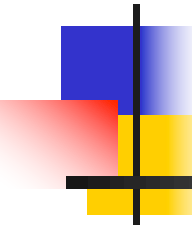
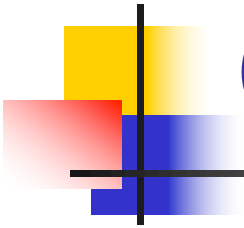


COMP231

Query Optimization

Prepared by Raymond Wong

Presented by Raymond Wong



Outline

- Introduction
- Materialization and On-the-fly
- Join over Two Tables
- Join over Multiple Tables

Introduction

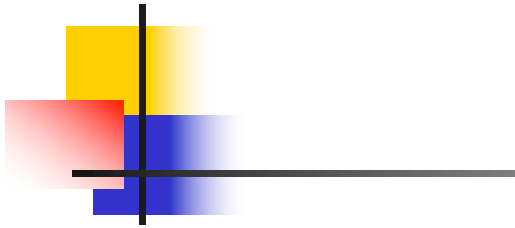
Find the name of all students with age greater than 20 who take 231.

- After users write SQL statements,
  - They want to obtain the results quickly
  - E.g., student (sid, sname, age)
    take (sid, cid)

```
select sname
from student S, take T
where S.sid = T.sid
    and  T.cid = "231"
    and S.age > 20
```

- However, there are many ways to obtain the results

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$



- SQL

```
select sname
from student S, take T
where S.sid = T.sid
    and  T.cid = "231"
    and S.age > 20
```

- Translate the query to relational algebra operations

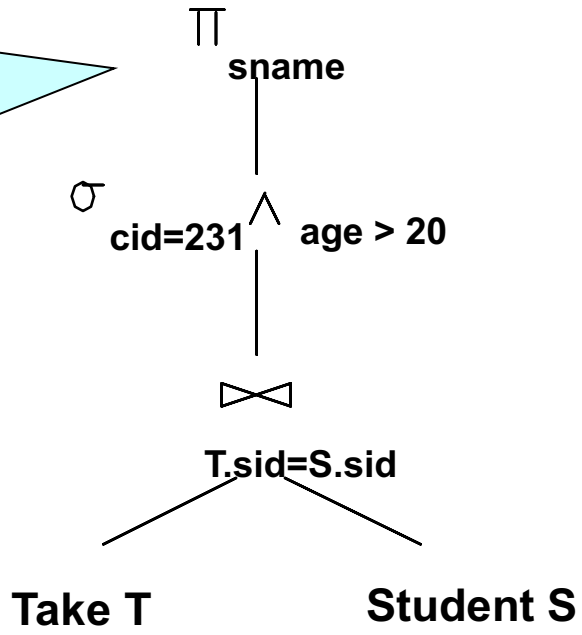
$$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$$

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

Evaluation Plan 1

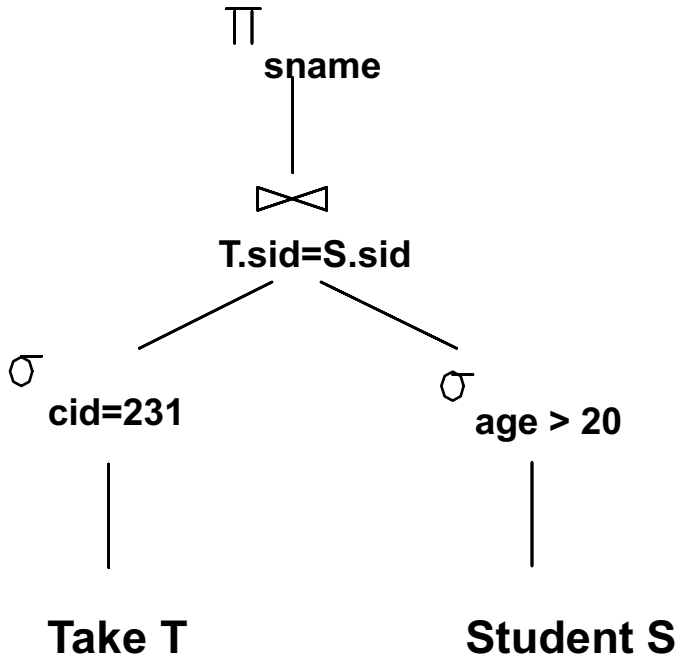
Evaluation plan

- Nodes are relational operators or tables
- Edges point to where the input comes from



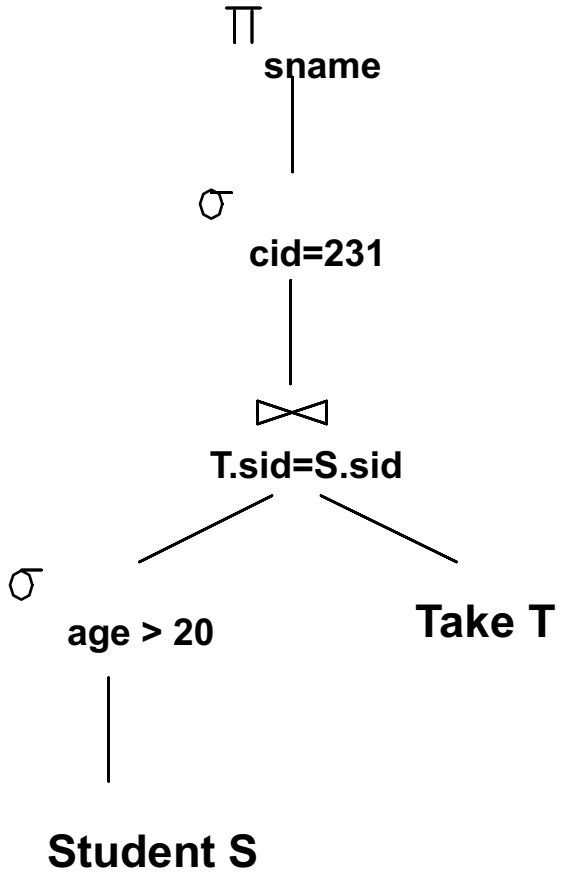
$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

Evaluation Plan 2



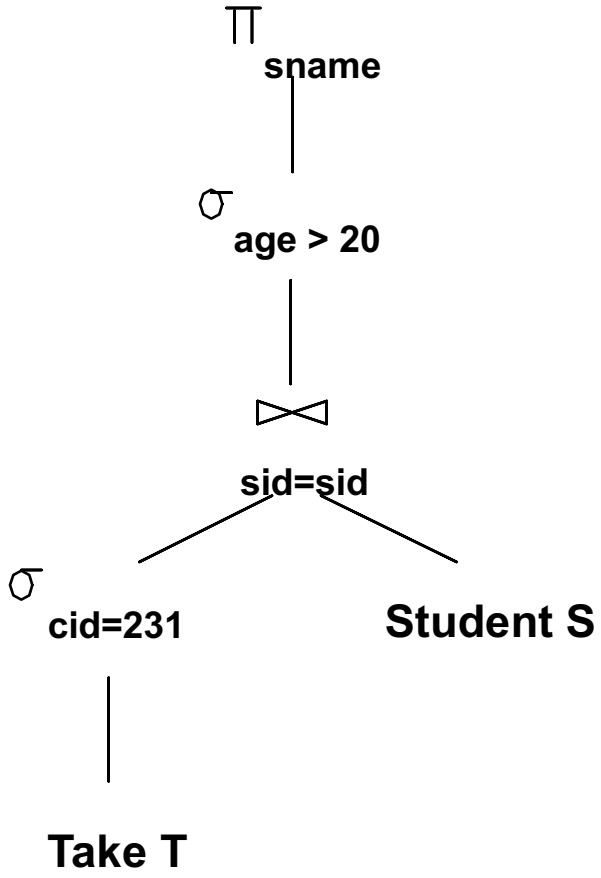
$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

Evaluation Plan 3

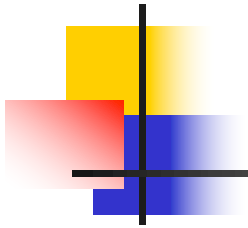


$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

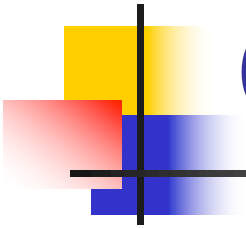
Evaluation Plan 4



$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$



- DBMS's job is to optimize the users' query by
  - Converting the query to a number of evaluation plans
  - Estimate the cost of each evaluation plan
  - Find the evaluation plan with the lowest cost



