Appunti SQL DML/DQL - Guida Completa

Introduzione al Linguaggio SQL

SQL (Structured Query Language) è il linguaggio standard per la gestione di database relazionali.

Versioni principali:

- **SQL-86**: Costrutti base
- **SQL-89 (SQL1)**: Integrità referenziale
- **SQL-92 (SQL2)**: SQL Interattivo, sistema tipi
- **SQL:1999 (SQL3)**: Modello ad oggetti
- **SQL:2003**: SQL/JRT, SQL/XML
- **SQL:2006**: Estensione di SQL/XML
- **SQL:2008**: Lievi aggiunte

Componenti principali:

1. **DDL (Data Definition Language)**: Creazione e modifica dello schema
2. **DML/DQL (Data Manipulation/Query Language)**: Interrogazioni, inserimento, eliminazione e modifica dati

---

1. COMANDI DI INTERROGAZIONE BASE

SELECT

Comando principale per estrarre dati da una o più tabelle.

```
SELECT Nome, Cognome

FROM IMPIEGATI

WHERE Ufficio = 'A'
```

FROM

Specifica le tabelle da cui estrarre i dati. Con più tabelle si effettua il prodotto cartesiano.

```
SELECT *

FROM IMPIEGATI, SEDI
```

WHERE

Definisce le condizioni di selezione. Supporta operatori booleani (AND, OR, NOT).

```
SELECT Nome

FROM IMPIEGATI

WHERE Stipendio > 20000 AND Ufficio = 'B'
```

---

2. OPERATORI DI CONFRONTO

LIKE

Confronto di stringhe con pattern:

- `_` : un singolo carattere
- `%` : sequenza di caratteri

```
SELECT Codice

FROM IMPIEGATI

WHERE Nome LIKE 'M_R%O'
```

BETWEEN

Verifica se un valore è in un intervallo.

```
SELECT Nome

FROM IMPIEGATI

WHERE Stipendio BETWEEN 24000 AND 34000
```

IS NULL / IS NOT NULL

Verifica la presenza o assenza di valori NULL.

```
SELECT Nome

FROM IMPIEGATI

```

```
WHERE Stipendio IS NULL
```

---

3. ALIAS E MODIFICATORI

AS

Rinomina colonne o tabelle (alias).

```
SELECT Nome AS Name, Cognome AS LastName

FROM IMPIEGATI AS I
```

DISTINCT

Rimuove le righe duplicate dal risultato.

```
SELECT DISTINCT Nome

FROM IMPIEGATI

WHERE Stipendio > 20000
```

ORDER BY

Ordina i risultati in modo ascendente (ASC) o discendente (DESC).

```
SELECT *

FROM IMPIEGATI

WHERE Ufficio = 'A'

ORDER BY Stipendio DESC
```

LIMIT

Limita il numero di righe restituite.

```
SELECT *

FROM IMPIEGATI
```

```
LIMIT 10
```

---

4. OPERATORI AGGREGATI

COUNT

Conta il numero di righe o valori.

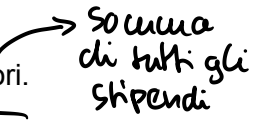
```
SELECT COUNT(*) AS Contatore

FROM STRUTTURATI

WHERE Dipartimento = 'Fisica'
```

SUM

Calcola la somma dei valori.




```
SELECT SUM(Stipendio) AS Totale

FROM STRUTTURATI

WHERE Dipartimento = 'Fisica'
```

AVG

Calcola la media dei valori.

```
SELECT AVG(Stipendio) AS StipendioMedio

FROM STRUTTURATI
```

MAX / MIN

Restituisce il valore massimo o minimo.

```
SELECT MAX(Stipendio) AS MaxStipendio

FROM STRUTTURATI

WHERE Tipo = 'Associato'
```

5. RAGGRUPPAMENTO

GROUP BY

Raggruppa le righe per applicare operatori aggregati a ciascun gruppo.

```
SELECT Dipartimento, COUNT(*) AS Numero

FROM STRUTTURATI

GROUP BY Dipartimento
```

HAVING

Filtra i gruppi creati da GROUP BY (applicato dopo il raggruppamento).

```
SELECT Dipartimento

FROM STRUTTURATI

GROUP BY Dipartimento

HAVING COUNT(*) >= 2
```

NOTA: WHERE filtra le righe prima del raggruppamento, HAVING filtra i gruppi dopo.

