Proceedings of the IASTED International Conference
Software Engineering and Applications (SEA'99)
October 6-8, 1999, Scottsdale, Arizona, USA

AN OBJECT-ORIENTED APPROACH FOR MANAGING A NETWORK OF DATABASES

Shu-Ching Chen
School of Computer Science
Florida International University
Miami, FL 33199

Mei-Ling Shyu
School of Electrical and
Computer Engineering
Purdue University
West Lafayette, IN 47907

Chi-Min Shu
Department of Environmental
Safety Engineering
National Yunlin University of
Science and Technology
Yunlin, Taiwan, R.O.C

ABSTRACT

A large scaled network may consist of hundreds of disparate and autonomous databases. Users in such an information-providing environment usually access information from those databases in the same or similar application domains so that there is no need to handle all the entities from all the databases. That is, in most cases, only a subset of databases is required for the users' requests. As the number of databases increases, the need to manage such a network of databases increases. In this paper, we present a split/cluster approach using the object-oriented technique to allow users to incrementally and dynamically access the information they want without being overwhelmed with all of the unstructured information. The approach is based on the affinity relationships of the databases and is performed recursively to split these databases into clusters. Then a cluster hierarchy is formed to provide different levels of abstractions for the users. This framework provides a flexible means of sharing information to all the databases. Theoretical terms along with a running example are presented.
Key words: Object-oriented databases, affinity, clustering, splitting.

1. INTRODUCTION

Integrating heterogeneous databases is a challenging problem since incompatibilities exist among all the databases [3]. To provide as transparent as possible a database schema, conflicts need to be resolved before it can provide a view to the users. A number of researchers have investigated the problem of integrating heterogeneous databases [1] [2] [5]. However, the issues of conflict resolutions are not discussed in this paper since we try to focus on managing the network of databases to help users better utilize the information in the databases.

In such a large scaled database network, queries tend to traverse data related to the same or similar application domains and which reside in different databases. Most of the queries request information from a small fraction of the databases in the network without the need to show all the entities of all the databases. This motivates us to split the network of databases recursively into beneficial clusters based on the access behavior of application queries. An example of grouping close to the concept of ours is the Internet [4]. The Internet is a computer network consisting of several connected subnetworks. Every subnetwork follows its own communication protocols and is usually set up to serve some special purposes. One difference between these two concepts is that all the subnetworks provide almost the same set of information; while each cluster in the proposed approach can provide diverse sets of information.

In this paper, an object-oriented split/cluster approach is proposed. The object-oriented paradigm is adopted since things in the world around us have properties or features; we can think of data as an object class with its defining attributes. The affinity measures between every pair of databases are formalized and calculated based on the access behavior of application queries. Each query may be activated several times and hence each query has its access frequency. Therefore, the access frequency of a query per time period should be taken into account in the affinity measures. The splitting procedure is based on the affinity relationships of the databases and is performed recursively to split these databases into clusters. After the split/cluster step, a cluster hierarchy is generated. The cluster hierarchy provides different levels of abstractions and hence allows users to incrementally and dynamically access the pieces of information they want without being overwhelmed with all of the unstructured information. The constructed clusters can be used as the unit not only for query processing but also for discovering the object-oriented relationships such as superclass, subclass, and equivalence relationships, which is the subject of a forthcoming paper. For those users who wish

to access only parts of the databases, they can access the data from the appropriate clusters without going through the whole network of databases. In other words, the proposed approach provides a flexible means of sharing information to all the databases.

This paper is organized as follows. In Section 2, the proposed object-oriented approach with relative affinity formulations and the split/cluster procedure is introduced. A simple example is given to illustrate the steps of the split/cluster procedure. Section 4 concludes this paper.

2. PROPOSED OBJECT-ORIENTED APPROACH

A set of historical queries which are issued to the databases in the network is used as *a priori* for the split/cluster procedure. We use the relative affinity values to measure how frequently two databases have been accessed together in the set of historical queries. Realistically, it cannot be expected that the user applications are able to specify these affinity values and hence formulas need to be defined.

2.1. Relative Affinity Measures

Let Q = $\{q_1, q_2, \ldots, q_q\}$ be the set of queries that run on the set of databases D = $\{d_1, d_2, \ldots, d_d\}$ in the large scaled database environment. Define the variables:

• $\mathbf{use_i}(\cdot)$ = a vector of length **q** indicating the usage patterns of d_i with respect to all the queries in Q. For each database d_i, $\mathbf{use_i}(\cdot)$ is defined as follows and the kth entry of $\mathbf{use_i}(\cdot)$ denotes the usage pattern of d_i with respect to q_k.

$$\mathbf{use_i}(\cdot) = \begin{cases} 1 & \text{if object classes in } d_i \text{ is accessed by } q_k \\ 0 & \text{otherwise} \end{cases}$$

• $\mathbf{access}(\cdot)$ = a vector of length **q** indicating the access frequencies of the queries in Q per time period. The *kth* entry of $\mathbf{access}(\cdot)$ denotes the access frequency for query q_k.

