Anonymization et Masquage Dynamique avec PostgreSQL

true

Anonymization et Masquage Dynamique avec PostgreSQL

TITRE : Anonymization et Masquage Dynamique avec PostgreSQL
SOUS-TITRE :

BONJOUR

- Damien Clochard
- PostgreSQL DBA & Co-foundateur de Dalibo
- Président de l'association PostgreSQLFr
- Je ne suis pas juriste !

MON CHEMIN

MENU

- RGPD : 1 an plus tard…
- Pourquoi c'est difficile ?
- Flux d'anonymisation
- PostgreSQL Anonymizer
- 7 Techniques d'anonymisation

---

true

RGPD

- Droits Individuels
- Principes
- Impact
- Pseudonymisation vs. Anonymisation

RGPD : LES DROITS INDIVIDUELS

- droit à l'information (Art. 13 et Art. 14)
- droit d'accès (Art. 15)
- droit de rectification (Art. 16)
- droit à la portabilité (Art 20)
- droit d'opposition (Art. 21)
- **droit à l'oubli** (Art. 17)
- **droit à la limitation du traitement** (Art. 18)
- droit de décision automatisée (Art. 22)

(sources: Individual Rights)

RGPD: PRINCIPES & CONCEPTS

- Licéité, loyauté, transparence
- Sécurité
- Minimisation des données
- **Privacy By Design**
- Data Protection By Design
- **Pseudonymisation**
- **Limitation du stockage**
- Précision
- Limitation des finalités

(source: GDPR Principles)

true

SANCTIONS ARE COMING

- Juillet 2019 : 110M€ pour Marriott (UK)
- Juillet 2019 : 204M€ pour British Airways (UK)
- Juin 2019 : 400k€ pour Sergic (France)
- Juin 2019 : 250 k€ pour LaLiga (Espagne)
- Mai 2019 : 170 k€ pour la ville de Bergen (Norvège)
- Avril 2019 : 200k€ pour Airbus (France)
- etc.

(source: GDPR Enforcement Tracker)

ATTENTION À L'ARTICLE 32 !

La plupart des scanctions concernent l'article 32 :

« Mesures techniques et organisationnelles insuffisantes pour assurer la sécurité de l'information »

Autrement dit : **"Fuites de données"**

(source Article 32 - Sécurité du traitement)

PSEUDONYMISATION

« traitement de données à caractère personnel de telle façon que celles-ci ne puissent plus être attribuées à une personne concernée précise sans avoir recours à des informations supplémentaires »

PSEUDONYMISATION != ANONYMISATION

- Pseudonymisation est méthode de protection de données.
- Exemples : Chiffrement ou Hashage
- Les données peuvent reconstuites avec des **données supplémentaires**
- Les données pseudonymisées reste soumises au RGPD
- L'anonymisation est la seule véritable porte de sortie

---

true

POURQUOI C'EST DIFFICILE ?

- Singularisation / Singling Out
- Recoupements / Linkability
- Inférence

(source: WP29 Opinion on Anonymisation Techniques)

SINGULARISATION / SINGLING OUT

Identifier un sujet à partir de valeurs rares ou extrèmes

```
SELECT * FROM employees;
```

```
  id   |  name           | job  | salary
-------+-----------------+------+--------
 1578  | xkjefus3sfzd    | NULL |   1498
 2552  | cksnd2se5dfa    | NULL |   2257
 5301  | fnefckndc2xn    | NULL |  45489
 7114  | npodn5ltyp3d    | NULL |   1821
```

RECOUPEMENTS / LINKABILITY

Identifier un sujet en utilisant des jeux de données externes

- Notes Netflix + Notes IMDB
- Registre d'un hopital + Liste électorale

(sources: Netflix prize + Hospital Reidentification)

INFÉRENCE

Identifier un sujet en regroupant plusieurs **identifiants indirects**

87% de la population des Etats-Unis est identifiable à partir de la date de naissance, du genre et du code postal.

(source : Latanya Sweeney)

---

true

C'EST PERDU D'AVANCE !

