American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)

ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

http://asrjetsjournal.org/

Alternatives for Storing and Validating XBRL Data

Ivan Belev*

University of National and World Economy, ul. "8-mi Dekemvri", Studentski grad, Sofia 1700, Bulgaria

Email: i.belev@unwe.bg

Abstract

The purpose of this article is to explore the alternatives for storing and validating data in Extensible Business Reporting Language (XBRL) format. The article gives a brief overview of XBRL and the most common components, the history of XBRL and the latest trends in XBRL including Inline XBRL. Simple examples are provided in order to describe the XBRL format and structure and Inline XBRL. The author builds on his previous research and professional experience in the area of XBRL and different XBRL software tools and concepts to outline the best solutions for solving the research problem - storing and validating XBRL instances. The research is significant because it analyzes combined information from different XBRL related sources and the author's practical knowledge to reach a conclusion how to store and validate data in XBRL format.

Keywords: XBRL; eXtensible Business Reporting Language; Inline XBRL; store; validate.

1. Introduction

In the development of information technology and computer science one can say that the key driver is and has always been – data. Data was always in the center of every problem that information technology and computers tried to solve. And as the progress moved on, so did data evolve, becoming bigger, more diverse and more complex. Nearing the end of the second decade of the twenty-fist century we can say without any doubt that data is practically everywhere. The type of data that is most important for this study is the financial and business data in and across organizations.

---

* Corresponding author

According to some common definitions [1], financial or business data is data that relates to the financial health of a business. Some common types of data according to the same source are:

- Assets data;
- Property data;
- Liabilities data;
- Equity data;
- Risk related data;
- Shares and stocks data;
- Etc.

The financial and business data in the organizations became more complex over time as all types of data did. But for this type of data there was one key requirement that changed the game – it became subject to different kinds of regulation and had to be reported. In most cases reporting regulations came from outside the organization – by the government or other higher authority. In other cases, reporting regulations came from the organizations itself, especially for large holding organizations and multi-international companies. Different detail of data became subject to reporting in different time periods – annually, quarterly, monthly or even on a more frequent scale. Combining this requirement with the ever-growing amount of data, some authors [2] stated that the key for business organizations to succeeding in such an environment is to undergo the process of digital transformation. According to other sources [3] public organizations such as government structures, use digital transformation technologies as well. The key moment that this article is focused on is the evolution and digital transformation of financial reporting, driven by the emergence and further development of the Extensible Business Reporting Language, abbreviated as XBRL.

2. Extensive Business Reporting Language (XBRL)

2.1. XBRL brief overview and history

The acronym XBRL stands for Extensible Business Reporting Language. The creation of XBRL was made possible after the World Wide Web Consortium published the XML (Extensible Markup Language) in the year 1998. XBRL is one of the many XML based languages that uses similar language syntax. The invention of the language was most credited to an American accountant by the name of Charles Hoffman [4]. In the same year that XML was first published, Hoffman created the first financial statements using XML tags and structure. The idea was to create a common and independent syntax that can be used to describe various types of reported data with different data structure. The first step was further developed by other organizations and people until the first XBRL specification was released in 2000. Like XML, XBRL became an open standard and is currently managed by the XBRL International Consortium. The last stable version at the time of this article (October 2019) is 2.1, published in 2013.

Today XBRL is mainly used for two reasons:

- By companies that are subject to regulation by a higher authority organization - both the company and

the authority organization use XBRL as a format for data exchange;

- For data exchange between two or more companies, usually when part of a group or holding structure.

Using XBRL syntax for the data exchange makes it possible for the exchanging parties to have a common understanding of the type of data that is reported, regardless of the specific structure of the data. This is achieved by combining several different XBRL components.

2.2. Key elements of XBRL

All the XBRL components are described in detail in the official XBRL specification [5], published by the XBRL International Consortium. Two key components are necessary if a company is to be able to create a report in XBRL format. The first component is the XBRL instance file (also called XBRL report) and the second component is the XBRL taxonomy package. The XBRL taxonomy package defines the different business items that can be reported under the same taxonomy package version, as well as all the necessary submission details. A taxonomy package is created for each type of reporting. Every taxonomy package has a specific type and version that define all the details about the data facts, that can be reported. It can be said that the XBRL taxonomy acts as a common dictionary for all the companies, that report data under the same reporting regulation. The facts that can be reported are described in the XBRL taxonomy package as concepts. The taxonomy packages are available publicly for all the companies so they can use it when creating or reading the XBRL reports. In most cases the taxonomy packages are complex and include relations to other taxonomies using different types of references. The whole set of interrelated taxonomies is referred to as a Discoverable Taxonomy Set (DTS). The DTS combines all the concepts for a specific type of reporting. For example, insurance companies in the European union can report the concept “Income/gains and losses in the period” under the taxonomy term code “ei189” (according to the corresponding taxonomy – Solvency II version 2.4.0, supervised by the European Insurance and Occupational Pensions Authority – EIOPA)[6]. The taxonomy package includes an element with the name “ei189” that contains additional attributes like:

- Identifier;
- Substitution group;
- Period type;
- Nullable flag;
- Element type;
- Enumeration values reference link.

