Query Optimization: Rule-Base Optimization and Size Estimation

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

Leaning Objectives

After this lecture, you should be able to:

- Use relational algebra equivalence rules to determine if two RA expressions are equivalent.
- Develop RA-based heuristic to optimize query execution
- Estimate the size of RA operations

Outline

- Introduction to Query Optimization
- Rule-Based Optimization
  - RA Equivalence Rules
  - Heuristic Based Optimizations
- Estimate Sizes of Operations

Optimization

- *So far, we talked about physical implementations. Next step: how do we use them?*

- At the heart of the database engine
- Step 1: convert the SQL query to a logical plan
- Step 2: find a better logical plan, find an associated physical plan
- (Feed the physical plan into the query processor.)

Converting from SQL to Logical Plans

Need to start someplace..

The easy cases:

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

$\pi_{a1,\ldots,an}(\sigma_C(R1 \;\; x \;\; R2 \; x \;\; \ldots \;\; x \;\; Rk))$

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

Uses "extended" relational algebra, with gamma and delta

$\pi_{a1,\ldots,an}(\gamma_{b1,\ \ldots,\ bm,\ aggs}(\sigma_C(R1 \; x \; R2 \; x \;\; \ldots \;\; x \; Rk)))$

In most of these cases, the x will be a $\bowtie$

Optimization: Logical Query Plan

- Now we have one logical plan. Let's try to make it better.
- Ingredient 1: **Algebraic laws**: what are the ways in which an expression or tree may be rewritten without changing the meaning
- Ingredient 2: **Optimizations**: if there are multiple ways to write the same query, which one to choose.
  - Rule-based (heuristics): apply laws that *seem* to result in cheaper plans
  - Cost-based: estimate size and cost of intermediate results, search systematically for best plan

The three components of an optimizer

- Algebraic laws
- A cost estimator
- Optimization
  - Rule based
  - Cost based

Outline

- ✓ Introduction to Query Optimization
- Rule-Based Optimization
  - RA Equivalence Rules
  - Heuristic Based Optimizations
- Estimate Sizes of Operations

Algebraic Laws

Hold independent of set or bag..

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

Algebraic Laws

- Laws involving selection (set semantics):
  - $\sigma_{C \text{ AND } C'}(R) = \sigma_C(\sigma_{C'}(R)) = \sigma_C(R) \cap \sigma_{C'}(R)$
  - $\sigma_{C \text{ OR } C'}(R) = \sigma_C(R) \cup \sigma_{C'}(R)$
  - $\sigma_C (R \bowtie S) = \sigma_C (R) \bowtie S$
    - When is this true?
      - Certainly true when C involves only attributes of R
      - What if it involves attributes of R and S?
        - For example: R(X, Y), S(Y, Z)
        - Say: Y = 3; X = 3 ^ Z = 5

Laws involving selection (set semantics) Cont.

- $\sigma_C(R - S) = \sigma_C(R) - S$
  - When is this true?
- $\sigma\ C\ (R \cup S) = \sigma\ C\ (R) \cup \sigma\ C\ (S)$
- $\sigma\ C\ (R \cap S) = \sigma\ C\ (R) \cap S$

Algebraic Laws

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

  - Simplify as much as possible by pushing down predicates
    - As close to the relations as possible

Algebraic Laws

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

Algebraic Laws

- Laws involving selection (set semantics):
  - $\sigma_{C \text{ AND } C'}(R) = \sigma_C(\sigma_{C'}(R)) = \sigma_C(R) \cap \sigma_{C'}(R)$
  - $\sigma_{C \text{ OR } C'}(R) = \sigma_C(R) \cup \sigma_{C'}(R)$ 🡐 *what about this one?*
  - $\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$
    - Only when C involves only attributes of R
  - $\sigma_C(R - S) = \sigma_C(R) - S$
  - $\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$
  - $\sigma_C(R \cap S) = \sigma_C(R) \cap S$

Exercise: Think about which of these hold for bag semantics....

Algebraic Laws

- Laws involving selection (bag semantics??):
  - $\sigma_{C \text{ OR } C'}(R) = \sigma_C(R) \cup \sigma_{C'}(R)$ 🡐 *what about this one?*

Algebraic Laws

- Laws involving projections
  - $\Pi_M(\Pi_N(R)) = \Pi_{M \cap N}(R)$
  - $\Pi_M(R \bowtie S) = \Pi_N(\Pi_P(R) \bowtie \Pi_Q(S))$
    - Where N, P, Q are appropriate subsets of attributes of M that are "needed afterwards"

- Example R(A,B,C,D), S(E, F, G)
  - $\Pi_{A,B,G}(R \bowtie_{D=E} S) = \Pi_{?}(\Pi_{?}(R) \bowtie \Pi_{?}(S))$

