Query Processing and Advanced Queries

Query Optimization (1)
Introduction

- How to apply the algebraic laws to improve a logical query plan?
- Goal: minimize the size (number of tuples, number of attributes) of intermediate results.
- Push selections down in the expression tree as far as possible.
- Push down projections, or add new projections where applicable.
Improving Logical Query Plans

Pushing Selections

- Replace the left side of one of these (and similar) rules by the right side:

\[ \sigma_{p_1 \land p_2} (R) \rightarrow \sigma_{p_1} [\sigma_{p_2} (R)] \]

\[ \sigma_p (R \bowtie S) \rightarrow [\sigma_p (R)] \bowtie S \]

- Can greatly reduce the number of tuples of intermediate results.
**Pushing Projections**

- Replace the left side of one of these (and similar) rules by the right side:

\[ \pi_x [\sigma_p (R)] \rightarrow \pi_x \{ \sigma_p [\pi_{xz} (R)] \} \]

- Reduces the number of attributes of intermediate results and possibly also the number of tuples.
Pushing Projections

Consider the following example:

\[
R(A,B,C,D,E) \\
P: (A=3) \land (B=\text{“cat”})
\]

Compare:

\[
\pi_E \{\sigma_p (R)\} \quad \text{vs.} \quad \pi_E \{\sigma_p \{\pi_{ABE}(R)\}\}
\]
**Pushing Projections**

- What if we have indexes on A and B?

  B = “cat”  
  A=3

  Intersect pointers to get pointers to matching tuples

- Efficiency of logical query plan may depend on choices made during refinement to physical plan.

- No transformation is always good!
Improving Logical Query Plans

**Grouping Associative / Commutative Operators**

- For operators which are commutative and associative, we can order and group their arguments arbitrarily.
- In particular: natural join, union, intersection.
- As the last step to produce the final logical query plan, group nodes with the same (associative and commutative) operator into one n-ary node.
- Best grouping and ordering determined during the generation of physical query plan.
Improving Logical Query Plans

Grouping Associative / Commutative Operators

Before:

```
  U  
 /   
C   D  E
 /     |
A     B
```

After:

```
  U  
 /   
C   D  E
 /     |
A     B
```
So far, we have parsed and transformed an SQL query into an optimized logical query plan. In order to refine the logical query plan into a physical query plan, we consider alternative physical plans, estimate their cost, and pick the plan with the least (estimated) cost. We have to estimate the cost of a plan without executing it. And we have to do that efficiently!
When creating a physical query plan, we have to decide on the following issues.

- order and grouping of operations that are associative and commutative,
- algorithm for each operator in the logical plan,
- additional operators which are not represented in the logical plan, e.g. sorting,
- the way in which intermediate results are passed from one operator to the next, e.g. by storing on disk or passing one tuple at a time.
Intermediate relations are the output of some relational operator and the input of another one. The size of intermediate relations has a major impact on the cost of a physical query plan. It impacts in particular:
- the choice of an implementation for the various operators and
- the grouping and order of commutative / associative operators.
A method for estimating the size of an intermediate relation should be
- reasonably accurate,
- efficiently computable,
- not depend on how that relation is computed.

We want to rank alternative query plans w.r.t. their estimated costs.

Accuracy of the absolute values of the estimates not as important as the accuracy of their ranks.
Size estimates make use of the following statistics for relation R:

- \( T(R) \): # tuples in R
- \( S(R) \): # of bytes in each R tuple
- \( B(R) \): # of blocks to hold all R tuples
- \( V(R, A) \): # distinct values for attribute A in R.
- \( \text{MIN}(R,A) \): minimum value of attribute A in R.
- \( \text{MAX}(R,A) \): maximum value of attribute A in R.

Statistics need to be maintained up-to-date under database modifications!
### Estimating the Cost of Operations

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>1</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>cat</td>
<td>1</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>30</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>40</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>bat</td>
<td>1</td>
<td>50</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

- **A**: 20 byte string
- **B**: 4 byte integer
- **C**: 8 byte date
- **D**: 5 byte string

\[
T(R) = 5 \\
S(R) = 37 \\
V(R,A) = 3 \\
V(R,C) = 5 \\
V(R,B) = 1 \\
V(R,D) = 4
\]
Estimating the Cost of Operations

- Size estimate for $W = R_1 \times R_2$
  
  $T(W) = T(R_1) \times T(R_2)$
  
  $S(W) = S(R_1) + S(R_2)$

- Size estimate for $W = \sigma_{A=a} (R)$

- Assumption: values of $A$ are uniformly distributed over the attribute domain

  $T(W) = T(R)/V(R,A)$
  
  $S(W) = S(R)$
Estimating the Cost of Operations

- Size estimate for $W = \sigma_{z \geq \text{val}} (R)$
  - **Solution 1**: on average, half of the tuples will satisfy an inequality condition
    \[ T(W) = T(R)/2 \]
  - **Solution 2**: more selective queries are more frequent, e.g. professors who earn more than $200,000$ (rather than less than $200,000$)
    \[ T(W) = T(R)/3 \]
Solution 3: estimate the number of attribute values in query range

- Use minimum and maximum value to define range of the attribute domain.
- Assume uniform distribution of values over the attribute domain.
- Estimate is the fraction of the domain that falls into the query range.
Estimating the Cost of Operations

\[
\begin{array}{|c|c|}
\hline
R & Z \\
\hline
\end{array}
\]

\[
\text{MIN}(R,Z)=1 \quad \text{V}(R,Z)=10
\]

\[
\text{W}= \sigma_{z \geq 15} (R) \quad \text{MAX}(R,Z)=20
\]

\[
f = \frac{20-15+1}{20-1+1} = \frac{6}{20} = 0.3 \quad \text{(fraction of range)}
\]

\[
T(W) = f \times T(R)
\]
Size estimate for $W = R1 \bowtie R2$

Consider only *natural join* of $R1(X,Y)$ and $R2(Y,Z)$.

