LOCALITY IN LOGICAL DATABASE SYSTEMS: A FRAMEWORK FOR ANALYSIS

by

EDWARD JAMES McCABE

S. B., Massachusetts Institute of Technology
(1976)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

(July 21, 1978)

Signature of Author...
Alfred P. Sloan School of Management
July 21, 1978

Certified by...
Thesis Supervisor

Accepted by..
Chairman, Department Committee

LOCALITY IN LOGICAL DATABASE SYSTEMS: A FRAMEWORK FOR ANALYSIS

by

EDWARD JAMES McCABE

Submitted to the Alfred P. Sloan School of Management
on July 21, 1978 in partial fulfillment of the requirements
for the Degree of Master of Science.

ABSTRACT

This thesis proposes measures for database locality and reports on their subsequent application to a series of reference strings from a large database. These measures are organized into temporal locality measures and spatial locality measures. The research concentrates on examining the interface between the user and the database system to minimize the influence of data models and their implementation on the results. Thus it differs from previous work by virtue of its framework and its perspective.

The reference strings are taken from a large IBM IMS database, the property database for the County of Riverside, California. Six temporal locality measures are applied to the reference strings. They indicate that there is a significant degree of temporal locality in the database. Two spatial locality measures are applied to the same data. These reveal that there is no appreciable spatial locality. Suggestions for further work in this area are presented.

Thesis Supervisor: Stuart E. Madnick
Associate Professor of
Management Science

CONTENTS

Abstract 2

1 Introduction and plan of thesis 6
    1.1 Introduction 6
    1.2 Significance of the problem 7
    1.3 Specific goals and accomplishments 8
    1.4 General structure of the thesis 9

2 Locality 10
    2.1 Program locality 10
    2.2 Types of locality 18
    2.3 Literature about database locality 25
    2.4 Implications of database locality 27

3 Measures 29
    3.1 Why measures 29
    3.2 Temporal locality measures 30
    3.3 Spatial locality measures 58

4 Experiments 66
    4.1 The database 66
    4.2 Temporal locality measures 69
    4.3 Spatial locality measures 88

5 Discussion and conclusions 104
    5.1 Introduction 104
    5.2 Summary 104
    5.3 Further work 105

Bibliography 106

FIGURES

1 Only one record referenced 34
2 Each record referenced once 36
3 Only one record referenced 38
4 Each record referenced once 39
5 No consecutive references to same record 44
6 All references to same record consecutive 46
7 Transactions and records referenced v. time 7Ø
8 Transactions and records referenced v. time 71
9 Transactions and records referenced v. time 72
1Ø Records referenced v. transactions 75
11 Records referenced v. transactions 77
12 Transactions, runs, and records referenced v. time ... 79
13 Pairs of references to the same record 84
14 Number of references v. stack distance 86
15 Number of record I/Os v. stack distance 88
16 Cooccurrences - block size = 5Ø records 9Ø
17 Cooccurrences - block size = 5Ø records 92
18 Cooccurrences - block size = 1ØØ records 93
19 Cooccurrences - block size = 1ØØ records 94
2Ø Cooccurrences - block size = 2ØØ records 95
21 Cooccurrences - block size = 2ØØ records 96
22 Weighted cooccurrence - window size = 5Ø 98
23 Weighted cooccurrence - window size = 5Ø 99
24 Weighted cooccurrence - window size = 1ØØ 1ØØ
25 Weighted cooccurrence - window size = 1ØØ 1Ø1
26 Weighted cooccurrence - window size = 2ØØ 1Ø2
27 Weighted cooccurrence - window size = 2ØØ 1Ø3

TABLES

1 Sample reference string 32
2 Sample calculation of records referenced 33
3 Sample reference string 41
4 Sample calculation of runs 42
5 Expected number of records referenced j times 49
6 Sample reference string 51
7 Sample calculation of distance 51
8 $Pr(d = j \mid d \leq n)$ for $N = 1000$, $n = 100$ 54
9 Sample reference string 56
10 Sample calculation 56
11 Final tableau 57
12 Record I/Os v. stack distance 58
13 Sample reference string 60
14 Sample reference string 63
15 Sample calculation 64
16 Breakdown of data used in analysis 69
17 Results of linear regression 76
18 Number of transactions per record 81
19 Expected number of records referenced j times 82

LOCALITY IN LOGICAL DATABASES

Chapter 1.

Introduction and plan of thesis

1.1 Introduction

The subject of this research is to investigate ways of measuring a property of reference strings generated by users of database systems. A reference string is a list of the records requested from a database in chronological order. The property chosen for investigation is known as the principle of locality.

1.1.1 What is locality

Locality is usually discussed in the context of program behavior. In this context, locality is:

> "the idea that a computation will, during an interval of time, favor a subset of the information available to it." (1)

Database locality is:

> the idea that the users of a database will, during an interval of time, favor a subset of the information available to them.

Consider the following example. The telephone directory for the city of Boston contains approximately

(1) Denning [1968], Resource Allocation in Multiprocess Computer Systems, p. 3.

400,000 listings, but most of us do not resort to it to call our parents, friends, or business acquaintances. Instead we have taken their phone numbers from the phone book and placed them into personal directories (some have probably been memorized). When we do place a call, we poll this storage hierarchy according to its access speed (first, our memory; second, the personal directory; and lastly, the telephone directory). This system works because we tend to call the same people day after day. If our calls were placed at random, we would have to resort to the phone directory many more times than we actually do. This favored subset (parents, friends, and business acquaintances) changes over time as people move, we meet new friends, clients change, etc.

1.2 Significance of the problem

Easton [1975] notes that:

> "One motivation for studying references to the data base is the availability of on-line storage devices with extremely large capacities. (For example, devices exist that can store more than 10**10 bytes of data.) Information concerning the structure of the data base reference string is a basic requirement for studies of a system that uses such a device as a backing store for disk storage." (2)

(2) Easton [1975], p.550.

This is the motivation behind this research. If large information utilities (like Madnick's INFOPLEX [1975]) are to be based on storage hierarchy systems, we must understand the underlying phenomenon, locality. As Madnick [1975] points out:

> "If all references to information in the system were random and unpredictable, there would be little utility for the intermediate levels of storage technologies. Most practical applications result in clustered references such that during any interval of time only a subset of the information is actually used..." (3)

1.3 Specific goals and accomplishments

The specific goals and accomplishments of this thesis, which are elaborated later, are:

- Review the literature on program locality and propose extensions of the basic themes to database locality.

- Introduce the research being conducted on various aspects of database locality.

- Propose measures for database locality based on Madnick's decomposition of locality into its temporal and spatial components.

(3) Madnick [1975], p. 582.

- Apply those measures to a series of database reference strings and analyse the results.

- Discuss the extension of this work to experiments with larger databases and speculate on locality in those databases.

1.4 General structure of the thesis

The rough outline of this work follows. The purpose of Chapter 2 is to acquaint the reader with the work done on program locality and introduce him to database locality. Chapter 3 proposes measures for both temporal and spatial database locality. In Chapter 4, these measures will be applied to a series of database reference strings and the results will be analysed. Finally, in Chapter 5, conclusions will be drawn about the effectiveness of the measures and their possible extension to other larger databases.

Chapter 2.

Locality

This chapter focuses on formalizing the definition of locality given in the introduction. By examining the fruits of the research done on program locality, the reader will gain a perspective useful in dealing with database locality. In particular, the definition of locality given above will be decomposed into two components: temporal and spatial. It has been shown that this distinction provides valuable insight into the behavior of demand paging systems. It also provides us with a framework for analysing database locality. The literature on database locality is examined. Finally, the implications of recognizing database locality and reconciling it with the database's structure are discussed.

2.1 Program locality

The principle of locality as it applies to programs (program locality) has been the subject of most of the research on locality. Section 2.1.1 gives a quick overview of the roots of program locality and the research community's efforts to model and measure it. Section 2.1.2 examines storage hierarchy systems, which rely on program locality to meet cost/performance criteria, and the programming techniques that should be adopted to maximize locality.

2.1.1 Historical perspective

This subsection begins by noting the speculation on locality's origins. Then it recapitulates Spirn and Denning's taxonomy of the models of locality. This taxonomy divided the subject material into two perspectives: an intrinsic view, which focuses on the state of the program as determining its reference pattern; and an extrinsic view, concentrating on the observable properties of the program as it executes (e.g. the memory reference sequence). Some of these models will surface again when the measures for database locality are presented.

2.1.1.1 Origins of program locality

Program locality has been attributed to programmers' own heuristic techniques. DO loops, arrays, and subroutines are all manifestations of the quest to simplify and generalize solutions to problems. Denning [1968] enumerates these factors in greater detail:

> "1. Sequential instruction streams. Both programmers and compilers tend to organize sequentially the instructions that direct the activity of a process; this is especially true in single-address machines (i.e., those with a

program counter). If a process fetches an instruction from a given page, it is highly probable that it will soon fetch another instruction, in sequence, from the same page.

2. Functional modularity. Program modules are organized and executed by function.

3. Content-related data organization. Information is usually grouped by content into segments, and is normally referenced that way; thus, references will occur in clusters to a content-related region in name space.

4. Looping. Programs often loop within a set of pages.

5. People. Realizing that their programs will run on a paged machine and that page transfers are costly, programmers tend to organize their algorithms so that activity is localized within subsets of their information. Moreover, people have been studying methods of minimizing interpage references at execution time." (4)

It is these characteristics which give rise to locality.

In higher level procedural languages, these patterns may be obscured by the compiler (alphabetizing variables before allocating storage for them), but often the programmer takes advantage of this behavior to produce more efficient code (e.g. suffixing variables, grouping them into structures or arrays).

(4) Denning [1968], Resource Allocation in Multiprocess Computer Systems, p. 40-41.

2.1.1.2 Taxonomy

Several models have been developed for examining and analysing program locality. These models form the basis for various definitions of locality inasmuch as their parameters provide quantitative measures for assessing a program's locality. The taxonomy presented by Spirn and Denning [1972] divides these models into two groups. The first group, the intrinsic models, identifies the models which are based on some knowledge of the program's structure. These models assume that the locality at any given time is a function of the state of the program at that moment. Consequently they predict the probability of referencing any given location as a function of the state of the program. The models in this group are:

1) page reference distribution functions,
2) the independent reference model (IRM),
3) the locality model, and
4) the LRU (least recently used) stack model or SLRUM.

For example, the simplest intrinsic model is the independent reference model. According to this model, the probability of referencing a given location at any instant in time is the same regardless of state. (In essence, one reference is <u>independent</u> of any other.) As you may suspect, it has been shown that the IRM produces poor fits to actual programs. (5)

On the other hand, the SLRUM, a more complex model, produces a good approximation to the real world behavior of programs. (6) This model is based on the memory contention stack generated by the LRU replacement algorithm. At any given time, the location at the top of the stack is the most recently used location. Subsequent references to different locations cause the stack to be pushed down to accomodate the referenced location. Let x(i) be a location in memory (not necessarily ordered on i). Let s(t) be the stack at time t, s(t) = {x(1), x(2), ..., x(n)}. If the program references the ith item in the stack at time t, then s(t+1) = {x(i), x(1), x(2), ..., x(i-1), x(i+1), ..., x(n)}. In this model, the probability of referencing the ith item in the stack (the stack distance probability, a(i)) at any given time is constant. Thus the state of the program (as represented by its stack, s(t)) determines the probability of referencing any given location.

(5) Spirn and Denning [1972], p. 614.

(6) Spirn and Denning [1972], p. 62Ø.

The extrinsic models are those that can be derived from the observable properties of the memory reference sequence. They are:

1) the locality sequence based on time intervals,
2) the locality sequence based on disjoint sets of pages, and
3) the working set, W(t,T) - the set of pages referenced among the last T references at time t.

