# Course Databases

## Code: KEN2110

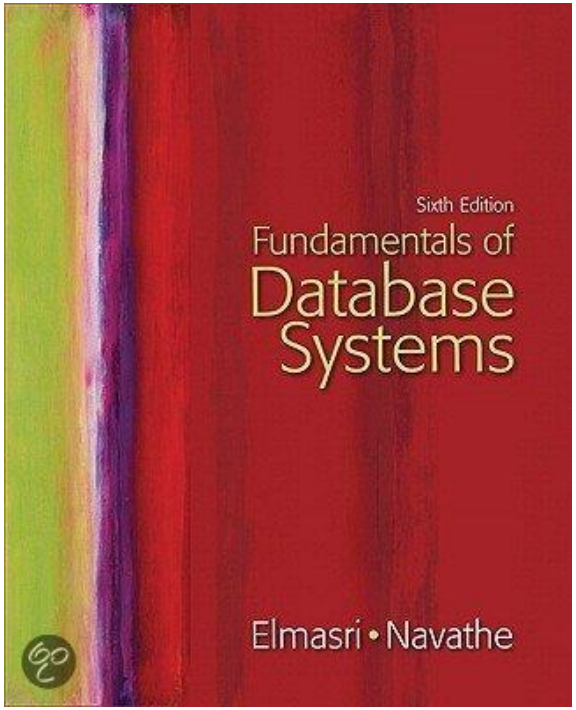

Universiteit Maastricht

# Teacher

- Mena Habib
  - Assistant Professor at DKE.
  - PhD 2014 (University of Twente)
  - Interests: Natural Language Processing, Information Extraction, Social Media Analytics, Web Data Science.

# Book

- Fundamentals Of Database Systems
  - Ramez Elmasri, Shamkant Navathe



# Agreement

- You Can:
  - Come late!
  - Leave early!
  - Send me emails
  - Eat in the lecture
  - Interrupt the lecture by a question (I may ask you to postpone it a bit)
  - Drop by my office (St. Servaasklooster 39, Room 1.001)

# Agreement

- You Can NOT:
  - Make noise!
  - Expect me to reply your emails immediately.

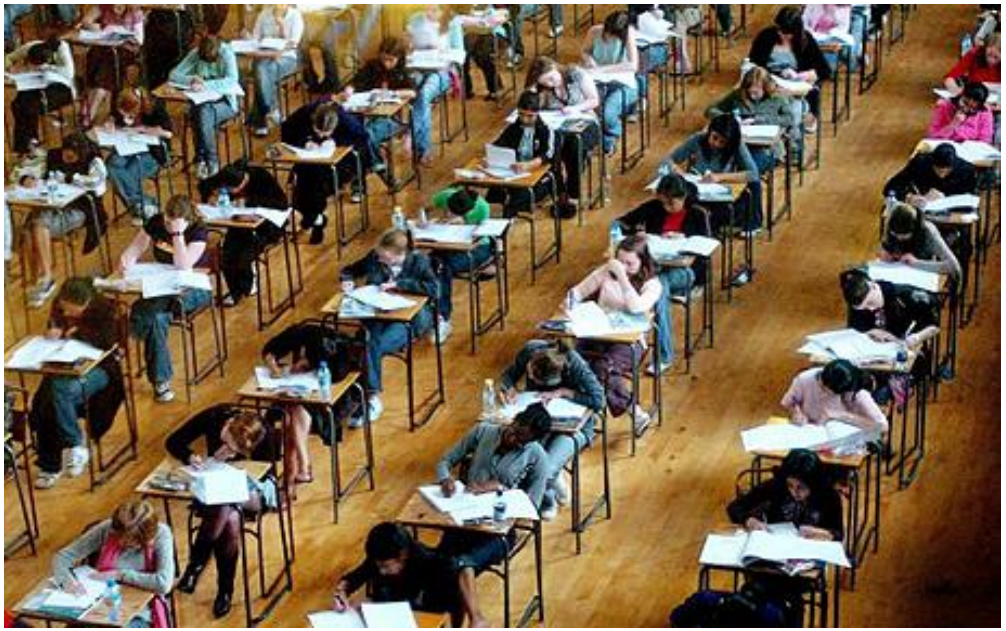
# Agreement

- You Can NOT:
  - Make noise!
  - Expect me to reply your emails immediately.

