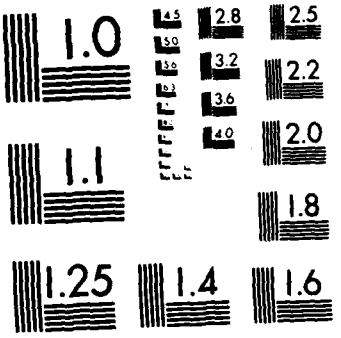
AD-A134 964 FINAL TECHNICAL REPORT FOR CONTRACT N00014-76-C-0944 1/1
(U) MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR
COMPUTER SCIENCE M HAMMER MAR 78 N00014-76-C-0944
UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963 A

Massachusetts Institute of Technology

Laboratory for Computer Science

FINAL TECHNICAL REPORT

for

Contract N00014-76-C-0944

Advanced Research Projects Agency

Department of Defense

Office of Naval Research

Principal Investigator: Prof. Michael Hammer

March 1978

This report is intended to serve as the final technical report for contract number N00014-76-C-0944 of the Advanced Research Projects Agency, monitored by the Office of Naval Research.

Under the sponsorship of this contract, we have conducted research in two areas of data base management: self-organizing data bases and automatic data error detection and correction. This letter summarizes the results of our work; further technical details are provided by the papers and technical reports listed in the bibliography.

Self Organizing Data Bases

In contemporary data base management systems, the full responsibility for physical data base design, the process of choosing storage structures and access mechanisms for a data base, falls on the human data base administrator (DBA). The choice of the physical design for a data base will determine the performance of application systems that use that data base, since the data base system's transaction processor must utilize the structures selected by data base design. The objective of the physical design process is to choose storage structures and access methods that can be used by the data base system to effect efficient execution of the mix of transactions for which the data base will be used.

The nature of a good physical design for a data base depends on the kinds and frequencies of the transactions that will be conducted against it; on numerous characteristics

of the data itself; and on the method of operation of the data base system. In theory, the DBA should gather information on the global usage pattern of the data base, and then consider all alternative physical designs for it, assessing for each one the performance it will provide in the given usage context. In reality, this approach is not feasible. It is very difficult for the DBA to gain an accurate model of the overall usage of a large shared data base; furthermore, a DBA rarely possesses the accurate model of system operation needed to accurately evaluate proposed designs; and in any event, the number of possible designs is generally too large to allow such an exhaustive analysis.

Furthermore, a shared data base is not a static entity; as applications evolve, its pattern of use will shift. Consequently, a data base must be regularly redesigned; its structure must adapt to change, and must be matched to future, rather than past, usage.

Given the complexity of the problem of selecting a good data base design, it is not surprising that contemporary data bases, whose structures are selected by human DBA's, are often poorly designed. In response to this situation, the notion of a self-organizing data base system has been developed; in this approach, the responsibility for data base design is transferrred from the DBA to the data base system. The system gathers the information needed to select a good design and systematically utilizes it to construct a design well-matched to the observed context.

We have achieved several important results in this area of self-organizing data

bases. First, we have developed a complete and realistic architecture for a self-organizing data base system. Earlier studies of this problem have either ignored key issues or have so simplified the context as to make their results meaningless in realistic environments. Our architecture is organized around four principal modules: a parameter acquisitor, which monitors the actual usage of the data base and gathers statistics on the global usage pattern of the data base and on its internal characteristics; a forecaster, which projects these observed statistics into the future; a file design evaluator, which assigns a figure of merit to a proposed data base design in the context of a given usage pattern and the data base's internal characteristics; and a heuristic design proposer, which identifies a small set of promising data base designs to submit to detailed evaluation, thereby avoiding the need for analysing the full set of possible designs. The file design evaluator in turn utilizes a transaction cost estimator, which assesses the cost that the data management system would incur in processing a transaction against a data base that has specified internal characteristics and that is organized in the proposed way. This cost estimator utilizes a model of the data base system's performance to determine how the system would process the transaction in question; the data base characteristics determine the cost that this processing would incur.

