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

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 a public-domain, 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 has 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 Postgres Questions (mailto:questions@postgresql.org) mailing list is a good place to have user questions answered. Other mailing lists are available; consult the web page 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 Postgres Documentation (mailto:docs@postgresql.org) 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 v6.5) 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).

Y2K Statement

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

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 6.5 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	v6.5	1999-05-26	(Andreas Zeugswetter (mailto:Andreas.Zeugswetter@telecom.at))
BSDI	x86	v6.5	1999-05-25	(Bruce Momjian (mailto:maillist@candle.pha.pa.us)
FreeBSD 2.2.x-4.0	x86	v6.5	1999-05-25	(Tatsuo Ishii (mailto:t-ishii@sra.co.jp), Marc Fournier (mailto:scrappy@hub.org))
DGUX 5.4R4.11	m88k	v6.3	1998-03-01	v6.4 probably OK. Needs new maintainer. (Brian E Gallew (mailto:geek+@cmu.edu))
Digital Unix 4.0	Alpha	v6.4	1998-10-29	Minor patchable problems (Pedro J. Lobo (mailto:pjlobo@euitt.upm.es))
HPUX	PA-RISC	v6.4	1998-10-25	Both 9.0x and 10.20 (Tom Lane (mailto:tgl@sss.pgh.pa.us), Stan Brown (mailto:stanb@awod.com))
IRIX 6.5	MIPS	v6.4	1998-12-29	IRIX 5.x is different (Mark Dalphin (mdalphin@amgen.com))

linux 2.0.x	Alpha	v6.3.2	1998-04-16	Mostly successful. Needs work for v6.4. (Ryan Kirkpatrick (mailto:rkirkpat@nag.cs.col-orado.edu))
linux 2.0.x/libc5	x86	v6.4	1998-10-27	(Thomas Lockhart (mailto:lockhart@alumni.ca-ltech.edu))
linux 2.0.x/glibc2	x86	v6.5	1999-05-24	(Thomas Lockhart (mailto:lockhart@alumni.ca-ltech.edu))
linux 2.0.x	MIPS	v6.4	1998-12-16	Cobalt Qube (Tatsuo Ishii (mailto:t-ishii@sra.co.jp))
linux 2.0.x	Sparc	v6.4	1998-10-25	(Tom Szybist (mailto:szybist@boxhill.co-m))
linuxPPC 2.1.24	PPC603e	v6.4	1998-10-26	Powerbook 2400c (Tatsuo Ishii (mailto:t-ishii@sra.co.jp))
mklinux DR3	PPC750	v6.4	1998-09-16	PowerMac 7600 (Tatsuo Ishii (mailto:t-ishii@sra.co.jp))
NetBSD	arm32	v6.5	1999-04-14	(Andrew McMurry (mailto:a.mcmurry1@physi-cs.oxford.ac.uk))
NetBSD/i386 1.3.2	x86	v6.4	1998-10-25	(Brook Milligan (mailto:brook@trillium.NM-SU.Edu))
NetBSD	m68k	v6.4.2	1998-12-28	Mac SE/30 (Mr. Mutsuki Nakajima, Tatsuo Ishii (mailto:t-ishii@sra.co.jp))
NetBSD-current	NS32532	v6.4	1998-10-27	small problems in date/time math (Jon Buller (mailto:jonb@metronet.com-))
NetBSD/sparc 1.3H	Sparc	v6.4	1998-10-27	(Tom I Helbekkmo (mailto:tih@hamartun.priv.n-o))
NetBSD 1.3	VAX	v6.3	1998-03-01	(Tom I Helbekkmo (mailto:tih@hamartun.priv.n-o))

SCO OpenServer 5	x86	v6.5	1999-05-25	(Andrew Merrill (mailto:andrew@compclass.-com))
SCO UnixWare 7	x86	v6.5	1999-05-25	(Andrew Merrill (mailto:andrew@compclass.-com))
Solaris	x86	v6.4	1998-10-28	(Marc Fournier (mailto:scrappy@hub.org))
Solaris 2.6-2.7	Sparc	v6.4	1998-10-28	(Tom Szybist (mailto:szybist@boxhill.co-m), Frank Ridderbusch (mailto:ridderbusch.pad@sn-i.de))
SunOS 4.1.4	Sparc	v6.3	1998-03-01	Patches submitted (Tatsuo Ishii (mailto:t-ishii@sra.co.jp))
SVR4	MIPS	v6.4	1998-10-28	No 64-bit int compiler support (Frank Ridderbusch (mailto:ridderbusch.pad@sn-i.de))
Windows	x86	v6.4	1999-01-06	Client-side libraries or ODBC/JDBC. No server yet. (Magnus Hagander (mha@sollentuna.net)
Windows NT	x86	v6.5	1999-05-26	Working with the Cygwin library. (Daniel Horak (mailto:Dan.Horak@email.c-z))

Platforms listed for v6.3.x and v6.4.x should also work with v6.5, but we did not receive explicit confirmation of such at the time this list was compiled.

