Introduction

A considerable amount of work has been done on data bases for molecular structure data [1, 3, 7, 8, 9, 12, 15, 18, 20], although most of this work provided for only limited molecular structures. However, in an earlier paper [7], a comprehensive relational data base structure, based on a two-path approach, was proposed for holding chemical structure data. The proposed data base structure allowed retrieval of chemical structures that contained correct IUPAC carbon atom occurrence numbers, and for retrieval of chemical substructures, recursively, down to the final atomic level. The two-path relational structure can be used with all chemical compounds, regardless of size and complexity.

A data base structure capabable of holding the substructure of every conceivable type of molecule is clearly useful, but new chemical structures result from chemical reactions and processes, so that a molecular structure data base is not complete without an associated data base for chemical reactions and processes. Alhough a comprehensive molecular structure data base would be large, given the large number of known molecules, a chemical process data base would be even larger, since any one compound can participate in a large number of reactions and processes. Nevertheless, it is clear that a complete chemical (as opposed to physical chemical) knowledge base would consist of a comprehensive molecular structure data base together with a comprehensive chemical reaction/process data base. Although some constructive work has been done on chemical

process data bases, particularly in chemical engineering [2, 4, 16], there appear to be no reports in the literature on how to structure a comprehensive chemical process data base.

In this paper a proposal for the structure of a relational data base for chemical reaction and chemical process data is presented. For the purposes of the paper a reaction is merely a chemical process that cannot be broken down into subprocesses; and a process is a chemical process, involving one or more reactions, and not a physical process, such as a diffusion process for separation of compounds. Thus some processes are reactions and do not break down, whereas other processes are not single reactions and break down into either a set of processes or set of reactions, or any combination of the two.

Two problems had to be solved in the development of a comprehensive relational data base structure. The first problem was how to have a relation for chemical process data when the number of input compounds (reactants) and output compounds (products) varies from one chemical process to another, given that the major feature of a relation is that the number of fields or tuples is fixed [5, 14]. There are only two possible solutions and both are presented.

The second problem is the need for a universal data base structure to handle the subdivision of processes into subproceses, subsubprocesses, and so on, with many levels of substructure, down to reactions, when any given level of the breakdown can have a substructure that can be a process sequence, process cycle (such as the Krebs cycle), process fan, process funnel, for example, or an arbitrarily complex structure. Many solutions are possible, mostly

of great complexity, where it is difficult to be sure that every process substructure can be handled. The solution presented in this paper is relatively simple (although somewhat counter-intuitive), and is guaranteed to handle every conceivable process breakdown.

A relational data base was chosen for the project since it is widely agreed that relational data bases are are superior to every other kind of data base [5, 13, 14]. There are major advantages to using relational data bases to hold both chemical structure data and chemical process data. Conceptual files are constrained to be relations, so that mathematical set theory and predicate calculus can be applied to them. The outstanding advantage is that even with complex ad hoc searches of the data base, it is usually unnecessary to write a program in a procedural language, as is the case with ordinary computer files and earlier data base approaches [5]. It is necessary only to specify what kind of data is to be retrieved using a non procedural relational data base manipulation language, of which the most common is SQL [5, 14]. Without the constraint that conceptual files be relations, SQL would be impossible.

SQL can be used with the data base proposed in this paper, and examples of its use are included. SQL is the standard data base manipulation and retrieval language with such common relational data bases systems as DATABASE2 [10], ORACLE [11], and INGRES [17].

Basic chemical reactions data base

The proposed data base structure is best understood by considering the data base in two steps. The first step simply involves a data

