Preserving databases using SIARD

Case Study 2

Experiences working with large databases and their preservation

v1.0 Review version

CEF eArchiving Building Block, E-ARK3

CEF-TC-2019-3 eArchiving

Cover Sheet

Document Status:

Status

Document Approver(s)

Name	Role

Document Reviewer(s)

Name	Role

Summary of Changes:

Version	Date	Created by	Short Description of Changes
0.1	09.07.2020	Team	
0.9	23.09.2020	Team	Summary added

1.0	30.09.2020	Team	ready to review

Experiences working with LOBs outside of SIARD-files

Introduction

This document covers the handling of LOBs in SIARD. It focuses on the various ways to preserve LOBs along with SIARD snapshots and how to handle large LOBs.

What are large objects (LOB)?

One of the main challenges when working with database preservation using SIARD is the handling and storage of large binary objects (LOBs). In database management systems, Large Objects (LOBs) are a set of data types that can hold large amounts of data. A LOB can hold up to 128 TB depending on how the database is configured. LOBs enable application access and manipulate the data efficiently. LOB data can be semi-structured (for example an XML file) or unstructured (for example image as a binary file). LOBs include both binary large objects (BLOB) and character large objects (CLOB).

Most current information systems and databases still operate in a hybrid manner – while some of the data is held within the database as “native” relational data, other pieces of information are stored as binary files (text documents, images, drawings, etc.) either within the database or next to it (i.e. referenced from the database).

Agencies produce more and more data, and the volumes in the databases increase - and the SIARD archives increase correspondingly in size. This is caused by a large number of LOBs, large LOBs, or a combination of these. This can become problematic and create a need for segmentation of LOBs and/or large tables.

Alternative methods for preserving LOB

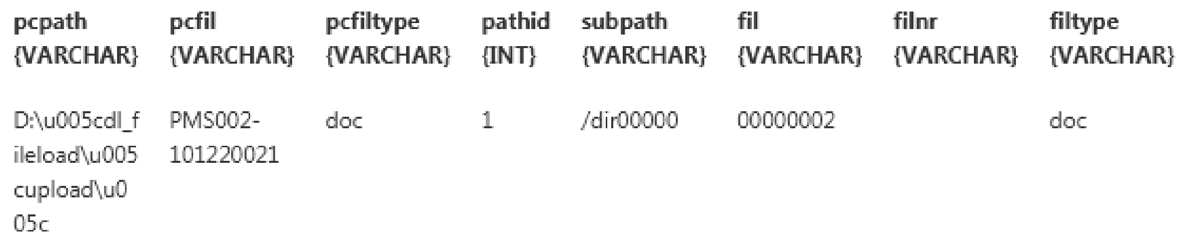
The current version of the SIARD standard (SIARD 2.1) allows LOBs to be stored both internally and externally, inside the SIARD file. It is also possible to have them stored outside the SIARD file. The SIARD 2.2 standard (currently on review) shows how segmentation shall be carried out.

Preserved in SIARD-file

Large objects can be stored inline in the table[number].xml file, internally as separate file entries inside the SIARD archive, or externally as standalone files in the file system. LOBs inside the SIARD archive is not part of the scope of this document.

Preserved outside SIARD-file

Many choose to store the files separately, i.e. LOBs stored outside the SIARD file. This topic is not handled in the current version of the standard. The National Archives of Norway (NAN) have exclusively received SIARD-databases with LOBs outside the SIARD files. The screenshot below shows part of a file reference table and is an example of this. It shows a database with file references split into several columns. In this database, the file path is built up from subpath + fil + filtype. So the file would be found at \dir00000\000000002.doc.



pcpath {VARCHAR}	pcfil {VARCHAR}	pcfiltype {VARCHAR}	pathid {INT}	subpath {VARCHAR}	fil {VARCHAR}	filnr {VARCHAR}	filtype {VARCHAR}
D:\u005cdl_fileload\u005cupload\u005c	PMS002-101220021	doc	1	/dir00000	00000002		doc

As part of the handling and validation process at NAN, these file references are updated to ease accessibility. This is done with a script that concatenates the content in the three columns and stores the result in a new column in the relevant table. The column contains file references as strings.

Even though the unified file references are a step up compared to the original file references, this solution raises some issues, because they are not machine-readable. The SIARD 2.2 standard (currently on internal review) suggests that these kinds of file references are stored as an XML node with additional metadata connected to the file, such as checksum, file size, file location, etc.