> Note: For Windows NT, the server-side port of Postgres has recently been accomplished. The Cygnus library is required to compile it.

Unsupported Platforms

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 2-2. Possibly Incompatible Platforms

OS	Processor	Version	Reported	Remarks
MacOS	all	v6.3	1998-03-01	Not library compatible; use ODBC/JDBC
NextStep	x86	v6.x	1998-03-01	Client-only support; v1.0.9 worked with patches (David Wetzel (mailto:dave@turbocat.de))
SVR4 4.4	m88k	v6.2.1	1998-03-01	Confirmed with patching; v6.4.x will need TAS spinlock code (Doug Winterburn (mailto:dlw@seavme.xroads.com))
Ultrix	MIPS,VA-X?	v6.x	1998-03-01	No recent reports; obsolete?

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:

```
Directory and file names:
  --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=incdir
                         site header files for tk/tcl, etc in DIR
  --with-libs=incdir
                         also search for libraries in DIR
  --with-libraries=libdir
                         also search for libraries in DIR
  --enable-locale        enable locale support
  --enable-recode        enable cyrillic recode support
  --with-mb=encoding
                         enable multi-byte support
  --with-pgport=portnum
                         change default startup 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
  --with-odbc            build ODBC driver package
  --with-odbcinst=odbcdir
                         change default directory for odbcinst.ini
  --enable-cassert       enable assertion checks (debugging)
  --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++.

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.

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

A few of the many variables which 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) (send mail to subscribe (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).

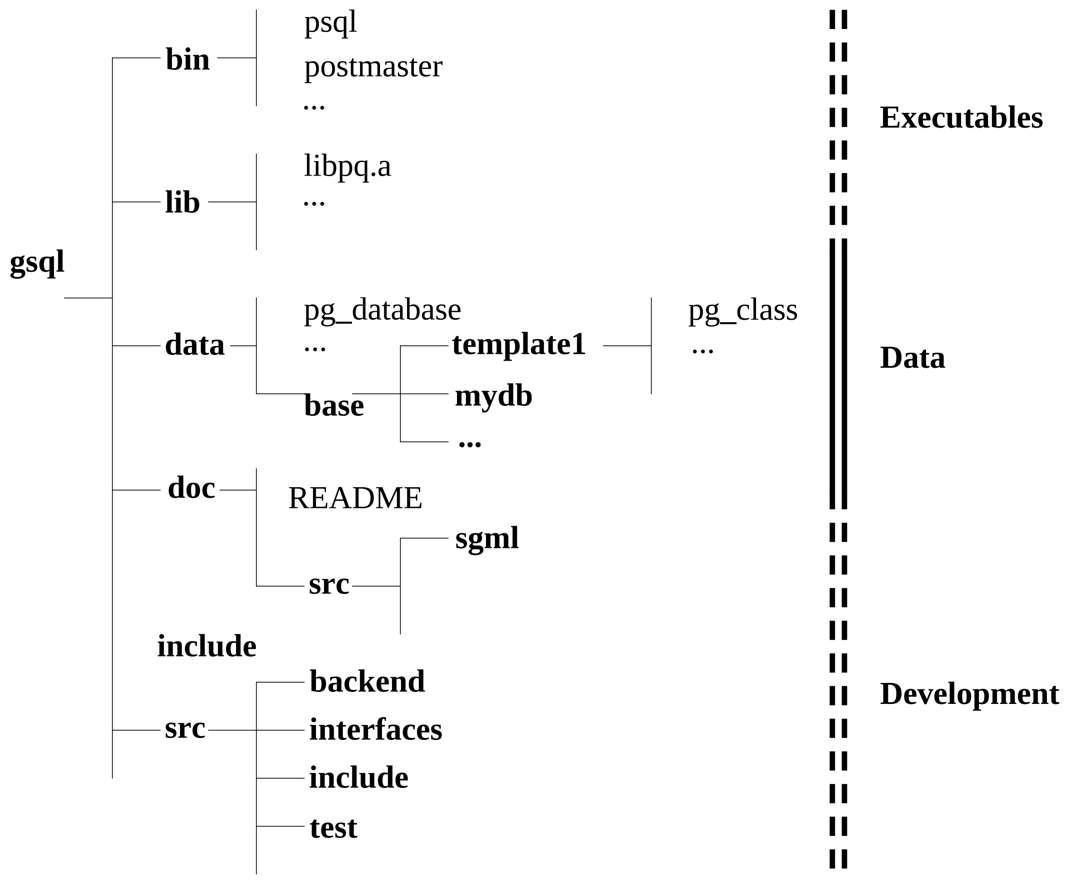
Table 3-1. Kerberos Parameter Examples

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

Complete installation instructions for Postgres v6.5.

Before installing Postgres, you may wish to visit www.postgresql.org (http://www.postgresql.org) for up to date information, patches, etc.

These installation instructions assume:

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).

Commands were tested on RedHat Linux version 5.2 using the tcsh shell. Except where noted, they will probably work on most systems. Commands like ps and tar may vary wildly between platforms on what options you should use. Use common sense before typing in these commands.

Our Makefiles require GNU make (called gmake in this document). They will not work with non-GNU make programs. If you have GNU make installed under the name make instead of gmake , then you will use the command make instead. That's OK, but you need to have the GNU form of make to succeed with an installation.

Requirements to Run Postgres

Up to date information on supported platforms is at http://www.postgresql.org/docs/admin/install.htm (http://www.postgresql.org/docs/admin/install.htm). In general, most Unix-compatible platforms with modern libraries should be able to run Postgres.

Although the minimum required memory for running Postgres is as little as 8MB, there are noticable improvements in runtimes for the regression tests when expanding memory up to 96MB on a relatively fast dual-processor system running X-Windows. The rule is you can never have too much memory.

Check that you have sufficient disk space. You will need about 30 Mbytes for /usr/src/pgsql, about 5 Mbytes for /usr/local/pgsql (excluding your database) and 1 Mbyte for an empty database. The database will temporarily grow to about 20 Mbytes during the regression tests. You will also need about 3 Mbytes for the distribution tar file.

We therefore recommend that during installation and testing you have well over 20 Mbytes free under /usr/local and another 25 Mbytes free on the disk partition containing your database. Once you delete the source files, tar file and regression database, you will need 2 Mbytes for /usr/local/pgsql, 1 Mbyte for the empty database, plus about five times the space you would require to store your database data in a flat file.

To check for disk space, use

