CS430/630
Database Management Systems
Spring 2018

Gabriel Ghinita

University of Massachusetts at Boston

People & Contact Information

- Instructor: Gabriel Ghinita
  - **Email:** Gabriel.Ghinita AT umb DOT edu (preferred contact)
  - **Web:** http://www.cs.umb.edu/~gghinita
  - **Phone:** (617) 287-6479
  - **Office:** Science Building, 3rd Floor, Room 88 (S-3-88)

- TA: Mohammad Hadianpour
  - **Email**: Mohammad.Hadian001 AT umb DOT edu
  - **Office**: S-3-124A

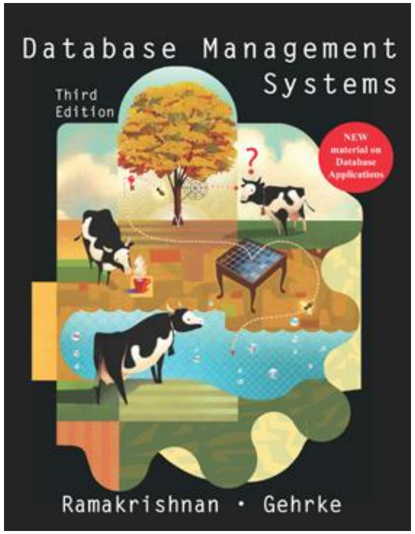
- Course-related emails (for instructor and for TA)
  - Subject line **MUST BEGIN** with **[CS430]** or **[CS630]**

Course Info

- Lecture Hours
  - Tue and Thu , 7:00-8:15pm
- Office Hours
  - Tue & Thu 5:30-7:00pm
  - By appointment (send email)
- Class URL
  - http://www.cs.umb.edu/~gghinita/cs430/
  - http://www.cs.umb.edu/~gghinita/cs630/

Textbook & Recommended Readings

- Textbook
  - *Database Management Systems*, 3rd Edition by Ramakrishnan and Gehrke


Database Management Systems
Third Edition
NEW material on Database Applications
Ramakrishnan • Gehrke

- Other recommended texts
  - *Database System Concepts*, Silberschatz, Korth and Sudarshan, 6th Edition
  - *Database Principles, Programming, and Performance*, P. E. O'Neil and E. J. O'Neil
  - Other resources will be posted in the links section of the site

Prerequisites

- Data Structures and Algorithms
  - CS310
- Programming
  - CS240
- Discrete Math
- Familiarity with UNIX OS
  - Exercises will be executed on Oracle 12G server running on a Unix machine in the CS dept (DBS3 Machine)

Grading

- Final exam (40%) – open book
- Midterm (30%) – open book (Thu March 22nd)
  - Open book does NOT include electronic devices!

- 6 homework assignments
  - 5% each
  - Assignments for CS630 will have additional questions
  - Assignments are individual – submit your own work only!
  - No plagiarism! See student code of conduct

- Lecture attendance is mandatory

Course Materials

- Class URL
  - http://www.cs.umb.edu/~gghinita/cs430/
  - http://www.cs.umb.edu/~gghinita/cs630/

- Blackboard
  - Discussion forums

- Make sure you create Unix course accounts, and that you enroll these accounts for 630 ("apply" procedure)

University Policies

- **Student Conduct**: Students are required to adhere to the University Policy on Academic Standards and Cheating, to the University Statement on Plagiarism and the Documentation of Written Work, and to the Code of Student Conduct as delineated in the University Catalog and Student Handbook. The Code is available online at:

  https://www.umb.edu/editor_uploads/images/life_on_campus/

- **Accommodations:** Section 504 of the Americans with Disabilities Act of 1990 offers guidelines for curriculum modifications and adaptations for students with documented disabilities. If applicable, students may obtain adaptation recommendations from the Ross Center for Disability Services, CC-UL Room 211, (617-287-7430). The student must present these recommendations and discuss them with each professor within a reasonable period, preferably by the end of Drop/Add period.

