SUBQUERY

Subquery

Una subquery è un comando Select, racchiuso tra parentesi tonde, inserito all'interno di un comando SQL, per esempio un'altra Select.

Le subquery possono essere utilizzate nei seguenti casi:

- In espressioni di confronto
- In espressioni di confronto quantificato
- In espressioni IN
- In espressioni EXISTS
- Nel calcolo di espressioni

Tipi di subquery

Targa	Cod_mod	Categoria	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm

- Tre tipologie: scalare, di colonna, di tabella
- Subquery Scalare: è un comando Select che restituisce un solo valore.
- SELECT Max(Cilindrata) FROM Veicoli
- SELECT Cod_Categoria FROM Veicoli Where Targa="123456"

- Subquery di Colonna: è un comando Select che restituisce una colonna
- SELECT cod_categoria FROM Veicoli

- Subquery di Tabella: è un comando Select che restituisce una tabella
- SELECT Targa, Cod_mod, Posti FROM Veicoli

Subquery in espressioni di confronto

Categorie

Cod_cat	Nome_cat

Veicoli

Targa	Cod_mod	Categoria	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm

Vogliamo trovare tutti i veicoli della categoria "Autovettura"

Mediante Join

```
Select Veicoli.*
From Categorie, Veicoli
Where Categoria=Cod_cat
  and Nome_Cat = 'Autovettura'
```

Mediante Subquery

```
Select * From Veicoli
Where Categoria = (Select cod_cat
                   From Categorie
                   Where nome_cat='Autovettura')
```

Questa subquery restituisce un tipo scalare, ossia restituisce un singolo valore

Uso di subquery con funzioni di gruppo

Ricordiamo che nelle select semplici non è possibile utilizzare contemporaneamente funzioni di gruppo e funzioni su singole righe. Questo viene reso possibile mediante l'uso delle subquery.

Esempio: tutti i veicoli di cilindrata superiore alla media delle cilindrate.

```
Select *
From Veicoli
Where Cilindrata> (select AVG(Cilindrata)
                   From Veicoli)
```

Esempio: La tarda dei veicoli di cilindrata massima.

```
Select targa
From Veicoli
Where Cilindrata = (Select max(cilindrata)
                    From Veicoli)
```

Veicoli

Targa	Cod_mod	Categoria	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm

Subquery + Join

Veicoli

Targa	Cod_mod*	Categoria	Cilindrata	Cod_combust.	cav.Fis	Velocita	Posti	Imm

Modelli

Cod_Mod	Nome_Mod	Cod_Fab	Num_versioni

Selezionare i modelli che presentano più versioni del numero minimo di versioni dei veicoli a benzina (Cod_combustibile='01').

```
Select * From Modelli
Where num_versioni >
                    (Select min (num_versioni)
                        From Veicoli, Modelli
                        Where Veicoli.cod_mod=Modelli.cod_mod
                            And cod_combust='01')
```

Subquery Annidate

Veicolic

Targa	Cod_mod*	Categoria	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm

Modelli

Cod_Mod	Nome_Mod	Cod_Fab*	Cilind_Media

Fabbrica

Cod_Fab	Nome_Fab

Selezionare targa e velocità dei veicoli che appartengono a Modelli prodotti nella Fabbrica FIAT

```
Select targa, velocita
From Veicoli
Where cod_mod in (select cod_mod From Modelli
                  Where cod_fab = (select cod_fab From Fabbrica
                                   Where Nome_Fab='FIAT') )
```

Questa subquery restituisce un tipo scalare, ossia restituisce un singolo valore

Esempio

Veicoli

Targa	Cod_mod*	Categoria	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm

Modelli

Cod_Mod	Nome_Mod	Cod_Fab	Num_versioni

Selezionare i modelli che presentano più versioni del numero minimo di versioni dei veicoli a benzina (Cod_combustibile='01'). Già realizzata mediante subselect + join

```
Select * From Modelli
Where num_versioni > (Select min (num_versioni)
                      From Modelli
                      Where cod_mod IN (select cod_mod
                                        From veicoli
                                        Where cod_combust='01') )
```

