SOTTO-INTERROGAZIONI **CORRELATE**

- Negli esempi visti ogni sotto-interrogazione
  - viene eseguita **una volta**
  - il risultato è usato per selezionare tutte le tuple
- Vogliamo un meccanismo per definire sotto-interrogazioni che dipendono dalla specifica **tupla candidata**
- Esempio: si vogliono determinare titolo, regista ed anno dei film la cui valutazione è superiore alla media delle valutazioni dei film **dello stesso regista**
  - vogliamo confrontare la valutazione di ciascun film con la media delle valutazioni dei soli film dello stesso regista

SOTTO-INTERROGAZIONI CORRELATE

- Serve un'interrogazione esterna che selezioni i film in base ad un predicato sulla valutazione:

  SELECT titolo, **regista**, anno FROM Film

  WHERE valutaz > (*media delle valutazioni dei film*

  del regista del film candidato);

- La sotto-interrogazione deve calcolare la media delle valutazioni dei film del regista di ogni tupla candidata

  SELECT AVG(valutaz) FROM Film

  WHERE regista = (*valore di **regista** nella tupla candidata*);

SOTTO-INTERROGAZIONI CORRELATE

STRATEGIA NAIVE E IRREALISTICA PER L'ESECUZIONE PER ILLUSTRARNE IL SIGNIFICATO

Quando l'interrogazione esterna valuta se una tupla candidata appartiene al risultato

- *chiama* la sotto-interrogazione sul regista **x** del film in esame
  - **x** gioca il ruolo di un parametro nella "chiamata"
- la sotto-interrogazione calcola la media **m** delle valutazioni dei film di **x**
- l'interrogazione esterna confrontare la valutazione del film in esame con **m**

SOTTO-INTERROGAZIONI CORRELATE

- In questo tipo di interrogazioni ogni esecuzione della sotto-interrogazione è **correlata** al valore di uno o più attributi delle tuple candidate nella interrogazione principale da cui il nome
- valgono le usuali regole di scope
  - nella subquery si può fare riferimento a quanto definito nella query esterna ma non viceversa
  - eventuali nomi uguali nella subquery *coprono* quelli della query esterna
- in molti casi le relazioni nella subquery coincidono con un sottoinsieme di quelle nella query esterna
  - la relazione nella subquery copre quella della query esterna
  - per poter fare riferimento alle colonne delle tuple candidate nella subquery si fa uso degli **alias di relazione** che sono un meccanismo di renaming

ALIAS DI RELAZIONE

- Analogo agli alias di colonna
- Si definisce nella clausola FROM facendo seguire il nome della relazione da
  - un identificatore, oppure
  - AS seguito da un identificatore
- Esempio:

```
SELECT titolo, regista, anno FROM Film X
WHERE valutaz > (     SELECT AVG(valutaz)
                      FROM Film
                      WHERE regista = X.regista);
```

- Utile anche per
  - abbreviare la scrittura di query

```
SELECT X.a FROM
LaMiaTabellaConNomeSignificativoQuindiLungo X
```

  - fare riferimento a due diverse tuple della stessa relazione

```
SELECT DISTINCT X.regista FROM Film X, Film Y
WHERE X.anno = Y.anno AND X.regista = Y.regista
                AND X.titolo <> Y.titolo;
```

EXISTS E NOT EXISTS

- Le sotto-interrogazioni correlate sono spesso usate in combinazione con l'operatore EXISTS (eventualmente in forma negata NOT EXISTS)
- Data una interrogazione Q il predicato EXISTS(Q)
  - restituisce il valore Booleano TRUE se sq restituisce almeno una tupla
  - restituisce il valore Booleano FALSE altrimenti
  - non restituisce mai il valore UNKNOWN
- Esempio: i registi di cui sono usciti (almeno) due film diversi lo stesso anno

```
SELECT DISTINCT regista FROM Film X
WHERE EXISTS (    SELECT * FROM Film
                        WHERE regista = X.regista
                              AND anno = X.anno
                              AND titolo <> X.titolo);
```

INTERSEZIONE E DIFFERENZA ⇔ [NOT] EXISTS

- Intersezione e differenza si possono definire in SQL anche tramite [NOT] EXISTS
- Esempio: determinare gli anni in cui sono usciti sia film di Tim Burton ma non sia film di Quentin Tarantino

```
SELECT anno FROM Film WHERE regista = 'tim burton'
INTERSECT                                          MINUS
SELECT anno FROM Film WHERE regista = 'quentin tarantino'
                                                   diventa
SELECT DISTINCT anno FROM Film
WHERE regista = 'tim burton' AND
NOT EXISTS (     SELECT * FROM Film F
                 WHERE regista = 'quentin tarantino'
                       AND anno = F.anno);
```

- Altra soluzione usando IN?

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

INTERSEZIONE E DIFFERENZA ⇔ [NOT] EXISTS

In generale

SELECT C1,...,CK FROM R1 WHERE P1
INTERSECT
SELECT C1,...,CK FROM **R2 WHERE P2**

si può esprimere anche come

SELECT C1,...,CK FROM R1 WHERE P1 AND
EXISTS (SELECT * FROM **R2**
WHERE P2 AND R1.C1=R2.C1 AND ...
AND R1.CK = R2.CK)

SELECT C1,...,CK FROM R1 WHERE P1
MINUS
SELECT C1,...,CK FROM **R2 WHERE P2**

si può esprimere anche come

SELECT C1,...,CK FROM R1 WHERE P1 AND
NOT EXISTS (SELECT * FROM **R2**
WHERE P2 AND R1.C1=R2.C1 AND...
AND R1.CK = R2.CK)

Divisione

- le sotto-interrogazioni correlate e l'operatore di NOT EXISTS permettono di esprimere l'operazione di divisione, per cui SQL non prevede un operatore apposito
- la specifica della divisione in SQL richiede di ragionare in base al concetto di controesempio, in base alla tautologia

$$\forall z(\exists y\ p(z,y)) \Leftrightarrow \neg\ \exists z(\neg\ \exists y\ p(z,y)$$

Divisione

- esempio: determinare i codici dei clienti che hanno noleggiato tutti i film di Tim Burton
- viene "riformulata" come: determinare i codici dei clienti per cui non è possibile determinare un film di Tim Burton che il cliente non ha mai noleggiato

```
SELECT DISTINCT codCli FROM Noleggio X
  WHERE NOT EXISTS (SELECT * FROM Film F
                    WHERE regista = 'tim burton' AND
                    NOT EXISTS (SELECT *
                                FROM Noleggio NATURAL JOIN Video
                                 WHERE codCli = X.codCli
                                    AND titolo = F.titolo
                                    AND regista = F.regista));
```

Divisione

- un modo alternativo di esprimere la divisione è mediante l'uso di funzioni di gruppo
- per determinare i clienti che hanno noleggiato tutti i film di Tim Burton confrontiamo il numero di film di Tim Burton con il numero dei film di Tim Burton noleggiati dal cliente

```
SELECT codCli FROM Noleggio NATURAL JOIN Video
WHERE regista = 'tim burton'
GROUP BY codCli
HAVING COUNT(DISTINCT titolo) =
                    (SELECT COUNT(DISTINCT titolo)
                     FROM Film
                     WHERE regista = 'tim burton');
```