

## What is PostgreSQL?

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux and Windows. It includes most SQL92 and SQL99 data types. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces and exceptional documentation.



PostgreSQL  
[www.postgresql.org](http://www.postgresql.org)



PostgreSQL and the PostgreSQL logo are trademarks of The PostgreSQL Global Development Group

## About PostgreSQL

An enterprise class database, PostgreSQL boasts sophisticated features such as Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, and write ahead logging for fault tolerance. It supports international character sets, multibyte character encodings, Unicode, and it is locale-aware for sorting, case-sensitivity, and formatting. It is highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL systems in production environments that manage in excess of 4 terabytes of data.

**Fultus™**



Published by Fultus Corporation  
[www.fultus.com](http://www.fultus.com)



PostgreSQL  
[www.postgresql.org](http://www.postgresql.org)



# PostgreSQL 8.4

## The SQL Language

### Volume I



By The PostgreSQL Global Development Group

PostgreSQL 8.4 • The SQL Language





**PostgreSQL 8.4**  
**Official Documentation**

**The SQL Language**

**Volume I**



*Fultus™ Books*

# PostgreSQL



## **PostgreSQL 8.4 Official Documentation**

### **The SQL Language**

#### **Volume I**

ISBN 1-59682-158-2

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.....	12
List of Examples .....	15
License.....	16
Abstract .....	17
Preface .....	18
What is PostgreSQL? .....	18
A Brief History of PostgreSQL .....	19
Conventions .....	22
Further Information .....	22
Bug Reporting Guidelines .....	23
<b>Part I Tutorial .....</b>	<b>28</b>
<b>Chapter 1. Getting Started.....</b>	<b>29</b>
1.1. Installation.....	29
1.2. Architectural Fundamentals .....	29
1.3. Creating a Database .....	30
1.4. Accessing a Database.....	32
<b>Chapter 2 The SQL Language.....</b>	<b>34</b>
2.1. Introduction .....	34
2.2. Concepts .....	34
2.3. Creating a New Table .....	35
2.4. Populating a Table With Rows.....	36
2.5. Querying a Table .....	37
2.6. Joins Between Tables .....	39
2.7. Aggregate Functions .....	41
2.8. Updates.....	43
2.9. Deletions.....	43
<b>Chapter 3. Advanced Features .....</b>	<b>44</b>
3.1. Introduction .....	44
3.2. Views.....	44
3.3. Foreign Keys .....	44
3.4. Transactions .....	45

## Volume I

---

3.5. Window Functions.....	48
3.6. Inheritance.....	51
3.7. Conclusion.....	53
<b>Part II. The SQL Language.....</b>	<b>54</b>
<b>Chapter 4. SQL Syntax.....</b>	<b>55</b>
4.1. Lexical Structure.....	55
4.1.1. Identifiers and Key Words.....	56
4.1.2. Constants.....	57
4.1.2.1. String Constants.....	58
4.1.2.2. String Constants with C-Style Escapes.....	58
4.1.2.3. String Constants with Unicode Escapes.....	59
4.1.2.4. Dollar-Quoted String Constants.....	60
4.1.2.5. Bit-String Constants.....	61
4.1.2.6. Numeric Constants.....	61
4.1.2.7. Constants of Other Types.....	62
4.1.3. Operators.....	63
4.1.4. Special Characters.....	63
4.1.5. Comments.....	64
4.1.6. Lexical Precedence.....	64
4.2. Value Expressions.....	66
4.2.1. Column References.....	67
4.2.2. Positional Parameters.....	67
4.2.3. Subscripts.....	67
4.2.4. Field Selection.....	68
4.2.5. Operator Invocations.....	68
4.2.6. Function Calls.....	69
4.2.7. Aggregate Expressions.....	69
4.2.8. Window Function Calls.....	70
4.2.9. Type Casts.....	71
4.2.10. Scalar Subqueries.....	72
4.2.11. Array Constructors.....	73
4.2.12. Row Constructors.....	74
4.2.13. Expression Evaluation Rules.....	76
<b>Chapter 5. Data Definition.....</b>	<b>77</b>
5.1. Table Basics.....	77
5.2. Default Values.....	79
5.3. Constraints.....	80
5.3.1. Check Constraints.....	80
5.3.2. Not-Null Constraints.....	82

5.3.3. Unique Constraints .....	83
5.3.4. Primary Keys.....	84
5.3.5. Foreign Keys .....	84
5.4. System Columns.....	87
5.5. Modifying Tables .....	89
5.5.1. Adding a Column.....	89
5.5.2. Removing a Column .....	90
5.5.3. Adding a Constraint .....	90
5.5.4. Removing a Constraint.....	91
5.5.5. Changing a Column's Default Value .....	91
5.5.6. Changing a Column's Data Type .....	91
5.5.7. Renaming a Column .....	92
5.5.8. Renaming a Table .....	92
5.6. Privileges .....	92
5.7. Schemas .....	93
5.7.1. Creating a Schema.....	94
5.7.2. The Public Schema .....	95
5.7.3. The Schema Search Path.....	95
5.7.4. Schemas and Privileges .....	96
5.7.5. The System Catalog Schema .....	97
5.7.6. Usage Patterns .....	97
5.7.7. Portability .....	98
5.8. Inheritance.....	98
5.8.1. Caveats.....	101
5.9. Partitioning.....	102
5.9.1. Overview .....	102
5.9.2. Implementing Partitioning.....	103
5.9.3. Managing Partitions.....	106
5.9.4. Partitioning and Constraint Exclusion .....	107
5.9.5. Alternative Partitioning Methods .....	109
5.9.6. Caveats.....	110
5.10. Other Database Objects.....	110
5.11. Dependency Tracking .....	111
<b>Chapter 6. Data Manipulation.....</b>	<b>113</b>
6.1. Inserting Data .....	113
6.2. Updating Data .....	114
6.3. Deleting Data .....	115
<b>Chapter 7. Queries .....</b>	<b>116</b>
7.1. Overview .....	116

## Volume I

---

7.2. Table Expressions.....	117
7.2.1. The FROM Clause.....	117
7.2.1.1. Joined Tables.....	117
7.2.1.2. Table and Column Aliases.....	121
7.2.1.3. Subqueries.....	122
7.2.1.4. Table Functions .....	123
7.2.2. The WHERE Clause .....	124
7.2.3. The GROUP BY and HAVING Clauses .....	125
7.2.4. Window Function Processing.....	127
7.3. Select Lists .....	128
7.3.1. Select-List Items .....	128
7.3.2. Column Labels .....	128
7.3.3. DISTINCT.....	129
7.4. Combining Queries.....	130
7.5. Sorting Rows.....	130
7.6. LIMIT and OFFSET .....	131
7.7. VALUES Lists .....	132
7.8. WITH Queries.....	133
<b>Chapter 8. Data Types .....</b>	<b>137</b>
8.1. Numeric Types .....	138
8.1.1. Integer Types.....	139
8.1.2. Arbitrary Precision Numbers .....	139
8.1.3. Floating-Point Types.....	141
8.1.4. Serial Types .....	142
8.2. Monetary Types.....	143
8.3. Character Types.....	144
8.4. Binary Data Types.....	146
8.5. Date/Time Types .....	148
8.5.1. Date/Time Input .....	149
8.5.1.1. Dates .....	150
8.5.1.2. Times.....	151
8.5.1.3. Time Stamps .....	152
8.5.1.4. Special Values.....	153
8.5.2. Date/Time Output .....	153
8.5.3. Time Zones .....	154
8.5.4. Interval Input .....	156
8.5.5. Interval Output .....	158
8.5.6. Internals .....	159
8.6. Boolean Type .....	159

