Lecture 24: Query Execution

Wednesday, November 28, 2001

Outline

- Query execution
  - Sorting based algorithms (6.5)
  - Two pass algorithms based on hashing (6.6)
  - Two pass algorithms based on indexes (6.7)
- Query optimization
  - From SQL to logical plans (7.3)
  - Algebraic laws (7.2)

Recall: Cost Parameters

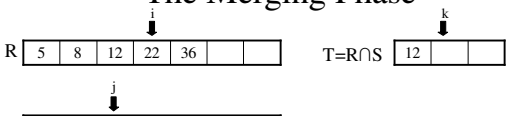
- B(R) = number of blocks for relation R
- T(R) = number of tuples in relation R
- V(R, a) = number of distinct values of attribute a
- M = main memory size, in blocks

Two-Pass Algorithms Based on Sorting

Binary operations: R ∩ S, R U S, R – S

- Idea: sort R, sort S, then do the right thing
- A closer look:
  - Step 1: split R into runs of size M, then split S into runs of size M. Cost: 2B(R) + 2B(S)
  - Step 2: merge M/2 runs from R; merge M/2 runs from S; ouput a tuple on a case by cases basis
- Total cost: 3B(R)+3B(S)
- Assumption: B(R)+B(S)<= M^2

The Merging Phase




```
R∩S:
repeat
  case
    R[i] < S[j] : i++;
    R[i] = S[j] : T[k++]=R[i++]; j++;
    R[i] > S[j] : j++;
until done;
```

```
R-S:
repeat
  case
    R[i] < S[j] :
    R[i] = S[j] :
    R[i] > S[j] :
until done;
```


Two-Pass Algorithms Based on Sorting

Join R⋈S

- Start by sorting both R and S on the join attribute:
  - Cost: 4B(R)+4B(S) (because need to write to disk)
- Read both relations in sorted order, match tuples
  - Cost: B(R)+B(S)
- Difficulty: many tuples in R may match many in S
  - If at least one set of tuples fits in M, we are OK
  - Otherwise need nested loop, higher cost
- Total cost: 5B(R)+5B(S)
- Assumption: B(R) <= M^2, B(S) <= M^2

Two-Pass Algorithms Based on Sorting

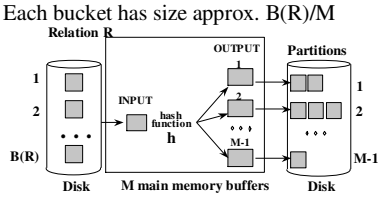
Join R⋈ S

- If the number of tuples in R matching those in S is small (or vice versa) we can compute the join during the merge phase
- Total cost: 3B(R)+3B(S)
- Assumption: B(R) + B(S) <= M^2

7

Two Pass Algorithms Based on Hashing

- Idea: partition a relation R into buckets, on disk
- Each bucket has size approx. B(R)/M



- Does each bucket fit in main memory ?
  - Yes if B(R)/M <= M, i.e. B(R) <= M^2

8

Hash Based Algorithms for δ

- Recall: δ(R) = duplicate elimination
- Step 1. Partition R into buckets
- Step 2. Apply δ to each bucket (may read in main memory)

- Cost: 3B(R)
- Assumption:B(R) <= M^2

9

Hash Based Algorithms for γ

- Recall: γ(R) = grouping and aggregation
- Step 1. Partition R into buckets
- Step 2. Apply γ to each bucket (may read in main memory)

- Cost: 3B(R)
- Assumption:B(R) <= M^2

10

Hash-based Join

- R ⋈ S
- Recall the *main memory hash-based join*:
  - Scan S, build buckets in main memory
  - Then scan R and join

11

Partitioned Hash Join

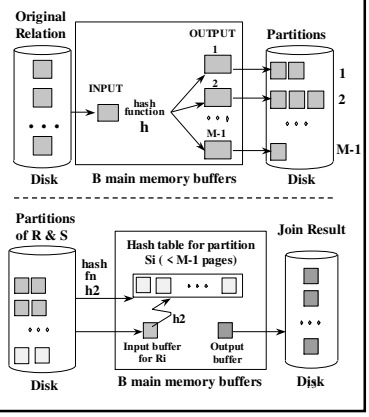
R⋈ S

- Step 1:
  - Hash S into M buckets
  - send all buckets to disk
- Step 2
  - Hash R into M buckets
  - Send all buckets to disk
- Step 3
  - Join every pair of buckets

12

Hash-Join

- Partition both relations using hash fn **h**: R tuples in partition i will only match S tuples in partition i.
- Read in a partition of R, hash it using **h2 (<> h!)**. Scan matching partition of S, search for matches.



