A Cooperative Database System (CoBase) for Query Relaxation*

Wesley W. Chu, Hua Yang, and Gladys Chow
Computer Science Department
University of California at Los Angeles
E-mail: {*wwc, hua, gchow*}*@cs.ucla.edu*

Abstract

A new generation of information systems that integrates knowledge base technology with database systems is presented for providing cooperative (approximate and conceptual) query answering. Based on the database schema and application characteristics, data are organized into Type Abstraction Hierarchies (TAHs). The higher levels of the hierarchy provide a more abstract data representation than the lower levels. Generalization (moving up in the hierarchy) and specialization (moving down the hierarchy) are the key operations in deriving cooperative query answers for the user. Based on the context, the TAHs can be constructed automatically from databases. A relaxation manager is developed to provide control for query relaxations. A mediator architecture is developed to provide scalability and extensibility. GoBase allows users to specify queries graphically and incrementally on the map, greatly improving querying capabilities. CoBase has been demonstrated to answer imprecise queries for ARPI transportation planning.

Introduction

Consider asking a query of a human expert. If the posed query has no answer or the complete data for an answer is not available, you do not simply get a null response. The human expert attempts to understand the gist of your query, to suggest or answer related questions, to infer an answer from data that is accessible, or to give an approximate answer. The goal of cooperative database research is to create information systems with these characteristics (Gaasterland, Godfrey, & Minker 1992a). The key is the integration of a knowledge base which represents the data semantics.

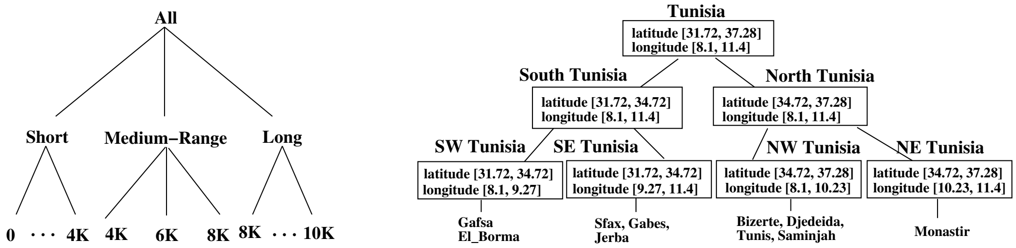
Cuppens and Demolombe (Cuppens & Demolombe 1988) provide cooperative answers by rewriting the query to add variables to the query vector, which carry relevant information to the user. The rewrite is not a generalization of the query, it is an extension of the query to provide more information. Motro (Motro 1988) proposes allowing the user to select the direction of relaxation and thus to indicate which relaxed answers may be of interest. Hemerly, *et al.* (Hemerly, Casanova, & Furtado 1994) uses a predefined user model and maintains a log of previous interactions to avoid misconstruction when providing additional information. All the above approaches are rule-based and difficult to scale up. To remedy this shortcoming, we will present a structured approach and its implementation to cooperative query answering for database systems.

Query Relaxation via Type Abstraction Hierarchies

Query relaxation enlarges the search range or relaxes an answer scope to include additional information. Enlarging and shrinking a query scope can be accomplished by viewing the queried objects at different conceptual levels, since an object representation has wider coverage at a higher level and inversely, more narrow coverage at a lower level. Although linking different level object representations can be made in terms of explicit rules (Cuppens & Demolombe 1988), such linking lacks a systematic organization to guide the query transformation process. To remedy this problem, we propose the notion of a type abstraction hierarchy (TAH) (Chu, Chen, & Lee ; Chu & Chen 1994) for providing an efficient and organized framework for cooperative query processing. A TAH represents objects at different levels of abstraction. For example, in Figure 1, the *Medium-Range* (*i.e.*, from 4,000 to 8,000 ft.) in the TAH for *runway-length* is a more abstract representation than a specific runway length in the same TAH (*e.g.*, 6,000 ft). Likewise, *SW Tunisia* is a more abstract representation than individual airports (*e.g.*, *Gafsa*). A higher-level and more abstract object representation corresponds to multiple lower-levels and more specialized object representations. Querying an abstractly represented object is equivalent to querying multiple specialized objects.

A query can be modified by relaxing the query conditions via such operations as *generalization* (moving up the TAH) and *specialization* (moving down the TAH), e.g., from 6,000ft to Medium-Range to [4,000ft,

*This work was supported by ARPA contract N00174-91-C-0107.



