Introduction to Database Systems

Instructor: Maryam Hasan
maryamhn@sfsu.edu

Course Syllabus

Course Title: Introduction to Database Systems
Course Number: CSC 675/775

Schedule Section1: Thursday 4 pm to 6:45pm
Schedule Section2: Friday 4pm to 6:45pm
Office Hours: Thursday 6:45 to 7:40pm
Friday 3pm to 4pm
Office Location: Thornton hall, room 961

Prerequisite:
CSC 413 with grade of C or better.
Students who have completed CSC 675 may not take CSC 775 for credit.
Students should be familiar with memory based data structures, including binary search trees and hash tables.
knowledge of the Java language and object oriented design.

Course Title: Introduction to Database Systems
Course Number: CSC 675/775

Textbook:
Database Management Systems (3rd Edition), By Ramakrishnan, Publisher: McGraw-Hill, 2003

Course Website: ilearn.sfsu.edu
The course material will be available through ilearn.

Method of Evaluation
4 Homework assignments: 25%
In-class quizzes/tests/participation: 10%
Midterm exam: 25%
Final exam: 25%
Course Project: 15%

Course Title: Introduction to Database Systems
Course Number: CSC 675/775

Course Policies:

Attendance: Attendance is mandatory.

Class pop-up quizzes: No announcement in advance.

Homeworks: Homework assignments will be due before class time on the due date. Late homeworks will not be accepted.

Project: A group project with 3 or 4 members.

Exams: There are two exams in this course, midterm and final. Rescheduling an exam will only be allowed in highly selective and pre-approved cases. If the scheduled exam dates are in conflict with your religious observances, you must notify the instructor, in writing, at least two weeks in advance of the exam.

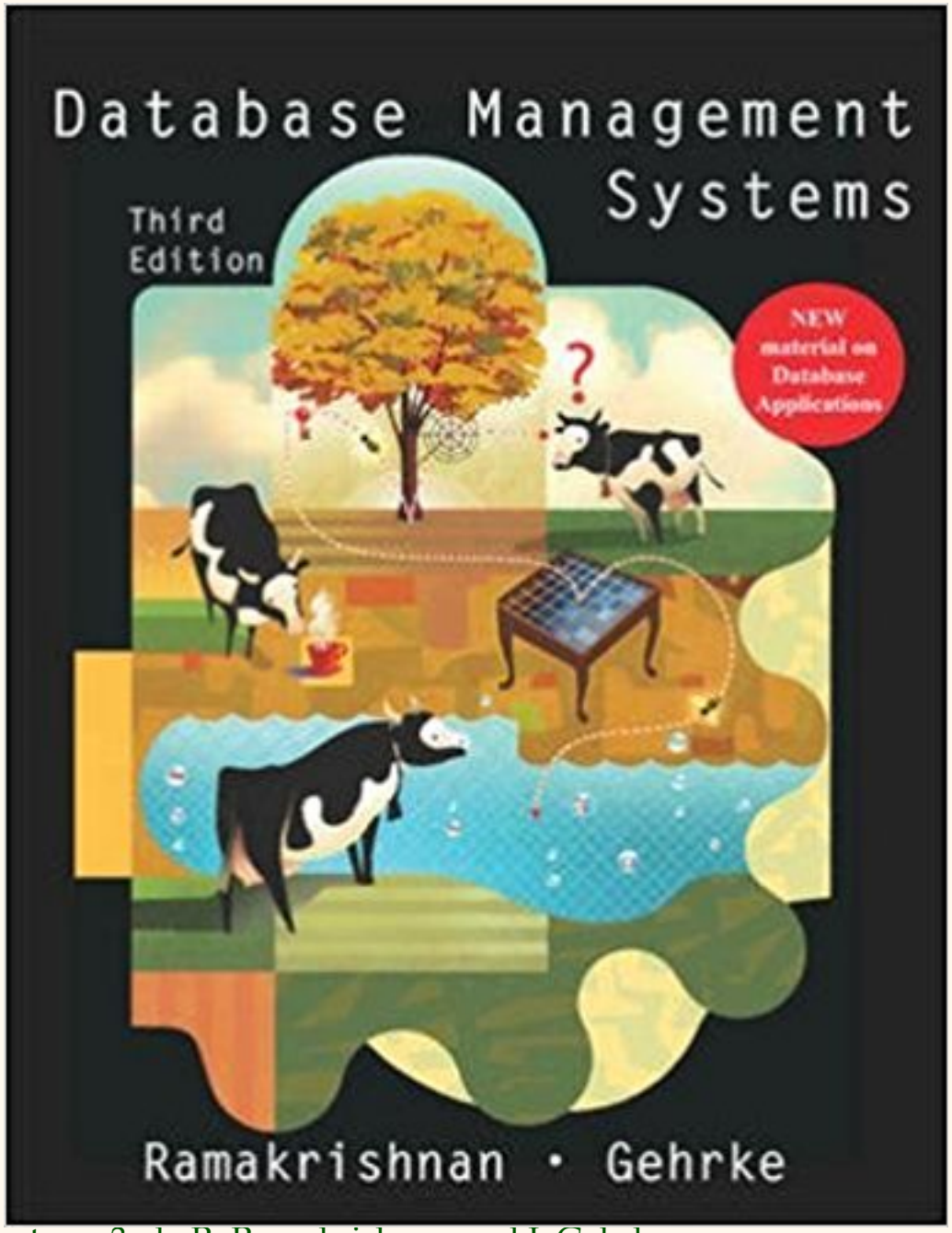
Course Title: Introduction to Database Systems
Course Number: CSC 675/775

