



PostgreSQL
www.postgresql.org



PostgreSQL 8.4

Server Programming

Volume III



By The PostgreSQL Global Development Group



Linbrary™ - Linux Documentation Library
www.linbrary.com



PostgreSQL 8.4
Official Documentation

Server Programming

Volume III



Fultus™ Books

PostgreSQL



PostgreSQL 8.4 Official Documentation

Server Programming

Volume III

ISBN 1-59682-160-4

Copyright © 1996-2009 The PostgreSQL Global Development Group

Cover design and book layout by Fultus Corporation



Published by Fultus Corporation

Publisher Web: *www.fultus.com*

Linbrary - Linux Library: *www.linbrary.com*

Online Bookstore: *store.fultus.com*

email: *production@fultus.com*



This material may only be distributed subject to the terms and conditions set forth in the BSD License (presently available at <http://www.postgresql.org/about/licence>).

PostgreSQL and PostgreSQL logo are trademarks or registered trademarks of The PostgreSQL Global Development Group, in the U.S. and other countries. All product names and services identified throughout this manual are trademarks or registered trademarks of their respective companies.

The author and publisher have made every effort in the preparation of this book to ensure the accuracy of the information. However, the information contained in this book is offered without warranty, either express or implied. Neither the author nor the publisher nor any dealer or distributor will be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

Table of Contents

List of Tables.....	8
List of Examples	9
License.....	10
Abstract	11
Part V. Server Programming.....	12
Chapter 34. Extending SQL	13
34.1. How Extensibility Works.....	13
34.2. The PostgreSQL Type System.....	14
34.2.1. Base Types	14
34.2.2. Composite Types.....	14
34.2.3. Domains	14
34.2.4. Pseudo-Types.....	14
34.2.5. Polymorphic Types	14
34.3. User-Defined Functions	15
34.4. Query Language (SQL) Functions.....	16
34.4.1. SQL Functions on Base Types.....	17
34.4.2. SQL Functions on Composite Types	19
34.4.3. SQL Functions with Output Parameters.....	22
34.4.4. SQL Functions with Variable Numbers of Arguments.....	23
34.4.5. SQL Functions with Default Values for Arguments	23
34.4.6. SQL Functions as Table Sources.....	24
34.4.7. SQL Functions Returning Sets.....	25
34.4.8. SQL Functions Returning TABLE.....	26
34.4.9. Polymorphic SQL Functions.....	27
34.5. Function Overloading	28
34.6. Function Volatility Categories	29
34.7. Procedural Language Functions	31
34.8. Internal Functions	31
34.9. C-Language Functions	32
34.9.1. Dynamic Loading.....	32
34.9.2. Base Types in C-Language Functions.....	34
34.9.3. Version 0 Calling Conventions.....	37

Volume III

34.9.4. Version 1 Calling Conventions.....	39
34.9.5. Writing Code.....	42
34.9.6. Compiling and Linking Dynamically-Loaded Functions.....	43
34.9.7. Extension Building Infrastructure.....	46
34.9.8. Composite-Type Arguments	48
34.9.9. Returning Rows (Composite Types).....	49
34.9.10. Returning Sets	51
34.9.11. Polymorphic Arguments and Return Types.....	57
34.9.12. Shared Memory and LWLocks	58
34.10. User-Defined Aggregates.....	59
34.11. User-Defined Types	62
34.12. User-Defined Operators	66
34.13. Operator Optimization Information.....	66
34.13.1. COMMUTATOR	67
34.13.2. NEGATOR	68
34.13.3. RESTRICT.....	68
34.13.4. JOIN.....	69
34.13.5. HASHES	70
34.13.6. MERGES	71
34.14. Interfacing Extensions To Indexes	72
34.14.1. Index Methods and Operator Classes.....	72
34.14.2. Index Method Strategies	73
34.14.3. Index Method Support Routines	75
34.14.4. An Example	76
34.14.5. Operator Classes and Operator Families.....	79
34.14.6. System Dependencies on Operator Classes	82
34.14.7. Special Features of Operator Classes	83
Chapter 35. Triggers.....	85
35.1. Overview of Trigger Behavior	85
35.2. Visibility of Data Changes	87
35.3. Writing Trigger Functions in C.....	88
35.4. A Complete Example.....	90
Chapter 36. The Rule System	94
36.1. The Query Tree.....	94
36.2. Views and the Rule System	97
36.2.1. How SELECT Rules Work.....	97
36.2.2. View Rules in Non-SELECT Statements	102
36.2.3. The Power of Views in PostgreSQL.....	103
36.2.4. Updating a View.....	104

