Discussion #7

Elena, Diwanshu

July 22, 2020

Logistics

- Final Project Presentation: Sign up here
  - Starts Thursday, 7/23.
  - Please be ready 5 minutes before your presentation.
  - Video of how to share ppt and YouTube videos in BBCU posted here
  - 12-13 minutes presentation followed by some questions.
- Project report/presentation/code: Due midnight, July 29.
- Final Exam on July 29, 6 am - 11:59 pm.
  - Exam guidelines posted here
- Review Session by Prof. H. Nguyen: July 27, 9:30 am - 11:30 am.

Outline

- Join Algorithms
- Query Performance

Join Operator

Algorithms for equijoin:

- nested loop join
- block nested loop join
- index nested loop join
- block index nested loop join
- sort merge join
- hash join

Example

```
SELECT *

FROM R,S

WHERE R.a = S.a
```

Nested Loop Join (1)

Algorithm:

```
for each page P_R in R
    for each page  P_s in S
        join the tuples on P_R with the tuples in P_s
```

Cost:

$I/O = M_R + M_s * M_R$

We ignore the cost of writing the output to disk!

- M_R : number of pages in R table
- M_S : number of pages in S table

Nested Loop Join (2)

- Which relation should be the outer relation in the loop?
  - The smaller of the two relations
- How many buffer frames(=pages) we need?
  - only 3 pages suffice

Block Nested Loop Join (1)

Now assume that we have B buffer frames(=pages). We need one frame for the inner table, one for the result and the rest can ALL be used for the outer table.

Algorithm:

```
for each block of B-2 pages from R
  for each page  P_s in S
     join the tuples from the block with the tuples in P_s
```

Cost:

$$I/O = M_R + M_s * \left\lceil \frac{M_R}{B-2} \right\rceil$$

- M_R : number of pages in R table
- M_S : number of pages in S table

Nested Loop Join (NLJ) vs Block NLJ

Example

- **M_R = 500 pages**
- **M_S = 1000 pages**
- **100 tuples / page**
- **B = 12**

NLJ I/O: 500 + 500*1000 = **500000**

BNLJ I/O: 500 + 1000 * (500 /(12-2)) = **50500**

The difference in I/O cost in an order of magnitude!

Index Nested Loop Join

Now assume that we have an index on the join attribute.

Algorithm:

```
for each page P_R  in R
   for each tuple r in R
      probe the index of S to retrieve any matching tuples
```

Cost:

$I/O = M_R + |R| * I$

- I is the I/O cost of searching an index and depends on the type of index and whether it is clustered or not

Block Index Nested Loop Join

Now assume that we have an index on the join attribute and B buffer frames.

Algorithm:

```
for each block of B-2 pages in R
    sort the tuples in the block
    for each tuple r in the block
        probe the index of S to retrieve any matching tuples
```

Sort Merge Join

Algorithm:

Generate sorted runs for **R** (external sorting)

Generate sorted runs for **S** (external sorting)

