Mastering PostgreSQL 18: A Comprehensive Guide

This book provides an in-depth look into PostgreSQL 18, offering a complete walkthrough from basic concepts to advanced functionalities. Perfect for database administrators, developers, and anyone looking to harness the full power of PostgreSQL.

Table of Contents

Appendices

PostgreSQL: Documentation: 18: 1. What Is PostgreSQL?

Navigation

News

- **May 8, 2025:** PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released!
- **PostgreSQL 18 Beta 1 Released!** Read more

Documentation Links

PostgreSQL 18

Documentation → PostgreSQL 18

Supported Versions

- Current (17, 16, 15, 14, 13)
- 16
- 15
- 14
- 13

Development Versions

- 18
- devel

Unsupported Versions

- 12
- 11
- 10
- 9.6
- 9.5
- 9.4
- 9.3
- 9.2
- 9.1
- 9.0
- 8.4
- 8.3
- 8.2

Note: This documentation is for an unsupported version of PostgreSQL. You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Section: 1. What Is PostgreSQL?

---

1. What Is PostgreSQL?

PostgreSQL is an object-relational database management system (ORDBMS) based on POSTGRES, Version 4.2, 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.

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
- updatable 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.

Navigation

Footer

2. A Brief History of PostgreSQL

Prev Up Next
2. A Brief History of PostgreSQL Preface Home

2. A Brief History of PostgreSQL

2.1. The Berkeley POSTGRES Project

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 decades of development behind it, *PostgreSQL* is now the most advanced open-source database available anywhere.

Another take on the history presented here can be found in Dr. Joe Hellerstein's paper *"Looking Back at Postgres"* hell18.

2.1. 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 ston86, and the definition of the initial data model appeared in rowe87. The design of

the rule system at that time was described in ston87a. The rationale and architecture of the storage manager were detailed in ston87b.

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 ston90a, was released to a few external users in June 1989. In response to a critique of the first rule system (ston89), the rule system was redesigned (ston90b), 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 IBM), which is now owned by IBM, picked up the code and commercialized it. In late 1992, *POSTGRES* became the primary data manager for the Sequoia 2000 scientific computing project described in ston92.

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.

2.2. 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.

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). (Interface library libpq was named after PostQUEL.) 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).

2.3. 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.

Postgres is still considered an official project name, both because of tradition and because people find it easier to pronounce *Postgres* than *PostgreSQL*.

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 each *PostgreSQL* release since then can be found at https://www.postgresql.org/docs/release/.

Navigation

Prev	Up	Next
1. What Is PostgreSQL?	Home	3. Conventions

Footer

PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

3. Conventions

The following conventions are used in the synopsis of a command: brackets (`[` and `]`) indicate optional parts. Brace (`{` and `}`) and vertical lines (`|`) indicate that you must choose one alternative. Dots (`...`) mean that the preceding element can be repeated. All other symbols, including parentheses, should be taken literally.

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.

Prev	Up	Next
2. A Brief History of PostgreSQL	Preface	4. Further Information

---

---

3. Conventions	Prev	Up	Next
	history.html	preface.html	resources.html

Note: The original content contains many nested divs, navigation, and links that are preserved here as plain Markdown. Due to the complex HTML structure, some visual formatting is simplified.

PostgreSQL: Documentation: 18: 4. Further Information

Navigation Menu

- Home
- About
- Download
- Documentation
- Community
- Developers
- Support
- Donate
- Your account

Latest News

- May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released!
- PostgreSQL 18 Beta 1 Released!

Documentation Navigation

Documentation → PostgreSQL 18

Supported Versions:

- Current (17)
- 16
- 15
- 14
- 13

Development Versions:

- 18
- devel

Unsupported Versions:

- 12
- 11
- 10
- 9.6
- 9.5
- 9.4
- 9.3
- 9.2
- 9.1
- 9.0
- 8.4
- 8.3
- 8.2
- 8.1
- 8.0
- 7.4
- 7.3
- 7.2
- 7.1

Important Notice

Further Information

---

4. Further Information

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

Resources

- **Wiki**
  The PostgreSQL wiki contains the project's FAQ (Frequently Asked Questions) list, TODO list, and detailed information about many more topics.

- **Web Site**
  The PostgreSQL web site 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, share experiences with other users, and contact the developers. Consult the PostgreSQL web site for details.

- **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.

Navigation

Additional Notes

3. **Conventions**

Footer

PostgreSQL: Documentation: 18: 5. Bug Reporting Guidelines

Navigation

Bug Reporting Guidelines

5.1. Identifying Bugs

When you find a bug in *PostgreSQL*, we want to hear about it. Your bug reports are essential 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 help you form bug reports that can be handled effectively. While not mandatory, following them is generally advantageous.

We cannot promise to fix every bug immediately. Critical bugs affecting many users or obvious issues are prioritized. Sometimes we tell you to update to a newer version to see if the bug persists, or decide it cannot be fixed until a major rewrite is done. If you need immediate help, consider obtaining a commercial support contract.

5.1. Recognizing Bugs

Before reporting, please verify the documentation to ensure what you are trying is correct. If the documentation is unclear or a program behaves differently than described, that is a bug.

Circumstances considered bugs include, but are not limited to:

- Program terminates with a fatal signal or operating system error message pointing to a problem.
- Produces wrong output.
- Refuses valid input.
- Accepts invalid input without error.
- Fails to compile, build, or install on supported platforms.

"Program" refers to any executable, not only the backend process.

Slow or resource-consuming behavior is not necessarily a bug. Check documentation or ask on mailing lists for tuning advice. Non-compliance with SQL standards is not always a bug unless explicitly claimed.

Always review the TODO list and FAQ for known issues before proceeding.

5.2. What to Report

The key is to state facts clearly. Avoid speculation or vague descriptions.

Every bug report should include:

- **Exact steps to reproduce** from start-up, self-contained, including necessary SQL commands and data setup.
- **Output received**, with error messages and logs if applicable.
- **Expected output**.
- **Command-line options** and configuration details.
- **Differences from standard installation.**

- **PostgreSQL version** (via `SELECT version();` or `--version`).
- **Platform info** (kernel, libraries, hardware).

