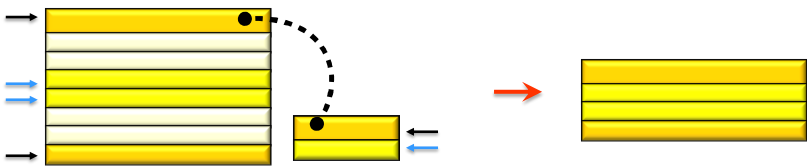


Il linguaggio SQL: query innestate



Dario Maio
<http://bias.csr.unibo.it/maio/>

Annalisa Franco

Il linguaggio SQL: query innestate 1

DB di riferimento per gli esempi

Imp

CodImp	Nome	Sede	Ruolo	Stipendio
E001	Rossi	S01	Analista	2000
E002	Verdi	S02	Sistemista	1500
E003	Bianchi	S01	Programmatore	1000
E004	Gialli	S03	Programmatore	1000
E005	Neri	S02	Analista	2500
E006	Grigi	S01	Sistemista	1100
E007	Violetti	S01	Programmatore	1000
E008	Aranci	S02	Programmatore	1200
E009	Rosi	S03	Analista	2200
E010	Porpori	S04	Analista	2200
E011	Magenti	S04	Sistemista	1200

Sedi

Sede	Responsabile	Citta
S01	Biondi	Milano
S02	Mori	Bologna
S03	Fulvi	Milano
S04	Castani	Bologna

Prog

CodProg	Citta
P01	Milano
P01	Bologna
P02	Bologna

Il linguaggio SQL: query innestate 2

Subquery

➤ Oltre alla forma "flat" vista sinora, in SQL è anche possibile esprimere **condizioni che si basano sul risultato di altre interrogazioni** (subquery, o query innestate o query nidificate):

```

SELECT  CodImp  -- impiegati delle sedi di Milano
FROM    Imp
WHERE   Sede IN (SELECT  Sede
                  FROM    Sedi
                  WHERE   Citta = 'Milano')

```


Sede
S01
S03

➤ La subquery restituisce l'insieme di sedi ('S01','S03'), e quindi il predicato nella clausola WHERE esterna equivale a:

```

WHERE   Sede IN ('S01', 'S03')

```


 Il linguaggio SQL: query innestate 3

Subquery scalari

➤ Gli **operatori di confronto** =, <, ... si possono usare solo se la subquery restituisce non più di una tupla (**subquery "scalare"**):

```

SELECT  CodImp  -- impiegati con stipendio minimo
FROM    Imp
WHERE   Stipendio = (SELECT  MIN(Stipendio)
                  FROM    Imp)

```

MinStip
1000


➤ La presenza di vincoli può essere sfruttata a tale scopo:

```

SELECT  Responsabile
FROM    Sedi
WHERE   Sede = (SELECT  Sede -- al massimo una sede
                  FROM    Imp
                  WHERE   CodImp = 'E001')

```

Sede
S01


 Il linguaggio SQL: query innestate 4

Subquery: caso generale

➤ Se la subquery può restituire più di un valore si devono usare le forme:

- **<op> ANY**: la relazione <op> vale per **almeno uno** dei valori;
- **<op> ALL**: la relazione <op> vale per **tutti** i valori.

```

SELECT Responsabile
FROM Sedi
WHERE Sede = ANY (SELECT Sede
                  FROM Imp
                  WHERE Stipendio > 1500)

SELECT CodImp -- impiegati con stipendio minimo
FROM Imp
WHERE Stipendio <= ALL (SELECT Stipendio
                      FROM Imp)
  
```

➤ La forma = **ANY** equivale a **IN**.


 Il linguaggio SQL: query innestate 5

Subquery: livelli multipli di innestamento

➤ Una subquery può fare uso a sua volta di altre subquery. Il risultato si può ottenere risolvendo a partire dal blocco più interno:


```

SELECT CodImp
FROM Imp
WHERE Sede IN (SELECT Sede
              FROM Sedi
              WHERE Citta NOT IN (SELECT Citta
                                FROM Prog
                                WHERE CodProg = 'P02'))
  
```

➤ **Attenzione a non sbagliare quando ci sono negazioni!** Nell'esempio, i due blocchi interni **NON** sono equivalenti a:

```

WHERE Sede IN (SELECT Sede
              FROM Sedi, Prog
              WHERE Sedi.Citta <> Prog.Citta
              AND Prog.CodProg = 'P02')
  
```


 Il linguaggio SQL: query innestate 6



Subquery: quantificatore esistenziale

- Mediante **EXISTS** (SELECT * ...) è possibile verificare se il risultato di una subquery restituisce almeno una tupla:

```
SELECT Sede
FROM   Sedi S
WHERE  EXISTS (SELECT *
               FROM   Imp
               WHERE  Ruolo = 'Programmatore')
```