We do not know how the $Y$ values in $R1$ and $R2$ relate:
- disjoint, i.e. $T(R1 \bowtie R2) = 0$,
- $Y$ may be a foreign key of $R1$ and the primary key of $R2$, i.e. $T(R1 \bowtie R2) = T(R1)$,
- all the $R1$ and all the $R2$ tuples have the same $Y$ value, i.e. $T(R1 \bowtie R2) = T(R1) \times T(R2)$.
Estimating the Cost of Operations

- Make several simplifying assumptions.
- *Containment of value sets:*
  
  \[ V(R1,Y) \leq V(R2,Y) \Rightarrow \]
  
  every Y value in R1 is in R2

  \[ V(R2,Y) \leq V(R1,Y) \Rightarrow \]
  
  every Y value in R2 is in R1

- This assumption is satisfied when Y is foreign key in R1 and primary key in R2.
- Is also approximately true in many other cases.
Preservation of value sets:
If A is an attribute of R1 but not of R2, then
\[ V(R1 \bowtie R2, A) = V(R1, A). \]
Again, holds if the join attribute Y is foreign key in R1 and primary key in R2.
Can only be violated if there are “dangling tuples” in R1, i.e. R1 tuples that have no matching partner in R2.
Uniform distribution of attribute values:
the values of attribute A are uniformly distributed over their domain, i.e. \( P(A=a_1) = P(A=a_2) = \ldots = P(A=a_k) \).

This assumption is necessary to make cost estimation tractable.

It is often violated, but nevertheless allows reasonably accurate ranking of query plans.
Independence of attributes:
the values of attributes A and B are independent from each other, i.e. $P(A=a \mid B=b) = P(A=a)$ and $P(B=b \mid A=a) = P(B=b)$.

This assumption is necessary to make cost estimation tractable.

Again, often violated, but nevertheless allows reasonably accurate ranking of query plans.
Suppose that $t_1$ is some tuple in $R_1$, $t_2$ some tuple in $R_2$.

What is the probability that $t_1$ and $t_2$ agree on the join attribute $Y$?

If $V(R_1,Y) \leq V(R_2,Y)$, then the $Y$ value of $t_1$ appears in $R_2$, because of the containment of value sets.

Assuming uniform distribution of the $Y$ values in $R_2$ over their domain, the probability of $t_2$ having the same $Y$ value as $t_1$ is $1/V(R_2,Y)$.
If $V(R2,Y) \leq V(R1,Y)$, then the $Y$ value of $t2$ appears in $R1$, and the probability of $t1$ having the same $Y$ value as $t2$ is $1 / V(R1,Y)$.

$T(W) =$ number of pairs of tuples from $R1$ and $R2$ times the probability that an arbitrary pair agrees on $Y$.

$T(R1 \bowtie R2) = T(R1) T(R2) / \max(V(R1,Y), V(R2,Y))$. 
For complex query expressions, need to estimate T,S,V results for intermediate results.

For example, \( W = [\sigma_{A=a} (R1) ] \bowtie R2 \)

treat as relation \( U \)

\[
T(U) = T(R1)/V(R1,A) \\
S(U) = S(R1) \\
\]

Also need \( V(U, \ast) \) for all attributes of \( U(R1) \)!
Estimating the Cost of Operations

R 1

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>cat</td>
<td>1</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>30</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>40</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>bat</td>
<td>1</td>
<td>50</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

V(R1,A)=3
V(R1,B)=1
V(R1,C)=5
V(R1,D)=3

U = \sigma_{A=a} (R1)

V(U,A) = 1  V(U,B) = 1  V(U,C) = \frac{\text{|R1|}}{V(R1,A)}

V(U,D) ... somewhere in between
R1(A,B), R2(A,C).

Consider join \( U = R1 \bowtie R2 \).

Estimate \( V \) results for \( U \).

\[ V(U,A) = \min \{ V(R1, A), V(R2, A) \} \]

Holds due to containment of value sets.

\[ V(U,B) = V(R1, B) \]
\[ V(U,C) = V(R2, C) \]

Holds due to preservation of value sets.
Consider the following example:

\[ Z = R1(A,B) \Join \bowtie R2(B,C) \bowtie R3(C,D) \]

\[ T(R1) = 1000 \quad V(R1,A)=50 \quad V(R1,B)=100 \]
\[ T(R2) = 2000 \quad V(R2,B)=200 \quad V(R2,C)=300 \]
\[ T(R3) = 3000 \quad V(R3,C)=90 \quad V(R3,D)=500 \]

Group and order as \( (R1 \bowtie R2) \bowtie R3 \)
Partial result: $U = R_1 \bowtie R_2$

$T(U) = \frac{1000 \times 2000}{200}$

$V(U,A) = 50$

$V(U,B) = 100$

$V(U,C) = 300$
Final result: $Z = U \bowtie R3$

$$T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300}$$

$V(Z,A) = 50$
$V(Z,B) = 100$
$V(Z,C) = 90$
$V(Z,D) = 500$