Multi-tenant database systems : the good, the bad, the ugly

Who are you listening to?

- Pierre Ducroquet (French, sorry for your ears)
- Developer turned PostgreSQL DBA since about 10 years now
  - Working at Entr’ouvert, small french (mostly-)SaaS company

- Sometimes I wish I had not seen things

First, let’s get your interested

- How many tables do you have on your production server?
  - I said server, no plural
- How many indexes?

```
239 579 tables

965 942 indexes
```

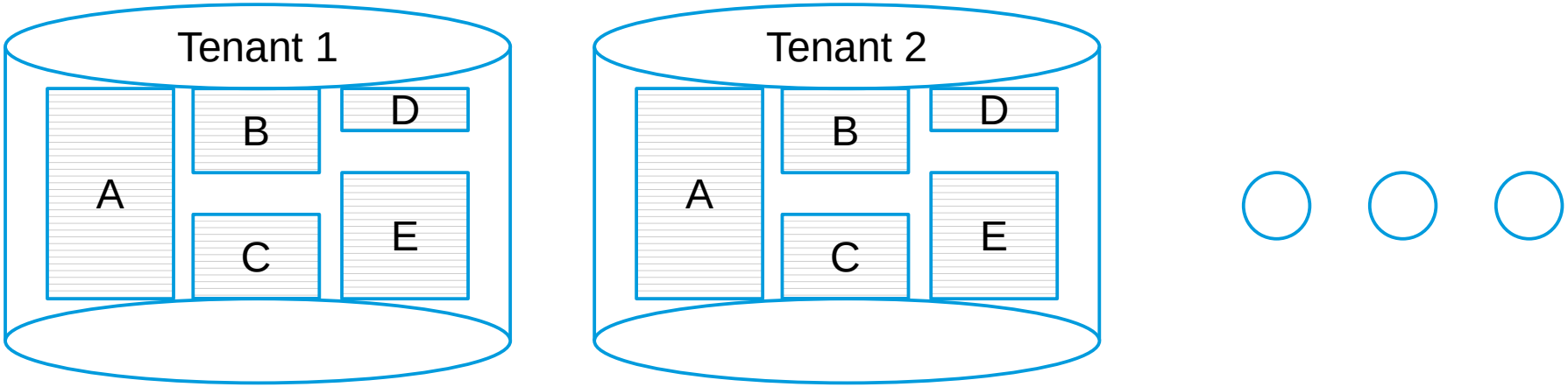
What is a multi-tenant system

- Wikipedia says:

  Software multitenancy is a software architecture in which a single instance of software runs on a server and serves multiple tenants. A tenant is a group of users who share a common access with specific privileges to the software instance.

