Database Management Systems

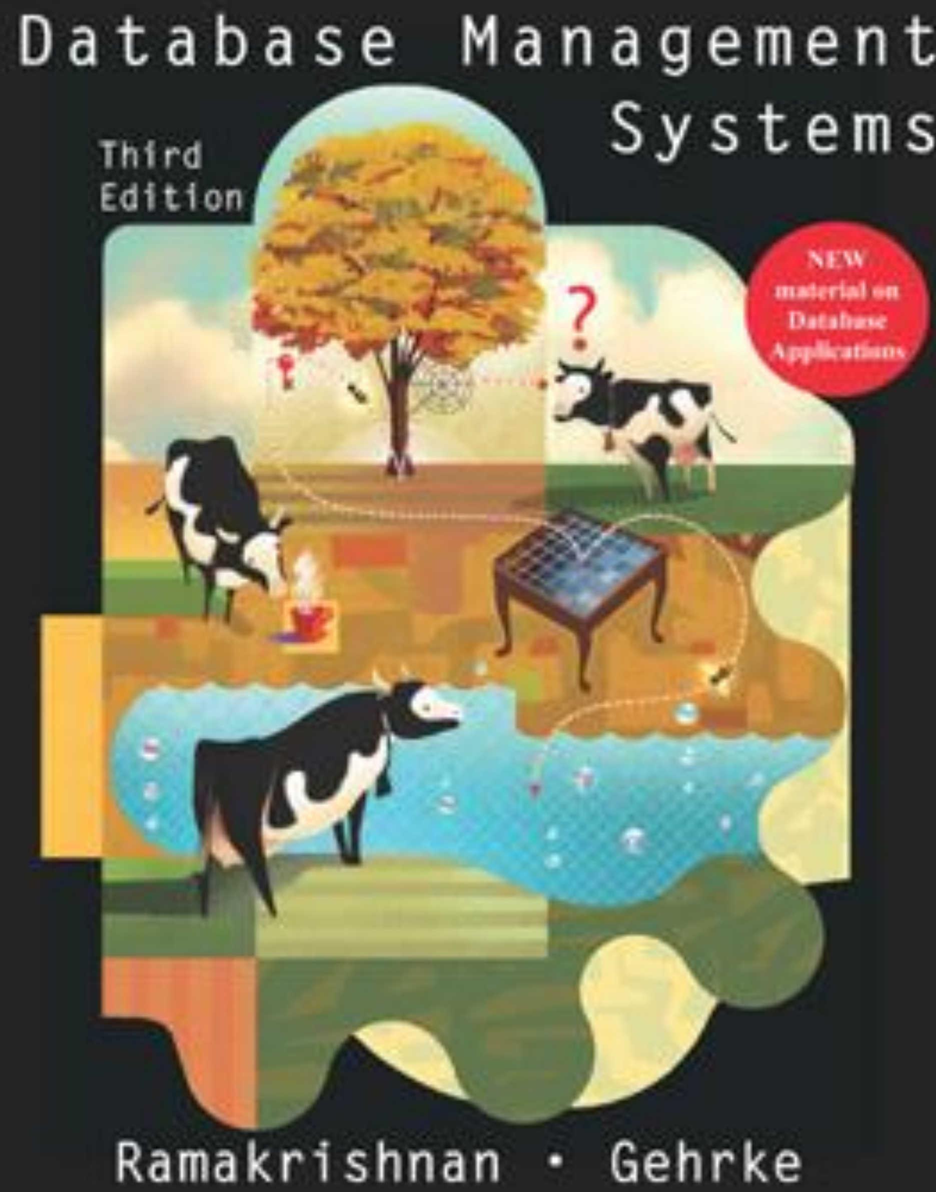
Ehsan Noei
e.noei@utoronto.ca

UNIVERSITY OF TORONTO

Course

- Database Management Systems
  - Database Systems
  - Relational Model
  - Logical Design
  - Schema Refinement and Normal Form
  - SQL

Textbook


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

Problem

- Suppose you want to gather students’ information. How do you do it?
- What information?
  - Name
  - E-mail
  - Age
  - GPA

name	email	age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

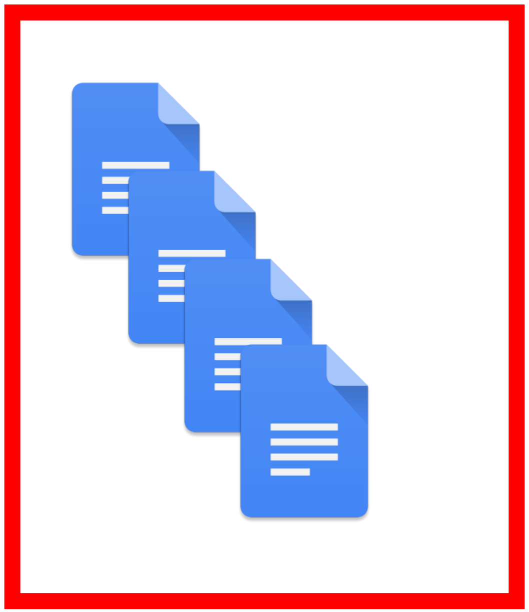
Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