Merge the sorted runs for **R** and **S**

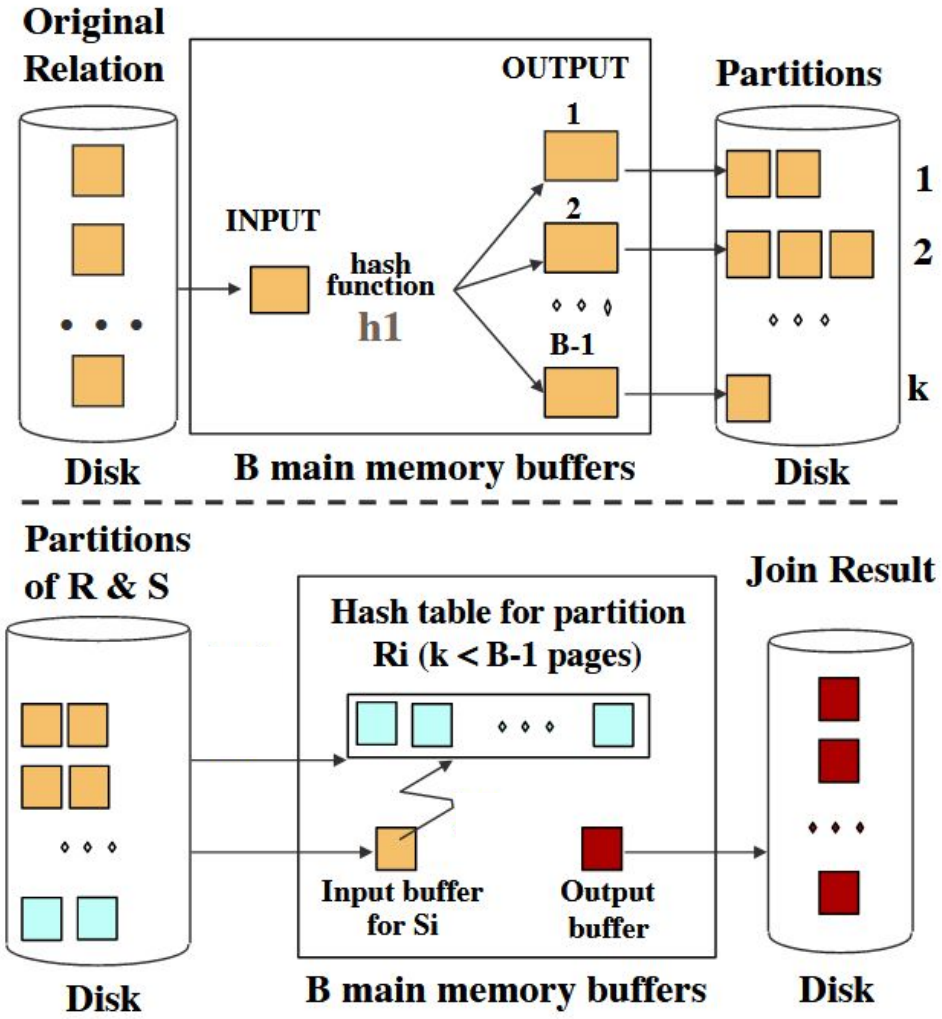
    While merging check for the join condition

Hash Join

Strat with a hash function *h* on the join attribute.

Partition phase: partition **R** and **S** into *k* partitions using *h*

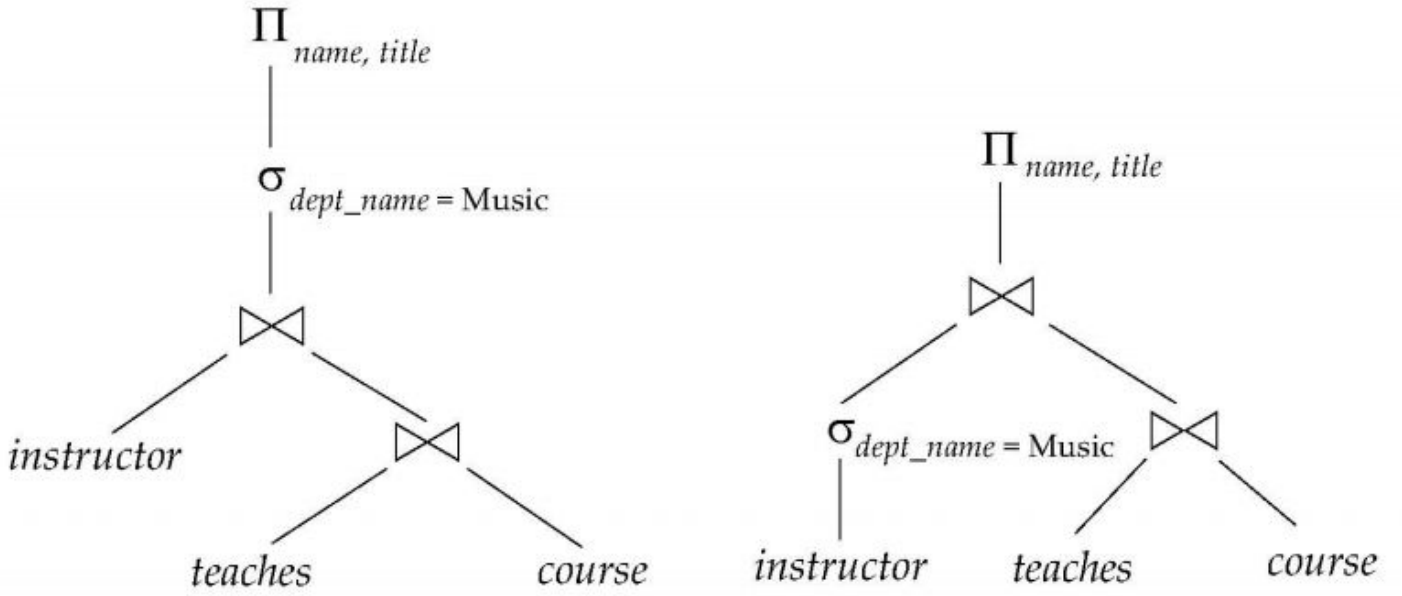
Matching phase: join each partition of **R** with the corresponding (same hash function) partition of **S** BNLJ.



