CSC675/775
Introduction to Database Systems

Instructor: Maryam Hasan
maryamhn@sfsu.edu

This Lecture

Course Syllabus

Motivation

Definition of DBMS

Data models & the relational data model & NoSQL

Schemas & data independence

Transactions

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 7:45pm
Friday 3pm to 4pm
Office Location: Online via Zoom

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 Syllabus: reading references

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

Another reading reference:
Fundamentals of DatabDatabase Systems: Design, Implementation, & Management 13th, by Carlos Coronel

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

Course Syllabus:Evaluation

Method of Evaluation

4 Homework assignments: 25%
In-class quizzes/discussion/participation: 10%
Midterm exam: 25%
Final exam: 25%
Course Project: 15%

Course Syllabus: Tasks

Attendance: Attendance is mandatory.

Class quizzes: No announcement in advance.

Homeworks: Homework assignments will be due one week after they are posted on ilearn.

- Late work penalized 20% per day
- Late homeworks will not be accepted after the solutions are posted.

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 one weeks in advance of the exam.

Course Syllabus: 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)

Course Material policy:

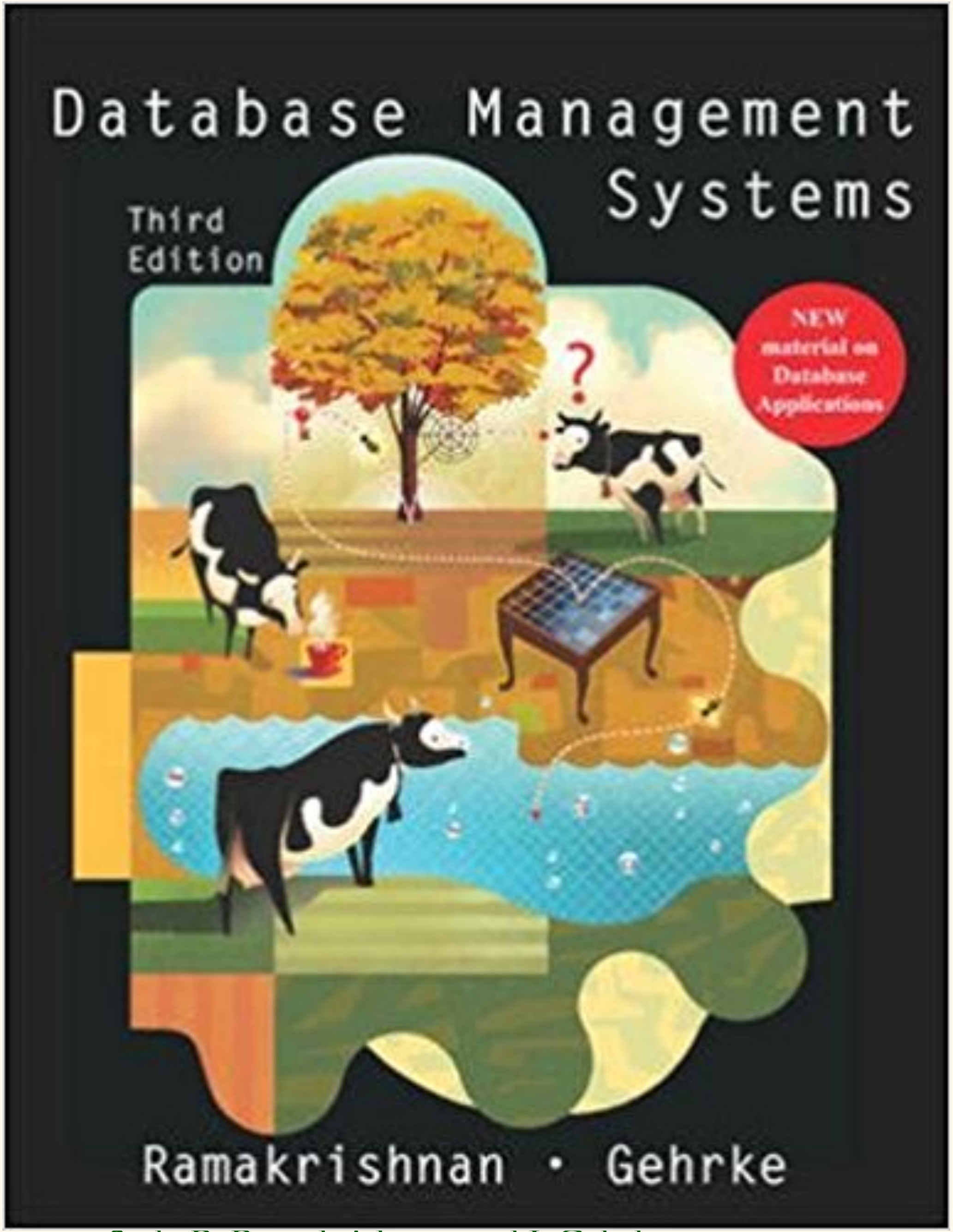
Reproducing and distributing course material including homework assignments/ solutions, exams and lecture slides and recordings to any outside service or website is prohibited.

