PostgreSQL Observed—and Explained

Stacey Haysler Karen Jex

PgConf.EU October 24, 2024

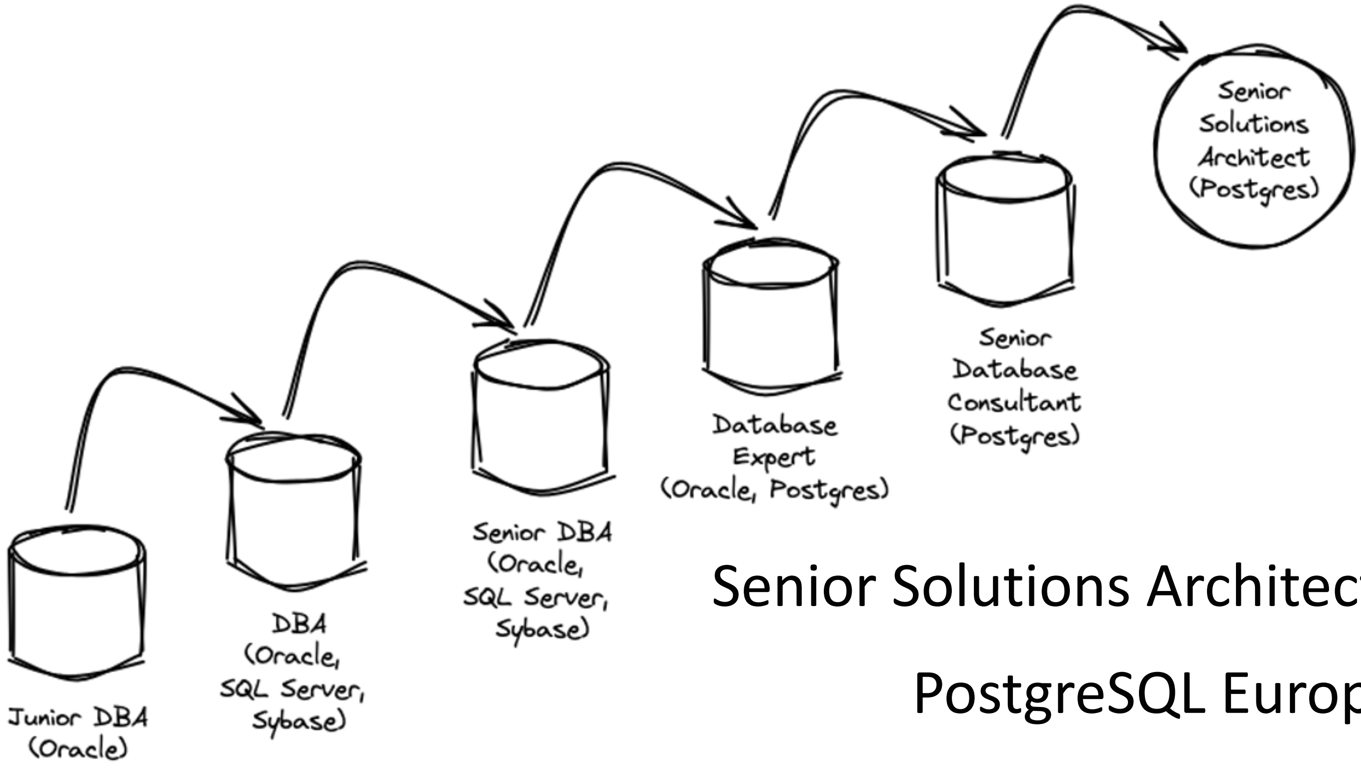
Stacey Haysler

CFO and COO at PGX, Inc.

President, U.S. PostgreSQL Association

Organizer, San Francisco Bay Area PUG

Most definitely not an engineer

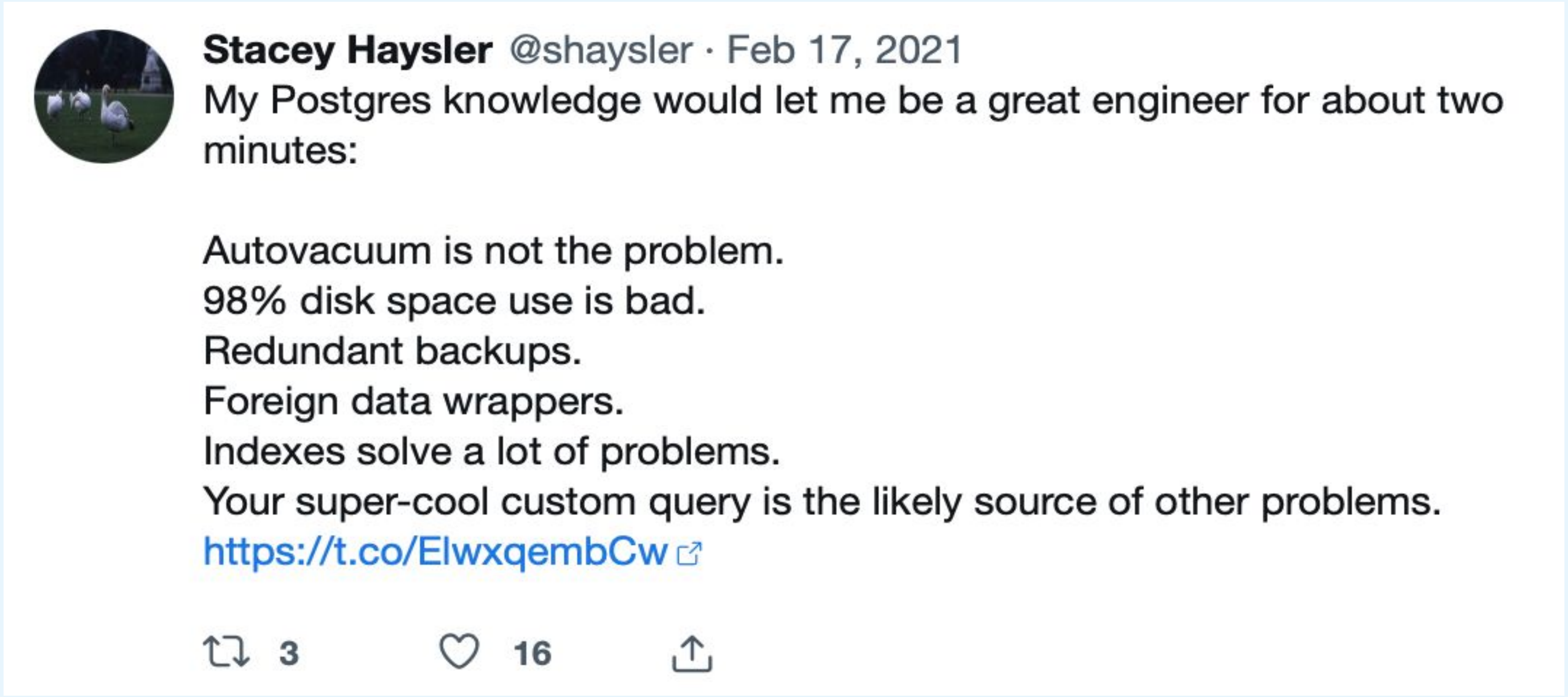

Junior DBA (Oracle)
DBA (Oracle, SQL Server, Sybase)
Senior DBA (Oracle, SQL Server, Sybase)
Database Expert (Oracle, Postgres)
Senior Database Consultant (Postgres)
Senior Solutions Architect (Postgres)

Senior Solutions Architect @ Crunchy Data

PostgreSQL Europe Board Member

PostgreSQL Europe Diversity Committee Chair

Why this talk



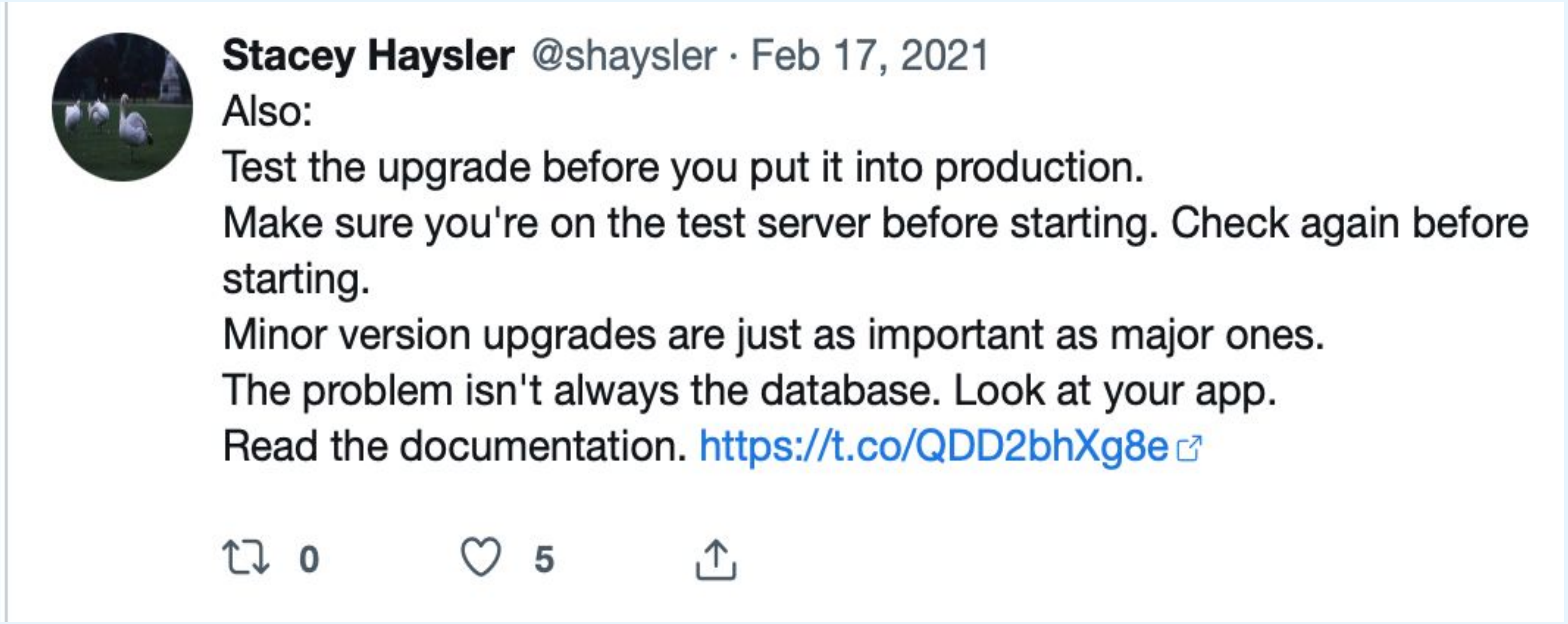
Stacey Haysler @shaysler · Feb 17, 2021

My Postgres knowledge would let me be a great engineer for about two minutes:

Autovacuum is not the problem.
98% disk space use is bad.
Redundant backups.
Foreign data wrappers.
Indexes solve a lot of problems.
Your super-cool custom query is the likely source of other problems.
https://t.co/ElwxqembCw

3 16

Why this talk



Stacey Haysler @shaysler · Feb 17, 2021

Also:

Test the upgrade before you put it into production.

