A Multi-Resolution Relational Data Model*

Robert L. Read Donald S. Fussell Avi Silberschatz

Department of Computer Sciences
University of Texas
Austin, TX 78712-1188
Email: {read, fussell, avi}@cs.utexas.edu

Abstract

The use of data at different levels of information content is essential to the performance of multimedia, scientific, and other large databases because it can significantly decrease I/O and communication costs. The performance advantages of such a *multi-resolution* scheme can only be fully exploited by a data model that supports the convenient retrieval of data at different levels of information content. In this paper we extend the relational data model to support multi-resolution data retrieval. In particular, we introduce a new partial set construct, called the *sandbag*, that can support multi-resolution for the types of data used in a wide variety of next-generation database applications, as well as traditional applications. We extend the relational algebra operators to analogous operators on sandbags. The resulting extension of the relational algebra is sound and forms a foundation for future database management systems that support these types of next-generation applications.

*This material is based in part upon work supported by the Texas Advanced Technology Program under Grant No. ATP-024, the National Science Foundation under Grant Nos. IRI-9003341 and IRI-9106450, and grants from the IBM and Hewlett-Packard corporations.

Proceedings of the 18th VLDB Conference
Vancouver, British Columbia, Canada 1992

1 Introduction

Manipulating very large data objects such as images, sounds and scientific data incurs large I/O and communication costs. A relatively unexplored approach to decreasing these costs is to retrieve and use a smaller version of an object rather than the complete object when such an approach is feasible. For some types of data, such as representations of continuous functions, (e.g., images and sounds), we can compute a smaller version of the data, or an *approximation*, that retains the character of the data and is satisfactory for many purposes. An approximation provides less information than the data that is its source, but is completely consistent with it. Such an approximation is called *partial* or *incomplete* data, similar in principle to the partial data of existing models of incomplete information [1-13]. When an approximation suffices and is significantly smaller than the original, complete data object, retrieving it instead may incur lower costs because fewer bytes are accessed or moved.

For example, consider a multimedia database that contains raster images. A typical high-resolution (1024×1024 pixel) full color (24-bit) raster image contains 3 megabytes. In typical current computing environments, such an image is likely to take more than a second to retrieve (perhaps much more). However, a 256×256 pixel 8-bit color raster image of the same scene is satisfactory for many purposes. This approximation is 1/48th the size of the original, complete image. The costs of operations on large bodies of data are dominated by throughput limitations rather than overhead, seek time, and propagation delay, so lower resolution images can be retrieved much more rapidly than higher resolution images. The lower resolution pictures will often be useful in their own right or as rapidly appearing previews of the complete pictures. We call a system that can produce approximations as well as complete data a *multi-resolution* data retrieval system. The term multi-resolution is borrowed from

graphics, but we apply it to all kinds of data.

Many applications of growing importance [14] manipulate huge quantities of data. These include multimedia databases, voice-mail systems, image processing applications, HDTV, graphics applications such as CAD/CAM, flight simulators and virtual reality systems, geographic [15] and astronomic databases [16], and scientific applications such as seismic processing. A multi-resolution data retrieval system is essential to the performance of these applications. Multi-resolution is a natural approach already used in an *ad hoc* manner to decrease I/O, storage, and communication costs [17]. The approach presented in this paper is a logical but novel extension of ideas in the fields of graphics [18, 19, 20] and databases to systematically address the needs of these types of next-generation applications.

Database management systems (DBMSs) are commonly used for accounting and record keeping and other "traditional" applications because they provide convenient data storage and retrieval services for these types of data. If DBMSs are to fully address the needs of the growing number of applications that demand multi-resolution, they must be extended to retrieve data at multiple resolutions conveniently. We propose to accomplish this by:

- extending existing data models to give precise meaning to multi-resolution data and queries,
- allowing the user to control the resolution of query results, and
- developing techniques for efficiently implementing this extended model.

This paper describes a formal multi-resolution relational data model that forms a foundation of and a necessary first step towards a practical multi-resolution DBMS. In Section 2 we motivate multi-resolution and show that existing data models and techniques do not suffice to exploit it conveniently. Section 3 presents multi-resolution primitive types and tuples. Section 4 introduces a new construct for representing incomplete information about sets, called the *sandbag*. and discusses its expressiveness. Section 5 defines operators analogous to the standard relational operators for sandbags. The sandbag and these operators form a multi-resolution data model and an algebra that is a generalization of the relational algebra. Finally, in Section 6 we briefly mention some of the future work that needs to be accomplished in order to usefully implement and exploit this data model.

The proofs of all theorems stated in this paper appear in a technical report [21]. We address the implementation and computational complexity of sandbag operations in [22]. These issues are beyond the scope of this paper and not discussed further here.

2 Why Multi-Resolution

Multi-resolution is the concept of viewing data at different levels of information content. The fields of denotational semantics [13, 23] and information theory [24, 25, 26, 27] provide an intuitive and a formal definition of *information content* and other concepts that underlie multi-resolution (*approximation, consistency, resolution, information-theoretic partial order*). We repeat the informal definitions of these concepts here to provide the reader the necessary intuitions.

Data describes the real world. Some data is more descriptive than other data. For instance, the daily list of stock volumes, opening prices and closing prices is more descriptive of market history than averages and indices computed over many stocks, such as the Dow Jones Industrial Average. Similarly, a high quality audio recording is more descriptive of music played than a poor, scratchy recording. The more descriptive data is, the more *information* it contains. We use the term *resolution* synonymously with "information content".

Only by considering the meaning of computer-manipulated data, or the descriptions of the real world we obtain from it, can we define the information-theoretic notion of approximation. A data object X *approximates* a data object Y if every world described by Y is described by X. The approximates relation is a natural partial order of data that could be called the "goodness" or "precision" of the data. X approximates Y if and only if Y describes the world better than X, and is consistent with X. Intuitively, if Y describes the world, then X is a version of Y that tells us less about the world than does Y, but from which we will not draw any false conclusions. The meaning of data, and hence the notion of approximation, is always application dependent, as is the case with the example of raster images mentioned in the introduction.

We focus on a particular property of approximations that is generally true, though not universally obtained in practice.

> If object X approximates object Y and X is lower-resolution than Y, then X requires less space to be represented by a computer than Y.

Accessing a large object requires many accesses to main memory and/or many expensive I/O operations. Our goal is to use the general relationship between approximations and space to improve performance by computing against lower-resolution data when possible. Because the greatest savings in space, and hence

time, are possible when approximating very large objects, our examples and motivating applications tend to emphasize such objects. However, our proposed framework applies to all sizes of objects.

