Ranked Query Processing: 
a) Order-based Paradigm

Kevin Chen-Chuan Chang

Ranking—Ordering according to the degree of some fuzzy notions:
- Similarity (or dissimilarity)
- Relevance
- Preference

Query models for order-based paradigm—On the better-than graph
- Better-than graph
- Best-Matches-Only (BMO) query model
  - Retrieve maximal elements
  - Thus also called maximal vector
  - These maximal elements form the “skyline”!
- On better-than graph, how to process BMO?

When multiple dimensions are available--
- Assume the database stores the information of a set of flights
- For each flight
  - Its price
  - Its route (travel-time or distance traveled)
- A user would retrieve all the “interesting” flights
  - A flight is interesting if and only if there is no other cheaper and shorter (route) at the same time
The overall preference combines the dimensions

- P1 LOWEST(price)
  - a → b → i → c → h → g → d → m → f → n → k → l
- P2 LOWEST(distance)
  - k → m → i → h → n → f → e → d → c → a → b → e
- \( P := \{\text{price, distance}\}, \langle P1, P2 \rangle \)

BMO: Maximal elements of P?

- Is a maximal?
- Is b maximal?
- Is c maximal?

Distance | Price
---------|-------
\( a \)  | 1     |
\( b \)  | 2     |
\( c \)  | 3     |
\( d \)  | 4     |
\( e \)  | 5     |
\( f \)  | 6     |
\( g \)  | 7     |
\( h \)  | 8     |
\( i \)  | 9     |
\( j \)  | 10    |

Skyline Operation

- Dominance:
  - A point dominates another point if it is no worse in all dimensions, and better in at least one dimension

- Skyline:
  - A set of all points in the dataset that are not dominated by any other point in the dataset

Why is it called “skyline”?

Also called: Pareto curve, Maximum Vector

- What do you see in the Chicago skyline?

What is skyline: An example

- Query:
  - \text{SELECT * FROM flights}
  - \text{SKYLINE OF price MIN, distance MIN}
- What dominates what?
- What points constitute the skyline?
Skyline Algorithms: We will look at a few examples

- Block nested loop (BNL)
- Divide and Conquer
- Bitmap
- NN

Block Nested Loop (Börzsönyi et al., 2001)

- Conceptually: Nested loop joins—
  - Joining the table with itself
  - Compare every pair of points to check dominance

<table>
<thead>
<tr>
<th>Price Distance</th>
<th>Price Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>a 1 9</td>
<td>a 1 9</td>
</tr>
<tr>
<td>a 4 8</td>
<td>a 4 8</td>
</tr>
<tr>
<td>a 5 8</td>
<td>a 5 8</td>
</tr>
<tr>
<td>a 7 9</td>
<td>a 7 9</td>
</tr>
<tr>
<td>a 9 10</td>
<td>a 9 10</td>
</tr>
<tr>
<td>b 10 4</td>
<td>b 10 4</td>
</tr>
<tr>
<td>c 2 10</td>
<td>c 2 10</td>
</tr>
<tr>
<td>d 1 10</td>
<td>d 1 10</td>
</tr>
<tr>
<td>g 3 6</td>
<td>g 3 6</td>
</tr>
<tr>
<td>h 5 8</td>
<td>h 5 8</td>
</tr>
<tr>
<td>i 3 4</td>
<td>i 3 4</td>
</tr>
<tr>
<td>k 4 6</td>
<td>k 4 6</td>
</tr>
<tr>
<td>n 8 3</td>
<td>n 8 3</td>
</tr>
</tbody>
</table>

Block Nested Loop -- Implementation

- One-pass scan:
  - Scan the table, maintain a window of current skyline points
  - Return the window at the end

<table>
<thead>
<tr>
<th>Scan</th>
<th>Skyline Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scan</th>
<th>Skyline Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

Any problems?

Block Nested Look- Improvements

How if the window overflow?

- Multi-pass algorithm
  - Scan the table, write any overflow to temp file
  - Scan the temp file; repeat till done

<table>
<thead>
<tr>
<th>Scan</th>
<th>Pass 1</th>
<th>Pass 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>tempfile</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>tempfile</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>tempfile</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>tempfile</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>tempfile</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>tempfile</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>tempfile</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>tempfile</td>
</tr>
</tbody>
</table>

Scan TempFile
Block Nested Look – Improvements

How if the window overflow? [B"orzs"onyi et al., 2001]

- Divide and conquer
  - Divide all the points into several groups such that each group fits in memory
  - Process the groups separately
  - Merge their results

- Smart merging possible
  - If s3 not empty then disregard s2
  - Use s3 to purge s1, s4

However, BN L-based approaches are not incremental – Want progressive processing!

