Database Procedures

Version 1.98

Created date:	26 January 2012
Last updated:	24 August 2017
Review Due:	24 August 2018
Authors:	Stewart Waller, Jen Mitcham, Kieron Niven, Michael Charno, Tim Evans, Jenny O'Brien and Ray Moore
Maintained by:	Digital Archivists
Previous version:	Live

1 Purpose of this document

This document covers the procedures required to archive and present databases or information supplied in database format. There are three possible ways the data may be presented:

- as online searchable databases (previously known as Special Collections)
- as downloadable files constituting part of a Project Archive
- as catalogue records in the ArchSearch database

These all require different approaches. The same ingest/quality checking and documentation requirements should be applied in all cases, although dissemination strategies will differ.

A useful description of database types from the *Guides to Good Practice*:

> Databases can be divided into a number of types (known as models or architectures). The two of these most commonly found in use in archaeology are Flat File and Relational databases although there is a slowly growing movement towards the use of Object-oriented database models. The flat file model is broadly similar to that of a spreadsheet in that tabular data is organised into horizontal rows, representing records, and vertical columns or fields representing a type of value or attribute to be recorded. In flat file databases there can be an inherent looseness in the way that data is defined and recorded along with a significant duplication of sets of information from record to record. The relational model addresses these and other issues by requiring a data structure to be pre-defined and by splitting related groups of attributes into separate tables which are then linked together through key fields (Primary or Foreign keys). In contrast to spreadsheets and many flat file databases, most database applications allow (and in fact require) the strict specification - in terms of field length, data type (numeric, etc.) of the data types to be recorded.

As with charts generated from spreadsheet data, databases can potentially consist of more than just data values. Forms, used for data entry or for running queries, are often the only way in which many users interact with databases and can be viewed as part of the database but separate from the data itself. Likewise, the queries and results or reports that result from user interaction may also be considered as 'non-data' components of a database.[1]

2 Formats

Offered format	Accepted	Preservation	Presentation	Notes
Microsoft Access **.mdb**	YES	Comma separated values **.csv**	Comma separated values **.csv**	See footnote.[2]
Microsoft Access 2007 onwards **.accdb**	YES	Comma separated values **.csv**	Comma separated values **.csv**	See footnote.[3]
Microsoft Excel **.xls**	YES	Comma separated values **.csv**	Comma separated values **.csv**	Although a proprietary Microsoft format, the Excel .xls format is widely used and can be imported by a number of third-party applications.
Microsoft Excel 200 onwards **.xlsx**	YES	Comma separated values **.csv**	Comma separated values **.csv**	A relatively new format from Microsoft, released with Office 2007. They chose to develop their own specification (OOXML) rather than use the existing ODF

[1] http://guides.archaeologydataservice.ac.uk/g2gp/DbSht_1

[2] Proprietary Microsoft format. Not read by many database packages. MS Access is the GUI, the actual database engine is MS Jet (msjet##.dll, where ## is the version number, plus the various msjt*.dll files) which is loaded by Access. There have been format changes to the MDB file, particularly between versions 2 and 3 of Jet and versions of Access using Jet versions below 3 are probably inaccessible. We can accept Access 97 and above, but support for Access 97 will be short-lived. Ideally versions earlier than 97 should be migrated prior to deposit. Where this isn't possible, OpenOffice might offer a possible migration route.

[3] The ACCDB format was first introduced in Access 2007 and continued with Access 2010 and was developed in order to include enhanced functionality over the previous MDB format. It is arguable that, from a robust database design standpoint, additional functions and enhancements such as Multivalued Fields and Attachments only increase the difficulty in preserving such databases. It is also worth noting that, although the format is the default in Access 2007 and 2010, files created in Access 2010 may not be completely compatible with Access 2007. The ACCDB format, as with previous .mdb files, continues to be based on the Jet Database Engine.

				international standard. The format consists of human readable XML files packed with other content within a single zipped file
dBASE .**dbf**	YES	Comma separated values **.csv**	Comma separated values **.csv**	Ashton-Tate's dBASE format but they only registered the name, not the format.It is now a generic format referred to as xBase and is used by a number of databases and read by even more. The file structure is simple and stable and consequently all versions can be handled.
OpenDocument Database **.odb**	YES	Comma separated values **.csv**	Comma separated values **.csv**	
Paradox Database **.db**	NO	n/a	n/a	Getting increasingly difficult to open these in Access or OpenOffice software (can be done, but MS requires plugins). So best to ask for it in CSV or similar export.
Delimited text (including **.txt**,**.csv**,**tsv**	YES	Comma separated values **.csv**	Comma separated values **.csv**	Always nice to know the delimiter!
Exchange formats				
JavaScript Object Notation JSON **.json**	YES	JavaScript Object Notation JSON **.json**	JavaScript Object Notation JSON **.json**	Human-readable text to transmit data objects consisting of attribute–value pairs and array data types. JSON files should be accompanied by 'JSON schema definition file' (?JSD or JSON).

XML **.xml**	YES	XML **.xml**	XML **.xml**	Good old xml! SHould be accompanied with appropriate XSD file.
Resource Description Framework RDF **.rdf**	YES	Resource Description Framework RDF **.rdf**	Resource Description Framework RDF **.rdf**	RDF/XML is a syntax, defined by the W3C, to express (i.e. serialize) an RDF (Resource Description Framework) graph as an XML document
Other stuff[4]				
Structured Query Language Data **.sql**	YES			This is the SQL used to build the table - not really something we should be offered but may occasionally surface as usual documentation. Information is just plain text so we can handle ok.
Column Definition File **.sch**	YES			Holds information on db column data types - useful documentation and in plain text form so can accept.

