PostgreSQL and Software Engineers

A Database from Software Engineering Perspective

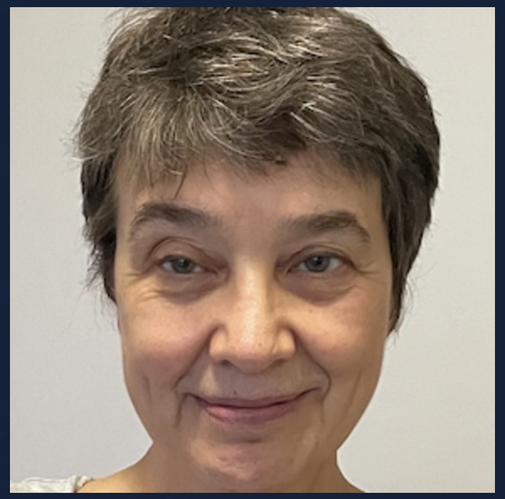
Hettie Dombrovskaya
Database Architect

PG Day Paris
2023

Who Am I

Database Architect at DRW
Local Organizer of Chicago PostgreSQL User Group

PG Day Chicago is on April 20, 2023!



Why This Topic?

- Because SE are our first and true customers, but they often have no voice in decision-making
- Because nobody wants to talk about these issues
- Because I work with SE and hear their complaints

Why Software Engineers are unhappy?

Because they can't work with databases using familiar techniques

- DB design tools
- Version control
- Deployments
- Tooling
- Security and Access management

… To be continued

Design Tools

Do we have anything to offer?

We often ask SE to stay away from designing tables.

But do they have other choices? It not tables, then what?

NORM-GEN project partially addresses this problem

ORM and Code Generation

We hate ORM, but..

ORM provides the missing element

App Developers won’t go back to hand-writing everything (and there is a ChatGPT anyway!)

We do not have anything like that in PostgreSQL – any tool which would help to generate the DB code

Version Control and Compare

How can we version a database?

For the start, a database can be perfectly fine not storing any code anywhere!

You use GitHub - greats, but your database can live without it

Is there any easy way to tell the differences between two databases/schemas?

What makes two database objects different?

- If the order of columns is different, are the tables different?
- If a constraint name is different, are tables different?

Deployments!!!

What does it mean to deploy a database change?

- When you deploy a new version of an application, you just compile the code. OK, may be not “just”, but still.. Whatever is in the GitHub, that’s what is running
- You can exactly do it with database objects…

What are the options?

- Request a separate deployment script
- Automatically generate a patch based on the source code diff
- And we are back to the question of what exactly makes two tables different

How to tell whether functions are different?

Tooling!

"Here is the library I am using" vs. "Let me give you a script"

We do not have tools … for anything

- How to check for tables sizes?
- How to check bloat?
- How to check which process is blocking me?

For pretty much all of that, there are only "scripts"

- They are all over internet and personal hard drives
- Even professional consultants do not have repositories for "these scripts."
- We "should not" use PostgreSQL catalog, but we use it anyway => versioning

Nobody validates them against any changes in versions, hardware, etc.

Security and Access Management

We hate it when applications are connected to a DB as superuser, but there is a reason for that!

How can you find all permission for a specific user?

- Do we have a command for that?
- And no, I do not mean the list of all granted roles

How can you compare permissions for two different users?

- Can two users do all the same things?
- Does it matter which role granted these privileges?

How to compare permissions in different environments?

We do not have solutions for everything, but we have some!

If we want to compare two environments,
We do not look at the source code
And we do not look at deployments' logs
We look at PostgreSQL catalog(s)

https://github.com/hettie-d/diff

Welcome to DIFF!

DIFF addresses three of the five mentioned issues:

- ***Versioning***
- ***Deployments***
- ***Access control***

How DIFF works

- Clone the repo and run _load_all.sql from the root directory to install **locally**. Do not install DIFF in the target databases. The installation includes adding postgres_fdw extension
- Setup each of the environments you want to compare calling

```
diff.catalog_fdw_setup(
   in p_database_alias text,
   in p_database text,
   in p_host text default 'localhost'::text,
   in p_port text default null::text,
   in p_user text default null::text,
   in p_password text default null::text)
```

What can DIFF do

For any pair of databases

Compare:

- List of schemas/ownership
- List of tables/views/mviews in schema
- List of columns in the table(s)
- Column details (types, defaults, nullables)
- List of constraints
- Permissions

Generate patches

- to make one environment to look exactly like another one

Compare schemas

```
select *
from diff.schema_compare(
‘airlines’,
‘hettie’);
```