Course Policies:

Cheating and plagiarism: Any form of cheating or plagiarism will incur very serious consequences. Carefully review the department's policy on this matter: http://www.cs.sfsu.edu/plagarism.html

Students with disabilities: Students with disabilities who need reasonable accommodations are encouraged to contact the instructor. The Disability Programs and Resource Center (DPRC) is available to facilitate the reasonable accommodations process. The DPRC is located in the Student Service Building and can be reached by telephone (voice/TTY 415-338-2472) or by email (dprc@sfsu.edu)

Week	Topic	Readings	Homework Assignments
1. January 30	Introduction and Basic Concepts	Ch 1	
2. February 6	Database Design (ER Model)	Ch 2	
3. February 13			
4. February 20	Relational Model	Ch 3	HW1
5. February 27		Ch 5	
6. March 5	Structured Query Language (SQL)		Project Design
7. March 12	Structured Query Language (SQL)	Ch 5	
8. March 19			
9. March 26	Spring Recess, No class		HW2
10. April 2	Structured Query Language (SQL) Relational Algebra	Ch 5	
11. April 9	Midterm exam		
13. April 16	Data Storage, Tree Indexes	Ch 8, Ch 10	Project Operation due
13. April 23	Data Storage, Tree Indexes	Ch 8, Ch 10	HW3
14. April 30	Transaction Management	Ch 16	
15. May 7	Concurrency Control, and Crash Recovery	Ch 17 Ch 18	HW4
16. May 14	Project Presentations, and Final exam review	Ch 17 Ch 18	
17. May 21	Final exam		


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

What you will learn in this course

- Overview of DBMS
- Data models & Database Design
  - ER Model
  - Relational Model
- Structured Query Language (SQL)
- Overview of data Storage and Indexing
- Tree indexes and Hash-based indexes
- External Sorting
- Transaction Management
- Concurrency control and crash recovery

