

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 SQL:2008 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



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



PostgreSQL
www.postgresql.org



PostgreSQL 9.0

Internals and Appendixes

Volume V



By The PostgreSQL Global Development Group

PostgreSQL 9.0

Internals and Appendixes

5



Linbrary[™] - Linux Documentation Library
www.linbrary.com



PostgreSQL 9.0
Official Documentation

Volume V
Internals and Appendixes



Fultus™ Books

PostgreSQL



PostgreSQL 9.0 Official Documentation

Volume V

Internals and Appendixes

ISBN-10: 1-59682-250-3

ISBN-13: 978-1-59682-250-4

Copyright © 1996-2011 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.....	16
License.....	18
Abstract.....	19
Part VII Internals.....	20
Chapter 44. Overview of PostgreSQL Internals.....	21
44.1. The Path of a Query	21
44.2. How Connections are Established	22
44.3. The Parser Stage	22
44.3.1. Parser.....	23
44.3.2. Transformation Process	23
44.4. The PostgreSQL Rule System	24
44.5. Planner/Optimizer	24
44.5.1. Generating Possible Plans	25
44.6. Executor.....	26
Chapter 45. System Catalogs.....	28
45.1. Overview	28
45.2. pg_aggregate.....	30
45.3. pg_am.....	30
45.4. pg_amop.....	31
45.5. pg_amproc.....	32
45.6. pg_attrdef.....	32
45.7. pg_attribute.....	33
45.8. pg_authid.....	34
45.9. pg_auth_members.....	35
45.10. pg_cast.....	36
45.11. pg_class.....	37
45.12. pg_constraint.....	39
45.13. pg_conversion.....	40
45.14. pg_database	41
45.15. pg_db_role_setting.....	42
45.16. pg_default_acl	42
45.17. pg_depend.....	43

45.18. pg_description.....	44
45.19. pg_enum.....	45
45.20. pg_foreign_data_wrapper	45
45.21. pg_foreign_server.....	45
45.22. pg_index.....	46
45.23. pg_inherits.....	47
45.24. pg_language.....	47
45.25. pg_largeobject.....	48
45.26. pg_largeobject_metadata	49
45.27. pg_namespace.....	49
45.28. pg_opclass.....	49
45.29. pg_operator.....	50
45.30. pg_opfamily.....	51
45.31. pg_pltemplate	51
45.32. pg_proc.....	52
45.33. pg_rewrite.....	54
45.34. pg_shdepend.....	55
45.35. pg_shdescription.....	56
45.36. pg_statistic.....	56
45.37. pg_tablespace	58
45.38. pg_trigger.....	58
45.39. pg_ts_config.....	59
45.40. pg_ts_config_map.....	60
45.41. pg_ts_dict.....	60
45.42. pg_ts_parser.....	61
45.43. pg_ts_template.....	61
45.44. pg_type.....	62
45.45. pg_user_mapping.....	65
45.46. System Views.....	66
45.47. pg_cursors.....	67
45.48. pg_group	67
45.49. pg_indexes.....	68
45.50. pg_locks	68
45.51. pg_prepared_statements.....	70
45.52. pg_prepared_xacts.....	71
45.53. pg_roles	72
45.54. pg_rules	72
45.55. pg_settings	73
45.56. pg_shadow.....	74

Table of Contents

45.57. pg_stats.....	74
45.58. pg_tables.....	76
45.59. pg_timezone_abbrevs.....	76
45.60. pg_timezone_names.....	77
45.61. pg_user.....	77
45.62. pg_user_mappings.....	78
45.63. pg_views.....	78
Chapter 46. Frontend/Backend Protocol.....	79
46.1. Overview.....	79
46.1.1. Messaging Overview.....	80
46.1.2. Extended Query Overview.....	80
46.1.3. Formats and Format Codes.....	81
46.2. Message Flow.....	81
46.2.1. Start-Up.....	82
46.2.2. Simple Query.....	84
46.2.3. Extended Query.....	86
46.2.4. Function Call.....	90
46.2.5. COPY Operations.....	91
46.2.6. Asynchronous Operations.....	92
46.2.7. Cancelling Requests in Progress.....	93
46.2.8. Termination.....	94
46.2.9. SSL Session Encryption.....	94
46.3. Streaming Replication Protocol.....	95
46.4. Message Data Types.....	96
46.5. Message Formats.....	97
46.6. Error and Notice Message Fields.....	114
46.7. Summary of Changes since Protocol 2.0.....	115
Chapter 47. PostgreSQL Coding Conventions.....	117
47.1. Formatting.....	117
47.2. Reporting Errors Within the Server.....	118
47.3. Error Message Style Guide.....	120
47.3.1. What goes where.....	121
47.3.2. Formatting.....	121
47.3.3. Quotation marks.....	121
47.3.4. Use of quotes.....	122
47.3.5. Grammar and punctuation.....	122
47.3.6. Upper case vs. lower case.....	122
47.3.7. Avoid passive voice.....	123
47.3.8. Present vs past tense.....	123

47.3.9. Type of the object.....	123
47.3.10. Brackets.....	123
47.3.11. Assembling error messages.....	124
47.3.12. Reasons for errors.....	124
47.3.13. Function names.....	124
47.3.14. Tricky words to avoid.....	124
47.3.15. Proper spelling.....	125
47.3.16. Localization.....	125
Chapter 48. Native Language Support.....	126
48.1. For the Translator.....	126
48.1.1. Requirements.....	126
48.1.2. Concepts.....	126
48.1.3. Creating and maintaining message catalogs.....	128
48.1.4. Editing the PO files.....	128
48.2. For the Programmer.....	129
48.2.1. Mechanics.....	129
48.2.2. Message-writing guidelines.....	131
Chapter 49. Writing A Procedural Language Handler.....	133
Chapter 50. Genetic Query Optimizer.....	137
50.1. Query Handling as a Complex Optimization Problem.....	137
50.2. Genetic Algorithms.....	137
50.3. Genetic Query Optimization (GEQO) in PostgreSQL.....	138
50.3.1. Generating Possible Plans with GEQO.....	139
50.3.2. Future Implementation Tasks for PostgreSQL GEQO.....	140
50.4. Further Reading.....	140
Chapter 51. Index Access Method Interface Definition.....	141
51.1. Catalog Entries for Indexes.....	141
51.2. Index Access Method Functions.....	143
51.3. Index Scanning.....	147
51.4. Index Locking Considerations.....	149
51.5. Index Uniqueness Checks.....	150
51.6. Index Cost Estimation Functions.....	152
Chapter 52. GiST Indexes.....	155
52.1. Introduction.....	155
52.2. Extensibility.....	155
52.3. Implementation.....	156
52.4. Examples.....	163
52.5. Crash Recovery.....	163

Table of Contents

Chapter 53. GIN Indexes	165
53.1. Introduction	165
53.2. Extensibility	165
53.3. Implementation	167
53.3.1. GIN fast update technique	167
53.3.2. Partial match algorithm	168
53.4. GIN tips and tricks	168
53.5. Limitations	169
53.6. Examples	170
Chapter 54. Database Physical Storage	171
54.1. Database File Layout	171
54.2. TOAST	173
54.3. Free Space Map	175
54.4. Visibility Map	176
54.5. Database Page Layout	176
Chapter 55. BKI Backend Interface	180
55.1. BKI File Format	180
55.2. BKI Commands	180
55.3. Structure of the Bootstrap BKI File	182
55.4. Example	182
Chapter 56. How the Planner Uses Statistics	184
56.1. Row Estimation Examples	184
Part VIII. Appendixes	191
Appendix A. PostgreSQL Error Codes	192
Appendix B. Date/Time Support	201
B.1. Date/Time Input Interpretation	201
B.2. Date/Time Key Words	202
B.3. Date/Time Configuration Files	203
B.4. History of Units	205
Appendix C. SQL Key Words	207
Appendix D. SQL Conformance	238
D.1. Supported Features	239
D.2. Unsupported Features	251
Appendix E. Release Notes	262
E.1. Release 9.0.3	262
E.1.1. Migration to Version 9.0.3	262
E.1.2. Changes	262
E.2. Release 9.0.2	264

