Using data archiving tools to preserve archival records in business systems – a case study

Neal Fitzgerald
GLAMATEK
Brisbane, Queensland, Australia
neal.fitzgerald@gmail.com

ABSTRACT
The preservation of archival records from government business systems is a pressing concern for archival institutions worldwide. Most business systems developed over the last 20 years do not have in-built recordkeeping functionality. Archivists and records managers face the task of identifying, extracting and preserving archival records from these systems. I use a public authority collection management system as a case study to explore how currently available archiving tools might form part of a practical method to identify, extract and prepare digital archival records for ingestion into a digital preservation archive.

Keywords
Digital preservation, SQL Server, databases, business systems.

1. BUSINESS SYSTEM ARCHITECTURE
The most common business system architecture is an application layer built on top of a commercial relational database layer. The database layer holds the system data in related tables and usually some code in *stored procedures* and *triggers* that perform database access and update functions. The application layer is made up of code modules that contain the business rules, manage workflows and generate user interface screens. It presents the data in different ways, summarizes it and produces reports. It turns the data into information. You need the application layer to make full sense of the database layer [4].

Database management systems use Structured Query Language (SQL) to access and manipulate data stored in rows and columns of related tables. Databases are designed to minimize redundant or repeated data. It is common to use codes to link to common values stored in separate lookup tables. Repeating fields and transactions are also decomposed into separate tables. *Views* are database objects that store a single SQL query. They are often used to bring back together decomposed elements of common system entities. They act like virtual tables that can be referenced in database and application code.

2. RECORDS IN BUSINESS SYSTEMS
Generally business systems are not recordkeeping systems. Data is fluid and changing. Historical data is often overwritten to reduce storage costs and keep the system running efficiently so it is hard to reconstruct the system state at any point in time. Historical data that is maintained is often not tamper proof [4]. Historical reports, summaries, snapshots or extracts created by the business system are often held outside the database. These could be printed on paper, or held in a file system, data warehouse or eDRMS (electronic document & records management system).

In the IT world a record is a row in a database table. To archivists records are information created, received and maintained as evidence and information by an organisation or person, in pursuance of legal obligations or in the transaction of business [ISO 15489]. I use the word in this sense throughout this paper.

In Queensland, a *public* record is any form of recorded information, either received or created by a public authority, which provides evidence of the business or affairs of that public authority [5]. Public records may need to be retained permanently or expire after a fixed period (the *retention period*).

The International Council on Archives has developed *Guidelines and Functional Requirements for Records in Business Systems* to make design suggestions for new systems and as a way to review recordkeeping functionality in existing systems. The guideline states that to identify records as evidence we need to:

1. Determine the broad business functions and specific activities and transactions carried out by the business system.
2. For each function, activity and transaction or business process managed by the system, consider what evidence is required to be retained by the organisation.
3. For each requirement for evidence, identify the content or data that make up the evidence.

Records might consist of a number of inter-related data elements connected across one or more tables [3].

3. THE PROBLEM
A recent Queensland Government ICT audit identified a number of business systems for decommissioning. Government wants to reduce the cost of software licensing, hardware maintenance and specialist skills. Systems containing data with ongoing business use will be kept running or virtualised until the data is archived, migrated or no longer required. Systems with data that is no longer currently used are to be switched off as soon as possible. Agencies need to identify the records in the business systems that have not expired and are still within their retention period. If all the records in the system have expired, then the systems can be switched off without further action. For systems with unexpired or permanent records, agencies need processes and tools to extract these records in a format that can be preserved and rendered for long term access.

4. TOOLS & APPROACHES
A common approach to preserving business system records is to export the contents of all of the tables in the database in an open XML format. If records must be deleted after their retention period expires (for example some criminal records), unwanted data must be deleted from the database before archiving or from the archive package after archiving. Tools like RODADB and SIARD that use this approach have functions to load the XML archive to a different SQL database platform to allow ongoing

access over time, but this requires knowledge of the database structure and SQL query skills.

Commercial database archiving software tools like HPAIO and CHRONOS [1] are primarily designed to purge data from large transactional databases to reduce storage costs and improve performance. They use a similar export-all-tables approach for retiring business systems, but they also have functionality to assemble 'data objects' (and so archival records) from their constituent columns and tables and extract these in XML format. If we preserve these archival records, users do not need knowledge of the database structure or SQL skills to access them.

Database warehousing software is used by some agencies for enterprise level reporting, trend analysis and data mining across data assets from different business systems. It may be possible to leverage this software to extract and preserve records [4].

