Producing Approximate Answers to Database Queries *

Susan V. Vrbsky and Jane W. S. Liu

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

We have designed and implemented a query processor, called APPROXIMATE, that makes approximate answers available if part of the database is unavailable or if there is not enough time to produce an exact answer. The accuracy of the approximate answers produced improves monotonically with the amount of data retrieved to produce the result. The exact answer is produced if all of the needed data are available and query processing is allowed to continue until completion. The monotone query processing algorithm of APPROXIMATE works within the standard relational algebra framework and can be implemented on a relational database system with little change to the relational architecture. We describe here the approximation semantics of APPROXIMATE that serves as the basis for meaningful approximations of both set-valued and single-valued queries. We show how APPROXIMATE is implemented to make effective use of semantic information, provided by an object-oriented view of the database, and describe the additional overhead required by APPROXIMATE.

1. Introduction

Many factors can make it impossible for a database to produce an exact answer to a query. A network partition or a host failure can cause some needed data to become inaccessible. There may not be enough time to acquire all the locks and retrieve all the data needed to answer a query. For many applications it may be better for a database to produce an approximate answer when it is not possible to produce an exact answer [1,2]. We have designed and implemented an approximate query processor, called APPROXIMATE, that can produce approximate answers to database queries for these applications [3].

The problem of how to define and improve approximate answers has been addressed by many researchers [4-7]. Buneman, Davidson, and Watters [4] introduced the approximation semantics where approximations of a set-valued answer are defined in terms of the subsets and supersets of the exact answer. The approximations are monotonically improving as the subsets and supersets approach the exact answer, and an approximate answer produced earlier is never contradicted by an approximate

* This work was partially supported by NASA Contract No. NASA NAG 1-613, ONR Contract No. NVY N00014-92-J-1146, and AFOSR Contract No. 1-5-26932.

answer produced later. Ozsoyoglu et al. have addressed the problem of producing approximate answers to queries with time constraints [5]. Monotonically improving approximations of the exact answer, which are subsets of increasing sizes, are produced. Fragments of the data are processed at a time to produce these subsets. Motro [6] developed a system, called VAGUE, that provides approximate answers to vague queries in which the query qualifications are imprecise. The response closest to the query qualifications according to a distance function is the answer to a vague query. An intensional answer [7] provides an approximation that is derived without accessing the physical data in the database. General characteristics about the data objects in the exact answer are provided instead of the data objects themselves.

APPROXIMATE implements monotone query processing; an initial approximation of the exact answer is produced when query processing begins based on the information it maintains for this purpose. The approximate answer produced improves as more data are retrieved to answer the query according to the partial-order relation defined by the approximate relational model [8]. APPROXIMATE returns the exact answer if all of the needed data are available and if there is enough time to continue with the processing. The latest, best available approximate answer is returned if the user demands an answer before query processing is completed. In contrast, during traditional query processing, if there is not enough time or not all of the data are available, no answer is provided. APPROXIMATE assumes the information contained in the database and the query expression are precise. APPROXIMATE can be one of several query processors available to the system. It is implemented in a relational database system and requires little or no change to the underlying relational architecture.

This paper describes the semantics of approximation supported by APPROXIMATE. After approximations of query answers are defined in Sections 2 and 3, Section 4 describes how approximate answers are produced by APPROXIMATE. Section 5 discusses future directions of this work.

2. Approximate Relational Model

There is a natural way to approximate answers of set-valued queries; an exact answer E to such a query is a set of data objects. All the data objects in a subset of E certainly belong to E, and a data object in a superset of E is possibly also in E. Therefore, a meaningful approximation of any exact answer E can be defined in terms of a subset and a superset of E. Specifically, an approximation A of an exact answer E is the union of two sets of data objects: a *certain set* C, where $C \subseteq E$, and a *possible set* P, where $(C \cup P) \supseteq E$. C is the set of data objects certainly in E; it is produced from the stored data processed thus far. P is the set of data objects that may be in E. Data objects in P are produced based on the meta data, information about the stored data, maintained by the query processor. This approximation is denoted by the 2-tuple $A = (C, P)$.

Any exact answer E has many approximations. Given a set of approximations of E, a partial order relation $\geq$ for comparing them can be defined over the set as follows. One approximation $A_i = (C_i, P_i)$ is better than or equal to another $A_j = (C_j, P_j)$, denoted as $A_i \geq A_j$, if $P_i \subseteq P_j$ and $C_i \supseteq C_j$. This partially ordered set of all approximations of E is a lattice. In the lattice, $A_0 = (\emptyset, \upsilon)$ is the least element and the worst possible approximation of E, where υ is the cartesian product of all the domains in the schema of E. υ is the set of all possible data objects which could be in E. It can be generated without reading any data. The greatest element of the lattice is the best possible approximation and is E itself, which is represented by $(E, \emptyset)$.