Data Output | Explain | Messages | Notifications

	location text	object_name text	object_type text	object_owner text
1	airlines	temporal_relationships	schema	hettie
2	airlines	postgres_air_large	schema	postgres
3	airlines	bt_tutorial	schema	postgres
4	airlines	norm	schema	hettie
5	airlines	bitemporal_internal	schema	hettie

Compare tables/mviews /views

```
select *
from diff.tables_compare(
'airlines’,
'hettie’,
'postgres_air');
```

	location text	object_name text	object_type text	object_owner text
1	airlines	booking_name	table	postgres
2	airlines	flight_calc	view	postgres
3	hettie	flight_calc	table	postgres
4	airlines	flight_departure	view	postgres
5	airlines	flight_departure_mv	matview	postgres
6	airlines	flight_stats	view	postgres

Compare columns

```
select *
from diff.columns_compare(
'airlines’,
'hettie’,
'postgres_air');
```

	location text	table_name text	column_name text	data_type text
5	airlines	booking_name	destination	text
6	airlines	booking_name	flight_id	integer
7	airlines	booking_name	leg_num	integer
8	airlines	booking_name	rank	bigint
9	airlines	flight_calc	departure_airport	character
10	hettie	flight_calc	departure_airport	text
11	airlines	flight_calc	flight_id	integer
12	hettie	flight_calc	flight_id	bigint
13	airlines	flight_departure	departure_airport	character
14	airlines	flight_departure	departure_date	date
15	airlines	flight_departure	flight_id	integer
16	airlines	flight_departure	num_passengers	bigint

Compare columns in a table

```
select * from diff.columns_compare(
‘airlines’,
‘hettie’,
’postgres_air’,
‘'frequent_flyer’);
```

	location text	table_name text	column_name text	data_type text
1	hettie	frequent_flyer	secondary_email	text

Complete columns compare

```
select *
from diff.full_columns_compare(
'hettie’,
'airlines’,
'postgres_air’,
'frequent_flyer');
```

	location text	table_name text	ordinal_position integer	column_name text	data_type text	nullable text	default_val text
1	airlines	frequent_flyer	8	email	text	NOT NU...	[null]
2	hettie	frequent_flyer	8	email	text		[null]
3	hettie	frequent_flyer	11	secondary_email	text		[null]

Constraints compare

```
select * from diff.constraint_compare('airlines’,
'hettie’,
'postgres_air')
```

	location text	table_name text	constraint_type text	ref_table text	const_def text
1	hettie	postgres_ai...	foreign key	postgres...	FOREIGN KEY (frequent_flyer_id) REFE...
2	airlines	postgres_ai...	foreign key	postgres...	FOREIGN KEY (aircraft_code) REFERE...

Generate patch

```
select * from diff.generate_patch_table (’hettie’,
‘airlines’,
’postgres_air’,
‘frequent_flyer’);

   alter table postgres_air.frequent_flyer
   add secondary_email text ;
   alter column email drop NOT NULL;

select * from diff.generate_patch_table(‘airlines’,
’hettie’,
’postgres_air’,
‘frequent_flyer’);

   alter table postgres_air.frequent_flyer drop secondary_email;
   alter column email set NOT NULL
```

Generate constraint patch

```
select * from diff.generate_patch_constraint('airlines’,
'hettie’,
'postgres_air’,
'account');
```

```
alter table  postgres_air.account drop constraint frequent_flyer_id_fk;
```

```
select * from diff.generate_patch_constraint('hettie’,
'airlines’,
'postgres_air’,
'account’);
```

```
alter table  postgres_air.account add constraint frequent_flyer_id_fk
FOREIGN KEY (frequent_flyer_id) REFERENCES
postgres_air.frequent_flyer(frequent_flyer_id)
```

Tooling

No more "Let me give you a script"

dba_tools schema

All tools are ”packaged”, and deployed in each new database.

Deployment is repeatable

New GitHub commit => deployment

Permissions management package

- Security definer function & event triggers
- Enables granting individual permissions to no-login roles and no-login roles to users
- Objects ownership and default permissions

Objects sizes package

- Tables/indexes sizes
- Tables/indexes bloat
- All you can find in Postgres Wiki and anywhere on the internet☺

To be expanded

- Anything we use more than twice, is packaged
- Any DBA on call can use it straight from DB
- PG version -independent

Access Management

Do not let me start!

How can you figure out what a user can and can’t do?