Esempio di SELECT nidificate

nome e reddito del padre di Franco

Mediante join:

```
SELECT Nome, Reddito
FROM Persone, Paternita
WHERE Nome = Padre AND Figlio = 'Franco'
```

Mediante subquery:

```
SELECT Nome, Reddito
FROM Persone
WHERE Nome =(SELECT Padre
             FROM Paternita
             WHERE Figlio = 'Franco')
```

Si può usare = poiché la query interna restituisce un solo valore

Paternità

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Persone

NOME	ETA	REDDITO
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	29
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Esempio

Dobbiamo estrapolare i redditi dei padri

Dobbiamo estrapolare i redditi dei figli

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
```

Mediante Join

Persone P

NOME	ETA	REDDITO
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	29
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

join

Paternità

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

join

Persone F

NOME	ETA	REDDITO
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	29
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Esempio =ANY

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

Mediante Join

```
SELECT distinct P.Nome, P.Reddito
  FROM Persone P, Paternita, Persone F
  WHERE P.Nome = Padre AND Figlio = F.Nome
   AND F.Reddito > 20
```

Mediante subquery

```
SELECT Nome, Reddito
FROM Persone
WHERE Nome IN (SELECT Padre
      FROM Paternita
      WHERE Figlio =ANY (SELECT Nome
                         FROM Persone
                         WHERE Reddito > 20))
```

Paternità

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Persone

NOME	ETA	REDDITO
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	29
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

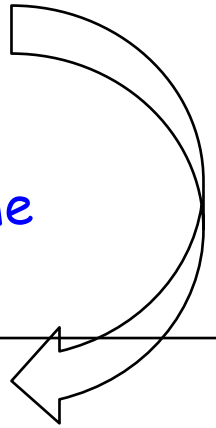
Interrogazioni nidificate, commenti

- La forma nidificata è meno dichiarativa, ma talvolta più leggibile (richiede meno variabili)
- La forma piana e quella nidificata possono essere combinate
- Le subquery non possono contenere operatori insiemistici
- Il problema si supera facilmente con intersezione e differenza (per cui esiste una forma alternativa) ma non sempre per l'unione

Esempio

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

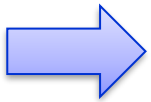
```
SELECT distinct P.Nome, P.Reddito, F.Reddito
  FROM Persone P, Paternita, Persone F
  WHERE P.Nome = Padre AND Figlio = F.Nome
   AND F.Reddito > 20
```



Mediante Join

```
SELECT Nome, Reddito, ????
FROM Persone
WHERE Nome in (SELECT Padre
   FROM Paternita
   WHERE Figlio = any (SELECT Nome
                FROM Persone
                WHERE Reddito > 20) )
```

Mediante Subquery (?)



Interrogazioni nidificate, commenti, 3

- regole di visibilità simili a quelle delle procedure nei linguaggi di programmazione:
  - 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

Regole di Visibilità

- La seguente query è 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)
```

  - D1 non è visibile nella seconda query nidificata in quanto le due subquery sono allo stesso livello

Confronto su più attributi

- Il confronto con il risultato di una query nidificata può essere basato su più attributi

- Esempio: Trovare tutti gli studenti che hanno un omonimo:

```
SELECT *
  FROM Studenti S
  WHERE (Nome, Cognome) IN (SELECT Nome, Cognome
          FROM Studenti S2
          WHERE S2.Matricola <> S.Matricola)
```

Condizione importante!

