CSE 344: Section 7 Analysis and Optimization

November 9th, 2017

Administrivia

- HW6 out (AWS) - Get started!
  - Due November 21st

- Pick up midterms from CSE front office

Query Optimization

Query -> Logical Plan (many) -> Physical Plan (many)

Cost-based estimation:

Enumerate different plans -> Compute expected costs -> Select a plan

Indexing

Hashtable vs B+ Tree

Clustered vs Unclustered

Primary vs Secondary

Sequential File vs Heap File

Indexing Syntax

```
CREATE INDEX idx ON table(a);

CREATE INDEX idx ON table(a, b);

CREATE INDEX idx ON table(a, b, c, d, ...);
```

(covering index)

```
CREATE UNIQUE INDEX idx ON table(a);

CREATE CLUSTERED INDEX idx ON table(a);
```

Cost Estimation: Factors

B(R) = # blocks for relation R

T(R) = # tuples for relation R

V(R, a) = # of unique values of attribute a in relation R

M = # of available memory pages

Selectivity

When A = c: $f = 1/V(R, a)$

When A < c: $f = (c - \min(R, a))/(\max(R, a) - \min(R, a))$

When c1 < A < c2: $f = (c2 - c1)/(\max(R, a) - \min(R, a))$

Cost Estimation: Selection (σ)

Table scan = B(R)

Point Selection:

- Index Based Selection (clustered) = B(R)/V(R, a)
- Index Based Selection (unclustered) = T(R)/V(R, a)

Cost Estimation: Hash Join (⋈)

R joined with S (assume R is smaller in size)

B(R) + B(S)

One pass (look at each table once) if B(R) <= M

Cost Estimation: Nested Loop Join (⋈)

Naive: B(R) + T(R)B(S)

```
for each tuple t1 in R do
    for each tuple t2 in S do
        if t1 and t2 join then output (t1,t2)
```

Why B(R) + T(R)B(S)? Where do the B's come from?
To get the data from R into memory, we scan R one block at a time.
Then, to do the matching for the output, we consider each tuple in R and scan S (one block at a time) to try and match on ONE tuple from R.
In main memory (free in terms of IO operations), we are deconstructing these blocks and doing tuple-to-tuple comparisons to join as with the following refinements.

Cost Estimation: Nested Loop Join (⋈)

Page-at-a-time: B(R) + B(R)B(S)

```
for each page of tuples r in R do
    for each page of tuples s in S do
        for all pairs of tuples t1 in r, t2 in s
            if t1 and t2 join then output (t1,t2)
```

Cost Estimation: Nested Loop Join (⋈)

Block-nested-loop: B(R) + B(R)B(S)/(M-1)

```
for each group of M-1 pages r in R do
    for each page of tuples s in S do
        for all pairs of tuples t1 in r, t2 in s
            if t1 and t2 join then output (t1,t2)
```

Cost Estimation: Sort-Merge Join (⋈)

B(R) + B(S)

One pass (look at each table once) if B(R) + B(S) <= M

Cost Est.: Index Nested Loop Join ($\bowtie_\theta$)

If S is clustered: B(R) + T(R)B(S)/V(S,a)

If S is unclustered: B(R) + T(R)T(S)/V(S,a)

Why T(R) instead of B(R) as the second term's multiplier in each equation?
We scan R as represented by the first term.
To match the tuples we can't assume whole blocks of R have the same attribute value to join on, thus when we read in S, we must read the parts of S we care about corresponding to every tuple.

AWS key points

- Lab machines or attu (Java 8)
- Run locally first
- Auto termination!
- Use a new output directory each time
    - Or delete the old output first