(a) *runway_length* (b) *airport location in Tunisia*

Figure 1: Type Abstraction Hierarchies

8,000ft]. In addition, queries may have *conceptual* conditions such as "runway-length = Medium-Range." This condition can be transformed into *specific* query conditions by specialization. Query modification may also be specified explicitly by the user through a set of *cooperative operators* such as 'similar-to', 'approximate', 'near-to', *etc.* This approach was also adopted in the cooperative deductive databases by Gaasterland, *et al.* (Gaasterland, Godfrey, & Minker 1992b) for providing query relaxations.

The notion of multi-level object representation is not captured by the conventional semantic network and object-oriented database approaches for the following reasons: Grouping objects into a class and grouping several classes into a super-class only provide a common "title" (type) for the involved objects without concern for the object instance values and without introducing abstract object representations. Grouping several objects together and identifying their aggregation as a single (complex) object does not provide abstract instance representations for its component objects. Therefore, an object-oriented database deals with information only at two general layers: the meta-layer and the instance layer. Since forming an object-oriented type hierarchy does not introduce new instance values, it is impossible to introduce an additional instance layer. In the Type Abstraction Hierarchy, instances of a super-type and a sub-type may have different representations and can be viewed at different instance layers. Such multiple layer knowledge representation is essential for cooperative query answering.

Knowledge for query relaxation can be expressed as a set of logical rules, but such a rule-based approach (Cuppens & Demolombe 1989; Hemerly, Casanova, & Furtado 1994) lacks a systematic organization to guide the query transformation process. TAHs provides a much simpler and more intuitive representation for query relaxation, and do not have the complexity of the inference that exists in the rule-based system. As a result, the TAH structure can easily support flexible relaxation control, which is important to improve relaxation accuracy and efficiency. Further, knowledge represented in a TAH is customized, thus changes in one TAH represent only a localized update and do not affect other TAHs, simplifying TAH maintenance.

We have developed tools to generate TAHs automatically from data sources (Merzbacher & Chu 1993; Chu & Chiang 1994), which enable our system to scale up and extend to large data sources. An intelligent dictionary/directory in the system lists the location and characteristics (*e.g.*, context and user type) of the TAHs.

Cooperative SQL (CoSQL)

The cooperative operations consist of the following four types: context free, context sensitive, control, and interactive. The context free and context sensitive cooperative operators can be used in conjunction with attribute values specified in the WHERE clause. The relaxation control operators can only be used on attributes specified in the WHERE clause, and the control operators have to be specified in the WITH clause after the WHERE clause. The interactive operators can be used alone as command inputs.

Context free operations.

- *Approximate operator*, $\wedge v$, relaxes the specified value v within the approximate range that is predefined by the user. For example, $\wedge$9am transforms into the interval (8am, 10am).
- *Between* (v_1, v_2) specifies the interval for an attribute. For example, time between (7am, $\wedge$9am) transforms into (7am, 10am). The transformed interval is pre-specified either by the user or the system.
- *Within* $(v_1, v_2, ..., v_n)$ specifies a set membership for an attribute value. For example, `airport within ('LAX','BURBANK')`

Context sensitive.

- *Near-to X* is used for specification of spatial nearness of object X. The "near-to" measure is context and user sensitive. "Nearness" can be specified by the user. For example, `near-to 'BIZERTE'` requests the list of cities located within a certain Euclidean distance (depending on the context) from the city 'BIZERTE'.

- *Similar-to X based-on* $((a_1\ w_1)(a_2\ w_2)...(a_n\ w_n))$ is used to specify a set of objects semantically similar to the target object X based on a set of attributes $(a_1, a_2, ..., a_n)$ specified by the user. Weights $(w_1, w_2, ..., w_n)$ may be assigned to each of the attributes to reflect the relative importance in considering the similarity measure. The set of similar objects can be ranked by the similarity. The similarity measures that computed from the nearness (e.g. weighted mean square error) of the pre-specified attributes to that of the target object. The set size is bound by a pre-specified nearness threshold.

Control Operators.

- *Relaxation-order* $(a_1, a_2, ..., a_n)$ specifies the order of the relaxation among the attributes $(a_1, a_2, ..., a_n)$ (*i.e.*, a_i precedes a_{i+1}).