Be thorough: lengthy reports are better than missing crucial info. Consider consulting this article for more tips.

5.3. Where to Report Bugs

- Send reports to the mailing list: `<pgsql-bugs@lists.postgresql.org>`.
- Use the web form at PostgreSQL bug report page.
- For security issues, report privately to `<security@postgresql.org>`.
- Do **not** send bug reports to user lists like `<pgsql-sql@lists.postgresql.org>` or `<pgsql-general@lists.postgresql.org>`.
- Do **not** send reports to `<pgsql-hackers@lists.postgresql.org>` unless needed for further review.
- Documentation bugs should be reported to `<pgsql-docs@lists.postgresql.org>`.
- Porting issues go to `<pgsql-hackers@lists.postgresql.org>`.

Note: Due to spam, all mailing lists are moderated, so there may be delays. To subscribe, visit https://lists.postgresql.org/.

End of Page

PostgreSQL: Documentation: 18: Chapter 1. Getting Started

Chapter 1. Getting Started

Table of Contents

PostgreSQL: Documentation: 18: Chapter 2. The SQL Language

Documentation → PostgreSQL 18

Supported Versions

Current (17)

/

16
/

15
/

14
/

13

Development Versions

18
/

devel

Unsupported versions

12
/

11
/

10
/

9.6
/

9.5
/

9.4
/

Chapter 2. The SQL Language

Table of Contents

Prev Up Next

1.4. Accessing a Database Home 2.1. Introduction

Footer

PostgreSQL: Documentation: 18: Chapter 3. Advanced Features

Documentation Navigation

Development Versions:

- 18
- devel

Unsupported versions:

- 12
- 11
- 10
- 9.6
- 9.5
- 9.4
- 9.3
- 9.2
- 9.1
- 9.0
- 8.4
- 8.3
- 8.2
- 8.1
- 8.0
- 7.4
- 7.3
- 7.2

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 3. Advanced Features

Table of Contents

Footer

PostgreSQL: Documentation: 18: Chapter 4. SQL Syntax

- 9.3
- 9.2
- 9.1
- 9.0
- 8.4
- 8.3
- 8.2
- 8.1
- 8.0
- 7.4
- 7.3
- 7.2
- 7.1

Chapter 4. SQL Syntax

Chapter 4. SQL Syntax

Table of Contents

This chapter describes the syntax of SQL.

It forms the foundation for understanding the following chapters which will go into detail about how SQL commands are applied to define and modify data.

We also advise users who are already familiar with SQL to read this chapter carefully because it contains several rules and concepts that are implemented inconsistently among SQL databases or that are specific to *PostgreSQL*.

Support Note

the other supported versions listed above instead.

Footer

Privacy Policy | Code of Conduct | About PostgreSQL | Contact

---

Chapter 5. Data Definition

Table of Contents

This chapter covers how one creates the database structures that will hold one's data. In a relational database, the raw data is stored in tables, so the majority of this chapter is devoted to explaining how tables are created and modified and what features are available to control what data is stored in the tables. Subsequently, we discuss how tables can be organized into schemas, and how privileges can be assigned to tables. Finally, we will briefly look at other features that affect the data storage, such as inheritance, table partitioning, views, functions, and triggers.

Prev	Up	Next
4.3. Calling Functions	Table Basics	Basic Table Definitions

PostgreSQL: Documentation: 18: Chapter 14. Performance Tips

Navigation Menu

Latest News

May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Documentation Navigation

Supported Versions:

Current (17) / 16 / 15 / 14 / 13

Development Versions:

18 / devel

Unsupported versions:

12 / 11 / 10 / 9.6 / 9.5 / 9.4 / 9.3 / 9.2 / 9.1 / 9.0 / 8.4 / 8.3 / 8.2 / 8.1 / 8.0 / 7.4 / 7.3 / 7.2 / 7.1

Note

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 14. Performance Tips

Navigation Buttons

Prev
Up
Next

Chapter 14. Performance Tips

Table of Contents

Performance Tips Details

Query performance can be affected by many things. Some of these can be controlled by the user, while others are fundamental to the underlying design of the system. This chapter provides some hints about understanding and tuning *PostgreSQL* performance.

Navigation

Footer

PostgreSQL: Documentation: 18: Chapter 15. Parallel Query

Navigation

Latest News:

PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Breadcrumbs

Supported Versions

Development Versions

Unsupported Versions

12, 11, 10, 9.6

Important Notice

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 15. Parallel Query

Chapter 15. Parallel Query Content

Table of Contents

Section: 15.1 — How Parallel Query Works

PostgreSQL can devise query plans that leverage multiple CPUs to answer queries faster. This feature is known as *parallel query*. Many queries cannot benefit from parallel query, either due to limitations of the current implementation or because there is no faster query plan than the serial version. However, when beneficial, the speedup can be significant—sometimes more than four times faster. Queries touching large data amounts but returning few rows to the user tend to benefit most.

This chapter explains how parallel query works and when it can be used, helping users understand what to expect.

Navigation Summary

Prev	Up	Next
non-durability.html	sql.html	how-parallel-query-works.html
Previous Section	Home	Next Section
14.5. Non-Durable Settings	Home	15.1. How Parallel Query Works

Footer

PostgreSQL: Documentation: 18: Chapter 6. Data Manipulation

May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

- 10
- 9.6
- 9.5
- 9.4
- 9.3
- 9.2
- 9.1
- 9.0
- 8.4
- 8.3
- 8.2
- 8.1
- 8.0
- 7.4
- 7.3

Chapter 6. Data Manipulation

Table of Contents

The previous chapter discussed how to create tables and other structures to hold your data. Now it is time to fill the tables with data. This chapter covers how to insert, update, and delete table data. The chapter after this will finally explain how to extract your long-lost data from the database.