What you will learn in this course

This class teaches **the basics** of how to use and manage data

Chapter 1
OVERVIEW OF DATABASE SYSTEMS

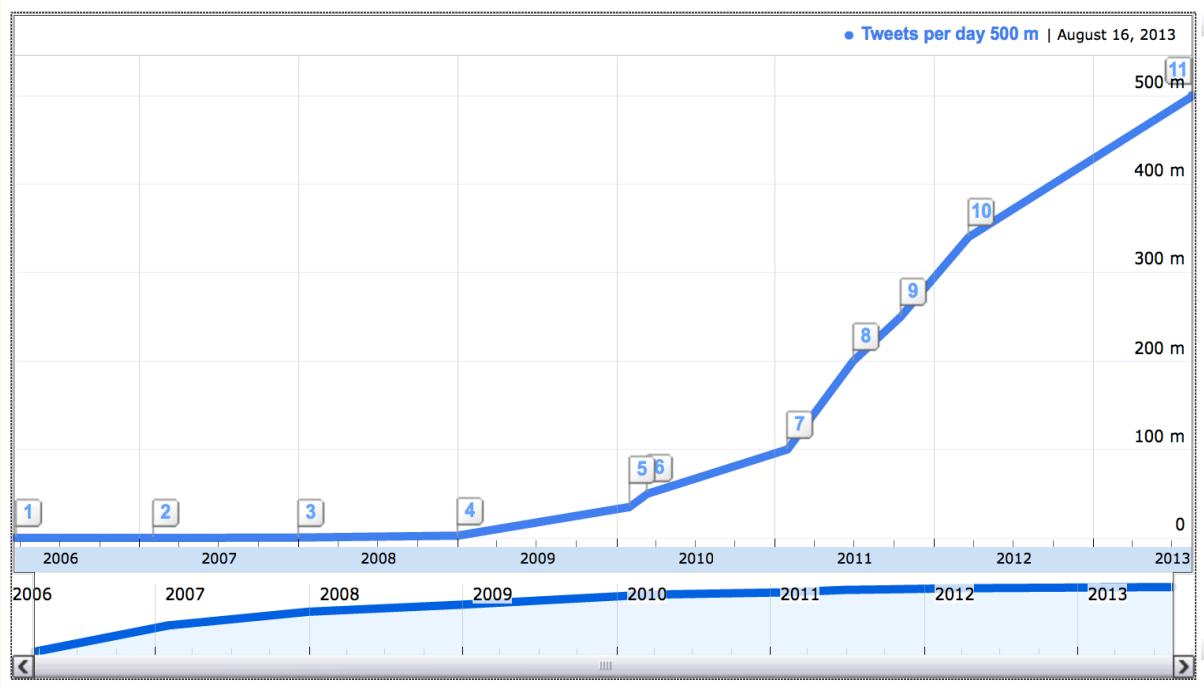
This Lecture

- Motivation
- Definition of DBMS
- Data models & the relational data model
- Schemas & data independence

The world is increasingly **driven by data…**

Data is growing

- High velocity and volume of data
  - More than 6,000 tweets per second (**500 million** tweets per day) are posted on average, (see below)



The number of tweets posted per day
http://www.internetlivestats.com/twitter-statistics

Why do we care about data?

- Data is everywhere!
- Data brings Knowledge.
- Knowledge brings power.

Why should **you** study databases?

- To know **fundamentals of data management**
  - How to design databases, query databases, build applications with them.
- To know **how database management systems work**

Many great computer systems ideas started in DB.

File systems and DBMS

File System:

- In a file system data are directly stored in a set of files on disk.
- Allow read/write/seek/protection on a file
- data is physically accessed and not integrated

DBMS:

- A DBMS is a collection of computer programs that is created for the management (i.e. organization, storage and retrieval) of several databases
- data is logically accessed and integrated:
  - query language

File systems VS. DBMS
*Why should **you** use databases?*

Reason1: concurrency
- Two users edit the same file.
- Whose changes is saved first?

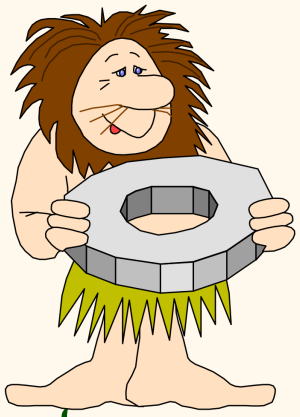
Reason2: crash
- A user is updating a file.
- The power goes out
- What changes are saved?

Reason3: Big data
- Searching or sorting a huge file

Why Use a DBMS?

- Data independence:
  - A DBMS provides an abstract of the data that hides details of data representation and storage
- Efficient data access:
  - A DBMS utilize techniques to store and retrieve data efficiently
- Data integrity and security:
  - A DBMS can enforce access controls to different users
- Concurrent access and crash recovery:
  - Users can think they are using a single-user system.
- Reduced application development time:
  - Many tasks are handled by the DBMS
- Uniform data administration

What Is a Database?



- A database is a very large, integrated collection of structured data.

- A DB models a real-world enterprise
  - Entities (e.g., students, courses, faculty)
  - Relationships between entities (e.g., Jonh is taking CS564)

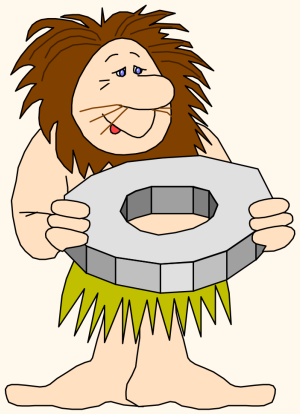
Examples of Database Applications

- Universities
- Banking systems
- Airline systems
- What else?!