Archiving databases with many LOBs

In this chapter, we present the challenges encountered when archiving the Estonian Buildings Registry. The registry was archived by the National Archives of Estonia (NAE) in 2019, and it included:

- a 400 GB database;
- 198 tables and 1098 relations;
- 9 TB of external LOBs (binary documentation on buildings – permits, pictures of the construction process, drawings, etc.);
- geographical data (coordinates of buildings) stored within the database using the Oracle-specific SDO_GEOMETRY format.

In the following, we describe in more detail the initial planning, testing, setup, migration, and transfer of the Buildings Registry.

Planning

The process of archiving the Estonian Buildings Registry started with the planning phase where the (digital) archivists of the National Archives of Estonia analysed the registry to understand it's technical setup and characteristics.

The analysis of LOBs revealed that there was only one field in one table, which was connected to the external file system and included links to external LOBs. However, the data type used in this field was the Oracle-specific BFILE which was not supported by DBPTK, meaning that it was not possible to include the external LOBs into the SIARD snapshot by default. Two potential solutions were identified:

- ask the database administrator to convert all BFILE file paths into VARCHAR format, and create a whole new database field for the converted values;
- ask the developers of DBPTK to add BFILE support natively into the software.

NAE opted for the second option for two main reasons. The first reason was the integrity and reusability of views and queries. Introducing a new field for the file path means that ideally, you analyse all the above for the use of file path information in BFILE format. If found, you have to modify the view or query as well by replacing the BFILE field with the VARCHAR field. The second reason was that Oracle is widely used in Estonia, and it is very likely that the same BFILE issue will be encountered several times in the future. As a result, native Oracle BFILE format support became available within DBPTK in autumn 2019.

A similar problem was identified with the geographical data stored in the Oracle SDO_GEOMETRY format. The interest of the NAE was to archive all geodata in the open GML format, but SDO_GEOMETRY to GML conversion support did not exist within DBPTK. Again the two potential solutions were either to ask the database administrator to do the necessary conversion or to update DBPTK. For the same reasons as mentioned above NAE opted for the further development of DBPTK, the SDO_GEOMETRY - GML conversion became available also in autumn 2019.

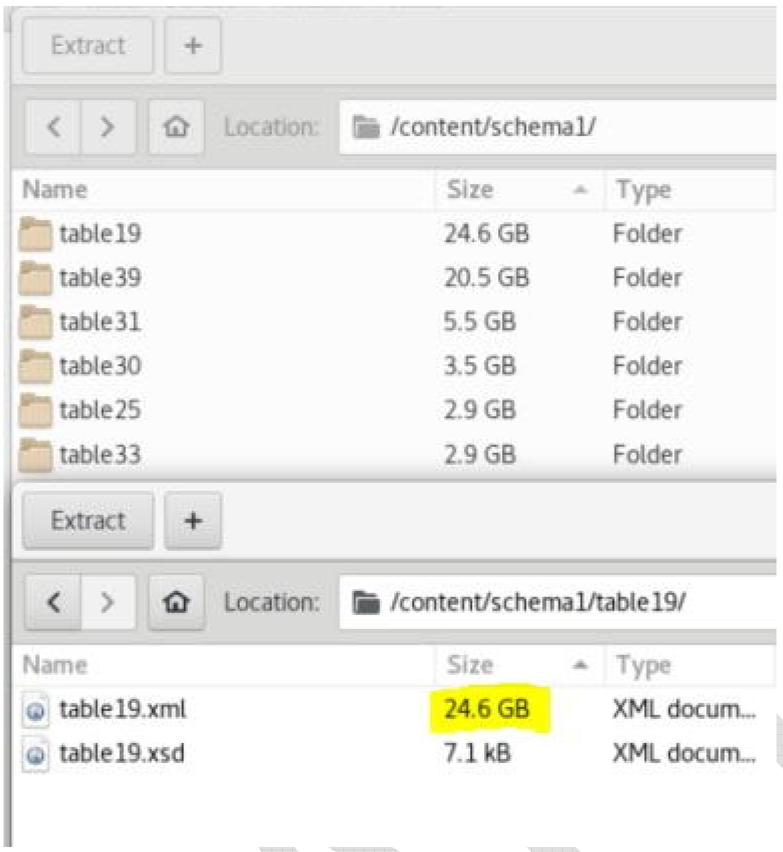
Initial technical testing

The approach of the National Archives of Estonia is to start technical testing as soon as possible and to carry out tests on actual data as much as possible. This approach is deeply rooted in our experience that the (few) technical issues you might get in database archiving are most efficiently discovered when running the snapshot creation in practice, as opposed to having long theoretical discussions about "what might go wrong".