Make sure you're on the test server before starting. Check again before starting.

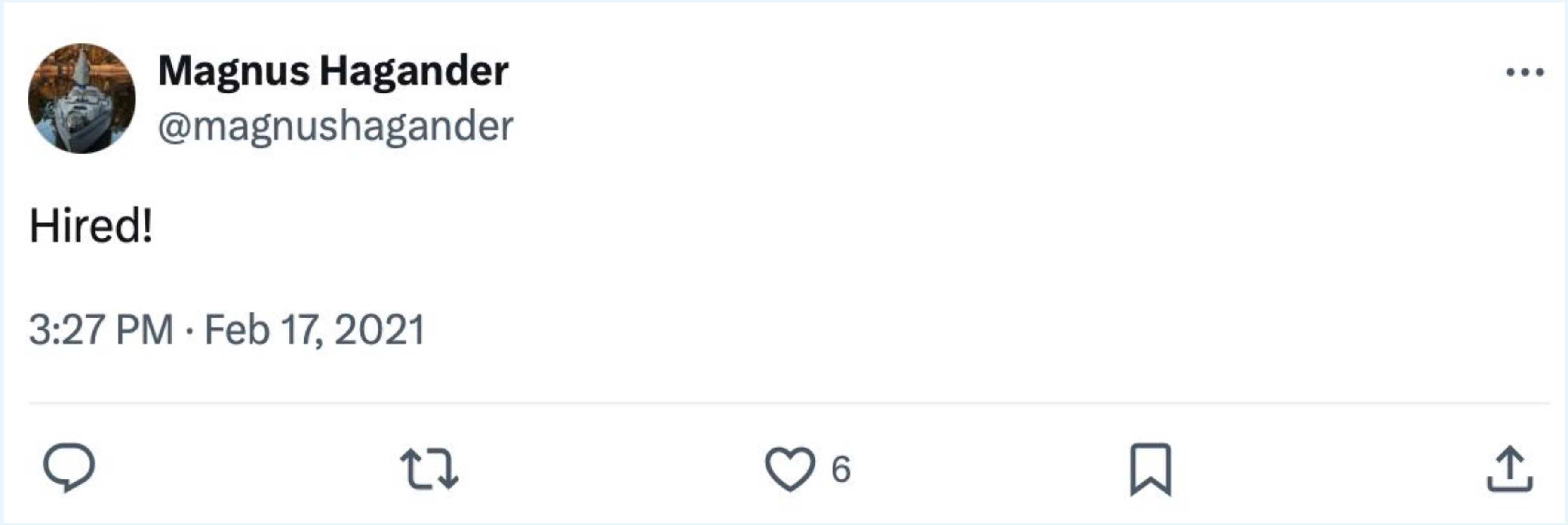
Minor version upgrades are just as important as major ones.

The problem isn't always the database. Look at your app.

Read the documentation. https://t.co/QDD2bhXg8e

0 5

Why this talk


Magnus Hagander
@magnushagander

Hired!

3:27 PM · Feb 17, 2021

6

Why this talk

Why this talk

The Assistant

Autovacuum

Autovacuum

https://www.postgresql.org/docs/current/runtime-config-autovacuum.html#AUTOVACUUM

“Optional but highly recommended”

Autovacuum

https://www.postgresql.org/docs/current/runtime-config-autovacuum.html#AUTOVACUUM

“Aha! That’s what’s slowing things down. I’ll just switch autovacuum off and everything will speed up.”

Anonymous PostgreSQL User

(Who probably later regretted their life choices)

Autovacuum

https://www.postgresql.org/docs/current/sql-vacuum.html

https://www.postgresql.org/docs/current/routine-vacuuming.html#VACUUM-FOR-WRAPAROUND

- Launches VACUUM / ANALYZE as needed
- SET log_autovacuum_min_duration=0
- Will run to avoid transaction ID wraparound even if disabled

Tuning Autovacuum

https://www.postgresql.org/docs/current/runtime-config-autovacuum.html

- autovacuum_max_workers
- autovacuum_naptime
- autovacuum_vacuum_threshold
- autovacuum_vacuum_insert_threshold
- autovacuum_analyze_threshold
- autovacuum_vacuum_scale_factor
- autovacuum_vacuum_insert_scale_factor
- autovacuum_analyze_scale_factor
- autovacuum_freeze_max_age
- autovacuum_multixact_freeze_max_age
- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit

98% disk space use is bad.

98% Disk Space Use is Bad!

Give your Database Breathing Space!

98% Disk Space Use is Bad!

Data Directory Full

```
ERROR:  could not extend file
"base/20429/2187": No space left on device
HINT:  Check free disk space.
```

98% Disk Space Use is Bad!
WAL Directory Full

Image par Pete Linforth de Pixabay

98% Disk Space Use is Bad!
WAL Directory Full

“I’ll just delete some WAL files

to make some space.”

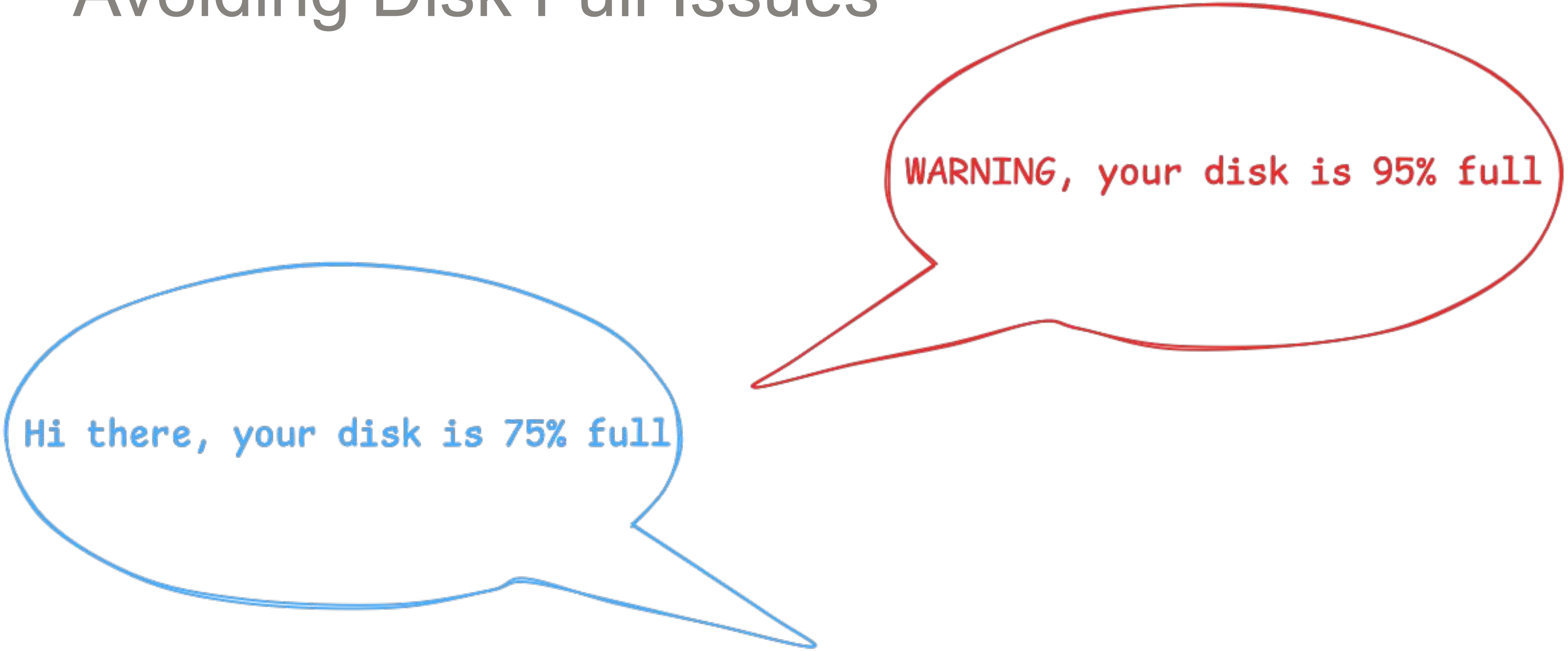
Anonymous PostgreSQL User

98% Disk Space Use is Bad!

Fixing Disk Full Issues

98% Disk Space Use is Bad!

