L05: SQL

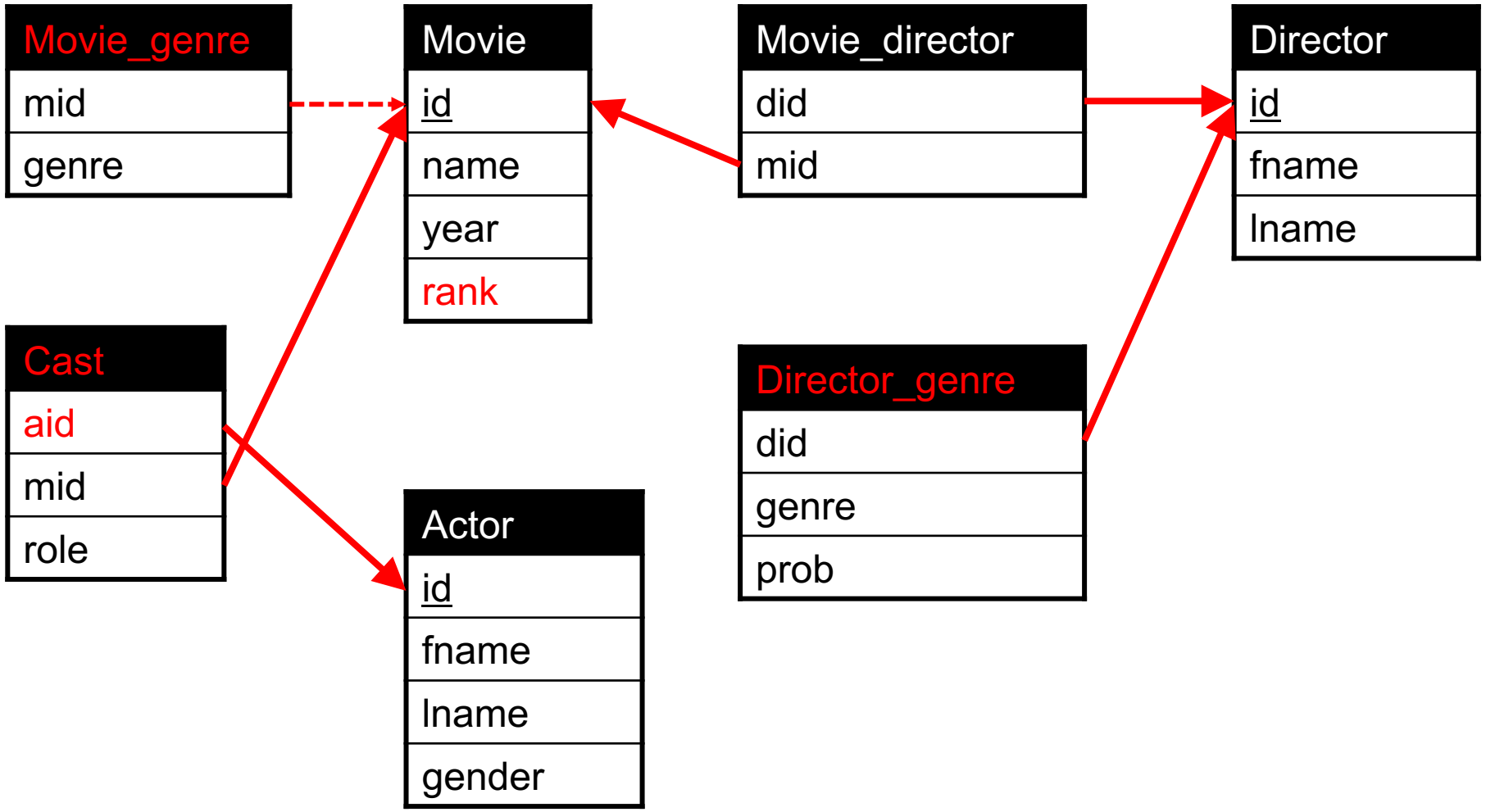
Announcements!

