Database Management Systems

Introduction

Debanga Raj Neog
Mehta Family School of Data Science and Artificial Intelligence (MFSDS&AI)
IIT Guwahati

Slides courtesy:
Prof. Ashok Singh Sairam, IITG

Course Details

- **Room**: 5205 (Core 5)
- **Attendance**: 75% to pass the course (institute policy)
- **Textbook**: Following books can be referred for resources:
    - "Database System Concepts" (6e) by Avi Silberschatz, Henry F. Korth, and S. Sudarshan
    - "Database Management Systems" (3e) by R. Ramakrishnan, J. Geherke.

Grading Scheme:

Quizzes	20%
Final examination	80%

About the Course

- Overview of data models
- Basics of the relational database model
- Developing database applications
  - Emphasis on application development
  - PHP (Hypertext Preprocessor) and SQL (Structured Query Language)
  - Supported by DA 215 (DBMS Lab)
- At the end of the course, you should be able to understand database requirements and translate it into a valid database design

Database Systems

- DBMS contains information about a particular enterprise
  - Collection of interrelated data that is valuable, large, and accessed by multiple users and applications, often at the same time
  - Set of programs to access and update the data
  - An environment that is both *convenient* and *efficient* to use
- A modern database system is a complex software system whose task is to manage a large, complex collection of data.
- Databases touch all aspects of our lives

Database Applications Examples

- **Enterprise Information:**
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes.
- **Manufacturing**: management of production, inventory, orders, supply chain.
- **Banking and finance:**
  - Customer information, accounts, loans, and banking transactions.
  - Credit card transactions
  - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- **Universities:** registration, grades

Database Applications Examples

- **Airlines**: reservations, schedules
- **Telecommunication**: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- **Web-based services:**
    - Online retailers: order tracking, customized recommendations
    - Online advertisements
- **Document databases**
- **Navigation systems:** For maintaining the locations of various places of interest along with the exact routes of roads, train systems, buses, etc.

Purpose of Database Systems

In the early days, database applications were built directly on top of file systems, which leads to:

- Data redundancy and inconsistency: data is stored in multiple locations and file formats resulting in duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Purpose of Database Systems (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Ex: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Data Model

- Data Model
  - mathematical formalism (or conceptual way) for describing the data

- The description generally consists of three parts:
  - Structure of the data
  - Operations on the data
  - Constraints on the data

Structure of the data

- Schema (e.g., table names, attribute names)
  - Describe the **conceptual** structure of the data

- Different from data structure (e.g., list, array)
  - Data structure can be seen as a **physical** data model

Operations on the Data

- Two kinds of operations
  - operations that retrieve information
  - operations that change the database
- Query language (e.g., SQL)
  - Describe what operations that can be performed on data
- Different from programming languages (e.g., PHP)
  - Support a set of limited operations
  - Allow for query optimizations

Constraints on the data

- Why does it matter?
  - Ensure the correctness of data
- Constraints (e.g., balance > 0, acctid# is unique)
  - describe limitations on what the data can be.
- Different kinds of constraints
  - Domain constraints, Eg. date field
  - Integrity constraints, Eg. acctd_id cannot be null

Relational Model

- Main concept: **relation**, basically a table with rows and columns.

Name	Roll Number	Subject	CPI
John	12345	DBMS	8.9
Tom	45678	DBMS	7.8

Students

- Every relation has a **schema**, which describes the columns, or fields.

  students(name: string, roll number: integer, subject: string, CPI: real)

- Integrity constraints: Conditions that the records in a relation must satisfy.

  E.g. Roll number must be unique

Database Management System (DBMS)

