K-Anonymization as Spatial Indexing: Toward Scalable and Incremental Anonymization

Tochukwu Iwuchukwu
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706
tochukwu@cs.wisc.edu

Jeffrey F. Naughton
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706
naughton@cs.wisc.edu

ABSTRACT

In this paper we observe that *k*-anonymizing a data set is strikingly similar to building a spatial index over the data set, so similar in fact that classical spatial indexing techniques can be used to anonymize data sets. We use this observation to leverage over 20 years of work on database indexing to provide efficient and dynamic anonymization techniques. Experiments with our implementation show that the R-tree index-based approach yields a batch anonymization algorithm that is orders of magnitude more efficient than previously proposed algorithms and has the advantage of supporting incremental updates. Finally, we show that the anonymizations generated by the R-tree approach do not sacrifice quality in their search for efficiency; in fact, by several previously proposed quality metrics, the compact partitioning properties of R-trees generate anonymizations superior to those generated by previously proposed anonymization algorithms.

1. INTRODUCTION

The problem of anonymity in published data has been widely studied in recent years. Organizations may release private data for the purposes of facilitating useful data analysis and research, for example, patients' medical records may be released by a clinic to aid a medical study. While such data sharing has its benefits, we must, however, contend with the issue of privacy for those individuals to whom information in the shared data pertain. *K-anonymity* [24, 25, 26, 29, 30] has been proposed as a means to preserving privacy in data releases. Put simply, the private data set is modified so that each record is indistinguishable from at least $k-1$ other records. Indistinguishability is defined in terms of any set of attributes that can be used to uniquely identify an individual. This set of attributes has been called a *quasi-identifier* [7] in the literature. An example of a quasi-identifier is the set of attributes comprising *Age, Sex* and *Zipcode* [28]. Figure 1 illustrates how private data can be transformed to preserve anonymity. The 2-anonymous table in Figure 1(b) has three quasi-identifier attributes: Age, Sex and Zipcode and one sensitive attribute: Ailment. Each record in this table has the same quasi-identifier values as at least one other record.

Since the original definition of *k*-anonymity, there has been a great deal of research along two orthogonal lines: first, how to refine the definition of *k*-anonymity to provide different guarantees about the data (for example, augmenting the definition with *l*-diversity [21]); second, how to efficiently generate anonymizations of data sets that are as precise as possible while still respecting the definition of anonymity (for example, using heuristic algorithms with multidimensional partitioning [19].) Our work is an example of the second class of research.

We present an anonymization algorithm that substantially improves upon previously presented algorithms both with respect to efficiency and with respect to the quality of the anonymization produced. Specifically, the algorithm we present in this paper allows us to anonymize data sets containing at least 100 million records; also, by a recently presented metric for the quality of anonymization ("certainty" [33]), the anonymizations produced by our algorithm are approximately a factor of two better than previous algorithms.

The key to our algorithm is to exploit a striking parallel between the "classical" area of database indexing and the relatively new data privacy research domain, *k*-anonymity. Let T be a table with n quasi-identifier attributes $A_1, A_2, \ldots, A_n$. First, we observe that the eventual goal of all *k*-anonymization algorithms is to transform T by partitioning T into groups of records so that each group contains a minimum of k records. To illustrate the connection between indexing and anonymization, assume that BT is a B$^+$-tree index on the quasi-identifier attribute A_1. Note that every path from the root node in BT to a leaf node L produces a set of records in T whose A_1 values satisfy the constraint imposed by the path followed to reach L. Note then that the A_1 values for the records contained in L fall in the range [a, b] where a and b are the left and right separator values in L's parent node that border the "pointer" to L. When we apply this concept to every leaf node in BT, we can transform T into a new table T_1 by replacing every record's value in A_1 by the appropriate range of values — records in the same leaf node will have the same new A_1 value. Going one step further, a B$^+$-tree index places an occupancy constraint on all nodes in the tree, as such every leaf node in BT must contain between N_{min} and N_{max} records. With these properties of BT, namely, an implicit partitioning of the underlying table and a bounded occupancy constraint on all partitions, we get a "*k*-anonymous" table where $k = N_{\text{min}}$. Figure 1(c) shows an example of a B$^+$-tree index on the Age attribute of the original table in Figure 1(a) and corresponding to the 2-anonymous table in Figure 1(b).

In general, however, the table to be published may contain more than one quasi-identifier attribute, so rather than use B$^+$-trees, we suggest multidimensional spatial indexing, and the R-tree in particular, as the basis for anonymization. In Section 2, we discuss in greater detail how a variant of R-tree indexes provide algorithms for

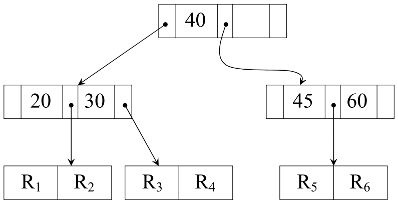
VLDB '07, September 23-28, 2007, Vienna, Austria.

RID	Age	Sex	Zipcode	Ailment
R_1	21	M	53706	anemia
R_2	26	M	53706	flu
R_3	32	F	53710	cancer
R_4	36	F	53715	torn acl
R_5	48	M	52108	flu
R_6	56	F	52100	whiplash

(a) Private Patient Table

Age	Sex	Zipcode	Ailment
[20 – 30]	M	53706	anemia
[20 – 30]	M	53706	flu
[30 – 40]	F	[53710 – 53715]	cancer
[30 – 40]	F	[53710 – 53715]	torn acl
[45 – 60]	*	[52100 – 52108]	flu
[45 – 60]	*	[52100 – 52108]	whiplash

(b) A 2-anonymous Patient Table



(c) B$^+$-tree index on the Age attribute

Figure 1: A 2-anonymous representation and database indexing of a patient table.

k-anonymizing a multi-attribute table. We discuss R-tree style indexes [5, 10, 27] and how these indexes are useful for *k*-anonymizing a data set, and properties that give them advantages over other types of indexing mechanisms. In particular, we focus on the variants of the R-tree that do not overlap partitions, for example, the R$^+$-tree [27]. This is due to the universally adopted practice in existing *k*-anonymization algorithms of generating only non-overlapping partitions in the anonymized data.

The connection between anonymization and spatial indexing is perhaps not entirely surprising, as [22] used a new special-purpose spatial index structure (the "pyramid tree") to anonymize objects moving in the spatial domain. However, to the best of our knowledge, ours is the first work to propose anonymization of non-spatial data by the use of a classical spatial index that is already implemented and distributed in commercial and open-source RDBMS products.

The indexing-anonymizing connection gives us a different perspective in the *k*-anonymization domain, has several advantages over previously proposed *k*-anonymization algorithms, and unifies several desired goals for anonymization into a single approach:

- An R-tree index-based approach to *k*-anonymization furnishes us with efficient index-construction algorithms that enable faster bulk anonymization times than previous techniques, even for memory-resident data sets.

- We further show that applying R-tree bulk-loading algorithms to anonymizing yields anonymization algorithms that perform well even on data sets much larger than main memory. This enables us to anonymize a data set of 100,000,000 records.

- We observe that minimal bounding boxes from the indexing domain [5, 10, 27] suggest anonymizations that leave gaps in the domain. This can yield far more precise anonymizations than previously proposed anonymization techniques, none of which consider leaving gaps. This opens up an interesting and novel aspect of the always-present tension between anonymization and precision that has not been previously explored in the *k*-anonymization literature.

- Spatial indexes are well-suited to exploit anticipated workloads while anonymizing data sets. Selecting specific quasi-identifier attributes on which to build an index and biased splitting algorithms are two ways that we can incorporate query workloads into the anonymization.

- A database owner may wish to distribute anonymized tables of different "granularity" to separate groups, reflecting her trust. For example, she may deliver a 5-anonymization of her table to a medical research group while delivering a 10-anonymous version to an insurance research group. Rather than re-anonymize the original table for each group, facing the danger of privacy violation in the presence of collusion, we exploit the tree structure of a spatial index for automatic generation of *multi-granular* anonymized data sets that preserves *k*-anonymity.

Since database indexes are specifically designed for record insertions, deletions and updates, by using them for anonymization, we automatically get a mechanism for incremental anonymization. However, incremental anonymization raises issues with respect to the preservation of privacy. If an attacker has external knowledge of which individual's records are being inserted, deleted or updated in a data set, then the attacker may be able to issue a series of queries over time and deduce sensitive information. While providing an incrementally updatable anonymization technique does not solve the inference problem, it is a much better platform for updates than current techniques, which could potentially require re-anonymization of the entire data set after each update.