Avoiding Disk Full Issues


WARNING, your disk is 95% full
Hi there, your disk is 75% full

Back-ups

$$1 = 0$$

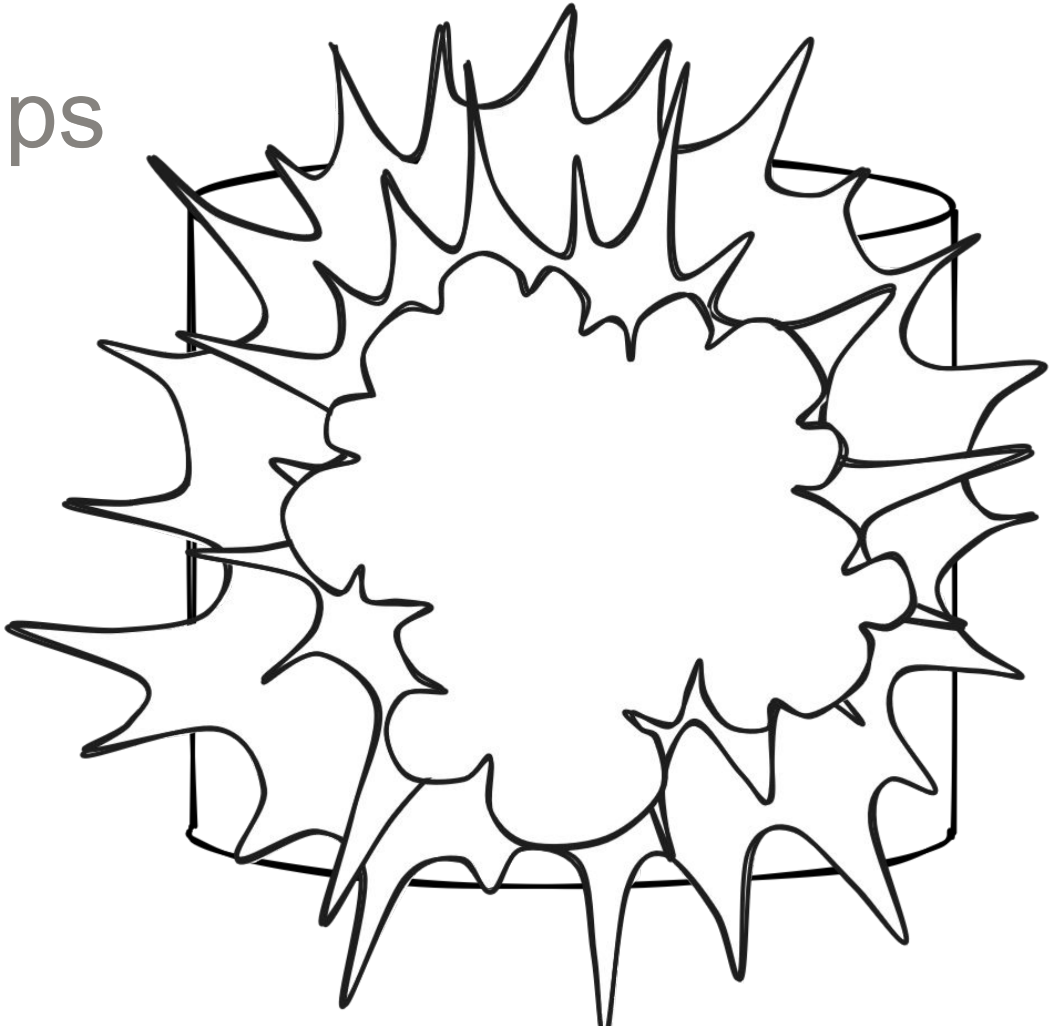
Backups

“With a single backup,

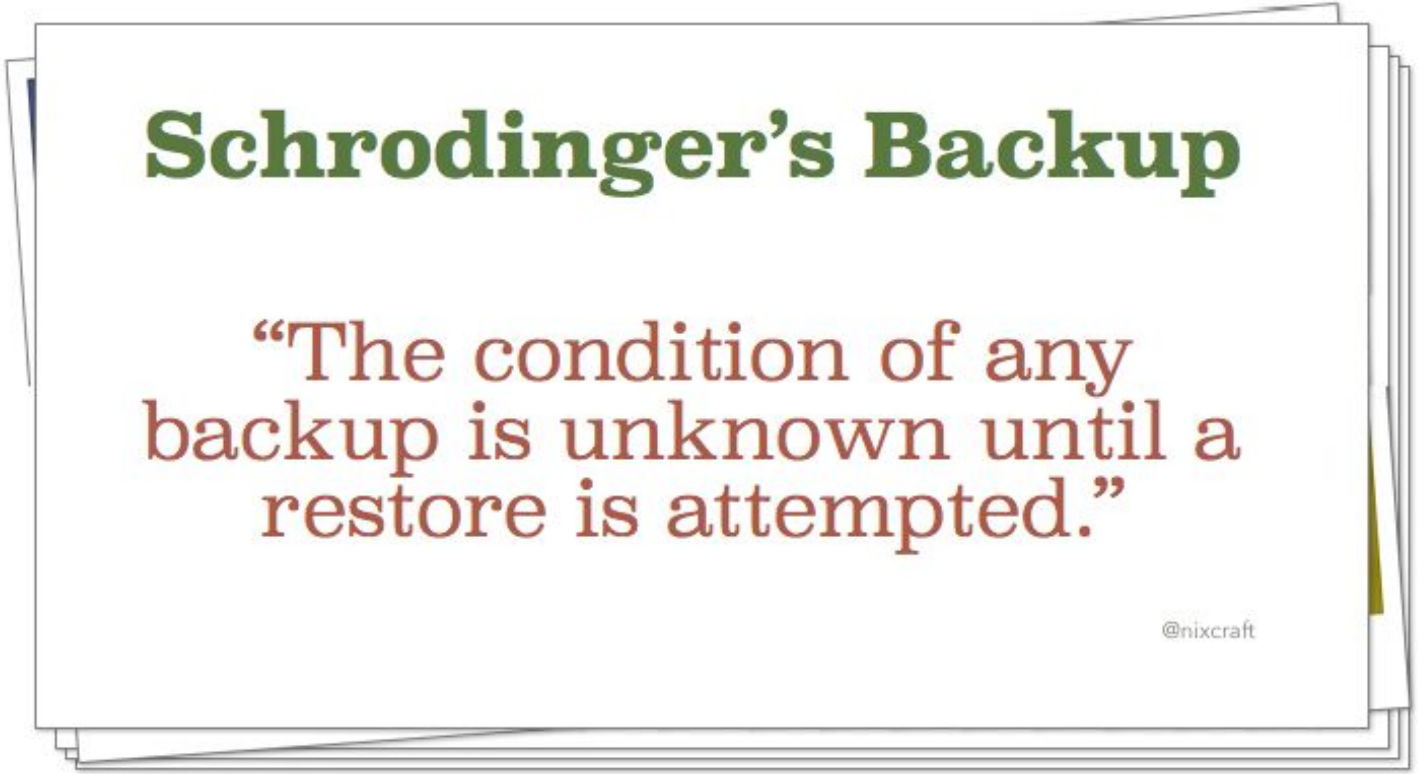
you’re already ahead of the game”

Karen Jex

Backups



Backups



Schrodinger's Backup

"The condition of any backup is unknown until a restore is attempted."

@nixcraft

Foreign Data Wrappers

Foreign Data Wrappers
“Foreign Data”

https://www.postgresql.org/docs/current/ddl-foreign-data.html

“PostgreSQL implements portions of the SQL/MED specification, allowing you to access data that resides outside PostgreSQL using regular SQL queries. Such data is referred to as foreign data.

Foreign data is accessed with help from a foreign data wrapper. A foreign data wrapper is a library that can communicate with an external data source, hiding the details of connecting to the data source and obtaining data from it.”

Foreign Data Wrappers

“Foreign Data”

Generic SQL Database
JDBC2
JDBC
VirtDB
APIs
ODBC
GDAL/OGR
SQL_Alchemy

Specific SQL Database
Sybase
PostgreSQL
DB2
Oracle
SQLite
Informix
MySQL
MonetDB
Firebird
SQL Server

NoSQL Database
BigTable
Neo4j
GridDB
Kyoto Tycoon
Redis
RDF Triplestores
RocksDB
CouchDB
Quasar
Riak
HBase
SPARQL
Cassandra
Kafka
RethinkDB
MongoDB
InfluxDB
WhiteDB
ClickHouse

Big Data
Google BigQuery
Hive
Impala
ORC
HDFS
Hadoop
Elasticsearch
Apache Arrow
file_fdw-gds

https://wiki.postgresql.org/wiki/Foreign_data_wrappers

Foreign Data Wrappers

“Foreign Data”

Logflare
Fixer.io
Dun & Badstreet
OAI-PMH
Telegram
Google
Heroku dataclips
Firebase
Facebook
ICAL
IMAP
Keycloak
Stripe
S3
pgsql-http
Web Wrappers
Stripe
WWW
RSS
Database.com
ICAL
IMAP
Database.com
Airtable
Mailchimp
Fixer.io
Google
Facebook
DynamoDB
Firebase
Parse
Git
Airtable
Git
ParquetS3
Cloudsmith.io
pgsql-http
Keycloak
S3CSV
Heroku dataclips
Telegram
WWW
S3
RSS
ICAL
Cloudsmith.io
Treasure Data
Logflare
Google Spreadsheets
Twitter
Mailchimp
Google Spreadsheets
OAI-PMH
S3CSV
pgsql-http
DynamoDB
Parse
Twitter
Open Weather Map
Treasure Data
ParquetS3
Open Weather Map
WWW
Dun & Badstreet

Geocode
Open Street Map
PBF
GeoJSON
GeoJSON
OGR
OGR
Open Street Map
GDAL
OGR
GDAL
GeoJSON
Open Street Map
PBF
GDAL
Geocode
GeoJSON
NewWord
NewWord
NewWord
Open Street Map
GeoJSON
GDAL
OGR
Geocode
Geo Wrappers
Geocode
NewWord
Open Street Map
GDAL
Geocode
PBF
OGR
NewWord
PBF
PBF

