie (mailto:Bill.Allie@mug.org)
Solaris	x86	v7.0	2000-04-12	Marc Fournier (mailto:scrappy@hub.org)
Solaris 2.5.1-2.7	Sparc	v7.0	2000-04-12	Peter Eisentraut (mailto:peter_e@gmx.net), Marc Fournier (mailto:scrappy@hub.org)
SunOS 4.1.4	Sparc	v7.0	2000-04-13	Tatsuo Ishii (mailto:t-ishii@sra.co.jp)
Windows/ Win32	x86	v7.0	2000-04-02	Client-side libraries or ODBC/JDBC. Magnus Hagander (mha@sollentuna.net)
WinNT/ Cygwin	x86	v7.0	2000-03-30	Working with the Cygwin library; Daniel Horak (mailto:horak@sit.plzen-city.cz))

Note: For Windows NT, the server-side port of Postgres uses the RedHat/Cygnus Cygwin library and toolset.

Unsupported Platforms

Platforms listed for v6.3.x-v6.5.x should also work with v7.0, but we did not receive explicit confirmation of such at the time this list was compiled. We include these here to let you know that these platforms *could* be supported if given some attention.

At the time of publication, the following platforms have not been tested for v7.0 or v6.5.x:

Table 2-2. Unsupported Platforms

OS	Processor	Version	Reported	Remarks
BeOS	x86	v7.0	2000-05-01	Client-side coming soon? Adam Haberlach (adam@newsnipple.com)
DGUX 5.4R4.11	m88k	v6.3	1998-03-01	v6.4 probably OK. Needs new maintainer. Brian E Gallew (mailto:geek+@cmu.edu)
NetBSD-current	NS32532	v6.4	1998-10-27	small problems in date/time math. Jon Buller (mailto:jonb@metronet.com)
NetBSD 1.3	VAX	v6.3	2000-05-01	v7.0 likely to work. Tom I Helbekkmo (mailto:tih@kpnQwest.no)
SVR4 4.4	m88k	v6.2.1	1998-03-01	v6.4.x will need TAS spinlock code Doug Winterburn (mailto:dlw@seavme.xroads.com))
SVR4	MIPS	v6.4	1998-10-28	No 64-bit int. Frank Ridderbusch (mailto:ridderbusch.pad@sni.de))
Ultrix	MIPS,VAX	v6.x	1998-03-01	No recent reports; obsolete?

There are a few platforms which have been attempted and which have been reported to not work with the standard distribution. Others listed here do not provide sufficient library support for an attempt.

Table 2-3. Incompatible Platforms

OS	Processor	Version	Reported	Remarks
MacOS	all	all	2000-05-01	Not library compatible; use ODBC/JDBC. X possible?
NextStep	x86	all	1998-03-01	Client-only v1.0.9 worked with patches David Wetzel (mailto:dave@turbocat.de)

Chapter 3. Configuration Options

Parameters for Configuration (configure)

The full set of parameters available in configure can be obtained by typing

```
$ ./configure --help
```

The following parameters may be of interest to installers:

```
Directories to install PostgreSQL in:
  --prefix=PREFIX         install architecture-independent files in
PREFIX
                          [/usr/local/pgsql]
  --bindir=DIR            user executables in DIR [EPREFIX/bin]
  --libdir=DIR            object code libraries in DIR [EPREFIX/lib]
  --includedir=DIR        C header files in DIR [PREFIX/include]
  --mandir=DIR            man documentation in DIR [PREFIX/man]
Features and packages:
  --disable-FEATURE       do not include FEATURE (same as
--enable-FEATURE=no)
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as
--with-PACKAGE=no)
--enable and --with options recognized:
  --with-template=template
                          use operating system template file
                              see template directory
  --with-includes=dirs    look for header files for tcl/tk, etc in
DIRS
  --with-libraries=dirs   look for additional libraries in DIRS
  --with-libs=dirs        alternate spelling of --with-libraries
  --enable-locale         enable locale support
  --enable-recode         enable cyrillic recode support
  --enable-multibyte      enable multibyte character support
  --with-pgport=portnum   change default postmaster port
  --with-maxbackends=n    set default maximum number of server
processes
  --with-tcl              build Tcl interfaces and pgtclsh
  --with-tclconfig=tcldir
                          tclConfig.sh and tkConfig.sh are in DIR
  --with-perl             build Perl interface and plperl
  --with-odbc             build ODBC driver package
  --with-odbcinst=odbcdir
                          change default directory for odbcinst.ini
  --enable-cassert        enable assertion checks (for debugging)
  --enable-debug          build with debugging symbols (-g)
  --with-CC=compiler
                          use specific C compiler
  --with-CXX=compiler
                          use specific C++ compiler
  --without-CXX           prevent building C++ code
```

Some systems may have trouble building a specific feature of Postgres. For example, systems with a damaged C++ compiler may need to specify `--without-CXX` to instruct the build procedure to skip construction of `libpq++`.

Use the `--with-includes` and `--with-libraries` options if you want to build Postgres using include files or libraries that are not installed in your system's standard search path. For example, you might use these to build with an experimental version of Tcl. If you need to specify more than one nonstandard directory for include files or libraries, do it like this:

```
--with-includes="/opt/tcl/include /opt/perl5/include"
```

Parameters for Building (make)

Many installation-related parameters can be set in the building stage of Postgres installation.

In most cases, these parameters should be placed in a file, `Makefile.custom`, intended just for that purpose. The default distribution does not contain this optional file, so you will create it using a text editor of your choice. When upgrading installations, you can simply copy your old Makefile.custom to the new installation before doing the build.

Alternatively, you can set variables on the make command line:

```
make [ variable=value [...] ]
```

A few of the many variables that can be specified are:

POSTGRESDIR

Top of the installation tree.

BINDIR

Location of applications and utilities.

LIBDIR

Location of object libraries, including shared libraries.

HEADERDIR

Location of include files.

ODBCINST

Location of installation-wide psqlODBC (ODBC) configuration file.

There are other optional parameters which are not as commonly used. Many of those listed below are appropriate when doing Postgres server code development.

CFLAGS

Set flags for the C compiler. Should be assigned with "+=" to retain relevant default parameters.

YFLAGS

Set flags for the yacc/bison parser. `-v` might be used to help diagnose problems building a new parser. Should be assigned with "+=" to retain relevant default parameters.

USE_TCL

Enable Tcl interface building.

HSTYLE

DocBook HTML style sheets for building the documentation from scratch. Not used unless you are developing new documentation from the DocBook-compatible SGML source documents in `doc/src/sgml/`.

PSTYLE

DocBook style sheets for building printed documentation from scratch. Not used unless you are developing new documentation from the DocBook-compatible SGML source documents in `doc/src/sgml/`.

Here is an example `Makefile.custom` for a PentiumPro Linux system:

```
# Makefile.custom
# Thomas Lockhart 1999-06-01

POSTGRESDIR= /opt/postgres/current
CFLAGS+= -m486 -O2

# documentation

HSTYLE= /home/tgl/SGML/db118.d/docbook/html
PSTYLE= /home/tgl/SGML/db118.d/docbook/print
```

Locale Support

> **Note:** Written by Oleg Bartunov. See Oleg’s web page (http://www.sai.msu.su/~megera/postgres/) for additional information on locale and Russian language support.

While doing a project for a company in Moscow, Russia, I encountered the problem that postgresql had no support of national alphabets. After looking for possible workarounds I decided to develop support of locale myself. I’m not a C-programer but already had some experience with locale programming when I work with perl (debugging) and glimpse. After several days of digging through the Postgres source tree I made very minor corections to src/backend/utils/adt/varlena.c and src/backend/main/main.c and got what I needed! I did support only for LC_CTYPE and LC_COLLATE, but later LC_MONETARY was added by others. I got many messages from people about this patch so I decided to send it to developers and (to my surprise) it was incorporated into the Postgres distribution.

People often complain that locale doesn’t work for them. There are several common mistakes:

Didn’t properly configure postgresql before compilation. You must run configure with --enable-locale option to enable locale support.

Didn’t setup environment correctly when starting postmaster. You must define environment variables LC_CTYPE and LC_COLLATE before running postmaster because backend gets information about locale from environment. I use following shell script (runpostgres):

```
#!/bin/sh

export LC_CTYPE=koi8-r
export LC_COLLATE=koi8-r
postmaster -B 1024 -S -D/usr/local/pgsql/data/ -o ’-Fe’
```

and run it from rc.local as

```
/bin/su - postgres -c "/home/postgres/runpostgres"
```

Broken locale support in OS (for example, locale support in libc under Linux several times has changed and this caused a lot of problems). Latest perl has also support of locale and if locale is broken **perl -v** will complain something like:

```
8:17[mira]:~/WWW/postgres>setenv LC_CTYPE not_exist
8:18[mira]:~/WWW/postgres>perl -v
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
LC_ALL = (unset),
    LC_CTYPE = "not_exist",
    LANG = (unset)
are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
```

Wrong location of locale files! Possible locations include: `/usr/lib/locale` (Linux, Solaris), `/usr/share/locale` (Linux), `/usr/lib/nls/loc` (DUX 4.0). Check **man locale** to find the correct location. Under Linux I did a symbolic link between `/usr/lib/locale` and `/usr/share/locale` to be sure that the next libc will not break my locale.

What are the Benefits?

You can use ~* and order by operators for strings contain characters from national alphabets. Non-english users definitely need that. If you won't use locale stuff just undefine the USE_LOCALE variable.

What are the Drawbacks?

There is one evident drawback of using locale - its speed! So, use locale only if you really need it.

Kerberos Authentication

Kerberos is an industry-standard secure authentication system suitable for distributed computing over a public network.

Availability

The Kerberos authentication system is not distributed with Postgres. Versions of Kerberos are typically available as optional software from operating system vendors. In addition, a source code distribution may be obtained through MIT Project Athena (ftp://athena-dist.mit.edu).

> **Note:** You may wish to obtain the MIT version even if your vendor provides a version, since some vendor ports have been deliberately crippled or rendered non-interoperable with the MIT version.

Users located outside the United States of America and Canada are warned that distribution of the actual encryption code in Kerberos is restricted by U. S. Government export regulations.

Inquiries regarding your Kerberos should be directed to your vendor or MIT Project Athena (info-kerberos@athena.mit.edu). Note that FAQLs (Frequently-Asked Questions Lists) are periodically posted to the Kerberos mailing list (mailto:kerberos@athena.mit.edu), subscribe to (mailto:kerberos-request@athena.mit.edu), and USENET news group (news:comp.protocols.kerberos).

Installation

Installation of Kerberos itself is covered in detail in the *Kerberos Installation Notes* . Make sure that the server key file (the `srvtab` or `keytab`) is somehow readable by the Postgres account.

Postgres and its clients can be compiled to use either Version 4 or Version 5 of the MIT Kerberos protocols by setting the KRBVERS variable in the file `src/Makefile.global` to the appropriate value. You can also change the location where Postgres expects to find the associated libraries, header files and its own server key file.

After compilation is complete, Postgres must be registered as a Kerberos service. See the *Kerberos Operations Notes* and related manual pages for more details on registering services.

Operation

After initial installation, Postgres should operate in all ways as a normal Kerberos service. For details on the use of authentication, see the *PostgreSQL User's Guide* reference sections for postmaster and psql.

In the Kerberos Version 5 hooks, the following assumptions are made about user and service naming:

- User principal names (anames) are assumed to contain the actual Unix/Postgres user name in the first component.
- The Postgres service is assumed to be have two components, the service name and a hostname, canonicalized as in Version 4 (i.e., with all domain suffixes removed).

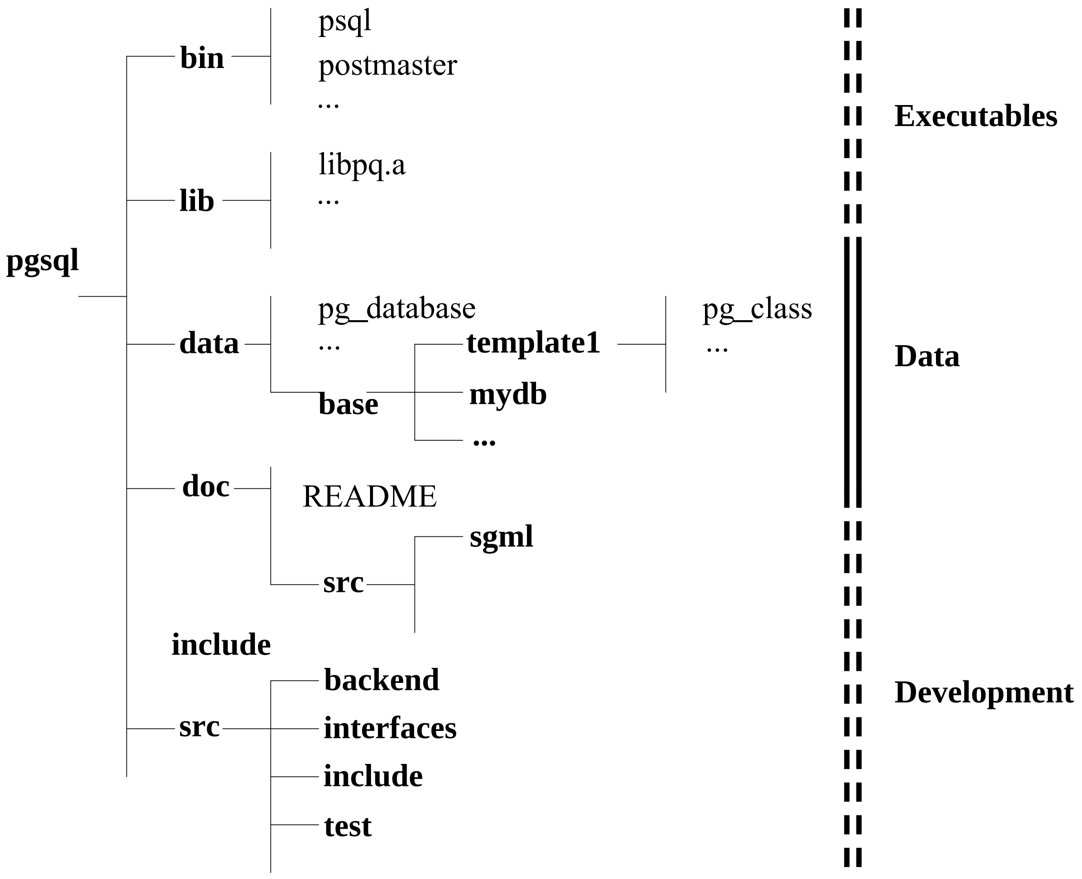
Table 3-1. Kerberos Parameter Examples

Parameter	Example
user	frew@S2K.ORG
user	aoki/HOST=miyu.S2K.Berkeley.EDU@S2K.ORG
host	postgres_dbms/ucbvax@S2K.ORG

Support for Version 4 will disappear sometime after the production release of Version 5 by MIT.

Chapter 4. System Layout

Figure 4-1. Postgres file layout



Postgres file layout shows how the Postgres distribution is laid out when installed in the default way. For simplicity, we will assume that Postgres has been installed in the directory `/usr/local/pgsql`. Therefore, wherever you see the directory `/usr/local/pgsql` you should substitute the name of the directory where Postgres is actually installed. All Postgres commands are installed in the directory `/usr/local/pgsql/bin`. Therefore, you should add this directory to your shell command path. If you use a variant of the Berkeley C shell, such as csh or tcsh, you would add

```
set path = ( /usr/local/pgsql/bin path )
```

in the .login file in your home directory. If you use a variant of the Bourne shell, such as sh, ksh, or bash, then you would add

```
PATH=/usr/local/pgsql/bin:$PATH
export PATH
```

to the .profile file in your home directory. From now on, we will assume that you have added the Postgres bin directory to your path. In addition, we will make frequent reference to "setting a shell variable" or "setting an environment variable" throughout this document. If you did not fully understand the last paragraph on modifying your search path, you should consult the Unix manual pages that describe your shell before going any further.

If you have not set things up in the default way, you may have some more work to do. For example, if the database server machine is a remote machine, you will need to set the PGHOST environment variable to the name of the database server machine. The environment variable PGPORT may also have to be set. The bottom line is this: if you try to start an application program and it complains that it cannot connect to the postmaster, you must go back and make sure that your environment is properly set up.

Chapter 5. Installation

Installation instructions for PostgreSQL 7.0.

If you haven't gotten the PostgreSQL distribution, get it from ftp.postgresql.org (ftp://ftp.postgresql.org), then unpack it:

```
> gunzip postgresql-7.0.tar.gz
> tar -xf postgresql-7.0.tar
> mv postgresql-7.0 /usr/src
```

Before you start

Building PostgreSQL requires GNU make. It will *not* work with other make programs. On GNU/Linux systems GNU make is the default tool, on other systems you may find that GNU make is installed under the name `gmake`. We will use that name from now on to indicate GNU make, no matter what name it has on your system. To test for GNU make enter

```
> gmake --version
```

If you need to get GNU make, you can find it at ftp://ftp.gnu.org.

Up to date information on supported platforms is at http://www.postgresql.org/docs/admin/ports.htm (http://www.postgresql.org/docs/admin/ports.htm). In general, most Unix-compatible platforms with modern libraries should be able to run PostgreSQL. In the `doc` subdirectory of the distribution are several platform-specific FAQ and README documents you might wish to consult if you are having trouble.

Although the minimum required memory for running PostgreSQL can be as little as 8MB, there are noticeable speed improvements when expanding memory up to 96MB or beyond. The rule is you can never have too much memory.

Check that you have sufficient disk space. You will need about 30 Mbytes for the source tree during compilation and about 5 Mbytes for the installation directory. An empty database takes about 1 Mbyte, otherwise they take about five times the amount of space that a flat text file with the same data would take. If you run the regression tests you will temporarily need an extra 20MB.

To check for disk space, use

```
> df -k
```

Considering today's prices for hard disks, getting a large and fast hard disk should probably be in your plans before putting a database into production use.

Installation Procedure

PostgreSQL Installation

For a fresh install or upgrading from previous releases of PostgreSQL:

1. Create the PostgreSQL superuser account. This is the user the server will run as. For production use you should create a separate, unprivileged account (`postgres` is commonly used). If you do not have root access or just want to play around, your own user account is enough.

   Running PostgreSQL as `root`, `bin`, or any other account with special access rights is a security risk; *don't do it*. The postmaster will in fact refuse to start as `root`.

   You need not do the building and installation itself under this account (although you can). You will be told when you need to login as the database superuser.

2. Configure the source code for your system. It is this step at which you can specify your actual installation path for the build process and make choices about what gets installed. Change into the `src` subdirectory and type:

   ```
   > ./configure
   ```

   followed by any options you might want to give it. For a first installation you should be able to do fine without any. For a complete list of options, type:

   ```
   > ./configure --help
   ```

   Some of the more commonly used ones are:

   --prefix=BASEDIR

   Selects a different base directory for the installation of PostgreSQL. The default is `/usr/local/pgsql`.

   --enable-locale

   If you want to use locales.

   --enable-multibyte

   Allows the use of multibyte character encodings. This is primarily for languages like Japanese, Korean, or Chinese.

   --with-perl

   Builds the Perl interface and plperl extension language. Please note that the Perl interface needs to be installed into the usual place for Perl modules (typically under `/usr/lib/perl`), so you must have root access to perform the installation step. (It is often easiest to leave out `--with-perl` initially, and then build and install the Perl interface after completing the installation of PostgreSQL itself.)

   --with-odbc

   Builds the ODBC driver package.

--with-tcl

Builds interface libraries and programs requiring Tcl/Tk, including libpgtcl, pgtclsh, and pgtksh.

3. Compile the program. Type

   ```
   > gmake
   ```

   The compilation process can take anywhere from 10 minutes to an hour. Your mileage will most certainly vary. Remember to use GNU make.

   The last line displayed will hopefully be

   ```
   All of PostgreSQL is successfully made. Ready to install.
   ```

4. If you want to test the newly built server before you install it, you can run the regression tests at this point. The regression tests are a test suite to verify that PostgreSQL runs on your machine in the way the developers expected it to. For detailed instructions see *Regression Test*. (Be sure to use the "parallel regress test" method, since the sequential method only works with an already-installed server.)
5. If you are not upgrading an existing system then skip to step 7.

   You now need to back up your existing database. To dump your fairly recent post-6.0 database installation, type

   ```
   > pg_dumpall > db.out
   ```

   If you wish to preserve object id's (oids), then use the -o option when running pg_dumpall. However, unless you have a special reason for doing this (such as using OIDs as keys in tables), don't do it.

   Make sure to use the pg_dumpall command from the version you are currently running. 7.0's pg_dumpall will not work on older databases. However, if you are still using 6.0, do not use the pg_dumpall script from 6.0 or everything will be owned by the PostgreSQL superuser after you reload. In that case you should grab pg_dumpall from a later 6.x.x release. If you are upgrading from a version prior to Postgres95 v1.09 then you must back up your database, install Postgres95 v1.09, restore your database, then back it up again.

   Caution

   You must make sure that your database is not updated in the middle of your backup. If necessary, bring down postmaster, edit the permissions in file `/usr/local/pgsql/data/pg_hba.conf` to allow only you on, then bring postmaster back up.

6. If you are upgrading an existing system then kill the database server now. Type

   ```
   > ps ax | grep postmaster
   ```

or

```
> ps -e | grep postmaster
```

(It depends on your system which one of these two works. No harm can be done by typing the wrong one.) This should list the process numbers for a number of processes, similar to this:

```
  263  ?  SW   0:00 (postmaster)
  777  p1 S    0:00 grep postmaster
```

Type the following line, with *`pid`* replaced by the process id for process `postmaster` (263 in the above case). (Do not use the id for the process "grep postmaster".)

```
> kill pid
```

> **Tip:** On systems which have PostgreSQL started at boot time, there is probably a startup file that will accomplish the same thing. For example, on a Redhat Linux system one might find that
>
> ```
> > /etc/rc.d/init.d/postgres.init stop
> ```
>
> works.

