# CPET 499/ITC 250 Web Systems

## Chapter 14
## Working with Databases
## Part 1 of 3

**Text Book:**
*** Fundamentals of Web Development, 2nd18, by Randy Connolly and Ricardo Hoar, published by Pearson**

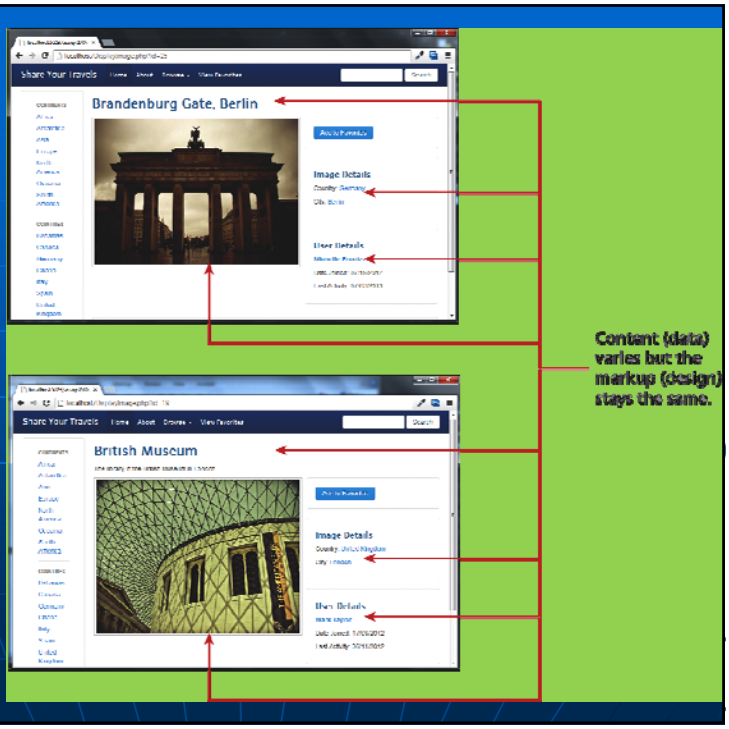
**Paul I-Hai Lin, Professor of ECET**
**http://www.etcs.pfw.edu/~lin**



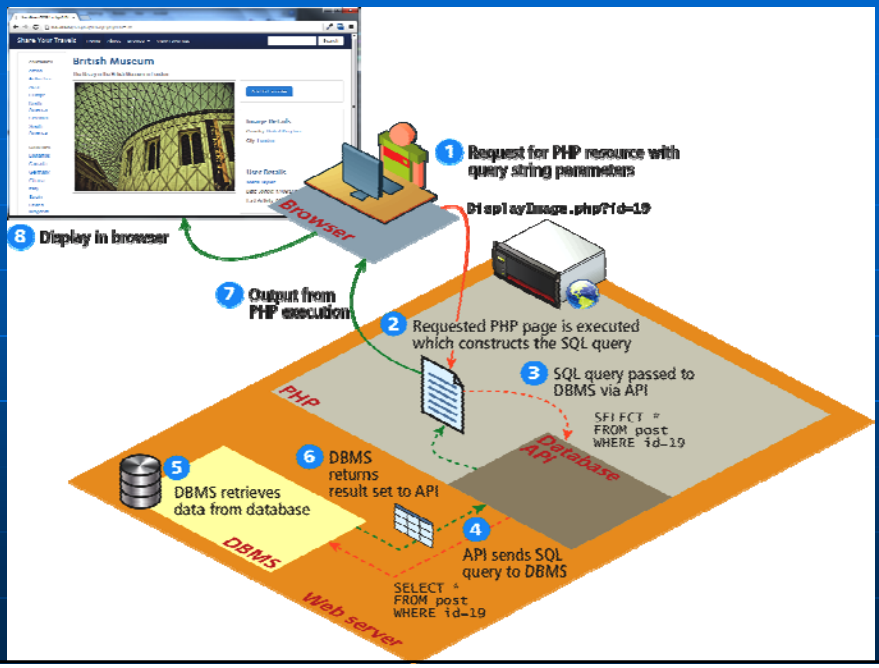
## Topics

- Database and Web Development
  - The Role of Database in Web Development
  - Database Design
  - Database Options
- SQL (Structure Query Language)
  - SELECT Statement
  - INSERT, UPDATE, and DELETE Statements
  - Transactions: Local Transactions, Distributed Transactions
  - Data Definition Statements
  - Database Indexes and Efficiency