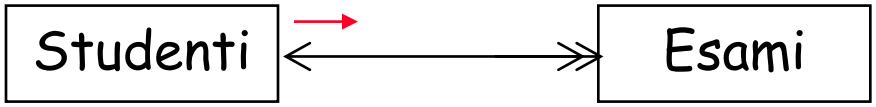
Commenti finali sulle query nidificate

- Query nidificate possono essere "meno dichiarative" in un certo senso ma spesso sono più facilmente interpretabili, in quanto si possono suddividere in blocchi più semplici da interpretare
- L'utilizzo di variabili deve rispettare le regole di visibilità cioè, una variabile può essere usata solo all'interno dello stesso blocco e in un blocco più interno
- Comunque, query nidificate complesse possono essere di difficile comprensione, soprattutto quando si usano molte variabili comuni tra blocchi diversi

QUANTIFICAZIONE

LA QUANTIFICAZIONE

- Tutte le interrogazioni su di una associazione multivalore vanno quantificate



- Non: gli studenti che hanno preso 30 (**ambiguo**!)

  ma:

  - Gli studenti che hanno preso sempre (o solo) 30: universale
  - Gli studenti che hanno preso qualche (almeno un) 30: esistenziale
  - Gli studenti che non hanno preso qualche 30 (senza nessun 30): universale
  - Gli studenti che non hanno preso sempre 30: esistenziale

LA QUANTIFICAZIONE

- Universale negata = esistenziale:
  - Non tutti i voti sono ≤ 24 = Almeno un voto >24 (esistenziale)
- Esistenziale negata = universale:
  - Non esiste voto diverso da 30 = Tutti i voti sono uguali a 30 (universale)

Any, All ed Exists

- le condizioni in SQL permettono anche il confronto fra un attributo e il risultato di una subquery che restituisce una colonna o una tabella
  - Operatore Scalare (ANY | ALL) SelectQuery
    - ANY: il predicato è vero se almeno uno dei valori restituiti da Query soddisfano la condizione
    - ALL: il predicato è vero se tutti i valori restituiti dalla Query soddisfano la condizione
  - quantificatore esistenziale
    - EXISTS SelectQuery
    - Il predicato è vero se la SelectQuery restituisce almeno una tupla

Subquery di confronto quantificato, regole

- La subquery deve essere una subquery di colonna
- La subquery deve essere inserita DOPO l'operatore di confronto quantificato
- Non è ammesso il confronto fra due subquery
- Nella subquery non è possibile utilizzare le clausole having e group by

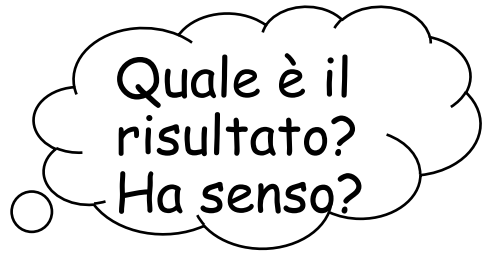
Subquery in espressioni EXISTS

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

IMPIEGATI(Matricola, Cognome, Nome, Mansione, IdReparto*,

StipendioAnnuale, PremioProduzione, DataAssunzione)

REPARTI(IdReparto,Nomereparto,Indirizzo,Città)



SELECT IdReparto **FROM** Reparti **WHERE EXISTS (**

SELECT * FROM Impiegati **WHERE** Mansione = 'Programmatore'**);**

Facendo uso di **NOT EXISTS** il predicato è vero se la subquery non restituisce alcuna Tupla.

Subquery in espressioni EXISTS

IMPIEGATI(Matricola, Cognome, Nome, Mansione, IdReparto*, StipendioAnnuale, PremioProduzione, DataAssunzione)

REPARTI(IdReparto,Nomereparto,Indirizzo,Città)

SELECT IdReparto **FROM** Reparti **WHERE EXISTS (**

SELECT * FROM Impiegati **WHERE** Mansione = 'Programmatore'**);**

- Occorre un meccanismo per rendere più flessibile la clausola exists, rendendo le condizioni della subquery dipendenti dalla tupla della query principale (query esterna):
  - Si introduce un legame fra la query e la subquery (query correlate)
  - Si definisce una variabile (un alias) nella query esterna, che si utilizza nella subquery

Subquery in espressioni EXISTS (query correlate)

- Tramite il predicato EXISTS è possibile effettuare il controllo sull'esistenza di righe che soddisfano specifiche condizioni. In questo caso la subquery è una subquery di tabella.

