Antonino Caserta

1) Qual è la differenza tra INNER JOIN, LEFT JOIN, RIGHT JOIN e FULL JOIN in SQL?

2) Cosa fa l'istruzione ORDER BY in SQL e come viene utilizzata?

3) Differenza tra un database relazionale e non relazionale.

4) Differenza tra primary key e foreign key.

5) Descrivi le operazioni di CRUD successivamente spiega il funzionamento della keyword WHERE

1

- **INNER JOIN** (o JOIN semplice): Restituisce le righe quando c'è almeno una corrispondenza in entrambe le tabelle. Se la condizione di join non è soddisfatta, le righe non vengono incluse nel risultato.

- **LEFT JOIN** (o LEFT OUTER JOIN): Restituisce tutte le righe dalla tabella di sinistra, e le righe corrispondenti dalla tabella di destra. Le righe dalla tabella di sinistra che non hanno corrispondenze nella tabella di destra vengono comunque incluse, ma con valori NULL nelle colonne della tabella di destra.

- **RIGHT JOIN** (o RIGHT OUTER JOIN): Analogamente al LEFT JOIN, ma restituisce tutte le righe dalla tabella di destra e le corrispondenti dalla tabella di sinistra. Le righe dalla tabella di destra che non hanno corrispondenze vengono incluse con valori NULL nelle colonne della tabella di sinistra.

- **FULL JOIN** (o FULL OUTER JOIN): Combina i risultati di LEFT JOIN e RIGHT JOIN. Restituisce tutte le righe quando c'è una corrispondenza in una delle tabelle. Le righe che non hanno corrispondenze in entrambe le tabelle vengono incluse con valori NULL nelle colonne dell'altra tabella.

2

L'istruzione **ORDER BY** viene utilizzata per ordinare i risultati di una query SQL in base al valore di una o più colonne. Può ordinare i dati in ordine ascendente (ASC) o discendente (DESC). Se non specificato, l'ordinamento è ASC per default. Esempio: `SELECT \* FROM utenti ORDER BY cognome ASC, nome DESC;` ordina prima per cognome in ordine ascendente e poi, per i cognomi uguali, ordina per nome in ordine discendente.

3

I database relazionali organizzano i dati in tabelle, con ogni tabella che rappresenta un tipo diverso di entità. Le tabelle sono composte da righe, che rappresentano le istanze delle entità, e colonne, che rappresentano gli attributi delle entità. Le relazioni tra le entità sono stabilite tramite chiavi primarie, che identificano univocamente ogni riga all'interno di una tabella, e chiavi esterne, che fanno riferimento a chiavi primarie in altre tabelle, permettendo di mantenere l'integrità referenziale tra i dati.

I database non relazionali, d'altro canto, non seguono questo schema fisso e possono archiviare dati in formati diversi. Questi database sono progettati per essere più flessibili rispetto ai database relazionali, permettendo una varietà di strutture dati più adatte a specifici tipi di applicazioni, come quelle che richiedono la scalabilità orizzontale o che lavorano con grandi volumi di dati strutturati in modo non convenzionale.

4

La **primary key** è un attributo o un insieme di attributi che identifica univocamente ogni riga in una tabella. Questa chiave è fondamentale per mantenere l'integrità dei dati all'interno della tabella, poiché impedisce l'esistenza di righe duplicate e garantisce che ogni riga sia facilmente identificabile e accessibile.

La **foreign key**, invece, è un attributo o un insieme di attributi in una tabella che fa riferimento alla primary key di un'altra tabella. Lo scopo della foreign key è stabilire e mantenere una relazione tra i dati in due tabelle, garantendo l'integrità referenziale all'interno del database. Questo significa che qualsiasi valore presente nella foreign key deve corrispondere a un valore esistente della primary key nella tabella a cui fa riferimento, o essere nullo se la relazione è opzionale.

5

Le operazioni di **CRUD** rappresentano il fondamento della gestione dei dati in un database, permettendo agli sviluppatori di creare, leggere, aggiornare ed eliminare i dati.

**Create**: Questa operazione permette di aggiungere nuovi dati al database. Si realizza attraverso comandi che inseriscono nuove righe nelle tabelle, specificando i valori per ogni colonna rilevante.

**Read**: L'operazione di lettura consente di consultare i dati presenti nel database. Si effettua principalmente attraverso query che selezionano dati da una o più tabelle, eventualmente filtrando e ordinando questi dati in base a specifici criteri.

**Update**: Modificare i dati esistenti è possibile attraverso l'operazione di aggiornamento, che permette di cambiare i valori degli attributi delle righe in una tabella basandosi su determinate condizioni.

**Delete**: Eliminare i dati dal database si effettua tramite l'operazione di cancellazione, che rimuove le righe selezionate da una tabella, seguendo specifiche condizioni.

La keyword **WHERE** ha un ruolo cruciale nelle operazioni di Read, Update e Delete, in quanto permette di specificare le condizioni che le righe devono soddisfare per essere incluse nell'operazione. Utilizzando WHERE, è possibile filtrare i dati in modo preciso, operando su attributi specifici e utilizzando operatori logici per definire condizioni complesse.