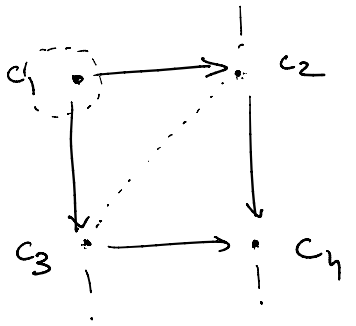
Lecture 22

Thursday, November 5, 2020 4:23 PM

Homework 3



$[c_2; c_3; c_4]$ ✓

$[c_3; c_2; c_4]$ ✓

$[c_2; c_4; c_3]$ ✗

Key: +3

ATTACK AT DAWN ← Plaintext

DWWDFN DW GDZQ ← Cipher text

] Caeser cipher

+ Key

ATTACKATDAWN

LEMONLEMONLE

LY...

] Vigenere cipher

How to evaluate Relational Algebra

- How to evaluate Cartesian products

To evaluate : $Q_1 * Q_2$

Let $T_1 = eval(Q_1)$

$T_2 = eval(Q_2)$

Create a new table T_3

For each $t_1 \in T_1$ $t_2 \in T_2$

Insert $(t_1 @ t_2)$ into T_3

Return T_3

Employees $\bowtie_{E.z=U.z}$ USPS

$= \sigma_{E.z=U.z}$ (Employees $\times$ USPS)

9 intermediate tuples.

Name	zip code
John	90089
David	90015
Mary	19104

Zip code	State
90089	CA
90015	CA
19104	PA

To evaluate Emp $\bowtie_{E.z=U.z}$ USPS

Name	Zip
J	90089
D	90015
M	19104

Z	S
90089	CA
90015	CA
19104	PA

With 20000 employees, 20000 zip codes → 40 billion intermediate tuples

20000 zip codes

Intermediate tuples with Cartesian products

~ 40 seconds on a modern machine

With targeted joins

Create 20000 tuples.

20000 * log(20000) operations

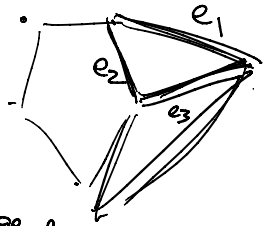
# of employees. operations to lookup zip code

~300 K operations

≤ 1 second.

Question: How do we evaluate joins quickly?

