SQL AND HYPERTEXT GENERATION OF STRATIGRAPHIC ADJACENCY MATRICES

Introduction

The process of excavating and recording an archaeological site by single deposits, where at all possible, is now generally accepted (Harris 1989, 18-21; Barker 1982, 200-203). Modern archaeological analysis, with its requirements for understanding site formation and process, demand methods of excavation and recording that allow the construction and reconstruction of the depositional history of a site. The excavation recording of single "contexts", units of stratification (Harris 1989, 34-36), offers the only comprehensive system for ensuring this access to the depositional history of a site. Use of single context recording ensures that all data collected on site has a common reference; the basic recording event correlates directly with basic depositional events.

Archaeological interpretation has always referred ultimately to deposition (Barker 1982, 11-12). Deposition is the basic constituent of all archaeological constructions by the nature of the record. Archaeologists do not have access to events, but to the depositional consequences of events. This creates a primacy of deposition and stratigraphy to which all other data must ultimately relate.

Though archaeological excavation and recording has, over the last 30 years, adapted to this change in perspective, the creation of the site report has continued to assume a relationship with the field record based more on abstract structures than on the interpretive process by which these structures are created. Site reports continue to offer the reader grand designs and self-evident constructions with no easy recourse to question or reassess these claims.

The present state of Information Technology may offer some solutions for this problem. Through Hypercard, or Hypercard like systems (Grace, Orton 1989), it is now possible to relate different components of the data within a single document. This allows one component of data to be summoned from any other component where a relation is identified. Plans may be called from within text, attributed held in a database may be called from a plan, and the primary site record may be called from within a stratigraphic model. This provides a multi-layered structure for the information objects, removing the false segregation of data from interpretation (Fig. 1). While reading the site report, assertions or queries may be checked by direct and immediate consultation of the plans, the data, the stratigraphic matrix or the site record. Any entity that can be stored on computer can be related to other entities from the site record or

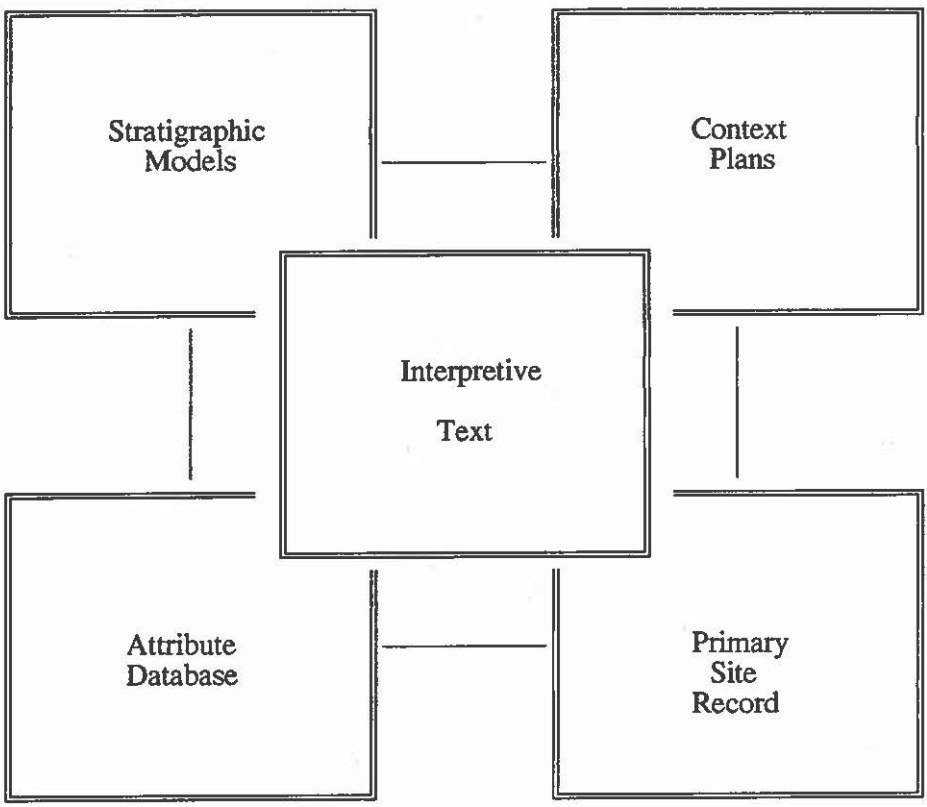


Fig. 1 — Interrelationships between entities of recorded and interpretive information.

report. These information entities may be called individually or in combination based on the criteria set by the reader rather than solely on those of the author. The limitations on what should be stored is governed only by the size and speed of the computer system.

It is not the purpose of such a system to offer "hi-tech" storage, but to offer a system of reporting that allows for the interactive deconstruction of that primary interpretive entity of archaeology — the site report. By relating these recorded entities to the arguments of their significance, and to other entities, their role in defence and critique becomes more direct and disputable. The interpretive movement from record to construct is more traceable and assessable.

The program presented here is but the earliest step in exploring such systems. Through the creation of a provisional stratigraphic matrix (Harris Matrix) from the recorded context adjacencies, it would be possible to use such a stratigraphic model to access other data related to its meaningful interpretation.