Outline

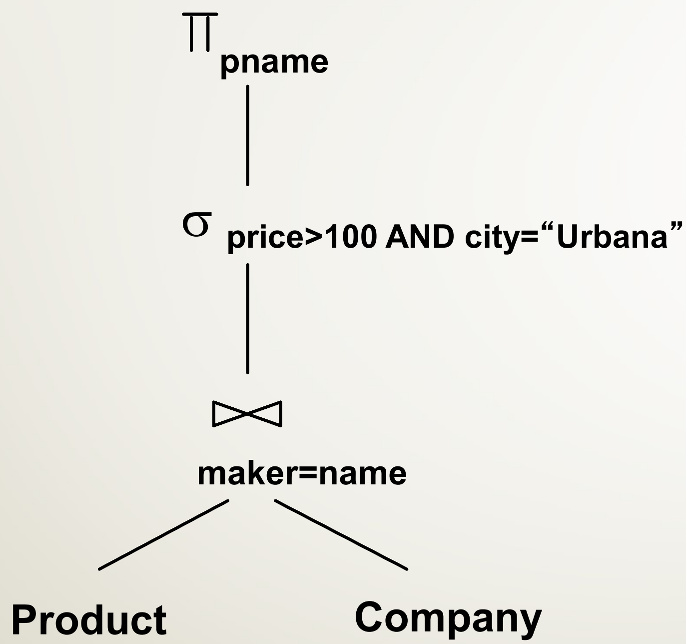
- ✓ Introduction to Query Optimization
- Rule-Based Optimization
  - ✓ RA Equivalence Rules
  - Heuristic Based Optimizations
- Estimate Sizes of Operations

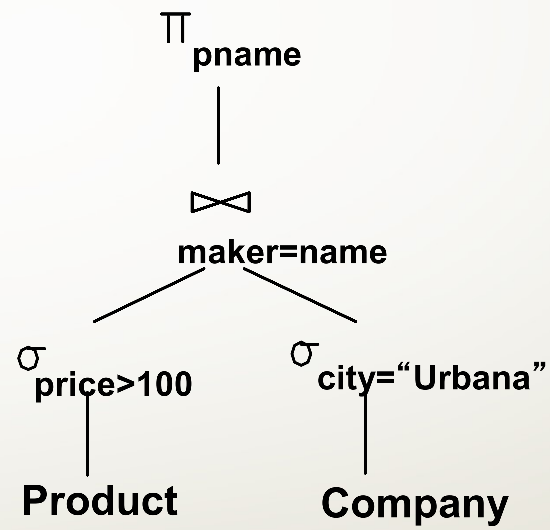
Heuristic Based Optimizations

- Rewriting of logical plan based on specific algebraic laws
- Results in better execution plans most of the time

Heuristic 1: Push selections down

Uses $\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie \sigma_C(S)$





The earlier we process selections, less tuples we need to manipulate higher up in the tree

Pushing selection down: More complex Settings

```
Select   y.name, Max(x.price)
From     product x, company y
Where    x.maker = y.name
GroupBy  y.name
Having   Max(x.price) > 100
```

⇒

```
Select  y.name, Max(x.price)
From    product x, company y
Where   x.maker=y.name  and
        x.price > 100
GroupBy y.name
```

- *For each company, find the maximum price among its products.*
- *But only display (company, maxprice) pair if maxprice > 100*
- Advantage: the size of the join will be smaller.
- Requires transformation rules specific to the grouping/aggregation operators.
- Won't work if we replace Max by Avg.

Heuristic 2

- (In some cases) push selections up, then down
- But ultimately pushed down
- $\sigma_C(R) \bowtie S = \sigma_C(R \bowtie S) = \sigma_C(R) \bowtie \sigma_C(S)$

Typically: apply selections as close to the relations as possible so as to reduce size of intermediate results

Pushing selection up?

Bargain view: for categories with some price<5, find the company that makes the cheapest product in that category, and that price

```
Select   V2.name, V2.price
From     V1, V2
Where    V1.category = V2.category  and
           V1.minprice = V2.price
```

```
Create View V1 AS
Select   x.category,
      Min(x.price) AS minprice
From     product x
Where    x.price < 5
GroupBy  x.category
```

```
Create View V2 AS
Select   y.name, x.category, x.price
From     product x, company y
Where    x.maker=y.name
```

Pushing selection up …

Bargain view: for categories with some price<5, find the company that makes the cheapest product in that category, and that price

```
Select   V2.name, V2.price
From     V1, V2
Where    V1.category = V2.category  and
           V1.minprice = V2.price AND V1.minprice < 5
```

```
Create View V1 AS
Select   x.category,
      Min(x.price) AS minprice
From     product x
Where    x.price < 5
GroupBy  x.category
```

```
Create View V2 AS
Select   y.name, x.category, x.price
From     product x, company y
Where    x.maker=y.name
```