Outline

- Introduction
- Materialization and On-the-fly
- Join over Two Tables
- Join over Multiple Tables

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

Materialization and On-the-fly

- Let ***op1*** and ***op2*** be two relational algebra operations, and ***op1*** is performed on the result of ***op2***
- The evaluation of ***op1*** is ***on-the-fly*** if the result of op2 is directly sent to ***op1*** (i.e, not stored in a temporary file)
- It is ***materialized*** if that result is stored in a temporary file first
- On-the-fly evaluation is called ***pipelined*** evaluation
- On-the-fly can be more efficient than materialized

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

Materialization and On-the-fly

- Materialization
- On-the-fly

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

Materialization

Disk

Student

sid	sname	age
100	Raymond	20
200	Peter	21

Take

sid	cid
100	231
200	231
200	170

sid	sname	age	cid
100	Raymond	20	231
200	Peter	21	231
200	Peter	21	170

Reading

Materialization

Writing

Memory

sid	sname	age
100	Raymond	20
200	Peter	21

sid	cid
100	231
200	231
200	170

Join

sid	sname	age	cid
100	Raymond	20	231
200	Peter	21	231
200	Peter	21	170

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

Materialization

Disk

Student

sid	sname	age
100	Raymond	20
200	Peter	21

Take

sid	cid
100	231
200	231
200	170

sid	sname	age	cid
100	Raymond	20	231
200	Peter	21	231
200	Peter	21	170

Memory

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

Materialization

Disk

Student

sid	sname	age
100	Raymond	20
200	Peter	21

Take

sid	cid
100	231
200	231
200	170

sid	sname	age	cid
100	Raymond	20	231
200	Peter	21	231
200	Peter	21	170

Reading

Output to the screen

Memory

sname
Peter

π_{sname}

sid	sname	age	cid
200	Peter	21	231

$\sigma_{cid=231 \wedge age>20}$

sid	sname	age	cid
100	Raymond	20	231
200	Peter	21	231
200	Peter	21	170

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

Materialization and On-the-fly

- Materialization
- On-the-fly

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

On-the-fly

Disk

Student

sid	sname	age
100	Raymond	20
200	Peter	21

Take

sid	cid
100	231
200	231
200	170

Reading

Memory

sid	sname	age
100	Raymond	20
200	Peter	21

sid	cid
100	231
200	231
200	170

Join

sid	sname	age	cid
100	Raymond	20	231
200	Peter	21	231
200	Peter	21	170

$\pi_{sname}(\sigma_{cid=231 \wedge age>20}(Take \bowtie_{Take.sid=Student.sid} Student))$

On-the-fly

Disk

Student

sid	sname	age
100	Raymond	20
200	Peter	21

Take

sid	cid
100	231
200	231
200	170

Output to the screen

Memory

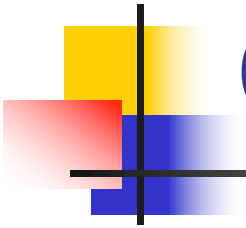
sname
Peter

π_{sname}

$\sigma_{cid=231 \wedge age>20}$

sid	sname	age	cid
200	Peter	21	231

sid	sname	age	cid
100	Raymond	20	231
200	Peter	21	231
200	Peter	21	170



Outline

- Introduction
- Materialization and On-the-fly
- Join over Two Tables
- Join over Multiple Tables

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Example Schema

- Assuming the following table sizes:
  - Student: 500 pages, 80 tuples/page, 50 bytes/tuple
  - Take: 1000 pages, 100 tuples/page, 40 bytes/tuple
- Assume that there are 200 courses in table Take

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Example Schema

- Join
  - Simple-nested loop Join
  - Block-nested loop Join
  - Sort-Merge Join
  - Index-Nested loop Join

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

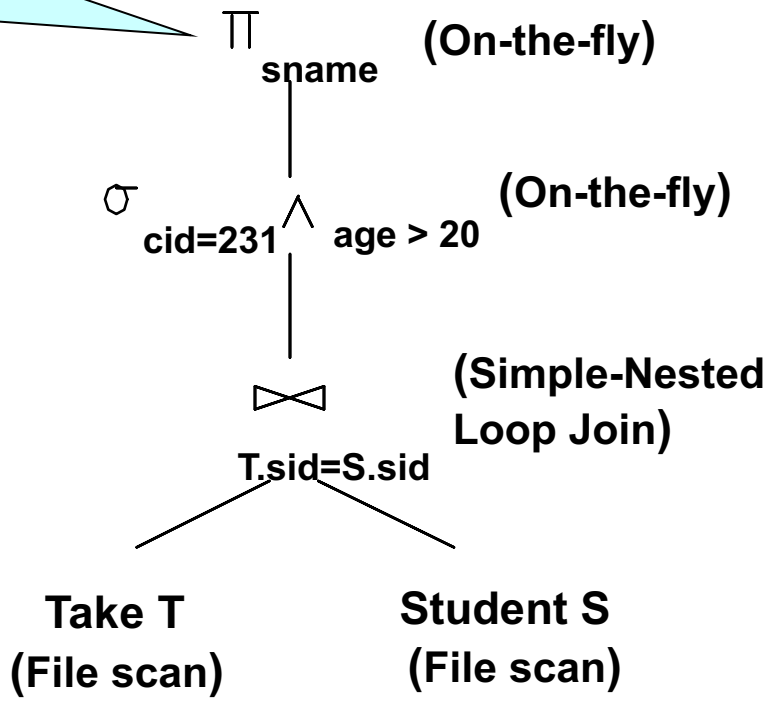
- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 1

We do NOT calculate the cost of writing the output. Why?

Suppose that Student is the outer relation

Cost = 500+500*1000
=500500 pages



Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

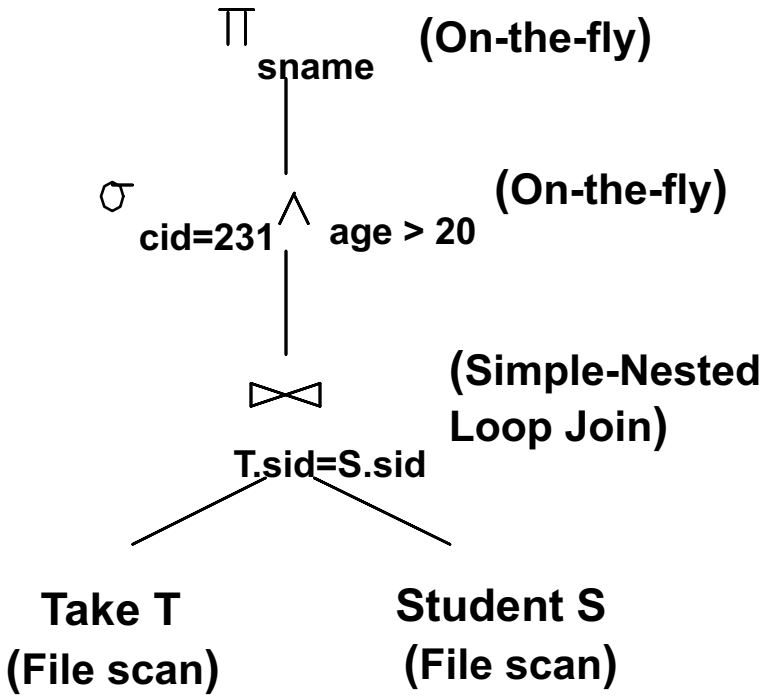
- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 1

