A two-path recursive relational database structure for molecular information systems

J. Bradley

Computer Science Department, University of Calgary, Calgary, Alberta, Canada

A recursive relational database structure, called the two-path structure, is proposed for holding data about the properties and structure of molecules. The two-path database approach solves certain severe restriction problems by providing two pathways to the structure of complex chemical entities. One pathway permits a fast descent to the atomic structure in terms of correct IUPAC carbon atom occurrence numbers. The other pathway descends through the molecular substructures, which can be referenced at any level in retrievals. The database appears to be capable of faithfully recording the structure of all chemical entities, including proteins and enzymes.

Keywords: chemical bond, enzyme, IUPAC, molecular structure, molecular information systems, protein, relational database, recursive relationship, two-path database

INTRODUCTION

Database techniques have been applied to specific chemical structure data in the past, mainly for use in generating graphics displays.[1–6] Although there have been significant reports in the literature[4,5] of the use of relational databases[7,8] with SQL for molecular information systems, there appear to be no proposals for structures for a relational database cabable of holding the structure of every conceivable compound, consistent with IUPAC naming conventions. Such a relational database structure is described in this paper. We have called it the two-path database structure. Two-path relational databases for molecular data can be used with SQL, the standard relational language for such common relational database systems as DATABASE2,[9] INGRES[10] and ORACLE.[11]

Two-path databases provide two pathways to the structure of an entity in cases where the entity is derived from one or more substructures. The two pathways are necessary, partly for reasons of disk space conservation, but mainly because of a need for conformance with atomic occurrence number standards of the International Union of Pure and Applied Chemistry (IUPAC).

BASIC SINGLE-LEVEL APPROACH

The basic single-level approach to the problem of relational molecular structure databases involves the use of a tuple (or record) to describe each bond in a molecule. Since the database is single-level, the bond is always between two atoms, and never between two substructures.

In Figure 1a there is an example of a single-level database where each bond between atoms is described by a tuple. The database contains information about the compounds propene and methoxyethane.

The relation CHEMICAL-COMPOUND has a tuple for each chemical compound. A tuple can have many additional fields, giving information about the physical and chemical properties of a complete chemical compound. We show only two such fields, namely boiling and melting points, as examples. Since the database is single level, substructures may not appear in CHEMICAL–COMPOUND.

In contrast, the relation CHEMICAL–BOND has a tuple for each bond in a compound. There is thus a one-to-many (1:n) relationship[12] between CHEMICAL–COMPOUND and CHEMICAL–BOND, since a chemical compound will have many bonds, but a specific bond occurs only in one entity.

The fields EL1 and ER1 give the entities (atoms in this case) at each end of a bond (left end (L), right end (R), where right and left serve only to distinguish ends of a bond, and not the geometry of a molecule). Carbon atoms are traditionally on the left.

The field BOND has the arbitrary value 1 for a single bond, 2 for a double bond, 3 for a triple bond, 1.5 for a resonance bond, and 0.1 for a hydrogen bond. (As we shall see, there is also a need for zero and negative BOND values, to denote imaginary and broken bonds.) To denote an atom, the standard chemical symbols are used (C for carbon, H for hydrogen, and so on).

EL1# and ER1# give the occurrence numbers of atoms of the same type. The first occurrence of a carbon atom, for example, has occurrence number 1, the next occurrence number 2, and so on. The occurrence number together with

Address reprint requests to Dr. Bradley at the Computer Science Department, University of Calgary, Calgary, Alberta, Canada.
Received 24 May 1989; accepted 10 October 1989

IUPACNAME	CODE	MP	BP
PROPENE	PRN	-185	-048
METHOXYETHANE	MXH	-139	-023

CHEMICAL-COMPOUND

CODE	EL1#	EL1	ER1#	ER1	BOND
PRN	1	C	2	C	2
PRN	1	C	1	H	1
PRN	1	C	2	H	1
PRN	2	C	3	H	1
PRN	2	C	3	C	1
PRN	3	C	4	H	1
PRN	3	C	5	H	1
PRN	3	C	6	H	1
MXH	1	C	1	O	1
MXH	2	C	1	O	1
MXH	1	C	1	H	1
MXH	1	C	2	H	1
MXH	1	C	3	H	1
MXH	2	C	4	H	1
MXH	2	C	5	H	1
MXH	2	C	6	H	1

CHEMICAL-BOND

Figure 1a. Single-level database

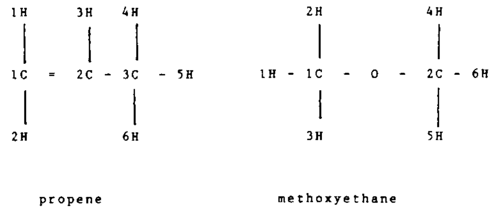


Figure 1b. Structures of propane and methoxyethane

the type of atom (EL1 or ER1 value) uniquely identifies an atom. For example, the double bond in propene has the structure

1C = 2C

As a result, the structure of propene and methoxyethane are as shown in Figure 1b, using the data from the database in Figure 1a.

The occurrence numbers to the left of the atomic symbols in Figure 1b would likely be omitted in any display generated by a graphics system. The occurrence numbers are crucial for the storage of the structure of any molecule, no matter how large or complex. They are also used by IUPAC in chemical nomenclature. The two-path approach, which we present shortly, is to a considerable extent due to a need for synchronization of occurrence numbers with both database and chemical methodology.

