

# CompSci 516

# Data Intensive Computing Systems

## Lecture 1

## Introduction and Data Models

Instructor: Sudeepa Roy

# Course Website

- <http://www.cs.duke.edu/courses/fall16/compsci516/>
- Please check frequently for updates

# Instructor

- Sudeepa Roy
  - [sudeepa@cs.duke.edu](mailto:sudeepa@cs.duke.edu)
  - <https://users.cs.duke.edu/~sudeepa/>
  - office hour: Mondays 1:30-2:30 pm, LSRC D325
- About myself
  - Assistant Professor in CS
  - PhD: UPenn, Postdoc: Univ. of Washington
  - Joined Duke CS in Fall 2015
  - Research interests:
    - Databases (theory and applications)
    - Data Analysis, causality, explaining answers
    - Uncertain data, data provenance, crowd sourcing

# TA

- Junghoon Kang
  - [jungkang@cs.duke.edu](mailto:jungkang@cs.duke.edu)
  - office hour: Thursdays 1–2 pm, North N303B
    - No office hour this week and on 09/05, Mon (Memorial day)
    - Additional office hour by Jung next week Tuesday 1 pm – 2 pm, North N303B

# Logistics

- Homework submission: Sakai
  - All enrolled students are already there
- Discussion forum: Piazza
  - All enrolled students are already there
  - Send me an email if you have not received a welcome email from Piazza
- Lecture slides will be uploaded before the class
  - but will be updated after the class

# Grading

- Three Homework: 30%
- Project: 20%
- Midterm: 20%
- Final: 30%

# Grading Strategy

- Relative grading
  - Topper of the class gets A+ irrespective of the number, and all and only “above expectation” performances get A+
  - No lowest grade or fixed distribution – so everyone can get very good grades by working hard!
  - The actual grade distribution at the end will depend on the performance of the entire class on all the components
  - If you are a borderline case for two grades, your **class participation** and effort throughout the semester will put you in the higher grade
    - Actively participate in the class!
    - Ask questions in class and on piazza
    - Answer each other’s questions on piazza
    - Send (anonymous or not) feedback, suggestions, or concerns on Piazza

# Homework

- Due in ~3 weeks after they are posted/previous hw is due
  - 2 weeks should be enough
  - Start early
- No late days – contact the instructor if you have a *\*valid\** reason to be late
  - Another exam, project, hw is not a valid reason – we will always be fair to all
  - Computer crash/sudden interview trips/medical issues (following official procedures) may count as valid reasons
  - No guarantee that your request will be granted – again, start early!
- To be done individually



# Homework Overview

- You will learn how to use traditional and new database systems in the homework
  - Have to learn them mostly on your own following tutorials available online and with some help from the TA
- HW1 and HW2 already on sakai! Have fun!
  - Work on them at your own pace
- HW1: Traditional DBMS
  - SQL and Postgres
  - Due on 09/16 (Fri)
- HW2: Distributed data processing
  - Spark and AWS
  - Will receive instructions for AWS (second part of the hw)
  - Due on 10/12 (Wed)
- HW3: NOSQL
  - e.g. MongoDB or DynamoDB
  - Will be posted later (after Transactions)

# Exams

- Midterm – Oct 5 (Wed)
- Final – TBD (by univ schedule)
  
- In class
- Closed book, closed notes, no electronic devices
- Total weight: 20 + 30 % = 50 %
- Exams will test your understanding of the material

# Projects

- 20% weight
- In groups of at most 3
  - You can look for group members through Piazza by announcing your general area of interest or if you have a problem in mind
  - Each group member should do approx. equal work
- Work done should be (at least) equivalent to two HWs, i.e.
  - the work for one hw \* 2 \* #group members
- Take it very seriously!
  - show your creativity and researcher-side
- There is an ACM SIGMOD Student Research Competition
  - With publication and \$ as reward, and winners go to ACM-wide competition
  - Separate categories for undergrads and grads
  - Deadline: November 18, 2016 (just the abstract)
  - <http://sigmod2017.org/student-research-competition/>