Also move the old directories out of the way. Type the following:

```
> mv /usr/local/pgsql /usr/local/pgsql.old
```

(substitute your particular paths).

7. Install the PostgreSQL executable files and libraries. Type

   ```
   > gmake install
   ```

   You should do this step as the user that you want the installed executables to be owned by. This does not have to be the same as the database superuser; some people prefer to have the installed files be owned by root.

8. If necessary, tell your system how to find the new shared libraries. How to do this varies between platforms. The most widely usable method is to set the environment variable LD_LIBRARY_PATH:

   ```
   > LD_LIBRARY_PATH=/usr/local/pgsql/lib
   > export LD_LIBRARY_PATH
   ```

   on sh, ksh, bash, zsh or

   ```
   > setenv LD_LIBRARY_PATH /usr/local/pgsql/lib
   ```

   on csh or tcsh. You might want to put this into a shell startup file such as `/etc/profile`.

On some systems the following is the preferred method, but you must have root access. Edit file `/etc/ld.so.conf` to add a line

```
/usr/local/pgsql/lib
```

Then run command **/sbin/ldconfig**.

If in doubt, refer to the manual pages of your system. If you later on get a message like

```
psql: error in loading shared libraries
libpq.so.2.1: cannot open shared object file: No such file or
directory
```

then the above was necessary. Simply do this step then.

9. Create the database installation (the working data files). To do this you must log in to your PostgreSQL superuser account. It will not work as root.

   ```
   > mkdir /usr/local/pgsql/data
   > chown postgres /usr/local/pgsql/data
   > su - postgres
   > /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
   ```

   The `-D` option specifies the location where the data will be stored. You can use any path you want, it does not have to be under the installation directory. Just make sure that the superuser account can write to the directory (or create it, if it doesn't already exist) before starting **initdb**. (If you have already been doing the installation up to now as the PostgreSQL superuser, you may have to log in as root temporarily to create the data directory underneath a root-owned directory.)

10. The previous step should have told you how to start up the database server. Do so now. The command should look something like

    ```
    > /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data
    ```

    This will start the server in the foreground. To make it detach to the background, you can use the `-S` option, but then you won't see any log messages the server produces. A better way to put the server in the background is

    ```
    > nohup /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data
    \
        </dev/null >>server.log 2>>1 &
    ```

11. If you are upgrading from an existing installation, dump your data back in:

    ```
    > /usr/local/pgsql/bin/psql -d template1 -f db.out
    ```

    You also might want to copy over the old `pg_hba.conf` file and any other files you might have had set up for authentication, such as password files.

This concludes the installation proper. To make your life more productive and enjoyable you should look at the following optional steps and suggestions.

Life will be more convenient if you set up some environment variables. First of all you probably want to include `/usr/local/pgsql/bin` (or equivalent) into your PATH. To do this, add the following to your shell startup file, such as `~/.bash_profile` (or `/etc/profile`, if you want it to affect every user):

```
> PATH=$PATH:/usr/local/pgsql/bin
```

Furthermore, if you set PGDATA in the environment of the PostgreSQL superuser, you can omit the `-D` for `postmaster` and `initdb`.

You probably want to install the man and HTML documentation. Type

```
> cd /usr/src/pgsql/postgresql-7.0/doc
> gmake install
```

This will install files under `/usr/local/pgsql/doc` and `/usr/local/pgsql/man`. To enable your system to find the man documentation, you need to add a line like the following to a shell startup file:

```
> MANPATH=$MANPATH:/usr/local/pgsql/man
```

The documentation is also available in Postscript format. If you have a Postscript printer, or have your machine already set up to accept Postscript files using a print filter, then to print the User’s Guide simply type

```
> cd /usr/local/pgsql/doc
> gunzip -c user.ps.tz | lpr
```

Here is how you might do it if you have Ghostscript on your system and are writing to a laserjet printer.

```
> gunzip -c user.ps.gz \
    | gs -sDEVICE=laserjet -r300 -q -dNOPAUSE -sOutputFile=- \
    | lpr
```

Printer setups can vary wildly from system to system. If in doubt, consult your manuals or your local expert.

The Adminstrator’s Guide should probably be your first reading if you are completely new to PostgreSQL, as it contains information about how to set up database users and authentication.

Usually, you will want to modify your computer so that it will automatically start the database server whenever it boots. This is not required; the PostgreSQL server can be run successfully from non-privileged accounts without root intervention.

Different systems have different conventions for starting up daemons at boot time, so you are advised to familiarize yourself with them. Most systems have a file `/etc/rc.local` or `/etc/rc.d/rc.local` which is almost certainly no bad place to put such a command. Whatever you do, postmaster must be run by the PostgreSQL superuser (`postgres`) *and not by root* or any other user. Therefore you probably always want to form your command lines along the lines of `su -c '...' postgres`.

It might be advisable to keep a log of the server output. To start the server that way try:

```
> nohup su -c ’postmaster -D /usr/local/pgsql/data > server.log
2>&1’ postgres &
```

Here are a few more operating system specific suggestions.

Edit file rc.local on NetBSD or file rc2.d on SPARC Solaris 2.5.1 to contain the following single line:

```
> su postgres -c "/usr/local/pgsql/bin/postmaster -S -D
/usr/local/pgsql/data"
```

In FreeBSD 2.2-RELEASE edit /usr/local/etc/rc.d/pgsql.sh to contain the following lines and make it chmod 755 and chown root:bin.

```
#!/bin/sh
[ -x /usr/local/pgsql/bin/postmaster ] && {
    su -l pgsql -c ’exec /usr/local/pgsql/bin/postmaster
        -D/usr/local/pgsql/data
        -S -o -F > /usr/local/pgsql/errlog’ &
    echo -n ’ pgsql’
}
```

You may put the line breaks as shown above. The shell is smart enough to keep parsing beyond end-of-line if there is an expression unfinished. The exec saves one layer of shell under the postmaster process so the parent is init.

In RedHat Linux add a file `/etc/rc.d/init.d/postgres.init` which is based on the example in `contrib/linux/`. Then make a softlink to this file from `/etc/rc.d/rc5.d/S98postgres.init`.

Run the regression tests against the installed server (using the sequential test method). If you didn’t run the tests before installation, you should definitely do it now. For detailed instructions see *Regression Test*.

To start experimenting with Postgres, set up the paths as explained above and start the server. To create a database, type

```
> createdb testdb
```

Then enter

```
> psql testdb
```

to connect to that database. At the prompt you can enter SQL commands and start experimenting.

Chapter 6. Installation on Win32

Build and installation instructions for Postgres v6.4 client libraries on Win32.

Building the libraries

The makefiles included in Postgres are written for Microsoft Visual C++, and will probably not work with other systems. It should be possible to compile the libaries manually in other cases.

To build the libraries, change directory into the `src` directory, and type the commands

```
copy include\config.h.win32 include\config.h
nmake /f win32.mak
```

This assumes that you have Visual C++ in your path.

The following files will be built:

`interfaces\libpq\Release\libpq.dll` - The dynamically linkable frontend library
`interfaces\libpq\Release\libpqdll.lib` - Import library to link your program to libpq.dll
`interfaces\libpq\Release\libpq.lib` - Static library version of the frontend library
`bin\psql\Release\psql.exe` - The Postgresql interactive SQL monitor

Installing the libraries

The only part of the library to really be installed is the `libpq.dll` library. This file should in most cases be placed in the `WINNT\SYSTEM32` directory (or in `WINDOWS\SYSTEM` on a Windows 95/98 system). If this file is installed using a setup program, it should be installed with version checking using the VERSIONINFO resource included in the file, to ensure that a newer version of the library is not overwritten.

If you plan to do development using libpq on this machine, you will have to add the `src\include` and `src\interfaces\libpq` directories to the include path in your compilers settings.

Using the libraries

To use the libraries, you must add the `libpqdll.lib` file to your project (in Visual C++, just right-click on the project and chose to add it).

Once this is done, it should be possible to use the library just as you would on a Unix platform.

Chapter 7. Runtime Environment

This chapter outlines the interaction between Postgres and the operating system.

Using Postgres from Unix

All Postgres commands that are executed directly from a Unix shell are found in the directory `.../bin`. Including this directory in your search path will make executing the commands easier.

A collection of system catalogs exist at each site. These include a class (`pg_user`) that contains an instance for each valid Postgres user. The instance specifies a set of Postgres privileges, such as the ability to act as Postgres super-user, the ability to create/destroy databases, and the ability to update the system catalogs. A Unix user cannot do anything with Postgres until an appropriate instance is installed in this class. Further information on the system catalogs is available by running queries on the appropriate classes.

Starting postmaster

Nothing can happen to a database unless the postmaster process is running. As the site administrator, there are a number of things you should remember before starting the postmaster. These are discussed in the installation and configuration sections of this manual. However, if Postgres has been installed by following the installation instructions exactly as written, the following simple command is all you should need to start the postmaster:

```
% postmaster
```

The postmaster occasionally prints out messages which are often helpful during troubleshooting. If you wish to view debugging messages from the postmaster, you can start it with the -d option and redirect the output to the log file:

```
% postmaster -d > pm.log 2>&1 &
```

If you do not wish to see these messages, you can type

```
% postmaster -S
```

and the postmaster will be "S"ilent. No ampersand ("&") is required in this case, since the postmaster automatically detaches from the terminal when -S is specified.

Using pg_options

Note: Contributed by Massimo Dal Zotto (mailto:dz@cs.unitn.it)

The optional file `data/pg_options` contains runtime options used by the backend to control trace messages and other backend tunable parameters. The file is re-read by a backend when it receives a SIGHUP signal, making thus possible to change run-time options on the fly without needing to restart Postgres. The options specified in this file may be debugging flags used by the trace package (`backend/utils/misc/trace.c`) or numeric parameters which can be used by the backend to control its behaviour.

All pg_options are initialized to zero at backend startup. New or modified options will be read by all new backends when they are started. To make effective any changes for all running backends we need to send a SIGHUP to the postmaster. The signal will be automatically sent to all the backends. We can also activate the changes only for a specific backend by sending the SIGHUP directly to it.

pg_options can also be specified with the `-T` switch of Postgres:

```
postgres options -T "verbose=2,query,hostlookup-"
```

The functions used for printing errors and debug messages can now make use of the *syslog(2)* facility. Message printed to stdout or stderr are prefixed by a timestamp containing also the backend pid:

```
#timestamp          #pid    #message
980127.17:52:14.173 [29271] StartTransactionCommand
980127.17:52:14.174 [29271] ProcessUtility: drop table t;
980127.17:52:14.186 [29271] SIIncNumEntries: table is 70% full
980127.17:52:14.186 [29286] Async_NotifyHandler
980127.17:52:14.186 [29286] Waking up sleeping backend process
980127.19:52:14.292 [29286] Async_NotifyFrontEnd
980127.19:52:14.413 [29286] Async_NotifyFrontEnd done
980127.19:52:14.466 [29286] Async_NotifyHandler done
```

This format improves readability of the logs and allows people to understand exactly which backend is doing what and at which time. It also makes easier to write simple awk or perl scripts which monitor the log to detect database errors or problem, or to compute transaction time statistics.

Messages printed to syslog use the log facility LOG_LOCAL0. The use of syslog can be controlled with the syslog pg_option. Unfortunately many functions call directly `printf()` to print their messages to stdout or stderr and this output can’t be redirected to syslog or have timestamps in it. It would be advisable that all calls to printf would be replaced with the PRINTF macro and output to stderr be changed to use EPRINTF instead so that we can control all output in a uniform way.

The format of the `pg_options` file is as follows:

```
# comment
option=integer_value  # set value for option
option                # set option = 1
option+               # set option = 1
option-               # set option = 0
```

Note that *`keyword`* can also be an abbreviation of the option name defined in `backend/utils/misc/trace.c`.

Example 7-1. pg_options File

For example my pg_options file contains the following values:

```
verbose=2
query
hostlookup
showportnumber
```

Recognized Options

The options currently defined are:

all

Global trace flag. Allowed values are:

0

Trace messages enabled individually

1

Enable all trace messages

-1

Disable all trace messages

verbose

Verbosity flag. Allowed values are:

0

No messages. This is the default.

1

Print information messages.

2

Print more information messages.

query

  Query trace flag. Allowed values are:

  0

    Don’t print query.

  1

    Print a condensed query in one line.

  4

    Print the full query.

plan

  Print query plan.

parse

  Print parser output.

rewritten

  Print rewritten query.

pretty_plan

  Pretty-print query plan.

pretty_parse

  Pretty-print parser output.

pretty_rewritten

  Pretty-print rewritten query.

parserstats

  Print parser statistics.

plannerstats

  Print planner statistics.

executorstats

  Print executor statistics.

shortlocks

  Currently unused but needed to enable features in the future.

locks

  Trace locks.

userlocks

  Trace user locks.

spinlocks
: Trace spin locks.

notify
: Trace notify functions.

malloc
: Currently unused.

palloc
: Currently unused.

lock_debug_oidmin
: Minimum relation oid traced by locks.

lock_debug_relid
: oid, if not zero, of relation traced by locks.

lock_read_priority
: Currently unused.

deadlock_timeout
: Deadlock check timer.

syslog
: syslog flag. Allowed values are:

  0
  : Messages to stdout/stderr.

  1
  : Messages to stdout/stderr and syslog.

  2
  : Messages only to syslog.

hostlookup
: Enable hostname lookup in ps_status.

showportnumber
: Show port number in ps_status.

nofsync
: Disable fsync on a per-backend basis.

Chapter 8. Security

Database security is addressed at several levels:

- Data base file protection. All files stored within the database are protected from reading by any account other than the Postgres superuser account.
- Connections from a client to the database server are, by default, allowed only via a local Unix socket, not via TCP/IP sockets. The backend must be started with the `-i` option to allow non-local clients to connect.
- Client connections can be restricted by IP address and/or user name via the `pg_hba.conf` file in PG_DATA.
- Client connections may be authenticated vi other external packages.
- Each user in Postgres is assigned a username and (optionally) a password. By default, users do not have write access to databases they did not create.
- Users may be assigned to *groups*, and table access may be restricted based on group privileges.

User Authentication

Authentication is the process by which the backend server and postmaster ensure that the user requesting access to data is in fact who he/she claims to be. All users who invoke Postgres are checked against the contents of the `pg_user` class to ensure that they are authorized to do so. However, verification of the user's actual identity is performed in a variety of ways:

From the user shell

A backend server started from a user shell notes the user's (effective) user-id before performing a `setuid` to the user-id of user *`postgres`*. The effective user-id is used as the basis for access control checks. No other authentication is conducted.

From the network

If the Postgres system is built as distributed, access to the Internet TCP port of the postmaster process is available to anyone. The DBA configures the pg_hba.conf file in the PGDATA directory to specify what authentication system is to be used according to the host making the connection and which database it is connecting to. See *pg_hba.conf(5)* for a description of the authentication systems available. Of course, host-based authentication is not fool-proof in Unix, either. It is possible for determined intruders to also masquerade the origination host. Those security issues are beyond the scope of Postgres.

Host-Based Access Control

Host-based access control is the name for the basic controls PostgreSQL exercises on what clients are allowed to access a database and how the users on those clients must authenticate themselves.

Each database system contains a file named `pg_hba.conf`, in its PGDATA directory, which controls who can connect to each database.

Every client accessing a database *must* be covered by one of the entries in `pg_hba.conf`. Otherwise all attempted connections from that client will be rejected with a "User authentication failed" error message.

The general format of the `pg_hba.conf` file is of a set of records, one per line. Blank lines and lines beginning with a hash character ("#") are ignored. A record is made up of a number of fields which are separated by spaces and/or tabs.

Connections from clients can be made using Unix domain sockets or Internet domain sockets (ie. TCP/IP). Connections made using Unix domain sockets are controlled using records of the following format:

```
local database authentication method
```

where

`database` specifies the database that this record applies to. The value `all` specifies that it applies to all databases.
`authentication method` specifies the method a user must use to authenticate themselves when connecting to that database using Unix domain sockets. The different methods are described below.

Connections made using Internet domain sockets are controlled using records of the following format.

```
host database TCP/IP address TCP/IP mask authentication method
```

The *`TCP/IP address`* is logically anded to both the specified *`TCP/IP mask`* and the TCP/IP address of the connecting client. If the two resulting values are equal then the record is used for this connection. If a connection matches more than one record then the earliest one in the file is used. Both the *`TCP/IP address`* and the *`TCP/IP mask`* are specified in dotted decimal notation.

If a connection fails to match any record then the *reject* authentication method is applied (see below).

Authentication Methods

The following authentication methods are supported for both Unix and TCP/IP domain sockets:

trust

The connection is allowed unconditionally.

reject

The connection is rejected unconditionally.

crypt

The client is asked for a password for the user. This is sent encrypted (using *crypt(3)*) and compared against the password held in the `pg_shadow` table. If the passwords match, the connection is allowed.

password

The client is asked for a password for the user. This is sent in clear and compared against the password held in the `pg_shadow` table. If the passwords match, the connection is allowed. An optional password file may be specified after the `password` keyword which is used to match the supplied password rather than the pg_shadow table. See pg_passwd.

The following authentication methods are supported for TCP/IP domain sockets only:

krb4

Kerberos V4 is used to authenticate the user.

krb5

Kerberos V5 is used to authenticate the user.

ident

The ident server on the client is used to authenticate the user (RFC 1413). An optional map name may be specified after the `ident` keyword which allows ident user names to be mapped onto Postgres user names. Maps are held in the file `$PGDATA/pg_ident.conf`.

Examples

```
# Trust any connection via Unix domain sockets.
local   trust
# Trust any connection via TCP/IP from this machine.
host    all     127.0.0.1       255.255.255.255         trust
# We don’t like this machine.
host    all     192.168.0.10    255.255.255.0           reject
# This machine can’t encrypt so we ask for passwords in clear.
host    all     192.168.0.3     255.255.255.0           password
# The rest of this group of machines should provide encrypted
passwords.
host    all     192.168.0.0     255.255.255.0           crypt
```

User Names and Groups

To define a new user, run the createuser utility program.

To assign a user or set of users to a new group, one must define the group itself, and assign users to that group. In Postgres these steps are not currently supported with a **create group** command. Instead, the groups are defined by inserting appropriate values into the `pg_group` system table, and then using the **grant** command to assign privileges to the group.

Creating Users

Creating Groups

Currently, there is no easy interface to set up user groups. You have to explicitly insert/update the `pg_group table`. For example:

```
jolly=> insert into pg_group (groname, grosysid, grolist)
jolly=>     values (’posthackers’, ’1234’, ’{5443, 8261}’);
INSERT 548224
jolly=> grant insert on foo to group posthackers;
CHANGE
jolly=>
```

The fields in `pg_group` are:

groname

The group name. This a name and should be purely alphanumeric. Do not include underscores or other punctuation.

grosysid

The group id. This is an int4. This should be unique for each group.

grolist

The list of pg_user id’s that belong in the group. This is an int4[].

Assigning Users to Groups

Access Control

Postgres provides mechanisms to allow users to limit the access to their data that is provided to other users.

Database superusers

Database super-users (i.e., users who have `pg_user.usesuper` set) silently bypass all of the access controls described below with two exceptions: manual system catalog updates are not permitted if the user does not have `pg_user.usecatupd` set, and destruction of system catalogs (or modification of their schemas) is never allowed.

Access Privilege

The use of access privilege to limit reading, writing and setting of rules on classes is covered in *grant/revoke(l)*.

Class removal and schema modification

Commands that destroy or modify the structure of an existing class, such as **alter**, **drop table**, and **drop index**, only operate for the owner of the class. As mentioned above, these operations are *never* permitted on system catalogs.

Functions and Rules

Functions and rules allow users to insert code into the backend server that other users may execute without knowing it. Hence, both mechanisms permit users to *trojan horse* others with relative impunity. The only real protection is tight control over who can define functions (e.g., write to relations with SQL fields) and rules. Audit trails and alerters on `pg_class`, `pg_user` and `pg_group` are also recommended.

Functions

Functions written in any language except SQL run inside the backend server process with the permissions of the user *`postgres`* (the backend server runs with its real and effective user-id set to *`postgres`*. It is possible for users to change the server's internal data structures from inside of trusted functions. Hence, among many other things, such functions can circumvent any system access controls. This is an inherent problem with user-defined C functions.

Rules

Like SQL functions, rules always run with the identity and permissions of the user who invoked the backend server.

Caveats

There are no plans to explicitly support encrypted data inside of Postgres (though there is nothing to prevent users from encrypting data within user-defined functions). There are no plans to explicitly support encrypted network connections, either, pending a total rewrite of the frontend/backend protocol.

User names, group names and associated system identifiers (e.g., the contents of `pg_user.usesysid`) are assumed to be unique throughout a database. Unpredictable results may occur if they are not.

Secure TCP/IP Connection

Author: From e-mail by Gene Selkov, Jr. (selkovjr@mcs.anl.gov) written on 1999-09-08 in response to a question from Eric Marsden.

One can use ssh to encrypt the network connection between clients and a Postgres server. Done properly, this should lead to an adequately secure network connection.

The documentation for ssh provides most of the information to get started. Please refer to http://www.heimhardt.de/htdocs/ssh.html for better insight.

A step-by-step explanation can be done in just two steps.

Running a secure tunnel via ssh

A step-by-step explanation can be done in just two steps.

1. Establish a tunnel to the backend machine, like this:

   ```
   ssh -L 3333:wit.mcs.anl.gov:5432 postgres@wit.mcs.anl.gov
   ```

   The first number in the -L argument, 3333, is the port number of your end of the tunnel. The second number, 5432, is the remote end of the tunnel -- the port number your backend is using. The name or the address in between the port numbers belongs to the server machine, as does the last argument to ssh that also includes the optional user name. Without the user name, ssh will try the name you are currently logged on as on the client machine. You can use any user name the server machine will accept, not necessarily those related to postgres.

2. Now that you have a running ssh session, you can connect a postgres client to your local host at the port number you specified in the previous step. If it’s psql, you will need another shell because the shell session you used in step 1 is now occupied with ssh.