Query Performance

- A single query can be executed through different algorithms or re-written in different forms and structures. Hence, the question of query optimization comes into the picture – Which of these forms or pathways is the most optimal?
- The query optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans.

Query Performance



Equivalence Rules

- An *equivalence rule* says that expressions of two forms are equivalent
  → Can replace expression of first form by second, or vice versa.

- Two relational algebra expressions are said to be *equivalent* if the two expressions generate the same set of tuples on every legal database instance.

Some Equivalence Rules

1. Conjunctive selection operations can be written as a sequence of individual selections. This is called a *sigma-cascade*.

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. Selection is commutative.

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

Some Equivalence Rules

3. All following projections can be omitted, only the first projection is required. This is called a *pi-cascade.*

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L_1}(E)$$

4. Selections on Cartesian Products can be re-written as Theta Joins.

$$\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$$

$$\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$$

Some Equivalence Rules

5. Theta Joins are commutative.

$$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$$

6. Join operations are associative. [Join Reordering]

   6a. Natural join operations are associative:

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

   6b. Theta joins are associative (θ2 involves attributes from only E2 and E3):

$$(E_1 \bowtie_{\theta 1} E_2) \bowtie_{\theta 2 \wedge \theta 3} E_3 = E_1 \bowtie_{\theta 1 \wedge \theta 3} (E_2 \bowtie_{\theta 2} E_3)$$

Some Equivalence Rules

7. Selection operation can be distributed. [Pushing Selection]

7a. When all the attributes in θ0 involve only the attributes of one of the expressions (E1) being joined.

$$\sigma_{\theta 0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_{\theta} E_2$$

7b. When θ1 involves only the attributes of E1 and θ2 involves only the attributes of E2.

$$\sigma_{\theta 1 \wedge \theta 2}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 1}(E_1)) \bowtie_{\theta} (\sigma_{\theta 2}(E_2))$$

Some Equivalence Rules

8. The projection operation distributes over the theta join operation as follows:

[Pushing Projection]

(a) if θ involves only attributes from $L_1 \cup L_2$:

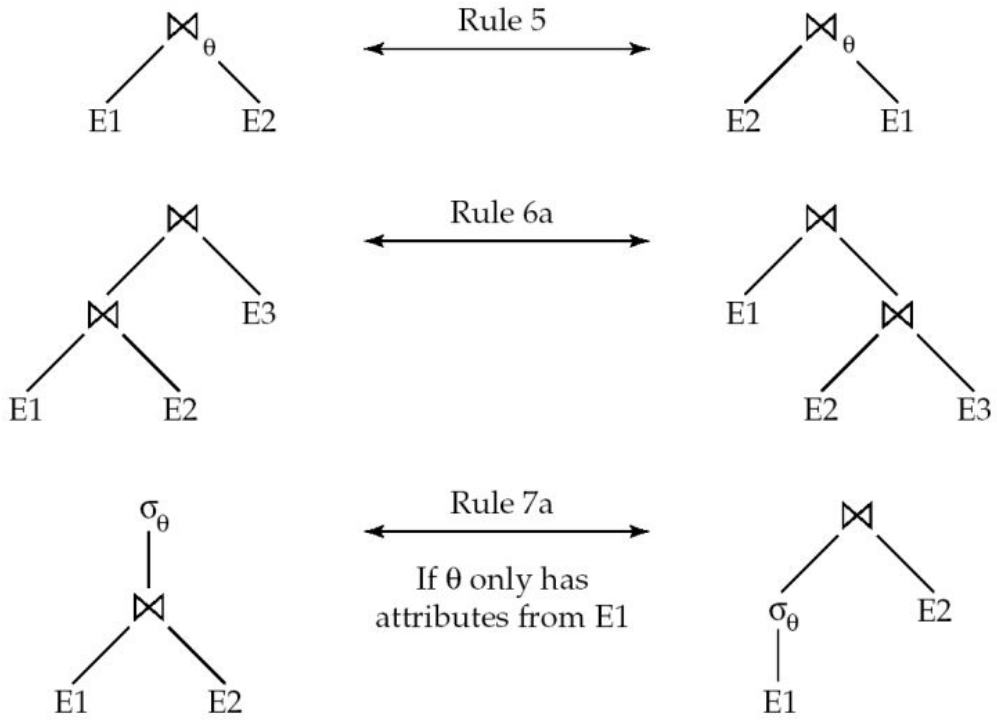
$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = (\Pi_{L_1}(E_1)) \bowtie_\theta (\Pi_{L_2}(E_2))$$

(b) Consider a join $E_1 \bowtie_\theta E_2$.

