SECTION 7

Relational Algebra and Query Plans

Practice problems

Today’s Overview

- Reminders
  - Project 3 due tomorrow 11pm
  - Homework 3 is out

- Optimistic Concurrency Control Worksheet (posted after week 6 section, see section website)
  - Multiversion timestamping (2 examples)
  - Validation (2 examples)

- Relational algebra and query plans <-> SQL
  - Worksheet for section

Section Worksheets Posted Online

- Optimistic Concurrency Control worksheet
- Relational Algebra worksheet

- Both are posted linked off the section site:
  - http://www.cs.washington.edu/education/courses/cse444/11wi/sections/index.html

Optimistic Concurrency Control

- Examples from worksheet posted during Week 6 to the section slides on the course website

Optimistic Concurrency Control

- Timestamps
  - Key Idea: The timestamp order defines the serialization order
  - Scheduler maintains:
    - TS(T) for all transactions T
    - RT(X), WT(X), C(T)
- Multiversion Timestamps
  - Keep multiple version of each data element along with the write timestamp
  - Will reduce number of aborts due to read-too-late problem
- Validation
  - Transaction informs schedule of its read and write sets before it validates

Multi-version timestamps

- Question 1

st1, st2, st3, st4, w1(A), com1, w2(A), w3(A), com3, r2(A), com2, r4(A), com4

- What will happen with a multi-version scheduler?
  - Each write creates a new copy of A unique to that transaction. The read attempts to read from the copy of A with the highest timestamp no greater than the timestamp of the read action's transaction.
  - R2(A) reads A2
  - R4(A) reads A2
- What would happen if we did not use a multi-version scheduler?
  - T2 rolled back because r2(A) would fail since a later transaction, T3, had already written to A

Multi-version timestamps

- Question 3

St1, st2, st3, st4, w1(A), com1, w4(A), com4, r3(A), com3, w2(A), com2

- What will happen with a multi-version scheduler?
  - W1(A) creates version A1, W4(A) creates version A4
  - R2(A) reads A1
  - W2(A) attempts to create version A2 whose previous version would be A1, but we see that the last read time of A1 was with T3. Thus, we have a write-to-late since T3 should have read A2 instead of A1.
- What would happen if we did not use a multi-version scheduler?
  - R3(A) would fail and rollback T3, because A has been written by later transaction T4
  - W2(A) would not fail since R3(A) has already been rolled back, however it would be ignored since W4(A) and T4 has already committed.

Validation

- Question 1

R1(A,B), R2(B,C), R3(C), V1, V2, V3, W1(A), W2(B), W3(C)

- What happens when this schedule is processed by a validation-based scheduler?
  - Does T1 validate?
    - Yes, nothing to check since it is the first to validate
  - Does T2 validate? T1 did not finish before T2 started or validated.
    - RS(T2) intersect WS(T1) = nothing
    - WS(T2) intersect WS(T1) = nothing
  - Does T3 validate? T1 and T2 both did not finish before T3 started or validated.

- Remember
  - For a previously validated transaction U that did not finish before T started we need to check RS(T) intersect WS(U). (When Fin(U) > Start(T))
  - For a previously validated transaction U not finished before T validated we need to check WS(T) intersect WS(U). (When Fin(U) > VAL(T))

Validation

- Question 2

R1(A,B), R2(B,C), R3(C), V1, V2, V3, W1(C), W2(B), W3(A)

- What happens when this schedule is processed by a validation-based scheduler?
  - Does T1 validate?
    - Yes, nothing to check since it is the first to validate
  - Does T2 validate? T1 did not finish before T2 started or validated.
    - RS(T2) intersect WS(T1) = {C} → rollback T2
  - Does T3 validate? T1 did not finish before T3 started or validated.
    - RS(T3) intersect WS(T1) = {C} -> rollback T3

- Remember
  - For a previously validated transaction U that did not finish before T started we need to check RS(T) intersect WS(U). (When Fin(U) > Start(T))
  - For a previously validated transaction U not finished before T validated we need to check WS(T) intersect WS(U). (When Fin(U) > VAL(T))

Relational Algebra

- **Query language** associated with relational model

Relational Algebra (1/3)