Desired:
- Compute the first few Skyline points almost instantaneously
- Compute more and more results incrementally

Bitmap Algorithm: Representation [Tan et al., 2001]

- For each dimension:
  - n distinct values \( \rightarrow \) n bits
  - A value as a bitmap of all no-higher bits = 1

<table>
<thead>
<tr>
<th>d1: price</th>
<th>d2: dist</th>
<th>d3: rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (1,1,2)</td>
<td>0 0 0 1 0 0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>b (3,2,1)</td>
<td>0 1 1 1 0 1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>c (4,1,1)</td>
<td>1 1 1 0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>d (2,3,2)</td>
<td>0 0 1 1 1 1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

Is b = (3, 2, 1) in the skyline?

- Any point with no-worse values in all dimensions?
  - 0110 & 0101 & 1111 = 0100
- Any point with a better value in some dimension?
  - 0010 | 0001 | 1001 = 1011
- Any point satisfying both?
  - 0100 & 1011 = 0000
- So, is b = (3,2,1) in the skyline?

<table>
<thead>
<tr>
<th>d1: price</th>
<th>d2: dist</th>
<th>d3: rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (1,1,2)</td>
<td>0 0 0 1 0 0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>b (3,2,1)</td>
<td>0 1 1 1 0 1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>c (4,1,1)</td>
<td>1 1 1 0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>d (2,3,2)</td>
<td>0 0 1 1 1 1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>
The Bitmap Algorithm

- for each point \( x \) in DB:
  - check if \( x \) is in skyline
  - output \( x \) if so
- Incremental indeed; bitmap computation efficient
- However, any problem?

Bitmap Algorithm: Problems

- Bitmaps are not dynamic structures
  - Hard to update
- Bitmaps can have prohibitive space overhead
  - How if there are many distinct values?
    - E.g., How about continuous values?
- No focus of directions at all in skyline search
  - Depend on what points you check first

NN – Finding the First Skyline Point [Kossmann et. al. 2002]

- Start by finding the nearest neighbor of the origin
  - I.e., the point \( p = (x, y) \) with the smallest \( \sqrt{x^2 + y^2} \)
  - How to find NN: Use NN algorithm based on R-tree.
- This NN point must be in the skyline
  - Otherwise?

NN – Are there other skyline points?

- Pruning -- What cannot be in the skyline?
  - Those dominated by point \( I \)
- Iteration – What may be in the skyline?
  - Non-dominated region 2 and 3
NN – Iteratively Process All the “ToDo” Regions until All Done

Order-based rank query evaluation – Still ongoing research.
- How optimal are these algorithms? Further improvement?
- Scale to high dimensionality?
- Generalize to non-BMO type of aggregations?

Thank You!

 Ranking Query Processing: 
b) Score-based Paradigms

Kevin Chen-Chuan Chang
Ranking– Ordering according to the degree of some fuzzy notions:
- Similarity (or dissimilarity)
- Relevance
- Preference

Relational DBMS scenarios– A brief overview

Relational DBMS–
- Value mapping: [Chaudhuri and Gravano, 1999]
  - Mapping top-k scores to Boolean selection ranges
  - May have to restart
- Cardinality mapping: [Carey and Kossman, 1997, 1998]
  - Pushing “limit k” down query tree
  - May have to restart

Our Focus: Middleware scenarios

Top-k algorithms rely on accesses to evaluate query scores
To each predicate $p_i$:
- Random access $RA(u_j)$
  - Return score of $u_j$ for $p_i$
- Sorted access $SA$
  - Return some next best object and its score for $p_i$
An algorithm performs a sequence of accesses:
A simple algorithm

- Sorted access on P1 then random accesses to P2, P2

\[
c_{80} = \min(p_1, p_2, p_3)
\]

Goal: Minimize the "access" cost

Access costs dominate in "middleware" scenarios

Cost model: aggregate of all access costs

Assumption: Monotonic scoring functions

- Monotonic:
  - \( f(x_1, \ldots, x_n) \leq f(x_1', \ldots, x_n') \) if \( x_i \leq x_i' \) for all \( i \)

- Why good for query evaluation?
  - Gives bounds for pruning data
  - Gives a simple function "surface" to maximize \( f \)

- Reasonable?
  - Analogy: Negation rarely used in Boolean queries
  - But, new "function-inference" front-ends also found this to be violated in many cases

The Naïve Algorithm

- Get all \( p_i[u] \) score for every object \( u \)
  - e.g., by complete sorted accesses
- Compute \( F[u] = F(p_1[u], \ldots, p_m[u]) \) for every \( u \)
- Sort
- Return top \( k \)

- Obviously expensive. Can we do better?
  - Note \( k \) is typically small

}\end{itemize}
FA– Fagin’s Algorithm (or the “First Algorithm”) [Fagin, 1996] [Wimmers et al., 1999]

Scenario: Sorted + Random Access Available
- Go in the lists with SA in parallel
- Do complete RA for every seen object to complete scores
- Maintain a buffer of current top-k objects
- Maintain a threshold $T$:
  - Upper-bound for all the unseen objects
- Stop:
  - When all lists so far share at least $k$ objects
  - Return the current top-k objects

Why is FA correct?
- At stop time, all seen objects are compared
- Can unseen objects have higher scores?
  - e.g., How about object 4? Upper bound?

