Introduction

This paper deals with relational data bases [10, 12] that contain what are commonly referred to as recursive [2], cyclic [17, 18], or bill-of-materials [13] many-to-many (n:m) associations. The two prominant applications are bills of materials and parts in manufacturing and design [2, 3, 13,17, 18] and the family tree of corporations [6]. We intend to consider this type of association in the light of requirements for manipulation by non procedural relational languages [10], particularily SQL [1, 7, 8, 14,16,19, 20], and an upward compatible extension to SQL called SQL/N [3, 4, 5] that depends in part on a relational data model that captures more meaning about the associations involved [5]. This data model involves the schema specification of modes of association,in addition to the usual relational data base specifications. The data model is thus much simpler than the RM/T model [12], in that the only additional construct is the mode of association.

We can say that a relation R_i participates in a recursive n:m association, that is, there is an n:m assocation between R_i and itself, if R_i contains a primary key attribute A_{ik}, and there exists a further relation $R_s(A_{s1}, A_{s2}, A_{sk}, \ldots)$ where attributes A_{s1} and A_{s2} are non primary keys drawn on the same domain as A_{ik}. As a result, there will be a 1:n association between R_i and R_s supported by the attribute pair (A_{ik}, A_{s1}) and a further 1:n association between R_i and R_s supported by the pair (A_{ik}, A_{s2}).In these associations R_{s1} and R_{s2} are foreign keys [12, 13, 14, 20] or "connection fields" [3, 4].

The recursive association between R_i and itself is an example of a non primitive association [5], since if we take a pair of associated R_i tuples, we cannot find a corresponding pair of matching attribute values. With a primitive association, such as a 1:n association between R_i and R_s, such a pair of matching attributes (A_{ik} and A_{s1} or A_{s2} values) will exist (except where the association is not based on equality of the attributes supporting it [5], a rare occurrence). With a non primitive association there is no matching pair of attributes in associated tuples. Instead, two relations

R_i and R_j are non primitively associated iff there exists a set of intermediate relations R_1, R_2, ... R_n, so that R_i is primitively associated with R_1, which is in turn primitively associated with R_2, and so on, with R_n finally associated with R_j. In other words, R_i and R_j are non primitively associated if they are at either end of a chain of primitively associated relations. If $j = k$, then we have a non primitive recursive association, the association type we are principly concerned with in this paper.

The mode of association concept

The fundamental mode of association concept (for non recursive situations) has been developed in detail in [5],although we shall repeat the essentials here, as we are intending to extend the concept for use with recursive associations. Modes of association have two major uses: (a) they permit the application of both the basic and natural quantifiers to all types of association in a manner similar to natural language, and (b) they permit the construction of an intent analyser in a query reduction system. When we have developed suitable modes of association for the recursive n:m association, readers will be able to see how it is that the combination of modes of association and natural quantifiers [2, 3, 4, 5] can give rise to greater expressive power than is available in SQL. SQL/N permits use of both natural quantifiers and modes of association.

Consider two relations $R_A(A_1, A_2, ...)$ and $R_B(B_1, B_2, ...)$, where there is an association supported by the common domain attributes A_d and B_d, thus making the association a primitive one. Two tuples $a(a_1, a_2, ..., a_d ...)$ and $b(b_1, b_2, ..., b_d, ...)$ are thus associated primitively if $a_d = b_d$. However, we need to know the meaning, if any, of the association. There could well be no meaning at all, so that although the two tuples are technically associated, the association (mode) could be entirely coincidental.

A mode of association is a set of propositions in logic $P(e_1, e_2)$, belonging to a general proposition type P that describes the association, where e_1 and e_2 are entities that are the subject and object of such a proposition, and are also entities described by a pair of associated tuples for which P is a mode.

We can illustrate this initially with a simple version of the Supplier-Parts [13] data base, where the supplier relation SR is R_A, and the part types relation PT is R_B:

SR(S#, STATUS, CITY)   PT(P#, COLOR, PCITY)

where CITY is the location of a supplier, and PCITY is the city where a part type is manufactured. Thus CITY is A_d and PCITY is B_d. Since PCITY and CITY are drawn on the same domain, they support a primitive association between SR and PT. We might have the modes of association W, P, and Q describing this association, where the symbolisms below represent the individual propositions:

W(S1, P2)	Supplier S1 is in the city of manufacture of P2
W(S3, P1)	Supplier S3 is in the city of manufacture of P1
...	
P(S2, P5)	Supplier S2 can same-day deliver part type P5.
P(S4, P3)	Supplier S4 can same-day deliver part type P3.
...	
Q(S2, P1)	Supplier S2 inventories part type P1
Q(S1, P4)	Supplier S1 inventories part type P4.
...	

Thus each of the association modes W, P, and Q can be seen to be a set of propositions about the supplier and part type entities that tell us more about the association that is supported by the pair (CITY, PCITY). The mode W is clearly a coincidental mode, as any proposition in this mode about two related supplier and part type entities merely states that they have a common city of location and manufacture. The other modes give greater meaning to the association. Thus a complete set of modes for a given association will fully describe the semantics of the association.

