Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 2005 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

December 2005

Managing Multi-Valued Attributes in Spreadsheet Applications

Clare Churcher
Lincoln University

Theresa McLennan
Lincoln University

Wendy Spray
Lincoln University

Follow this and additional works at: http://aisel.aisnet.org/pacis2005

Recommended Citation

Churcher, Clare; McLennan, Theresa; and Spray, Wendy, "Managing Multi-Valued Attributes in Spreadsheet Applications" (2005). *PACIS 2005 Proceedings*. 15.
http://aisel.aisnet.org/pacis2005/15

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2005 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Managing Multi-Valued Attributes in Spreadsheet Applications

Clare Churcher, Theresa McLennan and Wendy Spray
Lincoln University
New Zealand
churcher/mclennan/sprayw@lincoln.ac.nz

Abstract

End-users frequently prefer to use spreadsheets as a medium for storing simple data. There are a number of commonly used data arrangements to cope with data items that are connected with a many-many relationship. These arrangements offer the end-user simplicity in data entry and simple reporting. However as the application evolves it soon becomes apparent that these arrangements of data have serious drawbacks when it comes to querying and summarising the data. At this point it becomes necessary to move the data to a database. This is a non-trivial task with currently available tools. In this paper we review the data arrangements typically used to store many-many data relationships in a spreadsheet. We report on the development of an application to assist end-users to transform their spreadsheet data to normalised database tables. The application exploits the XML representation of spreadsheet data and employs the XML classes in Visual Studio.Net.

Keywords: Spreadsheet design, multi-valued data, end user computing.

1. Introduction

Scientists, economists, businesses and other computing end-users have widely adopted the spreadsheet as a tool for storing data. Spreadsheets are principally designed for calculations, manipulations and analysis of data but frequently people use them as a simple substitute for a small database. In order to get started on an initial task, minimal design is required and entering the data is simple. However as a project evolves problems inevitably arise with inconsistent data, querying, summarising and reporting.

Most commonly these problems can be solved by moving the data into a well designed database schema. The first step is, of course, to design a suitable schema and this problem is well addressed in countless books on analysis and design. Given that an appropriate schema has been determined there is still the considerable problem of taking the data from the spreadsheet and manipulating it into the appropriate normalised tables. This task is non-trivial and current tools do little more than allow the user to take specified ranges of cells into single tables. Where a user is faced with taking multi-valued attributes into rows in a normalised table there is very little help available. The attributes are often stored in several adjacent columns and this makes the task fiddly and time consuming to the extent that it is quite often abandoned. The user is faced with the prospect of persevering with a problematic spreadsheet or starting all over again in a database.

There are a number of common approaches that users employ to deal with multi-valued attributes in a spreadsheet, each with its own strengths and weaknesses. Such solutions commonly reflect the layout of a report that the user has in mind rather than the underlying relationships inherent in the data. This problem is also found in many databases where a user, inexperienced in analysis, will construct a single flat table to store data.

It is easy to argue that spreadsheets are not designed for storing data or that database schemas should be designed properly. This ignores the reality of the situation. People like the immediacy and apparent simplicity of spreadsheets and will, in all likelihood, continue to use them as a means of data storage.

There is considerable literature on managing the computational aspects of spreadsheets (see for example Panko, 1998, Randolph *et al* 2002 and Rajalingham *et al* 1999). There are also well known websites such as Panko's spreadsheet research site (http://panko.cba.hawaii.edu/ssr/) which contains much useful material with the focus on preventing and detecting errors in spreadsheet formulas. Also useful are Chadwick's Spreadsheet Methodology website (http://www.cms.gre.ac.uk/research/iirc/index.html) and best practice guidelines for developing traditional spreadsheet models to perform calculations (e.g. Read and Batson, 1999). However there is little work on designing a spreadsheet to be used for data storage.

Having realised that a spreadsheet or database should have been designed differently, there is still the non-trivial problem of transforming the data into a more appropriate form. Our aim is to develop tools that enhance end-users' abilities to evolve their applications "… *without dependence on IT professionals or IT departments*" [Sutcliffe and Mehandjiev 2004].

In this paper we review some ways data with many-many relationships is commonly arranged in a spreadsheet and discuss the usefulness and drawbacks of each of them. We then introduce an application that allows the end-user to transform this data into a format suitable for importing into normalised database tables. We exploit the XML representation of the spreadsheet data in order to do this.

2. Many-Many Relationships in a Database

As soon as a user needs to store data about more than one thing they will inevitably need to deal with multi-valued attributes or many-many relationships. Very simple examples include: club members having many skills; team members being available for several matches; employees working at many tasks; plants having many uses; students enrolling in many papers; or (the perennial) academics having many areas of expertise.

We will illustrate many-many relationships with the simple example of people having many hobbies as it is self explanatory. A class diagram for this relationship is shown in Figure 1.

