

1



# LINGUAGGIO SQL

## Parte 2

# FUNZIONI DI GRUPPO

Partiamo con un esempio

Determinare il numero di film presenti in catalogo, il numero dei loro registi e la loro valutazione minima, media e massima

- Non vogliamo estrarre informazioni da singole tuple in Film
- Vogliamo invece
  - Contare quante tuple ci sono nella relazione
  - Calcolare funzioni matematiche sull'**insieme** di valori contenuti nella colonna valutaz di **tutte** le tuple della relazione
- In seguito i dettagli, anticipiamo la soluzione

```
SELECT COUNT(*), MIN(valutaz), AVG(valutaz), MAX(valutaz)
FROM Film;
```

COUNT(*)	COUNT(DISTINCT regista)	MIN(valutaz)	AVG(valutaz)	MAX(valutaz)
14	4	3.00	3.55	4.00

# FUNZIONI DI GRUPPO/AGGREGATE

- Estraggono **informazioni aggregate** da **insiemi di dati**
- Si usano **solo** per costruire le espressioni nella clausola di proiezione di un'interrogazione
- La forma generale di applicazione è  
    <nome-fun>([DISTINCT] <espressione>)
  - in <espressione> possono comparire nomi di attributi, costanti, funzioni, **ma non funzioni aggregate**
  - l'applicazione può essere usata in espressioni più complesse
- L'insieme da usare come argomento si calcola valutando l'espressione su tutte le tuple che soddisfano la clausola di qualificazione dell'interrogazione
  - se compare DISTINCT eliminando i duplicati
  - escludendo i valori NULL

# FUNZIONI DI GRUPPO

## COUNT

Cardinalità dell'insieme passato come argomento

- sull'insieme vuoto restituisce 0
- è l'unica che si applica anche a insiemi di **tuple**
- può avere anche la forma

COUNT([DISTINCT] \*)

- la funzione restituisce il numero di tuple [(distinte)] che soddisfano la clausola di selezione

# FUNZIONI DI GRUPPO NUMERICHE

- Si applicano solo a insiemi di **valori** semplici (non tuple)
- Sull'insieme vuoto restituiscono NULL
- **MAX** (massimo) e **MIN** (minimo) si applicano a insiemi di valori **ordinati**
  - usare o meno DISTINCT non cambia il risultato
- **SUM** (somma) e **AVG** (average = media ) si applicano a insiemi di valori **numerici**
- Ci sono altre funzioni che non presentiamo, fra cui ad esempio **STDEV** (deviazione standard) e **VAR** (varianza)

# FUNZIONI DI GRUPPO - LIMITAZIONE

Esempio **sbagliato**

Determinare il titolo e il regista del film drammatico di valutazione minima

```
SELECT titolo, regista
```

```
FROM Film
```

```
WHERE genere = 'drammatico' AND valutaz = MIN(valutaz);
```

- Le funzioni di gruppo possono comparire solo nella clausola di proiezione
- Quindi non possono essere usati per filtrare le tuple nella clausola di selezione
- Avremo bisogno di una sotto-interrogazione (vedi seguito)

## FUNZIONI DI GRUPPO – ESEMPIO 1

Determinare il numero di film presenti in catalogo, il numero complessivo di registi che li hanno girati e la valutazione minima, media e massima di tali film

```
SELECT COUNT(*), COUNT(DISTINCT regista),  
       MIN(valutaz), AVG(valutaz), MAX(valutaz)  
FROM Film;
```

Risultato

COUNT(*)	COUNT(DISTINCT regista)	MIN(valutaz)	AVG(valutaz)	MAX(valutaz)
14	4	3.00	3.55	4.00

## FUNZIONI DI GRUPPO – ESEMPIO 2

Determinare il numero di film di genere drammatico presenti in catalogo, il numero complessivo di registi che li hanno girati e la valutazione minima, media e massima di tali film

```
SELECT COUNT(*), COUNT(DISTINCT
    regista),MIN(valutaz),
    AVG(valutaz), MAX(valutaz)
FROM Film
WHERE genere = 'drammatico';
```

