Corso di Laurea in Ingegneria Gestionale
SAPIENZA Università di Roma
Esercitazioni del corso di Basi di Dati
Prof.ssa Catarci e Prof.ssa Scannapieco

Anno Accademico 2012/2013

5 – SQL : Interrogazioni nidificate

Francesco Leotta

Ultimo aggiornamento : 17/04/2013

Interrogazioni Nidificate

- In generale si è visto che l'argomento della clausola ***where*** si basa su condizioni composte da predicati semplici (tramite gli operatori logici ***not***, ***and*** e ***or***), in cui ciascun predicato rappresenta un semplice confronto fra due valori
- SQL ammette anche l'uso di predicati con una struttura più complessa, in cui **si confronta un valore** (ottenuto come risultato di un'espressione valutata sulla singola riga) **con il risultato dell'esecuzione di un'interrogazione SQL**, definita direttamente nel predicato interno alla clausola ***where***
- ***Si parla in questo caso di* INTERROGAZIONI NIDIFICATE**
- **ATTENZIONE :** Se in un predicato si confronta un attributo con il risultato di un'interrogazione, sorge il ***problema di disomogeneità*** dei termini del confronto. Infatti, *da una parte si ha il risultato di un'interrogazione SQL* (in generale un *insieme di valori*), mentre *dall'altra abbiamo il valore di un attributo per la particolare riga*

Interrogazioni Nidificate

- Tale problema viene risolto da SQL tramite l'utilizzo di alcune parole chiave (***all***, ***any***, ***in***, ***not in***, ***exists***, ***not exists***) che estendono i normali operatori di confronto relazionale (=, <>, <, >, <=, >=)
- La parola chiave ***any*** specifica che la riga soddisfa la condizione se risulta vero il confronto (con l'operatore specificato) tra il valore dell'attributo per la riga ed **almeno uno** degli elementi restituiti dall'interrogazione nidificata
- La parola chiave ***all*** specifica che la riga soddisfa la condizione solo se **tutti gli elementi** restituiti dall'interrogazione nidificata rendono vero il confronto
- Ovviamente, la sintassi richiede la ***compatibilità di dominio*** tra l'attributo restituito dall'interrogazione nidificata e l'attributo con cui avviene il confronto

La parola chiave *any*

- ***ESEMPIO* : *Estrarre gli impiegati che lavorano in dipartimenti situati a Firenze***

Impiegato

Nome	Cognome	Dipart	StipAnn

Dipartimento

Nome	Città

Interrogazione Esterna

```
SELECT *
FROM Impiegato
WHERE Dipart = any (SELECT Nome
                    FROM Dipartimento
                    WHERE Città = 'Firenze')
```

Interrogazione nidificata (o Interna)

la parola chiave "***in***" coincide con "**= *any***"

L'utilizzo dell'una o dell'altra parola chiave porta allo stesso risultato

- *L'interrogazione seleziona le righe di* **Impiegato** *per cui il valore dell'attributo* ***Dipart*** *è uguale ad almeno uno dei valori dell'attributo* ***Nome*** *delle righe di Dipartimento*

Interrogazioni nidificate VS JOIN

- ***ESEMPIO* : *Estrarre gli impiegati che lavorano in dipartimenti situati a Firenze***

Impiegato

Nome	Cognome	Dipart	StipAnn

Dipartimento

Nome	Città

```
SELECT *
FROM Impiegato, Dipartimento D
WHERE Dipart = D.Nome AND D.Città = 'Firenze'
```

equivale a

```
SELECT *
FROM Impiegato JOIN Dipartimento D  ON  Dipart = D.Nome
WHERE D.Città = 'Firenze'
```

- L'interrogazione nidificata della slide precedente può essere quindi anche espressa mediante un **JOIN** tra le tabelle **Impiegato** e **Dipartimento**.
  - La scelta dell'una o dell'altra formulazione è dettata dal grado di leggibilità della soluzione.