  For example, `relaxation-order (runway_length, runway_width)` indicates that if no exact answer is found, then runway_length should be relaxed first. If still no answer is found, then relax the runway_width. If no relaxation-order control is specified, the system relaxes according to its default relaxation strategy.
- *Not-relaxable* $(a_1, a_2, ..., a_n)$ specifies the attributes $(a_1, a_2, ..., a_n)$ that should not be relaxed. For example, `not-relaxable location_name` indicates that the condition clause containing location_name must not be relaxed.
- *Preference-list* $(v_1, v_2, ..., v_n)$ specifies the preferred values $(v_1, v_2, ..., v_n)$ of a given attribute, where v_i is preferred over v_{i+1}. As a result, the given attribute is relaxed according to the order of preference that the user specifies in the preference list. For example, the attribute "food style," a user may prefer Italian food to Mexican food. If there are no such restaurants within the specified area, the query can be relaxed to include the foods similar to Italian food first, and then similar to Mexican food.
- *Unacceptable-list* $(v_1, v_2, ..., v_n)$ allows users to inform the system not to provide certain answers. This control can be accomplished by trimming parts of the TAH from searching. For example, "avoid airlines X and Y" tells the system that airlines X and Y should not be considered during relaxation. It not only provides more satisfactory answers to users, but also reduces search time.
- *Alternative-TAH (TAH-name)* allows users to use the TAHs of their choices. For example, a vacation traveler may want to find an airline based on its fare while a business traveler is more concerned with his schedule. To satisfy the different needs of the users, several TAHs of airlines can be generated, emphasizing different attributes (*e.g.*, price and nonstop flight).
- *Relaxation-level* (v) specifies the maximum allowable range of the relaxation on an attribute, *i.e.*, $[0, v]$.
- *Answer-set* (s) specifies the minimum number of answers required by the user. CoBase relaxes query conditions until enough number of approximate answers (*i.e.*, $\geq s$) are obtained.

User/System interaction operators.

- *Nearer, Further* provide users with the ability to control the "near-to" relaxation scope interactively. *Nearer* reduces the distance by a pre-specified percentage while *further* increases the distance by a pre-specified percentage.

Users can browse and edit relaxation control parameters to better suit their applications.

Example

Query: Find all airports in Tunisia similar to the Bizerte airport. Use the attributes runway_length_ft, runway_width_ft as criteria for similarity. Place more similarity emphasis on runway length than runway width; their corresponding weight assignments are 2 and 1, respectively. The following is the CoSQL version of the query:

```
SELECT aport_name
FROM aports, GEOLOC
WHERE aport_name SIMILAR-TO 'Bizerte'
      BASED-ON ((runway_length_ft 2.0)
                (runway_width_ft 1.0))
      AND country_state_name = 'TUNISIA'
      AND GEOLOC.geo_code = aports.geo_code
```

To select the set of the airport names that have the runway length and runway width similar to the ones for the airport in Bizerte, we shall first find all the airports in Tunisia, and therefore transform the query to:

```
SELECT aport_name
FROM aports, GEOLOC
WHERE country_state_name_ = 'TUNISIA'
      AND GEOLOC.geo_code = aports.geo_code
```

After retrieving all the airports in Tunisia, based on the runway length, runway width, and their corresponding weights, the similarity of these airports to "Bizerte" can be computed by the pre-specified *nearness* formula (*e.g.*, weighted mean squared error). The order in the similarity set is ranked according to the nearness measure, and the size of the similarity set is determined by the pre-specified nearness threshold.

A Scalable and Extensible Architecture

Figure 2 shows an overview of the CoBase System. Type abstraction hierarchies and relaxation ranges for the explicit operators are stored in a knowledge base (KB). There is a TAH directory storing the characteristics of all the TAHs in the system. When CoBase asks queries, it asks the underlying database systems. When an approximate answer is returned, the user can ask for an explanation of how the answer was derived or an annotated relaxation path (Minock & Chu 1999). A context-based semantic nearness will be provided to

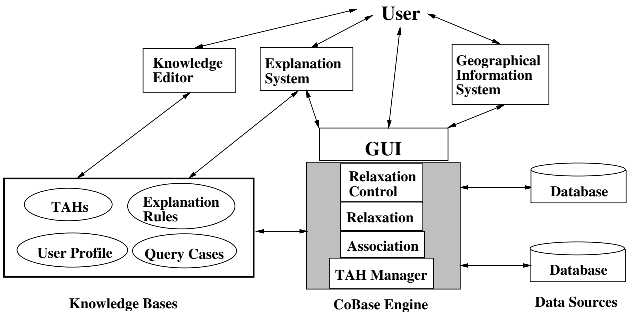


Figure 2: CoBase Functional Architecture.

rank the approximate answers (in order of nearness) against the specified query. A GUI displays the query, results, TAHs, and explanations of the relaxation processes. Based on user type and query context, associative information is derived from past query cases (Fouque, Chu, & Yau 1994). A user can construct TAHs from one or more attributes and modify the existing TAH in the KB.

