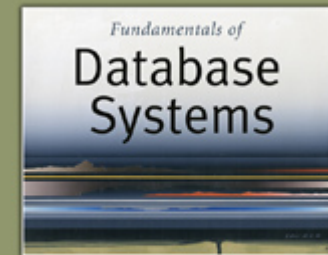


5th Edition

Elmasri / Navathe

Chapter 26

Web Database Programming using PHP



5th Edition

Elmasri / Navathe



Outline

- Overview
- Structured, semi-structured, and unstructured data
- PHP
- Example of PHP
- Basic features of PHP
- Overview of PHP Database programming

Overview

- Hypertext documents
 - Common method of specifying contents
 - Various languages
 - HTML (HyperText Markup Language)
 - Used for generating static web pages
 - XML (eXtensible Markup Language)
 - Standard for exchanging data over the web
 - PHP (PHP Hypertext Preprocessor {recursive acronym})
 - Dynamic web pages

Structured, semi-structured, and unstructured data

- **Structured data**
 - Information stored DB
 - Strict format
 - Limitation
 - Not all data collected is structured
- **Semi-structured data**
 - Data may have certain structure but not all information collected has identical structure
 - Some attributes may exist in some of the entities of a particular type but not in others
- **Unstructured data**
 - Very limited indication of data type
 - E.g., a simple text document

Semi-structured data

- Figure 26.1 represents semi-structured data as a graph
 - Note: difference between the two workers' data

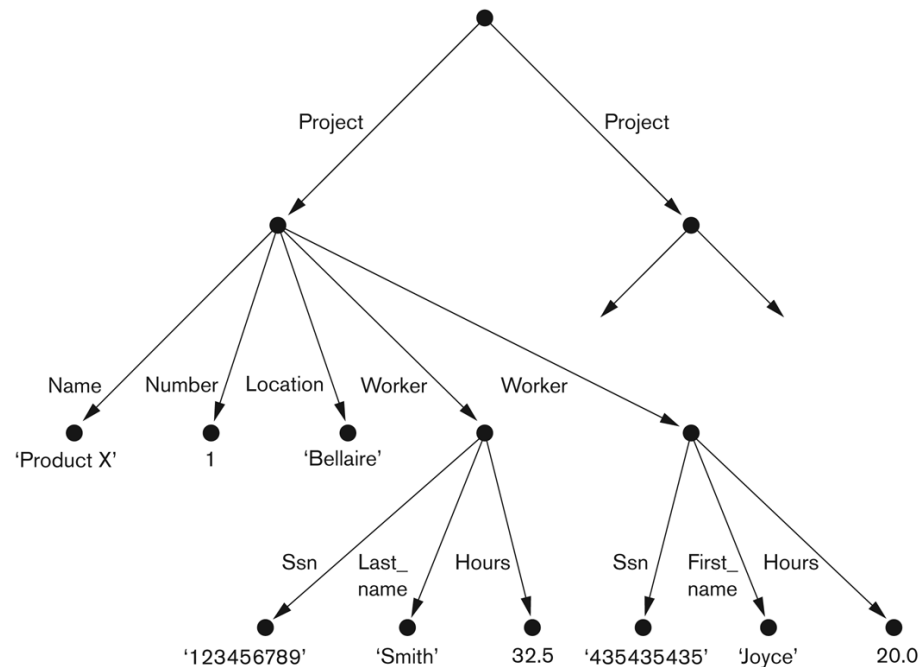


Figure 26.1
Representing semistructured data as a graph.

Semi-structured data (contd.)

- Key differences between semi-structured and structured data
 - Semi-structured data is mixed in with its schema
 - Sometimes known as self-describing data
 - Can be displayed as a graph (Figure 26.1)

Semi-structured data (contd.)