For example, the working set model uses the set of pages referenced among the last T references (which does not make any assumptions about the program's state) as a model of the program's locality.

2.1.2 Applications of program locality

Recognizing program locality and designing tools to capture it has received a good deal of attention. Particularly important applications of the principle are those in storage hierarchy systems, programming techniques, and reordering frequently used programs.

2.1.2.1 Storage hierarchy systems

Systems which rely on the principle of locality predate the formal recognition of the principle itself. Demand paging systems and segmentation memory management techniques are based on locality. Demand paging takes advantage of infrequent use of portions of a routine. (7) Segmentation recognizes the time v. space tradeoff for loading infrequently used routines into main memory. In one case (segmentation), the partitioning is done at the user's behest. In the other (paging), it is invisible.

Madnick [1973] argues that the principle of locality extends to all storage hierarchies:

> "[E]ach level [of a storage hierarchy] 'sees' a different view of the program. The high levels of the hierarchy must follow the microscopic instruction by instruction reference pattern whereas the middle levels follow a more gross subroutine by subroutine pattern. The very low levels are primarily concerned about the processor's references as it moves from subsystem to subsystem. We do not have any a priori guarantee that locality of reference holds equally true for all of these views, but we do have some reported evidence to encourage us." (8)

(7) The Atlas computer system, developed in 1961, used a demand paged memory hierarchy (described as the "Automatic Use of a Backing Store"), but the concept itself (locality) was not ennunciated until several years later when Denning and others began to model the performance of virtual memory systems.

(8) Madnick [1973], 56-57.

2.1.2.2 Programming techniques

Several researchers have investigated the effects of different programming techniques on locality of reference. Kuehner and Randell [1968] enumerated a set of "programming commandments" that included localizing activity for intervals instead of moving rapidly over the program's address space. (9) These commandments are particularly important for programs used frequently. Brawn and Gustavson [1968] report that:

> "The data indicate that, if reasonable programming techniques are employed, the automatic paging facility compares reasonably well (even favorably in some instances) with programmer controlled methods [e.g. overlays]. While not spectacular, these results nonetheless look good in view of the substantial savings in programmer time and debugging time that can still be realized even when constrained to employing reasonable virtual machine programming methods." (1Ø)

Essentially these authors urge the programmer to recognize locality and act accordingly.

2.1.2.3 Reordering frequently used programs

Hatfield and Gerald [1971] demonstrated the use of

(9) Another commandment deals with excessive modularity (one component of "structured programming"). It cautions against using program modules at the expense of additional page faults and dynamic control transfers.

(1Ø) Brawn and Gustavon [1968], 1Ø28-1Ø29.

computer displays of memory usage to assist them in reordering relocatable program sectors to substantially reduce the number of page exceptions (faults) in frequently used programs (e.g. assemblers, compilers). By interpreting the memory usage data as graphic evidence of locality, they sought to increase locality by clustering closely referenced sectors into the smallest set of pages. These displays gave them immediate feedback on the automated procedures they were employing to reorder the program sectors.

2.2 Types of locality

Thus far, we have dealt only with the notion of program locality; however, Madnick [1973] has identified two underlying phenomena of locality: temporal and spatial. This section examines these components and proposes definitions for their database components. Finally, it discusses the synthesis of these components.

One of the difficult concepts to resolve when extending the definition of locality to databases is identifying the database counterpart of "address". For our purposes that counterpart is a record, where:

> A record is the fundamental unit the user can deal with (be it to retrieve, store, or modify).

This definition has been adopted to maintain independence between the work done here and the implementations of database systems. Indeed, users of the same database may have different records (e.g. users in the personnel office might have access to individual employees' records, while those in the corporate strategy office might be restricted to aggregate statistics by plant or division).

2.2.1 Program locality

2.2.1.1 Temporal locality

Madnick's definition of temporal locality is:

> "If the logical addresses {a1, a2, ...} are referenced during the time interval t-T to t, there is a high probability that these same logical addresses will be referenced during the time interval t to t+T." (11)

Thus, a reference sequence which repeatedly references the same location in a period of time demonstrates a high degree of temporal locality. An example of this behavior for a program would be the reference sequence encountered when searching an array.

```
(1)  load i
(2)  add 1
(3)  store i
```

(11) Madnick [1973], p. 120.

```
(4)  load b(i)
(5)  compare
(6)  go to (1)
```

In this sequence of references, the same location ("i") is referenced on 2 of 3 data references.

2.2.1.2 Spatial locality

Madnick's definition of spatial locality is:

> "If the logical address a is referenced at time t, there is a high probability that a logical address in the range a-A to a+A will be referenced at time t+1." (12)

Here a reference to one location presages references in the near future to neighboring items. The literature on program locality usually defines neighboring items as those that are physically contiguous (in the same page). The example given for program locality in 2.2.1.1 above also demonstrates spatial locality, inasmuch as a reference to "b(i)" presages one to "b(i+1)".

2.2.1.3 Locality and its components

Though temporal and spatial locality are the underlying phenomena, "general locality" is the topic of most of the discussion in the literature. To reconcile our definitions

[12] Madnick [1973], p. 121.

with those in the literature, merge the definitions of temporal and spatial locality found above. The result is a definition for general locality:

> "If the logical addresses {a1, a2, ...} are referenced during the time interval t-T to t, there is a high probability that the logical addresses in the ranges a1-A to a1+A, a2-A to a2+A, ..., will be referenced during the time interval t to t+T." (13)

But the distinction between the two components is important. Hatfield [1972] noted an anomoly when studying the page fetch frequencies (the number of times it was necessary to fetch a page from the paging device) of programs with high locality. If the page size was halved, the frequency of page fetches occasionally more than doubled. Madnick [1973] followed this work by determining the upper bound on the increase in page fetch frequency and proposed an algorithm, "tuple-coupling", to limit the page fetch frequency to twice its former value when the page size was halved. In his report he observed:

> "In particular, we see, that whereas temporal locality policies are given explicit attention [by conventional removal algorithms], spatial locality policies are usually handled implicitly and subtely. The "least recently used", LRU, removal algorithm, for example, is very much concerned about the temporal aspects of the program's reference pattern. The spatial aspects are

(13) Madnick [1973], p. 121.

> handled as a by-product of the fact that the demand fetch algorithm must load an entire page (i.e., a spatial region) at a time and LRU removal decisions are based upon these pages. With these thoughts in mind, we can see that decreasing page size causes the conventional storage management algorithms to increase their sensitivity to temporal locality and decrease their sensitivity to spatial locality." (14)

2.2.2 Database locality

The idea that the probability of accessing any given record in a database might differ from that of accessing another record is not new. Knuth comments that a typical distribution was formulated by G. K. Zipf in 1949. (15) This distribution or "Zipf's Law" is based on Zipf's principle of least effort. One demonstration of this principle, the economy of words, involved word frequency counts in James Joyce's novel Ulysses. In this novel and in several other works, the rank of the word (in terms of its frequency of use) times the number of times the word was used was approximately equal to a constant. (See Zipf [1949] for more detail.)

(14) Madnick [1973], p. 122.

(15) Knuth [1973], p. 397.

LOCALITY IN LOGICAL DATABASES

2.2.2.1 Temporal locality

By substituting record for logical address, the definitions given above can be extended to apply to temporal locality in a database sense. In this context successive references to the same record by an applications program would be indicative of a high degree of locality. As an example, an applications program might generate these requests against the database.

(1) read record A

(2) modify record A

(3) print record A

This sequence, not atypical for a clerk modifying the contents of a record, demonstrates locality inasmuch as the same record is referenced three times in succession.

2.2.2.2 Spatial locality

Applying this concept to databases is not as simple as applying temporal locality to databases. Particularly bothersome is the notion of a neighboring record. Two intrinsic definitions of a neighboring record are possible.

1) Given a particular application, a neighboring record is an record logically related to the recently referenced record.

2) Given a database system, a neighboring record is an

> record physically grouped (i.e. in the same physical data record or nearby data record) with the record recently referenced.

Both these definitions have merit, but the former relies on knowledge of the application that is hard to obtain. The latter depends on the physical implementation of the database (which may be in response to a perception of the application's locality or may not be directly controllable).

An extrinsic definition has more merit for our purposes. A neighboring record to one recently referenced is one with a substantially higher probability of being referenced because the first record was referenced. The definition for spatial locality for databases becomes:

> If the record a is referenced at time t, there is a high probability that a record from the set of neighboring records (relative to a) will be referenced at time t+1.

2.2.3 Measuring database locality - practice

From the discussion above, it is apparent that there is no hard and fast rule for detecting locality. In fact, most of the database installations visited in the course of this research intimated that they had no way of detecting, much less explaining abnormal levels of activity for sets of records or individual records in a database. The intrinsic definition which relies on knowledge of the application is especially vulnerable to this criticism. An example of a hitherto unanticipated locality in a group health ·claims application is the large volume of surgery claims against the insurance company by workers of a company on strike. Rather than man the picket lines, workers apparently elect to undergo previously deferred elective surgery.

2.3 Literature about database locality

Most of the work on database locality has concentrated on the interface between the storage subsystem and the database system. The authors have drawn an analogy between the virtual memory paging system and the database system. In this context, the counterpart of the primary memory of a paging system is the database buffer pool space. The page's counterpart is the block (which contains a number of records). These researchers have concentrated on modeling the path segment reference string. (Path segments are records that must be accessed before the requested record or target segment can be referenced. This is similar to a tape in which the first 499 records must be accessed before the 5ØØth record may be referenced.) Often, the path segment reference string is reduced to a string of block references for the sake of convenience (i.e. the path segment references are converted to block references). Consequently, the uses of the model are in the determination of the effects on working set size, etc. of altering the block size or the database buffer pool space.

Easton [1975] used a simple Markov chain model to describe an interactive database path segment reference string and validated it using data from an interactive database system, the Advanced Administrative System (AAS, see Wimbrow [1971]). His model was found to accurately predict working set sizes. An interesting result of this work showed that as the window size is varied over three orders of magnitude that the miss ratio (the percentage of references not satisfied by the first level storage devices) varied by only a factor of three. This is quite contrary to the behavior of demand paging systems in which window size exponentially affects the miss ratio (generating a parachor curve).

Rodriguez-Rosell [1976] comments that this finding and his corroborating experiments carried out on an IMS system indicate that database reference strings exhibit weak locality. But, he argues that these reference strings display strong sequentiality. Consequently, prefetch is an attractive alternative to demand fetch in database systems.

In March, 1978, Easton published a paper which described another model for referenre strings. The basis of this model is the observation that:

> "once a page is referenced, there are often additional references to it within a relatively short period of time." (16)

He calls this property the time clustering of references (temporal locality). His new model distinguishes between two kinds of references to records. If a record was last referenced some arbitrary period of time, tau, before it is referenced again, this later reference is a primary reference; otherwise, the later reference is a secondary reference. The time between the last secondary reference and the subsequent primary reference is modeled as a random variable with a geometric distribution. From this he can accurately predict the page fault probability and the mean storage utilization. Again, this is verified by analysing trace data from an AAS and an IMS system.

2.4 Implications of database locality

The reason so many resources have been focused on recognizing locality and rearranging databases to match that pattern is that the performance can be dramatically

(16) Easton [1978], p. 197.

improved. As many database administrators can tell you, adding an inverted file or maintaining another set of set pointers (network) can reduce the run time of an applications package by several hundred percent.

Chapter 3.