Tentative Schedule:

Week	Topic	Readings	Homework Assignments
1. January 27	Introduction and Basic Concepts	Ch 1	
2. February 3 3. February 10	Database Design (ER Model)	Ch 2	HW1
4. February 17 5. February 24 6. March 3	Relational Model Relational Algebra	Ch 3 Ch 4	Project Design due
7. March 10 8. March 17	Structured Query Language (SQL)	Ch 5	HW2
9. March 24	No class, Spring Recess		
10. March 31	Data Storage, File management	Ch 8	
11. April 7	Midterm exam		
12. April 14	Tree Indexes	Ch10	HW3
13. April 21	Transaction Management	Ch 16	
14. April 28	Concurrency Control, and Crash Recovery	Ch 17 Ch 18	Project Operation due
15. May 5	NoSQL Databases External Merge Sort (?)	Ch13 (?)	
16. May 12	Project Presentations and Final review		HW4
17. May 19	Final exam		

Tentative Schedule:

Week	Topic	Readings	Homework Assignments
1. January 28	Introduction and Basic Concepts	Ch 1	
2. February 4	Database Design (ER Model)	Ch 2	
3. February 11			HW1
4. February 18	Relational Model	Ch 3	Project Design due
5. February 25	Relational Algebra	Ch 4	
6. March 4			
7. March 11	Structured Query Language (SQL)	Ch 5	
8. March 18			HW2
9. March 25	No class, Spring Recess		
10. April 1	Data Storage, File management	Ch 8	
11. April 8	Midterm exam		
12. April 15	Tree Indexes	Ch10	HW3
13. April 22	Transaction Management	Ch 16	
14. April 29	Concurrency Control, and Crash Recovery	Ch 17 Ch 18	Project Operation due
15. May 6	NoSQL Databases External Merge Sort (?)	Ch13 (?)	
16. May 13	Project Presentations and Final review		HW4
17. May 20	Final exam		



Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

What you will learn in this course

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

What you will learn in this course

This class teaches **the basics** of how to use and manage data:
that is, acquire, process, store, and deliver