2.1 Multi-Resolution is Demanded

In many applications, accessing low-resolution data is quite adequate. In some cases, the highest-resolution data cannot always be used. For instance, when a raster image has higher resolution than a device it is to be a displayed upon, a lower-resolution approximation must be produced. Such a reduction of resolution is done in an *ad hoc* manner in some graphic and scientific applications. For instance, in databases of images [16, 17], "browse images" are created at low-resolution so that an "overview" of the data is obtained, that allows preliminary examination of the data for quality or interest, and for publication. A similarly "zooming" capability is useful in cartographic applications. The enormous volume of data in astronomy, ecology, meteorology, geology, and geography [15] databanks could be better exploited by database technology that systematically and conveniently supports resolution control. Systematically treating smaller versions of very large data items is a first step towards the difficult problems of terabyte-sized databases of any kind of data, from images to financial data.

There are many applications where retrieving data at complete resolution takes too long, either because of some real-time constraint, or because the user is dissatisfied with performance. This problem can sometimes be addressed by retrieving low-resolution data quickly or before a deadline. The similar idea of *imprecise computation* has been suggested as an approach to meeting real-time constraints on database queries [5, 6, 28, 29, 30]. A multi-resolution system naturally extends and complements this approach to real-time databases by supporting larger tradeoffs in imprecision for time.

For instance, a technique for displaying images called *progressive transmission* [20] is to display a low-quality image immediately, so the user may peruse it while the larger, better image is being retrieved via a network. The image is then gradually or abruptly improved, or refined, until it is complete. Two similar ideas are *adaptive refinement* [18] and *progressive refinement* [19], that are the incremental computation of the features of a graphic image in order of their visual importance to a viewer. If a data model that supports multi-resolution is available, these ideas can be naturally extended to include the idea of retrieving a series of data objects, each of which is better than the last, until the complete answer is retrieved. We adopt the term *progressive refinement* to refer to this approach to data retrieval from disk, network, and main memory. Progressive refinement is a general technique for improving the utility of any query system, and is inherently a multi-resolution technique. Although networks, disks, memory and CPUs continue to improve rapidly, the volume of data demanded will always match their capabilities, and progressive refinment will remain a useful technique.

2.2 Existing Technology and Data Models are Insufficient

Existing database technology does not support multi-resolution or progressive refinement conveniently. The database management system should hide the implementation details of multi-resolution data retrieval, presenting resolution transparency to the user without sacrificing the user's ability to obtain specific resolutions. Computed fields and object-oriented approaches allow approximations to primitive data objects to be computed and returned. However, since the relationship between levels of resolution is not part of these data models, a burden is placed on the user to explicitly manage resolution without a unified or systematic framework. For instance, to implement progressive refinement of a query in existing systems, the user must submit a separate query explicitly for each desired improvement. Similarly, to meet some performance constraint, the user needs to guess which resolution will be produced timely. Without a model of multi-resolution, the system cannot dynamically adapt returned resolution to user's constraints. Furthermore, existing systems do not allow systematic approximation of sets, although sets are fundamental to most data models, and large sets with many elements are often manipulated.

A data model and query evaluation system that contains the relationship between data and approximations offers advantages that cannot be obtained by existing systems:

- Resolution of responses can be traded for speed to meet real-time constraints at compile time or at run time, based on the dynamic situation within the database.
- Progressive refinement has a precise meaning and can be provided systematically.
- Sets of values can be systematically approximated.
- The relationship between resolutions can be exploited by special storage structures and algorithms.

3 Multi-resolution Primitive Types and Tuples

In this section we introduce multi-resolution primitive types and multi-resolution tuples. Multi-resolution (MR) primitive types are needed because there are various types of data that we wish to treat at different resolutions that are more conveniently represented as primitive types rather than as tuples or relations. These types include images, sounds, and geometric figures. An MR primitive type $\mathcal{T}$ is a user-defined data type consisting of values at various resolutions. All MR primitive types are partially ordered by a user-defined relation $\sqsubseteq$, and have a bottom element $\perp$ that is $\sqsubseteq$ all elements. The partial order $\sqsubseteq$ corresponds to the *approximates* relation defined informally in Section 2. The highest-resolution elements of a primitive type are called the *total* elements, designated by the predicate $\underline{\text{tot}} : \mathcal{T} \mapsto \{\text{true}, \text{false}\}$. (We denote function application by an infix dot. Thus $\underline{\text{tot}}$ applied to x is written $\underline{\text{tot}}.x$.)

The database programmer defines the MR primitive types according to his or her needs, including the approximates relation ($\sqsubseteq$) and a type-specific bottom element $\perp$. These types should not pervert the intention that the formally defined relation $\sqsubseteq$ corresponds to the meaning of *approximates* informally defined in Section 2.

In an implementation, the user will provide application dependent functions that can produce low-resolution data from high-resolution data. A data definition language that supports the specification of these functions by the database programmer and use of such functions by the DBMS and its query evaluator are important implementation details that are future work.

Rich partially ordered primitive types with many different resolutions distinguish our approach from research concerned mostly with the semantics of null values. Of course, many primitives types within a given application, such as unique identifiers, booleans, and small integers, will have only one useful resolution level, plus a null value ($\perp$). These types, and the functions mentioned above, will be provided by the DBMS.

We illustrate these concepts with two typical examples. The first is the canonical example of a partially ordered data type of real intervals [23]. The second example is a multi-resolution primitive type whose members are rasters at different resolutions.

Example 1: A real number can be approximated by a real interval that contains it. We say a narrower interval is higher-resolution than a wider interval. The type of real intervals consists of all intervals, and is partially ordered by containment:

$$[a,b] \sqsubseteq [c,d] \equiv a \leq c \wedge b \geq d$$

The total elements of this type are the intervals $[a,a]$ that only contain one real number, e.g. $\underline{\text{tot}}.[3.14, 3.14] = \text{true}$. The bottom element ($\perp$) of this domain is $[-\infty, +\infty]$. □

Example 2: Consider a multi-resolution raster image primitive type with five distinct levels of resolution. An application programmer might construct the resolution levels such that:

- level 4 images are 1024×1024 24-bit color rasters,
- level 3 images are 256×256 8-bit color rasters,
- level 2 images are 128×128 black-and-white rasters,
- level 1 images are 16×16 black-and-white icons, and
- level 0 is a null value for this type.

