CS 133: Databases

Fall 2019

Lec 14 – 10/24

Prof. Beth Trushkowsky

Warm-up Exercise

(See exercise sheet. You can start before class.)

```
(a)  ORDER BY(day), two-way external merge sort
          |
     SELECT(bid=42), on the fly
          |
     SEQ SCAN(Reserves)
```

(b) 1000 + (0+ 10) +
(10+ 10) + (10+ 10) + (10+ 10) +
(10 + 0)
= 1080 I/Os

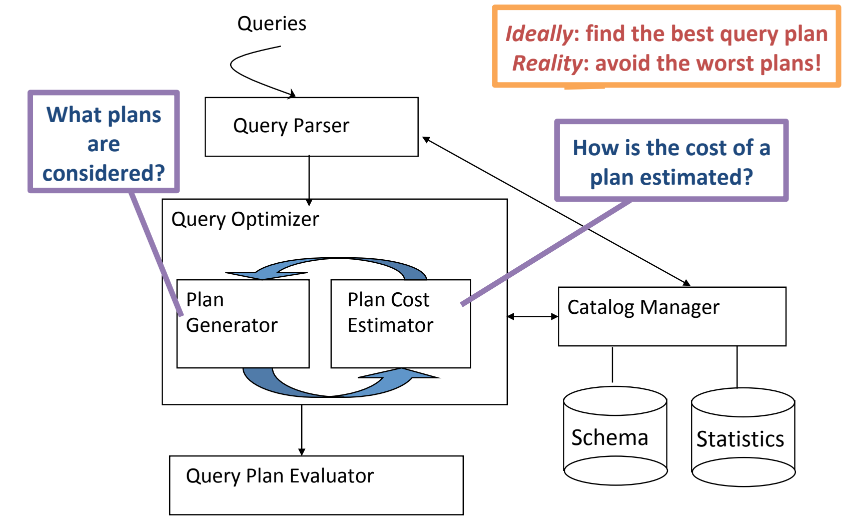
Adminstrivia

- No class next Tuesday!
  - No office hours Tuesday
- Monday office hours will be moved earlier in day
  - TBD, will post on Piazza
- This week's problem set is short

Goals for Today

- Continue to reason about estimating the result cardinality for selections and joins
  - System R heuristics
  - More advanced: histograms – Lab 3!

Cost-based Query Sub-System



Ideally: find the best query plan
Reality: avoid the worst plans!

What plans are considered?

How is the cost of a plan estimated?

Result Size Estimation for Joins

- For equi-join of R and S *range of result sizes (# tuples)*
  - If R and S have **no join attribute values** in common?
  - If join attributes are a **key for S**?
    - And if the join attributes ***also*** comprise a **foreign key in R**?

Result Size Estimation for Joins

- **General case**: relations have join attributes *a* in common, *a* is a key for neither
  - *Assume*: set of distinct R.*a* values is contained in set of S.*a*
  - Let Nkeys(*relation*) = number of distinct values in *relation*
  - ***Idea**:* each tuple of R has a $\frac{1}{\text{NKeys}(\mathbf{S})}$ chance of joining with each tuple in S

$$\text{NTuples}(\mathbf{R}) * \frac{\text{NTuples}(\mathbf{S})}{\text{NKeys}(\mathbf{S})}$$

  - Reversing above yields

$$\text{NTuples}(\mathbf{S}) * \frac{\text{NTuples}(\mathbf{R})}{\text{NKeys}(\mathbf{R})}$$

(use smaller of two if different)

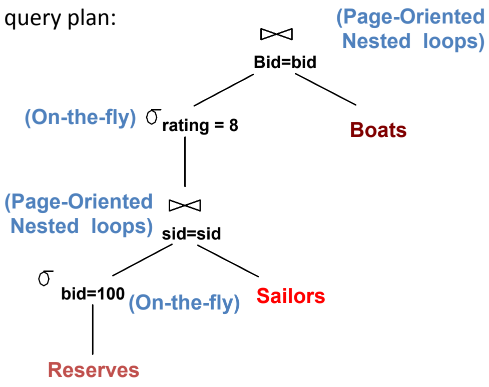
Exercise 2-3

2. Estimate the result cardinality for this SQL query:

```
SELECT *
FROM Sailors NATURAL JOIN Reserves
   NATURAL JOIN Boats;
```

Answer: number of tuples in Reserves (1000 pages, with 100 tuples/page)

3. Estimate the cost in I/Os of this query plan:


(Page-Oriented Nested loops)
⋈ Bid=bid
(On-the-fly) σ rating = 8
Boats
(Page-Oriented Nested loops)
⋈ sid=sid
σ bid=100
(On-the-fly)
Sailors
Reserves

