Basi Di Dati e di conoscenza

SQl- Interrogazioni

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Unione, intersezione e differenza
- Operatori aggregati
- Interrogazioni nidificate

Interrogazioni

- Le interrogazioni in SQL sono formulate in modo **dichiarativo** specificando cioè cosa si vuole ottenere e non come lo si vuole ottenere.

- L'interrogazione viene passata **all'ottimizzatore di interrogazioni** (query optimizer) che fa parte del DBMS. Questo la analizza e la traduce nel linguaggio di interrogazione interno al DBMS.

Interrogazioni

L'istruzione base per le interrogazioni è **select**

select *ListaAttributi* (*target list*)
from *ListaTabelle* (*clausola* **from**)
[**where** *Condizione*] (*clausola* **where**)

Più in dettaglio:

select *AttrEspr* [[**as**] *Alias*]{**,** *AttrEspr* [[**as**] *Alias*]}
from *Tabella* [[**as**] *Alias*]{**,** *Tabella* [[**as**] *Alias*]}
[**where** *Condizione* **]**

Seleziona le righe che soddisfano la condizione **where** fra quelle appartenenti al prodotto cartesiano delle tabelle in *ListaTabelle*.
Ogni colonna (tabella) può essere ridenominata con un alias.

Interrogazioni: esempio

Data una base di dati che contiene le tabelle:

```
IMPIEGATO(Nome, Cognome, Dipart, Ufficio, Stipendio, Città)
DIPARTIMENTO(Nome, Indirizzo,Città)
```

```
select Stipendio/12  as SalarioMensile
from     Impiegato
where Cognome = `Rossi`
```

Il risultato è una tabella con una colonna rinominata **SalarioMensile** e tante righe quanti sono gli impiegati che si chiamano Rossi.

Se si usa * dopo **select** si selezionano tutti gli attributi

Maternità

Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternità

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Persone

Madre	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

SQL-Select

- L'algebra relazionale non ammette duplicati, SQL li ammette. Quindi:

```
select Città
from Persona
```

estrae una lista di città in cui una città può comparire più volte.

Per i soli valori distinti:

```
select distinct nomicolonne
from tabella
where condizioni;
```

Esempio:

```
select distinct Città
from Persona;
```

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Unione, intersezione e differenza
- Operatori aggregati
- Interrogazioni nidificate

Selezione e Proiezione

Nome e reddito delle persone con meno di trenta anni

$$\mathbf{PROJ_{Nome,\ Reddito}(SEL_{Eta<30}(Persone))}$$

```
select Nome, Reddito
from Persone
where Eta < 30
```

Proiezione

Selezione

Selezione, senza proiezione

Nome, età e reddito delle persone con meno di trenta anni

$$\mathbf{SEL_{Eta<30}(Persone)}$$

```
select *   (* equivale a selezionare tutti gli attributi)
from Persone
where Eta < 30;
```

Proiezione, senza selezione

Nome e reddito di tutte le persone

$$\mathbf{PROJ}_{\mathbf{Nome,\ Reddito}}\mathbf{(Persone)}$$

```
select Nome, Reddito
from Persone;
```

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Unione, intersezione e differenza
- Operatori aggregati
- Interrogazioni nidificate

SELECT: abbreviazioni

- Selezionare tutti gli attributi

```
select *
from Persone
where Eta < 30

select Nome, Eta, Reddito
from Persone
where Eta < 30
```

Alias

- Assegna un alias alla colonna

```
select nomecolonna aliascolonna
from tabella;
```

- Assegna un alias alla tabella

```
select aliastabella.nomecolonna aliascolonna
from tabella as aliastabella;
```

ESEMPIO

```
select emp_id codice
from emp;

select p.nome as nome, p.reddito as reddito
from persone p
where p.eta < 30
```

SELECT: abbreviazioni

R(A,B)

```
select  *
from R
```

equivale (intuitivamente) a

```
select X.A as A, X.B as B
from R X
where true
```

Espressioni nella target list

```
select Reddito/2 as RedditoSemestrale
from Persone
where Nome = 'Luigi'
```

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Unione, intersezione e differenza
- Operatori aggregati
- Interrogazioni nidificate

Clausola **`where`**

- Operatori per la clausola **`where`**:

  `=` equivalenza

  `<>` diverso

  `!=` diverso

  `<` minore

  `>` maggiore

  `>=` maggiore o uguale

  `<=` minore o uguale

Clausola **`where`**

- Operatori per la clausola **`where`**:

  `between` … **`and`** … compreso tra … e …
  `in` appartanente a
  `like` del tipo
  `IS NULL` è un valore nullo / assente
  `IS NOT NULL` è un valore non nullo
  `NOT` negazione
  `AND` congiunzione
  `OR` disgiunzione

Clausola **where**

- Per i confronti fra stringhe è definito l'operatore `like`

- Il confronto è effettuato con una stringa che può contenere i caratteri speciali `%` e `_` .

