Basi di Dati
Modulo Laboratorio

Lezione 3: Interrogazioni SQL

DR. SARA MIGLIORINI

A.A. 2024-25

Comando SELECT

- Il processo/comando per recuperare i dati da una base di dati è chiamato interrogazione (query).
- In SQL, esiste solo un comando per interrogare un base di dati: SELECT.
- SELECT ha una sintassi che permette di specificare interrogazioni anche molto complesse e che possono richiedere anche ore di computazione.
- In questo corso si introduce il comando anche in formulazioni non semplici e si introdurranno anche delle pratiche per ottimizzare le interrogazioni.
- Come materiale didattico si consideri il cap. II.7 e VI.I-SELECT del manuale di PostgreSQL.

Comando SELECT: Sintassi

Sintassi SELECT semplificata

```
SELECT [ DISTINCT ]
[ * | expression [[ AS] output_name ] [, ...] ]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY grouping_element [, ...] ]
[ HAVING condition [, ...] ]
[ { UNION | INTERSECT | EXCEPT } [ DISTINCT ] other_select ]
[ ORDER BY expression [ ASC | DESC | USING operator ]]
```

Comando SELECT: Sintassi

- expression è un'espressione che determina un attributo.
- from_item è un'espressione che determina una sorgente per gli attributi.
- condition è un'espressione booleana per selezionare i valori degli attributi.
- grouping_element è un'espressione per poter eseguire operazioni su più valori di un attributo e considerare il risultato.

Comando SELECT: Scopo

L'esecuzione di una SELECT produce una relazione risultato che:

- ha come schema tutti gli attributi elencati nella clausola SELECT.
- ha come contenuto tutte le tuple t ottenute proiettando sugli attributi dopo SELECT, le tuple t' appartenenti al prodotto cartesiano delle tabelle ottenute dopo il FROM che soddisfano l'eventuale condizione nella clausola WHERE/HAVING/GROUP BY.

Clausola SELECT

Sintassi SELECT semplificata

```
SELECT [ DISTINCT ]
[{* | expression [[ AS] output_name ]} [, ...]]
```

- * è un'abbreviazione per indicare tutti gli attributi delle tabelle.
- expression è un'espressione che coinvolge gli attributi della tabella (può anche essere semplicemente il nome di un attributo).
- output_name è il nome assegnato all'attributo che conterrà il risultato della valutazione dell'espressione expression nella relazione risultato.
- DISTINCT: se presente richiede l'eliminazione delle tuple duplicate.

Clausola SELECT

Esempio (Visualizzare tutto il contenuto di una tabella)

Si assuma che la seguente tabella sia già stata definita e popolata:

Insegnamento		
codice	nome	num_crediti

Per visualizzare tutto il contenuto della tabella:

```
SELECT * FROM Insegnamento;
```

codice	nome	num_crediti
INF01	Lab Basi Dati 2	2
INF02	Analisi 1	6
INF03	Fisica 1	6

Clausola SELECT

Esempio (Visualizzare solo alcuni attributi di una tabella)

Si assuma che la seguente tabella sia già stata definita e popolata:

Studente					
matricola	nome	cognome	indirizzo	città	media

Per visualizzare solo una parte degli attributi della tabella:

```
SELECT matricola, cognome, nome FROM Studente;
```

matricola	cognome	nome
IN0001	Rossi	Marco
IN0002	Verdi	Paolo
IN0003	Bianchi	Dante

Clausola SELECT

Esempio (Visualizzare espressioni di attributi di una tabella)

Per visualizzare il contenuto della tabella concatenando attributi:

```
SELECT matricola,
    UPPER( cognome ) || ' ' || LOWER(nome) AS COGNOME_nome
FROM Studente;
```

matricola	COGNOME_nome
IN0001	ROSSI marco
IN0002	VERDI paolo
IN0003	BIANCHI dante

Clausola FROM

Sintassi

```
FROM from_item [, ...]
```

dove from_item può essere UNA delle seguenti clausole (versione semplificata):

1. table_name [[AS] alias [(column_alias [, ...])]]
   - Se sono presenti due o più tabelle, si esegue il prodotto cartesiano tra tutte le tabelle. Se ci sono attributi con lo stesso nome su tabelle diverse, nel SELECT questi attributi sono identificati anteponendo il nome della tabella e un '.': nomeTabella.nomeAttributo
2. (other_select) [AS] nomeRisultato [(column_alias [,...])]
   - è una SELECT innestata. Il risultato della SELECT interna è lo schema con nome nomeRisultato su cui fare la SELECT corrente.

