Cromwell in Ireland: a temporal database

Presented at DRH 2003

Authors

Brian Farrimond
Liverpool Hope University College
farrimb@hope.ac.uk

Dr Fiona Pogson
Liverpool Hope University College
pogsonf@hope.ac.uk

Lindy Parkinson
Liverpool Hope University College
parkinc@hope.ac.uk

Colette Redmond
Liverpool Hope University College
redmoc@hope.ac.uk

Dr Steve Presland
Liverpool Hope University College
preslas@hope.ac.uk

Proofs and correspondence to:
Brian Farrimond
IMC
Liverpool Hope University College
Hope Park
Liverpool
L16 8ND

Tel 0151 291 3606

Contents

Abstract

This paper describes work carried out at Liverpool Hope University College in developing temporal databases to be used in the teaching and learning of history. The objective has been to establish a simple but flexible means of recording temporal historical data and to provide software tools that enable the data to be analysed and to be displayed graphically. The work is illustrated by a temporal database modelling Cromwellian military action in Ireland - August 1649 to October 1651.

A temporal database is one that contains temporal data – representations of timestamped facts. Temporal databases have been the object of research since the mid 1970s and although as yet there is no commercial implementation of a temporal database management system (DBMS), existing relational DBMS can be used to record and analyse temporal data.

In order to be useful to the historian, data entry to a temporal database needs to be straightforward, quick and easy. Hence, a key objective of the authors' work has been to make data entry as straightforward but as comprehensive as possible.

A format for data capture, using a spreadsheet, is described. The format provides the historian with complete freedom in selecting the characteristics to be recorded about historical people – individuals or groups, such as armies - and things, such as locations. If the characteristics can be represented as numbers or text or sequences of numbers or sequences of text then they can be recorded in the spreadsheet. Each recorded characteristic can be time stamped in the spreadsheet.

The spreadsheet is automatically imported into a relational database structure. The authors then illustrate the usefulness of this database by demonstrating how time maps

can be automatically generated from the database. They go on to discuss how the facilities provided by the DBMS could be used to query the data directly.

1 Introduction

Temporal questions are questions that relate to time and they pervade the field of History. At the elementary level there are questions such as who was on the English throne at a particular date. More complex queries can be imagined that might be used to support or disprove a historical conjecture. e.g. “did Laud and Wentworth both attend the Privy Council on 5 May 1640?”. Temporal questions are, of course, not confined to History. Many applications ask temporal questions. For instance, an organisation might ask, "when was this employee absent from work?" or a hospital might ask "how long was the patient taking this type of drug?".

The information needed to answer historical questions is stored in databases of one kind or another. The databases may, for example, be historical archives stored on paper or they may be electronic databases stored on computers perhaps accessed through the Internet. The electronic databases currently in use can store large quantities of information efficiently and answer many complex queries.

Information held on computers are now mainly stored with the aid of a database management system (DBMS). Microsoft Access is a typical DBMS available on the PC. Large organisations will use more powerful DBMS systems such as Oracle (Oracle 2003). The DBMS provides facilities for storing and accessing the information in ways that avoid inconsistency and duplication. The DBMS enables the user to specify rules to be enforced such as not allowing an order to be stored for whom there is no corresponding customer. The DBMS provides powerful query languages such as SQL (Date, 2000), which enable the user to ask complex questions of the stored data. The DBMS usually has facilities for storing programs to process the information. Finally the

DBMS provides a range of security mechanisms for restoring lost or corrupted information and restricting access. However, the DBMS does not provide facilities for answering time-based questions easily (Date, 2000). This requires a temporal DBMS.

Despite research on temporal databases being carried out since the 1970s, there is still no major commercial temporal DBMS available on the market. A number of reasons have been put forward to explain why (Date 2000). These include the significant cost of disk storage, the effort needed to develop the required query languages and the lack of an academic consensus as to the best approach.

While we await the arrival of a fully temporal commercial DBMS, it is possible to implement a temporal database on a conventional DBMS. A number of examples have been developed including TimeDB (TimeDB, 1999). TimeDB translates temporal SQL statements into standard SQL statements, which are then executed on a commercial DBMS such as Oracle. Another example is the implementation of the TF-ORM temporal data model on Oracle (Hubler and Edelweiss, 2000). Hubler and Edelweiss discuss a number of other such implementations.

Work has been carried out at Liverpool Hope for a number of years on modelling historical data and displaying it through time maps. (Farrimond et al 2001a, Farrimond et al 2001b). The original models were stored as XML files (Bosak and Bray, 1999), which, while providing a mechanism for storing our data to any level of complexity, made the data quite inaccessible for carrying out other queries. Consequently it was decided to examine the possibilities of using temporal database technology to improve the accessibility of the temporal data.

It was recognised that the type of information handled by the historian can describe unlimited numbers of types of entities and unlimited numbers and types of characteristics for those entities. Hence, a temporal data model was sought which could encompass these features. The resulting proposed temporal data model is described in Section 2.

The production of software tools accessible to the non-technical historian has been a significant objective of the work. Consequently, the decision was taken to explore the possibility of using readily available software that is likely to be already in the possession of the historian and so the temporal database described in this paper is implemented on the Microsoft Access DBMS found within the Microsoft Office Suite (Microsoft Office 2003).

The non-technical historian requires a simple means of data entry and this paper proposes a mechanism using Microsoft Excel spreadsheets, which is also found within the Microsoft Office Suite. The structure of the spreadsheets is explained in Section 3.

In order to demonstrate useful ways of using the temporal database once it has been created, this paper describes how it can be used to generate time maps for use in the teaching and learning of history. Time map generation also illustrates how other data such as Geographical Information System (GIS) data may be integrated with data from the temporal database. Time maps are described in Section 4. The generation of time maps from temporal databases and from GIS data is described in Section 5.

