A DECOMPOSITION STORAGE MODEL

George P Copeland
Setrag N Khoshafian

Microelectronics And Technology Computer Corporation
9430 Research Blvd
Austin, Texas 78759

Abstract

This report examines the relative advantages of a storage model based on decomposition (of community view relations into binary relations containing a surrogate and one attribute) over conventional n-ary storage models

There seems to be a general consensus among the database community that the n-ary approach is better This conclusion is usually based on a consideration of only one or two dimensions of a database system The purpose of this report is not to claim that decomposition is better Instead, we claim that the consensus opinion is not well founded and that neither is clearly better until a closer analysis is made along the many dimensions of a database system The purpose of this report is to move further in both scope and depth toward such an analysis We examine such dimensions as simplicity, generality, storage requirements, update performance and retrieval performance

1 INTRODUCTION

Most database systems use an n-ary storage model (NSM) for a set of records This approach stores data as seen in the conceptual schema Also, various inverted file or cluster indexes might be added for improved access speeds The key concept in the NSM is that all attributes of a conceptual schema record are stored together For example, the conceptual schema relation

R	sur	a1	a2	a3
	s1	v11	v21	v31
	s2	v12	v22	v32
	s3	v13	v23	v33

contains a surrogate for record identity and three attributes per record The NSM would store si, v1i, v2i and v3i together for each record i

Some database systems use a fully transposed storage model, for example, RM (Lorie and Symonds 1971), TOD (Wiederhold et al 1975), RAPID (Turner et al 1979), ALDS (Burnett and Thomas 1981), Delta (Shibayama et al 1982) and (Tanaka 1983) This approach stores all values of the same attribute of a conceptual schema relation together Several studies have compared the performance of transposed storage models with the NSM (Hoffer 1976, Batory 1979, March and Severance 1977, March and Scudder 1984) In this report, we describe the advantages of a fully decomposed storage model (DSM), which is a transposed storage model with surrogates included The DSM pairs each attribute value with the surrogate of its conceptual schema record in a binary relation For example, the above relation would be stored as

a1	sur	val	a2	sur	val	a3	sur	val
	s1	v11		s1	v21		s1	v31
	s2	v12		s2	v22		s2	v32
	s3	v13		s3	v23		s3	v33

In addition, the DSM stores two copies of each attribute relation One copy is clustered on the value while the other is clustered on the surrogate These statements apply only to base (i e , extensional) data To support the relational model, intermediate and final results need an n-ary representation If a richer data model than normalized relations is supported, then intermediate and final results need a correspondingly richer representation

This report compares these two storage models based on several criteria Section 2 compares the relative complexity and generality of the two storage models Section 3 compares their storage requirements Section 4 compares their update performance Section 5 compares their retrieval performance Finally, Section 6 provides a summary and suggests some refinements for the DSM

2 SIMPLICITY AND GENERALITY

This Section compares the two storage models to illustrate their relative simplicity and generality Others (Abrial 1974, Deliyanni and Kowalski 1977, Kowalski 1978, Codd 1979) have argued for the semantic clarity and generality of representing each basic fact individually within the conceptual schema as the DSM does within the storage schema

2 1 Support Of Multivalued Attributes

A more comprehensive data model than normalized relations might allow multivalued attributes or repeating fields, where a single attribute of a record can have more than one value This is a feature of Childs' extended set theory model (1977) For example, the relation R1 in Section 1 might have two values v23 and v24 for attribute a2 in record s3

R	sur	a1	a2	a3
	s1	v11	v21	v31
	s2	v12	v22	v32
	s3	v13	v23,v24	v33

A practical example might be that each a2 contains a set of an employee's children

Support of multivalued attributes with the NSM leads to either reduced data independence or increased complexity

One alternative is to always further normalize the relations containing multivalued attributes (Fagin 1977) That is, for each multivalued attribute, a new relation is formed by projection on that attribute and its surrogate This alternative reduces physical data independence, since a change from a single-valued attribute to a multivalued attribute, or vice versa, in the conceptual schema causes a change in the storage structure

A second alternative is to directly support multivalued attributes in the storage model This alternative leads to complexity A more complex storage structure is required which supports multiple values in a field Also, if the records of R1 are clustered on attribute a2, the two values might dictate that record s3 be placed in two different cluster blocks A copy of the entire record could be stored in each cluster block This causes complexity during update, since all copies of the record must be found and updated Alternatively, the entire record could be stored in only one of the cluster blocks with some sort of linkage mechanism among the blocks This causes complexity during update, since the linkages must be maintained It also causes retrieval to be asymmetric, since indirections are sometimes but not always required

The DSM handles multivalued attributes with higher data independence and without additional complexity The second value would change only the a2 DSM relation

a2	sur	val
	s1	v21
	s2	v22
	s3	v23
	s3	v24

Changing an attribute from single valued to multivalued in the conceptual schema causes no change in the storage structure The storage structure is still the simple binary relation No update problems occur due to duplicated surrogates since surrogates are never modified

2 2 Support Of Entities

A more comprehensive data model than the original relational model might support the notion of entities, where an object's individual identity is preserved by explicitly representing it independently of its value (Smith and Smith 1978) This is a feature of several data models, including RM/T (Codd 1979) For example, the existence of an object could be represented in the data model with none or only a few of its attribute values known A practical example would be an employee who is hired but most of the employee's data is not yet known Entities also allow any or all of an employee's attribute values to change, yet preserving the continuity that the employee is the same person

Surrogates are a convenient technique for entity support In addition, the decomposition approach of the DSM dictates that a separate entity relation should be stored The DSM entity relation for the example in Section 1 would be

r	sur
	s1
	s2
	s3

If one or more of the attribute values were unknown for one of the entities, then those attribute values would not have an entry in the corresponding DSM attribute relations Only one copy of r would be stored, clustered on the surrogate This approach involves little additional complexity for update and retrieval to support entities Only the additional update of the entity relations is required for update Only the additional test for membership in the entity relations is required for retrieval

Without this degree of decomposition, the NSM would require explicit null values to be stored in the single relation

