Quiz Sheet No. 2 for *Architecture and Implementation of Database Systems*
Prof. Rudolf Bayer, Ph. D.
Institut für Informatik
SS 2003

Exercises for **Chapter 2.4: Prefix B-Trees**

1. Consider the following listing of paths from a Windows file system.
   a) Determine the shortest separators (assuming lexicographic ordering) between these keys. Strike out the parts not needed.
   b) Determine the partial separators between those keys, which are printed in bold.

Answer:

a)	b)
notebayer10\c$\WINDOWS\system32\SNMPAPI.DLL	
notebayer10\c$\WINDOWS\system32\SNMPS ~~*NAP.DLL*~~	
notebayer10\c$\WINDOWS\system32\SO ~~*FTPUB.DLL*~~	
notebayer10\c$\WINDOWS\system32\SOL ~~*.EXE*~~	**OL**
notebayer10\c$\WINDOWS\system32\SOR ~~*T.EXE*~~	
notebayer10\c$\WINDOWS\system32\SP ~~*IDER.EXE*~~	
notebayer10\c$\WINDOWS\system32\SPN ~~***IKE.DLL***~~	**PN**
notebayer10\c$\WINDOWS\system32\SPO ~~*OLSS.DLL*~~	
notebayer10\c$\WINDOWS\system32\SPOO ~~*LSV.EXE*~~	
notebayer10\c$\WINDOWS\system32\SR ~~***CLIENT.DLL***~~	**R**
notebayer10\c$\WINDOWS\system32\SRR ~~*STR.DLL*~~	
notebayer10\c$\WINDOWS\system32\SRS ~~*VC.DLL*~~	
notebayer10\c$\WINDOWS\system32\SRV ~~*SVC.DLL*~~	
notebayer10\c$\WINDOWS\system32\SS3 ~~*DFO.SCR*~~	
notebayer10\c$\WINDOWS\system32\SSB ~~***EZIER.SCR***~~	**SB**
notebayer10\c$\WINDOWS\system32\SSD ~~*PAPI.DLL*~~	
notebayer10\c$\WINDOWS\system32\SSDP ~~*SRV.DLL*~~	
notebayer10\c$\WINDOWS\system32\SSF ~~***LWBOX.SCR***~~	**F**
notebayer10\c$\WINDOWS\system32\SSM ~~*ARQUE.SCR*~~	
notebayer10\c$\WINDOWS\system32\SSMY ~~*PICS.SCR*~~	
notebayer10\c$\WINDOWS\system32\SSMYS ~~*T.SCR*~~	

Exercises for **Chapter 3: Query Optimization**

2. We are adding an new relation to the database schema on **Part** and **Supplier** in the script (cf. Chapter 3.3, page 5):

   create table Description
   (P# **integer**,
   Name **string**,
   Material **string**,
   Weight **real**) **key is** P#

   a) Formulate a query in SQL that retrieves the Name, QTY and PRICE of parts weighing less than 10 kg (selectivity: 0.3) that are supplied by companies in Munich (8 suppliers in Munich) and cost less than 100 $ (selectivity: 0.7).

Answer:

```
SELECT   D.Name,  P.QTY,  P.PRICE
FROM     Description AS D,  Part AS P,  Supplier AS S
WHERE    D.Weight < 10.0 AND S.S-City = 'Munich' AND P.PRICE < 100.0
   AND   D.P# = P.P#      AND   S.S# = P.S#
```

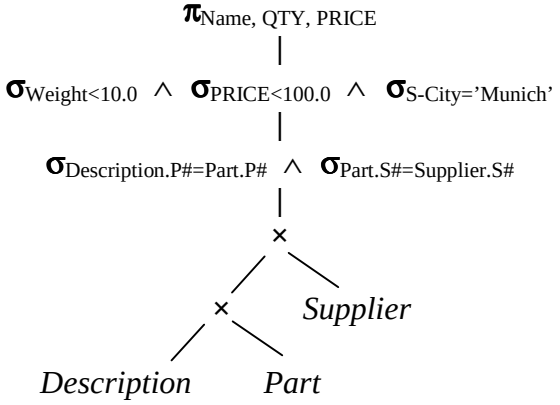
   b) Reformulate the same query using relational algebra.

Answer:

$\pi_{\text{Name, QTY, PRICE}}$ (
$\sigma_{\text{Weight<10.0}}(\textit{Description}) \bowtie_{\text{P\#}} \sigma_{\text{PRICE<100.0}}(\textit{Part}) \bowtie_{\text{S\#}} \sigma_{\text{S-City='Munich'}}(\textit{Supplier})$
)

   c) Sketch an operator tree for this query. Split the predicate in the WHERE clause into join and selection predicates which can be pushed down in the operator tree.