```
$ df -k
```

Installation Procedure

Postgres Installation

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

1. Read any last minute information and platform specific porting notes. There are some platform specific notes at the end of this file for Ultrix4.x, Linux, BSD/OS and NeXT. There are other files in directory /usr/src/pgsql/doc, including files FAQ-Irix and FAQ-Linux. Also look in directory ftp://ftp.postgresql.org/pub. If there is a file called INSTALL in this directory then this file will contain the latest installation information.

   Please note that a "tested" platform in the list given earlier simply means that someone went to the effort at some point of making sure that a Postgres distribution would compile and run on this platform without modifying the code. Since the current developers will not have access to all of these platforms, some of them may not compile cleanly and pass the regression tests in the current release due to minor problems. Any such known problems and their solutions will be posted in ftp://ftp.postgresql.org/pub/INSTALL.

2. Create the Postgres superuser account (postgres is commonly used) if it does not already exist.

   The owner of the Postgres files can be any unprivileged user account. It must not be root, bin, or any other account with special access rights, as that would create a security risk.

3. Log in to the Postgres superuser account. Most of the remaining steps in the installation will happen in this account.

4. Ftp file ftp://ftp.postgresql.org/pub/postgresql-v6.5.tar.gz (ftp://ftp.postgresql.org/pub/postgresql-v6.5.tar.gz) from the Internet. Store it in your home directory.

5. Some platforms use flex. If your system uses flex then make sure you have a good version. To check, type

   ```
   $ flex --version
   ```

   If the flex command is not found then you probably do not need it. If the version is 2.5.2 or 2.5.4 or greater then you are okay. If it is 2.5.3 or before 2.5.2 then you will have to upgrade flex. You may get it at ftp://prep.ai.mit.edu/pub/gnu/flex-2.5.4.tar.gz.

   If you need flex and don’t have it or have the wrong version, then you will be told so when you attempt to compile the program. Feel free to skip this step if you aren’t sure you need it. If you do need it then you will be told to install/upgrade flex when you try to compile Postgres.

You may want to do the entire flex installation from the root account, though that is not absolutely necessary. Assuming that you want the installation to place files in the usual default areas, type the following:

```
$ su -
$ cd /usr/local/src
ftp prep.ai.mit.edu
ftp> cd /pub/gnu/
ftp> binary
ftp> get flex-2.5.4.tar.gz
ftp> quit
$ gunzip -c flex-2.5.4.tar.gz | tar xvf -
$ cd flex-2.5.4
$ configure --prefix=/usr
$ gmake
$ gmake check
# You must be root when typing the next line:
$ gmake install
$ cd /usr/local/src
$ rm -rf flex-2.5.4
```

This will update files /usr/man/man1/flex.1, /usr/bin/flex, /usr/lib/libfl.a, /usr/include/FlexLexer.h and will add a link /usr/bin/flex++ which points to flex.

6. If you are not upgrading an existing system then skip to step 9. If you are upgrading an existing system then back up your database. For alpha- and beta-level releases, the database format is liable to change, often every few weeks, with no notice besides a quick comment in the HACKERS mailing list. Full releases always require a dump/reload from previous releases. It is therefore a bad idea to skip this step.

    Tip: Do not use the pg_dumpall script from v6.0 or everything will be owned by the Postgres super user.

    To dump your fairly recent post-v6.0 database installation, type

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

    To use the latest pg_dumpall script on your existing older database before upgrading Postgres, pull the most recent version of pg_dumpall from the new distribution:

    ```
    $ cd
    $ gunzip -c postgresql-v6.5.tar.gz \
        | tar xvf - src/bin/pg_dump/pg_dumpall
    $ chmod a+x src/bin/pg_dump/pg_dumpall
    $ src/bin/pg_dump/pg_dumpall > db.out
    $ rm -rf src
    ```

    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.

    If the pg_dumpall command seems to take a long time and you think it might have died, then, from another terminal, type

    ```
    $ ls -l db.out
    ```

    several times to see if the size of the file is growing.

    Please note that 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. You should also read the release notes which should cover any release-specific issues.

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.

7. If you are upgrading an existing system then kill the postmaster. Type

   ```
   $ ps -ax | grep postmaster
   ```

   This should list the process numbers for a number of processes. Type the following line, with pid replaced by the process id for process postmaster. (Do not use the id for process "grep postmaster".) Type

   ```
   $ kill pid
   ```

   to actually stop the process.

   Tip: On systems which have Postgres started at boot time, there is probably a startup file which will accomplish the same thing. For example, on my Linux system I can type

   ```
   $ /etc/rc.d/init.d/postgres.init stop
   ```

   to halt Postgres.

8. If you are upgrading an existing system then move the old directories out of the way. If you are short of disk space then you may have to back up and delete the directories instead. If you do this, save the old database in the /usr/local/pgsql/data directory tree. At a minimum, save file /usr/local/pgsql/data/pg_hba.conf.

   Type the following:

   ```
   $ su -
   $ cd /usr/src
   $ mv pgsql pgsql_6_0
   $ cd /usr/local
   $ mv pgsql pgsql_6_0
   $ exit
   ```

   If you are not using /usr/local/pgsql/data as your data directory (check to see if environment variable PGDATA is set to something else) then you will also want to move this directory in the same manner.

9. Make new source and install directories. The actual paths can be different for your installation but you must be consistent throughout this procedure.

   Note: There are two places in this installation procedure where you will have an opportunity to specify installation locations for programs, libraries, documentation, and other files. Usually it is sufficient to specify these at the gmake install stage of installation.

Type

```
$ su
$ cd /usr/src
$ mkdir pgsql
$ chown postgres:postgres pgsql
$ cd /usr/local
$ mkdir pgsql
$ chown postgres:postgres pgsql
$ exit
```

10. Unzip and untar the new source file. Type

    ```
    $ cd /usr/src/pgsql
    $ gunzip -c ~/postgresql-v6.5.tar.gz | tar xvf -
    ```

11. Configure the source code for your system. It is this step at which you can specify your actual installation path for the build process (see the --prefix option below). Type

    ```
    $ cd /usr/src/pgsql/src
    $ ./configure [ options ]
    ```

    a. Among other chores, the configure script selects a system-specific "template" file from the files provided in the template subdirectory. If it cannot guess which one to use for your system, it will say so and exit. In that case you’ll need to figure out which one to use and run configure again, this time giving the --with-template=TEMPLATE option to make the right file be chosen.

       Please Report Problems: If your system is not automatically recognized by configure and you have to do this, please send email to scrappy@hub.org (mailto:scrappy@hub.org) with the output of the program ./config.guess. Indicate what the template file should be.

    b. Choose configuration options. Check *Configuration Options* for details. However, for a plain-vanilla first installation with no extra options like multi-byte character support or locale collation support it may be adequate to have chosen the installation areas and to run configure without extra options specified. The configure script accepts many additional options that you can use if you don’t like the default configuration. To see them all, type

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

    c. Here is the configure script used on a Sparc Solaris 2.5 system with /opt/postgres specified as the installation base directory:

       ```
       $ ./configure --prefix=/opt/postgres \
           --with-template=sparc_solaris-gcc --with-pgport=5432 \
           --enable-hba --disable-locale
       ```

       Tip: Of course, you may type these three lines all on the same line.

12. Install the man and HTML documentation. Type

    ```
    $ cd /usr/src/pgsql/doc
    $ gmake install
    ```

    The documentation is also available in Postscript format. Look for files ending with .ps.gz in the same directory.

13. Compile the program. Type

    ```
    $ cd /usr/src/pgsql/src
    $ gmake all >& make.log &
    $ tail -f make.log
    ```

The last line displayed will hopefully be

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

Remember, gmake may be called make on your system. At this point, or earlier if you wish, type control-C to get out of tail. (If you have problems later on you may wish to examine file make.log for warning and error messages.)

> Note: You will probably find a number of warning messages in make.log. Unless you have problems later on, these messages may be safely ignored.

If the compiler fails with a message stating that the flex command cannot be found then install flex as described earlier. Next, change directory back to this directory, type

