A Case-Based Reasoning Approach for Associative Query Answering*

Gilles Fouqué, Wesley W. Chu, and Henrick Yau

Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024
{gfouque, wwc, henrick}@cs.ucla.edu

Abstract. Traditional DBMS only retrieves data that perfectly match the user query and also requires the user to know the detailed database schema. Often, it is desirable to obtain additional relevant information to a query. In this paper, we present a method to provide useful information to the user that he does not explicitly asked for. Such domain specific knowledge associated to a given query depends on each user's goal and knowledge. Thus, we propose using the Case Based Reasoning paradigm to integrate past user experience and the current goal in order to guide the association. Useful associations are incrementally acquired from observations of past experiences and adapted to answer the current user query. A prototype of the associative query answering system using the proposed method has been implemented on top of the cooperative database system, CoBase, at UCLA. Our preliminary experimental results reveal that it is a feasible and scalable method for association control.

1 Introduction

Traditional query answering systems require a perfect match between the user query and the data to be retrieved. This approach requires the user to have a detailed knowledge of both the database schema and the query language. Moreover, the user has to know exactly what information to search. *Cooperative query answering* has been recently developed to relax these limitations where different approaches are used to broaden the scope of the user query. Cooperative databases extend the traditional database query languages by providing new operators (*"near-to", "similar-to"*) and by providing explanation of the relaxation process and quality of the answers [CMB93]. In this paper, we focus on a specific aspect of cooperative databases: the inclusion of information not explicitly asked by the user but relevant to his query. This operation is called *associative query answering* [CD91][CHF92] that provides additional information that either the user does not know how to ask for or does not know is available.

We propose to use query context to infer the user search goal. The query context is based on the analysis of the user's previous experience: his previous

* This work is supported by DARPA contract N00174-91-C-0107

searches, and the ability of the system to infer his successive goals. A Case-Based Reasoner[RS89] is used to store, index, select, and adapt this experience and while learning from previous system successes and failures[FM93].

In this paper, we first present an example of associative query answering, followed by a presentation of the association system architecture. Then, we discuss CBR learning from user feedback. Finally, we present our CBR implementation.

2 Example of Associative Query Answering

Let us consider the following query: *"List all airports in Tunisia with runway length approximately equal to 10.000 feet"*

The answers provided by CoBase[CMB93] with approximate operators are:

Airport Name	*Runway Length in Feet*
Jerba	10171.00
Monastir	9700.00
Tunis	10500.00

It would be desirable to provide additional relevant information, even when not requested in the query. Further, this additional information depends on user type. For example, in the context of a transportation application, if the user is a pilot, then information on weather and runway quality are useful as follows:

Query Answers		Associative Query Answers	
Airport Name	*Runway Length in Feet*	*Weather Condition*	*Runway Quality*
Jerba	10171.00	SUNNY	GOOD
Monastir	9700.00	RAIN	GOOD
Tunis	10500.00	FOGGY	DAMAGED

If the user is a transportation planner trying to find the best schedule for delivery, then relevant information for the user might be: the category of the airport (civilian or military), the availability of a close seaport and the availability of refrigeration capacities as shown below:

Query Answers		Associative Query Answers		
Airport Name	*Runway Length in Feet*	*Military or Civilian*	*Seaport Name*	*Refrigerated Storage Capacity in Tons*
Jerba	10171.00	C	None	0
Monastir	9700.00	C	Monastir	0
Tunis	10500.00	C	Tunis	1000.00

Therefore, in selecting the list of relevant subjects, the associations are based on query context as well as user type.