Categorie

Cod_cat	Nome_cat

Veicoli

Targa	Cod_mod	Categoria*	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm

Esempio: trovare i nomi di tutte le categorie per cui è presente almeno un veicolo

```
Select Nome_cat
From Categorie
Where EXISTS (Select *
              From Veicoli
              Where Categorie.cod_cat= Veicoli.categoria)
```

Condizione importante

Clausola NOT EXISTS

- Per verificare l'assenza di righe che rispondono a una determinata condizione è definita la forma negativa NOT EXISTS

Categorie

Cod_cat	Nome_cat

Veicoli

Targa	Cod_mod	Categoria*	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm

Esempio:
tutte le categorie per cui non è presente nessun veicolo

```
Select Cat_nome
From Categorie
Where NOT EXISTS (Select *
                  From Veicoli
                  where Categorie.Cod_cat= Veicoli. categoria)
```

RICORDIAMO LA SINTASSI DEL WHERE

- Combinazione booleana di predicati tra cui:
  - Expr Comp Expr
  - Expr Comp (Sottoselect che torna un valore)
  - **[NOT] EXISTS (Sottoselect)**
- Inoltre:
  - Espr Comp (ANY | ALL) (Sottoselect)
  - Expr [NOT] IN (Sottoselect) (oppure IN (v1,..,vn))
- Comp: <, =, >, <>, <=, >=

LA QUANTIFICAZIONE ESISTENZIALE: EXISTS

- Gli studenti con almeno un voto sopra 27
- In SQL:

```
SELECT s.Nome
FROM Studenti s
WHERE EXISTS (SELECT *
              FROM Esami e
              WHERE e.Matricola = s.Matricola AND e.Voto > 27)
```

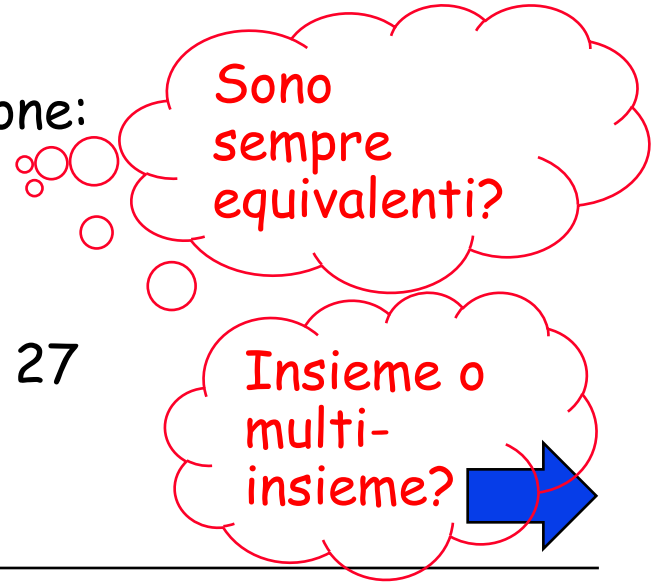
LA QUANTIFICAZIONE ESISTENZIALE: GIUNZIONE

- Gli studenti con almeno un voto sopra 27, tramite EXISTS:

```
SELECT s.Nome
FROM Studenti s
WHERE EXISTS (SELECT *
              FROM Esami e
              WHERE e.Matricola = s.Matricola AND e.Voto > 27)
```

- Stessa quantificazione esistenziale, tramite giunzione:

```
SELECT s.Nome
FROM Studenti s, Esami e
WHERE e.Matricola = s.Matricola AND e.Voto > 27
```