Limitations of the simple single-level approach

The simple single-level approach above could, in theory, hold the structure of every conceivable molecule, but is limited in two important ways. First, the same chemical group, for example a benzene or pyrimidine ring, occurs in many compounds, possibly many times within the same compound. In the database in Figure 1a the atomic structure for benzene would have to be replicated for every molecule containing a benzene ring derivitive. Thus repetition of the atomic structure of such groups, for every compound in which they occur, is an unacceptable waste of storage space. It has to be remembered that some 2 million compounds have been described in the literature.

The other limitation is that we cannot use SQL to express queries involving these common subgroups, such as benzene rings. Thus we could not pose queries such as:

> What compounds have two distinct benzene rings?
> What compounds have exactly have one purine and one pyrimidine group?

and so on. Nevertheless, with the database in Figure 1a, we can apply SQL to the molecular structure at the atomic level, even if we cannot deal with chemical groupings such as benzene rings. For example, the request

> List the unsaturated (have one or more double carbon-carbon bonds) compounds

can be easily expressed in SQL as

```
SELECT IUPACNAME FROM CHEMICAL
WHERE CODE IN (SELECT CODE FROM
              CHEMICAL–BOND
              WHERE EL1 = 'C' AND ER1 = 'C'
                AND BOND = 2);
```

The database in Figure 1a is probably the best we can do using a single-level approach. If we want to be able to deal with substructures, however, we need a multiple-level recursive approach.

Problems with multiple-level relational databases

Recursive bill-of-materials database structures[12,13] are commonly used for databases describing objects made up of substructures, which in turn are made up of subsubstructures, and so on. Such databases are commonly used in the assembly and maintenance of complex objects in manufacturing plants. They are thus multiple-level databases. They are recursive or cyclic databases because a relation in the database participates in a many-to-many relationship with itself,[12,13] thus giving rise to multiple levels corresponding to the levels of substructure, subsubstructure, and so on, in the entities described by the database. In theory, there is no limit to the number of levels of nesting of substructures that can be employed in a bill-of-materials database.

In order to eliminate redundancy due to the same basic chemical groupings appearing in many compounds, and to allow SQL expressions to reference basic structures (methyl groups, benzene rings, purines, amino acid residues, and so on) a recursive database structure is needed for molecular structure data. However, we cannot use the standard bill-of-materials database structure. The problem is that in addition to subgroupings within groupings, chemical bonding between the atoms of the groupings and subgroupings must be specified at all levels. It is the need for provision for this bonding at all levels that gives rise to a rather special database structure for molecular structure data.

At first sight it may appear that an obvious modification of the bill-of-materials database structure will serve, where each atom in a bond is always uniquely specified in terms of the structures it is within. For example, in an expanded tuple of CHEMICAL–BOND in Figure 1a to handle molecules with three levels of structure, instead of 3 C we might use 2 XYZ 5 PQR 3 C, meaning carbon occurrence 3 within PQR substructure occurrence 5 within XYZ substructure occurrence 2. However, this modification will not work, because of space wastage and IUPAC carbon atom occurrence number problems, as the following discussion shows.

Suppose we attempt the multiple levels of a normal bill-of-materials database, with the additional fields to allow for the specification of each chemical bond. We have the following disadvantages.

(1) The number of fields in the CHEMICAL–BOND relation increases in proportion to the number of levels needed for the most complex compound, so that a great deal of space is wasted in the CHEMICAL–BOND relation. The number of fields increases because each tuple must describe a chemical bond, and therefore must identify two atoms at either end of the bond. Thus if we need to refer to atom 2 XYZ 3 PQR 3 C, as mentioned previously, we need to add 2 × 4 or eight additional fields to CHEMICAL–BOND in Figure 1a. But with large numbers of simpler molecules the additional fields would have null values and so a very great deal of disk space would be wasted.

(2) The occurrence numbers used with carbon atoms, in particular, will in many cases not match standard IUPAC chemical nomenclature. This should be obvious. For example, suppose we use the numbering system 1–6 for the carbon atoms of a benzene ring, and store the structure of benzene in CHEMICAL–BOND. Then, for example, if we attempt to store 2,3,6,7-tetrachlorodibenzo-*p*-dioxin (the toxin dioxin) as being based on two benzene rings, the correct IUPAC carbon atom occurrence numbers cannot be extracted from the database.

And we have the following advantage.

(1) Compounds can always be referenced, in searches of the database, in terms of common substructures, no matter what level of recursion is involved.

We might try to get around the wasted space problem by restricting the database to two levels, using the CHEMICAL–BOND relation only for structure data. If so, we have the following disadvantage.

(1) We cannot reference many chemical compounds, in searches of the database, in terms of common substructure in cases where the substructure is more than one level down from the original compound.

And we have the following advantages.

(1) The number of fields in the CHEMICAL–BOND relation does not depend on the complexity of the molecules, and there is little waste of space.

(2) The numbering (occurrence number) system used for carbon atoms can match IUPAC nomenclature.

The two-path approach involves a subtle database structure that has all of the advantages of the two approaches above and none of the disadvantages. The two-path structure allows the user, in many cases, to choose between two distinct pathways to be followed in extracting information about the structure of any chemical entity.

THE TWO-PATH APPROACH

Before presenting the two-path approach in detail, the concept of bond cleavage tuples has to be introduced, since it is crucial to the two-path solution.

Use of bond-cleavage tuples

It is clearly useful, from both an information retrieval and storage space utilization viewpoint, to use common groupings of atoms, such as the methyl or hydroxyl grouping, in both the CHEMICAL–COMPOUND (to be called CHEMICAL–ENTITY henceforth) and extended CHEMICAL–BOND relations. However, a very great refinement of this technique is both possible and desirable, when it is considered that many different groupings are formed by removal of one or more atoms from some chemical compound.