- Key differences between semi-structured and structured data
 - Schema information:
 - names of attributes, relationships, and classes in the semi-structured data as intermixed with their data values in the same data structure
 - Semi-structured data has no requirement for pre-defined schema to contain data

Unstructured data

- Limited indication of data types
 - E.g., web pages in html contain some unstructured data
 - Figure 26.2 shows part of HTML document representing unstructured data

```
<HTML>
<HEAD>
...
</HEAD>
<BODY>
  <H1>List of company projects and the employees in each project</H1>
  <H2>The ProductX project:</H2>
  <TABLE width="100%" border=0 cellpadding=0 cellspacing=0>
    <TR>
      <TD width="50%"><FONT size="2" face="Arial">John Smith:</FONT></TD>
      <TD>32.5 hours per week</TD>
    </TR>
    <TR>
      <TD width="50%"><FONT size="2" face="Arial">Joyce English:</FONT></TD>
      <TD>20.0 hours per week</TD>
    </TR>
  </TABLE>
  <H2>The ProductY project:</H2>
  <TABLE width="100%" border=0 cellpadding=0 cellspacing=0>
    <TR>
      <TD width="50%"><FONT size="2" face="Arial">John Smith:</FONT></TD>
      <TD>7.5 hours per week</TD>
    </TR>
    <TR>
      <TD width="50%"><FONT size="2" face="Arial">Joyce English:</FONT></TD>
      <TD>20.0 hours per week</TD>
    </TR>
    <TR>
      <TD width="50%"><FONT size="2" face="Arial">Franklin Wong:</FONT></TD>
      <TD>10.0 hours per week</TD>
    </TR>
  </TABLE>
  ...
</BODY>
</HTML>
```

Figure 26.2

Part of an HTML document representing unstructured data.

PHP

- Open source
- General purpose scripting language
- Interpreter engine in C
 - Can be used on nearly all computer types
- Particularly suited for manipulation of text pages
- Manipulates (dynamic html) at the Web server
 - Conversely, JavaScript is downloaded and executed on the client
- Has libraries of functions for accessing databases

A simple PHP Example

- Suppose the file containing program segment P1 is stored at www.myserver.com/example/greeting.php

(a)

```
//Program Segment P1:
0) <?php
1) // Printing a welcome message if the user submitted their name
   // through the HTML form
2) if ($_POST['user_name']) {
3)     print("Welcome,  ");
4)     print($_POST['user_name']);
5) }
6) else {
7)     // Printing the form to enter the user name since no name has
   // been entered yet
8)     print <<<_HTML_
9)     <FORM method="post" action="$_SERVER['PHP_SELF']">
10)    Enter your name: <input type="text" name="user_name">
11)    <BR/>
12)    <INPUT type="submit" value="SUBMIT NAME">
13)    </FORM>
14)    _HTML_;
15) }
16) ?>
```

A simple PHP Example

- When user types the url, the PHP interpreter will start interpreting produce form in 26.3 (b)

(a)

```
//Program Segment P1:  
0) <?php  
1) // Printing a welcome message if the user submitted their name  
   // through the HTML form  
2) if ($_POST['user_name']) {  
3)   print("Welcome, ");  
4)   print($_POST['user_name']);  
5) }  
6) else {  
7)   // Printing the form to enter the user name since no name has  
   // been entered yet  
8)   print <<<_HTML_  
9)   <FORM method="post" action="$_SERVER['PHP_SELF']">  
10)  Enter your name: <input type="text" name="user_name">  
11)  <BR/>  
12)  <INPUT type="submit" value="SUBMIT NAME">  
13)  </FORM>  
14)  _HTML_;  
15) }  
16) ?>
```

(b)



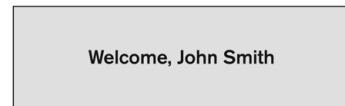
Enter your name:

(c)



Enter your name:

(d)



Welcome, John Smith

Figure 26.3

- (a) PHP program segment for entering a greeting,
(b) Initial form displayed by PHP program segment,
(c) User enters name *John Smith*, (d) Form prints
welcome message for *John Smith*.

Overview of basic features of PHP

- PHP variables, data types, and programming constructs
 - Variable names start with \$ and can include characters, letters, numbers, and _.
 - No other special characters are permitted
 - Are case sensitive
 - Can't start with a number
 - Variables are not types
 - Values assigned to variables determine their type
 - Assignments can change the type
 - Variable assignments are made by =