• $\mathbf{rel(i,\ j)} = \sum_{k=1}^{q} \mathbf{use_i}(\mathbf{q_k}) \times \mathbf{use_j}(\mathbf{q_k}) \times \mathbf{access}(\mathbf{q_k})$
= the affinity value of database d_i and d_j.

• **M**=a matrix of size $g \times g$ indicating the *affinity measures* of the databases in a group DB_GROUP_{IJ} with respect to all queries in Q assuming $DB\_GROUP_{IJ} = \{d_1, d_2, \ldots, d_g\}$. The $\mathbf{rel(i,\ j)}$ value is placed at the *(i,j)th* entry in **M**. Note that **M** is a symmetric matrix and for simplicity, only the entries which $i \leq j$ are computed. The *(i,j)th* entry will have the same result as the *(j,i)th* entry and the *(i,i)th* entry will not be used in the split/cluster procedure.

• $\mathbf{PP(i,j)}$ = a *closeness difference* function which calculates the closeness difference between column i for d_i and column j for d_j. Let **O** represent a temporary matrix in the split/cluster procedure which contains the first several columns of **M**. For every possible pair of neighbors d_i and d_j, $\mathbf{PP(i,j)}$ is defined as follows:

$\mathbf{PP(i,j)}$=**M**(1,i)-**O**(1,j) if d_i is put to the left of d_j
$\mathbf{PP(j,i)}$=**O**(1,j)-**M**(1,i) if d_i is put to the right of d_j

2.2. Split/Cluster Procedure

The objective of the split/cluster procedure is to find several clusters of databases that are accessed together more frequently by the set of queries. For a large scaled database environment, this split/cluster procedure should be invoked iteratively to form the cluster hierarchy.

The split/cluster procedure takes the primary data as inputs, computes the entities of the matrix **M**, calculates the closeness difference values, permutes its columns, and then generates an updated matrix **M**. A function $\mathbf{PP(i,j)}$ is defined to calculate the difference of two affinity values of the nearby neighbors for each possible position (d_i, d_j) based on the entries in **M**. The permutation is done by considering the minimum of the **PP** values for each database. The $\mathbf{PP(i,j)}$ function is designed to be the closeness difference for two columns i and j. Let column i be the one that needs to be placed in the temporary matrix **O** where **O** consists of the first several columns of matrix **M**. Column i can be placed on the left or right of column j in **O**. The main idea is to position column i in the place which satisfies two conditions: its affinity measure should be less than or equal to the affinity measure of its left neighbor and greater than or equal to the affinity measure of its right neighbor. For the leftmost or the rightmost position of **O**, simply consider one of the above two conditions because it has only one neighbor in such cases. For each closeness difference value, check whether it is less than zero. If yes, ignore this possible position since a negative difference means the required conditions are not satisfied.

Since the procedure computes only the closeness differences of the nearby neighbors and considers the minimum of the differences, it tends to partition the matrix **M** into two clusters - one is in the upper left corner and the other is in the lower right corner. In general the border for the split is not very clear-cut. For this purpose, a splitting phase is proposed to decide the split point. The splitting phase compares the mean value of the first column with each individual value in that column in **M**. If the individual value is greater than or equal to the mean value, then it belongs to the upper left corner group. Other-

wise, it belongs to the lower right corner group. Two clusters can therefore be generated at each iteration. The mean value of the first column is chosen to be the splitting criterion since the first column tends to have the larger affinity values. However, there must be some stopping criteria to end the iterations. There are two stopping criteria for each split/cluster procedure iteration: (1) when the size of a cluster is one, i.e. the number of databases in the cluster is one, and (2) when the size of a cluster is less than four. If one of the above conditions is satisfied, then there is no more splitting for that cluster since it makes no sense to have a cluster with only one element in the cluster. Otherwise, each cluster executes the split/cluster procedure iteratively until one of the conditions is met.

Initially, the split/cluster procedure is applied to all the databases in the network. The procedure is iterated until no more splitting is permitted. Steps for the split/cluster procedure:

1. Preparation of the primary data:

   The primary data required are **access(·)** and $\mathbf{use_i}(\cdot)$ where i=1 ... d (d is the total number of databases). These vectors are given as *a priori* from a set of historical queries. However, since the application queries issued to the databases can be recorded per time period (say monthly or annually), the required data can be updated accordingly.