Suppose that Take is the outer relation

Cost = 1000+1000*500
=501000 pages

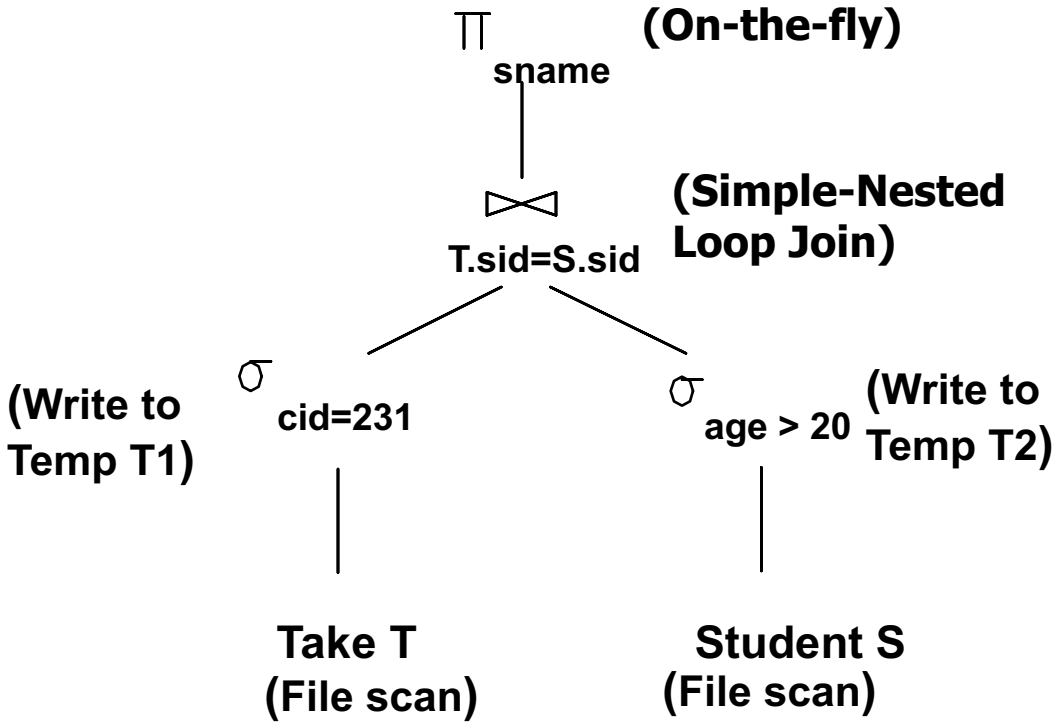
If Student is the outer relation,
then the cost is 500500 pages
If Take is the outer relation,
then the cost is 501000 pages
Which one should we choose?



Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 2



Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 2

Cost of Reading T = 1000 pages

Size of T1 = 1000/200 = 5 pages

Cost of Writing T1 = 5 pages

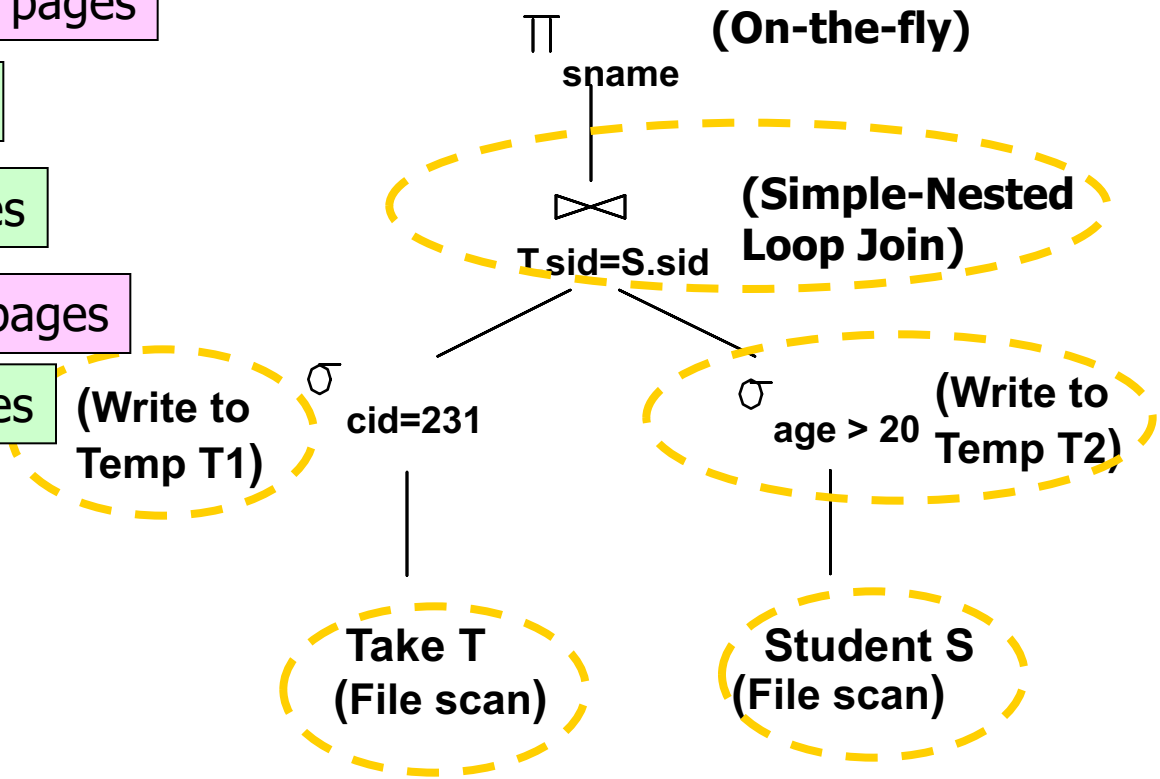
Cost of Reading S = 500 pages

Size of T2 = 500/2 = 250 pages

Cost of Writing T2 = 250 pages

Suppose that T1 is the outer relation

Cost of Join
= 5 + 5*250
= 1255 pages

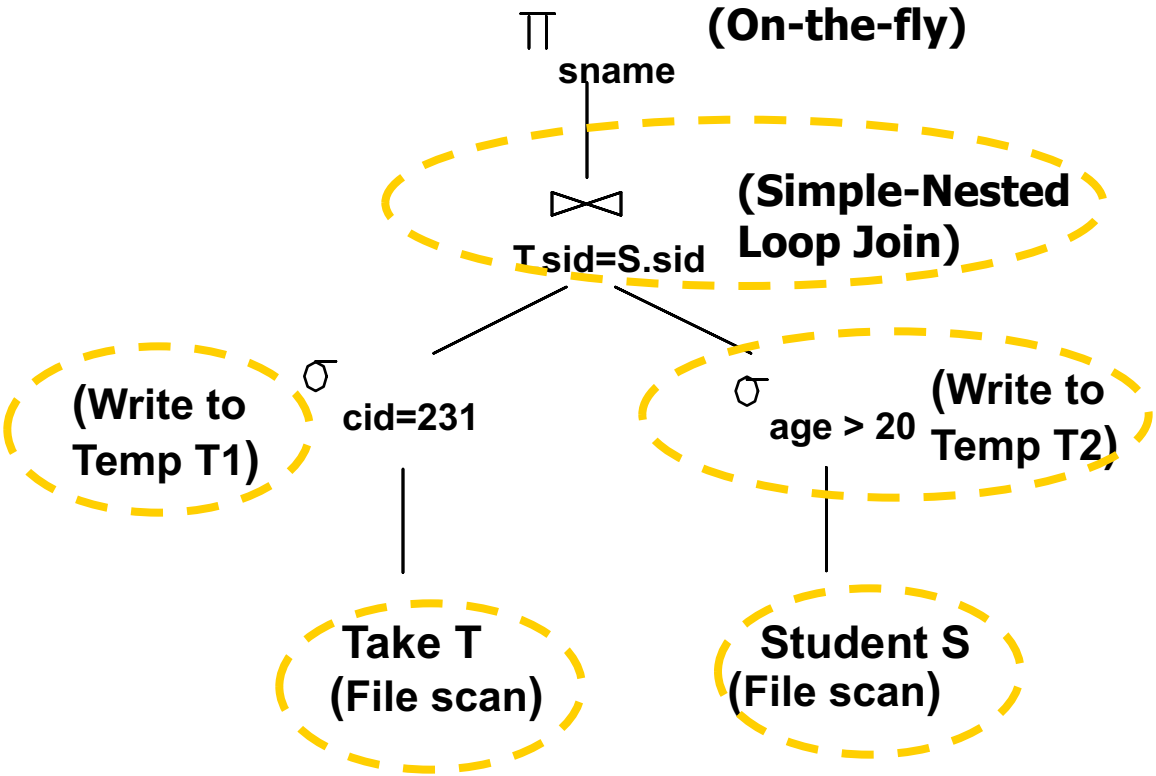


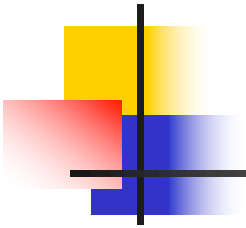
Total cost = 1000 + 5 + 500 + 250 + 1255 = 3010 pages

