Working with Databases

Chapter 14

Randy Connolly and Ricardo Hoar

Fundamentals of Web Development

Chapter 14

Chapter 14 cont.

9 Summary

Chapter 14

Databases and Web Development

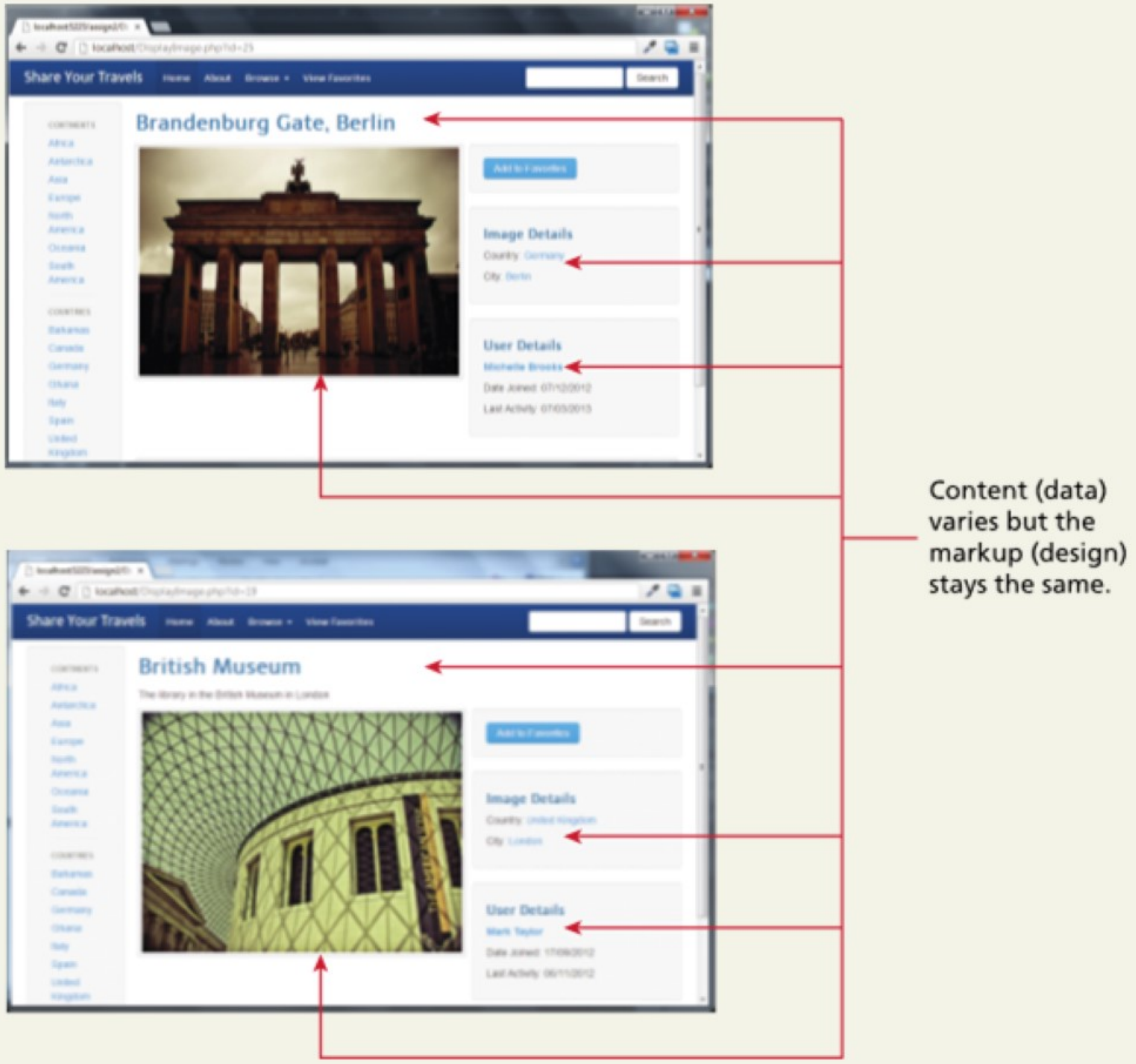
The Role of Databases in Web Development

Databases provide a way to implement one of the most important software design principlesnamely, that:

one should separate that which varies from that which stays the same .

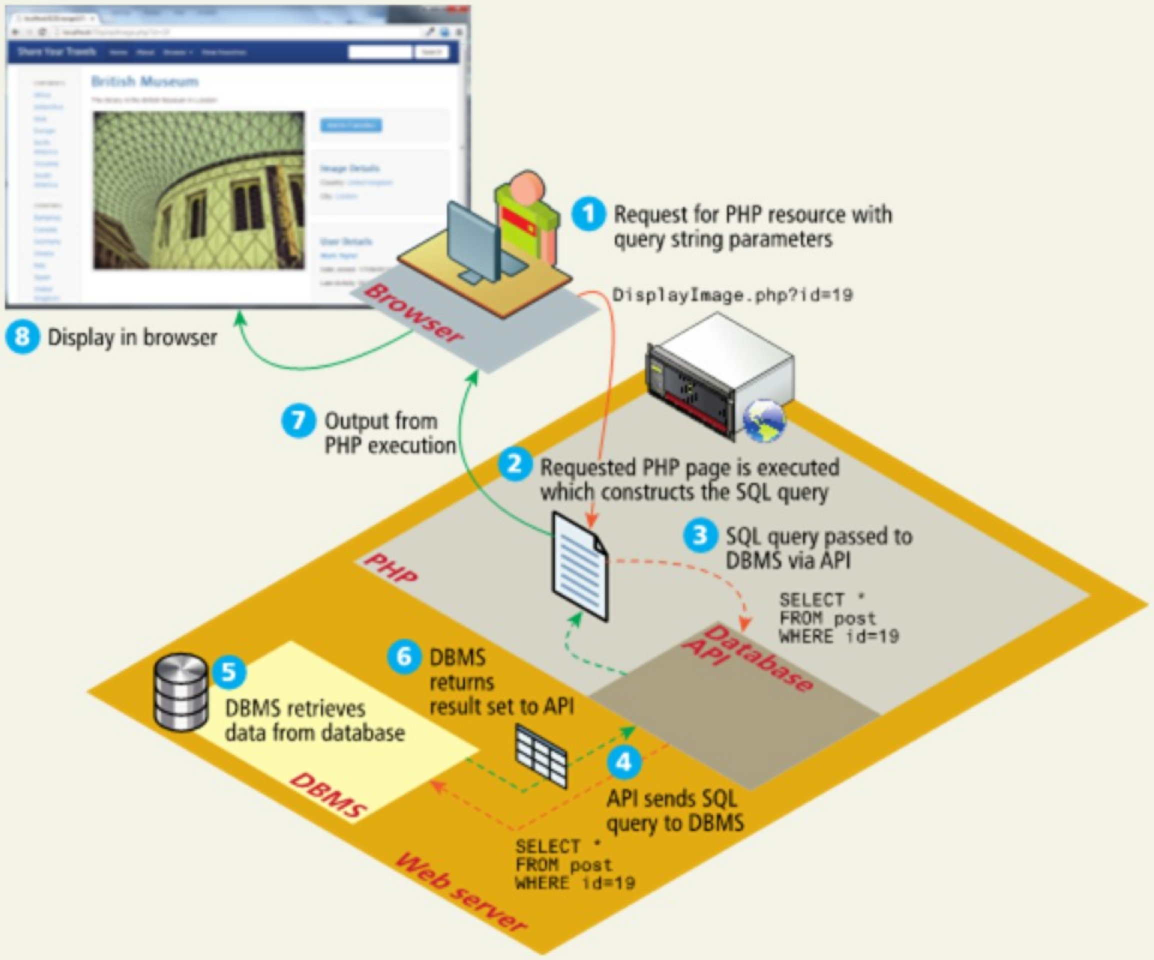
Databases and Web Development

The Role of Databases in Web Development



Databases and Web Development

How websites use databases

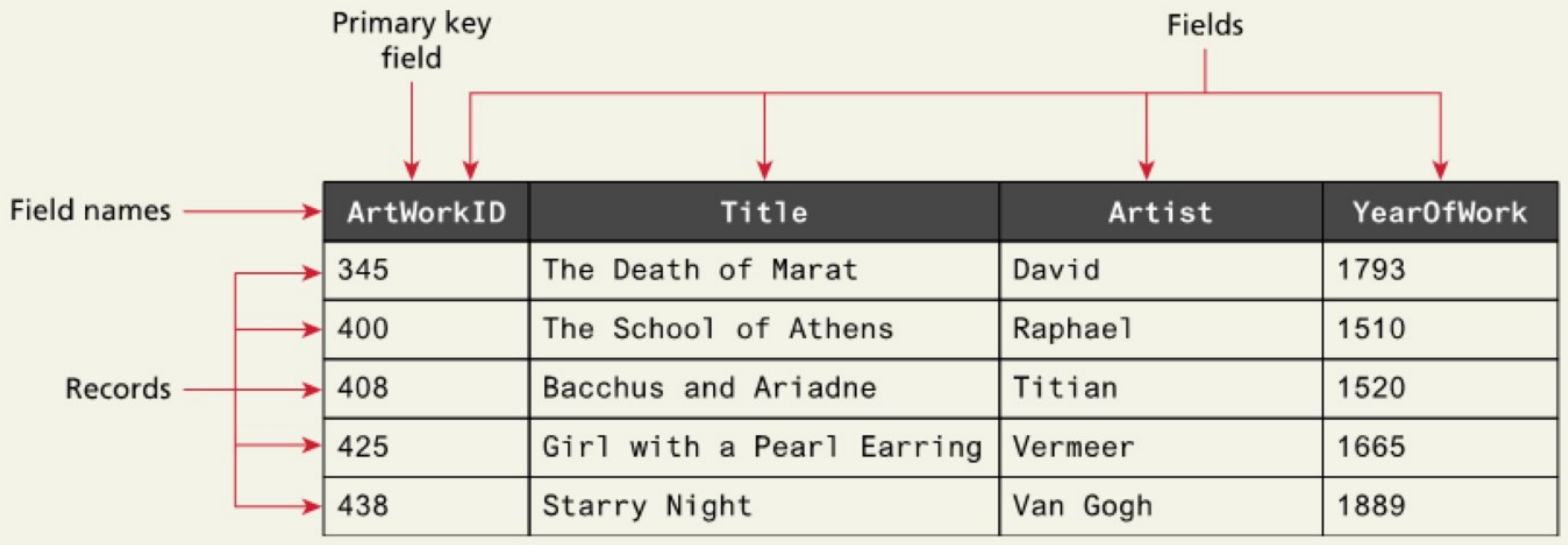


1 Request for PHP resource with query string parameters
DisplayImage.php?id=19
2 Requested PHP page is executed which constructs the SQL query
3 SQL query passed to DBMS via API
SELECT * FROM post WHERE id=19
4 API sends SQL query to DBMS
SELECT * FROM post WHERE id=19
5 DBMS retrieves data from database
6 DBMS returns result set to API
7 Output from PHP execution
8 Display in browser
Browser
PHP
Database API
DBMS
Web server
British Museum

Databases and Web Development

Database Design

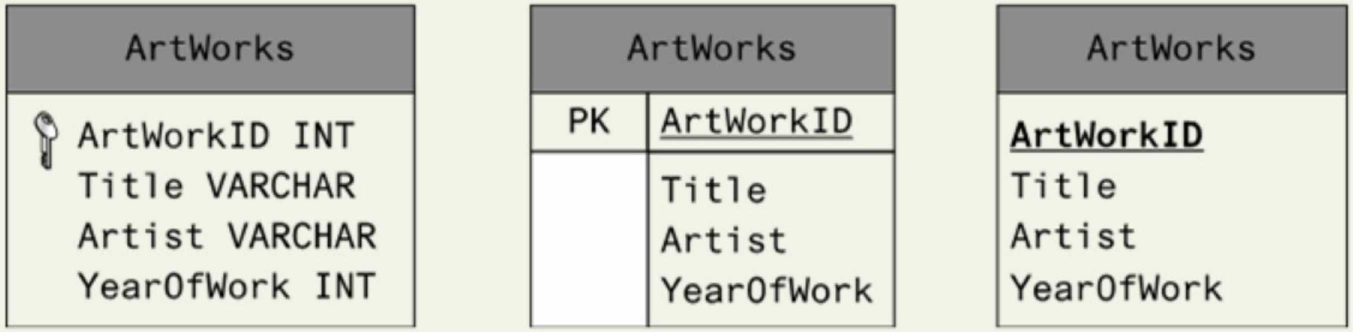
Normally taught in an entire course. This is a refresher.



ArtWorkID	Title	Artist	YearOfWork
345	The Death of Marat	David	1793
400	The School of Athens	Raphael	1510
408	Bacchus and Ariadne	Titian	1520
425	Girl with a Pearl Earring	Vermeer	1665
438	Starry Night	Van Gogh	1889