LA QUANTIFICAZIONE ESISTENZIALE: GIUNZIONE

Studenti

MATRIC	NOME	COGNOME	DATAISCRI
282320	Gianna	Neri	04-MAG-20
369871	Mario	Rossi	04-MAG-20
515140	Mario	Verdi	04-MAG-20
090456	Mario	Bianchi	04-MAG-20
579555	Luigi	Rossi	04-MAG-20
018701	Luca	Bianchi	04-MAG-20

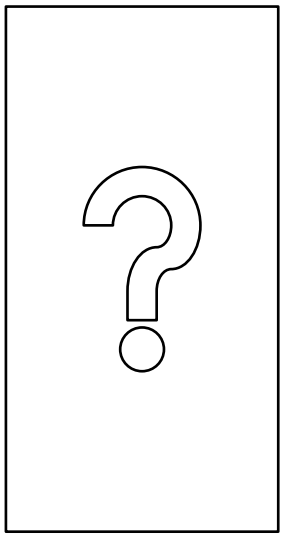
Esami

CODICE	MATRIC	VOTO	DATAESAME
A	369871	30	04-MAG-20
B	369871	29	10-MAG-21
C	369871	30	10-GIU-21
D	369871	27	20-GIU-21
A	515140	30	15-MAG-21
B	515140	28	12-MAG-21
C	090456	27	13-MAG-21
D	090456	26	14-GIU-21
A	018701	30	12-LUG-21
D	282320	20	12-GEN-21

```
SELECT s.Nome
FROM Studenti s
WHERE EXISTS (SELECT *
              FROM Esami e
              WHERE e.Matricola = s.Matricola AND e.Voto > 27)
```

```
SELECT s.Nome
FROM Studenti s, Esami e
WHERE e.Matricola = s.Matricola AND e.Voto > 27
```

```
SELECT distinct s.Nome FROM Studenti s , Esami e
WHERE e.Matricola = s.Matricola AND e.Voto > 27
```



LA QUANTIFICAZIONE ESISTENZIALE: GIUNZIONE

Studenti

MATRIC	NOME	COGNOME	DATAISCRI
282320	Gianna	Neri	04-MAG-20
369871	Mario	Rossi	04-MAG-20
515140	Mario	Verdi	04-MAG-20
090456	Mario	Bianchi	04-MAG-20
579555	Luigi	Rossi	04-MAG-20
018701	Luca	Bianchi	04-MAG-20

Esami

CODICE	MATRIC	VOTO	DATAESAME
A	369871	30	04-MAG-20
B	369871	29	10-MAG-21
C	369871	30	10-GIU-21
D	369871	27	20-GIU-21
A	515140	30	15-MAG-21
B	515140	28	12-MAG-21
C	090456	27	13-MAG-21
D	090456	26	14-GIU-21
A	018701	30	12-LUG-21
D	282320	20	12-GEN-21

```
SELECT s.Nome
FROM Studenti s
WHERE EXISTS (SELECT *
              FROM Esami e
              WHERE e.Matricola = s.Matricola AND e.Voto > 27)
```

NOME
Mario
Mario
Luca

```
SELECT s.Nome
FROM Studenti s, Esami e
WHERE e.Matricola = s.Matricola AND e.Voto > 27
```

NOME
Mario
Mario
Mario
Mario
Mario
Luca

```
SELECT distinct s.Nome FROM Studenti s , Esami e
WHERE e.Matricola = s.Matricola AND e.Voto > 27
```

NOME
Mario
Luca

LA QUANTIFICAZIONE ESISTENZIALE: GIUNZIONE

Studenti

MATRIC	NOME	COGNOME	DATAISCRI
282320	Gianna	Neri	04-MAG-20
369871	Mario	Rossi	04-MAG-20
515140	Mario	Verdi	04-MAG-20
090456	Mario	Bianchi	04-MAG-20
579555	Luigi	Rossi	04-MAG-20
018701	Luca	Bianchi	04-MAG-20