The reporting companies, also called entities, prepare files in XBRL format with the data, that is reported for a specific period. These files are called XBRL instance documents. XBRL instance documents provide the actual reported data. Every XBRL instance document references the specific type and version of XBRL taxonomy (DTS). Every reported fact in the instance document should reference a term (concept), that is described in the corresponding taxonomy package. XBRL instances can have a lot of different elements, described in the XBRL specification [5], but the most common elements include contexts, items, tuples and units.

One of the key elements in an XBRL instance document is the "context" element. The "context" element most commonly includes information about:

- The reporting entity (company) - identified by an alfa-numeric identifier;
- The reporting period – a point in time (instant) or a time interval with start and end dates;
- Optional reporting scenario – used to describe whether the data shows actual values, budget estimations, etc.

The "unit" element is used to specify type of measure for facts, that can be measured – for example, the specific monetary currency. The actual reporting values are described in XBRL instance documents using the "item" or the "tuple" elements. Facts that can be expressed by a single value are reported using "item" elements (for example – Net income), while rows of data are reported using "tuples" (for example – company assets). Each reported "item"/ "tuple" references a specific "context" and "unit" by their identifier in order to describe the reporting company, period, scenario and unit of measure for the reported fact. Having in mind the previous example about reporting "Income/gains and losses in the period" for an insurance company, the XBRL instance file should contain an "item" element that references at least:

- The taxonomy concept "ei189", defined in the DTS, referenced by the id "s2hd_ei189";
- A "context" element, defined in the same XBRL instance file, referenced by its name;
- A "unit" element, defined in the same XBRL instance file, referenced by its identifier.

Here is how this element could look like as a line of XBRL code in an XBRL instance document:

<s2hd_ei189 contextRef="context_189" decimals="2" unitRef="euro_units">1900000.00</s2hd_ei189>

2.3. From XBRL to Inline XBRL

The adoption of XBRL as a reporting standard solved problems, that reporting entities and regulators were facing. One of the most significant problems was the lack of common reporting format, which lead to the following resultant problems:

- Before XBRL each of the reports had diverse format and data structure. Additional tasks ware required to standardize the formats and data structures that took time and effort, while facing short deadlines;
- Reports were difficult to validate, and many times had to be filed repetitive times until the authority approved the data;
- There was not much room for automation of the reporting process.

Reporting in XBRL provided a solution for those problems but lead to other difficulties for the reporting entities and regulators. Because of its complexity, XBRL is designed to be processed by computers. The format is very hard to be interpreted by the human brain without the help of a so called XBRL processing engine. This led to the inability of the reporting users to understand the reported data after it has been converted to XBRL format.

The solution to this problem came with the introduction of Inline XBRL, also referred as iXBRL. Inline XBRL was created in 2010 and first published in 2011. Like XBRL, iXBRL is an open standard, managed by the XBRL International Consortium. Inline XBRL provides human readable reports in the form of a HyperText Markup Language (HTML) documents with embedded XBRL markup. Reporting in iXBRL format provides a single report that is at the same time understandable by humans and XBRL computer processing engines. The key differences between XBRL and Inline XBRL are summarized in Table 1 [7]:

Table 1: XBRL and Inline XBRL – key differences [7]

Key Feature	XBRL	Inline XBRL
Markup is readable by:	Computers only	Computers and humans
Instance file extension:	.xml / .xbrl	.html / .xhtml
File Encoding:	XML	XML and XHTML
Requires XBRL Reader:	YES	NO
Visual formatting capabilities:	Very Limited	Flexible
Markup complexity:	High	Lower

Following the same example from 2.2, an Inline XBRL instance file can provide visualization of the reported concept “Income/gains and losses in the period” in a HTML table row using the combined HTML and XBRL construct in Figure 1:

```
<table>

<tr><th>Fact</th><th>Value</th></tr>

<tr>

<td> Income/gains and losses in the period:</td>

<td><ix:s2hd_ei189 name=" s2hd_ei189" contextRef="context_189" decimals="2"
unitRef="euro_units">1900000.00</ix:s2hd_ei189></td>

</tr>

</table>
```

