

Segreteria Studenti Università

Le tabelle di riferimento sono:

Corsi (NomeCorso, CognomeDocente, Crediti, Anno)

Studenti(Matricola, Cognome, Nome, DataNascita)

Esami(Studente, Corso, DataE, Voto, Lode)

PROVE (NomeProva, Voto, StudenteEsame,
CorsoEsame)

In pratica non ammettiamo che uno studente può sostenere l'esame più volte, ovvero registriamo il voto solo di un esame.

Segreteria Studenti di Università

Tabelle

Corsi (NomeCorso, CognomeDocente, Crediti, Anno)

Studenti(Matricola, Cognome, Nome, DataNascita)

Esami(Studente, Corso, DataE, Voto, Lode)

NB: Uno studente può sostenere l'esame più volte, quindi
Studente, Corso non può essere chiave.

/* Creare un Dominio un nuovo dominio "dominio_voto" in SQL
di tipo intero con valore di default pari a 0 e valori ammissibili
fra 0 e 30 */

Create domain DOMINIO_VOTO

as integer check (value>0 and value <31)

/* modificare la tabella Esami (Studente, Corso, DataE, Voto,
Lode) in cui Voto ha come tipo il nuovo dominio creato
precedentemente, con valore di default 18

Alter Table Esami alter column Voto Type DOMINIO_VOTO;

Alter table Esami alter column Voto SET default 18;

Vincoli di
Integrità
Referenziale

Esercizio: Studenti

CONTENUTO TABELLA Studenti

```

36
37 select *
38 from Studenti;
39

```

| matricola | cognome | nome | data nascita |
|-----------|---------|---------|--------------|
| 001 | Primo | Mario | 1993-12-29 |
| 002 | Secondo | Franco | 1993-07-23 |
| 003 | Terzo | Michele | 1993-07-14 |
| 004 | Quarto | Mauro | 1993-08-11 |

CONTENUTO TABELLA Corsi

```

59 select *
60 from Corsi;
61

```

| nome corso | cognome docente | crediti | anno |
|---------------------------|-----------------|---------|------|
| Fondamenti di Informatica | Saggese | 6 | 1 |
| Tecnologie per il Web | D'Aniello | 12 | 2 |
| Sistemi Informativi | Gaeta | 6 | 3 |
| Lingua Inglese | Jones | 12 | 2 |
| Lingua Tedesca | Muller | 12 | 3 |
| Basi di Dati | D'Acerno | 6 | 3 |

CONTENUTO TABELLA Esami

```

78 select *
79 from Esami;
80

```

| studente | corso | data | voto | lode |
|----------|-----------------------|------------|------|-------|
| 001 | Tecnologie per il Web | 2015-07-25 | 30 | false |
| 001 | Basi di Dati | 2015-02-14 | 30 | true |
| 001 | Lingua Inglese | 2015-03-19 | 28 | false |
| 003 | Basi di Dati | 2015-02-14 | 19 | false |
| 003 | Sistemi Informativi | 2015-02-14 | 28 | false |

SUGGERIMENTO

Per scrivere bene una Query, inizialmente:
Iniziare dalla clausola FROM: Quali tabelle dobbiamo coinvolgere per ottenere il risultato
Proseguire con la clausola WHERE, trascuriamo inizialmente la PROIEZIONE, scrivendo SELECT *

Questo approccio facilita la verifica della correttezza del risultato, quando siamo certi della correttezza sostituiamo nella SELECT l'* con i nomi degli attributi di interesse

Sotto Query

Determinare il numero di studenti la cui media è superiore a 25.

Sotto Query

Determinare il numero di studenti la cui media è superiore a 25.

Suggerimento

Iniziamo ad individuare gli studenti che hanno la media superiore a 25. Questa Query diverrà la Sottoquery della Query che conta gli Studenti

Select E.Studente

From Esami E

Group By E.Studente

Having AVG(E.Voto)>=25;

Sotto Query

Determinare il numero di studenti la cui media è superiore a 25.