   ```
   psql -h localhost -p 3333 -d mpw
   ```

   Note that you have to specify the `-h` argument to cause your client to use the TCP socket instead of the Unix socket. You can omit the port argument if you chose 5432 as your end of the tunnel.

Chapter 9. Adding and Deleting Users

createuser enables specific users to access Postgres. dropuser removes users and prevents them from accessing Postgres.

These commands only affect users with respect to Postgres; they have no effect on a user's other privileges or status with regards to the underlying operating system.

Chapter 10. Disk Management

Alternate Locations

It is possible to create a database in a location other than the default location for the installation. Remember that all database access actually occurs through the database backend, so that any location specified must be accessible by the backend.

Alternate database locations are created and referenced by an environment variable which gives the absolute path to the intended storage location. This environment variable must have been defined before the backend was started and must be writable by the postgres administrator account. Any valid environment variable name may be used to reference an alternate location, although using variable name with a prefix of PGDATA is recommended to avoid confusion and conflict with other variables.

> **Note:** In previous versions of Postgres, it was also permissable to use an absolute path name to specify an alternate storage location. The environment variable style of specification is to be preferred since it allows the site administrator more flexibility in managing disk storage. If you prefer using absolute paths, you may do so by defining "ALLOW_ABSOLUTE_DBPATHS" and recompiling Postgres To do this, either add this line
>
> ```
> #define ALLOW_ABSOLUTE_DBPATHS 1
> ```
>
> to the file `src/include/config.h`, or by specifying
>
> ```
>  CFLAGS+= -DALLOW_ABSOLUTE_DBPATHS
> ```
>
> in your `Makefile.custom`.

Remember that database creation is actually performed by the database backend. Therefore, any environment variable specifying an alternate location must have been defined before the backend was started. To define an alternate location PGDATA2 pointing to `/home/postgres/data`, first type

```
% setenv PGDATA2 /home/postgres/data
```

to define the environment variable to be used with subsequent commands. Usually, you will want to define this variable in the Postgres superuser’s `.profile` or `.cshrc` initialization file to ensure that it is defined upon system startup. Any environment variable can be used to reference alternate location, although it is preferred that the variables be prefixed with "PGDATA" to eliminate confusion and the possibility of conflicting with or overwriting other variables.

To create a data storage area in PGDATA2, ensure that `/home/postgres` already exists and is writable by the postgres administrator. Then from the command line, type

```
% setenv PGDATA2 /home/postgres/data
% initlocation $PGDATA2
Creating Postgres database system directory /home/postgres/data

Creating Postgres database system directory /home/postgres/data/base
```

To test the new location, create a database test by typing

```
% createdb -D PGDATA2 test
% dropdb test
```

Chapter 11. Managing a Database

If the Postgres postmaster is up and running we can create some databases to experiment with. Here, we describe the basic commands for managing a database.

Creating a Database

Let's say you want to create a database named mydb. You can do this with the following command:

```
% createdb dbname
```

Postgres allows you to create any number of databases at a given site and you automatically become the database administrator of the database you just created. Database names must have an alphabetic first character and are limited to 31 characters in length. Not every user has authorization to become a database administrator. If Postgres refuses to create databases for you, then the site administrator needs to grant you permission to create databases. Consult your site administrator if this occurs.

Accessing a Database

Once you have constructed a database, you can access it by:

- running the Postgres terminal monitor program (psql) which allows you to interactively enter, edit, and execute SQL commands.
- writing a C program using the `libpq` subroutine library. This allows you to submit SQL commands from C and get answers and status messages back to your program. This interface is discussed further in the *PostgreSQL Programmer's Guide*.

You might want to start up psql, to try out the examples in this manual. It can be activated for the *`dbname`* database by typing the command:

```
psql dbname
```

You will be greeted with the following message:

```
Welcome to psql, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

dbname=>
```

This prompt indicates that the terminal monitor is listening to you and that you can type SQL queries into a workspace maintained by the terminal monitor. The psql program responds to escape codes that begin with the backslash character, "\". For example, you can get help on the syntax of various Postgres SQL commands by typing:

```
dbname=> \h
```

Once you have finished entering your queries into the workspace, you can pass the contents of the workspace to the Postgres server by typing:

```
dbname=> \g
```

This tells the server to process the query. If you terminate your query with a semicolon, the backslash-g is not necessary. psql will automatically process semicolon terminated queries. To read queries from a file, instead of entering them interactively, type:

```
dbname=> \i filename
```

To get out of psql and return to Unix, type

```
dbname=> \q
```

and psql will quit and return you to your command shell. (For more escape codes, type backslash-h at the monitor prompt.) White space (i.e., spaces, tabs and newlines) may be used freely in SQL queries. Single-line comments are denoted by two dashes ("`--`"). Everything after the dashes up to the end of the line is ignored. Multiple-line comments, and comments within a line, are denoted by "`/* ... */`", a convention borrowed from Ingres.

Destroying a Database

If you are the database administrator for the database mydb, you can destroy it using the following Unix command:

```
% dropdb dbname
```

This action physically removes all of the Unix files associated with the database and cannot be undone, so this should only be done with a great deal of forethought.

It is also possible to destroy a database from within an SQL session by using

```
> drop database dbname
```

Backup and Restore

> **Caution**
>
> Every database should be backed up on a regular basis. Since Postgres manages it's own files in the file system, it is *not advisable* to rely on system backups of your file system for your database backups; there is no guarantee that the files will be in a usable, consistant state after restoration.

Postgres provides two utilities to backup your system: pg_dump to backup individual databases and pg_dumpall to backup your installation in one step.

An individual database can be backed up using the following command:

```
% pg_dump dbname > dbname.pgdump
```

and can be restored using

```
cat dbname.pgdump | psql dbname
```

This technique can be used to move databases to new locations, and to rename existing databases.

Large Databases

Author: Written by Hannu Krosing (hannu@trust.ee) on 1999-06-19.

Since Postgres allows tables larger than the maximum file size on your system, it can be problematic to dump the table to a file, since the resulting file will likely be larger than the maximum size allowed by your system.

As pg_dump writes to stdout, you can just use standard *nix tools to work around this possible problem:

Use compressed dumps:

```
% pg_dump dbname | gzip > filename.dump.gz
```

reload with

```
% createdb dbname
% gunzip -c filename.dump.gz | psql dbname
```

or

```
% cat filename.dump.gz | gunzip | psql dbname
```

Use split:

```
% pg_dump dbname | split -b 1m - filename.dump.
```

reload with

```
% createdb dbname
% cat filename.dump.* | pgsql dbname
```

Of course, the name of the file (*filename*) and the content of the pg_dump output need not match the name of the database. Also, the restored database can have an arbitrary new name, so this mechanism is also suitable for renaming databases.

Chapter 12. Troubleshooting

Postmaster Startup Failures

There are several common reasons for the postmaster to fail to start up. Check the postmaster's log file, or start it by hand (without redirecting standard output or standard error) to see what complaint messages appear. Some of the possible error messages are reasonably self-explanatory, but here are some that are not:

```
FATAL: StreamServerPort: bind() failed: Address already in use
        Is another postmaster already running on that port?
```

This usually means just what it suggests: you accidentally started a second postmaster on the same port where one is already running. However, if the kernel error message is not "Address already in use" or some variant of that wording, there may be a different problem. For example, trying to start a postmaster on a reserved port number may draw something like

```
$ postmaster -i -p 666
FATAL: StreamServerPort: bind() failed: Permission denied
        Is another postmaster already running on that port?
```

```
IpcMemoryCreate: shmget failed (Invalid argument) key=5440001,
size=83918612, permission=600
FATAL 1:  ShmemCreate: cannot create region
```

A message like this probably means that your kernel's limit on the size of shared memory areas is smaller than the buffer area that Postgres is trying to create. (Or it could mean that you don't have SysV-style shared memory support configured into your kernel at all.) As a temporary workaround, you can try starting the postmaster with a smaller-than-normal number of buffers (-B switch). You will eventually want to reconfigure your kernel to increase the allowed shared memory size, however. You may see this message when trying to start multiple postmasters on the same machine, if their total space requests exceed the kernel limit.

```
IpcSemaphoreCreate: semget failed (No space left on device)
key=5440026, num=16, permission=600
```

A message like this does *not* mean that you've run out of disk space; it means that your kernel's limit on the number of SysV semaphores is smaller than the number Postgres wants to create. As above, you may be able to work around the problem by starting the postmaster with a reduced number of backend processes (-N switch), but you’ll eventually want to increase the kernel limit.

Client Connection Problems

Once you have a running postmaster, trying to connect to it with client applications can fail for a variety of reasons. The sample error messages shown here are for clients based on recent versions of libpq --- clients based on other interface libraries may produce other messages with more or less information.

```
connectDB() -- connect() failed: Connection refused
Is the postmaster running (with -i) at ’server.joe.com’ and
accepting connections on TCP/IP port ’5432’?
```

This is the generic "I couldn’t find a postmaster to talk to" failure. It looks like the above when TCP/IP communication is attempted, or like this when attempting Unix-socket communication to a local postmaster:

```
connectDB() -- connect() failed: No such file or directory
Is the postmaster running at ’localhost’ and accepting connections
on Unix socket ’5432’?
```

The last line is useful in verifying that the client is trying to connect where it is supposed to. If there is in fact no postmaster running there, the kernel error message will typically be either "Connection refused" or "No such file or directory", as illustrated. (It is particularly important to realize that "Connection refused" in this context does *not* mean that the postmaster got your connection request and rejected it --- that case will produce a different message, as shown below.) Other error messages such as "Connection timed out" may indicate more fundamental problems, like lack of network connectivity.

```
No pg_hba.conf entry for host 123.123.123.123, user joeblow,
database testdb
```

This is what you are most likely to get if you succeed in contacting a postmaster, but it doesn’t want to talk to you. As the message suggests, the postmaster refused the connection request because it found no authorizing entry in its pg_hba.conf configuration file.

```
Password authentication failed for user ’joeblow’
```

Messages like this indicate that you contacted the postmaster, and it’s willing to talk to you, but not until you pass the authorization method specified in the pg_hba.conf file. Check the password you’re providing, or check your Kerberos or IDENT software if the complaint mentions one of those authentication types.

```
FATAL 1:  SetUserId: user ’joeblow’ is not in ’pg_shadow’
```

This is another variant of authentication failure: no Postgres create_user command has been executed for the given username.

```
FATAL 1:  Database testdb does not exist in pg_database
```

There’s no database by that name under the control of this postmaster. Note that if you don’t specify a database name, it defaults to your Postgres username, which may or may not be the right thing.

```
NOTICE:  Unrecognized variable client_encoding
```

This isn’t an error; in fact, it’s quite harmless. You’ll see this message at startup if you use a client compiled with MULTIBYTE support to connect to a server compiled without it. (The client is trying to tell the server what character set encoding it wants, but the server has no idea what it’s talking about.) If the message bothers you, use a client compiled with the same options as the server.

Debugging Messages

The postmaster occasionally prints out messages which are often helpful during troubleshooting. If you wish to view debugging messages from the postmaster, you can start it with the -d option and redirect the output to the log file:

```
% postmaster -d > pm.log 2>&1 &
```

If you do not wish to see these messages, you can type

```
% postmaster -S
```

and the postmaster will be "S"ilent. No ampersand ("&") is required in this case, since the postmaster automatically detaches from the terminal when -S is specified.

pg_options

Note: Contributed by Massimo Dal Zotto (mailto:dz@cs.unitn.it)

The optional file `data/pg_options` contains runtime options used by the backend to control trace messages and other backend tunable parameters. What makes this file interesting is the fact that it is re-read by a backend when it receives a SIGHUP signal, making thus possible to change run-time options on the fly without needing to restart Postgres. The options specified in this file may be debugging flags used by the trace package (`backend/utils/misc/trace.c`) or numeric parameters which can be used by the backend to control its behaviour. New options and parameters must be defined in `backend/utils/misc/trace.c` and `backend/include/utils/trace.h`.

pg_options can also be specified with the `-T` switch of Postgres:

```
postgres options -T "verbose=2,query,hostlookup-"
```

The functions used for printing errors and debug messages can now make use of the *syslog(2)* facility. Message printed to stdout or stderr are prefixed by a timestamp containing also the backend pid:

```
#timestamp          #pid    #message
980127.17:52:14.173 [29271] StartTransactionCommand
980127.17:52:14.174 [29271] ProcessUtility: drop table t;
980127.17:52:14.186 [29271] SIIncNumEntries: table is 70% full
980127.17:52:14.186 [29286] Async_NotifyHandler
980127.17:52:14.186 [29286] Waking up sleeping backend process
980127.19:52:14.292 [29286] Async_NotifyFrontEnd
980127.19:52:14.413 [29286] Async_NotifyFrontEnd done
980127.19:52:14.466 [29286] Async_NotifyHandler done
```

This format improves readability of the logs and allows people to understand exactly which backend is doing what and at which time. It also makes easier to write simple awk or perl scripts which monitor the log to detect database errors or problem, or to compute transaction time statistics.

Messages printed to syslog use the log facility LOG_LOCAL0. The use of syslog can be controlled with the syslog pg_option. Unfortunately many functions call directly `printf()` to print their messages to stdout or stderr and this output can't be redirected to syslog or have timestamps in it. It would be advisable that all calls to printf would be replaced with the PRINTF macro and output to stderr be changed to use EPRINTF instead so that we can control all output in a uniform way.

The format of the `pg_options` file is as follows:

```
# comment
option=integer_value  # set value for option
option                # set option = 1
option+               # set option = 1
option-               # set option = 0
```

Note that *`keyword`* can also be an abbreviation of the option name defined in `backend/utils/misc/trace.c`.

Refer to *Using pg_options* for a complete list of option keywords and possible values.

Chapter 13. Database Recovery

This section needs to be written. Volunteers?

Chapter 14. Regression Test

Regression test instructions and analysis.

The PostgreSQL regression tests are a comprehensive set of tests for the SQL implementation embedded in PostgreSQL. They test standard SQL operations as well as the extended capabilities of PostgreSQL.

There are two different ways in which the regression tests can be run: the "sequential" method and the "parallel" method. The sequential method runs each test script in turn, whereas the parallel method starts up multiple server processes to run groups of tests in parallel. Parallel testing gives confidence that interprocess communication and locking are working correctly. Another key difference is that the sequential test procedure uses an already-installed postmaster, whereas the parallel test procedure tests a system that has been built but not yet installed. (The parallel test script actually does an installation into a temporary directory and fires up a private postmaster therein.)

Some properly installed and fully functional PostgreSQL installations can "fail" some of these regression tests due to artifacts of floating point representation and time zone support. The tests are currently evaluated using a simple diff comparison against the outputs generated on a reference system, so the results are sensitive to small system differences. When a test is reported as "failed", always examine the differences between expected and actual results; you may well find that the differences are not significant.

The regression tests were originally developed by Jolly Chen and Andrew Yu, and were extensively revised/repackaged by Marc Fournier and Thomas Lockhart. From PostgreSQL v6.1 onward the regression tests are current for every official release.

Regression Environment

The regression testing notes below assume the following (except where noted):

Commands are Unix-compatible. See note below.
Defaults are used except where noted.
User postgres is the Postgres superuser.
The source path is /usr/src/pgsql (other paths are possible).
The runtime path is /usr/local/pgsql (other paths are possible).

Normally, the regression tests should be run as the postgres user since the ’src/test/regress’ directory and sub-directories are owned by the postgres user. If you run the regression test as another user the ’src/test/regress’ directory tree must be writeable by that user.

It was formerly necessary to run the postmaster with system time zone set to PST, but this is no longer required. You can run the regression tests under your normal postmaster configuration. The test script will set the PGTZ environment variable to ensure that timezone-dependent tests produce the expected results. However, your system must provide library support for the PST8PDT time zone, or the timezone-dependent tests will fail. To verify that your machine does have this support, type the following:

```
setenv TZ PST8PDT
date
```

The "date" command above should have returned the current system time in the PST8PDT time zone. If the PST8PDT database is not available, then your system may have returned the time in GMT. If the PST8PDT time zone is not available, you can set the time zone rules explicitly:

```
setenv PGTZ PST8PDT7,M04.01.0,M10.05.03
```

The directory layout is as follows:

input
: Source files that are converted using ’make all’ into some of the .sql files in the ’sql’ subdirectory

output
: .source files that are converted using ’make all’ into .out files in the ’expected’ subdirectory

sql
: .sql files used to perform the regression tests

expected
: .out files that represent what we *expect* the results to look like

results
: .out files that contain what the results *actually* look like. Also used as temporary storage for table copy testing.

tmp_check
: Temporary installation created by parallel testing script.

Regression Test Procedure

Commands were tested on RedHat Linux version 4.2 using the bash shell. Except where noted, they will probably work on most systems. Commands like `ps` and `tar` vary wildly on what options you should use on each platform. *Use common sense* before typing in these commands.

Postgres Regression Test

1. Prepare the files needed for the regression test with:

   ```
   cd /usr/src/pgsql/src/test/regress
           gmake clean
           gmake all
   ```

   You can skip "gmake clean" if this is the first time you are running the tests.

   This step compiles a C program with PostgreSQL extension functions into a shared library. Localized SQL scripts and output-comparison files are also created for the tests that need them. The localization replaces macros in the source files with absolute pathnames and user names.

2. If you intend to use the "sequential" test procedure, which tests an already-installed postmaster, be sure that the postmaster is running. If it isn’t already running, start the postmaster in an available window by typing

   ```
   postmaster
   ```

or start the postmaster daemon running in the background by typing

```
cd
      nohup postmaster > regress.log 2>&1 &
```

The latter is probably preferable, since the regression test log will be quite lengthy (60K or so, in Postgres 7.0) and you might want to review it for clues if things go wrong.

Note: Do not run `postmaster` from the root account.

3. Run the regression tests. For a sequential test, type

   ```
   cd /usr/src/pgsql/src/test/regress
    gmake runtest
   ```

   For a parallel test, type

   ```
   cd /usr/src/pgsql/src/test/regress
        gmake runcheck
   ```

   The sequential test just runs the test scripts using your already-running postmaster. The parallel test will perform a complete installation of Postgres into a temporary directory, start a private postmaster therein, and then run the test scripts. Finally it will kill the private postmaster (but the temporary directory isn't removed automatically).

4. You should get on the screen (and also written to file ./regress.out) a series of statements stating which tests passed and which tests failed. Please note that it can be normal for some of the tests to "fail" due to platform-specific variations. See the next section for details on determining whether a "failure" is significant.

   Some of the tests, notably "numeric", can take a while, especially on slower platforms. Have patience.

5. After running the tests and examining the results, type

   ```
   cd /usr/src/pgsql/src/test/regress
      gmake clean
   ```

   to recover the temporary disk space used by the tests. If you ran a sequential test, also type

   ```
   dropdb regression
   ```

Regression Analysis

The actual outputs of the regression tests are in files in the `./results` directory. The test script uses diff to compare each output file against the reference outputs stored in the `./expected` directory. Any differences are saved for your inspection in `./regression.diffs`. (Or you can run diff yourself, if you prefer.)

The files might not compare exactly. The test script will report any difference as a "failure", but the difference might be due to small cross-system differences in error message wording, math library behavior, etc. "Failures" of this type do not indicate a problem with Postgres.

Thus, it is necessary to examine the actual differences for each "failed" test to determine whether there is really a problem. The following paragraphs attempt to provide some guidance in determining whether a difference is significant or not.

Error message differences

Some of the regression tests involve intentional invalid input values. Error messages can come from either the Postgres code or from the host platform system routines. In the latter case, the messages may vary between platforms, but should reflect similar information. These differences in messages will result in a "failed" regression test which can be validated by inspection.

Date and time differences

Most of the date and time results are dependent on timezone environment. The reference files are generated for timezone PST8PDT (Berkeley, California) and there will be apparent failures if the tests are not run with that timezone setting. The regression test driver sets environment variable PGTZ to PST8PDT to ensure proper results.

Some of the queries in the "timestamp" test will fail if you run the test on the day of a daylight-savings time changeover, or the day before or after one. These queries assume that the intervals between midnight yesterday, midnight today and midnight tomorrow are exactly twenty-four hours ... which is wrong if daylight-savings time went into or out of effect meanwhile.

There appear to be some systems which do not accept the recommended syntax for explicitly setting the local time zone rules; you may need to use a different PGTZ setting on such machines.

Some systems using older timezone libraries fail to apply daylight-savings corrections to pre-1970 dates, causing pre-1970 PDT times to be displayed in PST instead. This will result in localized differences in the test results.

Floating point differences

Some of the tests involve computing 64-bit (float8) numbers from table columns. Differences in results involving mathematical functions of float8 columns have been observed. The float8 and geometry tests are particularly prone to small differences across platforms. Human eyeball comparison is needed to determine the real significance of these differences which are usually 10 places to the right of the decimal point.

Some systems signal errors from pow() and exp() differently from the mechanism expected by the current Postgres code.

Polygon differences

Several of the tests involve operations on geographic date about the Oakland/Berkley CA street map. The map data is expressed as polygons whose vertices are represented as pairs of float8 numbers (decimal latitude and longitude). Initially, some tables are created and loaded with geographic data, then some views are created which join two tables using the polygon intersection operator (##), then a select is done on the view. When comparing the results from different platforms, differences occur in the 2nd or 3rd place to the right of the decimal point. The SQL statements where these problems occur are the following:

```
QUERY: SELECT * from street;
      QUERY: SELECT * from iexit;
```

Random differences

There is at least one case in the "random" test script that is intended to produce random results. This causes random to fail the regression test once in a while (perhaps once in every five to ten trials). Typing

```
diff results/random.out expected/random.out
```

should produce only one or a few lines of differences. You need not worry unless the random test always fails in repeated attempts. (On the other hand, if the random test is *never* reported to fail even in many trials of the regress tests, you probably *should* worry.)