In the traditional relational model, an exact answer E is a standard relation. An approximation of a standard relation is called an *approximate relation.* As an example, we consider an AIRPORTS database that resides on-board an airplane. AIRPORTS contains the relation RUNWAYS (id, airport, length, obstructions). A pilot queries the database to locate all the runways with an easterly direction, with ids from 5 to 13, at airports O'Hare (ORD) and Midway (MDW) in Chicago. The exact answer to this query is the relation E shown in Figure 1(a). The relation A_1 shown in Figure 1(b) contains tuples on all runways at ORD and MDW. It gives all the tuples that are possibly in E and hence, is an approximation of E. The approximate relation A_2 shown in Figure 1(c) is another approximation of E. The first three tuples are certainly in E. They form a subset of E. The last three tuples are possibly in E. A_2 is a superset of E. It is a subset of A_1 and is a better approximation of E than A_1. The approximate relation A_3 in Figure 1(d) is another approximation of E. It is a subset of A_1 and is better than A_1, but is not comparable to A_2.

3. Approximations of Single-Valued Answers

The semantics of approximation defined by the approximate relational model can also serve as a basis for a meaningful semantics of approximation of single-valued queries. An exact answer to a single-valued query can be a single object retrieved from the database, the value of an aggregate function (such as "count"), or a value (such as "yes" or "no").

We consider the query "What is the color of car No. 20?" that exemplifies a special case of set-valued queries whose exact answer is a set of cardinality 1. The exact answer E is "maroon". Since any proper subset of E is the null set $\varnothing$, an approximate answer of E is, therefore, simply $(\varnothing, P_i)$ where

id	Airport	Length
6	MDW	7500
7	MDW	8000
9	ORD	11000
10	ORD	8000
13	ORD	10000

Figure 1(a). E: All easterly runways at ORD and MDW

id	Airport	Length
1	MDW	6000
6	MDW	7500
7	MDW	8000
9	ORD	11000
10	ORD	8000
13	ORD	10000
27	ORD	9500

Figure 1(b). A_1: All runways at ORD and MDW

id	Airport	Length
6	MDW	7500
7	MDW	8000
10	ORD	8000
9	ORD	11000
13	ORD	10000
27	ORD	9500

Figure 1(c). A_2: An approximation of E

id	Airport	Length
10	ORD	8000
13	ORD	10000
1	MDW	6000
6	MDW	7500
7	MDW	8000
9	ORD	11000

Figure 1(d). A_3: An approximation of E

the possible set P_i is a superset containing the exact answer. A possible value of P_i is {red, pink, maroon}, a set of three reddish colors. This answer improves as elements "red" and/or "pink" are deleted from P_i.

The exact answer to the query "How many cars of make Oldsmobile are available?" is the value of an aggregate function over the set O of data objects that satisfy the query constraints, such as make = Oldsmobile. In this case, the aggregate function is "count". Since the domain of the count function is a set of non-negative numbers in this case, as an approximation to the value of such an aggregate function, we can provide the values of the aggregate function defined over a certain set C and a superset $C \cup P$ of O; where (C, P) is an approximation of O. The values of the aggregate function count over the certain set and over the superset give us a superset of values, such as the range "2 to 10", or $\{2, 3, \cdots 10\}$. This approximate answer improves as more data objects are processed; the certain set increases in size and the possible set decreases in size. The counts over a bigger certain set and/or smaller possible set give us a smaller range, such as $\{4, 5, \cdots 9\}$.

Some queries require a yes or no answer, such as, "Is the number of runways at O'Hare greater than 10?". The answer is derived from the value of "count" over the set R of all runways at O'Hare. A meaningful approximation of the exact answer can be derived from an approximation (C, P) of the set R. The data objects that satisfy the query constraints, such as airport = O'Hare, are members of C. As long as the certain set in (C, P) contains 10 elements or less, the derived approximate answer remains "no". In addition to this value, we can also provide, as part of an approximate answer to such a query, a percentage value of the amount of data retrieved and processed thus far to produce (C, P), and the value of count over the current certain set C. An example is "No - 60%, number of runways ≥ 3". As more data are retrieved and processed, the percentage of data processed increases monotonically, and the value of the count function over C becomes closer to the exact value.

We can make the approximation semantics more meaningful by comparing subsets not only on the basis of their cardinalities, but also on the basis of some metric that measures the distances of their elements to the exact answer. We use such a measure of accuracy, called a *distance function*, to quantify how much better one approximate answer is than another. A distance function induces a partial-order relation over the set of all approximate answers of an exact answer E. The query processor uses this measure of accuracy to identify a good initial approximation of the exact answer when query processing starts. It also uses this measure to choose the next set of data objects to be retrieved and processed so that a series of approximate answers of improving accuracy are produced.

