EDITORIAL

Databases in Molecular Graphics

A survey of the computer graphics literature shows the importance of a link between a graphics facility and a database. This issue of Journal of Molecular Graphics *includes three papers that cover uses of databases in chemical information retrieval and holding 3D molecular data. This editorial gives an introduction to this set of papers.*

DATA ENTITIES

Before considering the way that data are stored in a modelling system, one needs to consider the data entities that are required. This task, so often done quickly and inefficiently, is one of the most important steps in building a graphics system. Any data entity overlooked at this early stage may cause the system builder to adopt a system of insufficient power or flexibility. For example, someone building a protein crystallography system some years ago might not have realised that protein crystallographers would become as interested as they have in modelling protein-substrate interactions on their graphics systems. This could have meant that the data structure adopted did not have sufficient flexibility to handle small molecules with ring systems. In the early stages, the system builder has either to carry out an exhaustive data analysis, or to provide a system which has the flexibility to add new entities. The choice is not clear cut because, as we shall see later, flexibility carries an overhead in data access speed.

DIFFERENT DATA STRUCTURES

A molecular model used to build a display list has three basic requirements:

- atomic positions
- atom connectivity
- atom properties and identifiers

Atomic positions are generally held as coordinates measured with respect to orthogonal axes, in Ångström units. This system allows calculations such as interatomic distances to be carried out efficiently.

Connectivity describes which pairs of atoms in the molecule are bonded. This information can be held in several ways, or in the case of the Ring system, the information is not held at all, but instead is calculated on the fly from the interatomic distances. Three methods of holding connectivity data are described in this section. They are:

- atom pairs
- linked lists
- pointers

Atom pairs are lists of pairs of atoms that are bonded. The pairs may be given in one of several forms. For example, the alanine amino acid residue's connectivity may be specified as:

```
N   CA
CA  CB
CA  C
C   O
```

where the atoms in the first column are bonded to those in the second. If the atoms are numbered in order N–CA–CB–C–O, the connectivity may be specified as:

```
1  2
2  3
2  4
4  5
```

In a linked list, there is no specific connectivity file, this information being kept in the atom file. By default, each atom is connected to the following one. As sometimes there are branches and dead-ends in the list, a column is included in the table to signify whether there is a connection to the next atom or if this is a branch end. In this system the alanine atom file may look like:

```
N     1.563    36.535   25.819   B
CA    0.681    36.693   26.994   B  B
CB    1.530    37.146   28.168   N
C    -0.468    37.645   26.701   B
O    -0.427    38.400   25.707   N
```

The decimal numbers are the atomic coordinates. B denotes that there is a bond between the atom and the following atom. N denotes that there is no bond. CA has two characters following the coordinates. This is because there is a fork here. A program turning this into a display would move on until it came to an N that signifies that two atoms are not to be bonded. Then it would go back looking for an entry in the second column. This would then connect the atom with the B in the second column to the atom following the atom with the N. Systems like this have been used for macromolecule specific systems although they have invariably had 'fudges' built in to overcome the problems of closing rings. Sometimes this has required the use of dummy atoms so that a particular position may be repeated in the linked list. When non-protein structures are entered into such a system the problems are considerable. The system is really only valid for a straightforward hierarchy and not a network.

Pointers can be used to indicate which atoms are bonded. The pointers specify the atom either in terms of its position in the file or its position in the remainder of the following file. With a pointer system, the alanine

 0263-7855/84/030066-04 $03.00

file could look like:

1	N	1.563	36.535	25.819	2	
2	CA	0.681	36.693	26.994	3	4
3	CB	1.530	37.146	28.168	0	
4	C	−0.468	37.645	26.701	5	
5	O	−0.427	38.400	25.707	0	

A pointer with the value zero indicates that there is no further connection.

Molecules can be considered to be made up of standard groups of atoms. For example, proteins are made up of the amino acids and DNA is made up of nucleotides. More generally, organic structures can be considered to have components such as methane, ethane, cyclohexane etc. In order to save storage, the connectivity for these standard groups can be kept in a dictionary. Thus, in a protein, there would only be one place where the connectivity of our alanine is stored. Everytime an alanine is encountered in the atom position file, the dictionary would be consulted to establish the connectivity. A complication with this type of system is that there still has to be some kind of mechanism to establish the connectivity between the standard groups. In proteins this is not difficult since the connection is always between the carbonyl of the first amino acid and the nitrogen of the second. However, even in this easy case, there is the complication of disulphide bridges and hydrogen bonds. Finally, there is also the problem that many proteins contain non-protein atoms. This may require that each protein is considered as a departure from the ideal case. Therefore, it should be easy for the user to add groups to the dictionary and make changes to the inter-group connectivity tables.