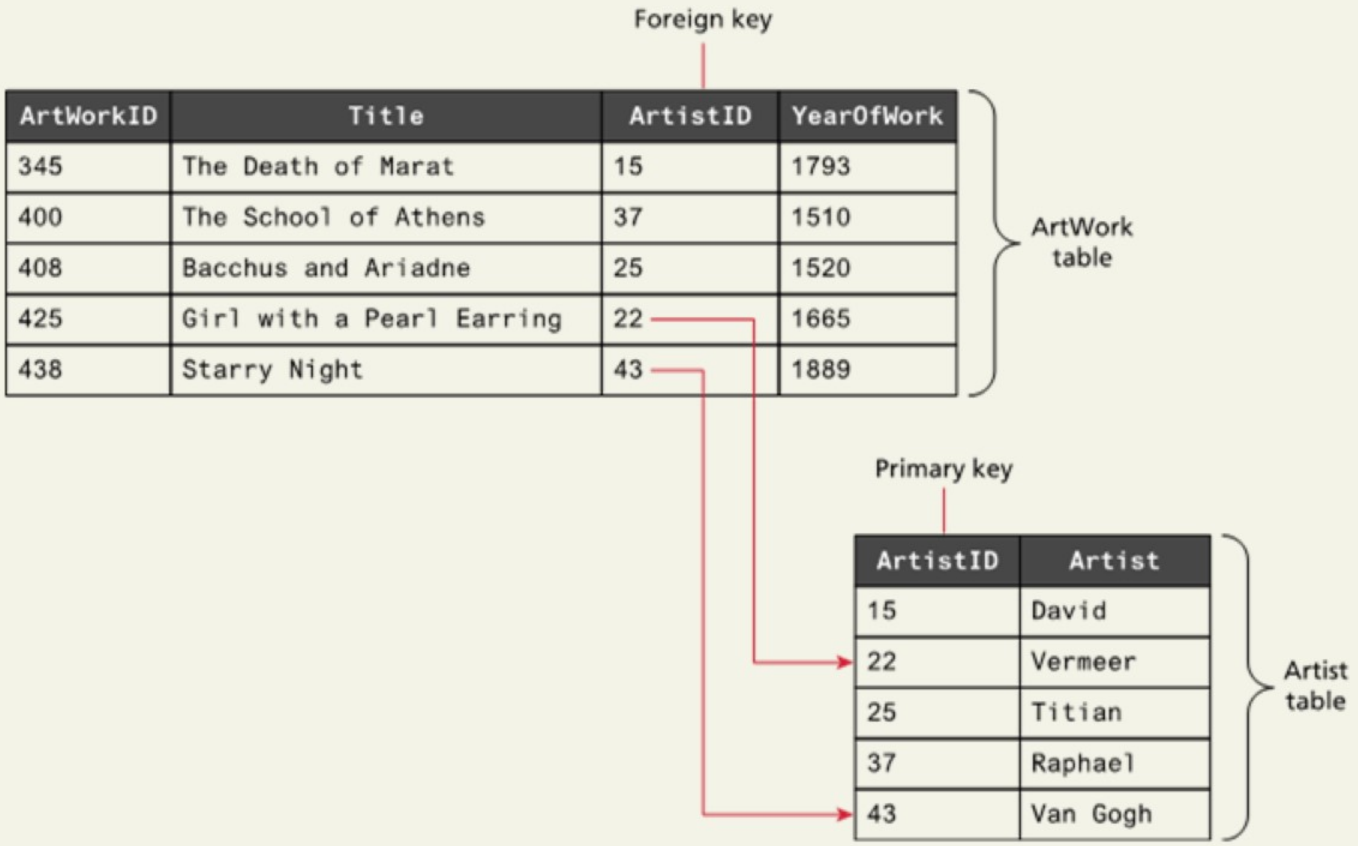
Databases and Web Development

Diagramming a table



Databases and Web Development

Foreign keys lining tables

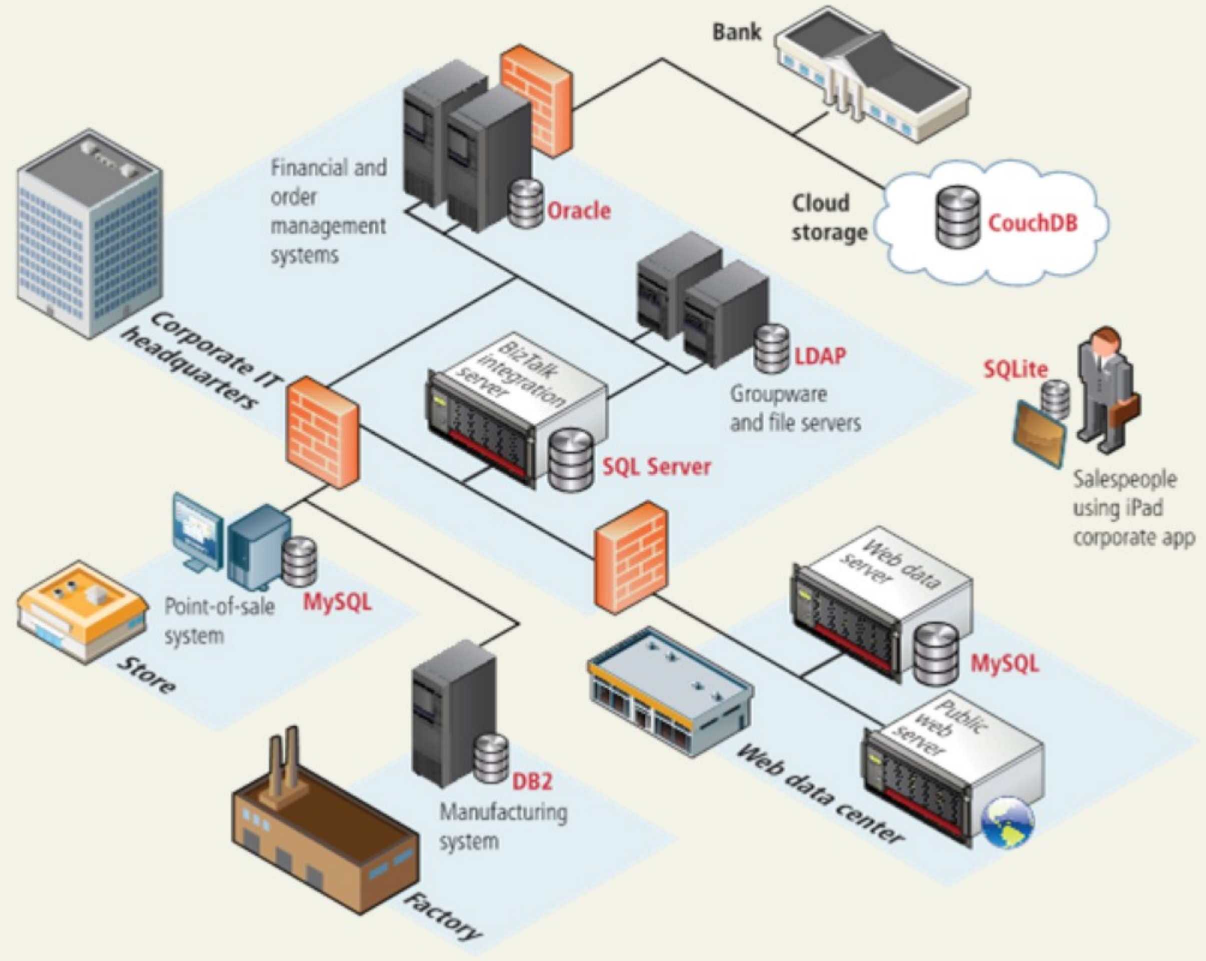


ArtWorkID	Title	ArtistID	YearOfWork
345	The Death of Marat	15	1793
400	The School of Athens	37	1510
408	Bacchus and Ariadne	25	1520
425	Girl with a Pearl Earring	22	1665
438	Starry Night	43	1889

ArtistID	Artist
15	David
22	Vermeer
25	Titian
37	Raphael
43	Van Gogh

Databases and Web Development

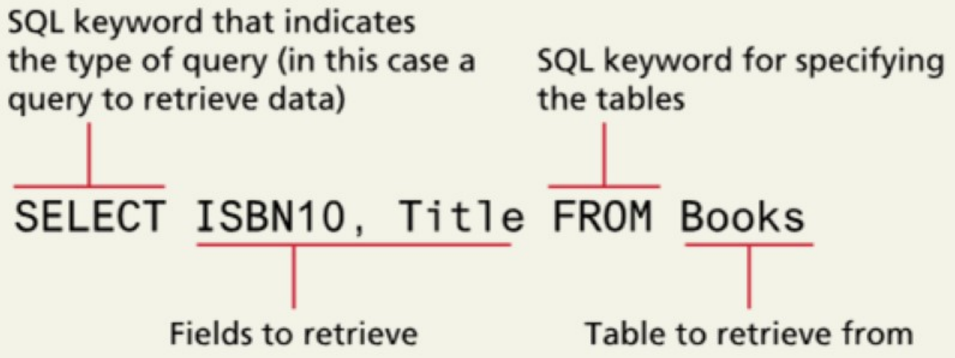
Database Options



Chapter 14

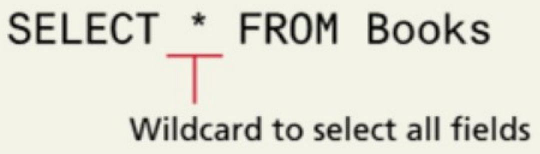
SQL

SELECT Statement



```
SELECT ISBN10, Title FROM Books
```

SQL keyword that indicates the type of query (in this case a query to retrieve data)
SQL keyword for specifying the tables
Fields to retrieve
Table to retrieve from

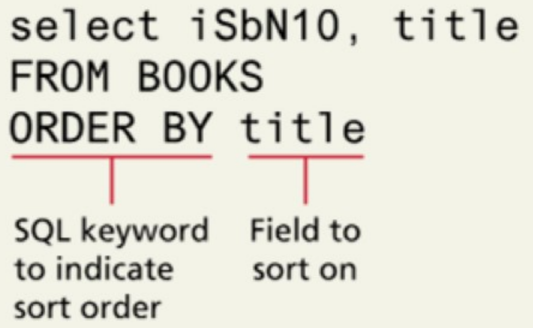


```
SELECT * FROM Books
```

Note: While the wildcard is convenient, especially when testing, for production code it is usually avoided; instead of selecting every field, you should select just the fields you need.

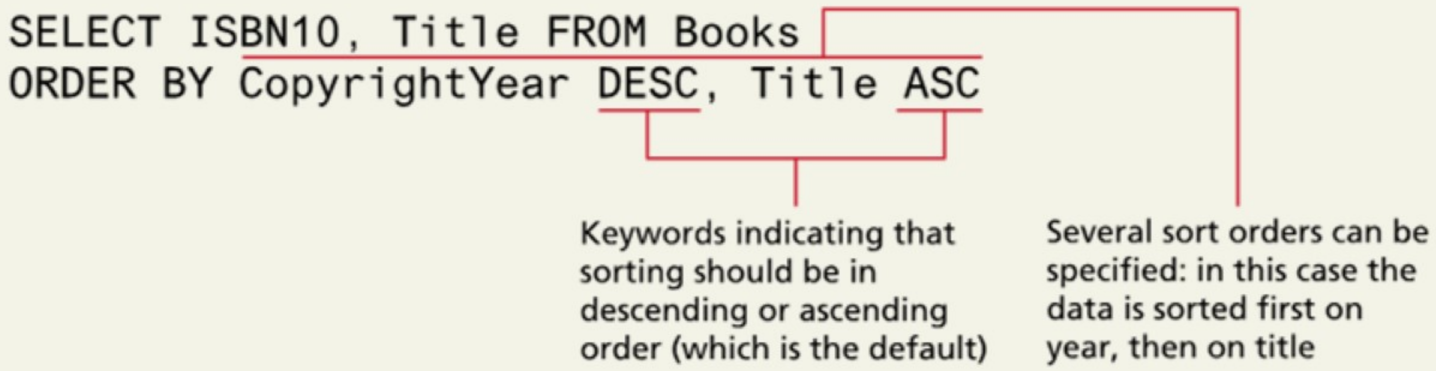
SQL

SELECT Statement



```
select iSbN10, title
FROM BOOKS
ORDER BY title
```

Note: SQL doesn't care if a command is on a single line or multiple lines, nor does it care about the case of keywords or table and field names. Line breaks and keyword capitalization are often used to aid in readability.



```
SELECT ISBN10, Title FROM Books
ORDER BY CopyrightYear DESC, Title ASC
```

Keywords indicating that sorting should be in descending or ascending order (which is the default)
Several sort orders can be specified: in this case the data is sorted first on year, then on title

SQL

Use the WHERE clause

```
SELECT isbn10, title FROM books
WHERE copyrightYear > 2010
```

SQL keyword that indicates to return only those records whose data matches the criteria expression

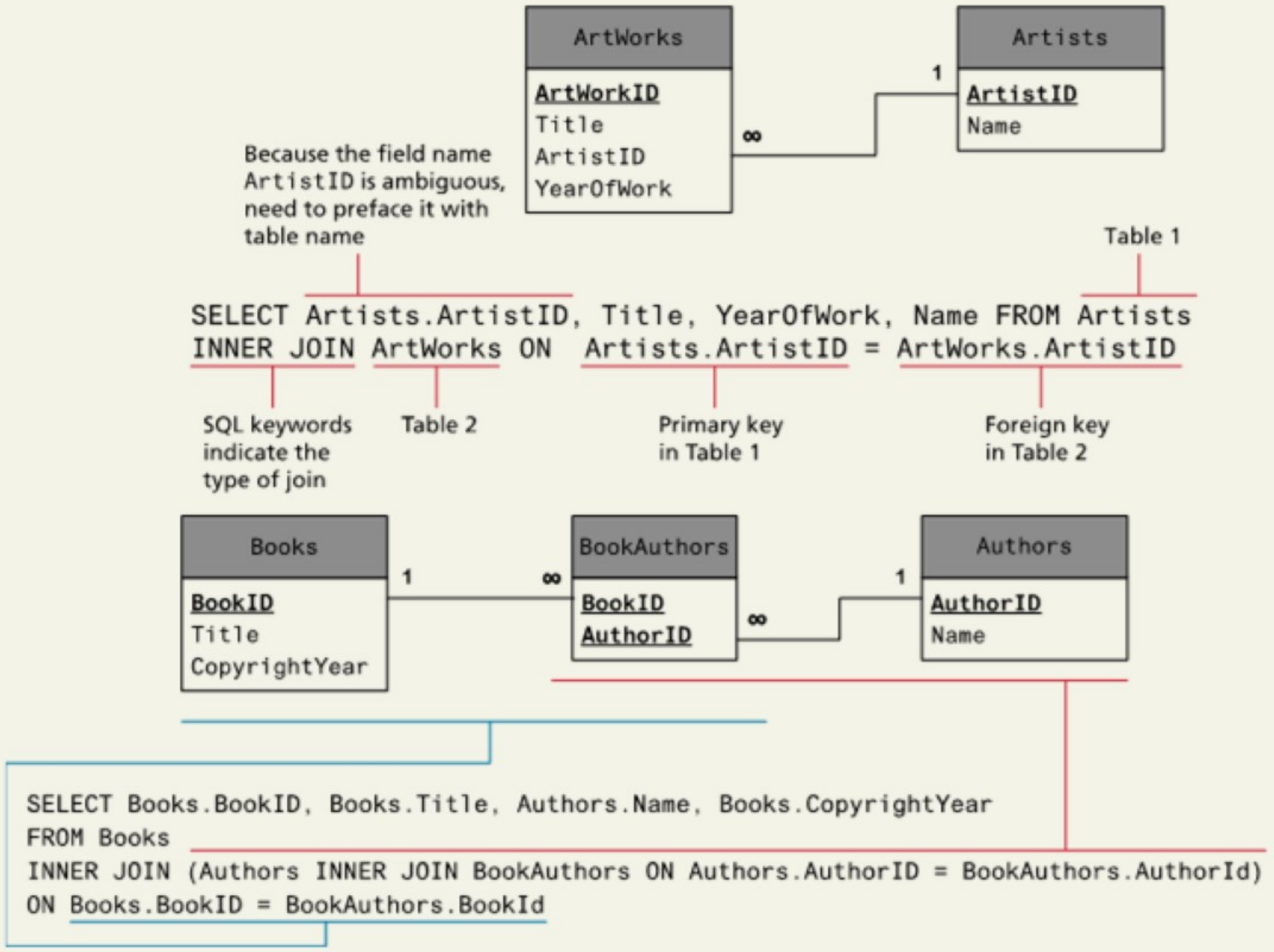
Expressions take form: field *operator* value

```
SELECT isbn10, title FROM books
WHERE category = 'Math' AND copyrightYear = 2014
```

Comparisons with strings require string literals (single or double quote)