- Facendo uso di **NOT EXISTS** il predicato è vero se la subquery non restituisce alcuna tupla.
- In entrambi i casi la cosa non è molto "interessante" in quanto il risultato della subquery è sempre lo stesso, **ovvero non dipende dalla specifica tupla del blocco esterno.**

Il linguaggio SQL: query innestate



7



Subquery correlate

- Se la subquery fa riferimento a "variabili" definite in un blocco esterno, allora si dice che è **correlata**:

```
SELECT Sede -- sedi con almeno un programmatore
FROM   Sedi S
WHERE  EXISTS (SELECT *
               FROM   Imp
               WHERE  Ruolo = 'Programmatore'
               AND    Sede = S.Sede)
```

- Adesso il risultato della query innestata dipende dalla sede specifica, e la semantica quindi diventa:
per ogni tupla del blocco esterno, considera il valore di S.Sede e risolvi la query innestata.

Il linguaggio SQL: query innestate



8

Subquery: "unnesting" (1)

- È spesso possibile ricondursi a una forma "piatta", ma la cosa non è sempre così ovvia. Ad esempio, l'interrogazione precedente si può anche scrivere:


```
SELECT DISTINCT Sede
FROM Sedi S, Imp I
WHERE S.Sede = I.Sede
AND I.Ruolo = 'Programmatore'
```
- Si noti la presenza del **DISTINCT**.
- La forma innestata è "più procedurale" di quella piatta e, a seconda dei casi, può risultare più semplice da derivare.
- Si ricordi comunque che in una subquery **non si possono usare operatori insiemistici** (UNION, INTERSECT e EXCEPT) e che **una subquery può comparire solo come operando destro in un predicato**.

Il linguaggio SQL: query innestate 9

Subquery: "unnesting" (2)

- Con la negazione le cose tendono a complicarsi. Ad esempio, per trovare le **sedi senza programmatori**, nella forma innestata basta sostituire **NOT EXISTS** a **EXISTS**,


```
SELECT Sede -- sedi senza programmatori
FROM Sedi S
WHERE NOT EXISTS (SELECT *
                  FROM Imp
                  WHERE Ruolo = 'Programmatore'
                  AND Sede = S.Sede)
```

ma la forma piatta:

```
SELECT DISTINCT Sede
FROM Sedi S JOIN Imp I ON (S.Sede = I.Sede)
WHERE I.Ruolo <> 'Programmatore'
```

restituisce le sedi in cui lavora almeno un impiegato con ruolo diverso da un programmatore.

Sede
S01
S02
S03
S04

Il linguaggio SQL: query innestate 10

Subquery: "unnesting" (3)

➤ Una soluzione "piatta" corretta (ma complessa) è:

```
SELECT DISTINCT Sede
FROM Sedi S LEFT OUTER JOIN Imp I ON
  (S.Sede = I.Sede) AND (I.Ruolo = 'Programmatore')
WHERE I.CodImp IS NULL
```

Sede
S04

- il join restituisce come risultato le sedi con relativi impiegati nel ruolo programmatore comprese le sedi senza programmatori;
- la clausola WHERE seleziona solo queste ultime.

➤ È facile sbagliare. Ad esempio la seguente query **non è corretta**:

```
SELECT DISTINCT Sede
FROM Sedi S LEFT OUTER JOIN Imp I ON (S.Sede = I.Sede)
WHERE I.Ruolo = 'Programmatore'
AND I.CodImp IS NULL
```

Sede

- il join restituisce come risultato le sedi con relativi impiegati (di qualsiasi ruolo) comprese le sedi senza impiegati;
- la clausola WHERE **non è mai soddisfatta!**

Il linguaggio SQL: query innestate 11

Un esempio complesso

Sede che ha il numero maggiore di impiegati con ruolo 'Programmatore'

```
SELECT Sede, COUNT(CodImp) AS NumProg
FROM Imp I1
WHERE I1.Ruolo = 'Programmatore'
GROUP BY I1.Sede
HAVING COUNT(CodImp) >= ALL
  (SELECT COUNT(CodImp)
   FROM Imp I2
   WHERE I2.Ruolo = 'Programmatore'
   GROUP BY I2.Sede)
```

Sede	NumProg
S01	2

In alternativa...

```
SELECT TOP(1) WITH TIES Sede, COUNT(CodImp)
FROM Imp
WHERE Ruolo = 'Programmatore'
GROUP BY Sede
ORDER BY COUNT(CodImp) DESC
```

N.B. molte query innestate non possono essere semplificate usando la clausola TOP (es. *impiegati che hanno uno stipendio superiore a quello medio per il relativo ruolo*).

