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

Al Dev (Alavoor Vasudevan) *alavoor@yahoo.com* v13.0, 23 October 1999

This document is a "practical guide" to very quickly setup a SQL Database engine and front end tools on a 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 a Application Database Server or as a Web Database Server. PostgreSQL implements a subset of International ISO/ANSI SQL standards of years 1998,92,89. 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 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 a 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.

Contents

1 Introduction **5**

2 Other Formats of this Document **7**

3 Laws of Physics applies to Software! **9**

4 What is PostgreSQL ? **11**

5 Where to get it ? **12**

6 PostgreSQL Quick-Steps **13**

6.1 Install and Test . . . 13

6.2 Maximum RPM . . . 14

6.3 Testing PyGreSQL - Python interface . . . 14

6.4 Testing Perl - Perl interface . . . 15

6.5 Testing libpq, libpq++ interfaces . . . 15

6.6 Testing Java interfaces . . . 15

6.7 Testing ecpg interfaces . . . 16

6.8 Testing SQL examples - User defined types and functions . . . 16

6.9 Testing Tcl/Tk interfaces . . . 16

6.10 Testing ODBC interfaces . . . 16

6.11 Testing MPSQL Motif-worksheet interfaces . . . 16

6.12 Verification 16

7 PostgreSQL Supports Extremely Large Databases greater than 200 Gig 17

8 How can I trust PostgreSQL ? Regression Test Package builds customer confidence 17

9 GUI FrontEnd Tool for PostgreSQL (Graphical User Interface) 18

10 Interface Drivers for PostgreSQL 19

10.1 ODBC Drivers for PostgreSQL 19

10.2 UDBC Drivers for PostgreSQL 20

10.3 JDBC Drivers for PostgreSQL 20

10.4 Java for PostgreSQL 20

11 Perl Database Interface (DBI) Driver for PostgreSQL 20

11.1 Perl 5 interface for PostgreSQL 20

11.2 Perl Database Interface DBI 21

11.2.1 WHAT IS DBI ? 21

11.2.2 DBI driver for PostgreSQL DBD-Pg-0.89 22

11.2.3 Technical support for DBI 22

11.2.4 What is DBI, DBperl, Oraperl and *perl? 22

11.2.5 DBI specifications 23

11.2.6 Compilation problems or "It fails the test" 25

11.2.7 Is DBI supported under Windows 95 / NT platforms? 25

11.2.8 What's DBM? And why use DBI instead ? 26

11.2.9 Is < insert feature here > supported in DBI? 26

11.2.10 Is DBI any use for CGI programming? 26

11.2.11 How do I get faster connection times with DBD Oracle and CGI? 26

11.2.12 How do I get persistent connections with DBI and CGI? 27

11.2.13 "When I run a perl script from the command line, it works, but, when I run it under the httpd, it fails!" Why? 27

11.2.14 Can I do multi-threading with DBI? 27

11.2.15 How can I invoke stored procedures with DBI? 27

11.2.16 How can I get return values from stored procedures with DBI? 27

11.2.17 How can I create or drop a database with DBI? 28

11.2.18 How are NULL values handled by DBI? 28

11.2.19 What are these func methods all about? 28

11.2.20 Commercial Support and Training 28

12 PostgreSQL Management Tools **28**

12.1 PGACCESS - A GUI Tool for PostgreSQL Management 28

12.2 Windows Interactive Query Tool for PostgreSQL (WISQL or MPSQL) 30

12.3 Interactive Query Tool (ISQL) for PostgreSQL called PSQL 30

12.4 MPMGR - A Database Management Tool for PostgresSQL 30

13 Setting up multi-boxes PostgreSQL with just one monitor **31**

14 Applications and Tools for PostgreSQL **33**

14.1 PostgreSQL 4GL for web database applications - AppGEN Development System 33

14.2 WWW Web interface for PostgresSQL - DBENGINE 34

14.3 Apache Webserver Module for PostgreSQL - NeoSoft NeoWebScript 34

14.4 HEITML server side extension of HTML and a 4GL language for PostgreSQL 35

14.5 America On-line AOL Web server for PostgreSQL 36

14.6 Problem/Project Tracking System Application Tool for PostgreSQL 37

14.7 Convert dbase dbf files to PostgreSQL 37

15 Web Database Design/Implementation tool for PostgreSQL - EARP **37**

15.1 What is EARP ? 37

15.2 Implementation 37

15.3 What you need to run EARP 38

15.4 How does it work ? 38

15.5 Where to get EARP ? 38

16 PHP Hypertext Preprocessor - Server-side html-embedded scripting language for PostgreSQL **39**

16.1 Major Features 39

16.2 Credits 40

16.3 PHP 3 - Brief History 40

16.4 So, what can I do with PHP/FI ? 41

16.5 A simple example 41

16.6 CGI Redirection 42

16.6.1 Apache 1.0.x Notes 42

16.6.2 Netscape HTTPD 43

16.6.3 NCSA HTTPD 43

16.7 Running PHP/FI from the command line 43

17 Python Interface for PostgreSQL **43**

17.1 Where to get PyGres ? 44

17.2 Information and support 44

18 Gateway between PostgreSQL and the WWW - WDB-P95 44

18.1 About wdb-p95 44

18.2 Does the PostgreSQL server, pgperl, and httpd have to be on the same host? 45

19 "C", "C++", ESQL/C language Interfaces and Bitwise Operators for PostgreSQL 45