Risultato

COUNT(*)	COUNT(DISTINCT regista)	MIN(valutaz)	AVG(valutaz)	MAX(valutaz)
3	3	3.20	3.57	4.00



## FUNZIONI DI GRUPPO – ESEMPIO 3

Determinare il numero di clienti che hanno effettuato almeno un noleggio, il numero dei registi dei film noleggiati e la durata massima (in giorni) dei noleggi

```
SELECT  COUNT(DISTINCT  codCli),  COUNT(DISTINCT  
        regista),  
        MAX((dataRest-dataNol) DAY) AS durataM  
FROM Noleggio NATURAL JOIN Video
```

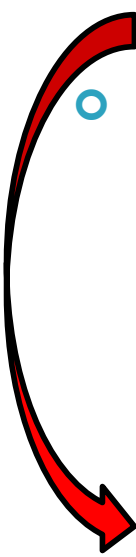
# OPERATORE DI RAGGRUPPAMENTO

## GROUP BY

- Permette di partizionare una relazione in base al valore di una o più colonne

- Esempio

determinare **per ogni regista** quanti suoi film con valutazione superiore a 3 sono presenti in catalogo



```
SELECT regista,  
COUNT(*) AS numeroFilmBuoni  
FROM Film  
WHERE valutaz >= 3  
GROUP BY regista;
```

- il risultato di un'interrogazione che contiene una clausola GROUP BY contiene tante tuple
  - quanti sono i gruppi di tuple risultanti dal partizionamento
  - quanti sono i valori distinti delle proiezioni sulle colonne usate per raggruppare

# OPERATORE DI RAGGRUPPAMENTO

## GROUP BY

- La clausola di proiezione di un'interrogazione contenente la clausola GROUP BY può solo includere:
  - una o più tra le colonne che compaiono nella clausola GROUP BY
  - funzioni di gruppo
- Il risultato contiene **una sola tupla per ogni gruppo**
  - le colonne su cui **non** si raggruppa delle tuple in uno **stesso gruppo** possono assumere **diversi valori**..quale scegliere?

# RAGGRUPPAMENTO ESEMPIO

non avrebbe senso  
perché in ogni gruppo ci  
sono anni diversi

Risultato di  
`SELECT regista, anno`  
`COUNT(*) AS`  
`numeroFilmBuoni`  
`FROM Film`  
`WHERE valutaz >= 3`  
**`GROUP BY regista;`**

titolo	regista	anno	genere	valutaz
underground	emir kusturica	1995	drammatico	3.20
edward mani di forbice	tim burton	1990	fantastico	3.60
nightmare before christmas	tim burton	1993	animazione	4.00
ed wood	tim burton	1994	drammatico	4.00
mars attacks	tim burton	1996	fantascienza	3.00
il mistero di sleepy hollow	tim burton	1999	horror	3.50
big fish	tim burton	2003	fantastico	3.10
la sposa cadavere	tim burton	2005	animazione	3.50
la fabbrica di cioccolato	tim burton	2005	fantastico	4.00
io non ho paura	gabriele salvatores	2003	drammatico	3.50
nirvana	gabriele salvatores	1997	fantascienza	3.00
mediterraneo	gabriele salvatores	1991	commedia	3.80
pulp fiction	quentin tarantino	1994	thriller	3.50
le iene	quentin tarantino	1992	thriller	4.00

4 gruppi: uno per ogni  
valore di regista

regista	numeroFilmBuoni
emir kusturica	1
tim burton	8
gabriele salvatores	3
quentin tarantino	2

ogni tupla qui rappresenta un  
gruppo di tuple della relazione su  
cui l'interrogazione è eseguita