Triangle Query



select n_1 n_2 n_3

from edge e_1 edge e_2 edge e_3

where $e_1 \cdot v_1 = e_2 \cdot v_2$ and

$e_2 \cdot v_1 = e_3 \cdot v_2$ and

$e_3 \cdot v_1 = e_1 \cdot v_2$

$n_1 = e_1 \cdot v_1$, $n_2 = e_2 \cdot v_1$ $n_3 = e_3 \cdot v_1$

There exists algorithm which takes $O(n^2)$ time

There exist tables where you output

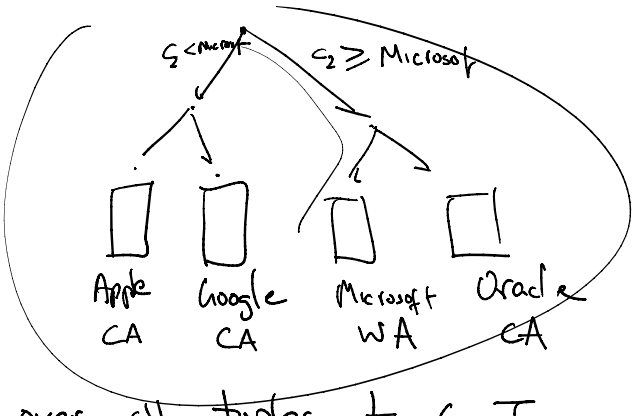
$O(n^{1.7})$ tuples

Inner Next Turn

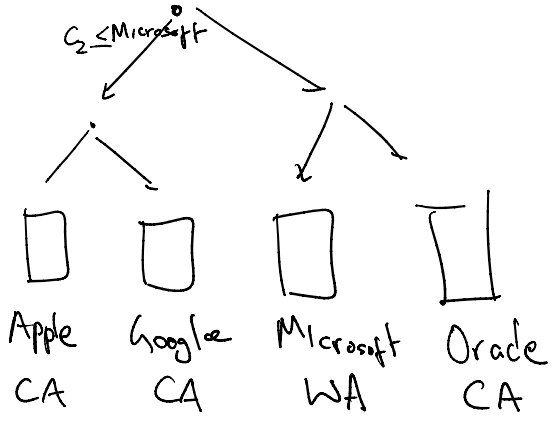
Loop Nest Join

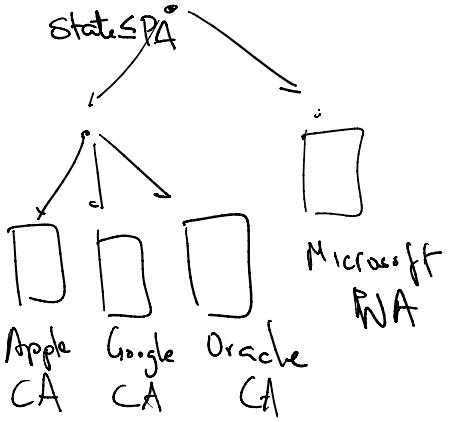
$$T_1 \bowtie_{T_1.c_1 = T_2.c_2} T_2$$

- Construct an index on $T_2.c_2$



- Loop over all tuples $t_1 \in T_1$
- Consult the column $t_1.c_1$
- Find if a corresponding entry exists in T_2
- $O(|T_1| \cdot \log(|T_2|) + |T_2|)$ ← To construct the index





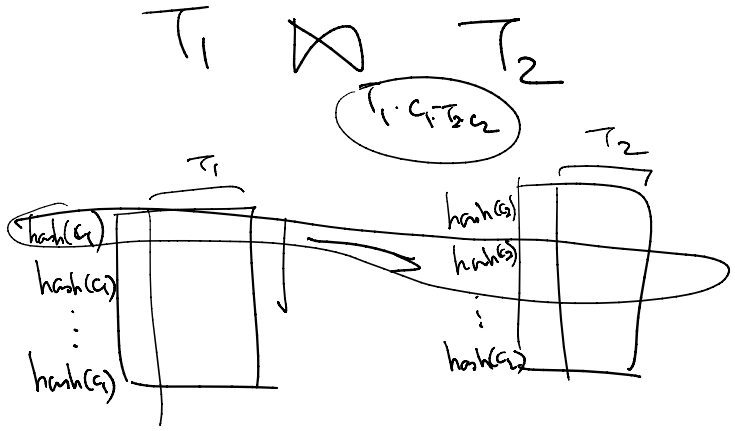
Apple	Google	Microsoft	Oracle
CA	CA	WA	CA

CA	CA	CA

"All companies named Microsoft".

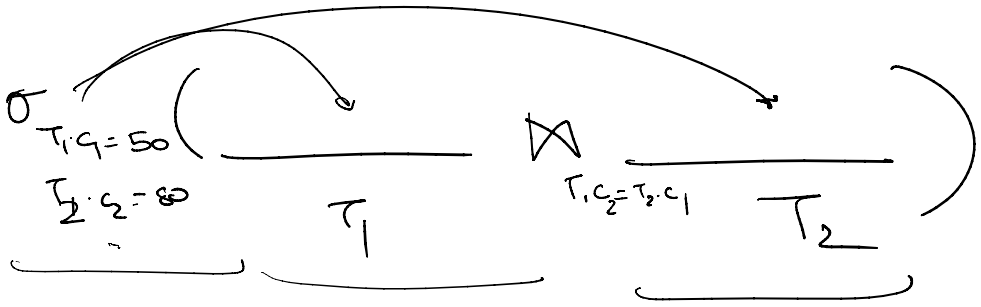
"All companies headquartered in WA".

Hash join



$T_1.c_1 < T_1.c_2$

Two optimizations



2000 tuples

2000 tuples

30000 tuples

30000 units of time. + (2000 log(|2000|) units

30000 units of time + (2000 log(|2000|) units)

"Filter them first"

$\sigma_{\substack{T_1 \cdot c_1 = 50 \\ T_2 \cdot c_2 = 80}} \left(T_1 \bowtie_{T_1 \cdot c_2 = T_2 \cdot c_1} T_2 \right)$

$\Rightarrow \left(\sigma_{T_1 \cdot c_1 = 50} T_1 \right) \bowtie_{T_1 \cdot c_2 = T_2 \cdot c_1} \left(\sigma_{T_2 \cdot c_2 = 80} T_2 \right)$

2000 units of time

40 tuples, say

2000 units of time

30 tuples, say.

1200 $\simeq$ 1500 operations.

