Error detection in data base systems*

by MICHAEL HAMMER
M.I.T. Laboratory for Computer Science
Cambridge, Massachusetts

ABSTRACT

Incorrect data poses a serious impediment to the effective use of computerized data bases. Conventional approaches to the design and implementation of automated data error detection systems are inadequate for large and complex data bases. Partly, this derives from the inherent intricacy of the problem, with decisions being required as to what checks to perform, how and when to do the checking, and how to respond should an error be found; writing procedures to accomplish these functions is a difficult programming task. Also at fault is the unrealistic and overly simplistic view of data correctness embodied in most contemporary systems.

"Intelligent" data checking systems are required, which possess more extensive knowledge of the data base environment. They will need to understand the structure of the world which the data base models; the way the data base is used, and the relative importance of its various components; the sources of the errors that might occur and the costs of detecting them; and the patterns and rates of errors that actually do occur. Such a system would then be in a position to detect a wide range of errors, allocating its resources in a systematic fashion and responding appropriately to different error situations.

INTRODUCTION

It is a truism that any decision-making system is only as good as the data which it uses; this is especially true of computer-based information systems. Yet while the number and complexity of such information systems have been dramatically increasing in the last few years, supported by great enhancements in the technology of data base management systems, comparatively little has been done to assure the quality and reliability of the data on which these systems depend. Many computerized data bases in use today suffer from high error rates in the data they receive, and are consequently riddled with bad data. And with incorrect data, the most efficient and sophisticated system is well-nigh useless.

There are numerous potential sources of error in a computer data base. Some of these originate in the computer system itself, deriving from such causes as hardware failure or interfering concurrent users; however, these are generally secondary sources and can be controlled by existing technology. The major error problem lies in data that arrives at the computer already corrupted. The possible reasons for this corruption are manifold, including human error in the initial recording or formatting of the data; faulty transcription of the data, either by human or machine (e.g., OCR scanners); accidentally lost, forgotten, or delayed data; deliberately falsified or omitted data; and the like. It is also useful to include in our notion of erroneous data those cases where no error has occurred in the transcription or transmission of the data, but rather where the data faithfully represent an illegal or impossible event. That is, the error is not in the data, but in the information it conveys.

In this paper, we consider the problem of detecting errors in data bases. We begin with a re-examination of the concept of data errors, and consider both the problems that they can cause and the difficulties that can arise from attempts to detect them. This leads us to consider the issues and problems of error detection, and the inadequacies of the current technology in this field. We proceed on the basis of this analysis to propose an alternative way of thinking about data integrity, an approach which has major implications for the design of error detection systems.

DATA ERRORS AND DATA INTEGRITY

It is worthwhile to re-examine the concept of error in the context of a data base. A data base is not just a collection of values; it serves as a model of some limited universe. At every point in time, the contents of the data base represent some configuration of that application world. Every such world is governed by some set of rules that determines the legitimacy or plausibility of its states, that specifies which configurations of

* This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under contract number N00014-75-0661.
enounter the problem of localizing the error: though we know which record is faulty, we have to determine which record is the one causing the problem. We may have to draw upon additional information to resolve this.

3. An inconsistency between field(s) of one record and fields(s) of related record(s). For example, there might be a rule that no individual’s salary may be greater than that of his manager. Should this rule be violated by the data base at some point, the error could only be localized to one of two possible offending records.

4. Some global pattern out of order in some set of records or in the data base as a whole. This would be a violation of some restriction, not on individual entities of the application domain, but on collections of them. These global patterns may involve aggregate functions (the average salary of all employees should not exceed $12,000), functionality relationships (every department has exactly one manager), and subset requirements (every manager is also an employee). These global pattern violations usually indicate not the presence of some particular faulty value, but rather of a general trend which violates expected norms. In these cases, it is often not the data that is wrong, but the events that they report, or sometimes even the the rules which define right and wrong in this environment.

5. Missing data. This notion includes blank fields, obsolete values (the new data is missing), entire records which are referenced but cannot be found, and lacking data which cannot be localized.

DATA ERRORS VERSUS DATA CHECKING

The hazards of allowing erroneous data into a data base hardly need belaboring. They include the obvious results of improperly executed operational activities and faulty decisions based on incorrect information. But bad data can also cause the malfunction of application programs that use the data base and can even degrade the performance of the data management software itself, sometimes to the point of failure; this occurs because programs frequently implicitly assume that the data which they manipulate are semantically reasonable and consequently satisfy various criteria. On a more systemic level, erroneous data can destroy the confidence of an organization’s personnel in the entire information system, with a deleterious impact on morale; frequently the result is an atmosphere which fosters even higher error rates, due to negligence and disinterest.