La precedente Query diviene una sottoquery di una Query che conta.

```
Select Count (*)
From
(
Select E.Studente
From Esami E
Group By E.Studente
Having AVG(E.Voto)>=25
)
AS MATRICOLASTUDENTI;
```

Interrogazioni Nidificate

- L'argomento della clausola WHERE si basa su condizioni composte da predicati semplici che vengono poi composti tramite operatori logici (P1 or P2 ... and Pn).
- Essenzialmente si tratta di una combinazione logica di predicati che consistono il più delle volte in un semplice confronto tra due valori
- SQL consente articolazioni di maggiore complessità: *si può confrontare un valore, ottenuto come risultato di una espressione valutata sulla singola riga, con il risultato dell'esecuzione di un'altra Query che viene definita nel predicato interno alla clausola WHERE.*

Interrogazioni Nidificate

- ❖ **Osservazione:** se in un predicato si effettua il confronto tra un attributo ed il risultato di una Query, ovviamente dobbiamo considerare il problema della **DISOMOGENEITA'** dei termini del confronto.
- ❖ In pratica è facile comprendere che da un lato del predicato (lato-attributo) abbiamo il valore di un attributo per una particolare riga, mentre dall'altro lato (lato-query) abbiamo il risultato di una Query, ovvero un insieme di valori.

Interrogazioni Nidificate

- ❖ **Il problema della disomogeneità** con una struttura più complessa 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 (=, <>, <, >, <=, >=).

Interrogazioni Nidificate

- 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.

Interrogazioni Nidificate

Determinare gli studenti che hanno superato almeno un esame con voto ≥ 25

```

Select Cognome, Nome      ←ESTERNA
  From Studenti
 Where 25 <= ANY
 (
      Select Voto          ← INTERNA
      From Esami
      Where Studente=Matricola
 );

```

Interrogazioni Nidificate

Determinare gli studenti che hanno superato almeno un esame con voto >25

Utilizzo di EXISTS

Select Cognome, Nome

From Studenti

Where Exists (

 Select Voto

 From Esami

 Where Studente=Matricola AND
 Voto>=25);

Interrogazioni Nidificate

Determinare gli studenti che hanno superato tutti gli esami con voto >25

Select Cognome, Nome

From Studenti

Where 25 <= ALL (

 Select Voto

 From Esami

 Where Studente=Matricola);

Attenzione ai NULL

| Tabelle di riferimento | | | | | |
|------------------------|--------|---------|---------|-----|---------|
| Maternità | Madre | Figlio | Persone | | |
| | Luisa | Maria | | | |
| | Luisa | Luigi | Nome | Età | Reddito |
| | Anna | Olga | Andrea | 27 | 21 |
| | Anna | Filippo | Aldo | 25 | 15 |
| | Maria | Andre | Maria | 55 | 42 |
| Paternità | Maria | Aldo | Anna | 50 | 35 |
| | Padre | Figlio | Filippo | 26 | 30 |
| | Sergio | Franco | Luigi | 50 | 40 |
| | Luigi | Olga | Franco | 60 | 20 |
| | Luigi | Filippo | Olga | 30 | 41 |
| | Franco | Andre | Sergio | 85 | 35 |
| | Franco | Aldo | Luisa | 75 | 87 |

| Operatori Insiemistici | | | | | |
|------------------------|--------|---------|---------|-----|---------|
| Maternità | Madre | Figlio | Persone | | |
| | Luisa | Maria | | | |
| | Luisa | Luigi | Nome | Età | Reddito |
| | Anna | Olga | Andrea | 27 | 21 |
| | Anna | Filippo | Aldo | 25 | 15 |
| | Maria | Andre | Maria | 55 | 42 |
| Paternità | Maria | Aldo | Anna | 50 | 35 |
| | Padre | Figlio | Filippo | 26 | 30 |
| | Sergio | Franco | Luigi | 50 | 40 |
| | Luigi | Olga | Franco | 60 | 20 |
| | Luigi | Filippo | Olga | 30 | 41 |
| | Franco | Andre | Sergio | 85 | 35 |
| | Franco | Aldo | Luisa | 75 | 87 |

OSSERVARE IL COMPORTAMENTO DI EXIST (istruzione) E IN() (funzione)

```

SELECT Nome,Reddito
FROM Persone
WHERE EXISTS (
    SELECT Figlio
    FROM Paternita
    UNION
    SELECT Figlio
    FROM Maternita )

```


Operatori Insiemistici

OSSERVARE IL COMPORTAMENTO DI **EXIST** (istruzione) E **IN()** (funzione)

❖ **IN** verifica che il valore di un dato campo ricada in una lista (predefinita o ricavata da una **SELECT**):

Ad esempio: **Where** attributo **IN** (20, 40, 60, 80, 100)

La condizione è soddisfatta se il valore di attributo ricade in quei 4 indicati.