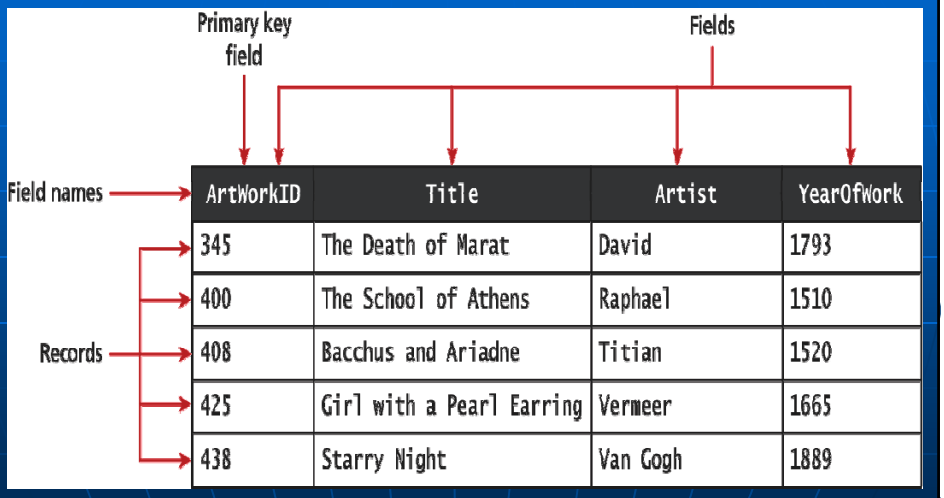
## Figure 14.1 Separating content from data