Overview of basic features of PHP

- PHP variables, data types, and programming constructs (contd.)
 - Main ways to express strings
 - Single-quoted strings (lines 0, 1, 2)
 - \' represents a quote in a string
 - Double-quoted strings (line 7)
 - Variable names can be interpolated
 - Here documents (line 8-11)
 - Enclose a part of a document between <<<DONMANE and end it with a single line containing the document name DONAME
 - Single and double quotes
 - The quotes should be straight quotes (') not (‘) or (’)

```
0) print 'Welcome to my Web site.';
1) print 'I said to him, "Welcome Home"';
2) print 'We\'ll now visit the next Web site';
3) printf('The cost is $%.2f and the tax is $%.2f', $cost, $tax) ;
4) print strtolower('AbCdE');
5) print ucwords(strtolower('JOHN smith'));
6) print 'abc' . 'efg'
7) print "send your email reply to: $email_address"
8) print <<<FORM_HTML
9) <FORM method="post" action="$_SERVER['PHP_SELF']">
10) Enter your name: <input type="text" name="user_name">
11) FORM_HTML
```

Figure 26.4
Illustrating basic PHP
string and text values.

Overview of basic features of PHP

- PHP variables, data types, and programming constructs (contd.)
 - String operations
 - (.) Is concatenate as in Line 6 of Figure 26.4
 - (strtolower()) converts string into lower case
 - Others as needed

```
0) print 'Welcome to my Web site.';
1) print 'I said to him, "Welcome Home"';
2) print 'We\'ll now visit the next Web site';
3) printf('The cost is $%.2f and the tax is $%.2f', $cost, $tax) ;
4) print strtolower('AbCdE');
5) print ucwords(strtolower('JOHN smith'));
6) print 'abc' . 'efg'
7) print "send your email reply to: $email_address"
8) print <<<FORM_HTML
9) <FORM method="post" action="$_SERVER['PHP_SELF']">
10) Enter your name: <input type="text" name="user_name">
11) _FORM_HTML
```

Figure 26.4
Illustrating basic PHP
string and text values.

Overview of basic features of PHP

- PHP variables, data types, and programming constructs (contd.)
 - **Numeric data types**
 - Follows C rules
 - See Line 3 of Figure 26.4

```
0) print 'Welcome to my Web site.';
1) print 'I said to him, "Welcome Home"';
2) print 'We\'ll now visit the next Web site';
3) printf('The cost is $%.2f and the tax is $%.2f', $cost, $tax) ;
4) print strtolower('AbCdE');
5) print ucwords(strtolower('JOHN smith'));
6) print 'abc' . 'efg'
7) print "send your email reply to: $email_address"
8) print <<<FORM_HTML
9) <FORM method="post" action="$_SERVER['PHP_SELF']">
10) Enter your name: <input type="text" name="user_name">
11) FORM_HTML
```

Figure 26.4
Illustrating basic PHP
string and text values.

Overview of basic features of PHP

- PHP variables, data types, and programming constructs (contd.)
 - Other programming constructs similar to C language constructs
 - for-loops
 - while-loops
 - if-statements

Overview of basic features of PHP

- PHP variables, data types, and programming constructs (contd.)
 - Boolean logic
 - True/false is equivalent to non-zero/zero
 - Comparison operators
 - ==, !=, >, >=, <, <=

Overview of basic features of PHP

- PHP Arrays
 - Allow a list of elements
 - Can be 1-dimensional or multi-dimensional
 - Can be **numeric** or **associative**
 - Numeric array is based on a numeric index
 - Associative array is based on a key => value relationship

Overview of basic features of PHP

- PHP Arrays
 - Line 0: \$teaching is a associative array
 - Line 1 shows how the array can be updated/accessed
 - Line 5: \$courses is a numeric array
 - No key is provided => numeric array

Figure 26.5

Illustrating basic PHP array processing.

```
0) $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                    'Graphics' => 'Kam');
1) $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
2) sort($teaching);
3) foreach ($teaching as $key => $value) {
4)     print " $key : $value\n";}
5) $courses = array('Database', 'OS', 'Graphics', 'Data Mining');
6) $alt_row_color = array('blue', 'yellow');
7) for ($i = 0, $num = count($courses); i < $num; $i++) {
8)     print '<TR bgcolor="' . $alt_row_color[$i % 2] . '">';
9)     print "<TD>Course $i is</TD><TD>$course[$i]</TD></TR>\n";
10) }
```

Overview of basic features of PHP

- PHP Arrays
 - There are several ways of looping through arrays
 - Line 3 and 4 show “**for each**” construct for looping through each and every element in the array
 - Line 7 and 10 show a traditional “**for loop**” construct for iterating through an array