Il linguaggio SQL: query innestate 12

Subquery: come eseguire la divisione


- Con le subquery è possibile eseguire la divisione relazionale.
- *"Sedi in cui sono presenti tutti i ruoli"* equivale a *"Sedi in cui **non esiste** un ruolo **non presente**"*:

```

SELECT Sede FROM Sedi S
WHERE NOT EXISTS (SELECT * FROM Imp I1
                  WHERE NOT EXISTS (SELECT * FROM Imp I2
                                    WHERE S.Sede = I2.Sede
                                    AND I1.Ruolo = I2.Ruolo))
  
```

Sede
S01
S02

- Il blocco più interno è valutato **per ogni combinazione** di S e I1.
- Il blocco intermedio funge da **"divisore"** (interessa I1.Ruolo).
- Data una sede S, se in S manca un ruolo:
 - la subquery più interna non restituisce nulla;
 - quindi la subquery intermedia restituisce almeno una tupla;
 - quindi la clausola WHERE non è soddisfatta per S.



Il linguaggio SQL: query innestate

13

La divisione: una formulazione alternativa

- L'espressione *"Sedi in cui sono presenti tutti i ruoli"* equivale anche a *"Sedi per cui il numero di ruoli distinti è **uguale** al numero totale di ruoli presenti"*.

```

SELECT Sede FROM Imp I
GROUP BY Sede
HAVING COUNT (DISTINCT Ruolo) =
      (SELECT COUNT (DISTINCT Ruolo) FROM Imp)
  
```


- Il blocco più interno conta il numero di ruoli distinti;
- il blocco esterno raggruppa gli impiegati per sedi e conta il numero di ruoli distinti su ciascuna sede...
- ...selezionando solo le sedi per cui tale numero è massimo.

Sede	NRuoli
S01	3
S02	3
S03	1

TotRuoli
3

→

Sede
S01
S02



Il linguaggio SQL: query innestate

14

La divisione: il semplice conteggio non è sempre possibile

- L'espressione "Sedi in cui sono presenti tutti i ruoli della sede 'S03'" **NON** equivale a "Sedi per cui il numero di ruoli distinti è uguale al numero di ruoli presenti nella sede 'S03'".
- Occorre imporre un vincolo: solo ruoli presenti in 'S03'.

```
SELECT Sede FROM Imp I
WHERE Ruolo IN (SELECT Ruolo FROM Imp I2
                WHERE Sede = 'S03')
GROUP BY Sede
HAVING COUNT (DISTINCT Ruolo) =
        (SELECT COUNT (DISTINCT Ruolo) FROM Imp
         WHERE Sede = 'S03')
```

Sede
S01
S02
S03

invece, senza condizione...

Sede
S03
S04

Il linguaggio SQL: query innestate

15

Subquery: aggiornamento dei dati

- Le subquery si possono efficacemente usare per aggiornare i dati di una tabella sulla base di criteri che dipendono dal contenuto di altre tabelle:

```
DELETE FROM Imp -- elimina gli impiegati di Bologna
WHERE Sede IN (SELECT Sede
                FROM   Sedi
                WHERE  Citta = 'Bologna')

UPDATE Imp -- aumenta lo stipendio agli impiegati
SET       Stipendio = 1.1*Stipendio
WHERE Sede IN (SELECT S.Sede      -- delle sedi di 'P02'
                FROM   Sede S, Prog P
                WHERE  S.Citta = P.Citta
                AND    P.CodProg = 'P02')
```

Il linguaggio SQL: query innestate

16



Subquery e CHECK

- Facendo uso di subquery nella clausola **CHECK** è possibile esprimere vincoli arbitrariamente complessi. Ad esempio:

Ogni sede deve avere almeno due programmatori

```
... -- quando si crea la TABLE Sedi
CHECK (2 <= (SELECT COUNT(*) FROM Imp I
            WHERE I.Sede = Sede -- correlazione
            AND I.Ruolo = 'Programmatore'))
```

Supponendo di avere due tabelle **ImpBO** e **ImpMI** e di volere che uno stesso **codice (CodImp)** non sia presente in entrambe le tabelle:

```
... -- quando si crea la TABLE ImpBO
CHECK (NOT EXISTS (SELECT * FROM ImpMI
                  WHERE ImpMI.CodImp = CodImp))
```

Il linguaggio SQL: query innestate



17



Sommario:

- Oltre alla forma "flat", in SQL è possibile fare uso di **subquery**.
- Una **subquery** che restituisca al massimo un valore è detta **scalare**, e per essa si possono usare i soliti operatori di confronto.
- Le forme **<op> ANY** e **<op> ALL** si rendono necessarie quando la **subquery** può restituire più valori.
- Il quantificatore esistenziale **EXISTS** è soddisfatto quando il risultato della subquery non è vuoto (e **NOT EXISTS** quando è vuoto).
- Una **subquery** si dice **correlata** se **referenzia variabili definite in un blocco ad essa più esterno**.
- In molti casi è possibile scrivere una query sia in forma piatta sia in forma innestata.

Il linguaggio SQL: query innestate



18