Figure 1: XBRL and HTML code in Inline XBRL

3. Storing and validating XBRL instances

3.1. Existing problems with XBRL storing and XBRL validation

The process of XBRL report submission usually undergoes the following steps:

- The reporting entity gathers the required reporting data as per the reporting requirements for the entity type, reporting type and reporting period;

- The data is transformed into XBRL or Inline XBRL format;
- The reporting entity validates the report before submission and stores the data for reference purposes;
- The reporting entity sends the XBRL/iXBRL report to the respective authority;
- The regulating authority validates the XBRL/iXBRL submission and stores the data for additional analysis.

The described submission process poses some problems, related to data handling, validation and data storing. The first problem in terms of process steps is related to the mechanism for gathering the required data for the XBRL report. The author's experience in the field of consulting companies for XBRL preparation shows that reporting entities usually have numerous data sources that provide the different parts of data that are necessary for the creation of the XBRL report. Some data may even be available only in Excel files. This creates difficulties for implementing an automated solution for extracting and preparing the data for the XBRL report. Another problem occurs at the step for validation of the XBRL report. XBRL data needs to be valid in more than one way:

- XBRL report structure and syntax – the XBRL report should conform to the XBRL format in terms of structure and syntax;
- XBRL logical validations – the XBRL report should meet numerous logical validations that cross reference different bits of data from different parts of the XBRL instance document (for example – The sum of investments for the period, reported as a single summary element should equal the sum of all investments, reported as multiple values in another part of the report).

Out of the two validation scenarios, the second one poses a far bigger problem due to the high diversity and large number of logical validation rules that need to be performed. For example, the European Insurance and Occupational Pensions Authority (EIOPA) [6] demands the validation of more than 200 logical rules for the submission of XBRL instances by the pension funds in Europe under Solvency II regulation. For insurance companies the number of logical validation rules is more than 1300. Table 2 describes a few example validation rules that aim to show the different level of complexity among validation rules:

Table 2: XBRL Solvency II validation rules example [6]

Rule ID	Validation Rule Formula	Validation Rule Description
BV118	{c0090} - data type constrains [ISO 8601: (yyyy-mm-dd)]	Data type should be ISO 8601
BV102	{c0010}={c0040}+{c0050}	The "Total" value reported in S.23.01 - Own funds should be equal to the sum of "Tier 2" and "Tier 3".
BV125	{S.02.01, r0750,c0010}+{S.02.01, r0760,c0010}+{S.02.01, r0780,c0010}+{S.02.01, r0840,c0010}+{S.02.01, r0860,c0010}+{S.02.01, r0870,c0010}+{S.02.01, r0880,c0010}={S.02.02, r0170,c0020}	The item "Any other liabilities" reported in template S.02.01 - Balance Sheet should be equal to sum of the same items in template S.02.02 - Assets and liabilities by currency.

The described validation problems can be observed on the receiving side as well – the regulating authority. Since the regulator will not accept XBRL instance documents that are not valid in all possible ways, there should be an automatic validation algorithm in place at the point of receiving the XBRL report. After the report has been validated successfully, the regulating authority is facing yet another problem – to store the XBRL instance document in a structured format, that enables the consecutive analysis and summarization of the data. The described validation, storage and retrieval processes of the XBRL instance data should not be error prone. On the contrary, those processes should be fast and reliable, so that the reporting deadlines are met.

3.2. Alternatives for XBRL storing and XBRL validation

The previously described problems can be solved using different technical and practical approaches. When it comes to gathering data and building an XBRL instance document based on that data there are a few different scenarios that the reporting entities could employ:

- When XBRL was first introduced as mandatory reporting format for different reporting purposes, most of the regulators first offered a solution to support the companies for the initial XBRL preparation phase. There were different software tools that were developed and offered to the reporting companies, so they can start building their first XBRL reports. For example, in the period 2015-2017 the European Insurance and Occupational Pensions Authority (EIOPA) developed a tool called Tool for Undertaking (T4U) [8] to support the initial phase for Solvency II reporting. The tool provided a user interface for input of data in a table form that can later be exported to XBRL instance file. The T4U tool supported a few taxonomy versions before it was discontinued;
- Other regulators provided web interface for their subsidiaries to input data manually and produce XBRL output;
- After the initial phase some of the reporting entities gained expertise in XBRL and developed in-house solutions for the generation of XBRL instance documents. The main problem with this alternative is not the initial development of the solution, but the update of the developed solution with every new taxonomy version. As it turns out, taxonomies are updated on a regular basis due to changes in the requirements of the regulating authorities;
- Other companies went on to buy an XBRL aiding software solution that would provide them with the means to generate XBRL instance documents as well as updates of the solution for every new taxonomy version and change. According to the author's experience in this area, this alternative proves to be the most convenient for the reporting entities. First of all, the so called XBRL processing engines are not at all expensive for large companies (some of them can cost less that 1000 Euro per year). Secondly, most of these solutions come on a subscription plan that includes future updates and fixes for new taxonomies. Besides that, the vendors for XBRL solutions usually work very closely with the regulators in order to provide on time updates prior to the reporting deadlines.

Following the options for the reporting entities for generating XBRL instance documents described above, the author can summarize the results: Using XBRL solutions, provided initially by the regulators (first two options above) is not a very good alternative because most of those solutions are being discontinued after the initial

reporting phase is over and lack support and taxonomy updates. Developing an in-house XBRL generating solution (third option above) is also not considered a good option by the author, because such a solution demands high efforts to keep it up to date after taxonomy changes and updates to the validation rules. Such changes and updates are not at all rear events and happen almost every reporting period. The last option is considered best by the author, because outsourcing the XBRL processing engine to an XBRL solution provider at a relatively small cost ensures security and mitigates the risks for ensuring the taxonomy and validation rules update in time for the next submissions. Moving further in the XBRL submission process, the next problems that were previously described are related to the validation of the XBRL instance data. Those issues are closely related to the format, that the instance data is stored in, because this is a key factor to the speed of the validation procedures that are carried out on the provided data. The syntax that is used to describe the validation rules as executable checks strongly depends on the form XBRL data is stored in:

- One alternative is storing the XBRL data in a relational database structure with separate tables for each reported template. This approach is employed by some of the XBRL processing engines (for example, it was used by the EIOPA T4U tool, as well as a widely used open source XBRL platform called Arelle [9]). The author's observation with this approach is that the processing speed for storing the data in the relational database and for carrying out the validation checks can be very low, depending on the amount of data. Also, every change in the taxonomy and the validation rules implies changes across the whole implementation - from the data model to the storing and validation procedures. The taxonomy package should also be a part of the database layer and would imply the same problems after a taxonomy update. Carrying out validations in this scenario means to execute multiple queries, selecting data from many tables and comparing the received results to the benchmark values;
- Another way for dealing with the validation and storage problem is by using a relational database in such a way that a part of the previously described problem is avoided. This can be achieved with a relational database, structured in a non-transactional schema, for example, a star (or snowflake) schema that is not used for transactional data but is more inherent to Business Intelligence and Data Warehouse models. This approach eliminates the need to create new tables and fields with every change of the taxonomy. The changes are done only on the data side, but not on the structure of the database model. The author's experience with this alternative shows better performance for storing the data, but an even higher level of complexity related to the validation rules processing. This approach doesn't eliminate the problems with storing the taxonomy package versions as they are not convertible to the described database model.
- A new way for providing better performance for storing and validating XBRL instance documents with no regard to the actual structure of the output is by ensuring in-memory functionality. The XBRL instance data is loaded in-memory, which provides very fast processing times for validation and storage procedures. This approach certainly gives better results according to the author's experience. Some software vendors use this approach regardless of the actual data model. This approach is even recommended by the XBRL International Consortium for processing large XBRL instance documents [10].
- With all the different approaches for storing and validating XBRL instance documents, the author

achieved best results using an XML database store.

The solution is proposed by a whitepaper from Oracle and UBmatrix [11] and uses Oracle's XML DB features to store the instance data using the native XML-based syntax. This empowers the query and validation procedures to use XML-based search capabilities that are executed very fast. The model was proposed in 2009 and allows for storing the taxonomy package in the same manner. The XBRL processing engine provider developed further this solution to include options for Microsoft SQL database platform. Combining XML DB features and In-memory processing model becomes a top candidate for being the best alternative for storing and validating XBRL instance data. The author draws the following conclusions about the best alternative for storing XBRL instance and taxonomy data reviewing the options, described above: The first option to store XBRL data in a traditional relational database model, suited for transactional data is not regarded by the author as a good option, because of the complex and slow database queries for processing validation rules. Import of data is relatively fast because different type of XBRL data is stored in separate database tables, but validation rules need to select data from many different tables. Also, the author worked with such a database model for a few years and experienced large workload when a taxonomy package and/or validation rules had to be updated. Many database tables and predefined rules had to be changed. With the second option the database changes with new taxonomy versions are less because of the specific database model that is used, but the star and snowflake schema models are still not suitable for the taxonomy descriptions and validation rules. The author considers this option to be better that the first one, but still it has major drawbacks with the processing of validation rules. Providing in-memory processing of the data input, output and validation procedures (third option above) proves to speed up the processing times. This alternative is considered by the author as a good one, but the author considers best to combine it with an XML database store. In the author's experience, using an XML-based database store with in-memory functionality achieved best results regarding ease of management and processing times of input, output and validation rule checks. Storing the XBRL instance data and corresponding taxonomy packages requires no need to change the database structure. Data extraction and validation rules are carried out using XML native operations and are very fast when compared to database SQL queries.

