L06: SQL

Announcements!

- Please pick up your name card
  - always come with your name card
  - If nobody answers my question, I will likely pick on those without a namecard or in the last row
- Polls on speed: we slow down and have another SQL lecture (likely no NoSQL)
- Use the anonymous feedback form
- HW3 and later: in teams

- Outline today:
  - HW1 together
  - outer joins, nulls

A word on capitalization

302

Product (pname, price, category, manufacturer)
Company (cname, stockprice, country)

Q: Find all US companies that manufacture products in the 'Gadgets' category!

```
SELECT   cname
FROM     Product P, Company
WHERE    country = 'USA'
AND      P.category = 'Gadgets'
AND      P.manufacturer = cname
```

My recommendation for capitalization

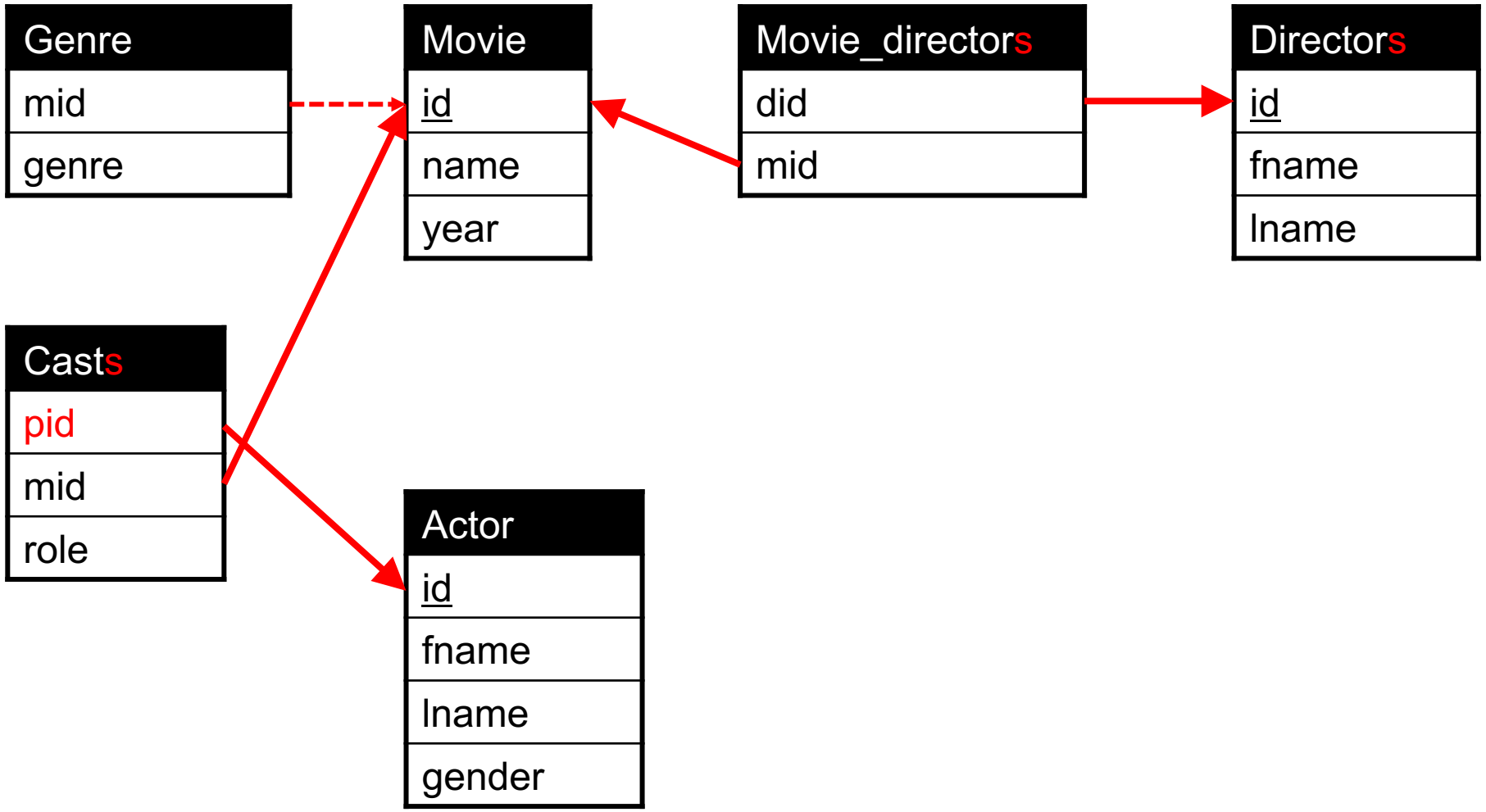
1. SQL keywords in ALL CAPS,
2. Table names with Initial Caps
3. Column names all in lowercase.