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 2



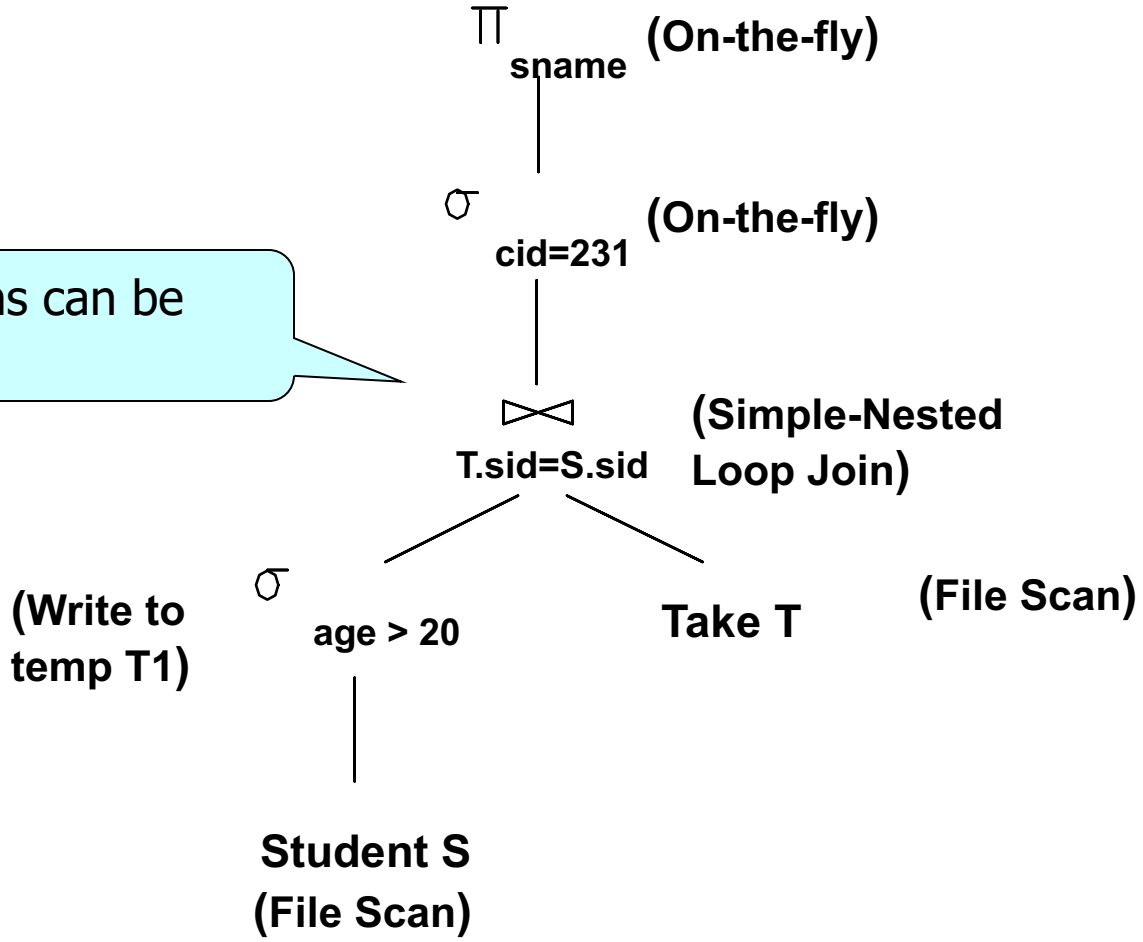


- Which evaluation plan is better?
- Plan 1 or Plan 2?
- Why?

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 3

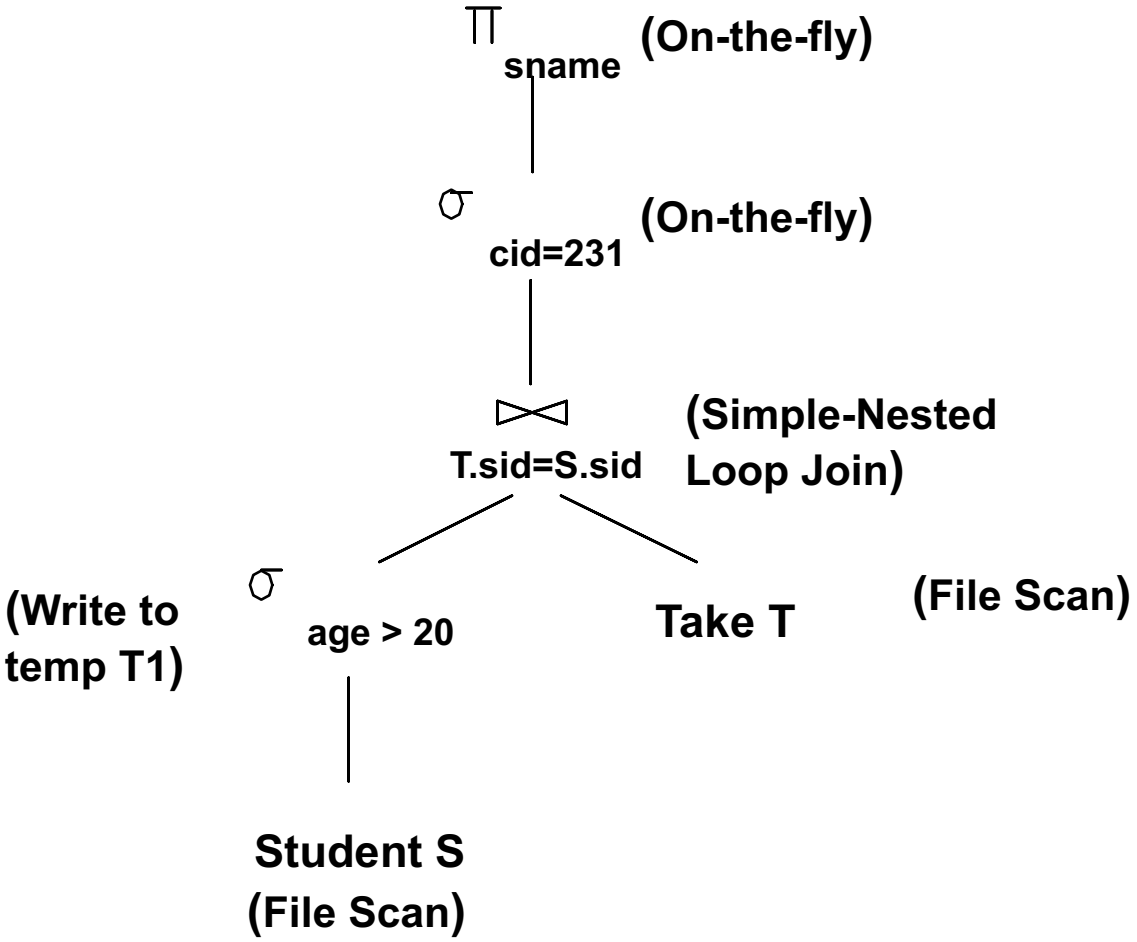


Similar derivations can be done.