The type of raster images consists of all images at these five resolution levels, in contrast to Example 1, which has an infinite number of resolution levels. The ordering $\sqsubseteq$ would be programmed by the user so that:

$X \sqsubseteq Y \equiv$ $X = Y$ or X and Y are computed from the same picture and X has a lower resolution level than Y.

The total elements of this type are the level 4 rasters. The level 0 element is the bottom element ($\perp$) of this type. □

Having defined the concept of multi-resolution primitive types, we can now define the multi-resolution tuple, that is constructed from user-defined and built-in MR primitive types. To specify a tuple-type, the user merely specifies a sequence of types; the cross-product of these types is the tuple-type. Because tuples are an inherent and basic type supported by the database, the user does not have to specify any of the operations required by a primitive type to define a tuple-type. However, we give definitions of some of these operations here, in order to define a complete data model.

Definition 1: For a tuple-type $\mathcal{T}$ constructed from a sequence of N primitive types $[\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_{N-1}]$,

- Given two tuples x and y in $\mathcal{T}$,
  $x \sqsubseteq y \equiv (\forall\, c : 0 \leq c \wedge c < N : x[c] \sqsubseteq y[c])$
- The bottom element ($\perp$) of $\mathcal{T}$ is the sequence of the bottom elements from each component type of the tuple-type.
- $\underline{\text{tot}}.x \equiv (\forall\, c : 0 \leq c \wedge c < N : \underline{\text{tot}}.(x[c]))$ □

Notice that tuples are partially ordered according to the resolution of their constituents.

4 Multi-Resolution Sets

The relational data model is built from primitive data types, tuples, and sets of tuples, or relations. Constructing information content-based partial orders of primitive types and tuples is straightforward, as we have seen, and has been proposed in various forms even though it is not common in practice. Some similar means of constructing a multi-resolution partial order of sets (or relations) is required because sets are essential to most data models, including the relational. Furthermore, sets with many members are large, and hence expensive. This is a surprisingly interesting problem [2, 3, 5, 6, 11].

An approach to approximating a set of total data elements from a partially ordered set has been suggested in [2], where the authors have introduced the *sandwich* concept. A sandwich describes a set B of total elements by "sandwiching" them between two sets. These are a *consistent set*, that contains tuples guaranteed to approximate (in the partial order of tuples) some tuple in B, and a *complete set*, each tuple of which is guaranteed to be approximated by some tuple in B. If these sets are restricted to total elements, as in [5, 6], then they are simply a subset and a superset of B, respectively. For instance, the set {Alice, Bob, Carla} could be approximated by the consistent set {Alice, Bob} and the complete set {Alice, Bob, Carla, Dan, Jo}.

This idea was used in [2] to construct a consistent semantics for complex objects [1] and to present a very expressive approximate query system. In contrast, this idea was used in [5] to construct a query system that exhibits progressive refinement, which is more in the spirit of this paper.

4.1 The Sandbag Scheme

Because the sandwich construct cannot capture information about cardinality, we extend the sandwich to a new partial set construct, called the *sandbag*. (Throughout this paper, the capital letters P, Q, S, and T stand for sandbags, the capital letters B and C stand for sets of total elements, and the calligraphic letters $\mathcal{D}$ and $\mathcal{T}$ stand for domains or types of elements that are partially ordered and have a bottom element.) For instance, consider the set $B = \{1.1, 3.14, 5.0\}$. The fact that B has three members cannot be represented by a sandwich. A related problem is that the sandwiching sets are restricted to co-chains in the partial order of the set's elements, so that even if cardinality could be captured, one could not represent the facts that B contains three elements, two of them are greater than 2.5, and one of them is 3.14.

The sandbag can represent these facts. The basic idea of the sandbag is to approximate a set B of total elements by using partial data elements. Each partial data element $d \in \mathcal{D}$ is mapped into a range that provides lower and upper bounds on the number of elements in B that are approximated by d. We call the number of total elements in some set approximated by d *the number above* d. To formalize this concept we define the following:

Definition 2: Let $\mathcal{D}$ be a partially ordered data domain, $B \subseteq \mathcal{D}$ be a set of total elements, and $d \in \mathcal{D}$ be any data element. The number of total elements above the element d in the set B is denoted by $\underline{\text{numa}}.B.d$, and defined by:

$$\underline{\text{numa}}.B.d = |\{x \mid d \sqsubseteq x \ \wedge \ x \in B\}| \quad \square$$

To illustrate this, consider B defined above, that can be considered a shorthand for the set of total elements from the partially ordered domain of real intervals (Example 1) {[1.1,1.1], [3.14,3.14], [5.0,5.0]}. The number of elements above the partial data element $d = [2.5, 6.0]$ is 2 ($\underline{\text{numa}}.B.d = 2$). A sandbag describing B might provide a lower bound for d of 1 and an upper bound of 3. This is consistent with B, but is not the most precise consistent bounding interval possible, that would of course be [2,2]. The capacity for imprecision is essential to the sandbag. Before presenting additional examples, we formally define the concept of a sandbag.

Definition 3: Let $\mathbb{N}'$ be the set of naturals ($\mathbb{N}$) plus infinity, $\mathbb{N} \cup \{\infty\}$, and let $\mathcal{D}$ be a partially ordered data domain. A sandbag S is two functions, denoted $\underline{\text{mina}}.S$ and $\underline{\text{maxa}}.S$.

- $\underline{\text{mina}}.S.d : \mathcal{D} \mapsto \mathbb{N}'$, a lower bound on the number of total elements in a set consistent with S that are above d.
- $\underline{\text{maxa}}.S.d : \mathcal{D} \mapsto \mathbb{N}'$, an upper bound on the number of total elements in a set consistent with S that are above d. $\square$

We now present a definition that formalizes the concept that a sandbag is consistent with, (or approximates) a set.

Definition 4: Let $\mathcal{D}$ be a partially ordered data domain, and $B \subseteq \mathcal{D}$ a set of total elements. We say that a sandbag S is consistent with a set B, denoted $\underline{\text{sbcons}}.S.B$, when:

$$(\forall\, d : d \in \mathcal{D} : \ \underline{\text{mina}}.S.d \leq \underline{\text{numa}}.B.d \ \wedge \ \underline{\text{maxa}}.S.d \geq \underline{\text{numa}}.B.d) \quad \square$$

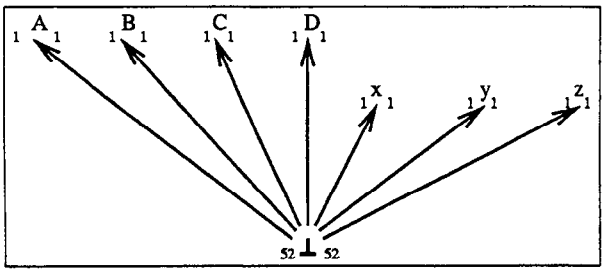

Figure 1: A Sandbag of Magazine Cover Rasters.

