Updated 1/20/2023

Topic 1: Data models and query languages
Unit 1: SQL (continued)
Lecture 4

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp23)

https://northeastern-datalab.github.io/cs7240/sp23/

1/20/2023

Pre-class conversations

- Last class summary
- Scribe clarification: 1 week vs end of semester

- Today:
  - SQL continued
  - Logic (our foundation)

PRELIMINARY

CS 7240: Topics and approximate agenda (Spring'23)

This schedule will be updated regularly as the class progresses. Check back frequently. I will usually post lecture slides by the end of the day following a lecture (thus the next day). I post them here on this website (or in Canvas if I think they are not yet ready to be released in public). Please also check our DATA lab seminar for talks of interest.

Topic 1: Data Models and Query Languages

- **Lecture 1 (Tue 1/10):** Course introduction / T1-U1 SQL / PostgreSQL setup / SQL Activities
- **Lecture 2 (Fri 1/13):** T1-U1 SQL
- **Lecture 3 (Tue 1/17):** T1-U1 SQL
- **Lecture 4 (Fri 1/20):** T1-U2 Logic & Relational Calculus
- **Lecture 5 (Tue 1/24):** T1-U1 Logic & Relational Calculus
- **Lecture 6 (Fri 1/27):** T1-U3 Relational Algebra & Codd's Theorem
- **Lecture 7 (Tue 1/31):** T1-U3 Relational Algebra & Codd's Theorem
- **Lecture 8 (Fri 2/3):** T1-U4 Datalog & Recursion
- **Lecture 9 (Tue 2/7):** T1-U4 Datalog & Recursion
- **Lecture 10 (Tue 2/10):** T1-U4 Datalog & Recursion

Topic 2: Complexity of Query Evaluation & Reverse Data Management

- **Lecture 11 (Tue 2/14):** T2-U1 Conjunctive Queries
- **Lecture 12 (Fri 2/17):** T2-U1 Conjunctive Queries
- **Lecture 13 (Tue 2/21):** T2-U2 Beyond Conjunctive Queries
- **Lecture 14 (Fri 2/24):** T2-U3 Provenance
- **Lecture 15 (Tue 2/28):** T2-U3 Provenance
- **Lecture 16 (Fri 3/3):** T2-U4 Reverse Data Management

Topic 3: Efficient Query Evaluation & Factorized Representations

- Spring break (Tue 3/7, Fri 3/10: Northeast Database day 2023 @ Northeastern)
- **Lecture 17 (Tue 3/14):** T3-U1 Acyclic Queries
- **Lecture 18 (Fri 3/17):** T3-U1 Acyclic Queries
- **Lecture 19 (Tue 3/21):** T3-U2 Cyclic Queries
- **Lecture 20 (Fri 3/24):** T3-U2 Cyclic Queries
- **Lecture 21 (Tue 3/28):** T3-U2 Cyclic Queries
- **Lecture 22 (Fri 3/31):** T3-U2 Cyclic Queries
- **Lecture 23 (Tue 4/4):** T3-U3 Factorized Representations
- **Lecture 24 (Fri 4/7):** T3-U4 Optimization Problems & Top-k
- **Lecture 25 (Tue 4/11):** T3-U4 Optimization Problems & Top-k

Topic 4: Normalization, Information Theory & Axioms for Uncertainty

- **Lecture:** Normal Forms & Information Theory
- **Lecture:** Axioms for Uncertainty

Topic 5: Linear Algebra & Iterative Graph Algorithms

- **Lecture:** Graphs & Linear Algebra
- **Lecture:** Computation Graphs

Project presentations

- **Lecture 26 (Fri 4/14): P4 Project presentations**
- **Lecture 27 (Tue 4/18): P4 Project presentations**

Outline: T1-U1: SQL

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k
  - [Recursion: moved to T1-U4: Datalog]

Sorting & Top-*k* evaluation with SQL

R

A	B
1	0
2	0
3	0
4	1

S

B	C
0	1
0	1
0	1
0	2

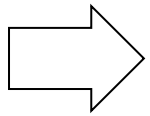
T

C	D
1	1
1	2
2	3
2	4

```
select   A, R.B, S.C, D

from     R, S, T
where    R.B=S.B and S.C=T.C
```

Result

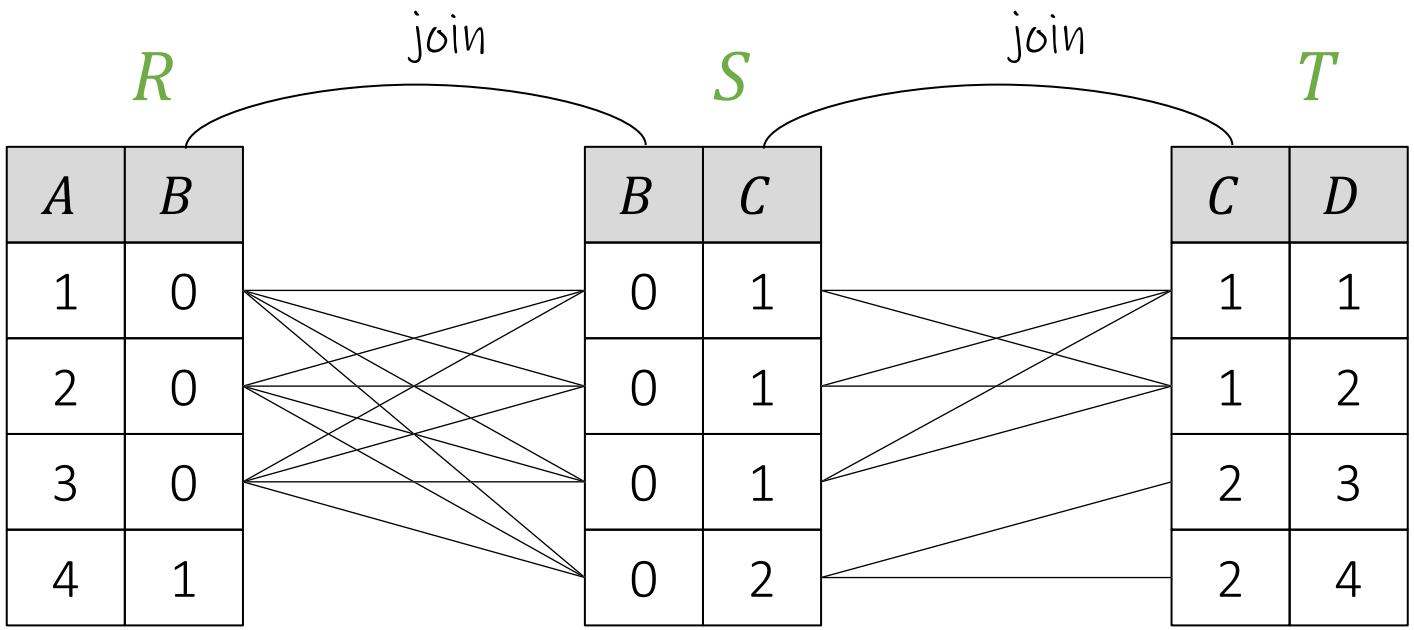


How many results do we get?