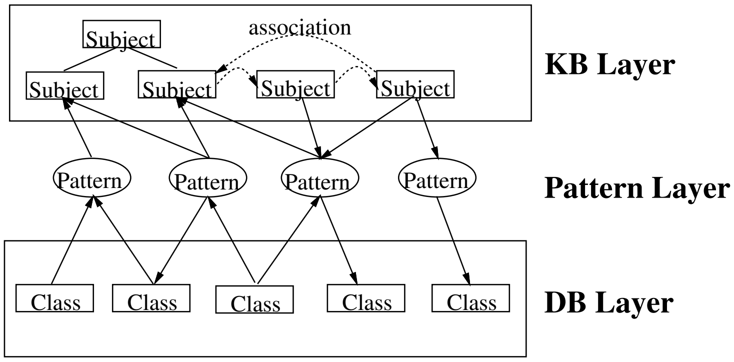

Fig. 1. *The DB-pattern-KB Architecture*

3 The association facility

3.1 The DB-pattern-KB architecture

Associative query answering requires the linkage of data and knowledge. We propose a DB-pattern-KB architecture (Figure *1*) for associative query answering.

- *The DataBase (DB) layer:* contains the relational or object-oriented databases. A class encapsulates the properties common to a group of data.
- *The Knowledge Base (KB) layer:* contains the application domain knowledge not already included in the database layer. This knowledge is composed of rules and facts. The KB layer is organized into sets corresponding to subjects as defined by the domain expert. The associations are performed in the KB layer by explicit links between different subjects.
- *The Pattern layer:* provides the interface between the DB and the KB layers. A pattern is a set of query conditions belonging to one or several subjects. It provides a precise and applicable knowledge representation.

Associative query answering is accomplished through tracing the dependencies between data distributed in multiple subjects. In such a framework, the knowledge base and database layers can be maintained independently[CC92].

3.2 Association control

Since association is a transitive process depending on user type and goal, it needs control for selection termination. In the previous example, we assumed that such association control was available. Otherwise, all the related subjects in the DB-pattern-KB framework would be selected for association. Therefore, we propose to use Case Based Reasoning (CBR) to control the association and to acquire information about the user goals. The user queries and their answers are grouped into cases. The Case Memory is composed of previous cases related by links between a case and its associations. Weights are assigned to the links to represent their usefulness for the association to a specific query context and user type. The CBR uses user's feedback to dynamically update the link weight between a case and its associations as its learning from its successes and failures.

3.3 The association representation

A set of queries with their corresponding answers and a domain specific knowledge base are stored as cases in the Case Memory. The cases provide attributes for association and their usefulness for association is learned during the use of the CBR. Associations to a user query are achieved by first adding a set of attributes to a select list of the user's query and then running the modified query.

In the Case Memory, an association is represented by an *association link* between two queries (cases) that share one or more attributes. Because all the attributes are not equally important for an association, a weight is assigned to each attribute of an association link. The larger the weight, the more relevant the association.

The cases in the Case Memory are acquired incrementally during the use of the CBR. There are two approaches to acquire the initial set of cases: *(a)* an automatic generation of queries based on the distribution of relations and attributes in the database, or *(b)* based on user's previous transactions.

In the first case, the association weights are initialized according to the uniform distribution. Thus, prior to any learning, each attribute has an identical chance for association. The association initialization weight for an attribute a_k shared by two cases C_i and C_j is:

$$W'(C_i \rightarrow C_j, a_k) = 1/t, \text{ for } k = 1, \ldots, t \tag{1}$$

When previous user transactions are used for the initialization of the association weights, the initialization weight of an association link can be based on additional information related to the distribution of each attribute in the Case Memory:

$$W(C_i \rightarrow C_j, a_k) = -\log_2(F(a_k)) \tag{2}$$

where: $\sum_{k=1}^{t} F(a_k) = 1$;
$F(a_k)$ = frequency of occurrence of attribute a_k in all the Case Memory.

