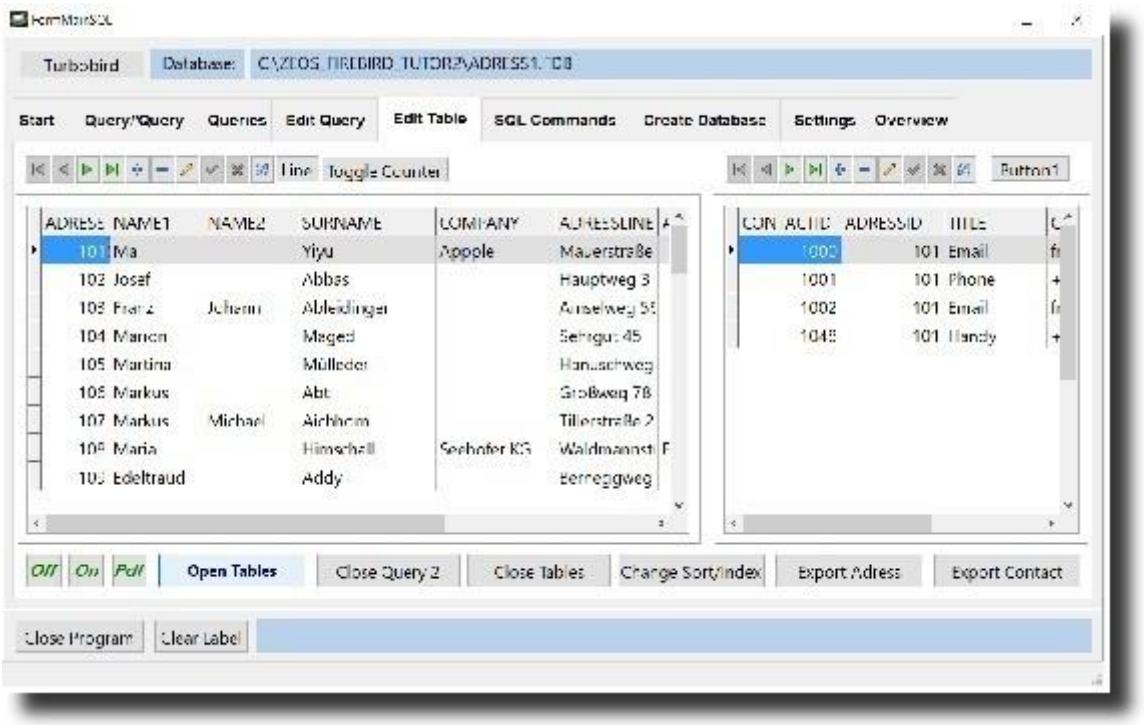
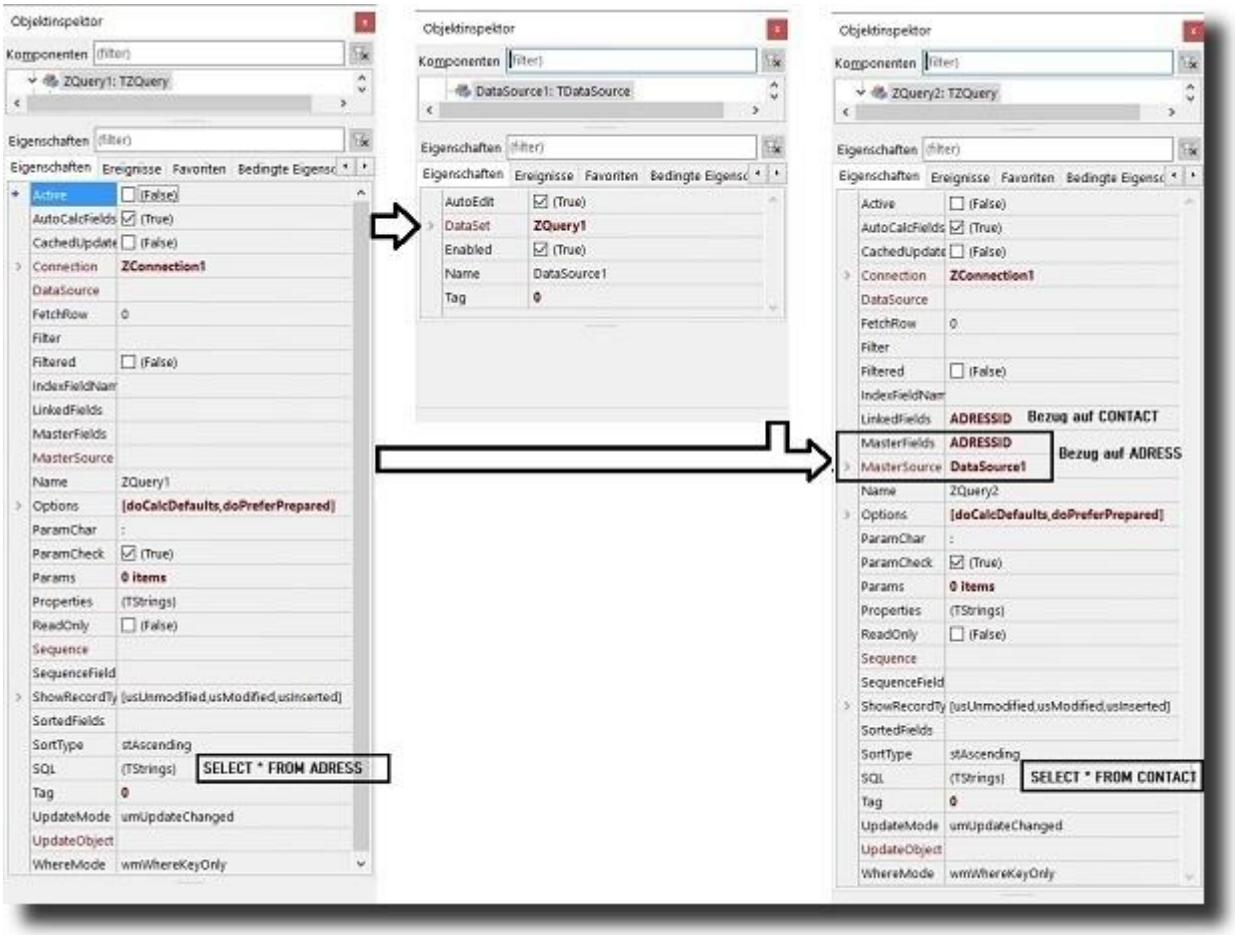
Edit Table



