3 Geavanceerde database-concepten

Ik had vele doelen bij het schrijven van deze tweede editie van mijn PHP-boek voor gevorderden. Het hoofddoel is het demonstreren van wat ik beschouw als 'geavanceerd' PHP-programmeren: dingen die u al doet beter doen; dingen doen die oppervlakkig zijn gerelateerd aan PHP; en profiteren van mogelijkheden van de taal waarmee de gemiddelde gebruiker misschien onbekend is. Een tweede doel is het beantwoorden van enkele vragen die vaak worden gesteld (aan mij of op andere wijze) in e-mails, forums en nieuwsgroepen. Dit hoofdstuk komt aan beide doelen tegemoet.

In het eerste voorbeeld ziet u hoe u een database gebruikt om sessiegegevens te bewaren. Dit biedt vele voordelen, in het bijzonder betere beveiliging. Dan volgt een grondige bespreking van het werken met (Amerikaanse) postcodes, gedemonstreerd aan de hand van een script voor afstandsberekening (bijvoorbeeld, hoe ver liggen verschillende winkels van een gegeven postcode). Het derde voorbeeld introduceert opgeslagen functies, een uitbreiding op MySQL in versie 5 (maar al enige tijd aanwezig in andere databases). Daarna beantwoord ik een veelgestelde vraag: hoe maakt u de resultaten van een query horizontaal op?

Let op: alle voorbeelden in dit hoofdstuk gebruiken de databaseapplicatie MySQL. De meeste technieken zijn echter implementaties van theorieën die niet afhankelijk zijn van het databaseplatform. Het is niet moeilijk ze te vertalen naar uw databaseapplicatie. Daarnaast worden uitsluitend Improved MySQL-functies gebruikt, beschikbaar vanaf PHP 5 en MySQL 4.1. Als u oudere versies van een van beide gebruikt, moet u de code aanpassen naar de eerdere (de oude standaard) MySQL-functies.

Sessies opslaan in een database

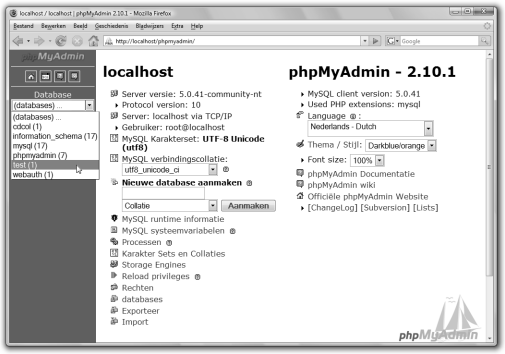
Standaard slaat PHP alle sessiegegevens op in tekstbestanden op de server. Gewoonlijk worden deze bestanden opgeslagen in een tijdelijke map (zoals `/tmp` bij UNIX en Mac OS X) met bestandsnamen die overeenkomen met de sessie-ID (bijvoorbeeld *ei26b4i2nup742uch09glmbh84*). PHP ondersteunt ook de mogelijkheid om dezelfde sessiegegevens in een database op te slaan.

De belangrijkste reden om dit te doen is een verbeterde beveiliging. Op gedeelde hostservers gebruiken alle websites dezelfde tijdelijke directory. Dit betekent dat tientallen applicaties op dezelfde locatie lezen en schrijven. Het is erg gemakkelijk om een script te maken dat alle gegevens uit alle bestanden in de sessiemap leest.

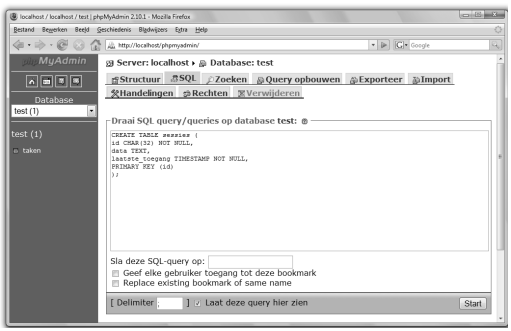
Ten tweede kunt u door de sessiegegevens naar een database te verplaatsen in het algemeen gemakkelijk meer informatie ophalen over de sessies van uw websites. U kunt een query uitvoeren naar het aantal actieve sessies en een back-up maken van sessiegegevens.

Een derde reden om sessiegegevens op te slaan in een database is het draaien van een site op meerdere servers. In dat geval kan dezelfde gebruiker tijdens een sessie pagina's ontvangen van verschillende servers. Sessiegegevens die zijn opgeslagen in een bestand op één server, zijn niet beschikbaar voor de pagina's op andere servers. Dit is niet iets waarmee veel ontwikkelaars te maken krijgen, maar als u het tegenkomt, is een databaseoplossing eigenlijk de enige optie.

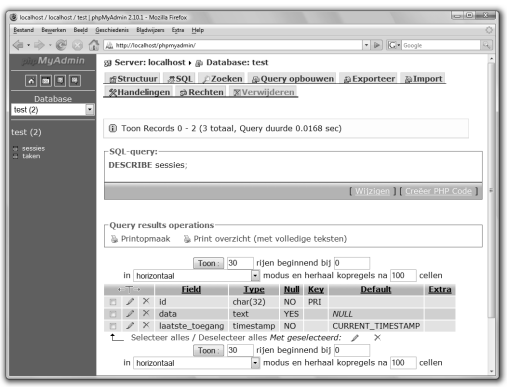
TIP Een andere oplossing voor de beveiligingskwestie op een gedeelde host is om de sessiedirectory voor uw site te wijzigen. Dit doet u door de functie `session_save_path()` aan te roepen voor elke aanroep van `session_start()`. U moet er natuurlijk ook voor zorgen dat de nieuwe directory bestaat en de juiste machtigingen heeft.



Figuur 3.1 Voor dit voorbeeld plaatsen we de sessietabel in de database `test`.



Figuur 3.2 Deze ene tabel handelt alle sessiegegevens af.



Figuur 3.3 De tabelstructuur bevestigen.

De sessietabel maken

De sessiegegevens wordt opgeslagen in een speciale tabel. Deze tabel kan deel uitmaken van een bestaande database (evenals de rest van uw applicatie) of zelfstandig bestaan. De tabel heeft ten minste drie kolommen nodig (**tabel 3.1**).

De tabel `session` kan meer dan drie kolommen krijgen, maar in elk geval deze drie. Bedenk dat veel dingen die u wellicht zou representeren in een andere kolom (bijvoorbeeld een gebruikers-ID), waarschijnlijk zijn opgeslagen in de sessiegegevens.

De sessietabel maken

1. Maak verbinding met uw MySQL-database via phpMyAdmin.

   U kunt ook de client `mysql` of een andere interface gebruiken als u dat wilt.

2. Selecteer de database `test` (**figuur 3.1**).

   ```
   USE test;
   ```

   Omdat dit slechts een voorbeeld is, maken we de tabel in de database `test`.

3. Maak de tabel `sessies` (**figuur 3.2**).

   ```
   CREATE TABLE sessies (
   id CHAR(32) NOT NULL,
   data TEXT,
   laatste_toegang TIMESTAMP NOT NULL,
   PRIMARY KEY (id)
   );
   ```

   De tabel bevat de drie basisvelden. De `id` is de primaire sleutel. Hij bevat altijd een string van 32 tekens en kan nooit `NULL` zijn. De kolom `data` bevat `TEXT` en kan `NULL` zijn (wanneer de sessie start, is er nog geen data). De kolom `laatste_toegang` is een `TIMESTAMP`. Deze wordt dus altijd bijgewerkt wanneer de sessie wordt gemaakt (bij `INSERT`) of gewijzigd (bij `UPDATE`).

4. Controleer de tabelstructuur (**figuur 3.3**).

   ```
   DESCRIBE sessies;
   ```

TIP U hoeft geen MySQL te gebruiken voor dit hoofdstuk; u kunt ook PostgreSQL, Oracle, SQLite of een andere database gebruiken.

TIP Als uw applicatie veel gegevens in sessies opslaat, moet u de grootte van de kolom voor sessiegegevens veranderen in `MEDIUMTEXT` of `LONGTEXT`.

Tabel 3.1 Een tabel met drie kolommen is genoeg om sessiegegevens in een database op te slaan.

Kolommen sessietabel	
Kolomtype	**Bewaart**
CHAR(32)	De sessie-ID
TEXT	De sessiegegevens
TIMESTAMP	De laatste keer dat de sessiegegevens werden gebruikt

De sessiefuncties definiëren

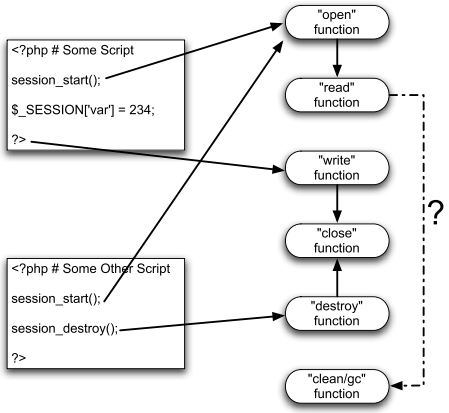
Nadat de databasetabel is gemaakt, is het opslaan van sessiegegevens in een database een proces in twee stappen (vanuit PHP gezien):

1. Definieer functies voor de interactie met de database.
2. Vertel PHP hoe het deze functies moet gebruiken.

   Voor de tweede stap wordt de functie `session_set_save_handler()` aangeroepen. Deze functie moet worden aangeroepen met zes argumenten, elk een functienaam (**tabel 3.2**).

   We bespreken kort wat elke functie moet ontvangen (als argumenten) en maken daarbij het volgende script. U moet nu al weten dat elke functie een booleaanse waarde moet retourneren, met uitzondering van de leesfunctie. Die functie retourneert altijd een string, zelfs als dat een lege string is.

   Figuur 3.4 is een overzicht van wanneer de verschillende functies worden aangeroepen (vanuit het perspectief van wat u doet in de PHP-code).



Figuur 3.4 Wanneer een sessie start, worden de functies 'openen' en 'lezen' aangeroepen. Bij de aanroep van de functie 'lezen', kan *garbage collection* plaatsvinden (afhankelijk van diverse factoren). Aan het einde van een script worden de gegevens geschreven en de functie 'sluiten' aangeroepen, tenzij de sessie werd vernietigd; in dat geval wordt de functie 'schrijven' niet aangeroepen.

Tabel 3.2 De functie `session_set_save_handler()` accepteert zes argumenten. Voor elk argument gebruikt u de naam van een functie die wordt aangeroepen wanneer een bepaalde gebeurtenis plaatsvindt.

Argumenten session_set_save_handler	
Volgorde	**Aan te roepen functie wanneer...**
1	een sessie wordt gestart
2	een sessie wordt gesloten
3	sessiegegevens worden gelezen
4	sessiegegevens worden geschreven
5	sessiegegevens worden vernietigd
6	oude sessiegegevens moeten worden verwijderd (oftewel bij de uitvoer van garbage collection)

Om nieuwe sessiehandlers te maken

1. Begin een nieuwe PHP-script in uw teksteditor of IDE (**script 3.1**).

   ```
   <?php # Script 3.1 - db_sessies.inc.php
   $sdbc = NULL;
   ```

   De variabele `$sdbc` bewaart de databaseverbinding. We starten hem hier en maken hem dan globaal in elke functie.