8.7. Enumerated Types .....	160
8.7.1. Declaration of Enumerated Types .....	160
8.7.2. Ordering .....	161
8.7.3. Type Safety .....	161
8.7.4. Implementation Details .....	162
8.8. Geometric Types.....	162
8.8.1. Points.....	163
8.8.2. Line Segments.....	163
8.8.3. Boxes.....	163
8.8.4. Paths .....	164
8.8.5. Polygons .....	164
8.8.6. Circles.....	164
8.9. Network Address Types .....	165
8.9.1. inet .....	165
8.9.2. cidr.....	165
8.9.3. inet vs. cidr .....	166
8.9.4. macaddr .....	166
8.10. Bit String Types .....	167
8.11. Text Search Types .....	168
8.11.1. tsvector.....	168
8.11.2. tsquery .....	169
8.12. UUID Type.....	170
8.13. XML Type .....	171
8.13.1. Creating XML Values.....	171
8.13.2. Encoding Handling .....	172
8.13.3. Accessing XML Values .....	173
8.14. Arrays .....	173
8.14.1. Declaration of Array Types.....	173
8.14.2. Array Value Input .....	174
8.14.3. Accessing Arrays .....	176
8.14.4. Modifying Arrays.....	177
8.14.5. Searching in Arrays.....	180
8.14.6. Array Input and Output Syntax .....	181
8.15. Composite Types .....	182
8.15.1. Declaration of Composite Types .....	183
8.15.2. Composite Value Input.....	184
8.15.3. Accessing Composite Types .....	184
8.15.4. Modifying Composite Types .....	185
8.15.5. Composite Type Input and Output Syntax .....	185

## Volume I

---

8.16. Object Identifier Types .....	187
8.17. Pseudo-Types .....	188
<b>Chapter 9. Functions and Operators .....</b>	<b>190</b>
9.1. Logical Operators .....	190
9.2. Comparison Operators .....	191
9.3. Mathematical Functions and Operators.....	193
9.4. String Functions and Operators .....	196
9.5. Binary String Functions and Operators .....	205
9.6. Bit String Functions and Operators .....	207
9.7. Pattern Matching.....	207
9.7.1. LIKE.....	208
9.7.2. SIMILAR TO Regular Expressions .....	209
9.7.3. POSIX Regular Expressions .....	210
9.7.3.1. Regular Expression Details.....	213
9.7.3.2. Bracket Expressions .....	216
9.7.3.3. Regular Expression Escapes .....	217
9.7.3.4. Regular Expression Metasyntax .....	219
9.7.3.5. Regular Expression Matching Rules .....	221
9.7.3.6. Limits and Compatibility.....	223
9.7.3.7. Basic Regular Expressions .....	223
9.8. Data Type Formatting Functions .....	224
9.9. Date/Time Functions and Operators .....	230
9.9.1. EXTRACT, date_part .....	233
9.9.2. date_trunc.....	237
9.9.3. AT TIME ZONE.....	238
9.9.4. Current Date/Time .....	239
9.9.5. Delaying Execution .....	241
9.10. Enum Support Functions.....	241
9.11. Geometric Functions and Operators .....	242
9.12. Network Address Functions and Operators.....	245
9.13. Text Search Functions and Operators .....	247
9.14. XML Functions .....	250
9.14.1. Producing XML Content .....	250
9.14.1.1. xmlcomment .....	250
9.14.1.2. xmlconcat .....	251
9.14.1.3. xmlelement .....	251
9.14.1.4. xmlforest.....	252
9.14.1.5. xmlpi .....	253
9.14.1.6. xmlroot .....	253

---

9.14.1.7. xmlagg .....	254
9.14.1.8. XML Predicates .....	254
9.14.2. Processing XML .....	254
9.14.3. Mapping Tables to XML .....	255
9.15. Sequence Manipulation Functions .....	259
9.16. Conditional Expressions .....	261
9.16.1. CASE .....	261
9.16.2. COALESCE .....	263
9.16.3. NULLIF .....	263
9.16.4. GREATEST and LEAST .....	263
9.17. Array Functions and Operators .....	264
9.18. Aggregate Functions .....	265
9.19. Window Functions .....	269
9.20. Subquery Expressions .....	270
9.20.1. EXISTS .....	271
9.20.2. IN .....	271
9.20.3. NOT IN .....	272
9.20.4. ANY/SOME .....	272
9.20.5. ALL .....	273
9.20.6. Row-wise Comparison .....	274
9.21. Row and Array Comparisons .....	274
9.21.1. IN .....	274
9.21.2. NOT IN .....	275
9.21.3. ANY/SOME (array) .....	275
9.21.4. ALL (array) .....	276
9.21.5. Row-wise Comparison .....	276
9.22. Set Returning Functions .....	277
9.23. System Information Functions .....	280
9.24. System Administration Functions .....	289
9.25. Trigger Functions .....	295
<b>Chapter 10. Type Conversion .....</b>	<b>296</b>
10.1. Overview .....	296
10.2. Operators .....	298
10.3. Functions .....	301
10.4. Value Storage .....	304
10.5. UNION, CASE, and Related Constructs .....	305
<b>Chapter 11. Indexes .....</b>	<b>307</b>
11.1. Introduction .....	307
11.2. Index Types .....	308

---

## Volume I

---

11.3. Multicolumn Indexes .....	310
11.4. Indexes and ORDER BY.....	311
11.5. Combining Multiple Indexes .....	312
11.6. Unique Indexes.....	313
11.7. Indexes on Expressions .....	314
11.8. Partial Indexes .....	315
11.9. Operator Classes and Operator Families.....	318
11.10. Examining Index Usage.....	319
<b>Chapter 12 Full Text Search.....</b>	<b>321</b>
12.1. Introduction .....	321
12.1.1. What Is a Document?.....	322
12.1.2. Basic Text Matching .....	323
12.1.3. Configurations .....	324
12.2. Tables and Indexes .....	325
12.2.1. Searching a Table.....	325
12.2.2. Creating Indexes.....	326
12.3. Controlling Text Search.....	327
12.3.1. Parsing Documents .....	328
12.3.2. Parsing Queries.....	329
12.3.3. Ranking Search Results .....	330
12.3.4. Highlighting Results .....	332
12.4. Additional Features .....	334
12.4.1. Manipulating Documents .....	334
12.4.2. Manipulating Queries.....	335
12.4.2.1. Query Rewriting .....	336
12.4.3. Triggers for Automatic Updates .....	338
12.4.4. Gathering Document Statistics .....	339
12.5. Parsers .....	340
12.6. Dictionaries .....	341
12.6.1. Stop Words .....	342
12.6.2. Simple Dictionary .....	343
12.6.3. Synonym Dictionary .....	344
12.6.4. Thesaurus Dictionary.....	345
12.6.4.1. Thesaurus Configuration.....	346
12.6.4.2. Thesaurus Example .....	347
12.6.5. Ispell Dictionary .....	348
12.6.6. Snowball Dictionary.....	349
12.7. Configuration Example.....	349
12.8. Testing and Debugging Text Search .....	351

---

12.8.1. Configuration Testing .....	351
12.8.2. Parser Testing .....	353
12.8.3. Dictionary Testing .....	354
12.9. GiST and GIN Index Types .....	355
12.10. psql Support .....	357
12.11. Limitations .....	359
12.12. Migration from Pre-8.3 Text Search .....	360
<b>Chapter 13. Concurrency Control .....</b>	<b>361</b>
13.1. Introduction .....	361
13.2. Transaction Isolation .....	361
13.2.1. Read Committed Isolation Level .....	362
13.2.2. Serializable Isolation Level .....	364
13.2.2.1. Serializable Isolation versus True Serializability .....	365
13.3. Explicit Locking .....	366
13.3.1. Table-Level Locks .....	366
13.3.2. Row-Level Locks .....	369
13.3.3. Deadlocks .....	370
13.3.4. Advisory Locks .....	371
13.4. Data Consistency Checks at the Application Level .....	372
13.5. Locking and Indexes .....	373
<b>Chapter 14. Performance Tips .....</b>	<b>374</b>
14.1. Using EXPLAIN .....	374
14.2. Statistics Used by the Planner .....	380
14.3. Controlling the Planner with Explicit JOIN Clauses .....	381
14.4. Populating a Database .....	384
14.4.1. Disable Autocommit .....	384
14.4.2. Use COPY .....	384
14.4.3. Remove Indexes .....	385
14.4.4. Remove Foreign Key Constraints .....	385
14.4.5. Increase maintenance_work_mem .....	385
14.4.6. Increase checkpoint_segments .....	385
14.4.7. Turn off archive_mode .....	385
14.4.8. Run ANALYZE Afterwards .....	386
14.4.9. Some Notes About pg_dump .....	386
<b>List of Volumes .....</b>	<b>388</b>
<b>Index .....</b>	<b>391</b>