Edit Table zeigt zwei DBGrids mit Tabellen, welche über 2 Querys, ZQuery1 und ZQuery2 miteinander verbunden sind. In einem Query wird die Datenmenge über eine SQL Abfrage bestimmt. Bei ZQuery1 haben wir als Abfrage "SELECT * FROM ADRESS" in der Property SQL stehen. "ZQuery.SQL:= 'SELECT * FROM ADRESS';" würde demnach im Programmcode stehen. SQL ist eine Stringliste (TStrings), das heißt eine Tabelle mit hinter- bzw. untereinander verbundenen Strings (Zeichenketten). freepascal deutsch english

In den folgenden Beschreibungen zu den anderen Beispielen werden die Verknüpfungen nicht mehr weiter erklärt, da das Prinzip immer wieder dasselbe ist.



Das obige Bild (kann mit Klicken vergrößert werden) zeigt die Verknüpfung der beiden Queries. In ZQuery1 werden alle Adressen angezeigt, denn der SQL Befehl "SELECT * FROM ADRESS" übersetzt bedeutet "Zeige mir alle Felder der Tabelle ADRESS"., wobei "*" für "alle Felder" steht und da keine weitere Einschränkung bei den Daten erfolgt, werden auch alle Adressen angezeigt. Diese Abfrage, deren Ergebnis im DBGrid1 (links) angezeigt wird, greift nur auf eine einzelne Tabelle (ADRESS) zu und die Daten können deshalb bearbeitet werden.

Im Tutorial sind die unten angeführten Schritte schon erledigt, die Beschreibung dient der Erläuterung der Vorgangsweise.