19.1 "C" interface 45

19.2 "C++" interface 45

19.3 ESQL/C 45

19.4 BitWise Operators for PostgreSQL 46

20 Japanese Kanji Code for PostgreSQL 46

21 PostgreSQL Port to Windows 95/Windows NT 46

22 Mailing Lists 47

22.1 Get a Free e-mail account 47

22.2 English Mailing List 47

22.3 Archive of Mailing List 48

22.4 Spanish Mailing List 48

23 Documentation and Reference Books 48

23.1 User Guides and Manuals 48

23.2 Online Documentation 48

23.3 Useful Reference Textbooks 49

23.4 ANSI/ISO SQL Specifications documents - SQL 1992, SQL 1998 50

23.5 Syntax of ANSI/ISO SQL 1992 50

23.6 Syntax of ANSI/ISO SQL 1998 50

23.7 SQL Tutorial for beginners 51

23.8 Temporal Extension to SQL92 51

23.9 Part 0 - Acquiring ISO/ANSI SQL Documents 52

23.10Part 1 - ISO/ANSI SQL Current Status 55

23.11Part 2 - ISO/ANSI SQL Foundation 57

23.12Part 3 - ISO/ANSI SQL Call Level Interface 58

23.13Part 4 - ISO/ANSI SQL Persistent Stored Modules 59

23.14Part 5 - ISO/ANSI SQL/Bindings 60

23.15Part 6 - ISO/ANSI SQL XA Interface Specialization (SQL/XA) 61

23.16Part 7 - ISO/ANSI SQL Temporal 61

23.16.1 INTRODUCTION 61
23.16.2 A CASE STUDY - STORING CURRENT INFORMATION 61
23.16.3 A CASE STUDY - STORING HISTORY INFORMATION 62
23.16.4 A CASE STUDY - PROJECTION 62
23.16.5 A CASE STUDY - JOIN 64
23.16.6 A CASE STUDY - AGGREGATES 65
23.16.7 SUMMARY 66
23.17 Part 8 - ISO/ANSI SQL MULTIMEDIA (SQL/MM) 67

24 Technical support for PostgreSQL **67**

25 Economic and Business Aspects **68**

26 List of Other Databases **68**

27 Internet World Wide Web Searching Tips **69**

28 Conclusion **69**

29 FAQ - Questions on PostgreSQL **70**

30 Copyright Notice **70**

A Appendix A - Syntax of ANSI/ISO SQL 1992 **70**

B Appendix B - SQL Tutorial for beginners **117**
B.1 Tutorial for PostgreSQL 117
B.2 Internet URL pointers 117

C Appendix C - Linux Quick Install Instructions **118**

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.

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. The most popular database systems are based on the International Standard Organisation (ISO) SQL specifications, which in turn is based on ANSI SQL (American) standards. The most 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.

There are more than 20 varieties of commercial/internet database systems which are being used in the world and many more will be coming in the near future. Without a standard like ANSI/ISO SQL, it would be very difficult for the customer to develop a application once and run on all the database systems. Customer wants to develop an application ONCE using ISO SQL, ODBC, JDBC and deploy on all varieties 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 the future ANSI SQL standards like SQL 1998 (SQL-3) and beyond will increasingly deal with Object databases and Object data types. 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 a 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 database community.

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, than 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 per 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. Customers will give first preference for the open systems software like PostgreSQL, Linux, etc...

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 out of 180 countries in the world so far have demonstrated the capability of designing and manufacturing computer chips/hardware. Design and manufacturing of computer chips is a advanced technology. It is a very complex process, capital intensive, requires large investments in plant and production machines which deal with 0.18 micron 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
Feature size(micron)	0.25	0.18	0.15	0.13	0.05	< 0.00001
Wafer size(mm)	200	300	300	300	450	Mol/Bio
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
Max power dissipation	70	90	110	130	175	600
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 a less difficult technology. Hence, manufacturing hardware/hard-goods is vital for national economy! Companies like Compaq, Dell, Sun Microsystems, HP, IBM who manufacture computers are major contributors to U.S economy today and in the future!!

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 create a Oracle database server system. But it would take him about 10 years (Oracle database server is about 10 man-years of work). One man-year is one person working full-time for one full year. If 10 people work for a year than it is 10 man-years spent.

Databases like Oracle, Informix, Sybase, IBM DB2 (Unix) are written using the "C" language and binaries are created by compiling the source code and than 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.

PostgreSQL is not just a free database but it is a good quality 'Internet Product'. Prediction is that demand for "Made in Internet" products will grow exponentially as it is capable of maintaining a high quality, low cost, extremely large user-base and developer-base. Those nations who do not use the 'Made in 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!

2 Other Formats of this Document

This document is published in 10 different formats namely - DVI, Postscript, Latex, 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/>

- Plain text format is in: <ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO>
- Translations to other languages like French, German, Spanish, Chinese, Japanese are in <ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO> Any help from you to translate to other languages is welcome.

The document is written using a tool called "SGML tool" which can be got from - <http://www.xs4all.nl/~cg/sgmltools/> 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)

This 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. 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.

3 Laws of Physics applies to Software!

In this chapter, it will be shown how science plays a 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. Objects include everything - for example PostgreSQL, time, mass, energy, planets, sun, moon, stars, galaxies, super-clusters, humans etc... are objects made by science. This chapter also shows how 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 than the closed source-code systems will eventually VANISH from this planet.

The paragraphs given below will show you - "how vast science is, how important it is for man and how it impacts software projects like PostgreSQL, Linux".

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. Various branches of sciences like physics, chemistry etc all merge into one grand region called Mathematics (which is also known as the Queen of all Sciences). Everything 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. It is the energy which made this world, human brains and many other things! And mass and energy are ONE and the **SAME** entity! The fact that mass and energy are same was unknown to people 100 years ago!

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. So STOP READING NOW ! The energy activity of neurons (brain cells) can be measured in the laboratory. For example, there are many instruments like 'Lie Detectors' and other medical instruments which can measure the energy activities of brain. 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.

As per science, it is impossible to build any system or theory (including a database software system) which will be 100 % perfect and bug free. It is like chasing a mirage, we will **NEVER** reach the goal of perfect system or theory. Detailed mathematical equations/discussions to prove that 'perfect system' (as well as 'imperfect system') is impossible, is a advanced topic and is beyond the scope of this document. And such a mathematical discussion deals with infinite number of dimensions (as well as primary dimensions) which are existing in nature. Unfortunately humans can see or feel only 4 dimensions but mathematics can easily explore many other dimensions. Other dimensions are **'infinitely smaller'** than the atoms and atoms themselves are very minute which human eyes cannot see them! Mathematics is very powerful as it can analyze and explain the birth/death of our universe. Our universe is almost **zero size** if you look from other universe and vice versa. That is, our universe is not visible (does not exist) for persons in other universe! And vice versa!! Theoretically, you can exit out of our universe and travel vast distances (billions of light years) in zero time and re-enter universe at a different point of space-time! Distance between our universe and other universes is actually zero!