PostgreSQL: Documentation: 18: Chapter 7. Queries

Navigation

Recent News

May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Documentation for PostgreSQL 18

Supported Versions:

- 16
- 15
- 14
- 13

Development Versions:

- 18
- devel

Unsupported Versions:

- 12
- 11
- 10
- 9.6
- 9.5
- 9.4
- 9.3
- 9.2
- 9.1
- 9.0
- 8.4
- 8.3
- 8.2
- 8.1
- 8.0
- 7.4
- 7.3
- 7.2
- 7.1

Warning

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 7. Queries

Navigation

Prev Up Next

Previous Home Next

Sections

queries-values.html

7.8 `WITH` Queries (Common Table Expressions)

queries-with.html

Content

The previous chapters explained how to create tables, how to fill them with data, and how to manipulate that data. Now we finally discuss how to retrieve the data from the database.

Navigation

Prev	Home	Next
6.4. Returning Data from Modified Rows	Home	7.1. Overview

Footer

Chapter 8. Data Types

Table of Contents

PostgreSQL has a rich set of native data types available to users. Users can add new types to PostgreSQL using the CREATE TYPE command.

Table 8.1 shows all the built-in general-purpose data types. Most of the alternative names listed in the *"Aliases"* column are the names used internally by PostgreSQL for historical reasons. In addition, some internally used or deprecated types are available, but are not listed here.

Table 8.1. Data Types

Name	Aliases	Description
`bigint`	`int8`	signed eight-byte integer
`bigserial`	`serial8`	autoincrementing eight-byte integer
`bit [ (<em>n</em>) ]`		fixed-length bit string
`bit varying [ (<em>n</em>) ]`	`varbit [ (<em>n</em>) ]`	variable-length bit string

Name	Aliases	Description
`boolean`	`bool`	logical Boolean (true/false)
`box`		rectangular box on a plane
`bytea`		binary data (“byte array”)
`character [ (<em>n</em>) ]`	`char [ (<em>n</em>) ]`	fixed-length character string
`character varying [ (<em>n</em>) ]`	`varchar [ (<em>n</em>) ]`	variable-length character string
`cidr`		IPv4 or IPv6 network address
`circle`		circle on a plane
`date`		calendar date (year, month, day)
`double precision`	`float8`	double precision floating-point number (8 bytes)
`inet`		IPv4 or IPv6 host address
`integer`	`int`, `int4`	signed four-byte integer
`interval [ <em>fields</em> ] [ (<em>p</em>) ]`		time span
`json`		textual JSON data
`jsonb`		binary JSON data, decomposed
`line`		infinite line on a plane
`lseg`		line segment on a plane
`macaddr`		MAC (Media Access Control) address

Name	Aliases	Description
`macaddr8`		MAC (Media Access Control) address (EUI-64 format)
`money`		currency amount
`numeric [ (<em>p</em>, <em>s</em>) ]`	`decimal [ (<em>p</em>, <em>s</em>) ]`	exact numeric of selectable precision
`path`		geometric path on a plane
`pg_lsn`		PostgreSQL Log Sequence Number
`pg_snapshot`		user-level transaction ID snapshot
`point`		geometric point on a plane
`polygon`		closed geometric path on a plane
`real`	`float4`	single precision floating-point number (4 bytes)
`smallint`	`int2`	signed two-byte integer
`smallserial`	`serial2`	autoincrementing two-byte integer
`serial`	`serial4`	autoincrementing four-byte integer
`text`		variable-length character string
`time [ (<em>p</em>) ] [ without time zone ]`		time of day (no time zone)
`time [ (<em>p</em>) ] with`	`timetz`	time of day, including time zone

Name	Aliases	Description
`time zone`		
`timestamp [ (<em>p</em>) ] [ without time zone ]`		date and time (no time zone)
`timestamp [ (<em>p</em>) ] with time zone`	`timestamptz`	date and time, including time zone
`tsquery`		text search query
`tsvector`		text search document
`txid_snapshot`		user-level transaction ID snapshot (deprecated; see `pg_snapshot`)
`uuid`		universally unique identifier
`xml`		XML data

Compatibility

The following types (or spellings thereof) are specified by SQL: `bigint`, `bit`, `bit varying`, `boolean`, `char`, `character varying`, `character`, `varchar`, `date`, `double precision`, `integer`, `interval`, `numeric`, `decimal`, `real`, `smallint`, `time` (with or without time zone), `timestamp` (with or without time zone), `xml`.

Each data type has an external representation determined by its input and output functions. Many of the built-in types have obvious external formats. However, several types are either unique to PostgreSQL, such as geometric paths, or have several possible formats, such as the date and time types. Some of the input and output functions are not invertible, i.e., the result of an output function might lose accuracy when compared to the original input.

PostgreSQL: Documentation: 18: Chapter 9. Functions and Operators

---

Navigation

Content

Overview

PostgreSQL provides a large number of functions and operators for the built-in data types. This chapter describes most of them, although additional special-purpose functions appear in relevant sections of the manual. Users can also define their own functions and operators, as described in Part V. The `psql` commands `\df` and `\do` can be used to list all available functions and operators, respectively.

The notation used throughout this chapter to describe the argument and result data types of a function or operator is like this:

```
repeat (text, integer) → text
```

which says that the function `repeat` takes one `text` and one `integer` argument and returns a result of type `text`. The right arrow is also used to indicate the result of an example, thus:

```
repeat('Pg', 4) → PgPgPgPg
```

If you are concerned about portability then note that most of the functions and operators described in this chapter, with the exception of the most trivial arithmetic and comparison operators and some explicitly marked functions, are not specified by the SQL standard. Some of this extended functionality is present in other SQL database management systems, and in many cases this functionality is compatible and consistent between the various implementations.

Footer Navigation

Prev	Up	Next
datatype-pseudo.html	sql.html	functions-logical.html

8.21. Pseudo-Types Home 9.1. Logical Operators

Footer

Copyright

Chapter 10. Type Conversion

Chapter 10. Type Conversion

Table of Contents

SQL statements can, intentionally or not, require the mixing of different data types in the same expression. *PostgreSQL* has extensive facilities for evaluating mixed-type expressions.

