Biorthogonally accessed three-dimensional two-photon memory for relational database operations

B. H. Olson, R. Paturi, and S. C. Esener

Memory bandwidth is a bottleneck for very large database machines. Parallel-access three-dimensional two-photon memories have the potential of achieving enormous throughput (>100 Gbit/s) and capacity (1 Tbit/cm^3) [Appl. Opt. **29,** 2058 (1990)] and, consequently, are well suited for this application. Our analysis shows that some operations can be completed more than 2 orders of magnitude faster with this type of memory than with a system based on serial-access storage. These particular memories have a further feature of being accessible in orthogonal directions. We show that this property, used in conjunction with a three-dimensional data-organization scheme designed for this approach, leads to improved performance by permitting the user a choice of accessing strategies for a given operation. © 1997 Optical Society of America

Key words: Parallel-access optical memory, 3-D data organization, relational database, biorthogonal access.

1. Introduction

As databases grow in size, they require increased memory bandwidth and parallel processing to complete queries in a reasonable amount of time. Currently very large database machines achieve their capacity and throughput by making use of an array of serial-access disks and processors.[1,2] The bandwidth of these storage systems can be expanded if the memories in the array are accessed in parallel. Mitkas and Berra[3] conclude that a relational database machine with a single parallel-access optical disk and a parallel optoelectronic processor has the potential of outperforming a relational database system based on an array of 30 serial-access disks, each with its own processor. However, parallel-access single-layer optical disks will not achieve capacities much greater than those of serial-access optical disks.

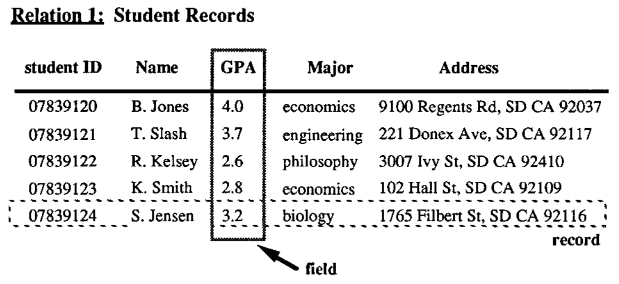
Three-dimensional (3-D) parallel-access optical memories are capable of achieving tremendous capacity in addition to throughput by storing data throughout the volume of the memory device as opposed to storing data on the surface. Recent advances in both volume holographic[4] and 3-D two-photon[5] page-access optical memories show great promise for these high-capacity–high-bandwidth storage devices. Mitkas and Irakliotis[6] suggest their use in relational database systems.

In relational database machines the performance of operations depends not only on the memory bandwidth but also on how the data is positioned and, consequently, accessed from the memory. The optimal data organization–access strategy for one database operation is not always best for another. A further property of the 3-D two-photon memory is that pages of data can be retrieved from it in a number of directions, potentially allowing for a system in which a user can use different memory-access approaches for different operations. Recently it has been suggested that accessing a 3-D two-photon memory in two orthogonal directions might be beneficial for relational database operations.[7] In this paper we propose a particular 3-D data-organization strategy for a relational database system to take full advantage of this accessing approach and to evaluate its performance as a function of system parameters by use of the Wisconsin benchmark.[8]

Our analysis shows that 3-D parallel-access optical

B. H. Olson is with the Jet Propulsion Laboratories, California Institute of Technology, Mail Stop 300-315, 4800 Oak Grove Drive, Pasadena, California 91109-8099. R. Paturi and S. C. Esener are with the Department of Electrical and Computer Engineering, University of California, San Diego, California 92093.

Received 13 May 1996; revised manuscript received 30 January 1997.
0003-6935/97/173877-12$10.00/0


Relation 1: Student Records

student ID	Name	GPA	Major	Address
07839120	B. Jones	4.0	economics	9100 Regents Rd, SD CA 92037
07839121	T. Slash	3.7	engineering	221 Donex Ave, SD CA 92117
07839122	R. Kelsey	2.6	philosophy	3007 Ivy St, SD CA 92410
07839123	K. Smith	2.8	economics	102 Hall St, SD CA 92109
07839124	S. Jensen	3.2	biology	1765 Filbert St, SD CA 92116

record
field

Fig. 1. Example of a relation that consists of student records. The rows of the table are termed records or tuples. The columns are fields or attributes.

Selection Example: Which students have a GPA of 3.0 or higher?

student ID	Name	GPA	Major	Address
07839120	B. Jones	4.0	economics	9100 Regents Rd, SD CA 92037
07839121	T. Slash	3.7	engineering	221 Donex Ave, SD CA 92117
07839124	S. Jensen	3.2	biology	1765 Filbert St, SD CA 92116

Fig. 2. Relation resulting from the selection query STUDENT RECORDS(GPA $\geq$ 3.0). The operations $<$, $>$, $\geq$, $\leq$, $=$, and Boolean complement are all valid selection operations.

memories, in general, are well suited for large relational database systems with low write requirements because of their potential capacity and throughput. The biorthogonal accessing approach used in conjunction with the proposed data-organization strategy is found to be additionally advantageous. The time required to retrieve data for a projection operation, for example, is close to the optimum value. This is not likely to be the case for a system with traditional page access that employs a data-organization strategy optimized for another operation.

We begin this paper with an introduction of relational database operations and their memory-accessing requirements. Following this we give a brief description of the subset of data-filtering operations from the Wisconsin benchmark, which we used to evaluate performance. In Section 2, we provide a description of the biorthogonal memory itself and the proposed 3-D data-organization strategy. We also identify problems such as memory fragmentation that can lead to degraded performance in real systems. In Section 3, we analyze the potential performance of a biorthogonally accessed 3-D two-photon memory, first considering the performance in an ideal system and then introducing the effect of the problems described in Section 2. In Section 4, we compare the potential performance of a 3-D two-photon memory with that obtained with other proposed or existing relational database systems.

2. Operation and System Description

A. Relational Database Operations

In relational databases, the data records, also called *tuples*, are grouped into sets called *relations*. All records that belong to a relation must be unique. Figure 1 depicts a relation comprising student records. The columns are termed fields or attributes, and the rows are the records. Relational database operations can be categorized into two groups: set operations and relation-oriented operations.[9] Some of these operations are performed on data from a single relation. Others combine data from several relations. The result of an operation is always a relation.

Biorthogonal access allows data to be retrieved more effectively for relational database operations. Data can be arranged so that certain fields can be retrieved by use of one access direction, as shown in Fig. 1. In the other access direction a subset of records can be retrieved. These two particular accessing features, which we term field-parallel access and record-parallel access, are particularly beneficial for relational database operations because they allow a user to isolate better the data desired for a given operation. We explore this further as we give examples of some relational database operations.