Script 3.1 (db_sessies.inc.php) Dit script definieert alle functionaliteit die nodig is om sessiegegevens in een database op te slaan. U kunt het invoegen in elke pagina waarin deze functionaliteit nodig is.

```
1   <?php # Script 3.1 - db_sessies.inc.php
2
3   /**
4    * Deze pagina is de functionele interface
5    * voor de opslag van sessiegegevens
6    * in een database.
7    * Deze pagina start ook de sessie.
8    */
9
10  // Globale variabele die wordt gebruikt voor
11  // de databaseverbindingen in alle sessiefuncties:
12  $sdbc = NULL;
13
14  // Definieer de functie open_sessie().
15  // Deze functie accepteert geen argumenten.
16  // Deze functie moet de databaseverbinding openen.
17  function open_sessie() {
18
19      global $sdbc;
20
21      // Maak verbinding met de database:
22      $sdbc = mysqli_connect('localhost', 'gebruikersnaam', 'wachtwoord', 'test') OR die
    ('Databaseverbinding kon niet worden gemaakt.');
23
24      return true;
25
26  } // Einde van de functie open_sessie().
27
28  // Definieer de functie sluit_sessie().
29  // Deze functie accepteert geen argumenten.
30  // Deze functie sluit de databaseverbinding.
```

Script 3.1 Vervolg

```
31  function sluit_sessie() {
32
33      global $sdbc;
34
35      return mysqli_close($sdbc);
36
37  } // Einde van de functie sluit_sessie().
38
39  // Definieer de functie lees_sessie():
40  // Deze functie accepteert een argument: het sessie-ID.
41  // Deze functie haalt de sessiegegevens op.
42  function lees_sessie($sid) {
43
44      global $sdbc;
45
46      // Databasequery uitvoeren:
47     $q = sprintf('SELECT data FROM sessies WHERE id="%s"', mysqli_real_escape_string($sdbc,
    $sid));
48      $r = mysqli_query($sdbc, $q);
49
50      // Haal de resultaten op:
51      if (mysqli_num_rows($r) == 1) {
52
53          list($data) = mysqli_fetch_array($r, MYSQLI_NUM);
54
55          // Retourneer de gegevens:
56          return $data;
57
58      } else { // Een lege string retourneren.
59          return '';
60      }
61
62  } // Einde van de functie lees_sessie().
63
64  // Definieer de functie schrijf_sessie()
65  // Deze functie accepteert twee argumenten:
66  // de sessie-ID en de sessiegegevens.
67  function schrijf_sessie($sid, $data) {
68
69      global $sdbc;
70
71      // Opslaan in de database:
72     $q = sprintf('REPLACE INTO sessies (id, data) VALUES ("%s", "%s")', mysqli_real_escape_
    string($sdbc, $sid), mysqli_real_escape_string($sdbc, $data));
73      $r = mysqli_query($sdbc, $q);
74
75      return mysqli_affected_rows($sdbc);
76
77  } // Einde van de functie schrijf_sessie().
78
79  // Definieer de functie vernietig_sessie().
80  // Deze functie accepteert een argument: de sessie-ID.
```

Script 3.1 Vervolg

```
81  function vernietig_sessie($sid) {
82
83      global $sdbc;
84
85      // Verwijderen uit de database:
86      $q = sprintf('DELETE FROM sessies WHERE id="%s"', mysqli_real_escape_string($sdbc,
    $sid));
87      $r = mysqli_query($sdbc, $q);
88
89      // Maak de array $_SESSION leeg:
90      $_SESSION = array();
91
92      return mysqli_affected_rows($sdbc);
93
94  } // Einde van de functie vernietig_sessie().
95
96  // Definieer de functie schoon_sessie().
97  // Deze functie accepteert een argument: een waarde in seconden.
98  function schoon_sessie($seconden) {
99
100     global $sdbc;
101
102     // Oude sessies verwijderen:
103     $q = sprintf('DELETE FROM sessies WHERE DATE_ADD(laatste_toegang, INTERVAL %d SECOND)
    < NOW()', (int) $seconden);
104     $r = mysqli_query($sdbc, $q);
105
106     return mysqli_affected_rows($sdbc);
107
108 } // Einde van de functie schoon_sessie().
109
110 # ********************************* #
111 # ***** EINDE VAN DE FUNCTIES ***** #
112 # ********************************* #
113
114 // Declareer de te gebruiken functies:
115 session_set_save_handler('open_sessie', 'sluit_sessie', 'lees_sessie', 'schrijf_sessie',
    'vernietig_sessie', 'schoon_sessie');
116
117 // Plaats hier eventuele wijzigingen van de sessie-instellingen.
118
119 // Start de sessie:
120 session_start();
121
122 ?>
```

2. Definieer de functie die een sessie opent.

```
function open_sessie() {
    global $sdbc;
    $sdbc = mysqli_connect('localhost',
→'gebruikersnaam', 'wachtwoord', 'test')
→OR die ('Databaseverbinding kon niet
→worden gemaakt.');
    return true;
}
```

   Deze functie accepteert geen argumenten. (Dat wil zeggen: wanneer PHP iets doet om een sessie te openen, wordt deze functie aangeroepen zonder er waarden aan door te geven.) Het doel van deze functie is het openen van een databaseverbinding.

3. Definieer de functie die een sessie sluit.

```
function sluit_sessie() {
    global $sdbc;
    return mysqli_close($sdbc);
}
```

   Ook deze functie accepteert geen argumenten. Zij sluit de databaseverbinding en retourneert het slagen van die bewerking.

4. Definieer de functie die sessiegegevens leest.

```
function lees_sessie($sid) {
    global $sdbc;
    $q = sprintf('SELECT data FROM
→sessies WHERE id="%s"',
→mysqli_real_escape_string($sdbc,
→$sid));
    $r = mysqli_query($sdbc, $q);
    if (mysqli_num_rows($r) == 1) {
        list($data) =
→mysqli_fetch_array($r, MYSQLI_NUM);
        return $data;
    } else {
        return '';
    }
}
```

   Deze functie ontvangt een argument: de sessie-ID (bijvoorbeeld *ei26b4i2nup742uch09glmbh84*). De functie moet de gegevens voor die sessie-ID uit de database ophalen en retourneren. Als dat niet kan, moet ze een lege string retourneren. Hoewel de sessie-ID veilig gebruikt moet kunnen worden in een URL, mag u nergens van uitgaan als het gaat om beveiliging, dus wordt voor de zekerheid de functie `mysqli_real_escape_string()` gebruikt (u kunt ook voorbereide statements gebruiken).

   Als u niet bekend bent met de functie `sprintf()`, die de query compileert, kunt u terecht in hoofdstuk 1, 'Geavanceerde PHP-technieken'.

5. Definieer de functie die gegevens naar de database schrijft.

```
function schrijf_sessie($sid, $data) {
    global $sdbc;
    $q = sprintf('REPLACE INTO sessies
→(id, data) VALUES ("%s", "%s")',
→mysqli_real_escape_string($sdbc,
→$sid), mysqli_real_escape_string($sdbc,
→$data));
    $r = mysqli_query($sdbc, $q);
    return mysqli_affected_rows($sdbc);
}
```

   Deze functie ontvangt twee argumenten: de sessie-ID en de sessiegegevens. De sessiegegevens zijn een genummerde versie van de array `$_SESSION` (**figuur 3.5**). Voor de query moet bij de eerste keer dat de sessierecord wordt gemaakt een `INSERT` in de database worden uitgevoerd en elke volgende keer een `UPDATE`. De minder bekende query `REPLACE` bereikt hetzelfde resultaat. Als een record bestaat waarvan de primaire sleutel gelijk is aan een waarde in deze query (dat wil zeggen, de sessie-ID), vindt een update plaats. Anders wordt een nieuwe record gemaakt.

6. Maak ten slotte de functie die de sessiegegevens vernietigt.

```
function vernietig_sessie($sid) {
    global $sdbc;
    $q = sprintf('DELETE FROM sessies
→WHERE id="%s"', mysqli_real_escape_
→string($sdbc, $sid));
    $r = mysqli_query($sdbc, $q);
    $_SESSION = array();
    return mysqli_affected_rows($sdbc);
}
```

Deze functie ontvangt bij haar aanroep een argument: de sessie-ID. Gewoonlijk gebeurt dit bij de aanroep van de functie `session_destroy()`. Deze functie voert in een databasequery met `DELETE` uit en maakt de array `$_SESSION` leeg.

7. Definieer de functie voor garbage collection.

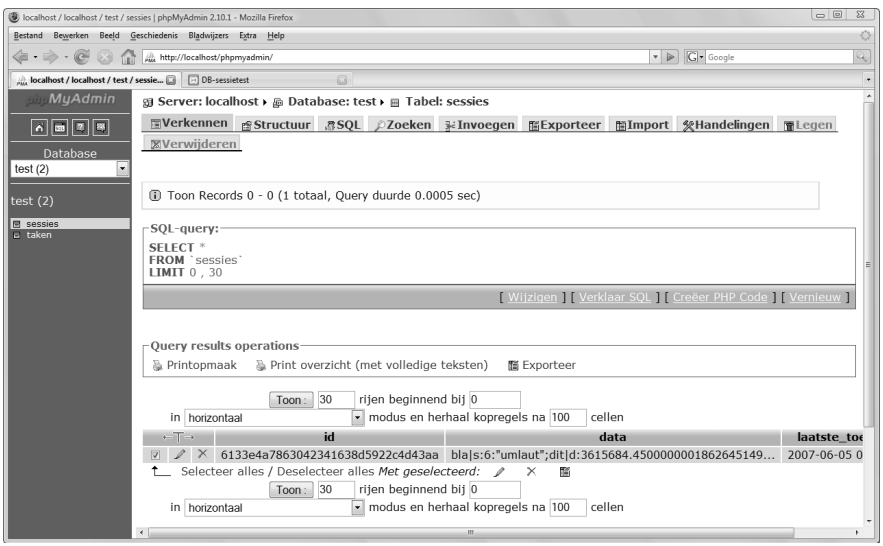
```
function schoon_sessie($seconden) {
    global $sdbc;
    $q = sprintf('DELETE FROM sessies
→WHERE DATE_ADD(laatste_toegang,
→INTERVAL %d SECOND) < NOW()', (int)
→$seconden);
    $r = mysqli_query($sdbc, $q);
    return mysqli_affected_rows($sdbc);
}
```

Garbage collection is iets waar de meeste PHP-programmeurs niet bij stilstaan. Men veronderstelt dat PHP oude sessies automatisch verwijdert. Er zijn twee relevante instellingen in PHP: wat beschouwd wordt als 'oud' en hoe waarschijnlijk het is dat garbage collection wordt uitgevoerd. Voor alle sessieactiviteit in een site is er een kans van *x* procent dat PHP garbage collection uitvoert (het precieze percentage is een PHP-instelling; de standaardwaarde is 1%). Hierbij worden alle 'oude' sessiegegevens vernietigd. Dus elke sessie start garbage collection maar probeert elke sessie op te ruimen.

De functie voor garbage collection ontvangt een tijd, in seconden, die bepaalt wat oud is. Dit kan worden gebruikt in een query met `DELETE` om sessies op te ruimen die langer dan de ingestelde tijd niet zijn gebruikt.

8. Vertel PHP de sessiefuncties te gebruiken.

```
session_set_save_handler('open_sessie',
→'sluit_sessie', 'lees_sessie',
→'schrijf_sessie', 'vernietig_sessie',
→'schoon_sessie');
```



Figuur 3.5 Sessiegegevens worden opgeslagen in een database (of bestand) als genummerde array. Bijvoorbeeld blals:6:"umlaut" betekent dat een string (s) van 6 tekens lang met de waarde *umlaut* is geïndexeerd op `bla`. Daarna wordt ditld:3615684.450000000 1862645149... gebruikt voor de index *dit* met een decimale (d) waarde, enzovoort.

9. Start de sessie.

```
session_start();
```

   Er zijn twee dingen die u hier moet weten. Ten eerste start de functie `session_set_save_handler()` geen sessie. U moet nog steeds `session_start()` aanroepen. Ten tweede moet u deze twee regels in deze volgorde gebruiken. Als u `session_start()` aanroept voor `session_set_save_handler()`, worden de handlers genegeerd.

10. Voltooi de pagina.

```
?>
```

11. Sla het bestand op als `db_sessies.inc.php` en plaats het in uw webdirectory.

TIP De schrijffunctie wordt pas aangeroepen als alle uitvoer naar de webbrowser is verzonden. Dan wordt de sluitfunctie aangeroepen. Zie figuur 3.4.

TIP Als `session.auto_start` in uw PHP-configuratie is ingeschakeld (wat betekent dat sessies automatisch starten voor elke pagina), kunt u de functie `session_set_save_handler()` niet gebruiken.

De nieuwe sessiehandlers gebruiken

Om de nieuwe sessiehandlers te gebruiken, hoeft u alleen maar de functie `session_set_save_handler()` aan te roepen, zoals is besproken in de vorige paragraaf. Er verandert verder niets aan wat u met sessies kunt doen, van het opslaan van gegevens in sessies tot het gebruiken van de opgeslagen gegevens en het vernietigen van de sessie.

Om dit te demonstreren maakt het volgende script enige sessiegegevens als deze niet bestaan, toont het alle sessiegegevens en vernietigt het zelfs de sessiegegevens als wordt geklikt op een hyperlink naar dezelfde pagina. Zoals zo vaak, schuilt er wel een addertje onder het gras...

