Anonymizing data in Postgres without losing your mind (or your schema)

Ahmet Gedemenli & Esther Minano

October 2025

xata

Content Review

1. The real data problem
2. Why anonymizing is tricky
3. What you can do in Postgres
4. The tool landscape
5. How pgstream works

The real data problem

Why copying production databases creates headaches

The Dev's Dilemma

You need a prod clone for debugging
but oops - now you've got customer PII sitting on your laptop.

What we all want: Real data for reproduction of bugs & realistic testing

But risk: Actual personal data & compliance violations

The Trade-Off: Convenience vs. Safety

Developer Needs	Risk / Concern
Accurate, representative data	Exposure of PII / secrets
Easy debugging & tracing	Non-compliance with laws
Reproducible production-like behavior	Leaks in dev / staging environments

Why anonymizing is tricky

Keeping schemas intact while making data safe and useful

Core Challenges

Schema complexity: Foreign keys, constraints, JSON fields

Multiple Anonymization Goals: Redaction, pseudonymization, masking

Performance & maintainability: Easy to evolve as schema changes

Legal & compliance pressure: GDPR / CCPA require care even in non-prod

What you can do in Postgres

Simple SQL, views, and JSON tricks you can use today

The simplest: Quick & dirty overwrites 1/2

```
UPDATE users
SET email = md5(email) || '@example.com',
    name = 'REDACTED',
    phone = NULL;
```

- Very easy to do
- Safe (if fully covers sensitive columns)
- Loses data variety & relationships
- Doesn't help JSON / nested fields

The simplest: Quick & dirty overwrites 2/2

```
UPDATE users
SET email = CASE
              WHEN is_vip OR email LIKE '%xata.io' THEN email
              ELSE md5(email) || '@example.com'
            END;
```

- VIP or internal accounts may retain real values
- Can be tuned by flags, environment, user roles
- Be careful about exceptions

Handling JSON Fields

```
UPDATE events
SET data = jsonb_set(data, '{ip}', '"REDACTED"');
```

- You can surgically alter JSON fields
- Remove or replace keys
- Need to handle nested objects, arrays
- Always test for missing keys

Views to the Rescue

```
CREATE [MATERIALIZED] VIEW safe_users AS
 SELECT id,
        md5(email) || '@example.com' AS email,
        'REDACTED' AS name
 FROM users;
```

- Devs query safe_users instead of users - no underlying PII exposed
- Row level security & permissions
- Schema changes might break this - use carefully
- Performance vs. Freshness

The tool landscape

Extensions, CLIs, and services for data masking

xata

PostgreSQL Anonymizer (anon)

- Runs inside Postgres as an extension
- Masking rules via SECURITY LABEL
- Supports dynamic and static masking
- Anonymized dumps

Simple and tightly integrated

Overhead to queries

No replication support

Greenmask

- OSS External CLI tool
- Rules configured via YAML file
- Dump → anonymize → restore


Good for batch jobs


Additional infra required


No replication support

pgstream

- OSS External CLI tool (snapshot)
- External long running service (replication)
- Rules configured via YAML file
- DDL tracking
- Multiple target support


Good for batch jobs


Replication support


Additional infra required

xata

Anonymization tools at a glance

Feature / Tool	PostgreSQL Anonymizer	Greenmask	pgstream
Runs inside Postgres	✓	✗	✗
External CLI (one-off)	✗	✓	✓
External service (continuous)	✗	✗	✓
Batch anonymization (dump/ restore)	✓ (static dump)	✓	✓ (snapshot)
Real-time / replication	✗	✗	✓
Column-level transformations	✓	✓	✓
Status	Active	Active	Active