Document Collection
XLSX
pg_dump
XML
Multi CDR
gzipped
Compressed File
JSON
XLSX
Parquet
CSV
TAR
Fixed-length
JSON
JSON
ZIP
Fixed-length
TAR
Parquet
Parquet
Multi CDR
Document Collection
gzipped
Compressed File
Text Array
ZIP
Multi-File
XLSX
Text Array
Compressed File
File Wrappers
Parquet
Fixed-length
CSV
Document Collection
Text Array
Fixed-length
Multi CDR
CSV
XML
pg_dump
ZIP
JSON
XML
gzipped
Document Collection
Multi-File
XLSX
pg_dump
ZIP
CSV
Multi CDR
TAR
gzipped
Multi-File
XML
TAR
Multi-File

https://wiki.postgresql.org/wiki/Foreign_data_wrappers

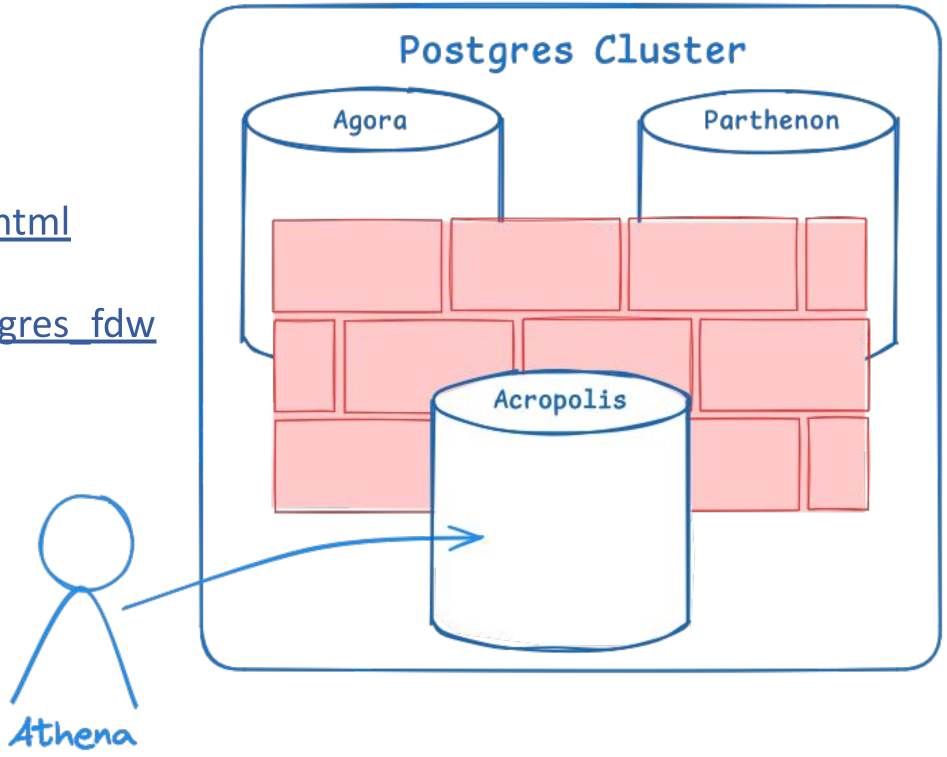
Foreign Data Wrappers

Even PostgreSQL is "Foreign"

postgres_fdw

www.postgresql.org/docs/current/postgres-fdw.html

www.crunchydata.com/blog/understanding-postgres_fdw



Foreign Data Wrappers

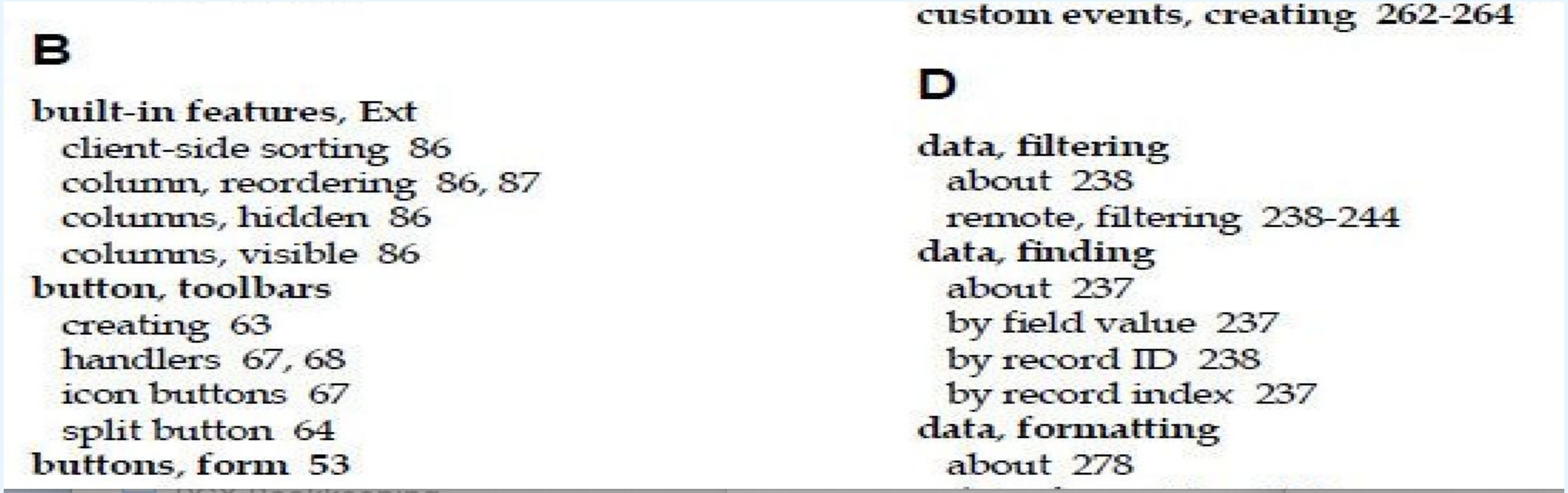
Write your Own FDW

“The foreign data wrappers included in the standard distribution are good references when trying to write your own. Look into the contrib subdirectory of the source tree.”

https://www.postgresql.org/docs/current/fdwhandler.html

Indexes and Primary Keys

- Indexes solve a lot of problems. Make sure your each of your tables has a primary key. You will be quite unhappy later if they don’t, because everything is going to run much slower than it could.



B

built-in features, Ext
  client-side sorting 86
  column, reordering 86, 87
  columns, hidden 86
  columns, visible 86
button, toolbars
  creating 63
  handlers 67, 68
  icon buttons 67
  split button 64
buttons, form 53

custom events, creating 262-264

D

data, filtering
  about 238
  remote, filtering 238-244
data, finding
  about 237
  by field value 237
  by record ID 238
  by record index 237
data, formatting
  about 278

Indexes and Primary Keys

Primary Keys

coffee_shops

store_location
Plaka
Kolonaki
Monastiraki

coffee_shop_sales

transaction_date	transaction_qty	product	store_location
2024-10-01	3	Ellinikos Kafes	Plaka
2024-10-01	2	Loukoumades	Kolonaki
2024-10-02	5	Baklava	Monastiraki

Indexes and Primary Keys

Primary Keys

coffee_shops

store_location
Plaka
Kolonaki
Monastiraki
Kolonaki

coffee_shop_sales

transaction_date	transaction_qty	product	store_location
2024-10-01	3	Ellinikos Kafes	Plaka
2024-10-01	2	Loukoumades	Kolonaki
2024-10-02	5	Baklava	Monastiraki

Indexes and Primary Keys

Primary Keys

coffee_shops

store_id (PK)	store_location
1	Plaka
2	Kolonaki
3	Monastiraki
4	Kolonaki

coffee_shop_sales

transaction_date	transaction_qty	product	store_id (FK)
2024-10-01	3	Ellinikos Kafes	1
2024-10-01	2	Loukoumades	2
2024-10-02	5	Baklava	3

Indexes and Primary Keys

Primary Keys

coffee_shops

store_id (PK)	store_location
1	Plaka
2	Kolonaki
3	Monastiraki
4	Kolonaki

coffee_shop_sales

transaction_date	transaction_qty	product	store_id (FK)
2024-10-01	3	Ellinikos Kafes	1
2024-10-01	2	Loukoumades	2
2024-10-02	5	Baklava	3
2024-10-01	3	Ellinikos Kafes	1

Indexes and Primary Keys

Primary Keys

coffee_shops

store_id (PK)	store_location
1	Plaka
2	Kolonaki
3	Monastiraki
4	Kolonaki

coffee_shop_sales

transaction_id (PK)	transaction_date	transaction_qty	product	store_id (FK)
101	2024-10-01	3	Ellinikos Kafes	1
102	2024-10-01	2	Loukoumades	2
103	2024-10-02	5	Baklava	3
104	2024-10-01	3	Ellinikos Kafes	1