Finally, the index-based approach to anonymization can exploit the efficiency inherent in index update and bulk-loading algorithms. Previous research in *k*-anonymizing algorithms has focused almost exclusively on the quality of the resulting anonymization, rather than on the speed with which that anonymization is achieved. An exception is the Mondrian algorithm from [19], where the authors present a polynomial time algorithm, thus making it practical to consider anonymizing large data sets. While absolute performance was not the goal of that paper, it is interesting to note that the approach suggested in that paper constitutes a top-down multidimensional spatial partitioning algorithm, whereas spatial index building algorithms represent a bottom-up spatial partitioning approach.

To investigate the quality and efficiency of both approaches, we reimplemented the Mondrian algorithm described in [19], and compared it to bottom-up index-based algorithms. We found that the bottom-up approach gave better quality as measured by the discernibility penalty [4], KL divergence [15] and the "certainty metric" [33]. Furthermore, experiments with our implementation also showed that the bottom-up approach adopted by index bulk-loading algorithms is an order of magnitude faster than the top-down Mondrian approach. It is an interesting area for future research to determine whether this is a fundamental property of all top-down vs. bottom-up approaches.

Note that our use of "top-down" vs. "bottom-up" methods differs from the usage in [33], where they use the terms to refer to two new $O(n^2)$ algorithms. While we did not have access to their code in order to do a comparison of these new algorithms with ours, as the authors of that paper note, it is not as efficient as the Mondrian algorithm (6X slower on 100,000 records), and it will not scale to large n (no n^2 algorithm can). Our experiments show that our bottom-up algorithm scales well at least for another factor of 1000 (up to 100 million records.) The issue of quality of the anonymizations produced is less clear, since while that paper reports better results than the Mondrian algorithm, it does not consider anything like our "compaction" procedure, which we found essential for good certainty metric scores.

2. R-TREE SPATIAL INDEXING

R-tree spatial indexes bring with them several desirable properties when applied to the problem of *k*-anonymization.

2.1 Scalability to Large Data Sets

To date, the *k*-anonymization literature has not considered algorithms for anonymizing data sets that do not fit in memory. Bulk-loading database indexes has almost by definition focused on such data sets. A number of bulk-loading techniques have been proposed for spatial indexes. Some of these techniques require spatial sorting based on space-filling curves [12, 13, 14] (e.g, the Hilbert curve or Z-ordering). While we experimented with such approaches, we found in our implementation that non-sorting bulk-loading techniques based on the "buffer-tree" [2, 6] worked better for higher dimensional data sets.

The buffer-tree is based on the idea of inserting multiple records simultaneously into the tree. Each internal node of the tree has an external buffer where records are temporarily stored. Multiple insertions are processed in the following way. An index node keeps and "blocks" arriving insertions in its buffer. When the number of records in the buffer exceeds a pre-defined threshold, all of the records are "re-activated" and advanced to the next level of the tree. Records are "terminated" when they are inserted into a leaf page. Figure 2 shows an example of a buffer-tree after a series of insertions have been processed. The buffer-tree consists of three nodes N_1, N_2, N_3 and five leaf pages $P_1, \ldots, P_5$. Assume that node buffers contain at most two pages and that a page has a maximum capacity of three records. Consider the insertion of record r_{25}. Since the root buffer is full, the insertions of the six records in the root buffer are "re-activated" and "pushed down" to the next level of the buffer-tree. After clearing a buffer, it may happen that buffers at the next level also become full. These overflows are again eliminated by clearing these buffers.

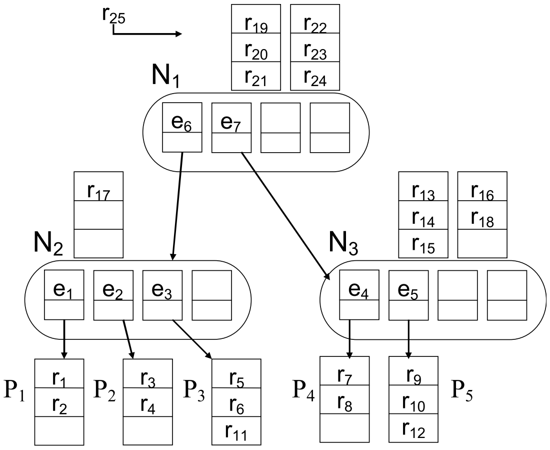


Figure 2: Example buffer-tree after inserting 24 records, a full root buffer and record r_{25} waiting to be inserted

A feature of the buffer-tree is that insertions traverse the tree from root to leaf while restructuring operations traverse the tree from the leaf backwards to the root. A restructuring operation consists of a split of an overflowing node (a node whose buffers are full) and an insertion of a new entry in its parent node. A restructuring operation is first triggered by the split of a full leaf page. Just as for record insertions, multiple restructuring operations are also processed simultaneously — an internal node defers an incoming insertion of an entry. When all subtrees of the node have finished their restructuring operations, the entries are then stored in the routing table of the node. This may again produce overflow and further restructuring operations. Figure 3 shows an example of a restructuring process after the records in the root buffer are cleared into the buffers of nodes N_2 and N_3.

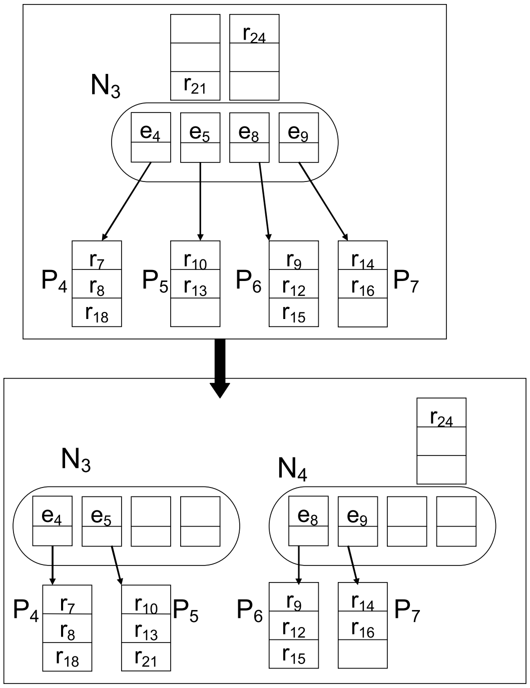


Figure 3: Splitting of the index node N_3 after clearing the root buffer

We can gain insight into the performance of the buffer tree algorithm by assuming an I/O model with the following parameters:

N = number of records in input data set

M = maximum number of records that the available memory can hold

B = maximum number of records that a page can hold

The authors in [6] show that the I/O cost for bulk-loading a buffer-tree for a data set of *N* records is $O(N/B \log_{M/B} N/B)$. Thus, buffer-trees achieve similar I/O cost bounds to external sorting. We also expect buffer-trees to have "good" performance when the input data set fits in memory. The buffer-tree amortizes the cost of inserting a set of records by deferring operations on the tree. This contrasts the tuple-loading approach that inserts records one by one into the index typically resulting in long load times (for example, each newly inserted record may cause a node split that increases the height of the tree index.)

Our experiments show reasonable performance for the buffer-tree algorithms in *k*-anonymizing larger-than-memory as well as memory-resident data sets using a spatial index.

2.2 Incremental Utility

In a dynamic environment, the spatial index is a natural mechanism for allowing changes to be made on a data set while maintaining a *k*-anonymous view. The prior anonymization algorithms in the literature work on a data set as a complete whole — they start with the complete (non-anonymized) data set as input, and produce a complete anonymized data set as output. If an anonymized data set is to be updated using these algorithms, the only option is to re-anonymize the original data set plus the new data records. This will be inefficient in scenarios with frequent updates.

Again, database indexes are designed to be incrementally updated. One concern that arises is whether using the incremental anonymization that results from processing updates one record at a time using an R-tree is of worse quality than one that would result from anonymizing the entire data set at once. Our experiments show that this is not the case — the incrementally updated anonymized data set has quality (measured by discernibility penalty, KL-divergence and certainty penalty) comparable to that of the bulk-anonymized data set.