An association is fully defined if we have a system of hypothetical propositions that enable all modes to be determined. For example, if we consider the association between the relations R_A and R_B, where the association is supported by the attribute pair (A_d, B_d), and A_1 and B_1 are the primary key attributes, then the following could define the modes of association W, F and G:

(a) $\forall a \forall b[(a.A_d = b.B_d) \Leftrightarrow W(a.A_1, b.A_2)]$

(b) $\forall a \forall b[W(a.A_1, b.A_2) \Rightarrow F(a.A_1, b.A_2)]$

(c) $\forall a \forall b[W(a.A_1, b.A_2) \Leftrightarrow G(a.A_1, b.A_2)]$

In the above system a and b are tuple variables. Each hypothetical proposition defines or generates a mode of association, with the generated mode on the right. The generator for the coincidental mode W must contain a biconditional; the other generators can have either a biconditional or implication. A set of modes, such as W, F and G, must be consistent, complete and independent if it is to fully describe an association. This is dealt with in [5].

As an example of the utility of such modes, suppose that we have the mode generators for the Supplier-Parts data base above for the primitive association between suppliers (SP) and part types (PT), as follows:

(a) $\forall SX \forall PX[(SX.CITY = PX.PCITY) \Leftrightarrow W(SX.S\#, PX.P\#)]$

(b) $\forall SX \forall PX[W(SX.S\#, PX.P\#) \Leftrightarrow SHIPPER(SX.S\#, PX.P\#)]$

Here SX and PX are tuple variables, and the individual proposition SHIPPER(S1, P3) is one of the set SHIPPER(SX.S#, PX.P#), and represents the English language proposition:

> Supplier S1 is a same day shipper of part type P3.

The expression SHIPPER SR then refers to to the suppliers that are same-day shippers of a (to be specified) part type. We can better illustrate this concept with the following retrieval request for the Supplier-Parts data base:

> Find the part numbers for each part type that can be same-day shipped by at least 3 supplier of status 10.

To begin with, it is clear that the SQL expression would be:

```
SELECT P# FROM PT, PTX
WHERE 3 >=  (SELECT COUNT(*) FROM SR
             WHERE STATUS = 10
             AND SR.CITY = PTX.CITY)
```

We note that the user must know that same-day shipment is the equivalent of matching CITY and PCITY values. The system has no information about the semantics of the association involved, and certainly cannot check that the expression structure is consistent with these semantics.

With SQL/N, we need schema definitions of the modes of association, to permit the data base system to capture more information about an association, and thus permit it to interpret any mode of association used in an SQL/N expression. It is possible to construct an intent analyser as part of the query reduction system, to check that the SQL/N expression structure is consistent with the semantics involved [5]. The SQL/N expression is:

```
SELECT P# FROM PT
WHERE FOR AT LEAST 3 SHIPPER SR TUPLES (STATUS = 10)
```

In the above expression, FOR AT LEAST 3 is a natural quantifier [2, 3,4,5], specifying a quantity of tuples for which specified conditions must hold. In carrying out the retrieval request we can conveniently imagine that each PT tuple is checked in turn, as is conventional with SQL. For a particular PT tuple being checked, the group of SHIPPER SR tuples are also checked, that is, the specific SR tuples that are associated by the association for which SHIPPER is a mode, specified in the schema. Because of the quantifier, at least 3 of these associated SR tuples must have a STATUS value 10.

The system will use the mode SHIPPER to determine the underlying concidental mode, and the fact that the association is based on equality of CITY and PCITY values. Using that part of the reduction system called an intent analyser, the system can check for retrieval requests that are inconsistent with the logic of the mode generators for the association based on the pair (CITY, PCITY), although the details of this are beyond the scope of this paper [5].

Level-1 association modes for recursive n:m associations

Let us now return to the relations R_i and R_s, where there are two separate primitive 1:n associations between R_1 and R_s, and where in consequence R_i is n:m non primitively associated with itself. Again, let A_{ik} be the primary key of R_i and A_{s1} and A_{s2} non key attributes in R_s that are drawn on the same domain as A_{ik}. In order to more easily follow the subsequent analysis and discussion, it is convenient to regard A_{ik} as identifying a part type, and A_{s1} as identifying a part type that contains a part type identified by A_{s2}. Alternatively, A_{ik} can be regarded as identifying a corporation, where A_{s1} identifies a corporation that has acquired another corporation identified by A_{s2} in a merger identified by A_{sk}. Thus in the former case a tuple of R_s would describe the position of one part (A_{a2}) within another (A_{s1}). In the latter case a tuple of R_s would describe an acqusition of one firm (A_{s2}) by another firm (A_{s1}), that is, what is refered to as a merger in financial circles. A diagram showing the two 1:n associations is given in Figure 1. For convenience of exposition we include a key attribute in R_s, namely the attribute A_{sk}. This attribute is not necessary for our theory, but it does make it simmpler to define many of the modes of association required.