In many cases a user does not need to understand the details of the type conversion mechanism. However, implicit conversions done by *PostgreSQL* can affect the results of a query. When necessary, these results can be tailored by using *explicit* type conversion.

This chapter introduces the *PostgreSQL* type conversion mechanisms and conventions. Refer to the relevant sections in Chapter 8 and Chapter 9 for more information on specific data types and allowed functions and operators.

Prev **Up** **Next**

9.31. Statistics Information Functions Home 10.1. Overview

PostgreSQL Document Version: 18

Chapter 11. Indexes

Table of Contents

Indexes are a common way to enhance database performance. An index allows the database server to find and retrieve specific rows much faster than it could do without an index. But indexes also add overhead to the database system as a whole, so they should be used sensibly.

Prev	Up	Next
10.6. `SELECT` Output Columns	Home	11.1. Introduction

Part II. The SQL Language

Chapter	Description
Chapter 11. Indexes	Part of PostgreSQL documentation

Indexes are a common way to enhance database performance. An index allows the database server to find and retrieve specific rows much faster than it could do without an index. But indexes also add overhead to the database system as a whole, so they should be used sensibly.

Previous | Up | Next
10.6. `SELECT` Output Columns | Home | 11.1. Introduction

PostgreSQL: Documentation: 18: Chapter 12. Full Text Search

News

- **May 8, 2025:** PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released!
- PostgreSQL 18 Beta 1 Released!

Navigation

Chapter 12. Full Text Search

Table of Contents

Navigation

Prev	Up	Next
11.12. Examining Index Usage	12.1. Introduction	12.1. Introduction

---

This documentation is for an unsupported version of PostgreSQL.

You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Footer

Documentation → PostgreSQL 18

Chapter 13. Concurrency Control

Table of Contents

This chapter describes the behavior of the *PostgreSQL* database system when two or more sessions try to access the same data at the same time. The goals in that situation are to allow efficient access for all sessions while maintaining strict data integrity. Every developer of database applications should be familiar with the topics covered in this chapter.

Chapter 13. Concurrency Control

Note: The above content contains links that are placeholders (e.g., `https://example.com/...`) for demonstration purposes.

PostgreSQL: Documentation: 18: Chapter 16. Installation from Binaries

---

Documentation → PostgreSQL 18

Supported Versions:

- Current (17))
- 16
- 15
- 14

Development Versions:

- 18
- devel

Chapter 16. Installation from Binaries

PostgreSQL is available in the form of binary packages for most common operating systems today. When available, this is the recommended way to install PostgreSQL for users of the system. Building from source (see Chapter 17) is only recommended for people developing PostgreSQL or extensions.

For an updated list of platforms providing binary packages, please visit the download section on the PostgreSQL website and follow the instructions for the specific platform.

PostgreSQL: Documentation: 18: Chapter 17. Installation from Source Code

---

Navigation

Latest News

- PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released!
- PostgreSQL 18 Beta 1 Released!

Breadcrumb Navigation

Supported Versions

Current: Current (17) / 16 / 15 / 14 / 13 / Supporting Versions List

Development Versions: 18 / devel

Unsupported versions: 12 / 11 / 10 / 9.6 / 9.5 / 9.4 / 9.3 / 9.2 / 9.1 / 9.0 / 8.4 / 8.3 / 8.2 / 8.1 / 8.0 / 7.4 / 7.3 / 7.2 / 7.1

Notice

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 17. Installation from Source Code

Chapter 17. Installation from Source Code

Table of Contents

Installation Notice

This chapter describes the installation of *PostgreSQL* using the source code distribution. If you are installing a pre-packaged distribution, such as an RPM or Debian package, ignore this chapter and see Chapter 16 instead.

Navigation

Prev	Up	Next
install-binaries.html	admin.html	install-requirements.html

Chapter	Title
16	Installation from Binaries

Chapter 17. Installation from Source Code

This chapter describes the installation of **PostgreSQL** using the source code distribution.
If you are installing a pre-packaged distribution, such as an RPM or Debian package, ignore this chapter and see **Chapter 16** instead.

Table of Contents

Source Code Installation

This chapter describes the installation of *PostgreSQL* using the source code distribution. If you are installing a pre-packaged distribution, such as an RPM or Debian package, ignore this chapter and see Chapter 16 instead.

Navigation

Footer

Links

- Privacy Policy
- Code of Conduct
- About PostgreSQL
- Contact

PostgreSQL: Documentation: 18: Chapter 26. High Availability, Load Balancing, and Replication

Chapter 26. High Availability, Load Balancing, and Replication

Table of Contents

Database servers can work together to allow a second server to take over quickly if the primary server fails (high availability), or to allow several computers to serve the same data (load balancing). Ideally, database servers could work together seamlessly. Web servers serving static web pages can be combined quite easily by merely load-balancing web requests to multiple machines. In fact, read-only database servers can be combined relatively easily too. Unfortunately, most database servers have a read/write mix of requests, and read/write servers are much harder to combine. This is because though read-only data needs to be placed on each server only once, a write to any server has to be propagated to all servers so that future read requests to those servers return consistent results.

This synchronization problem is the fundamental difficulty for servers working together. Because there is no single solution that eliminates the impact of the sync problem for all use cases, there are multiple solutions. Each solution addresses this problem in a different way, and minimizes its impact for a specific workload.

Some solutions deal with synchronization by allowing only one server to modify the data. Servers that can modify data are called *read/write*, *master* or *primary* servers. Servers that track changes in the primary are called *standby* or *secondary* servers. A standby server that cannot be connected to until it is promoted to a primary server is called a *warm standby* server, and one that can accept connections and serves read-only queries is called a *hot standby* server.

Some solutions are synchronous, meaning that a data-modifying transaction is not considered committed until all servers have

committed the transaction. This guarantees that a failover will not lose any data and that all load-balanced servers will return consistent results no matter which server is queried. In contrast, asynchronous solutions allow some delay between the time of a commit and its propagation to the other servers, opening the possibility that some transactions might be lost in the switch to a backup server, and that load balanced servers might return slightly stale results. Asynchronous communication is used when synchronous would be too slow.