SQL

Join together

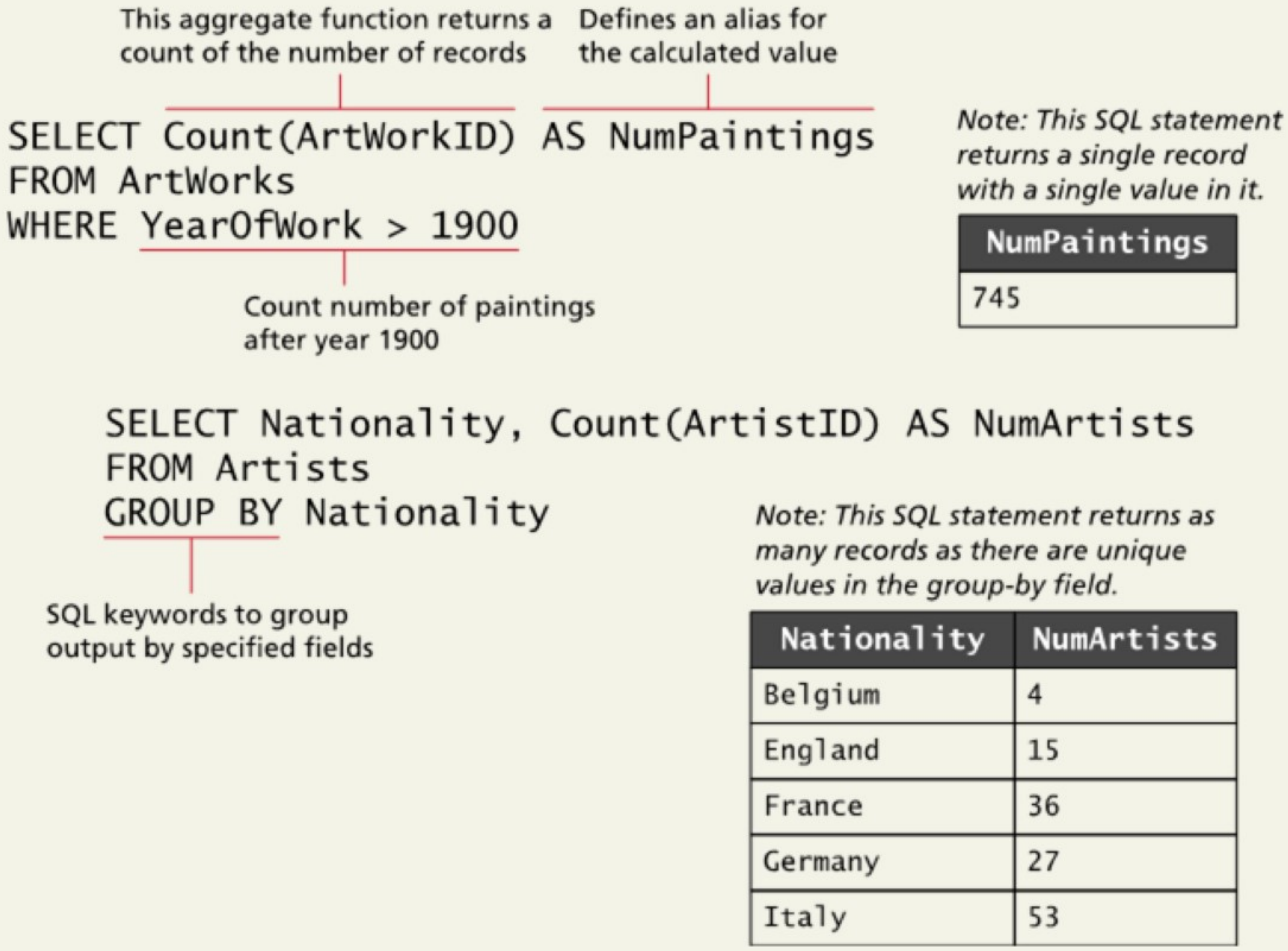


```
SELECT Artists.ArtistID, Title, YearOfWork, Name FROM Artists
INNER JOIN ArtWorks ON  Artists.ArtistID = ArtWorks.ArtistID
```

```
SELECT Books.BookID, Books.Title, Authors.Name, Books.CopyrightYear
FROM Books
INNER JOIN (Authors INNER JOIN BookAuthors ON Authors.AuthorID = BookAuthors.AuthorId)
ON Books.BookID = BookAuthors.BookId
```

SQL

Member group by



This aggregate function returns a count of the number of records

Defines an alias for the calculated value

```
SELECT Count(ArtWorkID) AS NumPaintings
FROM ArtWorks
WHERE YearOfWork > 1900
```

Count number of paintings after year 1900

Note: This SQL statement returns a single record with a single value in it.

NumPaintings
745

```
SELECT Nationality, Count(ArtistID) AS NumArtists
FROM Artists
GROUP BY Nationality
```

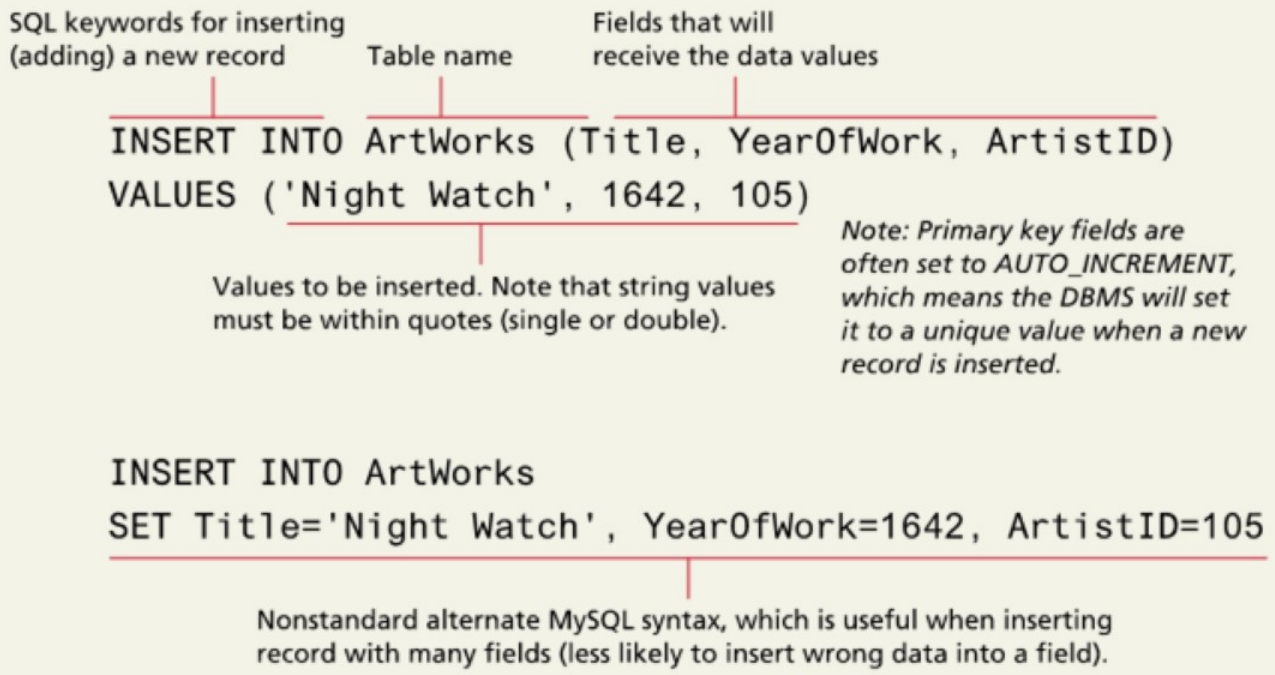
SQL keywords to group output by specified fields

Note: This SQL statement returns as many records as there are unique values in the group-by field.

Nationality	NumArtists
Belgium	4
England	15
France	36
Germany	27
Italy	53

SQL

INSERT, UPDATE, and DELETE Statements



SQL keywords for inserting (adding) a new record

Table name

Fields that will receive the data values

```
INSERT INTO ArtWorks (Title, YearOfWork, ArtistID)
VALUES ('Night Watch', 1642, 105)
```

Values to be inserted. Note that string values must be within quotes (single or double).

Note: Primary key fields are often set to AUTO_INCREMENT, which means the DBMS will set it to a unique value when a new record is inserted.

```
INSERT INTO ArtWorks
SET Title='Night Watch', YearOfWork=1642, ArtistID=105
```

Nonstandard alternate MySQL syntax, which is useful when inserting record with many fields (less likely to insert wrong data into a field).

SQL

INSERT, UPDATE, and DELETE Statements

```
UPDATE ArtWorks
SET Title='Night Watch', YearOfWork=1642, ArtistID=105
WHERE ArtWorkID=54
```

It is essential to specify which record to update, otherwise it will update all the records!

Specify the values for each updated field. *Note: Primary key fields that are AUTO_INCREMENT cannot have their values updated.*

SQL

INSERT, UPDATE, and DELETE Statements

```
DELETE FROM ArtWorks
WHERE ArtWorkID=54
```

It is essential to specify which record to delete, otherwise it will delete all the records!

SQL

Transactions

By starting the transaction, all database modifications within the transaction will only be permanently saved in the database if they all work

START TRANSACTION

INSERT INTO orders . . .

INSERT INTO orderDetails . . .

UPDATE inventory . . .

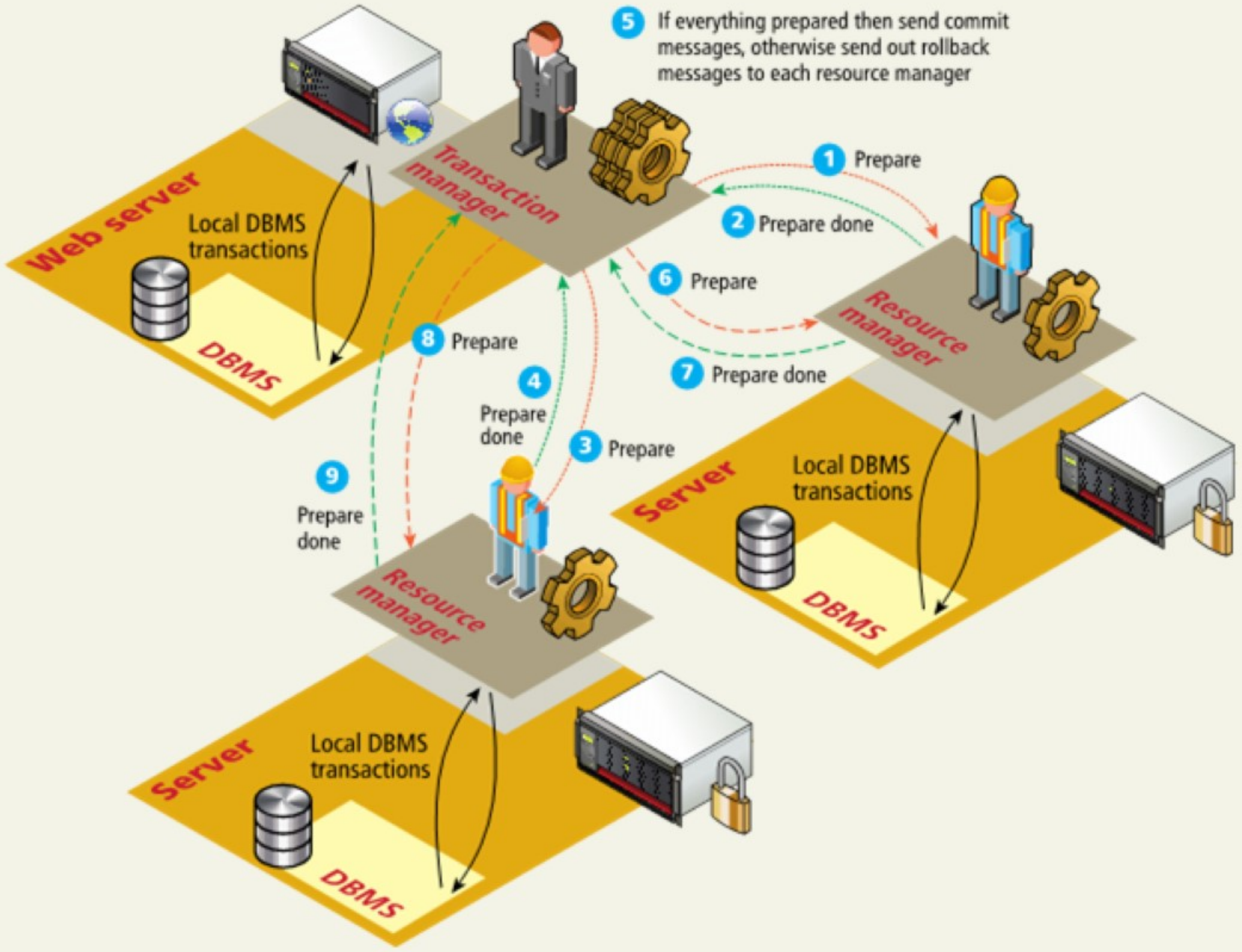
/* if we have made it here everything has worked so commit changes */

COMMIT

/* if we replace **COMMIT** with **ROLLBACK** then the three database changes would be "undone" */

SQL

Distributed Transactions



SQL

Data Definition Statements

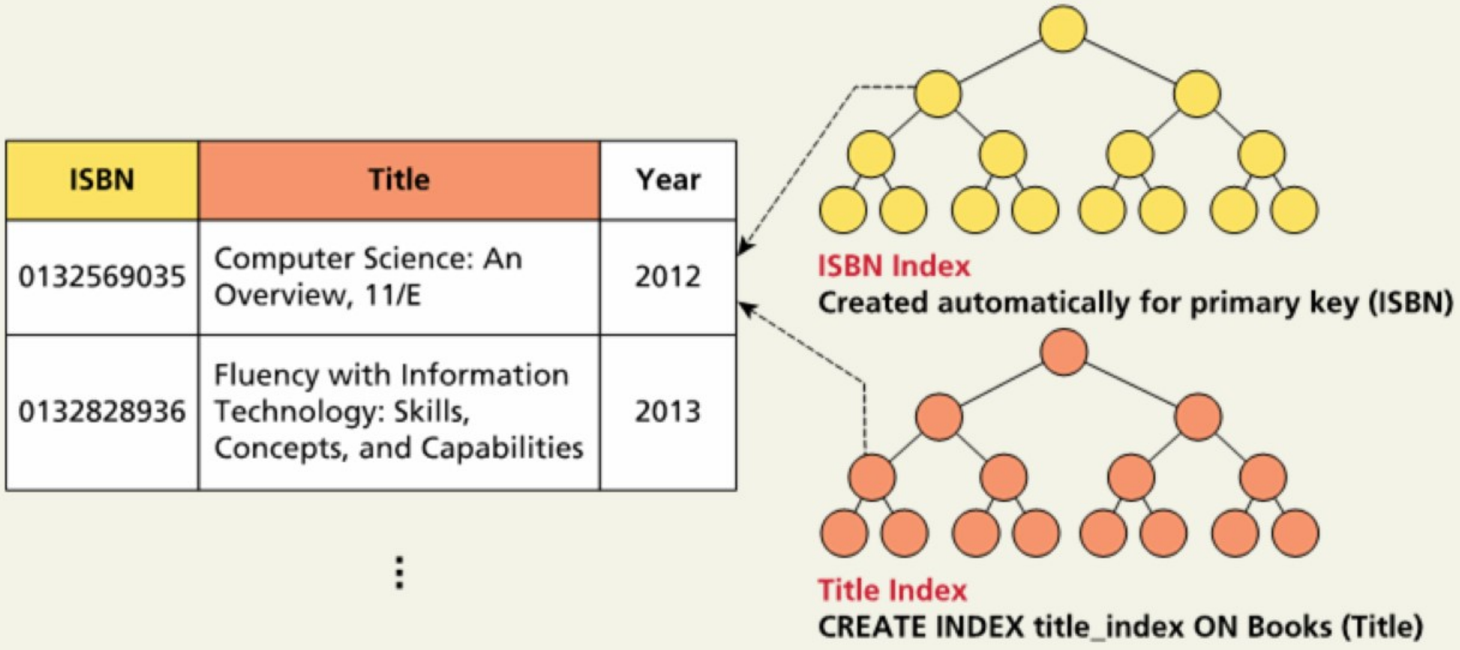
All of the SQL examples that you will use in this book are examples of **the Data Manipulation Language** features of SQL, that is, SELECT , UPDATE , INSERT , and DELETE .