Esami

CODICE	MATRIC	VOTO	DATAESAME
A	369871	30	04-MAG-20
B	369871	29	10-MAG-21
C	369871	30	10-GIU-21
D	369871	27	20-GIU-21
A	515140	30	15-MAG-21
B	515140	28	12-MAG-21
C	090456	27	13-MAG-21
D	090456	26	14-GIU-21
A	018701	30	12-LUG-21
D	282320	20	12-GEN-21

```
SELECT s.Nome, s.cognome, s.matricola
FROM Studenti s
WHERE EXISTS (SELECT *
  FROM Esami e WHERE
      e.Matricola = s.Matricola AND e.Voto > 27)
```

NOME	COGNOME	MATRIC
Mario	Rossi	369871
Mario	Verdi	515140
Luca	Bianchi	018701

```
SELECT s.Nome, s.cognome, s.matricola
FROM Studenti s, Esami e
WHERE e.MatricolaStudente = s.Matricola AND e.Voto > 27
```

NOME	COGNOME	MATRIC
Mario	Rossi	369871
Mario	Rossi	369871
Mario	Rossi	369871
Mario	Verdi	515140
Mario	Verdi	515140
Luca	Bianchi	018701

LA QUANTIFICAZIONE ESISTENZIALE: GIUNZIONE

Studenti

MATRIC	NOME	COGNOME	DATAISCRI
282320	Gianna	Neri	04-MAG-20
369871	Mario	Rossi	04-MAG-20
515140	Mario	Verdi	04-MAG-20
090456	Mario	Bianchi	04-MAG-20
579555	Luigi	Rossi	04-MAG-20
018701	Luca	Bianchi	04-MAG-20

Esami

CODICE	MATRIC	VOTO	DATAESAME
A	369871	30	04-MAG-20
B	369871	29	10-MAG-21
C	369871	30	10-GIU-21
D	369871	27	20-GIU-21
A	515140	30	15-MAG-21
B	515140	28	12-MAG-21
C	090456	27	13-MAG-21
D	090456	26	14-GIU-21
A	018701	30	12-LUG-21
D	282320	20	12-GEN-21

```
SELECT s.Nome, s.cognome, s.matricola
FROM Studenti s, Esami e
WHERE e.MatricolaStudente = s.Matricola AND e.Voto > 27
```

NOME	COGNOME	MATRIC
Mario	Rossi	369871
Mario	Rossi	369871
Mario	Rossi	369871
Mario	Verdi	515140
Mario	Verdi	515140
Luca	Bianchi	018701

```
SELECT distinct s.Nome, s.cognome, s.matricola
FROM Studenti s , Esami e
WHERE e.MatricolaStudente = s.Matricola AND e.Voto > 27
```

NOME	COGNOME	MATRIC
Mario	Verdi	515140
Mario	Rossi	369871
Luca	Bianchi	018701

Subquery di confronto quantificato, esempio 1

Veicoli

Targa	Cod_mod	Categoria	Cilindrata	Cod_combust.	cav.Fisc	Velocita	Posti	Imm

Selezionare tutti i veicoli con cilindrata inferiore ad almeno una delle cilindrate dei veicoli con combustibile 02.

```
Select *  FROM Veicoli
Where cilindrata < Any (select cilindrata From Veicoli
                        Where cod_combust='02')

Select * From Veicoli
Where cilindrata< (select max(cilindrata) From Veicoli
                   Where cod_combust='02' )
```

Subquery di confronto quantificato, esempio

Veicoli

Targa	Cod_mod	Categoria	Cilindrata	Cod_combust.	cav.Fisc	Velocita	Posti	Imm

Selezionare tutti i veicoli con cilindrata inferiore a tutte le cilindrate dei veicoli con combustibile 02 (quindi inferiore alla più bassa di queste cilindrate)

```
Select *  FROM Veicoli
Where cilindrata < All (select cilindrata From Veicoli
                        Where cod_combust='02')
```

```
Select * From Veicoli
Where cilindrata< (select min(cilindrata) From Veicoli
                   Where cod_combust='02')
```

=ANY e <>ALL

In particolare le forme =ANY (equivalentemente IN) e <>ALL (equivalentemente NOT IN), forniscono un modo alternativo per realizzare intersezione e differenza dell'algebra relazionale.