Partitioned Hash Join

- Cost: 3B(R) + 3B(S)
- Assumption: min(B(R), B(S)) <= M^2

Hybrid Hash Join Algorithm

- Partition S into k buckets
- But keep first bucket S_1 in memory, k-1 buckets to disk
- Partition R into k buckets
  - First bucket R_1 is joined immediately with S_1
  - Other k-1 buckets go to disk
- Finally, join k-1 pairs of buckets:
  - (R_2,S_2), (R_3,S_3), …, (R_k,S_k)

Hybrid Join Algorithm

- How big should we choose k ?
- Average bucket size for S is B(S)/k
- Need to fit B(S)/k + (k-1) blocks in memory
  - B(S)/k + (k-1) <= M
  - k slightly larger than B(S)/M

Hybrid Join Algorithm

Example: B(R) = 1000, B(S) = 500, M = 101

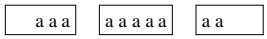
- Choose k slightly larger than 500/101: k = 6
- Buckets for S have size 500/6 = 83
- Keep 1 in memory, still more than k=6 blocks available to partition R

Hybrid Join Algorithm

- How many I/Os ?
- Recall: cost of partitioned hash join:
  - 3B(R) + 3B(S)
- Now we save 2 disk operations for one bucket
- Recall there are k buckets
- Hence we save 2/k(B(R) + B(S))
- Cost: (3-2/k)(B(R) + B(S)) = (3-2M/B(S))(B(R) + B(S))

Indexed Based Algorithms

- Recall that in a clustered index all tuples with the same value of the key are clustered on as few blocks as possible



- Note: book uses another term: "clustering index". Difference is minor…

19

Index Based Selection

- Selection on equality: $\sigma_{a=v}(R)$
- Clustered index on a: cost B(R)/V(R,a)
- Unclustered index on a: cost T(R)/V(R,a)

20

Index Based Selection

- Example: B(R) = 2000, T(R) = 100,000, V(R, a) = 20, compute the cost of $\sigma_{a=v}(R)$
- Cost of table scan:
  - If R is clustered: B(R) = 2000 I/Os
  - If R is unclustered: T(R) = 100,000 I/Os
- Cost of index based selection:
  - If index is clustered: B(R)/V(R,a) = 100
  - If index is unclustered: T(R)/V(R,a) = 5000
- Notice: when V(R,a) is small, then unclustered index is useless

21

Index Based Join

- $R \bowtie S$
- Assume S has an index on the join attribute
- Iterate over R, for each tuple fetch corresponding tuple(s) from S
- Assume R is clustered. Cost:
  - If index is clustered: B(R) + T(R)B(S)/V(S,a)
  - If index is unclustered: B(R) + T(R)T(S)/V(S,a)

22

Index Based Join

- Assume both R and S have a sorted index (B+ tree) on the join attribute
- Then perform a merge join (called zig-zag join)
- Cost: B(R) + B(S)

23

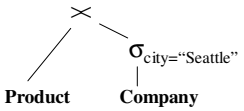
Example

Product(<u>pname</u>, maker), **Company**(<u>cname</u>, city)

Clustered index: **Product**.pname, **Company**.cname
Unclustered index: **Company**.city

```
Select Product.pname
From Product, Company
Where Product.maker=Company.cname
  and Company.city = "Seattle"
```

24



- Case 1: V(**Company**, city) ≈ T(**Company**)
  V(**Product**, maker) ≈ T(**Product**)
- Case 2: V(**Company**, city) << T(**Company**)
  V(**Product**, maker) ≈ T(**Product**)
- Case 3: V(**Company**, city) << T(**Company**)
  V(**Product**, maker) << T(**Product**)