# RAGGRUPPAMENTO

## ESEMPIO

- Partizionare i noleggi in base al cliente che li ha effettuati ed al regista del film noleggiato
- per ogni gruppo, determinare il numero di noleggi e la durata massima (in giorni) di tali noleggi

```
SELECT codCli, regista, COUNT(*) AS NumN,  
       MAX((dataRest-dataNol) DAY) AS durataM  
FROM Noleggio NATURAL JOIN Video  
GROUP BY codCli, regi
```

Risultato

codCli	regista	numN	durataM
6635	emir kusturica	1	1
6635	tim burton	8	4
6635	gabriele salvatores	1	?
6635	quentin tarantino	2	5
6642	emir kusturica	1	1
6642	tim burton	5	1
6642	gabriele salvatores	1	1
6642	quentin tarantino	1	2
6610	emir kusturica	1	2
6610	tim burton	4	1
6610	gabriele salvatores	2	1

# RAGGRUPPAMENTO

## ESEMPIO

- Se volessi anche il nome e cognome del cliente

```
SELECT codCli, nome, cognome, regista, COUNT(*) AS NumN,  
       MAX((dataRest-dataNol) DAY) AS durataM  
FROM Noleggio NATURAL JOIN Video NATURAL JOIN Cliente  
GROUP BY codCli, nome, cognome, regista;
```

Anche se nome, cognome non influenzano i gruppi (perché sono determinati da codCli) per poterli restituire bisogna includerli nel group by

# CLAUSOLA HAVING

- Specifica un filtro sui gruppi ottenuti dal partizionamento
- È una combinazione Booleana di predicati (o un atomo)
  - ciascun atomo **deve** coinvolgere funzioni di gruppo
  - non può contenere condizioni sui valori delle singole colonne
- Non vi è alcuna relazione tra le funzioni di gruppo eventualmente usate nella clausola SELECT e quelle usate nella clausola HAVING

# CLAUSOLA HAVING – ESEMPIO

Per ogni regista **che ha girato almeno due film prima del 2000**, determinare quanti sono tali film, di quanti generi diversi e la valutazione minima, media e massima di tali film

