7-09-2016

Kees Mandemakers and Fons Laan

Valorisatie LINKS: Ontwerp exportsysteem error-meldingen uit LINKS

1 Introduction

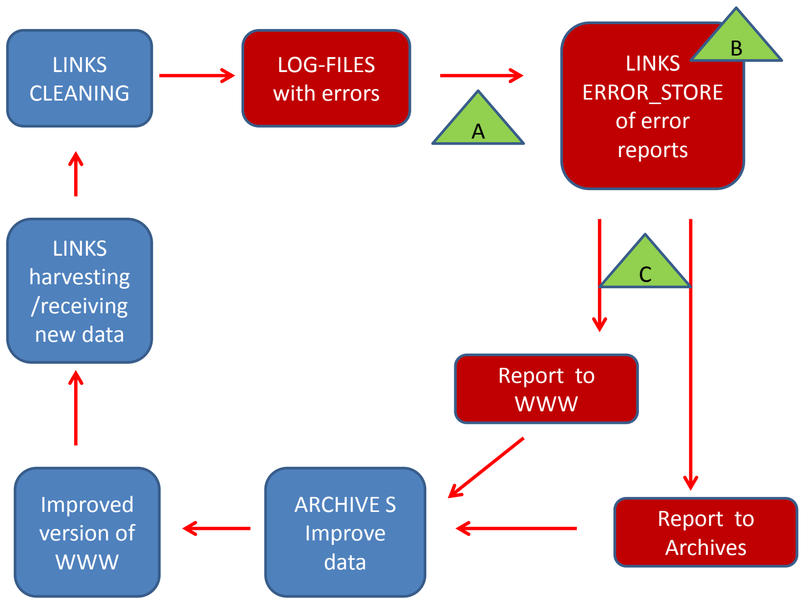
With the cleaning of the LINKS data each value is tested on validity and internal consistency. This results in many reports about serious and less serious problems in the data delivered by *WieWasWie*.

The goals of the valorisation-project is to report all problems to *WieWasWie* or the involved archive with a clear description of the error and the registration from which the error originates. *WieWasWie* will bring the reports to the archives with the goal that they will improve their datasets.

Section 2 describes the existing data structures that form the base for the migration of the data. Section 3 gives an overview of the logistical part of this project. Section 4 contains the table that keeps a time stamped overview of all delivered errors to the archives. But it is not the intention to overflow the archives with 'errors' especially for relatively light problems. Next step is that the archives will improve their data, resulting in qualitatively better original data in *WieWasWie*, resulting in a better dataset at the start of the LINKS process, ending in better matching results.

For a schematic overview of the whole process, see Picture 1.

Picture 1 Schematic overview of the error-traject.



2 Data structure

2.1 Log files

The reporting of the errors as such is based on two tables: a) a basic table in which each type of error is described and b) log files that register all discovered errors during the cleaning process of the data. The fields of these tables are described in the LINKS functional design, chapter 1.2 and 1.3. For a copy of these tables, see Appendix A.

Essentially the logfiles describe the content of the error, a code of the type of the error, the name of the archive and an exact description of the place where the error can be found in the register of which a specific value originates.

There are two types of errors: Values that are not valid and values that are inconsistent in combination with values from other fields. A large part of the tests on valid values is done by way of reference tables. The result of a comparison with a value in a reference table is handled by way of four codes:

y	The value is valid, no further action
x	The value is not evaluated yet
n	The value is not valid and not interpretable
u	The value is not valid but it was possible to standardize the value

All values coded with ‘x’, ‘n’ or u’ are written to the log files but only values coded with an ‘n’ or ‘u’ will be returned to the archives, especially the ones with value ‘n’.

Every type of error has its own error code which is –as said- written in the log file together with a general typing of the error. This indication has two values: ‘FT’, which means that the value is complete wrong or invalid or ‘WA’which warns that the value could be wrong and/or is wrong but will probably not hamper the matching process.

2.2 Storage files

A new table will be built for the storage of the errors that are originating from the logfiles.

The name of the table is ERROR_STORE and has the same record structure as the log-file (see Appendix A) with three extra fields:

T	Flag	Flag to indicate the status of the specific error report 0 No indication of status (not used in the described processing) 1 Not urgent 2 To be send 3 Sent to an archive
D	Date_export	Date of the moment that the error is exported
T	Destination	Name of the institution to which the error has been sent

2.3 Exported data

The tables with the exported errors are of *csv format*.

The name of these tables is constructed from three variables (to be found in the ERROR_STORE table): Source_Maintype_Date.csv

Source is provided by the field *id_source, Maintype* stands for the maintype of the source, e.g. GEB, HUW and OVL and *Date* shows the date of the creation of the table.

