EVL Validation Microservice

EVL Validation Microservice enables quick, easy-to-use and cost-effective validation of datasets. It is very useful in situations where complex automated testing tools may be too heavy and expensive. Good candidates for the validation tool are ETL or migration projects or quick quality checks of production data.

EVL Validation advantages

- Configuration via Excel or CSV files with pre-configuration option based on metadata
- Automatic data type and null values validation
- Setting of other validation criteria for e.g. number intervals, string lengths
- Possibility to add complex validation functions for entity relations
- Separating "wrong" data and logging of rejection reasons
- Setting conditions for breaking the job flow based on percentages, number of rows ...
- Fast implementation and rapid change management
- Low implementation and operating costs

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 files, databases–Oracle, Teradata, SQL Server, etc–and Hadoop streaming data like Kafka and Flume.

EVL Validation functions

All functions return NULL when input is NULL.

Data	Function	Description
String	String Length	Min/max string length
Any	Null value check	Check nullability of a field
String	Code page	Identifying characters with wrong code page
Date	Date interval	Setting Min and Max date interval
Date	Date format	Identifying non-standard date and time format
Number	Number interval	Setting Min and Max interval for integers, floats, and decimals
Specific	Entity relation check	Relations checking between 2 or more attributes
Specific	Validation function	Calling validation functions for complex conditions

EVL Validation project

A validation project consists of following steps:

1. unzipping EVL distribution and defining a few variables and paths
2. filling-in an Excel or CSV file defining source type (e.g. csv, Oracle, ...), table or file name and field names and validations functions to be applied
3. automatic generation of EVL jobs for each entity
4. running EVL jobs in a batch or individually
5. displaying rejected files containing wrong records and logs

Example

Following example shows an implementation of a very simplified validation.

Set variables:

```
# Source and target data directories
DATA_SOURCE_DIR="/some/path/source"
DATA_ANON_DIR="/some/path/validation"
```

Anonymization definition for file TEST:

Src	Entity	Attribute	Datatype	Null	Min	Max	Validation condition	Description
csv	TEST1	ID	int	No	0	5000		Setting number interval
csv	TEST1	ACC	int	No			ID > 500 && ACC > 3000	Attribute relation
csv	TEST1	RC	string				check_rc(RC)	Calling custom function
csv	TEST1	Sex	string		1	1	substr(RC,2,2)>50 && Sex=="F"	T est of sex validity based on RC
ORCL	TEST2	ID	string	No				String length check
ORCL	TEST2	Postcode	int		5	5		Postcode must be 5 digits
ORCL	TEST2	Text	string				Codepage: ISO-8859-2	Checking characters co-depage

Run:

```
#  generating evl jobs from the config file
evl run/generate_jobs.evl

# running the validation job for an entity "TEST1"
evl run/validate.test1.evl

# running the validation job for an entity "TEST2"
evl run/validate.test2.evl
```

Example of custom function rc_check():

```
stol(replace(RC,'/','')) % 11 == 0
```