In Aid Of R.T.F.M.

Corey Huinker

Corlogic

PgConf EU 2019

PostgreSQL has docs!

- and they're pretty good...[1]
- so you need to know what you're looking for
- or know the right words to describe your problem

---

[1] ...as a reference

Many People learn by example

- Visual, Auditory, Kinisthetic learning methods
- A reference only provides Visual
- Many people[1] learn from seeing one example, and only then can they grasp the abstractions.

[1] me

Example Code in PostgreSQL Docs

PostgreSQL has almost no examples

- Syntax Definitions don't count
- Existing documentation has pretty good function examples[1]

---

[1] https://www.postgresql.org/docs/current/functions-string.html

Function Usage Examples

- many more are needed
- especially edge cases
- but not necessarily on the same page as the definition

Command Examples in PostgreSQL Docs

- Show more common usage patterns
- Show context of the problem solved by the command
- Examples with discussions of the pros and cons of the technique demonstrated

The PostgreSQL Wiki

- Very little thought to organization
- The junk drawer of wisdom
- Many results are no longer relevant
- No vetting organization to my knowledge
- The postgresql.org core docs don't reference the wiki

So where is a knowlege seeker to turn?

If we don't teach them, they'll learn it on the street

Stack Exchange

- The blind leading the blind
- Popular answers ranked over good answers
- Examples are often very stale

Ok, so where else then?

Reddit

- Somewhat better quality of answer
- Too many homework questions being asked, which tires responders
- Question volume is pretty low, mostly reposts of articles and questions from...Stack Exchange

OK, seriously, where else is a knowlege seeker to turn?

Mailing Lists

- Not many people know they exist
- Must join to post
- Searching the list archives is not obvious
- Low volume

IRC / Slack

- Responsivenes varies by time of day
- Either you get an answer immediately or not at all
- A bit easier to shame people for asking us to do their homework
- A lot easier to mistake genuine questions for homework
- Immediate responses are not necessarily carefully considered
- It all goes in the bit bucket

Blogs

Blogs can be broken two categories:

1. Blogs written by people whose livelihood in centered around supporting or developing PostgreSQL
2. Anybody else

Blogs By Experts

- Extremely detailed descriptions of hyperspecific topics
- As developers grow more specialized this will only increase
- Of limited use to newer users

Blogs By The Unwashed Masses

- Written with the theory that "Quantity has a quality all its own."
- Often dole out information in the smallest of doses to increase the number of blog posts[1]
- Prioritizes exposure and soft-sell advertising over education
- Case in point, the common misattribution of the Quantity quote

---

[1] ::cough:: Medium ::cough::

Blogs By Bots

- Link farms that copypasta of other blogs
- Just throwing random SEO text onto a page with ads
- Makes one reconsider humanity

Youtube

- Ok...for step by step instructions
- Not well suited for cookbook style instructions as consumers can't copy/paste
- All the failings of blogs, but with interstital ads
- Much harder to skip over the boring parts

Google

- The primary means of, and barrier to, discoverability
- Has a dumb fondness for old versions of document pages
- Because that's where the clicks accumulated
- No matter what we do, Google could change their algorithm tomorrow
- So we can't optimize <u>for</u> google.

Google

- The /**current**/ links in the documentation are fairly new
- and will eventually accumulate a plurality of clicks
- At which point they can never be displaced by an older version.
- So we got that going for us[1]

```
[1] Spackler, Carl (1980)
```

We Are Not Alone

Language: Python

- It's nearly impossible to distinguish which version of python is used in an example
- Stack Exchange filled with examples from 2.X that crowd out 3.X examples
- Python 3.0 was released Dec 8, 2008
- Python 2.7 is EOL Jan 1, 2020
- Python 2.X (where X < 7) is already EOL

Language: Node.js[1]

Package management is a house of cards in a room full of hair dryers[2]

---

[1] Don't get him started.

[2] Oops. Oh, well.

