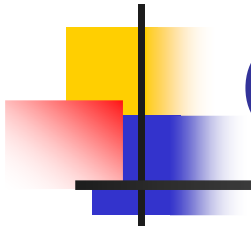
Database Indexes and K-anonymity

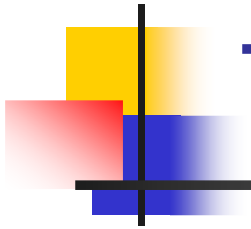
Tochukwu Iwuchukwu

Jeffrey F. Naughton



Conclusion

- There are striking similarities between
  - Building a spatial index over a data set, and
  - K-anonymizing a data set.
- We can exploit these similarities to:
  - Get fast anonymization algorithms without inventing anything.
  - Get high quality anonymization algorithms without inventing anything.
  - Open the door to anonymizing dynamic data sets (but someone will have to invent measures to address privacy problems introduced by updates.)

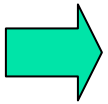


The problem

- In many cases we would like to release some information without tying that information to individuals.
- Example: medical records, where we want to release demographics and illnesses but not tie them to specific people.□
- First idea: strip away identifiers (name, id, and so forth.)
- Not good enough! (“linking attack”).

Quasi-identifiers and Linking

Name	Age	Zipcode	Diagnosis
Zach	26	53706	asthma
William	26	53708	flu
Vicky	32	52100	cancer
Tom	38	52100	flu
Sam	46	53706	back pain
Ron	46	53706	fever

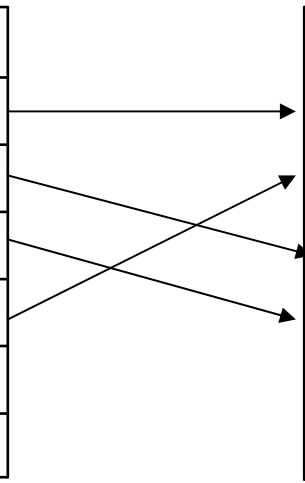


Age	Zipcode	Diagnosis
26	53706	asthma
26	53708	flu
32	52100	cancer
38	52100	flu
46	53706	back pain
46	53706	fever

- If we eliminate the “Name” field before publishing, are we safe?
- No - still have “quasi-identifiers”
  - (age, zipcode) in this case

Linking Attack

Name	Age	Zipcode
Zach	26	53706
William	26	53708
Vicky	32	52100
Tom	38	52100
Sam	46	53706
Ron	46	53706

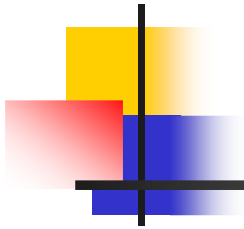


Age	Zipcode	Diagnosis
26	53706	asthma
38	52100	asthma
26	53708	cancer
32	52100	flu
46	53706	back pain
46	53706	fever

Public Table of Names and addresses

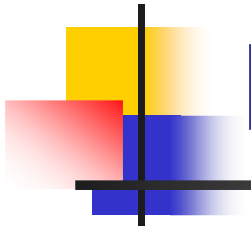
"De-identified" medical records

Four individuals in public table are uniquely identified by their age and zipcode values



K-anonymity

- Attempts to thwart linking attacks.
- Ensure that each record is indistinguishable from at least *k – 1* other records with respect to quasi-identifiers
- Definition: “Equivalence Class” or “Partition”
  - Set of tuples in a table that have the same quasi-identifier values.
- A table satisfies *k*-anonymity if every partition has cardinality at least *k*



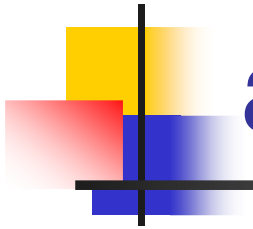
K-anonymity

Name	Age	Zipcode	Diagnosis
Zach	26	53706	asthma
William	26	53708	flu
Vicky	32	52100	cancer
Tom	38	52100	flu
Sam	46	53706	back pain
Ron	46	53706	fever

Age	Zipcode	Diagnosis
[20 – 29]	[53705 – 53709]	asthma
[20 – 29]	[53705 – 53709]	flu
[30 – 39]	[52100 – 52104]	cancer
[30 – 39]	[52100 – 52104]	flu
[45 – 49]	[53706 – 53710]	back pain
[45 – 49]	[53706 – 53710]	fever

- Every partition contains at least two records (*a 2-anonymous table*)
- Intuition: now linking attack can only connect individual to a pair of records.