Finally, in Section 6, the issue of how the temporal database may be used to answer more general temporal queries is addressed.

2 The temporal data model

2.1 Temporal databases

A temporal database is a database that contains time-stamped data. It associates the values stored in the database with particular times. Two kinds of timestamp have been identified. These are valid time and transaction time (Date, 2000). Valid time refers to recording the time at which a data item is believed to have been true. e.g. that the king of England from 1509 to 1547 was Henry VIII. Transaction time refers to recording the time when the data item was added to the database. Valid time may be modified as new information comes to light and historical interpretations change. However, transaction times cannot be changed like this since the database system itself records when data is added and maintains an audit trail. A change of interpretation or the discovery of new sources resulting in a modification of valid time for a piece of data will result in a modified record being present alongside the superseded record. The modified record will be time stamped with the time the change of interpretation was entered into the database (transaction time). The superseded record will remain time stamped with its earlier transaction time value.

It will be readily accepted that the ability of a database to maintain these audit trails will be invaluable to commercial organisations. However, historians may not see the relevance for their academic work. The work described in this paper does not take into consideration the need to record transaction time in order to explore valid time without causing unnecessary confusion.

It must be pointed out though that transaction time can play a very important role for the historian beyond the maintenance of audit trails. A type of historical "transaction time" can be discerned if the database recorded when a historian, with evidence, declared a data item to be true or changed a previous interpretation of the data item. If this type of transaction time was implemented then the changes of historical interpretation could form historical "audit trails" within the database and their analysis could well lead to new historical insights.

Strategies for storing data in databases are referred to as data models. Many data models have been proposed for temporal databases (Jensen et al, 1996). This paper proposes a radically new model derived from previous work creating historical models in XML (Farrimond et al, 2001a). The rest of this section describes a typical traditional relational data model used to describe the Cromwellian campaign in Ireland and shows how the same information can be stored in the new model.

2.2 Traditional data models

Relational databases have been used for some time in historical research. The aim has been to produce well-structured databases that meet a specific requirement (Harvey and Press, 1996). In the particular example concerning Cromwell and his armies in Ireland, a data model such as that shown in Fig. 1 may be produced:

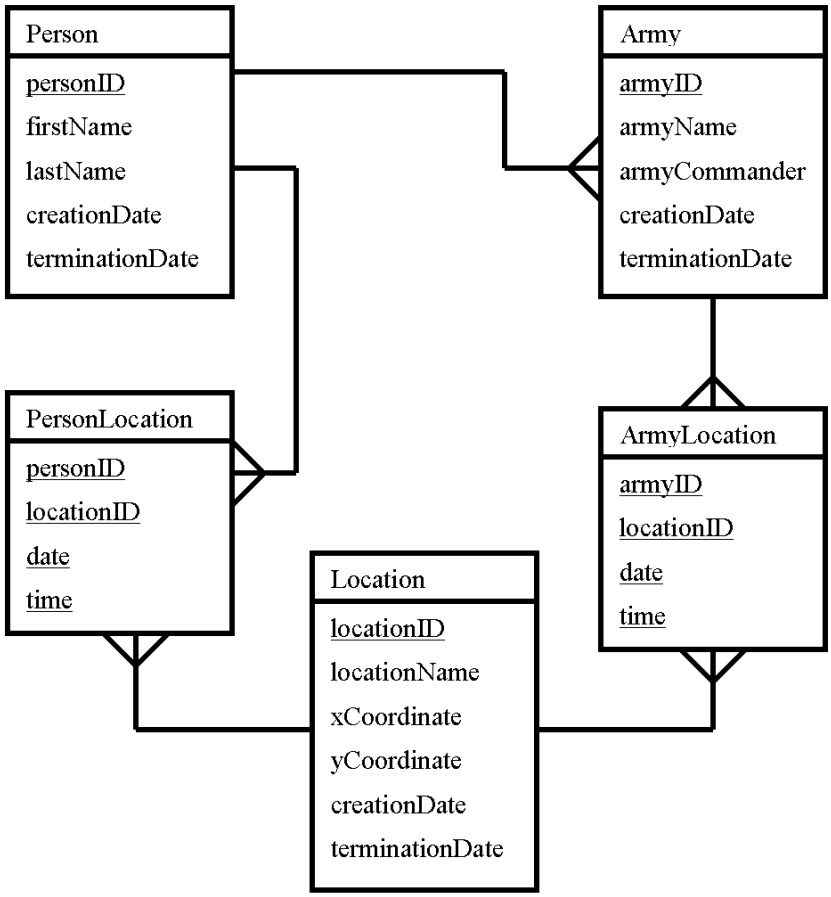


Fig. 1: Traditional data model for Cromwell in Ireland

Fig. 1 is an entity-relationship diagram (Chen, 1976). Some of the boxes in Fig. 1 represent entities of interest (persons, armies and locations) that are to be recorded in the database. Other boxes represent entities that help to provide links between entities of

interest (in particular, links between persons and locations in the case of the **PersonLocation** box and between armies and locations in the case of the **ArmyLocation** box). A box contains the entity name in the upper section and a list of attributes in the lower section. Those attributes that uniquely identify an instance of an entity are underlined forming what is known as a primary key. Hence *personID* (an ID number for the person) identifies unique persons while *locationID* (an ID number for the location) identifies unique locations. Note that the entities **PersonLocation** and **ArmyLocation** each have four attributes underlined indicating that a combination of four attributes are needed to identify a unique instance. Consequently, a particular *personID* value can appear in any number of different **PersonLocation** entity instances indicating that a particular person can be in a number of places over time.