Our proposed architecture has the following features: it accounts for the fact that the usage of a data base is not static but changes over time, thereby requiring that the data base design also evolve; it is sufficiently flexible to incorporate a variety of data base costs, including transaction processing, storage requirements, data base reorganization, and

application program retranslation; it does not significantly degrade data base performance during normal operation; it enables a near-optimal design to be located at acceptable cost, and it allows the DBA to trade off the quality of this design against the cost of finding it; it is sufficiently modular to allow for individual components to be separately constructed and utilized by a DBA; and it can accomodate as accurate a model of the data base system's performance as desired.

In addition to defining this overall architecture, we have specified the basic structure of each of these principal modules, and have implemented prototype versions of some of them. Our design for the parameter acquisitor enables it to gather detailed information that constitutes an accurate yet manageable picture of data base usage, and to do so without excessive storage or processing requirements. Our approach to usage prediction is based on the application of exponential smoothing techniques that have been developed in the context of inventory control; these techniques allow for accurate and efficient forecasting, based on the detection of trends.

We have developed general data base modelling techniques that will allow for the construction of transaction cost estimators for a variety of data base management systems. In order to instantiate, test, and evaluate these techniques, we have applied them to the construction of a cost estimator for a specific data base system, the Datacomputer. This effort has been based on a careful analysis of the design and operation of the Datacomputer. A preliminary version of this cost estimator has been implemented and tested; results

indicate that the cost figures it predicts for the processing of a transaction are very close to the actual costs incurred when the transaction is submitted to the Datacomputer.

Selecting the physical design of a data base entails the resolution of a potentially large number of individual design issues; which ones are relevant in a particular context depends on what access structures the given data base system can support and utilize. We have studied two of these design problems in detail: the selection of secondary indices for a file, and the partitioning of a logical file into separate physical subfiles. These are concerns that are relevant in a large number of different data base systems. In each case, our aim was to build a design proposer that would enable a near-optimal resolution of the issue to be made without conducting an exhaustive search of the full space of possible designs; these spaces are so large to prohibit their complete searching even for moderately sized data bases. To achieve this end, we have developed heuristic search techniques for navigating through large design spaces, and have implemented design selection systems that embody them. Each such system consists of a design proposer and a design evaluator; it takes as input the definition of a data base, the specification of its internal characteristics, and a usage pattern for it. The proposer and the evaluator cooperate to resolve, in that context, the particular design issue under consideration.

We have experimented with these systems in order to assess two aspects of their performance: the quality of the designs that they select and the efficiency with which they choose these designs. The former has been established by comparing the designs selected by

our heuristic systems with those chosen by exhaustive design selection systems. An exhaustive system submits every possible design to the design cost estimator, and thereby finds the truly optimal design; this establishes a benchmark against which the results of an heuristic system can be measured. The exhaustive system is, of course, far too slow to be usable in an operational environment; this fact also places a limit on the extent of the testing for which it can be used. Nonetheless, we were able to conduct a series of experiments with moderately complex data bases and a wide variety of usage patterns and data characteristics. The results of these experiments have been extremely encouraging. In virtually all of the cases that we considered, our heuristic system selected the same design that the exhaustive one did, and at a minute fraction of the cost. In the few cases where the heuristically selected design was non-optimal, its performance (as measured by the design evaluator) differed from that of the optimal design by only a few percent.

The absolute performance of the heuristic design system also suggests that our approach is a viable one. In the experiments that we conducted, the heuristic design system never required more than a few seconds of computer time to select its design. Furthermore, the growth rate of the system's operating time was only slightly worse than linear in the complexity of the data base (as opposed to an exponential growth for an exhaustive system). These facts combine to suggest that for realistic data bases, the heuristic system could resolve either the indexing or partitioning problems in a small number of minutes; this is a completely adequate performance, since the selection of a design is done infrequently and in an off-line environment. (By contrast, an exhaustive system would require many hours of

computer time to resolve the same issue.)