Languages like C and Go

- Finding code examples is hard because the name defeats searchability
- And the `-lang` suffix isn't used consistently
- pg Backrest and pg Barman ~~have~~ had a similar problem here

Operating System: Linux

- Package naming conventions not consistent across distros (Ubuntu/Redhat/Arch)
- Or even successive versions within a distro
- Evolving classification methodologies
- Some package maintainers ignore distribution methodologies
- Version numbering borne of marketing

Operating System: Android and iOS

- Permanent Beta Development
- Menuing systems that are flavor of the month.
- …So a HOWTO video made today
- …Is irrelevant in six months
- …But clogs up search results for years
- …Until the video that corrects it is irrelevant, too

We Have Some Advantages

- Our language(s) are fundamentally text, so screenshots are rarely necessary
- A strong commitment to backwards compatibility, so examples rarely break
- SQL standards ensure that users coming from other databases have a foundational understanding
- SQL standards ensure that when users do encounter differences with other databases, we have the moral high ground

Advantage: Purely Numerical Versions

Annual releases make for some intuitive age estimation

```
postgres=# SELECT date_part('year', current_date)
                 - 2007 AS new_pg_version;
 new_pg_version
----------------
             12
(1 row)
```

Advantage: Perils of Non-Numerical Versions

- Do I search for Disco or Dingo, Warty or Warthog?
- Android candy names always draw in non-technical results[1]
- OSX version names confuse major/minor updates, lack ordinality, bring in search results for zoo animals and vacation destinations

---

[1] They gave up with Android 10

What Can Be Done
(in the very short term)

Glossary of Terms for PostgreSQL We Need One

Googling this led to "Terminology and Notation"[1] and "Terminology" [2], both of which percolate up to "Conventions"[3]

---

[1] https://www.postgresql.org/docs/7.3/notation.html

[2] https://www.postgresql.org/docs/6.4/intro232.htm

[3] https://www.postgresql.org/docs/current/notation.html

Glossary of Terms for PostgreSQL Coming Soon

- Useful for helping users who are struggling to describe a problem find the correct search terms
- especially when translated into all the languages that the docs are currently translated
- it would itself be web-searchable and aid discovery
- Patch submitted October 13, 2019[4]

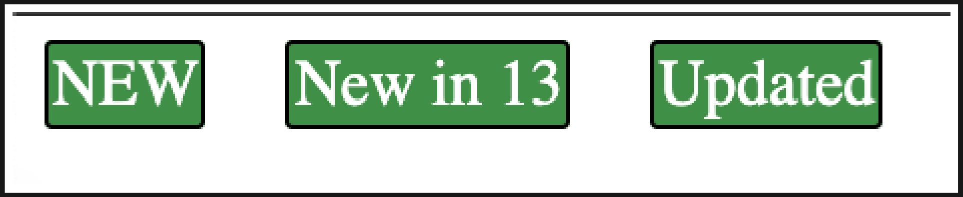
---

[1] https://commitfest.postgresql.org/25/2305/

Inter-version notations

- Inspired by document change red-lining
- Focuses the user on what is new, what did change, and what didn't change
- Was especially useful when reviewing rules changes for roller derby
- Direct HTML comparison tools exist (htmldiff)
- But don't fit our tool chain
- Others have this same issue

New/Updated Badges



- For features that are new in this release
- For features that have changed since the last release
- For text that has changed explaining a feature that has not
- Easy to clear out all "badges" when we start version N+1, and badge all doc commits after
- Patch is in the works

Cite-ability

- Citations by web page are too granular
- Need anchors within pages
- one anchor per function
- one per use-case example
- Anchors never die from one version to the next
- "retired" anchors to to the bottom of the page

What Can Be Done:
Within PostgreSQL Itself

psql Commands are pretty cryptic

```
\dgS+
```

DESCRIBE

```
DESCRIBE TABLE foo;
DESCRIBE FUNCTION bar(int, text);
```

- We should make `DESCRIBE` a server-level command
- Returns \d-something results?
- Or a composable result set?
- Possibly with a JSON output mode that is queryable
- Requires moving much of the \d-something code from pure client-side to client/server common tree

SHOW CREATE TABLE my_table

```
SHOW CREATE TABLE foo;
SHOW CREATE FUNCTION bar(int, text);
```

- For a given object, show the commands required to create that object as it currently exists in the db
- Depedent objects like indexes would be included
- Referenced objects like foreign key referenced tables would not
- requires moving much of the `pg_dump` code from pure client-side to client/server common tree

SHOW HELP command

```
SHOW HELP CREATE TABLE;
SHOW HELP ALTER FUNCTION;
```

- Ability for the server to fetch a subset of the docs in locale-specific language
- If not that, at least provide the canonical URL for documentation of that command
- Possibly implement this with a foreign data wrapper or extension

What Can Be Done
(stretch goals)

We need an example database

- MSSQL and Access have Northwind[1]
- Users to slowly accumulate familiarity with a complex dataset
- Example queries to rely on assumptions about table design, data volume, etc
- Avoid rebuilding the sample tables from scratch and complex `generate_series()` calls[2]