?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Sorting & Top-*k* evaluation with SQL




```
select   A, R.B, S.C, D

from     R, S, T
where    R.B=S.B and S.C=T.C
```

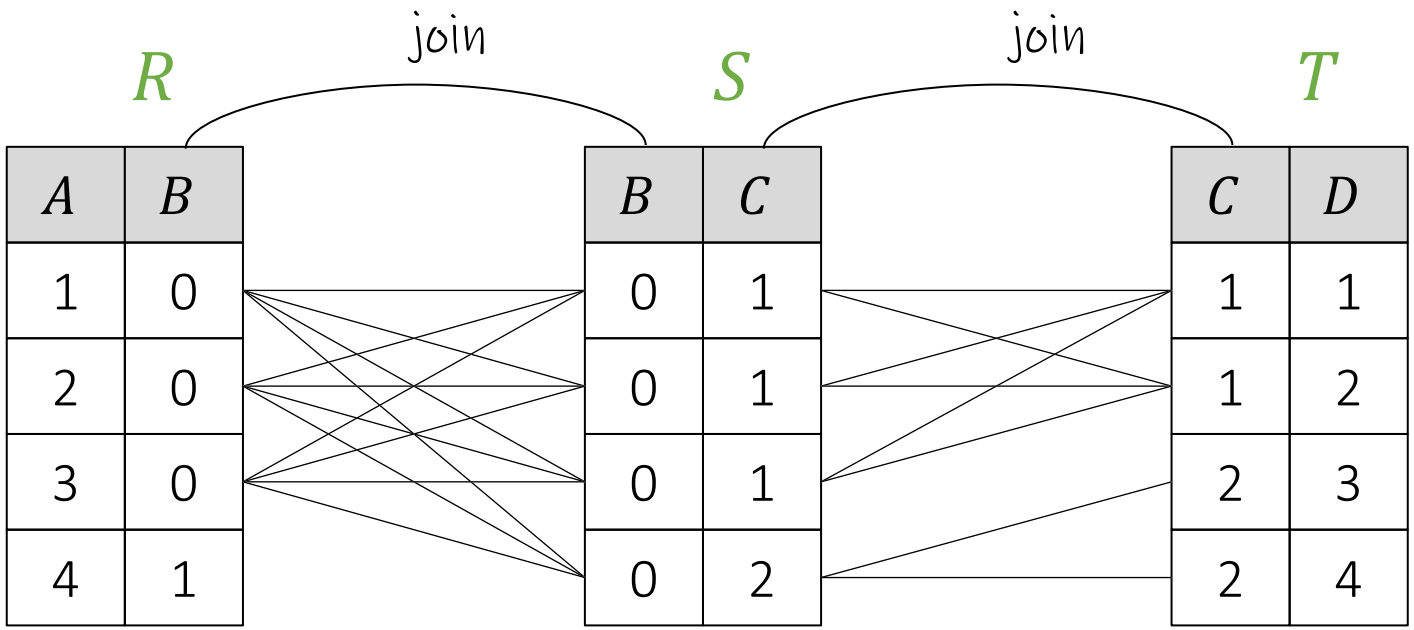
Result

How many results do we get?

?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Sorting & Top-*k* evaluation with SQL




```
select   A, R.B, S.C, D

from     R, S, T
where    R.B=S.B and S.C=T.C
```

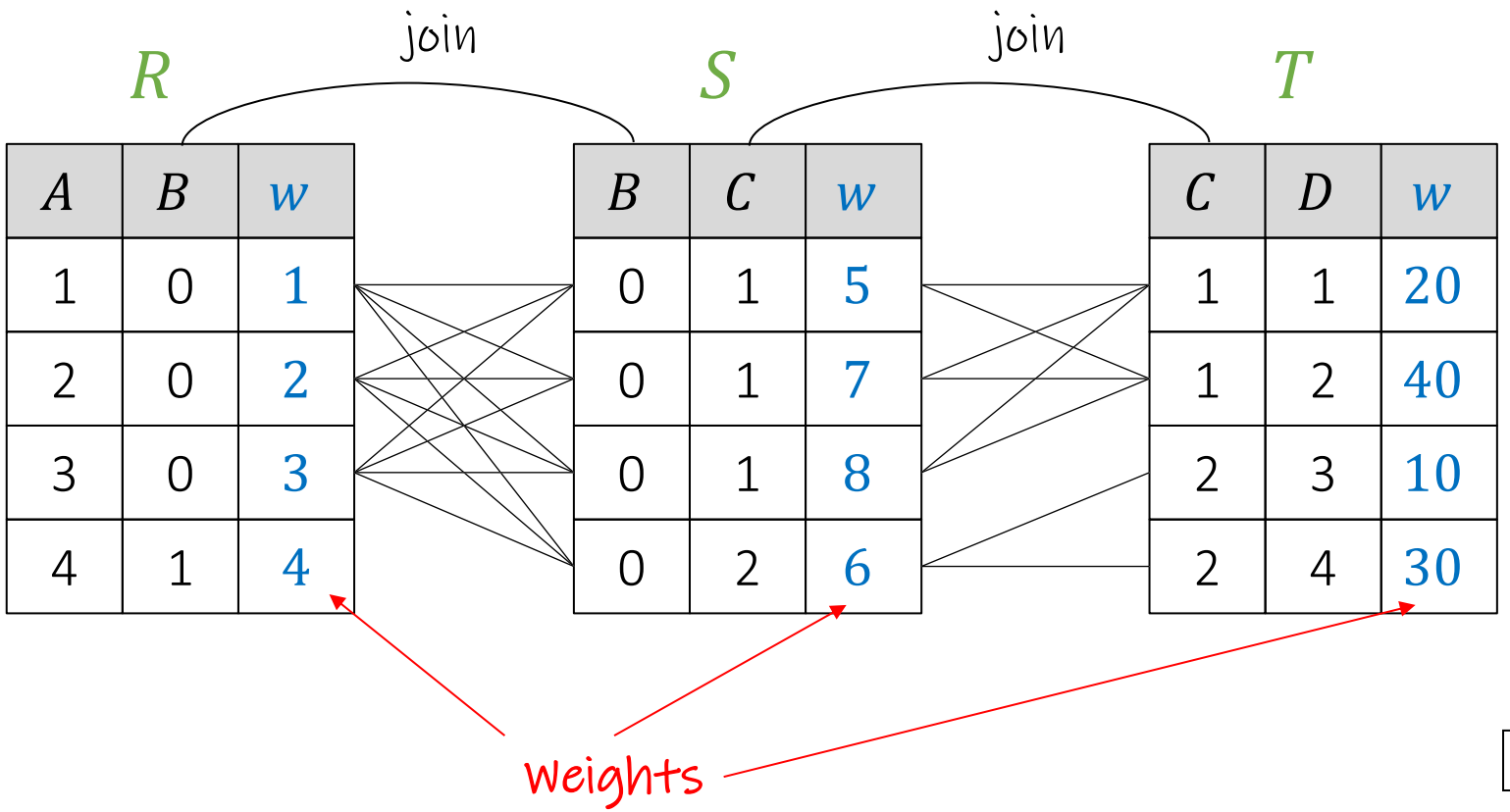
24 total results

Result

A	B	C	D
1	0	2	3
2	0	2	3
3	0	2	3
1	0	1	1
2	0	1	1
3	0	1	1
1	0	1	1
...	...	...	...

