Il predicato di join è `Video.colloc = Noleggio.colloc`. Il risultato dell'interrogazione è illustrato nella Figura 3.6. □

Dato che il risultato di un'operazione di join è una relazione, è possibile richiedere che tale relazione sia ordinata, anche in base a valori di colonne di relazioni diverse, o che eventuali duplicati siano eliminati.

Outer join. In R `JOIN` S non si ha traccia delle tuple di R che non corrispondono ad alcuna tupla di S. Questo non è sempre quello che si desidera. Per questo motivo SQL prevede un operatore di `OUTER JOIN` che aggiunge al risultato le tuple di R e/o S che non hanno partecipato al join, completandole con `NULL`. L'operatore di join originario, per contrasto, viene anche detto `INNER JOIN`. La variante `OUTER` può essere utilizzata sia per il join naturale sia per il theta-join.

Esistono diverse varianti dell'outer join, che vengono specificate premettendo il corrispondente qualificatore all'operatore `OUTER JOIN`. Consideriamo R `OUTER JOIN` S:

- `FULL`. Sia le tuple di R sia quelle di S che non partecipano al join vengono completate ed inserite nel risultato.
- `LEFT`. Le tuple di R che non partecipano al join vengono completate ed inserite nel risultato.
- `RIGHT`. Le tuple di S che non partecipano al join vengono completate ed inserite nel risultato.

Esempio 3.16 Supponiamo di voler visualizzare per ogni video contenente un film di Tim Burton di genere fantastico la sua collocazione, il titolo ed i codici dei clienti che l'hanno eventualmente noleggiato:

```
SELECT colloc, titolo, codCli
FROM Film NATURAL JOIN Video NATURAL LEFT OUTER JOIN Noleggio
WHERE regista = 'tim burton' AND genere = 'fantastico';
```

Senza l'utilizzo dell'outer join i video, quali il 1123, che non sono mai stati noleggiati non avrebbero fatto parte del risultato. Il risultato dell'interrogazione è illustrato nella Figura 3.6. □

3.2.5 Funzioni di gruppo e raggruppamento

Un'importante funzionalità che SQL aggiunge all'algebra relazionale è quella di poter estrarre dalle tuple di una relazione informazioni riassuntive, ottenute aggregando opportunamente i valori presenti in diverse tuple. I due principali costrutti che forniscono questa funzionalità sono le funzioni di gruppo e l'operatore di raggruppamento.

titolo
big fish
la sposa cadavere
la fabbrica di cioccolato
le iene

colloc	titolo	codCli
1113	big fish	6635
1113	big fish	6642
1114	big fish	6610
1115	edward mani di forbice	6635
1115	edward mani di forbice	6610
1122	la fabbrica di cioccolato	6635
1122	la fabbrica di cioccolato	6642
1123	la fabbrica di cioccolato	?

Figura 3.6: Risultati delle interrogazioni degli Esempi 3.15 e 3.16

3.2.5.1 Funzioni di gruppo

Le funzioni di gruppo, spesso chiamate anche *funzioni aggregate*, possono essere utilizzate nella clausola di proiezione di un'interrogazione e permettono di estrarre informazioni riassuntive da insiemi di valori. Una funzione di gruppo si applica all'insieme dei valori estratti dalle tuple che soddisfano la clausola di qualificazione dell'interrogazione. Le principali funzioni di gruppo previste da SQL sono:

- `MAX` determina il massimo in un insieme di valori;
- `MIN` determina il minimo in un insieme di valori;
- `SUM` esegue la somma dei valori in un insieme;
- `AVG` esegue la media dei valori in un insieme;
- `COUNT` determina la cardinalità di un insieme.

Oltre alle funzioni precedenti, in SQL:2003 sono previste funzioni per il calcolo della deviazione standard (funzione `STDEV`) e della varianza (funzione `VAR`). Il resto della discussione non tratterà queste funzioni addizionali.