2.3 Query Performance

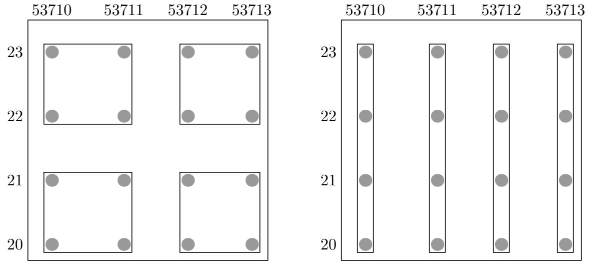
In a traditional database setting, the performance of a query on an index is typically determined by factors such as the time it takes to execute the query or how many nodes need to be searched to find all records that satisfy this query. For k-anonymity, we associate query performance with the number of records that are included in the answer that would not satisfy the same query on the original data. This is similar in spirit to the "precision" metric used in the Information Retrieval literature.

Denote W to be the set of leaf nodes (or partitions) in a database index that is searched due to a query Q_1 posed on the original data. Let Q_2 denote the same query on the anonymized data. Note that if a partition P is contained in W, then P is a candidate partition that *may* contain a record that satisfies Q_1. If $P \notin W$, then it is certain that P does not to contain any record that satisfies Q_1. Consider, for example, the following query on a table T:

```
SELECT COUNT(*)
FROM T
WHERE T.Age ≥ 25 AND T.Age ≤ 35
```

If the age interval for P is given as [40 – 50] then P will not be included in W. On the other hand, if the age interval is [20 – 30] then P will be included in W. Note that it is still possible that P does not contain any record that satisfies the query. The age values in P may actually range from 20 to 24. Nevertheless, since we have precise record values in the unanonymized data, the candidate partitions will be examined and the relevant records returned. On the other hand, the query Q_2 on anonymized data may return all records in W since we do not have exact information in this case. We define the error for Q_2 as $C_2 - C_1/C_1$ where C_1 and C_2 are the cardinalities of the result sets for Q_1 and Q_2 respectively. Intuitively, the error for Q_2 can be reduced if W contains fewer partitions. The entries in R-tree style indexes are maintained in minimum bounding rectangles (MBRs), giving the minimal extents of its entries. MBRs allow search and range queries on a spatial data set to be executed efficiently. By using MBRs, we increase the likelihood that a partition will not be included in W. In the example given above, using MBRs, the range on the Age attribute for P will be [20 – 24] thus P will not be included in W.

We note that the exact behavior of queries on anonymized data may differ for different applications. One may choose to take the data distribution into consideration when computing query results. Denote P.Age and Q.Age as the age interval for the partition P and query Q respectively. Denote $(P \cap Q)$.Age to be the intersection of the age intervals on P and Q. If Q.Age is $(25 \leq \text{age} \leq 35)$ and P.Age = [30 – 40] then $(P \cap Q)$.Age = [30 – 35]. Now, if we assume that the original data set is uniformly distributed on Age then we may compute the result for Q on P as $|P| \times |(P \cap Q).\text{Age}|/|P.\text{Age}|$. If P contains 10 tuples, then the query result for P is $10 \times (35 - 30)/(40 - 30) = 5$ tuples.



(a) Unbiased R^+-tree anonymized data

(b) Biased R^+-tree anonymized data

Figure 4: Targeting the R^+-tree to the Zipcode quasi-identifier attribute of a data set

However, regardless of the method used to evaluate query results, these results must be computed based on the set of all partitions, the set W, that may contain a satisfying record.

2.4 Query Workload Bias

Recent works [9, 11, 20, 31] have considered incorporating target query workloads into the anonymization of a data set. We can also tailor a spatial index to take advantage of advanced knowledge of the types of workload that will use the anonymized data. Consider a very simple scenario where majority of the data mining workloads are interested in the single quasi-identifier attribute A_0. If we build the index on A_0 (a one-dimensional index), then the k-anonymized data is clustered on A_0. Under a sort-based bulk-loading scheme, this results in sorting the data set on A_0. For our non-sort spatial index bulk-loading technique, preference is given to a pre-selected subset S of the quasi-identifier attributes when splitting partitions.

Consider the two sets of anonymized data in Figure 4. A query of the form

```
SELECT COUNT(*)
FROM T
WHERE T.Zipcode = z
```

will return more accurate results on the anonymized data generated by the biased R^+-tree (see Figure 4(b)) than identical queries on anonymized data generated by the unbiased R^+-tree (see Figure 4(a)). For the query type given above and the example data in Figure 4, queries on the biased anonymized data in Figure 4(b) will be twice as accurate as queries on the unbiased anonymized data in Figure 4(a) (assuming that a `COUNT` query on a partition returns the cardinality of that partition if the query region intersects with the partition). We find, in experiments, that when we build the R^+-tree with biased splitting policies for a subset S of the quasi-identifier attributes, queries on S have relatively higher accuracies than queries on anonymized data generated by another R^+-tree with an unbiased algorithm. Spatial indexes can hence be useful tools for anonymizing data sets when the types of query workloads that will use the data are known beforehand. Taking a cue from [33] that proposes a weighted certainty penalty metric, a spatial index can also incorporate query workloads into its splitting policies by assigning higher weights to the "more important" quasi-identifier attributes. As a consequence, it benefits the spatial index to split the more important attributes than the less important ones to arrive at a lower penalty score for the new partitions.

3. MULTI-GRANULAR K-ANONYMITY

While a primary goal of the privacy-preserving techniques in k-anonymity is to prevent re-identification of records, it may also be useful to control the precision or *granularity* of the information that the data owner releases to different entities, to limit the linking abilities of unknown adversaries.

DEFINITION 1. (GRANULARITY) *We say that a k-anonymous data set is an anonymous data set of granularity k.*

Suppose the administrators of a university hospital have agreed to deliver anonymized medical records to the following three entities; Entity 1: researchers at the same university as the hospital, Entity 2: researchers at a different university, Entity 3: the Internet. One may expect that the hospital administrators place more "trust" in Entity 1 than in Entity 2, Entity 3 being the least trusted. The notion of trust is subjective and may be associated with factors such as the data owner's perception of the target entity's ability or intent to re-identify records (Data released on the Web is probably more likely in danger of being compromised than data released to a small, local group of researchers.)

We wish to be able to produce multi-granular anonymizations $T^{'} = \{T_1^{'}, \dots, T_n^{'}\}$ of the original data set T while preserving k-anonymity for every individual in T in the presence of an adversary who is able to gain access to more than one anonymized table. Note that each $T_i^{'}$ is an anonymization of the same table T, that is, T is unchanged between anonymizations. While a full discussion of the inference problem that arises from releasing multiple anonymizations of the same data set is beyond the scope of this paper, in what follows we give a condition that, if satisfied, guarantees k-anonymity for any set of multi-granular anonymizations $T^{'}$ of T.

DEFINITION 2. (k-BOUND) *We say that the record r_i is k-bound in the original table T if there exists a subset of records $R \subseteq T$ such that $|R| \geq k$ and given any partition or equivalence class P in an anonymization of T, if $r_i \in P$, then $R \subseteq P$.*

LEMMA 1. *Let $T^{'} = \{T_1^{'}, \dots, T_n^{'}\}$ be a set of k_i-anonymizations of the table T, $k_i \geq k$. k-anonymity is preserved over $T^{'}$ if every record $r_j \in T$ is k-bound in T.*

PROOF. (SKETCH) Given any k_i-anonymization $T_i^{'}$ of T, r_j can not be linked to fewer than k records ($k_i \geq k$) using $T_i^{'}$ alone. An adversary may however be able to circumvent k-anonymity for r_j using the set of anonymizations $T^{'}$ to narrow down the candidate records for r_j to a set C containing fewer than k records.

Pick any record $r_j \in T$, for any new anonymization of T, the adversary may be able to produce a new set of candidate records $C^{'}$ for r_j such that $|C^{'}| \leq |C|$ where C is the previous set of candidate records for r_j as determined by the adversary. We know that $|C^{'}| \geq 1$ since $C^{'}$ must at least contain r_j. If r_j is k-bound in T, then R is always a candidate set for r_j, $R \subseteq C^{'}$, $|C^{'}| \geq k$. Thus correlating any new anonymization of T will result in at least k records being candidates for r_j. □

One may view a record that is k-bound in a table as being in a group that always "sticks together" in any new k-anonymization of that table.

