Postgres schema migrations using the expand/contract pattern

PGDay UK, September 9th 2025

Speaker

Andrew Farries
Staff Software Engineer @ Xata
andrew.farries@xata.io

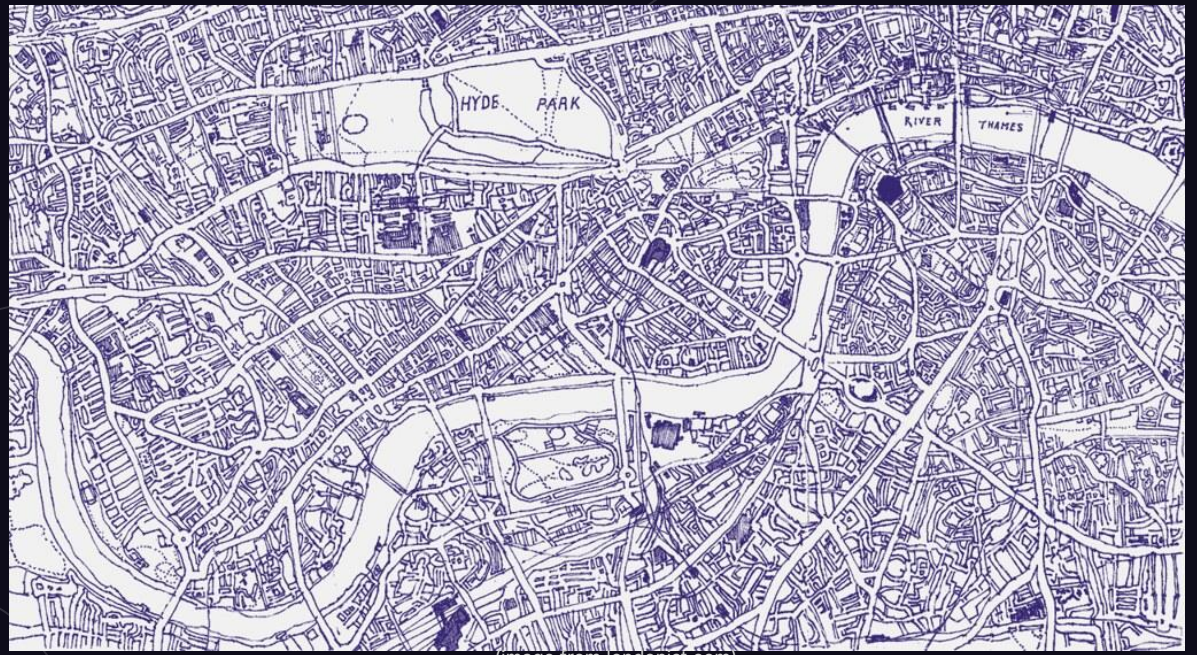
Outline

01 Schema change is hard

02 Common pitfalls

03 Expand contract migrations

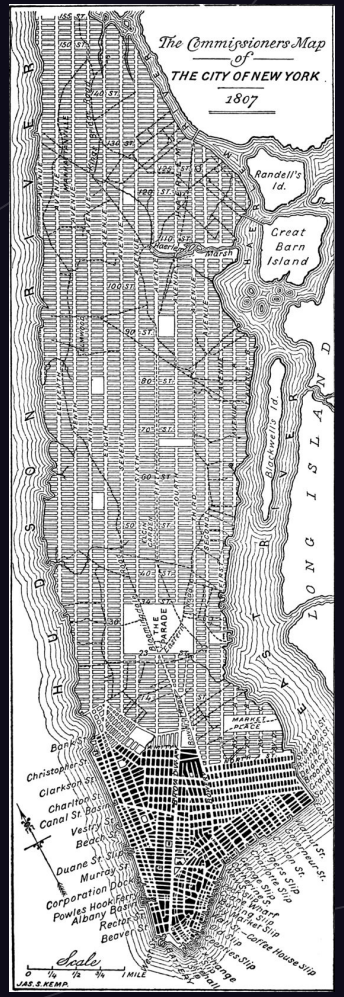
04 Expand/contract with pgroll



(image from londonist.com)