The compound benzene is a common example of this. Benzene is a 6-carbon ring, with resonance bonds between the carbon atoms and with each of the 6 carbons bonded to a hydrogen atom. Many chemical compounds are formed by removing a single hydrogen atom and replacing it with something else. If we replace it with chlorine, we get chlorobenzene, with a hydroxyl group, we get phenol (or hydroxybenzene), with an amine group we get aniline, and so on.

Thus we could regard a benzene molecule minus a hydrogen atom as a grouping, and store its structure in the database. However, that is not a good idea, because we can also form many compounds by replacing two hydrogens from benzene by other atoms or groupings. Replacing the 1-carbon hydrogen by chlorine, and the 3-carbon hydrogen by bromine, for example, gives 1-chloro-3-bromobenzene; if instead we use hydroxyl groups as replacements, we get 1,3-dihydroxybenzene, and so on. Thus a benzene molecule minus a specific two hydrogen atoms could also be regarded as a grouping and its structure could also be stored in the database. We can continue until we are left with the bare 6-carbon ring with no hydrogens, which is the final grouping.

Clearly, it makes more sense to store only the structure of benzene, and structure a compound based on benzene has having had one or more carbon–hydrogen bonds broken, and replaced by new bonds. Thus phenol would consist of benzene less a carbon–hydrogen bond and plus a carbon-hydroxyl bond. The cleavage of a bond in a compound to form a new compound would be noted in the database in the BOND field by giving it the value −1 (or −2 in the rare case of cleavage of a double bond). In the case of a double bond −1 would mean cleavage of one of the two bonds). This concept is illustrated in the two-path database approach.

The two-path database structure

The two-path database structure is best understood by studying an example, such as that in Figure 2. The structure in Figure 2 has to be thought about carefully. It has some quite subtle recursion features.

With complex molecules there are always two pathways to the atomic structure. One pathway allows fast descent to the IUPAC structure. The other pathway can involve many recursion iterations in descending to the atomic level, but will extract the derivative substructures that occur in the molecule at each iteration level. The structure diagram, showing the relationships between the relations for the database, is in Figure 3.

The relation CHEMICAL–ENTITY has a tuple for each chemical entity, whether molecule or chemical group or

IUPACNAME	M-CODE	A-CODE	MP	BP	TYPE
BENZENE	-	BNE	-	-	COMPOUND
HYDROXYBENZENE	HDB	-	-	-	COMPOUND
2,3,6,7-DICHLORODIBENZO-p-DIOXIN	DOX	-	-	-	COMPOUND
DIBENZO-p-DIOXIN	DBZ	DBZ	-	-	COMPOUND
HYDROXY	-	OH	-	-	GROUP

CHEMICAL-ENTITY

CODE	EL2#	EL2	EL1#	EL1	ER2#	ER2	ER1#	ER1	BOND
HDB	1	BNE	1	C	1	BNE	1	H	-1
HDB	1	BNE	1	C	1	OH	1	O	1
DOX	1	DBZ	2	C	1	DBZ	2	H	-1
DOX	1	DBZ	2	C	-	-	1	Cl	1
DOX	1	DBZ	3	C	1	DBZ	3	H	-1
DOX	1	DBZ	3	C	-	-	2	Cl	1
DOX	1	DBZ	6	C	1	DBZ	6	H	-1
DOX	1	DBZ	6	C	-	-	3	Cl	1
DOX	1	DBZ	7	C	1	DBZ	7	H	-1
DOX	1	DBZ	7	C	-	-	4	Cl	1
DBZ	1	BNE	2	C	1	BNE	2	H	-1
DBZ	1	BNE	3	C	1	BNE	3	H	-1
DBZ	1	BNE	2	C	-	-	1	O	1
DBZ	1	BNE	3	C	-	-	2	O	1
DBZ	2	BNE	2	C	2	BNE	2	H	-1
DBZ	2	BNE	3	C	2	BNE	3	H	-1
DBZ	2	BNE	2	C	-	-	1	O	1
DBZ	2	BNE	3	C	-	-	2	O	1

MOLECULAR-BOND

CODE	EL1#	EL1	ER1#	ER1	BOND
BNE	1	C	2	C	1.5
BNE	2	C	3	C	1.5
BNE	3	C	4	C	1.5
BNE	4	C	5	C	1.5
BNE	5	C	6	C	1.5
BNE	6	C	1	C	1.5
BNE	1	C	1	H	1
BNE	2	C	2	H	1
BNE	3	C	3	H	1
BNE	4	C	4	H	1
BNE	5	C	5	H	1
BNE	6	C	6	H	1
DBZ	1	C	2	C	1.5
DBZ	2	C	3	C	1.5
DBZ	3	C	4	C	1.5
DBZ	4	C	11	C	1.5
DBZ	11	C	12	C	1.5
DBZ	12	C	1	C	1.5
DBZ	1	C	1	H	1
DBZ	2	C	2	H	1
DBZ	3	C	3	H	1
DBZ	4	C	4	H	1
DBZ	11	C	2	O	1
DBZ	12	C	1	O	1
DBZ	5	C	6	C	1.5
DBZ	6	C	7	C	1.5
DBZ	7	C	8	C	1.5
DBZ	8	C	9	C	1.5
DBZ	9	C	10	C	1.5
DBZ	10	C	5	C	1.5
DBZ	5	C	5	H	1
DBZ	6	C	6	H	1
DBZ	7	C	7	H	1
DBZ	8	C	8	H	1
DBZ	9	C	1	O	1
DBZ	10	C	2	O	1
OH	1	O	1	H	1