The relations in a database can be thought of as sets and their records as elements of those sets. The set operations are the traditional set operations: union, difference, and Cartesian product. These are not described here. Descriptions can be found in a standard database text.[9] The basic relation-oriented operations are selection, projection, and join.

The selection operation, shown in Fig. 2, is performed on data from a single relation. It involves choosing records from a relation on the basis of the value of some subset of fields, referred to as the operand. The operations less than ($<$), greater than ($>$), greater than or equal to ($\geq$), less than or equal to ($\leq$), equal to ($=$), and Boolean complement are all valid selection operations. To perform this operation, one must identify and retrieve records satisfying the query. Frequently this can involve retrieving and examining all records, which can take an enormous amount of time in large databases. If only a few records satisfy the selection query, the amount of data that needs to be retrieved for this operation can be reduced significantly if the operand is retrieved with field-parallel access and the records determined to satisfy the selection query are retrieved with record-parallel access. In some instances, the value of the selection operand may correspond directly to its memory address. In this case record-parallel access might be preferable.

With a projection operation, shown in Fig. 3, all the data from a relation that belong to one or more fields are extracted to form a new relation with fewer fields. True projection also involves duplicate removal; otherwise, the result would not be a relation. Once again, if the relation is large, the process of extracting the data for this operation can be quite time consuming. If record-parallel access were used to perform this operation, all the data in the relation would have to be retrieved. If field-parallel access were used instead, the amount of data that would need to be retrieved could be reduced significantly because the desired subset of fields could be obtained directly.

Projection Example: **What Majors do students have?**

economics		economics
engineering	**duplicate removal** $\longrightarrow$	engineering
philosophy		philosophy
economics		biology
biology		

Fig. 3. Relation resulting from the projection operation STUDENT RECORDS(Major). With projection, a single field or several fields are extracted from a relation. Duplicate removal is sometimes necessary to guarantee that records in the resulting relation are unique.

The join operation combines relations. To illustrate joining we introduce a second relation, depicted in Fig. 4, that contains records that correspond to companies that might hire students. The result of a join operation (Fig. 5) is a subset of the Cartesian product of the two relations. Only those elements in the Cartesian product that satisfy the join query are in the join result. As with selection, $<$, $>$, $\geq$, $\leq$, $=$, and Boolean complement are all valid join operations.

Given the input–output and computational demands of the above operations, several approaches can be used to improve performance. Operations executed by a relational database machine are frequently complex, that is, they involve several basic operations. Selecting student–company pairs from the previous join example in which companies have positions open is an example of a complex operation. Significant performance improvement can be gained if operations are reordered in complex operations.[10] In the above example, performing the selection operation first would reduce the complexity of the join.

Another way to improve performance is to execute an operation in parallel. In addition to parallel hardware, use of this approach requires increased memory bandwidth. In current systems this bandwidth is achieved with an array of serial disks, but it can also be achieved with parallel-access optical storage.

B. Database Data Filter

Performance can also be gained if one filters out data deemed irrelevant to a particular query immediately, as it is retrieved from secondary storage. A device that provides this function is referred to as a data filter. This study assumes that an optoelectronic data filter is used in conjunction with the 3-D two-photon memory. A number of optoelectronic database data filters have been proposed.[3,11,12] The data filter considered in this study is assumed to support operations that can easily be done on the fly[13]: selection, projection, and selection–projection, all without duplicate removal. For further improvement of performance, some data filters transform or reorganize the retrieved data in addition to reducing it. The complexity of, for example, an equijoin operation (a join in which the comparison operation checks for equality) can be reduced if the two relations are hashed on the field over which they are being joined.[14] Data transformation and reorganization were not considered in this study. Filters that provide this function could, however, be added.

The data filter that was assumed in this study is shown in Fig. 6. It consists of a compander[15] that is flip-chip bonded onto an electronic chip capable of performing comparison and data-masking operations in parallel. The compander is a modified CCD array that is used to detect the data from the memory and to act as an interface between the data and the electronic processing chip.

Relation 2: **Companies hiring students**

Company	Number of positions open	Target Major
ABC investments	0	economics
Advanced Technology	0	engineering
XYZ BioTech	0	biology
American Banking	0	economics
Burger King	1	philosophy

Fig. 4. Relation containing companies potentially hiring students.

Join Example:

Which students should be paired with which companies based on their majors?

student ID	Name	GPA	Major	Number of positions	Company
07839120	B. Jones	4.0	economics	0	ABC investments
07839121	T. Slash	3.7	engineering	0	Advanced Technology
07839123	K. Smith	2.8	economics	0	ABC investments
07839124	S. Jensen	3.2	biology	0	XYZ BioTech
07839120	B. Jones	4.0	economics	0	American Banking
07839123	K. Smith	2.8	economics	0	American Banking
07839122	R. Kelsey	2.6	philosophy	1	Burger King

Fig. 5. Example of a join operation. Company records are appended to student records if the student's major equals the company's target major.

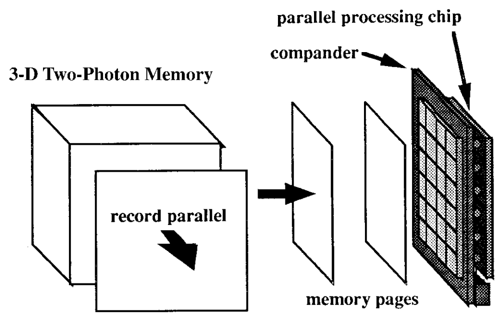


Fig. 6. Three-dimensional optical memory with an optoelectronic database data filter. The data filter consists of a compander that is flip-chip bonded onto an electronic processing chip. The compander is a modified CCD array that detects pages retrieved from the memory and acts as an interface between these data and the electronic processing chip.

C. Benchmark Description

Numerous benchmarks have been proposed to evaluate database systems.[16] We rely on the Wisconsin benchmark because it is commonly used to evaluate the performance of parallel relational database machines, including optoelectronic ones. For clarity, portions of the benchmark that were used are briefly described. The benchmark allows one to evaluate databases of different sizes by scaling the sizes of relations. The operations considered in this study are performed on same-size relations. However, other relations would exist in the memory at the same time.

The performance of the selection operation is examined for different selectivities. The *selectivity* of an operation refers to the number of records that satisfy a selection query. A relative selectivity of 10% means that 10% of the records in the relation satisfy the query. An absolute selectivity of 100 records means that 100 records satisfy the query regardless of the relation's size.

The Wisconsin benchmark also requires that the performance of selection operations be measured with different types of indexing as well as with no indexing. A relation with an indexed field is organized in some way (B-tree, hashing, etc.) according to the value of a subset of fields; an index key is assigned to each record on the basis of this value. With *clustered* indexing, the index key determines the physical location of the data. With *nonclustered* indexing, the index key is used to determine the address of a subset of records. Nonclustered indexing is not always preferable to no indexing. Insertion of records has a much higher cost with indexing. Also, for selection operations with moderately high selectivity (1–10%), a crude scan of all the records (selection with no indexing) is sometimes comparable in performance or even preferable to nonclustered indexing.[8] The Wisconsin benchmark considers the performance of the different types of selection operations for a discrete set of selectivities: an absolute selectivity of 1 and relative selectivities of 1% and 10%. It does not require selection with nonclustered indexing or no indexing for an absolute selectivity of 1. It also does not require selection with nonclustered indexing for a selectivity of 10%.