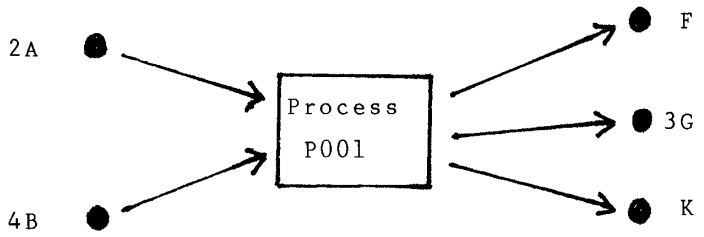

FIGURE 1

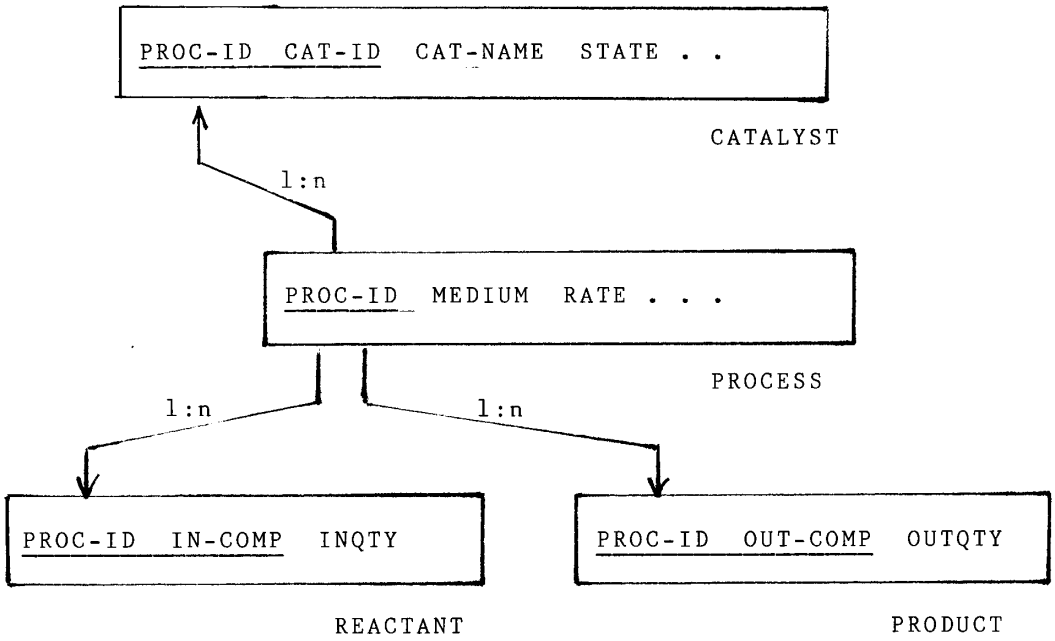

FIGURE 2a

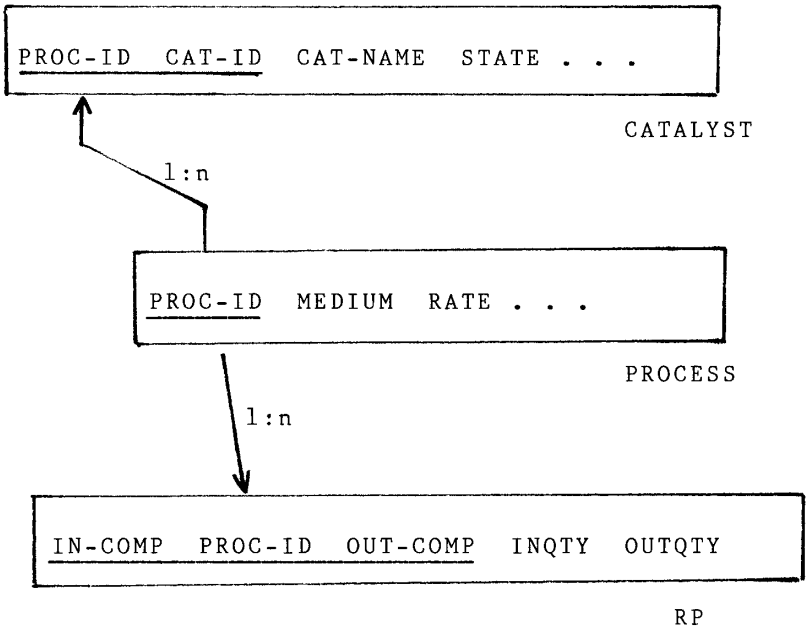
PROC-ID CAT-ID CAT-NAME STATE . . .
CATALYST
1:n
PROC-ID MEDIUM RATE . . .
PROCESS
1:n
IN-COMP PROC-ID OUT-COMP INQTY OUTQTY
RP

FIGURE 2b

base about reactions and processes, with no consideration of how a process can subdivide into subprocesses, and so on. Thus no distinction can be made initially between a reaction and a process, since the data base has no information on any further breakdown. Both have reactants and products, and physical properties, such a reaction rate data, and so on.

Because the number of reactants and processes varies from one reaction to another, a single relation for reaction/process data is not possible, since a relation, by definition, must have a fixed number of fields or attributes. In fact, using relational data base terminology, the dependency between a process (or reaction) and its reactants and products is multivalued [5, 14], since for one process there is a set of reactants and a set of products (Figure 1). Because of this multivalued dependency there are only two ways to place the reactant and product data in relations, as illustrated in Figures 2a and 2b.

In Figure 2a there are four relations, where primary keys (unique record identifiers) are underscored. One of these , PROCESS(PROC-ID, MEDIUM, RATE, EQUILIB, ...) is straight forward. Each record or tuple describes a process (or reaction) in terms of an identifier (PROC-ID) and in terms of physical properties (such a gaseous or aqueous (MEDIUM), reaction rate rate (RATE), equilibrium constant (EQUILIB), and so on, depending on user needs). A process identifier (PROC-ID) is necessary. This would have to be a numeric, alphabetic or alphanumeric code capable of serving the number of processes anticipated for the data base. (However, an international agreement, by which every known chemical process is assigned an

agreed code, is preferable). For purposes of this paper a 4 character code with process identifiers from P001 to P999 will be used.