# Project Topics

- Anything related to “Data”
  - Data management / processing / cleaning
  - Data visualization
  - Data exploration or analysis
  - Applications of data (to any field)
  - Theoretical findings with data
  - New tool for data analysis
- Choose a project according to your research interest
- You can check out major database conferences for ideas, e.g.
  - **Demonstrations** (build a prototype solving a problem or improving UI)
    - SIGMOD’16 : [http://sigmod2016.org/sigmod\\_demo\\_list.shtml](http://sigmod2016.org/sigmod_demo_list.shtml)
    - VLDB’16: <http://vldb2016.persistent.com/demonstrations.php>
  - **Research papers** (solve a problem, do experiments with data)
    - SIGMOD’16: [http://sigmod2016.org/sigmod\\_research\\_list.shtml](http://sigmod2016.org/sigmod_research_list.shtml)
    - VLDB’16: <http://www.vldb.org/pvldb/vol9.html>
  - You can check out previous years too, and conferences from your own research area

# Project Deliverables

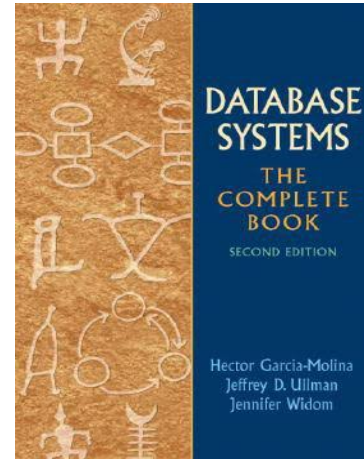
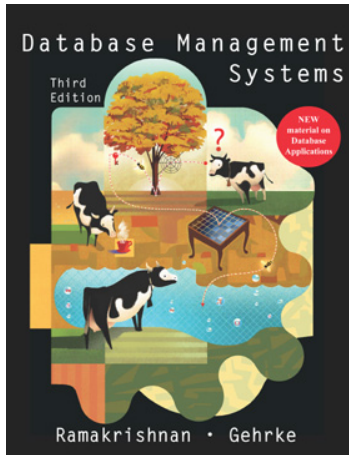
1. Project proposal (due: 9/21, 1-3 pages)
    - problem selection is part of the project
    - 3 weeks from now
    - but start asap, look for problems, do related work study, find an interesting question, do a few iterations with me (try to meet me once), all by the deadline
  2. Midterm progress report (due: 10/21, 3-5 pages)
  3. Final project report (due: 11/28, 4-8 pages)
  4. A final ~10 mins project presentation and/or demonstration (in the last 1-2 classes)
- The same document will be updated
    - Template and high level ideas will be posted on sakai soon

# Project Evaluation Criteria

Scale of 100:

1. Well-motivated? 10
2. Novel? 10
3. Comprehensive related work survey? 10
4. Quality of writing? 10
  - should reflect all other factors too except class presentation
5. Class presentation/demo? 15
  - should reflect all other factors too except writing
6. Technical contributions? 45
  - Problem formulation / Algorithms / Experiments / Theory / System / User interface / Efficiency / Usability / Dataset exploration etc.

# Reading Material



- Will mostly follow the “cowbook” by Ramakrishnan-Gehrke
  - The chapter numbers will be posted
- You do not have to buy the books, but it will be good to consult them from time to time
- You should be prepared to do quite a bit of reading from various books and papers

# What is this course about?

- This is a graduate-level database course in CS
- We will cover principles, internals, and applications of database systems in depth
- We will also have an introduction to a few advanced research topics in databases (later in the course)



# A Quick Survey

- Have you taken an undergrad database course earlier
  - CS 316/equivalent?
- Are you familiar with
  - SQL?
  - RA? ( $\sigma$ ,  $\Pi$ ,  $\times$ ,  $\bowtie$ ,  $\rho$ ,  $\cup$ ,  $\cap$ ,  $-$ )
  - Keys, foreign keys?
  - Index in databases?
  - Logic:  $\wedge$ ,  $\vee$ ,  $\forall$ ,  $\exists$ ,  $\neg$ ,  $\in$ ,  $\Rightarrow$
  - Transactions?
  - Map-reduce/Spark?
- Have you ever worked with a dataset?
  - relational database, text, csv, XML
- Have you ever used a database system?
  - PostGRES, MySQL, SQL Server, SQL Azure

