Updated 10/5/2022

Topic 1: SQL
L09: SQL advanced

Wolfgang Gatterbauer

CS3200 Database design (fa22)

https://northeastern-datalab.github.io/cs3200/fa22s3/

10/5/2022

Class warm-up

- Last class summary
- 4th HW: random groups again?
- Exam WED next week in class: namelist
- Comment on last in-class exercise

- SQL today: top-k, recursion

```sql
select count(distinct a1.id)
from actor a0, play p0, play p1, actor a1
where a0.id = p0.aid
and p0.mid = p1.mid
and p1.aid = a1.id
and a0.fname = 'Kevin'
and a0.lname = 'Bacon'
and a1.fname <> 'Kevin'
and a1.lname <> 'Bacon'
```

```sql
select count(distinct a1.id)
from actor a0, play p0, play p1, actor a1
where a0.id = p0.aid
and p0.mid = p1.mid
and p1.aid = a1.id
and a0.fname = 'Kevin'
and a0.lname = 'Bacon'
and a1.id not in (
    select id
    from actor
    where a1.fname = 'Kevin'
    and a1.lname = 'Bacon')
```

```sql
select count(distinct a1.id)
from actor a0, play p0, play p1, actor a1
where a0.id = p0.aid
and p0.mid = p1.mid
and p1.aid = a1.id
and a0.fname = 'Kevin'
and a0.lname = 'Bacon'
and a1.id <> a0.id
```

```
select count(distinct a1.id)
from actor a0, play p0, play p1, actor a1
where a0.id = p0.aid
and p0.mid = p1.mid
and p1.aid = a1.id
and a0.fname = 'Kevin'
and a0.lname = 'Bacon'
and a1.fname <> 'Kevin'
and a1.lname <> 'Bacon'
```

```
select count(distinct a1.id)
from actor a0, play p0, play p1, actor a1
where a0.id = p0.aid
and p0.mid = p1.mid
and p1.aid = a1.id
and a0.fname = 'Kevin'
and a0.lname = 'Bacon'
and (a1.fname <> 'Kevin'
or a1.lname <> 'Bacon')
```

```
select count(distinct a1.id)
from actor a0, play p0, play p1, actor a1
where a0.id = p0.aid
and p0.mid = p1.mid
and p1.aid = a1.id
and a0.fname = 'Kevin'
and a0.lname = 'Bacon'
and not (a1.fname = 'Kevin'
and a1.lname = 'Bacon')
```

Are Outer Joins redundant?

Another look at Outer Joins


```
SELECT  *
FROM    English FULL JOIN French
ON      English.eid = French.fid
```

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

?

FULL JOIN can be written as union of inner join with anti-joins

etext	eid	fid	ftext
One	1	1	Un
Two	2	NULL	NULL
Three	3	3	Trois
Four	4	4	Quatre
Five	5	5	Cinq
Six	6	6	Siz
NULL	NULL	7	Sept
NULL	NULL	8	Huit

Another look at Outer Joins


```
SELECT  *
FROM    English FULL JOIN French
ON      English.eid = French.fid
```

```
SELECT    etext,eid, fid, ftext
FROM      English Inner JOIN French
ON        English.eid = French.fid
UNION
SELECT    etext, eid, NULL, NULL
FROM      English
WHERE     NOT EXISTS(
          SELECT *
          FROM French
          WHERE eid=fid)
UNION
SELECT    NULL, NULL, fid, ftext
FROM      French
WHERE     NOT EXISTS(
          SELECT *
          FROM English
          WHERE eid=fid)
```

anti-join

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

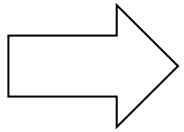
fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

etext	eid	fid	ftext
One	1	1	Un
Two	2	NULL	NULL
Three	3	3	Trois
Four	4	4	Quatre
Five	5	5	Cinq
Six	6	6	Siz
NULL	NULL	7	Sept
NULL	NULL	8	Huit

CASE, Coalesce

Join Illustration


```
select  *

        ?

from  English FULL JOIN French
on  English.eid = French.fid
```

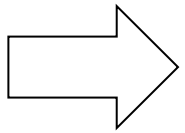


< 5

etext	eid	fid	ftext	new
One	1	1	Un	true
Two	2	NULL	NULL	true
Three	3	3	Trois	true
Four	4	4	Quatre	true
Five	5	5	Cinq	false
Six	6	6	Siz	false
NULL	NULL	7	Sept	NULL
NULL	NULL	8	Huit	NULL

Join Illustration


```
select  *,
  eid < 5 as new



from  English FULL JOIN French
on  English.eid = French.fid
```

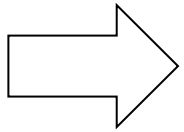


< 5

etext	eid	fid	ftext	new
One	1	1	Un	true
Two	2	NULL	NULL	true
Three	3	3	Trois	true
Four	4	4	Quatre	true
Five	5	5	Cinq	false
Six	6	6	Siz	false
NULL	NULL	7	Sept	NULL
NULL	NULL	8	Huit	NULL

Join Illustration


```
select  *,
  eid < 5 as new

        ?

from  English FULL JOIN French
on  English.eid = French.fid
```

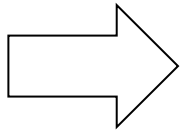


< 5

etext	eid	fid	ftext	new
One	1	1	Un	yes
Two	2	NULL	NULL	yes
Three	3	3	Trois	yes
Four	4	4	Quatre	yes
Five	5	5	Cinq	no
Six	6	6	Siz	no
NULL	NULL	7	Sept	NULL
NULL	NULL	8	Huit	NULL

Join Illustration


```
select  *,
  CASE
    WHEN eid < 5 THEN 'yes'
    WHEN eid >= 5 THEN 'no'
    ELSE null
  END as new
from  English FULL JOIN French
on  English.eid = French.fid
```



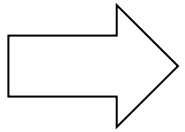
< 5

etext	eid	fid	ftext	new
One	1	1	Un	yes
Two	2	NULL	NULL	yes
Three	3	3	Trois	yes
Four	4	4	Quatre	yes
Five	5	5	Cinq	no
Six	6	6	Siz	no
NULL	NULL	7	Sept	NULL
NULL	NULL	8	Huit	NULL

Join Illustration


```
select  *,
  CASE
    WHEN eid < 5 THEN 'yes'
    WHEN eid >= 5 THEN 'no'
    ELSE null
  END as new
from  English FULL JOIN French
on  English.eid = French.fid
```

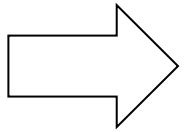
?



etext	eid	fid	ftext	new
One	1	1	Un	1
Two	2	NULL	NULL	2
Three	3	3	Trois	3
Four	4	4	Quatre	4
Five	5	5	Cinq	5
Six	6	6	Siz	6
NULL	NULL	7	Sept	7
NULL	NULL	8	Huit	8

Join Illustration

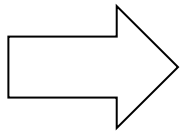

```
select  *,
  CASE
    WHEN eid is not null THEN eid
    ELSE fid

  END as new
from  English FULL JOIN French
on  English.eid = French.fid
```



etext	eid	fid	ftext	new
One	1	1	Un	1
Two	2	NULL	NULL	2
Three	3	3	Trois	3
Four	4	4	Quatre	4
Five	5	5	Cinq	5
Six	6	6	Siz	6
NULL	NULL	7	Sept	7
NULL	NULL	8	Huit	8

Join Illustration


```
select  *,
  COALESCE (eid, fid)



  as new
from  English FULL JOIN French
on  English.eid = French.fid
```



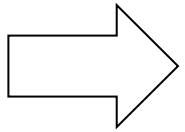
etext	eid	fid	ftext	new
One	1	1	Un	1
Two	2	NULL	NULL	2
Three	3	3	Trois	3
Four	4	4	Quatre	4
Five	5	5	Cinq	5
Six	6	6	Siz	6
NULL	NULL	7	Sept	7
NULL	NULL	8	Huit	8

Join Illustration



```
select  *,
  CASE
    WHEN eid is not null THEN eid
    ELSE fid
  END as new
from  English FULL JOIN French
on  English.eid = French.fid
```

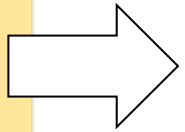
?



etext	eid	fid	ftext	new
One	1	1	Un	Both
Two	2	NULL	NULL	Left
Three	3	3	Trois	Both
Four	4	4	Quatre	Both
Five	5	5	Cinq	Both
Six	6	6	Siz	Both
NULL	NULL	7	Sept	Right
NULL	NULL	8	Huit	Right

Join Illustration


```
select  *,
  CASE
    WHEN eid is not null THEN
      CASE
        WHEN fid is not null THEN 'Both'
        ELSE 'Left'
      END
    ELSE 'Right'
  END as new
from  English FULL JOIN French
on  English.eid = French.fid
```

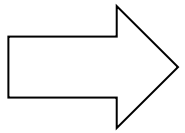


etext	eid	fid	ftext	new
One	1	1	Un	Both
Two	2	NULL	NULL	Left
Three	3	3	Trois	Both
Four	4	4	Quatre	Both
Five	5	5	Cinq	Both
Six	6	6	Siz	Both
NULL	NULL	7	Sept	Right
NULL	NULL	8	Huit	Right

Join Illustration

once a condition is met, CASE returns the result and stops reading

```
select  *,
  CASE
    WHEN eid is not null and
      fid is not null THEN 'Both'
    WHEN eid is not null THEN 'Left'
    ELSE 'Right'
  END as new


from  English FULL JOIN French
on  English.eid = French.fid
```



etext	eid	fid	ftext	new
One	1	1	Un	Both
Two	2	NULL	NULL	Left
Three	3	3	Trois	Both
Four	4	4	Quatre	Both
Five	5	5	Cinq	Both
Six	6	6	Siz	Both
NULL	NULL	7	Sept	Right
NULL	NULL	8	Huit	Right

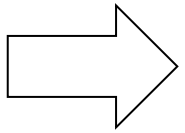
Witnesses with nulls

Witnesses with nulls

T'

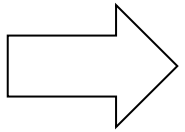
product	price
Apple	3
Apple	2
Banana	1
Banana	2
Banana	4

```
SELECT   product, price as mp
FROM     T
WHERE    price =
         (SELECT max(price)
         FROM     T)
```



?

```
SELECT   product, price as mp
FROM     T
WHERE    price >= ALL
         (SELECT price
         FROM     T)
```



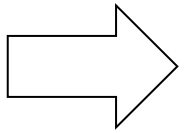
?

Witnesses with nulls

T'

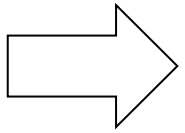
product	price
Apple	3
Apple	2
Banana	1
Banana	2
Banana	4

```
SELECT   product, price as mp
FROM     T
WHERE    price =
         (SELECT max(price)
         FROM     T)
```



product	mp
Banana	4

```
SELECT   product, price as mp
FROM     T
WHERE    price >= ALL
         (SELECT price
         FROM     T)
```



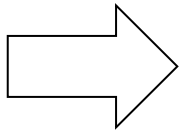
product	mp
Banana	4

Witnesses with nulls

T

product	price
Apple	null
Apple	2
Banana	1
Banana	2
Banana	4

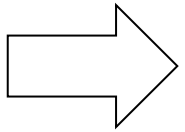
```
SELECT   product, price as mp
FROM     T
WHERE    price =
         (SELECT max(price)
         FROM     T)
```



product	mp
Banana	4

?

```
SELECT   product, price as mp
FROM     T
WHERE    price >= ALL
         (SELECT price
         FROM     T)
```



product	mp
Banana	4

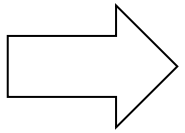
?

