Fall, 2005 CIS 550

Database and Information Systems

Homework 7 Solutions

Problem 1 [24 points]: Consider a relational algebra expression of the form $\sigma_c(\pi_l(R \bowtie S))$. Suppose that the equivalent expression with selections and projections pushed as much as possible, taking into account only relational algebra equivalences, is in one of the following forms. In each case list the attributes of R and S, list an original relational algebra expression – filling in selection conditions and projection lists (instead of $c, l, c1, l1$, etc.) – and show the relational algebra expresssion in the form shown below.

Example of a similar problem:
Given relations: $R(a, b), S(b, c)$
Suppose we have an original expression of the form $\sigma_c(R \bowtie S)$, for instance $\sigma_{b<5}(R \bowtie S)$.
A final expression of the form $\sigma_{c1}(R) \bowtie \sigma_{c2}(S)$ would be $\sigma_{b<5}(R) \bowtie \sigma_{b<5}(S)$.

1. *Equivalent maximally pushed form*: $\sigma_c(\pi_{l1}(\pi_{l2}(R) \bowtie (S)))$.

   Relations: $R(a, b, c), S(c, d)$
   Original expression: $\sigma_{(a=1)\wedge(d=2)}(\pi_{a,d}(R \bowtie S))$.
   Maximally pushed form: $\sigma_{(a=1)\wedge(d=2)}(\pi_{a,d}(\pi_{a,b}(R) \bowtie (S)))$.

2. *Equivalent maximally pushed form*: $\pi_l(\sigma_{c1}(\pi_{l1}(\pi_{l2}(\sigma_{c2}(R)) \bowtie S)))$.

   Note that in this case, the maximally pushed form only makes sense if we slightly modify the original expression to have a projection *after* the selection.
   Relations: $R(a, b), S(b, c, d)$
   Original expression: $\pi_d(\sigma_{(c<d)\wedge(b=1)}(R \bowtie S))$.
   Maximally pushed form: $\pi_d(\sigma_{c<d}(\pi_{c,d}(\pi_b(\sigma_{b=1}(R)) \bowtie S)))$.

Problem 2 [26 points]: Using Oracle on eniac (the `sql` command), we will have you specify query plans using the TPC-H benchmark data set of Homework 6. To do this assignment, you will need to repeat each query 5 times. Time it by using the Oracle command "set timing on" and reading the elapsed time.

Start with the query:

```
select count(*)
from zives.lineitem, zives.orders
where l_orderkey = o_orderkey
and o_totalprice < 1501
```

1. What index or indices would be most useful on the lineitem and orders tables?

   Certainly, an index on the o_orderkey attribute would be highly useful. Possibly useful might be one on o_totalprice, but that depends on how many answers have a value less than 1501.

2. As with before, we are going to use Oracle's *hints* to tell the optimizer how to run the query. Time the query:

   ```
   select /*+ USE_HASH(l o) */ count(*)
   from zives.lineitem l, zives.orders o
   where l_orderkey = o_orderkey
   and o_totalprice < 1501
   ```

   Average time is 0.2 sec.

3. Replace "USE_HASH" with "USE_MERGE" and repeat.

   Average time is 26 sec.

4. Replace with "USE_NL" and repeat.

   Average time is 0.06 sec.

5. Was there any substantive difference? Which plan or plans seemed to perform best? Explain why.

   Yes, there is substantive difference. The nested loops join is the only one that makes use of the (default) index on o_orderkey, which is a clustered index; it's the fastest.