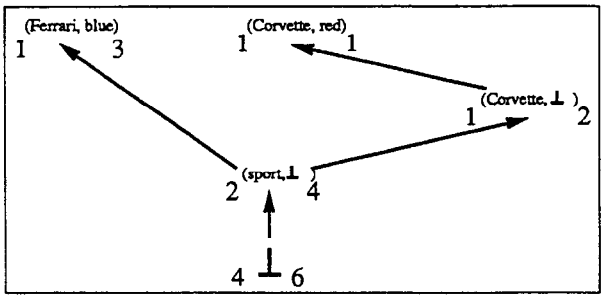

Figure 2: A Sandbag of Facts About a Set of Cars.

4.2 Using the Sandbag to Approximate Sets

The sandbag construct can usefully approximate sets of diverse types. For instance, it can represent a set of raster images of different resolutions. Raster images representing the covers of a weekly magazine for a year could be approximated by a sandbag that indicates there are exactly 52 images in the set, 4 of them (A, B, C, and D), are available at full resolution, 3 (x, y, and z) are available at some lower resolution, and no others are mentioned. This can be represented diagrammatically, as in Figure 1.

In this style of diagrammatic representation of a sandbag, a node represents an element of the multi-resolution type over which the sandbag is defined. An arrow indicates the approximation relationship (an arrow from p to q indicates $p \sqsubseteq q$). The small numbers to the left and right of a node indicate the mina and maxa value of that node, respectively. The height of a node in the diagram loosely corresponds to its resolution.

The sandbag can represent incomplete information in the form of constraints expressed via a partial order. For example, suppose it is known that a car rental company has between 4 and 6 cars ready for rental, 2 to 4 of them are sports cars of unspecified color, at most 2 are Corvettes, at least 1 is a blue Ferrari, and exactly 1 is a red Corvette. This can be represented by a sandbag, and is depicted in Figure 2.

Note that in this example we are using the partial ordering of tuples, and assuming partial orders for both the 'color' type and the 'make' type. 'color' and 'make' have a bottom element, and 'make' is assumed to have at least one intermediate resolution, of 'car style', of which 'sport' is an example. Thus, 'sport' $\sqsubseteq$ 'Corvette'.

One of the most interesting uses of a sandbag is to construct compact, imprecise representations of an ordered set. For example, consider a relation of bank accounts. If integer intervals are a primitive, the sandbag could represent a conglomeration of facts such as:

(a) At most five and at least four accounts between account #100 through #112 have a balance in the range \$60,000 to \$100,000.

(b) Account #107 has exactly \$43,737.

(c) Account #108 has between \$40,000 and \$49,000.

(d) Accounts #50 through #500 have exactly 100 accounts with less than \$10,000.

(e) Accounts #50 through #500 have at most 200 and at least 150 accounts with more than \$50,000.

These facts are depicted in Figure 3, wherein a partial data element is represented as a rectangle labeled near its lower left corner. The mina and maxa values of these partial elements are to the left and right of the label, respectively.

The mina and maxa functions enable queries on the sandbag to obtain interesting information. The precision in the results will depend on the quality of the facts from which it is constructed. For instance, given the facts above:

- A selection of accounts having between \$40,000 and \$45,000 will definitely include account #108 (mina applied to #108 will produce 1), but cannot exclude #107 (maxa applied to #107 will produce 1, and mina will produce 0).
- The number of accounts in the range #100 to #200 with less than \$8,000 is at most 100 (maxa

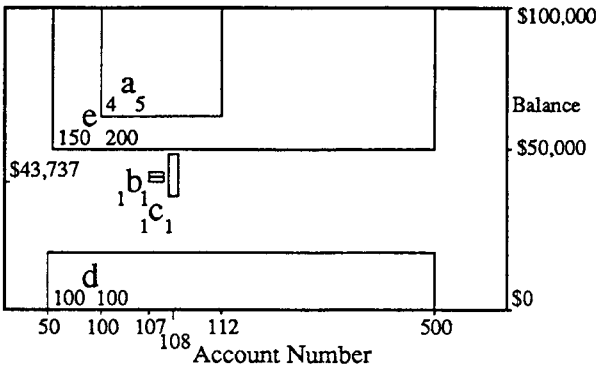

Figure 3: A Sandbag Approximating a Bank Account.

is 100), and we cannot be sure that there are any such accounts (mina is 0).

Information such as that depicted in Figure 3 can be combined with statistical sampling. As more samples are added to a sandbag, it becomes better and better defined. The space required by the sandbag depends on the quantity of such information that it contains as well as its implementation.

The sandbag can represent the fact that a tuple x is a *maybe tuple* [4, 11] in a relation, if $\underline{\text{mina}}.x = 0$ and $\underline{\text{maxa}}.x = 1$. The *maybe tuple* is closely related to the sandwich and the *I-table* [6]. Both provide a means of stating that the tuples in a relation cannot be excluded but are not known to be in the set. The sandbag can model these constructs in a straightforward manner without any particular use of an underlying partial order. Similarly, "normal" relations and sets consisting of only total elements, such as are used in traditional relational databases, are trivially modeled.

Thus, the sandbag can represent approximations of at least five distinct kinds of sets:

- sets of objects at various resolutions,
- incompletely specified sets,
- sets of points in continuous spaces,
- statistically sampled sets, and
- sets of normal tuples and maybe tuples.

As far as we know, this flexibility is not offered by any other representation of incomplete information about a set, nor is it systematically available in any proposed data model [1-13]. If the efficiency and convenience of the sandbag can match its expressiveness, it may be a useful part of multi-resolution databases. The sandbag construct is a tradeoff between flexibility and ease of implementation that leans towards flexibility.

4.3 A Partial Order of Sandbags

In this subsection we develop the machinery that allows sandbags to be general multi-resolution types. Fundamental to this is the notion that sandbags, like primitives types and tuples, are partially ordered by the amount of information they provide about the sets that they approximate. The tighter the bounds provided by a sandbag, the more informative it is. A sandbag P approximates a sandbag Q if the bounds provided by mina and maxa on Q are at least as tight as and consistent with those of P.