As is well-known, a many-to-many recursive association involving a relation R_i will generate either explosions or implosions [2, 3, 13,17,18]. If an R_i tuple describes a part type, then an explosion for a given part type gives the parts that part contains, the parts the contained part contains, and so on in hierarchical fashion, until we reach those parts that are atomic and thus contain no further parts. Conversely, an implosion for a given part type gives the parts that contain the given part type, the part types that contain these part types and so on until we come to those part types that are not contained by other part types. Thus we can distinguish three situations: (a) where we are dealing only with a given part type and the first level of the explosion or implosion hierarchy, (b) where more then one level, but not

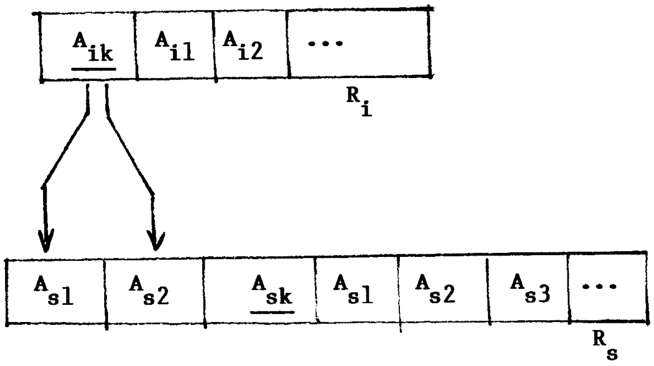

Figure 1. Primary key attributes underlined.

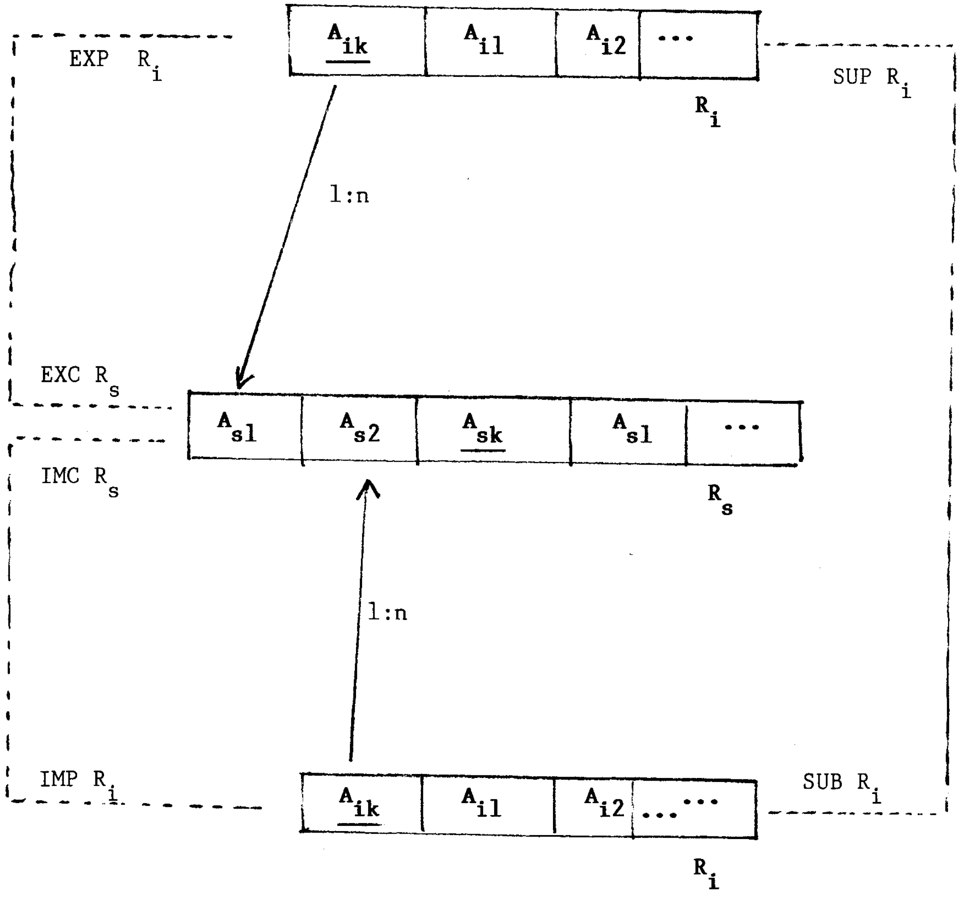

Figure 2. Modes of association for first level 1:n associations at left, and for first level n:m association at right

all levels, of the explosion/implosion hierarchy are involved, and (c) where the complete explosions and implosions are involved. Different modes of association are required for these separate cases, and we begin with the simplest case (a).

