

# CSC343H Y L 5201 Introduction to Databases

Instructor: Laurent Mignet  
[mignet@cs.toronto.edu](mailto:mignet@cs.toronto.edu)  
University of Toronto- Bahen 5222

05/15/03 CSC343  
L5201

L. Mignet

3

## Grading

- Assignment: 15% each = 45% of grade. Please write legibly, do not use pencil (if you can). Presentation will be considered.
- Midterm: June 26<sup>th</sup> in class. 1H30. 15%
- Final: TBA. 40% of grade. Must obtain 40% in final in order to pass the course.

## Staff

- Instructor: L. Mignet (Bahen 5222)
- TAs:
  - Solmaz Kolahi
  - Daniela Rosu
  - Miles Trochesset
- Tutorials: every Thursday 6-7 pm
- Course home page:  
[http://www.cs.toronto.edu/db/courses/343/Summer2003\\_L5201/](http://www.cs.toronto.edu/db/courses/343/Summer2003_L5201/)

05/15/03 CSC343  
L5201

L. Mignet

2

## Course will cover

- Introduction to databases
- Database conceptual design (Entity-Relationship model)
- Database Logical design (Relational model)
- Relational Query Language (Relational Algebra & SQL)
- Relational Model Theory (Functional dependencies)
- Logical Query Languages

05/15/03 CSC343  
L5201

L. Mignet

4

## WAIVER ...

- 263/265/(228,238)/378 CPGA 3.0
- 228 already waived if needed
- other at my discretion (209/238)
  
- Every cases: see me during the break or after the lecture (even if you already sent me an email)

## What is a DBMS?

- A very large, integrated collection of data describing activities of organizations.
- Models real-world.
  - Relation: (e.g. student, courses, instructors)
  - Relationships: (L. Mignet teaches CSC343)
- A DataBase Management system (DBMS) is a software package designed to store and manage databases

OK!  
Let's started

## History ...

- First DBMS: 60's (*Network Data Model*) by Bachman at General Electric. Standardized by CODASYL.
- Late 60's: IBM's IMS (Inf. Mgmt. Sys.) (*Hierarchical Data Model*).
- 1970: Edgar Codd (at IBM) proposed the *Relational Data Model*: strong theoretical basis.
- 80's-90's: Relational model consolidated. Research on query languages and data models.

## Why use a DBMS?

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security. Different users may access different data subsets.
- Uniform data administration.
- Concurrent access, recovery from crashes.

## Describing Data: Data Models

- A *data model* is a collection of concepts and constructs for describing data.
- A *schema* is a description of a particular collection of data, using a given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.
  - Every relation has a *schema*, which describes the columns, or fields.

## Files vs. DBMS

- Application must transfer large datasets between main memory and secondary storage (e.g. buffering, page-oriented access, 32-bit addressing, etc.).
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users.
- Crash recovery
- Security and access control.

## Describing Data: Data Models (cont.)

- The data model of the DBMS hides details - *Semantic Models* assist in the DB design process.
- Semantic Models allow an initial description of data in the "real world".
- A DBMS do not support directly all the features in a semantic model.
- Most widely used: Entity-Relationship model (E/R).

## The Relational Model (Introduction)

- Central construct: the RELATION : a set of records.
- Data is described through a SCHEMA specifying the name of the relation, and name and type of each field:
  - *Students*(*sid: string, name: string, login: string, age: integer, gpa:real*)
- Actual data: instance of the relations : a set of tuples, : {<53666, Jones, jones@cs, 18, 3.4>, <53688, Smith, smith@ee, 18, 3.2>, <53650, Smith, jones@math, 19, 3.8>, ...}
- Integrity constraints (condition every instance must verify) can also be specified.

05/15/03 CSC343  
L5201

L. Mignet

13

## Example: University Database

- Conceptual schema:
  - *Students*(*sid: string, name: string, login: string, age: integer, gpa:real*)
  - *Courses*(*cid: string, cname:string, credits:integer*)
  - *Enrolled*(*sid:string, cid:string, grade:string*)
    - describes data in terms of the data model of the DBMS
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):
  - *Course\_info*(*cid:string, enrollment:integer*)

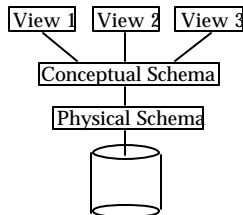
05/15/03 CSC343  
L5201

L. Mignet

15

## Levels of Abstraction

- Data is described at three Levels of Abstraction
- Many views, single conceptual (logical) schema and physical schema.
  - Views describe how users see the data (data tailored to different user groups) .
  - Conceptual schema defines logical structure.
  - Physical schema describes the files and indexes used.



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

05/15/03 CSC343  
L5201

L. Mignet

14

## Data Independence

- Advantage of using a DBMS: applications are (not totally) isolated from changes in the way data is structured and stored.
- Logical data independence: Protection from changes in *logical* structure of data (if the CS is changed, views can be redefined in terms of the new relations).
- Physical data independence: Protection from changes in *physical* structure of data.

☒ One of the most important benefits of using a DBMS!

05/15/03 CSC343  
L5201

L. Mignet

16



## Transaction: An Execution of a DB Program

- Key concept is *transaction*, which is an *atomic* sequence of database actions (reads/writes).
- Each transaction, executed completely, must leave the DB in a *consistent state* if DB is consistent when the transaction begins.
  - Users can specify some simple *integrity constraints* on the data, and the DBMS will enforce these constraints.
  - Beyond this, the DBMS does not really understand the semantics of the data.
  - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user's responsibility!

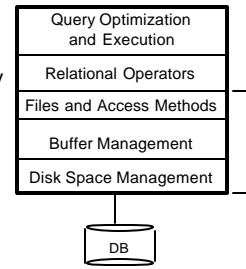
05/15/03 CSC343  
L5201

L. Mignet

21

## Structure of a DBMS

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.



05/15/03 CSC343  
L5201

L. Mignet

23

## Ensuring Atomicity

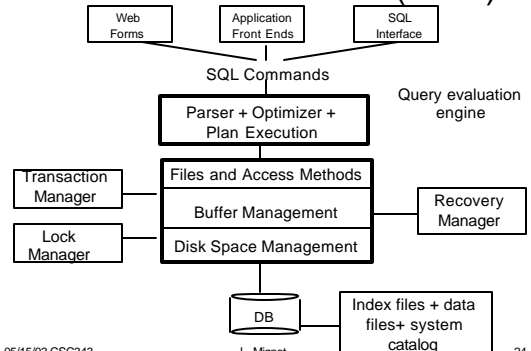
- DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a transaction.
- Idea: Keep a *log* (history) of all actions carried out by the DBMS while executing a set of transactions:
  - Before a change is made to the database, the corresponding log entry is forced to a safe location.
  - After a crash, the effects of partially executed transactions are *undone* using the log. (the change was not applied to database but to the log itself!)

05/15/03 CSC343  
L5201

L. Mignet

22

## Structure of a DBMS (cont.)



05/15/03 CSC343  
L5201

L. Mignet

24

## Databases make these folks happy ...

- End users and DBMS vendors
- DB application programmers
- Database administrator (DBA)
  - Designs logical /physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve

*Must understand how a DBMS works!*

## Summary

- DBMS used to maintain, query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- Levels of abstraction give data independence.
- A DBMS typically has a layered architecture.
- DBAs hold responsible jobs and are well-paid!
- DBMS R&D is one of the broadest, most exciting areas in CS.