The lines joining the boxes represent relationships between the entities. An important feature of a relationship is the number of entities involved. This feature is known as the **degree** of the relationship. The degree can be **one-to-one**, **one-to-many** or **many-to-many**.

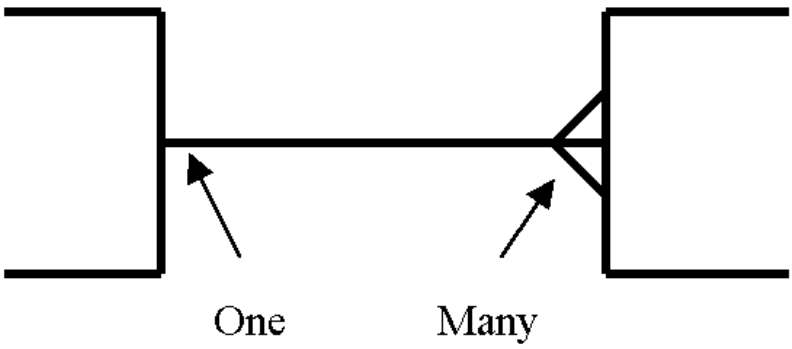


Fig. 2: A one-to-many relationship

Fig. 2 shows that the one side of a relationship is shown as a single line entering the entity while the many side is shown as three lines entering the entity. In Fig. 1, for example, the relationship between **Person** and **Army** is one-to-many. It is read as one person can be related to many armies but each army is related to one person. The one-to-many relationship is realised through adding the primary key attribute values of the entity on the "one" side to the entity on the "many" side. In this case, values of the primary key attribute of the **Person** entity, which is *personID*, appear in the *armyCommander* attribute of the **Army** entity. A particular value of *personID* can appear in any number of **Army** instances. In contrast, the presence of the *personID* value in the **Army** entity means that there is only ever one commander of an army in this scenario. In practice, one would expect that an army would have more than one commander over its lifetime. However, the restriction to a single commander here is used to illustrate the principle of this degree of relationship.

Now a person can be in many locations over time and a location can play host to many people. Hence there is a many-to-many relationship between person and location. This is implemented in the diagram by using a linking entity - **PersonLocation**. A *personID* value can appear in many instances of the **PersonLocation** entity and so can a *locationID*. Hence a person can appear in several locations and several people can visit a location. A particular location can be visited several times by a person and this will be recorded by **PersonLocation** instances having the same values for the *personID*, *locationID* pair but different values for date and/or time of day.

If new data to be entered into the database cannot be catered for by the entities modelled in Fig. 1 then new entities would have to be added to the database and new

relationships established. If an appropriate entity does exist in the database but the characteristics we wish to add are not catered for by the current attributes available then extra fields must be added to the tables. This could lead to unwieldy tables with many fields. It may also result in sparsely populated tables in which records exist that do not require all the attributes available in a given entity. The requirement to be able to add tables and fields forces the historian to become a database administrator.

2.3 Proposed new data model

This paper proposes a new data model that removes the need to add tables and fields to extend the database. This new model uses just three tables yet allows unlimited types of entity to be stored along with unlimited types of attributes. The values of each of these attributes are date stamped so that the model is in fact a temporal data model. The new data model is shown in Fig. 3.

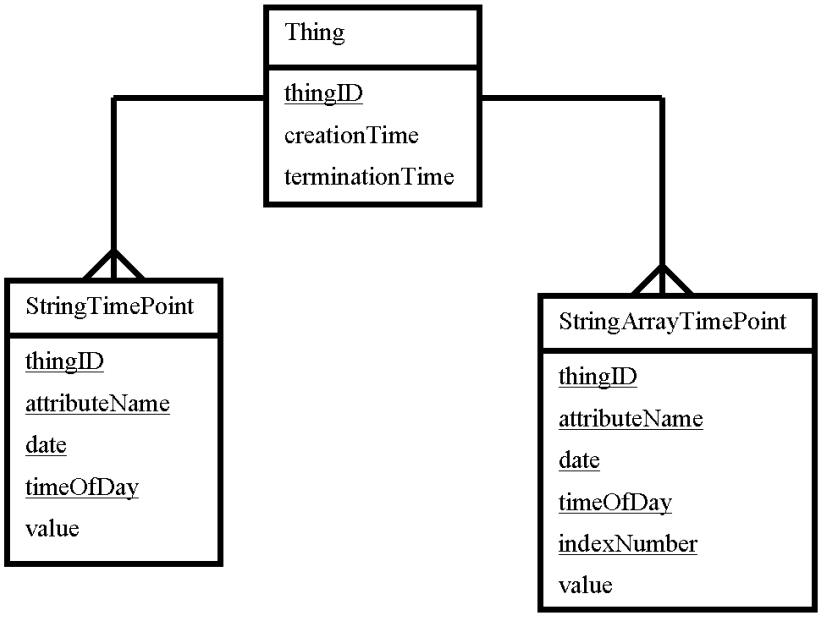


Fig. 3: The new data model

The new data model regards any entity (referred to as a "thing") as being describable in terms of attributes made up of either strings of characters (known as strings for short) or arrays (ordered lists) of strings of characters. The strings may contain numeric data such as size and location or they may contain text such as names and descriptions. A particular attribute may have different values for an object over time. These values are recorded as **timepoints**, which consist of thing ID, attribute name, date and time of day and the value at that time. The timepoints are represented by records in the **StringTimePoint** table.

Some attributes need to be expressed as an array of items. Each element in the array for a given time will be expressed as a record in the **StringArrayTimePoint** table. An example would be the sequence of points representing a river in a GIS system. In the new data model, the river's path would be represented as a collection of records in the **StringArrayTimePoint** table.

