A Reorganization Model Based on the Database Entropy Concept

K. T. Fung*
American Bell Inc., 307 Middletown/Lincroft Rd., Lincroft, New Jersey 07738, USA

The database entropy framework previously developed to characterize the learning process of a database is employed to define a reorganization model that can be used to permit a database system to adapt itself to the continuously changing environment. Such a model can readily be incorporated into any database management system to provide it with a certain desirable amount of intelligence.

INTRODUCTION

In a recent paper,[1] the entropy concept[2] has been used to develop a framework that can be used to characterize the learning process of a database. This framework, using the notion of database entropy measures, permits us to describe dynamically and quantitatively some of the most important operational characteristics of the database in concern during its active lifetime. These characteristics include, for example, the total amount of information stored in the database, the way the directory or the records are organized, the average time required to service a typical retrieve or update operation, and the total cost for maintaining the database. All of these characteristics change continuously as queries are issued to access the database by the users. The framework that has been developed is very general and can be used on a database of a varying degree of complexity ranging from a simple database to a distributed heterogeneous database. Such a framework not only is useful in the design and analysis of databases, but also can provide a basis for a unified approach to tackling a variety of related database problems. In this paper, we shall effectively make use of this framework to define a reorganization model which can be used to permit a database system to adapt itself to the continuously changing user requirements, and thus represents a significant contribution towards the design and implementation of intelligent databases, i.e. those databases that are capable of self-adaption.

THE FRAMEWORK

In this section, we briefly describe the database entropy framework that can be used to characterize the learning process of a database. For a more detailed description on this framework, see Refs 1, 9 and 10.

The various queries used to perform operations such as retrieve, update, delete and insert, and issued by the users to communicate with the database, contribute towards the behaviour of the database which in turn controls the immediate future behaviour of the database in response to forthcoming queries. A specific collection of all relevant factors describing the present behaviour of a database is labelled a possible state of knowledge of the database. The learning process of the database can be expressed in terms of these possible states of knowledge into which the database can go during its operation. The dynamic nature of the database system makes the entropy approach very well suited for taking care of the normally large collection of possible states of knowledge in a meaningful and yet manageable fashion.

Limiting ourselves to testable factors[3] that are of major concern to the users as well as the designers, we define a set of possible states of knowledge for the query system $E^Q = \{E_1^Q, \ldots, E_m^Q\}$ with their associated probabilities of occurrence $\{P_1^Q, \ldots, P_m^Q\}$. The entropy of the query system $H(Q)$, given by

$$H(Q) = -\sum_j P_j^Q \log P_j^Q \qquad (1)$$

is known as the state of knowledge of the query system at the time of consideration. Changes in the user conditions are reflected by changes in the E^Q and/or their associated probabilities of occurrence, and consequently in $H(Q)$.

We similarly define a set of possible states of knowledge for the database $E^D = \{E_1^D, \ldots, E_n^D\}$ with their associated probabilities of occurrence $\{P_1^D, \ldots, P_n^D\}$. The entropy of the database $H(D)$, given by

$$H(D) = -\sum_j P_j^D \log P_j^D \qquad (2)$$

is known as the state of knowledge of the database. $H(D)$ is a description of the information stored in the database, and changes accordingly with time as the database information content or structure changes.

The two sets E^Q and E^D form the Cartesian product $E^{QD} = E^Q \times E^D$. Each element of E^{QD} represents a possible state of knowledge of the combination of query and database systems, a collection of factors in both the query and the database systems that determine how the database will respond to the next incoming query given the specific possible states of knowledge of the query system and the database. The entropy $H(Q \times D)$, given by

$$H(Q \times D) = -\sum_k P_k^{QD} \log P_k^{QD} \qquad (3)$$

is known as the state of knowledge of the query-database system.

*The work reported here was done while the author was with the School of Computer Science, University of Windsor, Windsor, Ontario, Canada.

