

Let's make it better, now, together!

CREATING STREAMING CHAMPIONS
PIONS CREATING STREAMING CHAM
CHAMPIONS CREATING STREAMING C



13th October 2022: [PostgreSQL 15 Released!](#)

[Documentation](#) → [PostgreSQL 15](#)

Supported Versions: **Current** (15) / [14](#) / [13](#) / [12](#) / [11](#) / [10](#)

Development Versions: [devel](#)

Unsupported versions: [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](#)



PostgreSQL 15.0 Documentation

[Next](#)

PostgreSQL 15.0 Documentation

The PostgreSQL Global Development Group

Copyright © 1996–2022 The PostgreSQL Global Development Group

[Legal Notice](#)

Table of Contents

Preface

1. What Is PostgreSQL?
2. A Brief History of PostgreSQL
3. Conventions
4. Further Information
5. Bug Reporting Guidelines

I. Tutorial

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

II. The SQL Language

4. SQL Syntax
5. Data Definition

BEDROCK STREAMING

- Cutting-edge video streaming platforms to broadcasters and media companies in Europe
- More than 45 million users
- Founded by M6 Group in 2020
- 15 years' experience
- 400+ team members

SARAH HAÏM-LUBCZANSKI

- Documentation Architect at Bedrock Streaming, (*previously Technical Writer, previously PHP Developer*)
- I like Monty Python, riding my bike, eating pizzas, and learn new things everyday.



HOW I GOT HERE

- I read many documentations
- Training my colleagues

DISCLAIMER ABOUT POSTRESQL



I am not using Postgres right now!

POSTGRESQL TO ME, TODAY

- A counter example
- Walls of text <https://www.postgresql.org/docs/15/trigger-definition.html>

AGENDA

- #1 : Documentation today
- #2 : By topic
 - Architecture
 - Content
 - Contribution
- #3 : My conclusion

DOCUMENTATION NOWADAYS

DOC IS A TOOL

- Documentation = tool
- You can do everything yourself, sure!

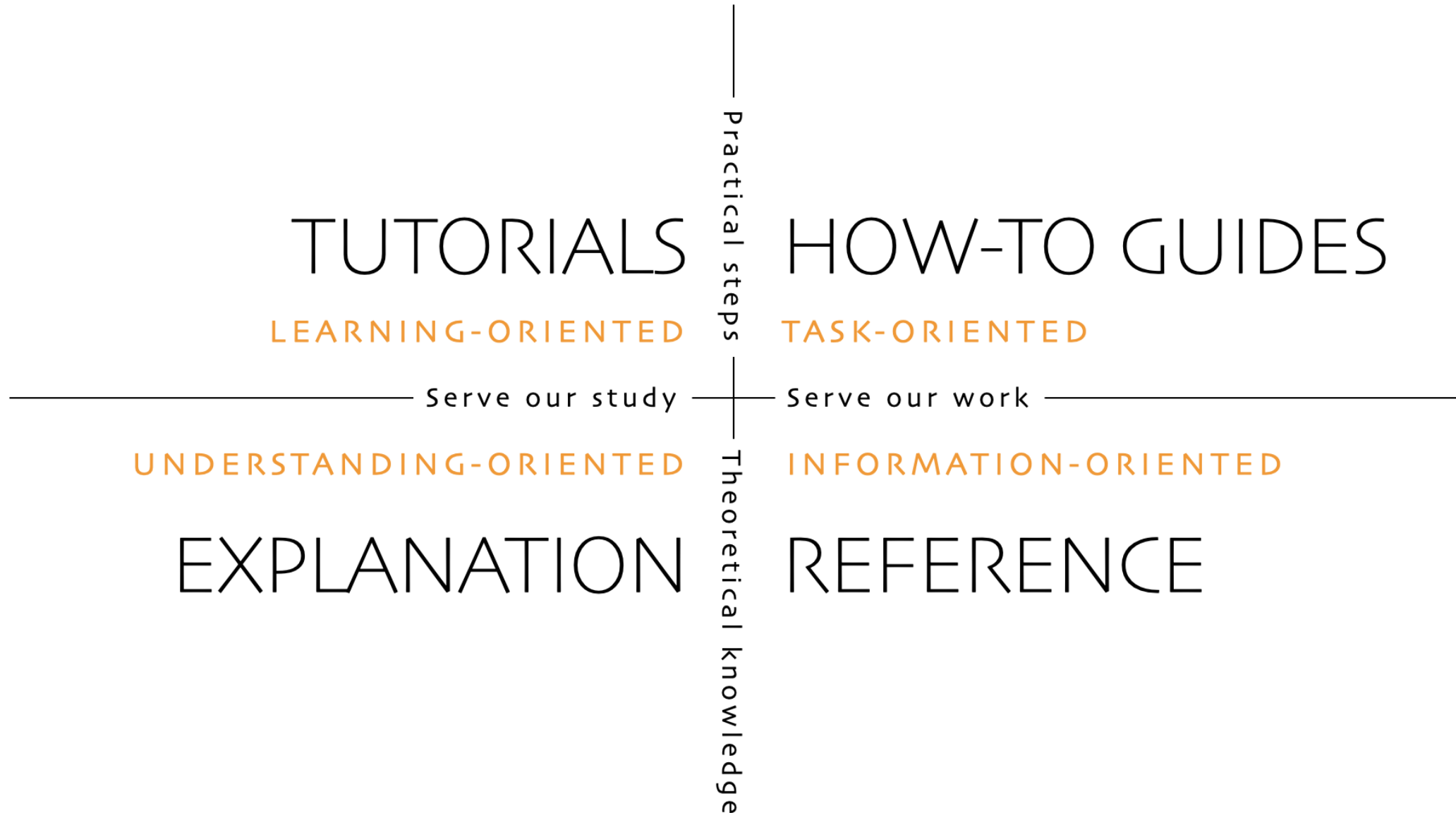


TODAY IN DOC: ARCHITECTURE

- In current documentation best practices

DIATAXIS?