The "expected" files

The `./expected/*.out` files were adapted from the original monolithic `expected.input` file provided by Jolly Chen et al. Newer versions of these files generated on various development machines have been substituted after careful (?) inspection. Many of the development machines are running a Unix OS variant (FreeBSD, Linux, etc) on Ix86 hardware. The original `expected.input` file was created on a SPARC Solaris 2.4 system using the `postgres5-1.02a5.tar.gz` source tree. It was compared with a file created on an I386 Solaris 2.4 system and the differences were only in the floating point polygons in the 3rd digit to the right of the decimal point. The original `sample.regress.out` file was from the postgres-1.01 release constructed by Jolly Chen. It may have been created on a DEC ALPHA machine as the `Makefile.global` in the postgres-1.01 release has PORTNAME=alpha.

Platform-specific comparison files

Since some of the tests inherently produce platform-specific results, we have provided a way to supply platform-specific result comparison files. Frequently, the same variation applies to multiple platforms; rather than supplying a separate comparison file for every platform, there is a mapping file that defines which comparison file to use. So, to eliminate bogus test "failures" for a particular platform, you must choose or make a variant result file, and then add a line to the mapping file, which is "resultmap".

Each line in the mapping file is of the form

```
testname/platformnamepattern=comparisonfilename
```

The test name is just the name of the particular regression test module. The platform name pattern is a pattern in the style of expr(1) (that is, a regular expression with an implicit ^ anchor at the start). It is matched against the platform name as printed by config.guess. The comparison file name is the name of the substitute result comparison file.

For example: the int2 regress test includes a deliberate entry of a value that is too large to fit in int2. The specific error message that is produced is platform-dependent; our reference platform emits

```
    ERROR:  pg_atoi: error reading "100000": Numerical result out of
range
```

but a fair number of other Unix platforms emit

```
    ERROR:  pg_atoi: error reading "100000": Result too large
```

Therefore, we provide a variant comparison file, int2-too-large.out, that includes this spelling of the error message. To silence the bogus "failure" message on HPPA platforms, resultmap includes

```
           int2/hppa=int2-too-large
```

which will trigger on any machine for which config.guess's output begins with 'hppa'. Other lines in resultmap select the variant comparison file for other platforms where it's appropriate.

Chapter 15. Release Notes

Release 7.0

This release contains improvements in many areas, demonstrating the continued growth of PostgreSQL. There are more improvements and fixes in 7.0 than in any previous release. The developers have confidence that this is the best release yet; we do our best to put out only solid releases, and this one is no exception.

Major changes in this release:

Foreign Keys

> Foreign keys are now implemented, with the exception of PARTIAL MATCH foreign keys. Many users have been asking for this feature, and we are pleased to offer it.

Optimizer Overhaul

> Continuing on work started a year ago, the optimizer has been improved, allowing better query plan selection and faster performance with less memory usage.

Updated psql

> psql, our interactive terminal monitor, has been updated with a variety of new features. See the psql manual page for details.

Join Syntax

> SQL92 join syntax is now supported, though only as INNER JOINs for this release. JOIN, NATURAL JOIN, JOIN/USING, JOIN/ON are available, as are column correlation names.

Migration to v7.0

A dump/restore using pg_dump is required for those wishing to migrate data from any previous release of Postgres. For those upgrading from 6.5.*, you may instead use pg_upgrade to upgrade to this release; however, a full dump/reload installation is always the most robust method for upgrades.

Interface and compatibility issues to consider for the new release include:

- The date/time types datetime and timespan have been superceded by the SQL92-defined types timestamp and interval. Although there has been some effort to ease the transition by allowing Postgres to recognize the deprecated type names and translate them to the new type names, this mechanism may not be completely transparent to your existing application.

- The optimizer has been substantially improved in the area of query cost estimation. In some cases, this will result in decreased query times as the optimizer makes a better choice for the preferred plan. However, in a small number of cases, usually involving pathological distributions of data, your query times may go up. If you are dealing with large amounts of data, you may want to check your queries to verify performance.

The JDBC and ODBC interfaces have been upgraded and extended.

The string function `CHAR_LENGTH` is now a native function. Previous versions translated this into a call to `LENGTH`, which could result in ambiguity with other types implementing `LENGTH` such as the geometric types.

Detailed Change List

```
Bug Fixes
---------
Prevent function calls exceeding maximum number of arguments (Tom)
Improve CASE construct (Tom)
Fix SELECT coalesce(f1,0) FROM int4_tbl GROUP BY f1 (Tom)
Fix SELECT sentence.words[0] FROM sentence GROUP BY
sentence.words[0] (Tom)
Fix GROUP BY scan bug (Tom)
Improvements in SQL grammar processing (Tom)
Fix for views involved in INSERT ... SELECT ... (Tom)
Fix for SELECT a/2, a/2 FROM test_missing_target GROUP BY a/2 (Tom)
Fix for subselects in INSERT ... SELECT (Tom)
Prevent INSERT ... SELECT ... ORDER BY (Tom)
Fixes for relations greater than 2GB, including vacuum
Improve propagating system table changes to other backends (Tom)
Improve propagating user table changes to other backends (Tom)
Fix handling of temp tables in complex situations (Bruce, Tom)
Allow table locking at table open, improving concurrent reliability
(Tom)
Properly quote sequence names in pg_dump (Ross J. Reedstrom)
Prevent DROP DATABASE while others accessing
Prevent any rows from being returned by GROUP BY if no rows
processed (Tom)
Fix SELECT COUNT(1) FROM table WHERE ...’ if no rows matching WHERE
(Tom)
Fix pg_upgrade so it works for MVCC (Tom)
Fix for SELECT ... WHERE x IN (SELECT ... HAVING SUM(x) > 1) (Tom)
Fix for "f1 datetime DEFAULT ’now’"  (Tom)
Fix problems with CURRENT_DATE used in DEFAULT (Tom)
Allow comment-only lines, and ;;; lines too. (Tom)
Improve recovery after failed disk writes, disk full (Hiroshi)
Fix cases where table is mentioned in FROM but not joined (Tom)
Allow HAVING clause without aggregate functions (Tom)
Fix for "--" comment and no trailing newline, as seen in perl
interface
Improve pg_dump failure error reports (Bruce)
Allow sorts and hashes to exceed 2GB file sizes (Tom)
Fix for pg_dump dumping of inherited rules (Tom)
Fix for NULL handling comparisons (Tom)
Fix inconsistent state caused by failed CREATE/DROP commands
(Hiroshi)
Fix for dbname with dash
Prevent DROP INDEX from interfering with other backends (Tom)
Fix file descriptor leak in verify_password()
Fix for "Unable to identify an operator =$" problem
Fix ODBC so no segfault if CommLog and Debug enabled (Dirk
Niggemann)
Fix for recursive exit call (Massimo)
Fix for extra-long timezones (Jeroen van Vianen)
Make pg_dump preserve primary key information (Peter E)
Prevent databases with single quotes (Peter E)
Prevent DROP DATABASE inside  transaction (Peter E)
ecpg memory leak fixes (Stephen Birch)
Fix for SELECT null::text, SELECT int4fac(null) and SELECT 2 +
(null) (Tom)
Y2K timestamp fix (Massimo)
Fix for VACUUM 'HEAP_MOVED_IN was not expected' errors (Tom)
Fix for views with tables/columns containing spaces  (Tom)
Prevent permissions on indexes (Peter E)
Fix for spinlock stuck problem when error is generated (Hiroshi)
Fix ipcclean on Linux
Fix handling of NULL constraint conditions (Tom)
Fix memory leak in odbc driver (Nick Gorham)
Fix for permission check on UNION tables (Tom)
Fix to allow SELECT 'a' LIKE 'a' (Tom)
Fix for SELECT 1 + NULL (Tom)
Fixes to CHAR
Fix log() on numeric type (Tom)
Deprecate ':' and ';' operators
Allow vacuum of temporary tables
Disallow inherited columns with the same name as new columns
Recover or force failure when disk space is exhausted(Hiroshi)
Fix INSERT INTO ... SELECT with AS columns matching result columns
Fix INSERT ... SELECT ... GROUP BY groups by target columns not
source columns(Tom)
Fix CREATE TABLE test (a char(5) DEFAULT text '', b int4) with
INSERT(Tom)
Fix UNION with LIMIT
Fix CREATE TABLE x AS SELECT 1 UNION SELECT 2
Fix CREATE TABLE test(col char(2) DEFAULT user)
Fix mismatched types in CREATE TABLE ... DEFAULT
Fix SELECT * FROM pg_class where oid in (0,-1)
Fix SELECT COUNT('asdf') FROM pg_class WHERE oid=12
Prevent user who can create databases can modifying pg_database
table(Peter E)
Fix btree to give a useful elog when key > 1/2 (page -
overhead)(Tom)
Fix INSERT of 0.0 into DECIMAL(4,4) field(Tom)

Enhancements
------------
New CLI interface include file sqlcli.h, based on SQL3/SQL98
Remove all limits on query length, row length limit still exists
(Tom)
Update jdbc protocol to 2.0 (Jens Glaser (mailto:jens@jens.de))
Add TRUNCATE command to quickly truncate relation (Mike Mascari)
Fix to give super user and createdb user proper update catalog
rights (Peter E)
Allow ecpg bool variables to have NULL values (Christof)
Issue ecpg error if NULL value for variable with no NULL indicator
(Christof)
Allow ^C to cancel COPY command (Massimo)
```

```
Add SET FSYNC and SHOW PG_OPTIONS commands(Massimo)
Function name overloading for dynamically-loaded C functions
(Frankpitt)
Add CmdTuples() to libpq++(Vince)
New CREATE CONSTRAINT TRIGGER and SET CONSTRAINTS commands(Jan)
Allow CREATE FUNCTION/WITH clause to be used for all language types
configure --enable-debug adds -g (Peter E)
configure --disable-debug removes -g (Peter E)
Allow more complex default expressions (Tom)
First real FOREIGN KEY constraint trigger functionality (Jan)
Add FOREIGN KEY ... MATCH FULL ... ON DELETE CASCADE (Jan)
Add FOREIGN KEY ... MATCH <unspecified> referential actions (Don
Baccus)
Allow WHERE restriction on ctid (physical heap location) (Hiroshi)
Move pginterface from contrib to interface directory, rename to
pgeasy (Bruce)
Change pgeasy connectdb() parameter ordering (Bruce)
Require SELECT DISTINCT target list to have all ORDER BY columns
(Tom)
Add Oracle’s COMMENT ON command (Mike Mascari
(mailto:mascarim@yahoo))
libpq’s PQsetNoticeProcessor function now returns previous
hook(Peter E)
Prevent PQsetNoticeProcessor from being set to NULL (Peter E)
Make USING in COPY optional (Bruce)
Allow subselects in the target list (Tom)
Allow subselects on the left side of comparison operators (Tom)
New parallel regression test (Jan)
Change backend-side COPY to write files with permissions 644 not 666
(Tom)
Force permissions on PGDATA directory to be secure, even if it
exists (Tom)
Added psql LASTOID variable to return last inserted oid (Peter E)
Allow concurrent vacuum and remove pg_vlock vacuum lock file (Tom)
Add permissions check for vacuum (Peter E)
New libpq functions to allow asynchronous connections:
PQconnectStart(),
   PQconnectPoll(), PQresetStart(), PQresetPoll(), PQsetenvStart(),
   PQsetenvPoll(), PQsetenvAbort (Ewan Mellor)
New libpq PQsetenv() function (Ewan Mellor)
create/alter user extension (Peter E)
New postmaster.pid and postmaster.opts under $PGDATA (Tatsuo)
New scripts for create/drop user/db (Peter E)
Major psql overhaul (Peter E)
Add const to libpq interface (Peter E)
New libpq function PQoidValue (Peter E)
Show specific non-aggregate causing problem with GROUP BY (Tom)
Make changes to pg_shadow recreate pg_pwd file (Peter E)
Add aggregate(DISTINCT ...) (Tom)
Allow flag to control COPY input/output of NULLs (Peter E)
Make postgres user have a password by default (Peter E)
Add CREATE/ALTER/DROP GROUP (Peter E)
All administration scripts now support --long options (Peter E,
Karel)
Vacuumdb script now supports --all option (Peter E)
ecpg new portable FETCH syntax
```

```
Add ecpg EXEC SQL IFDEF, EXEC SQL IFNDEF, EXEC SQL ELSE, EXEC SQL
ELIF
        and EXEC SQL ENDIF directives
Add pg_ctl script to control backend startup (Tatsuo)
Add postmaster.opts.default file to store startup flags (Tatsuo)
Allow --with-mb=SQL_ASCII
Increase maximum number of index keys to 16 (Bruce)
Increase maximum number of function arguments to 16 (Bruce)
Allow configuration of maximum number of index keys and arguments
(Bruce)
Allow unprivileged users to change their passwords (Peter E)
Password authentication enabled; required for new users (Peter E)
Disallow dropping a user who owns a database (Peter E)
Change initdb option --with-mb to --enable-multibyte
Add option for initdb to prompts for superuser password (Peter E)
Allow complex type casts like col::numeric(9,2) and
col::int2::float8 (Tom)
Updated user interfaces on initdb, initlocation, pg_dump, ipcclean
(Peter E)
New pg_char_to_encoding() and pg_encoding_to_char() functions
(Tatsuo)
Libpq non-blocking mode (Alfred Perlstein)
Improve conversion of types in casts that don’t specify a length
New plperl internal programming language (Mark Hollomon)
Allow COPY IN to read file that do not end with a newline (Tom)
Indicate when long identifiers are truncated (Tom)
Allow aggregates to use type equivalency (Peter E)
Add Oracle’s to_char(), to_date(), to_datetime(), to_timestamp(),
to_number()
        conversion functions (Karel Zak <zakkr@zf.jcu.cz>)
Add SELECT DISTINCT ON (expr [, expr ...]) targetlist ... (Tom)
Check to be sure ORDER BY is compatible with the DISTINCT operation
(Tom)
Add NUMERIC and int8 types to ODBC
Improve EXPLAIN results for Append, Group, Agg, Unique (Tom)
Add ALTER TABLE ... ADD FOREIGN KEY (Stephan Szabo)
Allow SELECT .. FOR UPDATE in PL/pgSQL (Hiroshi)
Enable backward sequential scan even after reaching EOF (Hiroshi)
Add btree indexing of boolean values, >= and <= (Don Baccus)
Print current line number when COPY FROM fails (Massimo)
Recognize POSIX time zone e.g. "PST+8" and "GMT-8" (Thomas)
Add DEC as synonym for DECIMAL (Thomas)
Add SESSION_USER as SQL92 keyword, same as CURRENT_USER (Thomas)
Implement SQL92 column aliases (aka correlation names) (Thomas)
Implement SQL92 join syntax (Thomas)
Make INTERVAL reserved word allowed as a column identifier (Thomas)
Implement REINDEX command (Hiroshi)
Accept ALL in aggregate function SUM(ALL col) (Tom)
Prevent GROUP BY from using column aliases (Tom)
New psql \encoding option (Tatsuo)
Allow PQrequestCancel() to terminate when in waiting-for-lock state
(Hiroshi)
Allow negation of a negative number in all cases
Add ecpg descriptors (Christof, Michael)
Allow CREATE VIEW v AS SELECT f1::char(8) FROM tbl
Allow casts with length, like foo::char(8)
```

```
New libpq functions PQsetClientEncoding(), PQclientEncoding()
(Tatsuo)
Add support for SJIS user defined characters (Tatsuo)
Larger views/rules supported
Make libpq’s PQconndefaults() thread-safe (Tom)
Disable // as comment to be ANSI conforming, should use -- (Tom)
Allow column aliases on views CREATE VIEW name (collist)
Fixes for views with subqueries (Tom)
Allow UPDATE table SET fld = (SELECT ...) (Tom)
SET command options no longer require quotes
Update pgaccess to 0.98.5
New SET SEED command
New pg_options.sample file
New SET FSYNC command (Massimo)
Allow pg_descriptions when creating tables
Allow pg_descriptions when creating types, columns, and functions
Allow psql \copy to allow delimiters(Peter E)
Allow psql to print nulls as distinct from "" [null](Peter E)

Types
-----
Many array fixes (Tom)
Allow bare column names to be subscripted as arrays (Tom)
Improve type casting of int and float constants (Tom)
Cleanups for int8 inputs, range checking, and type conversion (Tom)
Fix for SELECT timespan(’21:11:26’::time) (Tom)
netmask(’x.x.x.x/0’) is 255.255.255.255 instead of 0.0.0.0 (Oleg
Sharoiko)
Add btree index on NUMERIC (Jan)
Perl fix for large objects containing NUL characters (Douglas
Thomson)
ODBC fix for for large objects (free)
Fix indexing of cidr data type
Fix for Ethernet MAC addresses (macaddr type) comparisons
Fix for date/time types when overflows happened in computations
(Tom)
Allow array on int8 (Peter E)
Fix for rounding/overflow of NUMERIC type, like NUMERIC(4,4) (Tom)
Allow NUMERIC arrays
Fix bugs in NUMERIC ceil() and floor() functions (Tom)
Make char_length()/octet_length including trailing blanks (Tom)
Made abstime/reltime use int4 instead of time_t (Peter E)
New lztext data type for compressed text fields
Revise code to handle coercion of int and float constants (Tom)
Start at new code to implement a BIT and BIT VARYING type (Adriaan
Joubert)
NUMERIC now accepts scientific notation (Tom)
NUMERIC to int4 rounds (Tom)
Convert float4/8 to NUMERIC properly (Tom)
Allow type conversion with NUMERIC (Thomas)
Make ISO date style (2000-02-16 09:33) the default (Thomas)
Add NATIONAL CHAR [ VARYING ] (Thomas)
Allow NUMERIC round and trunc to accept negative scales (Tom)
New TIME WITH TIME ZONE type (Thomas)
Add MAX()/MIN() on time type (Thomas)
Add abs(), mod(), fac() for int8 (Thomas)
```

```
Rename functions to round(), sqrt(), cbrt(), pow() for float8
(Thomas)
Add transcendental math functions (e.g. sin(), acos()) for float8
(Thomas)
Add exp() and ln() for NUMERIC type
Rename NUMERIC power() to pow() (Thomas)
Improved TRANSLATE() function (Edwin Ramirez, Tom)
Allow X=-Y operators  (Tom)
Allow SELECT float8(COUNT(*))/(SELECT COUNT(*) FROM t) FROM t GROUP
BY f1; (Tom)
Allow LOCALE to use indexes in regular expression searches (Tom)
Allow creation of functional indexes to use default types

Performance
-----------
Prevent exponential space consumption with many AND’s and OR’s (Tom)
Collect attribute selectivity values for system columns (Tom)
Reduce memory usage of aggregates (Tom)
Fix for LIKE optimization to use indexes with multi-byte encodings
(Tom)
Fix r-tree index optimizer selectivity (Thomas)
Improve optimizer selectivity computations and functions (Tom)
Optimize btree searching for cases where many equal keys exist (Tom)
Enable fast LIKE index processing only if index present (Tom)
Re-use free space on index pages with duplicates (Tom)
Improve hash join processing (Tom)
Prevent descending sort if result is already sorted(Hiroshi)
Allow commuting of index scan query qualifications (Tom)
Prefer index scans in cases where ORDER BY/GROUP BY is required
(Tom)
Allocate large memory requests in fix-sized chunks for performance
(Tom)
Fix vacuum’s performance by reducing memory allocation requests
(Tom)
Implement constant-expression simplification (Bernard Frankpitt,
Tom)
Use secondary columns to be used to determine start of index scan
(Hiroshi)
Prevent quadruple use of disk space when doing internal sorting
(Tom)
Faster sorting by calling fewer functions (Tom)
Create system indexes to match all system caches (Bruce, Hiroshi)
Make system caches use system indexes (Bruce)
Make all system indexes unique (Bruce)
Improve pg_statistics management for VACUUM speed improvement (Tom)
Flush backend cache less frequently (Tom, Hiroshi)
COPY now reuses previous memory allocation, improving performance
(Tom)
Improve optimization cost estimation (Tom)
Improve optimizer estimate of range queries x > lowbound AND x <
highbound (Tom)
Use DNF instead of CNF where appropriate (Tom, Taral)
Further cleanup for OR-of-AND WHERE-clauses (Tom)
Make use of index in OR clauses (x = 1 AND y = 2) OR (x = 2 AND y =
4) (Tom)
Smarter optimizer computations for random index page access (Tom)
New SET variable to control optimizer costs (Tom)
```

```
Optimizer queries based on LIMIT, OFFSET, and EXISTS qualifications
(Tom)
Reduce optimizer internal housekeeping of join paths for speedup
(Tom)
Major subquery speedup (Tom)
Fewer fsync writes when fsync is not disabled (Tom)
Improved LIKE optimizer estimates (Tom)
Prevent fsync in SELECT-only queries (Vadim)
Make index creation use psort code, because it is now faster (Tom)
Allow creation of sort temp tables > 1 Gig

Source Tree Changes
-------------------
Fix for linux PPC compile
New generic expression-tree-walker subroutine (Tom)
Change form() to varargform() to prevent portability problems
Improved range checking for large integers on Alphas
Clean up #include in /include directory (Bruce)
Add scripts for checking includes (Bruce)
Remove un-needed #include's from *.c files (Bruce)
Change #include's to use <> and "" as appropriate (Bruce)
Enable WIN32 compilation of libpq
Alpha spinlock fix from Uncle George (mailto:gatgul@voicenet.com)
Overhaul of optimizer data structures (Tom)
Fix to cygipc library (Yutaka Tanida)
Allow pgsql to work on newer Cygwin snapshots (Dan)
New catalog version number (Tom)
Add Linux ARM
Rename heap_replace to heap_update
Update for QNX (Dr. Andreas Kardos)
New platform-specific regression handling (Tom)
Rename oid8 -> oidvector and int28 -> int2vector (Bruce)
Included all yacc and lex files into the distribution (Peter E.)
Remove lextest, no longer needed (Peter E)
Fix for libpq and psql on Win32 (Magnus)
Internally change datetime and timespan into timestamp and interval
(Thomas)
Fix for plpgsql on BSDI
Add SQL_ASCII test case to the regression test (Tatsuo)
configure --with-mb now deprecated (Tatsuo)
NT fixes
NetBSD fixes Johnny C. Lam (mailto:lamj@stat.cmu.edu)
Fixes for Alpha compiles
New multibyte encodings
```

Release 6.5.3

This is basically a cleanup release for 6.5.2. We have added a new pgaccess that was missing in 6.5.2, and installed an NT-specific fix.

