
# SQL: SCRIVERE INTERROGAZIONI E ANNIDARLE

*Patrizio Dazzi*
*a.a. 2017 - 2018*

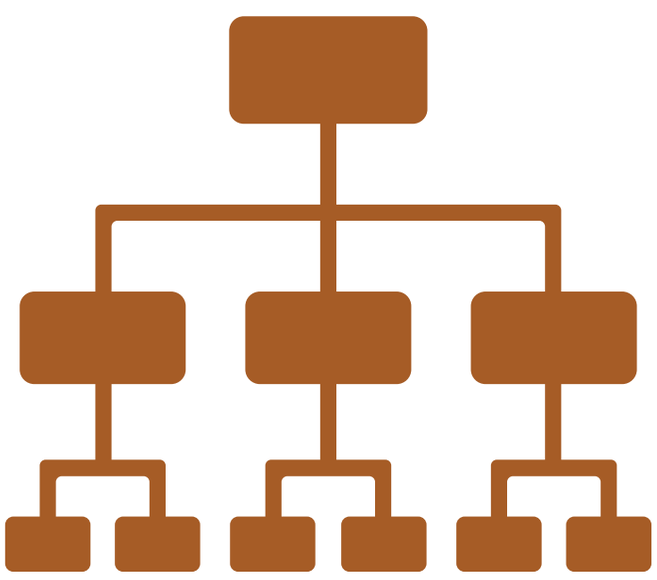
# COMUNICAZIONI

- Homeworks
  - oggi ultimo giorno per la consegna!!!!

- 3h di ricevimento rimaste prima del compitino

# PICCOLO RIASSUNTO DELLA PUNTATA PRECEDENTE

- SQL
  - introduzione
  - concetti
  - SELECT FROM WHERE
    - mapping dell’algebra relazionale
- Operatori insiemistici
  - unione, intersezione, differenza



# BREVE RICHIAMO: COME SCRIVERE LE INTERROGAZIONI

# FORMA STANDARD DELL'ALGEBRA RELAZIONALE

- Interrogazioni in algebra relazionale
  - risultato dell'applicazione di vari operatori
  - è possibile applicare gli operatori in ordine vario (es: prima selezioni o prima ridenominazione)

- Forma standard
  - qual è l'ordine standardizzato di applicazione degli operatori dell'algebra?

# FORMA STANDARD DELL'ALGEBRA RELAZIONALE

- Esempio
  - "Nome e Cognome dei professori ordinari che non hanno tesisti della laurea triennali"
- Strategia
  - (a) trovo nome e cognome di tutti i professori ordinari
  - (b) trovo nome e cognome dei professori che hanno tesisti della laurea triennale
  - faccio la differenza tra (a) e (b)

# (A) "COGNOMI E NOMI DEI PROF. ORDINARI"

Professori

| cod | cognome | nome | qualifica | facolta |
|---|---|---|---|---|
| **FT** | **Totti** | **Francesco** | **ordinario** | **Ingegneria** |
| **CV** | **Vieri** | **Christian** | **associato** | **Scienze** |
| **ADP** | **Del Piero** | **Alessandro** | **supplente** | **null** |

$$\text{ProfessoriOrdinari} = \sigma_{\text{qualifica = 'Ordinario'}} (\text{Professori})$$

| cod | cognome | nome | qualifica | facolta |
|---|---|---|---|---|
| **FT** | **Totti** | **Francesco** | **ordinario** | **Ingegneria** |

$$\text{NomiOrdinari} = \rho_{\text{cognome AS cognomeProf, nome AS nomeProf}} ( \pi_{\text{cognome, nome}} (\text{ProfessoriOrdinari}))$$

| cognomeProf | nomeProf |
|---|---|
| **Totti** | **Francesco** |

# (B) "COGNOMI E NOMI DI PROF. CON TESI TRIENNALI"

Professori

| cod | cognome | nome | qualifica | facolta |
|---|---|---|---|---|
| **FT** | **Totti** | **Francesco** | **ordinario** | **Ingegneria** |
| **CV** | **Vieri** | **Christian** | **associato** | **Scienze** |
| **ADP** | **Del Piero** | **Alessandro** | **supplente** | **null** |