---

# List of Tables

Table 4.1. Backslash Escape Sequences .....	58
Table 4.2. Operator Precedence (decreasing) .....	65
Table 8.1. Data Types.....	138
Table 8.2. Numeric Types .....	139
Table 8.3. Monetary Types.....	144
Table 8.4. Character Types.....	144
Table 8.5. Special Character Types .....	146
Table 8.6. Binary Data Types .....	146
Table 8.7. <code>bytea</code> Literal Escaped Octets .....	147
Table 8.8. <code>bytea</code> Output Escaped Octets .....	147
Table 8.9. Date/Time Types .....	148
Table 8.10. Date Input.....	150
Table 8.11. Time Input.....	151
Table 8.12. Time Zone Input.....	151
Table 8.13. Special Date/Time Inputs .....	153
Table 8.14. Date/Time Output Styles.....	154
Table 8.15. Date Order Conventions .....	154
Table 8.16. ISO 8601 interval unit abbreviations .....	157
Table 8.17. Interval Input.....	158
Table 8.18. Interval Output Style Examples .....	159
Table 8.19. Geometric Types.....	163
Table 8.20. Network Address Types .....	165
Table 8.21. <code>cidr</code> Type Input Examples .....	166
Table 8.22. Object Identifier Types .....	188
Table 8.23. Pseudo-Types.....	189
Table 9.1. Comparison Operators .....	191
Table 9.2. Mathematical Operators.....	194
Table 9.3. Mathematical Functions .....	195

---

Table 9.4. Trigonometric Functions .....	196
Table 9.5. SQL String Functions and Operators.....	197
Table 9.6. Other String Functions .....	201
Table 9.7. Built-in Conversions .....	205
Table 9.8. SQL Binary String Functions and Operators.....	206
Table 9.9. Other Binary String Functions.....	206
Table 9.10. Bit String Operators .....	207
Table 9.11. Regular Expression Match Operators .....	210
Table 9.12. Regular Expression Atoms.....	214
Table 9.13. Regular Expression Quantifiers .....	215
Table 9.14. Regular Expression Constraints .....	215
Table 9.15. Regular Expression Character-Entry Escapes .....	218
Table 9.16. Regular Expression Class-Shorthand Escapes .....	218
Table 9.17. Regular Expression Constraint Escapes.....	219
Table 9.18. Regular Expression Back References.....	219
Table 9.19. ARE Embedded-Option Letters .....	220
Table 9.20. Formatting Functions.....	224
Table 9.21. Template Patterns for Date/Time Formatting .....	226
Table 9.22. Template Pattern Modifiers for Date/Time Formatting .....	226
Table 9.23. Template Patterns for Numeric Formatting .....	228
Table 9.24. Template Pattern Modifiers for Numeric Formatting.....	229
Table 9.25. to_char Examples.....	230
Table 9.26. Date/Time Operators .....	231
Table 9.27. Date/Time Functions.....	233
Table 9.28. AT TIME ZONE Variants .....	238
Table 9.29. Enum Support Functions .....	242
Table 9.30. Geometric Operators.....	243
Table 9.31. Geometric Functions.....	244
Table 9.32. Geometric Type Conversion Functions.....	244
Table 9.33. cidr and inet Operators .....	245
Table 9.34. cidr and inet Functions .....	246
Table 9.35. macaddr Functions.....	247
Table 9.36. Text Search Operators.....	247
Table 9.37. Text Search Functions .....	249
Table 9.38. Text Search Debugging Functions .....	250
Table 9.39. Sequence Functions.....	259

---

## Volume I

---

Table 9.40. Array Operators .....	264
Table 9.41. Array Functions .....	265
Table 9.42. General-Purpose Aggregate Functions .....	266
Table 9.43. Aggregate Functions for Statistics .....	269
Table 9.44. General-Purpose Window Functions .....	270
Table 9.45. Series Generating Functions .....	277
Table 9.46. Subscript Generating Functions .....	278
Table 9.47. Session Information Functions .....	280
Table 9.48. Access Privilege Inquiry Functions .....	282
Table 9.49. Schema Visibility Inquiry Functions .....	285
Table 9.50. System Catalog Information Functions .....	286
Table 9.51. Comment Information Functions.....	288
Table 9.52. Transaction IDs and snapshots.....	289
Table 9.53. Snapshot components.....	289
Table 9.54. Configuration Settings Functions .....	289
Table 9.55. Server Signalling Functions .....	290
Table 9.56. Backup Control Functions.....	291
Table 9.57. Database Object Size Functions.....	292
Table 9.58. Generic File Access Functions .....	293
Table 9.59. Advisory Lock Functions .....	294
Table 12.1. Default Parser's Token Types .....	340
Table 13.1. SQL Transaction Isolation Levels.....	362
Table 13.2. Conflicting lock modes.....	369

# List of Examples

Example 8.1. Using the character types .....	146
Example 8.2. Using the <code>boolean</code> type .....	160
Example 8.3. Basic Enum Usage .....	161
Example 8.4. Enum Ordering .....	161
Example 8.5. Lack of Casting .....	162
Example 8.6. Comparing Different Enums by Casting to Text .....	162
Example 8.7. Using the bit string types .....	167
Example 9.1. XSLT stylesheet for converting SQL/XML output to HTML.....	258
Example 10.1. Factorial Operator Type Resolution.....	299
Example 10.2. String Concatenation Operator Type Resolution.....	300
Example 10.3. Absolute-Value and Negation Operator Type Resolution .....	301
Example 10.4. Rounding Function Argument Type Resolution .....	303
Example 10.5. Substring Function Type Resolution .....	304
Example 10.6. <code>character</code> Storage Type Conversion.....	305
Example 10.7. Type Resolution with Underspecified Types in a Union.....	306
Example 10.8. Type Resolution in a Simple Union .....	306
Example 10.9. Type Resolution in a Transposed Union.....	306
Example 11.1. Setting up a Partial Index to Exclude Common Values .....	316
Example 11.2. Setting up a Partial Index to Exclude Uninteresting Values .....	316
Example 11.3. Setting up a Partial Unique Index .....	317

# 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.

# Preface

This book is the official documentation of PostgreSQL. It has been written by the PostgreSQL developers and other volunteers in parallel to the development of the PostgreSQL software. It describes all the functionality that the current version of PostgreSQL officially supports.

To make the large amount of information about PostgreSQL manageable, this book has been organized in several parts. Each part is targeted at a different class of users, or at users in different stages of their PostgreSQL experience:

- *Part I* is an informal introduction for new users.
- *Part II* documents the SQL query language environment, including data types and functions, as well as user-level performance tuning. Every PostgreSQL user should read this.
- *Part III* describes the installation and administration of the server. Everyone who runs a PostgreSQL server, be it for private use or for others, should read this part.
- *Part IV* describes the programming interfaces for PostgreSQL client programs.
- *Part V* contains information for advanced users about the extensibility capabilities of the server. Topics include user-defined data types and functions.
- *Part VI* contains reference information about SQL commands, client and server programs. This part supports the other parts with structured information sorted by command or program.
- *Part VII* contains assorted information that might be of use to PostgreSQL developers.

## What is PostgreSQL?

PostgreSQL is an object-relational database management system (ORDBMS) based on *POSTGRES, Version 4.2*<sup>1</sup>, developed at the University of California at Berkeley Computer Science Department. POSTGRES pioneered many concepts that only became available in some commercial database systems much later.