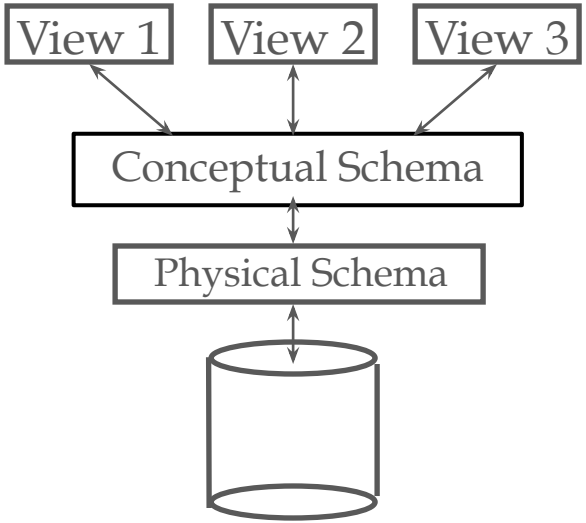
- Database: a large, integrated collection of data
- Models a real world enterprise
  - Entities (e.g. Students, Courses, Hospitals)
  - Relationships (e.g. Rohit is enrolled in the DBMS course)
  - Constraints (e.g. At least 5 students are enrolled in a course)
- Database Management System (DBMS)
  - A software package designed to store, manage and facilitate access to databases
  - Support CRUD (Create, Read, Update and Delete) operations

Advantages of DBMS

- Data Abstraction
- Concurrent Access and crash recovery
- Data Integrity and Security
- Efficient data access
- Data Administration
- Reduced application development time

Levels of abstraction

- Many views (also called external schema), single conceptual (logical) schema and physical schema.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.



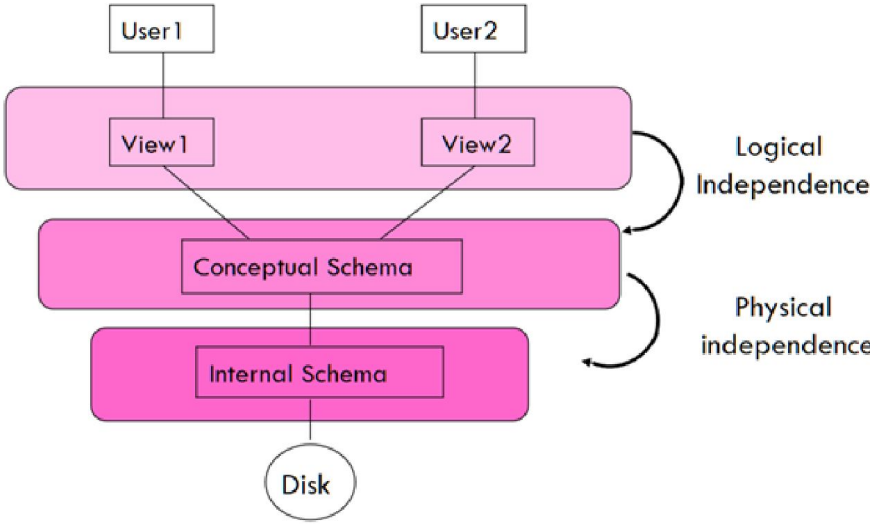
- Schemas are defined using DDL; data is modified/queried using DML.

Ex: University Database

- External Schema (View):
  - *Course_taught(cid:string, cname: string, fid: string, fname: string)*
- Conceptual schema:
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Faculty(fid: string, fname: string, sal:real)*
  - *Teaches(fid: string, cid:string)*
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Courses.

Data Independence

- Data independence is one of the most important advantage of using a DBMS
  - Logical data independence: Protection from changes in logical structure of data.
  - Physical data independence: Protection from changes in physical structure of data.



Ex: Data Independence

- Conceptual schema: Changes in faculty relation
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Faculty(fid: string, fname: string, sal:real)* *changed to two relations*
    Faculty_public(fid: string, fname: string, address: string)
    Faculty_private(fid: string, sal: real);
  - *Teaches(fid: string, cid:string)*
- External Schema (View): remains unchanged!
  - *Course_taught(cid:string, cname: string, fid: string, fname: string)*

Concurrency control

- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow,
  - keep the CPU working on several user programs concurrently.
- Interleaving actions of different user programs can lead to inconsistency:
  - e.g., cheque is cleared while account balance is being computed.
- DBMS ensures such anomalies don’t arise
  - Give users the illusion of a single-user system.

Files vs. Databases

DBMS	File System
DBMS is a collection of data.	File system is a collection of data.
abstract view of data, hides details	Provides detail of data representation and storage
Concurrent access, recovery from crashes	Does not have a crash mechanism
Data integrity and security	Difficult to protect a file
Provide sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.