- A framework to tidy your documentation
- <https://www.diataxis.fr>



BENEFITS OF DIATAXIS

- Clarity!
- Guide for documentation writers

EXAMPLE OF WEBSITES USING DIATAXIS

- MDN
- MS SQL Documentation
- Kubernetes Documentation

TODAY IN DOC: UX AND PERSONAS

PERSONAS

- Skills
- Needs
- Experience with your software

AUDIENCE

- It's about the role of the reader

IN THE DOCUMENTATION

- The audience is important
- Who is reading?

SOCIAL MEDIA ERA

- Seek help
- Get advices
- Compare solutions

TWITTER IS MY ADDICTION

- Too often relying on tweets
- But...this time...
- I used it to learn more about PostgreSQL

WHAT DOES TWITTER SAY?

- Good things about you!



Andrew Meredith

@a4w_m6h

...

Postgres is incredible documentation on how to build an RDBMS that happens to come with a production-ready reference implementation and extension framework.

[#PostgreSQL](#)

6:48 PM · Sep 17, 2021 · Twitter for Android



Richard Michael
@richardkmichael

...

It is difficult to overstate what a great experience interacting with the [#postgres](#) OSS community is.. mailing list is friendly & helpful, source code is beautiful & understandable, documentation is thorough & useful. Very motivating.

6:13 PM · Sep 9, 2021 · Twitter Web App

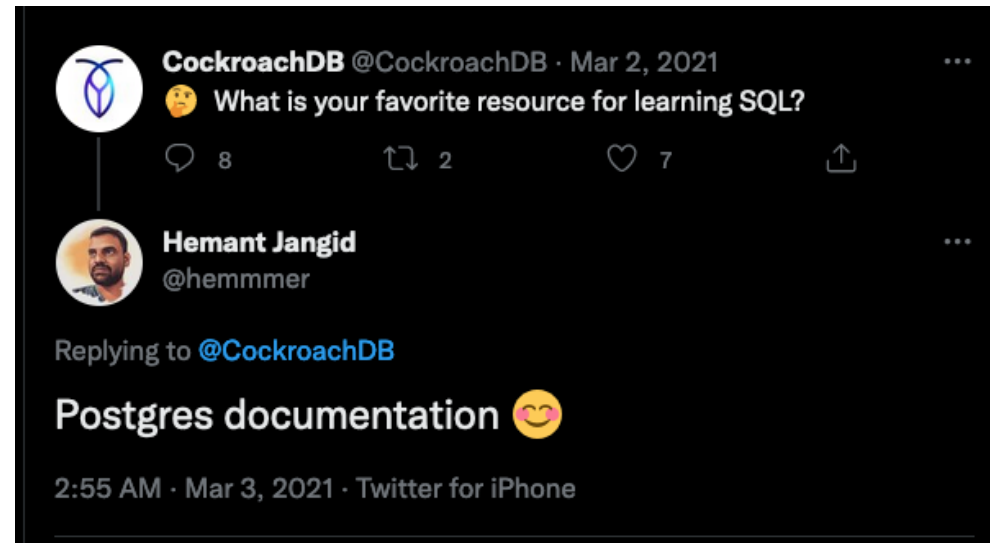


Andrew Meredith
@a4w_m6h



Yep - it's just me. Still over here being amazed with how good the documentation and community for Postgres are.

12:23 AM · Sep 28, 2021 · Twitter Web App





typedfemale
@typedfemale



i live my life in accordance with the postgres
documentation

1:55 AM · Jun 6, 2022 · Twitter for iPhone

NOW, IN POSTGRES DOC

ARCHITECTURE : THE GOOD

- Sections organized
- Many many chapters
 - Already in sections

SECTIONS

- Part I. Tutorial
- Part II. The SQL Language
- Part III. Server Administration
- Part IV. Client Interfaces
- Part V. Server Programming
- Part VI. Reference
- Part VII. Internals

WHEN I SAY MANY CHAPTERS...

Table of Contents

Preface

1. What Is PostgreSQL?
2. A Brief History of PostgreSQL
3. Conventions
4. Further Information
5. Bug Reporting Guidelines

I. Tutorial

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

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
15. Parallel Query

III. Server Administration

16. Installation from Binaries
17. Installation from Source Code
18. Installation from Source Code on Windows
19. Server Setup and Operation
20. Server Configuration
21. Client Authentication
22. Database Roles
23. Managing Databases
24. Localization
25. Routine Database Maintenance Tasks

25. Routine Database Maintenance Tasks
26. Backup and Restore
27. High Availability, Load Balancing, and Replication
28. Monitoring Database Activity
29. Monitoring Disk Usage
30. Reliability and the Write-Ahead Log
31. Logical Replication
32. Just-in-Time Compilation (JIT)
33. Regression Tests

IV. Client Interfaces

34. libpq — C Library
35. Large Objects
36. ECPG — Embedded SQL in C
37. The Information Schema

V. Server Programming

38. Extending SQL
39. Triggers
40. Event Triggers
41. The Rule System
42. Procedural Languages
43. PL/pgSQL — SQL Procedural Language
44. PL/Tcl — Tcl Procedural Language
45. PL/Perl — Perl Procedural Language
46. PL/Python — Python Procedural Language
47. Server Programming Interface
48. Background Worker Processes
49. Logical Decoding
50. Replication Progress Tracking
51. Archive Modules

VI. Reference

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

VII. Internals

52. Overview of PostgreSQL Internals
53. System Catalogs

II. PostgreSQL Client Applications

III. PostgreSQL Server Applications

VII. Internals

52. Overview of PostgreSQL Internals
53. System Catalogs
54. System Views
55. Frontend/Backend Protocol
56. PostgreSQL Coding Conventions
57. Native Language Support
58. Writing a Procedural Language Handler
59. Writing a Foreign Data Wrapper
60. Writing a Table Sampling Method
61. Writing a Custom Scan Provider
62. Genetic Query Optimizer
63. Table Access Method Interface Definition
64. Index Access Method Interface Definition
65. Generic WAL Records
66. Custom WAL Resource Managers
67. B-Tree Indexes
68. GIST Indexes
69. SP-GiST Indexes
70. GIN Indexes
71. BRIN Indexes
72. Hash Indexes
73. Database Physical Storage
74. System Catalog Declarations and Initial Contents
75. How the Planner Uses Statistics
76. Backup Manifest Format

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. Additional Supplied Programs
- H. External Projects

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. Additional Supplied Programs
- H. External Projects
- I. The Source Code Repository
- J. Documentation
- K. PostgreSQL Limits
- L. Acronyms
- M. Glossary
- N. Color Support
- O. Obsolete or Renamed Features

Bibliography

Index

SOME ARE SCARY

- Part V. Server Programming
- What will I program?