Five basic operators:

- Union ($\cup$) and Set difference ($-$)
- Selection: : $\sigma_{condition}(S)$
  - Condition is Boolean combination ($\wedge$,$\vee$) of terms
  - Term is: attribute op constant, attr. op attr.
  - Op is: <, <=, =, ≠, >=, or >
- Projection: $\pi_{list-of-attributes}(S)$
- Cross-product or cartesian product ($\times$)

Relational Algebra (2/3)

Derived or auxiliary operators:

- Intersection ($\cap$), Division (R/S)
- Join: $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$
- Variations of joins
  - Natural, equijoin, theta-join
  - Outer join and semi-join
- Rename $\rho_{B1,\ldots,Bn}(S)$

Relational Algebra (3/3)

Extensions for bags

- Duplicate elimination: δ
- Group by: γ [Same symbol as aggregation]
  - Partitions tuples of a relation into “groups”
- Sorting: τ

Other extensions

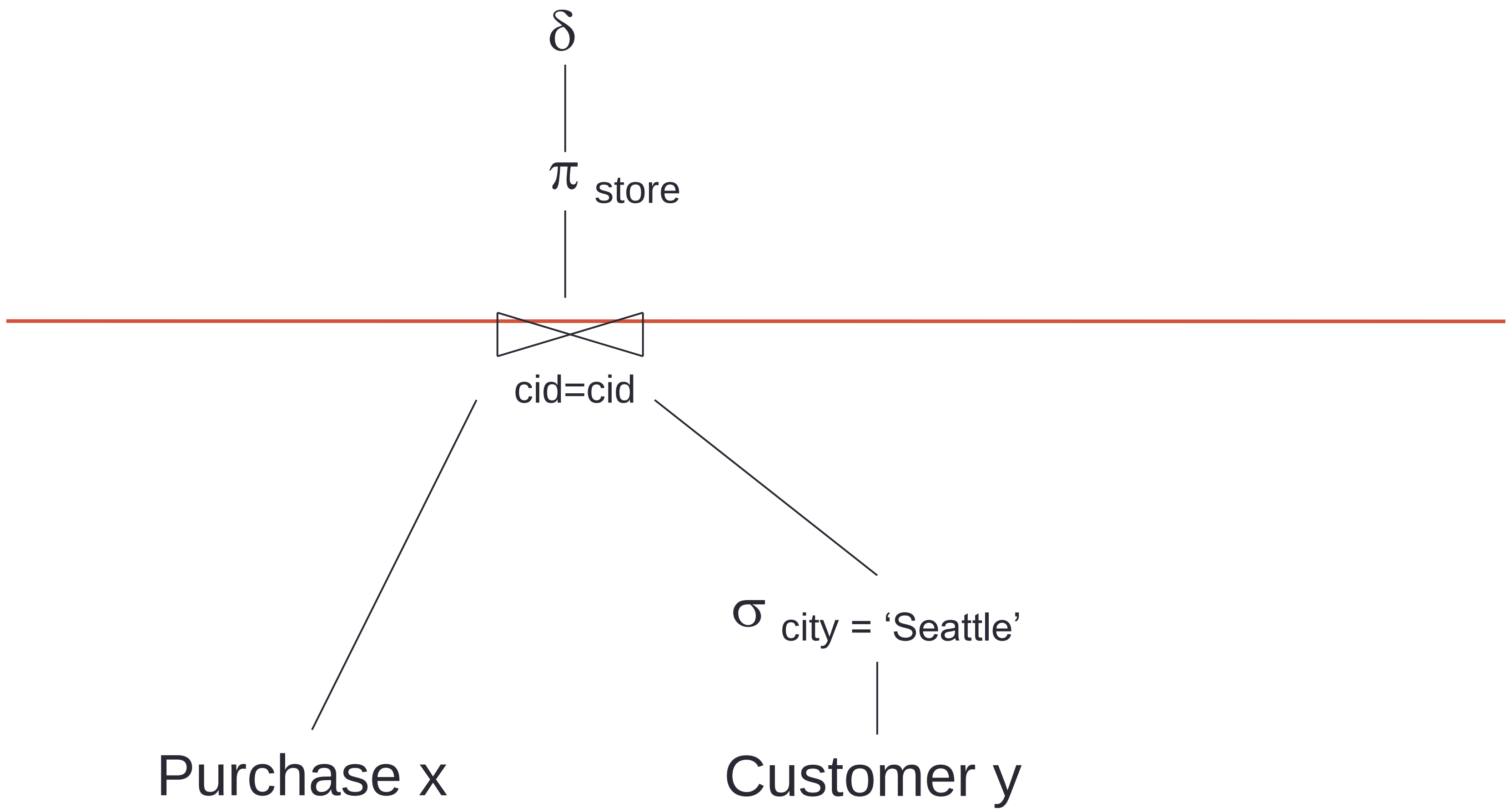
- Aggregation: γ (min, max, sum, average, count)

