CSE544: Principles of Database Systems

Query Optimization and Database Statistics

Announcements

- Homework 2 is posted, due May 6
  - SimpleDB
  - Understand existing code PLUS write more code
  - Start early!!
- Project M2 (Proposal) due April 26
  - Define clear, limited goals! Don't try too much
- Review 4 (Anatomy): due April 29

Outline

- Chapter 15 in the textbook
- Paper on selectivity of conjuncts

Query Optimization

- Why?
  - Because of data independence
- What?
  - Search among many equivalent logical/physical plans, choose the cheapest
- Who?
  - System R, 1979: super-influential, laid out most of the key concepts; see book, 15.6
  - Today's optimizers are much more advanced

Query Optimization

<u>**Three major components**</u>:

1. Search space

2. Plan enumeration algorithms

3. Cardinality and cost estimation

1. Search Space

- This is the set of all alternative plans that are considered by the optimizer

- Defined by:
  - The set of *algebraic laws*
  - The *set of plans* used by the optimizer

Relational Algebra Laws: Joins

Commutativity :	$R \bowtie S = S \bowtie R$
Associativity:	$R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$
Distributivity:	$R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$

Outer joins get more complicated

Relational Algebra Laws: Selections

R(A, B, C, D), S(E, F, G)

$$\sigma_{F=3} (R \bowtie_{D=E} S) = \quad ?$$

$$\sigma_{A=5 \text{ AND } G=9} (R \bowtie_{D=E} S) = \quad ?$$

Relational Algebra Laws: Selections

R(A, B, C, D), S(E, F, G)

$$\sigma_{F=3}(R \bowtie_{D=E} S) = R \bowtie_{D=E} (\sigma_{F=3}(S))$$

$$\sigma_{A=5 \text{ AND } G=9}(R \bowtie_{D=E} S) = \sigma_{A=5}(R) \bowtie_{D=E} \sigma_{G=9}(S)$$

Group-by and Join

R(A, B), S(C,D)

$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$?

Group-by and Join

R(A, B), S(C,D)

$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$$
$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} (\gamma_{C,\ sum(D)} S(C,D)))$$

These are very powerful laws.
They were introduced only in the 90's.

Laws Involving Constraints

Foreign key

Product(<u>pid</u>, pname, price, cid)
Company(<u>cid</u>, cname, city, state)

$\Pi_{pid,\ price}(\text{Product} \bowtie_{cid=cid} \text{Company}) = \quad ?$

Laws Involving Constraints

Foreign key

Product(<u>pid</u>, pname, price, cid)
Company(<u>cid</u>, cname, city, state)

$$\Pi_{pid,\ price}(\text{Product} \bowtie_{cid=cid} \text{Company}) = \Pi_{pid,\ price}(\text{Product})$$

Need a second constraint for this law to hold. Which ?

Why such queries occur

Foreign key

Product(pid, pname, price, cid)
Company(cid, cname, city, state)

```
CREATE VIEW CheapProductCompany
    SELECT *
    FROM Product x, Company y
    WHERE x.cid = y.cid and x.price < 100
```

```
SELECT pname, price
FROM CheapProductCompany
```

```
SELECT pname, price
FROM Product
WHERE price < 100
```

Law of Semijoins

- Input: R(A1,….An), S(B1,….,Bm)
- Output: T(A1,….,An)
- Semjoin is: $R \ltimes S = \Pi_{A1,\ldots,An} (R \bowtie S)$

