Reading:
Towards Indexing Functions: Answering Scalar Product Queries

Yu-Pin Liang (Jasmine)

What is the problem?

Background

- In a wide range of complex data analytic applications, the query processing often requires computing an expression that involves multiple columns in the relational database.
- In this study, a broad category of queries which can be expressed as the **scalar product** between a known expression (function) over multiple database attributes and an unknown set of parameters.

What is the problem?

Background

- However, in spite of many critical applications of **scalar product query**, surprisingly no generalized indexing scheme has been proposed to answer scalar product queries in an online and accurate manner.

What are the existing techniques and their limitations?

Existing techniques and limitations

1. Traditional whole table scan

2. Oracle 11.1 release has built-in support for indexing complex SQL functions over multiple attributes. However, their index does not support queries that consist of both complex functions as well as unknown parameters.

What makes it possible for the authors to address these limitations?

Planar index

In this work, the problem of fast online computation of scalar product queries in an accurate manner was addressed. To achieve this aim, a novel, lightweight, and generalized indexing scheme, called the **Planar index** was proposed, which can answer the queries very efficiently.

How is the problem actually solved in this paper?

1. **Offline** technique relies on indexing functions ϕ(x) for data points x with multiple sets of parallel hyperplanes, and pre-computing some information which is linear in the number of the data points.

How is the problem actually solved in this paper?

2. Online query evaluation consists of finding the optimal set of index hyperplanes for a given query, and then using the precomputed information to efficiently answer our queries in an exact manner.

How is the problem actually solved in this paper?

3. The key idea of **Planar** index is to allow very fast pruning of the data points without actually computing the scalar product for them.

Related work

- Half-space range searching
- Linear constraint queries
- Nearest neighbor queries
- Top-k queries with ranking function
- Index for moving objects.

Problem statement

Given a set of data points in $\mathbb{R}^d$ and an application specific function $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, we define two novel scalar product queries.

PROBLEM 1 (INEQUALITY QUERY). *Find all data points $\mathbf{x} \in \mathbb{R}^d$, which satisfy a scalar product inequality: $\langle \mathbf{a}, \phi(\mathbf{x}) \rangle \leq b$.*

PROBLEM 2 (TOP-k NEAREST NEIGHBOR QUERY). *Given some k, find the top-k data points $\mathbf{x}$ satisfying $\langle \mathbf{a}, \phi(\mathbf{x}) \rangle \leq b$, which also minimize $\frac{|\langle \mathbf{a}, \phi(\mathbf{x}) \rangle - b|}{|\mathbf{a}|}$.*

In this study, objective is to propose a generalized indexing scheme—which is easily maintainable and updatable —and which enables faster processing of both the scalar product queries in an accurate manner.

How is the performance of the proposed technique compared to that of the existing ones?

- Experiments to assess the performance of the **Planar index** for answering scalar product queries were presented.

Table 2: *Dataset characteristics.*

Dataset	# Data Points	# Dimension	Attribute Range
Indp	1,000,000	2 - 14	(1, 100)
Corr	1,000,000	2 - 14	(1, 100)
Anti	1,000,000	2 - 14	(1, 100)
CMoment	68,040	9	(-4.15, 4.59)
CTexture	68,040	16	(-5.25, 50.21)
Consumption	2,075,259	4	(0, 254)

Competing Method

- Compare the performance of Planar index with a baseline method that performs a naïve sequential scan over the entire dataset.

How is the performance of the proposed technique compared to that of the existing ones?

