22.14 Datenbanken – Exkurs Alternativen

Wenn Sie nur wenige Daten verwalten wollen und Ihnen MySQL oder PostgreSQL mit den notwendigen Datenbank-Servern zu aufwendig sind, dann lohnt ein Blick auf leichtgewichtige Alternativen. Das trifft auch für SQLite zu, deren Datenbanken und die in ihnen verankerten DB-Tabellen in einem proprietären Format vorliegen, das Sie ohne spezielle Programme nicht lesen können. Auf einige Features wie Multi-User-Betrieb müssen Sie dann aber verzichten. In diesem Kapitel werden Ihnen einige Alternativen vorgestellt.

22.14.1 Zenity

Das Programm Zenity (https://help.gnome.org/users/zenity/stable/index.html.de) ermöglicht die Erzeugung von grafischen Dialogen mit einfachen Konsolen-Kommandos. Mit dem vorgestellten Dialog-Programm können Sie zum Beispiel auf einfache Art Ihre Kontakte pflegen:

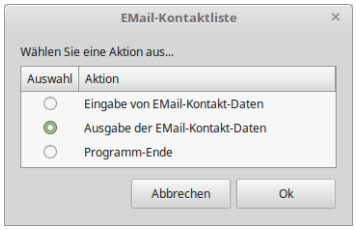

Abbildung 22.14.1.1: Oberfläche (GUI)

Dazu benötigen Sie nur ein Bash-Skript, das Sie nach Ihren Vorstellungen ändern können:

```
#!/bin/bash

#  Geändert: 24.04.2021
# Datum-Format: ISO 8601 yyyy-mm-dd
function eingabe() {
zenity --forms \
  --title="EMail-Kontakt hinzufügen" \
  --text="Geben Sie die geforderten Daten in das Formular ein!" \
       --separator="," \
       --add-entry="Vorname:" \
       --add-entry="Nachname:" \
       --add-entry="EMail-Adresse:" \
       --add-calendar="Geburtstag:" \
       --forms-date-format="%Y-%m-%d" >> liste.csv
}

function ausgabe() {
awk -F "," '{print $1"\n"$2,"\n"$3,"\n"$4}' liste.csv | zenity \
  --width=600 \
  --height=300 \
  --title="EMail-Kontaktliste" \
  --text="Wählen Sie einen Datensatz aus..." \
  --list \
  --column="Vorname" \
  --column="Name" \
  --column="EMail-Adresse" \
  --column="Geburtsdatum"  \
  --separator="," \
  --print-column=all
}

while true
do
menu="$(zenity --width=360 \
       --height=200 \
       --title="EMail-Kontaktliste" \
       --text="Wählen Sie eine Aktion aus..." \
       --list \
       --radiolist \
       --column="Auswahl" \
       --column="Aktion" \
       FALSE "Eingabe von EMail-Kontakt-Daten" \
       TRUE "Ausgabe der EMail-Kontakt-Daten" \
       FALSE "Programm-Ende")"
  if [ "$menu" = "Eingabe von EMail-Kontakt-Daten" ]; then
         eingabe
  elif [ "$menu" = "Ausgabe der EMail-Kontakt-Daten" ]; then
    ausgabe
  elif [ "$menu" = "Programm-Ende" ]; then
    exit
  fi
done
```

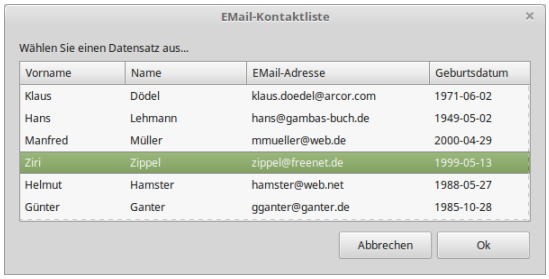

Abbildung 22.14.1.2: Ausgabe der Kontaktdaten

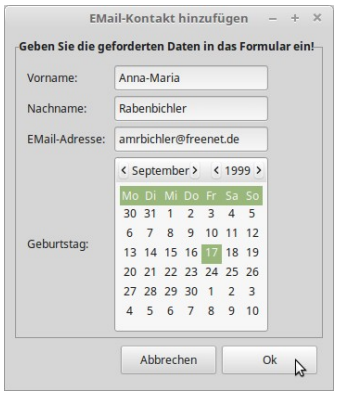

Abbildung 22.14.1.3: Eingabe der Kontaktdaten

Alle eingegebenen Daten werden in einer Text-Datei (liste.csv) gespeichert. Im Download-Bereich finden Sie neben der Datenbasis liste.csv auch die Programm-Datei liste.sh.

