in: Proc. 3rd Int. Conf. on Foundations of Data Organization and Algorithms (FODO), Paris, 1989, pp. 31-46

Cluster Mechanisms Supporting the Dynamic Construction of Complex Objects

H. Schöning A. Sikeler

University Kaiserslautern, Department of Computer Science, D-6750 Kaiserslautern, West Germany

Abstract

Non-standard database applications require adequate modeling facilities for their application objects which in general have an internal structure to be maintained by the database system. For this purpose, the database system has to provide fast access to such an object as a whole as well as to its components. In systems which support complex objects with a statically established structure, clustering of the objects' components along this structure is a widespread means to enhance efficiency. Systems which support the dynamic definition of complex objects' structures, however, cannot predict the characteristics of accesses to the database, and therefore have more problems in finding a storage structure that is useful for at least the majority of the accesses. In this paper, we propose a cluster mechanism that supports the flexibility and dynamism of the molecule-atom data model at the efficiency of static structure clustering. We discuss different alternatives for its design, taking into account the query processing strategies of the underlying database system. We address some problems concerning optimization that emerge from the dynamic structure definition and show some possible solutions.

1. Introduction

Non-standard database applications such as 3D-modeling for workpieces or VLSI chip design [DD86] require for various reasons adequate modeling facilities for their application objects. The notion of complex objects [BB84] is used to indicate that such objects have an internal structure maintained by the database system and that access is provided to the object as a whole as well as to its components. Obviously, interactive applications such as CAD require reasonable response times, and therefore demand efficient access to the complex objects provided by the database system. In systems which support complex objects with a statically established structure (e.g. NF^2 [SS86]), clustering of the objects' components along this structure is a widespread means of enhancing efficiency [Da86, DPS86]. Systems which support the dynamic definition of complex objects' structures, however, cannot predict the characteristics of accesses to the database, and therefore have more problems in finding a storage structure that is useful for at least the majority of the accesses. In this paper, we study a cluster mechanism (the so-called atom-cluster type) that is designed to support a lot of different database requests rather than to materialize a single object structure or query result. It may be compared to a materialized view, a database snapshot or a cached procedural field.

The conventional way to process queries on a relational view is to use query modification which translates the corresponding queries into ones on the base relations [St75]. An alternative approach, however, is to materialize a view, which means that the resulting relation is actually stored [BLT86, Ha87, SI84]. As a consequence of updating to the base relations, the materialized views may also require changes. These changes may be either executed at the end of each transaction, i.e. the materialized view is always up-to-date, or deferred by updating a materialized view just before data is retrieved from it [Ha87].

Database snapshots [AL80, LHMPW86], which are a related mechanism, are periodically refreshed, read-only replicas of a selected portion of the database which is defined by a query on one or more base relations. Snapshots are especially interesting in database applications which require the freezing of the database state e.g. for analysis, planning or reporting or in a distributed database serving as cost effective substitute for replicated data. Snapshots are not guaranteed to reflect the actual state of the database.

Another direction in materializing query results investigates the support of database procedures [SAH87]. Queries are stored in so-called procedural fields, i.e. attributes, in the same way data is stored in a relation. Accessing such a procedural field implies the execution of the queries stored in this field. However, performing such an access will

be generally slow. Hence, the result of these queries may be cached, i.e. computed once and stored within a specifically assigned area on secondary storage, the cache [Se87].

Similar to these approaches, we investigate the physically clustered materialization of a complex object type. In the next chapters we introduce our data model and the architecture of our data base system. The dynamic construction of molecules during query processing is described in chapter 4. Based on this, we discuss restrictions on the complexity of molecules materialized as atom clusters in chapter 5. Finally we consider the use of atom-cluster types in the process of molecule construction (chapter 6).

2. The Molecule-Atom Data Model

The molecule-atom data model (MAD model [Mi88a]) has been introduced to support the use of dynamically defined complex objects. Complex object types (molecule types) are defined in terms of their components, which may be either complex object types or basic object types (atom types). An atom type consists of some attribute types, and therefore may be compared to a relation in the relational model. The corresponding objects, called atoms, are similar to tuples. We allow a richer selection of data types than most conventional database systems do. Particularly, the two special data types IDENTIFIER and REFERENCE are used to explicitly express relationships between atoms. The *IDENTIFIER* attribute, which is present in each atom exactly once, contains a system-defined primary key (surrogate) to uniquely identify the atom. *REFERENCE* attributes contain one or more IDENTIFIER values, all pointing to atoms of the same atom type (typed references). There must be a corresponding "back reference" for each REFERENCE attribute, i.e., if atom type A has a REFERENCE attribute pointing to atom type B (for short, "REFERENCE attribute to B"), then atom type B must contain a corresponding REFERENCE attribute to A. Also, if the REFERENCE attribute of atom a contains the IDENTIFIER of atom b, the corresponding REFERENCE attribute of atom b has to contain the IDENTIFIER of a. This structural integrity (which is based on referential integrity) is enforced by the operations of the MAD model. Thus, there is a means to reflect 1:1, 1:n, and n:m relationships among atoms in a direct and symmetric way. The relationships between atoms, which are manifested by the values of the REFERENCE attributes, lead to the so-called atom network. So far, the MAD model is similar to the entity-relationship model [Ch76].

The relationships installed by REFERENCE attributes can be used for the definition of molecule types. The notation A.ab-B.bc-C means, for example, that for each atom *a* of type A all atoms of type B ("B atoms") that are referenced by a's REFERENCE attribute *ab* and all C atoms referenced by attribute *bc* of these B atoms are grouped to a molecule. This definition assigns a direction to the relationships between A and B, and B and C respectively. Hence, a molecule can be seen as a directed subgraph of the atom network having one root, the so-called *root atom* (in contrast to the *component atoms*). If A has only one REFERENCE attribute to B, A-B may be written instead of A.ab-B. Fig. 2.1 illustrates some molecule types and corresponding molecules for a sample database.

Besides hierarchical structures as introduced above, the MAD model also allows network-like and recursive molecule type definitions. When an atom type has more than one predecessor in the molecule type graph, an atom of this type only belongs to a corresponding molecule, if it has references to at least one atom of each of the predecessor types which also has to belong to a corresponding molecule (network-like semantic). Recursive molecule types repeat a component molecule type in several recursion levels. Recursion level 0 of a recursive molecule consists of a molecule of the component molecule type. All molecules of the component molecule type which are referenced by the recursion-defining REFERENCE attribute of a component molecule on level i form the level i+1 of recursion. If a component molecule appears in more than one recursion level, it only belongs to the lowest one. Thus, cycles in the recursive molecule are avoided, and the termination of this operator computing the transitive closure is guaranteed.