Indexes and Primary Keys

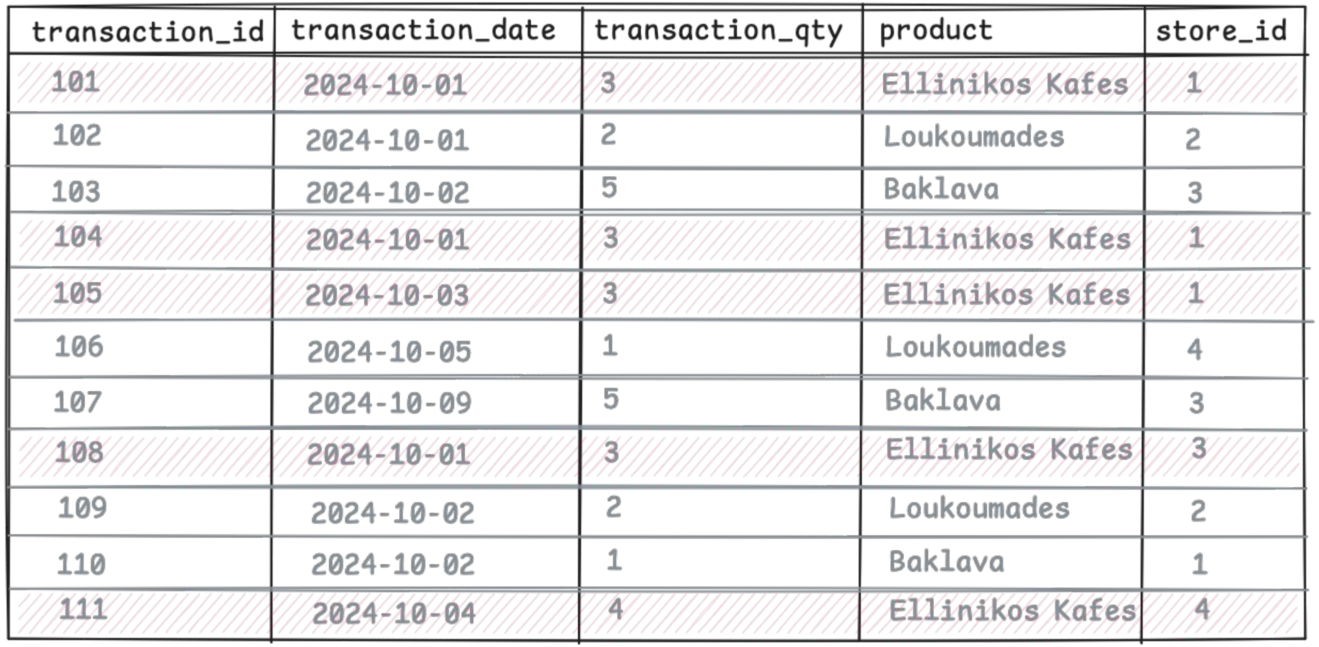
Primary Keys

https://www.postgresql.org/docs/current/ddl-constraints.html

- Uniquely identify a row in a table
- One or more columns
- Natural or surrogate
- Unique index
- Only one PK per table
- Needed for foreign key constraints

Indexes and Primary Keys

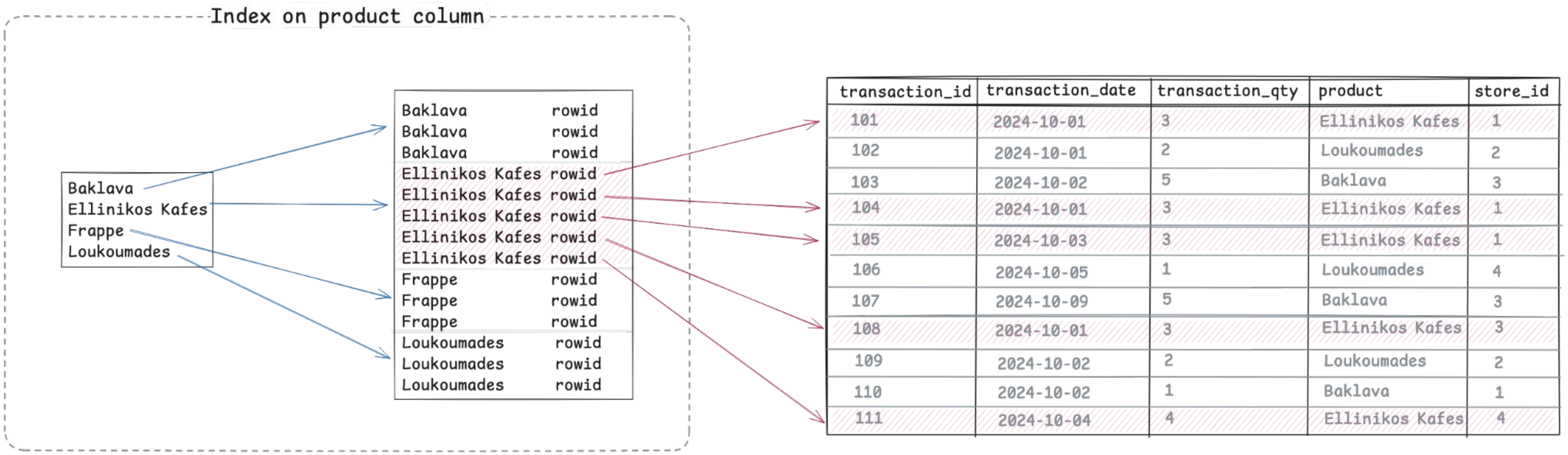
Indexes



transaction_id	transaction_date	transaction_qty	product	store_id
101	2024-10-01	3	Ellinikos Kafes	1
102	2024-10-01	2	Loukoumades	2
103	2024-10-02	5	Baklava	3
104	2024-10-01	3	Ellinikos Kafes	1
105	2024-10-03	3	Ellinikos Kafes	1
106	2024-10-05	1	Loukoumades	4
107	2024-10-09	5	Baklava	3
108	2024-10-01	3	Ellinikos Kafes	3
109	2024-10-02	2	Loukoumades	2
110	2024-10-02	1	Baklava	1
111	2024-10-04	4	Ellinikos Kafes	4

Indexes and Primary Keys

Indexes



transaction_id	transaction_date	transaction_qty	product	store_id
101	2024-10-01	3	Ellinikos Kafes	1
102	2024-10-01	2	Loukoumades	2
103	2024-10-02	5	Baklava	3
104	2024-10-01	3	Ellinikos Kafes	1
105	2024-10-03	3	Ellinikos Kafes	1
106	2024-10-05	1	Loukoumades	4
107	2024-10-09	5	Baklava	3
108	2024-10-01	3	Ellinikos Kafes	3
109	2024-10-02	2	Loukoumades	2
110	2024-10-02	1	Baklava	1
111	2024-10-04	4	Ellinikos Kafes	4

Indexes and Primary Keys

Indexes

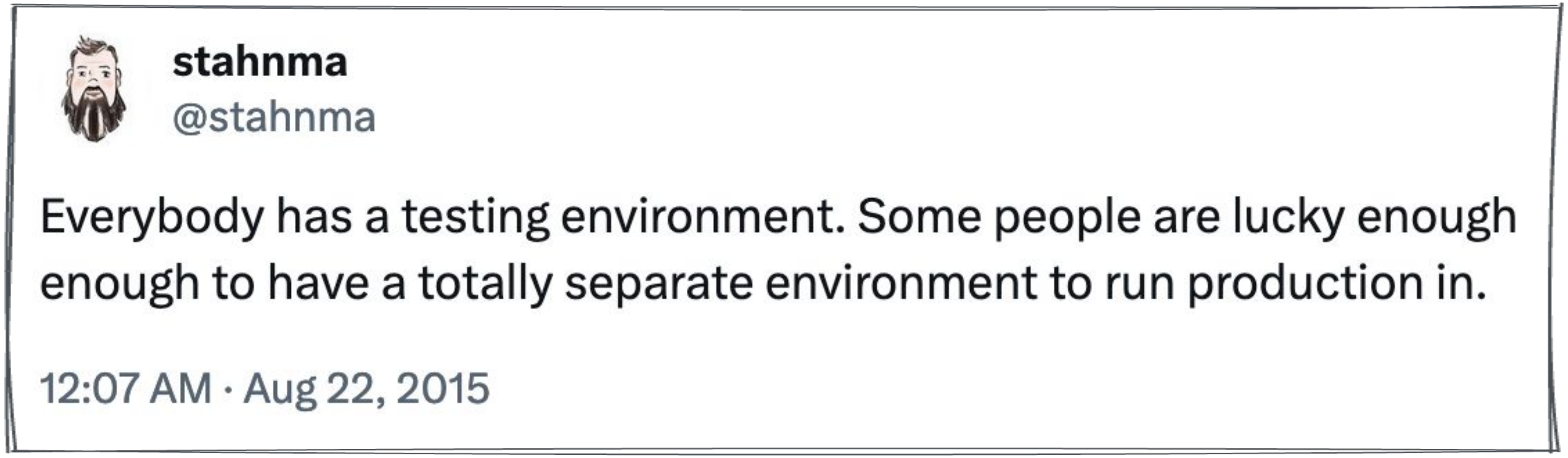
https://www.postgresql.org/docs/current/sql-createindex.html

- One or more columns
- Index on an expression
- Postgres index types: b-tree, Hash, GiST, SP-GiST, GIN, BRIN
- Default index: b-tree (balanced tree)
- Partial index
- Covering index

Testing

- Test the upgrade before you put it into production.

Testing



stahnma

@stahnma

Everybody has a testing environment. Some people are lucky enough enough to have a totally separate environment to run production in.

12:07 AM · Aug 22, 2015

Test on the *Test* Server

- Before running the test, make sure you are on the test server, not production.
- Check again before starting.

Test on the *Test* Server

“I accidentally connected to Production.”

DBA 5-word horror story

Version Upgrades, Great and Small

Version Upgrades, Great and Small

Minor Version Upgrades

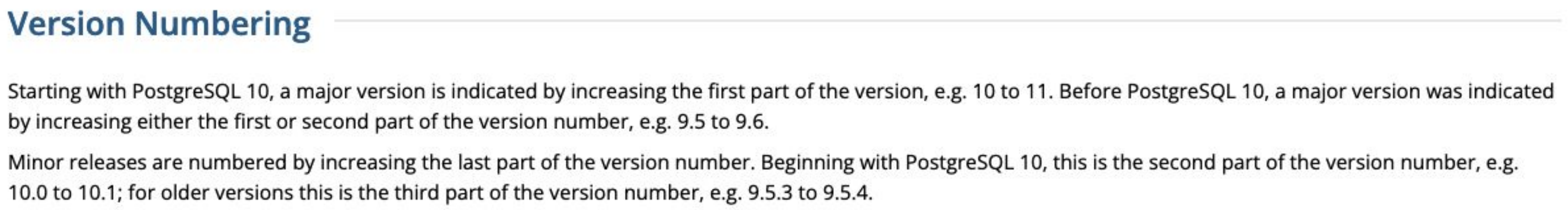
16.1: Also ensure that the `is_superuser` parameter is set correctly in such processes. No specific security consequences are known for that oversight, but it might be significant for some extensions.

16.2: Tighten security restrictions within `REFRESH MATERIALIZED VIEW CONCURRENTLY` (Heikki Linnakangas)
One step of a concurrent refresh command was run under weak security restrictions. If a materialized view's owner could persuade a superuser or other high-privileged user to perform a concurrent refresh on that view, the view's owner could control code executed with the privileges of the user running `REFRESH`. Fix things so that all user-determined code is run as the view's owner, as expected.

