How to Work with Software Engineers

Hettie Dombrovskaya
Senior Cloud consultant

Nordic PG Day 2022
Helsinki Mar 22, 2022

EDB

A Database From the SE Perspective

Why This Topic?

- Because SE are our first customers, and no body can “see” a database unless it is presented by an application
- Because (almost) nobody talks about these issues
- Because I want to give a new perspective to all of these old well-known problems

Why software engineers are unhappy?

- DB design tools
- Version control
- Deployment
- Access management
- Data branching

... and the list goes on!

Design tools

We often ask SE to stay away from designing tables

But what other choices they have?

It not tables – then what?

Version control and compare

- For 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 tables different?

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

Deployments!!!!

- 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…
- So 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
  - Function bodies diffs !!!!

Security/access management

- How to find all permissions for specific users?
- How to compare permissions of two different users?
- How to compare permissions in different environments?

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

How we compare schemas in different databases?

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

Using FDW for everything!

```
create or replace procedure diff.catalog_fdw_setup (
  p_database_alias text,
  p_database text,
  p_host text default 'localhost',
  p_port text default null,
  p_user text default null,
  p_password text default null)
language plpgsql
as
$BODY$
declare
  v_port text;
  v_user text;
  v_database_alias text;
  v_error text;
begin
  v_port :=coalesce(p_port, '5432');
  v_user :=coalesce (p_user, current_user);
  v_database_alias:=coalesce (p_database_alias, p_database);
  execute
    $$drop server if exists fs_$$||v_database_alias||
    $$ cascade;
    create server fs_$$||v_database_alias||
    $$ FOREIGN DATA WRAPPER postgres_fdw options
       (host $$||quote_literal(p_host) ||$$,
        port $$||quote_literal(v_port)||$$,
        dbname $$||quote_literal(p_database)||
        $$)$$;

  execute $$create user mapping for
  public server fs_$$||v_database_alias||
  execute $$create user mapping for
  public server fs_$$||v_database_alias||
        $$ OPTIONS (user $$||quote_literal(v_user)||
           case when p_password is not null then $$, password $$||quote_literal(p_password)
            else $$ $$
            end ||
            $$)$$;
  execute $$grant usage on foreign server  fs_$$||v_database_alias||$$ to public$$;
  execute $$drop schema if exists  $$||v_database_alias||$$_catalog_ft cascade$$;
  execute $$create schema $$||v_database_alias||$$_catalog_ft$$;
  execute $$grant usage on schema $$||v_database_alias||$$_catalog_ft to public$$;
  execute $$drop schema if exists $$||v_database_alias||$$_info_ft cascade$$;
  execute $$create schema $$||v_database_alias||$$_info_ft$$;
  execute $$grant usage on schema $$||v_database_alias||$$_info_ft to public$$;
  execute $$import foreign schema "pg_catalog" except   (pg_attribute,
                                          pg_replication_slots,
                                         pg_statistic,
                                         pg_stats)from server
        fs_$$||v_database_alias||$$ into $$||v_database_alias||$$_catalog_ft$$;

  execute $$create foreign table $$||v_database_alias||$$_catalog_ft.pg_attribute(
     attrelid oid,
     attname name,
     atttypid oid,
     ttstattarget integer,
     attlen smallint,
     attnum smallint,
     attndims integer,
     attcacheoff integer,
     atttypmod integer,
     attbyval boolean,
     attstorage "char"
```

What we can compare

- List of schemas/ownership
- List of tables/views/mviews in schema
- List of columns in the table(s)
- List of constraints
- Permissions

And then we can generate patches to get one environment
to look exactly as another one

```
38  create or replace function diff.schema_compare(p_source_1 text, p_source_2 text)
39  returns setof diff.object_diff_record
40  language plpgsql
41  as
42  $body$
43  begin
44  return query
45  execute
46  $sql$ select $sql$|| quote_literal(p_source_1)||
47  $sql$,
48   a.* from
49  (select schema_name::text,schema_owner::text from $sql$||p_source_1||
50    $sql$_info_ft.schemata
51   where schema_name not like 'pg_%'
52  except
53  select schema_name::text,schema_owner::text from $sql$||p_source_2||
54    $sql$_info_ft.schemata
55   where schema_name not like 'pg_%') a
56  union all
57  select $sql$|| quote_literal(p_source_2)||
58  $sql$,
59   a.* from
60  (select schema_name::text,schema_owner::text from $sql$||p_source_2||
61    $sql$_info_ft.schemata
62   where schema_name not like 'pg_%'
63  except
64  select schema_name::text,schema_owner::text from $sql$||p_source_1||
65    $sql$_info_ft.schemata
66   where schema_name not like 'pg_%') a
67   $sql$
68  ;
69   end;
70   $body$;
```

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

```
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]

Compare constraints

```
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...

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('airlin
es','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('het
tie');
```

	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

And we can generate patches!

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

Constraints patches

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)

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	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

```
select * from diff.db_privs_select ('airlines')
```

	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
- Dealing with views and materialized views
- Compare functions and procedures
- Finalize all patches generation

Most importantly: some tools like this should be a part of PostgreSQL, nobody should reinvent the wheel.

Other issues

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

Q&A