Sets of molecules can be inserted, deleted, updated or retrieved (selected) using the SQL-like **m**olecule **q**uery **l**anguage (MQL). We concentrate on the SELECT-statement, which is the most complex statement. It is used to extract a set of molecules of a certain type from the database. Its general form is

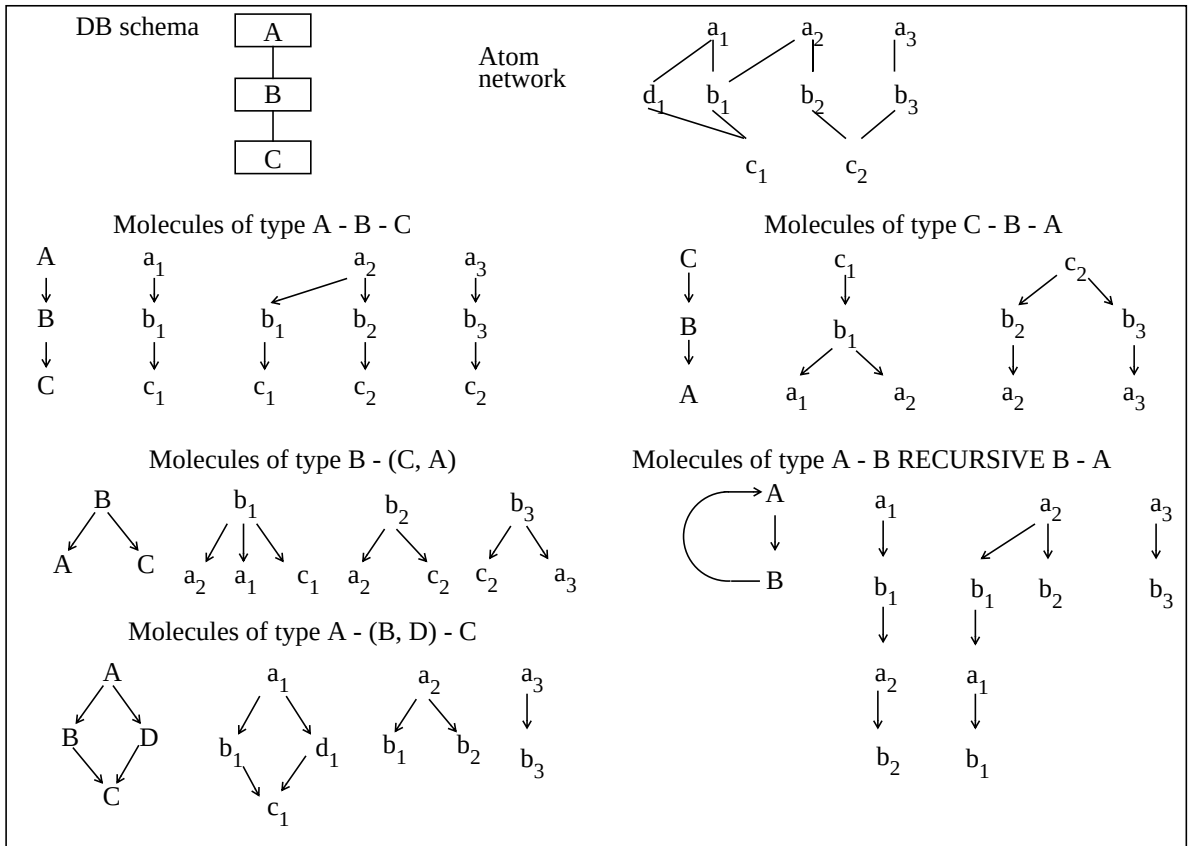

Figure 2.1: Example of molecules for various molecule type definitions

```
SELECT    <projection_clause>
FROM      <molecule type definition clause>
WHERE     <restriction clause>
```

The **molecule type definition clause** determines the environment (molecule type) to work on. If more than one molecule type is present in this clause, this defines a cartesian product of the molecule types.

The **restriction clause** contains a condition ranging over the environment molecule type. It may be of arbitrary complexity, and may contain quantifiers and SELECT-statements (nested sub-queries). Only those molecules which fulfil this condition (molecules that *qualify*) are members of the result set. If no restrictions are to be imposed, "WHERE <restriction clause>" may be omitted.

The **projection clause** specifies, which parts of the molecules are to belong to the result. According to this clause, atoms or attributes are removed from the qualifying molecules. A value-dependent projection is possible ("qualified projection") and can be specified using a SELECT-statement with the special molecule type definition clause "RESULT".

The resulting molecules are molecules of the type specified in the molecule type definition clause, which qualify under the conditions of the restriction clause and have undergone the projection specified in the projection clause.

The following example shows the effects of qualified projection:

```
SELECT    C,(  SELECT    B
               FROM      RESULT
               WHERE     B.Att1=7)
FROM      C - B - A
WHERE     FOR ALL A: A.Att1>0
```

Assume that b_2 of the above example fulfils $B.Att_1=7$ and a_2 and a_3 qualify concerning $A.Att_1>0$. Then the result set consists of the molecule c_2-b_2.

This short introduction to MQL neglects many aspects of our query language as well as of our data model. Nevertheless, it will be sufficient to understand the considerations in the following chapters. More information about the MAD model and MQL can be found in [Mi88b].

3. The PRIMA Architecture

So far, we have outlined the main features of the MAD model concerning the definition and manipulation of complex structured objects, i.e. molecules. In the following, we present a short overview of the concepts and ideas used for its implementation within the PRIMA system [HMMS87, Hä88]. Our implementation model for PRIMA (Fig. 3.1) distinguishes three different layers for mapping molecules which are visible at the MAD interface onto blocks stored on external devices:

- The main task of the *data system* [Sch88] is to transform the molecule-set-oriented MQL interface into lower level programs as well as their subsequent execution. This is done by first transforming the user-submitted MQL statements into valid, semantically equivalent, but not necessarily optimal query evaluation plans (compilation phase). In an optimization phase, these query evaluation plans (QEPs) are rearranged according to different heuristics in order to speed up their processing. Subsequently, these QEPs are evaluated yielding the desired result (execution phase). A detailed description of a QEP as well as of its evaluation during the execution phase follows in chapter 4.
- The *access system* [Si88a] provides an atom-oriented interface similar to the tuple-oriented interface of the Research Storage System (RSS) of System R [As81]. However, the access system is more powerful than RSS as outlined in the following. It allows for direct access and manipulation of a single atom as well as navigational atom-by-atom access to either homogeneous or heterogeneous atom sets. Manipulation operations (insert, modify, and delete) and direct access (retrieve) operate on single atoms identified by their logical address (or surrogate) which is used to implement the IDENTIFIER attribute as well as the REFERENCE attributes. In performing manipulation operations, the access system is responsible for the automatic maintenance of the referential integrity defined by the REFERENCE attributes by adjusting the appropriate back references.