E.2.1.	Migration to Version 9.0.2	264
E.2.2.	Changes.....	264
E.3.	Release 9.0.1	267
E.3.1.	Migration to Version 9.0.1	267
E.3.2.	Changes.....	268
E.4.	Release 9.0	269
E.4.1.	Overview	269
E.4.2.	Migration to Version 9.0.....	270
E.4.2.1.	Server Settings.....	270
E.4.2.2.	Queries	270
E.4.2.3.	Data Types.....	271
E.4.2.4.	Object Renaming.....	271
E.4.2.5.	PL/pgSQL	272
E.4.2.6.	Other Incompatibilities	273
E.4.3.	Changes.....	273
E.4.3.1.	Server.....	273
E.4.3.2.	Queries	277
E.4.3.3.	Object Manipulation.....	277
E.4.3.4.	Utility Operations.....	279
E.4.3.5.	Data Types.....	281
E.4.3.6.	Functions.....	282
E.4.3.7.	Server-Side Languages.....	284
E.4.3.8.	Client Applications.....	286
E.4.3.9.	Development Tools	288
E.4.3.10.	Build Options.....	289
E.4.3.11.	Source Code	290
E.4.3.12.	Contrib.....	293
Appendix F.	Additional Supplied Modules.....	295
F.1.	adminpack.....	296
F.1.1.	Functions implemented	296
F.2.	auto_explain.....	296
F.2.1.	Configuration parameters	297
F.2.2.	Example	298
F.2.3.	Author	298
F.3.	btree_gin.....	298
F.3.1.	Example usage	299
F.3.2.	Authors	299
F.4.	btree_gist.....	299
F.4.1.	Example usage	299

Table of Contents

F.4.2.	Authors.....	300
F.5.	chkpass.....	300
F.5.1.	Author	301
F.6.	citext	301
F.6.1.	Rationale.....	301
F.6.2.	How to Use It.....	302
F.6.3.	String Comparison Behavior.....	302
F.6.4.	Limitations.....	303
F.6.5.	Author	303
F.7.	cube	303
F.7.1.	Syntax	303
F.7.2.	Precision.....	304
F.7.3.	Usage	304
F.7.4.	Defaults	306
F.7.5.	Notes.....	306
F.7.6.	Credits	307
F.8.	dblink	307
	dblink_connect	307
	dblink_connect_u	309
	dblink_disconnect	310
	dblink	311
	dblink_exec	314
	dblink_open	315
	dblink_fetch	317
	dblink_close	319
	dblink_get_connections.....	320
	dblink_error_message	320
	dblink_send_query	321
	dblink_is_busy.....	322
	dblink_get_notify	322
	dblink_get_result.....	323
	dblink_cancel_query.....	325
	dblink_get_pkey.....	326
	dblink_build_sql_insert.....	327
	dblink_build_sql_delete	328
	dblink_build_sql_update	329
F.9.	dict_int.....	331
F.9.1.	Configuration	331
F.9.2.	Usage	331

F.10.	dict_xsyn	332
F.10.1.	Configuration	332
F.10.2.	Usage	332
F.11.	earthdistance	333
F.11.1.	Cube-based earth distances	334
F.11.2.	Point-based earth distances	335
F.12.	fuzzystrmatch	335
F.12.1.	Soundex	335
F.12.2.	Levenshtein	336
F.12.3.	Metaphone	336
F.12.4.	Double Metaphone	337
F.13.	hstore	337
F.13.1.	hstore External Representation	337
F.13.2.	hstore Operators and Functions	338
F.13.3.	Indexes	341
F.13.4.	Examples	341
F.13.5.	Statistics	342
F.13.6.	Compatibility	343
F.13.7.	Authors	343
F.14.	intagg	343
F.14.1.	Functions	343
F.14.2.	Sample Uses	344
F.15.	intarray	345
F.15.1.	intarray Functions and Operators	345
F.15.2.	Index Support	346
F.15.3.	Example	347
F.15.4.	Benchmark	347
F.15.5.	Authors	347
F.16.	isn	347
F.16.1.	Data types	347
F.16.2.	Casts	348
F.16.3.	Functions and Operators	349
F.16.4.	Examples	350
F.16.5.	Bibliography	350
F.16.6.	Author	351
F.17.	lo	351
F.17.1.	Rationale	351
F.17.2.	How to Use It	352
F.17.3.	Limitations	352

Table of Contents

F.17.4. Author	352
F.18. ltree.....	352
F.18.1. Definitions.....	352
F.18.2. Operators and Functions	354
F.18.3. Indexes.....	356
F.18.4. Example.....	356
F.18.5. Authors.....	358
F.19. oid2name	359
F.19.1. Overview.....	359
F.19.2. oid2name Options	359
F.19.3. Examples.....	360
F.19.4. Limitations.....	362
F.19.5. Author	363
F.20. pageinspect.....	363
F.20.1. Functions.....	363
F.21. passwordcheck	365
F.22. pg_archivecleanup	365
F.22.1. Usage	366
F.22.2. pg_archivecleanup Options	366
F.22.3. Examples.....	367
F.22.4. Supported server versions.....	367
F.22.5. Author	367
F.23. pgbench.....	367
F.23.1. Overview.....	368
F.23.2. pgbench Initialization Options	368
F.23.3. pgbench Benchmarking Options	369
F.23.4. pgbench Common Options	370
F.23.5. What is the "transaction" actually performed in pgbench?	371
F.23.6. Custom Scripts	371
F.23.7. Per-transaction logging.....	373
F.23.8. Good Practices.....	373
F.24. pg_buffercache.....	374
F.24.1. The pg_buffercache view	374
F.24.2. Sample output	375
F.24.3. Authors.....	376
F.25. pgcrypto.....	376
F.25.1. General hashing functions.....	376
F.25.1.1.digest ()	376
F.25.1.2.hmac ()	376

F.25.2. Password hashing functions	376
F.25.2.1.crypt()	377
F.25.2.2.gen_salt()	377
F.25.3. PGP encryption functions	379
F.25.3.1.pgp_sym_encrypt()	380
F.25.3.2.pgp_sym_decrypt()	380
F.25.3.3.pgp_pub_encrypt()	380
F.25.3.4.pgp_pub_decrypt()	380
F.25.3.5.pgp_key_id()	381
F.25.3.6.armor(),dearmor()	381
F.25.3.7.Options for PGP functions	381
F.25.3.8.Generating PGP keys with GnuPG	383
F.25.3.9.Limitations of PGP code	384
F.25.4. Raw encryption functions	384
F.25.5. Random-data functions	385
F.25.6. Notes.....	385
F.25.6.1.Configuration	385
F.25.6.2.NULL handling.....	386
F.25.6.3.Security limitations.....	386
F.25.6.4.Useful reading.....	386
F.25.6.5.Technical references	387
F.25.7. Author	387
F.26. pg_freemap	388
F.26.1. Functions.....	388
F.26.2. Sample output.....	389
F.26.3. Author	389
F.27. pgrowlocks.....	389
F.27.1. Overview	389
F.27.2. Sample output.....	390
F.27.3. Author	390
F.28. pg_standby.....	390
F.28.1. Usage.....	391
F.28.2. pg_standby Options	392
F.28.3. Examples.....	393
F.28.4. Supported server versions.....	394
F.28.5. Author	394
F.29. pg_stat_statements.....	394
F.29.1. The pg_stat_statements view.....	394
F.29.2. Functions.....	395

Table of Contents

F.29.3. Configuration parameters	396
F.29.4. Sample output	397
F.29.5. Author	397
F.30. pgstattuple.....	397
F.30.1. Functions.....	398
F.30.2. Authors.....	399
F.31. pg_trgm	400
F.31.1. Trigram (or Trigraph) Concepts	400
F.31.2. Functions and Operators	400
F.31.3. Index Support.....	401
F.31.4. Text Search Integration	401
F.31.5. References	402
F.31.6. Authors.....	402
F.32. pg_upgrade	402
F.32.1. Supported Versions	402
F.32.2. pg_upgrade Options	402
F.32.3. Upgrade Steps	404
F.32.4. Limitations in Migrating from PostgreSQL 8.3	407
F.32.5. Notes.....	408
F.33. seg.....	408
F.33.1. Rationale.....	408
F.33.2. Syntax	409
F.33.3. Precision.....	410
F.33.4. Usage	410
F.33.5. Notes.....	411
F.33.6. Credits	412
F.34. spi	412
F.34.1. refint.c - functions for implementing referential integrity.....	412
F.34.2. timetravel.c - functions for implementing time travel.....	413
F.34.3. autoinc.c - functions for autoincrementing fields	414
F.34.4. insert_username.c - functions for tracking who changed a table	414
F.34.5. moddatetime.c - functions for tracking last modification time.....	414
F.35. sslinfo	414
F.35.1. Functions Provided	415
F.35.2. Author	416
F.36. tablefunc	416
F.36.1. Functions Provided	416
F.36.1.1.normal_rand.....	417
F.36.1.2.crosstab(text)	418