  `_` rappresenta un carattere arbitrario

  % rappresenta un numero arbitrario di caratteri (anche zero).

- **Esempio**

```
select *
from Impiegato
where Cognome like '_o%i';  (Rossi, Rosi, Porti… etc)
```

Clausola **where**

Esempio `select` con `where`

- Confronto

```
select * from emp where job = 'clerck';
```

- Appartenenza

```
select ename, job, sal
from emp
where sal between 1200 and 5000;
```

- Somiglianza ortografica

```
select autore, qualific
from au
where autore like 'A%';
```

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Unione, intersezione e differenza
- Operatori aggregati
- Interrogazioni nidificate

Gestione dei valori nulli

Impiegati	Matricola	Cognome	Filiale	Età
	5555	Rossi	Roma	45
	6666	Neri	Roma	NULL

- Gli impiegati la cui età è o potrebbe essere maggiore di 40

$$SEL_{(Età>40)\ OR\ (Età\ IS\ NULL)}\ (IMPIEGATI)$$

```
select *
from Impiegati
where Eta > 40 or Eta is null;
```

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Unione, intersezione e differenza
- Operatori aggregati
- Interrogazioni nidificate

Selezione, proiezione e join

- Istruzioni SELECT con una sola relazione nella clausola FROM permettono di realizzare:
  - selezioni, proiezioni, ridenominazioni
- con più relazioni nella FROM si realizzano join (e prodotti cartesiani)

SQL e algebra relazionale

- **R1(A1,A2) R2(A3,A4)**

```
select distinct R1.A1, R2.A4
from   R1, R2
where  R1.A2 = R2.A3
```

- prodotto cartesiano (**FROM**)
- selezione (**WHERE**)
- proiezione (**SELECT**)

$$\mathbf{PROJ_{A1,A4}\ (SEL_{A2=A3}\ (R1\ JOIN\ R2))}$$

SQL - JOIN

- Se si vogliono estrarre informazioni da più tabelle, si pone come argomento della clausola from una lista delle tabelle.
- Se si deve formulare un join, è possibile farlo esplicitando il collegamento fra le due tabelle nella clausola **`where`**.

Esempio Estrarre i nomi degli impiegati e le città dove lavorano.

```
select Impiegato.Nome, Impiegato.Cognome,
   Dipartimento.Città
from Impiegato, Dipartimento
where Impiegato.Dipart = Dipartimento.Nome;
```

SQL JOIN

- Sintassi per il Join e il Join esterno:

```
select AttrEspr [[as] Alias]{, AttrEspr [[as] Alias]}
from Tabella        [[as] Alias]
{[ TipoJoin ] join Tabella [[as] Alias]on CondizioneJoin}
[ where AltraCondizione ]
```

TipoJoin può assumere i valori:
inner (default), *right [outer], left [outer], full [outer*]

Maternità

Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternità

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Selezione, proiezione e join

- I padri di persone che guadagnano più di 20

$$\mathbf{PROJ_{Padre}(paternita\ JOIN_{Figlio=Nome}\ SEL_{Reddito>20}(persone))}$$

```
select distinct Padre
   from Persone, Paternita
   where Figlio = Nome and Reddito > 20;
```

Selezione, proiezione e join

- Le persone che guadagnano più dei rispettivi padri; mostrare nome, reddito e reddito del padre

$$\text{PROJ}_{\text{Nome, Reddito, RP}}\ (\text{SEL}_{\text{Reddito>RP}}\ (\text{REN}_{\text{NP,EP,RP} \leftarrow \text{Nome,Eta,Reddito}}(\text{persone})\ \text{JOIN}_{\text{NP=Padre}}\ (\text{paternita JOIN}_{\text{Figlio =Nome}}\ \text{persone})))$$

```
select f.Nome, f.Reddito, p.Reddito
          from Persone p, Paternita, Persone f
          where p.Nome = Padre and
                Figlio = f.Nome and
                f.Reddito > p.Reddito
```