  Different kinds of scan operations are introduced as a concept to manage a dynamically defined set of atoms, to hold a current position in such a set, and to successively deliver single atoms. Some scan operations, however, are added in order to optimize retrieval access. Therefore, they may depend on the existence of a certain redundant storage structure. The *atom-type scan* delivers all atoms in a system-defined order based on the basic storage structure which always exists for each atom type. Similarly, the *sort scan* processes all atoms according to a specified sort criterion thereby utilizing the basic storage structure. However, since sorting an entire atom type is expensive and time consuming, a sort scan may be supported by an additional storage structure called sort order. Thus, a sort order consists of a homogeneous atom set materializing a sort operator. The *access-path scan* provides an appropriate means for fast value-dependent access based on different access path structures such as B-trees, grid files, and R-trees. The *atom-cluster type scan* as well as the *atom cluster scan* speed up the construction of frequently used molecules by allocating all atoms of a corresponding molecule in physical contiguity

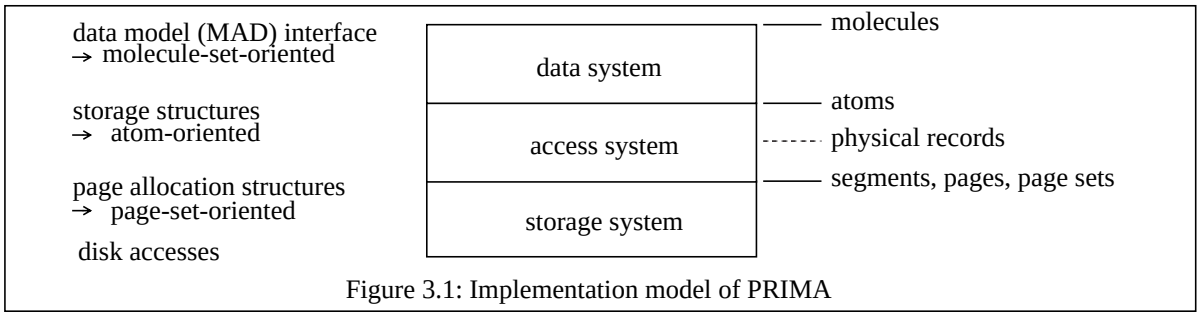

Figure 3.1: Implementation model of PRIMA

using a tailored storage structure called atom-cluster type. These atom clusters are the main subject of the rest of the paper.

The underlying concept is to make storage redundancy available outside the access system by offering appropriate retrieval operations (i.e. the choice of several different scans for a particular access decision by the optimizer of the data system), whereas in the case of update operations storage redundancy has to be concealed by the access system. As a consequence, physical records are introduced as byte strings of variable length which are stored consecutively in the containers offered by the storage system. Depending on the underlying storage structure, a physical record corresponds to either an entire atom (basic storage structure, sort order) or to a set of heterogeneous atoms (atom-cluster type). This establishes an n:m relationship among atoms and physical records which has to be maintained by a sophisticated address structure assigning each IDENTIFIER value a set of physical addresses and vice versa.

- The *storage system* [Si88b] as the lowest layer of PRIMA pursues two major tasks: It manages the database buffer and organizes the external storage devices, thus being responsible for the data exchange between main storage and disk storage. For this purpose, the database is divided into various segments consisting of a set of logically ordered pages. All pages of a segment are of equal size, and can be chosen for each segment independently to be 1/2, 1, 2, 4, or 8 kbytes, being kept fixed during the lifetime of a segment. Thus, the page size may be adapted to the specific access pattern of the segment in order to diminish either the conflict rate or the number of IO operations. The five page sizes, however, are not sufficient when considering the mapping process performed by the access system. Therefore, *page sequences* are introduced as predefined page sets supported by physical clustering. A page sequence is a set of logical consecutive pages of a segment which contain (from the viewpoint of the access system) one single object spanning these pages [DPS86]. Additionally, the storage system provides the means to handle not only such predefined page sets but also arbitrary page sets.

This overview may serve as a basis for the detailed discussion of the dynamic construction of molecules supported by an appropriate storage structure, i.e. the atom-cluster type, throughout the rest of the paper.

4. Query Processing in the Data System

When an MQL statement is given to the data system, it is compiled into an equivalent QEP forming a directed tree. The vertices of this tree are labeled with operators, while its arcs correspond to the data flow among the operators. The leaves of the QEP represent the operator “construction of simple molecules” (CSM), which builds up hierarchical, non-recursive molecules fulfilling some qualifications by using access system calls. The other vertices stand for operators like *recursion*, *aggregation*, etc.

The QEP may be subject to transformations by the optimizer, which is expected to generate a more efficient QEP by choosing appropriate access paths, selecting specific methods for each operator, and so on. Some of the possible choices for the CSM operator are discussed later.

When a QEP is executed, the operations indicated by its leaves are involved first, i.e., CSM operations are started. Whenever a molecule has been constructed, it is handed in a pipelined way to the next operator as indicated by the arcs in the QEP. The root operator of the QEP produces the final result set [HSS88].

Here we concentrate on the description of CSM, which constructs a set of molecules represented by

```
SELECT   P
FROM     M
WHERE    Q(M).
```

M is the definition of a non-recursive hierarchical molecule type, P is a coherent subgraph of M, containing the root atom type of M, and Q(M) is a restriction on M, which can be determined by the consideration of a single molecule. As a consequence, Q(M) is not allowed to contain any queries.

Conceptually, CSM scans the root atom type of M. For each atom of this type CSM fetches the successor atoms according to M, then their successor atoms, and so on. When the whole molecule is fetched, it is checked against Q(M). If it qualifies, the molecule is added to the result set, otherwise all atoms belonging to the molecule are erased.

Considering this conceptual method reveals two points of inefficiency:

- All atoms of the molecule are fetched before Q(M) is tested, even those, which are not needed to check the molecule's qualification. If the molecule is discarded, the accesses to these atoms were senseless.
- Atoms shared by multiple molecules are fetched many times.

To avoid the first problem, the computation of the corresponding result set is done in two phases. The first one checks for molecule qualification accessing only atoms needed to decide Q(M). As soon as Q(M) cannot be fulfilled any longer, the next molecule is considered. If all qualifying molecules have been found, their remaining atoms are fetched in the second phase. In order to avoid multiple accesses to one atom, atoms that are likely to be needed by another molecule are stored in a main memory atom buffer.

The two phases may be interleaved, i.e., when the first qualifying molecule has been determined, the second phase may be started for this molecule immediately. While the second phase can be implemented straightforwardly (following the values of the corresponding REFERENCE attributes), the first phase offers a great choice of possible strategies. Depending on the restriction Q(M), the optimizer chooses an atom type to be fetched first and an appropriate access path, if available. Further atoms are accessed in a sequence which is likely to show disqualification as soon as possible (e.g., try to evaluate very selective parts of Q(M) first). This procedure is illustrated by the following example:

```
SELECT   ALL
FROM     A - B - C
WHERE    EXISTS B: A.Att1=7 AND B.Att1=A.Att1
```

First, A atoms fulfilling $A.Att_1=7$ are fetched (if possible, via an existing access path). For all these atoms, the corresponding B atoms have to be fetched using the REFERENCE attribute values of the A atoms, and tested for the condition $B.Att_1=A.Att_1$. If one of them fulfils this condition, the corresponding A atom is the root of a qualifying molecule, and the second phase may be started for this molecule.

In some cases there are more efficient ways to compute the result, than to navigate along REFERENCE attribute values. By comparing estimated selectivity and access costs the optimizer chooses one specific access sequence and determines which atoms will be stored in the atom buffer.

5. Materializing Atom Sets

Constructing molecules out of single atoms will generally be very slow. Since all atoms are distributed amongst different segments and pages, in the worst case one page request (i.e. an IO operation), has to be initialized for each atom. In order to avoid this extreme overhead a special storage structure is required which allows for the physical clustering of an atom set within a page sequence. Thus, all atoms belonging to such an atom set may be read into the database buffer by a single storage system call which in turn utilizes chained IO in order to minimize disk access time. In this chapter we discuss some alternatives in designing such a storage structure called atom-cluster type and

its mapping onto the “containers” offered by the storage system. In particular, we would like to concentrate on the following problems:

- Which atom sets are to be materialized and how long?
- In which way is a materialized atom set to be updated due to a modification of the database?
- How are atom clusters utilized in constructing molecules?