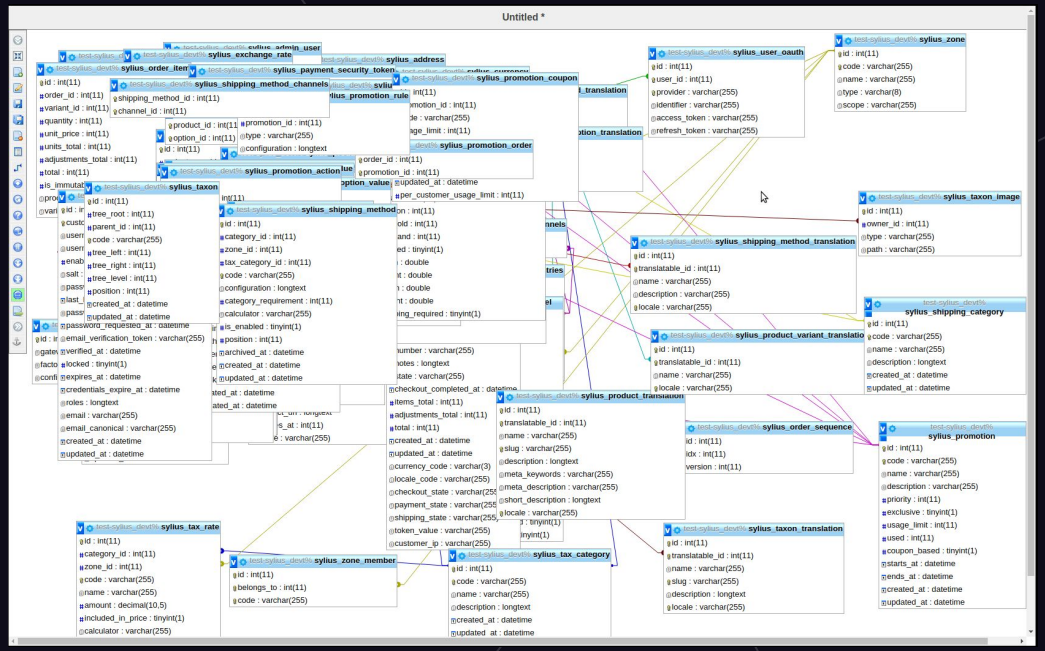
xata


HOT DOGES

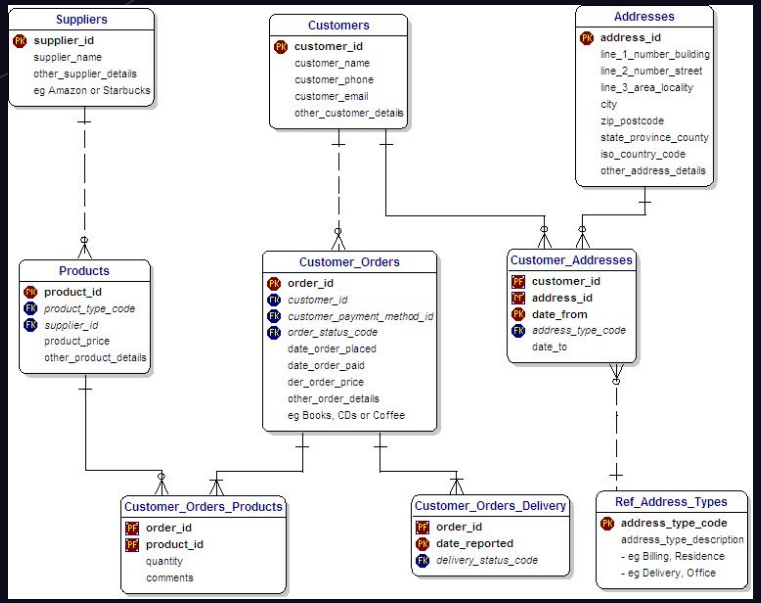

The Commissioners Map
of
THE CITY OF NEW YORK
1807
Randell's Id.
Great Barn Island
HUDSON RIVER
EAST RIVER
LONG ISLAND
Bank St.
Christopher St.
Clarkson St.
Charlton St.
Canal St.
Vestry St.
Beach St.
Duane St.
Murtey St.
Corporation Dock
Powles Hook Ferry
Albany Basin
Rector St.
Beaver St.
Cortes House Slip
Scale
1 MILE
JAS. S. KEMP.

xata

London



Manhattan



xata

"Database schemas are notoriously volatile, extremely concrete, and highly depended on. This is one reason why the interface between OO applications and databases is so difficult to manage, and why schema updates are generally painful."

