



Politecnico  
di Torino

DBG  
MG

# Interrogazioni avanzate

---

Linguaggio SQL

# Linguaggio SQL: interrogazioni avanzate

---

- Tabelle derivate
- CTE
- Query spaziali
- Query JSON

# Tabelle derivate

- Definisce una tabella temporanea che può essere utilizzata per ulteriori operazioni di calcolo
- La tabella derivata
  - ha la struttura di una **SELECT**
  - è definita all'interno di una clausola **FROM**
  - può essere referenziata come una normale tabella
- La tabella derivata permette di
  - calcolare più livelli di aggregazione
  - formulare in modo equivalente le interrogazioni che richiedono la correlazione

# Calcolo di aggregati a due livelli (n.1)

---

- Trovare la media massima (conseguita da uno studente)

STUDENTE (Matricola, AnnoIscrizione)

ESAME-SUPERATO (Matricola, CodC, Data, Voto)

- Risoluzione in 2 passi
  - trovare la media per ogni studente
  - trovare il valore massimo della media

# Calcolo di aggregati a due livelli (n.1)

---

- Trovare la media massima (conseguita da uno studente)

STUDENTE (Matricola, AnnoIscrizione)

ESAME-SUPERATO (Matricola, CodC, Data, Voto)

Passo 1: trovare la media per ogni studente

```
SELECT Matricola, AVG(Voto) AS MediaStudente
FROM ESAME-SUPERATO
GROUP BY Matricola
```

# Calcolo di aggregati a due livelli (n.1)

- Trovare la media massima (conseguita da uno studente)

STUDENTE (Matricola, AnnoIscrizione)

ESAME-SUPERATO (Matricola, CodC, Data, Voto)

Passo 2: trovare il valore massimo della media

```
SELECT MAX(MediaStudente)
```

```
FROM (SELECT Matricola, AVG(Voto) AS MediaStudente  
FROM ESAME-SUPERATO  
GROUP BY Matricola) AS MEDIE;
```

*Tabella derivata*

# Calcolo di aggregati a due livelli (n.2)

---

- Per ogni anno di iscrizione, trovare la media massima (conseguita da uno studente)

STUDENTE (Matricola, AnnoIscrizione)

ESAME-SUPERATO (Matricola, CodC, Data, Voto)

- Risoluzione in 2 passi
  - trovare la media per ogni studente
  - raggruppare gli studenti per anno di iscrizione e calcolare la media massima

# Calcolo di aggregati a due livelli (n.2)

---

- Per ogni anno di iscrizione, trovare la media massima (conseguita da uno studente)

STUDENTE (Matricola, AnnoIscrizione)

ESAME-SUPERATO (Matricola, CodC, Data, Voto)

- Passo 1: trovare la media per ogni studente

```
(SELECT Matricola, AVG(Voto) AS MediaStudente  
FROM ESAME-SUPERATO  
GROUP BY Matricola) AS MEDIE
```



# Calcolo di aggregati a due livelli (n.2)

- Per ogni anno di iscrizione, trovare la media massima (conseguita da uno studente)

STUDENTE (Matricola, AnnoIscrizione)

ESAME-SUPERATO (Matricola, CodC, Data, Voto)

- Passo 2: raggruppare gli studenti per anno di iscrizione e calcolare la media massima

```
SELECT Anno Iscrizione, MAX(MediaStudente)
FROM STUDENTE,
```

```
(SELECT Matricola, AVG(Voto) AS MediaStudente
FROM ESAME-SUPERATO
GROUP BY Matricola) AS MEDIE
```

← *Tabella derivata*

```
WHERE STUDENTE.Matricola=MEDIE.Matricola
```

← *Condizione di join*

```
GROUP BY AnnoIscrizione
```

# Correlazione con tabella derivata

- Per ogni prodotto, trovare il codice del fornitore che ne fornisce la quantità massima

F (CodF, NomeF, NSoci, Sede)

P (CodP, NomeP, Colore, Taglia, Magazzino)

FP (CodP, CodF, Qta)

- Risoluzione in 2 passi
  - Calcolare la Qta massima fornita per ogni prodotto
  - Selezionare i fornitori che forniscono la Qta massima, prodotto per prodotto

# Correlazione con tabella derivata

- Per ogni prodotto, trovare il codice del fornitore che ne fornisce la quantità massima