In this chapter, we discuss the first two problems, whereas the third problem is treated in chapter 6.

5.1 Different Kinds of Atom-Cluster Types

The choice as to when to materialize molecules and when to dynamically construct them time after time is highly application-dependent (frequency of accesses, frequency of updates, etc. [Ha87]). Therefore, we delegate this choice to an experienced database administrator who may define and release an atom-cluster type representing a materialized molecule type by a corresponding statement, although we currently investigate, how this choice may be supported or even automated by the system. Nevertheless, we have to define how an atom-cluster type may look like compared to the resulting molecule set of an MQL query, especially when considering that an atom-cluster type has to be maintained by the access system due to modifications of the database. Therefore, we want to investigate each of the three clauses of a general SELECT statement with respect to the effects of a modification operation on a single atom on molecules which are materialized in the appropriate form.

Projection Clause

Concerning the projection clause, the most interesting part is the qualified projection, which allows for a value-dependent selection of atoms within a single resulting molecule. For this purpose, a qualification criterion is specified which is evaluated for each environment molecule defined by the molecule type definition clause (and selected by the restriction clause). Modifying an atom belonging to such an environment molecule may have the following effects on the corresponding result molecule:

- Due to the modification the qualification criterion of the qualified projection may become invalid for certain atoms. Thus, atoms that till now belonged to the result molecule have to be removed.
- On the other hand, the qualification criterion may become valid for certain other atoms. So, atoms that till now did not belong to the result molecule have to be included.

The complexity of the different possibilities may be demonstrated by a quite simple example. Suppose, we have a DB schema of three atom types A, B and C and a query defining a hierarchical molecule type A-B-C with a qualified projection selecting those submolecules B-C which exactly contain two atoms of type C (Fig. 5.1). For the given database this query results in a single molecule as indicated by the dashed circle. Inserting a new root atom generates a new molecule in either case (since no restriction clause is specified). The structure of the resulting molecule, however, depends on the qualified projection as is shown by inserting a2 with a reference to b1 and a3 with a reference to b2 (Fig. 5.1a). Inserting a new component atom, on the other hand, only causes an already existing molecule to grow, e.g. inserting c4 with a reference to b2 (Fig. 5.1b), or to shrink, e.g. inserting c4 with a reference to b1 (Fig. 5.1c). The latter holds for modifying an atom, e.g. removing the reference to b1 in a2, and for deleting a component atom, e.g. deleting c2, whereas deleting a root atom always results in the deletion of the whole molecule.

Therefore, the following actions are generally necessary in order to determine the effects of a modification operation on the appropriate molecules (without a restriction clause being specified):

- Inserting a root atom, generates a new molecule, which has to be constructed according to the molecule type definition clause. The participating atoms have to be selected based on the qualified projection.
- If a component atom is inserted or either atom is modified, all environment molecules containing the inserted or modified atom have to be constructed and the qualified projection has to be applied again to these molecules. The corresponding resulting molecules replace the original resulting molecules identified by the appropriate root atoms.

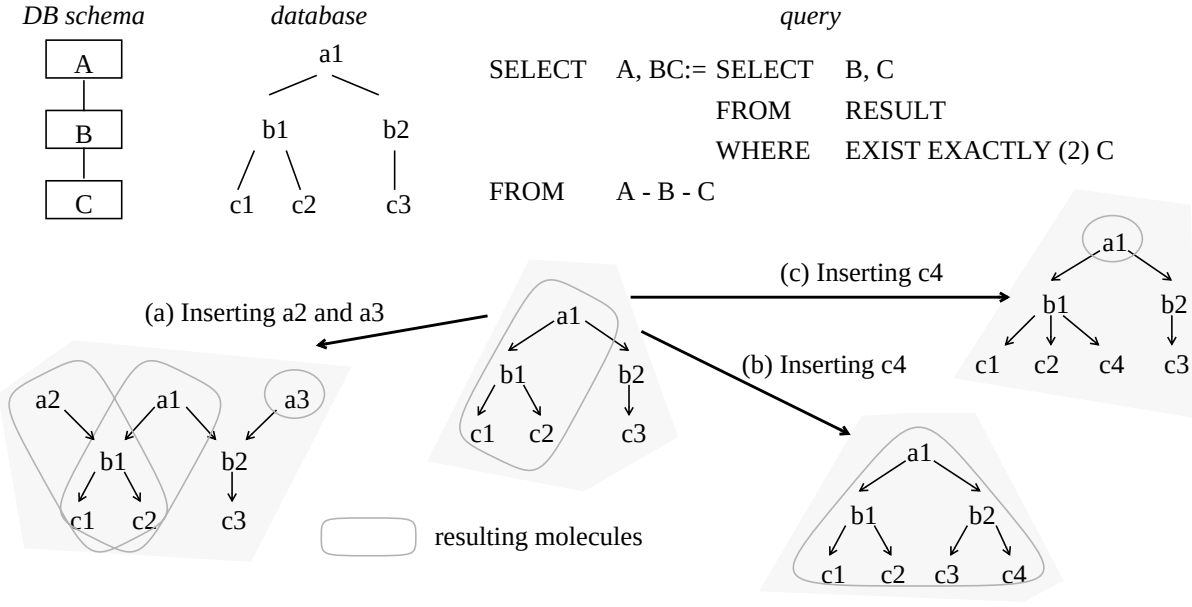

Figure 5.1: Effects caused by a qualified projection

- When a component atom is deleted, a similar process has to be performed. Initially, the environment molecules containing the atom to be deleted have to be determined. These molecules have to be reconstructed without the deleted atom, the qualified projection has to be applied and again the original resulting molecules have to be replaced by the new ones.
- Deleting a root atom, however, results in deleting the associated molecule.

Thus, molecules have to be constructed and possibly complex qualification criteria have to be evaluated in order to determine the effects of a simple modification operation concerning the appropriate molecules. This, however, is the task of the data system. As a consequence, the projection clause allowed in an atom-cluster type definition is restricted to the key word ALL:

```
SELECT      ALL
FROM        <molecule type definition clause>
WHERE       <restriction clause>
```

Restriction Clause

The effects of a restriction clause may be compared to those of a qualified projection. The sole difference is that the corresponding qualification criterion determines whether or not an environment molecule really belongs to the result set. Therefore, the following situations may be distinguished (Fig. 5.2):

- The modification operation on a single atom violates the qualification criterion for certain molecules (Fig. 5.2b, 5.2c and 5.2d). As a result, these molecules have to be removed.
- Due to the modification operation the qualification criterion becomes valid for certain molecules (Fig. 5.2a and 5.2c) which now have to be materialized.
- In spite of the modification operation the qualification criterion remains valid for some molecules (Fig. 5.2a). Thus, these molecules have to be updated in the appropriate way.

Therefore, the following actions have to be performed when initializing a modification operation on a single atom:

- When inserting an atom, first of all the environment molecules containing the inserted atom have to be constructed and the qualification criterion has to be evaluated for each of them in order to determine those resulting molecules which either have to be materialized from scratch or which replace the original resulting molecules. Furthermore, for each environment molecule which does not qualify it has to be checked whether or not a corresponding result molecule exists which then has to be removed.

