Minor 1 : Feb 6th
@ 8 am

for both
362/632

COL 362 & COL 632

SQL – Composition, Aggregation and Subqueries

24 Jan 2023

Important Announcement

- Instead of PostgreSQL 8.3.23, we will use PostgreSQL 8.4.22

For assignments 1&2,

- There was a serious technical issue with the 8.3.23 that makes it problematic for A1 & A2.

Lack of support for recursive queries & project setup.

- Build process remains the same as before

For project:

① We will use PGSQL 12 for appln.

② No restrictions for systems.

Composition of operators (1/2)

$\sigma_{Ages>42}\left(\Pi_{\underline{Name, Age}}(Actors)\right)$

σ

Actors

Name	Age	Addr
Priyanka Chopra	38	Mumbai
Anthony Hopkins	81	LA
Bill Nighy	69	LA
Abhishek Bachchan	45	Mumbai

Return the names and addresses of actors over 42

$$\Pi_{\underline{Name,Addr}}\left(\sigma_{Age>42}(Actors)\right)$$

Return the names of actors over 42 who live in Mumbai

$$\Pi_{Name,Addr}\left(\sigma_{\underline{Age>42\ \mathrm{AND}\ \mathrm{Addr}='\mathrm{Mumbai}'}}(Actors)\right)$$

Composition of operators (2/2)

Actors

Name	Age	Addr
Priyanka Chopra	38	Mumbai
Anthony Hopkins	81	LA
Bill Nighy	69	LA
Abhishek Bachchan	45	Mumbai

Movies

Name	Year	Title
Priyanka Chopra	2011	Don-II
Anthony Hopkins	2011	MI-IV
Bill Nighy	2009	Valkyrie
Abhishek Bachchan	2010	Raavan

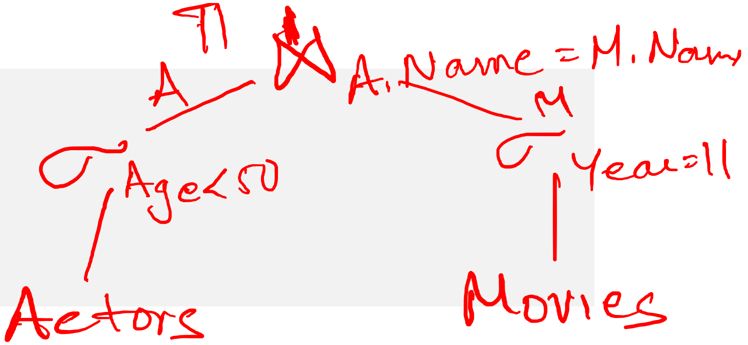
Name, Age, Addr, Year, Title

Return the names of actors below the age of 50 who have acted in a movie in 2011

$$\Pi_{Name}(\sigma_{Age<50\ \text{AND}\ Year=2011}(Actors \bowtie_{A.Name=M.Name} Movies))$$

$$\begin{aligned} Allmovies &= Actors \bowtie_{A.Name=M.Name} Movies \\ Movies1 &= \sigma_{Age<50\ \text{AND}\ Year=2011}(AllMovies) \\ Result &= \Pi_{Name}(Movies1) \end{aligned}$$

Equivalent Queries



Return the names of actors below the age of 50
who have acted in a movie in 2011

$$\Pi_{Name}(\sigma_{Age<50 \text{ AND } Year=2011}(Actors \bowtie_{A.Name=M.Name} Movies))$$

$$\underset{4}{\prod}_{Name}\left(\underbrace{\overset{\checkmark}{\sigma}_{Age<50}(Actors)}_{1} \underset{3}{\bowtie}_{A.Name=M.Name} \underbrace{\sigma_{Year=2011}(Movies)}_{2}\right)$$

The two queries are not identical;
they are, however, equivalent ➔ they give the same result on any database.

Notation (1/2)

σ π

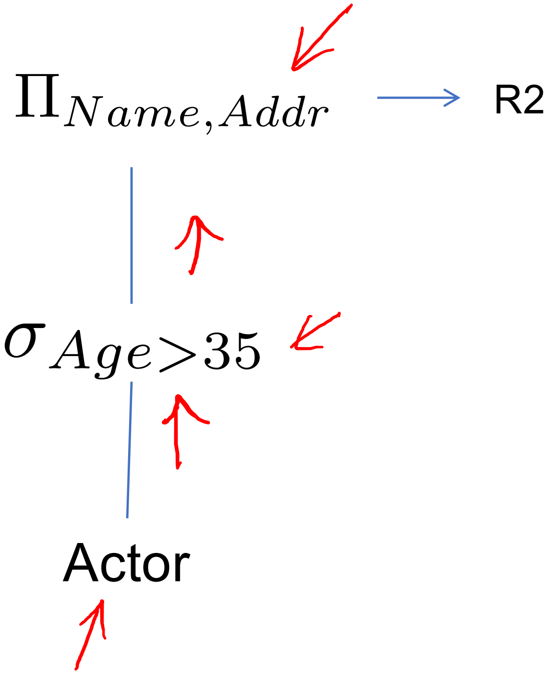
$\bowtie$

Return the names and addresses of actors over 35

$$\Pi_{Name,Addr}(\sigma_{Age>35}(Actor))$$

$$R1 = \sigma_{Age>35}(Actor)$$

$$R2 = \Pi_{Name,Addr}(R1)$$

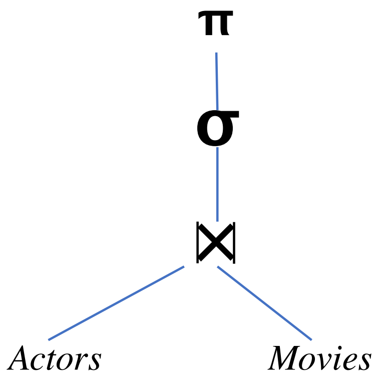


Notation (2/2)