PostgreSQL treats all in lowercase.
Except if you write:
create table "Product" (…)
This will preserve capitalization of table name
But … you need to always use quotations

More information: http://blog.lerner.co.il/quoting-postgresql/ , https://stackoverflow.com/questions/6331504/omitting-the-double-quote-to-do-query-on-postgresql

HW1

Big IMDB schema (Postgres)



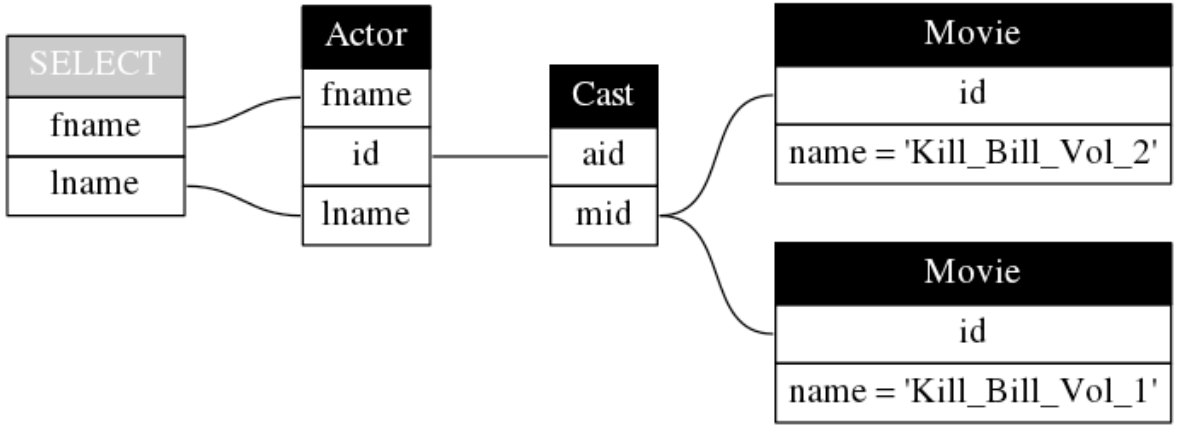
Quiz

IMDB (postgres)

Find the first/last names of all actors who appeared in both of the following movies: Kill Bill: Vol. 1 and Kill Bill: Vol. 2.

```
SELECT DISTINCT A.fname, A.lname
FROM    Actor A, Casts C, Movie M1, Movie M2
WHERE   M1.name = 'Kill Bill: Vol. 1'
    and M2.name = 'Kill Bill: Vol. 2'
    and M1.id = C.mid
    and M2.id = C.mid
    and C.pid = A.id
```