F.36.1.3.crosstabN(text)	420
F.36.1.4.crosstab(text, text)	421
F.36.1.5.connectby	424
F.36.2. Author	426
F.37. test_parser	427
F.37.1. Usage	427
F.38. tsearch2	428
F.38.1. Portability Issues	428
F.38.2. Converting a pre-8.3 Installation	429
F.39. unaccent	430
F.39.1. Configuration	430
F.39.2. Usage	430
F.39.3. Functions	431
F.40. uuid-osp	431
F.40.1. uuid-osp Functions	432
F.40.2. Author	433
F.41. vacuumlo	433
F.41.1. Usage	433
F.41.2. Method	434
F.41.3. Author	434
F.42. xml2	434
F.42.1. Deprecation notice	434
F.42.2. Description of functions	434
F.42.3. xpath_table	435
F.42.3.1.Multivalued results	437
F.42.4. XSLT functions	438
F.42.4.1.xslt_process	438
F.42.5. Author	438
Appendix G. External Projects	439
G.1. Client Interfaces	439
G.2. Procedural Languages	440
G.3. Extensions	440
Appendix H. The Source Code Repository	442
H.1. Getting The Source Via Git	442
Appendix I. Documentation	444
I.1. DocBook	444
I.2. Tool Sets	445
I.2.1. Linux RPM Installation	446
I.2.2. FreeBSD Installation	446

Table of Contents

I.2.3.	Debian Packages	447
I.2.4.	Manual Installation from Source	447
I.2.4.1.	Installing OpenJade	447
I.2.4.2.	Installing the DocBook DTD Kit	448
I.2.4.3.	Installing the DocBook DSSSL Style Sheets	448
I.2.4.4.	Installing JadeTeX	449
I.2.5.	Detection by configure	449
I.3.	Building The Documentation	450
I.3.1.	HTML	450
I.3.2.	Manpages	450
I.3.3.	Print Output via JadeTeX	450
I.3.4.	Overflow Text	451
I.3.5.	Print Output via RTF	451
I.3.6.	Plain Text Files	453
I.3.7.	Syntax Check	453
I.4.	Documentation Authoring	453
I.4.1.	Emacs/PSGML	454
I.4.2.	Other Emacs modes	455
I.5.	Style Guide	455
I.5.1.	Reference Pages	455
Appendix J. Acronyms		458
Bibliography		466
List of Volumes		468
Index		471
Linbrary™ Advertising Club (LAC)		475
Your Advertising Here		483

List of Tables

Table 45-1. System Catalogs	29
Table 45-2. pg_aggregate Columns	30
Table 45-3. pg_am Columns	31
Table 45-4. pg_amop Columns	32
Table 45-5. pg_amproc Columns	32
Table 45-6. pg_attrdef Columns	33
Table 45-7. pg_attribute Columns	34
Table 45-8. pg_authid Columns	35
Table 45-9. pg_auth_members Columns	36
Table 45-10. pg_cast Columns	36
Table 45-11. pg_class Columns	38
Table 45-12. pg_constraint Columns	40
Table 45-13. pg_conversion Columns	41
Table 45-14. pg_database Columns	42
Table 45-15. pg_db_role_setting Columns	42
Table 45-16. pg_default_acl Columns	42
Table 45-17. pg_depend Columns	43
Table 45-18. pg_description Columns	44
Table 45-19. pg_enum Columns	45
Table 45-20. pg_foreign_data_wrapper Columns	45
Table 45-21. pg_foreign_server Columns	46
Table 45-22. pg_index Columns	47
Table 45-23. pg_inherits Columns	47
Table 45-24. pg_language Columns	48
Table 45-25. pg_largeobject Columns	49
Table 45-26. pg_largeobject_metadata Columns	49
Table 45-27. pg_namespace Columns	49
Table 45-28. pg_opclass Columns	50
Table 45-29. pg_operator Columns	51
Table 45-30. pg_opfamily Columns	51

List of Tables

Table 45-31. pg_pltemplate Columns.....	52
Table 45-32. pg_proc Columns.....	54
Table 45-33. pg_rewrite Columns.....	54
Table 45-34. pg_shdepend Columns.....	55
Table 45-35. pg_shdescription Columns.....	56
Table 45-36. pg_statistic Columns.....	58
Table 45-37. pg_tablespace Columns.....	58
Table 45-38. pg_trigger Columns.....	59
Table 45-39. pg_ts_config Columns.....	60
Table 45-40. pg_ts_config_map Columns.....	60
Table 45-41. pg_ts_dict Columns.....	61
Table 45-42. pg_ts_parser Columns.....	61
Table 45-43. pg_ts_template Columns.....	61
Table 45-44. pg_type Columns.....	64
Table 45-45. typcategory Codes.....	65
Table 45-46. pg_user_mapping Columns.....	65
Table 45-47. System Views.....	66
Table 45-48. pg_cursors Columns.....	67
Table 45-49. pg_group Columns.....	68
Table 45-50. pg_indexes Columns.....	68
Table 45-51. pg_locks Columns.....	69
Table 45-52. pg_prepared_statements Columns.....	71
Table 45-53. pg_prepared_xacts Columns.....	71
Table 45-54. pg_roles Columns.....	72
Table 45-55. pg_rules Columns.....	73
Table 45-56. pg_settings Columns.....	73
Table 45-57. pg_shadow Columns.....	74
Table 45-58. pg_stats Columns.....	76
Table 45-59. pg_tables Columns.....	76
Table 45-60. pg_timezone_abbrevs Columns.....	77
Table 45-61. pg_timezone_names Columns.....	77
Table 45-62. pg_user Columns.....	77
Table 54-1. Contents of PGDATA.....	172
Table 54-2. Overall Page Layout.....	177
Table 54-3. PageHeaderData Layout.....	177
Table 54-4. HeapTupleHeaderData Layout.....	179

License

PostgreSQL is Copyright © 1996-2011 by the PostgreSQL Global Development Group and is distributed under the terms of the license of the University of California below.

Postgres95 is Copyright © 1994-5 by 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 9.0 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 VII.

Internals

This part contains assorted information that might be of use to PostgreSQL developers.

Chapter 44.

Overview of PostgreSQL Internals

Author: This chapter originated as part of *Enhancement of the ANSI SQL Implementation of PostgreSQL* (see page 466), Stefan Simkovic's Master's Thesis prepared at Vienna University of Technology under the direction of O.Univ.Prof.Dr. Georg Gottlob and Univ.Ass. Mag. Katrin Seyr.

This chapter gives an overview of the internal structure of the backend of PostgreSQL. After having read the following sections you should have an idea of how a query is processed. This chapter does not aim to provide a detailed description of the internal operation of PostgreSQL, as such a document would be very extensive. Rather, this chapter is intended to help the reader understand the general sequence of operations that occur within the backend from the point at which a query is received, to the point at which the results are returned to the client.

44.1. The Path of a Query

Here we give a short overview of the stages a query has to pass in order to obtain a result.

1. A connection from an application program to the PostgreSQL server has to be established. The application program transmits a query to the server and waits to receive the results sent back by the server.
2. The *parser stage* checks the query transmitted by the application program for correct syntax and creates a *query tree*.
3. The *rewrite system* takes the query tree created by the parser stage and looks for any *rules* (stored in the *system catalogs*) to apply to the query tree. It performs the transformations given in the *rule bodies*.

One application of the rewrite system is in the realization of *views*. Whenever a query against a view (i.e., a *virtual table*) is made, the rewrite system rewrites the user's query to a query that accesses the *base tables* given in the *view definition* instead.

4. The *planner/optimizer* takes the (rewritten) query tree and creates a *query plan* that will be the input to the *executor*.

It does so by first creating all possible *paths* leading to the same result. For example if there is an index on a relation to be scanned, there are two paths for the scan. One possibility is a simple sequential scan and the other possibility is to use the index. Next the cost for the execution of each path is estimated and the cheapest path is chosen. The cheapest path is expanded into a complete plan that the executor can use.

5. The executor recursively steps through the *plan tree* and retrieves rows in the way represented by the plan. The executor makes use of the *storage system* while scanning relations, performs *sorts* and *joins*, evaluates *qualifications* and finally hands back the rows derived.

In the following sections we will cover each of the above listed items in more detail to give a better understanding of PostgreSQL's internal control and data structures.