❖ **EXISTS** verifica solo che una subquery restituisca dei record, indipendentemente dal loro valore

Ad esempio: **Where EXISTS** (select 1 from table2 where <condizioni>)

La condizione è vera se la subquery restituisce anche solo una riga.

Interrogazioni Nidificate senza correlazioni

- Una 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 memorizzato nella tabella risultato.

Interrogazioni Nidificate senza correlazioni

- Questa **interpretazione** di una interrogazione nidificata è detta **semplice**
- Tale interpretazione è accettabile quanto l'interrogazione nidificata viene eseguita una sola volta, ovvero essa è corretta nel caso in cui le variabili di range definite nell'interrogazione più esterna non vengano utilizzate nell'ambito dell'interrogazione più interna.

Interrogazioni Nidificate con correlazione

- Spesso l'interrogazione nidificata fa riferimento al contesto dell'interrogazione esterna che la racchiude;
- Tale situazione accade tramite una variabile di range definita nell'interrogazione più esterna ed usata nell'ambito della interrogazione nidificata «interna».
- In questi casi si parla di Interrogazioni **nidificate con correlazione**.

Interrogazioni Nidificate con correlazione

- Nel caso di interrogazioni nidificate con correlazione, **l'interpretazione semplice fornita precedentemente non è più valida.**
- In questi casi 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).

Semantica delle espressioni “correlate”

- ❖ **L'interrogazione interna viene eseguita una volta per ciascuna ennupla della query esterna**

Interrogazioni Nidificate con correlazione

- 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)
```

- Nell'es. vengono trattate una a una le righe della **variabile I**; per ogni riga dell'esterna, 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 con correlazione

- 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)
```

- Si può osservare che l'interrogazione nidificata utilizza una variabile di range definita nell'interrogazione più esterna. In questo caso, per ogni riga esaminata nell'ambito della Query esterna, si deve valutare l'interrogazione nidificata.

Interrogazioni Nidificate con correlazione

```
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 considera la prima riga di **I** e si verifica se in **I1** esiste una tupla con stessi valori di **Nome, Cognome** con diverso valore di **Codice Fiscale**.
- Tale tupla non esiste, perciò l'interrogazione nidificata restituisce una relazione vuota e **EXISTS** restituisce il valore **FALSE**.

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

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

Interrogazioni Nidificate con correlazione

```
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 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**.
- La tupla esiste, perciò la Query nidificata restituisce una relazione non vuota e **EXISTS** restituisce il valore **TRUE**. La tupla **<'Carlo', 'Bianchi', 'B013'>** è parte del risultato della Query esterna.

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

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

Interrogazioni Nidificate con correlazione

```
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 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.
- La tupla esiste, perciò la query nidificata restituisce una relazione non vuota e **EXISTS** restituisce il valore **TRUE**. La tupla **<'Carlo','Bianchi','C014'>** è parte del risultato della query esterna

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

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

Interrogazioni Nidificate con correlazione

```
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 con correlazione

- Il processo - *per ogni riga esaminata nell'ambito dell'interrogazione esterna, si deve valutare l'interrogazione nidificata* - 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 (no allo stesso livello).

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 sottointerrogazioni non possono contenere operatori insiemistici (“l'unione si fa solo al livello esterno”)
- la limitazione non è significativa

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')
```

SELECT Tabelle di riferimento

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

Esempio Interrogazioni 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**

SELECT Tabelle di riferimento

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

Esempio Interrogazioni 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)
```

SELECT Tabelle di riferimento

| Maternità | | | Paternità | | | Persone | | |
|-----------|---------|--|-----------|---------|--|---------|-----|---------|
| Madre | Figlio | | Padre | Figlio | | Nome | Età | Reddito |
| Luisa | Maria | | | | | Andrea | 27 | 21 |
| Luisa | Luigi | | | | | Aldo | 25 | 15 |
| Anna | Olga | | | | | Maria | 55 | 42 |
| Anna | Filippo | | | | | Anna | 50 | 35 |
| Maria | Andre | | | | | Filippo | 26 | 30 |
| Maria | Aldo | | | | | Luigi | 50 | 40 |
| | | | Sergio | Franco | | Franco | 60 | 20 |
| | | | Luigi | Olga | | Olga | 30 | 41 |
| | | | Luigi | Filippo | | Sergio | 85 | 35 |
| | | | Franco | Andre | | Luisa | 75 | 87 |
| | | | Franco | Aldo | | | | |