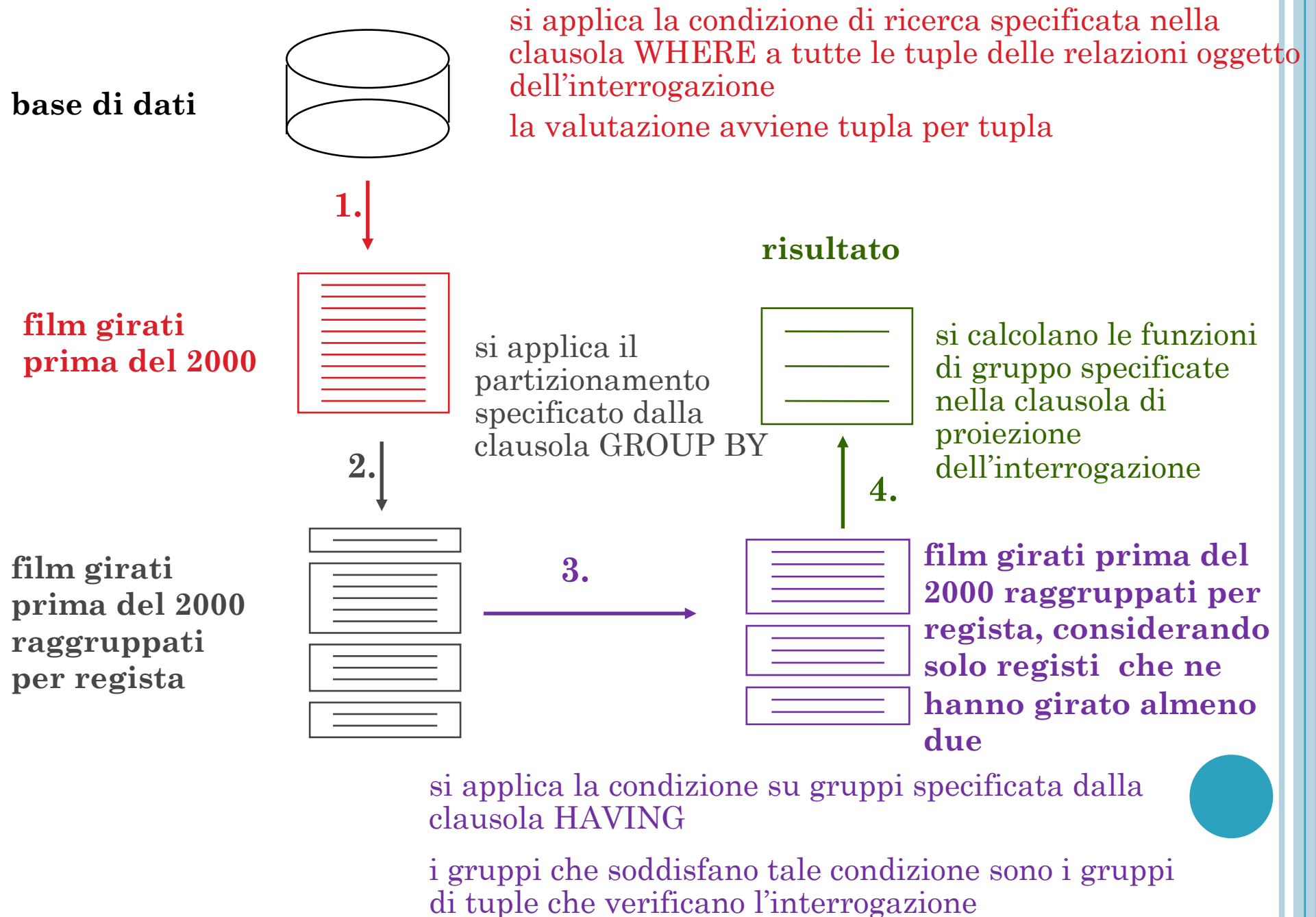
```
SELECT regista, COUNT(*) AS numF,  
        COUNT(DISTINCT genere) AS numG,  
        MIN(valutaz) AS minV, AVG(valutaz) AS avgV, MAX(valutaz) AS maxV  
FROM Film  
WHERE anno < 2000  
GROUP BY regista  
HAVING COUNT(*) >= 2;
```

Risultato

regista	numF	numG	minV	avgV	maxV
tim burton	5	5	3.00	3.62	4.00
gabriele salvatores	2	2	3.00	3.40	3.80
quentin tarantino	2	1	3.50	3.75	4.00



# RAGGRUPPAMENTO - MODELLO DI *ESECUZIONE*



# VALORI NULLI

## RIASSUNTO DELLE PUNTATE PRECEDENTI

- SQL usa una logica a tre valori TRUE (T), FALSE (F), UNKNOWN (?)
- UNKNOWN (?) = valore di verità di una condizione di ricerca applicata ad una data tupla non è determinabile
  - un atomo valutato su un attributo a valore nullo restituisce ?
  - il valore di verità di una formula dipende dalla propagazione secondo le funzioni booleane
- un'interrogazione restituisce solo le tuple su cui la formula di selezione vale TRUE
  - se il valore di verità è ? non viene restituita

# VALORI NULLI - ESEMPIO

**R**

A	B	C		A = a	B = b	A = a $\vee$ B = b
a	?	c <sub>1</sub>	t <sub>1</sub>	T	?	T
a <sub>1</sub>	b	c <sub>2</sub>	t <sub>2</sub>	F	T	T
a <sub>2</sub>	?	?	t <sub>3</sub>	F	?	?

