PROCESSING UNCERTAINTY QUERIES IN DATABASE MANAGEMENT SYSTEMS

Kaushik Chakrabarti
Sharad Mehrotra
Michael Ortega
Kriengkrai Porkaew

Department of Computer Science
University of Illinois at Urbana-Champaign
1301 W. Springfield Ave.
Urbana, IL, 61801

Robert Winkler

U.S. Army Research Laboratory
2800 Power Mill Road
Adelphi, MD, 20783-1197

ABSTRACT

Emerging applications, including many military applications, require explicit mechanisms to represent and process uncertainty in queries and in the data stored in databases. Most current approaches to supporting uncertainty in queries layer a reasoning component on top of existing relational database management systems (DBMSs) which resolves the uncertainty in queries outside of the DBMS. While the layered approach is attractive due to its simplicity and since it requires minimal extensions to existing DBMS technology, it has some fundamental shortcomings which limit its usefulness to only simplistic applications. This paper proposes an extended relational model together with a suitably extended relational algebra as an alternative mechanism to incorporating uncertainty in queries. In contrast to the layered approach, the proposed model allows uncertainty to permeate database processing overcoming many of its limitations. The paper identifies challenging research issues that we are currently addressing in developing the proposed framework.

INTRODUCTION

Emerging applications pose an increasing demand on database management systems (DBMSs) to store and process imprecise information alongside precise and structured information traditionally stored in databases. Such applications require many important extensions to existing DBMS technologies including:

- mechanisms to represent uncertainty in stored data.
- extensions to DBMS query languages to support uncertain queries.
- techniques for interactive query formulation and refinement tools to enable users in helping the system interpret their information needs.
- mechanisms to process uncertain queries.

To motivate the required extensions, we draw on examples from two application domains that may benefit from incorporating uncertainty in databases: (1) multimedia databases in which multimedia information is represented and retrieved based on its content, and (2) spatio-temporal database that tracks and stores information about moving objects in a battlefield.

Uncertainty in Stored Data: uncertainty in data may arise due to a variety of reasons from multiple different sources. For example, in an image DBMS, an image is represented as a collection of visual features (e.g., color histogram representing color feature, keywords describing the image, representation of shapes of objects in the image, etc.). These features taken together form an imprecise representation of the image content. In a spatio-temporal battlefield database, the location of an object may be uncertain due to the limited precision of the sensor tracking the object and the temporal latency between successive readings.

Mechanisms to represent uncertainty in stored data have received much research attention both in specialized contexts (e.g., geospatial data [GG89], spatio-temporal data [WCD+98]) and in the general context [Mot95].

Uncertainty in Queries: Traditionally, DBMSs support only specific queries that return data matching the query precisely. Examples include a query q "*return all objects in a given spatial location at a specific*

*Prepared through collaborative participation in the Advanced and Interactive Displays Consortium sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement No. DAAL01-96-2-0003. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. Michael Ortega is supported in part by CONACYT grant 89061.

time t". If an object o was at the given location at the specified time, o qualifies as an answer . Otherwise, it does not. The result RES_q of q is the unordered *set* of all qualifying objects In contrast to specific queries, in uncertain queries users are also interested in data similar to or close to the target. The result is a *ranked set* of answers, where the ranking estimates the degree of match or closeness of the answer to the query. In general, the model used for ranking, referred to as the *retrieval model*, depends upon the specific application and/or user posing the query. The retrieval model determines the interpretation of the relevance value associated with the answer.

Similar to the descriptional uncertainty in stored data, uncertain queries may arise due to a variety of reasons. For example, the user may lack the capability or find it cumbersome to express his/her information need as a precise query over the stored data. For example, in a multimedia DBMS, a user may wish to visualize images that depict a "sunset" or images that are *similar to* a given set of input images. Even though the user has a precise information need, he/she is unable to specify the query using the low-level features (e.g., color, texture, textual annotations) that are used to model the images in the database. This may be due to a lack of semantically powerful enough features to fully capture the user's intent, or alternatively, the low-level features provide too cumbersome an interface to be used directly in expressing the query.