- HW1 is due tonight
- HW2 groups are assigned

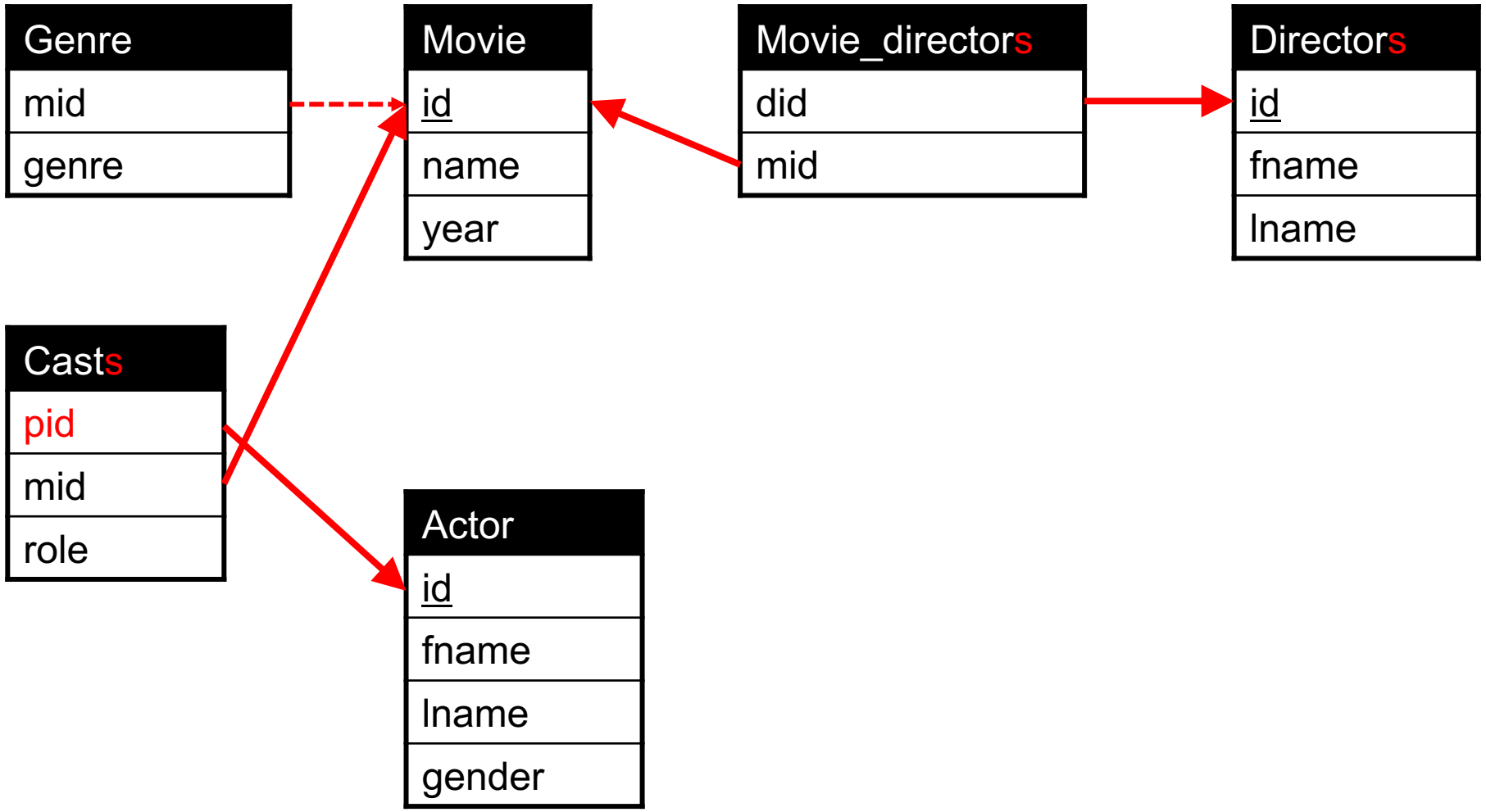
- Outline today:
  - nested queries and witnesses
  - We start with a detailed example!
  - outer joins, nulls?

Small IMDB schema (SQLite)

300



Big IMDB schema (Postgres)



Theta joins

What do these queries compute?

R
a
1
2

U
a
2
3
4

305

```
SELECT   R.a, U.a
FROM     R, U
WHERE    R.a < U.a
```

⇨ ?

```
SELECT   R.a, U.a
FROM     R, U
WHERE    R.a <= U.a
```

⇨ ?

A **Theta-join** allows for arbitrary comparison relationships (such as ≥).
An **equijoin** is a theta join using the equality operator.

Theta joins

305

R

a
1
2

U

a
2
3
4

What do these queries compute?

```
SELECT   R.a, U.a as b
FROM     R, U
WHERE    R.a < U.a
```

a	b
1	2
1	3
1	4
2	3
2	4

```
SELECT   R.a, U.a as b
FROM     R, U
WHERE    R.a >= U.a
```

a	b
2	2

A **Theta-join** allows for arbitrary comparison relationships (such as ≥).
An **equijoin** is a theta join using the equality operator.

