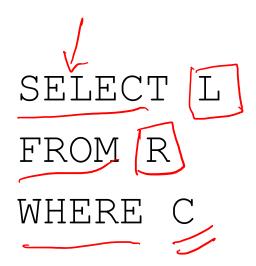
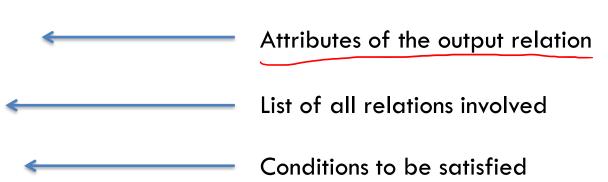
SQL

SQL – basic structure





Selection (1/2)

Actors

| Name | Age | Addr |
|----------------------|-----|--------|
| Priyanka Chopra | 36 | Mumbai |
| Anthony Hopkins | 81 | LA |
| Bill Nighy | 69 | LA |
| Abhishek Bachchan | 42 | Mumbai |

```
Return all actors living in Mumbai \sigma_{Addr='Mumbai'}(Actors)
SELECT Name, Age, Addr
FROM Actors
WHERE Addr = 'Mumbai'
SELECT \star
FROM Actors
```

WHERE Addr = 'Mumbai'

Selection (2/2)

Return all actors whose age is more than 35.

 $\sigma_{Age>35}(Actors)$

SELECT *
FROM Actors
WHERE Age > 35

Return all actors whose age is more than 35 and who live in Mumbai

 $\sigma_{Age>35 \text{ and } Addr='Mumbai}(Actors)$

```
SELECT *

FROM Actors

WHERE Age > 35 AND

Addr = 'Mumbai'
```

Projection

Actors

| Name | Age | Addr |
|-------------------|-----|--------|
| Priyanka Chopra | 36 | Mumbai |
| Anthony Hopkins | 81 | LA |
| Bill Nighy | 69 | LA |
| Abhishek Bachchan | 42 | Mumbai |

Return the name and age of all actors

$$\Pi_{Name,Age}(Actors)$$

SELECT Name, Age. FROM Actors

Return the addresses of the actors

$$\Pi_{Addr}(Actors)$$

SELECT Addr FROM Actors

* 4 Juples in she output

Juples in she outp

SELECT DISTINCT (Addr)
FROM Actors

SELECT DISTINCT (Name, Addr.)
FROM Actors

SELECT Name, DISTING (Addr.)
FROM Actors

Equi-Joins

Actors



| Name | Age | Addr | Name | Year | Title |
|------|-----|--------|------|------|----------|
| PC | 36 | Mumbai | PC | 2011 | Don-II |
| AH | 81 | LA | AH | 2011 | Thor: R |
| BN | 69 | LA | BN | 2009 | Valkyrie |
| AB | 42 | Mumbai | AB | 2010 | Raavan |

Return all information about actors and their movies

 $Actors \bowtie_{A.Name = \overline{M.Name}} Movies$

SELECT *

FROM Actors, Movies

WHERE Actors.Name = Movies.Name

Left outer joins

Actors

Movies

| Name | Age | Addr | Name | Year | Title |
|------|-----|--------|------|------|---------|
| PC | 36 | Mumbai | PC | 2011 | Don-II |
| AH | 81 | LA | AH | 2011 | Thor: R |
| BN | 69 | LA | AB | 2010 | Raavan |
| AB | 42 | Mumbai | | | |

Return all information about actors and their movies

 $Actors \bowtie_{A.Name=M.Name} Movies$

```
SELECT *
FROM Actors LEFT OUTER JOIN Movies
ON (Actors.Name = Movies.Name)
```

What happens when you compare something with a null value? Or when you compare a null with a null?