While it is easy to recognize the dangers of allowing erroneous data into a data base, there are also potential hazards in attempting to prevent this eventuality from occurring. The major pitfall in this regard is an overzealous commitment to the notion of data correctness, whereby the error detection process becomes an end in itself. This condition results from a loss of perspective, wherein the actual dangers of the errors that do occur are not accurately assessed nor the costs of...
their detection weighed against the damage they can cause.

In reality, not all data errors are of equal significance in the context of a particular information system. Different parts of a data base are used for different purposes, and consequently have unequal degrees of importance; this ought to be recognized by the error detection system. Any aspect of the data base may be said to have both a “sensitivity” and an “impact” with respect to its being in error. For example, the salary field in a personnel file is very sensitive to error; any faulty value will give rise to an improper action (namely, the cutting of an incorrect salary check). On the other hand, the age field, if used primarily in projecting future manpower requirements, is comparatively insensitive; it would take many erroneous values to diminish the utility of the aggregate of the age field values. Yet the potential impact of the latter error situation may be much greater; a bad high level decision, in contrast to one inaccurate action. It would be appropriate, then, for a data checking system to expend its resources in proportion to the importance and severity of the errors it is trying to detect.

If the foregoing idea is not appreciated, a disproportionate amount of system resource may be invested in error checking, well beyond what it realistically deserves. This investment may manifest itself in the expense of constructing a powerful and “complete” error checker, or in the actual processor time allocated to this function. There may be more insidious side effects as well: over-emphasis on error detection may impair the performance of the data management system as a whole. Excessive timidity in the presence of possible errors can delay the entire decision-making process, or may even reduce the over-all functionality of the whole information system. On the other hand, an unwarranted over-confidence in a faultily performing error checking system can lead to a false level of confidence in a data base that may be in fact contaminated with much bad data.

CONVENTIONAL APPROACHES

The state-of-the-art in automatic error detection/correction has not been highly developed. Most data editing that is currently done is usually confined to the simplest validity checks, assuring that various fields are of the appropriate data types or are within specified ranges. The structure of most error systems is that they are comprised of a collection of hand-coded, special purpose procedures (written in assembly code or some conventional programming language), each of which is designed to monitor some particular transaction with the data base, and assure both that the transaction is legal and that it leaves the data base in a consistent state. These programs are generally entirely ad hoc, and follow no well-defined approach or methodology.

This approach to error detection does not satisfac-

torily address the problem of complexity. Any data base that represents a real world system of interest is almost certain to be governed by a large and rich set of complex rules which delimit the legality of its data values. To reliably and effectively capture these rules in a disconnected set of procedures written in a conventional programming language is a task that strains the effective capability of contemporary programming technology.

In consequence, current data checking systems are frequently very expensive to build and, once built, are often highly unreliable. Furthermore, they are almost impossible to modify in an orderly and consistent way in order to meet new requirements, either in terms of evolving definitions of the errors to be detected or in modified characteristics of the operating environment.

Conventional methods of error detection usually check for error conditions on the occasion of every (primitive) update (insert, modify, delete) to the data base. This is expensive and can also lead to anomalous results. Consider for instance the restriction that no employee is to make more than his manager, and a series of transactions that increments everyone's salary by 8 percent. If an employee's raise is processed before his manager's, application of the test would compare a new salary with an old one and might raise a spurious flag, reflecting a transient condition in the data base.

In addition, current error systems are cumbersome, relying on primitive brute-force techniques to do the data checking; in general, they operate in a very inefficient manner. Partly this is a consequence of the complexity problem; the incorporation of any but the simplest techniques in the system structure described above might topple the entire structure. In another sense, this clumsiness is a result of the rigid and inflexible view of data correctness alluded to earlier. In this view, every datum is viewed as being either correct or incorrect, and all errors are assumed to be catastrophes of equal magnitude. There is no appreciation of the fact that while some fields must be error-free, others (because of the way they are used in the information system) can tolerate some measure of error; nor does it take into account the concept that the severity of an error might be measured in terms of its deviation from a legitimate value. Thus there is usually little relation between the importance of an error and the amount of resource expended to detect it.