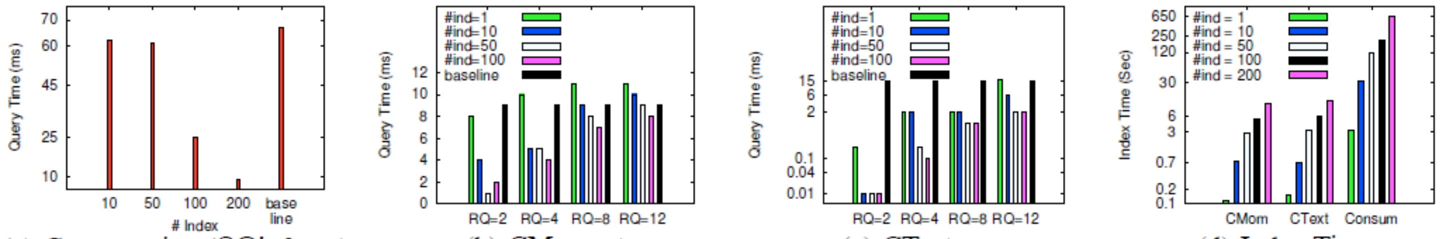
Table 1: *Time complexity of half-space range search algorithms: n number of data points, d dimensionality of the query space, t cardinality of the answer set, $\epsilon > 0$ any constant, $c = c(d)$ another constant.*

	Query time	*Preprocessing storage*	*Preprocessing time*
Agarwal et. al. [1]	$\mathcal{O}(n^{1-\frac{1}{d}+\epsilon} + t)$	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$
Matousek et. al. [19]	$\mathcal{O}(n^{1-\frac{1}{\lfloor d/2 \rfloor}}(\log n)^c + t)$	$\mathcal{O}(n \log \log n)$	$\mathcal{O}(n \log n)$
Arya et. al. [2]	$\tilde{\Omega}(\frac{n^{1-\frac{1}{d+1}}}{m^{\frac{1}{d+1}}} + t)$	$\tilde{\Omega}(m); \quad n \leq m \leq n^d$	$\mathcal{O}(n^{1+\epsilon} + m(\log n)^{\epsilon})$
Planar index [this work]	$\mathcal{O}(d \log n + t) \sim \mathcal{O}(dn)$	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$

Table 3: *Top-k nearest-neighbor-finding time using* Indp *dataset; # dimensions = 6, RQ=4, # index = 100*

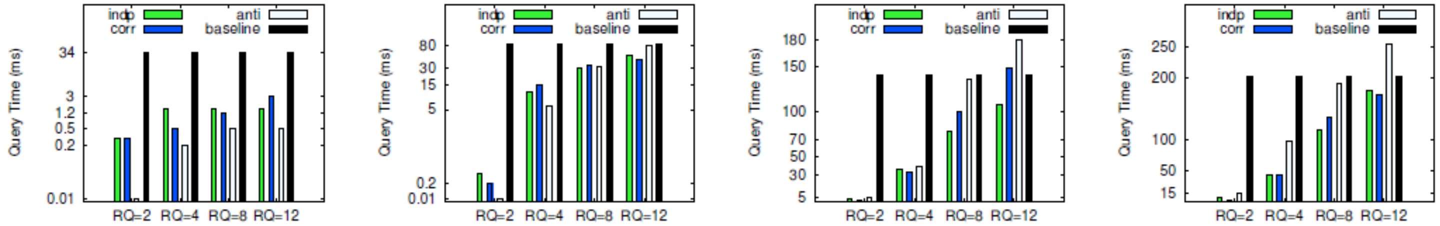
# Top-k	Checked Points/Total Points (%) [Planar index]	Query Time (ms) [Planar index]	Baseline Time (ms)
50	10.97	33	89
1000	11.29	36	89
10000	12.62	42	89

How is the performance of the proposed technique compared to that of the existing ones?



