Information Technology and Management 2, 27–51, 2001
© 2001 Kluwer Academic Publishers. Printed in The Netherlands.

Database Audit and Control Strategies

LEVENT V. ORMAN

Cornell University, Sage Hall, Ithaca, NY 14853, USA
E-mail: L.orman@cornell.edu

Abstract. Database management systems are the primary tools of automated record keeping, reporting, auditing, and control. Although they have significantly improved the efficiency and speed of record keeping, the ability to detect errors and maintain quality has not kept pace. There are three major strategies of error detection, auditing, and control to maintain integrity. The three strategies are introduced, and compared in terms of efficiency and effectiveness in eliminating errors. The optimum timing of audits under each strategy is computed. One strategy is shown to be completely dominated by the others. Hybrid strategies involving various combinations of the three pure strategies are introduced, and their performance computed. Hybrid strategies are shown to outperform the pure strategies under most realistic conditions. Finally, optimum audit strategies are devised, and their parameters computed.

Key Words: database auditing and control, database integrity, optimum timing of audits, audit performance, optimum audit strategies, accounting information systems

1. Introduction

Database management systems are the primary tools of automated record keeping, reporting, auditing, and control [1,8,17,25]. They are used to store and maintain a record of all business transactions, and also create and maintain accounting data that are derived from those raw transactions. As in manual bookkeeping systems, automated record keeping systems have to be monitored carefully to maintain data quality. A large variety of errors can seep into organizational databases, ranging from data entry errors to violations of accounting standards. In fact, there is considerable evidence documenting the poor quality of data in organizational databases [13,20,23]. It appears that although database systems have revolutionized bookkeeping and reporting in terms of efficiency and speed, the ability to detect errors and maintain quality has not kept pace.

The primary tool of database auditing and control is database integrity constraints [8,9,17,25]. They are logical statements that capture the semantics of data and the intent of the transactions. They have to be satisfied by the database for the data to be correct and complete. They can be incorporated into the database for automatic enforcement, as in internal control systems; or they can be created and executed on an *ad hoc* basis by the auditors to test the data quality, as in operational auditing systems. The term audit will be used throughout this article to refer to any procedure that detects errors in the database, whether it is automated or manual, and whether it is for internal control purposes or for operational audits. The timing of audits will also be used to refer to either the timing of automatic control procedures, or the timing of manual or semi-automated operational audits.

In their most general form, the integrity constraints can be maintained and enforced as external views that contain integrity violations. These external views can be considered queries that, when executed, return the data that cause violations of integrity. There is one external view for each database constraint, and they are expected to return empty so long as the database is in a correct state. They are either monitored by the database management system periodically, or even continuously, for internal control purposes [12,18]; or they are created and executed by auditors either on an *ad hoc* basis at periodic audits or on a continuous basis during operational compliance audits [11].

For internal control purposes, a continuous integrity monitoring and enforcement system appears more desirable than a periodic system, but there are serious implementation problems. A continuous, real-time integrity enforcement system requires the execution of all relevant integrity constraints after each transaction that updates the database, to ensure that no violations have resulted from the transaction. Few commercial systems can provide this level of service due to the high cost of executing integrity constraints. Integrity constraints are complex queries that are executed against very large databases, and executing a large number of them after each transaction is a very costly undertaking. In fact, a real-time integrity enforcement system for a large, highly active database can easily overwhelm the system resources and bring it to a halt, or at least cause major unacceptable delays in the processing of transactions [14,16,18,22]. Such delays themselves are significant sources of error in commercial databases due to obsolescence of data, making it impossible to provide an error-free database. All commercial systems are forced to tolerate significant levels of errors in their databases, and finding the optimum level is one objective of this article. The evidence suggests that most commercial databases have very high error rates [13,20,23]. Some actually have very little or no integrity enforcement, rendering the database potentially unreliable. Those that do have integrity enforcement, schedule them infrequently, and only during slow transaction periods [20]. More importantly, most commercial database systems provide no measurement and evaluation of errors and error rates for an effective statistical quality control system for data [2,3,11,15,19]. Operational audits have similar problems. They also use a variety of integrity constraints to test the error levels in the database during periodic audits. They can also use secondary integrity constraints in real time, to test the effectiveness of the primary integrity constraints used by internal controls. Secondary integrity constraints are more likely to be the tests of aggregate error levels, while the primary constraints are directed towards catching and correcting individual errors.

The objective in this article is to provide a methodology for statistical quality control for erroneous data in commercial databases. The methodology is used to compare various integrity enforcement techniques (also called auditing techniques). Under each technique, the optimum timing of database audits, and the optimum error rates are computed. Finally, the auditing techniques themselves are compared, and optimum audit strategies are derived. Section 2 introduces the major auditing techniques and their variations. Section 3 computes the optimum error rates under each technique and derives the optimum timing of audits to minimize error rates. Section 4 compares the three major techniques and their various combinations, and derives optimum audit strategies. It is shown that one technique is always dominated by the others, and the optimum strategy is always a combination of two remaining techniques. The exact parameters of the optimum hybrid strategy are derived in section 5. Section 6 relaxes the three major assumptions of the analysis. The major results are shown to continue to hold under these three major extensions, indicating a large degree of robustness in the models. Section 7 summarizes the conclusions.

2. Audit strategies

There are three major approaches to database auditing and integrity enforcement. The first is periodic enforcement where integrity constraints are executed at the end of fixed periods, and the errors and violations that have occurred during the period are caught and corrected. The errors occur at random points in time during the period, and they are carried in the system until the end of the period. Clearly, during the period, any queries and reports involving the erroneous data will produce erroneous responses. The determination of the optimum period is critical to this strategy to minimize the error rates in the stored data. The optimum period may be different for each integrity constraint depending on a variety of parameters such as the execution cost of the constraint, and the error rates in the relevant incoming transactions. Clearly, high execution cost of a constraint will lead to a longer optimum period, and larger error rates in the database will have to be tolerated. Similarly, larger error rates in incoming transactions will lead to a smaller optimum period, and larger execution costs for integrity constraints will have to be tolerated.

The second approach is a transaction counting approach where the constraints are executed after every n transactions. A transaction is an update operation which inserts, deletes, or modifies one or more records in the database. As in the previous approach, the optimum selection of the number of transactions n is critical to the efficiency of this approach, and to the minimization of error rates in the stored data. A special case of the counting approach is the real-time continuous integrity enforcement where the integrity constraints are executed after each transaction ($n = 1$). A real-time continuous integrity enforcement strategy is often considered ideal in the literature since it presumably allows no errors in the system, albeit at great cost. We will show in section 4 that real-time continuous integrity enforcement is rarely the optimum strategy even when we restrict the objective to the minimization of error rates in the stored data. Very frequent execution of the integrity constraints may actually increase the error rates in the stored data due to delays it causes in executing transactions, and the consequent delays in keeping the database up-to-date. Moreover, the strategy is often infeasible due to the high execution cost of integrity constraints as discussed in section 1.

The third approach is a variation on the real-time continuous integrity enforcement. Under this approach integrity constraints are executed after each transaction, but not in their full and complete form. A simplified version of each integrity constraint is executed to test if a new error was introduced into the database by a given transaction, assuming that the database was error free before the transaction [14,18]. This strategy is called "incremental integrity enforcement". The restriction of the constraint enforcement to a specific transaction leads to considerable simplification of integrity constraints and actually makes a real-time continuous integrity enforcement a feasible strategy. However, the simplification comes at a very high price: the execution of all relevant integrity constraints after each transaction albeit in their simplified form, since simplified constraints are only valid one-transaction at a time. Such frequent execution of integrity constraints may actually increase the average error rates in the stored data due to delays it causes in executing transactions and updating the database, as discussed above.

The major tradeoff is between correctness and timeliness of data. Both may contribute to the error rates in the database: one by entering incorrect data into the system, the other by failure to update the data and hence causing it to lapse into obsolescence. Either occurrence will be counted as an error, and the objective throughout this article will be to minimize the error rates in the stored data, whether it is caused by the entry of new erroneous data or by the failure to update the existing data. Error rates will be generalized to "cost functions" in section 6, by distinguishing between errors caused by correctness and timeliness. The failure to update is commonly caused by the queuing of transactions as they wait for the processing of previous transactions, and the execution of integrity constraints. The processing delays are exacerbated by the locking of records during the execution of integrity constraints to prevent interference from transactions. The possibility of interference between transactions and integrity constraints is demonstrated by the following example.

