

CSCI3170 Introduction to Database Systems

Tutorial 8 Query Processing

Table of Content

- Query Evaluation
 - Join
 - Selection
 - Projection
- Query Optimization

Schema

- Supplier(sid, sname, location)
 - 500 pages, 80 tuples/page
- Supplier_Part(sid, pid, quantity)
 - 1000 pages, 120 tuples/page

Example 1

```
Select *  
From Supplier S, Supplier_Part SP  
Where S.sid =SP.sid
```

Join Operation

- Nested Loops Join
 - A tuple at a time
 - A page at a time
- Block Nested Loops Join
- Index Nested Loops Join
- Sort-Merge Join

A tuple at a time

for each tuples s in S do (S is called outer relation)
 for each tuple sp in SP do (SP is called inner relation)
 if ($s.sid = sp.sid$) then
 add $\langle s \ sp \rangle$ to result set

Cost:

- Scan S : 500 I/O
- For each page of S , pages of SP read = $80 * 1000 = 80,000$
- Total = $500 + 500 * 80000 = 40,000,500$
- Switch S and SP , the total is $1000 + 1000 * 120 * 500 = 60,001,000$

\therefore 1 page of S needs 80000 pages of SP
Now we have 500 pages of S

A page at a time

Improve the join by joining a page of tuples at a time

```
for each page p of S
  for each page q of SP
    output all  $r \in p$  and  $sp \in q$  such that  $r.sid = sp.sid$ 
```

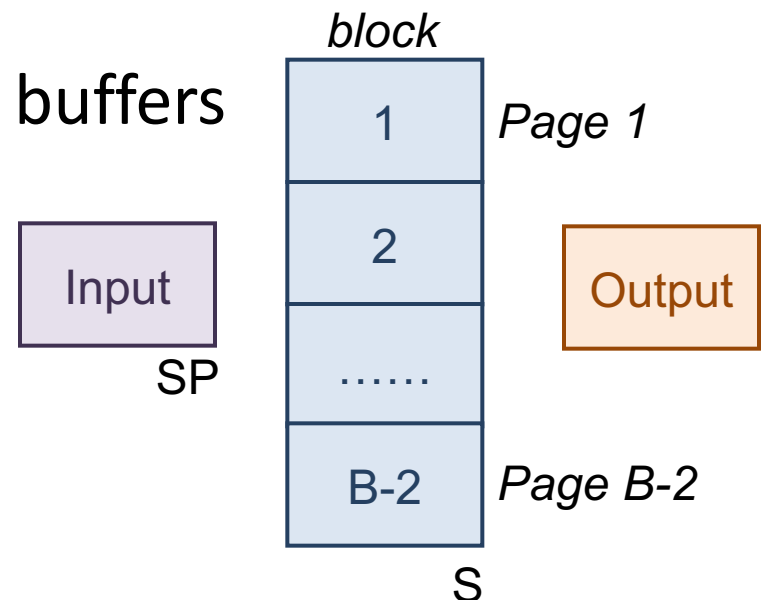
Cost

- Scan S: 500 I/Os
- For each page of S, read 1000 pages of SP
- Total: $500 + 500 \times 1000 = 500,500$