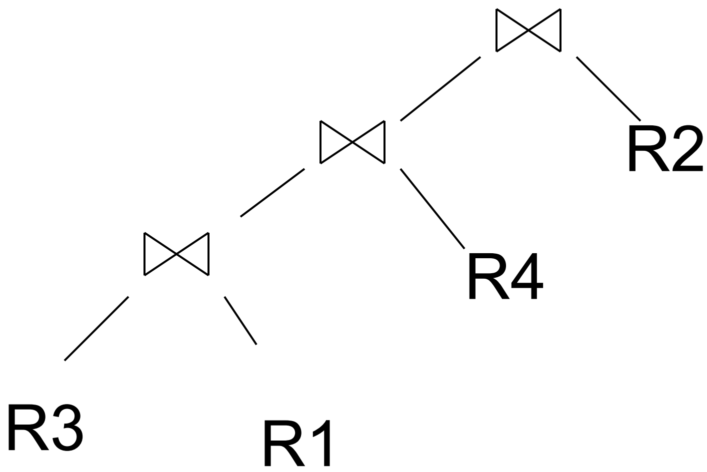
- The law of semijoins is:

$$\boxed{R \bowtie S = (R \ltimes S) \bowtie S}$$

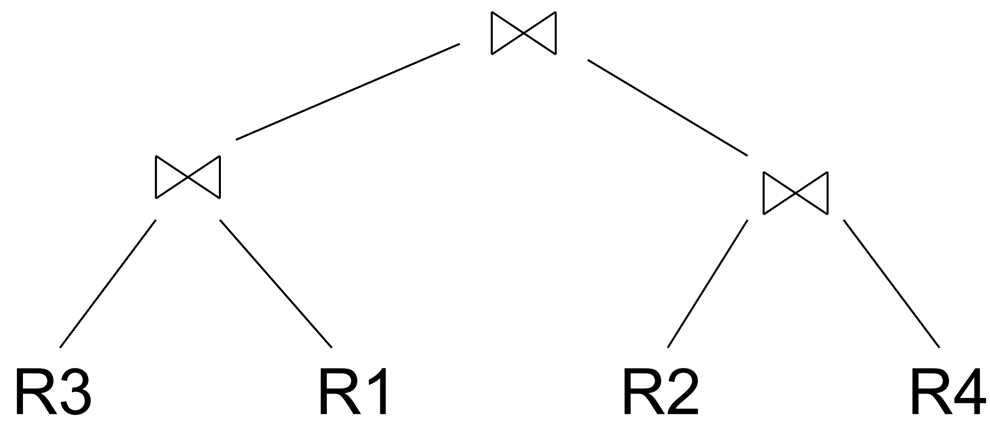
Laws with Semijoins

- Used in parallel/distributed databases
- Often combined with Bloom Filters
- Read pp. 747 in the textbook

Left-Deep Plans and Bushy Plans



Left-deep plan



Bushy plan

System R considered only left deep plans, and so do some optimizers today

Query Optimization

Three major components:

1. Search space
2. Algorithm for enumerating query plans
3. Cardinality and cost estimation

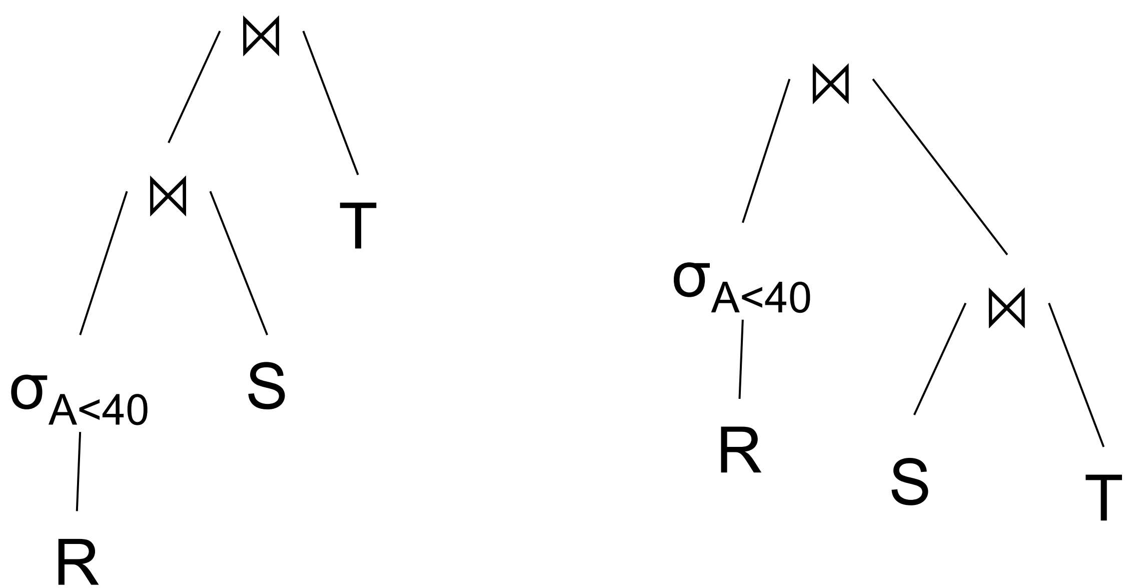
Enumerating Query Plans

- Dynamic programming
  - Pioneered by System R for computing optimal join order, used today by all advanced optimizers
  - See book (won't discuss in class)
- Search space pruning
  - Enumerate partial plans, drop unpromising partial plans
  - Bottom-up v.s. top-down plans
- Access path selection
  - Refers to the plan for accessing a single table

Complete Plans

R(A,B)
S(B,C)
T(C,D)

```
SELECT *
FROM R, S, T
WHERE R.B=S.B and S.C=T.C and R.A<40
```



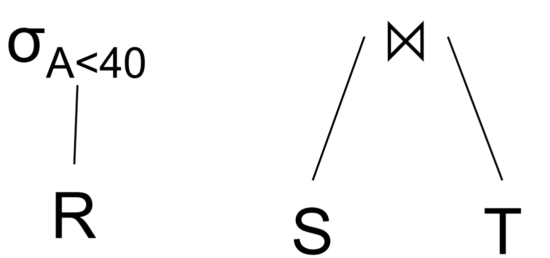
If the algorithm enumerates *complete* plans, then it is difficult to prune out unpromising sets of plans.

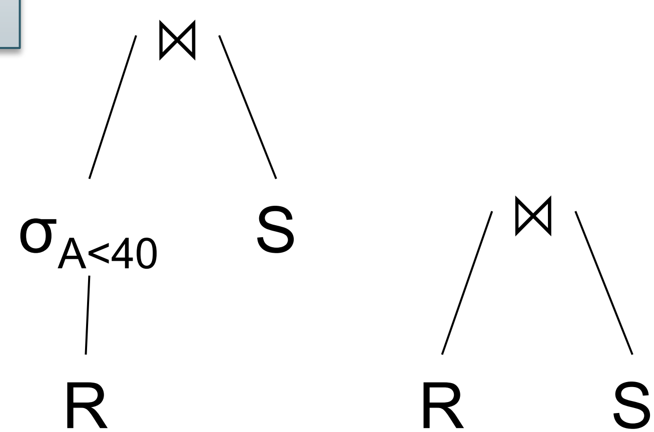
Bottom-up Partial Plans

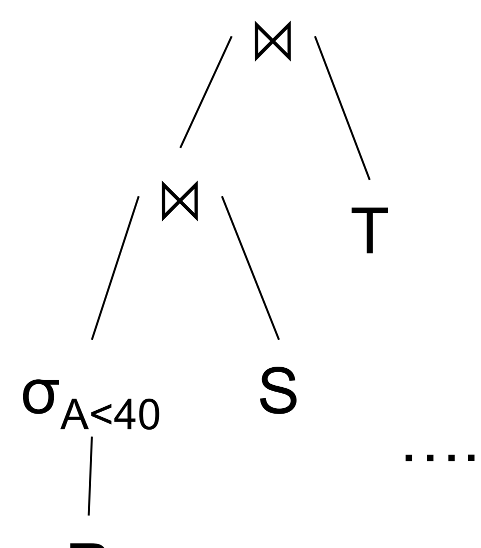
R(A,B)
S(B,C)
T(C,D)