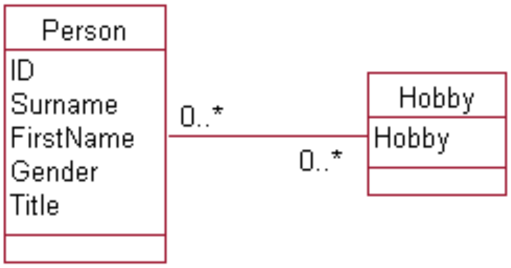
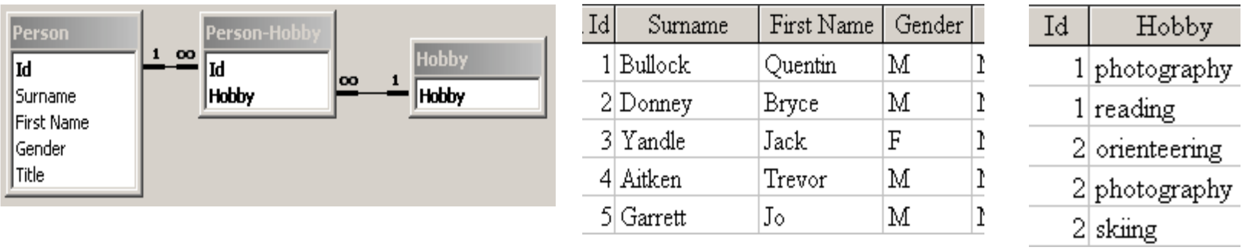


Figure 1. Many-many relationship

In a relational database this would be represented by three tables and Figure 2 shows these as they would appear in Microsoft Access. This allows attributes to be stored about people and hobbies and also, if required, about the relationship between them.



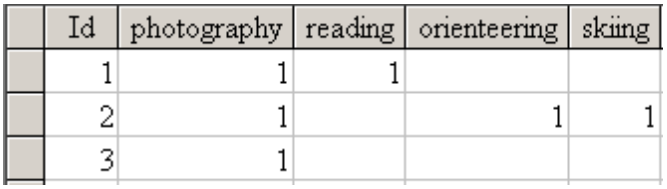
Id	Surname	First Name	Gender
1	Bullock	Quentin	M
2	Donney	Bryce	M
3	Yandle	Jack	F
4	Aitken	Trevor	M
5	Garrett	Jo	M

Id	Hobby
1	photography
1	reading
2	orienteering
2	photography
2	skiing

Schema Person Table Person-Hobby Table

Figure 2. Schema and some data for many-many relationships in a relational database

With the data in this form, SQL queries can be used to answer a multitude of questions. E.g. who is interested in this OR that, who is interested in this AND that, who is interested in everything, counts of interests for each person or people with particular interests. In database software such as Access we can also use this format to produce a pivot table or cross tab query, part of which is shown in Figure 3 where the counts of 1 in this case denote that a person has a particular interest.



Id	photography	reading	orienteering	skiing
1	1	1		
2	1		1	1
3	1			

Figure 3. Cross tab from Person-Hobby table

While this is the correct way to model a many-many relationship in a database, many end-users unfamiliar with design issues will (quite understandably) set up a single unnormalised table to represent the data.

In the next section we discuss arrangements we have observed being used in spreadsheets. Often these arrangements are exported to a database without modification which does not address their fundamental problems.

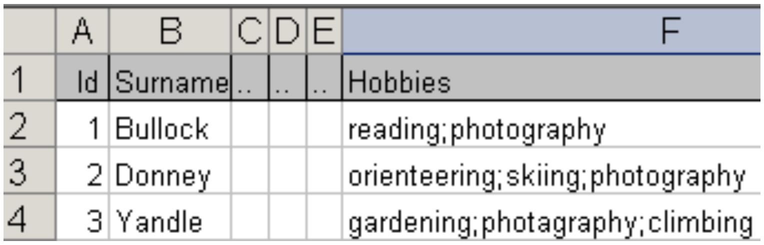
3. Many-Many Relationships in a Spreadsheet

Often a spreadsheet will be set up by a researcher, club or small business to keep information about just one thing; in this case, perhaps, club members and their details. As an after thought they may add a column to keep another attribute such as hobby and in time realise that in fact one hobby is insufficient. It is at this stage that the problems begin.

There are a number of ways this situation can be addressed in a spreadsheet, each with its own strengths and problems. In this section we describe four ways of storing the data for a many-many relationship and in the next section we compare their usefulness in different situations. All but one of these arrangements stores all the data in a single worksheet range.

3.1 Multi-valued Field

One solution is to put several values for Hobby in a single cell or field, perhaps separated by some sort of punctuation as in Figure 4. We will discuss ways to query this layout in Section 4, however it is worth noting that it is not possible to use standard spreadsheet features to validate data in order to prevent (for example) the different spellings of photography in rows 2 and 4.

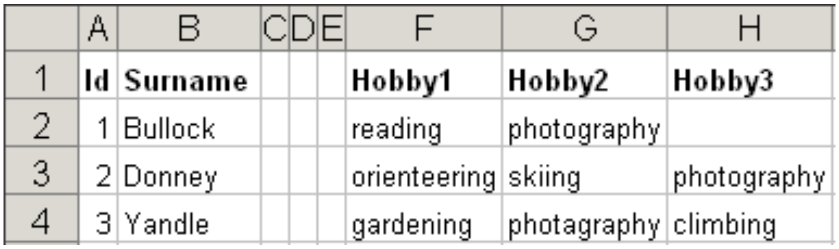


	A	B	C	D	E	F
1	Id	Surname	..	..	..	Hobbies
2	1	Bullock				reading;photography
3	2	Donney				orienteering;skiing;photography
4	3	Yandle				gardening;photagraphy;climbing

Figure 4. Multi-valued Hobby field

3.2 Repeated Columns

In this layout the multiple values for the hobby attribute are separated into (usually contiguous) columns as in Figure 5.