# Schedule

- Lecture on Monday
  - Theory
- Lab & Exercise on Tuesday
  - Solve assignments

# Grades



- Assignments
  - Practical Assignments
  - Project
  - 25%
  - Groups of 4-5 (Should be formed on BB b4 weekend)
- Final Exam
  - Closed book
  - 75%

# Who already knows something about Databases???

# What is the course about?

- Relationships!

# Course Contents

- Week 1: Introduction
- Week 2: Entity-Relationship (ER) Model
- Week 3: Mapping ER to Relational Model
- Week 4: Relational Data Model
- Week 5: Basic SQL
- Week 6: Advanced SQL
- Week 7: Database Normalization
- Week 8: Exam

# Chapter 1:
# Databases and Database Users

# Introduction

- **Database**
  - Collection of related data that represents some aspect of the real world (**Miniworld**)
  - Logically coherent collection of data with inherent meaning
  - Built for a specific purpose and users
  - Essential component of life in modern society
    - Bank, flight & hotel reservations, online shopping .. etc.

# Introduction

- Changes in the real world must be reflected in the database as soon as possible
- Example:
  - a customer buys a camera from ebay
  - events may happen (for example, an employee has a baby) that cause the information in the database to change

# Types of Databases

- **Traditional database applications**
  - Store textual or numeric information (ex: Students' DB)
- **Multimedia databases**
  - Store images, audio clips, and video streams digitally (ex: Sound Cloud, Youtube)
- **Geographic information systems (GIS)**
  - Store and analyze maps, weather data, and satellite images (ex: Google maps)

# Database size

- Example of a large commercial database
  - Amazon.com:

    >20 million books, CDs, videos, games, electronics, etc.

    >2 terabytes

    >15 million users a day

    Continuous transactions

    >100 people working on Amazon database
  - Facebook:

    http://blog.wishpond.com/post/115675435109/40-up-to-date-facebook-facts-and-stats

# Database management system (DBMS)

- Collection of programs
- Enables users to create and maintain a database
- facilitates the processes of *defining, constructing, manipulating,* and *sharing* databases among various users and applications
- Ex: MySQL, PostgreSQL, Microsoft SQL Server

# Definitions

- **Defining** a database
  - Specify the data types, structures, and constraints of the data to be stored
- **Meta-data**
  - Database definition or descriptive information

# Definitions

- **Constructing** a database
  - The process of storing the data on some storage medium that is controlled by the DBMS
- **Manipulating** a database
  - Query and update the database miniworld
  - Generate reports

# Definitions

- **Sharing** a database
  - Allow multiple users and programs to access the database simultaneously
- **Application program**
  - Accesses database by sending queries to DBMS
- **Query**
  - Causes some data to be retrieved

# Definitions

- **Transaction**
  - May cause some data to be read and some data to be written into the database
  - Ex: Withdraw money from ATM.
- **Protection** includes:
  - System protection (against hardware or software malfunction)
  - Security protection (against unauthorized or malicious access)
- **Maintain** the database system
  - Allow the system to evolve as requirements change over time