Uncertain queries may also arise in the spatio-temporal military application. For example, a user may be interested in the query *"Retrieve all objects to the north of and close to the object O"*. In this query, the predicates $NorthOf$ and $CloseTo$ are imprecise predicates i.e. it is not possible to state deterministically whether an object is north of O or close to O. For example, even an object very far from O may be perceived as close by the user if there are no objects closer to O. Therefore for such queries, it is not possible to deterministically separate the qualifying objects from the non-qualifying ones. However, it is possible to state whether an object is more northwards to O or closer to O compared to another one. Another, more complex example may be: *Retrieve all enemy tank units that will be closing or indirectly closing to friendly tanks or command centers over the next hour.* In the above query, *closing*, and *indirectly closing* are imprecise spatio-temporal predicates defined over lower-level properties of objects (e.g., distance between objects and their relative directions at various points of time). For example, *Closing* builds on *Distance* only, but *Indirectly Closing* relies on *Distance* and *Direction*, where *Distance*, and *Direction*, as discussed earlier, may themselves be imprecise concepts. In general, imprecise predicates may form a hierarchy.

Query Formulation and Refinement: DBMSs capable of handling uncertain queries must be combined with automated and/or human assisted reasoning mechanisms that enable the system to learn the interpretation of closeness or match between an object and the query thereby resolving the uncertainty. Without such a mechanism, the user will be forced to retry specific queries repeatedly with minor modifications until their information needs are satisfied. For example, in the image database application, an interactive *relevance feedback* mechanism may be used to map the high-level similarity query to a representation based on the lower-level features stored in the database [PM97]. Similar techniques can also be developed to learn the user's interpretation of the imprecise spatio-temporal predicates in the spatio-temporal database context. Whether query refinement is automatically achieved via a reasoning system, or is human assisted , the underlying DBMS must be capable of supporting the refinement process.

Processing Uncertain Queries: Given an uncertain query and a retrieval model, the DBMS must retrieve a set of matching objects along with their degree of match. Since the query writer is more interested in an object with higher relevance to the query than one with lower relevance, it is more logical to return the answers in the order of their relevance i.e. return a *ranked list* of objects instead of an unordered set of objects (as in a traditional DBMS) where the ranking is based on its relevance to the query. For example, in the query *"Retrieve all objects to the north of and close to the object O"*, the query writer expects the answers to be ranked based on their *northness* and *closeness* to O i.e. an object that is more northwards and closer to O should be ranked before an object less northwards and farther from O.) Note that a ranked list is a generalization of an unordered set since a set is a ranked list with elements in the set having absolute relevance (100% relevance) and elements not in the set having no relevance (0% relevance).

Most existing approaches to support uncertain queries in DBMSs layer a wrapper built as an application on top of existing DBMS which estimates the relevance of the answer to the user's information need. Such an approach is facilitated by the emerging object relational technology which allows for application writers to define new data types and intermix user-defined functions in database queries. In such an implementation, the user-

defined functions implement the imprecise operations (e.g., match between images, precision of moving objects) and associate a measure of precision with each object retrieved. The set of retrieved objects are then ranked outside of the database system by the reasoning component based on the application-specific criterion. Done in this way, imprecision in the processing does not permeate database query processing. Existing database models and query languages, based on processing crisp precise information, suffice for this task.

