DBCReader avec génération des requêtes SQL

Galathil

Important : Ce tutoriel a été rédigé par Nobodie pour le site wow-emu.fr. Je le recopie ici à l'identique, Il se peut que certaines informations soient obsolètes à ce jour. Bonne lecture !

DBCReader avec génération des requêtes SQL

Bonsoir à tous.

Comme vous le savez si vous avez lu mon tutoriel sur le format de fichier DBC, il est possible de développer notre propre lecteur de DBC. J'en ai mis un basique à disposition qui permet de lire les DBCs et d'en générer un fichier de type CSV.

C'est un début, mais ce n'est pas pratique. Je met donc à disposition un lecteur de DBC qui permet de générer une version SQL de ladite DBC qu'il ne vous restera plus qu'à intégrer en DB afin de pouvoir effectuer des requêtes dessus. Le script est fourni sans plus d'explications, à vous d'aller voir le tutoriel pour comprendre comment il fonctionne et comment l'utiliser, ça se passe par ici : **Le format DBC**.

Et voici le script Python 2 :

```
#!/usr/bin/python
import struct
import sys, getopt

def main(filename, argv):
inputfilename = ''
global outputfilename
outputfilename = ''
global semicolon
semicolon = ','
global verbose
verbose = False
showFormat = False
helper = filename+" [-hfv -s <semicolon> -o <outputfile>] -i <inputfile>"+"\n"

try:
  opts, args = getopt.getopt(argv,"hfvs:o:i:",["semicolon=", "ifile=", "ofile="])
except getopt.GetoptError:
  sys.stdout.write(helper)
  sys.exit(1)
for opt, arg in opts:
  if opt == '-h':
   sys.stdout.write(helper)
   sys.exit()
  elif opt == "-v":
   verbose = True
  elif opt in ("-s", "--semicolon"):
   semicolon = arg
  elif opt in ("-i", "--ifile"):
   inputfilename = arg
  elif opt in ("-o", "--ofile"):
   outputfilename = arg
  elif opt == "-f":
   showFormat = True
```

```
if not inputfilename:
  sys.stdout.write(helper)
  sys.exit(1)

if not outputfilename:
  outputfilename = inputfilename[:-4]+'.sql'

inputfile = open(inputfilename, "rb")
global outputfile
outputfile = open(outputfilename, "w")

try:
  WDBC = inputfile.read(4)
  if WDBC != "WDBC":
   sys.stderr.write("Not supported format\n")
   sys.exit(2)

  headers = readHeaders(inputfile)
  datas = readDatas(inputfile, headers)
  skipBytes(inputfile, 1)
  strings = readStrings(inputfile, headers)
  columns = identifyColumns(headers, datas, strings)
  if showFormat:
   sys.stdout.write("Headers : "+str(headers)+"\n")
   sys.stdout.write("Format : "+''.join(columns)+"\n")
   sys.exit(0)
  writeSQL(headers, datas, strings, columns)
finally:
  inputfile.close()
outputfile.close()

def write(string):
outputfile.write(string)
if verbose:
  sys.stdout.write(string)

def writeSQL(headers, datas, strings, columns):
strcreate = ""