---

<sup>1</sup> <http://s2k-ftp.cs.berkeley.edu:8000/postgres/postgres.html>

PostgreSQL is an open-source descendant of this original Berkeley code. It supports a large part of the SQL standard and offers many modern features:

- complex queries
- foreign keys
- triggers
- views
- transactional integrity
- multiversion concurrency control

Also, PostgreSQL can be extended by the user in many ways, for example by adding new

- data types
- functions
- operators
- aggregate functions
- index methods
- procedural languages

And because of the liberal license, PostgreSQL can be used, modified, and distributed by anyone free of charge for any purpose, be it private, commercial, or academic.

## **A Brief History of PostgreSQL**

The object-relational database management system now known as PostgreSQL is derived from the POSTGRES package written at the University of California at Berkeley. With over two decades of development behind it, PostgreSQL is now the most advanced open-source database available anywhere.

### **The Berkeley POSTGRES Project**

The POSTGRES project, led by Professor Michael Stonebraker, was sponsored by the Defense Advanced Research Projects Agency (DARPA), the Army Research Office (ARO), the National Science Foundation (NSF), and ESL, Inc. The implementation of POSTGRES began in 1986. The initial concepts for the system were presented in *The design of POSTGRES*<sup>2</sup>, and the definition of the initial data model appeared in *The POSTGRES data model*<sup>3</sup>. The design of the rule system at that time was described in *The design of the*

---

<sup>2</sup> <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M85-95.pdf>

<sup>3</sup> <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M87-13.pdf>

## Volume I

---

*POSTGRES rules system* (see "Bibliography" section). The rationale and architecture of the storage manager were detailed in *The design of the POSTGRES storage system*<sup>4</sup>.

POSTGRES has undergone several major releases since then. The first "demoware" system became operational in 1987 and was shown at the 1988 ACM-SIGMOD Conference. Version 1, described in *The implementation of POSTGRES*<sup>5</sup>, was released to a few external users in June 1989. In response to a critique of the first rule system (*A commentary on the POSTGRES rules system*<sup>6</sup>), the rule system was redesigned (*On Rules, Procedures, Caching and Views in Database Systems*<sup>7</sup>), and Version 2 was released in June 1990 with the new rule system. Version 3 appeared in 1991 and added support for multiple storage managers, an improved query executor, and a rewritten rule system. For the most part, subsequent releases until Postgres95 (see below) focused on portability and reliability.

POSTGRES has been used to implement many different research and production applications. These include: a financial data analysis system, a jet engine performance monitoring package, an asteroid tracking database, a medical information database, and several geographic information systems. POSTGRES has also been used as an educational tool at several universities. Finally, Illustra Information Technologies (later merged into *Informix*<sup>8</sup>, which is now owned by *IBM*<sup>9</sup>) picked up the code and commercialized it. In late 1992, POSTGRES became the primary data manager for the *Sequoia 2000 scientific computing project*<sup>10</sup>.

The size of the external user community nearly doubled during 1993. It became increasingly obvious that maintenance of the prototype code and support was taking up large amounts of time that should have been devoted to database research. In an effort to reduce this support burden, the Berkeley POSTGRES project officially ended with Version 4.2.

## Postgres95

In 1994, Andrew Yu and Jolly Chen added an SQL language interpreter to POSTGRES. Under a new name, Postgres95 was subsequently released to the web to find its own way in the world as an open-source descendant of the original POSTGRES Berkeley code.

---

<sup>4</sup> <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M87-06.pdf>

<sup>5</sup> <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M90-34.pdf>

<sup>6</sup> <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M89-82.pdf>

<sup>7</sup> <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M90-36.pdf>

<sup>8</sup> <http://www.informix.com/>

<sup>9</sup> <http://www.ibm.com/>

<sup>10</sup> [http://meteora.ucsd.edu/s2k/s2k\\_home.html](http://meteora.ucsd.edu/s2k/s2k_home.html)

---

Postgres95 code was completely ANSI C and trimmed in size by 25%. Many internal changes improved performance and maintainability. Postgres95 release 1.0.x ran about 30-50% faster on the Wisconsin Benchmark compared to POSTGRES, Version 4.2. Apart from bug fixes, the following were the major enhancements:

- The query language PostQUEL was replaced with SQL (implemented in the server). Subqueries were not supported until PostgreSQL (see below), but they could be imitated in Postgres95 with user-defined SQL functions. Aggregate functions were re-implemented. Support for the `GROUP BY` query clause was also added.
- A new program (`psql`) was provided for interactive SQL queries, which used GNU Readline. This largely superseded the old monitor program.
- A new front-end library, `libpgtcl`, supported Tcl-based clients. A sample shell, `pgtclsh`, provided new Tcl commands to interface Tcl programs with the Postgres95 server.
- The large-object interface was overhauled. The inversion large objects were the only mechanism for storing large objects. (The inversion file system was removed.)
- The instance-level rule system was removed. Rules were still available as rewrite rules.
- A short tutorial introducing regular SQL features as well as those of Postgres95 was distributed with the source code
- GNU make (instead of BSD make) was used for the build. Also, Postgres95 could be compiled with an unpatched GCC (data alignment of doubles was fixed).

## PostgreSQL

By 1996, it became clear that the name "Postgres95" would not stand the test of time. We chose a new name, PostgreSQL, to reflect the relationship between the original POSTGRES and the more recent versions with SQL capability. At the same time, we set the version numbering to start at 6.0, putting the numbers back into the sequence originally begun by the Berkeley POSTGRES project.

Many people continue to refer to PostgreSQL as "Postgres" (now rarely in all capital letters) because of tradition or because it is easier to pronounce. This usage is widely accepted as a nickname or alias.

The emphasis during development of Postgres95 was on identifying and understanding existing problems in the server code. With PostgreSQL, the emphasis has shifted to augmenting features and capabilities, although work continues in all areas.

Details about what has happened in PostgreSQL since then can be found in *Appendix E*.

### Conventions

This book uses the following typographical conventions to mark certain portions of text: new terms, foreign phrases, and other important passages are emphasized in *italics*. Everything that represents input or output of the computer, in particular commands, program code, and screen output, is shown in a monospaced font (`example`). Within such passages, italics (*example*) indicate placeholders; you must insert an actual value instead of the placeholder. On occasion, parts of program code are emphasized in bold face (**example**), if they have been added or changed since the preceding example.

The following conventions are used in the synopsis of a command: brackets ([ and ]) indicate optional parts. (In the synopsis of a Tcl command, question marks (?) are used instead, as is usual in Tcl.) Braces ({ and }) and vertical lines (|) indicate that you must choose one alternative. Dots (. . .) mean that the preceding element can be repeated.

Where it enhances the clarity, SQL commands are preceded by the prompt =>, and shell commands are preceded by the prompt \$. Normally, prompts are not shown, though.

An *administrator* is generally a person who is in charge of installing and running the server. A *user* could be anyone who is using, or wants to use, any part of the PostgreSQL system. These terms should not be interpreted too narrowly; this book does not have fixed presumptions about system administration procedures.

### Further Information

Besides the documentation, that is, this book, there are other resources about PostgreSQL:

#### Wiki

The PostgreSQL *wiki*<sup>11</sup> contains the project's *FAQ*<sup>12</sup> (Frequently Asked Questions) list, *TODO*<sup>13</sup> list, and detailed information about many more topics.

#### Web Site

The PostgreSQL *web site*<sup>14</sup> carries details on the latest release and other information to make your work or play with PostgreSQL more productive.

#### Mailing Lists

The mailing lists are a good place to have your questions answered, to share experiences with other users, and to contact the developers. Consult the PostgreSQL web site for details.

---

<sup>11</sup> <http://wiki.postgresql.org/>

<sup>12</sup> [http://wiki.postgresql.org/wiki/Frequently\\_Asked\\_Questions](http://wiki.postgresql.org/wiki/Frequently_Asked_Questions)

<sup>13</sup> <http://wiki.postgresql.org/wiki/ToDo>

<sup>14</sup> <http://www.postgresql.org/>

Yourself!