CCC–0010–4620/84/0027–0067 $02.50

The three entropies defined in (1), (2) and (3) are related by the following well-known relationship;[4]

$$H(Q \times D) \leq H(Q) + H(D) \quad (4)$$

The two conditional entropies $H_Q(D)$ and $H_D(Q)$ given by

$$H_Q(D) = H(Q \times D) - H(Q) \quad (5)$$

$$H_D(Q) = H(Q \times D) - H(D) \quad (6)$$

measure the degree of dependence of the database and query systems. In very general terms, we can say that $H_Q(D)$ measures the knowledge that the database lacks about the query system and vanishes if the database completely satisfies the requirements of the query system, and that $H_D(Q)$ on the other hand measures the knowledge that the database has but is not required by the query system, thus representing irrelevant knowledge contained in the database, and vanishes if the database contains no such irrelevant knowledge.

It is important to note that, depending on how E^Q and E^D are defined, the knowledge so measured can actually be negative knowledge representing undesirable characterizations (e.g. cost) of the database system. The interpretation of the associated entropies should be made accordingly.

The conditional entropies in (5) and (6) therefore can be used to characterize the learning process of the database system during its operational lifetime with respect to the factors that have been chosen to form the sets E^Q and E^D. These entropies convey quantitatively how the database system has been operating and how it is expected to operate in the immediate future.

In case we need to consider the combined effect of both characteristic measures as expressed by $H_Q(D)$ and $H_D(Q)$, it is convenient to introduce the concept of excess entropy (or mutual entropy).[4] By definition, the excess entropy of the two sets E^Q and E^D, denoted by C, is given by

$$C = H(Q) + H(D) - H(Q \times D) \quad (7)$$

$$= H(Q) - H_D(Q) \quad (8)$$

$$= H(D) - H_Q(D) \quad (9)$$

The excess and conditional entropies are related by the following equation:

$$C = H(Q \times D) - (H_D(Q) + H_Q(D)) \quad (10)$$

We can look at C as being a quantity which serves to characterize the combined state of knowledge of the query–database system. $C = 0$ when the database is constructed independent of the query system, and $C = H(Q) = H(D)$ when $H_Q(D) = H_D(Q) = 0$, i.e. when the database contains no irrelevant knowledge and completely satisfies the requirements of the query system.

THE REORGANIZATION MODEL

In many database systems, it is desirable, or even essential, in some cases to perform reorganization at fixed or variable time intervals on all or part of the information or structural contents so as to overcome the continuous deterioration in the overall performance caused by update operations or various other contributing factors. Included in this class of problems is the establishment of checkpoints to provide a basis for rollback recovery operations in case of system failures. Studies on the topic of database reorganization have been done by a number of researchers.[5–8] Based on the database entropy framework briefly described in the last section, we develop in this section a new model which can be used dynamically to determine when reorganization operations are necessary. In this model, it is not necessary to assume *apriori* specific forms of probability distribution for the cost, file growth or failure functions and other relevant parameters as required in all the previous work.

Assuming that the sets of possible states of knowledge E^Q and E^D have been appropriately constructed to reflect the performance criteria, so that the excess entropy C indeed is now a meaningful measure of the operational level of the query database system, we can use equation (8) to define a simple yet effective model that can be used by the database management system to perform reorganization in a self-adaptive environment. This model is shown in Fig. 1.

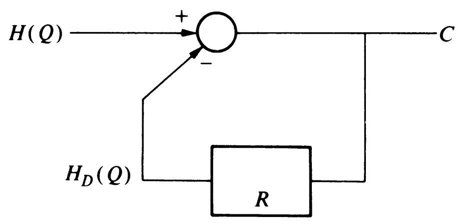

Figure 1. Reorganization model.

In the model, $H(Q)$ acts as the input information and C acts as the output information that is also feedback to block R. Block R refers to the actual reorganization procedure adopted in the system. The operation of the model is straightforward. At a certain time during the operation of the database, C assumes a value which is considered to be acceptable in terms of the overall performance criteria. As the user requirements change, so do $H(Q)$ and $H_D(Q)$. C in turn changes. If C, after the change, still remains within the acceptable range, no reorganization operations need be performed by procedure R. If, on the other hand, the new value of C now falls outside the acceptable range, then procedure R should perform the needed reorganization operations (repetitions of these operations may be necessary) to increase or decrease $H_D(Q)$ so as to bring C back to the acceptable range.