Studenti

| matr | cognome | nome | ciclo | anno | relatore |
|---|---|---|---|---|---|
| **111** | **Rossi** | **Mario** | **laurea tr.** | **1** | **null** |
| **222** | **Neri** | **Paolo** | **laurea tr.** | **2** | **null** |
| **333** | **Rossi** | **Maria** | **laurea tr.** | **1** | **null** |
| **444** | **Pinco** | **Palla** | **laurea tr.** | **3** | **FT** |
| **77777** | **Bruno** | **Pasquale** | **laurea sp.** | **1** | **FT** |
| **88888** | **Pinco** | **Pietro** | **laurea sp.** | **1** | **CV** |

ProfessoriConTesisti = Studenti $\bowtie_{\text{relatore = cod}}$ Professori

# (B) "COGNOMI E NOMI DI PROF. CON TESI TRIENNALI"

$$\text{ProfessoriConTesisti} = \text{Studenti} \bowtie_{\text{relatore} = \text{cod}} \text{Professori}$$

| matr | S.cognome | S.nome | ciclo | anno | relatore | P.cognome | P.nome | qualifica | facolta |
|---|---|---|---|---|---|---|---|---|---|
| 444 | Pinco | Palla | laurea tr. | 3 | FT | Totti | Francesco | ordinario | Ing. |
| 77777 | Bruno | Pasquale | laurea sp. | 1 | FT | Totti | Francesco | ordinario | Ing. |
| 88888 | Pinco | Pietro | laurea sp. | 1 | CV | Vieri | Christian | associato | Sc. |

$$\text{ProfessoriConTesiTriennali} = \sigma_{\text{ciclo} = \text{'laurea tr.'}} (\text{ProfessoriConTesisti})$$

| matr | S.cognome | S.nome | ciclo | anno | relatore | P.cognome | P.nome | qualifica | facolta |
|---|---|---|---|---|---|---|---|---|---|
| 444 | Pinco | Palla | laurea tr. | 3 | FT | Totti | Francesco | ordinario | Ing. |

$$\text{NomiProfConTesiTriennali} = \rho_{\text{cognome AS cognomeProf, nome AS nomeProf}} ( \pi_{\text{Professori.cognome, Professori.nome}} (\text{ProfessoriConTesiTriennali}))$$

| cognomeProf | nomeProf |
|---|---|
| **Totti** | **Francesco** |

# "COGNOMI E NOMI DEI PROF. ORDINARI SENZA TESI TRIENNALI"

NomiOrdinari =

$\rho_{\text{cognome AS cognomeProf, nome AS nomeProf}}$ (
$\pi_{\text{cognome, nome}}$ (
$\sigma_{\text{qualifica = 'Ordinario'}}$ (
Professori))

| cognomeProf | nomeProf |
|---|---|
| **Totti** | **Francesco** |

NomiProfConTesiTriennali =

$\rho_{\text{cognome AS cognomeProf, nome AS nomeProf}}$ (
$\pi_{\text{cognome, nome}}$ (
$\sigma_{\text{ciclo = 'laurea tr.'}}$ (
Studenti $\bowtie_{\text{relatore = cod}}$ Professori )))

| cognomeProf | nomeProf |
|---|---|
| **Totti** | **Francesco** |

Risultato = NomiOrdinari – NomiProfConTesiTriennali

| cognomeProf | nomeProf |
|---|---|

# RIASSUMENDO

Risultato =

sottointerrogazione n. 1

$\rho_{\text{cognome AS cognomeProf, nome AS nomeProf}}$ (
$\pi_{\text{cognome, nome}}$ (
$\sigma_{\text{qualifica = 'Ordinario'}}$ (
Professori)))

operatore insiemistico

–

sottointerrogazione n. 2

$\rho_{\text{cognome AS cognomeProf, nome AS nomeProf}}$ (
$\pi_{\text{cognome, nome}}$ (
$\sigma_{\text{ciclo = 'laurea tr.'}}$ (
Studenti $\bowtie_{\text{relatore = cod}}$ Professori )))

ridenominazioni

proiezioni

selezioni

join

# FORMA STANDARD

- Varie sottointerrogazioni (una o più)
  - correlate con operatori insiemistici

- Ciascuna sottointerrogazione
  - prima: eventuali join o prodotti cartesiani
  - poi: eventuali selezioni
  - poi: eventuali proiezioni
  - infine: eventuali ridenominazioni