2 3 Support Of Multiple Parent Relations

A data model with more generality than relations might allow multiple parent relations, where a single record can have more than one parent relation For example, the s3 record of relation R in Section 1 might also belong to a second relation Q

R	sur	a1	a2	a3
	s1	v11	v21	v31
	s2	v12	v22	v32
	s3	v13	v23	v33

Q	sur	a1	a2	a3
	s3	v13	v23	v33
	s4	v14	v24	v34

A practical example might be that R and Q describe employees for different companies which have recently merged and employee s3 now works for both companies

Support of multiple parent relations in the NSM leads to complexity One alternative is to store the s3 record redundantly in both relations This leads to update complexity, since all copies of the duplicated data must be found and updated In particular, when s3 is updated within one relation, it is difficult to determine where else it is stored A second alternative is to actually store the full record in one relation and to store a reference to it in all others This leads to complexity in both updating and retrieval An update to s3 within R would cause either cluster or inverted file indexes in Q to change Also, retrieval becomes asymmetric A third alternative is to completely reorganize the conceptual schema This reduces logical data independence, causing problems for existing applications, and reduces physical data independence, causing storage reorganization

The DSM would represent the relations using entity relations

r	sur
	s1
	s2
	s3

q	sur
	s3
	s4

a1	sur	val
	s1	v11
	s2	v12
	s3	v13
	s4	v14

a2	sur	val
	s1	v21
	s2	v22
	s3	v23
	s4	v24

a3	sur	val
	s1	v31
	s2	v32
	s3	v33
	s4	v34

This approach supports multiple parent relations without redundancy or reduced data independence

2 4 Support Of Heterogeneous Records

A data model with more generality than relations might allow heterogeneous records, where records of a single relation can have different attributes Such records usually have some attributes the same and some different For example, the following relation has two record types t1 and t2

R	sur	type	a1	a2	a3	a4
	s1	t1	v11	v21	v31	NA
	s2	t2	v12	v22	NA	v42
	s3	t2	v13	v23	NA	v43
	s4	t1	v14	v24	v34	NA

A practical example might be that the relation describes a set of employees, type t1 would describe salesmen and t2 engineers Both types would include name (a1) and birthdate (a2), but salesmen would have a company car (a3) and engineers would have a project (a4)

Direct support of heterogeneous records in the NSM would require explicit storage in each record of record type and/or an indicator (NA) that certain attributes are not applicable For retrievals referencing attributes which are not applicable for some records (e g , retrieve a1 where a3="v"), additional checking of type or NA would be required on each record, since records would be encountered of both types

The DSM would represent the relation as

a1	sur	val
	s1	v11
	s2	v12
	s3	v13
	s4	v14

a2	sur	val
	s1	v21
	s2	v22
	s3	v23
	s4	v24

a3	sur	val
	s1	v31
	s4	v34

a4	sur	val
	s2	v42
	s3	v43

This approach does not require storage in each record of either record type or an indicator of not applicable attributes No additional checking in each record for retrievals would be required, since only records which have relevant attributes would be encountered

The NSM could indirectly support heterogeneous records cleanly by partial decomposition, forming a new relation for each type The result for this example would be similar to the DSM representation above

R1	sur	a1	a2
	s1	v11	v21
	s2	v12	v22
	s3	v13	v23
	s4	v14	v24

t1	sur	a3
	s1	v31
	s4	v34

t2	sur	a4
	s2	v42
	s3	v43

Of course, the DSM has the advantage of doing this automatically without knowledge of the types

2 5 Support Of Directed Graphs

A data model with more generality than relations might allow a directed graph structure, where any object can have any number of objects as either parents or values This is the data structure of object-oriented languages such as Smalltalk (Goldberg and Robson 1983)

Decomposition can support directed graphs with minimum complexity Two types of binary relations are needed One type is the same as in the basic DSM, which represents the relationships between pure values and their entities A second type has the form

parent sur	child sur

This second type represents the parent/child relationships between entities Grouping of sets of relationships into relations could be based either on common instance set or on common type

2 6 Differential Files And Temporal Support

Forward differential files (Severance and Lohman 1976) provide a way to use RAM to reduce the number of disk accesses for update, while not increasing disk accesses for retrieval Backward differential files provide a way to reduce storage requirements and provide fast access to current data for a temporal data model (Copeland and Maier 1984), which maintains and provides access to the history of database states

Differential files could be directly supported in the NSM by representing an attribute modification by storing the full record This would result in low storage utilization To improve storage utilization in NSM systems, complexity is usually increased by the addition of individual attribute identifiers to distinguish which attributes are modified

Differential files can be directly supported in the DSM with efficient storage utilization, so that additional complexity is not needed Each attribute modification would be directly represented by a separate shorter DSM record

2 7 Storage Structures

The NSM must support records containing a variable and unbounded number of attributes Many alternative strategies have been used to represent n-ary records, such as a linked list, an index per record to its attributes and others (Batory 1984) The complexity of such strategies is evidenced by the fact that systems place an upper bound on the number of attributes allowed

For base (i e , extensional) data, the DSM need support only the simple binary relation Furthermore, the first attribute containing the surrogate is fixed in length Only one possibly variable-length value need be supported With the DSM, any number of conceptual schema attributes can be supported without additional complexity For temporary or final results, the DSM must support whatever structures are present in the data model

2 8 Access Methods

The NSM may employ many different access methods to improve performance over an exhaustive scan of all records We divide these into two classes, clustering and inverted files Clustering (Chang and Fu 1978) is usually faster when a large number of records qualify for retrieval, while inverted files (Cardenas 1975) are usually faster when few records qualify In either case, speed is a strong function of how many attributes in the retrieval predicate are also chosen as an index However, choosing all attributes as indexes usually yields update performance problems as well as additional storage space Thus, complex tradeoffs are required by users to efficiently tune the database system

The DSM in this report requires no tradeoffs and is automated Each decomposed relation is binary, containing one attribute value and one surrogate Two physical copies of each binary relation are stored, one indexed only on the attribute value and the other only on the surrogate Since each binary relation has a single index, single key clustering is always used for improved performance over a wide range of number of qualifying records Inverted files for secondary indexes are never needed No decisions are required by users or even by the system

