Towards Indexing Functions: Answering Scalar Product Queries

Arijit Khan, Pouya Yanki, Bojana Dimcheva, Donald Kossmann

Systems Group
ETH Zurich
Moving Objects
Intersection Finding

Position at a future time instance $t$

- $[x = r \cos(\omega t) \quad y = r \sin(\omega t)]$
- $[x = p + ut \quad y = q + vt]$

Moving Object Database

- $\rightarrow r, \omega$
- $\rightarrow p, q, u, v$
Moving Objects
Intersection Finding

Find all object pairs that will be within distance $S$ at time instance $t$

$$AX_1 + BX_2 + CX_3 + DX_4 + EX_5 + FX_6 + GX_7 \leq S^2$$

$$X_1 = r^2 + p^2 + q^2 + 2rp + 2rq$$
$$X_2 = 2[u(r-p) + v(r-q)]$$
$$X_3 = -2rp$$
$$X_4 = -2rq$$
$$X_5 = -2ru$$
$$X_6 = -2rv$$
$$X_7 = u^2 + v^2$$

$$A = 1$$
$$B = t$$
$$C = 1 + \sin(\omega t)$$
$$D = 1 + \cos(\omega t)$$
$$E = t[1+ \sin(\omega t)]$$
$$F = t[1+ \cos(\omega t)]$$
$$G = t^2$$
Moving Objects
Intersection Finding

Find all object pairs that will be within distance $S$ at time instance $t$

$$AX_1 + BX_2 + CX_3 + DX_4 + EX_5 + FX_6 + GX_7 \leq S^2$$

$$X_1 = r^2 + p^2 + q^2 + 2rp + 2r$$
$$X_2 = 2[u(r-p) + v(r-q)]$$
$$X_3 = -2rp$$
$$X_4 = -2rq$$
$$X_5 = -2ru$$
$$X_6 = -2rv$$
$$X_7 = u^2+v^2$$

Function (known)

Query Parameters (unknown)

$A = 1$
$B = t$
$C = 1+\sin(\omega t)$
$D = 1+\cos(\omega t)$
$E = t[1+\sin(\omega t)]$
$F = t[1+\cos(\omega t)]$
$G = t^2$
Moving Objects Intersection Finding

Find all object pairs that will be within distance $S$ at time instance $t$

$$AX_1 + BX_2 + CX_3 + DX_4 + EX_5 + FX_6 + GX_7 \leq S^2$$

Moving Object Database

- $r, \omega$ → r, ω
- $p, q, u, v$ → p, q, u, v

Scalar Product Query: $(A \ B \ C \ D \ E \ F \ G) \cdot (X_1 \ X_2 \ X_3 \ X_4 \ X_5 \ X_6 \ X_7) \leq S^2$

Query Parameters (unknown)  Function (known)  Inequality Parameter (unknown)
Scalar Product Query: \[(A \ B \ C \ D \ E \ F \ G) \cdot (X_1 \ X_2 \ X_3 \ X_4 \ X_5 \ X_6 \ X_7) \leq S^2\]

Moving Object Database

Moving Object Database

B-Tree Index will not work !!!

Query Parameters (unknown)

Function (known)

Inequality Parameter (unknown)
More Applications: Complex SQL Function

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>P2</td>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>P3</td>
<td>4</td>
<td>70</td>
</tr>
<tr>
<td>P4</td>
<td>6</td>
<td>50</td>
</tr>
</tbody>
</table>

**Patient Dataset for Heart-Rate Prediction**

**ARIMA Time Series Prediction Model:**

Heart-Rate at time $t = S \times t + B$
More Applications: Complex SQL Function

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>P2</td>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>P3</td>
<td>4</td>
<td>70</td>
</tr>
<tr>
<td>P4</td>
<td>6</td>
<td>50</td>
</tr>
</tbody>
</table>

**Patient Dataset for Heart-Rate Prediction**

ARIMA Time Series Prediction Model:

Heart-Rate at time $t = S \times t + B$

Find all patients for whom the predicted heart rate at time $t$ is more than an input threshold $H$.

