Query Processing: Cost-based Query Optimization

Physical Query Plan Optimization

Logical query rewriting can produce one (or more candidate) query plan(s). Physical query plan optimization stage involves the following operations:

- Selection of the order and grouping in which associative-and-commutative operations are to be executed.
- Selection of the appropriate algorithm for each logical query plan operator.
- Insertion of additional operations: scans, sorts, etc., needed for faster performance.
- Selection of the means of passing results of one operation to the next operation: through main memory buffer, through temporary disk storage or via tuple-at-a-time iterators.

Basics of Cost Estimation

In order to be able to tell, which query plans are better, we need to be able to predict/estimate the I/O costs of the plans. The I/O costs depend on two things:

- the specific chosen to execute each operation;
- (estimated) sizes of all intermediate results.

We know that given the sizes of the input relations, we can estimate the I/O costs of each execution algorithm. We, thus, would like to have an approach to estimation of sizes of intermediate results which has the following properties:

1. It yields accurate estimates (Accuracy);
2. It is easy to compute (Efficiency);

3. It is logically consistent: the estimates do not depend on how the intermediate result was computed, only on what the intermediate result looks like (Consistency)

Size estimation is a heuristic process. Some standard approaches are described below.

**Projection**

Let \( R \) be a relation, and consider the operation \( \pi_L(R) \).

Projection operation does not remove tuples from the relation. However, the size of each tuple shrinks \(^1\).

Let \( m \) be the size of a tuple in \( R \), and \( m' \) be the size of the tuple in \( \pi_L(R) \). \( m' \) can be computed in a straightforward manner from \( m \), knowing the schema of \( R \).

Then, \( B(\pi_L(R)) \) can be estimated as follows:

\[
B(\pi_L(R)) = \frac{m'}{m} B(R).
\]

**Selection**

**Case 1**: \( \sigma_{A=c}(R) \).

This operation will select only the tuples in \( R \) for which the value of attribute \( A \) is \( c \). There are \( V(R, A) \) different values of the attribute \( A \) in \( R \), so, we can estimate the number of tuples in the result as

\[
T(\sigma_{A=c}(R)) = \frac{T(R)}{V(R, A)}.
\]

**Case 2**: \( \sigma_{A<c}(R) \). (or any other inequality)

Standard estimation technique is

\[
T(\sigma_{A<c}(R)) = \frac{T(R)}{3}.
\]

Another possible solution is as follows. Let \( V_c(R, A, c) \) be the number of unique values of \( A \) in \( R \) that are less than \( c \). In this case, we can estimate

\[
T(\sigma_{A<c}(R)) = \frac{T(R) \cdot V_c(R, A, c)}{V(R, A)}.
\]

**Case 3**: \( \sigma_{A\neq c}(R) \).

Standard estimate, applicable when \( V(R, A) \) is very large is

\(^1\)A more general version of projection operation also may allow for increase in size of the tuple, but such increases can also be predicted fairly well.
\[ T(\sigma_{A \neq c}(R)) = T(R). \]

If \( V(R, A) \) is not large, while \( T(R) \) is large, the following estimate may be better:

\[ T(\sigma_{A \neq c}(R)) = T(R) \cdot \frac{V(R, A) - 1}{V(R, A)}. \]

**Case 4** \( \sigma_{C_1 \text{AND} C_2}(R) \).
Treat this as \( \sigma_{C_1} (\sigma_{C_2}(R)) \), and cascade the estimates.

**Case 4** \( \sigma_{C_1 \text{OR} C_2}(R) \).
We know that

\[ \max(T(\sigma_{C_1}(R)), T(\sigma_{C_2}(R))) \leq T(\sigma_{C_1 \text{OR} C_2}(R)) \leq T(\sigma_{C_1}(R)) + T(\sigma_{C_2}(R)). \]

The left-hand-side estimate corresponds to positive correlation, which states that one condition subsumes the other completely. The right-hand-side estimate corresponds to the negative correlation/mutual exclusion assumption, which states that no tuple can satisfy both conditions at the same time.

We can also construct an estimate for an independence assumption:

\[ T(\sigma_{C_1 \text{OR} C_2}(R)) = T(R)(1 - (1 - \frac{1}{V(R, A)})^2). \]

If we have better estimates \( m_1 \) and \( m_2 \) for \( \sigma_{C_1}(R) \) and \( \sigma_{C_2}(R) \), this becomes:

\[ T(\sigma_{C_1 \text{OR} C_2}(R)) = T(R)(1 - (1 - \frac{m_1}{T(R)})(1 - \frac{m_2}{T(R)})). \]

**Union**

For bag union \( T(R \cup_{\text{bag}} S) = T(R) + T(S) \).
For set union, we have

\[ \max(T(R), T(S)) \leq T(R \cup S) \leq T(R) + T(S). \]

A possible estimate is the mid-point:

\[ T(R \cup S) = \max(T(R), T(S)) + \frac{T(R) + T(S)}{2}. \]

**Intersection**

\[ 0 \leq T(R \cap S) \leq \min(T(R), T(S)). \]

One possible estimate is
\[ T(R \cap S) = \frac{\min(T(R), T(S))}{2}. \]

Another possibility is to use formulas for natural join, as \( R \cap S = R \bowtie S \). (Joins will be discussed later).

**Difference**

\[ T(R) \leq T(R - S) \leq \max(T(R) - T(S), 0). \]

A possible estimate is

\[ T(R - S) = \max(0, T(R) - \frac{1}{2}T(S)). \]

**Duplicate Elimination**

Generally speaking

\[ T(\delta(R)) = V(R, (A_1, \ldots, A_n)), \]

if \( R \)'s schema is \( R(A_1, \ldots, A_n) \). However, this information may not be immediately available.

One possible estimate (when \( T(R) \) is very large) is

\[ T(\delta(R)) = \Pi_{i=1}^n V(R, A_i), \]

i.e., the number of theoretically possible distinct tuples.

We can also use the rule

\[ T(\delta(R)) = \min(0.5 \cdot T(R), \Pi_{i=1}^n V(R, A_i)). \]

**Grouping and Aggregation**

Let \( L = (G_1, \ldots, G_k) \). If we know \( V(R, (G_1, \ldots, G_k)) \), then

\[ T(\gamma_L(R)) = V(R, (G_1, \ldots, G_k)). \]

Otherwise, we may estimate the number similarly to the case of duplicate elimination:

\[ T(\gamma_L(R)) = \Pi_{i=1}^k V(R, G_i), \]

i.e., the number of theoretically possible distinct tuples.

We can also use the rule

\[ T(\gamma_L(R)) = \min(0.5 \cdot T(R), \Pi_{i=1}^k V(R, G_i)). \]