F (CodF, NomeF, NSoci, Sede)

P (CodP, NomeP, Colore, Taglia, Magazzino)

FP (CodP, CodF, Qta)

- Passo 1: calcolare la Qta massima fornita per ogni prodotto

```
SELECT CodP, MAX(Qta) AS MQta
FROM FP
GROUP BY CodP
```

# Correlazione con tabella derivata

- Per ogni prodotto, trovare il codice del fornitore che ne fornisce la quantità massima

F (CodF, NomeF, NSoci, Sede)

P (CodP, NomeP, Colore, Taglia, Magazzino)

FP (CodP, CodF, Qta)

- Passo 2: selezionare i fornitori che forniscono la Qta massima, prodotto per prodotto

```
SELECT CodP, CodF
```

```
FROM FP,
```

```
(SELECT CodP, MAX(Qta) AS MQta  
FROM FP  
GROUP BY CodP) AS TMax
```

← *Tabella derivata*

```
WHERE FP.CodP = TMax.CodP
```

← *Condizione di join*

```
AND FP.Qta = TMax.MQta;
```

← *Correlazione*

# Common Table Expression

- Definisce una tabella temporanea che può essere utilizzata per ulteriori operazioni di calcolo
- La CTE
  - ha la struttura di una **SELECT**
  - è definita mediante la clausola **WITH**
  - può essere referenziata come una normale tabella
- La CTE è usata per
  - calcolare più livelli di aggregazione
  - formulare in modo equivalente le interrogazioni che richiedono la correlazione
- Riferimenti
  - a CTE *precedentemente* definite nella stessa clausola **WITH**
  - ricorsivo

# CTE vs Tabelle derivate

---

- La CTE è preferita quando
  - è necessario fare riferimento a una tabella derivata più volte in una singola query
  - è necessario eseguire lo stesso calcolo più volte in più parti della query
  - si vuole aumentare la leggibilità di query complesse

# Sintassi per definizione di CTE

WITH

cte\_1 [(campo\_A, ...)] AS

(CTE query 1)

{, cte\_X AS (CTE query X) }

SELECT campo\_A, campo\_B, ...  
FROM cte\_1

*Nome della CTE*

*Query associata alla CTE*

*Query*

# Calcolo di aggregati a due livelli (n.1)

---

- Trovare la media massima (conseguita da uno studente)

STUDENTE (Matricola, AnnoIscrizione)

ESAME-SUPERATO (Matricola, CodC, Data, Voto)

- Risoluzione in 2 passi
  - trovare la media per ogni studente
  - trovare il valore massimo della media



# Calcolo di aggregati a due livelli

---

- Trovare la media massima (conseguita da uno studente)

STUDENTE (Matricola, AnnoIscrizione)

ESAME-SUPERATO (Matricola, CodC, Data, Voto)

Passo 1: trovare la media per ogni studente

WITH MEDIE AS

(SELECT Matricola, AVG(Voto) AS MediaStudenti

FROM ESAME-SUPERATO

GROUP BY Matricola)

# Calcolo di aggregati a due livelli

---

- Trovare la media massima (conseguita da uno studente)

STUDENTE (Matricola, AnnoIscrizione)

ESAME-SUPERATO (Matricola, CodC, Data, Voto)

Passo 2: trovare il valore massimo della media

WITH MEDIE AS

(SELECT Matricola, AVG(Voto) AS MediaStudenti

FROM ESAME-SUPERATO

GROUP BY Matricola)

SELECT MAX(MediaStudenti)

FROM MEDIE;

# Calcolo di aggregati con granularità diversa

- Trovare tutte le compagnie aeree in cui il salario medio di tutti i piloti di quella compagnia è superiore alla media dei salari totali di tutti i piloti del database

PILOTI (CodP, Nome, Cognome, Compagnia, Salario)

- Risoluzione in 3 passi
  - trovare il salario medio per ogni compagnia
  - trovare il salario medio considerando tutti i piloti
  - trovare le compagnie con salario medio maggiore del salario medio globale

# Calcolo di aggregati con granularità diversa

---

- Passo 1: trovare il salario medio per ogni compagnia

```
WITH salarioMedioCompagnia AS  
  (SELECT Compagnia, AVG(Salario) AS AvgSalComp  
   FROM PILOTI  
   GROUP BY Compagnia)
```