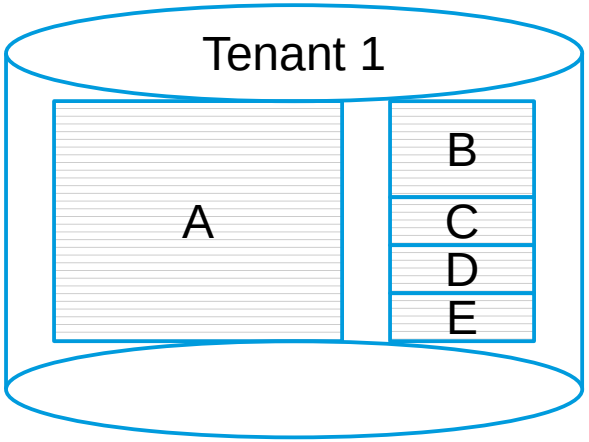
What is a multi-tenant system

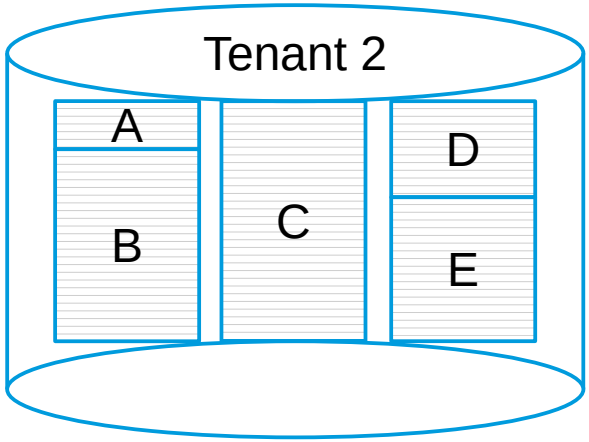
- Theory

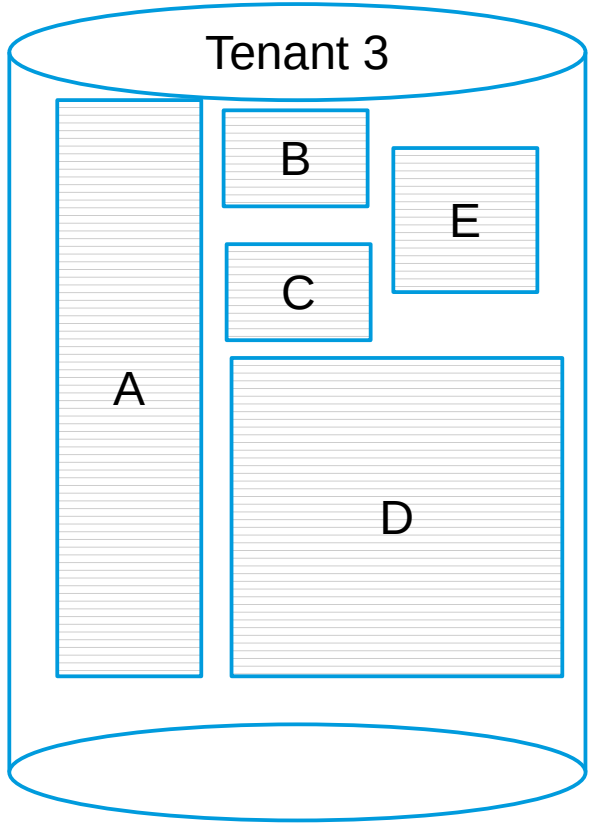


What is a multi-tenant system

- Theory vs reality…







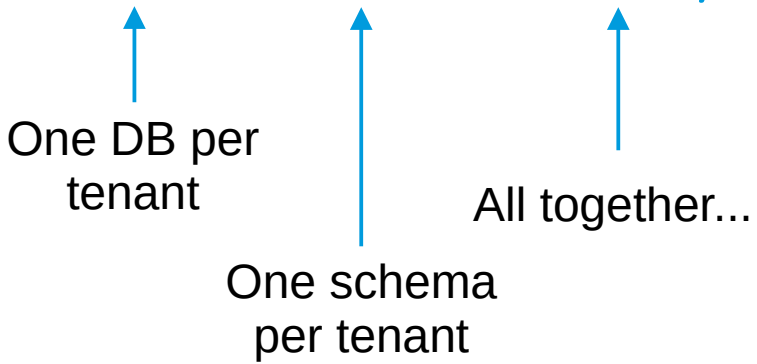
How to implement it in the database?

- Let’s look at a simple query.

```
TABLE tbl1;
```

- Let’s switch to a fully qualified name…

```
TABLE db1.schema1.tbl1;
```



How to implement it in the database?

- PostgreSQL extensions like Citus
- Spread a tenant_id column everywhere
- One database per tenant
- One schema per tenant

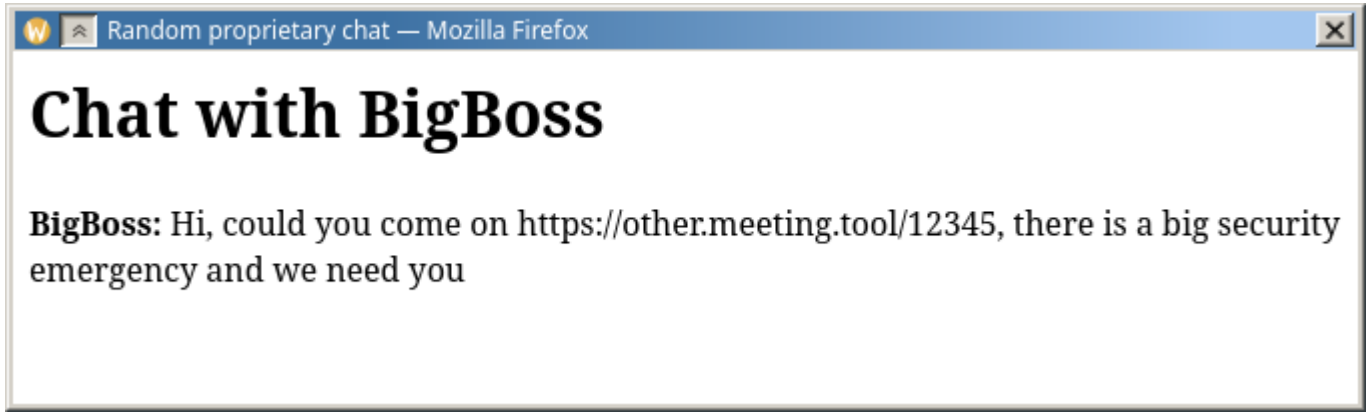
PostgreSQL extensions

- I only saw the Citus extension
- Never used it myself, so I can’t say much here that would be relevant
- I quote them, « Must design application for Citus »
- Most systems I saw evolved into multi-tenant later, sorry Citus…

tenant_id column

- Security (do you trust your developers?)
- Statistics and optimizer (correlations, correlations everywhere)
- Indexing is trickier (Should I add tenant_id in each index? Should I add tenant-specific indexes?)
- You ~~need~~ want PoWA
- Why is that query slow occasionnaly only ?
- Most tools work fine

And now for something completely different



Row Level Security for tenants ?

