EVL Data Generation

EVL Data Generation Microservice provides fast, automated and cost-effective method for data generation. Having proper test environment is a must in many areas like application development, implementing ETL processes, or stress testing. Often the tests can't be done on existing production data so data simulation is the only way how to achieve the goal. The simulated data have to comply with the real-life data patterns and data volumes should be close to the expected peaks.

- High productivity due metadata driven approach
- Custom domain lookups can be used
- Ability to include custom made data generation functions
- Parallel running of jobs and workflow monitoring

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 JSON and Excel files, databases—Oracle, Teradata, PostgeSQL, etc.—and streaming data like Kafka.

Configuration File

Based on a configuration CSV file, data generation jobs are generated together with a workflow to manage many targets easily.

Following table shows an example of such configuration file, say `crm.csv`, which would generate an Oracle table `accounts` and a file `cust.csv`:

Src	Entity	Field	Data type	Null	Datagen Type	EVL Value	Description
ORA	accounts	id	int	0	RANDOM_INCR		Unique incr. ID
ORA	accounts	cust_id	int	0	RANDOM _LOOKUP(cust.csv)		Customers from lookup
ORA	accounts	iban	string		RANDOM_IBAN		Valid IBAN
ORA	accounts	currency	string		RANDOM _LOOKUP(curr.csv)		Random currency
ORA	accounts	score	decimal(8,2)	0.8			Random value
ORA	accounts	valid_from	date				Random date
ORA	accounts	valid_to	date			random(*out->valid_from+1, *out->valid_to+3650)	Must be greater than valid_from
FILE	cust.csv	id	int	0	RANDOM_UNIQ		Unique ID
FILE	cust.csv	email	string		RANDOM_EMAIL		Email structure
FILE	cust.csv	person_id	string	0		random_rc()	Sum = 0 mod 11

Where credentials, connection strings, paths, etc., are set in a separate configuration file shared for one such configuration CSV file.

Datagen Type – this field contains either EVL predefined or custom defined aliases to any EVL functions.

EVL Value – for specific needs, like dependency on other fields (e.g. generated `valid_to` must be always greater than `valid_from`), any EVL code can be used. In very specific cases, like Czech and Slovak Personal ID number, which needs to fulfill divisibility by 11, custom C++ function can be used as well, like `random_rc()` above.

Null – the percentage of generated NULL values can be specified. Zero means that field is not nullable at all. Values between 0 and 1 then means the approximate percentage of generated NULL values for given field. Empty means that given field is nullable, but default percentage is used.

Min/max – there is also possibility to specify minimal and/or maximal generated value. (Not mentioned in above example.)

Build and Generate Data

Either from EVL Manager (graphical web interface) or by running in the shell:

```
$ evl datagen build configs/crm.csv
$ evl run workflow/datagen/crm.ewf
```

will generate two jobs (one for `accounts` table and one for `cust.csv` file) and a workflow with these two jobs and run the workflow to generate the data.

Exact number of generated records can be defined in a separate file, unless default number is used.

Data Generation Types

Here is the list of default Data Generation types, i.e. functions to be used to generate the data.

Type	Data type	Description	Example

`RANDOM or <empty>`	any	generic randomization, with min/max range	for string: `uisC7dsSacs`
			for date: `2001-12-14`
			for decimal: `-125001.44`

`RANDOM_INCR`	integers	keep the uniqueness	`1, 2, 3, 4,...`
`RANDOM_UNIQ`	integers	keep the uniqueness	`45582, 656, 97110198, 872,...`
`RANDOM_EMAIL`	string	generate emails	`UlfsTb@sFux.3t,...`

`RANDOM_IBAN`	string	keep IBAN validity	`ES91 2100 0418 4502 0005 1332,`
			`NL91 ABNA 0417 1643 00,...`

`RANDOM_LOOKUP(file)`	any	randomly from a `file`	`Richard, Donald,...`

RANDOM_INCR and **RANDOM_UNIQ** types produces the output in a unique way, the first one incrementally, the second one in non-ordered way. Particularly useful for IDs.

For detailed information see `https://docs.evltool.com/evl-data-generation`.