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

## **Bug Reporting Guidelines**

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

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

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

## **Identifying Bugs**

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

- A program terminates with a fatal signal or an operating system error message that would point to a problem in the program. (A counterexample might be a "disk full" message, since you have to fix that yourself.)
- A program produces the wrong output for any given input.
- A program refuses to accept valid input (as defined in the documentation).
- A program accepts invalid input without a notice or error message. But keep in mind that your idea of invalid input might be our idea of an extension or compatibility with traditional practice.

- PostgreSQL fails to compile, build, or install according to the instructions on supported platforms.

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

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

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

### **What to report**

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

The following items should be contained in every bug report:

- The exact sequence of steps *from program start-up* necessary to reproduce the problem. This should be self-contained; it is not enough to send in a bare `SELECT` statement without the preceding `CREATE TABLE` and `INSERT` statements, if the output should depend on the data in the tables. We do not have the time to reverse-engineer your database schema, and if we are supposed to make up our own data we would probably miss the problem.

The best format for a test case for SQL-related problems is a file that can be run through the `psql` frontend that shows the problem. (Be sure to not have anything in your `~/.psqlrc` start-up file.) An easy way to create this file is to use `pg_dump` to dump out the table declarations and data needed to set the scene, then add the problem query. You are encouraged to minimize the size of your example, but this is not absolutely necessary. If the bug is reproducible, we will find it either way.

If your application uses some other client interface, such as PHP, then please try to isolate the offending queries. We will probably not set up a web server to reproduce

---

your problem. In any case remember to provide the exact input files; do not guess that the problem happens for "large files" or "midsize databases", etc. since this information is too inexact to be of use.

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

**Note**

If you are reporting an error message, please obtain the most verbose form of the message. In `psql`, say `\set VERBOSITY verbose` beforehand. If you are extracting the message from the server log, set the run-time parameter `log_error_verbosity` (see "18.7. Error Reporting and Logging" section) to `verbose` so that all details are logged.

**Note**

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

- The output you expected is very important to state. If you just write "This command gives me that output." or "This is not what I expected.", we might run it ourselves, scan the output, and think it looks OK and is exactly what we expected. We should not have to spend the time to decode the exact semantics behind your commands. Especially refrain from merely saying that "This is not what SQL says/Oracle does." Digging out the correct behavior from SQL is not a fun undertaking, nor do we all know how all the other relational databases out there behave. (If your problem is a program crash, you can obviously omit this item.)
- Any command line options and other start-up options, including any relevant environment variables or configuration files that you changed from the default. Again, please provide exact information. If you are using a prepackaged distribution that starts the database server at boot time, you should try to find out how that is done.
- Anything you did at all differently from the installation instructions.
- The PostgreSQL version. You can run the command `SELECT version();` to find out the version of the server you are connected to. Most executable programs also

support a `--version` option; at least `postgres --version` and `psql --version` should work. If the function or the options do not exist then your version is more than old enough to warrant an upgrade. If you run a prepackaged version, such as RPMs, say so, including any subversion the package might have. If you are talking about a CVS snapshot, mention that, including its date and time.

If your version is older than 8.4.0 we will almost certainly tell you to upgrade. There are many bug fixes and improvements in each new release, so it is quite possible that a bug you have encountered in an older release of PostgreSQL has already been fixed. We can only provide limited support for sites using older releases of PostgreSQL; if you require more than we can provide, consider acquiring a commercial support contract.

- Platform information. This includes the kernel name and version, C library, processor, memory information, and so on. In most cases it is sufficient to report the vendor and version, but do not assume everyone knows what exactly "Debian" contains or that everyone runs on i386s. If you have installation problems then information about the toolchain on your machine (compiler, make, and so on) is also necessary.

Do not be afraid if your bug report becomes rather lengthy. That is a fact of life. It is better to report everything the first time than us having to squeeze the facts out of you. On the other hand, if your input files are huge, it is fair to ask first whether somebody is interested in looking into it. Here is an *article*<sup>15</sup> that outlines some more tips on reporting bugs.

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

When writing a bug report, please avoid confusing terminology. The software package in total is called "PostgreSQL", sometimes "Postgres" for short. If you are specifically talking about the backend server, mention that, do not just say "PostgreSQL crashes". A crash of a single backend server process is quite different from crash of the parent "postgres" process; please don't say "the server crashed" when you mean a single backend process went down, nor vice versa. Also, client programs such as the interactive frontend "psql" are completely separate from the backend. Please try to be specific about whether the problem is on the client or server side.

---

<sup>15</sup> <http://www.chiark.greenend.org.uk/~sgtatham/bugs.html>

## Where to report bugs

In general, send bug reports to the bug report mailing list at `<pgsql-bugs@postgresql.org>`. You are requested to use a descriptive subject for your email message, perhaps parts of the error message.

Another method is to fill in the bug report web-form available at the project's *web site*<sup>16</sup>. Entering a bug report this way causes it to be mailed to the `<pgsql-bugs@postgresql.org>` mailing list.

If your bug report has security implications and you'd prefer that it not become immediately visible in public archives, don't send it to `pgsql-bugs`. Security issues can be reported privately to `<security@postgresql.org>`.

Do not send bug reports to any of the user mailing lists, such as `<pgsql-sql@postgresql.org>` or `<pgsql-general@postgresql.org>`. These mailing lists are for answering user questions, and their subscribers normally do not wish to receive bug reports. More importantly, they are unlikely to fix them.

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

If you have a problem with the documentation, the best place to report it is the documentation mailing list `<pgsql-docs@postgresql.org>`. Please be specific about what part of the documentation you are unhappy with.

If your bug is a portability problem on a non-supported platform, send mail to `<pgsql-hackers@postgresql.org>`, so we (and you) can work on porting PostgreSQL to your platform.



### Note

Due to the unfortunate amount of spam going around, all of the above email addresses are closed mailing lists. That is, you need to be subscribed to a list to be allowed to post on it. (You need not be subscribed to use the bug-report web form, however.) If you would like to send mail but do not want to receive list traffic, you can subscribe and set your subscription option to `nomail`. For more information send mail to `<majordomo@postgresql.org>` with the single word `help` in the body of the message.

---

<sup>16</sup> <http://www.postgresql.org/>

# Part I.

# Tutorial

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 1.

## Getting Started

### 1.1. Installation

Before you can use PostgreSQL you need to install it, of course. It is possible that PostgreSQL is already installed at your site, either because it was included in your operating system distribution or because the system administrator already installed it. If that is the case, you should obtain information from the operating system documentation or your system administrator about how to access PostgreSQL.

If you are not sure whether PostgreSQL is already available or whether you can use it for your experimentation then you can install it yourself. Doing so is not hard and it can be a good exercise. PostgreSQL can be installed by any unprivileged user; no superuser (root) access is required.

If you are installing PostgreSQL yourself, then refer to *Chapter 15* for instructions on installation, and return to this guide when the installation is complete. Be sure to follow closely the section about setting up the appropriate environment variables.

If your site administrator has not set things up in the default way, you might have some more work to do. For example, if the database server machine is a remote machine, you will need to set the `PGHOST` environment variable to the name of the database server machine. The environment variable `PGPORT` might also have to be set. The bottom line is this: if you try to start an application program and it complains that it cannot connect to the database, you should consult your site administrator or, if that is you, the documentation to make sure that your environment is properly set up. If you did not understand the preceding paragraph then read the next section.

### 1.2. Architectural Fundamentals

Before we proceed, you should understand the basic PostgreSQL system architecture. Understanding how the parts of PostgreSQL interact will make this chapter somewhat clearer.

In database jargon, PostgreSQL uses a client/server model. A PostgreSQL session consists of the following cooperating processes (programs):

- A server process, which manages the database files, accepts connections to the database from client applications, and performs database actions on behalf of the clients. The database server program is called `postgres`.
- The user's client (frontend) application that wants to perform database operations. Client applications can be very diverse in nature: a client could be a text-oriented tool, a graphical application, a web server that accesses the database to display web pages, or a specialized database maintenance tool. Some client applications are supplied with the PostgreSQL distribution; most are developed by users.

