Flexter
How it Works

sonra
The Data Liberation Company

Sonra Intelligence Ltd.
6-9 Trinity Street,
Dublin 2
Ireland
CRO Ref 547986
www.sonra.io

Sonra Intelligence Ltd.
6-9 Trinity Street,
Dublin 2
Ireland
CRO Ref 547986
www.sonra.io

sonra
The Data Liberation Company

1. What is Flexter?

Flexter is a smart data parser that liberates data from the shackles of complex and proprietary file formats such as XML or Excel and makes it available in a an easy to digest format for business users.

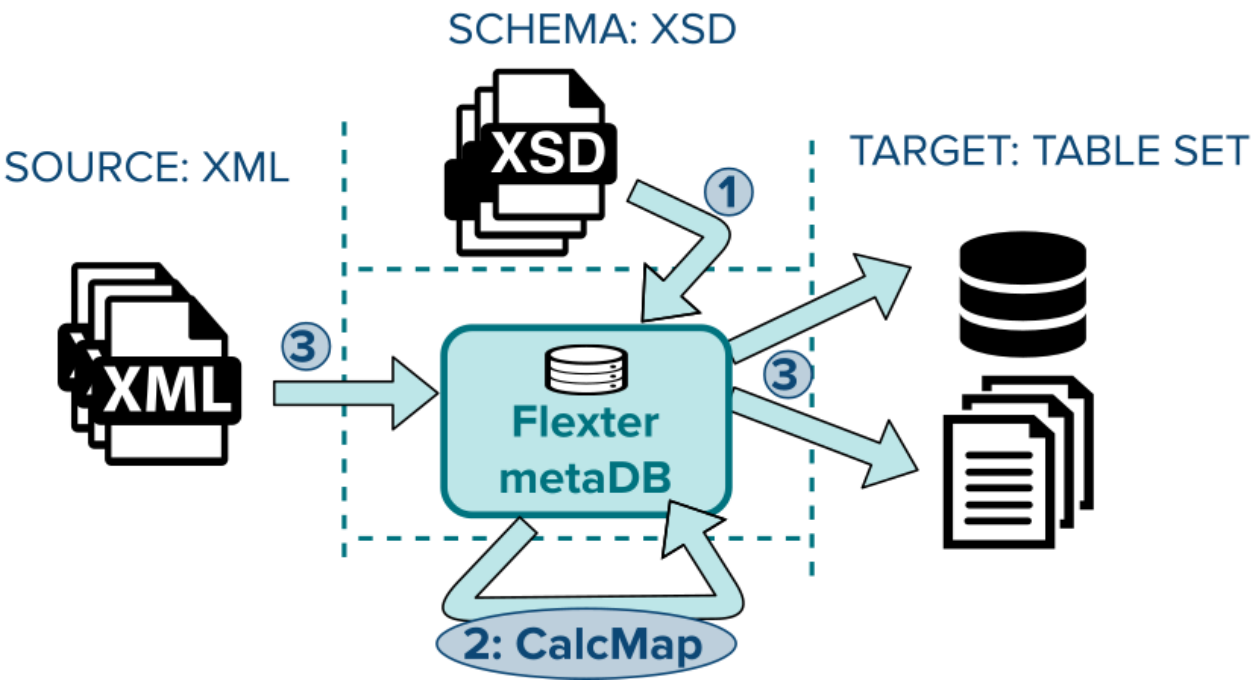
Existing data integration tools are not fit for purpose for processing large volumes of complex XML files:

- Processing XML files with these tools is a very manual, risky, and labour intensive development process. The risk of failure and the number of developer man days increases exponentially with the complexity of the XML files.
- Existing tools don't scale. They rely on a single server architecture and don't scale out to multiple servers. This limits the amount of data volume that can be processed.
- Processing XML files is very demanding on hardware resources. Existing tools again and again loop over the same XML files to extract all data. This is very inefficient.

2. How it works

Flexter platform consists of three functional modules:

- schema analyser (**xsd2er**)
- mapping generator (**CalcMap**)
- xml processor (**xml2er**)



4.1. Schema Analyser

Schema Analyser is a dedicated module that loads, parses out, processes and stores the XML schema information in *Flexter's* internal metadata DB. This step is only required to be performed once for each schema to be processed.

The schema analyser can be invoked using the command line tool **`xsd2er`**

Below are the logical steps executed by *Schema Analyser*:

Parses out XSDs

The XSD (XML Schema Definition) file prescribes how the XML files need to be built. It is an XML file itself and its information needs to be parsed (read and interpreted) to extract the schema information. Quite often and XSD schema is built in a modular and references other XSD files as children. All the relevant files need to be read into memory first to get the complete view of the schema.