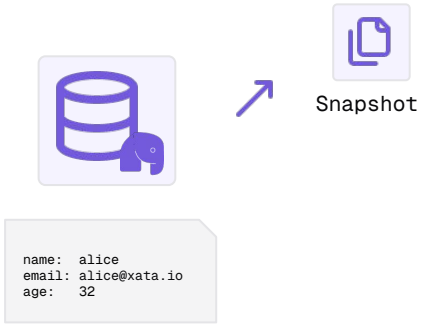
How pgstream works

Making snapshots and replication safer with transformations

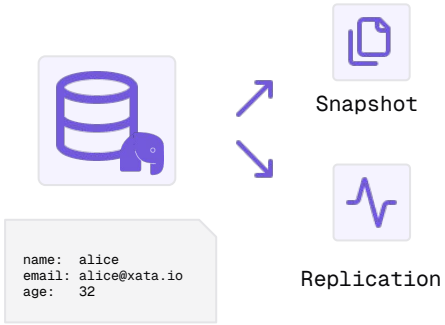
Anonymizing data with pgstream 1/8

```
name:  alice
email: alice@xata.io
age:   32
```

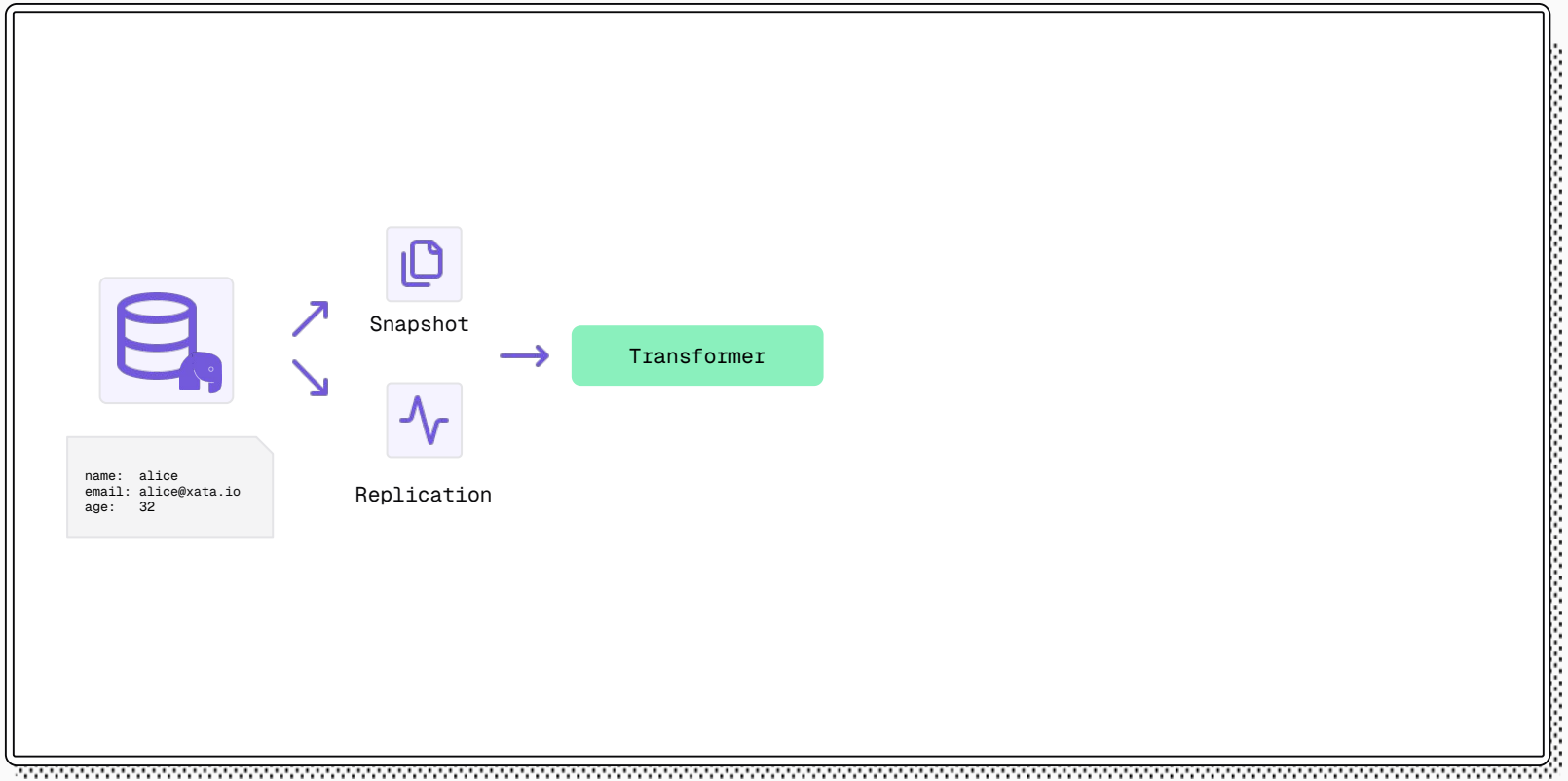
Anonymizing data with pgstream 2/8



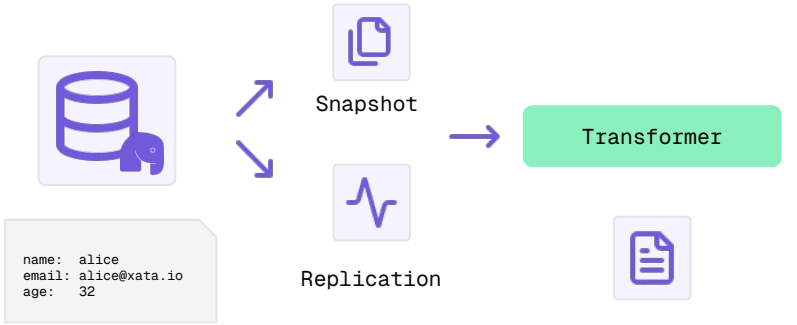
Anonymizing data with pgstream 3/8



Anonymizing data with pgstream 4/8

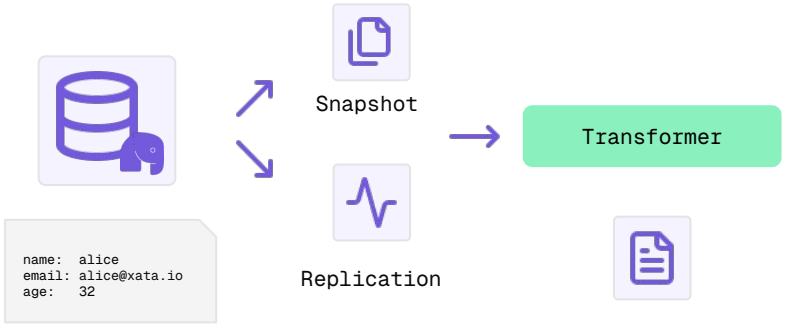


Transformation rules 1/6



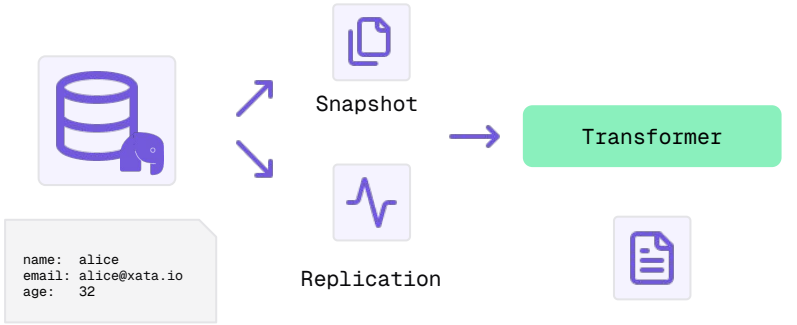

```
transformations:
  infer_from_security_labels: false
  dump_inferred_rules: false
  validation_mode: relaxed
  table_transformers:
    - schema: public
      table: test
      column_transformers:
        name:
          name: pg_anonymizer
          parameters:
            anon_function: anon.pseudo_first_name
            salt: mysalt
        email:
          name: masking
          parameters:
            type: email
        age:
          name: greenmask_integer
          parameters:
            min_value: 18
            max_value: 65
```