In the sections that follow, we examine the performances of various selection and projection operations for a biorthogonally accessed 3-D optical memory system. The specific operations considered in this study are the same as those in the Wisconsin benchmark except for the selectivity in selection operations and the operand size in the projection operation, both of which were varied continuously instead of being assigned discrete values. First, however, we briefly describe the memory and the proposed data-organization scheme, defining terms that are used for the duration of the paper.

D. Biorthogonally Accessed Three-Dimensional Two-Photon Memory

A parallel-access 3-D two-photon memory is a random-access memory in which bits are suspended throughout the volume of a cube or a rectangular solid, each bit occupying a unique physical location. Data are written through two-photon excitation by the intersection of two laser beams of different wavelengths at any location within the memory. An entire page of bits can be written by intersection of a spatially modulated information beam with an addressing beam,[6] as shown in Fig. 7. A page written in this way can be retrieved by its reillumination with an identically positioned addressing beam. Written bits when illuminated in this way fluoresce, and this fluorescence can be detected. The potential capacity of these memories is high (1 Tbit/cm^3)[17] because bits are stored in a volume as opposed to on a planar surface. Throughput is also high (>100 Gbit/s)[17] because the pages themselves can contain 10^6 bits and be accessed in a relatively short length of time. So far 100 data planes separated by 80 μm have been recorded and retrieved in such memory cubes with bit-error rates lower than 10^{-8}. Orthogonal addressing of the data leads to low-cost replayer sytems, such as the one shown in Fig. 8(a). By adding a second orthogonal readout system, we can easily extend this replayer to access the stored data in orthogonal directions, as shown in Fig. 8(b). We define the ability to retrieve planes in two orthogonal directions as *biorthogonal* access.

E. Memory Mapping

1. *Record-Parallel and Field-Parallel Access*

The proposed data-organization scheme is explained with reference to Fig. 8. Complete records are mapped onto the memory so that they are contained in a single (y–z) plane. These records are arranged on this plane such that data that belong to the same subset of fields are contained on the same x–y plane. Retrieving a y–z page that contains complete records results in record-parallel access. Retrieving an x–y page results in field-parallel access. With record-parallel access, a small set of records can be retrieved in parallel in one memory read. With field-parallel access, a subset of fields can be retrieved.

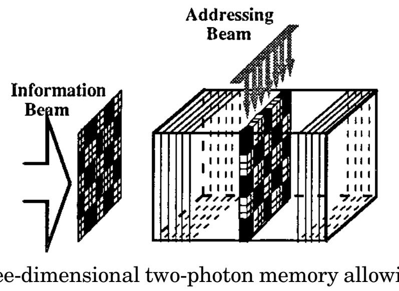
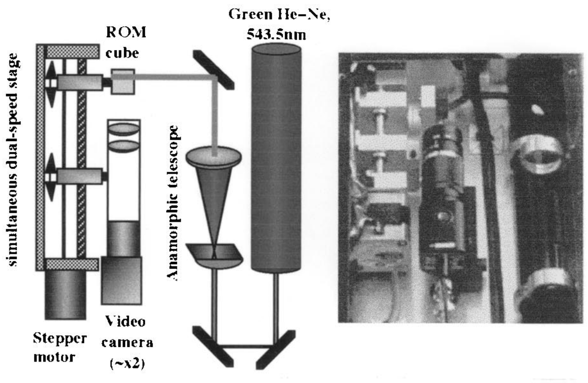
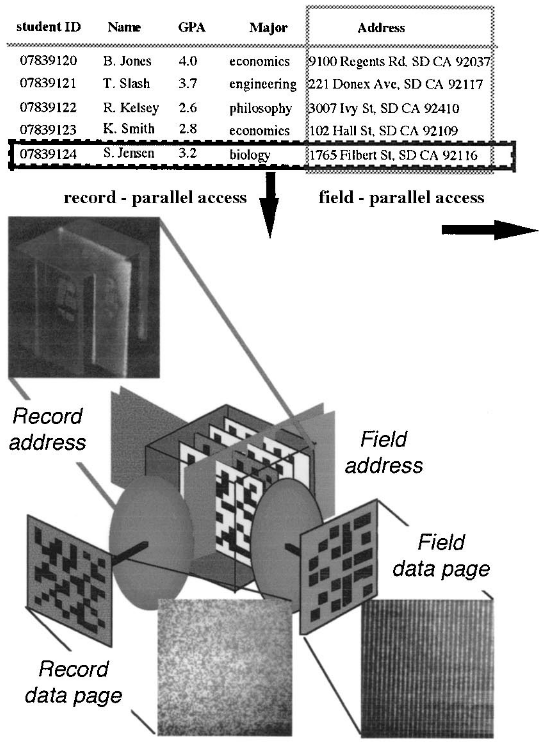


Fig. 7. Three-dimensional two-photon memory allowing a page of bits to be written in parallel. A desired page is retrieved by reillumination of the plane where it was stored with another beam of light. This figure is taken from Ref. 17.



(a)



student ID	Name	GPA	Major	Address
07839120	B. Jones	4.0	economics	9100 Regents Rd. SD CA 92037
07839121	T. Slash	3.7	engineering	221 Donex Ave. SD CA 92117
07839122	R. Kelsey	2.6	philosophy	3007 Ivy St, SD CA 92410
07839123	K. Smith	2.8	economics	102 Hall St, SD CA 92109
07839124	S. Jensen	3.2	biology	1765 Filbert St, SD CA 92116

(b)

Fig. 8. (a) Schematic diagram and photograph of a two-photon ROM demonstration replayer unit based on orthogonal addressing (courtesy of Call/Recall, Inc.). (b) Data organized in a biorthogonally accessed 3-D two-photon memory such that complete records are mapped onto y–z planes. The records are arranged on these planes such that data belonging to the same field are on the same x–y plane. Accessing a y–z plane in parallel is referred to as record-parallel access. Accessing an x–y plane is referred to as field-parallel access.

We can improve performance by choosing the best accessing method for a given operation. Usually projection is best accomplished with field-parallel access so that only the desired subset of fields needs to be retrieved. This readout mode is also advantageous for certain selection operations because it provides an efficient means of scanning through a set of records to determine which records satisfy a particular selection query; only the operand on which the selection operation is based needs to be retrieved. If it is determined that a record satisfies the selection query, it can be retrieved in one page read by use of record-parallel access. Reading out a single record by use of field-parallel access would require many more page reads. Certain selection operations require that a single record or a small subset of records be retrieved. This would be performed well with record-parallel access. In the sections that follow, the effect that the two accessing approaches have on performance is examined. At this point we describe some terms that are used throughout the paper.