Return the names of actors below the age of 50
who have acted in a movie in 2011

$$\Pi_{Name}(\sigma_{Age<50 \text{ AND } Year=2011}(Actors \bowtie_{A.Name=M.Name} Movies))$$

$$\begin{aligned} Allmovies &= Actors \bowtie_{A.Name=M.Name} Movies \\ Movies1 &= \sigma_{Age<50 \text{ AND } Year=2011}(AllMovies) \\ Result &= \Pi_{Name}(Movies1) \end{aligned}$$



Select count (Address) from Actors;

distinct

= 4

= 2

SQL - Composition of operators (1/2)

Actors

Name	Age	Addr	
Priyanka Chopra	38	Mumbai	
Anthony Hopkins	81	LA	✓
Bill Nighy	69	LA	✓
Abhishek Bachchan	45	Mumbai	

Return the names and addresses of actors over 42

$$\Pi_{Name,Addr}\left(\sigma_{Age>42}(Actors)\right)$$

select *

```
SELECT Name, Addr
FROM Actors
WHERE Age > 42
```

✓

Return the names of actors over 42 who live in Mumbai

$$\Pi_{Name}\left(\sigma_{Age>42 \text{ AND Addr='Mumbai'}}(Actors)\right)$$

```
SELECT Name
FROM Actors
WHERE Age > 42
AND   Addr = 'Mumbai'
```

Relational Calculus Algebra

SQL - Composition of operators (2/2)

Return the names of actors below the age of 50
who have acted in a movie in 2011

$$\Pi_{Name}(\sigma_{Age<50\ \mathrm{AND}\ Year=2011}(Actors \bowtie_{A.Name=M.Name} Movies))$$

X

$$\begin{aligned} Allmovies &= Actors \bowtie_{A.Name=M.Name} Movies \\ Movies1 &= \sigma_{Age<50\ \mathrm{AND}\ Year=2011}(AllMovies) \\ Result &= \Pi_{Name}(Movies1) \end{aligned}$$

Vianu's book excerpt

```
SELECT Actors.Name
FROM Actors, Movies
WHERE Age < 50
AND   Year = 2011
AND   Actors.Name = Movies.Name
```

JOIN

More operators

- Duplicate elimination $\boxed{\delta(R)}$
- Aggregation
  - count, min, max, sum, avg ✓
- Grouping γ ✓
- Sorting τ ✓

Aggregation and grouping (1/2)

- Grouping $\gamma_L(R)$
  - L is a list of grouping attributes and/or aggregate operators

Movies

Movie	City	Boxoffice
MI-IV	LA	2,000,000
Don-II	LA	500,000
MI-IV	NY	3,000,000

Return total boxoffice returns per movie

$$\gamma_{Movie,Sum(Boxoffice)}(Movies)$$

Movie	City	Boxoffice
MI-IV	LA	2,000,000
MI-IV	NY	3,000,000

Don-II	LA	500,000

Movie	Boxoffice
MI-IV	5,000,000
Don-II	500,000

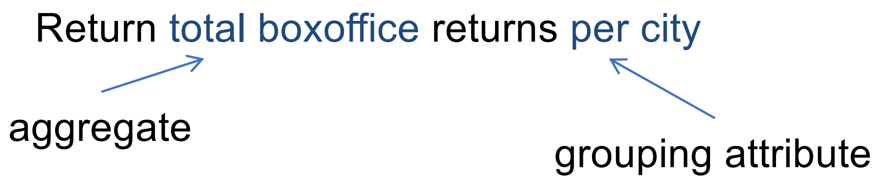
Aggregation and grouping (2/2)

- Grouping $\gamma_L(R)$
  - L is a list of grouping attributes and/or aggregate operators

Movie	City	Boxoffice
MI-IV	LA	2,000,000
Don-II	LA	500,000
MI-IV	NY	3,000,000

Movie	City	Boxoffice
MI-IV	LA	2,000,000
MI-IV	NY	3,000,000

Don-II	LA	500,000



Return total boxoffice returns per city

aggregate

grouping attribute

$$\gamma_{City,Sum(Boxoffice)}(Movies)$$

Movie	City	Boxoffice
MI-IV	LA	2,000,000
Don-II	LA	500,000

MI-IV	NY	3,000,000

City	Boxoffice
LA	2,500,000
NY	3,000,000

Aggregates in SQL

How many movies in the table ?

```
SELECT COUNT(*)
FROM MOVIES;
```

What is the average age of actors living in LA?

```
SELECT AVG(AGE)
FROM Actors
WHERE Actors.Address=‘LA’;
```

Find count of unique addresses in the relation

```
SELECT COUNT(DISTINCT Address)
FROM Actors;
```

Grouping and Aggregation in SQL

- Return total boxoffice returns per city

```
SELECT City,SUM(Boxoffice)
FROM MOVIES
GROUP BY City;
```

- Attributes in **select** clause outside of aggregate functions must appear in **group by** list

```
SELECT City, Movie, SUM(Boxoffice)
FROM MOVIES
GROUP BY City; Movie ;
```

- Return cities with total boxoffice returns more than 500,000
  - predicates in the **having** clause are applied after the formation of groups whereas predicates in the **where** clause are applied before forming groups

```
SELECT City,SUM(Boxoffice)
FROM MOVIES
GROUP BY City
HAVING SUM (Boxoffice) > 500000;
```