Witnesses with nulls

T

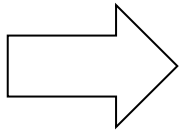
product	price
Apple	null
Apple	2
Banana	1
Banana	2
Banana	4

```
SELECT   product, price as mp
FROM     T
WHERE    price =
         (SELECT max(price)
         FROM     T)
```



product	mp
Banana	4

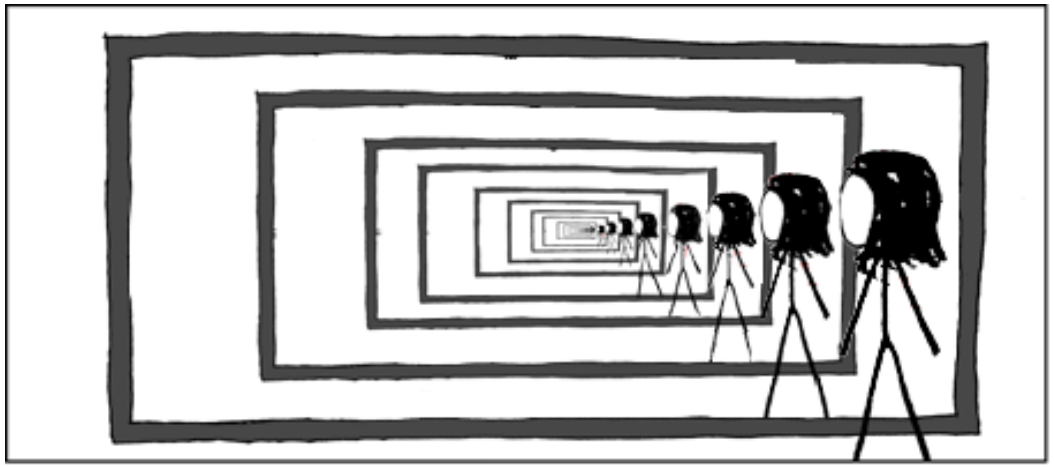
```
SELECT   product, price as mp
FROM     T
WHERE    price >= ALL
         (SELECT price
         FROM     T)
```

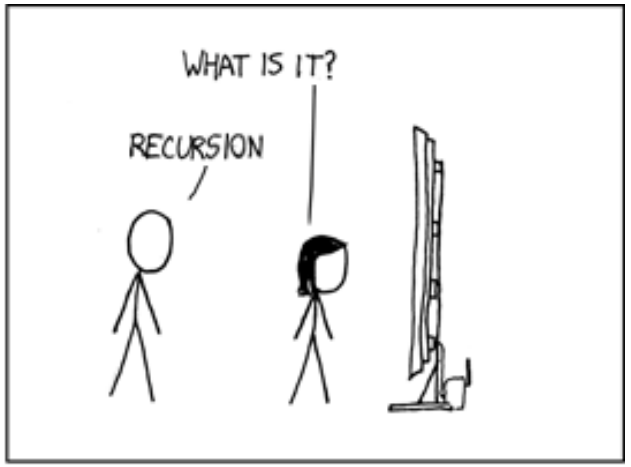


product	mp

Recursion with SQL

Recursion





Recursion occurs when a thing is defined in terms of itself (self-repetition).

Recursion and Iteration both repeatedly execute a set of instructions.

- Recursion (self-similarity) is when a statement in a function calls itself repeatedly.
- Iteration (repetition) is when a loop repeatedly executes until the controlling condition becomes false.

Figure Source: Fake XKCD: http://xkcdsw.com/1105

1. A simple recursive query

$X = X + 2$

non-recursive part

recursive part, contains reference to the query's output

```
WITH RECURSIVE T(n) as (
    values (1)
  UNION ALL
    select n+1
    from T
    where n<=3)
SELECT n FROM T
```

	n integer
1	1
2	2
3	3
4	4

Step	Working T.	Interm. T.
1	?	?
2		
3		
4		
5		

Recursive Query Evaluation

1. Evaluate the non-recursive term. For `UNION` (but not `UNION ALL`), discard duplicate rows. Include all remaining rows in the result of the recursive query, and also place them in a temporary *working table*.

2. So long as the working table is not empty, repeat these steps:

    a. Evaluate the recursive term, substituting the current contents of the working table for the recursive self-reference. For `UNION` (but not `UNION ALL`), discard duplicate rows and rows that duplicate any previous result row. Include all remaining rows in the result of the recursive query, and also place them in a temporary *intermediate table*.

    b. Replace the contents of the working table with the contents of the intermediate table, then empty the intermediate table.

Example slightly adapted from: https://www.postgresql.org/docs/current/queries-with.html#QUERIES-WITH-RECURSIVE

1. A simple recursive query

non-recursive part

recursive part, contains reference to the query's output

```
WITH RECURSIVE T(n) as (
    values (1)
  UNION ALL
    select n+1
    from T
    where n<=3)
SELECT n FROM T
```

	n integer
1	1
2	2
3	3
4	4

Step	Working T.	Interm. T.
1	1	
2	2	2
3	3	3
4	4	4
5		

Recursive Query Evaluation

1. Evaluate the non-recursive term. For `UNION` (but not `UNION ALL`), discard duplicate rows. Include all remaining rows in the result of the recursive query, and also place them in a temporary *working table*.

2. So long as the working table is not empty, repeat these steps:

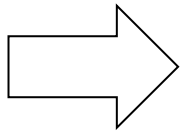
    a. Evaluate the recursive term, substituting the current contents of the working table for the recursive self-reference. For `UNION` (but not `UNION ALL`), discard duplicate rows and rows that duplicate any previous result row. Include all remaining rows in the result of the recursive query, and also place them in a temporary *intermediate table*.

    b. Replace the contents of the working table with the contents of the intermediate table, then empty the intermediate table.

Example slightly adapted from: https://www.postgresql.org/docs/current/queries-with.html#QUERIES-WITH-RECURSIVE

2. Fibonacci numbers: 0, 1, 1, 2, 3, 5, 8, 13, …

```
WITH RECURSIVE Fib as (
      select 0 as n,
             0 as "fib_n",
             1 as "fib_n+1"
   UNION ALL
      select n+1,
             "fib_n+1",
             "fib_n" + "fib_n+1"
      from Fib)
SELECT * FROM Fib
LIMIT 10;
```



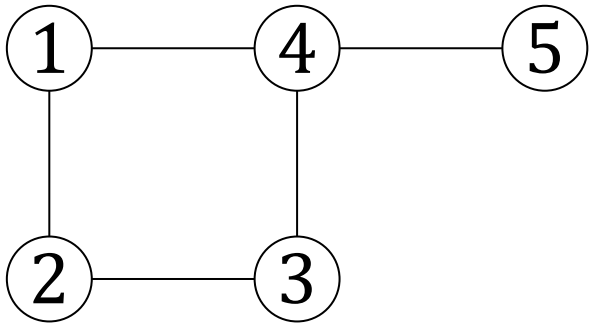
Fib

	n integer	fib_n integer	fib_{n+1} integer
1	0	0	1
2	1	1	1
3	2	1	2
4	3	2	3
5	4	3	5
6	5	5	8
7	6	8	13
8	7	13	21
9	8	21	34
10	9	34	55

Example slightly adapted from: https://www.cybertec-postgresql.com/en/recursive-queries-postgresql/
Wolfgang Gatterbauer. Database design: https://northeastern-datalab.github.io/cs3200/

Querying graphs

E for undirected edge, E(S,T)
S for source, T for target



"Find nodes that have at least two distinct neighbors"

E

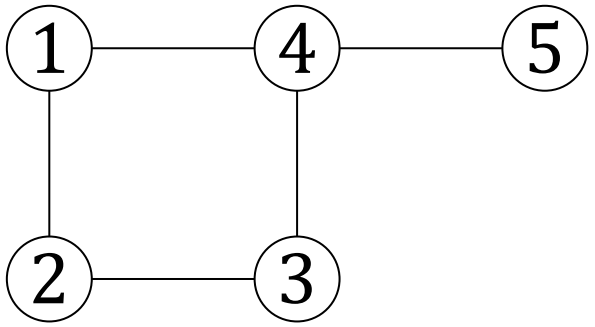
S	T
1	2
1	4
2	1
2	3
3	2
3	4
4	1
4	3
4	5
5	4

?

Querying graphs

E for undirected edge, E(S,T)
S for source, T for target

501



"Find nodes that have at least two distinct neighbors"

$\{x \mid \exists y, \exists z\ [E(x, y) \wedge E(x, z) \wedge y \neq z]\}$

$\{q(s) \mid \exists E1 \in E, \exists E1 \in E\ [Q.s = E1.s \wedge E1.s = E2.s \wedge E1.t \neq E2.t]\}$

E

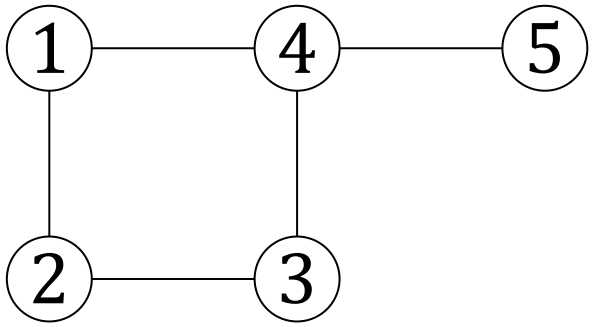
S	T
1	2
1	4
2	1
2	3
3	2
3	4
4	1
4	3
4	5
5	4

?

Querying graphs

E for undirected edge, E(S,T)
S for source, T for target

501



E

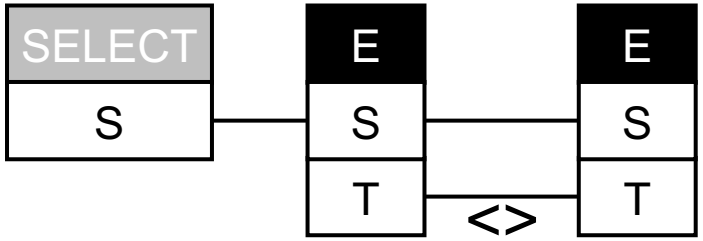
S	T
1	2
1	4
2	1
2	3
3	2
3	4
4	1
4	3
4	5
5	4

"Find nodes that have at least two distinct neighbors"

$\{x \mid \exists y, \exists z\ [E(x, y) \wedge E(x, z) \wedge y \neq z]\}$

$\{q(s) \mid \exists E1 \in E, \exists E1 \in E\ [Q.s = E1.s \wedge E1.s = E2.s \wedge E1.t \neq E2.t]\}$

```
select   distinct  E1.S
from     E E1, E E2
where    E1.S = E2.S
and      E1.T != E2.T
```

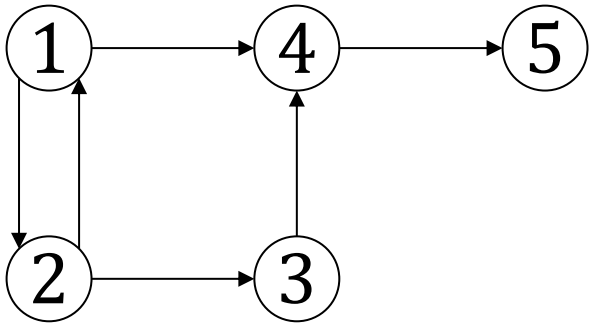



```
select   S
from     E
group by S
having   COUNT(T)>=2
```

3. Recursion on graphs

A for directed edges ("arcs") A(S,T)

501



"Find all paths (transitive closure)"

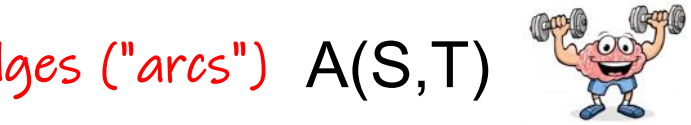
A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

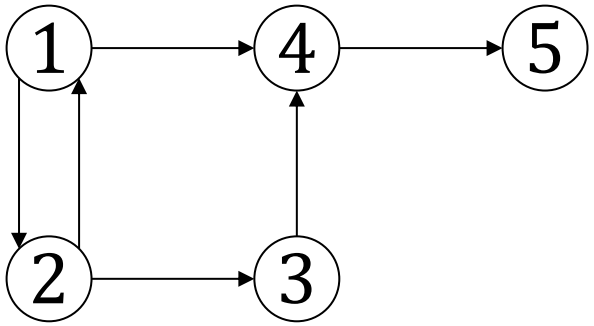
?

3. Recursion on graphs

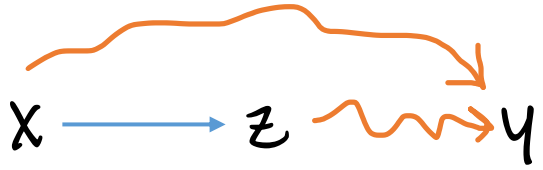
A for directed edges ("arcs") A(S,T)



501



"Find all paths (transitive closure)"



A

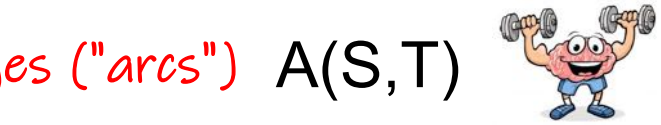
S	T
1	2
1	4
2	1
2	3
3	4
4	5

1. Create a path for every arc

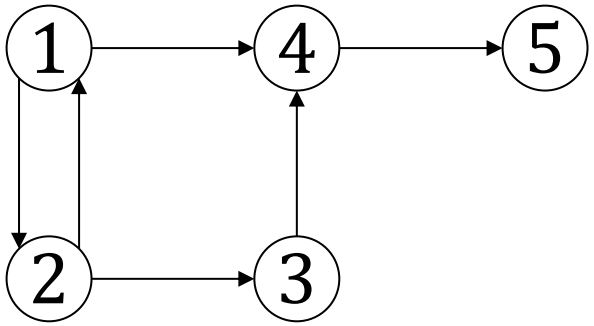
2. An arc + a path can make another path

3. Recursion on graphs

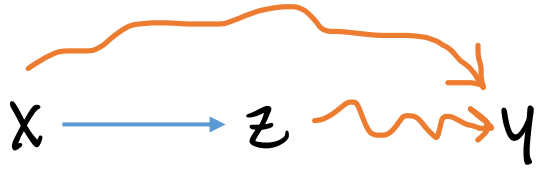
A for directed edges ("arcs") A(S,T)



501



"Find all paths (transitive closure)"



A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

For all nodes x and y:
If there is an arc from x to y,
then there is a path from x to y.

```
P(x,y) :- A(x,y).
P(x,y) :- A(x,z), P(z,y).
```

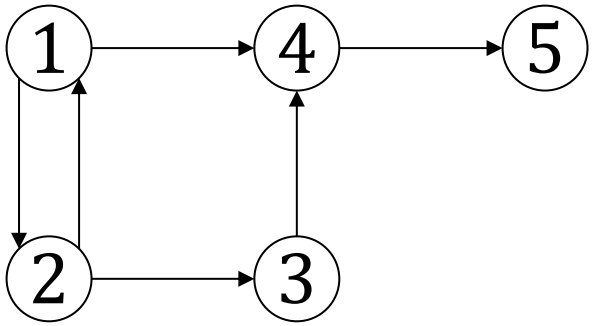
For all nodes x, z, and y:
If there is an arc from x to z, and there is a path from z to y
then there is a path from x to y.