Example 2.1. Consider a bank with two loan accounts containing a loan balance of 10 K each, and a cash reserve account containing 2 K. A constraint requires the cash reserves to be at least 10% of the total loans. This integrity constraint when used for internal control, would catch and reject violating loan transactions; and when used for auditing, would catch the violating banks and force them to raise their cash reserves.

(a) A transaction transfers 10 K from loan account1 to loan account2. The transaction has two major steps: T1 = an increase of 10 K in the loan balance of account1, T2 = a decrease of 10 K in the loan balance of account2. The integrity enforcement has three steps: I1 = retrieval of account1 balance, I2 = retrieval of the account2 balance, I3 = testing of the total. If the transaction and the integrity enforcement procedure are allowed to run concurrently, they can interfere with each other's operations and cause unpredictable results. Clearly, there is no violation of integrity by this transaction since it merely transfers the loan from one account to another, but if the steps are executed in such a sequence to interfere with each other a violation may be indicated. Consider the sequence (T1, I1, I2, T2, I3) which will indicate a violation since T1 will increase the the balance of account1 by 10 K, and I2 and I3 will receive account balances 20 K and 10 K with a total of 30 K which is clearly a violation of the constraint. On the other hand, the sequences (T1, T2, I1, I2, I3) or (T1, I1, T2, I2, I3) will both execute the integrity constraint correctly and cause no violations.

(b) Alternatively, the interaction between the constraint and the transaction may result in a failure to catch an actual violation. Assume the same example with T1 = an increase of 1 K in cash reserves, and T2 = an increase of 20 K in account1 loan balance. Clearly this transaction causes a violation of the cash reserve requirement. But, some sequences of execution will fail to catch the violation because of the interaction between the constraint and the transaction. Typically, (T1, I1, I2, T2, I3) will fail to catch the violation, but other sequences will execute correctly and indicate a violation.

Clearly, update transactions can interfere with the execution of integrity constraints if they are run concurrently. Moreover, the standard database locking procedures will not prevent this type of interference. All commercial databases lock the relevant records during an update. Such locking prevents database queries from retrieving half completed updates and returning incorrect results. However, these locking procedures are not sufficient when dealing with long-duration processes such as integrity constraint enforcement, since many transactions can update the database during integrity enforcement and auditing process, leading collectively to integrity violations. Locking the relevant records during each transaction separately and independently does not prevent interference. As an example, simply consider the example 2.1(a) above, where T1 and T2 are two separate transactions. Locking the database during each transaction separately does not solve the problem.

Consequently, during the execution of integrity constraints, all records relevant to the constraint are locked and no new updates are allowed to prevent interference between the integrity enforcement procedure and the changing state of the database. Locking during integrity enforcement (or query processing) is also known in the literature as "read-lock", as opposed to locking during updates which is called "write-lock" [4,10]. The number of records locked can vary from a few to complete files, but in general it is quite large even in incremental enforcement, since integrity constraints often involve large numbers of records especially with aggregate constraints such as averages, totals, maximums, and minimums. The database queries and report requests during the integrity enforcement can be answered, but update transactions arriving during the integrity enforcement are queued, and the system is updated after the integrity enforcement is complete. Consequently, during the time of integrity enforcement, the database becomes stale due to delays in processing the new updates. This is a clear disadvantage of very frequent integrity enforcement, and it has to be balanced against the cost of carrying errors in the system due to infrequent integrity enforcement. The objective in this article is to determine the optimum balance between these two costs.

3. Audit efficiency

The objective is to minimize the number of errors in the stored data. There are two sources of error:

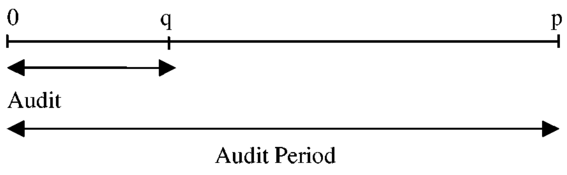
(a) entering incorrect data into the database system, and

(b) failing to update the data promptly, and hence causing it to lapse into obsolescence.

Either occurrence will be considered an error, and the objective in this article is to minimize the total number of errors in the database. The extensions to distinguish between the two types of errors, and to attach different costs to each will be considered in the last section. The audit tools used to control the error rates are integrity constraints, and the decision variable is the timing of audits. Frequent audits can reduce the errors caused by incorrect incoming transactions, but they increase the errors caused by failure to update promptly, since they lock records and block timely updates. The optimum timing of audits is a critical issue to balance these costs. This section will derive the optimum timing of audits, and the optimum error rates, under each of the three approaches to database auditing.

3.1. Periodic audit

Periodic audit approach involves audits at fixed intervals. Let p be the audit period, and q be the time it takes to execute the integrity constraint and complete the audit. The time q is a measure of the cost of each audit. It includes the execution time of the integrity constraint check, and also if a violation is detected, the time required to follow the audit trail, to locate the actual errors, and to correct them. We will assume p and q are deterministic, and q time units at the beginning of each period p are allocated to audit. Although there is a random element in q since some violations lead to time-consuming manual corrections, we will exclude these manual operations from q since they do not require locking of files.



Let r be the mean rate of arrival for atomic update transactions; and r_c and r_e be the probabilities that an atomic update transaction is correct or erroneous, respectively. We will assume that the transaction arrivals constitute a Poisson process, i.e., the time of arrival of each transaction is independent of all the previous arrivals. In other words, the interarrival times of transactions are independent [6]. The expected number of erroneous transactions arriving per unit time is $r_e r$. Since the transactions are distributed uniformly over the audit period p, from the independence assumption if Poisson arrivals, each will stay in the system for an expected period of $p/2$. All errors accumulated during the audit period i are also carried forward into the enforcement time q of period $i + 1$, leading to an additional holding time of q. Total expected time for the errors to be held in the system then is $p/2 + q$. $r_e r$ errors arriving per unit time, each held in the system for

$p/2+q$ time units, will lead to the expected number of errors in the system due to erroneous transactions

$$r_e r\left(\frac{p}{2}+q\right)=\frac{r_e r p}{2}+r_e r q. \tag{3.1}$$

Similarly, the expected number of correct transactions arriving during the audit is $r_c r q$, since the audit takes q time units. These transactions are placed in a queue, and their processing is delayed due to locking of files during the audit. The expected number of transactions waiting in the queue due to locking during the audit is $r_c r q/2$ since the transactions are distributed uniformly over q, from the independence assumption of Poisson arrivals. Each transaction waiting in the queue to be processed is interpreted as an error in the database, since it is an update that has not been processed, causing a record in the database to lapse into obsolescence. Consequently, the expected number of errors in the system during the audit period q due to queuing of transactions is also $r_c r q/2$, and the expected number of errors due to queuing in general is

$$\frac{r_c r q}{2}\times\frac{q}{p}=\frac{r_c r q^2}{2p} \tag{3.2}$$

since these errors occur only during the audit q out of the full period p. Finally, the total expected number of errors in the system is

$$\frac{r_e r p}{2}+r_e r q+\frac{r_c r q^2}{2p} \tag{3.3}$$

by adding (3.1) and (3.2).

The objective is to minimize the total expected number of errors with respect to the audit period p. By differentiation of (3.3)

$$\frac{\mathrm{d}}{\mathrm{d}p}\left(\frac{r_e r p}{2}+r_e r q+\frac{r_c r q^2}{2p}\right)=0,$$
$$\frac{r_e r}{2}-\frac{r_c r q^2}{2p^2}=0$$

implies

$$p=q\sqrt{\frac{r_c}{r_e}}. \tag{3.4}$$

The optimum number of errors in the system is obtained by substituting the optimum p value of (3.4) in equation (3.3):

$$\frac{r_e r q\sqrt{r_c/r_e}}{2}+r_e r q+\frac{r_c r q^2}{2q\sqrt{r_c/r_e}}=r_e r q+r q\sqrt{r_c r_e}. \tag{3.5}$$

Example 3.1. If 1% of the incoming transactions are in error ($r_c/r_e = 99$), and it takes 100 time units to enforce the integrity constraints ($q = 100$), then the optimum time period between audits is

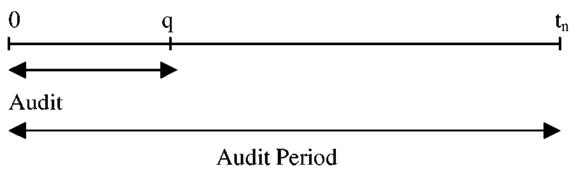
$$q\sqrt{\frac{r_c}{r_e}} = 1000 \text{ time units.}$$

Given that 1 transaction is expected to arrive per time unit ($r = 1$), the expected number of errors in the system under the optimum audit period would be

$$r_e rq + rq\sqrt{r_c r_e} = 11 \text{ errors.}$$

3.2. *Counting approach*

Counting approach does not have a fixed period, but maintains a count of incoming transactions and an audit is undertaken after a fixed number of transactions arrive. Let n be the number of transactions between two consecutive audits, and t_n be the time elapsed for n transactions to arrive. All other parameters remain the same as in section 3.1. The major difference from the previous case is the fact that the audit period t_n is a random variable corresponding to the sum of n interarrival times in a Poisson process (i.e., the arrival time of the nth Poisson arrival).