Fig. 3 indicates that a **Thing** entity instance may be related to many **StringTimePoint** instances and many **StringTimePointArray** instances through the two one-to-many relationships. Each **StringTimePointArray** entity instance and each **StringTimePointArray** entity instance is related to just one **Thing** entity instance.

Storing array data in this way is very inefficient in that each coordinate requires a separate record. Future work will focus on how to enhance the data model to increase the efficiency without loss of generality and simplicity of concept.

2.4 Features of the new data model

Attribute level time stamping

The new data model time stamps the historical objects' attributes. If a thing does not change over time then it will have just one **StringTimePoint** instance for each attribute for the thing and it will have a set of **StringArrayTimePoint** instances with the same time stamp for each array type attribute. If a thing changes just one non-array attribute just once then a single **StringTimePoint** instance will be added. If a thing changes one array attribute just once then a set of **StringArrayTimePoint**s will be added - one for each element of the modified array.

Recording time in the database

The time stamps are recorded in the database as they were written in the spreadsheet. They could be in any format and the historian producing the spreadsheet governs the choice. It is up to the applications using the database to be able to interpret the formats used.

Temporal granularity

Deciding the timestamp granularity, i.e. the accuracy to which time is recorded, is not an issue in the database since time and date are recorded as written by the historian. It becomes an issue for applications using the temporal data. For example, the application that generates the time maps, described later in this paper, reads the database records and generates an internal representation of the data in which the time and date values are translated into Julian dates as used by astronomers (Norton 1989). The sequence of timepoints for a particular object's attributes can then be stored in time order in the internal representation. The internal representation is then used to generate time map files

for interactive display. The temporal granularity of the map can then be set within the time map editor (Farrimond et al, 2001b) appropriate to the use to which the time map is to be put. The granularity determines the duration of the step which the map takes when map time is moved on one step.

Intervals

A significant issue in temporal databases is the distinction between the time at which an event took place and the interval of time for which a particular state persisted. An interval could be recorded by giving an attribute value two time stamps: the start of the interval and the end of the interval. Date and Darwen (Date, 2000) argue that a better solution is to introduce intervals as a single data type. In our database, we only record when the attribute took a new value. The application software is then responsible for determining the value of the attribute after the timestamped time. The generation of time maps, for example, makes use of a limited number of specific attributes. In some cases, it will assume that the value stays the same until a later timepoint changes it to a new value. An example is the change of name of a town. The town's name stays the time as time progresses until the time comes for the new name. At this point the name suddenly changes to the new name. In other cases, if there is a subsequent value, it will be assumed that linear interpolation takes place at times between the two timepoints. In the Cromwell example, the parliamentary army was in Dublin on 31 August 1649 and in Drogheda on 2 September 1649 and linear interpolation will be applied. The time map will show the army moving steadily between the two towns during this time interval.

Events

The flexibility of the data model enables us to identify events, which happen at an instance of time by attribute name with an optional attribute value. In the example database we use the attribute name *creation* to indicate the formation of detachments from Cromwell's army and the attribute name *termination* to indicate their disappearance as they are reabsorbed back into the main body of the army. The database user will be able to retrieve easily all the attribute names to assist in the formation of queries.

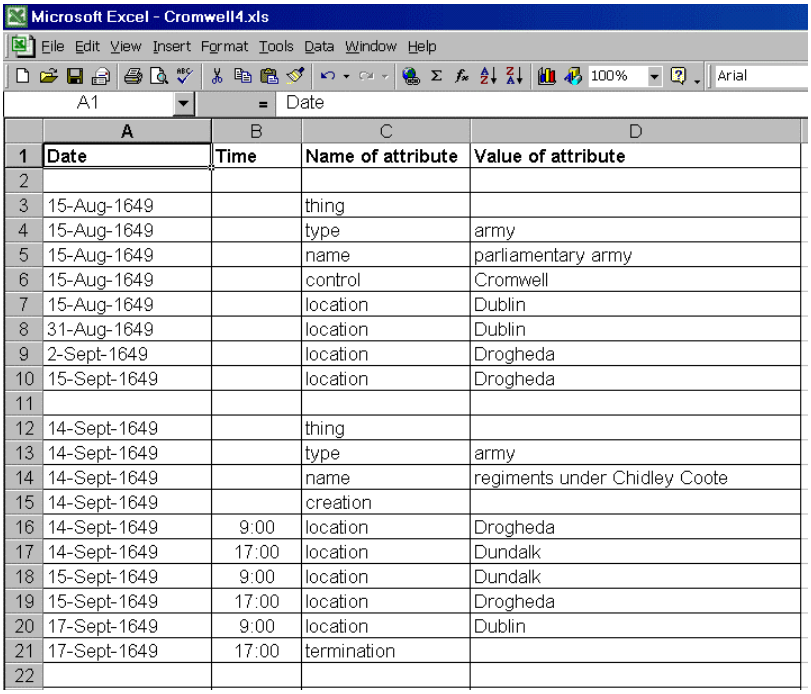
Database management system facilities

The temporal database lacks facilities to insert, update and delete with temporal consistency since it cannot impose temporal constraints such as preventing a person from being in two places at the same time. We look to future implementations of temporal database technology such as TSQL (SQL with temporal extensions) (Snodgrass, 1995) to provide this. In the meantime, the applications that use the temporal database must provide the consistency checking.

Having now described the way in which temporal data is to be stored, the next section describes how the historian can enter data into the database.

3 Data entry to the temporal database

In this section we describe how the historian user sets down the data in the form of an Excel spreadsheet that will be imported into the temporal database. This is a quick and straightforward method that can cope with a variety of ways of entering the information, providing the attribute names and values are correctly given. This allows the user to work thematically or chronologically or switch between the two. Errors in data entry can be easily rectified by returning to, and amending, the spreadsheet.