2 9 Physical Data Independence And Availability

Physical data independence means that changes to either the conceptual or the storage schema have minimal impact on each other Physical data independence is desirable because it allows the logical and physical aspects of the database system to be more quickly and easily controlled, so that the database system is more stable as it tries to adapt to changes in the dynamic real world The DSM provides a cleaner separation between the conceptual and storage schema than the NSM does We have already described how the DSM supports various extensions to the basic relational model with a higher level of data independence than the NSM

Using the DSM, changes in the conceptual schema have less impact on the storage schema For example, the addition or removal of an attribute to a conceptual schema relation is seen at the storage level as simply adding a new relation rather than modifying an existing one

Using the DSM, no changes in the storage schema are required for performance enhancement Since there are no changes in the storage schema, none can propagate up to the conceptual schema The NSM may require modification of either clustering or inverted file methods for performance tuning These modifications not only require complex human intervention, but may cause data in the conceptual schema to become unavailable during the modification

In addition, the impact of locking on availability can be minimized using the DSM with less complexity than using the NSM For example, locking a single attribute in the conceptual schema is seen at the storage level as locking only one of the DSM relations The other attributes are available for other transactions, as long as the attributes are not linked by integrity constraints This is true regardless of whether the concurrency control subsystem uses relation-level, record-level or block-level locking For the NSM, either the entire relation would have to be locked, or a more complex scheme for indicating which attributes are locked would have to be employed Block-level locking would force entire records to be locked using the NSM On the other hand, if locking of entire records is needed, the NSM would be more efficient

2 10 Reliability And Recovery

The simplicity of the DSM should improve the reliability of the database system There is less to implement and less chance of introducing bugs The KISS principle applies here The replicated data of the DSM should allow faster and simpler recovery from media failure

3 STORAGE REQUIREMENTS

This Section describes the relative storage requirements of the two storage models We first describe data storage, then index storage, then total storage

3 1 Data Storage Requirements

The DSM requires two copies of the data This has the obvious effect of a factor of 2 increase in the data storage requirements of the DSM compared to the NSM

The DSM requires the storage of duplicated surrogates A copy of each surrogate is required for each of the attributes This has the effect of a factor of from 1 to 2 increase in the data storage requirements of the DSM compared to the NSM, depending on the relative size of attribute values and surrogates

Various compression techniques exist for multiple attributes with the same value For example, run-length compression has been proposed for decomposed files (Eggers 1981) This property has been recognized in statistical databases (Shoshani et al 1982) With surrogates incorporated in the DSM, some of the versatility for compression is lost For example, the DSM relation clustered on surrogates cannot easily be compressed, except for the repeating surrogates involved in multivalued attributes However, the DSM still offers more flexibility for storage compression than the NSM For example, the DSM relation copy clustered on attribute values could store the set of records with the same attribute value together with the attribute value stored only once along with a list of their surrogates

The net effect of these three differences is quantified by the following ratio of DSM to NSM data storage requirements

$$2 * \frac{A*(AS+SS)}{A*AS+SS} * \frac{(AS/RV+SS)+(AS+SS)}{2*(AS+SS)} ,$$

where A is the number of attributes in the conceptual schema relation, AS is the average attribute size, SS is the surrogate size and RV is the average number of repeating fields The first factor accounts for the two copies of each DSM relation The second factor is the ratio of storage required for a conceptual schema record If attribute values are about the same size as surrogates, then this factor is about 2 If attribute values are much larger than surrogates, then this factor is about 1 The third factor is the ratio of compacted to non-compacted DSM storage If attribute values are much larger than surrogates and RV is large, this factor approaches 1/2 As RV approaches 1, this factor approaches 1 The net effect is that the NSM has between a 1 to 4 advantage for data storage Assuming typical constants to be A=10, AS=15, SS=5, RV=2, the factors are approximately

$$2 * 1 3 * 0 81,$$

so a typical value for the ratio of DSM to NSM data storage is 2 1

For data stored in a cache buffer, the DSM has an advantage Without considerable complexity, the NSM forces all attributes of a relation to be buffered together, even if the attributes differ significantly in usage frequency The DSM allows those attributes having a high usage frequency to easily be buffered independently from those in the same conceptual schema relation with a low usage frequency Thus, for a given buffer performance, the DSM usually requires less buffer space

3 2 Index Storage Requirements

An inverted file, such as a B-tree (Bayer and McCreight 1972), must resolve addressability down to each record That is, each B-tree must contain a key and a pointer for each record in its leaf nodes Even if abbreviated keys are used as in prefix B-trees (Bayer and Unterauer 1977), the size of a leaf node key approaches the size of an attribute value, since it must discriminate between each record's attribute value This causes the size of each B-tree to be about 1/A of the size of the relation, or more when pointers and non-leaf nodes are included Typically, several inverted files are needed per relation in the NSM, so that the total index storage requirements of the NSM often equals the size of its data storage requirements (Cardenas 1975)

Each of the two copies of a DSM relation has only one single key cluster index A cluster index is considerably smaller than the size of an inverted file index for two reasons The major reason is that a cluster index requires addressability only down to each disk block instead of each record This reduces the size of the cluster index by a factor equal to the number of DSM records per block, a number that would typically be several hundred A second reason is that the size of cluster keys are smaller than inverted file keys, since they need discriminate between ranges of values of large blocks of records instead of between each record's attribute value This reduces the size of a cluster index

The NSM could use a multikey clustering index (Bentley 1979, Nievergelt et al 1984) However, we discuss in Section 5 2 the limited utility of this approach

Even though 2*A cluster indexes are required for the DSM and only several inverted file indexes are required for the NSM, the total index requirements would typically be two orders of magnitude less for the DSM For the NSM, index size is of the same order as data size A typical value might be 50% For the DSM, index size is not significant compared to data size

3 3 Total Storage Requirements

The DSM is expected to increase data storage by a factor of between 1 and 4 with a typical value of 2 1 The DSM is expected to decrease index storage by roughly two orders of magnitude Index storage for the NSM often equals its data storage with a typical value of 50%, while DSM index requirements are not significant Thus, the total storage is larger for the DSM by a factor of between 1/2 and 4 with a typical value of 1 4