For the primitive primary (1:n) association between R_i and R_s supported by the attribute pair (A_{ik}, A_{s1}), we have the mode generators:

1a $\forall r_i \forall r_s[(r_i.A_{ik} = r_s.A_{s1}) \iff W_f(r_i.A_{ik}, r_s.A_{sk})]$

1b $\forall r_i \forall r_s[(W_f(r_i.A_{ik}, r_s.A_{sk}) \iff EXP(r_i.A_{ik}, r_s.A_{sk})]$

1c $\forall r_i \forall r_s[(W_f(r_i.A_{ik}, r_s.A_{sk}) \iff EXC(r_s.A_{sk}, r_i.A_{ik})]$

For the primitive primary (1:n) association between R_i and R_s supported by the attribute pair (A_{ik}, A_{s2}), we have the mode generators:

2a $\forall r_i \forall r_s[(r_i.A_{ik} = r_s.A_{s2}) \iff W_r(r_i.A_{ik}, r_s.A_{sk})]$

2b $\forall r_i \forall r_s[W_r(r_i.A_{ik}, r_s.A_{sk}) \iff IMP(r_i.A_{ik}, r_s.A_{sk})]$

2c $\forall r_i \forall r_s[W_r(r_i.A_{ik}, r_s.A_{sk}) \iff IMC(r_s.A_{sk}, r_s.A_{ik})]$

In the above, r_s and r_i are tuple variables for the respective relations R_s and R_i. It may help understanding of these definitions if we use the version of the data base shown in Figure 2, where R_i is duplicated for convenience. By convention, we may assume that it is the pair (A_i, A_{s1}) that supports the 1:n association that give the first level of an explosion, so that the first level of entities (part types or companies) in the explosion hierarchy is down (in Figure 2) from the entity being exploded. This convention is embodied in the set of modes generated in 1a, 1b and 1c above. W_f is the coincidental mode, and we use the modes EXP (explosion parent) and EXC (explosion child) to describe the 1:n association between R_i and R_s based on the pair (A_i, A_{s1}).

Thus if company C1 (an A_i, and A_{s1} value) is acquiring company C3 (an A_i and A_{s2} value), in merger M4 (an A_{sk} value), then we would have the following symbolic and English language propositions:

EXP(C1, M4) Company C1 is the explosion parent of merger M4.

EXC(M4, C1) Merger M4 is the explosion child of company C1

In addition, EXC R_s tuples are those R_s tuples that are EXC associated with a specific R_i tuple, and EXP R_i is that R_i tuple that is EXP associated with a specific R_s tuple.

Similarily IMP (implosion parent) and IMC (implosion child) are the modes for the 1:n association supported by the attribute pair (A_i, A_{s2}) as shown at bottom in Figure 2. Thus we have the following propositions for this association:

IMP(C3, M4) Company C3 is the implosion parent of merger M4.

IMC(M4, C3) Merger M4 is the implosion child of company C3.

IMC R_s tuples are also those R_s tuples that are IMC associated with a specific R_i tuple, and IMP R_i is the R_i tuple that is IMP associated with a specific R_s tuple.

Note that typically the non key and non foreign key fields of R_s (for example, A_{s3}, A_{s4}, ...) will hold information about the actual merger, in the case of the corporate family tree, or about the geometrical details of containing in the case of the bill of materials.

For example, suppose that R_i and R_s respectively describe companies and mergers, and that A_{s3} gives the percentage of shares that the acquiring company gains in the acquired company as a result of the merger. We might want the following retrieval carried out:

> Find the identifier for each firm where all but 2 immediate subsidiaries are more than 95% owned.

The SQL/N expression is:

SELECT A_{ik} FROM R_i

WHERE FOR ALL BUT 2 EXC R_s TUPLES ($A_{s3} > 95$)

We leave the construction of the corresponding SQL expression to the reader as an exercise. It is quite involved. The above expression uses the natural quantifier FOR ALL BUT 2.

As an example involving the implosion at just the first level of the implosion hierarchy, take the retrieval:

Find the identifier for each firm where one and all immediate parents each hold less than 5% of the stock.

The SQL/N expression is:

SELECT A_{ik} FROM R_i

WHERE FOR ONE AND ALL IMC R_s TUPLES ($A_{s3} < 5$)

Finally, to demonstrate the use of the mode EXP (IMP is used similarily), take the retrieval:

Find the merger identifiers for each merger in which the acquiring firm had assets of more than 10 million.

We assume that A_{i1} is the atribute of R_i that holds asset values. The SQL/N expression is:

SELECT A_{sk} FROM R_s

WHERE FOR EXACTLY ONE EXP R_i TUPLE ($A_{i1} > 10$)

We can now turn to n:m association between R_i and itself. We can use the coincidental modes W_f and W_r defined above (generator sets 1 and 2), and define a set of modes for the association (referring to Figure 2) between R_i at the top, and R_i at the bottom. It should be remembered that the association is non primitive and depends on the two primitive (1:n) associations between R_i and R_s, for which W_f and W_r are the coincidental modes.

We need distinct tuple variables for R_i. It is convenient to imagine rx for R_i at the top in Figure 2, and ry for R_i at the bottom.