36.3. Rules on INSERT, UPDATE, and DELETE	104
36.3.1. How Update Rules Work	104
36.3.1.1. A First Rule Step by Step	106
36.3.2. Cooperation with Views.....	109
36.4. Rules and Privileges	115
36.5. Rules and Command Status	116
36.6. Rules versus Triggers	117
Chapter 37. Procedural Languages.....	120
37.1. Installing Procedural Languages	120
Chapter 38. PL/pgSQL - SQL Procedural Language	123
38.1. Overview.....	123
38.1.1. Advantages of Using PL/pgSQL	123
38.1.2. Supported Argument and Result Data Types.....	124
38.2. Structure of PL/pgSQL.....	124
38.3. Declarations	126
38.3.1. Aliases for Function Parameters	127
38.3.2. Copying Types.....	129
38.3.3. Row Types	129
38.3.4. Record Types.....	130
38.3.5. RENAME	131
38.4. Expressions	131
38.5. Basic Statements	132
38.5.1. Assignment.....	132
38.5.2. Executing a Command With No Result	132
38.5.3. Executing a Query with a Single-Row Result	133
38.5.4. Executing Dynamic Commands.....	135
38.5.5. Obtaining the Result Status.....	138
38.5.6. Doing Nothing At All	139
38.6. Control Structures.....	140
38.6.1. Returning From a Function.....	140
38.6.1.1. RETURN	140
38.6.1.2. RETURN NEXT and RETURN QUERY	140
38.6.2. Conditionals	142
38.6.2.1. IF-THEN.....	142
38.6.2.2. IF-THEN-ELSE.....	142
38.6.2.3. IF-THEN-ELSIF	143
38.6.2.4. Simple CASE	144
38.6.2.5. Searched CASE.....	144
38.6.3. Simple Loops.....	145

Volume III

38.6.3.1. LOOP.....	145
38.6.3.2. EXIT.....	145
38.6.3.3. CONTINUE	146
38.6.3.4. WHILE	147
38.6.3.5. FOR (integer variant)	147
38.6.4. Looping Through Query Results	148
38.6.5. Trapping Errors	149
38.7. Cursors	151
38.7.1. Declaring Cursor Variables.....	151
38.7.2. Opening Cursors	152
38.7.2.1. OPEN FOR query.....	152
38.7.2.2. OPEN FOR EXECUTE.....	152
38.7.2.3. Opening a Bound Cursor.....	153
38.7.3. Using Cursors	153
38.7.3.1. FETCH	153
38.7.3.2. MOVE.....	154
38.7.3.3. UPDATE/DELETE WHERE CURRENT OF.....	154
38.7.3.4. CLOSE	155
38.7.3.5. Returning Cursors	155
38.7.4. Looping Through a Cursor's Result.....	156
38.8. Errors and Messages.....	157
38.9. Trigger Procedures	158
38.10. PL/pgSQL Under the Hood	164
38.10.1. Variable Substitution.....	165
38.10.2. Plan Caching.....	167
38.11. Tips for Developing in PL/pgSQL	169
38.11.1. Handling of Quotation Marks.....	170
38.12. Porting from Oracle PL/SQL.....	171
38.12.1. Porting Examples.....	172
38.12.2. Other Things to Watch For	178
38.12.2.1. Implicit Rollback after Exceptions	178
38.12.2.2. EXECUTE	179
38.12.2.3. Optimizing PL/pgSQL Functions	179
38.12.3. Appendix	179
Chapter 39. PL/Tcl - Tcl Procedural Language.....	182
39.1. Overview	182
39.2. PL/Tcl Functions and Arguments.....	183
39.3. Data Values in PL/Tcl.....	184
39.4. Global Data in PL/Tcl	184

39.5. Database Access from PL/Tcl	185
39.6. Trigger Procedures in PL/Tcl	187
39.7. Modules and the unknown command	189
39.8. Tcl Procedure Names	190
Chapter 40. PL/Perl - Perl Procedural Language	191
40.1. PL/Perl Functions and Arguments	191
40.2. Database Access from PL/Perl	195
40.3. Data Values in PL/Perl	198
40.4. Global Values in PL/Perl	198
40.5. Trusted and Untrusted PL/Perl	199
40.6. PL/Perl Triggers	200
40.7. Limitations and Missing Features	202
Chapter 41. PL/Python - Python Procedural Language	203
41.1. PL/Python Functions	203
41.2. Trigger Functions	207
41.3. Database Access	208
Chapter 42 Server Programming Interface	210
42.1. Interface Functions	210
42.2. Interface Support Functions	231
42.3. Memory Management	236
42.4. Visibility of Data Changes	243
42.5. Examples	243
List of Volumes	247
Index	250

