Selectivity Estimations

- previous slides assume that we "know" how many tuples qualify
- but this has to be estimated somehow
- similar for join ordering algorithms etc.
- cardinalities (and thus selectivities) are fundamental for query optimization
- we will now look at deriving some estimations

Examples

SQL examples for typical selectivity problems:

-
  ```
  select *
  from   rel r
  where r.a=10
  ```

-
  ```
  select *
  from   rel r
  where r.b>2
  ```

-
  ```
  select *
  from   rel1 r1,rel2 r2
  where r1.a=r2.b
  ```

The different problems require different approaches.

Heuristic Estimations

Some commonly used selectivity estimations:

predicate	selectivity	requirement
$A = c$	$1/\|D(A)\|$	if index on A
	$1/10$	otherwise
$A > c$	$(\max(A) - c)/(\max(A) - \min(A))$	if index on A, interpol.
	$1/3$	otherwise
$A_1 = A_2$	$1/\max(\|D(A_1)\|, \|(D(A_2)\|)$	if index on A_1 and A_2
	$1/\|D(A_1)\|$	if index on A_1 only
	$1/\|D(A_2)\|$	if index on A_2 only
	$1/10$	otherwise

Note: Without further statistics, $|D(A)|$ is typically only known (easily estimated) if A is a key or there is an index on A.

Using Histograms

- selectivity can be calculated easily by looking at the real data
- not feasible, therefore look at aggregated data
- histograms partition the data values into buckets

A histogram $H_A : B \rightarrow \mathbb{N}$ over a relation R partitions the domain of the aggregated attribute A into disjoint buckets B, such that

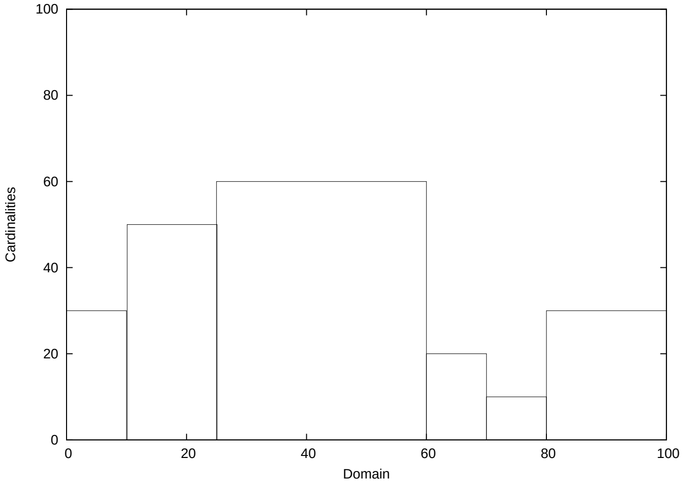
$$H_A(b) = |\{r | r \in R \wedge R.A \in b\}|$$

and thus $\sum_{b \in B} H_A(b) = |R|$.

Choosing B is very important, as we will see on the next slides.

Using Histograms (2)

A rough histogram might look like this:



Using Histograms (3)

Given a histogram, we can approximate the selectivities as follows:

$$A = c \qquad \frac{\sum_{b \in B : c \in b} H_A(b)}{\sum_{b \in B} H_A(b)}$$

$$A > c \qquad \frac{\sum_{b \in B : c \in b} \frac{\max(b) - c}{\max(b) - \min(b)} H_A(b) + \sum_{b \in B : \min(b) > c} H_A(b)}{\sum_{b \in B} H_A(b)}$$

$$A_1 = A_2 \qquad \frac{\sum_{b_1 \in B_1, b_2 \in B_2, b' = b_1 \cap b_2 : b' \neq \emptyset} \frac{\max(b') - \min(b')}{\max(b_1) - \min(b_1)} H_{A_1}(b_1) \frac{\max(b') - \min(b')}{\max(b_2) - \min(b_2)} H_{A_2}(b_2)}{\sum_{b_1 \in B_1} H_{A_1}(b_1) \sum_{b_2 \in B_2} H_{A_2}(b_2)}$$