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Sorting & Top-*k* evaluation with SQL




```
select   A, R.B, S.C, D,
         R.w + S.w + T.w as weight
from     R, S, T
where    R.B=S.B and S.C=T.C
order by weight ASC
```

Result

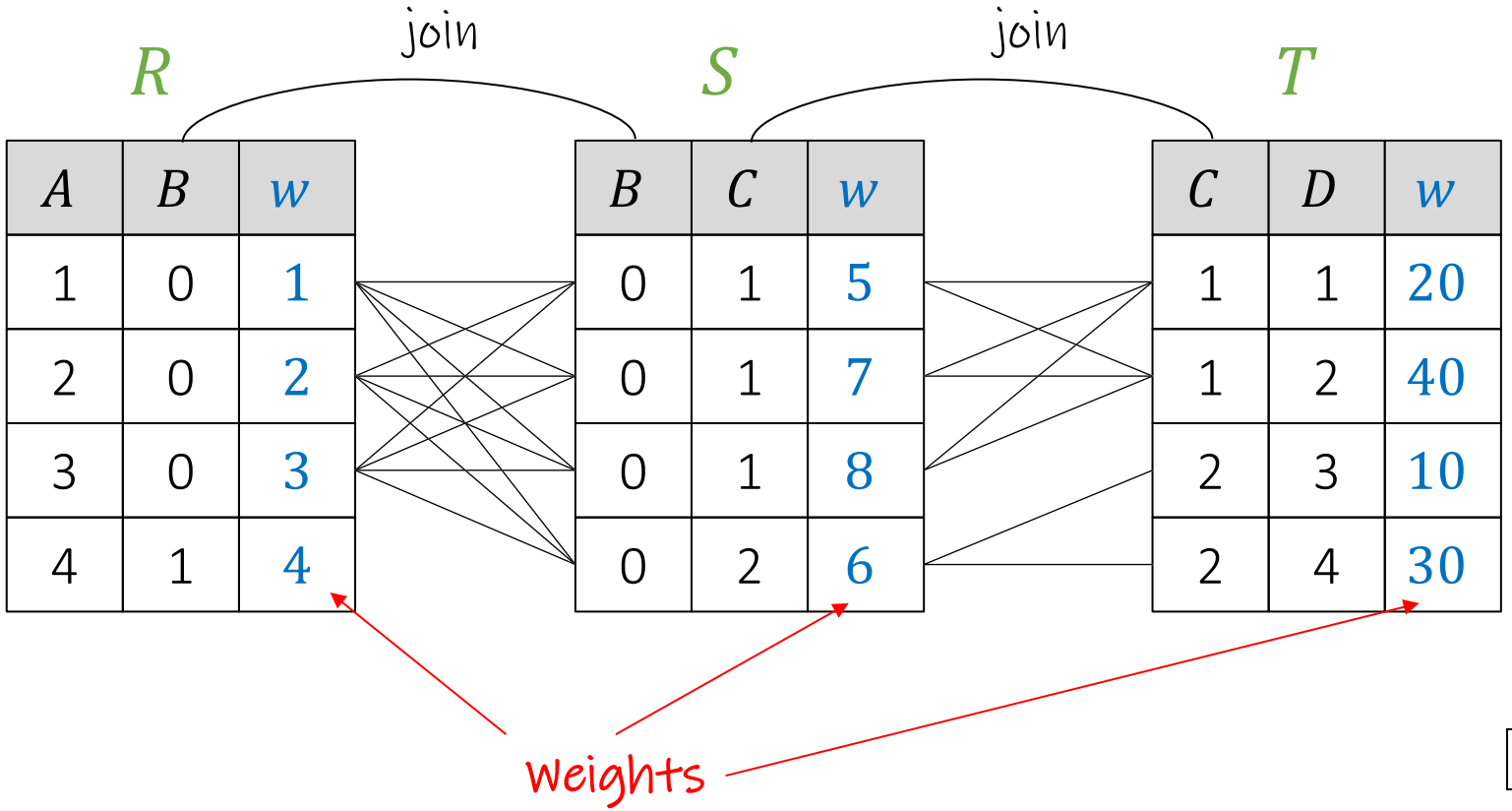
A	B	C	D	weight

What do we get now?

?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Sorting & Top-*k* evaluation with SQL




```
select   A, R.B, S.C, D,
         R.w + S.w + T.w as weight
from     R, S, T
where    R.B=S.B and S.C=T.C
order by weight ASC
```

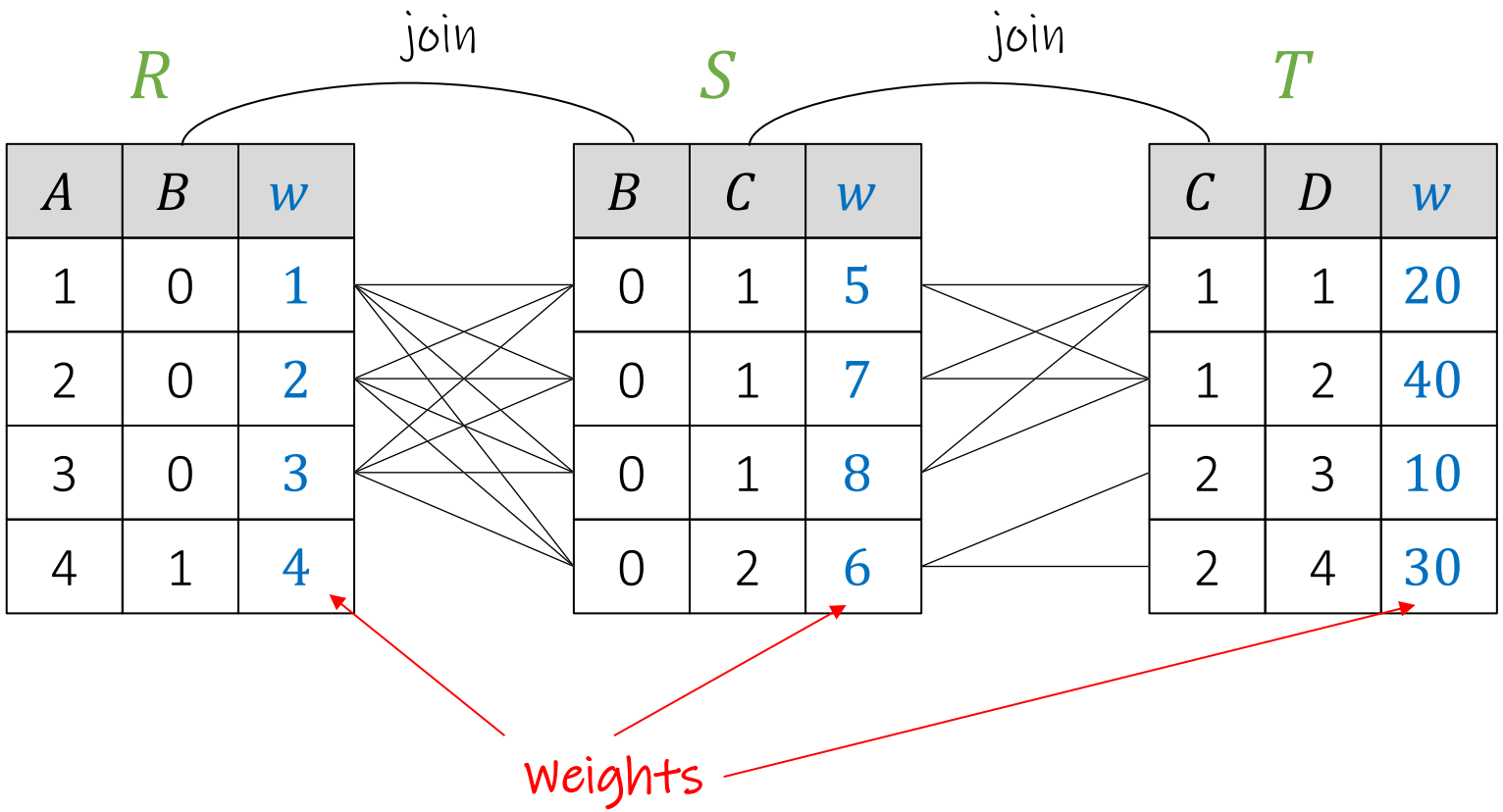
Return all 24 results in order of sum of weights

Result

A	B	C	D	weight
1	0	2	3	17
2	0	2	3	18
3	0	2	3	19
1	0	1	1	26
2	0	1	1	27
3	0	1	1	28
1	0	1	1	28
...	...	...	...	...

