Database−SQL−RDBMS HOW−TO document for Linux (PostgreSQL Object Relational Database System)

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Database−SQL−RDBMS HOW−TO document for Linux (PostgreSQL Object Relational Database System)

Al Dev (Alavoor Vasudevan) alavoor@yahoo.com

v45.0, 27 Jan 2001

This document is a "practical guide" to very quickly setup a SQL Database engine and front end tools on an Unix system. It also discusses the International standard language ANSI/ISO SQL and reviews the merits/advantages of the SQL database engine developed by the world−wide internet in an "open development" environment. It is about HOW−TO setup a next generation Object Relational SQL Database "PostgreSQL" on Unix system which can be used as an Application Database Server or as a Web Database Server. PostgreSQL attempts to implement current and future International ISO/ANSI SQL standards. This document also gives information on the database interface programs like Front End GUIs, RAD tools (Rapid Application Development), ODBC, JDBC drivers, "C", "C++", Java, Perl programming interfaces and Web Database Tools. Information given here applies to all Unix/Windows NT platforms and to all other SQL databases. It will be very useful for people who are new to Databases, SQL language and PostgreSQL. This document also has SQL tutorial, SQL syntax which would be very helpful for beginners. Experienced people will find this document as an useful reference guide. For students, the information given here will enable them to get the source code for PostgreSQL relational database system, from which they can learn as to how a RDBMS SQL database engine is created.

---

6. PostgreSQL Quick−Installation Instructions

7. Quick Start Guide

8. Performance Tuning of PostgreSQL server

9. PostgreSQL Supports Extremely Large Databases greater than 200 Gig

10. How can I trust PostgreSQL ? Regression Test Package builds customer confidence

11. Security of Database

12. GUI FrontEnd Tool for PostgreSQL (Graphical User Interface)

13. Interface Drivers for PostgreSQL

14. Perl Database Interface (DBI) Driver for PostgreSQL

15. PostgreSQL Management Tools

16. CPUs for PostgreSQL

17. Setting up multi−boxes PostgreSQL with just one monitor

18. Web−Application−Servers for PostgreSQL

24. Gateway between PostgreSQL and the WWW − WDB−P95

25. "C", "C++", ESQL/C language Interfaces and Bitwise Operators for PostgreSQL

26. Japanese Kanji Code for PostgreSQL

27. PostgreSQL Port to Windows 95/Windows NT

28. Mailing Lists

29. Documentation and Reference Books

---

1. Introduction

The purpose of this document is to provide comprehensive list of pointers/URLs to quickly setup PostgreSQL and also to advocate the benefits of Open Source Code system like PostgreSQL, Linux.

PostgreSQL is pronounced as Post−gres−cue−el (Postgres−QL) and **not** Postgre−es−cue−el.

Each and every computer system in the world needs a database to store/retrieve the information. The primary reason you use the computer is to store, retrieve and process information and do all these very quickly, thereby saving you time. At the same time, the system must be simple, robust, fast, reliable, economical and very easy to use. Database is the most **VITAL SYSTEM** as it stores mission critical information of every company in this world. Each and every industry in this world needs a database system. Industries like telecom, automobile, banks, airlines, etc.. will not function efficiently without a database system. The most popular database systems are based on the International Standard Organisation (ISO) SQL specifications and ANSI SQL (American) standards. The current specifications widely used in the industry are ISO/ANSI SQL 1992. Upcoming standard is the SQL 1998/99 which is also called SQL−3 is still under development. Popular database like Oracle, Sybase and Informix systems are based on these standards or are trying to implement these standards.

Without a standard like ANSI/ISO SQL, it would be very difficult for the customer to develop an application once and run on all the database systems. End user wants to develop an application ONCE using ISO SQL, ODBC, JDBC and deploy on all variety of database systems in the world.

The world's most popular FREE Database which implements some of the ISO SQL, ANSI SQL/98, SQL/92 and ANSI SQL/89 RDBMS is PostgreSQL. PostgreSQL is next generation Object relational database and is targeting on full compliance of SQL standards like ISO/ANSI SQL. PostgreSQL is the only free RDBMS in the world which supports Object databases and SQL. This document will tell you how−to install the database, how to set up the Web database, application database, front end GUIs and interface programs. It is strongly advised that you MUST write your database applications 100 % compliant to standards of ISO/ANSI SQL, ODBC, JDBC so that your application is portable across multiple databases like PostgreSQL, Oracle, Sybase, Informix etc.

You get the highest quality, and lot many features with PostgreSQL as it follows 'Open Source Code development model'. Open Source Code model is the one where the complete source code is given to you and the development takes place on the internet by an extremely vast network of human brains. Future trend shows that most of the software development will take place on the so called "Information Super−Highway" which spans the whole globe. In the coming years, internet growth will be explosive which will further fuel rapid adoption of PostgreSQL by the industry.

By applying the principles of statistics, mathematics and science to software quality, you get the best quality of software only in a 'Open Source Code System' like PostgreSQL, wherein the source code is open to a very vast number of human brains inter−connected by the information super−highway. Greater the number of human brains working, the better will be the quality of software. Open Source Code model will also prevent **RE−INVENTION OF WHEELS**, eliminates **DUPLICATION OF WORK** and will be very economical, saves time in distribution and follows the modern economic laws of optimizing the national and global resources. Once a software work is done by others, then you **DO NOT** need to re−do that again. You will not be wasting your valuable time on something which had already been **WELL DONE**. Your time is extremely precious and it must be utilized efficiently, because you have only 8 hours a day for doing work. As we will be entering the 21st century, there will be a change in the way that you get software for your use. Everybody will give first preference for the open source softwares like PostgreSQL, Linux.

If you buy binaries, you will not get any equity and ownership of source code. Source code is a very valuable asset and binaries have no value. Buying software *may* become a thing of the past. You only need to buy good hardware, it is worth spending money on the hardware and get the software from internet. Important point is that it is the computer hardware which is **doing bulk of the work**. Hardware is the real work horse and software is just driving it. Computer hardware is so much more complex that only 6 nations in the world so far have demonstrated the capability of designing and manufacturing computer chips/hardware. Design and manufacturing of computer chips is an advanced technology. It is a very complex process, capital intensive, requires large investments in plant and production machines which deal with 0.18 micron (even smaller than 0.18) technology. On a single small silicon chip millions of transistors/circuits are densely packed. Companies like Applied Material, AMD, Intel, Cyrix, Hitachi, IBM and others spent significant number of man-years to master the high-technology like Chip Design, Micro-electronics and Nano-electronics. Micro means (one-millionth of meter 10^-6), Nano means (one-billionth of meter 10^-9). Current technology uses micro-electronics of about 0.35 micron using aluminum as conductors and 0.25 micron sizes using copper as conductors of electrons. In near future the technology of 0.10 micron with copper and even nano-electronics will be used to make computer chips. Aluminum conductors will be phased out by copper on computer chips, as copper is a better conductor of electrons. In photolithography process extreme ultraviolet, X-ray or electron-beam techniques will be used to etch circuits for feature size less than 0.15 micron. In about 20 years from now, silicon chips will be phased out by molecular computers and bio chips which will be billions of times faster than silicon chips. Molecules are a group of atoms. And atoms are tiny particles which makes up everything that you see in this world. Molecular computers will use the molecules of matter as ultra-fast electronic on/off switches. When the switch is ON it indicates 1, and when it is OFF it indicates 0. All the computer programs in this world are based on binary (numbers 1 and 0). Table below shows the progress and future advancement trends of computer chips.

```
Advancement of chip capabilities in future
*******************************************
```

Item/Year	1997	1999	2001	2003	2012	2020	2030
Feature size(micron)	0.25	0.18	0.15	0.13	0.05	< 0.00001	atomic
Wafer size(mm)	200	300	300	300	450	Mol/Bio	Quantum
Min Operating Voltage	1.8-2.5	1.5-1.8	1.2-1.5	1.2-1.5	0.5-0.6	< 0.001	minute
Max power dissipation	70	90	110	130	175	600	minute
On-chip frequencey (MHz)	750	1,250	1,500	2,100	10,000	> 50,000	----
DRAM capacity	256 MB	1 GB	2 GB	4 GB	256 GB	> 1000GB	----

As you can see, it is hardware that is high technology and important and software is labor intensive but is a less difficult technology.

On other hand, each and every country in the world develops/makes software. In fact, any person in this world with a small low-cost PC can write software.

Databases like Oracle, Informix, Sybase, IBM DB2 (Unix) are written using the "C" language and binaries are created by compiling the source code and then they are shipped out to customers. Oracle, Sybase, Informix databases are 100 % "C" programs!!

Since a lot of work had been done on PostgreSQL for the past 14 years, it does not make sense to re-create from scratch another database system which satisfies ANSI/ISO SQL. It will be a great advantage to take the existing code and add missing features or enhancements to PostgreSQL and start using it immediately.

Prediction is that demand for "Internet products" like PostgreSQL will grow exponentially as it is capable of maintaining a high quality, low cost, extremely large user−base and developer−base. Those nations which do not use the 'Internet products' will be seriously missing "World−wide Internet Revolution" and will be left far behind other countries. The reason is "Internet" itself is the world's **LARGEST** "software company" and is a large software "power house"!

1.1 Quantum Computers − Quantum Physics Useful !!

As you can see from above table "Advancement of chip capabilities in future" in the years after 2030, database systems like PostgreSQL will be running on **Quantum Computers**. Quantum Computers rely on an atomic particle's traits, such as direction of spin, for creating a state. For example, when the spin is up, a particle could be read as **"one"**, when its spin is down, the particle would be read as **"zero"**. Atoms and nuclei can exist in a state of superposition, where the values of one, zero and the range in between can be represented concurrently. By entangling the spins of atoms, **"qubits"** can become wired together, enabling them to function as a collective whole, bringing about a nonlinear computational power that **far surpasses** the capabilities of supercomputers available today!! At atomic level Quantum Physics comes to assistance to better understand the behaviour of atomic particles.

---

2. Laws of Physics apply to Software!

In this chapter, it will be shown how science plays an important role in the creation of various objects like software, this universe, mass, atoms, energy and even yourself! This chapter also shows why knowledge of science is very important before you start using the products of science.

The golden rule is − *"You MUST not use a product without understanding how it is created!!"* This rule applies to everything − database sytems, computer system, operating system, this universe and even your own human body! It means that you should have complete source code and information about the system. It is important to understand how human body and atoms inside human body works since humans are creating PostgreSQL, MS Windows95 etc..

Creation is a very important step. Persons who are using the objects of science must know how it is created. This applies to even computer systems and PostgreSQL. A majority of people do not have knowledge of science and hence do not know how systems like MS Windows NT/95, Oracle, human body and this universe are created. A vast majority of people do not know what made the universe and MS Windows 95/NT and what is inside it. Complex systems are built from very simple basic building blocks like − millions of universes are created, each universe in turn has millions of super−clusters, each super−cluster has millions of galaxies, each galaxy has millions of stars, some stars have many planets, each planet in turn is made up billions of atoms.*(In the history of this world, **only one universe was created by a man** in ancient India eons ago, but no other case had been reported in the modern history. There is only one man−made universe)* Creating an universe is a much more advanced technology and is more advanced than the atomic bomb which was dropped on Hiroshima and Nagasaki causing **horrible destruction**. Modern nuclear weapons are so tiny and powerful that if such a single nuclear bomb is dropped in pacific ocean then it can completely vaporise the planet earth! The total variety of weapons are infinity. There are weapons to even terminate the universes (it is not a good idea to give nuke weapons technology to every person). Nuclear weapons and other more **powerful divine weapons** were used in the battle field in ancient India! Nobody believed Albert Eienstein (a scientist of 1900's) when he said nuclear weapons can be made which can vaporise big cities.

Software like MS Windows 95 is created simply by "C" and assembler language programs which simply uses 1 and 0 and ***universes like ours are created simply by dashing TWO dissimilar but proper of combination of tiny atomic particles of other dimensions.*** *(Something interesting happened* ***just before*** *dashing of tiny particles)* A human body is created by dashing two dissimilar but proper combination of tiny cells!! *(Something interesting happened* ***just before*** *dashing of tiny cells)* Humans inherited the properties of this universe. The universe you are currently living in was NOT there − all the atoms inside the universe was not there and not even TIME was existing!! Baby universe was born during **big bang** and started expanding and kept growing. Even today our universe is still expanding and is not static!! A person from another universe by name **'Brahma'** created this universe you are currently living in. ***Knowledge is the MOTHER of this universe!!*** 'Brahma' loved on 'Mother knowledge' before the baby universe in which you are living was born!! It is a deal similar to how you were born! Without any 'genes' from Mother Knowledge it is not possible create even a small "C" program!

At some point our universe will close down (in a big crunch) and all the atoms inside the universe will completely vanish and dissappear! All the atoms that you see inside this universe will be gone!

Total number of universes that can be created is **INFINITY** and similarly total number of operating systems that can be created is also **infinity**!! It is infinite cyclic process where universes are born and then later die down. There are millions of universes, which are classified into 3 major categories. Infinite number universes and infinite variety of multi−dimensional atoms collapse down into few *primary−dimensional−universe*. And primary−dimensional universes collapse down into one single focus entity called *'eeshwar'* (eeshara is a ***sanskrit*** word). Very advanced mathematical equations support this theory.

The laws of science and statistics favour the open−source code system like PostgreSQL and Linux. As the internet speed is increasing everyday, and internet is becoming more and MORE reliable, the open−source code system will gain very rapid momentum. And, if rules of statistics and laws of physics are correct, awareness of science grows and when **IGNORANT** people start learning science then the closed source−code systems will eventually vanish from this planet.

Developing a project like PostgreSQL requires resources like energy and time, hence PostgreSQL is a product of energy and time. Since energy and time can be explained only by science, there is a direct co−relation between physics and software projects like PostgreSQL, Linux. Laws of science (Physics) applies everywhere and at all the times, to anything that you do, even while you are developing the software projects.

Physics is in action even while you are talking (sound waves), walking (friction between ground and your feet), reading a book or writing software. Every science in this world has a deep root in mathematics, including PostgreSQL. PostgreSQL uses 'Modern Algebra' which is a tiny branch of mathematics. Modern algebra deals with 'Set Theory', 'Relational Algebra', science of Groups, Rings, Collections, Sets, Unions, Intersections, Exclusions, Domains, Lists, etc...

The software like PostgreSQL is existing today because of the energy and time. And mass and energy are ONE and the **SAME** entity. There are infinite number of methods to *unlock* mass and convert it into enery. Mass is a highly concentrated energy. The fact that mass and energy are same was unknown to people 100 years ago! And even **today** it is unknown to world population that internet is the largest software "power house" and the largest "software company" in the world!

Cells in the human brains consume energy while processing (creating software), by converting the chemical energy from food into electrical and heat energy. Even while you are reading this paragraph, the cells in your brain are burning out the fuel and are using tiny amounts of energy. All of these implies that human brain is a thermodynamic heat engine. Because human brain is a thermodynamic engine, the laws of thermodynamics applies to brain and hence thermodynamics has indirect effects on software like PostgreSQL.

There can be infinite number of colors, computer langauages, computer chip designs and theories but there CANNOT be ONE SINGLE PERFECT color, computer language, design or system! What you can have is only a NEAR PERFECT color(wavelength), system, database, or theory! **Nature is like a *KALIEDOSCOPE* −** there are infinite number of dimensions, infinite variety particles of other dimensions but they all combine into very few primary dimensions and vice−versa.

By combining the energies of millions of people around the world via internet it is possible to achieve a **NEAR PERFECT** system (including a database software system). Individually, the energy of each person will be minute, but by networking a large number of people, the total energy will be huge which can be focused on a project to generate a near perfect system.

The energy is measured in Joules, kiloJoules or kilograms of mass, and time is measured in seconds or hours. And power is energy divided by time and is measured in Watts or kiloWatts .

```
        Energy of each person = y Joules
or in terms of mass
        Energy of each person = y grams
The conversion factor between mass and energy is E = m * c * c
where 'c' is the speed of light and 'm' is the mass.
        Time = 8 hours (This is constant since each person has only 8 hours a day)
        Power = Energy / Time
                 = (y / (8 * 60 * 60) ) Watts
        Total Power of the world = n * (y / (8 * 60 * 60) ) Watts
where n = number of persons working on the project.
```

From the above equation it is clear that increasing the 'n' will greatly improve the quality of product. Greater the 'n' then greater will be the power (in KiloWatts). You can wonder how much total energy (in KiloJoules) and total power (in KiloWatts) the global internet can focus on a system like Linux and PostgreSQL!

It is very clear that internet can network a vast number of people, which implies internet has a lot of energy and time which can produce much higher quality software products in much shorter time as compared to commercial companies. Even very big companies like Microsoft and IBM cannot overpower and overrule the laws of Physics but will eventually **SURRENDER UNTO** laws of science!

Conclusion is − because of laws of science, 'open source code' system like PostgreSQL, Linux will prevail and will be always much better than 'closed source code' system and it is possible to prove this statement scientifically. Man should not waste time creating too many duplicate software products.

3. What is PostgreSQL ?

PostgreSQL is a free database, complete source code is given to you and is an Object−Relational Database System targetting on ANSI ISO/SQL 1998, 92 and runs on diverse hardware platforms and Operating systems. The ultimate objective and the final goal of PostgreSQL is to become 100 % compliant to ANSI/ISO SQL and also to become the number ONE open generic Database in the world.

PostgreSQL is pronounced as Post−gres−cue−el (Postgres−QL) and **not** Postgre−es−cue−el.

Today, PostgreSQL is the **most advanced** system in the world and it is surprising that many commercial database systems could not match the quality, features and capabilities of PostgreSQL !! PostgreSQL is the joint effort of many nations around the globe and is a project similar to **International Space Station**. PostgreSQL will remain the **number one database system** for many decades into future since it is an open−source code system.

The fundamental idea behind PostgreSQL is − once a module of code is written than you should not waste even a milli−second of your time trying to re−invent it!!

Informix Universal server (released 1997) is based on earlier version of PostgreSQL because Informix bought Illustra Inc. and integrated with Informix. Illustra database was based on Postgres (earlier version of PostgreSQL).

PostgreSQL is an enhancement of the POSTGRES database management system, a next−generation DBMS research prototype. While PostgreSQL retains the powerful data model and rich data types of POSTGRES, it replaces the PostQuel query language with an extended subset of SQL.

PostgreSQL development is being performed by a team of Internet developers who all subscribe to the PostgreSQL development mailing list. The current coordinator is Marc G. Fournier

- scrappy@postgreSQL.org

This team is now responsible for all current and future development of PostgreSQL. Ofcourse, the database customer himself is the developer of PostgreSQL! The development load is distributed among a very large number of database end−users on internet.

The authors of PostgreSQL 1.01 were Andrew Yu and Jolly Chen. The original Postgres code, from which PostgreSQL is derived, was the effort of many graduate students, undergraduate students, and staff programmers and working under the direction of Professor Michael Stonebraker at the University of California, Berkeley.

Millions of PostgreSQL is installed as Database servers, Web database servers and Application data servers. It is very sophisticated object relational database system (ORDBMS).

PostgreSQL runs on Solaris, SunOS, HPUX, AIX, Linux, Irix, Digital Unix, BSDi,NetBSD, FreeBSD, SCO unix, NEXTSTEP, Unixware and all and every flavor of Unix. Port to Windows NT is done using Cygnus cygwin32 package.

PostgreSQL and related items in this document are subject to the COPYRIGHT from University of California, Berkeley.

3.1 White Paper

PostgreSQL details in nutshell:

- Title: PostgreSQL SQL RDBMS Database (Object Relational Database Management System)
- Current Version: 7.0.1
- Age: PostgreSQL is 15 years old. Developed since 1985
- Authors: Developed by millions/universities/companies on internet for the past 15 YEARS

The white paper on PostgreSQL is at http://www.greatbridge.com.

PostgreSQL is pronounced as Post−gres−cue−el (Postgres−QL) and **not** Postgre−es−cue−el.

4. Which one? PostgreSQL or MySQL ?

4.1 PostgreSQL defeated Oracle, IBM DB2, MS SQL server and others!!

PostgreSQL defeated Oracle 8 (and 8i), IBM DB2, MS SQL server, Sybase, Interbase and MySQL in standard benchmark tests in performance, speed, scalability and reliability! Read the benchmarks at http://www.aldev.8m.com or at http://aldev.webjump.com

4.2 MySQL and other duplicate RDBMSes

MySQL is another open−source SQL server, but it does not support transactions. It is suitable for very small databases and does not support advanced SQL functionalities. Whereas PostgreSQL is an enterprise strength database supporting transactions and almost all SQL constructs. PostgreSQL is much more advanced than commercial databases like Oracle, Sybase and Informix. PostgreSQL supports very advanced locking mechanisms and many more advanced features which are not available in commercial database systems!!

In near future development of MySQL will be dropped, since MySQL is duplicate product working towards ANSI SQL. We would take the most advanced and mature open−source SQL server and drop all others as we do not have lots of time (to deal with multiple RDBMSes)!! In fact, you do not have time to deal with just one powerful SQL server like PostgreSQL! And all the MySQL users will be migrated to PostgreSQL. Also MySQL is a 'quasi−commercial' product unlike PostgreSQL which is open−source and there is no license fee. There is no need for another SQL database system as PostgreSQL is already here in this world!!

Duplicate products like MySQL confuse the user base and causes division of resources. For a **"NEAR PERFECT"** system there must be **only one** system and everybody in the world must work on it!! Duplicate products cause more harm than good and hence division of resources must be strongly discouraged. This already happened in case of commercial database systems like Oracle, Sybase, Informix and MS SQL server which caused splintering of user base and often they are incompatible. ***I want put the source code of SQL server under your control!!!***

You do not need hundreds of database systems, all you need is just one best database server which happens to be 'PostgreSQL'.

WARNING: It is possible to create infinite number of database systems for a given specification like ANSI SQL!!

Features which are missing in MySQL and which PostgreSQL supports are −

- Transactions
- Stored Procedures
- Triggers (update, insert and delete)
- Object oriented databases
- Advanced locking systems, concurrency management under multi−user, mutli−transactions environment
- Sub−queries
- Server−side cursors
- Query caching
- Locking of databases

- Better table join supports (JOIN, UNION, MINUS, INTERSECT, outer join)
- And many more advanced features − too numerous to list here.

MySQL is at http://www.tcx.se

4.3 Limitations of MySQL

PostgreSQL should be compared with systems like Oracle, both are really true ACID compliant robust systems developed over a very long time. It is very much wrong to compare MySQL with Oracle or MySQL with PostgreSQL. For more details read Why Not MySQL. Hence, it will be a very serious mistake to replace Oracle with MySQL!! If you want to replace Oracle then consider PostgreSQL.

---

5. Where to get it ?

You can buy Redhat Linux CDROM, Debian Linux CDROM or Slackware Linux CDROM which already contains the PostgreSQL in package form (both source code and binaries) from :

- Linux System Labs Web site: http://www.lsl.com/ (7 U.S. dollars)
- Cheap Bytes Inc Web site: http://www.cheapbytes.com/ (7 U.S. dollars)
- Debian Main Web site : http://www.debian.org/vendors.html

PostgreSQL organisation is also selling 'PostgreSQL CDROM' which contains the complete source code and binaries for many Unix operating systems as well as full documentation.

- PostgreSQL CDROM from main Web site at : http://www.postgresql.org 30 (U.S. dollars)

Binaries only distribution of PostgreSQL:

- The maintainer of PostgreSQL RPMs is Lamar Owen and is at lamar.owen@wgcr.org
- PostgreSQL source RPM and binaries RPM http://www.ramifordistat.net/postgres
- PostgreSQL source RPM and binaries RPM http://www.postgresql.org Click on "Latest News" and click on Redhat RPMs.
- PostgreSQL source RPM and binaries RPM http://www.redhat.com/pub/contrib/i386/ and ftp site is at ftp://ftp.redhat.com/pub/contrib/i386/
- Binaries site for Solaris, HPUX, AIX, IRIX, Linux : ftp://ftp.postgresql.org/pub/bindist

WWW Web sites:

- Primary Web site: http://www.postgresql.org/
- Secondary Web site: http://logical.thought.net/postgres95/
- http://www.itm.tu−clausthal.de/mirrors/postgres95/
- http://s2k−ftp.cs.berkeley.edu:8000/postgres95/
- http://xenium.pdi.net/PostgreSQL/
- http://s2k−ftp.cs.berkeley.edu:8000/postgres95/

The ftp sites are listed below :−

- Primary FTP: ftp://ftp.postgresql.org/pub

- Secondary FTP: ftp://ftp.chicks.net/pub/postgresql
- ftp://ftp.emsi.priv.at/pub/postgres/
- ftp://ftp.itm.tu−clausthal.de/pub/mirrors/postgres95
- ftp://rocker.sch.bme.hu/pub/mirrors/postgreSQL
- ftp://ftp.jaist.ac.jp/pub/dbms/postgres95
- ftp://ftp.luga.or.at/pub/postgres95
- ftp://postgres95.vnet.net:/pub/postgres95
- ftp://ftpza.co.za/mirrors/postgres
- ftp://sunsite.auc.dk/pub/databases/postgresql
- ftp://ftp.task.gda.pl/pub/software/postgresql
- ftp://xenium.pdi.net/pub/PostgreSQL

PostgreSQL source code is also available at all the mirror sites of sunsite unc (total of about 1000 sites around the globe). It is inside the Red Hat Linux distribution in /pub/contrib/i386/postgresql.rpm file.

- For list of mirror sites go to ftp://sunsite.unc.edu

6. PostgreSQL Quick−Installation Instructions

PostgreSQL is pronounced as Post−gres−cue−el (Postgres−QL) and **not** Postgre−es−cue−el.

This chapter will help you to install and run the database very quickly in less than 5 minutes.

6.1 Install and Test

Quick Steps to Install, Test, Verify and run PostgreSQL Login as root.

```
# cd /mnt/cdrom/RedHat/RPMS
# man rpm
# ls postgre*.rpm
# rpm −qpl postgre*.rpm | less (to see list of files)
# rpm −qpi postgre*.rpm (to see info of package)
# cat /etc/passwd | grep postgres
```

Note: If you see a 'postgres' user, you may need to backup and clean up the postgres home directory postgres and delete the unix user 'postgres' or rename the unix user 'postgres' to something like 'postgres2'. Install must be "clean slate"

```
# rpm −i postgre*.rpm (Must install all packages clients, devel, data
         and main for pgaccess to work )
# man chkconfig
# chkconfig −−add postgresql  (to start pg during booting)
# /etc/rc.d/init.d/postgresql start  (to start up postgres)
# man xhost
# xhost +  (To give display access for pgaccess)
# su − postgres
bash$ man createdb
bash$ createdb mydatabase
```

```
bash$ man psql
bash$ psql mydatabase
..... in psql press up/down arrow keys for history line editing or \s

bash$ export DISPLAY=<hostname>:0.0
bash$ man pgaccess
bash$ pgaccess mydatabase
```

Now you can start **rapidly BANGING away** SQL commands at psql or pgaccess.

```
bash$ cd /usr/doc/postgresql*
```

Here read all the FAQs, User, Programmer, Admin guides and tutorials.

6.2 PostgreSQL RPMs

See also *"Installation Steps"* from http://www.ramifordistat.net/postgres

The maintainer of PostgreSQL RPMs is Lamar Owen and is at lamar.owen@wgcr.org More details about PostgreSQL is at http://www.postgresql.org

6.3 Maximum RPM

Familiarize with RedHat RPM package manager to manage the PostgreSQL installations. Download the 'Maximum RPM' textbook from http://www.RPM.org look for the filename maximum−rpm.ps.gz And read it on linux using the gv command −

```
# gv maximum-rpm.ps.gz
```

There is also rpm2deb which converts the RPM packages to Debian linux packages.

6.4 Examples RPM

Examples are needed to do testing of various interfaces to PostgreSQL. Install the postgresql examples directory from −

- Linux cdrom − postgresql−*examples.rpm
- postgresql−*examples.rpm from http://www.aldev.8m.com and mirrors at webjump, angelfire, geocities, virtualave, bizland, theglobe, spree, infoseek, bcity, 50megs
- PostgreSQL source code tree postgresql*.src.rpm and look for examples, testing or tutorial directories

6.5 Testing PyGreSQL − Python interface

Install examples package, see Examples RPM and then do −

```
bash$ cd /usr/lib/pgsql/python
bash$ createdb thilo
bash$ psql thilo
thilo=> create table test (aa char(30), bb char(30) );
thilo=> \q
bash$ /usr/bin/python
>>> import _pg
>>> db = _pg.connect('thilo', 'localhost')
>>> db.query("INSERT INTO test VALUES ('ping', 'pong')")
>>> db.query("SELECT * FROM test")
eins|zwei
----+----
ping|pong
(1 row)
>>>CTRL+D
bash$
..... Seems to work - now install it properly
bash$ su - root
# cp /usr/lib/pgsql/python/_pg.so /usr/lib/python1.5/lib-dynload
```

6.6 Testing Perl − Perl interface

Install examples package, see Examples RPM and then do −

```
root# chown -R postgres.postgres /var/lib/pgsql/examples
bash$ cd /var/lib/pgsql/examples/perl5
bash$ perl ./example.pl
```

Note: If the above command does not work then do this. Gloabl var @INC should include the Pg.pm module in directory site_perl hence use −I option below

```
bash$ perl -I/usr/lib/perl5/site_perl/5.005/i386-linux-thread ./example.pl
```

.... You ran the perl which is accessing PostgreSQL database!!

Read the example.pl file for using perl interface.

6.7 Testing libpq, libpq++ interfaces

Install examples package, see Examples RPM and then do −

```
root# chown -R postgres.postgres /var/lib/pgsql/examples
bash$ cd /var/lib/pgsql/examples/libpq
bash$ gcc testlibpq.c -I/usr/include/pgsql -lpq
bash$ export PATH=$PATH:.
bash$ a.out
```

```
bash$ cd /var/lib/pgsql/examples/libpq++
bash$ g++ testlibpq0.cc -I/usr/include/pgsql -I/usr/include/pgsql/libpq++
-lpq++ -lpq -lcrypt
bash$ ./a.out  (Note: Ignore Error messages if you get any - as below)
> create table foo (aa int, bb char(4));
No tuples returned...
status = 1
Error returned: fe_setauthsvc: invalid name: , ignoring...
> insert into foo values ('4535', 'vasu');
No tuples returned...
status = 1
Error returned: fe_setauthsvc: invalid name: , ignoring...
> select * from foo;
aa   |bb   |
-----|-----|
4535 |vasu |
Query returned 1 row.
>
>CTRL+D
bash$
```

.... You ran direct C/C++ interfaces to PostgreSQL database!!

6.8 Testing Java interfaces

Install examples package, see Examples RPM and also install the following −

- Get JDK jdk−*glibc*.rpm from ftp://ftp.redhat.com/pub/contrib/i386 or from http://www.blackdown.org
- Get postgresql−jdbc−*.rpm ftp://ftp.redhat.com/pub/contrib/i386

```
root# chown -R postgres.postgres /var/lib/pgsql/examples
bash$ cd /var/lib/pgsql/examples/jdbc
bash$ echo $CLASSPATH
 --> Should show
CLASSPATH=/usr/lib/pgsql/jdbc7.0-1.2.jar:.:/home/java/jdk1.2.2/lib:/usr/lib/pgsql:/usr/lib/pgsql/classes.zip:/usr/lib/pgsql/pg.jar

with proper jdbc*.jar version numbers.
And the directories /usr/lib/pgsql and /usr/libjdk*/lib should contain *.jar files.

bash$ export CLASSPATH=/usr/lib/pgsql/jdbc7.0-1.2.jar:.:/home/java/jdk1.2.2/lib:/usr/lib/pgsql:/usr/lib/pgsql/classes.zip:/usr/lib/pgsql/pg.jar

Edit all psql.java file  and comment out the 'package' line.
bash$ javac psql.java
bash$ java psql jdbc:postgresql:template1 postgres < password >
[1] select * from pg_tables;
tablename       tableowner      hasindexes      hasrules
pg_type postgres        true    false   false
pg_attribute    postgres        true    false   false
[2]
CTRL+C
bash$
```

.... You ran direct Java interfaces to PostgreSQL database!

6.9 Testing ecpg interfaces

Install examples package, see Examples RPM and then do −

```
root# chown -R postgres.postgres /var/lib/pgsql/examples
bash$ cd /var/lib/pgsql/examples/ecpg
bash$ ecpg test1.pgc -I/usr/include/pgsql
bash$ cc test1.c -I/usr/include/pgsql -lecpg -lpq -lcrypt
bash$ createdb mm
bash$ ./a.out
```

.... You ran Embedded "C"−SQL to PostgreSQL database!