The Harris Matrix (HARRIS 1989, 34-39) has been the preferred method for presenting the non-redundant relationships between contexts since the late 1970's (Fig. 2). Its principles are simple and basic, and should lend themselves to computer generation (BISHOP, WILCOCK 1986). The Harris Matrix is a 2-dimensional model of the non-redundant adjacencies between 3-dimensional stratigraphic units. Harris defines the location of a unit of stratification as: « its position between the undermost (or earliest) of the units which lie above it and the uppermost (or latest) of all the units which lie below it and with which the unit has a physical contact, all other superpositional relationships being redundant » (HARRIS 1989, 34).

We seek to demonstrate how relatively modest developments in data handling techniques may result in significant time savings for archaeologists engaged in the analysis of site records. Moreover we hope that, with further development, the algorithms presented will enable robust and verifiable production of site records in an effective framework for other archaeological datasets.

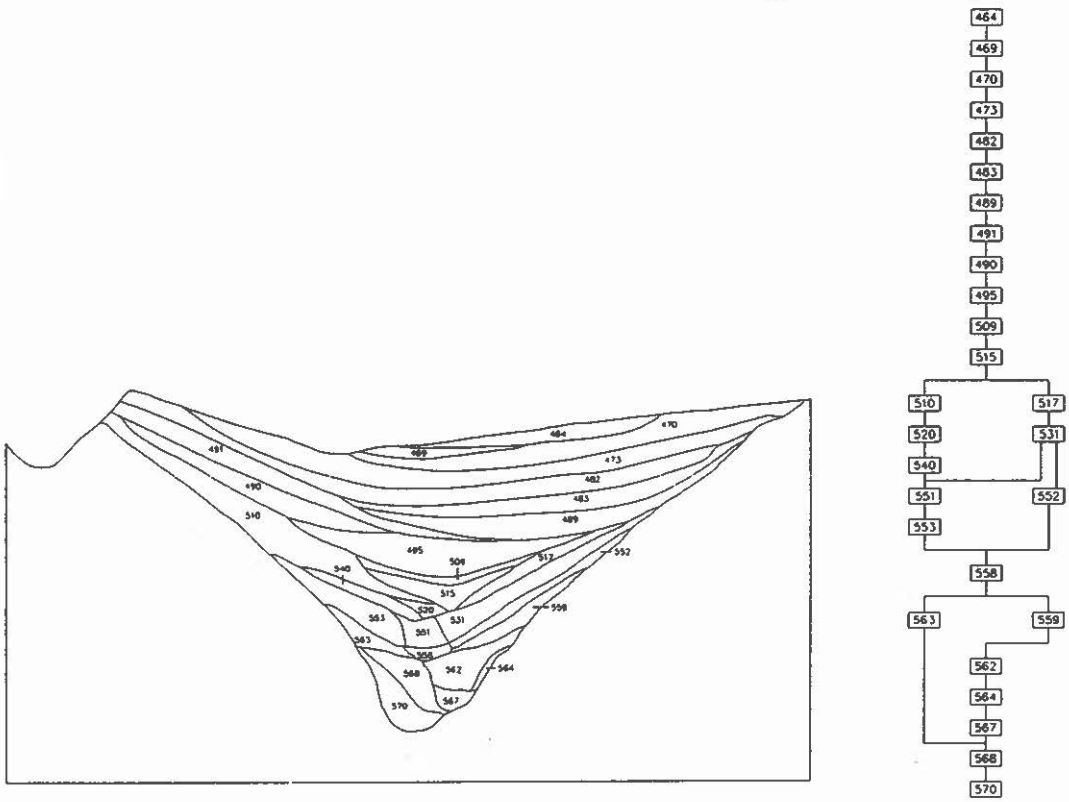


Fig. 2 — Harris Matrix of ditch section (after HARRIS 1989).

System components

The system, implemented in outline form, relies upon recent developments in the provision of graphical interfaces for existing relational database management systems (RDBMS). Many such interfaces are being instigated, largely under initiatives directed towards the development of various Geographical Information Systems. This research draws upon an implementation of the well known Oracle RDBMS mounted on a Apple Macintosh personal computer. This configuration allows the database to be interrogated through a graphical interface provided by a HyperCard style package. The two packages communicate via a Structured Query Language (SQL) which enables powerful database query commands to be embedded in application programs.

The Oracle RDBMS

Any relational database relies upon a data model which views information as a collection of data-tables. Each table may have a number of columns or fields which may hold data in text or numeric formats. Each row in a table constitutes a tuple or record. Tables may be related to each other through the identification of key fields which are common to each table. Through simple set theoretic operations on these tables we are able to implement complex data relations and effect sophisticated and accurate data retrievals. All RDBMS implement a variety of such operations including set UNIONS, PROJECTIONS and JOINS (Harrington 1988, 109-133).