The adequacy of our heuristic design systems can only be conclusively proven by their continued use in operational settings over an extended period of time. Nevertheless, the laboratory experiments that we have conducted strongly suggest the viability of both the general architecture of our design system and of the specific heuristic search techniques that we have devised.

In summary, then, the products of our research in this area include the following:

1) an architecture for a self-organizing data base system;

2) designs for the individual modules of such a system;

3) modelling techniques to enable the prediction of the costs that a data base system will incur in processing a transaction against a data base organized in a specified way;

4) a preliminary system, based on the foregoing principles, for estimating the cost of transactions with the Datacomputer;

5) heuristic design techniques for accurately resolving the index selection and file

partitioning problems at acceptable cost;

6) a prototype system, based on these techniques, for resolving these issues for specific model data management systems.

In addition, the fourth and sixth items could be readily combined to construct an index selection system for the Datacomputer, which could be used in designing data bases residing on that system.

Although we believe that we have made substantial inroads on the problem of improving the data base design process through the use of self-organizing systems, further efforts will be required in order to realize the full potential of this approach and to transfer our research results to operational settings. Among the problems that remain are the following:

1) Generalization of our modelling techniques to allow for the cost estimation of a larger class of transactions against a wider variety of data base organizations.

2) Extension of our cost estimation facilities to account for additional measures of transaction processing costs. Our current system uses the number of I/O events (page faults) as the cost figure for processing a transaction. This should be extended to account for processor utilization; these two figures should also be combined in a way

that accounts for the total system load to produce an estimate of the real time that will elapse during the processing of a transaction.

3) Development of techniques for evaluating a proposed design in the context of a large usage pattern. Our current design evaluator invokes the transaction cost estimator for each transaction occurring in the usage pattern. Techniques for extracting a small number of representative transactions, appropriately weighted, from a usage pattern will be needed in order to handle a pattern comprised of a very large number of different transactions.

4) Assessment of our techniques for parameter acquisition and forecasting. It is necessary to determine if our proposed methods can indeed effectively capture and project the global characteristics and pattern of use of a large shared data base. Such an assessment can only be accomplished through analyses of the usage histories of several operational data bases.

5) Development of heuristic design techniques for complete, integrated data base design. In modern data base systems, the data base design problem has a great many aspects, all of which interact. Our efforts to date have separately focused on two individual design issues. This work must be extended to address the simultaneous resolution of these and the many other issues that a data base designer must confront. The space of possible designs becomes enormously larger in the presence of so many

degrees of design freedom, and extensions to our heuristic techniques will be needed to cope in this environment.

6) Application of our transaction cost estimation techniques to additional operational data base systems, and the development of data base design systems for them.

Our group is continuing research in several of these areas, especially the first, second, and fifth items on this list. We are currently extending our cost estimator for the Datacomputer to handle additional file structures and to provide richer cost figures. We are also studying the problem of synthesizing a complete data base design in the context of a wide variety of available file structures and access methods.

Error Detection and Correction

The problem of data quality and accuracy is a serious one in contemporary data base systems; despite major advances in the capabilities of data management systems, much of the data contained in the data bases they manage is erroneous. There are a variety of sources for such data faults; as a rule, they derive from errors made in the recording, transcription, or transmission of new data, which, by the time it reaches the data base to which it is being submitted, has been corrupted and no longer expresses the information it was meant to represent. The current state of the art in the detection of erroneous data is based on the use of edit routines, which are programs, manually constructed by a

programmer, that inspect incoming data and apply to them a set of reasonableness tests that ascertain if the data are implausible and consequently erroneous. This approach requires a substantial programming effort, and so results in error detection systems that are unreliable, incomplete, costly to construct, and inefficient to operate. The principal focus of our research in this area has been on the development of new technology to address and alleviate the error detection problem.