Solutions can also be categorized by their granularity. Some solutions can deal only with an entire database server, while others allow control at the per-table or per-database level.

Performance must be considered in any choice. There is usually a trade-off between functionality and performance. For example, a fully synchronous solution over a slow network might cut performance by more than half, while an asynchronous one might have a minimal performance impact.

The remainder of this section outlines various failover, replication, and load balancing solutions.

Prev Up Next

Chapter 26. High Availability, Load Balancing, and Replication

Table of Contents

PostgreSQL: Documentation: 18: Chapter 27. Monitoring Database Activity

---

Navigation

Latest News

PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Documentation Navigation

Note

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 27. Monitoring Database Activity

Chapter 27. Monitoring Database Activity

Table of Contents

Finding Out What the System Is Doing

A database administrator often wonders, *"What is the system doing right now?"* This chapter discusses how to find that out.

Several tools are available for monitoring database activity and analyzing performance. Most of this chapter is devoted to describing PostgreSQL's cumulative statistics system, but one should not neglect regular Unix monitoring programs such as `ps`, `top`, `iostat`, and `vmstat`. Also, once one has identified a poorly-performing query, further investigation might be needed using PostgreSQL's EXPLAIN command. Section 14.1 discusses `EXPLAIN` and other methods for understanding the behavior of an individual query.

Navigation

Prev | Up | Next

26.4. Hot Standby | Home | 27.1. Standard Unix Tools

Footer

PostgreSQL: Documentation: 18: Chapter 28. Reliability and the Write-Ahead Log

---

Navigation

Latest News

- PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released!
- PostgreSQL 18 Beta 1 Released!

Documentation Overview

Note

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 28. Reliability and the Write-Ahead Log

< Prev	Up	Home	Next >
diskusage.html	admin.html	Part III. Server Administration	wal-reliability.html

---

Chapter Overview

Chapter 28. Reliability and the Write-Ahead Log

Table of Contents

This chapter explains how to control the reliability of PostgreSQL, including details about the Write-Ahead Log.

Navigation

< Prev	Up	Next >
diskusage.html	admin.html	wal-reliability.html
27.6. Monitoring Disk Usage	Home	28.1. Reliability
Previous		Next

Footer

PostgreSQL: Documentation: 18: Chapter 29. Logical Replication

Chapter 29. Logical Replication

Chapter 29. Logical Replication

Table of Contents

Logical replication is a method of replicating data objects and their changes, based upon their **replication identity** (usually a primary key). We use the term *logical* in contrast to **physical** replication, which uses exact block addresses and byte-by-byte replication. PostgreSQL supports both mechanisms concurrently, see Chapter 26. Logical replication allows fine-grained control over both data replication and security.

Logical replication uses a *publish* and *subscribe* model with one or more *subscribers* subscribing to one or more *publications* on a *publisher* node.

Subscribers pull data from the publications they subscribe to and may subsequently re-publish data to allow cascading replication or more complex configurations.

When logical replication of a table typically starts, PostgreSQL takes a snapshot of the table's data on the publisher database and copies it to the subscriber. Once complete, changes on the publisher since the initial copy are sent continually to the subscriber. The subscriber applies the data in the same order as the publisher so that transactional consistency is guaranteed for publications within a single subscription. This method of data replication is sometimes referred to as transactional replication.

The typical use-cases for logical replication are:

- Sending incremental changes in a single database or a subset of a database to subscribers as they occur.
- Firing triggers for individual changes as they arrive on the subscriber.
- Consolidating multiple databases into a single one (for example for analytical purposes).
- Replicating between different major versions of PostgreSQL.
- Replicating between PostgreSQL instances on different platforms (for example Linux to Windows).
- Giving access to replicated data to different groups of users.
- Sharing a subset of the database between multiple databases.

The subscriber database behaves in the same way as any other PostgreSQL instance and can be used as a publisher for other databases by defining its own publications. When the subscriber is treated as read-only by application, there will be no conflicts from a single subscription. On the other hand, if there are other writes done either by an application or by other subscribers to the same set of tables, conflicts can arise.

Prev | Up | Next
28.6. WAL Internals | Home | 29.1. Publication

PostgreSQL: Documentation: 18: Chapter 30. Just-in-Time Compilation (JIT)

Navigation

Latest News

May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Documentation Navigation

Unsupported versions:

12 / 11

Notice

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 30. Just-in-Time Compilation (JIT)

Chapter 30. Just-in-Time Compilation (JIT)

Table of Contents

This chapter explains what just-in-time compilation is, and how it can be configured in *PostgreSQL*.

Navigation

Footer

Chapter 18. Server Setup and Operation

Table of Contents

This chapter discusses how to set up and run the database server, and its interactions with the operating system.

The directions in this chapter assume that you are working with plain *PostgreSQL* without any additional infrastructure, for example a copy that you built from source according to the directions in the preceding chapters. If you are working with a pre-packaged or vendor-supplied version of *PostgreSQL*, it is likely that the packager has made special provisions for installing and starting the database server according to your system's conventions. Consult the package-level documentation for details.

Prev | Up | Next

17.7. Platform-Specific Notes

---

the other supported versions listed above instead.

Chapter 18. Server Setup and Operation

Table of Contents

This chapter discusses how to set up and run the database server, and its interactions with the operating system.

The directions in this chapter assume that you are working with plain *PostgreSQL* without any additional infrastructure, for example a copy that you built from source according to the directions in the preceding chapters. If you are working with a pre-packaged or vendor-supplied version of *PostgreSQL*, it is likely that the packager has made special provisions for installing and starting the database server according to your system's conventions. Consult the package-level documentation for details.

