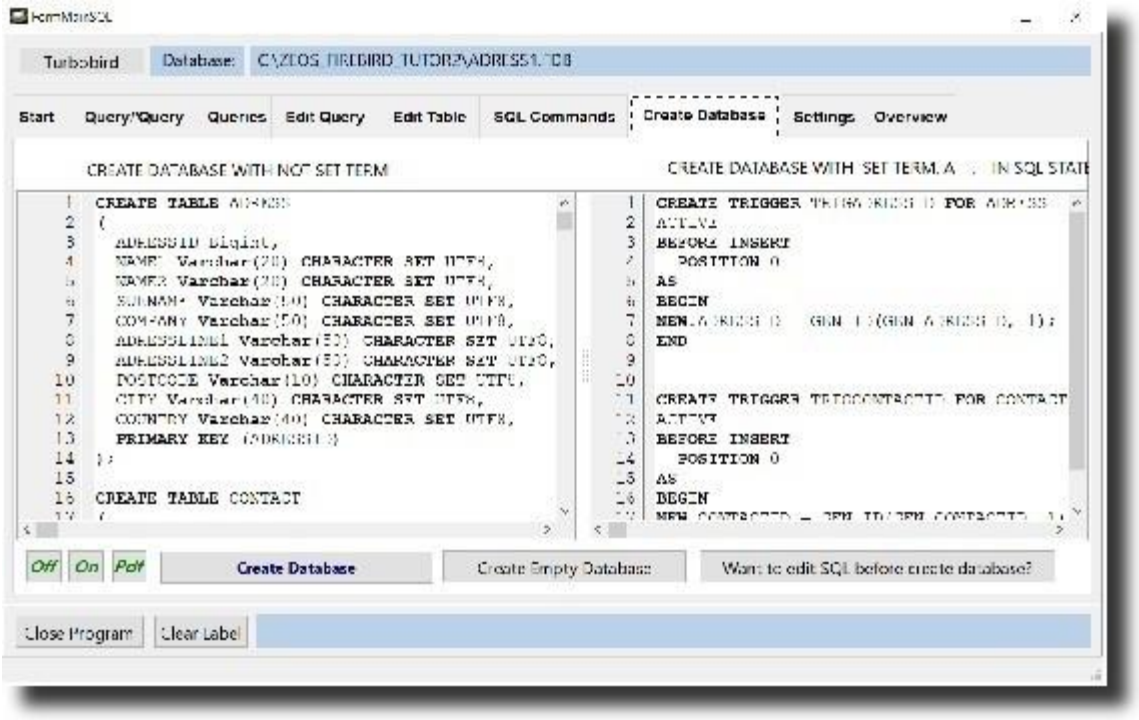
Create Database



Die Buttons in der vorletzten Zeile

OFF
Lädt eine RTF Datei mit Ihrem Standardeditor. Diese ist im Verzeichnis RTF gespeichert. Sie enthält diesen Text, welcher editierbar ist. Dies soll dazu dienen, wenn Sie mit dem Tutorial experimentieren, dass Sie sich Notizen machen können.

ON
Lädt die Webseite mit diesem Text. Dieser wird wahrscheinlich von Zeit zu Zeit ergänzt, je nach Fragen der User dieses Tutorials. Auf diese Weise ist immer eine aktuelle Version der Beschreibung verfügbar.

Pdf
Wird diesen Text im PDF Format enthalten, um diesen auch verfügbar zu haben, wenn man Offline ist.

Diese 3 Buttons sind Seiten-bezogen und bringen jeweils die für die Seite bereitgestellten Informationen auf den Bildschirm. Sie werden in weiterer Folge nicht mehr erwähnt.

Die Seite "https://www.w3schools.com/" bietet oben rechts in Form einer Weltkugel (neben der Lupe) viele Sprachen an, darunter auch deutsch. Es werden Links zu verschiedenen Seiten angeboten, um die Möglichkeit zu bieten, sich seine Lieblingsseite aussuchen zu können. W3schools bietet auf jeden Fall umfassende Erklärungen zu den SQL Statements.

Create Database
Erzeugt die bei Settings voreingestellte Datenbank. Die SQL Befehle zum Erstellen der Tabellen und um die Demodaten zu erzeugen werden in den 2 Editorfenstern "CREATE DATABASE WITH NOT SET TERM" und "CREATE DATABASE WITH "SET TERM ^;" IN SQL STATEMENT.

Im linken Editorfenster werden der Reihe nach die (Datensatz-)Tabelle "ADRESS" und "CONTACT" mit "CREATE TABLE ..." erstellt.

CREATE TABLE deutsch - CREATE TABLE english

Danach folgt "ALTER TABLE CONTACT ...". In diesem SQL Befehl wird die Verknüpfung der Adressdaten ("ADRESS") mit den Kontaktdaten ("CONTACT") vorgenommen. "FOREIGN KEY" sorgt in diesem Fall dafür, dass es bei den Kontaktdaten keine Datensatz gibt, welcher nicht einer Adresse zugeordnet ist.