SELECT *
FROM R, S, T
WHERE R.B=S.B and S.C=T.C and R.A<40

If the algorithm enumerates partial bottom-up plans, then pruning can be done more efficiently





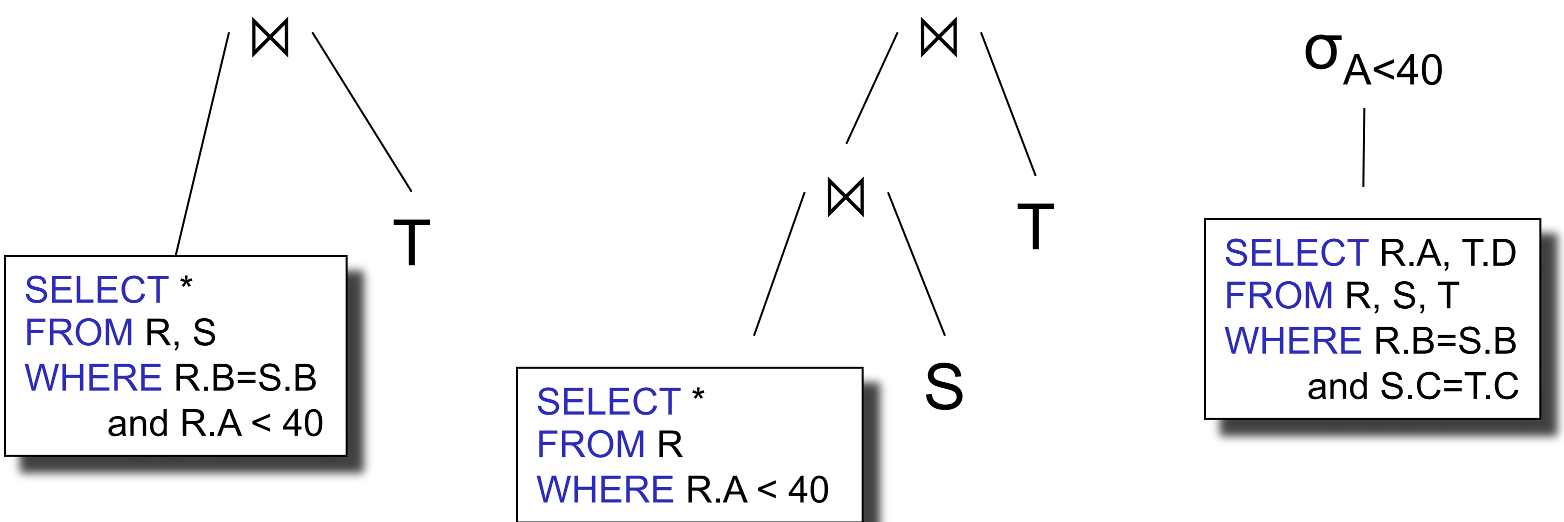


Top-down Partial Plans

R(A,B)
S(B,C)
T(C,D)

```
SELECT *
FROM R, S, T
WHERE R.B=S.B and S.C=T.C and R.A<40
```

Same here.



.....

Access Path Selection

Supplier(sid,sname,scategory,scity,sstate)

B(Supplier) = 10k
T(Supplier) = 1M

$\sigma_{\text{scategory = 'organic'} \wedge \text{scity='Seattle'}}$ (Supplier)

V(Supplier,city) = 1000
V(Supplier,scategory)=100

Clustered index on scity
Unclustered index on (scategory,scity)

Access plan options:

- Table scan: cost = ?
- Index scan on scity: cost = ?
- Index scan on scategory,scity: cost = ?