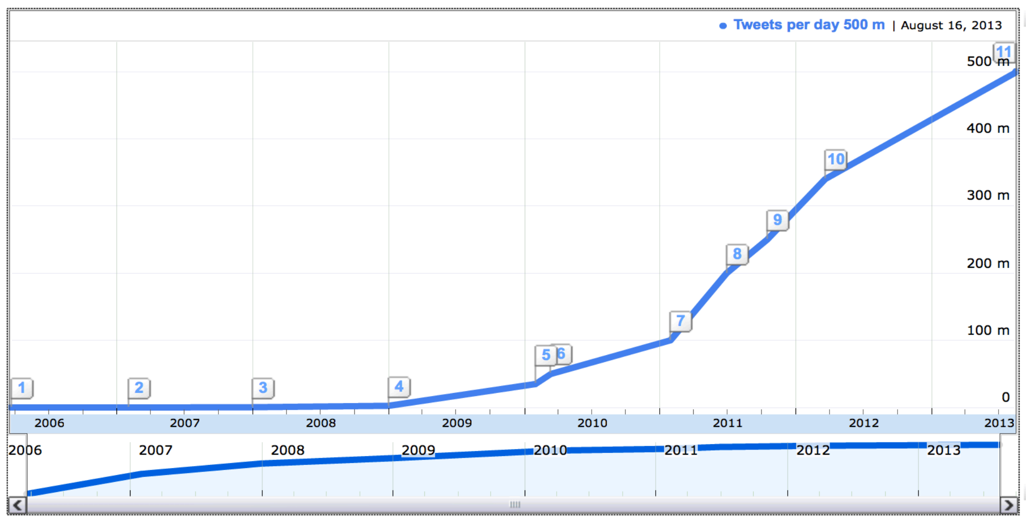
Chapter 1

OVERVIEW OF DATABASE SYSTEMS

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

Data is growing

- High velocity of data
- Large volume of data
- Wide variety of data:
  - text, image, video, audio



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

- In 2020, people created 1.7 MB of data every second.
- By 2025, 200+ zettabytes of data will be in cloud storage around the globe.
- By the end of 2020, **44 zettabytes** will make up the entire digital universe.
- Every day, **300 billion emails** are sent, and **500 million Tweets** are made.

Why do we care about data?

- Data is everywhere!
- Data brings Knowledge.

Storing and Managing Data is one of the primary uses of computers

Why should **you** study databases?

To know **fundamentals of data management**

- **How to use database management systems**
  - How to design databases, query databases, build applications with them.

- **How database management systems work**
  - Data storage, Indexing, Concurrency control, Crash recovery, External sorting

Many great computer systems ideas started in DB.

File systems and DBMS

File System:

- Provided as part of the operating systems
- 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: Big data

- Searching or sorting a huge file

Reason2: Concurrency

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

Reason3: Crash

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

More…

- fine-grained, content-based access control
- Data redundancy and inconsistency

File systems VS. DBMS

- The file systems has the following drawbacks:
  - Difficulties with large data
  - Problems with concurrent accesses
  - Problems with recovering after crash
  - Problems of fine-grained, content-based access control
  - Problems of data integrity
  - Data redundancy and inconsistency

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.

- Data collections are:
  - **Large**: data size is larger than the capacity of computer main memories
  - **Persistent**: the data last for a long period of time which is independent from the executions of the application programs that create and use the data
  - **Structured**: created based on a data model
  - **Shared**: used by different applications and users

Examples of Database Applications

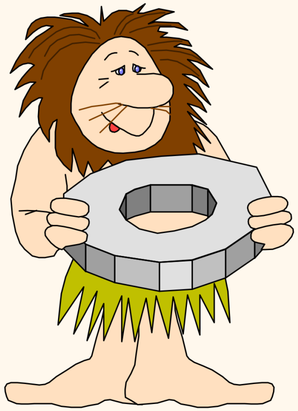
- Database Examples:
  - Universities
  - Banking systems
  - Airline systems
  - What else?!

A Motivating, Running Example

- A DB models a real-world enterprise
  - Entities
  - Relationships between entities

- 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, reliable 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, SQLite
  - Recently: In-memory (memSQL), column-oriented databases (Vertica)

- 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.
  - Redis, Cassandra, MongoDB, Neo4J, Oracle NoSQL