There is also a **Data Definition Language** (DDL) in SQL, which is used for creating tables, modifying the structure of a table, deleting tables, and creating and deleting databases

While the book's examples do not use these database administration statements within PHP, your instructor may, and you may find yourself using them indirectly within something like the phpMyAdmin management tool anyhow.

SQL

Database Indexes and Efficiency



ISBN	Title	Year
0132569035	Computer Science: An Overview, 11/E	2012
0132828936	Fluency with Information Technology: Skills, Concepts, and Capabilities	2013

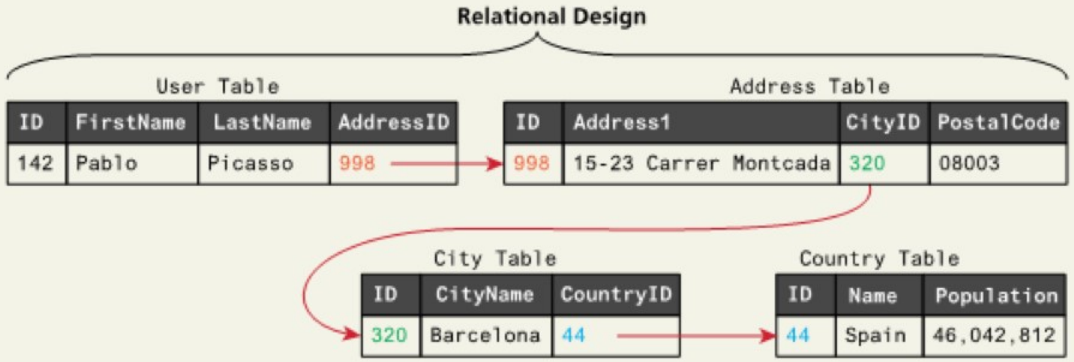
⋮

ISBN Index
Created automatically for primary key (ISBN)
Title Index
CREATE INDEX title_index ON Books (Title)

Chapter 14

NoSQL

A different way of thinking



User Table

ID	FirstName	LastName	AddressID
142	Pablo	Picasso	998

Address Table

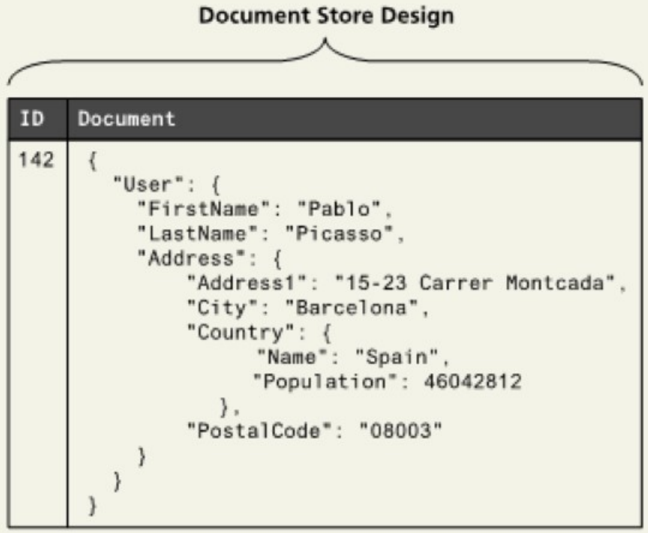
ID	Address1	CityID	PostalCode
998	15-23 Carrer Montcada	320	08003

City Table

ID	CityName	CountryID
320	Barcelona	44

Country Table

ID	Name	Population
44	Spain	46,042,812


Document Store Design

ID	Document
142	(see below)

```
{
  "User": {
    "FirstName": "Pablo",
    "LastName": "Picasso",
    "Address": {
        "Address1": "15-23 Carrer Montcada",
        "City": "Barcelona",
        "Country": {
              "Name": "Spain",
             "Population": 46042812
          },
        "PostalCode": "08003"
    }
  }
}
```

NoSQL

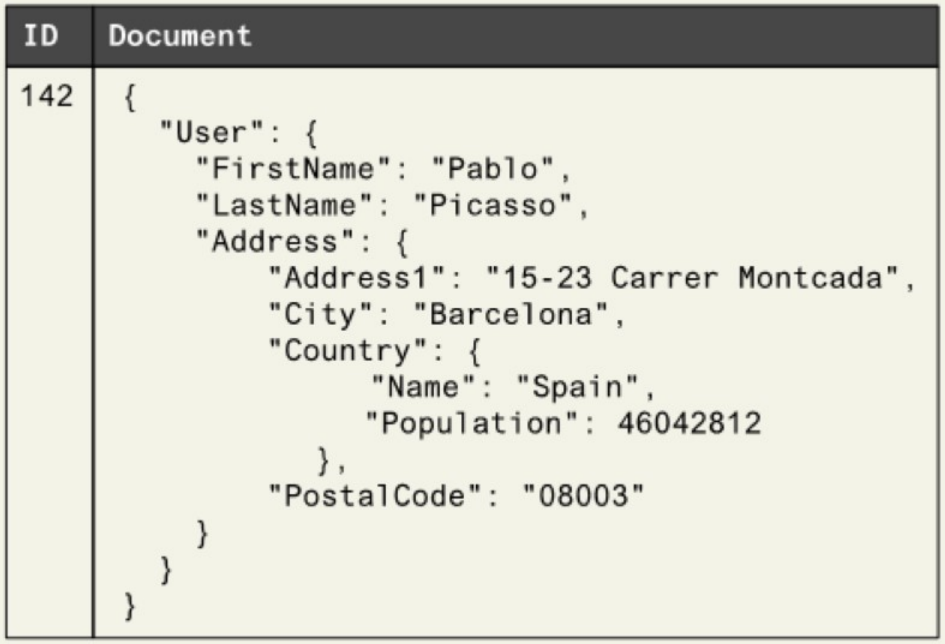
Key-Value Stores

- **Key-value stores** alone are very simplistic in that each record consists of one key and one value (i.e., is, they are analogous to PHP arrays).
- fast retrieval through means such as a hash function
- No need for indexes

NoSQL

Document Stores

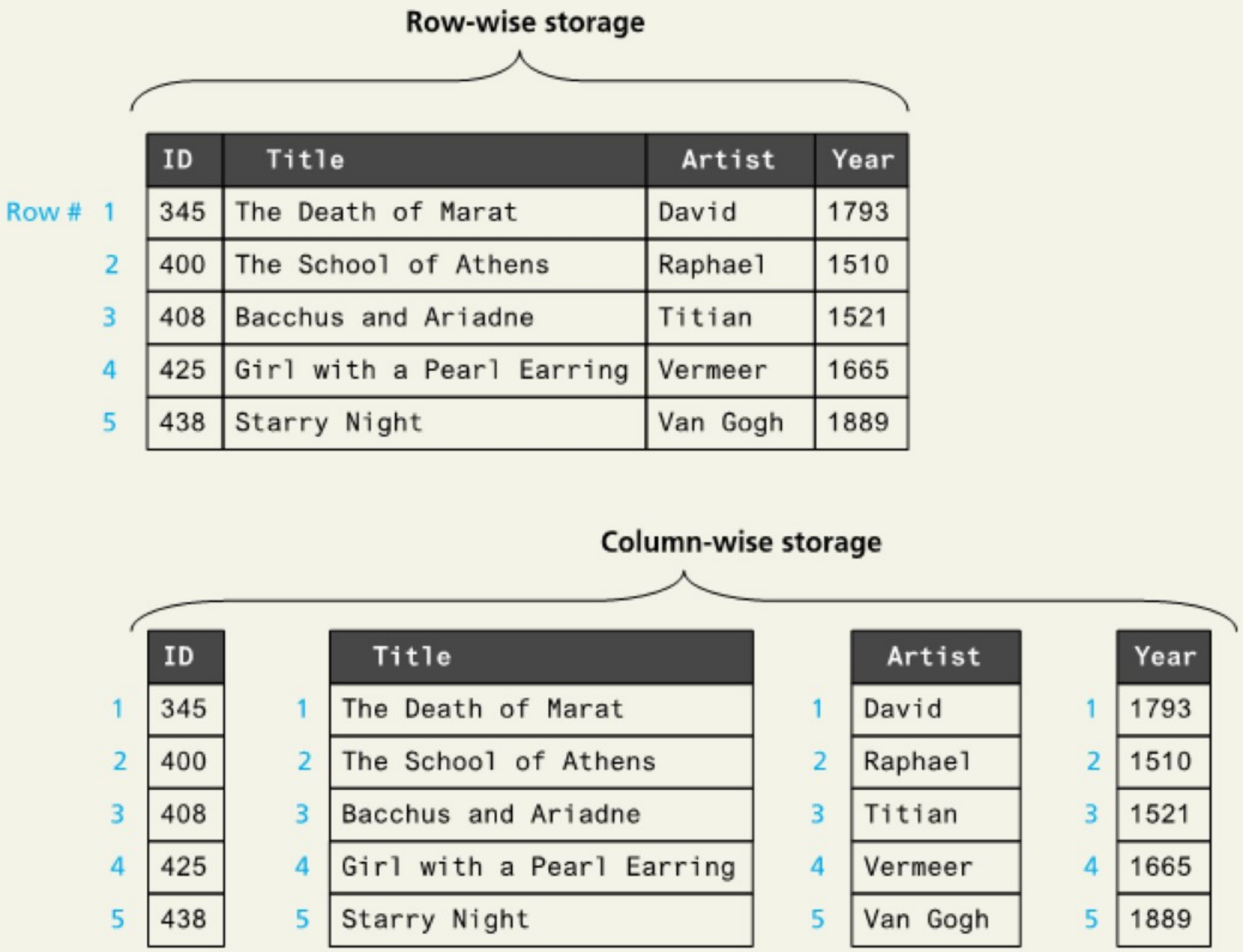
Document Stores associate keys with values, but unlike key-value stores, they call that value a **document**.



ID	Document
142	`{ "User": { "FirstName": "Pablo", "LastName": "Picasso", "Address": { "Address1": "15-23 Carrer Montcada", "City": "Barcelona", "Country": { "Name": "Spain", "Population": 46042812 }, "PostalCode": "08003" } } }`

NoSQL

Column Stores



Row #	ID	Title	Artist	Year
1	345	The Death of Marat	David	1793
2	400	The School of Athens	Raphael	1510
3	408	Bacchus and Ariadne	Titian	1521
4	425	Girl with a Pearl Earring	Vermeer	1665
5	438	Starry Night	Van Gogh	1889

	ID
1	345
2	400
3	408
4	425
5	438

	Title
1	The Death of Marat
2	The School of Athens
3	Bacchus and Ariadne
4	Girl with a Pearl Earring
5	Starry Night

	Artist
1	David
2	Raphael
3	Titian
4	Vermeer
5	Van Gogh

	Year
1	1793
2	1510
3	1521
4	1665
5	1889

Chapter 14

1 Databases and Web Development

2 SQL

3 NoSQL

4 Database APIs

5 Managing a MySQL Database

6 Accessing MySQL in PHP

7 Case Study Schemas

8 Sample Database Techniques

Database APIs

PHP MySQL APIs

- **MySQL extension**. This was the original extension to PHP for working with MySQL and has been replaced with the newer mysqli extension.
- **mysqli extension**. This extension provides both a procedural and an object-oriented approach. This extension also supports most of the latest features of MySQL.
- **PHP data objects (PDOs)**. provides an abstraction layer that with the appropriate drivers can be used with any database, and not just MySQL databases. However, it is not able to make use of all the latest features of MySQL.

Database APIs

Deciding on a Database API

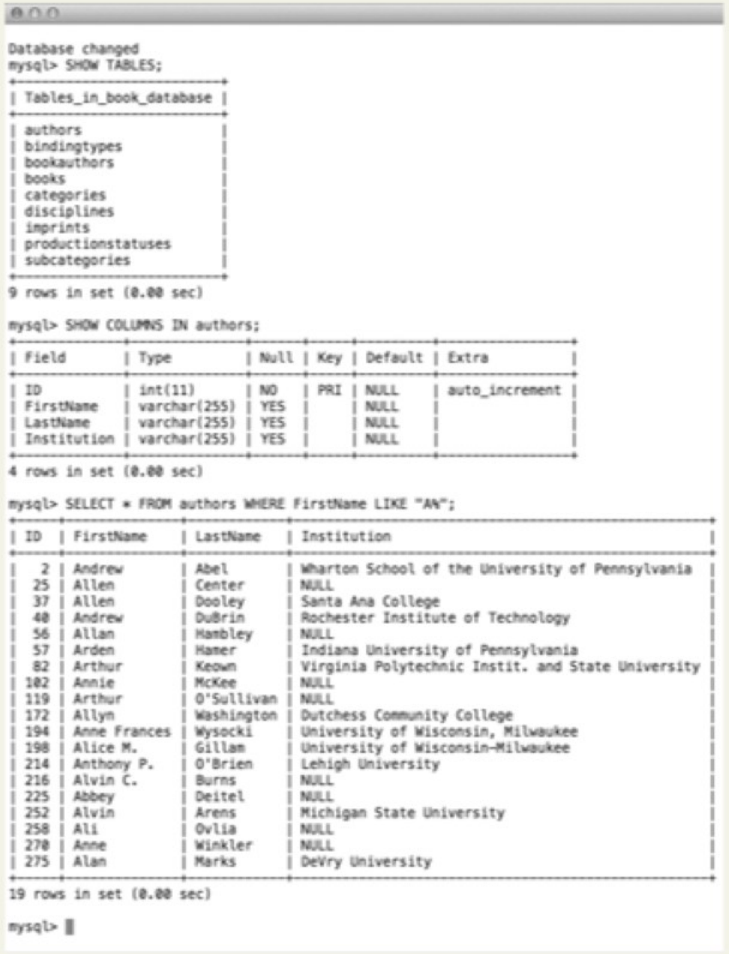
While PDO is unable to take advantage of some features of MySQL, there is a lot of merit to the fact that PDO can create database-independent PHP code

- Like many things in the web world, there is no single best choice.
- As the chapter (and book) proceed, we will standardize on the object-oriented, database-independent PDO approach.