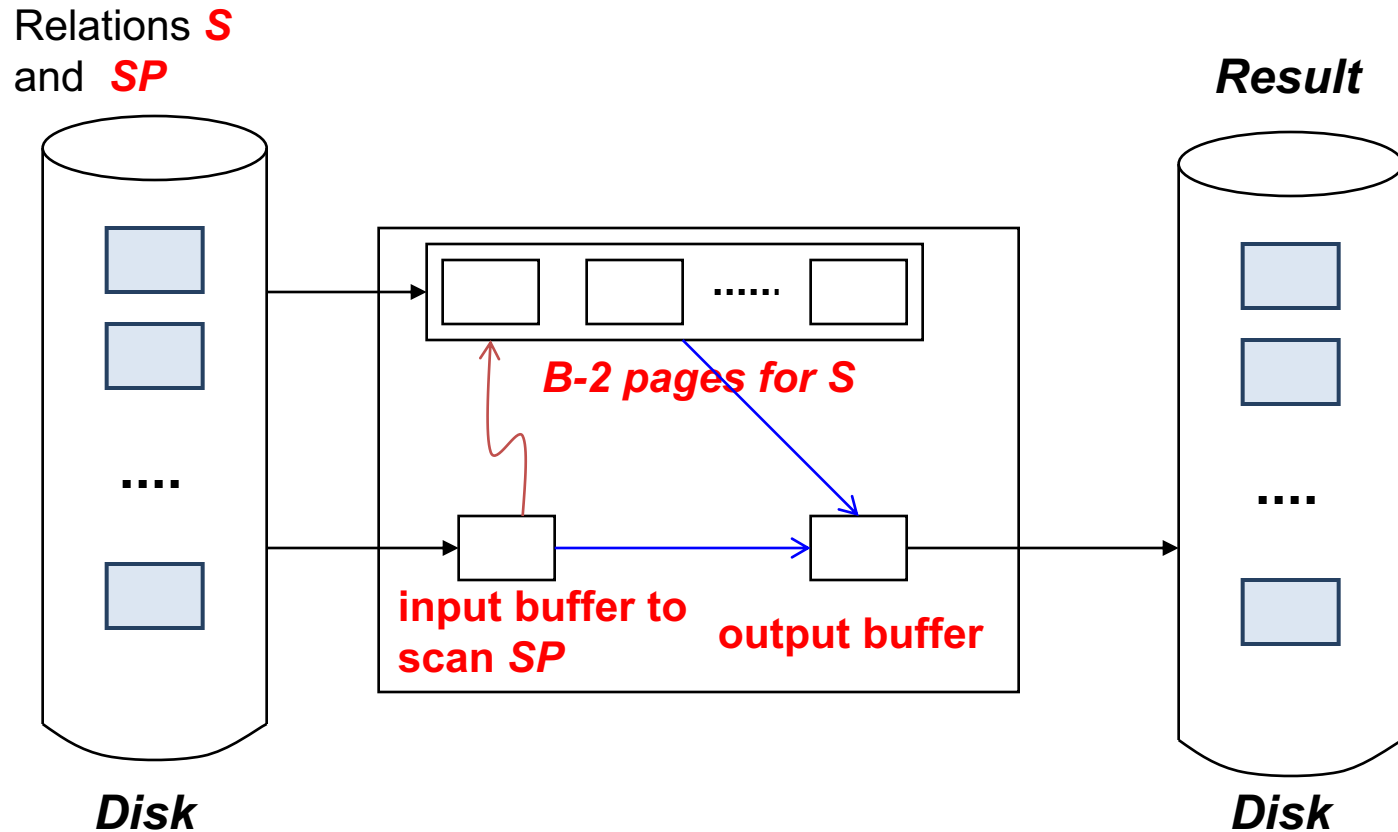
Block Nested Loops Join

```
foreach block P for S
  foreach page q for SP
    foreach s ∈ P and sp ∈ q such that s.sid = sp.sid
      add <s sp> to the result.
```

- Suppose we assign enough buffers to hold B pages. ($B > 2$)
 - 1 for input buffer (for SP)
 - 1 for output buffer
 - $B-2$ for outer relation (for S)



Block Nested Loops Join (Cont.)



Block Nested Loops Join (Cont.)