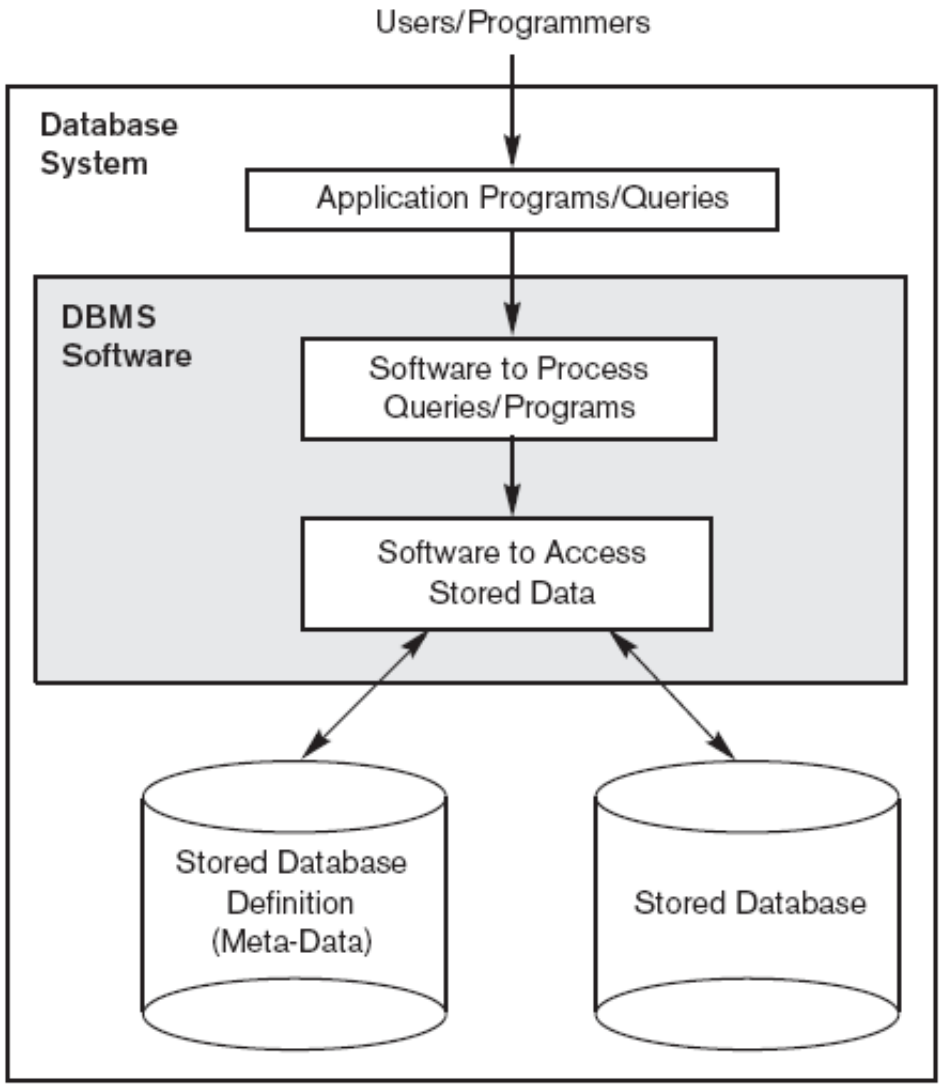



**Figure 1.1**
A simplified database system environment.

# An Example

- UNIVERSITY database
  - Information concerning students, courses, and grades in a university environment
- **Data records**
  - STUDENT
  - COURSE
  - SECTION
  - GRADE_REPORT
  - PREREQUISITE

**STUDENT**

| Name | Student_number | Class | Major |
|---|---|---|---|
| Smith | 17 | 1 | CS |
| Brown | 8 | 2 | CS |

**COURSE**

| Course_name | Course_number | Credit_hours | Department |
|---|---|---|---|
| Intro to Computer Science | CS1310 | 4 | CS |
| Data Structures | CS3320 | 4 | CS |
| Discrete Mathematics | MATH2410 | 3 | MATH |
| Database | CS3380 | 3 | CS |

**SECTION**

| Section_identifier | Course_number | Semester | Year | Instructor |
|---|---|---|---|---|
| 85 | MATH2410 | Fall | 07 | King |
| 92 | CS1310 | Fall | 07 | Anderson |
| 102 | CS3320 | Spring | 08 | Knuth |
| 112 | MATH2410 | Fall | 08 | Chang |
| 119 | CS1310 | Fall | 08 | Anderson |
| 135 | CS3380 | Fall | 08 | Stone |