In a recent blog post [6] State Records of NSW discuss a number of methods of preserving and presenting the information in business systems to suit different classes of users. A searchable collection of pdfs might suit family researchers. An open source SQL database might suit agency IT staff to re-create reports from an archived system. RDF linked open data might suit a researcher wanting to create visualizations of the data.

In business systems applications, records as evidence of business transactions will often be presented to the user on a single screen or a set of related screens. At the National Archives of Sweden the 'preservation object' in a business system is documented with a screen shot, a mapping of the screen fields to database table columns, and the corresponding SQL query [2]. If the assembled archived records in XML format can be rendered in a form similar to the original business application screen including the field values, screen labels, field ordering and grouping, this would provide a human friendly way to present the records for access and also provide a visual check of the record accuracy.

5. RECORD PRESERVATION MODEL

I propose a four stage preservation model illustrated in Figure 1.

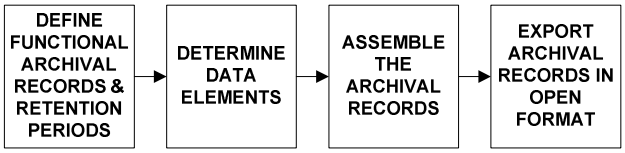

Figure 1. Preserving archival records in business systems.

1. Identify the archival records by analyzing the functions of the system that assist decision making or document business processes or actions and determine their retention periods.

2. Determine the data elements that document the system transactions performed to fulfill these functions by reviewing system documentation, application code, database structures and user interface screens and interviewing knowledgeable staff.

3. Assemble the data elements to produce a consolidated representation of the archival record.

4. Extract and export the consolidated records whose retention periods have not expired in an open format.

6. THE CASE STUDY

This case study uses an agency collection management system. It has a staff module for maintenance of the collection catalogue and a web based search interface for public access to the collection.

6.1 Defining the functional archival records

The agency has a retention and disposal schedule, which specifies the agency archival records held in the collection management system at a high level and their retention periods. One set of records is the digitized image register which I use as an example. The retention and disposal schedule specifies that the metadata describing the images needs to be preserved until superseded.

6.2 Determining the data elements

6.2.1 Reviewing the application screens

My starting point for determining the data elements making up the image register records was the corresponding enquiry screen in the application staff module. This screen showed descriptive metadata, staff who created and approved the image, and virtual exhibitions in which it appears.

6.3 Assembling the archival records

A 19.7Gb backup of the Microsoft SQL Server database layer was restored into a Microsoft SQL Server 2008 R2 installation on a Dell Optiplex 990 PC with Intel core i5-2400 @3.1GHz with 8Gb of RAM, running the Windows 7 64-bit operating system.

I used the SQL Server Management Studio tool to construct an SQL query to model the image register archival record. As I proceeded I compared the query results against values displayed for the example images on the image register enquiry screen.

6.3.1 Reviewing the database

The database has over 300 tables. Table and column names are generally descriptive. There are no entity relationship (ER) diagrams and no declared foreign keys, so I looked at database and application code for clues to the table relationships.

Reviewing the table names, I found a candidate main table for the image register record query with column names matching the screen labels and two large object binary columns containing jpeg images. These images are access copies and do not need to be preserved in the record. Preservation TIFF versions are kept elsewhere on the file system.

A simple SQL query on this table returned some values matching those on the screen and some codes. I found a stored procedure that displays image metadata for the public web interface. It contained an SQL query that showed joins to some of the lookup tables. I added these tables to my record query and the results matched the screen except for the staff and exhibitions data.

6.3.2 Reviewing the application code

I found the application module that produced the image register screen. It contained somewhat cryptic and fragmented Visual Basic code that constructed an SQL query. It used the same joins to the lookup tables found in the database layer stored procedure. The code also showed the joins to the staff lookup and exhibitions tables. I added these joins to the SQL query and it returned all data as displayed on the image register screen.

6.3.3 Creating the archival record table

I created a *view* object to document my query and used the query to make a new database table with all the elements of the record.

Tables in relational databases have 1-to-1, 1-to-many or many-to-many relationships. Many-to-many relationships are usually decomposed into two 1-to-many relationships via a chaining table. The relationship between image register table and code lookup

tables are one to one. A single image code field corresponds to one value in the lookup table. The relationship between the image register and the exhibition tables is many-to-many. We can represent this by two 1-to-many relationships. An image may be in more than one exhibition. An exhibition has a number of images. The image screen shows the image details and lists all the exhibitions in which it appears in a scrolling window and the exhibition screen shows the exhibition details and lists its images.