Clausola FROM

3. from_item [NATURAL] join_type from_item [ON condition]
   - è la clausola JOIN.
   - Metodo più sofisticato di 1) che permette di selezionare un sottoinsieme del prodotto cartesiano di due o più tabelle. join_type specifica il tipo di join. I tipi principali sono riassunti nelle slide seguenti.
4. . . . opzioni più complesse che non vengono trattate in questo corso.

Clausola FROM

Esempio (Visualizzare parte del prodotto cartesiano Impiegato x Reparto)

Considerando le tabelle Impiegato e Reparto della lezione precedente,

```
SELECT I.cognome , R.telefono
FROM impiegato AS I, reparto AS R;
```

cognome	telefono
Rossi	02 8020000
Verdi	02 8020000
Rossi	02 8027900
Verdi	02 8027900

cognome è un attributo di Impiegato, telefono è un attributo di Reparto.

Nota

Questa query è senza ambiguità sui nomi: può essere scritta in modo più compatto.

```
SELECT cognome, telefono
FROM Impiegato, Reparto;
```

Clausola FROM

Esempio (Visualizzare parte del prodotto cartesiano Impiegato x Reparto)

Considerando le tabelle Impiegato e Reparto della lezione precedente,

```
SELECT I.cognome AS "Cognome", R.telefono AS TEL
FROM Impiegato AS I, Reparto AS R;
```

Cognome	tel
Rossi	02 8020000
Verdi	02 8020000
Rossi	02 8027900
Verdi	02 8027900

Clausola FROM: uso di ALIAS

- SQL consente di associare un nome alternativo (alias) alle tabelle che compaiono come argomento della clausola FROM.
  - table_name AS alias [(column_alias [, ...])]
- Ogni volta che si introduce un alias per una tabella si dichiara una variabile di tipo tupla che varia sul contenuto della tabella di cui è alias.

Clausola FROM

Esempio (Visualizzare tutte le coppie di Impiegato e la lunghezza del cognome del primo componente della coppia.)

```
SELECT I1.cognome, I2.cognome, CHAR_LENGTH (I1.cognome ) AS len
FROM Impiegato AS I1, Impiegato AS I2;
```

cognome	cognome	len
Rossi	Rossi	5
Rossi	Verdi	5
Verdi	Rossi	5
Verdi	Verdi	5

I nomi della tabella risultato sono esatti: quindi nella tabella risultato ci sono due colonne con stesso nome.

Clausola WHERE

Sintassi
`WHERE condition`

1. condition è un'espressione booleana combinando condizioni semplici C_i con gli operatori AND, OR, NOT.
2. C_i := expr [θ {expr | const}]

   dove

   - expr := espressione che contiene riferimenti agli attributi delle tabelle che compaiono nella clausola FROM.
   - $\theta \in \{=,<>,<,>,<=,>=\}$
   - const := valore dei domini di base.
3. Una riga soddisfa condition se l'espressione è vera quando valutata con i valori della riga.
4. Una riga che non soddisfa condition è scartata dal risultato finale.

Clausola WHERE

Esempio (Uso condizione semplice)

Si consideri la tabella dell'esempio 1. Per visualizzare solo gli insegnamenti di almeno 6 crediti, è sufficiente porre la condizione nella clausola WHERE.

```
SELECT nome_ins AS ins, numero_crediti AS crediti
FROM Insegnamento
WHERE numero_crediti >= 6;
```

ins	crediti
Analisi 1	6
Fisica 1	6

Clausola WHERE

Esempio (Visualizzare con più condizioni semplici)

Si assuma la tabella dell'esempio 2 così popolata:

Studente					
matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Clausola WHERE

Esempio (Visualizzare con più condizioni semplici)

Tutti gli studenti con cognome Rossi e che risiedono a Verona:

```
SELECT cognome, nome, città FROM Studente
WHERE cognome = 'Rossi' AND città = 'Verona';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Clausola WHERE

Esempio (Visualizzare con più condizioni semplici)

Tutti gli studenti con cognome Rossi e che risiedono a Verona:

```
SELECT cognome, nome, città FROM Studente
WHERE cognome = 'Rossi' AND città = 'Verona';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli			va	28.6666
IN0003	Bianch			NA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

cognome	nome	città
Rossi	Marco	Verona
Rossi	Matteo	Verona

Clausola WHERE

Esempio (Visualizzare con più condizioni semplici)

Tutti gli studenti con cognome Rossi o Bianchi e che risiedono a Verona:

```
SELECT cognome, nome, città FROM Studente
WHERE ( cognome = 'Rossi' OR cognome = 'Bianchi') AND città = 'Verona';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Clausola WHERE

Esempio (Visualizzare con più condizioni semplici)

Tutti gli studenti con cognome Rossi o Bianchi e che risiedono a Verona:

```
SELECT cognome, nome, città FROM Studente
WHERE ( cognome = 'Rossi' OR cognome = 'Bianchi') AND città = 'Verona';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli			va	28.6666
IN0003	Bianch			NA	18.345
IN0004	Rossi			na	24.567
IN0005	Verdi	Luca	via nuova	va	28.6666
IN0006	Nocasa	Caio			

cognome	nome	città
Rossi	Marco	Verona
Rossi	Matteo	Verona

La tupla `(Bianchi, Dante, VERONA)` non è selezionata perché `'VERONA' <> 'Verona'`.

Usare `UPPER()` e `LOWER()` aiuta a risolvere molti problemi di confronto tra stringhe.

Clausola WHERE

Esempio (Visualizzare tutte le coppie di Impiegato che hanno cognome diverso ma di uguale lunghezza)

Si considerando le tabelle della lezione 2.

```
SELECT I1.cognome, I2.cognome, CHAR_LENGTH (I1.cognome ) AS len
FROM Impiegato AS I1, Impiegato AS I2
WHERE I1.cognome <> I2.cognome AND
    CHAR_LENGTH (I1.cognome ) = CHAR_LENGTH (I2.cognome );
```

cognome	cognome	len
Rossi	Verdi	5
Verdi	Rossi	5

Clausola WHERE

Esempio (Visualizzare con il NOT)

Tutti gli studenti che NON abitano a Verona:

```
SELECT cognome, nome, città FROM Studente
WHERE NOT città = 'Verona';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Clausola WHERE

Esempio (Visualizzare con il NOT)

Tutti gli studenti che NON abitano a Verona:

```
SELECT cognome, nome, città FROM Studente
WHERE NOT città = 'Verona';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli			va	28.6666
IN0003	Bianch			NA	18.345
IN0004	Rossi			na	24.567
IN0005	Verdi				28.6666
IN0006	Nocasa	Caio			

cognome	nome	città
Paoli	Paolo	Padova
Bianchi	Dante	VERONA
Verdi	Luca	Va

Operatore LIKE

Nella clausola WHERE può apparire l'operatore LIKE per il confronto di stringhe. LIKE è un operatore di pattern matching. I pattern si costruiscono con i caratteri speciali _ e %:

- _ = 1 carattere qualsiasi.
- % = 0 o più caratteri qualsiasi.

Sintassi

```
WHERE attributo [NOT] LIKE 'pattern';
```

Clausola WHERE

Esempio

Tutti gli studenti di una città che inizia con 'V' e finisce con 'a'.

```
SELECT cognome, nome, città FROM Studente
WHERE città LIKE 'V%a';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Clausola WHERE

Esempio

Tutti gli studenti di una città che inizia con 'V' e finisce con 'a'.