CREATE FUNCTION Critical_Patient (  
    INPUT double Threshold, double Time  
RETURN PatientID  
FROM Patient  
WHERE  $S \times Time + B \geq H$ )

Unknown
More Applications: Complex SQL Function

Patient Dataset for Heart-Rate Prediction

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>P2</td>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>P3</td>
<td>4</td>
<td>70</td>
</tr>
<tr>
<td>P4</td>
<td>6</td>
<td>50</td>
</tr>
</tbody>
</table>

Scalar Product Query

(Time, 1) • (S, B) ≥ H

Unknown Function (known)
## More Applications: Complex SQL Function

### Patient Dataset for Heart-Rate Prediction

<table>
<thead>
<tr>
<th>Patient ID</th>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>P2</td>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>P3</td>
<td>4</td>
<td>70</td>
</tr>
<tr>
<td>P4</td>
<td>6</td>
<td>50</td>
</tr>
</tbody>
</table>

\[ S \times \text{Time} + B \geq H \]

Can we be more efficient than a sequential scan?
Problem Statement

- Inequality Query
  
  Find all data points $x$ that satisfy: 
  $$(a, F(x)) \geq b$$

- Top-k Nearest Neighbor Query
  
  Find the top-k data points $x$ satisfying $$(a, F(x)) \geq b$$, that also minimize: 
  $$|(a, F(x)) - b|/|a|$$

Applications:

- moving-object-intersection finding
- half-space range search
- complex SQL functions

Applications:

- top-k nearest points to hyper plane
- active learning
Related Work

- **Half-space Range Searching**
  - Agarwal et. al. [PODS '98], Matousek et. al. [Computational Geometry '92]

- **Linear Constraint Queries**
  - Goldstein et. al. [PODS '97]

- **Nearest Neighbor Queries**
  - Liu et. al. [ICML '12], Jain et. al. [NIPS '10]

- **Top-k Queries with Ranking Function**
  - Chang et. al. [SIGMOD '00], Xin et. al. [SIGMOD '07], Li et. al. [SIGMOD '05], Ilyas et. al. [ACM Comp. Survey '08], Hristidis et. al. [SIGMOD '01], Ram et. al. [KDD '12]

- **Index for Moving Objects**
  - Nascimento et. al. [R-tree, SAC '98], Sistla et. al. [ICDE '97], Kollios et. al. [PODS '99], Saltenis et. al. [TPR-Tree, SIGMOD '00], Jensen et. al. [B^x-Tree, VLDB '04], Tao et. al. [SIGMOD '02], Zhang et. al. [MBR Tree, VLDB J '12]
Related Work

**Half-space Range Searching**
- Agarwal et. al. [PODS '98], Matousek et. al. [Computational Geometry '92]

**Linear Constraint Queries**
- Goldstein et. al. [PODS '97]

**Nearest Neighbor Queries**
- Liu et. al. [ICML '12], Jain et. al. [NIPS '10], Chang et. al. [SIGMOD '00], Xin et. al. [SIGMOD '07], Li et. al. [SIGMOD '05], Ilyas et. al. [ACM Comp. Survey '08], Hristidis et. al. [SIGMOD '01], Ram et. al. [KDD '12]

**Top-k Queries with Ranking Function**
- Nascimento et. al. [R-tree, SAC '98], Sistla et. al. [ICDE '97], Kollios et. al. [PODS '99], Saltenis et. al. [B^x-Tree, SIGMOD '00], Jensen et. al. [B^x-Tree, VLDB '04], Tao et. al. [B^x-Tree, VLDB J '04], Zhang et. al. [MBR Tree, VLDB J '12]
Planar Index: Geometrical Indexing

Query Processing using Planar Index
Moving Objects
Intersection Finding

Find all object pairs that will be within distance \( S \) at time instance \( t \)

\[ AX_1 + BX_2 + CX_3 + DX_4 + EX_5 + FX_6 + GX_7 \leq S^2 \]

Scalar Product Query: \((A \ B \ C \ D \ E \ F \ G) . (X_1 \ X_2 \ X_3 \ X_4 \ X_5 \ X_6 \ X_7) \leq S^2\)

- Red nodes: \( r, \omega \)
- Green nodes: \( p, q, u, v \)

Query Parameters (unknown)  Function (known)  Inequality Parameter (unknown)
Planar Index: Geometrical Indexing

Query Processing using Planar Index
Planar Index: Geometrical Indexing

One Planar Index = Collection of Parallel Hyper-planes Passing through the Data Points

Query Processing using Planar Index
Planar Index:
Geometrical Indexing

Query: \((a, x) \geq b\)

Query Processing using Planar Index
Planar Index: Geometrical Indexing

Query: $(a, x) \geq b$

Query Processing using Planar Index
Planar Index: Geometrical Indexing

Query: \((a, x) \geq b\)

Query Processing using Planar Index
Planar Index: Geometrical Indexing

Query: \((a, x) \geq b\)