Measures

3.1 Why measures

The brief survey of the field presented in Section 2.3 shows that the work on database locality has focused on modeling and analysing the requests for blocks of records issued by the database system to the storage subsystem. Indeed, these studies have concentrated on hierarchical database systems with the predictable result:

> "Data base reference strings have been found to exhibit strong sequentiality in addition to weak locality." (17)

There are two problems with the research that has been done so far. First, the concentration on the interface between the storage system and the database system has led to conclusions that can not be generalized to other types of database systems (e.g. network and relational). (I suspect that Rodríguez-Rosell's assertion that reference strings exhibit strong sequentiality is particularly subject to this criticism due to the nature of the IMS hierarchical data model.) The second fault with the research is that it ignores an important distinction between the types of locality. This distinction has proven valuable in the case

(17) Rodríguez-Rosell[1976], p. 13.

of program locality, but has been ignored in the work done on database locality.

The measures presented here aim to correct this situation by examining the interface between the user and the database system. This makes it possible to distinguish between sequentiality inherent in the application (i.e. always processing credit card authorization requests in ascending order) and sequentiality induced by the access method employed by the database system (i.e. HISAM in IMS). The insight gained from these measures into the processes generating the requests should remain valid as the database is modified or restructured. (18)

To facilitate the analysis presented in the rest of the paper, we will present the measures in terms of transactions to the database. A transaction is an action by a user against one record in the database.

3.2 Temporal locality measures

In this section, the measures for temporal locality will be presented. Some of the measures will be illustrated

(18) It is possible that the users of the database system have adopted their mode of operation in view of the performance characteristics of the database, but this possibility will be ignored.

by extreme cases to guide the analyst examining his own data. Some will use statistical tests to prove or disprove a hypothesis about how the records were selected. The key to using these measures is understanding the aspects of temporal locality that each captures.

3.2.1 Database references v. time

A starting point in any analysis of database activity should be the examination of database activity over time. This sets up the ground work for subsequent analysis, inasmuch as it pinpoints periods of unusual activity for the database.

Not only should be number of references to the database (transactions) be plotted against time, but the number of records referenced should also be plotted. The number of records referenced is the cumulative number of unique records referenced by transactions. Since temporal locality addresses the question of the probability of referencing the same record again, the cumulative number of unique records referenced represents the observed frequency with which records were referenced.

For example, let Table 1 represent the transactions and the records each transaction references (i.e. transaction #4 references record "A").

Sample reference string

transaction	record
1	A
2	B
3	A
4	A
5	B
6	A
7	C
8	C
9	C
1Ø	A

Table 1.

Then the number of records referenced at any transaction is simply the number of different records referenced to that point (i.e. 2 records, "A" and "B", have been referenced by transactions 1-4).

Sample calculation of records referenced

transaction	record	records referenced
1	A	1
2	B	2
3	A	2
4	A	2
5	B	2
6	A	2
7	C	3
8	C	3
9	C	3
10	A	3

Table 2.

This is illustrated as follows. Suppose that only one record was referenced by 20 transactions over a period of time, then the graph might look like:

Only one record referenced

t = transactions

r = records referenced

```
20 +                                        t
   |                                      t
   |                                    t
   |                                  t
   |                                t
15 +                              t
   |                            t
   |                          t
   |                        t
   |                      t
10 +                    t
   |                  t
   |                t
   |              t
   |            t
 5 +          t
   |        t
   |      t
   |    t
   |  trr r r r r r r r r r r r r r r r r r
 0 +
   -+---------+---------+---------+---------+-

    0         5         10        15        20

                       Time
```

Figure 1.

(At time = 10, all 10 transactions thus far have referenced the same record.) (19) In this case we would say that there

[19] The time scale here and in the rest of the figures in this section has been arbitrarily chosen. Note that these measures do not assume that transactions arrive at a constant rate.

is a high degree of temporal locality in the database, since during the interval of observation the probability of referencing the same record is one.

If on the other hand each transaction accessed a different record, the graph would show that the two lines were superimposed.

Each record referenced once

t = transactions

r = records referenced

```
20 +                                        tr
   |                                      tr
   |                                    tr
   |                                  tr
   |                                tr
15 +                              tr
   |                            tr
   |                          tr
   |                        tr
   |                      tr
10 +                    tr
   |                  tr
   |                tr
   |              tr
   |            tr
 5 +          tr
   |        tr
   |      tr
   |    tr
   |  tr
 0 +
   -+---------+---------+---------+---------+-

    0         5        10        15        20

                    Time
```

Figure 2.

(At time = 10, each of the 10 transactions has referenced a different record.) This example demonstrates little or no temporal locality, since the probability of referencing the same record in the interval is zero.

These graphs display the extreme cases. In all probabilitv, real data would yield something in between.

3.2.2 Number of records referenced v. number of transactions

Though the graph described above gives us some indication of the temporal locality in the database, it is difficult to arrive at an idea of the consistency of this behavior over time. By plotting number of records referenced v. number of transactions, the time bias can be eliminated. (Lunch hours, coffee breaks, and other periods when there were few transactions to the database will be compressed.)

The shape of this curve tells us how locality changes over time. The closer the slope of the curve is to zero, the higher the degree of temporal locality since each transaction tends to reference a previously referenced record. Conversely, the closer the slope of the curve is to one, the lower the degree of temporal locality. In this case, each transaction references a previously untouched record.

For example, the best case for temporal locality woula produce a plot that looked like:

Only one record referenced

```
r = records referenced

      2Ø +
         |
         |
         |
         |
      15 +
         |
         |
         |
         |
      1Ø +
         |
         |
         |
         |
       5 +
         |
         |
         |
         |  r r r r r r r r r r r r r r r r r r r r
       Ø +
         -+---------+---------+---------+---------+-

          Ø         5        1Ø        15        2Ø

                      Transactions
```

Figure 3.

This curve is a straight line and has a slope of zero since each transaction references the same record. The worst case for temporal locality would be:

LOCALITY IN LOGICAL DATABASES

```
                Each record referenced once

r = records referenced

20 +                                        r
   |                                      r
   |                                    r
   |                                  r
   |                                r
15 +                              r
   |                            r
   |                          r
   |                        r
   |                      r
10 +                    r
   |                  r
   |                r
   |              r
   |            r
 5 +          r
   |        r
   |      r
   |    r
   |  r
 0 +
   -+---------+---------+---------+---------+-

    0         5         10        15        20

                  Transactions
```

Figure 4.

This curve is a straight line with a slope of one (i.e. every reference touches a different record).

Within these bounds the curve is constrained to be monotonically non-decreasing, since the cumulative number of records referenced can not decrease. The slope of the curve at any one point would reflect the average number of records referenced per transaction and is everywhere constrained to be between zero and one inclusive. (This can be taken as one of the quantitative measures of temporal locality.)

The second derivative of the records referenced with respect to the number of transactions gives us an indication of the change in temporal locality at any given point. A positive second derivative is indicative of decreasing temporal locality, since transactions are referencing more previously unreferenced records. A negative second derivative indicates that temporal locality is increasing as more transactions reference previously referenced items.

3.2.3 Runs v. time

Another method that may be used in examining database temporal locality is to identify runs (successive references to the same record) and plot the cumulative number of runs v. time (the run curve). As the length of the runs increases (and correspondingly the number of runs decreases) the run curve will lag beneath the number of transactions curve. The length of a run is indicative of temporal locality since it shows a record's probability of being referenced on the next transaction to the database. This is particularly true in systems where only one user is allowed in the database at any time (as is the case with many database systems when the user wants to restructure or modify the database). In a multi-threaded machine with several users issuing transactions at any given instant, the number of runs may not be an accurate indicator of temporal locality since users' transactions will be interleaved.

Our definition of a run is similar to that of the reduced block reference string derived from a program's address trace as consecutive references to the same block (or record in this case) are compressed to form one reference. For example, if Table 3 is a reference string.

Sample reference string

transaction	record
1	A
2	B
3	A
4	A
5	B
6	A
7	C
8	C
9	C
10	A

Table 3.

Then the number of runs would be:

Sample calculatioı. of runs

transaction	record	runs
1	A	1
2	B	2
3	A	3
4	A	3
5	B	4
6	A	5
7	C	6
8	C	6
9	C	6
10	A	7

Table 4.

Transaction 4 marks the end of a run of length 2 (transactions 3-4). By transaction 4, there have been 3 runs. The average length of the runs could be used as a quantitative measure of temporal locality subject to the constraints discussed above. In this case, the average length of a run for record "A" is (1 + 2 + 1 + 1)/4 or 1.25, for "B" is 1 and for "C" is 3.

One of the problems with using the records referenced measure (section 3.2.2) is that it has an infinite memory for records referenced. For example, if there were 1000 transactions between successive references to the same record, the number of records referenced v. number of transactions curve would be a line whose slope was equal to the long run ratio of the number of records referenced to the number of transactions. This would belie the temporal locality in the database.

Once again, it is possible to establish bounds on the run curve. At a worst case for temporal locality, each reference to the database would reference a different record than the preceding references. (This presupposes that there is more than one record selected in the given period of time.) Thus the run curve would be superimposed on the number of transactions curve. Figure 5 shows this:

No consecutive references to same record

t = transactions

R = runs

r = records referenced

```
 20 +                                        tR
    |                                      tR
    |                                    tR
    |                                  tR
    |                                tR
 15 +                              tR
    |                            tR
    |                          tR
    |                        tR
    |                      tR
 10 +                    tR
    |                  tR
    |                tR
    |              tR                        r
    |            tR                    r r r
  5 +          tR              r r r r
    |        tR          r r r
    |      tR      r r r
    |    tRr r r r
    |  tRr
  0 +
    -+---------+---------+---------+---------+-

     0         5        10        15        20

                     Time
```

Figure 5.

By time = 10, there had been 4 records referenced by 10 transactions. There were 10 runs, consequently each run had a length of 1 transaction.

The best possible case for temporal locality would be that where the first time a record was referenced was followed by all other references to the same record. Thus the number of runs to any point in time would be identical to the number of records referenced. This would be identical with the number of records referenced curve. Figure 6 demonstrates this:

All references to same record consecutive

t = transactions

R = runs

r = records referenced

```
2Ø +                                        t
   |                                      t
   |                                    t
   |                                  t
   |                                t
15 +                              t
   |                            t
   |                          t
   |                        t
   |                      t
1Ø +                    t
   |                  t
   |                t
   |              t                         Rr
   |            t                     RrRrRr
 5 +          t               RrRrRrRr
   |        t           RrRrRr
   |      t       RrRrRr
   |    t RrRrRrRr
   |  tRRr
 Ø +
   -+---------+---------+---------+---------+-

    Ø         5        1Ø        15        2Ø

                    Time
```

Figure 6.

In this figure, by time = 1Ø there had been 4 runs, 4 records referenced, and 1Ø transactions.

3.2.4 Number of references per record

Another metric which goes hand-in-hand with those mentioned here is the distribution of the number of records referenced once, twice, thrice, etc. If we model our database as an urn with N distinct balls from which we are making n picks we can test the following hypothesis:

> At any given point in time, any record is equally likely to be referenced.

This is a binomial process with x(i) the event of picking the ith ball x times. Let j be the number of times a record is picked. Given that $j \leq n$ (there are at least as many picks as the number of times the record is selected) and p (the probability of picking a particular ball), then:

$$Pr(x(i) = j) = \binom{n}{j} p^{j} (1 - p)^{n - j}$$

Equation 1.

For example, given n picks, what is Pr(x(i) = 3)?

For n = 3, Pr(x(i) = 3) is:

$$p^3$$

For n = 4, Pr(x(i) = 3) is:

$$p^3 (1 - p) + p^2 (1 - p) p + p (1 - p) p^2 + (1 - p) p^3$$

In general (for j = 3):

$$\Pr(x(i) = 3) = \binom{n}{3} p^{3} (1 - p)^{n - 3}$$

This formula becomes unwieldy for large n since it requires computing n choose j. Instead we will use the approximation found in Equation 2. (2Ø) (21)

$$\Pr(x(i) = j) = e^{-np} \frac{np^{j}}{j\,!}$$

Equation 2.