Relational Algebra

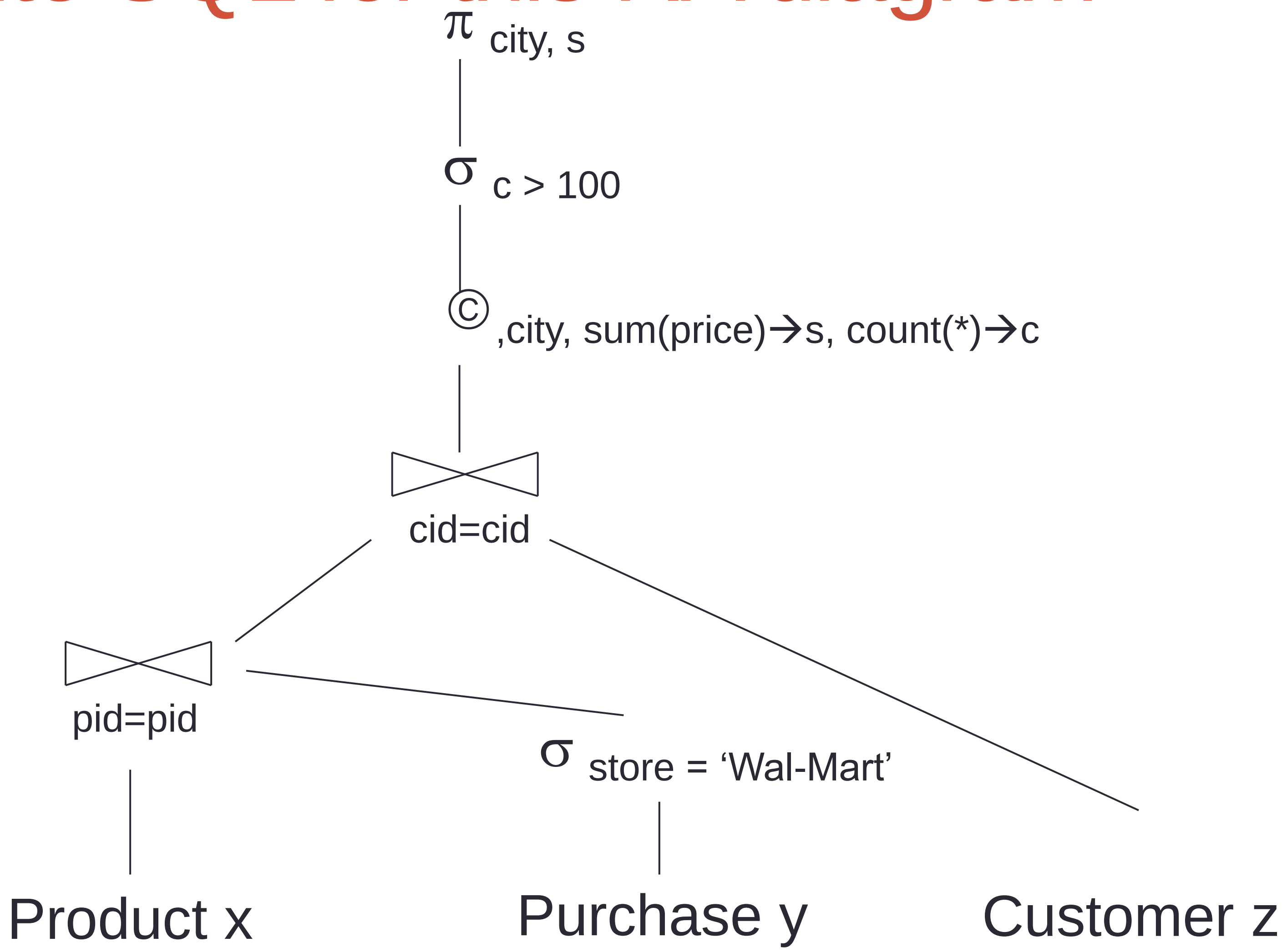
- Warm-up!