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

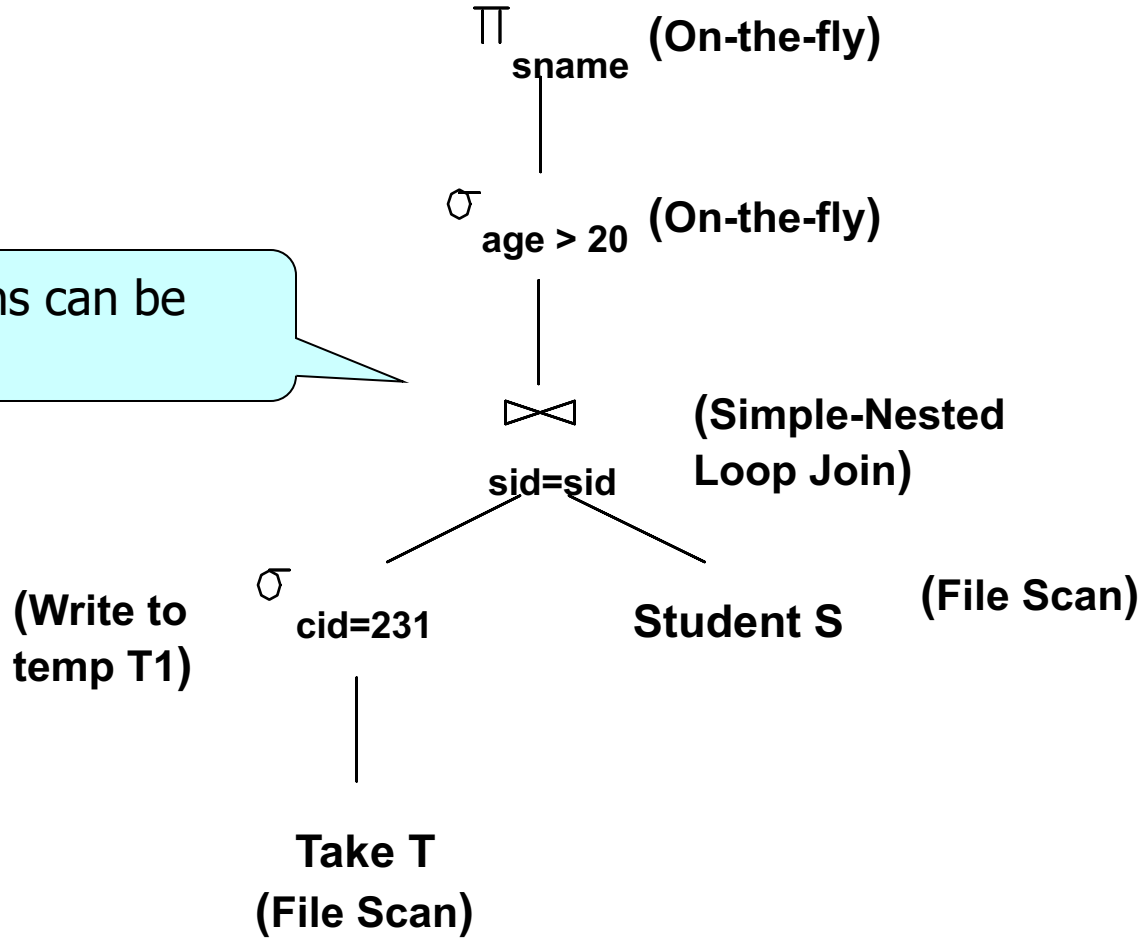
Evaluation Plan 3



Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 4



Similar derivations can be done.

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Example Schema

- Join
  - Simple-nested loop Join
  - Block-nested loop Join
  - Sort-Merge Join
  - Index-Nested loop Join

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

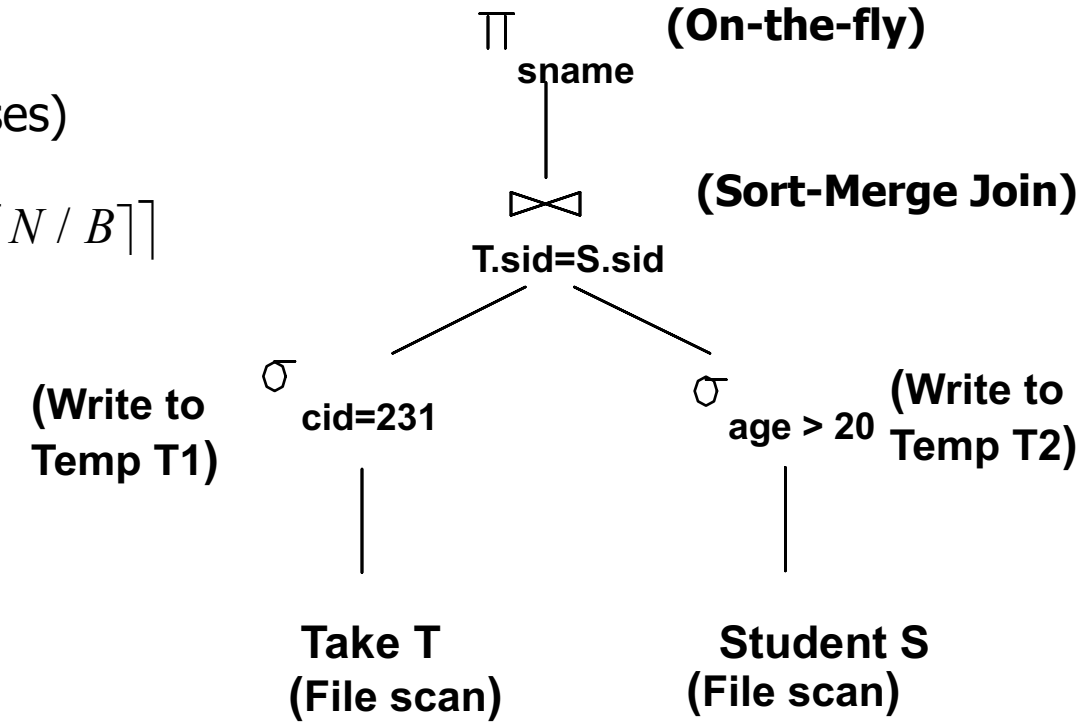
- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Size of the buffer = 3 pages

Evaluation Plan 2

Cost of Sorting = 2N * (# of passes)

where no. of passes= $1+\lceil \log_{B-1}\lceil N/B \rceil \rceil$



Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 2

Size of the buffer = 3 pages

Cost of Reading T = 1000 pages

Size of T1 = 1000/200 = 5 pages

Cost of Writing T1 = 5 pages

Cost of Reading S = 500 pages

Size of T2 = 500/2 = 250 pages

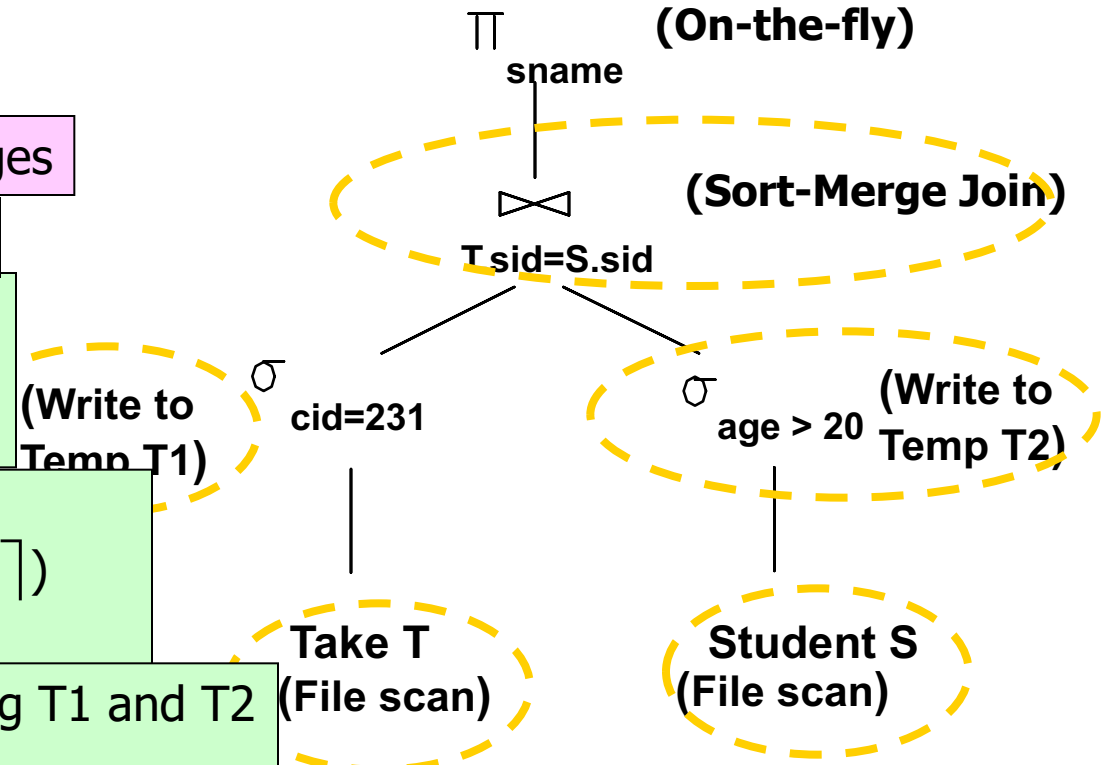
Cost of Writing T2 = 250 pages

Cost of Sorting T1
= $2*5*(1+\lceil \log_{3-1} \lceil 5/3 \rceil \rceil)$
= 20 pages