In the case of huge databases, the execution of "quick and dirty tests" is problematic. The main issue is the demand for resources – time for running the SIARD snapshot creation (many days), and the storage space needed to store the resulting SIARD file (more than 10 TB in the case of the Buildings Registry). As well, we want to automatically validate each test snapshot using DBPTK, and this process might again take many days.

In 2019 NAE had no other option than to create a clone of the initial database using infrastructure (servers and storage) which were specifically acquired for the archival process. As there was the need to run multiple tests, this infrastructure had to remain in place for many months, creating a considerable financial cost.

Another issue became apparent when starting to analyse the SIARD data in detail. Because larger tables within the database consisted of millions of rows, the resulting XML files were also huge (see image below). This was an issue when the validation of the SIARD file prompted errors or warnings in the snapshot. In the validation report, every warning and error refers to the content so to understand the issues the XML files had to be accessed. We did not find any software that could open a 24GB XML file, so the only way was to split it into smaller parts. For example the Linux command xml_split -s 10m -b row -n 3 table19.xml creates 2461 files from table19.xml. After breaking up the file, the grep command can be used to find the correct 10-megabyte piece which can be opened with any XML or text editor.



Extract +

Location: /content/schema1/

Name	Size	Type
table19	24.6 GB	Folder
table39	20.5 GB	Folder
table31	5.5 GB	Folder
table30	3.5 GB	Folder
table25	2.9 GB	Folder
table33	2.9 GB	Folder

Extract +

Location: /content/schema1/table19/

Name	Size	Type
table19.xml	24.6 GB	XML docum...
table19.xsd	7.1 kB	XML docum...

Based on these experiences, NAE proposed to add the possibility of partial SIARD creation into DBPTK. The idea behind this proposal was as follows:

- most errors encountered during SIARD are not unique to a single row but occur on most rows of a table;
- when limiting the number of rows exported from tables to a representative number (for example, 1000) the export still has to deal with all the complexity of the tables, data types, etc.;
- thus, when limiting the number of rows we can still do the export (and can validate the export) on real data, and expect that we will encounter all technical issues (for example, with handling references to external LOBs or with geodata formats);
- we get a structurally complete SIARD snapshot, can validate it and ultimately identify and solve technical issues very quickly (hours as opposed to days);
- we can open individual XML tables stored within the SIARD snapshot and do a deep analysis of the data without special splitting;
- we do not need extensive and costly infrastructure for testing purposes (only for the actual migration -- see next chapter).

This partial SIARD creation option was added into DBPTK in 2020, meaning that now you can add SQL-based “where” clauses to the import-config module to specify the number of records included to the test-migration. For example, the tables referring to external LOBs (or just containing millions of records) can be cut to only a few rows so that time for testing will be short. However, the result will still be a functioning and validatable SIARD file.

Setup

This step covers the setup of the infrastructure (hardware) needed to create and store the SIARD snapshot of the database. This step also includes the setup of a separate copy of the full database. For large databases, the time needed to create a SIARD snapshot can be many days. It is essential for the integrity of the archived snapshot that the database content remains the same throughout the process, i.e. that no data is added, deleted, or modified. In most cases, this can only be achieved when establishing an exact copy (i.e. a clone) of the database specifically for SIARD creation.

The main discussion during the setup phase was how to calculate the amount of storage needed. Agencies typically do not have tens of terabytes of free space, meaning that the cost of hardware can quickly become a significant factor within the whole process as every terabyte can greatly increase the cost of whole migration. Often it is quite hard to predict the final storage amount needed for the database snapshot. It would be good practice to set up a separate copy on a file system that resides on some logical volume (Logical Volume Management), so it can be easily increased by adding hard drives when needed, and no processes need to be restarted.

