Query Optimization Part II

CPS 216
Advanced Database Systems

Announcements (April 19)

2

- 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

3

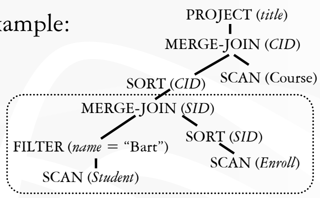
Query optimization

- 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:

Input to SORT(CID):



- ❖ 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: $R.A$ 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
- ❖ $|Q| \approx \lceil |R| / |\pi_A R| \rceil$
  - $1 / |\pi_A R|$ is the selectivity factor of predicate ($A = v$)
  - ☞This predicate reduces the size of input table by the selectivity factor

Conjunctive predicates

- Q: $\sigma_{A = u \text{ and } B = v} R$
- Additional assumption: $(A = u)$ and $(B = v)$ are independent
  - Example:
  - Counterexample:
- $|Q| \approx \lceil |R| / (|\pi_A R| \cdot |\pi_B R|) \rceil$
  - Reduce the input size by all selectivity factors

Negated and disjunctive predicates

- Q: $\sigma_{A \neq v} R$
  - $|Q| \approx \lceil |R| \cdot (1 - 1/|\pi_A R|) \rceil$
    - Selectivity factor of $\neg p$ is (1 – selectivity factor of p)
- Q: $\sigma_{A = u \text{ or } B = v} R$
  - $|Q| \approx \lceil |R| \cdot (1/|\pi_A R| + 1/|\pi_B R|) \rceil$?
  - $|Q| \approx \lceil |R| \cdot (1 - (1 - 1/|\pi_A R|) \cdot (1 - 1/|\pi_B R|)) \rceil$

Range predicates

- Q: $\sigma_{A > v} R$
- Not enough information!
  - Just pick, say, $|Q| \approx \lceil |R| \cdot 1/3 \rceil$
- With more information
  - Largest $R.A$ value: high($R.A$)
  - Smallest $R.A$ value: low($R.A$)
  - $|Q| \approx \lceil |R| \cdot (\text{high}(R.A) - v) / (\text{high}(R.A) - \text{low}(R.A)) \rceil$
    - 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 designer to represent invalid values!

Two-way equi-join

10

- 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 $\pi_B R \subseteq \pi_B S$
  - Certainly not true in general
- $|Q| \approx \lceil |R| \cdot |S| / \max(|\pi_B R|, |\pi_B S|) \rceil$
  - Selectivity factor of $R.B = S.B$ is $1/\max(|\pi_B R|, |\pi_B S|)$

Multi-table equi-join

11

- 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

Multi-table equi-join (cont'd)

12

- 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
  - $R.B = S.B$: $1/\max(|\pi_B R|, |\pi_B S|)$
  - $S.C = T.C$: $1/\max(|\pi_C S|, |\pi_C T|)$
  - $|Q| \approx \lceil (|R| \cdot |S| \cdot |T|) / (\max(|\pi_B R|, |\pi_B S|) \cdot \max(|\pi_C S|, |\pi_C T|)) \rceil$

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
  - Accurate estimate is not needed
  - Maybe okay if we overestimate or underestimate consistently
  - May lead to very nasty optimizer "hints"

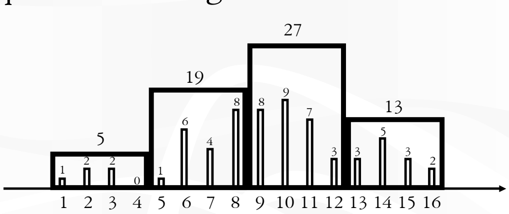
```
SELECT * FROM Student WHERE GPA > 3.9;
SELECT * FROM Student WHERE GPA > 3.9 AND GPA > 3.9;
```

- Next: better estimation using more information (histograms)

Histograms

- Motivation
  - $|R|$, $|\pi_A R|$, high($R.A$), low($R.A$)
    - Too little information
  - Actual distribution of $R.A$: (v_1, f_1), (v_2, f_2), …, (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

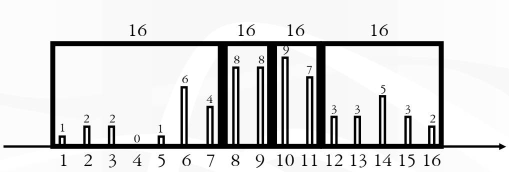
Construction and maintenance

- Construction
  - If high($R.A$) and low($R.A$) are known, use one pass over R to construct an accurate equi-width histogram
    - Keep a running count for each bucket
  - If scanning is unacceptable, use sampling
    - Construct a histogram on R_{sample}, and scale frequencies by $|R| / |R_{sample}|$
- Maintenance
  - Incremental maintenance: for each update on R, increment/decrement the corresponding bucket frequencies
  - Periodical recomputation: because distribution changes slowly

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: $\sigma_{A \geq 7 \text{ and } A \leq 16} R$
  - [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 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

Using an equi-height histogram

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

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 $\sum_i VAR_i$ 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

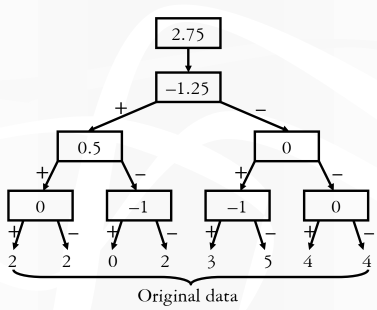
- ❖ Mathematical tool for hierarchical decomposition of functions and signals
- ❖ Haar wavelets: recursive pair-wise averaging and differencing at different resolutions
  - Simplest wavelet basis, easy to implement

Resolution	Averages	Detail coefficients
3	[2, 2, 0, 2, 3, 5, 4, 4]	
2	[2, 1, 4, 4]	[0, –1, –1, 0]
1	[1.5, 4]	[0.5, 0]
0	[2.75]	[–1.25]

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

Haar wavelet coefficients

- ❖ Hierarchical decomposition structure



Wavelet-based histogram

25

- 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

26

- Q: $\sigma_{A > u \text{ and } A \leq v} R$
- $|Q| = C(v) - C(u)$
- 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: $\sigma_{A = v} R$?

Summary of histograms

27

- 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 ↔ bigger size, and higher construction and maintenance costs