3. Recursion on graphs

```
P(x,y) :- A(x,y).
P(x,y) :- A(x,z), P(z,y).
```

A(S,T)

501



1st iteration

A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

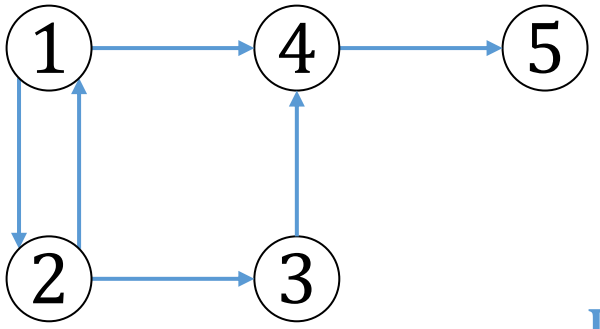
?

3. Recursion on graphs

```
P(x,y) :- A(x,y).
P(x,y) :- A(x,z), P(z,y).
```

A(S,T)

501



A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

1st iteration

2nd iteration

P

1	2
2	1
2	3
1	4
3	4
4	5

P=A from 1st rule

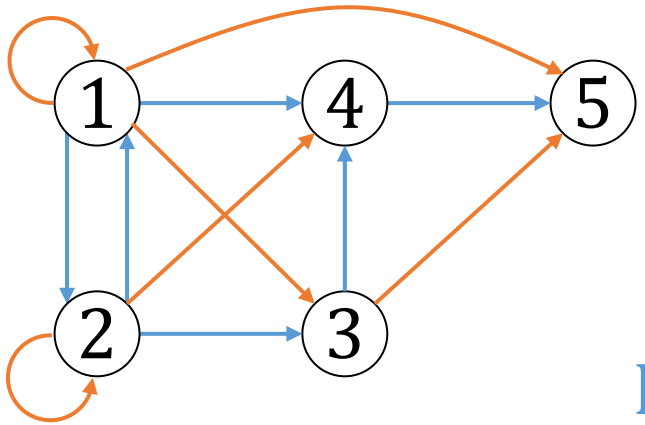
?

3. Recursion on graphs

A(S,T)

501

```
P(x,y) :- A(x,y).
P(x,y) :- A(x,z), P(z,y).
```



A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

1st iteration

P

1	2
2	1
2	3
1	4
3	4
4	5

P=A from 1st rule

2nd iteration

P'

1	2
2	1
2	3
1	4
3	4
4	5

1st rule

P

1	1
2	2
1	3
2	4
1	5
3	5

2nd rule

3rd iteration

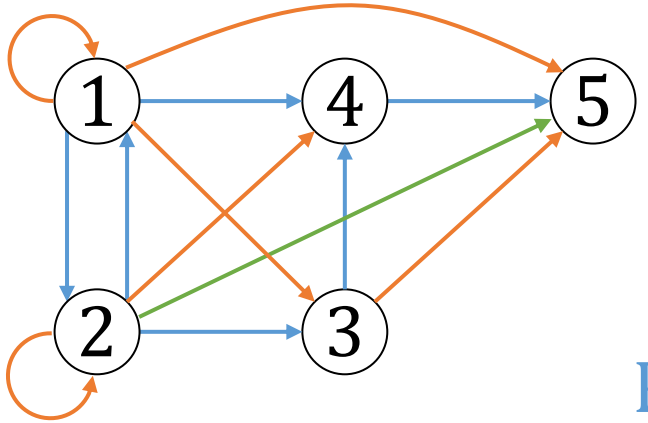
?

3. Recursion on graphs

A(S,T)

501

```
P(x,y) :- A(x,y).
P(x,y) :- A(x,z), P(z,y).
```



A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

1st iteration

P

1	2
2	1
2	3
1	4
3	4
4	5

P=A from 1st rule

2nd iteration

P'

1	2
2	1
2	3
1	4
3	4
4	5

1st rule

P

1	1
2	2
1	3
2	4
1	5
3	5

2nd rule

3rd iteration

P'

1	1
2	2
1	3
2	4
1	5
3	5

2nd rule

P

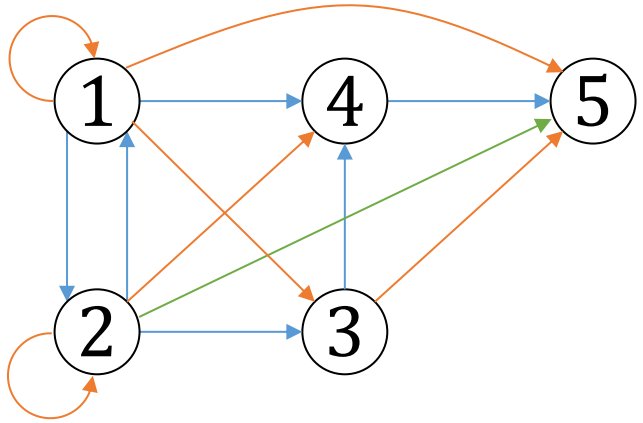
2	5

3. Recursion on graphs

A(S,T)

501

```
P(x,y) :- A(x,y).
P(x,y) :- A(x,z), P(z,y).
```



A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

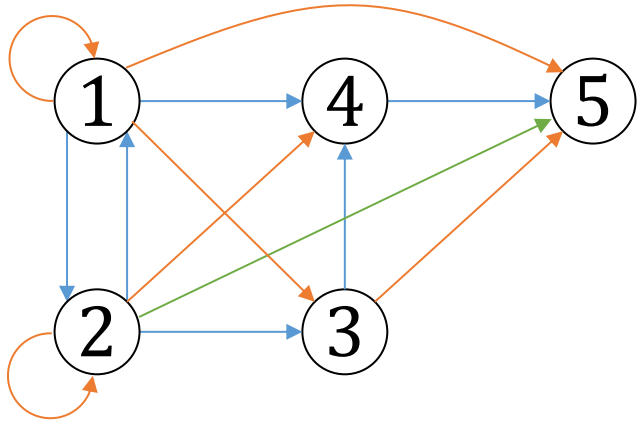
In SQL ?

3. Recursion on graphs

A(S,T)

501

```
P(x,y) :- A(x,y).
P(x,y) :- A(x,z), P(z,y).
```



A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

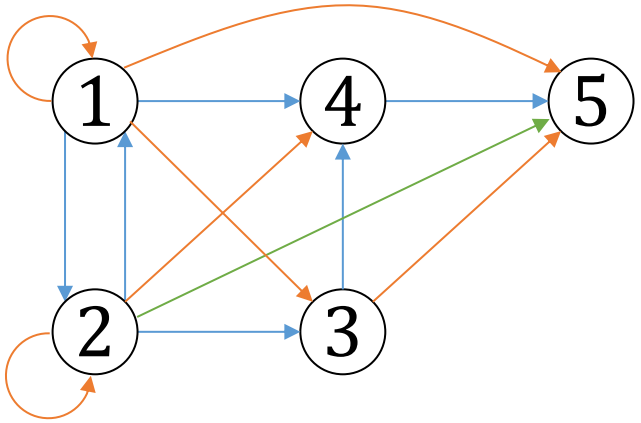
```
WITH RECURSIVE P AS (
     ?
   UNION
     ?

SELECT *
FROM P
```

3. Recursion on graphs

A(S,T)

501

```
P(x,y) :- A(x,y).
P(x,y) :- A(x,z), P(z,y).
```



A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

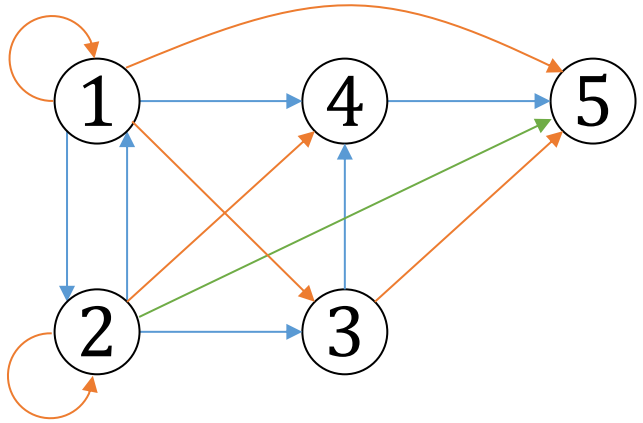
```
WITH RECURSIVE P AS (
    SELECT S, T
    FROM A
  UNION
    SELECT A.S, P.T
    FROM A, P
    WHERE A.T = P.S)
SELECT *
FROM P
```

3. Recursion on graphs

A(S,T)

501

```
P(x,y) :- A(x,y).
P(x,y) :- A(x,z), P(z,y).
```



A

S	T
1	2
1	4
2	1
2	3
3	4
4	5

Strictly speaking, this process is iteration, not recursion:

```
WITH RECURSIVE P AS (
    SELECT S, T
    FROM A
  UNION
    SELECT A.S, P.T
    FROM A, P
    WHERE A.T = P.S)
SELECT *
FROM P
```

Recursion and Iteration both repeatedly execute a set of instructions.

- Recursion (self-similarity) is when a statement in a function calls itself repeatedly.
- Iteration (reptition) is when a loop repeatedly executes until the controlling condition becomes false.

Small Detail in a video for a widely used textbook Ch06

(Hoffer, Ramesh, Topi: Modern database management, 10th ed, 2010)
Chapter 6: Introduction to SQL

Are these two queries equivalent? Hoffer video for Ch6 at around 20:20

Query

```
SELECT MaterialName, Material, Width
FROM RawMaterial_t
WHERE Material NOT IN ('Cherry', 'Oak')
  AND Width > 10;
```

Query

```
SELECT MaterialName, Material, Width
FROM RawMaterial_t
WHERE (Material != 'Cherry' OR Material != 'Oak')
  AND Width >10;
```

Source: http://mediaplayer.pearsoncmg.com/_ph_bp2_cc_set.title.Chapter_6%26%2358%3B_Writing_SQL_Queries%26%2358%3B_Part_1_%26%23150%3B_Single_Table_Queries_/bp_hoffer_mdm_10/hoffer10e_ch6_video.m4v

Venn Diagram

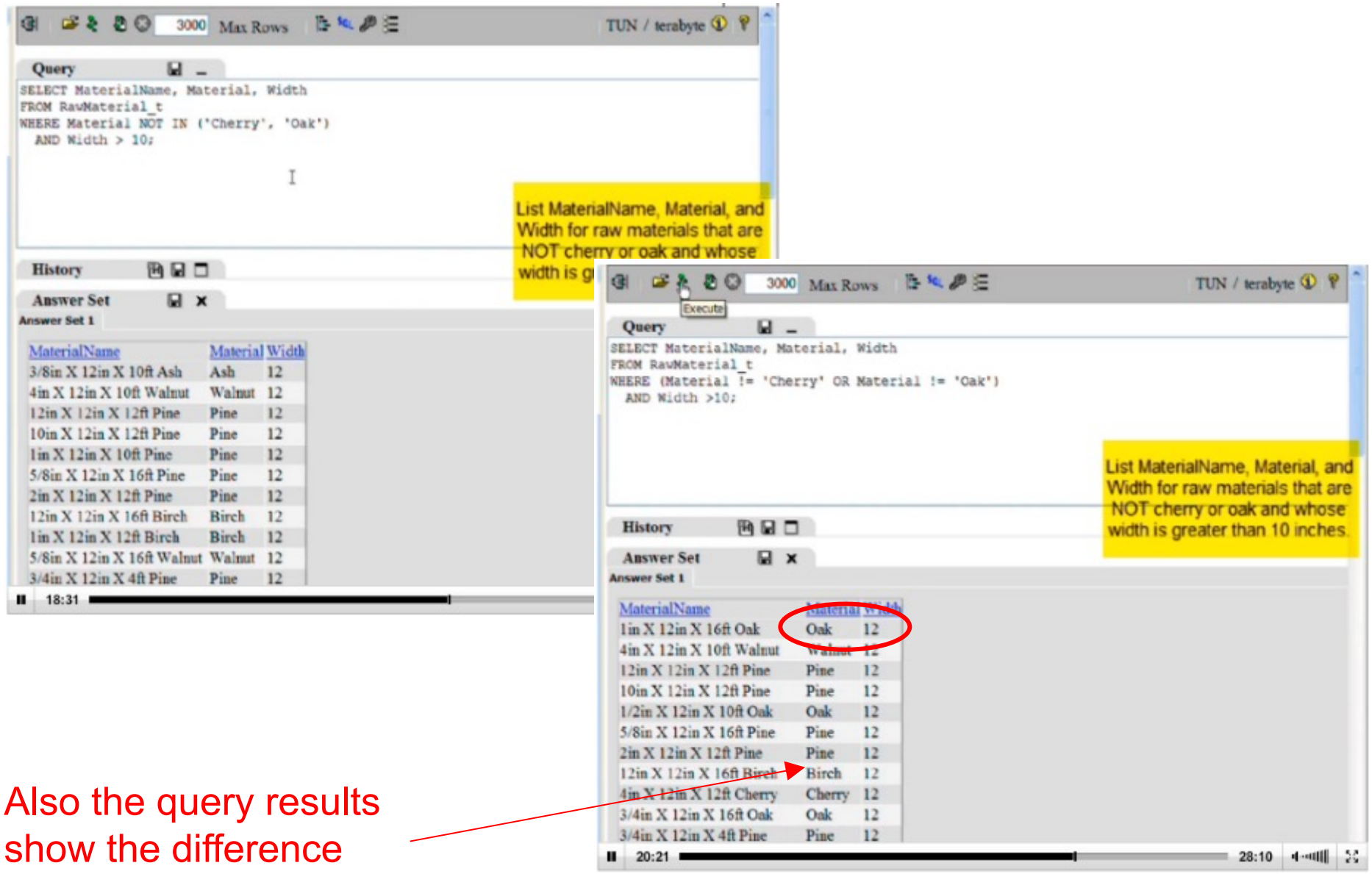
Query

```
SELECT MaterialName, Material, Width
FROM RawMaterial_t
WHERE Material NOT IN ('Cherry', 'Oak')
  AND Width > 10;
```

Query

```
SELECT MaterialName, Material, Width
FROM RawMaterial_t
WHERE (Material != 'Cherry' OR Material != 'Oak')
  AND Width >10;
```

Are these two queries equivalent?