# What will be covered?

- Database concepts
  - Data Models, SQL, Views, Constraints, RA, Normalization
- Principles and internals of database management systems (DBMS)
  - Indexing, Query Execution-Algorithms-Optimization, Transactions, Parallel and Distributed Query Processing, Map Reduce
- Advanced and research topics in databases
  - e.g. Datalog, NOSQL, Data mining, Data warehouse
  - More will be added in the “TBD” lectures
- We will go fast for some basic topics in databases
  - Data model, SQL, RA

# Background

- You should have some understanding (at the CS undergraduate level)
  - data structure, discrete maths, algorithms
  - databases
  - or have to learn these yourself as necessary
- Need to pickup new coding framework and programming languages on your own
  - and how to process data using them
  - Homework assignments will mostly be self-taught
  - ...with help from the TA
- Will involve some mathematical and analytical reasoning too

# Why should we care about databases?

- We are in a data-driven world
- “Big Data” is supposed to change the mode of operation for almost every single field
  - Science, Technology, Healthcare, Business, Manufacturing, Journalism, Government, Education, ...
- We must know how to collect, store, process, and analyze such data

# Why should we care about databases?

- From “Big Data” wiki:

## Science

“The Large Hadron Collider experiments represent about **150 million sensors delivering data 40 million times per second**. There are nearly 600 million collisions per second. If all sensor data were recorded in LHC, .... this is equivalent to **500 quintillion ( $5 \times 10^{20}$ ) bytes per day**, almost 200 times more than all the other sources combined in the world.”



# Why should we care about databases?

## Technology

- From “Big Data” wiki:
  - eBay.com uses two data warehouses at 7.5 PB ( $\times 10^{12}$ ) and 40PB as well as a 40PB Hadoop cluster for search, consumer recommendations, and merchandising
  - Facebook handles 50 billion photos from its user base
  - As of August 2012, Google was handling roughly 100 billion searches per month



# Why should we care about databases?

- From “Big Data” wiki:
  - **Healthcare**: digitization of patient’s data, prescriptive analytics
  - **Media**: Tailor articles and advertisements that reach targeted people, validate claims
    - “Computational Journalism” project in Duke DB group
  - **Manufacturing**: supply planning
  - **Sports**: improve training, understanding competitors

Healthcare  
Media  
Manufacturing  
Sports  
.....

# Why should we care about databases?

- Simply storing such large datasets in a flat file stops working at some point
  - Need efficient model, storage, and processing
- A DBMS takes care of such issues – the user only has to run queries to process such datasets
  - much simpler than writing low level code



# Today

- DBMS
- Data Models
- [RG] 1.1, 1.3-1.5

# What is a Database?

- A database is a collection of data
  - typically related and describing activities of an organization
- A database may contain information about
  - Entities
    - students, faculty, courses, classroom
  - Relationships between entities
    - students' enrollment, faculty teaching courses, rooms for courses

# Why use a DBMS

- i.e. why not use file system and a programming language?
- Suppose a company has a large collection of data on employees, departments, products, sales etc.
- Requirements:
  - Quickly answer questions on data
    - Note that all the data may not fit in main memory
  - Concurrent access: apply changes consistently
  - Restricted access (e.g. salary)

# Why use a DBMS?

- A DBMS is a piece of software (i.e. a big program written by someone else) that makes these tasks easier
  - Quick access
  - Robust access
  - Safe access
  - Simpler access
- **Next: some nice properties of a DBMS**

# Why use a DBMS?

## 1. Data Independence

- Application programs should not be exposed to the data representation and storage
- DBMS provides an abstract view of the data

## 2. Efficient Data Access

- A DBMS utilizes a variety of sophisticated techniques to store and retrieve data (from disk) efficiently

# Why use a DBMS?

## 3. Data Integrity and Security

- DBMS enforces “integrity constraints” – e.g. check whether total salary is less than the budget
- DBMS enforces “access controls” – whether salary information can be accessed by a particular user

## 4. Data Administration

- Centralized professional data administration by experienced users can manage data access, organize data representation to minimize redundancy, and fine tune the storage