Self Join

FROM Actors Al, Alters Ar

Return all grandparents and their grand children

Actors

Actors_1

| Name | Age | Addr | Parent | Name | Age | Addr | Parent |
|------|-----|--------|-----------|------|-----|--------|-----------|
| PC | 36 | Mumbai | Madhu | PC | 36 | Mumbai | Madhu |
| AH | 81 | LA | Muriel | AH | 81 | LA | Muriel |
| BN | 69 | LA | Catherine | BN | 69 | LA | Catherine |
| AB | 42 | Mumbai | Jaya | AB | 42 | Mumbai | Jaya |
| Jaya | 63 | Mumbai | Indira | Jaya | 63 | Mumbai | Indira |

FROM Actors AS Actors1 Actors AS Actors2
WHERE Actors1.Parent = Actors2.Name

Composition of operators (1/2)

Actors

| Name | Age | Addr |
|-------------------|-----|--------|
| Priyanka Chopra | 36 | Mumbai |
| Anthony Hopkins | 81 | LA |
| Bill Nighy | 69 | LA |
| Abhishek Bachchan | 42 | Mumbai |

Return the names and addresses of actors over 35

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

Return the names of actors over 35 who live in Mumbai

$$\Pi_{Name}(\sigma_{Age>35 \text{ and } Addr='Mumbai'}(Actor))$$

SELECT Name
FROM Actors
WHERE Age > 35
AND Addr = 'Mumbai'



Composition of operators (2/2)

```
FROM Actors. Name

FROM Actors, Movies

WHERE Age < 50

AND Year = 2011

AND Actors. Name = Movies. Name
```

More operators

- Duplicate elimination
- Extended projection
- Aggregation
 - count, min, max, sum, avg
- Grouping
- Sorting

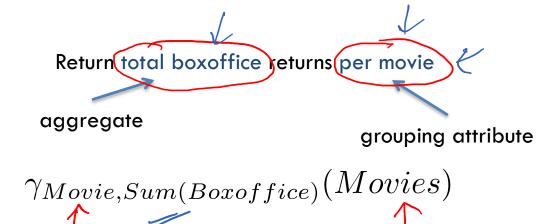
Aggregation and grouping (1/2)

• Grouping $\gamma_L(R)$

-L is a list of grouping attributes and/or aggregate

operators

| Movie | City | Boxoffice |
|---------|------|-----------|
| Thor: R | LA | 2,000,000 |
| Don-II | LA | 500,000 |
| Thor: R | NY | 3,000,000 |



| Movie | City | Boxoffice |
|---------|------|-----------|
| Thor: R | LA | 2,000,000 |
| Thor: R | NY | 3,000,000 |

| Don-II | LA | 500,000 |
|--------|----|---------|
|--------|----|---------|

| Movie | Boxoffice |
|---------|-----------|
| Thor: R | 5,000,000 |
| Don-II | 500,000 |

Aggregation and grouping (2/2)

| Movie | City | Boxoffice |
|---------|------|-----------|
| Thor: R | LA | 2,000,000 |
| Don-II | LA | 500,000 |
| Thor: R | NY | 3,000,000 |

Return movies for which total boxoffice returns were greater than 1,000,000

```
SELECT Movie, SUM (Boxoffice)
FROM Movies
GROUP BY Movie
HAVING SUM (Boxoffice) > 1000000
```

Return total boxoffice returns per movie

```
YMovie, Sum(Boxoffice) (Movies)

SELECT Movie, SUM (Boxoffice)

FROM Movies

GROUP BY Movie
```

Sorting

Sorting tuples by column

SELECT * FROM Movies ORDER BY City

SELECT Movie, Boxoffice FROM Movies ORDER BY Boxoffice DESC

FROM Movies GROUP BY Movie

HAVING/SUM (Boxoffice) > 1000000 ORDER BY BO DESC

Morre

SELECT Movie, SUM (Boxoffice)