Individual atoms have to be referenced in order to build the display list. There are many other instances when an atom in the molecule would need to be identified. So there is a requirement for atom identifiers such as integers or the standard labels given in the alanine example. It is important that the identifiers are unique within the molecule and for that reason two or more identifiers are often used or the identifier is made up from two or more components. For example, in the protein case, the system usually needs to know:

- the atom identifier
- the amino acid type
- the amino acid sequence number
- the chain identifier

So part of an atom position file may now look like Table 1.

This fragment is in the first chain. Each atom can be uniquely identified without having individual atom numbers, although some researchers prefer to have these as well.

An alternative way of storing these data is to use a data structure from a high-level language. For example, these protein data could be kept in a PL/1 structure, declared as:

```
1 ATOMS,
  2 CHAIN NUM CHAR(5),
    3 RES NUM CHAR(5),
    3 RESTYPE CHAR(3),
      4 ATOM TYPE CHAR(3),
      4 XYZ(3) CHAR(8),
```

Note that the data types are all given as characters. This allows the user to edit the atom positions file using a general-purpose editor. However, for efficient storage and usage, it is better if numerical values are stored so that they can be used for calculations without conversion.

Table 1. Part of an atom position file

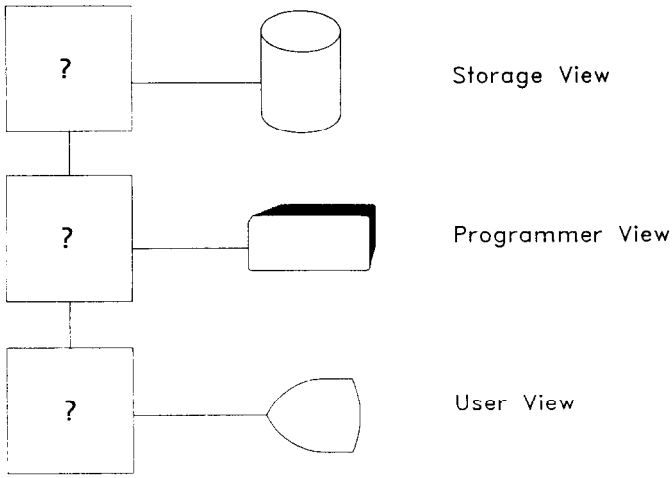
Chain No.	Residue type	Residue No.	Atom type	X	Y	Z
1	ALA	28	N	8.828	2.070	−10.409
1	ALA	28	CA	17.630	2.038	−9.559
1	ALA	28	C	16.596	1.226	−10.338
1	ALA	28	O	16.023	0.259	−9.817
1	ALA	28	CB	17.029	3.442	−9.384
1	LEU	29	N	16.372	1.631	−11.574
1	LEU	29	CA	15.399	0.941	−12.430
1	LEU	29	C	15.550	−0.563	−12.664
1	LEU	29	O	14.558	−1.304	−12.704
1	LEU	29	CB	15.386	1.569	−13.837
1	LEU	29	CG	14.957	3.015	−13.964
1	LEU	29	CD1	15.388	3.552	−15.308
1	LEU	29	CD2	13.460	3.140	−13.816
1	GLU	30	N	16.789	−0.990	−12.817
1	GLU	30	CA	17.075	−2.411	−13.054
1	GLU	30	C	16.611	−3.138	−11.791
1	GLU	30	O	16.011	−4.220	−11.863
1	GLU	30	CB	18.587	−2.663	−13.311
1	GLU	30	CG	18.989	−4.089	−13.739
1	GLU	30	CD	20.436	−4.217	−14.347

The data structure shown is a hierarchy which stores each value the necessary number of times. Instead of repeating the chain number for each atom in the chain, it is stored once at the beginning of the chain. Similarly, the residue number and residue type are stored once for each residue.

The choice of atom properties to be included in the database is largely dependent on user application. Changes in user interests have meant that there has to be some mechanism for adding new data entities. This is often achieved using an independent data file which is opened by the application procedures requiring the data. For example, a complicated energy calculation that requires individual atomic charges would use a data file holding these data which is external to the system.