As is typical of client/server applications, the client and the server can be on different hosts. In that case they communicate over a TCP/IP network connection. You should keep this in mind, because the files that can be accessed on a client machine might not be accessible (or might only be accessible using a different file name) on the database server machine.

The PostgreSQL server can handle multiple concurrent connections from clients. To achieve this it starts ("forks") a new process for each connection. From that point on, the client and the new server process communicate without intervention by the original `postgres` process. Thus, the master server process is always running, waiting for client connections, whereas client and associated server processes come and go. (All of this is of course invisible to the user. We only mention it here for completeness.)

### 1.3. Creating a Database

The first test to see whether you can access the database server is to try to create a database. A running PostgreSQL server can manage many databases. Typically, a separate database is used for each project or for each user.

Possibly, your site administrator has already created a database for your use. He should have told you what the name of your database is. In that case you can omit this step and skip ahead to the next section.

To create a new database, in this example named `mydb`, you use the following command:

```
$ createdb mydb
```

If this produces no response then this step was successful and you can skip over the remainder of this section.

If you see a message similar to:

```
createdb: command not found
```

then PostgreSQL was not installed properly. Either it was not installed at all or your shell's search path was not set to include it. Try calling the command with an absolute path instead:

```
$ /usr/local/pgsql/bin/createdb mydb
```

The path at your site might be different. Contact your site administrator or check the installation instructions to correct the situation.

Another response could be this:

```
createdb: could not connect to database postgres: could not connect to server:
No such file or directory
Is the server running locally and accepting
connections on Unix domain socket "/tmp/.s.PGSQL.5432"?
```

This means that the server was not started, or it was not started where `createdb` expected it. Again, check the installation instructions or consult the administrator.

Another response could be this:

```
createdb: could not connect to database postgres: FATAL:  role "joe" does not exist
```

where your own login name is mentioned. This will happen if the administrator has not created a PostgreSQL user account for you. (PostgreSQL user accounts are distinct from operating system user accounts.) If you are the administrator, see *Chapter 20* for help creating accounts. You will need to become the operating system user under which PostgreSQL was installed (usually `postgres`) to create the first user account. It could also be that you were assigned a PostgreSQL user name that is different from your operating system user name; in that case you need to use the `-U` switch or set the `PGUSER` environment variable to specify your PostgreSQL user name.

If you have a user account but it does not have the privileges required to create a database, you will see the following:

```
createdb: database creation failed: ERROR:  permission denied to create database
```

Not every user has authorization to create new databases. If PostgreSQL refuses to create databases for you then the site administrator needs to grant you permission to create databases. Consult your site administrator if this occurs. If you installed PostgreSQL yourself then you should log in for the purposes of this tutorial under the user account that you started the server as<sup>1</sup>.

You can also create databases with other names. PostgreSQL allows you to create any number of databases at a given site. Database names must have an alphabetic first character

---

<sup>1</sup> As an explanation for why this works: PostgreSQL user names are separate from operating system user accounts. When you connect to a database, you can choose what PostgreSQL user name to connect as; if you don't, it will default to the same name as your current operating system account. As it happens, there will always be a PostgreSQL user account that has the same name as the operating system user that started the server, and it also happens that that user always has permission to create databases. Instead of logging in as that user you can also specify the `-U` option everywhere to select a PostgreSQL user name to connect as.

and are limited to 63 characters in length. A convenient choice is to create a database with the same name as your current user name. Many tools assume that database name as the default, so it can save you some typing. To create that database, simply type:

```
$ createdb
```

If you do not want to use your database anymore you can remove it. For example, if you are the owner (creator) of the database `mydb`, you can destroy it using the following command:

```
$ dropdb mydb
```

(For this command, the database name does not default to the user account name. You always need to specify it.) This action physically removes all files associated with the database and cannot be undone, so this should only be done with a great deal of forethought.

More about `createdb` and `dropdb` can be found in *createdb* (see "*createdb command*" section) and *dropdb* (see "*dropdb command*" section) respectively.

## 1.4. Accessing a Database

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

- Running the PostgreSQL interactive terminal program, called *psql*, which allows you to interactively enter, edit, and execute SQL commands.
- Using an existing graphical frontend tool like pgAdmin or an office suite with ODBC or JDBC support to create and manipulate a database. These possibilities are not covered in this tutorial.
- Writing a custom application, using one of the several available language bindings. These possibilities are discussed further in *Part IV*.

You probably want to start up `psql` to try the examples in this tutorial. It can be activated for the `mydb` database by typing the command:

```
$ psql mydb
```

If you do not supply the database name then it will default to your user account name. You already discovered this scheme in the previous section using `createdb`.

In `psql`, you will be greeted with the following message:

```
psql (8.4.0)
Type "help" for help.
```

```
mydb=>
```

The last line could also be:

```
mydb=#
```

That would mean you are a database superuser, which is most likely the case if you installed PostgreSQL yourself. Being a superuser means that you are not subject to access controls. For the purposes of this tutorial that is not important.

If you encounter problems starting `psql` then go back to the previous section. The diagnostics of `createdb` and `psql` are similar, and if the former worked the latter should work as well.

The last line printed out by `psql` is the prompt, and it indicates that `psql` is listening to you and that you can type SQL queries into a work space maintained by `psql`. Try out these commands:

```
mydb=> SELECT version();
                version
-----
 PostgreSQL 8.4.0 on i586-pc-linux-gnu, compiled by GCC 2.96, 32-bit
(1 row)

mydb=> SELECT current_date;
      date
-----
 2002-08-31
(1 row)

mydb=> SELECT 2 + 2;
?column?
-----
         4
(1 row)
```

The `psql` program has a number of internal commands that are not SQL commands. They begin with the backslash character, `"\"`. Some of these commands were listed in the welcome message. For example, you can get help on the syntax of various PostgreSQL SQL commands by typing:

```
mydb=> \h
```

To get out of `psql`, type:

```
mydb=> \q
```

and `psql` will quit and return you to your command shell. (For more internal commands, type `\?` at the `psql` prompt.) The full capabilities of `psql` are documented in [psql](#). If PostgreSQL is installed correctly you can also type `man psql` at the operating system shell prompt to see the documentation. In this tutorial we will not use these features explicitly, but you can use them yourself when it is helpful.

# 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

- \$ .....66
- \* .....128
- abs .....193
- acos.....193
- age .....230
- aggregate function .....41
  - built-in .....265
  - invocation.....69
- alias
  - for table name in query .....39
  - in the FROM clause.....121
  - in the select list .....128
- ALL .....270, 274
- AND (operator) .....190
- any .....188, 265, 270, 274
- anyarray .....188
- anyelement.....188
- anyenum.....188
- anynonarray .....188
- arbitrary precision numbers .....139
- area .....242
- ARRAY .....73, 173
  - accessing.....176
  - constant .....174
  - constructor .....73
  - declaration .....173
  - determination of result type.....305
  - I/O .....181
  - modifying.....177
  - searching .....180
- array\_agg.....265
- array\_append .....264
- array\_cat.....264
- array\_dims .....264
- array\_fill .....264
- array\_length.....264
- array\_lower.....264
- array\_ndims.....264
- array\_prepend .....264
- array\_to\_string .....264
- array\_upper .....264
- ascii.....196
- asin .....193
- AT TIME ZONE .....238
- atan.....193
- atan2.....193
- autocommit bulk-loading data.....384
- auto-increment ..... *See* serial
- average .....41, 265
- backslash escapes.....58
- backup .....289
- BETWEEN.....191
- BETWEEN SYMMETRIC.....191
- bigint .....61, 139
- bigserial .....142
- binary data .....146
  - functions.....205
- binary string
  - concatenation.....205
  - length.....205