**GRADE_REPORT**

| Student_number | Section_identifier | Grade |
|---|---|---|
| 17 | 112 | B |
| 17 | 119 | C |
| 8 | 85 | A |
| 8 | 92 | A |
| 8 | 102 | B |
| 8 | 135 | A |

**PREREQUISITE**

| Course_number | Prerequisite_number |
|---|---|
| CS3380 | CS3320 |
| CS3380 | MATH2410 |
| CS3320 | CS1310 |

**Figure 1.2**
A database that stores student and course information.

# An Example

- Specify (define) structure of records of each file (table) by specifying **data type** for each **data element**
  - String of alphabetic characters
  - Integer
  - Etc.

# An Example

- Construct UNIVERSITY database
  - Store data to represent each student, course, section, grade report, and prerequisite as a record in appropriate file
- Relationships among the records
- Manipulation involves querying and updating

# An Example

- Examples of queries:
  - Retrieve the transcript (a list of all courses and grades of ‘Smith’)
  - List the names of students who took the section of the ‘Database’ course offered in fall 2008 and their grades in that section
  - List the prerequisites of the ‘Database’ course

# An Example

- Examples of updates:
  - Change the class of ‘Smith’ to class 2
  - Create a new section for the ‘Database’ course for this semester
  - Enter a grade of ‘A’ for ‘Smith’ in the ‘Database’ section of last semester

# Characteristics of the Database Approach

- Traditional **file processing**
  - Each user defines and implements the files needed for a specific software application
- **Database** approach
  - Single repository maintains data that is defined once and then accessed by various users

# Characteristics of the Database Approach

- Main characteristics of database approach
  - Self-describing nature of a database system
  - Insulation between programs and data, and data abstraction
  - Support of multiple views of the data
  - Sharing of data and multiuser transaction processing
  - Security and authorization subsystem which creates accounts

# Advantages of Using the DBMS Approach

- Controlling redundancy (entering the same data multiple times)
  - **Data normalization**

**Figure 1.6**
Redundant storage of Student_name and Course_name in GRADE_REPORT. (a) Consistent data. (b) Inconsistent record.

(a)

**GRADE_REPORT**

| Student_number | Student_name | Section_identifier | Course_number | Grade |
|---|---|---|---|---|
| 17 | Smith | 112 | MATH2410 | B |
| 17 | Smith | 119 | CS1310 | C |
| 8 | Brown | 85 | MATH2410 | A |
| 8 | Brown | 92 | CS1310 | A |
| 8 | Brown | 102 | CS3320 | B |
| 8 | Brown | 135 | CS3380 | A |

(b)

**GRADE_REPORT**

| Student_number | Student_name | Section_identifier | Course_number | Grade |
|---|---|---|---|---|
| 17 | Brown | 112 | MATH2410 | B |

# Advantages of Using the DBMS Approach

- Providing storage structures and search techniques for efficient query processing
  - **Indexes, Query processing and optimization**
  - **Buffering** and **caching**

# Advantages of Using the DBMS Approach

- Consistency
  - Transactions take the database from one consistent (valid) state into another
  - Kinds of consistency, i.e. not all database states are allowable:
    - Internal consistency, for example(Referential integrity)
    - Enterprise rules

# Advantages of Using the DBMS Approach

- Atomicity
  - Transactions are atomic – they don't have parts (conceptually)
  - All or nothing! can't be executed partially.