Since a seldomly used attribute carries more specific information for association, its weight is initialized with a higher value than that of frequently used attributes *(2)*. A Case Memory example is shown in Figure *2*. C_1, C_2, C_3, and C_4 are four previous user queries. The initialization of association weights, $W(C_i \rightarrow C_j, a_k)$, is symmetric and equal to $W(C_j \rightarrow C_i, a_k)$, for $i, j = 1, 2, 3, 4; i \neq j$. The Case Memory includes 4 association links, each one with a single attribute: *runway_length_ft* or *airport_name*. Thus, the total number of attributes used in the association links is $4 * 2 = 8$. From *(2)*, the initial association weights for attribute *runway_length_ft* and *airport_name* are[2]: $W(C_1 \rightarrow C_3, runway\_length\_ft) =$

[2] latter, the association weights are normalized by the highest association weight in the Case Memory

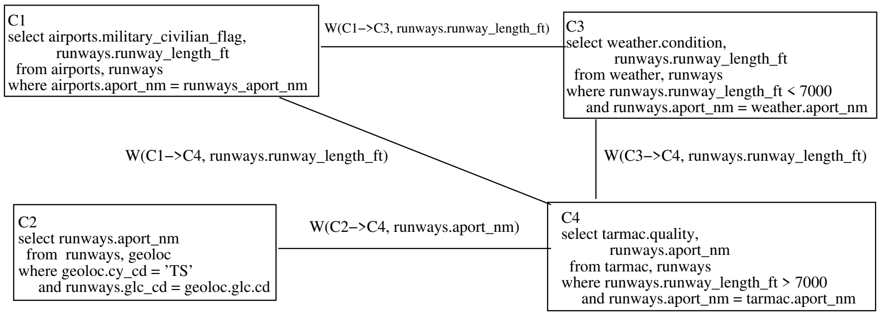

Fig. 2. *Case Memory*

$W(C_1 \rightarrow C_4, runway\_length\_ft) = W(C_3 \rightarrow C_4, runway\_length\_ft) = -\log_2(F(runway\_length\_ft)) = -\log_2(3 * 2/8) = 0.42$ and $W(C_2 \rightarrow C_4, airport\_name) = -\log_2(F(airport\_name)) = -\log_2(2 * 1/8) = 2$.

The attributes associated to the attribute *airport-name* is considered more specific than attributes associated to *runway_length_ft*. Consequently, if the user query includes both attributes, the attributes provided for association by the association link through *airport_name* between C_2 and C_4 are more important than those provided by the association links *runway_length_ft* between C_1 and C_3, between C_1 and C_4, and between C_3 and C_4.

3.4 Selection and ranking of associated attributes

We shall now discuss the selection and ranking processes of finding the set of similar cases for the user query. To find the attributes associated with the user query Q, the CBR searches the Case Memory for all association links contained in the set of attributes in Q.

The CBR searches in the Case Memory for the association links that use one or more of the attributes in Q. The cases C_i's related to these association links are selected. The CBR evaluates their usefulness from the comparison of Q and C_i's query contexts, and the weights of the association links.

Two queries perfectly match when they have identical relations, attributes, and attribute values. When two queries share the same relation and attribute but with different values, we use domain specific knowledge to measure the relevance of the substitution of one value by another. This knowledge is represented in the Type Abstraction Hierarchies (TAH) where attribute values are clustered depending on a nearness function represented as a correlation measure[MC93]. This nearness function is based on several features: *a) answer set locality* — instances that are often selected together are considered to have higher nearness; *b) inter-attribute relationship* — the nearness of two values of the same attribute depends on the number of values they share with the other attributes in the relation. Therefore, two cities both having a seaport and in the same country are more correlated than two cities having no shared attribute values. Thus, two queries with identical attributes but poorly correlated values are not equivalent.

The similarity between the user query Q and the query context of case C_i is:

$$S(Q, C_i) = \sum_{k=1}^{t} C_i(a_k) * Q(a_k) * \gamma(a_k; x, y) \tag{3}$$

where:

$$Q(a_k) = \begin{cases} 1 \text{ if the attribute } a_k \text{ is used in the user query } Q, \text{ for } k = 1, \ldots, t, \\ 0 \text{ otherwise} \end{cases}$$

t = total number of attributes in the database.

$$C_i(a_k) = \begin{cases} 1 \text{ if the attribute } a_k \text{ is used in the case } C_i, \text{ for } k = 1, \ldots, t \\ 0 \text{ otherwise} \end{cases}$$

$\gamma(a_k; x, y)$ = the correlation value between the values x in Q and y in C_i in the TAH for attribute a_k.