Mediator Architecture

We use the concept of *mediation* (Wiederhold 1992) to decompose our cooperative answering system into reusable and interacting components. A *Mediator* is a software module that takes some input set of information, intelligently analyzes the information from a specific viewpoint, and produces a set of conclusions based on its analysis. Often, a Mediator needs additional information (knowledge and/or data) to perform its analysis. This information can be required as inputs to the Mediator's analysis, the Mediator can seek the assistance of other Mediators in fulfilling its information needs. This latter mode (Mediators assisting Mediators) is called *dynamic matching*. Specifically, a Mediator with an information need will report this need to the environment and dynamically match with the Mediators that can fulfill the need.

Figure 3 displays the various cooperative mediators: Relaxation, TAH , Association, Explanation, Data Source, and Directory. These mediators are connected selectively to meet applications' needs. An application that requires relaxation and explanation capabilities, for example, will entail a linking of Relaxation and Explanation mediators. Our mediator architecture allows incremental growth with application. When the demand for certain mediators increases, additional copies of the mediators can be added to reduce the loading, thus the system is scalable. For example, there are multiple copies of relaxation mediator and TAH mediator in Figure 3. Further, different types of mediators can be interconnect together and communicate with each other via a common communication protocol (*e.g.*, KQML) to perform a joint task. Thus, the architecture is extensible (Chu *et al.* 1996).

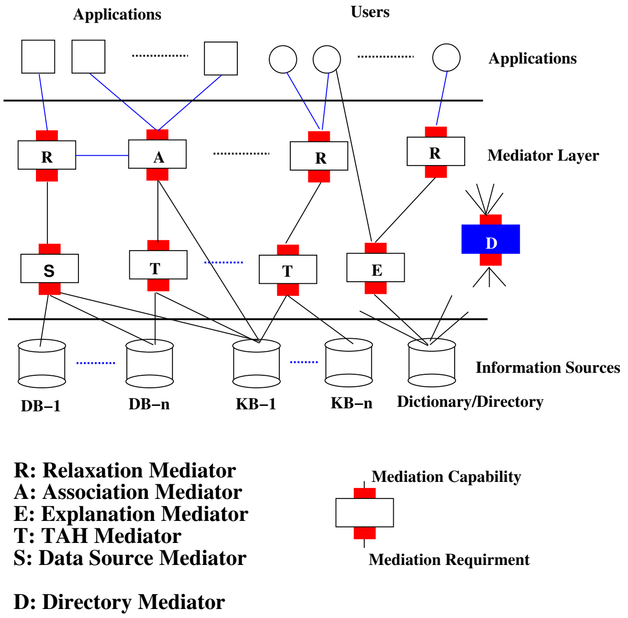


Figure 3: A Scalable and Extensible Cooperative Information System

Relaxation Mediator

Query relaxation is the process of understanding the semantic context, intent of a user query and modifying the query constraints with the guidance of the customized knowledge structure (TAH) into "near" values that provide "best-fit" answers. The flow of the relaxation process is depicted in Figure 4. When a CoSQL query is presented to the Relaxation Mediator, the system first go through a pre-processing phase. During the pre-processing, the system first relaxes any context free and/or context sensitive cooperative operators in the query. All relaxation control operations specified in the query will be processed. The information will be stored in the relaxation manager and ready to be used if the query requires relaxation. The modified SQL query is then presented to the underlying database system for execution. If no answers are returned, then the cooperative query system, under the direction of the Relaxation Manager, relaxes the queries by query modification. This is accomplished by traversing along the TAH node for performing generalization and specialization and rewriting the query to include a larger search scope. The relaxed query is then executed, and if there is no answer, we repeat the relaxation process until we obtain one or more approximate answers. If the system fails to produce an answer due to over-trimmed TAHs, the relaxation manager will deactivate certain relaxation rules to restore part of a trimmed TAH to broaden the search scope until answers are found. Finally, the answers are post-processed (*e.g.*, ranking, filtering, *etc*).

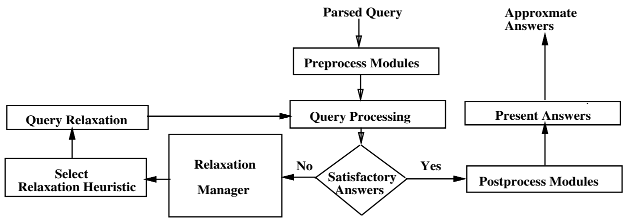


Figure 4: Flow chart for Processing CoBase queries.

Relaxation Control Relaxation without control may generate more approximations than the user can handle. The policy for relaxation control depends on many factors, including user profile, query context, and relaxation control operators as defined in Section . The Relaxation Manager combines those factors via certain policies (minimizing search time or nearness, for example) to restrict the search for approximate answers. We allow the input query to be annotated with control operators to help guide the mediator in query relaxation operations.