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Sorting & Top-*k* evaluation with SQL




```
select   A, R.B, S.C, D,
         R.w + S.w + T.w as weight
from     R, S, T
where    R.B=S.B and S.C=T.C
order by weight ASC
limit    6
```

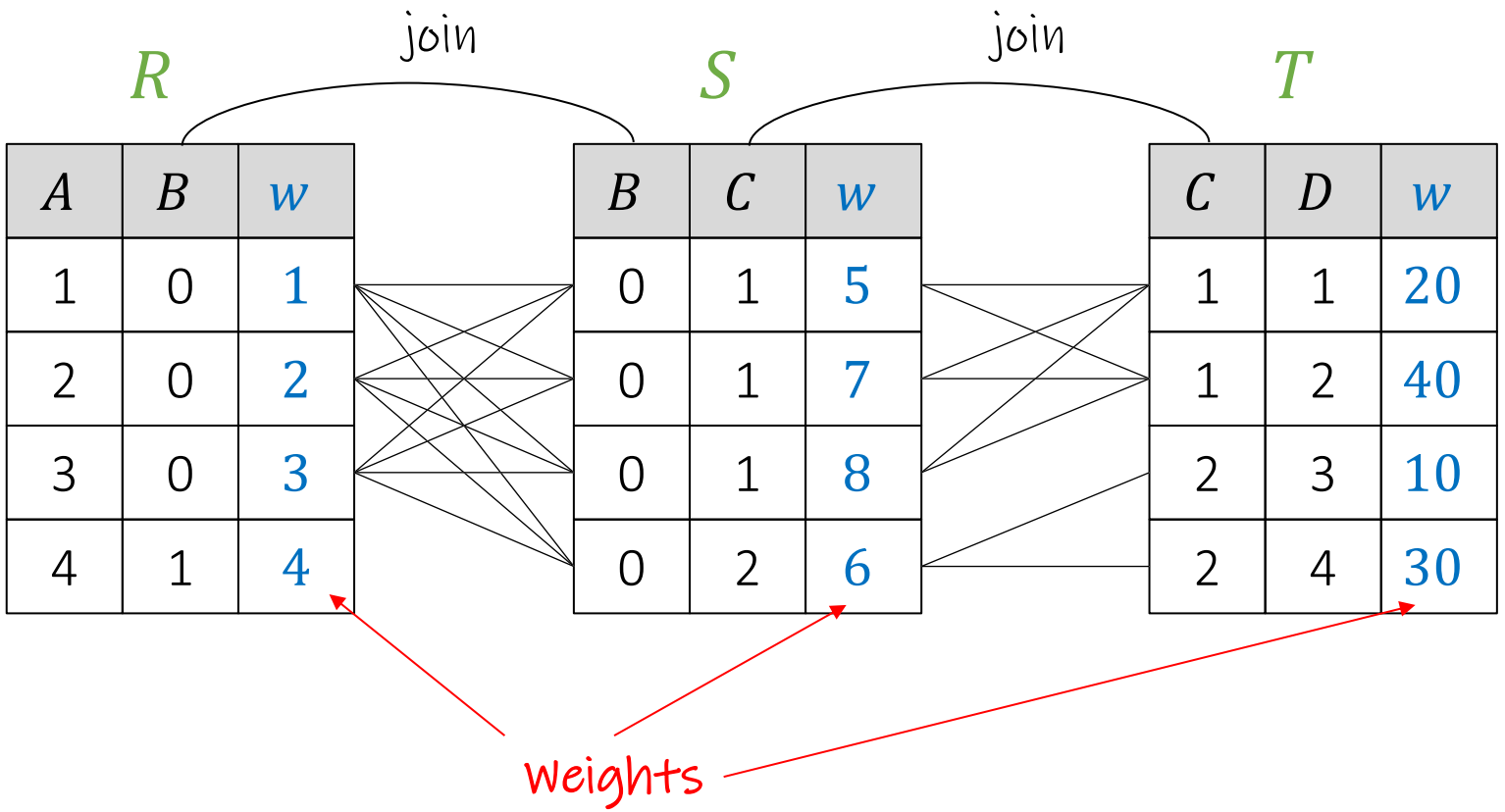
What do we get now?

Result

A	B	C	D	weight
1	0	2	3	17
2	0	2	3	18
3	0	2	3	19
1	0	1	1	26
2	0	1	1	27
3	0	1	1	28
1	0	1	1	28
...	...	...	...	...

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Sorting & Top-*k* evaluation with SQL




```
select   A, R.B, S.C, D,
         R.w + S.w + T.w as weight
from     R, S, T
where    R.B=S.B and S.C=T.C
order by weight ASC
limit    6
```

What do we get now?

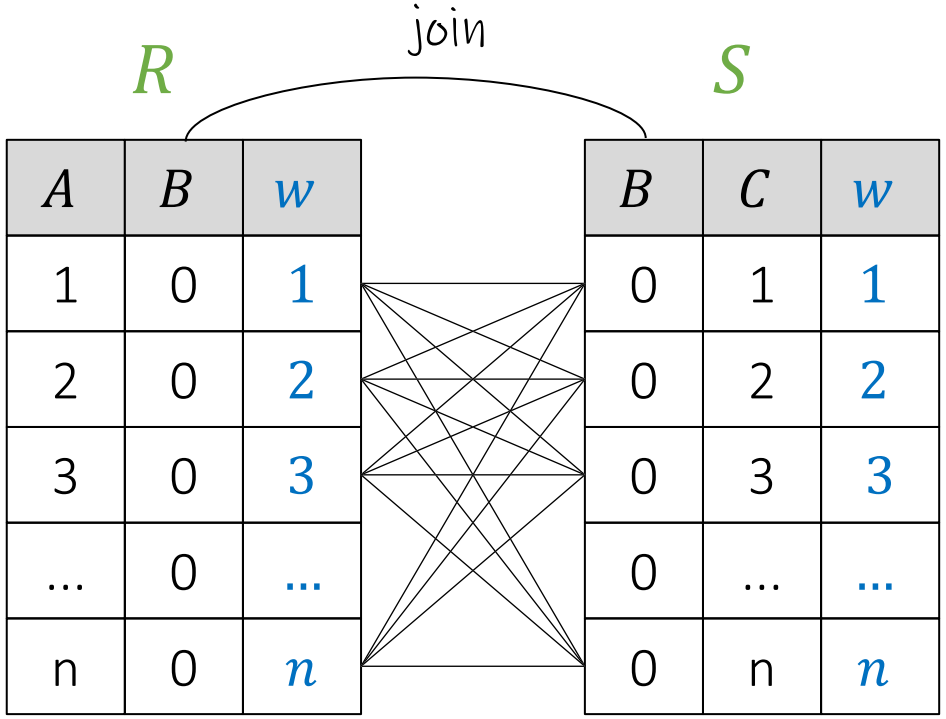
Result

A	B	C	D	weight
1	0	2	3	17
2	0	2	3	18
3	0	2	3	19
1	0	1	1	26
2	0	1	1	27
3	0	1	1	28
1	0	1	1	28
...	...	...	...	...