The expected number of erroneous transactions per unit time is $r_e r$. Since the transactions are expected to be distributed uniformly over t_n from the independence assumption of Poisson arrivals, each error will stay in the system for an expected period of $t_n/2$. All errors accumulated during a period i are also carried forward into the enforcement time q of period $i+1$, leading to an additional holding time q. Consequently, each error is held in the system for $t_n/2 + q$. The expected number of errors in the system due to erroneous transactions is then

$$r_e r\left(\frac{t_n}{2} + q\right) = \frac{r_e r t_n}{2} + r_e rq. \tag{3.6}$$

Similarly, the expected number of correct transactions per unit time is $r_c r$, and the expected number of correct transactions arriving during the enforcement time q is $r_c rq$. Since $r_c rq$ transactions are expected to be distributed uniformly over time q, and since they are expected to be queued due to locking during the audit, the expected number of correct transactions waiting in the queue during the audit is $r_c rq/2$. Since the transactions are queued only during the audit q out of the whole period t_n, the expected number of errors in the system due to queuing and the consequent failure to update promptly is

$$\frac{r_c r q}{2} \times \frac{q}{t_n} = \frac{r_c r q^2}{2t_n}. \tag{3.7}$$

The arrival time t_n of the nth transaction follows a γ_n distribution [5], and its probability density function is given as

$$r\mathrm{e}^{-rt_n}\frac{(rt_n)^{n-1}}{(n-1)!}.$$

The probability of an arbitrary time unit falling within t_n is

$$\frac{t_n}{n/r} r\mathrm{e}^{-rt_n}\frac{(rt_n)^{n-1}}{(n-1)!} \quad \text{since } n/r \text{ is the expected value of } t_n. \tag{3.8}$$

Finally, the expected number of errors in the system during an arbitrary time period is

$$\begin{aligned}
&\int_0^\infty \left(\frac{r_e r t_n}{2} + r_e r q + \frac{r_e r q^2}{2t_n}\right)\frac{t_n}{n/r} r\mathrm{e}^{-rt_n}\frac{(rt_n)^{n-1}}{(n-1)!}\,\mathrm{d}t_n \quad \text{by (3.6)–(3.8)}\\
&= \int_0^\infty \frac{r_e r^{n+2}}{2n(n-1)!}t_n^{n+1}\mathrm{e}^{-rt_n} + \frac{r_e r^{n+2} q}{n(n-1)!}t_n^{n}\mathrm{e}^{-rt_n} + \frac{r_e r^{n+2} q^2}{2n(n-1)!}t_n^{n-1}\mathrm{e}^{-rt_n}\,\mathrm{d}t_n\\
&= -\frac{r_e^{n+2}}{2n(n-1)!}\sum_{k=0}^{n+1}\frac{(n+1)!t_n^k\mathrm{e}^{-rt_n}}{k!r^{n-k+2}} - \frac{r_e^{n+2} q}{n(n-1)!}\sum_{k=0}^{n}\frac{n!t_n^k\mathrm{e}^{-rt_n}}{k!r^{n-k+2}}\\
&\quad - \frac{r_e r^{n+2} q^2}{2n(n-1)!}\sum_{k=0}^{n-1}\frac{(n-1)!t_n^k\mathrm{e}^{-rt_n}}{k!r^{n-k}}\Bigg|_0^\infty\\
&\left(\text{since } \int x^n\mathrm{e}^{-rx}\,\mathrm{d}x = -\sum_{k=0}^{n}\frac{n!x^k}{k!r^{n-k+1}} \text{ through repeated integration by parts [6]}\right)\\
&= \frac{r_e(n+1)}{2} + r_e r q + \frac{r_c r^2 q^2}{2n}.
\end{aligned} \tag{3.9}$$

The objective is to minimize the total expected number of errors with respect to the transaction count per audit period, n. By differentiation of (3.9):

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}n}\left(\frac{r_e(n+1)}{2} + r_e r q + \frac{r_c r^2 q^2}{2n}\right) = 0,\\
&\frac{r_e}{2} - \frac{r_c r^2 q^2}{2n^2} = 0
\end{aligned}$$

implies

$$n = rq\sqrt{\frac{r_c}{r_e}}. \tag{3.10}$$

The optimum number of errors in the system is obtained by substituting the optimum n value of (3.10) in equation (3.9):

$$\frac{r_e rq\sqrt{r_c/r_e}}{2}+\frac{r_e}{2}+r_e rq+\frac{r_c r^2 q^2}{2rq\sqrt{r_c/r_e}}=r_e rq+rq\sqrt{r_c r_e}+\frac{r_e}{2}. \tag{3.11}$$

Example 3.2. If 1% of the incoming transactions are in error ($r_c/r_e = 99$), and it takes 100 time units to enforce the integrity constraints ($q = 100$), and one transaction is expected to arrive per time period ($r = 1$), then the optimum number of transactions between audits is

$$rq\sqrt{\frac{r_c}{r_e}} = 1000.$$

The expected number of errors in the system under the optimum audit period would be

$$r_e rq + rq\sqrt{r_c r_e} + \frac{r_e}{2} = 11.005 \text{ errors}.$$

It is important to note that the number of errors under counting strategy (3.11) is always greater than the number of errors under periodic auditing strategy (3.5), and hence the counting approach can never be the optimum strategy. In particular, $n = 1$, the execution of the constraint after every transaction, which is often touted as the "ideal" solution, can never be the optimum strategy.

3.3. *Incremental audit*

Incremental audit involves the execution of integrity constraints after each transaction, but not in their full and complete form. A simplified version of the integrity constraint is executed after each transaction to test if a new error has been introduced into the database, assuming that the database was error free before the transaction. There is considerable evidence that significant simplification of integrity constraints is possible if one is only interested in detecting the new errors introduced into the system by the transaction in process [14,18]. The major disadvantage of this approach is the need to execute the integrity constraint after each transaction, albeit in its simplified form, since simplified constraints are only valid for one-transaction at a time. This approach is analogous to a pollution control system in a lake where only the incoming streams are tested for new incoming pollution as opposed to the lake itself. Testing the incoming streams turns out to be much cheaper, but it requires continuous monitoring to catch all violations. Testing the lake is much more expensive, but it does not have to be done on a continuous basis. Periodic testing is sufficient to detect all violations that have arrived up to that point in time. Similar problems arise in inventory control where inventory could be checked periodically or updated incrementally as transactions arrive. The problem here is unique because of the unique locking and queuing requirements in integrity enforcement which will eventually lead to the hybrid strategies of section 4.

Let q_i be the time required to execute an integrity constraint incrementally, i.e., only for one transaction, and only to test for new violations introduced by that transaction.

$q_{\rm i}$ is much smaller than q [18]. All other parameters are as in section 3.1. Transactions arrive at a mean rate r of a Poisson process, and each is tested for a time $q_{\rm i}$. Arriving transactions are queued as they wait to be processed and incorporated into the database. This is an M/D/1 queue with Poisson arrivals at rate r, and a single deterministic server with service time $q_{\rm i}$. The mean queue length in an M/D/1 queue is $r^2q_{\rm i}^2/(2(1-rq_{\rm i}))$ [7].