Let us consider the following example of using control operators to improve the relaxation process. Suppose a pilot is searching for an airport with an 8,000 feet runway in Bizerte but there is no airport in Bizerte that meets the specifications. There are many ways to relax the query in terms of location and runway length. If the pilot specifies the "relaxation-order" to relax the location attribute first, then the query modification generalizes the location 'Bizerte' to '*NW*_Tunisia' (as shown in Figure 1) and specializes the locations 'Bizerte', 'Djedeida', 'Tunis', and 'Saminjah', thus broadening the search scope of the original query. If, in addition, we know that the user is interested only in the airports in NW and SW Tunisia and does not wish to shorten the required runway length, the system can eliminate the search in East Tunisia and also avoid airports with short and medium runways, as shown in Figure 5. As a result, we can limit the query relaxation to a narrower scope by trimming the type abstraction hierarchies, thus improving both the system performance and the answer relevance.

Spatial Relaxation In geographical queries, spatial operators such as 'located,' 'within,' 'contain,' 'intersect,' 'union,' and 'difference' are used. When there are no exact answers for a geographical query, both its spatial and non-spatial conditions can be relaxed to obtain the approximate answers. CoBase operators also can be used for describing approximate spatial relationships. For instance, "an aircraft-carrier is *near to* seaport Sfax." *Approximate spatial operators*, such as 'near-to' and 'between' are developed for the approximate spatial relationships. Spatial approximation depends on *contexts* and *domains* (Mark & Frank 1989; Subramanian & Adam 1993). For example, a **hospital** *near to* LAX is different from an **airport** *near to* LAX.

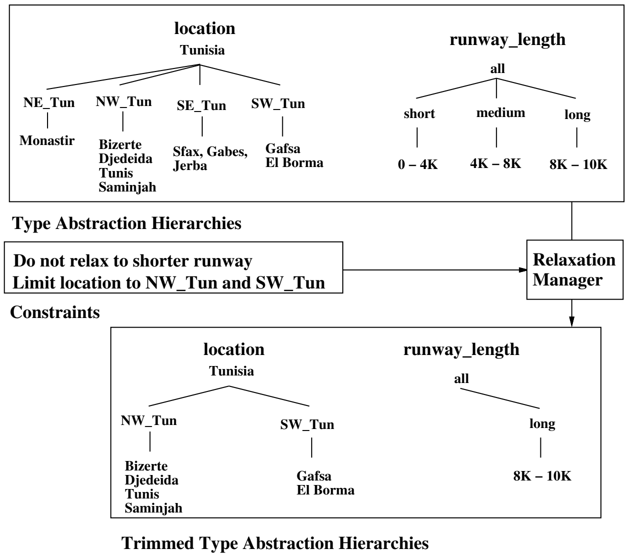


Figure 5: TAH trimming based on the constraints specified by the relaxation control operators.

Likewise, the *nearness* of a hospital in a **metropolitan** area is different from the one in a **rural** area. Thus spatial conditions should be relaxed differently in different circumstances. A common approach to this problem is the use of pre-specified ranges. This approach requires experts to provide such information for all possible situations, which is difficult to scale up to larger applications or to extend to different domains. Since TAHs are user and context sensitive, they can be used to provide context-sensitive approximation. We can generate TAHs based multi-dimensional spatial attributes (MTAHs).

Further, the MTAH for the object locations can be generated based on latitude and longitude from the distribution of the objects. The distance between nearby objects is context-sensitive: the denser the location distribution, the smaller the distance among the objects. In Figure 6, for example, the default neighborhood distance in Area 3 is smaller than the one in Area 1. Thus, when a set of airports are clustered based on their locations, the ones in the same cluster of the MTAH are much closer to each other than to those outside the cluster. Thus, they can be considered 'near-to' each other. We can apply the same approach to other approximate spatial operators, such as 'between' (*i.e.*, a cluster 'near-to' the center of two objects). MTAHs also can be used to provide context-sensitive query relaxation. For example, consider the query: "Find an airfield at the city Sousse." Since there is no airfield located exactly at Sousse, this query therefore can be relaxed to obtain approximate answers. First, we locate the city Sousse with latitude 35.83 and longitude 10.63. Using the MTAH in Figure 6, we find Sousse is covered by Area 4. Thus, the airport Monastir is