Chapter 14

Managing a MySQL Database

Command-Line Interface



```
Database changed
mysql> SHOW TABLES;
+-------------------------+
| Tables_in_book_database |
+-------------------------+
| authors                 |
| bindingtypes            |
| bookauthors             |
| books                   |
| categories              |
| disciplines             |
| imprints                |
| productionstatuses      |
| subcategories           |
+-------------------------+
9 rows in set (0.00 sec)

mysql> SHOW COLUMNS IN authors;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| ID          | int(11)      | NO   | PRI | NULL    | auto_increment |
| FirstName   | varchar(255) | YES  |     | NULL    |                |
| LastName    | varchar(255) | YES  |     | NULL    |                |
| Institution | varchar(255) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM authors WHERE FirstName LIKE "A%";
+-----+--------------+------------+---------------------------------------------------+
| ID  | FirstName    | LastName   | Institution                                       |
+-----+--------------+------------+---------------------------------------------------+
|   2 | Andrew       | Abel       | Wharton School of the University of Pennsylvania  |
|  25 | Allen        | Center     | NULL                                              |
|  37 | Allen        | Dooley     | Santa Ana College                                 |
|  40 | Andrew       | DuBrin     | Rochester Institute of Technology                 |
|  56 | Allan        | Hambley    | NULL                                              |
|  57 | Arden        | Hamer      | Indiana University of Pennsylvania                |
|  82 | Arthur       | Keown      | Virginia Polytechnic Instit. and State University |
| 102 | Annie        | McKee      | NULL                                              |
| 119 | Arthur       | O'Sullivan | NULL                                              |
| 172 | Allyn        | Washington | Dutchess Community College                        |
| 194 | Anne Frances | Wysocki    | University of Wisconsin, Milwaukee                |
| 198 | Alice M.     | Gillam     | University of Wisconsin-Milwaukee                 |
| 214 | Anthony P.   | O'Brien    | Lehigh University                                 |
| 216 | Alvin C.     | Burns      | NULL                                              |
| 225 | Abbey        | Deitel     | NULL                                              |
| 252 | Alvin        | Arens      | Michigan State University                         |
| 258 | Ali          | Ovlia      | NULL                                              |
| 270 | Anne         | Winkler    | NULL                                              |
| 275 | Alan         | Marks      | DeVry University                                  |
+-----+--------------+------------+---------------------------------------------------+
19 rows in set (0.00 sec)

mysql> █
```

Managing a MySQL Database

Command-Line Interface

To launch an interactive MySQL command-line session, you must specify the host, username, and database name to connect to as shown below:

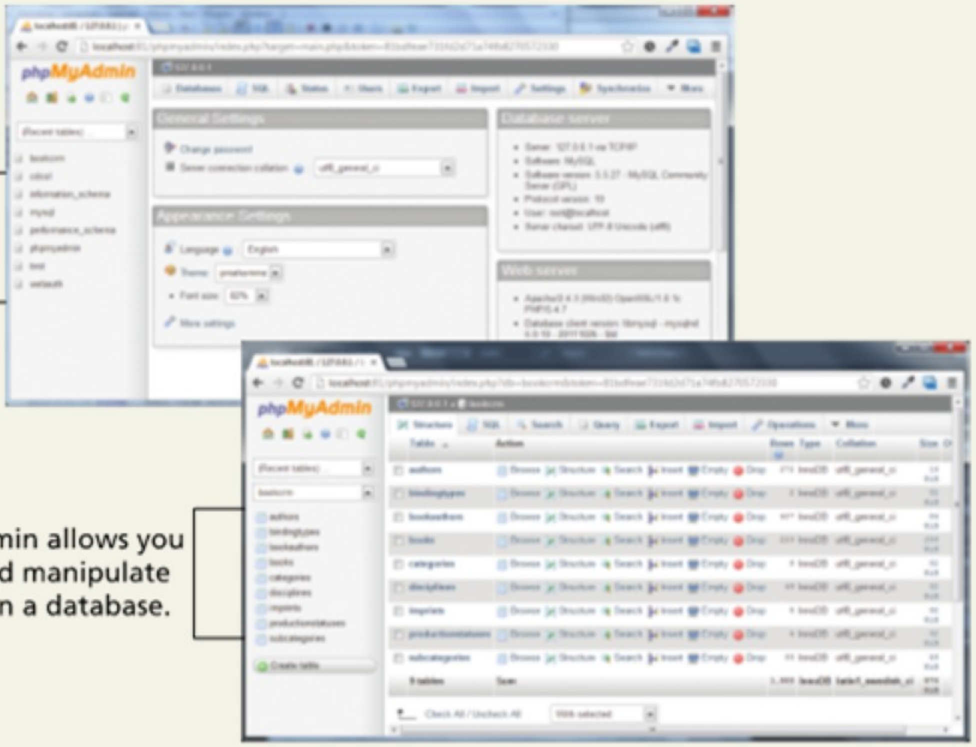
```
mysql -h 192.168.1.14 -u bookUser –p
```

To import commands from a file called commands.sql , for example, we would use the < operation:

```
mysql –h 192.168.1.14 –u bookUser –p < commands.sql
```

Managing a MySQL Database

phpMyAdmin

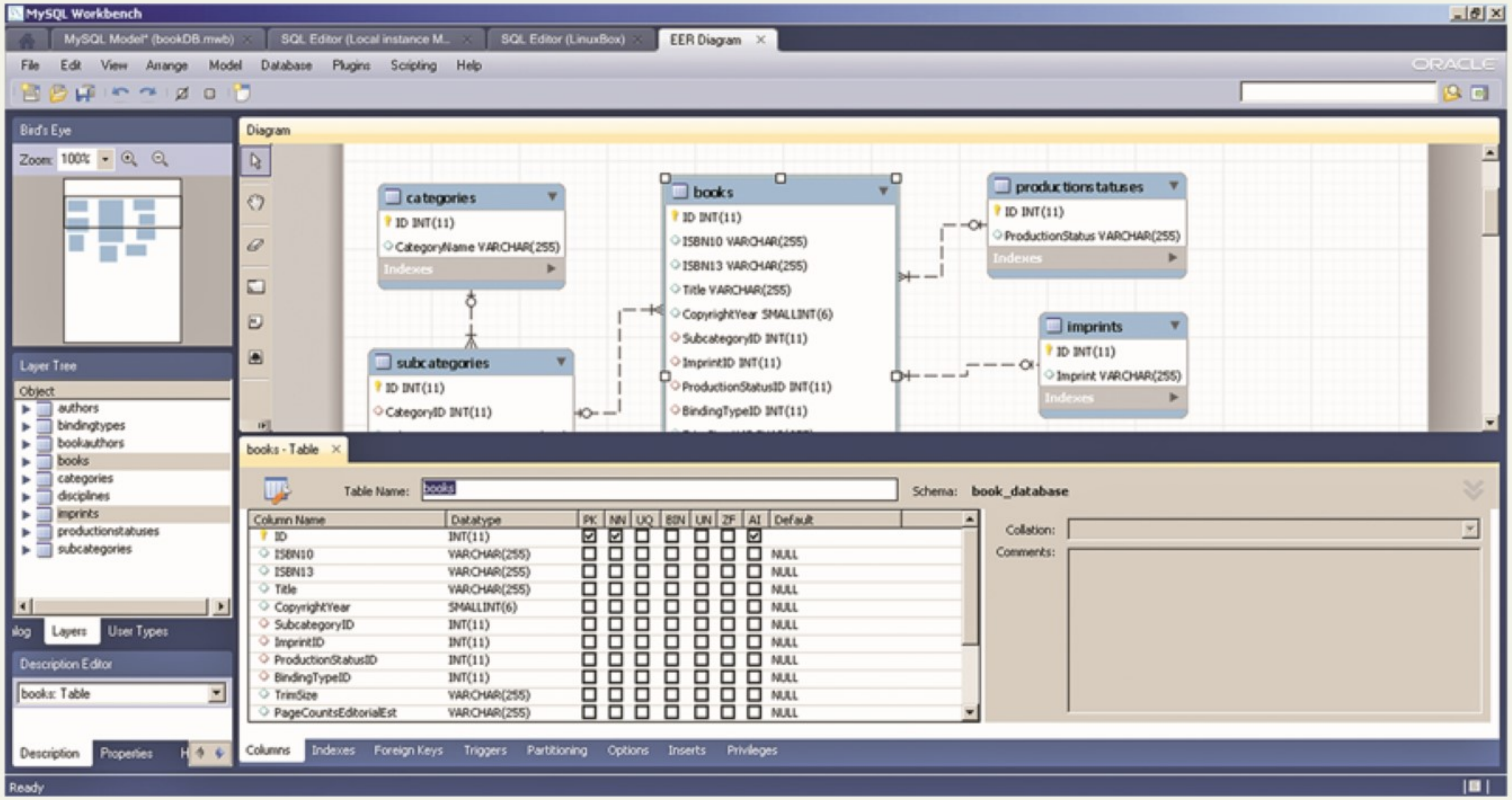


MySQL has a number of predefined databases it uses for its own operation.

phpMyAdmin allows you to view and manipulate any table in a database.

Managing a MySQL Database

MySQL Workbench



categories
- ID INT(11)
- CategoryName VARCHAR(255)
- Indexes

subcategories
- ID INT(11)
- CategoryID INT(11)

books
- ID INT(11)
- ISBN10 VARCHAR(255)
- ISBN13 VARCHAR(255)
- Title VARCHAR(255)
- CopyrightYear SMALLINT(6)
- SubcategoryID INT(11)
- ImprintID INT(11)
- ProductionStatusID INT(11)
- BindingTypeID INT(11)

productionstatuses
- ID INT(11)
- ProductionStatus VARCHAR(255)
- Indexes

imprints
- ID INT(11)
- Imprint VARCHAR(255)
- Indexes

books - Table

Table Name: books   Schema: book_database

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
ID	INT(11)	☑	☑	☐	☐	☐	☐	☑	
ISBN10	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	NULL
ISBN13	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	NULL
Title	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	NULL
CopyrightYear	SMALLINT(6)	☐	☐	☐	☐	☐	☐	☐	NULL
SubcategoryID	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
ImprintID	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
ProductionStatusID	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
BindingTypeID	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
TrimSize	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	NULL
PageCountsEditorialEst	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	NULL

Collation:
Comments:

Chapter 14

Accessing MySQL in PHP

Basic Connection Algorithm

1. Connect to the database.
2. Handle connection errors.
3. Execute the SQL query.
4. Process the results.
5. Free resources and close connection.

Accessing MySQL in PHP

Basic Connection Algorithm

```
      <?php

      try {
         $connString = "mysql:host=localhost;dbname=bookcrm";
         $user = "testuser";
         $pass = "mypassword";
  1
         $pdo = new PDO($connString,$user,$pass);
         $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

         $sql = "SELECT * FROM Categories ORDER BY CategoryName";
  3      $result = $pdo->query($sql);

         while ($row = $result->fetch()) {
  4         echo $row['ID'] . " - " . $row['CategoryName'] . "<br/>";
         }
  5      $pdo = null;
      }
      catch (PDOException $e) {
  2      die( $e->getMessage() );
      }

      ?>
```

Accessing MySQL in PHP

Connecting to a Database (mysqli peocedural)

```
// modify these variables for your installation
$host = "localhost";
$database = "bookcrm";
$user = "testuser";
$pass = "mypassword";
$connection = mysqli_connect($host, $user, $pass, $database);
```

Accessing MySQL in PHP

Connecting to a Database (PDO Object-oriented)

```
// modify these variables for your installation
$connectionString = "mysql:host=localhost;dbname=bookcrm";
$user = "testuser";
$pass = "mypassword";
$pdo = new PDO($connectionString, $user, $pass);
```

Accessing MySQL in PHP

Handling Connection Errors - mysqli

```
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME);

// mysqli_connect_errno returns the last error code

if ( mysqli_connect_errno() ) {

        die( mysqli_connect_error() );
        // die() is equivalent to exit()

}
```

Accessing MySQL in PHP

Handling Connection Errors - PDO

```
try {
        $connString = "mysql:host=localhost;dbname=bookcrm";
        $user = DBUSER;
        $pass = DBPASS;
        $pdo = new PDO($connString,$user,$pass);
        . . .
}
catch (PDOException $e) {
        die( $e->getMessage() );
}
```

Accessing MySQL in PHP

Executing the Query

```
$sql = "SELECT * FROM Categories ORDER BY CategoryName";

// returns a mysqli_result object

$result = mysqli_query($connection, $sql);

OR

$result = $pdo->query($sql);
```

Accessing MySQL in PHP

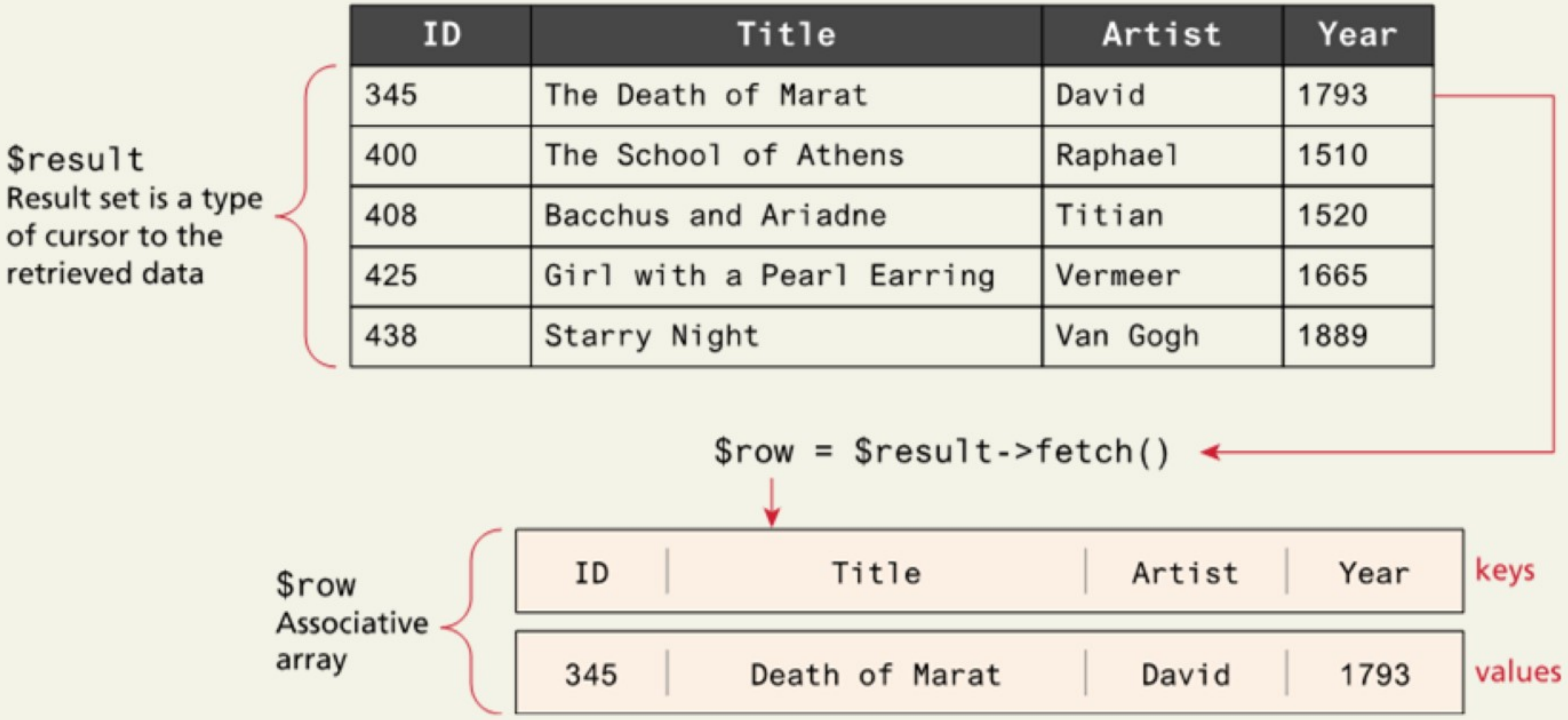
Processing the Query Results

```
$sql = "SELECT * FROM Categories ORDER BY CategoryName";
// run the query
$result = $pdo->query($sql);
// fetch a record from result set into an associative array
while ($row = $result->fetch()) {
        // the keys match the field names from the table
        echo $row['ID'] . " - " . $row['CategoryName'];
        echo "<br/>";
}
```