La parola chiave *all*

- ***ESEMPIO* : *Estrarre i dipartimenti in cui non lavorano persone di cognome "Rossi"***

Impiegato

Nome	Cognome	Dipart	StipAnn

Dipartimento

Nome	Città

```
SELECT Nome
FROM Dipartimento
WHERE Nome <> all (SELECT Dipart
                   FROM Impiegato
                   WHERE Cognome = 'Rossi')
```

la parola chiave "***not in***" coincide con "<> ***all***"

L'utilizzo dell'una o dell'altra parola chiave porta allo stesso risultato

- *L'interrogazione nidificata seleziona i valori dell'attributo* ***Dipart*** di tutte *le righe in cui il* ***Cognome*** vale "Rossi". *La condizione è soddisfatta da quelle righe di* **Dipartimento** *per cui il valore dell'attributo* ***Nome*** *non fa parte dei nomi prodotti dall'interrogazione nidificata*

Interpretazione semplice

- Un'interpretazione molto semplice delle interrogazioni nidificate consiste nell'assumere che l'interrogazione nidificata (o *interna*) **venga eseguita prima** di analizzare le righe dell'interrogazione *esterna*
- Si può ipotizzare che il risultato dell'interrogazione nidificata venga salvato in una tabella temporanea; il controllo sulle righe dell'interrogazione esterna può essere fatto accedendo direttamente al risultato temporaneo
- Questa interpretazione (detta **semplice**), in cui l'interrogazione nidificata viene eseguita ***una sola volta*** , *è* ***corretta*** nel caso in cui *le variabili di range definite nell'interrogazione più esterna* ***non vengano utilizzate*** *nell'ambito dell'interrogazione più interna*
- Consideriamo la di nuovo la precedente interrogazione...

Interpretazione semplice

- ***ESEMPIO* : *Estrarre i dipartimenti in cui non lavorano persone di cognome "Rossi"***

Impiegato

Nome	Cognome	Dipart	StipAnn

Dipartimento

Nome	Città

```
SELECT Nome
FROM Dipartimento
WHERE Nome <> all (SELECT Dipart
                   FROM Impiegato
                   WHERE Cognome = 'Rossi')
```

1. Il sistema può eseguire dapprima l'interrogazione nidificata che estrae il valore dell'attributo ***Dipart*** per tutti gli impiegati di cognome "Rossi"

2. A questo punto, per ciascun dipartimento, si controlla che il valore dell'attributo ***Nome*** non sia incluso nella tabella prodotta, utilizzando l'operatore **<> all**

Interrogazioni con correlazione

- Talvolta l'interrogazione nidificata fa riferimento al contesto dell'interrogazione esterna che la racchiude; tipicamente ciò accade tramite una variabile di range definita nell'interrogazione più esterna ed usata nell'ambito della interrogazione nidificata
- Si parla in questo caso di ***Interrogazioni nidificate con correlazione***

ESEMPIO* : *Estrarre gli impiegati che hanno degli omonimi (stesso nome e cognome, ma diverso codice fiscale)

Impiegato

Nome	Cognome	CodFiscale

```
SELECT *
FROM Impiegato I
WHERE exists  (SELECT *
                FROM Impiegato I1
                WHERE I.Nome=I1.Nome   and
                        I.Cognome = I1.Cognome   and
                        I.CodFiscale <> I1.CodFiscale)
```

l'operatore ***exists*** ammette come parametro un'interrogazione nidificata e restituisce il valore **VERO** solo se l'interrogazione nidificata fornisce un risultato **non vuoto**

Interrogazioni con correlazione

- Nel caso di interrogazioni nidificate con correlazione, l'interpretazione *semplice* fornita precedentemente **non è più valida**
- In questo caso è necessario che l'interrogazione nidificata venga valutata separatamente per ogni riga prodotta nella valutazione dell' interrogazione esterna. La nuova interpretazione è la seguente:
  - per ogni riga esaminata nell'ambito dell'interrogazione esterna, si deve valutare l'interrogazione nidificata (che quindi, in questo caso, non può essere calcolata a priori, **ma deve essere ricalcolata per ogni riga dell'interrogazione esterna**)
  - tale processo può essere ripetuto un numero arbitrario di volte, pari al numero arbitrario di nidificazioni che possono essere utilizzate nell' interrogazione
- **ATTENZIONE :** Per quanto riguarda la ***visibilità*** delle variabili di range, vale la restrizione che **una variabile è usabile solo nell'ambito dell'interrogazione in cui è definita** o **nell'ambito di un'interrogazione nidificata (a qualsiasi livello) all'interno di essa**

Interrogazioni con correlazione

- Riprendiamo l'esempio precedente...