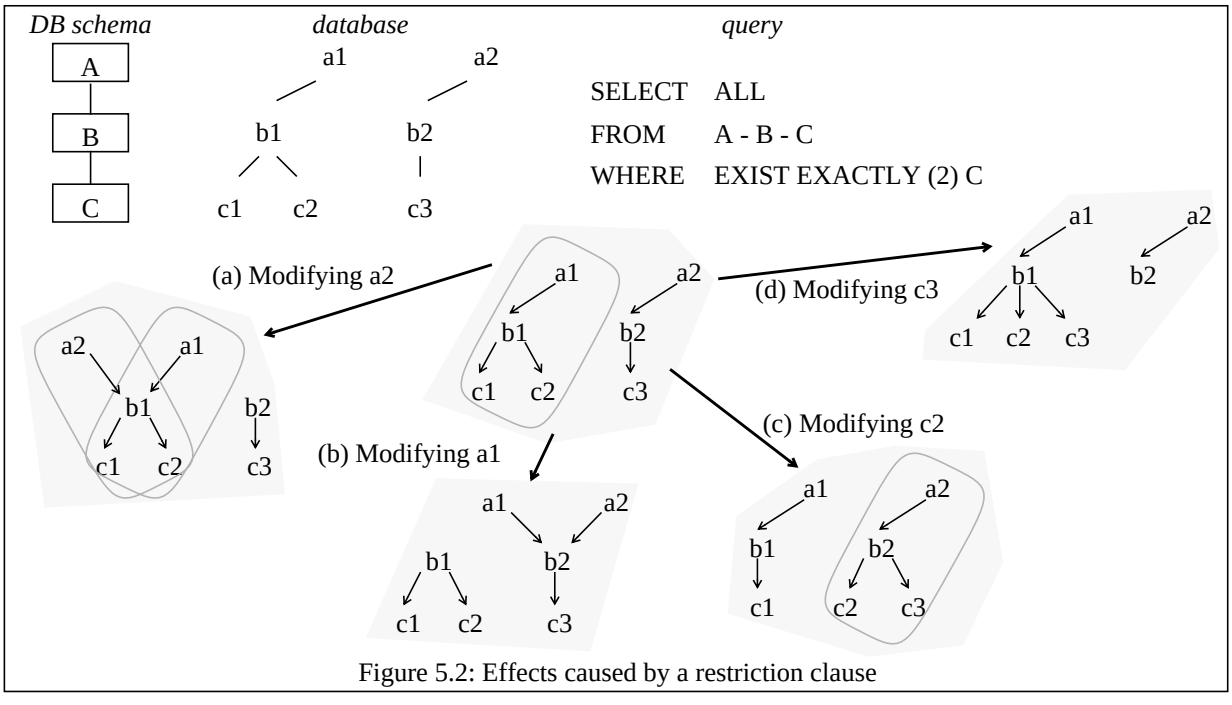

Figure 5.2: Effects caused by a restriction clause

- If an atom is modified, two steps are necessary. In a first step, the existing resulting molecules containing the atom to be modified have to be removed. In the second step, the same procedure as for an insertion is mandatory. The environment molecules containing the modified atom are constructed, the qualification criterion is evaluated, and molecules are either materialized anew, replaced or removed.
- When deleting an atom, again the environment molecules containing the atom to be deleted have to be constructed. Within these molecules the atom has to be removed and the molecules have to be reconstructed in the appropriate way. Evaluating the qualification criterion delivers those resulting molecules which have to be materialized, replaced, or removed, respectively.

And again, the majority of these actions are in the responsibility of the data system. Therefore, the atom-cluster type definition is further restricted to a query of the following form:

SELECT ALL
FROM <molecule type definition clause>

Molecule Type Definition Clause

With respect to the molecule type definition clause we have to distinguish between hierarchical, network-like and recursive molecule type structures:

- With a network-like molecule type structure similar problems occur as in the case of a qualified projection. This is due to the semantics of a net structure (cf. chapter 2): Inserting an atom may establish a net structure, i.e. some molecules have to be expanded, deleting an atom may violate the net structure, i.e. atoms have to be removed from some molecules, and modifying an atom may cause both cases to arise (Fig. 5.3). Thus, similar actions as

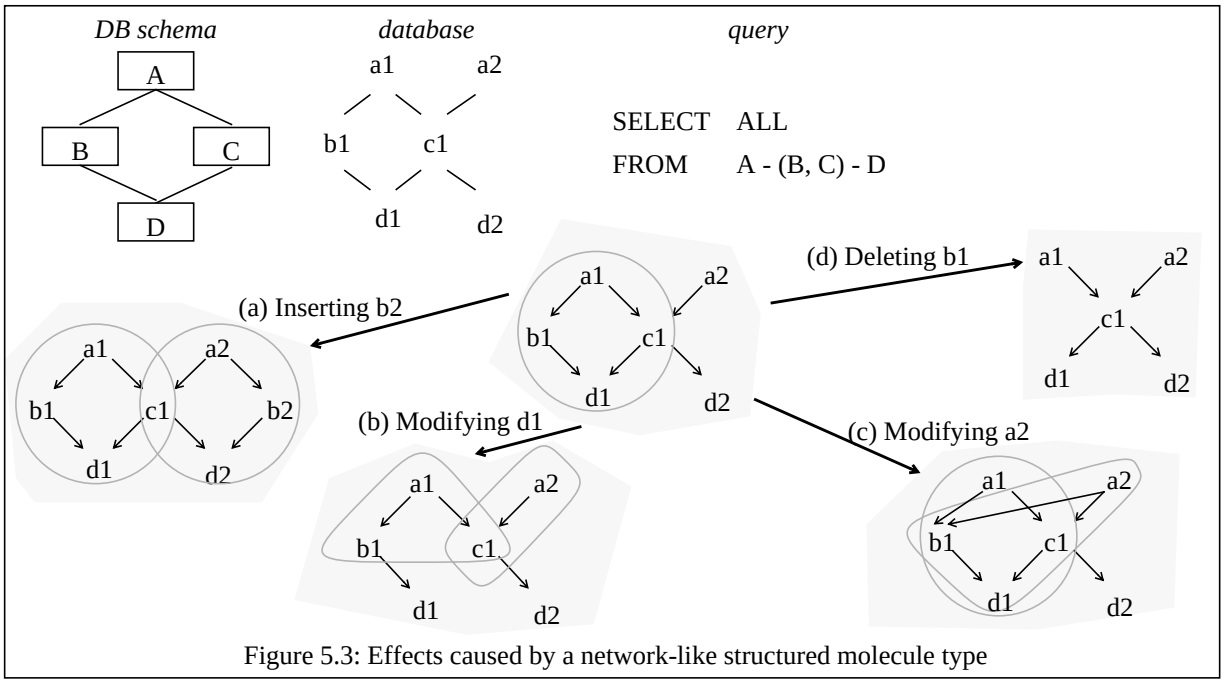

Figure 5.3: Effects caused by a network-like structured molecule type

in the case of a qualified projection are necessary, most of which are a task for the data system, and, as a consequence, network-like molecule type structures are not allowed within an atom-cluster type definition.