- Let L_1 and L_2 be sets of attributes from E_1 and E_2, respectively.
- Let L_3 be attributes of E_1 that are involved in join condition θ, but are not in $L_1 \cup L_2$, and
- let L_4 be attributes of E_2 that are involved in join condition θ, but are not in $L_1 \cup L_2$.

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1)) \bowtie_\theta (\Pi_{L_2 \cup L_4}(E_2)))$$

Pictorial Depiction of Equivalence Rules



True/False

Q. Consider the following database schema: R(A, B), S(A, B, C), T(B, D, E).

The following two relational algebra queries are equivalent:

$$Q_1 = \sigma_{A=1,B>2}((R \bowtie S) \bowtie T)$$
$$Q_2 = (\sigma_{A=1,B>2}(R \bowtie S)) \bowtie T$$

True/False

Q. Consider the following database schema: R(A, B), S(A, B, C), T(B, D, E).

The following two relational algebra queries are equivalent:

$$Q_1 = \sigma_{A=1,B>2}((R \bowtie S) \bowtie T)$$
$$Q_2 = (\sigma_{A=1,B>2}(R \bowtie S)) \bowtie T$$

True

True/False

Q. Consider the following database schema: R(A, B), S(A, B, C), T(B, D, E).

The following two relational algebra queries are equivalent:

$$Q_3 = \pi_E(\sigma_{D=1}(T \bowtie S))$$
$$Q_4 = \pi_B(S) \bowtie \pi_{B,E}(\sigma_{D=1}(T))$$

True/False

Q. Consider the following database schema: R(A, B), S(A, B, C), T(B, D, E).

The following two relational algebra queries are equivalent:

$$Q_3 = \pi_E(\sigma_{D=1}(T \bowtie S))$$
$$Q_4 = \pi_B(S) \bowtie \pi_{B,E}(\sigma_{D=1}(T))$$

False

Practice Question

Consider the two relations:
Company(cname, city, president)
Product(pname, maker, price)

Consider the SQL query:

```
SELECT  c.cname, c.president
FROM     Company c, Product p
WHERE   (c.cname=p.maker) AND (p.price = 100) AND
(c.city = 'Madison')
```

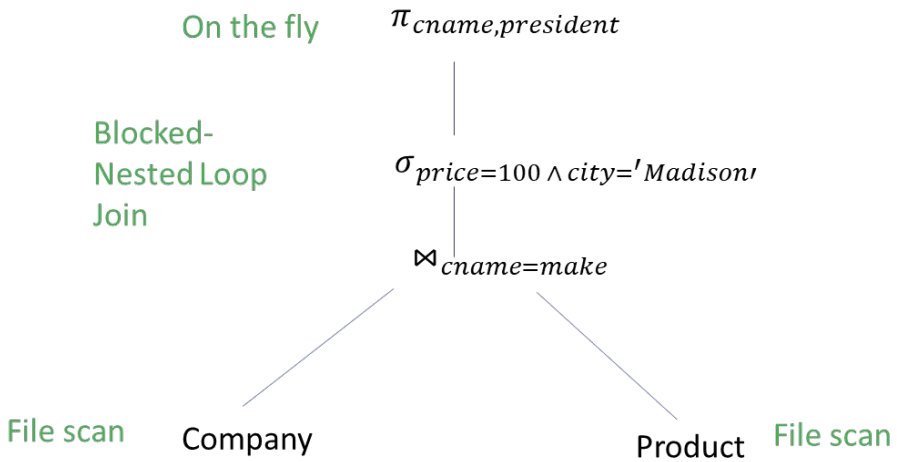
Q(a). Draw a logical query plan tree for the above SQL query. This plan tree should join Company and Product, then perform a selection on price and city, then project out cname and president.