Our research has led to several important results. The first, and the one from which the rest derive, is a general and thorough analysis of the nature of the problem of erroneous data, and of possible solutions to it. We have examined the different kinds of errors that may afflict a data base; their sources and causes; the effects that they can have on the operational and administrative functions that utilize the data base; the goals of, and limitations on, a data error detection system, and the trade-offs between the two; the relationships among error detection, error correction, and error prevention; principles on which to base the design of an error detection system, and their consequences for general information system design; strategies for detecting errors in data; and a particular methodology (and associated system architecture) for automating the error detection function. Although some of the issues that we explored had been considered in the past, our work in the field was the most thorough and also the first to systematize the area and to produce a document summarizing its principal issues.. The programmatic overview that we produced not only served as the basis for our further research in this field but also represents a compendium of techniques and guidelines applicable to current operational problems.

Our architecture for an automatic error detection system is based on the use of rules (also known as constraints or assertions) that define the legitimate configurations of the data base; any data whose incorporation into the data base would cause the violation of any of these rules is defined as being erroneous. Since a data base is not just a collection of values, but is a model of some application system, these rules derive from the natural structure and limitations of the physical system being modelled. A responsible human authority (such as the data base administrator) defines the set of rules for his data base and submits them to our system, which is then responsible for determining whether incoming data item violates any of the rules. The advantages of this approach derive from the fact that it is easier to specify a set of rules than a set of procedures, that a set of rules can readily be modified should circumstances warrant it, that a set of rules can be examined for completeness and consistency, and that an automatic system can apply rules to detect erroneous data in a reliable fashion.

Our first result in designing this rule-based system was the determination of the requirements for a constraint expression language and the specification of a particular language meeting these requirements. In our language, a constraint has three principal components: the rule itself; a statement of the occasions on which the rule must hold; and the specification of the response that must be taken when a violation of the rule is detected. Some rules should be verified after each update to the data base; for others, this would result in false errors deriving from a transient state occurring during a batch of updates.

Our language identifies the possible occasions on which a rule must hold and enables the rule specifier to select the one most appropriate for each case. We also provide the specifier with a variety of possible responses to detected violations, ranging from allowing the update to proceed with a signal being sent to an appropriate authority to invoking an arbitrary user-supplied procedure.

Our specification language for the rules themselves is based on a particular analysis of the structure of data base constraints. First, we impose a taxonomy on the universe of constraints based on their uses. Furthermore, we do not view a constraint as an arbitrary, unstructured predicate on the state of the data base; we see it as limiting the values of certain objects in the data base (called the constrained objects) in terms of an expression involving certain other objects (called the constraining objects). We also provide a measure of the complexity of a constraint in terms of the relationships that hold between the constrained and constraining object. This structure and taxonomy for constraints provides a guide to a rule specifier in preparing his rules and is also of value to a system checking incoming data against the rules. Although our language is oriented towards the expression of constraints for relational data bases, its principles are applicable to other environments as well.

The major problem facing a rule-based approach to error detection is that of implementation efficiency. The brute-force approach, i.e., reevaluating each of a large set of complex predicates on the occasion of each update to a data base, is likely to be unacceptably

inefficient. We have devised techniques for automatically synthesizing efficient error detection procedures from declarative constraints; these are procedures that will be invoked when data is submitted to the data base. The techniques that we have devised generate procedures that are competitive (in terms of efficiency) with those that might be produced by human programmers. These procedures will only check for the violation of a constraint if the incoming data can possibly cause its violation; data that are obviously irrelevant to a constraint do not cause its reevaluation. Furthermore, potentially erroneous data is first subjected to inexpensive tests that may determine that it does not violate the constraint; only if the data fails these tests will the actual constraint be evaluated. For example, consider the constraint that "no employee makes more than his manager". Our system would observe that the only data base transactions that can affect this constraint are those that change an employee's slaray or his manager; in all other circumstances, the constraint is sure to remain unviolated. Furthermore, if the change to an employee's salary causes it to be reduced, then again the constraint cannot be affected and so need not be evaluated; only if the employee's new salary is greater than his old one does the constraint actually have to be evaluated by retrieving the manager's salary and comparing it to the employee's. The procedure generated to inspect incoming data would incorporate these tests and would be organized as just described.