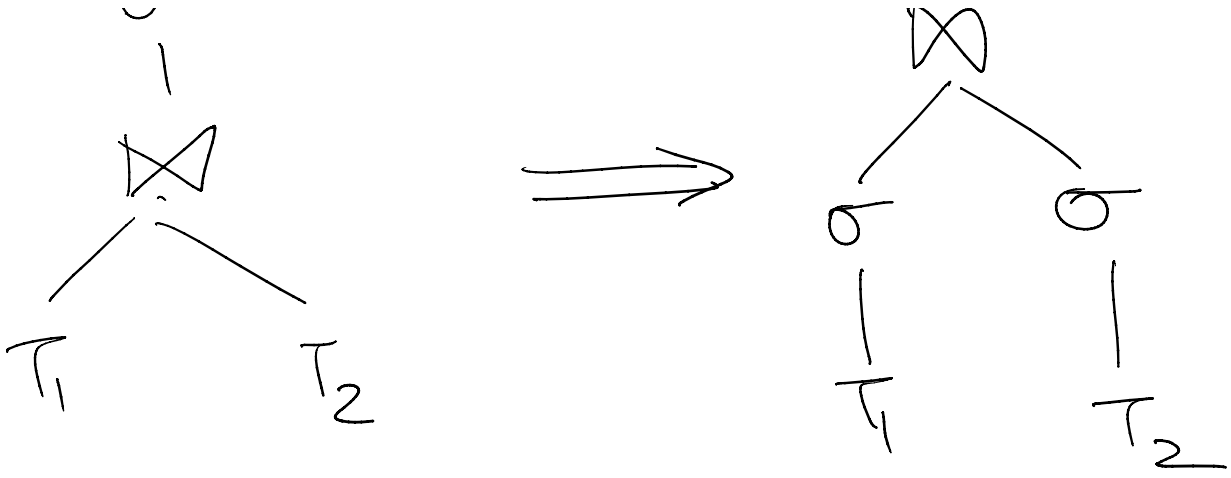
5500 operations

"Selection push down" / Distributivity.