Definition 5: Let $\mathcal{D}$ be a partially ordered data domain. The approximates relation $\sqsubseteq$ on sandbags is defined by:

$$P \sqsubseteq Q \equiv (\forall x : x \in \mathcal{D} : \underline{\text{mina}}.P.x \leq \underline{\text{mina}}.Q.x \wedge \underline{\text{maxa}}.P.x \geq \underline{\text{maxa}}.Q.x) \ \square$$

The relation $\sqsubseteq$ defined as above is in fact a partial order of sandbags, because it is reflexive, transitive, and antisymmetric.

Sandbags have a bottom element and a notion of totality, just as did primitives and tuples. If for all x the lower limit $\underline{\text{mina}}.S.x = 0$ and the upper limit $\underline{\text{maxa}}.S.x = \infty$, then S provides no information whatsoever. This is the null value of sandbags in our model. Since a sandbag S is a pair of functions $\underline{\text{mina}}.S$ and $\underline{\text{maxa}}.S$, we formally define a sandbag by defining the value of $\underline{\text{mina}}.S$ and $\underline{\text{maxa}}.S$ on every element of $\mathcal{D}$.

Definition 6: The bottom element of the sandbag type is denoted $\perp_{\text{SB}}$ and defined by:

- $\underline{\text{mina}}.\perp_{\text{SB}}.x = 0$
- $\underline{\text{maxa}}.\perp_{\text{SB}}.x = \infty$

for every element x in a partially ordered data domain $\mathcal{D}$. $\square$

The bottom-most element of sandbags under the partial order $\sqsubseteq$ is in fact $\perp_{\text{SB}}$.

Definition 7: Let $\mathcal{D}$ be a partially ordered data domain. A sandbag S is *total*, denoted $\underline{\text{tot}}.S$, if and only if:

$$(\forall x : x \in \mathcal{D} : \underline{\text{mina}}.S.x = \underline{\text{maxa}}.S.x) \ \square$$

Total sandbags are in fact the top-most elements of the type of sandbags. If a sandbag is total, there is at most one set consistent with it, and the members of that set can be exactly determined from the sandbag. Between these two extremes of zero and total information, a sandbag provides varying degrees of information about a set it approximates. Alternatively, we think of a sandbag as limiting the number of sets consistent with it. The more informative a sandbag is, the fewer sets are consistent with it. All sets are consistent with $\perp_{\text{SB}}$ (i.e. $(\forall B :$ B is a set of total elements from $\mathcal{D} : \underline{\text{sbcons}}.\perp_{\text{SB}}.B)$).

Unfortunately, a sandbag need not be consistent with any set, because the values of mina and maxa can easily contradict each other.

Definition 8: Let S be a sandbag and B be a set of total elements. We say that S is *externally consistent*, denoted by $\underline{\text{extcons}}.S$, if and only if: $(\exists B : : \underline{\text{sbcons}}.S.B) \ \square$

As might be expected, we are exclusively concerned with externally consistent sandbags, because we are only interested in using sandbags to inform us about actual sets.

The sandbag concept was inspired by [2], in which the authors develop the sandwich representation of partial sets and a meaningful partial order of that representation. The sandbag concept extends that work because it is a more expressive partially ordered set construct. Thus, sandbags may be very useful in data models based on complex objects or higher-order relations. Because we are primarily concerned with improving performance by retrieving approximations and thus saving I/O costs, we develop a flat relational model for the sake of simplicity rather than a higher-order model.

5 A Multi-Resolution Relational Algebra

A relation is a set of tuples. We call a sandbag over a multi-resolution tuple-type a *multi-resolution relation.* Following tradition, we construct a relational-like algebra by defining analogs for each relational operators ($\cap$, $\cup$, $-$, $\times$, σ, π, and $\bowtie$) that take sandbags as arguments and produce a sandbag as a result. We define each of these operators so as to produce the best results possible that are guaranteed to be consistent with the arguments. It is therefore not surprising that each of these MR operators enjoys properties essential to a partial data model:

Property 1 (Soundness): A binary sandbag operator *setop'* is said to be *sound* with respect to a set operator *setop* if and only if for all sandbags P and Q and sets B and C:

$$\begin{array}{ll} & \underline{\mathsf{sbcons}}.P.B \ \wedge \ \underline{\mathsf{sbcons}}.Q.C \\ \Rightarrow & \underline{\mathsf{sbcons}}.(P \ setop' \ Q).(B \ \ setop \ \ C) \ \square \end{array}$$

Intuitively, an MR operator is sound if the result of applying it to approximations to total sets is always consistent with the result of the standard relational operator applied to the total sets.

Property 2 (Completeness): A binary sandbag operator *setop'* is said to be *complete* with respect to a set operator *setop* if and only if for externally consistent sandbags P and Q and sets B and C:

$$\begin{array}{l} (\forall P, Q, x : : \\ \ (\exists B, C : \underline{\mathsf{sbcons}}.P.B \ \wedge \ \underline{\mathsf{sbcons}}.Q.C : \\ \ \underline{\mathsf{numa}}.(B \ \ setop \ \ C).x = \underline{\mathsf{mina}}.(P \ setop' \ Q).x) \\ \wedge (\exists B, C : \underline{\mathsf{sbcons}}.P.B \ \wedge \ \underline{\mathsf{sbcons}}.Q.C : \\ \ \underline{\mathsf{numa}}.(B \ \ setop \ \ C).x = \underline{\mathsf{maxa}}.(P \ setop' \ Q).x)) \ \square \end{array}$$

Intuitively, an MR operator is complete if it provides the most informative result that is guaranteed to be consistent.

An operator has the completeness property[1] if for each partial data element there exist some sets consistent with its arguments for which the bounds provided are the correct numa value. Two further useful properties follow from Soundness and Completeness:

Property 3 (Monotonicity): A binary sandbag operator *setop'* is *monotonic* if and only if:

$$P \sqsubseteq Q \ \wedge \ S \sqsubseteq T \ \Rightarrow \ P \ setop' \ S \sqsubseteq Q \ setop' \ T \ \square$$

Intuitively, an MR operator is monotonic if the results of the operator never get worse (contain less information) when applied to arguments that are better (contain more information, or are higher-resolution).

Property 4 (Totality Preservation): A binary sandbag operator *setop'* is *totality preserving* if and only if:

$$\underline{\mathsf{tot}}.P \ \wedge \ \underline{\mathsf{tot}}.Q \ \Rightarrow \ \underline{\mathsf{tot}}.(P \ setop' \ Q) \ \square$$

Intuitively, an MR operator preserves totality if it produces a total result when applied to total arguments.

These properties are defined analogously for unary operators, such as projection of a list L (π_L) and selection by a predicate θ (σ_θ).