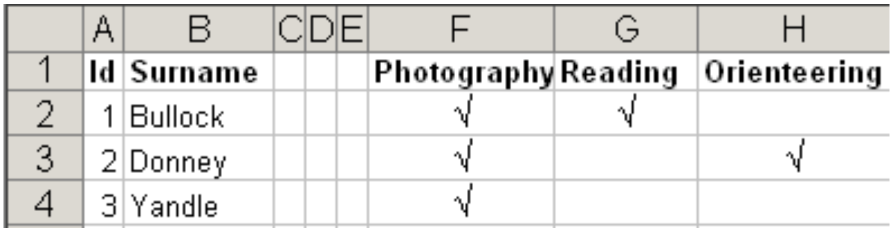
	A	B	C	D	E	F	G	H
1	**Id**	**Surname**				**Hobby1**	**Hobby2**	**Hobby3**
2	1	Bullock				reading	photography	
3	2	Donney				orienteering	skiing	photography
4	3	Yandle				gardening	photagraphy	climbing

Figure 5. Repeated Hobby columns

Once again we have problems with inconsistent spelling. This can be remedied by creating a separate list of hobbies and referencing this list in an Excel validation on each of the Hobby columns. This is more-or-less equivalent to having a separate Hobby table and referential integrity as in Figure 2.

3.3 Categories as Columns

A third option of representing a many-many relationship is to have a column for each possible value of Hobby and represent a person's interest in a hobby with some entry in the appropriate cell as shown in Figure 6. This overcomes the problem of incorrectly spelt categories.



	A	B	C	D	E	F	G	H
1	**Id**	**Surname**				**Photography**	**Reading**	**Orienteering**
2	1	Bullock				√	√	
3	2	Donney				√		√
4	3	Yandle				√		

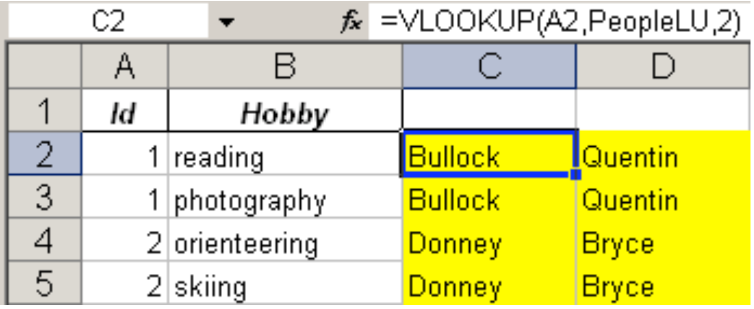
Figure 6. Hobby categories as columns

3.4 Normalised Ranges

It is also possible to keep the data in a spreadsheet in a similar manner to the normalised tables in a database. Each Person-Hobby pair is entered on a separate row while the data about each Person and Hobby (for validation purposes) is kept in a separate range elsewhere. In many respects this arrangement is now almost isomorphic to the database schema in Figure 2.

By using a VLookup function it is possible to "see" all the related person information in one contiguous range as shown in Figure 7. This is a bit like mimicking a join in a relational database.

While this solution is possible, it is unlikely to be used as it is more conceptually difficult than the single range solutions previously discussed. Furthermore it offers few of the advantages (such as powerful querying) that relational database software can provide if data is stored in this way.



C2 fx =VLOOKUP(A2,PeopleLU,2)

	A	B	C	D
1	***Id***	***Hobby***		
2	1	reading	Bullock	Quentin
3	1	photography	Bullock	Quentin
4	2	orienteering	Donney	Bryce
5	2	skiing	Donney	Bryce

Figure 7. Normalised ranges and look-ups

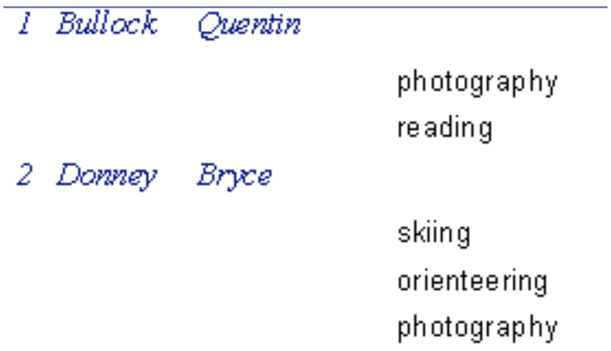
4. Comparing Different Data Representations

When using a spreadsheet to store data (as opposed to performing calculations) end-users generally design the spreadsheet to reflect the way they would like to input or output their data. Being able to query or summarise the data is often a secondary consideration. In this section we compare the methods in the previous section with respect to reporting, querying and summarising the data. Surprisingly it is not at all clear which of the methods is most useful.

4.1 Reporting

The multi-valued field and repeated column arrangements (Figures 4 and 5) satisfy the need to quickly and concisely print the hobbies associated with a particular person. The categories as columns arrangement is also useful where the total number of possible categories is small enough to conveniently fit across a printed page.