Im DBGrid2 (rechts) wollen wir uns die Kommunikationsdaten der jeweiligen Person von ADRESS anzeigen lassen. Zuerst einmal müssen wir der zugeordneten ZQuery2 mitteilen, welche Daten wir abrufen wollen. Dies erledigen wir, indem wir der Eigenschaft ZQuery.SQL den SQL Befehl "SELECT * FROM CONTACT" zuweisen. Ohne weitere Schritte würde jetzt das DBGrid2 alle Kontaktdaten aller Kunden anzeigen. Deshalb müssen wir ZQuery2 noch mitteilen, dass wir nur jeweils die Kontaktdaten der links angezeigten Adresse sehen wollen. Als erstes weisen wir DataSource1 in DataSet den Wert ZQuery1 zu, der in der Propertyliste aus einer Liste der vorhandenen ZQueries ausgewählt werden kann. Dann müssen wir ZQuery2 noch mitteilen, welche Daten jeweils angezeigt werden sollen. Das erledigen wir in der Propertyliste von ZQuery2. Dort befinden sich drei Properties, die wir setzen müssen.

MasterSource: Hier wird eingestellt, von welcher Tabelle (ADRESS) wir die Verknüpfung beziehen. Dies geschieht, indem wir DataSource1, die mit ZQuery1 verbundene TDataSource in MasterSource einstellen. Damit weiß ZQuery2, von welcher Tabelle die Anzeige der Daten abhängig ist. Jetzt müssen wir noch einstellen, welches Feld für die Verknüpfung zuständig ist.

Masterfields: Dies können mehrere als auch nur ein Feld sein. Für diesen Zweck darf/dürfen das/die verwendete(n) Feld(er) in der "Master"-Tabelle nur (ein) eindeutige(s) Indexfeld(er) sein, das heißt, jeder Wert darf nur einmal vorkommen. In der Fachsprache nennt man das einen Primärindex. Wir verwenden in unserem Beispiel ADRESSID (ADRESS) und CONTACTID (CONTACT) als Primärindex und diese werden in der Datenbank als fortlaufende Nummer automatisch erzeugt. Wenn für jede Tabelle eine solche fortlaufende Nummer als Primärindex angelegt wird, ist die Arbeit mit der Datenbank um einiges leichter. Der Primärindex sollte auf jeden Fall kein Feld sein, mit dem der Anwender arbeitet.

LinkedFields: In diese Property geben wir ebenfalls "ADRESSID" ein. Dies ist jetzt jedoch nicht die ADRESSID von ADRESS, sondern von CONTACT. Immer, wenn ein neuer Kontakt eingegeben wird, wird dem Feld ADRESSID von CONTACT jetzt automatisch der Wert der ADRESSID von ADRESS zugewiesen. Auf diese Weise weiß die Datenbank, welche Kontakte sie welcher Adresse zuordnen soll. Da wir ADRESSID von CONTACT als bei Anlage der Datenbank "Foreign Key" gekennzeichnet haben ist es nicht möglich, dass in dieses Feld eine ID - in unserem Fall Nummer - für die keine Adresse existiert, erhält.

Mehr dazu: Wie eine Datenbank organisieren - Primär und Fremdschlüssel

Als Nächstes verbinden wir DataSource2 mit ZQuery2. Und zu guter Letzt DataSource1 und DataSource2 jeweils mit DBGrid1 und DBGrid2. Nun ist das
Gerüst zur Daten Anzeige und Bearbeitung fertig.

Ein Problem haben wir noch. Wenn wir Adressen einfügen wird die Adresse zwar in die Datenbank übernommen, aber der neue Datensatz steht nicht für die Erfassung von Kontaktdaten zur Verfügung. Da schreiben wir uns im Datenbankformular eine kleine Routine, mit der wir das möglich machen. Wir klicken bei ZQuery1 beim Objektinspektor bei "AfterPost" auf die drei Punkte. Es öffnet sich eine leere Procedure und in diese schreiben wir "QueryRefresh(DataSet as TZQuery);" Refresh heißt, dass die Daten wieder physikalisch vom Speichermedium eingelesen werden und jetzt sicher zur Verfügung stehen.

