

Data Masking and Anonymization for PostgreSQL



true

Data Masking and Anonymization for PostgreSQL

TITRE : Data Masking and Anonymization for PostgreSQL

SOUS-TITRE :

- The Anonymization Challenge
- 8 Strategies
- PostgreSQL Anonymizer
- The Future

ID CARD

John Doe

EMAIL : d*****@g*****.com

COMPANY : ab436232efab1dc5

PROJECTS :
[CONFIDENTIAL]



FOSDEM PG DAY

MY STORY

LET'S DO THIS !

- jailer
 - pgantomizer
 - ARX
 - pgsync
 - anomyze-it
 - anonymizer
 - mat2
 - Talend
 - open anonymizer
-

MEH.

- Do I really need Java or Ruby for this ?
 - Describe data manipulations in a YAML file ?
 - Extract data from Postgres and load them back ?
-

So I started building my own SQL script...



WHAT IS ANONYMIZATION ?

Modify a dataset to avoid any identification while remaining suitable for testing, data analysis and data processing.

WHY ?

- Open Data
 - Continuous Integration
 - Functional Testing
 - Analytics
 - Audit
 - Development
-

STATIC VS DYNAMIC ANONYMIZATION

- **Dynamic Masking** offers an altered view of the real data without modifying it. Some users may only read the masked data, others may access the authentic version.
 - **Permanent Alteration** is the definitive action of substituting the sensitive information with uncorrelated data. Once processed, the authentic data cannot be retrieved.
-

WHY IT'S HARD

- Singling out
 - Linkability
 - Indirect Identifiers
-

SINGLING OUT

The possibility to isolate a record and identify a subject in the dataset.

```
SELECT * FROM employees;
```

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

LINKABILITY

Identify a subject in the dataset using other datasets

- Netflix Ratings + IMDB Ratings ¹
 - Hospital visits + State voting records ²
-

INDIRECT IDENTIFIERS

87% of the U.S. Population are uniquely identified by date of birth, gender and zip code ³

¹https://www.cs.utexas.edu/~shmat/shmat_oak08netflix.pdf

²<https://techscience.org/a/2015092903/>

³<http://latanyasweeney.org/work/identifiability.html>

THIS IS A LOSING GAME !

You can't measure the **usefulness** of the anonymized dataset

You can't prove that **re-identification** is impossible ⁴

WHAT DOES THE GDPR SAY ?

- « Anonymization is hard » (WP29 Opinion 05/2014)
 - « Pseudonymization is enough » (Recital 83)
 - « Data Protection By Design » (Article 25)
-

⁴<http://randomwalker.info/publications/no-silver-bullet-de-identification.pdf>

8 STRATEGIES TO ANONYMIZE DATA

EXAMPLE

```
CREATE TABLE marriott_client (  
  id SERIAL,  
  name TEXT NOT NULL,  
  passwd TEXT NOT NULL,  
  address TEXT,  
  age INTEGER,  
  points INTEGER,  
  phone TEXT  
);
```

0. SAMPLING

0. SAMPLING

-- Work only on 20% of a table

```
SELECT * FROM marriott_client  
TABLESAMPLE BERNOULLI(20);
```

0. SAMPLING

Sampling is not Anonymization but...

- Direct implementation of the “Data Minimisation” principle of GDPR
 - Reducing dataset will reduce the risk of re-identification
 - The anonymization process will be faster
 - Use `pg_sample` to keep referential integrity among several tables
-

1. SUPPRESSION

1. SUPPRESSION

-- Just remove the data

```
UPDATE marriott_client  
SET name = NULL;
```

```
UPDATE marriott_client  
SET points= 0;
```

```
UPDATE marriott_client  
SET passwd = "CONFIDENTIAL";
```

1. SUPPRESSION

- Simple and Efficient
 - Direct implementation of the “Data Minimisation” principle of GDPR
 - Breaks integrity constraints (**CHECK / NOT NULL**)
 - Useless for functional testing
-

2. RANDOM SUBSTITUTION

2. RANDOM SUBSTITUTION

-- Replace the data with a purely random value

```
UPDATE marriott_client  
SET name = md5(random()::text);
```

```
UPDATE marriott_client  
SET points= 100*random();
```

2. RANDOM SUBSTITUTION

- Simple and Fast
 - Avoid breaking **NOT NULL** constraints
 - Still useless for functional testing
-

3. ADDING NOISE

3. ADDING NOISE

```
-- Randomly "shifting" the value of +/- 25%
UPDATE marriott_client
SET points = points * (1+(2*random()-1) * 0.25) ;
```