No errors are allowed into the database because of the real-time continuous monitoring. Consequently, the only errors are caused by delays in updating the system. All transactions waiting in the queue, and all transaction in process are considered errors, since they contain valid data that have not yet been incorporated into the database. The total expected number of errors then is the sum of expected number of transactions in the queue and the expected number of transactions in process. There is at most one transaction in process at a time. r transactions per unit time each taking $q_{\rm i}$ time units to be processed results in $rq_{\rm i}$ expected transactions in process at a given time. The total expected number of errors in the system at a given time then is

$$rq_{\rm i} + \frac{r^2q_{\rm i}^2}{2(1-rq_{\rm i})}. \tag{3.12}$$

Example 3.3. If the mean arrival rate of transactions is 1 and the incremental integrity enforcement time is 0.8 time units. The expected number of errors in the system is 2.4. If the transaction rate goes up to 1.2, the expected number of errors goes up to 12.46. This approach is extremely sensitive to transaction arrival rate, but completely independent of the error rates in the incoming transactions. Incremental audit appears ideal for low volume transaction systems with high error rates.

4. Hybrid strategies

The minimum possible error rate under each approach is

$$E_{\rm p} = r_{\rm e}rq + rq\sqrt{r_{\rm c}r_{\rm e}} \qquad \text{under periodic audit from (3.5),}$$

$$E_{\rm c} = r_{\rm e}rq + rq\sqrt{r_{\rm c}r_{\rm e}} + \frac{r_{\rm e}}{2} \qquad \text{under the counting approach from (3.11),}$$

$$E_{\rm i} = rq_{\rm i} + \frac{r^2q_{\rm i}^2}{2(1-rq_{\rm i})} \qquad \text{under incremental audit from (3.12).}$$

Periodic audit dominates the counting approach since $E_{\rm p}$ is always less than $E_{\rm c}$. Out of the remaining two strategies, neither one dominates the other everywhere, and hence a search for an optimum audit strategy has to consider both. Moreover, a hybrid strategy combining the two approaches is possible and has to be considered in comparison to the two pure strategies.

A hybrid audit strategy combining periodic audit with incremental audit has two sets of audits. A percentage of incoming transactions are captured and tested incrementally for new errors they may introduce. Remaining transactions are allowed to update the database without integrity testing, and consequently may introduce errors into the database. Periodically a comprehensive audit is undertaken to eliminate those errors using the periodic audit strategy. Clearly, this is a combination of two strategies. Some errors are caught at the entry point through incremental audit, and remaining errors are caught in the database through a periodic audit. The parameters of such a hybrid strategy, and the error rates under hybrid strategies have to be derived.

Let α be the percentage of incoming transactions tested incrementally at the entry point. Then $(1-\alpha)$ is the percentage of transactions that are allowed into the database without testing. Now, we have two streams of incoming transactions: α-stream is the transactions tested for integrity at the entry point, and they arrive at a rate of αr; $(1-\alpha)$ stream is the transactions directly entered into the database, and they arrive at a rate $(1-\alpha)r$.

Consider the $(1-\alpha)$ stream first. Erroneous transactions arrive at a rate $(1-\alpha)r_e r$ per time unit. Assume that a periodic audit is performed at every p periods, and the audit takes q time units as discussed in section 3.1. Since the transactions are expected to be distributed uniformly over the audit period p, from the independence assumption of Poisson arrivals, each error will stay in the system for an expected period of $p/2$. All errors accumulated during the audit period i are also carried forward into the enforcement time q of period $i+1$, leading to an additional holding time q. Total expected holding time for each error then is $p/2+q$. $(1-\alpha)$ $r_e r$ errors arriving per unit time, each held for $p/2+q$ leads to expected number of errors in the system due to erroneous transactions in the $1-\alpha$ stream:

$$(1-\alpha)r_e r\left(\frac{p}{2}+q\right). \tag{4.1}$$

Consider the correct transactions of the $1-\alpha$ stream. They constitute a Poisson process with the mean arrival rate of $(1-\alpha)r_c r$ since $r_c r$ is the mean arrival rate of the correct transactions in the original stream. If these transactions arrive during an audit, they have to be queued, and held in the queue until the audit is completed, because of the locking of files during the audit. Under the hybrid strategy, there are two types of audits, and both have to be considered in this context. Periodic audits take q time units, and repeated every p time units, in effect taking q/p per time unit. Incremental audits take q_i time units for every transaction of the α stream, or $q_i\alpha r$ per time unit. The expected number of correct transactions arriving during the periodic audit q/p is $(1-\alpha)r_c rq/p$. Each one of these transactions will be held in the queue for an expected period of $q/2$, since they are distributed uniformly over the audit time q. The resulting number of errors in the system due to delayed update during the periodic audit is

$$\frac{(1-\alpha)r_c rq}{p}\times\frac{q}{2}=\frac{(1-\alpha)r_c rq^2}{2p}. \tag{4.2}$$

The expected number of correct transactions arriving during an incremental audit $q_i\alpha r$ is $(1-\alpha)r_c r\times q_i\alpha r=(1-\alpha)\alpha r_c r^2 q_i$. Each one of these transactions will be held in the queue for an expected period of $q_i/2$, since they are distributed uniformly over the audit

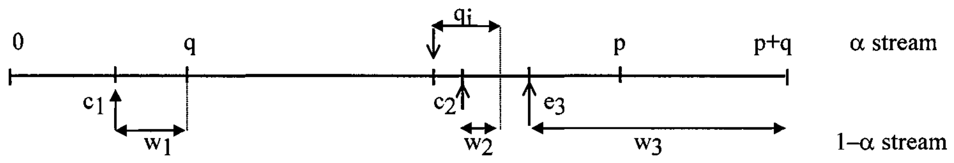


Figure 1. Processing $1-\alpha$ stream transactions, where e_n = erroneous transactions, c_n = correct transactions, w_n = waiting time for a transaction e_n or c_n.

period q_{i}. The resulting number of errors in the system due to delayed update during the incremental audits is

$$(1-\alpha)\alpha r_{\mathrm{c}} r^2 q_{\mathrm{i}} \times \frac{q_{\mathrm{i}}}{2} = \frac{(1-\alpha)\alpha r_{\mathrm{c}} r^2 q_{\mathrm{i}}^2}{2}. \tag{4.3}$$

The total expected number of errors due to correct transactions in the $1-\alpha$ stream is the sum of (4.2) and (4.3)

$$\frac{(1-\alpha)r_{\mathrm{c}} r q^2}{2p} + \frac{(1-\alpha)\alpha r_{\mathrm{c}} r^2 q_{\mathrm{i}}^2}{2}. \tag{4.4}$$

Now consider the α stream. These are the transactions that are tested for integrity at the entry point. They arrive at a mean rate of αr of a Poisson process. They are processed within a deterministic service time of q_{i}. They constitute an M/D/1 queue with Poisson arrivals at rate αr, and a single deterministic server with service time q_{i}. The expected queue length in such an M/D/1 queue is $\alpha^2 r^2 q_{\mathrm{i}}^2/(2(1-\alpha r q_{\mathrm{i}}))$ as discussed in section 3.3 [7]. All transactions waiting in the queue, and all transactions in process are considered errors, since they contain valid data that have not yet been incorporated into the database. The total expected number of errors then is the sum of expected number of transactions in the queue, and the expected number of transactions in process. There is at most one transaction in process at a time. αr transactions per unit time, each taking q_{i} time units to be processed, results in $\alpha r q_{\mathrm{i}}$ expected transactions in process at a given time. The total number of expected errors in the system due to queuing during the incremental audit of α stream transactions is:

$$\alpha r q_{\mathrm{i}} + \frac{\alpha^2 r^2 q_{\mathrm{i}}^2}{2(1-\alpha r q_{\mathrm{i}})}. \tag{4.5}$$

Finally, consider the α stream arrivals during the periodic audit. These transactions are queued and held in the queue until the periodic audit is completed because of the locking of the files during the audit. After the audit they have to be processed in turn, which leads to additional waiting time as they wait for each other. During he audit, the queue length starts at 0, and steadily increases to an expected value of $\alpha r q$, with an expected length of $\alpha r q/2$, since the transaction arrivals are expected to be uniformly distributed over q, from the independence of Poisson arrivals. During the subsequent clearing of the queue, the queue length starts at $\alpha r q$, and steadily diminishes down to 0, with an