# Why use a DBMS?

## 5. Concurrent Access and Crash Recovery

- DBMS schedules concurrent accesses to the data such that the users think that the data is being accessed by only one user at a time
- DBMS protects data from system failures

## 6. Reduced Application Development Time

- Supports many functions that are common to a number of applications accessing data
- Provides high-level interface
- Facilitates quick and robust application development

# When NOT to use a DBMS?

- DBMS is optimized for certain kind of workloads and manipulations
- There may be applications with tight real-time constraints or a few well-defined critical operations
- Abstract view of the data provided by DBMS may not suffice
- To run complex, statistical/ML analytics on large datasets



# Data Model

- Applications need to model some real world units
- Entities:
  - Students, Departments, Courses, Faculty, Organization, Employee, ...
- Relationships:
  - Course enrollments by students, Product sales by an organization
- A data model is a collection of high-level data description constructs that hide many low-level storage details

# Data Model

Can Specify:

## 1. Structure of the data

- like arrays or structs in a programming language
- but at a higher level (conceptual model)

## 2. Operations on the data

- unlike a programming language, not any operation can be performed
- allow limited sets of queries and modifications
- a strength, not a weakness!

## 3. Constraints on the data

- what the data can be
- e.g. a movie has exactly one title

# Important Data Models

- Structured Data
- Semi-structured Data
- Unstructured Data

What are these?

# Important Data Models

- **Structured Data**
  - All elements have a fixed format
  - **Relational Model** (table)
- **Semi-structured Data**
  - Some structure but not fixed
  - Hierarchically nested tagged-elements in tree structure
  - XML
- **Unstructured Data**
  - No structure
  - text, image, audio, video

# Relational Data Model

- Proposed by Edward (Ted) Codd in 1970
  - won Turing award for it!
- Motivation:
  - Simplicity
  - Better logical and physical data independence

# Relational Data Model

Students				
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith1@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- The data description construct is a Relation
  - Represented as a “table”
  - Basically a “set” of records (**set semantic**)
    - order does not matter
    - and all records are distinct
- however, it is true for the relational model, not for standard DBM
  - allow duplicate rows (**bag semantic**)
  - unless restricted by key constraints. **Why?**

# Relational Data Model

Students				
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith1@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- The data description construct is a Relation
  - Represented as a “table”
  - Basically a “set” of records (**set semantic**)
  - order does not matter
  - and all records are distinct
- however, it is true for the relational model, not for standard DBM
  - allow duplicate rows (**bag semantic**)
  - unless restricted by key constraints. **Why?**

# Bag vs. Set

Students				
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith1@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- **Bag:** {1, 1, 2, 2, 3, 2, 1, 5, 6, 1}      **Set:** {1, 2, 3, 5, 6}
- Why “bag semantic” and not “set semantic” in standard DBMSs?
  - Primarily performance reasons
  - Duplicate elimination is expensive (requires sorting)
  - Some operations like “projection”s are much more efficient on bags than sets



# Relational Data Model

Students				
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith1@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Diagram annotations:

- An arrow labeled "Attribute/Column/Field" points to the header row.
- An arrow labeled "Value" points to the cell containing "2.0".
- An arrow labeled "Tuple/Row/Record" points to the row containing "53650".

What is a poorly chosen attribute in this relation?

- Relational database = a set of relations
- A Relation : made up of two parts
  1. Schema
  2. Instance

# Schema and Instance

- One schema can have multiple instances
- Schema:
  - A template for describing an entity/relationship (e.g. students)
  - specifies name of relation + name and type of each columne.g. **Students(sid: string, name: string, login: string, age: integer, gpa: real).**
- Instance:
  - When we fill in actual data values in a schema
  - a table, has rows and columns
  - each row/tuple follows the schema and domain constraints
  - #Rows = cardinality, #fields = degree / arity
  - example below