Le funzioni `SUM` e `AVG` sono definite solo per insiemi di valori numerici, mentre le funzioni `MAX`, `MIN` e `COUNT` possono essere applicate anche ad altri insiemi di valori. Tutte le funzioni possono essere usate con il qualificatore `DISTINCT`; in tal caso, eventuali valori duplicati sono eliminati prima di applicare la funzione (notare che l'eliminazione dei duplicati è significativa solo per le funzioni `SUM`, `AVG` e `COUNT`). Ad eccezione della funzione `COUNT`, tutte le funzioni devono operare solo su insiemi di valori semplici (ad esempio insiemi di numeri o di stringhe) e non su insiemi di tuple. Nel caso più semplice tale insieme è denotato da un nome di colonna; possono essere tuttavia usate anche espressioni aritmetiche. La funzione `COUNT` può avere tre tipi di argomenti:

1. un nome di colonna: in tal caso la funzione restituisce il numero di valori non nulli presenti nella colonna;

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

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

Figura 3.7: Risultati delle interrogazioni dell'Esempio 3.17

2. un nome di colonna preceduto dal qualificatore `DISTINCT`: in tal caso la funzione restituisce il numero di valori distinti e non nulli presenti nella colonna;

3. il carattere speciale '`*`': in tal caso la funzione restituisce il numero di tuple.

Come nel caso delle espressioni aritmetiche, le colonne della relazione risultato ottenute dall'applicazione di funzioni di gruppo sono colonne virtuali; ad una colonna virtuale, per default, viene assegnato il nome della funzione usata nel calcolo della colonna. Le funzioni di gruppo possono a loro volta essere usate in espressioni aritmetiche.

Esempio 3.17 Consideriamo le seguenti interrogazioni.

Q1 Vogliamo 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;
```

Q2 Vogliamo ora 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';
```

Il risultato delle interrogazioni, che contiene in entrambi i casi una sola tupla, è illustrato nella Figura 3.7. □

Un ultimo aspetto da puntualizzare riguarda quale valore viene restituito da una funzione di gruppo calcolata su un insieme vuoto. Questa situazione si verifica se non esiste alcuna tupla che soddisfa la condizione di ricerca dell'interrogazione. Il valore calcolato dipende dalla specifica funzione. La funzione `COUNT` restituisce il valore numerico 0, mentre tutte le altre funzioni restituiscono il valore nullo.

3.2.5.2 Raggruppamento

L'operatore di raggruppamento permette di suddividere le tuple di una relazione in base al valore di una o più colonne di tale relazione. Le colonne da usare nel partizionamento sono specificate in un'apposita clausola `GROUP BY` del comando `SELECT`. Le tuple su cui si esegue il partizionamento sono solo le tuple che verificano la clausola di qualificazione del comando `SELECT`. Le tuple che non verificano tale clausola non partecipano al partizionamento.

Il risultato di un comando `SELECT` che contiene la clausola di raggruppamento contiene tante tuple quanti sono i gruppi di tuple risultanti dal partizionamento. La clausola di proiezione di un'interrogazione contenente una clausola `GROUP BY` può includere solamente colonne che compaiono nella clausola `GROUP BY` oppure funzioni di gruppo. Non è invece possibile includere in tale clausola colonne delle relazioni che non compaiano nella clausola `GROUP BY`. Una volta raggruppate le tuple, infatti, si "perde" la possibilità di riferire il valore di tali colonne delle tuple originarie: il risultato contiene una sola tupla per ogni gruppo mentre le singole colonne delle tuple nel gruppo possono contenere diversi valori.

Esempio 3.18 Per ogni regista vogliamo determinare quanti suoi film sono presenti in catalogo, 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
GROUP BY regista;
```

Il risultato dell'interrogazione è illustrato nella Figura 3.8. Essendo quattro i gruppi ottenuti dal partizionamento in base ai valori della colonna `regista`, il risultato contiene quattro tuple. Notiamo che sono stati assegnati dei nomi alle colonne virtuali i cui valori sono calcolati mediante funzioni di gruppo. □

Il raggruppamento può essere applicato a relazioni ottenute come risultato di operazioni di join. In particolare, la clausola `GROUP BY` può avere come argomenti colonne di relazioni diverse, tra quelle che partecipano all'operazione di join.

Esempio 3.19 Vogliamo partizionare i noleggi in base al cliente che li ha effettuati ed al regista del film noleggiato. Per ogni gruppo, vogliamo 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, regista;
```

regista	numF	numG	minV	avgV	maxV
emir kusturica	1	1	3.20	3.20	3.20
tim burton	8	5	3.00	3.58	4.00
gabriele salvatores	3	3	3.00	3.43	3.80
quentin tarantino	2	1	3.50	3.75	4.00