ARE ALL OF THESE USEFUL ?

Appendix J. Documentation

Table of Contents

- J.1. DocBook
- J.2. Tool Sets
 - J.2.1. Installation on Fedora, RHEL, and Derivatives
 - J.2.2. Installation on FreeBSD
 - J.2.3. Debian Packages
 - J.2.4. macOS
 - J.2.5. Detection by configure
- J.3. Building the Documentation
 - J.3.1. HTML
 - J.3.2. Manpages
 - J.3.3. PDF
 - J.3.4. Plain Text Files
 - J.3.5. Syntax Check
- J.4. Documentation Authoring
 - J.4.1. Emacs
- J.5. Style Guide
 - J.5.1. Reference Pages

PostgreSQL has four primary documentation formats:

- Plain text, for pre-installation information
- HTML, for on-line browsing and reference
- PDF, for printing

PREFACE

Preface

Table of Contents

- 1. What Is PostgreSQL?
- 2. A Brief History of PostgreSQL
 - 2.1. The Berkeley POSTGRES Project
 - 2.2. Postgres95
 - 2.3. PostgreSQL
- 3. Conventions
- 4. Further Information
- 5. Bug Reporting Guidelines
 - 5.1. Identifying Bugs
 - 5.2. What to Report
 - 5.3. Where to Report Bugs

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

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

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

SQL, IT'S BASIC

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
- 15. Parallel Query

SQL, IT'S EVERYWHERE (1)

37. THE INFORMATION SCHE

V. Server Programming

38. Extending SQL

39. Triggers

SQL, IT'S EVERYWHERE (2)

51. Archive Modules

VI. Reference

I. SQL Commands

ARCHITECTURE : IMPROVEMENTS

- Add section description in the header
- Organize by type of content
- Rename some sections?

5.4. Constraints


Chapter 5. Data Definition

[Home](#)

[Next](#)

AND IN 2000, A RE-ORG WAS DISCUSSED

Documentation organization

From: Peter Eisentraut <peter_e(at)gmx(dot)net>
To: pgsql-docs(at)postgresql(dot)org
Subject: Documentation organization
Date: 2000-06-29 17:28:13
Message-ID: Pine.LNX.4.21.0006291616400.397-100000@localhost.localdomain
Views: [Raw Message](#) | [Whole Thread](#) | [Download mbox](#) | [Resend email](#)
Thread: 2000-06-29 17:28:13 from Peter Eisentraut <peter_e(at)gmx(dot)net> 
Lists: [pgsql-docs](#)

I've been looking into ways to handle the User/Admin/Programmer vs Integrated dichotomy a little better. I think the intergrated document as we know it needs to go. There's too much weirdness in the order (e.g., the tutorial coming after the FE/BE protocol, release notes somewhere in the middle, etc.) that is created by just pasting together the "guides". Also, the chapters are numbered differently, which makes referring to them by number impossible.

TAKE YOUR TIME

- 26 years!
- You don't change it so quickly

LOOKING STACKOVERFLOW : THE BAD





IMPROVEMENT / MORE SUMMARIES

- Maybe some « shortcuts » pages are missing
- A TL;DR ?

DOCUMENTATION TO GO!



Excerpts from this Manual

- MySQL Backup and Recovery
- MySQL Globalization
- MySQL Information Schema
- MySQL Installation Guide
- Security in MySQL
- Starting and Stopping MySQL
- MySQL and Linux/Unix
- MySQL and Windows
- MySQL and macOS
- MySQL and Solaris
- Building MySQL from Source
- MySQL Restrictions and Limitations
- MySQL Partitioning
- MySQL Tutorial
- MySQL Performance Schema
- MySQL Replication
- Using the MySQL Yum Repository
- MySQL NDB Cluster 8.0

UX AND DESIGN

tone and jokes

- The content is clear and very readable.
- Jokes are well done and non-intrusive.

CODE EXAMPLES : SQL

- Clear
- Useful

FORMATTING OF THE CONTENT

Tip

When you create many interrelated tables it is wise to choose a consistent naming pattern for the tables and columns. For instance, there is a choice of using singular or plural nouns for table names, both of which are favored by some theorist or other.

MORE CALLOUTS?

[Products](#)[Solutions](#)[Resources](#)[Company](#)[Pricing](#)

MongoDB Documentation

[← Back To Develop Applications](#)

MongoDB Manual

4.4

► Introduction

► Installation

► The mongo Shell

► MongoDB CRUD Operations

► Aggregation

► Data Models

▼ Transactions

Drivers API

Production Considerations

Production Considerations
(Sharded Clusters)

Transactions and Operations

► Indexes

► Security

► Change Streams

IMPORTANT

In most cases, multi-document transaction incurs a greater performance cost over single document writes, and the availability of multi-document transactions should not be a replacement for effective schema design. For many scenarios, the [denormalized data model \(embedded documents and arrays\)](#) will continue to be optimal for your data and use cases. That is, for many scenarios, modeling your data appropriately will minimize the need for multi-document transactions.

For additional transactions usage considerations (such as runtime limit and oplog size limit), see also [Production Considerations](#).

TIP

See also:

[Outside Reads During Commit](#)

Transactions and Operations

Distributed transactions can be used across multiple operations, collections, databases, documents, and, starting in MongoDB 4.2, shards.

For transactions:

- You can specify read/write (CRUD) operations on **existing** collections. For a list of CRUD

Select your language

C C

On this page

Transactions API

Transactions and Atomicity

Transactions and Operations

Transactions and Sessions

Read Concern/Write Concern/Read Preference

General Information

Additional Transactions Topics

TOO MUCH CLUTTER ?