- In the case of a recursive molecule type structure the construction of the corresponding molecules is based on a breadth-first strategy, in order to guarantee that each component molecule referenced in more than one recursion level only belongs to the lowest one. However, when inserting a new atom or modifying or deleting an existing atom, the breadth-first strategy may cause different results in relating atoms to a certain recursion level. Therefore, the corresponding molecules have to be reconstructed by either of the modification operations being once again the task of the data system.
- With a hierarchical molecule type structure, however, a simple evaluation of the “down” and “up” references is sufficient in order to determine the molecules affected by a modification operation as we will demonstrate later on. This evaluation may be performed by the access system, since, in any case, the access system has to check the REFERENCE attributes with respect to the referential integrity.

As a consequence, only molecule types with a hierarchical structure may be materialized. Thus, an atom-cluster type definition looks like the following:

```
DEFINE ATOM_CLUSTER_TYPE <name> AS
            SELECT   ALL
            FROM     <one hierarchical structured molecule type)
```

5.2 Maintaining an Atom-Cluster Type

In the case of a hierarchical structured molecule type the access system is able to decide on its own which materialized molecules, i.e. atom clusters, are affected by an update operation to the database.

Inserting an atom

If the atom to be inserted corresponds to a root atom, a new atom cluster has to be generated. For this purpose, the corresponding “down” REFERENCE attributes specified in the atom-cluster type definition have to be evaluated and the referenced atoms have to be collected. For these atoms, in turn, the “down” REFERENCE attributes have to be evaluated until the leaf atoms are reached. The inserted root atom as well as the collected atoms are gathered up in a new atom cluster.

If the atom to be inserted corresponds to a component atom, the “up” REFERENCE attribute pointing to the predecessor of the appropriate atom type has to be evaluated in order to determine the corresponding predecessor atoms of the inserted atom. For each of these predecessor atoms the physical address of the appertaining atom clusters can be extracted from their address structure. These atom clusters have to be expanded by the inserted atom as well as by its successor atoms which are collected in the same way as above by evaluating the appropriate “down” REFERENCE attributes.

Modifying an atom

If the modification does not affect a REFERENCE attribute that defines the structure of an atom cluster, all atom clusters to which the modified atom belongs have to be determined by utilizing the address structure of this atom and within these atom clusters the corresponding atom has to be modified in the appropriate way. Thus, the usual management of replicas has to be performed.

If an “up” REFERENCE attribute pointing to the predecessors of the corresponding atom is modified, the old attribute value has to be compared to the new attribute value in order to determine those predecessors which are removed and those which are added. For each new predecessor the same procedure as for inserting an component atom may be applied. If predecessors are removed, the modified atom as well as its successors have to be removed from the corresponding atom clusters.

If a “down” REFERENCE attribute referring to successors of the atom is modified, the old attribute value and the new attribute value must be again compared in order to determine those successors which are removed or added, respectively. Each new successor as well as its successors have to be included into the atom clusters determined by the modified atom. Each removed successor as well as its successors, however, have to be deleted from the corresponding atom clusters.

Deleting an atom

If a root atom is deleted, the whole atom cluster determined by this atom has to be removed. If a component atom is deleted, this atom as well as all its dependent atoms, i.e. successors, have to be removed from the atom clusters determined by the deleted atom. This may be done in the same way as deleting a successor which has to be removed when an atom is modified.

In either case, if an atom is added to or removed from an atom cluster, the address structure of the corresponding atom has to be modified in order to indicate all existing replicas of an atom.

However, one special case has to be considered in more detail. A single atom may be multiply included in an atom cluster if the relationship between two atom types is multi-valued. In this case, a corresponding atom may have more than one predecessor within an atom cluster. As a consequence, if atoms are stored without duplicates within an atom cluster, appropriate information has to be kept in order to decide quickly whether or not an atom really has to be added to or removed from an atom cluster. This information, however, is part of the storage structure of an atom cluster which we will describe in the following.

5.3 The Storage Structure of an Atom-Cluster Type

The concept of atom clusters has been introduced in order to speed up construction of frequently used molecules by allocating all atoms of a corresponding molecule in physical contiguity. For this purpose different steps in mapping an atom cluster onto a page or page sequence are introduced:

From a logical point of view an atom cluster corresponds either to a heterogeneous or to a homogeneous atom set described by a so-called *characteristic atom*. This characteristic atom simply contains references to all atoms, grouped by atom types, belonging to the atom cluster (Fig. 5.4a). Moreover, the characteristic atom contains for each reference to such an atom an appropriate reference. Thus, the characteristic atom has to be evaluated in order

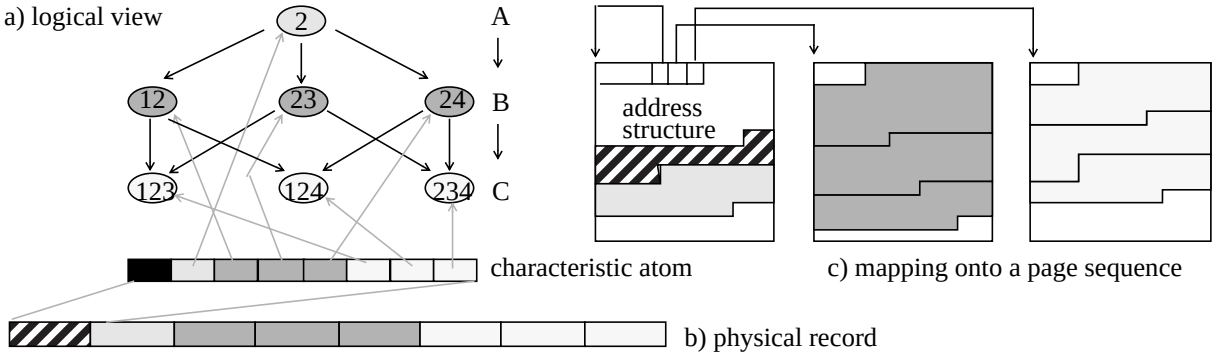

Figure 5.4: Example of an atom cluster

to determine whether or not an atom really has to be added to or removed from an atom cluster or whether only the characteristic atom has to be adapted.

Each atom cluster is mapped onto a so-called physical record, i.e. a byte string of variable length containing the characteristic atom as well as all atoms referenced by the corresponding characteristic atom (Fig. 5.4b). All atoms are included only once although there may exist multiple references to one atom.

The physical record, in turn, is mapped onto a page or a page sequence depending on its current length. If the record goes beyond the page size assigned to the segment of the corresponding atom-cluster type, this record is mapped onto a page sequence. Otherwise, the record is mapped onto a single page. Within a page multiple records may be stored subsequently without intermediate space, whereas a page sequence always contains a single physical record. The mapping of a physical record onto a page sequence is performed as follows (Fig. 5.4c): All atoms of a single atom type are placed into a subrecord. All subrecords are subsequently mapped onto pages. If a subrecord exceeds the free space available within a page, a new page is allocated. If a subrecord requires multiple pages, these pages are exclusively used by the subrecord. However, in order to quickly locate an atom within an atom cluster, i.e. within a page sequence, an additional address structure is required which also depends on the current length of the physical record. If a record fits into a single page, no additional address structure is necessary, since a sequential search is always performed within a page. If the record is spread over a page sequence, the address structure initially consists of a simple table indicating the (first) page to which the appropriate subrecord is mapped for each atom type. In addition, each subrecord which requires multiple pages contains a further table indicating the page in which it is stored for each atom in order to avoid an exhaustive sequential scan over these pages.

