Processing Joins

Want to join relations r and s

nested loop join:

For each tuple t_1 in r

  For each tuple t_2 in s

    if test(t_1, t_2)

    then ~~output~~ ~~t~~ to output (t_1 join t_2)

n_r = # of tuples in r $\quad$ b_r = # of blocks in r

n_s = " in s $\quad$ b_s = # of block in s

# of block i/os = $n_r * b_s + b_r$

$n_r = 10000$, $b_r = 500$

$n_s = 50000$ $\quad$ $b_s = 2500$

# of block i/os = $10^4 \times 2500 + 500$

Nested block loop join:

For each block B_1 of r

  For each block B_2 of s

    For each tuple t_1 in B_1

      For each tuple t_2 in B_2

        if test(t_1, t_2)

          output (join of t_1 and t_2)

# of block i/o s = $b_r * b_s + b_r$

For the previous example, this value = $500 * 2500 + 500$

Suppose that we have $M \geq 2$ memory blocks available for input relations

We read $M-1$ blocks of outer relation r into memory

and 1 block of inner relation s into memory.

# of block i/o s = $\left\lceil \frac{b_r}{M-1} \right\rceil * b_s + b_r$

For the previous example, $M=4$,

# of block i/os = $\left\lceil \frac{500}{3} \right\rceil * 2500 + 500$

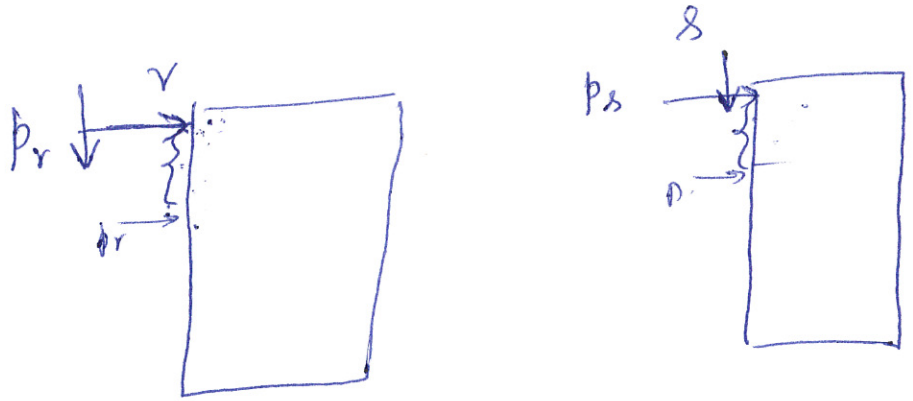
# of seeks = $2 * \left\lceil \frac{b_r}{M-1} \right\rceil$

Merge join:

~~A Want to~~

Want to evaluate $r \bowtie s$

Assume that r and s are sorted on the common attribute A.



~~t_r~~ p_r = 1st tuple in r , p_s = 1st tuple in s

t_r = tuple pointed to by p_r, t_s tuple pointed to by p_s.

while $p_r \neq$ null and $p_s \neq$ null

```
if   t_r[A] < t_s[A]
     advance p_r

else if t_s[A] < t_r[A]
     advance p_s

else    % t_r[A] = t_s[A]
{ read all successive tuples of r
  having the same value for attribute A
  into memory X_r while advancing
  pointer p_r ;
  Similarly read all successive tuples of s
  having same value for attribute A
  into memory X_s while advancing p_s ;
  join tuples in X_r with tuples in X_s . }
```

Cost of Merge join

$$\# \text{ block } i/o\text{s} = b_r + b_s$$

$$\# \text{ of seeks} = \frac{b_r}{M_r} + \frac{b_s}{M_s}$$

M_r = # of memory blocks for r

M_s = " " for s

Hash Join:

Want to Compute $r \bowtie s$ where A is the common attribute.

Choose a hash function h : Set of keyvalues $\rightarrow \{0, 1, \ldots, m-1\}$

For each tuple t in r
    $i := h(t[A])$
    Write t into temporary relation R_i.

For each tuple t in s
    $i := h(t[A])$
    Write t into temporary relation S_i.

For each $i = 0$ to $m-1$
    read R_i into memory
    and build an in memory index on R_i, on the attribute A.
    } probe relation

    For each tuple t in S_i ——— probe relation
        probe the index on R_i
        and join t with tuples in R_i having same attribute value for A

Cost analysis for hash join:

block i/o s : $3(b_r + b_s) + 4m$

of seeks : $2(b_r + b_s) + 4m$.

Index join:

Want to Cmpute $r \bowtie s$

A is the Common attribute.

Assume there is an index on s, based on attribute A.

For each block B of r

  For each tuple t_i in B

    { Use the index on s, to retrieve
    { all tuples in s whose A attribute
    { value = $t_i[A]$ into the set X

    join t_i with all tuples in X

# of block i/o s = $b_s + n_s * (L_s + 1)$

L_s = # of levels in the index of s

For one example, # of block i/o s = $500 + 10^4 * 3$

# of seeks =

Cost Analysis of index join.

of block i/o's = $b_r + b_s$

# of seeks = $\frac{b_r}{M_r} + \frac{b_s}{M_s}$

where M_r = # of memory blocks allocated to r

M_s = " " to s.

For the previous example

# of block i/o's = 500 + 2500 = 3000

# of seeks = 50 + 250 if $M_r = M_s = 10$

If r and s are not sorted

then we need to sort them:

Cost of Sorting:

Cost of Sorting r:

# of block i/o's = $b_r \left(2 \left\lceil \log_{M-1} (b_r/M) \right\rceil + 1\right)$