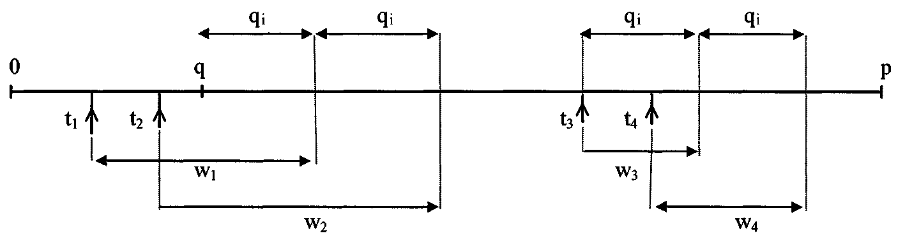


Figure 2. Processing α stream transactions, where w_n = waiting time for transaction t_n.

expected queue length of again $\alpha rq/2$. The expected queue length during the audit and subsequent clearing is then

$$\frac{\alpha rq}{2}. \tag{4.6}$$

The final step is to compute the time needed for the audit plus queue clearing, to determine how long (4.6) is the expected queue length, and to how long is the remaining time during which (4.5) is the expected queue length. The audit takes q time units. The clearing time is $\alpha rq/((1/q_i) - \alpha r)$ since the queue of length αrq is processed at a rate of $1/q_i$, while new transactions are arriving at a rate of αr, resulting in an effective processing rate of $(1/q_i) - \alpha r$. The proportion of time p taken by auditing and clearing is

$$\frac{1}{p}\left(q + \frac{\alpha rqq_i}{1-\alpha rq_i}\right) = \frac{q}{p(1-\alpha rq_i)}$$

and the remaining time is

$$1 - \frac{q}{p(1-\alpha rq_i)}.$$

The total expected queue length is the weighted average of (4.5) and (4.6):

$$\begin{aligned} &\frac{\alpha rq}{2}\,\frac{q}{p(1-\alpha rq_i)} + \left(\alpha rq_i + \frac{\alpha^2 r^2 q_i^2}{2(1-\alpha rq_i)}\right)\left(1 - \frac{q}{p(1-\alpha rq_i)}\right) \\ &\quad = \frac{\alpha rq^2}{2p(1-\alpha rq_i)} + \alpha rq_i + \frac{\alpha^2 r^2 q_i^2}{2(1-\alpha rq_i)} - \frac{\alpha rqq_i}{p(1-\alpha q_i)} - \frac{\alpha^2 r^2 qq_i^2}{2p(1-\alpha rq_i)^2}. \end{aligned} \tag{4.7}$$

The total expected number of errors under a hybrid strategy is the sum of (4.1), (4.4), (4.7):

$$\begin{aligned} E &= (1-\alpha)r_e r\left(\frac{p}{2} + q\right) + \frac{(1-\alpha)r_c rq^2}{2p} + \frac{(1-\alpha)\alpha_c r^2 q_i^2}{2} \\ &\quad + \frac{\alpha rq^2}{2p(1-\alpha rq_i)} + \alpha rq_i + \frac{\alpha^2 r^2 q_i^2}{2(1-\alpha rq_i)} - \frac{\alpha rqq_i}{p(1-\alpha q_i)} - \frac{\alpha^2 r^2 qq_i^2}{2p(1-\alpha rq_i)^2}. \end{aligned} \tag{4.8}$$

The proportion of transactions α selected for incremental audit, and the fixed audit period p are design variables for the hybrid strategy. They should be selected optimally to minimize E. The design question for a hybrid strategy then is:

$$\begin{aligned}&\underset{\alpha,p}{\text{minimize}}\ E\\&\text{subject to:}\quad 0\leqslant\alpha\leqslant 1,\\&\qquad\qquad\qquad p\geqslant 0.\end{aligned}$$

Let E_{h} be the minimum value for E, i.e., the minimum error rate under the hybrid strategy; and let α_{h} and p_{h} denote the optimum α and p values. We will characterize these values and compare them to the pure strategies in the next section, to devise an overall optimum audit strategy.

5. Optimum audit strategy

To find the overall optimum strategy, all three approaches introduced in section 3 and their hybrids have to be considered. The minimum possible error rates under each of the three pure strategies is:

$$\begin{aligned}E_{\mathrm{p}}&=r_{\mathrm{e}}rq+rq\sqrt{r_{\mathrm{c}}r_{\mathrm{e}}} &&\text{under periodic audit from (3.5),}\\E_{\mathrm{c}}&=r_{\mathrm{e}}rq+rq\sqrt{r_{\mathrm{c}}r_{\mathrm{e}}}+\frac{r_{\mathrm{e}}}{2} &&\text{under the counting approach from (3.11),}\\E_{\mathrm{i}}&=rq_{\mathrm{i}}+\frac{r^2q_{\mathrm{i}}^2}{2(1-rq_{\mathrm{i}})} &&\text{under the incremental audit from (3.12).}\end{aligned}$$

Periodic audit dominates the counting approach since E_{p} is always less than E_{c}. The search for the optimum strategy then has to consider only the periodic and incremental approaches and their hybrids. The minimum possible error rate under the hybrid approach is:

$$\begin{aligned}E_{\mathrm{h}}&=\underset{\alpha,p}{\text{minimize}}\ E\\&\text{subject to:}\quad 0\leqslant\alpha\leqslant 1,\\&\qquad\qquad\qquad p\geqslant 0,\end{aligned}$$

where

$$\begin{aligned}E={}&(1-\alpha)r_{\mathrm{e}}r\left(\frac{p}{2}+q\right)+\frac{(1-\alpha)r_{\mathrm{c}}rq^2}{2p}+\frac{(1-\alpha)\alpha_{\mathrm{c}}r^2q_{\mathrm{i}}^2}{2}\\&+\frac{\alpha rq^2}{2p(1-\alpha rq_{\mathrm{i}})}+\alpha rq_{\mathrm{i}}+\frac{\alpha^2r^2q_{\mathrm{i}}^2}{2(1-\alpha rq_{\mathrm{i}})}-\frac{\alpha rqq_{\mathrm{i}}}{p(1-\alpha q_{\mathrm{i}})}-\frac{\alpha^2r^2qq_{\mathrm{i}}^2}{2p(1-\alpha rq_{\mathrm{i}})^2}.\end{aligned}$$

The first step in developing an optimum strategy is a comparison of the two remaining pure strategies to determine the conditions under which one produces better results than the other. That will be followed by an analysis of the hybrid strategy in comparison to the pure strategies. A comparison of the two pure strategies shows that for small values of q_{i}

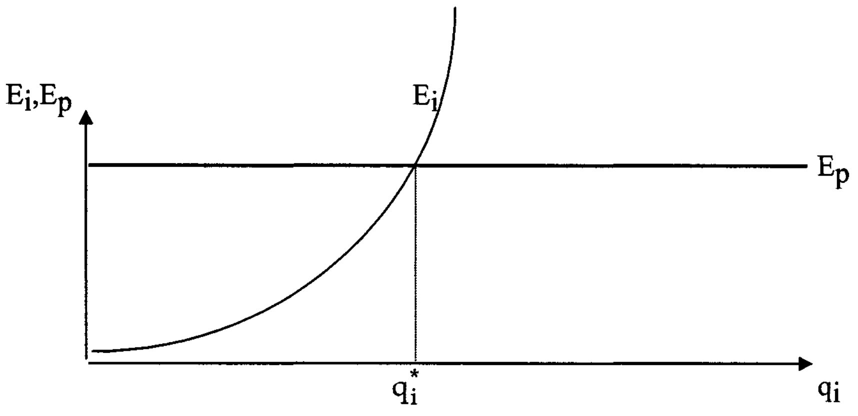


Figure 3. Error rates under periodic and incremental audits as a function of q_i.

the incremental approach gives smaller error rates, but for large values of q_i the periodic approach performs better. E_i is an increasing function of q_i, but E_p is independent of q_i. The only critical parameter is the threshold value of q_i under which E_i dominates, and over which E_p dominates. By setting $E_i = E_p$ and solving the resulting quadratic equation, we get a threshold value:

$$q_i^* = \frac{1 + rq(r_e + \sqrt{r_e r_c}) - \sqrt{1 + r^2 q^2 (r_e + \sqrt{r_e r_c})^2}}{r}.$$

Intuitively, lower q_i means cheaper incremental audit due to constraint simplification, and at $q_i < q_i^*$ incremental audit dominates the periodic audit.