Even though there are infinite number of dimensions, they all can be derived/generated from a small number of PRIMARY dimensions. That is, infinite number of dimensions can combine and collapse into primary dimensions. Primary dimensions simply absorb other dimensions without themselves getting destroyed. Mathematicians use these primary dimensions to understand the birth and death of universes. The universe where you are currently living started with a **BIG BANG** billions of years ago (roughly 20 billion years ago) which was caused by the interactions of atomic particles of other dimensions. Just before the big bang there was a tiny point where length, breadth, height and time was ZERO (that is, our universe was NOT THERE!!) and other universes and primary dimensions were existing. Time itself was NOT there and atoms, stars, planets, galaxies were NOT there! The atoms inside your body were NOT there!! So many things happened even BEFORE the time was born!

Big bang and hence birth of our universe was caused by few atoms of primary dimensions. **SOMEONE** (something?) caused the dashing of few tiny atoms of other dimensions to create our universe, and new dimensions time, length, breadth, height was born! We see someone's hand in this process. That process is **not very well understood** by man. Man is trying to generate another universe in the lab by simulating the big bang event (Huge accelerator is under construction in Europe, another construction in Dallas, USA was stopped by US congress due to budget cuts). There are atoms of other dimensions just as we have atoms in our universe. Theoretically, you can create/generate infinite number of universes! This process is reversible, that is our universe can completely close down and vanish into few atoms of other dimensions! It is similar in anology as to how YOU were born from two tiny cells which DASHED against each other to create one single cell. That single tiny cell divided and multiplied into 6 trillion cells to become a 6 foot tall human (that is you!). There are some similarities between humans and universes, universes are born and later die, very much similar to humans.

Since PERFECT state (as well as IMPERFECT state) is impossible, universes like ours are born and later die down in a cyclic process. BUT there can NO PERFECT death, only transformation is permitted by science! Our universe is currently expanding at a tremendous rate, it is not static but is very dynamic. This universe will keep expanding untill something interferes to collapse this universe by applying pressures with external dimensions to aborb and annihilate our universe!! Two possibilities are - removing the mass from our universe via black holes (a slow process), or injecting more mass into our universe via white holes (a rapid big crunch process). There can be millions of universes in existence but there can be very few universes which are built from primary dimensions. You are not only made up of atoms of this universe but also by atoms of other dimensions! In black holes the atoms and particles from our universe get sucked into and are completely transformed and converted to particles of other dimensions! In the center of black hole time is zero and length, breadth, height is zero! And black hole is EXIT/ENTRY door to and from other universes. Entry/Exit doors to other universes can open up anywhere, even inside your body!

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.

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. Human beings are creatures of finite energy and resources, frequently guilty of errors ranging from the trivial to the profound. Because of our propensity for error varies broadly with particular skills, particular experience and the vagaries of the moment, the discovery and elimination of software bugs produced by ordinary human error can be greatly facilitated by bringing a great many minds to bear on the problem. In addition, more individuals represent extra person hours of code and its asocciated function.

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 big companies like Microsoft, IBM cannot overpower and overrule the laws of Physics but will eventually **SURRENDER UNTO** laws of science!

Today, there are too many SQL RDBMS databases in the world aiming at only one specification ANSI/ISO SQL. **Man must not waste his time creating too many SIMILAR/IDENTICAL software packages** and there are many other topics in science which need good attention.

Conclusion is - because of laws of science, 'open source code' system like PostgreSQL, Linux will be always much better than 'closed source code' system and it is possible to prove this statement scientifically.

4 What is PostgreSQL ?

PostgreSQL Version 6.5.2 is a free database, complete source code is given to you and is a Object-Relational Database System near compliant (getting closer) with ANSI SQL1998,92,89 and runs on diverse hardware platforms and Operating systems. 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

```
cd /usr/src/postgresql6.5.2
man patch
patch -p0 < patchfile
make clean
make
```

The patch files are located in

- PostgreSQL patches : `<ftp://ftp.postgresql.org/pub/patches>`

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 will also guide, steer, control, monitor and dictate the future of ANSI/ISO SQL. That is, the implementation and ideas first take place in PostgreSQL and than later be incorporated into the ANSI/ISO SQL.

Informix Universal server (released 1997) is based on earlier version of PostgreSQL because Informix bought Illustra Inc. and integrated with Informix. Illustra database was completely 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. Many others have contributed to the porting, testing, debugging and enhancement of the code. The original Postgres code, from which PostgreSQL is derived, was the effort of many graduate students, undergraduate students, and staff programmers working under the direction of Professor Michael Stonebraker at the University of California, Berkeley.

The original name of the software at Berkeley was Postgres. When SQL functionality was added in 1995, its name was changed to Postgres95. The name was changed at the end of 1996 to PostgreSQL.

Millions of PostgreSQL is installed as Database servers, Web database servers and Application data servers. It is much more advanced and is a object oriented relational database (ORDBMS).

PostgreSQL can store more data types than traditional datatypes like integer, characters, etc. - you get to create user-defined types, functions, inheritance etc. 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 95/NT is underway.

- Title: PostgreSQL SQL RDBMS Database (Object Relational Database Management System)
- Current Version: 6.5.2
- Age: PostgreSQL is 14 years old. Developed since 1985
- Authors: Developed by millions/universities/companies on internet for the past 14 YEARS

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

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:

- You can run PostgreSQL without compiling the source. Get binaries for Intel-Linux from <ftp://www.redhat.com/pub/contrib/i386/> file is postgresql-6.5.2.i386.rpm. This is in the redhat package 'rpm' format and it contains both source and binaries for PostgreSQL.
- Binaries site for Solaris, HPUX, AIX, IRIX, Linux : <ftp://ftp.postgresql.org/pub/bindist> If you compile on any platform please upload to site, so that it will be useful for others.
- ftp site : Get binaries for Intel-Linux from <ftp://ftp.redhat.com/pub/contrib/i386/> file is postgresql-6.5.2.i386.rpm. This is in the redhat package 'rpm' format and it contains both source and binaries for PostgreSQL.

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-Steps

This chapter will help you to install and run the database very quickly in less than 10 minutes.