Structured Query Language (SQL)

SQL has emerged as the de facto standard for interfaces to relational databases. It differs from many other computer languages in that it is a non-procedural language. This means that the user has only to specify WHAT data he requires rather than HOW the data should be retrieved. Thus the low level retrieval strategies are left to the RDBMS which will optimise data access taking account of any relevant indices. However SQL is not a panacea for all programming tasks. Its set based approach to data management is not applicable to all data queries. Thus all major RDBMS have facilities which allow the use of embedded SQL in third generation programming languages (such as PASCAL, C, or even FORTRAN). This facility enables the development of flexible computer programs which are, to a degree, independent of the RDBMS chosen (Harrington 1988, 179-207).

HyperCard/SuperCard

Oracle for the Macintosh has been fully integrated with the HyperCard

which provides a graphical interface to the RDBMS. HyperCard supports objects oriented programming features through small code fragments, called scripts, which enable navigation through the HyperCard environment and the development of simple programs whilst maintaining a simple user interface (GRACE, ORTON 1989). The SuperCard product extends the functionality of HyperCard by enabling the definition of complex object oriented graphic structures and the provision of fully configurable pull down menus.

A RDBMS FOR ARCHAEOLOGICAL STRATIGRAPHY

The Relational data model

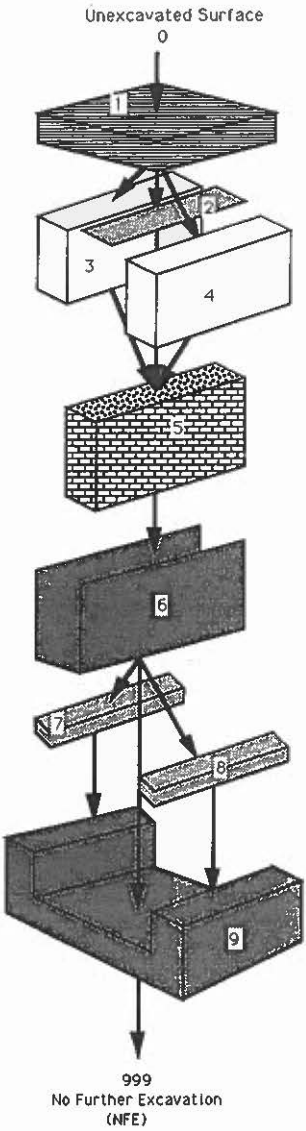
Single unit excavation is, as we have seen, undertaken on the basis of the identification of contexts. One result of excavation carried out under such a scheme is the production of context descriptions which denote the type and form of the deposit. Information about the physical relationship of the unit with other contexts which it may overlie or underlie is also recorded since this is of fundamental importance to the depositional (structural) interpretation of the site. Thus to model these relationships a simple schema for structural interpretation might include a unique context identifier (id) and the identifiers of units which are physically adjacent. To illustrate this scheme we will model the relationships illustrated in the diagram below (Fig. 3) (after HARRIS 1989).

Thus we describe these relationships via two tables context_OV and context_UN, the first relating to relationships A OVerlies B, the second A UNderlies C. Context_ov should, of course, be the compliment of context—un since there should be a one to one correspondence between the tuples of each table. However errors in field coding often mean that the records are rather less complete than one would wish. Therefore the data set used in further analysis is derived from the merging of these two tables. Thus if we describe our two simple tables as:

Table	*Field 1*	*Field 2*
context_ov	id (integer)	overlays (integer)
context_un	id (integer)	underlies (integer)

We achieve our working data set, context_MER, by MERging the two sets using the SQL command:

```
Insert into context_mer (id,overlays)
Select (id,overlays) from context_ov UNION
Select (underlies,id) from context_un;
```



ID	OVERLAYS
0	1
1	2
1	3
1	4
1	5
1	6
1	7
1	8
2	5
3	5
3	6
3	7
3	9
4	5
4	6
4	8
4	9
5	6
5	9
6	7
6	8
6	9
7	9
8	9
9	999

Fig. 3 — Sample dataset.

This command demonstrates the flexibility of SQL in performing a Union of two data sets to create a third whilst transposing the columns of the second set. Of course if all data is correctly coded this command simply results in two entries in context_mer for each physical overlay. Thus the command may be refined to:

```
Insert into context_mer (id,overlays)
Select (id,overlays) from context_ov UNION
(Select underlies,id from context_un where NOT IN
(Select id,overlays) from context_ov);
```

Thus even at this level the utilisation of a RDBMS may offer significant rewards to the archaeologist enabling the validation of data prior to sequencing.

Building the stratigraphic sequence

The context_mer table may now be manipulated to eliminate the redundant physical relationships in an attempt to understand the stratigraphy. The full description of context_mer is given as:

Table	*Field 1*	*Field 2*	*Field 3*
context_mer	id (integer)	overlays (integer)	treelevel (integer)

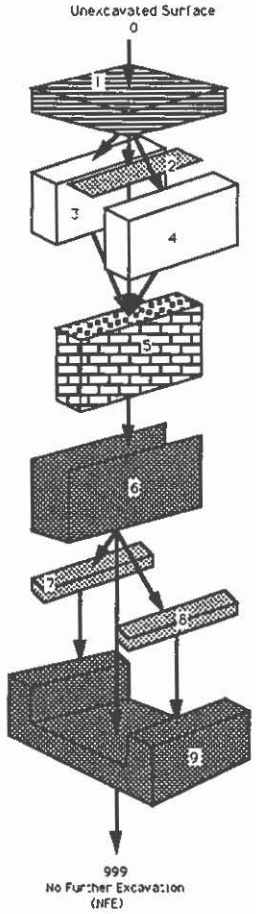
The third field is used in the derivation of the stratigraphic sequence. The derivation of this sequence is explained in the following diagram (Fig. 4).