Figure 26.5

Illustrating basic PHP array processing.

```
0) $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                    'Graphics' => 'Kam');
1) $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
2) sort($teaching);
3) foreach ($teaching as $key => $value) {
4)     print " $key : $value\n";}
5) $courses = array('Database', 'OS', 'Graphics', 'Data Mining');
6) $alt_row_color = array('blue', 'yellow');
7) for ($i = 0, $num = count($courses); i < $num; $i++) {
8)     print '<TR bgcolor="' . $alt_row_color[$i % 2] . '>';
9)     print "<TD>Course $i is</TD><TD>$course[$i]</TD></TR>\n";
10) }
```

Overview of basic features of PHP

- PHP Functions
 - Code segment P1' in Figure 26.6 has two functions
 - display_welcome()
 - display_empty_form()
 - Line 14-19 show how these functions can be called

```
//Program Segment P1':
0) function display_welcome() {
1)     print("Welcome,  ");
2)     print($_POST['user_name']);
3) }
4)
5) function display_empty_form(); {
6) print <<<_HTML_
7) <FORM method="post" action="$_SERVER['PHP_SELF']">
8) Enter your name: <INPUT type="text" name="user_name">
9) <BR/>
10) <INPUT type="submit" value="Submit name">
11) </FORM>
12) _HTML_;
13) }
14) if ($_POST['user_name']) {
15)     display_welcome();
16) }
17) else {
18)     display_empty_form();
19) }
```

Figure 26.6
Rewriting program
segment P1 as P1'
using functions.

Overview of basic features of PHP

- PHP Functions
 - Code segment in Figure 26.7 has function
 - `course_instructor($course, $teaching_assignments)`
 - with two parameters `$course`
 - holding the course name
 - and `$teaching_assignments`
 - holding the teacher associated with the course

```
0) function course_instructor ($course, $teaching_assignments) {
1)     if (array_key_exists($course, $teaching_assignments)) {
2)         $instructor = $teaching_assignments[$course];
3)         RETURN "$instructor is teaching $course";
4)     }
5)     else {
6)         RETURN "there is no $course course";
7)     }
8) }
9) $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                    'Graphics' => 'Kam');
10) $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
11) $x = course_instructor('Database', $teaching);
12) print($x);
13) $x = course_instructor('Computer Architecture', $teaching);
14) print($x);
```

Figure 26.7

Illustrating a function with arguments and return value.

Overview of basic features of PHP

■ PHP Functions

- Function call in Line 11 will return the string “Smith is teaching Database”

```
0) function course_instructor ($course, $teaching_assignments) {
1)     if (array_key_exists($course, $teaching_assignments)) {
2)         $instructor = $teaching_assignments[$course];
3)         RETURN "$instructor is teaching $course";
4)     }
5)     else {
6)         RETURN "there is no $course course";
7)     }
8) }
9) $teaching = array('Database' => 'Smith', 'OS' => 'Carrick',
                    'Graphics' => 'Kam');
10) $teaching['Graphics'] = 'Benson'; $teaching['Data Mining'] = 'Kam';
11) $x = course_instructor('Database', $teaching);
12) print($x);
13) $x = course_instructor('Computer Architecture', $teaching);
14) print($x);
```

Figure 26.7

Illustrating a function with arguments and return value.

Overview of basic features of PHP

- PHP Functions
 - Can also call OO functions (not discussed in this chapter)

Overview of basic features of PHP

■ PHP Observations

- Built-in PHP function `array_key_exists($k,$a)` returns true if the value in `$k` as a key in the associative array `$a`
- Function arguments are passed by value
- Return values are placed after the RETURN keyword
- Scope rules apply as with other programming languages

Overview of basic features of PHP

- PHP Server Variables and Forms
 - There are a number of built-in entries in PHP function. Some examples are:
 - `$_SERVER['SERVER_NAME']`
 - This provides the Website name of the server computer where PHP interpreter is running
 - `$_SERVER['REMOTE_ADDRESS']`
 - IP address of client user computer that is accessing the server
 - `$_SERVER['REMOTE_HOST']`
 - Website name of the client user computer

Overview of basic features of PHP

- PHP Server Variables and Forms
 - Examples contd.
 - `$_SERVER['PATH_INFO']`
 - The part of the URL address that comes after backslash (/) at the end of the URL
 - `$_SERVER['QUERY_STRING']`
 - The string that holds the parameters in the IRL after ?.
 - `$_SERVER['DOCUMENT_ROOT']`
 - The root directory that holds the files on the Web server