22.14.2 Datenbank mit Struktur

Im Kapitel '7.2.2 Projekt mit Strukturen' wird Ihnen ein Projekt vorgestellt, das die Datensätze intern in einer Struktur speichert. Sie können in den Datensätzen navigieren, Daten hinzufügen, Daten ändern und anzeigen. Als Datenbasis dient eine Datei, in der die Daten gambas-spezifisch serialisiert abgespeichert werden, die Sie auch aus der Datei wieder auslesen können:

Abbildung 22.14.2.1: Oberfläche der Datenverwaltung (GUI)

22.14.3 Datenbank mit einer Text-Datei als Datenbasis

Im Download-Bereich finden Sie das Projekt-Archiv db.textfile.tar.gz für die Verwaltung von Daten.

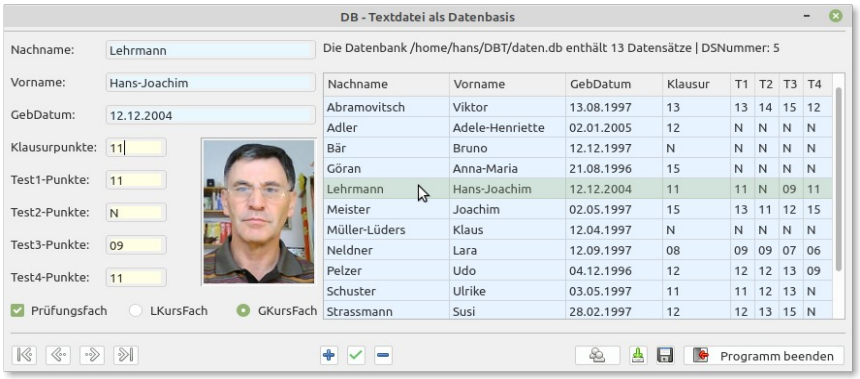

Abbildung 22.14.3.1: Oberfläche der Datenverwaltung (GUI)

Sie können auch in diesem Gambas-Programm in den Datensätzen navigieren, Daten hinzufügen, Daten ändern und anzeigen. Als Datenbasis dient jedoch eine einfache Text-Datei, in der die Datensätze zeilenweise abgespeichert werden, die Sie auch aus dieser Datei wieder auslesen können.

22.14.4 Datenbanksystem Recutils

Recutils implementiert Datenbanken, die auf reinen Textdateien basieren. Sie verwalten umfangreiche Datensätze in der Datenbank mit wenigen Shell-Kommandos. Einen guten Einstieg in Recutils finden Sie in den folgenden Artikeln im Netz oder im Hilfesystem:

- https://www.gnu.org/software/recutils/
- https://www.gnu.org/software/recutils/manual/
- https://www.linux-community.de/ausgaben/linuxuser/2012/01/textdatei-basiertes-datenbanksystem-recutils/.
- Konsole: $ info recutils

Recutils installieren Sie auf Ihrem System – wenn es nicht bereits installiert ist – am einfachsten über die Anwendungsverwaltung. Alternativ geben Sie in einem Terminal folgende Zeilen ein:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install recutils
```

Im nächsten Abschnitt wird demonstriert, wie Sie

- eine Recutils-Datenbank anlegen,
- ein Record-Set definieren, wobei ein Record-Set einer DB-Tabelle mit DB-Feldern entspricht,
- sich die Definition eines Record-Sets ansehen,
- DB-Daten in ein Record-Set eingeben,
- DB-Daten aus einem Record-Set auslesen,
- DB-Daten in einem Record-Set ändern,
- DB-Daten in einer csv-Datei speichern (Dump),
- DB-Daten aus einer csv-Datei in ein Record-Set einlesen und
- einen DB-Report in einem selbst definierten Format (→ Template) erzeugen.

Für administrative Aufgaben wie das Anlegen einer Recutils-Datenbank oder die Definition von Record-Sets (DB-Tabellen) benötigen Sie nur einen Text-Editor Ihrer Wahl. Alle anderen Arbeiten werden mit speziellen Konsolen-Programmen wie *recsel* oder *rec2csv* aus den Recutils-Tools erledigt.

(1)
Anlegen einer Recutils-Datenbank mit einem Record-Set (Record-Set in Recutils = DB-Tabelle)

Geben Sie den folgenden Text in einen Text-Editor wie xed unter Mint ein. Speichern Sie den Text in der Textdatei liste.rec. Nach dem Abspeichern besitzen Sie bereits eine Recutils-Datenbank in der Datei liste.rec mit einer (leeren) Tabelle mit dem Tabellennamen *Liste*!