SQL – JOIN esplicito

Esempio Padre e madre di ogni persona

```
select paternita.figlio, padre, madre
from maternita, paternita
where paternita.figlio = maternita.figlio;
```

```
select madre, paternita.figlio, padre
  from maternita join paternita on
paternita.figlio = maternita.figlio;
```

Join su più tabelle

- Le persone che guadagnano più dei rispettivi padri; mostrare nome, reddito e reddito del padre

```
select f.Nome, f.Reddito, p.Reddito
     from (Persone p join Paternita on p.Nome = Padre)
          join Persone f on Figlio = f.Nome
     where f.Reddito > p.Reddito


     select f.Nome, f.Reddito, p.Reddito
     from Persone p, Paternita, Persone f
     where p.Nome = Padre and
          Figlio = f.Nome and
          f.Reddito > p.Reddito
```

Join esterno

- Padre e, se nota, madre di ogni persona

```
select Paternita.Figlio, Padre, Madre
from Paternita left join Maternita
        on Paternita.Figlio = Maternita.Figlio
```

```
select Paternita.Figlio, Padre, Madre
from Paternita left outer join Maternita
        on Paternita.Figlio = Maternita.Figlio
```

- **outer** è opzionale

Outer Join

```
select Paternita.Figlio, Padre, Madre
from Maternita join Paternita
  on Maternita.Figlio = Paternita.Figlio

select Paternita.Figlio, Padre, Madre
from Maternita left outer join Paternita
  on Maternita.Figlio = Paternita.Figlio

select Paternita.Figlio, Padre, Madre
from Maternita full outer join Paternita
  on Maternita.Figlio = Paternita.Figlio
```

Che cosa produce ?

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Operatori aggregati
- Unione, intersezione e differenza
- Interrogazioni nidificate

Ordinamento del risultato

- Nome e reddito delle persone con meno di trenta anni in ordine alfabetico

```
select Nome, Reddito
       from Persone
       where Eta < 30
```

```
select Nome, Reddito
              from Persone
              where Eta < 30
              order by Nome asc
```

Persone

Nome	Reddito
Andrea	21
Aldo	15
Filippo	30

Persone

Nome	Reddito
Aldo	15
Andrea	21
Filippo	30

asc e **desc** sono le keyword da utilizzare per la tipologia di ordinamento

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Operatori aggregati
- Unione, intersezione e differenza
- Interrogazioni nidificate

Unione, intersezione e differenza

- La `select` da sola non permette di fare **unioni**; serve un costrutto esplicito:

```
select  …
union  [all]
select ...
```

  - **Esempio**

```
select *
  from A
  union
  select *
  from B
```

- i duplicati vengono eliminati (a meno che si usi `all`) anche dalle proiezioni!

Differenza

- La differenza viene fatta con l'operatore **except**
- Per due blocchi di query A e B, restituisci tutti i risultati da A che non sono presenti anche in B, omettendo eventuali duplicati.
  - Le due tabelle devono avere gli stessi nomi di attributo e domini comparabili

```
select *
  from A
  except
  select *
  from B
```

```
select Nome
from Impiegato
except
select Cognome as Nome
from Impiegato
```

- Si può esprimere anche con le query nidificate (vedremo in seguito in dettaglio)

```
select * from A
where NOT EXISTS (
    select * from B
    where A.column_name = B.column_name    (column_name è chiave)
```

- Alcuni DBMS (ORACLE) utilizzano l'operatore **MINUS**

Intersezione

```
select Nome
  from Impiegato
  intersect
  select Cognome as Nome
  from Impiegato
```

- equivale a

```
select I.Nome
  from Impiegato I, Impiegato J
  where I.Nome = J.Cognome
```

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Operatori aggregati
- Unione, intersezione e differenza
- Interrogazioni nidificate

Operatori aggergati

- In algebra relazionale le espressioni vengono valutate sulle singole tuple in successione.
- Talvolta però possono essere necessarie informazioni derivabili dall'esame di tutte le tuple o di più tuple contemporanemente.
- SQL prevede una serie di operatori aggregati:
  - **`avg`** calcola la media dei valori per la colonna
  - **`count`** calcola il conteggio di righe per la colonna
  - **`max`** calcola il massimo per la colonna
  - **`min`** calcola il minimo per la colonna
  - **`sum`** calcola la somma dei valori per la colonna

Operatori aggergati

- **Sintassi:**

count (< ***** | [**distinct | all**] *ListaAttributi* >)
< **sum|max|min|avg** >([**distinct | all**] *AttrEspr*)

