Kapitel 9
SAP DB und Perl

Neben der Zugriffsmöglichkeit mit einem Tool wie dbmcli oder einem Programm wie dem SQL Studio kann auf die SAP DB auch mit Hilfe verschiedener Programmiersprachen zugegriffen werden, denn oft möchte man die Fähigkeiten einer Datenbank erweitern oder neue Funktionen hinzufügen.

Diese Programmiersprachen besitzen eine Schnittstelle zur Datenbank. Diese Schnittstelle, oft in Form einer DLL implementiert, liefert Befehle zum Login, der Abfrage und Bearbeitung der Datenbank und für die Trennung. Diese Ansprache des Datenbankservers kann sowohl mit eigenen Befehlen der Datenbank, und besser, weil allgemein verständlicher, mit SQL-Befehlen erfolgen. Diese Implementierung der SQL-Befehle in eine Programmiersprache wird als *Embedded SQL* bezeichnet.

Beispiele auf der CD-ROM

Auf der SAP-DB-CD werden Beispiele mitgeliefert. Diese verschiedenen Beispiele mit unterschiedlichen Programmiersprachen sind im Verzeichnis *ftp/7.4* in der Datei *firststeps* gespeichert.

Weil die Installation des Perl-Interpreters bei vielen Linux-Distributionen bereits im Standardpaket erfolgt, wird im folgenden exemplarisch die Ansprache von SAP DB durch Perl beschrieben. Müssen Sie Perl erst downloaden, finden Sie es auf *www.perl.com* (Unix) und *www.activestate.com* (Windows).

Zuerst muß der Datenbankserver gestartet sein. Man fährt ihn wie bereits beschrieben mit *x_server* hoch. Bei der Standard-Installation befindet er sich in */opt/sapdb/indep_prog/bin*. Ist dieses Verzeichnis nicht in die Pfad-Variable aufgenommen, muß der komplette Pfad beim Aufruf mit angegeben werden. Man darf sich nicht davon irritieren lassen, daß der Server die Erfolgsmeldung »Vserver startet« ausgibt.

Damit ist die Grundvoraussetzung für den Start des Clients geschaffen. Er wird mit dem Aufruf

```
dbmcli -s -d TST -u dbm,dbm db_start
```

gestartet. Die Datenbank ist jetzt zwar gestartet, jedoch noch nicht bereit, Befehle entgegenzunehmen. Sie befindet sich im »kalten« Zustand (cold). Also ist zusätzlich die folgende Initialisierung nötig:

```
dbmcli -s -d TST -u dbm,dbm db_warm
```

Zur Erinnerung: Der Status der gestarteten Datenbank wird mit

```
dbmcli -s -d TST -u dbm,dbm db_warm
```

abgefragt, so daß sichergestellt ist, daß der Datenbankserver wirklich läuft.

In Perl müssen zwei Module für datenbankspezifische Befehle geladen werden. Nach dem Einbinden von *sapdb* mit *use* steht der neue Befehl *sql* zur Verfügung. Ihm übergibt man dann die üblichen SQL-Statements wie *CREATE TABLE* oder *SELECT* als Parameter (Listing 1). Das Packagemodul mit dem Namen *demodbData.pm* enthält die Benutzer, die in der Datenbank TST bekannt sind.

```
#!/usr/bin/env perl
use sapdb;
use demodbData;

sub sayHello(session) {
  my($session) = @_;
  my($cursor)  = $session->sql(
     "CREATE TABLE table_name(nachname CHAR(12))");
  my($cursor)  = $session->sql(
     "INSERT INTO table_name(nachname) VALUES('wichmann')");
  my($cursor)  = $session->sql(
     "SELECT COUNT(*) FROM table_name");
  my($cursor)  = $session->sql(
     "SELECT NAME FROM table_name");
  my($row)     = $cursor->next();
  my($result);
  # row is a tuple with one element
  ($result,)   = $row->[0];
  print "$result\n";
}

sub createSession {
  my($username, $password, $dbname, $host) = @_;
  my($session) = sapdb::connect($username, $password,
                                $dbname, $host);
  return $session;
}

sub main {
  my(@argv) = @ARGV;
  my($username, $password, $dbname,
     $host) = parseUserArgs(@ARGV);
```

```
  my($session) = &createSession($username, $password,
                                $dbname, $host);
  &sayHello($session);
  $session->commit ();  # not necessary for a Select,
                        # but good style anyway
}

&main();
```

Listing 9.1: Das Perl-Skript zur Verwaltung der Beispieldatenbank

Listing 2 zeigt das Programm, mit dem eine Tabelle gelöscht wird.

```
#!/usr/bin/env perl
use sapdb;
use demodbData;

sub sayHello(session) {
  my($session) = @_;
  my($cursor)  = $session->sql("DROP TABLE table_names1");
}

sub createSession {
  my($username, $password, $dbname, $host) = @_;
  my($session) = sapdb::connect($username, $password,
                                $dbname, $host);
  return $session;
}

sub main {
  my(@argv) = @ARGV;
  my($username, $password, $dbname,
     $host) = parseUserArgs(@ARGV);
  my($session) = &createSession($username, $password,
                                $dbname, $host);
  &sayHello($session);
  $session->commit();
}

&main();
```

Listing 9.2: Löschen der Tabelle

Auf diese Weise lassen sich aus Perl-Programmen beliebige SQL-Anweisungen an den Datenbank-Server senden.