ATOMIC-BOND

Figure 2. Example of the two-path database structure

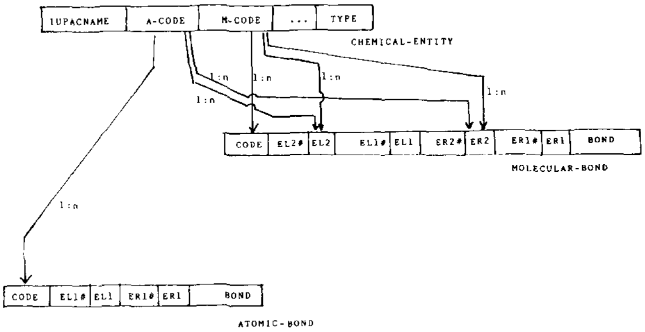


Figure 3. The general structure of a two-path database for chemical entities

substructure, and can contain information about the physical/chemical properties of the entity, such as boiling and melting points, where relevant. The fields M-CODE and A-CODE refer to codes used for the chemical entity in the CODE fields of the relations MOLECULAR–BOND and ATOMIC–BOND respectively (and also the EL2 and ER2 fields of MOLECULAR–BOND).

A tuple in CHEMICAL–ENTITY may have either an M-CODE value referring to tuples in MOLECULAR–BOND, or an A-CODE value referring to tuples in ATOMIC–STRUCTURE, or both M-CODE and A-CODE values referring to tuples in both MOLECULAR–BOND and ATOMIC–BOND. There is thus a 1 : *n* relationship between CHEMICAL–ENTITY and ATOMIC–BOND, and five distinct 1 : *n* relationships between CHEMICAL–ENTITY and MOLECULAR–BOND. The fact that CHEMICAL–ENTITY participates in several one-to-many (1 : *n*) relationships with MOLECULAR–BOND means that CHEMICAL–ENTITY participates in a many-to-many relationship with itself, and thus in a recursive many-to-many relationship.[8,12,13]

A group of tuples in ATOMIC–BOND with the same CODE field value gives the structure, in terms of atoms only and not higher level substructures, of the chemical entity whose code is CHEMICAL–ENTITY.A-CODE matches that CODE value. Each tuple describes a chemical bond.

A group of tuples in MOLECULAR–BOND with the same CODE field value gives the structure, in terms of both atoms and substructures, of the chemical entity whose code is CHEMICAL–ENTITY.M-CODE matches that CODE value. Each tuple describes a chemical bond.

Extraction of chemical structure information from a two-path database

To use the database to extract structure information about any common chemical entity that does not have substructures, or which is not itself a derivitive of some other entity, only the relations CHEMICAL–ENTITY and ATOMIC–BOND need be used. For example, in Figure 2, if we want to find out about benzene, we need only access the record for benzene in CHEMICAL–ENTITY, and the related records (code BNE) in ATOMIC–BOND. Thus the structure data is extracted in a single iteration of the database. For example, if we wanted to know how many carbon–hydrogen bonds there are in benzene, we would execute:

```
SELECT COUNT (*) FROM ATOMIC–BOND
WHERE EL1 = 'C' AND ER1 = 'H' AND
    CODE IN (SELECT A-CODE FROM
                CHEMICAL–ENTITY WHERE
                IUPACNAME = 'BENZENE')
```

We can tell from the CHEMICAL–ENTITY tuple for benzene that it has no substructure made up of other common chemical entities, since there is no value in the M-CODE field, and a value only in the A-CODE field. If we want a list of the names of the simple molecules in the database, we simply execute:

```
SELECT IUPACNAME FROM CHEMICAL–ENTITY
WHERE M-CODE IS NULL.
```

[Note that some relational database systems do not support NULL values; in that case any convenient code can be used to represent NULL, such as 0 for numeric fields and blank for alphanumeric fields.]

In contrast, if we need the structure of the benzene derivitive hydroxybenzene (or phenol), the tuple in CHEMICAL–ENTITY for that compound has an M-CODE value, and no A-CODE value. This tells us that the structure is given in MOLECULAR–BOND and is derived from one or more substructures. If we use this M-CODE value (HDB) to access MOLECULAR–BOND we find that the matching tuples indicate that this compound is formed by cleaving a hydrogen bond in benzene and replacing the hydrogen atom with the hydroxy group (–OH). The structure of benzene (and the hydroxy group, if required), and the relevant carbon numbers used in MOLECULAR–BOND can be found by referring to ATOMIC–BOND.

These two cases are simple and obvious. The cases of dioxin (2,3,6,7-tetrachlorodibenzo-*p*-dioxin) and dibenzo-*p*-dioxin illustrate how the database handles complex molecules. Dioxin is derived from dibenzo-*p*-dioxin by chlorination. But many other substances are derived from dibenzo-*p*-dioxin. At the same time dibenzo-*p*-dioxin is derived from two benzene molecules.

If we look up dibenzo-*p*-dioxin in CHEMICAL–ENTITY, we see that it has field values in both M-CODE and A-CODE. This means that we can get at its structure using either of two pathways.