Interrogazioni nidificate, commenti, 2

- 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 Interrogazioni nidificate

- 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))
```

19

SELECT Tabelle di riferimento

Maternità

| Madre | Figlio |
|-------|---------|
| Luisa | Maria |
| Luisa | Luigi |
| Anna | Olga |
| Anna | Filippo |
| Maria | Andre |
| Maria | Aldo |

Paternità

| Padre | Figlio |
|--------|---------|
| Sergio | Franco |
| Sergio | Olga |
| Luigi | Filippo |
| Franco | Andre |
| 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 |
| Luigi | 75 | 87 |

20



DB Algoritmi, 2. Ediz. © F. Fummi, 2012

Interrogazioni nidificate

- regole di visibilità:
 - non è possibile fare riferimenti a variabili definite in blocchi più interni
 - se un nome di variabile è omissso, 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

Quantificazione esistenziale

- Ulteriore tipo di condizione
 - **EXISTS** (Sottoespressione)

Esempio Quantificazione esistenziale

- 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)

```

SELECT Tabelle di riferimento

| Maternità | Madre | Figlio | Persone | | |
|-----------|--------|---------|---------|-----|---------|
| | | | Nome | Età | Reddito |
| | Luisa | Maria | | | |
| | Luisa | Luigi | Andrea | 27 | 21 |
| | Anna | Olga | Aldo | 25 | 15 |
| | Anna | Filippo | Maria | 55 | 42 |
| | Maria | Andre | Anna | 50 | 35 |
| | Maria | Aldo | Anna | 50 | 35 |
| Paternità | Padre | Figlio | Filippo | 26 | 30 |
| | Sergio | Franco | Luigi | 50 | 40 |
| | Luigi | Olga | Franco | 60 | 20 |
| | Luigi | Filippo | Olga | 30 | 41 |
| | Franco | Andre | Sergio | 85 | 35 |
| | Franco | Aldo | Luisa | 75 | 87 |

Differenza e nidificazione

Estrarre in nomi che non sono cognomi per qualche impiegato

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

select Nome
from Impiegato I
where not exists (select *
                  from Impiegato
                  where Cognome = I.Nome)
```

Esercizio

```
-- =====
-- DROP delle tabelle FORNITORI, PRODOTTI e CATALOGO
-- =====

DROP TABLE IF EXISTS Fornitori CASCADE;
DROP TABLE IF EXISTS Prodotti CASCADE;
DROP TABLE IF EXISTS Catalogo CASCADE;
-- =====

-- Creazione delle tabelle
CREATE TABLE Fornitori( fid          CHAR(2) PRIMARY KEY,
                        nome          CHAR(20),
                        indirizzo     CHAR(20) );

