Query Optimization
Part II

CPS 216
Advanced Database Systems

Announcements (April 19)
- Homework #4 (last one; short) will be assigned this Thursday
- Homework #3 graded; grades posted
- Project demo period April 28 – May 1
  - Please email me to sign up for a 30-minute slot
- Final exam on May 2 (Monday 2-5pm)

Review of the bigger picture

Query optimization
- Consider a space of possible plans (April 7)
  - Rewrite logical plan to combine “blocks” as much as possible
  - Each block will then be optimized separately
  - Fewer blocks → larger plan space
- Estimate costs of plans in the search space (today)
- Search through the space for the “best” plan (next lecture)

Cost estimation

Physical plan example:

```
PROJECT (sid)
MERGE-JOIN (CID)
SORT (CID) SCAN (Course)
MERGE-JOIN (SID)
SORT (SID) SCAN (Enroll)
FILTER (name = “Bart”)
```

- We have: cost estimation for each operator
  - Example: SORT(CID) takes \(2 \times B(input)\)
    - But what is \(B(input)\)?
- We need: size of intermediate results

Simple statistics

- Suppose DBMS collects the following statistics for each table \(R\)
  - Size of \(R\): \(|R|\)
  - For each column \(A\) in \(R\), the number of distinct \(A\) values: \(|\pi _A R|\)
  - Assumption: \(RA\) values are uniformly distributed over \(\pi _A R\) (i.e., all values have the same count in \(R\))
- Statistics are traditionally re-computed periodically; accurate statistics are not required for estimation

Selections with equality predicates

- \(Q: \sigma _A = v R\)
- Additional assumption: \(v\) does appear in \(R\)
- \(|Q| \approx \left[\frac{|R|}{|\pi _A R|}\right]\)
  - \(1/|\pi _A R|\) is the selectivity factor of predicate \((A = v)\)
  - This predicate reduces the size of input table by the selectivity factor
Conjunctive predicates

- $Q: \sigma_A = a$ and $B = v$
- Additional assumption: $(A = a)$ and $(B = v)$ are independent
  - Example: age and gender
  - Counterexample: major and advisor

- $|Q| \approx \left\lfloor |R| \cdot \left(\frac{\pi_A R \cdot |\pi_B R|}{\prod d}ight) \right\rfloor$
  - Reduce the input size by all selectivity factors

Range predicates

- $Q: \sigma_A > r$
- Not enough information!
  - Just pick, say, $|Q| \approx \left\lfloor |R| \cdot 1/3 \right\rfloor$
- With more information
  - Largest $R_A$ value: high($R_A$)
  - Smallest $R_A$ value: low($R_A$)
  - $|Q| \approx \left\lfloor \left(\frac{\text{high}(R_A) - r}{\text{low}(R_A) - \text{high}(R_A)}\right) \cdot |R| \right\rfloor$
  - Additional assumption: uniform spread
  - In practice: sometimes the second highest and lowest are used instead
    - The highest and the lowest are often used by inexperienced database designs to represent invalid values!

Negated and disjunctive predicates

- $Q: \sigma_A \neq a$
  - Selectivity factor of $a$ is $(1 - \text{selectivity factor of } a)$

- $Q: \sigma_A = a$ or $B = v$
  - Selectivity factor of $a$ is $(1 - \text{selectivity factor of } a)$

- $Q: \sigma_A = a$ or $B = v$
  - No! Rows satisfying $(A = a)$ and $(B = v)$ are counted twice
  - $|Q| \approx \left\lfloor \left(1 - \frac{\text{selectivity factor of } a}{\text{selectivity factor of } v}\right) \cdot |R| \right\rfloor$
    - Intuition: $(A = a)$ or $(B = v)$ is equivalent to $\neg (\neg (A = a) \land \neg (B = v))$

Two-way equi-join

- $Q: R(A, B) \bowtie S(B, C)$
- Additional assumption: containment of value sets
  - Every row in the “smaller” table (one with fewer distinct values for the join column) joins with some row in the
    other table
  - That is, if $|\pi_B R| \leq |\pi_B S|$ then $\pi_B R \leq \pi_B S$
  - Certainly not true in general