Although simple to implement, there are three major shortcomings with the layered approach which limits its usefulness to only simplistic applications. First, such an approach needs to map the imprecise queries (i.e. queries with imprecise predicates e.g, CloseTo, NorthOf) to precise queries (e.g., spatial range queries) that can be executed on an existing DBMS. It may not be always feasible to express imprecise queries in a precise query language. For example, consider an image retrieval system where a common query might be: "retrieve the 10 images that are most similar to the given image". Since the image similarity depends on so many features and each feature is usually a very high dimensional vector with an arbitrary distance metric [ORC+97], it is not feasible to express the above query as a range query. Second, since reasoning is performed outside the database query processing, the layered approach may suffer from significant performance overhead since the DBMS may generate a very large number of answers which are then post-processed (ranked) by the wrapper. Also, since the user might only be interested in the best few matches, a large portion of the processing performed by the DBMS to answer the query may be wasted. This could have been avoided if the imprecise reasoning were intermixed more tightly with the database processing. Finally, as mentioned previously, an important aspect of a system handling uncertain queries is the ability to support query refinement in which the system is able to receive user feedback on the answers returned and to refine the query based on the feedback. The refined query is then executed again and a refined set of answers is returned. The process is iterated until the returned answer set converges to the desired answer set. In the layered approach, since the DBMS generates the entire answer set at each iteration, incorporating query refinement effectively and efficiently is not easy.

Proposed Approach: Motivated by the above, in this paper we propose an extended relational database model in which relations, instead of being interpreted as an unordered set of tuples, are interpreted as ranked lists based on a ranking expression (RE). Associated with each tuple in the relation is a belief value which determines the ranking of the tuple in the relation. The belief value represents the degree to which the tuple satisfies the RE associated with the relation. Relational operators (e.g., projection, selection, join, etc.) are appropriately extended to take ranked lists as inputs and generate a ranked list whose RE depends upon the REs associated with the input relations as well as the operation performed.

Done in this fashion, uncertainty permeates the database query processing as opposed to only the modeling. Imprecision is taken advantage of to compute answers in decreasing order of relevance to the user. In addition, the list of answers can be stopped before it is complete if the user deems all relevant objects have been retrieved already, thus saving computation time. In [ORC+97], we have shown how a subset of the required operations can be efficiently computed over ranked lists.

The remainder of the paper is developed as follows. In the following section, we briefly survey the proposed extension to the relational model to support uncertain queries and discuss how query refinement can be incorporated in the model. The following section discusses numerous research issues which we are currently addressing in developing the proposed framework.

EXTENDED RELATIONAL MODEL

To provide a framework to support uncertain queries, we propose an extended relational model and query language in which relations, instead of being interpreted as unordered sets of tuples, are instead interpreted as *ranked lists* based on a *ranking expression* (RE). Associated with each tuple in the relation is a belief value which determines the ranking of the tuple in the relation. The belief value represents the degree to which the tuple satisfies the RE associated with the relation. Relational operators (e.g., projection, selection, join, etc.) are appropriately extended to take ranked lists as input and to generate a ranked list whose RE depends upon the REs associated with the input relations as well as the operation performed. For example, in the query Q *"Retrieve all objects to the north of and close to the object O"*, the result set is ranked based on the predicates $NorthOf(O)$ and $CloseTo(O)$, its RE being the boolean expression $NorthOf(O) \wedge CloseTo(O)$.

Extending the semantics of relations to ranked lists provides a powerful framework to incorporate uncertain queries in a DBMSs. First, the model does not make any assumptions about the mechanism used to resolve the

uncertain predicates (that is, to rank objects based on the individual predicates). Any application-dependent retrieval model can be used to determine the interpretation of the relevance values. Furthermore, different imprecise predicates in the same query can be resolved using different models. As long as the mechanism results in a ranked list of answers, it can be incorporated into the framework. Second, the model only specifies that each RE have the syntax of a boolean expression. The model does not specify how to interpret these expressions in order to generate belief values based on which the answers can be ranked. For example, the RE can be interpreted as an expression in fuzzy logic. Similarly, a probabilistic interpretation of REs can also be used. This allows for a powerful extensible framework for supporting uncertain queries.