CREATE TABLE Prodotti( pid          CHAR(3) PRIMARY KEY,
                        nome          CHAR(20),
                        colore        CHAR(20) );
CREATE TABLE Catalogo( fid          CHAR(2),
                        pid          CHAR(3),
                        costo         REAL,
                        FOREIGN KEY (fid) REFERENCES FORNITORI(fid),
                        FOREIGN KEY (pid) REFERENCES PRODOTTI(pid),
                        PRIMARY KEY(fid,pid));
```

Esercizio

Prodotti, 16 Righe

| | pid [PK] character (3) | nome character (20) | colore character (20) |
|----|---------------------------|------------------------|--------------------------|
| 1 | P1 | Volante | Nero |
| 2 | P2 | Volante | Rosso |
| 3 | P3 | Carrozzeria | Nero |
| 4 | P4 | Carrozzeria | Rosso |
| 5 | P5 | Carrozzeria | Verde |
| 6 | P6 | Cerchione | Nero |
| 7 | P7 | Cerchione | Rosso |
| 8 | P8 | Ruota | Nero |
| 9 | P9 | Sedile | Nero |
| 10 | P10 | Sedile | Rosso |
| 11 | P11 | Sedile | Verde |
| 12 | P12 | Tappetino | Nero |
| 13 | P13 | Tappetino | Rosso |
| 14 | P14 | Tappetino | Verde |
| 15 | P15 | Casco | Rosso |
| 16 | P16 | Casco | Verde |

Script "negoziario.sql"

Fornitori, 5 Righe

| | fid [PK] character (2) | nome character (20) | indirizzo character (20) |
|---|---------------------------|------------------------|-----------------------------|
| 1 | F1 | ACME | via Hollywood |
| 2 | F2 | Ingegneria | via Eudossiana |
| 3 | F3 | Sapienza | via Scarpa |
| 4 | F4 | DIS | via Ariosto |
| 5 | F5 | Gest | via Buonarroti |

Catalogo 32 Righe

| | fid [PK] character (2) | pid [PK] character (3) | costo real |
|----|---------------------------|---------------------------|---------------|
| 1 | F1 | P1 | 100 |
| 2 | F1 | P2 | 100 |
| 3 | F1 | P3 | 500 |
| 4 | F1 | P4 | 500 |
| 5 | F1 | P5 | 500 |
| 6 | F1 | P6 | 70 |
| 7 | F1 | P7 | 70 |
| 8 | F1 | P8 | 180 |
| 9 | F1 | P9 | 220 |
| 10 | F1 | P10 | 220 |
| 11 | F1 | P11 | 220 |
| 12 | F1 | P12 | 50 |
| 13 | F1 | P13 | 50 |
| 14 | F1 | P14 | 50 |
| 15 | F1 | P15 | 90 |
| 16 | F1 | P16 | 90 |
| 17 | F2 | P2 | 120 |
| 18 | F2 | P3 | 550 |
| 19 | F2 | P4 | 550 |
| 20 | F2 | P5 | 550 |

SQL DML – Query

Prodotti

| | pid [PK] character (3) | nome character (20) | colore character (20) |
|---|---------------------------|------------------------|--------------------------|
| 1 | P1 | Volante | Nero |
| 2 | P2 | Volante | Rosso |
| 3 | P3 | Carrozzeria | Nero |

Fornitori

| | fid [PK] character (2) | nome character (20) | indirizzo character (20) |
|---|---------------------------|------------------------|-----------------------------|
| 1 | F1 | ACME | via Hollywood |
| 2 | F2 | Ingegneria | via Eudossiana |
| 3 | F3 | Sapienza | via Scarpa |

Catalogo

| | fid [PK] character (2) | pid [PK] character (3) | costo real |
|---|---------------------------|---------------------------|---------------|
| 1 | F1 | P1 | 100 |
| 2 | F1 | P2 | 100 |
| 3 | F1 | P3 | 500 |
| 4 | F1 | P4 | 500 |

Trovare i prodotti forniti dalla ACME e da nessun altro

SQL DML – Query

Prodotti

| | pid [PK] character (3) | nome character (20) | colore character (20) |
|---|---------------------------|------------------------|--------------------------|
| 1 | P1 | Volante | Nero |
| 2 | P2 | Volante | Rosso |
| 3 | P3 | Carrozzeria | Nero |

Fornitori

| | fid [PK] character (2) | nome character (20) | indirizzo character (20) |
|---|---------------------------|------------------------|-----------------------------|
| 1 | F1 | ACME | via Hollywood |
| 2 | F2 | Ingegneria | via Eudossiana |
| 3 | F3 | Sapienza | via Scarpa |

Catalogo

| | fid [PK] character (2) | pid [PK] character (3) | costo real |
|---|---------------------------|---------------------------|---------------|
| 1 | F1 | P1 | 100 |
| 2 | F1 | P2 | 100 |
| 3 | F1 | P3 | 500 |
| 4 | F1 | P4 | 500 |

Trovare i prodotti forniti dalla ACME e da nessun altro

Trovo tutti i prodotti forniti da ACME