The stratigraphic sequence may be viewed as a network or graph. The first part of the structuring of this graph is the determination of the longest possible route from the base of the network (context 0) to each of the stratigraphic units. This can be viewed as a sorting of the graphs so that each context is placed at its earliest (deepest) possible occurrence. This is achieved by the SQL statement:

```
Update context_mer set treelevel = &thislevel
where id in
(select overlays from context_mer where treelevel = &thislevel-1);
```

Initialising the graph with a treelevel of zero, for contexts which are not otherwise overlain (id = 0), the local variable &thislevel is incremented until no further changes are made to the table. The bracketed select statement returns the ids of all units which are overlain by units at &thislevel-1 and thus update of the next level is achieved. Since the statement is applied repetitively we find that units are forced to their lowest possible position within the resultant tree.

Unfortunately transcription and interpretation errors in the site coding will often mean that nonsensical linkages may exist in the data set. Figure 5 below shows such a situation where a link from context 7 to context 5 has been added. The effect of this is that all dependent children below this point are forced to ever lower treelevels as the function is repeatedly applied. It is possible to trap such errors by monitoring the number of units updated, however the errors must be isolated before the tree building sequence can be completed.




```
SQL> Update context mer set treelevel=0
where id =0;
SQL> select * from context mer;

        ID   OVERLAYS  TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2
         1          3
         1          4
         1          5
         1          6
         1          7
         1          8
         2          5
         3          5
         3          6
         3          7
         3          9
         4          5
         4          6
         4          8
         4          9
         5          6
         5          9
         6          7
         6          8
         6          9
         7          9
         8          9
         9        999

25 records selected.
```

```
SQL> update context mer set treelevel=1
where id in
(select overlays from context mer where
treelevel=1-1)

7 records updated.

SQL> select * from context mer
where treelevel>Null;

        ID   OVERLAYS  TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2          1
         1          3          1
         1          4          1
         1          5          1
         1          6          1
         1          7          1
         1          8          1
```

```
SQL> update context mer set treelevel=2
where id in
(select overlays from context mer where
treelevel=2-1)

16 records updated.

SQL> select * from context mer
where treelevel>Null;

        ID   OVERLAYS  TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2          1
         1          3          1
         1          4          1
         1          5          1
         1          6          1
         1          7          1
         1          8          1
         2          5          2
         3          5          2
         3          6          2
         3          7          2
         3          9          2
         4          5          2
         4          6          2
         4          8          2
         4          9          2
         5          6          2
         5          9          2
         6          7          2
         6          8          2
         6          9          2
         7          9          2
         8          9          2
```

```
SQL> update context mer set treelevel=3
where id in
(select overlays from context mer
where treelevel=3-1)

6 records updated.

SQL> select * from context mer
where treelevel>Null;

        ID OVERLAYS TREELEVEL
---------- -------- ---------
         0        1         0
         1        2         1
         1        3         1
         1        4         1
         1        5         1
         1        6         1
         1        7         1
         1        8         1
         2        5         2
         3        5         2
         3        6         2
         3        7         2
         3        9         2
         4        5         2
         4        6         2
         4        8         2
         4        9         2
         5        6         3
         5        9         3
         6        7         3
         6        8         3
         6        9         3
         7        9         3
         8        9         3
         9      999         3
```

