PostgreSQL Anonymizer and the battle for privacy

DALIBO
L'expertise PostgreSQL

Bonjour

- Damien Clochard
- Co-fondateur de DALIBO
- Vice-Président de l’association PostgreSQLFr
- Développeur Principal de PostgreSQL Anonymizer

8 ans déjà !

- Projet lancé par DALIBO en 2018
- 100 % open-source, pas de vendor lock-in
- Soutenu par ANR – DGFIP - EFLUID

Menu

- Le Paradoxe de la vie privée
- 6 principes de protection
- Implémenter ces principes avec PostgreSQL Anonymizer
- Nouveautés de la version 3.0

Le Paradoxe de la vie privée

Un concept flou

- Droit à l’intimité
- « tous les éléments non publics relatifs à une personne »

Un concept en évolution constante

Une intuition partagée

A mesure que nos vies sont de plus en plus numérisées

Notre besoin d’intimité grandit

Eric Schmidt n’est pas d’accord

> *If you have something that you don’t want anyone to know, maybe you shouldn’t be doing it in the first place.*

Eric Schmidt, Google CEO, 2009

Une bataille

- Les géants du numérique veulent redéfinir la notion même de “vie privée”
- Les usagers attendent plus de protection
- En tant que DBA, vous êtes en plein milieu de cette bataille

Le quotidien DBA postgres

6 principes fondamentaux

- « Privacy By Design »
- Réduction de la surface d’attaque (ASR)
- Séparation des roles
- Miminisation
- Evaluation des risques
- « Privacy By Default »

Privacy By Design

La politique de protection de la vie privée devrait être

écrite au moment même de la conception d’une application

Réduction de la surface d’attaque

Minimiser le risque de fuite de données

en réduisant la circulation et points de stockage

Séparation des roles

Généraliser l’usage de comptes nommés

et définir des règles de masquages adaptées à chacun

Minimisation

Le type et le volume d’infomation collectée doit être aux usages appropriés et nécessaires pour accomplir le but recherché

Evaluation du risque

Etudier fréquemment les risques d’attaque et de fuites de données :

origine, nature, sévérité, etc…

Privacy By Default

Si vous ne savez pas si une colonne contient des données privées

traitez la comme si elle contenait des données privées

PostgreSQL Anonymizer

Disponible partout

- Fonctionne avec toutes les versions majeures de Postgres
- Red Hat / Rocky / SUSE / Debian / Ubuntu
- Docker / Ansible / CNPG
- Google Cloud / Azure / IBM Cloud / Alibaba / Crunchy Bridge / Neon / …
- EnterpriseDB / Postgres Pro / Greenplum
- 1 version majeure par an

Un moteur de masquage

6 méthodes de masquage différentes

- Masquage Statique
- Masquage Dynamique
- Sauvegarde Anonymisée
- Vues de Masquage
- Masking Data Wrappers
- **NEW !** Réplication Anonymisée

Une boite à outils

10 types de fonctions

- Destruction
- Bruit
- Permutation
- Données Aléatoire
- Données Synthétique
- Pseudonymisation
- Hachage
- Masquage partielle
- Généralization
- Altération d’Iimages

Charger et créer l'extension

```
ALTER DATABASE foo SET session_preload_libraries = 'anon';

\connect foo

CREATE EXTENSION anon;
```

Privacy By Design : Principes

- L’extension a une approche déclarative de l’anonymization.
- Définit des **règles de masquage** dans le modèle de données
- Grace à la syntaxe `SECURITY LABEL`

Privacy By Design : exemple

```
SELECT * FROM people;

   id    | firstname | lastname |    phone
---------+-----------+----------+-------------
 153478 | Sarah     | Conor    | 0609110911
```

Privacy By Design : masquage avec une donnée factice

```
SECURITY LABEL FOR anon ON COLUMN people.lastname
  IS 'MASKED WITH FUNCTION anon.dummy_last_name()';
```

Privacy By Design : destruction

```
SECURITY LABEL FOR anon ON COLUMN people.id
  IS 'MASKED WITH VALUE NULL';
```

Privacy By Design : masquage partiel

```
SECURITY LABEL FOR anon ON COLUMN people.phone
  IS 'MASKED WITH FUNCTION anon.partial(phone,2,$$******$$,2)';
```

Privacy By Design : Masquage Statique

```
SELECT anon.anonymize_table('people');
```

Privacy By Design : Masquage Statique

```
SELECT * FROM people;

   id    | firstname | lastname   |    phone
---------+-----------+------------+-------------
         | Sarah     | Stranahan  | 06******11
```