# FORMA STANDARD

- Attenzione
  - la forma standard non è necessariamente la più efficiente
  - esempio: anticipare la selezioni

$$\text{NomiProfConTesiTriennali} = \rho_{\text{cognome AS cognomeProf, nome AS nomeProf}} ( \pi_{\text{cognome, nome}} ( \sigma_{\text{ciclo = 'laurea tr.'}} ( \text{Studenti} \bowtie_{\text{relatore = cod}} \text{Professori} )))$$

# FORMA STANDARD

- Ordine alternativo:

$$\text{NomiProfConTesiTriennali} = \rho_{\text{cognome AS cognomeProf, nome AS nomeProf}} ( \pi_{\text{cognome, nome}} ( \sigma_{\text{ciclo = 'laurea tr.'}} (\text{Studenti}) \bowtie_{\text{relatore = cod}} \text{Professori} ))$$

esecuzione più efficiente:
si eliminano subito le ennuple
irrilevanti

# FORMA STANDARD (1)

- Metodo di scrittura delle interrogazioni
  - Stabilire se è necessario utilizzare operatori insiemistici e dividere in sottointerrogazioni
  - Per ogni sottointerrogazione, decidere da quali tabelle prelevare i dati
- Se le tabelle sono più di una,
  - (strategia a) metterle in prodotto cartesiano oppure
  - (strategia b) metterle in join con le condizioni opportune

# FORMA STANDARD (2)

- Scrivere le eventuali selezioni
  - (Strategia a) incluse le condizioni di Join
- Scrivere le eventuali proiezioni
- Scrivere le eventuali ridenominazioni finali
- Rimettere il tutto insieme applicando gli operatori insiemistici

# FORMA STANDARD – SUGGERIMENTI

- Suggerimento n.1
  - dare un nome a ciascun risultato parziale
  - rimettere assieme gli operatori solo alla fine

- Suggerimento n.2
  - tenere traccia dello schema dei risultati parziali generati da ciascun operatore
  - elenco degli attributi (nomi e tipi)
  - aiuta a scrivere le operazioni successive

# INTERROGAZIONI: METODO DI SCRITTURA

- Scrivere l'interrogazione in algebra relazionale utilizzando la forma standard

- Tradurre gli operatori nella sintassi di SQL

- Stabilire se sono necessari operatori insiemistici
  - dividere in sottointerrogazioni

# INTERROGAZIONI: METODO DI SCRITTURA (2)

- Per ogni sottointerrogazione
  - decidere da quali tabelle prelevare i dati

- Decidere eventuali alias

- Se le tabelle sono più di una,
  - strategia a) prodotti cartesiani oppure
  - strategia b) join con le condizioni opportune

# INTERROGAZIONI: METODO DI SCRITTURA (3)

- Scrivere le eventuali selezioni
  - strategia a) incluse le condizioni di Join

- Scrivere le eventuali proiezioni
  - e le eventuali funzioni aggregative

- Scrivere le eventuali eliminazione di duplicati
- Scrivere le eventuali ridenominazioni finali
- Scrivere gli eventuali operatori di ordinamento
- Rimettere le sottointerrogazioni insieme

# INTERROGAZIONI NIDIFICATE

# PICCOLA PREMESSA: CREAZIONE VISTE

- Essenzialmente sono query memorizzate nella base di dati
  - tabelle virtuali risultato di query sql

```sql
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Memorizzate e andiamo oltre…

# INTERROGAZIONI NIDIFICATE

- **SELECT Nidificate**
  - la clausola WHERE di una SELECT contiene un'altra SELECT

- **Due possibili schemi di uso**
  - condizioni basate su valori semplici (SELECT che restituiscono un singolo valore)
  - condizioni basate su collezioni (SELECT ordinarie che restituiscono insiemi di ennuple)

# INTERROGAZIONI NIDIFICATE

- **Condizioni su valori semplici**
  - confrontano il valore di un attributo con il risultato di una SELECT “scalare”
  - operatori: >, <, =, >=, <=, <>, LIKE, IS NULL

- **SELECT “scalare”**
  - SELECT che restituisce un’unica ennupla con un un unico attributo
  - tipicamente: funzione aggregativa

# INTERROGAZIONI NIDIFICATE

- **Esempio**: lo studente con la matricola più alta