3a. $\forall rx \forall ry \forall r_s [W_f(rx.A_{ik}, r_s.A_{sk}) \wedge W_r(ry.A_{ik}, r_s.A_{sk}) \Leftrightarrow W_R(rx.A_{ik}, ry.A_{ik})]$

3b $\forall rx \forall ry [W_R(rx.A_{ik}, ry.A_{ik}) \Leftrightarrow SUB(ry.A_{ik}, ry.A_{ik})]$

3c $\forall rx \forall ry[W_R(rx.A_{ik}, ry.A_{ik}) \Leftrightarrow SUP(rx.A_{ik}, ry.A_{ik})]$

If we are dealing with a bill of materials, and part type P1 immediately contains part type P5, then we could have the following symbolic and English-language propositions:

SUB(P5, P1) Part type P5 is immediately subsidiary to or contained in part type P1.

SUP(P1, P5) Part type P1 is immediately superior to or contains part type P5.

Furthermore SUB R_i tuples will describe the part types immediately contained by a given part type, and SUP R_i tuples the part types that contain a given part type.

Let us continue to suppose that the data base in Figure 2 respresents a bill of materials, and that attribute A_{i2} gives the length of a part. If we now have the retrieval:

> Find the part number for each part type that immediately contains not more than 6 12-inch long part types.

The SQL/N expression would be

SELECT A_{ik} FROM R_i
WHERE FOR AT MOST 6 SUB R_i TUPLES (A_{i2} = 12)

Here FOR AT MOST 6 is a natural quantifier. The corresponding SQL expression is illustrative of the semantics predefined in the mode SUB:

SELECT A_{ik} FROM R_i, RX
WHERE 6 <= (SELECT COUNT (*) FROM R_i, RX
        WHERE A_{i2} = 12
        AND A_{ik} IN (SELECT A_{s2} FROM R_s
                WHERE A_{s1} = RX.A_{ik}))

Here we must spell out the nature of the n:m association in terms of the 1:n association between R_i (top in Figure 2) and R_s, and the 1:n association between R_i (bottom in Figure 2) and R_s. With the SQL/N expression, this is done in the schema declared specification for the coincidental mode W_R (in mode set 3 above), on which the mode SUB is based. In a way, a mode of association can be looked upon as being the non procedural equivalent of a function in a procedural language. It does nothing that cannot be done without it, but once constructed and understood, saves a lot of programming or specification effort.

As an illustration of the use of the other mode SUP for the n:m association, we could have the retrieval:

> Find the part number of each part type that is immediately contained in exactly 4 part types that are 12 inches long.

Remembering that the mode SUB is used with explosions, and the mode SUP with implosions, the SQL/N expression is:

SELECT A_{ik} FROM R_i

WHERE FOR 4 SUP R_i (A_{i2} = 12)

In any explosion or implosion the hierarchy involved contains multiple levels. With an explosion, we go from R_i to R_s and back to R_i to obtain the entities at the first level, as is illustrated above. To get the entities at the next level, we repeat this procedure, and so on. The same is true for implosions, except that we use the converse set of 1:n associations.

If desired, we can use the modes SUB and SUP with any specified number of levels, since SQL/N permits nesting after the WHERE clause. However, care must be taken with the logic, since, unlike the case with the basic quantifiers, natural quantifiers do not so readily "flow through" from one level to another. To see this, suppose that we have the retrieval:

> Obtain the part number for each part type for which at least 1 part type two levels down in the explosion is 12 inches long.

The SQL/N expression is:

SELECT A_{ik} FROM R_i
WHERE FOR AT LEAST 1 SUB R_i TUPLE
    (FOR AT LEAST 1 SUB R_i TUPLE (A_{i2} = 12))

By repeated nesting in the manner shown in the expression, we can express retrieval of entities dependent on properties multiple levels down in the explosion (or up in the implosion). We used the basic quantifier FOR AT LEAST 1 (existential quantifier) in the above expression, since at least 1 entity two levels down implies at least 1 entity 1 level down for which there is at least 1 (SUB) associated entity a further level down, thus permitting the neat nesting in the expression. The same expression structure could have been used had the basic (universal) quantifier FOR ALL been applied to the entities two levels down. But if we use any of the natural quantifiers this implication will not hold. For example, at least 4 part types two levels down 12 inches long does not mean at least 4 part types one level down that contain these part types. Let us suppose for the moment that SUB(2) is the mode of assocaition between R_i at the top in Figure 2, and R_i two levels down. In that case, the above expression could be rewritten:

SELECT A_{ik} FROM R_i
WHERE FOR AT LEAST 1 SUB(2) R_i TUPLES (A_{i2} = 12)

Any of the natural quantifiers could meaningfully replace the existential quantifier in the above SQL/N expression, something that would not be meaningful in most cases with the former expression using nesting and the (1-level) mode SUB. We have yet to examine the basis for (multiple-level) modes of association, like SUB(2), that permit quntification of associated entities, more than one level apart in the hierachy for an explosion or implosion.