The usefulness for Q of an attribute a in case C_j associated with C_i, $U(Q, a)$, depends on the similarity between the user query Q and C_i, $S(Q, C_i)$, and the weight of the attributes in the association link, $W(C_i \rightarrow C_j)$; that is,

$$U(Q, a) = S(Q, C_i) * \sum_{k=1, a \neq a_k}^{t} W(C_i \rightarrow C_j, a_k) \tag{4}$$

The list of associated attributes is ranked based on their usefulness for association to Q *(4)* and presented to the user for feedback. The selected attributes are appended to the user query to derive the associated information.

3.5 An indexing schema for reducing computation complexity

The user query Q is first matched with the association links to select the cases that are likely to provide useful associated attributes. A traditional browsing of the Case Memory association links is computationally too costly since it increases exponentially with the number of stored cases. To reduce the computation complexity, we propose to use an inverted file of the Case Memory to compare Q to a limited set of association links that are more likely to lead to useful cases for associations. The inverted file can be generated immediately after the initialization of the Case Memory and updated during the learning process. In the inverted file a branch is created in the indexing tree for each association link in the Case Memory. The attributes in a branch are ordered based on their weights in the association links. The total weight of each branch is computed by summing the corresponding association weights of that branch. The highest the total weight, the more relevant is the associated attribute provided by that association link. A link is created between the inverted file and the Case Memory to list all the association links for each branch leaf.

The selection process using the proposed indexing schema is as follows: *a)* based on the set of possible combinations of attributes used in user query Q,

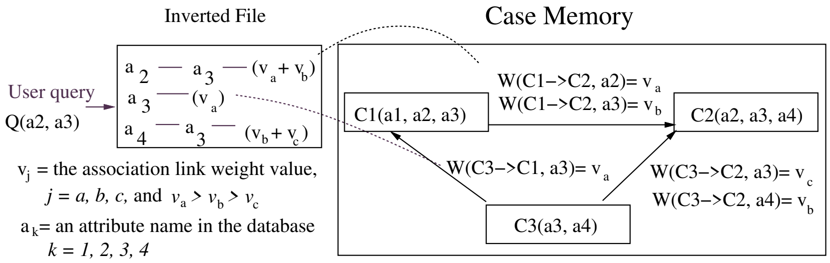

Fig. 3. *The indexing schema*

.

select a set of association links and their corresponding total weight from the indexing trees in the inverted file, *b)* select the m highest association links and their corresponding cases, *c)* compute the usefulness of association for the attributes of each association link from *(4)*.

By using this inverted file, the complexity of the selection process is in $O(n^2)$ where n is the number of attributes in Q. Note the complexity is independent of the number of stored cases. This efficient indexing schema is possible because we know the entire set of database attributes used in the user queries.

An example of the indexing schema with three cases $C_1(a_1, a_2, a_3), C_2(a_2, a_3, a_4)$, and $C_3(a_3, a_4)$ is presented in Figure *3*. Three association links are represented in the Case Memory with the attributes shared by the related cases and weights. For the branches included in the user query Q, the relations between the leaves of the inverted file trees and their corresponding association links are represented by broken lines. Q has two attributes a_2 and a_3, four combinations of these attributes are possible: $\{(a_2), (a_3), (a_2, a_3), (a_3, a_2)\}$. Two of these are in the indexing trees: (a_3) with the weight v_a, and (a_2, a_3) with the weight $v_a + v_b$. Cases C_1 and C_3 are selected for association with Q. Since the association weight for C_1 is larger than that of C_3, C_1 is appended to Q before C_3. A threshold can be used based on a maximum number of association links or on a minimum association weight to limit the number of associated attributes.