Transformation rules 2/6



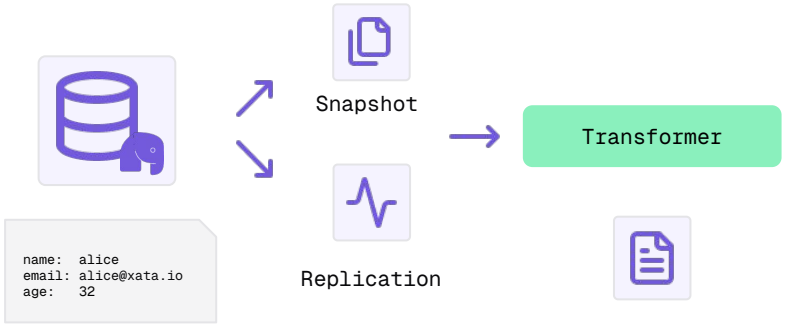

```
transformations:
  infer_from_security_labels: false
  dump_inferred_rules: false
  validation_mode: relaxed
  table_transformers:
    - schema: public
      table: test
      column_transformers:
        name:
          name: pg_anonymizer
          parameters:
            anon_function: anon.pseudo_first_name
            salt: mysalt
        email:
          name: masking
          parameters:
            type: email
        age:
          name: greenmask_integer
          parameters:
            min_value: 18
            max_value: 65
```

Transformation rules 3/6



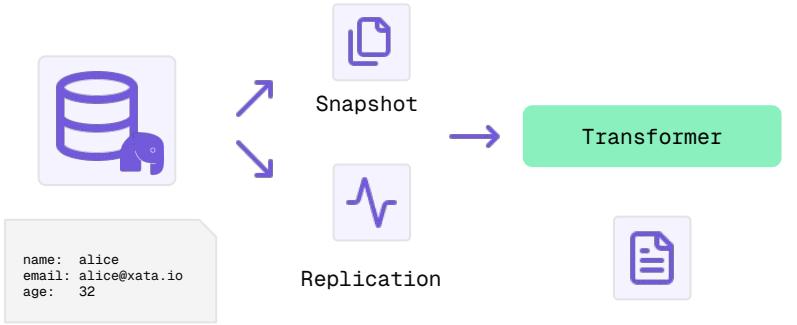

```
transformations:
  infer_from_security_labels: false
  dump_inferred_rules: false
  validation_mode: relaxed
  table_transformers:
    - schema: public
      table: test
      column_transformers:
        name:
          name: pg_anonymizer
          parameters:
            anon_function: anon.pseudo_first_name
            salt: mysalt
        email:
          name: masking
          parameters:
            type: email
        age:
          name: greenmask_integer
          parameters:
            min_value: 18
            max_value: 65
```

Transformation rules 4/6



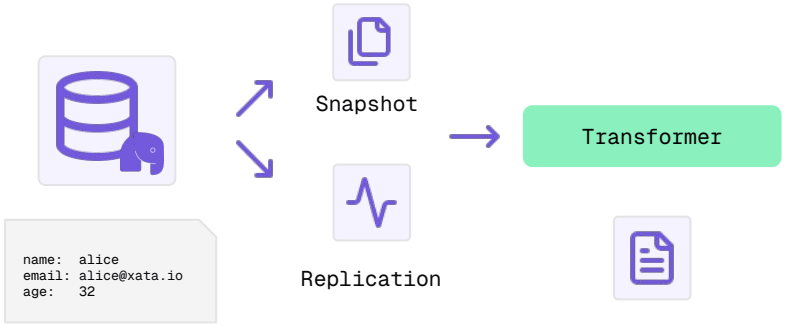

```
transformations:
  infer_from_security_labels: false
  dump_inferred_rules: false
  validation_mode: relaxed
  table_transformers:
    - schema: public
      table: test
      column_transformers:
        name:
          name: pg_anonymizer
          parameters:
            anon_function: anon.pseudo_first_name
            salt: mysalt
        email:
          name: masking
          parameters:
            type: email
        age:
          name: greenmask_integer
          parameters:
            min_value: 18
            max_value: 65
```