The tables will contain an export from the ERROR_STORE table with the following fields:

N	id_log	Primary key (autonumber)
N	id_source	Secondary key to the archive
T	archive	Name of the archive where the errors originate from
T	location	Location of the registration (original value)
T	reg_type	Type of the registration (original value)
T	date	Date of the registration (original value)
T	sequence	Number of sequence of the registration (original value)
T	role	Role of the person in the registration (original value).
T	guid	GUI identifier of the registration in which the problem occurs.
T	content	Content of the report

The identification of each error is the result of the combination of the fields *id_source, reg_type, date (year), sequence number* and *role.* The field *guid* was meant as a persistent identifier for the certificate. Since the distribution of these persistent numbers was not done at the archives themselves, it is not sure how consistent these identifiers are. In other words there is a big chance that one and the same certificate has got (or will get) more than one persistent identifier.

3 Work process

The work processes are organized by way of three scripts (see picture 1), that run from the LINKS servers in the subdirectory c:\...\links_logs\export_reports.

A The collecting and importing the log files.

The first script will import the date from the logfiles into the tabel ERROR_STORE. This table is situated in the subdirectory c:\...\links_logs\export_reports.

This script will select the logfiles by way of the date range (the date is included in the name of the logfiles).

In principle all errors will be exported except the ones codes with an ‘x’. These are the records with the following values in the field *type*:
21, 31, 41, 51, 61, 71, 81, 91, 141, 251, 1009, 1109

The deletion of logfiles in the directory c:\...\inks_logs will be done manually.

At the moment of exporting the script will check if the record is already included or not. This will be checked on the combination of the following fields:

id_source
location
reg_type
date
sequence
role
content

Only not included records will be added to the system.   The value of the flag will be put at ‘1’.

B The selection of data to be distributed to the archives.

This script selects the errors that should be distributed to the archives in the short run. The script will change the value of the field *flag* into 2 records in the table ERROR_STORE for those errors that have to be handled in the short run.

The selection of error types that will be handled in the short run is handled in a manual way by changing the list in the script.

Deleting of records in the table ERROR_STORE will be done manually. Cleaning is a repeating system and it is no point to keep all reports in the storage. However, it is best practice to keep the records that have been sent to the archives for at least five years.

C Exporting the data

In this script the data are selected by testing on the field flag. All records with value 2 will be included in the export_files. The script continues with the construction of exportfiles for each archive and setting the field *flag* on 3 for each exported record and putting a date in the field *Date_export*. For the name and content of the file, see section 2.3.

The files will be sent in a manual way (for the moment). We will expect improved data back within a time span from one to five years. So, after five years existing reports in the storage may be deleted and fresh reports may be welcomed by the system.

Administrative files showing e.g. the frequency of errors per archive will be created in a manual way.

Appendix A Table structures from the LINKS cleaning programme.

LOG

The table LOG contains all reports that are generated by the functions of LINKS as a whole. For the inventory of all possible types of reporting the following table REF_REPORT.

The LOG tables have the following structure:

N	id_log	Primary key
N	id_source	Secondary key to the archive / source
T	archive	Name of the archive where the sources originate from Content of [SOURCE]![archive]
T	location	Location of the source Content of [REGISTRATION_O]![registration_location]
T	reg_type	Type of the source [REGISTRATION_O]![registration_type]
T	date	Date of the source [REGISTRATION_O]![registration_date]
T	sequence	Number of sequence of the source (registration) [REGISTRATIE_O]![registratie_volgnr]
T	role	Role of the person [REGISTRATION_O]![role]
T	guid	GUI identifier of the registration in which the problem occurs.
N	reg_key	Primary key of the record in table REGISTRATION_O in which the problem occurs [id_registration].
N	pers_key	Primary key of the record in table PERSON_O in which the problem occurs [id_person].
T	report_class	Basic classification of the report, see table REF_REPORT
T	report_type	Specification of the report in code, see table REF_REPORT
T	content	Content of the report
D	date_time	Date and time of the report

The fields id_source, archive and role are not working properly:

Id_source	It seems that no source number is included if the cleaning is including more than one source, that is not correct.

Archive	The name of the archive is included in the table sources, this could be implemented in a very late stage
Role	This field can be implemented by way of the table Ref_Role which presents the name of the role for each role number.

REF_REPORT

The table REF_REPORT offers a list of all possible messages that our generated through the converting, parsing and linking process.

N	id_report	Primary key
T	class	Basic classification of the report, values like: SY System report about the process FT Error (or most probable an error) WA Warning for possible error
N	type	Specification of the report in code, values like 11 [meaning: Family name not included in reference table] 21 [meaning: Age in weeks is over 200]
T	content	Content of the report, values like [11] Familiy name: <[Family name]> not included in reference table [21] Number of age in weeks is above 200, weeks: <[weeks]> Within brackets the content causing the report record..

Values ending with a 1, 3, 5 en 9 are reserved for reference tables and **within the range *31 – 99*** have a specific value.

@1	Standard does not exist yet (standard_code='x')
@3	Original value is not valid (standard_code='n')
@5	Original value is not valid but it was possible to arrive at a valid standard (standard)_code='u')
@9	The value at standard_code in the reference table belonging to s apsecific original is not valid (other character than 'y', 'n', 'u' of 'x').