Curiously, in spite of the considerable reporting facilities in relational database software such as Access, it is not simple to produce a report with people and hobbies all on one line from a schema such as Figure 2. We are constantly being approached by users wanting such one line reports and only being able to produce the standard report with grouping as in Figure 8.



```
1  Bullock   Quentin
                                  photography
                                  reading
2  Donney    Bryce
                                  skiing
                                  orienteering
                                  photography
```

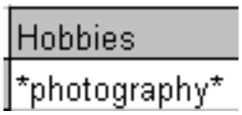
Figure 8. Output from a standard report writer

There is no simple solution to this reporting problem in database software and it would be useful for many people to be able to take data from a relational schema and transform it into something like Figures 4 or 5 for reporting purposes. We discuss this further in Section 5.

4.2 Simple Queries

Data in a relational schema such as Figure 2 is able to be queried using all the power of SQL. How do the different spreadsheet representations compare? The three single table arrangements (Figures 4 to 6) are useful for data entry and for concise reporting of the data. However eventually spreadsheet users will want to ask questions such as "Who is interested in Photography?". The categories as columns arrangement (Figure 6) is the most simple for answering this type of question as it can be achieved by simply applying an appropriate autofilter to the photography column.

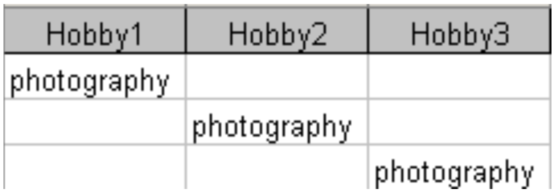
The other methods of storage require knowledge of advanced filters, criteria tables and wildcards. Of these, the unsophisticated multi-valued field storage of Figure 4 is the easiest to deal with by using a simple criteria table with wildcards as in Figure 9.



Hobbies
photography

Figure 9. Criteria table for filtering multi-valued cells

For the repeated column arrangement, Figure 5, we have to query every hobby column. This can be achieved with the more complex criteria table shown in Figure 10. This suffers from the problem that whenever as extra column is added (e.g. Hobby4) the criteria table needs to be updated. The complexity of advanced filtering is starting to get beyond the comfort level of many end-users. With this level of skill the user would probably be well placed to move to a database and employ the querying capabilities there.

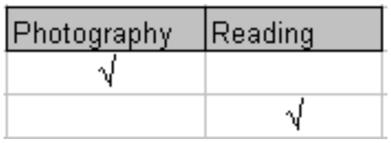


Hobby1	Hobby2	Hobby3
photography		
	photography	
		photography

Figure 10. Criteria table for filtering repeated columns

4.3 More Complex Queries

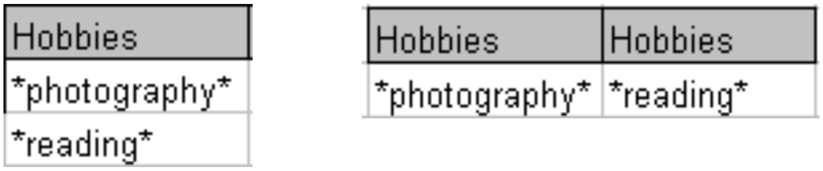
Queries involving AND and OR are also likely to be required as the application evolves. The categories as columns arrangement deals easily with an AND query by simply applying autofilters to two columns. An OR query for this arrangement requires Excel's advanced filter and a criteria table as in Figure 11. In spreadsheet software other than Excel, the filtering options are a little different. For example the filter in OpenOffice.org's Calc can also perform ORs between fields.



Photography	Reading
√	
	√

Figure 11. Criteria table for OR queries with categories as columns

The multi-valued field arrangement can answer AND and OR queries using Excel's advanced filter and relatively simple AND and OR criteria tables as in Figure 12.



Hobbies
photography
reading

Hobbies	Hobbies
photography	*reading*

Figure 12. Criteria tables for OR (left) and AND (right) queries on multi-valued fields

For the repeated columns arrangement (Figure 5) the criteria tables for AND and OR become unreasonably awkward and unwieldy.

The normalised ranges (Figure 7) are also very difficult to query in a spreadsheet for other than the simple "Who is interested in photography?" type queries.

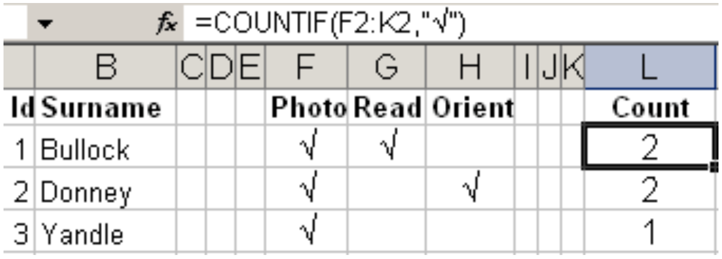
4.4 Summarising Data

Another likely use of data such as this is to ask questions such as “How many people are interested in each of the hobbies?” or “How many interests do people have on average?”

For the database schema (Figure 2) these questions can be easily answered with SQL queries using grouping.

