Lecture 23:

Monday, November 25, 2002

1

Outline

- Query execution: 15.1 – 15.5
- Query optimization: algebraic laws 16.2

2

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

[a a a] [a a a a a] [a a]

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

3

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)

4

Index Based Selection

- Example:
  B(R) = 2000
  T(R) = 100,000
  V(R, a) = 20

  cost of $\sigma_{a=v}(R)$ = ?
- Table scan:
  – If R is clustered: B(R) = 2,000 I/Os
  – If R is unclustered: T(R) = 100,000 I/Os
- Index based selection:
  – If index is clustered: B(R)/V(R,a) = 100
  – If index is unclustered: T(R)/V(R,a) = 5,000
- Notice: when V(R,a) is small, then unclustered index is useless

5

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)

6

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)

7

Example

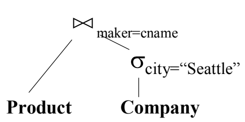
Product(<u>pname</u>, maker), **Company**(<u>cname</u>, city)

Clustered index: **Product**.<u>pname</u>, **Company**.<u>cname</u>
Unclustered index: **Product**.maker, **Company**.city

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

8

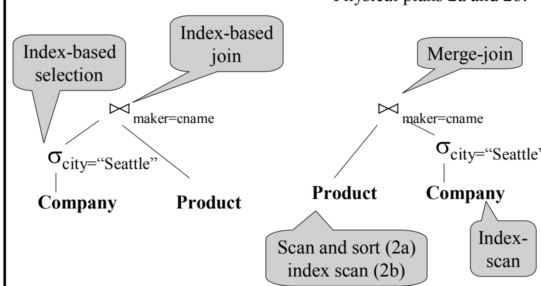
Logical Plan:



9

Physical plan 1:

Physical plans 2a and 2b:



10

- Plan 1:
  – Index-based selection: T(**Company**) / V(**Company**, city)
  – Index-based join: × T(**Product**) / V(**Product**, maker)
- Plan 2:
  – Table scan and selection on Company: B(**Company**)
  – Plan 2a: scan and sort: 3B(**Product**)
  – Plan 2b: index-scan: T(**Product**)
  – Merge-join: their sum

Plan 1: T(**Company**)/V(**Company**,city) × T(**Product**)/V(**Product**,maker)
Plan 2a: B(**Company**) + 3B(**Product**)
Plan 2b: B(**Company**) + T(**Product**)

11

Example

T(**Company**) = 5,000	B(**Company**) = 500	M = 100
T(**Product**) = 100,000	B(**Product**) = 1,000	

- Case 1: V(**Company**, city) ≈ T(**Company**)
  V(**Product**, maker) ≈ T(**Product**)
  V(**Company**,city) = 2,000
  V(**Product**,maker) = 20,000
- Case 2: V(**Company**, city) << T(**Company**)
  V(**Product**, maker) ≈ T(**Product**)
  V(**Company**,city) = 20
  V(**Product**,maker) = 20,000
- Case 3: V(**Company**, city) << T(**Company**)
  V(**Product**, maker) << T(**Product**)
  V(**Company**,city) = 20
  V(**Product**,maker) = 100

12

Which Plan is Best ?

Plan 1: T(**Company**)/V(**Company**,city) × T(**Product**)/V(**Product**,maker)
Plan 2a: B(**Company**) + 3B(**Product**)
Plan 2b: B(**Company**) + T(**Product**)

Case 1:

Case 2:

Case 3:

13

Optimization

- Chapter 16
- At the hart of the database engine
- Step 1: convert the SQL query to some logical plan
- Step 2: find a better logical plan, find an associated physical plan

14

Converting from SQL to Logical Plans

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

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

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

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

15

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"
```

16

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 ?

17

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")
```

18

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…

19

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)
```

20

Optimization: the Logical Query Plan

- Now we have one logical plan
- Algebraic laws:
  - 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
- All modern database optimizers use a cost-based optimizer
  - Why ?

21

The three components of an optimzer

We need three things in an optimizer:

- Algebraic laws
- An optimization algorithm
- A cost estimator

22

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
  - $R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$

23

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
  - $\sigma_C(R - S) = \sigma_C(R) - S$
  - $\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$
  - $\sigma_C(R \cap S) = \sigma_C(R) \cap S$

24

Algebraic Laws

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

25

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)
  - $\Pi_{A,B,G}(R \bowtie S) = \Pi_{?}(\Pi_?(R) \bowtie \Pi_?(S))$

26

Algebraic Laws

Laws involving grouping and aggregation:

- $\delta(\gamma_{A,\ agg(B)}(R)) = \gamma_{A,\ agg(B)}(R)$
- $\gamma_{A,\ agg(B)}(\delta(R)) = \gamma_{A,\ agg(B)}(R)$ if agg is "duplicate insensitive"
  - Which of the following are "duplicate insensitive" ? sum, count, avg, min, max
- $\gamma_{A,\ agg(D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$
  $\gamma_{A,\ agg(D)}(R(A,B) \bowtie_{B=C} (\gamma_{B,\ agg(D)}S(C,D)))$
  - Why is this true ?
  - Why would we do it ?

27