Séparation des roles

- Declarer qu'un rol est masqué
- Les règles de masquages sont appliquées automatiquement à ce role
- Par principe un "role masqué" est en lecture seule
- Les autres roles continuent de lire/écrire les données réelles

Séparation des roles : Masquage Dynamique

```
CREATE ROLE skynet LOGIN;

ALTER ROLE skynet SET anon.transparent_dynamic_masking TO true;

SECURITY LABEL FOR anon ON ROLE skynet IS 'MASKED';

GRANT pg_read_all_data to skynet;
```

Séparation des roles : Masquage Dynamique (masqué)

```
SET ROLE skynet;

SELECT * FROM people;

 id | firstname | lastname  |    phone
----+-----------+-----------+-------------
    | Sarah     | Donahue   | 06******11
```

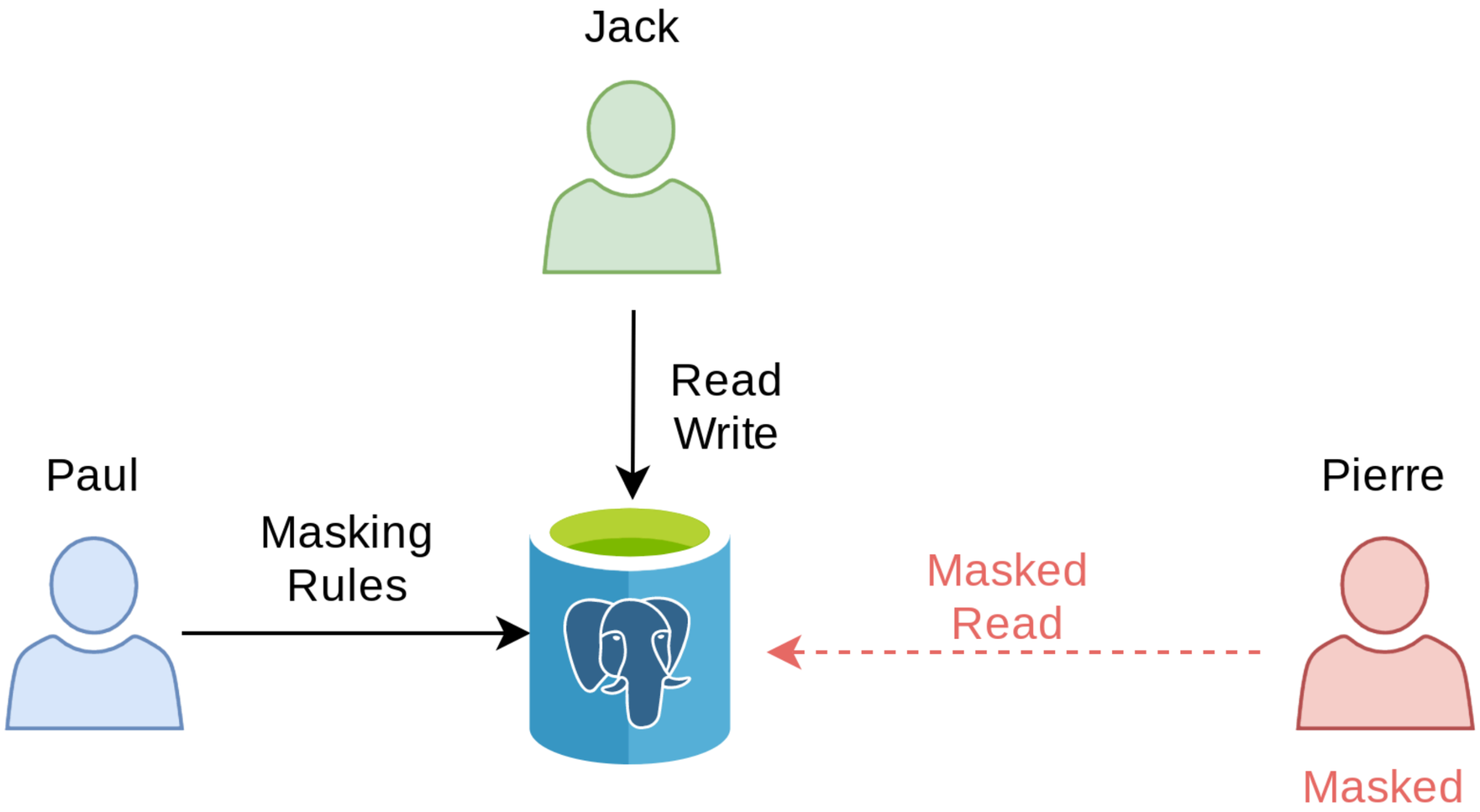
Séparation des roles : Masquage Dynamique (superuser)

```
SET ROLE postgres;

SELECT * FROM people;

 id     | firstname | lastname |   phone
--------+-----------+----------+------------
 153478 | Sarah     | Conor    | 0609110911
```