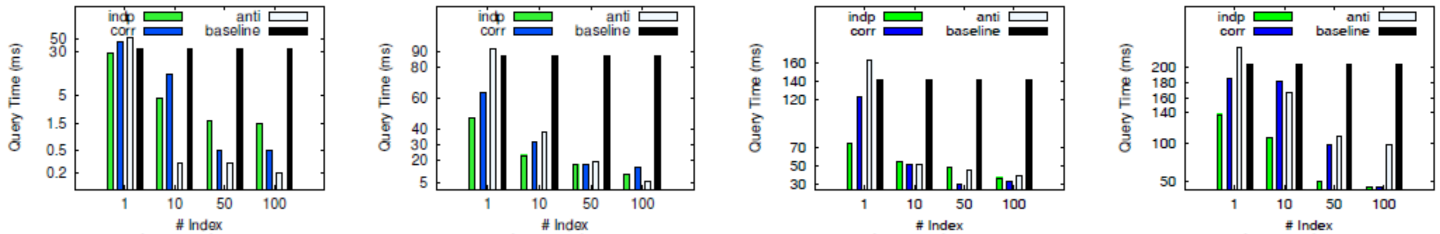
(a) Consumption (SQL func.) (b) CMoment (c) CTexture (d) Index Time

Figure 6: Index and query-processing times using real-world datasets (*Consumption*, *CMoment*, and *CTexture*)



(a) Dimension=2 (b) Dimension=6 (c) Dimension=10 (d) Dimension=14

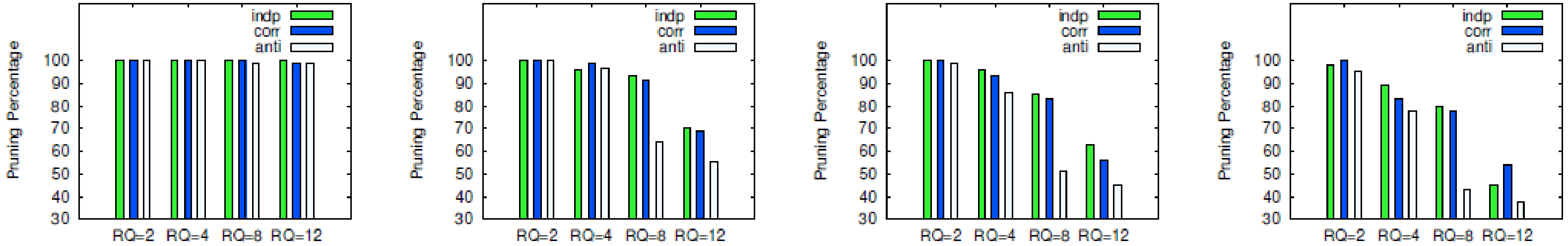
Figure 7: Query-processing time using synthetic datasets (*Indp*, *Corr*, and *Anti*): # dimensions = 2 ∼ 14, and randomness of query (RQ) varied from 2 ∼ 12, # index = 100. Baseline running times are for any of the three synthetic datasets.



(a) Dimension=2 (b) Dimension=6 (c) Dimension=10 (d) Dimension=14

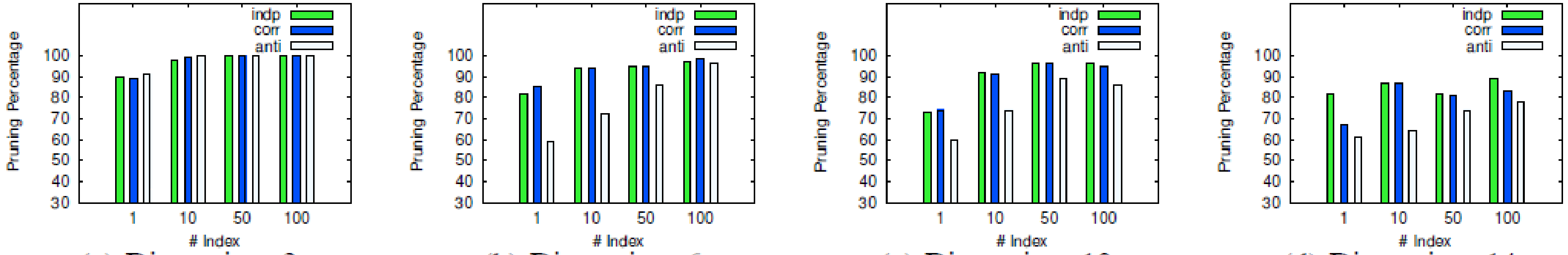
Figure 8: Query-processing time using synthetic datasets (*Indp*, *Corr*, and *Anti*): # dimensions = 2 ∼ 14 and # index = 1 ∼ 100, randomness of query (RQ) = 4. Baseline running times are for any of the three synthetic datasets.