The second relation is CATALYST (PROC-ID, CAT-ID, CAT-NAME, STATE, ...), each record of which holds data about a catalyst for the reaction or process. CAT-ID gives the identifier for the catalyst, PROC-ID gives the process identifier for the process in which the catalyst is used, CAT-NAME gives the common name of the catalyst, and the other fields give descriptive data about the catalyst, such as STATE (solid, liquid, powder, hot wire, ...), and so on. This relation is needed because some reactions, and many processes, require more than one catalyst. Thus catalyst data simply cannot be placed in the the PROCESS record for a specific process or reaction.

The other two relations in Figure 2a are REACTANT (PROC-ID, IN-COMP, INQTY) and PRODUCT (PROC-ID, OUT-COMP, OUTQTY). A record in REACTANT identifies a process with PROC-ID, gives an input compound (IN-COMP) or reactant molecule of the process, and gives the quantity of the reactant moleculue (INQTY) required to balance the reaction equation for the process. Similarly, a record in PRODUCT identifies a process (PROC-ID), and gives an output compound (OUT-COMP) or product molecule, and gives the quantity of the product molecule (OUTQTY) required to balance the reaction equation. Thus, with the data base in Figure 2a, the data for the reactions:

P001: $2A + 4B = F + 3G + K$

P002: $4P + A = 3S + 4F$

would be in the relations REACTANT and PRODUCT as follows:

PROC-ID	IN-COMP	INQTY
P001	A	2
P001	B	4
P002	P	4
P002	A	1

REACTANT

PROC-ID	OUT-COMP	OUTQTY
P001	F	1
P001	G	3
P001	K	1
P002	S	3
P002	F	4

PRODUCT

For the purposes of the paper we use upper case letters to identify chemical compounds. In practice, the identifiers used in a molecular structure data base, typically IUPAC names, would be used.

It should be understood that REACTANT essentially lists the input compounds to processes, with a single record for each input compound to a specific process. The primary key is a therefore a composite of PROC-ID and IN-COMP. Similarly, PRODUCT lists the output compounds from a process, with a single record for each output compound from a specific process, with PROC-ID OUT-COMP as a composite primary key. This may seem counter-intuitive. A structure where each record listed all the input compounds, or all the output compounds, to a process is more appealing. But because such records would have varying numbers of fields, and are therefore not relations, they cannot be used.

Note the relationships in Figure 2a. There is a one-to-many (1:n) relationship [5] between PROCESS and REACTANT, faciliated by the field PROC-ID, since for one process there can be many reactants. There is also a 1:n relationship between PROCESS and PRODUCT, also based on PROC-ID, since for on process there can be many products. Finally, there is a 1:n relationship between PROCESS and CATALYST, also based on the field PROC-ID.

(Actually, the relationship between catalysts and processes is many-to-many, since a catalyst can be used in many processes and vice versa. However, the proposed design use of the 1:n relationship instead is simpler, at the usual expense of an increase (very slight in this case) in data redundancy, since a record of CATALYST describes a catalyst used in a process, and not merely a catalyst. As a consequence the primary key is the composite PROC-ID CAT-ID. Note that, in the proposed data base, if a given type of catalyst X is used in two separate processes, perhaps in different states, two separate CATALYST records will be needed. Furthermore, if a parent process breaks down into subprocesses, two or more of which used the same catalyst X, perhaps in different forms or states (such hot wire in one subprocess, and particulates in another subprocess), the CATALYST record for that parent process would need to use a STATE value such as MULTIPLE; this would indicate that the detailed data for the X catalyst with each of the subprocesses is in the group of CATALYST records for those subprocesses. See the discussion later on subprocesses.)

The alternative data base in Figure 2b can be derived from the data base in Figure 2a. The relations PROCESS and CATALYST are

the same. However the relation RP (IN-COMP, PROC-ID, OUT-COMP, IN-QTY, OUTQTY) is a natural join [5, 14] of the relations REACTANT and PRODUCT with join field PROC-ID. A record of RP is even less intuitively appealing than the records of REACTANT and PRODUCT. A record of RP essentially lists one of the input compounds to a process and one of the output compounds. For any process there will be a record for each combination of reactant and product. Thus if there are 3 reactants and 4 products, it will take 12 RP records to describe the process. This is a consequence of the fact that RP contains a multivalued dependency [5, 14]. It is also means that there is a gread deal of data redundancy in RP, which is removed by replacing it with REACTANT and PRODUCT. The relation RP equivalent to the relations REACTANT and PRODUCT above is:

IN-COMP	PROC-ID	OUT-COMP	INQTY	OUTQTY
A	P001	F	2	1
A	P001	G	2	3
A	P001	K	2	1
B	P001	F	4	1
B	P001	G	4	3
B	P001	K	4	1
P	P002	S	4	3
P	P002	F	4	4
A	P002	S	1	3
A	P002	F	1	4

RP

Note the 1:n relationship between PROCESS and RP in Figure 2b, facilitated by the field PROC-ID. It is not very obvious but is based on the existence of many pairs of reactant and product for a given process.