- A doc about the doc?
- You are not alone

Version

SQL Server 2022 Preview ▾

 Filter by title

SQL Server

Docs navigation tips

Previous versions 2005-2014

> Overview

> Business continuity

> Database design

> Development

> Internals & Architecture

> Installation

> Migrate & load data

> Manage, monitor, & tune

> Query data

> Reporting & Analytics

> Security

> Tools

> Tutorials

 Download PDF

Learn / SQL /

SQL Server docs navigation guide

Article • 08/11/2022 • 2 minutes to read • 5 contributors

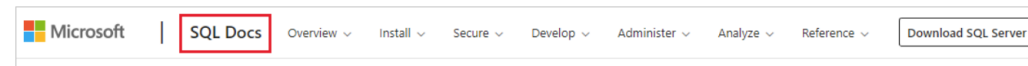
 Feedback

This topic provides some tips and tricks for navigating the SQL Server technical documentation space.

Hub page

The SQL Server hub page can be found at <https://aka.ms/sqldocs> and is the entry point for finding relevant SQL Server content.

You can always navigate back to this page by selecting **SQL Docs** from the header at the top of every page within the SQL Server technical documentation set:



Offline documentation

If you would like to view the SQL Server documentation on an offline system, you have two options to do so. You can either create a PDF wherever you are in the SQL Server technical documentation, or you can download the offline content using [SQL Server offline Help Viewer](#).

In this article

[Hub page](#)[Offline documentation](#)[TOC symbols](#)[TOC search](#)

Show more ▾

VERSIONNING

- It's expected by the readers

VERSIONS : THE GOOD

- Delivered with the software
- Archived versions are accessible

VERSIONS

- Many versions
- Latest is visible
- Why so many versions available?

Manuals

You can view the manual for an older version or download a PDF of a manual from the below table.

Online Version	PDF Version
15 / Current	A4 PDF (13.5 MB) • US PDF (13.4 MB)
14	A4 PDF (13.3 MB) • US PDF (13.2 MB)
13	A4 PDF (12.9 MB) • US PDF (12.8 MB)
12	A4 PDF (12.6 MB) • US PDF (12.5 MB)
11	A4 PDF (12.3 MB) • US PDF (12.2 MB)
10	A4 PDF (12.0 MB) • US PDF (11.9 MB)
Development snapshot	PDF version not available

Looking for documentation for an older, unsupported, version? Check the **archive** of older manuals.

IMPROVEMENT / VERSIONING

- What am I viewing?
- Is it obsolete?
- issue coming from external website

[Documentation](#) → [PostgreSQL 12](#)
Supported Versions: [Current \(15\)](#) / [14](#) / [13](#) / [12](#) / [11](#) / [10](#)
Development Versions: [devel](#)
Unsupported versions: [9.6](#)

[Prev](#)[Up](#)

Chapter 15. Parallel Query
Part II. The SQL Language

[Home](#)[Next](#)

[Chapter 15. Parallel Query](#)

EXAMPLE: DOCUSAURUS SHOWS VERSION

[Docusaurus](#)[Docs](#)[API](#)[Blog](#)[Showcase](#)[Community](#)

2.1.0 ▼

🌐 English ▼

[Introduction](#)[Getting Started](#) ▼[Installation](#)[Configuration](#)[Playground](#)[TypeScript Support](#)[Guides](#) >[Advanced Guides](#) ▼[Architecture](#)[Plugins](#)[Routing](#)[Static site generation](#)> [Advanced Guides](#)> [Routing](#)

Version: 2.1.0

Routing

Docusaurus' routing system follows single-page application conventions: one route, one component. In this section, we will begin by talking about routing within the three content plugins (docs, blog, and pages), and then go beyond to talk about the underlying routing system.

Routing in content plugins

Every content plugin provides a `routeBasePath` option. It defines where the plugins append their routes to. By default, the docs plugin puts its routes under `/docs`; the blog plugin, `/blog`; and the pages plugin, `/`. You can think about the route structure like this:

IMAGES

DIAGRAMS, THEORY

- Explain
- Confirms
- Visual learners

DUCK DB

● DuckDB

Documentation ▾

Blog

GitHub

Contributing

Installation

▸ Guides

▾ Documentation

Connect

▸ Data Import

▸ Client APIs

▾ SQL

Introduction

▾ Statements

Overview

Select

Insert

Delete

Update

Create Schema

Create Table

Create View

Create Sequence

Create Macro

Drop

Alter Table

Copy

Export

▸ Query Syntax

▸ Data Types

▸ Expressions

▸ Functions

Indexes

Aggregates

Window Functions

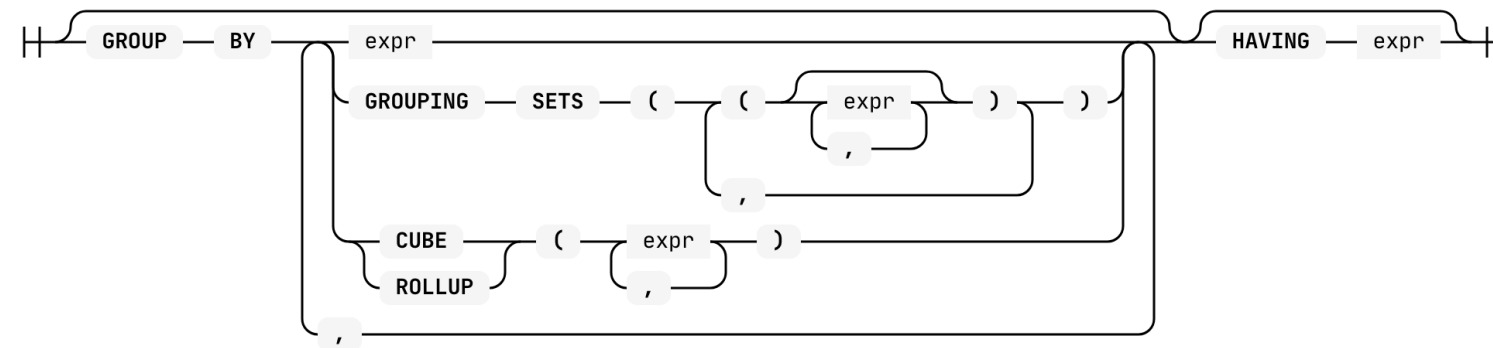
can also be used to quickly see a snapshot of the data when exploring a data set. The `sample` clause is applied right after anything in the `from` clause (i.e. after any joins, but before the `where` clause or any aggregates). See the [sample](#) page for more information.