People working with databases

- Database implementers
  - Oracle, MySQL, SqlServer (from MS), Ms-Access, PostgreSQL,
- Application Programmer
  - Build packages that facilitate data access for end users
  - Build enterprise applications/web services on top of DBMS
- Database Administrator
  - Design conceptual and physical schema
  - Security and authorization – prevent unauthorised access
  - Data availability and recovery
  - Database tuning: Modify the DB to cater to users needs which evolve over time

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley (Michael Stonebraker) begins Ingres prototype
    - Oracle releases first commercial relational database
  - High-performance (for the era) transaction processing

History of Database Systems (Cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
    - Wisconsin, IBM, Teradata
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce

History of Database Systems (Cont.)

- 2000s
  - Big data storage systems
    - Google BigTable, Yahoo PNuts, Amazon,
    - “NoSQL” systems.
  - Big data analysis: beyond SQL
    - Map reduce and friends
- 2010s
  - SQL reloaded
    - SQL front end to Map Reduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases

Summary

- **Database**: contains one or more relation (table)
- **Relation** (or table): contains tuples (and attributes)
- **Tuple** (or row): a set of attributes which generally represent an entity (person, music track, …)
- **Attribute** (column or field): One of possibly many elements of the data corresponding to the object represented by the row
- **Data Model**: Description of how data is stored and processed
- **Schema**: Description of particular collection of data using a data model

Database Management Systems

PHP Basics

Debanga Raj Neog
Mehta Family School of Data Science and Artificial Intelligence (MFSDS&AI)
IIT Guwahati

Slides courtesy:
Prof. Ashok Singh Sairam, IITG

Why PHP?

- PHP: **P**HP **H**ypertext **P**reprocessor
- Widely used in Web Application Development
- A server-side scripting language executed on the server rather than user's web browser
- Universal language for the Web
- Extends HTML pages by adding server executed code segments to HTML pages
    - Output of PHP is merged into HTML
    - It is a server side scripting language
- Connect to database and run SQL queries

PHP: Topics for Today

- Variables
- Datatypes
- Operators
    - Mathematical/General operators
    - Comparison operators
    - Logical operators
- Control structures Loops, Output statements
- Casting
- File Operations
- …

About PHP

- Syntax inspired by Perl
  - Dollar sign to start variable names, associative arrays
- Syntax inspired by C
  - Curly braces, semicolons, no significant whitespace
- Extends HTML to add segments of PHP within an HTML file
- White space is generally ignored (not in strings)
- Code block: { //code here }

PHP	**Perl**

`<?php`
`  $x = 5;`
`  $y = 10;`
`  $sum = $x + $y;`
`?>`
	`my $x = 5;`
	`my $y = 10;`
	`my $sum = $x + $y;`

PHP	**C**

`<?php`
`  function greet() {`
`  echo "Hello,World!";`
`  }`
`?>`
	`void greet() {`
	`  printf("Hello,World!");`
	`}`

Starting with PHP

- All PHP commands are enclosed within special start and end tags:

```
<?php
   ...PHP Code...
?>
```

- When the code is executed, PHP converts the code inside the “<?php” and “?>” tags to regular HTML code!
- Everything outside these tags is ignored by PHP and returned as is

First PHP Program

- Start the web server
- Write a html code
  - Ex:

```
<!DOCTYPE html>
<html>
    <h1>
      My first PHP program
    </h1>
</html>
```

- Save the file, say index.php in a subfolder inside **htdocs** folder
- Run the code: http://localhost/<folder_name>/index.php

Extend a Little More

- Add a PHP command to the file.

```
<!DOCTYPE html>
<html>
     <h1> My first PHP program </h1>
     <p>
     <?php
          echo “Hello world\n”;
     ?>
     </p>
</html>
```

Variables in PHP

- PHP variables must begin with a “$” sign followed by a letter or underscore, followed by any number of letters/numbers/underscores
- There is no need to declare the variable
- Implicit type casting of the variable
- Case-sensitive (`$Foo != $foo != $f0o`)
- Global and locally-scoped variables
  - Global variables can be used anywhere
  - Local variables restricted to a function or class

Reference: https://www.php.net/manual/en/language.variables.basics.php

Data Types in PHP

- There are 4 basic data types
  - Boolean – Specifies a TRUE or FALSE value, `$click = true;`
  - Integers – can be defined as, `$age=8;`
  - Floating point – fractional numbers, `$pi=3.14;`
  - Strings - Sequence of characters. May be enclosed in either double quotes or single quotes.
    - More about Strings in the next slide
- The data type of a variable can be retrieved by the function `gettype($variable_name)`