Transformation rules 5/6



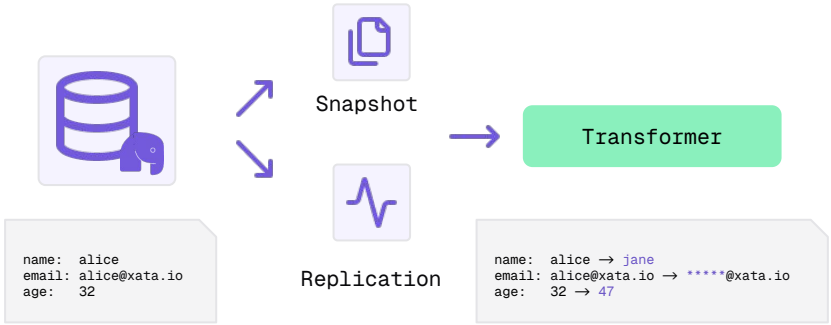

```
transformations:
  infer_from_security_labels: false # Postgres `anon` masking rules
  dump_inferred_rules: false
  validation_mode: relaxed
  table_transformers:
    - schema: public
      table: test
      column_transformers:
        name:
          name: pg_anonymizer
          parameters:
            anon_function: anon.pseudo_first_name
            salt: mysalt
        email:
          name: masking
          parameters:
            type: email
        age:
          name: greenmask_integer
          parameters:
            min_value: 18
            max_value: 65
```

Transformation rules 6/6

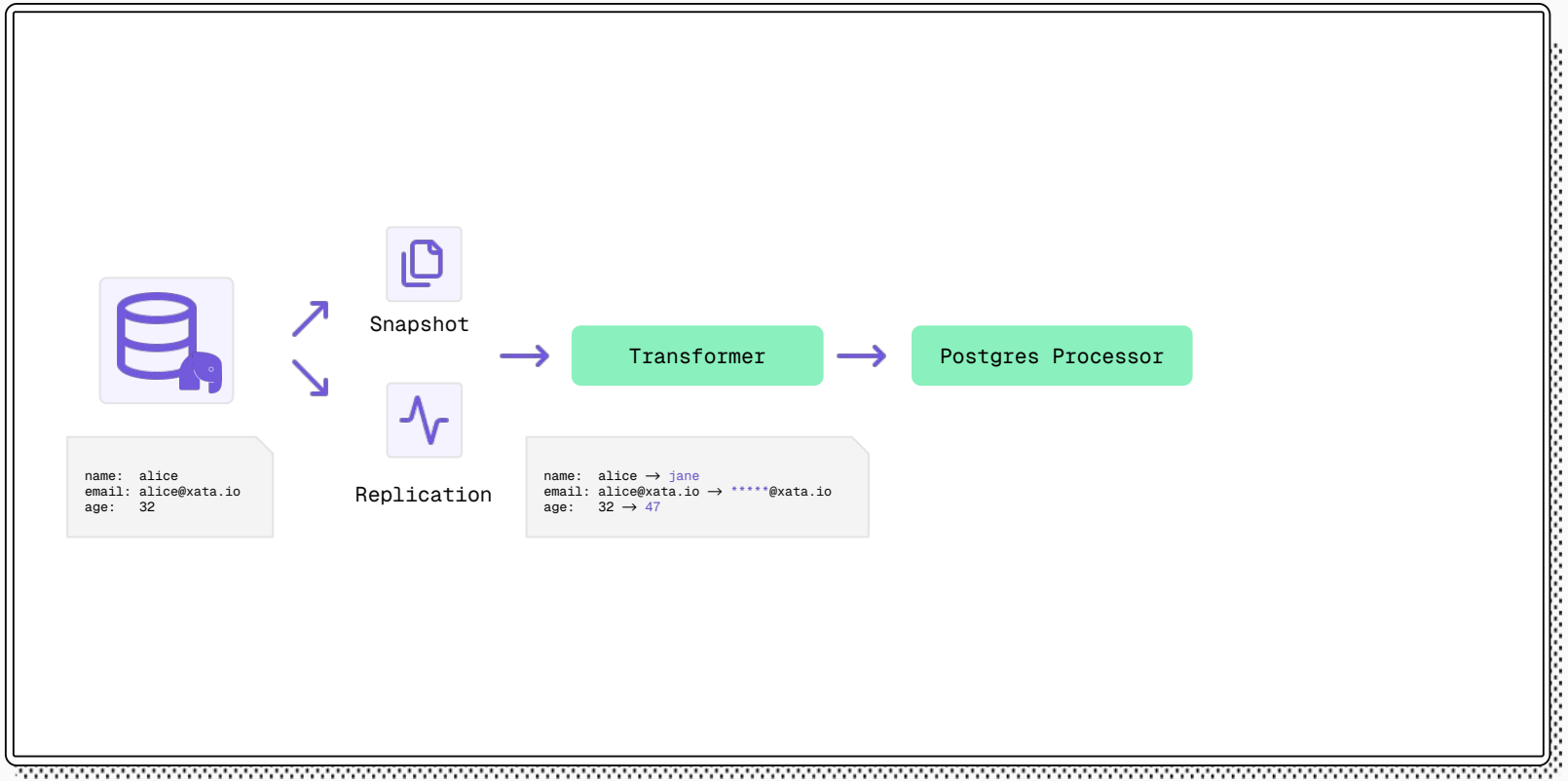



```
transformations:
  infer_from_security_labels: false
  dump_inferred_rules: false
  validation_mode: relaxed # one of relaxed, strict or table_level
  table_transformers:
    - schema: public
      table: test
      column_transformers:
        name:
          name: pg_anonymizer
          parameters:
            anon_function: anon.pseudo_first_name
            salt: mysalt
        email:
          name: masking
          parameters:
            type: email
        age:
          name: greenmask_integer
          parameters:
            min_value: 18
            max_value: 65
```