Practice Question

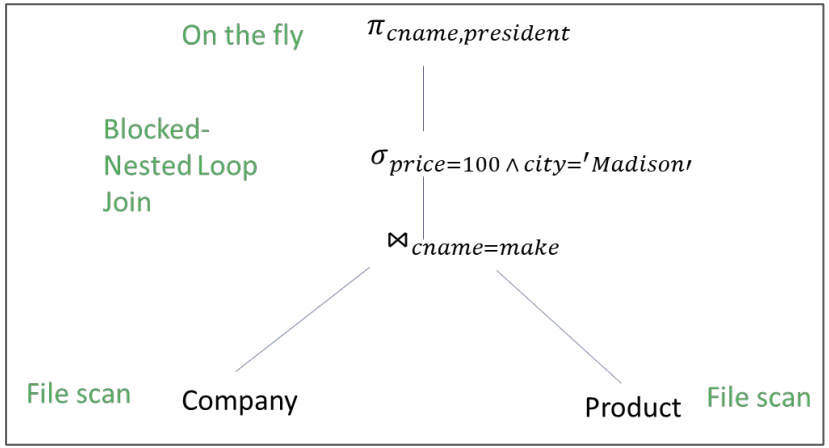
Consider the two relations:
Company(cname, city, president)
Product(pname, maker, price)

Consider the SQL query:
SELECT c.cname, c.president
FROM Company c, Product p
WHERE (c.cname=p.maker) AND (p.price = 100) AND (c.city = 'Madison')

Q(a). Draw a logical query plan tree for the above SQL query. This plan tree should join Company and Product, then perform a selection on price and city, then project out cname and president.



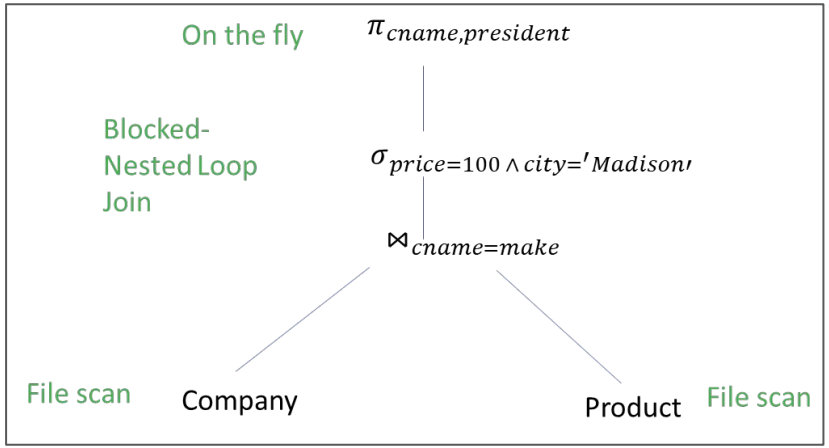
Practice Question

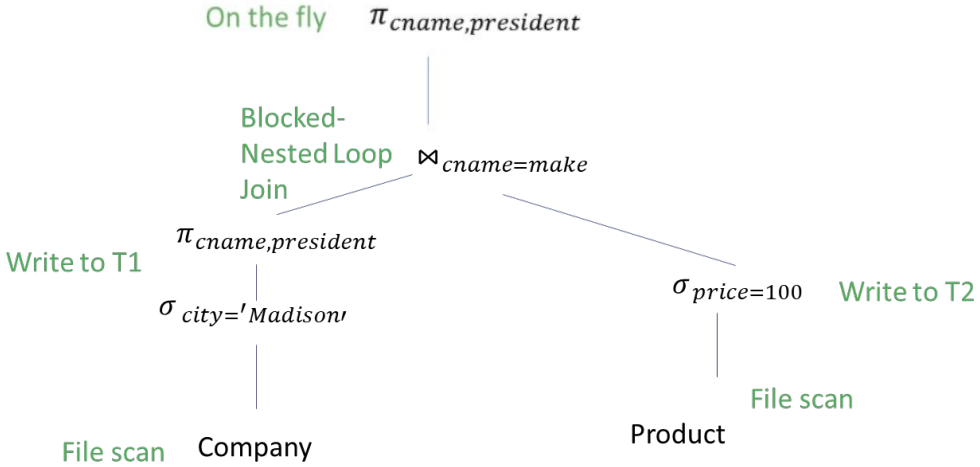


Q(b). Consider the logical query plan that you have drawn in Part a. Draw a new logical query plan that pushes selections and projections down as far as possible. Please state also the rules that you have applied.