Nor does the conventional system structure provide a systematic capacity for response to an error situation. Once the presence of an error has been detected, it is then necessary to localize the error and specify exactly which data value is at fault; in many cases, the state-of-the-art is incapable of doing this satisfactorily. After an error has been identified, there are several possible responses. The options include rejecting the erroneous data; allowing it into the data base, signalling its occurrence and/or marking it as questionable; putting it
into a suspended status, pending the receipt of additional clarifying information; attempting automatic error correction. The choice of response should be determined by the importance of the error, the reliability of the error detection procedure, and the particulars of what causes the error. However, current systems are inflexible in this regard. They are especially weak in automatic error correction, relying on ad hoc techniques which can easily go astray.

All of these problems are greatly exacerbated in the context of large data bases with high data input rates. There the requirements of the data checking system can easily overwhelm the available system resources. These circumstances have resulted in data bases that are full of bad data despite extensive data checking, and whose utility is grossly restricted because of the unreliability of much of the data.

In some applications, the current technology is adequate, because of the simplicity of the world of application and the consequent limited range of possible data errors. However, the current state-of-the-art will not be able to meet the demands of data bases of increasing size, complexity, and sensitivity. As data bases are used to model ever larger and richer domains, and are relied on for more critical decisions, the need for reliable data will be ever more pressing and more difficult to satisfy.

AN ALTERNATIVE APPROACH

In view of the problems that have arisen as a result of a non-systematic approach to error detection, we propose the following basic principles to be adhered to in the design of error detection systems.

1. Expect errors, and be prepared to fail because of them in a "soft" manner. Do not design the kernel of the information system under the hypothesis that all data is correct, and then delegate to an error-checking subsystem the responsibility of bringing about that state of affairs. The goal of perfectly correct data is unattainable one. We can minimize the presence of certain errors subject to various constraints, but cannot expect to eliminate them altogether.

2. Formulate the rules of the application world first; then relate them to the constructs of the data base; and then identify potential logical errors in terms of the data base transactions. Attempt to ensure the completeness and consistency of the world model before proceeding further.

3. Evaluate the importance of the various error types with respect to the decision process, considering both sensitivity and impact. Allocate the resources of the data checking system in line with this evaluation.

4. Anticipate the kinds of errors that may occur, identifying their sources, estimating their relative frequencies, and determining the costs of defecting them. Be prepared to adapt to changes in any of these factors: in the definitions of what constitutes an error; in patterns of actual error occurrence; in relative importance of errors; and in the costs of error detection. Learn from historical situations and detect emerging patterns as well.

We believe it is possible to construct an error-detection system in accordance with these principles, and are in fact currently engaged in such an effort. Below, we summarize the most salient features of the system design.

1. The system will be declarative, rather than procedural. That is, the rules governing the legitimacy of data values will not be implicitly embedded in the code of the procedures that perform data-checking, but will be stated as explicit assertions by an individual knowledgeable in the domain which the data base is modeling. The primitives used in the expression of these rules (called constraints) will be chosen to be naturally descriptive of the kinds of worlds most frequently represented in data base systems; thus a non-programmer would be able to describe the constraints directly, without programmer intervention.

The declarative approach has numerous advantages over procedural techniques, besides ease of expression. Because the rules are all listed in a single place, it is possible to make modifications to them in a reliable and simple fashion. Since the procedure that uses the constraints to check the data is only written once, it is possible to verify its accuracy reliably and hence have confidence in its correct operation. In addition, the system (rather than an applications programmer) will bear the responsibility for determining and checking all the appropriate rules for a given transaction. Finally, it will be possible to examine the explicit constraints themselves, looking for incompleteness, redundancy, and inconsistency.

2. The constraints will do far more than express legal ranges of values for specific individual fields; they will describe rich structural aspects of the world in question, thus enabling far more sophisticated error checking than is currently done. In order to detect many kinds of errors of this sort, it is necessary for the checking program to possess a good deal of rich and complex "knowledge" of the domain in question. Conventional data base management facilities, and data editing programs that rely on them, do not have the capability to represent and utilize knowledge of the requisite form.

3. The occasions at which errors will be detected will correspond to "conceptual transactions" or "structured operations" with the data base; these are appropriately selected sequences of data base transactions that form representations of real-world events. It is these complex transformations that actually occur in the world that the data base models, and so it is appropriate to check them (rather than each of their primitive constituents) for validity. This approach will
avoid many spurious situations and will facilitate suitable response to the detection of an error.