Links definitions

The schema definition specification is generally strongly based on typing where a more complex type's definition refers to one or more basic types to specify its own layout. This implies multi-level inheritance that needs to be resolved to discover the complete information about each complex type's hierarchical structure. For each specified XML element, including the root element, the Schema Analyser needs to follow the full inheritance path to find out about its final structure.

Renders the map

Once the definitions are linked together it's possible to build the complete information on the allowed layout of the source XML. We call this process "rendering of the XML schema definition". At the end of this process a complete XML map is built that contains all information on the elements, their hierarchy, the attributes and their datatypes.

Stores in the internal metadata

The resulting XML layout map is then stored in the *Flexter's* internal metadata DB for reuse.

4.2. Mapping Generator

Now that we know the exact layout of the source XML it is possible to generate the relational equivalent.

Flexter's module, *Mapping Generator* generates the output schema layout and the mapping to it. Various optimisations of the target schema can be applied during this step.

The *Mapping Generator* executes the following steps:

Load XML layout from the metadata

Connects to internal *Flexter* metadata DB to retrieve the complete map built by the *Schema Analyser* module

Analyse the layout of the XML

Identifies identical elements, conflicting names, cardinality of the parent-child relationships

Apply requested optimizations

Based on the selected optimizations the schema type definitions are pre-processed, grouped, merged, and linked. This produces an intermediate result where for each input datapoint (XPath) the association to its target endpoint (table & column) is stored.

Convert to relational equivalent

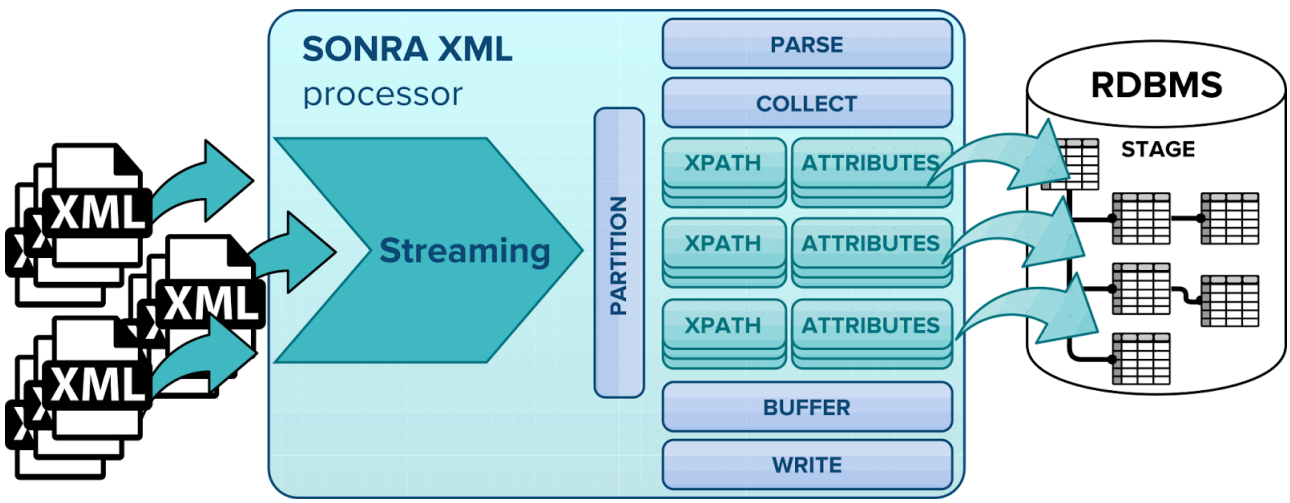
Various fields are added to make the transitions to relational format possible (primary-foreign keys, self-contained elements as table columns). Also naming conflicts are resolved at this stage. We also generate a source to target map that maps the data lineage from the source elements to the target columns and tables.

Store in internal metadata database

The information about the relational layout and the mapping is stored into *Flexter's* metadata DB for subsequent use.

4.3. XML Processor

The XML Processor module takes the information generated from the two previous steps, processes the XML and writes the data to the relational target schema.



Sonra Intelligence Ltd.
6-9 Trinity Street,
Dublin 2
Ireland
CRO Ref 547986
www.sonra.io

The *XML Processor* executes the following steps:

Load metadata

Connects to internal *Flexter* metadata DB to pull the complete map of XML built by *Mapping Generator* (source layout, target layout & mapping)

Loads XML data

Loads all available XML data files (file, archive, zip, Excel container)

Parse, group, partition & filter

Parses XML data into attributes and elements and caches it in memory.
Based on the mapping information it knows the destination target structure for each data point and partitions the whole dataset accordingly so that the write process can happen in parallel.

Writes out the output

Each writing process uses a filter to locate data intended for the same target. Subsequently it connects to the target (jdbc/file) and buffers the data before ultimately writing it into the target in the batch fashion.