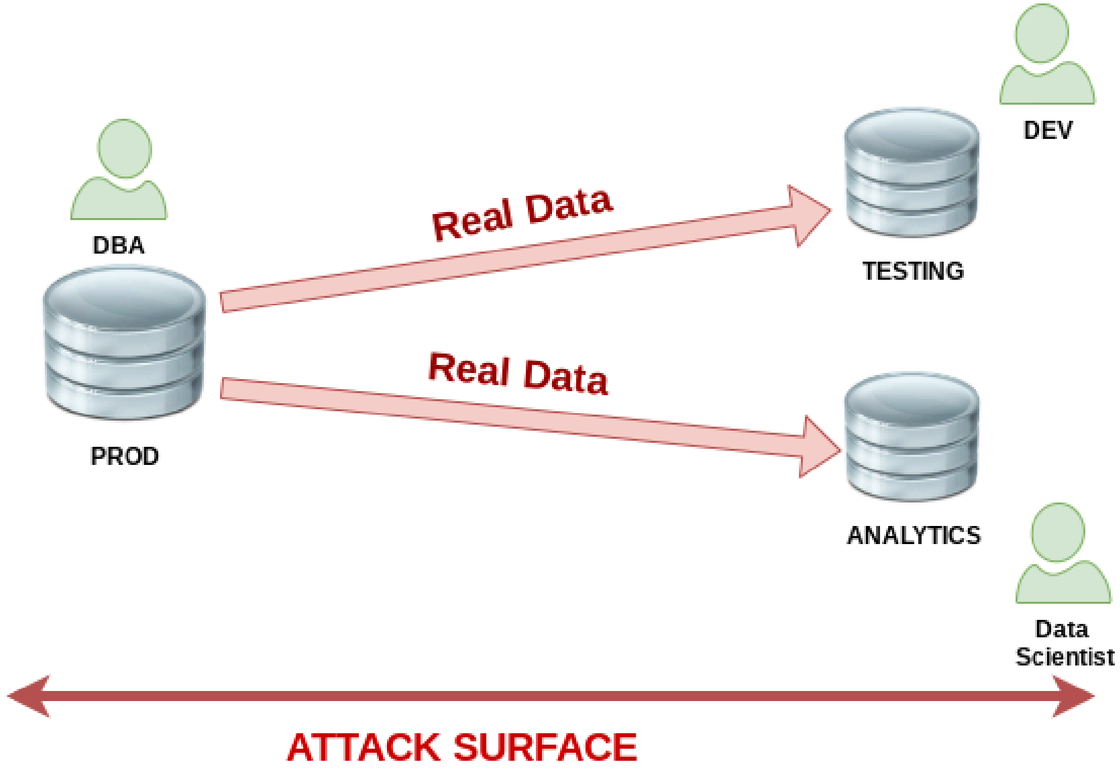
Séparation des roles : Vue d'ensemble



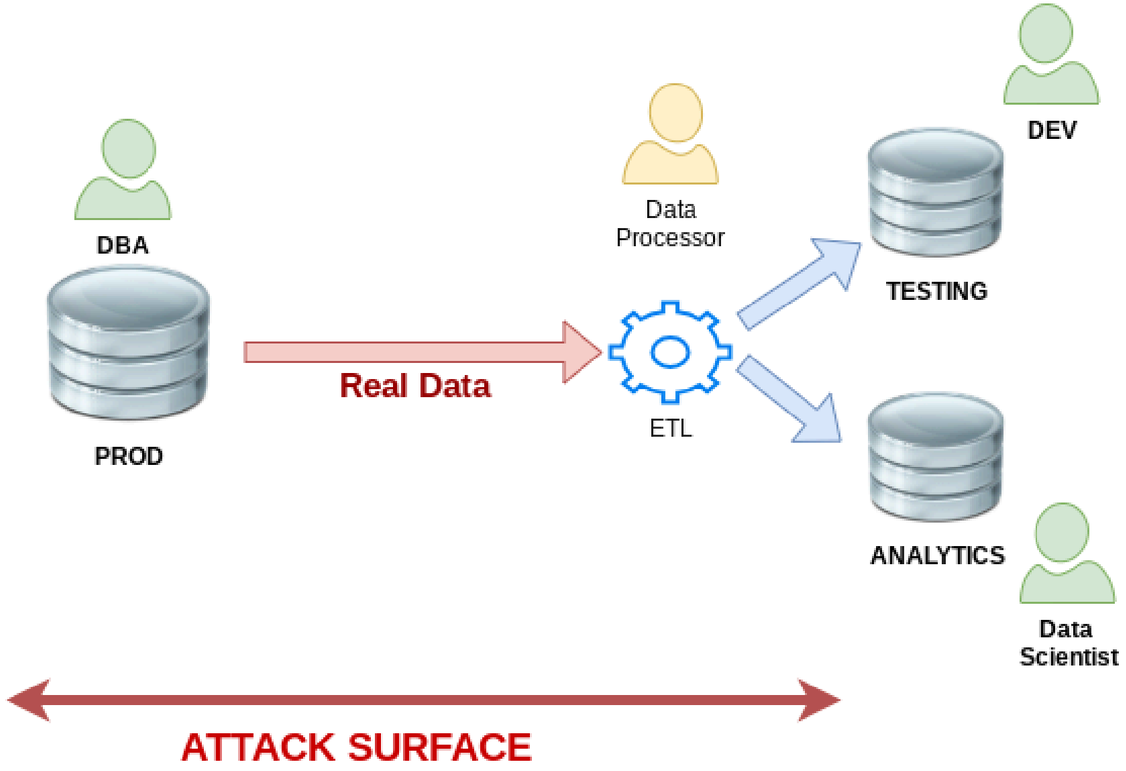
Réduction de la surface d’attaque (ASR)