FOREIGN KEY deutsch - FOREIGN KEY english (compact)

Dateneule: Primär- und Fremdschlüssel

Als Nächstes werden die zwei Befehle zum Erzeugen der Generatoren - das sind Aktivitäten, die direkt in der Datenbank ausgeführt werden, ausgeführt. "CREATE "GENERATOR GEN_ADRESSID;" erstellt den Generator zum Generieren des fortlaufenden Zählers für die Tabelle "ADRESS", der folgende für die Tabelle "CONTACT". Außerdem wird dem Generator der Anfangswert mit "SET GENERATOR" zugewiesen, bei den Adressen 143, bei den Kontaktdaten 1084.

CREATE GENERATOR deutsch - CREATE GENERATOR english

Als Nächstes werden mit "INSERT INTO ADRESS" und "INSERT INTO CONTACT" die Demodaten in die Datenbank geschrieben.

INSERT INTO deutsch - INSERT INTO english

Damit wären die SQL Befehle des linken Textfensters abgearbeitet. Es folgen nun die SQL Befehle des rechten Fensters. Wichtig ist auch, immer darauf zu achten, dass benötigte Werte immer schon vorhanden sind. So kann man keinen Index für eine Tabelle erstellen, die noch nicht vorhanden ist. Es werden hier zwei TRIGGER erstellt, die einen neuen Datensatz mit einem Wert für die ID-Spalte versorgen und diesen vorher jeweils um 1 hochzählen. Für diese SQL Befehle müssen wir vorher Adjust Delimiter auf "dtSetTerm" setzen. Dies wird hier im Programmcode erledigt, bzw. die SQL Befehle im rechten Editorfenster werden immer im Modus dtSetTerm ausgeführt, im linken Fenster mit dtDefault.

Hier müssen wir ein bisschen weiter ausholen. In SQL und ISQL (Interbase-SQL) gibt es die Möglichkeit den Standard-Delimiter ";" mittels des Befehls "SET TERM" zu ändern. Notwendig ist dies bei einer SQL-Sequenz wie unserer:

SET TERM ^ ;
CREATE TRIGGER TRIGADRESSID FOR ADRESS
ACTIVE

```
BEFORE INSERT
  POSITION 0
AS
BEGIN
NEW.ADRESSID = GEN_ID(GEN_ADRESSID, 1);
END ^
SET TERM ; ^
```

GEN_ID(GEN_ADRESSID, 1) ; END ^
Das Semikolon hier ist Teil des SQL-Befehls und nicht das Ende des SQL-Befehls. Normalerweise würde die Erstellung des TRIGGERS hier abgebrochen. Durch "SET TERM ^ ;" wird der SQL-Ausführungsroutine mitgeteilt, dass der Standard-Delimiter (Trennzeichen zwischen zwei SQL-Statements/Befehlen) von ";" auf "^" ausgetauscht ist. Das heißt, dass das Ende von *"CREATE PROCEDURE* .. " dass dem "*END*" folgende "^" ist statt des normalerweise verwendeten Semikolons. Der Befehl am Ende "SET TERM ; ^" macht das rückgängig, das heißt der Standard-Delimiter ist jetzt wieder ";".

Das wäre kein großes Problem, aber Firebird hat in seinem SQL "SET TERM" nicht implementiert. Die gängigen Datenbankverwaltungsprogramme haben das intern im Programm gelöst. Das hilft uns hier aber nicht weiter. So haben die Entwickler von Zeos in die Komponente TSQLProcessor die Möglichkeit eingebaut den Standard-Delimiter zu ändern. Es gibt 5 Möglichkeiten zum Einstellen: Default, dtDelimiter, dtEmptyLine, dtGo und dtSetTerm. "SET TERM" wird in diesem Fall im SQL Befehl nicht verwendet.

CREATE TRIGGER deutsch - CREATE TRIGGER englisch

Für uns für unsere Aufgaben sind 2 Einstellungen relevant:

dtDefault: wenn im SQL-Befehl intern *kein Semikolon* verwendet wird
dtSetTerm: wenn intern im SQL-Befehl *ein Semikolon* verwendet wird

Create Empty Database
Erzeigt eine neue Datenbank ohne Inhalte (keine Tabellen und Daten).

Want to edit SQL before create database?
Die beiden Textfenster sind standardmäßig nicht editierbar. Wenn Sie den Button klicken, dann können Sie die SQL Befehle editieren.

Zum Abschluss dieser Seite

Alles Weitere zur Datenbank Erstellung ist im Tutorial "Firebird, Lazarus & Zeos I" schon erläutert worden. Hier geht es vorrangig um die Arbeit mit den Datenbank Komponenten und um ein bisschen Basiswissen über SQL. Grundlegende SQL Kenntnisse sind Voraussetzung um eine Datenbankapplikation mit Firebird, unabhängig von Programmiersprache und den verwendeten Tools, erstellen zu können.

<>