In the model, the input to procedure R is C and the output is $H_D(Q)$. For a particular procedure R, a plot of $H_D(Q)$ against C, usually in the form of a discontinuous curve, can be used conveniently to describe quantitatively the operation of the procedure. It is easy to see that a useful measure of the relative merit of a reorganization procedure R is the ratio

$$\left.\frac{\Delta H_D(Q)}{\Delta C}\right| C = c \quad (11)$$

where $\Delta H_D(Q)$ is the increase (decrease) in $H_D(Q)$ for an increase (decrease) of ΔC in C at $C = c$. We indicate this rate of change by $\| R \|_c$. It follows that, if the acceptable level of C is in the neighborhood of c, then the larger

$\| R \|_c$ is, the more effective procedure R would be—this usually translates to a smaller number of repetitions of the reorganization operations required. It is also obvious that a first requirement for a procedure R to be at all effective is that $\| R \|_c > 0$.

AN EXAMPLE

To demonstrate the application of the reorganization method developed in the previous section, we look at a very simple example. Although this example is rather hypothetical, it serves well to illustrate the effectiveness and more importantly, the distinctiveness of the entropy approach.

Let a simple database be made up of 4 types of records, numbered 1 to 4. We assume that records of 3 types are stored in a main database whereas records of the remaining type are stored in a secondary database. The secondary database is used to store records of that type that are referenced less frequently and are for this reason displaced from the main database. Naturally, we assume that this secondary database is implemented on devices that have much inferior operational characteristics to those of the devices on which the main database is implemented. As the query requirements change with time, the records that are stored in the secondary database may have to be brought up to the main database and vice versa so as to ensure a certain acceptable level of performance. One level of performance might be the hit ratio, where hit ratio (to be denoted by h) is the percentage of queries which need not access secondary storage for completion.

Let $p_t(i)$ be the estimated probability of occurrence of a reference to record type i in a query at time t. Then

$$h = 1 - p_t(i_t)$$

if records of type i_t are currently stored in the secondary database.

Let the $p_t(i)$s for the record types for 20 consecutive times 0, 1, . . ., 19 be as given in Table 1. The $p_t(i)$s have been chosen in such a way that the effects of the entropy approach can be captured distinctly. It can be seen that during the period, $p_t(1)$ remains constant, whereas $p_t(2)$ changes very slowly between the two values of 2 and 3 and $p_t(3)$ and $p_t(4)$ change rapidly between the two values of 1 and 3.

Algorithm A

Let us now describe a swapping algorithm based on the entropy approach. We call this Algorithm A. To facilitate its understanding, we shall present it directly through its application to the example.

We assume that at time 0, records of type 4 are stored in the secondary database (the record type stored in the secondary database at each time is underlined in Table 1).

Let us form the set $\{E_1^Q, \ldots, E_4^Q\}$ where E_j^Q is the possible state of knowledge that a record, which has been referenced during the period $t-2$ (in general $t-T$) to t, is of type j. It is straightforward to compute, for each E_j^Q, the probability of occurrence p_j^Q from the given $p_t(i)$s (see Table 2) and $H(Q)$ (see Table 3, column 1).

Let us now construct the set $\{E_{D1}^Q, \ldots, E_{D4}^Q\}$, where E_{Dj}^Q is the event that a record referenced during the period $t-2$ (in general $t-T$) to t and residing in the main memory (and not the secondary database) is of type j. The probability p_{Dj}^Q obviously depends on which record type is stored in the secondary database at that time and can also readily be computed from the $p_t(i)$s. Once the p_{Dj}^Qs are known, $H_D(Q)$ can be computed.

The excess entropy C at any time is then given by

$$C = H(Q) - H_D(Q)$$

At time 0, records of type 4 are in the secondary database. The p_{Dj}^Qs are computed accordingly and are shown as the first row entries in Table 2. The corresponding values for $H_D(Q)$ and C are given as first row entries in Table 3 (columns 2 and 3, respectively).

We now assume that if the value of C is higher than 0.130, (in general S) then swapping is needed.