- $Q: R(A, B) \bowtie S(B, C)$
  - Selectivity factor of $R.B = S.B$ is $1/\max(|\pi_B R|, |\pi_B S|)$

Multi-table equi-join

- $Q: R(A, B) \bowtie S(B, C) \bowtie T(C, D)$
- What is the number of distinct C values in the join of $R$ and $S$?
- Additional assumption: preservation of value sets
  - A non-join attribute does not lose values from its set of possible values
  - That is, if $A$ is in $R$ but not $S$, then $\pi_A(R \bowtie S) = \pi_A R$
  - Certainly not true in general

- $Q: R(A, B) \bowtie S(B, C) \bowtie T(C, D)$
  - Start with the product of relation sizes
  - $|R| \cdot |S| \cdot |T|$
  - Reduce the total size by the selectivity factor of each join predicate
  - $R.B = S.B$: $1/\max(|\pi_B R|, |\pi_B S|)$
  - $S.C = T.C$: $1/\max(|\pi_C S|, |\pi_C T|)$
  - $|Q| \approx \left(\frac{|R| \cdot |S| \cdot |T|}{\max(|\pi_B R|, |\pi_B S|) \cdot \max(|\pi_C S|, |\pi_C T|)}\right)$
Recap: cost estimation with simple stats

- Using similar ideas, we can estimate the size of projection, duplicate elimination, union, difference, aggregation (with grouping)
- Lots of assumptions and very rough estimation
  - Maybe okay if we overestimate or underestimate consistently
  - May lead to very nasty optimizer “hints”
- Next: better estimation using more information (histograms)

Histograms

- Motivation
  - Too little information
  - Actual distribution of \( R.A \): \((v_1, f_1), (v_2, f_2), \ldots, (v_n, f_n)\)
    - \( f_i \) is frequency of \( v_i \), or the number of times \( v_i \) appears as \( R.A \)
    - Too much information
- Anything in between?
  - Partition the domain of \( R.A \) into buckets
  - Store a small summary of the distribution within each bucket
  - Number of buckets is the “knob” that controls the resolution

Equi-width histogram

- Divide the domain into \( B \) buckets of equal width
- Store the bucket boundaries and the sum of frequencies of the values within each bucket

Using an equi-width histogram

- \( Q: \sigma_A = 5 \) \( R \)
  - 5 is in bucket [5, 8] (with 19 rows)
  - Assume uniform distribution within the bucket
    - \(|Q| \approx 19/4 \approx 5\) \( (|Q| = 1\), actually)\)

- \( Q: \sigma_A \ge 7 \) and \( \le 16 \) \( R \)
  - [7, 16] covers [9, 12] (27) and [13, 16] (13)
  - [7, 16] partially covers [5, 8] (19)
  - \(|Q| \approx 19/2 + 27 + 13 \approx 50\) \( (|Q| = 52\), actually)\)

- \( Q: R(A, B) \bowtie S(B, C) \)
  - Consider only joining buckets in histograms for \( R.B \) and \( S.B \)
  - Rows in other buckets do not join
  - Within the joining buckets, use simple rules

Equi-height histogram

- Divide the domain into \( B \) buckets with roughly the same number of rows in each bucket
- Store this number and the bucket boundaries
- Intuition: high frequencies are more important than low frequencies
Construction and maintenance

- Construction
  - Sort all \( R.A \) values, and then take equally spaced splits
    - Example: 1 2 2 3 4 7 8 9 10 10 10 11 11 12 12 14 16 ...
  - Sampling also works

- Maintenance
  - Incremental maintenance
    - Merge adjacent buckets with small counts
    - Split any bucket with a large count
      - Select the median value to split
      - Need a sample of the values within this bucket to work well
  - Periodic recomputation also works

Histogram tricks

- Store the number of distinct values in each bucket
  - To remove the effects of the values with 0 frequency
    - These values tend to cause underestimation
  - Assume uniform spread (the difference between this value and the next value with non-zero frequency)
- Compressed histogram
  - Store \((v_i, f_i)\) pairs explicitly if \(f_i\) is high
  - For other values, use an equi-width or equi-height histogram
- Self-tuning
  - Analyze feedback from query execution engine to refine histograms
  - Aboulnaga and Chaudhuri, SIGMOD 1999

