Top-k Selection Queries over Relational Databases: Mapping Strategies and Performance Evaluation

NICOLAS BRUNO\textsuperscript{1} \quad SURAJIT CHAUDHURI\textsuperscript{2} \quad LUIS GRAVANO\textsuperscript{1}

\textsuperscript{1}Columbia University
\textsuperscript{2}Microsoft Research

Presented by: Romans Kasperovics, Ivan Zorzi
Outline

1. Introduction
2. Query Model
3. Static Evaluation Strategies
4. Dynamic Strategy
5. Experimental Settings
6. Experimental Results
7. Limitations and Alternatives
Top-k selection query: it retrieves only a small number of tuples $k$ that best match a given selection condition.

The answer to a top-k query: an ordered set of tuples, where the ordering criterion is how well each tuple matches the given query.

The quality of the match is determined by a given distance function.
Real-estate database
- House(Price, Number_of_Bedrooms)

Customer query:
- Houses with 4 bedrooms and price around $300,000

The database system should:
- Rank all the available houses taking into account the user preference
- Return the top k houses for the user to inspect

If there is no exact match:
- Return the k tuples that are closest to the query
Top-k queries are not yet effectively supported by most RDBMSs

Key challenges:
- avoid sequential scan of the data
- provide this functionality efficiently for a wide variety of distance functions
- improve the optimization of top-k queries using
  - existing data structures (i.e. indexes)
  - statistics (e.g. histograms)
Introduction

Goal

- **Not** to develop stand-alone algorithms or data structures for the nearest-neighbor problem over multidimensional data
- Mapping a top-k selection query to a traditional range selection query that can be optimized and executed by any standard RDBMS
The technique must be *smart* enough to grant that:
- The k closest matches are likely to be **included** in the answer to the generated range query

But...
- If the range selection query returns fewer than k tuples, the query needs to be **restarted**
- If the range selection query returns too many tuples, a lot more than k tuples need to be **compared** with the query
Query Model

Overview

- Example
- SQL-like Notation
- Distance Functions
Query Model

Example

- Employee(Age, Hourly_Wage)
- Query q = (30, 20)
- Answer to a top-10 selection is:
  - An ordered sequence consisting of 10 employees in the Employee relation that are closest to
    - 30 years of age
    - making an hourly wage of $20
  according to a given distance function
The distinguishing feature is the `ORDER BY` clause.

We are interested in only the k answers that best match the given `WHERE` clause.
Distance functions

Definition

- Given a top-k query $q$ and a distance function $Dist$, the database system with relation $R$ uses $Dist$ to determine how closely each tuple in $R$ matches the target values $q_1, \ldots, q_n$ specified in query $q$.
- Given tuple $t$ and query $q$, $Dist(q, t)$ is a positive real number.
Distance functions
Metrics

The following distance metrics are adopted:

\[
\begin{align*}
\text{Sum}(q, t) &= \| q - t \|_1 = \sum_{i=1}^{n} |q_i - t_i| \\
\text{Eucl}(q, t) &= \| q - t \|_2 = \sqrt{\sum_{i=1}^{n} (q_i - t_i)^2} \\
\text{Max}(q, t) &= \| q - t \|_{\infty} = \max_{i=1}^{n} |q_i - t_i|
\end{align*}
\]
Distance functions

Example

- Employee(Age, Hourly_Wage)
- tuple \( t = (50, 35) \)
- query \( q = (30, 20) \)
- \( t \) will have a distance of:
  - \( \text{Max}(q, t) = \max(|30 - 50|, |20 - 35|) = 20 \)
  - \( \text{Eucl}(q, t) = \sqrt{(30 - 50)^2 + (20 - 35)^2} = 25 \)
  - \( \text{Sum}(q, t) = |30 - 50| + |20 - 35| = 35 \)
Distance functions

