**Gestione della Connessione ai Database in Java**

Java offre la possibilità di collegarsi a vari tipi di database relazionali tramite delle classi e metodi specifici, che simulano un client. In sostanza, Java agisce come un client per inviare query e ricevere risposte dal database.

**Cos'è un Driver?**

Un *driver* è un piccolo software che fa da tramite tra Java e il DBMS (Database Management System). Java utilizza il driver per inviare e ricevere dati dal database, traducendo le query in un formato comprensibile per il DBMS e convertendo i risultati in formati utilizzabili da Java.

**File JAR**

Il file **JAR** (*Java Archive*) è un file compresso che contiene classi organizzate in package. All'interno del progetto, un JAR è necessario per includere i driver di connessione ai database. Ad esempio, per connettersi a MySQL si utilizza un JAR specifico per il driver MySQL.

**Passaggi per Connettersi a un Database**

1. **Caricare il Driver**  
   Anche se non si utilizza immediatamente, è importante collegare il driver al progetto. Questo si fa con il comando:

Class.forName("com.mysql.jdbc.Driver");

Bisogna specificare il nome completo della classe all'interno del JAR del driver.

1. **Aprire la Connessione**  
   Dopo aver caricato il driver, si può aprire la connessione con il database:

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/nomeDB", "user", "password");

​Dove nomeDB è il nome del database, user è il nome utente e password la password per l’accesso.

1. **Creare uno Statement o PreparedStatement**Gli **Statement** e **PreparedStatement** sono oggetti che permettono di inviare comandi SQL al database. Si distinguono per il modo in cui gestiscono i parametri:
   * Statement: esegue query SQL senza parametri specifici.
   * PreparedStatement: permette di precompilare una query con parametri, migliorando le prestazioni e la sicurezza contro SQL injection.

Esempio:

Statement stmt = con.createStatement();

1. **Inviare la Query e Ottenere un ResultSet**  
   Dopo aver creato uno statement, possiamo inviare una query e ottenere un **ResultSet**, che contiene il risultato sotto forma di elenco di record:

ResultSet rs = stmt.executeQuery("SELECT \* **FROM** tabella");

1. **Lettura dei Dati**  
   Il ResultSet restituisce i record ottenuti dalla query. È possibile leggere questi dati utilizzando il metodo next() che avanza tra i record uno alla volta

while (rs.next()) {

System.out.println(rs.getString("nome\_colonna"));}

​next() restituisce true finché esistono record nel ResultSet.

1. **Chiusura della Connessione**  
   È importante chiudere la connessione al termine dell'uso per liberare le risorse.

con.close();

​

**Concetti Addizionali**

**Commit** e **Rollback**:

quando si modificano i dati nel database, si possono utilizzare le operazioni commit per salvare definitivamente le modifiche e rollback per annullarle.

Carico e Bilanciamento: quando molte connessioni sono attive, i DBMS utilizzano il mirroring e il bilanciamento di carico per ottimizzare le prestazioni e rispondere rapidamente.

Prodotto Cartesiano: evitare il prodotto cartesiano nelle query, poiché può generare un numero enorme di risultati, rallentando le risposte del database.