# Calcolo di aggregati con granularità diversa

---

- Passo 2: trovare il salario medio del database

```
WITH salarioMedioCompagnia AS
    (SELECT Compagnia, AVG(Salario) AS AvgSalComp
     FROM PILOTI
     GROUP BY Compagnia),
mediaSalario AS
    (SELECT AVG(Salario) AS MediaSal
     FROM PILOTI )
```

# Calcolo di aggregati con granularità diversa

- Passo 3: trovare le compagnie con salario medio maggiore del salario medio globale

```
WITH salarioMedioCompagnia AS
```

```
(SELECT Compagnia, AVG(Salario) AS AvgSalComp
```

```
FROM PILOTI
```

```
GROUP BY Compagnia),
```

```
mediaSalario AS
```

```
(SELECT AVG(Salario) AS MediaSal
```

```
FROM PILOTI )
```

```
SELECT Compagnia
```

```
FROM salarioMedioCompagnia, mediaSalario
```

```
WHERE salarioMedioCompagnia. AvgSalComp >  
mediaSalario.MediaSal;
```

# CTE referenziate

---

- Considerando le distanze medie percorse per ciascuna città, calcolare la distanza massima percorsa per ciascuna provincia

CITTA (CodC, NomeC, Provincia)

AUTISTA (CodA, NomeA, Cognome, CodC)

CORSA\_GIORNALIERA (Data, CodA, Importo, Distanza)

- Risoluzione in 3 passi
  - calcolare la distanza percorsa per ogni città da ogni autista
  - calcolare la distanza media per ogni città
  - calcolare la distanza massima per provincia

# CTE referenziate

---

- Passo 1: calcolare la distanza percorsa per ogni città da ogni autista

WITH totDistanzaAutista AS

```
( SELECT SUM(Distanza) AS distanzaTot, CG.CodA, CG.CodC, NomeC, Provincia
FROM CORSA_GIORNALIERA CG, CITTA C, AUTISTA A
WHERE CG.CodA=A.CodA AND A.CodC=C.CodC
GROUP BY CG.CodA, CG.CodC, NomeC, Provincia )
```



# CTE referenziate

- Passo 2: calcolare la distanza media per ogni città

WITH totDistanzaAutista AS

```
(SELECT SUM(Distanza) AS distanzaTot, CG.CodA, CG.CodC, NomeC, Provincia  
FROM CORSA_GIORNALIERA CG, CITTA C, AUTISTA A  
WHERE CG.CodA=A.CodA AND A.CodC=C.CodC  
GROUP BY CG.CodA, CG.CodC, NomeC, Provincia),
```

distanzaMedia AS

```
( SELECT AVG(distanzaTot) AS avgDist, CodC, NomeC, Provincia  
FROM totDistanzaAutista  
GROUP BY CodC, NomeC, Provincia )
```

# CTE referenziate

- Passo 3: calcolare la distanza massima per provincia

WITH totDistanzaAutista AS

```
(SELECT SUM(Distanza) AS distanzaTot, CG.CodA, CG.CodC, NomeC, Provincia
FROM CORSA_GIORNALIERA CG, CITTA C, AUTISTA A
WHERE CG.CodA=A.CodA AND A.CodC=C.CodC
GROUP BY CG.CodA, CG.CodC, NomeC, Provincia),
```

distanzaMedia AS

```
( SELECT AVG(distanzaTot) AS avgDist, CodC, NomeC, Provincia
FROM totDistanzaAutista
GROUP BY CodC, NomeC, Provincia )
```

```
SELECT MAX(avgDist), Provincia
FROM distanzaMedia
GROUP BY Provincia
```

# Sintassi CTE ricorsive

WITH RECURSIVE

cte\_1 AS

*Nome della CTE*

(CTE query 1

*Query iniziale*

UNION ALL

CTE query 2

*Query ricorsiva*

)

SELECT \*

FROM cte\_1

# CTE ricorsive

- Per ciascun impiegato, trovare il boss e il livello nella gerarchia

IMPIEGATI (CodI, Nome, Cognome, BossID\*)