Distribution of Distances

(a) Sum

(b) Eucl

(c) Max
Consider a relation $R$ and a distance function $Dist$ defined over $R$. Let $q = (q_1, \ldots, q_n)$ be a top-k query over $R$, and let $t = (t_1, \ldots, t_n)$ and $t' = (t'_1, \ldots, t'_n)$ be two arbitrary tuples in $R$ such that $\forall i \ |t'_i - q_i| = |t_i - q_i|$. (In other words, $t'$ is at least as close to $q$ as $t$ for all attributes.) Then, $Dist(q, t') = Dist(q, t)$.
Query Processing Strategy

Steps

- **Search**: Given a top-k query \( q \) over \( R \), use a multidimensional histogram \( H \) to estimate a search distance \( d_q \), such that the region \( \text{reg}(q, d_q) \) that contains all possible tuples at distance \( d_q \) or lower from \( q \) is expected to include \( k \) tuples.

- **Retrieve**: Retrieve all tuples in \( \text{reg}(q, d_q) \) using a range query that encloses this region as tightly as possible.

- **Verify/Restart**: If there are at least \( k \) tuples in \( \text{reg}(q, d_q) \), return the \( k \) tuples with the lowest distances. Otherwise, choose a higher value for \( d_q \) and restart the procedure.
There are different strategies to identify the search distance $d_q$.

Ideally, the search distance $d_q$ should enclose exactly $k$ tuples.

In practice: try to find a value of $d_q$ such that $reg(q, d_q)$ encloses at least $k$ tuples, but not many more.
Choice of $d_q$ is guided by some statistics about relation $R$:

- $n$-dimensional histogram $H$ describes the distribution of values of $R$

$$H = \{(b_1, f_1), \ldots, (b_m, f_m)\}$$

where,

- each bucket $b_i$ defines a hyper rectangle included in $\text{domain}(R)$
- each frequency $f_i$ is the number of tuples in $R$ that lie inside $b_i$
Static Evaluation Strategies

Procedure

- $d_q$ is chosen as follows:
  - Create a small *synthetic* relation $R'$ which has one distinct tuple for each bucket in $H$
  - Compute $\text{Dist}(q, t)$ for every tuple $t$ in $R'$
  - $d_q = \max_{t \in T} \text{Dist}(q, t)$, where $T$ is the set of closest $k$ tuples in $R'$ for $q$
Static Evaluation Strategies

Strategies

- NoRestarts
- Restarts
- Inter1
- Inter2
Static Evaluation Strategies

NoRestarts

- Search distance $d_{NR_q}$ is high enough to guarantee that no restarts are ever needed
- **Verify/Restart** step always finishes successfully, without ever having to enlarge $d_q$ and restart the process
- $t_b$ is a tuple in $b$’s $n$-rectangle with the following property:
  
  $$\text{Dist}(q, t_b) = \max_{t \in T_b} \text{Dist}(q, t)$$

  where $T_b$ is the set of all potential tuples in the $n$-rectangle associated with $b$.

- **LEMMA**: Let $q$ be a top-k query over a relation $R$. Let $d_{NR_q}$ be the search distance computed by strategy NoRestarts for query $q$ and distance function $\text{Dist}$. Then, there are at least $k$ tuples $t$ in $R$ such that $\text{Dist}(q, t) \leq d_{NR_q}$. 
Static Evaluation Strategies

Restarts

- Search distance $d_{R_q}$ is the lowest among those search distances that might result in no restarts.
- $d_{R_q}$ is the lowest distance that might result in no restarts in the Verify/Restart step.
- $t_b$ is a tuple in $b$’s $n$-rectangle with the following property:
  \[
  \text{Dist}(q, t_b) = \min_{t \in T_b} \text{Dist}(q, t)
  \]
  where $T_b$ is the set of all potential tuples in the $n$-rectangle associated with $b$.
- LEMMA: Let $q$ be a top-k query over a relation $R$. Let $d_{R_q}$ be the search distance computed by strategy Restarts for query $q$ and distance function $\text{Dist}$. Then, there are fewer than $k$ tuples $t$ in $R$ such that $\text{Dist}(q, t) < d_{R_q}$.
Static Evaluation Strategies