Alle sessieactiviteiten vereisen toegang tot de database en dus een databaseverbinding. De verbinding wordt geopend wanneer de sessie start en gesloten wanneer de sessie sluit. Dit is geen probleem, alleen dat de schrijf- en sluitfuncties worden aangeroepen na de uitvoer van een script (zie figuur 3.4). Zoals u wellicht al weet, sluit PHP automatisch alle databaseverbindingen wanneer de uitvoer van een script stopt. Voor het volgende script betekent dit dat nadat het script is uitgevoerd, de databaseverbinding automatisch wordt gesloten, *en daarna* proberen de sessiefuncties gegevens naar de database te schrijven en de verbinding te sluiten. Dit leidt tot verwarrende fouten (en de – geloof me – lange zoektocht naar 'waar is de databaseverbinding gebleven?'). Om dit probleem te voorkomen, moet de functie `session_write_close()` worden aangeroepen voor het script wordt beëindigd. Deze functie roept de schrijf- en sluitfuncties aan terwijl de databaseverbinding nog actief is.

Om de nieuwe sessiehandlers te gebruiken

1. Begin een nieuw PHP-script in uw teksteditor of IDE (**script 3.2**).

```
<?php # Script 3.2 - sessies.php
```

Script 3.2 (sessies.php) Dit script voegt de pagina `db_sessies.inc.php` in (script 3.1), zodat sessiegegevens worden opgeslagen in een database. Om in een pagina een nieuwe sessie te maken, bestaande gegevens te gebruiken en de sessie te sluiten, maken we een aantal condities in plaats van een paar pagina's.

```
1   <?php # Script 3.2 - sessies.php
2
3   /**
4    * Deze pagina doet enkele rare dingen met sessies.
5    * Hij voegt het script db_sessions.inc.php in,
6    * zodat de sessiegegevens worden opgeslagen in
7    * een database.
8    */
9
10  // Voeg het bestand voor sessies in:
11  // dat bestand start de sessie al.
12  require_once('db_sessies.inc.php');
13  ?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
14       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
15  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="nl" lang="nl">
16  <head>
17    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
18    <title>DB-sessietest</title>
19  </head>
20  <body>
21  <?php
22
23  // Sla enkele dummygegevens op in de sessie
24  // als er geen gegevens aanwezig zijn.
25  if (empty($_SESSION)) {
26
27      $_SESSION['bla'] = 'umlaut';
28      $_SESSION['dit'] = 3615684.45;
29      $_SESSION['dat'] = 'blauw';
30
31      // Toon een melding die aangeeft wat er gebeurt:
32      echo '<p>Sessiegegevens opgeslagen.</p>';
33
34  } else { // Toon de eerder opgeslagen gegevens.
35      echo '<p>Bestaande sessiegegevens:<pre>' . print_r($_SESSION, 1) . '</pre></p>';
36  }
37
38  // De gebruiker uitloggen, indien van toepassing:
39  if (isset($_GET['uitloggen'])) {
40
41      session_destroy();
42      echo '<p>Sessie vernietigd.</p>';
43
44  } else { // Toon de link "Uitloggen":
45      echo '<a href="sessies.php?uitloggen=ja">Uitloggen</a>';
46  }
```

Script 3.2 Vervolg

```
47
48  // Geef de sessiegegevens weer:
49  echo '<p>Sessiegegevens:<pre>' . print_r($_SESSION, 1) . '</pre></p>';
50
51  ?>
52  </body>
53  </html>
54  <?php session_write_close(); ?>
```

2. Voeg het bestand `db_sessies.inc.php` in.

```
require_once('db_sessies.inc.php');
?>
```

De functie `session_start()`, die staat in `db_sessies.inc.php`, moet worden aangeroepen voordat er iets naar de webbrowser wordt verzonden, dus dit bestand moet u voor de XHTML invoegen.

3. Maak het hoofdgedeelte van de XHTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD
→XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/
→DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/
→xhtml" xml:lang="nl" lang="nl">
<head>
  <meta http-equiv="content-type"
→content="text/html; charset=iso-8859-1"
→/>
  <title>DB-sessietest</title>
</head>
<body>
```

4. Sla wat gegevens op in de sessie wanneer deze leeg is.

```
<?php
if (empty($_SESSION)) {
    $_SESSION['bla'] = 'umlaut';
    $_SESSION['dit'] = 3615684.45;
    $_SESSION['dat'] = 'blauw';
} else {
    echo '<p>Bestaande
→sessiegegevens:<pre>' .
→print_r($_SESSION, 1) . '</pre></p>';
}
```

Het opslaan van gegevens in een databasegestuurde sessie werkt niet anders dan de reguliere methode. Deze conditie wordt gebruikt om sessies te repliceren in meerdere pagina's. De eerste keer dat de pagina laadt, worden nieuwe gegevens in de sessie opgeslagen. De tweede keer dat de pagina laadt, zijn de bestaande gegevens beschikbaar. De functie `print_r()` wordt gebruikt als snelle manier om sessiegegevens weer te geven.

5. Maak de functionaliteit voor het uitloggen.

```
if (isset($_GET['uitloggen'])) {
    session_destroy();
    echo '<p>Sessie vernietigd.</p>';
} else {
    echo '<a href="sessies.
→php?uitloggen=ja">Uitloggen</a>';
}
```

Wederom wordt deze conditie gebruikt om een site met meerdere pagina's te simuleren. Wanneer de pagina wordt geopend, wordt de hyperlink 'Uitloggen' getoond. Klikt de gebruiker op de link, dan wordt `?uitloggen=ja` doorgegeven in de URL, wat deze zelfde pagina opdraagt de sessie te vernietigen.

6. Geef de sessiegegevens weer.

```
echo '<p>Sessiegegevens:<pre>' .
→print_r($_SESSION, 1) . '</pre></p>';
```

Dit is grotendeels een herhaling van de code in stap 4. In tegenstelling tot die regel, wordt deze toegepast wanneer de pagina voor de eerste keer laadt. Hij wordt ook gebruikt om het effect van de sessievernietiging te onthullen.

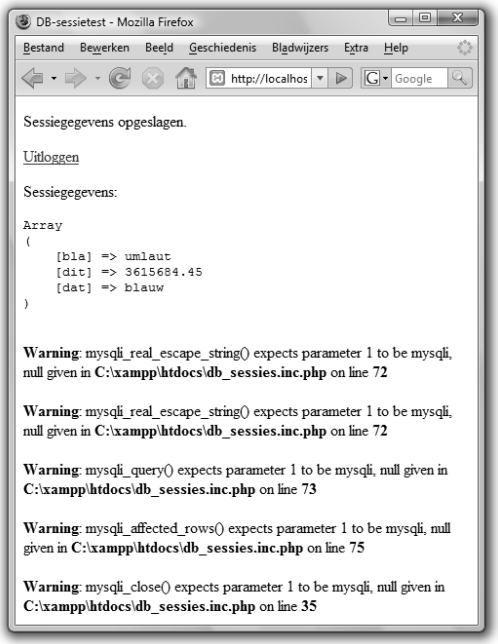
7. Voltooi de PHP en HTML.

```
?>
</body>
</html>
```

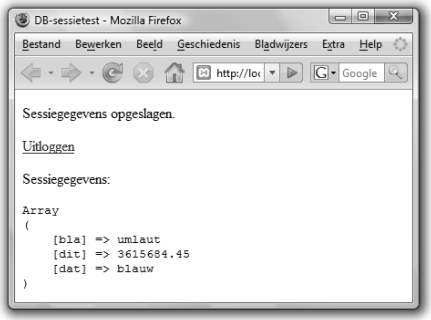
8. Roep de functie `session_write_close()` aan.

   ```
   <?php session_write_close(); ?>
   ```

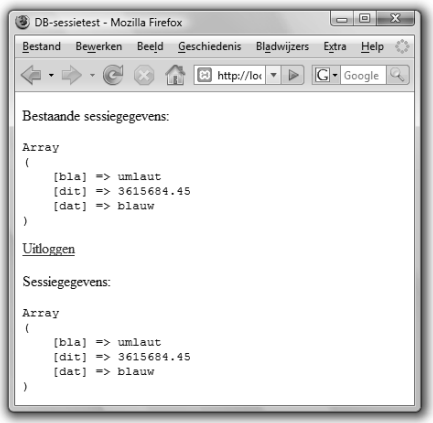
   Het maakt niet uit waar u deze functie in het script aanroept, zolang alle wijzigingen van de sessiegegevens maar voorbij zijn. Als u deze functie niet gebruikt, kunnen de resultaten er uitzien zoals in **figuur 3.6**.



Figuur 3.6 Omdat PHP zo aardig is om geopende databaseverbindingen te sluiten nadat een script is uitgevoerd, beschikken de functies `schrijf_sessie()` en `sluit_sessie()` – als ze later worden aangeroepen – niet meer over een databaseverbinding.



Figuur 3.7 Het resultaat wanneer de pagina voor het eerst laadt.



Figuur 3.8 Door de pagina opnieuw te laden, krijgt hij toegang krijgen tot de eerder opgeslagen sessiegegevens.

Figuur 3.9 Klikken op de hyperlink 'Uitloggen' leidt tot het vernietigen van de sessie.

9. Sla het bestand op als `sessies.php`, plaats het in uw webdirectory (in dezelfde map als `db_sessies.inc.php`) en test het in uw webbrowser (**Figuur 3.7**, **3.8** en **3.9**).

TIP U hebt de functie `session_write_close()` ook nodig als een site frames gebruikt. Door haar aan te roepen, kunt u de toegang van een pagina tot een sessie sluiten, zodat een andere pagina sneller laadt (omdat maar een script tegelijk dezelfde sessie kan gebruiken).

TIP U moet `session_write_close()` ook aanroepen voor u browsers doorverwijst met een aanroep van `header()`. Dit geldt alleen wanneer u uw eigen sessiehandlers gebruikt.

Aan de slag met Amerikaanse postcodes

Vaak heeft men op websites behoefte aan afstandberekeningen tussen adressen. Hoewel u altijd de volledige route kunt vinden via MapQuest of Google Maps, kunt u simpele afstandschattingen uitvoeren met alleen postcodes (ten minste, in de Verenigde Staten).

Elke postcode (*zip code*) heeft een geografische lengte en breedte. Neem twee van deze punten op aarde, gooi er wat ingewikkeld rekenwerk tegenaan en u hebt een redelijk precieze afstand. In deze paragraaf ziet u hoe u de benodigde postcodedata verkrijgt, maken we een tabel met winkels die een van de twee punten verschaft en kijken we vervolgens naar de formule die de afstanden berekent.

De postcodetabel maken

Dit voorbeeld is gebaseerd op een database met de breedte- en lengtegraden van alle Amerikaanse postcodes. Er zijn drie bronnen voor deze informatie:

- commerciële postcodedatabases;
- gratis postcodedatabases;
- gratis ZCTA-databases.

De eerste optie verschaft de meest accurate en actuele informatie, maar kost geld (meestal niet veel). De tweede optie is gratis (gratis!), maar moeilijker te vinden en mogelijk niet actueel. Op het moment van schrijven verschaft de website `www.cfdynamics.com/zipbase/` deze informatie, hoewel niet te zeggen valt hoelang nog. U kunt ook op het web zoeken naar alternatieven.

De laatste optie, ZCTA, de Zip Code Tabulation Areas, is een database die de Amerikaanse overheid heeft gemaakt voor eigen doeleinden. Deze database negeert ongeveer tienduizend postcodes die worden gebruikt door het Amerikaanse postbedrijf of bepaalde organisaties. Hij groepeert ook sommige postcodes en gebruikt tekens om andere te representeren. Voor het leeuwendeel van de postcodes voldoet deze informatie prima. Een bron voor de ZCTA-database is `http://zips.sourceforge.net`, die u op SourceForge.net vindt als u zoekt op 'zip code'.

De postcodedatabase maken

1. Zoek een gegevensbron.

   Welke bron (van de gegeven typen en specifieke voorbeelden) u gebruikt, hangt af van uw situatie. Hoe belangrijk is precisie? Wat wilt u uitgeven? Als secundaire overweging, welke bronnen zijn beschikbaar (zoek op het web en SourceForge)? In dit voorbeeld wordt de versie van `www.cfdynamics.com/zipbase/` gebruikt.

2. Maak de database (**figuur 3.10**).

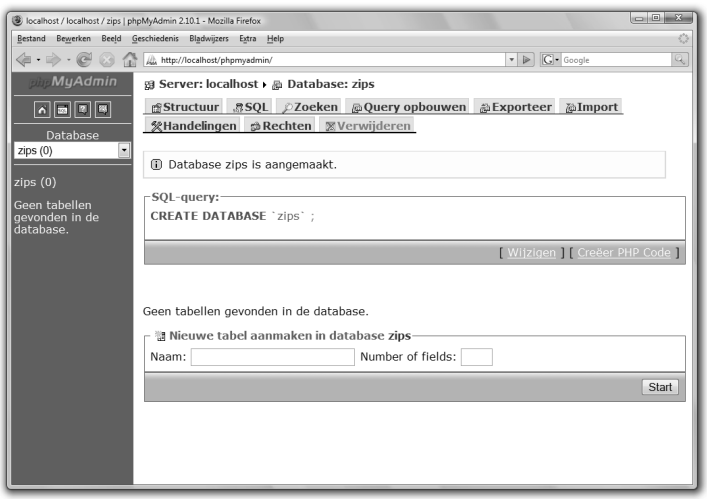
   ```
   CREATE DATABASE zips;
   ```

   Ik maak een MySQL-database genaamd `zips` in phpMyAdmin. U kunt de volgende stappen grotendeels voltooien met de opdrachtregelclient `mysql`, MySQL Administrator of een andere tool.

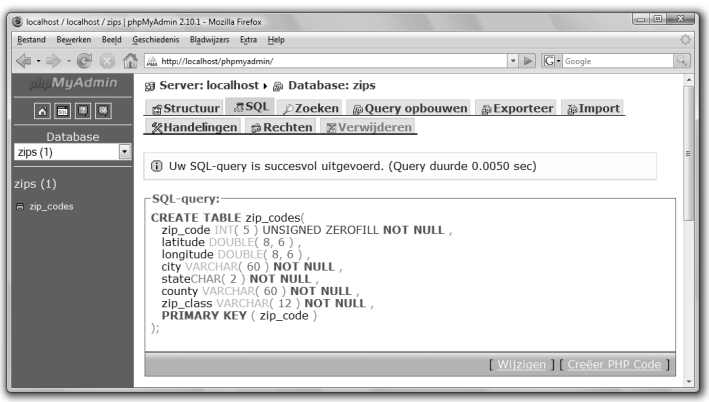
3. Maak een tabel die overeenkomt met de gegevens in het gegevensbestand (**figuur 3.11**).

   ```
   CREATE TABLE zip_codes (
   zip_code INT(5) UNSIGNED ZEROFILL NOT
   NULL,
   latitude DOUBLE(8,6),
   longitude DOUBLE(8,6),
   city VARCHAR(60) NOT NULL,
   state CHAR(2) NOT NULL,
   county VARCHAR(60) NOT NULL,
   zip_class VARCHAR(12) NOT NULL,
   PRIMARY KEY (zip_code)
   );
   ```

   Sommige bronnen verschaffen al de benodigde SQL-opdrachten om de tabel te maken en zelfs de gegevens in te voegen. In dat geval kunt u stap 3 en 4 overslaan. Anders moet u een tabel

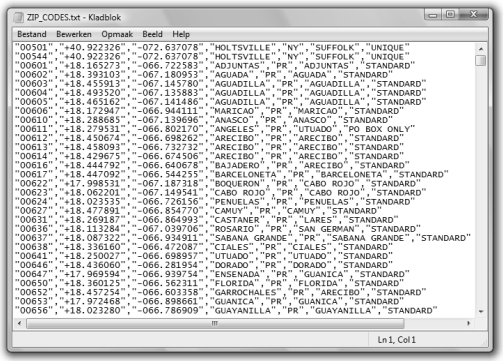


Figuur 3.10 Maak een nieuwe database voor dit voorbeeld.



Figuur 3.11 De hoofdtabel heeft een structuur die is gebaseerd op de in te voegen gegevens (figuur 3.12).

maken waarvan de structuur overeenkomt met de in te voegen gegevens. **Figuur 3.12** toont het bestand dat ik heb gedownload. De kolom `zip_code` met de postcode, die de primaire sleutel is, moet een unsigned integer van vijf cijfers met voorloopnullen zijn. De kolommen met de breedte- en lengtegraad moeten een zwevende-kommagetal zijn. Omdat, in dit voorbeeld, sommige records geen waarden hebben voor de breedtegraad en lengtegraad, kunnen deze kolommen niet worden gedefinieerd als `NOT NULL`. De voorbeeldgegevens hebben nog vier kolommen: de plaatsnaam, een afkorting van twee letters voor de staat, de gemeente en de postcodeklasse.



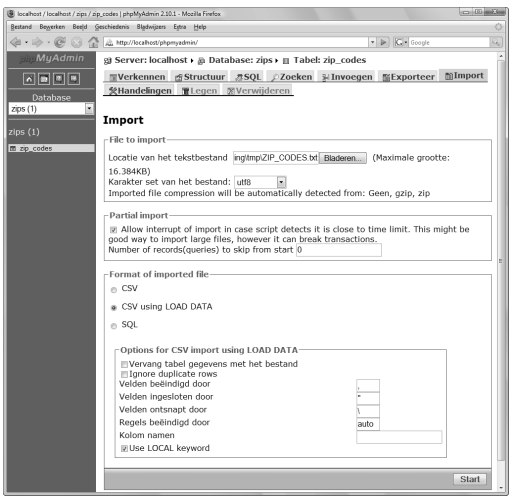
Figuur 3.12 Het gegevensbestand uit dit voorbeeld.

4. Importeer de gegevens (**figuur 3.13**).