In most database applications today, storage capacity is a less critical issue than performance Most technological projections expect the storage capacity per dollar for magnetic disks to continue to improve by a factor of about two every three years However, seek times are expected to improve by only a factor of about two over the next decade, and latency times are not expected to improve significantly Write-once and rewritable optical disks are expected to provide even higher capacity per dollar than magnetic disks but with higher seek and latency times due to the more massive optical heads and slower rotation speeds Thus, for future databases, storage capacity will be an increasingly less critical issue than performance

4 UPDATE PERFORMANCE

This Section compares the update performance of the two storage models

4 1 Modifying An Attribute

Modifying a single attribute under the NSM requires one disk write for the block containing the attribute's record If the attribute has an inverted file, then an additional (usually single but sometimes multiple) disk write is required for the block containing that part of the inverted file that must be updated The probability that an attribute has an inverted file is I/A, where I is the number of inverted files for the relation Thus, the average number of writes is slightly greater than 1+I/A

The DSM requires three disk writes, one for the DSM relation clustered on surrogate and two for the DSM relation clustered on value since the modified record will usually have to move In addition, there is a low probability that each of the cluster indexes will require change This probability is inversely proportional to the number of DSM records per block (typically several hundred), since a cluster index requires change only upon overflow/underflow Thus the average number of writes is roughly 3

The above comparison assumes that the modification must be written to disk If instead, battery backup is used to make the RAM buffer non-volatile, disk writes can be delayed by storing the modification in differential files in the buffer Writes to disk can then be done later as a periodic or background task

4 2 Inserting Or Deleting A Record

Inserting or deleting an entire record under the NSM requires one disk write for the block containing the new or old record In addition, a (usually single but sometimes multiple) disk write is required for each inverted file of the NSM relation, so that the average number of disk writes is slightly greater than 1+I

The DSM requires two disk writes for each attribute of the record, one for each copy In addition, there is a low probability that each of the cluster indexes may require change, so that the average number of disk writes is slightly greater than 2*A

If battery backup is used to make the RAM buffer non-volatile, disk writes can be delayed by using a differential file in the buffer

5 RETRIEVAL PERFORMANCE

This Section compares the retrieval performance of the two storage models We first present performance equations and families of curves for several parameters for conjunctive retrieval patterns Then we discuss the key parameters causing performance differences in the two storage models Next we discuss the impact of limited buffer space for intermediate results Finally, we examine the potential concurrency of the two storage models and the impact of multiple disks

5 1 Conjunctive Retrieval Patterns

A comparison of retrieval performance is complicated by a dependence on the retrieval pattern We approach this problem by limiting our comparison to a generalized conjunctive retrieval pattern

The comparison is also complicated by a dependence on the number of records r that qualify for each step in the retrieval process We approach this problem by defining performance comparisons as a function of r Where multiple steps are involved, we make the simplifying assumption that r is the same for each step

We assume that the NSM has an inverted file index on each attribute constrained in a predicate This assumption favors the NSM, since it is rarely the case that all attributes of a relation have an inverted file index because of update overhead Thus, we are actually comparing the DSM, which requires no performance tuning, with a well tuned NSM In reality, the DSM should have an advantage where workload characteristics are not static

We assume for simplicity that each relation has A attributes of equal size AS plus a surrogate of size SS, and all relations have R records Using an effective disk block size of BS, the number of blocks NB required by a NSM relation is

$$NB = \text{ceiling}(R/\text{floor}(BS/(A*AS+SS)))$$

The number of blocks DB required by each DSM relation is

$$DB = \text{ceiling}(R/\text{floor}(BS/(AS+SS)))$$

We assume that all relations are on disk prior to retrieval execution and that the number of disk reads/writes provide a reasonable approximation to retrieval performance We ignore processor costs, since processor speed is projected to improve faster than disk speed We also ignore disk accesses for indexes for simplicity When retrieving qualified records from disk, the required number of disk block reads is strongly affected by whether the records are clustered or randomly distributed Clustering allows r qualifying records to be retrieved with the following number of disk block reads

$$CLB(BS,AS,SS,a,r) = \text{ceiling}(r*(a*AS+SS)/BS),$$

where a is the number of attributes in each storage relation For the NSM, a=A, and for the DSM, a=1

For accessing records randomly distributed onto disk blocks, a formula for the expected number of disk blocks containing at least one of r records is given by Yao (1977)

$$RMB(B,R,r) = B*(1 - \prod_{i=1}^{r} \frac{R-R/B-i+1}{R-i+1}),$$

where B is the total number of blocks For the NSM, B=NB, and for the DSM, B=DB

The following is a generalized conjunctive retrieval expression

```
ANS(X1,    ,Xnpa) <==
           R1(S1,    ,"V1",    ,"V2",    ,X1,    ),
           R2(S2,    ,X2,    ,S1,    ,Xnpa,    ),
             ,
           Rnjr(    ,"Vnca",Snjr-1)
```

where npa is the number of projected attributes needed for the final result, njr is the number of joined relations, and nca is the number of constrained attributes The Xi's and "Vi"'s can be spread throughout any of the relations Joins on surrogate Si are required between adjacent relations We assume the constrained attributes and the projected attributes are disjoint, so that

$$nca + npa =< A*njr$$

An equation for the number of disk blocks nb accessed using the NSM is

$$nb = njr*RMB(NB,R,r),$$

since r records are randomly distributed across each of the njr relations

The DSM equivalent retrieval expression is

ANS(X1,   ,Xnpa) <==
R11(S1,"V1"),R12(S1,"V2"),R13(S1,X1)   ,
R21(S2,X2),R22(S2,S1),R23(S2,Xnpa)   ,
,
Rnjr1(Snjr,"Vnca"),Rnjr2(Snjr,Snjr-1)

An equation for the number of disk blocks db accessed for the DSM is

$$db = nca*CLB(BS,AS,SS,1,r) + (njr-1+npa)*RMB(DB,R,r),$$

since r records are clustered in each of the nca relations which have a constrained value, r surrogates are randomly distributed over each of the relations involved in the njr-1 joins and the npa projections

