X. Query Optimization

Lecture Topics

- Representing Plans Using Relational Algebra
- Transformation Rules
- Statistics
- Estimation

Query Optimization Overview

Generally, there are many possible access plans for processing a given query.

The costs of these plans may differ substantially.

The optimizer must try to pick a reasonable plan.

To illustrate, we will:

- express queries in the relational algebra
- show how these expressions can be transformed
- compare the costs of the original and transformed expressions

Using the Relational Algebra

Notation:

- $\pi_{\text{attribute\_list}}(R)$: project attributes from R
- $\sigma_{\text{condition}}(R)$: select tuples from R
- $R_1 \bowtie R_2$: natural join of R_1 and R_2

select Credit, Description
from Student S, Register R, Course C
where S.Studnum = R.Studnum
and R.Course = C.Course
and Surname = 'Smith'

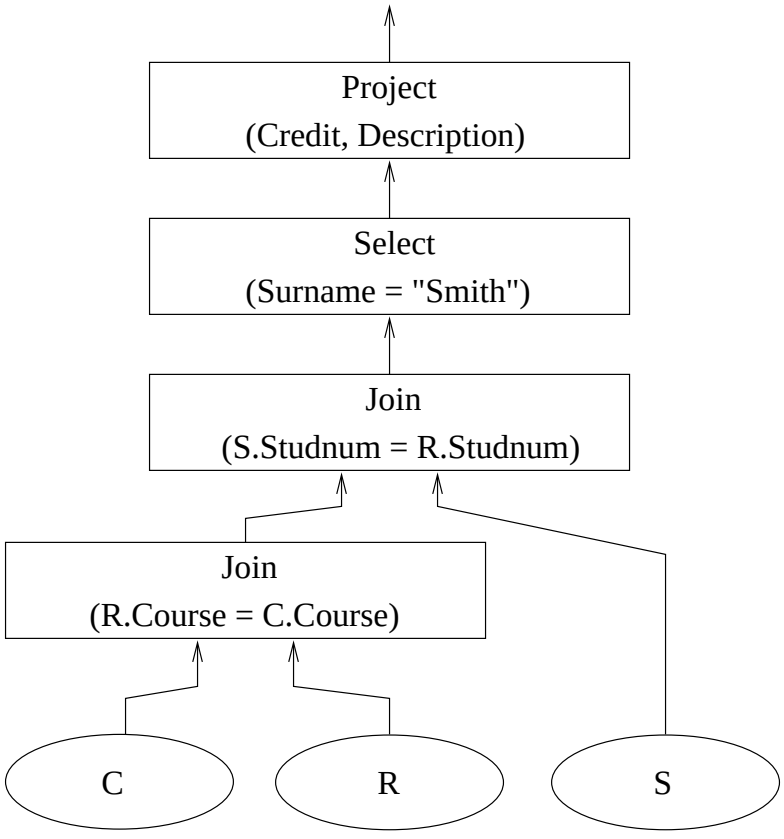
becomes:

$$\pi_{\text{Credit,Description}}(\sigma_{\text{Surname}='\text{Smith}'}(S \bowtie (R \bowtie C)))$$

Relational Algebra (cont.)

$$\pi_{\mathsf{Credit,Description}}(\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S \bowtie (R \bowtie C)))$$

is the same as:



Early Selection

The relation $S \bowtie R \bowtie C$ has one tuple for every course in which each student is registered. We only care about Smith's courses, and we can select these tuples from the Student relation before performing the join.

becomes

$$\pi_{\mathsf{Credit,Description}}((\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S)) \bowtie (R \bowtie C))$$

In general:

$$\sigma_{R1.A=x}(R1 \bowtie R2) \equiv (\sigma_{R1.A=x}(R1) \bowtie R2)$$

general rule: perform selections early

Conjunctive Selection Conditions

If a selection condition is not simple, the previous transformation may not apply directly.

select *
from Section S, Class C
where S.Course = C.Course
and S.Section = C.Section
and S.Empnum = 111
and C.Day = 'Tuesday'

becomes

$$\sigma_{\text{S.Empnum}=111 \wedge \text{C.Day}='\text{Tuesday}'}(S \bowtie C)$$

which can be transformed to

$$\sigma_{\text{S.Empnum}=111}(\sigma_{\text{C.Day}='\text{Tuesday}'}(S \bowtie C))$$

Conjunctive Selection Conditions (cont.)

The early selection rule may now be applied to produce:

$$\sigma_{\text{S.Empnum}=111}(S \bowtie \sigma_{\text{C.Day}='\text{Tuesday}'}(C))$$

and again to produce

$$(\sigma_{\text{S.Empnum}=111}(S)) \bowtie (\sigma_{\text{C.Day}='\text{Tuesday}'}(C))$$

In general:

$$\sigma_{R1.A1=c1 \wedge R2.A2=c2}(R) \equiv \sigma_{R1.A1=c1}(\sigma_{R2.A2=c2}(R))$$

$$\sigma_{R1.A1=c1 \wedge R2.A2=c2}(R) \equiv \sigma_{R2.A2=c2}(\sigma_{R1.A1=c1}(R))$$

Early Projection

Projection will not reduce the number of tuples in a relation, but it can reduce the size of each tuple by eliminating unnecessary attributes. However, we cannot simply transform this:

$$\pi_{\mathsf{Credit,Description}}(\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S \bowtie (R \bowtie C)))$$

into this:

$$\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S \bowtie (R \bowtie \pi_{\mathsf{Credit,Description}}(C)))$$