This mapping mechanism described so far goes beyond other approaches concerning clustering of heterogeneous record sets, e.g. the physical representation of an NF^2 tuple [SS86] implemented in AIM-P [Da86] or DASDBS [DPS86] and the proposal of [KCB88], because we deal with replicated data and guarantee the automatic maintenance of consistency among these replicas.

5.4 Retrieval on Atom-Cluster Types

Concerning the retrieval operations on atom-cluster types and atom clusters one has to consider that the access system provides an atom-oriented interface, i.e. it is not possible to pass an atom cluster as a whole across the interface between the access system and the data system. Therefore, when accessing an atom cluster the appropriate characteristic atom is delivered, whereas the remaining atoms have to be requested by additional operations. As a consequence, the access system offers two different scan operations supporting a simple and efficient processing of atom clusters:

The *atom-cluster type scan* allows for the sequential processing of all atom clusters, i.e. characteristic atoms, of a certain atom-cluster type. In order to restrict the result set of such a scan an unqualified projection clause as well as a qualification criterion may be specified. Using a projection clause those atom types, i.e. the references to the corresponding atoms, are defined which are needed for further processing. The qualification criterion selects those atom clusters satisfying the corresponding condition, which has to fulfil the so-called single-scan property

[DPS86], i.e. it has to be decidable during a single scan through an atom cluster. If this condition refers to the IDENTIFIER values of certain atoms which have to be contained in the desired atom clusters, the access system may utilize the corresponding address structure in order to speed up processing by determining the associated atom clusters without access to the data.

Since the atom-cluster type scan only delivers the characteristic atoms of the qualified atom cluster, the remaining atoms may be selected by either utilizing the corresponding IDENTIFIER values for direct access or by utilizing an atom cluster scan.

The *atom cluster scan* supports the sequential processing of all atoms of a certain atom type within a single atom cluster. Again, a projection clause as well as a selection criterion (decidable on a single atom) may be specified in order to restrict the result set to the attributes of these atoms which are required.

6. Using Atom Clusters for Efficient Molecule Construction

The usefulness of clustering atoms of heterogeneous types is obvious in systems that support static hierarchical objects, e.g., NF^2-tuples [SS86]. All records belonging to a structured tuple form a cluster, yielding minimal costs when accessing the whole tuple. Restrictions referring to the internal structure of the tuple can be evaluated very cheaply, since the structure is reflected by the cluster, and all data needed are in main memory after the cluster has been read. Furthermore, when access to subrecords is allowed only by reference to the whole tuple, there is no need for other, redundant data representations.

The MAD model, however, allows the dynamic definition of complex objects (molecules), as described in chapter 2. Therefore, it is more difficult to determine, which atoms should be clustered following a molecule type structure. On the other hand, when the optimizer chooses an access sequence for *construction of simple molecules*, it has to decide, whether or not existing atom clusters should be used. To give some hints, when the use of atom clusters may enhance efficiency, we consider the two phases of query evaluation introduced in chapter 4. We illustrate the different cases by examples, which suppose the existence of an atom-cluster type A-B-C, and of a fast access path on $A.Att_1$.

The first phase decides whether a molecule fulfils Q(M). If Q(M) contains conditions which reference the molecule structure, i.e., require parts of the molecule to be built up before the condition can be tested, an atom-cluster type scan, possibly enhanced by an appropriate selection criterion, may be useful, because it allows cheap access to all atoms involved. In the following example, the condition cannot be evaluated before the whole molecule is built up. The most efficient way to do so is to use the atom-cluster type.

```
SELECT    ALL
FROM      A - B - C
WHERE     EXIST EXACTLY (4) C: C.Att1=A.Att2
```

An atom-cluster type scan delivers the characteristic molecule of an atom cluster. The atoms of this cluster are then available in main memory, and thus can be cheaply accessed either by direct access using their IDENTIFIER value or by an atom cluster scan (cf. chapter 5).

An important criterion for the use of atom clusters in the first phase is the coincidence of the direction of the molecule type structure with that of the atom-cluster type. In the following example the atom-cluster type is of no use:

```
SELECT    ALL
FROM      C - B - A
WHERE     EXIST EXACTLY (4) A: C.Att1=A.Att2
```

Since access to an atom cluster is very fast when the IDENTIFIER value of one of its atoms is known (cf. chapter 5), it may be combined with an access path scan, as in the following example:

SELECT ALL
FROM A - B - C
WHERE $A.Att_1$=7 AND FOR ALL C: $C.Att_1$>$A.Att_3$

In this case, all A atoms with Att_1=7 are determined via the access path scan. Then, the corresponding atom clusters are read to evaluate the rest of the condition. A more complex case occurs, if there is no access path on $A.Att_1$. One could use an atom type scan instead of the access path scan, accessing the rest of the molecule as above, or could alternatively employ an atom-cluster type scan with the selection criterion $A.Att_1$=7. The condition FOR ALL C: $C.Att_1$>$A.Att_3$, however, cannot be evaluated within the access system, as described in chapter 5. Which of these two alternatives is better depends on the selectivity of $A.Att_1$=7. If it is high, the first one is preferable, because checking the condition is cheaper. If it is low, the first alternative again causes many fetches of A atoms by the atom cluster access which are already in memory due to the atom type scan and already known to the data system.

While the second phase of query evaluation normally has no impact on the decisions concerning the first phase, this does not hold, when atom-cluster types are available. If, for example, molecule type and atom-cluster type have an identical structure, the second phase becomes very cheap, if the atom-cluster type is used to evaluate the restriction in the first phase. Therefore, costs of both phases have to be taken into consideration for optimization purposes. The following case illustrates that the considerations made during the first phase (last example) can be utilized for the second phase, too:

SELECT ALL
FROM A - B - C
WHERE $A.Att_1$=5

Here again, the question is whether to use an atom type scan and access the atom cluster by its IDENTIFIER value, or to use an atom-cluster type scan with an appropriate selection criterion.

Even if the direction of the molecule type structure and that of the atom-cluster type structure do not match, atom-cluster types can be useful to traverse the molecule (in opposite direction). In the following example, it is a promising approach to look for the qualifying A atoms first, possibly supported by an access path. The corresponding C atoms can be found cheaply by reading the atom clusters of type A-B-C containing these A atoms. Afterwards the remaining atoms of types B and A have to be fetched by direct access using the appropriate REFERENCE attribute values:

SELECT ALL
FROM C - B - A
WHERE EXISTS $A.Att_1$ = 7,

Atom-cluster types may accelerate the second phase, even if they are only subgraphs of the molecule types, as shown below:

SELECT ALL
FROM D - A - B - C

Since access to atom clusters via its root's IDENTIFIER is fast, A-B-C submolecules referenced by D atoms may be fetched in this way.