WHERE clause

|| WHERE — expr ||

The WHERE clause specifies any filters to apply to the data. This allows you to select only a subset of the data in which you are interested. Logically the `WHERE` clause is applied immediately after the `FROM` clause.

GROUP BY/HAVING clause



The GROUP BY clause specifies which grouping columns should be used to perform any aggregations in the `SELECT` clause. If the `GROUP BY` clause is specified, the query is always an aggregate query, even if no aggregations are present in the `SELECT` clause.

DIAGRAMS IN POSTGRES DOC

7.2.3. The GROUP BY and HAVING Clauses

After passing the WHERE filter, the derived input table might be subject to grouping, using the GROUP BY clause, and elimination of group rows using the HAVING clause.

```
SELECT select_list
  FROM ...
  [WHERE ...]
  GROUP BY grouping_column_reference [, grouping_column_reference]...
```

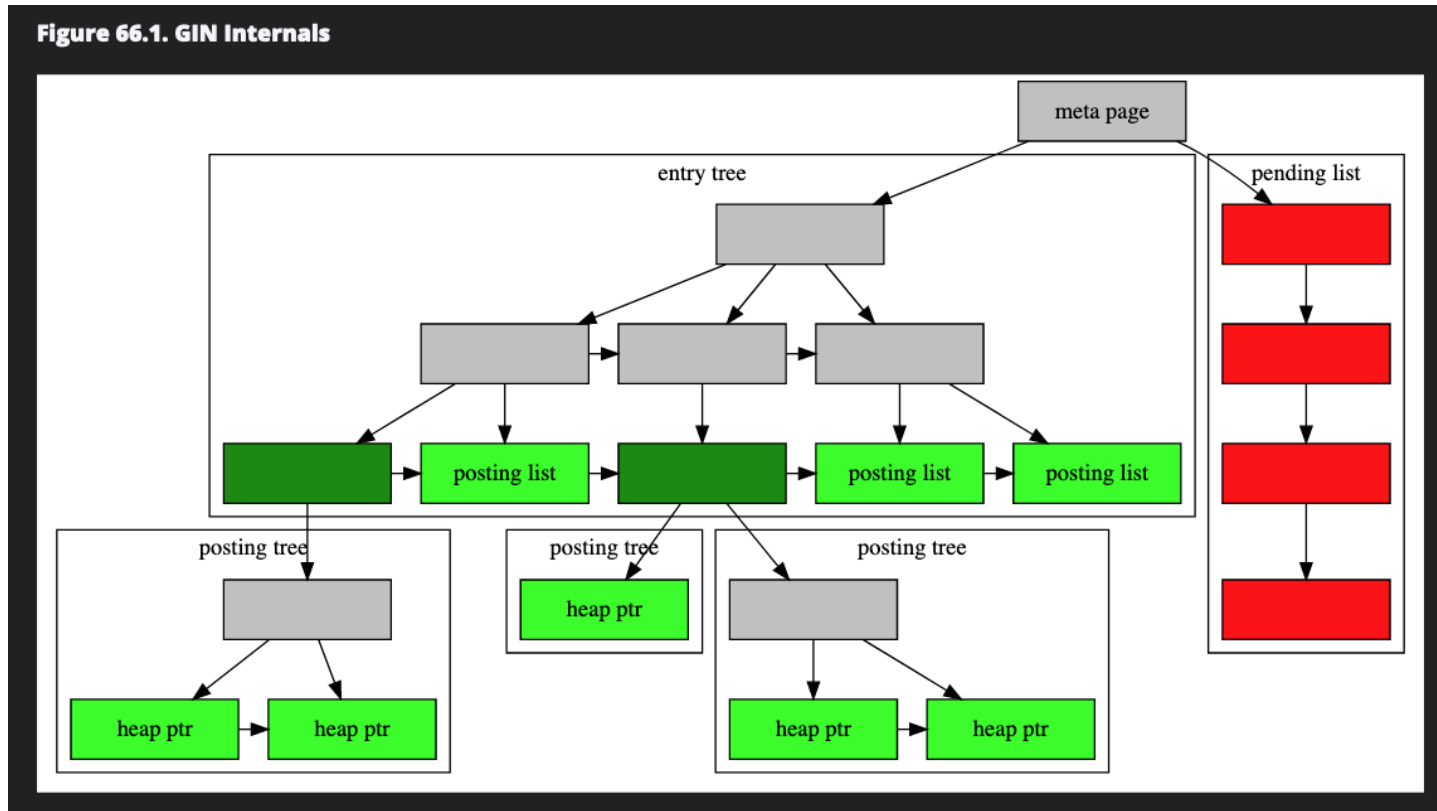
The **GROUP BY** clause is used to group together those rows in a table that have the same values in all the columns listed. The order in which the columns are listed does not matter. The effect is to combine each set of rows having common values into one group row that represents all rows in the group. This is done to eliminate redundancy in the output and/or compute aggregates that apply to these groups. For instance:

```
=> SELECT * FROM test1;
 x | y
---+---
 a | 3
 c | 2
 b | 5
 a | 1
(4 rows)

=> SELECT x FROM test1 GROUP BY x;
 x
---
 a
 b
 c
(3 rows)
```

In the second query, we could not have written `SELECT * FROM test1 GROUP BY x`, because there is no single value for the column `y` that could be associated with each group. The grouped-by columns can be referenced in the select list since they have a single value in each group.