```
$ gmake clean
```

then recompile again.

Compiler options, such as optimization and debugging, may be specified on the command line using the COPT variable. For example, typing

```
$ gmake COPT="-g" all >& make.log &
```

would invoke your compiler's -g option in all steps of the build. See src/Makefile.global.in for further details.

14. Install the program. Type

    ```
    $ cd /usr/src/pgsql/src
    $ gmake install >& make.install.log &
    $ tail -f make.install.log
    ```

    The last line displayed will be

    ```
    gmake[1]: Leaving directory '/usr/src/pgsql/src/man'
    ```

    At this point, or earlier if you wish, type control-C to get out of tail. Remember, gmake may be called make on your system.

15. If necessary, tell your system how to find the new shared libraries. You can do one of the following, preferably the first:

    a. As root, edit file /etc/ld.so.conf. Add a line

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

       to the file. Then run command /sbin/ldconfig.

    b. In a bash shell, type

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

    c. In a csh shell, type

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

    Please note that the above commands may vary wildly for different operating systems. Check the platform specific notes, such as those for Ultrix4.x or and for non-ELF Linux.

    If, when you create the database, you get the message

    ```
    pg_id: can't load library 'libpq.so'
    ```

    then the above step was necessary. Simply do this step, then try to create the database again.

16. If you used the --with-perl option to configure, check the install log to see whether the Perl module was actually installed. If you've followed our advice to make the Postgres files be owned by an unprivileged userid, then the Perl module won't have been installed, for lack

of write privileges on the Perl library directories. You can complete its installation, either now or later, by becoming the user that does own the Perl library (often root) (via su) and doing

```
$ cd /usr/src/pgsql/src/interfaces/perl5
$ gmake install
```

17. If it has not already been done, then prepare account postgres for using Postgres. Any account that will use Postgres must be similarly prepared.

    There are several ways to influence the runtime environment of the Postgres server. Refer to the Administrator’s Guide for more information.

    > Note: The following instructions are for a bash/sh shell. Adapt accordingly for other shells.

    a. Add the following lines to your login environment: shell, ~/.bash_profile:

       ```
       PATH=$PATH:/usr/local/pgsql/bin
             MANPATH=$MANPATH:/usr/local/pgsql/man
             PGLIB=/usr/local/pgsql/lib
             PGDATA=/usr/local/pgsql/data
             export PATH MANPATH PGLIB PGDATA
       ```

    b. Several regression tests could fail if the user’s locale collation scheme is different from that of standard C locale.

       If you configure and compile Postgres with the --enable-locale option then set locale environment to C (or unset all LC_* variables) by putting these additional lines to your login environment before starting postmaster:

       ```
       LC_COLLATE=C
       LC_CTYPE=C
       export LC_COLLATE LC_CTYPE
       ```

    c. Make sure that you have defined these variables before continuing with the remaining steps. The easiest way to do this is to type:

       ```
       $ source ~/.bash_profile
       ```

18. Create the database installation from your Postgres superuser account (typically account postgres). Do not do the following as root! This would be a major security hole. Type

    ```
    $ initdb
    ```

19. Set up permissions to access the database system. Do this by editing file /usr/local/pgsql/data/pg_hba.conf. The instructions are included in the file. (If your database is not located in the default location, i.e. if PGDATA is set to point elsewhere, then the location of this file will change accordingly.) This file should be made read only again once you are finished. If you are upgrading from v6.0 or later you can copy file pg_hba.conf from your old database on top of the one in your new database, rather than redoing the file from scratch.

20. Briefly test that the backend will start and run by running it from the command line.

    a. Start the postmaster daemon running in the background by typing

    ```
    $ cd
    $ postmaster -i
    ```

    b. Create a database by typing

    ```
    $ createdb
    ```

    c. Connect to the new database:

    ```
    $ psql
    ```

    d. And run a sample query:

    ```
    postgres=> SELECT datetime 'now';
    ```

    e. Exit psql:

    ```
    postgres=> \q
    ```

    f. Remove the test database (unless you will want to use it later for other tests):

    ```
    $ destroydb
    ```

21. *Run postmaster in the background from your Postgres superuser account (typically account postgres). Do not run postmaster from the root account!*

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

    Here are some suggestions on how to do this, contributed by various users.

    Whatever you do, postmaster must be run by the Postgres superuser (postgres?) and not by root. This is why all of the examples below start by switching user (su) to postgres. These commands also take into account the fact that environment variables like PATH and PGDATA may not be set properly. The examples are as follows. Use them with extreme caution.

    If you are installing from a non-privileged account and have no root access, then start the postmaster and send it to the background:

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

    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.

In RedHat Linux edit file /etc/inittab to add the following as a single line:

```
pg:2345:respawn:/bin/su - postgres -c
    "/usr/local/pgsql/bin/postmaster -D/usr/local/pgsql/data
    >> /usr/local/pgsql/server.log 2>&1 </dev/null"
```

(The author of this example says this example will revive the postmaster if it dies, but he doesn't know if there are other side effects.)

22. Run the regression tests. The file /usr/src/pgsql/src/test/regress/README has detailed instructions for running and interpreting the regression tests. A short version follows here:

    a. Type

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

       You do not need to type gmake clean if this is the first time you are running the tests.

       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 tests to "fail" on some platforms. The script says a test has failed if there is any difference at all between the actual output of the test and the expected output. Thus, tests may "fail" due to minor differences in wording of error messages, small differences in floating-point roundoff, etc, between your system and the regression test reference platform. "Failures" of this type do not indicate a problem with Postgres. The file ./regression.diffs contains the textual differences between the actual test output on your machine and the "expected" output (which is simply what the reference system produced). You should carefully examine each difference listed to see whether it appears to be a significant issue.

       For example,

       > For a i686/Linux-ELF platform, no tests failed since this is the v6.5 regression testing reference platform.

       Even if a test result clearly indicates a real failure, it may be a localized problem that will not affect you. An example is that the int8 test will fail, producing obviously incorrect output, if your machine and C compiler do not provide a 64-bit integer data type (or if they do but configure didn't discover it). This is not something to worry about unless you need to store 64-bit integers.

       Conclusion? If you do see failures, try to understand the nature of the differences and then decide if those differences will affect your intended use of Postgres. The regression tests are a helpful tool, but they may require some study to be useful.

       After running the regression tests, type

       ```
       $ destroydb regression
       $ cd /usr/src/pgsql/src/test/regress
       $ gmake clean
       ```

       to recover the disk space used for the tests. (You may want to save the regression.diffs file in another place before doing this.)

23. If you haven't already done so, this would be a good time to modify your computer to do regular maintainence. The following should be done at regular intervals:

    Minimal Backup Procedure

    1. Run the SQL command VACUUM. This will clean up your database.
    2. Back up your system. (You should probably keep the last few backups on hand.) Preferably, no one else should be using the system at the time.

    Ideally, the above tasks should be done by a shell script that is run nightly or weekly by cron. Look at the man page for crontab for a starting point on how to do this. (If you do it, please e-mail us a copy of your shell script. We would like to set up our own systems to do this too.)

24. If you are upgrading an existing system then reinstall your old database. Type

    ```
    $ cd
    $ psql -e template1 < db.out
    ```

    If your pre-v6.2 database uses either path or polygon geometric data types, then you will need to upgrade any columns containing those types. To do so, type (from within psql)

    ```
    UPDATE FirstTable SET PathCol = UpgradePath(PathCol);
    UPDATE SecondTable SET PathCol = UpgradePath(PathCol);
    ...
    VACUUM;
    ```

    UpgradePath() checks to see that a path value is consistant with the old syntax, and will not update a column which fails that examination. UpgradePoly() cannot verify that a polygon is in fact from an old syntax, but RevertPoly() is provided to reverse the effects of a mis-applied upgrade.

25. If you are a new user, you may wish to play with Postgres as described below.

26. Clean up after yourself. Type

    ```
    $ rm -rf /usr/src/pgsql_6_5
    $ rm -rf /usr/local/pgsql_6_5
    # Also delete old database directory tree if it is not in
    #  /usr/local/pgsql_6_5/data
    $ rm ~/postgresql-v6.5.tar.gz
    ```

27. You will probably want to print out the documentation. 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 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.

    ```
    $ alias gshp='gs -sDEVICE=laserjet -r300 -dNOPAUSE'
    $ export GS_LIB=/usr/share/ghostscript:/usr/share/ghostscript/fonts
    $ gunzip user.ps.gz
    $ gshp -sOUTPUTFILE=user.hp user.ps
    $ gzip user.ps
    $ lpr -l -s -r manpage.hp
    ```

28. The Postgres team wants to keep Postgres working on all of the supported platforms. We therefore ask you to let us know if you did or did not get Postgres to work on you system. Please send a mail message to pgsql-ports@postgresql.org (mailto:pgsql-ports@postgresql.org) telling us the following:

    The version of Postgres (v6.5, 6.4.2, beta 981014, etc.).

Your operating system (i.e. RedHat v5.1 Linux v2.0.34).

Your hardware (SPARC, i486, etc.).

Did you compile, install and run the regression tests cleanly? If not, what source code did you change (i.e. patches you applied, changes you made, etc.), what tests failed, etc. It is normal to get many warning when you compile. You do not need to report these.

29. Now create, access and manipulate databases as desired. Write client programs to access the database server. In other words, enjoy!

Playing with Postgres

After Postgres is installed, a database system is created, a postmaster daemon is running, and the regression tests have passed, you'll want to see Postgres do something. That's easy. Invoke the interactive interface to Postgres, psql:

```
% psql template1
```

(psql has to open a particular database, but at this point the only one that exists is the template1 database, which always exists. We will connect to it only long enough to create another one and switch to it.)

The response from psql is:

```
Welcome to the POSTGRESQL interactive sql monitor:
  Please read the file COPYRIGHT for copyright terms of POSTGRESQL

   type \? for help on slash commands
   type \q to quit
   type \g or terminate with semicolon to execute query
 You are currently connected to the database: template1

template1=>
```

Create the database foo:

```
template1=> create database foo;
CREATEDB
```

(Get in the habit of including those SQL semicolons. Psql won't execute anything until it sees the semicolon or a "\g" and the semicolon is required to delimit multiple statements.)

Now connect to the new database:

```
template1=> \c foo
connecting to new database: foo
```

("slash" commands aren't SQL, so no semicolon. Use \? to see all the slash commands.)

And create a table:

```
foo=> create table bar (i int4, c char(16));
CREATE
```

Then inspect the new table:

```
foo=> \d bar

Table    = bar
```

```
+----------------+---------------+-------+
|   Field        |     Type      | Length|
+----------------+---------------+-------+
| i              | int4          |     4 |
| c              | (bp)char      |    16 |
+----------------+---------------+-------+
```

And so on. You get the idea.

The Next Step

Questions? Bugs? Feedback? First, read the files in directory /usr/src/pgsql/doc/. The FAQ in this directory may be particularly useful.

If Postgres failed to compile on your computer then fill out the form in file /usr/src/pgsql/doc/bug.template and mail it to the location indicated at the top of the form.

Check on the web site at http://www.postgresql.org For more information on the various support mailing lists.

Porting Notes

Check for any platform-specific FAQs in the doc/ directory of the source distribution.

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 command

```
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 &
```

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

```
% postmaster -S
```

and the postmaster will be "S"ilent. Notice that there is no ampersand ("&") at the end of the last example so postmaster will be running in the foreground.

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.

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.

notifyunlock

Unlock of pg_listener after notify.

notifyhack

Remove duplicate tuples from pg_listener.

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.

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.

Chapter 9. Adding and Deleting Users

createuser enables specific users to access Postgres. destroyuser removes users and prevents them from accessing Postgres. Note that these commands only affect users with respect to Postgres; they have no effect on users 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
% destroydb test
```

Chapter 11. Managing a Database

Now that Postgres 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 mydb
```

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 16 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 mydb database by typing the command:

```
% psql mydb
```

You will be greeted with the following message:

```
Welcome to the Postgres interactive sql monitor:

  type \? for help on slash commands
  type \q to quit
  type \g or terminate with semicolon to execute query
You are currently connected to the database: mydb

mydb=>
```

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:

```
mydb=> \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:

```
mydb=> \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, say myFile, instead of entering them interactively, type:

```
mydb=> \i fileName
```

To get out of psql and return to UNIX, type

```
mydb=> \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:

```
% destroydb mydb
```

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.

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.

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 &
```

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

```
% postmaster -S
```

and the postmaster will be "S"ilent. Notice that there is no ampersand ("&") at the end of the last example so postmaster will be running in the foreground.

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 developed by Jolly Chen and Andrew Yu. It tests standard SQL operations as well as the extended capabilities of PostgreSQL.

These tests have recently been revised by Marc Fournier and Thomas Lockhart and are now packaged as functional units which should make them easier to run and easier to interpret. From PostgreSQL v6.1 onward the regression tests are current for every official release.

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 current tests are evaluated using a simple "diff" algorithm, and are sensitive to small system differences. For apparently failed tests, examining the differences may reveal that the differences are not significant.

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).

Regression Environment

The regression test is invoked by the make command which compiles a C program into a shared library in the current directory. Localized shell scripts are also created in the current directory. The output file templates are massaged into the ./expected/*.out files. The localization replaces macros in the source files with absolute pathnames and user names.

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

Directory Layout

Note: This should become a table in the previous section.