Prev | Up | Next

PostgreSQL: Documentation: 18: Chapter 19. Server Configuration

---

Navigation Menu

News

May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Breadcrumb Navigation

Supported Versions

Current:

Current, (17), 16, 15, 14, 13

Development Versions:

18 / devel

Unsupported versions:

12, 11, 10, 9.6, 9.5, 9.4, 9.3, 9.2, 9.1, 9.0, 8.4, 8.3, 8.2, 8.1, 8.0, 7.4, 7.3, 7.2, 7.1

Note

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 19. Server Configuration

Navigation

Prev	Up	Next
Event Log Registration on Windows	Admin Setting Parameters	

Chapter Title

Chapter 19. Server Configuration

Table of Contents

Paragraph

There are many configuration parameters that affect the behavior of the database system. In the first section of this chapter, we describe how to interact with configuration parameters. The subsequent sections discuss each parameter in detail.

Navigation

Footer

PostgreSQL: Documentation: 18: Chapter 20. Client Authentication

---

Documentation → PostgreSQL 18

Chapter 20. Client Authentication

Table of Contents

When a client application connects to the database server...

It specifies which PostgreSQL database user name it wants to connect as, much the same way one logs into a Unix computer as a particular user. Within the SQL environment, the active database user name

determines access privileges to database objects — see Chapter 21 for more information. Therefore, it is essential to restrict which database users can connect.

Note

As explained in Chapter 21, PostgreSQL actually manages privileges in terms of *roles*. In this chapter, we use *database user* to mean *role with the* `LOGIN` *privilege*.

Authentication is the process by which the database server establishes the identity of the client, and by extension determines whether the client application (or the user who runs the client application) is permitted to connect with the requested database user name.

PostgreSQL offers a number of different client authentication methods. The method used to authenticate a particular client connection can be selected based on (client) host address, database, and user.

PostgreSQL database user names are logically separate from user names of the operating system in which the server runs. If all the users of a particular server also have accounts on the server's machine, it makes sense to assign database user names that match their operating system user names. However, a server that accepts remote connections might have many database users who have no local OS account, and in such cases, there need be no connection between database user names and OS user names.

Prev	Up	Next
runtime-config-short.html	admin.html	auth-pg-hba-conf.html

Short Options	Home	20.1. The `pg_hba.conf` File
19.18	index.html	20.1. The `pg_hba.conf` File

Footer

PostgreSQL: Documentation: 18: Chapter 21. Database Roles

Navigation

News

PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Breadcrumbs

Warning

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 21. Database Roles

Chapter 21. Database Roles

Table of Contents

PostgreSQL manages database access permissions using the concept of *roles*. A role can be thought of as either a database *user*, or a *group* of database users, depending on how the role is set up. Roles can own database objects (for example, tables and functions) and can assign privileges on those objects to other roles to control who has access to which objects. Furthermore, it is possible to grant *membership* in a role

to another role, thus allowing the member role to use privileges assigned to another role.

The concept of roles subsumes the concepts of *users* and *groups*. In PostgreSQL versions before 8.1, users and groups were distinct kinds of entities, but now there are only roles. Any role can act as a user, a group, or both.

This chapter describes how to create and manage roles. More information about the effects of role privileges on various database objects can be found in Section 5.8.

Navigation

Footer

PostgreSQL: Documentation: 18: Chapter 22. Managing Databases

---


```
  <div>
    <span>
      <i></i>
    </span>
  </div>
</div>
```

May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Supported Versions: - [Current] (https://www.postgresql.org/docs/current/managing-databases.html) (17) - [16](https://www.postgresql.org/docs/16/managing-databases.html) - [15](https://www.postgresql.org/docs/15/managing-databases.html) - [14](https://www.postgresql.org/docs/14/managing-

databases.html) - [13](https://www.postgresql.org/docs/13/managing-databases.html)
Development Versions: - [18]
(https://www.postgresql.org/docs/18/managing-databases.html) - [devel]
(https://www.postgresql.org/docs/devel/managing-databases.html)
Unsupported versions: - [12]
(https://www.postgresql.org/docs/12/managing-databases.html) - [11]
(https://www.postgresql.org/docs/11/managing-databases.html) - [10]
(https://www.postgresql.org/docs/10/managing-databases.html) - [9.6]
(https://www.postgresql.org/docs/9.6/managing-databases.html) - [9.5]
(https://www.postgresql.org/docs/9.5/managing-databases.html) - [9.4]
(https://www.postgresql.org/docs/9.4/managing-databases.html) - [9.3]
(https://www.postgresql.org/docs/9.3/managing-databases.html) - [9.2]
(https://www.postgresql.org/docs/9.2/managing-databases.html) - [9.1]
(https://www.postgresql.org/docs/9.1/managing-databases.html) - [9.0]
(https://www.postgresql.org/docs/9.0/managing-databases.html) - [8.4]
(https://www.postgresql.org/docs/8.4/managing-databases.html) - [8.3]
(https://www.postgresql.org/docs/8.3/managing-databases.html) - [8.2]
(https://www.postgresql.org/docs/8.2/managing-databases.html) - [8.1]
(https://www.postgresql.org/docs/8.1/managing-databases.html) - [8.0]
(https://www.postgresql.org/docs/8.0/managing-databases.html) - [7.4]
(https://www.postgresql.org/docs/7.4/managing-databases.html) - [7.3]
(https://www.postgresql.org/docs/7.3/managing-databases.html) - [7.2]
(https://www.postgresql.org/docs/7.2/managing-databases.html) - [7.1]
(https://www.postgresql.org/docs/7.1/managing-databases.html)
This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the [current]
(https://www.postgresql.org/docs/current/managing-databases.html)
version, or one of the other supported versions listed above instead.

Chapter 22. Managing Databases

Table of Contents

Every instance of a running PostgreSQL server manages one or more databases. Databases are therefore the topmost hierarchical level for organizing SQL objects (“database objects”). This chapter describes the properties of databases, and how to create, manage, and destroy them.

Prev Up Next
21.6. Function Security Home 22.1. Overview

Chapter 23. Localization

Table of Contents

This chapter describes the available localization features from the point of view of the administrator. *PostgreSQL* supports two localization facilities:

- Using the locale features of the operating system to provide locale-specific collation order, number formatting, translated messages, and other aspects. This is covered in Section 23.1 and Section 23.2.
- Providing a number of different character sets to support storing text in all kinds of languages, and providing character set translation between client and server. This is covered in Section 23.3.