---

6. OPERAZIONI INSIEMISTICHE

UNION

Combina i risultati di due SELECT eliminando i duplicati.

```
SELECT Nome, Cognome FROM STRUTTURATI

UNION

SELECT Nome, Cognome FROM TECNICI
```

INTERSECT

Restituisce solo le righe presenti in entrambi i risultati.

```
SELECT Nome, Cognome FROM STRUTTURATI

INTERSECT

SELECT Nome, Cognome FROM TECNICI
```

EXCEPT

Restituisce le righe presenti nella prima SELECT ma non nella seconda.

```
SELECT Nome, Cognome FROM STRUTTURATI

EXCEPT

SELECT Nome, Cognome FROM TECNICI
```

---

7. JOIN

INNER JOIN (JOIN)

Combina righe di due tabelle basandosi su una condizione.

```
SELECT Modello

FROM GUIDATORI JOIN VEICOLI

ON GUIDATORI.NrPatente = VEICOLI.NrPatente

WHERE Nome = 'Sara'
```

LEFT JOIN

Tutte le righe della tabella di sinistra + corrispondenze a destra (NULL se assenti).

```
SELECT *

FROM GUIDATORI LEFT JOIN VEICOLI

ON GUIDATORI.NrPatente = VEICOLI.NrPatente
```

RIGHT JOIN

Tutte le righe della tabella di destra + corrispondenze a sinistra (NULL se assenti).

```
SELECT *

FROM GUIDATORI RIGHT JOIN VEICOLI

ON GUIDATORI.NrPatente = VEICOLI.NrPatente
```

FULL JOIN

Tutte le righe di entrambe le tabelle, con NULL dove manca corrispondenza.

```
SELECT *

FROM GUIDATORI FULL JOIN VEICOLI

ON GUIDATORI.NrPatente = VEICOLI.NrPatente
```

---

8. QUERY ANNIDATE

Le query annidate permettono di confrontare valori con i risultati di altre query.

Sintassi base:

```
SELECT Attributi

FROM Tabella

WHERE Attributo operatore (SELECT ... FROM ... WHERE ...)
```

Esempio - Stipendio massimo:

```
SELECT CODICE

FROM STRUTTURATI

WHERE STIPENDIO = (SELECT MAX(STIPENDIO) FROM STRUTTURATI)
```

ERRORE COMUNE: Non si possono mescolare operatori aggregati e colonne normali nella SELECT senza GROUP BY.

-- SBAGLIATO!

```
SELECT CODICE, MAX(STIPENDIO)

FROM STRUTTURATI
```

Operatori per query annidate:

ANY

La condizione è vera se il confronto è vero per ALMENO UNO dei valori restituiti.

```
SELECT NOME, COGNOME

FROM STRUTTURATI

WHERE DIPARTIMENTO = 'INFORMATICA'

  AND STIPENDIO = ANY (

    SELECT STIPENDIO

    FROM STRUTTURATI

    WHERE DIPARTIMENTO = 'FISICA'

  )
```

ALL

La condizione è vera se il confronto è vero per TUTTI i valori restituiti.

```
SELECT NOME, COGNOME

FROM STRUTTURATI

WHERE DIPARTIMENTO = 'INFORMATICA'

  AND STIPENDIO > ALL (

    SELECT STIPENDIO

    FROM STRUTTURATI

    WHERE DIPARTIMENTO = 'FISICA'

  )
```

IN

Verifica se un valore è contenuto nel risultato della query annidata.

```
SELECT NOME, COGNOME

FROM IMPIEGATI AS I

WHERE (I.NOME, I.COGNOME) IN (

  SELECT NOME, COGNOME

  FROM IMPIEGATI AS I2

  WHERE I.CODICE <> I2.CODICE

)
```

EXISTS

Restituisce TRUE se la query annidata restituisce almeno una riga.

```
SELECT NOME, COGNOME

FROM IMPIEGATI AS I

WHERE EXISTS (

  SELECT *

  FROM IMPIEGATI AS I2

  WHERE I.NOME = I2.NOME

    AND I.COGNOME = I2.COGNOME

    AND I.CODICE <> I2.CODICE

)
```

Tipi di query annidate:

1. **Semplici**: Nessun passaggio di binding tra contesti. Valutate dalla più interna alla più esterna.