ESEMPIO* : *Estrarre gli impiegati che hanno degli omonimi (stesso nome e cognome, ma diverso codice fiscale)

Impiegato

Nome	Cognome	CodFiscale

```
SELECT *
FROM Impiegato I
WHERE exists  (SELECT *
               FROM Impiegato I1
               WHERE I.Nome=I1.Nome   and
                     I.Cognome = I1.Cognome   and
                     I.CodFiscale <> I1.CodFiscale)
```

Si può osservare che l'interrogazione nidificata utilizza una variabile di range definita nell' interrogazione più esterna. Perciò, in questo caso, per ogni riga esaminata nell'ambito dell' interrogazione esterna, si deve valutare l'interrogazione nidificata.
Nell'esempio vengono considerate una ad una le righe della variabile I; per ciascuna di queste righe, viene eseguita l'interrogazione nidificata che restituisce o meno l'insieme vuoto a seconda che vi siano o meno degli omonimi della persona

Interrogazioni Nidificate

- Vediamo un esempio con un'istanza della tabella Impiegato...

```
SELECT *
FROM Impiegato I
WHERE exists  (SELECT *
               FROM Impiegato I1
               WHERE I.Nome=I1.Nome   and
                     I.Cognome = I1.Cognome   and
                     I.CodFiscale <> I1.CodFiscale)
```

1. Si considera la prima riga di **I** e si verifica se in **I1** esiste una tupla con stessi valori di ***Nome***, ***Cognome*** e diverso valore di ***Codice Fiscale***

   Tale tupla non esiste, perciò l'interrogazione nidificata restituisce una relazione vuota e ***exists*** restituisce il valore **FALSO**

I

Nome	Cognome	CodFiscale
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014

I1

Nome	Cognome	CodFiscale
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014

Interrogazioni Nidificate

```
SELECT *
FROM Impiegato I
WHERE exists  (SELECT *
               FROM Impiegato I1
               WHERE I.Nome=I1.Nome   and
                     I.Cognome = I1.Cognome   and
                     I.CodFiscale <> I1.CodFiscale)
```

2. Si considera la seconda riga di **I** e si verifica se in **I1** esiste una tupla con stessi valori di ***Nome***, ***Cognome*** e diverso valore di ***Codice Fiscale***

   Tale tupla esiste, perciò l'interrogazione nidificata restituisce una relazione non vuota e ***exists*** restituisce il valore **TRUE**

   La tupla *<'Carlo','Bianchi','B013'>* farà parte del risultato dell'interrogazione esterna

I

Nome	Cognome	CodFiscale
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014

I1

Nome	Cognome	CodFiscale
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014

Interrogazioni Nidificate

```
SELECT *
FROM Impiegato I
WHERE exists  (SELECT *
               FROM Impiegato I1
               WHERE I.Nome=I1.Nome   and
                     I.Cognome = I1.Cognome   and
                     I.CodFiscale <> I1.CodFiscale)
```

3. Si considera la terza riga di **I** e si verifica se in **I1** esiste una tupla con stessi valori di ***Nome***, ***Cognome*** e diverso valore di ***Codice Fiscale***

   Tale tupla esiste, perciò l'interrogazione nidificata restituisce una relazione non vuota e ***exists*** restituisce il valore **TRUE**

   La tupla *<'Carlo','Bianchi','C014'>* farà parte del risultato dell'interrogazione esterna

