

*Corso di Laurea in Ingegneria Gestionale  
Sapienza Università di Roma*

# ***Corso di Basi di Dati***

## ***A.A. 2019/2020***

6 – SQL : Interrogazioni nidificate

Tiziana Catarci

# Interrogazioni nidificate

---

- ▶ In generale si è visto che l'argomento della clausola **WHERE** si basa su condizioni composte da predicati semplici (combinati 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

<u>Nome</u>	Città
-------------	-------

### Interrogazione Esterna

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

In questo caso, l'utilizzo di "IN" coincide con "= ANY". Portano entrambi allo stesso risultato.

### Interrogazione nidificata (o Interna)

- ▶ 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

<u>Nome</u>	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

<u>Nome</u>	Città
-------------	-------

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

In questo caso, l'utilizzo di “NOT IN” coincide con “<> ALL”. Portano entrambi allo stesso risultato.

- ▶ L'interrogazione nidificata seleziona quelle righe dell'attributo **Dipart** 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 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

<u>Nome</u>	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 <> **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	<u>CodFiscale</u>
------	---------	-------------------

```
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	<u>CodFiscale</u>
------	---------	-------------------

```
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 **FALSE**.

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  $\langle \text{'Carlo', 'Bianchi', 'B013'} \rangle$  farà parte del risultato dell'interrogazione esterna.

I

Nome	Cognome	<u>CodFiscale</u>
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014

I1

Nome	Cognome	<u>CodFiscale</u>
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	<u>CodFiscale</u>
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014

**I1**

Nome	Cognome	<u>CodFiscale</u>
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	<u>CodFiscale</u>
Mario	Rossi	A012
Carlo	Bianchi	B013
Carlo	Bianchi	C014



***Risultato finale  
dell'interrogazione***

Nome	Cognome	<u>CodFiscale</u>
Carlo	Bianchi	B013
Carlo	Bianchi	C014

# Interrogazioni nidificate – Esempio 1

---

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

## Impiegato

Nome	Cognome	<u>CodFiscale</u>
------	---------	-------------------

```
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	<u>CodFiscale</u>
------	---------	-------------------

```
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 molti DBMS**...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	<u>CodFiscale</u>
------	---------	-------------------

```
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 **molti DBMS**...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 puri, ovvero i cantanti che hanno eseguito solo canzoni di cui erano anche autori

**Cantante**

<u>Nome</u>	<u>Canzone</u>
-------------	----------------

**Autore**

<u>Nome</u>	<u>Canzone</u>
-------------	----------------

```
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**

<u>Nome</u>	<u>Canzone</u>
-------------	----------------

**Autore**

<u>Nome</u>	<u>Canzone</u>
-------------	----------------

```
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

<u>Nome</u>	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')
```

23



Nome	Cognome
Carlo	Bianchi
Marco	Franco

6 – SQL : Interrogazioni nidificate

# 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)
```

24



Dipart

Direzione

6 – SQL : Interrogazioni nidificate



# 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

<u>Nome</u>	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)
```

25



Dipart
Direzione

6 – SQL : Interrogazioni nidificate

# 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');
```



RedditoCompI
25

# Esercizio

---

## Negozi

<u>ID</u>	Nome	Città
-----------	------	-------

## Prodotti

<u>Codice</u>	Nome	Marca
---------------	------	-------

## Listino

<u>Negozi</u>	<u>Prodotto</u>	Prezzo
---------------	-----------------	--------

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

```
SELECT P.Codice, 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 Listino L2
        WHERE L2.Prodotto = P.Codice
      )
```

# Esercizio proposto per casa

---

**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 per casa

---

**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 per casa

---

- ▶ 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