List of Tables

Table 34-1. Equivalent C Types for Built-In SQL Types.....	36
Table 34-2. B-tree Strategies.....	73
Table 34-3. Hash Strategies.....	74
Table 34-4. GiST Two-Dimensional "R-tree" Strategies.....	74
Table 34-5. GIN Array Strategies.....	75
Table 34-6. B-tree Support Functions.....	75
Table 34-7. Hash Support Functions.....	75
Table 34-8. GiST Support Functions.....	76
Table 34-9. GIN Support Functions.....	76

List of Examples

Example 37-1. Manual Installation of PL/pgSQL	122
Example 38-1. Quoting values in dynamic queries.....	138
Example 38-2. Exceptions with UPDATE/INSERT	151
Example 38-3. A PL/pgSQL Trigger Procedure.....	161
Example 38-4. A PL/pgSQL Trigger Procedure For Auditing.....	162
Example 38-5. A PL/pgSQL Trigger Procedure For Maintaining A Summary Table	164
Example 38-6. Porting a Simple Function from PL/SQL to PL/pgSQL	173
Example 38-7. Porting a Function that Creates Another Function	175
Example 38-8. Porting a Procedure With String Manipulation and OUT Parameters	176
Example 38-9. Porting a Procedure from PL/SQL to PL/pgSQL.....	178

License

PostgreSQL is released under the BSD license.

PostgreSQL Database Management System
(formerly known as Postgres, then as Postgres95)

Portions Copyright (c) 1996-2009, The PostgreSQL Global Development Group

Portions Copyright (c) 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Abstract

Welcome to the *PostgreSQL 8.4 Official Documentation*! After many years of development, PostgreSQL has become feature-complete in many areas. This release shows a targeted approach to adding features (e.g., authentication, monitoring, space reuse), and adds capabilities defined in the later SQL standards.

Part V.

Server Programming

Welcome to the PostgreSQL Tutorial. The following few chapters are intended to give a simple introduction to PostgreSQL, relational database concepts, and the SQL language to those who are new to any one of these aspects. We only assume some general knowledge about how to use computers. No particular Unix or programming experience is required. This part is mainly intended to give you some hands-on experience with important aspects of the PostgreSQL system. It makes no attempt to be a complete or thorough treatment of the topics it covers.

After you have worked through this tutorial you might want to move on to reading *Part II* to gain a more formal knowledge of the SQL language, or *Part IV* for information about developing applications for PostgreSQL. Those who set up and manage their own server should also read *Part III*.

Chapter 37.

Procedural Languages

PostgreSQL allows user-defined functions to be written in other languages besides SQL and C. These other languages are generically called *procedural languages* (PLs). For a function written in a procedural language, the database server has no built-in knowledge about how to interpret the function's source text. Instead, the task is passed to a special handler that knows the details of the language. The handler could either do all the work of parsing, syntax analysis, execution, etc. itself, or it could serve as "glue" between PostgreSQL and an existing implementation of a programming language. The handler itself is a C language function compiled into a shared object and loaded on demand, just like any other C function.

There are currently four procedural languages available in the standard PostgreSQL distribution: PL/pgSQL (*Chapter 38* - page 123), PL/Tcl (*Chapter 39* - page 182), PL/Perl (*Chapter 40* - page 191), and PL/Python (*Chapter 41* - page 203). There are additional procedural languages available that are not included in the core distribution. *Appendix G* (Vol.V) has information about finding them. In addition other languages can be defined by users; the basics of developing a new procedural language are covered in *Chapter 48* (Vol.V).

37.1. Installing Procedural Languages

A procedural language must be "installed" into each database where it is to be used. But procedural languages installed in the database `template1` are automatically available in all subsequently created databases, since their entries in `template1` will be copied by `CREATE DATABASE`. So the database administrator can decide which languages are available in which databases and can make some languages available by default if he chooses.

For the languages supplied with the standard distribution, it is only necessary to execute `CREATE LANGUAGE language_name` to install the language into the current database. Alternatively, the program *createlang* (Vol.IV) can be used to do this from the shell command line. For example, to install the language PL/pgSQL into the database `template1`, use:

```
createlang plpgsql template1
```

The manual procedure described below is only recommended for installing custom languages that `CREATE LANGUAGE` does not know about.