```
SELECT cognome, nome, città FROM Studente
WHERE città LIKE 'V%a';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli			va	28.6666
IN0003	Bianch			NA	18.345
IN0004	Rossi			na	24.567
IN0005	Verdi				28.6666
IN0006	Nocasa	Caio			

cognome	nome	città
Rossi	Marco	Verona
Rossi	Matteo	Verona
Verdi	Luca	Va

Operatore SIMILAR TO

- L'operatore SIMILAR TO è un LIKE più espressivo che accetta, come pattern, un sottoinsieme delle espressioni regolari POSIX (versione SQL).
- Esempi di componenti di espressioni regolari:
  - _ = 1 carattere qualsiasi.
  - % = 0 o più caratteri qualsiasi.
  - * = ripetizione del precedente match 0 o più volte.
  - + = ripetizione del precedente match UNA o più volte.
  - {n,m} = ripetizione del precedente match almeno n e non più di m volte.
  - [...] = ... è un elenco di caratteri ammissibili

Operatore SIMILAR TO

Esempio

Studenti con cognome che inizia con 'A' o 'B', o 'D', o 'N' e finisce con 'a':

```
SELECT cognome, nome, città FROM Studente
WHERE cognome SIMILAR TO '[ABDN]{1}%a';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Operatore SIMILAR TO

Esempio

Studenti con cognome che inizia con 'A' o 'B', o 'D', o 'N' e finisce con 'a':

```
SELECT cognome, nome, città FROM Studente
WHERE cognome SIMILAR TO '[ABDN]{1}%a';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli				28.6666
IN0003	Bianch			NA	18.345
IN0004	Rossi				24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

cognome	nome	città
Nocasa	Caio	

Operatore BETWEEN

- Nella clausola WHERE può apparire l'operatore BETWEEN per testare l'appartenenza di un valore ad un intervallo.

Sintassi

```
WHERE attributo [NOT] BETWEEN valore_iniziale AND valore_finale;
```

Operatore BETWEEN

Esempio

Tutti gli studenti che hanno matricola tra 'IN0002' e 'IN0004'.

```
SELECT cognome, nome, matricola FROM Studente
WHERE matricola BETWEEN 'IN0002' AND 'IN0004';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Operatore BETWEEN

Esempio

Tutti gli studenti che hanno matricola tra 'IN0002' e 'IN0004'.

```
SELECT cognome, nome, matricola FROM Studente
WHERE matricola BETWEEN 'IN0002' AND 'IN0004';
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi				27.50
IN0002	Paoli				28.6666
IN0003	Bianch				18.345
IN0004	Rossi				24.567
IN0005	Verdi				28.6666
IN0006	Nocasa	Caio			

cognome	nome	matricola
Rossi	Marco	IN0001
Paoli	Paolo	IN0002
Bianchi	Dante	IN0003
Rossi	Matteo	IN0004

Operatore IN

- Nella clausola WHERE può apparire l'operatore [NOT] IN per testare l'appartenenza di un valore ad un insieme.

Sintassi

```
WHERE attributo [NOT] IN ( valore [, ...] );
```

Operatore IN

Esempio

Tutti gli studenti che hanno matricola nell'elenco 'IN0001', 'IN0003' e 'IN0005'.

```
SELECT cognome, nome, matricola FROM Studente
WHERE matricola IN('IN0001 ','IN0003 ','IN0005 ');
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Operatore IN

Esempio

Tutti gli studenti che hanno matricola nell'elenco 'IN0001', 'IN0003' e 'IN0005'.

```
SELECT cognome, nome, matricola FROM Studente
WHERE matricola IN('IN0001 ','IN0003 ','IN0005 ');
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi				27.50
IN0002	Paoli				28.6666
IN0003	Bianchi				18.345
IN0004	Rossi				24.567
IN0005	Verdi				28.6666
IN0006	Nocasa	Caio			

cognome	nome	matricola
Rossi	Marco	IN0001
Bianchi	Dante	IN0003
Verdi	Luca	IN0005

Operatore IS NULL

- Nella clausola WHERE può apparire l'operatore IS [NOT] NULL per testare se un valore è NOT KNOWN (NULL) o no.

Sintassi

```
WHERE attributo IS [ NOT ] NULL;
```

Nota

In SQL, NULL non è uguale a NULL.

NON SI PUÒ usare '=' o '<>' con il valore NULL!

Operatore IS NULL

Esempio

Tutti gli studenti che NON hanno una città.

```
SELECT cognome, nome, città FROM Studente
WHERE città IS NULL;
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Operatore IS NULL

Esempio

Tutti gli studenti che NON hanno una città.