## Figure 14.2 How website use databases

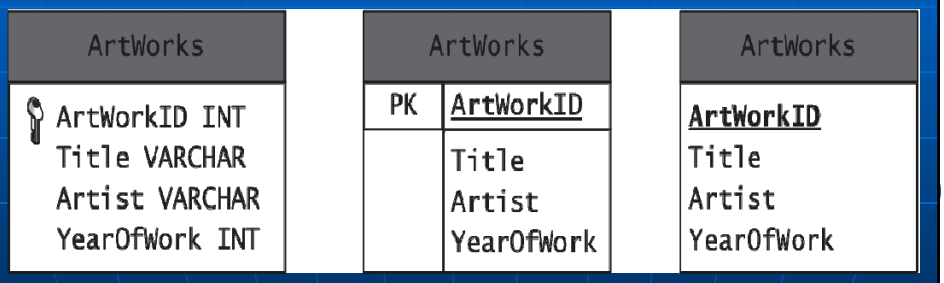


## Figure 14.3 A database table






## Figure 14.4 Diagraming a table

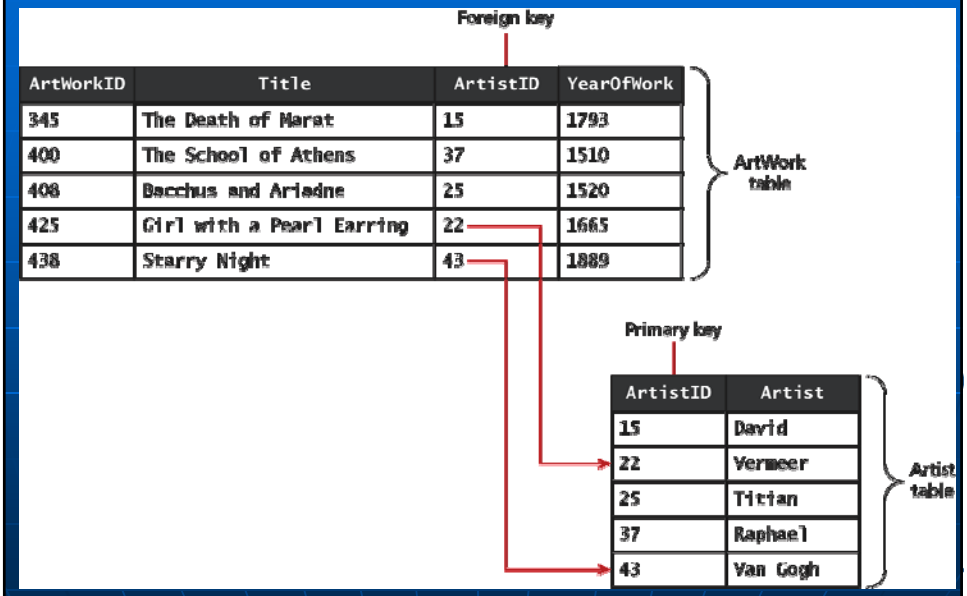


## Table 14.1 Common Database Table Data Types

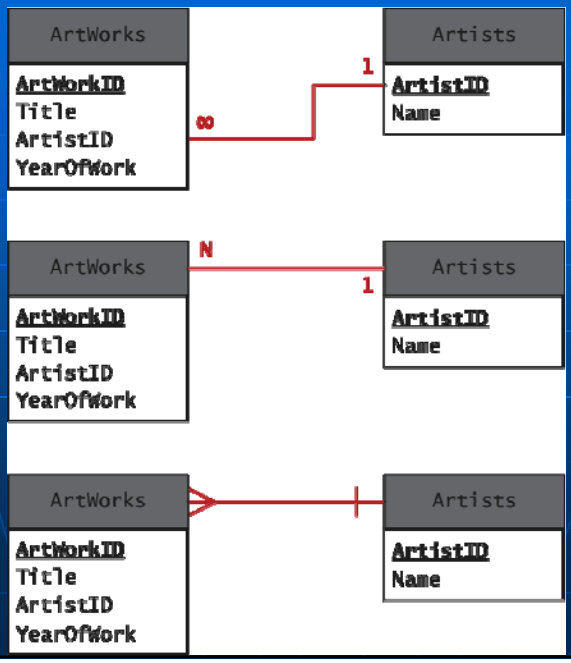
- BIT – BOOLEAN or BOOL
- BLOG – Binary Large Object (images and other data objects)
- CHAR(n) – a fixed number of character (n = number of chars) that are padded with spaces to fill the field
- DATE – also TIME and DATETIME data types
- FLOAT - also DOUBLE, DECIMAL data types
- INT - also SMALLINT data type
- VARCHAR(n) – a variable number of characters (n = maximum number of chars with no space padding)