Company(cname, city, president)
Product(pname, maker, price)

Practice Question





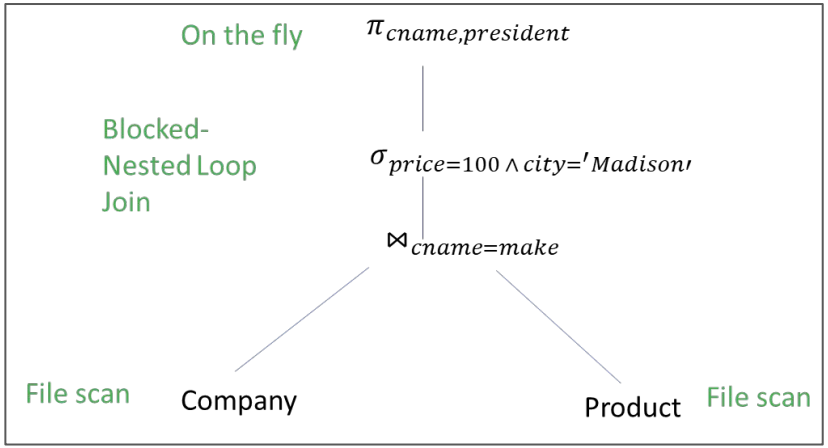
Q(b). Consider the logical query plan that you have drawn in Part a. Draw a new logical query plan that pushes selections and projections down as far as possible. Please state also the rules that you have applied.

Company(cname, city, president)
Product(pname, maker, price)

Practice Question

Q(c). Given the following information:
T(Company) = 2000 records
T(Product) = 20000 records
B(Company) = 200 pages
B(Product) = 200 pages
V(Product, price) = 2500
V(Company, city) = 40
You always used the block-nested loop join. Both relations are clustered.
M = 100 (pages)

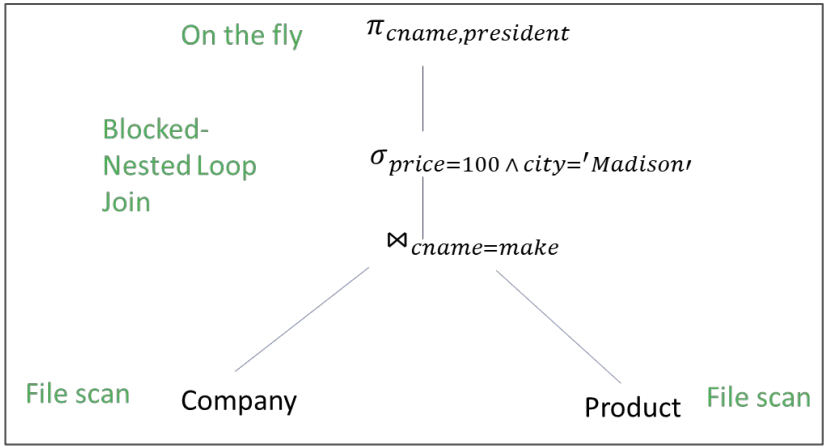
Please calculate I/O cost for each query plan in (a) and (b)



Practice Question

Q(c). Given the following information:
T(Company) = 2000 records
T(Product) = 20000 records
B(Company) = 200 pages
B(Product) = 200 pages
V(Product, price) = 2500
V(Company, city) = 40
You always used the block-nested loop join. Both relations are clustered.
M = 100 (pages)

Please calculate I/O cost for each query plan in (a) and (b)



Cost = Cost of Blocked-nested loop join = *Scan of outer table + #outer blocks*scan of inner table*

#outer blocks = ⌈#pagesofouter / blocksize⌉

Here, Scan of outer table = 200, scan of inner table = 200, #outer blocks = ⌈200/98⌉ = 3

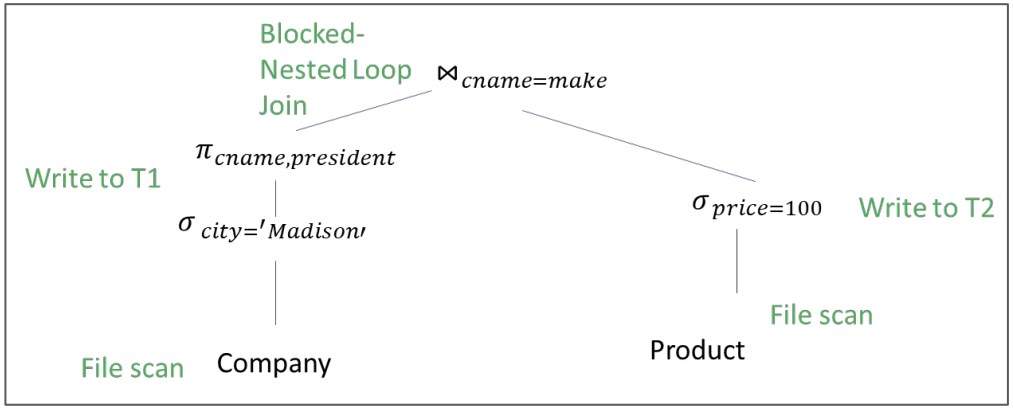
Cost: 200 + 3*200 = 800 I/O

Remember: This cost is an estimate, and not a precise number.