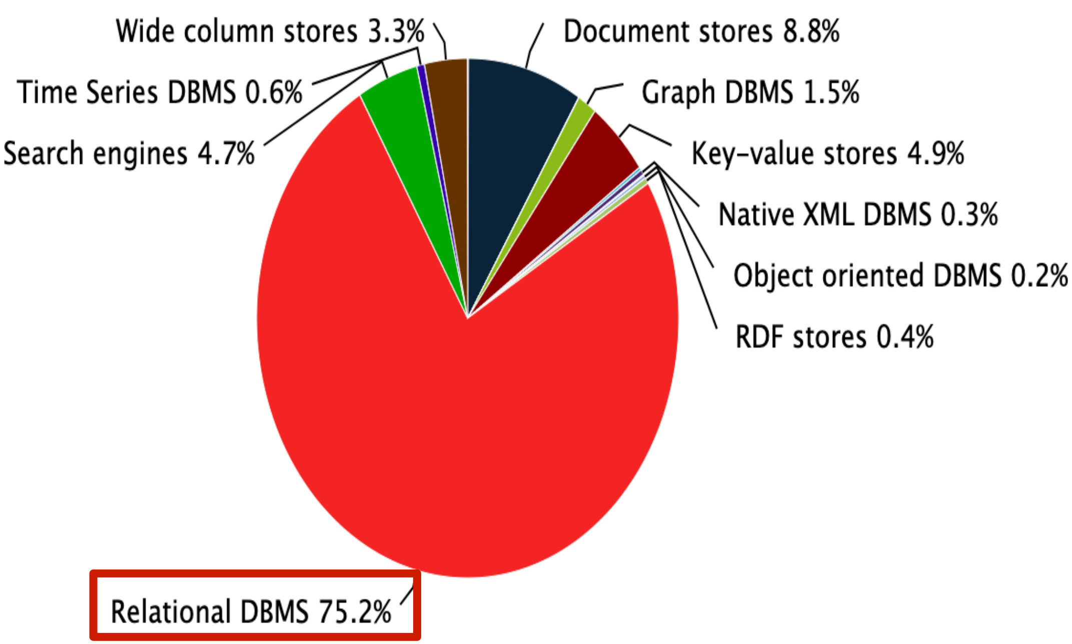
- Distributed DBMS:
  - Amazon redshift, MS Azure

NoSQL Data Types

Four main types:

- key-value stores
- document databases
- column-family (aka big-table) stores
- graph databases

Relational DBs and SQL are popular



© 2019, DB-Engines.com

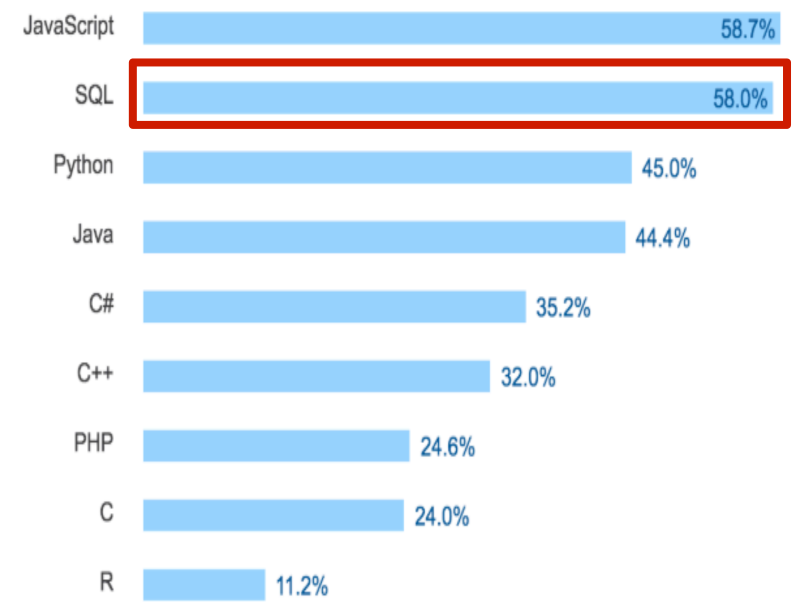


Image: Stack Overflow Developer Survey 2017

RDBMS: Most popular DB model
(according to web mentions)