- **Esempio:** Determinare il numero degli impiegati che si chiamano Rossi

```
select count(*)
from Impiegato
where nome= 'Rossi';
```

- **Esempio** Calcola il massimo salario, il minimo salario e la differenza tra massimo e minimo

```
select max(sal), min(sal), max(sal)-min(sal)
from emp;
```

Operatori aggregati: `group by`

- Gli operatori aggregati vengono applicati a tutte le righe che vengono prodotte come risultato dell'operazione.
- Può essere necessario applicare l'operatore solo ad un **sottoinsieme** delle righe.
- SQL non ammette che nella stessa target list compaiano funzioni aggregate ed espressioni a livello di riga, come il nome di un attributo.
- L'operatore **`group by`** specifica come suddividere le tabelle in sottoinsiemi.

- **Esempio**: Calcolare lo stipendio massimo per dipartimento

```
select Dipart, max(Stipendio)
from Impiegato
group by Dipart
```

`group by`: Esempio

- Il numero di figli di ciascun padre

```
select padre, count(*)  AS NumFigli
from paternita
group by padre
```

Paternita

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Padre	NumFigli
Sergio	1
Luigi	2
Franco	2

Raggruppamenti e target list

- scorretta

```
select padre, avg(f.reddito), p.reddito
from persone f join paternita on figlio = nome join
     persone p on padre = p.nome
group by padre
```

- corretta

```
select padre, avg(f.reddito), p.reddito
from persone f join paternita on figlio = nome join
     persone p on padre = p.nome
group by padre, p.reddito
```

`group by` e valori nulli

A	B
1	11
2	11
3	null
4	null

```
select B, count (*)
from R group by B
```

B	
11	2
null	2

```
select A, count (*)
from R group by A
```

A	
1	1
2	1
3	1
4	1

```
select A, count (B)
from R group by A
```

A	
1	1
2	1
3	0
4	0

Raggruppamenti e clausala **having**

- Può essere anche necessario restringere i gruppi attraverso l'applicazione di condizioni.
- Se le condizioni sono verificabili a livello delle singole righe, basta utilizzare la clausola **where**, altrimenti si aggiunge una condizione alla **group by** attraverso la clausola **having**

```
select Dipart, sum(Stipendio) as SommaStipendi
from Impiegati
group by Dipart
having sum(Stipendio) > 100
```

- Se non si specifica group by e si usa **having** da solo la condizione è applicata a tutte le righe. Il problema è che se la condizione non è verificata, il risultato sarà vuoto

Raggruppamenti e clausala **having**

- I padri i cui figli hanno un reddito medio maggiore di 25

```
select padre, avg(f.reddito)
      from persone f join paternita on
                           figlio = nome
      group by padre
      having avg(f.reddito) > 25;
```

Forma estesa della **select**

- La forma di select cui siamo arrivati dopo le estensioni viste è quindi:

SelectSQL **::= select** *ListaAttributiOEspressioni*
from *ListaTabelle*
[**where** *CondizioniSemplici*]
[**group by** *ListaAttributiDiRaggruppamento*]
[**having** *CondizioniAggregate*]
[**order by** *ListaAttributiDiOrdinamento*]

where o having?

- I padri i cui figli sotto i 30 anni hanno un reddito medio maggiore di 25

```
select padre, avg(f.reddito)
from persone f join paternita on
                          figlio = nome
where eta < 30
group by padre
having avg(f.reddito) > 25
```

Contenuti della lezione

- Interrogazioni
- Selezioni e proiezioni
- Alias e abbreviazioni
- Clusola **`where`**
- Gestione dei valori nulli
- Join
- Ordinamento
- Operatori aggregati
- Unione, intersezione e differenza
- Interrogazioni nidificate

Interrogazioni nidificate

- E' possibile anche realizzare clausole **`where`** in cui il confronto non avviene fra predicati semplici o fra valori, ma fra valori e risultati di interrogazioni.
- Il risultato dell'interrogazione è un **attributo** o una **lista di attributi**.
- **Problema** di confrontare un valore con un insieme di valori (il risultato della interrogazione).
  - Gli operatori di confronto vengono estesi con i quantificatori **esistenziale** e **universale**:
    - **`all`** il confronto è **vero** se è **vero** per **tutte** le righe del risultato dell'interrogazione.
    - **`any`** il confronto è **vero** se è **vero** per **almeno una** delle righe del risultato dell'interrogazione.