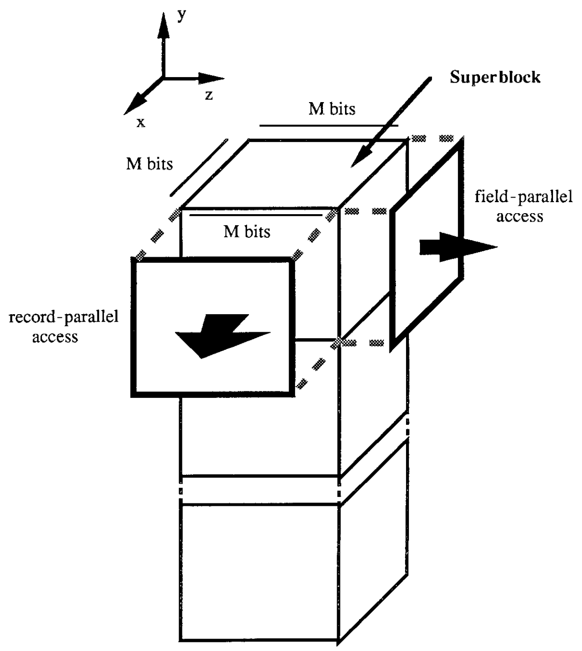


Fig. 9. Memory comprising bit cubes called superblocks. A superblock can be viewed as a sequence of pages that can be accessed randomly in either of two orthogonal directions, from any superblock, in time t. Each superblock contains M^3 bits with M bits on a side; thus pages read from these superblocks contain M^2 bits.

2. *Data Organization*

In this paper we assume that the memory comprises a number of bit cubes that we refer to as superblocks, as shown in Fig. 9. A superblock can be viewed as a sequence of pages that can be accessed randomly in either of two orthogonal directions. Each superblock contains M^3 bits and has M bits on a side; thus pages read from these superblocks contain M^2 bits. The time required to access a page is t.

The data-organization scheme that we propose is illustrated in Fig. 10. Records are arranged on y–z planes so that each x–y plane contains w bits of a record. It is assumed that w divides M. The parameter w affects both capacity and performance, and its effect is examined throughout the paper. For example, it affects the time required to retrieve a record with field-parallel access; approximately r/w page reads are required to retrieve a record of size r. We refer to the set of r/w x–y pages needed to contain a complete record as a block.

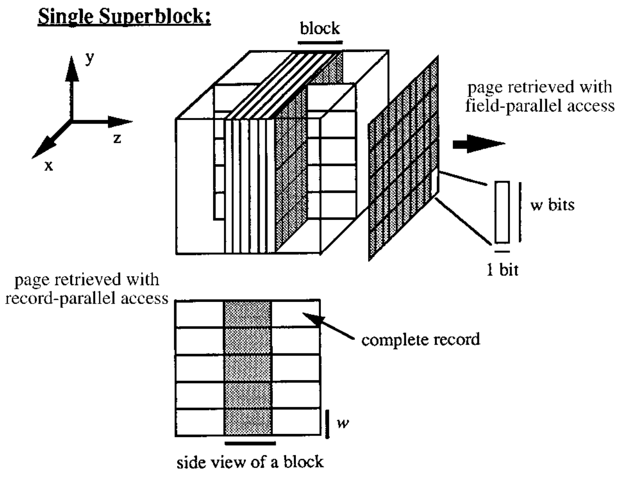

Single Superblock:
block
y
z
x
page retrieved with field-parallel access
w bits
1 bit
page retrieved with record-parallel access
complete record
w
side view of a block

Fig. 10. Records mapped onto the memory such that they are contained on single y–z planes. The data on these planes are arranged so that each page accessed in the field-parallel direction contains w bits of a record. A complete record of size r can be accessed either in one memory read with record-parallel access or in approximately r/w page reads with field-parallel access. This set of x–y pages is referred to as a block.

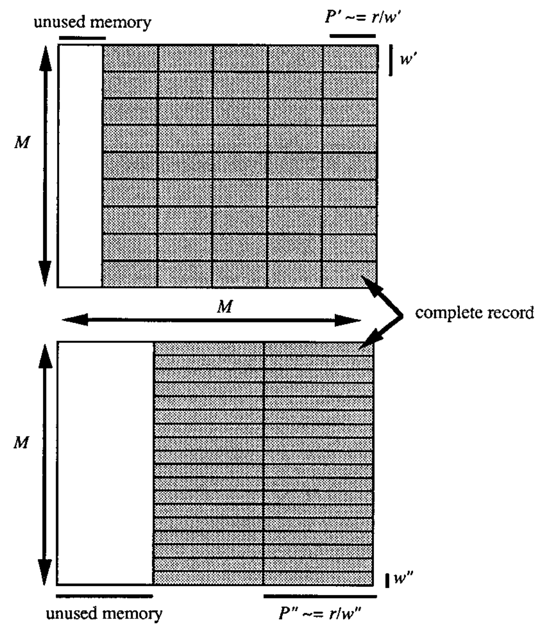


Fig. 12. Memory fragmentation that reduces capacity and bit-retrieval efficiency with record-parallel access. Two pages retrieved with record-parallel access are shown. w' and w'' are chosen so that they divide M. It is likely that P' and P'' will not divide M. As a result, some memory on each record-parallel page will not be used, and entire field-parallel pages that are perpendicular to this will be empty as well. The capacity penalty that results from this, on average, increases when w is reduced.

3. Fragmentation

With the proposed data-organization scheme, there will sometimes be gaps in the memory that contain no data. This memory-fragmentation problem will affect capacity and also performance because pages retrieved from the memory will not always be full. Gaps can occur in two ways and are dependent on the value of w. It is assumed that data from multiple fields can be contained on the same field-parallel page and that the data in each group of w bits may belong to different records. The first type of gap that we consider is illustrated in Fig. 11. This particular kind of memory fragmentation is likely to have little effect on the parameters that were considered in this study. For example, less than 1% wastage would occur with w equal to 2^4 bits.