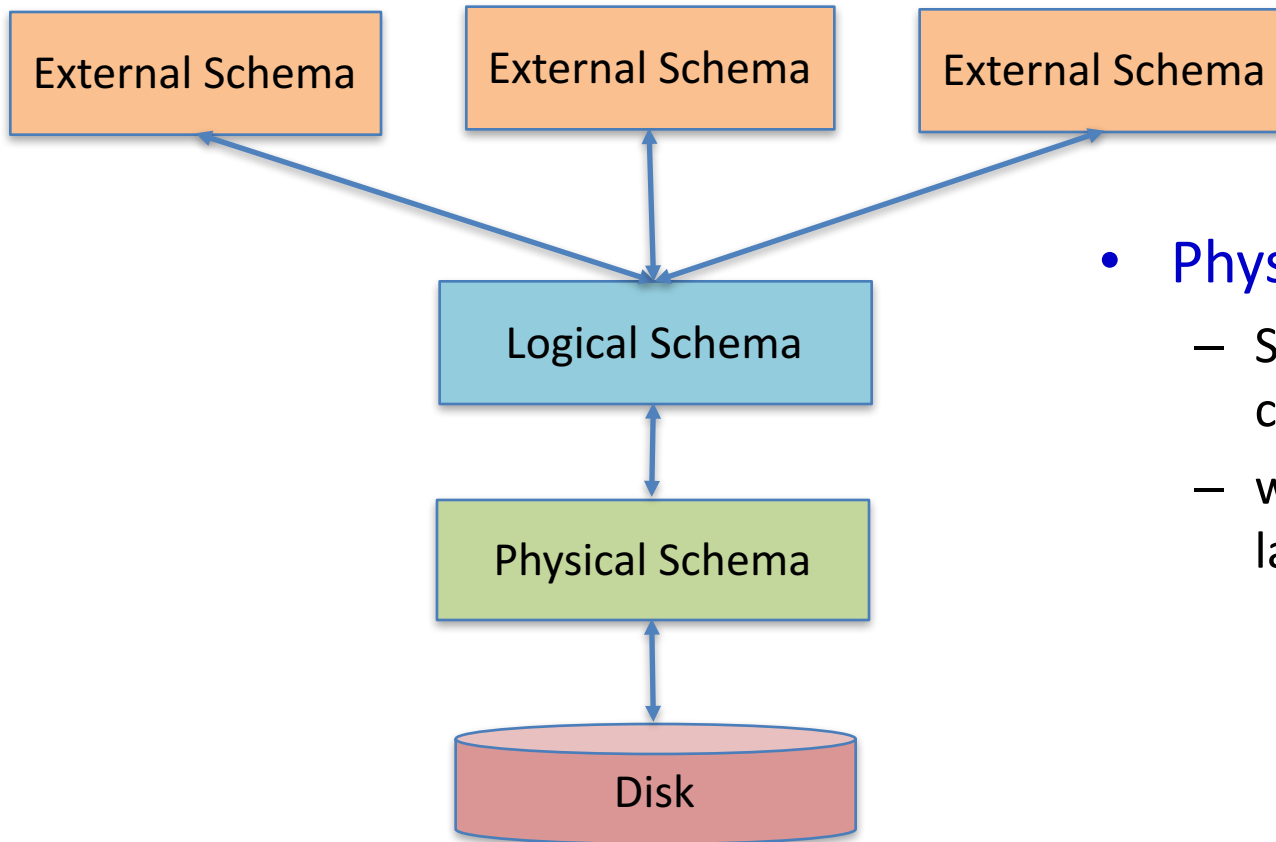
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith1@math	19	3.8

Cardinality = 3, degree = 5

# Relational Database: Definitions

- Relational database = a set of relations
- Relations : made up of 2 parts:
- **Schema**
  - A template for describing an entity/relationship (e.g. students)
  - specifies name of relation + name and type of each column
  - Students(sid: string, name: string, login: string, age: integer, gpa: real).
  - **Instance** : a table, has rows and columns
    - each row/tuple follows the schema and domain constraints
    - #Rows = cardinality, #fields = degree / arity.
- Can think of a relation as a **set of rows or tuples**, i.e., all rows are **distinct**
  - however, it is true for the relational model, all rows distinct not for standard DBMS that allow duplicate rows. **Why?**