3. ADDING NOISE

- The dataset remains meaningful
 - `AVG()` and `SUM()` are similar to the original
 - works only for dates and numeric values
 - "extreme values" may cause re-identification ("singling out")
-

4. ENCRYPTION

4. ENCRYPTION

```
-- uses an encryption algorithm
CREATE EXTENSION pgcrypto;

-- generate a random salt and throw it away
UPDATE marriott_client
SET name = crypt('name', gen_salt('md5'));
```

4. ENCRYPTION

- Respect the **UNIQUE** constraint
 - Possible implementation of "Pseudonymization"
 - The transformation can be **IMMUTABLE**
 - Functional testing is weird
 - If the key is stolen, authentic data can be revealed.
-

5. SHUFFLING

5. SHUFFLING

```
-- Mixing values within the same column
WITH p1 AS (
  SELECT row_number() over (order by random()) n,
         points AS points1
  FROM marriott_client),
p2 AS (
  SELECT row_number() over (order by random()) n,
         id AS id2
  FROM marriott_client )
UPDATE marriott_client
SET points = p1.points1
FROM p1 join p2 on p1.n = p2.n
WHERE id = p2.id2;
```

5. SHUFFLING

- The dataset remains meaningful
 - Perfect for Foreign Keys
 - Works bad with low distribution (ex: boolean)
-

6. FAKING / MOCKING

6. FAKING / MOCKING

*-- replace data with **random-but-plausible** values.*

```
UPDATE marriott_client  
SET address = fake_address();
```

6. FAKING / MOCKING

- The faking function is hard to write (see faker)
 - For complex data types, it's hard produce relevant synthetic data
 - Not appropriate for analytics because the values are not "real"
-

7. PARTIAL SUPPRESSION

7. PARTIAL SUPPRESSION

-- "01 42 92 81 00" becomes "XX XX XX 81 00"

```
UPDATE marriott_client
SET phone = 'XX XX XX ' || substring(phone FROM 9 FOR 5 );
```

7. PARTIAL SUPPRESSION

- The user can still recognize his/her own data
 - Transformation is **IMMUTABLE**
 - Works only for TEXT / VARCHAR types
-

8. GENERALIZATION

8. GENERALIZATION

*-- Instead of "Client X is 28 years old",
-- Let's say "Client X is between 20 and 30 years old."*

```
CREATE TABLE anonymous_client
AS SELECT
    id,
    '*' AS name,
    int4range(age/10*10, (age/10+1)*10) AS age
FROM marriott_client
;
```

8. GENERALIZATION

- The data type has changed
 - Breaks CI, functional tests and any use related to the application.
 - Fine for data analytics and aggregation.
 - Risk of singling-out.
-

RECAP

- Suppression : Useless attributes
 - Random : Useless attributes with constraints
 - Noise (Numeric and Dates) : Dev / CI / Functional Testing
 - Encryption (Text) : UNIQUE attributes
 - Shuffle : Foreign keys / Analytics
-
- Faking : Dev / CI / Functional Testing
 - Partial (Text) : Direct Identifiers
 - Generalization (Numeric and Dates) : Analytics
-



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

WHAT IS THIS ?

- Started as a personal project last year
 - Now part of the “Dalibo Labs” initiative
 - This is a prototype !
-

GOALS

- Transform data inside PostgreSQL
 - Implement useful features (noise, shuffling, faking, etc.)
 - Define anonymization policy with SQL statements
 - PoC for Dynamic Masking
-

INSTALL

```
$ sudo pgxn install postgresql_anonymizer
```

LOAD

```
CREATE EXTENSION anon CASCADE;  
SELECT anon.load();
```

RANDOM

```
UPDATE marriott_client  
SET birth=anon.random_date_between('01/01/1920',now());
```

NOISE

```
-- shift date d1 on table t1 by +/- 2 years  
SELECT anon.add_noise_on_datetime_column(t1,d1,'2 years');
```

SHUFFLE

```
SELECT anon.shuffle_column(marriott_client,points);
```

FAKING

```
UPDATE marriott_client  
SET company = anon.fake_company();
```

PARTIAL

```
-- replace 01 42 92 81 00 by XX XX XX 81 00  
UPDATE marriott_client  
SET phone = anon.partial(phone,0, 'XX XX XX ', 5);
```

DECLARATIVE DYNAMIC MASKING !