☹



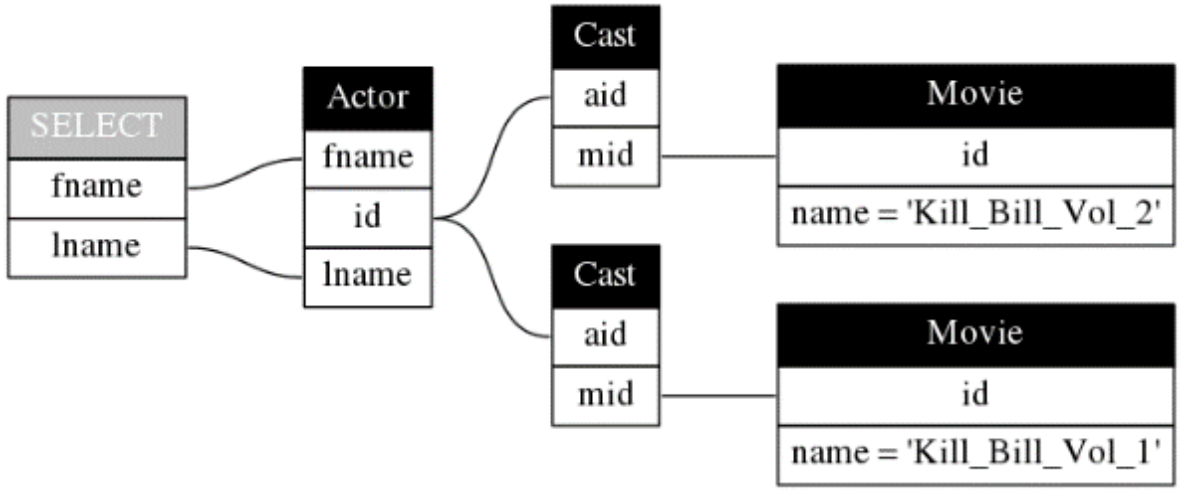
Picture Source: http://queryviz.com/online

Quiz

IMDB (postgres)

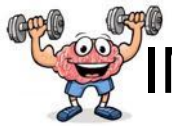
Find the first/last names of all actors who appeared in both of the following movies: Kill Bill: Vol. 1 and Kill Bill: Vol. 2.

```
SELECT DISTINCT A.fname, A.lname
FROM    Actor A, Casts C, Movie M1, Movie M2, Casts C2
WHERE   M1.name = 'Kill Bill: Vol. 1'
    and M2.name = 'Kill Bill: Vol. 2'
    and M1.id = C.mid
    and M2.id = C2.mid
    and C.pid = A.id
    and C2.pid = A.id
```



Picture Source: http://queryviz.com/online

Quiz



IMDB (postgres)

Find the first/last names of all actors who appeared in both of the following movies: Kill Bill: Vol. 1 and Kill Bill: Vol. 2.

```
SELECT  A.id, A.lname, A.fname,
FROM    actor A, cast C, movie M
WHERE   M.id = C.mid
  AND   A.id = C.pid
  AND   (M.name = 'Kill Bill: Vol. 1'
         OR M.name = 'Kill Bill: Vol. 2')
GROUP BY A.id, A.lname, A.fname
HAVING  count(M.id) > 1
```

What if an actor played two roles in Kill Bill 1?

Null Values

3-valued logic example

- Three logicians walk into a bar. The bartender asks: "Do all of you want a drink?"
- The 1st logician says: "I don't know."
- The 2nd logician says: "I don't know."
- The 3rd logician says: "Yes!"

Nulls in SQL

- Whenever we don't have a value, we can put a NULL
- Can mean many things:
  - Value does not exists
  - Value exists but is unknown
  - Value not applicable
  - Etc.
- The schema specifies for each attribute if it can be NULL (nullable attribute) or not
- How does SQL cope with tables that have NULLs ?

