Faster joins using sorting

Overview

In the last video, we saw how to do (Naïve) loop join to handle joins, e.g. natural join

Here, we will see a more advanced technique called sort join

Can We Go Faster?

Yes, we can!

Equijoins

Equijoin $R \bowtie_{A=B} S$ is defined as $\sigma_{A=B}(R \times S)$

A, B are the **join attributes**

Stores

code	city
12345	1
678910	2

Employees

name	depart
Oscar	12345
Janice	678910
David	678910

Stores $\bowtie_{code=depart}$ Employees

code	city	name	depart
12345	1	Oscar	12345
678910	2	Janice	678910
678910	2	David	678910

If **R** is sorted on **A** *and* **S** is sorted on **B**, *then* $R \bowtie_{A=B} S$ can be computed with one pass over **R** and **S** + run time equal to the size of the output

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**

Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**

Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…
2	2	99	11	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**

Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…
2	2	99	11	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…
2	2	99	11	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…
2	2	99	11	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…
2	2	99	11	…
4	4	52	74	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…
2	2	99	11	…
4	4	52	74	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…
2	2	99	11	…
4	4	52	74	…
4	4	52	9	…

Merging

as in merge sort

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

Running time: $O(|\mathbf{R}| + |\mathbf{S}| + \text{size of output})$

R

A	C	…
1	5	…
1	4	…
2	99	…
3	15	…
4	52	…
…	…	…

sorted

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

sorted

=

A	B	C	D	…
2	2	99	50	…
2	2	99	94	…
2	2	99	11	…
4	4	52	74	…
4	4	52	9	…
…	…	…	…	…

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	...
1	5	...
2	4	...
2	99	...
2	15	...
4	52	...
...	...	...

duplicates

$\bowtie_{A=B}$

S

B	D	...
2	50	...
2	94	...
2	11	...
4	74	...
4	9	...
...	...	...

=

A	B	C	D	...

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**

Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…
2	2	4	94	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…
2	2	4	94	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…
2	2	4	94	…
2	2	4	11	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…
2	2	4	94	…
2	2	4	11	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…
2	2	4	94	…
2	2	4	11	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**

Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…
2	2	4	94	…
2	2	4	11	…
2	2	99	50	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Goal: compute $R \bowtie_{A=B} S$
Assume: R is sorted on A and S is sorted on B

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…
2	2	4	94	…
2	2	4	11	…
2	2	99	50	…

duplicates

Remember tuples in S that match with the current value of A

Merging With Duplicates in Column A

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…
2	2	4	94	…
2	2	4	11	…
2	2	99	50	…
2	2	99	94	…

Remember tuples in S that match with the current value of **A**

Merging With Duplicates in Column A

Runtime: O(|R|+|S|+size of output)

Goal: compute **R** $\bowtie_{A=B}$ **S**
Assume: **R** is sorted on **A** and **S** is sorted on **B**

R

A	C	…
1	5	…
2	4	…
2	99	…
2	15	…
4	52	…
…	…	…

duplicates

$\bowtie_{A=B}$

S

B	D	…
2	50	…
2	94	…
2	11	…
4	74	…
4	9	…
…	…	…

=

A	B	C	D	…
2	2	4	50	…
2	2	4	94	…
2	2	4	11	…
2	2	99	50	…
2	2	99	94	…
…	…	…	…	…

Remember tuples in S that match with the current value of **A**

Faster Joins With Sorting

Sort Join Algorithm:

Compute R $\bowtie_{A=B}$ S:

1. Sort R on A
2. Sort S on B
3. Merge the sorted R and S

Running time: $O(|R| \times \log_2 |R|)$

Running time: $O(|S| \times \log_2 |S|)$

Running time: $O(|R|+|S|+\text{size of output})$

Typical running time: $O(|R|\log_2|R| + |S|\log_2|S|)$

- If not "too many" values in A occur multiple times
- E.g., this is the case if A is a key

Having a run time depending on the size of output is called output sensitive

Typically much faster than Nested Loop Join

- Same time in the worst case, because output can have size up to $|R| \times |S|$

Remarks

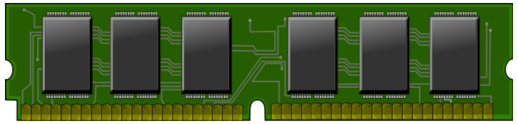
Various **join algorithms** in practice:

- Index joins
- Hash joins
- Multiway joins: join more than two relations at once

Can compute **other operations of relational algebra** using similar methods as those in this lecture

We've neglected that relations are stored on disk

Running Time vs Disk Accesses

fast  ⟷  slow

Relevant parameters:

- **B** = size of a disk block (typically 512→4096 bytes)
- **M** = number of disk blocks that fit into available RAM

Algorithm	No. of elementary operations	No. of disk accesses
Reading a relation **R**	$O(\lvert R \rvert)$	$O\left(\frac{\lvert R \rvert}{B}\right)$
Sorting **R** on attribute **A**	$O(\lvert R \rvert \log_2 \lvert R \rvert)$	$O\left(\frac{\lvert R \rvert}{B} \log_M \frac{\lvert R \rvert}{B}\right)$

External memory merge sort

Summary

We saw one approach to do faster joins, namely sort joins

Sort join has run time close to linear (i.e. like sort) + the size of the output