Witnesses: with joins (1/6)

315

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Witnesses: with joins (2/6)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Our Plan:

- 1. Compute max price in a subquery for a given company

S max(price)
F P
W cid = 1

Witnesses: with joins (2/6)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Our Plan:

- 1. Compute max price in a subquery for a given company

1.
```
SELECT  max(P1.price)
FROM    Product2 P1
WHERE   P1.cid = 1
```

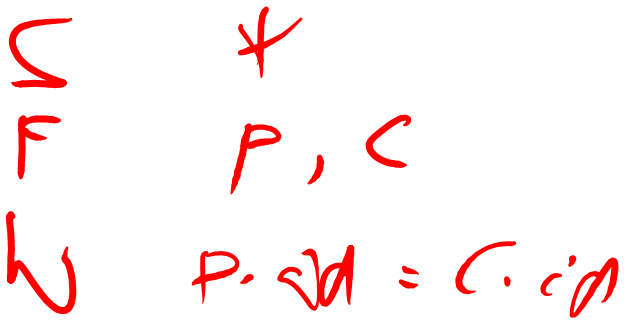
Witnesses: with joins (3/6)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Our Plan:

- 1. Compute max price in a subquery for a given company
- 2. Compute each product and its price, per company



1.
```
SELECT  max(P1.price)
FROM    Product2 P1
WHERE   P1.cid = 1
```

Witnesses: with joins (3/6)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Our Plan:

- 1. Compute max price in a subquery for a given company
- 2. Compute each product and its price, per company

2.
```
SELECT  C2.cname, P2.pname, P2.price
FROM    Company2 C2, Product2 P2
WHERE   C2.cid = P2.cid
```

1.
```
SELECT  max(P1.price)
FROM    Product2 P1
WHERE   P1.cid = 1
```

Witnesses: with joins (3/6)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Our Plan:

- 1. Compute max price in a subquery for a given company
- 2. Compute each product and its price, per company
- 3. Compare the price with the max price

2.
```
SELECT  C2.cname, P2.pname, P2.price
FROM    Company2 C2, Product2 P2
WHERE   C2.cid = P2.cid
```

1.
```
SELECT  max(P1.price)
FROM    Product2 P1
WHERE   P1.cid = 1
```

Witnesses: with joins (4/6)

315

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Our Plan:

- 1. Compute max price in a subquery for a given company
- 2. Compute each product and its price, per company
- 3. Compare the price with the max price

```
SELECT  C2.cname, P2.pname, P2.price
FROM    Company2 C2, Product2 P2
WHERE   C2.cid = P2.cid
    and P2.price =
        (SELECT max(P1.price)
        FROM    Product2 P1
        WHERE   P1.cid = C2.cid)
```

How many aliases do we actually need?

Witnesses: with joins (5/6)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Our Plan:

- 1. Compute max price in a subquery for a given company
- 2. Compute each product and its price, per company
  and compare the price with the max price

```
SELECT  cname, pname, price
FROM    Company2, Product2
WHERE   Company2.cid = Product2.cid
    and price =
        (SELECT max(price)
        FROM    Product2
        WHERE   cid = Company2.cid)
```

We need no single alias here.

Next: can we eliminate the max operator in the inner query?

Witnesses: with joins (6/6)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Our Plan:

- 1. Compute all prices in a subquery, for a given company
- 2. Compute each product and its price, per company
  and compare the price with the all prices

```
SELECT  cname, pname, price
FROM    Company2, Product2
WHERE   Company2.cid = Product2.cid
    and price >= ALL
        (SELECT price
        FROM    Product2
        WHERE   cid = Company2.cid)
```

But: "ALL" does not work in SQLite ☹

Witnesses: with FROM (1/3)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Another Plan:

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

F

1.
```
SELECT   cid, max(price) as MP
FROM     Product2
GROUP    BY cid
```
) x, P, C

Finding the maximum price for each company was easy.
But we want the "witnesses", i.e. the products with max price.

Witnesses: with FROM (2/3)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Another Plan:

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

2.
```
SELECT  C2.cname, P2.pname, X.MP
FROM    Company2 C2, Product2 P2,
        (SELECT  cid, max(price) as MP
        FROM     Product2
        GROUP    BY cid) as X
WHERE   C2.cid = P2.cid
    and C2.cid = X.cid
    and P2.price = X.MP
```

Let's write the same query with a "WITH" clause

Witnesses: with FROM (3/3)

Product2 (pname, price, cid)
Company2 (cid, cname, city)

Q: For each company, find the most expensive product + its price

Another Plan with WITH:

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

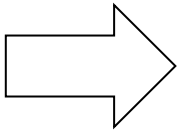
```
WITH    X(cid, MP) as
        (SELECT  cid, max(price)
        FROM     Product2
        GROUP    BY cid)
SELECT  C2.cname, P2.pname, X.MP
FROM    Company2 C2, Product2 P2, X
WHERE   C2.cid = P2.cid
    and C2.cid = X.cid
    and P2.price = X.MP
```

Witnesses: with aggregates per group (1/8)

First: How to get the product that is sold with maximum price?