(1) *A-path, or direct atomic pathway*. We can use the A-CODE value (''DBZ'') and ATOMIC–BOND to get the structure in terms of atoms only (not in terms of substructures) with the IUPAC numbering code in the carbon occurrence number fields (EL1# and ER1#). Thus the ATOMIC–BOND tuples with CODE value ''DBZ'' give each atomic bond in dibenzo-*p*-dioxin. However, if we use just CHEMICAL–ENTITY and ATOMIC–BOND we could not ask for information about substructures, for example, the number of benzene rings that dibenzo-*p*-dioxin contained.

(2) *M-path, or recursive molecular substructure pathway*. To get the structure in terms of substructures we need to use the M-CODE value ''DBZ'' and reference the tuples in MOLECULAR–BOND with code ''DBZ''. These tuples give the structure in terms of benzene, whose atomic structure is found in ATOMIC–BOND. The result of using MOLECULAR–BOND tuples with lower level references to ATOMIC–BOND tuples to get the structure of dibenzo-*p*-dioxin will be correct both in terms of substructures and atoms, but will NOT give the correct IUPAC number scheme.

It is for this reason that the database structure is called a two-path structure.

However the dibenzo-*p*-dioxin structure generated by the molecular substructure pathway (M-path), in terms of the benzene substructure, cannot be used for obtaining the structure of substances derived from dibenzo-*p*-dioxin, such as dioxin, since the the carbon-atom numbering scheme generated will not be the IUPAC one. Nevertheless, this method of getting at the substructure will enable queries about substructure to be handled. For example, suppose we want the number of benzene rings in dibenzo-*p*-dioxin. We would execute:

```
SELECT MAX (EL2#) FROM
MOLECULAR–BOND
WHERE EL2 = 'BNE' AND
    CODE IN (SELECT M-CODE FROM
    CHEMICAL–ENTITY
        WHERE IUPACNAME = 'DIBENZO-p-
        DIOXIN')
```

[At first glance it might be thought that the maximum value could occur in the field ER2#, and not EL2#. This is not so, for the highest occurrence number of any substructure is guaranteed to occur in EL2#, since the substructure cannot occur without at least one bond cleavage and replacement. With a bond cleavage and replacement, the substructure will occur on the left.]

Now suppose we need the structure of dioxin, which is a derivitive of dibenzo-*p*-dioxin. The tuple in CHEMICAL–ENTITY gives only an M-CODE value ("DOX"), indicating that the substance is derived from other substances, portrayed in the matching records of MOLECULAR–BOND. These records show that dioxin is formed from dibenzo-*p*-dioxin by replacement of hydrogen atoms on four carbon atoms by chlorine atoms. The carbon atom occurrence numbers match the IUPAC numbers (2, 3, 6, 7) and in the MOLECULAR–BOND tuples these carbon atoms are shown as belonging to the dibenzo-*p*-dioxin molecule. To get at the structure of dibenzo-*p*-dioxin, we have two pathways:

(1) A-path via DBZ ATOMIC–BOND tuples that give the structure only in terms of constituent atoms but with correct IUPAC carbon atom occurrence numbers;

(2) M-path via DBZ MOLECULAR–BOND tuples that give the structure in terms of benzene rings but in the end without correct IUPAC carbon atom occurrrence numbers.

Obviously we take pathway (1). The correct IUPAC structure for dioxin is shown in Figure 4.

Some rules are needed for correct use of the database. Suppose we have an unknown compound XYZ, whose structure we wish to determine. We use the following general algorithm:

(1) access the record for XYZ in CHEMICAL–ENTITY.

(2) if there is an A-CODE value and no M-CODE value, the compound has no substructure but other compounds may be derived from it (that is, it may occur within other compounds). Use the reference to ATOMIC–BOND to get the atomic structure with correct IUPAC carbon atom occurrence numbers, that is, follow the A-path.

(3) If there is an M-CODE value the compound has substructure, and if there is an A-CODE in addition the compound is used to derive other compounds (at a higher level). Follow the A-path using A-CODE to get the atomic structure of XYZ with correct IUPAC carbon atom occurrence numbers. Follow the M-path down through multiple levels, where possible, only to determine constituent substructures within XYZ (including atoms).

(4) If there is an M-CODE value and no A-CODE value, the compound has substructure but is not used to derive any other compound. Follow the M-path to the next level down and then use the A-path to obtain the final IUPAC atomic structure for XYZ. Continuation on the M-path down through multiple levels, where possible, will give the substructures within XYZ.

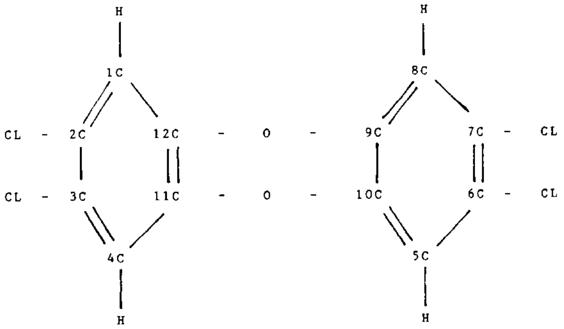


Figure 4. IUPAC Structure for dioxin

Avoidance of replication in the ATOMIC–BOND relation

At first glance it may be thought that the ATOMIC–BOND relation in the two path database suffers from the same defect as that in the simple single-level database in Figure 1, namely that every substructure, such as a benzene ring, must be replicated for every compound in which it occurs. Careful consideration shows that this is not the case. Replication for benzene occurs only once for a class of molecules with similar structures. For example, for the large class of compounds derived from dibenzo-*p*-dioxin, the structure of a pair of benzene rings occurs only once, so that a great reduction in replication is achieved.