In spreadsheets there are numerous, powerful functions to produce various statistics (e.g. DCount, DAverage, CountIF) and also tools such as pivot tables (cross tabs), data tables and frequency distributions. Some of these tools can be quite daunting if used infrequently. The different arrangements of data lend themselves to analysis by some or all of these tools.

The categories as columns arrangement can produce totals for rows and columns with simple formulas to count cells containing values as shown in Figure 13.



fx =COUNTIF(F2:K2,"√")

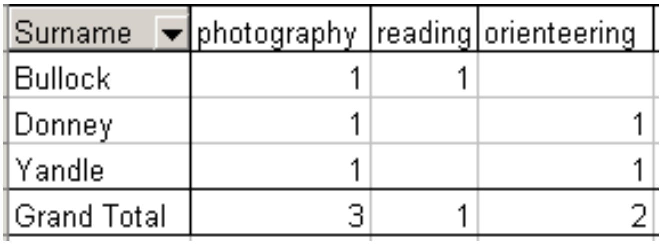
	B	C	D	E	F	G	H	I	J	K	L
Id	Surname				Photo	Read	Orient				Count
1	Bullock				√	√					2
2	Donney				√		√				2
3	Yandle				√						1

Figure 13. Summarising categories as columns

This method of counting is prone to errors when cells may contain additional blanks or other non printable characters. Using other functions (e.g. Trim) helps but it requires considerable care and discipline to keep these summaries (as well as the data) accurate.

The multi-value and repeated columns arrangements are very difficult to get overall summaries such as the above although it is possible to use criteria tables and DCounts for specific queries (e.g. How many Hobbies does Bullock have?).

With the normalised ranges it is very difficult to get overall summaries as it stands but this arrangement is the only one that allows us to easily produce a pivot table (simple cross tab). Part of this is shown in Figure 14.



Surname	photography	reading	orienteering
Bullock	1	1	
Donney	1		1
Yandle	1		1
Grand Total	3	1	2

Figure 14. Pivot table produced from normalised ranges

4.5 Summary

Each of the data arrangements has different strengths. A relational database schema clearly is the best for data integrity and querying but it does have a slight (although often annoying) shortcoming with producing reports of each person and all their hobbies on one line.

Within a spreadsheet the repeated columns arrangement provides nice output and for this reason is very commonly used for many-many relationships such as this one. However it is the most troublesome for querying and summarising.

The unsophisticated multi-valued field is fine for output and is surprisingly easy to query. However, other than by using the cumbersome Excel data table feature, it is almost impossible to summarise.

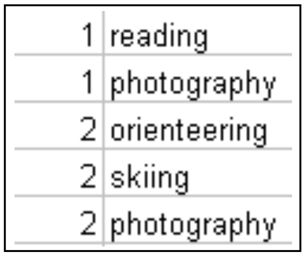
The normalised ranges are not at all intuitive for a spreadsheet user although they do provide some data integrity similar to a database. Querying is difficult but this arrangement does allow the easy creation of a pivot table.

5. Transforming Data

None of the spreadsheet arrangements described in the previous section provides everything that a user might typically require. Ideally data of this type should be kept in a normalised database schema such as in Figure 2. This would provide all the querying, summarising and most of the reporting requirements we have discussed. However the problem then is how to transform the data from each of these arrangements into two or more database tables.

The People table, in the relational schema (Figure 2) can be easily produced from any of the spreadsheet arrangements. The ranges can be saved as text and read into a database, and in Access the import wizard can import them directly.

The real problem arises in producing the middle table combining people and hobbies. For the arrangement in Figures 4 to 6 the data needs to be taken from multiple values or multiple columns and recast as a series of person-hobby pairs as in Figure 15.



1	reading
1	photography
2	orienteering
2	skiing
2	photography

Figure 15. Format required for normalised table.

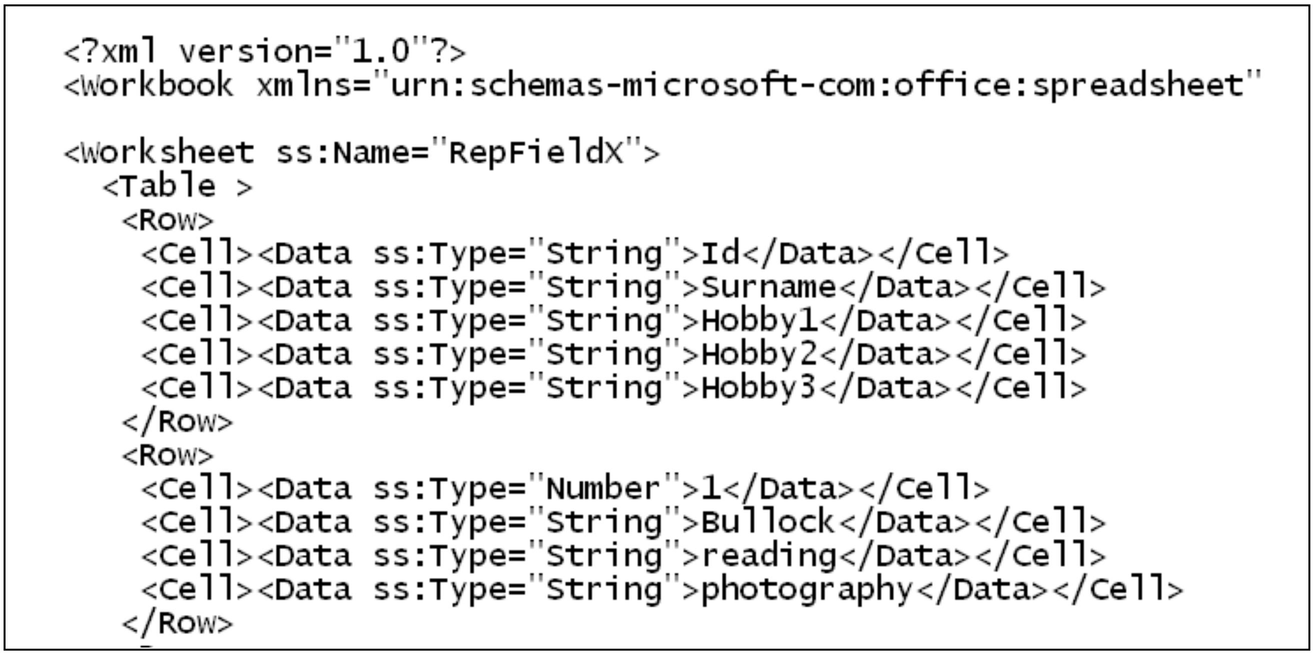
Once we have this output it is a simple matter create the Hobby table.