## Figure 14.5 Foreign keys links tables

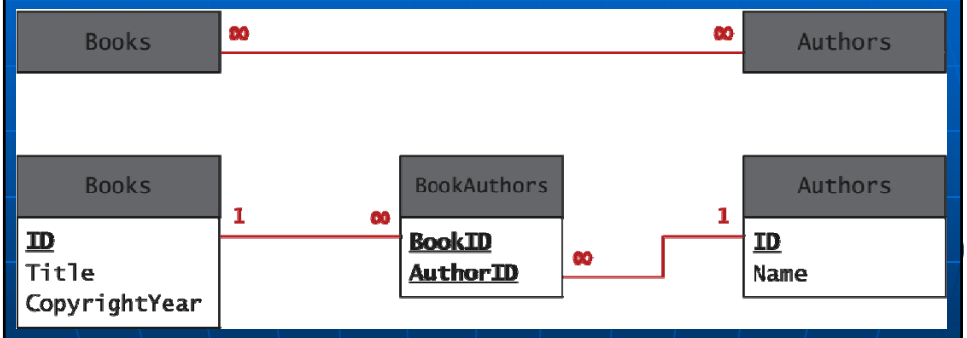


## Figure 14.6 Diagraming a one-to-many relationship




9

## Figure 14.7 Implementing a many-to-many relationship




CPET 499/ITC 250 Web Systems, Paul I. Lin

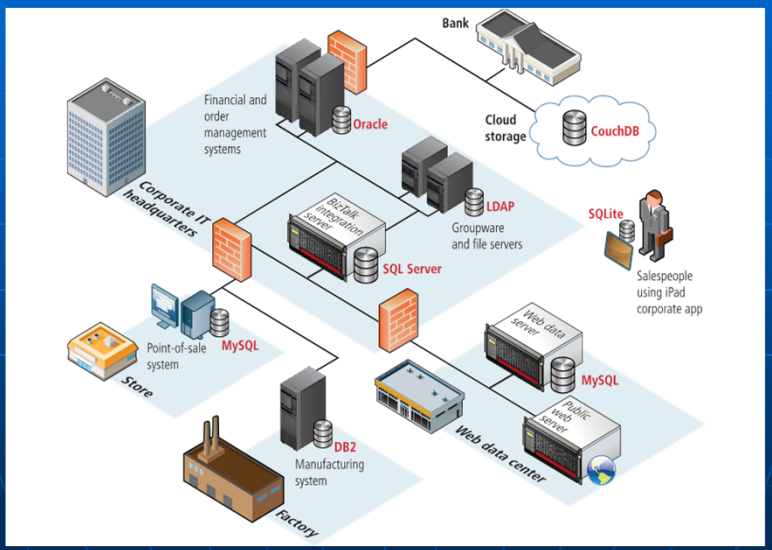
10

## Figure 14.8 Database in the Enterprise – Database Options

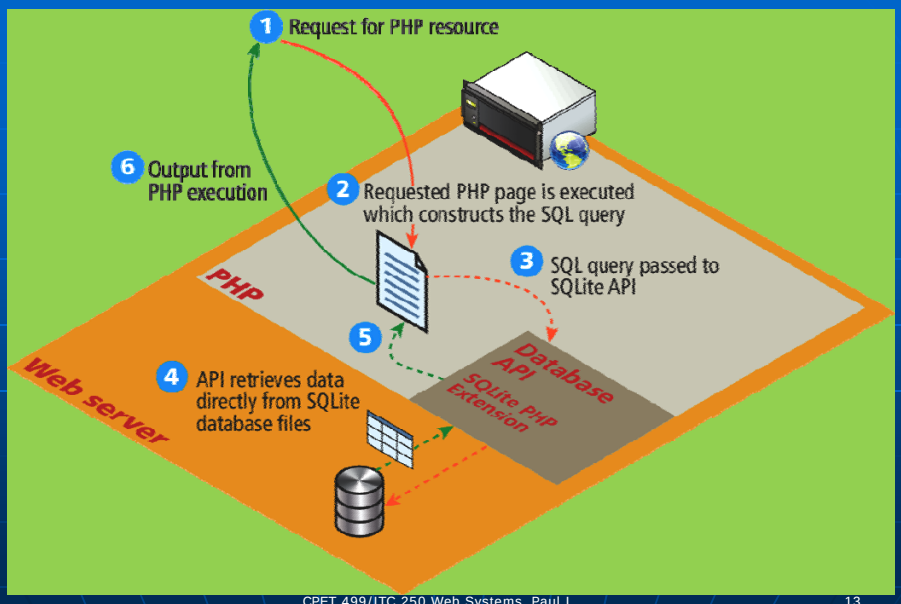
- SQL Databases
  - Financial and order management systems
  - Manufacturing system
  - Point of sales
  - Groupware and file servers
- Examples
  - DB2
  - Oracle DB
  - Microsoft SQL
  - MySQL
  - PostgreSQL
  - etc