Modes of association for multiple explosion/implosion levels

We shall initially restrict discussion to the case of 2 hierarchical levels past level 0, since the principles involved are the same for n levels. The various modes that can be defined are illustrated in Figure 3. For convenience, we assume that in an explosion, we are traversing the relation replicas in a downward direction, and the converse for an implosion. In general any entity that can be exploded can be imploded. Hence the structure of the diagram.

We shall show how some of these modes are defined, as this should be sufficient

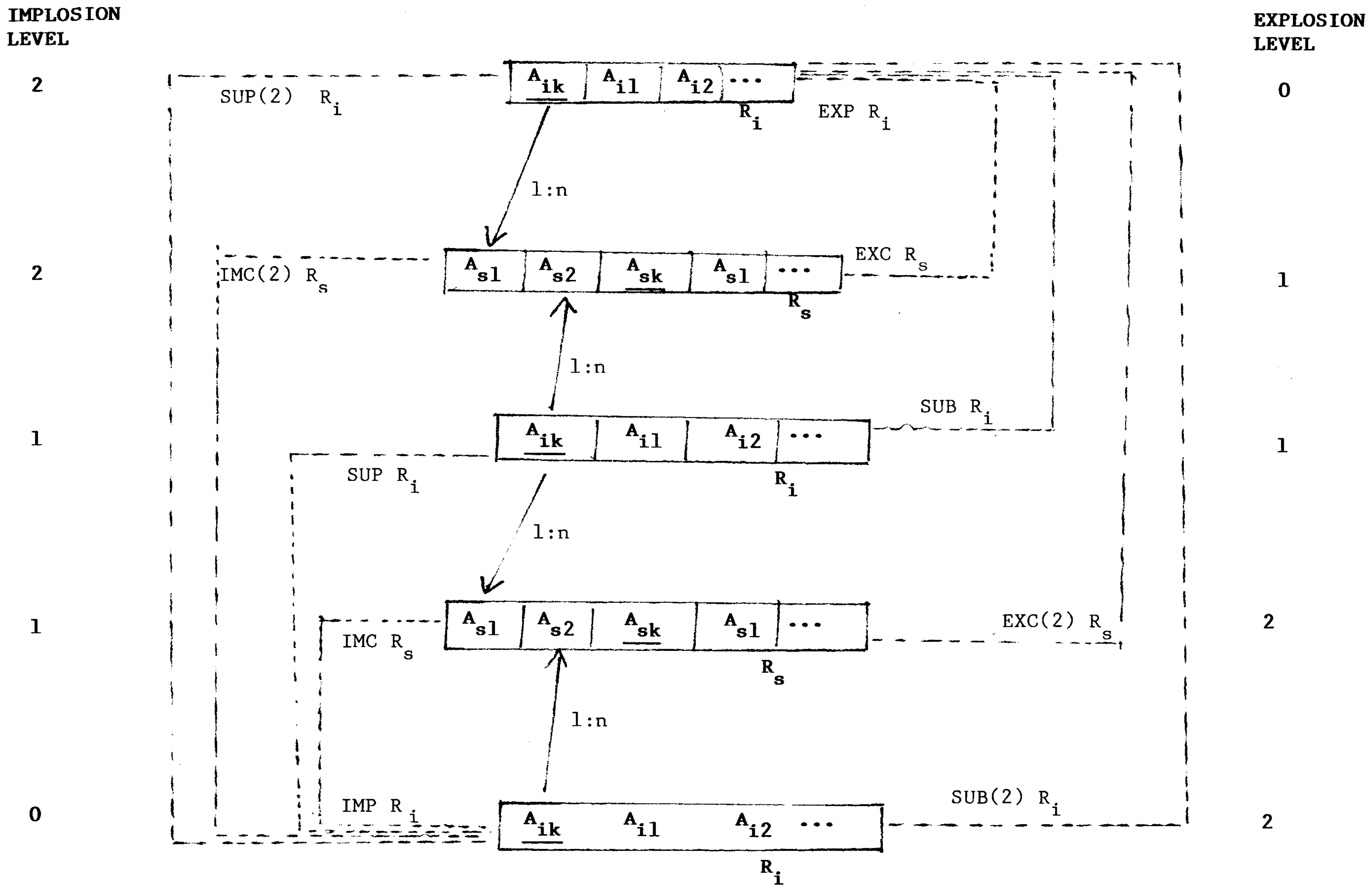

Figure 3. Association modes for both 1:n and n:m associations for 1 and 2 explosion/implosion levels

for understanding how to construct definitions for the others. In the previous section we have already defined the 1-level modes EXP, EXC, IMC, and IMP for the case of the 1:n associations btween R_i and R_s, as well as the modes SUP and SUB for the case of the n:m association between R_i and R_i.

The mode EXC(2) deals with an entity in R_i and the details of containment or merger 2 levels down. Thus if corporation C2 owns C12 and C14, and C12 has acquired C130 in merger M15, then the proposition EXC(2)(M15, C2) could represent the English langauge proposition:

Merger M15 is the explosion child, two levels down, of company C2.