A Motivating, Running Example

- Consider building a university database:
  - Entities:
    - *Students*
    - *Courses*
    - *Professors*
  - Relationships:
    - *Who takes what*
    - *Who teaches what*

What Is a DBMS?



- A *Database Management System (DBMS)* is a software package designed to store and manage databases.
  - Set of programs that provide efficient and convenient access to the data

- Relational Database Management Systems (RDBMS)
  - Most widely used database systems

In this class we learn about Relational databases with transactions!

Example DBMS

- Relational DBMS:
  - Oracle, IBM DB2, Microsoft SQL server
  - Open source: MySQL, PostgreSQL
  - Recently: In-memory, column-oriented databases

- No-SQL databases:
  - Provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.
  - Hadoop, Spark, Cassandra, MongoDB, Oracle NoSQL

- Distributed DBMS:
  - Amazon redshift, MS Azure

Data Models

- A *data model* is a collection of concepts for describing data that hides low level details.
  - 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.

Example: University Database

- Logical schema:
  - *Students(sid: string, name: string, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*

sid	Name	Gpa
101	Bob	3.2
123	Mary	3.8

An instance of the Students relation

cid	cname	credits
564	564-2	4
308	417	2

An instance of the Courses relation

Example: University Database

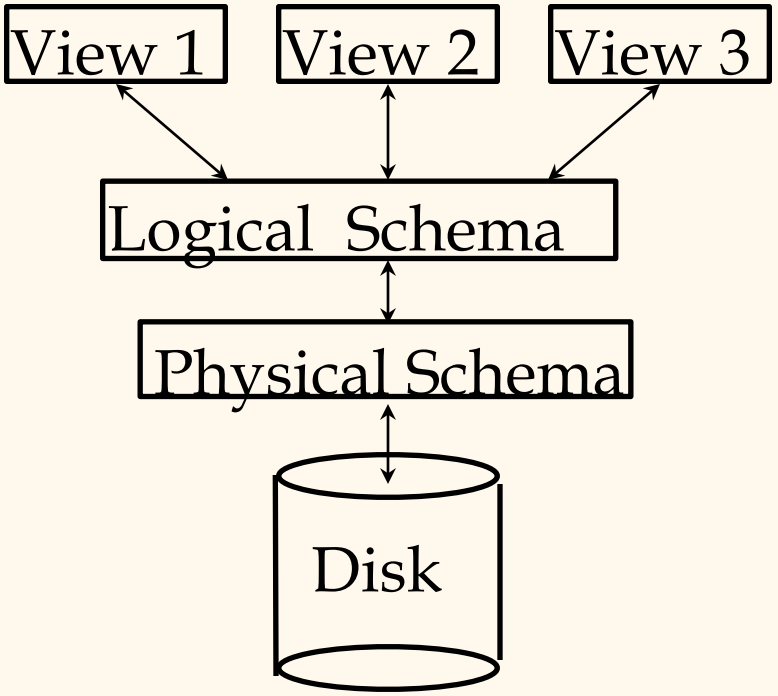
- Logical schema:
  - *Students(sid: string, name: string, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - Enrolled(sid: string, cid: string, grade: string)

sid	cid	Grade
123	564	A

An instance of the Enrolled relation

Levels of Abstraction of Schema

- Many *views*, single *conceptual (logical) schema* and *physical schema*.
  - **Views** describe how users see the data. Derived from logical view.
  - **Logical/conceptual schema** defines logical structure (previous slide).
  - **Physical schema** describes the files and indexes used.



☛ *Schemas are defined using DDL*

Example: University Database

- Logical/conceptual schema:
  - *Students(sid: string, name: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*

- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.

- External Schema (Views):
  - *Course_info(cid: string, fname: string)*

Data Independence *

- Applications do not need to know about how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data.
- *Physical data independence*: Protection from changes in *physical* structure of data.