Even add-ins such as 4Tops (http://www.4tops.com/, 2004), which have extended Access import features, still require multiple passes to successfully transform the arrangements of Figure 5 into this form and cannot cope at all with Figures 4 or 6.

5.1 XML Representation of Data

We have investigated exploiting the XML nature of Excel and Access in OfficeXP (http://www.w3.org/XML/ and http://www.microsoft.com/office/xml/default.mspx) to facilitate this data transformation.

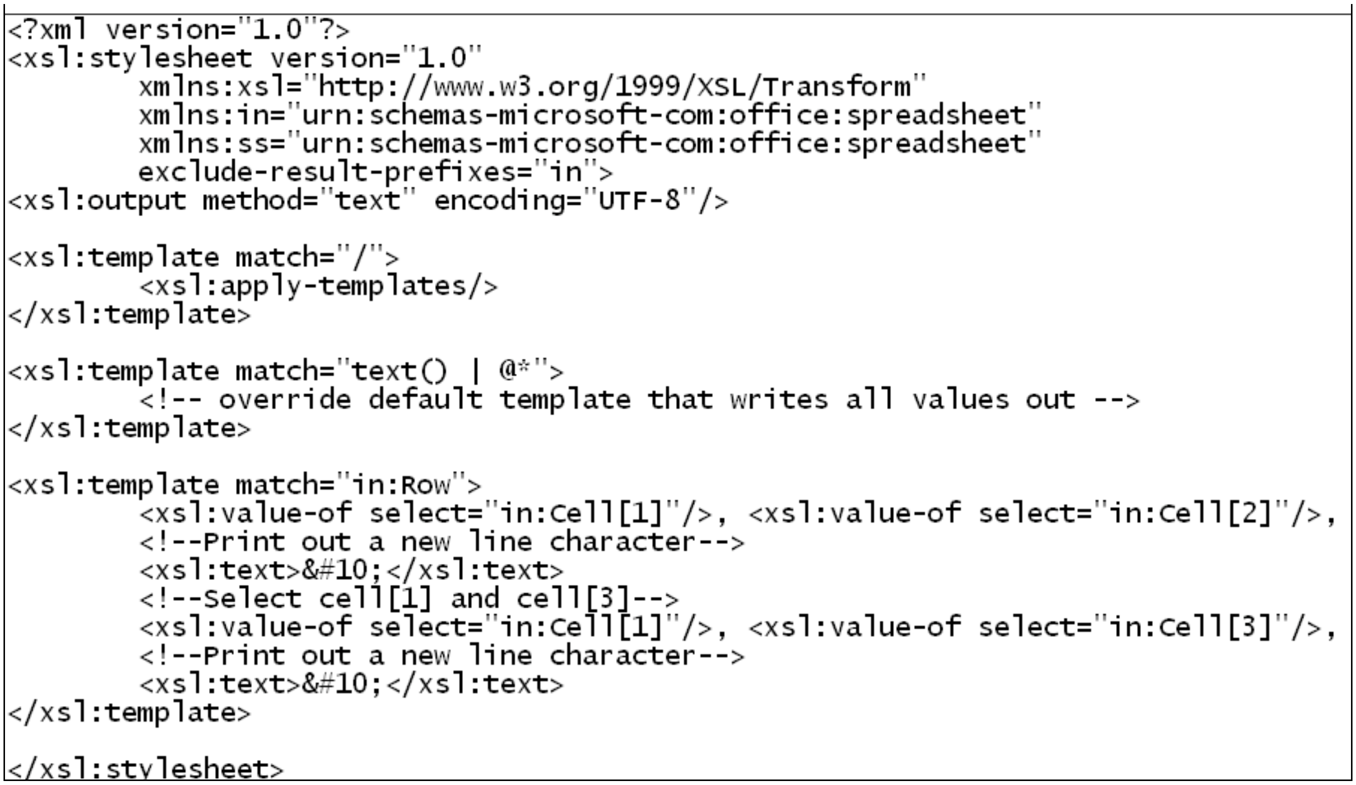
In Excel each workbook, worksheet, row and cell is treated as an element or node of a hierarchy. Cells can have attributes (for formatting and type) and also contain data nodes with the actual data values in them. For the repeated columns arrangement of Figure 5 the XML representation for the main columns of the first two rows (without any special spreadsheet formatting) is shown in Figure 16.



```
<?xml version="1.0"?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"

<Worksheet ss:Name="RepFieldX">
  <Table >
   <Row>
    <Cell><Data ss:Type="String">Id</Data></Cell>
    <Cell><Data ss:Type="String">Surname</Data></Cell>
    <Cell><Data ss:Type="String">Hobby1</Data></Cell>
    <Cell><Data ss:Type="String">Hobby2</Data></Cell>
    <Cell><Data ss:Type="String">Hobby3</Data></Cell>
   </Row>
   <Row>
    <Cell><Data ss:Type="Number">1</Data></Cell>
    <Cell><Data ss:Type="String">Bullock</Data></Cell>
    <Cell><Data ss:Type="String">reading</Data></Cell>
    <Cell><Data ss:Type="String">photography</Data></Cell>
   </Row>
```