For more detailed discussion, see the *Guides to Good Practice*.[5]

3 Documentation / Metadata

The following documentation is required for any database.

Database Documentation	
Element	**Description**
Filename	
Description	
Creation Date	Either date created or date of last

[4] Ideally we shouldn't be taking SQL dumps but these files can be used as supporting documentation.
[5] http://guides.archaeologydataservice.ac.uk/g2gp/DbSht_2

	edit/deposit
Software used	
Software version	
Entity relationship diagram filename	
Supporting documentation filename	
Table Documentation	
Table Name	
Table Description	
Primary keys	
Foreign keys	
Field Name	
Field Description	
Field Data type	
Field length	

Supporting Documentation

This should include:

- codes used
- units of measurement used in specific fields

4 Accessioning checks

- Do we have necessary documentation?
- Scan for data consistency. For special collections (online search) these issues should be flagged up to the depositor at accession.
- Presence of forms/sql etc - we can't do anything with these
- Orphaned tables and records or empty or unrelated tables.

Significant Properties:

- The actual data within the database - including field headings and the values themselves. Associated with this is the use of special characters in the dataset, from ampersands to greek characters (common in dating/scientific data). These must be identified and preserved.
- Relationships between tables - it is important that the relationships between tables/sheets are documented and understood.

5. How to convert files

Before converting files, it is advised to carry out a few checks.

- Check that referential integrity has been enforced for related tables. Run queries looking for duplicates and orphan records and highlight any issues with depositor. Where controlled vocabularies have been used to complete fields, make sure that the vocabularies have been controlled! We may want to create drop down lists of these terms in order to make the field searchable but this looks messy if there are consistency issues with data entry.
- If the database is supposed to link to a collection of images. Make sure there is a field in the database that holds the exact image name of the associated image file We don't want to have to do too much work at this end ensuring that images and database match up
- Check tables for duplicated rows, they are likely to result in incorrect or excessively duplicated records between linked tables. You can check by running a query such as SELECT field1, field2, … FROM table GROUP BY field1, field2, …
- Check text fields where the length of the data is the same as the field length – it may indicate truncated values.
- Check for embedded new lines, tabs and quotes, these may corrupt exported delimited text files.
- Scan text fields for the presence of common delimiter characters (',', '|', …). These will determine the need for text qualifiers
- Scan text fields for CP1252 characters (ASCII values between 128 and 161 - smart quotes, some accented characters, em dashes etc. There's also a handy lists.[6] These are not preserved in CSVs that use ANSI encoding. If present you'll need to convert the file encoding to UNICODE UTF-8 (see below).
- Scan text fields for characters beyond ASCII 167. If present then accented or other characters exist (as used in French, Gaelic, Ancient Greek and so on) and the Code Page or language of the original data must be determined. ANSI files preserve these characters, but it's worth recording that they exist within the documentation.

Carriage Returns

Carriage returns are a pain, software can assume that they signify a new line of data. Carriage returns need to be removed from your file before loading.

'Dodgy' characters

As highlighted above, ansi files do not preserve certain Windows-1252 characters. Files containing any troublesome characters should thus be saved with UNICODE encoding. If exporting from Access or OpenOffice software there should be options in the save/export facility to allow you to choose this setting.

Any technical specifications such as encoding should be recorded in the Process tab of the CMS.

If the data is to be used in an online interface, you'll need to replace these with html.

5. How to convert files

Microsoft Access Database .mdb

1. The ADS Toolkit[7] has a very good function to export all Access tables as delimited text. It also removes those pesky carriage returns for you!

[6] https://en.wikipedia.org/wiki/Windows-1252
[7] Available internally.

2. Use the 'Export' function in Access to export each table as a CSV ensuring field headings (and text qualifiers) are included.

OpenOffice Database (ODB)

As with exporting from Access, export each table as a CSV ensuring field headings (and text qualifiers) are included.

Dbase file (DBF)

1. Import into Access and use the 'Export' function in Access to export each table as a CSV ensuring field headings (and text qualifiers) are included

File-naming

Where possible files should retain the same name as the original (though the file extension may be different).

Where multiple tables of a database are being converted, folder name should reflect the name of the original database and the filename the name of the table the data came from, for example:

table name.csv.

It may however be necessary to change the table names. For example an MS Access data table may have the name 'Catalogue Flaked Lithics, Sand test pits, early survey sites etc' which isn't ideal and causes errors when you try and export it as a delimited text file. Or a table called 'descriptions' which is clearly a typo and will look bad when listed as a download on our web pages.

Any changes to database or table names within the preservation or dissemination versions should be recorded in the Process metadata.

5 Post-migration checking

- Check row counts after export
- Check text field lengths
- Check all tables have been exported
- Check any special characters have been preserved.

Storage

Data should be stored in appropriately named folders, as described in the ADS Repository Operations. Any directory structure from the SIP should try to be retained in the AIP. In some cases editing/restructuring may be required, any restructuring must be recorded in the Process table in the CMS.

Preservation

All data, not matter what the dissemination strategy, should have a preservation version. This should be structured as follows:

```
/preservation
        /{original_structure}
                table1.csv
                table2.csv
                table3.csv
```

/documentation
relationship_diag.tif

Dissemination: Simple Downloads

Storage of the dissemination copy of the database will depend on the dissemination strategy.

/dissemination/
/{original_structure}
table1.csv
table2.csv
Table3.csv
/documentation
relationship_diag.jpg

Dissemination: Online database (Special Collections)

The decision as to whether to disseminate the data behind special collections is dependent on the depositor. If they wish for data to also be available in this way then follow the instructions for the above.

Dissemination: Archsearch

Datasets incorporated into Archsearch do not need to be made available for download (unless specifically requested, which is very rare).