Also the query results show the difference

Source: http://mediaplayer.pearsoncmg.com/_ph_bp2_cc_set.title.Chapter_6%26%2358%3B_Writing_SQL_Queries%26%2358%3B_Part_1_%26%23150%3B_Single_Table_Queries_/bp_hoffer_mdm_10/hoffer10e_ch6_video.m4v

Top-k, LIMIT,
Fetch first with ties

Witnesses revisited

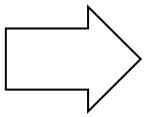
Product (pname, price, cid)

Product

PName	Price	cid
Gizmo	15	1
SuperGizmo	20	1
iTouch1	300	2
iTouch2	300	2

Q: Find the most expensive product + its price

```
SELECT  pname, price
FROM    Product
WHERE   price =
        (SELECT  max(price)
        FROM     Product)
```



PName	Price
iTouch1	300
iTouch2	300

Witnesses revisited

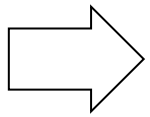
Product (pname, price, cid)

Product

PName	Price	cid
Gizmo	15	1
SuperGizmo	20	1
iTouch1	300	2
iTouch2	300	2

Q: Find the most expensive product + its price

```
SELECT  pname, price
FROM    Product
ORDER BY price DESC
LIMIT 1
```



PName	Price
iTouch1	300
iTouch2	300

?

not part of QL standard!

Witnesses revisited

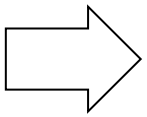
Product (pname, price, cid)

Product

PName	Price	cid
Gizmo	15	1
SuperGizmo	20	1
iTouch1	300	2
iTouch2	300	2

Q: Find the most expensive product + its price

```
SELECT  pname, price
FROM    Product
ORDER BY price DESC
LIMIT 1
```



PName	Price
iTouch1	300

not part of SQL standard!

Witnesses revisited

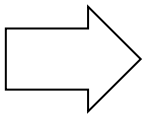
Product (pname, price, cid)

Product

PName	Price	cid
Gizmo	15	1
SuperGizmo	20	1
iTouch1	300	2
iTouch2	300	2

Q: Find the most expensive product + its price

```
SELECT  pname, price
FROM    Product
ORDER BY price DESC
FETCH FIRST 1 ROWS ONLY
```



PName	Price
iTouch1	300

part of SQL standard!

Witnesses revisited

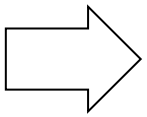
Product (pname, price, cid)

Product

PName	Price	cid
Gizmo	15	1
SuperGizmo	20	1
iTouch1	300	2
iTouch2	300	2

Q: Find the most expensive product + its price

```
SELECT  pname, price
FROM    Product
ORDER BY price DESC
FETCH FIRST 1 ROWS WITH TIES
```



PName	Price
iTouch1	300
iTouch2	300

part of SQL standard!

Actual commit message for this feature

projects / postgresql.git / commitdiff

summary | shortlog | log | commit | commitdiff | tree
raw | patch | inline | side by side (parent: 26a944c)

Support FETCH FIRST WITH TIES

```
author    Alvaro Herrera <alvherre@alvh.no-ip.org>
          Tue, 7 Apr 2020 20:22:13 +0000 (16:22 -0400)
committer Alvaro Herrera <alvherre@alvh.no-ip.org>
          Tue, 7 Apr 2020 20:22:13 +0000 (16:22 -0400)
```

```
WITH TIES is an option to the FETCH FIRST N ROWS clause (the SQL
standard's spelling of LIMIT), where you additionally get rows that
compare equal to the last of those N rows by the columns in the
mandatory ORDER BY clause.

There was a proposal by Andrew Gierth to implement this functionality in
a more powerful way that would yield more features, but the other patch
had not been finished at this time, so we decided to use this one for
now in the spirit of incremental development.
```

Commit message for PostgreSQL 13: https://git.postgresql.org/gitweb/?p=postgresql.git;a=commitdiff;h=357889eb17bb9c9336c4f324ceb1651da616fe57

Most expensive products for each company

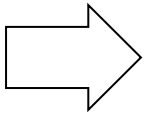
Product

PName	Price	cid
Gizmo	15	1
SuperGizmo	20	1
iTouch1	300	2
iTouch2	300	2

Company

cid	cname	city
1	GizmoWorks	Oslo
2	Apple	MountainView

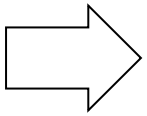
Find for each company id, the maximum price amongst its products [cid, cname, mp]



cid	cname	mp
1	GizmoWorks	20
2	Apple	300

cid	mp
1	20
2	300

Find for each company id, the product(s) with maximum price amongst its products (Recall that "group by cid" can just give us one single tuple per cid)



cid	cname	mp	pname
1	GizmoWorks	20	SuperGizmo
2	Apple	300	iTouch1
2	Apple	300	iTouch2

Most expensive products for each company

Product (pname, price, cid)
Company (cid, cname, city)

Q: For each company, find the most expensive products + their price

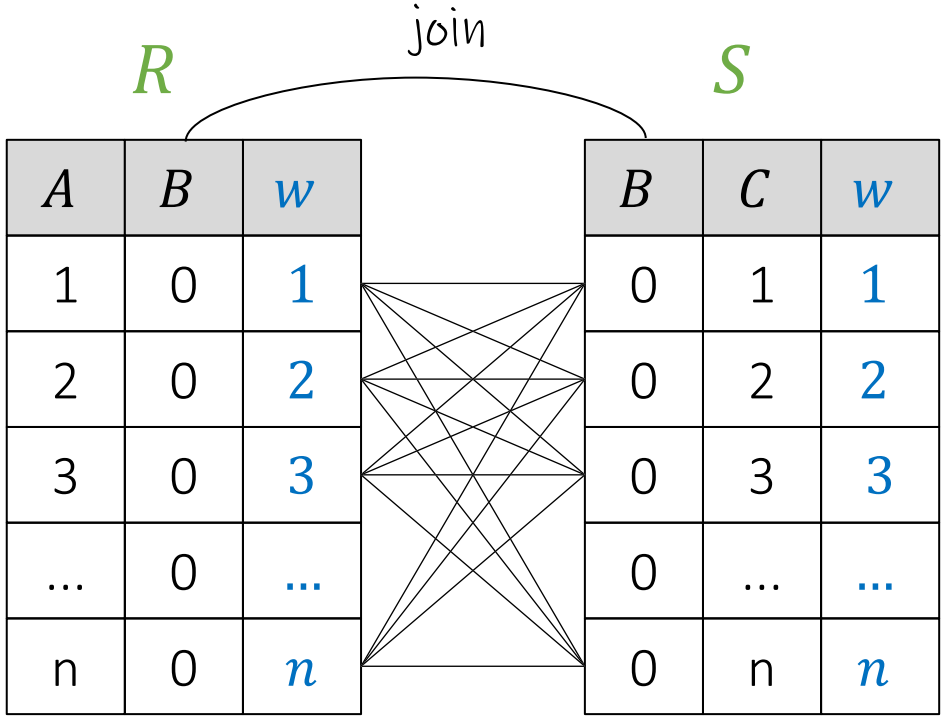
```
WITH  X as
      (SELECT   cid, max(price) as MP
      FROM      Product
      GROUP     BY cid)
SELECT cname, pname, X.MP
FROM   Company C, Product P, X
WHERE  C.cid = P.cid
   and C.cid = X.cid
   and P.price = X.MP
```

Plan with WITH:

- 1. Create a table that lists the max price for each company id
- 2. Join this table with the remaining tables

Top k and the current limits of LIMIT

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

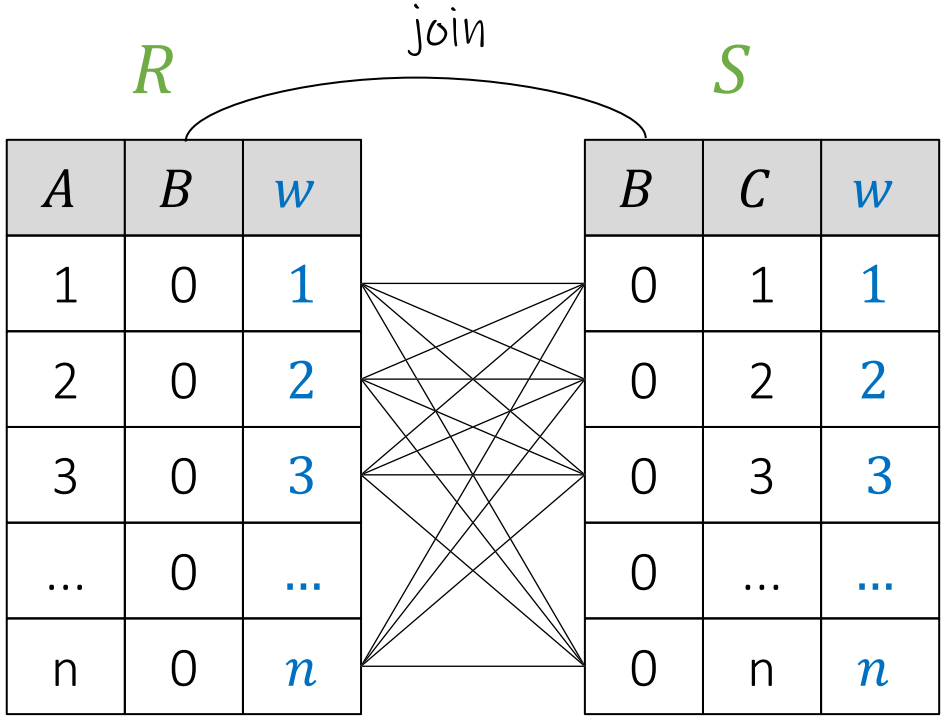
B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

```
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
```

What will this query return?

?