- Cost (Assume $B = 52$. So S divided into 10 blocks.)
 - Scan S : 500 I/Os
 - For each block of S , scan SP , for 1000 I/Os
 - Total: $500 + 10 \times 1000 = 10,500$
 - Switch S and SP , the cost is: $1000 + 20 \times 500 = 11,000$
- Observation:
 - Choice of outer and inner relation will affect the cost.
 - Choose the **smaller one** as the outer relation
 - The buffer size will affect the cost
 - The bigger is the buffer, the fewer is the I/O cost
 - Trade off between space and time

Index Nested Loops Join

- Assume we have an index on sid of S then
for each $sp \in SP$
 for each $s \in S$ where $s.sid = sp.sid$
 add $\langle s, sp \rangle$ to the result
- Note that for each tuple in SP, we use index to find the matching tuple in S.
- Assume we use an extendible hash index


Cost

- Scan SP: 1000 I/Os
- For each tuple in SP, an average of 1.2 I/O to get to the bucket page containing the matching S data entry, retrieve the S tuple for 1 I/O
- Note: sid is the primary key of Supplier relation
- Total: $1000 + 120 \times 1000 \times (1+1.2) = 265,000$ I/Os

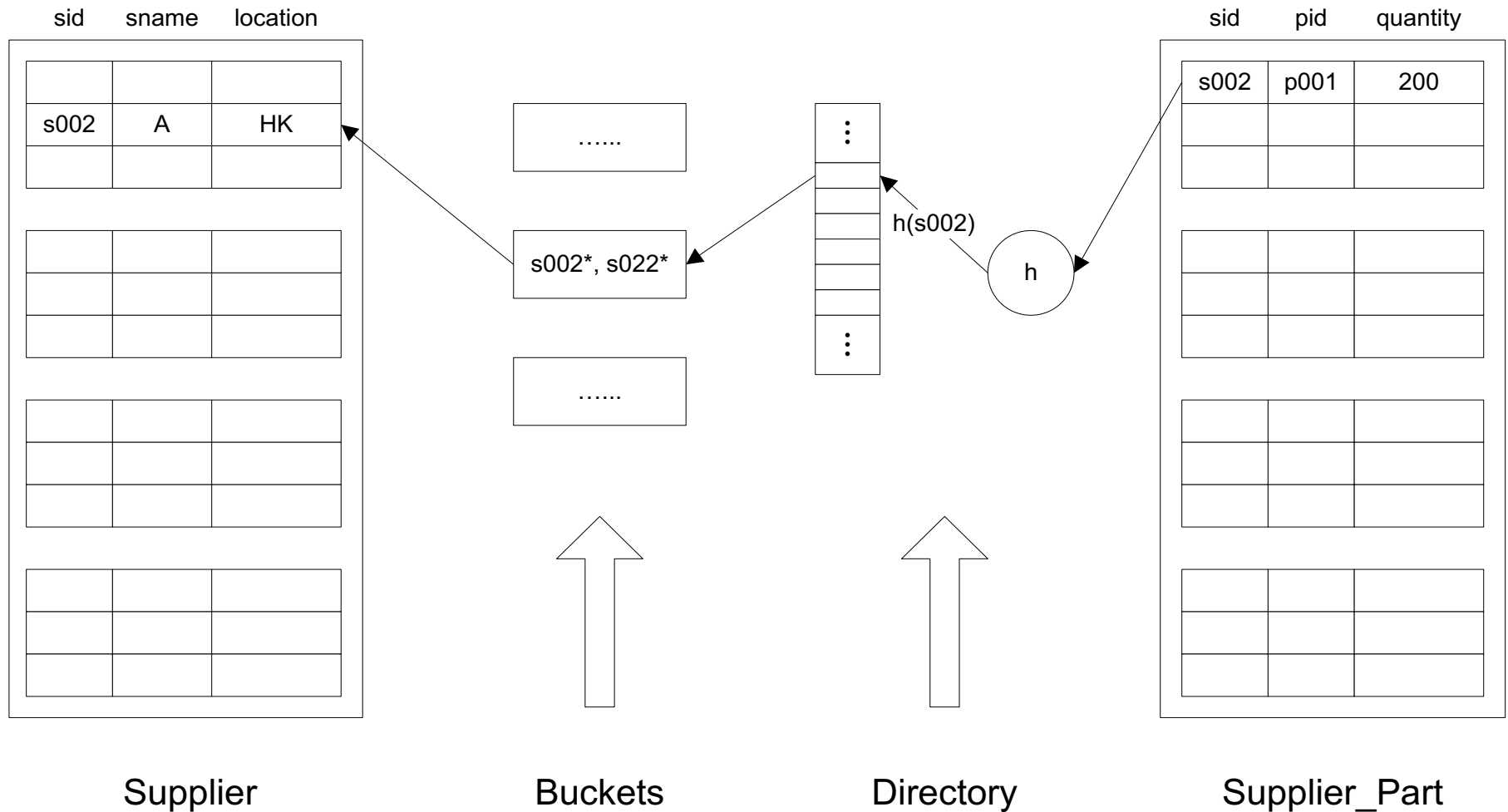
Obtained using experiments



∴ Hash all tuples of SP
i.e. 120×1000 tuples



Index Nested Loops Join (2)



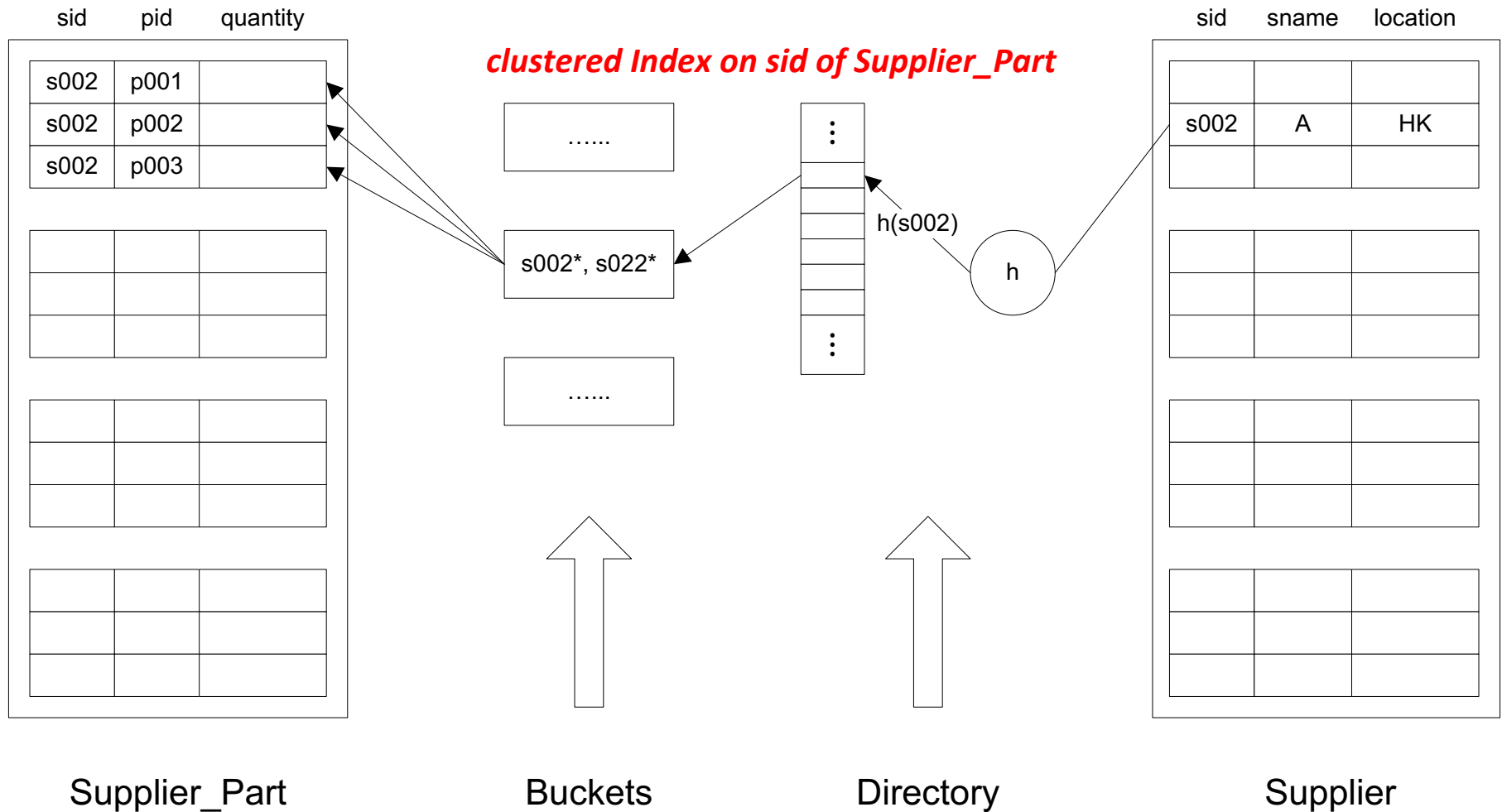
Index Nested Loops Join (3)