Microsoft Excel - Cromwell4.xls

File Edit View Insert Format Tools Data Window Help

A1 = Date

	A	B	C	D
1	Date	Time	Name of attribute	Value of attribute
2				
3	15-Aug-1649		thing	
4	15-Aug-1649		type	army
5	15-Aug-1649		name	parliamentary army
6	15-Aug-1649		control	Cromwell
7	15-Aug-1649		location	Dublin
8	31-Aug-1649		location	Dublin
9	2-Sept-1649		location	Drogheda
10	15-Sept-1649		location	Drogheda
11				
12	14-Sept-1649		thing	
13	14-Sept-1649		type	army
14	14-Sept-1649		name	regiments under Chidley Coote
15	14-Sept-1649		creation	
16	14-Sept-1649	9:00	location	Drogheda
17	14-Sept-1649	17:00	location	Dundalk
18	15-Sept-1649	9:00	location	Dundalk
19	15-Sept-1649	17:00	location	Drogheda
20	17-Sept-1649	9:00	location	Dublin
21	17-Sept-1649	17:00	termination	
22				

Fig. 4: Fragment of the Cromwell in Ireland spreadsheet

Fig. 4 shows a fragment of the spreadsheet recording the movements of Cromwell and his armies in Ireland. The first two columns in the spreadsheet record the date and time of day that provide the time stamps for the attributes. The third column gives the attribute names. In the case of string attributes, the fourth column gives the attribute

value. In the case of string array attributes, each column from the fourth column onwards can record an element of a string array.

The thing to which an attribute belongs is determined by the special attribute name **`thing`**. All attribute rows following the appearance of the attribute name **`thing`** belong to the same thing until the next appearance of the attribute name **`thing`** or the end of the spreadsheet. The application software using the database may determine that two recorded things are in fact the same thing in real life. This is done, for example, in the generation of time maps by assuming that if two things have the same value for the name attribute and are of the same type then they are the same thing in real life.

The Excel spreadsheet is saved from within Excel as a text file. A software tool is used to import the data in the text file into the temporal database. Once the data is in the temporal database, it is ready for use. The next section introduces one usage, which is to display temporal data in the form of time maps.

4 Time maps

4.1 What is a time map?

The first use of the temporal database has been to combine it with GIS data to generate two dimensional time maps. A time map is an electronic map with controls that can move the map objects back and forth through time. The objects are grouped into layers, which can be manipulated independently.

Time maps can be created either graphically through the TMap editor (Farrimond et al 2001a)(Farrimond et al 2001b) or through the software tools accompanying the historical temporal database described later in this paper.

The time maps are stored in XML format, which can be read and displayed by a Java Applet, embedded in a web page (Budd, 2000). Hence the maps can be made accessible to colleagues, students and the general public over the Internet.

The web page contains controls for changing the map time, zooming in and out and hiding and revealing layers. Names of objects can also be displayed or hidden as required.

4.2 The Cromwell in Ireland Time Map

The time map generated from the Cromwell in Ireland temporal database depicts the course of the Cromwellian conquest of Ireland over a two-year timeframe. Fig. 5 shows the Cromwell in Ireland time map when it is first opened.

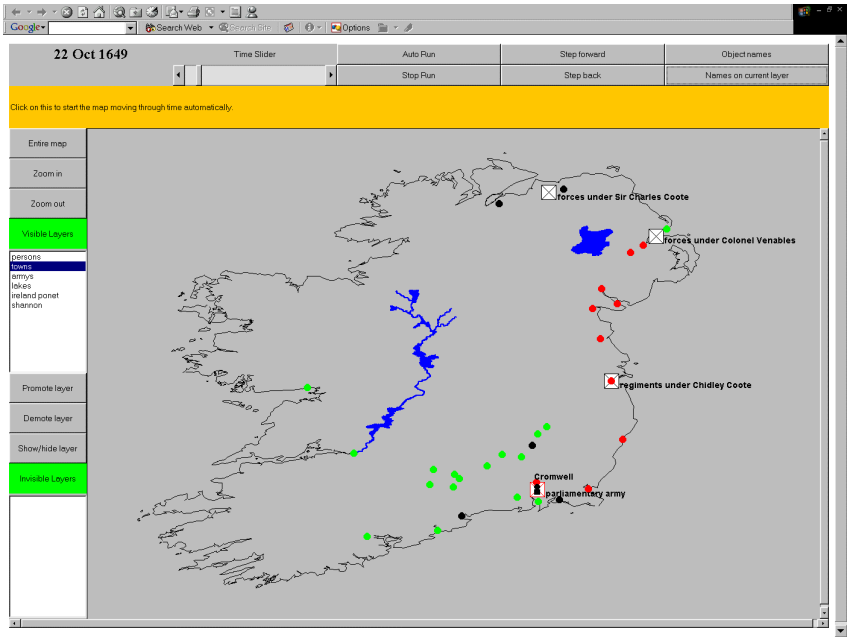


Fig. 5: Time map of Cromwell in Ireland on first opening

The map time (shown currently as 22 October 1649) is displayed at the upper left. The layer controls are down the left hand side. The user can display or hide layers and change the order in which they are painted. The time controls and the label controls are along the top. Time can be changed by using the scroll bar (known as a time slider) or by using the buttons that allow time to run freely or be single stepped forward or backwards. The other buttons are used to control the display of object labels.

The time map serves as a dynamic means to portray the changing characteristics of individuals, groups and locations (for instance, changes in leadership of armed forces or political control of geographical areas) as well as the movement of moveable 'things'.

The use of layers permits the time map to show or hide different features if required, enabling it to serve particular thematic requirements. The timesteps can be set to suit the user's requirements and the zoom facility provides greater clarity where the map is particularly detailed as shown in Fig. 6.