Robert C. Martin, Clean Architecture

Common pitfalls

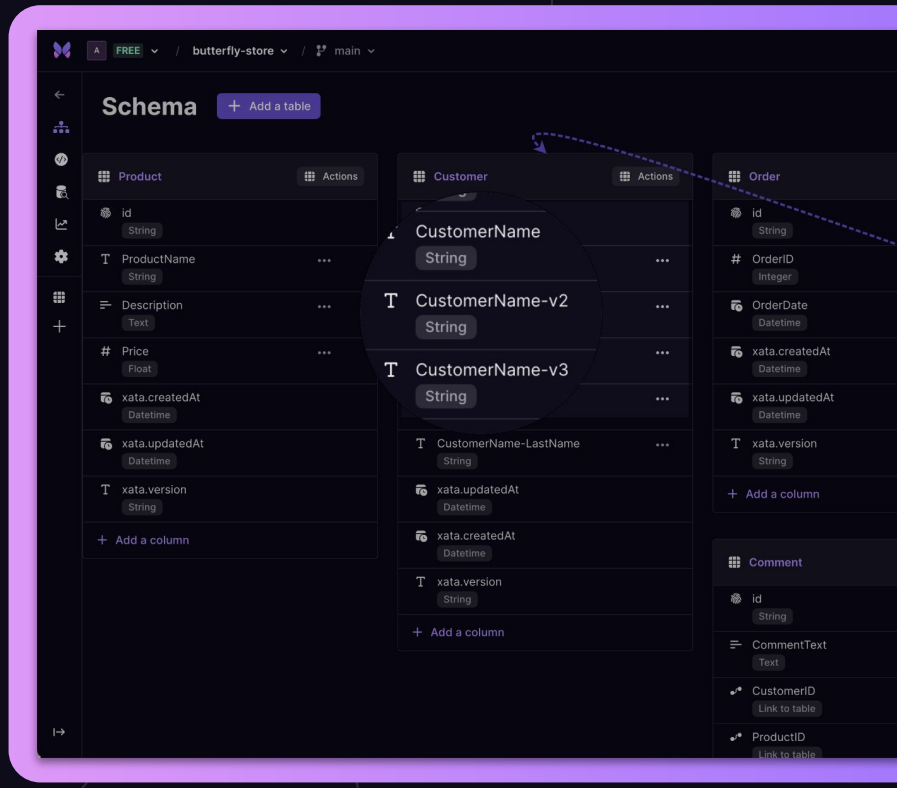
Additive-only changes and schema debt

What is it?

The act of never modifying or removing columns, only adding new ones

Why is this bad?

- Bugs and performance implications
- General confusion
- Long-lived compatibility code