Accessing MySQL in PHP

Processing the Query Results

```
$sql = "select * from Paintings";
$result = $pdo->query($sql);
```



ID	Title	Artist	Year
345	The Death of Marat	David	1793
400	The School of Athens	Raphael	1510
408	Bacchus and Ariadne	Titian	1520
425	Girl with a Pearl Earring	Vermeer	1665
438	Starry Night	Van Gogh	1889

$result
Result set is a type of cursor to the retrieved data

```
$row = $result->fetch()
```

$row
Associative array

	ID	Title	Artist	Year
keys	ID	Title	Artist	Year
values	345	Death of Marat	David	1793

Accessing MySQL in PHP

Freeing Resources and Closing Connection

```
//closes the connection

mysqli_close($connection);

// closes connection and frees the resources used by the PDO object

$pdo = null;
```

Accessing MySQL in PHP

Working with Parameters

```
$sql = "UPDATE Categories SET CategoryName='Web' WHERE
        CategoryName='Business'";
$count = $pdo->exec($sql);
echo "<p>Updated " . $count . " rows</p>";
```

Accessing MySQL in PHP

Working with Parameters – Technique 1 ? Placeholders

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (?,?,?,?,
?,?,?)";
$statement = $pdo->prepare($sql);
$statement->bindValue(1, $_POST['isbn']);
$statement->bindValue(2, $_POST['title']);
$statement->bindValue(3, $_POST['year']);
$statement->bindValue(4, $_POST['imprint']);
$statement->bindValue(5, $_POST['status']);
$statement->bindValue(6, $_POST['size']);
$statement->bindValue(7, $_POST['desc']);
$statement->execute();
```

Accessing MySQL in PHP

Working with Parameters – Technique 1 ? Placeholders *with Array*

```
/* can pass an array, to be used in order */

$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (?,?,?,?,
?,?,?)";
$statement = $pdo->prepare($sql);
$statement->execute array(array($_POST['isbn'],
$_POST['title'],$_POST['year'], $_POST['imprint'], $_POST['status'],
$_POST['size'],$_POST['desc']));
```

Accessing MySQL in PHP

Working with Parameters – Technique 2 - named parameters

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (:isbn,
:title, :year, :imprint, :status, :size, :desc) ";
$statement = $pdo->prepare($sql);
$statement->bindValue(':isbn', $_POST['isbn']);
$statement->bindValue(':title', $_POST['title']);
$statement->bindValue(':year', $_POST['year']);
$statement->bindValue(':imprint', $_POST['imprint']);
$statement->bindValue(':status', $_POST['status']);
$statement->bindValue(':size', $_POST['size']);
$statement->bindValue(':desc', $_POST['desc']);
$statement->execute();
```

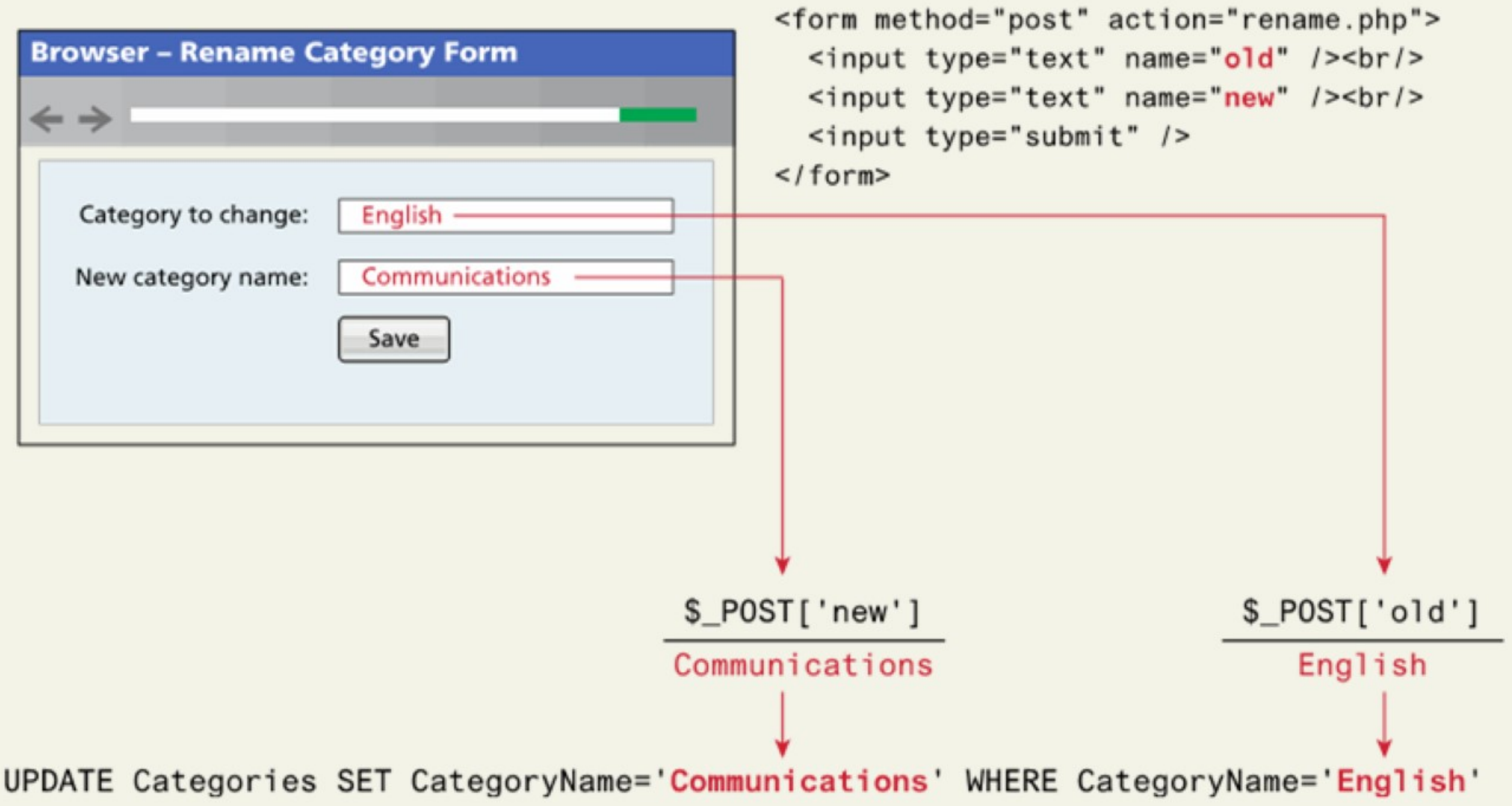
Accessing MySQL in PHP

Working with Parameters – Technique 2 - named parameters *with Array*

```
$sql = "INSERT INTO books (ISBN10, Title, CopyrightYear, ImprintId,
ProductionStatusId, TrimSize, Description) VALUES (:isbn,
:title, :year, :imprint, :status, :size, :desc) ";
$statement = $pdo->prepare($sql);
$statement->execute(array(':isbn' => $_POST['isbn'],
                          ':title'=> $_POST['title'],
                          ':year'=> $_POST['year'],
                          ':imprint'=>  $_POST['imprint'],
                          ':status'=> $_POST['status'],
                          ':size'=> $_POST['size']
                          ':desc'=> $_POST['desc']));
```

Accessing MySQL in PHP

Getting user input into a query



```
<form method="post" action="rename.php">
  <input type="text" name="old" /><br/>
  <input type="text" name="new" /><br/>
  <input type="submit" />
</form>
```

```
$_POST['new']          $_POST['old']
Communications         English
```

```
UPDATE Categories SET CategoryName='Communications' WHERE CategoryName='English'
```

Accessing MySQL in PHP

Using Transactions

```
$pdo = new PDO($connString,$user,$pass);

try {
        // begin a transaction
        $pdo->beginTransaction();
        // a set of queries: if one fails, an exception will be thrown
        $pdo->query("INSERT INTO Categories (CategoryName) VALUES ('Philosophy')");
        $pdo->query("INSERT INTO Categories (CategoryName) VALUES ('Art')");
        // if we arrive here, it means that no exception was thrown
        $pdo->commit();
} catch (Exception $e) {
        // we must rollback the transaction since an error occurred with insert
        $pdo->rollback();
}
```

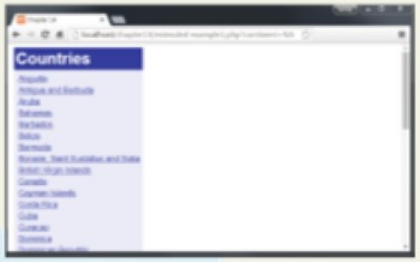
Accessing MySQL in PHP

Advanced example

```
<?php
// get database connection details
require_once('config-travel.php');

// retrieve continent from querystring
$continent = 'EU';
if (isset($_GET['continent'])) {
   $continent = $_GET['continent'];
}
?>
...
<h1>Countries</h1>
<?php
try {
    $pdo = new PDO(DBCONNSTRING,DBUSER,DBPASS);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // construct parameterized query - notice the ? parameter
    $sql = "SELECT * FROM geocountries WHERE Continent=? ORDER BY CountryName ";
    // run the prepared statement
    $statement = $pdo->prepare($sql);
    $statement->bindValue(1, $continent);
    $statement->execute();
    // output the list
    echo makeCountryList($statement);
}
catch (PDOException $e) {
   die( $e->getMessage() );
}
finally {
   $pdo = null;
}

function makeCountryList($statement) {
    $htmlList= '<ul>';
    $foundOne = false;
    while ($row = $statement->fetch()) {
        $foundOne = true;
        $htmlList .= '<li>';
        $htmlList .= '<a href="country.php?iso=' . $row['ISO'] . '">';
        $htmlList .= $row['CountryName'];
        $htmlList .= '</a>';
        $htmlList .= '</li>';
    }
    $htmlList.='</ul>';

    if ($foundOne) return $htmlList;
    return 'No countries found';
}
?>
```

config-travel.php

```
<?php
define('DBHOST', 'localhost');
define('DBNAME', 'travel');
define('DBUSER', 'testuser2');
define('DBPASS', 'mypassword');
define('DBCONNSTRING',
        'mysql:host=localhost;dbname=travel');
?>
```



Countries

Anguilla
Antigua and Barbuda
Aruba
Bahamas
Barbados
Belize
Bermuda
Bonaire, Saint Eustatius and Saba
British Virgin Islands
Canada
Cayman Islands
Costa Rica
Cuba
Curacao
Dominica

