SQL

SQL – basic structure

```
SELECT L
FROM R
WHERE C
```

- SELECT L ← Attributes of the output relation
- FROM R ← List of all relations involved
- WHERE C ← Conditions to be satisfied

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 *
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
```

4 tuples in the output

Return the addresses of the actors

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

```
SELECT Addr
FROM Actors
```

```
SELECT DISTINCT (Addr)
FROM Actors
```

duplicate elimination op.

```
SELECT DISTINCT(Name, Addr)
FROM Actors
```

```
[SELECT Name, DISTINCT(Addr)
FROM Actors
```

Equi-Joins

Actors

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

Movies

Name	**Year**	**Title**
PC	2011	Don-II
AH	2011	Thor: R
BN	2009	Valkyrie
AB	2010	Raavan

Return all information about actors and their movies

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

```
SELECT *
FROM Actors, Movies
WHERE Actors.Name = Movies.Name
```

Left outer joins

Actors

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

Movies

Name	**Year**	**Title**
PC	2011	Don-II
AH	2011	Thor: R
AB	2010	Raavan

Return all information about actors and their movies

$$\underline{Actors} ⟕_{A.Name=M.Name} Movies$$

```
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?

FROM Actors A1,
Actors A2

Self Join

Return all grandparents and their grand children

Actors

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

Actors_1

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

Alias

```
SELECT *
FROM Actors AS Actors1, Actors AS Actors2
WHERE Actors1.Parent = Actors2.Name
```

A

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

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

```
SELECT Name, Addr
FROM Actors
WHERE Age > 35
```

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)

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

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

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

$S \rightarrow Name$

$F \rightarrow A, M$

$W \rightarrow 3$

```
SELECT 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

Return total boxoffice returns per movie

aggregate

grouping attribute

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

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 total boxoffice returns per movie

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

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

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
```

Sorting

τ , γ , σ

$\tau \left[\sigma_{S(BO) > 1M} \left(\gamma_{Movie, Sum(BO)} (M) \right) \right]$

$\rho \left(\sigma \left(\gamma_{M, Sum(BO)} (M) \right) \right)$

$\rho (\gamma)$

- Sorting tuples by column

```
SELECT *
FROM Movies
ORDER BY City
```

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

```
SELECT Movie, Boxoffice
FROM Movies
ORDER BY Boxoffice DESC
```

descending

```
SELECT Movie, SUM (Boxoffice) AS BO
FROM Movies
GROUP BY Movie
HAVING SUM (Boxoffice) > 1000000
ORDER BY BO DESC
```

Movie	BO

Subqueries (1/2)

FROM Movies AS M,
Actors AS A

- Temporary relations, constants

```
SELECT L
FROM R1, R2, …, Rn
WHERE C
```

$\rightarrow$ Replace any of these relations with another query

Instead of named relations, create relations

```
SELECT L
FROM R
WHERE C1 op v1
```

operator

value

Attr

const

Set operations

$\rightarrow$ Replace the value with a query which returns a constant $\rightarrow$ query returning 1 row, 1 column

$\rightarrow$ Replace the value with a query which returns a relation

Select * from Movies

Subqueries (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 AVG(Age)
        FROM Actors)
```

1 row 1
1 col

Conditions involving relations (1/2)

```
SELECT L
FROM R
WHERE C1 op v1
```

Replace the value with a query which returns a constant

Replace the value with a query which returns a relation

```
SELECT *
FROM Actors
WHERE Name IN
      (SELECT Name
       FROM Movies)
```

- C1 IN R, C1 NOT IN R
- C1 > ALL R, C1 op ANY R
- C1 > ANY R, C1 op ANY R

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 AVG(Age)
       FROM Actors)
```

✓

```
SELECT *
FROM Actors
WHERE Age < ALL
      (SELECT Age
       FROM Actors)
```

→ NULL relation
empty relation

```
SELECT *
FROM Actors
WHERE Age < ANY
      (SELECT Age
       FROM Actors)
```

← all tuples except one with largest Age value

Reading and Practical HW

- Correlated subqueries
- Union, intersection, difference operations