Table 1. $p_t(i)$s for 20 consecutive times

t	$p_t(1)$	$p_t(2)$	$p_t(3)$	$p_t(4)$
0	0.4	0.3	0.3	$\underline{0.1}$
1	0.4	0.3	0.1	$\underline{0.3}$
2	0.4	0.2	0.3	$\underline{0.1}$
3	0.4	0.2	0.1	$\underline{0.3}$
4	0.4	$\underline{0.2}$	0.3	0.1
5	0.4	$\underline{0.2}$	0.1	0.3
6	0.4	$\underline{0.2}$	0.3	0.1
7	0.4	$\underline{0.2}$	0.1	0.3
8	0.4	$\underline{0.2}$	0.3	0.1
9	0.4	$\underline{0.2}$	0.1	0.3
10	0.4	$\underline{0.2}$	0.3	0.1
11	0.4	$\underline{0.2}$	0.1	0.3
12	0.4	$\underline{0.3}$	0.3	0.1
13	0.4	0.3	$\underline{0.1}$	0.3
14	0.4	0.2	$\underline{0.3}$	0.1
15	0.4	0.2	$\underline{0.1}$	0.3
16	0.4	0.2	$\underline{0.3}$	0.1
17	0.4	$\underline{0.2}$	0.1	0.3
18	0.4	$\underline{0.2}$	0.3	0.1
19	0.4	$\underline{0.2}$	0.1	0.3

Table 2. The probabilities of occurrence p_j^Qs and p_{Dj}^Qs

t	p_1^Q	p_2^Q	p_3^Q	p_4^Q	p_{D1}^Q	p_{D2}^Q	p_{D3}^Q	p_{D4}^Q
0	0.364	0.272	0.272	0.091	0.400	0.300	0.300	0.000
1	0.364	0.272	0.182	0.182	0.445	0.333	0.222	0.000
2	0.375	0.250	0.219	0.156	0.444	0.296	0.259	0.000
3	0.387	0.226	0.161	0.226	0.500	0.292	0.208	0.000
4	0.400	0.200	0.233	0.167	0.500	0.000	0.291	0.209
5	0.400	0.200	0.167	0.233	0.500	0.000	0.209	0.291
6	0.400	0.200	0.233	0.167	0.500	0.000	0.291	0.209
7	0.400	0.200	0.167	0.233	0.500	0.000	0.209	0.291
8	0.400	0.200	0.233	0.167	0.500	0.000	0.291	0.209
9	0.400	0.200	0.167	0.233	0.500	0.000	0.209	0.291
10	0.400	0.200	0.233	0.167	0.500	0.000	0.291	0.209
11	0.400	0.200	0.167	0.233	0.500	0.000	0.209	0.291
12	0.387	0.226	0.226	0.161	0.500	0.000	0.292	0.208
13	0.375	0.250	0.156	0.219	0.444	0.296	0.000	0.259
14	0.375	0.250	0.219	0.156	0.480	0.320	0.000	0.200
15	0.387	0.226	0.161	0.226	0.461	0.269	0.000	0.269
16	0.400	0.200	0.233	0.167	0.522	0.260	0.000	0.217
17	0.400	0.200	0.167	0.233	0.500	0.000	0.209	0.291
18	0.400	0.200	0.233	0.167	0.500	0.000	0.291	0.209
19	0.400	0.200	0.167	0.233	0.500	0.000	0.209	0.291

Table 3. Entropies $H(Q)$, $H_D(Q)$ and C

t	$H(Q)$	$H_D(Q)$	C
0	0.563	0.472	0.091
1	0.581	0.460	0.121
2	0.580	0.465	0.115
3	0.580	0.448	0.132
4	0.576	0.449	0.127
5	0.576	0.449	0.127
6	0.576	0.449	0.127
7	0.576	0.449	0.127
8	0.576	0.449	0.127
9	0.576	0.449	0.127
10	0.576	0.449	0.127
11	0.576	0.449	0.127
12	0.580	0.449	0.131
13	0.580	0.464	0.116
14	0.580	0.451	0.129
15	0.581	0.461	0.120
16	0.576	0.443	0.133
17	0.576	0.449	0.127
18	0.576	0.449	0.127
19	0.576	0.449	0.127

Thus, at time 0 since $C = 0.091$, no swapping is necessary.

At time 1, with records of type 4 still in the secondary database, the p^Q_{Bj}s, $H_D(Q)$ and C are computed and shown as the second row entries in the respective tables. C is now 0.121, again acceptable.

Similar entries are made for time 2 with $C = 0.115$, still acceptable.