… and then down

Bargain view: for categories with some price<5, find the company that makes the cheapest product in that category, and that price

```
Select   V2.name, V2.price
From     V1, V2
Where    V1.category = V2.category  and
           V1.minprice = V2.price AND V1.minprice < 5
```

```
Create View V1 AS
Select   x.category,
      Min(x.price) AS minprice
From     product x
Where    x.price < 5
GroupBy  x.category
```

```
Create View V2 AS
Select   y.name, x.category, x.price
From     product x, company y
Where    x.maker=y.name AND x.price < 5
```

Outline

- ✓ Introduction to Query Optimization
- ✓ Rule-Based Optimization
  - ✓ RA Equivalence Rules
  - ✓ Heuristic Based Optimizations
- Estimating Sizes of Operations

Estimating Sizes

- Need size in order to estimate cost
- Example:
  - Cost of partitioned hash-join $R \bowtie S$ is $3B(R) + 3B(S)$
    - $B(R) = T(R)/$ block size
    - $B(S) = T(S)/$ block size
  - So, we need to estimate $T(R)$, $T(S)$

Estimating Sizes: Selection

Estimating the size of a selection

- $S = \sigma_{A=c}(R)$
  - V(R, A) = number of distinct values of A in R
  - T(S) can be anything from 0 to T(R) – V(R,A) + 1
  - *Why 0? Why T(R) – V(R,A) + 1?*
  - With the assumption of uniform distribution, value:
    - T(S) = T(R)/V(R,A)

Estimating Sizes: Join

Estimating the size of a join, $R \bowtie_A S$, extremes:

- When the set of A values are disjoint, then
  $T(R \bowtie_A S) = 0$
- When A is a key in S and a foreign key in R, then
  $T(R \bowtie_A S) = T(R)$

Estimating Sizes: Join

Simplifying assumptions:

- *Containment of values*: if V(R,A) <= V(S,A), then the set of A values of R is included in the set of A values of S
  - Note: this holds when A is a foreign key in R, and a key in S; but we assume this more broadly (even if this condition is not true)
- *Preservation of value sets*:
  - for any other attribute B, from one of the relations

    $V(R \bowtie_A S, B) = V(R, B) \quad (\text{or } V(S, B))$
  - but for the joining attribute (A):
    $V(R \bowtie_A S, A) = \min(V(R,A), V(S,A))$

Estimating Sizes: Join

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

- Then each tuple r in R joins *some* tuple(s) in S
  - How many ?
  - On average T(S)/V(S,A)
  - r will contribute T(S)/V(S,A) tuples in $R \bowtie_A S$
- Hence $T(R \bowtie_A S) = T(R)\ T(S)\ /\ V(S,A)$

In general: $T(R \bowtie_A S) = T(R)\ T(S)\ /\ \max(V(R,A),V(S,A))$

Estimating Sizes

Example:

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

Estimating Sizes

Example:

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

Answer: $T(R \bowtie_A S) = 10000 * 20000/200 = 1M$

Estimating Sizes: Joining on multiple attributes

Joins on more than one attribute:

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

  $T(R)\, T(S)/[\max(V(R,A),V(S,A)) * \max(V(R,B),V(S,B))]$

Three way join

- $T(R) = 10{,}000$, $T(S) = 20{,}000$, $T(U) = 10{,}000$
- $R(A,B)$, $S(B, C)$, $U(C, D)$
- $V(R, B) = 100$, $V(S, B) = 50$, $V(S, C) = 40$, $V(U, C) = 20$

Three way join

- $T(R) = 10{,}000$, $T(S) = 20{,}000$, $T(U) = 10{,}000$
- $R(A,B)$, $S(B, C)$, $U(C, D)$
- $V(R, B) = 100$, $V(S, B) = 50$, $V(S, C) = 40$, $V(U, C) = 20$

R join S = 10,000 * 20,000/100

S join U = 20,000 * 10,000/40

R join U = 10,000 * 10,000

S join U join R (in any order) = 10,000*20,000*10,000/(40*100)

Size Estimation Summary

- Estimating the size of a selection

$S = \sigma_{A=c}(R)$: $T(S) = T(R)/V(R,A)$

- Estimating the size of a join:

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

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

$T(R \times S) = T(R)\ T(S)$

Outline

- ✓ Introduction to Query Optimization
- ✓ Rule-Based Optimization
  - ✓ RA Equivalence Rules
  - ✓ Heuristic Based Optimizations
- ✓ Estimating Sizes of Operations