☛ *One of the most important benefits of using a DBMS!*

Queries in a DBMS

- Queries in University database:
  - What is the name of student with id 123567
  - How many students are enrolled in CS564
  - Is any student with a GPA less than 3.0 enrolled in CS564

☛*data is modified/queried using DML.*

This Lecture

- Transactions
- Concurrency & locking
- Atomicity & logging
- Summary

Challenges with Many Users

Performance: Concurrent execution of user programs is essential for good DBMS performance.

- Because disk accesses are frequent, and relatively slow, it is important to hide the disk latency by doing more CPU work on several user programs concurrently
- DBMS provides concurrent access of several users

Suppose that our university database application serves 1000's of users or more- what are some **challenges?**

Concurrency Control

Consistency: Concurrent access can lead to update problems.

- Interleaving actions of different user programs can lead to inconsistency:
  - E.g., check is cleared while account balance is being computed.

- DBMS ensures such problems don't arise: users can pretend they are using a single-user system.

Transactions

Transaction: Any Execution of a user program

- This is the basic unit of change in a DBMS.

- Key concept is *transaction*, which is an *atomic* sequence of database actions (reads/writes).
  - If a user cancels a transaction, it should be as if nothing happened!

Atomicity: An action either completes *entirely* or *not at all*

Transactions

- Each transaction, executed completely, must leave the DB in a *consistent state*.
  - Users can specify some *integrity constraints* on the data, and the DBMS will enforce these constraints (e.g., 'each course is assigned to exactly one room').
  - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user's responsibility!

Consistency: An action results in a state which conforms to all integrity constraints

However, note that the DBMS does not understand the *real* meaning of the constraints– consistency burden is still on the user!

Scheduling Concurrent Transactions

- DBMS ensures that execution of {T1, ... , Tn} is equivalent to some *serial* execution T1' ... Tn'.
- One way to accomplish this: **Locking**

Before reading/writing an object, a transaction requests a lock on the object, and waits till the DBMS gives it the lock.

Strict 2PL locking protocol: All locks are released at the end of the transaction.

Idea: If T_i wants to write to an item x and T_j wants to read x, then T_i, T_j **conflict.** Solution via **locking:**

- one of them, say Ti, will obtain the lock on X first and Tj is forced to wait until Ti completes; this effectively orders the transactions.

Scheduling Concurrent Transactions

What if T_i and T_j need X and Y, and T_i asks for X before T_j, and T_j asks for Y before T_i?
(Deadlock!) Ti or Tj is aborted and restarted!

All concurrency issues handled by the DBMS...

Crash Recovery

❖DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a transaction.

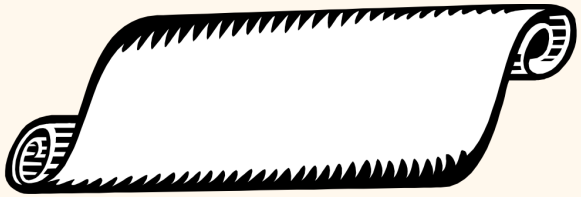
❖One way to accomplish this: **Write-ahead logging (WAL)**

Crash Recovery

- WAL protocol:
  - 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 (disk).
    - After a crash, the effects of partially executed transactions are *undone* using the log. (Thanks to WAL, if log entry wasn't saved before the crash, corresponding change was not applied to database!)

All atomicity issues also handled by the DBMS...

The Log



- The following actions are recorded in the log:
  - *Ti writes an object*: the old value and the new value.
    - Log record must go to disk *before* the changed page!
  - *Ti commits/aborts*: a log record indicating this action.
- Log records chained together by transaction id, so it's easy to undo a specific transaction (e.g., to resolve a deadlock).
- Log is often *duplexed* and *archived* on "stable" storage.

> - All log related activities (such as lock/unlock, dealing with deadlocks etc.) are handled by the DBMS.