Our techniques for generating efficient data checking procedures are based on the careful logical analysis of the given constraints and the determination of the potential impact that the different kinds of transactions with the data base may have on the

constraints. Therefore, our techniques are representation independent and so are applicable to a wide variety of data base systems that utilize different physical structuring and access methods. We are in the process of constructing a prototype system, based on these techniques, for generating efficient data checking procedures.

We have also conducted research on the problem of automatic data error correction. The most conservative response to take upon the detection of an erroneous data item is to return it to its originator and request its resubmission. However, in many instances, this cautious approach is unsatisfactory, and it would be appropriate to attempt to deduce from the faulty data what the original value had been prior to its corruption. We have developed artificial intelligence techniques for accomplishing this goal in some contexts; obviously, this approach has its limitations and must be exercised with caution, but in certain circumstances it can be of value in enhancing data quality.

Our approach to error correction is based on the premise that errors are not caused by magic, but rather are introduced into data by a finite set of well-defined error mechanisms. The problem of error correction then becomes one of identifying the precise locus of an error in a data item, and of determining which mechanism was responsible for this error; this mechanism can then be "inverted" at the locus, and the original value recovered. Our analysis of a faulty data item is based on the observable properties of the item, the a priori likelihoods of occurrence of the various error mechanisms, and on the a

posteriori likelihoods of the several candidate "corrected" values. This latter factor is based on the fact that in most situations, a data value cannot simply be classified as "reasonable" or "unreasonable"; instead, it must be assigned a value that measures the degree of its plausibility. As we have conceived it, the automatic data error correction problem is akin to that of computerized medical diagnosis; indeed, our techniques resemble some of those proposed in that field.

We have developed the principles of an error correction system based on these concepts, and have designed and implemented an initial prototype of such a system. A second, more complete version is currently under development and will be subjected to extensive testing.

In summary, the major results of our research in automatic data error detection are as follows:

1) a programmatic analysis of the issues and principles underlying the detection, correction, and prevention of errors in data;

2) the development of a rule-based methodology for error detection and an associated architecture for an automatic system;

3) the design of a structured language for the systematic expression of constraints,

including facilities for specifying the time when the constraint is to be verified and the action to be taken on the detection of its violation; the language is based on a model of the nature and structure of constraints;

4) the development of techniques for synthesizing efficient error checking procedures from declarative constraints;

5) principles for a systematic approach to automatic error correction, based on a model of the error-making process.

We believe that our work represents a significant advance towards the goal of improving data quality through the use of rule-based automatic error detection systems, that our research can be exploited by others working in the field, and that it will eventually be incorporated into operational systems. However, there remain several important issues that must be addressed before powerful, intelligent error detection systems become a reality. Among these are the following:

1) Development of techniques for testing a given set of constraints for completeness and consistency. Although it is more convenient to inspect and understand a set of rules than it is an equivalent set of procedures, it is by no means straightforward to determine if a given set of rules contains an inconsistency or is incomplete in some way.

2) Design of a methodology that can guide the DBA in specifying the set of constraints for his data base. The most effective way to achieve a complete and consistent set of constraints is to have the constraints produced in some principled and systematic way that is based on the semantic structure of the application.

3) Extension of our constraint specification language to express "fuzzy" aspects of a problem domain that cannot be modelled by simple boolean-valued predicates. A way of capturing such information and of utilizing it in assessing the correctness of a data value will be required. We currently employ one such technique in our error correction system.

4) Construction of heuristic technques for checking constraints that cannot be directly verified at acceptable cost. The techniques that we have devised represent efficient, but nonetheless complete and deterministic, methods of checking constraints. Yet in some cases, because of the inherent complexity of the constraint or because of a high data input rate, the most efficient method possible for exactly checking a constraint may be unacceptably costly. In such cases, alternative error checking techniques are necessary, which sacrifice same accuracy in exchange for enormous gains in performance; in particular, such techniques may achieve the desired level of efficiency at the cost of delayed detection of some errors. One possible approach to this would utilize a spectrum of truth values for a constraint ranging from "false" through "barely true" to "very true". If a constraint is found to be "very true", then it is probably safe to

suspend its checking for some period of time.