Esempio: intersezione dei modelli con cilindrata inferiore a 1400 e quelli con codice fabbrica uguale a 001

```
Select cod_modello
From Veicoli
Where cilindrata < 1400 and
Cod_modello = ANY [IN] (select cod_modello
                        From Veicoli
                        Where cod_fab='001')
```

<>ALL (NOT IN)

Differenza fra le due tabelle precedenti: modelli con cilindrata inferiore a 1400 che non sono prodotti dalla fabbrica dal codice 001

```
Select cod_modello
From Veicoli
Where cilindrata<1400
and cod_modello  <> ALL [NOT IN] (Select cod_modello
                                  From Veicoli
                                  Where cod_fabbrica='001')
```

LA QUANTIFICAZIONE ESISTENZIALE: ANY

- ANY equivale ad EXISTS
- La solita query:

```
SELECT s.Nome    FROM Studenti s
WHERE EXISTS (SELECT *  FROM Esami e
                        WHERE e.Matricola = s.Matricola AND e.Voto > 27)
```

- Si può esprimere anche tramite ANY:

```
SELECT s.Nome   FROM Studenti s
WHERE s.Matricola =ANY (SELECT e.Matricola  FROM Esami e
                                    WHERE e.Voto >27)

SELECT s.Nome   FROM Studenti s
WHERE 27 <ANY (SELECT e.Voto FROM Esami e
                        WHERE e.Matricola = s.Matricola)
```

LA QUANTIFICAZIONE ESISTENZIALE: IN

- IN è solo un'abbreviazione di =ANY
- La solita query:

```
SELECT s.Nome   FROM Studenti s
WHERE s.Matricola =ANY (SELECT e.Matricola  FROM Esami e
                        WHERE e.Voto >27)
```

- Si può esprimere anche tramite IN:

```
SELECT s.Nome   FROM Studenti s
WHERE s.Matricola   IN   (SELECT e.Matricola  FROM Esami e
                          WHERE e.Voto >27)
```

Semantica delle espressioni "correlate"

- La query più interna può usare variabili della query esterna
- L'interrogazione interna viene eseguita una volta per ciascuna ennupla dell'interrogazione esterna

Esempio: trovare tutti gli studenti che hanno un omonimo (usando EXISTS):

```
SELECT * FROM Studenti S
  WHERE EXISTS (SELECT *
                FROM Studenti S2
                WHERE S2.Nome = S.Nome
                 AND S2.Cognome = S.Cognome
                 AND S2.Matricola <> S.Matricola)
```

Semantica delle espressioni "correlate", 2

- Esempio: trovare tutti gli studenti che NON hanno un omonimo:

```
SELECT *  FROM Studenti S
  WHERE NOT EXISTS (SELECT *
        FROM Studenti S2
        WHERE S2.Nome = S.Nome
         AND S2.Cognome = S.Cognome
         AND S2.Matricola <> S.Matricola)
```