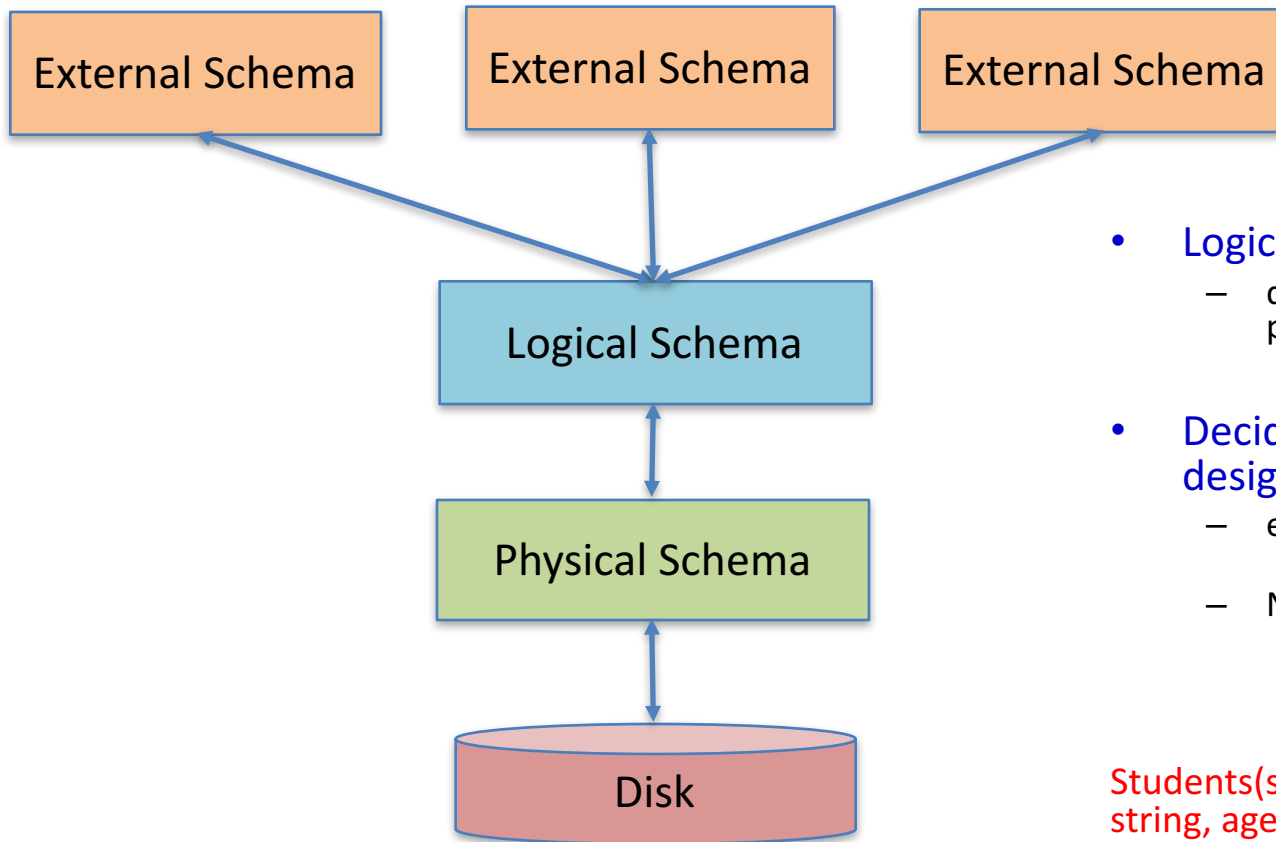
# Levels of Abstractions in a DBMS



- **Physical schema**

- Storage as files, row vs. column store, indexes
- will discuss these in later lectures