At time 3, C is now computed and has an unacceptable value of 0.132. A reorganization step needs to be done. Records of type 4 need to be swapped into the main database. But which one of the records should be swapped out to the secondary database? According to the result in the previous section, it is most appropraite to select the record type that has the steepest slope in its $C - H_D(Q)$ plot. We therefore compute, for each j,

$$R = \frac{\Delta H_D(Q)}{\Delta C}$$

where ΔC is computed for times $t - 1$ and t, and $\Delta H_D(Q)$ is computed for times t and $t + 1$ by assuming that records of type j are swapped out of the main database and the reference patterns at time $t + 1$ are according to the $p_{t+1}(j)$s.

The records of that type j which has the largest R are then chosen as those to be swapped out to the secondary database.

In our example, at time 3, records of type 2 are accordingly chosen to be swapped out to the secondary database.

The values for C from time 4 to and including 11 indicate that no swapping is needed at those times. However, at time 12, with records of type 2 still in the secondary database, C becomes 0.131, an unacceptable value. A reorganization operation is executed and records of types 2 and 3 are swapped. From time 13 to time 19, the operation of the reorganization process continues in the same fashion, resulting in the corresponding entries in Tables 2 and 3.

For the example, Algorithm A results in 3 swaps. The hit ratio is 0.8 or better, except at 2 times when it is 0.7 (not counting times in which swapping takes place).

Algorithm B

As a comparison, we now look at the following simple reorganization algorithm based on a more conventional approach: at each time, records of the type least referenced are swapped out to the secondary database if they are not already there.

For our example, Algorithm B will result in 19 swaps with a hit ratio of 0.9 at all times.

The simple example shows the distinctiveness and effectiveness of the reorganization process based on the entropy approach. With Algorithm A, although the hit ratio sometimes suffers slightly, the number of swaps is significantly reduced. More interesting, however, is the following: Algorithm B can actually be obtained from Algorithm A by using $T = 0$ and $S = 0$. Thus Algorithm A is really a more general method than Algorithm B. This is actually not a very surprising finding, since the entropy approach has always been regarded as a more general method of solution. One implication is that the reorganization model presented in this paper is a potential candidate for use as a tool in the unified comparison of various diverse reorganization models.

CONCLUSIONS

We have presented a reorganization model that is based on the database entropy framework previously developed to characterize the learning process of a database. Whereas most of the other reorganization models available in literature include specific analytic expressions relating reorganization intervals to various relevant parameters (such as the expressions for the optimal checkpoint interval in Ref. 6 and those for the optimal reorganization interval in Ref. 8), the reorganization interval in our model is determined dynamically. Our reorganization model can therefore be most effectively employed to permit a database to adapt itself to continuously changing user requirements. As is evident from the illustrative example given in the preceding section, the model is conceptually simple and can readily be implemented in practice to provide a certain desired amount of intelligence in any kind of database management systems, from very simple ones to the more complex heterogeneous distributed database systems. In addition, we feel that the generality of the model makes it a potential candidate for use as a tool in the unified comparison of various reorganization models.

REFERENCES

1. K. T. Fung, A framework for characterizing the learning process of a database. *Kybernetes* **10**, 167–171 (1981).
2. A. I. Khinchin, *Mathematical Foundations of Information Theory*, Dover, New York (1957).
3. E. T. Jaynes, Where do we stand on maximum entropy. In *The Maximum Entropy Formalism*, edited by R. D. Levine and M. Tribus, MIT Press, Cambridge, Massachusetts, pp. 15–118 (1979).

4. M. H. Van Emden, Hierarchical decomposition of complexity. *Machine Intelligence* **5**, 361–380 (1969).
5. K. M. Chandy, A survey of analytic models of rollback and recovery strategies. *Computer* **8**, 40–47 (May 1975).
6. E. Gelenbe, On the optimum checkpoint interval. *JACM* **26**, 259–270 (1979).
7. B. Shneiderman, Optimal data base reorganization points. *CACM* **23**, 362–365 (1973).
8. S. B. Yao, D. S. Das and T. J. Teorey, A dynamic data base reorganization algorithm. *ACM TODS* **1**, 159–174 (1976).
9. K. T. Fung, Dynamic measurement of promptness and coherence of a distributed data base: the data base entropy approach. *Kybernetes* **11**, 205–209 (1982).
10. K. T. Fung and C. M. Lam, The data base entropy concept and its application to the data allocation problem. *INFOR* **18**, 354–363 (1980).

Received January 1982