xata

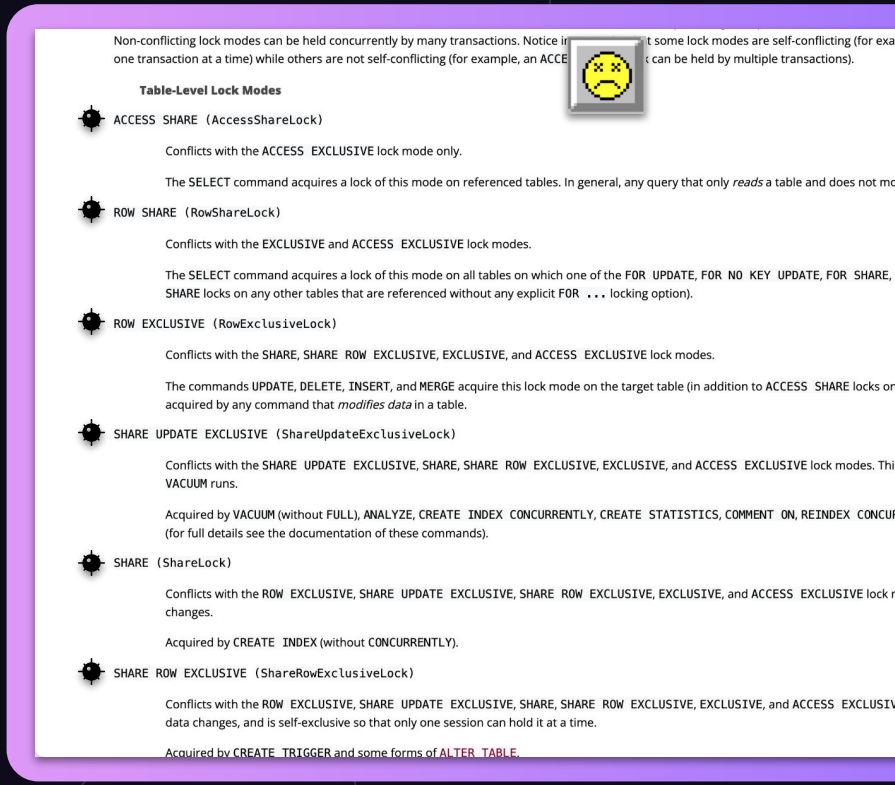
The locking minefield

What is it?

PostgreSQL offers good ways to control locking your database, but you need to know what you're doing

Why is this bad?

- Tables are inaccessible
- Query queuing
- Testing is hard



Testing schema migrations

What is it?

Confidence that a migration will succeed in production is hard to obtain with limited data available in lower environments.

Why is this bad?

- Failures only detected in production
- Problems provisioning realistic data-sets
- Lack of confidence in migrations

Rolling back your changes

What is it?

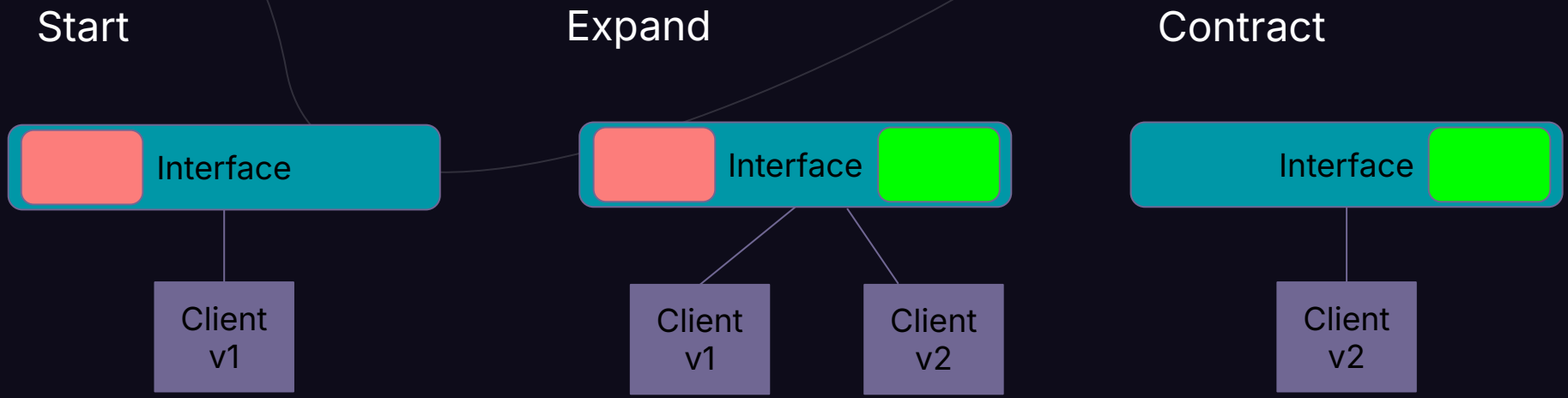
We’re all human; mistakes occur. Having to roll back the changes you made in production can be painful

Why is this bad?