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Top-k is evaluated inefficiently by modern DBMS's

606



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

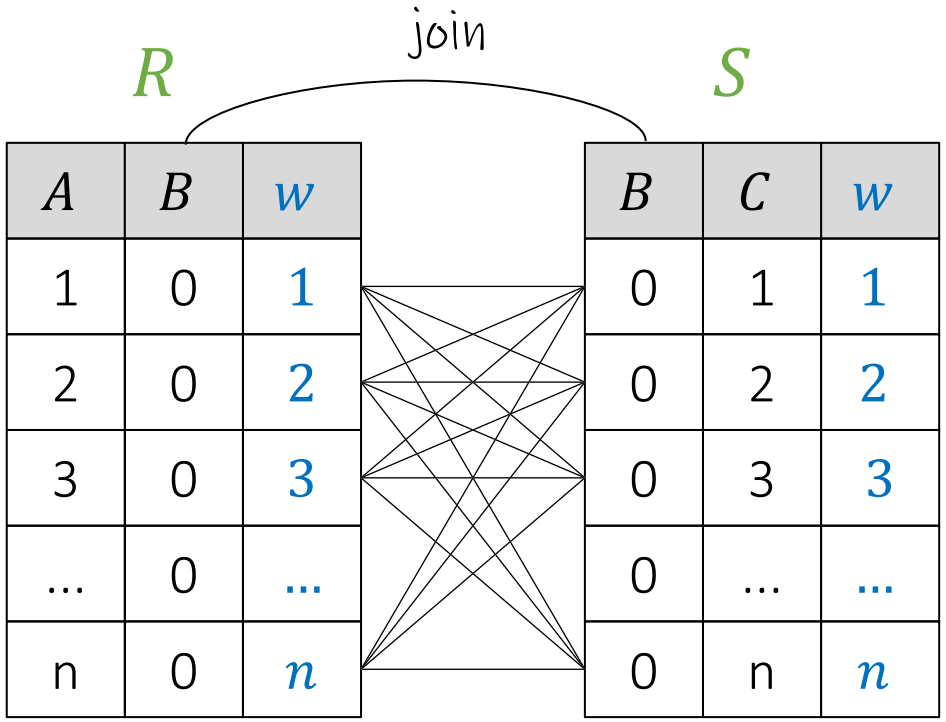
```
select   A, R.B, S.C,
         R.w + S.w as weight
from     R, S
where    R.B=S.B
order by weight ASC
limit    1
```

What will this query return?

?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Top-k is evaluated inefficiently by modern DBMS's



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

```
select   A, R.B, S.C,
         R.w + S.w as weight
from     R, S
where    R.B=S.B
order by weight ASC
limit    1
```

Result

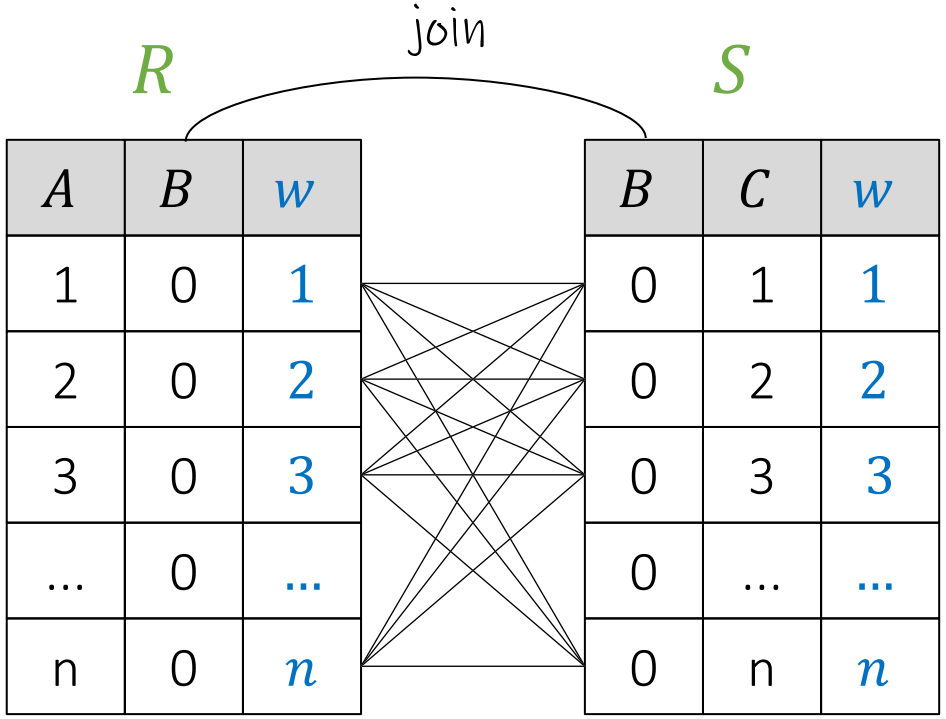
A	B	C	weight
1	0	1	2
1	0	2	3
2	0	1	3
3	0	1	4
2	0	2	4
1	0	3	4
4	0	1	5
…	…	…	…
n	0	n	n*n

Can you see any possible problem of this query as n gets bigger?

?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Top-k is evaluated inefficiently by modern DBMS's



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

```
select   A, R.B, S.C,
         R.w + S.w as weight
from     R, S
where    R.B=S.B
order by weight ASC
limit    1
```

Result

A	B	C	weight
1	0	1	2
1	0	2	3
2	0	1	3
3	0	1	4
2	0	2	4
1	0	3	4
4	0	1	5
…	…	…	…
n	0	n	n*n

n^2 total results. But we are only interested in the top-1.

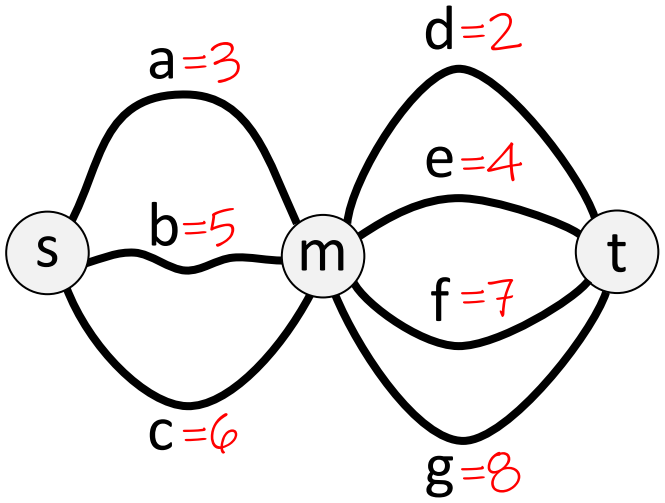
Problem: Database first calculates all n^2 results before sorting.

Question: is there any way to push the sorting behind the join?

?

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

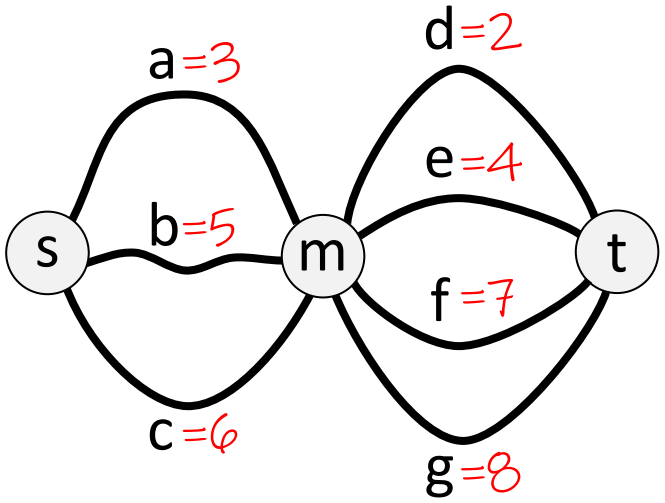
Digression: Distributivity = efficient factorization



What is the shortest path from s to t?

