22.12 Sicherung von Datenbanken und DB-Tabellen (Dump)

Es ist eine gute Idee, wenn Sie Ihre Datenbanken oder DB-Tabellen in regelmäßigen Abständen sichern. Sie sollten

- eine komplette Datenbank (DB-Schema und Daten aller DB-Tabellen) sichern oder
- eine DB-Tabelle einer bestimmten Datenbank sichern oder
- nur das Schema einer Datenbank oder DB-Tabelle sichern.

In diesem Kapitel wird Ihnen ein Projekt vorgestellt, dass eine komplette Datenbank der DBMS SQLite, PostgreSQL und MySQL jeweils in einer SQL-Datei sichert. Das gewählte Format hat den Vorteil, dass Sie die Sicherung bei Notwendigkeit sehr einfach wieder einspielen können.

Achtung: Das Projekt müssen Sie an das verwendete DBMS anpassen! Sie werden nicht umhinkommen, die *nicht* benötigten Quelltext-Abschnitte auszukommentieren oder zu löschen. Besondere Sorgfalt erfordert das Modul MDBS, in dem Sie u.a. die Zugangsdaten für die zu sichernde Datenbank und deren Namen festlegen.

Hier der Quelltext-Ausschnitt aus dem Modul MDBS.module für das DBMS PostgreSQL:

```
If Not $DBConPG1.Opened Then
   $DBConPG1.Type = "postgresql"
   $DBConPG1.Port = 5432
   $DBConPG1.User = "test"
   $DBConPG1.Password = "test"
   $DBConPG1.Host = "localhost"     '-- DBHost is `localhost ` or  an IP adddress
   $DBConPG1.Name = "test"          '-- Name of the database
   If $DBConPG1.User = Null Or If $DBConPG1.Password = Null Then
      Error.Raise(("No database credentials"))
   Endif
   $DBConPG1.Open()
Endif
```

Die Programmoberfläche ist einfach strukturiert:

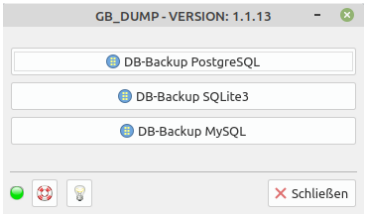

Abbildung 22.12.1: Programm-Oberfläche

Das Programm verfügt über eine Programm-Hilfe. Es wird auch geprüft, ob das Programm auf das Netz zugreifen kann. Das ist u. U. für PostgreSQL und MySQL erforderlich, wenn Sie für die Host-Eigenschaft eine IP-Adresse angeben. Außerdem wird festgestellt, ob die Datenbank-Server für PostgreSQL und MySQL installiert und gestartet sind. Auch Syntaxfehler in den Shell-Anweisungen werden bemerkt und angezeigt.