Figure 16. Main tags of the XML representation of repeated column data (Figure 5)



```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:in="urn:schemas-microsoft-com:office:spreadsheet"
        xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
        exclude-result-prefixes="in">
<xsl:output method="text" encoding="UTF-8"/>

<xsl:template match="/">
        <xsl:apply-templates/>
</xsl:template>

<xsl:template match="text() | @*">
        <!-- override default template that writes all values out -->
</xsl:template>

<xsl:template match="in:Row">
        <xsl:value-of select="in:Cell[1]"/>, <xsl:value-of select="in:Cell[2]"/>,
        <!--Print out a new line character-->
        <xsl:text>&#10;</xsl:text>
        <!--Select cell[1] and cell[3]-->
        <xsl:value-of select="in:Cell[1]"/>, <xsl:value-of select="in:Cell[3]"/>,
        <!--Print out a new line character-->
        <xsl:text>&#10;</xsl:text>
</xsl:template>

</xsl:stylesheet>
```

Figure 17. XSLT to create a normalised text output from a key field in Column 1 and repeated values in columns 2 and 3.

In order to generate the person-hobby pairs as in Figure 15 we need to nominate a key column(s) (in this case Id), and the columns which need to be normalised into rows of pairs (in this case all the Hobby columns).

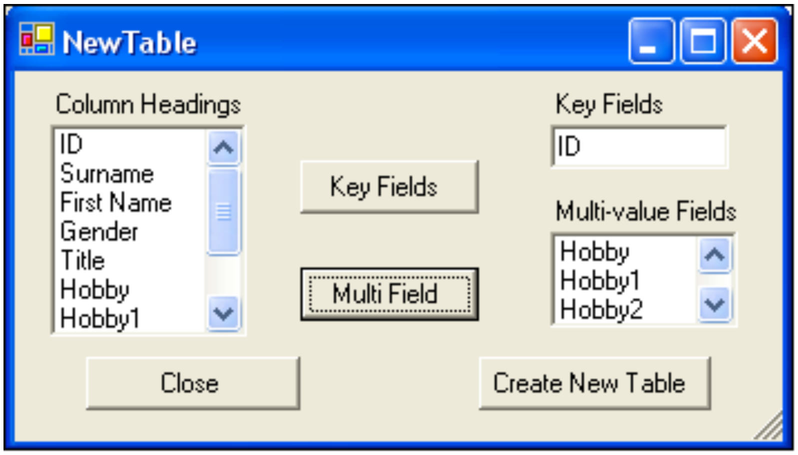
It is possible to use standard style sheet technology to parse each row in the XML hierarchy and output the key value and the value of each of the hobbies. Figure 17 shows a very simplified XSLT style sheet (http://www.w3.org/TR/xslt) for the case where the key field is in column 1 and the repeated columns are columns 2 and 3. Further elaboration of this transformation can be found in Zhang 2003.

5.2 A Normalisation Transformation Application

Using the XML classes available in Visual Studio.NET we have developed an application which allows end-users to transform their data into a form suitable for reading into a normalised database table as in Figure 15.

The application uses an ADO connection object (http://msdn.microsoft.com/data/Default.aspx) to allow the user to connect to any Excel spreadsheet. Having determined the data range they wish to transform they are presented with a form as in Figure 18. The box on the left displays the headers of the columns in the range and the user can choose the key column(s) and those columns which contain the multi-valued data.

The data is then parsed using the .NET XML classes and is output in a format similar to that shown in Figure 19. Each row contains a value for the key field (in this case Id) and one value from the repeated columns. This is a format consistent with the normalised table of Figure 15 or of the normalised spreadsheet range of Figure 7.