?

Digression: Distributivity = efficient factorization



What is the shortest path from s to t?

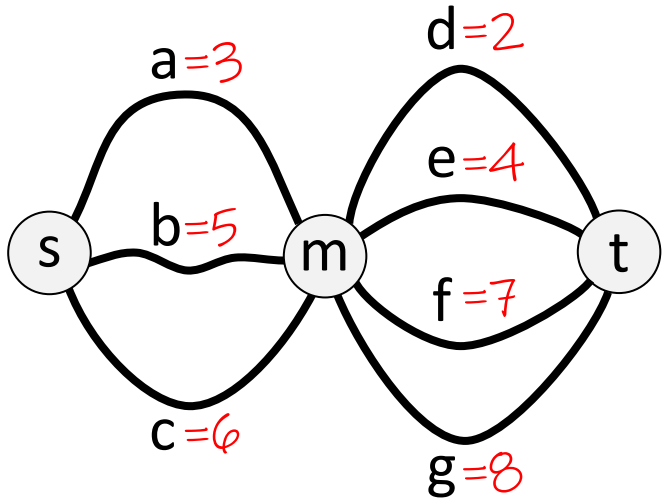
Answer: 5 = 3 + 2

min [a + d, a + e, a + f, a + g, …, c + g]

min[3+2, 3+4, 3+7, 3+8, …, 6+8]

?

Digression: Distributivity = efficient factorization



What is the shortest path from s to t?

Answer: 5 = 3 + 2

Principle of optimality from Dynamic Programming: *irrespective of the initial state and decision, an optimal solution continues optimally from the resulting state*

min [a + d, a + e, a + f, a + g, ..., c + g]

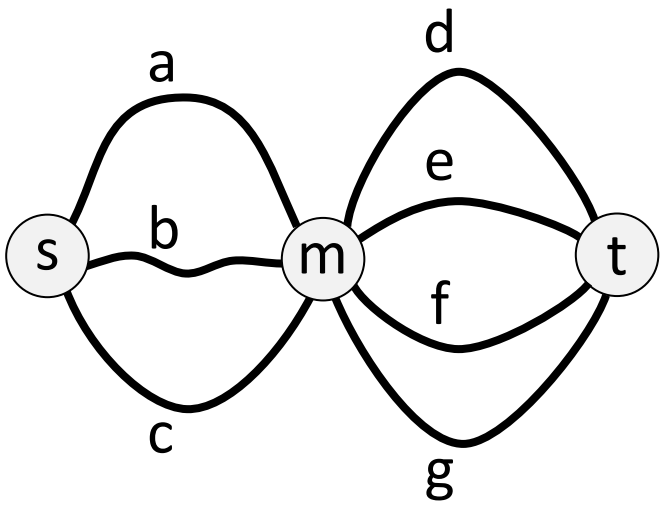
min[3+2, 3+4, 3+7, 3+8, ..., 6+8]

= min [a, b, c] + min [d, e, f, g]

min[3,5,6] + min[2,4,7,8]

min[x,y]+z = min[(x+z), (y+z)]
(+ distributes over min)

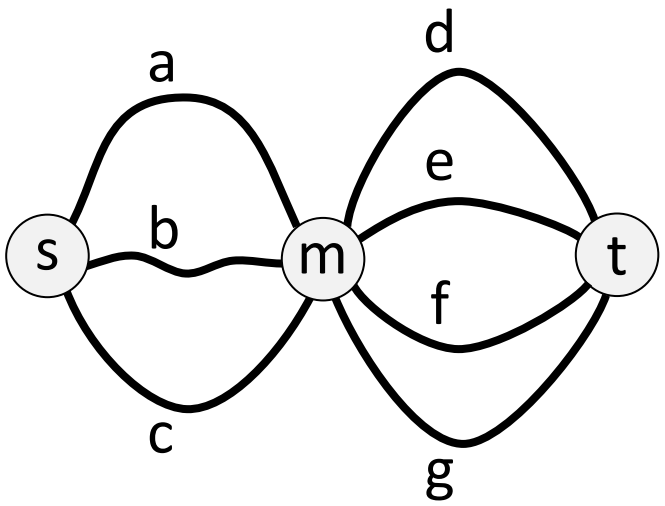
Digression: Distributivity = efficient factorization



How many paths are there from s to t?

?

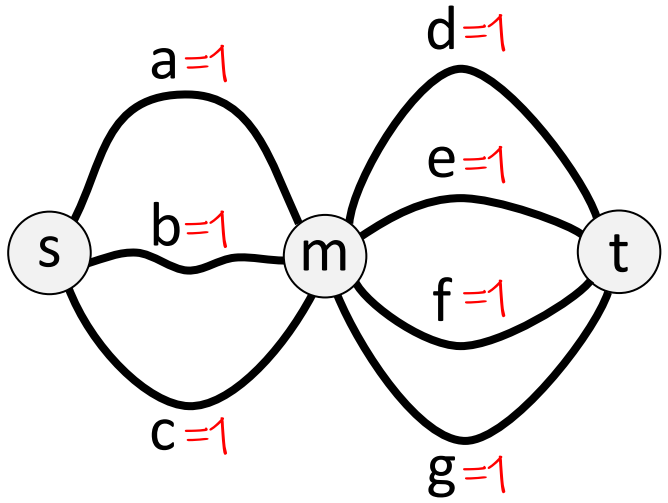
Digression: Distributivity = efficient factorization



How many paths are there from s to t?

Answer: $12 = 3 \cdot 4$

Digression: Distributivity = efficient factorization



How many paths are there from s to t?

Answer: 12 = 3 · 4

The more general algebraic structure behind these two examples are "semirings" (much more on those later in class)

count [a·d, a·e, a · f, a · g, …, c · g]

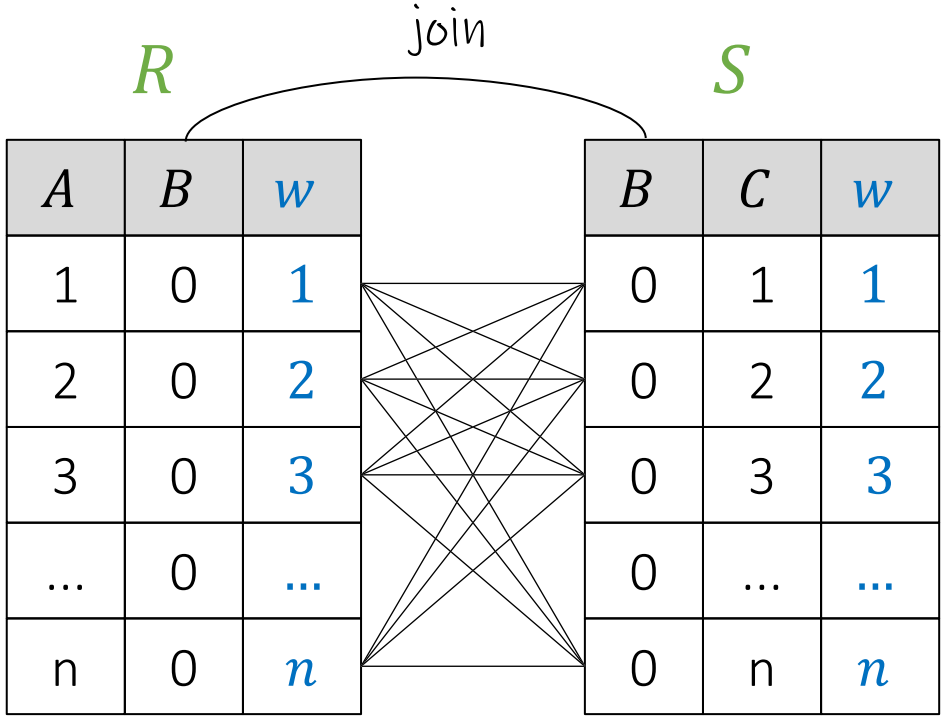
count[1·1, 1·1, 1·1, 1·1, …, 1·1] (12)