Graphs of the nb/db ratio as a function of r are provided in Figures 2 through 4, assuming typical constants to be R=100,000, BS=5,000, A=10, AS=15, SS=5 We discuss these graphs in the following Section

5 2 Key Parameters For Retrieval

There are five key parameters causing performance differences in the two storage models One is full clustering on attribute values in the DSM vs inverted files for the NSM A second is the reduced size of the DSM relations A third is the number of attributes constrained in each retrieval A fourth is the number of attributes projected in each retrieval A fifth is the number of join relations in the conceptual schema

An inherent property of the DSM is that each attribute relation is fully clustered on its attribute value Full clustering is possible if only a single attribute requires indexing in each relation If a relation has multiple attributes which require a cluster indexing, then clustering must be compromised, so that the records containing a particular attribute are spread over many cluster blocks An approximation for the number of blocks which must be read using multidimensional clustering is given by (Chang and Fu 1978)

$$b = B^{1-p/d},$$

where B is the total number of blocks containing a relation, d is the number of cluster attributes (i e , dimensions), and p is the number of attributes bound to a constant in a conjunctive retrieval predicate that are also cluster attributes For the DSM, d=1, so that b=1 This function assumes that r is small enough so that all qualifying records fit into one block For larger r, CLB blocks are required If clustering were used for the NSM, d would typically be much larger than 1, so that b is quite large unless all cluster attributes are bound in a conjunctive retrieval predicate For the example used in Section 5 1, B=3125 for the NSM If d were only 2 and p were 1, b would be about 56 blocks even if r were 1 For this reason, inverted files usually offer higher performance with the NSM Inverted files do not cluster records on any attribute, so that RMB blocks are required Figure 1 illustrates the ratio of random to full clustered access with the number of blocks held constant (i e , we use B=NB in RMB and a=A in CLB), so that the size difference of the two storage models is factored out

$$\frac{RMB(NB,R,r)}{CLB(BS,AS,SS,A,r)}$$

Note that the fluctuations in the curve are due to the ceiling and floor functions, having break points at multiples of the number of records per block for the NSM (32) and DSM (250) Full clustering has no effect for either r=1 or r=R, but has a major effect for intermediate values of r The effect of clustering should have an increasingly important impact as database usage patterns move from single record to set retrieval One example of this would be retrievals with nonunique access keys or with ranges (Bentley 1979) Another example would be an environment with workstations and database servers on local area networks, where communications overhead is high

The reduced size of the DSM relations reduces the total number of blocks for each stored binary relation, since only one attribute is present in each DSM relation instead of A for the NSM Note that we have assumed attributes of equal length in our comparison Actually, the effect of reduced size of the DSM relations is even more important when the size of different attributes vary greatly, such as a mixture of text and formatted data in the same NSM relation The ratio of the number of blocks is approximately (i e , neglecting the ceiling and floor functions included earlier in NB and DB)

$$\frac{NB}{DB} = \frac{A*AS+SS}{AS+SS}$$

For the constants of Section 5 1 used in the graphs, this is 7 8 This provides an advantage for the DSM whenever an attribute must be exhaustively scanned Figure 1 illustrates the effect of size on random access

$$\frac{RMB(NB,R,r)}{RMB(DB,R,r)}$$

This improves performance significantly for intermediate to large r The RMB function is