bit string	check constraint.....80
constant .....61	chr.....196
data type.....167	cid.....187
bit strings	cidr .....165
functions.....207	circle.....164
bit_and.....265	clock_timestamp .....230
bit_length .....196	CLUSTER
bit_or.....265	of databases..... <i>See</i> database cluster
bitmap scan.....312	cmax.....87
bool_and.....265	cmin.....87
bool_or.....265	COALESCE.....263
Boolean	col_description .....280
data type.....159	column.....34, 77
operators ..... <i>See</i> operators, logical	adding.....89
box (data type).....163	removing.....90
B-tree..... <i>See</i> index	renaming .....92
btrim.....196	system column.....87
bytea.....146	column data type
CASCADE	changing.....91
foreign key action.....84	column reference.....67
with DROP .....111	COMMENT
CASE.....261	about database objects.....280
determination of result type.....305	in SQL.....64
case sensitivity	common table expression ..... <i>See</i> WITH
of SQL commands.....56	comparison
cbrt .....193	operators .....191
ceiling.....193	row-wise.....274
center.....193	subquery result row .....270
char.....144	composite type .....182
char_length .....196	constant .....184
character .....144	constructor .....74
character string	concurrency .....361
concatenation.....196	conditional expression .....261
constant .....58	configuration
data types .....144	of the server
length.....196	functions.....289
character varying .....144	conjunction.....190

constant .....	57	data type.....	137
constraint.....	80	category .....	296
adding.....	83	constant .....	62
check .....	80	conversion.....	296
foreign key .....	84	enumerated (enum) .....	160
name.....	80	numeric.....	138
NOT NULL .....	82	type cast.....	71
primary key.....	84	database	
removing.....	91	reating.....	30
unique.....	83	database cluster .....	34
constraint exclusion .....	107	date.....	148, 150
convert .....	196	current .....	239
convert_from .....	196	nstants.....	153
convert_to.....	196	output format .....	153
COPY .....	36	.....	<i>See also formatting</i>
correlation .....	265	date_part .....	230, 233
cos.....	193	date_trunc .....	230, 237
cot.....	193	decode.....	196
count .....	41	default value .....	79
covariance		changing.....	91
population.....	265	degrees.....	193
sample.....	265	delay.....	241
CREATE TABLE.....	35	DELETE .....	43, 115
createdb .....	30	deleting.....	115, 242, 269
cross join.....	117	dirty read.....	361
cstring .....	188	disjunction.....	190
ctid.....	87	DISTINCT .....	37, 129
cume_dist.....	269	div .....	193
current_catalog.....	280	document	
current_database.....	280	text search .....	322
current_date.....	230	dollar quoting.....	60
current_schema .....	280	double precision.....	141
current_time.....	230	DROP TABLE .....	35
current_timestamp.....	230	DTD.....	171
current_user .....	280	duplicate.....	37
currval.....	259	duplicates .....	129
data cluster.....	<i>See database cluster</i>	encode.....	196

enumerated types.....	160	has_any_column_privilege.....	280
escape string syntax.....	58	has_column_privilege.....	280
every.....	265	has_database_privilege.....	280
EXCEPT.....	130	has_foreign_data_wrapper_privilege...280	
EXISTS.....	270	has_function_privilege.....	280
exp.....	193	has_language_privilege.....	280
EXPLAIN.....	374	has_schema_privilege.....	280
expression		has_server_privilege.....	280
order of evaluation.....	76	has_table_privilege.....	280
syntax.....	66	has_tablespace_privilege.....	280
extract.....	230, 233	hash.....	<i>See index</i>
FALSE.....	159	HAVING.....	41, 125
field selection.....	68	height.....	242
first_value.....	269	hierarchical database.....	34
floating point.....	141	history	
floor.....	193	of PostgreSQL.....	19
foreign key.....	44, 84	identifier	
format_type.....	280	length.....	56
formatting.....	224	syntax of.....	56
full text search.....	321	IFNULL.....	263
data types.....	168	IN.....	270, 271
functions and operators.....	168	index.....	307
function.....	190	and ORDER BY.....	311
in the FROM clause.....	123	B-tree.....	308
invocation.....	69	combining multiple indexes.....	312
type resolution in an invocation.....	301	examining usage.....	319
generate_series.....	277	GIN.....	308
generate_subscripts.....	277	text search.....	355
get_bit.....	205	GiST.....	308
get_byte.....	205	text search.....	355
GIN.....	<i>See index</i>	hash.....	308
GiST.....	<i>See index</i>	locks.....	373
GREATEST.....	263	multicolumn.....	310
determination of result type.....	305	on expressions.....	314
GROUP BY.....	41, 125	partial.....	315
grouping.....	125	unique.....	313
GUID.....	170	inet (data type).....	165

- 
- inet\_client\_addr.....280
  - inet\_client\_port.....280
  - inet\_server\_addr .....280
  - inet\_server\_port .....280
  - inheritance.....51, 98
  - initcap .....196
  - INSERT .....36, 113
  - inserting.....113
  - int2..... *See* smallint
  - int4..... *See* integer
  - int8..... *See* bigint
  - integer .....61, 139
  - internal.....188
  - INTERSECT .....130
  - interval.....148, 156
    - output format .....158
      - ..... *See also* formatting
  - IS DISTINCT FROM .....191, 274
  - IS DOCUMENT .....254
  - IS FALSE.....191
  - IS NOT DISTINCT FROM .....191, 274
  - IS NOT FALSE.....191
  - IS NOT NULL.....191
  - IS NOT TRUE .....191
  - IS NOT UNKNOWN .....191
  - IS NULL.....191
  - IS TRUE .....191
  - IS UNKNOWN .....191
  - isclosed .....242
  - isfinite .....230
  - ISNULL.....191
  - isopen.....242
  - join.....39, 117
    - controlling the order.....381
    - cross .....117
    - left.....117
    - natural.....117
    - outer.....39, 117
    - right.....117
    - self .....39
  - justify\_days.....230
  - justify\_hours .....230
  - justify\_interval.....230
  - key word
    - syntax of .....56
  - label..... *See* alias
  - lag.....269
  - language\_handler .....188
  - last\_value .....269
  - lastval.....259
  - lead.....269
  - LEAST.....263
    - determination of result type.....305
  - left join.....117
  - length.....242
    - of a binary string
      - ..... *See* binary strings, length
    - of a character string
      - ..... *See* character string, length
  - length(tsvector) .....334
  - LIKE .....208
  - LIMIT .....131
  - line segment.....163
  - linear regression.....265
  - ln .....193
  - localtime .....230
  - localtimestamp .....230
  - lock.....366
    - advisory.....371
  - log.....193
  - lower .....196
  - lpad .....196
  - lseg .....163
  - ltrim.....196
-

MAC address .....	<i>See</i> macaddr	number	
macaddr (data type) .....	166	constant .....	61
max .....	41	numeric .....	61
md5 .....	196	numeric (data type) .....	139
min .....	41	numnode .....	335
mod .....	193	NVL .....	263
MVCC .....	361	obj_description .....	280
name		object identifier	
qualified .....	94	data type .....	187
syntax of .....	56	object-oriented database .....	34
unqualified .....	95	octet_length .....	196
NaN .....	<i>See</i> not a number	OFFSET .....	131
natural join .....	117	oid .....	187
negation .....	190	column .....	87
network		ONLY .....	117
data types .....	165	opaque .....	188
nextval .....	259	operator .....	190
nonrepeatable read .....	361	invocation .....	68
NOT (operator) .....	190	logical .....	190
not a number		precedence .....	64
double precision .....	141	syntax .....	63
numeric (data type) .....	139	type resolution in an invocation .....	298
NOT IN .....	270, 274	operator class .....	318
NOTNULL .....	191	operator family .....	318
not-null constraint .....	82	operators	
now .....	230	logical .....	190
npoints .....	242	OR (operator) .....	190
nth_value .....	269	ORDER BY .....	37, 130
ntile .....	269	outer join .....	117
null value		OVER clause .....	70
comparing .....	191	overlay .....	196
default value .....	79	parameter	
in DISTINCT .....	129	syntax .....	67
with check constraints .....	80	parenthesis .....	66
with unique constraints .....	83	partitioning .....	102
NULLIF .....	263	path (data type) .....	164
		pattern matching .....	207

