Data Management and Data Structuring

Peter K Austin
ELAP, Department of Linguistics
SOAS, University of London
24th June 2008

Outline

- Data management
- Properties of data
- Structured data management
- Relational data model
- XML
- Example

We can choose our values/priorities

- Standards & compliance
- Adeptness with tools
- Modelling of phenomena, architecture of data
- Dissemination/publishing
- Preserving
- Ethics, responsibility, protocol
- Range, comprehensiveness
- Intellectual rigour

- Which are priorities?
- Which are dispensible?

Data should be at least:

- explicit & robust
- consistent
- meaningful
- conventional
- adaptable, convertible, machine readable etc
- useful

Data portability

- Bird and Simons 2003:
- (for language documentation) our data needs to have integrity, flexibility, longevity and broad utility
- complete
- explicit
- documented
- preservable
- transferable
- accessible
- adaptable
- not technology-specific
Data management

- The way that data is structured is, in itself, information
- Structured data allows:
  - Usage including manipulation, conversion, derivation
  - Preservation
  - Machine readability

Data management system

- A data management system is a system you design for storing files and metadata:
  - Information about content (including structures)
  - Relationship between files and pieces of information
- It is not necessarily tied to any particular software, or even a computer

Data modelling

Data modelling is the process of designing your data management system:

- What information do you need to record?
- What are the units of information?
- What are their properties (attributes)?
- What are the relationships between the units of information?
- How is all this likely to change in the future?
- What kinds of structures are needed to store these?

Data management

- Two well-known ways of storing structured data:
  - Relational formats
  - Extensible Markup Language (XML)
- These are formats, not softwares or hardwares
- Any well-structured and documented data could be OK, but:
  - Less community of usage so less tools, support
  - ... (so) errors more likely and harder to diagnose

Directories and filenames

- Directories (folders):
  - Do (only) provide additional naming
  - ... and implicit hierarchical relationships
  - Can encourage bad practice
  - Cannot represent relationships between information within files
  - Can be platform specific

Filenames

- A (too) simple management system:
  - The information about the recording is captured in the filenames:
    1st_int_john_5Aug.wav
    market_conv.mj.wav
    ....
  - What does the code ‘int’ mean?
  - What information about the recording is missing?
note: file naming is still important, however!

Structured data management
- example of a simple management system:
  - a table in MS Word, Excel, Filemaker etc
  - don’t need to pack all information into filenames:
- some information is about the data
- some is about relationships between data
- a separate table should define the codes
- formalise the relationships within the data:
  - need unique identifiers

What does this achieve?
- conceptual/intellectual validity
- machine readable
- scalable, searchable, modular
- in fact, portable:
  - complete
  - explicit
  - documented
  - preservable
  - transferable
  - accessible
  - adaptable
  - not technology-specific

Relational data modelling
- a way of organising data
- a relational database architecture:
  - is not a machine
  - is not software
- it is composed of:
  - multiple tables containing records (rows) of data
  - relationships between records of data
  - …that’s all

Tables
- each record (row) represents one ‘entity’
- each field (column) represents a type of attribute
- each cell represents one unit of data

FOSF - a special table arrangement
- Field oriented standard format – developed by SIL and used by several applications programs
- each record begins and ends with a blank line (two carriage returns)
- each field is on a separate line beginning with the field label (always \xx ) and ending with a carriage return
- each cell (unit of data) is the material between space (after the field label) and carriage return
Example - dictionary

- ‘entry’ table
- we need room for multiple senses:
  - but how many?
- solution: use a **different table** for senses
- each sense can be linked to the entry it belongs to via a reference to the Entry’s **primary key**
- ...
- a sense can be linked to the entry it belongs to via a reference to the Entry’s **primary key**
  - in the new sense table, this is called a foreign key
- this is a **one-to-many** relationship:
  - one entry can have multiple senses
  - every sense belongs to exactly one entry

More complicated relationships

- so far, simplest lexical data only
- what if we wanted to relate sentence examples example to every relevant entry?
  - an additional table can express the relationships

Relational database software

- all RDB software uses the ‘tables and keys’ model described here:
  - MS Access, Oracle, MySQL, Filemaker
- they differ in what they additionally offer:
  - user interfaces (MS Access)
  - scalability, enforcement of data integrity (Oracle)
  - free-cost (MySQL)
  - etc

Markup format - XML

- XML came out of SGML - a system for incremental and collaborative “enrichment” of texts
- XML design principles
  - 1. XML shall be straightforwardly usable over the Internet.
  - 2. XML shall support a wide variety of applications.
  - 3. XML shall be compatible with SGML.
  - 4. It shall be easy to write programs which process XML documents.
  - 5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
  - 6. XML documents should be human-legible and reasonably clear.
  - 7. The XML design should be prepared quickly.
  - 8. The design of XML shall be formal and concise.
  - 9. XML documents shall be easy to create.
  - 10. Terseness is of minimal importance.
XML Introduction

- XML is a way of creating explicit formal structures using only plain text.
- Structures are defined by tags in angle brackets:
  - eg: `<noun>`
- Tags are usually in pairs:
  - A start/open tag, and an end/close tag:
    - `the <noun> dog </noun> chased ...`
- But can also be single and closed:
  - `the dog <pause /> sat down`
- Tags can have attributes with values:
  - `the <noun num="1"> dog </noun> sat down`
- You can name your tags, attributes or values (almost) anything.
- There are some restrictions:
  - You can have hierarchies, but not overlaps:
    - `<a>the<b><c>cat</c> sat</b> on the mat</a>`
    - `<a>the<b><c>cat</c> sat</b> on the mat</a>`

XML Uses

- XML can be thought of as:
  - As a stream (eg: a stream of text)
  - And/or
    - As a (tree) structure (eg: a dictionary, ontology etc)
- For many applications, XML is how the data is stored underneath:
  - It is created automatically (it's still good to know about!)
- There are good applications that allow you to create XML without typing in plain text:
  - Eg: oXygen, XMLSpy
  - They also ensure it is well-formed XML

What does marking up as XML do?

- Makes your existing structures explicit
- Creates machine readable, exchangeable, preservable structured data
- Makes your stupid decisions explicit
- Create machine readable, exchangeable, preservable junk

This is only a part of documentation skills

- Consultation and elicitation:
  - Obtain knowledge about an endangered language and its communities
- Recording:
  - Record the knowledge/performance of the documentation participants
- Data management:
  - Supports: input, store, manipulate, preserve, adapt, share etc
- Analysis, dissemination, etc ...