Figure 1 Effects Of Clustering And Size

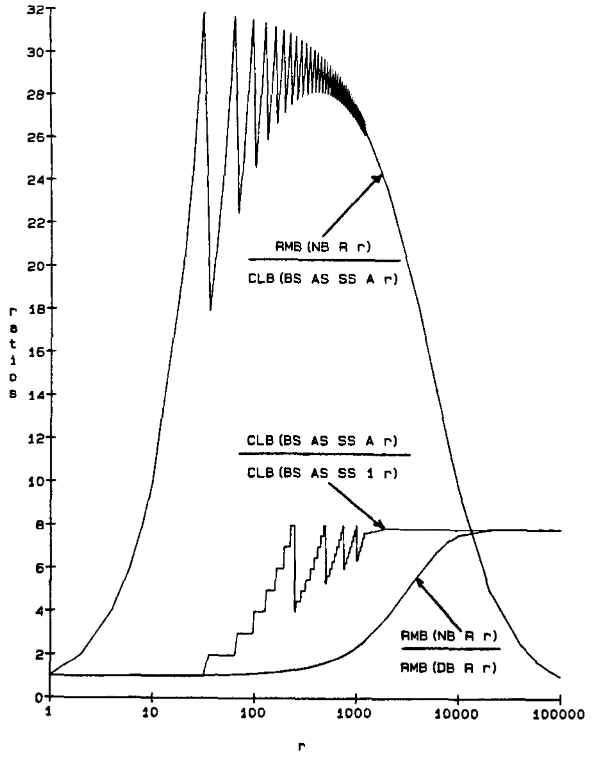


Figure 2 Varying The Number Of Projected Attributes

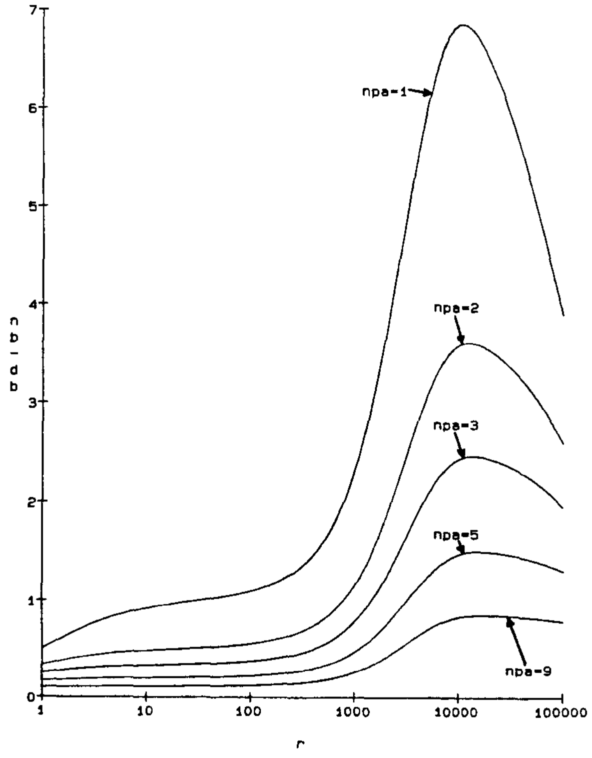


Figure 3 Varying The Number of Constrained Attributes

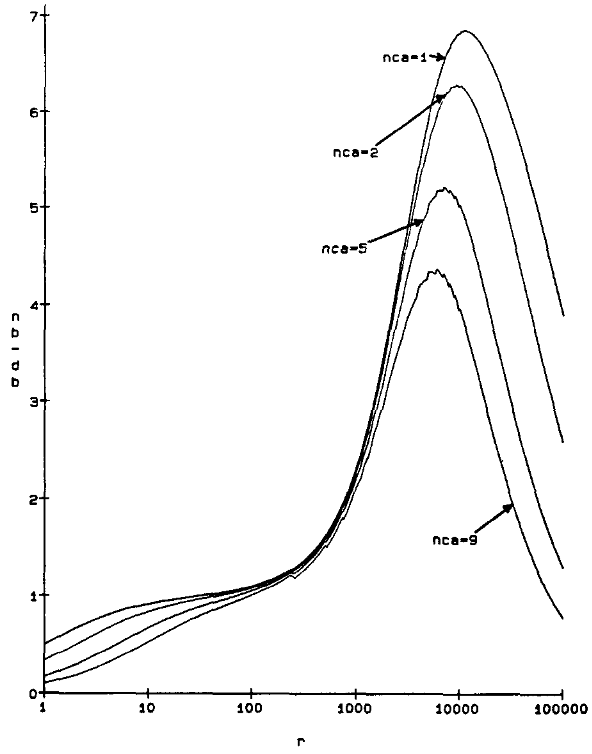
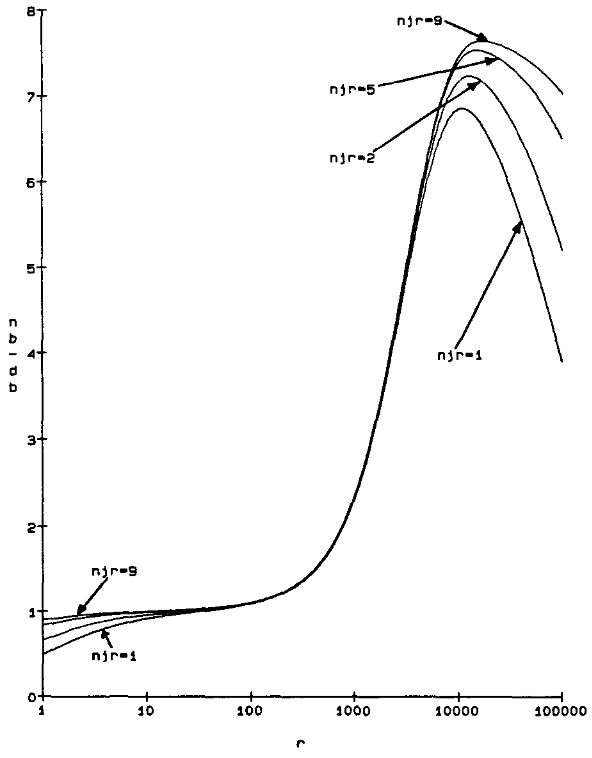


Figure 4 Varying The Number Of Joined Attributes



almost linear in r for small r Figure 1 also illustrates the effect of size on clustered access

$$\frac{CLB(BS,AS,SS,A,r)}{CLB(BS,AS,SS,1,r)}$$

This would improve performance for all but very small r, even if the NSM could use full clustering Note again the break points at multiples of the number of records per block for the NSM (32) and DSM (250)

Figure 2 illustrates the effect of varying the number of projected attributes npa For low npa, the DSM is worse for low r but better for intermediate to large r However, when npa approaches A, the DSM is worse for all r This is due to the large number of joins required for the DSM Figure 3 illustrates the effect of varying the number of constrained attributes nca The lower right portion of Figure 3 is unlikely to occur in reality, since a large nca should considerably limit r The effect of varying nca is similar to varying npa, but with some improvement for the DSM due to clustering Later in Section 5, we describe other phenomena which reduces these negative effects for the DSM

Figure 4 illustrates the effect of varying the number of joined relations njr in the conceptual schema The DSM is slightly worse for low r, but much better for intermediate to large r As njr increases, the DSM improves This is due to the fact that joins are faster for the DSM, so that when the number of joins in the conceptual schema are increased, NSM performance decreases faster than for the DSM

5 3 Effect Of Limited Buffer Space

The above retrieval performance comparison assumed unlimited buffer space for intermediate results of database operations such as join In this Section, we describe the effects of limited buffer space In NSM systems, limited buffer space for intermediate results often causes many additional disk accesses

The join operation between two relations is used here as an example of the effect of limited buffer space for intermediate results within an operation If both relations are already fully sorted on the join attribute, then a merge join (Bitton et al 1983) is possible A merge join can be performed with a single pass on each relation, since it requires little buffer space for intermediate results If one or both of the relations are not sorted on the join attribute, then sorting is required before the merge can take place If the size exceeds the available buffer space, then a slow external sort is required NSM relations are seldom sorted on the join attribute, since only one attribute or surrogate can be used for sorting of each stored relation Since DSM relations are sorted (fully clustered relations are sorted) on each attribute and surrogate, fewer sorts are required When a sort is required, the reduced size of the DSM relations will more often allow a fast internal sort

The DSM also has an advantage because most intermediate results between different operations are smaller For example, let us examine attributes which are required in the final result but are not involved in other operations It is simple with the DSM to delay access of these attributes until the last steps in retrieval processing For the NSM, it is tempting to capture these attributes earlier if they occur in the same records as attributes required for other operations This is because these attributes must be accessed anyway and accessing them as a separate operation is relatively inefficient for the NSM The problems are that they can significantly increase the size of intermediate results and that many of them are not required in the final result because of later restrictions