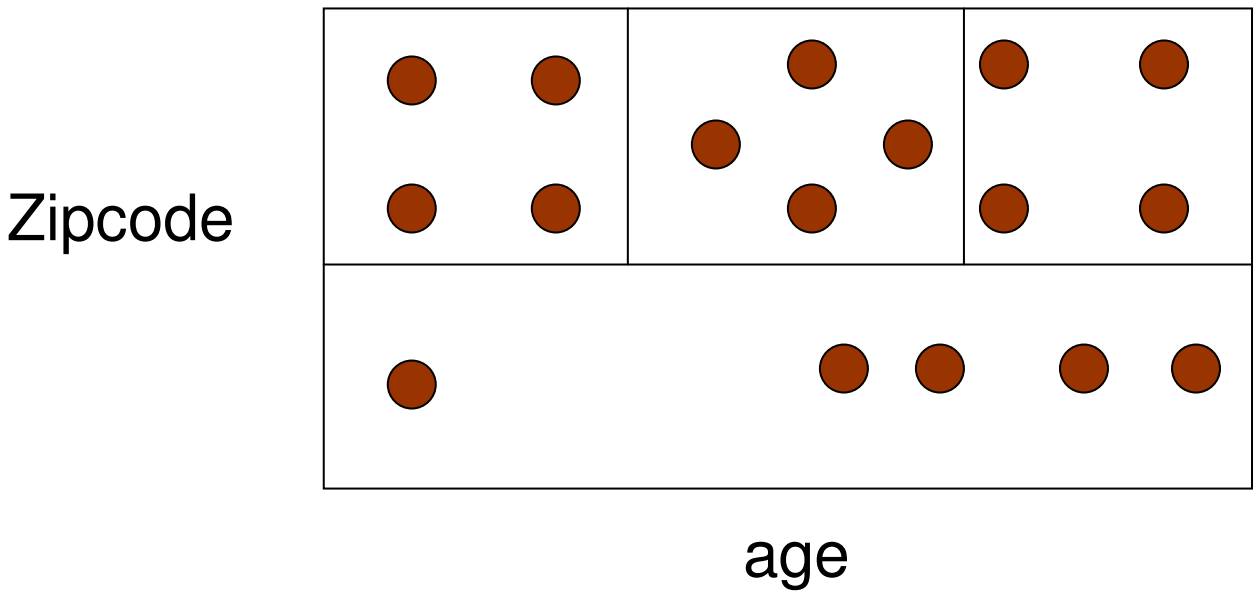
So how do you achieve k-anonymity?

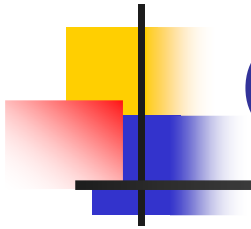


- Most common basic idea: replace quasi-identifier values with ranges of values.
- The ranges define regions
  - Two quasi-identifiers (as in previous example) mean rectangles
  - Three quasi-identifiers would mean 3D solids.
- All points with quasi-identifier values in the same region belong to the same equivalence class.

Visualizing Regions

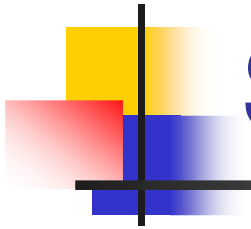
- For example, a 4-anonymous data set might look like this:





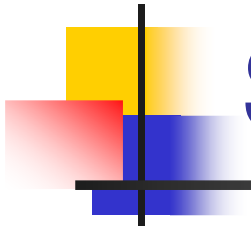
Connection to Indexing

- To someone who has worked with spatial data in a DBMS, the partitions in the previous picture look a lot like the partitions of a spatial index.
- Spatial indexes partition space, with at least *k* and at most 2*k* records per partition.
- Done for efficiency reasons -
  - Partitions map to pages
  - <= 2*k* records fit on page,
  - < *k* would waste space.