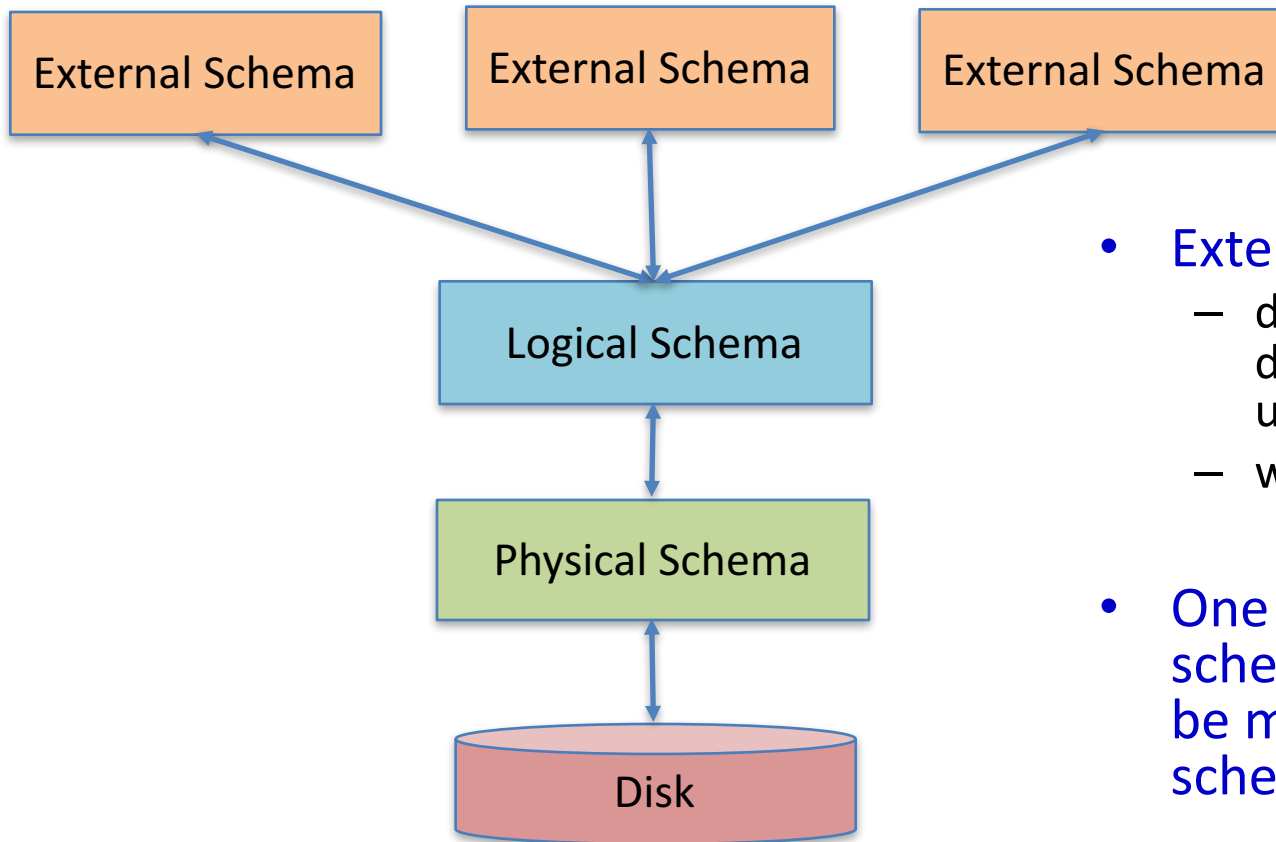
# Levels of Abstractions in a DBMS



- Logical/Conceptual schema
  - describes the stored data in the physical schema
- Decided by conceptual schema design
  - e.g. ER Diagram
    - not covered in this course
  - Normalization
    - will be covered

Students(sid: string, name: string, login: string, age: integer, gpa: real)

# Levels of Abstractions in a DBMS



- External schema
  - different “views” of the database to different users
  - will discuss views later
- One physical and logical schema but there can be multiple external schemas

# Data Independence

- Application programs are insulated from changes in the way the data is structured and stored
- A very important property of a DBMS
- Logical and Physical

# Logical Data Independence

- Users can be shielded from changes in the logical structure of data
- e.g. Students:  
`Students(sid: string, name: string, login: string, age: integer, gpa: real)`
- Divide into two relations  
`Students_public(sid: string, name: string, login: string)`  
`Students_private(sid: string, age: integer, gpa: real)`
- Still a “view” Students can be obtained using the above new relations
  - by “joining” them with sid
- A user who queries this view Students will get the same answer as before



