EVL Anonymization Microservice

EVL Anonymization Microservice enables fast, automated and cost-effective anonymization of data sets. It can be used for pseudonymization and anonymization of the production data according to GDPR requirements as well as for the protection of commercially sensitive data from developers, testers and other outside contractors.

EVL Anonymization advantages

- Custom functions can be easily designed and embedded into the solution
- Low implementation and operating costs
- Combination of anonymization techniques: Randomization, Tokenization, Encryption, Masking

EVL Microservices are built on top of the core EVL software and retain its flexibility, robustness, high productivity, and ability to read data from various sources; including csv and Excel files, databases – Oracle, Teradata, SQL Server, etc – and Hadoop streaming data like Kafka.

EVL anonymization functions

Following methods, functions and their variations are available on several data types. All functions return NULL when input is NULL.

Method	Data type	EVL Function	Examples
Masking	string	`str_mask_left()` `str_mask_right()`	`str_mask_left("1234 5678 9012 3456",4,'X')` ⇒ `"XXXX XXXX XXXX 3456"`, i.e. mask by `X` from left, but keep 4 characters from right
Anonymization	string	`anonymize()` *)	`anonymize("abcd",2,8)` ⇒ `"s8L7df"`, i.e. returns a string of the length between 2 and 8
Anonymization	numbers	`anonymize()` *)	`anonymize(573,0,1023)` ⇒ 850, i.e. returns an integer between 0 and 1023
Anonymization	date, timestamp	`anonymize()` *)	`anonymize(date("2018-05-25"),1,6,15)` ⇒ 2019-09-17, i.e. return given date ± 1 year, ± 6 months and ± 15 days
Unique anonymization	integral data types	`anonymize_uniq()` **)	`anonymize_uniq((uint)133)` ⇒ 85.189.556 i.e. return uint, but no other than number 133 can return 85.189.556, so this mapping is unique
Encryption	string	`encrypt()` `decrypt()`	`encrypt("abcd")` ⇒ `"99bd ... c4u8"`, `decrypt("99bd ... c4u8")` ⇒ `"abcd"`, i.e. return encrypted/decrypted value
Tokenization	string	`anonymize(s, len(s),len(s))`	Tokenzation is actually only specific application of `anonymize()` function
Hashing	string	`sha256sum()`	`sha256sum("abcd")` ⇒ `"fc4b5fd6 ... b801d62c"`
Salted hash	string	`sha256sum()`	just add a salt and do the checksum

*) For given value and given salt produces the same output, but might happen that two different values obtain the same anonymized value.
**) For given value and given salt produces the same output, but in a unique way, so bijection is guaranteed. Particularly useful for IDs.

EVL Anonymization project

An anonymization project consists of following steps:

1. unzipping EVL distribution and defining several variables and paths
2. filling-in a CSV file defining source type (e.g. CSV, Oracle, ...), entity (e.g. table or file name) and field names, their order and data types and anonymization functions to be applied
3. running generation of EVL jobs for each entity
4. running EVL jobs in a batch or individually

Example

Following example shows an implementation of anonymization data for a development and test environment of one banking application.

Set variables

```
DATA_SOURCE_DIR="/some/path/source"            # Source data directories
DATA_ANON_DIR="/some/path/anon"                # Target data directories
export EVL_ANON_SALT_PATH="/some/path/.salt"   # Path to salt
```

Anonymization definition for file test.csv

Src	Entity	Ord	Field	Data type	Null	Anon type	EVL Func.	Description
FILE	test.csv	1	ID	int	No	ANON_UNIQ		Unique identifier of the person
FILE	test.csv	2	ACCOUNT	int	No	ANON_UNIQ		Unique account number
FILE	test.csv	3	RC	string			anon_rc(IN)	Personal ID (must be Mod 11), customer's custom function
FILE	test.csv	4	ST_DATE	date("%m/%d/%Y")		ANONYMIZE		Start date of the account
FILE	test.csv	5	SCORE	decimal(15,2)		ANONYMIZE		Score of the account holder
FILE	test.csv	6	DESC	string		ANONYMIZE		Description of the account
FILE	test.csv	7	TEXT	string				Free text - no anonymization

Run

```
evl run/generate_jobs.evl  # generating evl jobs from the config file
evl run/anon.test.evl      # running the anonymization job for an entity "TEST"
```

Example data – one record

Before:

```
87981042|178|5606206199|06/08/2017|1539.34|
Account has been established on another name then changed|He prefers blue color
```

After:

```
998451644|1716305276|5802153599|07/09/2016|741133154.40|
jZy96jPqkiH8GMYhdj9Ti6O8TdPVQKDciDmd8Nyi|He prefers blue color
```

Case Study

One bank needed to provide production data for the development team so the data couldn't be re-identified by keeping the entity relationships. The source were 100+ tables stored in csv files, SQL Server, Informix and Oracle. The target for the anonymized development data was Oracle database. Customer filled-in one configuration file containing all data definitions and anonymization types and parameters leading to the source files (directories for csv files and connect strings to databases). The EVL anonymization jobs were created automatically and run in parallel batches with great performance: e.g. the anonymization of one file containing 10 million rows took 50 seconds.