3.1 A Hierarchical Algorithm for Generating Multi-Granular Anonymized Data Sets

A straightforward approach to generating anonymized data sets of different granularity is to re-anonymize T to obtain $T_1^{'}, \dots, T_n^{'}$. One must then verify that k-anonymity is preserved for each record over all anonymizations of T.

```
Algorithm LeafScan
INPUT: Set of ordered leaf nodes $N$, granularity parameter $k_1$
OUTPUT: A new set of partitions $S$
LS1. $S \leftarrow$ empty set of partitions
LS2. while $N \neq \emptyset$
LS3. $P \leftarrow$ empty partition
     while $|P| \leq k_1$
         $L \leftarrow$ next leaf node in $N$
         Add all records in $L$ to $P$
         $N \leftarrow N - L$
LS4. if the total number of records in the remaining leaf nodes
     in $N$ is less than $k_1$, then remove these records from $N$
     and add them to $P$
LS5. update generalized quasi-identifier values for every record
     in $P$
LS6. $S \leftarrow S \cup P$. Continue LS2
LS7. Return $S$
```

Figure 5: Leaf scan algorithm

We can, however, take advantage of the tree structure of spatial indexes to generate data sets of different granularity that automatically guarantees that k-anonymity is maintained for the collection of anonymized data sets. This technique exploits Lemma 1 by effectively *binding* each record r to some pre-determined set of k records.

Let *SI* be a multi-dimensional spatial index on the original data set with the following properties.

- Leaf nodes in *SI* contain between k and ck records, some constant c.
- Internal nodes in *SI* contain between l and m entries.

From each level in *SI*, we can automatically generate anonymized data sets of granularity $k, lk, l^2k, \dots, l^hk$, where h is the height of *SI*. To generate an anonymized table of granularity l^ik, we map each node N_j at level i to each partition P_j in the anonymized table. The records in P_j are all the records contained in the set of leaf nodes in the subtree rooted at N_j.

Using this *hierarchical* algorithm for generating multi-granular anonymized data sets guarantees k-anonymity over all anonymizations of T. To see why, pick any record r_i in T and let P_j be any partition from an anonymization of T. If P_j contains r_i, then P_j contains the leaf node L in the subtree for which P_j is the root. Thus, from Lemma 1, every record in L is k-bound, r_i is k-bound.

In generating multi-granular anonymized data sets via the hierarchical algorithm on a spatial index, the data set owner can, at the very least, guarantee the anonymity generated by the leaf nodes in the spatial index. In other words, if the leaf nodes produce a k-anonymous data set (every leaf node contains a minimum of k records), then it can be guaranteed that releasing other data sets at other granularity $k_1 > k$ will not violate k-anonymity. If an adversary manages to obtain multiple versions of anonymized tables, with the goal of re-identifying individuals, she can only recover the information revealed in the finest granular (most precise) anonymized data set in her possession.

3.2 A Leaf Scan Algorithm for Generating Multi-Granular Anonymized Data Sets

In this section, we describe an algorithm that rather than generate anonymized tables in a hierarchical fashion, utilizes the "sequential ordering" of nodes on the same tree level. The granularity of the data sets that can be generated using a hierarchical algorithm is restricted by the threshold on the minimum number of records in any leaf node in the tree. This minimum occupancy threshold on leaf nodes enforces the property that every subtree contain a minimum number of records. Assume that all nodes (including leaf nodes) in the spatial index *SI* contain between two and four entries. Then, at best, we can have anonymous data sets of granularity, k, $2k$, $4k$, $8k$, $\ldots, 2^h k$. Using the hierarchical algorithm, we will be unable to generate a 6-anonymous data set say, the best we can do is an 8-anonymous data set. (Of course by definition of k-anonymity, an 8-anonymous table is also 6-anonymous.) Given a request for a data set of granularity k_1, we can further improve on the hierarchical algorithm by scanning the leaf nodes in order and partitioning the leaf nodes in groups of k_1/k. In our current example, since every leaf node contains at least two records, if $k_1 = 6$, we scan the leaf nodes, forming groups of three leaf nodes each except for the last group that may contain between three and five nodes.

Figure 5 shows the *leaf scan* algorithm for performing multi-granular anonymizations of a data set. Note that since each leaf node may contain between k and ck records, we may be able to form groups containing less than k_1/k leaf nodes. The algorithm initializes a new group or partition with the next leaf node; if this group contains at least k_1 records, we are done with this group and start a new one. Otherwise a new leaf is added to the current group. The algorithm stops adding leaf nodes to a group when the total number of records in the group is at least k_1. New groups are iteratively created until the last step, when the records in the remaining leaf nodes is less than k_1, we add these leaf nodes to the current and last group, terminating the algorithm. Since the records in a group may span multiple leaf nodes, the algorithm recomputes new generalized quasi-identifier values based on all the records in a group. We use this approach in our implementation to generate data sets of different granularity. As the results in Section 5 will show, execution times for anonymizing a data set is independent of the actual anonymity parameter k since generating an anonymous data set of any granularity requires one full scan of all the leaf nodes (after building the index on a base k value).

By Lemma 1, k-anonymity is also preserved when we generate multi-granular anonymized data sets with the leaf scan algorithm. To see why, we observe that the leaf scan algorithm always forms partitions from *whole* leaf nodes. If P_j is any partition from an anonymization of T and $r_i \in P_j$ then P_j contains the leaf node partition L that contains r_i, thus r_i is always "bound" to the records in L and $|L| \geq k$.

4. A COMPACTION PROCEDURE

In the process of treating the k-anonymization problem as an indexing problem, we recognized that we could dramatically increase the precision of anonymized data sets by employing some of the techniques for improving query performance on the R-tree style indexes. By using minimum bounding boxes, these spatial indexes leave gaps in the domain where gaps correspond to spatial portions of the domain that do not contain any record. We, thus, propose a *compaction* procedure to increase the precision of an anonymized data set generated for any index, such as the grid file [23], that does not maintain MBRs for its records. Since every k-anonymization algorithm, whether viewed as an indexing technique or not, essentially creates partitions in the original data set, the compaction technique can be retrofitted to previously proposed non-index-based approaches to give dramatic improvements as well.

The goal of the compaction procedure is to regenerate, for each partition P in a k-anonymous data set D, another partition P_1 with a possibly "more precise" description about the records in P. The compaction algorithm is a simple one — it scans each partition $P \in D$ and creates the minimum bounding boxes. For each numerical quasi-identifier attribute, the compaction algorithm generates a new range where the end points are the minimum and maximum values that occur for records in P. For each categorical attribute, the procedure removes all values from the set that do not occur in P. Where generalization hierarchies are used in place of sets, the procedure chooses the lowest common ancestor in the hierarchy for all the values in P. The old generalized values for every record in P are then replaced with the new, more compact values. Figure 6(b) depicts an example application of the compaction procedure to anonymized data in Figure 6(a).

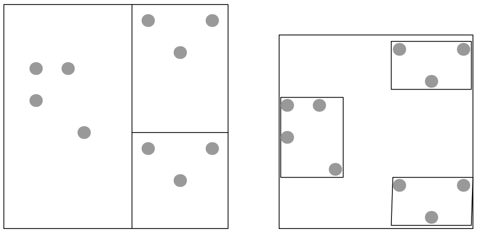

(a) uncompacted data (b) compacted data

Figure 6: Applying the compaction procedure to partitions

In the rest of the paper, we will refer to the data before applying the compaction process as the **uncompacted** data and the data after applying the compaction process as the **compacted** data. A benefit of the compaction procedure is that, when compared to results from the same queries on the original data, query results on compacted data are more accurate than queries on uncompacted data. Due to the introduction of gaps in the anonymized data, queries that would have otherwise returned non-empty result sets for one or more partitions now return results that are more in tune with the original data set. In experiments, we see dramatic improvements in accuracy for queries on compacted data over the same queries on uncompacted data. We should note that the simple nature of the compaction procedure facilitates its application to data generated by any k-anonymization algorithm.

It is also reasonable to expect the execution costs for the compaction process to be relatively small when compared to actual anonymization costs as its basic operation is a single pass over each partition to determine minimum and maximum values for numerical attributes and minimal sets for categorical attributes. These relatively small compaction costs are verified through experimental results shown in Section 5 by running the compaction procedure on anonymized data generated by a previously proposed k-anonymization algorithm.

This compaction process may lead one to an uneasy feeling that "more is being revealed" than would be revealed if the anonymized data set were not compacted. This is actually true. For example, an adversary can "know" that there is no individual in a "gap" area, something they could not deduce without compaction. This is an example of the tension between anonymization procedures and data utility. But this is really an issue in all k-anonymization research.