How is FA “optimal”? Can you make it more efficient?
- FA:
  - For string, monotone $F$, sorted accesses optimal up to a constant factor, with high probability.
  - Can you stop earlier than round 3?
Then, there have been various algorithms, for different scenarios...

<table>
<thead>
<tr>
<th>Sorted Access</th>
<th>Random Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = 1$ (cheap)</td>
<td>$r = 1$ (cheap)</td>
</tr>
<tr>
<td>$s = 1$ (cheap)</td>
<td>FA, TA, QuickCombine</td>
</tr>
<tr>
<td>$s = h$ (expensive)</td>
<td>$r = h$ (expensive)</td>
</tr>
<tr>
<td>$s = h$ (expensive)</td>
<td>CA, SR-Combine</td>
</tr>
<tr>
<td>$s = \infty$ (impossible)</td>
<td>$r = \infty$ (impossible)</td>
</tr>
<tr>
<td>$s = \infty$ (impossible)</td>
<td>NRA, StreamCombine</td>
</tr>
</tbody>
</table>

Improving FA: TA [Fagin et al., 2001], Quick-combine [Guentzer et al., 2000], Multi-step [Nepal and Ramakrishna, 1999]

Scenario: Sorted + Random Access Available
- Go in the lists with SA in parallel
- Do complete RA for every seen object to complete scores
- Maintain a buffer of current top-k objects
- Maintain a threshold $T$
  - Upper-bound for all the unseen objects
- Stop:
  - When all current top-k objects scored greater than $T$
- Return these objects as top-k

Why is TA correct?
- At stop time, all seen objects are compared
- Can unseen objects have higher scores?
  - e.g., How about object 4? Upper bound?

$F = p_1 + p_2$

Buffer

Threshold $T = .80$
Observations: Any Problem with TA?

- How does it handle SA?
  - Equal-depth parallel SA to every list

- How does it handle RA?
  - Exhaustive RA for every seen object
  - How if RA expensive? (Algorithm CA)
  - How if RA not possible? (Algorithm NRA)

How if random accesses not supported?

- The combined score of an object has two parts:
  - Upper bound score:
    - From seen exact scores and unseen max score
  - Lower bound score:
    - From seen exact scores and unseen min score

- An object is in top-k if
  - Its lower bound score is greater than the upper bound scores of all unseen objects

In contrast, how if sorted accesses not possible?

Scenario: When SA not supported

- Perform random “probes” when necessary
  - The object with current highest score
- Schedule predicates to minimize probes
- Return an object as top-k when
  - It is fully probed
  - Its score is higher than the (upper bounds of) the rest not in top-k
When sorted accesses not possible

Upper-bound of the unseen scores

<table>
<thead>
<tr>
<th>ID</th>
<th>x</th>
<th>p_1</th>
<th>p_2</th>
<th>(\min(x, p_1, p_2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.90</td>
<td>0.60</td>
<td>0.75</td>
<td>b: 0.78</td>
</tr>
<tr>
<td>b</td>
<td>0.95</td>
<td>0.70</td>
<td>0.78</td>
<td>a: 0.75</td>
</tr>
<tr>
<td>c</td>
<td>0.75</td>
<td>0.70</td>
<td>0.70</td>
<td>b: 0.78</td>
</tr>
<tr>
<td>d</td>
<td>0.60</td>
<td>0.90</td>
<td>0.90</td>
<td>a: 0.75</td>
</tr>
<tr>
<td>e</td>
<td>0.50</td>
<td>0.70</td>
<td>0.80</td>
<td>b: 0.78</td>
</tr>
</tbody>
</table>

Candidates Queue

MPro [Chang and Hwang, 2002], Upper [Bruno et. al. 2002] –

Probes optimization: Is the cost of random probes minimal?

- What object to probe next?
  - By necessary probes to analytically determine [Chang and Hwang, 2002]
    - Current top object must be further probed (by any algorithm)
  - For such object, what predicate to probe next?
    - MPro: Global scheduling – one schedule for all
      - Cost-based optimization based on selectivity and cost
    - Upper: Local scheduling – schedule for each obj
      - Use expected scores of unknown objects

So, what do we have so far...

<table>
<thead>
<tr>
<th>Sorted Access</th>
<th>Random Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>s = 1</td>
<td>r = 1 (cheap)</td>
</tr>
<tr>
<td>s = h</td>
<td>TA, QuickCombine</td>
</tr>
<tr>
<td>s = (\infty)</td>
<td>TA, QuickCombine</td>
</tr>
</tbody>
</table>

What do you think?

Challenge: Various Cost Scenarios

- Vary in capabilities
- Vary in costs:
  - over sources
  - over access types
  - over time

Thus requires “generality” over cost scenarios and “adaptivity” to the given runtime setting
Score-based ranked query evaluation – Still ongoing research

- A unified algorithms for all?
  - Currently: ad-hoc algorithms for each scenario
  - Do not cover all scenarios

- How optimal are these algorithms?
  - Cost-based optimization studied at MPro

- Unified, cost-based optimization?

Thank You!