Another kind of memory fragmentation is more significant. It is further assumed that a record can always be retrieved with one memory read. The problem that can arise is explained with reference to Fig. 12. This figure shows two pages retrieved with record-parallel access, each page employing a different record-placement strategy, i.e., different values for w. In Fig. 12 w' and w'' are chosen so that they divide M. However, P' and P'', which are approximately r/w' and r/w'', respectively, do not divide M. As a result, some memory on each record-parallel page will not be used. Correspondingly, entire field-parallel pages that are perpendicular to this record-parallel page will be empty as well. On average there will be $P/2$ such pages per superblock, where $P = r/w$ is the number of pages in a block. The related capacity penalty, on average, increases when w is reduced. This fragmentation problem also affects data retrieval by use of record-parallel access; with field-parallel access, the empty pages could be avoided. We consider the effect of this type of memory fragmentation in Subsection 3.E. In our analysis we use the parameter α to represent the data-retrieval efficiency arising from memory fragmentation when record-parallel access is used. In Fig. 12 it is the ratio of the area of the shaded region over the area of the entire page. This can be expressed mathematically as

$$\alpha = \left(\frac{r}{wM}\right)\left\lfloor\frac{Mw}{r}\right\rfloor. \tag{1}$$

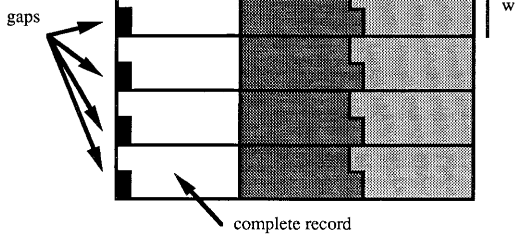


Fig. 11. Page retrieved with record-parallel access showing gaps leading to memory fragmentation.

In this expression and those that follow, the brackets $\lfloor\ldots\rfloor$ denote the integer less than or equal to the

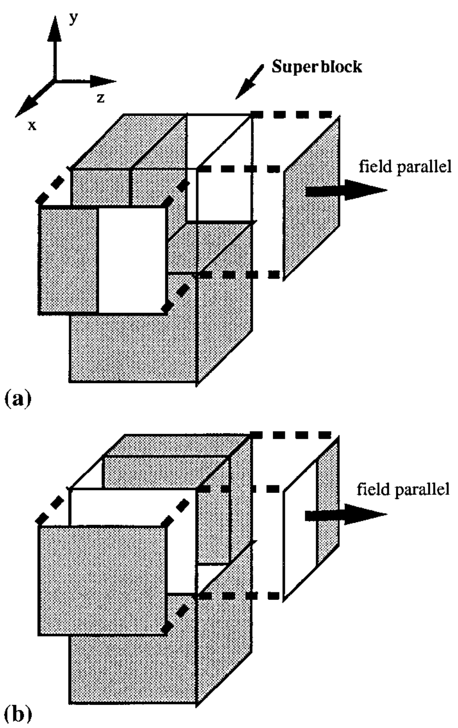


Fig. 13. Packing: (a) Field-parallel pages packed first. Some pages retrieved with record-parallel access will contain data from other relations. (b) Record-parallel pages are packed first. Field-parallel access can be adversely affected.

operand, and the brackets $\lceil \ldots \rceil$ denote the integer greater than or equal to the operand in brackets.

4. *Packing*

Another factor that can have an even greater impact on performance is packing. A relation may not completely fill all the superblocks in which it is contained; it is also likely to start and end in the middle of a superblock. As a result, the first and last superblocks will usually contain other relations or be empty. Records residing in these partially filled superblocks can be placed such that they fill field- or record-parallel pages first. This process is shown in Fig. 13. In general if field-parallel pages are filled first, the time required to perform operations with record-parallel access will increase by a factor of approximately $1/B$, where B is the minimum number of superblocks required for the operation. This is because, on average, two additional half-superblocks of data that do not contain the relation of interest will have to be read. The time required to perform operations with field-parallel access is not affected. If, on the other hand, record-parallel pages are filled first, the record-parallel access time is not affected, but the field-parallel access time is increased by a factor of $1/B$. The effect of adverse packing decreases for larger relations but can be significant for small relations when there are few superblocks and B is small. We also examine, in Subsection 3.F, the effect of data packing on performance.

Table 1. Variables Used for Performance Calculations

Parameter	Description	Value or Expression
R	Number of records in relation	10^6
r	Record size	(208×8) bits
M^2	Memory page size	1024^2 bits
M	Number of memory pages in a superblock	1024
t	Time to access a memory page	10 μs
w	Field-parallel word size	Variable
α	Page-retrieval efficiency due to fragmentation of type II	$r/Mw \lfloor Mw/r \rfloor$
B	Minimum number of superblocks required to store the relation	$\lceil rR/\alpha M^3 \rceil \cong 2$
N	Number of records satisfying the selection query	Variable
f	Operand size for selection operation	32 bits
p	Operand size for projection operation	Variable

3. Performance Study

A. Performance Study Parameters

In this section we analyze the potential performance of a biorthogonally accessed 3-D optical memory for a subset of relational database operations commonly used in data filters. The particular operations that were considered were projection and selection with and without indexing. We begin by looking first at the effect that w and the accessing method have on performance in an ideal situation, neglecting the effects of memory fragmentation and packing. We consider the effects that these two factors can have below.

For clarity the parameters and values that are used in the expressions and graphs in this section and in Section 4 are summarized in Table 1. When numerical results are given, it is assumed that the relation contains 10^6 records. This particular relation size was chosen to facilitate comparison with other existing or proposed relational database machines. A single two-photon memory could hold a much larger-sized relation.

B. Ideal Projection

The projection operation requires that one or more fields be extracted from a relation to form a new relation. This operation also requires duplicate removal. In our analysis we considered only the time that would be required to retrieve the data for the operation, not the time for duplicate removal, because that would have required the analysis of a complete database system. The times given in this section for projection, therefore, cannot be compared with times obtained for other systems that use the Wisconsin benchmark because the benchmark requires duplicate removal.

The projection operation is almost always performed best with field-parallel access because only the desired subset of fields needs to be retrieved.

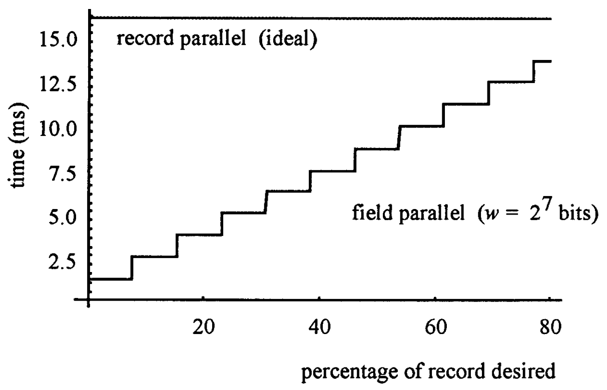


Fig. 14. Time to retrieve the data required for a projection operation. With field-parallel access only, the desired subset of fields needs to be retrieved. With record-parallel access, the entire relation needs to be retrieved. This graph neglects the effects of packing and memory fragmentation.