```
LOAD DATA INFILE '/tmp/ZIP_CODES.txt'
INTO TABLE zip_codes
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n';
```

Het kan even duren voor u deze stap klaar hebt (misschien is phpMyAdmin ook geschikter hiervoor). De query `LOAD DATA INFILE` neemt de inhoud van een tekstbestand en plaatst deze in de gegeven tabel. Deze stap werkt alleen als het aantal kolommen in de tabel overeenkomt met het aantal waarden in elke rij in het tekstbestand. Misschien moet u ook de waarden `FIELDS TERMINATED BY`, `ENCLOSED BY` en `LINES TERMINATED BY` veranderen om overeen te komen met uw tekstbestand. In de MySQL-handleiding vindt u meer over deze syntaxis.

De naam van het tekstbestand moet overeenkomen met het absolute pad naar het bestand op uw computer.



Figuur 3.13 Importeer de gegevens naar de tabel.

5. Verwijder kolommen die u niet nodig hebt (**figuur 3.14**).

```
ALTER TABLE zip_codes
DROP COLUMN zip_class;
```

De gegevens van de postcodeklasse `zip_class` zaten in het gedownloade bestand, maar zijn eigenlijk overbodig.

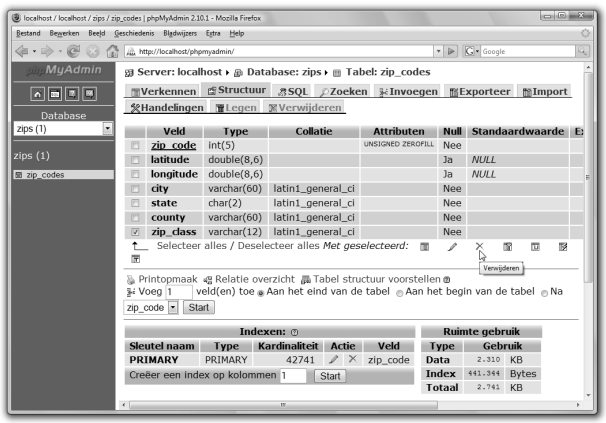
6. Voeg indices toe en update de gegevens, indien nodig.

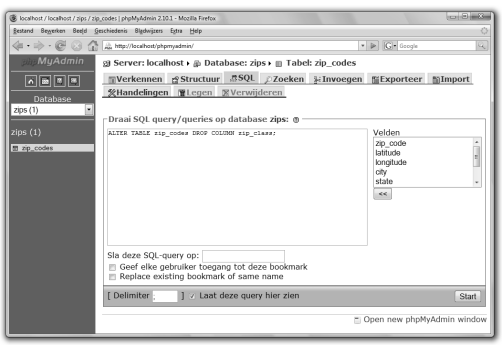
   Ik heb de lege velden voor de breedte- en lengtegraad gevuld met formeel betere `NULL`-waarden. Die query is:

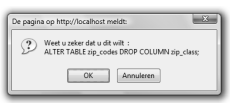
   ```
   UPDATE zip_codes
   SET latitude=NULL, longitude=NULL
   WHERE latitude='';
   ```

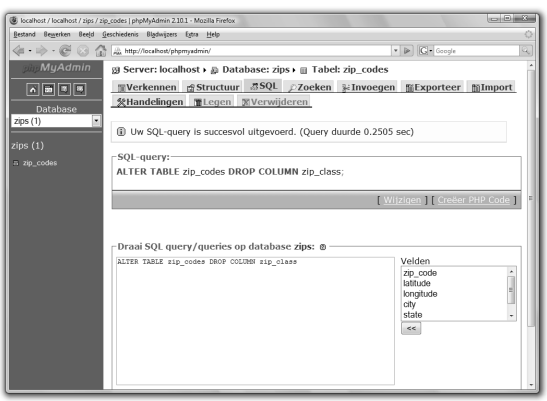
7. Controleer enige postcode-informatie (**figuur 3.15**).

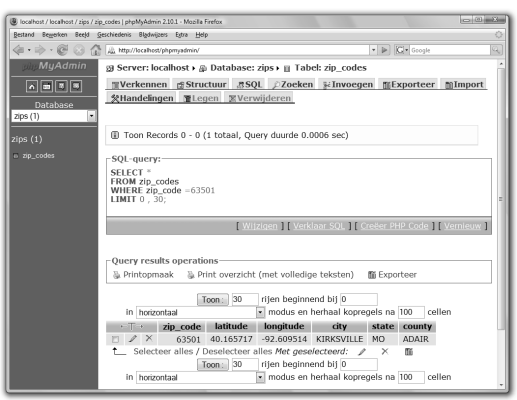
   ```
   SELECT * FROM zip_codes
   WHERE zip_code=63501;
   ```











Figuur 3.14 Verwijder overbodige kolommen.

Figuur 3.15 De informatie voor een postcode.

De winkeltabel maken

Na het maken van de postcodetabel, is het tijd om de andere benodigde tabel te maken. Voor dit voorbeeld willen we de afstand kunnen berekenen tussen een gegeven postcode (zoals het huisadres van een gebruiker) en een lijst met winkels. Daarvoor is een tabel `stores` nodig.

U kunt in de tabel zetten wat u wilt. Hij lijkt waarschijnlijk op **tabel 3.3**.

Omdat de plaats en staat gekoppeld zijn aan de postcode en die informatie al in de tabel `zip_codes` staat, kunnen die kolommen verdwijnen. De kolom `Address2` krijgt ook de standaardwaarde `NULL`, omdat dit veld niet nodig is voor alle winkeladressen.

De winkeltabel maken

1. Open de database `zips` met phpMyAdmin, de client `mysql` of een andere interface.
2. Maak de tabel `stores` (**figuur 3.16**).

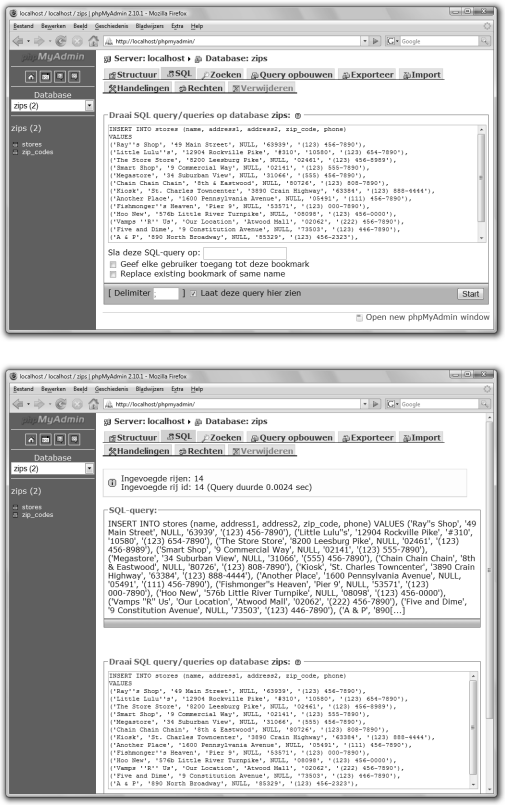
```
CREATE TABLE stores (
store_id SMALLINT(5) UNSIGNED NOT NULL
→AUTO_INCREMENT,
name VARCHAR(60) NOT NULL,
address1 VARCHAR(100) NOT NULL,
address2 VARCHAR(100) DEFAULT NULL,
zip_code INT(5) UNSIGNED ZEROFILL NOT
→NULL,
phone VARCHAR(15) NOT NULL,
PRIMARY KEY (store_id),
KEY (zip_code)
);
```

De tabel heeft het gegevensmodel uit tabel 3.3, alleen zijn de plaats en staat weggelaten (want die staan in de tabel `zip_codes`). De kolom `zip_code` moet hier precies zo worden gedefinieerd als in de tabel `zip_codes`, omdat beide tabellen worden samengevoegd (zie het kader 'Samenvoegingen optimaliseren' aan het einde van deze paragraaf).

Figuur 3.16 De tweede en laatste tabel maken.

Tabel 3.3 Een voorbeeldrecord voor een winkel.

Winkelinformatie	
Kolom	**Voorbeeld**
Name	Ray's Shop
Address 1	49 Main Street
Address 2	Suite 230
City	Arlington
State	Virginia
Zip Code	22201
Phone	(123) 456-7890



Figuur 3.17 Plaats enige voorbeeldrecords in de tabel `stores`.

3. Vul de tabel `stores` (**figuur 3.17**).