# Advantages of Using the DBMS Approach

- Isolation
  - The effects of a transaction are not visible to other transactions until it has completed
  - From outside the transaction has either happened or not
  - Even though transactions execute concurrently, it appears to each transaction T, that others executed either before T or after T, but not both.
  - Uses locks

# Advantages of Using the DBMS Approach

- Durability
  - Once a transaction has completed, its changes are made permanent
  - Even if the system crashes, the effects of a transaction must remain in place
  - Uses backups and log files

# Database Meta-data

- Database system contains complete definition of structure and constraints
- This information is stored in the catalog and is called **Meta-data**
  - Describes structure of the database
- Database catalog used by:
  - DBMS software
  - Database users who need information about database structure

# Self-Describing Nature of a Database System

- Whenever a request is made (e.g. access Name of a STUDENT), the DBMS software refers to the catalog to determine the structure of the STUDENT file (table) and the position and size of the Name data item within a STUDENT record.

# Self-Describing Nature of a Database System

**RELATIONS**

| Relation_name | No_of_columns |
|---|---|
| STUDENT | 4 |
| COURSE | 4 |
| SECTION | 5 |
| GRADE_REPORT | 3 |
| PREREQUISITE | 2 |

**COLUMNS**

| Column_name | Data_type | Belongs_to_relation |
|---|---|---|
| Name | Character (30) | STUDENT |
| Student_number | Character (4) | STUDENT |
| Class | Integer (1) | STUDENT |
| Major | Major_type | STUDENT |
| Course_name | Character (10) | COURSE |
| Course_number | XXXXNNNN | COURSE |
| .... | .... | ..... |
| .... | .... | ..... |
| .... | .... | ..... |
| Prerequisite_number | XXXXNNNN | PREREQUISITE |

**Figure 1.3**
An example of a database catalog for the database in Figure 1.2.

# Support of Multiple Views of the Data

- **View**
  - Subset of the database
  - Contains **virtual data** derived from the database files but is not explicitly stored

# Support of Multiple Views of the Data

(a)

**TRANSCRIPT**

| Student_name | Student_transcript | | | | |
|---|---|---|---|---|---|
| | Course_number | Grade | Semester | Year | Section_id |
| Smith | CS1310 | C | Fall | 08 | 119 |
| | MATH2410 | B | Fall | 08 | 112 |
| Brown | MATH2410 | A | Fall | 07 | 85 |
| | CS1310 | A | Fall | 07 | 92 |
| | CS3320 | B | Spring | 08 | 102 |
| | CS3380 | A | Fall | 08 | 135 |

(b)

**COURSE_PREREQUISITES**

| Course_name | Course_number | Prerequisites |
|---|---|---|
| Database | CS3380 | CS3320 |
| | | MATH2410 |
| Data Structures | CS3320 | CS1310 |

**Figure 1.5**
Two views derived from the database in Figure 1.2. (a) The TRANSCRIPT view. (b) The COURSE_PREREQUISITES view.

# Actors on the Scene

- The people whose jobs involve the day-to-day use of a large database are the *actors on the scene*.
- For a small personal database, one person typically defines, constructs, and manipulates the database, and there is no sharing.
- In large organizations, many people are involved in the design, use, and maintenance of a large database with hundreds of users.

# Actors on the Scene

- **Database administrators (DBA)** are responsible for:
  - Authorizing access to the database
  - Coordinating and monitoring its use
  - Acquiring software and hardware resources
- **System analysts**
  - Determine requirements of end users

# Actors on the Scene

- **Database designers** are responsible for:
  - Identifying the data to be stored
  - Choosing appropriate structures to represent and store this data driven by user requirements
  - Designing views based on user requirements
- **Application programmers**
  - Implement these specifications as programs
- **End users**
  - People whose jobs require access to the database
  - Queries, updates, report generating

# Workers behind the Scene

- **DBMS system designers and implementers**
  - Design and implement the DBMS modules and interfaces as a software package
- **Tool developers**
  - Design and implement **tools** (usually, independent, optional packages)
- **Operators and maintenance personnel**
  - Responsible for running and maintenance of hardware and software environment for database system