The hybrid strategy is a generalization of the two pure strategies. It is a combination of the two pure strategies where α is the critical parameter that determines the mix. α percent of the transactions are processed using the incremental approach, and $(1 - \alpha)$ percent of the transactions are processed using the periodic audit approach. Clearly, when $\alpha = 0$, E_h should reduce to E_p since all transactions will be processed using the periodic approach, and when $\alpha = 1$, E_h should reduce to E_i since all transactions will be processed incrementally. Setting $\alpha = 0$ we get:

$$E = \frac{r_e r p}{2} + r_e r q + \frac{r_c r q^2}{2p}$$

and minimizing with respect to p yields $p = q\sqrt{r_c/r_e}$ and $E = r_e r q + r q\sqrt{r_c r_e} = Ep$ as discussed in section 3. Similarly, setting $\alpha = 1$ we get:

$$E = rq_i + \frac{r^2 q_i^2}{2(1 - rq_i)} + \frac{rq^2}{2p} + \frac{r^2 q^2 q_i}{2p}$$

and minimizing with respect to p yields $p = \infty$ and

$$E = rq_i + \frac{r^2 q_i^2}{2(1 - rq_i)} = E_i \quad \text{as expected.}$$

Establishing the pure strategies as special cases of the hybrid strategy reduces the problem to finding the optimum hybrid strategy. The optimum hybrid strategy is the overall optimum solution. The problem then is the nonlinear mathematical programming problem:

$$\begin{aligned} &\underset{\alpha,p}{\text{minimize}}\ E \\ &\text{subject to:}\quad 0 \leqslant \alpha \leqslant 1, \\ &\qquad\qquad\quad\ p \geqslant 0, \end{aligned}$$

where α and p are the design parameters, α indicating the mix between the two strategies, and p indicating the optimum audit period. The optimum value of p can be solved symbolically by taking the partial derivative of E with respect to p and finding the roots of the resulting quadratic equation:

$$\frac{\mathrm{d}E}{\mathrm{d}p} = \frac{(1-\alpha)r_e r}{2} - \frac{(1-\alpha)r_c r q^2}{2p^2} - \frac{\alpha r q^2}{2p^2(1-\alpha r q_i)} - \frac{\alpha^2 r^2 q q_i^2}{2p^2(1-\alpha r q_i)^2} + \frac{\alpha r q q_i}{p^2(1-\alpha r q_i)} = 0$$

yielding

$$p^* = \sqrt{\frac{\alpha q^2}{(1-\alpha)r_e(1-\alpha r q_i)} - \frac{2\alpha q_i}{(1-\alpha)r_e(1-\alpha r q_i)} - \frac{\alpha^2 r q q_i^2}{(1-\alpha)r_e(1-\alpha\, r q_i)^2} + \frac{r_c q^2}{r_e}}.$$

It is straightforward to check that the second partial derivative with respect to p is positive and this is indeed a minimizing p value. The nonlinear mathematical programming problem now reduces to a single variable minimization problem:

$$\begin{aligned} &\underset{\alpha}{\text{minimize}}\ E \\ &\text{subject to:}\quad 0 \leqslant \alpha \leqslant 1, \\ &\qquad\qquad\quad\ p = p^* \end{aligned} \tag{5.1}$$

and it is straightforward to solve using the bisection method. The critical variable once again is q_i, the cost of an incremental audit. For very small values of q_i, we expect the optimum strategy to be the pure incremental approach with $\alpha = 1$ (i.e., all transactions tested incrementally), and $p = \infty$ (i.e., no periodic audit is done). Indeed, setting $q_i = 0$ in E, all remaining terms can be eliminated by choosing $\alpha = 1$ and $p = \infty$, leading to an error rate of $E = 0$. At the other extreme, for very large values of q_i, we expect the optimum strategy to be the pure periodic audit. Indeed, setting $q_i = \infty$, all terms containing q_i can be eliminated by choosing $\alpha = 0$, and the remaining terms are

$$E = \frac{r_e r p}{2} + r_e r q + \frac{r_c r q^2}{2p},$$

which is indeed equivalent to the periodic audit error rate of equation (3.3). The optimum α values as a function of q_i are shown in figure 4.

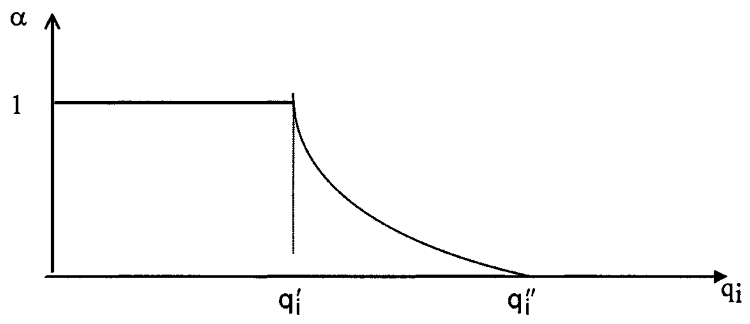


Figure 4. The optimum α values indicating the correct mix of the two pure strategies, as a function of q_i (the cost of an incremental audit).

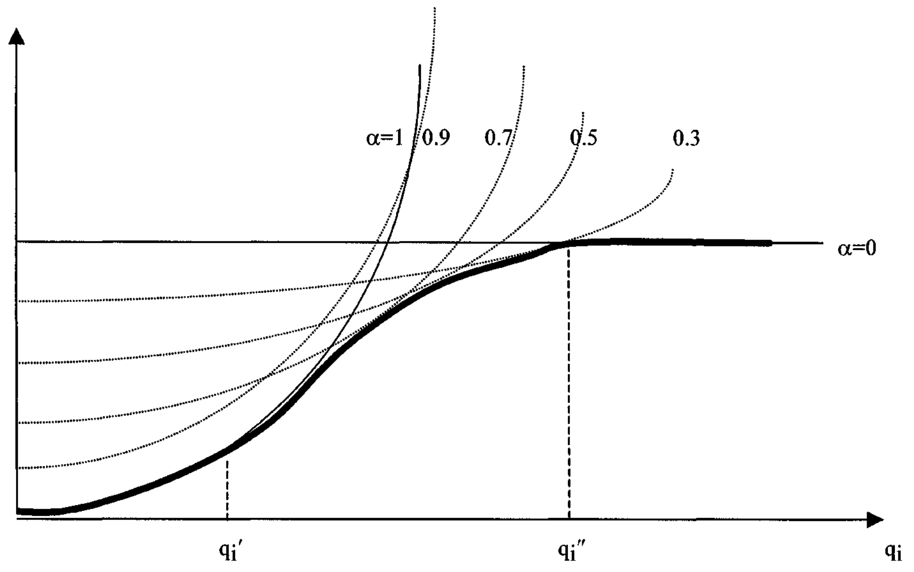


Figure 5. Error rates under a hybrid strategy as a function of q_i (the cost of incremental audits), and parameterized with respect to α (the mixing parameter). The envelope curve is the overall optimum strategy E_h.

The critical values are q_i' and q_i'', where below q_i' the pure incremental audit is optimum, above q_i'' the pure periodic audit is optimum, and between q_i' and q_i'' a hybrid audit strategy is needed. The α value determining the optimum hybrid strategy is obtained by solving the nonlinear mathematical programming problem (5.1).

The minimum possible error rates E_h under the optimum strategy are computed similarly from the solution of nonlinear programming problem (5.1). Again, for very small values of q_i, since the optimum strategy is pure incremental, the optimum error rates are equal to E_i. For very large values of q_i, since the optimum strategy is periodic, the error rates are equal to E_p. The critical threshold values are again q_i' and q_i''. Between q_i' and q_i''a hybrid strategy is optimal defined by an α value $0 < \alpha < 1$. Figure 5 shows E (error rate under hybrid strategies) as a function of q_i and parameterized with respect

Table 1

q_i	α	E_i	E_p	E_h
0.1	1	0.12	10.95	0.12
0.2	1	0.24	10.95	0.24
0.5	1	0.76	10.95	0.76
0.8	1	2.43	10.95	2.43
0.9	1	4.96	10.95	4.96
1.0	0.42	∞	10.95	9.88
1.2	0.28	∞	10.95	10.27
1.3	0.24	∞	10.95	10.39
1.4	0.21	∞	10.95	10.49
1.5	0.18	∞	10.95	10.58
1.6	0.16	∞	10.95	10.65
1.7	0.14	∞	10.95	10.70
1.8	0.12	∞	10.95	10.75
2.0	0.09	∞	10.95	10.83
2.5	0.02	∞	10.95	10.94
3.0	0	∞	10.95	10.95
4.0	0	∞	10.95	10.95

to α. It is easy to see that for small q_i, $\alpha = 1$ is the optimum; for large q_i, $\alpha = 0$ is the optimum; and for q_i between q_i' and q_i'', the envelope curve of all E curves produces the optimum E_h.