## Figure 14.8 Database in the Enterprise – Database Options



## Figure 14.9 SQLite

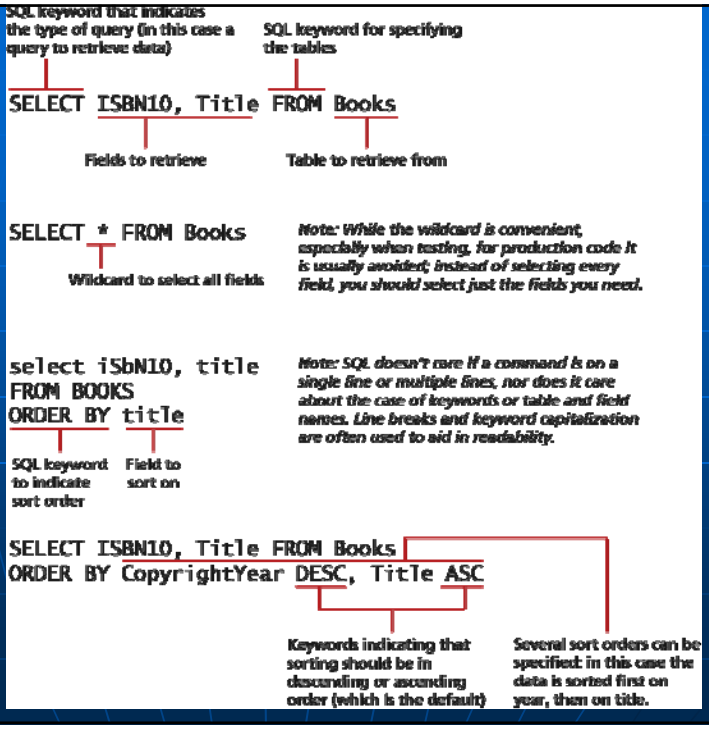





## SQL

- SELECT statement – retrieve data from the database
- INSERT statement – add new records
- UPDATE statement – update existing records
- DELETE statement – delete existing records

Figure 14.10 SQL SELECT from a single table




## Figure 14.11 Using the WHERE clause

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

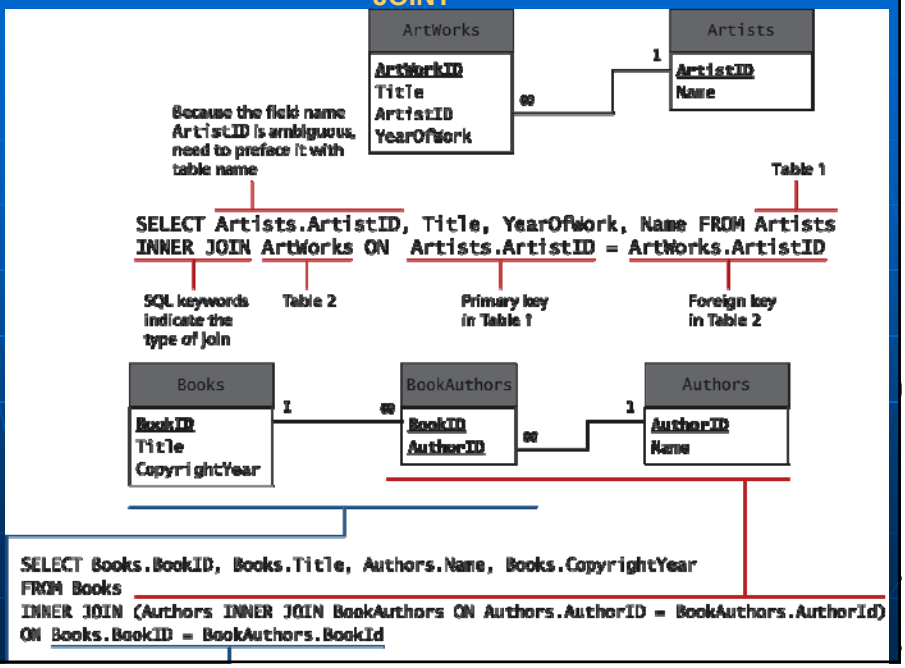
SQL keyword that indicates to return only those records whose data matches the criteria expression

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)