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
```

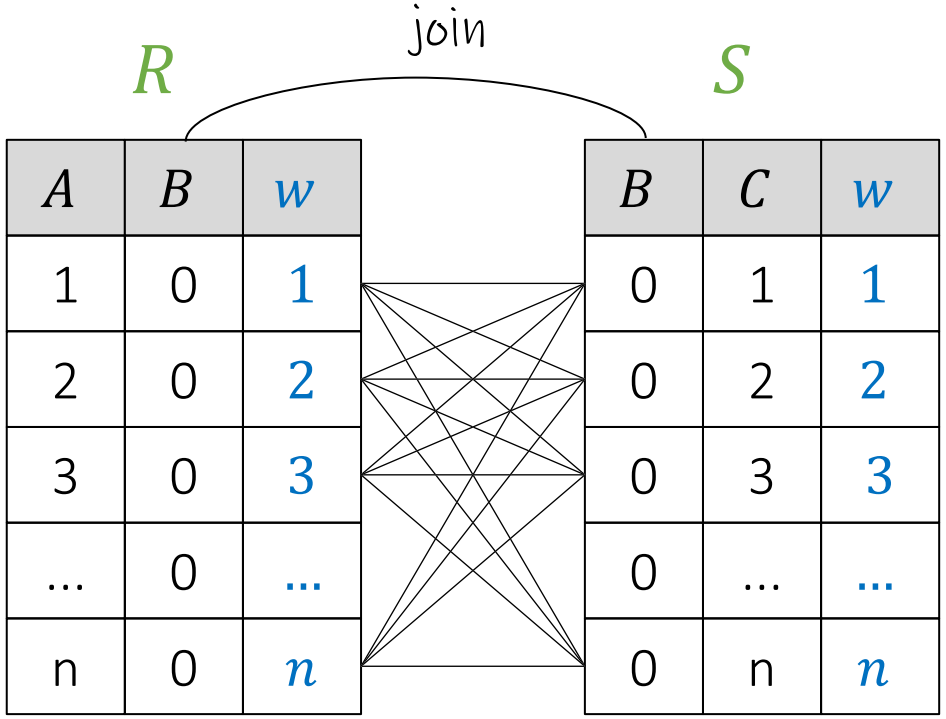
Result

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?

?

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

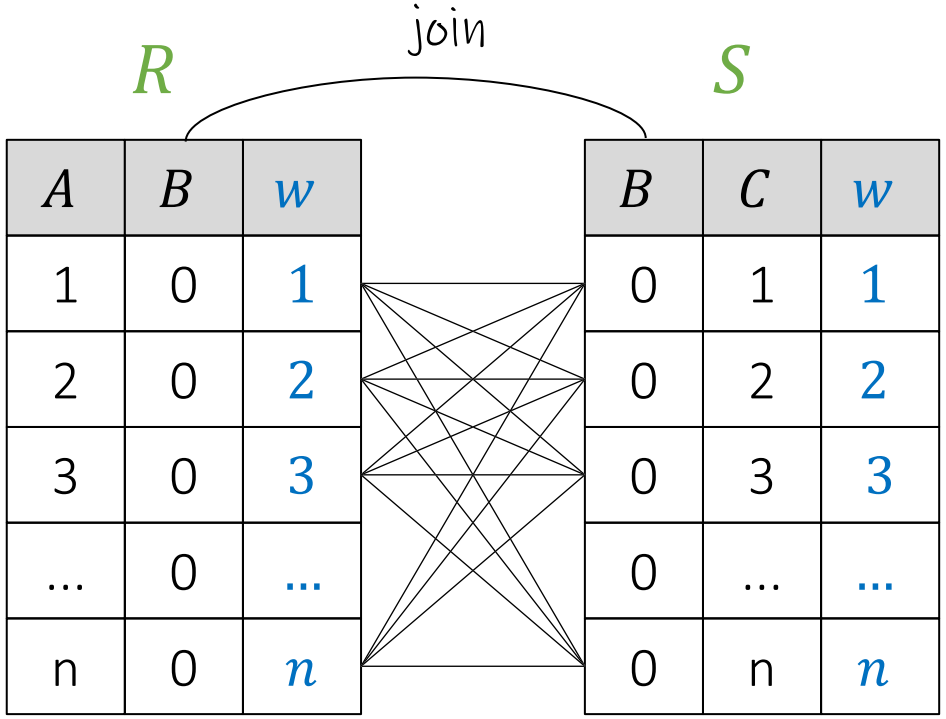
n^2 total results. But we are only interested in the top-1.

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

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

?

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
-------------------------
```

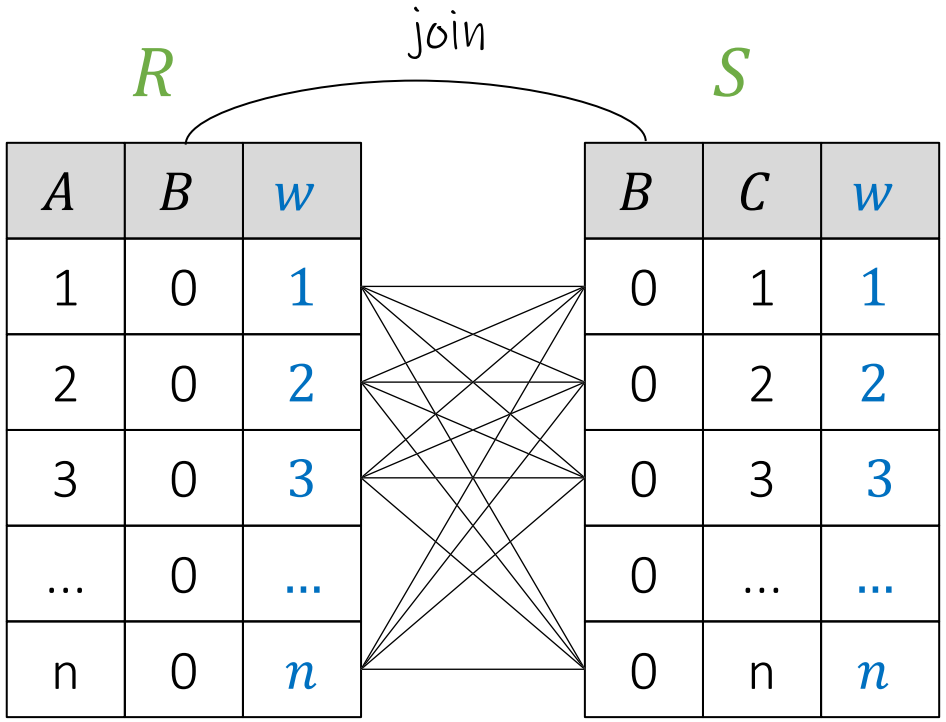
?

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



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;
```

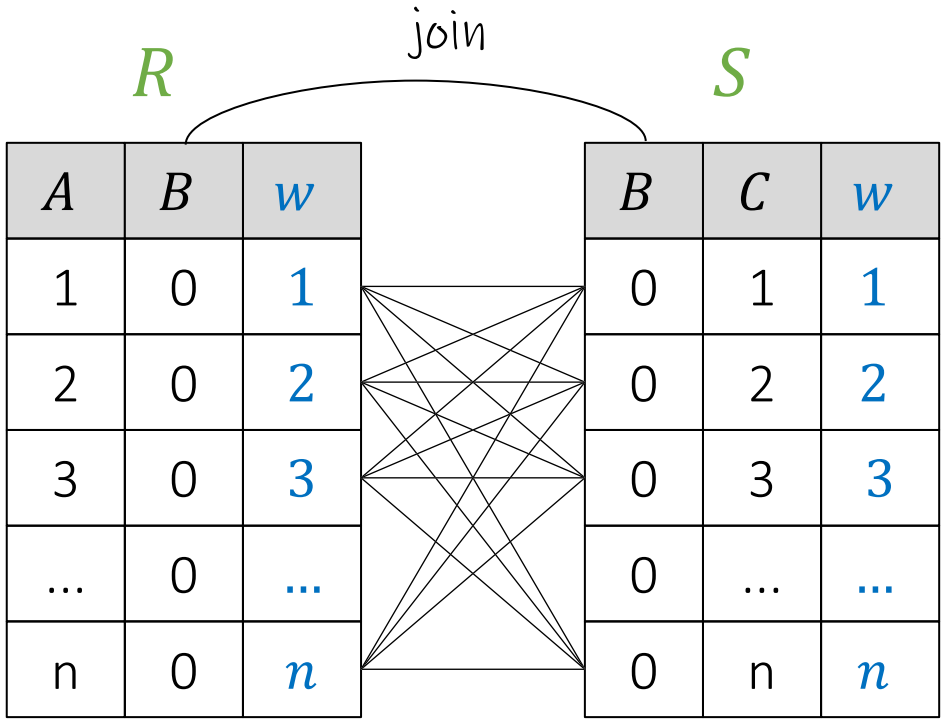
```
-------------------------
-- Query 2
-------------------------
SELECT R.A, X.B, S.C, X.W as weight
FROM R, S,
    (SELECT T1.B, W1, W2, W1+W2 W
    FROM
        (SELECT B, MIN(W) W1
        FROM R
        GROUP BY B) T1,
        (SELECT B, MIN(W) W2
        FROM S
        GROUP BY B) T2
    WHERE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

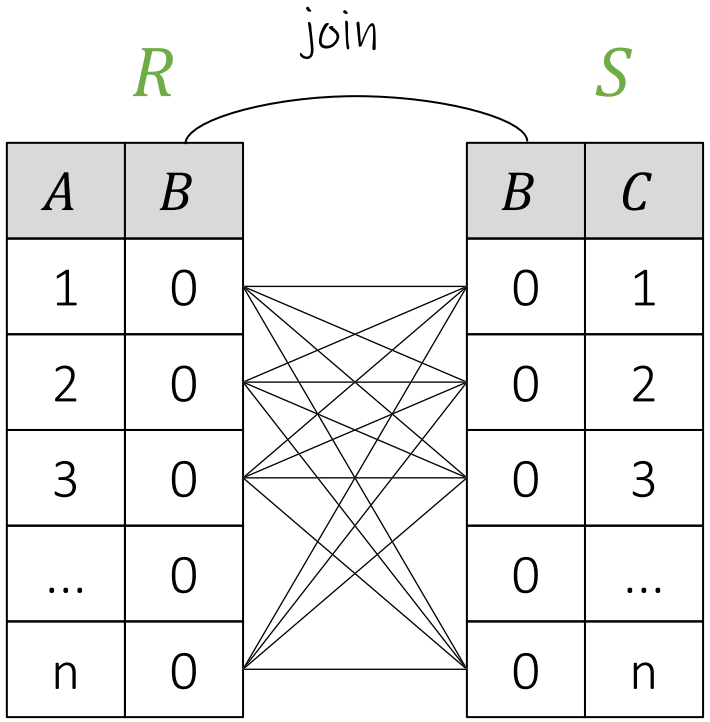
```
--------------------------
-- Query 2
--------------------------
SELECT R.A, X.B, S.C, X.W as weight
FROM R, S,
    (SELECT T1.B, W1, W2, W1+W2 W
    FROM
        (SELECT B, MIN(W) W1
        FROM R
        GROUP BY B) T1,
        (SELECT B, MIN(W) W2
        FROM S
        GROUP BY B) T2
    WHERE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

Even Grouping Aggregates can be improved




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

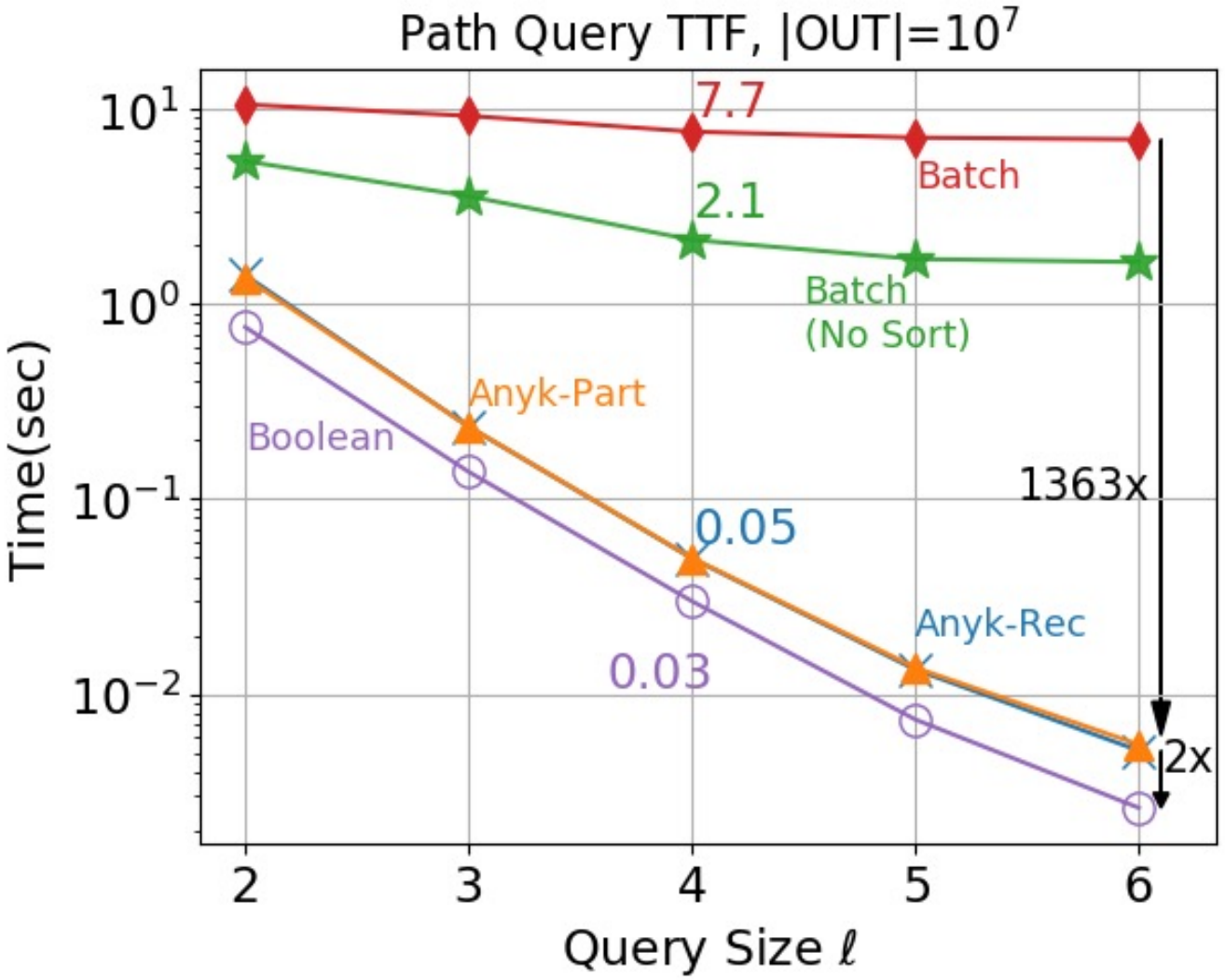
```
--------------------------
-- Query 2
--------------------------
SELECT SUM(C) as C
INTO record2
FROM
     (SELECT T1.B, C1, C2, C1*C2 C
     FROM
          (SELECT B, COUNT(*) C1
          FROM R
          GROUP BY B) T1,
          (SELECT B, COUNT(*) C2
          FROM S
          GROUP BY B) T2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