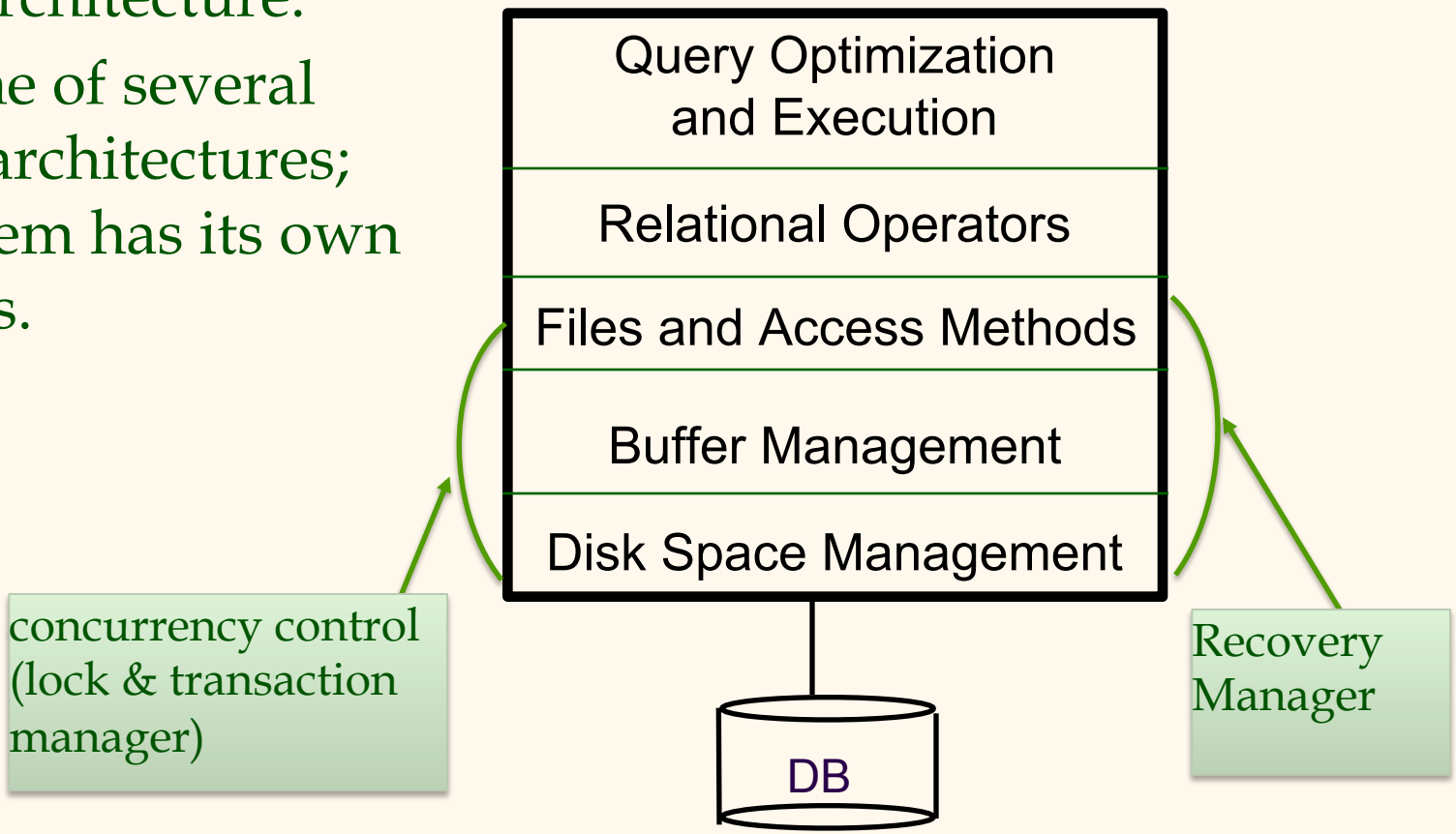
Databases make these folks happy ...

- End users
  - Users who use the existing application to interact with the database.
  - E.g., online library system, ticket booking systems, ATMs etc
- DB application programmers
  - They are the developers who interact with the database using DML queries.
  - DML queries are written in the application programs like C, JAVA
- DB Designers and Developers
  - Designs logical / physical schemas
  - They communicate with the end-users and understand their needs
  - They directly interact with the database by means of query language like SQL
- DB administrators (DBA)
  - Handles security and authorization
  - Data availability, backup, crash recovery
  - Database tuning
- DBMS vendors, programmers
  - Oracle, IBM, MS, Sybase,

Must understand how a DBMS works!

Structure of a DBMS

- A typical DBMS has a layered architecture.
- This is one of several possible architectures; each system has its own variations.



History of Database Technology

- Relational Model based Systems:
  - Relational model was originally introduced in 1970
  - Relational Model was researched and experimented within IBM Research and several universities.
  - RDBMS Products emerged in the early 1980s

Summary

- DBMS are used to maintain, query and manage 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.