6.1 Install and Test

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

```
# rpm -qpl postgre*.rpm (to see list of files. For docs 'man rpm')
# 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 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 )
# chkconfig --add postgresql  (to start pg during booting, see 'man
chkconfig')
# cp /usr/lib/pgsql/python/_pg.so /usr/lib/python1.5/lib-dynload (for
python)
# /etc/rc.d/init.d/postgresql start  (to start up postgres)
# su - postgres
bash$ createdb mydatabase  (this will create a database by name
'mydatabase')
                 (for doc 'man createdb')
bash$ psql mydatabase   (for doc 'man psql')
..... in psql press up/down arrow keys for history line editing or \s
bash$ export DISPLAY=<hostname>:0.0; pgaccess mydatabase; (see 'man
pgaccess')
```

Now you can start banging away at SQL commands at pgaccess or psql !!

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

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

More details about PostgreSQL is at <http://www.postgresql.org>

6.2 Maximum RPM

Download the 'Maximum RPM' textbook from <http://www.RPM.org> the filename is maximum-rpm.ps.gz And read it on linux using gv command -

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

6.3 Testing PyGreSQL - Python interface

```
bash$ cd /usr/lib/pgsql/python
bash$ createdb thilo
bash$ psql thilo
thilo=> create table test (aa char(30), bb char(30) );
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$
..... Yow! Seems to work - now install it properly
bash$ su - root
# cp /usr/lib/pgsql/python/_pg.so /usr/lib/python1.5/lib-dynload
```

6.4 Testing Perl - Perl interface

```
bash$ cd /usr/doc/postgresql-6.5.2/examples/perl5
```

Note: 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/5.004/i386-linux-thread ./example.newstyle
```

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

Read the example files for using perl interface

6.5 Testing libpq, libpq++ interfaces

```
bash$ cd /usr/doc/postgresql-6.5.2/examples/libpq++
bash$ su root   --> to change ownership of examples
# chown -R postgres /usr/doc/postgresql-6.5.2/examples
# exit
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$
```

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

6.6 Testing Java interfaces

For this you MUST install the jdk-*glibc*.rpm package (Java RPM packages)

```
bash$ cd /usr/doc/postgresql-6.5.2/examples/jdbc
bash$ echo $CLASSPATH
 --> Should show  CLASSPATH=/usr/lib/jdk-1.1.6/lib/classes.zip
bash$ export CLASSPATH=$CLASSPATH:.:/usr/lib/pgsql/postgresql.jar
Edit all psql.java file  and comment out the 'package' line.
bash$ 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$
```

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

6.7 Testing ecpg interfaces

```
bash$ cd /usr/doc/postgresql-6.5.2/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
```

.... Wow!! You ran Embedded "C"-SQL to PostgreSQL database!!

6.8 Testing SQL examples - User defined types and functions

```
bash$ cd /usr/doc/postgresql-6.5.2/examples/sql
Under-development..
```

6.9 Testing Tcl/Tk interfaces

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

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

6.10 Testing ODBC interfaces

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

6.11 Testing MPSQL Motif-worksheet interfaces

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

6.12 Verification

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

```
# 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-6.5.2/src/test/regress
bash$ more README
bash$ make clean; make all runtest
bash$ more regress.out
```

7 PostgreSQL Supports Extremely Large Databases greater than 200 Gig

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.

For extremely large databases, it is strongly advised to use 64-bit machines like Digital Alpha cpu, Sun Ultrasparc 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..

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

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 package than 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 package.

9 GUI FrontEnd Tool for PostgreSQL (Graphical User Interface)

Web browser will be the most popular GUI front-end in the future. A major portion of code should be written in Web server scripting (and compiling) language PHP/Zend 16 () and with little bit of JavaScript on web-client side. It is recommended that you migrate your legacy Windows 95/NT applications to PHP/Zend.

The best tools are

- PHP and Zend (script and compiler) 16 ()
- 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 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. PHP Web server scripting with Javascrpt client scripting.
2. Perl scripting language using Perl-Qt or Perl-Tk 11 ()
3. Omnipresent language C++ with QtEZ, QT, Lesstiff or Motif.
4. Java but its programs run very slow.

There are other tools available - PostgreSQL has Tcl/Tk interface library in the distribution called 'pgTcl'. There is a 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>
- FreeBuilder <ftp://ftp.redhat.com/pub/contrib/i386/free*.rpm>
- SpecTCL <ftp://ftp.redhat.com/pub/contrib/i386/spec*.rpm>
- Kanchenjunga Java RAD Tool for PostgreSQL <http://www.man.ac.uk/~whaley/kj/kanch.html>
- Applixware Tool <http://www.redhat.com>
- XWPE X Windows Programming Environment <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.

10 Interface Drivers for PostgreSQL

10.1 ODBC Drivers for PostgreSQL

ODBC stands for 'Open DataBase Connectivity' 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* <http://www.insightdist.com/psqlodbc> is already included in the distribution. See main web site <http://www.postgresql.org>. It is included on the PostgreSQL CDROM.
- <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.

There is a project called the FreeODBC Pack Package . There's no PostgreSQL Version there, maybe you can help.

- <http://www.ids.net/~bjepson/freeODBC/> This is a free of cost version of ODBC.

10.2 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.

10.3 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.

- <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>

10.4 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>

11 Perl Database Interface (DBI) Driver for PostgreSQL

11.1 Perl 5 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 interface for PostgreSQL is included in the distribution of PostgreSQL. Check in src/pgsql_perl5 directory.

- Pgsql_perl5 contact Email: *E.Mergl@bawue.de*
- Another source from - <ftp://ftp.kciLink.com/pub/PostgresPerl-1.3.tar.gz>
- 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/>
- 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
- 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.*

11.2 Perl Database Interface DBI

11.2.1 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 information for this DBI section is obtained from 'DBI FAQ' doc whose author is Alligator Descartes and reproduced here with his permission.

- Alligator Descartes Hermetica is at *descarte@hermetica.com*

11.2.2 DBI driver for PostgreSQL DBD-Pg-0.89

Get DBD-Pg-0.89.tar.gz from below

- DBD-Pg-0.89 : <http://www.perl.com/CPAN/modules/by-module/DBD/>
- Comprehensive Perl Archive Network CPAN <http://www.perl.com/CPAN>
- DBI drivers list and DBI module pages <http://www.hermetica.com/technologia/perl/DBI>
- DBI information is at <http://www.fugue.com/dbi/>
- Primary ftp site <ftp://ftp.demon.co.uk/pub/perl/db>
- Miscellaneous DBI link <http://www-ccs.cs.umass.edu/db.html>
- Miscellaneous DBI link <http://www.odmg.org/odmg93/updates_dbarry.html>
- Miscellaneous DBI link <http://www.jcc.com/sql_stnd.html>
- PostgreSQL database <http://www.postgresql.org>

REQUIREMENTS:

- - build, test and install Perl 5 (at least 5.002)
- - build, test and install the DBI module (at least 0.89)
- - build, test and install PostgreSQL (at least 6.2)

11.2.3 Technical support for DBI

Please send comments and bug-reports to

- *E.Mergl@bawue.de*

Please 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.

11.2.4 What is DBI, DBperl, Oraperl and *perl?

To quote Tim Bunce, the architect and author of DBI:

"DBI is a database access Application Programming Interface (API) for the Perl Language. The DBI API Specification defines a set of functions, variables and conventions that provide a consistent database interface independent of the actual database being used."

In simple language, the DBI interface allows users to access multiple database types transparently. So, if you connecting to an Oracle, Informix, mSQL, Sybase or whatever database, you don't need to know the underlying mechanics of the 3GL layer. The API defined by DBI will work on all these database types.

A similar benefit is gained by the ability to connect to two different databases of different vendor within the one perl script, ie, I want to read data from an Oracle database and insert it back into an Informix database all within one program. The DBI layer allows you to do this simply and powerfully.

DBperl is the old name for the interface specification. It's usually now used to denote perl4 modules on database interfacing, such as, oraperl, isqlperl, ingperl and so on. These interfaces didn't have a standard API and are generally not supported.

Here's a list of DBperl modules, their corresponding DBI counterparts and support information. DBI driver queries should be directed to the dbi-users mailing list.