Draw relational algebra query plan

```
SELECT DISTINCT x.store
FROM Purchase x, Customer y
WHERE x.cid = y.cid
and y.city = 'Seattle'
```


δ
π store
cid=cid
σ city = ‘Seattle’
Purchase x
Customer y

Write SQL for this RA diagram



Relational Algebra to SQL

```
SELECT z.city, sum(x.price)
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid
and y.store = 'Wal-Mart'
GROUP BY z.city
HAVING count(*) > 10
```

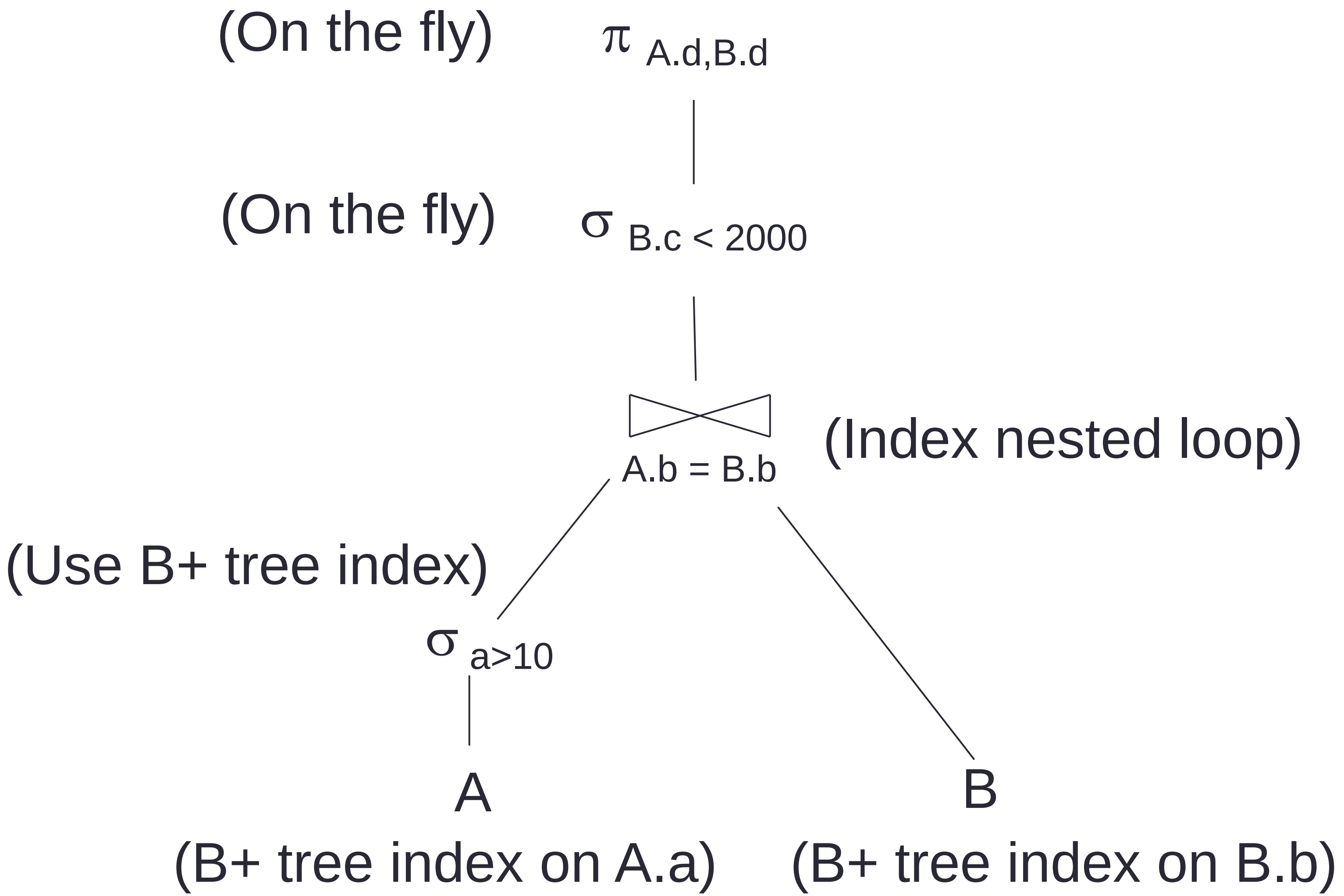
More problems from worksheet