strcreate += "CREATE TABLE IF NOT EXISTS "+outputfilename[:-4]+"(\n"
i = 0
for col in columns:
  if col == 'i':
   strcreate += "INTEGER"+str(i)+" int(11) DEFAULT 0,\n"
  elif col == 'f':
   strcreate += "FLOAT"+str(i)+" float DEFAULT 0,\n"
  elif col == 's':
   strcreate += "STRING"+str(i)+" varchar(255) DEFAULT '',\n"
  i += 1
strcreate = strcreate[:-2]+");\n\n"
write(strcreate)

write("REPLACE INTO "+outputfilename[:-4]+" VALUES\n")

for rec in xrange(headers["nbrec"]):
  recstr = "("
  for field in xrange(headers["nbfields"]):
   col = columns[field]
   if col == 'i':
```

```
     recstr += str(struct.unpack('i', datas[rec][field])[0]) + semicolon
    elif col == 'f':
     recstr += str(struct.unpack('f', datas[rec][field])[0]) + semicolon
    elif col == 's':
     s = struct.unpack('i', datas[rec][field])[0]
     if not s:
      recstr += "NULL" + semicolon
     else:
      recstr = recstr + '"'+str(strings[s])+'"' + semicolon
   if rec == headers["nbrec"]-1:
    recstr = recstr[:-len(semicolon)]+");"
   else:
    recstr = recstr[:-len(semicolon)]+"),\n"
   write(recstr)



def identifyColumns(headers, datas,strings):
columns = ['i']*headers["nbfields"]
for field in range(headers["nbfields"]):
  if headers["ssize"] > 1:
   possiblyAString = True
  else :
   possiblyAString = False

  possiblyAFloat = True
  for rec in range(headers["nbrec"]):
   value = struct.unpack('i', datas[rec][field])[0]
   if not value:
    continue
   if not(value & 0xFF800000) or (value & 0xFF800000 == 0xFF800000):
    possiblyAFloat = False
   if value != 1 and value not in strings:
    possiblyAString = False

  if possiblyAString:
   columns[field] = 's'
   continue
  if possiblyAFloat:
   columns[field] = 'f'
   continue
return columns

def readHeaders(inputfile):
headers = {}
headers["nbrec"] = struct.unpack('i', inputfile.read(4))[0]
headers["nbfields"] = struct.unpack('i', inputfile.read(4))[0]
headers["recsize"] = struct.unpack('i', inputfile.read(4))[0]
headers["ssize"] = struct.unpack('i', inputfile.read(4))[0]
return headers

def readDatas(inputfile, headers):
data = []
for irec in xrange(headers["nbrec"]):
  rec = []
  for jfield in xrange(headers["nbfields"]):
   bytes = inputfile.read(4)
   rec.append(bytes)
  data.append(rec)
return data
```

```
return data

def skipBytes(inputfile, count):
inputfile.read(count)

def readStrings(inputfile, headers):
strings = {}
counter = 1
while counter < headers["ssize"]:
  s = extractString(inputfile)
  strings[counter] = s
  counter += len(s)+1
return strings

def extractString(inputfile):
s = ""
car = inputfile.read(1)
while ord(car) != 0:
  s = s+car
  car = inputfile.read(1)
return s


if __name__ == "__main__":
main(sys.argv[0], sys.argv[1:])
```