String

- A string is a sequence of characters
- String literals can use single or double quotes
- In single quoted strings, variable values are not expanded

```
$age = 37;
$stringTest = 'I am $age years old'; // output: I am $age years old
$stringTest = “I am $age years old”; // output: I am 37 years old
```

- Function inside double quotes will not be executed
- Concatenation is “.”

```
$conc = “is ” .  “a ”  .  “composed ”  .  “string”;
echo $conc; // output: is a composed string
```

Comments

Multiple ways to add comments in PHP

```
1. echo ‘This is a test’; // C++ style comment
2. /* C style multiline

       comment */

3. echo ‘Another type of comment’; # Shell-style comment
```

Expressions

- Mathematical operators: `+ - / * %`
- Implicit type conversion

```
<?php
  $result = “6” + 7;
  echo “$result”; // Output is 13
?>
```

- “6” is converted to an integer because the operands of “+” is numeric

Mathematical Operators

Operator	Meaning
=	Assignment
+	Addition
-	Subtraction
/	Division
%	Division, returns modulus
.	String concatenation
++,--, +=,-=,….	Side-effect assignment

Comparison Operators

Operator	Meaning
==	Equal to
===	Equal to and of same type
!=	Not equal to or Not of same type
<>	Not equal to
<, <=, >, >=	Less than, Less than equal to etc.
?=	Ternary operator, test if value is true or false

Operator Precedence

Precdence	Operator
1	!
2	*, /, %
3	+, -, .
4	<, <=, >, >=
5	==, !=, ===, !==
6	&&
7	||

Ex: Equal to operators

```
$x=99; // Integer Value
$y='99'// String value
// Compare $x and $y
if ($x == $y)
    echo 'Same content';
else
    echo 'Different content';
```

Output: Same Content

```
// Compare $x and $y
if ($x === $y)
    echo 'Data type and value
both are same';
else
    echo 'Data type or value
are different';
```

Output: Data type or value are different

Ex: Ternary Operator

- logical test ? value if true : value if false

```
<?php
  function calculate_overtime($shift) {
        $hours = $shift;
         return $hours > 8 ? 10 * 100: 8 * 100;
    }
?>
```

Control Structure

Conditional-if:

- `if - else`

Multiway:

- `if-elseif-else`

```
<?php
if (1)
{
     echo “A”;
}
elseif (0)
{
     echo “B”;

}
else {
     echo “C”;
}
?>
```

Loop

- `while`
- `do-while`
- `for(;;)`
- `foreach`
- `break` – Breaking out of a loop; Terminates the loop statement and transfers execution to the statement immediately following the loop.
- `continue` – End the current iteration, jump to the top of the loop and start next iteration

Loop

```
<?php
$num = 11;
$upper = 10;
$lower = 1;

do {
  echo ($num*$lower)”\n”;
  $lower++;
} while ($lower <= $upper)
?>
```

```
<?php
$num = 11;
$upper = 10;
$lower = 1;

while ($lower <= $upper)
{
  echo ($num*$lower)”\n”;
  $lower++;
}
?>
```

Casting

- Casting refers to the conversion of a particular variable's data type to another data type
- Operator defines how operands will behave unlike other programming languages, where operands define how an operator behaves
- Implicit typecasting: No need for user to mention a specific type conversion
- Explicit typecasting: user cast a variable to a particular data type according to requirement

Example:

```
$string = "7";
$num = (int) $string;
echo $num;
echo gettype($num)
```

Casting Examples

- PHP converts expression values silently and aggressively.
  - division forces operands to be floating point
  - concatenation converts operands to strings

```
$a = 56; $b = 12;
$c = $a / $b;
echo "C: $c\n";
$d = "100" + 36.25 + TRUE;
echo "D: ". $d . "\n";
echo "D2: ". (string) $d . "\n";
```

```
$e = (int) 9.9 - 1;
echo "E: $e\n";
$f = "sam" . 25;
echo "F: $f\n";
```

Output Statements

- `printf()` and `sprint()` functions are supported same as that of C
- echo vs. `printf()`
  - `echo “Total amount of order is $total”;`
  - `printf(“Total amount of order is %s”,$total);`

Type	Meaning
b	Interpret as an integer and print as a binary number.
c	Interpret as an integer and print as a character.
d	Interpret as an integer and print as a decimal number.
f	Interpret as a double and print as a floating point number.
o	Interpret as an integer and print as an octal number.
s	Interpret as a string and print as a string.
x	Interpret as an integer and print as a hexadecimal number with lowercase letters for the digits a-f.
X	Interpret as an integer and print as a hexadecimal number with uppercase letters for the digits A-F.