Cost of Sorting T2
= $2*250*(1+\lceil \log_{3-1} \lceil 250/3 \rceil \rceil)$
= 4000 pages

Cost of Merging = cost of reading T1 and T2
= 5 + 250 = 255 pages

Total cost = 1000 + 5 + 500 + 250 + 20 + 4000 + 255 = 6030 pages

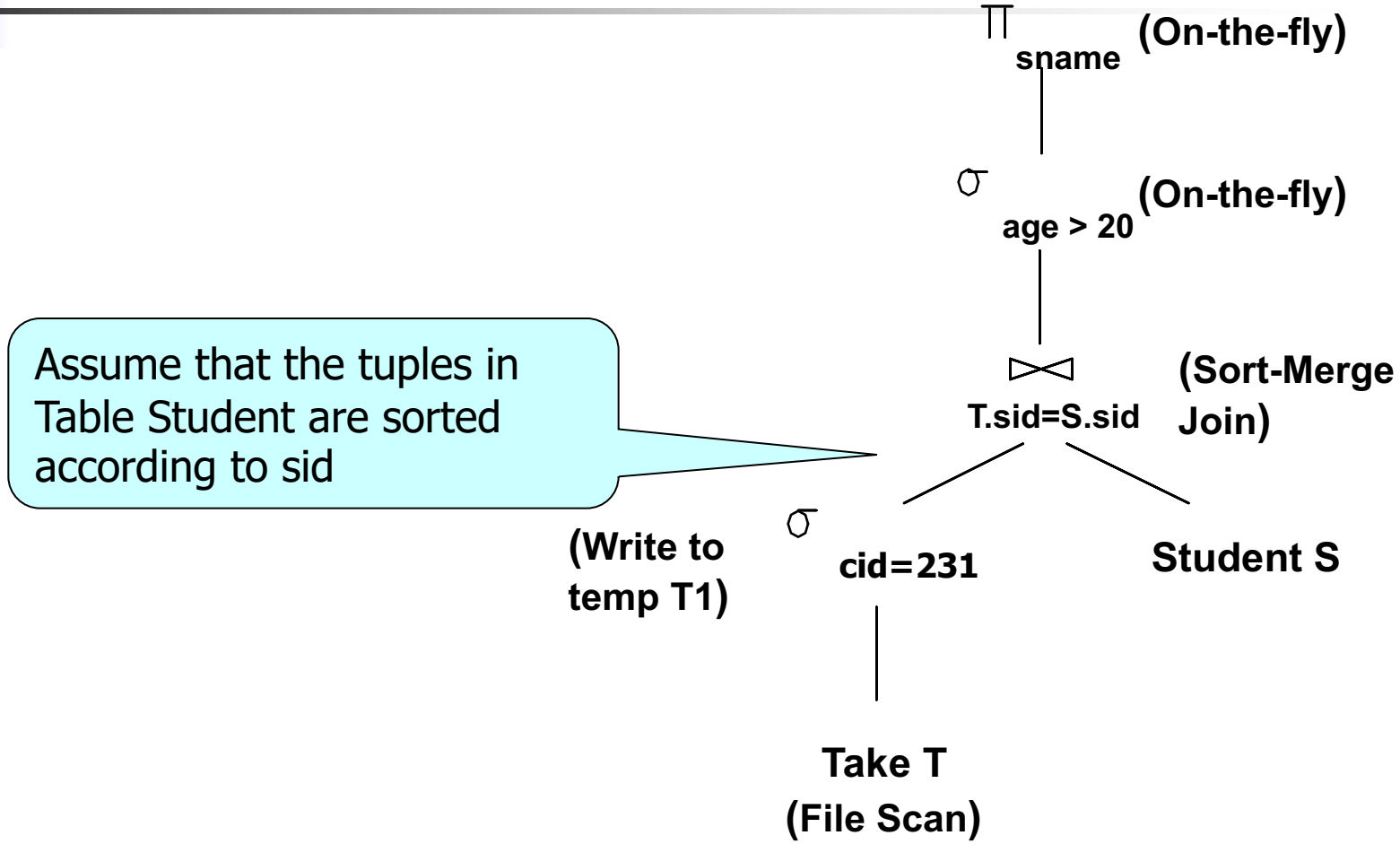


Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 3

Size of the buffer = 3 pages



Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 3

Size of the buffer = 3 pages

Cost of Reading T = 1000 pages

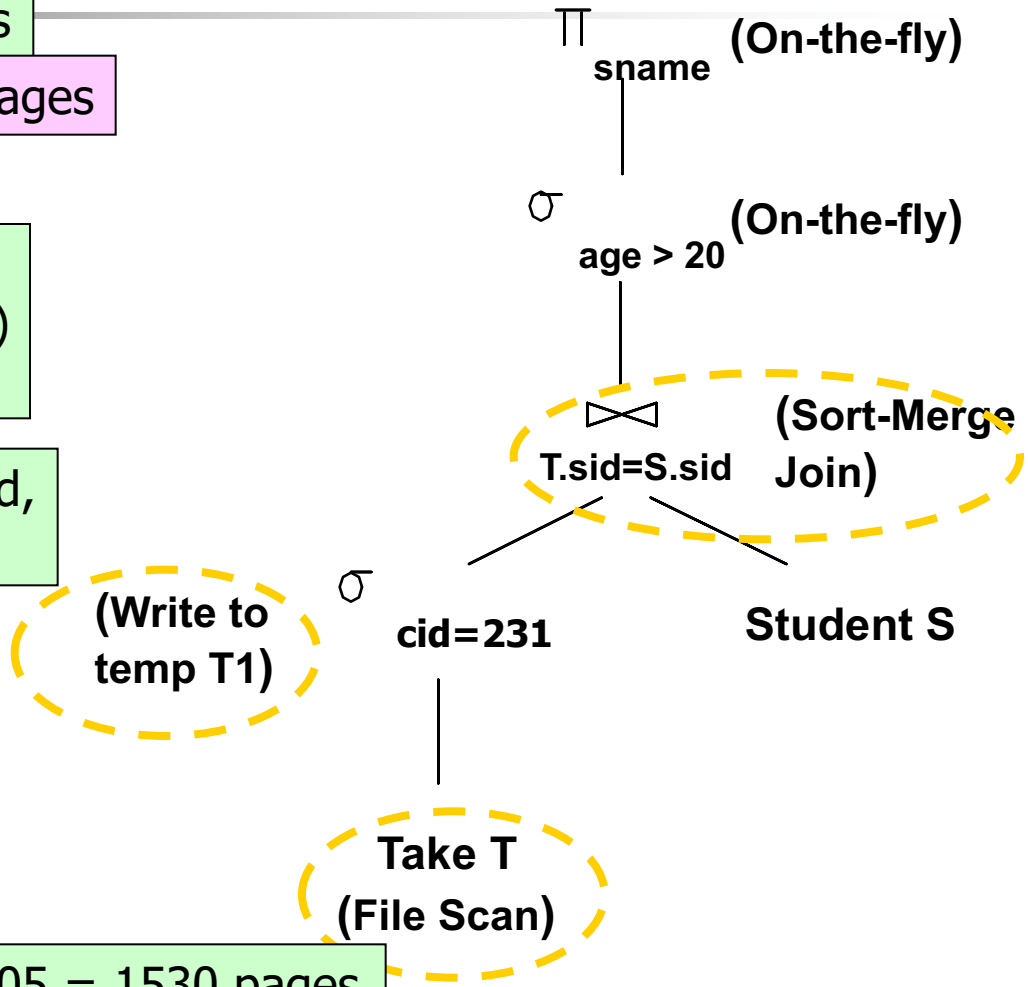
Size of T1 = 1000/200 = 5 pages

Cost of Writing T1 = 5 pages

Cost of Sorting T1
= $2*5*(1+\lceil \log_{3-1} \lceil 5/3 \rceil \rceil)$
= 20 pages