In general, we may use projection to eliminate any attributes that will not be used in the result, and that will not be used in subsequent joins or selections.

general rule: perform projection as early as possible

Join Order

Natural joins are associative

$$(R_1 \bowtie R_2) \bowtie R_3 \equiv R_1 \bowtie (R_2 \bowtie R_3)$$

and commutative

$$R_1 \bowtie R_2 \equiv R_2 \bowtie R_1$$

The join order may have an impact on the cost of a query access plan. Consider our modified plan:

$$\pi_{\mathsf{Credit,Description}}(\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S) \bowtie (R \bowtie C))$$

We could do this:

$$\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S) \bowtie (R \bowtie C)$$

or this:

$$(\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S) \bowtie R) \bowtie C$$

or this:

$$(\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S) \bowtie C) \bowtie R$$

Join Order Example

Assume that $|S| = 1000$, $|R| = 5000$, and $|C| = 500$. On average, each student is registered for five courses.

If we do this

$$\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S) \bowtie (R \bowtie C))$$

we must produce $R \bowtie C$, which has one tuple for each course registration (by any student), i.e., 5000 tuples.

Join Order Example (cont.)

If we do this

$$(\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S) \bowtie R) \bowtie C$$

we produce an intermediate relation which has one tuple for each course registration by a student named Smith. If there are only a few Smith's among the 1,000 students (say there are 10), this relation will contain about 50 tuples.

Join Order Example (cont.)

If we do this

$$(\sigma_{\mathsf{Surname}='\mathsf{Smith}'}(S) \bowtie C) \bowtie R$$

we see that S and C have no common attributes. The intermediate relation will be the cross product of the two relations. Assuming 10 Smiths, this will have 5000 tuples.

general rule: perform the most selective joins first

Selectivity and Join Size Estimation

Because

- the cost of an operation depends on the size of its input, and
- the output of one operation often becomes the input of another, and
- the order of execution of operations can be controlled

database systems need some way to estimate the size of the output of an operation, such as a selection, or a join.

These estimates are often made using (often unrealistic) assumptions of **uniformity** and **independence**.

Uniformity : all possible values of attribute A are equally likely to occur in a relation

Independence : the likelihood that a tuple will have $A_1 = a_1$ does not depend on what value it has for attribute A_2.

For example, in the Student relation of the student database there are surname and birthdate attributes. The DBMS might assume that the surname "Smith" is no more likely to occur than any other surname, and that a student named "Smith" is no more likely than a student named "Jones" to have birthdate in May.

Statistics

Database systems often maintain statistics about the relations in the database.

The purpose of these statistics is to aid in estimating the size of the output of a select operation or a join operation.

Assume that R is a relation and that A is an attribute of R. Common statistics include:

- $|R|$: the cardinality of R, i.e., the number of tuples in R
- $min(R, A)$: the minimum value for A in R
- $max(R, A)$: the maximum value for A in R
- $distinct(R, A)$: the number of distinct values of A that occur in R

As the database is modified, statistics need to be updated. Unlike indices, statistics are not usually updated automatically. Instead, the DBA controls when statistics should be updated.

Selectivity Estimation

The selectivity of a selection $\sigma_{\mathsf{condition}}(R)$ is defined as:

$$\mathsf{sel}(\sigma_{\mathsf{condition}}(R)) = \frac{|\sigma_{\mathsf{condition}}(R)|}{|R|}$$

If the DBMS has no other information, it will estimate selectivity using simple rules based on its statistics. For example:

$$\mathsf{sel}(\sigma_{A=c}(R)) \approx \frac{1}{distinct(R, A)}$$

$$\mathsf{sel}(\sigma_{A\leq c}(R)) \approx \frac{c - min(R, A)}{max(R, A) - min(R, A)}$$

$$\mathsf{sel}(\sigma_{A\geq c}(R)) \approx \frac{max(R, A) - c}{max(R, A) - min(R, A)}$$

$$\mathsf{sel}(\sigma_{\mathsf{cond1}\wedge\mathsf{cond1}}(R)) \approx \mathsf{sel}(\sigma_{\mathsf{cond1}}(R))\mathsf{sel}(\sigma_{\mathsf{cond2}}(R))$$

Join Size Estimation

The number of tuples in the result of a join depends on the number of values that match in the join attributes of the two tables. Some examples:

- Suppose we join Register (R) and Student (S) . Their common attribute is Studnum, which is the key of Student. We may estimate the join size as:

$$|R \bowtie S| = |S|\frac{|R|}{|S|} = |R|$$

  May joins are foreign key joins, like this one.

- Suppose we join Student (S) and Employee (E) using Student.Surname and Employee.Empname as the join attributes. We might estimate the join size as:

$$|S \bowtie E| \approx |S|\frac{|E|}{distinct(E, \mathsf{Empname})}$$

  or as

$$|S \bowtie E| \approx |E|\frac{|S|}{distinct(S, \mathsf{Surname})}$$

  In practice, we would take the minimum of these two values.

Summary

Query plans represented in the relational algebra may be transformed using simple transformation rules.

Using statistics, the details (e.g., selection method) of a plan may be determined, and its cost may be estimated.

The optimizer selects a low-cost plan for execution.