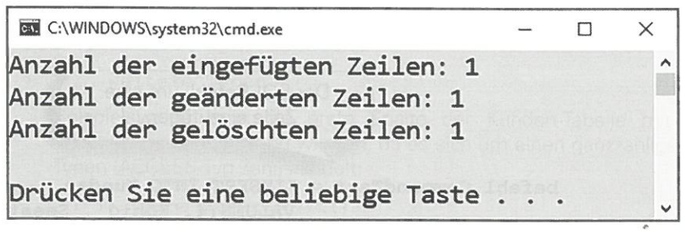
```
            anzahl = befehl.ExecuteNonQuery();
            Console.WriteLine("Anzahl der geänderten Zeilen: "
                              + anzahl);

                                  Ein DELETE-Befehl

            befehl.CommandText = "DELETE FROM Kunden
                                  WHERE name = 'Knudsen';";

            anzahl = befehl.ExecuteNonQuery();
            Console.WriteLine("Anzahl der gelöschten Zeilen: "
                              + anzahl);
            Console.WriteLine();

        }
        catch(Exception ausnahme)
        {
            Console.WriteLine("Datenbankfehler: "
                              + ausnahme.Message);
        }

                                  Schließen der Datenbank-Verbindung.

        finally
        {
            if (offen == true) dBVerbindung.Close();
        }
    }
}
```

Nach dem Starten werden die drei *Nicht-Select-SQL-Befehle* abgesetzt und die Anzahl der betroffenen Zeilen wird nach jedem Befehl ausgegeben:



Zum Vergleich: Die Kundentabelle vor und nach den SQL-Befehlen:

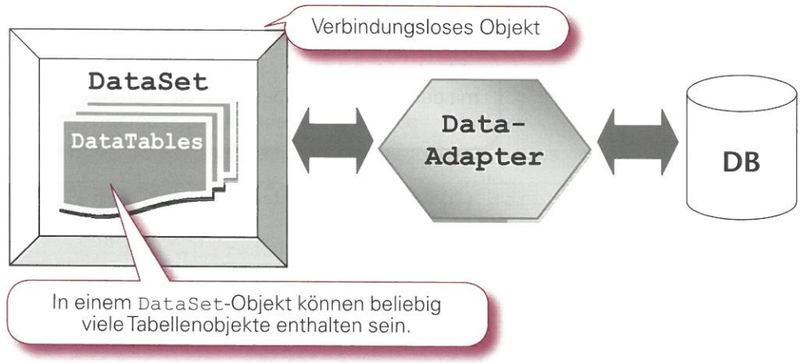
Vorher:

Nachher:

Der neue Datensatz erhält von ACCESS automatisch die ID 4. ACCESS achtet nicht auf lückenlose IDs nach dem Löschen und Einfügen von Datensätzen.

10.1.5 DataAdapter und DataSet

Bislang wurde die Datenbank geöffnet und sequenziell abgefragt oder über entsprechende SQL-Befehle modifiziert. Diese Vorgehensweise ist praktikabel, aber nicht sehr komfortabel. Aus diesem Grund gibt es die Möglichkeit, das komplette Ergebnis der Abfrage in einem speziellen Objekt der Klasse `DataSet` zu speichern. Dieses Objekt kann dann unabhängig von der Datenbank bearbeitet werden und erst im Anschluss findet eine Synchronisierung mit der Datenbank statt. Deshalb spricht man bei solchen Objekten auch davon, dass sie **verbindungslos** sind. Für den korrekten Austausch der Daten zwischen diesen Objekten und der Datenbank sorgen dann **Adapter-Objekte**. Die folgende Abbildung verdeutlicht diesen Zusammenhang:



Das folgende Beispiel zeigt die Verwendung von `DataSet` und `DataAdapter`:

```
using System;                          Einbinden der benötigten
using System.Data;                     Namensräume.
using System.Data.OleDb;


namespace DB_Zugriff_CSharp
{
    class CDBZugriff
    {                                  Den Verbindungsstring mit
                                       der Provider-Angabe und der
    static void Main(string[] args)    Datenquelle festlegen.
    {
        string verbindungsstring =
            "Provider=Microsoft.ACE.OLEDB.12.0;
             Data Source=C:\\Temp\\Kunden.accdb";
```