- Assume an index on sid of SP

```
foreach s ∈ S do  
    foreach sp ∈ SP where s.sid = sp.sid  
        add <s, sp> to the result
```

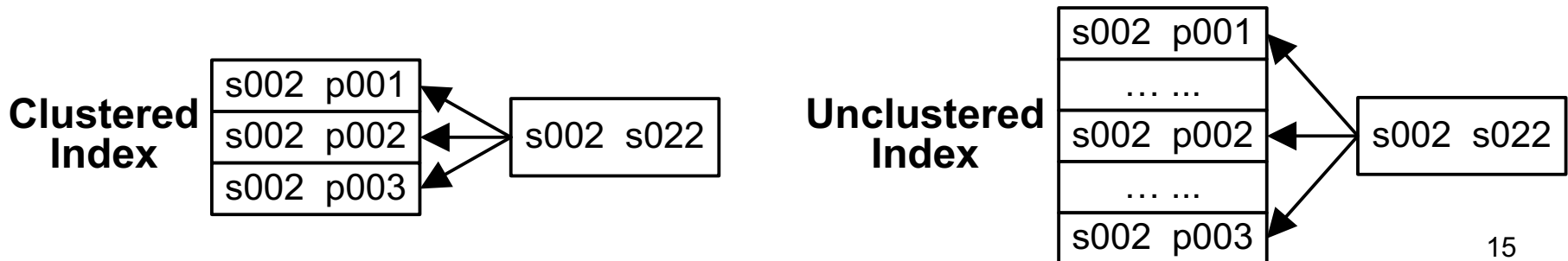
- Note that for each tuple in S, we use index to find match tuple in SP.

Index Nested Loops Join (4)



Index Nested Loops Join (5)

- Scan S: 500 I/Os
- For each tuple in S, an average of 1.2 I/O to get to the bucket page containing the matching SP data entry
- For each matching SP data entry, retrieve the SP tuple for 1 I/O.
- Estimation: 40000 suppliers supply 120000 parts, so each supplier supplies 3 parts on average.
 - Clustered Index (**all 3 parts in same page**):
 $\text{total} = 500 + 40000 \times 2.2 = 88,500$
 - Unclustered Index (**3 parts in different pages**):
 $\text{total} = 500 + 40000 \times 1.2 + 40000 \times 3 = 168,500$



Sort-Merge Join

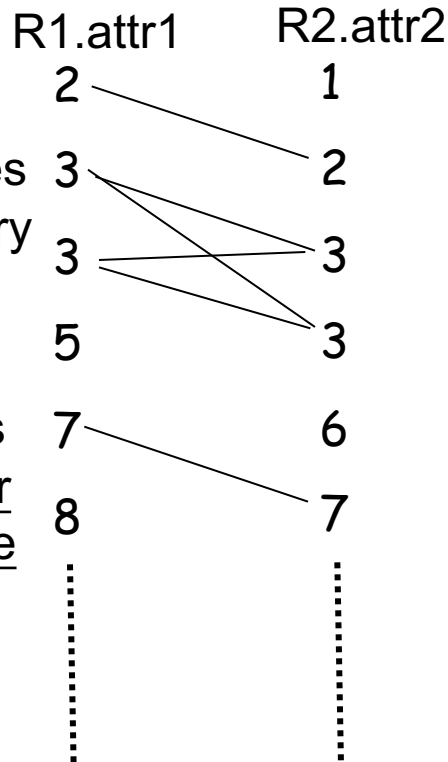
- $S \bowtie_{i=j} SP$
- Sort Supplier and Supplier_Part in ascending order on the sid, then scan them to do a merge
- Scan S until s.sid \geq sp.sid
- Scan SP until sp.sid \geq s.sid
- At this point, all S tuples with same value in S_i (current S group) and all SP tuples with same value in SP_j (current SP group) match. Output $\langle s, sp \rangle$ for all pairs of such tuples.
- Resume scanning S and SP