More histograms

- More in Poosala et al., SIGMOD 1996
- V-optimal\((V, F)\) histogram
  - Avoid putting very different frequencies into the same bucket
  - Partition in a way to minimize \(\sum VAR_i\), overall, where \(VAR_i\) is the frequency variance within bucket \(i\)
- MaxDiff\((V, A)\) histogram
  - Define area to be the product of the frequency of a value and its spread
  - Insert bucket boundaries where two adjacent areas differ by large amounts
  - A bit easier to construct than V-optimal; comparable performance

Wavelets

- Mathematical tool for hierarchical decomposition of functions and signals
- Haar wavelets: recursive pair-wise averaging and differencing at different resolutions
  - Simplest wavelet basis, easy to implement

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Averages</th>
<th>Detail coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>[2, 2, 0, 2, 3, 5, 4, 4]</td>
<td>[0.5, –1, –1, 0]</td>
</tr>
<tr>
<td>2</td>
<td>[2, 1, 4, 4]</td>
<td>[0.5, 0]</td>
</tr>
<tr>
<td>1</td>
<td>[1.5, 4]</td>
<td>[0.5, 0]</td>
</tr>
<tr>
<td>0</td>
<td>[2.75]</td>
<td>[–1.25]</td>
</tr>
</tbody>
</table>

Haar wavelet decomposition: \([2.75, –1.25, 0.5, 0, 0, –1, –1, 0]\)

Using an equi-height histogram

- \(Q: \sigma_A = 4 \quad R\)
  - 5 is in bucket \([1, 7]\) \(16\)
  - Assume uniform distribution within the bucket
  - \(|Q| \approx 16/7 \approx 2\) \(|(|Q| = 1, \text{actually})\)
  - \(Q: \sigma_A \geq 7 \text{ and } A \leq 16 \quad R\)
  - \([7, 16]\) covers \([8, 9], [10, 11], [12, 16]\) (all with 16)
  - \([7, 16]\) partially covers \([1, 7]\) \(16\)
  - \(|Q| \approx 16/7 + 16 + 16 + 16 \approx 50\)
    \(|(|Q| = 52, \text{actually})\)
  - Join similar to equi-width histogram

Haar wavelet coefficients

- Hierarchical decomposition structure

```
<table>
<thead>
<tr>
<th></th>
<th>0.5</th>
<th>–1.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>1</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>+</td>
<td>–</td>
</tr>
</tbody>
</table>
```

Original data

<table>
<thead>
<tr>
<th></th>
<th>–0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>–1</td>
<td>–1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>–0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>–1</td>
<td>–1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>–0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>–1</td>
<td>–1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Wavelet-based histogram

- Idea: use a compact subset of wavelet coefficients to approximate the data distribution
  - Matias et al., *SIGMOD* 1998
  - Transform the distribution function which maps \( v_i \) to \( f_i \)

- Steps
  - Compute cumulative data distribution function \( C(v) \)
    - \( C(v) \) is the number of tuples with \( R.A \leq v \)
  - Compute wavelet transform of \( C \)
  - Coefficient thresholding: keep only the coefficients that are largest in absolute normalized value
    - For Haar wavelets, divide coefficients at resolution \( j \) by \( 2^{j/2} \)

Using a wavelet-based histogram

- \( Q: \sigma_A > u \) and \( A \leq v \) \( R \)
- \( |Q| = C(v) - C(u) \)
- Search the tree to reconstruct \( C(v) \) and \( C(u) \)
  - Worst case: two paths, \( O(\log N) \), where \( N \) is the size of the domain
  - If we just store \( B \) coefficients, it becomes \( O(B) \), but answers are now approximate

- What about \( Q: \sigma_A = v \) \( R \)?
  - Same as \( \sigma_A > \text{predecessor}(v) \) and \( A \leq v \) \( R \)

Summary of histograms

- Wavelet-based histograms are shown to work better than traditional bucket-based histograms
- The trick of using cumulative distribution for range query estimation also works for bucket-based histograms
- Trade-off: better accuracy \( \leftrightarrow \) bigger size, and higher construction and maintenance costs