In the following subsection, we first present a relational algebra that defines the semantics of the relational operators on ranked lists. Subsequently, we discuss how query refinement techniques can be integrated into the extended relational model. Finally, we propose an extension to SQL to support a "RANK BY" clause in select-from-where queries to be able to express uncertain queries. Since this extension is syntactic rather than semantic, it is equally applicable to the layered approach for uncertain query processing.

Relational Algebra for Ranked Lists. Traditional relational algebra defines the semantics of the relational operators on sets of tuples. For example, relational algebra defines the meaning of the join operation between two relations (i.e. sets). We need to define the semantics of relational operators when the operands are ranked lists instead of sets. Since relations or sets in traditional relational algebra can be considered as a special case of ranked lists, such an algebra is a generalization of traditional relational algebra i.e. it provides the same semantics as traditional relational algebra when all attributes and queries are precise. In this section, we present a relational algebra for ranked lists.

We define a boolean expression associated with any ranked list L. We refer to it as the *rank expression (RE)* of L. A ranked list L with RE R is denoted by L^R. The RE defines the semantics of relational operators over ranked lists. Intuitively, the RE R of a ranked list L is the boolean expression of the predicates based on which L is ranked. For example, for the query Q, the result set is ranked based on the predicates $NorthOf(O)$ and $CloseTo(O)$, its RE being the boolean expression $NorthOf(O) \wedge CloseTo(O)$. The interpretation of the boolean operators in the RE depends on the inference model used. For example, the meaning of $\wedge$ in the above RE depends on the inference model. The resulting ranked list L_{res} is generated from the individual ranked lists L_1 and L_2 ranked individually by the predicates $NorthOf(O)$ and $CloseTo(O)$ respectively depending on the operator connecting them in the RE (in this case, $\wedge$) and its interpretation by the inference model. For example, in the fuzzy model, the relevance of any object in L_{res} is the minimum of the relevance of the object in L_1 and L_2 whereas in the probabilistic model, assuming the presence of the object in L_1 and L_2 as independent events, the relevance of the object in L_{res} is the product of its relevances in L_1 and L_2. Since the ranking of L_{res} is based on the relevances, the ranking of L_{res} depends on the inference model. The RE thus defines the semantics of the relational operators which is interpreted by the retrieval model.

To develop the relational algebra over ranked lists, we define the REs of ranked lists produced by each relational operator. Note that a crisp predicate can never appear in an RE since a crisp predicate is only used to filter data items but not to rank them. Due to space limitations, we present the REs corresponding to only the select operation in this paper.

Selection Operation. Selections are based on predicates. A predicate may either be crisp i.e. involving a crisp attribute (like height > 500) or non-crisp (like position_of_object *NorthOf* position_of_given_object). If p is a predicate, a_p denotes the attribute of the predicate p. For conciseness of representation, we represent the RE as a boolean expression of the attributes associated with the predicates instead of the predicates themselves. If p and q are two predicates, the following are also valid selection predicates:

- Disjunction of p and q: $p \vee q$
- Conjunction of p and q: $p \wedge q$
- Negation of p: $\neg p$

When predicates are not crisp, not all possible combinations generated by disjunctions, conjunctions and negations make intuitive sense. For example, selections are usually never based on unguarded negations $\sigma_{\neg q}$ or negation in the disjunction $\sigma_{p \vee \neg q}$, where q is a non-crisp predicate.

- Simple crisp predicate: Crisp predicates do not affect ranking. If p is a crisp predicate, $\sigma_p(L^R) \rightarrow L^R$.
- Simple non-crisp predicate: A non-crisp predicate p affects the ranking. If the list was ranked based on

R before the selection, the list will be ranked based on both R and p after the selection. So $\sigma_p(L^R) \rightarrow L^{\{R \wedge a_p\}}$.