5 4 Potential Concurrency

We consider three types of concurrency in database operations parallelism within each operation, and both parallelism and pipelining among multiple operations

The potential of parallelism within each operation is the same for the two storage models, since the number of records in each DSM relation is the same as in the NSM

The potential of parallelism and pipelining among multiple operations is greater for the DSM The DSM relations which correspond to the same conceptual schema relation are often joined on the surrogate For example, the DSM pattern

```
ANS(X1,X2,X3) <==
          Ri(S,X1),Rj(S,X2),Rk(S,X3),Rl(S,"v")
```

has a potential of either parallelism or pipelining The algebra tree for this pattern would be

```
      join(sur)/project(val)
      _________|_________
    |    |    |          |
   Ri   Rj   Rk   select("v")/project(sur)
                         |
                        Rl
```

The select/project on Rl is performed first since it is relatively fast due to clustering on the attribute value and since most of Rl would typically be filtered The result is a set of surrogates which need to be sorted to begin a merge join This sorting prevents pipelining the select result into the join operation After the sort, however, either a 4-way parallel or a 3-stage pipeline merge join can begin using the DSM copy clustered on surrogate for Ri, Rj and Rk

5 5 Multiple Disks

The DSM allows a simple scheme to exploit multiple disks It is based on the heuristic that clustering among blocks is important for the select operator with a constrained attribute but is not important for joins on surrogate which require random access It is also based on the heuristic that placement of each copy of a DSM relation on different disks improves reliability

For example, suppose we have four attributes and four disks Each of the DSM relations clustered on attribute would be placed on a separate disk

disk1	disk2	disk3	disk4
a1	a2	a3	a4

This placement provides clustering within each attribute for efficient execution of the select operation, since head movement between blocks of an attribute value interval is minimized It also provides parallel access to different attributes for retrieval patterns with nca>1, effectively making nca=1

Each of the DSM relations clustered on surrogate would be spread across three disks in equal surrogate ranges (sri)

	disk1	disk2	disk3	disk4
a1		sr1	sr2	sr3
a2	sr3		sr1	sr2
a3	sr2	sr3		sr1
a4	sr1	sr2	sr3	

This placement provides parallel access within each attribute for efficient joins on surrogate Since the surrogate ranges are skewed so that the same range on different attributes are on separate disks, it provides parallel access of multiple attributes for efficient m-way merge joins on surrogate This is important for retrieval patterns with npa>1, effectively making npa=1 for small r, which is the case where the DSM performance is lowest It also provides increased reliability from disk failure by ensuring that the two copies of a DSM relation are on different disks

This simplified scheme assumes that the number of disks is the same as the number of attributes and that attributes are of equal size so that disks have equal space utilization These unrealistic assumptions can be relaxed at the expense of some parallelism Multiple attributes could be stored per disk for small attributes or if disks are fewer than attributes Alternatively, attributes could be spread over multiple disks for large attributes or if attributes are fewer than disks

For the NSM, some gains can be obtained by distributing the NSM relation over multiple disks This is similar to the second placement above for the DSM However, this provides no corresponding advantage to the first placement above for the DSM where each attribute is clustered on a different disk, nor are there any gains in reliability

6 SUMMARY AND FUTURE WORK

The DSM offers simplicity Simple systems have several major advantages over complex systems One advantage is that a set of fewer and simpler functions, given fixed development resources, can be either further tuned in software or pushed further into hardware to improve performance This is similar to the RISC (Patterson and Ditzel 1980) approach in general purpose architectures A second advantage is that many alternative cases with different processing strategies can less often be exploited, since the cases are not always recognized A third advantage is reduced user involvement, since less performance tuning is required by users A fourth advantage of simplicity is reliability

The DSM offers more generality with simple extensions Section 2 described how the DSM can support data models which allow multivalued attributes, entities, multiple parent relations, heterogeneous records, directed graphs and a temporal dimension with some simple extensions

The DSM offers increased physical data independence and availability

The DSM requires from 1/2 to 4 times as much total storage with a typical value of 1 4 However, it offers improved recovery from failure Also, storage requirements are not as critical in most database systems as performance and reliability

The relative update performance of the two storage models is dependent on whether attribute modifications or record inserts and deletes are more frequent and on the number of NSM inverted files In general, the DSM reduces update performance However, as RAM becomes cheaper, differential files can be used to reduce this difference

The relative retrieval performance of the two storage models is dependent on the number of attributes involved in retrievals and the size of intermediate and final results In general, the DSM requires more disk accesses for a large number of retrieval attributes and small intermediate and final results, but otherwise requires fewer disk accesses It is not clear at this time whether current or future database or knowledge base retrieval pattern mixes favor the DSM Some experience with logic programming in knowledge base applications indicates that the average number of attributes on the left side of Horn clause predicate is 2 8 (Murakami et al 1984) Statistical database applications often have a large number of attributes per record but very few attributes per retrieval (Turner et al 1979, Teitel 1977) Many of the DSM performance problems can be reduced using multiple disks Also, as RAM becomes cheaper, more of the database can be cached The DSM allows individual attributes to be cached This better utilizes cache space, since not all attributes of a conceptual schema relation have the same frequency of use Although more joins are required by the DSM, each join is faster This is again similar to the RISC approach where more instructions are generated but each is simpler and executes faster

Our retrieval performance comparison assumed the NSM was highly tuned, so that every constrained or join attribute had an inverted file index This is unlikely in reality Thus the DSM would have an advantage in an environment where workload characteristics are not static This is again similar to the RISC approach where a neutral instruction set is provided The complex instructions in CISC architectures are highly tuned to efficiently implement a certain language and OS at the expense of others