5) Development of mechanisms for automatically synthesizing constraints, and of means for checking them, based on observed historical patterns of data. Even if the DBA, by following a prescribed methodology, succeeds in identifying and expressing all the constraints on the data base of which he is aware, there may be other facts and relationships of which he is unaware but which can be extracted from the actual data and exploited in analyzing incoming data. These might include trends and other kinds of dynamic information that are difficult to determine in a static fashion. Such information may constitute new constraints or may determine efficient methods of checking existing ones.

We are currently continuing our investigations in several of these areas. However, there is another dimension in which our work on error detection can be extended. Our approach to error detection is based on using knowledge of the semantics of the data base to determine implausible (and hence incorrect) data; this represents the first systematic exploitation of data base semantics to provide an advanced functional data management capability. Our work in this area should serve as the basis for new technology to support a variety of "intelligent" data management functions and thereby underlie a new generation of decision support systems. The possible applications of data semantics range from sophisticated user assistance systems, which help a user determine and specify the query he wishes to pose to a data base, to

heuristic query optimizers, which determine an acceptable (though perhaps imprecise) low-cost response to a user's complex query. We are currently building on our work in error detection and are investigating several of these issues.

Bibliography

1. Hammer, M. "Self-Adaptive Automatic Data Base Design." Proceedings of the 1977 National Computer Conference, June 1977.

2. Hammer, M. and Chan, A. "Index Selection in a Self-Adaptive Data Base Management System" Proceedings of the 1976 SIGMOD International Conference on Management of Data, June 1976.

3. Hammer, M. and Chan, A. "Acquisition and Utilization of Usage Patterns in Relational Data Base Implementation." Proceedings of the IEEE Joint Workshop on Pattern Recognition and Artificial Intelligence, June 1976.

4. Hammer, M. and Chan, A. "The Heuristic Selection of Secondary Indices", to appear in ACM Transactions on Database Systems

5. Chan, A. Index Selection in a Self-Adaptive Relational Data Base Management System. M.I.T., Laboratory for Computer Science, LCS/TR-166, 1976.

6. Niamir, B. Attribute Partitioning in a Self-Adaptive Relational Database Management System. M.I.T. Laboratory for Computer Science, LCS/TR-192, 1978.

7. Hammer, M. and Niamir, B., "Heuristic Determination of Optimal File Partitions", to be published.

8. Chan, A. and Niamir, B., "Estimating Record Accessing Cost in Blocked Database Organizations", to be published.

9. Hammer, M. and Berkowitz, B., "A Transaction Cost Estimator for the Datacomputer", to be published.

10. Hammer, M. "Error Detection in Data Base Systems." Proceedings of the 1976 National Computer Conference, June 1976.

11. Hammer, M. and McLeod, D., "Semantic Integrity in a Relational Data Base System." Proceedings of the ACM International Conference on Very Large Data Bases, September 1975.

12. Hammer, M. and Mcleod, D. "A Framework for Data Base Semantic Integrity." Proceedings of Second International Conference on Software Engineering, October 1976.

13. Hammer, M. and McLeod, D. "The Semantic Data Model: A Modelling Mechanism for Data Base Applications", to be published in Proceedings of the 1978 SIGMOD

International Conference on Management of Data, June 1978.

14. Hammer, M. and Sarin, S., "Efficient Monitoring of Data Base Assertions", Proceedings of the 1978 SIGMOD International Conference on Management of Data, June 1978.

15. McLeod, D. High Level Expression of Semantic Integrity Specification in a Relational Data Base System, M.I.T. Laboratory for Computer Science, LCS/TR-165, June 1976.