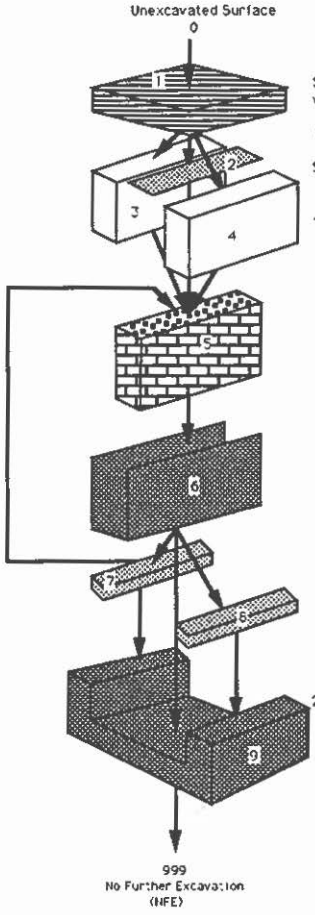
```
SQL> update context mer set treelevel=4
where id in
(select overlays from context mer where
treelevel=4-1)

6 records updated.

SQL> select * from context mer
where treelevel>Null;

        ID OVERLAYS TREELEVEL
---------- -------- ---------
         0        1         0
         1        2         1
         1        3         1
         1        4         1
         1        5         1
         1        6         1
         1        7         1
         1        8         1
         2        5         2
         3        5         2
         3        6         2
         3        7         2
         3        9         2
         4        5         2
         4        6         2
         4        8         2
         4        9         2
         5        6         3
         5        9         3
         6        7         4
         6        8         4
         6        9         4
         7        9         4
         8        9         4
         9      999         4
```

```
SQL> update context mer set treelevel=5
where id in
(select overlays from context mer
where treelevel=5-1)

3 records updated.

SQL> select * from context mer
where treelevel>Null ;

        ID OVERLAYS TREELEVEL
---------- -------- ---------
         0        1         0
         1        2         1
         1        3         1
         1        4         1
         1        5         1
         1        6         1
         1        7         1
         1        8         1
         2        5         2
         3        5         2
         3        6         2
         3        7         2
         3        9         2
         4        5         2
         4        6         2
         4        8         2
         4        9         2
         5        6         3
         5        9         3
         6        7         4
         6        8         4
         6        9         4
         7        9         5
         8        9         5
         9      999         5
```

```
SQL> update context mer set
treelevel=6
where id in
(select overlays from context mer
where treelevel=6-1)

1 record updated.

SQL> select * from context mer
where treelevel>Null;

        ID OVERLAYS TREELEVEL
---------- -------- ---------
         0        1         0
         1        2         1
         1        3         1
         1        4         1
         1        5         1
         1        6         1
         1        7         1
         1        8         1
         2        5         2
         3        5         2
         3        6         2
         3        7         2
         3        9         2
         4        5         2
         4        6         2
         4        8         2
         4        9         2
         5        6         3
         5        9         3
         6        7         4
         6        8         4
         6        9         4
         7        9         5
         8        9         5
         9      999         6
```

```
SQL> update context mer set treelevel=7
where id in
(select overlays from context mer where treelevel=7-1)

0 records updated.
```

Fig. 4 — Building the stratigraphic sequence.




```
SQL> insert into context_rer (id,overlays)
values (7,5);

1 record created.

SQL> select * from context_rer;

        ID OVERLAYS TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2
         1          3
         1          4
         1          5
         1          6
         1          7
         1          8
         2          5
         3          5
         3          6
         3          7
         3          9
         4          5
         4          6
         4          8
         4          9
         5          6
         5          9
         6          7
         6          8
         6          9
         7          9
         8          9
         9        999
         7          5

26 records selected.
```

```
SQL> update context_rer set treelevel=1
where id in
(select overlays from context_rer where
treelevel=1-1)

7 records updated.

SQL> select * from context_rer;

        ID OVERLAYS TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2          1
         1          3          1
         1          4          1
         1          5          1
         1          6          1
         1          7          1
         1          8          1
         2          5
         3          5
         3          6
         3          7
         3          9
         4          5
         4          6
         4          8
         4          9
         5          6
         5          9
         6          7
         6          8
         6          9
         7          9
         8          9
         9        999
         7          5

26 records selected.
```

```
SQL> update context_rer set treelevel=2
where id in
(select overlays from context_rer where
treelevel=2-1)

17 records updated.

SQL> select * from context_rer;

        ID OVERLAYS TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2          1
         1          3          1
         1          4          1
         1          5          1
         1          6          1
         1          7          1
         1          8          1
         2          5          2
         3          5          2
         3          6          2
         3          7          2
         3          9          2
         4          5          2
         4          6          2
         4          8          2
         4          9          2
         5          6          2
         5          9          2
         6          7          2
         6          8          2
         6          9          2
         7          9          2
         8          9          2
         9        999
         7          5          2

26 records selected.
```

```
SQL> update context_mer set treelevel=3
where id in
(select overlays from context_mer where
treelevel=3-1)

9 records updated.

SQL> select * from context_mer;

        ID OVERLAYS TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2          1
         1          3          1
         1          4          1
         1          5          1
         1          6          1
         1          7          1
         1          8          1
         2          5          2
         3          5          2
         3          6          2
         3          7          2
         3          9          2
         4          5          2
         4          6          2
         4          8          2
         4          9          2
         5          6          3
         5          9          3
         6          7          3
         6          8          3
         6          9          3
         7          9          3
         8          9          3
         9        999          3
         7          5          3

26 records selected.
```

```
SQL> update context_mer set treelevel=4
where id in
(select overlays from context_mer where
treelevel=4-1)

9 records updated.

SQL> select * from context_mer;

        ID OVERLAYS TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2          1
         1          3          1
         1          4          1
         1          5          1
         1          6          1
         1          7          1
         1          8          1
         2          5          2
         3          5          2
         3          6          2
         3          7          2
         3          9          2
         4          5          2
         4          6          2
         4          8          2
         4          9          2
         5          6          4
         5          9          4
         6          7          4
         6          8          4
         6          9          4
         7          9          4
         8          9          4
         9        999          4
         7          5          4

26 records selected.
```

```
SQL> update context_mer set treelevel=5
where id in
(select overlays from context_mer where
treelevel=5-1)

9 records updated.

SQL> select * from context_mer;

        ID OVERLAYS TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2          1
         1          3          1
         1          4          1
         1          5          1
         1          6          1
         1          7          1
         1          8          1
         2          5          2
         3          5          2
         3          6          2
         3          7          2
         3          9          2
         4          5          2
         4          6          2
         4          8          2
         4          9          2
         5          6          5
         5          9          5
         6          7          5
         6          8          5
         6          9          5
         7          9          5
         8          9          5
         9        999          5
         7          5          5

26 records selected.
```

```
SQL> update context_mer set treelevel=6
where id in
(select overlays from context_mer
 where treelevel=6-1)

9 records updated.

SQL> select * from context_mer;

        ID OVERLAYS TREELEVEL
---------- ---------- ----------
         0          1          0
         1          2          1
         1          3          1
         1          4          1
         1          5          1
         1          6          1
         1          7          1
         1          8          1
         2          5          2
         3          5          2
         3          6          2
         3          7          2
         3          9          2
         4          5          2
         4          6          2
         4          8          2
         4          9          2
         5          6          6
         5          9          6
         6          7          6
         6          8          6
         6          9          6
         7          9          6
         8          9          6
         9        999          6
         7          5          6

26 records selected.
```