2. **Complesse**: Passaggio di binding attraverso variabili condivise. Le query interne vengono valutate per ogni tupla esterna.

Esempio di query complessa (omonimi):

```
SELECT NOME, COGNOME

FROM IMPIEGATI AS I

WHERE (I.NOME, I.COGNOME) = ANY (

  SELECT NOME, COGNOME

  FROM IMPIEGATI AS I2

  WHERE I.NOME = I2.NOME

    AND I.COGNOME = I2.COGNOME

    AND I.CODICE <> I2.CODICE

)
```

Alternativa con SELF-JOIN:

```
SELECT NOME, COGNOME

FROM IMPIEGATI AS I, IMPIEGATI AS I2

WHERE I.NOME = I2.NOME

  AND I.COGNOME = I2.COGNOME

  AND I.CODICE <> I2.CODICE
```

---

9. VISTE (VIEWS)

Le viste sono "tabelle virtuali" ottenute da interrogazioni su altre tabelle.

Creazione:

```
CREATE VIEW NomeView [(ListaAttributi)]

AS SELECT ...

[WITH [LOCAL | CASCADE] CHECK OPTION]
```

Esempio:

```
CREATE VIEW STUDENTI(CODICE, NOME, COGNOME, DATANASCITA) AS

SELECT CODICE, NOME, COGNOME, NASCITA

FROM PROFESSORI
```

Utilizzi delle viste:

1. **Indipendenza logica/esterna**: Separare la struttura fisica da quella vista dagli utenti
2. **Semplificare query complesse**: Creare strutture intermedie riutilizzabili
3. **Retro-compatibilità**: Mantenere strutture obsolete dopo ristrutturazioni

Esempio pratico - Dipartimento con spesa massima:

-- STEP 1: Creare la vista

```
CREATE VIEW SPESEDIPARTIMENTI(NOMEDIP, SPESA) AS

SELECT DIPARTIMENTO, SUM(STIPENDIO)

FROM STRUTTURATI

GROUP BY DIPARTIMENTO
```

-- STEP 2: Usare la vista

```
SELECT NOMEDIP

FROM SPESEDIPARTIMENTI

WHERE SPESA = (SELECT MAX(SPESA) FROM SPESEDIPARTIMENTI)
```

Proprietà delle viste:

- I dati NON sono memorizzati fisicamente (tranne viste materializzate)
- Esistono solo a livello di schema
- L'aggiornamento è limitato e dipende dal DBMS

Aggiornabilità:

- Viste da una sola tabella: generalmente aggiornabili
- Viste da più tabelle: spesso NON aggiornabili

WITH CHECK OPTION:

Impedisce aggiornamenti che farebbero uscire le tuple dalla vista.

```
CREATE VIEW PROFESSORIRICCHI(CODICE, NOME, COGNOME, STIPENDIO) AS

SELECT CODICE, NOME, COGNOME, STIPENDIO

FROM PROFESSORI

WHERE STIPENDIO >= 30000

WITH CHECK OPTION
```

Se si prova a ridurre lo stipendio sotto 30000, l'operazione viene bloccata.

---

10. CTE (COMMON TABLE EXPRESSIONS)

Le CTE sono viste temporanee valide solo per una singola query.

Sintassi:

```
WITH NomeTabella(Attributi) AS (

  SELECT ...

)

SELECT ...

FROM NomeTabella

WHERE ...
```

Esempio:

```
WITH SPESEDIPARTIMENTI(NOMEDIP, SPESA) AS (

  SELECT DIPARTIMENTO, SUM(STIPENDIO)

  FROM STRUTTURATI

  GROUP BY DIPARTIMENTO
```

```
)

SELECT NOMEDIP

FROM SPESEDIPARTIMENTI

WHERE SPESA = (SELECT MAX(SPESA) FROM SPESEDIPARTIMENTI)
```

Differenza con VIEW: La CTE non esiste a livello di schema, è valida solo per la query corrente.