ASR : example basique

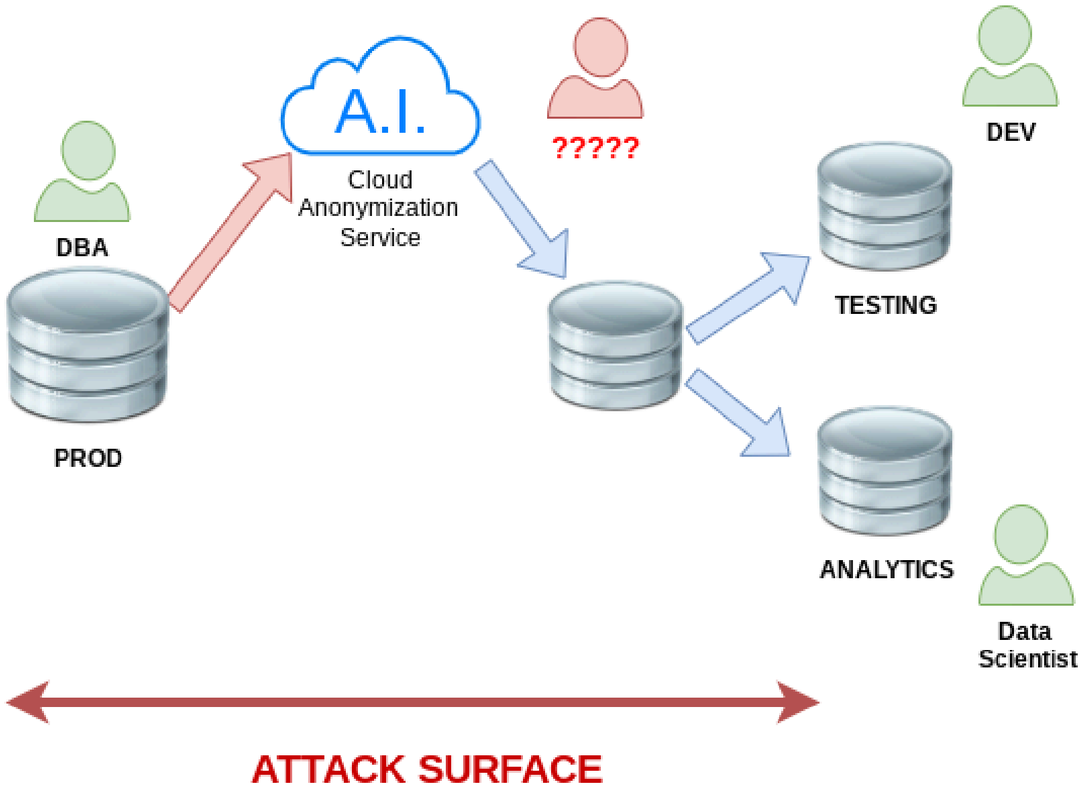
ASR : le pire


DEV
DBA
Real Data
TESTING
Real Data
PROD
ANALYTICS
Data Scientist
ATTACK SURFACE

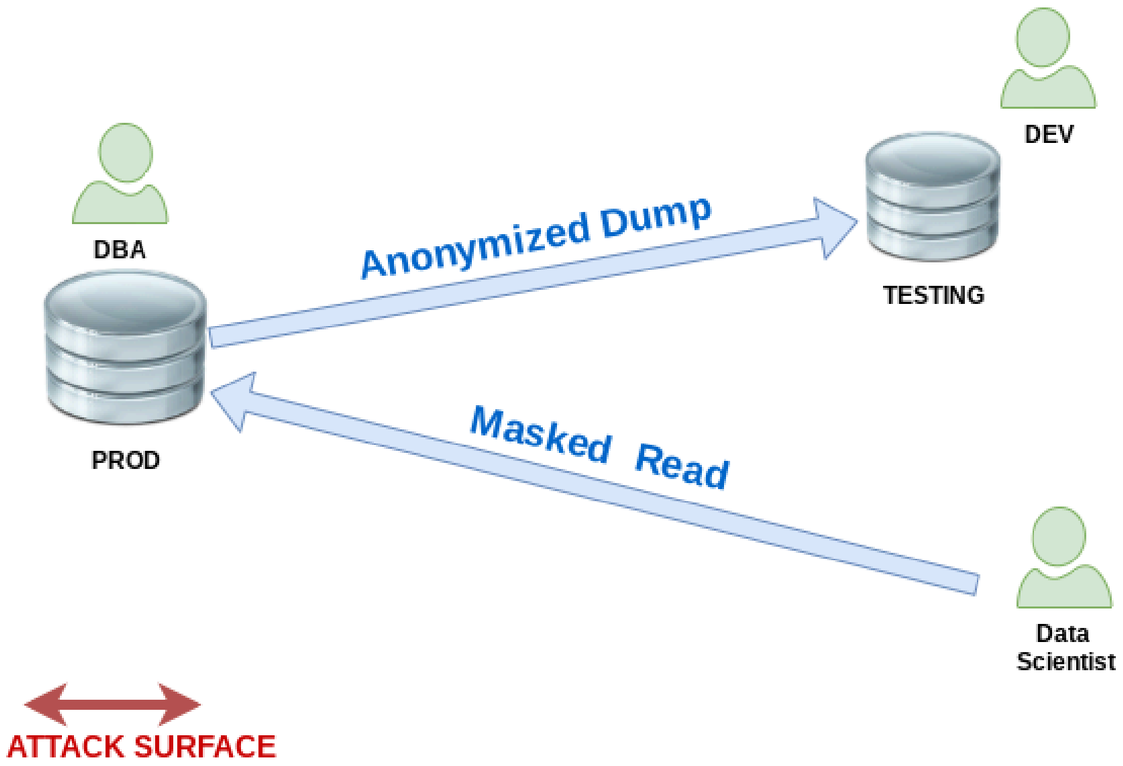
ASR : ETL


DBA
PROD
Real Data
Data
Processor
ETL
DEV
TESTING
ANALYTICS
Data
Scientist
ATTACK SURFACE

ASR : WTF


A.I.
Cloud
Anonymization
Service
?????
DBA
PROD
DEV
TESTING
ANALYTICS
Data
Scientist
ATTACK SURFACE

ASR : PostgreSQL Anonymizer



ASR : Sauvegardes Anonymisées (configuration)

```
CREATE ROLE anon_dumper LOGIN PASSWORD 'CHANGEME';

ALTER ROLE anon_dumper SET anon.transparent_dynamic_masking = True;

SECURITY LABEL FOR anon ON ROLE anon_dumper IS 'MASKED';

GRANT pg_read_all_data to anon_dumper;
```

ASR : Sauvegardes Anonymisées (export)

```
$ pg_dump -h localhost -U dump_anon [...] foo > anonymous_dump.sql
```

On peut maintenant partager le fichier

`anonymous_dump.sql`

et le recharger sur des environements de tests / dev / etc.