---

pclose .....	242	pg_read_file .....	289
percent_rank .....	269	pg_relation_size .....	289
permission.....	<i>See</i> privilege	pg_reload_conf.....	289
pg_advisory_lock.....	289	pg_rotate_logfile .....	289
pg_advisory_lock_shared .....	289	pg_size_pretty .....	289
pg_advisory_unlock .....	289	pg_sleep.....	241
pg_advisory_unlock_all.....	289	pg_start_backup.....	289
pg_advisory_unlock_shared .....	289	pg_stat_file.....	289
pg_cancel_backend.....	289	pg_statistic .....	380
pg_client_encoding.....	196	pg_stats.....	380
pg_column_size.....	196	pg_stop_backup .....	289
pg_conf_load_time.....	280	pg_switch_xlog.....	289
pg_conversion_is_visible.....	280	pg_table_is_visible.....	280
pg_current_xlog_insert_location .....	289	pg_tablespace_databases .....	280
pg_current_xlog_location .....	289	pg_tablespace_size.....	289
pg_database_size .....	289	pg_terminate_backend.....	289
pg_function_is_visible.....	280	pg_total_relation_size .....	289
pg_get_constraintdef .....	280	pg_try_advisory_lock.....	289
pg_get_expr .....	280	pg_try_advisory_lock_shared .....	289
pg_get_function_arguments.....	280	pg_ts_config_is_visible .....	280
pg_get_function_identity_arguments...280		pg_ts_dict_is_visible.....	280
pg_get_function_result.....	280	pg_ts_parser_is_visible .....	280
pg_get_functiondef.....	280	pg_ts_template_is_visible.....	280
pg_get_indexdef.....	280	pg_type_is_visible .....	280
pg_get_keywords.....	280	pg_typeof .....	280
pg_get_ruledef.....	280	pg_xlogfile_name.....	289
pg_get_serial_sequence.....	280	pg_xlogfile_name_offset .....	289
pg_get_triggerdef.....	280	phantom read .....	361
pg_get_userbyid.....	280	pi .....	193
pg_get_viewdef.....	280	plainto_tsquery .....	329
pg_has_role.....	280	point.....	163
pg_is_other_temp_schema .....	280	polygon.....	164
pg_ls_dir.....	289	popen .....	242
pg_my_temp_schema.....	280	position.....	196
pg_opclass_is_visible .....	280	POSTGRES.....	29
pg_operator_is_visible .....	280	power .....	193
pg_postmaster_start_time .....	280	predicate locking.....	365

---

primary key.....	84	regular expression.....	209, 210
privilege.....	92	.....	<i>See also pattern matching</i>
for schemas .....	96	relation.....	34
querying .....	280	relational database .....	34
psql.....	32	repeat .....	196
qualified name.....	94	replace.....	196
query .....	37, 116	RESTRICT	
query plan .....	374	foreign key action .....	84
querytree .....	335	with DROP .....	111
quotation marks		right join .....	117
and identifiers.....	56	round .....	193
escaping.....	58	row .....	34, 74, 77
quote_ident .....	196	row type .....	182
quote_literal .....	196	constructor .....	74
quote_nullable .....	196	row_number .....	269
radians .....	193	row-wise comparison .....	274
radius .....	242	rpad.....	196
random .....	193	rtrim .....	196
rank .....	269	schema .....	93
real.....	141	creating.....	94
record.....	188	current .....	95, 280
rectangle .....	163	public .....	95
referential integrity .....	44, 84	removing.....	94
regclass .....	187	search path.....	95
regconfig.....	187	current .....	280
regdictionary.....	187	search_path.....	95
regexp_matches.....	210	SELECT.....	37, 116
regexp_replace.....	210	select list .....	128
regexp_split_to_array.....	210	sequence .....	259
regexp_split_to_table.....	210	and serial type .....	142
regoper.....	210	serial.....	142
regoperator.....	210	serial4.....	142
regproc.....	210	serial8.....	142
regprocedure .....	210	serializability .....	365
regression intercept .....	265	SET .....	289
regression slope.....	265	set difference.....	130
regtype.....	187	set intersection.....	130

- 
- set operation .....130
  - set returning functions
    - functions.....277
  - set union .....130
  - set\_bit.....205
  - set\_byte.....205
  - setseed .....193
  - setval.....259
  - setweight .....334
  - shobj\_description.....280
  - SHOW.....289
  - sign.....193
  - signal
    - backend processes.....289
  - SIMILAR TO .....209
  - sin .....193
  - sleep .....241
  - smallint .....139
  - SOME.....265, 270, 274
  - sorting.....130
  - split\_part .....196
  - sqrt.....193
  - standard deviation.....265
    - population.....265
    - sample.....265
  - statement\_timestamp .....230
  - statistics .....265
    - of the planner.....380
  - string ..... *See* character string
  - string\_to\_array .....264
  - strip .....334
  - strpos.....196
  - subquery .....41, 72, 122, 270
  - subscript .....67
  - substr .....196
  - substring.....196, 205, 209, 210
  - sum.....41
  - superuser.....32
  - suppress\_redundant\_updates\_trigger..295
  - syntax
    - SQL.....55
  - system catalog
    - schema .....97
  - table.....34, 77
    - creating .....77
    - inheritance .....98
    - modifying.....89
    - partitioning .....102
    - removing .....77
    - renaming .....92
  - table expression.....117
  - table function.....123
  - tableoid .....87
  - tan.....193
  - text.....144
  - text search .....321
    - data types .....168
    - functions and operators .....168
    - indexes.....355
  - tid .....187
  - time .....148, 151
    - constants.....153
    - current .....239
    - output format .....153
      - ..... *See also* formatting
  - time span.....148
  - time with time zone .....148, 151
  - time without time zone .....148, 151
  - time zone .....154
    - conversion.....238
  - timeofday .....230
  - timestamp.....148, 152
  - timestamp with time zone .....148, 152
  - timestamp without time zone .....148, 152
-

to_ascii .....	196	txid_snapshot_xmax .....	280
to_char .....	224	txid_snapshot_xmin .....	280
to_date .....	224	txid_visible_in_snapshot .....	280
to_hex .....	196	type .....	<i>See data type</i>
to_number .....	224	type cast .....	61, 71
to_timestamp .....	224	UESCAPE .....	56, 59
to_tsquery .....	329	Unicode escape	
to_tsvector .....	328	in identifiers .....	56
token .....	55	in string constants .....	59
transaction .....	45	UNION .....	130
transaction isolation .....	361	determination of result type .....	305
transaction isolation level .....	361	unique constraint .....	83
read committed .....	362	unnest .....	264
serializable .....	364	unqualified name .....	95
transaction_timestamp .....	230	UPDATE .....	43, 114
translate .....	196	updating .....	114
trigger .....	188	upper .....	196
for updating a derived tsvector column		user	
.....	338	current .....	280
trim .....	196	UUID .....	170
TRUE .....	159	value expression .....	66
trunc .....	193	VALUES .....	132
ts_debug .....	351	determination of result type .....	305
ts_headline .....	332	varchar .....	144
ts_lexize .....	354	variance .....	265
ts_parse .....	353	population .....	265
ts_rank .....	330	sample .....	265
ts_rank_cd .....	330	version .....	32, 280
ts_rewrite .....	336	view .....	44
ts_stat .....	339	void .....	188
ts_token_type .....	353	WHERE .....	124
tsquery (data type) .....	169	width .....	242
tsvector (data type) .....	168	width_bucket .....	193
tsvector concatenation .....	334	window function .....	48
txid_current .....	280	built-in .....	269
txid_current_snapshot .....	280	invocation .....	70
txid_snapshot_xip .....	280	order of execution .....	127

---

WITH .....	133	xmlcomment .....	250
in SELECT .....	133	xmlconcat .....	251
xid.....	187	xmlelement.....	251
xmax.....	87	xmlforest.....	252
xmin .....	87	xmlparse .....	171
XML .....	171	xmlpi .....	253
XML export .....	255	xmlroot .....	253
XML option.....	171	xmlserialize .....	171
xmlagg .....	254, 265	XPath.....	254