On ne peut pas prouver que la **ré-identification** est impossible

(source: De-indentification still doesn't work)

LE RGPD RECONNAIT LE PROBLÈME

« Pour déterminer si une personne physique est identifiable, il convient de prendre en considération l'ensemble des **moyens raisonnablement susceptibles d'être utilisés** par le responsable du traitement ou par toute autre personne pour identifier la personne physique directement ou indirectement »

(source: Recital 26)

MESURER LA PRISE DE RISQUE

Il faut mesurer le “risque raisonable” de ré-identification de manière régulière.

---

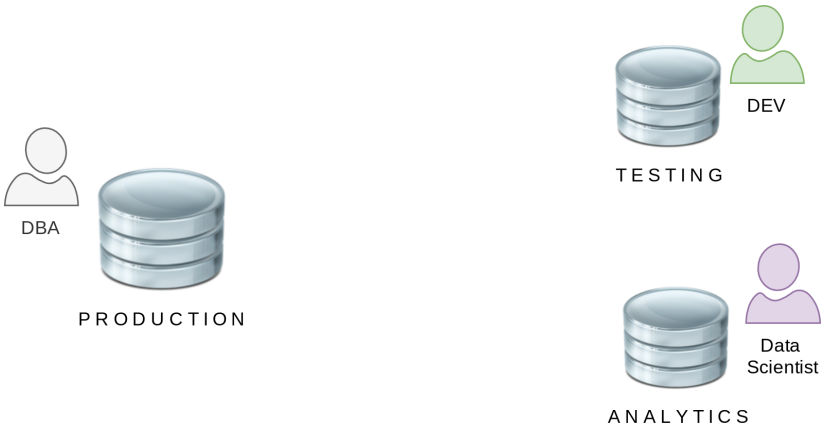
true

FLUX D'ANONYMISATION

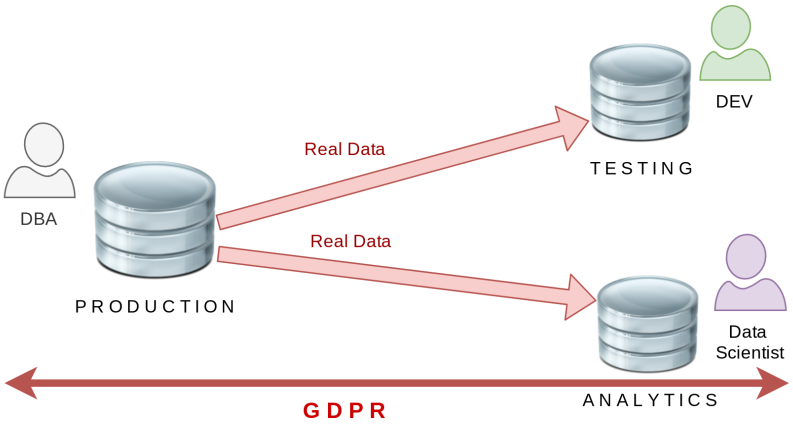
Objectif : minimiser le risque de fuites de donnée réduisant la surface d'attaque