Minimisation

Minimisation

- Dans la plupart des cas, un role masqué n'a pas besoin de **TOUTES** les données
- Un echantillon est suffisant pour les tests, demo, etc.
- Connaissez-vous la clause `TABLESAMPLE` ?

Minimisation : Exemple 1/2

Imaginons une table de logs HTTP

On veut supprimer les addresses IP et conserver uniquement 10% des lignes

```
CREATE TABLE http_logs (
  id integer NOT NULL,
  date_opened DATE,
  ip_address INET,
  url TEXT
);
```

Minimisation : Exemple 2/2

```
SECURITY LABEL FOR anon ON COLUMN http_logs.ip_address
  IS 'MASKED WITH VALUE NULL';

SECURITY LABEL FOR anon ON TABLE http_logs
  IS 'TABLESAMPLE BERNOULLI(10)';
```

Evaluation des risques

- L'extension implémente le concept de `k-anonymity`
- Le facteur k décrit à quel point les individus sont "indistingables" dans un jeu de données.

Privacy By Default

Privacy By Default : Exemple 1/3

Reprenons la table de logs HTTP

```
CREATE TABLE http_logs (
    date_opened DATE,
    ip_address INET,
    url TEXT
    browser_agent DEFAULT 'unknown'
  );
```

Privacy By Default 2/3

```
SELECT * FROM access_logs LIMIT 1;

  date_opened   |      ip_addr      |     url      |     browser_agent
----------------+-------------------+--------------+------------------------
 2009-01-08     | 192.168.100.128 | /home.html | Mozilla/5.0 (Windows)
```

Privacy By Default 3/3

Now let’s activate privacy by default:

```
ALTER DATABASE foo SET anon.privacy_by_default = True;
```

Privacy By Default : Démasquer les colonnes

```
SECURITY LABEL FOR anon ON COLUMN access_logs.url
  IS 'NOT MASKED';
```

```
SECURITY LABEL FOR anon ON COLUMN access_logs.date_opened
  IS $$ MASKED WITH FUNCTION pg_catalog.date_trunc('year',birth) $$;
```

Privacy By Default : Appliquer les règles

Application des règles

```
SELECT anon.anonymize_database();
 anonymize_database
--------------------
 t
```

Privacy By Default : Résultat

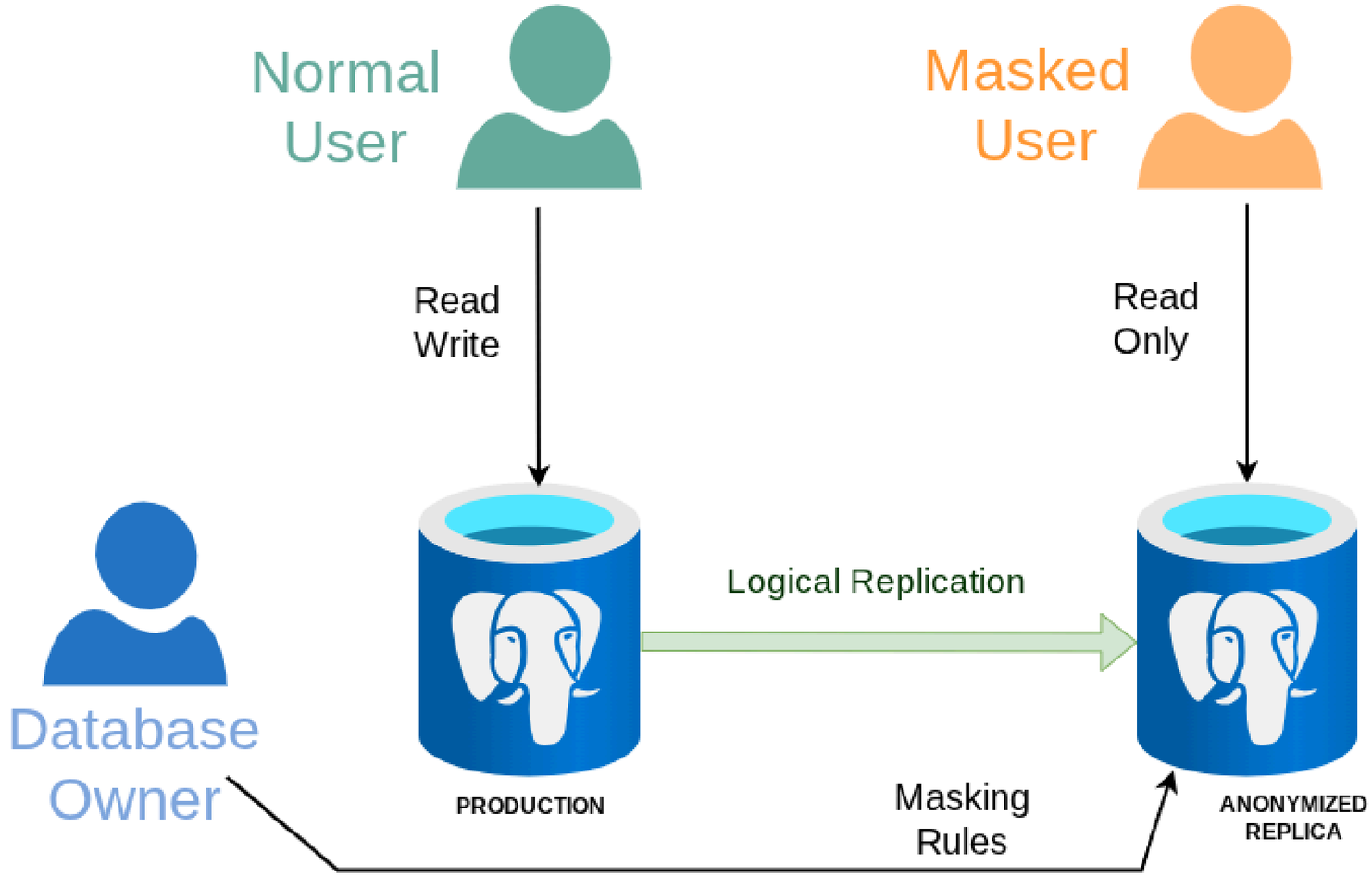
```
SELECT * FROM access_logs LIMIT 1;

 date_open  | ip_addr | url           | browser_agent
------------+---------+---------------+---------------
 2009-01-01 |         | /home.html    | unknown
```