The expected number of records referenced j times is found by multiplying Pr(x(i) = j) by the number of records. Given 1ØØ transactions to a 1ØØØ record database you would expect that:

(2Ø) Wonnacott recommends using the Poisson distribution for rare events when np (the number of trials times the probability of success) < 5. Wonnacott [1977], p. 17Ø.

(21) Chou recommends using this approximation for n > 1ØØ and p < Ø.Ø1. Chou [1975], p. 186.

Expected number of records referenced j times
N = 1000, n = 100, p = 0.001

j	E(x(i) = j)	E(x(i) = j)
	(binomial)	(Poisson)
0	904.792	904.837
1	90.570	90.484
2	4.488	4.524
3	0.147	0.151
4	0.004	0.004
5	0.000	0.000

Table 5.

Note that the sum of the number of records referenced for all possible values of j is equal to N (the number of records) and that the weighted sum of E(x(i) = j) for j > 0 equals n (the number of picks). Note too, that the Poisson approximation yields:

$$E(x(i) = j) = E(x(i) = j - 1) \frac{np}{j}$$

If there is a significant degree of temporal locality in the database, we would expect that the number of records referenced twice would be much larger than expected, since a high degree of temporal locality implies a high probability of referencing a previously referenced record again. A chi-squared test for goodness of fit will be used to demonstrate that the selection process is not random and consequently that there is temporal locality.

3.2.5 Distances between successive references to the same record

If some portion of the population is referenced more than once, the distance between successive references to the same item gives us an important clue to the degree of temporal locality in the database. The metric we will use for distance, the number of intervening transactions, will isolate this measure from the disturbances injected by stochastic events (coffee breaks, lunch hours, etc.). If the temporal locality is high, we would expect to find that the distribution of distances favored the shorter distances. Examining the distribution of these distances may shed some light on this aspect of temporal locality that was left unexplored by the number of records referenced v. number of transactions plot and was crudely examined by the run curve. First, an example of the operational definition of our metric is presented.

If an item is referenced two or more times, order the accesses to the record chronologically, pair the accesses (take N accesses to the record and generate N-1 pairs by combining successive references to the same record), and record the number of intervening transactions. Thus if the reference string was:

Sample reference string

transaction	record
1	A
2	B
3	A
4	A
5	B
6	A
7	C
8	C
9	C
1Ø	A

Table 6.

Then the reference pairs generated would be:

Sample calculation of distance

transaction	record	next use	distance
1	A	3	2
2	B	5	3
3	A	4	1
4	A	6	2
5	B	-	-
6	A	1Ø	4
7	C	8	1
8	C	9	1
9	C	-	-
1Ø	A	-	-

Table 7.

If this were a random process with the probability of picking a particular record at any given point in time equal to p, we could model it as a binomial process. Let d be the random variable whose value is the number of the pick on which the given record appears. Then the probability of picking a given record at pick j given that it is picked within n picks is:

$$\Pr(d = j \mid d \leq n) = \frac{p\,(1 - p)^{j - 1}}{\sum_{i = 1}^{n} p\,(1 - p)^{i - 1}}$$

Equation 3.

The denominator is the sum of a finite geometric series (a = p and r = (1 - p)). As such it reduces to:

$$\frac{p - p\,(1 - p)^{n}}{1 - (1 - p)}$$

Equation 4.

Consequently,

$$\Pr(d = j \mid d \leq n) = \frac{p\,(1 - p)^{j - 1}}{1 - (1 - p)^{n}}$$

Equation 5.

If the number of records is much larger than the number of picks and the number of picks greater than 10, the probability of any j for $j \leq n$ is approximately 1/n or uniformly distributed. We can show this if we substitute 1/N (where N is the number of records in the database) for p in the preceding equation, the result is:

$$\Pr(d = j \mid d \leq n) = \frac{(1 - \frac{1}{N})^{j - 1}}{(1 - (1 - \frac{1}{N})^{n}\,)\,N}$$

Simplifying:

$$\Pr(d = j \mid d \leq n) = - \frac{N^{n - j}\,(N - 1)^{j - 1}}{(N - 1)^{n} - N^{n}}$$

But for $N \gg 0$, $n > 10$:

$$(N - 1)^{n} = N^{n} - nN^{n - 1}$$

Therefore, using this assumption:

$$Pr(d = j \mid d \leq n) = \frac{- j + N + 1}{nN}$$

And since N >> j:

$$Pr(d = j \mid d \leq n) = \frac{1}{n}$$

For example, examine Table 8, which shows the probability density function for selected values of j using the exact formula (given 1000 records and 100 picks). (The approximation yields the value 1/100 or 0.01.) Table 8 demonstrates the accuracy of the approximation.

$Pr(d = j \mid d \leq n)$ for N = 1000, n = 100

j	Pr(d = j)
1	0.010503
11	0.010399
21	0.010295
31	0.010193
41	0.010091
51	0.009991
61	0.009891
71	0.009793
81	0.009695
91	0.009599
100	0.009513

Table 8.

3.2.6 SLRUM

The LRU Stack Model (SLRUM) used for program locality was modified to use records instead of memory addresses. (22) This model is based on the memory contention stack generated by the LRU replacement algorithm. At any given time, the item at the top of the stack is the most recently used record. Subsequent references to different records cause the stack to be pushed down to accomodate the referenced record. By recording the number of times transactions are satisfied by the i_th_ most recently referenced record as a function of i, the stack distance probabilities can be estimated.

The stack distance probability, a(i), is the probability of a reference being satisfied by the i_th_ item from the top of the stack. The distance, i, varies from 1 (the top of the stack) to N (where N is the number of records in the database). Strictly speaking if there is temporal locality, the a(i)s should be monotonically non-increasing as the distance from the top of the stack increases. A less restrictive criteria for temporal locality holds that $\min(a(1) + a(2) + \ldots + a(m)) \geq$

(22) See Mattson, et. al. [1970] for a more through discussion of this model applied to program locality.

max(a(m+1) + a(m+2) + ... + a(N)) for a memory of size m.

For example, assume that Table 9 is the reference string.

Sample reference string

transaction	record
1	A
2	B
3	A
4	A
5	B
6	A
7	C
8	C
9	C
10	A

Table 9.

The measure is constructed by noting the references to each level of memory.

Sample calculation
References v. stack distance.

	ref # 1		ref # 2		ref # 3	
Depth	Hits	Order	Hits	Order	Hits	Order
1	0	A	0	B	0	A
2	0	*	0	A	1	B
3	0	*	0	*	0	*
infinity	1	*	2	*	2	*

Table 10.

Thus, reference 3 to record "A" was satisfied by the record at depth (stack distance) 2 and was the first reference to a record at that depth. Record "A" was then placed at the top of the stack. (References to previously unreferenced records are recorded as references to a depth of infinity, as displayed in references 1 and 2.) This continues until the final reference is logged.

Final tableau
References v. stack distance.

	ref # 1Ø	
Depth	Hits	Order
1	3	A
2	4	C
3	Ø	B
infinity	3	*

Table 11.

If we assume that one I/O is required to fetch a record for the processor when the record is not in memory, the number of record I/Os necessary to access the records in the sample reference string can be tabulated as a function of memory size (see Table 12).

Record I/Os v. stack distance

Depth	record I/Os
0	10
1	9
2	6
3	3

Table 12.

If memory were large enough to hold all three records, only three record I/Os would be required to satisfy the sample reference string.

If this table is plotted as a function of the memory size, a parachor curve for this reference string can be derived that resembles that constructed for programs. This curve traces the number of references made to the ith most recently referenced item. In this manner, it is possible for the database administrator to say for any given amount of memory the number of record I/Os he would expect.

3.3 Spatial locality measures

Spatial locality measures are more difficult to come to grips with because of the definitional problems of neighborhoods. The two measures we will use here are both extrinsic, in that they do not assume any a priori knowledge of the reference string or the process generating the reference string. The difficulty in interpreting these measures stems from the fact that there is no way to deal with the large number of points generated by such measures. Inasmuch as spatial locality implies a relationship between a minimum of two points, the number of measurements to be taken is approximately the number of records squared. (If the measure is symmetric, this number may be trimmed by one half.)

This is the classic problem of cluster analysis. To deal with this large number of points cognitively it is necessary for us to group them in some fashion, but by limiting ourselves to an extrinsic measure we have no idea how this grouping should be done. The key to this problem is to define a measure that will allow us to construct meaningful groups of records based solely on the value of the measure.

3.3.1 Cooccurrences

If there is spatial locality in a database, we would expect to find clusters of references to various groups of records. This is a consequence of the definition of spatial locality (i.e. "there is a high probability that a record from the set of neighboring records will be referenced"). For example, records "A" and "B" are members of a set of neighboring records; therefore, a reference to record "A" would probably be followed by a reference to record "B".

For the purposes of the measure presented here we will partition the reference string into blocks of equal length. For each pair of records referenced, we will count the number of blocks in which both occurred (a cooccurrence) and the number of blocks in which either occurred (an occurrence). The value of the measure for that particular pair of records will be the result obtained by dividing the number of cooccurrences by the number of occurrences.

For example, if the reference string were:

Sample reference string

transaction	record
1	A
2	B
3	A
4	A
5	B
6	A
7	C
8	C
9	C
10	A

Table 13.

and the block size was 5 references (i.e. block 1 contained references to records "A" and "B", while block 2 contained references to records "A" and "C"), then our measure for spatial locality would show that "A" and "B" were related at the Ø.5 level. ("A" and "B" cooccurred in 1 block, but "A" or "B" occurred in 2 blocks.) Records "B" and "C" have a cooccurrence measure of Ø since they do not appear in the same block. The other pair of records, "A" and "C", are related at the Ø.5 level.

If there are N records referenced in the reference string, this measure generates N (N - 1)/2 points. The value of the measure for a pair ranges from Ø (if the records never cooccur) to 1 (if all blocks that contain a reference to "A" contain a reference to "B"). Note that this measure does not discriminate between numerous cooccurrences in numerous occurrences and a single cooccurrence in one occurrence. (i.e. If records "A" and "B" cooccurred in 1ØØ blocks and "A" or "B" occurred in 1ØØ blocks, the measure for "A" and "B" would be 1. If records "C" and "D" cooccurred in 1 block and occurred in 1 block, their measure would be 1, too.)

Hence, the values obtained will be a function of the reference string and the block size. An interesting feature of this measure is its sensitivity to changes in block size. Increasing the block size can result in a decrease in the values of the measure! This anomaly can occur when the new block size is not a multiple of the old. Suppose that the block size is 3, transaction 8 references record "A", and transaction 9 references record "B". If these were the only references to these records, the value of the measure for the "A" and "B" pair would be 1. However, if the block size is increased by 1 to 4, the value of the measure is Ø (because transaction 8 is in block 2 and transaction 9 is in block 3). This anomaly does not occur if the block sizes are multiples of one another.

If we incorporate this restriction into the measure we can examine the changes in the distribution of the values as the block size is varied. This should indicate the degree of spatial locality.

3.3.2 Weighted cooccurrences

A measure which takes the relative number of each type of record into account is the weighted cooccurrence measure. This measure differs from the previous one in that it employs a moving window of W references around each instance of "A" instead of partitioning references into blocks and since it counts the number of "B"s in that window as distinct from noting the mere presence of a "B".

To determine the value of the measure for the pair of record "A" and record "B", simply count the number of "B"s that occur within W transactions (forward or backward) of the transaction that references "A".