We have examined a particular type of DSM, where two copies of each DSM relation are stored, one fully clustered on the attribute and the second fully clustered on the surrogate Several other types of DSM deserve examination One might store only one copy of each DSM relation, fully clustered on the attribute value and with an inverted file on the surrogate This alternative has the advantage of reducing storage and update requirements Such a comparison would make clearer the performance impact of the second DSM copy Delta (Shibayama et al 1982) uses a DSM that stores a single copy with primary clustering on attribute values and secondary clustering on surrogates Yet another alternative is a hybrid which stores two copies, one copy is a set of DSM relations fully clustered on attribute value and the second copy is an NSM relation fully clustered on the surrogate Other alternative DSM types are also possible and deserve examination In addition, our retrieval performance model should be enhanced to directly include the effects of limited buffer space, concurrency, index searching, multiple disks, more general retrieval patterns, and other phenomena A major purpose of this report is to encourage research in these areas

Acknowledgements

Thanks to David Maier of Oregon Graduate Center and Haran Boral of MCC for their helpful comments and encouragement

References

J R Abrial, "Data Semantics," in Data, Base Management, J W Klimbie and K L Koffeman, eds , North-Holland Pub Co (1974)

D S Batory, "On Searching Transposed Files," ACM Transactions On Database Systems, Vol 4, No 4 (December 1979)

D S Batory, "Modeling The Storage Architectures Of Commercial Database Systems," University of Texas at Austin (1984)

R Bayer and E McCreight, "Organization And Maintenance Of Large Ordered Indexes," Acta Informatica, Vol 1 (1972)

R Bayer and K Unterauer, "Prefix B-Trees," ACM Transactions On Database Systems, Vol 2, No 1 (March 1977)

J L Bentley, "Multidimensional Binary Search Trees In Database Applications," IEEE Transactions On Software Engineering, Vol SE-5, No 4 (July 1979)

R Burnett and J Thomas, "Data Management Support For Statistical Data Editing," Proceedings Of The First Lawrence Berkeley Laboratory Workshop On Statistical Database Management (December 1981)

D Bitton, H Boral, D J DeWitt and W K Wilkinson, "Parallel Algorithms For The Execution Of Relational Database Operations," ACM Transactions On Database Systems, Vol 8, No 3 (September 1983)

A F Cardenas, "Analysis And Performance Of Inverted Data Base Structures," Communications Of The ACM, Vol 18, No 5 (May 1975)

J M Chang and K S Fu, "Dynamic Clustering Techniques For Physical Database Design," School of Electrical Engineering, Purdue University TR-EE 78-49 (December 1978)

D L Childs, "Extended Set Theory A General Model For Very Large, Distributed, Backend Information Systems," Proceedings Of The Third International Conference On Very Large Databases (October 1977)

E F Codd, "Extending The Data Base Relational Model To Capture More Meaning," ACM Transactions On Database Systems, Vol 4, No 4 (December 1979)

G P Copeland and D Maier, "Making Smalltalk A Database System," Proceedings Of The ACM SIGMOD Conference, SIGMOD Record, Vol 14, No 2 (June 1984)

A Deliyanni and R A Kowalski, "Logic And Semantic Networks," Proceedings of the Workshop On Logic And Data Bases, Toulouse (November 1977)

S J Eggers, F Olken and A Shoshani, "A Compression Technique For Large Statistical Databases," Proceedings Of The Seventh International Conference On Very Large Databases (September 1981)

R Fagin, "Multivalued Dependencies And A New Normal Form For Relational Databases," ACM Transactions On Database Systems, Vol 2, No 3 (September 1977)

A Goldberg and D Robson, Smalltalk-80 The Language And Its Implementation, Addison-Wesley Pub Co (1983)

A J Hoffer, "An Integer Programming Formulation Of Computer Data Base Design Problems," Information Sciences 11 (1976)

R A Kowalski, "Logic For Data Description," in Logic And Data Bases, H Gallaire and J Minker (eds), Plenum Press, New York (1978)

R A Lorie and A J Symonds, "A Relational Access Method For Interactive Applications," Data Base Systems, Courant Computer Science Symposia, Vol 6, Prentice-Hall (1971)

S T March and D G Severance, "The Determination Of Efficient Record Segmentations And Blocking Factors For Shared Data Files," ACM Transactions On Database Systems, Vol 2, No 3 (September 1977)

S T March and G D Scudder, 'On The Selection Of Efficient Record Segmentations And Backup Strategies For Large Shared Files," ACM Transactions On Database Systems, Vol 9, No 3 (September 1984)

K Murakami, T Kakuta and R Onai, "Architectures And Hardware Systems Parallel Inference Machine And Knowledge Base Machine," Proceedings Of The International Conference On Fifth Generation Computer Systems, Tokyo (November 1984)

J Nievergelt, H Hinterberger and K C Sevcik, "The Grid File An Adaptable, Symmetric Multikey File Structure," ACM Transactions On Database Systems, Vol 9, No 1 (March 1984)

D Patterson and D Ditzel, "The Case For The Reduced Instruction Set Computer," Computer Architecture News, Vol 8, No 6, ACM (October 1980)

D C Severance and C M Lohman, "Differential Files Their Application To The Maintenance Of Large Databases," ACM Transactions On Database Systems, Vol 1, No 3 (September 1976)

S Shibayama, T Kakuta, N Miyazaki, H Yokota and K Murakami, A Relational Database Machine With Large Semiconductor Disk And Hardware Relational Algebra Processor," New Generation Computing Vol 2 (1984)

A Shoshani, F Olken and H K T Wong, "Characteristics Of Scientific Databases," Lawrence Berkeley Laboratory, LBL-17582 (1982)

J M Smith and D C P Smith, "Principles Of Database Conceptual Design," Proceedings Of The NYU Symposium On Database Design, New York (May 1978)

Y Tanaka, "A Data-Stream Database Machine With Large Capacity," in Advanced Database Machine Architectures, D K Hsiao, ed , Prentice-Hall (1983)

R F Teitel, "Relational Database Models and Social Science Computing," Proceedings Of Computer Science And Statistics Tenth Annual Symposium On The Interface, Gaithersburg, Maryland, National Bureau Of Standards (April 1977)

M J Turner R Hammond and F Cotton, "A DBMS For Large Statistical Databases," Proceedings Of The Fifth International Conference On Very Large Databases (October 1979)

G Wiederhold, J F Fries and S Weyl, "Structured Organization Of Clinical Data Bases," Proceedings Of The National Computer Conference, AFIPS Press (May 1975)