<u>CodI</u>	Nome	Cognome	BossId*
1	Domenic	Leaver	5
2	Cleveland	Hewins	1
3	Kakalina	Atherton	7
4	Roxanna	Fairlie	NULL
5	Hermie	Comsty	4
6	Pooh	Goss	8
7	Faulkner	Challiss	5

# CTE ricorsive

---

```
WITH RECURSIVE gerarchia AS (  
  SELECT  CodI, Nome, Cognome, BossID, 0 AS livello  
  FROM IMPIEGATI  
  WHERE  BossID IS NULL  
  
  UNION ALL  
  
  SELECT  I.CodI, I.Nome, I.Cognome, I.BossID, livello +1  
  FROM IMPIEGATI I, gerarchia G  
  WHERE I.BossID = G.CodI  
)  
  
SELECT  G.Nome, G.Cognome, I.Nome AS NomeBoss, I.Cognome AS CognomeBoss, livello  
FROM  gerarchia G LEFT JOIN IMPIEGATI I ON G.BossID= I.CodI  
ORDER BY livello;
```

# Query spaziali

- I dati spaziali possono essere rappresentati da geometrie diverse
  - Point
  - Polygon
  - Line,
  - ecc.
- MySQL fornisce funzioni per:
  - creare geometrie in vari formati (WKT, WKB, interno)
  - convertire geometrie tra diversi formati
  - accedere alle proprietà qualitative o quantitative di una geometria
  - descrivere delle relazioni tra due geometrie
  - creare nuove geometrie a partire da quelle esistenti

# Creazione geometrie (MySQL)

---

- **Point**(x, y)
  - costruisce un punto utilizzando le sue coordinate
- **LineString**(pt [, pt] ...)
  - costruisce una linea utilizzando i punti forniti (almeno 2)
- **Polygon**(ls [, ls] ...)
  - costruisce un poligono a partire da una serie di linee

```
INSERT INTO t1 (pt_col) VALUES(Point(1,2));
```

# Proprietà delle geometrie (MySQL)

- **ST\_Dimension(g)**
  - restituisce la dimensione intrinseca del valore geometrico g
  - la dimensione può essere -1, 0, 1 o 2
- **ST\_Envelope(g)**
  - restituisce il rettangolo di delimitazione minimo (MBR) per il valore geometrico g
  - il risultato viene restituito come valore di poligono definito dai punti d'angolo del rettangolo di delimitazione
- **ST\_GeometryType(g)**
  - restituisce una stringa che indica il nome del tipo di geometria di cui l'istanza di geometria g è membro



# Proprietà delle geometrie (MySQL)

- **ST\_X(p)**
  - restituisce il valore della coordinata X del Punto p
- **ST\_Y(p)**
  - restituisce il valore della coordinata Y del Punto p
- **ST\_Length(ls)**
  - restituisce la lunghezza di una Linea
- **ST\_Area(poly)**
  - restituisce l'area di un poligono
- **ST\_Centroid(poly)**
  - restituisce il centroide di un poligono

# Relazioni tra geometrie (MySQL)

- **ST\_Difference**(g1, g2)
  - restituisce una geometria che rappresenta la differenza dell'insieme di punti dei valori della geometria g1 e g2
- **ST\_Intersects**(g1, g2)
  - restituisce 1 o 0 per indicare se g1 interseca spazialmente g2
- **ST\_Distance\_Sphere**(g1, g2 [, radius])
  - restituisce la distanza sferica minima tra due punti e/o più punti su una sfera, in metri
  - l'argomento opzionale **radius** deve essere indicato in metri
    - se omesso, il raggio predefinito è 6.370.986 metri

```
SELECT ST_Distance_Sphere(ST_GeomFromText('POINT(0 0)'), ST_GeomFromText('POINT(180 0)));
```

RESULT
20015042.813723423

# Query JSON

- JSON, acronimo per JavaScript Object Notation, è un formato per lo scambio dei dati in applicazioni client-server
- Funzioni dati JSON dipendono dal DBMS utilizzato
- Funzioni dati JSON usate per
  - creare dati in formato JSON
  - cercare all'interno del JSON in base al path fornito
  - modificare campi del JSON