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)
```

Persone

NOME	ETA	REDDITO
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	29
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

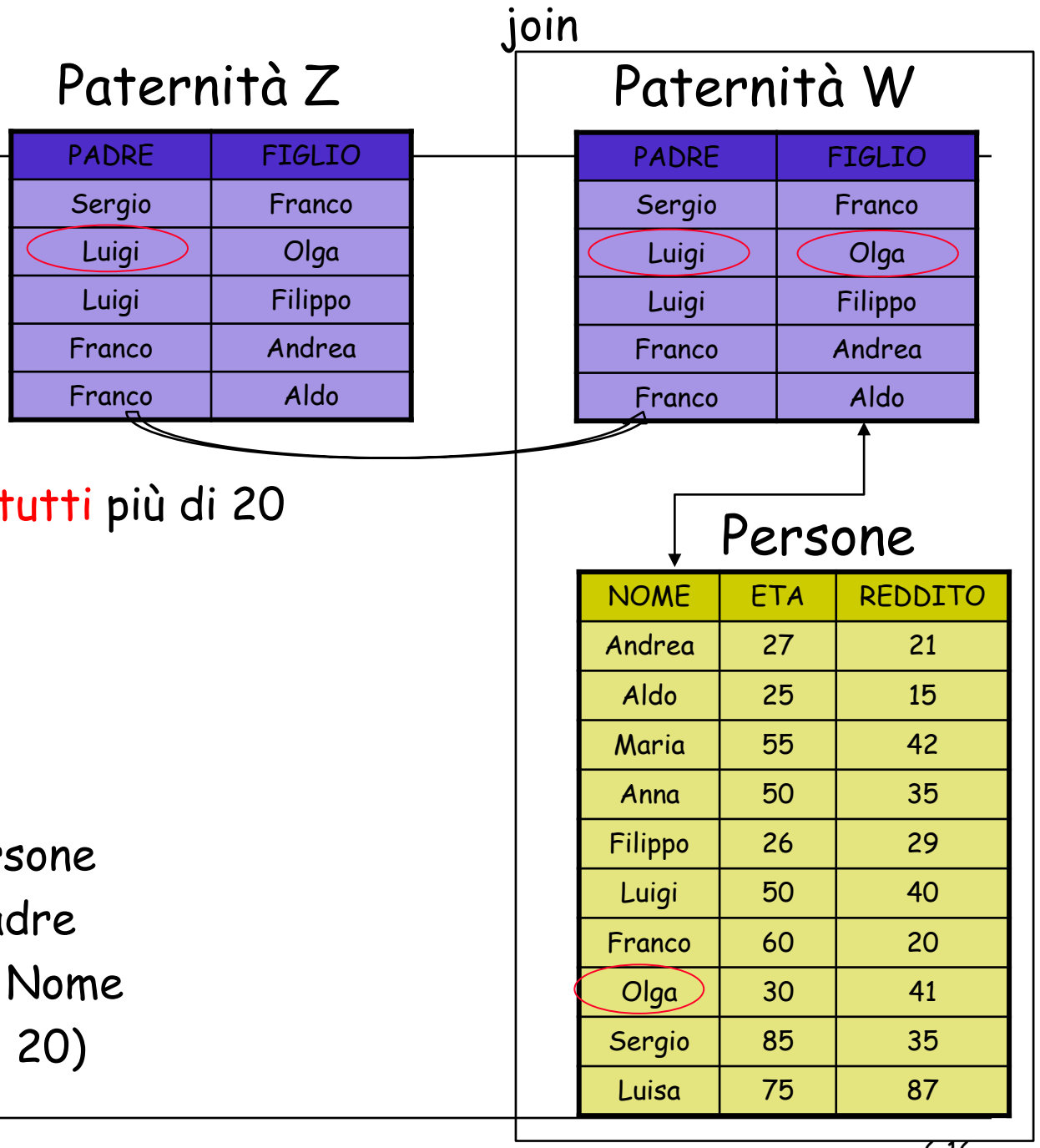
Maternità

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

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)
```

RIASSUMENDO

- La quantificazione esistenziale si fa con:
  - Exists (il più espressivo)
  - Giunzione
  - =Any, >Any, <Any…
  - IN
- =Any, >Any, <Any, IN,… non aggiungono potere espressivo
- Il problema vero è: **non confondere esistenziale con universale**!