6.10 Testing SQL examples − User defined types and functions

Install examples package, see Examples RPM and then do −

```
root# chown -R postgres.postgres /var/lib/pgsql/examples
bash$ cd /var/lib/pgsql/examples/sql
Under-development..
```

6.11 Testing Tcl/Tk interfaces

Example of Tcl/Tk interfaces is pgaccess program. Read the file /usr/bin/pgaccess using an editor −

```
bash$ view /usr/bin/pgaccess
bash$ export DISPLAY=<hostname of your machine>:0.0
bash$ createdb mydb
bash$ pgaccess mydb
```

6.12 Testing ODBC interfaces

1. Get the win32 pgsql odbc driver from http://www.insightdist.com/psqlodbc/
2. See also /usr/lib/libpsqlodbc.a

6.13 Testing MPSQL Motif−worksheet interfaces

Get the RPMs from http://www.mutinybaysoftware.com

6.14 Verification

To verify the top quality of PostgreSQL, run the Regression test package :− Login as root −

```
# rpm -i postgresql*test.rpm
And see README file or install the source code tree which has regress directory
# rpm -i postgresql*.src.rpm
# cd /usr/src/redhat/SPECS
# more postgresql*.spec   (to see what system RPM packages you need to
install)
# rpm -bp postgresql*.spec  (.. this will prep the package)

Regression test needs the Makefiles and some header files like *fmgr*.h
which can be built by -
# rpm --short-circuit -bc postgresql*.spec ( .. use short circuit to
bypass!)
Abort the build by CTRL+C, when you see 'make -C common  SUBSYS.o'
By this time configure is successful and all makefiles and headers
are created. You do not need to proceed any further
# cd /usr/src/redhat/BUILD
# chown -R postgres postgresql*
# su - postgres
bash$ cd /usr/src/redhat/BUILD/postgresql-6.5.3/src/test/regress
bash$ more README
bash$ make clean; make all runtest
bash$ more regress.out
```

6.15 Emergency Bug fixes

Sometimes emergency bug fix patches are released after the GA release of PostgreSQL. You can apply these optional patches depending upon the needs of your application. Follow these steps to apply the patches − Change directory to postgresql source directory

```
# rpm -i postgresql*.src.rpm
# cd /usr/src/postgresql6.5.3
# man patch
# patch -p0 < patchfile
# make clean
# make
```

The patch files are located in

- PostgreSQL patches : ftp://ftp.postgresql.org/pub/patches

7. Quick Start Guide

Refer also to Quick Installation chapter.

7.1 Creating, Dropping, Renaming Database

You can use the user friendly GUI called 'pgaccess' to create and drop databases, or you can use the command line 'psql' utility.

```
If you are logged in as root, switch user to 'postgres' :
# xhost +  (To give display access for pgaccess)
# su − postgres
bash$ man createdb
bash$ createdb mydatabase
bash$ man psql
bash$ psql mydatabase
..... in psql press up/down arrow keys for history line editing or \s

bash$ export DISPLAY=<hostname>:0.0
bash$ man pgaccess
bash$ pgaccess mydatabase
```

Now you can start **rapidly BANGING away** SQL commands at psql or pgaccess !!

To drop the database do :

```
bash$ man dropdb
bash$ man destroydb   (for older versions of pgsql)
bash$ dropdb <dbname>
```

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

```
> drop database <dbname>
```

To rename a database see Backup and Restore

7.2 Creating, Dropping users

To create new users, login as unix user 'postgres'. You can use user friendly GUI tool called 'pgacess' to create, drop users.

```
bash$ man pgaccess
bash$ pgaccess <database_name>
```

and click on "Users" tab and then click Object|New or Object|Delete

You can also use command line scripts. Use the shell script called 'createuser' which invokes psql

```
bash$ man createuser
bash$ createuser <username>
bash$ createuser -h host -p port -i userid <username>
```

To drop a postgres user, use shell script 'destroyuser' −

```
bash$ man dropuser
bash$ man destroyuser  (older versions of pgsql)
bash$ destroyuser
```

7.3 Creating, Dropping Groups

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

```
bash$ su - postgres
bash$ psql <database_name>
..... in psql press up/down arrow keys for history line editing or \s

psql=> insert into pg_group (groname, grosysid, grolist)
psql=> values ('posthackers', '1234', '{5443, 8261}' );
INSERT 58224
psql=> grant insert on foo to group posthackers;
CHANGE
psql=>
```

The fields in **pg_group** are: **groname** The group name. This name should be purely alphanumeric; do not include underscores or other punctuation.

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

grolist The list of **pg_user** IDs that belong in the group. This is an int4[].

To drop the group:

```
bash$ su - postgres
bash$ psql <database_name>
..... in psql press up/down arrow keys for history line editing or \s

psql=> delete from pg_group where groname = 'posthackers';
```

7.4 Create, Edit, Drop a table

You can use user friendly GUI tool 'pgaccess' or command line tool 'psql' to create, edit or drop a table in a database.

```
bash$ man pgaccess
bash$ pgaccess <database_name>
```

Click on Table | New | Design buttons.

```
bash$ man psql
bash$ psql <database_name>
..... in psql press up/down arrow keys for history line editing or \s
```

At psql prompt, give standard SQL statements like 'create table', 'alter table' or 'drop table' to manipulate the tables.

7.5 Create, Edit, Drop records in a table

You can use user friendly GUI tool 'pgaccess' or command line tool 'psql' to create, edit or drop records in a database table.

```
bash$ man pgaccess
bash$ pgaccess <database_name>
```

Click on Table | < pick a table > | Open buttons.

```
bash$ man psql
bash$ psql <database_name>
..... in psql press up/down arrow keys for history line editing or \s
```

At psql prompt, give standard SQL statements like 'insert into table_name', 'update table_name' or 'delete from table_name' to manipulate the tables.

7.6 Switch active Database

You can use user friendly GUI tool 'pgaccess' or command line tool 'psql' to switch active database.

```
bash$ man pgaccess
bash$ pgaccess <database_name>
```

Click on Database | Open buttons.

```
bash$ man psql
bash$ psql <database_name>
..... in psql press up/down arrow keys for history line editing or \s

psql=> connect <database_name> <user>
```

7.7 Backup and Restore database

PostgreSQL provides two utilities to back up your system: **pg_dump** to backup individual databases, and **pg_dumpall** to back up all the databases in just one step.

```
bash$ su - postgres
bash$ man pd_dump
bash$ pd_dump <database_name> > database_name.pgdump

To dump all databases -
bash$ man pg_dumpall
bash$ pg_dumpall -o > db_all.out

To reload (restore) a database dumped with pg_dump:
bash$ cat database_name.pgdump | psql <database_name>

To reload (restore) all databases dumped with pg_dumpall:
bash$ psql -e template1 < db_all.out
```

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

WARNING: Every database should be backed up on a regular basis. Since PostgreSQL manages its own files in the file sysetem, it is not advisable to rely on system backups of your file system for your database backups; there is no guarantee that the files will be in an usable, consistent state after restoration.

BACKUP LARGE DATABASES: Since Postgres allows tables larger than the maximum file size on your system, it can be problematic to dump the table to a file, because the resulting file likely will be larger than the maximum size allowed by your system. As **pg_dump** writes to **stdout**, you can just use standard unix tools to work around this possible problem − use compressed dumps.

```
bash$ pg_dump <database_name> | gzip > filename.dump.gz
Reload with :
bash$ createdb <database_name>
bash$ gunzip -c filename.dump.gz | psql <database_name>
Or
bash$ cat filename.dump.gz | gunzip | psql <database_name>
Use split:
bash$ pg_dump <database_name> | split -b 1m - filename.dump.
Note: There is a dot (.) after filename.dump in the above command!!
You can reload with:
bash$ man createdb
bash$ createdb <database_name>
bash$ cat filename.dump.* | pgsql <database_name>
```

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

Backup LARGE Objects: Large objects are not handled by pg_dump. The directory *contrib/pg_dumplo* of the Postgres source tree contains a program that can do that.

FILESYSTEM BACKUP : You can use the linux OS tools and commands to backup the entire database. But you **must completely** shutdown the postgresql database server before doing backup or restore with this method. The filesystem backup or restore may be **2 to 3 times faster** than the pg_dump command, but only disadvantage is that you **must completely** shutdown the database server. It is very highly recommended that you use backup and restore tools like Arkeia, Bru which are given in Mic−Lin analogy list sub−heading "Backup and Restore Utility" at http://aldev.8m.com and mirror sites are at webjump, angelfire, geocities, virtualnet, bizland, theglobe, spree, infoseek, bcity, 50megs .The OS commands to use are −

```
bash$ man tar
bash$ tar −cvf backup.tar /usr/local/pgsql/data
or using compression
bash$ tar −zcvf backup.tgz /usr/local/pgsql/data
```

INCREMENTAL BACKUP : This is in todo list and will appear in future release of PostgreSQL.

7.8 Security of database

See the chapter on PostgreSQL Security.

7.9 Online help

It is very important that you should know how to use online help facilities of PostgreSQL, since it will save you lot of time and provides very quick access to information.

See the online man pages on various commands like createdb, createuser, etc..

```
bash$ man createdb
```

See also online help of psql, by typing \h at psql prompt

```
bash$ psql mydatabase
psql> \h

Tip: In psql press up/down arrow keys for history line editing or \s
```

7.10 Creating Triggers and Stored Procedures

To create triggers or stored procedures, First run 'createlang' script to install 'plpgsql' in the particular database you are using. If you want by default then install it in 'template1' and subsequent created databases will be clones of template1. See 'createlang' web page in User guide at /usr/doc/postgresql−7.0.2/user/index.html.

```
bash$ man createlang
bash$ createdb mydb
bash$ export PGLIB=/usr/lib/pgsql
bash$ createlang plpgsql mydb
bash$ createlang plpgsql template1
```

See also the trigger, stored procedures examples in Examples RPM . One sample code from examples RPM:

```
create function tg_pfield_au() returns opaque as '
begin
    if new.name != old.name then
        update PSlot set pfname = new.name where pfname = old.name;
    end if;
    return new;
end;
' language 'plpgsql';

create trigger tg_pfield_au after update
    on PField for each row execute procedure tg_pfield_au();
```

Another trigger example sample code:

```
create trigger check_fkeys_pkey_exist
        before insert or update on fkeys
        for each row
        execute procedure
        check_primary_key ('fkey1', 'fkey2', 'pkeys', 'pkey1', 'pkey2');
```

You must also install the TEST package − postgresql−test−7.0.2−2.rpm and read the example sql scripts at − /usr/lib/pgsql/test/regress/sql

To see the list of triggers in database do −

```
bash$ psql mydb
psql=> \?
psql=> \dS
psql=> \d pg_trigger
psql=> select tgname from pg_trigger order by tgname;
```

To see the list of functions and stored procedures in database do −

```
bash$ psql mydb
psql=> \?
psql=> \dS
psql=> \d pg_proc
psql=> select proname, prosrc from pg_proc order by proname;
psql=> \df
```

7.11 PostgreSQL Documentation

More questions, read the fine manuals of PostgreSQL which are very extensive. PostgreSQL documentation is distributed with package. See the 'User's Guide', 'Programmer's Guide', 'Administrator's Guide' and other manuals. The release docs are at http://www.postgresql.org/users−lounge/docs.

8. Performance Tuning of PostgreSQL server

Generally database server is standalone box connected to network. Since database server is the only unix process which runs on the CPU, you can do several optimizations to speed up the server.

8.1 OS Tuning for Database server

To get more bang for a given CPU processing power, do the following:−

- Recompile linux kernel to make it small and lean. Remove items which are not used. See kernel howto at http://www.linuxdoc.org/HOWTO/Kernel−HOWTO.html
- Turn off unneccessary unix processes − on linux/unix systems run chkconfig

```
bash$ su - root
bash# man chkconfig
bash# chkconfig --help
bash# chkconfig --list | grep on | less
From the above list, turn off the processes you do not want to start automatically -
bash# chkconfig --level 0123456 <service name> off
Next time when the machine is booted these services will not be started.
Now, shutdown the services manually which you just turning off.
bash# cd /etc/rc.d/init.d
bash# ./<service name> stop
```

- Do not run any other application processes which are unnecessary.
- Do not leave X−Window running unattended. Because X−window processes consume memory, CPU load and can be a serious security hole from outside attacks. The X−window managers generally used are KDE, GNOME, CDE, XDM and others. You must exit the X−window immediately after using and most of the time you should see command line console login prompt on the database server machine.

8.2 Tuning Database server process

General tuning tips:

- Indices can speed up queries. The **explain** command allows you to see how PostgreSQL is interpreting your query, and which indices are being used.
- Use the **cluster** command to group data in base tables to match an index. See the **man cluster(1)** manual page for more details.
- If you are doing a lot of inserts, consider doing them in a large batch using the **copy** command. This is much faster than individual inserts.
- Statements not in a **begin work/commit** transaction block are considered to be in their own transaction. Consider performing several statements in a single transaction block. This reduces the transaction overhead. Also consider dropping and recreating indices when making large data changes.
- It is suggested that you purchase the "Performance Tuning guide" and tuning support from PostgreSQL Corp..

Specialized tuning tips:

- Internal tuning of PostgreSQL is a complex topic. You need a sound knowledge of source code and internals of postgresql. It is strongly recommended that only professionals attempt specialized tuning tips given below:
- You can disable **fsync()** by starting the postmaster with a **−o −F** option. This will prevent **fsync()** from flushing to disk after every transaction. But there is risk of losing data due to power/media failure. You can reduce the risk of losing data due to power failure by having the APC UPS (Uninterrupted Power Supply) and media failures by disk RAID systems (Antares−Sparc−Raid system, Software−Raid system, Old−Software−Raid system, Root−Raid system, Boot−Root−Raid system) to gaurd against media failures.
- Use the postmaster **−B** option to increase the number of shared memory buffers used by the back−end processes. If you make this parameter too high, the postmaster may not start up because you've exceeded your kernel's limit on shared memory space. Each buffer is 8K and the default is 64 buffers.
- Use the back−end **−S** option to increase the maximum amount of memory used by each backend process for temporary sorts. The **−S** value is measured in kilobytes, and the default is 512 (i.e., 512K). It is unwise to make this value too large, or you may run out of memory when query invokes several concurrent sorts.

---

9. PostgreSQL Supports Extremely Large Databases greater than 200 Gig

PostgreSQL is already used by many companies supporting large databases. The following techniques are suggested :

9.1 CPU types − 32−bit or 64−bit

Performance of 32−bit cpu machines will decline rapidly when the database size exceeds 5 GigaByte. You can run 30 gig database on 32−bit cpu but the performance will be degraded. Machines with 32−bit cpu imposes a limitation of 2 GB on RAM, 2 GB on file system sizes and other limitations on the operating system. Use the special filesystems for linux made by SGI, IBM or HP or ext3−fs to support file−sizes greater than 2 GB on 32−bit linux machines.

For extremely large databases, it is strongly advised to use 64−bit machines like Digital Alpha cpu, Sun Ultra−sparc 64−bit cpu, Silicon graphics 64−bit cpu, Intel Merced IA−64 cpu, HPUX 64bit machines or IBM 64−bit machines. Compile PostgreSQL under 64−bit cpu and it can support huge databases and large queries. Performance of PostgreSQL for queries on large tables and databases will be several times faster than PostgreSQL on 32−bit cpu machines. Advantage of 64−bit machines are that you get very large memory addressing space and the operating system can support very large file−systems, provide better performance with large databases, support much larger memory (RAM), have more capabilities etc..

9.2 Multiple CPUs

For large databases it is recommended that you use SMP boxes which have 4, 16 or 32 CPUs. Alternatively, you can use 4 or 5 single CPU boxes and you can partition the database into 4 or 5 seperate databases and each database running on a seperate box. Each CPU will be connected with fast NIC (100MBits) ethernet card. For example − if you have 200 tables in a database, you can distribute 200 tables to 4 database each having 50 tables. In this way, you are distributing the load evenly among 4 seperate machines. This is a cheaper alternative to 4−way CPU box. You would use **'Queries across multiple databases'**, NFS mounts in LAN, 'CREATE VIEW' for read−only tables to accomplish this task. And each CPU "can see" all the databases i.e all the 200 tables. In future PostgreSQL may provide support for **'Queries across multiple databases'** (already in the TODO list), which may appear in upcoming version 7.1. For example, queries across multiple databases using aliases a, b for table names can be like −

```
select a.col1, a.col2, b.col4, b.col7
from
        database1.my_tablea a, database2.my_tableb b
where
        a.col1 = b.col3 and
        a.col4 = b.col9;

update my_tablea
set
        col1 =  b.col2
from
        database1.my_tablea a, database2.my_tableb b
where
        a.col4 = b.col9;
```

9.3 Replication Server

Replication server for large enterprises/businesses is available at http://www.erserver.com and from http://www.pgsql.com. The support is sold ($$$$s) commercially by PostgreSQL Inc. You use replication server to provide redundancy and high availability. Replication server is a complex, sophisticated product.

10. How can I trust PostgreSQL ? Regression Test Package builds customer confidence

Thanks to *"Laws of Physics"*, it is possible to **SCIENTIFICALLY** verify whether PostgreSQL is working as per ISO/ANSI SQL specifications. To validate PostgreSQL, regression test package (src/test/regress) is included in the distribution. Regression test package will verify the standard SQL operations as well as the extensibility capabilities of PostgreSQL. The test package already contains hundreds of SQL test programs.

You should use the computer's high−speed power to validate the PostgreSQL, instead of using human brain power. Computers can carry out software regression tests millions or even billions of times faster than humans can. Modern computers can run billions of SQL tests in a very short time. In the near future the speed of computer will be several zillion times faster than human brain! Hence, it makes sense to use the power of computer to validate the software.

You can add more tests just in case you need to, and can upload to the primary PostgreSQL web site if you feel that it will be useful to others on internet. Regression test package helps build customer confidence and trust in PostgreSQL and facilitates rapid deployment of PostgreSQL on production systems.

Regression test package can be taken as a **"VERY SOLID"** technical document mutually agreed upon between the developers and end−users. PostgreSQL developers extensively use the regression test package during development period and also before releasing the software to public to ensure good quality.

Capablilities of PostgreSQL are directly reflected by the regression test package. If a functionality, syntax or feature exists in the regression test package then it is supported, and all others which are NOT listed in the package MAY not be supported by PostgreSQL!! You may need to verify those and add it to regression test package.

---

11. Security of Database

Database security is addressed at several levels:

- Database 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 an local UNIX socket, not via TCP/IP sockets. The back−end must be started with the −i option to allow nonlocal clients to connect.
- Client connections can be restricted by IP address and/or username via the **pg_hba.conf** file in **$PG_DATA**.
- Client connections may be authenticated via other external packages.
- Each user in Postgres is assigned an 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 priveleges.

11.1 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 an 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)** (man 5 pg_hba.conf) 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.

11.2 Host−Based Access Control

Host−based access control is the name for the basic controls PostgreSQL exercises on what clients are allowed to access a database and how the users on those clients must authenticate themselves. Each database system contains a file named **pg_hba.conf**, in its **$PGDATA** directory, which controls who can connect to each database. Every client accessing a database must be covered by one of the entries in **pg_hba.conf**. Otherwise all attempted connections from that client will be rejected with a **"User authentication failed"** error message.

See online man page of **pg_hba.conf(5)** (man 5 pg_hba.conf).

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

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

```
local database authentication method
```

where

database specifies the database that this record applies to. The value **all** specifies that it applies to all databases.

authentication method specifies the method an user must use to authenticate themselves when connecting to that database using Unix domain sockets. The different methods are described below.

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

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

The **TCP/IP** address is *logically and'ed* to both the specified TCP/IP mask and the TCP/IP address of the connecting client. If the two resulting values are equal then the record is used for this connection. If a connection matches more than one record then the earliest one in the file is used. Both the TCP/IP address and the TCP/IP mask are specified in dotted decimal notation. If a connection fails to match any record then the reject authentication method is applied (see Authentication Methods).

11.3 Authentication Methods

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

- **trust** The connection is allowed unconditionally.
- **reject** The connection is rejected unconditionally.
- **crypt** The client is asked for a password for the user. This is sent encrypted (using crypt(3)) and compared against the password held in the pg_shadow table. If the passwords match, the connection is allowed.
- **password** The client is asked for a password for the user. This is sent in clear and compared against the password held in the **pg_shadow** table. If the passwords match, the connection is allowed. An optional password file may be specified after the password keyword which is used to match the supplied password rather than the **pg_shadow** table. See **pg_passwd**.

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

- **krb4** Kerberos V4 is used to authenticate the user.
- **krb5** Kerberos V5 is used to authenticate the user.
- **ident** The ident server on the client is used to authenticate the user (RFC 1413). An optional map name may be specified after the **ident** keyword which allows ident user names to be mapped onto Postgres user names. Maps are held in the file **$PGDATA/pg_ident.conf**.

Here are some examples:

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

11.4 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 SQL **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.

11.5 Secure TCP/IP Connection via SSH

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

The documentation for **ssh** provides most of the information to get started. Please refer to http://www.heimhardt.de/htdocs/ssh.html for better insight. A step−by−step explanation can be done in just two steps.

Running a secure tunnel via ssh: A step−by−step explanation can be done in just two steps.

- Establish a tunnel to the back−end machine, like this:

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

- The first number in the **−L** argument, **3333**, is the port number of your end of the tunnel. The second number, **5432**, is the remote end of the tunnel −− the port number your backend is using. The name or the address in between the port numbers belongs to the server machine, as does the last argument to **ssh** that also includes the optional user name. Without the user name, **ssh** will try the name you are currently logged on as on the client machine. You can use any user name the server machine will accept, not necessarily those related to postgres.
- Now that you have a running **ssh** session, you can connect a postgres client to your local host at the port number you specified in the previous step. If it's **psql**, you will need another shell because the shell session you used in step 1 is now occupied with **ssh**.

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

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

11.6 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.

```
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.
```

Inquiries regarding your Kerberos should be directed to your vendor or MIT Project Athena. Note that FAQLs (Frequently−Asked Questions Lists) are periodically posted to the Kerberos mailing list (send mail to subscribe), and USENET news group.

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(also, see Table below):

- 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: Kerberos Parameter Examples
------------------------------------------------------
Parameter      Example
------------------------------------------------------
user           frew@S2K.ORG
user           aoki/HOST=miyu.S2K.Berkeley.EDU@S2K.ORG
host           postgres_dbms/ucbvax@S2K.ORG
------------------------------------------------------
```

---

12. GUI FrontEnd Tool for PostgreSQL (Graphical User Interface)

Web browser will be the most popular GUI front−end in the future. It is recommended that you migrate all of your "legacy" Windows 95/NT applications to Web−based application.

You should use Web−Application Servers like Enhydra (Java based) or Zope (Python based) or OpenACS .

Best web−scripting (and compiling) language is PHP+Zend compiler **PHP** is extremely powerful as it combines the power of Perl, Java, C++, Javascript into one single language and it runs on all OSes − unixes and Windows NT/95.

The best tools in the order of preference are −

- Enhydra at Enhydra plus Borland Java JBuilder for Linux http://www.inprise.com
- Zope at Zope
- OpenACS at OpenACS
- PHP script and Zend compiler at PHP+Zend compiler
- X−Designer supports C++, Java and MFC http://www.ist.co.uk/xd

- Qt for Windows95 and Unix at http://www.troll.no and ftp://ftp.troll.no
- Code Crusader is on linux cdrom, freeware based on MetroWorks Code Warrior http://www.kaze.stetson.edu/cdevel/code_crusader/about.html
- Code Warrior from MetroWorks http://www.metrowerks.com
- GNU Prof C++ IDE from (Redhat) Cygnus http://www.cygnus.com
- Borland C++ Builder for Linux http://www.inprise.com
- Borland Java JBuilder for Linux http://www.inprise.com

Language choices in the order of preference are −

1. Java but its programs run very slow and has license fees. C++ is **5 times faster** than Java!!
2. Python (Powerful object oriented scripting language).
3. PHP Web server scripting, HTML, DHTML with Javascrpt client scripting and Java−Applets.
4. Perl scripting language using Perl−Qt or Perl−Tk Perl Database Interface
5. Omnipresent and Omnipotent language C++ (GNU g++):
   - Fast CGI(written in GNU C++) with Javascript/Java−Applets as Web−GUI−frontend.
   - GNU C++ and QtEZ or QT
   - GNU C++ with Lesstiff or Motif.

There are other tools available − PostgreSQL has Tcl/Tk interface library in the distribution called 'pgTcl'. There is an IDE (integrated development environment) for Tcl/Tk called SpecTcl.

- Lesstiff Motif tool ftp://ftp.redhat.com/pub/contrib/i386/lesstiff*.rpm
- Vibe Java/C++ is at http://www.LinuxMall.com/products/00487.html
- JccWarrior ftp://ftp.redhat.com/pub/contrib/i386/jcc*.rpm
- Tcl/Tk http://www.scriptics.com
- Object oriented extension of Tcl called INCR at http://www.tcltk.com
- Visual TCL site http://www.neuron.com
- Visual TCL Redhat rpm at ftp://ftp.redhat.com/pub/contrib/i386/visualtcl*.rpm
- http://sunscript.sun.com/
- http://sunscript.sun.com/TclTkCore/
- ftp://ftp.sunlabs.com/pub/tcl/tcl8.0a2.tar.Z
- Java FreeBuilder ftp://ftp.redhat.com/pub/contrib/i386/free*.rpm
- SpecTCL ftp://ftp.redhat.com/pub/contrib/i386/spec*.rpm
- Java RAD Tool for PostgreSQL Kanchenjunga http://www.man.ac.uk/~whaley/kj/kanch.html
- Applixware Tool http://www.redhat.com
- XWPE X Windows Programming Environment http://www.identicalsoftware.com/xwpe/ or at http://www.rpi.edu/~payned/xwpe/ ftp://ftp.redhat.com/pub/contrib/i386/xwpe*.rpm
- XWB X Windows Work Bench ftp://ftp.redhat.com/pub/contrib/i386/xwb*.rpm
- NEdit ftp://ftp.redhat.com/pub/contrib/i386/nedit*.rpm

You can also use Borland C++ Builder, Delphi, Borland JBuilder, PowerBuilder on Windows95 connecting to PostgreSQL on unix box through ODBC/JDBC drivers.

---

13. Interface Drivers for PostgreSQL

13.1 ODBC Drivers for PostgreSQL

ODBC stands for 'Open DataBase Connectivity' established by Microsoft, is a popular standard for accessing information from various databases from different vendors. Applications written using the ODBC drivers are guaranteed to work with various databases like PostgreSQL, Oracle, Sybase, Informix etc..

- PostODBC is already included in the distribution. See main web site http://www.postgresql.org. It is included on the PostgreSQL CDROM.
- Open source code ODBC project is at http://www.iodbc.org
- http://www.openlinksw.com Open Link Software Corporation is selling ODBC for PostgreSQL and other databases. Open Link also is giving away free ODBC (limited seats) check them out.
- Insight ODBC for PostgreSQL http://www.insightdist.com/psqlodbc This is the official PostODBC site.
- FreeODBC package http://www.ids.net/~bjepson/freeODBC/ This is a free of cost version of ODBC.
- ODBC 32 Explorer for PostgreSQL http://members.nbci.com/anhr

13.2 UDBC Drivers for PostgreSQL

UDBC is a static version of ODBC independent of driver managers and DLL support, used to embed database connectivity support directly into applications.

- http://www.openlinksw.com Open Link Software Corporation is selling UDBC for PostgreSQL and other databases. Open Link also is giving away free UDBC (limited seats) check them out.

13.3 JDBC Drivers for PostgreSQL

JDBC stands for 'Java DataBase Connectivity'. Java is a platform independent programming language developed by Sun Microsystems. Java programmers are encouraged to write database applications using the JDBC to facilitate portability across databases like PostgreSQL, Oracle, informix, etc. If you write Java applications you can get JDBC drivers for PostgreSQL from the following sites:

JDBC driver is already included in the PostgreSQL distribution in postgresql−jdbc*.rpm.

- http://www.demon.co.uk/finder/postgres/index.html Sun's Java connectivity to PostgreSQL
- ftp://ftp.ai.mit.edu/people/rst/rst−jdbc.tar.gz
- http://www.openlinksw.com Open Link Software Corporation is selling JDBC for PostgreSQL and other databases. Open Link also is giving away free JDBC (limited seats) check them out.
- JDBC UK site http://www.retep.org.uk/postgres
- JDBC FAQ site http://eagle.eku.edu/tools/jdbc/faq.html

The JDBC home, guide and FAQ are located at −

- JDBC HOME http://splash.javasoft.com/jdbc
- JDBC guide http://www.javasoft.com/products/jdk/1.1/docs/guide/jdbc
- JDBC FAQ http://javanese.yoyoweb.com/JDBC/FAQ.txt

See the section − Testing Java PostgreSQL interface

13.4 Java for PostgreSQL

Java programmers can find these for PostgreSQL very useful.

- ftp://ftp.redhat.com/pub/contrib/i386 and see postgresql−jdbc−*.rpm
- http://www.blackdown.org

See the section −

14. Perl Database Interface (DBI) Driver for PostgreSQL

14.1 Perl interface for PostgreSQL

PERL is an acronym for 'Practical Extraction and Report Language'. Perl is available on each and every operating system and hardware platform in the world. You can use Perl on Windows95/NT, Apple Macintosh iMac, all flavors of Unix (Solaris, HPUX, AIX, Linux, Irix, SCO etc..), mainframe MVS, desktop OS/2, OS/400, Amdahl UTS and many others. Perl runs **EVEN** on many unpopular or generally−unknown operating systems and hardware!! So do not be surprised if you see perl running on a very rarely used operating system. You can imagine the vast extent of the user base and developer base of Perl. Perl language has a very long life just like "C" language, and Perl will be in use for **thousands of years** in the future! Perl runs **10 times** faster than Java and sometimes faster than even "C". Java is a very complex system with virtual machine and interpreter which makes it extremely slow, unstable and unreliable. Perl is very simple, fast and object oriented.

Perl interface for PostgreSQL is included in the distribution of PostgreSQL. Check in src/pgsql_perl5 directory.

- Pgsql_perl5 contact Email: E.Mergl@bawue.de
- Perl Home page : http://www.perl.com/perl/index.html
- Perl tutorial, look for Tutorial title at : http://reference.perl.com/
- Perl FAQ is at : http://www.yahoo.com/Computers_and_Internet/Programming_Languages/Perl/
- First get Mother of all Perl Modules from http://www.perl.com/CPAN/modules/by−module/CPAN and type '/usr/bin/cpan', 'man CPAN', see thousands of modules http://www.perl.com/CPAN−local/modules/by−module.
- Perl GUI User Interfaces Perl−Qt rpm : ftp://ftp.redhat.com/pub/contrib/i386 and look for PerlQt−1.06−1.i386.rpm
- Perl GUI User Interfaces Perl−Qt : http://www.accessone.com/~jql/perlqt.html
- Perl GUI User Interfaces Perl−XForms : ftp://ftp.redhat.com/pub/contrib/i386 and look for Xforms4Perl−0.8.4−1.i386.rpm
- Perl GUI User Interfaces Perl−Tk : ftp://ftp.redhat.com/pub/contrib/i386
- Perl GUI kits : http://reference.perl.com/query.cgi?ui
- Perl Database Interfaces : http://reference.perl.com/query.cgi?database
- Perl to "C" translator : http://www.perl.com/CPAN−local/modules/by−module/B/ and look for Compiler−a3.tar.gz
- Compile Perl to to executable. Perl2Exe is a command line utility for converting perl scripts to executable files http://www.indigostar.com/perl2exe.htm
- Bourne shell to Perl translator : http://www.perl.com/CPAN/authors/id/MERLYN/sh2perl−0.02.tar.gz
- awk to Perl a2p and sed to Perl s2p is included with the PERl distribution.
- See also the newsgroups for PERL at comp.lang.perl.*

14.2 Perl Database Interface DBI

WHAT IS DBI ?

The Perl Database Interface (DBI) is a database access Application Programming Interface (API) for the Perl Language. The Perl DBI API specification defines a set of functions, variables and conventions that provide a consistent database interface independent of the actual database being used. The Database Drivers (Perl DBI) initiative has standardized the interface to a number of commercial database engines, so that you can move from, say, Oracle to PostgreSQL with a minimum of effort.

DBD driver for PostgreSQL

Before you install DBD PostgreSQL (Driver) you must install DBI. Get DBI driver from

- First get Mother of all Perl Modules from http://www.perl.com/CPAN/modules/by−module/CPAN and type '/usr/bin/cpan', 'man CPAN', see thousands of modules http://www.perl.com/CPAN−local/modules/by−module.
- DBI Modules http://www.perl.com/CPAN−local/modules/by−module/DBI.
- DBI Modules http://www.symbolstone.org/technology/perl/DBI
- DBI FAQ http://www.symbolstone.org/technology/perl/DBI/doc/faq.html
- References for Perl DBI http://www.symbolstone.org/technology/perl/DBI
- DBI Mailing Lists http://www.fugue.com/dbi
- Perl Database references http://www.perl.com/reference/query.cgi?section=database
- Download DBI rpm (Caution: may be old version) http://rpmfind.net/linux/rpm2html/search.php?query=DBI

Get DBD−Pg from below

- First get Mother of all Perl Modules from http://www.perl.com/CPAN/modules/by−module/CPAN and type '/usr/bin/cpan', 'man CPAN', see thousands of modules http://www.perl.com/CPAN−local/modules/by−module.
- DBD Modules http://www.perl.com/CPAN−local/modules/by−module/DBD. and look for DBD−pg files or at DBD.
- Comprehensive Perl Archive Network CPAN http://www.perl.com/CPAN Go here select 'Database' (located above Search box) and click on 'Go' button.
- Pre−compiled package for Windows NT/2000 is available at http://www.edmund−mergl.de/export/DBD−Pg.zip
- Download DBD rpm (Caution: may be old version) http://rpmfind.net/linux/rpm2html/search.php?query=DBD
- Perl Modules (thousands of them) http://www.perl.com/CPAN−local/modules/by−module.

Technical support for DBI

- Send comments and bug−reports to and include the output of perl −v, and perl −V, the version of PostgreSQL, the version of DBD−Pg, and the version of DBI in your bug−report. E.Mergl@bawue.de

DBI Documents

There are a few information sources on DBI.

POD documentation: PODs are chunks of documentation usually embedded within perl programs that document the code ``in place'', providing an useful resource for programmers and users of modules. POD for

DBI and drivers is beginning to become more commonplace, and documentation for these modules can be read with the following commands.