Sort-Merge Join (Cont.)

Case 1:

Both join attributes are not the primary key

R1 is scanned once, R2 group is scanned once per matching R1 tuple

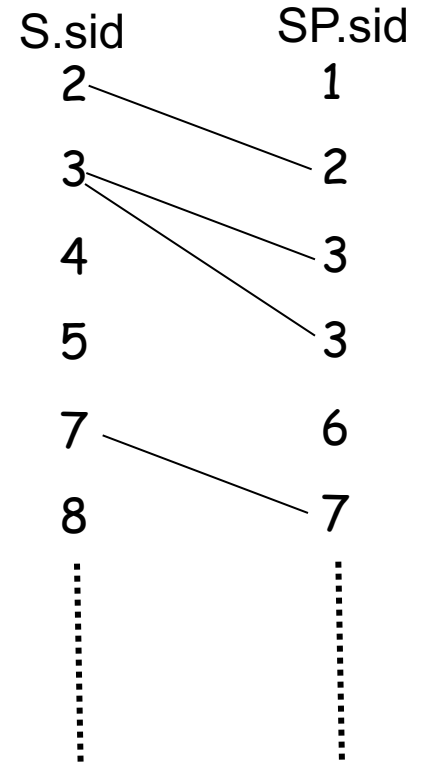


Case 2:

One of the join attribute is the primary key

Both S, SP are scanned once

Cost of join is the sum of the size of S and SP, plus the cost of sorting these two relations



Query Optimization

- Motivation
 - Ideal: find the best plan
 - Practical: avoid the worst plan
- Optimization steps
- Query Evaluation Plan
- Pipelined Evaluation
- An example

Optimization Steps

- A query is essentially treated as a σ - Π - \bowtie algebra expression
- Optimizing such a relational algebra expression involves two basic steps:
 - Enumerating alternative plans for expression evaluation.
 - Estimating the cost of each plan and choosing the plan with the lowest cost.

Query Evaluation Plan

- An extended algebra **tree** with annotations
- Each node indicating the access methods to use for each table and the implementation method to use for each relational operator.

Example

```
Select S.sname
From Supplier S, Supplier_Part SP
Where  S.sid = SP.sid and
       SP.pid > 'p800' and
       S.location = 'HK'
```

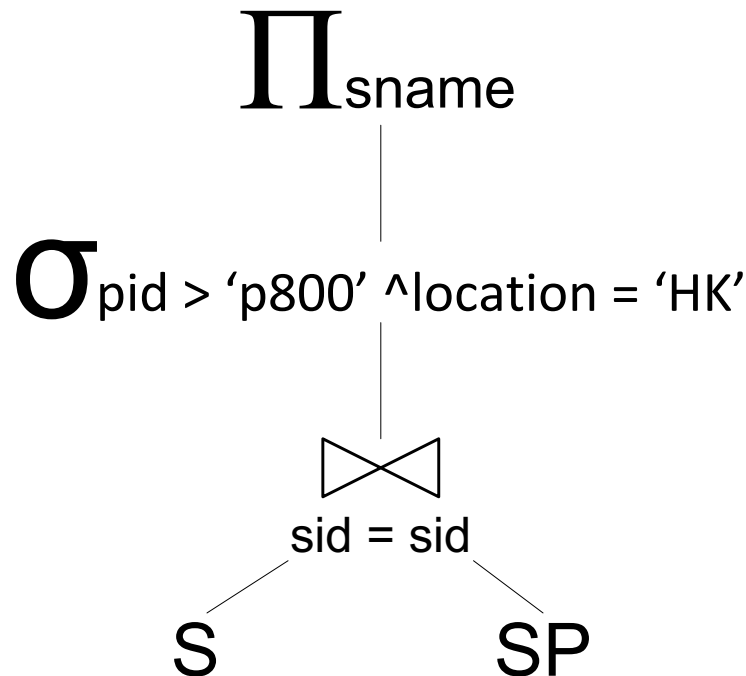
- Supplier:
 - 500 pages, 80 tuples/page
 - 40 possible locations, data distribute uniformly.21
- Supplier_Part:
 - 1000 pages, 120 tuples/page
 - Max part id is p1000, min part id is p0001
 - data distribute uniformly