```
SELECT cognome, nome, città FROM Studente
WHERE città IS NULL;
```

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli				28.6666
IN0003	Bianch			NA	18.345
IN0004	Rossi				24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

cognome	nome	città
Nocasa	Caio	

Operatore ORDER BY

- La clausola ORDER BY ordina le tuple del risultato in ordine rispetto agli attributi specificati,

Sintassi

```
ORDER BY attributo [{ ASC | DESC }] [, ... ];
```

- dove ASCendente è lo standard.

Operatore ORDER BY

Esempio

Tutti gli studenti in ordine decrescente rispetto al cognome e crescente (lessicografico) rispetto al nome.

```
SELECT cognome, nome FROM Studente
ORDER BY cognome DESC, nome ;
```

cognome	nome
Verdi	Luca
Rossi	Marco
Rossi	Matteo
Paoli	Paolo
Nocasa	Caio
Bianchi	Dante

Operatori di aggregazione

- Sono operatori che permettono di determinare un valore considerando i valori ottenuti da una SELECT.
- Due tipi principali:
  - COUNT.
  - MAX, MIN, AVG, SUM.
- Nella sezione 9.20 del manuale PostgreSQL c'è l'elenco di tutti gli operatori di aggregazioni implementati in PostgreSQL.
- Esempio: SELECT MAX(matricola) FROM studenti;
- Quando si usano gli operatori aggregati, dopo la SELECT non possono comparire espressioni che usano i valori presenti nelle singole tuple perché il risultato è sempre e solo una tupla.
  - Esempio: SELECT MAX(matricola), ~~città~~ FROM studenti; non è consentito.

Operatore di aggregazione COUNT

- Restituisce il numero di tuple significative nel risultato dell'interrogazione.

Sintassi

```
COUNT({ * | expr | ALL expr | DISTINCT expr }])
```

- dove expr è un'espressione che usa attributi e funzioni di attributi ma non operatori di aggregazione.
- Tre casi comuni:
  - COUNT(*) ritorna il numero di tuple nel risultato dell'interrogazione.
  - COUNT(expr) ritorna il numero di tuple in ciascuna delle quali il valore expr è non nullo.
  - COUNT(ALL expr) è un alias a COUNT(expr).
  - COUNT(DISTINCT expr) come con COUNT(expr) ma con l'ulteriore condizione che i valori di expr siano distinti.

Operatore di aggregazione COUNT

Esempio

Conta tutti gli studenti presenti in tabella.

```
SELECT COUNT(*) AS N
FROM Studente;
```

N
6

Conta tutte le città della tabella Studente.

```
SELECT COUNT(città) AS N
FROM Studente;
```

N
5

Conta tutte le città distinte della tabella Studente ignorando le maiuscole/minuscole.

```
SELECT COUNT(DISTINCT LOWER(città)) AS N
FROM Studente;
```

N
3

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Operatori di aggregazione SUM/MAX/MIN/AVG

- Determinano un valore numerico (SUM/AVG) o alfanumerico (MAX/MIN) considerando le tuple significative nel risultato dell'interrogazione.

Sintassi

```
op ([ expr | DISTINCT expr ])
```

- op = SUM | AVG | MAX | MIN;
- expr è un'espressione che usa uno o più attributi e funzioni di attributi.

Nota

Il significato di DISTINCT è uguale a quanto detto per COUNT.

Operatore di aggregazione COUNT

Esempio

Calcola la media delle medie degli studenti.

```
SELECT AVG(media)::DECIMAL(5,2)
FROM Studente;
```

N
25.55

Calcola la media delle medie distinte degli studenti..

```
SELECT AVG(DISTINCT media)::DECIMAL(5,2)
FROM Studente;
```

N
24.77

matricola	cognome	nome	indirizzo	città	media
IN0001	Rossi	Marco	via nobile	Verona	27.50
IN0002	Paoli	Paolo	via dritta	Padova	28.6666
IN0003	Bianchi	Dante	via storta	VERONA	18.345
IN0004	Rossi	Matteo	via nobile	Verona	24.567
IN0005	Verdi	Luca	via nuova	Va	28.6666
IN0006	Nocasa	Caio			

Nota

`expr::newType` converte il valore di `expr` a un valore del dominio `newType` (casting).
In SQL standard l'istruzione casting è: `CAST(expr AS newType)`. Il simbolo :: è un'abbreviazione di PostgreSQL.