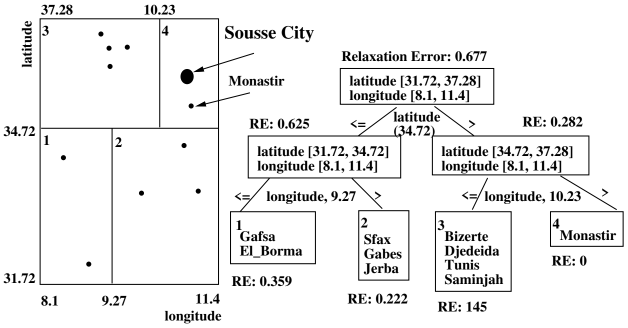


Figure 6: An MTAH for the airports in Tunisia and its corresponding two-dimensional space.

returned. Unfortunately, it is not an airfield. So the query is further relaxed to the neighboring cluster–the four airports in Area 3 are returned: Bizerte, Djedeida, Tunis, and Saminjah. Since only Djedeida and Saminjah are airfields, these two will be returned as the approximate answers.

MTAHs are automatically generated from databases by using our clustering method that minimizes relaxation error (Chu & Chiang 1994). They can be constructed for different contexts and user type. For example, it is critical to distinguish a friendly airport from an enemy airport. The use of a MTAH for friendly airports restricts the relaxation only within the set of friendly airports, even though some enemy airports are geographically nearby. This restriction significantly improves the accuracy and flexibility of spatial query answering. The integration of spatial and cooperative operators provides more expressiveness and context-sensitive answers. For example, the user is able to pose such queries as, "find the airports similar-to LAX and near-to City X." When there are no answers available, both 'near-to' and 'similar-to' can be relaxed based on the user's preference (*i.e.*, a set of attributes). To relax 'near-to', airports from neighboring clusters in the MTAH are returned. Similarly, the set of attributes that the 'similar-to' operator based on are relaxed by their respective TAHs.

TAH Mediator

The CoBase TAH Mediator (Figure 7) provides three conceptually separate, yet interlinked, functions to peer mediators. These are the TAH Directory, the TAH Management, and the TAH Editing facilities.

Usually, a system contains a large number of TAHs. In order to allow other mediators to determine which TAHs exist within the system and the characteristics of those TAHs, the TAH Mediator contains an intelligent directory. This directory serves to link the characteristics, context, and user type with the TAHs themselves. This directory is searchable by characteristics, user type, context, name, or any combination thereof, enabling peer mediators to locate TAHs even if they

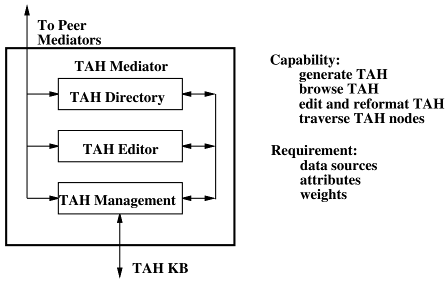


Figure 7: The TAH Mediator

have only a partial picture of the TAH they desire. Further, the TAH directory is capable of approximate matching, so if the exact TAH a client desires is unavailable in the system, the TAH Directory lists those TAHs which may be useful.

The TAH Management facility provides client mediators with TAH traversal functions (e.g. specialization and generalization), data extraction functions (for reading the information out of TAH nodes), and formatting functions (to eliminate the need for peer mediators to understand the varied internal formats of different types of TAHs). These capabilities present a common interface, so that peer mediators can traverse and extract data from a TAH without knowing about the particular type or structure of the TAHs they are using. Further, the TAH Mediator supports a TAH editor which allows users to edit TAHs to suit their specific needs. The TAH editor handles recalculation of all information contained within TAH nodes (e.g. the coverage and relaxation error) during the editing process and supports exportation and importation of entire TAHs if a peer mediator wishes to modify a TAH itself.

Implementation

Object-Oriented Implementation

We employ the object-oriented paradigm to implement CoBase. In this implementation, queries, database schema/interface, cooperative operators, and TAHs are represented as objects. Because of *polymorphism* and *encapsulation*, differences among various specific query languages, databases, cooperative functions, or types of TAHs are hidden behind abstract objects. Thus, query relaxation and other cooperative modules can be built upon these abstract objects with little knowledge about the implementation of the specific objects. For example, we have different objects for different databases including relational databases (*e.g.*, Oracle or SyBase object) and object-oriented databases (*e.g.*, Objectivity object). Each provides its own interface for database access. But CoBase modules only see a *database* object which has an abstract interface to access any database. This implementation enables these modules to work easily with different databases.