2. Computation of the entries in **M**:

   $\mathbf{rel(i,j)} = \sum_{k=1}^{q} \mathbf{use_i(q_k)} \times \mathbf{use_j(q_k)} \times \mathbf{access(q_k)}$, where **i, j**=1 ... d.

3. Determination of the cluster size:

   Each cluster in the cluster hierarchy is an input to the split/cluster procedure. The size of a cluster (g) is the number of databases in the cluster. Initially, g=d since the input cluster consists of all the databases in the network. Assuming

   $DB\_GROUP_{IJ} = \{d_1, d_2, \ldots, d_g\}$, $g <= d$
   $\Longrightarrow$ the size of $DB\_GROUP_{IJ} = g$.

4. **for** $loop1$ = 1 to g-1

   /* initialization of the matrix **O** */
   /* place the first $loop1$+1 columns of **M** into **O** */
   $\mathbf{O}(\cdot, 1) = \mathbf{M}(\cdot, 2)$; $\mathbf{O}(\cdot, 2) = \mathbf{M}(\cdot, 2)$;
   ...
   $\mathbf{O}(\cdot, loop1{+}1) = \mathbf{M}(\cdot, loop1{+}1)$;
   for $loop2$ = $loop1$+2 to g
   - For each column in the the remaining g-($loop1$+1) columns, calculate a **PP** vector for the $loop2$ possible positions for that column. Select the position by the minimum **PP** value.
     /* position selections */
   - Once the position for the column is determined, permute the columns in **O** if necessary.
     Place the column into its corresponding position in **O**. /* column permutations */

   end
   Once the positions for the remaining g-($loop1$+1) columns are determined, the permutations of corresponding rows are performed so that the relative positions in **O** are maintained.
   /* row permutations */
   M = **O** /* update **M** */
   end

5. Splitting phase:

   Compute the mean value of the first column of the matrix **M**. This mean value is then used as the criterion for the splitting phase. Two clusters are generated from the matrix **M** after the splitting phase is applied.

6. Stopping criteria checking for each cluster generated in step 5:

   - If the size of the cluster is one, then no splitting for this cluster and stop.
     Else goto step 5 for each cluster.
   - If the size of the cluster is less than four, then no splitting for this cluster and stop.
     Else goto step 5 for each cluster.

7. Generating a cluster hierarchy:

   After all the clusters execute the split/cluster procedure and finish the stopping criteria checking, a cluster hierarchy for all the databases in the network can be created.

3. AN EXAMPLE

In this section, a simple example is used to illustrate the proposed split/cluster procedure. Once the network of databases is partitioned into several clusters and each cluster consists of one or more databases which have high affinity relationships, the cost of query processing can be reduced.

Example: Suppose there are 10 databases in the network and the historical data consists of 8 queries. Let $D = \{d_1, d_2, \ldots, d_{10}\}$ and $Q = \{q_1, q_2, \ldots, q_8\}$. Assume the following $\mathbf{use_i}(\cdot)$ where i=1 ... 10, and **access(·)** values are the required primary data obtained from the set of historical data.

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
d_1	110	70	25	0	110	0	110	95	25	0
d_2	70	105	35	0	70	0	70	70	35	0
d_3	25	35	190	30	25	30	25	125	190	30
d_4	0	0	30	65	0	65	0	0	30	65
d_5	110	70	25	0	160	0	110	95	25	0
d_6	0	0	30	65	0	65	0	0	30	65
d_7	110	70	25	0	110	0	110	95	25	0
d_8	95	70	125	0	95	0	95	195	125	0
d_9	25	35	190	30	25	30	25	125	190	30
d_{10}	0	0	30	65	0	65	0	0	30	65

Figure 1: Initial Affinity Measure Matrix **M**. Each entity (i,j) in **M** has the relative affinity value **rel(i,j)**, where i,j=1 to 10.

	d_7	d_5	d_1	d_8	d_2	d_9	d_3	d_4	d_{10}	d_6
d_7	110	110	110	95	70	25	25	0	0	0
d_5	110	160	110	95	70	25	25	0	0	0
d_1	110	110	110	95	70	25	25	0	0	0
d_8	95	95	95	195	70	125	125	0	0	0
d_2	70	70	70	70	105	35	35	0	0	0
d_9	25	25	25	125	35	190	190	30	30	30
d_3	25	25	25	125	35	190	190	30	30	30
d_4	0	0	0	0	0	30	30	65	65	65
d_{10}	0	0	0	0	0	30	30	65	65	65
d_6	0	0	0	0	0	30	30	65	65	65

Figure 2: Final Affinity Measure Matrix. The dashed lines separate two clusters.