```
The POD for the DBI Specification can be read with the command
        $ perldoc DBI

Users of the Oraperl emulation layer bundled with DBD::Oracle, may
read up on how to program with the Oraperl interface by typing:
        $ perldoc Oraperl

Users of the DBD::mSQL module may read about some of the private
functions and quirks of that driver by typing:
        $ perldoc DBD::mSQL

The Frequently Asked Questions is also available as
POD documentation. Read this by typing:
        $ perldoc DBI::FAQ

POD in general - Information on writing POD, and on the philosophy of POD in
general, can be read by typing:
        $ perldoc perlpod
```

Users with the Tk module installed may be interested to learn there is a Tk−based POD reader available called tkpod, which formats POD in a convenient and readable way.

See also −

- Information from DBI mailing lists http://www.symbolstone.org/technology/perl/DBI/tidbits
- DBI Perl Journal website http://www.tpj.com
- ``DBperl'' This article, published in the November 1996 edition of ``Dr. Dobbs Journal''.
- ``The Perl5 Database Interface'' a book to be written by Alligator Descartes and published by O'Reilly and Associates.

The mailing lists that users may participate in are:

- Mailing lists http://www.fugue.com/dbi
- **dbi−announce** Email: dbi−announce−request@fugue.com with a message body of 'subscribe'
- **dbi−dev** For developers Email: dbi−dev−request@fugue.com with a message body of 'subscribe'
- **dbi−users** general discussion Email: dbi−users−request@fugue.com with a message body of 'subscribe'
- US Mailing List Archives http://outside.organic.com/mail−archives/dbi−users/
- European Mailing List Archives http://www.rosat.mpe−garching.mpg.de/mailing−lists/PerlDB−Interest

Is DBI supported under Windows 95 / NT platforms?

The DBI and DBD::Oracle Win32 ports are now a standard part of DBI, so, downloading DBI of version higher than 0.81 should work fine. You can access Microsoft Access and SQL−Server databases from DBI via ODBC. Supplied with DBI−0.79 (and later) is DBI 'emulation layer' for the Win32::ODBC module. It's called DBI::W32ODBC. You will need the Win32::ODBC module.

- Win32 DBI http://www.symbolstone.org/technology/perl/DBI
- Win32 ODBC http://www.roth.net
- Perl interface to Microsoft SQL server http://www.algonet.se/~sommar/mssql

Commercial Support and Training

PERL CLINIC : The Perl Clinic can arrange commercial support contracts for Perl, DBI, DBD::Oracle and Oraperl. Support is provided by the company with whom Tim Bunce, author of DBI, works. For more information on their services, please see :

- Support http://www.perlclinic.com
- Support http://www.perldirect.com
- Training http://www.westlake.com/training

14.3 Testing Perl interface

See the section −

---

15. PostgreSQL Management Tools

15.1 PGACCESS − A GUI Tool for PostgreSQL Management

PgAccess is a Tcl/Tk interface to PostgreSQL. It is already included in the distribution of PostgreSQL. You may want to check out this web site for a newer copy

- http://www.flex.ro/pgaccess
- If you have any comment, suggestion for improvements, e−mail to : teo@flex.ro

Usage of pgaccess −

```
# man xhost
# xhost +
# su - postgres
bash$ man pgaccess
bash$ export DISPLAY=<hostname>:0.0
bash$ pgaccess mydatabase
```

Features of PgAccess

PgAccess windows − Main window, Table builder, Table(query) view, Visual query builder.

Tables

- opening tables for viewing, max 200 records (changed by preferences menu)
- column resizing, dragging the vertical grid line (better in table space rather than in the table header)
- text wrap in cells − layout saved for every table
- import/export to external files (SDF,CSV)
- filter capabilities (enter filter like (price>3.14)

- sort order capabilities (enter manually the sort field(s))
- editing in place
- improved table generator assistant
- improved field editing

Queries

- define , edit and stores "user defined queries"
- store queries as views
- execution of queries
- viewing of select type queries result
- query deleting and renaming
- Visual query builder with drag & drop capabilities. For any of you who had installed the Tcl/Tk plugin for Netscape Navigator, you can see it at work clicking here

Sequences

- defines sequences, delete them and inspect them Functions
- define, inspect and delete functions in SQL language

Future implementation will have

- table design (add new fields, renaming, etc.)
- function definition
- report generator
- basic scripting

INFORMATION ABOUT LIBGTCL

You will need the PostgreSQL to Tcl interface library libgtcl, lined as a Tcl/Tk 'load'−able module. The libpgtcl and the source is located in the PostgreSQL directory /src/interfaces/libpgtcl. Specifically, you will need a libpgtcl library that is 'load'−able from Tcl/Tk. This is technically different from an ordinary PostgreSQL loadable object file, because libpgtcl is a collection of object files. Under Linux, this is called libpgtcl.so. You can download from the above site a version already compiled for Linux i386 systems. Just copy libpgtcl.so into your system library director (/usr/lib). One of the solutions is to remove from the source the line containing load libpgtcl.so and to load pgaccess.tcl not with wish, but with pgwish (or wishpg) that wish that was linked with libpgtcl library.

If you get crypt not found during compilation pgaccess source tree then use −lcrypt.

15.2 GtkSQL Graphical Query Tool for PostgreSQL

GtkSQL is a graphical query tool (like PostgreSQL's psql). It is released under the GNU GPL license, and was developped using Gtk+ 1.2.3 and PostgreSQL 6.3. The main site of GtkSQL is at http://gtksql.sourceforge.net

Its main features are :

1. multiple SQL buffers
2. SQL keywords, table names and fields autocompletion

3. easy displaying of table definition
4. PostgreSQL and MySQL support (and easy addition of other databases)

The current version is GtkSQL v. 0.3. You can get the source from https://sourceforge.net/project/?form_grp=533

15.3 Windows Interactive Query Tool for PostgreSQL (WISQL or MPSQL)

MPSQL provides users with a graphical SQL interface to PostgresSQL. MPSQL is similar to Oracle's SQL Worksheet and Microsoft SQL Server's query tool WISQL. It has nice GUI and has history of commands. Also you can cut and paste and it has other nice features to improve your productivity.

- http://www.troubador.com/~keidav/index.html
- Email: keidav@whidbey.com
- http://www.ucolick.org/~de/ in file tcl_syb/wisql.html
- http://www.troubador.com/~keidav/index.html
- Email: de@ucolick.org

15.4 Interactive Query Tool (ISQL) for PostgreSQL called PSQL

ISQL is for character command line terminals. This is included in the distribution, and is called "psql". Very similar to Sybase ISQL, Oracle SQLplus. At unix prompt give command 'psql' which will put you in psql> prompt.

```
bash# su - postgres
bash$ man psql
bash$ psql mydatabase
Type \h to see help of commands.
```

Very user friendly and easy to use. Can also be accessed from shell scripts.

15.5 MPMGR − A Database Management Tool for PostgresSQL

MPMGR will provide a graphical management interface for PostgresSQL. You can find it at

- http://www.mutinybaysoftware.com/
- Email: keidav@mutinybaysoftware.com
- http://www.troubador.com/~keidav/index.html
- Email: keidav@whidbey.com
- http://www.ucolick.org/~de in file tcl_syb/wisql.html
- WISQL for PostgreSQL http://www.ucolick.org/~de/Tcl/pictures
- Email: de@ucolick.org

15.6 PgAdmin, PhpPgAdmin tools

- PgAdmin tool for Windows 95/NT Database design tool for PostgreSQL for Windows 95/NT http://www.pgadmin.freeserve.co.uk
- Web based admin tool − PhpPgAdmin for Postgresql is at http://www.phpwizard.net/projects/phpPgAdmin

15.7 PgBash − SQL shell tool

PgBash has functionality which is similar to psql. And, PgBash provides the useful functionality in making flexible interactive operational environment using bash alias, function, the history editing, etc.

The main site of PgBash is at http://www.psn.co.jp/PostgreSQL/pgbash/index−e.html

PgBash is a shell which included the "direct SQL" or the "embedded SQL" interface for PostgreSQL by the improvement on the bash(current version 2.03) shell. PgBash can be used as a log−in shell, a sub−shell(the shell started from a shell) and a shell program use.

Here, "Direct SQL" shows functionality which immediately outputs the result to the "standard output". And "Embedding SQL" shows functionality which sets the retrieval result into the shell variable and which deals with in shell script language.

SQL statement(terminal character is a semicolon) is handled as one shell command, it is possible to execute the SQL statement with pipeline, redirection and background_job options. In addition, by using the exec_sql command, it is possible to execute SQL statement with the various options. The executive example of SQL is shown next.

```
prompt> /usr/local/bin/pgbash                 .....Start of pgbash
pgbash> connect to db2@xxx.com user sakaida; .....connect to database
pgbash> select * from test limit 100; | more .....with pipeline
pgbash> select * from test; > /tmp/sel.dat & .....with redirect+background_Job
pgbash> addr='Osaka'
pgbash> insert into test values(
>  111,'name',                                .....can start new line
> '$addr'                                     .....use shell variable
> );
pgbash> connect to db3@yyy.com user postgres;
pgbash> set connection db2;                   .....set current D/B to db2
pgbash> select * from test;                   .....select db2's test
pgbash> exec_sql -d db3 "select * from test3".....change connection to db3
pgbash> ls
pgbash> begin;
pgbash> declare cur cursor for select * from test;
pgbash> fetch in cur into :AA,:BB;            .....set shell variables
pgbash> echo "AA=$AA, BB=$BB"
pgbash> end;
pgbash> fc fetch                              .....edit history and go
pgbash> !echo                                 .....retry echo
pgbash> disconnect all                        .....disconnect all connections
pgbash> exit                                  .....End of pgbash
```

15.8 Webmin Tool for PostgreSQL

The webmin tool (administration of an Unix machine through a, secure if you want, webpage) has a PostgreSQL module as of the latest release (version 0.82). With this module you can add users, groups, databases, tables, view tables.

You can find webmin on

---

16. CPUs for PostgreSQL

See the document http://metalab.unc.edu/LDP/HOWTO/CPU−Design−HOWTO.html for list of CPUs available for PostgreSQL and also this document gives details on CPU design.

The following CPUs (both 64−bit and 32−bit) are available for PostgreSQL. All these CPUs run Linux.

- Main CPU site is : Google Search engine CPU site "Computers>Hardware>Components>Microprocessors" http://directory.google.com/Top/Computers/Hardware/Components/Microprocessors

The following is GNU/GPL open−source CPU list:

- Open−source CPU site − Google Search "Computers>Hardware>Open Source" http://directory.google.com/Top/Computers/Hardware/Open_Source
- OpenRISC 1000 Free 32−bit processor IP core competing with proprietary ARM and MIPS is at http://www.opencores.org/cores/or1k
- OpenRISC 2000 is at http://www.opencores.org
- European Space Agency's ESA−32bit and ESA−64bit CPUs "LEON" Sparc http://www.estec.esa.nl/wsmwww/leon
- GNU/GPL Freedom 64−bit F−CPU http://www.f−cpu.org or at http://f−cpu.tux.org mirror site at http://www.f−cpu.de
- STM 32−bit, 2−way superscalar RISC CPU http://www.asahi−net.or.jp/~uf8e−itu
- Free microprocessor and DSP IP cores written in Verilog or VHDL http://www.cmosexod.com
- Free hardware cores to speed development http://www.scrap.de/html/opencore.htm
- Opencores org − open source, free IP cores http://www.opencores.org
- Linux open hardware and free EDA systems http://opencollector.org
- ARM CPU http://www.arm.com/Documentation
- Cogent CPUs http://www.cogcomp.com

The following is commercial CPU list:

- Russian E2k 64−bit CPU (Very fast CPU !!!) website : http://www.elbrus.ru/roadmap/e2k.html Elbrus is now partnered (alliance) with Sun Microsystems of USA
- Korean CPU from Samsung 64−bit CPU original from DEC Alpha http://www.samsungsemi.com Alpha−64bit CPU is at http://www.alpha−processor.com Now there is collaboration between Samsumg, Compaq of USA on Alpha CPU
- Intel IA 64 http://developer.intel.com/design/ia−64
- Transmeta crusoe CPU and in near future Transmeta's 64−bit CPU http://www.transmeta.com

- Sun Ultra−sparc 64−bit CPU http://www.sun.com or http://www.sunmicrosystems.com
- MIPS RISC CPUs http://www.mips.com
- Silicon Graphics MIPS Architecture CPUs http://www.sgi.com/processors
- IDT MIPS Architecture CPUs http://www.idt.com
- IBM Power PC (motorola) http://www.motorola.com/SPS/PowerPC/index.html
- Motorola embedded processors. SPS processor based on PowerPC, M−CORE, ColdFire, M68k, or M68HC cores http://www.mot−sps.com
- Hitachi SuperH 64−bit RISC processor SH7750 http://www.hitachi.com sold at $40 per cpu in quantities of 10,000
- Fujitsu 64−bit processor http://www.fujitsu.com
- HAL−Fujitsu (California) Super−Sparc 64−bit processor http://www.hal.com also compatible to Sun's sparc architecture.
- Seimens Pyramid CPU from Pyramid Technologies
- Intel X86 series 32−bit CPUs Pentiums, Celeron etc..
- AMDs X86 series 32−bit CPUs K−6, Athlon etc..
- National's Cyrix X86 series 32−bit CPUs Cyrix etc..
- Other CPUs from other countries (Taiwan, Korea, Japan) ?? Let me know...

Other important CPU sites are at −

- World−wide 24−hour news on CPUs http://www.newsnow.co.uk/cgi/NewsNow/NewsLink.htm?Theme=Processors
- The computer architecture site is at http://www.cs.wisc.edu/~arch/www
- ARM CPU http://www.arm.com/Documentation
- Great CPUs http://www.cs.uregina.ca/~bayko/cpu.html
- Microdesign resources http://www.mdronline.com

---

17. Setting up multi−boxes PostgreSQL with just one monitor

If you do want to spend money on hardware switches than you can **use VNC (Virtual Network Computing) Technology** from the telecom giant AT & T. VNC is GPLed and is a free software. Using VNC you can run PostgreSQL programs on computers without monitors and display on remote boxes with monitors!! But the boxes must be connected via ethernet Network Interface Cards. VNC is at http://www.uk.research.att.com/vnc

You can stack up multiple CPU−boxes and connect to just one monitor and use the KVM (Keyboard, Video, Monitor) switch box to select the host. This saves space and avoids a lot of clutter and also eliminates monitor, keyboard and the mouse (saving anywhere from 100 to 500 US dollars per set).

Using this switch box, you can stack up many PostgreSQL servers (development, test, production), Web servers, ftp servers, Intranet servers, Mail servers, News servers in a tower shelf. The switch box can be used for controlling Windows 95/NT or OS/2 boxes as well.

Please check out these sites:

- DataComm Warehouse Inc at 1−800−328−2261. They supply all varieties of computer hardware http://www.warehouse.com 4−port Manual KVM switch (PS/2) is about $89.99 Part No. DDS1354
- Network Technologies Inc http://www.networktechinc.com/servswt.html (120 dollars/PC 8 ports) which lists 'Server Switches' and 'Video only switches'

- Scene Double Inc, England http://www.scene.demon.co.uk/qswitch.htm
- Cybex corporation http://www.cybex.com
- Raritan Inc http://www.raritan.com
- RealStar Solutions Inc http://www.real−star.com/kvm.htm
- Belkin Inc http://www.belkin.com
- Better Box Communications Ltd. http://www.betterbox.com/info.html
- Go to nearest hardware store and ask for "Server Switch" also known as "KVM Auto Switches".

Search engine yahoo to find more companies with "Server Switches" or "KVM Switches".

It is strongly recommended to have a dedicated unix box for each PostgreSQL data−server for better performance. No other application program/processes should run on this box. See the Business section of your local newspapers for local vendors selling only intel box, 13" monochrome monitor (very low cost monitor). Local vendors sell just the hardware **without** any Microsoft Windows/DOS. You do not need a color monitor for the database server, as you can do remote administration from color PC workstation.

You can buy bare−bone computer hardware from online stores. You can get good rates in "Online Auctions"

- Online store and auction hall http://www.egghead.com
- Online store http://www.buy.com
- Bidding store http://www.ubid.com

Get RedHat (or some other distribution of) Linux cdrom from below −

- Linux System Labs Web site: http://www.lsl.com/ 7 (U.S. dollars)
- Cheap Bytes Inc Web site: http://www.cheapbytes.com/ 7 (U.S. dollars)

Make sure that the hardware you purchase is supported by Redhat Linux. Check the ftp site of Redhat for recommended hardware like SCSI adapters, video cards before buying. For just $ 600 you will get a powerful intel box with Redhat Linux running PostgreSQL. Use odbc/jdbc/perl/tcl to connect to PostgreSQL from Windows95, OS/2, Unix Motif or web browser (e.g. Redbaron, Opera, Netscape, 20 others). (Web browsers are very fast becoming the standard GUI client).

Using KVM switch you can control many cpu boxes by just one monitor and one keyboard!

---

18. Web−Application−Servers for PostgreSQL

Several Web−Application−Servers work with PostgreSQL both open−source and commercial versions. The popular open−source Web−Application−Servers are Perl based Application Servers like Mason, WIRM, Velocigen, Enhydra(Java) and Zope(Python) and commercial Web−Application−Servers are IBM Websphere, BEA Weblogic.

It is recommeded that you use secure web server like Apache + mod_ssl + OpenSSL. See Redhat StrongHold secure server at http://www.c2.net/products/sh3.

Web Application Servers can be classified according to the programming language which they support. You must choose a Web Application server based on the programming language which **you like the most**.

Classifications of Web Application servers are:

- Based on **PERL** language
- Based on **PHP** language (which is similar to PERL, little Java−like)
- Based on **Python** language (Object oriented scripting language)
- Based on **Java** language (Sun Microsystems Java)
- Based on **Tcl** language (Tcl/Tk − called "Tickle" scripting language)
- Based on **C++** language (C++ and CORBA)
- Based on **Pike** (C++ like scripting language)

18.1 PERL Web Application Servers

Perl language has a very long life just like "C" language, and Perl will be in use for **a long time** in the future! Perl runs **3 times** faster than Java for some operations (but Java runs faster than perl for some operations). Java is a very complex system with virtual machine and interpreter which makes it extremely slow, unstable and unreliable. Perl is very simple, fast and object oriented.

Also Perl programs can be easily compiled for even better performance. Use Perl2Exe which is a command line utility for converting perl scripts to executable files http://www.indigostar.com/perl2exe.htm

The following Web Applicaiton servers are available for PERL

- Mason http://www.masonhq.com is a powerful Perl−based web site development and delivery engine. With Mason you can embed Perl code in your HTML and construct pages from shared, reusable components.

- BingoX http://opensource.cnation.com/projects/BingoX is an open source, object oriented Web Application Framework written in mod_perl meant to dramatically reduce the time required to build large dynamic, database driven web sites and applications.

- SmartWorker is a collection of Perl classes that allow you to build web applications like they were true applications and not just HTML templates with random embedded code. SmartWorker http://www.smartworker.org

- Apache−Perl integration project With mod_perl it is possible to write Apache modules entirely in Perl. In addition, the persistent interpreter embedded in the server avoids the overhead of starting an external interpreter and the penalty of Perl start−up time. Visit http://perl.apache.org and also see **mod_perl_garden** project at http://modperl.sourcegarden.org

- Apache::ASP http://www.apache−asp.org provides an Active Server Pages port to the Apache Web Server with Perl as the host scripting language. Apache::ASP allows a developer to create dynamic web applications with session management and embedded perl code. There are also many powerful extensions, including XML taglibs, XSLT rendering, and new events not originally part of the ASP API.

- WIRM (Web Interface Repository Manager) is a Perl−based application server that provides a high−level programming environment for developing web information systems. The WIRM consists of an object−relational database and a suite of Perl interfaces for visualizing, integrating and analyzing heterogeneous multimedia data. WIRM provides facilities for creating context−sensitive views over a multimedia database, allowing developers to rapidly build dynamic web sites that adapt their content and presentation to multiple classes of end−users. Visit http://www.wirm.org

- EmbPerl http://perl.apache.org/embperl Embperl gives you the power to embed Perl code in your HTML documents. Using Perl means being able to use a very elaborate programming language, which is widely used for WWW purposes. You can also use hundreds of Perl modules which have already been written − including DBI − for database access to a growing number of database systems.
- ePerl http://www.engelschall.com/sw/eperl interprets an ASCII file bristled with Perl 5 program statements by evaluating the Perl 5 code while passing through the plain ASCII data. It can operate in various ways: As a stand−alone Unix filter or integrated Perl 5 module for general file generation tasks and as a powerful Webserver scripting language for dynamic HTML page programming.
- XPP http://opensource.cnation.com/projects/XPP XPP stands for 'XPP Parses Perl' or 'XPML Page Parser', and is a fast/efficient HTML parser that parses embedded perl, as well as HTML like tags, from dynamic html pages called XPML pages.
- Gamla − a perl−based RAD and application server The gamla project aims to create a Rapid Application Development (RAD) tool and a web application server based on Perl. All the source code produced by the Gamla project will be under the public domain. Gamla at http://gamla.iglu.org.il
- AxKit http://www.axkit.org is an XML Application Server for Apache (and mod_perl). It provides on−the−fly conversion from XML to any format, such as HTML, WAP or text using either W3C standard techniques, or flexible custom code. AxKit also uses a built−in Perl interpreter to provide some amazingly powerful techniques for XML transformation. The emphasis with AxKit is on separation of content from presentation. The pipelining technique that AxKit uses allows content to be converted to a presentable format in stages, allowing certain platforms to see data differently to others. AxKit allows web designers to focus on web site design, content developers to work on a purely content basis, and webmasters to focus on their core competencies.

Commercial Web Application Servers for Perl:

- Zelerate AllCommerce http://www.zelerate.org/html/eng/home.shtml is a commerce, content, customer and relationship management system. This high−performance, scalable Internet application is written in Perl and uses a backend database.
- VelociGen serves dynamic content stored in XML, the database or live data feeds as fast as static HTML − up to 60x faster than CGI without the need to modify your exiting application. VelociGen also makes new development easier with server−side XML tags, crash protection and load balancing across multiple machines. VelociGen plugs seamlessly into any Web server on any platform, increasing server performance and speeding the response times of dynamic content driven web sites. VelociGen can process large volumes of simultaneous requests as much as 10x faster than Java Servlets and 4x faster than Cold Fusion. Velocigen http://www.binevolve.com/velocigen

18.2 PHP Web Application Servers

The following Web Applicaiton servers are available for PHP

- Midgard PHP Web Application server is based on the PHP scripting language and PHP runs extremely fast − faster than Java. The main site of Midgard is at http://www.midgard−project.org PHP can be compiled with Zend compiler and optimizer http://www.zend.com. PHP runs very fast and is about 5 to 10 times faster than Java. See Midgard

Installation and also PHP HOWTO at http://www.linuxdoc.org/HOWTO/PHP−HOWTO.html

- Ariadne http://www.muze.nl/software/ariadne is a web application system. It consists of a complete framework for the easy development and management of web applications, using PHP. The system uses a modular approach, using abstract interfaces for all transactions. This results in maximum freedom to change parts of the systems workings or add new functionality without needing to reprogram other parts

18.3 Lutris Corp "Enhydra Enterprise" (Java)

Enhydra supports PostgreSQL database. Enhydra is an immensely popular Java/XML/J2EE Web−Application−Server created by 'Lutris Corporation'. It is the world's best Java/XML/J2EE Web−Application server. It supports EJB, Servlets, JSP, JNDI, JDBC, JTA, CORBA, XMLC/Rocks, DODS and internationalization. It is used by many large fortune 500 companies in US and Europe. Companies like "French Telecom" are directly sponsoring the Enhydra. It is written in 100% pure Java and is available from http://www.enhydra.org. Enhydra is an open source code project but is commercially sold and supported by Lutris Corp. Visit http://www.lutris.com

See tutorial on setting up the PostgreSQL with Enhydra and see also Setup database with Enhydra.

You would use Borland Corp's JBuilder along with Enhydra. JBuilder is at http://www.inprise.com

See also Enterprise Java HOWTO at http://www.linuxdoc.org/HOWTO/Enterprise−Java−for−Linux−HOWTO.html

18.4 Zope (Python)

Python is becoming immensely popular "pure" object−oriented scripting language. Zope is a Web−Application server and provides interfaces to PostgreSQL. Zope is available at http://www.zope.org Python is at http://www.python.org

18.5 OpenACS (Tcl Language)

OpenACS (Open ArsDigita Community System) http://openacs.org is an advanced toolkit for building scalable, community−oriented web applications. It relies on AOLserver, a web/application server, and PostgreSQL, a true ACID−compliant RDBMS. These are two high−quality products available for free under open−source licenses.

ACS is created by ArsDigita, http://www.arsdigita.com, their ACS (ArsDigita Community System) attempts to be as DB independent as possible, though it is based on Oracle (hence Open ACS have to take time out to port it).

See also http://www.appserver−zone.com

18.6 C++, CORBA Web Application Servers

- PortalSphere Web Application Server is built in C++ and runs on Unix (and Linux) for ultimate in speed and stability. Strictly adhering to the CORBA standard, PortalSphere supports both the standard Internet HTTP communications protocol and the IIOP point−to−point protocol for ultra−high−speed client−server links. Coupled with direct (native) access to all popular databases, these features give PortalSphere lightening−fast performance and the unique inherent ability to support real−time events over the internet. PortalSphere is − Up to 100 X faster than HTTP/CGI, Direct (native) access to most popular databases, Scalable to 10,000+ concurrent user sessions. Visit PortalSphere at http://www.portalsphere.com/overview.html.

- FlashPoint C++,C,PERL Web Application Server project exists to support high speed web application service in a multi−threaded environment, to support a variety of development languages including C & C++, and to support good software engineering practices to a degree difficult in many other environments. It can be used alongside Apache, and in some instances can replace it, depending on your needs Visit http://www.bouldersoftware.com/products/flashpoint and download the from FlashPoint Redhat RPM package.
- "C Server Pages" http://cserverpages.20m.com is One efficient and Scalable Application Server written in C/C++, which powers web server pages written in C++ and Templates with Dynamics Elements embedded. You can use the approach you prefer or both. You can build your business objects using C++. Your pages can be the CORBA clients for any ORB in the market. Has connectivity to all SQL databases.

18.7 Pike, Roxen Web Application Server

Pike is a dynamic programming language with a syntax similar to C++. It is simple to learn, does not require long compilation passes and has powerful built−in data types allowing simple and fast data manipulation. Pike is released under the GNU/GPL general public license.

Pike is a very powerful object oriented scripting language and since it's syntax is identical to C++ it is expected that it's popularity will **explode** in coming years.

Pike is at http://pike.roxen.com and Roxen web server is at http://www.roxen.com.

Roxen is a modular web server that has a complete DB interface, and includes Postgres support. It has full support for SSL, and is released under the GPL. Roxen is written using Pike scripting language.

18.8 Web Application Servers Directory

Visit Web Application Servers directory which has "Yellow Pages".

---

19. Applications and Tools for PostgreSQL

19.1 PostgreSQL 4GL for web database applications − AppGEN Development System

AppGEN can be downloaded from

- http://www.man.ac.uk/~whaley/ag/appgen.html
- ftp://ftp.mcc.ac.uk/pub/linux/ALPHA/AppGEN.

AppGEN is a high level fourth generation language and application generator for producing World Wide

Web (WWW) based applications. These applications are typically used over the internet or within a corporate intranet. AppGEN applications are implemented as C scripts conforming to the Common Gateway Interface (CGI) standard supported by most Web Servers.

To use AppGEN you will need the following :−

PostgresSQL, relational database management system

A CGI compatible web server such as NCSA's HTTPD

An ansi C compiler such as GCC

AppGEN consists of the following Unix (Linux) executables :−

- defgen, which produces a basic template application from a logical data structure. The applications are capable of adding, updating, deleting and searching for records within the database whilst automatically maintaining referential integrity.
- appgen, the AppGEN compiler which compiles the appgen source code into CGI executable C source and HTML formatted documents ready for deployment on a web server.
- dbf2sql, an utility for converting dBase III compatible .dbf files into executable SQL scripts. This enables data stored in most DOS/Windows based database packages to be migrated to a SQL server such as PostgresSQL.
- In addition, AppGEN comprises of a collection of HTML documents, GIF files and Java applets which are used at runtime by the system. And of course, like all good software, the full source code is included.

The author, Andrew Whaley, can be contacted on

- andrew@arthur.smuht.nwest.nhs.uk

19.2 WWW Web interface for PostgresSQL − DBENGINE

dbengine a plug 'n play Web interface for PostgreSQL created by Ingo Ciechowski. It is at

- http://www.cis−computer.com/software/dbengine

About DBENGINE : dbengine is an interface between the WWW and Postgres95 which provides simple access to any existing database within just a few minutes.

PHP gives you a Perl like language in your documents, but no real Perl while AppGen and wdb−p95 require that you create some configuration file for each of your databases −− sound's like you'll first of all have to learn some sort of new meta language before you can get started.

Unlike other tools you don't have to learn any special programming or scripting language to get started with dbengine. Also there's no configuration file for each database, so you don't have to get familiar with such a new structure. However − in case you want to gain access to the full features of dbengine it'd be a good idea to know the Perl language.

The whole system can be configured by simple manipulations of an additional database that contains closer information about how to visualize your database access. You can even specify virtual Fields which are calculated on the fly right before they're displayed on the screen.

19.3 Apache Webserver Module for PostgreSQL − NeoSoft NeoWebScript

Apache is a well−known Web Server. And a module to interface PostgreSQL to Apache Webserver is at −

- http://www.neosoft.com/neowebscript/

NeoWebScript is a programming language that allows both simple and complex programs to be embedded into HTML files.

When an HTML page containing embedded NeoWebScript is requested, the NeoWebScript−enabled webserver executes the embedded script(s), producing a webpage containing customized content created by the program.

NeoWebScript is a fast, secure, easy to learn way to do powerful, server−based interactive programming directly in the HTML code in web pages. With NeoWebScript, counters, email forms, graffiti walls, guest books and visitor tracking are all easy, even for a beginning programmer. See how well NeoWebScript holds its' own vs. PERL and JavaScript.

If you'd like to install NeoWebScript on your webserver, your Webmaster needs to read our Sysop FAQ to get started. Theory of Operations will explain how NeoWebScript works, while installation will take them through the steps. Management deals with configuration issues and running the server, tests let you verify correct NeoWebScript operation, and troubleshooting deals with server problems.

There is no cost to you to use NeoWebScript−2.2 for your ISP, your intranet, or your extranet. You'll see a full license when you register to download, but it costs $ 99 if you want to embed it in your own product or use it in a commerce (eg. SSL) server.

NeoWebScript is a module for the Apache webserver that allows you to embed the Tcl/Tk programming language in your webpages as a scripting tool. It was invented by Karl Lehenbauer, NeoSoft's Chief Technical Officer, and documented, enhanced and extended by NeoSoft's programmers and technical writers.

The Apache webserver is the world's most popular webserver, accounting for 68 % of the sites polled.

Tcl/Tk is the powerful, free, cross−platform scripting language developed by Dr. John Ousterhout. In his own words

"Tcl/Tk lets software developers get the job done ten times faster than with toolkits based on C or C++. It's also a great glue language for making existing applications work together and making them more graphical and Internet−aware."

Karl Lehenbauer, Founder and Chief Technical Officer of NeoSoft, has been part of Tcl/Tk development from the very beginning. Together with Mark Diehkans, they authored Extended Tcl, also known as TclX or NeoSoft Tcl, a powerful set of extensions to the language. Many of the current core Tcl commands originated in Extended Tcl, and were then imported into the core language by Dr. Ousterhout.

NeoSoft Inc., 1770 St. James Place, Suite 500, Houston, TX 77056 USA

19.4 HEITML server side extension of HTML and a 4GL language for PostgreSQL

Tool heitml is another way to interface postgres with the world wide web. For more details contact