```
SELECT matr, cognome, nome
FROM Studenti
WHERE matr = (SELECT max(matr)
              FROM Studenti);
```

| max(matr) |
| --- |
| 88888 |

**per ogni ennupla di Studenti, il valore della matricola viene confrontato con il numero 88888**

# INTERROGAZIONI NIDIFICATE

- **Condizioni su valori non scalari (collezioni)**
  - confrontano il valore di un attributo con il risultato di una SELECT generica (collezione di ennuple)
  - operatori: ordinari combinati con ANY, ALL

- **ANY**
  - "un elemento qualsiasi della collezione"; es: = ANY, oppure IN

- **ALL**
  - "tutti gli elementi della collezione"; es: > ALL

# INTERROGAZIONI NIDIFICATE

- **Esempio**: lo studente con la matricola più alta (senza funzioni aggregative)

```
SELECT matr, cognome, nome
FROM Studenti
WHERE matr >= ALL (SELECT matr
                   FROM Studenti);
```

**per ogni ennupla di Studenti, il valore della matricola viene confrontato con tutte le matricole**

| matr |
|---|
| 111 |
| 222 |
| 333 |
| 444 |
| 77777 |
| 88888 |

# INTERROGAZIONI NIDIFICATE

- **Sintatticamente**
  - no ORDER BY nelle SELECT nidificate

- **Semantica**
  - ogni volta che è necessario verificare la condizione, viene calcolato il risultato della SELECT interna
  - il processo si può ripetere a più livelli
  - in pratica: memorizzazione in una tabella temporanea

# INTERROGAZIONI NIDIFICATE

- **Nota**: Le interrogazioni nidificate possono sostituire i join
- **Esempio**: voti riportati in corsi della laurea triennale

```
SELECT voto
FROM Esami
WHERE corso =ANY (SELECT cod
                  FROM Corsi
                  WHERE ciclo='laurea tr.');
```

| cod |
| --- |
| PR1 |
| ASD |

**Stessa semantica del join**

# INTERROGAZIONI NIDIFICATE

- **Nota**: Le interrogazioni nidificate possono sostituire intersezione e differenza
- **Esempio**: cognome e nome dei professori ordinari che non hanno tesisti

```
SELECT cognome, nome
FROM Professori
WHERE qualifica='ordinario' AND
cod <>  ALL (SELECT  DISTINCT relatore
                FROM Studenti );
```

| relatore |
|---|
| FT |
| VC |

**<>ALL uguale a  NOT IN**

**=  ANY uguale a IN**

# INTERROGAZIONI NIDIFICATE

- **Metodologicamente**
  - i join si realizzano applicando i join
  - le operatori insiemistiche si realizzano applicando gli operatori insiemistici

- **Quando può servire la nidificazione**
  - nei sistemi DBMS in cui non c'è intersezione o differenza
  - condizioni nella WHERE su aggregati
    es: lo studente con la media più alta

# NIDIFICAZIONE, VISTE, POTERE ESPRESSIVO

- **Utilizzo nelle viste**
  - esprimere interrogazioni altrimenti inesprimibili

- **Esempio**: Studenti con la media più alta
  - per calcolare la media di ciascuno studente serve un raggruppamento
  - condizione nidificata sui gruppi
  - non è possibile nidificare la HAVING (nidificazione solo nella WHERE)

# NIDIFICAZIONE, VISTE, POTERE ESPRESSIVO

- Soluzione con le viste

```
CREATE VIEW StudentiConMedia AS
SELECT matr, cognome, nome, avg(voto) as media
FROM Esami JOIN Studenti ON studente=matr
GROUP BY matr, cognome, nome;
```

```
SELECT matr, cognome, nome
FROM StudentiConMedia
WHERE media = (SELECT max(media)
               FROM StudentiConMedia);
```

StudentiConMedia

| matr | cognome | nome | media |
|---|---|---|---|
| 111 | Rossi | Mario | 20,7 |
| 222 | Neri | Paolo | 24,5 |
| 333 | Rossi | Maria | 25,8 |
| 444 | Pinco | Palla | 19,6 |
| 77777 | Bruno | Pasquale | 26 |
| 88888 | Pinco | Pietro | 26 |

# NIDIFICAZIONE, VISTE, POTERE ESPRESSIVO

```
SELECT avg(count(cod))
FROM Professori
GROUP  BY facolta;
```

```
CREATE  VIEW Facolta AS
SELECT facolta, count(*) AS numdocenti
FROM Professori
GROUP  BY facolta;
```