When tables with a 1-to-many relationship are joined in an SQL query, the result is a *Cartesian product* of the two tables. In our example, if an image occurs in a number of exhibitions, there will be a row for each exhibition and the image data will be repeated with each row of the result. Because images rarely appear in multiple exhibitions, the output from the record query in this case is not significantly larger than the size of the original tables.

In other cases there may be a large number of child rows for a large number of parent rows. An assembled record table could be orders of magnitude larger and add significantly to the archive size. I used the SQL Server XML functions with my query to create XML with a single copy of the image data for each image and nested exhibition elements. With the full dataset the query failed with a memory error, common in my experience on a number of platforms. An alternative is to use XSLT style sheets after archiving to create hierarchically structured XML.

6.4 Exporting archival records

I was able to download trial versions of SIARD, RODADB and HPAIO. CHRONOS staff demonstrated their software by WebEx. The case study database had to be prepared to allow the tools to connect by enabling TCP/IP, opening ports, starting services and enabling SQL Server authentication.

6.4.1 HP Application Information Optimizer[1]

HPAIO was formerly known as HPDBA. HPAIO version 7.02 is complex to install and operate. It archives data as XML or CSV documents. It can export these documents to HP's TRIM eDRMS. It is quite complex to install and operate and has a lot of functionality not required in our decommissioning use case.

The HPAIO Designer tool lets you create *models* of *business objects* using a visual drag and drop interface to join the database tables. You select a driving table and add lookup, chaining or transactional tables, essentially creating the equivalent of an SQL query. You can select a subset of columns in each table, rename table columns and add selection conditions. A pdf document can be produced listing the components that make up the model.

I used this tool to model the image register archival records using the tables and joins discovered during the previous analysis. HPAIO produced a nested hierarchical XML. The extract failed for the full record set with a memory error. HPAIO embeds binary image files within the XML, which would hinder monitoring these embedded objects for format obsolescence over time.

6.4.2 CHRONOS

CHRONOS[2] database XML export format is simple and compact. It shows table structures, the queries in view objects, and the code in stored procedures and triggers. Each table is stored in a compressed CSV zip file with checksums stored for each row. It can export a *view* object as if it was a real table, assembling the data and creating a CSV file. If we create a *view* from our archival record this gives a simple record extraction method. Exporting the records as tables with the more compact compressed CSV format rather than XML reduces the impact of the Cartesian product problem described in 6.3.3 above. CHRONOS has user access control, functionality to support WORM storage devices and SHA-512 encryption to increase security and prevent data tampering. We hope to have a pilot installation to test soon.

6.4.3 SIARD / SIARDK

The Swiss Federal Archives publishes the SIARD[3] (Software Independent Archiving of Relational Databases) open standard database archiving XML format. They provide free tools to export SQL databases to SIARD XML and tools to import the SIARD XML into various database management systems for access.

The SIARD archive package is a ZIP64 packaged hierarchy of folders and files. The *header* folder contains the SIARD schema, a *metadata.xml* file (describing the database tables, views and stored procedures), and an XSL style sheet for viewing the package contents in a web browser. In the *content* folder there is a folder for each table with the data in *table.xml* and a schema describing the table structure in *table.xsd*. ZIP64 allows for large packages whereas the ZIP format is restricted to 4Gb.

I installed SIARD version 1.50. The tool is java based and easy to install and use. The database was converted to an 18.7 Gb SIARD ZIP64 package in about 4 hours. There is a graphical user interface for inputting the connection parameters and initiating export and import tasks. The graphical user interface allows browsing of SIARD archive packages. It also allows extra descriptive metadata to be added to the package.

SIARD stores large object binary elements such as large text blocks, images or video as separate files in the archive package referenced from the XML. SIARD does store the contents of database views, but did not store any code from stored procedures, only their name and parameter declarations. Potentially valuable information about database structure and system function is lost.

6.4.4 RODADB

The RODADB command line tool has been derived from the database ingestion and deployment components of the RODA digital preservation repository[4]. It exports the database table definitions and table content for all the tables in the database in DBML XML format. The software also allows DBML to be converted to a set of SQL statements that can be used to re-create the tables in another database platform for access. I downloaded RODADB version 1.1.1. It is java based application and easy to install. The command line syntax is straight forward. RODADB also stores the XML and large object binary images separately. The output is a folder containing a single XML file with pointers to the image files in the same folder. The case study database export produced a 21Gb XML file and about 90,000 binary image files (12Gb) in about 3.5 hours.

---

[1] http://www8.hp.com/au/en/software-solutions/software.html?compURI=1175612#.UXTEHyJMj4-

[2] http://www.csp-sw.de/en/inhalt.php?kategorie=c271_CHRONOS

[3] http://www.bar.admin.ch/dienstleistungen/00823/00825/index.html?lang=en

[4] http://www.keep.pt/produtos/roda/?lang=en

The DBML contains only table and key definitions and table data. Views and stored procedures are ignored. The lack of a graphical user interface might discourage some users. The DBML XML file is significantly larger than the corresponding SIARD XML files, but more directly readable, because it spells out the table column names for each row.