4. The system will be designed to operate independently of the actual data base system, and to reside on a remote processor. This processor will be dedicated to the task of data checking, but it will be able to communicate with the data base facility itself by means of a communication link. It will utilize the constraints which model the domain of the data base to examine data to be submitted to the data base; should it detect an error, the offending transaction can be identified and appropriately handled.

In order to rule on the validity of a transaction, the checking processor may need to inspect some of the values in the data base itself; it will obtain the items it needs by issuing a request for them to the data management system over the communication channel, receiving the results in the same way. As time goes on, the checking processor will build up a local version of those parts of the data base which it most frequently requests, thus obviating the need for much of the data transmission.

This strategy opens up several important possibilities. First of all, by detecting errors closer to the source of their occurrence, the opportunity for their timely correction is greatly enhanced; thus the overall performance of the information system is improved.

Furthermore, the data checker can be made relatively independent of the actual data management system which is responsible for the data. Variations in the “back-end” data manager need only have a minor reflection in the operation of the “front-end” error detection processor. This is especially important in an environment where one conceptual data base may be distributed among several data management systems. A final potentiality is for distributed data checking, with multiple front-ends, each processing a separate stream of source data. This can address the problem of checking the correctness of large data bases with high input data rates.

5. The system will adopt a flexible view of the concept of error. While perfectly correct data is a laudable goal, in reality it is an unattainable one, and compromises must be effected in its pursuit. The system will make the necessary trade-offs, in the realm of efficiency versus accuracy, in a systematic fashion. Rather than swamping its resources with a futile and inappropriate effort to detect all errors, the system will concentrate on validating those items that require the highest levels of accuracy. Resources will be allocated to checking other fields based on their relative importance. The data base administrator will provide the system with the information regarding the comparative significance of various data items in the decision making process.

In deciding how to allocate its resources, the system will also consider the effectiveness of various checks. It will have to determine the actual frequencies of various errors; combining this information with an estimate of the cost of detecting each of the errors in question and with a measure of the importance of the item being validated, the system will be able to proceed in an optimal fashion. The system will also be adaptive, automatically adjusting to changing patterns of errors.

6. The system will possess the degree of “world-knowledge” needed to process errors in an intelligent fashion. The first aspect of this is error localization, identifying exactly which data values are at fault. Once an error has been detected and localized, an appropriate response must be made. The range of such responses is great, and the one to choose may depend on numerous factors and, for a given error type, may even change over time. The system will be flexible enough to accommodate many possible responses, and will be in a position to select the right one.

The most ambitious of all responses is, of course, the automatic correction of the error in question, to restore the original value. The system will possess a facility for the data base administrator to provide a model of the error-making process in the system in question, which describes the loci where, and means by which, errors do occur. In those circumstances which the data base administrator identifies, this model can be utilized to reverse errors that do occur.

The ultimate goal of any error system should be error prevention: the identification of the sources of the errors that actually occur, and the instigation of remedies to forestall their further occurrence. The system will utilize the model of the error process, together with observed patterns of error, to pinpoint the origins of the most common errors and identify them to the data base administrator.

In order to achieve the capabilities just outlined, it will be necessary to utilize novel implementation techniques of a statistical character, that will enable the system to attain maximum data reliability subject to the constraints imposed by available resources. For example, the system could sample the incoming data in order to determine the actual rates of different transaction types and errors. If some errors are found to occur extremely infrequently, then those parts of the system used for their detection can be temporarily disabled. This may allow some erroneous values to slip by undetected, but this should be acceptable if the fields in question can tolerate the expected level of error. The sampling should be reactivated from time to time to ensure the constancy of the error rate. This approach results in application of resources where they are most needed. Other useful statistical information would include various summaries of the data base contents (in order to detect emerging global patterns) as well as patterns of error types and erroneous fields. These can be used to pinpoint the sources of the most common errors as well as indicate when an error is not in the transaction but in the master file itself.

To satisfactorily address the manifold demands of
large and complex data bases, it will be necessary to rely on heuristic techniques that gain efficiency at the expense of perfect accuracy. For example, we might associate with various fields of the data base different levels of reliability, based on observed error patterns. These measures can help in localizing an error indicated by a data base conflict. Another idea would be to utilize concepts from "fuzzy logic," and abandon the notion of situations being only correct or incorrect. We might evaluate some predicate at a given time and find it to be "very true", so entirely satisfied that in all likelihood it would take a very long time for a normal series of transactions to make it turn false; we could then reasonably suspend the further checking of this predicate for a long time.