σ

1

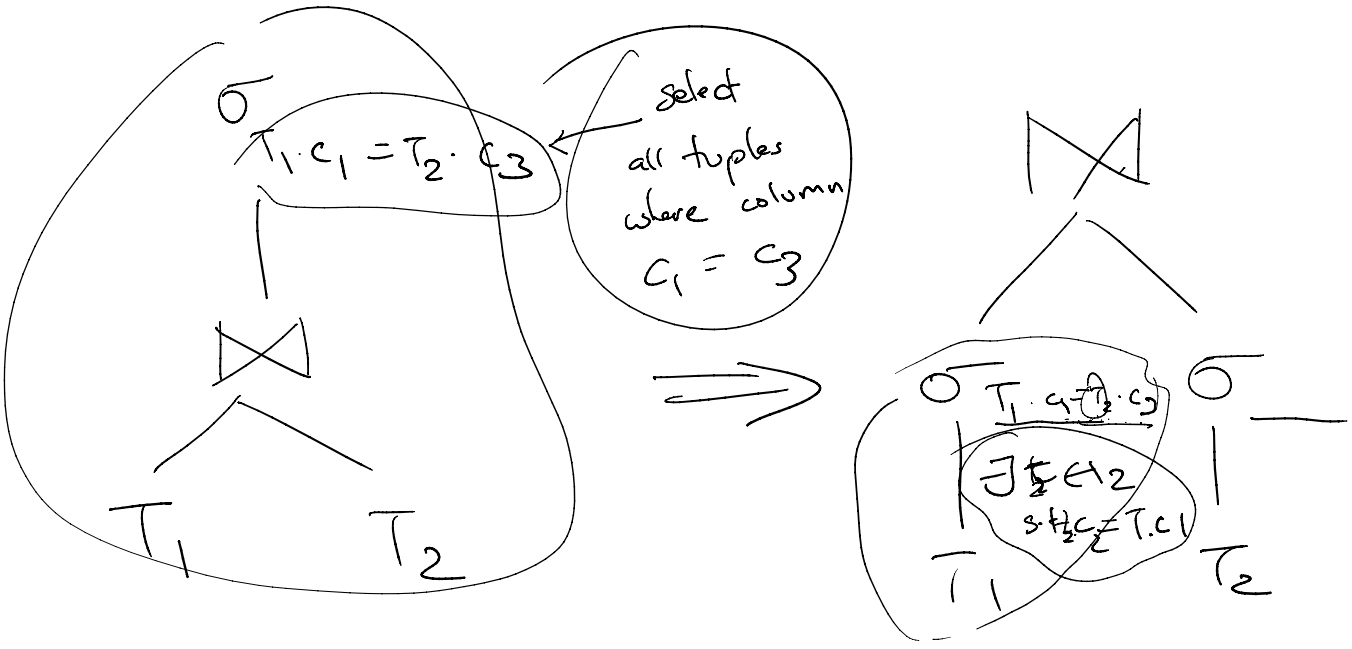
$\bowtie$



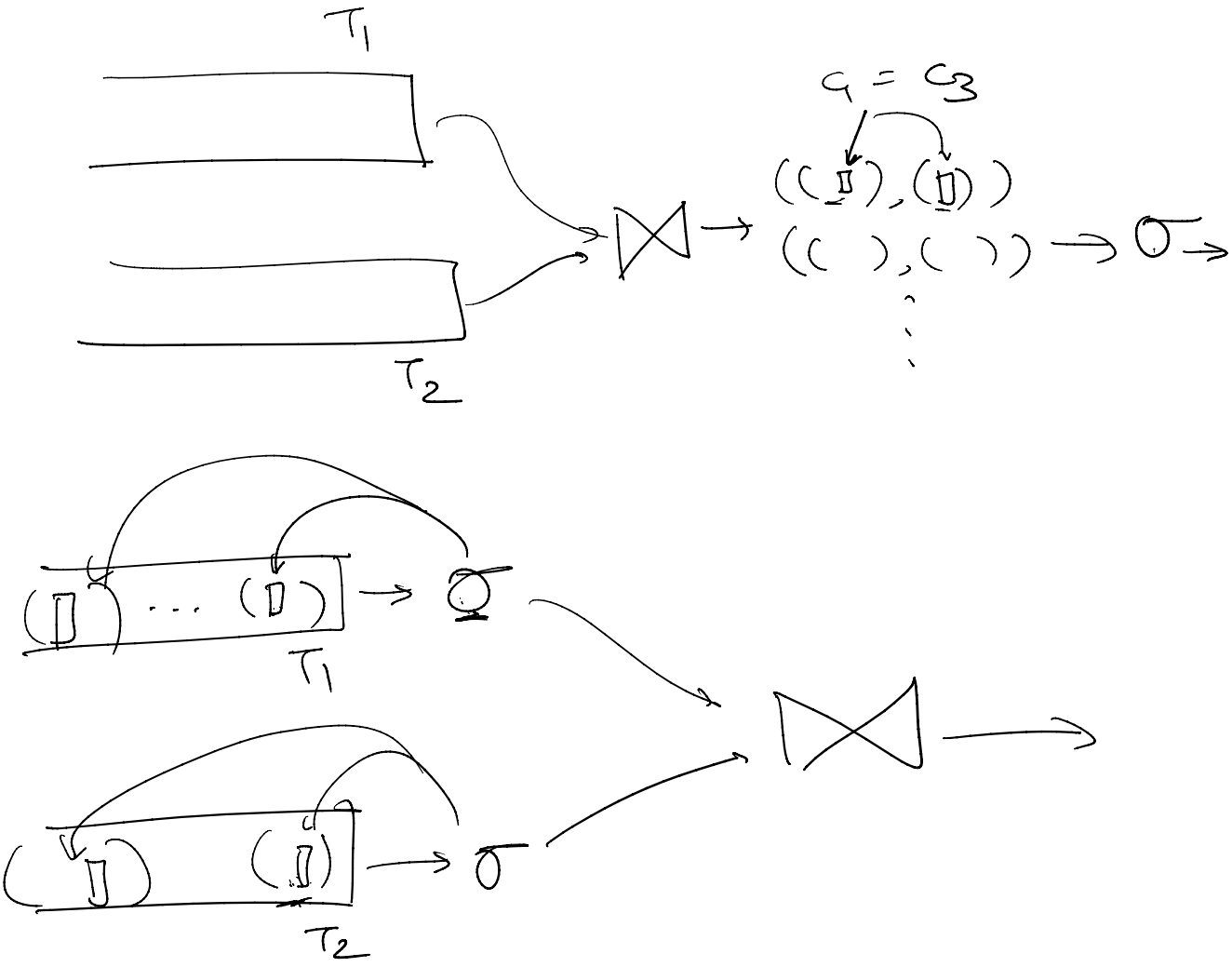
performance improvement

Question 1: Why does this optimization work?