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



Path query with
constant size output
and increasing query size

Tziavalis+ [PVLDB'20], [SIGMOD'20 tutorial], [PVLDB'21], [ICDE'22 tutorial]

https://northeastern-datalab.github.io/anyk/
https://northeastern-datalab.github.io/topk-join-tutorial/
https://northeastern-datalab.github.io/responsive-dbms-tutorial/

Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB 2020. http://www.vldb.org/pvldb/vol13/p1582-tziavelis.pdf
Wolfgang Gatterbauer. Database design: https://northeastern-datalab.github.io/cs3200/

More Practice

1. What does this query return?

Actor

id	name	gender
1	Alice	f
2	Bob	m

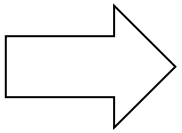
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  movie.name
FROM    movie, play, actor
WHERE   play.aid = actor.id
AND     movie.id = play.mid
AND     actor.id NOT IN
        (SELECT  a2.id
        FROM     actor a2
        WHERE    a2.gender = 'm')
```



?

1. What does this query return?

Actor

id	name	gender
1	Alice	f
2	Bob	m

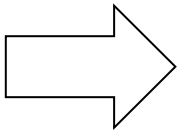
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  movie.name
FROM    movie, play, actor
WHERE   play.aid = actor.id
AND     movie.id = play.mid
AND     actor.id NOT IN
        (SELECT  a2.id
        FROM     actor a2
        WHERE    a2.gender = 'm')
```



name
Kill Bill

1. The Full join (aka "Universal Relation")

Select *

id	name	gender	aid	mid	role	id	name
1	Alice	f	1	1	role 1	1	Kill Bill
2	Bob	m	2	1	role 2	1	Kill Bill
2	Bob	m	2	1	role 3	1	Kill Bill

```
SELECT *
FROM   movie, play, actor
WHERE  play.aid = actor.id
AND    movie.id = play.mid
```

```
SELECT a2.id
FROM   actor a2
WHERE  a2.gender = 'm'
```

id
2

More information about the "universal relation": https://en.wikipedia.org/wiki/Universal_relation_assumption
Wolfgang Gatterbauer. Database design: https://northeastern-datalab.github.io/cs3200/

2. How to get the number of actors for each movie?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  m.name
FROM
WHERE
GROUP BY
```

?

2. How to get the number of actors for each movie?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

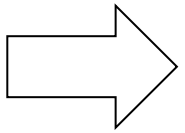
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  m.name, count(p.aid)
FROM    movie m, play p
WHERE   m.id = p.mid
GROUP BY m.name
```



name	(no name)
Kill Bill	4

?

2. How to get the number of actors for each movie?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

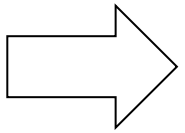
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  m.name, count(p.aid)
FROM    movie m, play p
WHERE   m.id = p.mid
GROUP BY m.id
```



?

2. How to get the number of actors for each movie?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

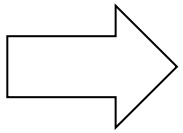
aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

When `GROUP BY` is present, or any aggregate functions are present, it is not valid for the `SELECT` list expressions to refer to ungrouped columns except within aggregate functions or when the ungrouped column is functionally dependent on the grouped columns, since there would otherwise be more than one possible value to return for an ungrouped column. A functional dependency exists if the grouped columns (or a subset thereof) are the primary key of the table containing the ungrouped column.

```
SELECT  m.name, count(p.aid)
FROM    movie m, play p
WHERE   m.id = p.mid
GROUP BY m.id
```



name	(no name)
Kill Bill	3
Kill Bill	1

?

Notice that this query may give erroror on some databases or older variants.

PostgreSQL can interpret the primary key (PK) m.id ⟶ m.name since version 9.1

Screenshot from: https://www.postgresql.org/docs/current/sql-select.html

2. How to get the number of actors for each movie?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

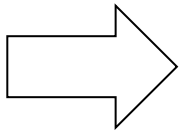
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  m.name, count(p.aid)
FROM    movie m, play p
WHERE   m.id = p.mid
GROUP BY m.id, m.name
```



name	(no name)
Kill Bill	3
Kill Bill	1

?

Screenshot from: https://www.postgresql.org/docs/current/sql-select.html
Wolfgang Gatterbauer. Database design: https://northeastern-datalab.github.io/cs3200/

2. How to get the number of actors for each movie?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

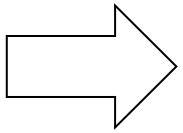
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  m.name, count(distinct p.aid)
FROM    movie m, play p
WHERE   m.id = p.mid
GROUP BY m.id, m.name
```



name	(no name)
Kill Bill	2
Kill Bill	1

3. How to get the number of actors for '%Bill%'?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  m.name, count(distinct p.aid)
FROM    movie m, play p
WHERE   m.id = p.mid

GROUP BY m.id, m.name
```

?

3. How to get the number of actors for '%Bill%'?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  m.name, count(distinct p.aid)
FROM    movie m, play p
WHERE   m.id = p.mid
AND     m.name like '%Bill%'
GROUP BY m.id, m.name
```

4. How to get the number of roles for each actor?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

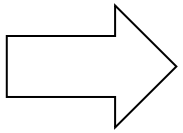
aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT
FROM
WHERE
GROUP BY
```

?



name	(no name)
Alice	1
Bob	2
Charly	1

4. How to get the number of roles for each actor?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

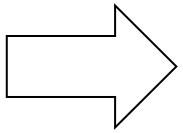
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  a.name, count(*)
FROM    actor a, play p
WHERE   a.id = p.aid
GROUP BY a.id, a.name
```



name	(no name)
Alice	1
Bob	2
Charly	1

4. How to get the number of roles for each actor?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  a.name, count(distinct p.aid)
FROM    actor a, play p
WHERE   a.id = p.aid
GROUP BY a.id, a.name
```

⇨ ?

4. How to get the number of roles for each actor?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  a.name, count(distinct p.aid)
FROM    actor a, play p
WHERE   a.id = p.aid
GROUP BY a.id, a.name
```

name	(no name)
Alice	1
Bob	1
Charly	1

Will always show 1 (since there is only one distinct aid per group grouped by aid *by def.*)

5. How to get the number of movies for each actor?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

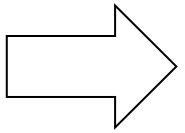
aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT
FROM
WHERE
GROUP BY
```

?



name	(no name)
Alice	1
Bob	1
Charly	1

5. How to get the number of movies for each actor?

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

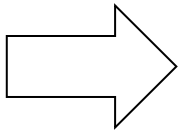
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  a.name, count(distinct p.mid)
FROM    actor a, play p
WHERE   a.id = p.aid
GROUP BY a.id, a.name
```



name	(no name)
Alice	1
Bob	1
Charly	1

6. How to get the number of roles for each male actor

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT
FROM
WHERE
AND
GROUP BY
```

?

name	(no name)
Bob	2
Charly	1

6. How to get the number of roles for each male actor

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

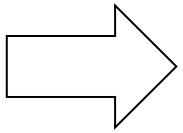
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  a.name, count(distinct a.id)
FROM    actor a, play p
WHERE   a.id = p.aid
AND     a.gender = 'm'
GROUP BY a.id, a.name
```



name	(no name)
Bob	2
Charly	1

7. How to get male actors with number of roles>1

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

Play

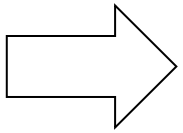
aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT
FROM
WHERE
AND
GROUP BY
```

?



name	(no name)
Bob	2

7. How to get male actors with number of roles>1

Actor

id	name	gender
1	Alice	f
2	Bob	m
3	Charly	m

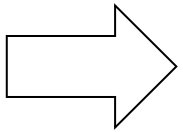
Play

aid	mid	role
1	1	role 1
2	1	role 2
2	1	role 3
3	2	role 4

Movie

id	name
1	Kill Bill
2	Kill Bill

```
SELECT  a.name, count(*)
FROM    actor a, play p
WHERE   a.id = p.aid
AND     a.gender = 'm'
GROUP BY a.id, a.name
HAVING count(*) > 1
```



name	(no name)
Bob	2

Linear regression (as simple example for ML with SQL)

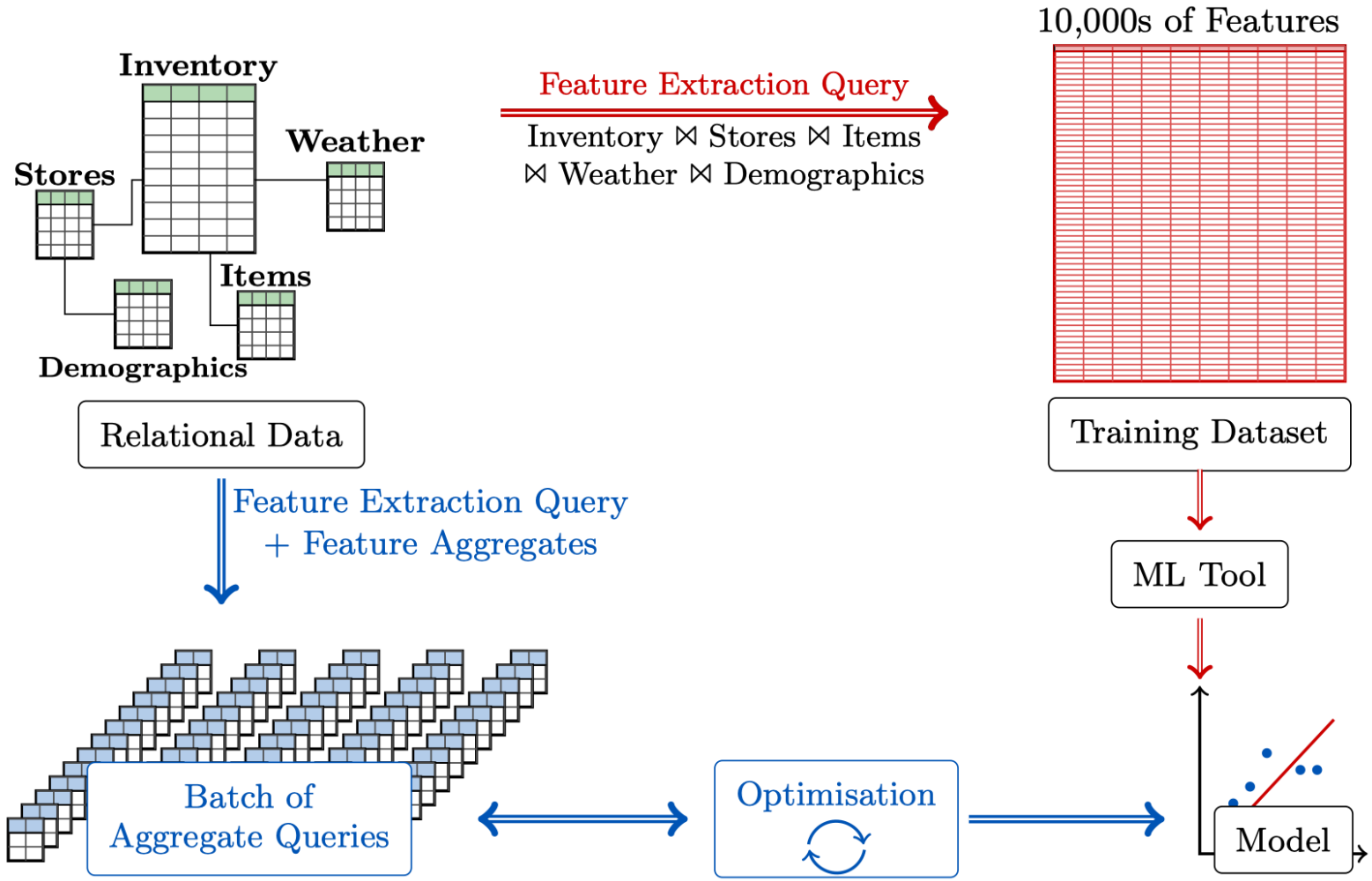


Figure 2: Learning over relational data. In structure-agnostic learning (the top flow in red), a feature extraction query constructs the data matrix on which the model is trained using a machine learning library. In structure-aware learning (the bottom flow in blue), sufficient statistics is computed over the input database using a batch of aggregate queries and then optimisation is performed on this statistics to obtain the model.

Source: Dan Olteanu, "The relational data borg is learning", PVLDB 2020. https://doi.org/10.14778/3415478.3415572
Wolfgang Gatterbauer. Database design: https://northeastern-datalab.github.io/cs3200/

Conjecture

The **learning time and accuracy** of the model can be **drastically improved** by **exploiting the structure and semantics** of the underlying multi-relational database.