Chapter 14

1 Databases and Web Development

2 SQL

3 NoSQL

4 Database APIs

5 Managing a MySQL Database

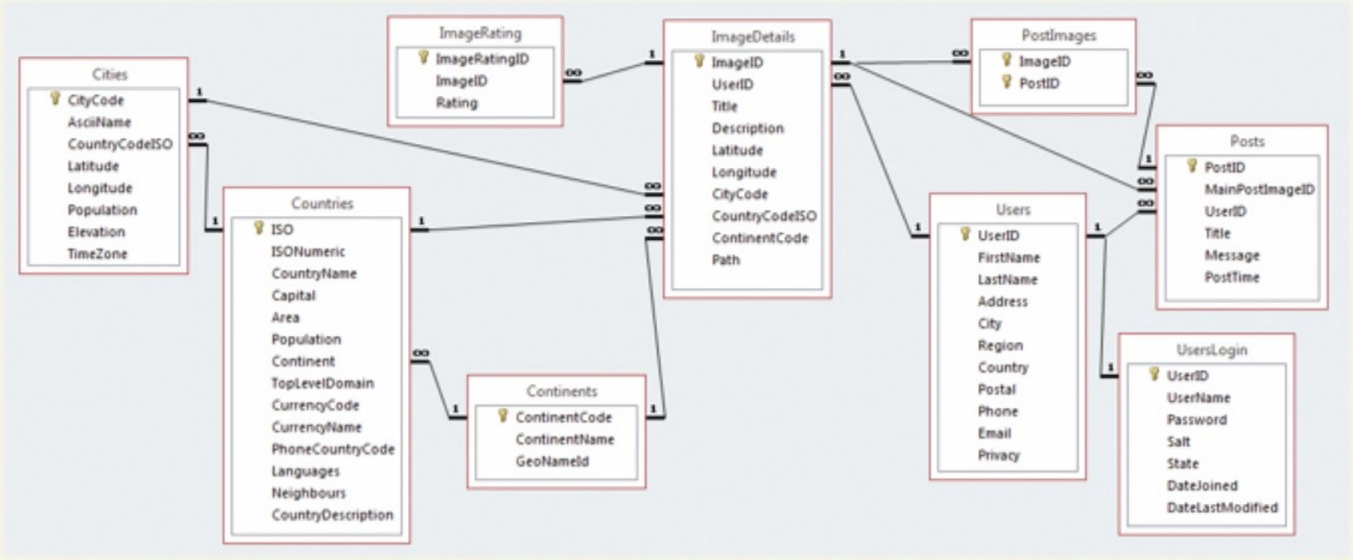
6 Accessing MySQL in PHP

7 Case Study Schemas

8 Sample Database Techniques

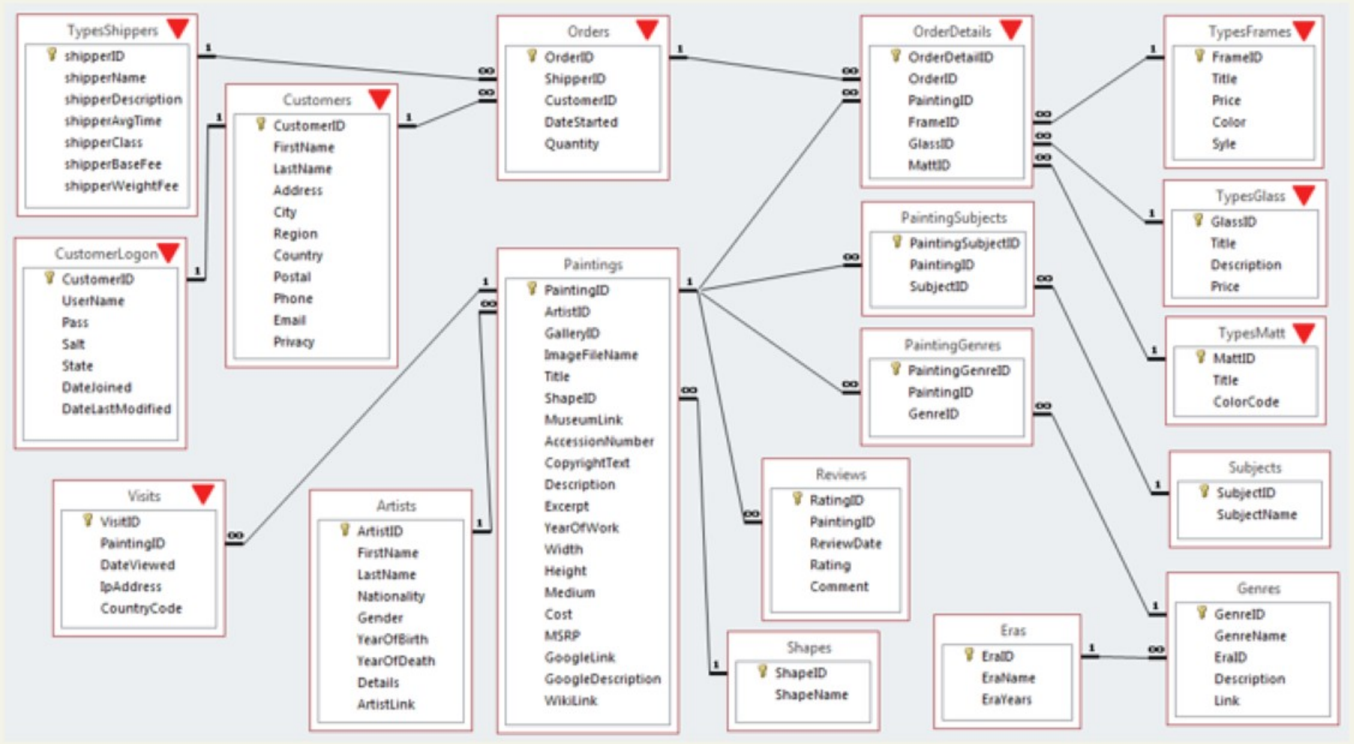
Case Study Schemas

Travel Photo Sharing Database



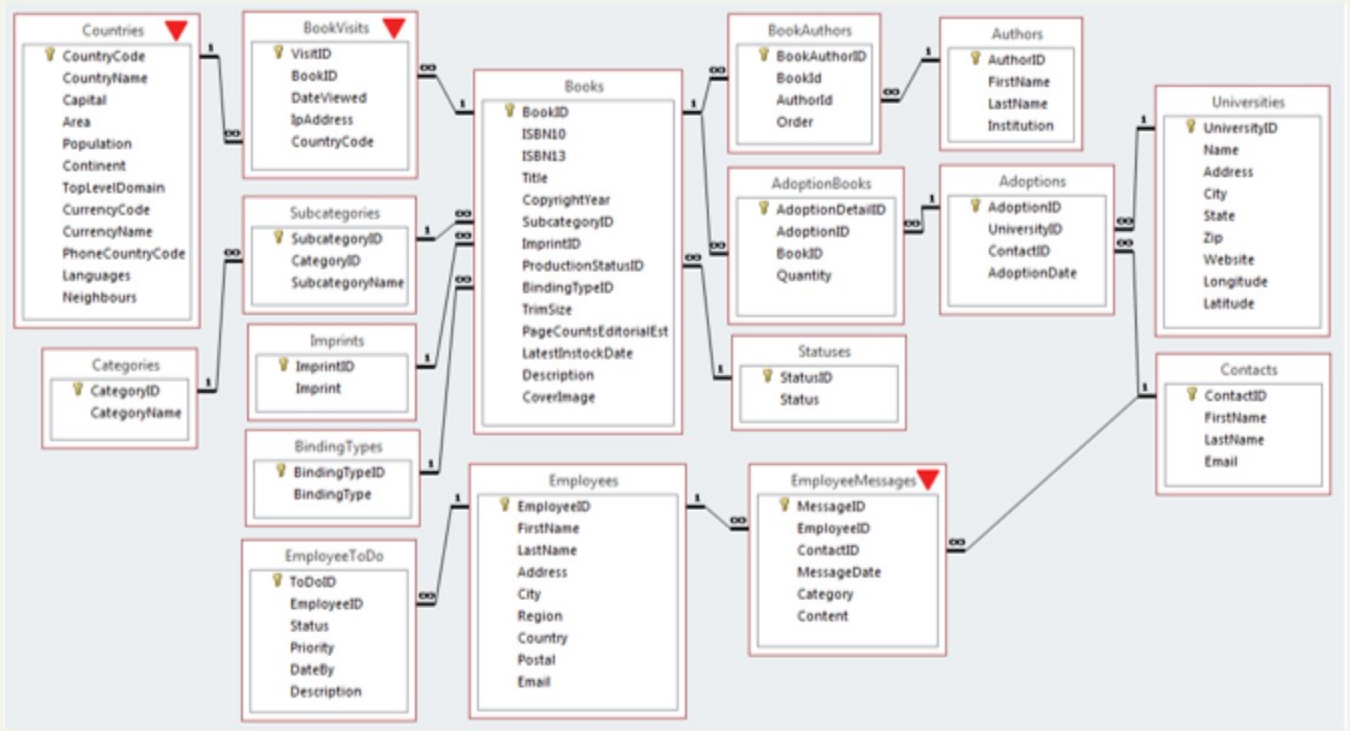
Case Study Schemas

Art Database



Case Study Schemas

Book CRM Database



Chapter 14

1 Databases and Web Development

2 SQL

3 NoSQL

4 Database APIs

5 Managing a MySQL Database

6 Accessing MySQL in PHP

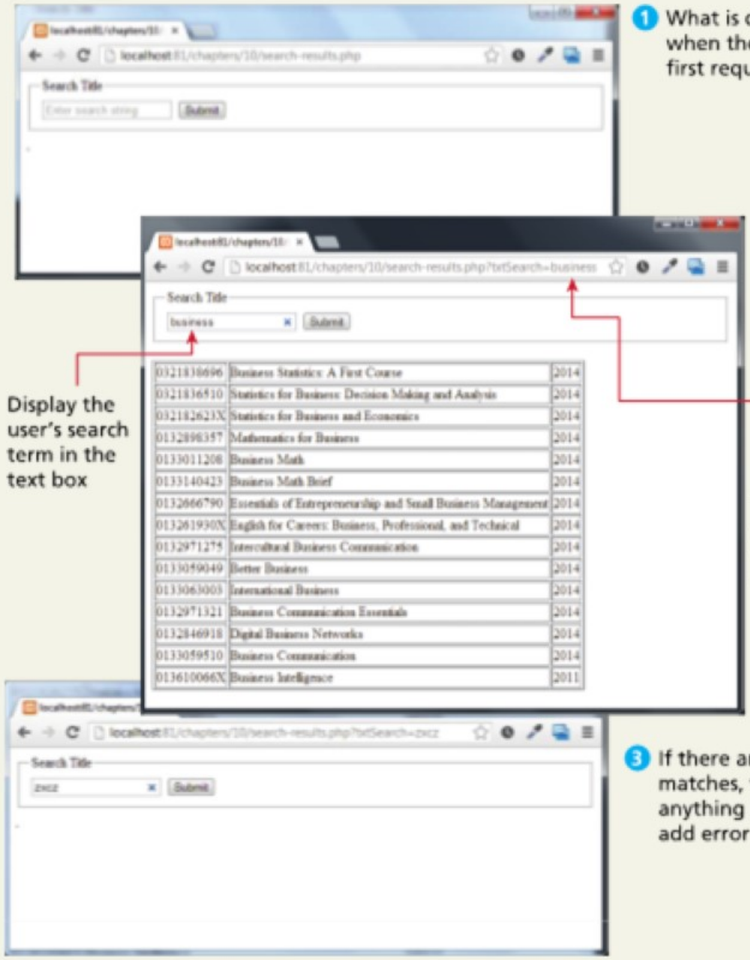
7 Case Study Schemas

8 Sample Database Techniques

Chapter 14

Sample Database Techniques

Search and Results Page



localhost:81/chapters/10/search-results.php

Search Title

Enter search string | Submit

localhost:81/chapters/10/search-results.php?txtSearch=business

Search Title

business | Submit

0321838696	Business Statistics: A First Course	2014
0321836510	Statistics for Business: Decision Making and Analysis	2014
032182623X	Statistics for Business and Economics	2014
0132898357	Mathematics for Business	2014
0133011208	Business Math	2014
0133140423	Business Math Brief	2014
0132666790	Essentials of Entrepreneurship and Small Business Management	2014
013261930X	English for Careers: Business, Professional, and Technical	2014
0132971275	Intercultural Business Communication	2014
0133059049	Better Business	2014
0133063003	International Business	2014
0132971321	Business Communication Essentials	2014
0132846918	Digital Business Networks	2014
0133059510	Business Communication	2014
013610066X	Business Intelligence	2011

localhost:81/chapters/10/search-results.php?txtSearch=zxcz

Search Title

zxcz | Submit

1 What is displayed when the page is first requested

2 Search results displayed in simple HTML table

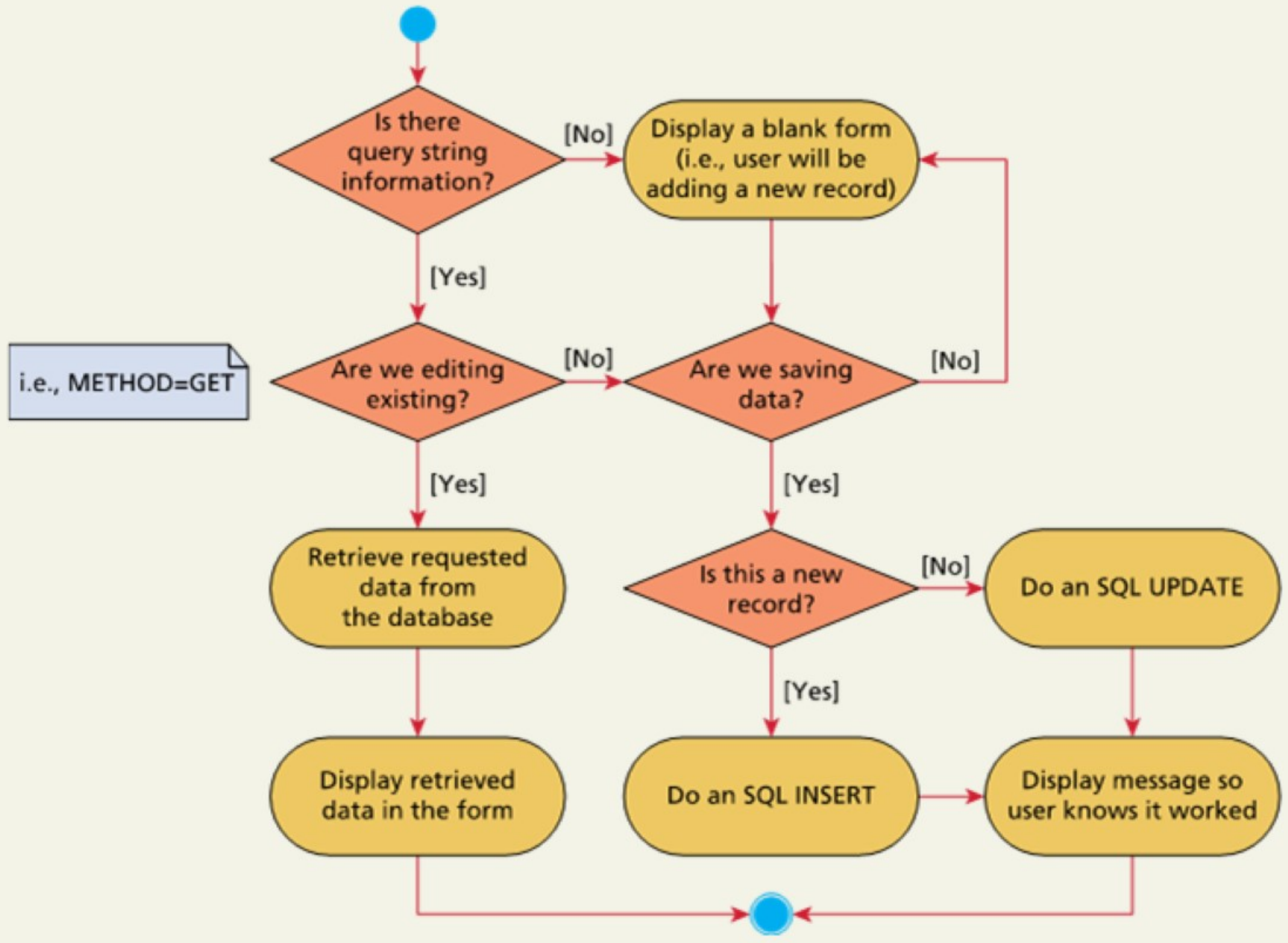
Display the user's search term in the text box

To aid in debugging, we will use HTTP GET.

3 If there are no matches, won't display anything (later we can add error messages)

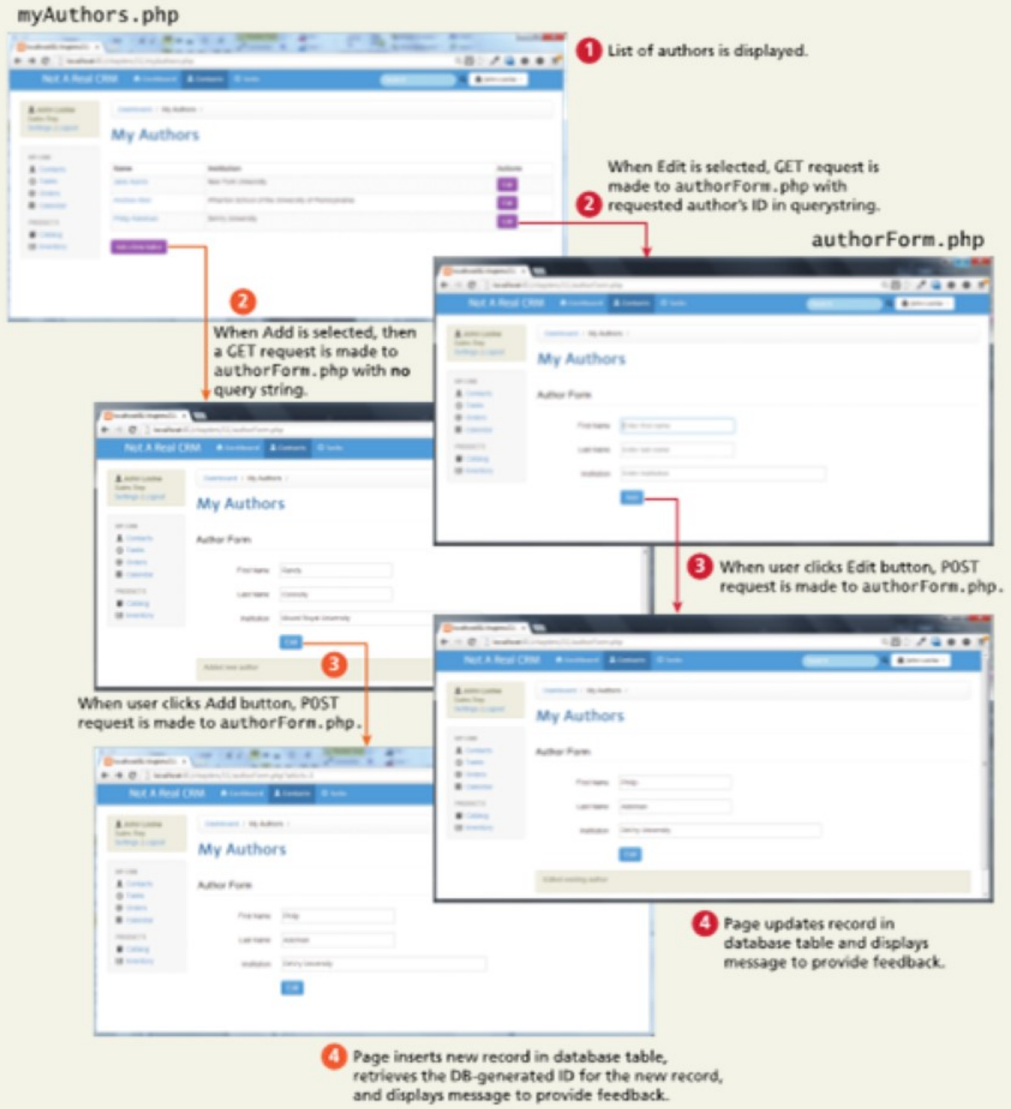
Sample Database Techniques

Editing a Record


Is there query string information?
[No]
Display a blank form (i.e., user will be adding a new record)
[Yes]
i.e., METHOD=GET
Are we editing existing?
[No]
Are we saving data?
[No]
[Yes]
[Yes]
Retrieve requested data from the database
Is this a new record?
[No]
Do an SQL UPDATE
[Yes]
Display retrieved data in the form
Do an SQL INSERT
Display message so user knows it worked