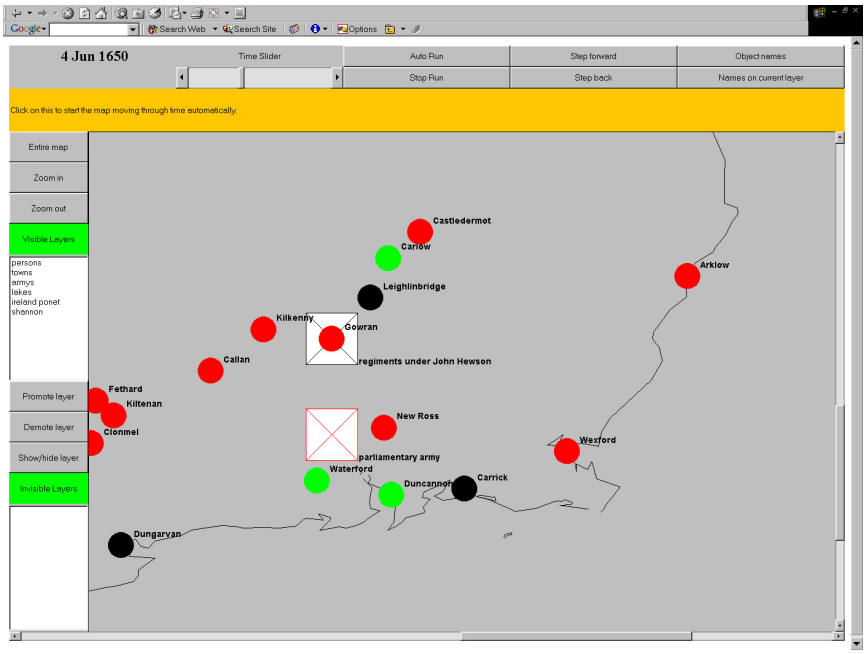


Fig. 6: South East Ireland on 4 June 1650

The time map is therefore a very flexible tool, tailored to particular requirements, which frees the user from reliance on existing, often unrelated, maps that are not drawn to the same scale and/or that do not display information coherently. The user is able to select a specific timeframe and number and type of objects, fixed and moving, and decides on the appropriate level of detail within the constraints of the GIS data. The user can add additional data at a later stage to enable the map to develop in accordance with changing requirements. Accessible via the web, students can work through it outside class at their own pace.

The use of a temporal database to provide data moves the time map away from its earlier, time consuming, method of construction, which required the user to either sketch points or trace over a scanned image in the time map editor, TMap (Farrimond et al, 2001).

The next section describes how the data in the temporal database is used to generate time maps.

5 Using the temporal database to create time maps

5.1 System architecture

Fig. 7 below shows the architecture of the system used to create the temporal database and to generate time maps.

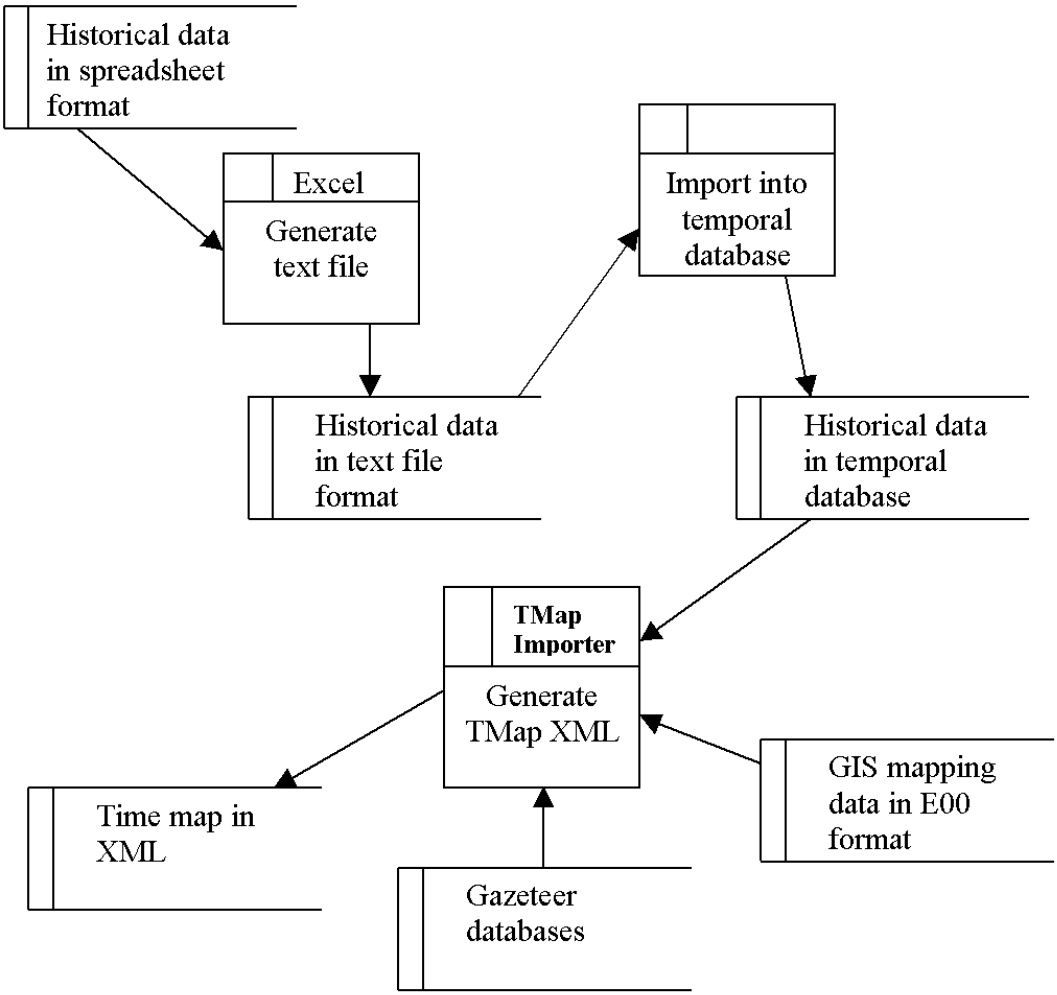


Fig. 7: System architecture

The process of creating the temporal database from spreadsheets was described earlier in Section 3. The software tool, *TMap Importer*, described in Section 5.2,

generates time maps from the data in the temporal database in combination with GIS data, making use of electronic gazeteers to locate places mentioned in the historical data.