- Conjunction of crisp predicates: Since conjunctions of crisp predicates are also crisp predicates and crisp predicates do not affect the ranking, $\sigma_{p \wedge q}(L^R) \rightarrow L^R$ where p and q are crisp predicates. Similarly, disjunction and negation of crisp predicates do not change the ranking expression.
- Conjunction of crisp and non-crisp predicates: Only the non-crisp attributes affect the ranking. Since the predicate is a conjunction, the items in the retrieved set are ranked based on its original ranking criterion as well as q where p and q are the crisp and non-crisp predicates respectively. So $\sigma_{p \wedge q}(L^R) \rightarrow L^{\{R \wedge a_q\}}$.
- Disjunction of crisp and non-crisp predicates: An item can appear in the answer set because it satisfies either the crisp predicate or the non-crisp predicate or both. If p and q are the crisp and non-crisp predicates respectively, an item in the retrieved set is ranked based on either the original ranking criterion only (if the item satisfies p) or on both the original criterion and q (if the item does not satisfy p). So $\sigma_{p \vee q}(L^R) \rightarrow L^{\{R \vee (R \wedge a_q)\}}$ *.
- Conjunction of crisp and non-crisp predicates with negation: If the crisp predicate p appears in negative form, the ranking is based on the original ranking criterion and the non-crisp predicate q. So $\sigma_{\neg p \wedge q}(L^R) \rightarrow L^{\{R \wedge a_q\}}$. If the non-crisp attribute q appears in negative form, the ranking is based on the original criterion and the inverse of the ranking produced by predicate q. So $\sigma_{p \wedge \neg q}(L^R) \rightarrow L^{\{R \wedge \neg a_q\}}$.
- Conjunction of non-crisp predicates: In this case, the final ranking is based on the original ranking and both non-crisp predicates. If p and q are the predicates, $\sigma_{p \wedge q}(L^R) \rightarrow L^{\{R \wedge a_p \wedge a_q\}}$.
- Conjunction of non-crisp predicates with negation: If p and q are the predicates and q appears with a negation, the final ranking is based on the original ranking, the ranking produced by p and the inverse of the ranking produced by q. So $\sigma_{p \wedge \neg q}(L^R) \rightarrow L^{\{R \wedge a_p \wedge \neg a_q\}}$.
- Disjunction of non-crisp predicates: An item can appear in the answer set because it satisfies either of the two predicates or both. If p and q are the two predicates, an item in the retrieved set is ranked based on either the original ranking criterion and ranking produced by p (if the item satisfies p more than it satisfies q) or the original ranking criterion and the ranking produced by q (if the item satisfies q more than it satisfies p). So $\sigma_{p \vee q}(L^R) \rightarrow L^{\{(R \wedge a_p) \vee (R \wedge a_q)\}}$.

Incorporating Query Refinement. In DBMSs supporting imprecise queries, it may not be possible for a system to retrieve the desired set of answers in the first attempt. The reason is that an imprecise predicate may be a combination of several sub predicates, the relative emphasis among which may vary from user to user. Different degrees of emphasis on different sub predicates produces different answers. Query refinement is a technique to learn the relative emphasis among the various sub predicates from user feedback and to appropriately reformulate the query to obtain a refined set of answers that better satisfies the user's information need.

Refinement of imprecise queries has primarily been studied in specialized contexts. For example, in information retrieval literature, where textual documents are stored based on keywords they contain, extensive work has been done on using relevance feedback mechanism to map a user's information need to a keyword based representation [SFV83]. Further, we have adapted the relevance feedback mechanism to map high-level similarity queries to feature-based representations in image databases [RHMO97b, RHMO97a]. In each of these mechanisms, importance of a feature/component is modeled using weights where a higher weight corresponds to higher importance. Relevance of an object to the query is estimated as a weighed combination of similarity values based on low-level features. Query refinement is achieved by adjusting the weights based on user feedback. Recently, we have generalized the relevance feedback mechanism to arbitrary hierarchies, where a feature may itself be composed of lower level features [PM97]. Weighted summation of similarity based on each component features is used to estimate the similarity based on a higher level feature. A single process of feedback is used to update the weights at each level of the hierarchy.