```
Module Name Database Required   Author          DBI
----------- -----------------   ------          ---
Sybperl     Sybase              Michael Peppler DBD::Sybase
                                <mpeppler@datamig.com>
                                http://www.mbay.net/~mpeppler
Oraperl     Oracle 6 & 7        Kevin Stock     DBD::Oracle
                                <dbi-users@fugue.com>
Ingperl     Ingres              Tim Bunce &     DBD::Ingres
                                Ted Lemon
                                <dbi-users@fugue.com>
Interperl   Interbase           Buzz Moschetti  DBD::Interbase
                                <buzz@bear.com>
Uniperl     Unify 5.0           Rick Wargo      None
                                <rickers@coe.drexel.edu>
Pgperl      Postgres            Igor Metz       DBD::Pg
                                <metz@iam.unibe.ch>
Btreeperl   NDBM                John Conover    SDBM?
                                <john@johncon.com>
Ctreeperl   C-Tree              John Conover    None
                                <john@johncon.com>
Cisamperl   Informix C-ISAM     Mathias Koerber None
                                <mathias@unicorn.swi.com.sg>
Duaperl     X.500 Directory     Eric Douglas    None
            User Agent
```

However, some DBI modules have DBperl emulation layers, so, DBD::Oracle comes with an Oraperl emulation layer, which allows you to run legacy oraperl scripts without modification. The emulation layer translates the oraperl API calls into DBI calls and executes them through the DBI switch.

Here's a table of emulation layer information:

```
Module                  Emulation Layer     Status
------                  ---------------     ------
DBD::Oracle     Oraperl             Complete
DBD::Informix   Isqlperl            Under development
DBD::Sybase     Sybperl             Working? ( Needs verification )
DBD::mSQL       Msqlperl            Experimentally released with
                                    DBD::mSQL-0.61
```

The Msqlperl emulation is a special case. Msqlperl is a perl5 driver for mSQL databases, but does not conform to the DBI Specification. It's use is being deprecated in favour of DBD::mSQL. Msqlperl may be downloaded from CPAN via:

- <http://www.perl.com/cgi-bin/cpan_mod?module=Msqlperl>

11.2.5 DBI specifications

There are a few information sources on DBI.

- DBI Specification <http://www.hermetica.com/technologia/perl/DBI/doc/dbispec>

There are two specifications available at this link, the new DBI Draft Specification which is a rapidly changing document as the development team drive towards a stable interface, and the old historical DBperl Specification out of which the current DBI interface evolved.

The later document should be regarded as being of historical interest only and should not serve as a programming manual, or authoratative in any sense. However, it is still a very useful reference source.

POD documentation PODs are chunks of documentation usually embedded within perl programs that document the code "in place", providing a 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 DBI Specification The POD for the DBI Specification can be read with the command

perldoc DBI

Oraperl 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

This will produce an updated copy of the original oraperl man page written by Kevin Stock for perl4. The oraperl API is fully listed and described there.

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

perldoc DBD::mSQL

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

perldoc DBI::FAQ

This may be more convenient to people not permanently, or conveniently, connected to the Internet.

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.

Rambles, Tidbits and Observations : There are a series of occasional rambles from various people on the DBI mailing lists.

- <http://www.hermetica.com/technologia/perl/DBI/tidbits>

"DBI – The perl5 Database Interface" This is an article written by Alligator Descartes and Tim Bunce on the structure of DBI. It was published in issue 5 of "The Perl Journal". It's extremely good. Go buy the magazine. In fact, buy all of them. "The Perl Journal"s WWW site is:

- <http://www.tpj.com>

"DBperl" This article, published in the November 1996 edition of "Dr. Dobbs Journal" concerned DBperl.

"The Perl5 Database Interface" This item is a book to be written by Alligator Descartes and published by O'Reilly and Associates.

Mailing Lists There are three mailing lists for DBI run by Ted Lemon. These can all be subscribed to and unsubscribed from via the World Wide Web at

- Mailing lists <http://www.fugue.com/dbi>

The lists that users may participate in are:

dbi-announce This mailing list is for announcements only. If you cannot successfully use the form on the above WWW page, please subscribe to the list in the following manner:

- Email: *dbi-announce-request@fugue.com* with a message body of 'subscribe'

dbi-dev This mailing list is intended for the use of developers discussing ideas and concepts for the DBI interface, API and driver mechanics. Only any use for developers, or interested parties. If you cannot successfully use the form on the above WWW page, please subscribe to the list in the following manner:

- Email: *dbi-dev-request@fugue.com* with a message body of 'subscribe'

dbi-users This mailing list is a general discussion list used for bug reporting, problem discussion and general enquiries. If you cannot successfully use the form on the above WWW page, please subscribe to the list in the following manner:

- Email: *dbi-users-request@fugue.com* with a message body of 'subscribe'

Mailing List Archives

- 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>

11.2.6 Compilation problems or "It fails the test"

If you have a core dump, try the Devel::CoreStack module for generating a stack trace from the core dump. Devel::CoreStack can be found on CPAN at:

- <http://www.perl.com/cgi-bin/cpan_mod?module=Devel::CoreStack>

Email the dbi-users Mailing List stack trace, module versions, perl version, test cases, operating system versions and any other pertinent information. The more information you send, the quicker developers can track problems down. If you send us nothing, expect nothing back.

11.2.7 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 an experimental DBI 'emulation layer' for the Win32::ODBC module. It's called DBI::W32ODBC. You will need the Win32::ODBC module.

- Win32 DBI <http://www.hermetica.com/technologia/perl/DBI/win32>
- Win32 ODBC <http://www.roth.net>

11.2.8 What's DBM? And why use DBI instead ?

UNIX was originally blessed with simple file-based "databases", namely the dbm system. dbm lets you store data in files, and retrieve that data quickly. However, it also has serious drawbacks.

File Locking

The dbm systems did not allow particularly robust file locking capabilities, nor any capability for correcting problems arising through simultaneous writes [to the database].

Arbitrary Data Structures

The dbm systems only allows a single fixed data structure: key-value pairs. That value could be a complex object, such as a [C] struct, but the key had to be unique. This was a large limitation on the usefulness of dbm systems.

However, dbm systems still provide a useful function for users with simple datasets and limited resources, since they are fast, robust and extremely well-tested. Perl modules to access dbm systems have now been integrated into the core Perl distribution via the AnyDBM_File module.

To sum up, DBM is a perfectly satisfactory solution for essentially read-only databases, or small and simple datasets. However, for more powerful and scaleable datasets, not to mention robust transactional locking, users are recommended to use DBI.

11.2.9 Is < insert feature here > supported in DBI?

Given that we're making the assumption that the feature you have requested is a non-standard database-specific feature, then the answer will be no.

DBI reflects a generic API that will work for most databases, and has no database-specific functionality.

However, driver authors may, if they so desire, include hooks to database-specific functionality through the func method defined in the DBI API. Script developers should note that use of functionality provided via the func methods is unlikely to be portable across databases.

11.2.10 Is DBI any use for CGI programming?

In a word, yes! DBI is hugely useful for CGI programming! In fact, CGI programming is one of two top uses for DBI.

DBI confers the ability to CGI programmers to power WWW-fronted databases to their users, which provides users with vast quantities of ordered data to play with. DBI also provides the possibility that, if a site is receiving far too much traffic than their database server can cope with, they can upgrade the database server behind the scenes with no alterations to the CGI scripts.

11.2.11 How do I get faster connection times with DBD Oracle and CGI?

The Apache httpd maintains a pool of httpd children to service client requests. Using the Apache mod_perl module by Doug MacEachern, the perl interpreter is embedded with the httpd children. The CGI, DBI, and your other favorite modules can be loaded at the startup of each child. These modules will not be reloaded unless changed on disk. For more information on Apache, see the Apache Project's WWW site:

- Apache Project WWW site `<http://www.apache.org>`
- Mod_perl module `<http://www.perl.com/cgi-bin/cpan_mod?module=mod_perl>`

11.2.12 How do I get persistent connections with DBI and CGI?

Using Edmund Mergl's Apache::DBI module, database logins are stored in a hash with each of these httpd child. If your application is based on a single database user, this connection can be started with each child. Currently, database connections cannot be shared between httpd children. Apache::DBI can be downloaded from CPAN via:

- <http://www.perl.com/cgi-bin/cpan_mod?module=Apache::DBI>

11.2.13 "When I run a perl script from the command line, it works, but, when I run it under the httpd, it fails!" Why?

Basically, a good chance this is occurring is due to the fact that the user that you ran it from the command line as has a correctly configured set of environment variables, in the case of DBD::Oracle, variables like $ORACLE_HOME, $ORACLE_SID or TWO_TASK. The httpd process usually runs under the user id of nobody, which implies there is no configured environment. Any scripts attempting to execute in this situation will correctly fail. To solve this problem, set the environment for your database in a BEGIN ( ) block at the top of your script. This will solve the problem. Similarly, you should check your httpd error logfile for any clues, as well as the "Idiot's Guide To Solving Perl / CGI Problems" and "Perl CGI Programming FAQ" for further information. It is unlikely the problem is DBI-related. Read BOTH these documents carefully!

- Idiot's Guide to Solving Perl / CGI problems <http://www.perl.com/perl/faq/index.html>

11.2.14 Can I do multi-threading with DBI?

As of the current date, no. Perl does not support multi-threading. However, multi-threading is expected to become part of the perl core distribution as of version 5.005, which implies that DBI may support multi-threading fairly soon afterwards. For some OCI example code for Oracle that has multi-threaded SELECT statements, see:

- <http://www.hermetica.com/technologia/oracle/oci/orathreads.tar.gz>

11.2.15 How can I invoke stored procedures with DBI?

Assuming that you have created a stored procedure within the target database, eg, an Oracle database, you can use $dbh->do to immediately execute the procedure. For example,

$dbh->do("BEGIN someProcedure END");

11.2.16 How can I get return values from stored procedures with DBI?

Remember to perform error checking, though!