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10



Product	mp
Banana	4

```
SELECT   product, max(price) as mp
FROM
WHERE
GROUP BY
HAVING
```

???

Witnesses: with aggregates per group (2/8)

First: How to get the product that is sold with maximum price?

Purchase *1) Find the maximum price*

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10

(no name)
4

```
SELECT   max(price)
FROM     Purchase
```

Witnesses: with aggregates per group (3/8)

308

First: How to get the product that is sold with maximum price?

Purchase *2) Now you need to find product with price = maximum price*

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10

Product	mp
Banana	4

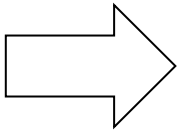
```
SELECT  P2.product, P2.price as mp
FROM    Purchase P2
WHERE   P2.price = (
            SELECT  max(price)
            FROM    Purchase
)
```

Witnesses: with aggregates per group (4/8)

First: How to get the product that is sold with maximum price?

Purchase *Another way to formulate this query*

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10



Product	mp
Banana	4

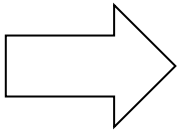
```
SELECT   P2.product, P2.price as mp
FROM     Purchase P2
WHERE    P2.price >= ALL (
             SELECT   price
             FROM     Purchase
)
```

Witnesses: with aggregates per group (5/8)

Second: How to get the product that is sold with max sales (=quantities sold)?

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10



Product	sales
Banana	70

```
SELECT
FROM
WHERE
GROUP BY
HAVING
```

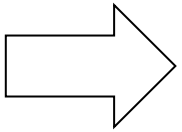
???

Witnesses: with aggregates per group (6/8)

Second: How to get the product that is sold with max sales (=quantities sold)?

Purchase *1: find the total sales (sum of quantity) for each product*

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10



Product	sales
Bagel	40
Banana	70

```
SELECT    product, sum(quantity) as sales
FROM      Purchase
GROUP BY product
```

Witnesses: with aggregates per group (7/8)

308

Second: How to get the product that is sold with max sales?

2: we can use the same query as nested query

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10

(no name)
40
70

```
SELECT    sum(quantity)
FROM      Purchase
GROUP BY product
```

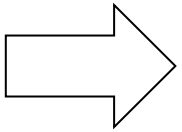
Witnesses: with aggregates per group (8/8)

308

Second: How to get the product that is sold with max sales?

Purchase *3: putting things together*

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10



Product	sales
Banana	70

```
SELECT    product, sum(quantity) as sales
FROM      Purchase
GROUP BY product
HAVING    sum(quantity) >= ALL (
            SELECT    sum(quantity)
            FROM      Purchase
            GROUP BY product
          )
```

Next: Can you write the query without the "ALL" quanitfier?

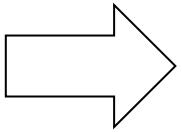
Witnesses: with aggregates per group (8/8)

Second: How to get the product that is sold with max sales?

Another way to formulate this query without "ALL"

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	2	20
Banana	1	50
Banana	2	10
Banana	4	10



Product	sales
Banana	70

```
SELECT    product, sum(quantity) as sales
FROM      Purchase
GROUP BY  product
HAVING    sum(quantity) =
          (SELECT max (Q)
          FROM
                (SELECT    sum(quantity) Q
                FROM       Purchase
                GROUP BY product) X
          )
```

Understanding nested queries

More SQL Queries

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 5.1 An Instance $S3$ of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 5.2 An Instance $R2$ of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 5.3 An Instance $B1$ of Boats

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

More nested Queries 1

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have reserved a red boat.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN
            ( SELECT R.sid
            FROM Reserves R
            WHERE R.bid IN
                        ( SELECT B.bid
                        FROM Boats B
                        WHERE B.color = 'red'))
```

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

More nested Queries 2

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have reserved a boat that is not red.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN
        ( SELECT R.sid
        FROM Reserves R
        WHERE R.bid not IN
                ( SELECT B.bid
                FROM Boats B
                WHERE B.color = 'red'))
```