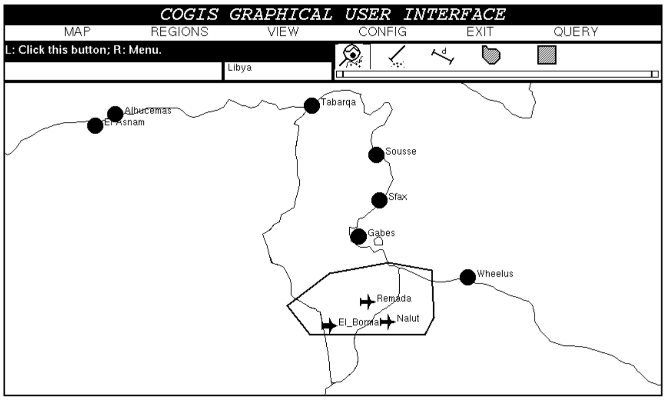


Figure 8: The GUI for the map server to display all the airports in a user-specified area.

Moreover, the object-oriented approach provides easy enhancements to CoBase. Adding a new cooperative function, for instance, simply entails defining a new cooperative operator object. The modification effects are localized and minimized through abstraction. Further, *inheritance* allows code sharing and reusing, thus reducing the development and maintenance effort. For example, functions common to all databases can be implemented in the abstract database object and inherited by all specific database objects. In this object-oriented paradigm, CoBase consists of multiple reusable modules. These modules are built on top of abstract objects. Thus, they can be easily ported to other systems by building *host-compliant* objects.

Graphical User Interfaces

The GUI for the relaxation was implemented in C++ and X/MOTIF. It enables users to browse the database, visualize the query relaxation process, and view the answers. Further, a map server was integrated into CoBase (Chu *et al.* 1994). It allows a user to pose queries based on geographic objects such as countries, states, cities, airports, rivers, *etc.* An important feature of these objects is that they can be *spatially* related via 'located,' 'within,' 'contain,' 'intersect,' 'union,' and 'difference.' These operators can be used in our system to specify spatial conditions in queries. For example, a query such as, "find all the airports in the region specified on the map" retrieves the airports in the user-specified region and displays the airport objects on the map, as shown in Figure 8. More importantly, it allows a user to ask complex queries by the simple 'point-and-click' method. The GUI for the explanation supports a hypertext-like browser for explanation. It allows users to interactively ask for more definition, elaboration, justification, or simply summarization (Minock & Chu 1999).

Performance Evaluation

We have measured the CoBase performance based on the execution of a set of queries on the CoBase testbed developed at UCLA for the ARPI transportation domain. The performance measure includes response time for query relaxation and explanation (Chu *et al.* 1996), and the quality of answers. The response time depends on the type of queries (*e.g.*, size of joins, number of joins) as well as the amount of relaxation and explanation required to produce an answer. The quality of the answer depends on the amount of relaxation involved. The user is able to specify the relaxation control to reduce the response time and also to specify the requirement of answer accuracy. Using a SUN SPARC-20 workstation, the typical cost for a relaxation operation is about 2 seconds. Each database retrieval takes about 10 seconds. And the explanation cost is about 2 seconds. Thus, the cost for relaxation and explanation is fairly small (Chu *et al.* 1996). The major cost is due to database retrieval which depends on the number of relaxations required before obtaining a satisfactory answer. For queries with multiple dependent attributes, the use of MTAH can decrease the number of relaxations, thus reducing the overall cost.

Technology Transfer of CoBase

CoBase stemmed from the transportation planning application for relaxing query conditions. CoBase has been linked with SIMS (Arens & Knoblock 1992) and LIM (McKay, Pastor, & Finin 1992) as a knowledge server for the planning system. SIMS performs query optimizations for distributed databases, and LIM provides high level language query input to the database. A Technical Integration Experiment (TIE) has been performed to demonstrate the feasibility of this integrated approach. CoBase technology is being implemented for the ARPI transportation application. Currently, we are applying the CoBase methodology to match medical image (X-ray, MRI) features (Chu, Cardenas, & Taira 1995), approximate matching of emitter signals in electronic warfare applications, and approximate query answering for Logistic Anchor Desk.

Conclusions

We have presented a structured approach that uses Type Abstraction Hierarchy for providing query modification. TAHs provide multi-level knowledge representation and can be generated automatically from data sources. Further, TAHs are user and context sensitive, and easy to customize and maintain. Thus, TAHs are scalable to large systems. We have also extended the SQL to CoSQL. This extended language provides cooperative operators that allow the user to explicitly specify relaxation operations and controls. These operators are very general and can be applied to different domains. Combination of these operators in a query can greatly improve the expressive power. Based on user profile and application contexts, the relaxation manager limits the search scope and also filters out the unsuitable answers. An explanation facility is included which summarizes the query modifications and relaxation process, and also provides the nearness of the approximate answer to the exact answer. Further, a cooperative geographic information system has been developed to provide information retrieval and analysis based on geographic objects (*e.g.*, countries, cities, airports) and spatial relations (*e.g.*, 'located,' 'within,' 'between,' 'near-to,' 'closest'), and to perform context-sensitive relaxation of geographic relations. Currently, we are transferring CoBase technologies for content-based approximate matching of medical images, and for approximate query answering in Logistic Anchor Desk for logistical planning.