Overview of PHP Database Programming

- Connecting to the database
 - Must load PEAR DB library module DB.php
 - DB library functions are called using DB::`<function_name>`
 - The format for the connect string is:
 - `<DBMS>://<userid>:<password>@<DBserver>`
 - For example:
 - `$d=DB::connect('oci8://ac1:pass12@www.abc.com/db1')`

Overview of PHP Database Programming

- Figure 26.8 Example
 - Connecting to the database
 - Creating a table
 - Inserting a record

Figure 26.8

Connecting to a database, creating a table, and inserting a record.

```
0) require 'DB.php';
1) $d = DB::connect('oci8://acct1:pass12@www.host.com/db1');
2) if (DB::isError($d)) { die("cannot connect - " . $d->getMessage());}
   ...
3) $q = $d->query("CREATE TABLE EMPLOYEE
4)   (Emp_id INT,
5)   Name VARCHAR(15),
6)   Job VARCHAR(10),
7)   Dno INT) " );
8) if (DB::isError($q)) { die("table creation not successful - " .
   $q->getMessage()); }
   ...
9) $d->setErrorHandler(PEAR_ERROR_DIE);
   ...
10) $eid = $d->nextID('EMPLOYEE');
11) $q = $d->query("INSERT INTO EMPLOYEE VALUES
12)   ($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno'])" );
   ...
13) $eid = $d->nextID('EMPLOYEE');
14) $q = $d->query('INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?)',
15) array($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno']));
```

Overview of PHP Database Programming

- Examples of DB connections
 - MySQL: mysql
 - Oracle: oci8 (for versions 7, 8, 9)
 - SQLite: sqlite
 - MS SQL Server: mssql
 - Mini SQL: msql
 - Informix: ifx
 - Sybase: sybase
 - Any ODBC compliant DB: odbc
 - Others...

Overview of PHP Database Programming

- Figure 26.8 Example
 - Line 1 connects
 - Line 2 tests the connection

Figure 26.8

Connecting to a database, creating a table, and inserting a record.

```
0) require 'DB.php';
1) $d = DB::connect('oci8://acct1:pass12@www.host.com/db1');
2) if (DB::isError($d)) { die("cannot connect - " . $d->getMessage());}
   ...
3) $q = $d->query("CREATE TABLE EMPLOYEE
4)   (Emp_id INT,
5)   Name VARCHAR(15),
6)   Job VARCHAR(10),
7)   Dno INT)" );
8) if (DB::isError($q)) { die("table creation not successful - " .
   $q->getMessage()); }
   ...
9) $d->setErrorHandler(PEAR_ERROR_DIE);
   ...
10) $eid = $d->nextID('EMPLOYEE');
11) $q = $d->query("INSERT INTO EMPLOYEE VALUES
12)   ($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno'])" );
   ...
13) $eid = $d->nextID('EMPLOYEE');
14) $q = $d->query('INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?)',
15) array($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno']) );
```


Overview of PHP Database Programming

- Form data collection and record insertion
 - Figure 26.8 Line 10-12 shows how information collected via forms can be stored in the database

Figure 26.8

Connecting to a database, creating a table, and inserting a record.

```
0) require 'DB.php';
1) $d = DB::connect('oci8://acct1:pass12@www.host.com/db1');
2) if (DB::isError($d)) { die("cannot connect - " . $d->getMessage());}
   ...
3) $q = $d->query("CREATE TABLE EMPLOYEE
4)   (Emp_id INT,
5)   Name VARCHAR(15),
6)   Job VARCHAR(10),
7)   Dno INT)");
8) if (DB::isError($q)) { die("table creation not successful - " .
   $q->getMessage()); }
   ...
9) $d->setErrorHandler(PEAR_ERROR_DIE);
   ...
10) $eid = $d->nextID('EMPLOYEE');
11) $q = $d->query("INSERT INTO EMPLOYEE VALUES
12)   ($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno'])");
   ...
13) $eid = $d->nextID('EMPLOYEE');
14) $q = $d->query('INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?)');
15) array($eid, $_POST['emp_name'], $_POST['emp_job'], $_POST['emp_dno'] );
```