With record-parallel access the entire relation always needs to be retrieved. The time required to retrieve the data necessary for a projection operation with record-parallel access is given below. It is simply the time required to retrieve a single page, t, multiplied by the number of record-parallel pages needed to store the relation:

$$t\left\lceil \frac{Rr}{M^2} \right\rceil. \tag{2}$$

The time required to retrieve the data by use of field-parallel access is given in expression (3), if the desired subset of fields is assumed to be p bits. The first two terms represent the time required to retrieve the p bits from a single block in the memory. The last term is the number of blocks needed for the relation:

$$t\left\lceil \frac{p}{w} \right\rceil \left\lceil \frac{Rw}{M^2} \right\rceil. \tag{3}$$

The time required to retrieve the data for a projection operation with the two accessing methods is plotted in Fig. 14 as a function of the operand size. The operand size is augmented in byte increments, even though this could physically correspond to reading out fractions of fields. The superior performance of field-parallel access can be seen clearly. For small operand sizes and small values of w, the time required to retrieve the data with field-parallel access can require a factor of 50 less time than does record-parallel access. The jagged appearance of the field-parallel trace is a result of the fact that the operand size p is not always a multiple of w bits. The time required to perform this operation with field-parallel access is minimized when p is a multiple of w, which is more likely to occur when w is small.

C. Ideal Selection with Clustered Indexing

The performance of selection with clustered indexing was also considered. With this operation the location of the records is known *a priori*, and the records that satisfy the selection criterion are assumed to be adjacent. Two different data-ordering schemes are assumed. With record-parallel access, consecutive records are on the same record-parallel page, but with field-parallel access, consecutive records are on the same field-parallel page. The Wisconsin benchmark requires that the time required to return and format the result to the user be included for this particular selection operation. This overhead is not included in this analysis—only the time required to retrieve the desired records is included.

In principle it is best to perform this operation with record-parallel access. In practice, when one includes the effects of memory fragmentation and packing, this may not be the case. The time to perform this operation with record-parallel access is given below, under the assumption that N is the number of records that satisfies the selection query. The time is approximately equal to the page access time, t, multiplied by N, and divided by the number of records on a page ($\sim M^2/r$). There is a good chance that the set of consecutive selected records will start and end in the middle of a record-parallel page and that an additional page read will be required. This is taken into consideration in expression (4):

$$t\left[(N-1)\left\lfloor \frac{M^2}{r} \right\rfloor^{-1} + 1\right]. \tag{4}$$

The expression for the time required to perform this operation with field-parallel access is similar and is given below. With field-parallel access, r/w page reads are required to retrieve a complete record. This set of pages, as was mentioned in Subsection 2.E.2, is referred to as a block. The time to perform this operation with this form of access is roughly equal to the amount of time required to read out a block, tr/w, multiplied by the number of blocks that would be needed to contain the N selected records: Nw/M^2. Expression (5) also includes the probability that an additional block may need to be retrieved:

$$t\left(\frac{r}{w}\right)\left[\frac{(N-1)w}{M^2} + 1\right]. \tag{5}$$

The time required to perform selection with no indexing is plotted in Fig. 15 as a function of relative selectivity for the two accessing modes. As mentioned above, the set of contiguous selected records occupies a certain minimum number of pages in the case of record-parallel access or blocks in the case of field-parallel access and is likely to start and end in the middle of a page or block, thereby requiring an additional page or block read. The overhead of having to read an additional block with field-parallel access is greater than the overhead of retrieving an additional page with record-parallel access. Thus, under ideal conditions this operation is performed best with record-parallel access. With field-parallel access, the overhead is larger when there are more pages in a block (when w is small). Performing very-low-selectivity operations, such as retrieving a single

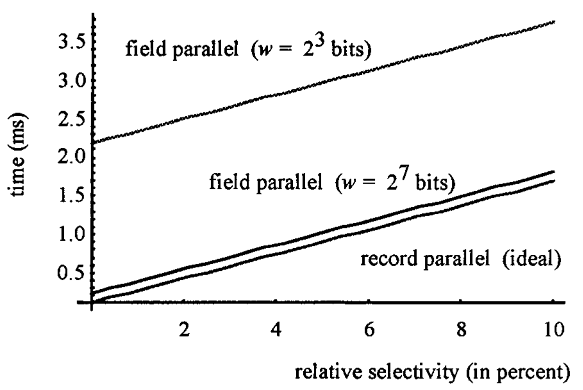


Fig. 15. Time to perform selection with clustered indexing. The effects of packing and memory fragmentation have been neglected.

record, is also performed best with record-parallel access. With this strategy only one page read is necessary; with field-parallel access an entire block would have to be retrieved.

D. Ideal Selection with No Indexing

The performance of selection with no indexing was also considered. With this operation the locations of the records that satisfy the selection query are not known *a priori*, and all records have to be searched to determine which satisfy the selection query. Records that satisfy the query would then be retrieved. It is assumed that selected records are uniformly distributed throughout the memory.

Ideally the accessing method that should be used for this operation would depend on the selectivity of the operation. Very-low-selectivity operations should be performed by searching with field parallel access and retrieving selected records with record parallel access. With moderately high-selectivity operations, all pages have to be retrieved, so either field- or record-parallel access could be used.

In general, if this operation were performed with record-parallel access, all record-parallel pages that contained the relation would have to be retrieved, so that each record could be examined to determine if it satisfied the selection query. Thus the time required to perform this operation with record-parallel access is the same for all selectivities and is the same as the time to retrieve records for a projection operation with record-parallel access.

This operation can also be performed with field-parallel access or by the combination of field- and record-parallel access. The first accessing approach is termed field–field and the latter field–record. With the two approaches, the search part of the operation is performed with field-parallel access; only the operand that contains field(s) is retrieved. Records that satisfy the selection query are read out with record-parallel access (for the field–record approach) or with field-parallel access (for field–field). The search part of the operation requires time T_{search}; an expression is given in Eq. (8) and is tabulated for different values of w in Table 2. It is assumed that

Table 2. Effect of w on T_{search}[a]

w (bits)	T_{search} (ms)
2^3	0.32
2^5	0.31
2^6	0.62
2^7	1.23

[a]T_{search} increases when w is larger than f. In this example, f is 32 bits.

$\lceil f/w \rceil$, where f is the size of the selection operand, field-parallel page reads are needed for every block. (There is a probability that a block could require one more page be read than this.) Searching is inefficient when w is a much larger than f. The time required to retrieve the selected records is referred to as the readout time.