As mentioned above, this dedicated infrastructure was set up even before testing in the case of the Estonian Building Registry and remained available throughout the multiple months of creating and validating various test snapshots, the official migration, transfer, and acknowledgement by the NAE. This was a significant cost factor for the transferring agency, in future large-size transfers NAE intends to only require this for the final migration. This should reduce the need for specific hardware to some weeks (not months).

Migration

The fourth step was to carry out the official migration to produce the SIARD snapshot to be transferred to and stored at the National Archives of Estonia. The migration consisted of two steps: the creation of a full SIARD snapshot of the whole database, and the creation of an access-oriented SIARD representation of the database consisting of selected materialised views and queries as individual tables.

The main process took about 28 hours to execute with DBPTK, and the result was a snapshot that consisted of a main SIARD file (size 77 GB) and 18,311,003 LOBs stored outside the SIARD (size 9.2 TB). However, there were also three unsuccessful tries to create the snapshot, meaning that it took about a week in total to create the full SIARD snapshot.

One of the key lessons learned from the failed attempts was the importance of having a clone where absolutely nothing is done with the data for archival purposes. For example, the first unsuccessful try was because the host of the cloned system forgot to turn off automatic maintenance scripts which started running at around midnight and stopped the SIARD creation (which had been running for more than 12 hours.). Another example is about having “active triggers”. These are internal procedures which activate when reading information from certain tables or queries and write information into other tables, effectively destroying the integrity of the created SIARD snapshot. The worst thing about the “active trigger” issue is that it does not stop the SIARD creation and does not provide any errors or warnings; you can only discover the error after validation. In this case, even the NAE staff did not understand the reason for validation errors for quite some while, because the message we got was about a mismatch in private and foreign keys of some tables. Only later we understood it was because triggers added new lines into the database during the migration.

Another problem that we introduced ourselves was about defining the field size threshold for LOBs. Namely, DBPTK allows users to set a value for LOB extraction, all fields which are larger than the value will be stored outside of the main SIARD file as an external LOB. Unfortunately, at NAE, we set this threshold too low, all geodata fields (i.e. GML fragments) were larger than the threshold and got stored as external LOBs. As there were millions of fields with geodata in the initial database, the migration created millions of individual small files (less than 1 KB). This error was not noticed in time by NAE staff and resulted in further problems during the transfer which we discuss below.

The second step of the migration, creating SIARD from materialised views, was very straightforward and took less than an hour.

Transfer

In the final step, we transferred the SIARD snapshot to NAE and carried out the final validation. As mentioned above the whole snapshot consisted of millions of individual files, many of which were small (less than 1 KB). As we choose to transfer the snapshots using sFTP the number of small files turned out to be a real problem. Due to the overhead of transferring many small files, the whole transfer took about a week, despite the two locations (agency and archive) being connected through high-speed networks. This experience taught us to package content (using zip, tar, or similar) before starting any data-transfer as moving many small files is very time-consuming for FTP-clients.

Summary

This study continues from Case Study 1, where national archives share their overall experiences with using SIARD for archiving relational databases. In this study, we focus explicitly on the problems and solutions around the archiving of databases which include large objects, for example, text files, videos, drawings, or photos. Usually, such files are located outside the database, and columns of the tables only contain links (file paths or some other IDs) to the corresponding files. To a lesser extent, these large objects can also be stored within the database, formatted as binary or textual data types (BLOB, CLOB, etc.).

The key findings and recommendations on handling database archiving in the case of large objects are:

- The most important activity when archiving large databases is to map all the columns related to LOBs and estimate the total storage amount needed for database snapshots.
- If the size of LOBs is large, then it is reasonable to extract and store these files outside the SIARD snapshot.

- SIARD creation should be tested long before the actual migration to reduce the resources needed for the whole project.
- Initial testing may take only a few rows from each table, but the results will show how the SIARD tools can handle different databases and file system architectures. For example, those tests can indicate that some columns with file paths or base paths may need to be converted or some folders in the file system have to be remounted.
- We suggest validating all test SIARD files.
- It is recommended to create a separate “clone-copy” from the database only for archiving, and check that nothing changes in that clone during the migration of the SIARD snapshot.

This case study explains the process of creating database snapshots using the Database Preservation Toolkit (DBPTK). We regard DBPTK to be a good tool for creating, validating, and handling SIARD files from large databases with LOBs. Our experiences show also that the upcoming version of the SIARD specification is very promising to get even better support for Large Objects inside or outside the database.