Example 1/3

- Employee(Age, Hourly_Wage)
- query \( q = (20, 15) \)
- Histogram \( H \) with three buckets, \( b_1, b_2, \) and \( b_3 \)
  - \( b_1 = 40 \)
  - \( b_2 = 5 \)
  - \( b_3 = 15 \)
- **NoRestart** is used to build \( Employee' \)
- \( Employee' \) will consist of three tuples \( t_1, t_2, \) and \( t_3 \) (which are as far from \( q \) as their corresponding bucket boundaries permit)
  - \( f_{t_1} = 40 \)
  - \( f_{t_2} = 5 \)
  - \( f_{t_3} = 15 \)
- **Max** distance function is used to find the top 10 tuples for \( q \)
  - \( Max(q, t_1) = 35 \)
  - \( Max(q, t_2) = 20 \)
  - \( Max(q, t_3) = 30 \)
We need tuple $t_2$ (frequency 5) and tuple $t_3$ (frequency 15) to get the top-10 tuples.

Therefore: search distance $d_{NR_q}$ will be $\text{Max}(q, t_3) = 30$.

The original relation Employee is guaranteed to contain at least 10 tuples with distance $d_{NR_q} = 30$ or lower to query $q$. 
Static Evaluation Strategies

Example 3/3
Static Evaluation Strategies

Inter1 & Inter2

- \( \text{Inter1} = (2dR_q + dNR_q)/3 \)

- \( \text{Inter2} = (dR_q + 2dNR_q)/3 \)
Dynamic Strategy

- Static strategy Restart: \( d_q = dR_q \)
- Static strategy NoRestart: \( d_q = dNR_q \)
- Static strategy Inter1: \( (2dR_q + dNR_q)/3 \)
- Static strategy Inter2: \( (dR_q + 2dNR_q)/3 \)
- Dynamic strategy:

\[
d_q(\alpha) = dR_q + \alpha \cdot (dNR_q - dR_q), \quad 0 \leq \alpha \leq 1
\]
Dynamic Strategy

Optimum $\alpha$

- For each query $q$ there’s an optimum $\alpha_q$ 😊
- It’s not possible to determine $\alpha_q$ without scanning the data 😞
- Workload $Q = \{q_1, \ldots, q_m\}$ of similar top-k queries
- We look for single $\alpha^*$ that minimizes average error for the whole $Q$ and similar workloads

$$\text{totalTuples}(Q, \alpha) = \sum_{q_i \in Q} \left( \text{tuples}(q_i, d_{q_i}(\alpha)) + \begin{cases} 0, & \text{if } \text{tuples}(q_i, d_{q_i}(\alpha)) \geq k \\ \text{tuples}(q_i, d_{\text{NR}q_i}), & \text{otherwise} \end{cases} \right)$$
Dynamic Strategy
Minimization of $\text{totalTuples}$

- Suppose workload $Q$ is fixed
- Find $\alpha$ for which $\text{totalTuples}(Q, \alpha)$ reaches its minimum
Dynamic Strategy

Approximation of \( tuples(q, d) \)

- To find a minimum of \( totalTuples(Q, \alpha) \) we need to calculate \( tuples(q_i, d_{qi}(\alpha)) \)
- Calculation of \( tuples(q, d) \) is expensive
- Approximation \( tuples'(q, d) \) is proposed as

\[
tuples'(q, d_q(\alpha)) = T_q^{I} + \alpha \left( T_q^{I+1} - T_q^{I} \right)
\]

where

\[
T_q^{i} = tuples \left( q, dR_q + \frac{i}{G} (dNR_q - dR_q) \right), \ i \in \{0, 1, \ldots, G\}
\]
Dynamic Strategy

Approximation of $tuples(q,d)$
Dynamic Strategy

Interpolation $tuples'(q, d_q(\alpha)) = T^T_q + \alpha (T^T_{q+1} - T^T_q)$
Dynamic Strategy

Algorithm to calculate $T_q^i$

Procedure **calculate**T (D: Data Set, Q: Workload, G: integer)