SELECT \* FROM R WHERE A=a OR B=b;

solo le tuple t1 e t2 verificano l'interrogazione

# VALORI NULLI

## ESEMPIO RELATIVO AI NOLEGGI IN CORSO

- SELECT colloc FROM Noleggio

WHERE codCli = 6635 AND dataRest > DATE '15-Mar-2012';

non restituisce i noleggi in corso

- SELECT colloc FROM Noleggio

WHERE codCli = 6635 AND (dataNol > DATE '20-Mar-2012' OR dataRest > DATE '20-Mar-2012');

restituisce anche noleggi in corso, purché siano iniziati dopo il 20 Marzo

- SELECT colloc FROM Noleggio

WHERE codCli = 6635 AND NOT dataRest < DATE '15-Mar-2024';

non restituisce i noleggi in corso

# VALORI NULLI

Le interrogazioni:

```
SELECT colloc FROM Noleggio
```

```
WHERE dataRest = CURRENT_DATE OR NOT dataRest =  
CURRENT_DATE;
```

```
SELECT colloc FROM Noleggio WHERE dataRest = dataRest;
```

non restituiscono tutti i noleggi, ma solo i noleggi terminati, cioè

```
SELECT colloc FROM Noleggio
```

```
WHERE dataRest IS NOT NULL;
```

# VALORI NULLI

## Ribadiamo

- nelle espressioni (ad es. aritmetiche) se un argomento è NULL allora il valore dell'intera espressione è NULL
- esempio: le tuple relative a noleggi correnti hanno durata (dataRest – dataNol) DAY indeterminata
- nel calcolo delle funzioni di gruppo vengono escluse le tuple che hanno valore nullo per la colonna su cui la funzione è calcolata
- questo ha come conseguenza che, ad esempio,  
 $\text{SUM}(e1 + e2)$  può dare risultato diverso da  
 $\text{SUM}(e1) + \text{SUM}(e2)$
- una funzione di gruppo può comunque restituire NULL se applicata a un insieme di valori vuoto o contenente il solo valore NULL

## VALORI NULLI - ESEMPIO

- SELECT SUM(A+B) FROM R;

- SUM(3, NULL, NULL) = 3

- SELECT SUM(A) + SUM(B);

- SUM(1, NULL, 4) + SUM(2, 3, NULL) = 5 + 5 = 10;

R		
A	B	
1	2	t <sub>1</sub>
?	3	t <sub>2</sub>
4	?	t <sub>3</sub>

# VALORI NULLI

- se  $e1$  ed  $e2$  sono NULL,  $e1=e2$  non è vero (è ?)
- ⇒ la presenza di due tuple distinte con valori nulli non viene impedita dalla specifica di vincolo UNIQUE (definito basandosi su uguaglianza)
- ⇒ due espressioni con valore nullo non sono uguali ma non sono distinte, cioè vengono considerate duplicati
  - SELECT DISTINCT: si ha al più un NULL nel risultato
  - GROUP BY: si ha al più un gruppo per il valore NULL

esiste un predicato IS DISTINCT FROM che coincide con  $\neq$  tranne che per il valore nullo, cioè se  $e1$  ed  $e2$  sono NULL:

- $e1 \neq e2$  restituisce UNKNOWN, ma
- $e1$  IS DISTINCT FROM  $e2$  restituisce FALSE



# SOTTO-INTERROGAZIONI

- Si usa una query per rappresentare un valore in un atomo
  - clausola di selezione di SELECT ma anche di INSERT, DELETE, UPDATE come vedremo
- Esempio: determinare il titolo di tutti i film che hanno la stessa valutazione di 'le iene' di 'quentin tarantino'

**SELECT titolo FROM Film**

**WHERE valutaz = (**

**SELECT valutaz FROM Film**

**WHERE titolo = 'le iene' and**

**regista = 'quentin tarantino');**

**interrogazione  
esterna**

**sotto-  
interrogazione**

- la sotto-interrogazione restituisce 4.00, usato nel predicato dell'interrogazione esterna
  - la query  
SELECT titolo FROM Film  
WHERE valutaz = 4  
produce lo stesso risultato ma non è equivalente
    - richiede di formulare ed eseguire due query (meno efficiente)
    - non si propagano le modifiche
    - particolarmente pericoloso in caso di accessi concorrenti (= sempre!!!!)
- l'interrogazione restituisce 'nightmare before christmas', 'ed wood', 'la fabbrica di cioccolato' e 'le iene'