The discussion above shows the difficulty of optimizer decisions when atom-cluster types have to be taken into account. Access strategies become much more complex, and cost models have to be developed which cope with the critical parameters introduced above.

7. Conclusions

Dynamism in complex object definitions, as supported by the MAD model, requires new concepts for clustering. In the PRIMA system, heterogeneous sets of atoms can be clustered according to a specific molecule structure. This storage structure is kept redundantly, and therefore serves only for efficiency enhancement. It is managed by the access system, which guarantees transparency of this redundant data for all update operations but, in turn, offers

operations to retrieve atoms via atom clusters. To keep the division of labor between access system and data system, the complexity of atom cluster definitions has to be restricted to correspond to a single hierarchical molecule type without any projection or restriction. Since the access to an atom cluster causes all appertaining atoms be loaded into main memory, the construction of a corresponding molecule can be done very efficiently. Thus, the dynamism of molecule type definition provided by the MAD model can be achieved in many cases at the efficiency of cluster mechanisms for static structures.

In spite of this quite simple structure it is difficult to decide whether the construction of molecules by the data system should be done with or without the use of existing atom clusters. For these reason, we see the following major fields for further research:

- Finding a good access strategy is more complicated than in the conventional case, if atom-cluster types are defined. Therefore, the optimizer must be extended by new rules and new cost models, which have to be developed.
- The usefulness of each atom-cluster type definition has to be checked, in order to destroy the definition when the update overhead goes beyond the efficiency gain for retrieval. For this purpose, appropriate statistics have to be kept. On the other hand, one needs hints, as to which atom-cluster type definitions could be useful to enhance efficiency. It still has to be investigated which kinds of statistics are required for these task. Furthermore, if there is any good heuristic to support either of both decisions, the system should initialize the corresponding actions by itself rather than by order of the database administrator.
- The structure of an atom-cluster type must be chosen very carefully. It is to support as many queries as possible, and as good as possible. Thus, one has to find a good compromise between “completeness” (i.e., all atoms of the molecule type definition clause of a query are contained in the atom-cluster type with the appropriate structure) and “generality” (i.e., the optimizer chooses the structure for the evaluation of many database requests, which in general means, for many different queries). If atom clusters of one type overlap (i.e., do not represent a strongly-hierarchical structure), an atom-cluster type scan causes several atoms be read multiply from disk. This should obviously be avoided. When the atom clusters are accessed using the value of their root atom’s IDENTIFIER attribute, however, they may serve well to enhance the speed of a query evaluation. Thus, the usefulness of atom-cluster type definitions depends even on optimizer strategies. For this reason, we claim for a system controlled atom-cluster type definition, as already mentioned above.

Acknowledgement

We would like to thank T. Härder and B. Mitschang for their helpful comments on a earlier version of this paper which helped to improve the presentation of the important issues. Thanks are also due to H. Neu and I. Littler for preparing the manuscript. Furthermore, we would like to thank the anonymous referees for their fruitful comments and the inspiration of new ideas.

References

AL80 Adiba, M.E., Lindsay, B.G.: Database Snapshots, in: Proc. 6th VLDB, Montreal, 1980, pp. 86-91.

As81 Astrahan, M.M., et al.: A History and Evaluation of System R, in: CACM 24:10, 1981, pp. 632-646.

BB84 Batory, D.S., Buchman, A.P.: Molecular Objects, Abstract Data Types and Data Models: A Framework, in: Proc. 10th VLDB, Singapore, 1984, pp. 172-184.

BLT86 Blakeley, J.A., Larson, P.-A., Tompa, F.W.: Efficiently Updating Materialized Views, in: Proc. SIGMOD Conf., Washington, 1986, pp. 61-71.

Ch76 Chen, P.P.: The Entity-Relationship-Model - Toward a Unified View of Data, in: ACM TODS 1:1, 1976, pp. 9-36.

Da86 Dadam, P., et al.: A DBMS Prototype to Support Extended NF2-Relations: An Integrated View on Flat Tables and Hierarchies, in: Proc. SIGMOD Conf., Washington, 1986, pp. 356-367.

DD86 Dittrich, K.R., Dayal, U. (eds): Proc. Int. Workshop on Object-Oriented Database Systems, Pacific Grove, 1986.

DPS86 Deppisch, U., Paul, H.-B., Schek, H.-J.: A Storage System for Complex Objects, in: [DD86], pp. 183-195.

Ha87 Hanson, E.N.: A Performance Analysis of View Materialization Strategies, in: Proc. SIGMOD Conf., San Francisco, 1987, pp. 440-453.

Hä88 Härder, T. (ed.): The PRIMA Project - Design and Implementation of a Non-Standard Database System, SFB 124 Research Report No. 26/88, University Kaiserslautern, 1988.

HMMS87 Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - A DBMS Prototype Supporting Engineering Applications, in: Proc. 13th VLDB, Brighton, 1987, pp. 433-442.

HSS88 Härder, T., Schöning, H., Sikeler, A.: Parallelism in Processing Queries on Complex Objects, appears in: Proc. Int. Symp. on Databases in Parallel and Distributed Systems, Austin, Texas, 1988, pp. 131-143.

KCB88 Kim, W., Chou, H.-T., Banerjee, J.: Operations and Implementation of Complex Objects, in: IEEE Transactions on Software Engineering 14:7, 1988, pp. 985-996.

LHMPW86 Lindsay, B., Haas, L., Mohan, C., Pirahesh, H., Wilms, P.: A Snapshot Differential Refresh Algorithm, in: Proc. SIGMOD Conf., Washington, 1986, pp. 53-60.

Mi88a Mitschang, B.: Towards a Unified View of Design Data and Knowledge Representation, in: Proc. 2nd Int. Conf. on Expert Database Systems, Tysons Corner, Virginia, 1988, pp. 33-49.

Mi88b Mitschang, B.: Ein Molekül-Atom-Datenmodell für Non-Standard-Anwendungen - Anwendungsanalyse, Datenmodellentwurf und Implementierungsaspekte, Ph.D. Thesis, University Kaiserslautern, 1988.

SAH87 Stonebraker, M., Anton, J., Hanson, E.: Extending a Database System with Procedures, in: ACM TODS 12:3, 1987, pp. 350-376.

Sch88 Schöning, H.: The PRIMA Data System: Query Processing of Molecules, in: [Hä88], pp. 101-115.

Se87 Sellis, T.K.: Efficiently Supporting Procedures in Relational Database Systems, in: Proc. SIGMOD Conf., San Francisco, 1987, pp. 278-291.

SI84 Shmueli, O., Itai, A.: Maintenance of Views, in: Proc. SIGMOD Conf., Boston, 1984, pp. 240-255.

Si88a Sikeler, A.: Buffer Management in a Non-Standard Database System, in: [Hä88] pp. 37-67.

Si88b Sikeler, A.: Supporting Object-Oriented Processing by Redundant Storage Structures, in: Proc. Int. Conf. on Computing and Information (ICCI ’89), Toronto, 1989.

SS86 Schek, H.-J., Scholl, M.H.: The Relational Model with Relation-Valued Attributes, in: Information Systems 2:2, 1986, pp. 137-147.

St75 Stonebraker, M.: Implementation of Integrity Constraints and Views by Query Modification, in: Proc. SIGMOD Conf., San Jose, 1975, pp. 65-78.