For example, the discernibility penalty [4] rewards anonymization procedures that do a good job of putting no more than k data points in a partition. This reveals more information than another anonymization that has partitions with more than k data points. To see this, suppose that anonymization A puts $k' > k$ data points in a number of partitions, but anonymization B puts only k in each par-

Category	Description
Compiler	gcc 3.2.2
Operating system	Tao Linux release 1
CPU	Intel Pentium 4 3.00Ghz
Memory	1GB
Hard Disk	Seagate ATA/ATAPI-6

Table 1: System configuration

tition. Then with anonymization B, the "attacker" knows for every data record that the sensitive value associated with that record must be one of the alternatives appearing in the *k* records in the partition. On the other hand, with anonymization A, for any data element in a "large" partition, the attacker only knows that the data record has a sensitive value among the values found in the $k' > k$ records in the partition, which in general may be a larger set. Another way of putting this is that if a procedure reduces the number of records in a partition from k' to k, it has "revealed" information, but the discernibility penalty says that the quality of the anonymization has improved. In an extreme case, if $k' = N$, the number of elements in the data set, the attacker gains almost no information from the anonymized data; but of course, then the anonymized data is not useful for any non-trivial analysis. The discernibility penalty tries to "penalize" the anonymization algorithm for this; one way of viewing it is that it tries to encourage disclosing as much information as possible while still not violating *k*-anonymity.

The recently proposed "certainty metric" [33] has the same character. It rewards anonymization procedures for creating partitions with small perimeters. Intuitively, a smaller perimeter for a partition *P* means that the quasi-identifier for the elements in *P* can be known from the outside to be restricted to a smaller set of values than would be the case if the perimeter were larger. Once again, the goal is to reveal as precise information as possible without violating *k*-anonymity. Our "shrinking" or "compaction" procedure is yet another step in this direction. It tries to bound partitions of *k*-elements as tightly as possible while still not violating the *k*-anonymity requirement.

We note that recently there has been work on augmenting the definitions of *k*-anonymity to provide stronger guarantees. For example, in *l*-diversity [21], the *k*-anonymity requirement is extended to require a certain degree of diversity in the sensitive values of the records in a partition. Our shrinking procedure is orthogonal to this kind of requirement — whatever the requirement, it tries to find the smallest bounding box on the *k*-elements that still satisfies the requirements (*k*-anonymity, or *k*-anonymity and *l*-diversity.)

We argue that this is the correct way to deal with information disclosure in the *k*-anonymous framework. If one thinks too much information is being revealed, one should strengthen the restrictions on the definition of what constitutes an allowable partition (for example, adding *l*-diversity to *k*-anonymity) rather than trust that the anonymization procedure will only generate "loose" or "imprecise" partitionings in some uncontrolled way. To reiterate, our philosophy is that the definition of what is an allowable partition is taken as input; the goal of an anonymization procedure is to produce the "best" or "most precise" partitioning that respects the definition. That is what the shrinking procedure attempts to do.

5. EXPERIMENTS AND RESULTS

We carried out experiments on two data sets for empirical evaluation of *k*-anonymization with spatial indexes. A spatial index bulk-loading algorithm, such as the buffer-tree algorithm we used, can be viewed as a bottom-up algorithm because it attempts to assign records to regions as they are processed. By comparison, the polynomial-time algorithm suggested in the Mondrian paper [19] can be viewed as top-down, because its first step is to partition the entire space, then to partition the resulting sub-regions, and so forth. Although the goal of the Mondrian algorithm was not absolute performance, we implemented their top-down algorithm to compare the efficiency and quality of the top-down approach with the bottom-up approach inherent in spatial indexing. We are grateful to the authors of that paper for providing us with a copy of their Java prototype implementation of the algorithm, as well as the data sets they used in their experiments.

The first data set we used was a real world data set, the Lands End data set. The configuration for this data set was identical to [18]. The Lands End data set contained customer sale information and 4,591,581 records. It had eight attributes comprising *zipcode, order date, gender, style, price, quantity, cost* and *shipment*. Unlike [18] however, hierarchical constraints were eliminated by imposing an intuitive ordering on the values for each categorical attribute in the data set. Each record in the resulting data was 32 bytes and the entire data set was approximately 147 MB in size. The second data set was synthetically generated and had nine attributes comprising *salary, commission, age, education level, car, zipcode, house value, house years* and *loan*. The configuration for the synthetic data was based on the generator introduced in [1]. We generated 100 million records, each record was 36 bytes, resulting in a data set size of 3.6 GB.

We built the R^+-trees on all eight and nine attributes for the Lands End and synthetic data sets respectively (every attribute was part of the quasi-identifier). As a result of the numerical recoding on the original data sets, the schema for an anonymized table is as follows: each quasi-identifier value for a record *t* in the original data set is replaced, in the anonymized data, by the interval, on that quasi-identifier, of the MBR containing *t*. As a consequence, we were also able to perform query experiments described in Section 5.4 by specifying numerical ranges in the query predicates.

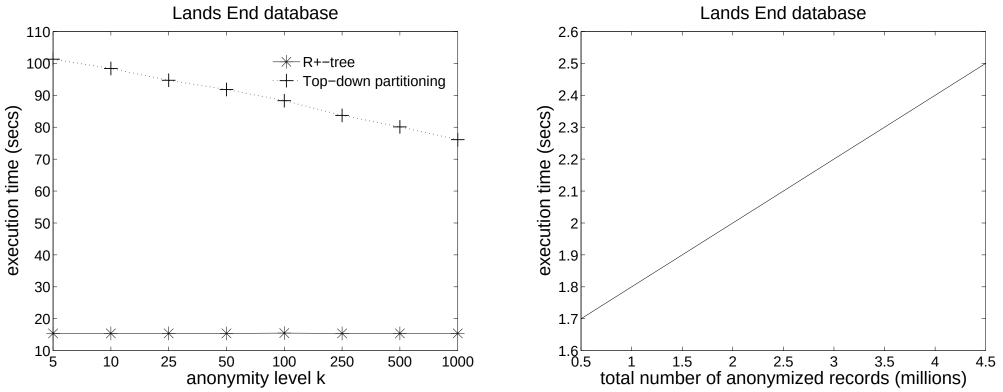
Table 1 gives a description of the system configurations used in all experiments.

5.1 Performance Evaluation

We used the Lands End data for the first set of experiments to compare running times for the R^+-tree bulk-loading to a top-down multi-dimensional partitioning approach. We also ran experiments to evaluate incremental anonymization performance for the R^+-tree. For these experiments, both algorithms were each allocated a maximum buffer size of 256 MB. Each experiment was carried out five times while flushing the system file buffers between runs, and we report average cold running times.

Figure 7(a) shows execution times for R^+-tree bulkload and top-down approach on the Lands End data set for different anonymity levels *k* = 5, 10, 25, 50, 100, 250, 500, 1000. As *k* increases, the execution times for the top-down algorithm decreases since there are fewer recursive partitioning steps. Results show that the spatial indexing approach consistently outperforms the top-down technique suggesting that the former is more efficient than a top-down recursive partitioning scheme even for bulk anonymization. Notice that the execution times for the R^+-tree is independent of *k*. This is due to the fact that we choose a base *k* for the bulkload process. For these experiments, we selected base *k* = 5. For the actual input *k* parameter, we used the leaf scan algorithm described in Section 3.2 to construct the final partitions. For example if *k* = 5, then the mapping is one or more leaf nodes to one partition. If *k* = 10, then we map two or more leaf nodes to one partition.