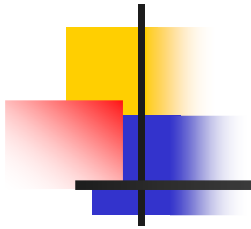
So the main idea…

- To anonymize a data set:
  - Treat it as an *n*-dimensional spatial data set, where *n* is the number of quasi-identifiers.
  - “pretend” that pages can hold *2k* points, where *k* is the anonymity parameter
  - Build spatial index over the data set.
  - Use leaves as partitions for *k*-anonymity.



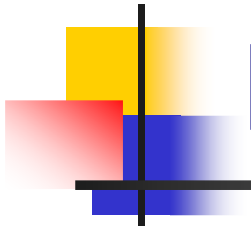
So what?

- Well, the connection between anonymity and indexing is interesting.
- Any tangible benefits?
  - Many years of research on fast, scalable index building and maintenance algorithms.
  - Indexes designed to be integrated in DBMS (could support a “k-anonymous file” storage structure).
  - Indexes designed to support dynamic data sets (more on this later.)



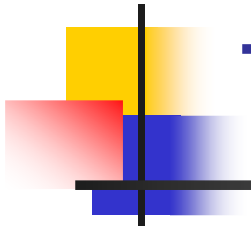
Is indexing really effective?

- To find out, implemented anonymization as bulk-loading in R^+-trees.
- Specific algorithm: “buffer-tree bulk-loading.”
- Ran performance numbers.
- Result: bulk-loading faster than previously proposed anonymization algorithms
  - Especially when data set is larger than memory.
  - But anonymization algorithms moving target…
  - Recent work on scalable Mondrian narrows gap, it will be interesting to see how this plays out.



Experimental Configuration

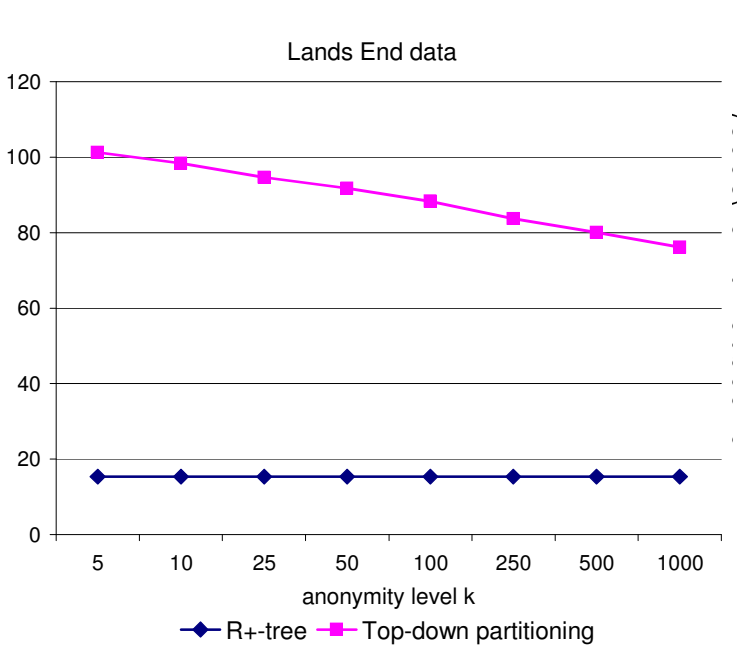
- System configurations
  - C++
  - Tao Linux 1.0
  - Intel Pentium 4, 3 GHz processor
  - 1 GB memory
- Lands End dataset
  - Eight quasi-identifiers
  - 4,591,581 records
- Synthetic data set
  - Nine quasi-identifiers
  - 100 million records