Manual Procedural Language Installation

A procedural language is installed in a database in four steps, which must be carried out by a database superuser. (For languages known to `CREATE LANGUAGE`, the second and third steps can be omitted, because they will be carried out automatically if needed.)

1. The shared object for the language handler must be compiled and installed into an appropriate library directory. This works in the same way as building and installing modules with regular user-defined C functions does; see *Section 34.9.6*. Often, the language handler will depend on an external library that provides the actual programming language engine; if so, that must be installed as well.
2. The handler must be declared with the command

```
CREATE FUNCTION handler_function_name()
  RETURNS language_handler
  AS 'path-to-shared-object'
  LANGUAGE C;
```

The special return type of `language_handler` tells the database system that this function does not return one of the defined SQL data types and is not directly usable in SQL statements.

3. Optionally, the language handler can provide a "validator" function that checks a function definition for correctness without actually executing it. The validator function is called by `CREATE FUNCTION` if it exists. If a validator function is provided by the handler, declare it with a command like

```
CREATE FUNCTION validator_function_name(oid)
  RETURNS void
  AS 'path-to-shared-object'
  LANGUAGE C;
```

4. The PL must be declared with the command

```
CREATE [TRUSTED] [PROCEDURAL] LANGUAGE language-name
  HANDLER handler_function_name
  [VALIDATOR validator_function_name] ;
```

The optional key word `TRUSTED` specifies that ordinary database users that have no superuser privileges should be allowed to use this language to create functions and trigger procedures. Since PL functions are executed inside the database server, the `TRUSTED` flag should only be given for languages that do not allow access to database server internals or the file system. The languages PL/pgSQL, PL/Tcl, and PL/Perl are considered trusted; the languages PL/TclU, PL/PerlU, and PL/PythonU are designed to provide unlimited functionality and should *not* be marked trusted.

Example 37-1 shows how the manual installation procedure would work with the language PL/pgSQL.

The following command tells the database server where to find the shared object for the PL/pgSQL language's call handler function.

```
CREATE FUNCTION plpgsql_call_handler() RETURNS language_handler AS
    '$libdir/plpgsql' LANGUAGE C;
```

PL/pgSQL has a validator function, so we declare that too:

```
CREATE FUNCTION plpgsql_validator(oid) RETURNS void AS
    '$libdir/plpgsql' LANGUAGE C;
```

The command:

```
CREATE TRUSTED PROCEDURAL LANGUAGE plpgsql
    HANDLER plpgsql_call_handler
    VALIDATOR plpgsql_validator;
```

then defines that the previously declared functions should be invoked for functions and trigger procedures where the language attribute is plpgsql.

In a default PostgreSQL installation, the handler for the PL/pgSQL language is built and installed into the "library" directory. If Tcl support is configured in, the handlers for PL/Tcl and PL/TclU are also built and installed in the same location. Likewise, the PL/Perl and PL/PerlU handlers are built and installed if Perl support is configured, and the PL/PythonU handler is installed if Python support is configured.

Example 37-1. Manual Installation of PL/pgSQL

List of Volumes

Volume I. The SQL Language

Preface

- What is PostgreSQL?
- A Brief History of PostgreSQL
- Conventions
- Further Information
- Bug Reporting Guidelines

Part I. Tutorial

1. Getting Started
2. The SQL Language
3. Advanced Features

Part II. The SQL Language

4. SQL Syntax
5. Data Definition
6. Data Manipulation
7. Queries
8. Data Types
9. Functions and Operators
10. Type Conversion
11. Indexes
12. Full Text Search
13. Concurrency Control
14. Performance Tips

Volume II. Server Administration

Part III. Server Administration

15. Installation from Source Code
16. Installation from Source Code on Windows
17. Server Setup and Operation

18. Server Configuration
19. Client Authentication
20. Database Roles and Privileges
21. Managing Databases
22. Localization
23. Routine Database Maintenance Tasks
24. Backup and Restore
25. High Availability, Load Balancing, and Replication
26. Monitoring Database Activity
27. Monitoring Disk Usage
28. Reliability and the Write-Ahead Log
29. Regression Tests

Part IV. Client Interfaces

30. libpq - C Library
31. Large Objects
32. ECPG - Embedded SQL in C
33. The Information Schema