DATABASE MANAGEMENT

Another way of storing data is to use a database system. A database takes away the need for the programmer to understand how the data are physically stored. Instead, the programmer has a conceptual view of the data which simplifies data access. Figure 1 shows the three views of data present when a database system is used. First there is the 'storage view' which is a secret to all except the system builders. This view details how the data are packed onto the storage medium. Second, the 'programmer view' is a conceptual view designed to permit easy access. Finally, there is the 'user view' which is the graphics display of the molecular properties. A database system can also include a high-level data access system that allows the programmer to carry out powerful operations on the data without writing complicated programs. There are two types of database commonly available today. They are hierarchical and relational.



The hierarchical database approach developed from the use of data structures. A group using the hierarchical approach is UCSF, USA. The database for the MIDAS system at UCSF was designed and implemented by Tom Ferrin and Conrad Huang[1]. The data are kept on direct access storage devices and are linked by systems of pointers. The programmers have no concept of how the data are stored and use a data access language to obtain the item of data they want. It is important to appreciate that in this kind of database items of data have no individual meaning unless they are qualified by their hierarchical pathways. For example, if a programmer used the access statement:

Get N in Alanine

insufficient specifications would have been given uniquely to identify one atom. Even if a residue number was given this would not be enough as the residues may only be unique within each chain. So the statement would have to be:

Get N in Residue Number 10 in Chain Number 2

This appears complicated but it is still very simple compared with having to write an access program to extract such instances. Adding and deleting data can create complicated conceptual as well as physical problems for the programmer. These problems are usually overcome by making restrictions on the programmer.

Hierarchical databases are very good for modelling hierarchical systems but in other instances they create problems. Namely, how can something that is not hierarchical be made to appear so. This problem is normally resolved using dummy atoms or specifying particular atoms more than once. A more general approach is the relational system. The essence of the relational approach is that the data are considered to be stored in tables called relations. These tables obey certain rules:

- Rows must be unique within the table.
- Columns must be unique within the table.
- Columns and rows may appear in any order.

The nomenclature for relational databases may be described simply (see Table 2).

A relational system for storing molecules may appear as Table 3. It includes a relation holding the atom data and a relation holding the connectivity. The title of the relation is shown in upper case. Below the title are the column headings. These examples have been taken from the relational database in use at UKSC[2].

The atom relation describes the atoms and their coordinates. In order to rid the system of the need for standard naming conventions the atom id is separate to the atom type. As we use the Brookhaven atom id in this example, this information is repeated. However, the user working on small molecules may prefer to use a set of numbers for identifiers. If more atom data are required, they can be added to this relation as extra domains. The connectivity describes the connections between standard chemical groups that are held in the system dictionary. A group identifier relation describes the group type of each Group Id. The group dictionary is the relation that describes the intra-group connectivities. The alanine entry in the group dictionary may look like Table 4.

The power of the relational database approach is in the operations that can be carried out on the data. The UKSC system* uses a system called relational algebra,

Table 2. Nomenclature for relational databases

Relation jargon		Simple		Programmer
Relation	=	Table	=	File
Tuple	=	Row	=	Record
Domaain	=	Column	=	Field

Table 3. Example of relational system for storing molecules

ATOM RELATION

Residue No.	Atom ID	X	Y	Z	Atom type
1001	N	8.828	2.070	−10.409	N
1001	CA	17.630	2.038	−9.559	C
1001	C	16.596	1.226	−10.338	C
1001	O	16.023	0.259	−9.817	O
1001	CB	17.029	3.442	−9.384	C
1012	N	16.372	1.631	−11.574	N
1002	CA	15.399	0.941	−12.430	C
1002	C	15.550	−0.563	−12.664	C

GROUP CONNECTIONS

Group Id1	Group Id2	Connection type
1001	1002	Peptide
1002	1003	Peptide
1003	1004	Peptide
1003	1012	Disulphide

Table 4. Example of alanine entry in the group dictionary

GROUP DICTIONARY

Group Id	Atom Type1	Atom Type2
ALA	N	CA
ALA	CA	CB
ALA	CA	C
ALA	C	O

* Readers familiar with relational databases will notice that these relations are not in third normal form, this means that there is some redundancy, and entities such as the GROUP ID (ALA) are repeated. This is a trade-off against efficiency in this particular implementation.