How is the performance of the proposed technique compared to that of the existing ones?



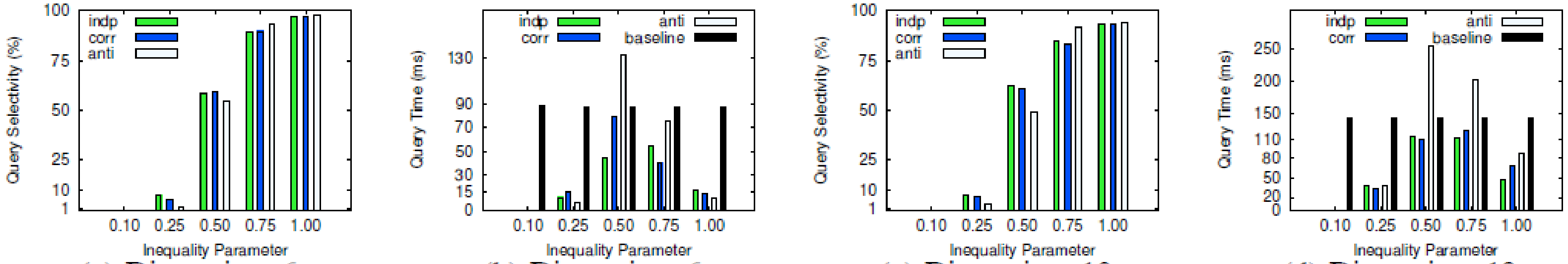
(a) Dimension=2 (b) Dimension=6 (c) Dimension=10 (d) Dimension=14

Figure 9: Pruning percentage for synthetic datasets: # dimensions = 2 ∼ 14, and randomness of query (RQ) = 2 ∼ 12, # index = 100.



(a) Dimension=2 (b) Dimension=6 (c) Dimension=10 (d) Dimension=14

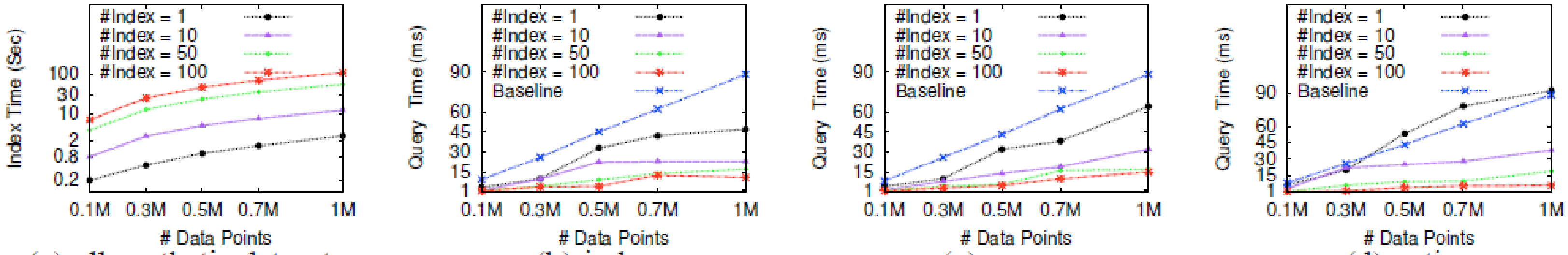
Figure 10: Pruning percentage for synthetic datasets: # dimensions = 2 ∼ 14 and # index = 1 ∼ 100, randomness of query (RQ) = 4.