I

Nome	Cognome	CodFiscale
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014

I1

Nome	Cognome	CodFiscale
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014

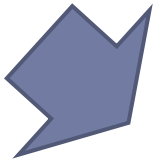
Interrogazioni Nidificate

ESEMPIO* : *Estrarre gli impiegati che hanno degli omonimi (stesso nome e cognome, ma diverso codice fiscale)

```
SELECT *
FROM Impiegato I
WHERE exists  (SELECT *
               FROM Impiegato I1
               WHERE I.Nome=I1.Nome    and
                     I.Cognome = I1.Cognome    and
                     I.CodFiscale <> I1.CodFiscale)
```

Impiegato

Nome	Cognome	CodFiscale
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014



Risultato finale dell'interrogazione

Nome	Cognome	CodFiscale
Carlo	Bianchi	B013
Carlo	Bianchi	C014

Interrogazioni Nidificate – Esempio 1

ESEMPIO* : *Estrarre gli impiegati che non hanno degli omonimi

Impiegato

Nome	Cognome	CodFiscale

```
SELECT *
FROM Impiegato I
WHERE not exists  (SELECT *
                   FROM Impiegato I1
                   WHERE I.Nome=I1.Nome    and
                   I.Cognome = I1.Cognome    and
                   I.CodFiscale <> I1.CodFiscale)
```

Interrogazioni Nidificate – Esempio 2

***ESEMPIO* : Estrarre i nomi degli impiegati che sono anche cognomi**

Impiegato

Nome	Cognome	CodFiscale

```
SELECT Nome
FROM Impiegato
INTERSECT
SELECT Cognome
FROM Impiegato
```

o, in alternativa

```
SELECT Nome
FROM Impiegato
WHERE Nome in (SELECT Cognome
               FROM Impiegato)
```

L'**Intersezione** insiemistica **non è supportata nativamente da MySQL**...ma è facilmente ottenibile tramite interrogazioni nidificate

Interrogazioni Nidificate – Esempio 3

ESEMPIO* : *Estrarre i nomi degli impiegati che non sono cognomi per qualche impiegato

Impiegato

Nome	Cognome	CodFiscale

```
SELECT Nome
FROM Impiegato
EXCEPT
SELECT Cognome
FROM Impiegato
```

o, in alternativa

```
SELECT Nome
FROM Impiegato
WHERE Nome not in (SELECT Cognome
                   FROM Impiegato)
```

La **Differenza** insiemistica **non è supportata nativamente da MySQL**...ma è facilmente ottenibile tramite interrogazioni nidificate

Interrogazioni Nidificate – Esempio 4

ATTENZIONE = Questa interrogazione è corretta?

```
SELECT Dipart
FROM Impiegato
WHERE Dipart in (SELECT  Nome
                 FROM Dipartimento D1
                 WHERE Nome = 'Produzione')  or
      Dipart in (SELECT  Nome
                 FROM Dipartimento D2
                 WHERE D1.Città = D2.Città)
```

NO → se un'interrogazione possiede sotto-interrogazioni annidate allo stesso livello, le variabili introdotte nella clausola ***FROM*** di una interrogazione **non potranno essere usate** nell'ambito di un'altra interrogazione allo stesso livello (mentre potranno essere usate in interrogazioni situate a livelli inferiori)

Interrogazioni Nidificate – Esempio 5

ESEMPIO* : *Date le relazioni* Cantante *ed* Autore *in figura, estrarre i cantautori, ovvero i cantanti che hanno eseguito alcune canzoni di cui erano anche autori

Cantante

Nome	Canzone

Autore

Nome	Canzone

```
SELECT Nome
FROM Cantante
WHERE Nome not in (SELECT Nome
                   FROM Cantante C
                   WHERE Nome not in (SELECT Nome
                                      FROM Autore
                                      WHERE Autore.Canzone = C.Canzone))
```

La prima interrogazione nidificata non ha alcun legame con l'interrogazione esterna e quindi può essere eseguita in modo del tutto indipendente

La seconda interrogazione nidificata presenta invece un legame con l'interrogazione esterna (*Autore.Canzone = C.Canzone)*