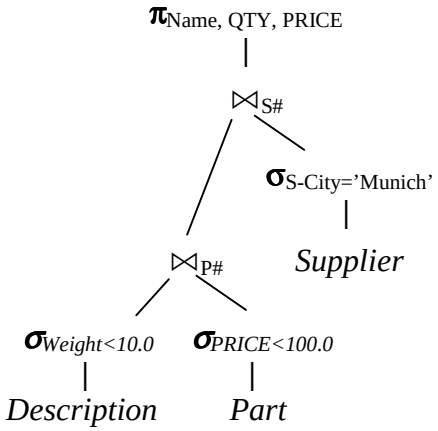
Answer:



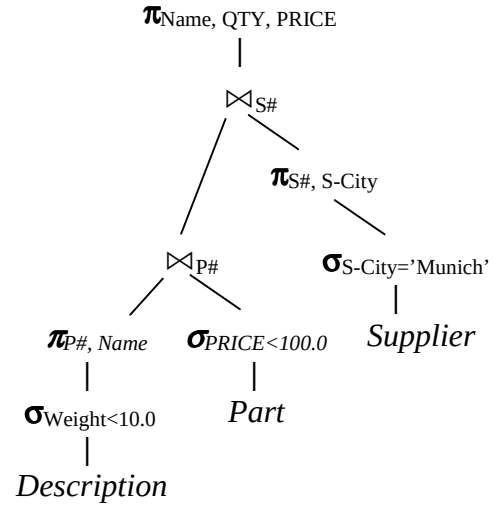
d) Optimize the operator tree.

Answer:

Push down selections:



Consider projections:



e) What are the most costly operations? Assume that:
|Part| = 90,000
|Supplier| = 150
|Description| = 30,000

Answer: [assuming that there are no indexes at all]

selections:

table scan necessary for selections on each relation *Part* (90,000 tuples), *Supplier* (150 tuples) and *Description* (30,000 tuples);
Costs: O(n) for each table scan

- selectivity of "Weight < 10.0" (on *Description*) is 0.3
  $\Rightarrow$ 30,000 * 0.3 = 9,000 tuples are selected from *Description*
- 8 tuples (suppliers in Munich) are selected from *Supplier*
- selectivity of "PRICE < 100.0" (on *Part*) is 0.7
  $\Rightarrow$ 90,000 * 0.7 = 63,000 tuples are selected from *Part*

joins [assuming execution order ((*Description* $\bowtie$ *Part*) $\bowtie$ *Supplier*)]:

- *Description* $\bowtie$ *Part*:
  cross product of 9,000 $\times$ 63,000 tuples,
  Costs: O(9,000 * 63,000) for nested loop join

  Suppose a selectivity of 0.02 for this join operation
  $\Rightarrow$ 9,000 * 63,000 * 0.02 = 11,340,000 tuples as an intermediate result
- ((*Description* $\bowtie$ *Part*) $\bowtie$ *Supplier*):
  cross product of 11,340,000 $\times$ 8 tuples,
  Costs: O(11,340,000 * 8) for nested loop join

  Suppose a selectivity of 0.1 for this join operation
  $\Rightarrow$ 11,340,000 * 8 * 0.1 = 9,072,000 tuples in the final result

f) How can secondary indexes be used to speed up query execution? Suggest appropriate indexes.

Answer:

selections:

assuming secondary indexes on the selection attributes (Weight, S-City, PRICE) the costs are reduced to O(result size), where the result size is computed as shown in the answer to problem *2e)*:

- *Costs for* $\sigma_{\text{Weight<10.0}}$(*Description*): O(9,000)
- *Costs for* $\sigma_{\text{PRICE<100.0}}$(*Part*): O(63,000)
- *Costs for* $\sigma_{\text{S-City='Munich'}}$(*Supplier*): O(8)

joins [assuming execution order ((*Description* ⋈ *Part*) ⋈ *Supplier*)]:

assuming primary indexes on all key attributes (which are also join attributes):

- *Description* ⋈ *Part*:
  Costs: O(9,000 + 63,000) for merge join
  (suppose 11,340,000 tuples for the intermediate result)
- ((*Description* ⋈ *Part*) ⋈ *Supplier*):
  Costs: O(11,340,000 + 8) for merge join
  (suppose 9,072,000 tuples in the final result)