

Comp 421 – Assignment 3

	Products	Stores	Store prices
Tuples	20 000	1000	4 000 000
Pages	600	80	40 000

Question 1

- 20000 tuples
- 600 data pages
- Prefix: 20 bytes
- Full: 30 bytes
- 200 different types
- 50 producers
- Rid has 10 bytes
- Pointer has 6 bytes
- Leaf pages are filled about 70%
- Index page has 4000 Bytes

- 1) We have $200 * 50 = 10000$ possible different values. Then as its uniformly distributed we have 10000 data entries
The number of rids per data entry is $\left(\frac{\text{number of tuples}}{\text{diff values}}\right) = \frac{20000}{10000} = 2$
The average length of a data entry is $\text{size of key} + \text{nb of rids} * \text{size(rids)} = 30 * 2 + 2 * 10 = 80$
- 2) The size of an index entry is $20 + 20 + 6 = 46$. Then the average number of index entry per intermediate page is $\frac{4000}{46} = 86$ and the average number of data entry per page is $\frac{0.7 * 40000}{80} = 35$. If we have a tree of height 2 then we can cover at most $86 * 86 = 7396$ different cases then we need a tree of height 3 as $86 * 86 * 86 = 636056 \gg 10000$.
Number of leaf pages is $\frac{10000}{35} = 285$ and as $\frac{285}{86} > 3$ there is 4 intermediate pages.

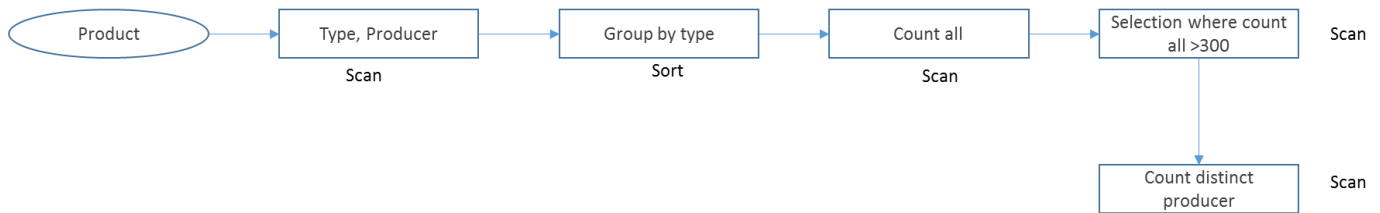
Question 2

Question 2.1

The first execution plan is the following



We have 600 pages and as we are searching on arbitrary attribute we have a cost of 600



The cost is the same here in number of pages. However as we filter with the count > 300 the number of types for which we need to count the distinct producer is considerably reduced and as DISTINCT is a costly operation it will result into an improvement.

Question 2.2

We have 40 000 data pages and 4M entries so there is 100 entries in each pages.

a)

- i) As the pid is not sorted we need to go through all possible values. And when we have a corresponding row we can just check the inStock is inferior to Y and return this row or not. Then the cost is 40 000. As we don't know if the table is sorted using $X=200$ and $Y=10$ will be the same cost
- ii) As we have 20 000 products and 4 million store prices we can suppose we will have in average a match of 200 tuples which means that we will in the worst case get 200 pages. Then the *cost* = #of leaf pages + #of data pages = $1 + 200 = 201$. This will be the same for $X=200$ and $Y=10$
- iii) As inStock is uniformly distributed between 1 and 500 we have on average a result of $\frac{4M \cdot Y}{500} = 8000Y$ tuples matching. But as we are using an unclustered index the results are spread across all pages. (Ignoring if $Y < 5$) the cost will be 40 000 pages + Y leafs page. In the case were $Y=10$ we have a cost of 40010.
- iv) With both index we will get Y index pages and $\frac{200 \cdot Y}{500} = 0.4Y$ data pages to check. Then the cost is $Y + 0.4Y$
So in the case $X=200$ and $Y=10$ we will have *cost* = $10 + 0.4 \cdot 10 = 14$

b) Changing to a clustered index on pid will not change the cost (Still going to be 2). However changing to a clustered index on inStock will considerably improve the cost. The matching tuples will be clustered into a few adjacent data page so we will access only those few data page.