- Regular user can see the real data
 - Others can only view anonymized data
-

CREATE A MASKED USER

```
CREATE ROLE skynet;  
COMMENT ON ROLE skynet IS 'MASKED';
```

PUT MASKS ON COLUMNS

```
-- Random Mask  
COMMENT ON COLUMN marriott_client.surname  
IS  
'MASKED WITH FUNCTION anon.random_last_name()';  
  
-- Partial Mask  
COMMENT ON COLUMN marriott_client.phone  
IS  
'MASKED WITH FUNCTION anon.partial(phone,2,$$*-***-***$$,2)';
```

NORMAL USER WILL SEE :

```
SELECT * FROM marriott_client WHERE id = '800';
  id | firstname | surname      | phone
-----+-----+-----+-----
  800 | Sarah     | Connor      | 408-555-1439
(1 row)
```

MASKED USER WILL SEE :

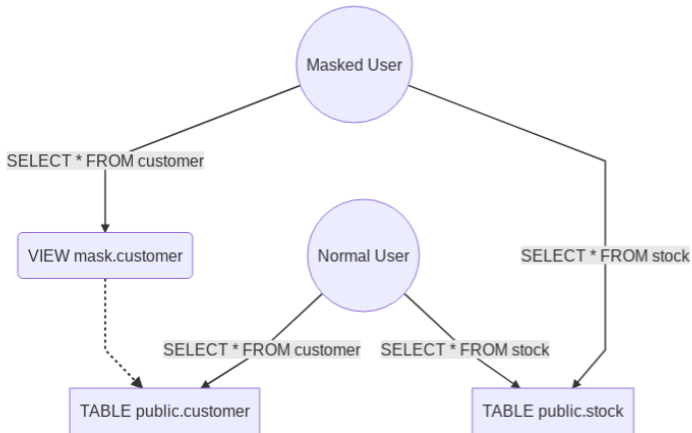
```
SET ROLE skynet;
SELECT * FROM marriott_client WHERE id = '800';
  id | firstname | surname      | phone
-----+-----+-----+-----
  800 | Sarah     | Nunziata     | 40*-***-*19
(1 row)
```

MASKED USERS CAN'T READ/WRITE MASKED COLUMNS

```
SET ROLE skynet;
SELECT * FROM marriott_client
WHERE surname ILIKE 'CONNOR';
(0 rows)

SET ROLE skynet;
DELETE FROM marriott_client
WHERE surname ILIKE 'CONNOR';
ERROR: permission denied for view marriott_client
```

HOW IT WORKS



JUST 1 BIG FAT HACK

Basically :

- 500 lines of pl/pgsql
- A misappropriation of the **COMMENT** syntax
- An event trigger on DDL commands
- Silently creates a “masking view” upon the real table
- Tricks masked users with **search_path**
- use of TABLESAMPLE with **tms_system_rows** for random functions

LIMITS

- PostgreSQL 9.6 and later
 - Only one schema
 - What if the columns COMMENTS are really used ?
 - Masked users can't use pg_dump
 - Performances ?
 - `SELECT * from pg_stats`
-

TOWARD «ANONYMITY BY DESIGN»

Any anonymization policy should be defined as close as possible to the data. Just like integrity constraints, security rules and triggers. PostgreSQL should allow developers to declare how a column will be masked.

EXTEND POSTGRES DDL DIALECT

```
ALTER TABLE marriott_client
ALTER COLUMN email
ADD MASK WITH FUNCTION foo(email);
```

```
GRANT UNMASK TO admin;
```

(MS SQL Server already has it)

THE PLAN


- Step 1 : build a PoC in pl/pgsql
 - Step 2 : implement C functions
 - Step 3 : build a patch for Postgres ?
-

HOW TO HELP US

- Feedback and bugs !
 - images and geodata
 - Ideas on how to implement this as an extension
 - Advice about extending the SQL syntax
 - Research on K-Anonymity and Differential_Privacy
-

CONTACT ME !

ID CARD

<p>Damien Clochard</p> <p>EMAIL : daamien@gmail.com</p> <p>COMPANY : dalibo</p> <p>PROJECTS : https://github.com/daamien/ https://gitlab.com/daamien/</p>	
---	--

FOSDEM PG DAY

NOT JUST ABOUT GDPR....

Free software communities must lead the way to build a future where privacy and anonymity are available to everyone. And of course PostgreSQL has an important role to in this domain because it's by far the world's most dynamic and innovative database engine.

LINKS

- No silver bullet: De-identification still doesn't work
 - Dynamic Data Masking With MS SQL Server
 - Ultimate Guide to Data Anonymization
 - UK ICO Anonymisation Code of Practice
-

MORE LINKS

- L. Sweeney, Simple Demographics Often Identify People Uniquely, 2000
 - How Google anonymizes data
 - IAPP's Guide To Anonymisation
 - A Face Is Exposed for AOL Searcher No. 4417749
-

PHOTO CREDITS

qiagen, arvin_benitez, tonynewell, gorbould, vshioshvili, w4nd3r10st, ysn, 125222692@N04, thecostumeguild