Migration to v6.5.3

A dump/restore is *not* required for those running 6.5.*.

Detailed Change List

```
Updated version of pgaccess 0.98
NT-specific patch
Fix dumping rules on inherited tables
```

Release 6.5.2

This is basically a cleanup release for 6.5.1. We have fixed a variety of problems reported by 6.5.1 users.

Migration to v6.5.2

A dump/restore is *not* required for those running 6.5.*.

Detailed Change List

```
subselect+CASE fixes(Tom)
Add SHLIB_LINK setting for solaris_i386 and solaris_sparc
ports(Daren Sefcik)
Fixes for CASE in WHERE join clauses(Tom)
Fix BTScan abort(Tom)
Repair the check for redundant UNIQUE and PRIMARY KEY
indices(Thomas)
Improve it so that it checks for multi-column constraints(Thomas)
Fix for Win32 making problem with MB enabled(Hiroki Kataoka)
Allow BSD yacc and bison to compile pl code(Bruce)
Fix SET NAMES working
int8 fixes(Thomas)
Fix vacuum’s memory consumption(Hiroshi,Tatsuo)
Reduce the total memory consumption of vacuum(Tom)
Fix for timestamp(datetime)
Rule deparsing bugfixes(Tom)
Fix quoting problems in mkMakefile.tcldefs.sh.in and
mkMakefile.tkdefs.sh.in(Tom)
This is to re-use space on index pages freed by vacuum(Vadim)
document -x for pg_dump(Bruce)
Fix for unary operators in rule deparser(Tom)
Comment out FileUnlink of excess segments during mdtruncate()(Tom)
Irix linking fix from Yu Cao >yucao@falcon.kla-tencor.com<
Repair logic error in LIKE: should not return LIKE_ABORT
   when reach end of pattern before end of text(Tom)
Repair incorrect cleanup of heap memory allocation during
transaction abort(Tom)
Updated version of pgaccess 0.98
```

Release 6.5.1

This is basically a cleanup release for 6.5. We have fixed a variety of problems reported by 6.5 users.

Migration to v6.5.1

A dump/restore is *not* required for those running 6.5.

Detailed Change List

```
Add NT README file
Portability fixes for linux_ppc, Irix, linux_alpha, OpenBSD, alpha
Remove QUERY_LIMIT, use SELECT...LIMIT
Fix for EXPLAIN on inheritance(Tom)
Patch to allow vacuum on multi-segment tables(Hiroshi)
R-Tree optimizer selectivity fix(Tom)
ACL file descriptor leak fix(Atsushi Ogawa)
New expresssion subtree code(Tom)
Avoid disk writes for read-only transactions(Vadim)
Fix for removal of temp tables if last transaction was
aborted(Bruce)
Fix to prevent too large tuple from being created(Bruce)
plpgsql fixes
Allow port numbers 32k - 64k(Bruce)
Add ^ precidence(Bruce)
Rename sort files called pg_temp to pg_sorttemp(Bruce)
Fix for microseconds in time values(Tom)
Tutorial source cleanup
New linux_m68k port
Fix for sorting of NULL’s in some cases(Tom)
Shared library dependencies fixed (Tom)
Fixed glitches affecting GROUP BY in subselects(Tom)
Fix some compiler warnings (Tomoaki Nishiyama)
Add Win1250 (Czech) support (Pavel Behal)
```

Release 6.5

This release marks a major step in the development team’s mastery of the source code we inherited from Berkeley. You will see we are now easily adding major features, thanks to the increasing size and experience of our world-wide development team.

Here is a brief summary of the more notable changes:

Multi-version concurrency control(MVCC)

This removes our old table-level locking, and replaces it with a locking system that is superior to most commercial database systems. In a traditional system, each row that is modified is locked until committed, preventing reads by other users. MVCC uses the natural multi-version nature of PostgreSQL to allow readers to continue reading consistent data during writer activity. Writers continue to use the compact pg_log transaction system. This is all performed without having to allocate a lock for every row like traditional database systems. So, basically, we no longer are restricted by simple table-level locking; we have something better than row-level locking.

Hot backups from pg_dump

pg_dump takes advantage of the new MVCC features to give a consistant database dump/backup while the database stays online and available for queries.

Numeric data type

We now have a true numeric data type, with user-specified precision.

Temporary tables

Temporary tables are guaranteed to have unique names within a database session, and are destroyed on session exit.

New SQL features

We now have CASE, INTERSECT, and EXCEPT statement support. We have new LIMIT/OFFSET, SET TRANSACTION ISOLATION LEVEL, SELECT ... FOR UPDATE, and an improved LOCK TABLE command.

Speedups

We continue to speed up PostgreSQL, thanks to the variety of talents within our team. We have sped up memory allocation, optimization, table joins, and row transfer routines.

Ports

We continue to expand our port list, this time including WinNT/ix86 and NetBSD/arm32.

Interfaces

Most interfaces have new versions, and existing functionality has been improved.

Documentation

New and updated material is present throughout the documentation. New FAQs have been contributed for SGI and AIX platforms. The *Tutorial* has introductory information on SQL from Stefan Simkovics. For the *User's Guide*, there are reference pages covering the postmaster and more utility programs, and a new appendix contains details on date/time behavior. The *Administrator's Guide* has a new chapter on troubleshooting from Tom Lane. And the *Programmer's Guide* has a description of query processing, also from Stefan, and details on obtaining the Postgres source tree via anonymous CVS and CVSup.

Migration to v6.5

A dump/restore using pg_dump is required for those wishing to migrate data from any previous release of Postgres. pg_upgrade can *not* be used to upgrade to this release because the on-disk structure of the tables has changed compared to previous releases.

The new Multi-Version Concurrency Control (MVCC) features can give somewhat different behaviors in multi-user environments. *Read and understand the following section to ensure that your existing applications will give you the behavior you need.*

Multi-Version Concurrency Control

Because readers in 6.5 don't lock data, regardless of transaction isolation level, data read by one transaction can be overwritten by another. In other words, if a row is returned by **SELECT** it doesn't mean that this row really exists at the time it is returned (i.e. sometime after the statement or transaction began) nor that the row is protected from being deleted or updated by concurrent transactions before the current transaction does a commit or rollback.

To ensure the actual existence of a row and protect it against concurrent updates one must use **SELECT FOR UPDATE** or an appropriate **LOCK TABLE** statement. This should be taken into account when porting applications from previous releases of Postgres and other environments.

Keep the above in mind if you are using `contrib/refint.*` triggers for referential integrity. Additional technics are required now. One way is to use **LOCK parent_table IN SHARE ROW EXCLUSIVE MODE** command if a transaction is going to update/delete a primary key and use **LOCK parent_table IN SHARE MODE** command if a transaction is going to update/insert a foreign key.

> **Note:** Note that if you run a transaction in SERIALIZABLE mode then you must execute the **LOCK** commands above before execution of any DML statement (**SELECT/INSERT/DELETE/UPDATE/FETCH/COPY_TO**) in the transaction.

These inconveniences will disappear in the future when the ability to read dirty (uncommitted) data (regardless of isolation level) and true referential integrity will be implemented.

Detailed Change List

```
Bug Fixes
---------
Fix text<->float8 and text<->float4 conversion functions(Thomas)
Fix for creating tables with mixed-case constraints(Billy)
Change exp()/pow() behavior to generate error on
underflow/overflow(Jan)
Fix bug in pg_dump -z
Memory overrun cleanups(Tatsuo)
Fix for lo_import crash(Tatsuo)
Adjust handling of data type names to suppress double quotes(Thomas)
Use type coercion for matching columns and DEFAULT(Thomas)
Fix deadlock so it only checks once after one second of sleep(Bruce)
Fixes for aggregates and PL/pgsql(Hiroshi)
Fix for subquery crash(Vadim)
Fix for libpq function PQfnumber and case-insensitive names(Bahman
Rafatjoo)
Fix for large object write-in-middle, no extra block, memory
consumption(Tatsuo)
Fix for pg_dump -d or -D and  quote special characters in INSERT
Repair serious problems with dynahash(Tom)
Fix INET/CIDR portability problems
Fix problem with selectivity error in ALTER TABLE ADD COLUMN(Bruce)
Fix executor so mergejoin of different column types works(Tom)
Fix for Alpha OR selectivity bug
Fix OR index selectivity problem(Bruce)
Fix so \d shows proper length for char()/varchar()(Ryan)
Fix tutorial code(Clark)
Improve destroyuser checking(Oliver)
```

```
Fix for Kerberos(Rodney McDuff)
Fix for dropping database while dirty buffers(Bruce)
Fix so sequence nextval() can be case-sensitive(Bruce)
Fix !!= operator
Drop buffers before destroying database files(Bruce)
Fix case where executor evaluates functions twice(Tatsuo)
Allow sequence nextval actions to be case-sensitive(Bruce)
Fix optimizer indexing not working for negative numbers(Bruce)
Fix for memory leak in executor with fjIsNull
Fix for aggregate memory leaks(Erik Riedel)
Allow username containing a dash GRANT permissions
Cleanup of NULL in inet types
Clean up system table bugs(Tom)
Fix problems of PAGER and \? command(Masaaki Sakaida)
Reduce default multi-segment file size limit to 1GB(Peter)
Fix for dumping of CREATE OPERATOR(Tom)
Fix for backward scanning of cursors(Hiroshi Inoue)
Fix for COPY FROM STDIN when using \i(Tom)
Fix for subselect is compared inside an expression(Jan)
Fix handling of error reporting while returning rows(Tom)
Fix problems with reference to array types(Tom,Jan)
Prevent UPDATE SET oid(Jan)
Fix pg_dump so -t option can handle case-sensitive tablenames
Fixes for GROUP BY in special cases(Tom, Jan)
Fix for memory leak in failed queries(Tom)
DEFAULT now supports mixed-case identifiers(Tom)
Fix for multi-segment uses of DROP/RENAME table, indexes(Ole Gjerde)
Disable use of pg_dump with both -o and -d options(Bruce)
Allow pg_dump to properly dump GROUP permissions(Bruce)
Fix GROUP BY in INSERT INTO table SELECT * FROM table2(Jan)
Fix for computations in views(Jan)
Fix for aggregates on array indexes(Tom)
Fix for DEFAULT handles single quotes in value requiring too many
quotes
Fix security problem with non-super users importing/exporting large
objects(Tom)
Rollback of transaction that creates table cleaned up properly(Tom)
Fix to allow long table and column names to generate proper serial
names(Tom)

Enhancements
------------
Add "vacuumdb" utility
Speed up libpq by allocating memory better(Tom)
EXPLAIN all indices used(Tom)
Implement CASE, COALESCE, NULLIF  expression(Thomas)
New pg_dump table output format(Constantin)
Add string min()/max() functions(Thomas)
Extend new type coercion techniques to aggregates(Thomas)
New moddatetime contrib(Terry)
Update to pgaccess 0.96(Constantin)
Add routines for single-byte "char" type(Thomas)
Improved substr() function(Thomas)
Improved multi-byte handling(Tatsuo)
Multi-version concurrency control/MVCC(Vadim)
New Serialized mode(Vadim)
Fix for tables over 2gigs(Peter)
```

```
New SET TRANSACTION ISOLATION LEVEL(Vadim)
New LOCK TABLE IN ... MODE(Vadim)
Update ODBC driver(Byron)
New NUMERIC data type(Jan)
New SELECT FOR UPDATE(Vadim)
Handle "NaN" and "Infinity" for input values(Jan)
Improved date/year handling(Thomas)
Improved handling of backend connections(Magnus)
New options ELOG_TIMESTAMPS and USE_SYSLOG options for log
files(Massimo)
New TCL_ARRAYS option(Massimo)
New INTERSECT and EXCEPT(Stefan)
New pg_index.indisprimary for primary key tracking(D’Arcy)
New pg_dump option to allow dropping of tables before
creation(Brook)
Speedup of row output routines(Tom)
New READ COMMITTED isolation level(Vadim)
New TEMP tables/indexes(Bruce)
Prevent sorting if result is already sorted(Jan)
New memory allocation optimization(Jan)
Allow psql to do \p\g(Bruce)
Allow multiple rule actions(Jan)
Added LIMIT/OFFSET functionality(Jan)
Improve optimizer when joining a large number of tables(Bruce)
New intro to SQL from S. Simkovics’ Master’s Thesis (Stefan, Thomas)
New intro to backend processing from S. Simkovics’ Master’s Thesis
(Stefan)
Improved int8 support(Ryan Bradetich, Thomas, Tom)
New routines to convert between int8 and text/varchar types(Thomas)
New bushy plans, where meta-tables are joined(Bruce)
Enable right-hand queries by default(Bruce)
Allow reliable maximum number of backends to be set at configure
time
      (--with-maxbackends and postmaster switch (-N backends))(Tom)
GEQO default now 10 tables because of optimizer speedups(Tom)
Allow NULL=Var for MS-SQL portability(Michael, Bruce)
Modify contrib check_primary_key() so either "automatic" or
"dependent"(Anand)
Allow psql \d on a view show query(Ryan)
Speedup for LIKE(Bruce)
Ecpg fixes/features, see src/interfaces/ecpg/ChangeLog file(Michael)
JDBC fixes/features, see src/interfaces/jdbc/CHANGELOG(Peter)
Make % operator have precedence like /(Bruce)
Add new postgres -O option to allow system table structure
changes(Bruce)
Update contrib/pginterface/findoidjoins script(Tom)
Major speedup in vacuum of deleted rows with indexes(Vadim)
Allow non-SQL functions to run different versions based on
arguments(Tom)
Add -E option that shows actual queries sent by \dt and
friends(Masaaki Sakaida)
Add version number in startup banners for psql(Masaaki Sakaida)
New contrib/vacuumlo removes large objects not referenced(Peter)
New initialization for table sizes so non-vacuumed tables perform
better(Tom)
Improve error messages when a connection is rejected(Tom)
Support for arrays of char() and varchar() fields(Massimo)
```

```
Overhaul of hash code to increase reliability and performance(Tom)
Update to PyGreSQL 2.4(D’Arcy)
Changed debug options so -d4 and -d5 produce different node
displays(Jan)
New pg_options: pretty_plan, pretty_parse, pretty_rewritten(Jan)
Better optimization statistics for system table access(Tom)
Better handling of non-default block sizes(Massimo)
Improve GEQO optimizer memory consumption(Tom)
UNION now suppports ORDER BY of columns not in target list(Jan)
Major libpq++ improvements(Vince Vielhaber)
pg_dump now uses -z(ACL’s) as default(Bruce)
backend cache, memory speedups(Tom)
have pg_dump do everything in one snapshot transaction(Vadim)
fix for large object memory leakage, fix for pg_dumping(Tom)
INET type now respects netmask for comparisons
Make VACUUM ANALYZE only use a readlock(Vadim)
Allow VIEWs on UNIONS(Jan)
pg_dump now can generate consistent snapshots on active
databases(Vadim)

Source Tree Changes
-------------------
Improve port matching(Tom)
Portability fixes for SunOS
Add NT/Win32 backend port and enable dynamic loading(Magnus and
Daniel Horak)
New port to Cobalt Qube(Mips) running Linux(Tatsuo)
Port to NetBSD/m68k(Mr. Mutsuki Nakajima)
Port to NetBSD/sun3(Mr. Mutsuki Nakajima)
Port to NetBSD/macppc(Toshimi Aoki)
Fix for tcl/tk configuration(Vince)
Removed CURRENT keyword for rule queries(Jan)
NT dynamic loading now works(Daniel Horak)
Add ARM32 support(Andrew McMurry)
Better support for HPUX 11 and Unixware
Improve file handling to be more uniform, prevent file descriptor
leak(Tom)
New install commands for plpgsql(Jan)
```

Release 6.4.2

The 6.4.1 release was improperly packaged. This also has one additional bug fix.

Migration to v6.4.2

A dump/restore is *not* required for those running 6.4.*.

Detailed Change List

```
Fix for datetime constant problem on some platforms(Thomas)
```

Release 6.4.1

This is basically a cleanup release for 6.4. We have fixed a variety of problems reported by 6.4 users.

Migration to v6.4.1

A dump/restore is *not* required for those running 6.4.

Detailed Change List

```
Add pg_dump -N flag to force double quotes around identifiers.  This
is
        the default(Thomas)
Fix for NOT in where clause causing crash(Bruce)
EXPLAIN VERBOSE coredump fix(Vadim)
Fix shared-library problems on Linux
Fix test for table existance to allow mixed-case and whitespace in
        the table name(Thomas)
Fix a couple of pg_dump bugs
Configure matches template/.similar entries better(Tom)
Change builtin function names from SPI_* to spi_*
OR WHERE clause fix(Vadim)
Fixes for mixed-case table names(Billy)
contrib/linux/postgres.init.csh/sh fix(Thomas)
libpq memory overrun fix
SunOS fixes(Tom)
Change exp() behavior to generate error on underflow(Thomas)
pg_dump fixes for memory leak, inheritance constraints, layout
change
update pgaccess to 0.93
Fix prototype for 64-bit platforms
Multi-byte fixes(Tatsuo)
New ecpg man page
Fix memory overruns(Tatsuo)
Fix for lo_import() crash(Bruce)
Better search for install program(Tom)
Timezone fixes(Tom)
HPUX fixes(Tom)
Use implicit type coercion for matching DEFAULT values(Thomas)
Add routines to help with single-byte (internal) character
type(Thomas)
Compilation of libpq for Win32 fixes(Magnus)
Upgrade to PyGreSQL 2.2(D’Arcy)
```

Release 6.4

There are *many* new features and improvements in this release. Thanks to our developers and maintainers, nearly every aspect of the system has received some attention since the previous release. Here is a brief, incomplete summary:

Views and rules are now functional thanks to extensive new code in the rewrite rules system from Jan Wieck. He also wrote a chapter on it for the *Programmer’s Guide*.

Jan also contributed a second procedural language, PL/pgSQL, to go with the original PL/pgTCL procedural language he contributed last release.

We have optional multiple-byte character set support from Tatsuo Iishi to complement our existing locale support.

Client/server communications has been cleaned up, with better support for asynchronous messages and interrupts thanks to Tom Lane.

The parser will now perform automatic type coercion to match arguments to available operators and functions, and to match columns and expressions with target columns. This uses a generic mechanism which supports the type extensibility features of Postgres. There is a new chapter in the *User's Guide* which covers this topic.

Three new data types have been added. Two types, inet and cidr, support various forms of IP network, subnet, and machine addressing. There is now an 8-byte integer type available on some platforms. See the chapter on data types in the *User's Guide* for details. A fourth type, serial, is now supported by the parser as an amalgam of the int4 type, a sequence, and a unique index.

Several more SQL92-compatible syntax features have been added, including **INSERT DEFAULT VALUES**

The automatic configuration and installation system has received some attention, and should be more robust for more platforms than it has ever been.

Migration to v6.4

A dump/restore using pg_dump or pg_dumpall is required for those wishing to migrate data from any previous release of Postgres.