Two-path structure diagrams

An initial disadvantage of a two-path database is that it can be difficult for a beginner to visualize the way in which the data is structured; because of the recursive nature of the database, the structure diagram in Figure 3 is not much help. The author has found that what can be called AM-pathway diagrams are a great help. There is an example in Figure 5, which applies to the database in Figure 2. A black rectangle indicates a (black box) entity, whose internal structure is not known, whereas an open box indicates a known structure with correct IUPAC numbers. A dashed arrow indicates an M-path, and a solid arrow indicates an A-path.

From the diagram, it can be seen at a glance that DOX is derived from a single DBZ, that a DBZ can be considered as made up either of two BNEs or just a collection of atoms, and that HDB is derived from a BNE and an OH group, each of which is a collection of atoms.

An AM-pathway diagram is probably also indispensible for anyone designing a two-path database. A designer can choose between many different possible structures, since, for a given large set of compounds, a large number of correct two-path databases are possible. For example, a two-path database could be designed with dioxin shown as derived

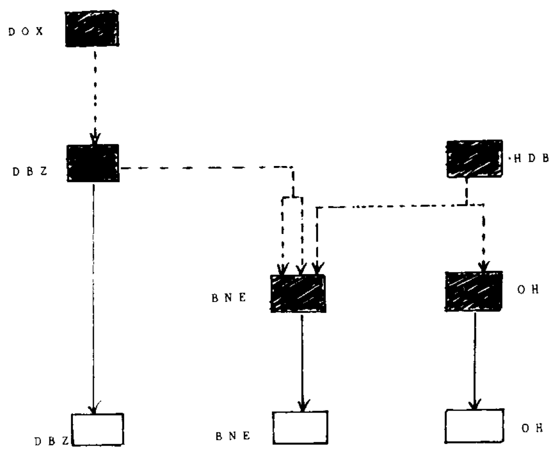


Figure 5. Example of an AM-pathway

from dibenzo-*p*-dioxin as in Figure 2 and Figure 5. However, it would not be wrong to show dioxin as derived only from benzene—it would simply cost more in disk storage.

Note that there is nothing theoretical about the two-path database approach. Provided there is access to a relational database system, such as ORACLE, it is a straightforward matter to build a small two-path database for chemical compounds. A large database is admittedly a major effort, and construction of the AM-pathway diagram for the database first is recommended.

Current lack of a recursive version of SQL

A difficulty, which appears to be temporary, is the current lack of a commercial SQL implementation that can handle an unknown number of levels of recursion. This is a well-known SQL limitation. Language extensions to SQL have been proposed, for example SQL/NQ,[13] which will allow for recursion, but have not yet been implemented. (A recursive SQL implementation is no trivial undertaking, and is a major software development involving many staff-years.) This means that we cannot currently apply SQL when we do not know beforehand how many levels are needed, for example, in the retrieval:

How many benzene rings are there in dioxin?

However, this does not mean that the two-path structure is defective. It merely means that it cannot be fully exploited until we have a commercially available recursive extension to SQL, which cannot be too far in the future.

Correlation between structures obtained by two different pathways

Occasionally the user will encounter a difficulty in relating structures generated by using the A and M paths separately, and for which there appears to be no simple solution. For example, with dibenzo-*p*-dioxin the A-path gives the correct structure in terms of IUPAC numbers. The M-path gives the correct constituents of the structure, but with carbon occurrence numbers that are not the correct IUPAC numbers. There is no way, using only SQL, to relate the carbons of the structures generated by the separate pathways.

We thus have a problem relating a substructure from any entity, obtained via the M-path, to the corresponding portion of the entity's complete structure, with correct IUPAC numbers, obtained via the A-path.

The probable solution is a combined molecular data and knowledge base, equipped with rules for correlating entities generated by the A and M paths, together with the manipulation system for performing the inferences. However, for the present, when the structures are displayed by the user, in most cases, where needed, the A and M pathways correlations should be either obvious, or capable of being simply deduced by the user.

Two-path database for proteins and enzymes

The database structure in Figure 2 appears to be capable of holding the structure of all known chemical compounds in conformance with IUPAC numbering rules. It is interesting to consider how structure information about protein molecules would be held in a two-path relational database. Because of their variety, importance and complexity, database techniques are particularly relevant to protein and enzyme molecular information systems.[2,3,14–16]

A chain of amino acid residues is a polypeptide, and a protein is a group of one or more (usually intertwined) long polypeptide chains occurring naturally. Note that conventionally a chain, such as that for three amino acid residues (A, B, C) in a chain A–B–C, begins with an amino group, so that the chain is HO–A–B–C–H and not H–A–B–C–OH

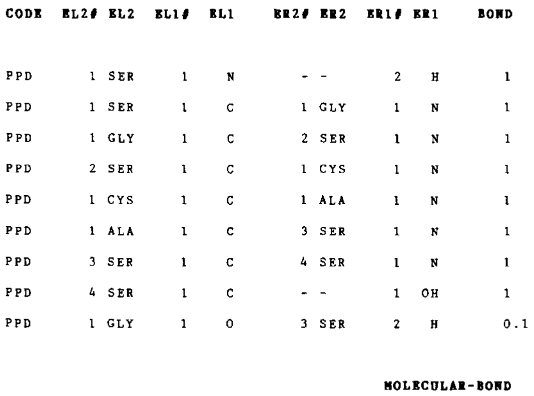
In addition to the polypeptide bonds that form the chain, in a protein there can be

(1) cross-link bonds between side chains and
(2) hydrogen bonds between residues.