DIAGRAMS IN POSTGRES DOCUMENTATION



- Color choices
- One of the three images

RECOMMENDED TOOL

[projects](#) / [postgresql.git](#) / blob

[summary](#) | [shortlog](#) | [log](#) | [commit](#) | [commitdiff](#) | [tree](#)
[blame](#) | [history](#) | [raw](#) | [HEAD](#)

Allow nodeSort to perform Datum sorts for byref types

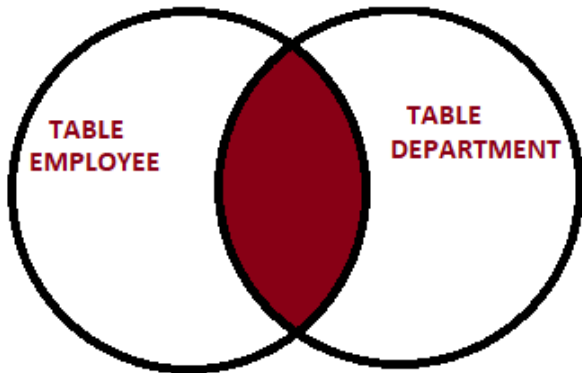
[\[postgresql.git\]](#) / [doc](#) / [src](#) / [sgml](#) / [images](#) / [README](#)

```
1 Images
2 =====
3
4 This directory contains images for use in the documentation.
5
6 Creating an image
7 -----
8
9 A variety of tools can be used to create an image. The appropriate
10 choice depends on the nature of the image. We prefer workflows that
11 involve diffable source files.
12
13 These tools are acceptable:
14
15 - Graphviz (https://graphviz.org/)
16 - Ditaa (http://ditaa.sourceforge.net/)
17
```

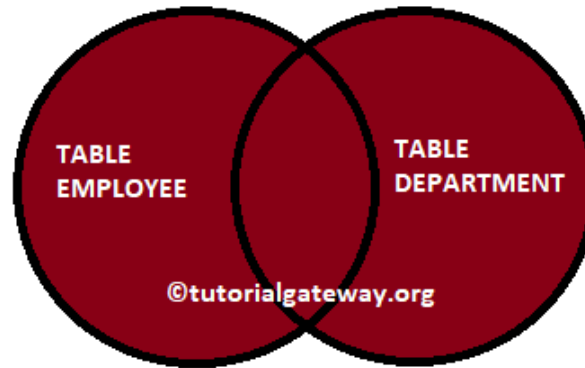
DIAGRAMS

- What can we find on the Web?

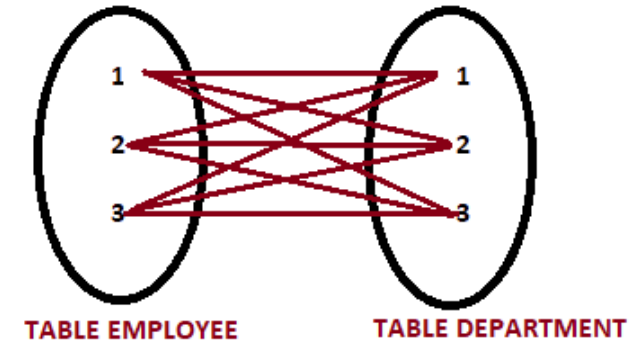
INNER JOIN EXAMPLE



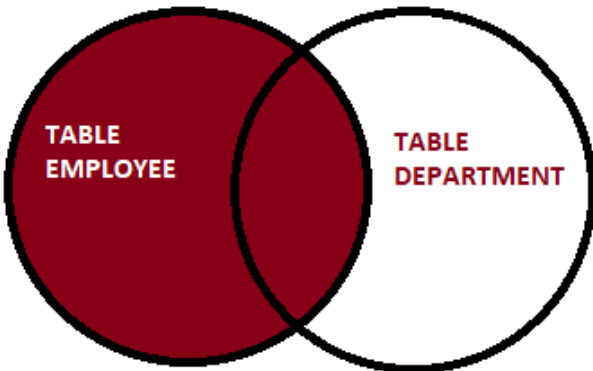
FULL JOIN EXAMPLE



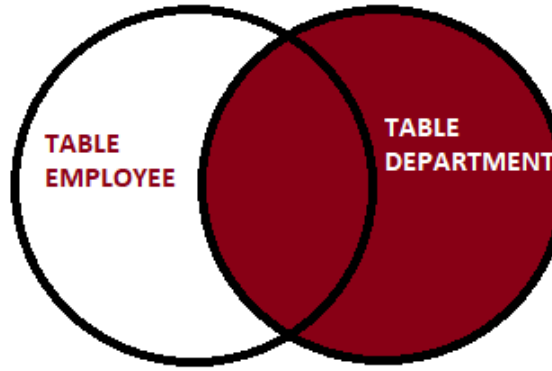
CROSS JOIN EXAMPLE



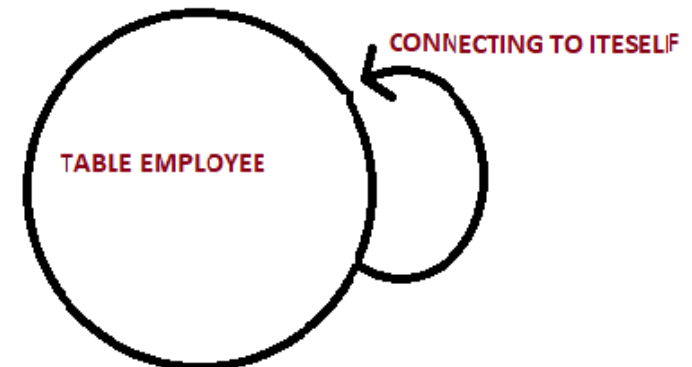
LEFT JOIN EXAMPLE



RIGHT JOIN EXAMPLE



SELF JOIN EXAMPLE



MAILING-LIST

Re: Images in the official documentation

From: Pavel Golub <pavel(at)microolap(dot)com>
To: Jürgen Purtz <juergen(at)purtz(dot)de>, pgsql-docs(at)lists(dot)postgresql(dot)org
Subject: Re: Images in the official documentation
Date: 2018-07-19 12:06:08
Message-ID: [1554502154.20180719150608@gf.microolap.com](#)
Views: [Raw Message](#) | [Whole Thread](#) | [Download mbox](#) | [Resend email](#)
Thread:

2018-07-19 12:06:08 from Pavel Golub <pavel(at)microolap(dot)com> 

Lists: [pgsql-docs](#)

Hello, Jürgen.