C'est aussi une application directe du principe **Limitation du stockage**

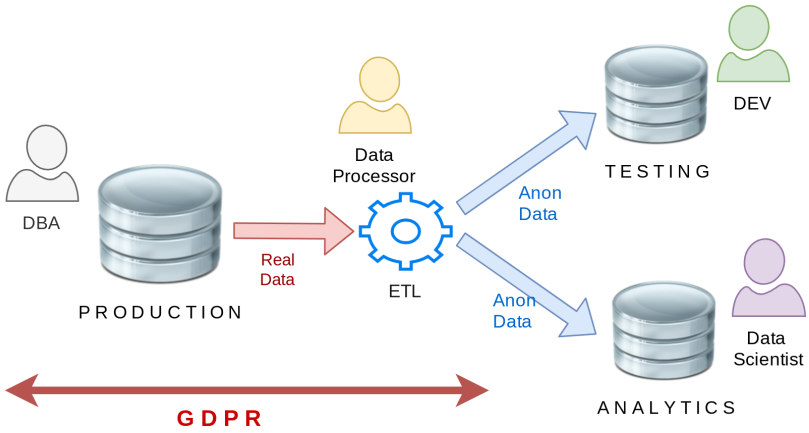
EXEMPLE DE BASE



PAS D'ANONYMIZATION

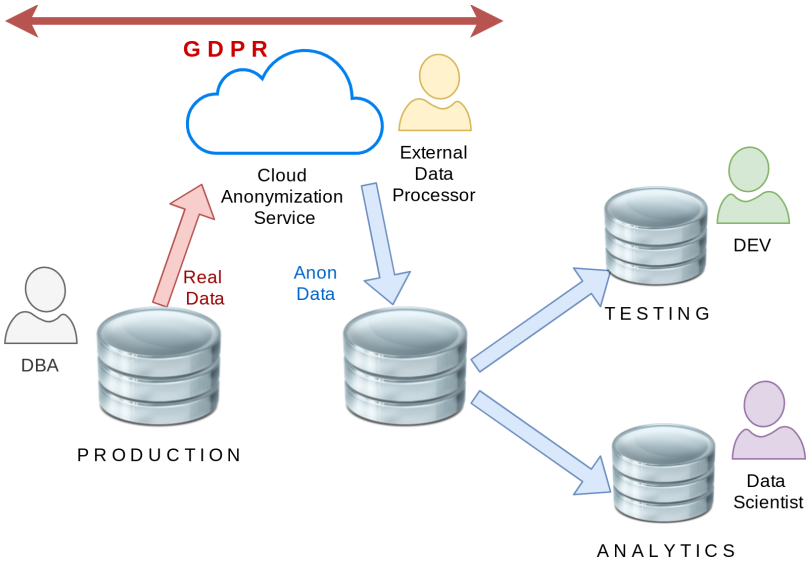


ETL

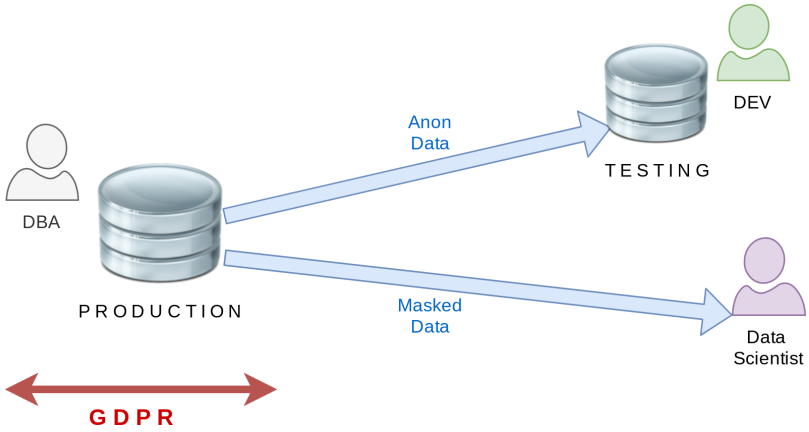


true

ANONYMISATION DANS LES NUAGES (!?!)



POSTGRESQL ANONYMIZER



PostgreSQL Anonymizer

OBJECTIFS

- Declarer des règles de masquage à l'intérieur de la base de données
- Anonymiser à l'intérieur de PostgresQL
- Masquage Dynamique ou Substition Permanente
- Un panel de fonctions de masquage
- Rependre la syntaxe du Masquage Dynamique de SQL Server

EXAMPLE : DONNÉES RÉELLES

```
=# SELECT * FROM customer;
 id  |   full_name      |   birth    | zipcode | fk_shop
-----+------------------+------------+---------+---------
 911 | Chuck Norris     | 1940-03-10 | 75001   | 12
 112 | David Hasselhoff | 1952-07-17 | 90001   | 423
```


true

EXEMPLE : DONNÉES ANONYMISÉES

```
=# SELECT * FROM customer;
 id  |     full_name     |   birth    | zipcode | fk_shop
-----+-------------------+------------+---------+---------
 911 | Michel Duffus     | 1970-03-24 | 63824   | 12
 112 | Andromache Tulip  | 1921-03-24 | 38199   | 423
```