ERROR PREVENTION

Perhaps more important than these techniques which can be used to implement or facilitate error detection, are those measures that should be taken to prevent errors from occurring in the first place, or diminish their potential for harm if they do occur. Primary (and most obvious) of these is the adequate training of those who collect, record, and transmit data. But these individuals must also be provided with incentives to encourage accuracy in the submission or processing of data; someone who feels he derives no benefit from an information system is unlikely to expend much effort on its behalf. There are other "human factors" issues that can have enormous impact on the error rate. For example, the design of the forms on which data is recorded is a very important consideration, which when improperly done, can have disastrous consequences. Such questions as the layout of a form, the size and location of the different fields it contains, the redundancy of the information it carries, and the extent to which it is prefilled, all require careful thought.

The global structure of the information system will also impact the reliability of the data base. For example, it is important to provide multiple levels of protection, so that should an error get past one level of detection it will not immediately get into the data base and linger there forever. One mechanism to achieve this is a feedback system, whereby parts of the data base are directed to those responsible for (or familiar with) them, and then subjected to human review. Distributed data checking is another technique that can contribute to data reliability. It results in more timely error detection and more rapid correction of errors because of proximity of the detection to the source. It also makes for greater system efficiency, with the data base computer only checking for errors that cannot be caught at the source.

The structure of the files that actually comprise the data base also has an effect on the reliability of the data base. Some contemporary information systems are based on a single large file, consisting of a great number of identically structured records, each with many fields. Such a structure does not facilitate the prevention or detection of errors. Every transaction has potential access to all fields of a record and, should it go awry, can have unlimited consequences within the record. Thus, to prevent disastrous situations, all transactions require inspection and validation. A different kind of file structure can be more conducive to data correctness. For example, a multi-file structure wherein critical data is not stored together with less critical information enables us to concentrate our checking resources on transactions that manipulate the former. Similarly, a separation of stable, seldom changing data from that which is highly volatile can prevent erroneous transactions with the latter from influencing the former. Indeed, it may be appropriate to impose a multi-level structure on the data, for example by age. The first level might be the most recent data, not yet fully verified as to its accuracy; the second, the currently active data, into which new data is moved as it is validated; and the third, old seldom-used data, including statistical summaries. The record structure should also be carefully chosen to encourage intra-record checking, and to facilitate inter-record checking when it must be done.

CONCLUSION

In this paper, we have presented the framework of an approach to the construction of "intelligent" error detection systems. Further research is needed to formulate and develop the principles which underlie the advanced approach to error processing described above; having developed these concepts, it will be necessary to assure their viability and practicality by building systems that embody them, and use these systems in realistic problem environments. There are several major conceptual areas which require substantial development in the course of this research: the representational scheme for data base constraints; optimization techniques for efficient data checking; communication between a front-end processor and the back-end data base; measures of criticality and reliability of data, and the heuristics that make use of them; error correction as driven by a model of the error process. Nevertheless, this approach holds great promise for attaining the needed levels of reliability in the large and complex data bases of the future.

REFERENCES

2. Goudem, J. A., "Large Scale Data Base Systems—Current
Deficiencies and User Requirements," in D. Jardine (ed.), 
Data Base Management Systems, North-Holland Publishing 
3. Eswaran, K. P. and D. D. Chamberlin, "Functional Specifi­
cations of a Subsystem for Database Integrity," Proceedings 
of International Conference on Very Large Data Bases, 
4. Graves, R. W., "Integrity Control in a Relational Data 
Description Language," Proceedings of ACM Pacific Con­
5. Hamner, M. M. and D. J. McLeod, "Semantic Integrity in a 
Relational Data Base System," Proceedings of International 
Conference on Very Large Data Bases, Framingham MA, 
22-24 September 1975.
6. McLeod, D. J., "High Level Domain Definition in a Relational 
Data Base System," Proceedings of 1976 ACM SIGMOD-
SIGPLAN Conference on Data, Salt Lake City, UT, 22-24 
March 1976.
7. Stonebraker, M. R., High Level Integrity Assurance in Re­
lational Data Base Management Systems, Electronics Re­
search Laboratory Report ERL-M473, University of Cali­
fornia, Berkeley CA, 16 August 1974.