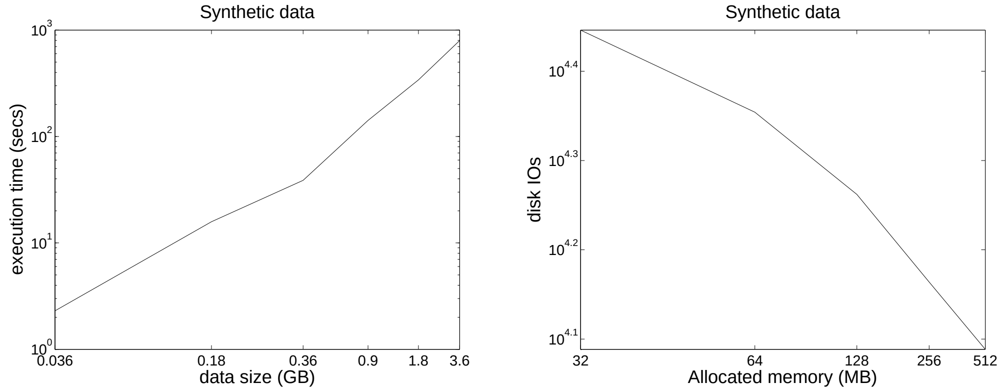
We start the incremental anonymization experiments by first bulk-



(a) Anonymization execution time comparison

(b) R$^+$-tree incremental anonymization times using batch size of 0.5 million records (*k* = 10)

Figure 7: Execution times on the Lands End database



(a) Execution times for varied data set sizes, allocated memory = 256 MB

(b) IO costs, data set size = 3.6GB

Figure 8: R$^+$-tree anonymization scaling to large data sets

loading and anonymizing the first 0.5 million records from the Lands End data set. Then we subsequently select new batches of 0.5 million records to be anonymized. Figure 7(b) shows the running times for the R$^+$-tree for incrementally anonymizing each batch. Since a top-down approach is not incremental, it would have to re-anonymize the entire data set on each batch insert.

5.2 Scalability

The synthetic data set was used for the second set of experiments to show that our spatial indexing techniques scale well to the size of the input data set. We did not test the top-down approach for scaling to larger-than-memory data sets because the version of the Mondrian algorithm described in [19] was not designed to handle such data sets.

We used the R$^+$-tree to *k*-anonymize data sets of different sizes ranging from one million (0.036 GB) to 100 million records (3.6 GB). We evaluate how the spatial index performs when the data set is too large to fit in main memory. We allotted 256MB to the anonymization process and from the allotted memory, we use 4MB to buffer the input data and the remaining memory was used for the index buffers. The index was able to hold a major portion of the records in its buffers when the data set is less than 5 million records (about 180MB). Figure 8(a) shows that our algorithms adapt gracefully as the size of the input data is increased.

We also conducted experiments to ascertain the effect of available computer memory on the performance of our algorithms. Figure 8(b) shows the total number of explicit I/O system calls made during the anonymization process while varying the size of memory allotted to the process. These results show that I/O costs increase by less than a factor of two when the allotted memory is reduced by a factor of two, also indicating that the buffer tree algorithm performs well as a spatial indexing bulk-loading algorithm for larger-than-memory data sets.

5.3 Anonymization Quality Experiments

Since anonymization quality is an essential property of any *k*-anonymization algorithm, we conducted a third set of experiments — quality experiments on anonymized data generated by the R$^+$-tree and the top-down approach proposed in [19]. We also evaluated the quality of anonymized data after executing compaction as a post-processing step on data generated by that approach.

In choosing algorithms with which to compare our algorithm, we were guided by the following considerations. First, since we had access to the code of the Mondrian algorithm, it was a logical

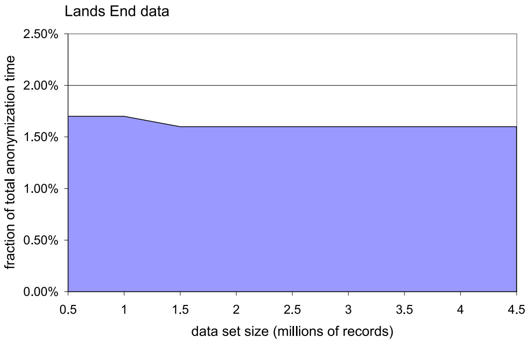


Figure 9: Compaction cost as a percentage of total anonymization execution time (k = 10)

candidate for comparison. Second, we did not consider comparing against any algorithm that had already been shown to produce lower quality anonymizations than Mondrian. This eliminated [4, 18, 24]. Finally, since our focus in this paper is on scalable anonymization algorithms, we did not consider algorithms with running times greater than $O(n^2)$. This eliminated [4, 11, 18, 24, 26, 33].

First, we measured the running times for compaction relative to total anonymization times while choosing samples of different sizes ranging from 0.5 million to 4.5 million records from the Lands End data set. Figure 9 shows that the times for compaction are small relative to the anonymization times.

We used the discernibility metric [4], KL divergence [15] and the certainty metric proposed in [33] to measure and compare quality with anonymized data generated from the Lands End data set by the top-down approach proposed in [19]. Before continuing with the anonymization quality experiments, we define these three quality measures.

Let T be a k-anonymous table with m quasi-identifier attributes, $A_1, \ldots, A_m$, and comprising n partitions or equivalence classes, $P_1, \ldots, P_n$.

DEFINITION 3. (DISCERNIBILITY PENALTY). *The discernibility penalty score for* T, $DM(T) = \sum_{i=1}^{n} |P_i|^2$

The discernibility measure, in effect, assigns a penalty to each tuple $t \in T$ based on the size of the partition P_i, given that $t \in P_i$.

DEFINITION 4. (CERTAINTY PENALTY). *The certainty penalty score for* T *is*

$$CM(T) = \sum_{t \in T} NCP(t)$$

where NCP(t) is the weighted normalized certainty penalty assigned to the tuple t.

$$NCP(t) = \sum_{i=1}^{m} \left(w_i \times \frac{|t.A_i|}{|T.A_i|} \right)$$

w_i is the weight associated with the quasi-identifier attribute A_i to reflect its *importance* in the anonymized data. In our quality experiments, we set the weights for all quasi-identifier attributes to 1. If A_i is a numerical attribute then $|t.A_i|$ is the range of the generalized value for t on A_i. For example, if t.Age = [20 – 30], then $|t.A_i|$ = 10. $|T.A_i|$ is the range of all tuples in T on A_i. Otherwise A_i is a categorical attribute and assuming the existence of a generalization hierarchy tree H for A_i, $|t.A_i|$ is the number of leaf nodes in the subtree of H for which the node for the generalized value $t.A_i$ is root. $|T.A_i|$ is the total number of leaf nodes in H.

DEFINITION 5. (KL-DIVERGENCE). *The KL divergence for* T, $KL(T) = \sum_{t \in T} p_t^{(1)} \log \frac{p_t^{(1)}}{p_t^{(2)}}$ *where* $p_t^{(1)}$ *and* $p_t^{(2)}$ *are the probabilities of the tuple t according to the original and anonymized tables respectively.*

See [15] for more detailed descriptions on the computation of $p_t^{(1)}$ and $p_t^{(2)}$.

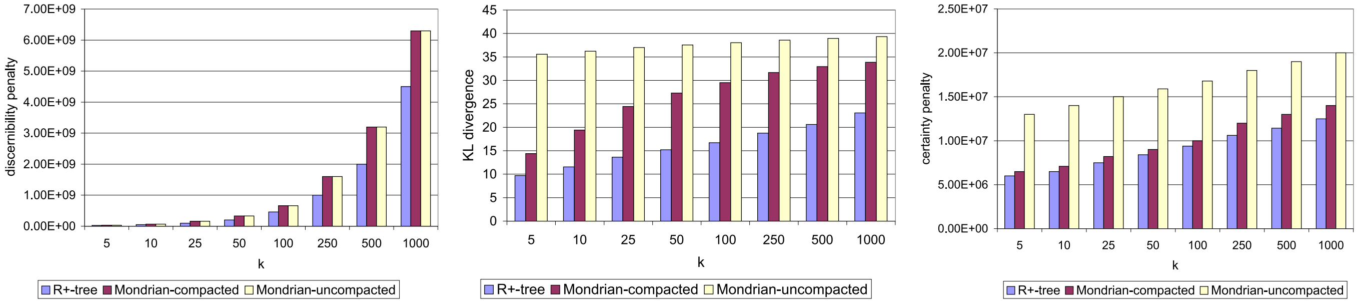
Figure 10 shows that exploiting the minimum bounding rectangle property of the R$^+$-tree clearly pays off as it generates much better anonymized data quality than the top-down approach without compaction. The Mondrian partitioning is based on the heuristic that it should split the quasi-identifier attribute with the largest range of values, whereas the R-tree splits by trying to minimize the area of the resulting partitions. For the data set used in our experiments, minimizing area results in better discernibility, KL-divergence and certainty penalty. Results in Figures 10(b) and 10(c) show that after compaction, the anonymized data quality for the top-down approach greatly improves, suggesting that one of the main differences between the anonymization quality produced by the top-down and spatial index approaches arises from the compaction implicit in the spatial index.