Fig. 5 — The effect of coding errors on the stratigraphic sequence.

At present this can only be achieved automatically by a statement such as:

```
Update context_mer set overlays = overlays + 9000
where overlays in
(select id from context_mer where treelevel = (select max(treelevel) from context_mer)
intersect select overlays from context_mer
where treelevel < (select max (treelevel) from context_mer))
and treelevel = (select max(treelevel) from context_mer
```

This will block all potential error candidates by terminating the tree building procedure through the creation of fictious unit numbers. The context_mer table must then have the treelevel field reinitialised before the tree build is repeated. Several such corrections may be necessary on a large or complex site.

Creation of Harris matrix

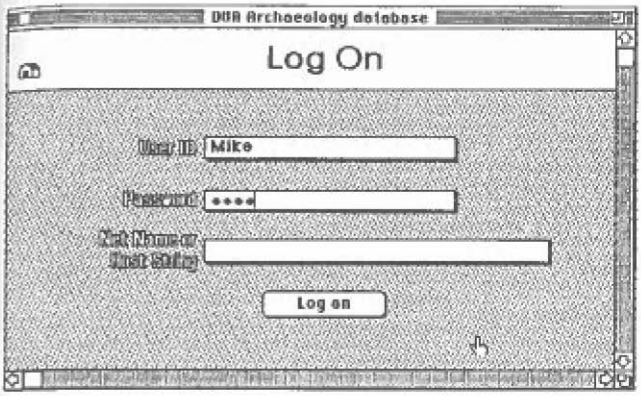
In order to make sense of the large volumes of data generated from complex sites it is desirable that a graphical representation of the site stratigraphy be produced. A standard method for viewing such data is the Harris matrix. In this representation all redundant relationships in the stratigraphy are ignored. Three situations will dictate whether a link between units should be drawn:

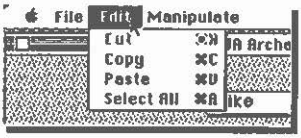
i) if the context has only one child then the link will always be drawn;
ii) if the context has more than one child then links to children at the next treelevel will always be drawn;
iii) if the context has more than one child then links to children which cannot be reached by an alternative route will be drawn.

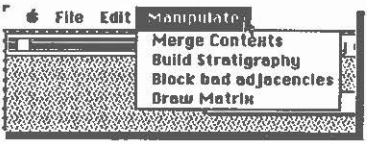
A system has been implemented which allows the identification of cases i) and ii) from the above triad. Case iii) remains, as yet, unresolved. The implementation draws upon a further virtual table which is created as a VIEW of context_mer. This view, called Children, is constructed by the SQL command:

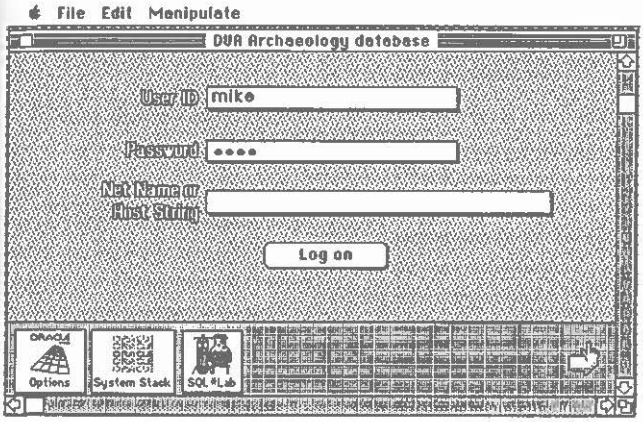
```
Create view Children(id,nochildren) as
select id,count(distinct overlays) from
context_mer group by id;
```

This statement counts the number of children which underlie each context and is used to test for case i) above. In order to draw the matrix a graphical interface to Oracle is required. As we have indicated the SuperCard package provides just such an interface and also allows us to integrate our SQL commands within a familiar Macintosh user environment of windows , icons, mice and pull down menus (WIMPS). The embedded SQL commands which enact the tree building procedure are enabled as special menu items (see Fig. 6 for implementation).