For example, let Table 14 be a reference string and the window size, W, assume the value 2.

Sample reference string

transaction	record
1	A
2	B
3	A
4	A
5	B
6	A
7	C
8	C
9	C
1Ø	A

Table 14.

Then the value of the measure for the "A" and "C" pair is determined by summing the "# in window" column in Table 15. (This column reflects the number of "C"s which lie in the window, W = 2, surrounding each reference to "A". Thus the value for transaction 6 is 2.)

Sample calculation
Weighted cooccurrence measure.

transaction	record	# in window
1	A	Ø
2	B	-
3	A	Ø
4	A	Ø
5	B	-
6	A	2
7	C	-
8	C	-
9	C	-
1Ø	A	2

Table 15.

The value of the measure for "A" and "C" is 4.

This measure is symmetric (i.e. the value of "A" and "C" is the same as the value of "C" and "A"), but unlike the previous measure it has no readily apparent upper limit. The minimum value the measure may assume is Ø, which would mean that no references to "C" occurred within W references of "A".

Once again, the measure is a function of the reference string and the window size. Because of the moving window, this measure demonstrates no anomaly as window size increases. The distribution of the values will serve as the indicator of spatial locality.

Chapter 4.

Experiments

4.1 The database

These measures were applied to a set of five reference strings obtained by monitoring a large database. The database, described by Hackathorn [1976], is the property database for the County of Riverside, California.

The property database is implemented on an IBM 370/158 with two million bytes of main storage and over two billion bytes of online disk storage. The database management system used is IBM's IMS, version 2.3. The property database contains approximately 400,000 records and occupies 500 million bytes of disk storage. Each record in the property database (i.e. land parcel) can be identified by one of three primary identifiers: (1) assessment number, (2) situs address, and (3) assessee name. The principal identifier used in the day-to-day transactions is the assessment number. (The number of transactions to the property database which used the assessment number as the identifier is shown in Table 16 as the "with id" column. The percentage of the transactions to the property database that used the assessment number as the identifier appears in the adjacent column.) A transaction may not include an identifier if: (1) it was syntactically incorrect because of operator error, (2) it was a dummy transaction to initialize the screen menu on the CRT, or (3) it dealt with multiple identifiers in a complex manner (e.g. subdivisions).

Over one hundred transaction programs have been implemented to support seven functional areas of county government (valuation, identification, exemption, value certification, tax rate establishment, tax accounting, and public service). The main departments of the County of Riverside with direct involvement in the property system are the Assessor, Auditor, Tax Collector, Recorder, and Building Department. Four title insurance companies have direct access to the system for inquiry purposes.

In its raw form, the data for the experiment came from users of the database system. Typically, a user would enter an eight character transaction code and a variable length message (which included the record identifier) on one of the 35 terminals connected to the system. This information (the transaction and the message) would be logged by a common service facility to the system log tape before it was passed to the database system. Three to six reels of magnetic tape were generated by a typical day's transactions. The data from the system log tapes were reformatted by the IMS Statistical Analysis Utility (IMSTATS) and processed by routines written by Hackathorn to prepare them for his experiments.

The transaction data contained a number of fields for each transaction. For our purposes we are only interested in the identifier of the record affected by the transaction and the time the transaction took place (not part of the transaction itself but added by the telecommunications facility). To simplify this analysis, only those transactions which reference records by assessment number have been included in this experiment. The transactions were garnered from Hackathorn's data for the dates August 19, 1975, August 22, 1975, August 25, 1975, August 29, 1975, and September 5, 1975. Of the transactions so culled, a number were unusuable because the time of the transaction against the identified record could not be determined (apparently this is due to the IMSTATS malfunction). Table 16 presents a breakdown of the relevant statistics.

Breakdown of data used in analysis

date	day	# of trans	with id	%	with time	%
Aug 19, 1975	Tue	7753	4088	52.7	4014	98.2
Aug 22, 1975	Fri	10281	8803	85.6	6584	74.8
Aug 25, 1975	Mon	11315	9779	86.4	5845	59.8
Aug 29, 1975	Fri	8743	4802	54.9	1983	41.3
Sep 5, 1975	Fri	8196	7570	92.4	7240	95.6

Table 16.

These transactions were themselves placed in a database for the experiments. Most of the analysis herein was performed using programs found in MIT's Consistent System.

4.2 Temporal locality measures

4.2.1 Database references v. time

Both number of transactions and number of records referenced were plotted against time for each of the five days. The plots for August 19, August 25, and September 5 follow.

Transactions and records referenced v. time
August 19, 1975

t = transactions

r = records referenced

```
5ØØØ +
     |
     |
     |
     |
4ØØØ +                                        t t t
     |                                    t t t
     |                                t t t
     |                                t
     |                              t t
3ØØØ +                            t t
     |                            t
     |                          t t
     |                          t
     |                        t t
2ØØØ +                        t
     |                      t t           r r r r r
     |                      t         r r r
     |                    t t     r r r
     |                  t t     r r
1ØØØ +          t t   t t     r r
     |          t           r r
     |        t t       r r r
     |      t t r r   r r
     |    t trr r
   Ø +trtrtrr
     -+-----+-----+-----+-----+-----+-----+-----+-----+-

      Ø           4           8          12          16
            2           6          1Ø          14
```

Time (hours)

Figure 7.

Transactions and records referenced v. time
August 25, 1975

t = transactions

r = records referenced

```
6000 +
     |                                    t     t t
     |                                    t
     |                                  t t
     |                                t t
     |                                t
     |                              t t
     |                            t t
     |                          t t
4000 +                          t
     |                        t t
     |                        t
     |                      t t
     |                      t             r     r r
     |                    t t         r r r
     |                    t       r r r
     |                  t t     r r
     |                t t     r r
2000 +                t     r r
     |                t   r r
     |              t t r r
     |              t r r
     |            t t r
     |            t r r
     |          t trr
     |        t trr
     |    t trtrr
   0 +trtrtrr
     -+---------+---------+---------+---------+---------

      0         2         4         6         8

                 Time (hours)
```

Figure 8.

Transactions and records referenced v. time
September 5, 1975

t = transactions

r = records referenced

```
8ØØØ +
     |
     |
     |
     |
     |                                            t
6ØØØ +                                          t t
     |                                      t t t
     |                        t           t t
     |                      t t
     |                      t
     |                    t t
4ØØØ +                  t t
     |                t t
     |                t                           r
     |              t t                     r r r r
     |              t       r r           r r
     |            t t     r r
2ØØØ +            t   r r r
     |          t t r r
     |        t t   r
     |        t r r r
     |      t trr
     |      trr
   Ø +trtrtrtr
     -+-----+-----+-----+-----+-----+-----+-----+-----+-

      Ø           4           8          12          16
            2           6          1Ø          14

                    Time (hours)
```

Figure 9.

In every case it was immediately apparent that the database activity was not constant with time. There were periods of inactivity scattered throughout the day. (Of course, it is impossible to say if the malfunction in the IMSTATS routine mentioned above was the cause of the apparent inactivity.) In any event this convinced us that time was not the best base for future analysis.

Another feature of the transactions plotted was that there was some temporal locality in the course of the day. For example, examine the plot for August 19. Approximately 1Ø hours after the start of the day's activities, 1,4ØØ records had been referenced by a total of 3,ØØØ transactions. In particular, the slope of the number of records referenced curve seemed to average about 1/2 the slope of the number of transactions curve. This would indicate that in the long run there was one record referenced for the first time for every two references to the database. However, it is difficult to demonstrate from this curve that that behavior was constant throughout the day, because of the time skew effect.

4.2.2 Number of records referenced v. number of transactions

Number of records referenced v. number of transactions was plotted for each day in the sample to normalize for the uneven levels of transaction activity over time. The slope of this line is the average number of records referenced per transaction. The closer the slope is to zero, the higher the degree of temporal locality. As you can see in Figure 1Ø, the slope of the line is approximately 1/2, confirming the suspicions raised in the previous section that one record is referenced for every two transactions.

Records referenced v. transactions
August 19, 1975

```
r = records referenced

2000 +
     |
     |                              r r
     |                            r r
     |                          r r
1500 +                          r
     |                        r r
     |                      r r
     |                    r r
     |                  r r
1000 +                r r
     |                r
     |              r r
     |            r r
     |          r r
 500 +        r r
     |      r r
     |    r r
     |    r
     |  r r
   0 +r r
     -+-------+-------+-------+-------+-------+-

      0     1000    2000    3000    4000    5000

                 Transactions
```

Figure 10.

A linear regression was run using transactions as the independent variable and records referenced as the dependent for each of the five days. Table 17 presents the results.

Results of linear regression
Transactions v. records referenced.

date	day	# of trans	index of fit	slope
Aug 19, 1975	Tue	4014	.999	.466
Aug 22, 1975	Fri	6584	.990	.547
Aug 25, 1975	Mon	5845	.996	.559
Aug 29, 1975	Fri	1983	.948	.497
Sep 5, 1975	Fri	7240	.999	.532

Table 17.

The shape of the curve is another matter. If the day's transactions started up by referencing some large portion of the records to be used that day we would see a curve like Figure 11.

```
         Records referenced v. transactions

r = records referenced

       2Ø +
          |
          |                              r r r r r r
          |                          r r
          |                      r r
       15 +                    r
          |                  r
          |                r
          |              r
          |            r
       1Ø +          r
          |          r
          |        r
          |        r
          |      r
        5 +      r
          |    r
          |    r
          |  r
          |  r
        Ø +
          -+---------+---------+---------+---------+-

           Ø        1Ø        2Ø        3Ø        4Ø

                       Transactions
```

Figure 11.

Here the slope of the curve is one through the first 1Ø transactions as each transaction references an unreferenced record. Transactions 1Ø through 2Ø reference only 5 previously unreferenced records, whereas transactions 2Ø through 3Ø reference only 3 more. Finally transactions 3Ø through 4Ø deal only with previously referenced records.

Note that the temporal locality is increasing and the second derivative is negative. The slope of the curve for August 19, 1975 (see Figure 1Ø) suggests that there is little if any change in the temporal locality in the course of the day.

4.2.3 Runs v. time

The run curve was plotted for each of the five days. Figure 12 is typical of the results obtained.

Transactions, runs, and records referenced v. time
August 19, 1975

t = transactions

R = runs

r = records referenced

```
5000 +
     |
     |
     |
     |
4000 +                                        t t t
     |                                    t t t R R
     |                                t t tRR R R
     |                                tRR R
     |                              t tR
3000 +                            t tRR
     |                            tRR
     |                          t tR
     |                          tRR
     |                        tRtR
2000 +                        tR
     |                      t tR          r r r r r
     |                      tRR       r r r
     |                    tRtR    r r r
     |                  tRtR    r r
1000 +          tRtR  tRtR    r r
     |          tR          r r
     |        tRtR      r r r
     |      tRtRr r   r r
     |    tRtRr r
   0 +tRtRtRr
     -+-----+-----+-----+-----+-----+-----+-----+-----+-

      0           4           8          12          16
            2           6          10          14

                   Time (hours)
```

Figure 12.

From the above figure we see that after 1Ø hours of operation, approximately 3,ØØØ transactions resulted in 2,8ØØ runs and 1,4ØØ referenced records. Apparently there were very few instances of consecutive references to the same record. This was an interesting result, especially in view of the previous findings (one record referenced per two transactions on average). It is indicative of a lower degree of temporal locality than otherwise indicated. At a second glance; however, it can be explained by the operating environment of the database system. This is a multiprogramming system. There are many users operating on the database at a particular moment. Thus, it is unlikely that any particular user will be able to push two transactions through the system before another user's transaction is interleaved.