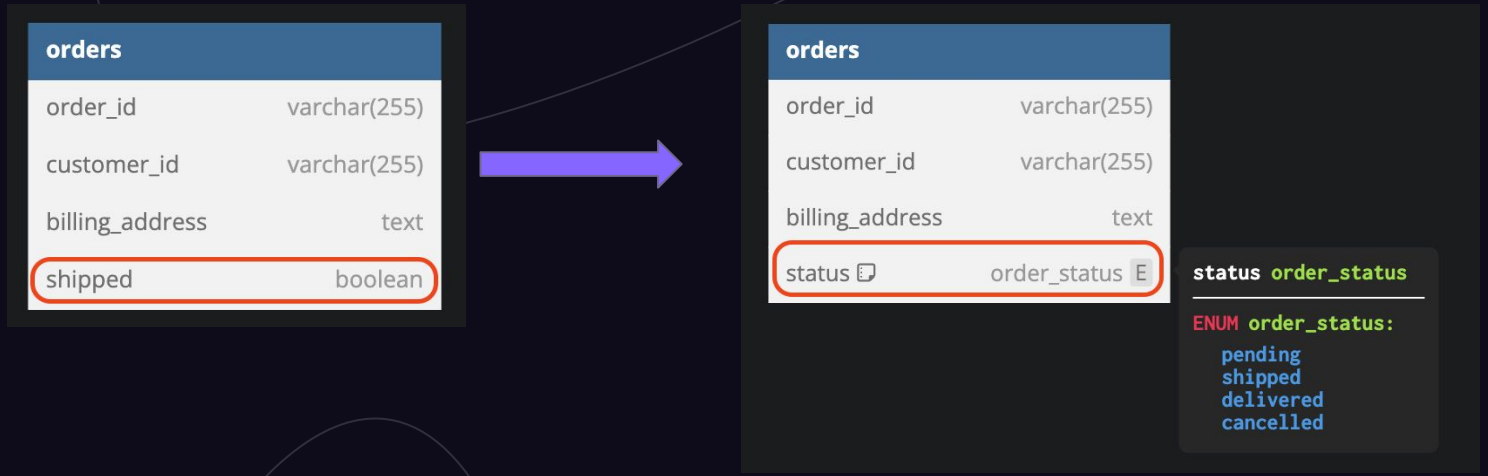
- Unplanned maintenance
- Often untested
- Time consuming