Interrogazioni Nidificate – Esempio 5

L'interrogazione relativa all'esempio precedente avviene seguendo queste fasi

1. L'interrogazione (***SELECT*** *Nome* ***FROM*** *Cantante C…*) legge tutte le righe della tabella **Cantante**
2. Per ognuna delle righe di C viene valutata l'interrogazione più interna (***SELECT*** *Nome* ***FROM*** *Autore…*) , che restituisce i nomi degli autori della canzone il cui titolo compare nella riga di C che viene considerata. Se il nome del cantante non compare tra gli autori (quindi non è un cantautore puro), allora il nome viene selezionato
3. Dopo che l'interrogazione nidificata ha terminato di analizzare le righe di C (costruendo la tabella contenente i nomi dei cantanti che non sono cantautori puri), viene eseguita l'interrogazione più esterna, la quale restituirà tutti i nomi di cantanti che non compaiono nella tabella ottenuta come risultato dell'interrogazione nidificata

Interrogazioni Nidificate – Esempio 5bis

ESEMPIO** (con soluzione alternativa)* **:** ***Date le relazioni* Cantante *ed* Autore *in figura, estrarre i cantautori puri, ovvero i cantanti che hanno eseguito solo canzoni di cui erano anche autori

Cantante

Nome	Canzone

Autore

Nome	Canzone

```
SELECT Nome
FROM Cantante
EXCEPT
SELECT Nome
FROM Cantante C
WHERE Nome not in (SELECT Nome
                   FROM Autore
                   WHERE Autore.Canzone = C.Canzone)
```

Espressione alternativa ottenuta utilizzando l'operatore insiemistico **EXCEPT**

Interrogazioni Nidificate – Esempio 6

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre i nomi ed i cognomi degli Impiegati che lavorano in Dipartimenti situati a Torino**

```
SELECT Nome, Cognome
FROM Impiegato
WHERE Dipart = any (SELECT Nome
                    FROM Dipartimento
                    WHERE Città = 'Torino')
```

Nome	Cognome
Carlo	Bianchi
Marco	Franco

Interrogazioni Nidificate – Esempio 7

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre il dipartimento dell'impiegato che guadagna lo stipendio massimo (usando l'operatore aggregato *max*)**

```
SELECT Dipart
FROM Impiegato
WHERE StipAnn = (SELECT max(StipAnn)
                 FROM Impiegato)
```

Dipart
Direzione

Interrogazioni Nidificate – Esempio 8

Impiegato

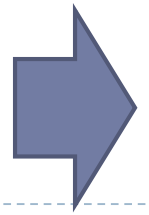
Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre il dipartimento dell'impiegato che guadagna lo stipendio massimo (senza l'operatore max)**

```
SELECT Dipart
FROM Impiegato
WHERE StipAnn >= all
                (SELECT StipAnn
                 FROM Impiegato)
```



Dipart
Direzione

Interrogazioni Nidificate – Esempio 9

ESERCIZIO **:Trovare il reddito complessivo dei figli di Gianni e Maria**

Persone

Nome	Reddito	Eta	Sesso

Genitori

Figlio	Genitore

```
SELECT sum(P.reddito) AS redditocompl
FROM Persone P
WHERE Nome =any(
        SELECT Figlio
        FROM Genitori
        WHERE Genitore='Gianni')
        AND
        Nome =any(SELECT Figlio
        FROM Genitori
        WHERE Genitore='Maria');
```

RedditoCompl
25

Esercizio

Negozi

ID	Nome	Città

Prodotti

Codice	Nome	Marca

Listino

Negozi	Prodotto	Prezzo

ESERCIZIO **:Trovare, per ciascun prodotto, la città in cui viene venduto al prezzo più basso**

```
SELECT distinct P.Nome, N.Citta
FROM Negozi N, Listino L, Prodotti P
WHERE N.ID=L.Negozi AND P.Codice=L.Prodotto AND
        prezzo <= ALL (
                SELECT prezzo
                FROM Prodotti P2, Listino L2
                WHERE P2.Codice=P.Codice AND
                        L2.Prodotto = P2.Codice
                )
```

Esercizio Proposto

ESERCIZIO* : *Siano date le seguenti relazioni

- Fornitori(**fid**:*integer*, **fnome**:*string*, **indirizzo**:*string*)
- Pezzi(**pid**:*integer*, **pnome**:*string*, **colore**:*string*)
- Catalogo(**fid**:*integer*, **pid**:*integer*, **costo**:*real*)