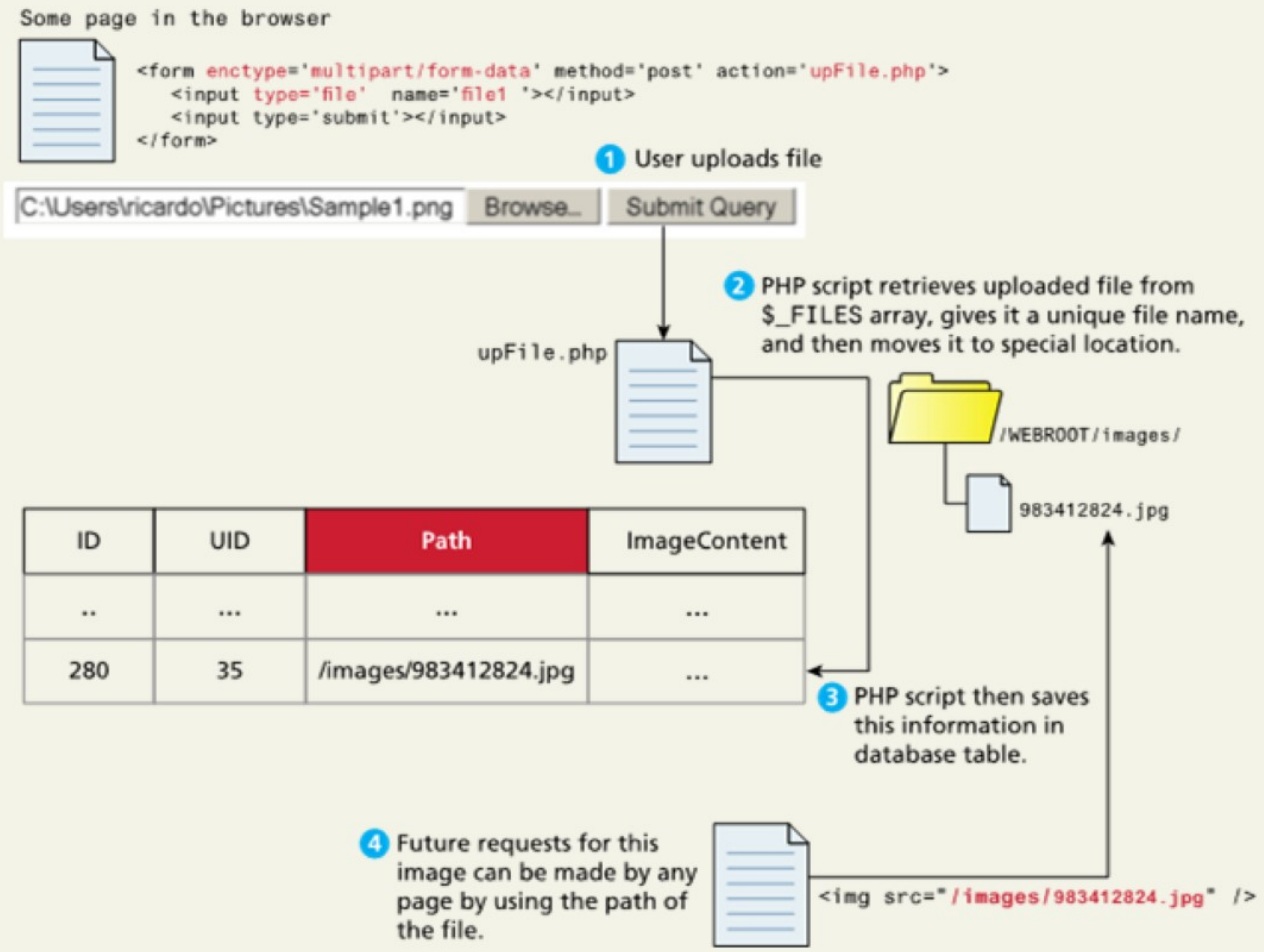
Sample Database Techniques

Editing a Record


myAuthors.php
1 List of authors is displayed.
2 When Edit is selected, GET request is made to authorForm.php with requested author's ID in querystring.
authorForm.php
2 When Add is selected, then a GET request is made to authorForm.php with no query string.
3 When user clicks Edit button, POST request is made to authorForm.php.
3 When user clicks Add button, POST request is made to authorForm.php.
4 Page updates record in database table and displays message to provide feedback.
4 Page inserts new record in database table, retrieves the DB-generated ID for the new record, and displays message to provide feedback.
My Authors

Sample Database Techniques

Saving and Displaying Raw Files in the Database



Some page in the browser

```
<form enctype='multipart/form-data' method='post' action='upFile.php'>
   <input type='file'  name='file1 '></input>
   <input type='submit'></input>
</form>
```

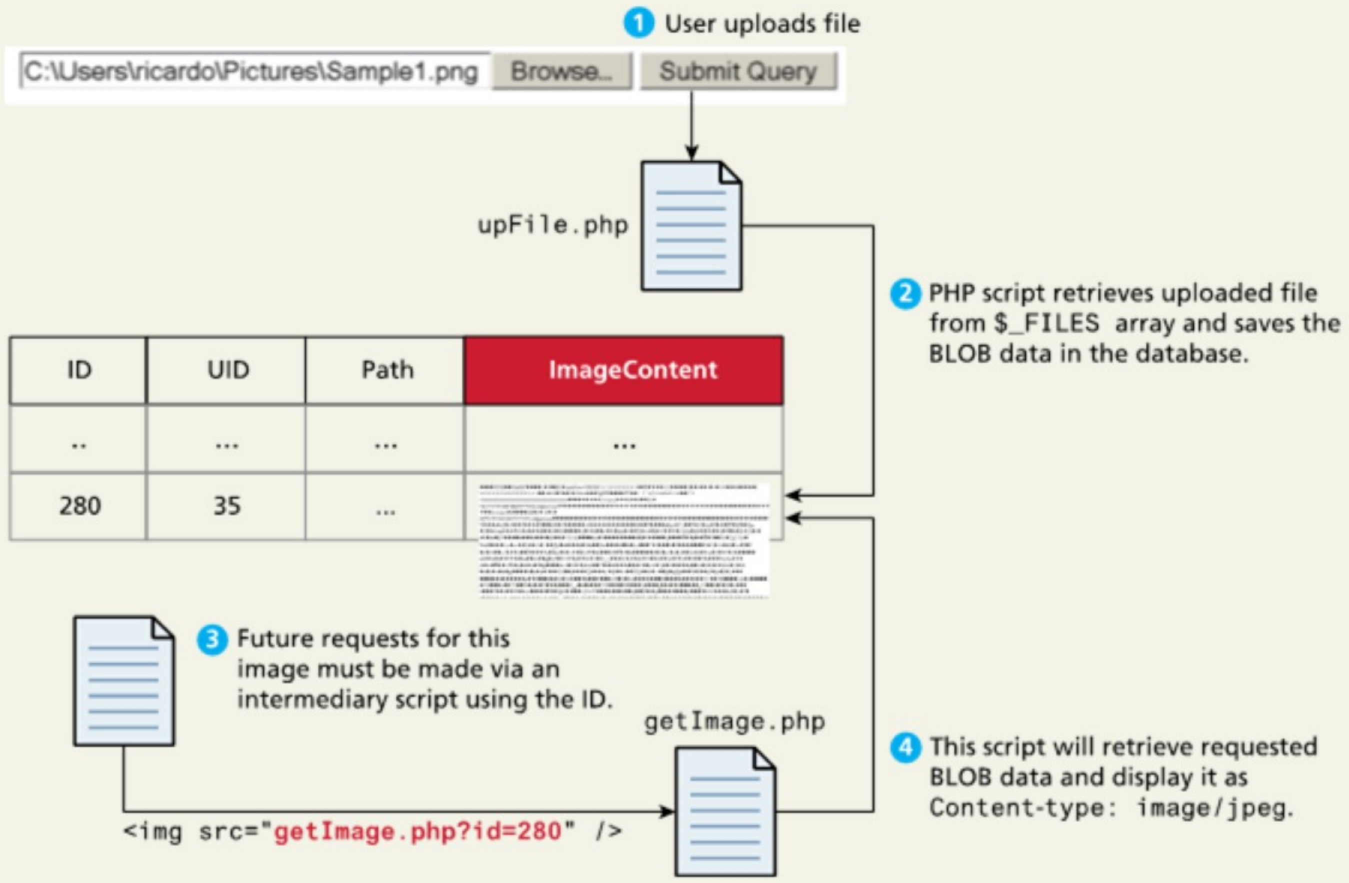
1 User uploads file
C:\Users\ricardo\Pictures\Sample1.png
Browse...
Submit Query
2 PHP script retrieves uploaded file from $_FILES array, gives it a unique file name, and then moves it to special location.
upFile.php
/WEBROOT/images/
983412824.jpg
3 PHP script then saves this information in database table.
4 Future requests for this image can be made by any page by using the path of the file.

ID	UID	Path	ImageContent
..	...	...	...
280	35	/images/983412824.jpg	...

```
<img src="/images/983412824.jpg" />
```

Sample Database Techniques

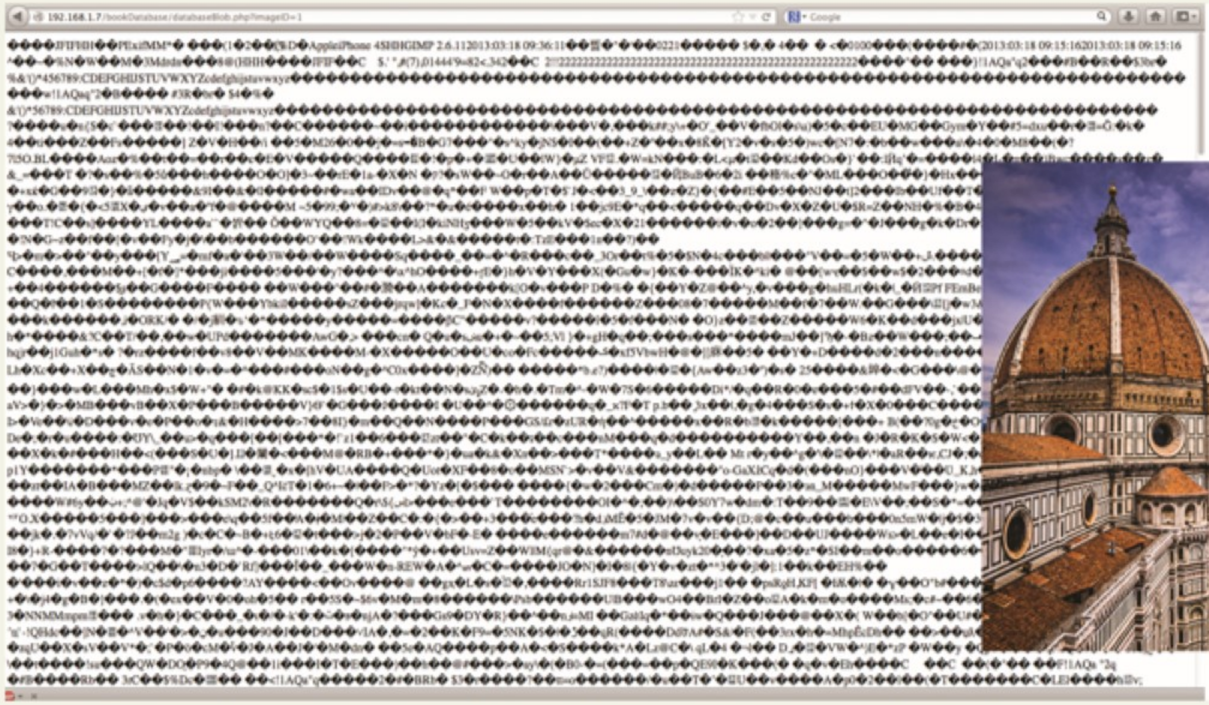
Using BLOBs to store images



1 User uploads file
C:\Users\ricardo\Pictures\Sample1.png
Browse...
Submit Query
upFile.php
2 PHP script retrieves uploaded file from $_FILES array and saves the BLOB data in the database.
ID
UID
Path
ImageContent
..
...
...
...
280
35
...
3 Future requests for this image must be made via an intermediary script using the ID.
getImage.php
<img src="getImage.php?id=280" />
4 This script will retrieve requested BLOB data and display it as Content-type: image/jpeg.

Sample Database Techniques

Headers matter



```
����JFIFHH��PExifMM*� ���(1�2��(%D�AppleiPhone 4SHHGIMP 2.6.112013:03:18 09:36:11��
^��-�%N�W��M�3Mdda���8@(HHH�����JFIF��C   5.' ",#(7),01444'9=82<.342��C
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz������������������������������
���w!1AQaq"2�B����� #3R�br� $4�%�
&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz����������������������������
[illegible]
```

Chapter 14 cont.

9 Summary

Summary

Key Terms

- abstraction layer
- aggregate functions
- binary tree
- BLOB
- column store
- composite key
- connection
- connection string
- database
- database API
- data integrity
- data definition language (DDL)
- data duplication
- data manipulation language
- database normalization
- distributed transactions
- document stores
- field
- foreign key
- hash table
- index
- inner join
- join
- key-value stores
- local transactions
- many-to-many relationship
- MySQL
- named parameter
- No-SQL database
- object-oriented API
- one-to-many relationship
- one-to-one relationship
- phpMyAdmin
- prepared statement
- primary key
- procedural API
- query
- record
- result set
- sanitization
- schema
- SQL
- SQL script
- table
- transaction
- two-phase commit

Summary

Questions?