Example (cont.)

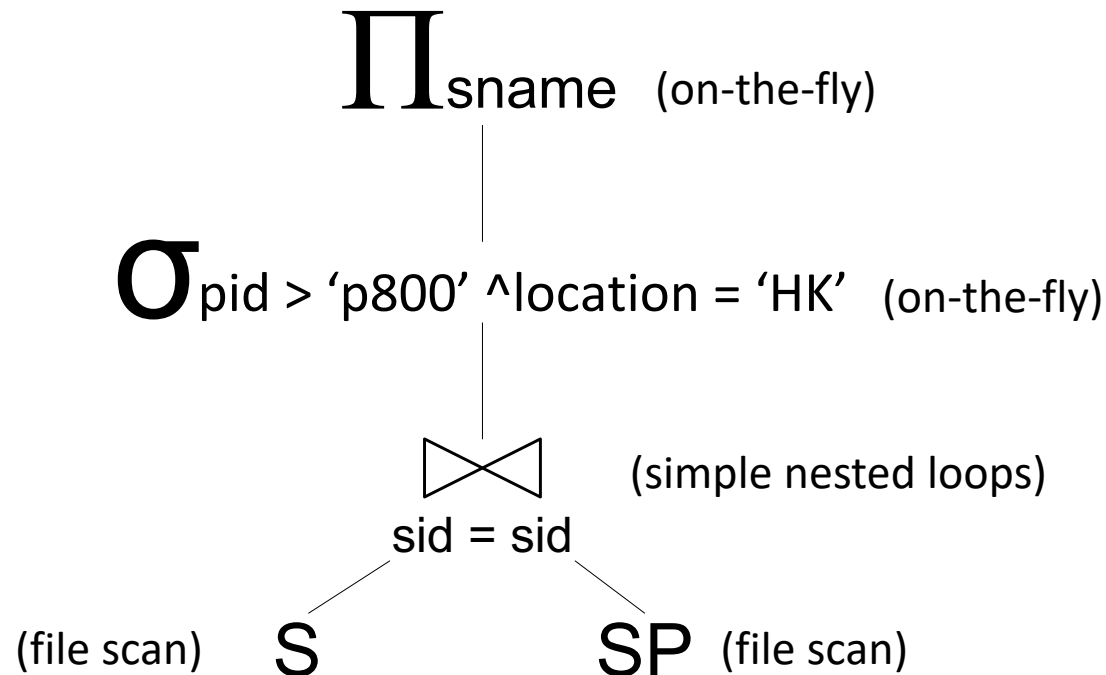
- Relation algebra of that query:

$$\Pi_{\text{sname}}(\sigma_{\text{pid} > \text{'p800'} \wedge \text{location} = \text{'HK'}}(S \bowtie_{\text{sid}=\text{sid}} SP))$$

- This algebra can be shown as a tree:



Full evaluation plan



Cost: $500 + 500 \times 1000 = 500500$

Pushing Selection

- May greatly reduce the data involved in a join
- In the example, do the selection and project first.
- The join will involve only sid, sname attributes and small portion of data.

Example