Detailed Change List

```
Bug Fixes
---------
Fix for a tiny memory leak in PQsetdb/PQfinish(Bryan)
Remove char2-16 data types, use char/varchar(Darren)
Pqfn not handles a NOTICE message(Anders)
Reduced busywaiting overhead for spinlocks with many backends (dg)
Stuck spinlock detection (dg)
Fix up "ISO-style" timespan decoding and encoding(Thomas)
Fix problem with table drop after rollback of transaction(Vadim)
Change error message and remove non-functional update message(Vadim)
Fix for COPY array checking
Fix for SELECT 1 UNION SELECT NULL
Fix for buffer leaks in large object calls(Pascal)
Change owner from oid to int4 type(Bruce)
Fix a bug in the oracle compatibility functions btrim() ltrim() and
rtrim()
Fix for shared invalidation cache overflow(Massimo)
Prevent file descriptor leaks in failed COPY's(Bruce)
Fix memory leak in libpgtcl's pg_select(Constantin)
Fix problems with username/passwords over 8 characters(Tom)
Fix problems with handling of asynchronous NOTIFY in backend(Tom)
Fix of many bad system table entries(Tom)

Enhancements
```

```
------------
Upgrade ecpg and ecpglib,see src/interfaces/ecpc/ChangeLog(Michael)
Show the index used in an EXPLAIN(Zeugswetter)
EXPLAIN  invokes  rule system and shows plan(s) for rewritten
queries(Jan)
Multi-byte awareness of many data types and functions, via
configure(Tatsuo)
New configure --with-mb option(Tatsuo)
New initdb --pgencoding option(Tatsuo)
New createdb -E multibyte option(Tatsuo)
Select version(); now returns PostgreSQL version(Jeroen)
Libpq now allows asynchronous clients(Tom)
Allow cancel from client of backend query(Tom)
Psql now cancels query with Control-C(Tom)
Libpq users need not issue dummy queries to get NOTIFY messages(Tom)
NOTIFY now sends sender’s PID, so you can tell whether it was your
own(Tom)
PGresult struct now includes associated error message, if any(Tom)
Define "tz_hour" and "tz_minute" arguments to date_part()(Thomas)
Add routines to convert between varchar and bpchar(Thomas)
Add routines to allow sizing of varchar and bpchar into target
columns(Thomas)
Add bit flags to support timezonehour and minute in data
retrieval(Thomas)
Allow more variations on valid floating point numbers (e.g. ".1",
"1e6")(Thomas)
Fixes for unary minus parsing with leading spaces(Thomas)
Implement TIMEZONE_HOUR, TIMEZONE_MINUTE per SQL92 specs(Thomas)
Check for and properly ignore FOREIGN KEY column constraints(Thomas)
Define USER as synonym for CURRENT_USER per SQL92 specs(Thomas)
Enable HAVING clause but no fixes elsewhere yet.
Make "char" type a synonym for "char(1)" (actually implemented as
bpchar)(Thomas)
Save string type if specified for DEFAULT clause handling(Thomas)
Coerce operations involving different data types(Thomas)
Allow some index use for columns of different types(Thomas)
Add capabilities for automatic type conversion(Thomas)
Cleanups for large objects, so file is truncated on open(Peter)
Readline cleanups(Tom)
Allow psql  \f \ to make spaces as delimiter(Bruce)
Pass pg_attribute.atttypmod to the frontend for column field
lengths(Tom,Bruce)
Msql compatibility library in /contrib(Aldrin)
Remove the requirement that ORDER/GROUP BY clause identifiers be
included in the target list(David)
Convert columns to match columns in UNION clauses(Thomas)
Remove fork()/exec() and only do fork()(Bruce)
Jdbc cleanups(Peter)
Show backend status on ps command line(only works on some
platforms)(Bruce)
Pg_hba.conf now has a sameuser option in the database field
Make lo_unlink take oid param, not int4
New DISABLE_COMPLEX_MACRO for compilers that can’t handle our
macros(Bruce)
Libpgtcl now handles NOTIFY as a Tcl event, need not send dummy
queries(Tom)
libpgtcl cleanups(Tom)
```

```
Add -error option to libpgtcl’s pg_result command(Tom)
New locale patch, see docs/README/locale(Oleg)
Fix for pg_dump so CONSTRAINT and CHECK syntax is correct(ccb)
New contrib/lo code for large object orphan removal(Peter)
New psql command "SET CLIENT_ENCODING TO ’encoding’" for multi-bytes
feature, see /doc/README.mb(Tatsuo)
/contrib/noupdate code to revoke update permission on a column
Libpq can now be compiled on win32(Magnus)
Add PQsetdbLogin() in libpq
New 8-byte integer type, checked by configure for OS support(Thomas)
Better support for quoted table/column names(Thomas)
Surround table and column names with double-quotes in
pg_dump(Thomas)
PQreset() now works with passwords(Tom)
Handle case of GROUP BY target list column number out of
range(David)
Allow UNION in subselects
Add auto-size to screen to \d? commands(Bruce)
Use UNION to show all \d? results in one query(Bruce)
Add \d? field search feature(Bruce)
Pg_dump issues fewer \connect requests(Tom)
Make pg_dump -z flag work better, document it in manual page(Tom)
Add HAVING clause with full support for subselects and
unions(Stephan)
Full text indexing routines in contrib/fulltextindex(Maarten)
Transaction ids now stored in shared memory(Vadim)
New PGCLIENTENCODING when issuing COPY command(Tatsuo)
Support for SQL92 syntax "SET NAMES"(Tatsuo)
Support for LATIN2-5(Tatsuo)
Add UNICODE regression test case(Tatsuo)
Lock manager cleanup, new locking modes for LLL(Vadim)
Allow index use with OR clauses(Bruce)
Allows "SELECT NULL ORDER BY 1;"
Explain VERBOSE prints the plan, and now pretty-prints the plan to
the postmaster log file(Bruce)
Add Indices display to \d command(Bruce)
Allow GROUP BY on functions(David)
New pg_class.relkind for large objects(Bruce)
New way to send libpq NOTICE messages to a different location(Tom)
New \w write command to psql(Bruce)
New /contrib/findoidjoins scans oid columns to find join
relationships(Bruce)
Allow binary-compatible indices to be considered when checking for
valid
indices for restriction clauses containing a constant(Thomas)
New ISBN/ISSN code in /contrib/isbn_issn
Allow NOT LIKE, IN, NOT IN, BETWEEN, and NOT BETWEEN
constraint(Thomas)
New rewrite system fixes many problems with rules and views(Jan)
        * Rules on relations work
        * Event qualifications on insert/update/delete work
        * New OLD variable to reference CURRENT, CURRENT will be
remove in future
        * Update rules can reference NEW and OLD in rule
qualifications/actions
        * Insert/update/delete rules on views work
        * Multiple rule actions are now supported, surrounded by
parentheses
        * Regular users can create views/rules on tables they have
RULE permits
        * Rules and views inherit the permissions on the creator
        * No rules at the column level
        * No UPDATE NEW/OLD rules
        * New pg_tables, pg_indexes, pg_rules and pg_views system
views
        * Only a single action on SELECT rules
        * Total rewrite overhaul, perhaps for 6.5
        * handle subselects
        * handle aggregates on views
        * handle insert into select from view works
System indexes are now multi-key(Bruce)
Oidint2, oidint4, and oidname types are removed(Bruce)
Use system cache for more system table lookups(Bruce)
New backend programming language PL/pgSQL in backend/pl(Jan)
New SERIAL data type, auto-creates sequence/index(Thomas)
Enable assert checking without a recompile(Massimo)
User lock enhancements(Massimo)
New setval() command to set sequence value(Massimo)
Auto-remove unix socket file on startup if no postmaster
running(Massimo)
Conditional trace package(Massimo)
New UNLISTEN command(Massimo)
Psql and libpq now compile under win32 using win32.mak(Magnus)
Lo_read no longer stores trailing NULL(Bruce)
Identifiers are now truncated to 31 characters internally(Bruce)
Createuser options now availble on the command line
Code for 64-bit integer supported added, configure tested, int8
type(Thomas)
Prevent file descriptor leaf from failed COPY(Bruce)
New pg_upgrade command(Bruce)
Updated /contrib directories(Massimo)
New CREATE TABLE DEFAULT VALUES statement available(Thomas)
New INSERT INTO TABLE DEFAULT VALUES statement available(Thomas)
New DECLARE and FETCH feature(Thomas)
libpq’s internal structures now not exported(Tom)
Allow up to 8 key indexes(Bruce)
Remove ARCHIVE keyword, that is no longer used(Thomas)
pg_dump -n flag to supress quotes around indentifiers
disable system columns for views(Jan)
new INET and CIDR types for network addresses(TomH, Paul)
no more double quotes in psql output
pg_dump now dumps views(Terry)
new SET QUERY_LIMIT(Tatsuo,Jan)

Source Tree Changes
-------------------
/contrib cleanup(Jun)
Inline some small functions called for every row(Bruce)
Alpha/linux fixes
Hp/UX cleanups(Tom)
Multi-byte regression tests(Soonmyung.)
Remove --disabled options from configure
Define PGDOC to use POSTGRESDIR by default
```

```
Make regression optional
Remove extra braces code to pgindent(Bruce)
Add bsdi shared library support(Bruce)
New --without-CXX support configure option(Brook)
New FAQ_CVS
Update backend flowchart in tools/backend(Bruce)
Change atttypmod from int16 to int32(Bruce, Tom)
Getrusage() fix for platforms that do not have it(Tom)
Add PQconnectdb, PGUSER, PGPASSWORD to libpq man page
NS32K platform fixes(Phil Nelson, John Buller)
Sco 7/UnixWare 2.x fixes(Billy,others)
Sparc/Solaris 2.5 fixes(Ryan)
Pgbuiltin.3 is obsolete, move to doc files(Thomas)
Even more documention(Thomas)
Nextstep support(Jacek)
Aix support(David)
pginterface manual page(Bruce)
shared libraries all have version numbers
merged all OS-specific shared library defines into one file
smarter TCL/TK configuration checking(Billy)
smarter perl configuration(Brook)
configure uses supplied install-sh if no install script found(Tom)
new Makefile.shlib for shared library configuration(Tom)
```

Release 6.3.2

This is a bugfix release for 6.3.x. Refer to the release notes for v6.3 for a more complete summary of new features.

Summary:

Repairs automatic configuration support for some platforms, including Linux, from breakage inadvertently introduced in v6.3.1.

Correctly handles function calls on the left side of BETWEEN and LIKE clauses.

A dump/restore is NOT required for those running 6.3 or 6.3.1. A ’make distclean’, ’make’, and ’make install’ is all that is required. This last step should be performed while the postmaster is not running. You should re-link any custom applications that use Postgres libraries.

For upgrades from pre-v6.3 installations, refer to the installation and migration instructions for v6.3.

Detailed Change List

```
Changes
-------
Configure detection improvements for tcl/tk(Brook Milligan, Alvin)
Manual page improvements(Bruce)
BETWEEN and LIKE fix(Thomas)
fix for psql \connect used by pg_dump(Oliver Elphick)
New odbc driver
pgaccess, version 0.86
qsort removed, now uses libc version, cleanups(Jeroen)
fix for buffer over-runs detected(Maurice Gittens)
fix for buffer overrun in libpgtcl(Randy Kunkee)
fix for UNION with DISTINCT or ORDER BY(Bruce)
gettimeofday configure check(Doug Winterburn)
Fix "indexes not used" bug(Vadim)
docs additions(Thomas)
Fix for backend memory leak(Bruce)
libreadline cleanup(Erwan MAS)
Remove DISTDIR(Bruce)
Makefile dependency cleanup(Jeroen van Vianen)
ASSERT fixes(Bruce)
```

Release 6.3.1

Summary:

Additional support for multi-byte character sets.

Repair byte ordering for mixed-endian clients and servers.

Minor updates to allowed SQL syntax.

Improvements to the configuration autodetection for installation.

A dump/restore is NOT required for those running 6.3. A ’make distclean’, ’make’, and ’make install’ is all that is required. This last step should be performed while the postmaster is not running. You should re-link any custom applications that use Postgres libraries.

For upgrades from pre-v6.3 installations, refer to the installation and migration instructions for v6.3.

Detailed Change List

```
Changes
-------
ecpg cleanup/fixes, now version 1.1(Michael Meskes)
pg_user cleanup(Bruce)
large object fix for pg_dump and tclsh (alvin)
LIKE fix for multiple adjacent underscores
fix for redefining builtin functions(Thomas)
ultrix4 cleanup
upgrade to pg_access 0.83
updated CLUSTER manual page
multi-byte character set support, see doc/README.mb(Tatsuo)
configure --with-pgport fix
pg_ident fix
big-endian fix for backend communications(Kataoka)
SUBSTR() and substring() fix(Jan)
several jdbc fixes(Peter)
libpgtcl improvements, see libptcl/README(Randy Kunkee)
Fix for "Datasize = 0" error(Vadim)
Prevent \do from wrapping(Bruce)
Remove duplicate Russian character set entries
Sunos4 cleanup
Allow optional TABLE keyword in LOCK and SELECT INTO(Thomas)
CREATE SEQUENCE options to allow a negative integer(Thomas)
Add "PASSWORD" as an allowed column identifier(Thomas)
Add checks for UNION target fields(Bruce)
Fix Alpha port(Dwayne Bailey)
Fix for text arrays containing quotes(Doug Gibson)
Solaris compile fix(Albert Chin-A-Young)
Better identify tcl and tk libs and includes(Bruce)
```

Release 6.3

There are *many* new features and improvements in this release. Here is a brief, incomplete summary:

Many new SQL features, including full SQL92 subselect capability (everything is here but target-list subselects).

Support for client-side environment variables to specify time zone and date style.

Socket interface for client/server connection. This is the default now so you may need to start postmaster with the `-i` flag.

Better password authorization mechanisms. Default table permissions have changed.

Old-style *time travel* has been removed. Performance has been improved.

Note: Bruce Momjian wrote the following notes to introduce the new release.

There are some general 6.3 issues that I want to mention. These are only the big items that can not be described in one sentence. A review of the detailed changes list is still needed.

First, we now have subselects. Now that we have them, I would like to mention that without subselects, SQL is a very limited language. Subselects are a major feature, and you should review your code for places where subselects provide a better solution for your queries. I think you will find that there are more uses for subselects than you may think. Vadim has put us on the big SQL map with subselects, and fully functional ones too. The only thing you can't do with subselects is to use them in the target list.

Second, 6.3 uses unix domain sockets rather than TCP/IP by default. To enable connections from other machines, you have to use the new postmaster -i option, and of course edit pg_hba.conf. Also, for this reason, the format of pg_hba.conf has changed.

Third, char() fields will now allow faster access than varchar() or text. Specifically, the text and varchar() have a penalty for access to any columns after the first column of this type. char() used to also have this access penalty, but it no longer does. This may suggest that you redesign some of your tables, especially if you have short character columns that you have defined as varchar() or text. This and other changes make 6.3 even faster than earlier releases.

We now have passwords definable independent of any Unix file. There are new SQL USER commands. See the pg_hba.conf manual page for more information. There is a new table, pg_shadow, which is used to store user information and user passwords, and it by default only SELECT-able by the postgres super-user. pg_user is now a view of pg_shadow, and is SELECT-able by PUBLIC. You should keep using pg_user in your application without changes.

User-created tables now no longer have SELECT permission to PUBLIC by default. This was done because the ANSI standard requires it. You can of course GRANT any permissions you want after the table is created. System tables continue to be SELECT-able by PUBLIC.

We also have real deadlock detection code. No more sixty-second timeouts. And the new locking code implements a FIFO better, so there should be less resource starvation during heavy use.

Many complaints have been made about inadequate documenation in previous releases. Thomas has put much effort into many new manuals for this release. Check out the doc/ directory.

For performance reasons, time travel is gone, but can be implemented using triggers (see pgsql/contrib/spi/README). Please check out the new \d command for types, operators, etc. Also, views have their own permissions now, not based on the underlying tables, so permissions on them have to be set separately. Check /pgsql/interfaces for some new ways to talk to Postgres.

This is the first release that really required an explanation for existing users. In many ways, this was necessary because the new release removes many limitations, and the work-arounds people were using are no longer needed.

Migration to v6.3

A dump/restore using pg_dump or pg_dumpall is required for those wishing to migrate data from any previous release of Postgres.

Detailed Change List

```
Bug Fixes
---------
Fix binary cursors broken by MOVE implementation(Vadim)
Fix for tcl library crash(Jan)
Fix for array handling, from Gerhard Hintermayer
Fix acl error, and remove duplicate pqtrace(Bruce)
Fix psql \e for empty file(Bruce)
Fix for textcat on varchar() fields(Bruce)
Fix for DBT Sendproc (Zeugswetter Andres)
Fix vacuum analyze syntax problem(Bruce)
Fix for international identifiers(Tatsuo)
Fix aggregates on inherited tables(Bruce)
Fix substr() for out-of-bounds data
Fix for select 1=1 or 2=2, select 1=1 and 2=2, and select
sum(2+2)(Bruce)
Fix notty output to show status result.  -q option still turns it
off(Bruce)
Fix for count(*), aggs with views and multiple tables and
sum(3)(Bruce)
Fix cluster(Bruce)
Fix for PQtrace start/stop several times(Bruce)
Fix a variety of locking problems like newer lock waiters getting
        lock before older waiters, and having readlock people not
share
        locks if a writer is waiting for a lock, and waiting writers
not
        getting priority over waiting readers(Bruce)
Fix crashes in psql when executing queries from external
files(James)
Fix problem with multiple order by columns, with the first one
having
```

```
        NULL values(Jeroen)
Use correct hash table support functions for float8 and int4(Thomas)
Re-enable JOIN= option in CREATE OPERATOR statement (Thomas)
Change precedence for boolean operators to match expected
behavior(Thomas)
Generate elog(ERROR) on over-large integer(Bruce)
Allow multiple-argument functions in constraint clauses(Thomas)
Check boolean input literals for ’true’,’false’,’yes’,’no’,’1’,’0’
        and throw elog(ERROR) if unrecognized(Thomas)
Major large objects fix
Fix for GROUP BY showing duplicates(Vadim)
Fix for index scans in MergeJion(Vadim)

Enhancements
------------
Subselects with EXISTS, IN, ALL, ANY keywords (Vadim, Bruce, Thomas)
New User Manual(Thomas, others)
Speedup by inlining some frequently-called functions
Real deadlock detection, no more timeouts(Bruce)
Add SQL92 "constants" CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP,
        CURRENT_USER(Thomas)
Modify constraint syntax to be SQL92-compliant(Thomas)
Implement SQL92 PRIMARY KEY and UNIQUE clauses using indices(Thomas)
Recognize SQL92 syntax for FOREIGN KEY. Throw elog notice(Thomas)
Allow NOT NULL UNIQUE constraint clause (each allowed separately
before)(Thomas)
Allow Postgres-style casting ("::") of non-constants(Thomas)
Add support for SQL3 TRUE and FALSE boolean constants(Thomas)
Support SQL92 syntax for IS TRUE/IS FALSE/IS NOT TRUE/IS NOT
FALSE(Thomas)
Allow shorter strings for boolean literals (e.g. "t", "tr",
"tru")(Thomas)
Allow SQL92 delimited identifiers(Thomas)
Implement SQL92 binary and hexadecimal string decoding (b’10’ and
x’1F’)(Thomas)
Support SQL92 syntax for type coercion of literal strings
        (e.g. "DATETIME ’now’")(Thomas)
Add conversions for int2, int4, and OID types to and from
text(Thomas)
Use shared lock when building indices(Vadim)
Free memory allocated for an user query inside transaction block
after
        this query is done, was turned off in <= 6.2.1(Vadim)
New SQL statement CREATE PROCEDURAL LANGUAGE(Jan)
New Postgres Procedural Language (PL) backend interface(Jan)
Rename pg_dump -H option to -h(Bruce)
Add Java support for passwords, European dates(Peter)
Use indices for LIKE and ~, !~ operations(Bruce)
Add hash functions for datetime and timespan(Thomas)
Time Travel removed(Vadim, Bruce)
Add paging for \d and \z, and fix \i(Bruce)
Add Unix domain socket support to backend and to frontend
library(Goran)
Implement CREATE DATABASE/WITH LOCATION and initlocation
utility(Thomas)
Allow more SQL92 and/or Postgres reserved words as column
identifiers(Thomas)
```

```
Augment support for SQL92 SET TIME ZONE...(Thomas)
SET/SHOW/RESET TIME ZONE uses TZ backend environment
variable(Thomas)
Implement SET keyword = DEFAULT and SET TIME ZONE DEFAULT(Thomas)
Enable SET TIME ZONE using TZ environment variable(Thomas)
Add PGDATESTYLE environment variable to frontend and backend
initialization(Thomas)
Add PGTZ, PGCOSTHEAP, PGCOSTINDEX, PGRPLANS, PGGEQO
        frontend library initialization environment
variables(Thomas)
Regression tests time zone automatically set with "setenv PGTZ
PST8PDT"(Thomas)
Add pg_description table for info on tables, columns, operators,
types, and
        aggregates(Bruce)
Increase 16 char limit on system table/index names to 32
characters(Bruce)
Rename system indices(Bruce)
Add ’GERMAN’ option to SET DATESTYLE(Thomas)
Define an "ISO-style" timespan output format with "hh:mm:ss"
fields(Thomas)
Allow fractional values for delta times (e.g. ’2.5 days’)(Thomas)
Validate numeric input more carefully for delta times(Thomas)
Implement day of year as possible input to date_part()(Thomas)
Define timespan_finite() and text_timespan() functions(Thomas)
Remove archive stuff(Bruce)
Allow for a pg_password authentication database that is separate
from
        the system password file(Todd)
Dump ACLs, GRANT, REVOKE permissions(Matt)
Define text, varchar, and bpchar string length functions(Thomas)
Fix Query handling for inheritance, and cost computations(Bruce)
Implement CREATE TABLE/AS SELECT (alternative to
SELECT/INTO)(Thomas)
Allow NOT, IS NULL, IS NOT NULL in constraints(Thomas)
Implement UNIONs for SELECT(Bruce)
Add UNION, GROUP, DISTINCT to INSERT(Bruce)
varchar() stores only necessary bytes on disk(Bruce)
Fix for BLOBs(Peter)
Mega-Patch for JDBC...see README_6.3 for list of changes(Peter)
Remove unused "option" from PQconnectdb()
New LOCK command and lock manual page describing deadlocks(Bruce)
Add new psql \da, \dd, \df, \do, \dS, and \dT commands(Bruce)
Enhance psql \z to show sequences(Bruce)
Show NOT NULL and DEFAULT in psql \d table(Bruce)
New psql .psqlrc file startup(Andrew)
Modify sample startup script in contrib/linux to show syslog(Thomas)
New types for IP and MAC addresses in contrib/ip_and_mac(TomH)
Unix system time conversions with date/time types in
contrib/unixdate(Thomas)
Update of contrib stuff(Massimo)
Add Unix socket support to DBD::Pg(Goran)
New python interface (PyGreSQL 2.0)(D’Arcy)
New frontend/backend protocol has a version number, network byte
order(Phil)
Security features in pg_hba.conf enhanced and documented, many
cleanups(Phil)
```

```
CHAR() now faster access than VARCHAR() or TEXT
ecpg embedded SQL preprocessor
Reduce system column overhead(Vadmin)
Remove pg_time table(Vadim)
Add pg_type attribute to identify types that need length (bpchar,
varchar)
Add report of offending line when COPY command fails
Allow VIEW permissions to be set separately from the underlying
tables.
        For security, use GRANT/REVOKE on views as appropriate(Jan)
Tables now have no default GRANT SELECT TO PUBLIC.  You must
        explicitly grant such permissions.
Clean up tutorial examples(Darren)

Source Tree Changes
-------------------
Add new html development tools, and flow chart in /tools/backend
Fix for SCO compiles
Stratus computer port Robert Gillies
Added support for shlib for BSD44_derived & i386_solaris
Make configure more automated(Brook)
Add script to check regression test results
Break parser functions into smaller files, group together(Bruce)
Rename heap_create to heap_create_and_catalog, rename heap_creatr
        to heap_create()(Bruce)
Sparc/Linux patch for locking(TomS)
Remove PORTNAME and reorganize port-specific stuff(Marc)
Add optimizer README file(Bruce)
Remove some recursion in optimizer and clean up some code
there(Bruce)
Fix for NetBSD locking(Henry)
Fix for libptcl make(Tatsuo)
AIX patch(Darren)
Change IS TRUE, IS FALSE, ... to expressions using "=" rather than
        function calls to istrue() or isfalse() to allow
optimization(Thomas)
Various fixes NetBSD/Sparc related(TomH)
Alpha linux locking(Travis,Ryan)
Change elog(WARN) to elog(ERROR)(Bruce)
FAQ for FreeBSD(Marc)
Bring in the PostODBC source tree as part of our standard
distribution(Marc)
A minor patch for HP/UX 10 vs 9(Stan)
New pg_attribute.atttypmod for type-specific info like varchar
length(Bruce)
Unixware patches(Billy)
New i386 ’lock’ for spin lock asm(Billy)
Support for multiplexed backends is removed
Start an OpenBSD port
Start an AUX port
Start a Cygnus port
Add string functions to regression suite(Thomas)
Expand a few function names formerly truncated to 16
characters(Thomas)
Remove un-needed malloc() calls and replace with palloc()(Bruce)
```

Release 6.2.1

v6.2.1 is a bug-fix and usability release on v6.2.