```
Helmut Emmelmann H.E.I. Informationssyteme GmbH
Wimpfenerstrasse 23 Tel. 49−621−795141
68259 Mannheim Germany Fax. 49−621−795161
```

- E−mail Mr.Helmut Emmelmann at emmel@h−e−i.de
- Heitml main web site http://www.heitml.com
- Heitml secondary web site http://www.h−e−i.deom

heitml is a server side extension of HTML and a 4GL language at the same time. People can write web applications in the HTML style by using new HTML−like tags.

heitml (pronounced "Hi"−TML) is an extension of HTML and a full−featured 4th generation language that enables Web−based Applications to interact with data stored in SQL databases, without resorting to complex CGI scripts.

heitml extends HTML on the sever side, dynamically converting ".hei" files to HTML format and so is compatible with any web browser.It embraces the familiar, easy−to−use HTML syntax and provides a large assortment of pre−developed Tags and Libraries to take care of tasks that formerly required CGI. As XML, heitml provides user defined tags. With heitml the user defined markup can be translated to HTML and send to a browser.

heitml targets both HTML designers and professional programmers alike. HTML designers can use heitml Tags to build dynamic web pages, access SQL databases, or create complete web applications. Counters, registration databases, search forms, email forms, or hierarchical menues can all be created simply by using the pre−developed HTML−like Tags found in the many Component Libraries.

For programmers heitml embeds a complete forth generation language in HTML

```
(e.g. <if>, <while>, and <let> Tags),
```

plus powerful expression evaluation with integer, real, boolean, string, and tuple data types. Tuples have reference semantics as in modern object oriented languages and are stored on a heap. heitml variables including all complex data structures stored on the heap maintain their values between pages using the Session Mode. It is possible to define your own tags or environment tags and even re−define HTML−tags.

heitml makes it possible to

− − − develop Web Sites in a structured and modular way, drastically reducing maintenance overhead.

− − − develop intelligent and interactive Web Sites, with content that dynamically adapts itself to user needs.

− − − show the content of SQL databases with no programming other than to use our library of prefined "dba" Tags.

− − − develop complex database and Catalog Shopping applications using Session Variables heitml runs on Linux with any Web Server using the CGI interface, and is especially fast (avoiding the CGI overhead) within the APACHE Web Server using the apache API. Currently MSQL (Version 1 and 2), PostgreSQL (Version 6), mysql, and the yard databases are supported). heitml also works on Linux, BSDi, Solaris and SunOS, as well as Windows NT with CGI and ISAPI and ODBC and Windows 95.

heitml (on linux) is free for research, non−commercial and private usage. Commercial Web Sites must pay a licensing fee. The fully operational version of heitml is available for a trial period downloaded freely. (Note, however, that each ".hei" Web Page you develop will display a message identifying it as the version for non−commercial use. After registration, you will receive a key to switch off the message without having to re−install the program.)

heitml (pronounced "Hi"−TML) significantly extends and enhances the functionality of HTML by definable tags and full programming features. This makes dynamic content and database applications possible simply within the HTML world, without CGI and without external scripting or programming languages. This means you, as an HTML author, can embed applications in your web pages, simply by using some new tags without CGI and without programming. As an advanced user or programmer on the other hand you can create and program powerful tag libraries. This approach makes heitml suitable for HTML newcomers and professional programmers alike. heitml runs on the web server and dynamically generates HTML, so heitml is compatible with the internet standards and with any web browser. It allows full access to databases while shielding the user from any unneccessary CGI complexity. heitml has been developed according to the newst research and in compiler construction and transaction systems.

heitml pages are developed just the same way as HTML pages, with a text editor or HTML editor, and placed on the web server as usual. However now pages can contain dynamic heitml tags and access tag libraries. You can use these tags to access the database, to create dynamic content, to send emails, and even to create powerful applications like registration databases and shopping systems.

HTML newcomers and professional programmers alike will be amazed at how quickly and easily they can design exciting applications like our Interactive Guestbook without resorting to complex and difficult to learn CGI scripts, simply by using the tools provided in our dba Library.

heitml is accompanied by a wide range of tag libraries, to create guestbooks, database maintenance applications, extensible query forms, powerful email forms or structure your web site using a hierarchic menu. These tools are ready to go, just add the corresponding tags to your web site.

As an experienced programmer you can make fully use of the heitml persistent dynamic tuple architecture : heitml is not just a scripting language with dynamic typing, full power expression evaluation, recursive procedures and extensive parameter passing features, but it also features persistent dynamic tuples to automatically keep session data of any size.

19.5 America On−line AOL Web server for PostgreSQL

The no−cost commercial webserver, AOLserver supports database connections to PostgreSQL for more information see

- AOL Web Server home http://www.aolserver.com
- Introduction to AOLserver by Philip Greenspun http://photo.net/wtr/aolserver/introduction−1.html

AOLserver is a fast, fully multithreaded, Tcl enabled webserver. But not only that, it is a complete database−backed web development platform. With AOLserver you can have multiple pooled connections to PostgreSQL (and other RBDMSs) that can be shared among different threads. AOLserver has a Tcl and C

APIs that allow you to develop powerful dynamic websites. All this since 1995. It is licensed under the APL (AOLserver Public License) or the GPL, thus being totally free software. The Tcl API is the most useful for web sites. AOLserver has a set of powerful Tcl calls, such as ns_sendmail (to send e−mail), ns_httpget (to fetch an URL), ns_schedule (a cron−like feature to schedule procedures to run at specific times), etc. You can also extend AOLserver's capabilities very easily with the Tcl API. Each AOLserver virtual server can have its own "library" of private Tcl scripts that are parsed by AOLserver and become accessible to any page within that virtual server. You can develop pages for AOLserver in three ways: − Plain HTML − .tcl pages −− these are tcl programs that can return HTML via the ns_write call. − .adp pages −− AOL Dynamic Pages. You develop your pages in plain HTML but you can scape to Tcl code by using <% %> or <%= %> much alike PHP or ASP. While AOLserver is a great webserver with a superb architecture, where it really shines is in database connectivity. AOLserver has its own database abstraction layer that enables you to have it connected to different RDBMSs without changing your code at all. The connections do the RDBMS are pooled, persistent and are shared among different threads. This allows for very fast connections and efficient use of resources. AOLserver has drivers for all major RDBMSs: PostgreSQL, Oracle, Sybase, Informix, Illustra, Solid, Interbase, MySQL.

19.6 Problem/Project Tracking System Application Tool for PostgreSQL

This is at

- http://www.homeport.org/~shevett/pts/

19.7 Convert dbase dbf files to PostgreSQL

The program dbf2msql works fine with mSQL and PostgreSQL. You can find it at

- ftp://ftp.nerosworld.com/pub/SQL/dbf2sql/
- ftp://ftp.postgresql.org/pub/contrib/dbf2pg−3.0.tar.gz
- Pg2Xbase is a better package then dbf2pg http://w3.man.torun.pl/~makler/prog/pg2xbase

This program was written by Maarten Boekhold, Faculty of Electrical Engineering TU Delft, NL Computer Architecture and Digital Technique section

- M.Boekhold@et.tudelft.nl

You can also use a python method to read dbf files and load into a postgres database.

- See http://www.python.org

19.8 Convert Microsoft Access MDB database files to PostgreSQL

MDB Tools is a planned set of libraries and utilities to facilitate exporting data from MS Access databases (mdb files) into a multiuser database such as Oracle, Sybase, DB2, Informix, MySQL, Postgresql, or similar.

- Get MDB tool from http://mdbtools.sourceforge.net
- Mailing list http://lists.sourceforge.net/mailman/listinfo/mdbtools−dev

19.9 Zeos Client

"Zeos" a program products for development and administration of the database applications, with use OpenSource SQL−servers − MySQL, PostgreSQL and InterBase http://www.zeos.dn.ua/eng/index.html

19.10 Report Writer in Java

Generic Report Writer is a menu−driven report writer. It is not a drag−and−drop interface. Works with PostgreSQL, MySQL, and Access. It probably will work on any other database for which you have a Type 4 JDBC version 1 driver. It is at

---

20. Database Design Tool − Entity Relation Diagram Tool

"DeZign for databases" (http://www.heraut.demon.nl/dezign/index.html) is a database development tool using an entity relationship diagram. It visually supports the lay out of the entities and relationships and automatically generates SQL−schemas for most leading databases.

"DeZign for databases" supports the logical and physical data−level from a single specification achieved by using automatic foreign key migration at design−time. Multiple display options include entity/primary key/attributes inclusive foreign keys/attributes exclusive foreign keys. "DeZign for databases" also supports domains (user defined datatypes).

Reports generated by DeZign can be used for conveying complex designs in simplified format to managers at various management levels. You can generate reports, datadictionaries and databases by one simple click. The following databases are supported: Oracle, Interbase, IBM DB2, Sybase, MS Access (95/97/200), MS SQL Server, Paradox, dBase, Informix, SQL−Anywhere, MySQL and PostgreSQL.

Heraut "DeZign for databases" is at (http://www.heraut.demon.nl)

---

21. Web Database Design/Implementation tool for PostgreSQL − EARP

- http://www.oswego.edu/Earp
- ftp://ftp.oswego.edu in the directory 'pub/unix/earp'.

21.1 What is EARP ?

The "Easily Adjustable Response Program" (EARP) created by David Dougherty. EARP is a Web Database Design/Implementation tool, built on top of the PostgreSQL database system. Its functionality includes:

- A Visual Design System.
- A sendmail interface. (can handle incoming and outgoing mail)
- An Enhanced Security Mechanism.
- A cgi driver.

21.2 Implementation

The main implementation of EARP is a CGI binary which runs under the http daemon to provide access to the database server. All of the design tools are built into the driver, no design takes place over anything but the web. The tools themselves require a graphical browser, the compatibility of objects designed with the tools is implementation independent, based on designing individuals preferences.

21.3 How does it work ?

One of the main features of EARP is that it uses an Object Oriented approach to producing html pages which interface to the database. Most pages will consist of several objects. Each object is produced by some sort of tool and given a name, objects are then linked together in a callable sequence by the page tool. Objects are also reusable across multiple pages. Basic tools exist for HTML, Querys, Grabbing input from forms, Extendable Formatting of Query and Input objects, and Linking together of objects into other objects. More advanced tools include the mail tool and the multithreaded query tool.

Another feature of EARP is advanced security. Access to various areas of the EARP system can be limited in a variety of ways. To facilitate its advanced security, EARP performs checks for each connection to the system, determining what ids and groups the connecting agent belongs to. Access to areas is defined seperately, and the combination decides if access to a specific area of Earp is allowed. Moreover, all that is required to implement the security features is an http server that supports basic (or better) user authentication.

21.4 Where to get EARP ?

EARP is available via anonymous ftp from

- ftp://ftp.oswego.edu in the directory 'pub/unix/earp'.

---

22. PHP Hypertext Preprocessor − Server−side html−embedded scripting language for PostgreSQL

WWW Interface Tool is at −

- http://www.php.net
- http://www.vex.net/php

PHP also has a compiler called Zend which will vastly improve the performance. First you will write your application in PHP scripting language during development, testing and debugging. Once the project is ready for deployment you will use the Zend compiler to compile the PHP to create executable which will run very fast.

Old name is Professional Home Pages (PHP) and new name is PHP Hypertext Pre−Processor

- Mirror sites are in many countries like www.COUNTRYCODE.php.net
- http://www.fe.de.php.net
- http://www.sk.php.net
- http://php.iquest.net/
- Questions e−mail to : rasmus@lerdorf.on.ca

PHP is a server−side html−embedded scripting language. It lets you write simple scripts right in your .HTML files much like JavaScript does, except, unlike JavaScript PHP is not browser−dependant. JavaScript is a client−side html−embedded language while PHP is a server−side language. PHP is similar in concept to Netscape's LiveWire Pro product. If you like free fast−moving software that comes with full source code you will probably like PHP.

- The PostgreSQL support code was written by Adam Sussman asussman@vidya.com

22.1 Major Features

- Standard CGI, FastCGI and Apache module support − As a standard CGI program, PHP can be installed on any Unix machine running any Unix web server. With support for the new FastCGI standard, PHP can take advantage of the speed improvements gained through this mechanism. As an Apache module, PHP becomes an extremely powerful and **lightning fast** alternative to CGI programmimg.
- Access Logging − With the access logging capabilities of PHP, users can maintain their own hit counting and logging. It does not use the system's central access log files in any way, and it provides real−time access monitoring. The Log Viewer Script provides a quick summary of the accesses to a set of pages owned by an individual user. In addition to that, the package can be configured to generate a footer on every page which shows access information. See the bottom of this page for an example of this.
- Access Control − A built−in web−based configuration screen handles access control configuration. It is possible to create rules for all or some web pages owned by a certain person which place various restrictions on who can view these pages and how they will be viewed. Pages can be password protected, completely restricted, logging disabled and more based on the client's domain, browser, e−mail address or even the referring document.
- PostgresSQL Support − Postgres is an advanced free RDBMS. PHP supports embedding PostgreSQL "SQL queries" directly in .html files.
- RFC−1867 File Upload Support − File Upload is a new feature in Netscape 2.0. It lets users upload files to a web server. PHP provides the actual Mime decoding to make this work and also provides the additional framework to do something useful with the uploaded file once it has been received.
- HTTP−based authentication control − PHP can be used to create customized HTTP−based authentication mechanisms for the Apache web server.
- Variables, Arrays, Associative Arrays − PHP supports typed variables, arrays and even Perl−like associative arrays. These can all be passed from one web page to another using either GET or POST method forms.
- Conditionals, While Loops − PHP supports a full−featured C−like scripting language. You can have if/then/elseif/else/endif conditions as well as while loops and switch/case statements to guide the logical flow of how the html page should be displayed.
- Extended Regular Expressions − Regular expressions are heavily used for pattern matching, pattern substitutions and general string manipulation. PHP supports all common regular expression operations.
- Raw HTTP Header Control − The ability to have web pages send customized raw HTTP headers based on some condition is essential for high−level web site design. A frequent use is to send a Location: URL header to redirect the calling client to some other URL. It can also be used to turn off cacheing or manipulate the last update header of pages.
- On−the−fly GIF image creation − PHP has support for Thomas Boutell's GD image library which makes it possible to generate GIF images on the fly.
- ISP "Safe Mode" support − PHP supports an unique "Safe Mode" which makes it safe to have multiple users run PHP scripts on the same server.

- Many more new features are being added in newer releases of PHP. Visit the main web site at http://www.php.net
- It's Free! − One final essential feature. The package is completely free. It is licensed under the GNU/GPL which allows you to use the software for any purpose, commercial or otherwise.

22.2 PHP − Brief History

PHP began life as a simple little cgi wrapper written in Perl. The name of this first package was Personal Home Page Tools, which later became Personal Home Page Construction Kit.

A tool was written to easily embed SQL queries into web pages. It was basically another CGI wrapper that parsed SQL queries and made it easy to create forms and tables based on these queries. This tool was named FI (Form Interpreter).

PHP/FI version 2.0 is a complete rewrite of these two packages combined into a single program. It evolved to a simple programming language embedded inside HTML files. PHP eliminates the need for numerous small Perl cgi programs by allowing you to place simple scripts directly in your HTML files. This speeds up the overall performance of your web pages since the overhead of forking Perl several times has been eliminated. It also makes it easier to manage large web sites by placing all components of a web page in a single html file. By including support for various databases, it also makes it trivial to develop database enabled web pages. Many people find the embedded nature much easier to deal with than trying to create separate HTML and CGI files.

Now PHP/FI is renamed as PHP.

22.3 So, what can I do with PHP ?

The first thing you will notice if you run a page through PHP is that it adds a footer with information about the number of times your page has been accessed (if you have compiled access logging into the binary). This is just a very small part of what PHP can do for you. It serves another very important role as a form interpreter cgi, hence the FI part of the old name. For example, if you create a form on one of your web pages, you need something to process the information on that form. Even if you just want to pass the information to another web page, you will have to have a cgi program do this for you. PHP makes it extremely easy to take form data and do things with it.

22.4 A simple example

Suppose you have a form:

```
<FORM ACTION="/cgi-bin/php.cgi/~userid/display.html" METHOD=POST>
<INPUT TYPE="text" name="name">
<INPUT TYPE="text" name="age">
<INPUT TYPE="submit">
<FORM>
```

Your display.html file could then contain something like:

```
< ?echo "Hi $ name, you are $ age years old!<p>" >
```

It's that simple! PHP automatically creates a variable for each form input field in your form. You can then use these variables in the ACTION URL file.

The next step once you have figured out how to use variables is to start playing with some logical flow tags in your pages. For example, if you wanted to display different messages based on something the user inputs, you would use if/else logic. In our above example, we can display different things based on the age the user entered by changing our display.html to:

```
<?
    if($age>50);
        echo "Hi $name, you are ancient!<p>";
    elseif($age>30);
        echo "Hi $name, you are very old!<p>";
    else;
        echo "Hi $name.";
    endif;
>
```

PHP provides a very powerful scripting language which will do much more than what the above simple example demonstrates. See the section on the PHP Script Language for more information.

You can also use PHP to configure who is allowed to access your pages. This is done using a built−in configuration screen. With this you could for example specify that only people from certain domains would be allowed to see your pages, or you could create a rule which would password protect certain pages. See the Access Control section for more details.

PHP is also capable of receiving file uploads from any RFC−1867 compliant web browser. This feature lets people upload both text and binary files. With PHP's access control and logical functions, you have full control over who is allowed to upload and what is to be done with the file once it has been uploaded. See the File Upload section for more details.

PHP has support for the PostgreSQL database package. It supports embedded SQL queries in your .HTML files.

PHP also has support for the mysql database package. It supports embedded SQL queries in your .HTML files.

22.5 CGI Redirection

Apache 1.0.x Notes

A good way to run PHP is by using a cgi redirection module with the Apache server. Please note that you do not need to worry about redirection modules if you are using the Apache module version of PHP. There are two of these redirection modules available. One is developed by Dave Andersen

- angio@aros.net

and it is available at

- ftp://ftp.aros.net/pub/util/apache/mod_cgi_redirect.c

and the other comes bundled with Apache and is called mod_actions.c. The modules are extremely similar. They differ slightly in their usage. Both have been tested and both work with PHP.

Check the Apache documentation on how to add a module. Generally you add the module name to a file called Configuration. The line to be added if you want to use the mod_actions module is:

Module action_module mod_actions.o

If you are using the mod_cgi_redirect.c module add this line:

Module cgi_redirect_module mod_cgi_redirect.o

Then compile your httpd and install it. To configure the cgi redirection you need to either create a new mime type in your mime.types file or you can use the AddType command in your srm.conf file to add the mime type. The mime type to be added should be something like this:

```
application/x−httpd−php phtml
```

If you are using the mod_actions.c module you need to add the following line to your srm.conf file:

```
Action application/x−httpd−php /cgi−bin/php.cgi
```

If you are using mod_cgi_redirect.c you should add this line to srm.conf:

```
CgiRedirect application/x−httpd−php /cgi−bin/php.cgi
```

Don't try to use both mod_actions.c and mod_cgi_redirect.c at the same time.

Once you have one of these cgi redirection modules installed and configured correctly, you will be able to specify that you want a file parsed by PHP simply by making the file's extension .phtml. Furthermore, if you add index.phtml to your DirectoryIndex configuration line in your srm.conf file then the top−level page in a directory will be automatically parsed by php if your index file is called index.phtml.

Netscape HTTPD

You can automatically redirect requests for files with a given extension to be handled by PHP by using the Netscape Server CGI Redirection module. This module is available in the File Archives on the PHP Home Page. The README in the package explicitly explains how to configure it for use with PHP.

NCSA HTTPD

NCSA does not currently support modules, so in order to do cgi redirection with this server you need to modify your server source code. A patch to do this with NCSA 1.5 is available in the PHP file archives.

22.6 Running PHP from the command line

If you build the CGI version of PHP, you can use it from the command line simply typing: php.cgi filename where filename is the file you want to parse. You can also create standalone PHP scripts by making the first line of your script look something like:

```
#!/usr/local/bin/php.cgi −q
```

The "−q" suppresses the printing of the HTTP headers. You can leave off this option if you like.

22.7 PHPGem package

PHPGem is a PHP−script which accelerates the creation of PHP−scripts for working with tables. It works with different SQL−servers such as PostgreSQL, MySQL, mSQL, ODBC, and Adabas. You input a description of and parameters for your tables' fields (field name, on/off searching in the field, etc.), and PHPGem outputs another PHP−script which will work with the tables (view/add/edit/delete/duplicate entries and search). PHPGem works with multi−level nested tables. PHPGem allows you to specify a level of access for each table and for each field for each user. PHPGem also support images.

PHPGem is at

---

23. Python Interface for PostgreSQL

Python in an interpreted, object orientated scripting language. It is simple to use (light syntax, simple and straighforward statements), and has many extensions for building GUIs, interfacing with WWW, etc. An intelligent web browser (HotJava like) is currently under development (november 1995), and this should open programmers many doors. Python is copyrighted by Stichting S Mathematisch Centrum, Amsterdam, The Netherlands, and is freely distributable. It contains support for dynamic loading of objects, classes, modules, and exceptions. Adding interfaces to new system libraries through C code is straightforward, making Python easy to use in custom settings. Python is a very high level scripting language with X interface. Python package is distributed on Linux cdroms includes most of the standard Python modules, along with modules for interfacing to the Tix widget set for Tk.

PyGreSQL is a python module that interfaces to a PostgreSQL database. It embeds the PostgreSQL query library to allow easy use of the powerful PostgreSQL features from a Python script. PyGreSQL is written by D'Arcy J.M. Cain and Pascal Andre.

- New site of PyGreSQL http://www.druid.net/pygresql/
- Maintained by D'Arcy at http://www.druid.net/~darcy/
- Old site is at ftp://ftp.via.ecp.fr/pub/python/contrib/Database/PyGres95.README
- D'Arcy J.M. Cain darcy@druid.net
- Pascal Andre andre@chimay.via.ecp.fr
- Pascal Andre andre@via.ecp.fr

23.1 Where to get PyGres ?

The home sites of the differents packages are:

- Python ftp://ftp.python.org:/pub/www.python.org/1.5/python1.5b2.tar.gz
- PyGreSQL ftp://ftp.druid.net/pub/distrib/PyGreSQL−2.1.tgz
- Old site ftp://ftp.via.ecp.fr/pub/python/contrib/Database/PyGres95−1.0b.tar.gz

You should anyway try to find some mirror site closer of your site. Refer to the information sources to find these sites. PyGreSQL should reside in the contrib directories of Python and PostgreSQL sites.

23.2 Information and support

If you need information about these packages please check their web sites:

- Python : http://www.python.org/
- PostgreSQL : http://epoch.cs.berkeley.edu:8000/postgres95/index.html
- PyGreSQL ftp://ftp.druid.net/pub/distrib/PyGreSQL−2.1.tgz
- Old site PyGreSQL : http://www.via.ecp.fr/via/products/pygres.html

For support :

- Mailing list for PyGreSQL. You can join by sending email to majordomo@vex.net with the line "subscribe pygresql name@domain" in the body replacing "name@domain" with your own email address.
- Newsgroup for Python : newsgroup comp.lang.python
- PyGreSQL : contact Andre at andre@via.ecp.fr for bug reports, ideas, remarks

23.3 Testing Python interface

See the section −

24. Gateway between PostgreSQL and the WWW − WDB−P95

24.1 About wdb−p95

WDB−P95 − A Web interface to PostgreSQL Databases was created by J. Douglas Dunlop It is at

- New WDB from J Rowe is at http://www.lava.net/beowulf/programming/wdb
- New versions of WWW−WDB is at http://www.eol.ists.ca/~dunlop/wdb−p95/
- For questions or to join Mailing lists contact dunlop@eol.ists.ca

This is a modified version of wdb−1.3a2 which provides a gateway to a the WWW for PostgreSQL. This version also requires a Browser that is capable of handling HTML Tables for the tabular output. This is not required by the original wdb and can be fairly easily reverted.

You can try out CASI Tape and Image Query. You can have a peek at the Form Definition File (FDF) which is used to create the CASI Tape and Image Query too, which includes a JOIN of 2 tables.

This release contains all files necessary to install and run WDB−P95 as an interface to your PostgreSQL databases. To port this system to other database should be relatively easy − provided that it supports standard SQL and has a Perl interface.

24.2 Does the PostgreSQL server, pgperl, and httpd have to be on the same host?

No − the PostgreSQL server does not have to be on the same host. As WDB−P95 is called by the http daemon, they have to be on the same host. − And as WDB−P95 was written to use Pg.pm − pgperl has to be on the same host too. Pgperl was written using the libpq library, so it will be able to access any PostgreSQL server anywhere in the net, just like any other PostgreSQL client. As illustrated below

(WWW Client (Netscape)) => (HTTP Server (NCSA's http) + WDB−P95 + pgperl + libpq)=> (PostgreSQL server)

Curly brackets () represent machines.

Each machine can be of a different type : NT, SUN, HP, ... but you need the libpq interface library for the machine type where you plan to use WDB−P95, as you need it to compile pgperl. (The system was designed to use HTML tables so a recent WWW client is best)

---

25. "C", "C++", ESQL/C language Interfaces and Bitwise Operators for PostgreSQL

25.1 "C" interface

It is included in distribution and is called 'libpq'. Similar to Oracle OCI, Sybase DB−lib, Informix CLI libraries.

25.2 "C++" interface

It is included in distribution and is called 'libpq++'. See the section − Testing C and C++ PostgreSQL interface

25.3 ESQL/C

ESQL/C 'Embedded C Pre−compiler' for PostgreSQL ESQL/C is like Oracle Pro*C, Informix ESQL/C. The PostgreSQL ESQL/C is an SQL application−programming interface (API) enables the C programmer to create custom applications with database−management capabilities. The PostgreSQL ESQL/C allows you to use a third−generation language with which you are familiar and still take advantage of the Structured Query Language (SQL).

ESQL/C consists of the following pieces of software:

- The ESQL/C libraries of C functions provide access to the database server.
- The ESQL/C header files provide definitions for the data structures, constants, and macros useful to the ESQL/C program.
- The ESQL/C preprocessor, is a source−code preprocessor that converts a C file containing SQL statements into an executable file.

It is at

- ESQL/C for PostgreSQL is already included in the distribution.
- Main site ftp://ftp.lysator.liu.se/pub/linus
- Email : linus@epact.se

See the section − Testing Embedded SQL/C interface to PostgreSQL

To use Vim color editor to edit 'ecpg' files (*.pgc), you must do the following:−

```
bash$ su - postgres
bash$ mkdir $HOME/vim
```

```
And create a file '$HOME/vim/myfilestypes.vim' which has these lines

    " myfiletypefile
        au! BufRead,BufNewFile *.pgc    set filetype=esqlc
```

You should have $HOME/.gvimrc file. If not create one, refer to Vim howto doc at http://metalab.unc.edu/LDP/HOWTO/Vim−HOWTO.html Put the following line in $HOME/.gvimrc file

```
let myfiletypefile = "~/vim/myfiletypes.vim"
```

Now, if you edit with

```
bash$ gvim sample.pgc
```

you will get the color syntax highlighted.

25.4 BitWise Operators for PostgreSQL

Bitwise operators was written by Nicolas Moldavsky

- nico@overnet.com.ar

"C" functions that implement bitwise operators (AND, OR, XOR, bit complement) on pgsql. Get them by anonymous FTP from

- ftp://ftp.overnet.com.ar/pub/utils/linux/bitpgsql.tgz

Makefile for Linux is included.

26. Japanese Kanji Code for PostgreSQL

It is at the following site

- ftp://ftp.sra.co.jp/pub/cmd/postgres/

27. PostgreSQL Port to Windows 95/Windows NT

PostgreSQL binaries for Windows NT is available from :

- Windows NT PostgreSQL binaries http://www.askesis.nl
- http://www.postgresql.org

Download the binaries and unpack and follow instructions in Install PostgreSQL from step 13.

If you want to re−compile the source tree then follow the instructions given below. Porting to NT is done using Cygnus cygwin32 package which has gcc, gmake for Win NT/95.

- Cygwin 32 package is at http://www.cygnus.com/misc/gnu−win32

At this site and get the file cdk.exe (self−extractor file for gnu−win32)

27.1 Authors of NT port

The authors of Windows NT port of PostgreSQL are −

- Daniel Horak horak@mmp.plzen−city.cz
- Joost Kraaijeveld JKraaijeveld@askesis.nl
- Kevin Lo kevlo@FreeBSD.org
- Home page of NT port is at http://www.freebsd.org/~kevlo/postgres/portNT.html

27.2 Install the Cygwin package

1. Download ftp://go.cygnus.com/pub/sourceware.cygnus.com/cygwin/latest/full.exe
2. Run full.exe and install in c:\Unix\Root directory.
3. Run Cygwin, Type 'mount −−help' for docs. You can use −f switch to force mount. And then run "umount / " and "mount c:\Unix\Root /"

27.3 Tuneup Bash Window

After installing the Cygwin package, do the following to setup the working environment:

1. Install the Vi editor 'Vim'. See http://metalab.unc.edu/LDP/HOWTO/Vim−HOWTO.html

2. The default window of cygwin bash is black−background window with 24 lines. To set the background color and size of bash window, click on NT−Start−>Control−panel−>MS DOS console and change the background color to grey and size to 70 lines.

(OR) right click on Window titlebar and change property.

3. Edit cygnus.bat in c:\cygnus\cywinb20 and set the following −

```
set HOME=c:\cygnus\cygwinb20
bash --login
```

And also edit the /.bash_profile and put these lines

```
set -o vi
PATH=$PATH:/usr/local/bin:/usr/bin
export PATH
```

4. To enable the command−line history editing give −

bash$ set −o vi

Using the l,k,j,h you can use the vi commands to edit the command line history commands. You can repeat or modify previous commands, saves typing time.

5. You can do mount of drives/directories using this command −

```
bash$ umount /
bash$ mount "c:\cygnus"  /
bash$ mount "c:\cygnus\cygwin-b20\postgres" /usr/local/pgsql
```

6. See online help with −

```
bash$ mount --help
bash$ ls --help
```

27.4 Install the Andy Piper tools

1. Go to ftp://ftp.xemacs.org/pub/xemacs/aux/ and download cygwin−b20−local.tar.bz2 in the c:/Unix/Root directory.
2. cd c:/Unix/Root; bunzip2 cygwin−b20−local.tar.bz2
3. tar −xvf cygwin−b20−local.tar
4. cd /local/bin; sh check_cygwin_setup.sh
5. After doing step 4, you see the following message:

```
You don't have /bin would you like to mount cygwin as /bin?"
[ y/n ]
Select 'n',  and the other options are selected 'y'
```

6. mount c:/Unix/Root/cygwin−b20/H−i586−cygwin32/i586−cygwin32/bin /bin
7. cd c:/Unix/Root/cygwin−b20/H−i586−cygwin32/i586−cygwin32; mkdir libexec share man etc sbin info
8. cp −R /local/{ bin,libexec,share,man,etc,sbin,info,include }

27.5 Install Ludovic Lange's Cygwin32 IPC package

1. Go to http://www.multione.capgemini.fr/tools/pack_ipc and download cygwin32_ipc−1.03.tgz in c:/Unix/Root directory.
2. tar −zxvf cygwin32_ipc−1.03.tgz
3. cd cygwin32_ipc−1.03/src and run 'make'
4. mkdir −p c:/usr/local/{bin,include,lib,include/sys}