## Figure 14.12 SQL SELECT from multiple table using an INNER JOINT




## Figure 14.13 Using GROUP BY with aggregate functions

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 |

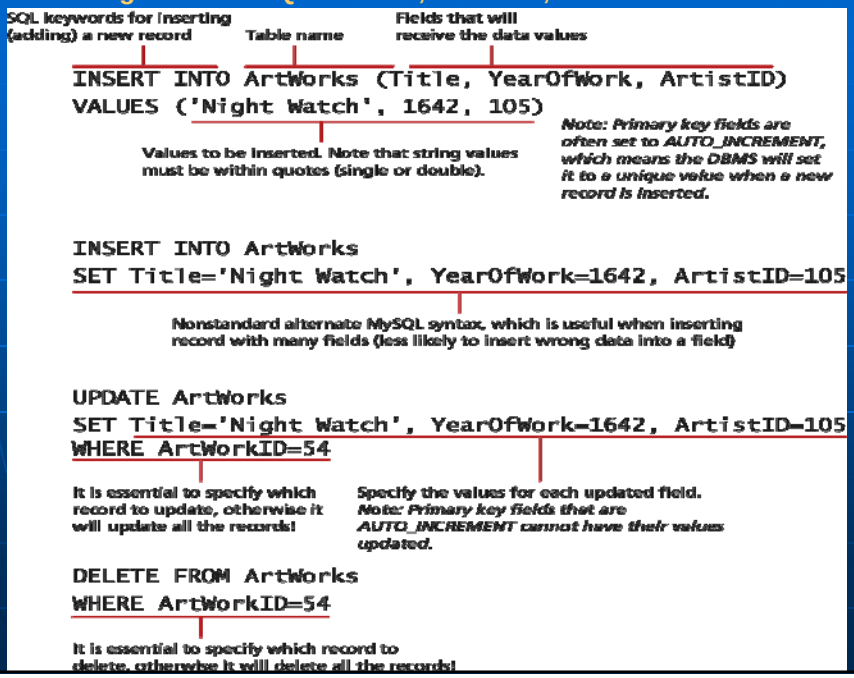
```
SELECT Nationality, Count(ArtistID) AS NumArtists
FROM Artists
GROUP BY Nationality
```

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 |

## Figure 14.14 SQL INSERT, UPDATE, and DELETE




## Local and Distributed Transactions

- **Transactions**
  - A transaction refers to a sequence of steps that are treated as a single unit, and provide a way to gracefully handle errors and keep your data properly consistent when error occur.
  - Local Transactions: transactions handled by the DBMS
  - Distributed Transactions: transactions involve multiple hosts, several of which we may have no control over.

## Local and Distributed Transactions

- Web Storefront Example
  - A customer: Checkout, Verified address, entered credit card, select shipping option
  - Click Pay for Order Button, what happen?

1. Write order records to the website database.
2. Check credit card service to see if payment is accepted.
3. If payment is accepted, send message to legacy ordering system.
4. Remove purchased item from warehouse inventory table and added to the order shipped table.
5. Send message to shipping provider



## Local and Distributed Transactions

- Web Storefront Example
- Distributed Transactions
  - Local database writes
  - External credit car processor
  - External legacy ordering system
  - External shipping systems