Course Overview

- Relational Data Model
- Relational Algebra
- Structured Query Language
  - The most important part of the course
- Conceptual design – the ER model
- Database application development
  - Java, PL/SQL
- Design Theory
- Database Security

What is a DBMS?

- Specialized software that provides:
  - Uniform and transparent access to data
    - Application-independence
    - Application/user is oblivious to internal data organization
    - Data organization may change, but applications need not change
  - Efficient access to data
    - Fast search capabilities, indexing
  - Data consistency
    - E.g., cannot delete student record if grade records still in DBMS
  - Concurrent access to data
  - Persistent storage and recovery from failure
  - Security

Why study databases?

- Databases are ubiquitous
  - Behind all web service providers there is a DBMS
    - Most often a very large-scale one
  - Corporations use DBMS for business processes, HR, etc
  - Scientific computing relies on very large amounts of data
    - Humane genome data
    - Biochemistry data (protein sequences)
    - Astronomy data
    - High-energy physics
- DBAs are very well–paid!
  - And even in other IT areas, DBMS skills are a must

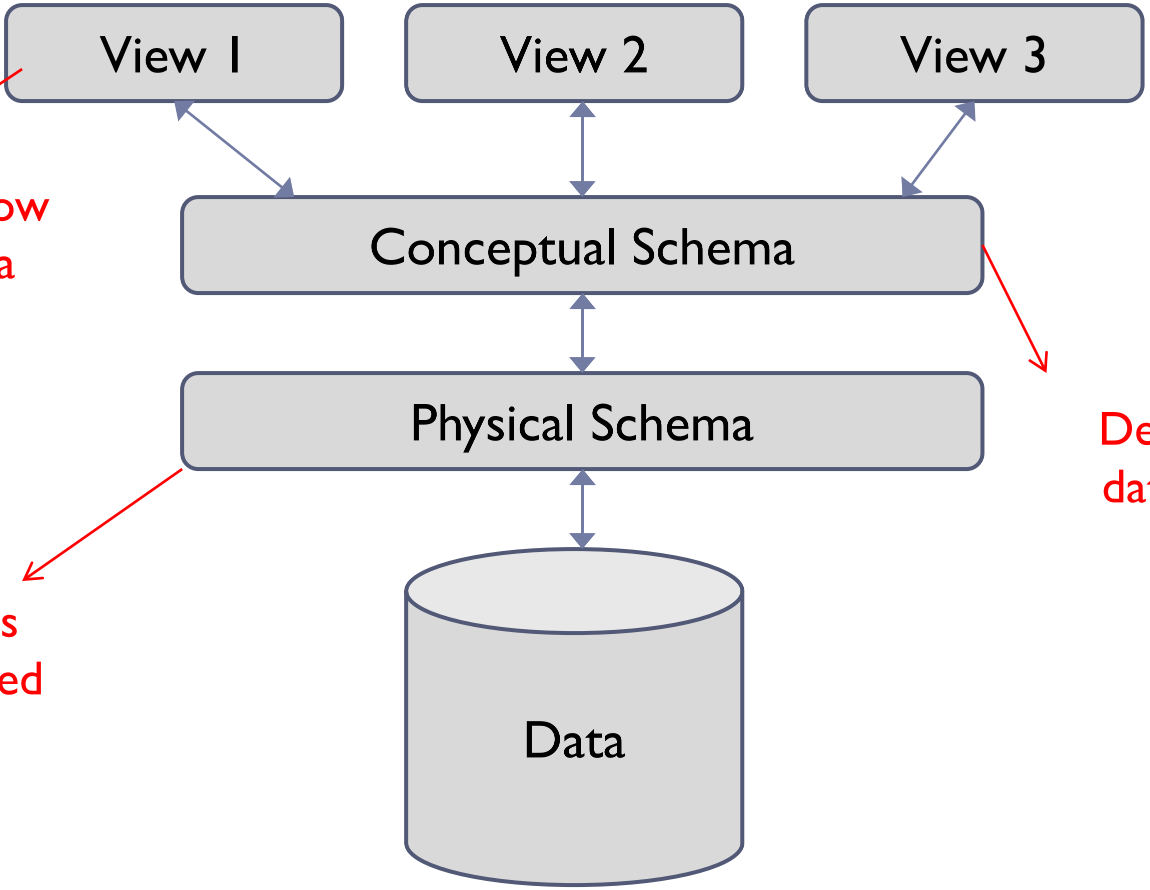
A bit of history …

- First data stores were file systems
  - Does not conform to transparency and uniformity desiderata
  - Search (within file) most often linear
  - Not portable
  - Doesn't handle concurrency properly
    - Sequential access only
- Early DBMS appeared in the 60s
  - Driven by banking and airline industry
  - Relatively small record size, and many concurrent accesses
  - Two prominent models: hierarchical model (tree) and network model (graph)
    - Lack of support for high-level query languages

A bit of history (contd.)

- Relational Databases
  - Major breakthrough, paper written by Codd (1970)
  - *Relations* (tables) with rows (records) and columns (fields)
    - Relationships and constraints among tables
  - Structured Query Language (SQL): high-level, declarative
    - Data definition/ manipulation language
  - Fast search – use of index structures
  - Data access language independent from internal organization
- Newer paradigms
  - Object-oriented and multimedia DB
  - Data Stream Management Systems (DSMS)
  - MapReduce

Levels of Abstraction



Views define how users see data

Defines logical data structure

Describes files and indexes used

Database Management Systems 3rd ed, Ramakrishnan and Gehrke

The Relational Model

Data Model

- Structure of data
  - Relational model uses tables
  - Programming languages deal with arrays, collections, etc
- Operations on the data
  - Queries: operations that retrieve information
  - Modifications: operations that change data
- Constraints
  - Domain constraints (the simplest): e.g., age must be numeric