Practice Question

Q(c). Given the following information:
T(Company) = 2000 records
T(Product) = 20000 records
B(Company) = 200 pages
B(Product) = 200 pages
V(Product, price) = 2500
V(Company, city) = 40
You always used the block-nested loop join. Both relations are clustered.
M = 100 (pages)

Please calculate I/O cost for each query plan in (a) and (b)

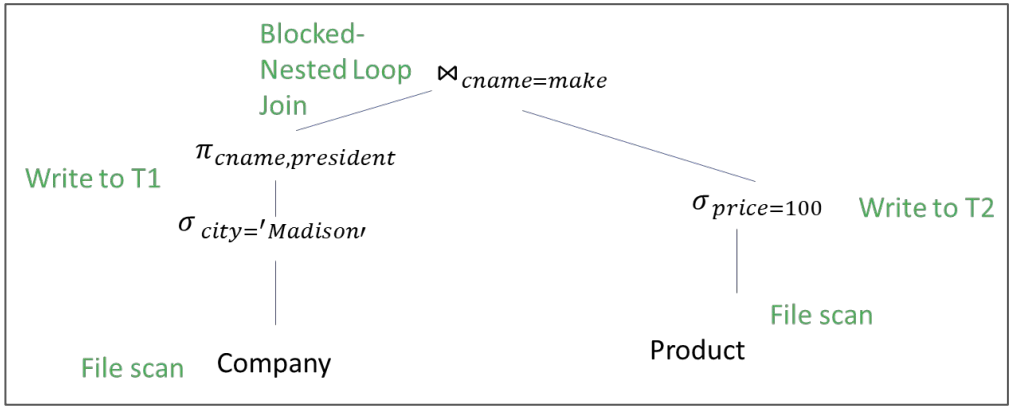


Cost = (Cost to scan company table + perform selection on city) + (Cost to scan product table + perform selection on price) + (Cost to read from T1 into memory) + (Cost to read from T2 into memory)

Practice Question

Q(c). Given the following information:
T(Company) = 2000 records
T(Product) = 20000 records
B(Company) = 200 pages
B(Product) = 200 pages
V(Product, price) = 2500
V(Company, city) = 40
You always used the block-nested loop join. Both relations are clustered.
M = 100 (pages)

Please calculate I/O cost for each query plan in (a) and (b)



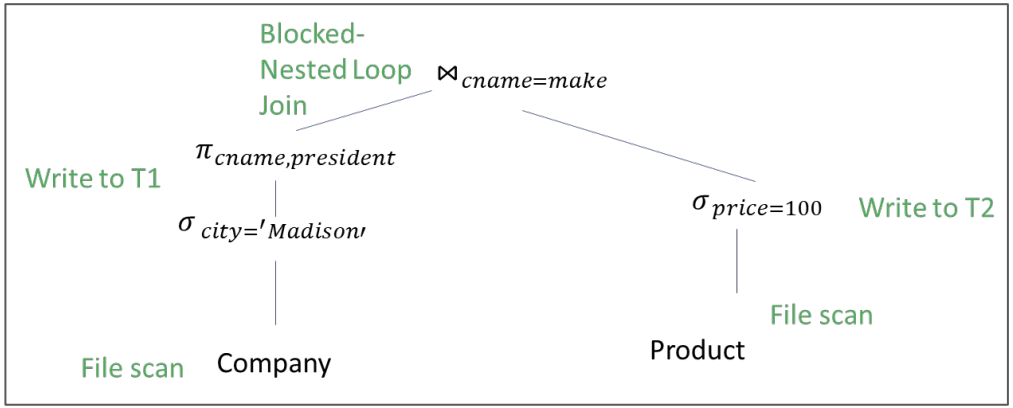
Cost = (Cost to scan company table + perform selection on city) + (Cost to scan product table + perform selection on price) + (Cost to read from T1 into memory) + (Cost to read from T2 into memory)

(Cost to scan company table + perform selection on city) = 200 + 200/40 = **200 + 5 I/O**

Practice Question

Q(c). Given the following information:
T(Company) = 2000 records
T(Product) = 20000 records
B(Company) = 200 pages
B(Product) = 200 pages
V(Product, price) = 2500
V(Company, city) = 40
You always used the block-nested loop join. Both relations are clustered.
M = 100 (pages)