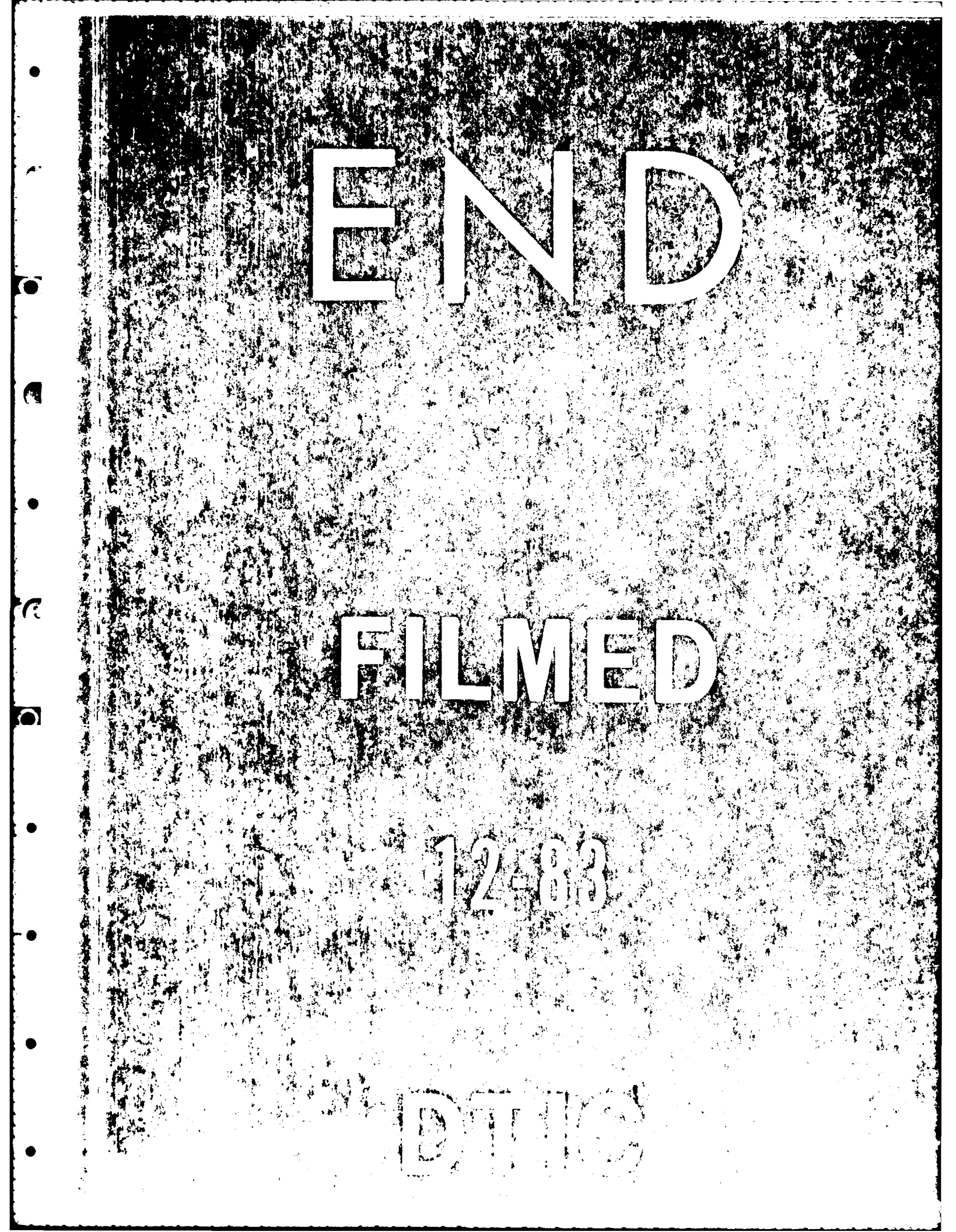
16. Sarin, S. Efficient Checking of Database Semantic Integrity Assertions, M.I.T. Laboratory for Computer Science, LCS/TR-198, April 1978.

Contract N00014-76-C-0944

Financial Position

Total Contract Value	$231,000
Total Expenditures and Commitments	225,716
Unexpended Balance	5,284

This information is based on the MIT Account Statement and may change after completion of final audit and reconciliation. A final financial report will be prepared by the MIT Comptrollers Accounting Office.


END
FILMED
12-83
DTIC