4.2.4 Number of references per record

To characterize the temporal locality in this database, the number of transactions per record was profiled. The profile for August 19, 1975 is presented in Table 18 (i.e. 1277 records were only used in one transaction, 394 records were used twice, 75 records were referenced thrice, etc.).

Number of transactions per record
August 19, 1975

number of transactions	number records
1	1277
2	394
3	75
4	64
5	20
6	4
7	1
8	1
10	1
18	1
416	1
435	1
450	1

Table 18.

Note that a substantial number of records were the subject of two or more transactions. A statistical test was performed on the actual v. the expected number of records referenced once, twice, etc.

In particular, Equation 2 (page 48) in section 3.2.4 yields the expected number of records referenced as a function of j. If n = 4014 (the number of trials) and p = 1/400000 (the probability of picking one record) then Table 19 has the expected number of items referenced j times.

Expected number of records referenced j times
N = 400000, n = 4014, p = 0.0000025

j	E(x(i) = j)
0	396006.070
1	3973.921
2	19.939
3	0.067
4	0.000
5	0.000

Table 19.

A simple chi-squared test for goodness of fit can be used to test the hypothesis that the records were randomly selected. The chi-squared statistic is computed by taking the sum of the squares of the differences between the actual and expected number of observations in each category and dividing by the expected number of observations. (23) Subsequently, this statistic is compared with those found in a table indexed by probability and degrees of freedom.

Let's divide the actual and expected number of records referenced into two groups, those records referenced two or more times and those referenced one or zero times. Since there are two categories and the underlying distribution is

(23) Each category must have an expected number of five or more observations. Several of the original categories may be combined to meet this criterion.

known, we have one degree of freedom. $E(x(i) \geq 2) = 20.006$, but the actual number is 564. The chi-squared statistic for this data is 14,792, which indicates that the actual number is many thousand standard deviations removed from the expected number. This would lead me to reject the hypothesis that records are selected at random.

4.2.5 Distances between successive references to the same record

These tests have not yet distinguished the case where items are referenced in some initialization state and not referenced again until considerably later. By examining those records referenced two or more times, it is possible to see how much intervening activity there is between successive references to the same record.

From section 3.2.5, we saw that given a large number of records and a reasonable number of picks, that $Pr(d = j \mid d \leq n) = 1/n$. In this database with its 400,000 records and thousands of picks, we would expect that assumption would hold and that the density function for Pr(d) would be uniform. What we observe in this database is something completely different. In this database references to the same item occur very close to one another (i.e.

approximately 75 pairs of references to the same record were separated by exactly 1Ø transactions to the database). In particular the distribution of Pr(d) observed is:

```
          Pairs of references to the same record
                     August 19, 1975

n = number of pairs of references

250 +    n
    |
    |  n
    |  n
2ØØ +
    |
    |      n
    |
15Ø +
    |        n n
    |
    |            n
1ØØ +
    |            n
    |              n
    |                n
 5Ø +                  n
    |                    n
    |                      n n
    |                          n n n n
  Ø +                              n   n n n n n
    -+-------------+-------------+-------------+-

     Ø            1Ø            2Ø            3Ø

          Distance between successive references
                        (records)
```

Figure 13.

From section 4.2.4, we know that there were 2,183 pairs of references. (Each record referenced more than once generates N-1 pairs of references, where N is the number of transactions that referenced that record.) A count of the number of pairs reveals that 75% had a distance of 11 or fewer. Thus in 1,637 cases out of 2,183, successive references to the same record occurred within the space of 11 transactions to the database. This indicates that there is a high degree of temporal locality in the database.

4.2.6 SLRUM

The observed number of references at each stack distance was computed for each day's transactions. Figure 14 shows the data for August 19, 1975.

Number of references v. stack distance
August 19, 1975

n = number of references

```
3ØØ +
    |      n
    |
    |
    |
    |    n
    |
    |  n
    |
2ØØ +          n
    |        n
    |
    |
    |            n
    |              n
    |
    |
    |                n
1ØØ +
    |                  n
    |
    |                    n
    |                      n
    |                        n
    |                          n n n
    |                                n     n
    |                                  n n     n n
  Ø +                                        n     n n
    -+---------+---------+---------+---------+--------

     Ø         5        1Ø        15        2Ø
```

Stack distance (records)

Figure 14.

In the course of the 4,Ø14 transactions on August 19, approximately 7Ø transactions referenced the 1Øth most recently referenced record. Note that this measure differs from the previous one in that if there were two intervening transactions between references to the record in question, that measure would show a distance of two but this measure would treat it as one reference.

This figure also indicates that there is a high degree of temporal locality in the database. The next figure gives us an indication of the kind of performance benefits we are talking about. In particular, Figure 15 shows that if we held the last 10 records in memory and were using an LRU algorithm we could reduce the number of I/Os required to fetch the records in the reference string from 4,104 (no records in memory) to approximately 2,400.

Number of record I/Os v. stack distance
August 19, 1975

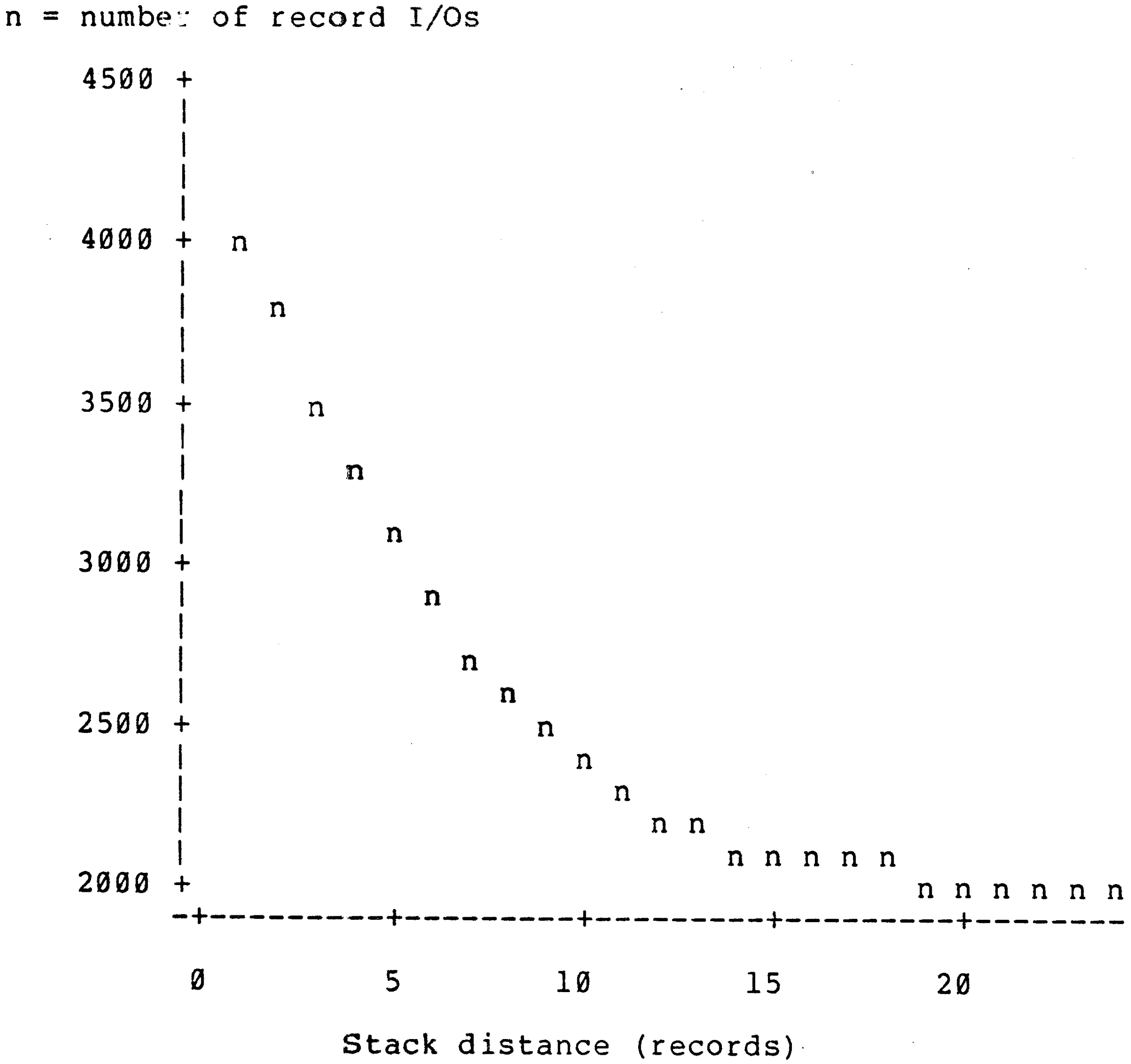


Figure 15.

4.3 Spatial locality measures

The measures outlined in section 3.3 were applied to this prôperty database. As noted above, they generated a large number of points (even though we excluded those records from the reference string that were referenced two or fewer times). For example, on August 19 there were 170 records referenced three or more times. These records generated 14,365 points for each trial (using different block or window sizes).

In this section, the values of a measure for a particular experiment are categorized into 100 intervals of equal length. The number of values in each category was tabulated. Finally, the number of values in each category was plotted against the value of the category.

For example, in the experiment that produced Figure 16 there were 14,365 values. (One value for each pair of records referenced.) These values ranged from 0 to 1. Thus the interval length was 0.01. The number of values in each interval was counted. (i.e. How many values were between 0.00 and 0.01, between 0.01 and 0.02, between 0.02 and 0.03, etc.?) Approximately 12,670 values fell between 0.00 and 0.01 for that figure. This plot conveys a rough idea of the distribution of the values.

4.3.1 Coöccurrences

As outlined in section 3.3.1, the cooccurrences measure was evaluated for the reference string of August 19, 1975. Figure 16 displays the distribution of the values for a block size of 50 references.

Cooccurrences - block size = 50 records
August 19, 1975

```
n = number of points

 1.4000 +
*10** 4 |
        |n
 1.2000 +
*10** 4 |
        |
 1.0000 +
*10** 4 |
        |
 0.8000 +
*10** 4 |
        |
 0.6000 +
*10** 4 |
        |
 0.4000 +
*10** 4 |
        |
 0.2000 +
*10** 4 |
        |             n     n
 0.0000 +n n n n n n n n n n n n n n n n n n n n n
*10** 4 -+-------+-------+-------+-------+-------+-

      0.0000  0.2000  0.4000  0.6000  0.8000  1.0000

                 Cooccurrence measure
```

Figure 16.

The mean of the values was Ø.Ø426. The standard deviation was Ø.157. As you can see, a substantial number of the values (approximately 12,67Ø) were in the interval from Ø.ØØ to Ø.Ø1. To examine the points outside the Ø.ØØ to Ø.Ø5 interval it was necessary to exclude those points inside that interval. Figure 17 displays the distribution for the remaining points on an expanded ordinate.

```
          Cooccurrences - block size = 50 records
               Minimum x value plotted = 0.05
                      August 19, 1975

n = number of points

     400 +              n
         |                    n
         |
         |
         |
     300 +
         |
         |
         |
         |
     200 +                                        n
         |
         |
         |
         |
     100 +          n
         |
         |
         |                          n
         |  n
       0 +  n n n n n n n n n n n n n n n n n n n n
         -+-------+-------+-------+-------+-------+-

       0.0000  0.2000  0.4000  0.6000  0.8000  1.0000

                    Cooccurrence measure
```

Figure 17.

As shown on the expanded plot, there are few values greater than 0.05.