Acknowledgment

The research and development of CoBase has been a team effort. We would like to acknowledge the past and present CoBase members, Chris Larson, Michael Minock, Guogen Zhang, Ladislav Berkovich, Jason Robins, Henrick Yau, Gilles Fouques, Matthew Merzbacher, Frank Meng, and Brad Perry for their contributions in design and implementation efforts.

References

Arens, Y., and Knoblock, C. 1992. Planning and reformulating queries for semantically–modelled multidatabase systems. In *Proceedings First International Conference on Information and Knowledge Management (CIKM)*, 92–101.

Chu, W. W., and Chen, Q. 1994. A structured approach for cooperative query answering. *IEEE Transactions on Knowledge and Data Engineering* 6(5):738–749.

Chu, W. W., and Chiang, K. 1994. Abstraction of high level concepts from numerical values in databases. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases.*

Chu, W.; Yang, H.; Chiang, K.; Ribeiro, B.; and Chow, G. 1994. GoGIS: a cooperative geographical information system. In *Proceedings of SPIE Conference on Knowledge-Based Artificial Intelligence Systems in Aerospace and Industry*, 53–63.

Chu, W. W.; Yang, H.; Chiang, K.; Minock, M.; Chow, G.; and Larson, C. 1996. CoBase: A scalable and extensible cooperative information system. To appear in Journal of Intelligent Information Systems.

Chu, W. W.; Cardenas, A.; and Taira, R. 1995. KMeD: A knowledge-based multimedia medical distributed database system. *Information Systems* 20(2):75–96.

Chu, W. W.; Chen, Q.; and Lee, R. C. Cooperative query answering via type abstraction hierarchy. In Deen, S., ed., *Cooperating Knowledge Based Systems.* Springer-Verlag Inc. 271.

Cuppens, F., and Demolombe, R. 1988. Cooperative answering: a methodology to provide intelligent access to databases. In *Proceedings of the 2th International Conference on Expert Database Systems.*

Cuppens, F., and Demolombe, R. 1989. How to recognize interesting topics to provide cooperative answering. *Information Systems* 14(2):163–173.

Fouque, G.; Chu, W.; and Yau, H. 1994. A case-based reasoning approach for associative query answering. In *Proceedings of the Eighth International Symposium on Methodologies for Intelligent Systems.*

Gaasterland, T.; Godfrey, P.; and Minker, J. 1992a. An overview of cooperative answering. *Journal of Intelligent Information Systems* 1:123–157.

Gaasterland, T.; Godfrey, P.; and Minker, J. 1992b. Relaxation as a platform of cooperative answering. *Journal of Intelligent Information Systems* 1:293–321.

Hemerly, A.; Casanova, M.; and Furtado, A. 1994. Exploiting user models to avoid misconstruals. In Demolombe, R., and Imielinski, T., eds., *Nonstandard Queries and Nonstandard Answers.* Oxford Science Publications. 73–98.

Mark, D., and Frank, A. 1989. Concepts of space and spatial language. In *Proceedings of the 9th International Symposium on Computer-Assisted Cartography*, 538–556.

McKay, D. P.; Pastor, J.; and Finin, T. W. 1992. View-Concepts: Knowledge-based access to databases. In *Proceedings First International Conference on Information and Knowledge Management (CIKM)*, 84–91.

Merzbacher, M., and Chu, W. W. 1993. Pattern-based clustering for database attribute values. In *Proceedings of AAAI Workshop on Knowledge Discovery.*

Minock, M., and Chu, W. 1999. Explanation for cooperative information systems. In *Proceedings of the Ninth International Symposium on Methodologies for Intelligent Systems.*

Motro, A. 1988. VAGUE: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems* 6(3):187–214.

Subramanian, R., and Adam, N. 1993. Ill-defined spatial operators in geographic databases: their nature and query processing strategies. In *Proceedings of ACM Workshop on Advances in Geographical Information System*, 88–93.

Wiederhold, G. 1992. Mediators in the architecture of future information systems. *IEEE Computer* 25(3):38–49.