Access Path Selection

Supplier(sid,sname,scategory,scity,sstate)

B(Supplier) = 10k
T(Supplier) = 1M

$\sigma_{\text{scategory = 'organic'} \wedge \text{scity='Seattle'}}$ (Supplier)

V(Supplier,city) = 1000
V(Supplier,scategory)=100

Clustered index on scity
Unclustered index on (scategory,scity)

Access plan options:

• Table scan:	cost =	10k	= 10k
• Index scan on scity:	cost =	10k/1000	= 10
• Index scan on scategory,scity:	cost =	1M/1000*100	= 10

Query Optimization

Three major components:

1. Search space

2. Algorithm for enumerating query plans

3. Cardinality and cost estimation

3. Cardinality and Cost Estimation

- Collect statistical summaries of stored data
- Estimate size (=cardinality) in a bottom-up fashion
  - This is the most difficult part, and still inadequate in today's query optimizers
- Estimate cost by using the estimated size
  - Hand-written formulas, similar to those we used for computing the cost of each physical operator

Statistics on Base Data

- Collected information for each relation
  - 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

Size Estimation Problem

S = SELECT list
  FROM R1, …, Rn
  WHERE $cond_1$ AND $cond_2$ AND . . . AND $cond_k$

Given T(R1), T(R2), …, T(Rn)
Estimate T(S)

How can we do this ? Note: doesn't have to be exact.

Size Estimation Problem

S = SELECT list
  FROM R1, …, Rn
  WHERE $cond_1$ AND $cond_2$ AND . . . AND $cond_k$

Remark: $T(S) \leq T(R1) \times T(R2) \times \ldots \times T(Rn)$

Selectivity Factor

- Each condition *cond* reduces the size by some factor called *selectivity factor*
- Assuming independence, multiply the selectivity factors

Example

R(A,B)
S(B,C)
T(C,D)

```
SELECT *
FROM R, S, T
WHERE R.B=S.B and S.C=T.C and R.A<40
```

T(R) = 30k, T(S) = 200k, T(T) = 10k

Selectivity of R.B = S.B is 1/3
Selectivity of S.C = T.C is 1/10
Selectivity of R.A < 40 is ½

What is the estimated size of the query output ?

Rule of Thumb

- If selectivities are unknown, then:
  selectivity factor = 1/10
  [System R, 1979]

Using Data Statistics

- Condition is $A = c$ /* value selection on R */
  - Selectivity = 1/V(R,A)
- Condition is $A < c$ /* range selection on R */
  - Selectivity = (c - Low(R, A))/(High(R,A) - Low(R,A))T(R)
- Condition is $A = B$ /* $R \bowtie_{A=B} S$ */
  - Selectivity = 1 / max(V(R,A),V(S,A))
  - (will explain next)

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

Size Estimation for Join

Example:

- $T(R) = 10000$, $T(S) = 20000$
- $V(R,A) = 100$, $V(S,B) = 200$
- How large is $R \bowtie_{A=B} S$?

Histograms

- Statistics on data maintained by the RDBMS
- Makes size estimation much more accurate (hence, cost estimations are more accurate)

Histograms

Employee(<u>ssn</u>, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 19, 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) = 19, max(age) = 68

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

Estimate = 25000 / 50 = 500   Estimate = 25000 * 6 / 50 = 3000

Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Empolyee, age) = 50
min(age) = 19, 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) = 19, 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
- V-Optimal histograms

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..20	20..29	30-39	40-49	50-59	> 60
Tuples	1800	2000	2100	2200	1900	1800

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

V-Optimal Histograms

- Defines bucket boundaries in an optimal way, to minimize the error over all point queries
- Computed rather expensively, using dynamic programming
- Modern databases systems use V-optimal histograms or some variations

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 of Query Optimization

- Three parts:
  - search space, algorithms, size/cost estimation
- Ideal goal: find optimal plan. But
  - Impossible to estimate accurately
  - Impossible to search the entire space
- Goal of today's optimizers:
  - Avoid very bad plans