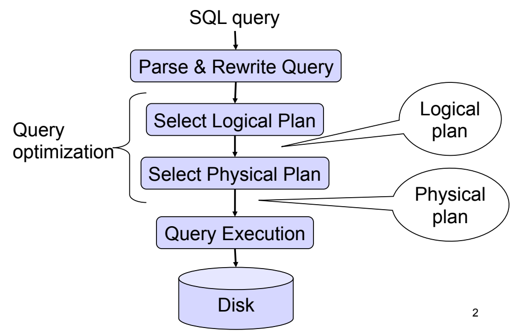
CSE 444: Database Internals

Lecture 10
Query Optimization (part 1)

Review: Query Evaluation Steps



What We Already Know…

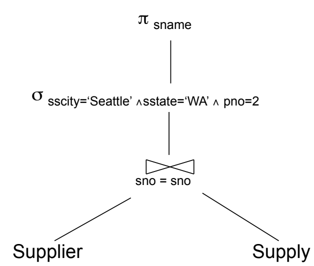
```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)
```

For each SQL query….

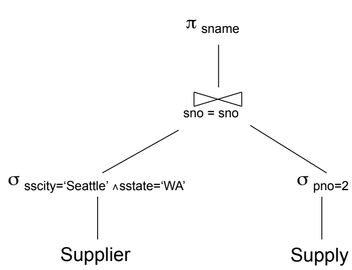
```
SELECT S.sname
FROM Supplier S, Supply U
WHERE S.scity='Seattle' AND S.sstate='WA'
AND S.sno = U.sno
AND U.pno = 2
```

There exist many logical query plan…

Example Query: Logical Plan 1



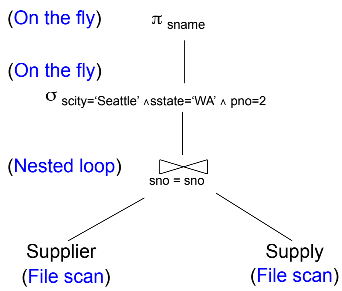
Example Query: Logical Plan 2



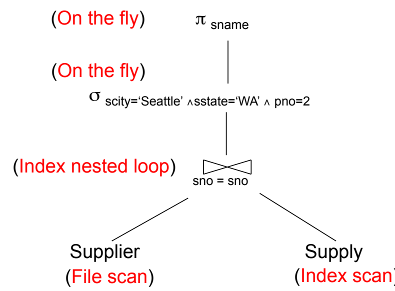
What We Also Know

- For each logical plan…
- There exist many physical plans

Example Query: Physical Plan 1



Example Query: Physical Plan 2



Query Optimizer Overview

- **Input**: A logical query plan
- **Output**: A good physical query plan
- **Basic query optimization algorithm**
  - Enumerate alternative plans (logical and physical)
  - Compute estimated cost of each plan
    - Compute number of I/Os
    - Optionally take into account other resources
  - Choose plan with lowest cost
  - This is called cost-based optimization

Today

Estimating the cost of a query plan

Estimating Cost of a Query Plan

- We already know how to
  - Compute the cost of different operations
    - In terms of number IOs
    - Normally should also consider CPU and network
- We still need to
  - Compute cost of retrieving tuples from disk
    - We can use different access paths
    - Queries have more sophisticated predicates than equality
  - Compute cost of a complete plan

Access Path

- **Access path**: a way to retrieve tuples from a table
  - A file scan
  - An index *plus* a matching selection condition
- Index *matches* selection condition if it can be used to retrieve just tuples that satisfy the condition
  - Example: Supplier(sid,sname,scity,sstate)
  - B+-tree index on (scity,sstate)
    - matches scity='Seattle'
    - does not match sid=3, does not match sstate='WA'

Access Path Selection

- Supplier(sid,sname,scity,sstate)
- Selection condition: sid > 300 ∧ scity='Seattle'
- Indexes: B+-tree on sid and B+-tree on scity
- Which access path should we use?
- We should pick the **most selective** access path

Magda Balazinska - CSE 444, Spring 2012 13

Access Path Selectivity

- **Access path selectivity is the number of pages retrieved if we use this access path**
  - Most selective retrieves fewest pages
- As we saw earlier, **for equality predicates**
  - Selection on equality: $\sigma_{a=v}(R)$
  - V(R, a) = # of distinct values of attribute a
  - 1/V(R,a) is thus the reduction factor
  - Clustered index on a: cost **B(R)/V(R,a)**
  - Unclustered index on a: cost **T(R)/V(R,a)**
  - (we are ignoring I/O cost of index pages for simplicity)

Magda Balazinska - CSE 444, Spring 2012 14

Selectivity for Range Predicates

Selection on range: $\sigma_{a>v}(R)$

- How to compute the selectivity?
- Assume values are uniformly distributed
- Reduction factor X
- X = (Max(R,a) - v) / (Max(R,a) - Min(R,a))
- Clustered index on a: cost **B(R)*X**
- Unclustered index on a: cost **T(R)*X**

Magda Balazinska - CSE 444, Spring 2012 15

Back to Our Example

- Selection condition: **sid > 300 ∧ scity='Seattle'**
  - Index I1: B+-tree on sid clustered
  - Index I2: B+-tree on scity unclustered
- Let's assume
  - V(Supplier,scity) = 20
  - Max(Supplier, sid) = 1000, Min(Supplier,sid)=1
  - B(Supplier) = 100, T(Supplier) = 1000
- **Cost I1: B(R) * (Max-v)/(Max-Min) = 100*700/999 ~ 70**
- **Cost I2: T(R) * 1/V(Supplier,scity) = 1000/20 = 50**

Magda Balazinska - CSE 444, Spring 2012 16

Selectivity with Multiple Conditions

What if we have an index on multiple attributes?