Since S is sorted according to sid, we do NOT need to sort S

Cost of Merging
= Cost of Reading T1 and S
= 5 + 500 = 505 pages



Total cost = 1000 + 5 + 20 + 505 = 1530 pages

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Example Schema

Size of the buffer = 3 pages

- Join
  - Simple-nested loop Join
  - Block-nested loop Join
  - Sort-Merge Join
  - Index-Nested loop Join

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

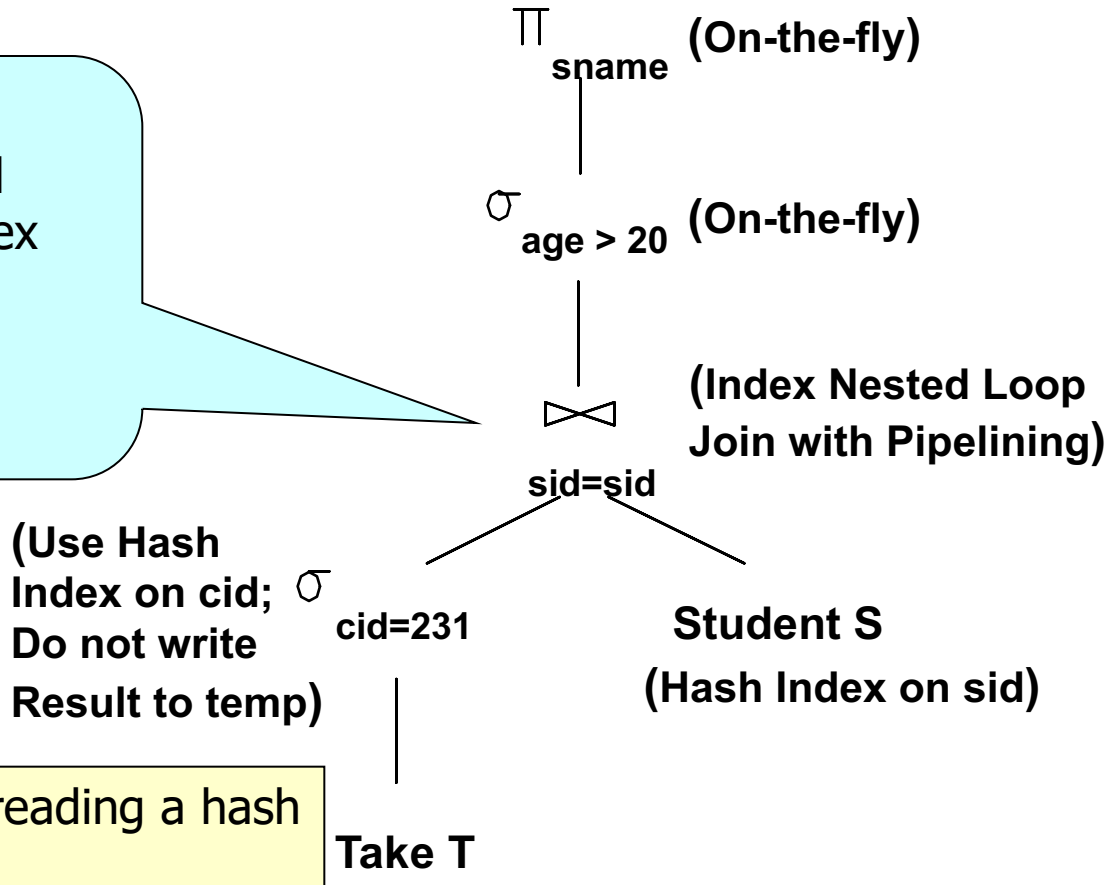
- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 4

Size of the buffer = 3 pages

Assume the following.

1. T is sorted according to cid
2. T has a clustered hash index on cid
3. S has an unclustered hash index on sid



Assume that the (average) cost of reading a hash index is 1.2 pages

Student: 500 pages, 80 tuples/page, 50 bytes/tuple
Take: 1000 pages, 100 tuples/page, 40 bytes/tuple

- 200 courses in table Take
- 1, 2, …, 40 in attribute Age of table Student

Evaluation Plan 4

Size of the buffer = 3 pages

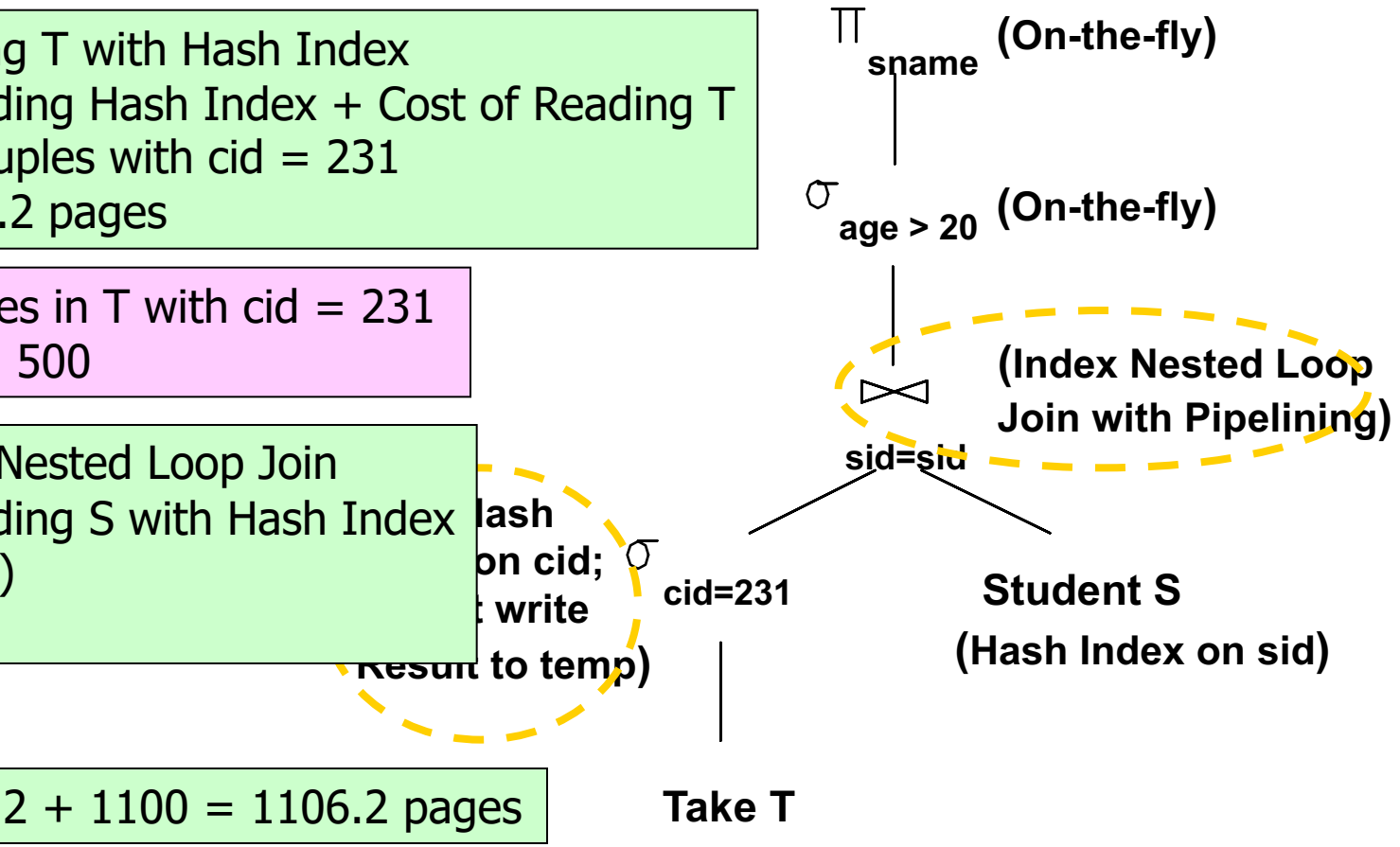
Size of the temp result = 1000/200 = 5 pages

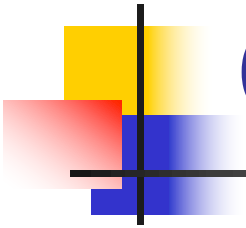
Cost of Reading T with Hash Index
= Cost of Reading Hash Index + Cost of Reading T containing tuples with cid = 231
= 1.2 + 5 = 6.2 pages

No. of tuples in T with cid = 231
= 100*5 = 500

Cost of Index Nested Loop Join
= Cost of Reading S with Hash Index
= 500*(1.2+1)
= 1100 pages

Total cost = 6.2 + 1100 = 1106.2 pages





Outline

- Introduction
- Materialization and On-the-fly
- Join over Two Tables
- Join over Multiple Tables

Join over Multiple Tables

- Relational Algebra Equivalences
- Left-Deep Plan

Relational Algebra Equivalences

Allow us to choose different join orders and to "push" selections and projections ahead of joins.