Interrogazioni nidificate

Una **select** **nidificata** è composta da una **select** esterna e una o più **select** interne nella clausola **where**

Esempio

```
select *
from Persone
where Età > (select MAX(Età) from Persone where Residenza = 'Roma')

select ename
from emp
where deptno = (select deptno from emp where ename ='Allen');
```

In alternativa…

```
select x.ename
from emp x, emp y
where x.deptno=y.deptno and y.ename ='Allen';
```

Interrogazioni nidificate

- Nome e reddito del padre di Franco

```
select Nome, Reddito
from Persone, Paternita
where Nome = Padre and Figlio = 'Franco';
```

```
select Nome, Reddito
from Persone
where Nome = ( select Padre
                 from Paternita
                      where Figlio = 'Franco');
```

Interrogazioni nidificate

- L'uso delle interrogazioni nidificate può anche eliminare la necessità degli alias.

```
select I1.Nome
from Impiegato I1, Impiegato I2
where I1.Nome = I2. Nome and
  I1.Dipart ='Produzione' and
              I2.Dipart <> 'Produzione'
```

- equivale a

```
select Nome
from Impiegato
where Nome = any (select Nome
                  from Impiegato
            where Dipart = 'Produzione')
                and Dipart <> 'Produzione'
```

Interrogazioni nidificate

```
select *
    from Impiegato
    where Dipart = any (select Nome
                        from Dipartimento
                        where Citta='Firenze')
```

- Il risultato è una tabella che comprende tutte le righe di **Impiegato** per cui il valore **Dipart** è uguale ad almeno uno dei valori di Nome in **Dipartimento**, limitatamente alle tuple per cui Citta='Firenze'.
- Lo stesso risultato si poteva ottenere con un join, ma così, specialmente per interrogazioni complesse, è più leggibile.

Interrogazioni Nidificate

- Non tutte le interrogazioni nidificate corrispondono però ad un join.

```
select Nome
from Dipartimento
where Nome <> all ( select Dipart
                    from Impiegato
                    where Cognome='Rossi' )
```

- La condizione è verificata per le righe che NON contengono un certo valore, quindi non è esprimibile mediante un join, che richiede una corrispondenza fra valori.
- Però è equivalente a:

$\Pi_{Nome}(Dipartimento) - \Pi_{Dipart}\ (\sigma_{Cognome='Rossi'}(Impiegato)))$

le due query potevano essere unite da **except**

Nota **=any** e **<>all** si possono anche scrivere **in** e **not in**

Interrogazioni nidificate

Un'altra equivalenza può essere evidenziata con operatori aggregati.

```
select Dipart
      from Impiegato
      where Stipendio =
                  (select max(Stipendio)
                   from Impiegato)
```

equivale a

```
select Stipendio
      from Impiegato
      where Stipendio >= all ( select Stipendio
                               from Impiegato )
```

Query nidificate con più attributi

- Quando esiste un'uguaglianza fra un insieme di attributi è possibile, considerare query nidificate che riportano più attributi, e considerare una lista di attributi racchiusa fra parentesi tondi.

```
select *
from Persona P
      where (Nome, Cognome) not in
                (select Nome, Cognome
                   from Persona Q
                   where
                     P.CodFiscale <> Q.CodFiscale)
```

Interrogazioni nidificate, interpretazione

- La forma nidificata è “meno dichiarativa”, ma talvolta più leggibile (richiede meno variabili)
- La forma piana e quella nidificata possono essere combinate
- Le sottointerrogazioni non possono contenere operatori insiemistici (“l’unione si fa solo al livello esterno”); la limitazione non è significativa poiché gli operatori insiemistici vengono poco usati

Query nidificate

- Nome e reddito dei padri di persone che guadagnano più di 20

```
select distinct P.Nome, P.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
    and F.Reddito > 20

select Nome, Reddito
from Persone
where Nome in (select Padre
          from Paternita
          where Figlio = any (select Nome
                              from Persone
                              where Reddito > 20))
```

notare la `distinct`

Query nidificate

- Nome e reddito dei padri di persone che guadagnano più di 20

```
select distinct P.Nome, P.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
   and F.Reddito > 20

select Nome, Reddito
from Persone
where Nome in (select Padre
        from Paternita, Persone
        where Figlio = Nome
        and Reddito > 20)
```