INSTALLER

A partir du dépot RPM de la communauté :

```
$ yum install https://.../pgdg-redhat-repo-latest.noarch.rpm
$ yum install postgresql_anonymizer12
```

CONFIGURER

```
shared_preload_libraries = '[...], anon'
```

CHARGER

```
=# CREATE EXTENSION IF NOT EXISTS anon CASCADE;
=# SELECT anon.load();
```

DECLARER UNE RÈGLE DE MASQUAGE

```
SECURITY LABEL FOR anon
ON COLUMN customer.zipcode
IS 'anon.random_zipcode()';
```

ENSUITE 3 APPROCHES SONT POSSIBLES

- Substitution Permanente (“In-Place Anonymization”)
- Export Anonyme (“Anonymous Dump”)
- Masquage Dynamique (“Dynamic Masking”)

SUBSTITUTION PERMANENTE

```
=# SELECT anon.anonymize_column('customer','zipcode');
=# SELECT anon.anonymize_table('customer');
=# SELECT anon.anonymize_database();
```

SUBSTITUTION PERMANENTE

Réécriture (UPDATE) de toutes les lignes de toutes les tables masquées.

---

true

EXPORT ANONYME

```
=# SELECT anon.dump();
```

EXPORT ANONYME

```
$ psql [...] -qtA -c 'SELECT anon.dump()' foo > foo.sql
```

MASQUAGE DYNAMIQUE

Exemple de base :

```
=# SELECT * FROM people;
 id | fistname | lastname |   phone
----+----------+----------+------------
 T1 | Sarah    | Conor    | 0609110911
(1 row)
```

MASQUAGE DYNAMIQUE

Etape 1 : Activer le moteur de masquage

```
=# CREATE EXTENSION IF NOT EXISTS anon CASCADE;
=# SELECT anon.start_dynamic_masking();
```

MASQUAGE DYNAMIQUE

Etape 2 : Déclarer utilisateur masqué

```
=# CREATE ROLE skynet LOGIN;
```

```
=# SECURITY LABEL FOR anon ON ROLE skynet
-# IS 'MASKED';
```

L'utilisateur masqué a un accès en lecture seule aux données anonymisées.

MASQUAGE DYNAMIQUE

Etape 3 : Declarer les règles de masquage

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

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

MASQUAGE DYNAMIQUE

Etape 4 : Se connecter avec l'utilisateur masqué

```
$ psql peopledb -U skynet -c 'SELECT * FROM people;'
 id | fistname | lastname  |   phone
----+----------+-----------+------------
 T1 | Sarah    | Stranahan | 06******11
(1 row)
```

---

true

TECHNIQUES D'ANONYMISATION

L'extension fournit toute une batterie de fonctions pour appliquer 7 techniques d'anonymization:

- Destruction
- Bruit
- Brassage
- Randomization
- Imitation (faking)
- Destruction partielle
- Généralisation

DESTRUCTION

```
SECURITY LABEL FOR anon
  ON COLUMN users.address
  IS 'MASKED WITH VALUE ''[CONFIDENTIELLE]'' ';
```

DESTRUCTION

- Simple
- Rapide
- Efficace

BRUIT

```
=# SELECT anon.add_noise_on_numeric_column(user, salary, 0.1)
```

Toutes les valeurs seront décalées aléatoirement avec un ratio de +/- 10%

BRUIT

- Les données restent réalistes.
- `AVG()` et `SUM()` sont similaires à l'original
- marche uniquement avec les dates et les valeurs numériques
- Les "valeurs marginales" sont un risque de ré-identification ("singularisation")
- Attention aux attaques par répétition ! Ne pas utiliser avec le masquage dynamique

---

BRASSAGE

```
# SELECT anon.shuffle_column(employee, fk_company, id);
```

BRASSAGE

- Les données ne sont pas modifiées
- Parfait pour les clés étrangères
- Marche mal avec les distributions peu ou mal distribuées (ex: booléens)
- La table doit avoir une clé primaire