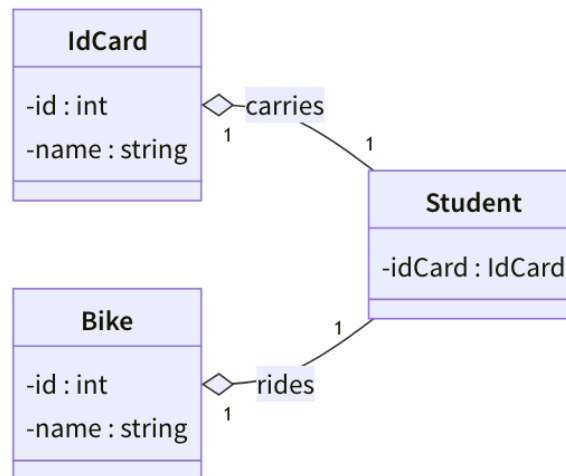
You wrote:

```
JP> Our discussion about grafics in the documentation reached to the
JP> conclusion that we shall use SVG, the importance to 'diff-ability'
JP> is rated differently, and there is no consensus about tools.
```


IMAGE AND MAINTENANCE

- It's tool's choice!
- MermaidJS
- XML-based diagrams

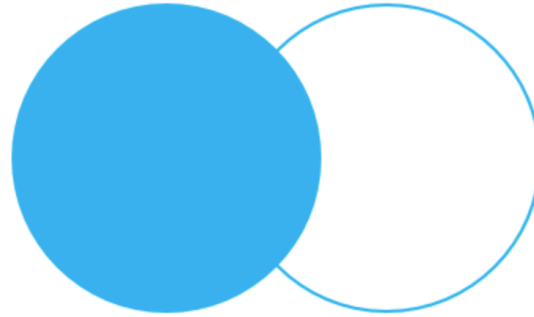
```
classDiagram
    direction RL
    class Student {
        -idCard : IdCard
    }
    class IdCard{
        -id : int
        -name : string
    }
    class Bike{
        -id : int
        -name : string
    }
    Student "1" --o "1" IdCard : carries
    Student "1" --o "1" Bike : rides
```



JOINING THE MOVEMENT?

- Why is this tutorial illustrated? <https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-joins/>
- And not the official documentation?
<https://www.postgresql.org/docs/15/queries-table-expressions.html>

TUTORIAL: WITH IMAGES



LEFT OUTER JOIN

To select rows from the left table that do not have matching rows in the right table, you use the left join with a `WHERE` clause. For example:

```
SELECT
  a,
  fruit_a,
  b,
  fruit_b
FROM
  basket_a
LEFT JOIN basket_b
  ON fruit_a = fruit_b
WHERE b IS NULL;
```

OFFICIAL DOC: TEXT ONLY

Qualified joins

```
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2 ON boolean_expression  
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2 USING ( join column list )  
T1 NATURAL { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN T2
```

The words **INNER** and **OUTER** are optional in all forms. **INNER** is the default; **LEFT**, **RIGHT**, and **FULL** imply an outer join.

The *join condition* is specified in the **ON** or **USING** clause, or implicitly by the word **NATURAL**. The join condition determines which rows from the two source tables are considered to “match”, as explained in detail below.

The possible types of qualified join are:

INNER JOIN

For each row **R1** of **T1**, the joined table has a row for each row in **T2** that satisfies the join condition with **R1**.

LEFT OUTER JOIN

First, an inner join is performed. Then, for each row in **T1** that does not satisfy the join condition with any row in **T2**, a joined row is added with null values in columns of **T2**. Thus, the joined table always has at least one row for each row in **T1**.

RIGHT OUTER JOIN

First, an inner join is performed. Then, for each row in **T2** that does not satisfy the join condition with any row in **T1**, a joined row is added with null values in columns of **T1**. This is the converse of a left join: the result table will always have a row for each row in **T2**.

FULL OUTER JOIN

First, an inner join is performed. Then, for each row in **T1** that does not satisfy the join condition with any row in **T2**, a joined row is added with null values in columns of **T2**. Also, for each row of **T2** that does not satisfy the join condition with any row in **T1**, a joined row with null values in the columns of **T1** is added.

TEXTUAL CONTENT

- Essential
- Easy to maintain
- Multi-channel & searchable

TEXTUAL CONTENT : THE GOOD IN PG DOC

- Quality
- Amount : almost everything is covered
- Detailed, thorough

WALLS OF TEXTS, PLURAL

- 1/ Trigger Overview <https://www.postgresql.org/docs/15/trigger-definition.html>
- 2/ GIN extensibility <https://www.postgresql.org/docs/13/gin-extensibility.html>
- 3/ TOAST Database Physical Storage
<https://www.postgresql.org/docs/13/storage-toast.html#STORAGE-TOAST-ONDISK>

BREAKIN' THE WALL

- Why is this page more clear than the official doc ?

https://www.tutorialspoint.com/postgresql/postgresql_triggers.htm

TEXTUAL CONTENT : IMPROVE

- Add images
- Bullet points are ok
- Use emphasis, bold

BE BOLD!