File Operations – fopen(), fclose(), feof()

- Opening a file

```
$fp = fopen("myfile.txt", "r");
```

- `fopen` if unable to open the file, it returns 0.

```
if ( !($fp = fopen("myfile.txt", "r") ) )
    exit("Unable to open the input file.");
```

- Closing a File: `fclose($fp);`
- The `feof` function is used to determine the end of file

```
if ( feof($fp) )
    echo "End of file<br>";
```

File Modes

Mode	Description
r	Read Only mode, with the file pointer at the start of the file.
r+	Read/Write mode, with the file pointer at the start of the file.
w	Write Only mode. Truncates the file (effectively overwriting it). If the file doesn't exist, fopen will attempt to create the file.
w+	Read/Write mode. Truncates the file (effectively overwriting it). If the file doesn't exist, fopen will attempt to create the file.
a	Append mode, with the file pointer at the end of the file. If the file doesn't exist, fopen will attempt to create the file.
a+	Read/Append, with the file pointer at the end of the file. If the file doesn't exist, fopen will attempt to create the file.

Reading From a File

- `fgets($fp,length);` // Read one line at a time, default length value 1024
- `fgetc($fp);` // Read one character at time
- `fscanf($fp, "%s %s %d", $name, $title, $age);`
  - Read one word at a time
- `fread()`: read an entire file
- `filesize()`: returns file size

Writing to a File

- `fwrite(file, txt, length)`
  - `file`: file pointer
  - `txt`: string to be written
  - `length`: maximum number of bytes to be written (optional)
  - `returns` the number of bytes written

Summary

- PHP basics
  - All PHP codes must be enclosed within <?php PHP Code ?>
  - Variables : start with $ followed by character or _
  - Datatypes : Boolean, Integer, Floating point and Strings
  - Operators
    - Mathematical/General operators : *. /, %, +, -, =, .,++,--, +=, -= ….
    - Comparison operators : ==, !=, <>, ===, !==, <=, <, >, >=, ?:
    - Logical operators : &&, !!, xor, !
  - Control structures, Loops, Output statements
  - Casting : Implicit and Explicit
  - File Operations

PHP Arrays

- Array types – Numeric and Associative
- Printing arrays
- Looping through an array
- Array functions

PHP Arrays: Numerically indexed arrays (1)

- The keyword `array` is a constructor
- Integer indices starting from 0

PHP Arrays: Numerically indexed arrays (2)

- The keyword `array` is a constructor
- Integer indices starting from 0
- Creating arrays
  - Start from an empty array and add items

```
<?php
   $words=array();
   $words[]="Hello";
   $words[]="World";
   echo $words[1];
?>
Output: World
```

PHP Arrays: Numerically indexed arrays (3)

- The keyword `array` is a constructor
- Integer indices starting from 0
- Creating arrays
  - Copy one array to the other using the “=“

```
<?php
  $words=array(“hello”,”world”);
  $greet=$words;
?>
```

PHP Arrays: Numerically indexed arrays (4)

- The keyword `array` is a constructor
- Integer indices starting from 0
- Creating arrays
  - Use the range() function to auto populate an array

```
<?php
  $numbers = range(1,10);
?>
$numbers[0] is assigned 1,
$numbers[1] is assigned 2,
and so on
```

PHP Arrays: Associative Arrays

- In associative arrays, we associate a key (or index) with each value

  key=>value

```
<?php
    $DA214=array("instructor"=>"Debanga","course
name"=>"DBMS");
    echo $DA214['course name'] ."\t". $DA214['instructor'],
"\n";
?>
```

Exercises: What are the outputs?

1. `echo “99” + 1;`
2. `echo “100” . 2;`
3.
```
$hours = 8;
echo $hours > 6 ? 20 * 100: 400;
```
4.
```
$num = 11;
$upper = 10;
$lower = 1;

do {
 echo ($num*$lower) . "\n";
 $lower++;
} while ($lower <= $upper);
```