Appendix 1

Comments on the determination of the number of partitions for MDD

This appendix shows that Liou and Yao's heuristic method of determination of the number of partitions for each partitioning dimension in their MDD file structure [LIOU77] is not optimal, even for the restricted class of queries that it is based upon. An improved method is presented.

Liou and Yao's determination of number of partitions

Determination of the values of m_i (number of partitions in each partitioning dimension) is based on the frequency of occurrence of query types. Each domain of attribute A_i consists of a set of discrete values. An atomic condition is defined as $A_i = a_i$, where a_i (i.e. the attribute value of attribute A_i) is in the domain of A_i. The most general query condition is defined as $q\alpha \,\hat{=} \bigwedge_{i \in \alpha} (A_i = a_i)$ where α is a string of integers *selected unrepetitively* from {1,2,...*K*}, where *K* is the number of attributes. For example, consider a partitioning based on branch region, age and surname. A query condition might be WHERE age = 27 AND surname = 'SMITH'. The determination of m_i is based on the assumption that queries are of this form (i.e. the integers in α are selected unrepetitively from *K*). The restriction that the integers in α are unrepetitive means that queries of the form $A_i = (a_{iv1}$ OR a_{iv2} OR ...), where a_{ivj} represents distinct attribute values from A_i, are barred from the calculation of m_i determination. In other words range queries such as WHERE age BETWEEN 27 AND 29 will not be considered in the determination of m_i values (of course, they are allowed to be asked, but this is a different matter).

Even ignoring this limitation (i.e. only point queries are considered), it is now shown that the heuristic method of determination of m_i is sub-optimal. Consider all possible query types as presented by Liou and Yao shown in table a1.1. The probability of a query type occurring is denoted by $P\alpha$, and the number of pages accessed by that query is $npa(\alpha)$.

$q\alpha$	$P\alpha$	$npa(\alpha)$
$A_1 = a_1$	P_1	N/m_1
$A_2 = a_2$	P_2	N/m_2
:		
$A_K=a_K$	P_K	N/m_K
$A_1=a_1 \wedge A_2 = a_2$	P_{12}	$N/(m_1 \times m_2)$
:		
$A_1 =a_1 \wedge A_2=a_2 \wedge \ldots A_K=a_K$	$P_{12\ldots K}$	1

$q\alpha$ = query type
$P\alpha$ = occurrence frequency of $q\alpha$
$npa(\alpha)$ = number of pages accessed by $q\alpha$

Table a1.1 All query types used to determine m_i

The average number of pages accessed by the query set is

$$\sum_a \left[Pa \cdot \mathrm{npa}(a)\right] = \sum_a \left[Pa \cdot \left(N / \prod_{i \in a} m_i\right)\right] \tag{a1.1}$$

Now ideally we want to determine m_1, m_2, ... m_K which minimise the total number of pages accessed for a given query set (i.e. various query types and their frequency of occurrence) within the constraint that the product of all m_i values is equal to the required number of pages, N (i.e. minimise $\sum_a \left[Pa \cdot \left(N / \prod_{i \in a} m_i\right)\right]$ subject to $\prod_{i=1}^{K} m_i = N$). This is analytically difficult, so a heuristic is presented.

It is suggested that when the probability of a query type is large, it is desirable to reduce the number of pages accessed, which is done by increasing the number of partitions for participating dimensions. Thus, m_i is made proportional to f_i, the sum of the probabilities of column$_i$ occurring in a query (i.e. $f_i \triangleq \sum_{a \text{ contains } i} Pa$), so $\frac{m_1}{f_1} = \frac{m_2}{f_2} = \ldots \frac{m_K}{f_K}$. The following formula is used to determine m_i values.

$$\left.\begin{array}{l} \frac{m_1}{f_1} = \frac{m_2}{f_2} = \ldots \frac{m_K}{f_K} \\ m_1 m_2 \ldots m_K = N \end{array}\right\} \rightarrow m_i = f_i \left(\frac{N}{\prod_{j=1}^{K} f_j}\right)^{1/K} \tag{a1.2}$$

Of course, as is shown in [LIOU77] the final values of m_i must be made integral, as equation (a1.2) does not guarantee this.

Making m_i proportional to f_i (as f_i is defined) is a reasonable approach in the case of atomic conditions. Consider a three dimensional case with N=1000, where N is the minimum number of pages required to store the data. Assume each of the attributes is referenced with equal frequency (query set 1 shown in table a1.2). Only the attribute references are shown rather than the full query type definition (e.g. A_i=a_i). Henceforth this simplification is used for clarity.

$q\alpha$	$P\alpha$	f_i
A	0.33	0.33
B	0.33	0.33
C	0.33	0.33

Table a1.2 Query set 1, atomic queries, equal attribute reference

Since all the queries are atomic, f_i, the sum of the probability of column$_i$ occurring in a query, is equivalent to the probability of the query.

f_1=f_2=f_3 $\Rightarrow$ m_1=m_2=m_3, and since $m_1 \times m_2 \times m_3 = N$, quite obviously all m_i = 10. This is quite reasonable, as all attributes are referenced identically by all queries.