```
SELECT P.pid, P.nome, P.colore
FROM Prodotti P, Catalogo C, Fornitori F
WHERE P.pid=C.pid
      AND C.fid=F.fid
      AND F.nome = 'ACME'
```

Da questi, è necessario togliere i prodotti che sono venduti anche da altri fornitori

SQL DML – Query

Prodotti

| | pid [PK] character (3) | nome character (20) | colore character (20) |
|---|---------------------------|------------------------|--------------------------|
| 1 | P1 | Volante | Nero |
| 2 | P2 | Volante | Rosso |
| 3 | P3 | Carrozzeria | Nero |

Fornitori

| | fid [PK] character (2) | nome character (20) | indirizzo character (20) |
|---|---------------------------|------------------------|-----------------------------|
| 1 | F1 | ACME | via Hollywood |
| 2 | F2 | Ingegneria | via Eudossiana |
| 3 | F3 | Sapienza | via Scarpa |

Catalogo

| | fid [PK] character (2) | pid [PK] character (3) | costo real |
|---|---------------------------|---------------------------|---------------|
| 1 | F1 | P1 | 100 |
| 2 | F1 | P2 | 100 |
| 3 | F1 | P3 | 500 |
| 4 | F1 | P4 | 500 |

Trovare i prodotti forniti dalla ACME e da nessun altro

```
SELECT P.pid, P.nome, P.colore
FROM
    Prodotti P,
    Catalogo C,
    Fornitori F
WHERE P.pid=C.pid
      AND C.fid=F.fid
      AND F.nome = 'ACME'
```

Trovare i prodotti forniti da altri fornitori diversi dalla ACME

```
SELECT P.pid, P.nome, P.colore
FROM
    Prodotti P,
    Catalogo C,
    Fornitori F
WHERE P.pid=C.pid
      AND C.fid=F.fid
      AND F.nome <> 'ACME'
```


SQL DML – Query

Trovare i prodotti forniti dalla ACME e da nessun altro

COME METTO INSIEME LE DUE QUERY ?

Trovare i prodotti forniti dalla ACME e da nessun altro

```
SELECT P.pid, P.nome, P.colore
FROM
    Prodotti P,
    Catalogo C,
    Fornitori F
WHERE P.pid=C.pid
AND C.fid=F.fid
AND F.nome = 'ACME'
```

Trovare i prodotti forniti da altri fornitori diversi dalla ACME

```
SELECT P.pid, P.nome, P.colore
FROM
    Prodotti P,
    Catalogo C,
    Fornitori F
WHERE P.pid=C.pid
AND C.fid=F.fid
AND F.nome <> 'ACME'
```

SQL DML – Query Nidificate

Prodotti

| | pid [PK] character (3) | nome character (20) | colore character (20) |
|---|---------------------------|------------------------|--------------------------|
| 1 | P1 | Volante | Nero |
| 2 | P2 | Volante | Rosso |
| 3 | P3 | Carrozzeria | Nero |

Fornitori

| | fid [PK] character (2) | nome character (20) | Indirizzo character (20) |
|---|---------------------------|------------------------|-----------------------------|
| 1 | F1 | ACME | via Hollywood |
| 2 | F2 | Ingegneria | via Eudossiana |
| 3 | F3 | Sapienza | via Scarpa |

Catalogo

| | fid [PK] character (2) | pid [PK] character (3) | costo real |
|---|---------------------------|---------------------------|---------------|
| 1 | F1 | P1 | 100 |
| 2 | F1 | P2 | 100 |
| 3 | F1 | P3 | 500 |
| 4 | F1 | P4 | 500 |

Trovare i prodotti forniti dalla ACME e da nessun altro

```
SELECT P.pid, P.nome, P.colore
FROM
    Prodotti P,
    Catalogo C,
    Fornitori F
WHERE P.pid=C.pid
AND C.fid=F.fid
AND F.nome = 'ACME'
```






EXCEPT