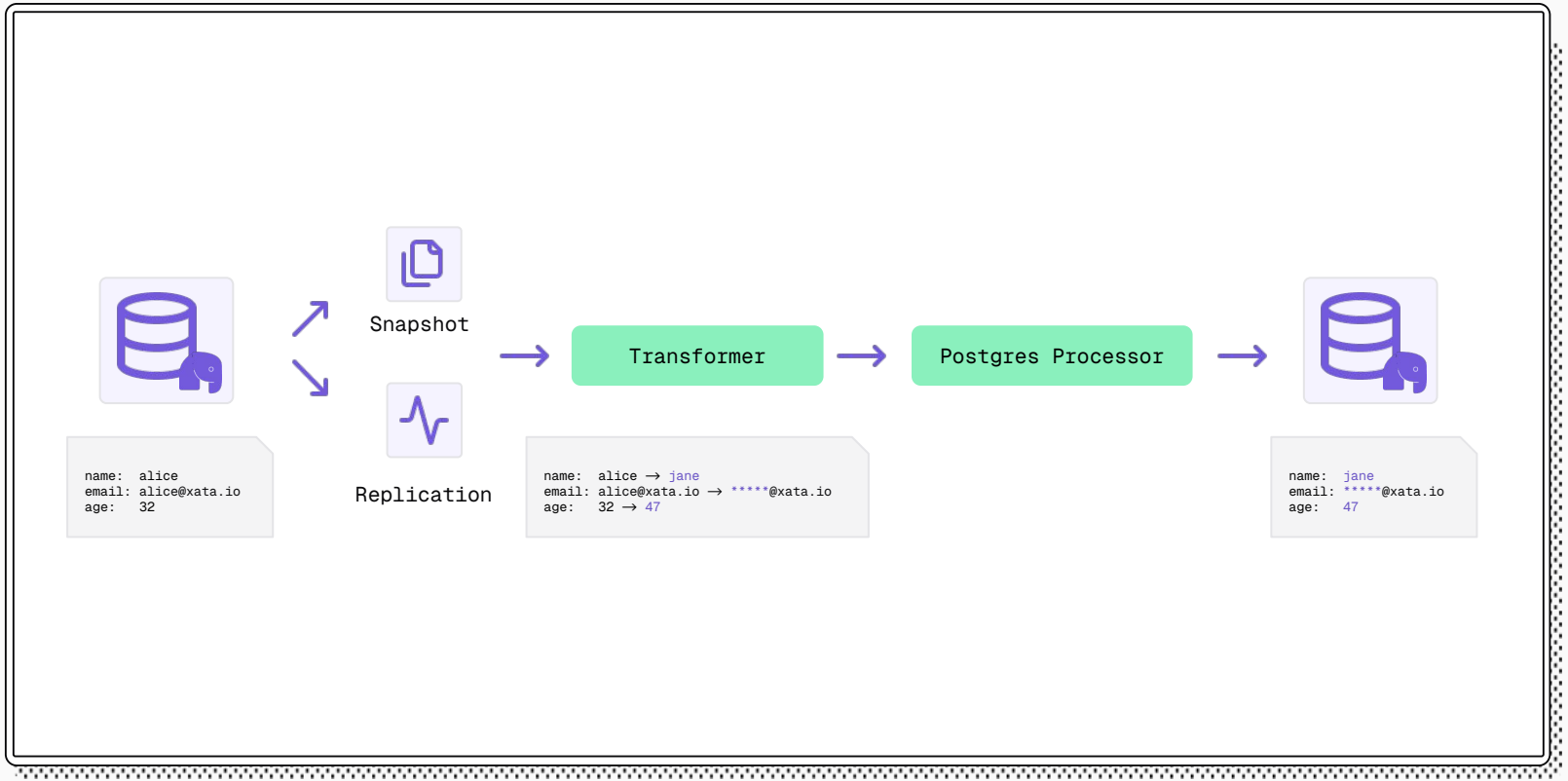
Anonymizing data with pgstream 6/8



Anonymizing data with pgstream 7/8



Anonymizing data with pgstream 8/8



xata

Strengths

- Snapshot and continuous replication
- Security label compatibility
- Extensive integration support
  - Greenmask transformers
  - Postgres anonymizer (anon) functions
- Rich custom transformers
  - json/jsonb type transformations
  - hstore type transformer
  - Templating support
  - Dynamic parameters

Templating and dynamic parameters example

```
{{ $email := "" }}
{{- if (ne .GetValue nil) -}}
  {{ $email = .GetValue }}
{{- else -}}
  {{ $email = .GetDynamicValue "secondary_email" }}
{{- end -}}
{{- if (not (contains "@xata" $email)) -}}
