Guida alla Programmazione in SQL

1. Cos'è SQL?

SQL (Structured Query Language) è il linguaggio standard per la gestione e manipolazione di database relazionali. Permette di definire, modificare, estrarre e gestire i dati attraverso un insieme di comandi standardizzati. SQL è utilizzato da vari sistemi di gestione di database (DBMS) come MySQL, PostgreSQL, Oracle, SQL Server e SQLite.

2. Installazione e Ambiente di Lavoro

Per utilizzare SQL, puoi installare un DBMS, come:

- **MySQL** o **PostgreSQL**: Sistemi open source, ampiamente usati.
- **SQL Server** di Microsoft o **Oracle Database**: Soluzioni commerciali.
- **SQLite**: Un database leggero che non richiede configurazione.

Molti DBMS forniscono interfacce grafiche come MySQL Workbench o pgAdmin per facilitare la gestione e la visualizzazione dei dati.

3. Concetti di Base nei Database Relazionali

Prima di iniziare, è importante comprendere alcuni concetti chiave:

- **Database**: Una raccolta organizzata di dati.
- **Tabelle**: Ogni database è costituito da tabelle che contengono i dati in righe (record) e colonne (attributi).
- **Record**: Una singola voce o riga in una tabella.
- **Colonna**: Un singolo attributo dei dati, come "nome" o "età".
- **Chiave Primaria (Primary Key)**: Una colonna che identifica univocamente ogni record in una tabella.
- **Chiave Esterna (Foreign Key)**: Una colonna che crea un collegamento tra due tabelle.

4. Creazione di una Tabella

La struttura di una tabella è definita con il comando `CREATE TABLE`. Ecco un esempio per creare una tabella `Studenti`:

sql
Copia codice

```sql
CREATE TABLE Studenti (
    ID INT PRIMARY KEY,
    Nome VARCHAR(50),
    Età INT,
    Classe VARCHAR(10)
```

```
);
```

- **ID**: Un numero intero che funge da chiave primaria.
- **Nome**: Una stringa fino a 50 caratteri.
- **Età**: Un numero intero per l'età dello studente.
- **Classe**: Una stringa che rappresenta la classe dello studente.

5. Inserimento di Dati: `INSERT INTO`

Per aggiungere dati alla tabella, si utilizza il comando `INSERT INTO`:

sql
Copia codice
```
INSERT INTO Studenti (ID, Nome, Età, Classe)
VALUES (1, 'Mario Rossi', 16, '4A');
```

Puoi inserire più record contemporaneamente:

sql
Copia codice
```
INSERT INTO Studenti (ID, Nome, Età, Classe)
VALUES
    (2, 'Luigi Bianchi', 17, '5B'),
    (3, 'Sara Verdi', 16, '4A');
```

6. Visualizzazione dei Dati: `SELECT`

Il comando `SELECT` recupera i dati da una tabella. Esempio:

sql
Copia codice
```
SELECT * FROM Studenti;
```

Questo mostra tutte le colonne e le righe della tabella `Studenti`. Per selezionare solo alcune colonne:

sql
Copia codice
```
SELECT Nome, Età FROM Studenti;
```

7. Condizioni di Selezione: `WHERE`

Il comando `WHERE` filtra i dati in base a una condizione:

sql
Copia codice
```
SELECT * FROM Studenti WHERE Età > 16;
```

8. Ordinamento dei Dati: `ORDER BY`

Usa `ORDER BY` per ordinare i risultati:

sql
Copia codice
```
SELECT * FROM Studenti ORDER BY Nome ASC;
```

- **ASC**: Ordine crescente (default).
- **DESC**: Ordine decrescente.

9. Aggiornamento dei Dati: `UPDATE`

Il comando `UPDATE` modifica i dati esistenti:

sql
Copia codice
```
UPDATE Studenti
SET Classe = '5A'
WHERE ID = 1;
```

10. Eliminazione dei Dati: `DELETE`

Per rimuovere dati, usa `DELETE`:

sql
Copia codice
```
DELETE FROM Studenti WHERE ID = 2;
```

⚠️ **Attenzione**: Se ometti `WHERE`, tutti i dati della tabella saranno cancellati!