The approach proposed in [PM97], can be generalized in a straightforward way for query refinement in relational algebra queries over ranked lists. Consider a query tree for a given relational algebra expression. Associated with each internal node is a vector of weights, one for each child of the node, signifying the importance of the child node in determining the relevance of a qualifying answer. Weights can be incorporated in the retrieval model to evaluate the ranking of an answer based on the RE associated with a node. Weight update process can then be used to refine the uncertain query based on user feedback.

*Notice that we do not simplify the expression $\{R \vee (R \wedge a_q)\}$ to R, even though in pure boolean logic they are equivalent, since, depending on the inference model used, the two expressions might not be equivalent

Expressing "RANK BY" queries in SQL. One alternative to express "rank by" queries in SQL is to keep it implicit i.e. query writers can freely intermix crisp predicates with imprecise ones and the RE corresponding to a query is implicitly generated by the DBMS from the query statement. This approach does not require any extension of the SQL syntax as imprecise predicates can be stated in the WHERE clause along with precise predicates. The other approach is to express "rank by" queries explicitly by extending the SQL syntax by a RANK BY clause. The query writer explicitly specifies the ranking criteria in the RANK BY clause while all the crisp predicates are stated in the WHERE clause. While the implicit approach is more general as it does not constrain how "rank by" predicates are intermixed with the crisp ones, it also represents a more severe departure from SQL semantics. Furthermore, it complicates the interpretation and implementation of conditional operators (i.e., **AND**, **OR**, and **NOT**), aggregation operations and nested queries which form the basic building blocks of SQL queries. For this reason, we develop an explicit approach to expressing uncertain queries in SQL.

Consider the query Q': *"retrieve all objects to the north of and close to a given object O and located at the same height as O_1.* In the explicit approach, the query can be expressed as:

SELECT	*
FROM	objects AS O
WHERE	O.height = O_1.height
RANKED BY	O NorthOf O_1 AND O CloseTo O_1

The RANK-BY clause should not be confused for ORDER-BY. The primary difference is that while the ORDER BY clause is *procedural* i.e., it specifies the attributes or functions defined on the attributes according to which the set of answers produced by the query is to be sorted as the final step before returning the answers, the RANK BY clause is *declarative*, just like the WHERE clause. Unlike the ORDER BY clause, the RANK BY clause is not used for post-processing the results returned by the query. Instead, the evaluation of the predicates in the RANK BY clause are intermixed with the predicates in the WHERE clause so as to optimize the query evaluation plan. Another difference is that while the parameters in the ORDER-BY clause is a list of attributes (or functions applied to attributes), parameter to the RANK-BY clause can be any boolean predicate which is interpreted based on the retrieval model. Finally, while the ORDER BY clause is either not permitted (or is ignored) in nested sub queries in SQL, the RANK BY clause can appear in nested sub queries just like the WHERE clause. The RANK BY clause can also appear in correlated queries i.e. a predicate in the RANK BY clause of the nested query can reference some attribute of a relation declared in the outer query.

RESEARCH ISSUES

A number of research issued need to addressed in developing the model proposed in this paper as a framework to support uncertain queries. These issues can be broadly classified into two categories which correspond to the *semantics* of the proposed framework for uncertain queries and to its *efficient implementation.* Relational operators over ranked lists discussed in the paper provide the basis for defining the semantics of the framework. The work needs to be extended to account for imprecise aggregations and also needs to be reconciled to handle imprecision in data. Several models for representing uncertainty in data both in specialized context (e.g., spatio-temporal data) as well as in general have been proposed in the past [Mot95]. We need to integrate these techniques for handling imprecise information within the framework for uncertain queries. A further research topic is to study specialized weight update approaches to query refinement for queries supporting imprecise spatio-temporal predicates. These approaches can then be integrated into the generalized relevance feedback mechanism proposed in the paper. Another important direction of research is to identify necessary and sufficient conditions that weight update algorithms must satisfy for convergence of relevance feedback process.