- Per le relazioni sussistono i seguenti vincoli di integrità:
- ***Fornitori.fid*** è **CHIAVE PRIMARIA** di *Fornitori*
- ***Pezzi.pid*** è **CHIAVE PRIMARIA** di *Pezzi*
- ***Catalogo.fid*** e ***Catalogo.pid*** sono **CHIAVE PRIMARIA** di *Catalogo*
- ***Catalogo.fid*** è **CHIAVE ESTERNA** verso *Fornitori.fid*
- ***Catalogo.pid*** è **CHIAVE ESTERNA** verso *Pezzi.pid*

Esercizio Proposto

ESERCIZIO* : *Siano date le seguenti relazioni

- Fornitori(**fid**:*string*, **fnome**:*string*, **indirizzo**:*string*)
- Pezzi(**pid**:*string*, **pnome**:*string*, **colore**:*string*)
- Catalogo(**fid**:*string*, **pid**:*string*, **costo**:*real*)

ESERCIZIO* : *Si calcolino le seguenti interrogazioni in SQL

- 1. Trovare i *pnome* dei pezzi per cui esiste un qualche fornitore
- 2. Trovare gli *fnome* dei fornitori che forniscono ogni pezzo
- 3. Trovare gli *fnome* dei fornitori che forniscono tutti i pezzi rossi
- 4. Trovare i *pnome* dei pezzi forniti dalla Acme e da nessun altro
- 5. Trovare i *fid* dei fornitori che ricaricano su alcuni pezzi più del costo medio di quel pezzo
- 6. Per ciascun pezzo, trovare gli *fnome* dei fornitori che ricaricano di più su quel pezzo

Esercizio Proposto

- 7. Trovare i *fid* dei fornitori che forniscono solo pezzi rossi
- 8. Trovare i *fid* dei fornitori che forniscono un pezzo rosso e un pezzo verde
- 9. Trovare i *fid* dei fornitori che forniscono un pezzo rosso o uno verde
- 10. Trovare i *pid* dei pezzi forniti da almeno due fornitori

Esercizio Proposto – 1\10

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

1) Trovare i pnome dei pezzi per cui esiste un qualche fornitore

```
SELECT distinct P.pnome
FROM Pezzi P, Catalogo C
WHERE P.pid = C.pid
```

Si effettua un *equi-join* tra il ***pid*** di **Catalogo** e il ***pid*** di **Pezzi**. Si ottengono così solo quei pezzi che hanno un fornitore.

Questo perché nella relazione **Pezzi** potrebbero esserci dei pezzi che non sono forniti da nessun fornitore

Esercizio Proposto – 2\10

Cfr. Es. Algebra Relazionale Esercizio 1.5

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

2) Trovare gli fnome dei fornitori che forniscono ogni pezzo

```
SELECT distinct F.fnome
FROM Fornitori F
WHERE NOT EXISTS (
                    (SELECT P.pid FROM Pezzi P)
                     EXCEPT
                    (SELECT C.Pid
                     FROM Catalogo C
                     WHERE C.fid = F.fid)
                  )
```

L'interrogazione interna va valutata e ri-calcolata per ogni tupla di **Fornitori** definita nell'interrogazione esterna

A EXCEPT B
- Viene restituita una relazione che contiene tutti i ***pid*** dei pezzi **non forniti** dal fornitore i-esimo. Se tale relazione è **vuota**, significa che il fornitore i-esimo **fornisce tutti i pezzi.**
In quest'ultimo caso, **NOT EXISTS** restituisce **TRUE** e l'***fnome*** dell'i-esimo fornitore considerato apparirà nella relazione risultato

***A.* (SELECT** P.Pid **FROM** Pezzi P**)**
- Calcola i ***pid*** di tutti i pezzi

***B.* (SELECT** C.Pid **FROM** Catalogo C **WHERE** C.fid=F.fid**)**
-Calcola i ***pid*** dei pezzi venduti dal fornitore i-esimo (che stiamo considerando attraverso la variabile di range F definita nell'interrogazione esterna)