5.2 Generating time maps

5.2.1 Importing data from temporal databases

TMap Importer looks for certain key attributes in the temporal database in order to build its time map XML. These are:

- **`location`**: textual name of place e.g. "Dublin"
- **`locationx`**: coordinate of position in decimal degrees of longitude
- **`locationy`**: coordinate of position in decimal degrees of latitude
- **`size`**: to give scale for symbols used if it is desired to represent, for example, the size of a city or army by symbol size.

The user is also able to choose to associate map-drawing colours for each value of a selected attribute. For example, suppose the allegiance of things is modelled by the attribute **`allegiance`** and the user wishes to use colour to represent the different allegiances on the map. The user selects **`allegiance`** causing each different allegiance value in the database to be automatically allocated a different colour. The user can then modify these colours.

To convert location names into latitude and longitude, the user selects GIS gazeteers in the form of databases derived from text files available with Digital Chart of the World (DCW) (ESRI, 2003). When a place name is read in from the temporal database as a **`location`** value, the gazeteers are searched for this name. If there are multiple occurrences of a location name (e.g. there are at least thirteen places called Cashel in

Ireland) then the user is shown a map locating the places already identified plus the locations of the candidate places for the current location name. The user selects the correct place and *TMap Importer* will use this one. The user can select that all subsequent appearances of this place name use the same location. If a location is missing from the database, the user is invited to click on the map generated by *TMap Importer* to identify its location. This map shows the places already identified to help the user in the expectation that the user knows its location relative to these places. Alternatively, the place may be entered into the database from the spreadsheet as another thing with attributes **`locationx`** and **`locationy`** giving its position.

In the cases of **`locationx`**, **`locationy`** and **`size`**, linear interpolation takes place for times between timepoints. If a moveable object such as a person or an army stays in a place for an interval of time before moving elsewhere, the spreadsheet writer must record the start time of the interval and the end time of the interval as separate timepoints to keep the object from moving by default gradually towards the next recorded location.

5.2.2 Importing data from GIS files

TMap Importer can read Digital Chart of the World GIS files in E00 format (Letham 2003). These are text files that are used to port data between GIS applications. *TMap Importer* extracts arc information from these files. In particular, *TMap Importer* makes use of the *ponet* (political network) files and the *dnnet* (drainage network) files.

The Digital Chart of the World data is at a scale of 1:1 million, which, although not sufficiently accurate for quantitative work, is quite sufficient for the use of our time map technology in teaching and learning history.

The GIS data is not time stamped so *TMap Importer* stamps the data with a user selected time.

5.2.3 Outputting TMap time maps

Once the data has been imported into *TMap Importer*, it is ready to generate either time map layers or else complete time maps.

When creating a complete time map, the user selects a start time and a stop time for the map. *TMap Importer* will automatically select a map origin, height and width to encompass the imported data. These settings can be modified later as described in Section 5.2.4.

It is sometimes more convenient to add data from a number of files to a map. This can easily be achieved by generating separate TMap layer files from the imported data rather than complete map files. The layers can then be combined into a single TMap time map.

The method for a user to combine several layers into a time map is for the user to create a time map from one of the imported data files and then select it as the "master" map file in the combination tool within *TMap Importer*. The user then selects the TMap layer files that he wishes to add to the master map.

5.2.4 Refining TMap maps with TMap Editor

Once the map has been made, the user can edit the map in the *TMap Editor* (Farrimond et al, 2001b), which contains tools, which allow the user to refine the map. Typical edits include:

- reducing the number of points on an arc to reduce file size;

- changing the colour and width of arcs;
- changing the map height, width and coordinate origin
- changing the map start and end times, the time step unit and the format of displayed time
- moving items between layers
- modifying the positions of map items
- adding extra items
- deleting unwanted items

6 Querying the database directly

We are able to use query languages such as SQL (Date, 2000) to query the database from within Access. However, temporal queries are difficult to express in SQL (Date, 2000). Extensions to SQL have been proposed that make it much easier to write temporal queries. TSQL is one example (Snodgrass, 1995). The TimeDB approach of converting temporal SQL statements into standard SQL statements look promising and offer an avenue of research while awaiting the arrival of commercial temporal databases (TimeDB, 1999). A further step would be to enable the historian to ask questions in natural language and have the interface translate them into appropriate SQL to query the database. This would remove the need for the historian to learn a new database query language.

7 Conclusion

Temporal databases offer significant possibilities for the historian. However, if it is to be useful, the database must be flexible and it is important that data entry is historian centred. The data model and the spreadsheet method of data entry described in this paper offer a simple but powerful solution.

A new temporal data model is proposed that offers complete flexibility without requiring the historian to become a database administrator. Applications can access the data in the temporal database and integrate it with data from other sources such as GIS to produce useful views on the data.

Future work will include the transfer of the data model to temporal DBMS as they become available to improve its robustness. It is important that mechanisms are added for appending data to an existing database. With respect to this, issues concerning unique identifiers require further research.