```
SELECT P.pid, P.nome, P.colore
FROM
    Prodotti P,
    Catalogo C,
    Fornitori F
WHERE P.pid=C.pid
AND C.fid=F.fid
AND F.nome = 'ACME'
```

| | pid [PK] character (3) | nome character (20) | colore character (20) |
|---|---------------------------|------------------------|--------------------------|
| 1 | P6 | Cerchione | Nero |
| 2 | P9 | Sedile | Nero |
| 3 | P11 | Sedile | Verde |
| 4 | P16 | Casco | Verde |

Esercizio: Studenti

Determinare il Corso in cui la Studente Mario Primo ha sostenuto il primo esame

| Data Output | | Explain | Messages | Notifications | | | | | | |
|-------------|---|---------|--|---------------|---|--|---|--|---|--|
| |  studente [PK] character varying (16) | |  corso [PK] character varying (50) | |  datae date | |  voto integer | |  lode boolean | |
| 1 | 001 | | Tecnologie per il Web | | 2014-06-24 | | 30 | | false | |
| 2 | 001 | | Basi di Dati | | 2015-01-13 | | 30 | | true | |
| 3 | 003 | | Basi di Dati | | 2015-01-13 | | 18 | | false | |
| 4 | 001 | | Lingua Inglese | | 2015-02-18 | | 29 | | false | |
| | | | | | | | | | | |

Esercizio: Studenti

Determinare il Corso in cui la Studente Mario Primo ha sostenuto il primo esame

Suggerimento

- 1) Redigiamo una query aggregata per individuare la data del primo esame sostenuto da Mario

Esercizio: Studenti

Determinare il Corso in cui la Studente Mario Primo ha sostenuto il primo esame

Suggerimento: Determiniamo la Data minima che chiameremo **Data_Primo_Esame**

```
Select MIN(E1.DataE) AS Data_Primo_Esame
FROM STUDENTI S1, ESAMI E1
WHERE
    S1.Matricola=E1.Studente AND
    S1.Cognome='Primo' AND
    S1.Nome='Mario';
```

JOIN ESAMI e
STUDENTI

Esercizio: Studenti

Determinare il Corso in cui la Studente Mario Primo ha sostenuto il primo esame

Suggerimento

Questa QUERY, di fatto rappresenta la data del primo esame.

```
Select MIN(E.DataE) AS Data_Primo_Esame
FROM STUDENTI S, ESAMI E
WHERE
    S.Matricola=E.Studente AND
    S.Cognome='Primo' AND
    S.Nome='Mario';
```

Dobbiamo trovare il modo per definirla
PrimoEsameMario ... semplice:

AS **PRIMOESAMEMARIO**



Esercizio: Studenti

Determinare il Corso in cui la Studente Mario Primo ha sostenuto il primo esame

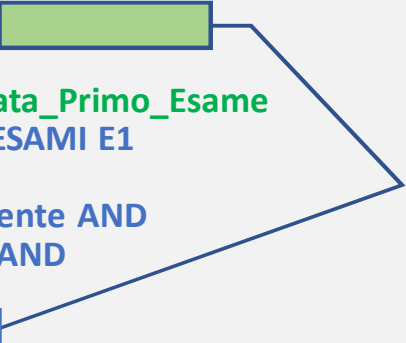
Suggerimento

«Battezziamo» la Query di nostro interesse **PRIMOESAMEMARIO** (essa è una tabella con una riga)

```

Select E.Corso
From ESAMI E, STUDENTI S, 
(
  Select MIN(E1.DataE) AS Data_Primo_Esame
  FROM STUDENTI S1, ESAMI E1
  WHERE
    S1.Matricola=E1.Studente AND
    S1.Cognome='Primo' AND
    S1.Nome='Mario'
) AS 

```


Esercizio: StudentiSuggerimento

Successivamente per trovare il nome del Corso che corrisponde alla data minima che abbiamo battezzato **PRIMOESAMEMARIO.Data_Primo_Esame**, ovvero dobbiamo selezionare tra gli esami che hanno come data proprio quella del risultato della query l'esame relativo a Mario Primo, avendo cura di fare il Join tra esame e studenti e aggregare il tutto e restituire il CORSO che ci interessa ovvero quello del primo esame. In pratica un blocco di questo tipo ...