# Chapter 2:
# Database System Concepts and Architecture

# Data Models, Schemas, and Instances

- **Data model**
  - Collection of concepts that describe the structure of a database
  - Provides means to achieve data abstraction
  - include a set of basic operations for retrievals and updates on the database

# Categories of Data Models

- **High-level** or **conceptual data models**
  - Close to the way many users perceive data
- **Representational data models**
  - Easily understood by end users
  - Also similar to how data organized in computer storage
  - Hiding details but easy to implement on a computer system
- **Low-level** or **physical data models**
  - Describe the details of how data is stored on computer storage media

# Categories of Data Models

## **High-level** or **conceptual data model**

- **Entity**
  - Represents a real-world object or concept
- **Attribute**
  - Represents some property of interest
  - Further describes an entity
- **Relationship** among two or more entities
  - Represents an association among the entities
  - **Entity-Relationship Model (ER)**

# Categories of Data Models

## Representational model

- **Relational Data Model (RD)**
  - Is the representational model used most frequently in traditional commercial DBMSs

# Categories of Data Models

## Physical data models

- Describe how data is stored as files in the computer
- **Index**
  - Structure that makes the search for particular database records efficient
  - Allows direct access to data using an index term or a keyword

# DBMS Languages

- In current DBMSs, the different types of languages are usually *not considered distinct languages*;
- Rather, a comprehensive integrated language (Ex: SQL) is used that includes constructs for conceptual schema definition, view definition, and data manipulation.
- **Data definition language (DDL)**
  - Defines both conceptual and internal schemas
  - Create tables, constraints .. Etc.
- **Data manipulation language (DML)**
  - Allows retrieval, insertion, deletion, modification
  - Select, Insert, Update, Delete data.

# DBMS Languages

- **View Definition Language (VDL),**
  - is used to specify user views
- **Storage Definition Language (SDL),**
  - is used to specify the internal (physical) schema

# DBMS Languages

- Typical example of a comprehensive database language:
  - **SQL:** it is a relational database language, which represents a combination of DDL, VDL, and DML, as well as statements for constraint specification, schema evolution, and other features.
  - The SDL (Storage Definition Language) was a component in early versions of SQL but has been removed from the language to keep it at the conceptual and external levels only.

# Database System Utilities

- In addition to possessing the software modules just described, most DBMSs have **database utilities** that help the DBA manage the database system. Common utilities have the following types of functions
- **Loading**
  - Load existing data files and use conversion tools
- **Backup**
  - Creates a backup copy of the database

# Database System Utilities

- **Performance monitoring**
  - Monitors database usage and provides statistics to the DBA
- **Other utilities**
  - sorting files, handling data compression, monitoring access, interfacing with the network, etc.

# Centralized and Client/Server Architectures for DBMSs

- **Centralized DBMSs Architecture**
  - All DBMS functionality, application program execution, and user interface processing carried out on one machine
  - Gradually, DBMS systems started to exploit the available processing power at the user side, which led to client/server DBMS architectures.

# Basic Client/Server Architectures

- **Client machines**
  - Provide user with:
    - Appropriate interfaces to utilize these servers
    - Local processing power to run local applications
- **Server**
  - System containing both hardware and software
  - Provides services to the client machines
    - Such as file access, printing, archiving, or database access

# Two-Tier Client/Server Architectures for RDBMSs

- Server handles
  - Query and transaction functionality related to SQL processing
- Client handles
  - User interface programs and application programs

# Two-Tier Client/Server Architectures

- Open Database Connectivity (ODBC)
  - Provides application programming interface (API)
  - Allows client-side programs to call the DBMS
- JDBC
  - Allows Java client programs to access one or more DBMSs through a standard interface

# Three-Tier and n-Tier Architectures for Web Applications

Many Web applications use an architecture called the **three-tier architecture**, which adds an intermediate layer between the client and the database server

- **Application server** or **Web server**
  - Adds intermediate layer between client and the database server
  - Runs application programs and stores business rules
  - Adds extra security before sending requests to the database server