Example 5.1. Given a transaction arrival rate $r = 1$, error rate $r_e = 0.01$ and $r_c = 0.99$, periodic audit taking 100 time units $q = 100$, the following table shows the optimum error rate E_h and the optimum mix α as a function of q_i. The error rates under the pure strategies E_i and E_p are also shown for comparison in table 1, and in figure 6, where $q_i' \approx 1.0$ and $q_i'' \approx 3.0$.

6. Extensions

Four major assumptions characterize the models of database auditing in this article. These assumptions will be relaxed in this section, and the models will be extended to accommodate the relaxed conditions. The general conclusions will continue to hold in all three models, indicating considerable robustness in the models.

6.1. *Distinction between types of errors*

Throughout this article, the error counts have been used as a measure of data quality. No distinction was made between errors resulting from incorrect arrivals, and errors resulting from delayed updates. Relaxing this assumption requires distinguishing between these two types of errors, and attaching different costs to them. The resulting model aims at minimizing the total cost of errors, rather than simply the error count where all errors were assumed to be equally important. This extension is especially use-

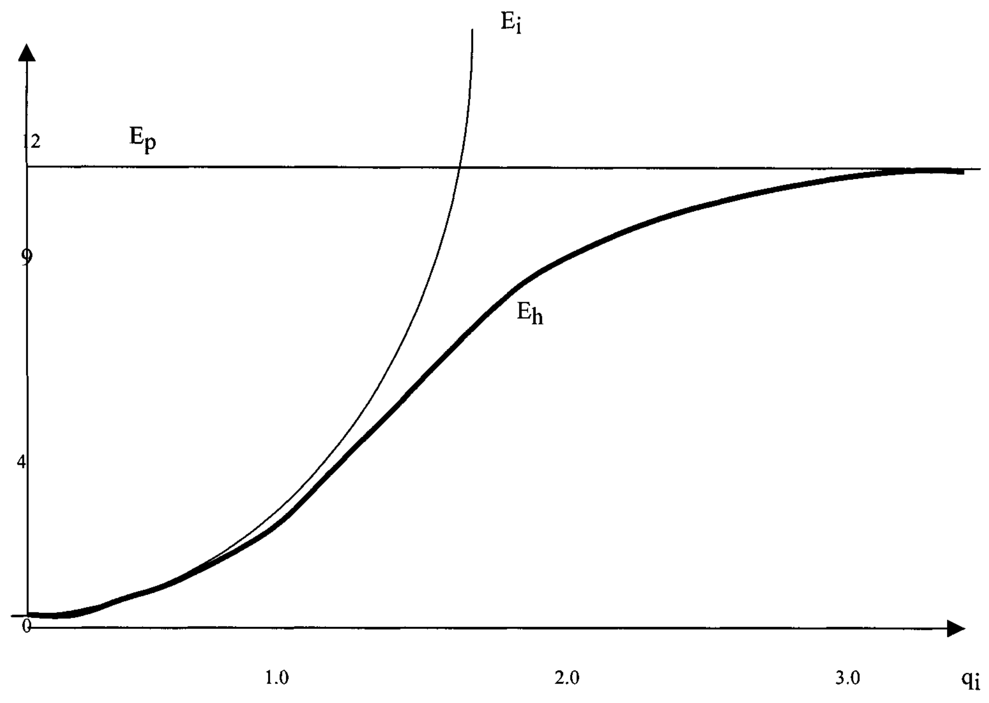


Figure 6. Error rates as a function of q_i for the optimum strategy in example 5.1.

ful when the data values change incrementally (as in salaries, ages, or some inventories), and delayed updates do not cause as serious a harm since the old values are not too far from the new correct values. The relative importance of each type of error is captured in the cost parameters assigned to each. The estimation of actual cost parameters for each situation is beyond the scope of his article, but given the unit error holding costs c_c and c_e for delayed updates (of correct data) and incorrect arrivals (erroneous data), respectively, the resulting extensions are straightforward. In periodic auditing, the errors resulting from incorrect arrivals (equation (3.1)) have the cost multiplier c_e, and the errors resulting from delayed updates (equation (3.2)) have the cost multiplier c_c. The resulting derivations are similar except for replacing r_e with $c_e r_e$, and replacing r_c with $c_c r_c$ throughout. Example 3.1 can also be extended. If $c_c = \$10$ and $c_e = \$1$, then the optimum audit period would be 316 time units, and the optimum cost would be \$20 per time unit.

The counting strategy continues to be dominated. The incremental audit has only delayed update errors, and hence the only cost multiplier is c_c in equation (3.12). Example 3.3 with $c_c = \$1$ would produce the optimum error costs of \$2.4 per time unit and \$12.48 per time unit for the two cases considered.

The hybrid approach has three components. Equation (4.5) is a count of incorrect arrivals and has the cost multiplier c_e. The other components are counts of various delayed updates and hence have the cost multiplier c_c. The resulting derivations lead to similar equations with the total error count E replaced by the total cost $c_c E$, and the r_e

replaced by $c_e r_e / c_c$ throughout. The basic results continue to hold. The optimum values of α and p are slightly higher given that $c_e > c_c$, since the higher cost associated with incorrect arrivals leads to more incremental auditing to catch more incorrect arrivals at the entry point, longer queues to take advantage of the smaller cost of delayed updates, and less frequent periodic audits for the fewer remaining errors.

6.2. Distinction among classes of data

The strategy of minimizing the error counts assumes that all errors are equally important. This assumption may need to be relaxed when some data are more critical to users than others, and consequently when the errors in some data are more costly than others. The distinction among data may be due to the frequency of use, or the type of applications that use them. Relaxing this assumption leads to slight changes in the models, but the basic results continue to hold. The database is partitioned into n classes where each class is associated with a different unit cost per error. Let $c_1, \ldots, c_n$ be the costs of holding an error for one time unit in classes $1, \ldots, n$, respectively. Let the transactions affecting each class arrive at rates $r_1, \ldots, r_n$. Given E as the error count, $r_1/r, \ldots, r_n/r$ are the proportion of errors related to each class. Multiplying with the appropriate costs, $(c_1 r_1/r + \cdots + c_n r_n/r)E$ is the total cost to be minimized. The constant multiplier changes the optimum value, but the optimum strategy (audit period p, mixing parameter α, etc.) remains the same for each constraint. The distinction among data classes becomes important if one needs to compare constraints among themselves, and assign priorities. This article treats constraints individually and determines optimum enforcement policies for each. If there are resource constraints, and not all constraints can be enforced optimally, then assigning priorities to constraints becomes critical, and since each constraint deals with a different class of data, the cost attached to each data class becomes also critical. The constraint-at-a-time analysis of this article on the other hand is not affected by classification of data.

6.3. Poisson arrivals assumption

The major statistical assumption underlying the mathematical models is the Poisson arrivals assumption for the transactions where the interarrival times are independent of each other. However, transactions do not always arrive individually. Some transactions are generated by programs or periodic manual processes, and arrive in clusters, which violates the independent arrivals assumption of a Poisson Process. Such arrivals require a Compound Poisson model, which involves Poisson arrivals where each arrival is a cluster containing a random number of transactions. Relaxing Poisson arrivals assumption leads to slight changes in the models, but the general results continue to hold. Assume that each Poisson arrival has a normally distributed number of transactions with mean μ and variance σ^2. This assumption holds for common internal transactions such as payroll and billing where the number of transactions in a cluster vary slightly around a relatively stable mean. The expected number of transactions per period is $r\mu$ and its variance is $r^2\sigma^2$. The expected number of errors under periodic audit can be computed by replacing r with $r\mu$ in all formulas. For $\mu = 1$, the numbers in example 3.1 remain unchanged irrespective of the variance. Counting approach continues to be dominated. The incremental audit is significantly different. Since each Poisson arrival contains a random number of transactions, the queue becomes a bulk arrival queue with each arrival containing a normally distributed number of transactions with mean μ and variance σ^2, and a constant service time per transaction of q_i. The expected number of transactions in such a bulk queue (queue length + number in service) is [21]

$$\mu r q_i + \frac{\mu r^2 q_i^2}{2(1 - \mu r q_i)} \left(\frac{\sigma^2}{\mu} + 1 \right).$$

For $\mu = 1$ and $\sigma^2 = 0$, this formula reduces to equation (3.12), and for $\mu = 1$ and $\sigma^2 = 1$ the number of errors in example 3.3 goes up from 2.4 to 4.0 which is the effect of variance on the error rate.