The expected value of the time required to read out records that satisfy the selection query is denoted as T_{ror} if record-parallel access is used and as T_{rof} if field-parallel access is used. With this notation the expression for the average time to perform ideal selection with no indexing and field–record access is given below, with the assumption that N records satisfy the selection query:

$$T_{\text{search}} + T_{\text{ror}}, \tag{6}$$

where

$$T_{\text{ror}} = t\left\lceil \frac{R}{\lfloor M^2/r \rfloor} \right\rceil \left[1 - \left(1 - \left\lceil \frac{R}{\lfloor M^2/r \rfloor} \right\rceil^{-1} \right)^N \right], \tag{7}$$

$$T_{\text{search}} = t\left\lceil \frac{f}{w} \right\rceil \left\lceil \frac{Rw}{M^2} \right\rceil. \tag{8}$$

The expression for T_{ror} is simply the probability that a page has a record of interest multiplied by the number of pages required for the relation. In Eq. (7) for T_{ror}, the quantity raised to the Nth power is the probability that a page has no records of interest, given that N records are selected. The probability that a page contains at least one selected record and needs to be read out is 1 minus this quantity raised to the Nth power. It should be noted that the selection operand is read out twice with this approach during both the searching and the readout portions of the operation. The first two terms in the expression for T_{search} represent the time required to read out the operand from a single block. The second term is the number of blocks in the memory.

The average time needed to perform selection with no indexing by use of field–field access is given below. The expression for T_{rof} is similar to the expression for T_{ror}: it is the probability of a block containing a record of interest, multiplied by the number of blocks needed for the relation, then multiplied by the number of pages in a block. The quantity raised to the Nth power is the probability that a block is empty given that N records are selected. The probability that a block has a selected record on it and needs to be retrieved is 1 minus this quantity raised to the Nth

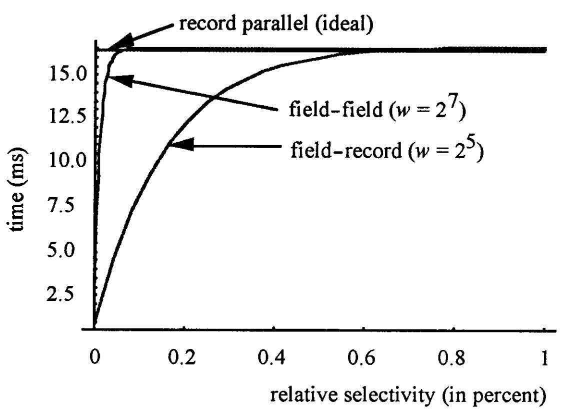


Fig. 16. Time to perform selection with no indexing. The effects of seek time, packing, and memory fragmentation have been neglected. The search time for field–record access is assumed to be 0.31 ms. With field–field access, the saturation occurs sooner if w is reduced.

power. It is assumed that the selection operand does not have to be read out twice during the search and readout portions of the operation. This is taken into consideration with the term multiplied by T_{rof} in expression (9):

$$T_{\text{search}} + \left[1 - \left(\frac{w}{r}\right)\left\lceil\frac{f}{w}\right\rceil\right]T_{\text{rof}}, \tag{9}$$

where

$$T_{\text{rof}} = t\left(\frac{r}{w}\right)\left\lceil\frac{Rw}{M^2}\right\rceil\left[1 - \left(1 - \left\lceil\frac{Rw}{M^2}\right\rceil^{-1}\right)^{N}\right]. \tag{10}$$

In Fig. 16 the time to perform ideal selection with no indexing is plotted versus selectivity for the three access modes. The time to perform this operation saturates for the two field-parallel-based accessing approaches. With field–record access, saturation occurs when it is likely that every record-parallel page has a selected record on it and needs to be read out. At this point the fraction of records that satisfy the selection criterion is roughly equal to 1 divided by the number of records on a page. For field–field access saturation happens when it is probable that every block needs to be read out: when the fraction of records that satisfies the selection criterion is approximately equal to 1 divided by the number of records in a block. When w is smaller the saturation occurs sooner. In the graph, w is set equal to 2^7 bits for field–field access. If 2^5 bits had been used it would have saturated so quickly that it would not have been visible.

E. Effect of Memory Fragmentation

At this point we have considered the performance of a biorthogonally accessed 3-D memory under ideal conditions. In real systems there are practical issues that can cause the performance to deviate from the ideal. One such problem is memory fragmentation, which occurs when records fit poorly onto record-parallel pages. As described in Subsection 2.E.3, this problem reduces the usable memory capacity and the bit-retrieval efficiency by a factor of α. The corresponding time to perform operations or portions of operations with record access is increased by approximately a factor of $(1 - 1/\alpha)$. This factor will generally be greatest when w is small. Table 3 lists values obtained for α and $1/\alpha$ for various values of w. When w is eight bits, the time required to perform this operation with record-parallel access is increased by 23% because $1/\alpha$ is equal to 1.23. When w is larger, however, there is little inefficiency.

This particular problem in some instances can make field-parallel access preferable, when under ideal conditions record-parallel access would have been preferred. For selection with clustered indexing, for example, field-parallel access with $w = 2^7$ bits can be preferable to record-parallel access if w is small and selectivity is high. This is also the case for selection with no indexing for high selectivities.

Table 3. Effect of w on α[a]

Size of w (bits)	α	$1/\alpha$
2^3	0.81	1.23
2^5	0.96	1.04
2^6	0.99	1.01
2^7	0.99	1.01

[a]The overhead for storing a record increases when w is small as a result of memory fragmentation.

F. Effect of Packing Strategy

Packing can have an impact on performance even greater than that of memory fragmentation. As described in Subsection 2.E.4, packing on average extends the time required for an operation by $1/B$, where B is the minimum number of superblocks required to store the relation, depending on the accessing technique used. Thus, if record-parallel pages are packed first, operations or portions of operations performed with field-parallel access will, on average, require $1/B$ more time to complete. However, the time to perform operations with record-parallel access will not be affected. Likewise, if record-parallel pages are packed first, the time for field-parallel access will similarly increase, but the time to perform this operation with record-parallel access will not be affected. In this study, B is equal 2. Thus the average time to perform operations can be increased by

Table 4. Time to Perform Selection with No Indexing with Selectivities of 1% and 10% by use of the Wisconsin Benchmark[a]

Record-Parallel Pages Packed First		Field-Parallel Pages Packed First	
Type of Access	Time (ms)	Type of Access	Time (ms)
Record parallel	16.45	Record parallel	24.67
Field–field access	24.19	Field–field access	16.13
Field–record access	16.92	Field–record access	16.75

[a]The value of w is 4 bytes.

Table 5. Time to Perform Selection with No Indexing for Gamma, PHOEBUS, and the 3-D Two-Photon Memory-Based Data Filter[a]

Memory Device and Selectivity	PHOEBUS	Gamma[b]	Record-Parallel Packing	Field-Parallel Packing
Device	Single parallel-access optical disk	30 sequential access disks	3-D two-photon memory (w = 4 bytes)	3-D two-photon memory (w = 4 bytes)
1% of records	0.42s	8.16s	0.0164s	0.0161s
10% of records	0.42s	10.82s	0.0164s	0.0161s

[a] Assuming a relation with 10^6 208-byte records.
[b] Thirty of Gamma's 32 processors were used.

