

Prova a creare tabelle SQL (relazioni)

Ecco 10 tabelle di esempio che puoi creare per esercitarti con SQL.

Ogni tabella rappresenta un diverso contesto e ti aiuterà a sperimentare con la creazione di tabelle e relazioni.

1. Tabella `studenti`

```
CREATE TABLE studenti (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(50) NOT NULL,  
    cognome VARCHAR(50) NOT NULL,  
    data_nascita DATE,  
    email VARCHAR(100) UNIQUE  
);
```

2. Tabella `esami`

```
CREATE TABLE esami (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome_esame VARCHAR(100) NOT NULL,  
    crediti INT NOT NULL,  
    docente VARCHAR(100),  
    data_esame DATE  
);
```

3. Tabella libri

```
CREATE TABLE libri (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    titolo VARCHAR(150) NOT NULL,  
    autore VARCHAR(100),  
    genere VARCHAR(50),  
    prezzo DECIMAL(10, 2),  
    pubblicazione DATE  
);
```

4. Tabella `prodotti_informatici`

```
CREATE TABLE prodotti_informatici (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome_prodotto VARCHAR(100) NOT NULL,  
    categoria VARCHAR(50),  
    prezzo DECIMAL(10, 2),  
    disponibilita INT  
);
```

5. Tabella `ordini`

```
CREATE TABLE ordini (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    data_ordine DATE NOT NULL,  
    id_cliente INT,  
    totale DECIMAL(10, 2)  
);
```

6. Tabella `clienti`

```
CREATE TABLE clienti (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(50),  
    cognome VARCHAR(50),  
    email VARCHAR(100) UNIQUE,  
    telefono VARCHAR(20)  
);
```

7. Tabella `film`

```
CREATE TABLE film (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    titolo VARCHAR(100) NOT NULL,  
    regista VARCHAR(100),  
    anno_produzione YEAR,  
    genere VARCHAR(50),  
    durata INT COMMENT 'Durata in minuti'  
);
```


8. Tabella biglietti

```
CREATE TABLE biglietti (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    id_film INT NOT NULL,  
    data_proiezione DATETIME NOT NULL,  
    prezzo DECIMAL(10, 2),  
    posti_disponibili INT,  
    FOREIGN KEY (id_film) REFERENCES film(id)  
);
```

9. Tabella dipendenti

```
CREATE TABLE dipendenti (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(50),  
    cognome VARCHAR(50),  
    data_assunzione DATE,  
    stipendio DECIMAL(10, 2),  
    reparto VARCHAR(50)  
);
```

10. Tabella vendite

```
CREATE TABLE vendite (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    data_vendita DATE NOT NULL,  
    id_prodotto INT NOT NULL,  
    quantita INT NOT NULL,  
    totale DECIMAL(10, 2),  
    FOREIGN KEY (id_prodotto) REFERENCES prodotti_informativi(id)  
);
```

Caratteristiche e obiettivi

- **Relazioni:** Alcune tabelle, come `biglietti`, includono chiavi esterne per creare relazioni tra tabelle (`FOREIGN KEY`).
- **Tipi di dati:** Ogni tabella usa una varietà di tipi di dati (`VARCHAR` , `DECIMAL` , `DATE` , `INT`), offrendo un'ottima pratica per sperimentare.
- **Vincoli:** Ci sono vincoli come chiavi primarie (`PRIMARY KEY`), chiavi esterne (`FOREIGN KEY`), valori unici (`UNIQUE`) e campi obbligatori (`NOT NULL`).

Tipi di dato

Spiegazione dettagliata dei tipi di dato utilizzati nelle istruzioni SQL delle tabelle:

Tipi di dato usati

1. `INT` (Integer)

- **Descrizione:** Memorizza numeri interi.
- **Utilizzo:** Usato per identificatori (ID), quantità o qualsiasi altro dato numerico senza decimali.
- **Esempio:**
 - `id` in quasi tutte le tabelle è un `INT` perché serve come identificatore unico.
 - `quantita` nella tabella `vendite`.

2. `VARCHAR(n)` (Stringa a lunghezza variabile)

- **Descrizione:** Memorizza stringhe di caratteri con lunghezza variabile fino a un massimo di `n` caratteri.
- **Utilizzo:** Usato per nomi, titoli, descrizioni e testi non strutturati.
- **Esempio:**
 - `nome` e `cognome` nella tabella `studenti`.
 - `titolo` e `genere` nella tabella `film`.

3. **DECIMAL(p, s)** (Numero decimale a precisione fissa)

- **Descrizione:** Memorizza numeri con decimali. **p** è la precisione totale (numero di cifre totali), e **s** è il numero di cifre dopo il punto decimale.
- **Utilizzo:** Ideale per valori monetari o altre misure che richiedono precisione.
- **Esempio:**
 - `prezzo` in `prodotti_informativi` e `biglietti`.
 - `totale` in `vendite` e `ordini`.

4. **DATE** (Data)

- **Descrizione:** Memorizza una data nel formato `YYYY-MM-DD`.
- **Utilizzo:** Perfetto per rappresentare date specifiche come date di nascita, date di assunzione o date di vendita.
- **Esempio:**
 - `data_nascita` nella tabella `studenti`.
 - `data_assunzione` nella tabella `dipendenti`.

5. DATETIME (Data e ora)

- **Descrizione:** Memorizza una data e un'ora nel formato `YYYY-MM-DD HH:MM:SS`.
- **Utilizzo:** Usato quando è necessario registrare sia la data sia l'ora.
- **Esempio:**
 - `data_proiezione` nella tabella `biglietti`.

6. **YEAR** (Anno)

- **Descrizione:** Memorizza solo un anno, utile quando la precisione a livello di anno è sufficiente.
- **Utilizzo:** Usato per rappresentare un anno specifico, come l'anno di produzione.
- **Esempio:**
 - `anno_produzione` nella tabella `film` .

7. **UNIQUE** (Vincolo di unicità)

- **Descrizione:** Garantisce che un valore in una colonna (o combinazione di colonne) sia unico in tutta la tabella.
- **Utilizzo:** Usato per identificatori univoci che non sono chiavi primarie.
- **Esempio:**
 - `email` nelle tabelle `studenti` e `clienti`.

8. **AUTO_INCREMENT**

- **Descrizione:** Assegna automaticamente un valore intero crescente per ogni nuovo record.
- **Utilizzo:** Usato con le chiavi primarie per generare ID univoci automaticamente.
- **Esempio:**
 - `id` in quasi tutte le tabelle.

9. COMMENT

- **Descrizione:** Permette di aggiungere commenti descrittivi alle colonne per migliorarne la leggibilità.
- **Utilizzo:** Utile per documentare colonne con usi specifici.
- **Esempio:**
 - `durata INT COMMENT 'Durata in minuti'` nella tabella `film`.

Perché questi tipi di dato?

- **Efficienza:** Ogni tipo è scelto per ottimizzare lo spazio e la performance. Ad esempio, `DECIMAL` è più preciso rispetto a `FLOAT` per valori monetari.
- **Sicurezza:** Vincoli come `NOT NULL` e `UNIQUE` garantiscono integrità dei dati.
- **Compatibilità:** Questi tipi di dato sono ampiamente supportati nei database SQL (MySQL, PostgreSQL, SQL Server, ecc.).