RANDOMISATION

```
=# SECURITY LABEL FOR anon
-# ON COLUMN employee.birth
-# IS 'MASKED WITH FUNCTION
-#     anon.random_date_between(''01/01/1920'',now())
-# ';
```

---

true

RANDOMISATION

- Simple et rapide
- Utile pour les colonnes avec la contrainte `NOT NULL`
- A éviter pour les stats et l'analytique

IMITATION (FAKING)

```
=# SECURITY LABEL FOR anon
-# ON COLUMN employee.lastname
-# IS 'MASKED WITH FUNCTION
-#     anon.fake_last_name()
-# ';
```

IMITATION (FAKING)

- Une version plus élaborée de la randomisation
- Parfait pour les dev, les démos, les tests CI, etc.
- Vous pouvez charger vos propres données factices !

DESTRUCTION PARTIELLE

```
=# SECURITY LABEL FOR anon
-# ON COLUMN employee.phone
-# IS 'MASKED WITH FUNCTION anon.partial(phone,4,'******',2)';
```

+33142928107 devient +331******07

DESTRUCTION PARTIELLE

- Parfait pour les numéros de téléphone, cartes de crédit, etc.
- Le sujet peut toujours reconnaitre sa donnée
- La transformation est `IMMUTABLE`
- Marche seulement pour les types TEXT / VARCHAR

GENERALIZATION

```
SELECT * FROM patient;
     ssn      | firstname |  zip  |   birth    |  disease
--------------+-----------+-------+------------+--------------
 253-51-6170 | Alice     | 47012 | 1989-12-29 | Flu
 091-20-0543 | Bob       | 42678 | 1979-03-22 | Allergy
 565-94-1926 | Caroline  | 42678 | 1971-07-22 | Flu
 510-56-7882 | Eleanor   | 47909 | 1989-12-15 | Acne
```

GENERALIZATION

```
CREATE MATERIALIZED VIEW generalized_patient AS
SELECT
  'REDACTED'::TEXT AS firstname,
  anon.generalize_int4range(zipcode,1000) AS zipcode,
  anon.generalize_daterange(birth,'decade') AS birth,
  disease
FROM patient;
```

---

true

GENERALIZATION

```
SELECT * FROM generalized_patient;
 firstname |      zip      |          birth          |  disease
-----------+---------------+-------------------------+--------------
 REDACTED  | [47000,48000) | [1980-01-01,1990-01-01) | Flu
 REDACTED  | [42000,43000) | [1970-01-01,1980-01-01) | Allergy
 REDACTED  | [42000,43000) | [1970-01-01,1980-01-01) | Flu
 REDACTED  | [47000,48000) | [1980-01-01,1990-01-01) | Acne
```

GENERALIZATION

- Les données restent **vraies** mais moins précises
- Utilise les types RANGE
- Idéal pour les statistiques et l'analyse de données
- Le dégré d'anonymisation est quantifiable avec l'indicateur de k-anonymat
- Le masque dynamique ne marche pas car le schéma est différent
- Inutilisable pour les tests d'intégration

EN RÉSUMÉ

- Les sanctions du RGPD sont bien réelles
- Les fuites de données sont le plus gros risque
- Reduire la surface d'attaque
- Anonymiser dès que possible
- Anonymiser dans la base de données
- Différentes techniques pour différents usages
- Le chiffrement n'est pas de l'anonymisation !

BATAILLE POUR LA VIE PRIVÉE

- Les développeurs doivent écrire les règles de masquage
- C'est difficile.... PostgreSQL est un bon point de départ
- La bataille de l'open source est gagnée,
- ... il faut maintenant se battre pour protéger la vie privée

---

COMMENT CONTRIBUER ?

- Feedback et bugs !
- Images et geodata
- rejoindre le projet sur :

https://gitlab.com/dalibo/postgresql_anonymizer

---

true

DALIBO RECRUTE

- DBA de Production
- DBA d'études
- Développeur⍰se backend Python
- Commercial Grands Comptes

https://www.dalibo.com/jobs

MERCI !

- Contact : damien.clochard@dalibo.com
- Follow : @daamien
- Nos autres Projets : Dalibo Labs