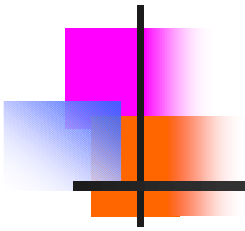
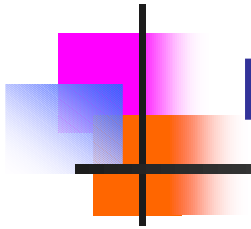


Il linguaggio SQL: query innestate

Sistemi Informativi L-A

Home Page del corso:
http://www-db.deis.unibo.it/courses/SIL-A/

Versione elettronica: SQLc-subquery.pdf



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

Sedi

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

Prog

CodProg	Citta
P01	Milano
P01	Bologna
P02	Bologna

Subquery

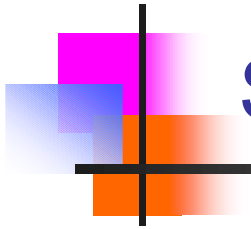
- Oltre alla forma “flat” vista sinora, in SQL è anche possibile esprimere delle condizioni che si basano sul risultato di altre interrogazioni (subquery, o query innestate o query nidificate)

```
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')
```



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)
```

- 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')
```

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

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')
```

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

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

Subquery: “unnesting” (1)

- È spesso possibile ricondursi a una forma “piatta”, ma la cosa non è sempre così ovvia. Ad esempio, nell’esempio 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

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, ma nella forma piatta:

```
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
```

- 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
```

perché la clausola WHERE non è mai soddisfatta!

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))
```

- Il blocco più interno viene 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

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
SET    Stipendio = 1.1*Stipendio
WHERE  Sede IN (SELECT S.Sede
                FROM   Sede S, Prog P
                WHERE  S.Citta = P.Citta
                  AND  P.CodProg = 'P02')
```

Subquery e CHECK

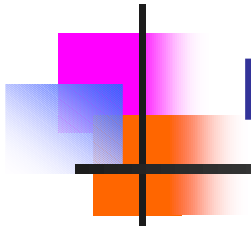
- Facendo uso di subquery nella clausola CHECK è possibile esprimere vincoli arbitrariamente complessi

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))
```



Riassumiamo:

- 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 che in forma innestata