Summary:

Allow strings to span lines, per SQL92.

Include example trigger function for inserting user names on table updates.

This is a minor bug-fix release on v6.2. For upgrades from pre-v6.2 systems, a full dump/reload is required. Refer to the v6.2 release notes for instructions.

Migration from v6.2 to v6.2.1

This is a minor bug-fix release. A dump/reload is not required from v6.2, but is required from any release prior to v6.2.

In upgrading from v6.2, if you choose to dump/reload you will find that avg(money) is now calculated correctly. All other bug fixes take effect upon updating the executables.

Another way to avoid dump/reload is to use the following SQL command from psql to update the existing system table:

```
update pg_aggregate set aggfinalfn = ’cash_div_flt8’
 where aggname = ’avg’ and aggbasetype = 790;
```

This will need to be done to every existing database, including template1.

Detailed Change List

```
Changes in this release
-----------------------
Allow TIME and TYPE column names(Thomas)
Allow larger range of true/false as boolean values(Thomas)
Support output of "now" and "current"(Thomas)
Handle DEFAULT with INSERT of NULL properly(Vadim)
Fix for relation reference counts problem in buffer manager(Vadim)
Allow strings to span lines, like ANSI(Thomas)
Fix for backward cursor with ORDER BY(Vadim)
Fix avg(cash) computation(Thomas)
Fix for specifying a column twice in ORDER/GROUP BY(Vadim)
Documented new libpq function to return affected rows,
PQcmdTuples(Bruce)
Trigger function for inserting user names for INSERT/UPDATE(Brook
Milligan)
```

Release 6.2

A dump/restore is required for those wishing to migrate data from previous releases of Postgres.

Migration from v6.1 to v6.2

This migration requires a complete dump of the 6.1 database and a restore of the database in 6.2.

Note that the pg_dump and pg_dumpall utility from 6.2 should be used to dump the 6.1 database.

Migration from v1.x to v6.2

Those migrating from earlier 1.* releases should first upgrade to 1.09 because the COPY output format was improved from the 1.02 release.

Detailed Change List

```
Bug Fixes
---------
Fix problems with pg_dump for inheritance, sequences, archive
tables(Bruce)
Fix compile errors on overflow due to shifts, unsigned, and bad
prototypes
         from Solaris(Diab Jerius)
Fix bugs in geometric line arithmetic (bad intersection
calculations)(Thomas)
Check for geometric intersections at endpoints to avoid rounding
ugliness(Thomas)
Catch non-functional delete attempts(Vadim)
Change time function names to be more consistent(Michael Reifenberg)
Check for zero divides(Michael Reifenberg)
Fix very old bug which made tuples changed/inserted by a commnd
        visible to the command itself (so we had multiple update of
        updated tuples, etc)(Vadim)
Fix for SELECT null, 'fail' FROM pg_am (Patrick)
SELECT NULL as EMPTY_FIELD now allowed(Patrick)
Remove un-needed signal stuff from contrib/pginterface
Fix OR (where x != 1 or x isnull didn't return tuples with x NULL)
(Vadim)
Fix time_cmp function (Vadim)
Fix handling of functions with non-attribute first argument in
        WHERE clauses (Vadim)
Fix GROUP BY when order of entries is different from order
        in target list (Vadim)
Fix pg_dump for aggregates without sfunc1 (Vadim)

Enhancements
------------
Default genetic optimizer GEQO parameter is now 8(Bruce)
Allow use parameters in target list having aggregates in
functions(Vadim)
Added JDBC driver as an interface(Adrian & Peter)
pg_password utility
Return number of tuples inserted/affected by INSERT/UPDATE/DELETE
etc.(Vadim)
Triggers implemented with CREATE TRIGGER (SQL3)(Vadim)
SPI (Server Programming Interface) allows execution of queries
inside
        C-functions (Vadim)
NOT NULL implemented (SQL92)(Robson Paniago de Miranda)
Include reserved words for string handling, outer joins, and
unions(Thomas)
```

```
Implement extended comments ("/* ... */") using exclusive
states(Thomas)
Add "//" single-line comments(Bruce)
Remove some restrictions on characters in operator names(Thomas)
DEFAULT and CONSTRAINT for tables implemented (SQL92)(Vadim &
Thomas)
Add text concatenation operator and function (SQL92)(Thomas)
Support WITH TIME ZONE syntax (SQL92)(Thomas)
Support INTERVAL unit TO unit syntax (SQL92)(Thomas)
Define types DOUBLE PRECISION, INTERVAL, CHARACTER,
        and CHARACTER VARYING (SQL92)(Thomas)
Define type FLOAT(p) and rudimentary DECIMAL(p,s), NUMERIC(p,s)
(SQL92)(Thomas)
Define EXTRACT(), POSITION(), SUBSTRING(), and TRIM()
(SQL92)(Thomas)
Define CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP (SQL92)(Thomas)
Add syntax and warnings for UNION, HAVING, INNER and OUTER JOIN
(SQL92)(Thomas)
Add more reserved words, mostly for SQL92 compliance(Thomas)
Allow hh:mm:ss time entry for timespan/reltime types(Thomas)
Add center() routines for lseg, path, polygon(Thomas)
Add distance() routines for circle-polygon, polygon-polygon(Thomas)
Check explicitly for points and polygons contained within polygons
        using an axis-crossing algorithm(Thomas)
Add routine to convert circle-box(Thomas)
Merge conflicting operators for different geometric data
types(Thomas)
Replace distance operator "<===>" with "<->"(Thomas)
Replace "above" operator "!^" with ">^" and "below" operator "!|"
with "<^"(Thomas)
Add routines for text trimming on both ends, substring, and string
position(Thomas)
Added conversion routines circle(box) and poly(circle)(Thomas)
Allow internal sorts to be stored in memory rather than in
files(Bruce & Vadim)
Allow functions and operators on internally-identical types to
succeed(Bruce)
Speed up backend startup after profiling analysis(Bruce)
Inline frequently called functions for performance(Bruce)
Reduce open() calls(Bruce)
psql:  Add PAGER for \h and \?,\C fix
Fix for psql pager when no tty(Bruce)
New entab utility(Bruce)
General trigger functions for referential integrity (Vadim)
General trigger functions for time travel (Vadim)
General trigger functions for AUTOINCREMENT/IDENTITY feature (Vadim)
MOVE implementation (Vadim)

Source Tree Changes
-------------------
HPUX 10 patches (Vladimir Turin)
Added SCO support, (Daniel Harris)
mkLinux patches (Tatsuo Ishii)
Change geometric box terminology from "length" to "width"(Thomas)
Deprecate temporary unstored slope fields in geometric code(Thomas)
Remove restart instructions from INSTALL(Bruce)
Look in /usr/ucb first for install(Bruce)
```

```
Fix c++ copy example code(Thomas)
Add -o to psql manual page(Bruce)
Prevent relname unallocated string length from being copied into
database(Bruce)
Cleanup for NAMEDATALEN use(Bruce)
Fix pg_proc names over 15 chars in output(Bruce)
Add strNcpy() function(Bruce)
remove some (void) casts that are unnecessary(Bruce)
new interfaces directory(Marc)
Replace fopen() calls with calls to fd.c functions(Bruce)
Make functions static where possible(Bruce)
enclose unused functions in #ifdef NOT_USED(Bruce)
Remove call to difftime() in timestamp support to fix SunOS(Bruce &
Thomas)
Changes for Digital Unix
Portability fix for pg_dumpall(Bruce)
Rename pg_attribute.attnvals to attdisbursion(Bruce)
"intro/unix" manual page now "pgintro"(Bruce)
"built-in" manual page now "pgbuiltin"(Bruce)
"drop" manual page now "drop_table"(Bruce)
Add "create_trigger", "drop_trigger" manual pages(Thomas)
Add constraints regression test(Vadim & Thomas)
Add comments syntax regression test(Thomas)
Add PGINDENT and support program(Bruce)
Massive commit to run PGINDENT on all *.c and *.h files(Bruce)
Files moved to /src/tools directory(Bruce)
SPI and Trigger programming guides (Vadim & D’Arcy)
```

Release 6.1.1

Migration from v6.1 to v6.1.1

This is a minor bug-fix release. A dump/reload is not required from v6.1, but is required from any release prior to v6.1. Refer to the release notes for v6.1 for more details.

Detailed Change List

```
Changes in this release
-----------------------
fix for SET with options (Thomas)
allow pg_dump/pg_dumpall to preserve ownership of all
tables/objects(Bruce)
new psql \connect option allows changing usernames without changing
databases
fix for initdb --debug option(Yoshihiko Ichikawa))
lextest cleanup(Bruce)
hash fixes(Vadim)
fix date/time month boundary arithmetic(Thomas)
fix timezone daylight handling for some ports(Thomas, Bruce, Tatsuo)
timestamp overhauled to use standard functions(Thomas)
other code cleanup in date/time routines(Thomas)
psql’s \d now case-insensitive(Bruce)
psql’s backslash commands can now have trailing semicolon(Bruce)
fix memory leak in psql when using \g(Bruce)
```

```
major fix for endian handling of communication to server(Thomas,
Tatsuo)
Fix for Solaris assembler and include files(Yoshihiko Ichikawa)
allow underscores in usernames(Bruce)
pg_dumpall now returns proper status, portability fix(Bruce)
```

Release 6.1

The regression tests have been adapted and extensively modified for the v6.1 release of Postgres.

Three new data types (datetime, timespan, and circle) have been added to the native set of Postgres types. Points, boxes, paths, and polygons have had their output formats made consistant across the data types. The polygon output in misc.out has only been spot-checked for correctness relative to the original regression output.

Postgres v6.1 introduces a new, alternate optimizer which uses *genetic* algorithms. These algorithms introduce a random behavior in the ordering of query results when the query contains multiple qualifiers or multiple tables (giving the optimizer a choice on order of evaluation). Several regression tests have been modified to explicitly order the results, and hence are insensitive to optimizer choices. A few regression tests are for data types which are inherently unordered (e.g. points and time intervals) and tests involving those types are explicitly bracketed with **set geqo to ’off’** and **reset geqo**.

The interpretation of array specifiers (the curly braces around atomic values) appears to have changed sometime after the original regression tests were generated. The current `./expected/*.out` files reflect this new interpretation, which may not be correct!

The float8 regression test fails on at least some platforms. This is due to differences in implementations of pow() and exp() and the signaling mechanisms used for overflow and underflow conditions.

The "random" results in the random test should cause the "random" test to be "failed", since the regression tests are evaluated using a simple diff. However, "random" does not seem to produce random results on my test machine (Linux/gcc/i686).

Migration to v6.1

This migration requires a complete dump of the 6.0 database and a restore of the database in 6.1.

Those migrating from earlier 1.* releases should first upgrade to 1.09 because the COPY output format was improved from the 1.02 release.

Detailed Change List

```
Bug Fixes
---------
packet length checking in library routines
lock manager priority patch
check for under/over flow of float8(Bruce)
multi-table join fix(Vadim)
SIGPIPE crash fix(Darren)
large object fixes(Sven)
allow btree indexes to handle NULLs(Vadim)
timezone fixes(D’Arcy)
```

```
select SUM(x) can return NULL on no rows(Thomas)
internal optimizer, executor bug fixes(Vadim)
fix problem where inner loop in < or <= has no rows(Vadim)
prevent re-commuting join index clauses(Vadim)
fix join clauses for multiple tables(Vadim)
fix hash, hashjoin for arrays(Vadim)
fix btree for abstime type(Vadim)
large object fixes(Raymond)
fix buffer leak in hash indices (Vadim)
fix rtree for use in inner scan (Vadim)
fix gist for use in inner scan, cleanups (Vadim, Andrea)
avoid unnecessary local buffers allocation (Vadim, Massimo)
fix local buffers leak in transaction aborts (Vadim)
fix file manager memmory leaks, cleanups (Vadim, Massimo)
fix storage manager memmory leaks (Vadim)
fix btree duplicates handling (Vadim)
fix deleted tuples re-incarnation caused by vacuum (Vadim)
fix SELECT varchar()/char() INTO TABLE made zero-length
fields(Bruce)
many psql, pg_dump, and libpq memory leaks fixed using Purify (Igor)

Enhancements
------------
attribute optimization statistics(Bruce)
much faster new btree bulk load code(Paul)
BTREE UNIQUE added to bulk load code(Vadim)
new lock debug code(Massimo)
massive changes to libpg++(Leo)
new GEQO optimizer speeds table multi-table optimization(Martin)
new WARN message for non-unique insert into unique key(Marc)
update x=-3, no spaces, now valid(Bruce)
remove case-sensitive identifier handling(Bruce,Thomas,Dan)
debug backend now pretty-prints tree(Darren)
new Oracle character functions(Edmund)
new plaintext password functions(Dan)
no such class or insufficient privilege changed to distinct
messages(Dan)
new ANSI timestamp function(Dan)
new ANSI Time and Date types (Thomas)
move large chunks of data in backend(Martin)
multi-column btree indexes(Vadim)
new SET var TO value command(Martin)
update transaction status on reads(Dan)
new locale settings for character types(Oleg)
new SEQUENCE serial number generator(Vadim)
GROUP BY function now possible(Vadim)
re-organize regression test(Thomas,Marc)
new optimizer operation weights(Vadim)
new psql \z grant/permit option(Marc)
new MONEY data type(D’Arcy,Thomas)
tcp socket communication speed improved(Vadim)
new VACUUM option for attribute statistics, and for certain columns
(Vadim)
many geometric type improvements(Thomas,Keith)
additional regression tests(Thomas)
new datestyle variable(Thomas,Vadim,Martin)
more comparison operators for sorting types(Thomas)
```

```
new conversion functions(Thomas)
new more compact btree format(Vadim)
allow pg_dumpall to preserve database ownership(Bruce)
new SET GEQO=# and R_PLANS variable(Vadim)
old (!GEQO) optimizer can use right-sided plans (Vadim)
typechecking improvement in SQL parser(Bruce)
new SET, SHOW, RESET commands(Thomas,Vadim)
new \connect database USER option
new destroydb -i option (Igor)
new \dt and \di psql commands (Darren)
SELECT "\n" now escapes newline (A. Duursma)
new geometry conversion functions from old format (Thomas)

Source tree changes
-------------------
new configuration script(Marc)
readline configuration option added(Marc)
OS-specific configuration options removed(Marc)
new OS-specific template files(Marc)
no more need to edit Makefile.global(Marc)
re-arrange include files(Marc)
nextstep patches (Gregor Hoffleit)
removed WIN32-specific code(Bruce)
removed postmaster -e option, now only postgres -e option (Bruce)
merge duplicate library code in front/backends(Martin)
now works with eBones, international Kerberos(Jun)
more shared library support
c++ include file cleanup(Bruce)
warn about buggy flex(Bruce)
DG-UX, Ultrix, Irix, AIX portability fixes
```

Release v6.0

A dump/restore is required for those wishing to migrate data from previous releases of Postgres.

Migration from v1.09 to v6.0

This migration requires a complete dump of the 1.09 database and a restore of the database in 6.0.

Migration from pre-v1.09 to v6.0

Those migrating from earlier 1.* releases should first upgrade to 1.09 because the COPY output format was improved from the 1.02 release.

Detailed Change List

```
Bug Fixes
---------
ALTER TABLE bug - running postgress process needs to re-read table
definition
Allow vacuum to be run on one table or entire database(Bruce)
Array fixes
Fix array over-runs of memory writes(Kurt)
Fix elusive btree range/non-range bug(Dan)
Fix for hash indexes on some types like time and date
Fix for pg_log size explosion
Fix permissions on lo_export()(Bruce)
Fix unitialized reads of memory(Kurt)
Fixed ALTER TABLE ... char(3) bug(Bruce)
Fixed a few small memory leaks
Fixed EXPLAIN handling of options and changed full_path option name
Fixed output of group acl permissions
Memory leaks (hunt and destroy with tools like Purify(Kurt)
Minor improvements to rules system
NOTIFY fixes
New asserts for run-checking
Overhauled parser/analyze code to properly report errors and
increase speed
Pg_dump -d now handles NULL’s properly(Bruce)
Prevent SELECT NULL from crashing server (Bruce)
Properly report errors when INSERT ... SELECT columns did not match
Properly report errors when insert column names were not correct
Psql \g filename now works(Bruce)
Psql fixed problem with multiple statements on one line with
multiple outputs
Removed duplicate system oid’s
SELECT * INTO TABLE . GROUP/ORDER BY gives unlink error if table
exists(Bruce)
Several fixes for queries that crashed the backend
Starting quote in insert string errors(Bruce)
Submitting an empty query now returns empty status, not just " "
query(Bruce)

Enhancements
------------
Add EXPLAIN manual page(Bruce)
Add UNIQUE index capability(Dan)
Add hostname/user level access control rather than just hostname and
user
Add synonym of != for <>(Bruce)
Allow "select oid,* from table"
Allow BY,ORDER BY to specify columns by number, or by non-alias
table.column(Bruce)
Allow COPY from the frontend(Bryan)
Allow GROUP BY to use alias column name(Bruce)
Allow actual compression, not just reuse on the same page(Vadim)
Allow installation-configuration option to auto-add all local
users(Bryan)
Allow libpq to distinguish between text value ’’ and null(Bruce)
Allow non-postgres users with createdb privs to destroydb’s
Allow restriction on who can create C functions(Bryan)
Allow restriction on who can do backend COPY(Bryan)
Can shrink tables, pg_time and pg_log(Vadim & Erich)
Change debug level 2 to print queries only, changed debug heading
layout(Bruce)
Change default decimal constant representation from float4 to
float8(Bruce)
European date format now set when postmaster is started
Execute lowercase function names if not found with exact case
```

```
Fixes for aggregate/GROUP processing, allow 'select
sum(func(x),sum(x+y) from z'
Gist now included in the distrubution(Marc)
Idend authentication of local users(Bryan)
Implement BETWEEN qualifier(Bruce)
Implement IN qualifier(Bruce)
Libpq has PQgetisnull()(Bruce)
Libpq++ improvements
New options to initdb(Bryan)
Pg_dump allow dump of oid's(Bruce)
Pg_dump create indexes after tables are loaded for speed(Bruce)
Pg_dumpall dumps all databases, and the user table
Pginterface additions for NULL values(Bruce)
Prevent postmaster from being run as root
Psql \h and \? is now readable(Bruce)
Psql allow backslashed, semicolons anywhere on the line(Bruce)
Psql changed command prompt for lines in query or in quotes(Bruce)
Psql char(3) now displays as (bp)char in \d output(Bruce)
Psql return code now more accurate(Bryan?)
Psql updated help syntax(Bruce)
Re-visit and fix vacuum(Vadim)
Reduce size of regression diffs, remove timezone name
difference(Bruce)
Remove compile-time parameters to enable binary distributions(Bryan)
Reverse meaning of HBA masks(Bryan)
Secure Authentication of local users(Bryan)
Speed up vacuum(Vadim)
Vacuum now had VERBOSE option(Bruce)

Source tree changes
-------------------
All functions now have prototypes that are compared against the
calls
Allow asserts to be disabled easly from Makefile.global(Bruce)
Change oid constants used in code to #define names
Decoupled sparc and solaris defines(Kurt)
Gcc -Wall compiles cleanly with warnings only from unfixable
constructs
Major include file reorganization/reduction(Marc)
Make now stops on compile failure(Bryan)
Makefile restructuring(Bryan, Marc)
Merge bsdi_2_1 to bsdi(Bruce)
Monitor program removed
Name change from Postgres95 to PostgreSQL
New config.h file(Marc, Bryan)
PG_VERSION now set to 6.0 and used by postmaster
Portability additions, including Ultrix, DG/UX, AIX, and Solaris
Reduced the number of #define's, centeralized #define's
Remove duplicate OIDS in system tables(Dan)
Remove duplicate system catalog info or report mismatches(Dan)
Removed many os-specific #define's
Restructured object file generation/location(Bryan, Marc)
Restructured port-specific file locations(Bryan, Marc)
Unused/uninialized variables corrected
```

Release v1.09

Sorry, we stopped keeping track of changes from 1.02 to 1.09. Some of the changes listed in 6.0 were actually included in the 1.02.1 to 1.09 releases.

Release v1.02

Migration from v1.02 to v1.02.1

Here is a new migration file for 1.02.1. It includes the ’copy’ change and a script to convert old ascii files.

> **Note:** The following notes are for the benefit of users who want to migrate databases from postgres95 1.01 and 1.02 to postgres95 1.02.1.
>
> If you are starting afresh with postgres95 1.02.1 and do not need to migrate old databases, you do not need to read any further.

In order to upgrade older postgres95 version 1.01 or 1.02 databases to version 1.02.1, the following steps are required:

1. Start up a new 1.02.1 postmaster
2. Add the new built-in functions and operators of 1.02.1 to 1.01 or 1.02 databases. This is done by running the new 1.02.1 server against your own 1.01 or 1.02 database and applying the queries attached at the end of thie file. This can be done easily through psql. If your 1.01 or 1.02 database is named "testdb" and you have cut the commands from the end of this file and saved them in addfunc.sql:

   ```
   % psql testdb -f addfunc.sql
   ```

   Those upgrading 1.02 databases will get a warning when executing the last two statements in the file because they are already present in 1.02. This is not a cause for concern.