RP is equivalent to the pair of relations REACTANT and PRODUCT, since RP is obtained from a join of the pair, and, furthermore, both REACTANT and PRODUCT can be regenerated from projections [5, 14] on RP. In normal data base practice RP would be replaced with REACTANT and PRODUCT, primarily since a relation with a multivalued dependency gives rise to updating difficulties, and secondarily because of the associated reduncdancy. However, since the reactants and products for a particular process are fixed, and

need never be subject to updating, either RP or the equivalent REACTANT and PRODUCT may be used. However, the author recommends use of REACTANT and PRODUCT, and not RP, mainly on grounds of ease of use.

These data base structures may give rise to some initial difficulties for unskilled users applying SQL to retrieve data from the data base. However, these difficulties are easily overcome if the user is proficient in SQL, as the following examples show :

Example 1. Find the products of each process that involves reactants A and B in aqueous solution.

(a) Using the data base in Figure 2a:

```
SELECT PROC-ID, OUT-COMP FROM PRODUCT
WHERE PROC-ID IN
(SELECT PROC-ID FROM PROCESS
WHERE MEDIUM = 'AQUEOUS'
AND PROC-ID IN (SELECT PROC-ID FROM REACTANT
                WHERE IN-COMP = 'A')
AND PROC-ID IN (SELECT PROC-ID FROM REACTANT
                WHERE IN-COMP = 'B'));
```

(b) Using the data base in Figure 2b:

```
SELECT PROC-ID, OUT-COMP FROM RP
WHERE PROC-ID IN
(SELECT PROC-ID FROM PROCESS
WHERE MEDIUM = 'AQUEOUS'
AND PROC-ID IN (SELECT PROC-ID FROM RP
                WHERE IN-COMP = 'A')
AND PROC-ID IN (SELECT PROC-ID FROM RP
                WHERE IN-COMP = 'B'));
```

It should therefore be clear that despite the unintuitive design, necessitated by the need to use only relations, the data base structures in Figures 2a and 2b are quite practical, and could store the essential reactant/product/environment data for every conceivable chemical process. (But in order to store information on how a process relates to its subprocesses, and how each of these subprocesses further relates to its subprocesses, a recursive extension to the data bases is needed). As a final example:

Example 2. What catalysts are used with gaseous processes in which compound K is a reactant?

```
SELECT *
FROM CATALYST WHERE PROC-ID IN
 (SELECT PROC-ID FROM PROCESS
  WHERE MEDIUM = 'GASEOUS' AND PROC-ID IN
        (SELECT PROC-ID FROM REACTANT
         WHERE IN-COMP = 'K'));
```

Data base extension for subprocesses

The way in which a process can break down into subprocesses, and each subprocess into further subprocesses, can be infinitely varied [2]. Readers familiar with biochemical processes, in particular, can testify to that [19]. For a given process, some common breakdowns are:

1. Straight sequence. A processes breaks down into a straight sequence of subprocesses, in which at least some of the output molecules from one process serves as input molecules to the next process in sequence.

2. Cycle. The simplest cycle will have one process feeding molecules into the cycle and one being fed molecules by the cycle. Within the processes in the cycle, at least some of the output molecules from one process will be fed as input

molecules to the next process in the cycle. With more complex cycles there can be more than one process feeding into the cycle and more than one being fed by the cycle.

3. Fan-out. A single process feeds output molecules into more than one sequence of processes.

4. Funnel. More than one process feeds output molecules into a single process.

The above breakdowns listed are not exhaustive. Many other breakdown structures, probably uncommon, are possible. Furthermore, there can be many levels of breakdown. Any of the processes in a breakdown can be a process that breaks down further, in any of the ways listed, and other ways besides.

The problem is how to devise a data base structure that will handle any of the wide variety of possible process breakdowns that can occur. There appears to be two general approaches to the problem. One approach is to devise individual data base structures (and thus relations) that will match the individual process sub-structures, such as those listed above. This approach was extensively reseached and proved futile, mainly because no matter how many additional structures were used, it was apparently always possible to show that there was a possible process breakdown that would not fit any of the proposed structures; thus it was not possible to prove that any of the structures were universally applicable