```
%rec: Liste ' Tabellenname
%doc: Kontakt-Liste (Vorname, Nachname, EMail-Adresse und Geburtsdatum) ' Tabellenbeschreibung
%mandatory: Vorname Nachname      ' Diese Felder dürfen nicht leer bleiben
%key: Id                          ' Schlüssel-Definition mit dem Schlüsselnamen Id
%type: Id int                     ' Der Datentyp des Feldes Id ist Integer – vordefinierter Typ 'int'
%auto: Id                         ' Der Wert von Id wird automatisch erhöht
%type: Vorname line               ' Der Datentyp des Feldes Vorname ist Text – vordefinierter Typ 'line'
%type: Nachname line              ' Der Datentyp des Feldes Nachname ist Text
%typedef: EMail_t email           ' Typ-Definition
%type: EMail-Adresse EMail_t      ' Der Datentyp des Feldes EMail ist 'email' (vordefinierter Typ)
%type: Geburtstag date            ' Der Datentyp des Feldes Geburtstag ist 'date' (vordefinierter Typ)
# ENDE TYP-DEFINITIONEN           ' Kommentare beginnen nur auf einer separaten Zeile mit #
```

(2)
Sehen Sie sich die Definition der Tabelle in der Datenbank mit dem Konsolen-Programm *recinf* an:

```
$ recinf -d liste.rec ' Als Parameter benötigen Sie -d oder --descriptor
%rec: Liste
%doc: Kontakt-Liste (Vorname, Nachname, EMail-Adresse und Geburtsdatum)
%mandatory: Vorname Nachname
%key: Id
%type: Id int
%auto: Id
%type: Vorname line
%type: Nachname line
%typedef: EMail_t email
%type: EMail EMail_t
%type: Geburtstag date
```

(3)
Prüfen Sie, wie viele Datensätze in der DB-Tabelle 'Liste' in der Datenbank gespeichert sind:

```
$ recinf liste.rec
0 Liste
```

Kein Datensatz vorhanden – was Sie nicht erschüttern wird. Aber das wird jetzt geändert!

(4)
Einen Datensatz in die Datenbank-Tabelle einfügen

Die Syntax ist einfach, um einen neuen Datensatz mit dem Konsolen-Programm *recins* komplett einzufügen. Achtung: Das Feld `Id` als Zeichensatzzähler wird p.d. automatisch eingetragen! Es gilt: -f = field, -v = value und -r = record, alternativ auch als Kombination aus f und v mit einem Doppelpunkt als Zuweisungsoperator wie in Variante 2 beim Geburtsdatum genutzt:

```
$ recins -t Liste -f "Vorname" -v "Anna" -f "Nachname" -v "Geier" -f "EMail" -v "anna.g@gmail.com" -f "Geb-
Datum" -v "1999-01-04" liste.rec
```

Variante 1:

```
$ recins -t Liste \
         -f "Vorname" -v "Anna" \
         -f "Nachname" -v "Geier" \
         -f "EMail" -v "anna.g@gmail.com" \
         -f "GebDatum" -v "1999-01-04" \
         liste.rec
```

Variante 2:

```
$ recins -t Liste -f "Vorname" -v "Anna" -f "Nachname" -v "Geier" -f "EMail" -v "anna.g@gmail.com" -r "Geb-
Datum:1999-01-04" liste.rec
```

(4)
Datensätze aus der DB-Tabelle *Liste* ausgeben

```
$ recsel -t Liste liste.rec
Id: 0
Vorname: Anna
Nachname: Geier
EMail: anna.g@gmail.com
GebDatum: 1999-01-04
```

Es werden weitere Datensätze eingegeben und dann wieder alle Datensätze ausgegeben:

```
$ recsel -t Liste liste.rec
Id: 0
Vorname: Anna
Nachname: Geier
EMailAdresse: anna.g@gmail.com
Geburtstag: 1999-01-04

Id: 1
Vorname: Hans
Nachname: Lehmann
EMailAdresse: hans@gambas-buch.de
Geburtstag: 1949-05-02
...
Id: 11
Vorname: Klaus
Nachname: Dödel
EMailAdresse: klaus.doedel@arcor.com
Geburtstag: 1971-06-02
```

(5.1)
Datensätze werden nur mit ausgewählten Feldern ausgegeben

```
$ recsel -t Liste -p Vorname,Nachname liste.rec
```

(5.2)
Datensätze werden mit ausgewählten Eigenschaften – auch über ~ 'RegEx' – ausgegeben

```
$ recsel -t Liste --expression "Vorname ~ 'H'" liste.rec     ' Vorname beginnt mit 'H'
$ recsel -t Liste -e "Vorname ~ 'Ha'" liste.rec              ' Vorname beginnt mit 'Ha'
$ recsel -t Liste -e "Nachname ~ 'F|N'" liste.rec            ' Nachname beginnt mit 'F' oder mit 'N'
$ recsel -t Liste -e "GebDatum >> '1997-01-01'" liste.rec    ' Vergleich mit << oder == oder >>
$ recsel -t Liste -e "Nachname ~ 'F|N' && GebDatum >> '1950-01-01'" liste.rec ' Logischer Vergleich mit AND
```

(5.3)
Datensätze sortiert ausgegeben – hier nach den Nachnamen

```
$ recsel -t Liste --sort="Nachname" liste.rec
```

(6)
Änderung des Inhaltes eines Feldes in einem ausgewählten Datensatz

```
$ recset -t Liste -e "Nachname='Geier'" -f "EMail" -s "anna.maria.geier@web.de" liste.rec
```

(7)
Einen bestimmten Datensatz komplett löschen

```
$ recdel -t Liste -e "Nachname='Geier'" liste.rec
```

(8)
Export der DB-Daten der Tabelle *Liste* in eine csv-Datei *liste.csv*

```
$ rec2csv -d "," -t Liste liste.rec > liste.csv ' Der Delimiter "," ist der Standard
```

(9)
Import von Datensätzen aus einer csv-Datei in eine Datenbank-Datei mit definiertem Header. Achtung: Alle vorhandenen Datensätze werden überschrieben!

```
$ csv2rec liste.csv > liste.rec
```

(10)
Für einen DB-Report sind zwei Schritte notwendig. Im ersten Schritt definieren Sie ein Template als Report-Format-Muster. Tragen Sie die folgenden drei Zeilen in eine Textdatei *liste.template* ein. Sie können das Muster frei gestalten. Die Feld-Namen stehen doppelten geschweiften Klammern und fungieren als Platzhalter für den Feldinhalt:

```
ID: {{Id}}
{{Vorname}} {{Nachname}} , Geburtstag: {{GebDatum}}
----
```

Im zweiten Schritt erzeugen Sie eine Report-Datei *liste.report* mit dem vorgegebenen Muster in der Template-Datei in der die ausgewählten Daten nach dem Nachnamen geordnet sind:

```
$ recsel --sort="Nachname" liste.rec | recfmt -f liste.template > liste.report
```

Das ist der gekürzte Inhalt der Report-Datei:

```
ID: 8
Bernd Bär , Geburtstag: 1996-10-22
----
ID: 11
Klaus Dödel , Geburtstag: 1971-06-02
...
ID: 9
Werner Wiesel , Geburtstag: 1990-06-02
----
ID: 3
Ziri Zippel , Geburtstag: 1999-05-13
----
```

Sie werden sich schon gefragt haben, warum Sie sich auch mit dem DBMS Recutils beschäftigen sollten? Das DBMS Recutils ist ein mächtiges System, für das es sich durchaus lohnen kann, eine grafische Oberfläche (GUI) mit Gambas zu entwickeln und zu testen! Noch interessanter wäre es, wenn von Ihnen eine Klasse *recutils.class* zur Verfügung gestellt würde.