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

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

Figura 3.8: Risultati delle interrogazioni degli Esempi 3.18, 3.19 e 3.20

Il risultato dell'interrogazione è illustrato nella Figura 3.8. □

SQL prevede inoltre la possibilità di specificare condizioni di ricerca su gruppi di tuple. Queste condizioni di ricerca permettono di scartare alcuni dei gruppi ricavati dal partizionamento ottenuto in base alla clausola `GROUP BY`. Condizioni di ricerca su gruppi di tuple sono specificate nell'apposita clausola `HAVING`. La condizione di ricerca nella clausola `HAVING` può essere una combinazione booleana di più predicati. Tali predicati, tuttavia, possono essere solo predicati che coinvolgono funzioni di gruppo. La clausola `HAVING`, infatti, non può contenere condizioni sui valori delle colonne delle singole tuple.[11] Notiamo infine che non vi è alcuna relazione tra le funzioni di gruppo eventualmente usate nella clausola `SELECT` e quelle usate nella clausola `HAVING`.

Esempio 3.20 Vogliamo eseguire un'interrogazione simile all'interrogazione dell'Esempio 3.19, ma siamo interessati solo ai film girati prima del 2000 ed ai gruppi che contengono almeno due tuple. L'interrogazione, che restituisce quanti sono tali film, di quanti generi diversi e la valutazione minima, media e massima di tali film, è la seguente:

[11] In realtà SQL:2003 prevede due utili funzioni su insiemi, `EVERY` e `ANY`, che permettono di condizionare l'inclusione di un gruppo nel risultato al fatto che una condizione su singola tupla sia soddisfatta, rispettivamente, da tutte le tuple o da almeno una tupla del gruppo. Tali funzioni non vengono però considerate nella trattazione per mancanza di supporto nei DBMS commerciali.

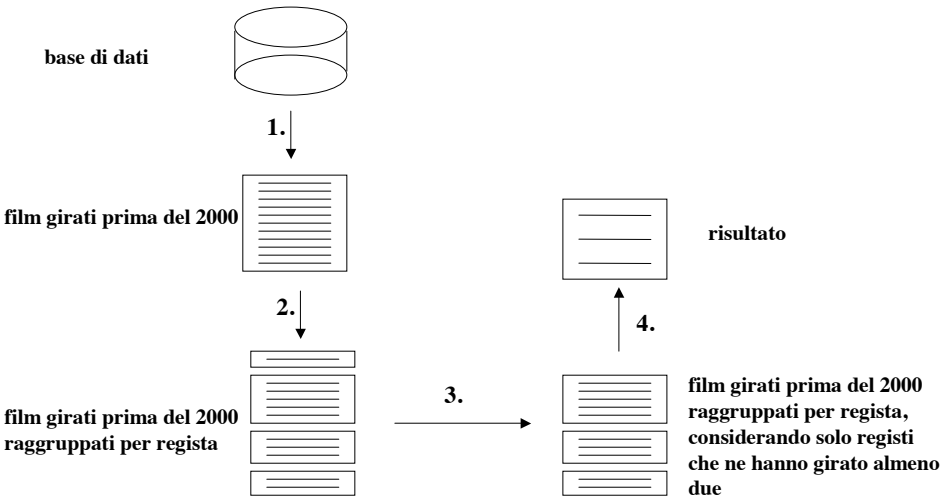

Figura 3.9: Esecuzione di interrogazioni con raggruppamento e clausola `HAVING`

```
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;
```

Il risultato dell'interrogazione è illustrato nella Figura 3.8. □

Per chiarire bene l'uso del meccanismo di partizionamento possiamo considerare il seguente semplice modello di "esecuzione" di interrogazioni, illustrato graficamente nella Figura 3.9 relativamente all'esecuzione dell'interrogazione dell'Esempio 3.20:

1. Si applica la condizione di ricerca specificata nella clausola `WHERE` a tutte le tuple delle relazioni oggetto dell'interrogazione. La valutazione avviene sulle singole tuple, quindi non è ovviamente possibile utilizzare funzioni di gruppo nella clausola `WHERE`.

2. Alle tuple ottenute al passo precedente si applica il partizionamento specificato dalla clausola `GROUP BY`.

3. Ad ogni gruppo di tuple ottenuto al passo precedente si applica la condizione di ricerca per i gruppi, specificata dalla clausola `HAVING`. Notiamo che la valutazione di tale condizione implica il calcolo di funzioni di gruppo.

AND	**T**	**F**	**?**
T	T	F	?
F	F	F	F
?	?	F	?

OR	**T**	**F**	**?**
T	T	T	T
F	T	F	?
?	T	?	?

NOT	**T**
T	F
F	T
?	?