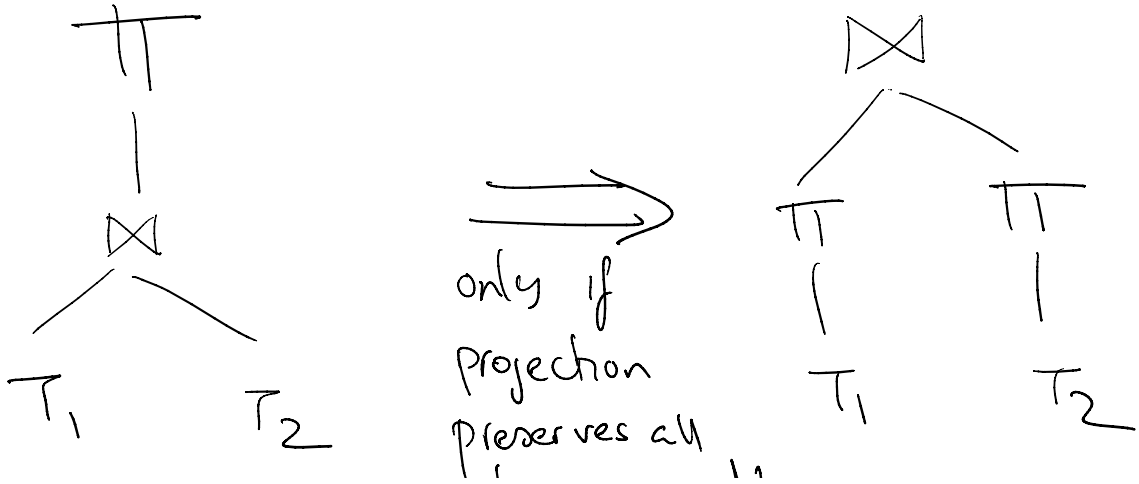
Question 2: Does it always make sense?



Selection pushdown works only if it is locally filtering the elements of the table.



Projection pushdowns



T_1 T_2 Π preserves all columns needed for join. l_1 l_2 ↑ reduces memory.

Question 1: Is this a performance improvement?

Question 2: Does it always make sense. (correctness)

Question 1a: Does projection pushdown preserve the number of tuples?

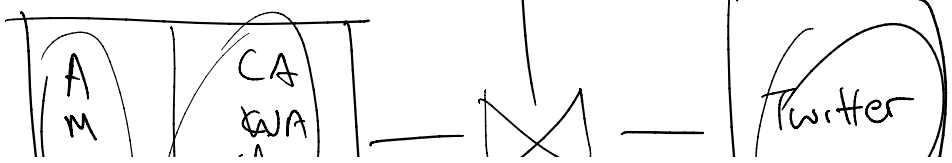
Company	State
Apple	CA
Microsoft	WA
Oracle	CA
Twitter	CA

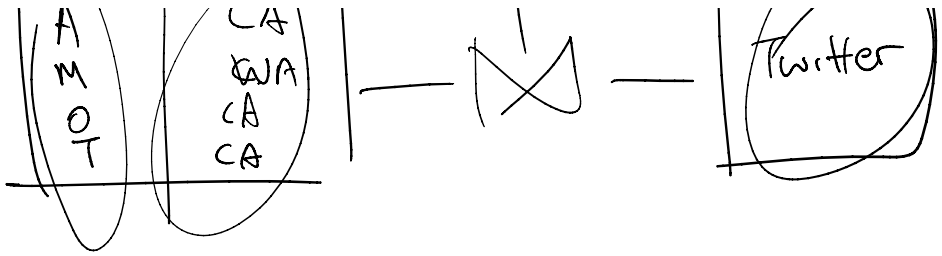
$\Pi_2 \Rightarrow$

State
CA
WA

Set semantics.

C S Π_3 Social media





"All states which have the headquarters of a social network."

Things not discussed

1. Query planning: which operations to execute & in what order

$$(T_1 \bowtie T_2 \bowtie T_3 \cdots \bowtie T_k)$$

$$T_1 \bowtie T_2 \quad \bowtie T_3$$

2000 2000 50

2. Rewriting strategies: selection pushdowns
projection pushdowns.

"Declarative programming"

How do you describe reachability in a graph?

edge