44.2. How Connections are Established

PostgreSQL is implemented using a simple "process per user" client/server model. In this model there is one *client process* connected to exactly one *server process*. As we do not know ahead of time how many connections will be made, we have to use a *master process* that spawns a new server process every time a connection is requested. This master process is called `postgres` and listens at a specified TCP/IP port for incoming connections. Whenever a request for a connection is detected the `postgres` process spawns a new server process. The server tasks communicate with each other using *semaphores* and *shared memory* to ensure data integrity throughout concurrent data access.

The client process can be any program that understands the PostgreSQL protocol described in *Chapter 46* (page 79). Many clients are based on the C-language library `libpq`, but several independent implementations of the protocol exist, such as the Java JDBC driver.

Once a connection is established the client process can send a query to the *backend* (server). The query is transmitted using plain text, i.e., there is no parsing done in the *frontend* (client). The server parses the query, creates an *execution plan*, executes the plan and returns the retrieved rows to the client by transmitting them over the established connection.

44.3. The Parser Stage

The *parser stage* consists of two parts:

- The *parser* defined in `gram.y` and `scan.l` is built using the Unix tools `bison` and `flex`.
- The *transformation process* does modifications and augmentations to the data structures returned by the parser.

44.3.1. Parser

The parser has to check the query string (which arrives as plain ASCII text) for valid syntax. If the syntax is correct a *parse tree* is built up and handed back; otherwise an error is returned. The parser and lexer are implemented using the well-known Unix tools bison and flex.

The *lexer* is defined in the file `scan.l` and is responsible for recognizing *identifiers*, the *SQL key words* etc. For every key word or identifier that is found, a *token* is generated and handed to the parser.

The parser is defined in the file `gram.y` and consists of a set of *grammar rules* and *actions* that are executed whenever a rule is fired. The code of the actions (which is actually C code) is used to build up the parse tree.

The file `scan.l` is transformed to the C source file `scan.c` using the program flex and `gram.y` is transformed to `gram.c` using bison. After these transformations have taken place a normal C compiler can be used to create the parser. Never make any changes to the generated C files as they will be overwritten the next time flex or bison is called.



Note

The mentioned transformations and compilations are normally done automatically using the *makefiles* shipped with the PostgreSQL source distribution.

A detailed description of bison or the grammar rules given in `gram.y` would be beyond the scope of this paper. There are many books and documents dealing with flex and bison. You should be familiar with bison before you start to study the grammar given in `gram.y` otherwise you won't understand what happens there.

44.3.2. Transformation Process

The parser stage creates a parse tree using only fixed rules about the syntactic structure of SQL. It does not make any lookups in the system catalogs, so there is no possibility to understand the detailed semantics of the requested operations. After the parser completes, the *transformation process* takes the tree handed back by the parser as input and does the semantic interpretation needed to understand which tables, functions, and operators are referenced by the query. The data structure that is built to represent this information is called the *query tree*.

The reason for separating raw parsing from semantic analysis is that system catalog lookups can only be done within a transaction, and we do not wish to start a transaction immediately upon receiving a query string. The raw parsing stage is sufficient to identify the transaction control commands (`BEGIN`, `ROLLBACK`, etc), and these can then be correctly executed without any further analysis. Once we know that we are dealing with an actual

query (such as `SELECT` or `UPDATE`), it is okay to start a transaction if we're not already in one. Only then can the transformation process be invoked.

The query tree created by the transformation process is structurally similar to the raw parse tree in most places, but it has many differences in detail. For example, a `FuncCall` node in the parse tree represents something that looks syntactically like a function call. This might be transformed to either a `FuncExpr` or `Aggref` node depending on whether the referenced name turns out to be an ordinary function or an aggregate function. Also, information about the actual data types of columns and expression results is added to the query tree.

44.4. The PostgreSQL Rule System

PostgreSQL supports a powerful *rule system* for the specification of *views* and ambiguous *view updates*. Originally the PostgreSQL rule system consisted of two implementations:

- The first one worked using *row level* processing and was implemented deep in the *executor*. The rule system was called whenever an individual row had been accessed. This implementation was removed in 1995 when the last official release of the Berkeley Postgres project was transformed into Postgres95.
- The second implementation of the rule system is a technique called *query rewriting*. The *rewrite system* is a module that exists between the *parser stage* and the *planner/optimizer*. This technique is still implemented.

The query rewriter is discussed in some detail in *Chapter 37* (Vol.III), so there is no need to cover it here. We will only point out that both the input and the output of the rewriter are query trees, that is, there is no change in the representation or level of semantic detail in the trees. Rewriting can be thought of as a form of macro expansion.

44.5. Planner/Optimizer

The task of the *planner/optimizer* is to create an optimal execution plan. A given SQL query (and hence, a query tree) can be actually executed in a wide variety of different ways, each of which will produce the same set of results. If it is computationally feasible, the query optimizer will examine each of these possible execution plans, ultimately selecting the execution plan that is expected to run the fastest.



Note

In some situations, examining each possible way in which a query can be executed would take an excessive amount of time and memory space. In particular, this occurs when executing queries involving large numbers of join operations. In order to determine a reasonable (not necessarily optimal) query plan in a reasonable amount of time, PostgreSQL uses a Genetic Query Optimizer (see *Chapter 50* - page 137) when the number of joins exceeds a threshold (see `geqo_threshold` - Vol.II).

The planner's search procedure actually works with data structures called *paths*, which are simply cut-down representations of plans containing only as much information as the planner needs to make its decisions. After the cheapest path is determined, a full-fledged *plan tree* is built to pass to the executor. This represents the desired execution plan in sufficient detail for the executor to run it. In the rest of this section we'll ignore the distinction between paths and plans.

44.5.1. Generating Possible Plans

The planner/optimizer starts by generating plans for scanning each individual relation (table) used in the query. The possible plans are determined by the available indexes on each relation. There is always the possibility of performing a sequential scan on a relation, so a sequential scan plan is always created. Assume an index is defined on a relation (for example a B-tree index) and a query contains the restriction `relation.attribute OPR constant`. If `relation.attribute` happens to match the key of the B-tree index and `OPR` is one of the operators listed in the index's *operator class*, another plan is created using the B-tree index to scan the relation. If there are further indexes present and the restrictions in the query happen to match a key of an index, further plans will be considered. Index scan plans are also generated for indexes that have a sort ordering that can match the query's `ORDER BY` clause (if any), or a sort ordering that might be useful for merge joining (see below).

If the query requires joining two or more relations, plans for joining relations are considered after all feasible plans have been found for scanning single relations. The three available join strategies are:

- *nested loop join*: The right relation is scanned once for every row found in the left relation. This strategy is easy to implement but can be very time consuming. (However, if the right relation can be scanned with an index scan, this can be a good strategy. It is possible to use values from the current row of the left relation as keys for the index scan of the right.)
- *merge join*: Each relation is sorted on the join attributes before the join starts. Then the two relations are scanned in parallel, and matching rows are combined to form join rows. This kind of join is more attractive because each relation has to be scanned only once. The required sorting might be achieved either by an explicit sort step, or by scanning the relation in the proper order using an index on the join key.
- *hash join*: the right relation is first scanned and loaded into a hash table, using its join attributes as hash keys. Next the left relation is scanned and the appropriate values of every row found are used as hash keys to locate the matching rows in the table.

When the query involves more than two relations, the final result must be built up by a tree of join steps, each with two inputs. The planner examines different possible join sequences to find the cheapest one.

If the query uses fewer than `geqo_threshold` (Vol.II) relations, a near-exhaustive search is conducted to find the best join sequence. The planner preferentially considers joins between any two relations for which there exist a corresponding join clause in the `WHERE` qualification (i.e., for which a restriction like `where rel1.attr1=rel2.attr2` exists). Join pairs with no join clause are considered only when there is no other choice, that is, a particular relation has no available join clauses to any other relation. All possible plans are generated for every join pair considered by the planner, and the one that is (estimated to be) the cheapest is chosen.

When `geqo_threshold` is exceeded, the join sequences considered are determined by heuristics, as described in *Chapter 50* (page 137). Otherwise the process is the same.

The finished plan tree consists of sequential or index scans of the base relations, plus nested-loop, merge, or hash join nodes as needed, plus any auxiliary steps needed, such as sort nodes or aggregate-function calculation nodes. Most of these plan node types have the additional ability to do *selection* (discarding rows that do not meet a specified Boolean condition) and *projection* (computation of a derived column set based on given column values, that is, evaluation of scalar expressions where needed). One of the responsibilities of the planner is to attach selection conditions from the `WHERE` clause and computation of required output expressions to the most appropriate nodes of the plan tree.