25

Optimization

- Start: convert SQL to Logical Plan
- Algebraic laws: are the foundation for every optimization
- Two approaches to optimizations:
  - Heuristics: apply laws that *seem* to result in cheaper plans
  - Cost based: estimate size and cost of intermediate results, search systematically for best plan

26

Converting from SQL to Logical Plans

```
Select a1, …, an
From R1, …, Rk
Where C
```

$\Pi_{a1,\ldots,an}(\sigma_C(R1 \times R2 \times \ldots \times Rk))$

```
Select a1, …, an
From R1, …, Rk
Where C
Group by b1, …, bl
```

$\Pi_{a1,\ldots,an}(\gamma_{b1,\ldots,bm,aggs}(\sigma_C(R1 \times R2 \times \ldots \times Rk)))$

27

Converting Nested Queries

```
Select distinct product.name
From product
Where product.maker in (Select company.name
                        From company
                        where company.city="Seattle")
```

```
Select distinct product.name
From product, company
Where product.maker = company.name AND
      company.city="Seattle"
```

28

Converting Nested Queries

```
Select distinct x.name, x.maker
From product x
Where x.color= "blue"
  AND x.price >= ALL (Select y.price
                      From  product y
                      Where x.maker = y.maker
                        AND y.color="blue")
```

How do we convert this one to logical plan ?

29

Converting Nested Queries

Let's compute the complement first:

```
Select distinct x.name, x.maker
From product x
Where x.color= "blue"
  AND x.price < SOME (Select y.price
                      From  product y
                      Where x.maker = y.maker
                        AND y.color="blue")
```

30

Converting Nested Queries

This one becomes a SFW query:

```
Select distinct x.name, x.maker
From product x, product y
Where x.color= "blue" AND x.maker = y.maker
  AND y.color="blue"  AND x.price < y.price
```

This returns exactly the products we DON'T want, so…

31

Converting Nested Queries

```
(Select x.name, x.maker
 From product x
 Where x.color = "blue")

EXCEPT

(Select x.name, x.maker
 From product x, product y
 Where x.color= "blue" AND x.maker = y.maker
   AND y.color="blue"  AND x.price < y.price)
```

32

Algebraic Laws

- Commutative and Associative Laws
  - $R \cup S = S \cup R$, $R \cup (S \cup T) = (R \cup S) \cup T$
  - $R \cap S = S \cap R$, $R \cap (S \cap T) = (R \cap S) \cap T$
  - $R \bowtie S = S \bowtie R$, $R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$
- Distributive Laws
  - $R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$

33

Algebraic Laws

- Laws involving selection:
  - $\sigma_{C \text{ AND } C'}(R) = \sigma_C(\sigma_{C'}(R)) = \sigma_C(R) \cap \sigma_{C'}(R)$
  - $\sigma_{C \text{ OR } C'}(R) = \sigma_C(R) \cup \sigma_{C'}(R)$
  - $\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$
    - When C involves only attributes of R
  - $\sigma_C(R - S) = \sigma_C(R) - S$
  - $\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$
  - $\sigma_C(R \cap S) = \sigma_C(R) \cap S$

34

Algebraic Laws

- Example: R(A, B, C, D), S(E, F, G)
  - $\sigma_{F=3}(R \underset{D=E}{\bowtie} S) =$?
  - $\sigma_{A=5 \text{ AND } G=9}(R \underset{D=E}{\bowtie} S) =$?

35

Algebraic Laws

- Laws involving projections
  - $\Pi_M(R \bowtie S) = \Pi_N(\Pi_P(R) \bowtie \Pi_Q(S))$
    - Where N, P, Q are appropriate subsets of attributes of M
  - $\Pi_M(\Pi_N(R)) = \Pi_{M,N}(R)$
- Example R(A,B,C,D), S(E, F, G)
  - $\Pi_{A,B,G}(R \underset{D=E}{\bowtie} S) = \Pi_{?}(\Pi_{?}(R) \underset{D=E}{\bowtie} \Pi_{?}(S))$

36