FROM Movies

GROUP_BY Movie

HAVING SUM (Boxoffice) > 1000000

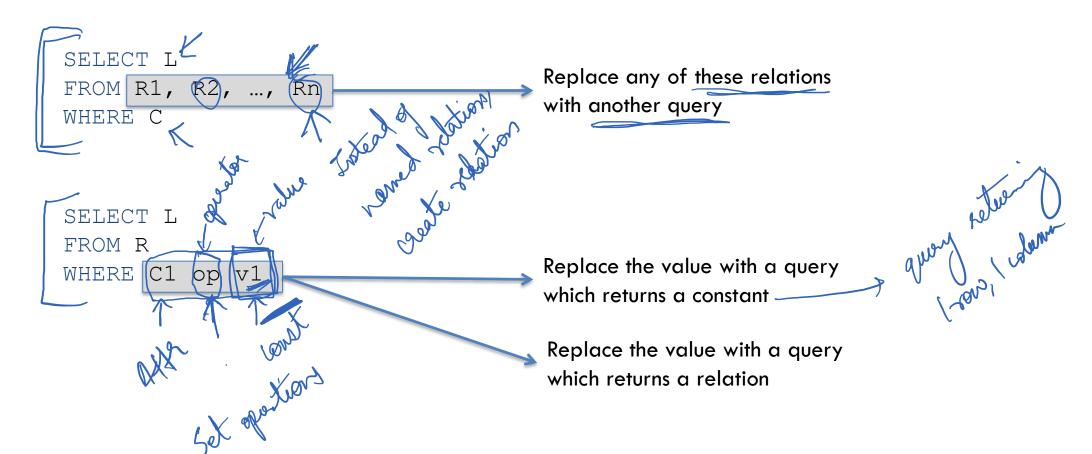
ORDER BY Movie

SELECT Movie, SUM (Boxoffice) AS BO

Subqueries (1/2)

FROM Morries AS M, Actors AS A

• Temporary relations, constants



Solet * John Works

Subqueries (2/2)

SELECT *
FROM (SELECT *
FROM MOVIES) AS DUMB

Actors

| | Name | Age | Addr |
|---|------|-----|--------|
| | PC | 36 | Mumbai |
|) | АН | 81 | LA |
| | BN | 69 | LA |
| | AB | 42 | Mumbai |

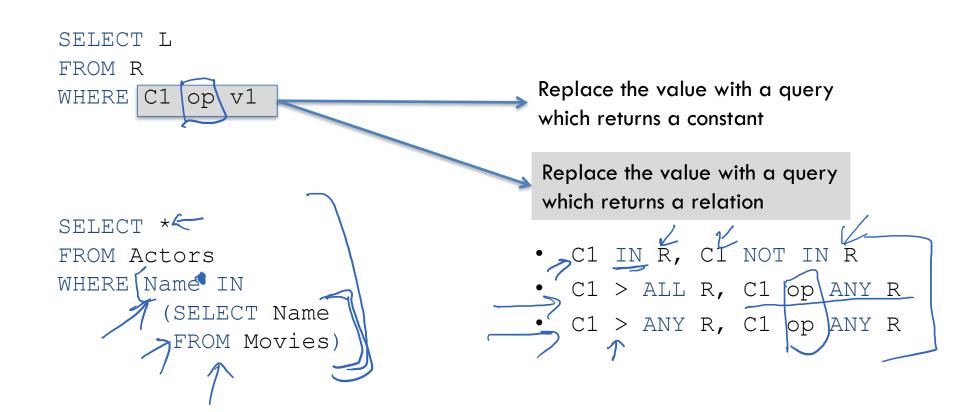
SELECT *

FROM Actors

WHERE Age < (SELECT AVG (Age))

FROM Actors)

Conditions involving relations (1/2)



Conditions involving relations (2/2)

```
SELECT *
FROM (SELECT *
FROM MOVIES) AS DUMB
```

Actors

| Name | Age | Addr |
|------|-----|--------|
| PC | 36 | Mumbai |
| AH | 81 | LA |
| BN | 69 | LA |
| AB | 42 | Mumbai |

```
SELECT *

FROM Actors

WHERE Age <
```

SELECT *

FROM Actors

WHERE Age < ALL

WHERE Age < (SELECT AVG(Age)
FROM Actors)

```
AB 42 Mumbai SELECT *
FROM Actors
WHERE Age < ANY
(SELECT Age
FROM Actors)

Where Age is a series of the first of the firs
```

- null platiers
amply polatiers

Reading and Practical HW

- Correlated subqueries
- Union, intersection, difference operations