16.3: However, a security vulnerability was found in the system views `pg_stats_ext` and `pg_stats_ext_exprs`, potentially allowing authenticated database users to see data they shouldn't. If this is of concern in your installation, follow the steps in the first changelog entry below to rectify it.

Version Upgrades, Great and Small

Version Numbering

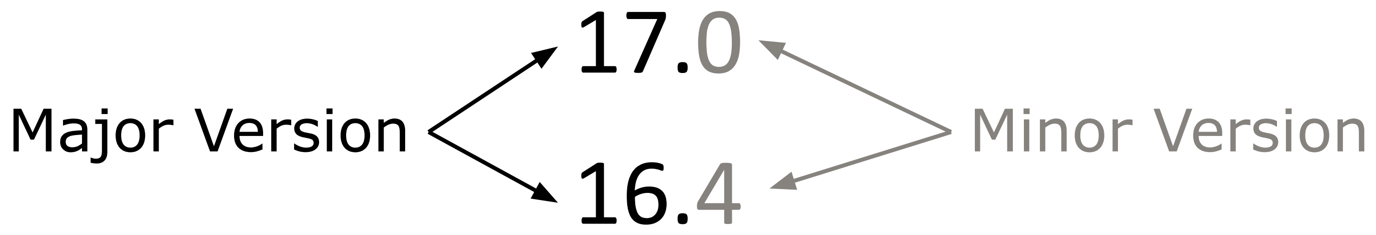


Version Numbering

Starting with PostgreSQL 10, a major version is indicated by increasing the first part of the version, e.g. 10 to 11. Before PostgreSQL 10, a major version was indicated by increasing either the first or second part of the version number, e.g. 9.5 to 9.6.

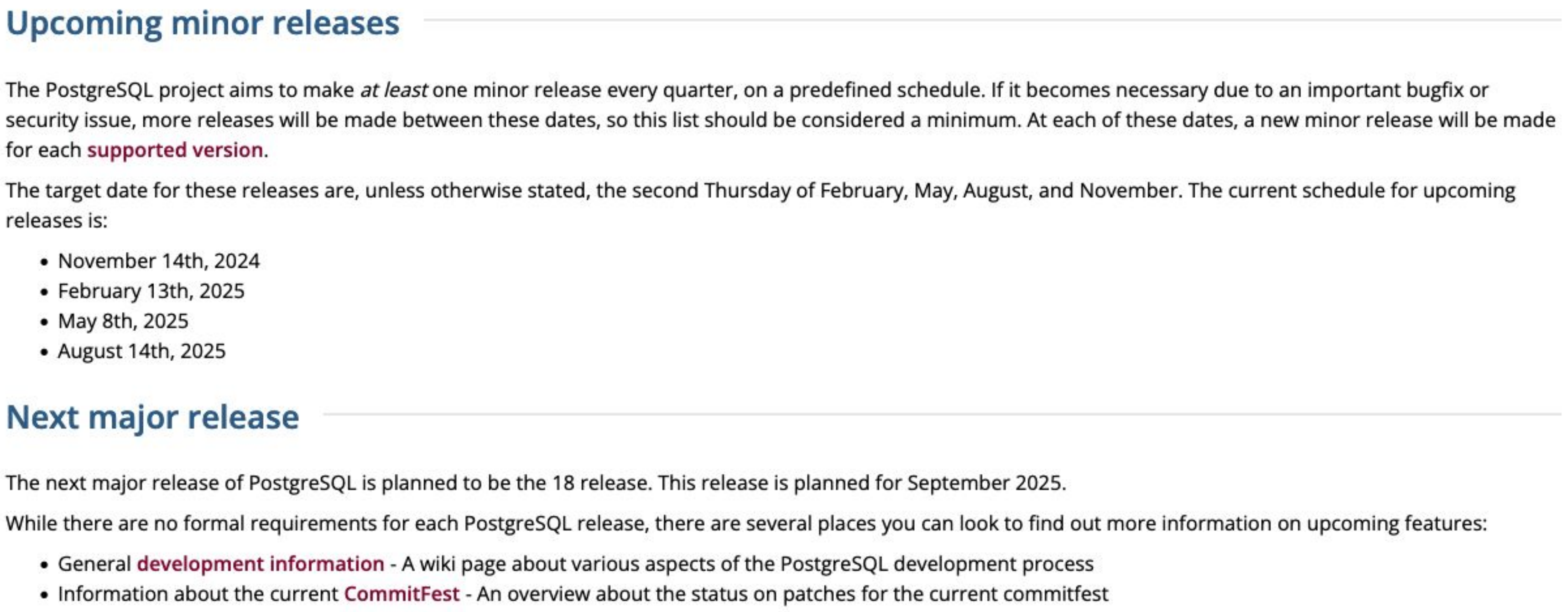
Minor releases are numbered by increasing the last part of the version number. Beginning with PostgreSQL 10, this is the second part of the version number, e.g. 10.0 to 10.1; for older versions this is the third part of the version number, e.g. 9.5.3 to 9.5.4.

https://www.postgresql.org/support/versioning



Version Upgrades, Great and Small

Release Roadmap



Upcoming minor releases

The PostgreSQL project aims to make *at least* one minor release every quarter, on a predefined schedule. If it becomes necessary due to an important bugfix or security issue, more releases will be made between these dates, so this list should be considered a minimum. At each of these dates, a new minor release will be made for each **supported version**.

The target date for these releases are, unless otherwise stated, the second Thursday of February, May, August, and November. The current schedule for upcoming releases is:

- November 14th, 2024
- February 13th, 2025
- May 8th, 2025
- August 14th, 2025

Next major release

The next major release of PostgreSQL is planned to be the 18 release. This release is planned for September 2025.

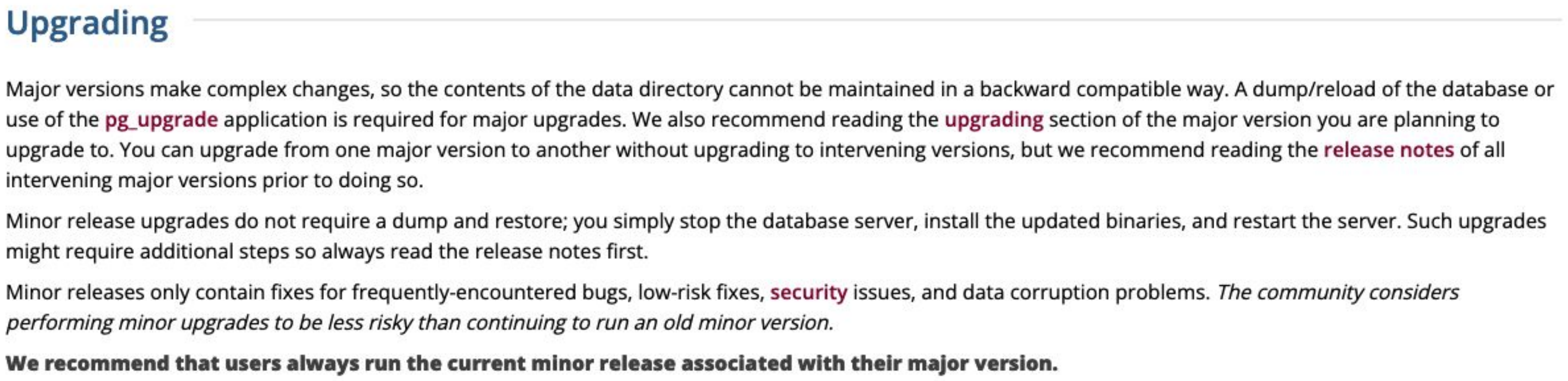
While there are no formal requirements for each PostgreSQL release, there are several places you can look to find out more information on upcoming features:

- General **development information** - A wiki page about various aspects of the PostgreSQL development process
- Information about the current **CommitFest** - An overview about the status on patches for the current commitfest

https://www.postgresql.org/developer/roadmap/

Version Upgrades, Great and Small

Minor Version Upgrades



Upgrading

Major versions make complex changes, so the contents of the data directory cannot be maintained in a backward compatible way. A dump/reload of the database or use of the pg_upgrade application is required for major upgrades. We also recommend reading the upgrading section of the major version you are planning to upgrade to. You can upgrade from one major version to another without upgrading to intervening versions, but we recommend reading the release notes of all intervening major versions prior to doing so.

Minor release upgrades do not require a dump and restore; you simply stop the database server, install the updated binaries, and restart the server. Such upgrades might require additional steps so always read the release notes first.

Minor releases only contain fixes for frequently-encountered bugs, low-risk fixes, security issues, and data corruption problems. *The community considers performing minor upgrades to be less risky than continuing to run an old minor version.*

We recommend that users always run the current minor release associated with their major version.

https://www.postgresql.org/support/versioning

Version Upgrades, Great and Small

Minor Version Upgrades

“We have to use PostgreSQL 15.3.”

Anonymous PostgreSQL User

Version Upgrades, Great and Small

Minor Version Upgrades

“The community considers performing minor upgrades to be ***less risky*** *than continuing to run an old minor version.”*

“We recommend that users always run the ***current minor release*** *associated with their major version.”*

https://www.postgresql.org/support/versioning

Version Upgrades, Great and Small

Major Version Upgrades

“We have to use PostgreSQL 13.”