```
cp /cygwin32_ipc-1.03/bin/* c:/usr/local/bin
cp /cygwin32_ipc-1.03/include/sys/* c:/usr/local/include/sys
cp /cygwin32_ipc-1.03/lib/* c:/usr/local/lib
cp c:/usr/local/bin/* /bin
cp c:/Unix/Root/cygwin-b20/H-i586-cygwin32/bin/* /bin
```

5. mount c:/usr/local/bin /usr/local/bin

```
mount c:/usr/local/include /usr/local/include
mount c:/usr/local/lib /usr/local/lib
cp /local/lib/* /usr/local/lib
```

27.6 Install PostgreSQL

1. Download the latest PostgreSQL source code
2. Postgres treats all files as binary files so the lf/cf stuff appeard, so we do steps 2, 3, 4, and 5:

```
mkdir -p c:/Postgres/{Source,Binary}
mkdir c:/Postgres/Binary/pgsql
mkdir -p /usr/src/pgsql
mkdir -p /usr/local/pgsql
```

3. Copy Postgres source code to c:/Postgres/Source directory, then tar −zxvf postgresql−6.5.3.tar.gz
4. mv postgresql−6.5.3 pgsql
5. Mount directories now −

```
mount -b c:/Postgres/Binary/pgsql /usr/local/pgsql
mount c:/Postgres/Source/pgsql /usr/src/pgsql
mount c:/Unix/Root/cygwin-b20/share /sw/cygwin-b20/share
```

6. mkdir −p /usr/local/pgsql/{bin,include,lib,data}
7. cd /usr/src/pgsql/src/win32
8. Copy header files −

```
cp un.h c:/Unix/Root/cygwin-b20/H-i586-cygwin32/i586-cygwin32/include/sys
cp endian.h c:/Unix/Root/cygwin-b20/H-i586-cygwin32/i586-cygwin32/include
cp tcp.h c:/Unix/Root/cygwin-b20/H-i586-cygwin32/i586-cygwin32/include/netinet
```

9. ln −s /usr/local/lib /usr/src/pgsql/src/backend/libpostgres.a
10. cd /usr/src/pgsql/src, then run './configure'
11. make > make.txt 2>&1
12. make install > make.install.txt 2>&1
13. cp /usr/local/pgsql/lib/pq.dll /usr/local/pgsql/bin
14. After the make install you had to change all the text files in the bin and lib diectories so that they did not contain cr/lf and eof stuff.
15. Using any editor to create .bashrc in / directory as belows:

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

16. source /.bashrc, then run 'initdb −−username=xxxx' Note that the owner of the DB system have to be different from root/administrator
17. Edit the file /usr/local/pgsql/data/pg_hba.conf, such as:

```
host        all     163.17.11.109   255.255.255.0   trust
```

18. ipc−daemon.exe&
19. postmaster −i&
20. Run ' psql −h host_name template1'

28. Mailing Lists

28.1 E−mail account for PostgreSQL

Get free e−mail accounts from

- In Yahoo http://www.yahoo.com click on e−mail
- In Lycos http://www.lycos.com click on new e−mail accounts
- In hotmail http://www.hotmail.com click on new e−mail accounts

Subscribe to PostgreSQL mailing list and Yahoo has additional feature of creating a seperate folder for PostgreSQL e−mails, so that your regular e−mail is not cluttered. Select menu Email− > Options− > Filters and pick seperate folder for email. With this e−mail account you can access mail from anywhere in the world as long as you have access to a web page.

If you have any other e−mail, you can use "Mail Filters" to receive automatically the PostgreSQL mails into a seperate folder. This will avoid mail cluttering.

28.2 English Mailing List

See the Mailing Lists Item on the main web page at :

- http://www.postgresql.org/
- Email questions to: pgsql−questions@postgresql.org
- Developers pgsql−hackers@postgresql.org
- Port specific questions pgsql−ports@postgresql.org
- Documentation questions pgsql−docs@postgresql.org

You will get the answers/replies back by e−mail in less than a day.

You can also subscribe to mailing lists. To subscribe or unsubscribe from the list, send mail to

- pgsql−questions−request@postgresql.org
- pgsql−hackers−request@postgresql.org
- pgsql−ports−request@postgresql.org
- pgsql−docs−request@postgresql.org

The body of the message should contain the single line

subscribe

(or)

unsubscribe

28.3 Archive of Mailing List

Also mailing lists are archived in html format at the following location −

- Date−wise listing available via MHonarc via the WWW at http://www.postgresql.org/mhonarc/pgsql−questions
- ftp://ftp.postgresql.org directory is /pub/majordomo

There is also search engine available on the PostgreSQL main web site specifically for pgsql questions.

28.4 Spanish Mailing List

Now there is an "unofficial" list of postgreSQL in Spanish. To subscribe the user has to send a message to:

- majordomo@tlali.iztacala.unam.mx

The body of the message should contain the single line:

inscripcion pgsql−ayuda

---

29. Documentation and Reference Books

29.1 User Guides and Manuals

The following are included in the PostgreSQL distribution in the postscript, HTML formats and unix man−pages. They are located in /usr/doc/postgresql* directory. If you have access to internet, you can find the documents listed below at http://www.postgresql.org/docs and at http://www.postgresql.org/users−lounge/docs.

- "Installation Guide"
- "User Guide" for PostgreSQL
- "Implementation Guide" detailing database internals of PostgreSQL.
- Online manuals.
- Online manuals in HTML formats.
- Also manuals in Postscript format for printing hard copies.

29.2 Online Documentation

- Listing and description of default data types and operators
  ```
  Is a part of PSQL command
  ```
- Listing of supported SQL keywords
  ```
  There is a script in the /tools directory in source code tree.
  ```
- Listings of supported statements −
  ```
  Use the command psql \h
  ```
- Basic relational database concepts under PostgreSQL (implementation) and several online examples (queries) −

```
Look at the regression tests at src/test. There you can find the directories
regress/sql and suite/*.sql and also see
<ref id="Examples RPM">
```

* Tutorial for PostgreSQL.

```
SQL tutorial scripts is in the directory src/tutorial
```

See also "SQL Tutorial for beginners" in Appendix B of this document Appendix B

29.3 Useful Reference Textbooks

* "Understanding the New SQL: A Complete Guide" − by Jim Melton and Alan R.Simon

```
Morgan Kaufman Publisher is one of best SQL books. This deals with SQL92.
```

* "A Guide to THE SQL STANDARD" − by C.J.Date

```
Addison-Wesley Publishing company is also a good book. Very popular book for SQL.
```

* SQL − The Standard Handbook, November 1992

```
Stephen Cannan and Gerard Otten
McGraw-Hill Book Company Europe , Berkshire, SL6 2QL, England
```

* SQL Instant Reference, 1993

```
Martin Gruber, Technical Editor: Joe Celko
SYBEX Inc.  2021 Challenger Drive Alameda, CA 94501
```

* C.J.Date, "An introduction to Database Systems" (6th Edition), Addison−Wesley, 1995, ISBN 0−201−82458−2

```
This book is the Bible of Database Management Systems.
The book details normalization, SQL, recovery, concurrency, security,
integrity, and extensions to the original relational model, current issues
like client/server systems and the Object Oriented model(s). Many
references are included for further reading. Recommended for most users.
```

* Stefan Stanczyk, "Theory and Practice of Relational Databases", UCL Press Ltd, 1990, ISBN 1−857−28232−9

```
Book details theory of relational databases, relational algebra, calculus
and normalisation. But it does not cover real world issues and examples
beyond simple examples. Recommended for most users.
```

* "The Practical SQL Handbook" Third Edition, Addison Wesley Developers Press ISBN 0−201−44787−8

```
Recommended for most users.
```

* Michael Stonebraker, "Readings in Database Systems", Morgan Kaufmann, 1988, ISBN 0−934613−65−6

```
This book is a collection of papers that have been published over the
years on databases. It's not for the casual user but it is really a
reference for advanced (post-graduate) students or database system
developers.
```

* C.J.Date, "Relational Database − Selected Readings", Addison−Wesley, 1986, ISBN 0−201−14196−5

```
This book is a collection of papers that have been published over the
years on databases. It's not for the casual user but it is really a
reference for advanced (post-graduate) students or database system
developers.
```

* Nick Ryan and Dan Smith, "Database Systems Engineering", International Thomson Computer Press, 1995, ISBN 1−85032−115−9

```
This book goes into the details of access methods, storage techniques.
```

* Bipin C. Desai, "An introduction to Database Systems", West Publishing Co., 1990, ISBN 0−314−66771−7

```
It's not for the casual user but it is for advanced (post-graduate)
students or database system developers.
```

* Joe Celko "INSTANT SQL Programming"

```
Wrox Press Ltd.
```

```
Unit 16, 20 James Road, Tyseley
Birmingham, B11 2BA, England
1995
```

• Michael Gorman "Database Management Systems: Understanding and Applying Database"

```
Technology
QED and John Wiley
1991
```

• Michael Gorman "Enterprise Database for a Client/Server Environment" QED and John Wiley

```
Presents the requirements of building client/server database
applications via repository metamodels and the use of ANSI standard SQL
1993
```

Hundreds of other titles on SQL are available! Check out a bookstore.

29.4 ANSI/ISO SQL Specifications documents − SQL 1992, SQL 1998

ANSI/ISO SQL specifications documents can be found at these sites listed below −

- http://www.naiua.org/std−orgs.html
- http://www.ansi.org/docs and click on file cat_c.html and search with "Database SQL"
- SQL92 standard http://www.jcc.com and click on file sql_stnd.html
- ANSI/ISO SQL specifications http://www.contrib.andrew.cmu.edu/~shadow/sql.html You will find SQL Reference here.

29.5 Syntax of ANSI/ISO SQL 1992

See Appendix A of this document Appendix A

29.6 Syntax of ANSI/ISO SQL 1998

The SQL 1998 (SQL 3) specification is still under development. See 'Electronic Access to the SQL3 Working Draft' of this document at SQL 1998

29.7 SQL Tutorial for beginners

See Appendix B of this document Appendix B

29.8 Temporal Extension to SQL92

- Document for Temporal Extension to SQL−92 ftp://FTP.cs.arizona.edu/tsql/tsql2/
- Temporal SQL−3 specification ftp://FTP.cs.arizona.edu/tsql/tsql2/sql3/

This directory contains the language specification for a temporal extension to the SQL−92 language standard. This new language is designated TSQL2.

The language specification present here is the final version of the language.

Correspondence may be directed to the chair of the TSQL2 Language Design Committee, Richard

T.Snodgrass, Department of Computer Science, University of Arizona, Tucson, AZ 85721,

- rts@cs.arizona.edu

The affiliations and e−mail addresses of the TSQL2 Language Design Committee members may be found in a separate section at the end of the language specification.

The contents of this directory are as follows.

spec.dvi,.ps TSQL2 Language Specification, published in September, 1994

bookspec.ps TSQL2 Language Specification, as it appears in the TSQL2 book, published in September, 1995 (see below).

sql3 change proposals submitted to the ANSI and ISO SQL3 committees.

Associated with the language specification is a collection of commentaries which discuss design decisions, provide examples, and consider how the language may be implemented. These commentaries were originally proposals to the TSQL2 Language Design Committee. They now serve a different purpose: to provide examples of the constructs, motivate the many decisions made during the language design, and compare TSQL2 with the many other language proposals that have been made over the last fifteen years. It should be emphasized that these commentaries are not part of the TSQL2 language specification per se, but rather supplement and elaborate upon it. The language specification proper is the final word on TSQL2.

The commentaries, along with the language specification, several indexes, and other supporting material, has been published as a book:

Snodgrass, R.T., editor, The TSQL2 Temporal Query Language, Kluwer Academic Publishers, 1995, 674+xxiv pages.

The evaluation commentary appears in the book in an abbreviated form; the full commentary is provided in this directory as file eval.ps

The file tl2tsql2.pl is a prolog program that tranlates allowed temporal logic to TSQL2. This program was written by Michael Boehlen

- boehlen@iesd.auc.dk

He may be contacted for a paper that describes this translation. This is a rather dated version of that program. Newer versions are available at

- http://www.cs.auc.dk/general/DBS/tdb/TimeCenter/Software

(the TimeDB and Tiger systems).

29.9 Part 0 − Acquiring ISO/ANSI SQL Documents

This document shows you how to (legally) acquire a copy of the SQL−92 standard and how to acquire a copy of the "current" SQL3 Working Draft.

The standard is copyrighted ANSI standard by ANSI, the ISO standard by ISO.

There are two (2) current SQL standards, an ANSI publication and an ISO publication. The two standards are word−for−word identical except for such trivial matters as the title of the document, page headers, the phrase

"International Standard" vs "American Standard", and so forth.

Buying the SQL−92 Standard

The ISO standard, ISO/IEC 9075:1992, Information Technology − Database Languages − SQL, is currently (March, 1993) available and in stock from ANSI at:

```
American National Standards Institute
1430 Broadway
New York, NY 10018 (USA)
Phone (sales): +1.212.642.4900
```

at a cost of US$230.00. The ANSI version, ANSI X3.135−1992, American National Standard for Information Systems − Database Language SQL, was not available from stock at this writing, but was expected to be available by some time between late March and early May, 1993). It is expected to be be priced at US$225.00.

If you purchase either document from ANSI, it will have a handling charge of 7% added to it (that is, about US$9.10). Overseas shipping charges will undoubtedly add still more cost. ANSI requires a hardcopy of a company purchase order to accompany all orders; alternately, you can send a check drawn on an US bank in US dollars, which they will cash and clear before shipping your order. (An exception exists: If your organization is a corporate member of ANSI, then ANSI will ship the documents and simply bill your company.)

The ISO standard is also available outside the United States from local national bodies (country standardization bodies) that are members of either ISO (International Organization for Standardization) or IEC (International Electrotechnical Commission). Copies of the list of national bodies and their addresses are available from ANSI or from other national bodies. They are also available from ISO:

```
International Organization for Standardization
Central Secretariat
1, rue de Varembi
CH−1211 Genhve 20
Switzerland
```

If you prefer to order the standard in a more convenient and quick fashion, you'll have to pay for the privilege. You can order ISO/IEC 9075:1992, Information Technology − Database Languages − SQL, from:

```
Global Engineering Documents
2805 McGaw Ave
Irvine, CA 92714 (USA)
USA
Phone (works from anywhere): +1.714.261.1455
Phone (only in the USA): (800)854−7179
```

for a cost of US$308.00. I do not know if that includes shipping or not, but I would guess that international shipping (at least) would cost extra. They will be able to ship you a document fairly quickly and will even accept "major credit cards". Global does not yet have the ANSI version nor do they have a price or an expected date (though I would expect it within a few weeks following the publication by ANSI and at a price near US$300.00).

Buying a copy of the SQL3 Working Draft

You can purchase a hardcopy of the SQL3 working draft from the ANSI X3 Secretariat, CBEMA (Computer and Business Equipment Manufacturers Association). They intend to keep the "most recent" versions of the SQL3 working draft available and sell them for about US$60.00 to US$65.00. You can contact CBEMA at:

```
CBEMA, X3 Secretariat
Attn: Lynn Barra
1250 Eye St.
Suite 200
Washington, DC 20005 (USA)
```

Lynn Barra can also be reached by telephone at +1.202.626.5738 to request a copy, though mail is probably more courteous.

Electronic Access to the SQL3 Working Draft

The most recent version (as of the date of this writing) of the SQL3 (both ANSI and ISO) working draft (and all of its Parts) is available by "anonymous ftp" or by "ftpmail" on:

```
   gatekeeper.dec.com

at

   /pub/standards/sql/
```

In this directory are a number of files. There are PostScript. files and "plain text" (not prettily formatted, but readable on a screen without special software).

In general, you can find files with names like:

```
sql−bindings−mar94.ps
sql−bindings−mar94.txt
sql−cli−mar94.ps
sql−cli−mar94.txt
sql−foundation−mar94.ps
sql−foundation−mar94.txt
sql−framework−mar94.ps
sql−framework−mar94.txt
sql−psm−mar94.ps
sql−psm−mar94.txt
```

As new versions of the documents are produced, the "mar94" will change to indicate the new date of publication (e.g., "aug94" is the expected date of the next publication after "mar94").

In addition, for those readers unable to get a directory listing by FTP, we have placed a file with the name:

```
ls
```

into the same directory. This file (surprise!) contains a directory listing of the directory.

Retrieving Files Directly Using ftp

This is a sample of how to use FTP. Specifically, it shows how to connect to gatekeeper.dec.com, get to the directory where the base document is kept, and transfer the document to your host. Note that your host must have Internet access to do this. The login is 'ftp' and the password is your email address (this is sometimes referred to as 'anonymous ftp'). The command 'type binary' is used to ensure that no bits are stripped from the file(s) received. 'get' gets one file at a time. Comments in the script below are inside angle brackets < like so

> .

```
% ftp gatekeeper.dec.com
Connected to gatekeeper.dec.com.
220- *** /etc/motd.ftp ***
     Gatekeeper.DEC.COM is an unsupported service of DEC Corporate Research.
     <...this goes on for a while...>
220 gatekeeper.dec.com FTP server (Version 5.83 Sat ... 1992) ready.
Name (gatekeeper.dec.com:<yourlogin here>): ftp  <anonymous also works>
331 Guest login ok, send ident as password.
Password: <enter your email address here >
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.  <or whatever>
Using binary mode to transfer files.
ftp> cd pub/standards/sql
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 9529
-r--r--r--  1 root     system     357782 Feb 25 10:18 x3h2-93-081.ps
-r--r--r--  1 root     system     158782 Feb 25 10:19 x3h2-93-081.txt
-r--r--r--  1 root     system     195202 Feb 25 10:20 x3h2-93-082.ps
-r--r--r--  1 root     system      90900 Feb 25 10:20 x3h2-93-082.txt
-r--r--r--  1 root     system    5856284 Feb 25 09:55 x3h2-93-091.ps
-r--r--r--  1 root     system    3043687 Feb 25 09:57 x3h2-93-091.txt
226 Transfer complete.
ftp> type binary
200 Type set to I.
ftp> get x3h2-93-082.txt
200 PORT command successful.
150 Opening BINARY mode data connection for x3h2-93-082.txt (90900 bytes).
226 Transfer complete.
90900 bytes received in 0.53 seconds (166.11 Kbytes/s)
ftp> quit
% <the file is now in your directory as x3h2-93-082.txt>
```

Retrieving Files Without Direct ftp Support

Digital Equipment Corporation, like several other companies, provides ftp email service. The response can take several days, but it does provide a service equivalent to ftp for those without direct Internet ftp access. The address of the server is:

ftpmail@decwrl.dec.com

The following script will retrieve the PostScript for the latest version of the SQL3 document:

```
reply joe.programmer@imaginary-corp.com
connect gatekeeper.dec.com anonymous
binary
compress
```

The following script will retrieve the PostScript for the latest version of the SQL3 document:

```
reply joe.programmer@imaginary-corp.com
connect gatekeeper.dec.com anonymous
binary
compress
uuencode
```

```
chdir /pub/standards/sql
get x3h2-93-091.ps
quit
```

The first line in the script commands the server to return the requested files to you; you should replace "joe.programmer@imaginary−corp.com" with your Internet address. The file in this example, x3h2−93−091.ps, is returned in "compress"ed "uuencode"d format as 34 separate email messages. If your environment does not provide tools for reconstructing such files, then you could retrieve the file as plain text with the following script:

```
reply joe.programmer@imaginary-corp.com
connect gatekeeper.dec.com anonymous
chdir /pub/standards/sql
get x3h2-93-091.ps
quit
```

But be warned, the .ps file will probably be sent to you in more than 70 parts!

To retrieve any particular file, other than x3h2−93−091.ps, simply replace "x3h2−93−091.ps" with the name of the desired file. To get a directory listing of all files available, replace "get x3h2−93−091.ps" with "dir".

29.10 Part 1 − ISO/ANSI SQL Current Status

This chapter is a source of information about the SQL standards process and its current state.

Current Status:

Development is currently underway to enhance SQL into a computationally complete language for the definition and management of persistent, complex objects. This includes: generalization and specialization hierarchies, multiple inheritance, user defined data types, triggers and assertions, support for knowledge based systems, recursive query expressions, and additional data administration tools. It also includes the specification of abstract data types (ADTs), object identifiers, methods, inheritance, polymorphism, encapsulation, and all of the other facilities normally associated with object data management.

In the fall of 1996, several parts of SQL3 went through an ISO CD ballot. Those parts were SQL/Framework, SQL/Foundation, and SQL/Bindings. Those ballots failed (as expected) with 900 or so comments. In Late January, there was an ISO DBL editing meeting that processed a large number of problem solutions that were either included with ballot comments or submitted as separate papers. Since the DBL editing meeting was unable to process all of the comments, the editing meeting has been extended. The completion of the editing meeting is scheduled for the end of July, 1997, in London.

Following the July editing meeting, the expectation is that a Final CD ballot will be requested for these parts of SQL. The Final CD process will take about 6 months and a DBL editing meeting, after which there will be a DIS ballot and a fairly quick IS ballot.

The ISO procedures have changed since SQL/92, so the SQL committees are still working through the exact details of the process.

If everything goes well, these parts of SQL3 will become an official ISO/IEC standard in late 1998, but the schedule is very tight.

In 1993, the ANSI and ISO development committees decided to split future SQL development into a multi−part standard. The Parts are:

- Part 1: Framework A non−technical description of how the document is structured.
- Part 2: Foundation The core specification, including all of the new ADT features.
- Part 3: SQL/CLI The Call Level Interface.
- Part 4: SQL/PSM The stored procedures specification, including computational completeness.
- Part 5: SQL/Bindings The Dynamic SQL and Embedded SQL bindings taken from SQL−92.
- Part 6: SQL/XA An SQL specialization of the popular XA Interface developed by X/Open
- Part 7:SQL/TemporalAdds time related capabilities to the SQL standards.

In the USA, the entirety of SQL3 is being processed as both an ANSI Domestic ("D") project and as an ISO project. The expected time frame for completion of SQL3 is currently 1999.

The SQL/CLI and SQL/PSM are being processed as fast as possible as addendums to SQL−92. In the USA, these are being processed only as International ("I") projects. SQL/CLI was completed in 1995. SQL/PSM should be completed sometime in late 1996.

In addition to the SQL3 work, a number of additional projects are being persued:

- SQL/MM An ongoing effort to define standard multi−media packages using the SQL3 ADT capabilities.
- Remote Data Access (RDA)

Standards Committee and Process

There are actually a number of SQL standards committees around the world. There is an international SQL standards group as a part of ISO. A number of countries have committees that focus on SQL. These countries (usually) send representatives to ISO/IEC JTC1/SC 21/WG3 DBL meetings. The countries that actively participate in the ISO SQL standards process are:

- Australia
- Brazil
- Canada
- France
- Germany
- Japan
- Korea
- The Netherlands
- United Kingdom
- United States

NIST Validation

SQL implementations are validated (in the Unites States) by the National Institute of Standards and Training (NIST). NIST currently has a validation test suite for entry level SQL−92. The exact details of the NIST validation requirements are defined as a Federal Information Processing Standard (FIPS). The current requirements for SQL are defined in FIPS 127−2. The Postscript and Text versions of this document can be retrieved from NIST. The current SQL Validated Products List can also be retrieved from NIST.

Standard SQL Publications and Articles

There are two versions of the SQL standard. Both are available from ANSI:

* ISO/IEC 9075:1992, "Information Technology −−− Database Languages −−− SQL"
* ANSI X3.135−1992, "Database Language SQL"

The two versions of the SQL standard are identical except for the front matter and references to other standards. Both versions are available from:

```
American National Standards Institute
1430 Broadway
New York, NY 10018
USA
Phone (sales): +1.212.642.4900
```

In additon to the SQL−92 standard, there is now a Technical Corrigendum (bug fixes):

```
* Technical Corrigendum 1:1994 to ISO/IEC 9075:1992
```

TC 1 should also be available from ANSI. There is only an ISO version of TC 1 −− it applies both to the ISO and ANSI versions of SQL−92.

In addition to the standards, several books have been written about the 1992 SQL standard. These books provide a much more readable description of the standard than the actual standard.

Related Standards

A number of other standards are of interest to the SQL community. This section contains pointers to information on those efforts. These pointers will be augmented as additional information becomes available on the web.

* SQL Environments (FIPS 193)
* Next Generation Repository Systems (X3H4) – a News Release calling for particpation in "Developing Standards for the Next Generation Repository Systems."

29.11 Part 2 – ISO/ANSI SQL Foundation

A significant portion of the SQL3 effort is in the SQL Foundation document:

* Base SQL/PSM capabilities (moved form SQL/PSM−92)
* New data types
* Triggers
* Subtables
* Abstract Data Types (ADT)
* Object Oriented Capabilities

There are several prerequisites to the object oriented capabilities:

* Capability of defining complex operations
* Store complex operations in the database
* External procedure calls - Some operations may not be in SQL, or may require external interactions

These capabilities are defined as a part of SQL/PSM

A great deal of work is currently being done to refine the SQL−3 object model and align it with the object model proposed by ODMG. This effort is described in the X3H2 and ISO DBL paper: Accomodating SQL3 and ODMG. A recent update on the SQL3/OQL Merger is also available.

SQL3 Timing

Work on SQL3 is well underway, but the final standards is several years away.

- International ballot to progress SQL3 Foundation from Working Draft to Committee Draft (CD) taking place fall, 1996.
- Ballot is expected to generate numerous comments
- A second CD ballot is likely to be required
- Draft International Standard ballot is likely to be take place in mid 1998
- International Standard could be completed by mid 1999.

The ANSI version of the standard will be on a similar schedule.

29.12 Part 3 − ISO/ANSI SQL Call Level Interface

The SQL/CLI is a programing call level interface to SQL databases. It is designed to support database access from shrink−wrapped applications. The CLI was originally created by a subcommittee of the SQL Access Group (SAG). The SAG/CLI specification was published as the Microsoft Open DataBase Connectivity (ODBC) specification in 1992. In 1993, SAG submitted the CLI to the ANSI and ISO SQL committees. (The SQL Access Group has now merged with X/Open consortium.)

SQL/CLI provides an international standard for:

- Implementation−independent CLI to access SQL databases
- Client−server tools can easily access database through dynamic Link Libraries
- Supports and encourages rich set of Client−server tools

SQL/CLI Timing

For the standards process, SQL/CLI is being processed with blinding speed.

- SQL/CLI is an addendum to 1992 SQL standard (SQL−92)
- Completed as an ISO standard in 1995
- ISO/IEC 9075−3:1995 Information technology −− Database languages −− SQL −− Part 3: Call−Level Interface (SQL/CLI)
- Current SQL/CLI effort is adding support for SQL3 features

29.13 Part 4 − ISO/ANSI SQL Persistent Stored Modules

SQL/PSM expands SQL by adding:

- Procedural language extensions
- Multi−statement and Stored Procedures
- External function and procedure calls

In addition to being a valuable application development tool, SQL/PSM provides the foundation support for the object oriented capabilities in SQL3.

Multi−statement and Stored Procedures

Multi−statement and stored procedures offer a variety of advantages in a client/server environment:

- Performance − Since a stored procedure can perform multiple SQL statements, network interaction with the client are reduced.
- Security − An user can be given the right to call a stored procedure that updates a table or set of tables but denied the right to update the tables directly
- Shared code − The code in a stored procedure does not have to be rewritten and retested for each client tool that accesses the database.
- Control − Provides a single point of definition and control for application logic.

Procedural Language Extensions

Procedural language add the power of a traditional programming language to SQL through flow control statements and a variety of other programming constructs.

Flow Control Statements

- If−then−else
- Looping constructs
- Exception handling
- Case statement
- Begin−End blocks

The procedural language extensions include other programming language constructs:

- Variable declarations
- Set statements for value assignment
- Get diagnostics for process and status information

In addition, all of the traditional SQL statements can be included in multi−statement procedures.

External Procedure and Function Calls

One feature frequently mentioned in the wish lists for many database products, and implemented in some, is a capability augmenting the built−in features with calls to user−written procedures external to the database software.

- Allows a particular site or application to add their own database functions
- Can be used throughout the database applications

The benefit of this capability is that it gives the database (and therefore database applications) access to a rich set of procedures and functions too numerous to be defined by a standards committee.

SQL/PSM Timing

SQL/PSM is proceeding quickly:

- SQL/PSM is an addendum to SQL−92
- International ballot to progress SQL/PSM from a Draft International Standard to an International Standard ended January, 1996.
- Editing meeting in May, 1996 did not resolve all of the comments
- Continuation of PSM Editing meeting is scheduled for September 30 through October 4, 1996
- The schedule is tight but there is a chance that PSM will be published with a 1996 date.
- The official designation will be: ISO/IEC DIS 9075−4:199? Information technology −− Database languages −− SQL −− Part 4: SQL Persistent Stored Modules (SQL/PSM)
- Work is well underway on adding SQL/PSM support for SQL3 features.

29.14 Part 5 − ISO/ANSI SQL/Bindings

For ease of reference, the programming language bindings have been pulled out into a separate document. The current version is simply an extract of the dynamic and embedded bindings from SQL−92.

A variety of issues remain unresolved for the programming language bindings.

For traditional programming language, mappings exist for the SQL−92 datatypes. However, mappings must be defined between SQL objects and programming language variables.

For object oriented languages, mapping must be defined for the current SQL datatypes and between the SQL object model and the object model of the object−oriented language.

The object model needs to stabilize before these can be addressed.

The language bindings will be completed as a part of SQL3.

29.15 Part 6 − ISO/ANSI SQL XA Interface Specialization (SQL/XA)

This specification would standardize an application program interface (API) between a global Transaction Manager and an SQL Resource Manager. It would standardize the function calls, based upon the semantics of ISO/IEC 10026, "Distributed Transaction Processing", that an SQL Resource Manager would have to support for two−phase commit. The base document is derived from an X/Open publication, with X/Open permission, that specifies explicit input and output parameters and semantics, in terms of SQL data types, for the following functions: xa_close, xa_commit, xa_complete, xa_end, xa_forget, xa_open, xa_prepare, xa_recover, xa_rollback, and xa_start.

ISO is currently attempting to fast−track the X/Open XA specification. The fast−track process adopts a current industry specification with no changes. The XA fast−track ballot at the ISO SC21, JTC 1 level started on April 27, 1995 and ends on October 27, 1995. If the XA specification is approved by 75% of the votes, and by 2/3 of the p−members of JTC 1, it will become an International Standard. If the fast−track ballot is approved, SQL/XA could become a standard in 1996.

29.16 Part 7 − ISO/ANSI SQL Temporal

Temporal SQL deals with time−related data. The concept is that it is useful to query data to discover what it looked like at a particular point in time. Temporal SQL is a December, 1994 paper by Rick Snodgrass describing the concepts.

X3 Announces the Approval of a New Project, ISO/IEC 9075 Part 7: SQL/Temporal is a press release related to SQL/Temporal.

```
--------------------------------------------------------------------------
                                Temporal SQL
                                ************
Rick Snodgrass (chair of the TSQL2 committee)
31-Dec-1994
```

Several people have questioned the need for additional support for time in SQL3 (as proposed by DBL RIO−75, requesting a new part of SQL to support temporal databases). The claim is that abstract data types (ADT's) are sufficient for temporal support. In this informational item, I argue, using concrete examples, that using columns typed with abstract data types is inadequate for temporal queries. In particular, many common temporal queries are either difficult to simulate in SQL, or require embedding SQL in a procedural language. Alternatives are expressed in TSQL2, a temporal extension to SQL−92.

INTRODUCTION

Valid−time support goes beyond that of a temporal ADT. With the later, a column is specified as of a temporal domain, such as DATE or INTERVAL (examples will be given shortly). With valid time, the rows of a table vary over time, as reality changes. The timestamp associated with a row of a valid−time table is interpreted by the query language as the time when the combination of values of the columns in the row was valid. This implicit timestamp allows queries to be expressed succinctly and intuitively.

A CASE STUDY − STORING CURRENT INFORMATION

The University of Arizona's Office of Appointed Personnel has some information in a database, including each employee's name, their current salary, and their current title. This can be represented by a simple table.

```
Employee(Name, Salary, Title)
```

Given this table, finding an employee's salary is easy.