- Ex: selection $\sigma_{a=v1 \wedge b=v2}(R)$ and index on <a,b>

How to compute the selectivity?

- Assume attributes are independent
- X = 1 / (V(R,a) * V(R,b))
- Clustered index on <a,b>: cost B(R)*X
- Unclustered index on <a,b>: cost T(R)*X

Magda Balazinska - CSE 444, Spring 2012 17

Back to Estimating Cost of a Query Plan

- We already know how to
  - Compute the cost of different operations (last week)
  - Compute cost of retrieving tuples from disk with different access paths
- We still need to
  - Compute cost of a complete plan

Magda Balazinska - CSE 444, Spring 2012 18

Computing the Cost of a Plan

- Estimate cardinality in a bottom-up fashion
  - Cardinality is the size of a relation (nb of tuples)
  - Compute size of *all* intermediate relations in plan
- Estimate cost by using the estimated cardinalities

- We learned how to compute the cost last week
- But how do we compute cardinalities?

Projection: Cardinality of Result

- Output cardinality same as input cardinality
  - Same number of input tuples
  - But tuples are smaller!

Selection: Cardinality of Result

- Multiply input cardinality by reduction factor
  - Similar to estimating access path selectivity
  - In fact, we also say *operator/predicate selectivity*
  - Condition is a = c /* value selection on R */
    - Selectivity = 1/V(R,a)
  - Condition is a < v /* range selection on R */
    - Selectivity = (v - Min(R, a))/(Max(R,a) - Min(R,a))
  - Multiple conditions: assume independence
    - Use product of the reduction factors for the terms
    - Condition is a=v1 ∧ b= v2
    - Selectivity = 1 / (V(R,a) * V(R,b))

Join: Cardinality of Result

- For joins R ⋈ S
  - Take product of cardinalities of relations R and S
  - Apply reduction factors for each term in join condition

  - Terms are of the form: column1 = column2
  - Reduction: 1/ (MAX(V(R,column1), V(S,column2))
    - **Assumes each value in smaller set has a matching value in the larger set (more on next slide)**

Assumptions

- *Containment of values*: if V(R,A) <= V(S,B), then the set of A values of R is included in the set of B values of S
  - Note: this indeed holds when A is a foreign key in R, and B is a key in S

- *Preservation of values*: for any other attribute C, V(R $\bowtie_{A=B}$ S, C) = V(R, C) (or V(S, C))

Selectivity of R $\bowtie_{A=B}$ S

Assume V(R,A) <= V(S,B)

- Each tuple t in R joins with T(S)/V(S,B) tuple(s) in S

- Hence T(R $\bowtie_{A=B}$ S) = T(R) T(S) / V(S,B)

In general: T(R $\bowtie_{A=B}$ S) = T(R) T(S) / max(V(R,A),V(S,B))