5.1 Relational Operators on Sandbags

In order to extend the relational algebra to a multi-resolution relational algebra as naturally as possible we define sandbag operators analogous to each relational operator. We denote the sandbag operator analogous to a relational operator *setop* by priming it, i.e. *setop'* . Thus, the MR relational operators corresponding to $\cap$, $\cup$, $-$, $\times$, σ, π, and $\bowtie$ will be denoted by $\cap'$, $\cup'$, $-'$, $\times'$, σ', π', and $\bowtie'$, respectively. Intuitively, Properties 1-4 will hold for sandbag operators only if we define them to produce the most informative sandbag guaranteed to be consistent with their arguments.

Definition 9: For any binary set operator *setop*, a binary sandbag operator *setop'* can be generated by:

- $\underline{\mathsf{mina}}.(P \ setop' \ Q).x =$
  $(\mathbf{min} \ B, C: \underline{\mathsf{sbcons}}.P.B \ \wedge \ \underline{\mathsf{sbcons}}.Q.C :$
  $\underline{\mathsf{numa}}.(B \ \ setop \ \ C).x)$
- $\underline{\mathsf{maxa}}.(P \ setop' \ Q).x =$
  $(\mathbf{max} \ B, C: \underline{\mathsf{sbcons}}.P.B \ \wedge \ \underline{\mathsf{sbcons}}.Q.C :$
  $\underline{\mathsf{numa}}.(B \ \ setop \ \ C).x) \ \square$

A similar but simpler definition suffices for unary operators.

[1] Maier[4] provides a terminology for partial data that defines the terms *faithful, precise, adequate,* and *restricted.* In the language of [4], a *sound* operator is *faithful,* and a *complete* operator is *adequate* and *restricted.* Property 2 does not imply *precision* in that terminology.

Definition 10: For any unary set operator *setop*, a unary sandbag operator *setop'* can be generated by:

- $\underline{\text{mina}}.(setop'.P).x =$
  $(\textbf{min}\ B : \underline{\text{sbcons}}.P.B : \underline{\text{numa}}.(setop.B).x)$
- $\underline{\text{maxa}}.(setop'.P).x =$
  $(\textbf{max}\ B : \underline{\text{sbcons}}.P.B : \underline{\text{numa}}.(setop.B).x)$ □

Any sandbag operator generated from Definition 9 or Definition 10 is well-defined if its arguments are externally consistent, so that the sets quantified over by **min** and **max** are non-empty.

Theorem 1: Any sandbag operator *setop'* generated from a set operator *setop* with either Definition 9 or Definition 10 has the Properties of Soundness, Completeness, Monotonicity, and Totality Preservation. □

In particular $\cap'$, $\cup'$, $-'$, $\times'$, σ', π', and $\bowtie'$ have the properties of Soundness, Completeness, Monotonicity and Totality Preservation.

Although these definitions are not constructive, nevertheless there exist reasonably efficient implementations of these operations that sacrifice the Completeness Property to gain speed [22]. Without Completeness, the bounds provided by mina and maxa are not guaranteed to be the best that can be derived; however, they are guaranteed to be consistent.

5.2 The Multi-resolution data model

The MR relational operators ($\cap'$, $\cup'$, $-'$, $\times'$, σ', π', and $\bowtie'$) form a query language over MR relations analogous to the relational algebra. We have defined these operations so that they are strictly more general than and completely consistent with the relational operators. This fact is expressed by Theorem 1, that is used to show that these operators on sandbags form a language that is a generalization of the relational algebra. We call this the *multi-resolution relational algebra*, although the algebraic laws of the relational algebra hold only in modified forms for our language.

Consider a relational database Y consisting of N relations $Y = \{R_0, R_1, R_2, ...R_{N-1}\}$. A corresponding *multi-resolution relational database* Z consists of N sandbags $Z = \{S_0, S_1, S_2, ...S_{N-1}\}$. We say an MR relational database Z *approximates* a relational database Y if and only if $(\forall\, i : i \geq 0 \wedge i < N : \underline{\text{sbcons}}.S_i.R_i)$. Note that the quality of Z is not specified; it may contain all of the information of Y (i.e., be total) or very little. A relational expression E over Y is a tree of relational operators that uses relations from Y. The corresponding *multi-resolution relational algebra* expression, denoted E', is an operator-for-operator substitution of sandbag operators for relational operators and sandbags from Z for relations from Y. The result of a query expression on an MR database is defined to be the result of applying the MR operators to the MR relations.

We extend the definitions of Soundness, Completeness, Monotonicity and Totality Preservation for individual operators to analogous properties on multi-resolution query languages, such as the MR relational algebra. We denote the result of applying an expression E to a database Y by $E.Y$. Let Z be an MR database approximating a relational database Y. For a query language we define the following four Query Language (QL) Properties:

Property 5 (QL Soundness): A query language is *sound* if and only if $\underline{\text{sbcons}}.(E'.Z).(E.Y)$, for every expression E. □

Property 6 (QL Completeness): A query language is *complete* if and only for every expression E, E' on Z always provide the tightest bounds consistent with Y.

$$(\forall x :: (\exists Y : Z \text{ approximates } Y : \underline{\text{numa}}.(E.Y).x = \underline{\text{mina}}.(E'.Z).x) \wedge (\exists Y : Z \text{ approximates } Y : \underline{\text{numa}}.(E.Y).x = \underline{\text{maxa}}.(E'.Z).x))$$ □

Property 7 (QL Monotonicity): A query language is *monotonic* if and only if any improvements in the information of the sandbags of Z do not make E' worse, for any query E. If Z and X are MR databases, $Z = \{S_0, S_1, S_2, ...S_{N-1}\}$ and $X = \{P_0, P_1, P_2, ...P_{N-1}\}$:

$$(\forall\, i : i \geq 0 \wedge i < N : S_i \sqsubseteq P_i) \Rightarrow E'.Z \sqsubseteq E'.X$$ □

Property 8 (QL Totality Preservation): A query language is *totality preserving* if and only if every expression is total ($\underline{\text{tot}}.(E'.Z)$) when every sandbag in Z is total. □

Theorem 2: The multi-resolution relational algebra has the Query Language Properties of Soundness, Monotonicity, and Totality Preservation, *but not Completeness.* □

Unfortunately, the QL Completeness Property does not hold for the multi-resolution relational algebra. A counter-example to completeness can be constructed from queries that mention the same relation more than once. If P is a non-total sandbag approximating a relation R in a database, the MR query expression $P\ -'\ P$ yields a non-total sandbag as a result that is consistent with non-empty sets.