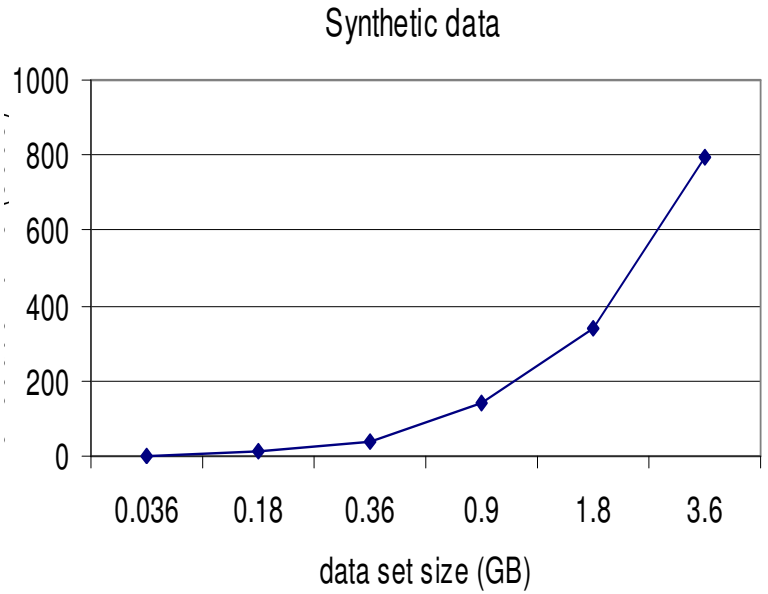


Terminology

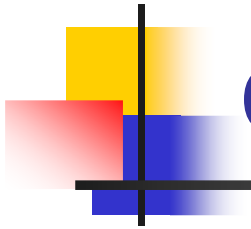
- Index bulk-loading is “bottom-up.”
  - Start filling a “page” with records;
  - When you get $2k$ records, split.
- Fastest algorithm for anonymization (Mondrian) is “top-down.”
  - Scan full dataset, choose splitting point
  - Recursively repeat

Performance and Scalability

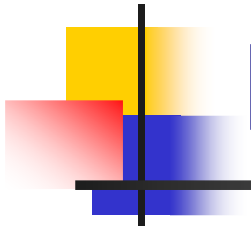




Fast - but what about quality of result?



- Important question: what do you mean by quality?
- Two previously proposed metrics:
  - Discernibility penalty.
  - Certainty metric.



Discernibility Penalty

- E = equivalence class
- DM = discernibility measure
- $DM = \sum_E |E|^2$
- The “penalty” for each record is the cardinality of its equivalence class
- More uniform equivalence classes mean lower penalty.
- Independent of how much an anonymization “blurs” quasi-identifier values.

More on discernibility

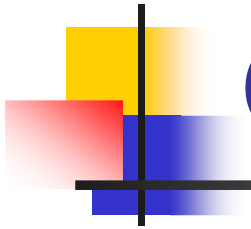
Age	Zipcode	Diagnosis
[20 – 29]	[53705 – 53709]	asthma
[20 – 29]	[53705 – 53709]	flu
[30 – 39]	[52100 – 52104]	cancer
[30 – 39]	[52100 – 52104]	flu
[40 – 49]	[53706 – 53710]	back pain
[40 – 49]	[53706 – 53710]	fever

Version 1

Age	Zipcode	Diagnosis
[20 – 29]	[52000 – 54000]	asthma
[20 – 29]	[52000 – 54000]	flu
[30 – 39]	[52000 – 54000]	cancer
[30 – 39]	[52000 – 54000]	flu
[40 – 49]	[52000 – 54000]	back pain
[40 – 49]	[52000 – 54000]	fever

Version 2

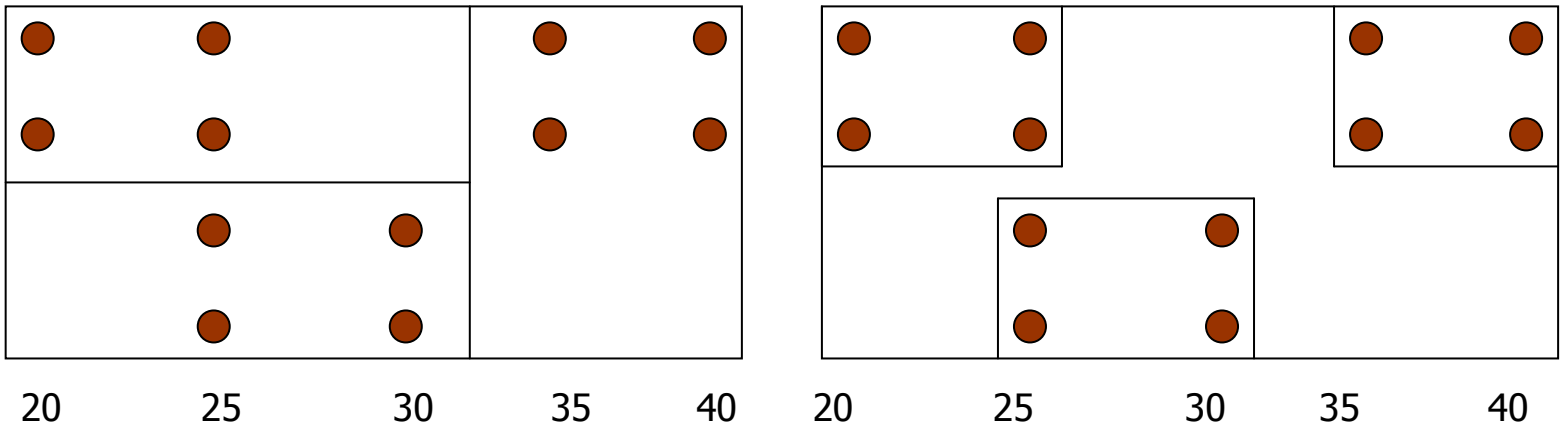
- Both tables have the same discernibility scores
- Version 1 describes zipcode more precisely than Version 2