The effect of a larger block size was evaluated by using block sizes of 100 and 200. Figure 18 presents the results for block size = 100.

```
          Cooccurrences - block size = 100 records
                      August 19, 1975

n = number of points

  1.4000 +
 *10** 4 |
         |
  1.2000 +n
 *10** 4 |
         |
  1.0000 +
 *10** 4 |
         |
  0.8000 +
 *10** 4 |
         |
  0.6000 +
 *10** 4 |
         |
  0.4000 +
 *10** 4 |
         |
  0.2000 +
 *10** 4 |
         |  n           n     n                   n
  0.0000 +n n n n n n n n n n n n n n n n n n n n n
 *10** 4 -+-------+-------+-------+-------+-------+-

       0.0000  0.2000  0.4000  0.6000  0.8000  1.0000

                    Cooccurrence measure
```

Figure 18.

The mean of the values was 0.066. The standard deviation was 0.202. The expanded plot looked like:

Cooccurrences - block size = 100 records
Minimum x value plotted = 0.05
August 19, 1975

```
n = number of points

     800 +
         |
         |
         |
         |                    n
     600 +
         |
         |
         |
         |                                        n
     400 +                n
         |
         |
         |
         |
     200 +
         |
         |    n
         |
         |          n               n
       0 +  n n n n n n n n n n n n n n n n n n n n
         -+-------+-------+-------+-------+-------+-

       0.0000  0.2000  0.4000  0.6000  0.8000  1.0000

                 Cooccurrence measure
```

Figure 19.

When the block size was 200, the results remained the same.

```
           Cooccurrences - block size = 200 records
                      August 19, 1975

n = number of points

  1.2000 +
 *10** 4 |n
         |
  1.0000 +
 *10** 4 |
         |
  0.8000 +
 *10** 4 |
         |
  0.6000 +
 *10** 4 |
         |
  0.4000 +
 *10** 4 |
         |
  0.2000 +
 *10** 4 |                   n
         |   n                                   n
  0.0000 +n n n n n n n n n n n n n n n n n n n n n
 *10** 4 -+-------+-------+-------+-------+-------+-

       0.0000  0.2000  0.4000  0.6000  0.8000  1.0000

                   Cooccurrence measure
```

Figure 20.

The mean of the values was 0.124. The standard deviation was 0.279. The expanded plot looks like:

```
                 Cooccurrences - block size = 2ØØ records
                      Minimum x value plotted = Ø.Ø5
                            August 19, 1975

n = number of points

    14ØØ +
         |
         |                    n
    12ØØ +
         |
         |
    1ØØØ +                                        n
         |
         |
     8ØØ +
         |
         |
     6ØØ +
         |
         |
     4ØØ +
         |    n         n
         |
     2ØØ +
         |        n
         |
       Ø +  n n n n n n n n n n n n n n n n n n n n
         -+-------+-------+-------+-------+-------+-

       Ø.ØØØØ  Ø.2ØØØ  Ø.4ØØØ  Ø.6ØØØ  Ø.8ØØØ  1.ØØØØ

                      Cooccurrence measure
```

Figure 21.

As we increase the block size, the mean and the standard deviation increase; but, the distribution har remained essentially unchanged. Since the vast majority of the values are in the interval from Ø.ØØ to Ø.Ø5, this database displays little spatial locality.

4.3.2 Weighted cooccurrences

This measure was also evaluated by applying it to the reference string of August 19, 1975. Figure 22 displays the distribution of the values for a window size of 50 references.

Weighted cooccurrence - window size = 50
August 19, 1975

n = number of points

```
 1.6000 +
*10** 4 |
        |
 1.4000 +n
*10** 4 |
        |
 1.2000 +
*10** 4 |
        |
 1.0000 +
*10** 4 |
        |
 0.8000 +
*10** 4 |
        |
 0.6000 +
*10** 4 |
        |
 0.4000 +
*10** 4 |
        |
 0.2000 +
*10** 4 |
        |
 0.0000 +n n n n n n n n n n n n n n n n n n n n n n n
*10** 4 -+-------+-------+-------+-------+-------+-------

      0.0000  0.2000  0.4000  0.6000  0.8000  1.0000
      *10** 4 *10** 4 *10** 4 *10** 4 *10** 4 *10** 4
```

Weighted cooccurrence measure

Figure 22.

The mean of the values was 5.58. The standard deviation was 152. Most of the values here are clustered in the first interval. The expanded plot (excluding the first interval) shows that an inconsequential number of values fell outside the first interval.

```
          Weighted cooccurrence - window size = 5Ø
               Minimum x value plotted = 1ØØ
                      August 19, 1975

n = number of points

      5Ø +
         |
         |
         |n
      4Ø +
         |
         |
         |
      3Ø +
         |
         |
         |
      2Ø +
         |
         |
         |
      1Ø +
         |
         |n
         |
       Ø +n n n n n n n n n n n n n n n n n
         -+-----+-----+-----+-----+-----+-----+-

       Ø.ØØØØ      Ø.4ØØØ      Ø.8ØØØ      1.2ØØØ
       *1Ø** 4     *1Ø** 4     *1Ø** 4     *1Ø** 4
             Ø.2ØØØ      Ø.6ØØØ      1.ØØØØ
             *1Ø** 4     *1Ø** 4     *1Ø** 4

                 Weighted cooccurrence measure
```

Figure 23.

The effect of a larger window size on the distribution was evalueated by using window sizes of 100 and 200. Figure 24 displays the results for window size = 100.

```
         Weighted cooccurrence - window size = 100
                     August 19, 1975

n = number of points

 1.4000  +n
*10** 4  |
         |
 1.2000  +
*10** 4  |
         |
 1.0000  +
*10** 4  |
         |
 0.8000  +
*10** 4  |
         |
 0.6000  +
*10** 4  |
         |
 0.4000  +
*10** 4  |
         |
 0.2000  +
*10** 4  |
         |
 0.0000  +n n n n n n n n n n n n n n n n n n n n n
*10** 4  -+---------+---------+---------+---------+------

          0        5000      10000     15000     20000

            Weighted cooccurrence measure
```

Figure 24.

The mean of the values was 11.1. The standard deviation was 291. The expanded plot looks like:

```
         Weighted cooccurrence - window size = 100
               Minimum x value plotted = 100
                      August 19, 1975

n = number of points

  80 +
     |
     |n
     |
     |
  60 +
     |
     |
     |
     |
  40 +
     |
     |
     |
     |
  20 +
     |
     |
     |
     |n n
   0 +  n n n n n n n n n n n n n n n n n
    -+-------+-------+-------+-------+-------+-

      0     5000    10000   15000   20000   25000

             Weighted cooccurrence measure
```

Figure 25.

When the window size was 200, the result was:

```
          Weighted cooccurrence - window size = 200
                      August 19, 1975

n = number of points

  1.6000 +
 *10** 4 |
  1.4000 +n
 *10** 4 |
  1.2000 +
 *10** 4 |
  1.0000 +
 *10** 4 |
  0.8000 +
 *10** 4 |
  0.6000 +
 *10** 4 |
  0.4000 +
 *10** 4 |
  0.2000 +
 *10** 4 |
  0.0000 +n n n n n n n n n n n n n n n n n n n n n
 *10** 4 -+---------+---------+---------+---------+-

          0        10000     20000     30000     40000

               Weighted cooccurrence measure
```

Figure 26.

The mean of the values was 21.7. The standard deviation was 551. The expanded plot looks like:

```
         Weighted cooccurrence - window size = 200
               Minimum x value plotted = 100
                      August 19, 1975

n = number of points

     100 +
         |
         |n
         |
      80 +
         |
         |
         |
      60 +
         |
         |
         |
      40 +
         |
         |
         |
      20 +
         |
         |
         |n n
       0 + n n n n n n n n n n n n n n n n n n n n
         -+---------+---------+---------+---------+-

          0       10000     20000     30000     40000

               Weighted cooccurrence measure
```

Figure 27.

Once again, the distribution of the values suggests that there is little temporal locality in the database.

LOCALITY IN LOGICAL DATABASES

Chapter 5.

Discussion and conclusions

5.1 Introduction

In the course of this research we have investigated a series of measures for analysing database locality. These measures were broken into two groups. In this chapter we will discuss the results of the experiments for the groups and offer some guidance for the reader interested in pursuing this work.

5.2 Summary

The first group of measures, the temporal locality measures, proved to be equal to the task at hand. They identified the presence and degree of temporal locality in the database.

The second group of measures, the spatial locality measures, encountered a number of problems in their application. The most significant problem was in the interpretation of the measures. Other than noting the character of the distributions and the shift with the increase in block size and window size, few courses of action suggested themselves to the author.

5.3 Further work

Several extensions to the work reported here seem worthwhile to this researcher. Particularly important work can be done by:

- Applying the measures defined here to other series of reference strings. One of the objectives of this research has been to separate the measures from the implementation of the database system. Identifying and validating the locality in another environment would serve this purpose.

- Defining new measures that retain the distinction between temporal and spatial locality. Other measures, especially for spatial locality, are needed.

The most vexing problem for the researcher in this area is the spatial locality measure. As mentioned in section 3.3 the number of values increases rapidly as the size of the database grows. Not only does this pose cognitive problems for the person interpreting these measures, but the computational work required to evaluate these measures can be excessive. The solution may be a combination of intrinsic and extrinsic measures (e.g. couple what we know about the application to some external measure).

BIBLIOGRAPHY

Anderberg, Michael R., Cluster Analysis for Applications, Academic Press, pp. 359 (1973).

Batson, Alan, Program Behavior at the Symbolic Level, Computer 9, 11, 21-26 (1976).

Brawn, Barbara S. and Frances G. Gustavson, Program Behavior in a Paging Environment, FJCC 33, 1019-1032 (1968).

Chou, Ya-lun, Statistical Analysis with Business and Economic Applications, Holt, Rinehart, and Winston, pp. 844 (1975).

Crow, Edwin L., Frances A. Davis, and Margaret W. Maxfield, Statistics Manual with Examples Taken from Ordnance Development, Dover Publications, pp. 258 (1960).

Denning, Peter J., On Modeling Program Behavior, SJCC 40, 937-944 (1972).

Denning, Peter J., Resource Allocation in Multiprocess Computer Systems, Project MAC TR-50, MIT, pp. 285 (1968).

Denning, Peter J., Thrashing: Its Causes and Prevention, FJCC 33, 915-922 (1968).

Denning, Peter J., Virtual Memory, Computing Surveys 2, 3, 153-189 (1970).

Easton, Malcom C., Model for Database Reference Strings Based on Behavior of Reference Clusters, IBM Journal of Research and Development 22, 2, 197-202 (1978).

Easton, Malcom C., Model for Interactive Data Base reference String, IBM Journal of research and Development 19, 6, 550-556 (1975).

Ferrari, Domenico, The Improvement of Program Behavior, Computer 9, 11, 39-47 (1976).

Fotheringham, John, Dynamic Storage Allocation in the Atlas Computer, Including an Automatic Use of a Backing Store, CACM 4, 10, 435-436 (1961).

Hackathorn, Richard Dale, Activity Analysis: A Methodology for the Discrete Process Modeling of Information Systems in Organizations, with an Application to a Government Database, Unpublished Ph.D. thesis, University of California at Irvine, Information Science, pp. 245 (1976).

Hatfield, D. J., Experiments on Page Size, Program Access Patterns, and Virtual Memory Performance, IBM Journal of Research and Development 16, 1, 58-66 (1972).

Hatfield, D. J. and J. Gerald, Program Restructuring for Virtual Memory, IBM Systems Journal 10, 3, 168-192 (1971).

Hedges, Robert Lewis, An Analysis of Locality in Paged Computer Memory Hierarchies, Unpublished Ph.D. thesis, Texas A&M University, Computer Science, pp. 145 (1974).

Johnson, Jerry W., Program Restructuring for Virtual Memory Systems, Project MAC TR-148, MIT, pp. 213 (1975).