50%, depending on the accessing technique used. Although the effect of adverse packing decreases for larger relations, it increases for smaller relations.

4. Performance Comparison with Other Systems

In this section we compare the performance of the 3-D biorthogonally accessed memory with that of two other parallel relational database machines, one existing and one proposed, by use of the operation selection with no indexing. Although we had wanted to compare the performance for other operations, this would have required the analysis of a complete relational database system. The present comparison is thus restricted to selection with no indexing because for the inclusion of other operations in our comparison the analysis of a complete relational database system would be required.

Table 4 shows the projected performance of the 3-D two-photon memory-based data filter for selection with no indexing by use of the Wisconsin benchmark, with w equal to 32 bits. The time to perform this operation with 1% and 10% selectivity is the same because all pages and blocks are likely to have at least one selected record on them. The selection times were calculated with the expressions introduced in Section 3, and the values are listed in Table 1.

The first machine that we considered in our performance comparison was Gamma,[1] a parallel relational database machine that was built at the University of Wisconsin and has 32 processors, each with its own serial-access disk. In this system data that belong to a single relation are distributed across all the disks to allow for parallel execution of queries. PHOEBUS (parallel hybrid optoelectronic database machine using disks),[3] the other system we considered, is a proposed system that relies on parallel-readout optical disk technology. In this system complete records are arranged on radial strips on the surface of an optical disk and read out in parallel, one at a time.

Table 5 lists the times to perform selection with no indexing for Gamma as well as the projected times for PHOEBUS and the biorthogonally accessed two-photon memory. Note that, when the benchmark was used to evaluate Gamma, only 30 processors and disks were used. For the two-photon memory, all pages have to be read out for this operation for the given selectivities. Thus the times for the two-photon memory reflect only the superior raw bandwidth and not the data-isolation features of the biorthogonal accessing approach. The superior performance is still clear. A more than 2 orders of magnitude improvement in performance can be gained by use of the two-photon memory instead of a system that relies on serial-access storage.

5. Conclusions

The 3-D two-photon memory, in general, was found to be well suited for very large databases with low write requirements because of its potential capacity, raw bandwidth, and random-access capability. Our calculations show that we can improve performance significantly by accessing pages of data from the memory in two orthogonal directions and by using a 3-D data-organization strategy designed for this approach. In such a system the user could choose between or combine these two access strategies to optimize the performance of a given operation. In our analysis we noticed that the particular way that records are oriented on pages in the memory affects performance and also memory fragmentation. The order in which the pages themselves are packed into the memory was also found to affect performance. No particular record-placement or record-packing strategy was found to be clearly advantageous for all operations. In a system this choice would have to be made by anticipation of the operations likely to be used most frequently. For the relation size that was considered in this study (10^6 records), the packing strategy was found to have a great impact on performance. This effect of packing would be reduced with larger relation sizes but would be increased with smaller-sized relations.

Both the throughput and the random-access capability of a memory affect the performance of relational database machines. A 3-D two-photon memory with biorthogonal access can be viewed as having improved random access, leading to improved performance. A shortcoming that all massively parallel-access optical memories have, in terms of this application, is that an entire page of records must be retrieved, even if only one record is desired. From this vantage point massively parallel-access memories appear to have reduced random-access capabilities.

The authors acknowledge the assistance of Walter Burkhard. This work was supported by the Air

Force Office of Scientific Research under grants F49620-93-I-0057 and F49620-93-1-0371.

References

1. D. J. DeWitt, S. Ghandeharizadeh, D. A. Schneider, A. Bricker, H. Hsiao, and R. Rasmussen, "The Gamma database machine project," IEEE Trans. Knowl. Data Eng. **2,** 44–61 (1990).
2. http://www.oracle.com/info/news/parallel.html.
3. P. A. Mitkas and P. B. Berra, "PHOEBUS: an optoelectronic database machine based on parallel optical disks," J. Parallel Distrib. Comp. **17,** pp. 230–244 (1993).
4. G. W. Burr, F. H. Mok, and D. Psaltis, "Storage of 10,000 holograms in $LiNbO_3$: Fe," in *Conference on Lasers and Electro-Optics*, Vol. 7 of 1994 OSA Technical Digest Series (Optical Society of America, Washington, D.C., 1994), p. 9.
5. M. Wang, S. E. Esener, F. B. McCormick, I. Çokgör, A. S. Dvornikov, and P. M. Rentzepis, "Experimental characterization of a two-photon memory," Opt. Lett. **22,** 558–560 (1997).
6. P. A. Mitkas and L. J. Irakliotis, "Three-dimensional optical storage for database processing," Opt. Mem. Neural Networks **3,** 217–229 (1994).
7. D. J. DeWitt, "The Wisconsin Benchmark: Past, Present and Future," in *The Benchmark Handbook for Database and Transaction Processing Systems*, 2nd. ed., J. Gray, eds. (Morgan Kaufmann, San Francisco, 1993, Chap. 4.
8. G. Gardin and P. Valduriez, *Relational Databases and Knowledge Bases* (Addison-Wesley, Menlo Park, Calif., 1989), Chap. 4.
9. G. Gardin and P. Valduriez, *Relational Databases and Knowledge Bases* (Addison-Wesley, Menlo, Park, Calif., 1989), Chap. 9.
10. R. A. Athale and M. W. Haney, "Optical implementation of numerical inequality detection and its applications to database machines," Opt. Lett. **17,** 1611–1613 (1992).
11. P. A. Mitkas, L. J. Irakliotis, F. R. Beyette, Jr., S. A. Feld, and C. W. Wilmsen, "Optoelectronic data filter for selection and projection," Appl. Opt. **33,** 1345–1353, (1994).
12. S. Y. W. Su, *Database Computers* (McGraw-Hill, San Francisco 1988), Chap. 4.
13. D. Schneider and D. J. DeWitt, "A performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment," in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, J. Clifford, B. G. Lindsay, and D. Maier, eds. (Association for Computing Machinery, New York, 1989), pp. 110–121.
14. P. Marchand, A. Krishnamoorthy, S. Esener, and U. Efron, "Optically augmented 3-D computer technology and architecture," in *Proceedings of the IEEE Massively Parallel Processing with Optical Interconnections Workshop* (Institute of Electrical and Electronics Engineers, New York, 1994), pp. 133–139.
15. J. Gray, ed., *The Benchmark Handbook for Database and Transaction Processing Systems* (Morgan Kaufmann, San Francisco, 1993).
16. S. Hunter, "Three-dimensional optical memory systems based on 2-photon excitation: system studies and component design," Ph.D. dissertation (University of California, San Diego 1994).
17. S. Hunter, F. Kiamilev, S. Esener, D. A. Parthenopoulos, and P. M. Rentzepis, "Potentials of 2-photon based 3-D optical memories for high performance computing," Appl. Opt. **29,** 2058–2066 (1990).