1. Ogni volta che un utente chiama OPEN() su una connessione, il pooler cerca una connessione disponibile nel pool , Se è disponibile una connessione in pool, la restituisce al chiamante invece di aprire una nuova connessione. , Quando l'applicazione chiama CLOSE() sulla connessione, il pooler la restituisce al set di connessioni attive in pool anziché chiuderlo; Una volta che la connessione viene restituita al pool, è pronta per essere riutilizzata alla successiva chiamata Open
2. B
3. Modalità connessa: fornisce accesso di sola lettura (e forward-only) ai dati nell'origine dati e la possibilità di eseguire comandi sull'origine;

Principali classi utilizzate in Connected Mode: • Connection (SqlConnection) • Command (SqlCommand) • DataReader (SqlDataRe)

I passaggi per un corretto Connected Mode sono:

Aprire Connessione ; Creare Command ; Creare Paramentri se necessario ; Eseguire Command (DataReader/NonQuery/Scalar) ; Lettura Dati a schermo ;Chiudere Connessione

Modalità disconnessa: consente di manipolare i dati recuperati dall'origine dati •,successivamente riconciliarli con l'origine dati.

Le classi disconnesse forniscono un modo comune di lavorare con i dati disconnessi, indipendentemente dall'ambiente di origine data.

Principali classi utilizzate in Disconnected Mode: • DataSet • DataTable • DataColumn • DataRow • Constraint • DataRelation

1. Il DataAdapter fornisce il ponte tra l'oggetto DataSet e l'origine dati; utilizza oggetti Command per eseguire comandi SQL sull'origine dati per caricare sia DataSet con dati sia per riconciliare le modifiche apportate ai dati nel DataSet con l'origine dati La classe DataAdapter permette di collegare l'origine dati e le classi disconnesse tramite le classi connesse. Si utilizza quindi nella modalità disconnessa.
2. Query syntax:

From persona in Persone

Group persona by persona.Nome,

Where persona.eta>18;

From persona in Persone

Group persona by persona.cognome,

Where persona.eta>18;

Lambda syntax:

Var personeByName=persone.GroupBy(p=>p.nome).where(p.=>p.età>18);

Var personeByLastName=persone.GroupBy(p=>p.cognome).where(p.=>p.età>18);

6)La concorrenza pessimistica implica il blocco di righe nell'origine dati per impedire agli altri utenti di apportare modifiche ai dati che possano influire sugli utenti correnti. La concorrenza ottimistica considera la presenza di una violazione nel caso in cui, dopo che un utente riceve un valore dal database, tale valore viene modificato da un altro utente prima che il primo utente abbia effettuato un tentativo di modifica.