Interrogazioni nidificate: commenti,

- La prima versione di SQL prevedeva solo la forma nidificata (o strutturata), con una sola relazione in ogni clausola `from`.
- Insoddisfacente:
  - la dichiaratività è limitata
  - non si possono includere nella target list attributi di relazioni nei blocchi interni

Esempio

- Nome e reddito dei padri di persone che guadagnano più di 20, con indicazione del reddito del figlio

```
select distinct P.Nome, P.Reddito, F.Reddito
  from Persone P, Paternita, Persone F
  where P.Nome = Padre and Figlio = F.Nome
     and F.Reddito > 20

  select Nome, Reddito, ????
  from Persone
  where Nome in (select Padre
           from Paternita
           where Figlio = any (select Nome
                               from Persone
                               where Reddito > 20))
```

Interrogazioni nidificate correlate

- Nelle query viste finora, per analizzare il risultato di una interrogazione nidificata si può supporre di valutare prima il risultato dell'interrogazione nidificata e poi quella della interrogazione che la contiene
  - Questo va anche a favore dell'efficienza in quanto l'interrogazione nidificata viene eseguita una sola volta.
- Nelle **query nidificate correlate,** esiste un riferimento tramite una variabile fra l'interrogazione nidificata e quella che la contiene (**passaggio di binding**).
  - In questo caso bisogna tornare alla definizione originaria di query, che implica il calcolo del prodotto cartesiano fra le tabelle e poi la verifica della condizione `where` separatamente per ciascuna riga.
- **Quindi**: per ogni riga della query esterna si valuta prima la query nidificata poi si calcola il predicato a livello di riga sulla query esterna.

Interrogazioni nidificate correlate: Visibilità delle variabili

- Le variabili SQL sono utilizzabili solo nell'ambito della query in cui sono definite o nell'ambito di una query nidificata all'interno di essa.
- Se due query sono allo stesso livello non possono condividere variabili
- **exists** è un operatore logico applicabile a query nidificate. Restituisce **vero** se la query dà un risultato **non nullo**, **falso** se è **nullo**.
- E' utilizzabile in modo significativo solo se esiste un passaggio di **binding** fra interrogazione esterna e interrogazione nidificata

`exist`: Esempio

- Le persone che hanno almeno un figlio

```
select *
from Persone
where exists ( select *
               from Paternita
               where Padre = Nome) or
      exists ( select *
               from Maternita
               where Madre = Nome)
```

exists

- Le persone che hanno almeno un omonimo (stesso nome e cognome, ma codice fiscale diverso)

```
select *
      from Persona P
      where exists ( select *
                     from Persona P1
                     where P1.Nome = P.Nome and
                     P1.Cognome = P.Cognome and
                     P1.CodFiscale <> P.CodFiscale
                   )
```

- In questo caso non è possibile eseguire prima la query nidificata, in quanto indeterminata se non si risolve il riferimento. Quindi per ogni riga dell'interrogazione esterna dovrà essere valutata l'interrogazione nidificata.

Interrogazioni nidificate: regole di visibilità

- regole di visibilità:
  - non è possibile fare riferimenti a variabili definite in blocchi più interni
  - se un nome di variabile è omesso, si assume riferimento alla variabile più "vicina"
- in un blocco si può fare riferimento a variabili definite in blocchi più esterni; la semantica base (prodotto cartesiano, selezione, proiezione) non funziona più, vedremo presto

Visibilità: esempio

- Le persone che hanno almeno un figlio

```
select *
from Persone
where exists (    select *
                  from Paternita
                  where Padre = Nome) or
      exists ( select *
                  from Maternita
                  where Madre = Nome)
```

Visibilità: esempio

- I padri i cui figli guadagnano tutti più di 20

```
 select distinct Padre
from Paternita Z
where not exists (
          select *
          from Paternita W, Persone
          where W.Padre = Z.Padre
               and W.Figlio = Nome
               and Reddito <= 20)
```

Visibilità: esempio

- I padri i cui figli guadagnano tutti più di 20

```
select distinct Padre
   from Paternita
   where not exists (
              select *
              from Persone
              where Figlio = Nome
                   and Reddito <= 20)
```

NO!!!

Semantica delle espressioni "correlate"

- L'interrogazione interna viene eseguita una volta per ciascuna ennupla dell'interrogazione esterna

Visibilità

- scorretta:

```
select *
from Impiegato
where Dipart in (select Nome
                 from Dipartimento D1
                 where Nome = 'Produzione') or
      Dipart in (select Nome
                 from Dipartimento D2
                 where D2.Citta = D1.Citta)
```