- Improvement/ Be **bold**
- Sometimes, being bold is **helpful for the reader**
- **Scanning** the content
- Looking for **information**

BEGINNERS

TODAY IN DOCS: BEGINNERS

- Getting Started
- Overview
- Synthetic views
- Examples

DIFFERENT MEDIA OR CHANNELS

- Tutorials
- Videos

AIRTABLE

 [Product](#) > [Solutions](#) > [Pricing](#) [Enterprise](#) > [Resources](#) >

[Home](#) > [Guides](#) > [Build your workflow](#) > [Add data with records](#)

WRITTEN BY



FILED UNDER

[Build your workflow](#)

TOPICS

- [Add data with records](#)
- [What is a record?](#)
- [Same record, different perspective](#)
- [Take action: Add records to your table](#)

SHARE



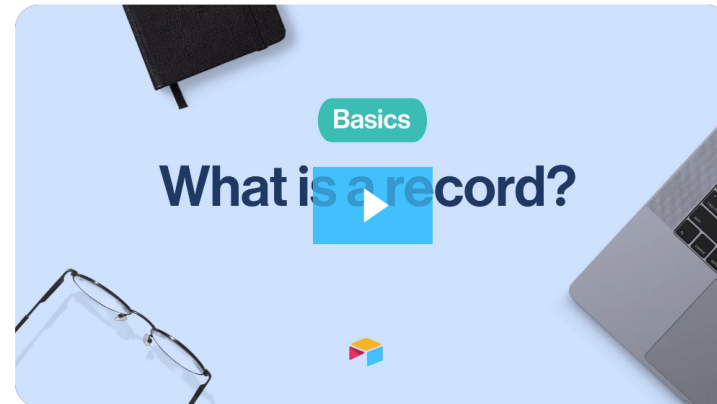
[← Guides](#)

[← Build your workflow](#)

Add data with records

With your base's structure set up, you're ready to start filling in all the items you need to track for your workflow.

Records are the core unit of your Airtable base. Each one might look like a simple item in a list, but they're actually the most important building blocks for a dynamic workflow.



What is a record?

A record is an individual item in a table, along with all of its relevant details. You can think of it as the individual unit of the table—if your table is organizing events at a conference, each record would be a presentation, or if you're producing a television series, each record would be an individual episode.

FIRST STEPS

- Guidance wanted

LEARNING SQL: GOOD



CockroachDB @CockroachDB · Mar 2, 2021

...

🤔 What is your favorite resource for learning SQL?



8



2



7



Hemant Jangid

...

@hemmmer

Replying to @CockroachDB

Postgres documentation 🥰

2:55 AM · Mar 3, 2021 · Twitter for iPhone

CHOOSE YOUR OWN ADVENTURE TALK



Postgres documentation

- How to begin
- Choose your own adventure!
 - Postgres DBA
 - Software developer
 - Database user

SUGGESTION

- Personas/paths by profile

CONTRIBUTING

CONTRIBUTING TODAY

- Technical pre-requisite
- Today : git, via github/gitlab

THE GOOD: CONTRIBUTE TO POSTGRES

[projects](#) /



Welcome to PostgreSQL's [git](#) repository! It serves mirrors of selected projects as Git pull and web access, and it offers project hosting with push access as well as user branch hosting. You can find more user branches at the [postgres organization account](#) on Github.

This repository also contains the [public repository](#) for PostgreSQL itself. It is updated within minutes of each commit from the master repository.

You can find out the URL for each project at the project's summary page.

You can find some help on how to use this site [here](#). You can create and manage your repositories [here](#).

Please contact the [web team](#) to report any problems with this site.

☐ re

[List all projects](#)

Project	Description	Owner	Last Change	
2ndquadrant_bdr.git	Obsolete repo for Bi-direction...	Abhijit Menon-Sen	6 years ago	summary shortlog log tree
advocacy.git	This repository contains pictu...	Jonathan Katz	10 years ago	summary shortlog log tree
automirror.git	The postgresql.org automirror...	Dave Page	11 years ago	summary shortlog log tree
bucardo.git	Bucardo		6 years ago	summary shortlog log tree
cfbot.git	cfbot repo	Stephen Frost	No commits	summary shortlog log tree
check_postgres.git	check_postgres		6 years ago	summary shortlog log tree
docbot.git	pg_docbot source code, website...	Andreas 'ads' Scherbaum	4 years ago	summary shortlog log tree
fosdem2021-content.git	Public content for the Postgre...	Andreas 'ads' Scherbaum	14 months ago	summary shortlog log tree
fosdem2021-static.git	Static content for the Postgre...	Andreas 'ads' Scherbaum	2 years ago	summary shortlog log tree

CONTRIBUTING THE PROCESS

- Today smoothing contribution
- We want to ease it
- Many channels and events
 - HacktoberFest

CONTRIBUTOR GUIDE

- Giving best practices, guidance
- PHP Contributor Guide (<http://doc.php.net/tutorial/>)

CONTRIBUTE TO POSTGRES DOCUMENTATION




- The Wiki
- CommitFest

CONTRIBUTION PROCESS

- Who will review my contribution?
- Who's in charge?
- What is the update cycle for the documentation?

CONTRIBUTE TO POSTGRES DOC

Documentation

- ☐ Provide a manpage for postgresql.conf
 - [A smaller default postgresql.conf](#) 
 - [A smaller default postgresql.conf](#) 
- ☐ Document support for N' ' national character string literals, if it matches the SQL standard
 - <http://archives.postgresql.org/message-id/1275895438.1849.1.camel@fsopti579.F-Secure.com> 

CHANGING THE DOCUMENTATION

- A form in footer

Submit correction

If you see anything in the documentation that is not correct, does not match your experience with the particular feature or requires further clarification, please use [this form](#) to report a documentation issue.

DOCUMENTATION REVIEW

- Governance is not clear
 - from an external point of view

IN CONCLUSION

A GOOD DOCUMENTATION

- Gain of time
- Really helpful
- Communicates your choices
- Teach your best practices

OLD AND IT SHOWS

- Old fashioned



CONTENT

- Written by experts
- For experts

WELCOMING

- Community is welcoming
- Beginners...could still struggle

IMPORTANT QUESTION



MY OPINION

- 25+ years old and it shows
- Improvements
 - Orientation page by persona
 - Renaming sections (BC break)
 - UX: more diagrams
 - Welcoming doc contributors

MERCI BEAUCOUP !

- Thanks !
- Thanks to Lætitia Avrot and the Postgres Conf team
 - Images, except screenshots, are from Wikimedia Commons

QUESTIONS?

- Contact me
 - Twitter: @sarahhaim or @mereteresa
 - Mastodon: @mereteresa@tetaneutral.net



BEDROCK

Creating Streaming Champions