which was first defined by Ted Codd[3]. The result of each operation is a new relation. The user can request atoms which have particular identifiers or which appear in certain groups. Interatomic distances can be computed and atom pairs selected on distance criteria. In this way it is possible to extract from the database all ion pairs within 4.0 Ångströms, using a few relational algebra statements. This power allows a great deal of research to be carried out without the need for writing, debugging, and compiling programs. However, the power of this system carries an overhead. It is much quicker to extract the alpha carbon atoms from a protein using a hierarchical system with a predefined pathway than it is to carry out the equivalent operation using a relational database. Here there is a clear choice which depends upon the use of the system. A system used for particular tasks, say fitting models to electron density maps, will be better served by a hierarchical database. A system which is used as a general research tool, and where the future applications are unknown, will benefit greatly from the more flexible relational approach.

A different, but similarly powerful approach, has been adopted by the University of Alberta in the M3 system[4]. This database system, called the D3 Information System, is based upon a directed graph model. Data are held in ordered tables which are accessed via labelled edges. This structure is capable of providing the same kind of power as the relational database, but the D3 implementation has not included a high-level interface, as in the UKSC system. Instead it has concentrated on a small set of useful operations for the application programmers. In particular, the author stresses the value of D3 for sorting, searching, and mapping problems, which are common in molecular graphics.

2D DATABASES

The information officers of the chemical and pharmaceutical industries keep data concerning the large number of compounds synthesized and the results of experiments using these compounds. These data are used in future research to prevent duplication of experiments and to give starting points for designing new compounds. Normally, no 3D structure is available for the compounds, although an energy minimized, or *ab initio* conformation may be. Therefore the nature of these molecular data are the atom types and connectivity. The chemical formula provides these data. Researchers have shown the value of computer graphics for entering, retrieving and updating the database. Such industrial databases contain many thousands of compounds and the mechanics of operating an efficient retrieval system with a large database requires some elegant programming. The problem is further complicated by the need for text information to be stored in conjunction with the formulae. This is often stored in a parallel system, using a general-purpose commercial product.

The user of the compound database normally constructs a 2D formula diagram interactively on the graphics device. This is achieved using a lightpen or tablet and pen. Having made the diagram, the user can ask for this structure to be added or retrieved from the database. The user can further specify this as a molecular fragment and look for compounds that contain this fragment. This substructure search facility is now an important tool in chemistry research.

FUTURE TRENDS

In recent years there has been an increase in interest in the use of databases for storing 3D coordinates. The benefits of the systems so far installed are becoming apparent and other users wish to have access to the increase in efficiency or power possible. In the next couple of years, development of molecular graphics software will probably lead most groups towards database management systems. Another point to be mentioned in passing is the development of expert systems or more generally, the use of artificial intelligence, in molecular modelling. Such systems require a knowledge base which goes beyond the simple data representation. The work presently being carried out at UCSF into this exciting aspect of this research may give the molecular graphics community some insight into the future requirements of molecular databases.

THE FOLLOWING PAPERS

The first three papers in this issue cover different aspects of molecular databases. First, there is Elder, Machin and Hull's paper which looks at a new retrieval method from the Cambridge Crystallographic Database. This paper will be of great interest to researchers using the Cambridge Database for examining small molecule structures both in pure and applied research. Second is a paper from Japan by Iga and Yakumota which discusses their implementation of a system for retrieving protein structures from the Brookhaven Protein Data Bank. The third paper is from Anderson from Molecular Design Ltd. This paper explains how a 2D database can be used for applied research, concentrating on the user interface.

In the next year I hope that we shall get some more papers on the use of databases with molecular graphics systems. The *Journal of Molecular Graphics* provides the ideal forum for the exchange of this kind of technical information.

Finally, I have to announce that this is the last issue of *Journal of Molecular Graphics* that I shall edit. The new editor is Dr W Graham Richards, Physical Chemistry Department, Oxford University, UK. I would like to thank all concerned for the support I have received over the last two years. All that remains is for me to wish Graham and the *Journal of Molecular Graphics* all the best for the future.

Andy Morffew

REFERENCES

1 **Ferrin, T E** University of California, San Francisco, CA, USA, personal communication

2 **Morffew, A J, Todd, S J P, Snelgrove, M J** *Comput. Chem.* (1983) Vol 7 No 1 pp 9–16

3 **Codd, E F** in *Data base systems* Courant computer science symposia series, NJ, USA, Vol 6

4 **Broughton, C G** *D3 Information Management System* University of Alberta, Canada (1982)