4. Monotone Query Processing

APPROXIMATE uses a monotone query processing algorithm to produce an approximate answer and maintains semantic information for an effective implementation of this algorithm. As an alternative to processing the possible tuples during query processing, APPROXIMATE works on templates of the possible tuples P in an approximate relation. The approach used in APPROXIMATE to generate templates of possible tuples is similar to the one used to produce intensional answers [7]. APPROXIMATE maintains an object-oriented view of the database and uses the information provided by this view to generate the templates. In this view, a base relation, or a segment of the relation, is a class. Tuples in the relation, or the segment, are instances of the corresponding class. The classes are organized into a collection of class hierarchies. Each class hierarchy supplies information about a base relation stored in the database. Examples of the types of information provided by a class hierarchy include the domains of the attributes of the instances of a class and the retrievable unit of data. This information is accessed along with the base relations during query processing.

The basic monotone query processing algorithm works as follows. It begins by representing a query by a query tree. Each node in the query tree represents a relation that is the result of a relational operation. An initial approximation is assigned to every node in the query tree. This initial approximation, and subsequent improved approximations of the standard relation represented by the node, are stored in an approximate object, which is created by APPROXIMATE for the node before query processing starts. The value of this approximate object gives an approximate relation of the standard relation. The object has three variables: the certain_part, possible_part, and OP.

The value of its certain_part is a certain set C containing all the data objects that are certainly in the standard relation. Initially, this certain set is empty for every approximate object. The value of its possible_part is a set $\mathbf{P}$ of possible classes. The set of all instances of the classes in $\mathbf{P}$ is a possible set in an approximation $(C, \mathbf{P})$ of the standard relation. Initially, the value of the possible_part in an approximate object at a leaf node gives a template of all possible tuples that can possibly be in the base relation represented by the leaf node. Again, this template is obtained from the meta data about the base relation maintained by the query processor. The initial values of the possible_part in an approximate object at a non-leaf node can be obtained from the initial values of the leaf nodes in the the subtree rooted at the node and the relational algebra operations represented by the nodes in the subtree. The value of the variable OP is set to be the relational algebra operation represented by the node. The OP of the approximate object at a leaf node is an approximate_read. An approximate_read returns a segment of the requested base relation at a time. Again, information on which segments can be returned is given by the view maintained by the query processor.

Figure 2 shows the value $\mathbf{A}_2$ of an approximate object corresponding to the relation A_2 in Figure 1(c). The possible tuples in P_2 are instances of $\mathbf{P}_2 = \{\text{ORD-long-runways}\}$, the possible class of long runways at ORD airport.

Because standard relational algebra operations cannot operate on approximate objects, APPROXIMATE uses as query processing primitives a set of approximate relational algebra operations and the approximate_read. Each operation accepts an approximate object(s) as an operand and produces an approximate object as its result [3]. As each approximate_read of a leaf node is carried out, each returned segment causes additional certain tuples to be added to, and possible classes to be deleted from, the current approximation of the base relation. The value of the leaf node improves as more segments are returned. The improvement in the leaf nodes is propagated upward to the root node by reevaluating the nodes in the query tree. The value of the root node is updated with better

id	Airport	Length
6	MDW	7500
7	MDW	8000
10	ORD	8000
{ORD-long-runways}		

Figure 2. $\mathbf{A}_2$: An approximate object

values each time the root node is reevaluated.

This monotone query processing algorithm differs from traditional query processing where each node of the query tree is evaluated only when all of the required data are available. If any required base relation is not accessible, no answer is ever produced. This all-or-nothing query processing strategy does not degrade gracefully. In contrast, APPROXIMATE produces a chain of increasingly better approximate answers at the root node of the query tree, each integrating the effect of additional data processed. None but the final, exact answer requires all base relation data be available before it can be produced. If query processing terminates prematurely, some approximate answer in the chain will be returned and the quality of the returned answer increases monotonically with time.

Every approximate relational algebra operation involves operations on the possible classes as well as the certain tuples in its operand(s). The approximate relational algebra operations are implemented incrementally, so the same number of relational algebra operations is applied to the certain tuples during approximate query processing as during traditional query processing [9]. To minimize the overhead required to produce an approximate answer, APPROXIMATE delays the evaluation of the possible classes and instead maintains a symbolic expression of the possible classes and relational algebra operations to be applied to them. A possible class is not evaluated until query processing must terminate and an approximate answer is produced or the user requests the evaluation.

