Branching in PostgreSQL

Noémi Ványi (@kvch)

Prague PostgreSQL Meetup: November Edition

Content review

01 Branch definition

02 Isolation options

03 Schema level isolation

04 Shared cluster

05 Takeaways, questions

A branch is an isolated database environment

to test schema changes safely

Database environment

- Cluster level objects
  - Roles
  - Extensions
- Database level objects
  - Schemas
- Schema level objects
  - Tables
  - Sequences
  - …

Isolation options

How can we isolate branches?

OPTION 1

SCHEMA LEVEL

A branch is a collection of database schemas. All users live in the same PostgreSQL database on the same PostgreSQL cluster.

OPTION 2

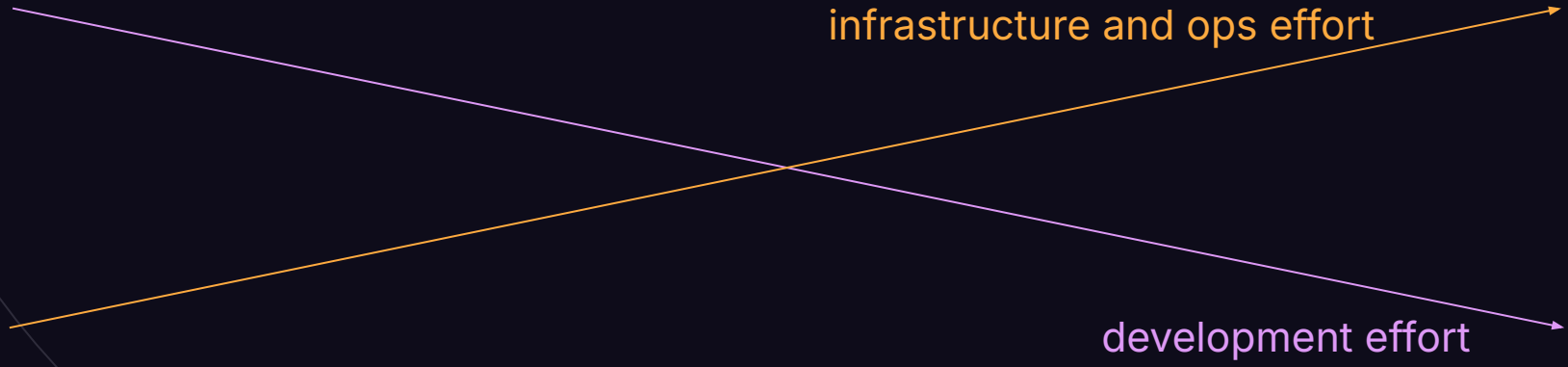
DATABASE LEVEL

A branch is a PostgreSQL database. Multiple users can live on the same PostgreSQL cluster.

OPTION 3

CLUSTER LEVEL

A branch is a PostgreSQL cluster.



Schema level isolation: Lightweight branches

Why opt for schema level isolation?

- Existing user branches were already in their own schemas
- No need to change our architecture significantly
- We only wanted to support update, insert, delete, truncate statements (or so we thought)
- Everything else was done by Xata REST API

Shared clusters

Our shared PostgreSQL clusters

SQL over HTTP

SQL proxy over HTTP

- In the connection pool we used a common role
- When a user connected, we set the role their branch owner role
- We had to control what statements users can run
    - SET and RESET ROLE is strictly forbidden
- We forbid several functions that could leak the internals
- Limited experience because we did not support transactions or anything session level

Tasks of the SQL proxy

- User submits their SQL statement using the REST endpoint
- SQL proxy parses the statement and decides if it is allowed
- If the statement has a syntax error or is forbidden, proxy returns an error
- If the statement is allowed, the proxy runs the statement in PostgreSQL
- It parses the result and returns it to the user in JSON or in array format

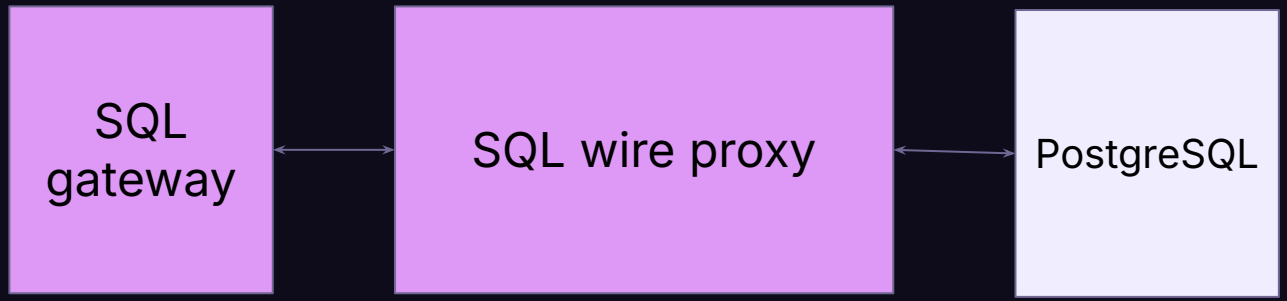
It’s all fun and games until someone wants to support wire protocol access

Wire protocol access use cases

- psql support is required
  - Create and manage tables, schemas
  - Export and import databases (pg_dump and pg_restore)
  - Stop long running queries

- ORM support
  - Several users interact with Xata using Drizzle

SQL wire proxy



Hiding database objects from other users

- When you list the schemas in a database, you can see the list of all schemas including the ones that you do not own or have no access to
- PostgreSQL lets you lookup object names based on OID
- With functions regclassout, regclass and to_regclass you can inspect objects in the database

Shadow catalog

- New schema named xata_catalog (invisible to users)
- Store all overwritten pg_catalog functions we need
- When SQL proxy detects a call to a catalog function, it checks if it has to be rewritten to our catalog function. If yes, the function in our shadow catalog is called.
- Otherwise the function is passed through as is

Enforce usage quotas on shared clusters

- Limit the number of concurrent connections to PostgreSQL in the gateway (same as HTTP proxy)
- With long running sessions, we decided to limit the session time length
- SQL parser found session timeout settings and if a user tried to set higher than the allowed time, we returned an error and asked the user to decrease the timeout

Multiple schema support in a branch

Xata branch = PostgreSQL schema
Schema name in PostgreSQL is branch ID
Example: bb_whateverid

Multiple schema support in a branch

Xata branch = set of PostgreSQL schemas
Schema name in PostgreSQL is branch ID + schema name suffix
Example: bb_whateverid = public schema
bb_whateverid_dev = dev schema

Multiple schema support in a branch

- Whenever a user interacts with their database objects in a schema, we need to rewrite the schema name from the virtual (user defined) to the physical schema name
- We rewrite all statements where a schema name is present

Multiple schema support in a branch

- We had to rewrite the outgoing names as well
- When pg_dump exports the database, we must return virtual schema names
- We overwrote two PostgreSQL functions _public_nspname and _public_nspname_sql_identifier
- If the schema name started with the branch prefix stored in xata.physical_schema_prefix or with bb_ prefix we stripped the branch ID or returned public
- Rewrite output of more functions: format_type, pg_get_expr, etc.

Multiple schema support in a branch

- When user sets search_path, set virtual search path in xata.search_path and store physical search path in search_path
- Read virtual search path from xata.search_path path and return it
- Override several function to return correct schema names like current_schema or current_schemas

Statements we could not support

- DO: limited support due to schema name rewriting
- Functions: limited support, requires plpgsql parsing and rewriting
- RESET ALL
- DISCARD ALL

Takeaways