Overview of PHP Database Programming

- Retrieval queries and Database tables
 - Figure 26.9 Lines 4-7 retrieves name and department number of all employee records
 - Uses variable \$q to store query results
 - \$q->fetchrow retrieves the next row/record

```
0) require 'DB.php';
1) $d = DB::connect('oci8://acct1:pass12@www.host.com/dbname');
2) if (DB::isError($d)) { die("cannot connect - " . $d->getMessage()); }
3) $d->setErrorHandler(PEAR_ERROR_DIE);
...
4) $q = $d->query('SELECT Name, Dno FROM EMPLOYEE');
5) while ($r = $q->fetchRow()) {
6)     print "employee $r[0] works for department $r[1] \n" ;
7) }
...
8) $q = $d->query('SELECT Name FROM EMPLOYEE WHERE Job = ? AND Dno = ?',
9)     array($_POST['emp_job'], $_POST['emp_dno']) );
10) print "employees in dept $_POST['emp_dno'] whose job is
    $_POST['emp_job']: \n"
11) while ($r = $q->fetchRow()) {
12)     print "employee $r[0] \n" ;
13) }
...
14) $allresult = $d->getAll('SELECT Name, Job, Dno FROM EMPLOYEE');
15) foreach ($allresult as $r) {
16)     print "employee $r[0] has job $r[1] and works for department $r[2] \n" ;
17) }
...
```

Figure 26.9
Illustrating database retrieval queries.

Overview of PHP Database Programming

- Retrieval queries and Database tables
 - Figure 26.9 Lines 8-13 is a dynamic query (conditions based on user selection)
 - Retrieves names of employees who have specified job and work in a particular department
 - Values for these are entered through forms

```
0) require 'DB.php';
1) $d = DB::connect('oci8://acct1:pass12@www.host.com/dbname');
2) if (DB::isError($d)) { die("cannot connect - " . $d->getMessage()); }
3) $d->setErrorHandler(PEAR_ERROR_DIE);
   ...
4) $q = $d->query('SELECT Name, Dno FROM EMPLOYEE');
5) while ($r = $q->fetchRow()) {
6)     print "employee $r[0] works for department $r[1] \n" ;
7) }
   ...
8) $q = $d->query('SELECT Name FROM EMPLOYEE WHERE Job = ? AND Dno = ?',
9)     array($_POST['emp_job'], $_POST['emp_dno']) );
10) print "employees in dept $_POST['emp_dno'] whose job is
     $_POST['emp_job']: \n"
11) while ($r = $q->fetchRow()) {
12)     print "employee $r[0] \n" ;
13) }
   ...
14) $allresult = $d->getAll('SELECT Name, Job, Dno FROM EMPLOYEE');
15) foreach ($allresult as $r) {
16)     print "employee $r[0] has job $r[1] and works for department $r[2] \n" ;
17) }
   ...
```

Figure 26.9
Illustrating database retrieval queries.

Overview of PHP Database Programming

- Retrieval queries and Database tables
 - Figure 26.9 Lines 14-17 is an alternative way of specifying a query and looping over its records
 - Function `$d=>getAll` holds all the records in `$allresult`
 - For loop iterates over each row

```
0) require 'DB.php';
1) $d = DB::connect('oci8://acctl:pass12@www.host.com/dbname');
2) if (DB::isError($d)) { die("cannot connect - " . $d->getMessage()); }
3) $d->setErrorHandling(PEAR_ERROR_DIE);
   ...
4) $q = $d->query('SELECT Name, Dno FROM EMPLOYEE');
5) while ($r = $q->fetchRow()) {
6)     print "employee $r[0] works for department $r[1] \n" ;
7) }
   ...
8) $q = $d->query('SELECT Name FROM EMPLOYEE WHERE Job = ? AND Dno = ?',
9)     array($_POST['emp_job'], $_POST['emp_dno']) );
10) print "employees in dept $_POST['emp_dno'] whose job is
     $_POST['emp_job']: \n"
11) while ($r = $q->fetchRow()) {
12)     print "employee $r[0] \n" ;
13) }
   ...
14) $allresult = $d->getAll('SELECT Name, Job, Dno FROM EMPLOYEE');
15) foreach ($allresult as $r) {
16)     print "employee $r[0] has job $r[1] and works for department $r[2] \n" ;
17) }
   ...
```

Figure 26.9
Illustrating database retrieval queries.

Summary

- Structured, semi-structured, and unstructured data
- Example of PHP
- Basic features of PHP
- Overview of PHP Database programming