Using Histograms - Remarks

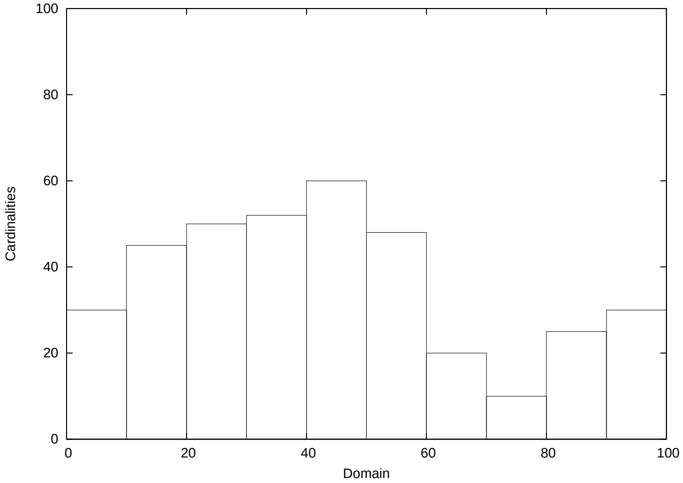
- estimations on previous slide can be improved
- in particular, the $A = c$ case is only a rough approximation
- requires more information
- if we interpret the histogram as a density function, $P(A = c) = 0$!
- a reasonable upper bound, though
- the $A > c$ case is more sound
- $A_1 = A_2$ assumes independence etc.

Building Histograms

- the buckets chosen greatly affect the overall quality
- histogram does not discern items within one bucket
- therefore: try to put items into different buckets
- how to choose the buckets?
- typical constraint: histogram size. *n* buckets (fixed)
- for a given set of data items, find a good histogram with *n* buckets
- additional constraint: data distribution is unknown (real data)

Building Histograms - Equiwidth

Partitions the domain into buckets with a fixed width



Building Histograms - Equiwidth (2)

Advantages:

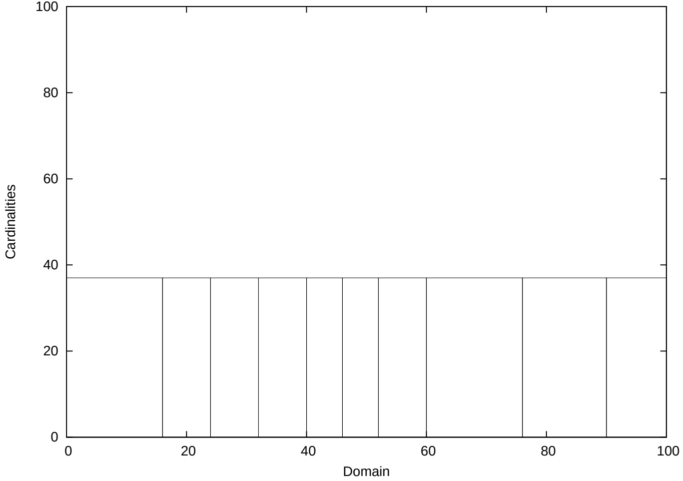
- easy to compute
- bucket boundaries can be computed (require no space)

Disadvantages:

- samples the domain uniformly
- does not handle skewed data well
- skew can lead to very uneven buckets
- greater estimation error in large buckets
- particular bad for zipf-like distributions

Building Histograms - Equidepth

Chooses the buckets to contain the same number of items



Building Histograms - Equidepth (2)

Advantages:

- adopts to data distribution
- reduces maximum error

Disadvantages:

- more involved (sort or similar)
- both boundaries and depth have to be stored (ties)

Very common histogram building technique

Building Histograms - Interpolation

- data is usually not completely random
- can we increase accuracy by interpolation?
- either within buckets (common) or instead of buckets (uncommon)
- histogram is a density function, not continuous, hard to interpolate
- use the equivalent distribution function instead
- very good for estimating $A > c$

Discussion

- estimations more complex in practice
- potentially different goals: maximum vs. average error
- histograms for derived values
- histogram convolution
- handling correlations
- multi-dimensional histograms
- cardinality estimators (sketches, MIPS etc.)