Esercizio Proposto – 3\10

Cfr. Es. Algebra Relazionale Esercizio 1.6

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

3) Trovare gli fnome dei fornitori che forniscono tutti i pezzi rossi

```
SELECT distinct F.fnome
FROM Fornitori F
WHERE NOT EXISTS (
        (SELECT * FROM Pezzi P
         WHERE P.colore = 'Rosso')
        EXCEPT
        (SELECT P1.Pid, P1.colore,
                P1.pnome
         FROM Catalogo C, Pezzi P1
         WHERE C.fid = F.fid  and
               C.pid = P1.pid)
)
```

L'interrogazione interna va valutata e ri-calcolata per ogni tupla di **Fornitori** definita nell'interrogazione esterna

A EXCEPT B

- Viene restituita una relazione che contiene tutti quei pezzi di colore rosso **non forniti** dal fornitore i-esimo. Se tale relazione è **vuota**, significa che il fornitore i-esimo **fornisce tutti i pezzi esistenti di colore rosso** (ciò non vuol dire che il fornitore i-esimo non possa fornire pezzi anche di altro colore...l'importante è che fornisca almeno tutti quelli di colore rosso)

In quest'ultimo caso, **NOT EXISTS** restituisce **TRUE** e l'***fnome*** dell'i-esimo fornitore considerato apparirà nella relazione risultato

***A.* (SELECT * FROM Pezzi P)**

- Calcola tutti i pezzi di colore rosso

***B.* (SELECT P1.Pid, P1.colore, P1.pnome FROM Catalogo C, Pezzi P1 WHERE C.fid=F.fid and C.pid = P1.pid)**

- Calcola i pezzi venduti dal fornitore i-esimo (che stiamo considerando attraverso la variabile di range F definita nell'interrogazione esterna)

Esercizio Proposto – 4\10

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

4) Trovare i pezzi forniti dalla ACME e da nessun altra

```
SELECT P.pid, P.pnome, P.colore
FROM Pezzi P, Catalogo C, Fornitori F
WHERE P.pid=C.pid and C.fid=F.fid and F.fnome = 'ACME'  and
        NOT EXISTS (SELECT * FROM  Catalogo C1, Fornitori F1
                    WHERE P.pid = C1.pid and C1.fid=F1.fid and
                     F1.fnome <> 'ACME'
                    )
```

Se la relazione restituita dall'interrogazione interna è **vuota**, **NOT EXISTS** restituisce **TRUE**.
A questo punto si verifica che il pezzo i-esimo sia almeno fornito dalla 'Acme'...in quest'ultimo caso, il pezzo considerato apparirà nella relazione risultato

L'interrogazione interna calcola tutti quei fornitori (il cui nome è diverso da 'Acme') che forniscono il pezzo i-esimo (che stiamo considerando attraverso la variabile di range P definita nell'interrogazione esterna).

Se tale relazione è **vuota**, significa che **il pezzo i-esimo non è fornito da alcun fornitore il cui nome è diverso da 'Acme'**

Esercizio Proposto – 5\10

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

5) Trovare i fid dei fornitori che ricaricano su alcuni pezzi più del costo medio di quel pezzo

```
SELECT distinct C.fid
FROM Catalogo C
WHERE C.costo > (SELECT AVG(C1.costo)
                 FROM  Catalogo C1
                 WHERE C1.pid = C.pid
                 )
```

L'interrogazione interna calcola la media del prezzo di vendita del pezzo i-esimo considerato (*C.pid*).

L'interrogazione esterna, per ciascun pezzo fornito in catalogo, **verifica che il prezzo del pezzo considerato sia superiore rispetto al prezzo medio di vendita di quel pezzo** (calcolato con l'interrogazione interna).

In quest'ultimo caso, l'interrogazione esterna restituisce il ***fid*** del fornitore che vende il pezzo i-esimo ad un prezzo superiore alla media

Esercizio Proposto – 6\10

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

6) Per ciascun pezzo, trovare gli fnome dei fornitori che ricaricano di più su quel pezzo

```
SELECT C.pid, F.fnome
FROM Fornitori F, Catalogo C
WHERE C.fid = F.fid
        and C.costo = (SELECT MAX(C1.costo)
                       FROM  Catalogo C1
                       WHERE C1.pid = C.pid
                      )
```

L'interrogazione interna calcola il massimo prezzo di vendita del pezzo i-esimo considerato (*C.pid*).