The expand / contract pattern

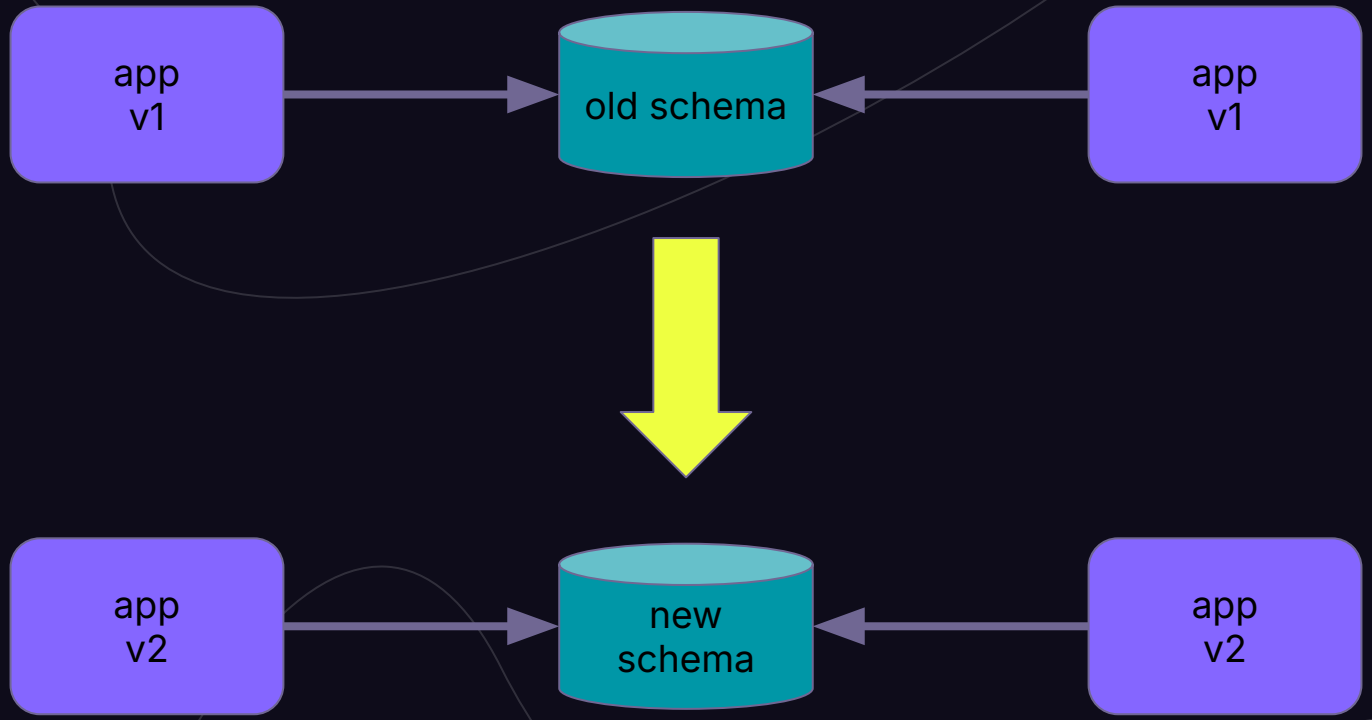
Expand & contract



Expand & contract

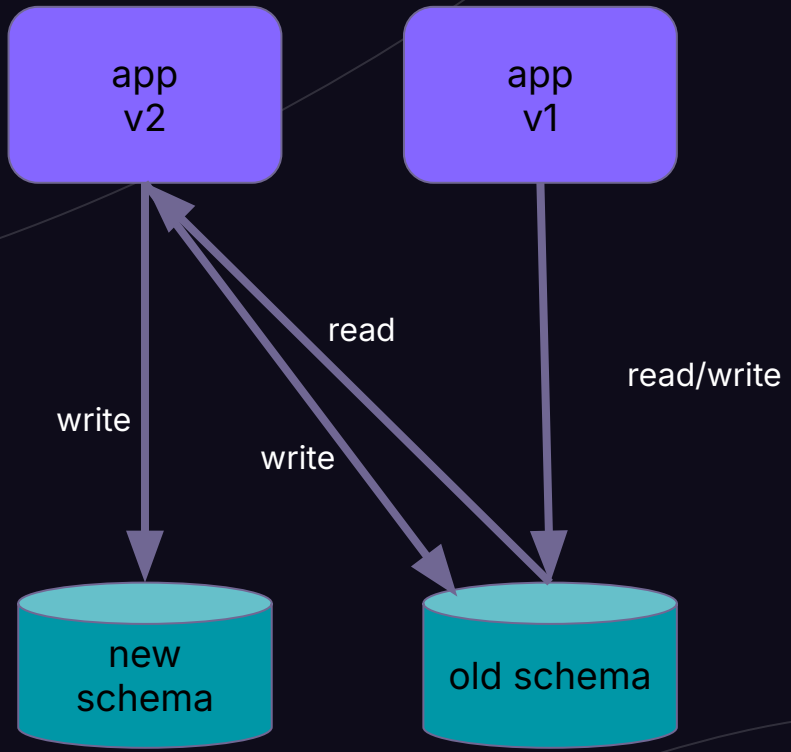


Big bang migration



expand/contract - dual write

id	customer_id	billing_address	shipped	status
3	5678	456 Somewhere Lane	False	<null>
4	1234	123 Somewhere Street	True	<null>

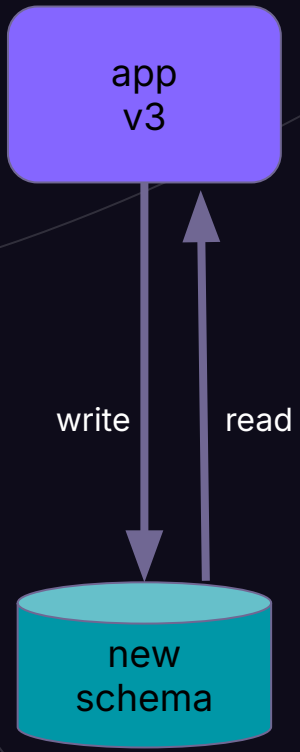


expand/contract - migrate

- Wait for the rollout of v2 to complete
- Run a data migration to backfill the `status` field