We also observe from Figure 10(a) that the discernibility scores for uncompacted and compacted Mondrian-anonymized data are identical. The compacted and uncompacted anonymized data comprise the same partitions with the same number of records per partition, although the former may describe these partitions with more precision on the quasi-identifier attributes. As a consequence of the discernibility penalty measuring quality based on the cardinalities of partitions alone, this metric is unable to measure differences between compacted and uncompacted anonymized data. This shows that the certainty penalty and KL-divergence measures can identify differences in quality in data anonymization that escape the discernibility penalty. Of course, while we did not see any in our experiments, it is also possible that there are examples of anonymizations in which certainty penalty detects differences that are also missed by KL-divergence and vice versa, in which case these metrics are incommensurate and future study would be warranted to determine the strengths and limitations of each.

Since a benefit of spatial indexes is that it is incremental, we also conducted tests to ascertain the effects, on quality, of anonymizing new records incrementally as opposed to a re-anonymization approach. (Earlier we discussed the impact of incremental anonymization on the efficiency of anonymization.) For the R$^+$-tree index, we first loaded a portion of the data (500,000 records), anonymized them, then incrementally anonymized additional batches of 500,000 records by inserting these records into the index. For Mondrian, we reanonymized the entire set of records. Results from our experiments validate our assertion that the R$^+$-tree's technique of adding new records to the index in such a way that the total area covered by the resulting MBRs is minimized is appropriate for maintaining good data quality. Figure 11 shows that anonymized data quality does not suffer from incremental anonymization, in fact, the R$^+$-tree anonymized data is still of higher quality than the re-anonymized data.

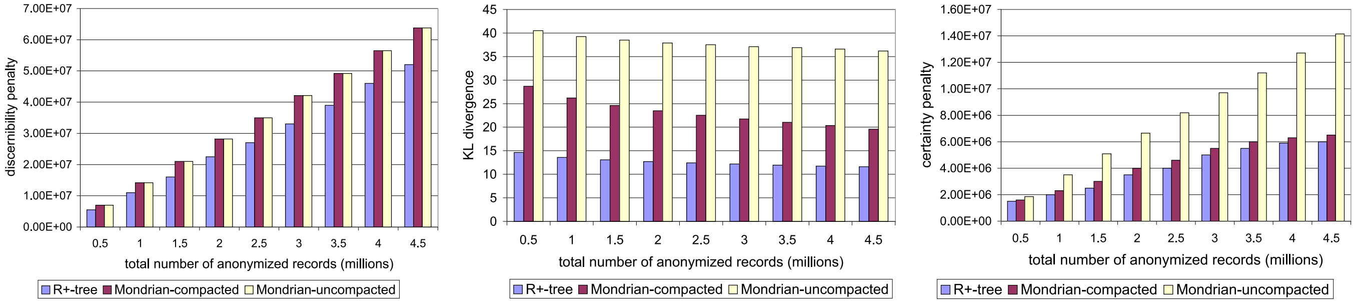
5.4 Query Accuracy

Since the KL-divergence and certainty penalty measures suggest that compacted data is at least as "good" or "better" than uncompacted data, we carried out a fourth set of experiments to evaluate the relative accuracy for queries on both types of anonymized data



(a) Discernibility penalty (b) KL Divergence (c) Certainty penalty

Figure 10: Quality comparisons on the Lands End database



(a) Discernibility penalty (b) KL Divergence (c) Certainty penalty

Figure 11: Incremental quality comparisons on the Lands End database (*k* = 10)

on the Lands End data set. We also compare accuracy for the same queries on the R$^+$-tree anonymized data. We executed 1000 randomly generated range queries over all eight attributes of the Lands End data set. These queries were of the form

SELECT COUNT(*)
FROM T
WHERE T.A$_1 \geq a_1$ AND T.A$_1 \leq b_1$
⋮
AND
⋮
T.A$_8 \geq a_8$ AND T.A$_8 \leq b_8$

Note that in the above eight-dimensional range query, each of the eight attributes A$_1 \ldots$ A$_m$ has both an upper and lower bound on its range. These bounds are set in the following manner: for each query, we pick two records r_1 and r_2 at random from the unanonymized data set and set each a_i, $i = 1, \ldots, 8$, to the smaller of $r_1.A_i$ and $r_2.A_i$, and b_i to the larger value.

A COUNT query Q on the anonymized data set T returns a count of the records in T that match Q. A record $r \in T$ is said to *match* the query Q if the region defined by r has a non-null intersection with the query region — r intersects with Q on every quasi-identifier attribute. For example, if T has two quasi-identifier attributes, Age and Zipcode, the record r = ([40 – 50], [53710 – 53720]) matches the query $Q = \big((45 \leq \text{age} \leq 55) \wedge (53700 \leq \text{zipcode} \leq 53715)\big)$. The record r = ([30 – 35], [53700 – 53715]) does not match this query. Queries on the original, unanonymized data set work in the traditional way. r matches Q on the original data if the point defined by r lies within the query region.

We compute the error for the query Q, Error(Q), as the normalized difference between the results of evaluating Q on the original and anonymized data sets.

$$Error(Q) = \frac{\text{count(anonymized) - count(original)}}{\text{count(original)}}$$

In our experiments, we report the average normalized error over the batch of 1000 queries. Figure 12(a) shows the average errors for different k values. The accuracy for queries on compacted-anonymized data is seen to improve over the accuracy for the same queries on uncompacted-anonymized data. Queries on the R$^+$-tree anonymized data have even smaller errors than on the Mondrian-compacted anonymized data suggesting that the spatial index does a good job of partitioning the original data even for an arbitrary query workload.

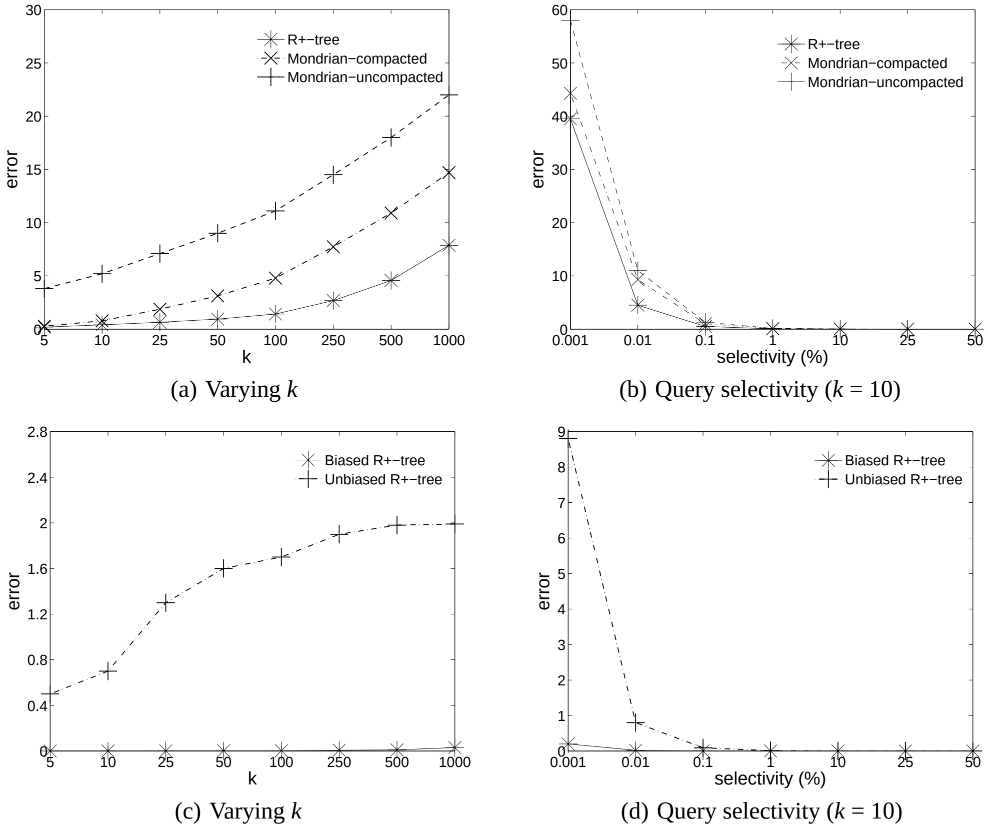
Figure 12(b) shows results when we vary the selectivity of the query on the data set. In general, the larger the cardinality of the query result, the smaller the error generated by running the query on anonymized data versus the original data. This will in turn tend to diminish differences between the accuracy of query results generated by queries run over data anonymized by different anonymization procedures. Even the benefit of compaction is seen to drop for large query results.