= count [a, b, c] · count [d, e, f, g]

count[1,1,1] · count[1,1,1,1]

+[x,y] · z = +[x·z,y·z]
(· distributes over +)

Top-k is evaluated inefficiently by modern DBMS's



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

```
-------------------------
-- Query 1
-------------------------
SELECT   A, R.B, S.C,
            R.W + S.W as weight
FROM      R, S
WHERE    R.B=S.B
ORDER BY weight ASC
LIMIT    1;
```

```
-------------------------
-- Query 2
-------------------------
```

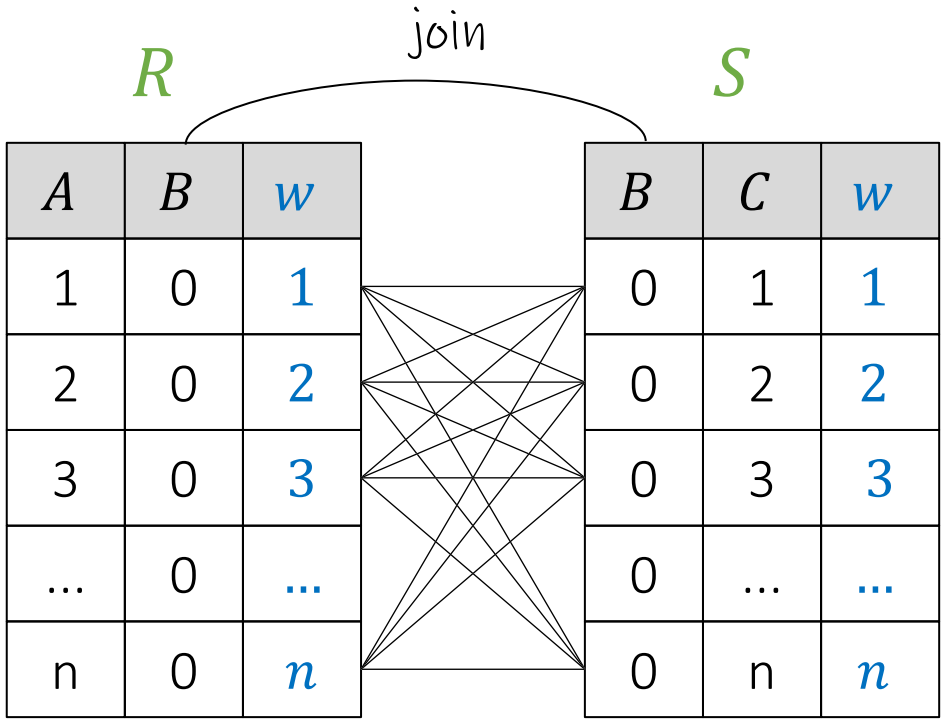
?

n=1000: t_{Q1}= 0.88 sec t_{Q2}=2 msec

n=5000: t_{Q1}=18.6 sec t_{Q2}=8 msec

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Top-k is evaluated inefficiently by modern DBMS's



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

Maximal intermediate result size is O(n) ☺
What is this algorithm called?

?

```
--------------------------
-- Query 1
--------------------------
SELECT   A, R.B, S.C,
           R.W + S.W as weight
FROM      R, S
WHERE    R.B=S.B
ORDER BY weight ASC
LIMIT    1;
```

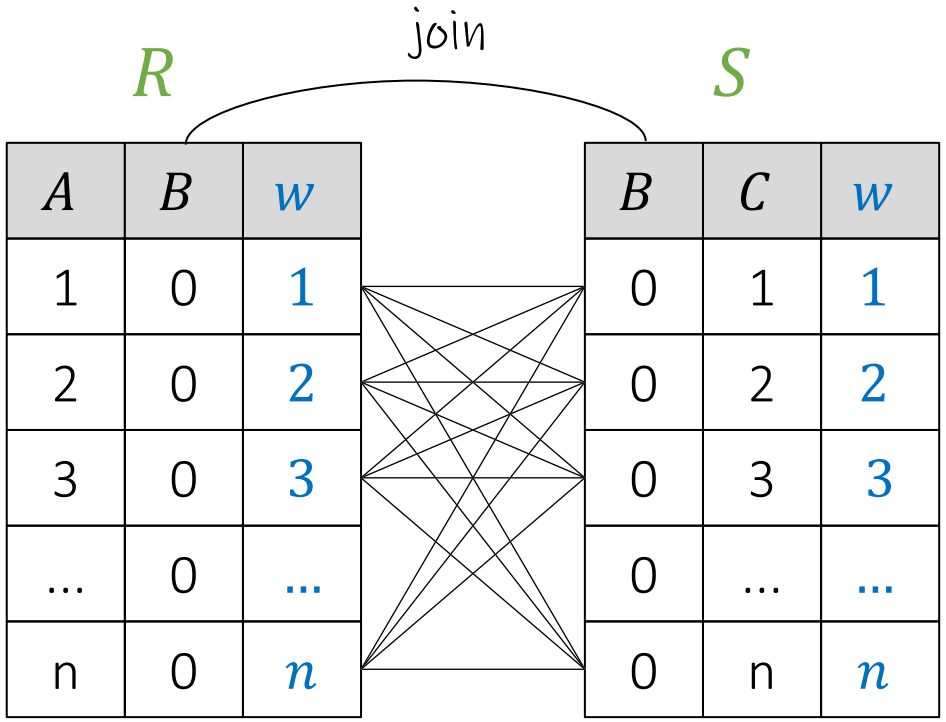
```
--------------------------
-- Query 2
--------------------------
SELECT R.A, X.B, S.C, X.W as weight
FROM R, S,
    (SELECT T1.B, W1, W2, W1+W2 W
    FROM
        (SELECT B, MIN(W) W1
        FROM R
        GROUP BY B) T1,
        (SELECT B, MIN(W) W2
        FROM S
        GROUP BY B) T2
    WHERE T1.B = T2.B
    ORDER BY W ASC
    LIMIT 1) X
WHERE X.B = R.B
AND X.W1 = R.W
AND X.B = S.B
AND X.W2 = S.W
LIMIT   1;
```

n=1000: t_{Q1}= 0.88 sec t_{Q2}=2 msec

n=5000: t_{Q1}=18.6 sec t_{Q2}=8 msec

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Top-k is evaluated inefficiently by modern DBMS's



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

Maximal intermediate result size is O(n) ☺
What is this algorithm called?