Dump/Reload Procedure

If you are trying to reload a pg_dump or text-mode ’copy tablename to stdout’ generated with a previous version, you will need to run the attached sed script on the ASCII file before loading it into the database. The old format used ’.’ as end-of-data, while ’\.’ is now the end-of-data marker. Also, empty strings are now loaded in as ’’ rather than NULL. See the copy manual page for full details.

```
sed ’s/^\.$/\\./g’ <in_file >out_file
```

If you are loading an older binary copy or non-stdout copy, there is no end-of-data character, and hence no conversion necessary.

```
-- following lines added by agc to reflect the case-insensitive
-- regexp searching for varchar (in 1.02), and bpchar (in 1.02.1)
create operator ~* (leftarg = bpchar, rightarg = text, procedure =
texticregexeq);
create operator !~* (leftarg = bpchar, rightarg = text, procedure =
texticregexne);
```

```
create operator ~* (leftarg = varchar, rightarg = text, procedure =
texticregexeq);
create operator !~* (leftarg = varchar, rightarg = text, procedure =
texticregexne);
```

Detailed Change List

```
Source code maintenance and development
 * worldwide team of volunteers
 * the source tree now in CVS at ftp.ki.net

Enhancements
 * psql (and underlying libpq library) now has many more options for
   formatting output, including HTML
 * pg_dump now output the schema and/or the data, with many fixes to
   enhance completeness.
 * psql used in place of monitor in administration shell scripts.
   monitor to be depreciated in next release.
 * date/time functions enhanced
 * NULL insert/update/comparison fixed/enhanced
 * TCL/TK lib and shell fixed to work with both tck7.4/tk4.0 and
tcl7.5/tk4.1

Bug Fixes (almost too numerous to mention)
 * indexes
 * storage management
 * check for NULL pointer before dereferencing
 * Makefile fixes

New Ports
 * added SolarisX86 port
 * added BSDI 2.1 port
 * added DGUX port
```

Release v1.01

Migration from v1.0 to v1.01

The following notes are for the benefit of users who want to migrate databases from postgres95 1.0 to postgres95 1.01.

If you are starting afresh with postgres95 1.01 and do not need to migrate old databases, you do not need to read any further.

In order to postgres95 version 1.01 with databases created with postgres95 version 1.0, the following steps are required:

1. Set the definition of NAMEDATALEN in src/Makefile.global to 16 and OIDNAMELEN to 20.
2. Decide whether you want to use Host based authentication.
   a. If you do, you must create a file name "pg_hba" in your top-level data directory (typically the value of your $PGDATA). src/libpq/pg_hba shows an example syntax.
   b. If you do not want host-based authentication, you can comment out the line

```
HBA = 1
```

in src/Makefile.global

Note that host-based authentication is turned on by default, and if you do not take steps A or B above, the out-of-the-box 1.01 will not allow you to connect to 1.0 databases.

3. Compile and install 1.01, but DO NOT do the initdb step.
4. Before doing anything else, terminate your 1.0 postmaster, and backup your existing $PGDATA directory.
5. Set your PGDATA environment variable to your 1.0 databases, but set up path up so that 1.01 binaries are being used.
6. Modify the file $PGDATA/PG_VERSION from 5.0 to 5.1
7. Start up a new 1.01 postmaster
8. Add the new built-in functions and operators of 1.01 to 1.0 databases. This is done by running the new 1.01 server against your own 1.0 database and applying the queries attached and saving in the file 1.0_to_1.01.sql. This can be done easily through psql. If your 1.0 database is name "testdb":

   ```
   % psql testdb -f 1.0_to_1.01.sql
   ```

   and then execute the following commands (cut and paste from here):

   ```
   -- add builtin functions that are new to 1.01

   create function int4eqoid (int4, oid) returns bool as ’foo’
   language ’internal’;
   create function oideqint4 (oid, int4) returns bool as ’foo’
   language ’internal’;
   create function char2icregexeq (char2, text) returns bool as
   ’foo’
   language ’internal’;
   create function char2icregexne (char2, text) returns bool as
   ’foo’
   language ’internal’;
   create function char4icregexeq (char4, text) returns bool as
   ’foo’
   language ’internal’;
   create function char4icregexne (char4, text) returns bool as
   ’foo’
   language ’internal’;
   create function char8icregexeq (char8, text) returns bool as
   ’foo’
   language ’internal’;
   create function char8icregexne (char8, text) returns bool as
   ’foo’
   language ’internal’;
   create function char16icregexeq (char16, text) returns bool as
   ’foo’
   language ’internal’;
   create function char16icregexne (char16, text) returns bool as
   ’foo’
   language ’internal’;
   create function texticregexeq (text, text) returns bool as ’foo’
   language ’internal’;
   ```

```
create function texticregexne (text, text) returns bool as 'foo'
language 'internal';

-- add builtin functions that are new to 1.01

create operator = (leftarg = int4, rightarg = oid, procedure =
int4eqoid);
create operator = (leftarg = oid, rightarg = int4, procedure =
oideqint4);
create operator ~* (leftarg = char2, rightarg = text, procedure
= char2icregexeq);
create operator !~* (leftarg = char2, rightarg = text, procedure
= char2icregexne);
create operator ~* (leftarg = char4, rightarg = text, procedure
= char4icregexeq);
create operator !~* (leftarg = char4, rightarg = text, procedure
= char4icregexne);
create operator ~* (leftarg = char8, rightarg = text, procedure
= char8icregexeq);
create operator !~* (leftarg = char8, rightarg = text, procedure
= char8icregexne);
create operator ~* (leftarg = char16, rightarg = text, procedure
= char16icregexeq);
create operator !~* (leftarg = char16, rightarg = text,
procedure = char16icregexne);
create operator ~* (leftarg = text, rightarg = text, procedure =
texticregexeq);
create operator !~* (leftarg = text, rightarg = text, procedure
= texticregexne);
```

Detailed Change List

```
Incompatibilities:
 * 1.01 is backwards compatible with 1.0 database provided the user
   follow the steps outlined in the MIGRATION_from_1.0_to_1.01 file.
   If those steps are not taken, 1.01 is not compatible with 1.0
database.

Enhancements:
 * added PQdisplayTuples() to libpq and changed monitor and psql to
use it
 * added NeXT port (requires SysVIPC implementation)
 * added CAST .. AS ... syntax
 * added ASC and DESC keywords
 * added 'internal' as a possible language for CREATE FUNCTION
   internal functions are C functions which have been statically
linked
   into the postgres backend.
 * a new type "name" has been added for system identifiers (table
names,
   attribute names, etc.)  This replaces the old char16 type.   The
   of name is set by the NAMEDATALEN #define in src/Makefile.global
 * a readable reference manual that describes the query language.
 * added host-based access control.  A configuration file
($PGDATA/pg_hba)
   is used to hold the configuration data.  If host-based access
control
```

```
   is not desired, comment out HBA=1 in src/Makefile.global.
 * changed regex handling to be uniform use of Henry Spencer’s regex
code
   regardless of platform.  The regex code is included in the
distribution
 * added functions and operators for case-insensitive regular
expressions.
   The operators are ~* and !~*.
 * pg_dump uses COPY instead of SELECT loop for better performance

Bug fixes:
 * fixed an optimizer bug that was causing core dumps when
   functions calls were used in comparisons in the WHERE clause
 * changed all uses of getuid to geteuid so that effective uids are
used
 * psql now returns non-zero status on errors when using -c
 * applied public patches 1-14
```

Release v1.0

Detailed Change List

```
Copyright change:
 * The copyright of Postgres 1.0 has been loosened to be freely
modifiable
   and modifiable for any purpose.  Please read the COPYRIGHT file.
   Thanks to Professor Michael Stonebraker for making this possible.

Incompatibilities:
 *  date formats have to be MM-DD-YYYY (or DD-MM-YYYY if you’re
using
   EUROPEAN STYLE).  This follows SQL-92 specs.
 *  "delimiters" is now a keyword

Enhancements:
 *  sql LIKE syntax has been added
 *  copy command now takes an optional USING DELIMITER
specification.
   delimiters can be any single-character string.
 *  IRIX 5.3 port has been added.
   Thanks to Paul Walmsley and others.
 *  updated pg_dump to work with new libpq
 *  \d has been added psql
   Thanks to Keith Parks
 *  regexp performance for architectures that use POSIX regex has
been
   improved due to caching of precompiled patterns.
   Thanks to Alistair Crooks
 *  a new version of libpq++
   Thanks to William Wanders

Bug fixes:
 *  arbitrary userids can be specified in the createuser script
 *  \c to connect to other databases in psql now works.
 *  bad pg_proc entry for float4inc() is fixed
 *  users with usecreatedb field set can now create databases
without
   having to be usesuper
 *  remove access control entries when the entry no longer has any
   permissions
 *  fixed non-portable datetimes implementation
 *  added kerberos flags to the src/backend/Makefile
 *  libpq now works with kerberos
 *  typographic errors in the user manual have been corrected.
 *  btrees with multiple index never worked, now we tell you they
don’t
   work when you try to use them
```

Postgres95 Beta 0.03

Detailed Change List

```
Incompatible changes:
 * BETA-0.3 IS INCOMPATIBLE WITH DATABASES CREATED WITH PREVIOUS
VERSIONS
   (due to system catalog changes and indexing structure changes).
 * double-quote (") is deprecated as a quoting character for string
literals;
   you need to convert them to single quotes (’).
 * name of aggregates (eg. int4sum) are renamed in accordance with
the
   SQL standard (eg. sum).
 * CHANGE ACL syntax is replaced by GRANT/REVOKE syntax.
 * float literals (eg. 3.14) are now of type float4 (instead of
float8 in
   previous releases); you might have to do typecasting if you
depend on it
   being of type float8.  If you neglect to do the typecasting and
you assign
   a float literal to a field of type float8, you may get incorrect
values
   stored!
 * LIBPQ has been totally revamped so that frontend applications
   can connect to multiple backends
 * the usesysid field in pg_user has been changed from int2 to int4
to
   allow wider range of Unix user ids.
 * the netbsd/freebsd/bsd o/s ports have been consolidated into a
   single BSD44_derived port.  (thanks to Alistair Crooks)

SQL standard-compliance (the following details changes that makes
postgres95
more compliant to the SQL-92 standard):
 * the following SQL types are now built-in: smallint, int(eger),
float, real,
   char(N), varchar(N), date and time.

   The following are aliases to existing postgres types:
                smallint -> int2
                integer, int -> int4
                float, real  -> float4
   char(N) and varchar(N) are implemented as truncated text types.
In
   addition, char(N) does blank-padding.
 * single-quote (’) is used for quoting string literals; ’’ (in
addition to
   \’) is supported as means of inserting a single quote in a string
 * SQL standard aggregate names (MAX, MIN, AVG, SUM, COUNT) are used
   (Also, aggregates can now be overloaded, i.e. you can define your
   own MAX aggregate to take in a user-defined type.)
 * CHANGE ACL removed. GRANT/REVOKE syntax added.
   - Privileges can be given to a group using the "GROUP" keyword.
        For example:
                GRANT SELECT ON foobar TO GROUP my_group;
        The keyword ’PUBLIC’ is also supported to mean all users.

        Privileges can only be granted or revoked to one user or
group
        at a time.

        "WITH GRANT OPTION" is not supported.  Only class owners can
change
        access control
   - The default access control is to to grant users readonly
access.
     You must explicitly grant insert/update access to users.  To
change
     this, modify the line in
                src/backend/utils/acl.h
     that defines ACL_WORLD_DEFAULT

Bug fixes:
 * the bug where aggregates of empty tables were not run has been
fixed. Now,
   aggregates run on empty tables will return the initial conditions
of the
   aggregates. Thus, COUNT of an empty  table will now properly
return 0.
   MAX/MIN of an empty table will return a tuple of value NULL.
 * allow the use of \; inside the monitor
 * the LISTEN/NOTIFY asynchronous notification mechanism now work
 * NOTIFY in rule action bodies now work
 * hash indices work, and access methods in general should perform
better.
   creation of large btree indices should be much faster.  (thanks
to Paul
   Aoki)

Other changes and enhancements:
 * addition of an EXPLAIN statement used for explaining the query
execution
   plan (eg. "EXPLAIN SELECT * FROM EMP" prints out the execution
plan for
   the query).
 * WARN and NOTICE messages no longer have timestamps on them. To
turn on
   timestamps of error messages, uncomment the line in
   src/backend/utils/elog.h:
        /* define ELOG_TIMESTAMPS */
 * On an access control violation, the message
        "Either no such class or insufficient privilege"
   will be given.  This is the same message that is returned when
   a class is not found.  This dissuades non-privileged users from
   guessing the existence of privileged classes.
 * some additional system catalog changes have been made that are
not
   visible to the user.

libpgtcl changes:
 * The -oid option has been added to the "pg_result" tcl command.
   pg_result -oid returns oid of the last tuple inserted.   If the
   last command was not an INSERT, then pg_result -oid returns "".
 * the large object interface is available as pg_lo* tcl commands:
   pg_lo_open, pg_lo_close, pg_lo_creat, etc.

Portability enhancements and New Ports:
 * flex/lex problems have been cleared up.  Now, you should be able
to use
   flex instead of lex on any platforms.  We no longer make
assumptions of
   what lexer you use based on the platform you use.
 * The Linux-ELF port is now supported.  Various configuration have
been
   tested:  The following configuration is known to work:
        kernel 1.2.10, gcc 2.6.3, libc 4.7.2, flex 2.5.2, bison 1.24
   with everything in ELF format,

New utilities:
 * ipcclean added to the distribution
   ipcclean usually does not need to be run, but if your backend
crashes
   and leaves shared memory segments hanging around, ipcclean will
   clean them up for you.

New documentation:
 * the user manual has been revised and libpq documentation added.
```

Postgres95 Beta 0.02

Detailed Change List

```
Incompatible changes:
 * The SQL statement for creating a database is ’CREATE DATABASE’
instead
   of ’CREATEDB’. Similarly, dropping a database is ’DROP DATABASE’
instead
   of ’DESTROYDB’. However, the names of the executables ’createdb’
and
   ’destroydb’ remain the same.

New tools:
 * pgperl - a Perl (4.036) interface to Postgres95
 * pg_dump - a utility for dumping out a postgres database into a
```

```
        script file containing query commands. The script files are
in a ASCII
        format and can be used to reconstruct the database, even on
other
        machines and other architectures. (Also good for converting
        a Postgres 4.2 database to Postgres95 database.)

The following ports have been incorporated into
postgres95-beta-0.02:
 * the NetBSD port by Alistair Crooks
 * the AIX port by Mike Tung
 * the Windows NT port by Jon Forrest (more stuff but not done yet)
 * the Linux ELF port by Brian Gallew

The following bugs have been fixed in postgres95-beta-0.02:
 * new lines not escaped in COPY OUT and problem with COPY OUT when
first
   attribute is a '.'
 * cannot type return to use the default user id in createuser
 * SELECT DISTINCT on big tables crashes
 * Linux installation problems
 * monitor doesn't allow use of 'localhost' as PGHOST
 * psql core dumps when doing \c or \l
 * the "pgtclsh" target missing from src/bin/pgtclsh/Makefile
 * libpgtcl has a hard-wired default port number
 * SELECT DISTINCT INTO TABLE hangs
 * CREATE TYPE doesn't accept 'variable' as the internallength
 * wrong result using more than 1 aggregate in a SELECT
```

Postgres95 Beta 0.01

Initial release.

Timing Results

These timing results are from running the regression test with the commands

```
% cd src/test/regress
% make all
% time make runtest
```

Timing under Linux 2.0.27 seems to have a roughly 5% variation from run to run, presumably due to the scheduling vagaries of multitasking systems.

v6.5

As has been the case for previous releases, timing between releases is not directly comparable since new regression tests have been added. In general, v6.5 is faster than previous releases.

Timing with `fsync()` disabled:

```
  Time   System
  02:00  Dual Pentium Pro 180, 224MB, UW-SCSI, Linux 2.0.36, gcc
2.7.2.3 -O2 -m486
  04:38  Sparc Ultra 1 143MHz, 64MB, Solaris 2.6
```

Timing with `fsync()` enabled:

```
  Time   System
  04:21  Dual Pentium Pro 180, 224MB, UW-SCSI, Linux 2.0.36, gcc
2.7.2.3 -O2 -m486
```

For the linux system above, using UW-SCSI disks rather than (older) IDE disks leads to a 50% improvement in speed on the regression test.

v6.4beta

The times for this release are not directly comparable to those for previous releases since some additional regression tests have been included. In general, however, v6.4 should be slightly faster than the previous release (thanks, Bruce!).

```
  Time   System
  02:26  Dual Pentium Pro 180, 96MB, UW-SCSI, Linux 2.0.30, gcc
2.7.2.1 -O2 -m486
```

v6.3

The times for this release are not directly comparable to those for previous releases since some additional regression tests have been included and some obsolete tests involving time travel have been removed. In general, however, v6.3 is substantially faster than previous releases (thanks, Bruce!).

```
  Time   System
  02:30  Dual Pentium Pro 180, 96MB, UW-SCSI, Linux 2.0.30, gcc
2.7.2.1 -O2 -m486
  04:12  Dual Pentium Pro 180, 96MB, EIDE, Linux 2.0.30, gcc 2.7.2.1
-O2 -m486
```

v6.1

```
  Time   System
  06:12  Pentium Pro 180, 32MB, EIDE, Linux 2.0.30, gcc 2.7.2 -O2
-m486
  12:06  P-100, 48MB, Linux 2.0.29, gcc
  39:58  Sparc IPC 32MB, Solaris 2.5, gcc 2.7.2.1 -O -g
```

Bibliography

Selected references and readings for SQL and Postgres.

Some white papers and technical reports from the original Postgres development team are available at the University of California, Berkeley, Computer Science Department web site (http://s2k-ftp.CS.Berkeley.EDU:8000/postgres/papers/)

SQL Reference Books

The Practical SQL Handbook , *Using Structured Query Language* , Judith Bowman, Sandra Emerson, and Marcy Darnovsky, 0-201-44787-8, 1996, Addison-Wesley, 1996.

A Guide to the SQL Standard , *A user's guide to the standard database language SQL* , C. J. Date and Hugh Darwen, 0-201-96426-0, 1997, Addison-Wesley, 1997.

An Introduction to Database Systems , C. J. Date, 1994, Addison-Wesley, 1994.

Understanding the New SQL , *A complete guide*, Jim Melton and Alan R. Simon, 1-55860-245-3, 1993, Morgan Kaufmann, 1993.

Abstract

Accessible reference for SQL features.

Principles of Database and Knowledge : Base Systems , Jeffrey D. Ullman, Computer Science Press , 1988 .

PostgreSQL-Specific Documentation

The PostgreSQL Administrator's Guide , Edited by Thomas Lockhart, 2000-05-01, The PostgreSQL Global Development Group.

The PostgreSQL Developer's Guide , Edited by Thomas Lockhart, 2000-05-01, The PostgreSQL Global Development Group.

The PostgreSQL Programmer's Guide , Edited by Thomas Lockhart, 2000-05-01, The PostgreSQL Global Development Group.

The PostgreSQL Tutorial Introduction , Edited by Thomas Lockhart, 2000-05-01, The PostgreSQL Global Development Group.

The PostgreSQL User's Guide , Edited by Thomas Lockhart, 2000-05-01, The PostgreSQL Global Development Group.

Enhancement of the ANSI SQL Implementation of PostgreSQL , Stefan Simkovics, O.Univ.Prof.Dr.. Georg Gottlob, November 29, 1998, Department of Information Systems, Vienna University of Technology .

Discusses SQL history and syntax, and describes the addition of INTERSECT and EXCEPT constructs into Postgres. Prepared as a Master's Thesis with the support of O.Univ.Prof.Dr. Georg Gottlob and Univ.Ass. Mag. Katrin Seyr at Vienna University of Technology.

The Postgres95 User Manual , A. Yu and J. Chen, The POSTGRES Group , Sept. 5, 1995, University of California, Berkeley CA.

Proceedings and Articles

Partial indexing in POSTGRES: research project , Nels Olson, 1993, UCB Engin T7.49.1993 O676, University of California, Berkeley CA.

A Unified Framework for Version Modeling Using Production Rules in a Database System , L. Ong and J. Goh, April, 1990, ERL Technical Memorandum M90/33, University of California, Berkeley CA.

The Postgres Data Model , L. Rowe and M. Stonebraker, Sept. 1987, VLDB Conference, Brighton, England, 1987.

Generalized partial indexes (http://simon.cs.cornell.edu/home/praveen/papers/partindex.de95.ps.Z) , P. Seshadri and A. Swami, March 1995, Eleventh International Conference on Data Engineering, 1995, Cat. No.95CH35724, IEEE Computer Society Press.

The Design of Postgres , M. Stonebraker and L. Rowe, May 1986, Conference on Management of Data, Washington DC, ACM-SIGMOD, 1986.

The Design of the Postgres Rules System, M. Stonebraker, E. Hanson, and C. H. Hong, Feb. 1987, Conference on Data Engineering, Los Angeles, CA, IEEE, 1987.

The Postgres Storage System , M. Stonebraker, Sept. 1987, VLDB Conference, Brighton, England, 1987.

A Commentary on the Postgres Rules System , M. Stonebraker, M. Hearst, and S. Potamianos, Sept. 1989, Record 18(3), SIGMOD, 1989.

The case for partial indexes (DBMS) (http://s2k-ftp.CS.Berkeley.EDU:8000/postgres/papers/ERL-M89-17.pdf) , M. Stonebraker, Dec. 1989, Record 18(no.4):4-11, SIGMOD, 1989.

The Implementation of Postgres , M. Stonebraker, L. A. Rowe, and M. Hirohama, March 1990, Transactions on Knowledge and Data Engineering 2(1), IEEE.

On Rules, Procedures, Caching and Views in Database Systems , M. Stonebraker and et al, June 1990, Conference on Management of Data, ACM-SIGMOD.