4 Learning from user feedback

User feedback is crucial for systems to understand the difference between the user query and the user information needs [SM83]. The user selects the associated attributes based on its relevancy to his problem. The unselected associated attributes are considered irrelevant. Such user feedback is used by the CBR to update the association weights. The CBR estimates the average association weight for an associated attribute a from the weight of the attributes a_k $(k = 1, \ldots, t)$ from all the cases that participated in selecting attribute a as follows:

$$W(a_k; a) = \frac{W(C_i \rightarrow C_j, a_k; a) + \ldots + W(C_m \rightarrow C_n, a_k; a)}{N} \tag{5}$$

where: $W(a_k;a)$ = average association weight of a_k for association with attribute a,
$W(C_i \rightarrow C_j, a_k; a)$ = association weight of a_k in the association link from C_i to C_j that participated in selecting attribute a,
a_k = attribute used in selecting attribute a,
N= number of association links that use a_k in selecting the attribute a.

The association weights are updated after the selection of associated attributes. In order for the CBR to appropriately update the association weights, a confidence measure $\alpha(a,p)$, $0 \leq \alpha(a,p) \leq 1$, is used to represent the confidence in the current user feedback for association with attribute a, based on the p previous associations on a. One possible approach is to use Bayesian statistics to estimate $\alpha(a,p)$.

After the $n+1^{th}$ user feedback for an attribute a_k that participates in selecting associated attribute a, the association weight, $W_{n+1}(C_i \rightarrow C_j, a_k; a)$, is the sum of the association weight at the n^{th} user feedback and the weight contribution from the current feedback.

$$W_{n+1}(C_i \rightarrow C_j, a_k; a) = W_n(C_i \rightarrow C_j, a_k; a) + f(W_n(C_i \rightarrow C_j, a_k; a), \alpha(a,p), W(a_k; a)) \tag{6}$$

where: $W_n(C_i \rightarrow C_j, a_k; a)$ = association weight after n feedbacks for the link between C_i and C_j for a_k to associate with a,
$W_{n+1}(C_i \rightarrow C_j, a_k; a)$ = association weight after $n+1$ feedbacks for the link between C_i and C_j for a_k to associate with a,
f = a learning function.

Similarly, the association weights updating formula for irrelevant associated attributes is:

$$W_{n+1}(C_i \rightarrow C_j, a_k; a) = W_n(C_i \rightarrow C_j, a_k; a) - f(W_n(C_i \rightarrow C_j, a_k; a), \alpha(a,p), W(a_k; a)) \tag{7}$$

Equations *(6)* and *(7)* are adapted from IR formulas in the vector processing environment to modify the initial query from user feedback[SB90].

The choice of the learning function depends on the application, the number of cases in the Case Memory, the attribute distribution in the Case Memory, the desired accuracy, and the efficiency of associations. A good learning function should stabilize the Case Memory after a certain training period. We have tested the user feedback mechanism with the following linear learning function:

$$W_{n+1}(C_i \rightarrow C_j, a_k; a) = W_n(C_i \rightarrow C_j, a_k; a) + (1 - W_n(C_i \rightarrow C_j, a_k; a)) * W(a_k; a) \tag{8}$$

Our experience with this simple learning function reveals that it provides a fast learning from initial user feedbacks. The learning slows down as more relevant feedbacks are provided. However, this learning function is very sensitive to irrelevant feedbacks. Therefore, more learning functions need to be developed. We plan to experiment and classify a selected set of learning functions (e.g. polynomials) to study the system behavior depending on user feedback quality.

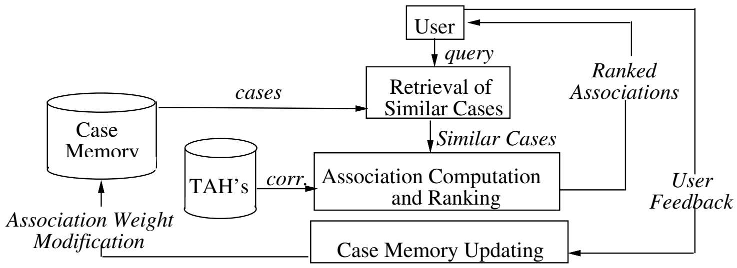

Fig. 4. *Associative Query Answering Algorithm*

.