```

Where S.Matricola=E.Studente AND
E.DataE=PRIMOESAMEMARIO.Data_Primo_Esame AND
S.Cognome='Primo' AND
S.Nome='Mario';

```

Esercizio: Studenti**Select E.Corso**From ESAMI E, STUDENTI S, 

(

Select MIN(E1.DataE) AS **Data_Primo_Esame**

FROM STUDENTI S1, ESAMI E1

WHERE

S1.Matricola=E1.Studente AND

S1.Cognome='Primo' AND

S1.Nome='Mario'

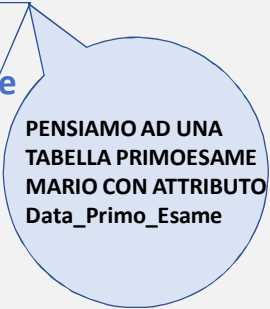
) AS **PRIMOESAMEMARIO****Where**

S.Matricola=E.Studente AND

E.DataE=**PRIMOESAMEMARIO.Data_Primo_Esame**

S.Cognome='Primo' AND

S.Nome='Mario';



PENSIAMO AD UNA
TABELLA PRIMOESAME
MARIO CON ATTRIBUTO
Data_Primo_Esame

Esercizio: Studenti

Determinare il Corso in cui la Studente Mario Primo ha sostenuto il primo esame

Questo esercizio precedentemente svolto può essere elegantemente svolto con una esterna e una query annidata.

Select E.Corso

From ESAMI E, STUDENTI S

WHERE

S.Cognome='Primo' AND

S.Nome='Mario' AND

S.Matricola=E.Studente AND

E.DataE = (

Select MIN(E1.DataE)

FROM ESAMI E1

WHERE

S.Matricola=E1.Studente);

BASI DI DATI

Materiale utilizzato e bibliografia

- Le slide utilizzate dai docenti per le attività frontali sono in gran parte riconducibili e riprese dalle slide originali (con alcuni spunti parziali ripresi dai libri indicati) realizzate da:
- ✓ autori del libro *Basi di Dati* (Atzeni e altri) testo di riferimento del corso *Basi di Dati* e sono reperibili su internet su molteplici link oltre che laddove indicato dagli stessi autori del libro;
- ✓ Prof.ssa Tiziana Catarci e dal dott. Ing. Francesco Leotta – corso di *Basi di Dati* dell'Università degli Studi La Sapienza di Roma al seguente link ed altri: <http://www.dis.uniroma1.it/~catarci/basidatGEST.html> (molto Interessanti anche le lezioni su YouTube).
- ✓ Prof. Luca Allulli e Umberto Nanni, *Libro Fondamenti di basi di dati*, editore HOEPLI (testo di facile lettura ed efficace).
- Diverse slide su specifici argomenti utilizzate dai docenti per le attività frontali sono anche in parte riconducibili e riprese dalle slide originali – facilmente reperibili e accessibili su internet - realizzate da:
- Prof.ssa Roberta Aiello – corso *Basi di Dati* dell'Università di Salerno
- Prof. Dario Maio - corso *Basi di Dati* dell'Università di Bologna al seguente link ed altri: <http://bias.csr.unibo.it/maio>
- Prof. Marco Di Felice - corso *Basi di Dati* dell'Università di Bologna al seguente link ed altri: <http://www.cs.unibo.it/difelice/dbsi/>
- Prof. Marco Maggini e prof. Franco Scarselli - corso *Basi di Dati* dell'Università di Siena ai seguenti link ed altri: [http://staff.icar.cnr.it/pontieri/didattica/LabSI/lezioni/_preliminari-DB1%20\(Maggini\).pdf](http://staff.icar.cnr.it/pontieri/didattica/LabSI/lezioni/_preliminari-DB1%20(Maggini).pdf)
- Prof. Fabio A. Schreiber - corso *Basi di Dati* del Politecnico di Milano al seguente link ed altri: <https://schreiber.faculty.polimi.it/BasidiDati0607/LucidTeoria/IntroduzioneCR.pdf>
- Prof.ssa Raffaella Gentilini - corso *Basi di Dati* dell'Università di Perugia al seguente link ed altri: <http://www.dmi.unipg.it/raffaella.gentilini/BD.htm>
- Prof. Enrico Giunchiglia - corso *Basi di Dati* dell'Università di Genova al seguente link ed altri: <http://www.star.dist.unige.it/~enrico/BasiDiDati/>
- Prof. Maurizio Lenzerini - corso *Basi di Dati* dell'Università degli Studi La Sapienza di Roma al seguente link ed altri: <http://didattica.info.altervista.org/Quinta/Database2.pdf>
- Prof.ssa Claudia D'Amato - corso *Basi di Dati* dell'Università di Bari al seguente link ed altri: <http://www.di.uniba.it/~cdamato/>

Atzeni, Ceri, Fraternali, Paraboschi, Torlone
Basi di dati Quinta edizione
McGraw-Hill Education, 2018