Certainty Penalty

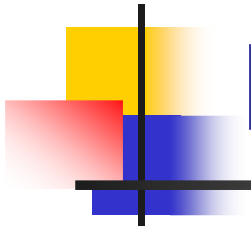
- Uses the “perimeter” of partitions to compute quality
- $\sum_E |E| * \text{Perimeter}(E)$
- Minimizing average perimeter of partitions means better quality.
- Perimeter is related to how much the quasi-identifier values have been “blurred.”

Property of R+-trees: Minimum bounding rectangles



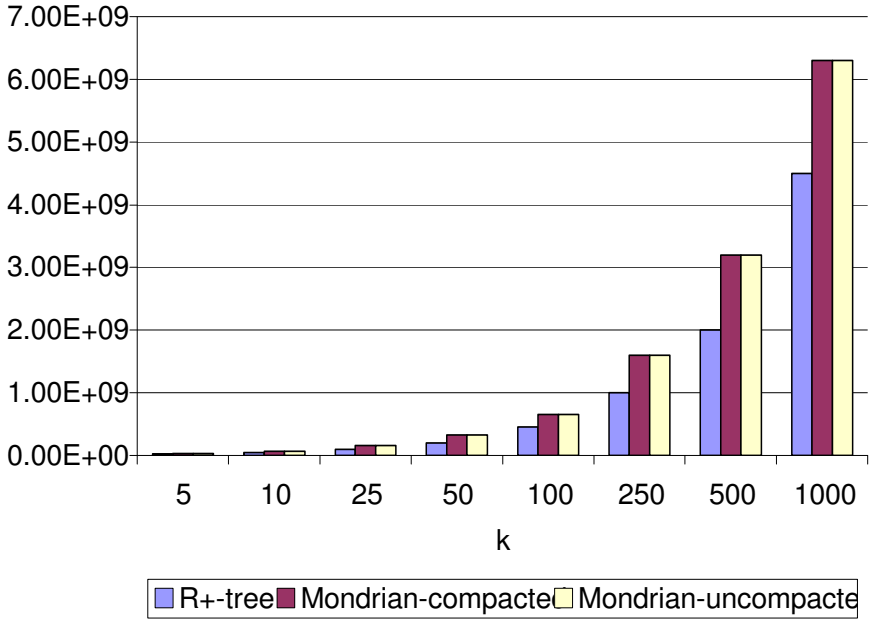
- R+-trees will give you the right partitioning, not the left.
- This tends to give much lower certainty penalty than the left approach.