In order to increase the usefulness of the temporal database, further query interfaces need to be developed. The mechanism of converting queries expressed in temporal SQL into standard SQL queries offers immediate opportunities for further investigations into the development of useful interaction tools for the historian. Longer term possibilities include the development of natural language interfaces that avoid the historian having to learn a database query language. This would complement the historian centred data entry mechanism via spreadsheet.

8 References

Bosak, J. and Bray, T. (1999). XML and the Second-Generation Web, *Scientific American*, May 1999 pp 79-83

Budd T. (2000). *Understanding Object-Oriented Programming with Java*, Addison Wesley

Chen P. (1976). *The Entity-Relationship Model - Toward a Unified View of Data.* ACM Transactions on Database Systems 1(1)

Date, C. J., (2000). *An Introduction to Database Systems* 7th Edition, Addison Wesley

ESRI (2003). *Digital Chart of the World on CD ROM*, http://www.esri.com/data/catalog/esri/esri_dcw.html (last accessed 8 November 2003)

Farrimond, B., Parkinson, L., and Pogson F. (2001a). Modelling history with XML, *DRH 2001 and 2002*, OHC, London. J. Anderson, A. Dunning and M. Fraser Ed

Farrimond, B., Parkinson, L., and Pogson F. (2001b). Teaching with Dynamic Maps on the Web, *IBITE 2001 International Conference on Information Technology in Higher Education,* Eindhoven

Harvey, C. and Press, J. (1996). *Databases in Historical Research*. St Martin's Press

Hubler, P. N. and Edelweiss, N. (2000). *Implementing a Temporal Database on Top of a Conventional Database: Mapping of the Data Model and Data Definition Management.* XV Simpósio Brasileiro de Banco de Dados, 2-4 October 2000

Jensen, C. S., Snodgrass, R. J., Soo, M. D. (1995). The TSQL2 Data Model, *The TSQL2 Temporal Query Language* **Snodgrass R. J. (Ed),** Kluwer

Letham G. (2003). *Working with E00 data*. http://spatialnews.geocomm.com/education/tutorials/e00data/ (last accessed 30 October 2003)

Microsoft Office (2003). *Microsoft Office Online*. http://www.microsoft.com/office/ (last accessed 8 November 2003)

Norton, A. P. (1989). *Norton's 2000.0 Star Atlas and Reference Handbook (Epoch 2000.0)* 18th Edition Ed. Ridpath I.

Oracle (2003). *Oracle Database,* http://www.oracle.com/ip/deploy/database/oracle9i/ (last accessed 8 November 2003)

Snodgrass, R. T. (Ed.) (1995). *The TSQL2 Temporal Query Language.* Kluwer

TimeDB (1999). *TimeDB - A Temporal Relational DBMS* , http://www.timeconsult.com/ (last accessed 9 November 2003)

Authors

Brian Farrimond

Brian Farrimond is a Senior Lecturer in IT at Liverpool Hope University College. He has been developing time map software since 1999 and has presented a number of papers on his work:.

Farrimond B, Parkinson L and Pogson F (2001), Modelling History with XML, *DRH 2001: Panel Session at Annual Digital Resources for the Humanities Conference.*

Farrimond B. (2001) Using Time Maps based in XML to teach History, *ITTE 2001: Annual Conference of the Association of Information Technology for Teacher Education*

Farrimond B, Parkinson L and Pogson F (2001),Teaching with Dynamic Maps on the Web, *BITE 2001: Bringing Information Technology to Education.*

He is a committee member in the Association for History and Computing (UK Branch).

He is leader of the Temporal Modelling Research Group based at Liverpool Hope (http://www.hope.ac.uk/imc/tmrg)

Dr Fiona Pogson

Dr Fiona Pogson is currently Head of History at Liverpool Hope. Her research interests include seventeenth-century political, religious and administrative history; in particular, court politics in the reign of Charles I. Besides contributing to several papers on time maps and using them in her undergraduate teaching, Fiona's published papers include:

'Wentworth as President of the Council of the North', in John C. Appleby and Paul Dalton (eds.), *Government, Religion and Society in Northern England, 1000-1700* (Stroud, 1997).

'Wentworth, the Saviles and the Office of *Custos Rotulorum* of the West Riding', *Northern History* 34 (1998), 205-210.

'Making and Maintaining Political Alliances during the Personal Rule of Charles I: Wentworth's Associations with Laud and Cottington', *History* 84 (1999), 54-73.

''Wentworth and the Northern Recusancy Commission', *Recusant History* 24 (1999), 271-287.

Fiona has also contributed to the following papers:

Farrimond B, Parkinson L and Pogson F (2001), Modelling History with XML, *DRH 2001: Panel Session at Annual Digital Resources for the Humanities Conference.*

Farrimond B, Parkinson L and Pogson F (2001),Teaching with Dynamic Maps on the Web, *BITE 2001: Bringing Information Technology to Education.*

Lindy Parkinson

Lindy is a Lecturer in IT at Liverpool Hope University College. She has recently completed her Masters in IT with a dissertation entitled *An Innovative Approach to the Storage of Historical Data.*

Lindy has contributed to two papers:

Farrimond B, Parkinson L and Pogson F (2001), Modelling History with XML, *DRH 2001: Panel Session at Annual Digital Resources for the Humanities Conference.*

Farrimond B, Parkinson L and Pogson F (2001),Teaching with Dynamic Maps on the Web, *BITE 2001: Bringing Information Technology to Education.*

Colette Redmond

Colette is a Graduate Assistant in IT at Liverpool Hope University College.

Dr Steve Presland

Steve is Head of IT at Liverpool Hope University College. He specialises in database analysis and design.