## SOTTO-INTERROGAZIONI

- Esempio più complesso: determinare i film la cui valutazione è superiore alla media

```
SELECT * FROM Film
```

```
WHERE valutaz > ( SELECT AVG(valutaz)  
                  FROM Film);
```

- la sotto-interrogazione restituisce il valore 3.55, quindi solo i film con valutazione superiore a tale valore vengono selezionati

titolo	regista	anno	genere	valutaz
edward mani di forbice	tim burton	1990	fantastico	3.60
nightmare before christmas	tim burton	1993	animazione	4.00
ed wood	tim burton	1994	drammatico	4.00
la fabbrica di cioccolato	tim burton	2005	fantastico	4.00
mediterraneo	gabriele salvatores	1991	commedia	3.80
le iene	quentin tarantino	1992	thriller	4.00

## SOTTO-INTERROGAZIONI

- Permettono di esprimere le interrogazioni di massimo e di minimo discusse in precedenza
- Esempio: determinare il titolo ed il regista del film drammatico di valutazione minima

SELECT titolo, regista FROM Film

WHERE genere = 'drammatico' AND

valutaz = (SELECT MIN(valutaz)

FROM Film

WHERE genere = 'drammatico');

## SOTTO-INTERROGAZIONI **SCALARI**

- Sono sotto-interrogazioni che restituiscono **un solo** valore
  - sintatticamente si usano come un valore di quel tipo
- Tutti gli esempi visti sono sotto-interrogazioni **scalari**
  - erano tutti numeri e si usavano come tali
- Se una sotto-interrogazione scalare restituisce più di una tupla si genera un errore a run-time
  - che senso avrebbe  $valutaz > \langle\langle tabella \rangle\rangle??$
- Se nessuna tupla verifica la sotto-interrogazione, viene restituito il valore NULL

## SOTTO-INTERROGAZIONI **TABLE SUBQUERY**

- Sotto interrogazioni che restituiscono una tabella (non un singolo valore)
- Per usarle in un atomo serve un quantificatore
- In logica se  $X = \{3, 5, 9\}$ 
  - $4 > X$  non ha senso (ed è sintatticamente scorretta)
  - $\exists x \in X. 4 > x$  è corretta e vale TRUE
  - $\forall x \in X. 4 > x$  è corretta e vale FALSE
- In SQL si usano i quantificatori **ANY** ( $\exists$ ) ed **ALL** ( $\forall$ ) inseriti tra l'operatore di confronto e la sotto-interrogazione
  - $\text{exp op\_confronto ANY Q}$
  - $\text{exp op\_confronto ALL Q}$
- $\text{exp op\_confronto ANY Q}$  (sinonimo  $\text{exp op\_confronto SOME Q}$ )
  - restituisce TRUE se la valutazione di **exp op\_confronto v** restituisce TRUE su **almeno una** tupla **v** ottenuta dalla valutazione di Q
  - ⇒ restituisce FALSE se la sotto-interrogazione non restituisce tuple
- $\text{exp op\_confronto ALL Q}$ 
  - restituisce TRUE se la valutazione di **exp op\_confronto v** restituisce TRUE su **tutte le** tuple **v** ottenute dalla valutazione di Q
  - ⇒ restituisce TRUE se la sotto-interrogazione non restituisce tuple

## TABLE SUBQUERY – ESEMPIO 1

- Determinare titolo, regista ed anno dei film più vecchi di **almeno un** film di Quentin Tarantino (si può fare anche con MAX)

SELECT titolo, regista, anno

FROM Film

WHERE anno < **ANY** ( SELECT anno FROM Film

WHERE regista = 'quentin tarantino');