One of the primary advantages of the framework for uncertain processing proposed in this paper over the layered approach is that it is much more amenable to optimization and improved performance. However, before we can validate the claim, research must be done on extending the indexing and query optimization technology to support ranked lists.

The requirements of index management in the proposed system is different from that in traditional DBMSs. The "rank by" predicates are defined over usually complex attributes (non-1NF) i.e. attributes that cannot be indexed by a simple B-tree (e.g., high dimensional feature vectors as in multimedia applications). Data structures that can index such complex attributes needs to developed. Also, different attributes have different notions of similarity (or distance). Since the aim of an index structure is to cluster together the objects with similar values of the attribute being indexed, new index structures need to developed for every different similarity measure used. A template-based indexing mechanism that can be easily configured for arbitrary similarity

functions needs to be developed. Furthermore, the index structure must support *"ranked search"* or *"k nearest neighbor search"*. Thus, a selection based on a "rank by" predicate can be executed on the index structure to generate a ranked list. Finally, before such indexing mechansims are integrated as access methods into a DBMS, efficient techniques to provide transactional access to the data via the index structure need to be developed [**?**].

Another important research issue is development of techniques to efficiently evaluate "rank by" queries in the DBMS. Depending on how REs are interpreted i.e. retrieval model used, efficient algorithms to compute relational operators which given rank ordered input streams produce a rank ordered output stream needs to be developed. These algorithms provide the building blocks for evaluating "rank by" queries. Another issue is that of query optimization. Query optimization techniques in traditional DBMSs may not be applicable in a system supported ranked search. For example, pushing selections may not always generate more efficient plans. The cost of performing such selections may be quite high depending on the presence or absence of appropriate indices. Deferring evaluation of expensive predicates to later point may improve overall query processing. Morever, since the user is only interested only in the top few answers, efficient algorithms to return the desired number of top answers with minimum amount of wasted work i.e. accessing minimum number of irrelevant objects needs to be developed.

REFERENCES

[GG89] Michael Goodchild and Sucharita Gopal. *The Accuracy of Spatial Databases*. Taylor and Francis, New York, 1989.

[Mot95] Amihai Motro. *Management of Uncertainty in Database Systems*, chapter 22. ACM Press, 1995.

[ORC+97] Michael Ortega, Yong Rui, Kaushik Chakrabarti, Alex Warshavsky, Sharad Mehrotra, and Thomas S. Huang. Supporting ranked boolean similarity queries in mars. Technical Report Sumbitted to TKDE Special issue on Data and Knowledge Management in Multimedia Systems, also available as tech report TR-MARS-97-12, University of Illinois at Urbana-Champaign, Aug. 1997.

*The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government.

[PM97] K. Porkaew and Sharad Mehrotra. A multimedia retrieval model with relevance feedback. Research Note TR-MARS-97-14, Dec. 1997.

[RHMO97a] Yong Rui, Thomas S. Huang, Sharad Mehrotra, and Michael Ortega. Automatic matching tool selection via relevance feedback in mars. *to appear in the 2nd Int. Conf. on Visual Information Systems*, 1997.

[RHMO97b] Yong Rui, Thomas S. Huang, Sharad Mehrotra, and Michael Ortega. A relevance feedback architecture in content-based multimedia information retrieval systems. In *Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries*. IEEE, 1997.

[SFV83] G. Salton, Edward Fox, and E. Voorhees. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, November 1983.

[WCD+98] Ouri Wolfson, Sam Chamberlain, Son Dao, Liqin Jiang, and Gisela Mendez. Cost and imprecision in modeling the position of moving objects. In *Proceedings of the Fourtheenth Itnernational Conference on Data Engineering*, February 1998.