For the hybrid strategy, using the modified formulas given above for each component, equation (4.8) is modified by replacing all occurences of r by μr, and by adding the multiplier $(\sigma^2/\mu) + 1$ to the terms 6 and 8. The optimum value of p is similar except for replacing all r with μr, and a multiplier $(\sigma^2/\mu) + 1$ for the 3rd term. For $\mu = 1$ and $\sigma^2 = 0$, the optimum audit period p reduces to the previous case with Poisson arrivals, and for $\mu = 1$ and $\sigma^2 > 0$, it is slightly larger than the previous Poisson case since $(\sigma^2/\mu) + 1 > 1$.

Similarly, the mixing parameter α values are also slightly larger due to larger values of p. Intuitively, bulk arrivals with a nonzero variance lead to slightly higher rate of incremental audit since incremental audit becomes more efficient than periodic audit, by adjusting better to the size of each arriving cluster. Correspondingly, the periodic audit becomes less frequent to accommodate remaining fewer errors.

6.4. Deterministic audit time assumption

The periodic audit time q, and the incremental audit time q_i have both been assumed to be deterministic.

This is a reasonable assumption since the nondeterministic components of an audit such as following an audit trail may not require locking. Nevertheless, the assumption can be relaxed to accommodate stochastic audit time. Let the periodic audit time be normally distributed with mean q and standard deviation s. By substituting the new random variable for the deterministic variable q in sections 3.1 and 3.2, the results are not affected and the error rates remain the same, since expected value of the error rates depend only on the expected value of the audit time, and they are independent of the standard deviation. The incremental analysis on the other hand is affected by the stochasticity of audit time. Let the incremental audit time be normally distributed with mean q_i and standard deviation s_i. The queue discipline changes to M/G/1 since the service time is

Table 2

q_i	s_i	α	E_i	E_p	E_h
0.8	0.1	1	2.44	10.95	2.44
0.8	0.5	1	3.03	10.95	3.03
0.9	0.1	1	5.00	10.95	5.00
0.9	0.5	1	6.20	10.95	6.20
1.0	0.1	0.43	∞	10.95	9.89
1.0	0.5	0.42	∞	10.95	9.92
1.2	0.1	0.29	∞	10.95	10.27
1.2	0.5	0.28	∞	10.95	10.28
1.3	0.1	0.25	∞	10.95	10.40
1.3	0.5	0.24	∞	10.95	10.41
2.5	0.1	0.02	∞	10.95	10.94
2.5	0.5	0.02	∞	10.95	10.94
3.0	0.1	0	∞	10.95	10.95
3.0	0.5	0	∞	10.95	10.95

$q_i' \approx 1.0$ and $q_i'' \approx 3.0$.

now stochastic. The mean queue length in an M/G/1 queue with arrival rate of r is and departure distribution with mean q_i and standard deviation s_i is [7]

$$\frac{r^2 q_i^2(1 + s_i^2/q_i^2)}{2(1 - rq_i)},$$

which changes equation (3.1), the expected number of errors in the system under incremental audit to:

$$rq_i + \frac{r^2 q_i^2(1 + s_i^2/q_i^2)}{2(1 - rq_i)}.$$

Example 3.3 produces slightly different values under these conditions. Given the same parameters, and the standard deviation $s_i = 0.8$, the expected number of errors in the system is 4.0, and if the standard deviation is 0.9, the error rate goes up to 4.16.

Similarly, the hybrid audit strategies have to be adjusted for stochastic audit time. By using the M/G/1 queue discipline, and the resulting error rates, equation (4.8) is slightly modified by multiplying the sixth and the eighth terms by $1 + s_i^2/q_i^2$. All conclusions continue to be valid, but the values of the decision variables of example 5.1 are slightly different as shown in table 2.

7. Conclusions

The three major strategies of database auditing has been introduced and compared in terms of effectiveness in detecting and eliminating errors. Optimum timing of audits under each strategy has been computed. One strategy was shown to be completely dominated by the others. The remaining two strategies, and their hybrids have been studied in detail. Hybrid strategies were shown to outperform the pure strategies under certain conditions. Overall optimum strategies have been devised, their parameters computed, and the necessary and sufficient condition, for their optimality have been derived. Finally, the four major assumptions of the models have been relaxed. The basic results continue to hold under the relaxed conditions, indicating considerable robustness in the models.

There are a number of parameters that need to estimated to apply this model correctly. Error rates and arrival rates of incoming transactions, audit times for periodic and incremental audits are all critical to this analysis. However the system itself is designed to detect and eliminate errors, and to perform audits, and hence those values are readily available to the model during the operational life of the system. Consequently, as the system operates, the original estimates can be checked against the actual observations, and the estimates can be updated dynamically to account for discrepancies. Such an approach produces a self improving system which does not require precise estimates of parameters before deployment, but allows for a continuously improving system which may start with very "rough" estimates of parameters.

References

[1] T. Amer, A.D. Bailey and P. De, A review of the computer information systems research related to accounting and auditing, Journal of Information Systems 1 (1987) 3–28.

[2] D.P. Ballou and H.L. Pazer, Designing information systems to optimize the accuracy-timeliness trade-off, Information Systems Research 6 (1995) 56–82.

[3] D.P. Ballou and H.L. Pazer, Modeling data and process quality in multi-input multi-output information systems, Management Science 31(2) (1985) 150–162.

[4] P.A. Bernstein, V. Hadzilacos and N. Goodman, *Concurrency Control and Recovery in Database Systems* (Addison-Wesley, 1986).

[5] M.J. Cerullo, General controls in computer systems, Computers and Security 4 (1985) 33–45.

[6] E. Cinlar, *Introduction to Stochastic Processes* (Prentice-Hall, 1975).

[7] R.B. Cooper, *Introduction to Queuing Theory* (Ceepress Books, 1990).

[8] C.J. Date, *An Introduction to Database Systems* (Addison-Wesley, 1995).

[9] G.B. Davis and R. Weber, The impact of advanced computer systems on controls and audit procedures: A theory and empirical test, Auditing: A Journal of Practice and Theory 46 (1986) 1–28.

[10] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques* (Morgan Kaufmann, 1993).

[11] S.M. Groomer and U.S. Murthy, Continuous auditing of database applications: An embedded audit module approach, Journal of Information Systems 3 (1989) 53–69.

[12] H.S. Koch, Auditing on-line systems: An evaluation of parallel versus continuous and intermittent simulation, Computers and Security 3 (1984) 9–19.

[13] K.C. Laudon, Data quality and due process in large interorganizational record systems, Communications of ACM 29(1) (1986) 4–11.

[14] W. McCune and L. Henschen, Maintaining state constraints in relational databases, Journal of ACM 36(1) (1989) 266–291.

[15] R.C. Morey, Estimating and improving the quality of information in MIS, Communications of ACM 25(5) (1982) 337–342.

[16] A. Motro, Integrity = validity + completeness, ACM Transactions on Database Systems 14(4) (1989) 480–502.

[17] L.V. Orman, A model management approach to business process reengineering, Journal of MIS 15(1) (1998) 187–212.

[18] L.V. Orman, Differential relational calculus for integrity maintenance, IEEE Transactions on Knowledge and Data Engineering 10(2) (1998) 328–341.
[19] D.G. Paradise and W.L. Fuerst, An MIS data quality methodology based on optimal error detection, Journal of Information Systems 5(1) (1991) 48–66.
[20] T.L. Redman, *Data Quality: Management and Technology* (Bantam Books, 1992).
[21] T.L. Saaty, *Elements of Queuing Theory* (McGraw-Hill, 1961).
[22] A. Segev and J.L. Zhao, Rule management in expert database systems, Management Science 40(6) (1994) 685–707.
[23] R.Y. Wang, V.C. Storey and C.P. Firth, A framework for the analysis of data quality research, IEEE Transactions on Knowledge and Data Engineering 7(4) (1995) 623–639.
[24] Y. Wand and R. Weber, A model of control and audit procedure change in evolving data processing systems, Journal of Accounting Research 11(3) (1989) 273–295.
[25] R. Weber, *EDP Auditing: Conceptual Foundations and Practices* (Prentice-Hall, 1988).