- $\mathbf{use_1}(\cdot)$ = [1 0 0 1 1 0 0 0];
- $\mathbf{use_2}(\cdot)$ = [0 0 0 1 0 0 1 0];
- $\mathbf{use_3}(\cdot)$ = [0 1 0 0 1 0 1 1];
- $\mathbf{use_4}(\cdot)$ = [0 0 1 0 0 0 0 1];
- $\mathbf{use_5}(\cdot)$ = [1 0 0 1 1 1 0 0];
- $\mathbf{use_6}(\cdot)$ = [0 0 1 0 0 0 0 1];
- $\mathbf{use_7}(\cdot)$ = [1 0 0 1 1 0 0 0];
- $\mathbf{use_8}(\cdot)$ = [0 1 0 1 1 0 0 0];
- $\mathbf{use_9}(\cdot)$ = [0 1 0 0 1 0 1 1];
- $\mathbf{use_{10}}(\cdot)$ = [0 0 1 0 0 0 0 1];
- $\mathbf{access}(\cdot)$ = [15 100 35 70 25 50 35 30];

With the availability of the primary data, the relative affinity values can be calculated. For example, the affinity measure for the entity **M**(1,5) can be obtained by the following way.

$$\mathbf{M}(1,5) = \mathbf{rel}(1,5)$$
$$= \sum_{k=1}^{8} \mathbf{use_1}(\mathbf{q_k}) \times \mathbf{use_5}(\mathbf{q_k}) \times \mathbf{access}(\mathbf{q_k})$$
$$= \text{access}(q_1) + \text{access}(q_4) + \text{access}(q_5) = 110.$$

Similarly, all the **rel(i,j)** entities for **M** can be computed. The initial affinity measure matrix **M** is shown in Figure 1. As shown in Figure 1, each entity (i,j) in **M** has the value of **rel(i,j)** which indicates the relative affinity measure for database d_i and d_j. In addition, **M** is symmetric so that the entity (i,j) has the same value as in the entity (j,i). For example, **M**(1,5) and **M**(5,1) have the same value 110.

Take the initial affinity measure matrix **M** and execute the first iteration, i.e., when $loop1$=1. According to our proposed split/cluster procedure, initially the first two columns of **M** are placed into the temporary matrix **O** and column 3 (i.e., d_3) is considered next. There are three possible positions for column 3: to the left of column 1 (computing **PP**(3,1)), in between column 1 and 2 (computing **PP**(1,3) and **PP**(3,2)), and to the right of column 2 (computing **PP**(2,3)).

Position 1: to the left of column 1,
PP(3,1) = **M**(1,3) - **O**(1,1) = 25 - 110 < 0;

Position 2: in between column 1 and column 2,
PP(1,3) = **O**(1,1) - **M**(1,3) = 110 - 25 = 85;
PP(3,2) = **M**(1,3) - **O**(1,2) = 25 - 70 < 0;

Position 3: to the right of column 2,
PP(2,3) = **O**(1,2) - **M**(1,3) = 70 - 25 = 45;

Since the **PP** values for positions 1 and 2 are negative, these two possibilities are ignored. Therefore, we select to place d_3 to the right of column 2. Similarly, all the other databases are calculated to get an updated matrix. Finally, the rows are permuted to be in the same order as the columns. The same steps are applied for iterations 2 to 9 to get the final affinity measure matrix **M** which is then used to illustrate the splitting phase.

First, the mean of the first column is calculated.

mean = (110+110+110+95+70+25+25+0+0+0)/10
= 54.5;

According to the proposed splitting phase, the mean value is used to consider the splitting of the matrix **M**. Therefore, two clusters are created: one is in the upper left corner and the other is in the lower right corner (see the dashed line in Figure 2). Let the upper left corner cluster be $DB\_GROUP_{21} = \{d_1, d_2, d_5, d_7, d_8\}$ and the lower right corner cluster be $DB\_GROUP_{22} = \{d_3, d_4, d_6, d_9, d_{10}\}$. Since both clusters contain more than three databases, each cluster needs to execute the split/cluster phase iteratively. Again, the mean value for each cluster needs to be calculated and used as the splitting criterion. Based on the mean values, the clusters DB_GROUP_{21} and DB_GROUP_{21} are further split into two more