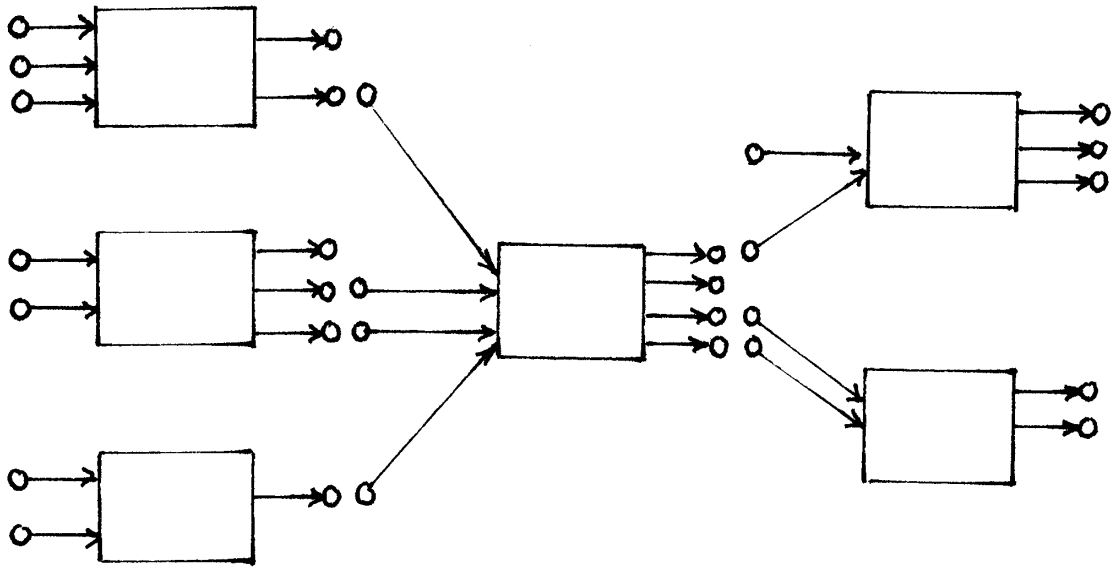

FIGURE 3

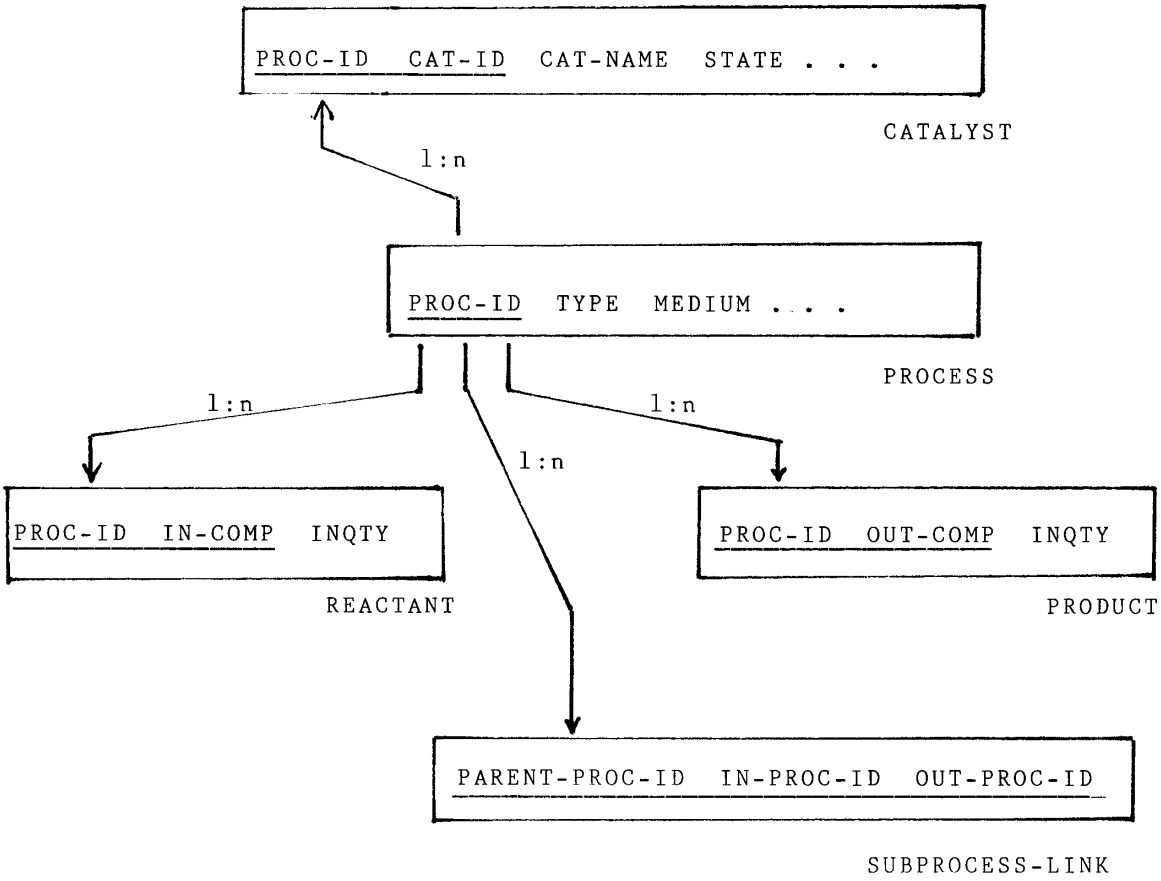
PROC-ID CAT-ID CAT-NAME STATE . . .
CATALYST
1:n
PROC-ID TYPE MEDIUM . . .
PROCESS
1:n
1:n
1:n
PROC-ID IN-COMP INQTY
REACTANT
PROC-ID OUT-COMP INQTY
PRODUCT
PARENT-PROC-ID IN-PROC-ID OUT-PROC-ID
SUBPROCESS-LINK

FIGURE 4

The alternative is to ignore the structure of the breakdowns and simply devise a data base structure that links any process with (a) those processes that feed molecules to it, and (b) those processes that are fed molecules by it. In the simplest implementation there is a record for each link. Because every process substructure, no matter how complex, consists of links between processes, it follows that such a data base structure must be able to handle every conceivable subprocess structure.