In a protein, there are usually a few of these cross-link bonds between side chains of different amino acids. Furthermore, parts of the sequence of a chain can coil into a helix (α-helix),[17] and for this to happen the hydrogen on the nitrogen of residue n will hydrogen bond to the oxygen on residue $n + 4$.

With a two-path database the structure of a protein in terms of amino acid residues would be held in MOLECULAR–BOND. Each tuple would describe a link between two amino acid residues of a chain. There could be several chains. In addition there would be a tuple for the H–N bond at one end of a chain and a C–OH bond at the other. This is illustrated in Figure 6, which shows the 7-residue polypepide serine–glysine–serine–cysteine–alanine–serine–serine (CODE "PPD"). The database also shows a hydrogen bond between residue 2 (glycine) and residue 6 (serine).

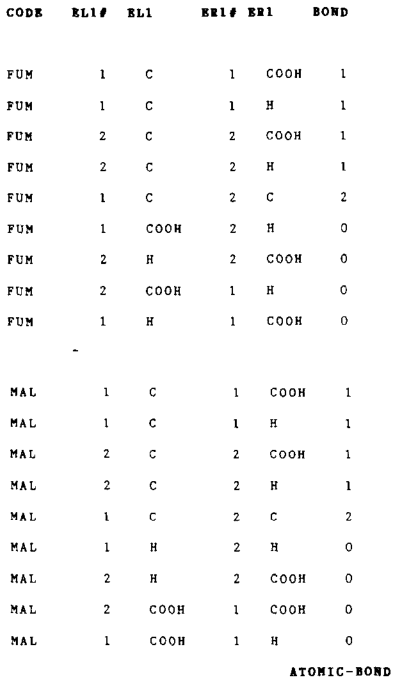
The atomic structure of amino acid residues of a protein would be stored mostly via ATOMIC–BOND, and sometimes via both MOLECULAR–BOND and ATOMIC–BOND. For example, alanine would be stored only in ATOMIC–BOND. But since serine is essentially a derivative of alanine, by hydroxylation, its structure would be obtainable via MOLECULAR–BOND which would portray a C–H bond cleavage in alanine and replacement by C–OH. There would also be tuples in MOLECULAR–BOND for



CODE	EL2#	EL2	EL1#	EL1	ER2#	ER2	ER1#	ER1	BOND
PPD	1	SER	1	N	-	-	2	H	1
PPD	1	SER	1	C	1	GLY	1	N	1
PPD	1	GLY	1	C	2	SER	1	N	1
PPD	2	SER	1	C	1	CYS	1	N	1
PPD	1	CYS	1	C	1	ALA	1	N	1
PPD	1	ALA	1	C	3	SER	1	N	1
PPD	3	SER	1	C	4	SER	1	N	1
PPD	4	SER	1	C	-	-	1	OH	1
PPD	1	GLY	1	O	3	SER	2	H	0.1

MOLECULAR-BOND

Figure 6. Two-path database showing 7-residue polypeptide



CODE	EL1#	EL1	ER1#	ER1	BOND
FUM	1	C	1	COOH	1
FUM	1	C	1	H	1
FUM	2	C	2	COOH	1
FUM	2	C	2	H	1
FUM	1	C	2	C	2
FUM	1	COOH	2	H	0
FUM	2	H	2	COOH	0
FUM	2	COOH	1	H	0
FUM	1	H	1	COOH	0
MAL	1	C	1	COOH	1
MAL	1	C	1	H	1
MAL	2	C	2	COOH	1
MAL	2	C	2	H	1
MAL	1	C	2	C	2
MAL	1	H	2	H	0
MAL	2	H	2	COOH	0
MAL	2	COOH	1	COOH	0
MAL	1	COOH	1	H	0

ATOMIC-BOND

Figure 7. ATOMIC–BOND relation with dummy bonds to distinguish fumaric and maleic acid.

any additional (cross-link) bonds between the amino-acid residues of the chain(s).

If the polypeptide chain has n residues, it takes $n + 1$ tuples in MOLECULAR–BOND to describe it if there are no side-chain cross-links or hydrogen bonds between residues. Each cross-link or hydrogen bond would need an additional tuple. Thus PPD with 7 residues and one hydrogen bond requires (7 + 1) + 1, or 9 tuples. It is therefore clear the structure of any protein, no matter how complex, can be stored in atomic detail, using the two-path database structure in Figure 2.

In the case of an enzyme, the structure of the coenzyme would be obtainable via either just ATOMIC–BOND, or via both MOLECULAR–BOND and ATOMIC–BOND.

Isomers

Consider fumaric and maleic acid. These compounds are isomers (geometrical isomers). They have the same chemical structure as far as the existence of specific chemical bonds is concerned, but yet are different because of the directions of the bonds in space. There are other types of isomers, such as optical isomers, involving right-handedness and left-handedness (as in *L*-alanine and *D*-alanine).[5] As presented so far, the two-path database in Figure 3 could not distinguish fumaric and malic acid, nor *L*-alanine and *D*-alanine.

The problem of isomers can be solved as far as a two-path database is concerned, by employing the concept of a zero strength (or dummy) bond. We simply add some additional tuples to the database. The additional tuples all have BOND value 0, indicating a dummy bond between atoms (or groups) that places the atoms, or projections of atoms onto a plane, in an imaginary ring structure. The order in the ring serves to distinguish the isomers. The ATOMIC–BOND relation in Figure 7 uses dummy bonds to distinguish fumaric and maleic acid. (For the sake of brevity, we have treated –COOH as an atom; strictly, in ATOMIC–BOND only atoms may be used.)