  - Other constraints: each student has unique matriculation #
- Prominent Data Models
  - Relational model
  - Object-relational model, semi-structured model (XML), E-R

Relational Model

- *Relational database*: a set of *relations*
- *Relation:*
  - two-dimensional *table*, with rows and columns

    #Rows = *cardinality*

    #Columns= *degree* (or *arity*)
  - Each row represents an entity
    - A student, a course, a movie
  - Each column represents a property of the entity
    - Student age, student matriculation #, student gpa
    - Column values are atomic (e.g., integer or string) within given domain
  - Rows are also called tuples or records; columns are also called fields or attributes

“Students” Relation or Table

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

Cardinality = 3, Degree = 5

Relational Schema

- *Schema*: specifies name of relation, plus name and domain of each column

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

- Each relation must have a schema
  - Similar to a data type in programming languages

- Relational database schema = collection of relations' schemas

More about Relations

- Relations are sets of tuples
  - Sets are NOT ordered
  - Do NOT retrieve by order number, but by content!

- Relation Instance
  - Contents of a relation may change over time
    - Tuples are added/deleted/modified
    - E.g., Students join or leave the university
  - Instance represents set of tuples at a certain point in time

- Schemas may change too
  - Although this is not frequent in practice
  - Changing schema is very expensive

Instance of "Students" Relation

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

Cardinality = 3, Degree = 5

Another Instance of “Students”

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8
53660	Korth	korth@math	22	3.6

Cardinality = 4, Degree = 5

Keys

- A key of a relation is a set of fields *K* such that:
  1. No two distinct tuples in ANY relation instance have same values in all key fields, and
  2. No subset of *K* is a key (otherwise *K* is a *superkey*)
- Key may not be unique
  - Multiple candidate keys may exist
  - One of the keys is chosen (by DBA) to be the *primary key*
- Keys are shown <u>underlined</u> in schema
- In the relational model, duplicate tuples do not exist!
  - But most DBMS implementations do allow duplicates
  - Keys constraints must be set by DBA to avoid duplicates

Example of Keys

Students(*<u>sid</u>*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)

- *sid* is a key; {*sid*, *name*} is a superkey

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

- In practice, it is not easy to know when there exists a unique attribute combination in the data (e.g., names)
  - artificial keys are created: student ID, customer ID, etc.
  - SSN is also often used for keys