Also EXC(2) R_s entities are those mergers where the subsidiaries of a given company have acquired further companies.

EXC(2) would be defined by the mode generator:

$$4.\quad \forall xr_i \forall yr_i \forall zr_s [W_R(XR_i.A_{ik}, YR_i.A_{ik}) \Leftrightarrow W_f(YR_i.A_{ik}, ZR_s.A_{sk}) = \mathrm{EXC}(2)(XR_i.A_{ik}, ZR_s.A_{sk})$$

The association described by the mode EXC(2) is clearly a non primitive one, since it depends the association chain from XR_i through YR_s through YR_i through ZR_s. This is illustrated in Figure 3.

Turning to the mode SUB(2), and using the example above, we could say that SUB(2)(C130, C2) denotes the proposition:

Company C130 is the subsidiary, two levels down, of the company C2.

Furthermore SUB(2) R_i are the subsidiaries, two levels down, of a given company.

SUB(2) is defined by the mode generator:

$$5.\quad \forall xr_i \forall yr_i \forall zr_s [W_R(XR_i.A_{ik}, YR_i.A_{ik}) \Leftrightarrow W_R(YR_i.A_{ik}, ZR_i.A_{ik}) = \mathrm{SUB}(2)(XR_i.A_{ik}, ZR_i.A_{ik})]$$

The modes EXP(2), IMP(2), IMC(2) can be defined with a mode generator similar to generator 4, and mode SUP(2) can be defined with a generator similar to generator 5. These modes are also illustrated in Figure 2.

As to the utility of these modes of association, let us suppose that R_i describes companies and R_s mergers, with A_{ik} as the company identifier, A_{s1} the identifier of a company that takes over a company identified by A_{s2} in a merger identified by A_{sk}, with A_{s3} giving the percent of shares held by the acquisitor following the acquisition.

Suppose that we have the retrieval request:

> Find the companies, where all but 2 immediate subsidiaries of immediate subsidiaries are 95% owned.

The SQL/N expression is:

SELECT A_{ik} FROM R_i
WHERE FOR ALL BUT 2 SUB(2) R_i TUPLES (A_{s3} = 95)

We can define modes SUP(n), SUB(n), EXC(n), EXP(n), IMC(n), IMP(n) where n is any positive integer, although for any mode M(1), we can, by default, use the name M. These modes are for handling entities n levels down or up in the explosion/implosion hierarchy. The modes would be generated using mode generators similar to generators 4 and 5, except that the chain of coincidental modes used at the left in the generator will contain a number of coincidental modes equal to n. We may refer to n as the level of the mode.

Suppose now that we want the explosion for part type P2, going 3 levels into the explosion hierarchy, assuming that R_i describes part types. Thus we want A_{ik} values from each duplication of R_i (see Figure 3, although it deals with only 2 levels), going down three levels, where the A_{ik} values retrieved are all part types contained geometrically by A_{ik}, where A_{ik} = 'P2' in the R_i duplication at level 0.

The SQL/N expression is:

SELECT A_{ik} FROM EACH R_i TUPLE,
A_{ik} FROM EACH SUB(1) R_i TUPLE,

A_{ik} FROM EACH SUB(2) R_i TUPLE,
A_{ik} FROM EACH SUB(3) R_i TUPLE
WHERE $R_i.A_{ik}$ = 'P2'

The explosion for P2, going 3 levels up, would be the same as the above expression with SUB changed to SUP.

In the above expression, if in the explosion hierarchy, P2 contained B1, C1 and B1 part types (in different geometrical positions), and if in turn B1 contained D11 and D13, and C1 contained E21 and E22, then the retrieved data would form the relation:

P2	B1	D11	...
P2	B1	D13	...
P2	C1	E21	...
P2	C1	E22	...
P2	B1	D11	...
P2	B1	D13	...

Further peripheral processing would be required if the data were to be displayed as a data tree.

A more detailed explosion (implosion) would give details of the geometrical aspects of containment in the case of part types, or legal ownership in the case of corporations, and would require attribute values from R_s (duplications) images as well. This would be expressed by the SQL/N expression:

SELECT A_{ik} FROM EACH R_i TUPLE,
A_{s2}, A_{s3} FROM EACH EXC(1) R_s TUPLE,
A_{s2}, A_{s3} FROM EACH EXC(2) R_s TUPLE,
A_{s2}, A_{s3} FROM EACH EXC(2) R_s TUPLE
WHERE $R_i.A_{ik}$ = 'P2'

Remember that A_{s2} gives a part type contained, and A_{s3} gives details of geometrical placement of A_{s2} within the containing part type. The repetitive parts of the above expressions can be eliminated by means of the SQL/N provision for repetition:

SELECT A_{ik} FROM EACH R_i TUPLE,

A_{s2}, A_{s3} FROM EACH EXC(N) R_s TUPLE FOR N = 1 UNTIL 3

WHERE $R_i.A_{ik}$ = 'P2'

Corresponding SQL expressions are difficult, and we leave them to the reader to ponder. It might be mentioned, that given published SQL syntax and semantics, there is doubt as to whether SQL expressions are possible.