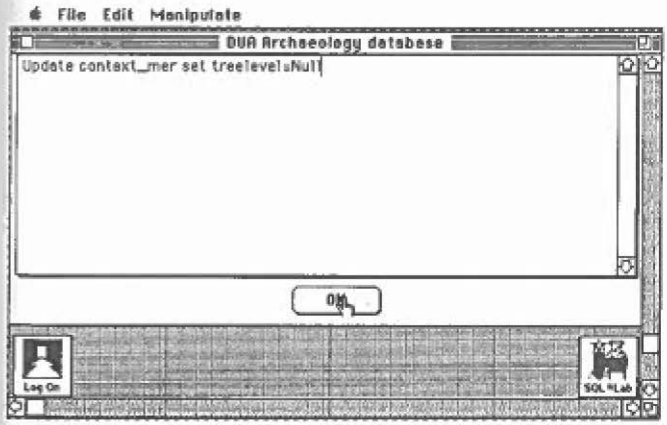


Fig. 6 — SuperCard Environment for Matrix manipulation.

For graphical output a Supercard button is created for each context. Buttons are complex entities within the SuperCard environment which allow linkages to other objects. However we use them simply as boxes which can be labeled with the context id (prefixed with the letter "c"). Starting with a nominal location for context 0 we traverse the tree in depth order. For each context with the current treelevel we extract information relating to overlays with criteria which satisfy test i) and ii) above. This is achieved by JOINing views of the context_mer table. The second view is simply a duplicate of context_mer. The resultant view is called Contexts and it is created by the command:

```
Create view (id,overlays,treelevel,ovlevel,nochildren) as
select unique C.id, C.overlays, C.treelevel, O.treelevel, nochildren
from context_mer C, context_mer O, children
where O.id = C.overlays and children.id = C.id;
```

The resultant table for the above data set is:

SQL > select * from contexts;

ID	*OVERLAYS*	*TREELEVEL*	*OVLEVEL*	*NOCHILDREN*
0	1	0	1	1
1	2	1	2	7
1	3	1	2	7
1	4	1	2	7
1	5	1	3	7
1	6	1	4	7
1	7	1	5	7
1	8	1	5	7
2	5	2	3	1
3	5	2	3	4
3	6	2	4	4
3	7	2	5	4
3	9	2	6	4
4	5	2	3	4
4	6	2	4	4
4	8	2	5	4
4	9	2	6	4
5	6	3	4	2
5	9	3	6	2
6	7	4	5	3
6	8	4	5	3
6	9	4	6	3
7	9	5	6	1
8	9	5	6	1

24 records selected.

Thus the SQL command to select valid linkages to be drawn reduces to:

```
SQL > select * from contexts where nochildren = 1 or ovlevel = treelevel + 1;
```

ID	*OVERLAYS*	*TREELEVEL*	*OVLEVEL*	*NOCHILDREN*
0	1	0	1	1
1	2	1	2	7
1	3	1	2	7
1	4	1	2	7
2	5	2	3	1
3	5	2	3	4
4	5	2	3	4
5	6	3	4	2
6	7	4	5	3
6	8	4	5	3
7	9	5	6	1
8	9	5	6	1

12 records selected.

To ensure an appropriate geometry for the tree a further view is utilised:

```
SQL > create view validlinks (id,nolinks) as
  select id, count (overlays) from context where ovlevel = treelevel + 1
  or nochildren = 1 group by id;

View created.

SQL > select contexts.id,overlays,treelevel,ovlevel,novalidlinks
  from contexts,validlinks
  where (ovlevel = treelevel + 1 or nochildren = 1)
  and validlinks.id = contexts.id
  order by treelevel,contexts.id,overlays;
```

ID	*OVERLAYS*	*TREELEVEL*	*OVLEVEL*	*NOVALIDLINKS*
0	1	0	1	1
1	2	1	2	3
1	3	1	2	3
1	4	1	2	3
2	5	2	3	1
3	5	2	3	1
4	5	2	3	1
5	6	3	4	1
6	7	4	5	2
6	8	4	5	2
7	9	5	6	1
8	9	5	6	1

12 records selected.

In order to enable the drafting of the resultant matrix substitution variables are employed, the final form of the output being shown in the Fig. 7.

Elementary drawing rules have been established which enable a relatively clear chart to be produced automatically. These rules obviously require further enhancement if they are to produce more than a provisional matrix. Provision must also be made for insertion of interpretive relationships and editing of the graphic representation. However, with the limitations outlined above, the