Chapter 23. Localization

Table of Contents

23.1. Locale Support

23.1.1. Overview

23.1.2. Behavior

23.1.3. Selecting Locales

23.1.4. Locale Providers

23.1.5. ICU Locales

23.1.6. Problems

23.2. Collation Support

23.2.1. Concepts

23.2.2. Managing Collations

23.2.3. ICU Custom Collations

23.3. Character Set Support

23.3.1. Supported Character Sets

23.3.2. Setting the Character Set

23.3.3. Automatic Character Set Conversion Between Server and Client

23.3.4. Available Character Set Conversions

23.3.5. Further Reading

PostgreSQL supports two localization facilities:

- Using the locale features of the operating system to provide locale-specific collation order, number formatting, translated messages, and other aspects. This is covered in Section 23.1 and Section 23.2.

- Providing a number of different character sets to support storing text in all kinds of languages, and providing character set translation between client and server. This is covered in Section 23.3.

Additional Information

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Navigation:

- Prev
- Up
- Next

PostgreSQL: Documentation: 18: Chapter 24. Routine Database Maintenance Tasks

---

Navigation

Chapter 24. Routine Database Maintenance Tasks

Table of Contents

Important Notice

PostgreSQL, like any database software, requires that certain tasks be performed regularly to achieve optimum performance. The tasks discussed here are *required*, but they are repetitive in nature and can easily be automated using standard tools such as *cron* scripts or Windows' *Task Scheduler*. It is the database administrator's responsibility to set up appropriate scripts, and to check that they execute successfully.

One obvious maintenance task is the creation of backup copies of the data on a regular schedule. Without a recent backup, you have no chance of recovery after a catastrophe (disk failure, fire, mistakenly dropping a critical table, etc.). The backup and recovery mechanisms available in PostgreSQL are discussed at length in Chapter 25.

The other main category of maintenance task is periodic *"vacuuming"* of the database. This activity is discussed in Section 24.1. Closely related to this is updating the statistics that will be used by the query planner, as discussed in Section 24.1.3.

Another task that might need periodic attention is log file management. This is discussed in Section 24.3.

check_postgres is available for monitoring database health and reporting unusual conditions. check_postgres integrates with Nagios and MRTG, but can be run standalone too.

PostgreSQL is low-maintenance compared to some other database management systems. Nonetheless, appropriate attention to these tasks will go far towards ensuring a pleasant and productive experience with the system.

Navigation (continued)

Footer

PostgreSQL: Documentation: 18: Chapter 25. Backup and Restore

---

Navigation Menu

News

- PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Navigation Breadcrumbs

Supported Versions:

- Current (17)
- 16
- 15
- 14
- 13

Development Versions:

- 18
- devel

Unsupported Versions:

- 12
- 11
- 10
- 9.6
- 9.5
- 9.4
- 9.3
- 9.2
- 9.1
- 9.0
- 8.4
- 8.3
- 8.2
- 8.1
- 8.0
- 7.4
- 7.3
- 7.2
- 7.1

Disclaimer

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 25. Backup and Restore

Chapter 25. Backup and Restore Content

Table of Contents

Important Notice

As with everything that contains valuable data, *PostgreSQL* databases should be backed up regularly. While the procedure is essentially simple, it is important to have a clear understanding of the underlying techniques and assumptions.

Approaches to Backing Up PostgreSQL Data:

- SQL dump
- File system level backup
- Continuous archiving

Each has its own strengths and weaknesses; each is discussed in turn in the following sections.

Navigation

Prev	Up	Next
24.3. Log File Maintenance	index.html	25.1. SQL Dump

Footer

PostgreSQL: Documentation: 18: Legal Notice

May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Documentation for PostgreSQL 18

Supported Versions:

- Current (17)
- 16
- 15
- 14

- 13

Development Versions:

- 18
- devel

Unsupported versions:

- 12
- 11
- 10
- 9.6
- 9.5
- 9.4
- 9.3
- 9.2
- 9.1
- 9.0
- 8.4
- 8.3
- 8.2
- 8.1
- 8.0
- 7.4

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Legal Notice

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.

Footer

Chapter 35. The Information Schema

Prev Up Part IV. Client Interfaces Home Next

Chapter 35. The Information Schema

Table of Contents

The information schema consists of a set of views that contain information about the objects defined in the current database. The information schema is defined in the SQL standard and can therefore be expected to be portable and remain stable — unlike the system catalogs, which are specific to *PostgreSQL* and are modeled after implementation concerns. The information schema views do not, however, contain information about *PostgreSQL*-specific features; to inquire about those you need to query the system catalogs or other *PostgreSQL*-specific views.

Note

When querying the database for constraint information, it is possible for a standard-compliant query that expects to return one row to return several. This is because the SQL standard requires constraint names to be unique within a schema, but *PostgreSQL* does not enforce this restriction. *PostgreSQL* automatically-generated constraint names avoid duplicates in the same schema, but users can specify such duplicate names.

This problem can appear when querying information schema views such as `check_constraint_routine_usage`, `check_constraints`, `domain_constraints`, and `referential_constraints`. Some other views have similar issues but contain the table name to help distinguish duplicate rows, e.g.,

`constraint_column_usage`, `constraint_table_usage`, `table_constraints`.