Verify

Query Processing using Planar Index
Planar Index: Geometrical Indexing

Query: \((a, x) \geq b\)

Query Processing using Planar Index

Accept

Verify

Reject
Planar Index: Time and Space Complexity

Index Time: $O(n \log n)$

Index Space: $O(n)$

Query Processing Time: $O(d \log n + t) \sim O(d n)$
Multiple Planar Indices

Query: \((a, x) \geq b\)
Best Index Selection at Query Time

Planar Index at Right is Better for the Given Query
Top-k Nearest Neighbor Query

Query: \((a, x) \geq b; \text{ Top-k Closest Points to Query Hyper plane}\)

Top-k Nearest Neighbor Query
Top-k Nearest Neighbor Query

Query: \((a, x) \geq b\); Top-2 Closest Points to Query Hyper plane

Reject

Top-k Nearest Neighbor Query
Top-k Nearest Neighbor Query

Query: \( (a, x) \geq b \); Top-2 Closest Points to Query Hyper plane

Verify

Reject

Top-k Nearest Neighbor Query

Process \( X_6, X_5 \)

Top-2 Buffer

\( X_5 \)
Top-k Nearest Neighbor Query

Query: \((a, x) \geq b;\) Top-2 Closest Points to Query Hyper plane

Lower Bound Distance

Top-k Nearest Neighbor Query
Top-k Nearest Neighbor Query

Query: \((a, x) \geq b\); Top-2 Closest Points to Query Hyper plane
Top-k Nearest Neighbor Query

Query: \((a, x) \geq b; \) Top-2 Closest Points to Query Hyper plane

Process \(X_2\)

Top-2 Buffer

Lower Bound Distance
Top-k Nearest Neighbor Query

Query: \((a, x) \geq b; \) Top-2 Closest Points to Query Hyper plane

- **Lower Bound Distance**
- **Prune**
- **Reject**

Top-k Nearest Neighbor Query

Process \(X_2\)

Top-2 Buffer

\[
\begin{array}{c|c}
X_3 \\
X_4 \\
\end{array}
\]
List of Experiments

**Datasets:**
- **Real-World:** CMoment, Ctexture, Electricity Consumption
- **Synthetic:** Independent, Correlated, Anti-Correlated

**List of Experiments:**
- Efficiency vs. No of Index
- Efficiency vs. No of Dimension
- Efficiency vs. Randomness of Query
- Efficiency vs. Query Selectivity
- Pruning Capacity vs. No of Index
- Pruning Capacity vs. No of Dimension
- Pruning Capacity vs. Randomness of Query
- Pruning Capacity vs. Query Selectivity
- Scalability of Index Building, Query Processing
- Dynamic Index Updating
- Memory Usage of Planar Index

**Experimentally Evaluated Planar Index in:**
- Moving-Object Intersection
- Top-k Nearest Neighbor Query
Dataset and Query

Datasets:

<table>
<thead>
<tr>
<th>Dataset Type</th>
<th># Data Points</th>
<th># Dimension</th>
<th># Attribute Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMoment (Real-World)</td>
<td>68,040</td>
<td>9</td>
<td>(−4.15, 4.59)</td>
</tr>
<tr>
<td>Independent (Synthetic)</td>
<td>1,000,000</td>
<td>2 - 14</td>
<td>(1, 100)</td>
</tr>
</tbody>
</table>

Query:

\[ Q_1 X_1 + Q_2 X_2 + \ldots + Q_d X_d \geq 75 \left( Q_1 + Q_2 + \ldots + Q_d \right) \]

Randomness of Query (QR): \( Q_i \in (1,n) \)
Efficiency (Real-World Dataset)

# Dimension = 9
# Index = 100

1.13 ~ 4.5 times better than Baseline
Efficiency (Synthetic Dataset)

# Index = 100

**Dimension = 2**

12 ~ 170 times better than Baseline

**Dimension = 6**

1.6 ~ 400 times better than Baseline
Application: Moving Object Intersection

Intersection Finding among $5K \times 5K$ Moving Objects

Objects moving with uniform velocity

12 ~ 50 times better than Baseline

Objects moving with acceleration

27 ~ 55 times better than Baseline
Conclusion

- Scalar product query widely applicable
- Planar index – one generalized index for many problems
- Application in moving object intersection finding

**Future Work:** Dynamic updates in planar indices based on past query workload

**Software and Dataset:** [http://people.inf.ethz.ch/khana/software/scalar.tar.gz](http://people.inf.ethz.ch/khana/software/scalar.tar.gz) (Publicly Available)