```
INSERT INTO stores (name, address1,
→address2, zip_code, phone)
VALUES
('Ray''s Shop', '49 Main Street', NULL,
→'63939', '(123) 456-7890'),
→('Little Lulu''s', '12904 Rockville
→ Pike', '#310', '10580', '(123) 654-
→ 7890'),
('The Store Store', '8200 Leesburg Pike',
→NULL, '02461', '(123) 456-8989'),
('Smart Shop', '9 Commercial Way', NULL,
→'02141', '(123) 555-7890'),
('Megastore', '34 Suburban View', NULL,
→'31066', '(555) 456-7890'),
('Chain Chain Chain', '8th & Eastwood',
→NULL, '80726', '(123) 808-7890'),
('Kiosk', 'St. Charles Towncenter',
→'3890 Crain Highway', '63384', '(123)
→888-4444'),
('Another Place', '1600 Pennsylvania
→Avenue', NULL, '05491', '(111) 456-
→7890'),
('Fishmonger''s Heaven', 'Pier 9', NULL,
→'53571', '(123) 000-7890'),
('Hoo New', '576b Little River Turnpike',
→NULL, '08098', '(123) 456-0000'),
('Vamps ''R'' Us', 'Our Location',
→'Atwood Mall', '02062', '(222) 456-
→7890'),
('Five and Dime', '9 Constitution
→Avenue', NULL, '73503', '(123) 446-
→7890'),
('A & P', '890 North Broadway', NULL,
→'85329', '(123) 456-2323'),
('Spend Money Here', '1209 Columbia Pike',
→NULL, '10583', '(321) 456-7890');
```

U kunt zelf bepalen wat voor records u invoert. U vindt deze SQL-query ook in de download bij dit boek (*www.pearsoneducation.nl*).

4. Selecteer het volledige adres van twee winkels (**Figuur 3.18**).

```
SELECT stores.*, zip_codes.city, zip_
→codes.state
FROM stores
LEFT JOIN zip_codes USING (zip_code)
→LIMIT 2;
```

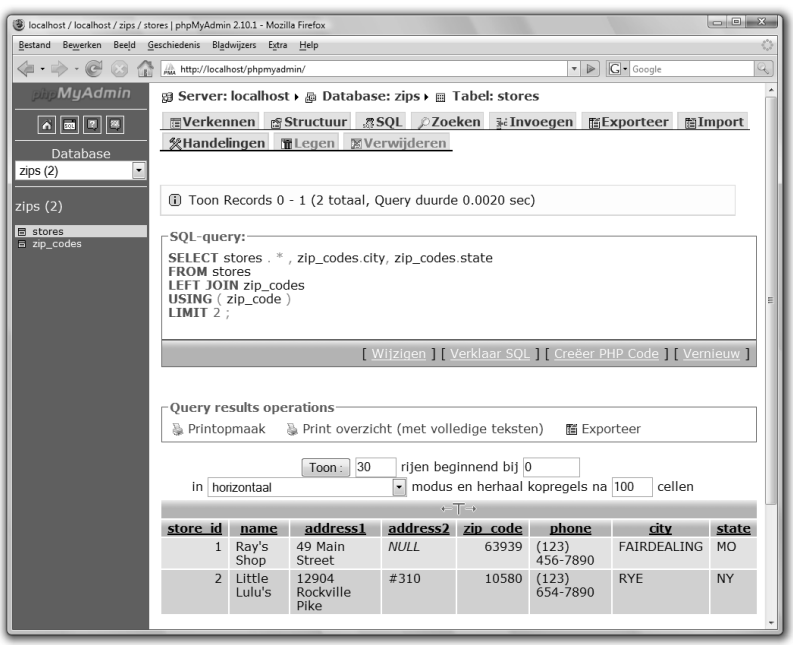
Om van een winkel het volledige adres te krijgen, inclusief stad en staat, moet u beide tabellen samenvoegen via de kolom `zip_code`, die in beide tabellen voorkomt. Als u hier niet bekend mee bent, kunt u het afsluitende LIMIT 2`\G` gebruiken in de client `mysql` om de resultaten te retourneren in verticale groepen in plaats van in horizontale rijen.

Samenvoegingen optimaliseren

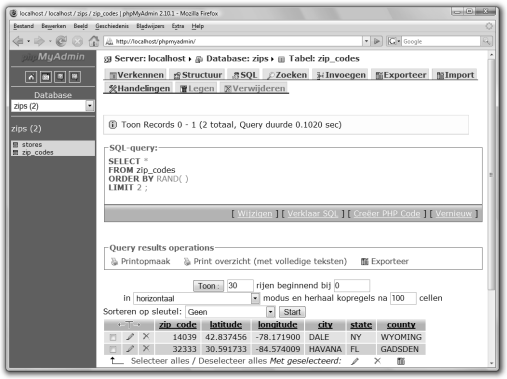
De MySQL-database doet veel om de efficiëntie te verbeteren, vaak zonder dat de doorsnee gebruiker dit beseft. Dit kan bestaan uit het veranderen van de kolomdefinitie of het stiekem veranderen hoe een query wordt uitgevoerd. Maar soms heeft MySQL wat hulp nodig.

Een samenvoeging is een dure query (in termen van databasebronnen), omdat ze voorwaardelijke overeenkomsten nodig heeft om tot stand te komen tussen twee of meer tabellen. In dit voorbeeld vindt een samenvoeging plaats tussen de tabellen `zip_codes` en `stores`, via de kolom `zip_code` van beide. Om MySQL deze samenvoegingen sneller te laten uitvoeren, moet u twee dingen doen.

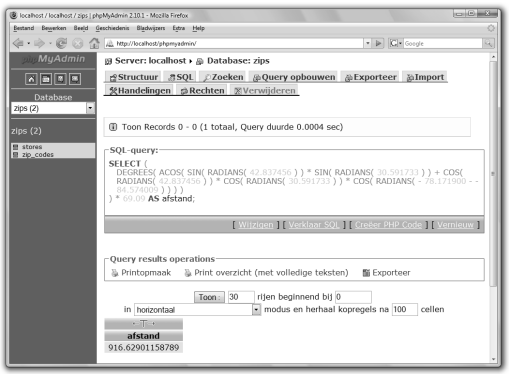
Ten eerste moet een index bestaan voor beide kolommen. Ten tweede moeten beide kolommen op precies dezelfde manier worden gedefinieerd. Als een kolom een `TINYINT` is en de ander een `INT`, gebruikt MySQL helemaal geen indices (en dat is niet goed).



Figuur 3.18 Door twee tabellen samen te voegen, kunt u een volledig winkeladres ophalen.



Figuur 3.19 Om de afstand tussen twee punten te controleren, selecteert u de informatie van twee willekeurige postcodes.



Figuur 3.20 Het resultaat van de afstandberekening, met de breedte- en lengtegraden uit figuur 3.19.

Afstanden berekenen

Nu we twee tabellen met gegevens hebben, wordt het tijd om de afstanden te berekenen. In PHP is de formule daarvoor:

```
$afstand = sin(deg2rad($a_latitude)) *
→sin(deg2rad($b_latitude)) + cos(deg2rad
→($a_latitude)) * cos(deg2rad($b_latitude))
→* cos(deg2rad($a_longitude -
→$b_longitude));
$afstand = (rad2deg(acos($distance))) *
→69.09;
```

Ik zou de formule in detail kunnen uitleggen, alleen begrijp ik hem niet echt (eerlijk gezegd heb ik me er nooit in verdiept). Ik weet alleen dat hij werkt en soms is dat genoeg.

In MySQL luidt dezelfde formule (die een paar andere functies vereist):

```
SELECT (DEGREES(ACOS(SIN(RADIANS(lat_a))
* SIN(RADIANS(lat_b))
+ COS(RADIANS(lat_a))
* COS(RADIANS(lat_b))
* COS(RADIANS(long_a - long_b)))))
→* 69.09;
```

Als we bijvoorbeeld de breedte- en lengtegraad nemen van twee willekeurige postcodes (**figuur 3.19**), retourneert deze berekening een waarde van ruim 916 mijl (**figuur 3.20**).

```
SELECT (DEGREES(ACOS(SIN(RADIANS(42.837456))
* SIN(RADIANS(30.591733))
+ COS(RADIANS(42.837456))
* COS(RADIANS(30.591733))
* COS(RADIANS(-78.171900 - -
84.574009))))) * 69.09 AS afstand;
```

Om al deze kennis in de praktijk te brengen, maken we een PHP-script dat de drie dichtstbijzijnde winkels retourneert voor een gegeven postcode.

Afstanden berekenen met MySQL

1. Begin een nieuw PHP-script in uw teksteditor of IDE en begin met de XHTML (**Script 3.3**).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD
→XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/
→xhtml" xml:lang="nl" lang="nl">
<head>
  <meta http-equiv="content-type"
→content="text/html; charset=iso-8859-1"
/>
  <title>Afstanden berekenen</title>
  <style type="text/css" title="text/
→css" media="all">
    .fout { color: #f30 }
    h3 { color: #00f }
  </style>
</head>
<body>
<?php # Script 3.3 - afstand.php
```

Twee CSS-klassen worden gebruikt om de pagina enige opmaak te geven.

Script 3.3 (afstand.php) Dit PHP-script retourneert de drie dichtstbijzijnde winkels voor een gegeven postcode met een berekening van de afstand.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="nl" lang="nl">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
6     <title>Afstanden berekenen</title>
7     <style type="text/css" title="text/css" media="all">
8       .fout { color: #f30 }
9       h3 { color: #00f }
10    </style>
11  </head>
12  <body>
13  <?php # Script 3.3 - afstand.php
14
15  /**
16   * Deze pagina gebruikt de database zips om de afstand
17   * te berekenen tussen een gegeven punt en enkele winkels.
18   * De drie dichtstbijzijnde winkels worden geretourneerd.
19   */
20
21  $postcode = 64154; // Postcode van de gebruiker.
22
23  // Een kop weergeven:
24  echo "<h2>Dichtstbijzijnde winkels voor $postcode:</h2>\n";
25
26  // Maak verbinding met de database:
27  $dbc = @mysqli_connect('localhost', 'gebruikersnaam', 'wachtwoord', 'zips') OR die ('<p
    class="fout">Databaseverbinding kon niet worden gemaakt.</body></html>');
28
29  // Haal de geografische breedte en lengte van het vertrekpunt op:
30  $q = "SELECT latitude, longitude FROM zip_codes WHERE zip_code='$postcode' AND latitude
    IS NOT NULL";
31  $r = mysqli_query($dbc, $q);
32
```

Script 3.3 Vervolg

```
// Haal de resultaten op:
if (mysqli_num_rows($r) == 1) {

    list($breedtegraad, $lengtegraad) = mysqli_fetch_array($r, MYSQLI_NUM);

    // Grote, complexe en niet om woorden verlegen hoofdquery:
    $q = "SELECT name, CONCAT_WS('<br />', address1, address2), city, state, stores.
zip_code, phone,
          ROUND(DEGREES(ACOS(SIN(RADIANS($breedtegraad))
          * SIN(RADIANS(latitude))
          + COS(RADIANS($breedtegraad))
          * COS(RADIANS(latitude))
          * COS(RADIANS($lengtegraad - longitude)))))
          * 69.09
          AS afstand
          FROM stores
          LEFT JOIN zip_codes
          USING (zip_code)
          ORDER BY afstand ASC
          LIMIT 3";
    $r = mysqli_query($dbc, $q);

    if (mysqli_num_rows($r) > 0) {

        // Toon de winkels:
        while ($rij = mysqli_fetch_array($r, MYSQLI_NUM)) {
            echo "<h3>$rij[0]</h3>
<p>$rij[1]<br />" . ucfirst(strtolower($rij[2])) . ", $rij[3] $rij[4]<br />
$rij[5] <br />
(circa $rij[6] mijl)</p>\n";

        } // Einde van de WHILE-lus.