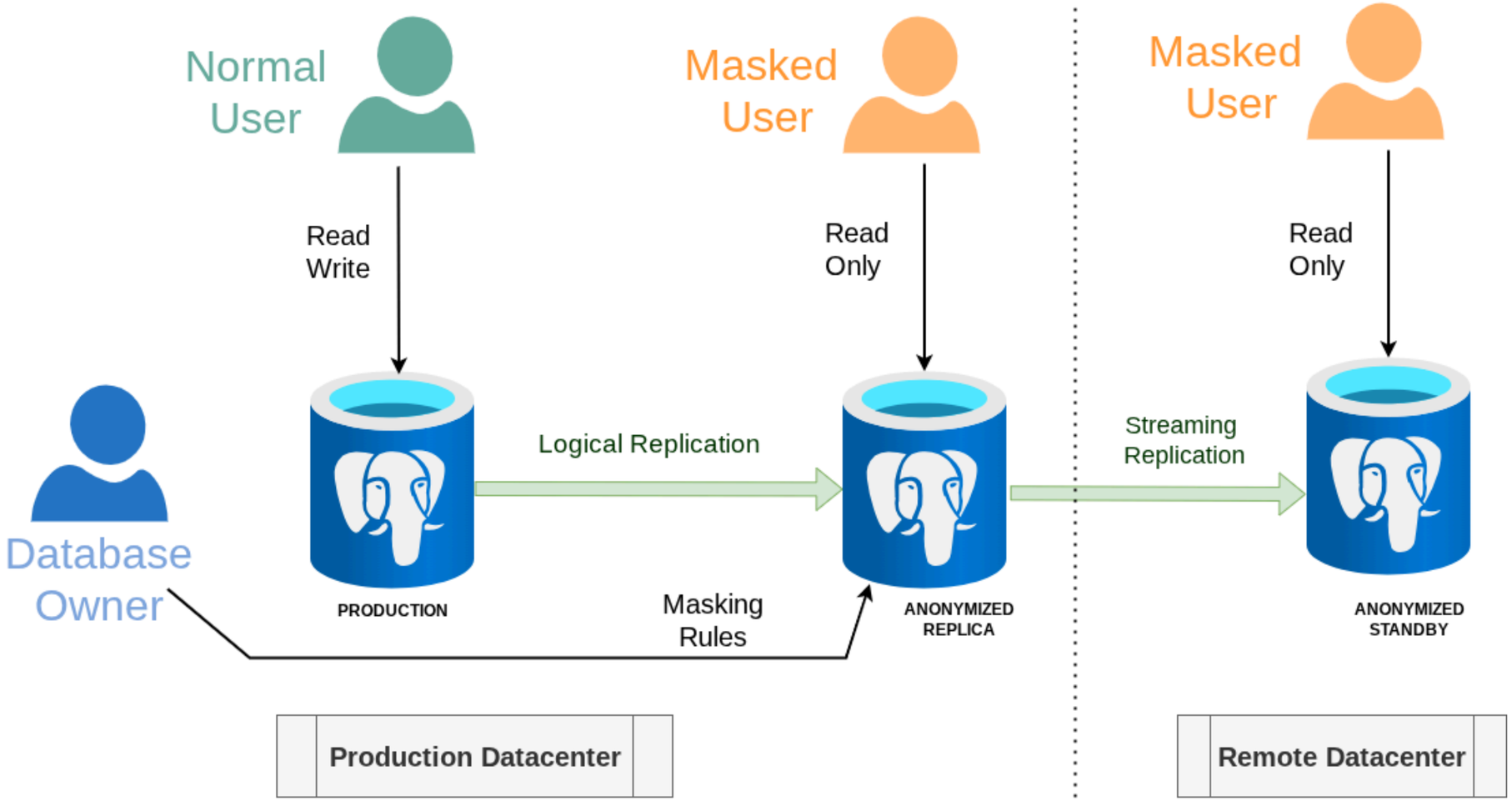
PostgreSQL Anonymizer 3

- Support de Postgres 18 (et fin de Postgres 13)
- Réplication Masquée
- Masquage Sélectif
- Masquage Statique en Parallèle
- Intégration avec l’opérateur CloudNative Postgres
- Confidentialité Différentielle

Réplication masquée



Réplication masquée + Standby



Réplication masquée : Limitations

- Même limitations que la réplication logique
- Les commandes DDL, séquences, Large Objects ne sont PAS repliquées
- Le mode REPLICA IDENTITY FULL n’est pas supporté.
- Les clés primaires ne doivent pas être masquées

Masquage Sélectif

Imaginons une table d'utilisateurs

```
SELECT * FROM users;
 id | login |                password                | admin
----+-------+---------------------------------------+-------
  1 | alice | adfsqfcksqhdqijsdizjdfiqqlq<iqq | f
  2 | bob   | a_very_bad_password                   | t
  3 | carol | 1234                                  | f
```

Masquage Sélectif

On supprime les colonnes login et password

```
SECURITY LABEL FOR anon ON COLUMN users.login    IS 'MASKED WITH VALUE NULL';
SECURITY LABEL FOR anon ON COLUMN users.password IS 'MASKED WITH VALUE NULL';
```

Mais on conserve les infos des admins :

```
SECURITY LABEL FOR anon ON TABLE users IS 'MASKED WHEN admin IS FALSE';
```

Masquage Sélectif

```
SELECT anon.anonymize_table('users');

SELECT * FROM users;
 id | login |              password               | admin
----+-------+-------------------------------------+-------