# Esempio di file JSON

```
{  
  nome: "Agriturismo Mario Bros",  
  indirizzo: {  
    via: "Via Idraulici",  
    numero: 1,  
    regno: "Funghetti",  
  },  
  recensioni: [  
    { testo: "Esperienza avventurosa",  
      timestamp: "2023-04-05T16:19:00",  
      voto: 5 }  
  ],  
  nRecensioni: 1,  
  tags: ["agriturismo", "natura"]  
}
```

*Chiave*

*Valore*

*Embedded JSON*

*Array*

# Creare JSON (MySQL)

- **JSON\_ARRAY**(target, candidate[, path])

- valuta un elenco di valori (eventualmente vuoto) e restituisce un array JSON contenente tali valori

```
SELECT JSON_ARRAY(1, "abc", NULL, TRUE, CURTIME()) AS RESULT;
```

RESULT
[1, "abc", null, true, "11:30:24.000000"]

- **JSON\_OBJECT**([key, val[, key, val] ...])

- valuta un elenco (eventualmente vuoto) di coppie chiave-valore e restituisce un oggetto JSON contenente tali coppie

```
SELECT JSON_OBJECT('id', 87, 'name', 'carrot') AS RESULT;
```

RESULT
{"id": 87, "name": "carrot"}

# Cercare all'interno del JSON (MySQL)

- **JSON\_CONTAINS**(target, candidate[, path])
  - restituisce 1 o 0
    - se un documento JSON *candidate* è contenuto nel documento JSON *target*
    - se il *candidate* si trova in un *path* specifico all'interno del documento *target*.
  - restituisce NULL
    - se qualsiasi argomento è NULL
    - se il *path* non identifica una sezione del documento *target*
  - notazione per path:
    - \$ : root del documento
    - dot notation per specificare il path (es. \$.a)
    - [i]: per accedere all'elemento i-esimo di un array
    - wildcard \* o \*\* (\$.\*)

```
SELECT JSON_CONTAINS('{"a": 1, "b": 2, "c": {"d": 4}}', '1', '$.a') AS RESULT;
```

RESULT
1

# Cercare all'interno del JSON (MySQL)

- **JSON\_EXTRACT**(json\_doc, path[, path])
  - restituisce i dati di un documento JSON nei path forniti come parametri
  - restituisce NULL se
    - qualsiasi argomento è NULL
    - nessun percorso individua un valore nel documento
- Alternativa:
  - usare l'operatore ->

```
SELECT c, JSON_EXTRACT(c, "$.id")
FROM jemp
WHERE JSON_EXTRACT(c, "$.id") > 1
ORDER BY JSON_EXTRACT(c, "$.name");
```

```
SELECT c, c->"$.id"
FROM jemp
WHERE c->"$.id" > 1
ORDER BY c->"$.name";
```

c	c->"\$.id"
{"id": "3", "name": "Barney"}	"3"
{"id": "4", "name": "Betty"}	"4"
{"id": "2", "name": "Wilma"}	"2"

# Modificare JSON (MySQL)

- **JSON\_ARRAY\_APPEND**(json\_doc, path, val[, path, val] ...)
  - appende i valori alla fine degli array indicati e restituisce il risultato

```
SELECT JSON_ARRAY_APPEND(['a', ['b', 'c'], 'd'], '$[1]', 1) AS RESULT;
```

RESULT
["a", ["b", "c", 1], "d"]

- **JSON\_INSERT**(json\_doc, path, val[, path, val] ...)
  - inserisce i valori nel documento JSON e restituisce il risultato

```
SELECT JSON_INSERT('{ "a": 1, "b": [2, 3]}', '$.a', 10, '$.c', '[true, false]') AS RESULT;
```

RESULT
{"a": 1, "b": [2, 3], "c": "[true, false]"}



# Modificare JSON (MySQL)

- **JSON\_SET**(json\_doc, path, val[, path, val] ...)
  - inserisce o aggiorna i valori del documento JSON e restituisce il risultato

```
SELECT JSON_SET('{ "a": 1, "b": [2, 3]}' '$.a', 10, '$.c', '[true, false]') AS RESULT;
```

RESULT
{"a": 10, "b": [2, 3], "c": "[true, false]"}

- **JSON\_REMOVE**(json\_doc, path, [, path] ...)
  - rimuove il path nel documento JSON e restituisce il risultato

```
SELECT JSON_REMOVE('["a", ["b", "c"], "d"]', '$[1]') AS RESULT;
```

RESULT
["a", "d"]