```
// interakcja z bazą danych…

conn.Close();          | Czyszczenie: zamknięcie
conn.Dispose();        | i zwolnienie połączenia
```

Listing 1.12 Połączenie z bazą danych wymaga przygotowania i zwolnienia

Oba procesy są zawsze identyczne, niezależnie od tego, czy czytamy z bazy danych, czy w niej zapisujemy, a także czy wykonujemy jedno czy więcej działań. Powyższy kod jest zwykle pisany za pomocą bloku `using` w następujący sposób:

```
using (var conn = new SqlConnection(connString))
{
     conn.Open();
     // interakcja z bazą danych…
}
```

Jest to krótsza i lepsza[9] metoda, ale w istocie niczym się nie różni od poprzedniej. Rozważmy kolejny przykład prostej klasy `DbLogger` z kilkoma metodami, które wchodzą w interakcję z bazą danych: `Log` wstawia podany komunikat dziennika, a `GetLogs` pobiera wszystkie wpisy dziennika od podanej daty.

```
using Dapper;                                  ← Udostępnia Execute i Query jako
// ...                                           metody rozszerzające połączenie

public class DbLogger
{
   string connString;                          ← Zakładamy, że jest to ustawiane
                                                 w konstruktorze
   public void Log(LogMessage msg)
   {
      using (var conn = new SqlConnection(connString))   ← Inicjalizacja
      {
      int affectedRows = conn.Execute("sp_create_log"     ← Czyszczenie jest
         , msg, commandType: CommandType.StoredProcedure);  wykonywane jako
      }                                                     część Dispose
   }

   public IEnumerable<LogMessage> GetLogs(DateTime since)
   {
      var sqlGetLogs = "SELECT * FROM [Logs] WHERE [Timestamp] > @since";
      using (var conn = new SqlConnection(connString))   ← Ustanawianie
      {
       return conn.Query<LogMessage>(sqlGetLogs           ← Wysyła zapytanie do bazy danych
            , new {since = since});                         i deserializuje wyniki
      }                                ← Czyszczenie
     }
   }
```

Zapisuje LogMessage w bazie danych

Listing 1.13 Duplikacja inicjacji/czyszczenia

[9] Jest to krótsze, gdyż Dispose zostanie wywołane przy wyjściu z bloku using, który z kolei wywoła Close. Tak jest lepiej, gdyż działanie zostanie opakowane w try/finally, więc połączenie zostanie zwolnione, nawet gdy w treści bloku using wystąpi wyjątek.