They must have reserved at least one boat in another color

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

More nested Queries 3

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have not reserved a red boat.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid not IN
            ( SELECT R.sid
            FROM Reserves R
            WHERE R.bid IN
                        ( SELECT B.bid
                        FROM Boats B
                        WHERE B.color = 'red'))
```

They can have reserved 0 or more boats in another color, but must not have reserved any red boat

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

More nested Queries 4

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

= Find the names of sailors who have reserved only red boats

Q: Find the names of sailors who have not reserved a boat that is not red.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid not IN
            ( SELECT R.sid
            FROM Reserves R
            WHERE R.bid not IN
                        ( SELECT B.bid
                        FROM Boats B
                        WHERE B.color = 'red'))
```

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

More nested Queries 5

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

= Find the names of sailors who have reserved all red boats

Q: Find the names of sailors so there is no red boat that is not reserved by him.

```
SELECT S.sname
FROM Sailors S
WHERE not exists
        ( SELECT B.bid
        FROM Boats B
        WHERE B.color = 'red'
        AND not exists
                ( SELECT R.bid
                FROM Reserves R
                WHERE R.bid = B.bid
                AND R.sid = S.sid))
```

To understand semantics of nested queries, think of a nested loops evaluation: For each Sailors tuple, check the qualification by computing the subquery

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

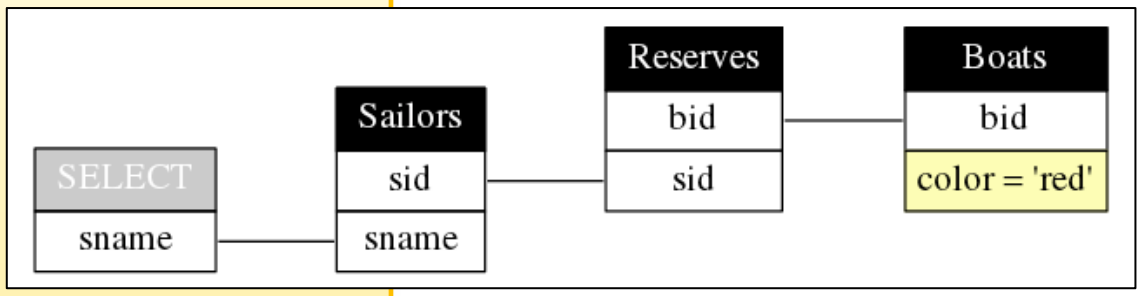
Once more: 1

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have reserved a red boat.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN
            ( SELECT R.sid
            FROM Reserves R
            WHERE R.bid IN
                        ( SELECT B.bid
                        FROM Boats B
                        WHERE B.color = 'red'))
```



Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

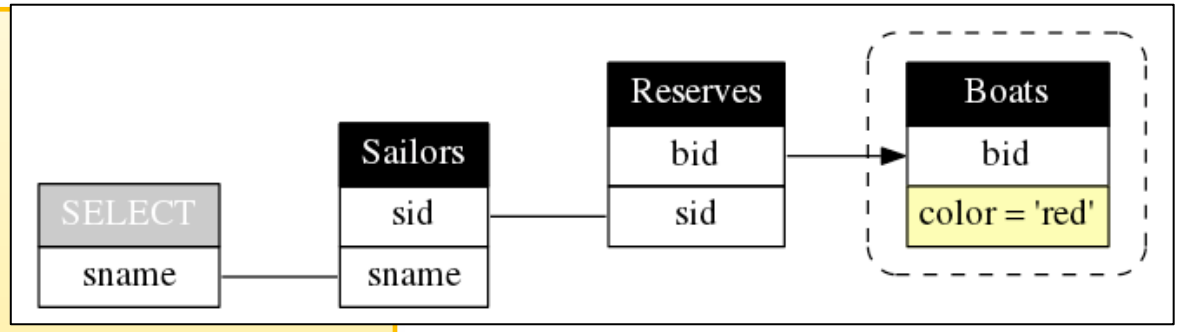
Once more: 2

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have reserved a boat that is not red.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN
            ( SELECT R.sid
            FROM Reserves R
            WHERE R.bid not IN
                        ( SELECT B.bid
                        FROM Boats B
                        WHERE B.color = 'red'))
```



Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

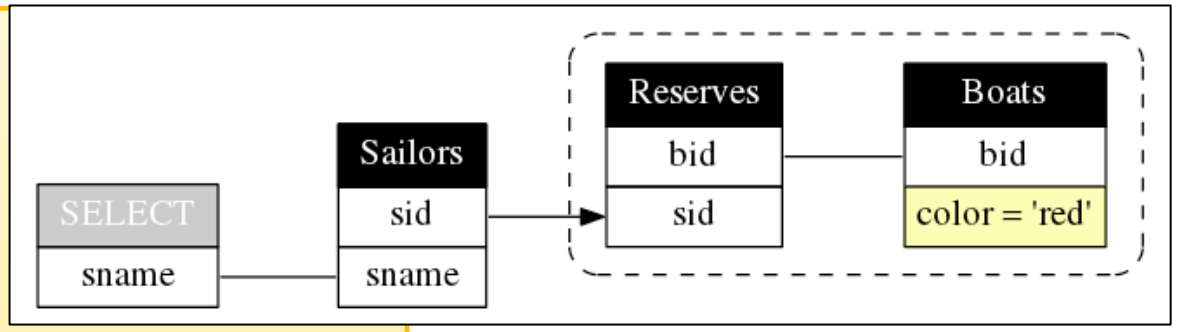
Once more: 3

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have not reserved a red boat.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid not IN
            ( SELECT R.sid
            FROM Reserves R
            WHERE R.bid IN
                        ( SELECT B.bid
                        FROM Boats B
                        WHERE B.color = 'red'))
```



Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

Once more: 4

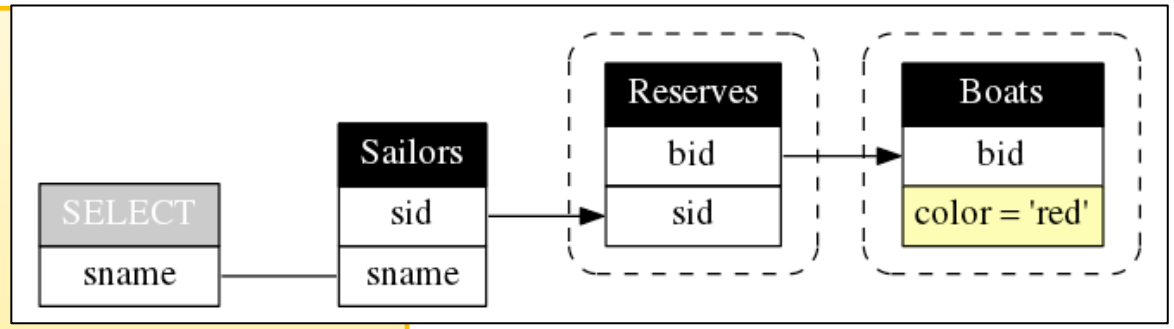
Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

= Find the names of sailors who have reserved only red boats

Q: Find the names of sailors who have not reserved a boat that is not red.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid not IN
            ( SELECT R.sid
            FROM Reserves R
            WHERE R.bid not IN
                        ( SELECT B.bid
                        FROM Boats B
                        WHERE B.color = 'red'))
```



Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

Once more: 5

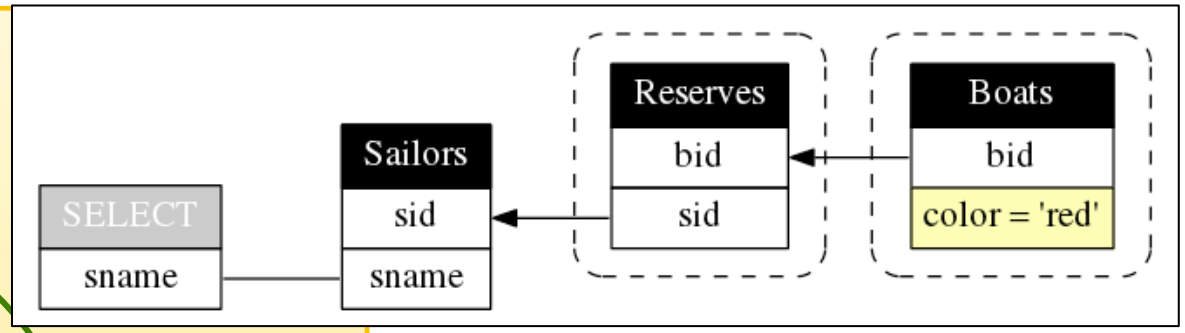
Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

= Find the names of sailors who have reserved all red boats

Q: Find the names of sailors so there is no red boat that is not reserved by him.

```
SELECT S.sname
FROM Sailors S
WHERE not exists
        ( SELECT B.bid
        FROM Boats B
        WHERE B.color = 'red'
        AND not exists
                ( SELECT R.bid
                FROM Reserves R
                WHERE R.bid = B.bid
                AND R.sid = S.sid))
```