## Local Transactions

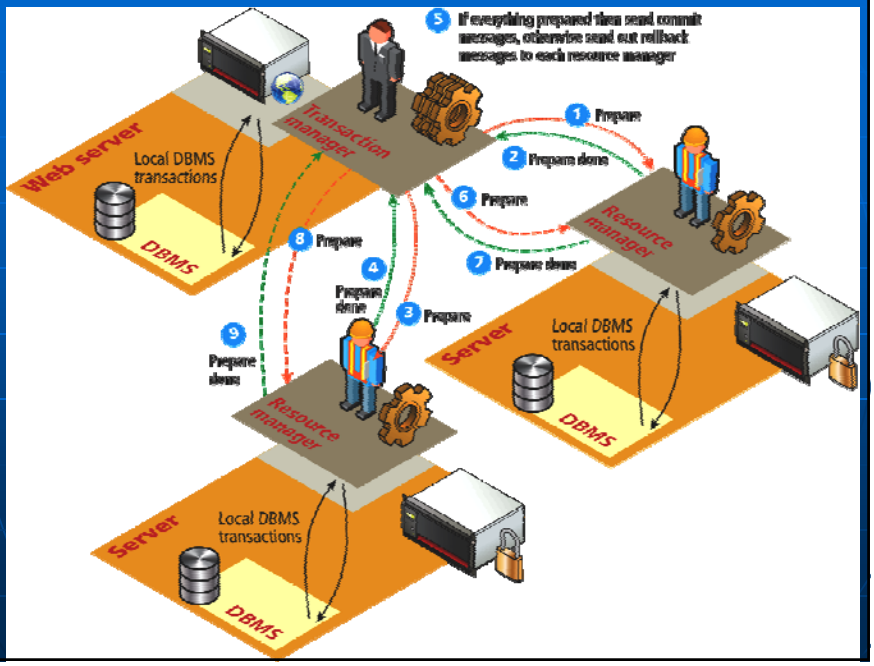
- Through SQL Statements or API calls
  - START TRACSACTION
  - COMMIT
  - ROLLBACK

- LISTING 14.1 SQL Commands for transaction processing

```
START TRANSACTION
INSERT INTO orders …
INSERT INTO orderDetails …
UPDATE inventory …
COMMIT
```



## Figure 14.15 Distributed transaction processing



## SQL

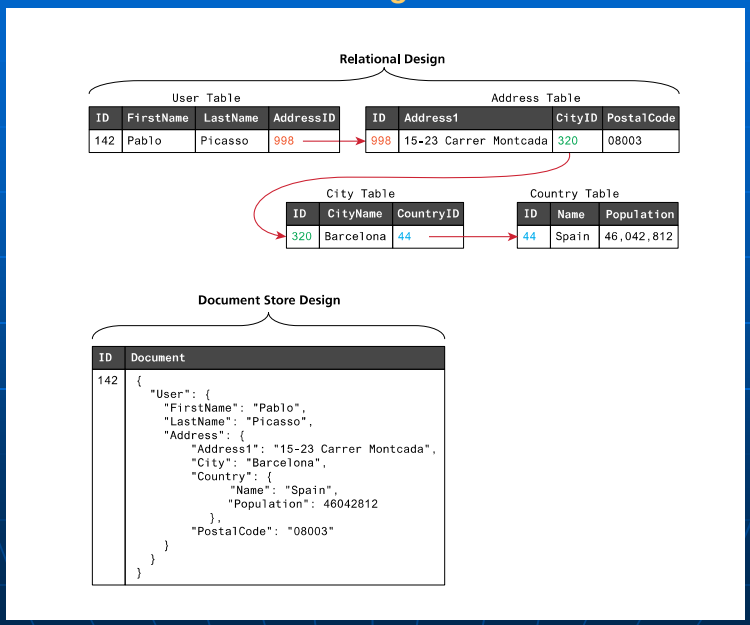
- Data Manipulation Language
  - SELECT, UPDATE, INSERT, DELETE
- Data Definition Language (DDL)
  - Creating Tables
  - Modifying the structure of a table
  - Deleting tables
  - Creating databases
  - Deleting databases
- We will use database administration statements through phpMyAdmin management tool.



## NoSQL Database in the Enterprise

- Non-SQL Databases
  - Documentation oriented, No relationship among stored data
  - Optimized to retrieve data using simple Key-Value syntax similar to PHP associative arrays
  - May be very large data sets
  - Examples:
    - Web server logs, geographical data
    - Twitter posted information
  - Non-SQL DB examples:
    - Couch DB, monoDB
    - Amazon SimpleDB , Google's Big Table