Dynamic programming

```
-------------------------
-- Query 1
-------------------------
SELECT   A, R.B, S.C,
           R.W + S.W as weight
FROM      R, S
WHERE    R.B=S.B
ORDER BY weight ASC
LIMIT    1;
```

$O(n^2)$

n=1000: t_{Q1}= 0.88 sec

n=5000: t_{Q1}=18.6 sec

```
-------------------------
-- Query 2
-------------------------
SELECT R.A, X.B, S.C, X.W as weight
FROM R, S,
    (SELECT T1.B, W1, W2, W1+W2 W
    FROM
        (SELECT B, MIN(W) W1
        FROM R
        GROUP BY B) T1,
        (SELECT B, MIN(W) W2
        FROM S
        GROUP BY B) T2
    WHERE T1.B = T2.B
    ORDER BY W ASC
    LIMIT 1) X
WHERE X.B = R.B
AND X.W1 = R.W
AND X.B = S.B
AND X.W2 = S.W
LIMIT   1;
```

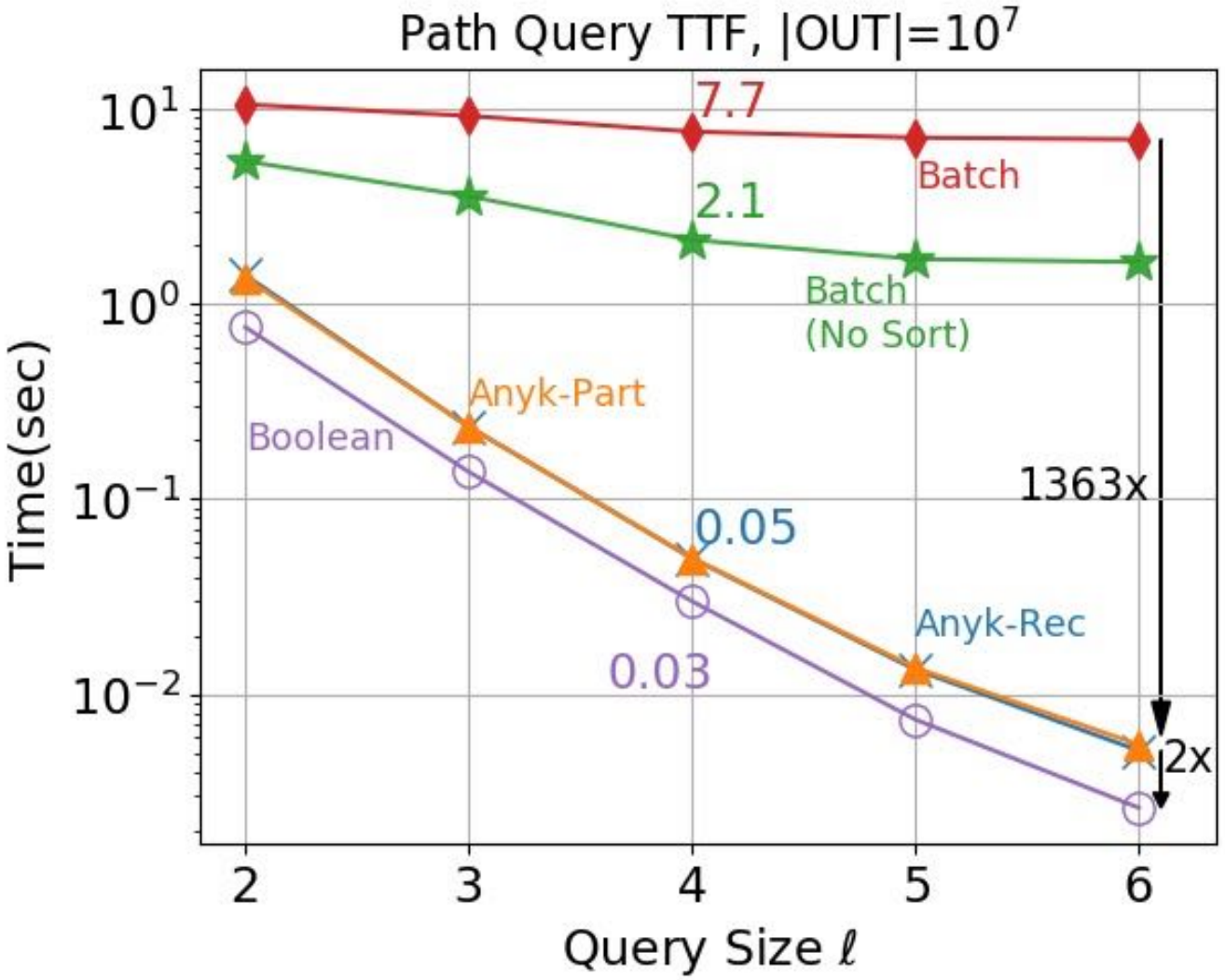
$O(n)$

t_{Q2}=2 msec

t_{Q2}=8 msec

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Any-*k*: Faster and more versatile than Top-*k*



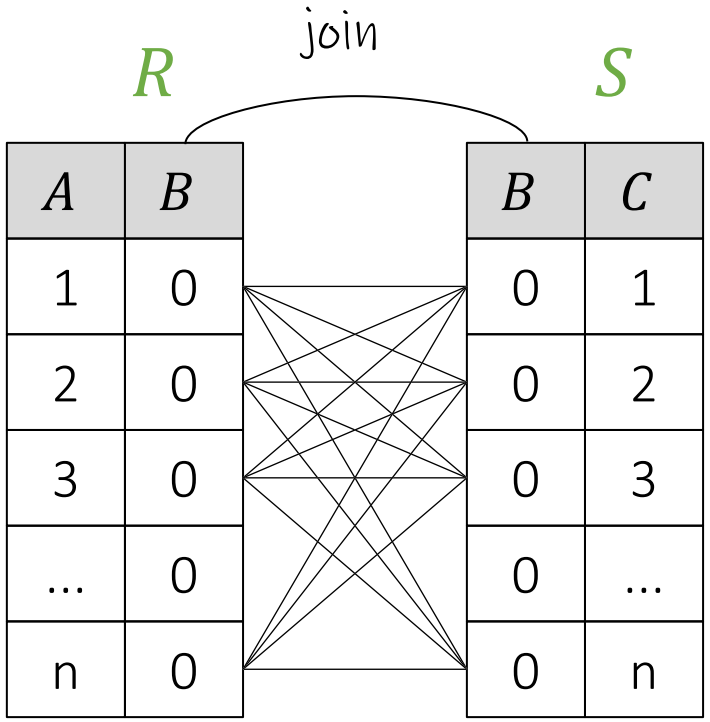
Path query with
constant size output
and increasing query size

Tziavalis+ [PVLDB'20], Tziavelis+ [SIGMOD'20 tutorial]

https://northeastern-datalab.github.io/anyk/
https://northeastern-datalab.github.io/topk-join-tutorial/
https://www.youtube.com/watch?v=KpUQayBuaQI&list=PL_72ERGKF6DR7kvGNwwjWlbpScKtGjt9R&index=2

Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB 2020. https://doi.org/10.14778/3397230.3397250
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Even Grouping Aggregates can be improved




```
-------------------------
-- Query 1
-------------------------
SELECT   count(*) CT
INTO record1
FROM      R, S
WHERE    R.B=S.B;
```

```
-------------------------
-- Query 2
-------------------------
SELECT SUM(C) as CT
INTO record2
FROM
    (SELECT T1.B, C1, C2, C1*C2 C
    FROM
        (SELECT B, COUNT(*) C1
        FROM R
        GROUP BY B) T1,
        (SELECT B, COUNT(*) C2
        FROM S
        GROUP BY B) T2
    WHERE T1.B = T2.B) X;
```

n=1000: t_{Q1}= 0.374 sec t_{Q2}=3 msec

n=5000: t_{Q1}= 10.021 sec t_{Q2}=5 msec

SQL example available at: https://github.com/northeastern-datalab/cs3200-activities/tree/master/sql

Outline: T1-U1: SQL

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k
  - [Recursion: moved to T1-U4: Datalog]