LAB 1 Notes

Petko Bakalov
Email: pbakalov@cs.ucr.edu
TA Web page: http://www.cs.ucr.edu/cs166/
Office Hours: Thursday 09:30-11:00, Surge 282 or by appointment (Please send e-mail at leas one day before the time you want to meet me)
Mailing list: http://www.cs.ucr.edu/mailman/listinfo/cs166

A) Hand-out Fortune Slips – Get a list of Students
- Your account has 70 Mb quota which enough
- 45 Mb for a postgres Database, so you have enough space
- If you don't have enough space talk to the system guys and send me e-mail.

B) Database Management Systems by Ramakrishnan & Gehrke - Textbook
http://www.cs.wisc.edu/~dbbook/
- lecture slides, solutions to odd exercises, software others
- About book: Points-to review + exercises excellent resource.

C) Lab Outline

- Ch.1: Overview of Database Systems
  Ch.2: Introduction to Database Design
  Ch.3: The Relational Model
  Ch.4: Relational Algebra
  Ch.5: SQL
  Ch.8: Storage and Indexing
  Ch.9: Storing Data: Disks and Files
  Ch.10: Tree-Structured Indexing
  Ch.11: Hash-Based Indexing
  Ch.12: Overview of Query Evaluation
  Ch.13: External Sorting
  Ch.14: Evaluation of Relational Operators
  Ch.15: A Typical Relational Query Optimizer
  Ch.16: Overview of Transaction Management

D) Chapter 1 – Introduction to Database Systems

1) What is a DBMS?
A software that supports the management of large collections of data.
It supports: Security, data integrity, efficient access, concurrency, data independence, reduces development time and administration, crash recovery.

We could also use proprietary approaches (e.g. everything in text files + application specific code) but that would be too costly.
This book emphasizes on:
- How to design a database that uses a DBMS effectively
- How to organize info in a DBMS and how to maintain it
- How to retrieve it efficiently.

Example
Suppose a supermarket with products. There are several cashiers (employees) managers(employees) and customers transactions. How can all these people work on the same data concurrently, securely, ask different questions (manager), if a crash everything should be restored to the last state

2) Advantages of a DBMS

1) **Data Independence.** Programs are independent of how data is stored on disks. (platform independent in some sense)
2) **Efficient Data Access.** (Indexes, Buffers Manager etc)
3) **Integrity & Security**
    a. Integrity. Check that the salary of an employee salary never exceeds the manager's salary
    b. Security. Enforce access control to different users
    c. Data Administration. Experts can fine tune performance. Monitor that the data is always safe.
4) **Concurrent Access & Crash Recovery**
   e.g. ATM example with 2 credit cards on same account (New York, San Fransisco). Balance is $10000. Peter and his wife withdraw $10000 simultaneously
5) **Reduced Development Time**
    a. Imagine writing all this proprietary source every time we need to computerize a system.
    b. DBMS are large pieces of software with many sophisticated functions.
    c. We don't exploit the potentials of a DBMS by simply creating tables and queries.

3) Disadvantages of using a DBMS

1) **Size Overhead** – Especially if we don't know how to fine tune the performance of such a DBMS (e.g. storing 100GB of data -> imagine all indexes, system catalogs)
2) **Time Overhead** – Real time applications may not afford it
3) If we don't need all the mechanisms flexible querying, security, concurrent access, crash recovery.
   Would somebody spend 200$ for buying a DBMS (+ development cost) in order to save a list of phones.

4) What is a Data Model?

A collection of high level data description constructs that hide many low-level storage details

- Network Data Model
- Hiererchical Data Model (IBM's IMS)
- Relational Data Model (Oracle, SQL Server, IBM DB2, postgres, ingress, Access, MySQL, …)

The internet is powered by several Relational Databases.

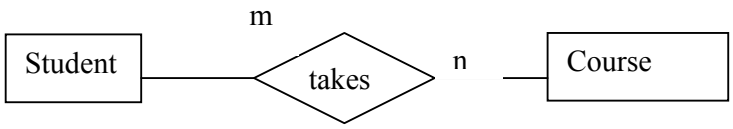
A **semantic data model** is a more abstract, high level data model that makes it easier to come up with a good initial description of the data , **e.g ER model**

5) Relational Model

Relation , a set of records (tuples)

Degree & Cardinality

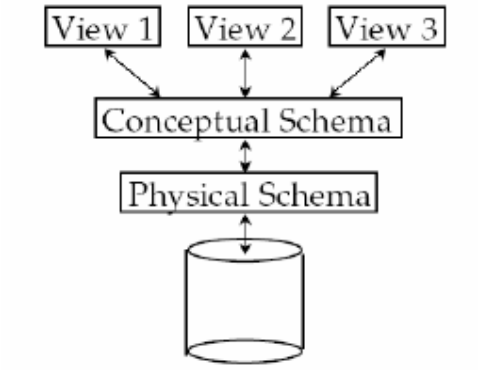
e.g, Student(ssn string, name string, age int, gpa real);
Course(id int, descr string)
Takes(string ssn, id int)



We may define **Integrity Constrains** (conditions that need to be satisfied by the tuples)
Primary Key Constrain -> Uniqueness
Foreign Key Constrain -> Must exist
Participation Constrains -> Total Vs. Partial

6) Levels of Abstraction

1) **Conceptual Schema** (or logical), the relations and relationships
2) **Physical Schema**, (or internal) storage details (the file organization, indexes)
3) **External Schema**, 0:M Views



DDL (Data Definiton Language)
-> SQL commands (both for querying and defining entities)
a) Conceptual & External schema definition
b) Physical Schema is defined with the SQL DDL in most databases

Logical Data Independence -> Achieved by External Schema (Views)
e.g. if we split Student -> GradStudent, UndergStudent we will still be able to answer whch courses is each student taking
Physical Data Independece - > Achieved by Conceptual Schema
Our application refers to the Relations rather than bytes on disk.

7) Queries in a DBMS

How many students are enrolled in CS14?
The relational model provides to powerful **querying languages**

- Relational Algebra
- Relational Calculus

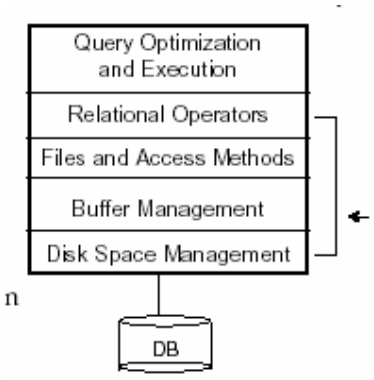
In real life we use a Data Manipulation Language (DML) which provides constructs for inserts, deletes, updates.
SQL at the end of the day provides

a) DML
b) DDL

8) DB People

- **Database Implementers –** create DBMS software (e.g. oracle's employees)
- **End Users –** store and retrieve data from a DBMS
- **Database Application Programmers** – implement functionality on top of external schemas. (they don't declare tables, indexes, views, etc) They write queries
- **DBA (Database Administrator)**
  - **Design Conceptual(relations) & Physical Schema (file organ. & indexes)**
  - **Security & Authorization**
  - **Data Availability and recovery from failures**
  - **Database Performance Tuning**

9) Database organization



What is a transaction?
A set of actions on a DBMS where the Atomicity-Consistency-Durability-Integrity properties are enforced.Suppose that you