Volume III. Server Programming**Part V. Server Programming**

34. Extending SQL
35. Triggers
36. The Rule System
37. Procedural Languages
38. PL/pgSQL - SQL Procedural Language
39. PL/Tcl - Tcl Procedural Language
40. PL/Perl - Perl Procedural Language
41. PL/Python - Python Procedural Language
42. Server Programming Interface

Volume IV. Reference**Part VI. Reference**

- I. SQL Commands
- II. PostgreSQL Client Applications
- III. PostgreSQL Server Applications

Volume V. Internals and Appendixes

Part VII. Internals

43. Overview of PostgreSQL Internals
44. System Catalogs
45. Frontend/Backend Protocol
46. PostgreSQL Coding Conventions
47. Native Language Support
48. Writing A Procedural Language Handler
49. Genetic Query Optimizer
50. Index Access Method Interface Definition
51. GiST Indexes
52. GIN Indexes
53. Database Physical Storage
54. BKI Backend Interface
55. How the Planner Uses Statistics

Part VIII. Appendixes

- A. PostgreSQL Error Codes
- B. Date/Time Support
- C. SQL Key Words
- D. SQL Conformance
- E. Release Notes
- F. Additional Supplied Modules
- G. External Projects
- H. The CVS Repository
- I. Documentation
- J. Acronyms

Bibliography

Index

Index

- data type
 - base14
 - composite14
 - internal organization34
 - user-defined62
- Digital UNIX..... *See* Tru64 UNIX
- dynamic loading32
- dynamic_library_path32
- elog
 - in PL/Perl195
 - in PL/Python.....208
 - in PL/Tcl185
- exceptions
 - in PL/PgSQL149
- EXIT
 - in PL/pgSQL145
- extending SQL13
- field
 - computed19
- FreeBSD
 - shared library43
- function
 - default values for arguments23
 - internal.....31
 - output parameter22
 - polymorphic14
 - RETURNS TABLE.....26
 - user-defined15
 - in C32
 - in SQL16
 - variadic23
 - with SETOF25
- global data
 - in PL/Python.....203
 - in PL/Tcl184
- HP-UX
 - shared library43
- IMMUTABLE29
- index
 - for user-defined data type72
- input function.....62
 - of a data type62
- instr172
- IRIX
 - shared library43
- library finalization function.....32
- library initialization function32
- Linux
 - shared library43
- loop
 - in PL/pgSQL145
- MacOS X
 - shared library43
- magic block32
- memory context
 - in SPI236
- NetBSD
 - shared library43
- null value
 - in PL/Perl191

PL/Python	203	with views	115
OpenBSD		procedural language	120
shared library	43	query tree	94
operator		quote_ident	
user-defined	66	use in PL/PgSQL	135
operator class	72	quote_literal	
operator family	79	use in PL/PgSQL	135
Oracle		quote_nullable	
porting from PL/SQL to PL/pgSQL	171	use in PL/PgSQL	135
ordering operator	82	RAISE	157
output function	62	range table	94
of a data type	62	reporting errors	
overloading		in PL/PgSQL	157
functions	28	RETURN NEXT	
operators	66	in PL/PgSQL	140
palloc	42	RETURN QUERY	
perl	191	in PL/PgSQL	140
pfree	42	RETURNING INTO	
pg_config		in PL/pgSQL	133
with user-defined C functions	42	rule	94
pgxs	46	and views	97
PIC	43	compared with triggers	117
PL/Perl	191	for DELETE	104
PL/PerlU	199	for INSERT	104
PL/pgSQL	123	for SELECT	97
PL/Python	203	for UPDATE	104
PL/SQL (Oracle)		SELECT INTO	
porting to PL/pgSQL	171	in PL/pgSQL	133
PL/Tcl	182	shared-preload-libraries	58
polymorphic function	14	Solaris	
polymorphic type	14	shared library	43
preparing a query		SPI	210
in PL/pgSQL	167	STABLE	29
in PL/Python	208	target list	94
in PL/Tcl	185	Tcl	182
privilege		TOAST	
with rules	115	and user-defined types	62

trigger	85	polymorphic	14
arguments for trigger functions	85	UnixWare	
compared with rules	117	shared library	43
in C	88	variadic function	23
in PL/pgSQL	158	view	
in PL/Python	207	implementation through rules	97
in PL/Tcl	187	updating	109
Tru64 UNIX		VOLATILE	29
shared library	43	volatility	
trusted		functions	29
PL/Perl	199	WHILE	
type		in PL/pgSQL	147