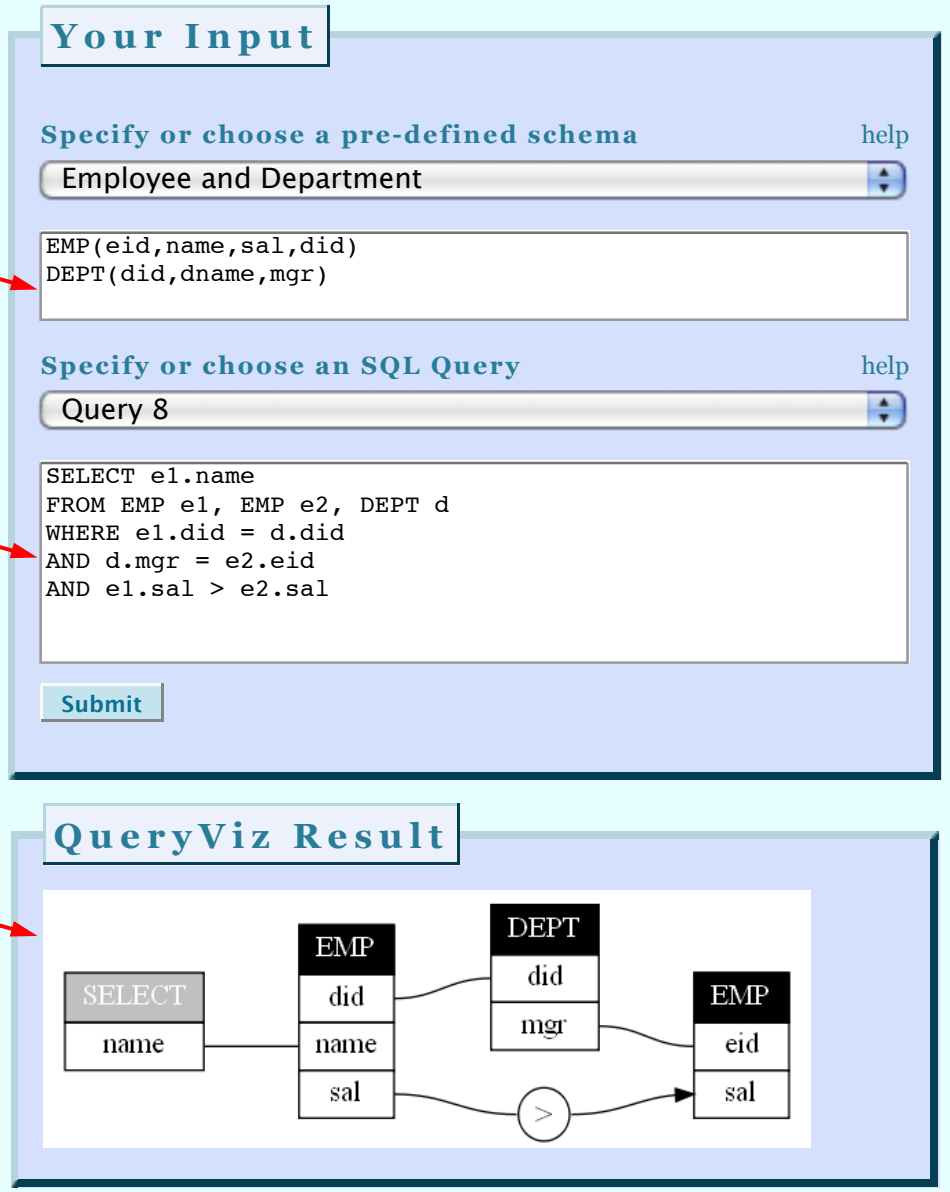
Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

http://queryviz.com

Input: Schema

Input Query

Output: Visualization



Multiset operations (Intersect, Except)

Recall Multisets (Bags)

Multiset X

Tuple
(1, a)
(1, a)
(1, b)
(2, c)
(2, c)
(2, c)
(1, d)
(1, d)

Equivalent Representations of a **Multiset**

$\boldsymbol{\lambda(X)}$= *"Count of tuple in X" (Items not listed have implicit count 0)*

Multiset X

Tuple	$\boldsymbol{\lambda(X)}$
(1, a)	2
(1, b)	1
(2, c)	3
(1, d)	2

Note: In a set all counts are {0,1}.