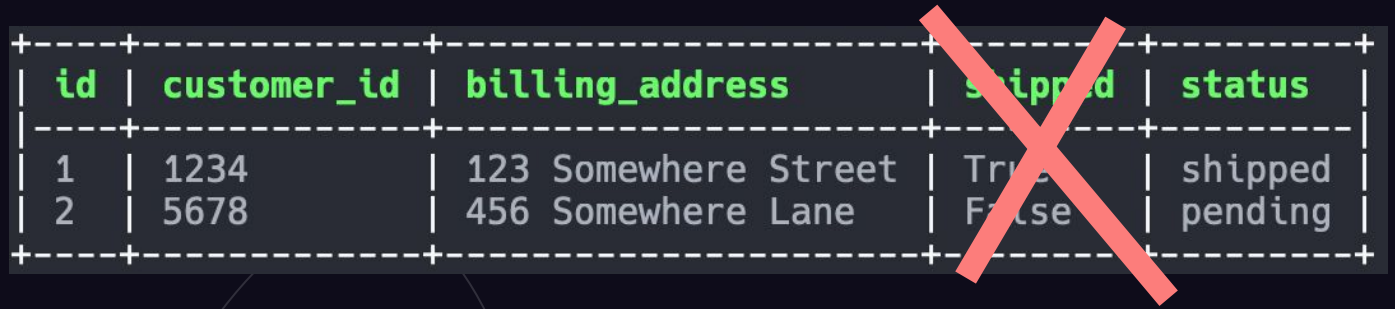
id	customer_id	billing_address	shipped	status
1	1234	123 Somewhere Street	True	shipped
2	5678	456 Somewhere Lane	False	pending

expand/contract - read new



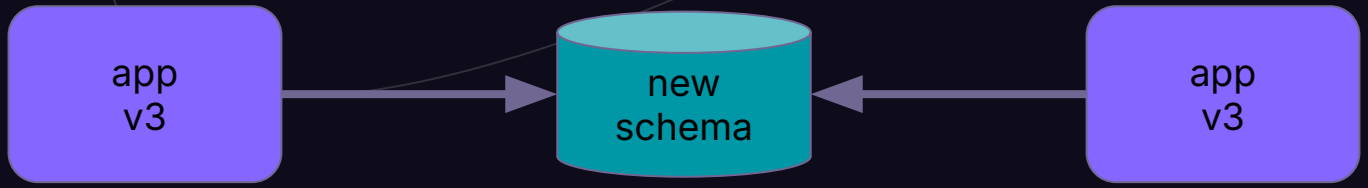
expand/contract - contract

- Once the rollout of v3 is complete, drop the `shipped` field
- The migration is complete



id	customer_id	billing_address	shipped	status
1	1234	123 Somewhere Street	True	shipped
2	5678	456 Somewhere Lane	False	pending

Expand / contract - complete



3 migrations required:

- Add the new field
- Backfill the new field
- Remove the old field

2 new application versions:

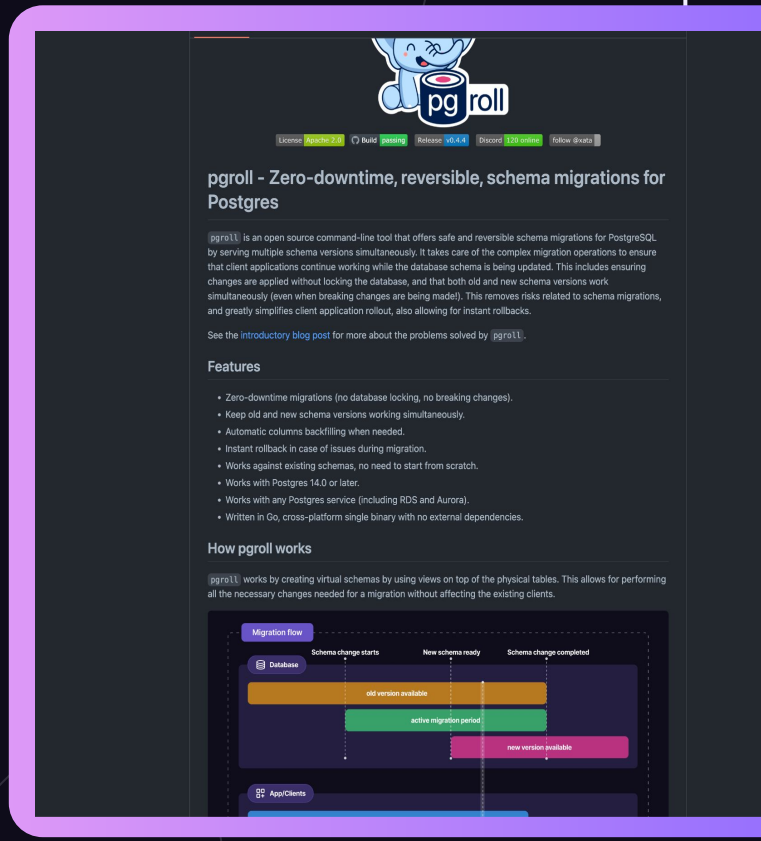
- Dual write
- Final version

Zero-downtime, reversible, schema migrations for Postgres

xata

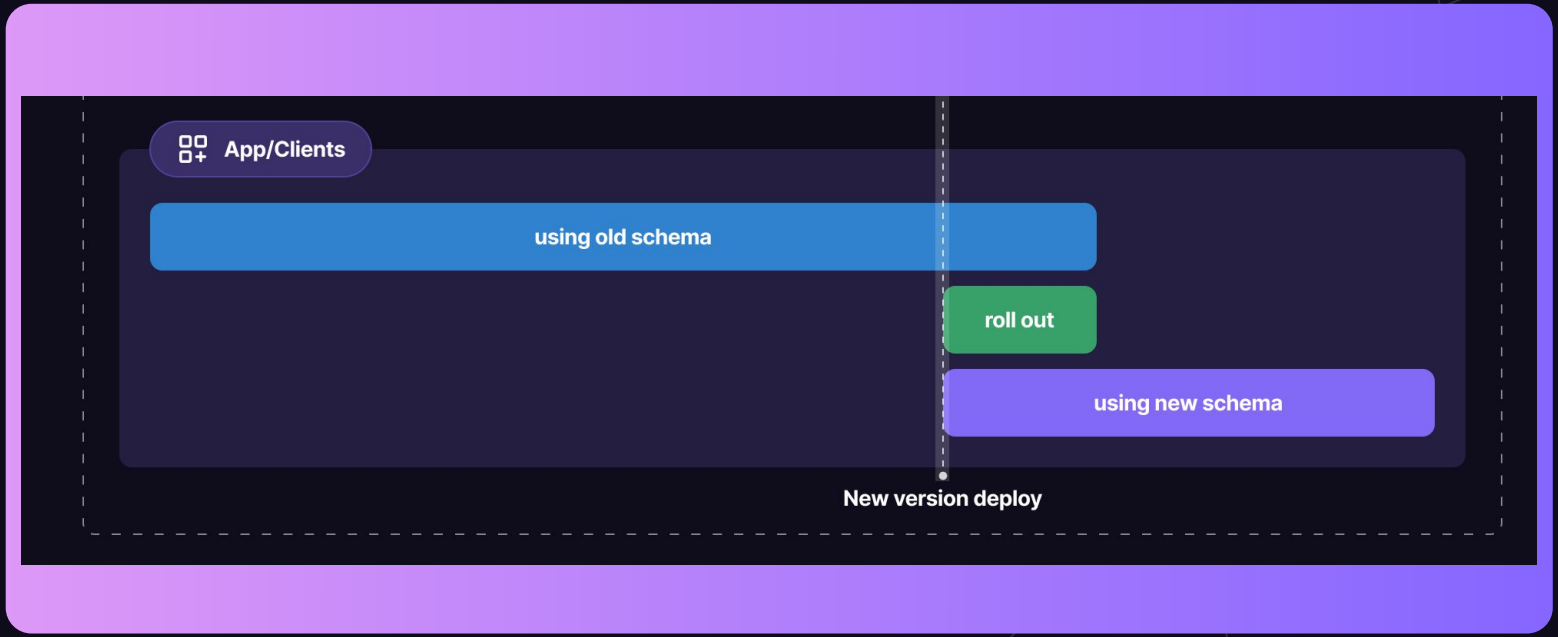
pgroll - design goals

- Build around the expand/contract pattern
- Keep migration logic out of the application layer
- Easy rollbacks
- No nasty surprises around locking behaviour
- Postgres only
- Open source



xata

Application rollouts



Demo

Lesson learned

- Expand contract is a powerful technique for schema change
- Migration tools should operate at a higher level than raw SQL
- Migrations are long-lived processes and migration tools should manage them end to end
- Data migrations should be handled by migration tools, not at the application level

Thank you!

 @xata

 xataio/pgroll

 @xata.io

 xata.io/discord