Source: Maximilian Schleich, "Learning Models over Relational Databases: A Brief Tutorial," SUM 2019: https://arxiv.org/abs/1911.06577
Wolfgang Gatterbauer. Database design: https://northeastern-datalab.github.io/cs3200/

Pushing "learning" algorithms into the database

- There is a lot of current interest in finding ways to bring ML (Machine Learning) algorithms closer to the data to speed up learning
  - including ways to exploit the structure (think schema) of the data
- Let's look at how you can perform linear regression, one of the simplest models learned from data, with SQL
- We use a famous quartet of datasets

Anscombe's quartet

Four data sets that are identical when examined using simple summary statistics, but which vary considerably when graphed

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Property	Same Value
Mean of *x* in each case	9
Variance of *x* in each case	11
Mean of *y* in each case	~7.50
Variance of *y* in each case	4.122 - 4.127
Correlation between *x* and *y*	0.816
Linear regression line	y = 3 + 0.5x

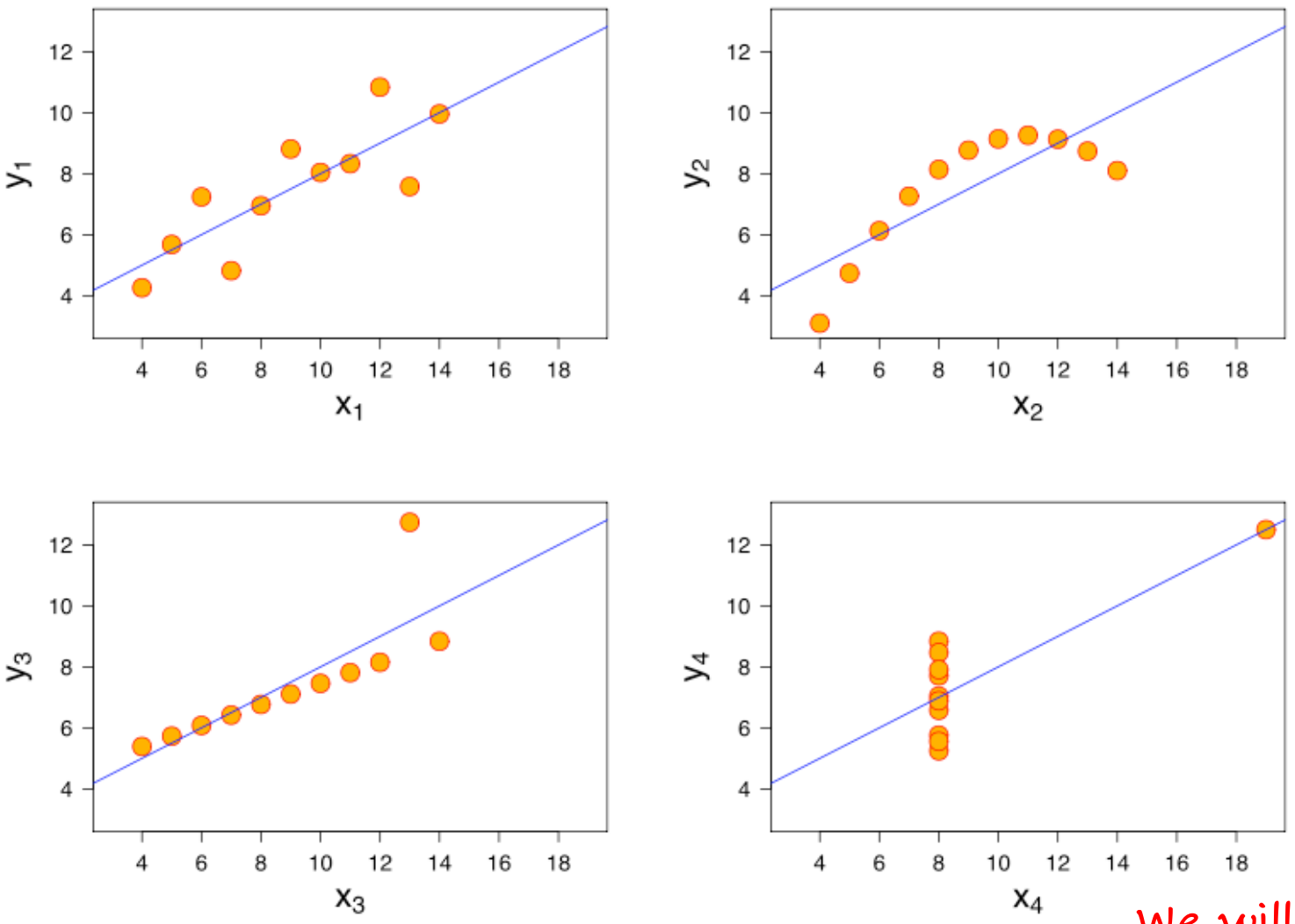
$$y = ax + b$$

We will fit a linear regression line to the data, i.e. calculate slope "a" and intercept "b"

Source: http://en.wikipedia.org/wiki/Anscombe's_quartet, F. J. Anscombe, "Graphs in Statistical Analysis," American Statistician, 1973. https://doi.org/10.1080/00031305.1973.10478966
Wolfgang Gatterbauer. Database design: https://northeastern-datalab.github.io/cs3200/

Anscombe's quartet

Four data sets that are identical when examined using simple summary statistics, but which vary considerably when graphed



Property	Same Value
Mean of *x* in each case	9
Variance of *x* in each case	11
Mean of *y* in each case	~7.50
Variance of *y* in each case	4.122 - 4.127
Correlation between *x* and *y*	0.816
Linear regression line	y = 3 + 0.5x

$$y = ax + b$$

We will fit a linear regression line to the data, i.e. calculate slope "a" and intercept "b"

Source: http://en.wikipedia.org/wiki/Anscombe's_quartet, F. J. Anscombe, "Graphs in Statistical Analysis," American Statistician, 1973. https://doi.org/10.1080/00031305.1973.10478966
Wolfgang Gatterbauer. Database design: https://northeastern-datalab.github.io/cs3200/

Linear Regression

$$y = ax + b$$

$$a = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \qquad \bar{x} = \text{avg}_i(x_i)$$

$$b = \bar{y} - a\bar{x} \qquad \bar{y} = \text{avg}_i(y_i)$$

Data1

x	y
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68

How to calculate slope "a" and intercept "b" with SQL?

?

Linear Regression

$$y = ax + b$$

$$a = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

$$b = \bar{y} - a\bar{x}$$

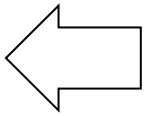
$$\bar{x} = \text{avg}_i(x_i)$$

$$\bar{y} = \text{avg}_i(y_i)$$

Data1

x	y
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68

xm	ym
9	7.5



```
select AVG(x) as xm, AVG(y) as ym
from Data1)
```

Linear Regression

$$y = ax + b$$

$$a = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \qquad \bar{x} = \text{avg}_i(x_i)$$

$$b = \bar{y} - a\bar{x} \qquad \bar{y} = \text{avg}_i(y_i)$$

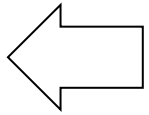
Data1

x	y
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68

X

xm	ym
9	7.5

a
0.5



```
with X as(
  select AVG(x) as xm, AVG(y) as ym
  from Data1)

  select SUM((x-xm)*(y-ym)) /
    SUM((x-xm)*(x-xm)) as a
  from X, Data1
```

Linear Regression

$$y = ax + b$$

$$a = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \qquad \bar{x} = \text{avg}_i(x_i)$$

$$b = \bar{y} - a\bar{x} \qquad \bar{y} = \text{avg}_i(y_i)$$

Data1

x	y
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68

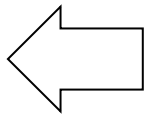
X

xm	ym
9	7.5

Y

a
0.5

a	b
0.5	3.0



```
with X as(
  select AVG(x) as xm, AVG(y) as ym
  from Data1),
Y as(
  select SUM((x-xm)*(y-ym)) /
    SUM((x-xm)*(x-xm)) as a
  from X, Data1)
select a, ym-a*xm as b
from X, Y
```

Linear Regression

$$y = ax + b$$

$$a = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \qquad \bar{x} = \text{avg}_i(x_i)$$

$$b = \bar{y} - a\bar{x} \qquad \bar{y} = \text{avg}_i(y_i)$$

Data1

x	y	xm	ym
10	8.04	9	7.5
8	6.95	9	7.5
13	7.58	9	7.5
9	8.81	9	7.5
11	8.33	9	7.5
14	9.96	9	7.5
6	7.24	9	7.5
4	4.26	9	7.5
12	10.84	9	7.5
7	4.82	9	7.5
5	5.68	9	7.5

X

Y

xm	ym	a
9	7.5	0.5

a	b
0.5	3.0

An alternative:

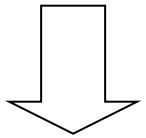
```
with X as(
  select  *, AVG(x) over() xm, AVG(y) over() ym
  from Data1),
Y as(
  select MIN(xm) xm, MIN(ym) ym,
    SUM((x-xm)*(y-ym)) / SUM((x-xm)*(x-xm)) a
  from X)
select a, ym-a*xm as b
from Y
```

Prepared statements & SQL Injection

Differences to minimum price

Purchase

product	price	quantity
Apple	3	20
Apple	2	20
Banana	1	50
Banana	2	10
Banana	4	10



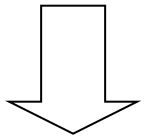
?

```
select  *
from  Purchase
where  product='Apple'
```

Differences to minimum price

Purchase

product	price	quantity
Apple	3	20
Apple	2	20
Banana	1	50
Banana	2	10
Banana	4	10



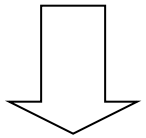
product	price	quantity
Apple	3	20
Apple	2	20

```
select  *
from  Purchase
where  product='Apple'
```

Differences to minimum price

Purchase

product	price	quantity
Apple	3	20
Apple	2	20
Banana	1	50
Banana	2	10
Banana	4	10



product	price	quantity
Apple	3	20
Apple	2	20

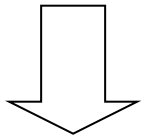
```
PREPARE  myquery(varchar) as
  select  *
  from  Purchase
  where  product=$1;
```

```
EXECUTE  myquery('Apple')
```

Differences to minimum price

Purchase

product	price	quantity
Apple	3	20
Apple	2	20
Banana	1	50
Banana	2	10
Banana	4	10



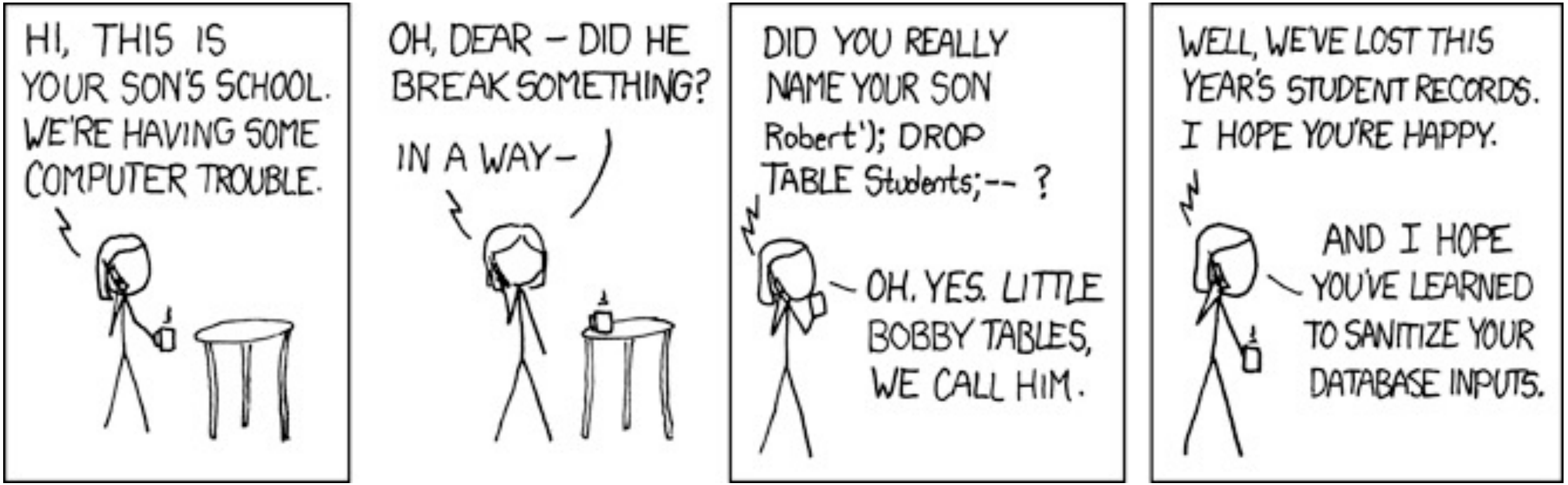
product	price	quantity
Banana	1	50
Banana	2	10
Banana	4	10

```
EXECUTE myquery('Banana')
```

```
DEALLOCATE myquery
```

For more information see https://www.postgresql.org/docs/current/sql-prepare.html

What happened here?