- **There is no easy way!**

Compare privileges on schemas

```
select * from diff.privs_compare(
'airlines’,
'hettie');
```

	location text	schema_name text	user_name text	object_type text	permission text
1	airlines	bitemporal_internal	hettie	schema	CREATE
2	airlines	bitemporal_internal	hettie	schema	USAGE
3	airlines	postgres_air_large	reporting	schema	USAGE
4	airlines	temporal_relationships	hettie	schema	USAGE
5	airlines	temporal_relationships	hettie	schema	CREATE

Compare privileges on tables

```
select * from
diff.privs_compare('airlines’,
'hettie’,
'postgres_air');
```

	location text	table_name text	user_name text	permission text
1	hettie	flight_calc	reporting	SELECT

Compare privileges on tables

```
select * from diff.privs_compare(
'airlines’,
'hettie’,
'postgres_air_large');
```

	location text	table_name text	user_name text	permission text
1	airlines	boarding_pass_aug	reporting	SELECT
2	airlines	boarding_pass_july	reporting	SELECT
3	airlines	boarding_pass_june	reporting	SELECT
4	airlines	boarding_pass_large	reporting	SELECT
5	airlines	boarding_pass_may	reporting	SELECT
6	airlines	booking_json	reporting	SELECT
7	airlines	booking_jsonb	reporting	SELECT
8	airlines	custom_field	reporting	SELECT
9	airlines	passenger_passport	reporting	SELECT

Compare privileges on schemas

```
select * from
diff.priv_schema_compare(‘airline
s’,’hettie’);
```

Data Output

	location text	schema_name text	user_name text	object_type text	permission text
1	airlines	bitemporal_internal	hettie	schema	CREATE
2	airlines	bitemporal_internal	hettie	schema	USAGE
3	airlines	postgres_air_large	reporting	schema	USAGE
4	airlines	temporal_relationships	hettie	schema	USAGE
5	airlines	temporal_relationships	hettie	schema	CREATE

Select privileges which are granted directly

```
select * from
diff.db_privs_direct_select('hettie');
```

	object_type text	object_name text	user_name text	schema_default_priv text	permission text
1	schema priv	postgres_air	hettie	schema	USAGE
2	schema priv	postgres_air	hettie	schema	CREATE
3	schema priv	postgres_air	reporting	schema	USAGE
4	table priv	postgres_air.flight_calc	reporting	n/a	SELECT

Compare all privileges

Different sets of privileges can result in identical sets of object privileges.

```
grant select on all tables in schema sch to new_user;

grant select on sch.t1 to new_user;
grant select on sch.t2 to new_user;
…

grant select on sch.tn to new_user;

grant select on all tables in schema sch to sch_read_role;
grant sch_read_role to new_user;
```

How to compare the final result?

Dealing with recursive roles

```
WITH RECURSIVE x AS(
  SELECT member::regrole,
         roleid::regrole AS role,
       roleid,
         member::regrole || ' -> ' || roleid::regrole AS path
  FROM pg_auth_members AS m
  UNION ALL
  SELECT x.member::regrole,
         m.roleid::regrole,
       m.roleid,
         x.path || ' -> ' || m.roleid::regrole
 FROM pg_auth_members AS m
    JOIN x ON m.member = x.role
  )
  SELECT member, role, roleid, path
  FROM x
  WHERE member::text not like 'pg%'
  AND member::text!='postgres'
  AND member::text not like 'rds%'
  and role::text not like 'pg%'
```

The whole function is 117 lines long …

Select all privileges on a database

```
select * from
diff.db_privs_select ('hettie’)
```

	object_type text	object_name text	user_name text	schema_default_priv text	permission text
1	schema priv	bitemporal_internal	hettie	schema	USAGE
2	schema priv	bitemporal_internal	hettie	schema	CREATE
3	schema priv	postgres_air_large	reporting	schema	USAGE
4	schema priv	postgres_air	hettie	schema	USAGE
5	schema priv	postgres_air	hettie	schema	CREATE
6	schema priv	postgres_air	reporting	schema	USAGE
7	schema priv	temporal_relationships	hettie	schema	USAGE
8	schema priv	temporal_relationships	hettie	schema	CREATE
9	table priv	postgres_air_large.cust…	reporting	n/a	SELECT
10	table priv	postgres_air_large.pass…	reporting	n/a	SELECT
11	table priv	postgres_air_large.boar…	reporting	n/a	SELECT
12	table priv	postgres_air_large.book…	reporting	n/a	SELECT
13	table priv	postgres_air_large.book…	reporting	n/a	SELECT
14	table priv	postgres_air_large.boar…	reporting	n/a	SELECT
15	table priv	postgres_air_large.boar…	reporting	n/a	SELECT
16	table priv	postgres_air_large.boar…	reporting	n/a	SELECT
17	table priv	postgres_air_large.boar…	reporting	n/a	SELECT

Future work

- Compare indexes
- Compare triggers
- Compare functions and procedures
- Finalize all patches generation

What should be documented in PostgreSQL?

Other issues

- Usage of pgTap
- Designing tools
- Test data sets
- Branching data

Q&A

Hettie Dombrovskaya
Database Architect DRW

hdombrovska@drwholdings.com

www.drw.com