This way of looking at any process in a process substructure can be visualized as shown in Figure 3. Notice that the structure is similar in structure to Figure 1. Because for any process, thare can be many processes feeding that process, and many process fed by that process, it follows that a process multidetermines its input processes and its output processes, that is, there is an intrinsic multivalued dependency [5, 14] between a process and its input and output processes.

An extension to the data base in Figure 2, that allows for this new structure is shown in Figure 4. The PROCESS, CATALYST, REACTANT and PRODUCT relations are the same as before, except for the additional field TYPE in the PROCESS relation. A record in PROCESS describes a process (or reaction) and TYPE simply gives a descriptor for the nature of any immediate breakdown of that process. For example, TYPE could have the value "CYCLE", if the process breaks down into a cycle, or "SEQUENCE", if it breaks down into a straight sequence, or "UNCLASSIFIABLE", if it breaks down in some difficult manner, or "NONE" if it does not break down, and so on. The field TYPE is not necessary but is useful for simple retrievals involving the breakdown type of a process.

The additional relation is SUBPROCESS-LINK (PARENT-PROC-ID, IN-PROC-ID, OUT-PROC-ID). A record of SUBPROCESS-LINK simply identifies a pair of processes, such that one process (IN-PROC-ID) feeds molecules to another process (OUT-PROC-ID), where both these processes are subprocesses in a parent process (PARENT-PROC-ID). If a subprocess of the breakdown has no input subprocess feeding it within the group of brealdown subprocesses, IN-PROC-ID will be set to NULL or equivalent; similarly, if a subprocess does not feed any subprocess within the group of breakdown subprocesses, OUT-PROC-ID will be set to NULL or equivalent. Thus if parent process P999 breaks down into subprocesses P100, P200, P300, and P008, where P100 feeds P200, and P200 feeds P300, and, in a second sequence, P100 feeds P008, which feeds P300, then the records of SUBPROCESS-LINK would be:

PARENT-PROC-ID	IN-PROC-ID	OUT-PROC-ID
P999	NULL	P100
P999	P100	P200
P999	P200	P300
P999	P100	P008
P999	P008	P300
P999	P300	NULL

SUBPROCESS-LINK

From SUBPROCESS-LINK the fundamental relation that contains the multivalued dependency, intrinsic to the situation where

a process is fed by many input processes and feeds many output processes (Figure 3), can be generated by a join of SUBPROCESS-LINK to itself (a join of copy X of SUBPROCESS-LINK with copy Y of SUBPROCESS-LINK):

```
SELECT X.PARENT-PROC-ID, X.OUT-PROC-ID, X.IN-PROC-ID, Y.OUTPROC-ID
    FROM SUBPROCESS-LINK X, SUBPROCESS-LINK Y
    WHERE X.OUT-PROC-ID = Y.IN-PROC-ID
```

The relation generated by the above join-performing SQL expression is more easily understood if renamed as the relation LINK-LINK(PARENT-PROC-ID, PROC-ID, IN-PROC-ID, OUT-PROC-ID), where PROC-ID identifies any subprocess of the process PARENT-PROC-ID, IN-PROC-ID identifies a subprocess that feeds that subprocess PROC-ID, and OUT-PROC-ID identifies a subprocess fed by subprocess PROC-ID. LINK-LINK thus links a subprocess both to a subprocess that feeds it and to a subprocess that is fed by it. The instance of LINK-LINK generated from the instance of SUBPROCESS-LINK above would be:

PARENT PROC-ID	PROC-ID	IN-PROC-ID	OUT PROC ID
P999	P100	NULL	P200
P999	P100	NULL	P008
P999	P200	P100	P300
P999	P008	P100	P300
P999	P300	P008	NULL
P999	P300	P200	NULL

LINK-LINK

LINK LINK and SUBPROCESS-LINK are thus equivalent, so that LINK LINK contains a multivalued dependency, in which the composite field PARENT PROC-ID PROC-ID multidetermines IN-PROC-ID and OUT PROC ID. The multivalued dependency can be eliminated by replacing LINK LINK with SUBPROCESS-LINK. SUBPROCESS-LINK is also more concise, so that it is better to use it in the data base in Figure 4 than LINK LINK, although there is nothing wrong with using LINK LINK instead. [Note that with SUBPROCESS-LINK it is remotely possible that a given sequential pair of processes could occur more than once in two separate parts of a breakdown, in which case an additional field OCCURRENCE-NUMBER, with integer values, will be needed to distinguish the two occurrences. Since the chance of this happening is remote, it is otherwise ignored in this paper, although it can be handled easily.]

