Query Optimization

DBMS Lecture notes 31st November 2011 Monday

By: Akash Deep Rawat(09010104).
Praveen kumar Chaturvedi(090101).

For querying a database in a DBMS ,Resources at disposal are CPU,RAM,I/O and Network.

Out of these 4 ,Network communication and I/O are bottleneck for the performance of DBMS.

• The task of a query optimizer is to accept as input a tree of relational algebra operators (or, equivalently, the parse tree for a query language statement), and to produce a query execution plan. This plan specifies exactly which operations should be performed, in which order.

– There are a large number of query execution plans that are equivalent (in terms of their result set) to a typical relational algebra expression. So one of the tasks of the query optimizer is to search among this solution space of equivalent query plans to choose the "best" one.
∗ The solution space is typically so large that an exhaustive search is impossible, so various heuristic techniques need to be utilized. In practise, the goal of a query optimizer is not to find the globally optimal query plan, but merely to avoid an extremely bad plan.

– Given a particular query plan, the second task of the query optimizer is to estimate the expense of executing that query plan (largely in terms of disk I/O operations).
∗ This is difficult to do precisely: in practise, much of this process comes down to a series of educated guesses and estimates.

Relational Algebra Equivalences
• Selections are commutative: they can be applied in any order, so we are free to apply the most highly-selective predicate first. Also, selections allow for cascading: we can rewrite a predicate of n terms of the form p1 ∧ p2 ∧ . . . ∧ pn into n distinct predicates, one for each term.
• Projections can also be cascaded: if we apply n projection operations after one another, the result is equivalent to applying only the last projection.
• Joins are commutative: the order of their operands can be changed without affecting the result. Joins are also associative: given three relations, joining them in any order will always produce the same final result relation.
– In combination, these two equivalences can produce a huge number of equivalent relational algebra expressions for a query involving a significant number of joins.
• There are some other equivalences involving multiple operators, such as the ability to produce a join from a cross-product and a selection, or to "push down" a selection, but these are all fairly logical.

note:σ and Π reduce the size of data ,while X(cross product) and Ж increase the data size.

Examples:

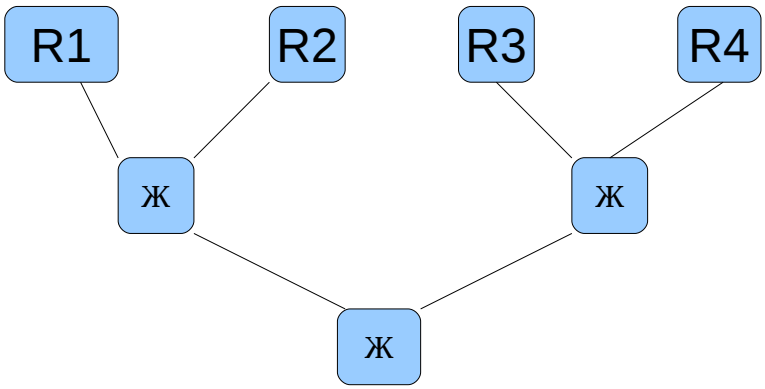
Π1(Π2(Π3(Π4(R)))) = Π1(R).

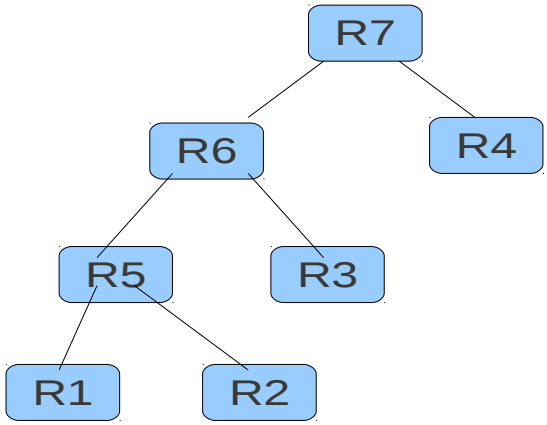
σ1(σ2(σ3(σ4(R)))) = σ1(R).

(R1 Ж R2) Ж (R4 Ж R5) = (((R1 Ж R2) Ж R4) Ж R5). Where 'Ж' is natural join.

Example (R1 Ж R2) Ж (R3 Ж R4). See below diagram for evaluation stratergy.

1.





note: R5 =(R1 Ж R2) and similarily for other nodes.

Note that First diagram is for evaluation stratergey (R1 Ж R2) Ж (R3 Ж R4)) while the second diagram is for evaluation stratergy (((R1 Ж R2) Ж R3)Ж R4).Although both stratergies give same result but cost of evaluation for both of them is different.

Also the 2nd Figure reprsents Pipelining stratergy in which output corresponding to Join operation on R1 and R2 is pipelined to R5 i.e. Output from R1 and R2 keeps coming on and keeps operating with R3 to produce output for next level hierarchy.The stratergy in figure 2 is called **left deep evaluation Plan.**

While in case of first evaluation stratergy, for o/p of (R1 Ж R2) and (R3 Ж R4) to get natural operated with Ж both (R1 Ж R2) and (R3 Ж R4) should get completely evaluated themselves first. Threfore a left-deep query plan means that fewer tuples need to be held in memory at any time.

Some points.

1. When a node in a query plan uses pipelining, it means that it produces its output one relation at a time, which is then immediately consumed by its parent in the query tree. As a result, there is no need to materialize the intermediate result relation (which is an expensive operation).

2. σ(selection) and Π(projection) operations are should be done as early as possible as they reduce the size of data, which reduces resources need for next set of operations.

3 . Even if clustered indexing is introduced , 2nd evaluation stratergy(left deep evaluation plan) is better than the 1st .