Anonymous PostgreSQL User

Version Upgrades, Great and Small

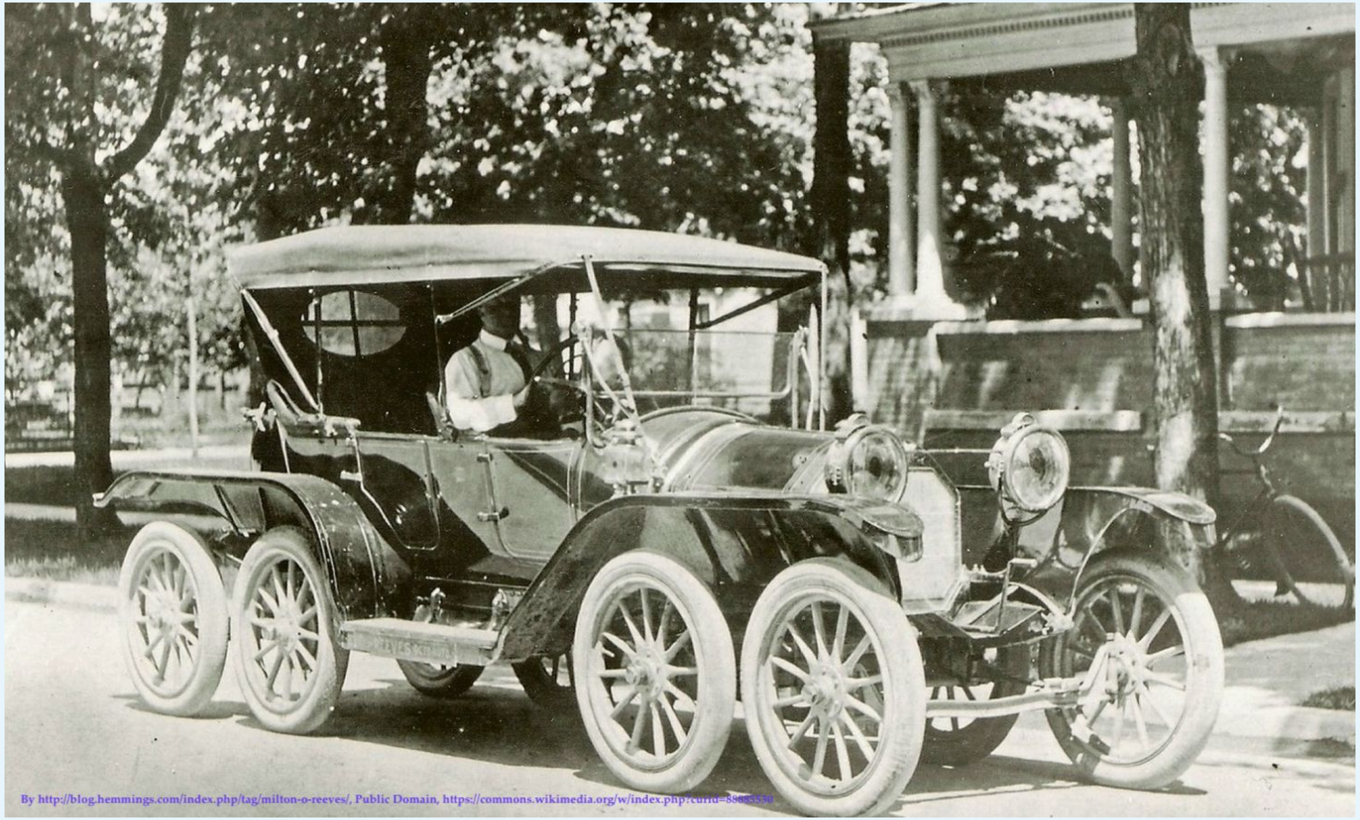
Major Version Upgrades

Queries

- Your super-cool custom query is a likely source of your problems.

```
 WITH items_externalsales AS (
   SELECT item_id, COUNT(*) as sales, SUM(price) as revenue FROM items_itempriorsale GROUP BY item_id
),
items_sales AS (
SELECT m.id,
       SUM(pol.quantity) AS quantity,
       SUM(pol.quantity * pol.price_each) AS revenue
      FROM items_item m
      JOIN items_itemproduct mp ON mp.item_id = m.id
      JOIN products_product p ON p.id = mp.product_id
      JOIN orders_productorderline pol ON pol.product_id = p.id
      JOIN orders_orderline ol ON ol.id = pol.orderline_ptr_id
      JOIN orders_order o ON o.id = ol.order_id
      WHERE ol.status = 'DONE'
      GROUP BY m.id
),
item_ratings AS (
SELECT m.id as item_id, avg(r.rating)::numeric(2,1) as rating
  FROM items_item m
  JOIN ratings_rating r ON m.id = r.item_id
  WHERE r.rating > 0
  GROUP BY m.id
  HAVING COUNT(*) >= 5
)
SELECT m.*,
       CASE WHEN days_asnew > 0 THEN revenue / days_asnew ELSE 0 END as revenue_per_day
 FROM (
   SELECT m.title,
         m.asnew_at as asnew_at,
          CASE WHEN m.asnew_at <= '2007-12-24'::DATE THEN TRUE ELSE FALSE END as approximate,
          m.active as active,
          m.special as special,
```

Queries



Queries

```
  month  | store_id | total_monthly_sales
---------+----------+--------------------
-
 2023-01 |        4 |            14153.00
 2023-02 |        4 |            13392.00
 2023-03 |        4 |            17246.00
 2023-04 |        4 |            20594.00
 2023-05 |        4 |            27312.00
 2023-06 |        4 |            28737.00
```

```
EXPLAIN ANALYZE
WITH ranked_sales AS (
  SELECT
    store_id,
    DATE_TRUNC('month', transaction_date) AS month,
    SUM(transaction_qty) AS total_monthly_unit_sales,
    RANK() OVER (PARTITION BY DATE_TRUNC('month', transaction_date)
    ORDER BY SUM(transaction_qty) DESC) AS sales_rank
  FROM coffee_shop_sales
  WHERE date_trunc('year', transaction_date) = date_trunc('year', now()) - interval '1 year'
  GROUP BY 1, 2)
SELECT
  to_char(month, 'YYYY-MM') AS month,
  store_id,
  round(total_monthly_unit_sales,2) AS total_monthly_sales
FROM ranked_sales
WHERE sales_rank = 1
ORDER BY month;
```

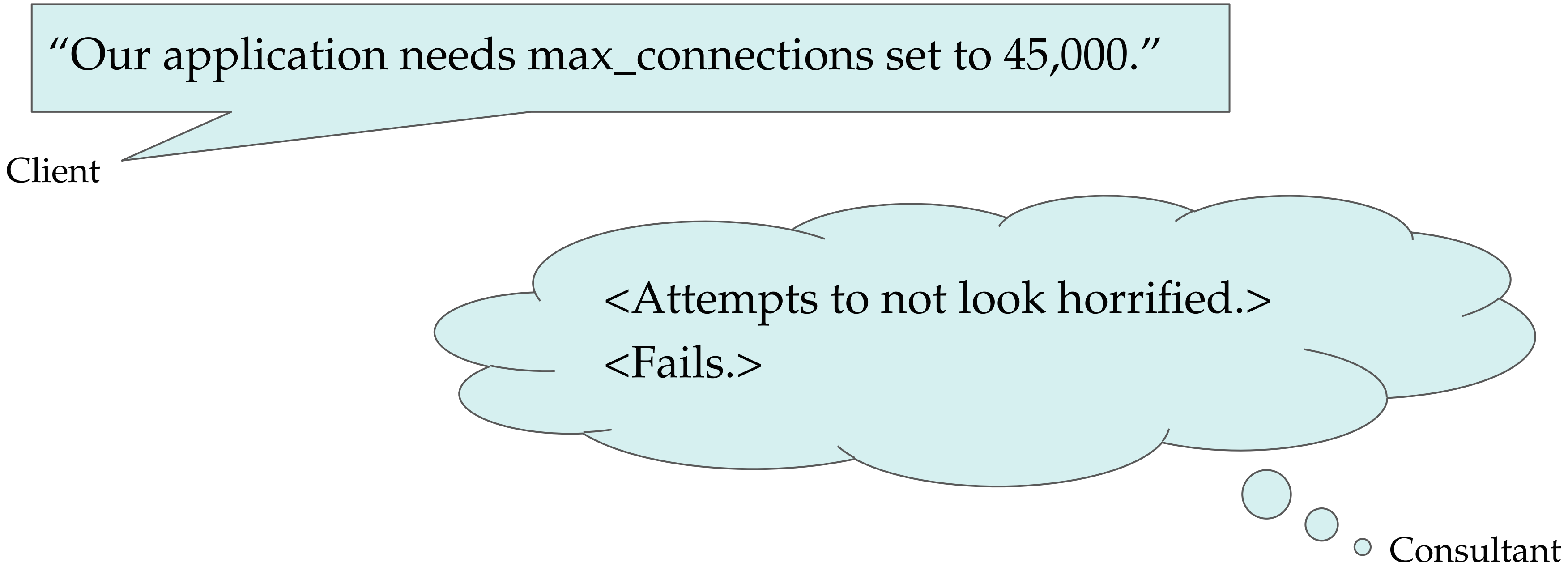
Queries

Execution Plans

```
                                                                                               QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=5087.15..5087.16 rows=2 width=68) (actual time=120.869..120.885 rows=6 loops=1)
   Sort Key: (to_char(ranked_sales.month, 'YYYY-MM'::text))
   Sort Method: quicksort  Memory: 25kB
   ->  Subquery Scan on ranked_sales  (cost=5069.65..5087.14 rows=2 width=68) (actual time=120.699..120.722 rows=6 loops=1)
         Filter: (ranked_sales.sales_rank = 1)
         ->  WindowAgg  (cost=5069.65..5081.30 rows=466 width=28) (actual time=120.691..120.712 rows=6 loops=1)
                  Run Condition: (rank() OVER (?) <= 1)
                  ->  Sort  (cost=5069.65..5070.82 rows=466 width=20) (actual time=120.686..120.703 rows=24 loops=1)
                  Sort Key: (date_trunc('month'::text, (coffee_shop_sales.transaction_date)::timestamp with time zone)), (sum(coffee_shop_sales.transaction_qty)) DESC
                  Sort Method: quicksort  Memory: 26kB
                  ->  Finalize GroupAggregate  (cost=4978.36..5049.00 rows=466 width=20) (actual time=114.155..120.691 rows=24 loops=1)
                           Group Key: coffee_shop_sales.store_id, (date_trunc('month'::text, (coffee_shop_sales.transaction_date)::timestamp with time zone))
                           ->  Gather Merge  (cost=4978.36..5038.72 rows=439 width=20) (actual time=114.078..120.683 rows=48 loops=1)
                                    Workers Planned: 1
                                    Workers Launched: 1
                                    ->  Partial GroupAggregate  (cost=3978.35..3989.32 rows=439 width=20) (actual time=102.044..108.127 rows=24 loops=2)
                                    Group Key: coffee_shop_sales.store_id, (date_trunc('month'::text, (coffee_shop_sales.transaction_date)::timestamp with time zone))
                                    ->  Sort  (cost=3978.35..3979.44 rows=439 width=16) (actual time=101.955..104.663 rows=74558 loops=2)
                                             Sort Key: coffee_shop_sales.store_id, (date_trunc('month'::text, (coffee_shop_sales.transaction_date)::timestamp with time zone))
                                             Sort Method: external merge  Disk: 2296kB
                                             Worker 0:  Sort Method: external merge  Disk: 2096kB
                                             ->  Parallel Seq Scan on coffee_shop_sales  (cost=0.00..3959.08 rows=439 width=16) (actual time=0.034..90.534 rows=74558 loops=2)
                                                      Filter: (date_trunc('year'::text, (transaction_date)::timestamp with time zone) = (date_trunc('year'::text, now()) - '1
year'::interval))
 Planning Time: 0.382 ms
 Execution Time: 121.485 ms
```

Your Application

The problem isn't always the database. Look at your app.