44.6. Executor

The *executor* takes the plan created by the planner/optimizer and recursively processes it to extract the required set of rows. This is essentially a demand-pull pipeline mechanism. Each time a plan node is called, it must deliver one more row, or report that it is done delivering rows.

To provide a concrete example, assume that the top node is a `MergeJoin` node. Before any merge can be done two rows have to be fetched (one from each subplan). So the executor recursively calls itself to process the subplans (it starts with the subplan attached to `lefttree`). The new top node (the top node of the left subplan) is, let's say, a `Sort` node and again recursion is needed to obtain an input row. The child node of the `Sort` might be a `SeqScan` node, representing actual reading of a table. Execution of this node causes the executor to fetch a row from the table and return it up to the calling node. The `Sort` node will repeatedly call its child to obtain all the rows to be sorted. When the input is exhausted (as indicated by the child node returning a `NULL` instead of a row), the `Sort` code performs the sort, and finally is able to return its first output row, namely the first one in sorted order. It keeps the remaining rows stored so that it can deliver them in sorted order in response to later demands.

The `MergeJoin` node similarly demands the first row from its right subplan. Then it compares the two rows to see if they can be joined; if so, it returns a join row to its caller. On

the next call, or immediately if it cannot join the current pair of inputs, it advances to the next row of one table or the other (depending on how the comparison came out), and again checks for a match. Eventually, one subplan or the other is exhausted, and the `MergeJoin` node returns `NULL` to indicate that no more join rows can be formed.

Complex queries can involve many levels of plan nodes, but the general approach is the same: each node computes and returns its next output row each time it is called. Each node is also responsible for applying any selection or projection expressions that were assigned to it by the planner.

The executor mechanism is used to evaluate all four basic SQL query types: `SELECT`, `INSERT`, `UPDATE`, and `DELETE`. For `SELECT`, the top-level executor code only needs to send each row returned by the query plan tree off to the client. For `INSERT`, each returned row is inserted into the target table specified for the `INSERT`. This is done in a special top-level plan node called `ModifyTable`. (A simple `INSERT ... VALUES` command creates a trivial plan tree consisting of a single `Result` node, which computes just one result row, and `ModifyTable` above it to perform the insertion. But `INSERT ... SELECT` can demand the full power of the executor mechanism.) For `UPDATE`, the planner arranges that each computed row includes all the updated column values, plus the *TID* (tuple ID, or row ID) of the original target row; this data is fed into a `ModifyTable` node, which uses the information to create a new updated row and mark the old row deleted. For `DELETE`, the only column that is actually returned by the plan is the *TID*, and the `ModifyTable` node simply uses the *TID* to visit each target row and mark it deleted.

Part VIII.

Appendixes

Appendix J.

Acronyms

This is a list of acronyms commonly used in the PostgreSQL documentation and in discussions about PostgreSQL.

ANSI

American National Standards Institute
(http://en.wikipedia.org/wiki/American_National_Standards_Institute)

API

Application Programming Interface
(<http://en.wikipedia.org/wiki/API>)

ASCII

American Standard Code for Information Interchange
(<http://en.wikipedia.org/wiki/Ascii>)

BKI

Backend Interface
(page 180)

CA

Certificate Authority
(http://en.wikipedia.org/wiki/Certificate_authority)

CIDR

Classless Inter-Domain Routing
(http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing)

CPAN

Comprehensive Perl Archive Network
(<http://www.cpan.org/>)

CRL

Certificate Revocation List
(http://en.wikipedia.org/wiki/Certificate_revocation_list)

CSV

Comma Separated Values
(http://en.wikipedia.org/wiki/Comma-separated_values)

CTE

Common Table Expression
(<http://www.postgresql.org/docs/9.0/static/queries-with.html>)

CVE

Common Vulnerabilities and Exposures
(<http://cve.mitre.org/>)

DBA

Database Administrator
(http://en.wikipedia.org/wiki/Database_administrator)

DBI

Database Interface (Perl)
(<http://dbi.perl.org/>)

DBMS

Database Management System
(<http://en.wikipedia.org/wiki/Dbms>)

DDL

Data Definition Language
(http://en.wikipedia.org/wiki/Data_Definition_Language),
SQL commands such as CREATE TABLE, ALTER USER

DML

Data Manipulation Language
(http://en.wikipedia.org/wiki/Data_Manipulation_Language),
SQL commands such as INSERT, UPDATE, DELETE

DST

Daylight Saving Time
(http://en.wikipedia.org/wiki/Daylight_saving_time)

ECPG

Embedded C for PostgreSQL
(Vol.II)

ESQL

Embedded SQL

(http://en.wikipedia.org/wiki/Embedded_SQL)

FAQ

Frequently Asked Questions

(<http://en.wikipedia.org/wiki/FAQ>)

FSM

Free Space Map

(page 175)

GEQO

Genetic Query Optimizer

(page 137)

GIN

Generalized Inverted Index

(page 165)

GiST

Generalized Search Tree

(page 155)

Git

Git

(http://en.wikipedia.org/wiki/Git_%28software%29)

GMT

Greenwich Mean Time

(<http://en.wikipedia.org/wiki/GMT>)

GSSAPI

Generic Security Services Application Programming Interface

(http://en.wikipedia.org/wiki/Generic_Security_Services_Application_Program_Interface)

GUC

Grand Unified Configuration

(Vol.II), the PostgreSQL subsystem that handles server configuration

HBA

Host-Based Authentication

(Vol.II)

HOT

Heap-Only Tuples

(<http://anoncvs.postgresql.org/cvsweb.cgi/pgsql/src/backend/access/heap/README.HOT>)

IEC

International Electrotechnical Commission

(http://en.wikipedia.org/wiki/International_Electrotechnical_Commission)

IEEE

Institute of Electrical and Electronics Engineers

(<http://standards.ieee.org/>)

IPC

Inter-Process Communication

(http://en.wikipedia.org/wiki/Inter-process_communication)

ISO

International Organization for Standardization

(<http://www.iso.org/iso/home.htm>)

ISSN

International Standard Serial Number

(<http://en.wikipedia.org/wiki/Issn>)

JDBC

Java Database Connectivity

(http://en.wikipedia.org/wiki/Java_Database_Connectivity)

LDAP

Lightweight Directory Access Protocol

(http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol)

MSVC

Microsoft Visual C

(http://en.wikipedia.org/wiki/Visual_C++)

MVCC

Multi-Version Concurrency Control

(Vol.I)

NLS

National Language Support

(http://en.wikipedia.org/wiki/Internationalization_and_localization)

ODBC

Open Database Connectivity
(http://en.wikipedia.org/wiki/Open_Database_Connectivity)

OID

Object Identifier
(Vol.I)

OLAP

Online Analytical Processing
(<http://en.wikipedia.org/wiki/Olap>)

OLTP

Online Transaction Processing
(<http://en.wikipedia.org/wiki/OLTP>)

ORDBMS

Object-Relational Database Management System
(<http://en.wikipedia.org/wiki/ORDBMS>)

PAM

Pluggable Authentication Modules
(http://en.wikipedia.org/wiki/Pluggable_Authentication_Modules)

PGSQL

PostgreSQL
(Vol.I)

PGXS

PostgreSQL Extension System
(Vol.III)

PID

Process Identifier
(http://en.wikipedia.org/wiki/Process_identifier)

PITR

Point-In-Time Recovery
(Vol.II)
(Continuous Archiving)

PL

Programming Languages (server-side)
(Vol.III)

POSIX

Portable Operating System Interface
(<http://en.wikipedia.org/wiki/POSIX>)

RDBMS

Relational Database Management System
(http://en.wikipedia.org/wiki/Relational_database_management_system)

RFC

Request For Comments
(http://en.wikipedia.org/wiki/Request_for_Comments)

SGML

Standard Generalized Markup Language
(<http://en.wikipedia.org/wiki/SGML>)

SPI

Server Programming Interface
(Vol.III)

SQL

Structured Query Language
(<http://en.wikipedia.org/wiki/SQL>)

SRF

Set-Returning Function
(Vol.III)

SSH

Secure Shell
(http://en.wikipedia.org/wiki/Secure_Shell)

SSL

Secure Sockets Layer
(http://en.wikipedia.org/wiki/Secure_Sockets_Layer)

SSPI

Security Support Provider Interface
(<http://msdn2.microsoft.com/en-us/library/aa380493.aspx>)

SYSV

Unix System V
(http://en.wikipedia.org/wiki/System_V)

TCP/IP

Transmission Control Protocol (TCP) / Internet Protocol (IP)
(http://en.wikipedia.org/wiki/Transmission_Control_Protocol)

TID

Tuple Identifier
(Vol.I)

TOAST

The Oversized-Attribute Storage Technique
(page 173)

TPC

Transaction Processing Performance Council
(<http://www.tpc.org/>)

URL

Uniform Resource Locator
(<http://en.wikipedia.org/wiki/URL>)

UTC

Coordinated Universal Time
(http://en.wikipedia.org/wiki/Coordinated_Universal_Time)

UTF

Unicode Transformation Format
(<http://www.unicode.org/>)

UTF8

Eight-Bit Unicode Transformation Format
(<http://en.wikipedia.org/wiki/Utf8>)

UUID

Universally Unique Identifier
(Vol.I)

WAL

Write-Ahead Log
(Vol.II)

XID

Transaction Identifier
(Vol.I)

XML

Extensible Markup Language
(<http://en.wikipedia.org/wiki/XML>)

Bibliography

Selected references and readings for SQL and PostgreSQL.