## Figure 14.17 Contrast between relational and NoSQL storage



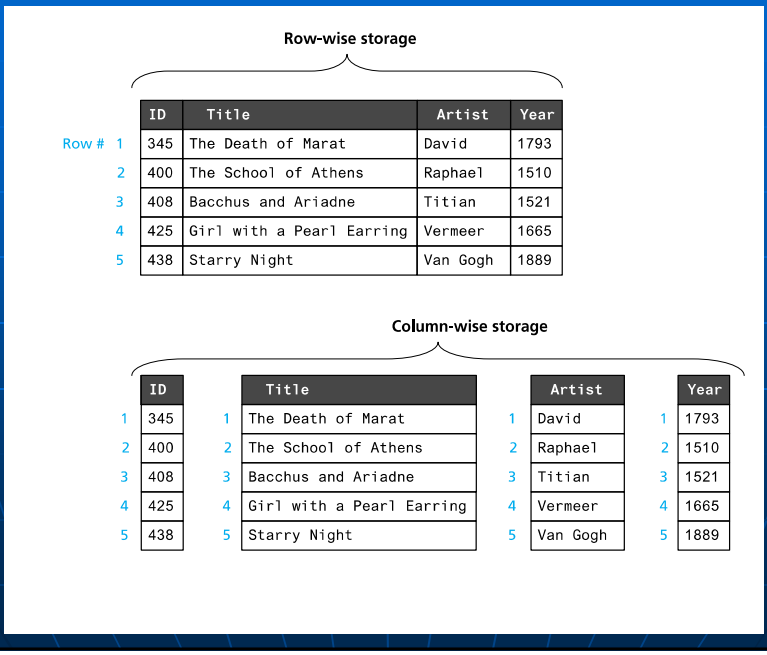

27

## Figure 14-17 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" } } } |

## Figure 14-18 Contrast between row and column wise stores






## Database APIs – PHP MySQL APIs

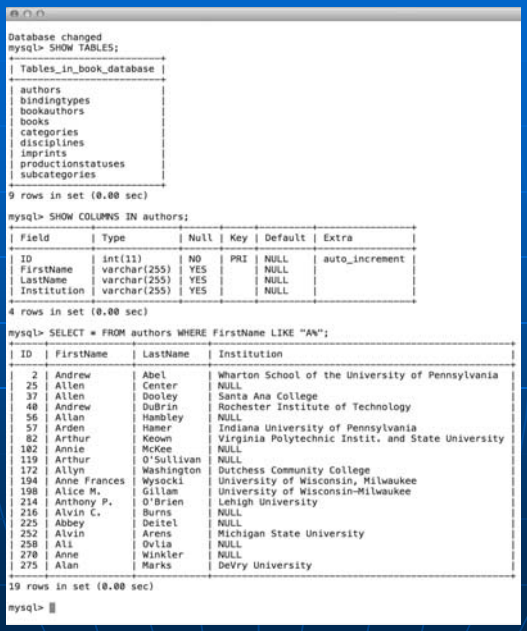
- **MySQL extension -** This was the original extension to PHP for working with MySQL and has been replaced with the newer **mysqli** extension.
- **mysqli extension -** This extension provides both a procedural and an object-oriented approach. This extension also supports most of the latest features of MySQL.
- **PHP data objects (PDOs) -** provide 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 APIs

- 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.



## 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 "An";
+-----+-------------+-----------+-----------------------------------------------------+
| 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:

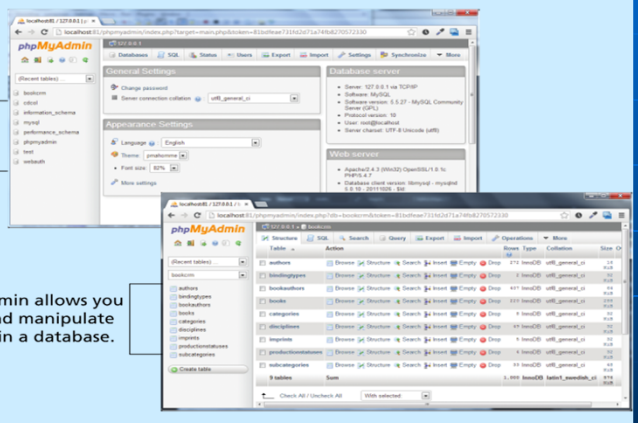
**mysql –h 192.168.1.14 –u bookUser –p < commands.sql**



## 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