SQL: Essential developer skill
(used to access RDBMS)

Data Models

- Build databases:
  - Data modeling
  - Query languages

- 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)*
  - *Enrolled(sid: string, cid: string, grade: string)*

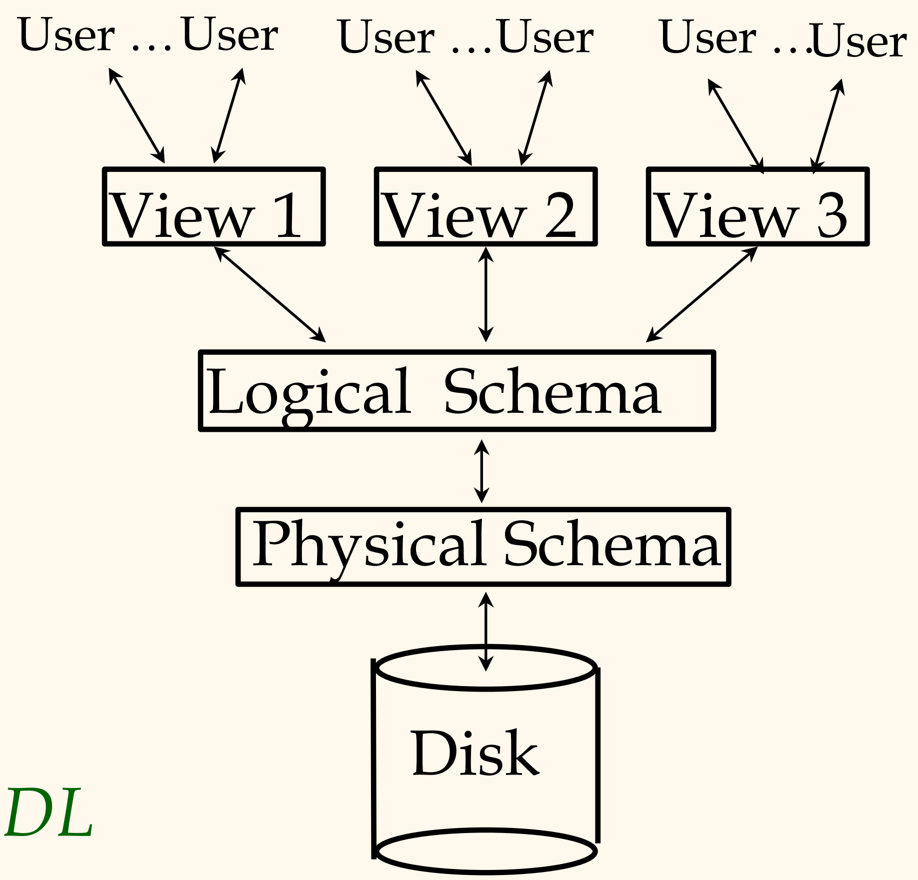
sid	Name	Gpa
101	Bob	3.2
123	Mary	3.8

sid	cid	Grade
123	564	A

cid	cname	credits
564	564-2	4
308	417	2

Levels of Abstraction of Schema

- Many *views*, single *conceptual (logical) schema* and *physical schema*.
  - **Views** describe how users see the data. Derived from logical layer.
  - **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, dept: 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.*

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 required for performance

Concurrency Control

Consistency: Concurrent access can lead to update problems.

- Interleaving actions of different user programs can lead to inconsistency:
  - E.g., deposit is submitted while account balance is being computed.
- DBMS ensures such problems don't arise: users can pretend they are using a single-user system.

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)**

Transactions

Transaction: Basic unit of change in a DBMS.