Some white papers and technical reports from the original POSTGRES development team are available at the University of California, Berkeley, Computer Science Department *web site*¹.

SQL Reference Books

Judith Bowman, Sandra Emerson, and Marcy Darnovsky, *The Practical SQL Handbook: Using SQL Variants*, Fourth Edition, Addison-Wesley Professional, ISBN 0-201-70309-2, 2001.

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

C. J. Date, *An Introduction to Database Systems*, Eighth Edition, Addison-Wesley, ISBN 0-321-19784-4, 2003.

Ramez Elmasri and Shamkant Navathe, *Fundamentals of Database Systems*, Fourth Edition, Addison-Wesley, ISBN 0-321-12226-7, 2003.

Jim Melton and Alan R. Simon, *Understanding the New SQL: A complete guide*, Morgan Kaufmann, ISBN 1-55860-245-3, 1993.

Jeffrey D. Ullman, *Principles of Database and Knowledge: Base Systems*, Volume 1, Computer Science Press, 1988.

PostgreSQL-Specific Documentation

Stefan Simkovics, *Enhancement of the ANSI SQL Implementation of PostgreSQL*, Department of Information Systems, Vienna University of Technology, November 29, 1998.

Discusses SQL history and syntax, and describes the addition of `INTERSECT` and `EXCEPT` constructs into PostgreSQL. Prepared as a Master's Thesis with the support of O. Univ. Prof. Dr. Georg Gottlob and Univ. Ass. Mag. Katrin Seyr at Vienna University of Technology.

A. Yu and J. Chen, The POSTGRES Group, *The Postgres95 User Manual*, University of California, Sept. 5, 1995.

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

Bibliography

Zelaine Fong, The design and implementation of the POSTGRES query optimizer², University of California, Berkeley, Computer Science Department.

Proceedings and Articles

Nels Olson, *Partial indexing in POSTGRES: research project*, University of California, UCB Engin T7.49.1993 O676, 1993.

L. Ong and J. Goh, "A Unified Framework for Version Modeling Using Production Rules in a Database System", *ERL Technical Memorandum M90/33*, University of California, Apr, 1990.

L. Rowe and M. Stonebraker, "*The POSTGRES data model*"³, Proc. VLDB Conference, Sept. 1987.

P. Seshadri and A. Swami, "Generalized Partial Indexes (*cached version*)"⁴, Proc. Eleventh International Conference on Data Engineering, 6-10 March 1995, IEEE Computer Society Press, Cat. No.95CH35724, 1995, 420-7.

M. Stonebraker and L. Rowe, "*The design of POSTGRES*"⁵, Proc. ACM-SIGMOD Conference on Management of Data, May 1986.

M. Stonebraker, E. Hanson, and C. H. Hong, "The design of the POSTGRES rules system", Proc. IEEE Conference on Data Engineering, Feb. 1987.

M. Stonebraker, "*The design of the POSTGRES storage system*"⁶, Proc. VLDB Conference, Sept. 1987.

M. Stonebraker, M. Hearst, and S. Potamianos, "*A commentary on the POSTGRES rules system*"⁷, *SIGMOD Record* 18(3), Sept. 1989.

M. Stonebraker, "*The case for partial indexes*"⁸, *SIGMOD Record* 18(4), Dec. 1989, 4-11.

M. Stonebraker, L. A. Rowe, and M. Hirohama, "*The implementation of POSTGRES*"⁹, *Transactions on Knowledge and Data Engineering* 2(1), IEEE, March 1990.

M. Stonebraker, A. Jhingran, J. Goh, and S. Potamianos, "*On Rules, Procedures, Caching and Views in Database Systems*"¹⁰, Proc. ACM-SIGMOD Conference on Management of Data, June 1990.

² <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/UCB-MS-zfong.pdf>

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

⁴ <http://citeseer.ist.psu.edu/seshadri95generalized.html>

⁵ <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M85-95.pdf>

⁶ <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M87-06.pdf>

⁷ <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M89-82.pdf>

⁸ <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M89-17.pdf>

⁹ <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M90-34.pdf>

¹⁰ <http://s2k-ftp.cs.berkeley.edu:8000/postgres/papers/ERL-M90-36.pdf>

List of Volumes

Volume I.

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.

Part III. Server Administration

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

List of Volumes

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. Recovery Configuration
27. Monitoring Database Activity
28. Monitoring Disk Usage
29. Reliability and the Write-Ahead Log
30. Regression Tests

Part IV. Client Interfaces

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

Volume III.

Part V. Server Programming

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

Volume IV.

Part VI. Reference

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

Volume V.**Part VII. Internals**

- 44. Overview of PostgreSQL Internals
- 45. System Catalogs
- 46. Frontend/Backend Protocol
- 47. PostgreSQL Coding Conventions
- 48. Native Language Support
- 49. Writing A Procedural Language Handler
- 50. Genetic Query Optimizer
- 51. Index Access Method Interface Definition
- 52. GiST Indexes
- 53. GIN Indexes
- 54. Database Physical Storage
- 55. BKI Backend Interface
- 56. 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

adminpack.....	296
auto_explain	296
auto_explain.log_analyze	
configuration parameter	297, 396
auto_explain.log_buffers	
configuration parameter	297, 396
auto_explain.log_format	
configuration parameter	297, 396
auto_explain.log_min_duration	
configuration parameter	297, 396
auto_explain.log_nested_statements	
configuration parameter	297, 396
auto_explain.log_verbose	
configuration parameter	297, 396
bison.....	126, 291
B-tree.....	141, 280, 341, 356
btree_gin.....	298
btree_gist.....	299
chkpass	300
citext.....	301
CLUSTER	
of databases	
See database cluster	126, 400
column.....	126, 400
commit_delay	
configuration parameter	270, 276
commit_siblings	
configuration parameter	270, 276
constraint.....	278
cube	303
cursor	
in PL/pgSQL	289
data area	
See database cluster	126, 400
data type.....	96, 271, 281
category	28, 79, 80, 269, 359, 368, 389
database cluster	126, 400
dblink.....	307, 311
dict_int.....	331
dict_xsyn	332
earthdistance.....	333
elog.....	118
encryption	
for specific columns	376
ereport	118
error codes	
list of	192
extensions.....	440
flex.....	126, 291
Free Space Map	175
FSM	
See Free Space Map	175
fsync configuration parameter	270, 276
full text search	281
full_page_writes	
configuration parameter	270, 276
function.....	349
type resolution	
in an invocation.....	143, 152, 282, 283, 343, 354, 363, 388, 395, 398, 431, 432

fuzzystrmatch.....335
 genetic query optimization.....137
 GEQO
 See genetic query optimization.....137
 geqo configuration parameter137
 geqo_effort configuration parameter137
 geqo_generations
 configuration parameter137
 geqo_pool_size
 configuration parameter137
 geqo_seed configuration parameter.....137
 geqo_selection_bias
 configuration parameter137
 geqo_threshold
 configuration parameter137
 GIN
 See index141, 280, 341, 356
 GiST
 See index141, 280, 341, 356
 hash
 See index141, 280, 341, 356
 hierarchical database.....126, 400
 history
 of PostgreSQL.....468
 hstore337
 index141, 280, 341, 356
 GIN165
 GiST155
 intagg.....343
 intarray345
 interfaces
 externally maintained439
 isn.....347
 key word
 list of207
 lex.....126, 291
 libedit.....126, 291