Why is Query Plan B faster?

- #3b from worksheet

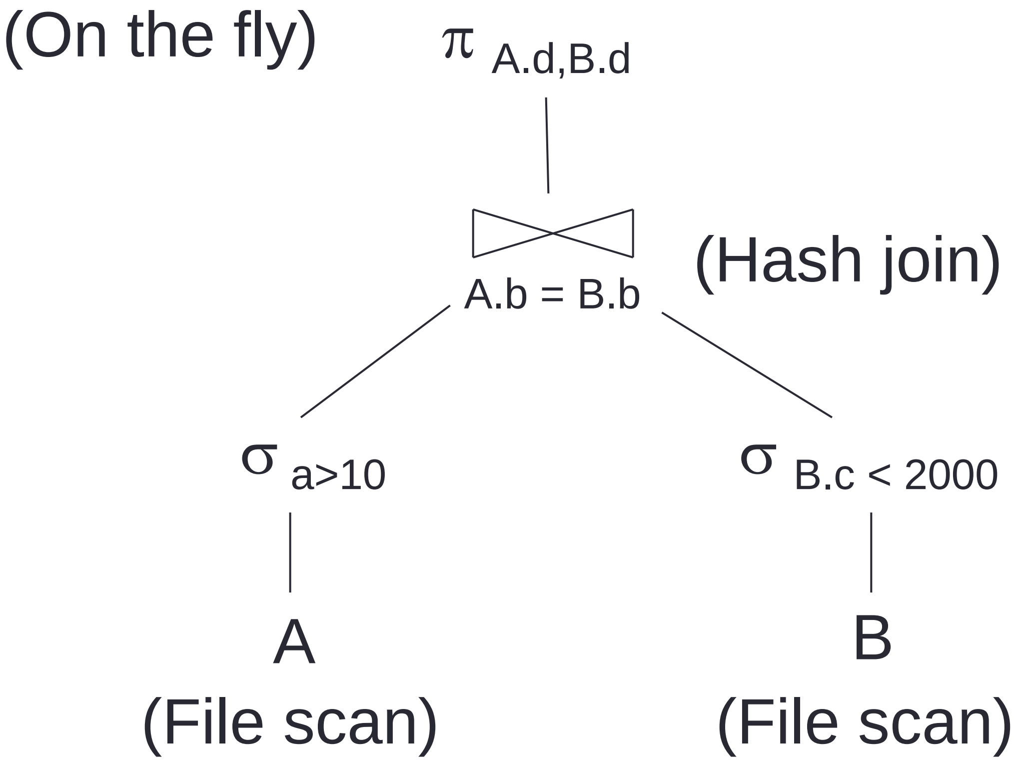
- For solution see http://www.cs.washington.edu/education/courses/cse444/11wi/sections/index.html

Plan A



(On the fly)
π A.d,B.d
(On the fly)
σ B.c < 2000
A.b = B.b
(Index nested loop)
(Use B+ tree index)
σ a>10
A
(B+ tree index on A.a)
B
(B+ tree index on B.b)

Plan B


(On the fly)
π A.d,B.d
A.b = B.b
(Hash join)
σ a>10
σ B.c < 2000
A
(File scan)
B
(File scan)