Null Values

- In SQL there are three Boolean values:
  - FALSE, TRUE, UNKNOWN

- If x= NULL then
  - Arithmetic operations produce NULL. E.g: 4*(3-x)/7
  - Boolean conditions are also NULL. E.g: x='Joe'
  - aggregates ignore NULL values

- Logical reasoning:
  - FALSE = 0
  - TRUE = 1
  - UNKNOWN = 0.5

x AND y = min(x,y)

x OR y = max(x,y)

NOT x = (1 – x)

Null Values: example

```
SELECT  *
FROM    Person
WHERE   (age < 25)
    and (height > 6 or weight > 190)
```

Person

Age	Height	Weight
20	NULL	200
NULL	6.5	170

Null Values: example

```
SELECT  *
FROM    Person
WHERE   (age < 25)
    and (height > 6 or weight > 190)
```

Person

Age	Height	Weight
20	NULL	200
~~NULL~~	~~6.5~~	~~170~~

Rule in SQL:
include only tuples that
yield TRUE

Null Values: example

```
SELECT  *
FROM    Person
WHERE   (age < 25)
    and (height > 6 or weight > 190)
```

Person

Age	Height	Weight
20	NULL	200
~~NULL~~	~~6.5~~	~~170~~

Rule in SQL:
include only tuples that yield TRUE

```
SELECT  *
FROM    Person
WHERE   age < 25  or age >= 25
```

Null Values: example

```
SELECT  *
FROM    Person
WHERE   (age < 25)
    and (height > 6 or weight > 190)
```

Person

Age	Height	Weight
20	NULL	200
~~NULL~~	~~6.5~~	~~170~~

Rule in SQL:
include only tuples that
yield TRUE

```
SELECT  *
FROM    Person
WHERE   age < 25  or age >= 25
```

Unexpected behavior

```
SELECT  *
FROM    Person
WHERE   age < 25  or age >= 25 or age IS NULL
```

Test NULL
explicitly

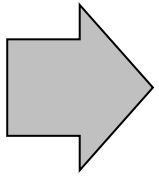
Null Values and Aggregates

373

T

gid	val
1	NULL
1	NULL
2	a
2	a
2	z
2	z
2	NULL
3	A
3	A
3	Z

```
SELECT gid,
       MAX(val) maxv,
       MIN(val) minv,
       COUNT(*) ctr,
       COUNT(val) ctv,
       COUNT(DISTINCT val) ctdv
FROM   T
GROUP BY gid
ORDER BY gid
```



?

Null Values and Aggregates

373

T

gid	val
1	NULL
1	NULL
2	a
2	B
2	z
2	z
2	NULL
3	A
3	A
3	Z

```
SELECT gid,
       MAX(val) maxv,
       MIN(val) minv,
       COUNT(*) ctr,
       COUNT(val) ctv,
       COUNT(DISTINCT val) ctdv
FROM   T
GROUP BY gid
ORDER BY gid
```

NULL is ignored by aggregate functions if you reference the column specifically. Exception: COUNT !

gid	maxv	minv	ctr	ctv	ctdv
1	NULL	NULL	2	0	0
2	z	B	5	4	3
3	Z	A	3	3	2

Inner Joins vs. Outer Joins

Alternaive Join Syntax

Item(name, category)
Purchase2(iName, store, month)

334

An "inner join":

```
SELECT  Item.name, Purchase2.store
FROM    Item, Purchase2
WHERE   Item.name = Purchase2.iName
```

Same as:

```
SELECT  Item.name, Purchase2.store
FROM    Item JOIN Purchase2 ON
        Item.name = Purchase2.iName
```

Item

Name	Category
Gizmo	Gadget
Camera	Photo
OneClick	Photo

Purchase2

iName	Store	Month
Gizmo	Wiz	8
Camera	Ritz	8
Camera	Wiz	9

Result

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Illustration

361

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

An "inner join":

```
SELECT  *
FROM    English, French
WHERE   eid = fid
```

etext	eid	fid	ftext
One	1	1	Un
Three	3	3	Trois
Four	4	4	Quatre
Five	5	5	Cinq
Six	6	6	Siz

Same as:

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

"JOIN"
same as
"INNER JOIN"

Illustration

361

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"
same as
"FULL OUTER JOIN"

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

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

~~
```
SELECT  *
FROM    English JOIN French
ON      eid = fid
```
~~

SQLite does not support "FULL OUTER JOIN"s ☹ (but "LEFT JOIN")