Knuth, Donald E., The Art of Computer Programming, Vol. 3, Addison-Welsey, pp. 722 (1973).

Kuehner, C. J. and B. Randell, Demand Paging in Perspective, FJCC 33, 1011-1018 (1968).

Madnick, Stuart Eliot, INFOPLEX - Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex, NCC 44, 581-586 (1975).

Madnick, Stuart Eliot, Storage Hierarchy Systems, Project MAC TR-107, MIT, pp. 153 (1973).

Mattson, R. L., J. Gecsei, D. R. Slutz, and I. L. Traiger, Evaluation Techniques for Storage Hierarchies, IBM

Systems Journal 9, 2, 78-117 (1970).

Rodriguez-Rosell, Juan, Empirical Data Reference Behavior in Data Base Systems, Computer 9, 11, 9-13 (1976).

Spirn, Jeffrey R., Distance String Models for Program Behavior, Computer 9, 11, 14-20 (1976).

Spirn, Jeffrey R. and Peter J. Denning, Experiments with Program Locality, FJCC 41, 611-621 (1972).

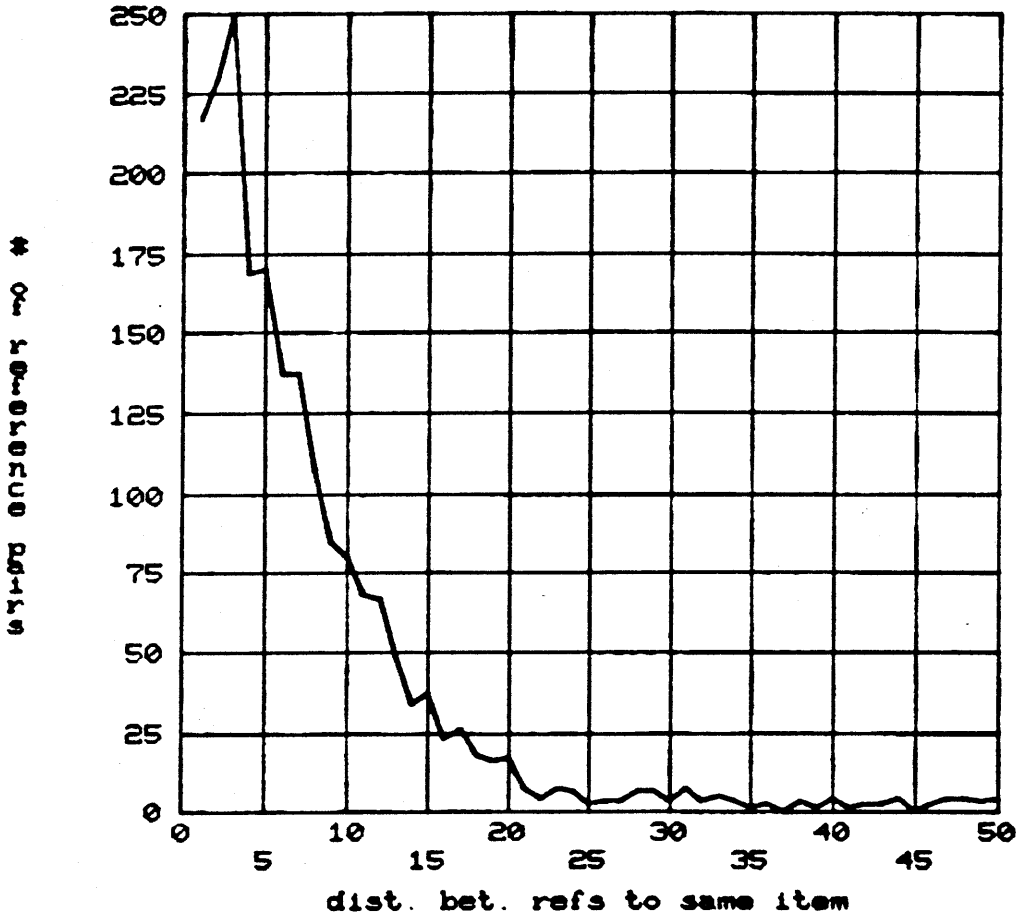
Williams, John G., Experiments in Page Activity Determination, SJCC 40, 739-747 (1972).

Wimbrow, J. H., A Large-Scale Interactive Administrative System, IBM Systems Journal 10, 4, 261-282 (1971).

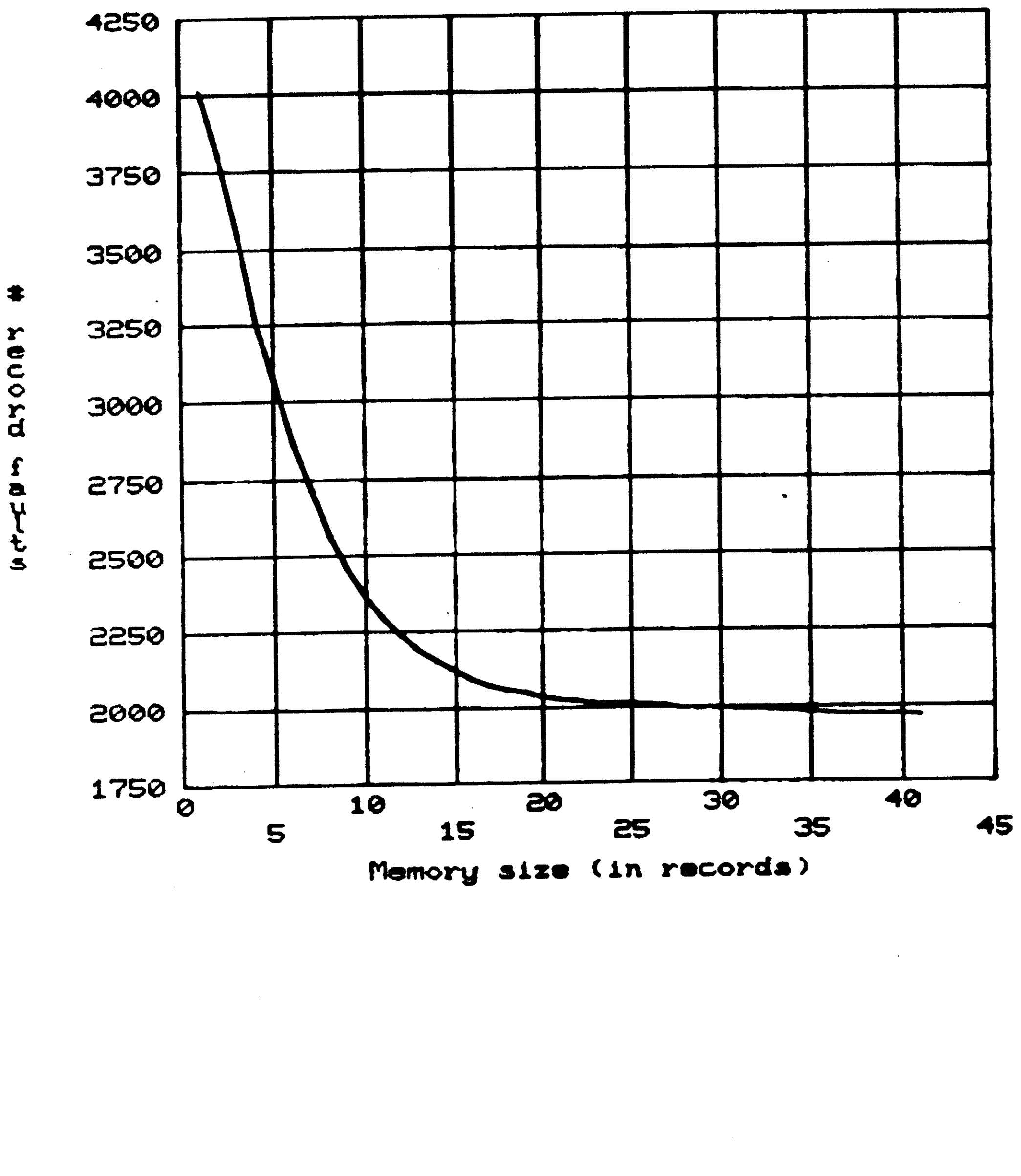
Wonnacott, Thomas H. and Ronald J. Wonnacott, Introductory Statistics for Business and Economics, John Wiley and Sons, 2nd edition, pp. 753 (1977).

Zipf, George Kingsley, Human Behavior and the Principle of Least Effort, Addison-Wesley Press, pp. 573 (1949).

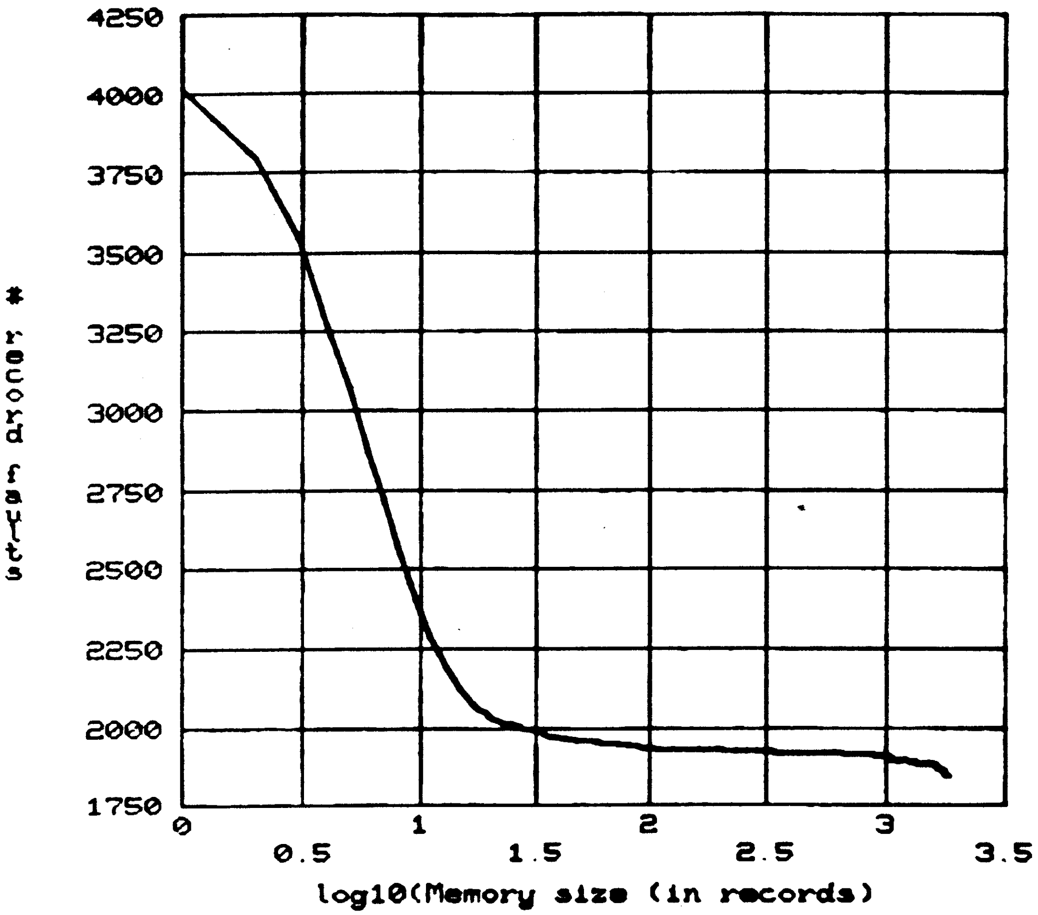
Data from transactions of August 19, 1975



Parachor curve for August 19, 1975



Parachor curve for August 19, 1975


# record faults
log10(Memory size (in records)