Figura 3.10: Tabelle di verità per la valutazione di combinazioni booleane di predicati

4. I gruppi ottenuti al passo precedente sono i gruppi che verificano l'interrogazione. Per ognuno di tali gruppi, vengono calcolate le funzioni di gruppo specificate nella clausola di proiezione dell'interrogazione. I valori restituiti da tali funzioni costituiscono il risultato dell'interrogazione.

3.2.6 Valori nulli

Un aspetto importante nella valutazione delle interrogazioni, di cui bisogna tener conto per una loro corretta formulazione, riguarda i valori nulli. Come già discusso, è possibile che una o più tuple in una relazione abbiano valori nulli per alcune colonne. Una prima questione riguarda qual è il risultato della valutazione di una condizione di ricerca applicata a tuple le cui colonne abbiano valori nulli. A tale riguardo, SQL usa una logica booleana a tre valori; i valori sono:

`TRUE` (T), `FALSE` (F), `UNKNOWN` (?).

Tale logica, rispetto alla classica algebra di Boole, introduce un terzo valore di verità, `UNKNOWN`, il quale semplicemente indica che il valore di verità di una condizione di ricerca applicata ad una data tupla non è determinabile (ovvero è sconosciuto). Il valore di verità di una condizione di ricerca è determinato in base ai seguenti, semplici principi:

- un predicato semplice valutato su una colonna di una tupla che ha valore nullo dà come risultato della valutazione il valore di verità `UNKNOWN`; notiamo anche che nel caso di un predicato della forma $C = C'$, con C e C' colonne aventi entrambe valore nullo, la valutazione restituisce il valore di verità `UNKNOWN`;

- il valore di verità di una combinazione booleana di predicati viene calcolato in base alle tabelle di verità riportate nella Figura 3.10.

Le tuple per cui il valore di verità della condizione di ricerca è `FALSE` o `UNKNOWN` non verificano la condizione di ricerca e non vengono, pertanto, restituite dall'interrogazione.

Esempio 3.21 Consideriamo le seguenti interrogazioni:

Q1
```
SELECT colloc FROM Noleggio
WHERE codCli = 6635 AND dataRest > DATE '15-Mar-2006';
```

L'interrogazione non restituisce i noleggi in corso, cioè quelli per cui `dataRest` ha valore `NULL`. I numeri di collocazione restituiti sono quindi 1121, 1122, 1113 e 1129.

Q2
```
SELECT colloc FROM Noleggio
WHERE codCli = 6635 AND (dataNol > DATE '20-Mar-2006'
                         OR dataRest > DATE '20-Mar-2006');
```

L'interrogazione restituisce anche noleggi in corso, cioè quelli per cui `dataRest` ha valore `NULL`, purché siano iniziati dopo il 20 Marzo. I numeri di collocazione restituiti sono 1127 e 1125.

Q3
```
SELECT colloc FROM Noleggio
WHERE codCli = 6635 AND NOT dataRest < DATE '15-Mar-2006';
```

L'interrogazione non restituisce i noleggi in corso, cioè quelli per cui `dataRest` ha valore `NULL`. I numeri di collocazione restituiti sono 1121, 1122, 1113 e 1129.

È interessante infine notare come interrogazioni quali le seguenti:

```
SELECT colloc FROM Noleggio
WHERE dataRest = CURRENT_DATE OR NOT dataRest = CURRENT_DATE;

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

non restituiscano tutti i noleggi, come potrebbe sembrare a prima vista, ma solo i noleggi terminati, cioè quelli per cui `dataRest` ha valore non nullo. □

Dato che le tuple con colonne aventi valore nullo non vengono restituite da predicati su tali colonne, un problema riguarda come ritrovare queste tuple. A tale scopo SQL fornisce il predicato speciale `IS NULL`, di cui esiste la forma negata `IS NOT NULL`, utile per eliminare dal risultato dell'interrogazione tuple con valori nulli. La valutazione del predicato `IS NULL`, applicato ad una data colonna di una tupla, restituisce il valore di verità `TRUE` se la tupla ha valore nullo per tale colonna. Viceversa, la valutazione del predicato `IS NOT NULL` restituisce `TRUE` se la tupla ha un valore diverso dal valore nullo.

Esempio 3.22 Per determinare le collocazioni dei video attualmente in noleggio al cliente 6635 formuliamo la seguente interrogazione:

```
SELECT colloc FROM Noleggio
WHERE codCli = 6635 AND dataRest IS NULL;
```

che restituisce i numeri di collocazione 1127 e 1125. Analogamente, l'interrogazione:

```
SELECT colloc FROM Noleggio
WHERE codCli = 6635 AND dataRest IS NOT NULL;
```

permette di determinare le collocazioni dei video che sono stati noleggiati e restituiti dal cliente 6635. □

Nelle espressioni (ad esempio aritmetiche), se un argomento è `NULL`, il valore dell'intera espressione è `NULL`. Per questo motivo, negli esempi dei paragrafi precedenti, le tuple relative a noleggi correnti hanno durata (cioè valore dell'espressione `dataRest - dataNol`) indeterminata, cioè `NULL`. Nel calcolo di una funzione di gruppo, invece, vengono escluse le tuple che hanno valore nullo nella colonna su cui la funzione è calcolata. Questo ha come conseguenza che, ad esempio, in presenza di valori nulli, l'espressione `SUM`$(e_1 + e_2)$ può dare un risultato diverso da `SUM(`e_1`) + SUM(`e_2`)`. Una funzione di gruppo restituisce infine `NULL` se applicata ad un insieme vuoto o ad un insieme contenente il solo valore `NULL`.

Come visto nell'esempio precedente, se due espressioni e_1 ed e_2 hanno valore `NULL`, $e_1 = e_2$ non è vero (è `UNKNOWN`). Il vincolo `UNIQUE` viene definito basandosi sull'operatore di uguaglianza e pertanto, poiché due valori nulli non sono considerati uguali, la presenza di due tuple distinte con valori nulli non viene impedita dalla specifica di un vincolo `UNIQUE`. Questo è il motivo per cui, ad esempio, la presenza di due tuple nella relazione `Noleggio` con lo stesso valore per `colloc` e valore nullo per `dataRest` non porta ad una violazione del vincolo `UNIQUE(colloc,dataRest)` definito per la relazione `Noleggio` nell'Esempio 3.2.

Due espressioni con valore nullo non sono quindi valutate uguali, ma non sono distinte, cioè vengono considerate duplicati. Ciò vuol dire che, nel caso di `SELECT DISTINCT`, si ha al più un `NULL` nel risultato e che, nel caso di `GROUP BY`, si ha al più un gruppo per il valore `NULL`. Menzioniamo infine che esiste un predicato SQL `IS DISTINCT FROM` che coincide con `<>` tranne che per il valore nullo, cioè se e_1 ed e_2 sono `NULL`, e_1 `<>` e_2 restituisce `UNKNOWN`, mentre e_1 `IS DISTINCT FROM` e_2 restituisce `FALSE`.

3.2.7 Sotto-interrogazioni

Uno dei motivi che rendono SQL un linguaggio potente è la possibilità di esprimere interrogazioni complesse in termini di interrogazioni più semplici. Come abbiamo già visto, la maggioranza dei predicati in SQL esegue confronti tra una colonna ed un valore costante. Molto spesso può essere necessario ottenere tali valori dalla base di dati, tramite un'opportuna interrogazione. Per agevolare la formulazione di tali interrogazioni, SQL fornisce il meccanismo delle *sotto-interrogazioni*. In particolare, la clausola `WHERE` di un'interrogazione, detta *interrogazione esterna*, può contenere un'altra interrogazione, detta *sotto-interrogazione* (subquery). Un primo semplice esempio di uso di sotto-interrogazioni è il seguente.

Esempio 3.23 Vogliamo determinare il titolo di tutti i film che hanno la stessa valutazione di "Le iene". Formuliamo la seguente interrogazione:

```
SELECT titolo FROM Film
WHERE valutaz = (SELECT valutaz FROM Film
                 WHERE titolo = 'le iene');
```

che restituisce i valori `nightmare before christmas`, `ed wood`, `la fabbrica di cioccolato` e `le iene`. La sotto-interrogazione:

```
SELECT valutaz FROM Film WHERE titolo = 'le iene';
```

restituisce come valore `4.00`, usato poi nel predicato dell'interrogazione esterna per determinare le tuple che soddisfano la condizione di ricerca. □

Notiamo che un approccio in cui si determinasse con una interrogazione separata la valutazione di "Le iene" e poi si usasse tale valore direttamente in un'interrogazione per determinare i film con tale valutazione (quindi senza l'uso delle sotto-interrogazioni) avrebbe una serie di inconvenienti. Prima di tutto, richiederebbe due interrogazioni diverse, invece di una. Lo svantaggio maggiore sarebbe però che una modifica alla valutazione richiederebbe di riscrivere la seconda interrogazione.

Tramite il meccanismo delle sotto-interrogazioni è possibile formulare richieste più complesse del semplice esempio visto in precedenza.

Esempio 3.24 Vogliamo determinare i film la cui valutazione è superiore alla media:

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

La sotto-interrogazione restitusce il valore `3.55`, quindi solo i film con valutazione superiore a tale valore vengono selezionati. Il risultato dell'interrogazione è illustrato nella Figura 3.11. □

Negli esempi visti finora ogni sotto-interrogazione restituisce un solo valore. Tali sotto-interrogazioni sono note come *sotto-interrogazioni scalari*. Se una sotto-interrogazione scalare restituisce più di una tupla, si genera un errore a run-time. Una particolarità delle sotto-interrogazioni scalari è che, se nessuna tupla verifica la sotto-interrogazione, viene restituito il valore `NULL`. Se, invece, la sotto-interrogazione resituisce un insieme di valori (*table subquery*) è necessario specificare come tali valori devono essere usati nel predicato. A tale scopo vengono introdotti i *quantificatori* `ANY` ed `ALL` che sono inseriti tra l'operatore di confronto e la sotto-interrogazione. `ANY` (come il suo sinonimo `SOME`) restituisce il valore di verità `TRUE` quando la valutazione dell'operatore di confronto restituisce `TRUE` su almeno una delle tuple ottenute dalla valutazione della sotto-interrogazione. In particolare, restituisce `FALSE` se la sotto-interrogazione non restituisce tuple. `ALL`, invece, restituisce il valore di verità `TRUE` quando la valutazione dell'operatore di confronto restituisce `TRUE` su tutte le tuple ottenute dalla valutazione della sotto-interrogazione. In particolare, restituisce `TRUE` se la sotto-interrogazione non restituisce tuple.

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

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

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

Figura 3.11: Risultati delle interrogazioni degli Esempi 3.24, 3.25 e 3.26

Esempio 3.25 Vogliamo determinare titolo, regista ed anno dei film più vecchi di *almeno un* film di Quentin Tarantino. Formuliamo la seguente interrogazione:[12]

```
SELECT titolo, regista, anno
FROM Film
WHERE anno < ANY (SELECT anno FROM Film
                  WHERE regista = 'quentin tarantino');
```

il cui risultato è illustrato nella Figura 3.11. □

Esempio 3.26 Vogliamo ora determinare titolo, regista ed anno dei film precedenti a *tutti* i film di Quentin Tarantino. Formuliamo la seguente interrogazione:

```
SELECT titolo, regista, anno
FROM Film
WHERE anno < ALL (SELECT anno FROM Film
                  WHERE regista = 'quentin tarantino');
```

il cui risultato è illustrato nella Figura 3.11. □

Sono definite le seguenti abbreviazioni per `ANY` ed `ALL`:

- `IN` equivalente ad `= ANY`;
- `NOT IN` equivalente ad $\neq$ `ALL`.

[12] Notiamo che sono possibili formulazione alternative di questa interrogazione e della successiva, tramite l'uso, rispettivamente, delle funzioni di gruppo `MIN` e `MAX`.

È inoltre possibile selezionare più di una colonna tramite una sotto-interrogazione. In tal caso è necessario apporre delle parentesi alla lista delle colonne a sinistra dell'operatore di confronto.

Esempio 3.27 Consideriamo le seguenti interrogazioni.

Q1 Elencare il titolo dei film di Quentin Tarantino presenti nella videoteca, 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');
```

L'interrogazione restituisce `pulp fiction`.

Q2 Elencare il titolo dei film di Quentin Tarantino presenti nella videoteca, usciti in un anno in cui non sono usciti film di Tim Burton presenti nella videoteca:

```
SELECT titolo FROM Film
WHERE regista = 'quentin tarantino'
      AND anno NOT IN (SELECT anno FROM Film
                       WHERE regista = 'tim burton');
```

L'interrogazione restituisce `le iene`.

Q3 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. Notiamo che l'interrogazione principale contiene la condizione `regista <> 'tim burton'` per evitare di restituire i film di Tim Burton stessi. □

Una sotto-interrogazione può includere nella propria condizione di ricerca un'altra sotto-interrogazione, condizioni di join e tutti i predicati visti finora. La clausola di qualificazione di un'interrogazione può contenere una qualsiasi combinazione di condizioni normali e condizioni con sotto-interrogazioni. Le sotto-interrogazioni possono essere usate anche all'interno dei comandi di modifica previsti da SQL (vedi Paragrafo 3.3).