---

11. ASSERZIONI

Le asserzioni definiscono vincoli generici a livello di schema (SQL2).

Sintassi:

```
CREATE ASSERTION NomeAsserzione CHECK (Condizione)
```

Esempi:

Vincolo sul voto:

```
CREATE ASSERTION VotoValido

CHECK (Voto IS NOT NULL AND Voto >= 18 AND Voto <= 30)
```

Tabella non vuota:

```
CREATE ASSERTION TabellaValida

CHECK (1 <= (SELECT COUNT(*) FROM STUDENTI))
```

Limite sul salario:

```
CREATE ASSERTION SALARIO_CONTROLLO

CHECK (NOT EXISTS (

  SELECT *

  FROM IMPIEGATI
```

```
  WHERE SALARIO > 35000

))
```

NOTA: Le asserzioni possono essere immediate o differite (verificate al termine di una transazione).

---

12. COMANDI DML (MODIFICA DATI)

INSERT INTO

Inserisce nuove righe in una tabella.

```
INSERT INTO IMPIEGATI(Codice, Nome, Cognome, Ufficio)

VALUES ('8', 'Vittorio', 'Rossi', 'A')
```

Inserimento da SELECT:

```
INSERT INTO IMPIEGATI

SELECT * FROM NUOVI_ASSUNTI WHERE DataAssunzione = '2024-01-01'
```

DELETE

Cancella righe che soddisfano una condizione.

```
DELETE FROM IMPIEGATI

WHERE Ufficio = 'A'
```

ATTENZIONE: Senza WHERE cancella TUTTE le righe!

UPDATE

Modifica i valori di uno o più attributi.

```
UPDATE IMPIEGATI

SET Nome = 'Mario'
```

```
WHERE Codice = 5
```

Aggiornamento multiplo:

```
UPDATE IMPIEGATI

SET Stipendio = Stipendio * 1.1,

    Ufficio = 'B'

WHERE Dipartimento = 'Vendite'
```

---

13. ERRORI COMUNI E BEST PRACTICES

❌ Errore: Mescolare aggregati e colonne normali

-- SBAGLIATO!

```
SELECT CODICE, MAX(STIPENDIO)

FROM STRUTTURATI
```

✓ Corretto: Usare query annidate

```
SELECT CODICE

FROM STRUTTURATI

WHERE STIPENDIO = (SELECT MAX(STIPENDIO) FROM STRUTTURATI)
```

❌ Errore: Aggregati nella WHERE

-- SBAGLIATO!

```
SELECT CODICE

FROM STRUTTURATI

WHERE STIPENDIO = MAX(STIPENDIO)
```

✓ Corretto: Aggregati nella SELECT o in subquery

```
SELECT CODICE

FROM STRUTTURATI

WHERE STIPENDIO = (SELECT MAX(STIPENDIO) FROM STRUTTURATI)
```

Best Practices:

1. Usare alias per migliorare la leggibilità
2. Preferire JOIN espliciti al prodotto cartesiano nella FROM
3. Usare DISTINCT solo quando necessario (impatto sulle performance)
4. Ordinare sempre i risultati quando l'ordine è importante
5. Commentare query complesse
6. Testare query annidate separatamente prima di combinarle

---

14. ORDINE DI VALUTAZIONE SQL

1. **FROM**: Selezione tabelle e JOIN
2. **WHERE**: Filtro sulle righe
3. **GROUP BY**: Raggruppamento
4. **HAVING**: Filtro sui gruppi
5. **SELECT**: Proiezione colonne e aggregati
6. **DISTINCT**: Rimozione duplicati
7. **ORDER BY**: Ordinamento
8. **LIMIT**: Limitazione risultati

Select * from tabella where CHAR_LENGTH (nome) = 4

↓

LENGHT (solo per singoli byte)