(a) Dimension=6 (b) Dimension=6 (c) Dimension=10 (d) Dimension=10

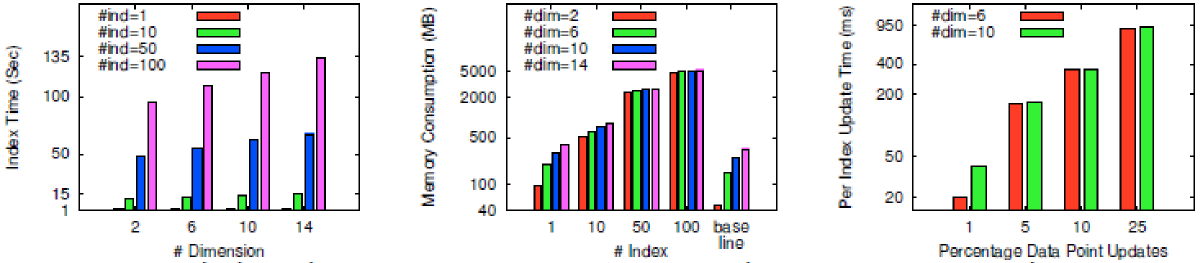
Figure 11: Query selectivity and query-processing time with varying inequality parameter: synthetic datasets, # index = 100, randomness of query (RQ) = 4. Baseline running times are for any of the three synthetic datasets.

How is the performance of the proposed technique compared to that of the existing ones?



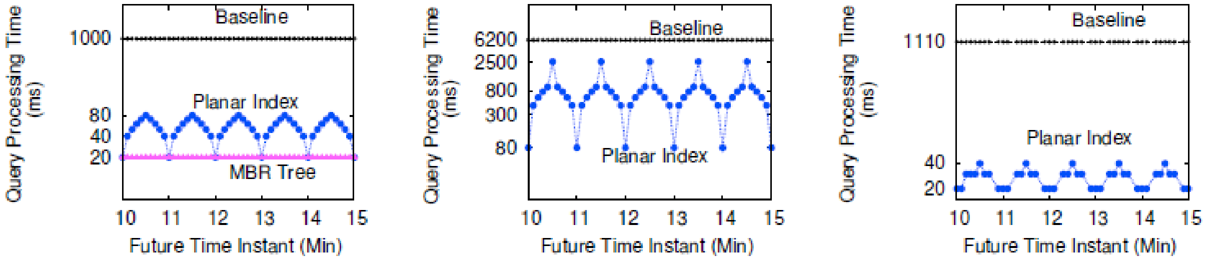
(a) all synthetic datasets (b) indp (c) corr (d) anti

Figure 12: Scalability with varying number of data points using synthetic datasets: # index= 1∼100, randomness of query (RQ) = 4, and # dimensions =6.



(a) Index Building Time (b) Memory Consumption (c) Dynamic Updates

Figure 13: Index construction time, memory usage, and dynamic index updates using synthetic datasets



(a) Linearly Moving Objects (b) Circular Moving Objects (c) Accelerating Objects

Figure 14: Planar index in finding moving object intersection: uniform and non-uniform workloads

Conclusion

1. In this paper, scalar product queries was studied — a widely applicable set of analytic queries whose parameters are known only at the time of querying.

2. Defined Planar index, a geometric approach that allows for online processing of scalar product queries in an efficient and accurate manner, as confirmed by an extensive experimental evaluation conducted on various synthetic and real-world datasets.

3. Show the applications of Planar index in the moving-objects-intersection problem and in active learning.

What do you think about this paper?

- The new indexing scheme ‘Planar’ for the scalar product queries is innovative and practical for answering online and offline queries.

- Authors also presented a sophisticated approach to compare Planar indexing and normal indexing and make Planar indexing more persuasive.

- However, how to implement the Planar indexing function under the real-world setting is still a concern need to be addressed.

Reference

- *Arijit Khan*, *Pouya Yanki*, *Bojana Dimcheva*, *Donald Kossmann*. (2014) Towards Indexing Functions: Answering Scalar Product Queries. SIGMOD '14: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data.