- Selections

$$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R)) \qquad (Commutative)$$

$$\sigma_{c1 \wedge \ldots \wedge cn}(R) \equiv \sigma_{c1}(\ldots \sigma_{cn}(R)) \qquad (Cascading)$$

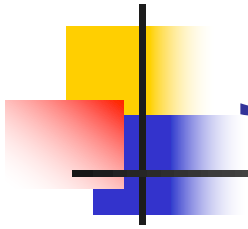
Relational Algebra Equivalences

- Join

$$R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T \qquad \textit{(Associative)}$$

$$(R \bowtie S) \equiv (S \bowtie R) \qquad \textit{(Commutative)}$$

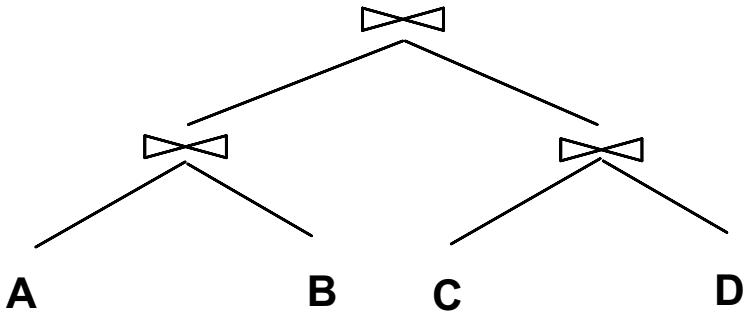
Join over Multiple Tables

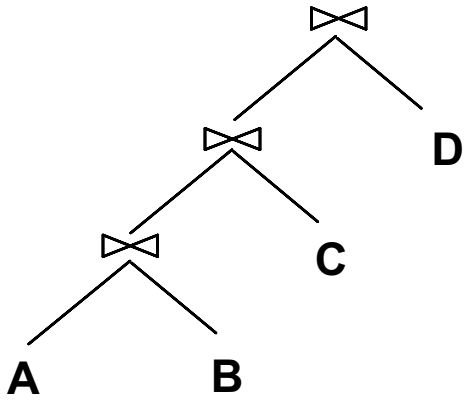


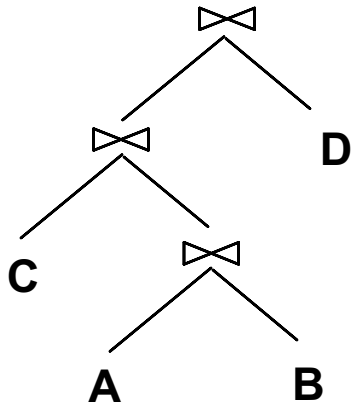
- Relational Algebra Equivalences
- Left-Deep Plan

Left-Deep Plans

- For multi-relation query, plan space can be very large and must be pruned.
  - As # of joins increases, # of alternative plans grows rapidly

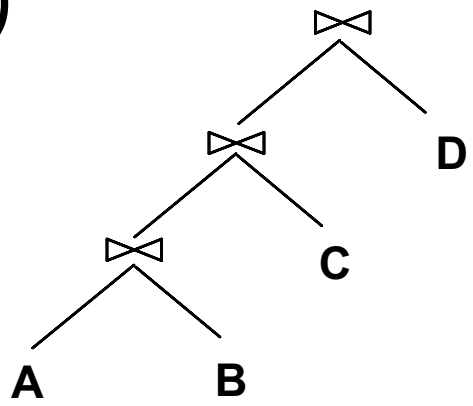






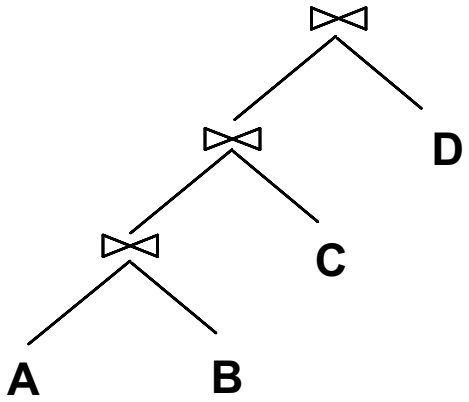
Left-Deep Plans

- Typically, *only* *left-deep trees* *(the right child of each join node is a base table)* are considered.
  - Significantly fewer, but still lots -- n! (24 for n = 4)



Left-Deep Plans

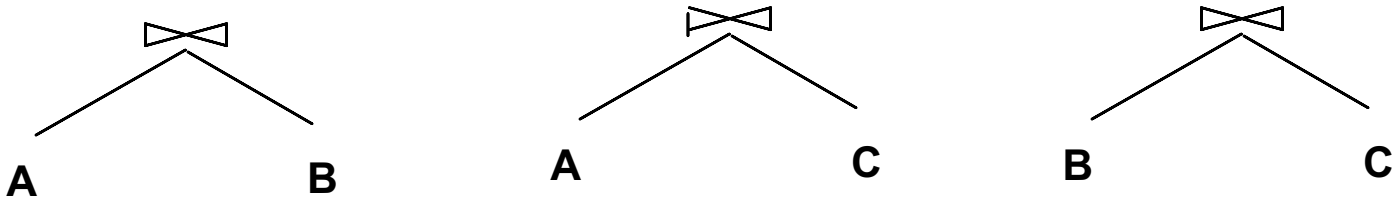
- Left-deep trees allow us to generate *fully pipelined* plans.
  - Intermediate results not written to temporary files.
  - Not all plans from left-deep trees are fully pipelined (e.g., sort-merge join).



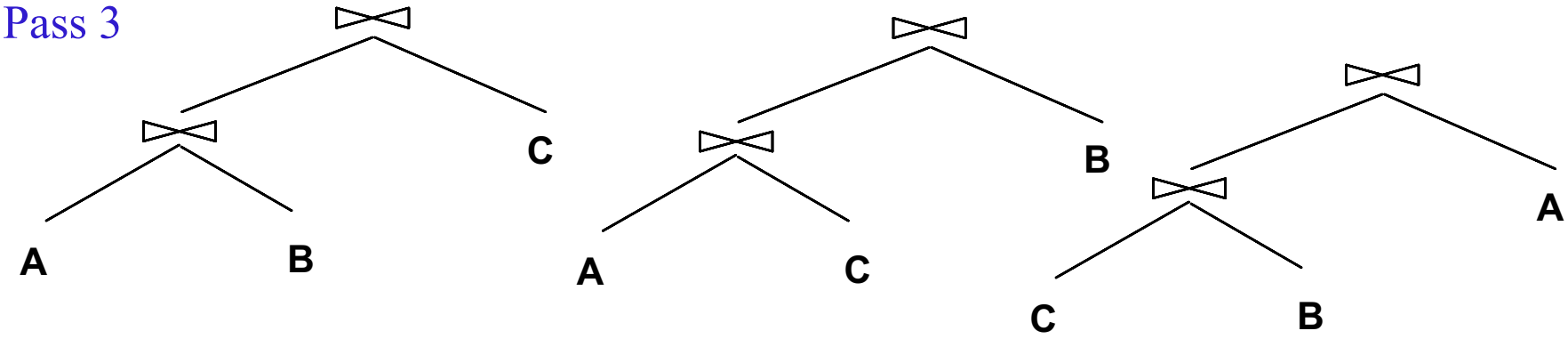
To join A, B, C

Pass 1

Pass 2



Pass 3



Enumeration of Left-Deep Plans

- Enumerated using N passes (if N relations joined):
  - Pass 1: Find best 1-relation plan for each relation.
  - Pass 2: Find best way to join result of each 1-relation plan (as outer) to another relation. *(All 2-relation plans.)*
  - Pass N: Find best way to join result of a (N-1)-relation plan (as outer) to the N'th relation. *(All N-relation plans.)*
- For each subset of relations, retain only:
  - Cheapest plan overall