```
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 represent what the results *actually* look
            like. Also used as temporary storage for table copy testing.
```

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.

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

Postgres Regression Configuration

1. The file /usr/src/pgsql/src/test/regress/README has detailed instructions for running and interpreting the regression tests. A short version follows here:

   If the postmaster is not already running, start the postmaster on an available window by typing

   ```
   postmaster
   ```

   or start the postmaster daemon running in the background by typing

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

   Run postmaster from your Postgres super user account (typically account postgres).

Note: Do not run postmaster from the root account.

2. If you have previously invoked the regression test, clean up the working directory with:

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

   You do not need to type "gmake clean" if this is the first time you are running the tests.

3. Build the regression test. Type

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

4. Run the regression tests. Type

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

5. 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". For the failed tests, use diff to compare the files in directories ./results and ./expected. If float8 failed, type something like:

   ```
   cd /usr/src/pgsql/src/test/regress
        diff -w expected/float8.out results
   ```

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

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

   to recover the temporary disk space used by the tests.

Regression Analysis

The results are in files in the ./results directory. These results can be compared with results in the ./expected directory using ’diff’. (The test script does this for you, and leaves the differences in ./regression.diffs.)

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.

OID differences

There are several places where PostgreSQL OID (object identifiers) appear in ’regress.out’. OID’s are unique 32-bit integers which are generated by the PostgreSQL backend whenever a table row is inserted or updated. If you run the regression test on a non-virgin database or run it multiple times, the OID’s reported will have different values. The following SQL statements in ’misc.out’ have shown this behavior: QUERY: SELECT user_relns() AS user_relns ORDER BY user_relns; The ’a,523676’ row is composed from an OID.

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.

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 folowing:

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

Random differences

There is at least one test case in random.out which is intended to produce random results. This causes random to fail the regression testing once in a while. Typing

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

should produce only one or a few lines of differences for this reason, but other floating point differences on dissimilar architectures might cause many more differences. See the release notes below.

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. (see below) The original sample.regress.out file was from the postgres-1.01 release constructed by Jolly Chen and is included here for reference. It may have been created on a DEC ALPHA machine as the Makefile.global in the postgres-1.01 release has PORTNAME=alpha.

Chapter 15. Release Notes

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.

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 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 or pg_dumpall is required for those wishing to migrate data from any previous release of Postgres.

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 coersion 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)

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 coersion 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

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 coersion 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 coersion 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
```

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.

SQL Reference Books

The Practical SQL Handbook, Using Structured Query Language , 3, Judity Bowman, Sandra Emerson, and Marcy Damovsky, 0-201-44787-8, 1996, Addison-Wesley, 1997.

A Guide to the SQL Standard, A user's guide to the standard database language SQL , 4, C. J. Date and Hugh Darwen, 0-201-96426-0, 1997, Addison-Wesley, 1997.

An Introduction to Database Systems, 6, C. J. Date, 1, 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, 1, Computer Science Press , 1988 .

PostgreSQL-Specific Documentation

The PostgreSQL Administrator's Guide , The Administrator's Guide , Edited by Thomas Lockhart, 1998-10-01, The PostgreSQL Global Development Group.

The PostgreSQL Developer's Guide , The Developer's Guide , Edited by Thomas Lockhart, 1998-10-01, The PostgreSQL Global Development Group.

The PostgreSQL Programmer's Guide , The Programmer's Guide , Edited by Thomas Lockhart, 1998-10-01, The PostgreSQL Global Development Group.

The PostgreSQL Tutorial Introduction , The Tutorial , Edited by Thomas Lockhart, 1998-10-01, The PostgreSQL Global Development Group.

The PostgreSQL User's Guide , The User's Guide , Edited by Thomas Lockhart, 1998-10-01, The PostgreSQL Global Development Group.

Enhancement of the ANSI SQL Implementation of PostgreSQL , Simkovics, 1998 , 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 , Yu and Chen, 1995 , 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 , Olson, 1993 , 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 , Ong and Goh, 1990 , L. Ong and J. Goh, April, 1990, ERL Technical Memorandum M90/33, University of California, Berkeley CA.

The Postgres Data Model , Rowe and Stonebraker, 1987 , 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 , Stonebraker and Rowe, 1986 , M. Stonebraker and L. Rowe, May 1986, Conference on Management of Data, Washington DC, ACM-SIGMOD, 1986.

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

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

A Commentary on the Postgres Rules System , Stonebraker et al, 1989, 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) , Stonebraker, M, 1989b, M. Stonebraker, Dec. 1989, Record 18(no.4):4-11, SIGMOD, 1989.

The Implementation of Postgres , Stonebraker, Rowe, Hirohama, 1990 , 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 , Stonebraker et al, ACM, 1990 , M. Stonebraker and et al, June 1990, Conference on Management of Data, ACM-SIGMOD.