- Is a sequence of read/write actions:
    - Atomicity
    - Consistency
    - Isolation
    - Durability

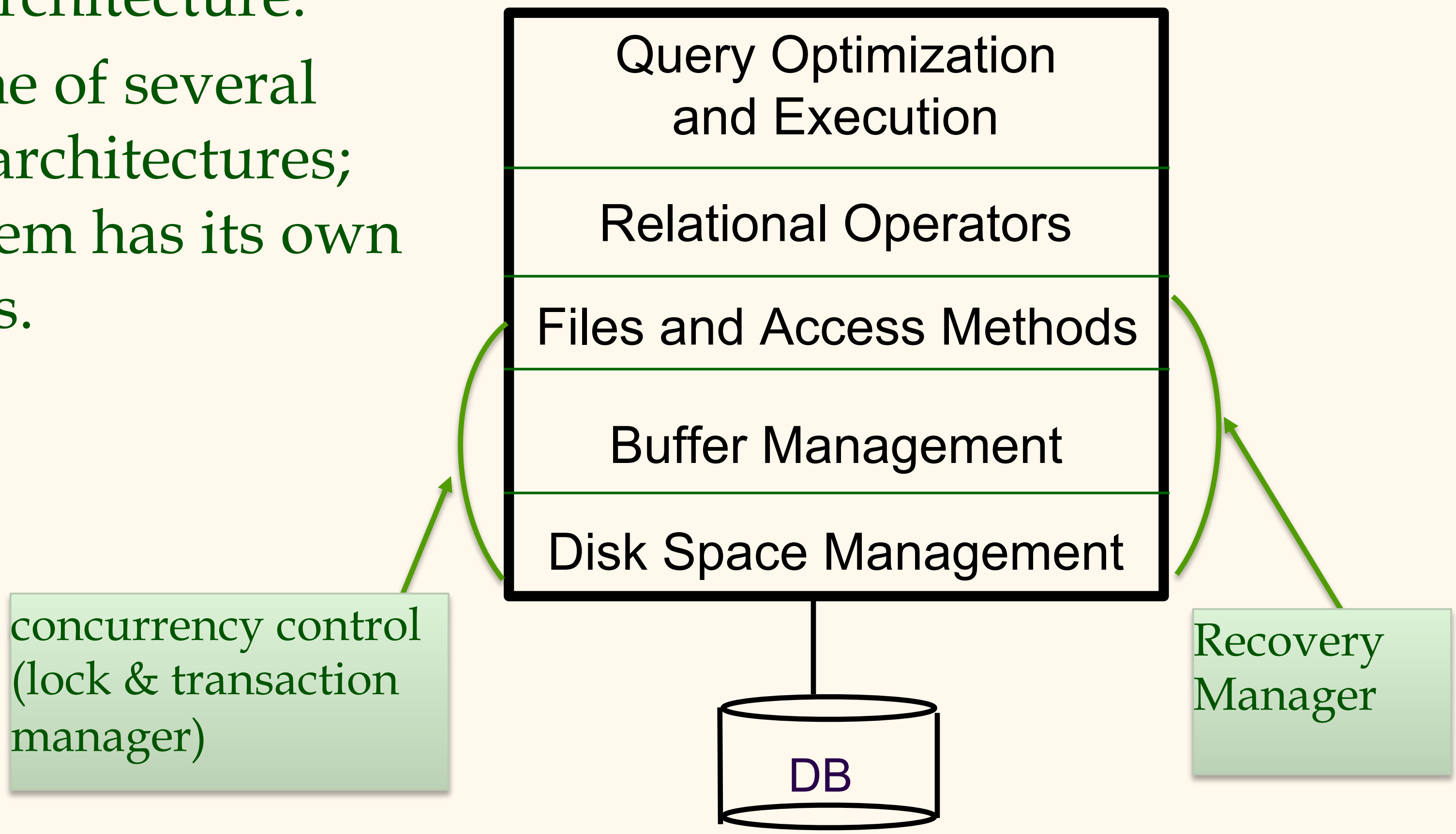
Databases make these folks happy ...

- 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
  - 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
    - 1980: IBM DB2, Oracle, Sybase
    - 1996: PostgreSQL – UC Berkeley Research Project

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.