Abbildung 22.12.2: Fehlermeldung: PostgreSQL-Server nicht gestartet

Abbildung 22.12.3: Fehlermeldung wegen eines Syntaxfehlers

Die Hauptlast im Programm trägt die Prozedur DBDumpShell(argDBType As String, argDBName As String) im Modul MDBS.module, dessen Quelltext vollständig angegeben und anschließend kommentiert wird:

```
[1]  ' Gambas class file
[2]
[3]  Private $hProcess As Process
[4]  Private $aErrorMessages As String[]
[5]  ...
[6]
[7]  '' Argument 1:argDBType -> DBMS type<br>''Argument 2:argDBName -> Database-Name
[8]  Public Sub DBDumpShell(argDBType As String, argDBName As String)
[9]
[10] Dim sShellCommand, sMessage As String
[11] Dim sDumpFilePath, sDatabasePath As String
[12]
[13] $aErrorMessages = New String[]
[14]
[15] Select Case Lower(argDBType)
[16]   Case "sqlite"
[17]     MAdditional.IsInstalled("sqlite3")
[18]     sDatabasePath = Shell$(MCreateDir.DBHost &/ argDBName)
[19]     sDumpFilePath = Shell$(MCreateDir.DataDir &/ "dump_sqlite." & argDBName & ".sql")
[20]     If Exist(sDumpFilePath) Then Kill sDumpFilePath
[21]
[22]     sShellCommand = Subst("&1 &2 &3 &4 &5", "sqlite3", sDatabasePath, ".dump", " > ", sDumpFilePath)
[23]
[24]     $hProcess = Shell sShellCommand For Read As "ShellProcess"
[25]     $hProcess.Wait()
[26]
[27]     If $hProcess.State = 0 And $hProcess.Value = 0 Then
[28]        sMessage = Subst(("The backup of the SQLite3 database &1 was successful!"), "<br><b>" & argDBName & "</b><br>")
[29]        Message.Info(sMessage)
[30]     Else
[31]        If $aErrorMessages.Count > 0 Then Message.Error($aErrorMessages.Join("\n"))
[32]     Endif
[33] '-------------------------------------------------------------------------------------------------------------------------
[34]   Case "postgresql"
[35]     MAdditional.IsInstalled("pg_dump")
[36]     sDumpFilePath = Shell$(MCreateDir.DataDir &/ "dump_pg_" & argDBName & ".sql")
[37]     If Exist(sDumpFilePath) Then Kill sDumpFilePath
[38]
[39]     sShellCommand = Subst("PGPASSWORD=&1 pg_dump --blobs --column-inserts", CDBS.DBConPG1.Password)
[40]     sShellCommand &= Subst(" --dbname=&1 --host=&2", argDBName, CDBS.DBConPG1.Host)
[41]     sShellCommand &= Subst(" --port=&1", CDBS.DBConPG1.Port)
[42]     sShellCommand &= Subst(" --username=&1 --file=&2", CDBS.DBConPG1.User, sDumpFilePath)
[43]
[44]     $hProcess = Shell sShellCommand For Read As "ShellProcess"
[45]     $hProcess.Wait()
[46]
[47]     If $hProcess.State = 0 And $hProcess.Value = 0 Then
[48]        sMessage = Subst(("The backup of the PG database &1 was successful!"), "<br><b>" & argDBName & "</b><br>")
[49]        Message.Info(sMessage)
[50]     Else
[51]        If $aErrorMessages.Count > 0 Then Message.Error($aErrorMessages.Join("\n"))
[52]     Endif
[53]
[54] '-- Only deletes the history of the *current* session!
[55]     Shell "unset HISTFILE && exit"
[56] '-------------------------------------------------------------------------------------------------------------------------
[57]   Case "mysql"
[58]     MAdditional.IsInstalled("mysqldump")
[59]     sDumpFilePath = Shell$(MCreateDir.DataDir &/ "dump_mysql_" & argDBName & ".sql")
[60]     If Exist(sDumpFilePath) Then Kill sDumpFilePath
[61]
[62]     sShellCommand = "mysqldump --comments --dump-date --no-tablespaces"
[63]     sShellCommand &= Subst(" --host=&1 --port=&2", CDBS.DBConMySQL1.Host, CDBS.DBConMySQL1.Port)
[64]     sShellCommand &= Subst(" --user=&1 -p&2", CDBS.DBConMySQL1.User, CDBS.DBConMySQL1.Password)
[65]     sShellCommand &= Subst(" &1 &2 &3", argDBName, ">", sDumpFilePath)
[66]
[67]     $hProcess = Shell sShellCommand For Read As "ShellProcess"
[68]     $hProcess.Wait()
[69]
[70]     If $hProcess.State = 0 And $hProcess.Value = 0 Then
[71]        sMessage = Subst(("The backup of the MySQL database &1 was successful!"), "<br><b>" & argDBName & "</b><br>")
[72]        Message.Info(sMessage)
[73]     Else
[74]        If $aErrorMessages.Count > 0 Then Message.Error($aErrorMessages.Join("\n"))
[75]     Endif
[76]
[77] '-- Only deletes the history of the *current* session!
[78]     Shell "unset HISTFILE && exit"
[79]
[80] End Select
[81]    If $hProcess Then $hProcess.Kill()
[82] End
[83]
[84] Public Sub ShellProcess_Error(argError As String)
[85]
[86] '-- Print argError
[87]     $aErrorMessages.Push(argError)
[88]
[89] End
```

Kommentar – bezogen auf das DBMS PostgreSQL:

- In Zeile 35 wird geprüft, ob das Sicherungsprogramm `pg_dump` für das DBMS PostgreSQL installiert ist. Ist das Programm nicht installiert, dann wird eine Meldung ausgegeben und das Programm beendet.

Abbildung 22.12.4: Fehlermeldung: Das Programm pg_dump ist nicht installiert

- Die Zeile 37 ist m.E. entbehrlich, da die Sicherungsdatei *.sql bei jeder Sicherung komplett überschrieben wird.
- In den Zeilen 39 bis 42 wird das Shell-Kommando definiert. Die eingesetzten Parameter und Optionen sind erprobt.
- Der Shell-Prozess wird in der Zeile 44 gestartet.
- Wenn der Prozess erfolgreich beendet wurde, dann wird eine Erfolgsmeldung ausgegeben. Im anderen Fall folgt eine Fehlermeldung.

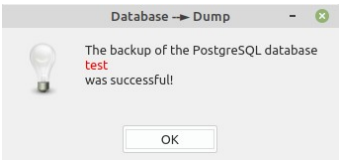

Abbildung 22.12.5: Die Sicherung der PostgreSQL-Datenbank 'test' war erfolgreich

- Die Shell-Instruktion in der Zeile 55 sorgt dafür, dass die Shell-History für den aktuellen Prozess nicht fortgeschrieben wird. Der Grund liegt darin, dass das Passwort im Shell-Kommando steht und somit im nach hinein nicht ausgelesen werden kann.

Es würde den Rahmen des Kapitels sprengen, wenn hier weitere Varianten des Exports und Imports von DB-Daten ausführlich beschrieben werden. Das auch vor dem Hintergrund, dass sich die einzelnen Kommandos für den Export und für den Import für die unterschiedlichen DBMS erheblich unterscheiden, die es für die Sicherung von kompletten Datenbanken, von einzelnen DB-Tabellen, von Schemata und für unterschiedliche Formate der Sicherungsdateien wie dump*.sql oder dump*.csv gibt.