This proves the MR query language is incomplete, and is an example of an algebraic law that fails to hold in unmodified form for the MR query language.

Incompleteness is a disadvantage that appears to be a fundamental problem of approaches that do not preserve identity, such as [6]. [4, 7, 12] discuss this problem. We choose to tolerate this disadvantage, for two reasons. Most importantly, any implementation based on our approach to constructing sandbags described in [22] must immediately sacrifice the Completeness Property (Property 2), because calculating the optimal mina and maxa values is NP-hard for that particular approach. Secondly, we suspect that this incompleteness has little impact on our overall purpose of supporting the resolution/performance tradeoff. A significant difference between our approach and most previous work on partial data models is our motivation of high performance rather than clean semantics for incomplete data.

5.3 Multi-resolution Queries and Progressive Refinement

A query in the multi-resolution relational algebra is essentially a relational query. This has the advantage that queries are easily understood, and the choice of response resolution is orthogonal to the formulation of the query.

This data model admits the concept of *progressive refinement* that is not available in the standard relational model. In both models, a query has a single, well-defined, *result*. However, in the MR model this query result has many lower-resolution approximations, that we call *responses*. We define P to be a response to a query E' on an MR database Z if and only if $P \sqsubseteq E'.Z$. Both the notions of *response* and *result* are mathematically defined based on the MR data model.

It is the low-resolution responses to queries that are the focus of our work, rather than the highest-resolution results, because our fundamental goal is to improve performance by allowing resolution of database query answers to be traded for speed. Furthermore, the idea of progressive refinement can now be made precise. A DBMS *progressively refines* the answer to some query when it produces a chain of responses, each of which approximates the result of the query and improves upon the last response. Because the responses to a query are sandbags and sandbags are partially ordered, we can depict the responses to a query diagrammatically, as in Figure 4.

We have just begun research into the implementation of the MR data model (See Section 6.) For the sake of explanation, we now present an example of progressive refinement as if a working MR DBMS exists.

Example 3: Consider an MR database Z consisting of a relation r of the scheme $r(id, pic)$, where id is of type integer (not a multi-resolution type) and pic is the raster type describe in Example 2. Consider the query $Q = \sigma_{id \geq 100 \wedge id \leq 150} r$, and suppose 10 tuples of r satisfy this selection predicate.

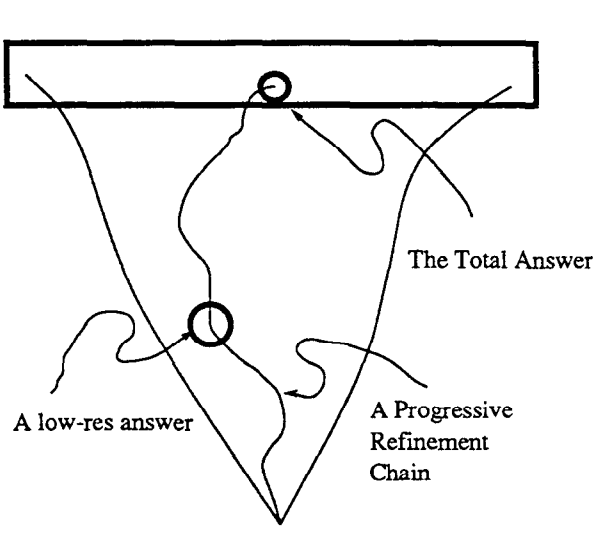

Figure 4: Progressive refinement of multi-resolution queries.

Suppose that rasters are stored in the DBMS in the following manner (either by the database programmers design or by the choice of the MR DBMS). Eight of the 10 scenes depicted by $Q.Z$ have level 2 approximations resident in main memory, all images in Z have a level 3 raster approximation stored on magnetic disk, and the total rasters (level 4) are stored on (relatively slow) CD-ROMs.

In response to Q, a sandbag S containing 8 level 2 rasters will be delivered to the user (perhaps a program that draws on a screen) in milliseconds. The MR DBMS then retrieves the 10 level 3 rasters from magnetic disk, and places them, one at a time, in S. In terms of the partial order of sandbags, $S \sqsubseteq Q.Z$ and S is being progressively refined upward. After about 5 seconds the retrieval of level 4 rasters is begun. Perhaps 60 seconds later the sandbag S is total, and $S = Q.Z$. □

The query in Example 3 may be terminated after a specific resolution is materialized to obtain a low-resolution response relatively quickly. (More powerful means of specifying response resolution in the query are the subject of future work.)

In this example, the full expressive power of the sandbag to represent cardinalities is not being employed. The sandbag S serves as a container of multi-resolution objects. The sandbag will often be used in this less than fully general role. Since id is not an MR type, Example 3 does not utilize the MR algebra in

an interesting manner, although it demonstrates what we consider the most important immediate use of an MR DBMS. Example 4 partially illustrates a multi-resolution join.

Example 4: Consider two relations that represent the positions of objects $(r(X,Y,A), p(X,Y,B))$. In this schema the attributes X and Y are of the real interval type defined in Example 1, and represent position on a grid. A and B are identifiers. The natural join query $Q = r \bowtie p$ calculates the objects in r and p that share positions. Let the relations r and p be represented by the sandbags R' and P'. Let the sandbag S be the response to the MR query $R' \bowtie' P'$. The schema of S is (X,Y,A,B).

Let us suppose that sandbags R' and P' are refined over time, either as their tuples are read or as objects report their positions with increasing precision. First, suppose it is known only that $|r| = 100$ and $|p| = 100$. Then $\underline{\text{maxa}}.S.\perp$ is 10,000. Suppose then that the exact position of some objects $a \in p$ and $b \in r$ are discovered. If these objects are in the same position, then $\underline{\text{mina}}.S.\langle[-\infty,+\infty],[-\infty,+\infty],a,b\rangle = 1$. If it is then determined that there are no objects with an X position greater than 100.0, $\underline{\text{maxa}}.S.\langle[100,+\infty],[-\infty,+\infty],\perp,\perp\rangle = 0$. As more precise information is added to R' and P', the $\underline{\text{mina}}$ and $\underline{\text{maxa}}$ values of S get closer, until S is total. □

Fundamentally little can be inferred about a join until the join attributes of its arguments are relatively well-defined. This example suggests a computational expense to implement the sandbag operations, undoubtably limiting their usefulness. However, we believe that by sacrificing completeness the sandbag can be implemented efficiently enough to be practical for some applications, even when used as in Example 4.