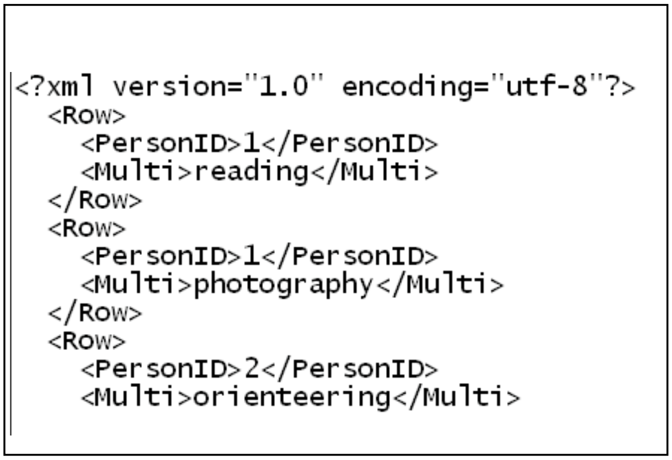


Column Headings: ID, Surname, First Name, Gender, Title, Hobby, Hobby1

Key Fields: ID

Multi-value Fields: Hobby, Hobby1, Hobby2

Figure 18. Interface for choosing columns to transform to normalised output.



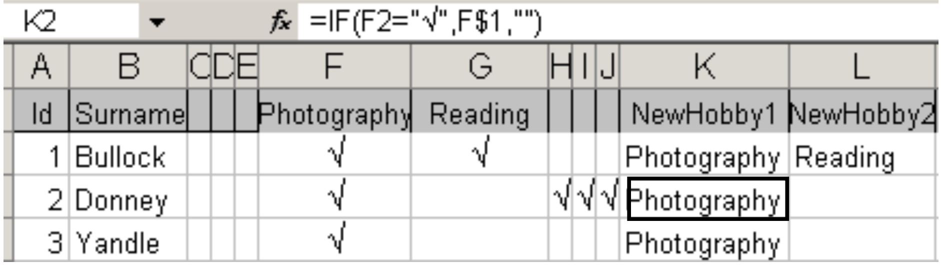
```
<?xml version="1.0" encoding="utf-8"?>
  <Row>
    <PersonID>1</PersonID>
    <Multi>reading</Multi>
  </Row>
  <Row>
    <PersonID>1</PersonID>
    <Multi>photography</Multi>
  </Row>
  <Row>
    <PersonID>2</PersonID>
    <Multi>orienteering</Multi>
```

Figure 19. XML output of normalised data.

This XML can then be easily transformed to a format (possibly even just text) which is able to be read into a database table or spreadsheet range.

5.3 Transforming Other Arrangements to Repeated Columns

The application described in the previous section transforms any data stored in the repeated columns arrangement (Figure 5). Data stored in a multi-valued field or as categories in columns are easily transformed to this arrangement using standard Excel features. Transforming the multi-valued field can be done quite simply with Excel’s Convert Text to Columns Wizard. A range of repeated columns can be created from the categories as columns arrangement with a simple formula as shown in Figure 19. The application can then be used on the newly created columns (in this case K:O).



K2 =IF(F2="√",F$1,"")

A	B	C	D	E	F	G	H	I	J	K	L
Id	Surname				Photography	Reading				NewHobby1	NewHobby2
1	Bullock				√	√				Photography	Reading
2	Donney				√		√	√	√	Photography	
3	Yandle				√					Photography	

Figure 19. Converting categories as columns(F:J) to repeated columns (K:O).

6. Conclusions

Spreadsheets are being used by end-users as a way of storing simple and not so simple data. Even quite simple projects inevitably have many-many relationships between the data elements. Spreadsheet users favour several arrangements (multi-valued field, repeated columns and categories as columns) which can generally support convenient input and reporting of these data. Unfortunately as the application evolves it soon becomes apparent that these arrangements have severe problems when it comes to querying and summarising the data.

At this point in the life-cycle of the application it is often necessary to move the data to a normalised database schema. However the process of transforming the data into normalised tables is far from trivial and is not supported by current tools.

We have developed an application which allows an end-user to connect to a spreadsheet and select those repeated columns that need to be transformed into a normalised table. The application employs the XML hierarchy structure of the spreadsheet data to output a new XML file which is suitable for creating the normalised table.

7. References

Panko, R.,"What we Know About Spreadsheet Errors", *Journal of End User Computing*, 10, No. 2, 1998, pp. 15-21.

Rajalingham, K., Chadwick, D., Knight, B. and Edwards, D., "An Integrated Spreadsheet Engineering Methodology (ISEM)", *Proc. IFIP TC11 WG11.5 Third Working Conference on Integrity and Internal Control in Information Systems*, Kluwer academic publishers, Amsterdam, The Netherlands, 1999, pp. 41-58.

Randolph, N., Morris, J. and Lee, G., "A Generalised Spreadsheet Verification Methodology", *Proc. Twenty-Fifth Australasian Computer Science Conference (ACSC2002), Conferences in Research and Practice in Information Technology* 4, Melbourne, Australia., 2002, pp. 215-222.

Read, N. and Batson, J., Spreadsheet Modelling Best Practice, *Institute of Charted Accountants for England and Wales*, Great Britain, 1999.

Sutcliffe, A. and Mehandjiev, N., "End-User Development", *Communications of the ACM* 47, No. 9, 2004, pp. 31-32.

Zhang, X. Y., *"An investigation into transforming spreadsheet data using XML",* Hons dissertation, Lincoln University, New Zealand, 2003.