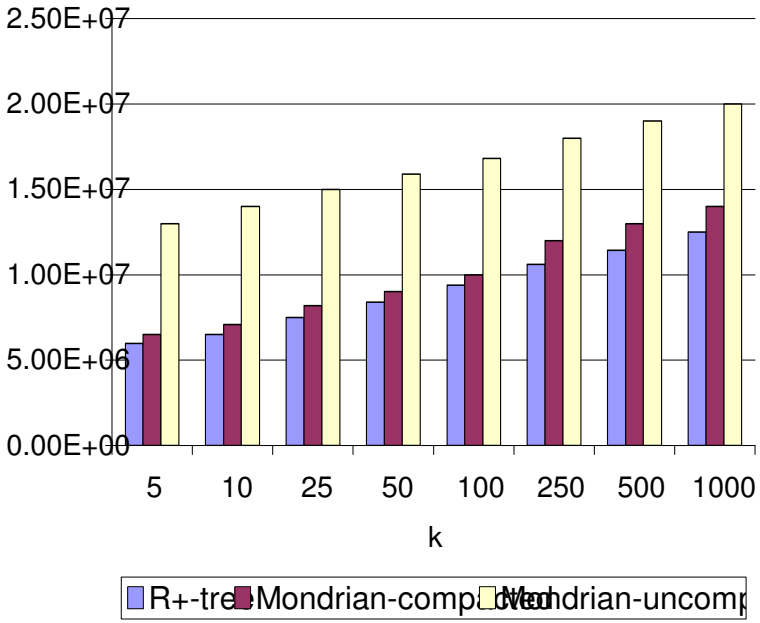
Note about Minimum Bounding Rectangles



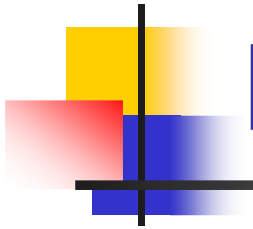
- Can easily apply “compacting” procedure to result of any anonymization algorithm as a post-processing step.
- Improves metrics like certainty.

Quality

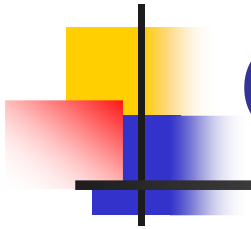

7.00E+09
6.00E+09
5.00E+09
4.00E+09
3.00E+09
2.00E+09
1.00E+09
0.00E+00
5
10
25
50
100
250
500
1000
k
R+-tree
Mondrian-compacte
Mondrian-uncompacte


2.50E+07
2.00E+07
1.50E+07
1.00E+07
5.00E+06
0.00E+00
5
10
25
50
100
250
500
1000
k
R+-tre
Mondrian-comp
Mondrian-uncomp

Is Using Minimum Bounding Rectangles a Good Idea?

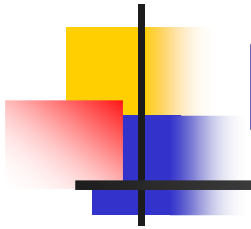


- Pro: reveal more information about data while still preserving k-anonymity.
- Con: reveal more information about data while still preserving k-anonymity.



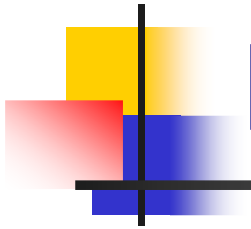
Our philosophy

- Anonymization algorithms should strive to maximize quality metrics while still satisfying definition of anonymization.
- If too much information is being revealed,
  - Augment definition of anonymity.
  - Don’t rely on “sloppy” implementation of definition.



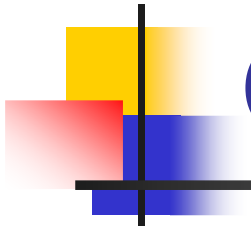
Dynamic Data

- Database indexes support efficient incremental indexing.
  - Most likely much faster than re-anonymizing from scratch on every update.
- So the indexing approach to anonymization immediately gives us a way to anonymize dynamic data sets.
- Is this safe?



Dynamic data (cont.)

- Publishing a sequence of k-anonymous data sets does not guarantee k-anonymity.
- Problem: watching inserts, deletes, and updates can violate anonymity.
  - Easy: delete until < k records in a partition
  - Harder: delete some records, insert some records, still have >= k, but observant adversary has learned something…
- So for dynamic data sets we need to augment indexing approach with some inference control mechanism to manage updates. [future work!]



Conclusion

- Spatial indexing provides a scalable, efficient approach to good quality *k*-anonymization
- Raises some interesting questions:
  - What other lessons from indexing can we exploit?
  - Can indexing exploit lessons from anonymization?
    - Workload specific splitting policies?
  - Is compaction a good idea? Do we need to change definitions to prevent it?
  - Can this form the basis for “anonymized table” storage option?
  - Can this form the basis for anonymization of dynamic data sets?