```
SELECT Salary
FROM Employee
WHERE Name = 'Bob'
```

Now the OAP wishes to record the date of birth. To do so, a column is added to the table, yielding the following schema.

```
Employee(Name, Salary, Title, DateofBirth DATE)
```

Finding the employee's date of birth is analogous to determining the salary.

```
SELECT DateofBirth
FROM Employee
WHERE Name = 'Bob'
```

A CASE STUDY − STORING HISTORY INFORMATION

The OAP wishes to computerize the employment history. To do so, they append two columns, one indicating when the information in the row became valid, the other indicating when the information was no longer valid.

Employee (Name, Salary, Title, DateofBirth, Start DATE, Stop DATE)

To the data model, these new columns are identical to DateofBirth. However, their presence has wide−ranging consequences.

A CASE STUDY − PROJECTION

To find an employee's current salary, things are more difficult.

```
SELECT Salary
FROM Employee
WHERE Name = 'Bob' AND Start <= CURRENT_DATE AND CURRENT_DATE <= Stop
```

This query is more complicated than the previous one. The culprit is obviously the two new columns. The OAP wants to distribute to each employee their salary history. Specifically, for each person, the maximal intervals at each salary needs to be determined. Unfortunately, this is not possible in SQL. An employee could have arbitrarily many title changes between salary changes.

```
Name    Salary  Title             DateofBirth   Start           Stop
----    ------  -----             -----------   -----           ----
Bob     60000   Assistant Provost 1945-04-09    1993-01-01      1993-05-30
Bob     70000   Assistant Provost 1945-04-09    1993-06-01      1993-09-30
Bob     70000   Provost           1945-04-09    1993-10-01      1994-01-31
Bob     70000   Professor         1945-04-09    1994-02-01      1994-12-31

                               Figure 1
```

Note that there are three rows in which Bob's salary remained constant at $70,000. Hence, the result should be two rows for Bob.

```
Name    Salary  Start           Stop
----    ------  -----           ----
Bob     60000   1993-01-01      1993-05-30
Bob     70000   1993-06-01      1994-12-31
```

One alternative is to give the user a printout of Salary and Title information, and have user determine when his/her salary changed. This alternative is not very appealing or realistic. A second alternative is to use SQL as much as possible.

```
CREATE TABLE Temp(Salary, Start, Stop)
AS      SELECT Salary, Start, Stop
        FROM Employee;
```

repeat

```
UPDATE Temp T1
SET (T1.Stop) = (SELECT MAX(T2.Stop)
                 FROM Temp AS T2
                 WHERE T1.Salary = T2.Salary AND T1.Start < T2.Start
                        AND T1.Stop >= T2.Start AND T1.Stop < T2.Stop)
WHERE EXISTS (SELECT *
              FROM Temp AS T2
              WHERE T1.Salary = T2.Salary AND T1.Start < T2.Start
                        AND T1.Stop >= T2.Start AND T1.Stop < T2.Stop)
until no rows updated;
```

```
DELETE FROM Temp T1

WHERE EXISTS (SELECT *
              FROM Temp AS T2
              WHERE T1.Salary = T2.Salary
                        AND ((T1.Start > T2.Start AND T1.Stop <= T2.Stop)
                        OR (T1.Start >= T2.Start AND T1.Stop < T2.Stop))
```

The loop finds those intervals that overlap or are adjacent and thus should be merged. The loop is executed log N times in the worst case, where N is the number of rows in a chain of overlapping or adjacent value−equivalent rows. The reader can simulate the query on the example table to convince him/herself of its correctness.

A third alternative is to use SQL only to open a cursor on the table. A linked list of periods is maintained, each with a salary. This linked list should be initialized to empty.

```
DECLARE emp_cursor CURSOR FOR
        SELECT Salary, Title, Start, Stop
        FROM Employee;
OPEN emp_cursor;
loop:
        FETCH emp_cursor INTO :salary, :start, :stop;
        if no-data returned then goto finished;
        find position in linked list to insert this information;
        goto loop;
finished:
CLOSE emp_cursor;
```

iterate through linked list, printing out dates and salaries

The linked list may not be necessary in this case if the cursor is ORDER BY Start.

In any case, the query, a natural one, is quite difficult to express using the facilities present in SQL−92. The query is trivial in TSQL2.

```
        SELECT Salary
        FROM Employee
```

A CASE STUDY − JOIN

A more drastic approach is to avoid the problem of extracting the salary history by reorganizing the schema to separate salary, title, and date of birth information (in the following, we ignore the date of birth, for simplicity).

```
        Employee1 (Name, Salary, Start DATE, Stop DATE)
        Employee2 (Name, Title, Start DATE, Stop DATE)
```

The Employee1 table is as follows.

```
Name    Salary  Start           Stop
----    ------  -----           ----
Bob     60000   1993-01-01      1993-05-30
Bob     70000   1993-06-01      1993-12-31
```

Here is the example Employee2 table.

```
Name    Title                   Start           Stop
----    ------                  -----           ----
Bob     Assistant Provost       1993-01-01      1993-09-30
Bob     Provost                 1993-10-01      1994-01-31
Bob     Professor               1994-02-01      1994-12-31
```

With this change, getting the salary information for an employee is now easy.

```
SELECT Salary, Start, Stop
FROM Employee1
WHERE Name = 'Bob'
```

But what if the OAP wants a table of salary, title intervals (that is, suppose the OAP wishes a table to be computed in the form of Figure 1)? One alternative is to print out two tables, and let the user figure out the combinations. A second alternative is to use SQL entirely. Unfortunately, this query must do a case analysis of how each row of Employee1 overlaps each row of Employee2; there are four possible cases.

```
SELECT Employee1.Name, Salary, Dept, Employee1.Start, Employee1.Stop
FROM Employee1, Employee2
WHERE Employee1.Name = Employee2.Name
     AND Employee2.Start <= Employee1.Start AND Employee1.Stop < Employee2.Stop
UNION
SELECT Employee1.Name, Salary, Dept, Employee1.Start, Employee2.Stop
FROM Employee1, Employee2
WHERE Employee1.Name = Employee2.Name
     AND Employee1.Start >= Employee2.Start AND Employee2.Stop < Employee1.Stop
        AND Employee1.Start < Employee2.Stop
UNION
SELECT Employee1.Name, Salary, Dept, Employee2.Start, Employee1.Stop
FROM Employee1, Employee2
WHERE Employee1.Name = Employee2.Name
     AND Employee2.Start > Employee1.Start AND Employee1.Stop < Employee2.Stop
        AND Employee2.Start < Employee1.Stop
UNION
SELECT Employee1.Name, Salary, Dept, Employee2.Start, Employee2.Stop
FROM Employee1, Employee2
WHERE Employee1.Name = Employee2.Name
     AND Employee2.Start > Employee1.Start AND Employee2.Stop < Employee1.Stop
```

Getting all the cases right is a challenging task. In TSQL2, performing a temporal join is just what one would expect.

```
SELECT Employee1.Name, Salary, Dept
FROM Employee1, Employee2
WHERE Employee1.Name = Employee2.Name
```

A CASE STUDY − AGGREGATES

Now the OAP is asked, what is the maximum salary? Before adding time, this was easy.

```
SELECT MAX(Salary)
FROM Employee
```

Now that the salary history is stored, we'd like a history of the maximum salary over time. The problem, of course, is that SQL does not provide temporal aggregates. The easy way to do this is to print out the information, and scan manually for the maximums. An alternative is to be tricky and convert the snapshot aggregate query into a non−aggregate query, then convert that into a temporal query. The non−aggregate query finds those salaries for which a greater salary does not exist.

```
        SELECT Salary
        FROM Employee AS E1
        WHERE NOT EXISTS (SELECT *
                          FROM Employee AS E2
                          WHERE E2.Salary > E1.Salary)
```

Converting this query into a temporal query is far from obvious. The following is one approach.

```
CREATE TABLE Temp (Salary, Start, Stop)
AS      SELECT Salary, Start, Stop
        FROM Employee;
INSERT INTO Temp
        SELECT T.Salary, T.Start, E.Start
        FROM Temp AS T, Employee AS E
        WHERE E.Start >= T.Start AND E.Start < T.Stop AND E.Salary > T.Salary;

INSERT INTO Temp
        SELECT T.Salary, T.Stop, E.Stop
        FROM Temp AS T, Employee AS E
        WHERE E.Stop > T.Start AND E.Stop <= T.Stop AND E.Salary > T.Salary;
DELETE FROM Temp T
WHERE EXISTS (SELECT *
              FROM Employee AS E
              WHERE ((T.Start => E.Start AND T.Start < E.Stop)
                        OR (E.Start >= T.Start AND E.Start < T.Stop))
                    AND E.Salary > T.Salary;
```

This approach creates an auxiliary table. We add to this table the lower period of a period subtraction and the upper period of a period subtraction. We then delete all periods that overlap with some row defined by the subquery, thereby effecting the NOT EXISTS. Finally we generate from the auxiliary table maximal periods, in the same way that the salary information was computed above. As one might imagine, such SQL code is extremely inefficient to execute, given the complex nested queries with inequality predicates.

A third alternative is to use SQL as little as possible, and instead compute the desired maximum history in a host language using cursors.

The query in TSQL2 is again straightforward and intuitive.

```
        SELECT MAX(Salary)
        FROM Employee
```

SUMMARY

Time−varying data is manipulated in most database applications. Valid−time support is absent in SQL. Many common temporal queries are either difficult to simulate in SQL, or require embedding SQL in a procedural language, due to SQL's lack of support for valid−time tables in its data model and query constructs.

Elsewhere, we showed that adding valid−time support requires few changes to the DBMS implementation, can dramatically simplify some queries and enable others, and can later enable optimizations in storage structures, indexing methods, and optimization strategies that can yield significant performance improvements.

With a new part of SQL3 supporting time−varying information, we can begin to address such applications, enabling SQL3 to better manage temporal data.

```
----------------------------------------------------------------------------
           Accredited Standards Committee* X3, Information Technology
NEWS RELEASE

Doc. No.:       PR/96-0002

Reply to:       Barbara Bennett at bbennett@itic.nw.dc.us

             X3 Announces the Approval of a New Project, ISO/IEC

                         9075 Part 7:  SQL/Temporal

Washington D.C., January 1996
----------------------------------------------------------------------------
```

−− Accredited Standards Committee X3, Information Technology is announcing the approval of a new project on SQL/Temporal Support, ISO/IEC 9075 Part 7, with the work being done in Technical Committee X3H2, Database. The scope of this proposed standard specifies a new Part of the emerging SQL3 standard, e.g., Part 7, Temporal SQL, to be extensions to the SQL language supporting storage, retrieval, and manipulation of temporal data in an SQL database environment. The next X3H2 meeting is scheduled for March 11−14, 1996 in Kansas.

Inquiries regarding this project should be sent to the

```
Chairman of X3H2,
Dr. Donald R. Deutsch,
Sybase, Inc., Suite 800,
6550 Rock Spring
Drive, Bethesda, MD  20817.
Email: deutsch@sybase.com.
```

An initial call for possible patents and other pertinent issues (copyrights, trademarks) is now being issued. Please submit information on these issues to the

```
X3 Secretariat at
1250 Eye Street
NW, Suite 200,
Washington DC  20005.
Email: x3sec@itic.nw.dc.us
FAX:  (202)638-4922.
```

29.17 Part 8 − ISO/ANSI SQL MULTIMEDIA (SQL/MM)

A new ISO/IEC international standardization project for development of an SQL class library for multimedia applications was approved in early 1993. This new standardization activity, named SQL Multimedia (SQL/MM), will specify packages of SQL abstract data type (ADT) definitions using the facilities for ADT specification and invocation provided in the emerging SQL3 specification. SQL/MM intends to standardize class libraries for science and engineering, full−text and document processing, and methods for the management of multimedia objects such as image, sound, animation, music, and video. It will likely provide an SQL language binding for multimedia objects defined by other JTC1 standardization bodies (e.g. SC18 for documents, SC24 for images, and SC29 for photographs and motion pictures).

The Project Plan for SQL/MM indicates that it will be a multi−part standard consisting of an evolving number of parts. Part 1 will be a Framework that specifies how the other parts are to be constructed. Each of the other parts will be devoted to a specific SQL application package. The following SQL/MM Part structure exists as of August 1994:

- Part 1: Framework A non−technical description of how the document is structured.
- Part 2: Full Text Methods and ADTs for text data processing. About 45 pages.
- Part 3: Spatial Methods and ADTs for spatial data management. About 200 pages with active contributions from Spatial Data experts from 3 national bodies.
- Part 4: General Purpose Methods and ADTs for complex numbers, Facilities include trig and exponential functions, vectors, sets, etc. Currently about 90 pages.

There are a number of standards efforts in the area of Spatial and Geographic information:

- ANSI X3L1 − Geographic Information Systems. Mark Ashworth of Unisys is the liason between X3L1 and ANSI X3H2. He is also the editor for parts 1, 3, and 4 of the SQL/MM draft.
- ISO TC 211 − Geographic information/Geomatics

---

30. Technical support for PostgreSQL

This is the order of problem solving:

- Your question can be answered by online manuals
- Enter a keyword in the search box
- Post your question in the mailing list

If you have any technical question or encounter any problem you can e−mail to:

- pgsql−questions@postgresql.org
- Newsgroup comp.databases.postgresql.general
- Newsgroup comp.databases.postgresql.hackers
- Newsgroup comp.databases.postgresql.doc
- Newsgroup comp.databases.postgresql.bugs
- Newsgroup linux.postgres
- Other Mailing lists http://www.postgresql.org

and expect e−mail answer in less than a day. As the user−base of internet product is very vast, and users support other users, internet will be capable of giving technical support to billions of users easily. Email support is much more convenient than telephone support as you can cut and paste error messages, program output etc.. and easily transmit to mailing list/newsgroup.

30.1 Commercial Support

PostgreSQL organisation is selling technical support to companies, the revenue generated will be used for maintaining several mirror sites (web and ftp) around the world. The revenue will also be used to produce printed documentation, guides, textbooks which will help the customers. They are at http://www.postgresql.org

Another company called 'Great Bridge Corporation' is doing development, sales and support of PostgreSQL. They are at http://www.greatbridge.com. It is a public company setup by 'Landmark Communications corp' and other venture capital firms to exclusively sell and support PostgreSQL to very large enterprises and corporations all over the world.

You can also take help from professional consulting firms like RedHat, Anderson, WGS (Work Group Solutions). Contact them for help, since they have very good expertise in "C", "C++" (PostgreSQL is written in "C") −

- Redhat Corp − Database consulting division http://www.redhat.com
- Work Group Solutions http://www.wgs.com
- Anderson Consulting http://www.ac.com

---

31. Economic and Business Aspects

Commercial databases pay many taxes like federal, state, sales, employment, social security, medicare taxes, health care for employees, bunch of benefits for employees, marketing and advertisement costs. All these costs do not go directly for the development of the database and do not improve the quality or technology of the database. When you buy a commercial database, some portion of the amount goes for overheads like taxes, expenses and balance for database R&D costs.

Also commercial databases have to pay for buildings/real−estates and purchase Unix machines, install and maintain them. All of these costs are passed onto customers.

PostgreSQL has the advantage over commercial databases as there is no direct taxes since it is made on the internet. A very vast group of people contribute to the development of the PostgreSQL. For example, in a hypothetical case, if there are one million companies in U.S.A and each contribute about $ 10 (worth of software to PostgreSQL) then each and every company will get ten million dollars!! This is the **GREAT MAGIC** of software development on internet.

Currently, PostgreSQL source code is about 2,50,000 lines of "C", "C++" code. If cost of each line of "C" code is $ 2 then the PostgreSQL is worth about $ 5,00,000 (half a million dollars!).

Many companies already develop in−house vast amount of "C", "C++" code. Hence by taking in the source code of PostgreSQL and collaborating with other companies on internet will greatly benefit the company saving time and efforts.

---

32. List of Other Databases

Listed below are other SQL databases for Unix, Linux.

- Click and go to Applications−>databases. http://www.caldera.com/tech−ref/linuxapps/linapps.html
- Click and go to Applications−>databases. http://www.xnet.com/~blatura/linapps.shtml
- Database resources http://linas.org/linux/db.html This was written by Linas Vepstas: linas@fc.net
- Free Database List http://cuiwww.unige.ch:80/~scg/FreeDB/FreeDB.list.html
- Browne's RDBMS List http://www.hex.net/~cbbrowne/rdbms.html written by Christopher B. Browne cbbrowne@hex.net
- SAL's List of Relational DBMS http://SAL.KachinaTech.COM/H/1/
- SAL's List of Object−Oriented DBMS http://SAL.KachinaTech.COM/H/2/
- SAL's List of Utilites and Other Databases http://SAL.KachinaTech.COM/H/3/
- ACM SIGMOD Index of Publicly Available Database Software http://bunny.cs.uiuc.edu/sigmod/databaseSoftware/

33. Internet World Wide Web Searching Tips

Internet is very vast and it has vast number of software and has an ocean of information underneath. It is growing at the rate of 300% annually world wide. It is estimated that there are about 10 million Web sites world wide!

To search for an information you would use search engines like "Yahoo", "Netscape", "Lycos" etc. Go to Yahoo, click on search. Use filtering options to narrow down your search criteria. The default search action is "Intelligent search" which is more general and lists all possiblities. Click on "Options" to select "EXACT phrase" search, "AND" search, "OR" search, etc.. This way you would find the information you need much faster. Also in the search menu, there are radio−buttons for searching in Usenet, Web−sites and Yahoo sites.

34. Conclusion

After researching all the available databases which are **free** and source code is available, it was found that ONLY PostgreSQL is the MOST mature, most widely used and robust RDBMS SQL free database (object relational) in the world.

PostgreSQL is very appealing since lot of work had already been done. It has ODBC and JDBC drivers, using these it is possible to write applications independent of the databases. The applications written in PostgreSQL using ODBC, JDBC drivers are easily portable to other databases like Oracle, Sybase and Informix and vice versa.

You may ask "But why PostgreSQL ?" The answer is, since it takes lot more time to develop a database system from scratch, it makes sense to pick up a database system which satisfies the following conditions −

A database system

- Whose source code is available − Must be a 'Open Source Code' system
- Has no license strings, no ownership strings attached to it
- Which can be distributed on internet
- Which had been on development for several years.
- Which satisfies standards like ISO/ANSI SQL 92 (and SQL 89)
- Which can satisfy future needs like SQL 3 (SQL 98)
- Which has advanced capabilities

And it just happens to be 'PostgreSQL' which satisfies all these conditions and is an appropriate software for this situation. You may say 'PostgreSQL' is a very strange name (It is pronounced as Post−gres−cue−el and not Postgre−es−cue−el. It's a very unusual name and it is very hard to pronounce). But my argument is − why change the name. This world is stuck with "PostgreSQL" forever and people all over the world love this name!!

35. FAQ − Questions on PostgreSQL

Please refer to the latest version of FAQ for General, Linux and Irix at

- http://www.postgresql.org/docs/faq−english.shtml

36. Other Formats of this Document

This document is published in 11 different formats namely − DVI, Postscript, Latex, Adobe Acrobat PDF, LyX, GNU−info, HTML, RTF(Rich Text Format), Plain−text, Unix man pages and SGML.

- You can get this HOWTO document as a single file tar ball in HTML, DVI, Postscript or SGML formats from − ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/other−formats/ and http://www.linuxdoc.org/docs.html#howto
- Plain text format is in: ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO and http://www.linuxdoc.org/docs.html#howto
- Single HTML file format is in: http://www.linuxdoc.org/docs.html#howto
- Translations to other languages like French, German, Spanish, Chinese, Japanese are in ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO and http://www.linuxdoc.org/docs.html#howto Any help from you to translate to other languages is welcome.

The document is written using a tool called "SGML−Tools" which can be got from − http://www.sgmltools.org Compiling the source you will get the following commands like

- sgml2html databasehowto.sgml (to generate html file)
- sgml2rtf databasehowto.sgml (to generate RTF file)
- sgml2latex databasehowto.sgml (to generate latex file)

LaTeX documents may be converted into PDF files simply by producing a Postscript output using **sgml2latex** (and dvips) and running the output through the Acrobat **distill** (http://www.adobe.com) command as follows:

```
bash$ man sgml2latex
bash$ sgml2latex filename.sgml
bash$ man dvips
bash$ dvips -o filename.ps filename.dvi
bash$ distill filename.ps
bash$ man ghostscript
bash$ man ps2pdf
bash$ ps2pdf input.ps output.pdf
bash$ acroread output.pdf &
```

Or you can use Ghostscript command **ps2pdf**. ps2pdf is a work−alike for nearly all the functionality of Adobe's Acrobat Distiller product: it converts PostScript files to Portable Document Format (PDF) files. **ps2pdf** is implemented as a very small command script (batch file) that invokes Ghostscript, selecting a special "output device" called **pdfwrite**. In order to use ps2pdf, the pdfwrite device must be included in the makefile when Ghostscript was compiled; see the documentation on building Ghostscript for details.

This howto document is located at −

- http://sunsite.unc.edu/LDP/HOWTO/PostgreSQL−HOWTO.html

Also you can find this document at the following mirrors sites −

- http://www.caldera.com/LDP/HOWTO/PostgreSQL−HOWTO.html
- http://www.WGS.com/LDP/HOWTO/PostgreSQL−HOWTO.html
- http://www.cc.gatech.edu/linux/LDP/HOWTO/PostgreSQL−HOWTO.html
- http://www.redhat.com/linux−info/ldp/HOWTO/PostgreSQL−HOWTO.html
- Other mirror sites near you (network−address−wise) can be found at http://sunsite.unc.edu/LDP/hmirrors.html select a site and go to directory /LDP/HOWTO/PostgreSQL−HOWTO.html

In order to view the document in dvi format, use the xdvi program. The xdvi program is located in tetex−xdvi*.rpm package in Redhat Linux which can be located through ControlPanel | Applications | Publishing | TeX menu buttons. To read dvi document give the command −

```
xdvi -geometry 80x90 howto.dvi
man xdvi
```

And resize the window with mouse. To navigate use Arrow keys, Page Up, Page Down keys, also you can use 'f', 'd', 'u', 'c', 'l', 'r', 'p', 'n' letter keys to move up, down, center, next page, previous page etc. To turn off expert menu press 'x'.

You can read postscript file using the program 'gv' (ghostview) or 'ghostscript'. The ghostscript program is in ghostscript*.rpm package and gv program is in gv*.rpm package in Redhat Linux which can be located through ControlPanel | Applications | Graphics menu buttons. The gv program is much more user friendly than ghostscript. Also ghostscript and gv are available on other platforms like OS/2, Windows 95 and NT, you view this document even on those platforms.

- Get ghostscript for Windows 95, OS/2, and for all OSes from http://www.cs.wisc.edu/~ghost

To read postscript document give the command −

```
gv howto.ps
ghostscript howto.ps
```

CAUTION: This document is large, total number of pages (postscript) if printed will be approximately 113 pages.

You can read HTML format document using Netscape Navigator, Microsoft Internet explorer, Redhat Baron Web browser or any of the 10 other web browsers.

You can read the latex, LyX output using LyX a X−Windows front end to latex.

---

37. Copyright and License

---

38. Appendix A − Syntax of ANSI/ISO SQL 1992

```
This file contains a depth-first tree traversal of the BNF
for the  language done at about 27-AUG-1992 11:03:41.64.
The specific version of the BNF included here is:  ANSI-only, SQL2-only.


<SQL terminal character> ::=
      <SQL language character>
    | <SQL embedded language character>

<SQL language character> ::=
      <simple Latin letter>
    | <digit>
    | <SQL special character>

<simple Latin letter> ::=
      <simple Latin upper case letter>
    | <simple Latin lower case letter>

<simple Latin upper case letter> ::=
          A | B | C | D | E | F | G | H | I | J | K | L | M | N | O
    | P | Q | R | S | T | U | V | W | X | Y | Z

<simple Latin lower case letter> ::=
          a | b | c | d | e | f | g | h | i | j | k | l | m | n | o
    | p | q | r | s | t | u | v | w | x | y | z

<digit> ::=
    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<SQL special character> ::=
      <space>
    | <double quote>
    | <percent>
    | <ampersand>
    | <quote>
    | <left paren>
    | <right paren>
    | <asterisk>
    | <plus sign>
    | <comma>
    | <minus sign>
    | <period>
    | <solidus>
    | <colon>
```

```
    | <semicolon>
    | <less than operator>
    | <equals operator>
    | <greater than operator>
    | <question mark>
    | <underscore>
    | <vertical bar>

<space> ::= !! <EMPHASIS>(space character in character set in use)

<double quote> ::= "

<percent> ::= %

<ampersand> ::= &

<quote> ::= '

<left paren> ::= (

<right paren> ::= )

<asterisk> ::= *

<plus sign> ::= +

<comma> ::= ,

<minus sign> ::= -

<period> ::= .

<solidus> ::= /

<colon> ::= :

<semicolon> ::= ;

<less than operator> ::= <

<equals operator> ::= =

<greater than operator> ::= >

<question mark> ::= ?

<underscore> ::= _

<vertical bar> ::= |

<SQL embedded language character> ::=
      <left bracket>
    | <right bracket>

<left bracket> ::= [

<right bracket> ::= ]

<token> ::=
      <nondelimiter token>
    | <delimiter token>
```

```
<nondelimiter token> ::=
      <regular identifier>
    | <key word>
    | <unsigned numeric literal>
    | <national character string literal>
    | <bit string literal>
    | <hex string literal>

<regular identifier> ::= <identifier body>

<identifier body> ::=
    <identifier start> [ ( <underscore> | <identifier part> )... ]

<identifier start> ::= <EMPHASIS>(!! See the Syntax Rules)

<identifier part> ::=
      <identifier start>
    | <digit>

<key word> ::=
      <reserved word>
    | <non-reserved word>

<reserved word> ::=
      ABSOLUTE | ACTION | ADD | ALL
    | ALLOCATE | ALTER | AND
    | ANY | ARE
    | AS | ASC
    | ASSERTION | AT
    | AUTHORIZATION | AVG
    | BEGIN | BETWEEN | BIT | BIT_LENGTH
    | BOTH | BY
    | CASCADE | CASCADED | CASE | CAST
    | CATALOG
    | CHAR | CHARACTER | CHAR_LENGTH
    | CHARACTER_LENGTH | CHECK | CLOSE | COALESCE
    | COLLATE | COLLATION
    | COLUMN | COMMIT
    | CONNECT
    | CONNECTION | CONSTRAINT
    | CONSTRAINTS | CONTINUE
    | CONVERT | CORRESPONDING | COUNT | CREATE | CROSS
    | CURRENT
    | CURRENT_DATE | CURRENT_TIME
    | CURRENT_TIMESTAMP | CURRENT_USER | CURSOR
    | DATE | DAY | DEALLOCATE | DEC
    | DECIMAL | DECLARE | DEFAULT | DEFERRABLE
    | DEFERRED | DELETE | DESC | DESCRIBE | DESCRIPTOR
    | DIAGNOSTICS
    | DISCONNECT | DISTINCT | DOMAIN | DOUBLE | DROP
    | ELSE | END | END-EXEC | ESCAPE
    | EXCEPT | EXCEPTION
    | EXEC | EXECUTE | EXISTS
    | EXTERNAL | EXTRACT
    | FALSE | FETCH | FIRST | FLOAT | FOR
    | FOREIGN | FOUND | FROM | FULL
    | GET | GLOBAL | GO | GOTO
    | GRANT | GROUP
    | HAVING | HOUR
    | IDENTITY | IMMEDIATE | IN | INDICATOR
    | INITIALLY | INNER | INPUT
    | INSENSITIVE | INSERT | INT | INTEGER | INTERSECT
```

```
     | INTERVAL | INTO | IS
     | ISOLATION
     | JOIN
     | KEY
     | LANGUAGE | LAST | LEADING | LEFT
     | LEVEL | LIKE | LOCAL | LOWER
     | MATCH | MAX | MIN | MINUTE | MODULE
     | MONTH
     | NAMES | NATIONAL | NATURAL | NCHAR | NEXT | NO
     | NOT | NULL
     | NULLIF | NUMERIC
     | OCTET_LENGTH | OF
     | ON | ONLY | OPEN | OPTION | OR
     | ORDER | OUTER
     | OUTPUT | OVERLAPS
     | PAD | PARTIAL | POSITION | PRECISION | PREPARE
     | PRESERVE | PRIMARY
     | PRIOR | PRIVILEGES | PROCEDURE | PUBLIC
     | READ | REAL | REFERENCES | RELATIVE | RESTRICT
     | REVOKE | RIGHT
     | ROLLBACK | ROWS
     | SCHEMA | SCROLL | SECOND | SECTION
     | SELECT
     | SESSION | SESSION_USER | SET
     | SIZE | SMALLINT | SOME | SPACE | SQL | SQLCODE
     | SQLERROR | SQLSTATE
     | SUBSTRING | SUM | SYSTEM_USER
     | TABLE | TEMPORARY
     | THEN | TIME | TIMESTAMP
     | TIMEZONE_HOUR | TIMEZONE_MINUTE
     | TO | TRAILING | TRANSACTION
     | TRANSLATE | TRANSLATION | TRIM | TRUE
     | UNION | UNIQUE | UNKNOWN | UPDATE | UPPER | USAGE
     | USER | USING
     | VALUE | VALUES | VARCHAR | VARYING | VIEW
     | WHEN | WHENEVER | WHERE | WITH | WORK | WRITE
     | YEAR
     | ZONE

<non-reserved word> ::=

       ADA
     | C | CATALOG_NAME
     | CHARACTER_SET_CATALOG | CHARACTER_SET_NAME
     | CHARACTER_SET_SCHEMA | CLASS_ORIGIN | COBOL | COLLATION_CATALOG
     | COLLATION_NAME | COLLATION_SCHEMA | COLUMN_NAME | COMMAND_FUNCTION
     | COMMITTED
     | CONDITION_NUMBER | CONNECTION_NAME | CONSTRAINT_CATALOG | CONSTRAINT_NAME
     | CONSTRAINT_SCHEMA | CURSOR_NAME
     | DATA | DATETIME_INTERVAL_CODE
     | DATETIME_INTERVAL_PRECISION | DYNAMIC_FUNCTION
     | FORTRAN
     | LENGTH
     | MESSAGE_LENGTH | MESSAGE_OCTET_LENGTH | MESSAGE_TEXT | MORE | MUMPS
     | NAME | NULLABLE | NUMBER
     | PASCAL | PLI
     | REPEATABLE | RETURNED_LENGTH | RETURNED_OCTET_LENGTH | RETURNED_SQLSTATE
     | ROW_COUNT
     | SCALE | SCHEMA_NAME | SERIALIZABLE | SERVER_NAME | SUBCLASS_ORIGIN
     | TABLE_NAME | TYPE
     | UNCOMMITTED | UNNAMED
```

```
<unsigned numeric literal> ::=
      <exact numeric literal>
    | <approximate numeric literal>

<exact numeric literal> ::=
      <unsigned integer> [ <period> [ <unsigned integer> ] ]
    | <period> <unsigned integer>

<unsigned integer> ::= <digit>...

<approximate numeric literal> ::= <mantissa> E <exponent>

<mantissa> ::= <exact numeric literal>

<exponent> ::= <signed integer>

<signed integer> ::= [ <sign> ] <unsigned integer>

<sign> ::= <plus sign> | <minus sign>

<national character string literal> ::=
    N <quote> [ <character representation>... ] <quote>
      [ ( <separator>... <quote> [ <character representation>... ] <quote> )... ]

<character representation> ::=
      <nonquote character>
    | <quote symbol>

<nonquote character> ::= !! <EMPHASIS>(See the Syntax Rules.)

<quote symbol> ::= <quote><quote>

<separator> ::= ( <comment> | <space> | <newline> )...

<comment> ::=
    <comment introducer> [ <comment character>... ] <newline>

<comment introducer> ::= <minus sign><minus sign>[<minus sign>...]