```
SELECT avg(numdocenti)
FROM Facolta;
```

# QUANTIFICAZIONE

# LA QUANTIFICAZIONE

- È opportuno quantificare le interrogazioni associate alle associazioni multivalore
- Esempio: cercare “gli studenti che hanno preso 30” è ambiguo
- Più correttamente, l’interrogazione va formulata in uno dei seguenti modi
  - 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 (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**)

# COME ABBIAMO VISTO PRECEDENTEMENTE... LA SINTASSI DEL WHERE

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

# LA QUANTIFICAZIONE ESISTENZIALE: EXISTS

- Gli studenti con almeno un voto sopra 27. Serve un quantificatore
  $\exists e \in \text{Esami}(s)$: e.Voto > 27

- In SQL diventa:

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

# LA QUANTIFICAZIONE ESISTENZIALE USANDO GIUNZIONE E WHERE

- Stessa quantificazione esistenziale, tramite giunzione:

```
SELECT s.Nome
FROM Studenti s JOIN Esami e ON s.matricola=e.studente
WHERE e.Candidato = s.Matricola AND e.Voto > 27
```

# LA QUANTIFICAZIONE ESISTENZIALE USANDO ANY

- La solita query:

```
SELECT s.Nome
FROM Studenti s
WHERE EXISTS (  SELECT *
                FROM Esami e
                WHERE e.Candidato = 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.Candidato = s.Matricola  )
```

# LA QUANTIFICAZIONE ESISTENZIALE USANDO 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  )
```

# RIASSUMENDO

- La quantificazione esistenziale si fa con:
  - Exists (il più espressivo)
  - Giunzione
  - =Any, >Any, <Any…
  - IN

- =Any, >Any, <Any, IN,… non aggiungono potere espressivo, ma possono semplificare la scrittura delle query

- **ATTENZIONE**: non confondere esistenziale con universale!

# LA QUANTIFICAZIONE UNIVERSALE

- Gli studenti che hanno preso solo 30
- Errore comune (e grave):

```
SELECT s.Nome
FROM Studenti s, Esami e
WHERE e.Candidato = s.Matricola AND e.Voto = 30
```

# LA QUANTIFICAZIONE UNIVERSALE CON NOT EXISTS

- Osservazione: PER OGNI si traduce con un **NOT EXISTS** nel quale la condizione del **WHERE** è negata

```
SELECT s.Nome  FROM Studenti s
WHERE NOT EXISTS (SELECT *
                  FROM Esami e
                  WHERE e.Candidato = s.Matricola AND e.Voto <> 30)
```

# LA QUANTIFICAZIONE UNIVERSALE CON ALL

- Consideriamo la solita query (studenti con tutti 30)
- Poiché la condizione "*e.Voto* = *30*" è semplice, la possiamo esprimere con ALL:

```
SELECT s.Nome
FROM Studenti s
WHERE 30 = ALL (  SELECT e.Voto
                  FROM Esami e
                  WHERE e.Candidato = s.Matricola )
```

# LA QUANTIF. UNIVERSALE E GLI INSIEMI VUOTI

- Trovare gli studenti che hanno preso solo trenta:

```
SELECT s.Nome
    FROM Studenti s
    WHERE NOT EXISTS (SELECT *
                      FROM Esami e
                      WHERE e.Candidato = s.Matricola AND e.Voto <> 30)
```

- Perché trovo anche Rossi? Cosa cambia se invece di **NOT EXISTS** uso **<>ANY**, **=ALL**, oppure **NOT IN**?

| Nome | Matricola | Provincia | AnnoNascita |
|---|---|---|---|
| Bianco | 1 | PI | 1970 |
| Verdi | 2 | PI | 1980 |
| Rossi | 3 | PI | 1980 |

| Mater. | Candidato | Voto |
|---|---|---|
| RC | 1 | 30 |
| IS | 2 | 30 |
| RC | 2 | 20 |

# GLI INSIEMI VUOTI

- Lo studente senza esami non soddisfa nessuna interrogazione esistenziale, ma soddisfa tutte quelle universali
- Se voglio gli studenti che hanno preso solo trenta, e hanno superato qualche esame:

```
SELECT s.Nome
FROM Studenti s
WHERE NOT EXISTS (SELECT *
                  FROM Esami e
                  WHERE e.Candidato = s.Matricola AND e.Voto <> 30)
      AND EXISTS (SELECT *
                  FROM Esami e
                  WHERE e.Candidato = s.Matricola)
```

- (La combinazione di exists e for all si può fare anche con la GROUP BY:
  Studenti-Join-Esami GROUP BY s.Matricola, s.Nome HAVING min(e.Voto)=30)

# FINE DELLA LEZIONE