  {{ masking "email" $email }}
{{- else -}}
  {{ $email }}
{{- end -}}
```

pgstream in Xata

- Powering onboarding workflow
- Clone production databases into Xata
  - Custom anonymization
  - Periodic snapshots for small databases (GB)
  - Snapshot + replication for large databases (TB)

Takeaways

Simple rules for safer data

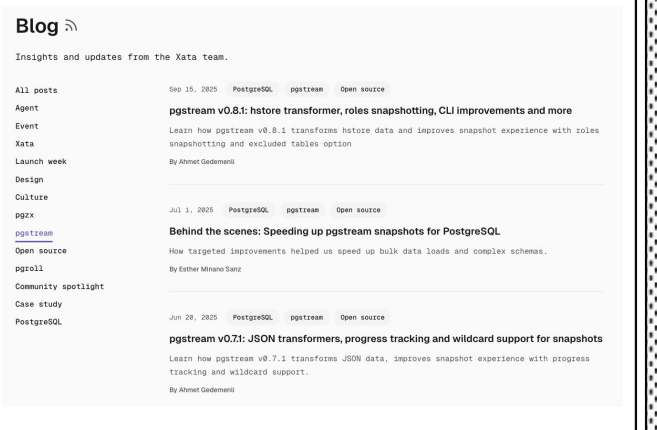
xata

Takeaways

- Plan for data to be copied and shared
- Build anonymization into your pipeline, not around it
- Pick tools that fit how you move data
- Keep transformations out of your schema

For more information

https://github.com/xataio/pgstream



https://xata.io/blog

A data platform for PostgreSQL

Thank you!

@xata