CSE 562
Database Systems

Query Processing:
Algebraic Optimization

Some slides are based or modified from originals by
Database Systems: The Complete Book,
Pearson Prentice Hall 2nd Edition
©2008 Garcia-Molina, Ullman, and Widom

cse@buffalo

Outline – Query Optimization

• Overview
• Relational algebra level
  – Algebraic Transformations
• Detailed query plan level
  – Estimate Costs
    – Estimating size of results
    – Estimating # of IOs
  – Generate and compare plans

Relational Algebra Optimization

• Transformation rules
  (preserve equivalence)
• What are good transformations?

Algebraic Rewritings:
Commutative and Associative Laws

• Question 1: Do the above hold for both sets and bags?
• Question 2: Do commutative and associative laws hold
  for arbitrary Theta Joins?
Algebraic Rewritings: Commutative and Associative Laws

**Union**

- Commutative:
  \[ R \cup S \rightarrow S \cup R \]
- Associative:
  \[ (R \cup S) \cup T \rightarrow R \cup (S \cup T) \]

**Intersection**

- Commutative:
  \[ R \cap S \rightarrow S \cap R \]
- Associative:
  \[ (R \cap S) \cap T \rightarrow R \cap (S \cap T) \]

**Question 1:** Do the above hold for both sets and bags?

**Question 2:** Is difference commutative and associative?

Algebraic Rewritings for Selection: Decomposition of Logical Connectives

**Question**

- \( \sigma_{\text{cond1}} \) AND \( \neg \sigma_{\text{cond2}} \)
- \( \sigma_{\neg \text{cond1}} \)
- \( \sigma_{\text{cond1}} \) OR \( \sigma_{\text{cond2}} \)

**Complete**

- \( R \)

Does it apply to bags?

Pushing Selection Through Binary Operators: Union and Difference

- **Union**
  \[ \sigma_{\text{cond1}} \]
  \[ \sigma_{\text{cond2}} \]

- **Difference**
  \[ \sigma_{\text{cond1}} \]
  \[ \neg \sigma_{\text{cond2}} \]

**Exercise:** Do the rules for intersection
The right direction requires that \( \text{cond} \) refers to \( S \) attributes only.

The right direction requires that \( \text{cond} \) refers to \( S \) attributes only.

The right direction requires that all the attributes used by \( \text{cond} \) appear in both \( R \) and \( S \).

**Pushing Selection Through Cartesian Product and Join**

**Rules: \( \pi + \sigma \) combined**

Let \( X = \) subset of \( R \) attributes
\( Z = \) attributes in predicate \( P \) (subset of \( R \) attributes)

\[
\pi_X[\sigma_P(R)] = \pi_X[\sigma_P(\pi_X(R))]
\]

**Pushing Simple Projections Through Binary Operators: Union**

- A projection is simple if it only consists of an attribute list

- **Question 1**: Does the above hold for both bags and sets?
- **Question 2**: Can projection be pushed below intersection and difference?
- Answer for both bags and sets

**Pushing Simple Projections Through Binary Operators: Join and Product**

- **Exercise**: Do the rule for theta join
- **Exercise**: Write the rewriting rule that pushes projection below theta join

- \( B \) is the list of \( R \) attributes that appear in \( A \)
- Similar for \( C \)

- **Question**: What is \( B \) and \( C \)?
**Rules: \( \pi + \sigma + \bowtie \) combined**

\[
\begin{array}{c}
\pi_{xy} \\
\sigma_{\text{cond}} \\
\bowtie \\
R \\
\downarrow \\
\uparrow \\
S \\
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\pi_{xy} \\
\sigma_{\text{cond}} \\
\bowtie \\
R \\
\downarrow \\
\uparrow \\
S \\
\end{array} \quad \begin{array}{c}
\pi_{xz'} \\
\downarrow \\
R \\
\uparrow \\
S \\
\end{array}
\]

- \( Z' = Z \cup \{ \text{attributes used in cond} \} \)

**Projection Decomposition**

- Let \( X = \text{set of attributes} \)
- \( Y = \text{set of attributes} \)
- \( XY = X \cup Y \)

**Some Rewriting Rules Related to Aggregation: SUM**

- \( \sigma_{\text{cond}}[\text{SUM}_{\text{GroupbyList};\text{GroupedAttribute}} \rightarrow \text{ResultAttribute}] (R) \)
- \( \text{SUM}_{\text{GroupbyList};\text{GroupedAttribute}} \rightarrow \text{ResultAttribute} [\sigma_{\text{cond}}(R)] \)
  - if \( \text{cond} \) involves only the \( \text{GroupbyList} \)

- \( \text{SUM}_{\text{GL};\text{GA}} \rightarrow \text{RA} (R \cup S) \)
  - \( \text{PLUS}_{\text{RA1};\text{RA2};\text{RA}}([\text{SUM}_{\text{GL};\text{GA}} \rightarrow \text{RA1}] R \triangleq [\text{SUM}_{\text{GL};\text{GA}} \rightarrow \text{RA2}] S] \)

- \( \text{SUM}_{\text{GL2};\text{RA1} \rightarrow \text{RA2}}[\text{SUM}_{\text{GL1};\text{GA} \rightarrow \text{RA1}} (R)] \)
- \( \text{SUM}_{\text{GL2};\text{GA} \rightarrow \text{RA2}} (R) \)

**Question:** does the above hold for both bags and sets?
Derived Rules: $\sigma + \bowtie$ combined

More Rules can be Derived:

$\sigma_{p\land q} (R \bowtie S) = [\sigma_p (R)] \bowtie [\sigma_q (S)]$

$\sigma_{p\land q\land m} (R \bowtie S) = \sigma_m [\sigma_p (R) \bowtie \sigma_q (S)]$

$\sigma_{p\lor q} (R \bowtie S) = [\sigma_p (R) \bowtie S] \cup [R \bowtie \sigma_q (S)]$

- $p$ only at $R$
- $q$ only at $S$
- $m$ at both $R$ and $S$

Which are “good” transformations?

- $\sigma_{p_1 \land p_2} (R) \rightarrow \sigma_{p_1} [\sigma_{p_2} (R)]$
- $\sigma_p (R \bowtie S) \rightarrow [\sigma_p (R)] \bowtie S$
- $R \bowtie S \rightarrow S \bowtie R$
- $\pi_x [\sigma_p (R)] \rightarrow \pi_x \{\sigma_p [\pi_x z (R)]\}$

Derivation for first one

$\sigma_{p\land q} (R \bowtie S) =$

$\sigma_{p} [\sigma_{q} (R \bowtie S)] =$

$\sigma_{p} [R \bowtie \sigma_{q} (S)] =$

$[\sigma_p (R)] \bowtie [\sigma_q (S)]$

Conventional Wisdom: Do Projects Early

$\pi_E \rightarrow \pi_A$ AND $B = \text{Late}$

$R(A, B, C, D, E) \rightarrow R$
But...

What if we have A, B indexes?

B = “cat”

A=3

Intersect pointers to get pointers to matching tuples

More Transformations in Textbook

- Eliminate common sub-expressions
- Other operations: duplicate elimination

Bottom line

- No transformation is always good at the logical query plan level
- Usually good:
  - early selections
  - elimination of Cartesian products
  - elimination of redundant sub-expressions
- Many transformations lead to “promising” plans
  - Commuting/rearranging joins
  - In practice too “combinatorially explosive” to be handled as rewriting of logical query plan