- Perfect exemple of a false good idea
- It works, but it’s not designed for this
- RLS aims for high, strict security standards, not your usual environment
  - You’ll have to tag everything as leakproof...

One database per tenant

- Security breachs must happen in the application before the DB starts being used, much harder
- No indexing drama (kind of)
- Tools will work fine (sort of)
- Connection pooling will suffer (a lot)
- Forget about pg_stat_statements (really)
- Open question : can you use logical replication ?
  - Once I have PostgreSQL 16, I'll tell you what happens...

One schema per tenant

- Security breachs are possible, but quite hard to do without a full SQL injection
- No indexing drama (kind of)
- Connection pooling will kind of work (sort of)
- Tools will endure pain (a lot)
- Forget about pg_stat_statements (don’t even bother installing it)

Tools, tools, what tools?

- pg_dump/pg_restore
  - -j : « Run the most time consuming steps concurrently »
    - Well… no
  - Toc.dat is going to be huge and its parsing is not optimal
    - Patchs pending, sorry for my lack of time

Tools, tools, what tools?

- Backups
  - You use pgbackrest, right ?
  - Enable repo bundle, the best new feature it had in the past years !
  - Being able to restore a single schema would be great
  - What is painful for backup tools ?
    - On-disc layout is going to get tricky for some tools
    - Many many many small files must be merged to reduce network IO

```
root@db2:/var/lib/postgresql/13/publik/base# ls -alS | head -n 5
total 75896
drwx------   2 postgres postgres 12193792 13 mars  12:03 54814802
drwx------   2 postgres postgres 10149888 13 mars  09:15 54774772
drwx------   2 postgres postgres  9277440 12 mars  14:19 55869318
drwx------   2 postgres postgres  2158592 12 mars  08:06 54753829
root@db2:/var/lib/postgresql/13/publik/base# find 54814802 -type f | wc -l
399279
```

Tools, tools, what tools?

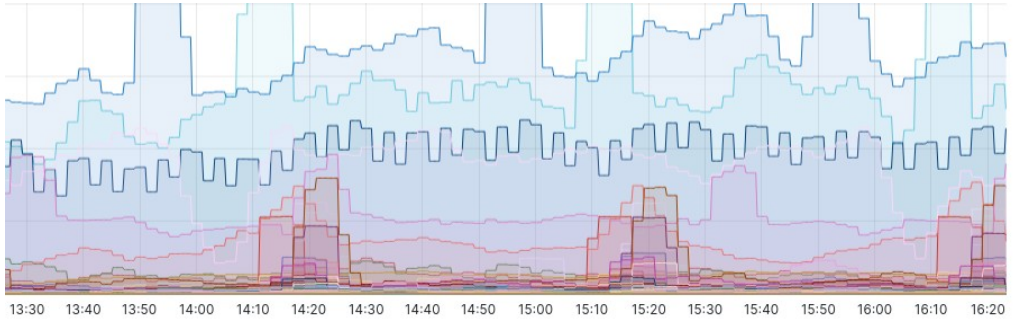
- pg_stat_statements
  - One database per tenant ?
    - You will need a huge value for pg_stat_statements.max
  - One schema per tenant ?
    - You will not see what happens per tenant
    - Normalization is broken here.

Tools, tools, what tools?

- Connection pooler
  - You often need one for performance
  - Most in-app poolers won’t optimize multi-tenants
  - One schema per tenant is tedious
    - Pgbouncer + one PostgreSQL extension to track search_path
  - One database per tenant requires huge configuration
    - Or pgbouncer has an automatic pool option, check it out...

Tools, tools, what tools?

- Monitoring tools…
  - Forget about graphs
  - No tool (to my knowledge) will split your database schemas



Rabbit holes, rabbit holes everywhere !

- There is no perfect solution.
- I think a mix of schema per tenant and database per tenant works quite well.
- If you can, measure and think it through before committing to any solution.
- And put on a happy face, our job would be boring without these.
  - Or worse, you could be working with Some Cloud SQL solutions…

Thanks !

Questions ?

Extra…

- Going too far, demonstration
  - Writing a PostgreSQL extension to work around impossible statistics
  - You know about optimizer « hints » ?
  - OFFSET 0 is one of them…