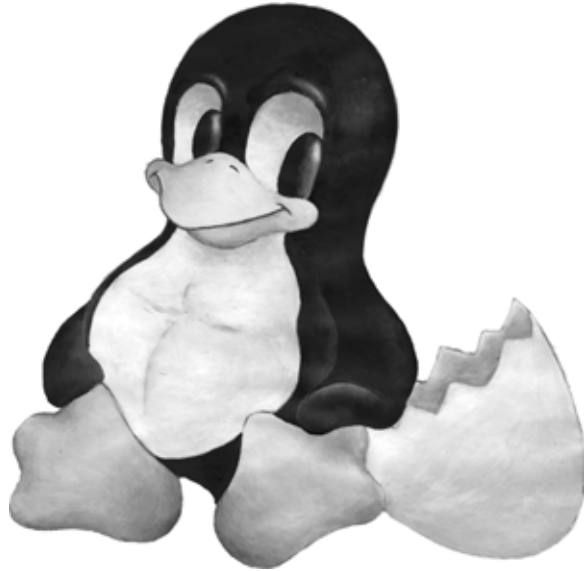
libperl126, 291
 libpython.....126, 291
 LIKE
 and locales302
 lo351
 lower
 and locales302
 ltree352
 make126, 291
 max_wal_senders
 configuration parameter273
 MVCC.....155, 165
 object-oriented database126, 400
 oid2name.....359
 operator349
 ORDER BY
 and locales302
 pageinspect.....363
 passwordcheck.....365
 perl.....126, 291
 pg_aggregate30
 pg_am.....30
 pg_amop31
 pg_amproc32
 pg_archivecleanup.....365
 pg_attrdef.....32
 pg_attribute33
 pg_auth_members35
 pg_authid.....34
 pg_buffercache.....374
 pg_cast.....36
 pg_class37
 pg_constraint.....39
 pg_conversion40
 pg_cursors.....67
 pg_database.....41
 pg_db_role_setting.....42



Index

pg_default_acl	42	pg_timezone_names	77
pg_depend	43	pg_trgm	400
pg_description	44	pg_trigger	58
pg_enum	45	pg_ts_config	59
pg_foreign_data_wrapper	45	pg_ts_config_map	60
pg_foreign_server	45	pg_ts_dict	60
pg_freespacemap	388	pg_ts_parser	61
pg_group	67	pg_ts_template	61
pg_index	46	pg_type	62
pg_indexes	68	pg_upgrade	402
pg_inherits	47	pg_user	77
pg_language	47	pg_user_mapping	65
pg_largeobject	48	pg_user_mappings	78
pg_largeobject_metadata	49	pg_views	78
pg_locks	68	pgbench	367
pg_namespace	49	pgcrypto	376
pg_opclass	49	pgrowlocks	389
pg_operator	50	pgstattuple	397
pg_opfamily	51	plperl.on_init configuration	
pg_pltemplate	51	parameter	331, 332, 385, 430
pg_prepared_statements	70	plperl.on_plperl_init configuration	
pg_prepared_xacts	71	parameter	331, 332, 385, 430
pg_proc	52	plperl.on_plperlu_init configuration	
pg_rewrite	54	parameter	331, 332, 385, 430
pg_roles	72	plperl.use_strict configuration	
pg_rules	72	parameter	331, 332, 385, 430
pg_settings	73	procedural language	440
pg_shadow	74	externally maintained	440
pg_shdepend	55	handler for	133
pg_shdescription	56	protocol	
pg_standby	390	frontend-backend	79
pg_stat_statements	394	query	270, 277
pg_statistic	56	readline	126, 291
pg_stats	74	relation	126, 400
pg_tables	76	relational database	126, 400
pg_tablespace	58	row	126, 400
pg_timezone_abbrevs	76		

row estimation		tsearch2	428
planner	184	type	
schema	28, 79, 80, 269, 359, 368, 389	See data type.....	96, 271, 281
seg	408	unaccent	143, 152, 282, 283, 343, 354, 363, 388, 395, 398, 430, 431, 432
SELECT	270, 277	upper	
sliced bread		and locales	302
See TOAST	173	uuid-osp	431
SPI		vacuum_defer_cleanup_age	
examples	412	configuration parameter	273
SQL/CLI	238	vacuumlo	433
SQL/Foundation.....	238	view	66
SQL/Framework.....	238	Visibility Map.....	176
SQL/JRT.....	238	VM	
SQL/MED.....	238	See Visibility Map.....	176
SQL/OLB.....	238	wal_buffers	
SQL/PSM.....	238	configuration parameter	270, 276
SQL/Schemata	238	wal_keep_segments	
SQL/XML	238	configuration parameter	273
sslinfo.....	414	wal_level	
synchronous_commit		configuration parameter	270, 276
configuration parameter	270, 276	wal_sender_delay	
table.....	126, 400	configuration parameter	273
tablefunc.....	416	wal_sync_method	
test_parser.....	427	configuration parameter	270, 276
text search	281	wal_writer_delay	
time zone		configuration parameter	270, 276
input abbreviations.....	203	xml2	434
to_char		yacc	126, 291
and locales	302	zlib.....	126, 291
TOAST.....	173		
versus large objects.....	155, 165		

Linbrary™ Advertising Club (LAC)



Linbrary™  Official Docs as a Real Books <http://www.linbrary.com>  **Linux Library**



Linbrary Advertising Club

Advertising



Linux Documentation Project - Machtelt Garrels

<http://www.tldp.org/>

Version	Title	Edition	ISBN- 10	ISBN- 13
TLDP	Introduction to Linux (Third Edition)	paperback	1-59682-199-X	978-1-59682-199-6
		eBook (pdf)	1-59682-200-7	978-1-59682-200-9
	Bash Guide for Beginners (Second Edition)	paperback	1-59682-201-5	978-1-59682-201-6
		eBook (pdf)	1-59682-202-3	978-1-59682-202-3

<http://www.linbrary.com/linux-tldp/>



Linbrary Advertising Club



Fedora Project Official Documentation

<http://docs.fedoraproject.org>

Version	Title	Edition	ISBN- 10	ISBN- 13
Fedora 14	Fedora 14 Installation Guide	paperback	1-59682-228-7	978-1-59682-228-3
		eBook (pdf)	1-59682-233-3	978-1-59682-233-7
	Fedora 14 User Guide	paperback	1-59682-229-5	978-1-59682-229-0
		eBook (pdf)	1-59682-234-1	978-1-59682-234-4
	Fedora 14 Security Guide	paperback	1-59682-230-9	978-1-59682-230-6
		eBook (pdf)	1-59682-235-X	978-1-59682-235-1
	Fedora 14 Storage Administration Guide	paperback	1-59682-231-7	978-1-59682-231-3
		eBook (pdf)	1-59682-236-8	978-1-59682-236-8
Fedora 14 Musicians Guide	paperback	1-59682-232-5	978-1-59682-232-0	
	eBook (pdf)	1-59682-237-6	978-1-59682-237-5	
Fedora 13	Fedora 13 Installation Guide	paperback	1-59682-212-0	978-1-59682-212-2
		eBook (pdf)	1-59682-217-1	978-1-59682-217-7
	Fedora 13 User Guide	paperback	1-59682-213-9	978-1-59682-213-9
		eBook (pdf)	1-59682-218-X	978-1-59682-218-4
	Fedora 13 Security Guide	paperback	1-59682-214-7	978-1-59682-214-6
		eBook (pdf)	1-59682-219-8	978-1-59682-219-1
	Fedora 13 SE Linux User Guide	paperback	1-59682-215-5	978-1-59682-215-3
		eBook (pdf)	1-59682-220-1	978-1-59682-220-7
	Fedora 13 Virtualization Guide	paperback	1-59682-216-3	978-1-59682-216-0
		eBook (pdf)	1-59682-221-X	978-1-59682-221-4