Question 3

We have

- 20 000 products on 600 pages
 - 4 000 000 store prices on 40 000 pages
 - 1000 Stores on 80 pages
1. We will get an output of 4 000 000 tuples as the outer join will get all possible storePrices and as all product are sold somewhere.
 2.
 - a) *cost* = nb of product pages + (nb of products * cost of getting store prices)
cost = $600 + 20000 \cdot 2 = 40\,600$
 - b) *cost* = nb of storeprices pages + (nb of storeprices * cost of getting product)
cost = $40\,000 + 4M \cdot 2 = 8\,040\,000$

$$c) \text{ cost} = \text{nb of product pages} + \frac{\text{nb of product pages} * \text{nb of storePrices pages}}{B-2}$$

$$\text{cost} = 600 + \frac{600 * 40\,000}{98} = 245498$$

$$d) \text{ cost} = \text{nb of storePrices pages} + \frac{\text{nb of storePrices pages} * \text{nb of product pages}}{B-2}$$

$$\text{cost} = 40\,000 + \frac{40\,000 * 600}{98} = 284898$$

$$e) \text{ cost} = 3 * 40\,000 + 3 * 600 = 121\,800$$

Question 4

$$\pi_{pid;pname;storeId}(\sigma_{\text{addresscontainsMontreal} \wedge \text{sellingprices} * \text{inStock} < 100} (\text{Products} \times \text{Stores}) \bowtie \text{StorePrices})$$

First we are going to select only store where the address contains Montréal before the join.

$$\pi_{pid;pname;storeId}(\sigma_{\text{sellingprices} * \text{inStock} < 100} ((\text{Products} \times \sigma_{\text{addresscontainsMontreal}}(\text{Stores})) \bowtie \text{StorePrices}))$$

Now we are going to select only the store prices where the selling's prices * inStock < 10 before join.

$$\pi_{pid;pname;storeId}((\text{Products} \times \sigma_{\text{addresscontainsMontreal}}(\text{Stores})) \bowtie \sigma_{\text{sellingprices} * \text{inStock} < 100}(\text{StorePrices}))$$

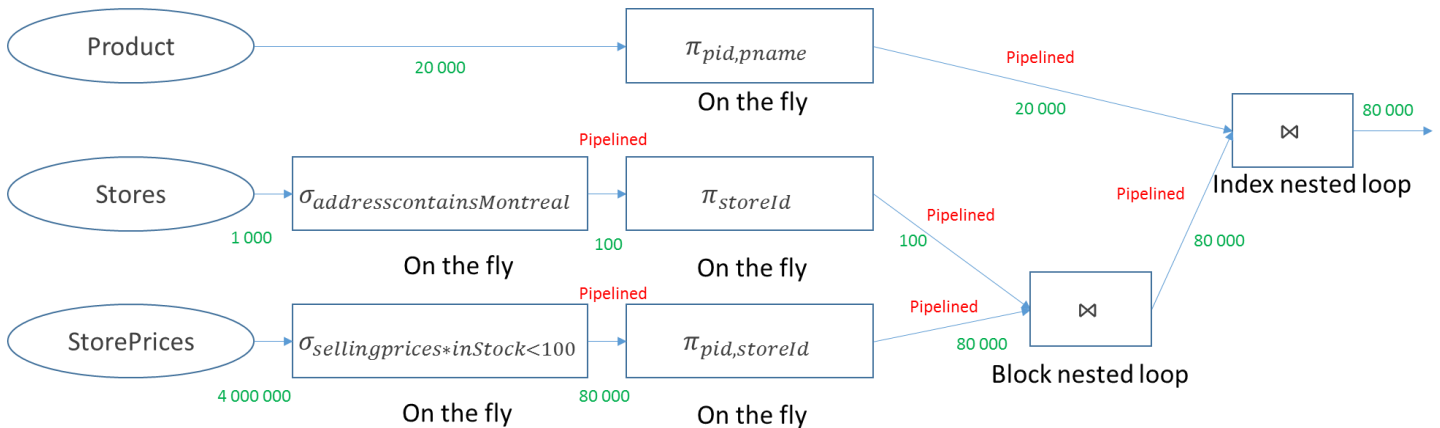
We are going to only select required column before joining.

$$\begin{aligned} & ((\pi_{pid,pname} \text{Products} \times \pi_{storeId}(\sigma_{\text{addresscontainsMontreal}}(\text{Stores}))) \\ & \bowtie \pi_{pid,storeId}(\sigma_{\text{sellingprices} * \text{inStock} < 100}(\text{StorePrices}))) \end{aligned}$$

Finally we are going to switch the cross product with a join by switching the order of joins

$$\begin{aligned} & \pi_{pid,pname} \text{Products} \\ & \bowtie \left[\pi_{storeId}(\sigma_{\text{addresscontainsMontreal}}(\text{Stores})) \right. \\ & \left. \bowtie \pi_{pid,storeId}(\sigma_{\text{sellingprices} * \text{inStock} < 100}(\text{StorePrices})) \right] \end{aligned}$$

This is the graph we get at the final state. In green are the estimated tuples flow.



Timothee Guerin
260447866