    } else { // Geen winkels gevonden.

        echo '<p class="fout">De zoekopdracht leverde geen winkels op.</p>';

    }

} else { // Ongeldige postcode.

    echo '<p class="fout">Er is een ongeldige postcode ingevoerd.</p>';

}

// Sluit de verbinding:
mysqli_close($dbc);

?>
</body>
</html>
```

2. Zoek het vertrekpunt.

```
$postcode = 64154; // Postcode van de
→gebruiker.
echo "<h2>Dichtstbijzijnde winkels voor
→$postcode:</h2>\n"
```

   Deze waarde kan ook via een formulier worden verkregen (na validatie, natuurlijk).

3. Maak verbinding met de database.

```
$dbc = @mysqli_connect('localhost',
→'gebruikersnaam', 'wachtwoord',
→'zips') OR die ('<p
→class="fout">Databaseverbinding kon
→niet worden gemaakt.</body></html>');
```

4. Definieer een query en voer hem uit.

```
$q = "SELECT latitude, longitude FROM
→zip_codes WHERE zip_code='$postcode'
→AND latitude IS NOT NULL";
$r = mysqli_query($dbc, $q);
```

   Deze eerste query (het script heeft er twee) valideert de postcode (of het inderdaad een Amerikaanse postcode is) en haalt de geografische breedte en lengte van die postcode op. Die informatie is nodig voor het berekenen van de afstanden tussen de gegeven postcode en de winkels. Omdat de hier gebruikte niet bij elke postcode een breedte- en lengtegraad bevat, is de voorwaarde `AND latitude IS NOT NULL` toegevoegd aan de component `WHERE`. Dit hoeft niet voor alle gegevenssets nodig te zijn.

5. Haal de resultaten van de query op.

```
if (mysqli_num_rows($r) == 1) {
    list($breedtegraad, $lengtegraad) =
→mysqli_fetch_array($r, MYSQLI_NUM);
```

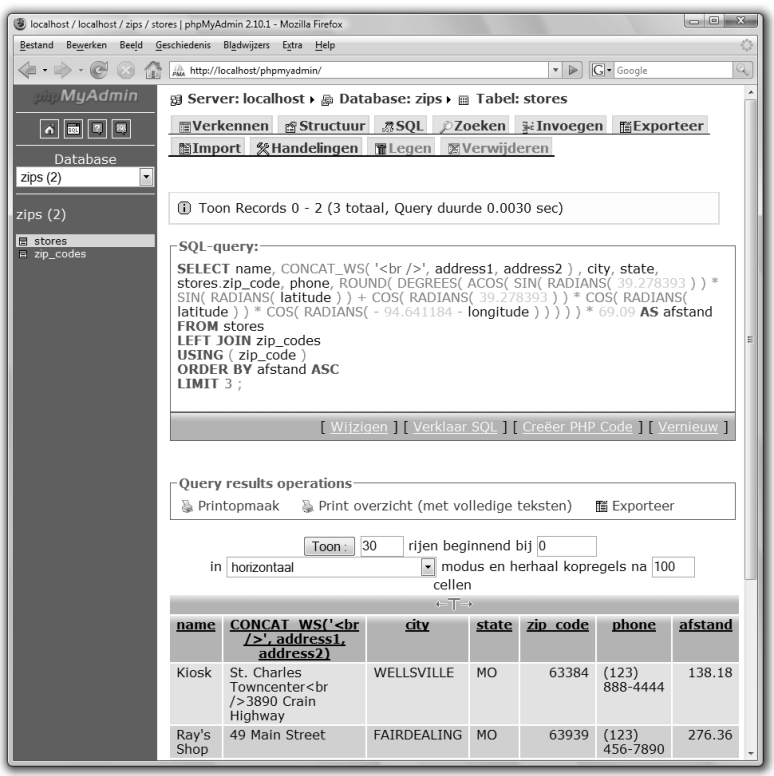
   Als een rij is geretourneerd, is de postcode geldig en wordt de geretourneerde informatie toegewezen aan deze twee variabelen.

6. Voer de hoofdquery uit.

```
$q = "SELECT name, CONCAT_WS('<br />',
→address1, address2), city, state,
→stores.zip_code, phone,
→ROUND(DEGREES(ACOS(SIN(RADIANS
→($breedtegraad))
      * SIN(RADIANS(latitude))
      + COS(RADIANS($breedtegraad))
      * COS(RADIANS(latitude))
      * COS(RADIANS($lengtegraad -
→longitude)))))
      * 69.09
      AS afstand
      FROM stores
      LEFT JOIN zip_codes
      USING (zip_code)
      ORDER BY afstand ASC
      LIMIT 3";
$r = mysqli_query($dbc, $q);
```

Deze hoofdquery is eigenlijk de kern van dit hele script. Zoals u in **figuur 3.21** ziet, retourneert deze query de naam, het volledige adres, het telefoonnummer en de afstand vanaf de gegeven postcode. De twee adresregels worden aaneengevoegd met `CONCAT_WS()`, die een `<br />` plaatst tussen de regels als `address2` een waarde heeft, maar anders alleen `address1` retourneert. De plaats- en staatnamen zijn afkomstig uit de tabel `zip_codes` en de `zip_code` kan uit een van beide tabellen komen. Het telefoonnummer wordt ook geretourneerd.

De grote, complexe berekening wordt ook geselecteerd. Voor de geografische breedte en lengte 'A' worden de waarden van de oorspronkelijke postcode gebruikt (al opgehaald door de eerdere query). Voor de breedte en lengte 'B' worden waarden uit deze query gebruikt. Slechts drie winkels worden geretourneerd en ze worden gesorteerd op de afstand, van klein naar groot.



Figuur 3.21 Het resultaat van de nogal omslachtige hoofdquery.

7. Geef de resultaten weer.

```
if (mysqli_num_rows($r) > 0) {
    while ($rij = mysqli_fetch_array($r,
→MYSQLI_NUM)) {
        echo "<h3>$rij[0]</h3>
<p>$rij[1]<br />" .
→ucfirst(strtolower($rij[2])) . ",
→$rij[3] $rij[4]<br />
$rij[5] <br />
(circa $rij[6] mijl)</p>\n";
    }
} else {
    echo '<p class="fout">De zoekopdracht
→leverde geen winkels op.</p>';
}
```

De resultaten worden weergegeven met een minimum aan opmaak. Als om welke reden dan ook geen winkel werd geretourneerd (wat niet zou moeten), verschijnt daarvoor een foutmelding (**figuur 3.22**).

8. Maak de conditie af die in stap 5 begon.

```
} else {
    echo '<p class="fout">Er is een
→ongeldige postcode ingevoerd.</p>';
}
```

Dit bericht verschijnt als een ongeldige postcode is opgegeven (**figuur 3.23**).

Figuur 3.22 Omdat er geen maximum is ingesteld voor de afstand tot een winkel, mag deze foutmelding nooit verschijnen. Toch is het beter om het zekere voor het onzekere te nemen.

Figuur 3.23 Het resultaat wanneer een ongeldige postcode wordt gebruikt (hier *77777*).

Figuur 3.24 De dichtstbijzijnde winkels voor postcode 64154.

Figuur 3.25 De dichtstbijzijnde winkels voor postcode 01026.

9. Voltooi de pagina.

```
mysqli_close($dbc);
?>
</body>
</html>
```

10. Sla het bestand op als `afstand.php`, plaats het in uw webdirectory en test het in uw webbrowser (**figuur 3.24**).
11. Verander de postcode en test opnieuw (**figuur 3.25**).

    Een postcode die begint met 0, moet u tussen aanhalingstekens plaatsen:

    ```
    $postcode = '01026';
    ```

    Als u dit niet doet, denkt PHP dat u een andere getalvorm gebruikt en vertaalt deze.

TIP Hoofdstuk 13, 'Ajax', gebruikt dit voorbeeld om de magie van Ajax te demonstreren.

TIP U kunt gemakkelijk het aantal geretourneerde winkels tot een bepaald gebied beperken door `WHERE afstand<=X` toe te voegen aan de hoofdquery.

Opgeslagen functies maken

Stored functions of *opgeslagen functies* vormen de helft van een groter concept genaamd *stored routines* of *opgeslagen routines*. (De andere helft bestaat uit *stored procedures* of *opgeslagen procedures*.) Via opgeslagen routines, onderdeel van vele databaseapplicaties, maar nieuw in MySQL sinds versie 5, kunt u een vaste codesequentie in de MySQL-server opslaan en deze sequentie aanroepen wanneer dat nodig is. U kunt het zien als het schrijven van uw eigen PHP-functies, maar dan in SQL.

Het onderwerp van opgeslagen routines is uitgebreid, maar ik geef u hier alvast een voorproefje. Meer over dit onderwerp vindt u in de MySQL-handleiding of het boek *MySQL: Visual QuickStart Guide* (Peachpit Press, 2006). Voor dit voorbeeld hebt u MySQL 5 of hoger nodig (of een databaseapplicatie die dezelfde functionaliteit ondersteunt, zoals PostgreSQL).

De basissyntaxis voor het maken van een opgeslagen functie is:

```
CREATE FUNCTION functienaam (argumenten)
→RETURNS type code
```

Gebruik voor de naam van de functie geen bestaande keywords, SQL-termen of functienamen. Zoals met de meeste dingen die u declareert in MySQL, moet u zich beperken tot alfanumerieke tekens en het onderstrepingsteken.

De argumentensectie wordt gebruikt om waarden door te geven aan de routine. De vermelde argumenten worden benoemd en krijgen typen die overeenkomen met de beschikbare gegevenstypen in MySQL:

```
CREATE FUNCTION mijnfunc (mijnvar1 INT,
→mijnvar2 CHAR) RETURNS type code
```

Lokale variabelen definiëren

Opgeslagen routines zijn net kleine programma's en u kunt ze zelfs hun eigen variabelen geven. Hiervoor gebruikt u het statement `DECLARE`:

```
DECLARE var_naam var_type
```

De naamgevingregels zijn bijna hetzelfde als altijd, maar u moet er absoluut voor zorgen dat uw variabelen uniek ID hebben. De typen komen overeen met de gegevenstypen van MySQL:

```
DECLARE var1 INT
DECLARE var2 DECIMAL(5,2)
DECLARE var3 VARCHAR(20)
```

De enige beperkingen bij het definiëren van variabelen zijn:

- De definities moeten plaatsvinden binnen een codeblok `BEGIN...END`.
- De definities moeten plaatsvinden voor andere statements (dat wil zeggen, definities moeten onmiddellijk na `BEGIN` komen).

Zodra u een variabele hebt gedefinieerd, kunt u een waarde eraan toewijzen via `SET`:

```
SET naam = waarde
```

Merk ook op dat deze variabelen van opgeslagen routines in tegenstelling tot variabelen in PHP niet beginnen met een dollarteken.

De codesectie van deze syntaxis is het belangrijkst. Omdat routines gewoonlijk bestaan uit meerdere regels, moet u een blok maken met `BEGIN` en `END`:

```
CREATE FUNCTION naam (argumenten) RETURNS
→type BEGIN
statement1;
statement2;
END
```

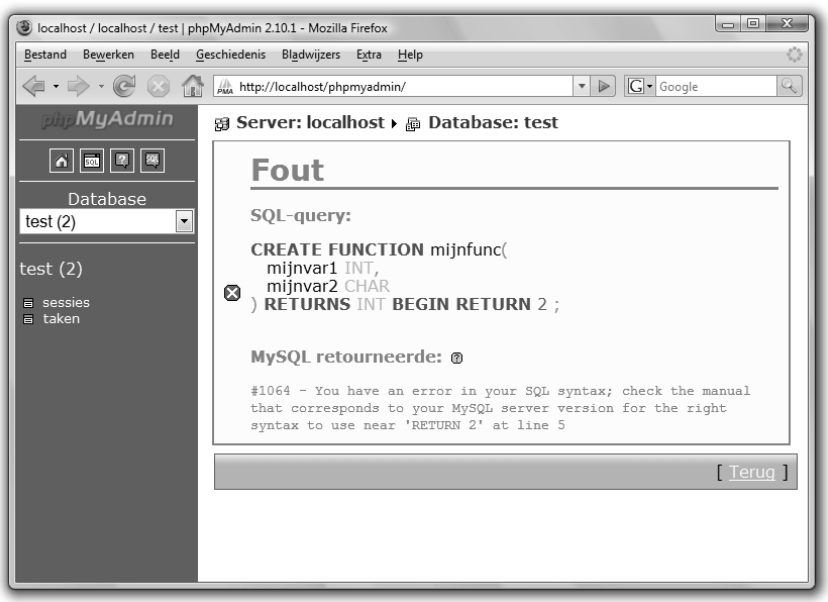
Binnen het codeblok eindigt elk statement met een puntkomma. Dit kan problemen geven: wanneer u deze opgeslagen functie toevoegt met clients zoals phpMyAdmin en `mysql`, denken deze dat de puntkomma het einde aangeeft van een direct uit te voeren opdracht (**figuur 3.26**). Om dit te voorkomen, kunt u het scheidingsteken (de puntkomma) veranderen in iets anders. Een andere optie is om MySQL Administrator te gebruiken; instructies hiervoor volgen zo meteen.

Opgeslagen functies moeten een component `RETURNS` hebben die het type waarde aangeeft dat de functie retourneert. Functies retourneren scalaire (enkelvoudige) waarden, als getal of als string. Hiervoor gebruikt u

```
RETURNS data
```

binnen de codebody van de functie. Het geretourneerde gegevenstype moet overeenkomen met het type dat is aangeduid in de openingsregel van de functiedefinitie. U kunt geen lijst met waarden retourneren vanuit een opgeslagen functie, maar omdat opgeslagen functies scalaire waarden retourneren, kunnen ze worden gebruikt in een query, net als bestaande MySQL-functies.

Al deze informatie, en de info in het begeleidende kader 'Lokale variabelen definiëren', vormt de gids in vijf minuten tot opgeslagen functies. In de volgende serie stappen ziet u hoe u de ingewikkelde afstandsformule (script 3.3) omzet naar een aanroepbare opgeslagen functie.



Figuur 3.26 U moet oppassen wanneer u opgeslagen routines probeert te maken met clients zoals phpMyAdmin.

Een opgeslagen functie gebruiken

1. Download en installeer MySQL Administrator.

   Deze gratis tool is onderdeel van het pakket GUI Tools van MySQL, een geweldige applicatiesuite. Er zijn versies beschikbaar voor de meeste besturingssystemen (Windows, Mac OS X en Linux en u kunt indien nodig uw eigen versie compileren.

2. Start de applicatie.

   Gebruikers van Windows kunnen de snelkoppeling in het menu Start gebruiken. Gebruikers van Mac OS kunnen op de applicatie zelf dubbelklikken.

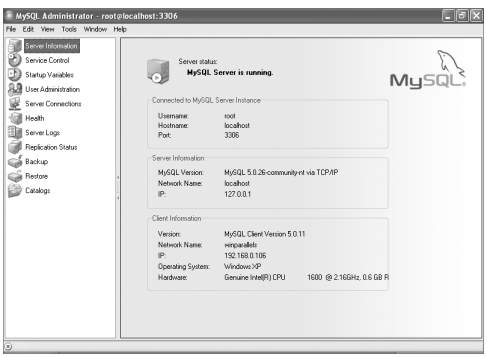
3. Typ in de eerste prompt (**figuur 3.27**) de juiste combinatie van gebruikersnaam, hostnaam en wachtwoord.

   Als MySQL wordt uitgevoerd op dezelfde computer, moet u waarschijnlijk *localhost* als host invoeren. U moet vervolgens *root* of een ander beheeraccount gebruiken, en het juiste gebruikerswachtwoord. Deze waarden komen overeen met de gebruikers en machtigingen die zijn vastgesteld binnen de MySQL-server.

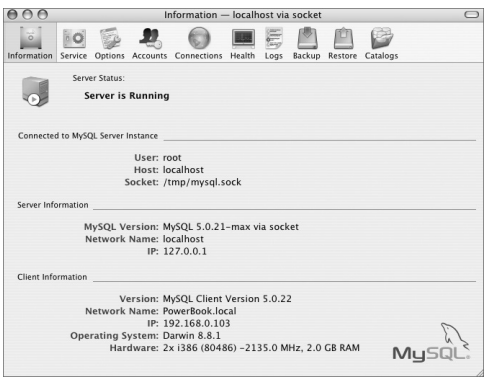
4. Klik op OK of Connect, afhankelijk van de applicatieversie, om de applicatie binnen te gaan.

   Als u een geldige combinatie van hostnaam, gebruikersnaam en wachtwoord hebt gebruikt, maakt u nu verbinding met de MySQL-server. **Figuur 3.28** toont het resultaat voor Windows; **Figuur 3.29** voor Mac OS X.

Figuur 3.27 De verbindingsprompt van MySQL Administrator, waar u de toegangsinformatie invoert voor de databaseserver waarmee u gaat werken.

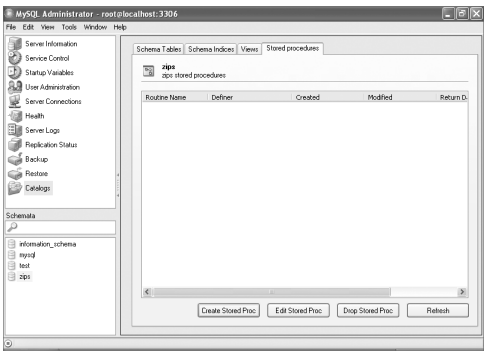


Figuur 3.28 MySQL Administrator voor Windows, direct na de aanmelding.

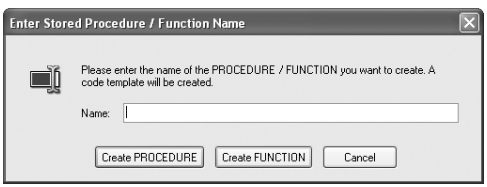


Figuur 3.29 MySQL Administrator voor Mac OS X.

Figuur 3.30 De weergave van de database `zips`.



Figuur 3.31 De volgende stap in Windows is klikken op Stored procedures.



Figuur 3.32 In deze prompt klikt u op Create FUNCTION.

5. Klik op Catalogs.

   De sectie Catalogs is waar u databases kunt bekijken en bewerken.

6. Klik op de database `zips` in de kolom Schemata (**Figuur 3.30**).

   Het is niet duidelijk waarom ze niet gewoon *Databases* zeggen in plaats van *Schemata* (of *Catalogs*), maar...

7. Zoek de editor voor opgeslagen functies

   De werking van MySQL Administrator op de diverse besturingssystemen is verbazend gevarieerd. In Windows doet u het volgende:

   1. Klik op Stored procedures (**figuur 3.31**).
   2. Klik op Create Stored Proc.
   3. Klik op Create FUNCTION in het volgende dialoogvenster (**figuur 3.32**).

In Mac OS X doet u dit:

1. Klik op Functions (**figuur 3.33**).
2. Klik op Create Function.

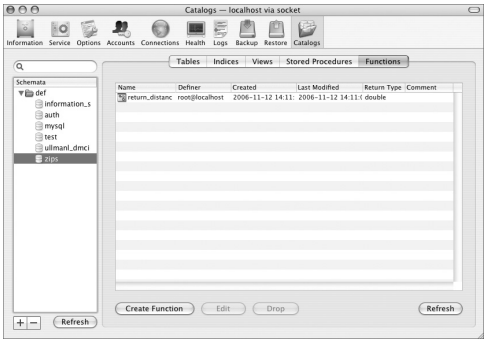
8. In het volgende venster typt u (**figuur 3.34**):

```
CREATE FUNCTION zips.return_distance
(lat_a DOUBLE, long_a DOUBLE,
lat_b DOUBLE, long_b DOUBLE)
RETURNS DOUBLE
BEGIN
    DECLARE distance DOUBLE;
    SET distance =
    SIN(RADIANS(lat_a)) *
    SIN(RADIANS(lat_b))
    + COS(RADIANS(lat_a))
    * COS(RADIANS(lat_b))
    * COS(RADIANS(long_a - long_b));
    RETURN((DEGREES(ACOS(distance))) *
→69.09);
END
```

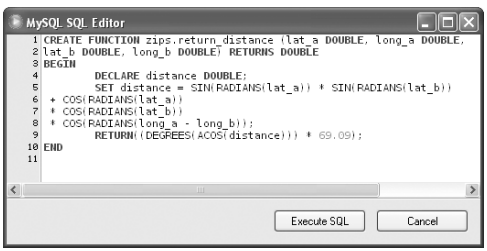
Deze code verpakt de ingewikkelde berekening in een opgeslagen functie. De functie heet *return_distance*. De syntaxis *databasenaam.functienaam* koppelt de functie aan een specifieke database (hier `zips`). De functie accepteert vier argumenten, allemaal van het type `DOUBLE`. Ze retourneert ook een variabele van het type `DOUBLE`.

De eerste stap in de functie is een variabele maken van het type `DOUBLE`. Dit maakt de berekening wat eenvoudiger. De variabele krijgt de waarde toegewezen van het merendeel van de berekening. Deze variabele wordt dan door nog een paar functies en enige bewerkingen gevoerd en vervolgens geretourneerd.

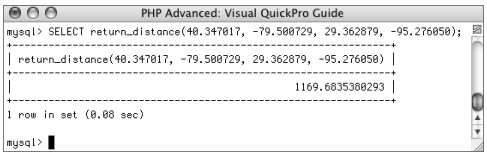
9. Klik op OK (Mac OS X) of op Execute SQL (Windows).



Figuur 3.33 Klik in het deelvenster Catalogs van Mac OS X op Functions.



Figuur 3.34 Gebruik SQL Editor (voor Windows) om de opgeslagen functie in te voeren.



Figuur 3.35 De opgeslagen functie wordt gebruikt om de SQL-query te vereenvoudigen.

10. Test de functie door de volgende query uit te voeren in de client `mysql` (**figuur 3.35**).

```
SELECT return_distance(40.347017,
→-79.500729, 29.362879,
→-95.276050);
```

    Dit is dezelfde query die werd uitgevoerd voor figuur 3.20, alleen roept hij nu de opgeslagen functie aan.

11. Als u wilt, kunt u `afstand.php` (script 3.3) wijzigen om de opgeslagen procedure aan te roepen.

    Hiertoe wijzigt u de hoofdquery in:

```
SELECT name, CONCAT_WS('<br />',
→address1, address2), city, state,
→stores.zip_code, phone,
→ROUND(return_distance($breedtegraad,
→$lengtegraad, latitude, longitude))
AS afstand
FROM stores
LEFT JOIN zip_codes
USING (zip_code)
ORDER BY afstand ASC
LIMIT 3
```

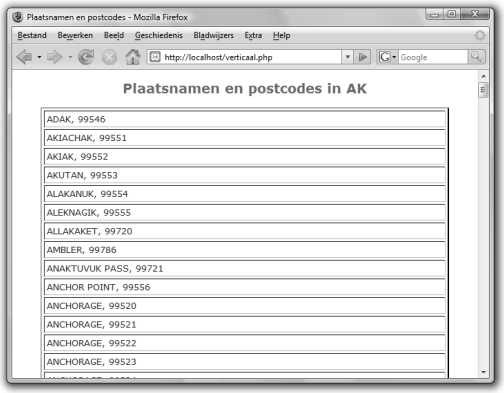
TIP Alle opgeslagen routines zijn gekoppeld aan een specifieke database. Dit heeft als bijkomend voordeel dat u de database niet hoeft te selecteren (`USE databasenaam`) bij hun aanroep. Dit betekent ook dat u een opgeslagen routine geen database kunt laten selecteren.

TIP Omdat opgeslagen routines zijn gekoppeld aan databases, worden bij het verwijderen van de database ook alle gekoppelde opgeslagen routines verwijderd.

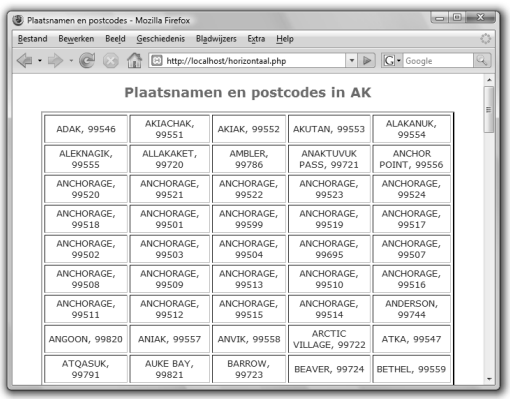
Resultaten horizontaal weergeven

Een van de veelgestelde vragen die ik krijg, betreft het horizontaal weergeven van resultaten van een query. Het is vrij gemakkelijk om resultaten op te halen en ze verticaal weer te geven (**figuur 3.36**), maar het verzorgen van een uitvoer als in **figuur 3.37** gaat sommige programmeurs boven de pet. Om de uitvoer op deze manier te verzorgen, gebruikt de code in figuur 3.37 een HTML-tabel met vijf records per rij.

Om dit te doen met PHP, is een *counter* of *teller* nodig die bijhoudt hoeveel records in een rij zijn geplaatst. Bij nul records moet een nieuwe rij worden gestart. Wanneer het maximumaantal records is geplaatst, moet de oude rij worden afgesloten. Dat wordt het doel van ons volgende script. Voor de gegevens gebruiken we de tabel `zip_codes` in de database `zips` (maar u kunt ook iets anders gebruiken).



Figuur 3.36 Een traditionele verticale weergave van records.



Figuur 3.37 Dezelfde gegevens als in figuur 3.36, weergegeven in een tabelvorm.

Resultaten horizontaal weergeven

1. Begin een nieuw PHP-script in uw teksteditor of IDE en start met de XHTML (**script 3.4**).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
→1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/
→xhtml" xml:lang="nl" lang="nl">
<head>
  <meta http-equiv="content-type"
→content="text/html; charset=iso-8859-1"
→/>
  <title>Plaatsnamen en postcodes</
→title>
  <style type="text/css" title="text/
→css" media="all">
    h2 {
      color: #960;
      font-family: Verdana, Geneva,
→Arial, Helvetica, sans-serif;
      font-size: 14pt;
      text-align: center;
    }
    td {
      color: #333;
      font-family: Verdana, Geneva,
→Arial, Helvetica, sans-serif;
      font-size: 10pt;
      text-align: center;
    }
    .fout {
      color: #f30;
    }
  </style>
</head>
<body>
<?php # Script 3.4 - horizontaal.php
```

Om het resultaat wat netter te maken, is opmaak gedefinieerd voor de CSS-klasse `fout` en de tags `<h2>` en `<td>`.

Script 3.4 (horizontaal.php) Dit PHP-script haalt alle plaatsnamen en postcodes op voor een gegeven staat. Ze worden niet weergegeven als een verticale lijst, maar in een tabel met vijf cellen per rij.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="nl" lang="nl">
4   <head>
5     <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
6     <title>Plaatsnamen en postcodes</title>
7     <style type="text/css" title="text/css" media="all">
8       h2 {
9         color: #960;
10        font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
11        font-size: 14pt;
12        text-align: center;
13      }
14      td {
15        color: #333;
16        font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
17        font-size: 10pt;
18        text-align: center;
```

Script 3.4 Vervolg

```
19      }
20      .fout {
21        color: #f30;
22      }
23    </style>
24  </head>
25  <body>
26  <?php # Script 3.4 - horizontaal.php
27
28  /**
29   * Deze pagina leest en toont alle plaatsnamen
30   * en postcodes in een bepaalde staat.
31   * De resultaten worden weergegeven in een tabel.
32   */
33
34  // Afkorting van weer te geven staat:
35  $staat = 'AK';
36
37  // Aantal per rij weer te geven items:
38  $items = 5;
39
40  // Een kop weergeven:
41  echo "<h2>Plaatsnamen en postcodes in $staat</h2>\n";
42
43  // Maak verbinding met de database:
44  $dbc = @mysqli_connect('localhost', 'username', 'password', 'zips') OR die ('<p
    class="fout">Databaseverbinding kon niet worden gemaakt.</body></html>');
45
46  // Haal de plaatsen en postcodes op, geordend op plaatsnaam:
47  $q = "SELECT city, zip_code FROM zip_codes WHERE state='$staat' ORDER BY city";
48  $r = mysqli_query($dbc, $q);
49
50  // Haal de resultaten op:
51  if (mysqli_num_rows($r) > 0) {
52
53      // Begin een tabel:
54        echo '<table border="2" width="90%" cellspacing="3" cellpadding="3"
    align="center">';
55
56      // Start een teller:
57      $i = 0;
58
59      // Haal alle records op:
60      while (list($plaatsnaam, $postcode) = mysqli_fetch_array($r, MYSQLI_NUM)) {
61
```

Script 3.4 Vervolg

```
62          // Moeten we een nieuwe rij beginnen?
63          if ($i == 0) {
64              echo "<tr>\n";
65          }
66
67          // Toon de record:
68          echo "\t<td>$plaatsnaam, $postcode</td>\n";
69
70          // Verhoog de teller:
71          $i++;
72
73          // Moeten we de rij beëindigen?
74          if ($i == $items) {
75              echo "</tr>\n";
76              $i = 0; // Teller opnieuw instellen.
77          }
78
79      } // Einde van de while-lus.
80
81      if ($i > 0) { // Laatste rij is gedeeltelijk leeg.
82
83          // Toon het benodigde aantal extra cellen:
84          for (;$i < $items; $i++) {
85              echo "<td> </td>\n";
86          }
87
88          // Maak de rij af:
89          echo '</tr>';
90
91      } // Einde van if ($i > 0).
92
93      // Sluit de tabel:
94      echo '</table>';
95
96  } else { // Verkeerde afkorting van een staat.
97
98      echo '<p class="fout">Er is een ongeldige afkorting van een staat opgegeven.</p>';
99
100 } // Einde van de eerste if.
101
102 // Sluit de databaseverbinding:
103 mysqli_close($dbc);
104
105 ?>
106 </body>
107 </html>
```

2. Stel de benodigde variabelen in en geef een kop weer.

```
$staat = 'AK';
$items = 5;
echo "<h2>Plaatsnamen en postcodes in
→$staat</h2>\n";
```

3. Maak verbinding met de database en voer de query uit.

```
$dbc = @mysqli_connect('localhost',
→'username', 'password', 'zips') OR die
→('<p class="fout">Databaseverbinding
→kon niet worden gemaakt.</body>
→</html>');
$q = "SELECT city, zip_code FROM
→zip_codes WHERE state='$staat' ORDER BY
→city";
$r = mysqli_query($dbc, $q);
if (mysqli_num_rows($r) > 0) {
```

   De query retourneert elke plaats en postcode in de staat in alfabetische volgorde van de plaatsnaam.

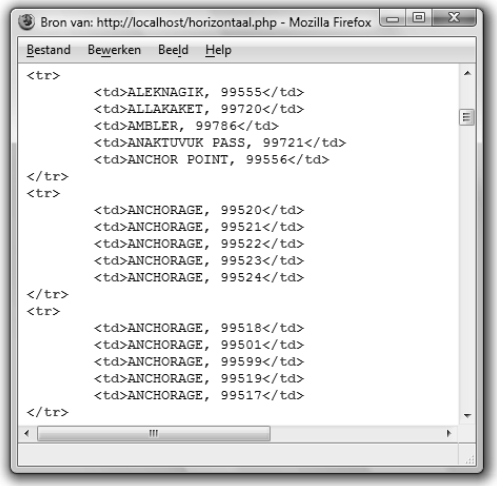
4. Begin een tabel en initialiseer een teller.

```
echo '<table border="2" width="90%" c
→ellspacing="3" cellpadding="3"
→align="center">
';
$i = 0;
```

   De teller `$i` volgt hoeveel items al in een rij zijn geplaatst.

5. Haal alle records op.

```
while (list($plaatsnaam, $postcode) =
→mysqli_fetch_array($r, MYSQLI_NUM)) {
```



Figuur 3.38 De XHTML-broncode voor een deel van de tabel.

6. Start een nieuwe rij, indien nodig.

```
if ($i == 0) {
    echo "<tr>\n";
}
```

Steeds wanneer binnen de lus `while` het eerste item in een rij moet worden geplaatst, moet een nieuwe rij worden gestart door de `<tr>` in te voegen. Dit geldt wanneer de lus voor het eerst wordt uitgevoerd (omdat `$i` aanvankelijk 0 is) en nadat `$i` opnieuw is ingesteld (nadat een rij is voltooid).

7. Geef de record weer en verhoog de teller.

```
echo "\t<td>$plaatsnaam, $postcode
→</td>\n";
$i++;
```

Om de XHTML-broncode van de pagina beter leesbaar te maken (**figuur 3.38**), verschijnt elk item op zijn eigen regel en ingesprongen met een tab.

8. Voltooi de rij, indien nodig.

```
if ($i == $items) {
    echo "</tr>\n";
    $i = 0;
}
```

Zodra de teller gelijk is aan het aantal in een rij te plaatsen items, moet die rij eindigen door `</tr>` in te voegen. Daarna moet de teller opnieuw worden ingesteld, zodat bij de volgende start van de lus ook een nieuwe rij word gestart.

9. Voltooi de lus met `while`.

```
} // Einde van de while-lus.
```

10. Voltooi de laatste rij, indien nodig.

```
if ($i > 0)
    for (;$i < $items; $i++) {
        echo "<td> </td>\n";
    }
    echo '</tr>';
} // Einde van if ($i > 0).
```

Deze stap wordt gemakkelijk vergeten. Tenzij het aantal weergegeven items gemakkelijk deelbaar is door het aantal items dat moet worden weergegeven per rij (dat wil zeggen, er is geen restant van die deling), is de laatste rij incompleet (**figuur 3.39**).

Als `$i` een andere waarde heeft dan 0, moeten extra cellen worden toegevoegd (als `$i` een waarde 0 heeft, is de laatste rij voltooid). Een lus met `for` kan hiervoor zorgen en begint met de actuele waarde van `$i` en stopt wanneer `$i` gelijk is aan `$items`. Een onbekende feit van de `for`-lus is dat elk van de drie delen optioneel is. Omdat er geen initiële expressie hoeft te worden geëvalueerd (zoals het instellen van `$i` op een waarde), begint de lus met `(;`.

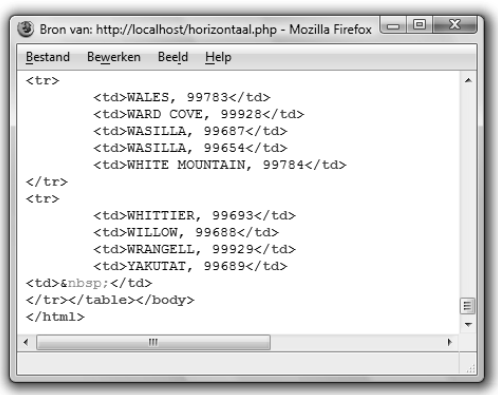
11. Sluit de tabel en voltooi de conditie die is gestart in stap 3.

```
    echo '</table>';
} else {
    echo '<p class="fout">Er is een
→ongeldige afkorting van een staat
→opgegeven.</p>';
} // Einde van de eerste if.
```

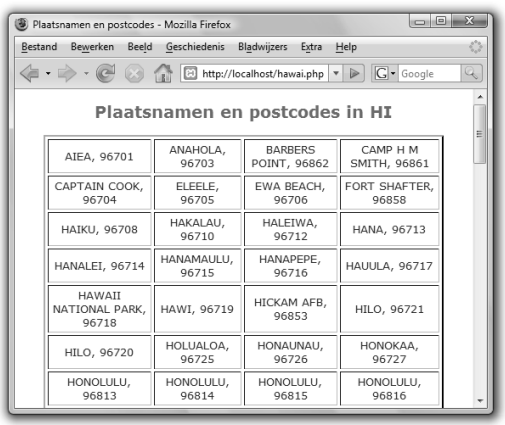
12. Sluit de databaseverbinding en voltooi de pagina.

```
mysqli_close($dbc);
?>
</body>
</html>
```

13. Sla het bestand op als `horizontaal.php`, plaats het in uw webdirectory en test het in uw webbrowser (figuur 3.37).
14. Verander de waarde van `$staat`, verander de waarde van `$items` en test opnieuw in een webbrowser (**figuur 3.40**).



Figuur 3.39 De laatste rij had vier items, dus moest een lege tabelcel worden gemaakt.



Figuur 3.40 Met twee snelle aanpassingen geeft het script nu alle plaatsnamen en postcodes voor een andere staat weer (hier Hawaï), vier per rij.