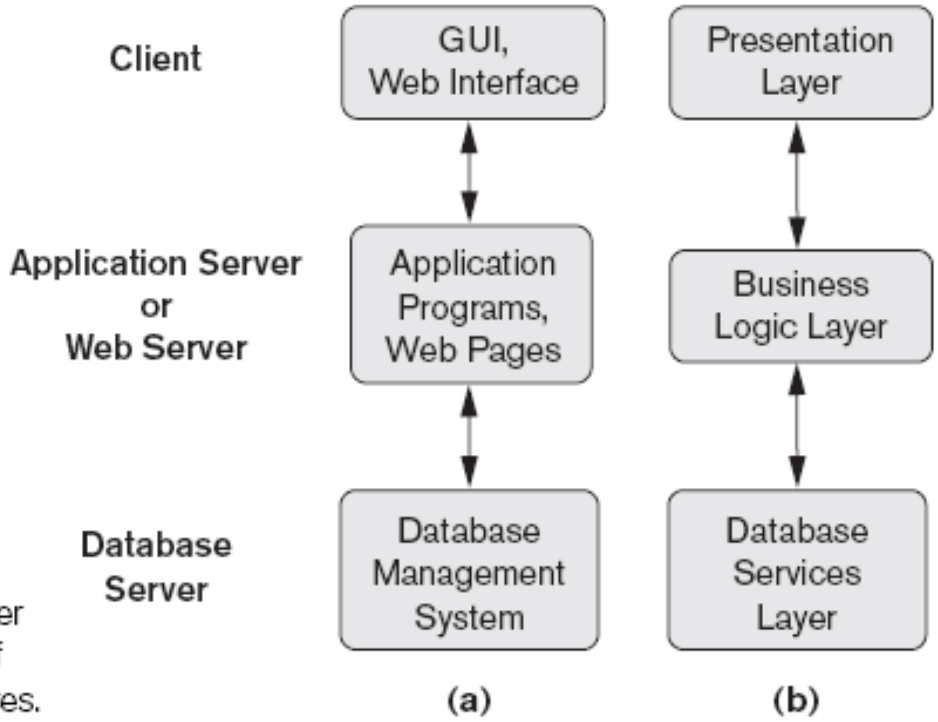



**Figure 2.7**
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

# NO SQL

- **N**ot **O**nly **SQL**
- The Benefits of NoSQL:
  - Geographically distributed architecture instead of expensive, monolithic architecture
  - Large volumes of rapidly changing structured, semi-structured, and unstructured data

# NoSQL Database Types

- **Graph stores** are used to store information about networks of data, such as social connections.
- **Document databases** pair each key with a complex data structure known as a document.
- **Key-value** stores are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value
- **Wide-column stores** such as HBase are optimized for queries over large datasets

# Document Store

- The central concept is the notion of a "document“ which corresponds to a row in RDBMS.
- A document comes in some standard formats like JSON
- Documents are addressed in the database via a unique key that represents that document.
- The database offers an API or query language that retrieves documents based on their contents.
- Documents are schema free, i.e., different documents can have structures and schema that differ from one another. (An RDBMS requires that each row contain the same columns.)

# JSON

```
{
    _id: ObjectId("51156a1e056d6f966f268f81"),
    type: "Article",
    author: "Derick Rethans",
    title: "Introduction to Document Databases with MongoDB",
    date: ISODate("2013-04-24T16:26:31.911Z"),
    body: "This arti…"
},
{
    _id: ObjectId("51156a1e056d6f966f268f82"),
    type: "Book",
    author: "Derick Rethans",
    title: "php|architect's Guide to Date and Time Programming with PHP",
    isbn: "978-0-9738621-5-7"
}
```

# Summary

- What is a DB?
- Why to use a DB?
- Who are the key players in a DB system?
- What are the categories of data models?
- What are the common DBMS languages?
- What are the common DB system architectures?
- What is the difference between RDBMS and NoSQL?