The relation SUBPROCESS-LINK does not just handle the first level breakdown of a process into its immediate subprocesses. It

will also handle the breakdown of immediate subprocesses into their subprocesses, that is, subsubprocesses, and so on, to any level of breakdown. This is so because the relationship between the relation PROCESS and the relation SUBPROCESS-LINK in Figure 4 is a recursive many-to-many relationship [5, 6, 7, 14]. Recursive relationships can handle multiple-level containments.

Although in principle all many-to-many recursive realtionships are the same, since they all deal with multiple-levels of containment, in practice there are many unique variations on the theme. The relationship between PROCESS and SUBPROCESS-LINK is a unique kind of recursive relationship since it can be used to generate a relation (LINK-LINK) with a multivalued dependency embedded in it. For example, it is quite different from the many-to-many relationship required to handle substructure of molecules [7].

Some SQL retrieval expressions illustrate how the data base might be used:

Example 3. What immediate subprocesses of P489 break down further?

```
SELECT IN-PROC-ID FROM SUBPROCESS-LINK
WHERE PARENT-PROC-ID = 'P489'
      AND IN-PROC-ID IN (SELECT PARENT-PROC-ID
                         FROM SUBPROCESS-LINK);
```

Example 4. Retrieve a full description of each process that has at least two levels of breakdown?

```
SELECT * FROM PROCESS
WHERE PROC-ID IN (SELECT PARENT-PROC-ID
       FROM SUBPROCESS-LINK WHERE PARENT-PROC-ID IN
           (SELECT IN-PROC-ID FROM SUBPROCESS-LINK));
```

Example 5. For each level-2 subprocess (i.e. for each subsub-process) of process P479, determine the subprocess identifier and the catalysts needed with that subprocess.

```
SELECT PROC-ID, CAT-ID, CAT-NAME
FROM CATALYST
WHERE PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
        WHERE PARENT-PROC-ID IN
       (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
        WHERE PARENT-PROC-ID = 'P479'));
```

Example 6 If process P456 has a cyclic level-1 substructure, get the level-1 substructure of that process, complete with reactants and products for each subprocess.

```
SELECT IN-COMP, IN-PROC-ID, OUT-COMP, OUT-PROC-ID
FROM REACTANT, SUBPROCESS-LINK , PRODUCT
  WHERE REACTANT.PROC-ID = SUBPROCESS-LINK.IN-PROC-ID
  AND PRODUCT.PROC-ID = SUBPROCESS-LINK.IN-PROC-ID
  AND PARENT-PROC-ID = 'P456'
  AND PARENT-PROC-ID IN (SELECT PROC-ID FROM PROCESS
                         WHERE TYPE = 'CYCLE');
```

(Note that it is strictly not necessary to include retrieval of OUT-PROC-ID in Retrieval 6, but it helps to identify how a subprocess retrieved fits into the overall process breakdown.)

Lack of a recursive version of SQL

A disadvantage of current releases of SQL is that the language does not have a facility for handling recursion to an unknown number of levels. For example, referring to the data base in Figure 4 again, suppose one wants the lowest level subprocesses of a given process, where it is not known in advance how many recursion levels down the lowest breakdown level lies. There is no SQL expression that can express the retrieval request. This is not a drawback of the data base design, but of SQL. Eventually, a recursive version of SQL will likely become commercially available, enabling

retrievals involving unknown numbers of recursion levels to be expressed. Currently there are two possible ways of dealing with this SQL defect in the case of the chemical process data base in Figure 4.

One approach is to use a distinct SQL expression for each subprocess level. Suppose the lowest level subprocesses of process P567 were needed. A single SQL expression for this is not possible if the number of levels down is not known in advance. However, an SQL expression could first be constructed to retrieve the subprocesses (and process structure if desired) of P567. Then similar SQL expressions could be used to retrieve the subprocesses of each of the retrieved subprocesses, and then similar SQL expressions for the subprocesses of these subprocesses, until finally no subproceses are retrieved and the bottom of the breakdown is reached. To be more specific, for the first level breakdown, the following could be used.

```
SELECT IN-PROC-ID, PARENT-PROC-ID FROM SUBPROCESS-LINK
WHERE PARENT-PROC-ID IN ('P567');
```

If this retrieves subprocesses, P432, P567 and P598, their subprocesses, if there are any, can be retrieved by:

```
SELECT IN-PROC-ID, PARENT-PROC-ID FROM SUBPROCESS-LINK
WHERE PARENT-PROC-ID IN ('P432', 'P567', 'P598');
```

A similar expression can be used to retrieve the subprocesses of these subprocesses, and so on.