This method is also suitable for a mix of conjunctive and atomic queries, where the f_i values due to the conjunctive predicates are in the same proportion as the f_i values due to the atomic conditions. For example, consider query set 2 shown in table a1.3.

qα	$P\alpha$
A	0.25
B	0.25
C	0.25
ABC	0.25

Table a1.3 Query set 2, atomic and conjunctive queries

The resultant f_i values are all identical (i.e. 0.5) since f_i is defined as the sum of the probability of column$_i$ being referenced in a query (e.g. f_A = 0.25 due to $q(A)$ plus 0.25 due to $q(ABC)$). Thus the resultant partitioning is identical to that derived for query set 1. Again, this is reasonable, since all dimensions are referenced similarly.

The problem

Now still with the 3 dimensional case, consider query set 3 which consists of the two query types shown in table a1.4.

qα	*P*α
A	0.5
BC	0.5

Table a1.4 Query set 3

Resultant f_i values for query set 3 are $f_A = f_B = f_C = 0.5$. Thus the partitioning will be symmetrical (10×10×10) as with the previous query sets. Table a1.5 shows the number of pages accessed, *npa*(α), by each of the query types and the average number of pages accessed.

*q*α	*P*α	*npa*(α)	*P*α × *npa*(α)
A	0.5	1000/10 = 100	50
BC	0.5	1000/100 = 10	5
		average *npa* =	55

Table a1.5 *npa* for query set 3 using Liou's determination of f_i

The average *npa* of 55 is not the optimum solution as is shown later. The reader may now guess that something is amiss. There are two equally likely query types, one of which references a single attribute, the other which references two attributes, and yet the partitioning is the same as for the cases where all queries reference all attributes equally. This is the result of making $f_i \triangleq \sum_{a \text{ contains } i} Pa$, since this definition of f_i effectively results in higher weighting being given to attributes which occur in conjunctive queries, than if those same attributes were to appear in atomic conditions.

Suggested solution

The problem is due to the way in which Liou and Yao see the original problem. They state that, "When *P*α is large, it is desirable to reduce *npa*(α)". Since, as shown in equation (a1.1), $\text{npa}(a) = N / \prod_{i \in a} mi$, as the number of referenced attributes increases, *npa*(α) decreases combinatorially (within the constraints set by the maximum possible number of partitions in any of the dimensions). A more appropriate method is to reduce *npa*(α) with respect to queries themselves, rather than with respect to frequency of attribute reference. This can be done by making $f_i \triangleq \sum_{a \text{ contains } i} \left(Pa / card(a) \right)$, where *card*(α) represents the number of attributes referenced by query *q*(α). For example if a query type references two columns, *A* and *B*, then instead of incrementing both f_A and f_B by *P*α, they would be incremented by *P*α/2 (and so $\sum_{i=1}^{K} fi = 1$ in all cases, rather than $\sum_{i=1}^{K} fi \geq 1$ by Liou and Yao's definition).

If this is done for query set 3 (previously shown in table a1.4) a better partitioning solution is found. First determine f_i.

$$q(A) = 0.5 \Rightarrow f_A = 0.5$$

$$q(AB) = 0.5 \Rightarrow \begin{cases} f_B = 0.5/2 = 0.25 \\ f_C = 0.5/2 = 0.25 \end{cases}$$

Using these f_i values, m_i values are determined using equation (a1.2).

$$m_A = f_A \left(\frac{N}{\prod_{j=1}^{K} f_j} \right)^{1/K} = 0.5 \times \sqrt[3]{\frac{1000}{0.5 \times 0.25 \times 0.25}} = 0.5 \times \sqrt[3]{32000} = 15.87$$

This long-winded approach is not needed to determine m_B and m_C, a re-arrangement of the expression $\frac{m_1}{f_1} = \frac{m_2}{f_2} = \ldots \frac{m_K}{f_K}$ suffices, thus

$$m_B = m_A \cdot \frac{f_A}{f_B} = 15.87 \cdot \frac{0.25}{0.5} = 7.94$$

Similarly, $m_C = 7.94$

Making m_i integral the values shown in table a1.6 are obtained.

dimension	calculated m_i	integral m_i
A	15.87	16
B	7.94	8
C	7.94	8

Table a1.6 Calculated and integral m_i values

Thus there are 16×8×8 = 1024 pages in total. Table a1.7 shows the number of pages accessed by queries in set 3 using this partitioning.

$q\alpha$	$P\alpha$	$npa(\alpha)$	$P\alpha \times npa(\alpha)$
A	0.5	1024/16 = 64	32
BC	0.5	1024/(8×8) = 16	8
		average *npa* =	40

Table a1.7 *npa* for query set 3 using improved f_i determination

The average *npa* in this case is 40, whereas the partitioning derived using Liou's method of determining f_i results in access to 37% more pages (i.e. 55) for the example query set, as shown in table a1.5.

In practice, it may be that the pessimistic solutions obtained using Liou and Yao's method may not be excessively poor if a large proportion of queries are atomic, or if, with a mix of atomic and conjunctive queries, the conjunctive queries reference the same columns as the atomic ones.