Script pour Python 3 :

```
#!/usr/bin/python
import struct
import sys, getopt

def main(filename, argv):
inputfilename = ''
global outputfilename
outputfilename = ''
global semicolon
semicolon = ','
global verbose
verbose = False
showFormat = False
helper = filename+" [-hfv -s <semicolon> -o <outputfile>] -i <inputfile>"+"\n"

try:
  opts, args = getopt.getopt(argv,"hfvs:o:i:",["semicolon=", "ifile=", "ofile="])
except getopt.GetoptError:
  sys.stdout.write(helper)
  sys.exit(1)
for opt, arg in opts:
  if opt == '-h':
   sys.stdout.write(helper)
   sys.exit()
  elif opt == "-v":
   verbose = True
  elif opt in ("-s", "--semicolon"):
   semicolon = arg
  elif opt in ("-i", "--ifile"):
   inputfilename = arg
  elif opt in ("-o", "--ofile"):
   outputfilename = arg
```

```
    outputfilename = arg
  elif opt == "-f":
    showFormat = True

if not inputfilename:
  sys.stdout.write(helper)
  sys.exit(1)

if not outputfilename:
  outputfilename = inputfilename[:-4]+'.sql'

inputfile = open(inputfilename, "rb")
global outputfile
outputfile = open(outputfilename, "w")

try:
  WDBC = inputfile.read(4)
  if WDBC.decode() != "WDBC":
    sys.stderr.write("Not supported format\n")
    sys.exit(2)

  headers = readHeaders(inputfile)
  datas = readDatas(inputfile, headers)
  skipBytes(inputfile, 1)
  strings = readStrings(inputfile, headers)
  columns = identifyColumns(headers, datas, strings)
  if showFormat:
    sys.stdout.write("Headers : "+str(headers)+"\n")
    sys.stdout.write("Format : "+''.join(columns)+"\n")
    sys.exit(0)
  writeSQL(headers, datas, strings, columns)
finally:
  inputfile.close()
outputfile.close()

def write(string):
outputfile.write(string)
if verbose:
  sys.stdout.write(string)

def writeSQL(headers, datas, strings, columns):
strcreate = ""
strcreate += "CREATE TABLE IF NOT EXISTS "+outputfilename[:-4]+"(\n"
i = 0
for col in columns:
  if col == 'i':
    strcreate += "INTEGER"+str(i)+" int(11) DEFAULT 0,\n"
  elif col == 'f':
    strcreate += "FLOAT"+str(i)+" float DEFAULT 0,\n"
  elif col == 's':
    strcreate += "STRING"+str(i)+" varchar(255) DEFAULT '',\n"
  i += 1
strcreate = strcreate[:-2]+");\n\n"
write(strcreate)

write("REPLACE INTO "+outputfilename[:-4]+" VALUES\n")

for rec in range(headers["nbrec"]):
  recstr = "("
  for field in range(headers["nbfields"]):
```

```
    col = columns[field]
    if col == 'i':
     recstr += str(struct.unpack('i', datas[rec][field])[0]) + semicolon
    elif col == 'f':
     recstr += str(struct.unpack('f', datas[rec][field])[0]) + semicolon
    elif col == 's':
     s = struct.unpack('i', datas[rec][field])[0]
     if not s:
      recstr += "NULL" + semicolon
     else:
      recstr = recstr + '"'+str(strings[s])+'"' + semicolon
   if rec == headers["nbrec"]-1:
    recstr = recstr[:-len(semicolon)]+");"
   else:
    recstr = recstr[:-len(semicolon)]+"),\n"
   write(recstr)



def identifyColumns(headers, datas,strings):
columns = ['i']*headers["nbfields"]
for field in range(headers["nbfields"]):
  if headers["ssize"] > 1:
   possiblyAString = True
  else :
   possiblyAString = False

  possiblyAFloat = True
  for rec in range(headers["nbrec"]):
   value = struct.unpack('i', datas[rec][field])[0]
   if not value:
    continue
   if not(value & 0xFF800000) or (value & 0xFF800000 == 0xFF800000):
    possiblyAFloat = False
   if value != 1 and value not in strings:
    possiblyAString = False

  if possiblyAString:
   columns[field] = 's'
   continue
  if possiblyAFloat:
   columns[field] = 'f'
   continue
return columns


def readHeaders(inputfile):
headers = {}
headers["nbrec"] = struct.unpack('i', inputfile.read(4))[0]
headers["nbfields"] = struct.unpack('i', inputfile.read(4))[0]
headers["recsize"] = struct.unpack('i', inputfile.read(4))[0]
headers["ssize"] = struct.unpack('i', inputfile.read(4))[0]
return headers

def readDatas(inputfile, headers):
data = []
for irec in range(headers["nbrec"]):
  rec = []
  for jfield in range(headers["nbfields"]):
```

```
    bytes = inputfile.read(4)
    rec.append(bytes)
  data.append(rec)
return data

def skipBytes(inputfile, count):
inputfile.read(count)

def readStrings(inputfile, headers):
strings = {}
counter = 1
while counter < headers["ssize"]:
  s = extractString(inputfile)
  strings[counter] = s
  counter += len(s)+1
return strings

def extractString(inputfile):
s = ""
car = inputfile.read(1)
while car.decode('UTF-8', 'ignore') != '\x00':
  c = car.decode('UTF-8','ignore')
  if c :
   s = s+c
  else :
   s = s+'_'
  car = inputfile.read(1)
return s


if __name__ == "__main__":
main(sys.argv[0], sys.argv[1:])
```

Bonne utilisation