What does this SQL do:

```
Robert'); DROP
TABLE STUDENTS; --
```

Source: http://xkcd.com/327/, http://stackoverflow.com/questions/332365/xkcd-sql-injection-please-explain

It's called SQL injection: Version 1

Let's say the name was used in a variable, `$Name`. You then run this query:

```
INSERT INTO Students VALUES ( '$Name' )
```

What you get is:

```
INSERT INTO Students VALUES ( 'Robert' );  DROP TABLE STUDENTS; --')
```

The `--` only comments the remainder of the line.

Source: http://xkcd.com/327/, http://stackoverflow.com/questions/332365/xkcd-sql-injection-please-explain

It's called SQL injection: Version 2

It drops the students table.

The original query in the school's program probably looks something like

```
var query = "SELECT * FROM Students WHERE (Name = '" + tbName.Text + "')";
```

This is the naive way to add user text to a query, and is *very bad*. So bad, one might even say *evil*. Since the student's name is `Robert'); DROP TABLE STUDENTS; --` the resulting query (after concatenation) is

```
SELECT * FROM Students WHERE (Name = 'Robert'); DROP TABLE Students; --')
```

which, in plain English, roughly translates to the two queries:

> Get everything from the Students table where the student's name is Robert.

and

> Delete the Students table and ignore everything else I say from this point on ') and any other query-breaking junk.

Source: http://xkcd.com/327/, http://stackoverflow.com/questions/332365/xkcd-sql-injection-please-explain

SQL injection

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

```
' or '1'='1
' or '1'='1' -- '
' or '1'='1' ({ '
' or '1'='1' /* '
```

```
SELECT * FROM users WHERE name = '' OR '1'='1';
```

```
SELECT * FROM users WHERE name = '' OR '1'='1' -- ';
```

Source: http://en.wikipedia.org/wiki/SQL_injection
Wolfgang Gatterbauer. Database design: https://northeastern-datalab.github.io/cs3200/

TJX Credit Card Numbers Theft

- In 2006, a $17.5 billion Fortune 500 firm
- Unauthorized intrusion resulting in lost of credit/debit card information
  - From May 2006 to January 2007?
  - From July 2005 to December 2006?
- TJX's overall losses: $1.35 billion – $4.5 billion

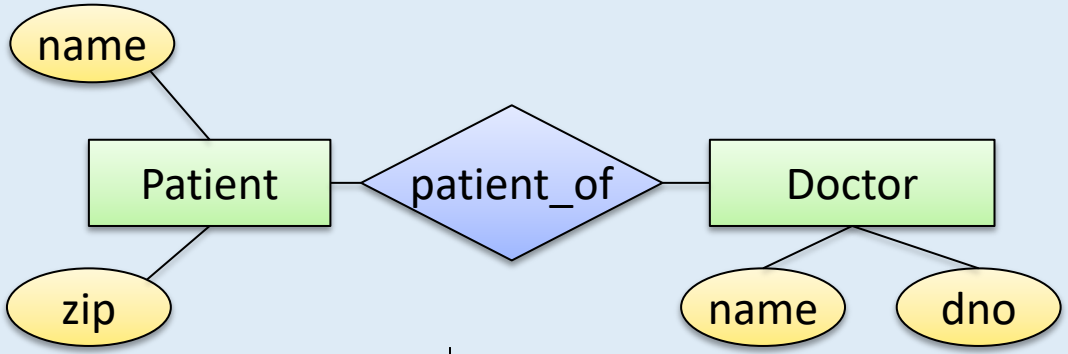
More pointers in case you are want to learn more:
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
http://en.wikipedia.org/wiki/SQL_injection

Database design
= Relational Data Modeling

Data modeling and Database Design Process

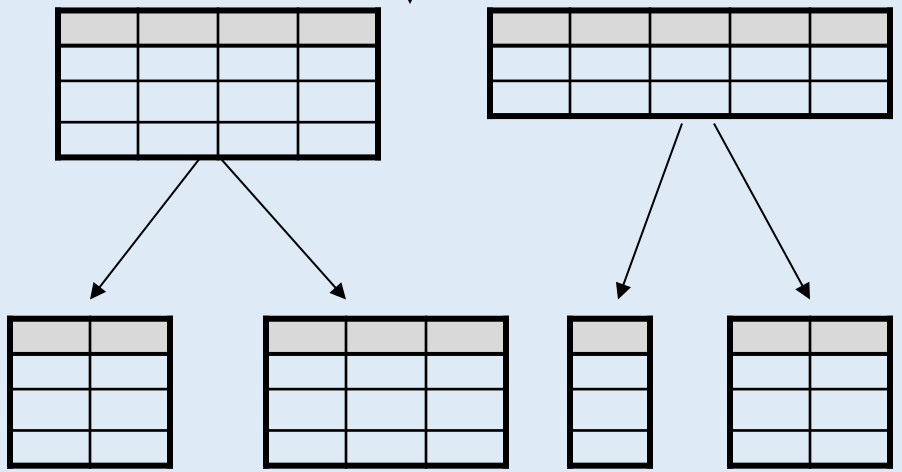
1. ER Diagram

Conceptual Model:
("technology independent")
describe main data items



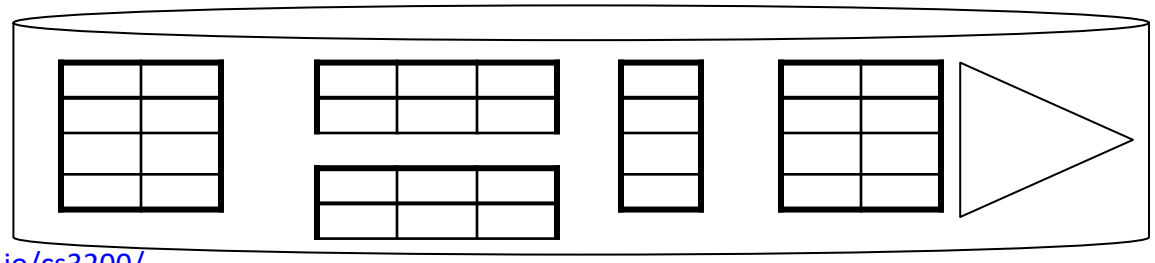
2. Relational Database Design

Logical Model
("for relational databases"):
Tables, Constraints
Functional Dependencies
Normalization:
Eliminates anomalies



3. Database Implementation

Physical Model
Physical storage details
Result: Physical Schema



Graphicacy

"Graphicacy is concerned with the capacities people require in order to interpret and generate information in the form of graphics."

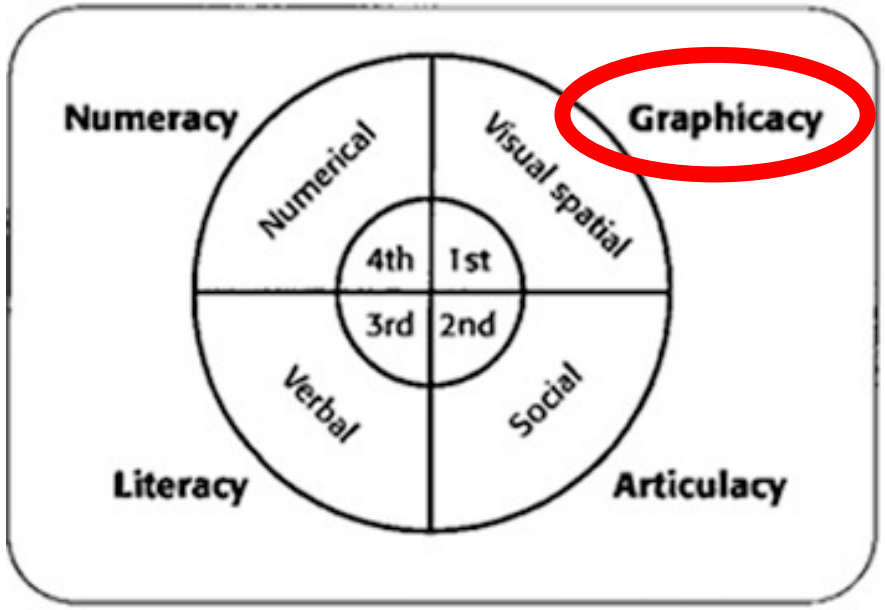
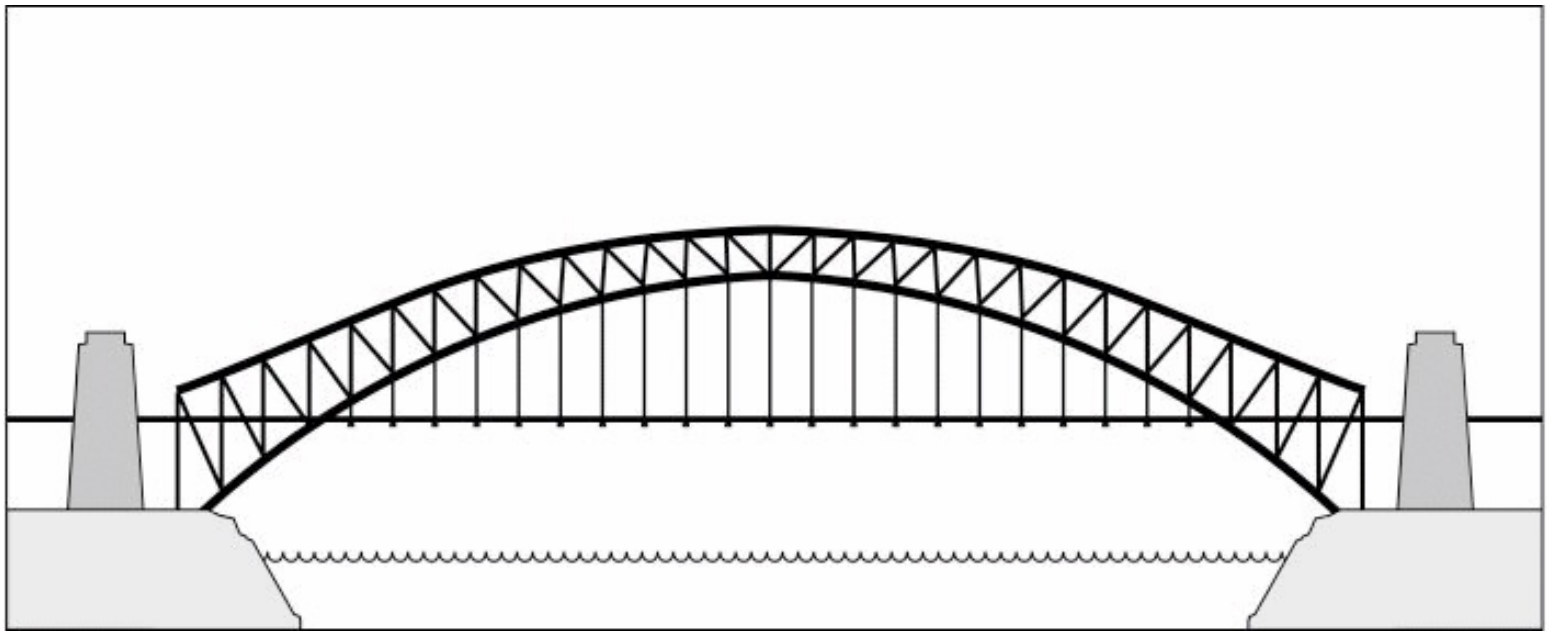


Figure 2. Balchin's "four types of ability."



Source: W.G.V Balchin, Graphicacy 1976, http://en.wikipedia.org/wiki/Graphicacy

9	W Oct 5	top-k (limit), recursion, SQL injection	SDK 4.4	G4, HW3
	M Oct 10	holiday		
10	W Oct 12	**Exam 1**		HW4
		Database Design and Normal Forms		
11	M Oct 17	ER diagrams: entities and relationships	SDK 6.1-6.5.2, 6.6, 7.9.1-7.9.3	
12	W Oct 19	ER diagrams: associative entities	SDK 6.9	G5
13	M Oct 24	ER diagrams: weak entities, connection traps	SDK 6.5.3	
14	W Oct 26	relational modeling, ERD notation overview & textbook figures	SDK, 6.7, 6.10	HW5, P0
15	M Oct 31	relational modeling: entities, relationships, associative entities	SDK 6.7	
16	W Nov 2	ONLINE CLASS, relational modeling: weak entities, enhanced ERD = subtypes & translation	SDK 6.8	G6, P1
17	M Nov 7	normalization: normal forms and FDs, BCNF, decompositions	SDK 7.8, 7.2, 7.3, 7.3.2, 7.5.2, SDK 7.3.1, 7.3.3, 7.1, 7.2	
18	W Nov 9	**Exam 2**		G7, HW6
		Transaction Processing		
19	M Nov 14	transactions: ACID, logging	SDK 17.1-17.4	
20	W Nov 16	concurrency: interleaving, conflict serializability	SDK 17.5-17.6, Stanford book Ch 18 (in Canvas)	P2
21	M Nov 21	locking, 2PL, recoverability, strict 2PL, deadlocks	SDK 17.7, 18.1.1-18.1.3, 18.2.2.1, Stanford book Ch 18 (in Canvas)	
	W Nov 23	holiday		