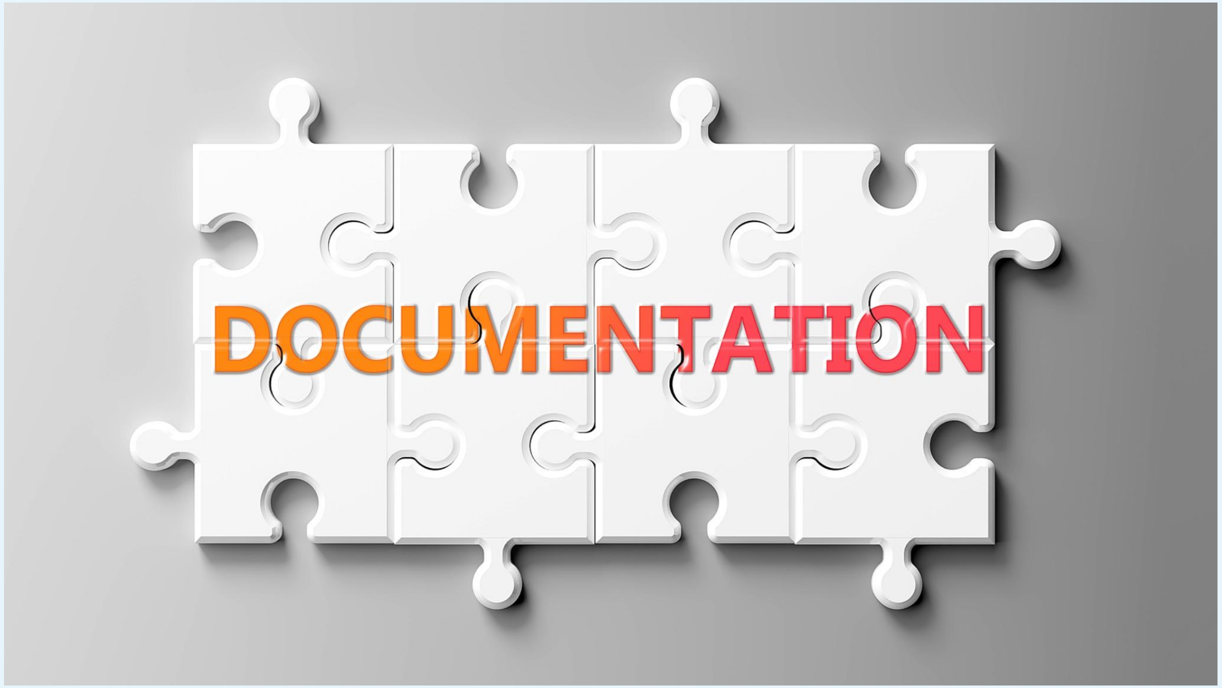

"Our application needs max_connections set to 45,000."
Client
<Attempts to not look horrified.>
<Fails.>
Consultant

Your Application

- max_connections parameter
  - Entire PostgreSQL instance
  - Default: 100
- Connection pooling
  - Separate pools for different use-cases
  - Don’t keep too many connections open
  - Minimise rapidly opening and closing connections

How to Avoid Problems

Read the documentation.



How to Avoid Problems

Read the Docs

Current version:

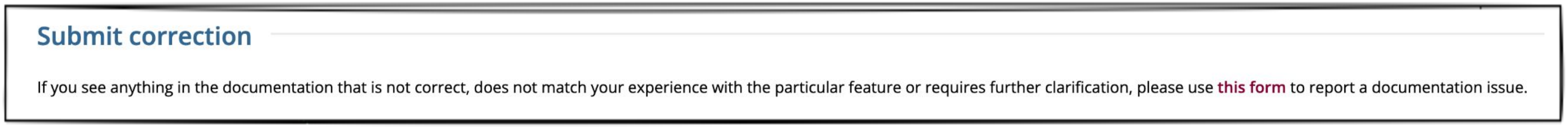
https://www.postgresql.org/docs/current/index/html

Hitchhiker's Guide to the Postgres Documentation:

https://l_avrot.gitlab.io/slides/doc_20220513.html#

How to Avoid Problems

Help to Make the Docs Better

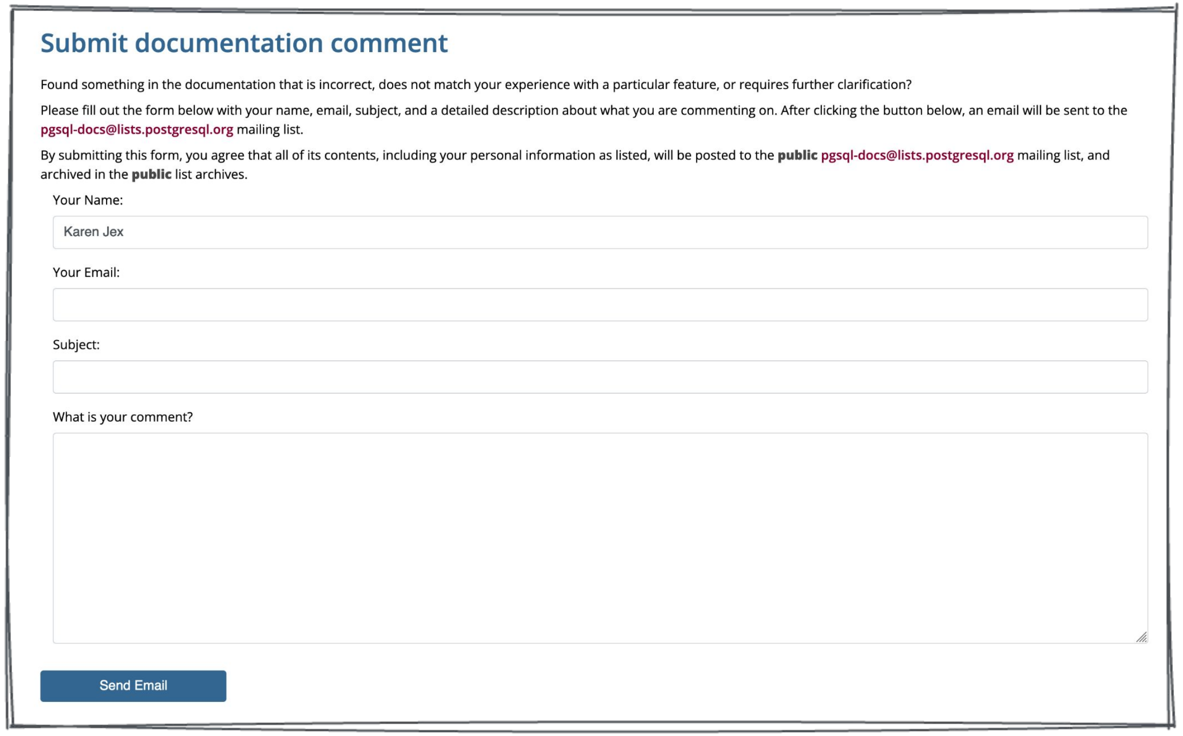

Submit correction

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.

"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."

How to Avoid Problems

Help to Make the Docs Better



Submit documentation comment

Found something in the documentation that is incorrect, does not match your experience with a particular feature, or requires further clarification?

Please fill out the form below with your name, email, subject, and a detailed description about what you are commenting on. After clicking the button below, an email will be sent to the pgsql-docs@lists.postgresql.org mailing list.

By submitting this form, you agree that all of its contents, including your personal information as listed, will be posted to the **public** pgsql-docs@lists.postgresql.org mailing list, and archived in the **public** list archives.

Your Name: Karen Jex

Your Email:

Subject:

What is your comment?

How to Avoid Problems

If you don’t know, ask. The only stupid question is the one you should have asked, and didn’t—and now you have severe problems, such as data loss, data corruption, or a server shutdown.

How to Avoid Problems

Ask Questions

PostgreSQL Mailing Lists:	https://lists.postgresql.org/
Postgres Slack:	postgresteam.slack.com

How to Avoid Problems

Wiki and Blogs

PostgreSQL Wiki:

https://wiki.postgresql.org/wiki/Main_Page

Planet PostgreSQL:

https://planet.postgresql.org/

How to Avoid Problems

Talk Recordings

PostgreSQL Europe	https://www.youtube.com/@pgeu
PgUS	https://www.youtube.com/@pgus
SF PUG	https://www.youtube.com/@sanfranciscobayareapostgre4592

How to Avoid Problems

And More

- Podcasts
- Conferences
- Meetups
- User Groups
- …

Questions?

Feedback!

https://www.postgresql.eu/events/pgconfeu2024/feedback

https://www.postgresql.eu/events/pgconfeu2024/feedback/5870/

Thank you!