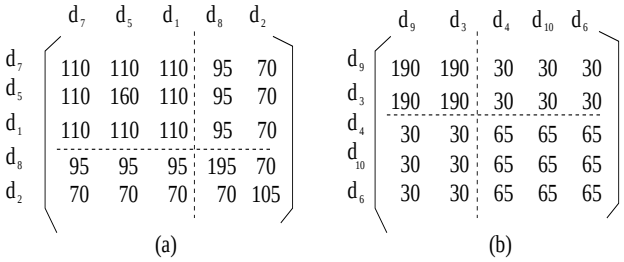

Figure 3: Splitting for the two clusters. Each cluster can be split into two more clusters (as shown in the dashed lines).

clusters individually (the dashed lines in Figure 3(a) and 3(b)). Since the numbers of databases in all the four clusters are less than four, the procedure stops.

mean = (110+110+110+95+70)/5 = 99
(for DB_GROUP_{21})
mean = (190+190+30+30+30)/5 = 94
(for DB_GROUP_{22})

After all the clusters execute the split/cluster procedure and the stopping criteria checking, a cluster hierarchy for all the databases in the network can be created. As shown in Figure 4, the cluster DB_GROUP_{11} which consists of all the databases in the network is at the root of the hierarchy. Initially, the split/cluster procedure starts with DB_GROUP_{11} and partitions it into two clusters DB_GROUP_{21} and DB_GROUP_{22}. Then, DB_GROUP_{21} can be partitioned into DB_GROUP_{31} and DB_GROUP_{32}, and DB_GROUP_{22} can be partitioned into DB_GROUP_{33} and DB_GROUP_{34}. Each finer cluster consists of its own databases. Those databases in the same cluster should be highly affiliated and be accessed for query information. This cluster hierarchy is then used to decide where a query should be searched for the requested information to reduce the cost of query processing.

4. CONCLUSIONS

In this paper, we propose an object-oriented approach to partition a large scaled network of databases into a set of clusters. We have formalized a new set of relative affinity measures to represent how frequently two databases have been accessed together by a set of historical queries. Affinity-based measures are both intuitively reasonable and understandable since they consider the access frequencies of queries. We gave a split/cluster procedure for clustering the databases. The split/cluster procedure includes a splitting phase and two stopping criteria, and is executed iteratively.

A simple example is run to illustrate the steps of the proposed split/cluster procedure. A cluster hierarchy which provides different levels of abstractions for users to incrementally and dynamically access the information is formed. Since a set of databases belonging to a certain application domain is placed in the same cluster and is required consecutively on some query access path, the number of platter switches for query processing can be reduced. Moreover, the constructed clusters can be used as the unit not only for query processing but also for discovering the object-oriented relationships such as superclass, subclass, and equivalence relationships.

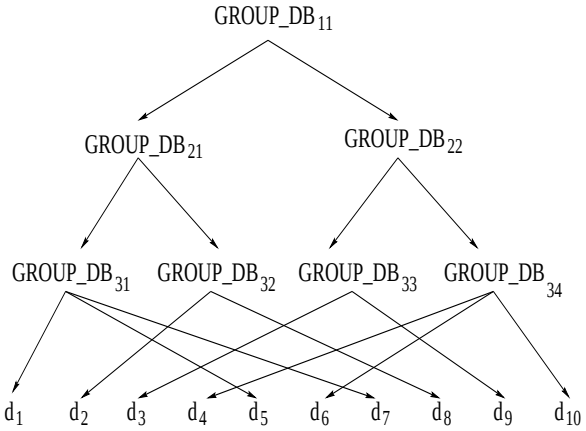

Figure 4: The resulting cluster hierarchy. The lowest level of the hierarchy consists of the individual databases in the network. The root cluster of the hierarchy consists of all the databases. The clusters at each level have their own member databases.

References

[1] D.M. Dilts and W. Wu, Using knowledge-based technology to integrate CIM databases, *IEEE Trans. Knowledge Data Eng.*, vol. 3(2), June 1991.

[2] W. Gotthard, P.C. Lockemann, and A. Neufeld, System-guided view integration for object-oriented databases, *Knowledge Data Eng.*, vol. 4(1), Feb. 1992.

[3] W. Litwin, L. Mark, and N. Roussopoulos, Interoperability of multiple autonomous databases, *ACM Computing Surveys*, 22, 1990, pp. 267-293.

[4] J.S. Quarterman and J.C. Hoskins, Notable computer networks, *Communication of ACM*, vol. 29(10), 1986, pp. 932-971.

[5] M.P. Reddy, B.E. Prasad, P.G. Reddy, and A. Gupta, A methodology for integration of heterogeneous databases, *IEEE Trans. Knowledge Data Eng.*, vol. 6(6), December 1994.