With each update to the value of the root node of the query tree, APPROXIMATE displays the certain tuples and the possible class names of the approximate answer. Figure 3 illustrates two approximate answers to the query "Select all the northerly and easterly runways at airports where the temp > 32° C". As this figure illustrates, an approximate answer allows the user to distinguish the data processed thus far from the data not yet processed. From the approximate answer in Figure 3(a), the user can see that the classes {IL-short-N} runways and {IL-short-E} runways have not been processed yet. Upon examining the approximate answer, a user may determine whether the approximate answer resulting from processing all of the long runways provides enough information and is a good enough answer. The user can request the evaluation of the possible classes, and Figure 3(b) illustrates the evaluation of the possible classes in Figure 3(a). Figure 3(c) illustrates an improved approximate answer.

5. Future Directions

We have described a monotone query processor called APPROXIMATE that produces an approximate answer if some of the data are not available or if there is not enough time to process a query. APPROXIMATE processes the data that are available or processes the data that can be processed within the time constraints. It produces a series of approximate answers that improves according to a partial-order relation. The same query processing strategy is used for producing approximate answers to set-valued queries and for single-valued queries, such as binary queries, aggregate queries, and set-valued queries with cardinality 1. APPROXIMATE uses semantic support, in the form of an object-oriented view and a predefined set of distance functions, to identify a good initial approximation and a strategy for improving an approximation.

In the future, we will determine the scalability and efficiency of the proposed scheme. To accomplish this, we need to interface the query processor to some real-life database systems and make the query processor as efficient and robust as we can. Some candidate databases are those used to support navigation, machine vision, and computer aided engineering, as well as databases on students'

id	Airport	Length	Temp
1	ORD	11000	35
2	CMI	10000	38
7	ORD	10500	35
33	MDH	11000	50

Possible Classes:
{IL-short-N, IL-short-E}

Figure 3(a). An approximate answer

id: { 32-36, 1-13 }
Airport: { CMI,JOT,MDH,MDW,ORD,SPI }
Length: { 7000-9500 }
Temp: { > 32 }

Figure 3(b). Possible class evaluation

id	Airport	Length	Temp
1	ORD	11000	35
2	CMI	10000	38
7	ORD	10500	35
33	MDH	11000	50
36	MDW	8000	35

Possible Classes: {IL-short-E}

Figure 3(c). An approximate answer

academic reports. A natural extension of this research is to consider the case when the database contains incomplete or partial values. This problem is closely related to the imprecise update problem. Imprecise updates introduce incomplete information and partial values into the database. We want to investigate the feasibility of partial updates that require no error recovery action to complete the update. Such an update may introduce uncertainty or incompleteness in the information contained in the database, but will not lead the database to an unsafe, and hence, unacceptable state.

References

[1] Lin, K. J., S. Natarajan, J. W. S. Liu, and T. Krauskopf, "Concord: A System of Imprecise Computations," *Proc. COMPSAC '87*, Tokyo, Japan, pp. 75-81, Oct. 1987.

[2] Chung, J. Y., J. W. S. Liu, and K. J. Lin, "Scheduling Periodic Jobs that Allow Imprecise Results," *IEEE Transactions on Computers*, Vol. 39, No. 9, pp. 1156-1174, Sept. 1990.

[3] Vrbsky, S. V. and J. W. S. Liu, "An Object-Oriented Query Processor That Produces Monotonically Improving Approximate Answers," *7th International Conference on Data Engineering*, Japan, pp. 472-481, April 1991.

[4] Buneman, P., S. B. Davidson, and A. Watters, "A Semantics for Complex Objects and Approximate Queries," *Proceedings of the 7th Symposium on the Principles of Database Systems*, pp. 305-314, March 1988.

[5] Ozsoyoglu, Bultekin, Z. Meral Ozsoyoglu and Wen-Chi Hou, "Research in Time- and Error-Constrained Database Query Processing," *Workshop on Real-Time Operating Systems and Software*, Virginia, May 1990.

[6] Motro, Amihai, "VAGUE: A User Interface to Relational Databases that Permits Vague Queries," *ACM Transactions on Office Information Systems*, Vol. 6, No. 3, pp. 187-214, July 1988.

[7] Chu, Wesley W., Rei-Chi Lee and Qiming Chen, "Using Type Inference and Induced Rules to Provide Intensional Answers," *7th International Conference on Data Engineering*, Japan, pp. 396-403, April 1991.

[8] Smith, Kenneth P., and J. W. S. Liu, "Monotonically Improving Approximate Answers to Relational Algebra Queries," *Proc. COMPSAC '89*, Orlando, Florida, Sept. 1989.

[9] Vrbsky, S. V. and J. W. S. Liu, "APPROXIMATE: A Query Processor That Produces Monotonically Improving Approximate Answers," submitted to *IEEE Trans. on Knowledge and Data Engineering*.