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

$\cap$

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

$=$

Multiset Z

Tuple	$\lambda(Z)$
(1, a)	2
(1, b)	0
(2, c)	2
(1, d)	0

$$\lambda(Z) = min(\lambda(X), \lambda(Y))$$

For sets, this is **intersection**

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

$\cup$

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

$=$

Multiset Z

Tuple	$\lambda(Z)$
(1, a)	7
(1, b)	1
(2, c)	5
(1, d)	2

$$\lambda(Z) = \lambda(X) + \lambda(Y)$$

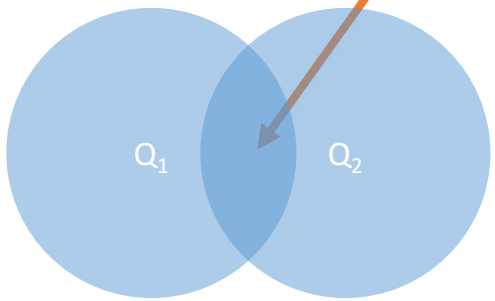
For sets,
this is **union**

Multiset Operations in SQL

Explicit Set Operators: INTERSECT

```
SELECT R.A
FROM   R, S
WHERE  R.A=S.A
INTERSECT
SELECT R.A
FROM   R, T
WHERE  R.A=T.A
```

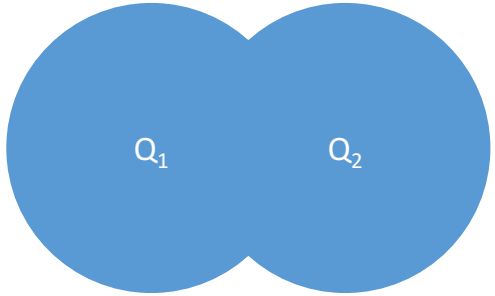
$$\{r.A \mid r.A = s.A\} \cap \{r.A \mid r.A = t.A\}$$



UNION

```
SELECT  R.A
FROM    R, S
WHERE   R.A=S.A
UNION
SELECT R.A
FROM    R, T
WHERE   R.A=T.A
```

$$\{r.A \mid r.A = s.A\} \cup \{r.A \mid r.A = t.A\}$$



Why aren't there duplicates?

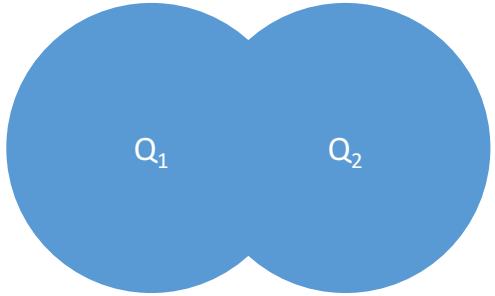
By default: SQL uses set semantics for INTERSECT and UNION!

What if we want duplicates?

UNION ALL

```
SELECT  R.A
FROM    R, S
WHERE   R.A=S.A
UNION ALL
SELECT R.A
FROM    R, T
WHERE   R.A=T.A
```

$$\{r.A \mid r.A = s.A\} \cup \{r.A \mid r.A = t.A\}$$

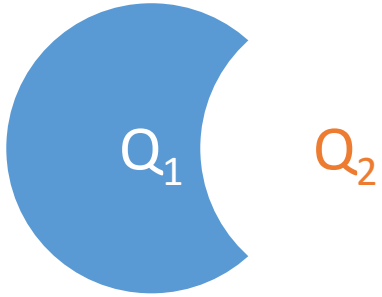


ALL indicates Multiset operations

EXCEPT

```
SELECT R.A
FROM   R, S
WHERE  R.A=S.A
EXCEPT
SELECT R.A
FROM   R, T
WHERE  R.A=T.A
```

$$\{r.A \mid r.A = s.A\} \backslash \{r.A | r.A = t.A\}$$



What is the multiset version?

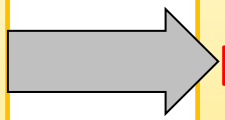
INTERSECT and EXCEPT*

R(a,b)
S(a,b)

```
(SELECT R.a, R.b
 FROM   R)

INTERSECT

(SELECT S.a, S.b
 FROM   S)
```



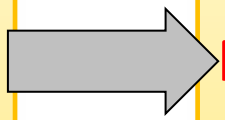
```
SELECT  R.a, R.b
FROM    R
WHERE
EXISTS  (SELECT *
         FROM   S
         WHERE  R.a=S.a
         and    R.b=S.b)
```

If R, S have no duplicates, then can write without sub-queries (HOW?)

```
(SELECT R.A, R.B
 FROM   R)

EXCEPT

(SELECT S.A, S.B
 FROM   S)
```



```
SELECT R.A, R.B
FROM   R
WHERE
NOT  EXISTS  (SELECT *
              FROM   S
              WHERE  R.A=S.A
              and    R.B=S.B)
```

*Not in all DBMSs. (SQLlite does not like the parentheses, Oracle uses "MINUS" instead of "EXCEPT")