```
$sth = $dbh->prepare( "BEGIN foo(:1, :2, :3); END;" );
$sth->bind_param(1, $a);
$sth->bind_param_inout(2, \$path, 2000);
$sth->bind_param_inout(3, \$success, 2000);
$sth->execute;
```

11.2.17 How can I create or drop a database with DBI?

Database creation and deletion are concepts that are entirely too abstract to be adequately supported by DBI. For example, Oracle does not support the concept of dropping a database at all! Also, in Oracle, the database server essentially is the database, whereas in mSQL, the server process runs happily without any databases created in it. The problem is too disparate to attack. Some drivers, therefore, support database creation and deletion through the private func methods. You should check the documentation for the drivers you are using to see if they support this mechanism.

11.2.18 How are NULL values handled by DBI?

NULL values in DBI are specified to be treated as the value undef. NULLs can be inserted into databases as NULL, for example:

```
$rv = $dbh->do( "INSERT INTO table VALUES( NULL )" );
```

but when queried back, the NULLs should be tested against undef. This is standard across all drivers.

11.2.19 What are these func methods all about?

The func method is defined within DBI as being an entry point for database-specific functionality, eg, the ability to create or drop databases. Invoking these driver-specific methods is simple, for example, to invoke a createDatabase method that has one argument, we would write:

```
$rv = $dbh->func( 'argument', 'createDatabase' );
```

Software developers should note that the func methods are non-portable between databases.

11.2.20 Commercial Support and Training

The Perl5 Database Interface is FREE software. IT COMES WITHOUT WARRANTY OF ANY KIND. However, some organizations are providing either technical support or training programs on DBI.

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 :

- <http://www.perl.co.uk/tpc>

12 PostgreSQL Management Tools

12.1 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, please feel free to e-mail to : *teo@flex.ro*

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

If you have any comment, suggestion for improvements e-mail to :

- *teo@flex.ro*

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.

12.2 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*

12.3 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.

```
Type \h to see help of commands.
```

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

12.4 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*

13 Setting up multi-boxes PostgreSQL with just one monitor

You can stack up the 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 you eliminate monitor, keyboard and the mouse (saving anywhere from 100 to 500 US dollars per set) and also avoid lot of clutter.

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 (saves you about $ 150). You do not need a color monitor for the database server, as you can do remote administration from color PC workstation. 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 client).

Using this idea you can control many cpu boxes by just one monitor and one keyboard!

Below is the **extract from networktechnic Inc** for the "Server Switches"

Order now call 800-742-8324 (toll free in USA)

To receive our catalog please Email your address to: sales@networktechinc.com

CONTROL MULTIPLE PC'S with one keyboard monitor & mouse These electronic switches will allow one keyboard, monitor & mouse to control up to 64 PC's. Embedded microcomputers simulate the presence of keyboard, monitor & mouse to all attached PC's 100% of the time.

Features and Applications

- Keyboard, front panel or remote control
- 9 pin D Serial or 6 pin miniDIN mouse support
- 5 pin DIN or 6 pin miniDIN keyboard support
- 1600x1200 video resolution with no degradation
- Uses standard cables
- Desktop or rackmount

Specifications - Keyboard

- All connectors are female
- 5 pin DIN or 6 pin miniDIN
- Will hard or soft boot all PC's 100% of the time

Controls - Buttons on Front

- Touch a button and be connected to that PC
- Hold any button in for more than 0.5 second and go into SCAN, BROADCAST or COMMAND mode
- LEDs on front indicate mode of operation Keyboard
- Type CTRL+* and go into COMMAND mode
- SCAN, BROADCAST or COMMAND are available

Mouse

* 9 pin D serial

- o NTI switch emulates Microsoft serial mouse to all PC's
- o 9 pin D male for mouse
- o 9 pin D female for PC's

* 6 pin miniDIN

- o NTI switch emulates IBM PS/2 style mouse to all PC's
- o All connectors are female

Wired remote

- Optional–must be purchased

- Operates same as "Buttons on Front"

Monitor-VGA

- Bandwidth is 150 MHz
- 1600X1200 resolution with no degradation
- All connectors are female

14 Applications and Tools for PostgreSQL

14.1 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, a utility fo 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*

14.2 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/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 3 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 mata 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.

License : dbengine is free software under the same terms as Perl. Read its licence if you aren't sure what you can or can't do. The bottom line is that this is a kinder and gentler version of the GNU licence – one that doesn't infect your work if you care to borrow from dbengine or package up pieces of it as part of a commercial product.

14.3 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

14.4 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.

14.5 America On-line AOL Web server for PostgreSQL

The no-cost commercial webserver, AOLserver version 2.3 supports database connections to PostgreSQL 6.2.1 and higher. for more info see

- AOL Web Server `<http://www.aolserver.com>`

14.6 Problem/Project Tracking System Application Tool for PostgreSQL

This is at

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

14.7 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/incoming/dbf2pg-3.0.tar.gz>

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>

15 Web Database Design/Implementation tool for PostgreSQL - EARP

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

15.1 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.

15.2 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.

15.3 What you need to run EARP

EARP will likely run on a variety of platforms with little or no porting. The known working platforms consist of the following:

- Solaris 2.5
- Linux 1.2.13+
- GNU C++
- PostgreSQL (Version 1.01 / 1.02)
- netsite server
- NCSA httpd
- GNU C++
- PostgreSQL (Version 1.01 / 1.02)
- NCSA httpd
- Apache httpd

The current (1.3) release of Earp was designed on top of the libpq release that came with PostgreSQL v1.01/1.02. If you are using a more recent version of Postgres, expect that the program will require some porting to work correctly. In the development version (Earp 2.0), libpq support is being incorporated as a module.

15.4 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.

15.5 Where to get EARP ?

EARP is available via anonymous ftp from

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

16 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 3) 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 3 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 3 is not browser-dependant. JavaScript is a client-side html-embedded language while PHP 3 is a server-side language. PHP 3 is similar in concept to Netscape's LiveWire Pro product. If you have the money, you run Netscape's Commerce Server and you run one of the supported operating systems, you should probably have a look at LiveWire Pro. If you like free fast-moving software that comes with full source code you will probably like PHP 3.

16.1 Major Features

Standard CGI, FastCGI and Apache module Support As a standard CGI program, PHP 3 can be installed on any Unix machine running any Unix web server. With support for the new FastCGI standard, PHP 3 can take advantage of the speed improvements gained through this mechanism. As an Apache module, PHP 3 becomes an extremely powerful and lightning fast alternative to CGI programmimg.

- Access Logging With the access logging capabilities of PHP 3, 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.

- Postgres Support Postgres is an advanced free RDBMS. PHP 3 supports embedding Postgres95 and 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 3 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 3 can be used to create customized HTTP-based authentication mechanisms for the Apache web server.
- Variables, Arrays, Associative Arrays PHP 3 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 3 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 3 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 3 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 3 supports a unique "Safe Mode" which makes it safe to have multiple users run PHP scripts on the same server.
- It's Free! One final essential feature. The package is completely free. It is licensed under the GPL which allows you to use the software for any purpose, commercial or otherwise. See the GNU Public License document for complete details.

16.2 Credits

* Large parts of this code were developed at and for the University of Toronto. Many thanks to Lee Oattes of the Network Development Department at the university for constant constructive criticism.

* The PostgreSQL support code was written by Adam Sussman

- *asussman@vidya.com*

* Countless others have helped test and debug the package.

16.3 PHP 3 - Brief History

PHP began life as a simple little cgi wrapper written in Perl. It was never intended to go beyond own private use. 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 has now evolved to the point where it is a simple programming language embedded inside HTML files. The original acronym, PHP, has stuck. It isn't really appropriate any longer. PHP/FI is used more for entire web sites today than for small Personal Home Page setups. By whatever name, it 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.

Throughout this documentation any references to PHP, FI or PHP/FI all refer to the same thing. The difference between PHP and FI is only a conceptual one. Both are built from the same source distribution. Now PHP/FI is renamed as PHP 3.

16.4 So, what can I do with PHP/FI ?

The first thing you will notice if you run a page through PHP/FI 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/FI can do for you. It serves another very important role as a form interpreter cgi, hence the FI part of the 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/FI makes it extremely easy to take form data and do things with it.

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

You can also use PHP/FI 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/FI 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/FI'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/FI has support for the PostgreSQL database package. It supports embedded SQL queries in your .HTML files. See the section on PostgreSQL Support for more information.

PHP/FI also has support for the mysql database package. It supports embedded SQL queries in your .HTML files. See the section on mysql Support for more information.

16.6 CGI Redirection

16.6.1 Apache 1.0.x Notes

A good way to run PHP/FI 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/FI. 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/FI.

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/fi 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.

16.6.2 Netscape HTTPD

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

16.6.3 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/FI file archives.

16.7 Running PHP/FI from the command line

If you build the CGI version of PHP/FI, 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/FI 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.

17 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*

17.1 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.

17.2 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

18 Gateway between PostgreSQL and the WWW - WDB-P95

18.1 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.

18.2 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)

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

19.1 "C" interface

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

19.2 "C++" interface

It is included in distribution and is called 'libpq++'.

19.3 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*

19.4 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.

20 Japanese Kanji Code for PostgreSQL

It is at the following site

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

21 PostgreSQL Port to Windows 95/Windows NT

Port to Windows 95/Windows NT is underway. Porting is being done using gcc, gmake for Win NT/95. To compile source code on win32 gnu-win32 program is used. GNU gcc is available for win32. Check this site -

- <http://www.cygnus.com/misc/gnu-win32>

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

Porting can also be done using the following "Unix-Emulator on NT" tool from

- <http://www.softway.com>

22 Mailing Lists

22.1 Get a Free e-mail account

Check out the free deals offered - 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.

22.2 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. See also the section 'Get a Free e-mail account' above. 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

22.3 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

22.4 Spanish Mailing List

Now there is an "unofficial" list of postgreSQL in Spanish. See also the section 'Free Account to Organise your PostgreSQL e-mails' above. 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

23 Documentation and Reference Books

23.1 User Guides and Manuals

The following are included in the PostgreSQL distribution in the postscript, HTML formats and unix man-pages. If you have access to internet, you can find the documents listed below at <http://www.postgresql.org/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.

23.2 Online Documentation

- Listing and description of default data types and operators