# Physical Data Independence

- The logical/conceptual schema insulates users from changes in physical storage details
  - how the data is stored on disk
  - the file structure
  - the choice of indexes
- The application remains unaltered
  - But the performance may be affected by such changes

# Semi-structured Data and XML


- XML: Extensible Markup Language
- Will not be covered in detail in class, but many datasets available to download are in this form
  - You will download the DBLP dataset in XML format and transform into relational form (in HW1)
- Data does not have a fixed schema
  - “Attributes” are part of the data
  - The data is “self-describing”
  - Tree-structured

# XML: Example

Attributes



```
<article mdate="2011-01-11" key="journals/acta/Saxena96">  
  <author>Sanjeev Saxena</author>  
  <title>Parallel Integer Sorting and Simulation Amongst CRCW  
    Models.</title>  
  <pages>607-619</pages>  
  <year>1996</year>  
  <volume>33</volume>  
  <journal>Acta Inf.</journal>  
  <number>7</number>  
  <url>db/journals/acta/acta33.html#Saxena96</url>  
  <ee>http://dx.doi.org/10.1007/BF03036466</ee>  
</article>
```



# Attribute vs. Elements

- Elements can be repeated and nested
- Attributes are unique and atomic

# Why XML?

- + Serves as a model suitable for integration of databases containing similar data with different schemas
  - e.g. try to integrate two student databases: S1(sid, name, gpa) and S2(sid, dept, year)
  - Many nulls if done in relational model, very easy in XML
- + Flexible – easy to change the schema and data
- Makes query processing more difficult

## Which one is easier?

- XML (semi-structured) to relational (structured)
- or
- relational (structured) to XML (semi-structured)?

# XML to Relational Model

- Problem 1: Repeated attributes

```
<book>
```

```
  <author>Ramakrishnan</author>
```

```
  <author>Gehrke</author>
```

```
  <title>Database Management Systems</title>
```

```
  <publisher> McGraw Hill
```

```
</book>
```

What is a good relational schema?

# XML to Relational Model

- Problem 1: Repeated attributes

```
<book>
```

```
  <author>Ramakrishnan</author>
```

```
  <author>Gehrke</author>
```

```
  <title>Database Management Systems</title>
```

```
  <publisher> McGraw Hill</publisher>
```

```
</book>
```

Title	Publisher	Author1	Author2

# XML to Relational Model

- Problem 1: Repeated attributes

```
<book>
```

```
  <author>Garcia-Molina</author>
```

```
  <author>Ullman</author>
```

```
  <author>Widom</author>
```

```
  <title>Database Systems – The Complete Book</title>
```

```
  <publisher>Prentice Hall</publisher>
```

```
</book>
```

Does not work

Title	Publisher	Author1	Author2



# XML to Relational Model

Book

BookId	Title	Publisher
b1	Database Management Systems	McGraw Hill
b2	Database Systems – The Complete Book	Prentice Hall

BookAuthoredBy

BookId	Author
b1	Ramakrishnan
b1	Gehrke
b2	Garcia-Molina
b2	Ullman
b2	Widom

# XML to Relational Model

- Problem 2: Missing attributes

```
<book>
  <author>Ramakrishnan</author>
  <author>Gehrke</author>
  <title>Database Management Systems</title>
  <publisher> McGraw Hill
  <edition>Third</edition>
</book>
<book>
  <author>Garcia-Molina</author>
  <author>Ullman</author>
  <author>Widom</author>
  <title>Database Systems – The Complete
Book</title>
  <publisher>Prentice Hall</publisher>
</book>
```

Bookid	Title	Publisher	Edition
b1	Database Management Systems	McGraw Hill	Third
b2	Database Systems – The Complete Book	Prentice Hall	null

# Summary

- Relational data model is the most standard for database managements
  - semi-structured model/XML is also used in practice – you will use them in hw assignments
  - unstructured data (text/photo/video) is unavoidable, but won't be covered in this class
- A DBMS provides data independence and insulates the application programmer from many low level details
- We will learn about those low level details as well as high level data management in this course

Very important

Understand the Course-Policy

See “what is allowed/not allowed”

will be reminded in every hw  
assignment too