<comment character> ::=
      <nonquote character>
    | <quote>

<newline> ::= !! <EMPHASIS>(implementation-defined end-of-line indicator)

<bit string literal> ::=
    B <quote> [ <bit>... ] <quote>
      [ ( <separator>... <quote> [ <bit>... ] <quote> )... ]

<bit> ::= 0 | 1

<hex string literal> ::=
    X <quote> [ <hexit>... ] <quote>
      [ ( <separator>... <quote> [ <hexit>... ] <quote> )... ]

<hexit> ::= <digit> | A | B | C | D | E | F | a | b | c | d | e | f

<delimiter token> ::=
      <character string literal>
    | <date string>
    | <time string>
    | <timestamp string>
```

```
    | <interval string>
    | <delimited identifier>
    | <SQL special character>
    | <not equals operator>
    | <greater than or equals operator>
    | <less than or equals operator>
    | <concatenation operator>
    | <double period>
    | <left bracket>
    | <right bracket>

<character string literal> ::=
    [ <introducer><character set specification> ]
    <quote> [ <character representation>... ] <quote>
      [ ( <separator>... <quote> [ <character representation>... ] <quote> )... ]

<introducer> ::= <underscore>

<character set specification> ::=
      <standard character repertoire name>
    | <implementation-defined character repertoire name>
    | <user-defined character repertoire name>
    | <standard universal character form-of-use name>
    | <implementation-defined universal character form-of-use name>

<standard character repertoire name> ::= <character set name>

<character set name> ::= [ <schema name> <period> ]
      <SQL language identifier>

<schema name> ::=
    [ <catalog name> <period> ] <unqualified schema name>

<catalog name> ::= <identifier>

<identifier> ::=
    [ <introducer><character set specification> ] <actual identifier>

<actual identifier> ::=
      <regular identifier>
    | <delimited identifier>

<delimited identifier> ::=
    <double quote> <delimited identifier body> <double quote>

<delimited identifier body> ::= <delimited identifier part>...

<delimited identifier part> ::=
      <nondoublequote character>
    | <doublequote symbol>

<nondoublequote character> ::= <EMPHASIS>(!! See the Syntax Rules)

<doublequote symbol> ::= <double quote><double quote>

<unqualified schema name> ::= <identifier>

<SQL language identifier> ::=
    <SQL language identifier start>
       [ ( <underscore> | <SQL language identifier part> )... ]

<SQL language identifier start> ::= <simple Latin letter>
```

```
<SQL language identifier part> ::=
      <simple Latin letter>
    | <digit>

<implementation-defined character repertoire name> ::=
    <character set name>

<user-defined character repertoire name> ::= <character set name>

<standard universal character form-of-use name> ::=
    <character set name>

<implementation-defined universal character form-of-use name> ::=
    <character set name>

<date string> ::=
    <quote> <date value> <quote>

<date value> ::=
    <years value> <minus sign> <months value>
        <minus sign> <days value>

<years value> ::= <datetime value>

<datetime value> ::= <unsigned integer>

<months value> ::= <datetime value>

<days value> ::= <datetime value>

<time string> ::=
    <quote> <time value> [ <time zone interval> ] <quote>

<time value> ::=
    <hours value> <colon> <minutes value> <colon> <seconds value>

<hours value> ::= <datetime value>

<minutes value> ::= <datetime value>

<seconds value> ::=
      <seconds integer value> [ <period> [ <seconds fraction> ] ]

<seconds integer value> ::= <unsigned integer>

<seconds fraction> ::= <unsigned integer>

<time zone interval> ::=
    <sign> <hours value> <colon> <minutes value>

<timestamp string> ::=
    <quote> <date value> <space> <time value>
        [ <time zone interval> ] <quote>

<interval string> ::=
    <quote> ( <year-month literal> | <day-time literal> ) <quote>

<year-month literal> ::=
      <years value>
    | [ <years value> <minus sign> ] <months value>
```

```
<day-time literal> ::=
      <day-time interval>
    | <time interval>

<day-time interval> ::=
    <days value>
      [ <space> <hours value> [ <colon> <minutes value>
        [ <colon> <seconds value> ] ] ]

<time interval> ::=
      <hours value> [ <colon> <minutes value> [ <colon> <seconds value> ] ]
    | <minutes value> [ <colon> <seconds value> ]
    | <seconds value>

<not equals operator> ::= <>

<greater than or equals operator> ::= >=

<less than or equals operator> ::= <=

<concatenation operator> ::= ||

<double period> ::= ..

<module> ::=
    <module name clause>
    <language clause>
    <module authorization clause>
    [ <temporary table declaration>... ]
    <module contents>...

<module name clause> ::=
    MODULE [ <module name> ]
      [ <module character set specification> ]

<module name> ::= <identifier>

<module character set specification> ::=
    NAMES ARE <character set specification>

<language clause> ::=
    LANGUAGE <language name>

<language name> ::=
    ADA | C | COBOL | FORTRAN | MUMPS | PASCAL | PLI

<module authorization clause> ::=
      SCHEMA <schema name>
    | AUTHORIZATION <module authorization identifier>
    | SCHEMA <schema name>
          AUTHORIZATION <module authorization identifier>

<module authorization identifier> ::=
    <authorization identifier>

<authorization identifier> ::= <identifier>

<temporary table declaration> ::=
    DECLARE LOCAL TEMPORARY TABLE
        <qualified local table name>
      <table element list>
      [ ON COMMIT ( PRESERVE | DELETE ) ROWS ]
```

```
<qualified local table name> ::=
    MODULE <period> <local table name>

<local table name> ::= <qualified identifier>

<qualified identifier> ::= <identifier>

<table element list> ::=
      <left paren> <table element> [ ( <comma> <table element> )... ] <right paren>

<table element> ::=
      <column definition>
    | <table constraint definition>

<column definition> ::=
    <column name> ( <data type> | <domain name> )
    [ <default clause> ]
    [ <column constraint definition>... ]
    [ <collate clause> ]

<column name> ::= <identifier>

<data type> ::=
      <character string type>
           [ CHARACTER SET <character set specification> ]
    | <national character string type>
    | <bit string type>
    | <numeric type>
    | <datetime type>
    | <interval type>

<character string type> ::=
      CHARACTER [ <left paren> <length> <right paren> ]
    | CHAR [ <left paren> <length> <right paren> ]
    | CHARACTER VARYING <left paren> <length> <right paren>
    | CHAR VARYING <left paren> <length> <right paren>
    | VARCHAR <left paren> <length> <right paren>

<length> ::= <unsigned integer>

<national character string type> ::=
      NATIONAL CHARACTER [ <left paren> <length> <right paren> ]
    | NATIONAL CHAR [ <left paren> <length> <right paren> ]
    | NCHAR [ <left paren> <length> <right paren> ]
    | NATIONAL CHARACTER VARYING <left paren> <length> <right paren>
    | NATIONAL CHAR VARYING <left paren> <length> <right paren>
    | NCHAR VARYING <left paren> <length> <right paren>

<bit string type> ::=
      BIT [ <left paren> <length> <right paren> ]
    | BIT VARYING <left paren> <length> <right paren>

<numeric type> ::=
      <exact numeric type>
    | <approximate numeric type>

<exact numeric type> ::=
      NUMERIC [ <left paren> <precision> [ <comma> <scale> ] <right paren> ]
    | DECIMAL [ <left paren> <precision> [ <comma> <scale> ] <right paren> ]
    | DEC [ <left paren> <precision> [ <comma> <scale> ] <right paren> ]
    | INTEGER
```

```
    | INT
    | SMALLINT

<precision> ::= <unsigned integer>

<scale> ::= <unsigned integer>

<approximate numeric type> ::=
      FLOAT [ <left paren> <precision> <right paren> ]
    | REAL
    | DOUBLE PRECISION

<datetime type> ::=
      DATE
    | TIME [ <left paren> <time precision> <right paren> ]
          [ WITH TIME ZONE ]
    | TIMESTAMP [ <left paren> <timestamp precision> <right paren> ]
          [ WITH TIME ZONE ]

<time precision> ::= <time fractional seconds precision>

<time fractional seconds precision> ::= <unsigned integer>

<timestamp precision> ::= <time fractional seconds precision>

<interval type> ::= INTERVAL <interval qualifier>

<interval qualifier> ::=
      <start field> TO <end field>
    | <single datetime field>

<start field> ::=
    <non-second datetime field>
        [ <left paren> <interval leading field precision> <right paren> ]

<non-second datetime field> ::= YEAR | MONTH | DAY | HOUR
    | MINUTE

<interval leading field precision> ::= <unsigned integer>

<end field> ::=
      <non-second datetime field>
    | SECOND [ <left paren> <interval fractional seconds precision> <right paren> ]

<interval fractional seconds precision> ::= <unsigned integer>

<single datetime field> ::=
      <non-second datetime field>
          [ <left paren> <interval leading field precision> <right paren> ]
    | SECOND [ <left paren> <interval leading field precision>
          [ <comma> <interval fractional seconds precision> ] <right paren> ]

<domain name> ::= <qualified name>

<qualified name> ::=
    [ <schema name> <period> ] <qualified identifier>

<default clause> ::=
      DEFAULT <default option>

<default option> ::=
      <literal>
```

```
      | <datetime value function>
      | USER
      | CURRENT_USER
      | SESSION_USER
      | SYSTEM_USER
      | NULL

<literal> ::=
        <signed numeric literal>
      | <general literal>

<signed numeric literal> ::=
      [ <sign> ] <unsigned numeric literal>

<general literal> ::=
        <character string literal>
      | <national character string literal>
      | <bit string literal>
      | <hex string literal>
      | <datetime literal>
      | <interval literal>

<datetime literal> ::=
        <date literal>
      | <time literal>
      | <timestamp literal>

<date literal> ::=
      DATE <date string>

<time literal> ::=
      TIME <time string>

<timestamp literal> ::=
      TIMESTAMP <timestamp string>

<interval literal> ::=
      INTERVAL [ <sign> ] <interval string> <interval qualifier>

<datetime value function> ::=
        <current date value function>
      | <current time value function>
      | <current timestamp value function>

<current date value function> ::= CURRENT_DATE

<current time value function> ::=
        CURRENT_TIME [ <left paren> <time precision> <right paren> ]

<current timestamp value function> ::=
        CURRENT_TIMESTAMP [ <left paren> <timestamp precision> <right paren> ]

<column constraint definition> ::=
      [ <constraint name definition> ]
      <column constraint>
        [ <constraint attributes> ]

<constraint name definition> ::= CONSTRAINT <constraint name>

<constraint name> ::= <qualified name>

<column constraint> ::=
```

```
      NOT NULL
    | <unique specification>
    | <references specification>
    | <check constraint definition>

<unique specification> ::=
    UNIQUE | PRIMARY KEY

<references specification> ::=
    REFERENCES <referenced table and columns>
      [ MATCH <match type> ]
      [ <referential triggered action> ]

<referenced table and columns> ::=
     <table name> [ <left paren> <reference column list> <right paren> ]

<table name> ::=
      <qualified name>
    | <qualified local table name>

<reference column list> ::= <column name list>

<column name list> ::=
    <column name> [ ( <comma> <column name> )... ]

<match type> ::=
      FULL
    | PARTIAL

<referential triggered action> ::=
      <update rule> [ <delete rule> ]
    | <delete rule> [ <update rule> ]

<update rule> ::= ON UPDATE <referential action>

<referential action> ::=
      CASCADE
    | SET NULL
    | SET DEFAULT
    | NO ACTION

<delete rule> ::= ON DELETE <referential action>

<check constraint definition> ::=
    CHECK
        <left paren> <search condition> <right paren>

<search condition> ::=
      <boolean term>
    | <search condition> OR <boolean term>

<boolean term> ::=
      <boolean factor>
    | <boolean term> AND <boolean factor>

<boolean factor> ::=
    [ NOT ] <boolean test>

<boolean test> ::=
    <boolean primary> [ IS [ NOT ]
          <truth value> ]
```

```
<boolean primary> ::=
      <predicate>
    | <left paren> <search condition> <right paren>

<predicate> ::=
      <comparison predicate>
    | <between predicate>
    | <in predicate>
    | <like predicate>
    | <null predicate>
    | <quantified comparison predicate>
    | <exists predicate>
    | <unique predicate>
    | <match predicate>
    | <overlaps predicate>

<comparison predicate> ::=
    <row value constructor> <comp op>
        <row value constructor>

<row value constructor> ::=
       <row value constructor element>
    | <left paren> <row value constructor list> <right paren>
    | <row subquery>

<row value constructor element> ::=
      <value expression>
    | <null specification>
    | <default specification>

<value expression> ::=
      <numeric value expression>
    | <string value expression>
    | <datetime value expression>
    | <interval value expression>

<numeric value expression> ::=
      <term>
    | <numeric value expression> <plus sign> <term>
    | <numeric value expression> <minus sign> <term>

<term> ::=
      <factor>
    | <term> <asterisk> <factor>
    | <term> <solidus> <factor>

<factor> ::=
    [ <sign> ] <numeric primary>

<numeric primary> ::=
      <value expression primary>
    | <numeric value function>

<value expression primary> ::=
      <unsigned value specification>
    | <column reference>
    | <set function specification>
    | <scalar subquery>
    | <case expression>
    | <left paren> <value expression> <right paren>
    | <cast specification>
```

```
<unsigned value specification> ::=
      <unsigned literal>
    | <general value specification>

<unsigned literal> ::=
      <unsigned numeric literal>
    | <general literal>

<general value specification> ::=
      <parameter specification>
    | <dynamic parameter specification>
    | <variable specification>
    | USER
    | CURRENT_USER
    | SESSION_USER
    | SYSTEM_USER
    | VALUE

<parameter specification> ::=
    <parameter name> [ <indicator parameter> ]

<parameter name> ::= <colon> <identifier>

<indicator parameter> ::=
    [ INDICATOR ] <parameter name>

<dynamic parameter specification> ::= <question mark>

<variable specification> ::=
    <embedded variable name> [ <indicator variable> ]

<embedded variable name> ::=
    <colon><host identifier>

<host identifier> ::=
      <Ada host identifier>
    | <C host identifier>
    | <COBOL host identifier>
    | <Fortran host identifier>
    | <MUMPS host identifier>
    | <Pascal host identifier>
    | <PL/I host identifier>

<Ada host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<C host identifier> ::=
    !! <EMPHASIS>(See the Syntax Rules.)

<COBOL host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<Fortran host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<MUMPS host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<Pascal host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<PL/I host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<indicator variable> ::=
    [ INDICATOR ] <embedded variable name>

<column reference> ::= [ <qualifier> <period> ] <column name>
```

```
<qualifier> ::=
      <table name>
    | <correlation name>

<correlation name> ::= <identifier>

<set function specification> ::=
      COUNT <left paren> <asterisk> <right paren>
    | <general set function>

<general set function> ::=
      <set function type>
          <left paren> [ <set quantifier> ] <value expression> <right paren>

<set function type> ::=
    AVG | MAX | MIN | SUM | COUNT

<set quantifier> ::= DISTINCT | ALL

<scalar subquery> ::= <subquery>

<subquery> ::= <left paren> <query expression> <right paren>

<query expression> ::=
      <non-join query expression>
    | <joined table>

<non-join query expression> ::=
      <non-join query term>
    | <query expression> UNION  [ ALL ]
          [ <corresponding spec> ] <query term>
    | <query expression> EXCEPT [ ALL ]
          [ <corresponding spec> ] <query term>

<non-join query term> ::=
      <non-join query primary>
    | <query term> INTERSECT [ ALL ]
          [ <corresponding spec> ] <query primary>

<non-join query primary> ::=
      <simple table>
    | <left paren> <non-join query expression> <right paren>

<simple table> ::=
      <query specification>
    | <table value constructor>
    | <explicit table>

<query specification> ::=
    SELECT [ <set quantifier> ] <select list> <table expression>

<select list> ::=
      <asterisk>
    | <select sublist> [ ( <comma> <select sublist> )... ]

<select sublist> ::=
      <derived column>
    | <qualifier> <period> <asterisk>

<derived column> ::= <value expression> [ <as clause> ]
```

```
<as clause> ::= [ AS ] <column name>

<table expression> ::=
    <from clause>
    [ <where clause> ]
    [ <group by clause> ]
    [ <having clause> ]

<from clause> ::= FROM <table reference>
    [ ( <comma> <table reference> )... ]

<table reference> ::=
      <table name> [ [ AS ] <correlation name>
          [ <left paren> <derived column list> <right paren> ] ]
    | <derived table> [ AS ] <correlation name>
          [ <left paren> <derived column list> <right paren> ]
    | <joined table>

<derived column list> ::= <column name list>

<derived table> ::= <table subquery>

<table subquery> ::= <subquery>

<joined table> ::=
      <cross join>
    | <qualified join>
    | <left paren> <joined table> <right paren>

<cross join> ::=
    <table reference> CROSS JOIN <table reference>

<qualified join> ::=
    <table reference> [ NATURAL ] [ <join type> ] JOIN
      <table reference> [ <join specification> ]

<join type> ::=
      INNER
    | <outer join type> [ OUTER ]
    | UNION

<outer join type> ::=
      LEFT
    | RIGHT
    | FULL

<join specification> ::=
      <join condition>
    | <named columns join>

<join condition> ::= ON <search condition>

<named columns join> ::=
    USING <left paren> <join column list> <right paren>

<join column list> ::= <column name list>

<where clause> ::= WHERE <search condition>

<group by clause> ::=
    GROUP BY <grouping column reference list>
```

```
<grouping column reference list> ::=
    <grouping column reference>
        [ ( <comma> <grouping column reference> )... ]

<grouping column reference> ::=
    <column reference> [ <collate clause> ]

<collate clause> ::= COLLATE <collation name>

<collation name> ::= <qualified name>

<having clause> ::= HAVING <search condition>

<table value constructor> ::=
    VALUES <table value constructor list>

<table value constructor list> ::=
    <row value constructor> [ ( <comma> <row value constructor> )... ]

<explicit table> ::= TABLE <table name>

<query term> ::=
      <non-join query term>
    | <joined table>

<corresponding spec> ::=
    CORRESPONDING [ BY <left paren> <corresponding column list> <right paren> ]

<corresponding column list> ::= <column name list>

<query primary> ::=
      <non-join query primary>
    | <joined table>

<case expression> ::=
      <case abbreviation>
    | <case specification>

<case abbreviation> ::=
      NULLIF <left paren> <value expression> <comma>
            <value expression> <right paren>
    | COALESCE <left paren> <value expression>
            ( <comma> <value expression> )... <right paren>

<case specification> ::=
      <simple case>
    | <searched case>

<simple case> ::=
    CASE <case operand>
      <simple when clause>...
      [ <else clause> ]
    END

<case operand> ::= <value expression>

<simple when clause> ::= WHEN <when operand> THEN <result>

<when operand> ::= <value expression>

<result> ::= <result expression> | NULL
```

```
<result expression> ::= <value expression>

<else clause> ::= ELSE <result>

<searched case> ::=
    CASE
      <searched when clause>...
      [ <else clause> ]
    END

<searched when clause> ::= WHEN <search condition> THEN <result>

<cast specification> ::=
    CAST <left paren> <cast operand> AS
        <cast target> <right paren>

<cast operand> ::=
      <value expression>
    | NULL

<cast target> ::=
      <domain name>
    | <data type>

<numeric value function> ::=
      <position expression>
    | <extract expression>
    | <length expression>

<position expression> ::=
    POSITION <left paren> <character value expression>
        IN <character value expression> <right paren>

<character value expression> ::=
      <concatenation>
    | <character factor>

<concatenation> ::=
    <character value expression> <concatenation operator>
        <character factor>

<character factor> ::=
    <character primary> [ <collate clause> ]

<character primary> ::=
      <value expression primary>
    | <string value function>

<string value function> ::=
      <character value function>
    | <bit value function>

<character value function> ::=
      <character substring function>
    | <fold>
    | <form-of-use conversion>
    | <character translation>
    | <trim function>

<character substring function> ::=
    SUBSTRING <left paren> <character value expression> FROM <start position>
                [ FOR <string length> ] <right paren>
```

```
<start position> ::= <numeric value expression>

<string length> ::= <numeric value expression>

<fold> ::= ( UPPER | LOWER )
     <left paren> <character value expression> <right paren>

<form-of-use conversion> ::=
    CONVERT <left paren> <character value expression>
        USING <form-of-use conversion name> <right paren>

<form-of-use conversion name> ::= <qualified name>

<character translation> ::=
    TRANSLATE <left paren> <character value expression>
        USING <translation name> <right paren>

<translation name> ::= <qualified name>

<trim function> ::=
    TRIM <left paren> <trim operands> <right paren>

<trim operands> ::=
    [ [ <trim specification> ] [ <trim character> ] FROM ] <trim source>

<trim specification> ::=
      LEADING
    | TRAILING
    | BOTH

<trim character> ::= <character value expression>

<trim source> ::= <character value expression>

<bit value function> ::=
    <bit substring function>

<bit substring function> ::=
    SUBSTRING <left paren> <bit value expression> FROM <start position>
        [ FOR <string length> ] <right paren>

<bit value expression> ::=
      <bit concatenation>
    | <bit factor>

<bit concatenation> ::=
    <bit value expression> <concatenation operator> <bit factor>

<bit factor> ::= <bit primary>

<bit primary> ::=
      <value expression primary>
    | <string value function>

<extract expression> ::=
    EXTRACT <left paren> <extract field>
        FROM <extract source> <right paren>

<extract field> ::=
      <datetime field>
    | <time zone field>
```

```
<datetime field> ::=
      <non-second datetime field>
    | SECOND

<time zone field> ::=
      TIMEZONE_HOUR
    | TIMEZONE_MINUTE

<extract source> ::=
      <datetime value expression>
    | <interval value expression>

<datetime value expression> ::=
      <datetime term>
    | <interval value expression> <plus sign> <datetime term>
    | <datetime value expression> <plus sign> <interval term>
    | <datetime value expression> <minus sign> <interval term>

<interval term> ::=
      <interval factor>
    | <interval term 2> <asterisk> <factor>
    | <interval term 2> <solidus> <factor>
    | <term> <asterisk> <interval factor>

<interval factor> ::=
    [ <sign> ] <interval primary>

<interval primary> ::=
      <value expression primary> [ <interval qualifier> ]

<interval term 2> ::= <interval term>

<interval value expression> ::=
      <interval term>
    | <interval value expression 1> <plus sign> <interval term 1>
    | <interval value expression 1> <minus sign> <interval term 1>
    | <left paren> <datetime value expression> <minus sign>
          <datetime term> <right paren> <interval qualifier>

<interval value expression 1> ::= <interval value expression>

<interval term 1> ::= <interval term>

<datetime term> ::=
      <datetime factor>

<datetime factor> ::=
      <datetime primary> [ <time zone> ]

<datetime primary> ::=
      <value expression primary>
    | <datetime value function>

<time zone> ::=
    AT <time zone specifier>

<time zone specifier> ::=
      LOCAL
    | TIME ZONE <interval value expression>

<length expression> ::=
```

```
              <char length expression>
            | <octet length expression>
            | <bit length expression>

      <char length expression> ::=
            ( CHAR_LENGTH | CHARACTER_LENGTH )
                <left paren> <string value expression> <right paren>

      <string value expression> ::=
              <character value expression>
            | <bit value expression>

      <octet length expression> ::=
            OCTET_LENGTH <left paren> <string value expression> <right paren>

      <bit length expression> ::=
            BIT_LENGTH <left paren> <string value expression> <right paren>

      <null specification> ::=
            NULL

      <default specification> ::=
            DEFAULT

      <row value constructor list> ::=
            <row value constructor element>
                [ ( <comma> <row value constructor element> )... ]

      <row subquery> ::= <subquery>

      <comp op> ::=
              <equals operator>
            | <not equals operator>
            | <less than operator>
            | <greater than operator>
            | <less than or equals operator>
            | <greater than or equals operator>

      <between predicate> ::=
            <row value constructor> [ NOT ] BETWEEN
              <row value constructor> AND <row value constructor>

      <in predicate> ::=
            <row value constructor>
              [ NOT ] IN <in predicate value>

      <in predicate value> ::=
              <table subquery>
            | <left paren> <in value list> <right paren>

      <in value list> ::=
            <value expression> ( <comma> <value expression> )...

      <like predicate> ::=
            <match value> [ NOT ] LIKE <pattern>
              [ ESCAPE <escape character> ]

      <match value> ::= <character value expression>

      <pattern> ::= <character value expression>

      <escape character> ::= <character value expression>
```

```
<null predicate> ::= <row value constructor>
    IS [ NOT ] NULL

<quantified comparison predicate> ::=
    <row value constructor> <comp op> <quantifier> <table subquery>

<quantifier> ::= <all> | <some>

<all> ::= ALL

<some> ::= SOME | ANY

<exists predicate> ::= EXISTS <table subquery>

<unique predicate> ::= UNIQUE <table subquery>

<match predicate> ::=
    <row value constructor> MATCH [ UNIQUE ]
        [ PARTIAL | FULL ] <table subquery>

<overlaps predicate> ::=
    <row value constructor 1> OVERLAPS <row value constructor 2>

<row value constructor 1> ::= <row value constructor>

<row value constructor 2> ::= <row value constructor>

<truth value> ::=
      TRUE
    | FALSE
    | UNKNOWN

<constraint attributes> ::=
      <constraint check time> [ [ NOT ] DEFERRABLE ]
    | [ NOT ] DEFERRABLE [ <constraint check time> ]

<constraint check time> ::=
      INITIALLY DEFERRED
    | INITIALLY IMMEDIATE

<table constraint definition> ::=
    [ <constraint name definition> ]
    <table constraint> [ <constraint attributes> ]

<table constraint> ::=
      <unique constraint definition>
    | <referential constraint definition>
    | <check constraint definition>

<unique constraint definition> ::=
            <unique specification> even in SQL3)
    <unique specification>
      <left paren> <unique column list> <right paren>

<unique column list> ::= <column name list>

<referential constraint definition> ::=
    FOREIGN KEY
        <left paren> <referencing columns> <right paren>
      <references specification>
```

```
<referencing columns> ::=
    <reference column list>

<module contents> ::=
      <declare cursor>
    | <dynamic declare cursor>
    | <procedure>

<declare cursor> ::=
    DECLARE <cursor name> [ INSENSITIVE ] [ SCROLL ] CURSOR
      FOR <cursor specification>

<cursor name> ::= <identifier>

<cursor specification> ::=
    <query expression> [ <order by clause> ]
      [ <updatability clause> ]

<order by clause> ::=
    ORDER BY <sort specification list>

<sort specification list> ::=
    <sort specification> [ ( <comma> <sort specification> )... ]

<sort specification> ::=
    <sort key> [ <collate clause> ] [ <ordering specification> ]

<sort key> ::=
      <column name>
    | <unsigned integer>

<ordering specification> ::= ASC | DESC

<updatability clause> ::=
    FOR
        ( READ ONLY |
          UPDATE [ OF <column name list> ] )

<dynamic declare cursor> ::=
    DECLARE <cursor name> [ INSENSITIVE ] [ SCROLL ] CURSOR
        FOR <statement name>

<statement name> ::= <identifier>

<procedure> ::=
    PROCEDURE <procedure name>
        <parameter declaration list> <semicolon>
      <SQL procedure statement> <semicolon>

<procedure name> ::= <identifier>

<parameter declaration list> ::=
      <left paren> <parameter declaration>
          [ ( <comma> <parameter declaration> )... ] <right paren>
    | <parameter declaration>...

<parameter declaration> ::=
      <parameter name> <data type>
    | <status parameter>

<status parameter> ::=
    SQLCODE | SQLSTATE
```

```
<SQL procedure statement> ::=
      <SQL schema statement>
    | <SQL data statement>
    | <SQL transaction statement>
    | <SQL connection statement>
    | <SQL session statement>
    | <SQL dynamic statement>
    | <SQL diagnostics statement>

<SQL schema statement> ::=
      <SQL schema definition statement>
    | <SQL schema manipulation statement>

<SQL schema definition statement> ::=
      <schema definition>
    | <table definition>
    | <view definition>
    | <grant statement>
    | <domain definition>
    | <character set definition>
    | <collation definition>
    | <translation definition>
    | <assertion definition>

<schema definition> ::=
    CREATE SCHEMA <schema name clause>
      [ <schema character set specification> ]
      [ <schema element>... ]

<schema name clause> ::=
      <schema name>
    | AUTHORIZATION <schema authorization identifier>
    | <schema name> AUTHORIZATION
          <schema authorization identifier>

<schema authorization identifier> ::=
    <authorization identifier>

<schema character set specification> ::=
    DEFAULT CHARACTER
        SET <character set specification>

<schema element> ::=
      <domain definition>
    | <table definition>
    | <view definition>
    | <grant statement>
    | <assertion definition>
    | <character set definition>
    | <collation definition>
    | <translation definition>

<domain definition> ::=
    CREATE DOMAIN <domain name>
        [ AS ] <data type>
      [ <default clause> ]
      [ <domain constraint>... ]
      [ <collate clause> ]

<domain constraint> ::=
    [ <constraint name definition> ]
```

```
        <check constraint definition> [ <constraint attributes> ]

    <table definition> ::=
        CREATE [ ( GLOBAL | LOCAL ) TEMPORARY ] TABLE
            <table name>
          <table element list>
          [ ON COMMIT ( DELETE | PRESERVE ) ROWS ]

    <view definition> ::=
        CREATE VIEW <table name> [ <left paren> <view column list>
                                      <right paren> ]
          AS <query expression>
          [ WITH [ <levels clause> ] CHECK OPTION ]

    <view column list> ::= <column name list>

    <levels clause> ::=
        CASCADED | LOCAL

    <grant statement> ::=
       GRANT <privileges> ON <object name>
         TO <grantee> [ ( <comma> <grantee> )... ]
           [ WITH GRANT OPTION ]

    <privileges> ::=
          ALL PRIVILEGES
        | <action list>

    <action list> ::= <action> [ ( <comma> <action> )... ]

    <action> ::=
          SELECT
        | DELETE
        | INSERT [ <left paren> <privilege column list> <right paren> ]
        | UPDATE [ <left paren> <privilege column list> <right paren> ]
        | REFERENCES [ <left paren> <privilege column list> <right paren> ]
        | USAGE

    <privilege column list> ::= <column name list>

    <object name> ::=
          [ TABLE ] <table name>
        | DOMAIN <domain name>
        | COLLATION <collation name>
        | CHARACTER SET <character set name>
        | TRANSLATION <translation name>

    <grantee> ::=
          PUBLIC
        | <authorization identifier>

    <assertion definition> ::=
        CREATE ASSERTION <constraint name> <assertion check>
          [ <constraint attributes> ]

    <assertion check> ::=
        CHECK
            <left paren> <search condition> <right paren>

    <character set definition> ::=
        CREATE CHARACTER SET <character set name>
            [ AS ]
```

```
          <character set source>
          [ <collate clause> | <limited collation definition> ]

    <character set source> ::=
          GET <existing character set name>

    <existing character set name> ::=
          <standard character repertoire name>
        | <implementation-defined character repertoire name>
        | <schema character set name>

    <schema character set name> ::= <character set name>

    <limited collation definition> ::=
        COLLATION FROM <collation source>

    <collation source> ::=
          <collating sequence definition>
        | <translation collation>

    <collating sequence definition> ::=
          <external collation>
        | <schema collation name>
        | DESC <left paren> <collation name> <right paren>
        | DEFAULT

    <external collation> ::=
        EXTERNAL <left paren> <quote> <external collation name> <quote> <right paren>

    <external collation name> ::=
          <standard collation name>
        | <implementation-defined collation name>

    <standard collation name> ::= <collation name>

    <implementation-defined collation name> ::= <collation name>

    <schema collation name> ::= <collation name>

    <translation collation> ::=
        TRANSLATION <translation name>
            [ THEN COLLATION <collation name> ]

    <collation definition> ::=
        CREATE COLLATION <collation name> FOR
            <character set specification>
          FROM <collation source>
            [ <pad attribute> ]

    <pad attribute> ::=
          NO PAD
        | PAD SPACE

    <translation definition> ::=
        CREATE TRANSLATION <translation name>
          FOR <source character set specification>
            TO <target character set specification>
          FROM <translation source>

    <source character set specification> ::= <character set specification>

    <target character set specification> ::= <character set specification>
```

```
<translation source> ::=
      <translation specification>

<translation specification> ::=
      <external translation>
    | IDENTITY
    | <schema translation name>

<external translation> ::=
    EXTERNAL <left paren> <quote> <external translation name> <quote> <right paren>

<external translation name> ::=
      <standard translation name>
    | <implementation-defined translation name>

<standard translation name> ::= <translation name>

<implementation-defined translation name> ::= <translation name>

<schema translation name> ::= <translation name>

<SQL schema manipulation statement> ::=
      <drop schema statement>
    | <alter table statement>
    | <drop table statement>
    | <drop view statement>
    | <revoke statement>
    | <alter domain statement>
    | <drop domain statement>
    | <drop character set statement>
    | <drop collation statement>
    | <drop translation statement>
    | <drop assertion statement>