  ```
  Is a a part of PSQL command in the release 6.5.2.
  ```

- Listing of supported SQL keywords

  ```
  There is a script in the /tools directory to do that
  ```

- 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.
```

- Tutorial for PostgreSQL.

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

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

23.3 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.

23.4 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.

23.5 Syntax of ANSI/ISO SQL 1992

See Appendix A of this document A ()

23.6 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 23.9 ()

23.7 SQL Tutorial for beginners

See Appendix B of this document B ()

23.8 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).

23.9 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 a 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".

23.10 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 a 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 Testing (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."

23.11 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.

23.12 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

23.13 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 - A 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.

23.14 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.

23.15 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.

23.16 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.

23.16.1 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.

23.16.2 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'
```

23.16.3 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.

23.16.4 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
```

23.16.5 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
```

23.16.6 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
```

23.16.7 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.
```

23.17 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

24 Technical support for PostgreSQL

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

- *pgsql-questions@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.

In the near future, PostgreSQL organisation will be selling technical support to large/small 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.

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

- Work Group Solutions `<http://www.wgs.com>`
- Anderson Consulting `<http://www.ac.com>`

25 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. 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 tax 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) than 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 2,00,000 lines of "C", "C++" code. If cost of each line of "C" code is $ 2 than the total cost of PostgreSQL as of today is $ 4,00,000 (four hundred thousand 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.

26 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/>

27 Internet World Wide Web Searching Tips

Internet is very vast and it has vast number of software and has a 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 a 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.

28 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.

29 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>

30 Copyright Notice

Copyright (c) 1997 Al Dev (Alavoor Vasudevan). All rights reserved.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. In no event shall the author of this document be liable for any damages whatsoever (including without limitation, special, incidental, consequential, or direct/indirect damages for personal injury, loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of this document.

Author offers no warranties or guarantees on fitness, usability, merchantability of this document. Brands, companies and product names mentioned in this document are trademarks or registered trademarks of their respective holders. Please refer to individual copyright notices of brands, companies and products mentioned in this document. It is your responsibility to read and understand the copyright notices of the organisations/companies/products/authors mentioned in this document before using their respective information.

A 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>
```

B Appendix B - SQL Tutorial for beginners

B.1 Tutorial for PostgreSQL

SQL tutorial is also distributed with PostgreSQL. The SQL tutorial scripts is in the directory src/tutorial

B.2 Internet URL pointers

The SQL tutorial for beginners can be found at

- <http://w3.one.net/~jhoffman/sqltut.htm>

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>

C 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://members.spree.com/technology/aldev/>
- Mirror site <http://aldev.8m.com>
- Mirror site <http://aldev.webjump.com>
- Mirror site <http://homepages.infoseek.com/~aldev1/index.html>
- Mirror site <http://www3.bcity.com/aldev/>