Answer:
Join 1 I/O cost = 1000+10*500

Join 1 produces 10 pages of tuples, which is then filtered to 1 page

Join 2 I/O cost = 1*200

Total = 6000 + 200 = 6200 I/Os

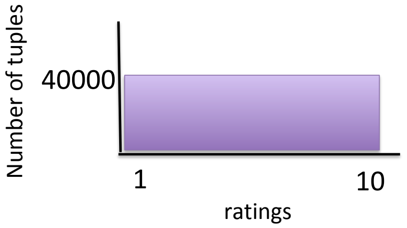
Histograms: Finer-Grained Statistics

- For better reduction factor estimates, many systems use **histograms**
- Histogram is ***approximation of a data distribution***

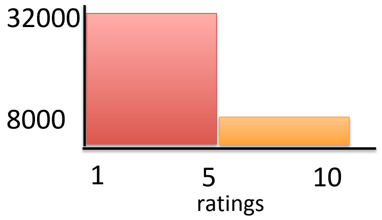
Histogram Example

- **Example**:
  Estimating how many Sailor tuples satisfy a predicate about *rating* (out of 40,000 total tuples)

Uniform distribution similar to histogram with one bucket

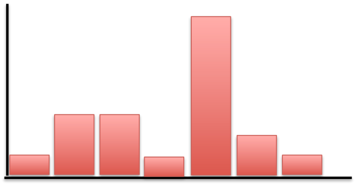


Still assume uniform *within* a bucket

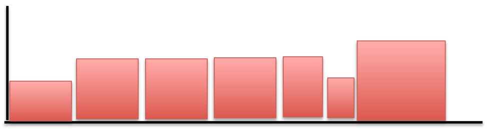


Equi-width vs. Equi-depth Histograms

- Equi-width
  - *# values represented* by each bucket is the same



- Equi-depth
  - *# of records* in each bucket is about the same



Exercise 4: Histograms

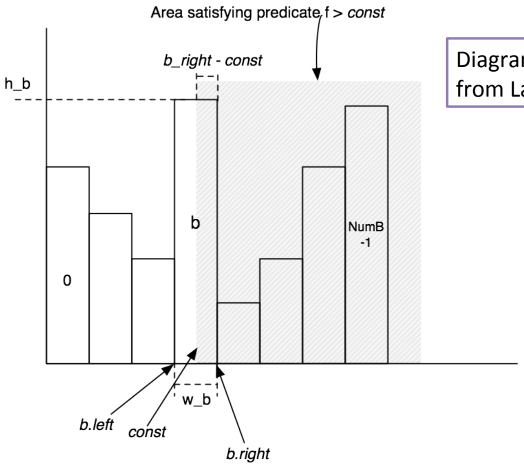


Diagram from Lab 3

Exercise 4

a) 41 values into 10 buckets. 4 in each, last one 5

b) Amount within bucket = 0.25*h_b tuples
→ Overall amount = (0.25*h_b tuples) / ntups

c) 0.25*h_b tuples + all tuples from buckets i > b
→ Divide sum by ntups

d) 0

Creating Equi-width histograms

- Suppose you want to be able to estimate the selectivity (reduction factor) for this query:

```
SELECT * FROM Sailors S
WHERE S.age = 40 AND S.rating > 5;
```

- Recall that we assume independence of terms and so the filter's RF is the **product of the terms' RFs**

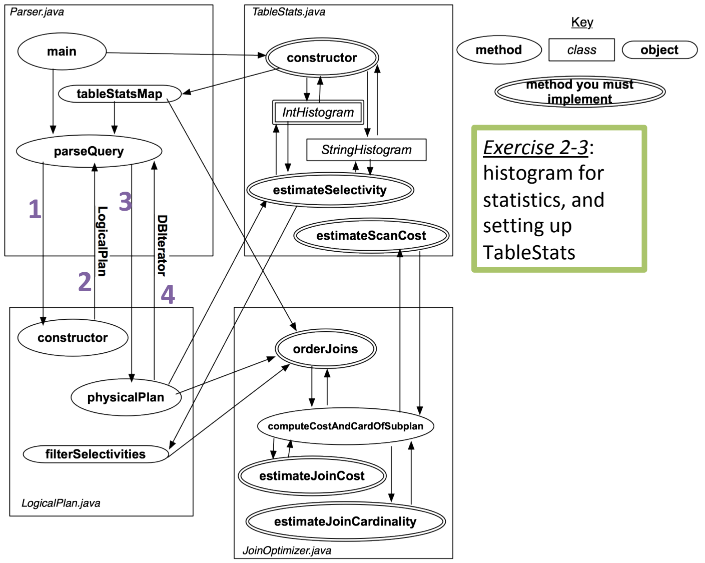
- Discuss with a neighbor:
  - How many histograms would we need?
  - Suppose you want to create new histogram(s) on an existing relation. Brainstorm what you would need to do. Think of the functionality from Exercise 4.

You can assume a fixed number of buckets

Lab 3: SimpleDb Optimizer

Statistics, like the Catalog, are memory-only in SimpleDb

Generated when Parser initialized



Exercise 2-3: histogram for statistics, and setting up TableStats

Exercise 1: understand formation of physical query plan in SimpleDb