Complete Example

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
  and y.pno = 2
  and x.scity = 'Seattle'
  and x.sstate = 'WA'
```

- Some statistics
  - T(Supplier) = 1000 records
  - T(Supply) = 10,000 records
  - B(Supplier) = 100 pages
  - B(Supply) = 100 pages
  - V(Supplier,scity) = 20, V(Suppliers,state) = 10
  - V(Supply,pno) = 2,500
  - Both relations are clustered
- M = 11

T(Supplier) = 1000 T(Supply) = 10,000 B(Supplier) = 100 B(Supply) = 100 V(Supplier,scity) = 20 V(Supplier,state) = 10 V(Supply,pno) = 2,500 M = 11

Physical Query Plan 1

(On the fly) π_{sname}

Selection and project on-the-fly
-> No additional cost.

(On the fly) $\sigma_{scity='Seattle' \wedge sstate='WA' \wedge pno=2}$

(Nested loop) $\bowtie_{sno = sno}$

Total cost of plan is thus cost of join:
= B(Supplier)+B(Supplier)*B(Supply)/(M-1)
= 100 + 100 * 100/10
= 1,100 I/Os

Supplier (File scan) Supply (File scan)

T(Supplier) = 1000 T(Supply) = 10,000 B(Supplier) = 100 B(Supply) = 100 V(Supplier,scity) = 20 V(Supplier,state) = 10 V(Supply,pno) = 2,500 M = 11

Physical Query Plan 2

(On the fly) π_{sname} (d)

(Sort-merge join) $\bowtie_{sno = sno}$ (c)

(Scan write to T1) (a) $\sigma_{scity='Seattle' \wedge sstate='WA'}$ — Supplier (File scan)

(Scan write to T2) (b) $\sigma_{pno=2}$ — Supply (File scan)

Total cost
= 100 + 100 * 1/20 * 1/10 (a)
+ 100 + 100 * 1/2500 (b)
+ 2 (c)
+ 0 (d)

Total cost ≈ **204 I/Os**

M = 11

Plan 2 with Different Numbers

What if we had:
10K pages of Supplier
10K pages of Supply

(On the fly) π_{sname} (d)

(Sort-merge join) $\bowtie_{sno = sno}$ (c)

(Scan write to T1) (a) $\sigma_{scity='Seattle' \wedge sstate='WA'}$ — Supplier (File scan)

(Scan write to T2) (b) $\sigma_{pno=2}$ — Supply (File scan)

Total cost
= 10000 + 50 (a)
+ 10000 + 4 (b)
+ 3*50 + 4 (c)
+ 0 (d)

Total cost ≈ 20,208 I/Os

Need to do a two-pass sort algorithm

T(Supplier) = 1000 T(Supply) = 10,000 B(Supplier) = 100 B(Supply) = 100 V(Supplier,scity) = 20 V(Supplier,state) = 10 V(Supply,pno) = 2,500 M = 11

Physical Query Plan 3

(On the fly) (d) π_{sname}

(On the fly) (c) $\sigma_{scity='Seattle' \wedge sstate='WA'}$

(b) $\bowtie_{sno = sno}$ (Index nested loop)

(Use hash index) 4 tuples (a) $\sigma_{pno=2}$

Supply (Hash index on pno) Assume: clustered

Supplier (Hash index on sno) Clustering does not matter

Total cost
= 1 (a)
+ 4 (b)
+ 0 (c)
+ 0 (d)

Total cost ≈ **5 I/Os**

Simplifications

- In the previous examples, we assumed that all index pages were in memory
- When this is not the case, we need to add the cost of fetching index pages from disk

Different Cost Models

- In previous examples, we considered IO costs
- Typically, want IO+CPU
- For parallel/distributed queries, add network bw
- If need to compare *logical* plans
  - Compute the cardinality of each *intermediate* relation
  - Sum up all the cardinalities

Statistics on Base Data

- All previous computations relied on information about the base relations
  - Number of blocks: B(R)
  - Number of tuples: T(R)
  - Min/max values
  - Number of distinct values: V(R,a)

Statistics on Base Data

- DBMSs collect a lot of info for base relations
  - Number of tuples (cardinality)
  - Indexes, number of keys in the index
  - Number of physical pages, clustering info
  - Statistical information on attributes
    - Min value, max value, number distinct values
    - Histograms
  - Correlations between columns (hard)
- Collection approach: periodic, using sampling

Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 8, max(age) = 68

$\sigma_{age=48}$(Empolyee) = ? $\sigma_{age>28 \text{ and } age<35}$(Empolyee) = ?

Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 8, max(age) = 68

$\sigma_{age=48}$(Empolyee) = ? $\sigma_{age>28 \text{ and } age<35}$(Empolyee) = ?

Estimate = 25000 / 50 = 500 Estimate = 25000 * 6 / 60 = 2500

Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 8, max(age) = 68

$\sigma_{age=48}$(Empolyee) = ? $\sigma_{age>28 \text{ and } age<35}$(Empolyee) = ?

Age:	0..20	20..29	30-39	40-49	50-59	> 60
Tuples	200	800	5000	12000	6500	500

Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 8, max(age) = 68

$\sigma_{age=48}$(Empolyee) = ? $\sigma_{age>28 \text{ and } age<35}$(Empolyee) = ?

Age:	0..20	20..29	30-39	40-49	50-59	> 60
Tuples	200	800	5000	12000	6500	500

Estimate = 1200 Estimate = 1*80 + 5*500 = 2580

Types of Histograms

- How should we determine the bucket boundaries in a histogram ?

Types of Histograms

- How should we determine the bucket boundaries in a histogram ?
- Eq-Width
- Eq-Depth
- Compressed

Employee(ssn, name, age)

Histograms

Eq-width:

Age:	0..20	20..29	30-39	40-49	50-59	> 60
Tuples	200	800	5000	12000	6500	500

Eq-depth:

Age:	0..35	36..40	41-44	45-52	53-56	> 57
Tuples	1800	2000	2100	2200	1900	1800

Compressed: store separately highly frequent values: (48,1900)

Difficult Questions on Histograms

- Small number of buckets
  - Hundreds, or thousands, but not more
  - WHY ?
- *Not* updated during database update, but recomputed periodically
  - WHY ?
- Multidimensional histograms rarely used
  - WHY ?

Summary

- What we know
  - Different types of physical query plans
  - How to compute the cost of a query plan
  - Although it is hard to compute the cost accurately
- We can now compare query plans!
- Next: Search space and enumeration algorithm