Set $\tau_j^k = 0$, for $j \in \{0, 1, \ldots, |Q|\}$ and $k \in \{0, 1, \ldots, G\}$

for each tuple $t_i$ in D // Sequential scan over D

for each query $q_j$ in Q

\[
d = \text{Dist}(t_i, q_j)
\]

if ($d \leq dR_{q_j}$) $\tau_j^0 ++$ // we count $t_i$ in $\tau_j^0$

else if ($d \leq dNR_{q_j}$)

\[
g = \left[ G \cdot \frac{d-dR_{q_j}}{dNR_{q_j}-dR_{q_j}} \right] // 0 < g \leq G
\]

$\tau_j^g ++$

// At this point, $T_{q_j}^k = \sum_{k'=0}^k \tau_j^{k'}$

Calculate and return all $T_{q_j}^k$, values
Dynamic Strategy
Sampling while Calculating $tuples'(q,d)$
Experimental Settings

Data Sets

- **Real-world**
  - Census2D and Census3D – two- and three-dimensional projections of a fragment of US Census Bureau data (210,138 tuples in each)
  - Cover4D – four-dimensional projection of the CovType data set, used for predicting forest cover types from cartographic variables (545,424 tuples)

- **Synthetic**
  - *Gauss* – predetermined number of overlapping multidimensional Gaussian bells (500,000 tuples)
  - *Array* – random grid, where frequencies are generated according to a Zipfian distribution and assigned to randomly chosen cells (500,000 tuples)
Experimental Settings

Data Sets (2)

(a) Census2D  
(b) Gauss  
(c) Array
Experimental Settings

Workloads

(a) Census2D data set.  (b) Biased workload.  (c) Uniform workload.

- 100 generated queries in each workload
Experimental Settings
Evaluation Techniques

- **Optimum Technique** – ideal guess of $d_q$ (practically achieved examining all the date)

- **Histogram-Based Techniques** (static and dynamic mapping strategies):
  - Restart
  - Inter1
  - Inter2
  - NoRestart
  - Dynamic

- Techniques Requiring Sequential Scans
Experimental Settings

Metrics

- Percentage of Restarts
- SOQ (Successful Original Query) Time – majority top-k queries do not require restart, so it makes sense to show this time separately
- IOQ (Insufficient Original Query) Time – average increase due to restart
- Number of tuples retrieved
Experimental Settings

Other Settings

- Histograms
  - Equi-Depth (multidimensional version)
  - MHist
- Indexes
  - Single column B+-tree indexes
  - Multi-column B+-tree index
- $k = 100$
- Distance function = Max
Validity of General Approach

The number of tuples included in an n-rectangle, enclosing actual top-100
Analysis and Comparison of Techniques

Comparison of different distance functions

(a) Execution time.

(b) Tuples retrieved.
Analysis and Comparison of Techniques
Time and tuples for biased workloads

(a) Execution time.
(b) Tuples retrieved.
Analysis and Comparison of Techniques
Time and tuples for uniform workloads

(a) Execution time.

(b) Tuples retrieved.
Analysis and Comparison of Techniques
Effect of use of different histograms (Equi-depth vs. MHist)

(a) Execution time (Biased workload).
(b) Tuples retrieved (Biased workload).
(c) Execution time (Uniform workload).
(d) Tuples retrieved (Uniform workload).
Analysis and Comparison of Techniques
Sensitivity to the data skew

(a) Execution time.

(b) Tuples retrieved.
Analysis and Comparison of Techniques
Effect of changing dimensionality

(a) Execution time.

(b) Tuples retrieved.
Analysis and Comparison of Techniques

Effect of changing $k$

[Graphs showing the analysis of $k$ selection queries for different techniques, including Gauss, Array, Census2D, Census3D, and Cover4D.]
Analysis and Comparison of Techniques

Effect of different indexes (multidimensional vs. unidimensional)

(a) Biased workload.

(b) Uniform workload.
Comparison with Sampling Based Techniques

(a) Biased workload.

(b) Uniform workload.
Limitations and Alternatives

Limitations:
- Only real-value attributes and distance functions are examined
- Default distance function is Max, when Euclidean distance seems to be more popular
- Default workload is biased, when uniform seems to be more popular
- Limited number of dimensions (4)

Alternatives:
- Sampling Based Techniques