To show that we could indeed use a spatial index to integrate anticipated workloads into the anonymization, we constructed a query workload that generated random queries on the Zipcode attribute of the Lands End data set. These were of the form

SELECT COUNT(*)
FROM T
WHERE T.Zipcode $\geq z_1$ AND T.Zipcode $\leq z_2$

The bounds for a range query on the Zipcode attribute are set in the following manner: we pick two records r_1 and r_2 at random from the unanonymized data set and set z_1 and z_2 to r_1.Zipcode and r_2.Zipcode such that $z_1 \leq z_2$.

We also built another R$^+$-tree, but this time, biased our node



(a) Varying *k*

(b) Query selectivity (*k* = 10)

(c) Varying *k*

(d) Query selectivity (*k* = 10)

Figure 12: Query error comparisons on the Lands End database

splitting algorithm to the Zipcode attribute. The biased splitting algorithm selects the Zipcode attribute as the splitting attribute for every split. Subsequently, we generated 1000 random queries from the workload and ran these queries on the anonymized Lands End data generated by the original (unbiased) R$^+$-tree index and the new (biased) R$^+$-tree index. From Figure 12(c), we see that by favoring one attribute, we were able to achieve significantly better query results than the index that did not account for the query workload. This suggests that we can use spatial indexes to exploit advanced knowledge of expected query workloads to generate "better" anonymized data. Finally, results in Figure 12(d) show that, although the biased R$^+$-tree outperforms its unbiased counterpart for various query selectivity, the differences diminish as we increase the selectivity on the original data set.

6. CONCLUSION

We have observed that since building an index over a data set leads to a natural partitioning of the data set, *k*-anonymity can be introduced by enforcing a minimum occupancy threshold on partitions. Moreover, we can take advantage of these indexes as a means for producing dynamic *k*-anonymous data sets. Experiments indicate that the spatial indexing approach to *k*-anonymization is scalable for very large data sets, is faster than previously proposed algorithms even for in-memory data sets and produces anonymizations with better quality as measured by the discernibility penalty [4], KL-divergence [15] and certainty metric [33]. Experiments with random queries over these anonymized data sets also show that the accuracy of the queries on the R$^+$-tree anonymized data have higher accuracy than the same queries on data anonymized by a previously proposed algorithm.

Substantial room for future work remains. Recently, [17] has extended the top-down approach to work for large data sets, in which case it would be again interesting to compare the top-down approach with the spatial index bulk-loading approach. Also, the compaction procedure raises interesting questions as in a sense it "reveals" more information about the data records than uncompacted anonymizations. This is just another example of the tension between anonymizing data and exposing the useful information therein. The philosophy behind *k*-anonymization is that preventing any attacker from isolating any individual from $k-1$ others is sufficient for privacy, and the compaction procedure maintains that philosophy.

Hence one can interpret our results on compaction as demonstrating that given current definitions of *k*-anonymity, compaction is important to producing high quality anonymization. However, if one is convinced that compaction reveals too much information, then our results indicate that the definitions of anonymity need to be augmented to prevent disclosures that result from "over-compaction," because compaction respects whatever definition of anonymity the R-tree building procedure is given as input. (That is, the R-tree splitting routine can incorporate, for example, (α, k)-anonymity [32] or *l*-diversity [21] just as easily as "vanilla" *k*-anonymity.) Deciding whether or not compaction is desirable is an interesting challenge area for future work.

While this paper makes a case for R-trees as a means for *k*-anonymizing data sets, we do not lay claim to R-trees as the panacea for *k*-anonymization as spatial indexing. Several R-tree variants [2, 3, 5, 6, 12, 13, 14] have been proposed to address different indexing issues and these issues and solutions may well translate into the anonymization domain. For example, long index construction times, especially for very large data sets, are addressed through the use of bulk-loading algorithms. This translates to fast data anonymization times. Packing algorithms [12, 13] have been employed to improve bulk load times while producing a good clustering of the data in the index. Good data clustering will result in better anonymized data quality. The authors in [8] describe methods for bulk-loading a database that are resilient to different spatial data distributions, improving the spatial clustering of nodes in the index. Such techniques may be employed in devising *k*-anonymization algorithms that generate “good quality” anonymized data for a wide variety of data sets with different distributions. Recent work [16] makes a case for quad-trees as indexes for multi-dimensional data sets. The choice of one type of index over another for indexing a data set may likely be reason enough for using the same index for *k*-anonymizing the data set.

7. ACKNOWLEDGMENTS

We would like to thank David DeWitt for many helpful comments and discussions.

8. REFERENCES

[1] R. Agrawal, S. Ghosh, T. Imielinski, and A. Swami. Database mining: A performance perspective. In *IEEE Transactions on Knowledge and Data Engineering*, volume 5, 1993.

[2] L. Arge. The buffer tree: A New Technique for Optimal Algorithms (Extended Abstract). In *WADS*, pages 334–345, 1995.

[3] L. Arge, M. de Berg, and H. J. Haverkort. The priority r-tree: A pratically efficient and worst-case optimal r-tree. In *SIGMOD*, 2004.

[4] R. Bayardo and R. Agrawal. Data privacy through optimal *k*-anonymity. In *ICDE*, 2005.

[5] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *ACM SIGMOD*, pages 322–331, 1990.

[6] J. Bercken, B. Seeger, and P. Widmayer. A generic approach to bulk loading multidimensional index structures. In *VLDB*, 1997.

[7] T. Dalenius. Finding a needle in a haystack or identifying anonymous census records. *Journal of Official Statistics*, 2(3):329–336, 1986.

[8] D. J. DeWitt, N. Kabra, J. Luo, J. Patel, and J. Yu. Client server paradise. In *VLDB*, pages 558–569, 1994.

[9] B. Fung, K. Wang, and P. Yu. Top-down specialization for information and privacy preservation. In *ICDE*, 2005.

[10] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD*, pages 47–57, 1984.

[11] V. Iyengar. Transforming data to satisfy privacy constraints. In *ACM SIGKDD*, 2002.

[12] H. Jagadish. Linear clustering of objects with multiple attributes. In *ACM SIGMOD*, 1990.

[13] I. Kamel and C. Faloutsos. On packing R-trees. In *International Conference on Information and Knowledge Management*, pages 490–499, 1993.

[14] I. Kamel and C. Faloutsos. Hilbert R-tree: An improved R-tree using fractals. In *VLDB*, 1994.

[15] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *SIGMOD*, 2006.

[16] Y. Kim and J. Patel. Rethinking choices for multi-dimensional point indexing: Making the case for the often ignored quadtree. In *CIDR*, 2007.

[17] K. LeFevre and D. DeWitt. Scalable anonymization algorithms for large data sets. Technical report, University of Wisconsin, Madison, 2007.

[18] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain *k*-anonymity. In *ACM SIGMOD*, 2005.

[19] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Mondrian multidimensional *k*-anonymity. In *ICDE*, 2006.

[20] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *ACM SIGKDD*, 2006.

[21] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. *l*-diversity: Privacy beyond *k*-anonymity. In *ICDE*, 2006.

[22] M. F. Mokbel, C. Chow, and W. G. Aref. The New Casper: Query processing for location services without compromising privacy. In *VLDB*, 2006.

[23] J. Nievergelt, H. Hinterberger, and K. Sevcik. The grid file: An adaptable, symmetric mulitkey file structure. In *ACM Transactions on Database Systems*, 1984.

[24] P. Samarati. Protecting respondents’ identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[25] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1998.

[26] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: *k*-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98-04, SRI Computer Science Laboratory, 1998.

[27] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R$^+$-tree: A dynamic index for multi-dimensional objects. In *VLDB*, pages 507–518, 1987.

[28] L. Sweeney. Uniqueness of simple demographics in the U.S. population. Technical report, Carnegie Mellon University, 2000.

[29] L. Sweeney. Achieving *k*-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[30] L. Sweeney. *k*-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[31] K. Wang, P. Yu, and S. Chakraborty. Bottom-up generalization: A data mining solution to privacy solution. In *ICDM*, 2004.

[32] R. C. Wong, J. Li, A. W. Fu, and K. Wang. (α, *k*)-anonymity: An enhanced *k*-anonymity model for privacy-preserving data publishing. In *ACM SIGKDD*, 2006.

[33] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W. Fu. Utility-based anonymization using local recoding. In *ACM SIGKDD*, 2006.