---

[1] I guess we do too: https://github.com/pthom/northwind_psql

[2] Oracle blogs Oracle Scratchpad, AskTom, Spectator Sport, really

We need a FREE example database

- Existing datasets often have legal encumbrances
- Ownership aggressively enforced[1]
- Mapping data often has spoiler data to prove copying[2]

---

[1] Muse, IMDB, etc

[2] see https://en.wikipedia.org/wiki/Trap_street

We need an INTERESTING example database

- Dry Subject matter (ex: USDA nutrition data)
- Flat data with very few relations (ex: Census data)
- Increasingly irrelevant subject matter (DVD Rental store)

Example Database Coverage

Database should be designed to test most major elements of postgresql:

Example Database Coverage

Database should be designed to test most major elements of postgresql:

- foreign keys,

Example Database Coverage

Database should be designed to test most major elements of postgresql:

- foreign keys, partitioning,

Example Database Coverage

Database should be designed to test most major elements of postgresql:

- foreign keys, partitioning, views,

Example Database Coverage

Database should be designed to test most major elements of postgresql:

foreign keys, partitioning, views, materialized views,

Example Database Coverage

Database should be designed to test most major elements of postgresql:

- foreign keys, partitioning, views, materialized views, all index types,

Example Database Coverage

Database should be designed to test most major elements of postgresql:

foreign keys, partitioning, views, materialized views, all index types, triggers (per row, per statement, event),

Example Database Coverage

Database should be designed to test most major elements of postgresql:

foreign keys, partitioning, views, materialized views, all index types, triggers (per row, per statement, event), functions in all languages shipped with core,

Example Database Coverage

Database should be designed to test most major elements of postgresql:

foreign keys, partitioning, views, materialized views, all index types, triggers (per row, per statement, event), functions in all languages shipped with core, at least one stored procedure,

Example Database Coverage

Database should be designed to test most major elements of postgresql:

foreign keys, partitioning, views, materialized views, all index types, triggers (per row, per statement, event), functions in all languages shipped with core, at least one stored procedure, generated columns,

Example Database Coverage

Database should be designed to test most major elements of postgresql:

foreign keys, partitioning, views, materialized views, all index types, triggers (per row, per statement, event), functions in all languages shipped with core, at least one stored procedure, generated columns, etc.

Database Features

- Database should be sized to test most major elements of postgresql
- but be loadable in < 5 mins on current hardware
- This database should itself be versioned, so that each iteration is a showcase of the corresponding postgresql version

Example code can "name check" the version of the database by starting the example with

```
SELECT * FROM database_version
```

We need a REPL

- A hosted example database
- Or an easy packaging of the example database
- Allow for easy reset back to baseline
- So users can learn by breaking things

REPL Thoughts

- Would allow users to "play along at home" with tutorials
- Postgres.app is a good foundation for this, but is OSX specific
- The example database could be `template2`, basically.

Collection of Recipes

- Beyond what the wiki already does
- Clear what versions where the recipe works
- Extensive citations to the most specific anchor
- Commentary on the reasoning behind the solution
- Sportscaster level of detail

Collection of Recipes

- Write the recipes against the example database
- Allows us to set up regression tests
- Or review once a year for correctness
- Google Summer Of Testing?

Refugee Welcome Guide

- Some mention of differences in terminology and concepts for a user coming from other common databases
- These can be very version specific, as those databases will themselves evolve with time, as will the list of important databases
- A chance to sign our own praises
- Maybe belongs in the wiki, maybe in the core docs

Archaeologists

- Sounds better than Vigilantes
- Collect submissions, look for examples on common websites
- Find good examples, incorporate them into the wiki
- Find bad examples, try to correct them in place
- Google Summer of Docs?

What can YOU do?

Setting a good example with code

Setting a good example with code

```
postgres=# SELECT CURRENT_DATE, version();
 current_date | version
--------------+----------------------------------------
 2019-10-15   | PostgreSQL 11.4 on x86_64-apple-darwin16.7.0,
(1 row)
```

Setting a good example with citations

- When citing the docs, cite to the most granular link possible
- Cite current version or `/current/`?
- Depends on whether we orphan anchors or not

Pitfalls When Documenting

- Be careful of graphical documentation
- The future will not be less concerned about accessibility than today
- There may be legal obligations for accessibility in the future

Conclusions

- A lot of information is out there
- Some of it is wrong
- Some of it used to be right
- We can't control all of it
- We <u>can</u> attempt to counter-balance it

Conclusions

- We could take stewardship of a lot more of postgresql lore than we do
- Doing so would enhance the reputation of well curated documentation
- And we'd all have to answer fewer dumb questions
- Because we could tell them to RTFM
- without that being an insult

Thank You