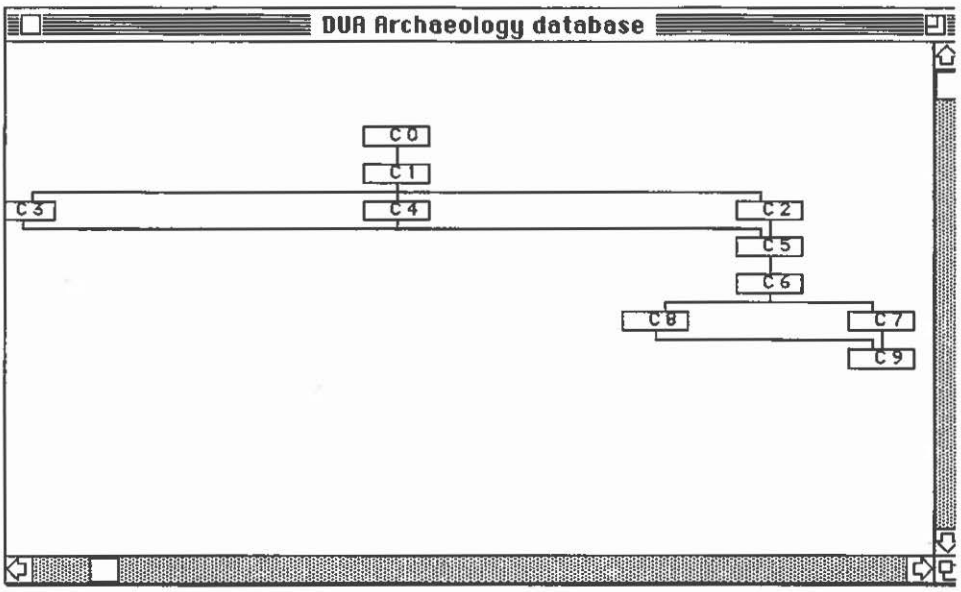


Fig. 7 — Adiacency matrix for sample dataset.

matrix does seem to provide a useful tool for site interpretation. Testing of the procedure against two sites of differing complexity demonstrates the effectiveness of the algorithms, especially in the second case where more than 500 contexts are modelled. The matrices resulting from this analysis are included as appendix A.

Conclusions

This article presents only the first stage of a larger research project and the program is missing many desirable features. It will be necessary to add the ability to edit the matrix, move single contexts, or groups of contexts, without losing the definitive relationships. Much has already been done to present errors in the data, and the next stage of development will focus on these matrix editing features.

However important these features are, it must be remembered that the fundamental importance of a computer generated stratigraphic matrix is not to duplicate present practice. The purpose of this matrix generation program is to provide a basic rigorous stratigraphic model with an interface which integrates this model with other forms of information.

Programs based, as this one is, on demonstrable adjacency and clear rules of relation provide a baseline for further interpretation of stratigraphic struc-

ture. Interpretive stratigraphic models based on such provisional matrices offer greater compatibility than interpretive models which develop from the primary record in that basic rules of redundancy may vary without documentation. Starting with a base model moves the debate from drawing conventions to issues of structural development and association.

By using a Hypertext interface the stratigraphic model has a further interpretive advantage. Context nodes on the matrix exist as objects which can be related to other Hypertext objects. This allows the interpretation of a matrix to be substantiated by reference to data, graphics, text or primary site records. The creation of the stratigraphic matrix in Supercard offers a presentation of the matrix, and subsequent interpretive matrices, that directly integrates the stratigraphic model with the information that supports or challenges that model.

Therefore, the purpose of this research is not simply to provide a quick and easy means of drawing a Harris Matrix, but is to provide a means by which stratigraphic models may be developed from basic principles and integrated with relevant information. Such an environment allows for the critical relation of data and interpretation, and the ability to deconstruct arguments back to a basic record. Though we feel that we have achieved many of these goals, it is clear that much is yet to be done. However, the major focus of this research remains the integration of archaeological information and its effect on modes of interpretation.

ACKNOWLEDGEMENTS

The system described has been implemented as part of a research collaboration between the Department of Urban Archaeology of the Museum of London (DUA) and the Department of Photogrammetry and Surveying at University College London (UCL P&S) under funding from the Autocarto/RICS education trust.

ROBIN BOAST
Department of Urban Archaeology
Museum of London

DAVE CHAPMAN
Department of Photogrammetry and Surveying
University College London

Appendix A. — Provisional adjacency matrix for Dominant House (1988) 520 contexts.

BIBLIOGRAPHY

BARKER P. 1982, *Techniques of Archaeology Excavation*, London, B. T. Batsford.

BISHOP S., WILCOCK J. 1976, *Archaeological Context Sorting by Computer: The Strata Program*, « Science and Archaeology », 17, 3-12.

GRACE R., ORTON C. 1989, *Hypercard as a teaching tool*, in *Computer Applications in Archaeology*, Oxford, British Archaeological Reports.

HARRINGTON J. L. 1988, *Relational Database Management for Microcomputers: Design and Implementation*, New York, Holt, Rinehart and Winston.

HARRIS E. 1989, *Principles of Archaeological Stratigraphy*, London, Academic Press.

ABSTRACT

As part of an on-going project for a London based Archaeological Information System, work on the automatic generation of stratigraphic adjacency matrices has been undertaken jointly between the Department of Urban Archaeology (MOL) and the Department of Photogrammetry and Survey (UCL). Such a program has been developed on an Apple Macintosh using SQL tree-walking techniques (under Oracle RDBMS) and a Hypertext interface which handles graphic presentation and manages rule based drawing conventions (using Supercard object-oriented Hypertext). The ultimate goal of this research is the creation of an interactive reporting structure which allows access to many levels of recorded and interpretive site information.