Die Procedure ist durch die Übernahme des Senders "***DataSet***: TDataSet" für die Ausführung des Befehls "QueryRefresh(***DataSet*** *as TZQuery*);" universell, das heißt, dieser Programmteil könnte für mehrere Queries verwendet werden.

Unit datamaintutor2, Zeile 179
```
procedure TDataDMTutor2.ZQuery1AfterPost(DataSet: TDataSet);
begin
  // die ausgeklammerten Programmteile habe ich weggelassen
  QueryRefresh(DataSet as TZQuery);
end;
```

Ein weiteres Problem taucht beim Löschen einer Adresse auf. Die Adresse kann erst gelöscht werden, wenn auch alle zugehörigen (mit foreign key) verbundenen Daten gelöscht worden sind. Das heißt, wir müssen erst alle zugehörigen Kontaktdaten löschen, bevor wir die

übergeordnete Adresse löschen können. Hier bietet sich BeforeDelete von ZQuery1 (ADRESS) an. Bevor die Adresse endgültig gelöscht wird, können wir hier in der Unit TDataDMTutor2 den Löschvorgang der Kontaktdaten unterbringen (Zeile 200). "for i:=0 to i-1 do begin" - i-1 deshalb, weil die Datensatznummerierung bei 0 und nicht bei 1 beginnt. Drei Datensätze sind deshalb mit 0,1,2 indexiert.

```
procedure TDataDMTutor2.ZQuery1BeforeDelete(DataSet: TDataSet);
var i: integer;
begin
  with (DataSet as TZQuery) do begin
    DisableControls; // Datenanzeige aus
    { Die Anzahl der Datensätze übernehmen }
    i:= ZQuery2.RecordCount; // RecordCount = Anzahl der Datensätze
    for i:=0 to i-1 do begin // Datensatzanzahl durchlaufen und löschen
      Delete; // einen Datensatz löschen
    end;
    EnableControls; // Datenazeige ein
  end;
end;
```

Unten in der vorletzten Zeile befinden sich einige Buttons. Zuerst die drei für die Beschreibung.

Open Tables
Mit diesem Button wird ZQuery1 und ZQuery2 geöffnet und die Daten am Bildschirm angezeigt.

Close Query 2
Schließt Query2 und die Daten werden im rechten DBGrid nicht mehr angezeigt. Ist gemacht, um zu zeigen, dass einzelne Tabellen geschlossen werden können und andere geöffnet bleiben.

Close Tables
Schließt alle zwei Queries. Mit Open Tables können diese wieder geöffnet werden.

Change Sort/Index
Wechselt bei der Adresse jeweils zwischen der Anzeigenreihenfolge mittels Index und den eingestellten Sortierfeldern. Bitte die Vorgangsweise/Arbeitsweise dem Sourcecode entnehmen.

Eport Adress, **Export Contact**
Die Daten werden jeweils für ADRESS oder CONTACT mittels der Kompoinente MyExPorter (Komponente TTexExporter) in eine lesbare Textdatei ausgegeben. Vorher wird noch um den Dateinamen gefragt.

Oberhalb der Anzeige der Daten haben wir auch einige Buttons.

Datenbank Navigator
Ganz links und ganz rechts befindet sich der Navigator für die Adressen und Kontakte. Einfach testen. Als einzigen Button möchte ich das X erwähnen. Es wird rot, wenn Daten geändert werden oder ein neuer Datensatz angelegt wird. Durch Klick auf dieses **X** wird die Änderung/Neuanlage rückgängig gemacht. Dies ist nützlich, wenn in einer nicht editierbaren Datenmenge zum editieren angefangen wurde. Ich habe hier bewusst bei allen Beispielen den Anwenderzugriff auf die Daten nicht gesperrt, damit zu sehen ist, was möglich ist.

Line
Zeigt, wie man eine ganze Linie anzeigen kann mit Sperre der Änderungsmöglichkeiten bei ADRESS.

Toggle Counter
Die ID-Spalten werden hier standardmäßig angezeigt, um die Arbeitsweise zu demonstrieren. Normalerweise, wenn ein Anwender mit der Datenbank arbeitet, würden die ID Spalten nicht angezeigt. Die Anzeige der IDs kann hier ausgeschaltet werden.

<>