<drop schema statement> ::=
    DROP SCHEMA <schema name> <drop behavior>

<drop behavior> ::= CASCADE | RESTRICT

<alter table statement> ::=
    ALTER TABLE <table name> <alter table action>

<alter table action> ::=
      <add column definition>
    | <alter column definition>
    | <drop column definition>
    | <add table constraint definition>
    | <drop table constraint definition>

<add column definition> ::=
    ADD [ COLUMN ] <column definition>

<alter column definition> ::=
    ALTER [ COLUMN ] <column name> <alter column action>

<alter column action> ::=
      <set column default clause>
    | <drop column default clause>

<set column default clause> ::=
    SET <default clause>
```

```
<drop column default clause> ::=
    DROP DEFAULT

<drop column definition> ::=
    DROP [ COLUMN ] <column name> <drop behavior>

<add table constraint definition> ::=
    ADD <table constraint definition>

<drop table constraint definition> ::=
    DROP CONSTRAINT <constraint name> <drop behavior>

<drop table statement> ::=
    DROP TABLE <table name> <drop behavior>

<drop view statement> ::=
    DROP VIEW <table name> <drop behavior>

<revoke statement> ::=
    REVOKE [ GRANT OPTION FOR ]
        <privileges>
        ON <object name>
      FROM <grantee> [ ( <comma> <grantee> )... ] <drop behavior>

<alter domain statement> ::=
    ALTER DOMAIN <domain name> <alter domain action>

<alter domain action> ::=
      <set domain default clause>
    | <drop domain default clause>
    | <add domain constraint definition>
    | <drop domain constraint definition>

<set domain default clause> ::= SET <default clause>

<drop domain default clause> ::= DROP DEFAULT

<add domain constraint definition> ::=
    ADD <domain constraint>

<drop domain constraint definition> ::=
    DROP CONSTRAINT <constraint name>

<drop domain statement> ::=
    DROP DOMAIN <domain name> <drop behavior>

<drop character set statement> ::=
    DROP CHARACTER SET <character set name>

<drop collation statement> ::=
    DROP COLLATION <collation name>

<drop translation statement> ::=
    DROP TRANSLATION <translation name>

<drop assertion statement> ::=
    DROP ASSERTION <constraint name>

<SQL data statement> ::=
      <open statement>
    | <fetch statement>
```

```
    | <close statement>
    | <select statement: single row>
    | <SQL data change statement>

<open statement> ::=
    OPEN <cursor name>

<fetch statement> ::=
    FETCH [ [ <fetch orientation> ] FROM ]
      <cursor name> INTO <fetch target list>

<fetch orientation> ::=
      NEXT
    | PRIOR
    | FIRST
    | LAST
    | ( ABSOLUTE | RELATIVE ) <simple value specification>

<simple value specification> ::=
      <parameter name>
    | <embedded variable name>
    | <literal>

<fetch target list> ::=
    <target specification> [ ( <comma> <target specification> )... ]

<target specification> ::=
      <parameter specification>
    | <variable specification>

<close statement> ::=
    CLOSE <cursor name>

<select statement: single row> ::=
    SELECT [ <set quantifier> ] <select list>
      INTO <select target list>
        <table expression>

<select target list> ::=
    <target specification> [ ( <comma> <target specification> )... ]

<SQL data change statement> ::=
      <delete statement: positioned>
    | <delete statement: searched>
    | <insert statement>
    | <update statement: positioned>
    | <update statement: searched>

<delete statement: positioned> ::=
    DELETE FROM <table name>
      WHERE CURRENT OF <cursor name>

<delete statement: searched> ::=
    DELETE FROM <table name>
      [ WHERE <search condition> ]

<insert statement> ::=
    INSERT INTO <table name>
      <insert columns and source>

<insert columns and source> ::=
      [ <left paren> <insert column list> <right paren> ]
```

```
            <query expression>
      | DEFAULT VALUES

<insert column list> ::= <column name list>

<update statement: positioned> ::=
    UPDATE <table name>
      SET <set clause list>
        WHERE CURRENT OF <cursor name>

<set clause list> ::=
    <set clause> [ ( <comma> <set clause> )... ]

<set clause> ::=
    <object column> <equals operator> <update source>

<object column> ::= <column name>

<update source> ::=
      <value expression>
    | <null specification>
    | DEFAULT

<update statement: searched> ::=
    UPDATE <table name>
      SET <set clause list>
      [ WHERE <search condition> ]

<SQL transaction statement> ::=
      <set transaction statement>
    | <set constraints mode statement>
    | <commit statement>
    | <rollback statement>

<set transaction statement> ::=
    SET TRANSACTION <transaction mode>
        [ ( <comma> <transaction mode> )... ]

<transaction mode> ::=
      <isolation level>
    | <transaction access mode>
    | <diagnostics size>

<isolation level> ::=
    ISOLATION LEVEL <level of isolation>

<level of isolation> ::=
      READ UNCOMMITTED
    | READ COMMITTED
    | REPEATABLE READ
    | SERIALIZABLE

<transaction access mode> ::=
      READ ONLY
    | READ WRITE

<diagnostics size> ::=
    DIAGNOSTICS SIZE <number of conditions>

<number of conditions> ::= <simple value specification>

<set constraints mode statement> ::=
```

```
    SET CONSTRAINTS <constraint name list>
        ( DEFERRED | IMMEDIATE )

<constraint name list> ::=
      ALL
    | <constraint name> [ ( <comma> <constraint name> )... ]

<commit statement> ::=
    COMMIT [ WORK ]

<rollback statement> ::=
    ROLLBACK [ WORK ]

<SQL connection statement> ::=
      <connect statement>
    | <set connection statement>
    | <disconnect statement>

<connect statement> ::=
    CONNECT TO <connection target>

<connection target> ::=
      <SQL-server name>
        [ AS <connection name> ]
          correspondence with Tony Gordon)
        [ USER <user name> ]
    | DEFAULT

<SQL-server name> ::= <simple value specification>

<connection name> ::= <simple value specification>

<user name> ::= <simple value specification>

<set connection statement> ::=
    SET CONNECTION <connection object>

<connection object> ::=
      DEFAULT
    | <connection name>

<disconnect statement> ::=
    DISCONNECT <disconnect object>

<disconnect object> ::=
      <connection object>
    | ALL
    | CURRENT

<SQL session statement> ::=
      <set catalog statement>
    | <set schema statement>
    | <set names statement>
    | <set session authorization identifier statement>
    | <set local time zone statement>

<set catalog statement> ::=
    SET CATALOG <value specification>

<value specification> ::=
      <literal>
    | <general value specification>
```

```
<set schema statement> ::=
    SET SCHEMA <value specification>

<set names statement> ::=
    SET NAMES <value specification>

<set session authorization identifier statement> ::=
    SET SESSION AUTHORIZATION
        <value specification>

<set local time zone statement> ::=
    SET TIME ZONE
        <set time zone value>

<set time zone value> ::=
      <interval value expression>
    | LOCAL

<SQL dynamic statement> ::=
      <system descriptor statement>
    | <prepare statement>
    | <deallocate prepared statement>
    | <describe statement>
    | <execute statement>
    | <execute immediate statement>
    | <SQL dynamic data statement>

<system descriptor statement> ::=
      <allocate descriptor statement>
    | <deallocate descriptor statement>
    | <set descriptor statement>
    | <get descriptor statement>

<allocate descriptor statement> ::=
    ALLOCATE DESCRIPTOR <descriptor name>
       [ WITH MAX <occurrences> ]

<descriptor name> ::=
    [ <scope option> ] <simple value specification>

<scope option> ::=
      GLOBAL
    | LOCAL

<occurrences> ::= <simple value specification>

<deallocate descriptor statement> ::=
    DEALLOCATE DESCRIPTOR <descriptor name>

<set descriptor statement> ::=
    SET DESCRIPTOR <descriptor name>
        <set descriptor information>

<set descriptor information> ::=
      <set count>
    | VALUE <item number>
        <set item information> [ ( <comma> <set item information> )... ]

<set count> ::=
    COUNT <equals operator> <simple value specification 1>
```

```
<simple value specification 1> ::= <simple value specification>

<item number> ::= <simple value specification>

<set item information> ::=
    <descriptor item name> <equals operator> <simple value specification 2>

<descriptor item name> ::=
      TYPE
    | LENGTH
    | OCTET_LENGTH
    | RETURNED_LENGTH
    | RETURNED_OCTET_LENGTH
    | PRECISION
    | SCALE
    | DATETIME_INTERVAL_CODE
    | DATETIME_INTERVAL_PRECISION
    | NULLABLE
    | INDICATOR
    | DATA
    | NAME
    | UNNAMED
    | COLLATION_CATALOG
    | COLLATION_SCHEMA
    | COLLATION_NAME
    | CHARACTER_SET_CATALOG
    | CHARACTER_SET_SCHEMA
    | CHARACTER_SET_NAME

<simple value specification 2> ::= <simple value specification>

<item number> ::= <simple value specification>

<get descriptor statement> ::=
    GET DESCRIPTOR <descriptor name> <get descriptor information>

<get descriptor information> ::=
      <get count>
    | VALUE <item number>
        <get item information> [ ( <comma> <get item information> )... ]

<get count> ::=
    <simple target specification 1> <equals operator>
         COUNT

<simple target specification 1> ::= <simple target specification>

<simple target specification> ::=
      <parameter name>
    | <embedded variable name>

<get item information> ::=
    <simple target specification 2> <equals operator> <descriptor item name>>

<simple target specification 2> ::= <simple target specification>

<prepare statement> ::=
    PREPARE <SQL statement name> FROM <SQL statement variable>

<SQL statement name> ::=
      <statement name>
    | <extended statement name>
```

```
<extended statement name> ::=
    [ <scope option> ] <simple value specification>

<SQL statement variable> ::= <simple value specification>

<deallocate prepared statement> ::=
    DEALLOCATE PREPARE <SQL statement name>

<describe statement> ::=
      <describe input statement>
    | <describe output statement>

<describe input statement> ::=
    DESCRIBE INPUT <SQL statement name> <using descriptor>

<using descriptor> ::=
    ( USING | INTO ) SQL DESCRIPTOR <descriptor name>

<describe output statement> ::=
    DESCRIBE [ OUTPUT ] <SQL statement name> <using descriptor>

<execute statement> ::=
    EXECUTE <SQL statement name>
      [ <result using clause> ]
      [ <parameter using clause> ]

<result using clause> ::= <using clause>

<using clause> ::=
      <using arguments>
    | <using descriptor>

<using arguments> ::=
    ( USING | INTO ) <argument> [ ( <comma> <argument> )... ]

<argument> ::= <target specification>

<parameter using clause> ::= <using clause>

<execute immediate statement> ::=
    EXECUTE IMMEDIATE <SQL statement variable>

<SQL dynamic data statement> ::=
      <allocate cursor statement>
    | <dynamic open statement>
    | <dynamic fetch statement>
    | <dynamic close statement>
    | <dynamic delete statement: positioned>
    | <dynamic update statement: positioned>

<allocate cursor statement> ::=
    ALLOCATE <extended cursor name> [ INSENSITIVE ]
        [ SCROLL ] CURSOR
      FOR <extended statement name>

<extended cursor name> ::=
    [ <scope option> ] <simple value specification>

<dynamic open statement> ::=
    OPEN <dynamic cursor name> [ <using clause> ]
```

```
<dynamic cursor name> ::=
      <cursor name>
    | <extended cursor name>

<dynamic fetch statement> ::=
    FETCH [ [ <fetch orientation> ] FROM ] <dynamic cursor name>
        <using clause>

<dynamic close statement> ::=
    CLOSE <dynamic cursor name>

<dynamic delete statement: positioned> ::=
    DELETE FROM <table name>
      WHERE CURRENT OF
          <dynamic cursor name>

<dynamic update statement: positioned> ::=
    UPDATE <table name>
      SET <set clause>
          [ ( <comma> <set clause> )... ]
        WHERE CURRENT OF
            <dynamic cursor name>

<SQL diagnostics statement> ::=
    <get diagnostics statement>

<get diagnostics statement> ::=
    GET DIAGNOSTICS <sql diagnostics information>

<sql diagnostics information> ::=
      <statement information>
    | <condition information>

<statement information> ::=
    <statement information item> [ ( <comma> <statement information item> )... ]

<statement information item> ::=
    <simple target specification> <equals operator> <statement information item name>

<statement information item name> ::=
      NUMBER
    | MORE
    | COMMAND_FUNCTION
    | DYNAMIC_FUNCTION
    | ROW_COUNT

<condition information> ::=
    EXCEPTION <condition number>
      <condition information item> [ ( <comma> <condition information item> )... ]

<condition number> ::= <simple value specification>

<condition information item> ::=
    <simple target specification> <equals operator> <condition information item name>

<condition information item name> ::=
      CONDITION_NUMBER
    | RETURNED_SQLSTATE
    | CLASS_ORIGIN
    | SUBCLASS_ORIGIN
    | SERVER_NAME
    | CONNECTION_NAME
```

```
      | CONSTRAINT_CATALOG
      | CONSTRAINT_SCHEMA
      | CONSTRAINT_NAME
      | CATALOG_NAME
      | SCHEMA_NAME
      | TABLE_NAME
      | COLUMN_NAME
      | CURSOR_NAME
      | MESSAGE_TEXT
      | MESSAGE_LENGTH
      | MESSAGE_OCTET_LENGTH

<embedded SQL host program> ::=
        <embedded SQL Ada program>
      | <embedded SQL C program>
      | <embedded SQL COBOL program>
      | <embedded SQL Fortran program>
      | <embedded SQL MUMPS program>
      | <embedded SQL Pascal program>
      | <embedded SQL PL/I program>

<embedded SQL Ada program> ::= !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL C program> ::=
        !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL COBOL program> ::= !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL Fortran program> ::=
    !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL MUMPS program> ::= !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL Pascal program> ::=
    !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL PL/I program> ::= !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL declare section> ::=
        <embedded SQL begin declare>
          [ <embedded character set declaration> ]
          [ <host variable definition>... ]
        <embedded SQL end declare>
      | <embedded SQL MUMPS declare>

<embedded SQL begin declare> ::=
    <SQL prefix> BEGIN DECLARE SECTION
        [ <SQL terminator> ]

<SQL prefix> ::=
        EXEC SQL
      | <ampersand>SQL<left paren>

<SQL terminator> ::=
        END-EXEC
      | <semicolon>
      | <right paren>

<embedded character set declaration> ::=
    SQL NAMES ARE <character set specification>

<host variable definition> ::=
```

```
      <Ada variable definition>
    | <C variable definition>
    | <COBOL variable definition>
    | <Fortran variable definition>
    | <MUMPS variable definition>
    | <Pascal variable definition>
    | <PL/I variable definition>

<Ada variable definition> ::=
    <Ada host identifier> [ ( <comma> <Ada host identifier> )... ] :
    <Ada type specification> [ <Ada initial value> ]

<Ada type specification> ::=
      <Ada qualified type specification>
    | <Ada unqualified type specification>

<Ada qualified type specification> ::=
      SQL_STANDARD.CHAR [ CHARACTER SET
         [ IS ] <character set specification> ]
          <left paren> 1 <double period> <length> <right paren>
    | SQL_STANDARD.BIT
          <left paren> 1 <double period> <length> <right paren>
    | SQL_STANDARD.SMALLINT
    | SQL_STANDARD.INT
    | SQL_STANDARD.REAL
    | SQL_STANDARD.DOUBLE_PRECISION
    | SQL_STANDARD.SQLCODE_TYPE
    | SQL_STANDARD.SQLSTATE_TYPE
    | SQL_STANDARD.INDICATOR_TYPE

<Ada unqualified type specification> ::=
      CHAR
          <left paren> 1 <double period> <length> <right paren>
    | BIT
          <left paren> 1 <double period> <length> <right paren>
    | SMALLINT
    | INT
    | REAL
    | DOUBLE_PRECISION
    | SQLCODE_TYPE
    | SQLSTATE_TYPE
    | INDICATOR_TYPE

<Ada initial value> ::=
    <Ada assignment operator> <character representation>...

<Ada assignment operator> ::= <colon><equals operator>

<C variable definition> ::=
      [ <C storage class> ]
      [ <C class modifier> ]
      <C variable specification>
    <semicolon>

<C storage class> ::=
      auto
    | extern
    | static

<C class modifier> ::= const | volatile

<C variable specification> ::=
```

```
      <C numeric variable>
    | <C character variable>
    | <C derived variable>

<C numeric variable> ::=
    ( long | short | float | double )
      <C host identifier> [ <C initial value> ]
            [ ( <comma> <C host identifier> [ <C initial value> ] )... ]

<C initial value> ::=
    <equals operator> <character representation>...

<C character variable> ::=
    char [ CHARACTER SET
             [ IS ] <character set specification> ]
      <C host identifier>
        <C array specification> [ <C initial value> ]
        [ ( <comma> <C host identifier>
          <C array specification>
                 [ <C initial value> ] )... ]

<C array specification> ::=
    <left bracket> <length> <right bracket>

<C derived variable> ::=
      <C VARCHAR variable>
    | <C bit variable>

<C VARCHAR variable> ::=
    VARCHAR [ CHARACTER SET [ IS ]
        <character set specification> ]
        <C host identifier>
            <C array specification> [ <C initial value> ]
          [ ( <comma> <C host identifier>
              <C array specification>
                      [ <C initial value> ] )... ]

<C bit variable> ::=
    BIT <C host identifier>
        <C array specification> [ <C initial value> ]
      [ ( <comma> <C host identifier>
        <C array specification>
                   [ <C initial value> ] )... ]

<COBOL variable definition> ::=
    (01|77) <COBOL host identifier> <COBOL type specification>
      [ <character representation>... ] <period>

<COBOL type specification> ::=
      <COBOL character type>
    | <COBOL bit type>
    | <COBOL numeric type>
    | <COBOL integer type>

<COBOL character type> ::=
    [ CHARACTER SET [ IS ]
          <character set specification> ]
    ( PIC | PICTURE ) [ IS ] ( X [ <left paren> <length> <right paren> ] )...

<COBOL bit type> ::=
    ( PIC | PICTURE ) [ IS ]
        ( B [ <left paren> <length> <right paren> ] )...
```

```
<COBOL numeric type> ::=
    ( PIC | PICTURE ) [ IS ]
      S <COBOL nines specification>
    [ USAGE [ IS ] ] DISPLAY SIGN LEADING SEPARATE

<COBOL nines specification> ::=
      <COBOL nines> [ V [ <COBOL nines> ] ]
    | V <COBOL nines>

<COBOL nines> ::= ( 9 [ <left paren> <length> <right paren> ] )...

<COBOL integer type> ::=
      <COBOL computational integer>
    | <COBOL binary integer>

<COBOL computational integer> ::=
    ( PIC | PICTURE ) [ IS ] S<COBOL nines>
      [ USAGE [ IS ] ] ( COMP | COMPUTATIONAL )

<COBOL binary integer> ::=
    ( PIC | PICTURE ) [ IS ] S<COBOL nines>
      [ USAGE [ IS ] ] BINARY

<Fortran variable definition> ::=
    <Fortran type specification>
    <Fortran host identifier>
        [ ( <comma> <Fortran host identifier> )... ]

<Fortran type specification> ::=
      CHARACTER [ <asterisk> <length> ]
          [ CHARACTER SET [ IS ]
                <character set specification> ]
    | BIT [ <asterisk> <length> ]
    | INTEGER
    | REAL
    | DOUBLE PRECISION

<MUMPS variable definition> ::=
    ( <MUMPS numeric variable> | <MUMPS character variable> )
        <semicolon>

<MUMPS numeric variable> ::=
    <MUMPS type specification>
      <MUMPS host identifier> [ ( <comma> <MUMPS host identifier> )... ]

<MUMPS type specification> ::=
      INT
    | DEC
          [ <left paren> <precision> [ <comma> <scale> ] <right paren> ]
    | REAL

<MUMPS character variable> ::=
    VARCHAR <MUMPS host identifier> <MUMPS length specification>
      [ ( <comma> <MUMPS host identifier> <MUMPS length specification> )... ]

<MUMPS length specification> ::=
    <left paren> <length> <right paren>

<Pascal variable definition> ::=
    <Pascal host identifier> [ ( <comma> <Pascal host identifier> )... ] <colon>
      <Pascal type specification> <semicolon>
```

```
<Pascal type specification> ::=
      PACKED ARRAY
          <left bracket> 1 <double period> <length> <right bracket>
        OF CHAR
          [ CHARACTER SET [ IS ]
                <character set specification> ]
    | PACKED ARRAY
          <left bracket> 1 <double period> <length> <right bracket>
        OF BIT
    | INTEGER
    | REAL
    | CHAR [ CHARACTER SET
                                [ IS ] <character set specification> ]
    | BIT

<PL/I variable definition> ::=
    (DCL | DECLARE)
        (   <PL/I host identifier>
          | <left paren> <PL/I host identifier>
                [ ( <comma> <PL/I host identifier> )... ] <right paren> )
    <PL/I type specification>
    [ <character representation>... ] <semicolon>

<PL/I type specification> ::=
      ( CHAR | CHARACTER ) [ VARYING ]
          <left paren><length><right paren>
          [ CHARACTER SET
                [ IS ] <character set specification> ]
    | BIT [ VARYING ] <left paren><length><right paren>
    | <PL/I type fixed decimal> <left paren> <precision>
          [ <comma> <scale> ] <right paren>
    | <PL/I type fixed binary> [ <left paren> <precision> <right paren> ]
    | <PL/I type float binary> <left paren> <precision> <right paren>

<PL/I type fixed decimal> ::=
      ( DEC | DECIMAL ) FIXED
    | FIXED ( DEC | DECIMAL )

<PL/I type fixed binary> ::=
      ( BIN | BINARY ) FIXED
    | FIXED ( BIN | BINARY )

<PL/I type float binary> ::=
      ( BIN | BINARY ) FLOAT
    | FLOAT ( BIN | BINARY )

<embedded SQL end declare> ::=
    <SQL prefix> END DECLARE SECTION
        [ <SQL terminator> ]

<embedded SQL MUMPS declare> ::=
    <SQL prefix>
      BEGIN DECLARE SECTION
        [ <embedded character set declaration> ]
        [ <host variable definition>... ]
      END DECLARE SECTION
    <SQL terminator>

<embedded SQL statement> ::=
    <SQL prefix>
      <statement or declaration>
```

```
    [ <SQL terminator> ]

<statement or declaration> ::=
      <declare cursor>
    | <dynamic declare cursor>
    | <temporary table declaration>
    | <embedded exception declaration>
    | <SQL procedure statement>

<embedded exception declaration> ::=
    WHENEVER <condition> <condition action>

<condition> ::=
    SQLERROR | NOT FOUND

<condition action> ::=
    CONTINUE | <go to>

<go to> ::=
    ( GOTO | GO TO ) <goto target>

<goto target> ::=
      <host label identifier>
    | <unsigned integer>
    | <host PL/I label variable>

<host label identifier> ::= !!<EMPHASIS>(See the Syntax Rules.)

<host PL/I label variable> ::= !!<EMPHASIS>(See the Syntax Rules.)

<preparable statement> ::=
      <preparable SQL data statement>
    | <preparable SQL schema statement>
    | <preparable SQL transaction statement>
    | <preparable SQL session statement>
    | <preparable implementation-defined statement>

<preparable SQL data statement> ::=
      <delete statement: searched>
    | <dynamic single row select statement>
    | <insert statement>
    | <dynamic select statement>
    | <update statement: searched>
    | <preparable dynamic delete statement: positioned>
    | <preparable dynamic update statement: positioned>

<dynamic single row select statement> ::= <query specification>

<dynamic select statement> ::= <cursor specification>

<preparable dynamic delete statement: positioned> ::=
   DELETE [ FROM <table name> ]
      WHERE CURRENT OF <cursor name>

<preparable dynamic update statement: positioned> ::=
   UPDATE [ <table name> ]
      SET <set clause list>
      WHERE CURRENT OF <cursor name>

<preparable SQL schema statement> ::=
      <SQL schema statement>
```

```
<preparable SQL transaction statement> ::=
      <SQL transaction statement>

<preparable SQL session statement> ::=
      <SQL session statement>

<preparable implementation-defined statement> ::=
    !! <EMPHASIS>(See the Syntax Rules.)

<direct SQL statement> ::=
    <directly executable statement> <semicolon>

<directly executable statement> ::=
      <direct SQL data statement>
    | <SQL schema statement>
    | <SQL transaction statement>
    | <SQL connection statement>
    | <SQL session statement>
    | <direct implementation-defined statement>

<direct SQL data statement> ::=
      <delete statement: searched>
    | <direct select statement: multiple rows>
    | <insert statement>
    | <update statement: searched>
    | <temporary table declaration>

<direct select statement: multiple rows> ::=
    <query expression> [ <order by clause> ]

<direct implementation-defined statement> ::=
    !!<EMPHASIS>(See the Syntax Rules)

<SQL object identifier> ::=
    <SQL provenance> <SQL variant>

<SQL provenance> ::= <arc1> <arc2> <arc3>

<arc1> ::= iso | 1 | iso <left paren> 1 <right paren>

<arc2> ::= standard | 0 | standard <left paren> 0 <right paren>

<arc3> ::= 9075

<SQL variant> ::= <SQL edition> <SQL conformance>

<SQL edition> ::= <1987> | <1989> | <1992>

<1987> ::= 0 | edition1987 <left paren> 0 <right paren>

<1989> ::= <1989 base> <1989 package>

<1989 base> ::= 1 | edition1989 <left paren> 1 <right paren>

<1989 package> ::= <integrity no> | <integrity yes>

<integrity no> ::= 0 | IntegrityNo <left paren> 0 <right paren>

<integrity yes> ::= 1 | IntegrityYes <left paren> 1 <right paren>

<1992> ::= 2 | edition1992 <left paren> 2 <right paren>
```

```
<SQL conformance> ::= <low> | <intermediate> | <high>

<low> ::= 0 | Low <left paren> 0 <right paren>

<intermediate> ::= 1 | Intermediate <left paren> 1 <right paren>

<high> ::= 2 | High <left paren> 2 <right paren>
```

39. Appendix B − SQL Tutorial for beginners

39.1 Tutorial for PostgreSQL

SQL tutorial is also distributed with PostgreSQL. The SQL tutorial scripts is in the directory src/tutorial

39.2 Internet URL pointers

The SQL tutorial for beginners can be found at

- Jim Hoffman's tutorial http://w3.one.net/~jhoffman/sqltut.htm
- Carnegie Mellon Univ http://www.heinz.cmu.edu/project/dbms Go here and click on 'technical'−>'SQL_examples.html' and others.
- Concord Univ http://www.cs.concordia.ca/Course_Notes/oracle/browser/node1.html

Comments or suggestions? Mail to

- Jim Hoffman jhoffman@one.net

The following are the sites suggested by John Hoffman:

- SQL Reference http://www.contrib.andrew.cmu.edu/~shadow/sql.html
- Ask the SQL Pro http://www.inquiry.com/techtips/thesqlpro/
- SQL Pro's Relational DB Useful Sites http://www.inquiry.com/techtips/thesqlpro/usefulsites.html
- Programmer's Source http://infoweb.magi.com/~steve/develop.html
- DBMS Sites http://info.itu.ch/special/wwwfiles Go here and see file comp_db.html
- DB Ingredients http://www.compapp.dcu.ie/databases/f017.html
- Web Authoring http://www.stars.com/Tutorial/CGI/
- Computing Dictionary http://wfn−shop.princeton.edu/cgi−bin/foldoc
- DBMS Lab/Links http://www−ccs.cs.umass.edu/db.html
- SQL FAQ http://epoch.CS.Berkeley.EDU:8000/sequoia/dba/montage/FAQ Go here and see file SQL_TOC.html
- SQL Databases http://chaos.mur.csu.edu.au/itc125/cgi/sqldb.html
- RIT Database Design Page http://www.it.rit.edu/~wjs/IT/199602/icsa720/icsa720postings.html
- Database Jump Site http://www.pcslink.com/~ej/dbweb.html
- Programming Tutorials on the Web http://www.eng.uc.edu/~jtilley/tutorial.html
- Development Resources http://www.ndev.com/ndc2/support/resources.htp
- Query List http://ashok.pair.com/sql.htm
- IMAGE SQL Miscellaneous http://jazz.external.hp.com/training/sqltables/main.html
- Internet Resource List http://www.eit.com/web/netservices.html

39.3 On−line SQL tutorials

Visit the following sites for on−line SQL tutorials

- SQL beginner course http://sqlcourse.com
- SQL advanced course http://sqlcourse2.com

---

40. Appendix C − Linux Quick Install Instructions

If you are planning to use PostgreSQL on Linux, and need help in installing Linux, then please visit the pointers given in this Appendix. They cover the following topics −

- Salient Features of Linux − Why Linux is better as a database server when compared with Windows 95/NT
- 10 minutes Linux Quick Install Instructions
- Microsoft−Linux Analogy List
- Quick Steps to Recompile the Linux Kernel

- Main site is at http://www.aldev.8m.com and mirrors at webjump, angelfire, geocities, virtualave, bizland, theglobe, spree, infoseek, bcity, 50megs

---

41. Appendix C − Midgard Installation

RPMs for Midgard from http://www.midgard−project.org/download/binaries currently do not include PostgreSQL, and hence you need to install from the source tar ball file .

Download the Midgard source tarball and read the INSTALL.REDHAT file −

```
bash# cd midgard-lib-1.4beta6
bash# ./configure --prefix=/usr/local --with-mysql=/usr/local --includedir=/usr/include/mysql --with-midgard=/usr/local --with-pgsql=/var/lib/pgsql --includedir=/usr/include/pgsql
bash# make
bash# make install
bash# ldconfig -v | grep -i midga
Copy the header files, just in case make install did not do that..
bash# cp *.h /usr/local/include


bash# cd ../mod_midgard-1.4beta5c
bash# ./configure --prefix=/usr/local --with-mysql=/usr/local --includedir=/usr/include/mysql --with-midgard=/usr --with-pgsql=/var/lib/pgsql --includedir=/usr/include/pgsql
bash# make
bash# make install
#modify apache line to correct /usr/.....
bash# vi /etc/httpd/conf/httpd.conf   (or it is /etc/apache/httpd.conf)
bash# /etc/init.d/apache restart
#apache should restart!!!


bash# cd ../midgard-php-1.4beta6
bash# ./configure '--with-apxs' '--with-mysql' '--with-pgsql' '--with-midgard' --prefix=/usr/local --with-midgard=/usr/local
```

```
bash# gvim Makefile
And add -I/usr/include/pgsql to INCLUDE variable.

Also add $(INCLUDE) to $(APXS) command as below -
libphp3.so: mod_php3.c libmodphp3-so.a  pcrelib/libpcre.a midgard/libphpmidgard.a
        -@test -f ./mod_php3.c || test -L ./mod_php3.c || $(LN_S) $(srcdir)/mod_php3.c ./mod_php3.c
        -@test -f ./mod_php3.c || test -h ./mod_php3.c || $(LN_S) $(srcdir)/mod_php3.c ./mod_php3.c
        $(APXS) -c -o libphp3.so  -I$(srcdir) $(INCLUDE) -I. -I/usr/local/include -I/usr/lib/glib/include  -Wl,'-rpath /usr/local/lib' ./mod_php3.c libmodphp3-so.a $(APXS_LDFLAGS)

bash# make
bash# make install
#modify apache line to correct /usr/.....
# and add lines like these -
        LoadModule php4_module        modules/libphp4.so
        AddModule mod_php4.c
        LoadModule php4_module        lib/apache/libphp4.so

        <IfModule mod_php4.c>
                AddType application/x-httpd-php4 .php4
                AddType application/x-httpd-php4 .php
                AddType application/x-httpd-php4-source .phps
                AddType application/x-httpd-php .php
        </IfModule>

bash# vi /etc/httpd/conf/httpd.conf   (or it is /etc/apache/httpd.conf)

bash# /etc/init.d/apache restart
#apache should restart!!!
```

41.1 Testing Midgard PHP Server

To test the installation do this − Create a file in your document root directory. I usually call it info.php and in it put this single line:

```
<?phpinfo()?>
```

Then load it up in your browser: http://localhost/info.php

You should see a nice summary page showing all sorts of information about your setup. You probably shouldn't leave this file around on a production server, but for debugging and general info during development, it is very handy.

41.2 Security OpenSSL

You may also need to get the RSA package for to enable SSL encryption from ftp://ftp.deva.net/pub/sources/crypto/rsaref20.1996.tar.Z See also OpenSSL RPM package on Linux cdrom (http://www.openssl.org

If you do not want the SSL to be enabled (or if you face any problem), then download the source RPM of Apache−Midgard and edit the *.spec file and comment out SSL and rebuild the RPM.