The alternative is to guess how many levels there are, and then construct an SQL expression to retrieve the subprocesses at the guessed level. If no processes are retrieved, then the number of levels guessed was too large. The number of levels in the expression can then be reduced by one and the expression tried again. If this fails reduce by one and try again, repeating until some processes are retrieved. These are the lowest level processes. If the first expression did not fail, then increase the number of levels in the expression by one, and repeat until a failure occurs. For example, in attempting to retrieve the bottom level subprocesses of process P567, if the guess is that the bottom is four levels down from the process level, the following expression could be tried:

```
SELECT IN-PROC-ID FROM SUBPROCESS-LINK
 WHERE PARENT-PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
 WHERE PARENT-PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
 WHERE PARENT-PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
     WHERE PARENT-PROCESS = 'P567')));
```

If this retrieves nothing try:

```
SELECT IN-PROC-ID FROM SUBPROCESS-LINK
 WHERE PARENT-PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
 WHERE PARENT-PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
      WHERE PARENT-PROCESS = 'P567'));
```

otherwise try:

```
SELECT IN-PROC-ID FROM SUBPROCESS-LINK
 WHERE PARENT-PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
 WHERE PARENT-PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
 WHERE PARENT-PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
 WHERE PARENT-PROC-ID IN (SELECT IN-PROC-ID FROM SUBPROCESS-LINK
      WHERE PARENT-PROCESS = 'P567'))));
```

Using techniques of this nature the lack of a recursive feature in SQL is no great handicap, and the user can quickly determine the nature of the breakdown of any process.

Graphics display of process and molecular structures

A data base with the structure in Figure 4 could be used to generate data on process substructure that could be displayed by a graphics display system. Displays could be similar to those shown in Figure 3. At the same time, if a comprehensive molecular structure data base [7] were available, the chemical structure of the reactants and products for each process could also be displayed. This would be especially valuable when the process (and breakdown subprocesses) involved changes to a giant molecule, such as an enzyme, protein or DNA molecule [8, 12]. If the giant molecule were involved in a complex series of changes, the processes for which were in the process data base, then the sequence of molecular structure changes could easily be displayed.

Summary

A relational data base for data about chemical processes is proposed. The data base can hold data about the physical attributes of chemical processes and about their reactants and products. It includes a relation about subprocesses, embodying a recursive many-to-many relationship. This recursive relationship holds data about the breakdown of any process into its subprocesses, and the breakdown of these subprocesses into their subprocesses and so on, to any desired breakdown level, and regardless of the complexity of the subprocess substructure (sequence, cycle, funnel, fan or other arbitrary structure).

The relational data base language SQL, commonly available commercially, can be used with the data base. Data about processes, in terms of physical properties, reactants and products, and breakdown into subprocesses, can be retrieved easily, normally by means of one or two SQL expressions. Because current releases of SQL do not have facilities for recursive expressions, SQL retrieval expressions involving descent to unpredictable subprocess breakdown levels cannot be constructed, unfortunately. However, it was shown that this difficulty can be circumvented by systematic use of multiple SQL expressions.

The chemical process data base structure presented in this contains no facilities for data about the structure (and substructure) of the molecules involved in the processes. However, in an earlier paper a relational data base structure was presented for

structure and substructure, down to the atomic level, of chemical compounds [7]. Where molecular substructure data is needed in addition to chemical process data, then both the chemical process data base and the molecular structure data base could be used. These data bases together would be especially useful with a graphics system, for displaying the effect of a sequence of chemical processes on molecular structure, particularly when giant molecules are involved.

References

1. Allen F.H., Kennard, O., 1983. The Cambridge Data Base of molecular structures, Perspect. Computing, 3(3), 28-43.

2. Attias, R., Dubois, J. E., 1990. Substructure systems, concepts and classification, J. Chem. Inf. Comput. Sci., 30(1), 2-7.

3. Bernstein, F.C., et al, 1977. The protein databank: A computer-based archival file for macromolecular structures, Journal of Molecular Biology, Vol. 112, 535-542.

4. Berzins, V., Jones, K., 1989. Maximizing the potential of process engineering databases, Comput. Integr. Process Eng., 114, 225-41.

5. Bradley, 1982. File & Database Techniques, Holt, Rinehart & Winston, New York.

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

7. Bradley, J., 1990. A two-path recursive relational database structure for molecular information systems, J. Mol. Graphics, Vol. 8, 2-10.

8. Bryant, S.H., Sternberg, M.J.E.,1987. Comparison of protein structural profiles by interactive computer graphics, Journal of Molecular Graphics, 5(1), 4-7.

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

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

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

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

13. Kim, Won, 1979. Relational database systems, ACM Computing Surveys, 11, pages 185-211.

14. Maier, D. 1983. Theory of Relational Databases, Computer Science Press, Potomac, Maryland.

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

16. Motard, R. L., 1989. Integrated computer-aided process engineering, Comput. Chem. Eng., 13(11-12), 1199-206.

17. Relational Technology Inc., 1984. INGRES Reference Manual, Berkeley, Calif.

18. Ricketts, D., 1987. Perq: interactive molecular modelling systems, Journal of Molecular Graphics, 5(2), 63-70.

19. Stryer, L., 1988, Biochemistry, W.H. Freeman & Co., New York.

20. Taylor, R, 1986. The Cambridge Structural Database in molecular graphics: Techniques for the rapid identification of conformal minima, Journal of Molecular Graphics, 4(2), 123-131.