Navigation

Prev	Up	Next
34.17. Internals Home		35.1. The Schema

PostgreSQL: Documentation: 18: Chapter 36. Extending SQL

Navigation

News

May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Documentation for PostgreSQL 18

Chapter 36. Extending SQL

Chapter 36. Extending SQL

Table of Contents

Extending PostgreSQL SQL Query Language

In the sections that follow, we will discuss how you can extend the PostgreSQL SQL query language by adding:

- functions (starting in Section 36.3)
- aggregates (starting in Section 36.12)
- data types (starting in Section 36.13)
- operators (starting in Section 36.14)
- operator classes for indexes (starting in Section 36.16)
- packages of related objects (starting in Section 36.17)

Navigation

Prev | Up | Next
Part V. Server Programming | Home | 36.1. How Extensibility Works

---

Footer

PostgreSQL: Documentation: 18: Chapter 37. Triggers

Navigation

Latest News

- PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Main Content

Documentation Navigation

Important Notice

This documentation is for an unsupported version of PostgreSQL.
You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Chapter 37. Triggers

Chapter 37. Triggers

Table of Contents

General Information

This chapter provides general information about writing trigger functions. Trigger functions can be written in most of the available procedural languages, including *PL/pgSQL* (Chapter 41), *PL/Tcl* (Chapter 42), *PL/Perl* (Chapter 43), and *PL/Python* (Chapter 44). After reading this chapter, you should consult the chapter for your favorite procedural language to find out the language-specific details of writing a trigger in it.

It is also possible to write a trigger function in C, although most people find it easier to use one of the procedural languages. It is not currently possible to write a trigger function in the plain SQL function language.

Navigation

Prev	Up	Next
36.18. Extension Building Infrastructure	Home	37.1. Overview of Trigger Behavior

---

PostgreSQL: Documentation: 18: Chapter 38. Event Triggers

Chapter 38. Event Triggers

Table of Contents

- 38.5. A Database Login Event Trigger Example

To supplement the trigger mechanism discussed in Chapter 37, PostgreSQL also provides event triggers. Unlike regular triggers, which are attached to a single table and capture only DML events, event triggers are global to a particular database and are capable of capturing DDL events.

Like regular triggers, event triggers can be written in any procedural language that includes event trigger support, or in C, but not in plain SQL.

Supporting text

To support the trigger mechanism discussed in Chapter 37, PostgreSQL also provides event triggers. Unlike regular triggers, which are attached to a single table and capture only DML events, event triggers are global to a particular database and are capable of capturing DDL events.

Like regular triggers, event triggers can be written in any procedural language that includes event trigger support, or in C, but not in plain SQL.

Navigation

Prev	Up	Next
37.4. A Complete Trigger Example	Home	38.1. Overview of Event Trigger Behavior

PostgreSQL: Documentation: 18: Chapter 39. The Rule System

Chapter 39. The Rule System

Table of Contents

This chapter discusses the rule system in *PostgreSQL*. Production rule systems are conceptually simple, but there are many subtle points involved in actually using them.

Some other database systems define active database rules, which are usually stored procedures and triggers. In *PostgreSQL*, these can be implemented using functions and triggers as well.

The rule system (more precisely speaking, the query rewrite rule system) is totally different from stored procedures and triggers. It modifies queries to take rules into consideration, and then passes the modified query to the query planner for planning and execution. It is very powerful, and can be used for many things such as query language procedures, views, and versions. The theoretical foundations and the power of this rule system are also discussed in ston90b and ong90.

Prev	Up	Next
event-trigger-database-login-example.html	server-programming.html	querytree.html

38.5. A Database Login Event Trigger Example

PostgreSQL: Documentation: 18: Chapter 40. Procedural Languages

---

Navigation

News

May 8, 2025: PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released! | PostgreSQL 18 Beta 1 Released!

Documentation for PostgreSQL 18

Supported Versions

- Current (17)
- 16
- 15
- 14
- 13

Development Versions

- 18
- devel

Unsupported Versions

- 12
- 11
- 10
- 9.6
- 9.5
- 9.4
- 9.3
- 9.2
- 9.1
- 9.0
- 8.4
- 8.3
- 8.2
- 8.1
- 8.0
- 7.4
- 7.3
- 7.2
- 7.1

Note: This documentation is for an unsupported version of PostgreSQL. You may want to view the same page for the current version, or one of

the other supported versions listed above instead.

Chapter 40. Procedural Languages

Chapter 40. Procedural Languages

Table of Contents

PostgreSQL allows user-defined functions to be written in languages besides SQL and C. These are called *procedural languages* (PLs). For a function written in a procedural language, the database server relies on a handler that understands the language's source text. This handler could perform parsing, syntax analysis, execution, or serve as *glue* between PostgreSQL and an existing language implementation. Handlers are C language functions compiled into shared objects loaded on demand.

Currently, PostgreSQL supports four procedural languages in its standard distribution: **PL/pgSQL** (Chapter 41), **PL/Tcl** (Chapter 42), **PL/Perl** (Chapter 43), and **PL/Python** (Chapter 44). Additional languages can be found in Appendix H, and users can define their own, with guidance in Chapter 57.

Navigation

Footer

PostgreSQL: Documentation: 18: Chapter 41. PL/pgSQL — SQL Procedural Language

---

Navigation

News

- PostgreSQL 17.5, 16.9, 15.13, 14.18, and 13.21 Released!
- PostgreSQL 18 Beta 1 Released!

Main Content

Chapter 41. PL/pgSQL — SQL Procedural Language

Documentation → PostgreSQL 18

Supported Versions:

- Current (17)
- 16, 15, 14, 13, 12, 11, 10, 9.6, 9.5, 9.4, 9.3, 9.2, 9.1, 9.0, 8.4, 8.3, 8.2, 8.1, 8.0

Development Versions:

- 18, devel

Unsupported Versions:

- 12, 11, 10, 9.6, 9.5, 9.4, 9.3, 9.2, 9.1, 9.0, 8.4, 8.3, 8.2, 8.1, 8.0, 7.4, 7.3, 7.2, 7.1

Note: This documentation is for an unsupported version of PostgreSQL. You may want to view the same page for the current version, or one of the other supported versions listed above instead.

Navigation Table

Prev	Up	Next
xplang-install.html	server-programming.html	index.html
40.1. Installing Procedural Languages		41.1. Overview

Chapter Title

Chapter 41. PL/pgSQL — SQL Procedural Language

Table of Contents

Support Links

Prev: Installing Procedural Languages | Up: Server Programming | Next: Chapter 41. Overview

---

Footer