7. RECOMMENDATIONS

We should preserve the database content in a format that can be rendered in a number of ways for presentation. This way we can satisfy the needs of different classes of users from members of the public who want a Google like search and easy presentation through a web browser, to agencies who want to manipulate archived data in a similar way to when the business system was still functioning, to future researchers who want datasets to mine, visualize and mash up with other datasets [6].

The preservation effort expended on any business system will depend on many factors including legal risk, information value, available skills, available resources and historical importance.

For business systems being decommissioned:

- Use archiving tools that produce XML
- Create a full XML export of database tables
- Preserve as much contextual documentation as possible
- Identify and assemble archival records before archiving
- Document the archiving process
- Optionally preserve the original database layer backup file

XML is an open, text based format that does not require specialist software to be rendered, so has a good chance of remaining accessible in the long term. It can easily be transformed using XSL style sheets to allow human friendly display formats for access, and the creation of open data sets for data mining and visualization. There are tools to load SIARD XML and RODA DBML to a number of SQL database platforms for access by those with SQL query skills.

A full database export will decrease the risk that important information is lost unintentionally. In some cases some data may be mandated for deletion and some duplicate or ephemeral data may need to be deleted before archiving to save storage costs.

Contextual documentation might include user manuals, screen shots, application and stored procedure code, database ER diagrams, application architecture or UML diagrams, retention and disposal schedules, records of interview with IT staff and expert users. System reports and summaries produced by the business system in the past may be useful artefacts.

Identifying and assembling archival records from their constituent columns and tables in the database before archiving will aid future accessibility. The process is an opportunity to gather together, synthesize and crystallize information from contextual documentation, application code, database layer code and expert knowledge. The dataset modeling tools in HPAIO can be used to achieve this. If using SIARD or RODADB, an SQL query can be used to assemble the record elements into a new database table before archiving. The developed query can also be documented and stored as a *view* object in the database layer. Some systems will have better documentation than the case study with database ER diagrams, declared foreign keys and views or stored procedures that correspond to the archival records. The process may be easy and straight forward.

If storage permits the original database backup file could also be preserved, so the archiving process itself can be verified while the database software and operating environments are still available.

Documenting the archiving process in detail will give future researchers confidence that the archived records are a true representation of the original records in the business system.

If budgets are tight and a simple solution is needed, I would recommend SIARD as the initial tool of choice. If possible, spend time to identify and assemble the elements of the archival records in new tables before archiving as SIARD XML.

Of the commercial products reviewed, CHRONOS looks very promising. It exports more elements of the database in more compact and open way. If an agency implementing the system has a requirement to archive records and other data from currently functioning business systems, the investment in licenses and learning how to use the product would be worthwhile.

8. FUTURE WORK

I propose to further test and refine this work using business systems implemented on other database platforms. I will further experiment with the tools examined here and with other open source and commercial data archiving and data warehousing tools including CHRONOS. I will explore using XSLT style sheets to transform the XML exported from the case study database using the tested tools. Use cases will include rendering human friendly access versions of the records, removing unwanted data, creating hierarchical XML representations of assembled archival records that have repeating data, preparing datasets for migration to a new collection management system, creating Submission Information Packages to ingest into a digital preservation archive.

9. REFERENCES

[1] Brandl, S., Keller-Marxer, P. 2007. *Long-term Archiving of Relational Databases with Chronos.* Proceedings of the PresDB'07 workshop, Edinburgh.

[2] Geber, M. 2012. *Database Archiving in Sweden*, Presentation at 'A Practical Approach to Database Archiving' workshop, Copenhagen.

[3] International Council on Archives 2008. *Principles and Functional Requirements for Records in Electronic Office Environments – Module 3: Guidelines and Functional Requirements for Records in Business Systems*.

[4] O'Kane, T., Somerville, C. *Data in Databases – it's not what you think.* Presentation at Future Perfect 2012, Wellington.

[5] Queensland State Archives 2013. *What is a Public Record?* http://www.archives.qld.gov.au/Recordkeeping/GRKDownloads/Documents/what_is_public_record_200409.pdf.

[6] State Records of NSW Future Proof blog 16 June 2013. *Migrating Business Systems to the Digital Archives* http://futureproof.records.nsw.gov.au/migrating-business-systems-to-the-digital-archives-a-post-from-the-digital-archives-team/