L'interrogazione esterna, per ciascun pezzo fornito in catalogo, **verifica che il prezzo del pezzo i-esimo considerato sia uguale al massimo prezzo di vendita di quel pezzo** (calcolato con l'interrogazione interna). In quest'ultimo caso, l'interrogazione esterna restituisce il nome del fornitore (e il pid del pezzo-iesimo) che vende il pezzo i-esimo al prezzo massimo

Esercizio Proposto – 7\10

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

7) Trovare i fid dei fornitori che forniscono solo pezzi rossi

```
SELECT F.fid
FROM Fornitori F
WHERE NOT EXISTS (SELECT *
                  FROM  Pezzi P, Catalogo C1
                  WHERE C1.fid = F.fid and
                        C1.pid = P.pid and
                        P.colore <> 'Rosso'
                  )
      and EXISTS (SELECT *
                  FROM Catalogo C1
                  WHERE C1.fid = F.fid)
```

La prima interrogazione interna verifica che l'i-esimo fornitore venda pezzi **non rossi**

La seconda interrogazione esterna verifica che l'i-esimo fornitore venda qualcosa (per evitare che eventuali fornitori che non vendono niente siano inclusi nel risultato)

L'interrogazione esterna inserisce il nome dell'i-esimo fornitore nella relazione risultato se la prima interrogazione interna restituisce una relazione vuota e la seconda interrogazione esterna restituisce una relazione non vuota

Esercizio Proposto – 8\10

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

8) Trovare i fid dei fornitori che forniscono qualche pezzo rosso e qualche pezzo verde

```
SELECT distinct C.fid
FROM Catalogo C, Pezzi P
WHERE C.pid = P.pid and P.colore = 'rosso'
INTERSECT
SELECT distinct C1.fid
FROM  Catalogo C1, Pezzi P1
WHERE C1.pid = P1.pid and P1.colore = 'verde'
```

La prima interrogazione calcola i ***fid*** dei fornitori (senza duplicati) che forniscono almeno un pezzo rosso

La seconda interrogazione calcola i ***fid*** dei fornitori (senza duplicati) che forniscono almeno un pezzo verde

L'**INTERSEZIONE** restituisce **i *fid* dei fornitori che forniscono *almeno* un pezzo rosso e un pezzo verde**

Esercizio Proposto – 9\10

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

9) Trovare i fid dei fornitori che forniscono qualche pezzo rosso o qualche verde

```
SELECT distinct C.fid
FROM Catalogo C, Pezzi P
WHERE C.pid = P.pid and P.colore = 'rosso'
UNION
SELECT distinct C1.fid
FROM  Catalogo C1, Pezzi P1
WHERE C1.pid = P1.pid and P1.colore = 'verde'
```

La prima interrogazione calcola i ***fid*** dei fornitori (senza duplicati) che forniscono almeno un pezzo rosso

La seconda interrogazione calcola i ***fid*** dei fornitori (senza duplicati) che forniscono almeno un pezzo verde

L'**UNIONE** restituisce **i *fid* dei fornitori che forniscono *almeno* un pezzo rosso oppure *almeno* un pezzo verde**

Esercizio Proposto – 10\10

Cfr. Es. Algebra Relazionale Esercizio 1.10

Fornitori

fid	fnome	indirizzo

Pezzi

pid	pnome	colore

Catalogo

fid	pid	costo

10) Trovare i pid dei pezzi forniti da almeno due fornitori

```
SELECT distinct C.pid
FROM Catalogo C
WHERE EXISTS (SELECT distinct C1.pid
                FROM  Catalogo C1
                WHERE C1.pid = C.pid and
                      C1.fid <> C.fid)
```

L' i-esimo pezzo compare nel risultato dell'interrogazione interna se esiste un altro fornitore che vende quel pezzo