To keep the size of the Case Memory manageable, certain links are discarded when their association weight decreases. One approach to avoid repetitive creation/deletion of links is to keep the weak association links in the Case Memory and change their status from active to passive. Active cases are used during the selection process, and the passive cases are are kept in the Case Memory as irrelevant cases. Updates of the Case Memory are applied to all cases.

5 Implementation

The association facility has been implemented on top of the cooperative database system CoBase[CMB93]. A predefined set of generic user models[MP92] is used. The association algorithm can be divided into three processes (Figure *4*):

– *Selection of similar cases*: query conditions are extracted as indexing features from the user query by its *select* and *where* clauses. The indexing schema is used to select association links included in the user query, while the set of cases similar to the user query selected from these association links.
– *Selection and ranking of associated attributes*: the similarity between the attribute values of the selected similar cases and the user query is computed from their correlation in pairwise TAH's for symbolic attributes[MC93] or from their relative difference for numeric attributes[CC94]. Associations are extracted from new attributes of the cases associated with the set of similar cases. Based on the user type, irrelevant associated attributes are filtered out. The selected cases for association are ranked by their usefulness*(4)*.
– *Updating of association weights*: the ranked set of associated attributes is presented to the user and his feedbacks are recorded. The association weights are then updated based on user feedbacks.

Currently, our Case Memory testbed consists of about 400 cases and runs on a SUN Sparc 10. Our experiments reveal that the time required to select and rank the four best attributes (a pre-specified association size threshold) for a given user query is less than 2 seconds, which is less than the query processing time.

6 Conclusion

In this paper we have developed a measure of "usefulness of association" to select relevant attributes for association. The association control is based on experience acquisition, user model, and query context. The evolution of association requirements are managed via CBR which adapts its knowledge from user feedbacks. The association facility infers the user goal of the current transaction from his previous transactions.
We have constructed a prototype on top of CoBase at UCLA for generating associate answers that use CBR for association control. Our preliminary experiences reveal that such association control is both feasible and scalable. Further experimentation is under way to study the behavior of the learning algorithm of using user feedback to modify association weights for selected applications.

References

[CC92] W.W. Chu and Q. Chen. Neighborhood and Associative Query Answering. *Intelligent Information Systems*, 1(3-4):355–382, 1992.

[CC94] W. Chu and K. Chiang. Abstraction of High Level Concepts from Numerical Values in Databases. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, Seattle, 1994.

[CD91] F. Cuppens and R. Demolombe. Extending Answer to Neighbor Entities in a Cooperative Answering Context. *Decision Support System*, pages 1–11, 1991.

[CHF92] M.A. Casanova, A.S. Hemerly, and A.L. Furtado. A Declarative Conceptual Modelling Language: Description and Example Applications. In *Proceedings of 4th International Conference on Advanced Information Systems Engineering*, pages 589–611, Manchester, UK, 1992.

[CMB93] W.W. Chu, M. Merzbacher, and L. Berkovich. The Design and Implementation of CoBase. In *Proceedings of SIGMOD 93*, pages 517–522, May 1993.

[FM93] G. Fouqué and S. Matwin. A Case-Based Approach to Software Reuse. *Journal of Intelligent Information Systems*, 2(2):165–197, 1993.

[MC93] M. Merzbacher and W.W. Chu. Pattern-based clustering for database attribute values. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, Washington, D.C., 1993.

[MP92] J.D. Moore and C.L. Paris. Exploiting User Feedback to Compensate for the Unreliability of User Models. *User Modeling and User-Adapted Interaction*, 2(4):287–330, 1992.

[RS89] C.K. Riesbeck and R.S. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Hillsdale, N.J., 1989.

[SB90] G. Salton and C. Buckley. Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science*, 41(4):288–297, June 1990.

[SM83] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. NcGraaw-Hill, New York, 1983.