6 Conclusions and Future Work

This paper introduces the *sandbag*, an expressive construct for modeling imprecise and uncertain information about a set. Based on the sandbag, we develop a multi-resolution data model that forms a basis for building a useful database management system that offers the performance advantages of multi-resolution. The data model provides a sound theoretical framework and makes precise the notions of *multi-resolution* and *progressive refinement*, that we have motivated. We hope that the similarity to the relational data model leverages the technology and widespread understanding of that model, both for the reader's understanding and to construct an efficient implementation. However, the idea of multi-resolution and the sandbag construct are applicable to other data models, such as object-oriented ones.

We are in the process of developing efficient algorithms to implement the sandbag and the sandbag operations [22]. At least two other goals must be met if a useful multi-resolution database management system is to be constructed.

Foremost, a query language extension must be developed that allows the user to conveniently specify the desired resolution of the response to a query. We plan to extend the MR relational algebra by allowing queries to be annotated with resolution/performance constraints.

Secondly, special algorithms, data structures, and storage structures that exploit the lower-bandwidth requirements of low-resolution data should be utilized systematically. These include a cache storage structure that directly utilizes the size relationship between resolution levels to manage storage of differing speed to provide good performance across the entire spectrum of resolutions. The quad tree representation of raster images is an inherently multi-resolution data structure. Structures such as a bit-striped transposed representation of sets of ordered values [31, 32] may allow multi-resolution to be applied to common, built-in types more efficiently.

Acknowledgements

Comments by the referees and the UT database research group have greatly improved this paper.

References

[1] F. Bancilhon and S. Khoshafian. Calculus for complex objects. In *PODS*, pages 53–59, 1986.

[2] P. Buneman, S. Davidson, and A. Watters. A semantics for complex objects and approximate queries. In *Seventh Symposium on the Principles of Database Systems*, pages 305–314, March 1988.

[3] J. M. Morrissey. Imprecise information and uncertainty in information systems. *ACM Transactions on Information Systems*, 6(2):159–180, 1990.

[4] D. Maier. *The Theory of Relational Databases.* Computer Science Press, 11 Taft Court, Rockville Maryland, 20850, 1983. Chapter 12.

[5] S. V. Vrbsky and J. W. S. Liu. An object-oriented query processor that returns monotonically improving answers. In *Proceedings of the 7th IEEE Conf. on Data Engineering*, Kobe, Japan, April 1991.

[6] K. Liu and R. Sunderraman. On representing indefinite and maybe information in relational databases. In *Proceedings: The Fourth International Conference on Data Engineering*, pages 250–257, February 1988.

[7] T. Imielinski and W. Lipski. On representing incomplete information in a relational database. In *7th International Conference on Very Large Data Bases*, pages 389–397, Cannes, France, September 1981.

[8] T. Imielinski and W. Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–790, October 1984.

[9] R. Reiter. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *Journal of the ACM*, 33(2):349–370, April 1986.

[10] J. Grant. Partial values in a tabular database model. *Information Processing Letters*, 9(2):92–99, 1979.

[11] L. G. DeMichiel. Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *Transactions on Knowledge and Data Engineering*, 1(4):485–493, December 1989.

[12] L. G. DeMichiel. *Performing database operations over mismatched domains.* PhD thesis, Stanford University, Stanford, CA, 1989.

[13] P. Buneman, A. Jung, and A. Ohori. Using powerdomains to generalize relational databases. *Theoretical Computer Science*, pages 23–55, 1991.

[14] A. Silberschatz, M. Stonebraker, and J. Ullman. Database systems: Achievements and opportunities. *Communications of the ACM*, 34(10):110–120, October 1991.

[15] W. E. Huxhold. *An Introduction to Urban Geographic Information Systems.* Oxford University Press, 200 Madison Avenue, New York, NY 10016, 1991.

[16] Personal communication with Dr. Anita Cochron about the Halley Watch Archive CD image library.

[17] A. Turtur, F. Prampolini, M. Fantini, R. Guarda, and M. A. Imperato. Idb: An image database system. *IBM Journal of Research and Development*, 35(1/2):88–96, January/March 1991.

[18] L. Bergman, H. Fuchs, E. Grant, and S. Spach. Image rendering by adaptive refinement. *Computer Graphics*, 20(4):29–38, August 1986.

[19] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics*, 22(4):75–84, August 1988.

[20] F. S. Hill, Jr., S. Walker, Jr., and F. Gao. Interactive image query system using progressive transmission. *Computer Graphics*, 17(3):323–330, July 1983.

[21] R. L. Read, D. S. Fussell, and A. Silberschatz. A multi-resolution relational data model. Technical Report TR-92-10, University of Texas at Austin, Department of Computer Sciences, Austin, TX 78712-1188, 1992.

[22] R. L. Read, D. S. Fussell, and A. Silberschatz. The sandbag construct: a method for imprecise set representation. In preparation.

[23] J. E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* MIT Press, Cambridge, Massachusetts, 1977.

[24] H. L. Resnikoff. *The Illusion of Reality.* Springer-Verlag, 1989.

[25] R. W. Hamming. *Coding and Information Theory.* Prentice-Hall, Englewood Cliffs, NJ, 07632, 1986.

[26] R. Hull. Relative information capacity of simple relational database schemata. *SIAM Journal of Computing*, 15(3):856–886, August 1986.

[27] C. Shum and R. Muntz. An information-theoretic study on aggregate responses. In Francois Bancilhon and David J. DeWitt, editors, *14th International Conference on Very Large Data Bases*, pages 479–490, Los Angeles, USA, August 1988.

[28] V. Lessen, J. Paulin, and E. Durfee. Approximate processing in real-time problems solving. *AI Magazine*, 9:49–61, March 1987.

[29] K. J. Lin, S. Natarajan, and J. W. S. Liu. Imprecise results: utilizing partial computations in real-time systems. In *Proceedings of the IEEE 8th Real-Time Systems Symposium*, San Jose, California, December 1987.

[30] K. B. Kenney and K. J. Lin. Structuring large real-time systems with performance polymorphism. In *Proceedings of the IEEE 11th Real-Time Systems Symposium*, pages 238–246, Orlando, Florida, December 1990.

[31] D. S. Batory. On searching transposed files. *ACM Transactions on Database Systems*, 4(4):531–544, December 1979.

[32] H. K. T. Wong, H. Liu, F. Olken, D. Rotem, and L. Wong. Bit transposed files. In A. Pirotte and Y. Vassiliou, editors, *11th International Conference on Very Large Data Bases*, pages 448–456, Stockholm, Sweden, August 1985.