11. Operatori e Condizioni

Alcuni operatori comuni in SQL:

- `=`: Uguale
- `<>` o `!=`: Diverso

- **`>`**, **`<`**, **`>=`**, **`<=`**: Maggiore, minore, ecc.
- **`BETWEEN`**: Seleziona un intervallo. Es. `WHERE Età BETWEEN 15 AND 18`
- **`LIKE`**: Cerca un pattern. Es. `WHERE Nome LIKE 'M%'` (nomi che iniziano con "M")
- **`IN`**: Seleziona valori specifici. Es. `WHERE Classe IN ('4A', '5A')`

12. Funzioni di Aggregazione

Le funzioni di aggregazione eseguono calcoli su gruppi di dati:

`COUNT()`: Conta il numero di record.
sql
Copia codice
```
SELECT COUNT(*) FROM Studenti;
```

-

`SUM()`: Somma dei valori.
sql
Copia codice
```
SELECT SUM(Età) FROM Studenti;
```

-

`AVG()`: Media dei valori.
sql
Copia codice
```
SELECT AVG(Età) FROM Studenti;
```

-

`MAX()` e **`MIN()`**: Massimo e minimo.
sql
Copia codice
```
SELECT MAX(Età), MIN(Età) FROM Studenti;
```

-

13. Raggruppamento dei Dati: `GROUP BY` e `HAVING`

`GROUP BY` raggruppa i dati per colonne specifiche. `HAVING` filtra gruppi dopo `GROUP BY`:

sql
Copia codice
```
SELECT Classe, COUNT(*)
FROM Studenti
GROUP BY Classe
HAVING COUNT(*) > 1;
```

14. Unione di Tabelle: JOIN

Le unioni permettono di combinare dati da più tabelle:

INNER JOIN: Restituisce le righe comuni a entrambe le tabelle.
sql
Copia codice

```
SELECT Studenti.Nome, Corsi.Corso
FROM Studenti
INNER JOIN Corsi ON Studenti.ID = Corsi.StudenteID;
```

-
- **LEFT JOIN**: Ritorna tutte le righe della prima tabella e le corrispondenti della seconda, o NULL se non corrispondono.
- **RIGHT JOIN**: L'opposto del LEFT JOIN.

15. Creazione di Vincoli

I vincoli garantiscono l'integrità dei dati:

- **PRIMARY KEY**: Identifica univocamente ogni record.
- **FOREIGN KEY**: Collega due tabelle.
- **NOT NULL**: Richiede un valore in una colonna.
- **UNIQUE**: Garantisce che i valori siano unici.

Esempio:

sql
Copia codice

```
CREATE TABLE Corsi (
    CorsoID INT PRIMARY KEY,
    Nome VARCHAR(50) NOT NULL,
    StudenteID INT,
    FOREIGN KEY (StudenteID) REFERENCES Studenti(ID)
);
```

16. Subquery

Una subquery è una query annidata in un'altra query:

sql
Copia codice

```
SELECT Nome FROM Studenti
WHERE Età = (SELECT MAX(Età) FROM Studenti);
```

17. Vista (`VIEW`)

Le viste sono query salvate che agiscono come tabelle temporanee:

sql
Copia codice
```sql
CREATE VIEW Studenti_Maggiorenni AS
SELECT * FROM Studenti WHERE Età >= 18;
```

18. Index

Gli indici velocizzano la ricerca dei dati in colonne specifiche:

sql
Copia codice
```sql
CREATE INDEX idx_età ON Studenti(Età);
```

19. Transazioni

Le transazioni permettono di eseguire più operazioni SQL come un'unità. Le transazioni sono gestite con i comandi `BEGIN`, `COMMIT` e `ROLLBACK`.

Esempio:

sql
Copia codice
```sql
BEGIN;

UPDATE Studenti SET Classe = '5A' WHERE ID = 1;

ROLLBACK; -- Annulla l'operazione
```

20. Conclusione

Questa guida copre i comandi e i concetti fondamentali di SQL. Conoscendo queste basi, puoi iniziare a esplorare funzioni più avanzate e sviluppare applicazioni più complesse usando SQL. Buon apprendimento!