Please calculate I/O cost for each query plan in (a) and (b)



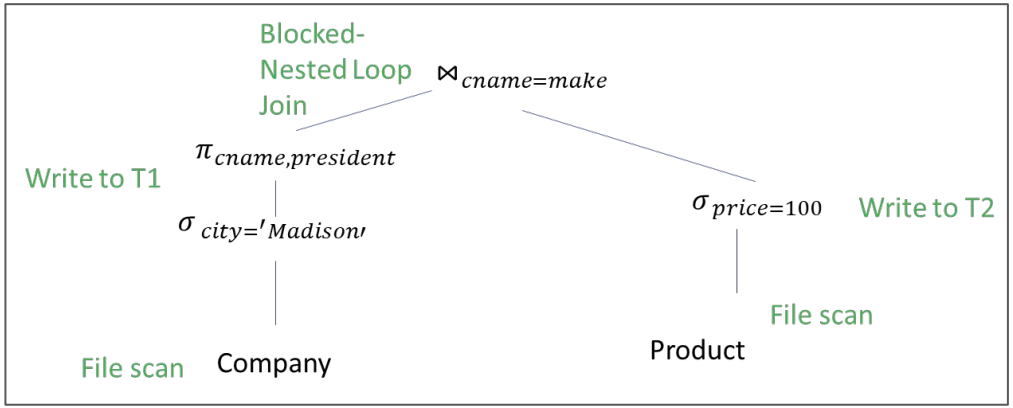
Cost = (Cost to scan company table + perform selection on city) + (Cost to scan product table + perform selection on price) + (Cost to read from T1 into memory) + (Cost to read from T2 into memory)

(Cost to scan product table + perform selection on price) = 200 + 200/2500 = **200 + 2/25 I/O**

Practice Question

Q(c). Given the following information:
T(Company) = 2000 records
T(Product) = 20000 records
B(Company) = 200 pages
B(Product) = 200 pages
V(Product, price) = 2500
V(Company, city) = 40
You always used the block-nested loop join. Both relations are clustered.
M = 100 (pages)

Please calculate I/O cost for each query plan in (a) and (b)



Cost = (Cost to scan company table + perform selection on city) + (Cost to scan product table + perform selection on price) + (Cost to read from T1 into memory) + (Cost to read from T2 into memory)

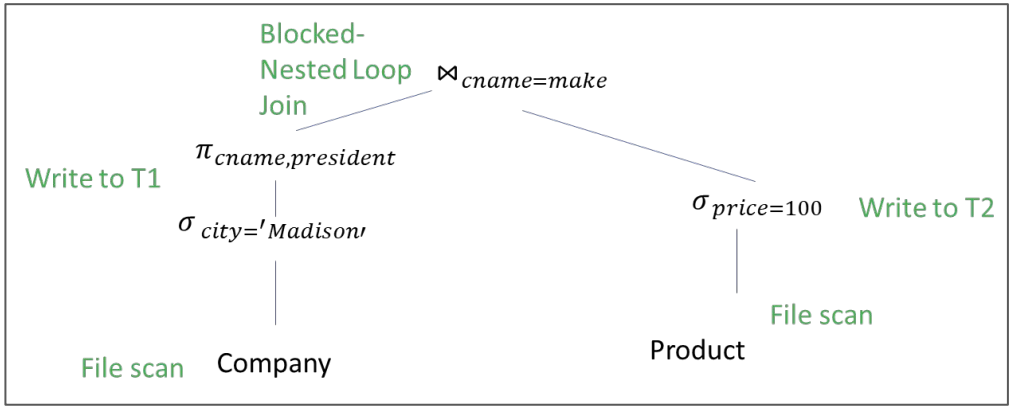
(Cost to read from T1 into memory) = **5 I/O**

(Cost to read from T2 into memory) = 2/25 of a page = **1 I/O**

Practice Question

Q(c). Given the following information:
T(Company) = 2000 records
T(Product) = 20000 records
B(Company) = 200 pages
B(Product) = 200 pages
V(Product, price) = 2500
V(Company, city) = 40
You always used the block-nested loop join. Both relations are clustered.
M = 100 (pages)

Please calculate I/O cost for each query plan in (a) and (b)



Cost = (Cost to scan company table + perform selection on city) + (Cost to scan product table + perform selection on price) + (Cost to read from T1 into memory) + (Cost to read from T2 into memory)

Cost: (200 + 5) + (200 + 2/25) + 5 + 1 = 412 I/O.

Good luck for your exams!

:-)