The dummy bond technique is simple and effective, and can be applied to all isomeric compounds, without it being necessary to clutter the database with angle and vector data. The very existence of a dummy bond can also be used to denote that a compound is an isomer. Thus the SQL expression:

```
SELECT IUPACNAME FROM
CHEMICAL–ENTITY WHERE A-CODE IN
(SELECT CODE
        FROM ATOMIC–BOND
        WHERE BOND = 0);
```

gives a list of all isomers in the database. (The reader is invited to extract the structures of fumaric and maleic acid from the database and observe the distinction.)

Three-dimensional structure of molecules

The two-path database structure proposed so far can hold the structure of the most complex molecules, but only in terms of constituent groupings and atoms. We have not so far discussed recording of data in the database about the structure of molecules in 3-dimensional space.[17] Three-dimensional structure is important in applications involving the chemical activity of molecules, which depends on surface structure and the electrical potentials (van der Waals potentials) at the surface.[6,15] The surface structures of enzymes is an important area of ongoing research.

The simplest way to include 3-D structure information would be to include Cartesian cordinates for each of the pairs of atoms in a bond tuple of ATOMIC–BOND. Thus for the left atom we would need three fields LX, LY, and LZ, and for the right atom we need RX, RY, and RZ, with all six coordinate fields placed in ATOMIC–BOND. This is the conventional thing to do and we have nothing to add here. However, since compounds will frequently be portrayed in MOLECULAR–BOND as having been derived via bond cleavage and replacement involving compounds stored in ATOMIC–BOND, geometric computation, using the coordinates in ATOMIC–BOND, will be needed for the 3-D structure of these MOLECULAR–BOND compounds (such as DOX and HDB in Figure 5). This nonpredictive approach would enable the 3-D structure of any molecule to be generated.

SUMMARY

A relational database structure, called the two-path structure, has been proposed for holding the structure of all chemical entities. The conventional bill-of-materials recursive database structure for complex objects cannot be applied to a database for chemical entities because of restrictions involving the need to specify the bonding between high level substructures in terms of bonds between atoms, the lowest level entities in the structure. There are further restrictions of design freedom because conventional IUPAC carbon atom occurrence numbers must be obtainable from the database. The two-path database solves the restriction problems by providing two pathways to the structure of complex chemical entities. One pathway allows a descent to the atomic structure in terms of correct IUPAC carbon atom occurrence numbers in just one or two iterations of the structure. The other pathway allows many iterations of descent in terms of substructures, for purposes of referencing substructures at any level in retrievals. A two-path database is recursive in nature.

The database appears to be capable of faithfully recording the structure of all chemical entities, including proteins and enzymes. The two-path approach is based on a tuple per chemical bond. By using negative bond strengths, a substructure can be recorded as a molecule minus some bonds plus some other bonds. Information needed to distinguish chemical isomers can be included quite easily, by using dummy bonds that form imaginary rings.

REFERENCES

1 Adamson, G. W., Bird, J. M., Palmer, G. and Warr, W. A. In-house chemical data bases of Imperial Chemical Industries. *J. Mol. Graphics* 1986, **4**(3), 165–177

2 Allen, F. H. and Kennard, O. The Cambridge Data Base of molecular structures. *Perspect. Computing* 1983, **3**(3), 28–43

3 Bernstein, F. C., et al. The protein databank: A computer-based archival file for macromolecular structures. *J. Molec. Bio.* 1977, **112**, 535–542

4 Editorial: Databases in molecular graphics, *J. Mol. Graphics* 1984, **2**(3), 66–69

5 Morfew, A. T., Todd, S. J. P. and Snelgrove, M. J. The use of a relational database for holding molecule data in a molecular graphics system. *Computers & Chemistry* 1983, **7**(1), 9–16

6 Taylor, R. The Cambridge Structural Database in molecular graphics: Techniques for the rapid identification of conformal minima. *J. Mol. Graphics* 1986, **4**(2), 123–131

7 Kim, W. Relational database systems. *ACM Computing Surveys* 1979, **11**, 185–211

8 Maier, D. *Theory of Relational Databases*. Computer Science Press, Potomac, MD, 1983

9 IBM Corp., IBM DATABASE2 General Information, Form GC26-4073, IBM Corp., Endicott, New York, 1984

10 Relational Technology Inc. *INGRES Reference Manual*. Berkeley, CA, 1984

11 ORACLE Corp., ORACLE SQL/UFI Reference Guide, Menlo Park, CA, 1984

12 Bradley, J. *File & Database Techniques*. Holt, Rinehart & Winston, New York, 1982

13 Bradley, J. Recursive relationships and natural quantifier set theoretic expression techniques. *Computer Journal,* in press.

14 Bryant, S. H. and Sternberg, M. J. E. Comparison of protein structural profiles by interactive computer graphics. *J. Mol. Graphics* 1987, **5**(1), 4–7

15 Jakes, S. E. and Willet, P. Pharmacophoric pattern matching in files of 3-dimensional chemical structures: Selection of inter-atomic distance screens, *J. Mol. Graphics* 1986, **4**(1), 12–20

16 Ricketts, D. Perq: interactive molecular modelling systems. *J. Mol. Graphics* 1987, **5**(2), 63–70

17 Fletterick, R. J., Schroer, T. and Matela, R. J. *Molecular structure: Macromolecules in Three Dimensions*. Blackwell Scientific Publications, London, 1985