Linbrary Advertising Club



Fedora Project Official Documentation

<http://docs.fedoraproject.org>

Version	Title	Edition	ISBN- 10	ISBN- 13
Fedora 12	Fedora 12 Installation Guide	paperback	1-59682-179-5	978-1-59682-179-8
		eBook (pdf)	1-59682-184-1	978-1-59682-184-2
	Fedora 12 User Guide	paperback	1-59682-180-9	978-1-59682-180-4
		eBook (pdf)	1-59682-185-X	978-1-59682-185-9
	Fedora 12 Security Guide	paperback	1-59682-181-7	978-1-59682-181-1
		eBook (pdf)	1-59682-186-8	978-1-59682-186-6
	Fedora 12 SE Linux User Guide	paperback	1-59682-182-5	978-1-59682-182-8
		eBook (pdf)	1-59682-187-6	978-1-59682-187-3
Fedora 12 Virtualization Guide	paperback	1-59682-183-3	978-1-59682-183-5	
	eBook (pdf)	1-59682-188-4	978-1-59682-188-0	
Fedora 11	Fedora 11 Installation Guide	paperback	1-59682-142-6	978-1-59682-142-2
		eBook (pdf)	1-59682-146-9	978-1-59682-146-0
	Fedora 11 User Guide	paperback	1-59682-143-4	978-1-59682-143-9
		eBook (pdf)	1-59682-147-7	978-1-59682-147-7
	Fedora 11 Security Guide	paperback	1-59682-144-2	978-1-59682-144-6
		eBook (pdf)	1-59682-148-5	978-1-59682-148-4
	Fedora 11 SE Linux User Guide	paperback	1-59682-145-0	978-1-59682-145-3
		eBook (pdf)	1-59682-149-3	978-1-59682-149-1
<i>http://www.linlibrary.com/fedora/</i>				



Linlibrary Advertising Club



Ubuntu Official Documentation

<http://www.ubuntu.com/>

Version	Title	Edition	ISBN- 10	ISBN- 13
Ubuntu 10.10	Ubuntu 10.10 Installation Guide	paperback	1-59682-238-4	978-1-59682-238-2
		eBook (pdf)	1-59682-242-2	978-1-59682-242-9
	Ubuntu 10.10 Desktop Guide	paperback	1-59682-239-2	978-1-59682-239-9
		eBook (pdf)	1-59682-243-0	978-1-59682-243-6
	Ubuntu 10.10 Server Guide	paperback	1-59682-240-6	978-1-59682-240-5
		eBook (pdf)	1-59682-244-9	978-1-59682-244-3
Ubuntu 10.10 Packaging Guide	paperback	1-59682-241-4	978-1-59682-241-2	
	eBook (pdf)	1-59682-245-7	978-1-59682-245-0	
Ubuntu 10.04 LTS	Ubuntu 10.04 LTS Installation Guide	paperback	1-59682-203-1	978-1-59682-203-0
		eBook (pdf)	1-59682-207-4	978-1-59682-207-8
	Ubuntu 10.04 LTS Desktop Guide	paperback	1-59682-204-X	978-1-59682-204-7
		eBook (pdf)	1-59682-208-2	978-1-59682-208-5
	Ubuntu 10.04 LTS Server Guide	paperback	1-59682-205-8	978-1-59682-205-4
		eBook (pdf)	1-59682-209-0	978-1-59682-209-2
	Ubuntu 10.04 LTS Packaging Guide	paperback	1-59682-206-6	978-1-59682-206-1
		eBook (pdf)	1-59682-210-4	978-1-59682-210-8



Linbrary Advertising Club



Ubuntu Official Documentation

<http://www.ubuntu.com/>

Version	Title	Edition	ISBN- 10	ISBN- 13
Ubuntu 9.10	Ubuntu 9.10 Installation Guide	paperback	1-59682-171-X	978-1-59682-171-2
		eBook (pdf)	1-59682-175-2	978-1-59682-175-0
	Ubuntu 9.10 Desktop Guide	paperback	1-59682-172-8	978-1-59682-172-9
		eBook (pdf)	1-59682-176-0	978-1-59682-176-7
	Ubuntu 9.10 Server Guide	paperback	1-59682-173-6	978-1-59682-173-6
		eBook (pdf)	1-59682-177-9	978-1-59682-177-4
Ubuntu 9.10 Packaging Guide	paperback	1-59682-174-4	978-1-59682-174-3	
	eBook (pdf)	1-59682-178-7	978-1-59682-178-1	
Ubuntu 9.04	Ubuntu 9.04 Installation Guide	paperback	1-59682-150-7	978-1-59682-150-7
		eBook (pdf)	1-59682-154-X	978-1-59682-154-5
	Ubuntu 9.04 Desktop Guide	paperback	1-59682-151-5	978-1-59682-151-4
		eBook (pdf)	1-59682-155-8	978-1-59682-155-2
	Ubuntu 9.04 Server Guide	paperback	1-59682-152-3	978-1-59682-152-1
		eBook (pdf)	1-59682-156-6	978-1-59682-156-9
	Ubuntu 9.04 Packaging Guide	paperback	1-59682-153-1	978-1-59682-153-8
		eBook (pdf)	1-59682-157-4	978-1-59682-157-6
<i>http://www.linbrary.com/ubuntu/</i>				



Linbrary Advertising Club



PostgreSQL Official Documentation

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

Version	Title	Edition	ISBN- 10	ISBN- 13
PostgreSQL 9.0	PostgreSQL 9.0 Volume I. The SQL Language	paperback	1-59682-246-5	978-1-59682-246-7
		eBook (pdf)	1-59682-251-1	978-1-59682-251-1
	PostgreSQL 9.0 Volume II. Server Administration	paperback	1-59682-247-3	978-1-59682-247-4
		eBook (pdf)	1-59682-252-X	978-1-59682-252-8
	PostgreSQL 9.0 Volume III. Server Programming	paperback	1-59682-248-1	978-1-59682-248-1
		eBook (pdf)	1-59682-253-8	978-1-59682-253-5
	PostgreSQL 9.0 Volume IV. Reference	paperback	1-59682-249-X	978-1-59682-249-8
		eBook (pdf)	1-59682-254-6	978-1-59682-254-2
	PostgreSQL 9.0 Volume V. Internals & Appendixes	paperback	1-59682-250-3	978-1-59682-250-4
		eBook (pdf)	1-59682-255-4	978-1-59682-255-9
PostgreSQL 8.04	PostgreSQL 8.04 Volume I. The SQL Language	paperback	1-59682-158-2	978-1-59682-158-3
		eBook (pdf)	1-59682-163-9	978-1-59682-163-7
	PostgreSQL 8.04 Volume II. Server Administration	paperback	1-59682-159-0	978-1-59682-159-0
		eBook (pdf)	1-59682-164-7	978-1-59682-164-4
	PostgreSQL 8.04 Volume III. Server Programming	paperback	1-59682-160-4	978-1-59682-160-6
		eBook (pdf)	1-59682-165-5	978-1-59682-165-1
	PostgreSQL 8.04 Volume IV. Reference	paperback	1-59682-161-2	978-1-59682-161-3
		eBook (pdf)	1-59682-166-3	978-1-59682-166-8
	PostgreSQL 8.04 Volume V. Internals & Appendixes	paperback	1-59682-162-0	978-1-59682-162-0
		eBook (pdf)	1-59682-167-1	978-1-59682-167-5
http://www.linlibrary.com/postgresql/				




Linlibrary Advertising Club



The Apache Software Foundation Official Documentation

<http://www.apache.org/>

Version	Title	Edition	ISBN- 10	ISBN- 13
Apache Web Server 2.2	Apache HTTP Server 2.2 Vol.I. Server Administration	paperback	1-59682-191-4	978-1-59682-191-0
		eBook (pdf)	1-59682-195-7	978-1-59682-195-8
	Apache HTTP Server 2.2 Vol.II. Security & Server Programs	paperback	1-59682-192-2	978-1-59682-192-7
		eBook (pdf)	1-59682-196-5	978-1-59682-196-5
	Apache HTTP Server 2.2 Vol.III. Modules (A-H)	paperback	1-59682-193-0	978-1-59682-193-4
		eBook (pdf)	1-59682-197-3	978-1-59682-197-2
	Apache HTTP Server 2.2 Vol.IV. Modules (I-V)	paperback	1-59682-194-9	978-1-59682-194-1
		eBook (pdf)	1-59682-198-1	978-1-59682-198-9
http://www.linbrary.com/apache-http/				

Version	Title	Edition	ISBN- 10	ISBN- 13
Subversion 1.6	Subversion 1.6 Version Control with Subversion	paperback	1-59682-169-8	978-1-59682-169-9
		eBook (pdf)	1-59682-170-1	978-1-59682-170-5
				
http://www.linbrary.com/subversion/				

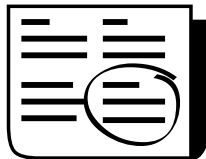


Linbrary Advertising Club

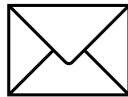


Linbrary Advertising Club

Your Advertising Here



More Books Coming Soon!!!



Please Feel Free to Contact Us at

production@fultus.com