- risultato (l'ultimo film di Quentin Tarantino è del 1994)

titolo	regista	anno
edward mani di forbice	tim burton	1990
nightmare before christmas	tim burton	1993
mediterraneo	gabriele salvatores	1991
le iene	quentin tarantino	1992

## TABLE SUBQUERY – ESEMPIO 2

- Determinare titolo, regista ed anno dei film più vecchi di **tutti i** film di Quentin Tarantino (si può fare anche con MIN)

```
SELECT titolo, regista, anno
```

```
FROM Film
```

```
WHERE anno < ALL ( SELECT anno FROM Film
```

```
WHERE regista = 'quentin tarantino');
```

- risultato (il primo film di Quentin Tarantino è del 1992)

titolo	regista	anno
edward mani di forbice	tim burton	1990
mediterraneo	gabriele salvatores	1991

## SOTTO-INTERROGAZIONI – ALL $\Leftrightarrow$ MAX/MIN

- Le interrogazioni

SELECT ... FROM ... WHERE .... exp >= ALL SELECT exp' ....

e

SELECT ... FROM ... WHERE .... exp >= SELECT MAX(exp') ....

Sono equivalenti?

- Esempio: per determinare il titolo ed il regista del film drammatico di valutazione minima

SELECT titolo, regista FROM Film

WHERE genere = 'drammatico' AND

valutaz >= ALL ( SELECT valutaz FROM Film WHERE genere = 'drammatico');

equivale a

SELECT titolo, regista FROM Film

WHERE genere = 'drammatico' AND

valutaz = SELECT MAX(valutaz) FROM Film WHERE genere = 'drammatico';

????

- In assenza di valori nulli sì
  - altrimenti la prima si valuta in FALSE, la seconda potrebbe ritornare TRUE



## SOTTO-INTERROGAZIONI - APPARTENENZA

- Usando ANY ed ALL si può definire la (non) appartenenza insiemistica
  - exp **IN** Q definito come exp = **ANY** Q
    - in logica  $x \in X \Leftrightarrow \exists x' \in X. x' = x$
  - exp **NOT IN** Q definito come exp **<>** **ALL** Q
    - in logica  $x \notin X \Leftrightarrow \forall x' \in X. x' \neq x$
- Esempio: elencare il titolo dei film di Quentin Tarantino usciti nello stesso anno di un film di Tim Burton  
SELECT titolo  
FROM Film  
WHERE regista = 'quentin tarantino'  
AND anno IN (SELECT anno FROM Film WHERE regista = 'tim burton');
- Risultato: 'pulp fiction' (uscito nel 1994)

# SOTTO-INTERROGAZIONI – APPARTENENZA

## ESEMPIO

- Restituire il titolo dei film di Quentin Tarantino usciti in un anno in cui non sono usciti film di Tim Burton

```
SELECT titolo FROM Film
```

```
WHERE regista = 'quentin tarantino'
```

```
AND anno NOT IN ( SELECT anno FROM Film
```

```
WHERE regista = 'tim burton');
```

- Risultato: le iene (1992)

# SOTTO-INTERROGAZIONI – APPARTENENZA TUPLE

- È possibile selezionare più di una colonna tramite una sotto-interrogazione
- È necessario racchiudere fra parentesi la lista delle colonne a sinistra dell'operatore di confronto
- Esempio: elencare il titolo dei film presenti nella videoteca con lo stesso anno di uscita e genere di un film di Tim Burton

`SELECT titolo FROM Film`

`WHERE regista <> 'tim burton' AND`

`(anno,genere) IN ( SELECT anno, genere FROM Film`

`WHERE regista = 'tim burton');`

- l'interrogazione non restituisce alcuna tupla

## SOTTO-INTERROGAZIONI

# SOTTOLINEANDO L'OVVIO...

- Una sotto-interrogazione può avere al suo interno un'altra sotto-interrogazione, predicati di join, clausole group by e tutti i predicati visti
- la clausola di qualificazione di una interrogazione può contenere una qualsiasi combinazione di condizioni normali e condizioni con sotto-interrogazioni