4. Recommendations

Reviewing the results in the previous section of this article the author defines the following recommendations:

- For the generation of XBRL instance data, the author recommends the reporting entities to use software solutions from XBRL solution providers, provided on a subscription basis. This way the reporting organizations will receive at a relatively low cost a reliable solution with included updates for every change of taxonomy and validation rules;
- For storing and validating XBRL data, the author recommends the use of XML database store in combination with in-memory processing. This way the processing speed is very fast and practically almost no changes to the database model are necessary for the adoption of new taxonomy versions and changes in the validation rulesets. Storing taxonomy data is achieved in the same way.

5. Conclusion

This paper combined the review of XBRL related sources and the personal experience of the author in the field of XBRL to give an overview on XBRL and Inline XBRL and to assess the alternatives for storing and validating XBRL instance data. The author has years of experience in working with XBRL with multiple XBRL processing platforms as well as in developing custom XBRL solution components, related to storing and validating XBRL instances. Based on the research the author concludes that at the time of this article the most effective XBRL processing scenarios include the use of XML database features in combination with in-memory processing.

6. Future Scope

The author sets future scope of analysis, related to bringing more detail to the research by conducting more precise measurements to quantify the extent of superiority of the proposed solution over the other options.

References

[1] S. Grimslay. "What Is Financial Data? - Definition & Concept Video", Study.com: https://study.com/academy/lesson/what-is-financial-data-definition-lesson-quiz.html, Jan. 24, 2018 [Oct. 20, 2019].

[2] S. Yordanova, K. Stefanova, 2019, "Major technologies and practical aspects of the digital transformation of business in a big data environment", Business Management, D. A. Tsenov Academy of Economics, Svishtov, Bulgaria, issue 1 Year 20, pages 5-24.

[3] P. Milev, 2018, "Opportunities for Implementing Internet Monitoring in Public Organizations", Economic and Social Alternatives, University of National and World Economy, Sofia, Bulgaria, Issue 2, pages 58-69, April, ISSN 1314-6556.

[4] S. Roohani, "What is the History of XBRL?", XBRL Education: http://www.xbrleducation.com/edu/history.htm, 2008 [Oct. 20, 2019]

[5] Extensible Business Reporting Language (XBRL) 2.1, XBRL.org: http://www.xbrl.org/Specification/XBRL-2.1/REC-2003-12-31/XBRL-2.1-REC-2003-12-31+corrected-errata-2013-02-20.html [Oct. 20, 2019]

[6] EIOPA Solvency II Directive, EIOPA: https://eiopa.europa.eu/regulation-supervision/insurance/solvency-ii [Oct. 20, 2019]

[7] What is iXBRL? Differences between XBRL and iXBRL, DataTracks: https://www.datatracks.co.uk/ixbrl-blog/what-is-ixbrl, Mar. 20, 2012 [Oct.20, 2019]

[8] XBRL Tool for Undertakings (T4U), EIOPA: https://eiopa.europa.eu/regulation-

supervision/insurance/tool-for-undertakings [Oct. 20, 2019]

[9] Arelle – Open source XBRL platform, http://arelle.org/ [Oct. 20, 2019]

[10] Notes on the Processing of Large XBRL Instances 1.0, XBRL.org: https://www.xbrl.org/WGN/large-instance-processing/WGN-2012-10-31/large-instance-processing-WGN-WGN-2012-10-31.html [Oct. 20, 2019]

[11] UNLOCKING XBRL CONTENT, An effective database solution for storing and accessing XBRL documents, An Oracle & UBmatrix Whitepaper, Sep 2009, https://www.oracle.com/technetwork/database/database-technologies/xmldb/unlockingxbrlcontent-130016.pdf [Oct. 20, 2019]