Terminal explosions and implosions

In all cases so far, we have dealt with a specific number of levels in an explosion or implosion. However, when we want the complete or terminal explosion (or implosion) for an entity, we want retrieval of the entire hierarchy for the explosion (or implosion), even though the number of levels will not be known beforehand, since the number of levels in an explosion (or implosion) depends on the individual entity being exploded (or imploded).

For example, suppose that we want the explosion for entity (company) C4, where we do not know beforehand how many levels are in the explosion hierarchy. We could use the SQL/N expression:

SELECT A_{ik} FROM [EACH] R_i [TUPLE],

A_{ik} FROM [EACH] SUB(N) R_i [TUPLE] FOR N = 1 UNTIL END [OF EXPLOSION]

WHERE $R_i.A_{ik}$ = 'C4'

Where the implosion is needed SUP(N) will replace SUB(N). Items in parenthesis are to aid readability only and may be omitted. When full details of legal ownership (or geometry of containment) are needed, the SQL/N expression for the implosion if C4 will be:

SELECT A_{ik} FROM [EACH] R_i [TUPLE],

A_{s1}, A_{s3} FROM [EACH] EXP(N) R_s [TUPLE] FOR N = 1 UNTIL END [OF IMPLOSION]

WHERE $R_i.A_{ik}$ = 'C4'

Expressions involving complete explosions and implosions cannot be constructed in conventional SQL. The expressions above are for the simplest possible explosion or implosion retrievals. Much more complex retrievals involve specified conditions for the entities at different levels of the explosion. SQL/N permits expression of such retrievals with a minimum of syntax.In addition, the language permits expression of fine shades of meaning that cannot easily be expressed even in English.

Conclusions

The concept of a mode of association can be applied to the case of non primitive recursive associations (many-to-many-to-many ..., and one-to-many-to many ... and ... to-many to-many-to-one) that occur in data bases bill-of-materials and similar data bases. These modes of association an be used to specify entities at different levels in implosions and implosions, and when, in addition, natural quantifiers are applied to such specified entities, a highly flexible and effective method of expression results. This method of expression is used in SQL/N, the upward compatible extension to conventional SQL. As a result SQL/N can be used to express retrievals involving bills of materials or corporate family trees with only a few lines as compared with either many lines of SQL or the impossibility of an SQL expression.

References

1. Astrahan, M.M., et al. System R, relational approach to data base management. ACM Trans. on Database Sys., 1, 2, 1976, 297-314.

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

3. Bradley,J.,Introduction to Database Management in Business, Holt, Rinehart & Winston, New York, 1983.

4. Bradley, J., SQL/N and attribute/relation associations implicit in functional dependencies, J. Int. Comp. & Info. Science, 12(2), 1983

5. Bradley, J., SQL/N and modes of association in relational data bases Research Report No. 84,145/3, University of Calgary, Computer Science Dept., Calgary Alberta, Canada.

6. Bradley, J. Guide to Investment Data Bases, Burgess Communications, Mineapolis, Min., 1985.

7.Chamberlin, D.D.,et al, SEQUEL2, A unified approach to data definition, manipulation and control, IBM J. Res. & Dev., Nov. 1976, 560-575.

8. Chamberlin, D.D.et al. Support for repetitive transactions and ad hoc queries in System R. ACM Trans.on Database Sys., 6, 1, 79-94, 1981.

9. P. P. S. Chen. The entity-relationship model - towards a unified view of data. ACM Trans. on Database Sys., 1, 1, 1976, 9-36.

10. Codd, E. F. Relational completeness of data base sublanguages.In "Data base systems", Courant Computer Science Symposia, Vol.6, Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

11. Codd, E. F., Extending the relational data base model to capture more meaning, ACM trans. on Database Sys., 4, 4, 1979, 397-434.

12. Codd, E. F. Relational database: A practical design for productivity, CACM, 25, 2, 1982, 109-117.

13. Introduction to Data Base Systems, Addison-Wesley, Reading, Mass., 1981.

14. Introduction to DB2, Addison-Wesley, Reading, Mass., 1984.

15. Hilbert, D., and Ackerman, W., Principles of Mathematical Logic, Chelsea Publishing Co., New York, 1950 (Translation of: Grundzuge der Theoretischen Logik, 2nd ed., Springer, Berlin, 1938.

16. Kim, W.,On oprimizing an SQL-like nested query, ACM Trans. on Database Sys., 7, 3, 1982, 363-372.

17. Ullman, J. D. Principles of Database Systems, Computer Science Press, Rockville, Md., 1982.

18. Weiderhold, G., Database Design, 2nd ed., McGraw-Hill, New york, 1983.

19. Welty, C., and D. W. Stemple Human factors comparison of procedural and non procedural query languages. ACM Trans. on Database Sys., 6, 4, 1981, 626-649.

20. Wilmot, R. B., Foreign keys decrease adaptibility of database designs, CACM, 27, 12, 1984, 1237-49.