  1 |       |                                     | f
  2 | bob   | a_very_bad_password                 | t
  3 |       |                                     | f
```

Masquage Statique en Parallèle

- Le masquage statique est séquentiel !
- Possibilité de répartir le travail sur plusieurs jobs
- `SELECT anon.anonymize_database_parallel(4)`

Intégration avec CloudNative Postgres

- Publication d’images OCI (PG18+)
- Facilité d’intégration dans les instances CNPG

Confidentialité Différential

- Une nouvelle approche pour conserver la valeur statistique d’un jeu de données
- Début d’un travail de fond
- Coopération avec le CNRS / Financé par l’ANR

En résumé

Avancer pas à pas :

- **Privacy by Design** est le principe fondamental is the key concept
- Puis **Séparation des roles** et **Réduction de la surface d’attaque**
- Enfin : **Minimisation**, **Evaluation**, **Privacy By Default**

Offre de services

- Pas de version Open-Core / Pas de Vendor Lock-In
- Formation / Workshops
- Aide à la conception
- Déploiement d’un PoC
- Support
- Data Masking Pack

Data Masking Pack

- Support illimité : Niveau 1 à 4
- Garantie de Temps d’Intervention: 4h
- 1 journée d’étude par an (distanciel ou sur site)
- Réunions de co-pilotage des développements (2 par an)
- Diagnostic de conformité réglementaire (1 par an)

Merci !

Projet

https://gitlab.com/dalibo/postgresql_anonymizer/

Tutoriel

https://dali.bo/anon_tuto

Contact

damien.clochard@dalibo.com

Questions ?