Papers to Computers

Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

Papers to Computers

Database



File Database

Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

Column

Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

Row, Record

Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

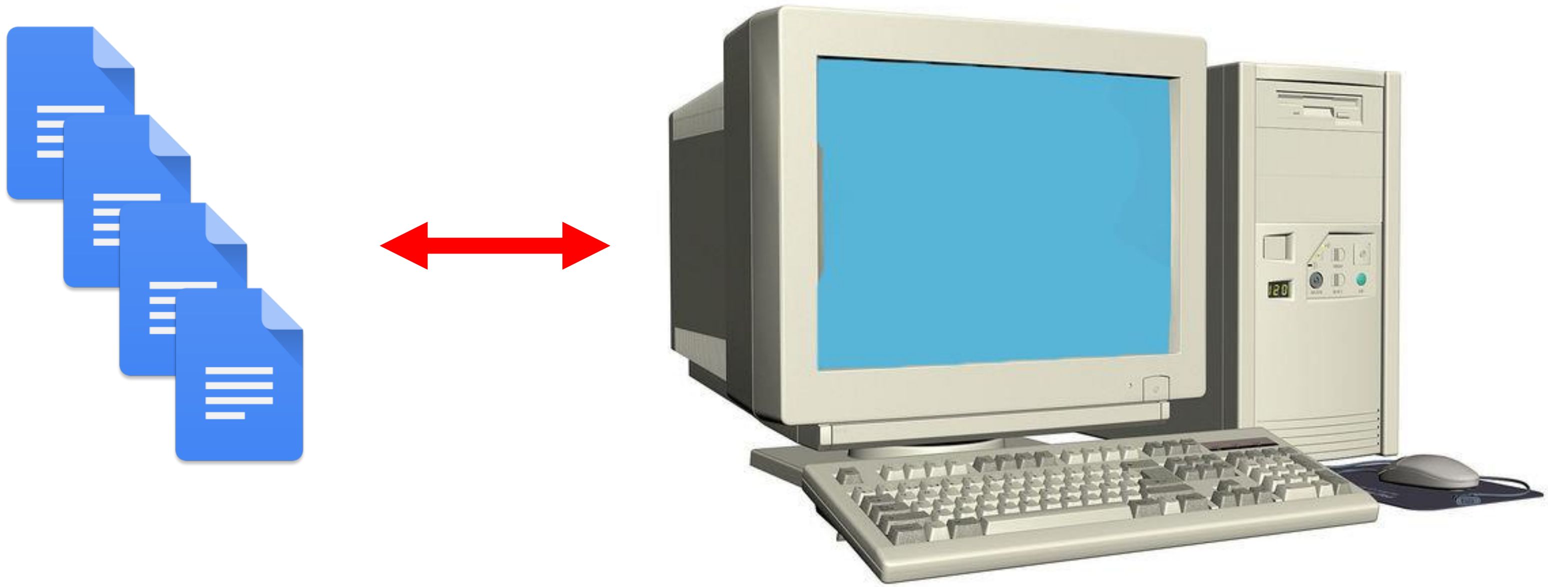
Attribute

Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

Table, Relation

Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

Query



Querying Database

- What is Emily’s GPA?

Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

Table Name: Students

Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

Querying Database

Select GPA
  from Students
where
  name = ‘Emily’

Name	E-mail	Age	GPA
Jones	jo@jo.ca	19	3.4
Emily	em@em.ca	20	3.5

Design Problems

- What if there are more than one “Emily” in the table?
  - Assign a unique ID to each entity.
- What if we need more information about Emily recorded in other files and tables?
  - Use the same unique ID across tables.

Students

ID	Name	E-mail	Age	GPA
1	Jones	jo@jo.ca	19	3.4
2	Emily	em@em.ca	20	3.5

Family

ID	Name	Father	Country	Student_ID
1	Steve	Paul	Canada	200
2	Emily	Pete	Sweden	2

More Problems?

- What if the data in so large and cannot be place in the memory?
- What if we delete ‘Emily’ from one table? Should it be removed from the other ones? How?
- etc.

What Is a DBMS?

- A *Database Management System (DBMS)* is a software package designed to store and manage databases.

Files vs. DBMS (Example Scenario)

- A company has a large collection (say, 500 GB) of data on employees, departments, products, sales, and so on.
- This data is accessed concurrently by several employees.
- Questions about the data must be answered quickly, changes made to the data by different users must be applied consistently, and access to certain parts of the data (e.g., salaries) must be restricted.

Storing Data as Files Drawbacks

- We probably do not have 500 GB of main memory to hold all the data. We must therefore store data in a storage device such as a disk or tape and bring relevant parts into main memory for processing as needed.
- Even if we have 500 GB of main memory, on computer systems with 32-bit addressing, we cannot refer directly to more than about 4 GB of data. We have to program some method of identifying all data items.

Storing Data as Files Drawbacks

- We have to write special programs to answer each question a user may want to ask about the data. These programs are likely to be complex because of the large volume of data to be searched.
- We must protect the data from inconsistent changes made by different users accessing the data concurrently. If applications must address the details of such concurrent access, this adds greatly to their complexity.

Storing Data as Files Drawbacks

- We must ensure that data is restored to a consistent state if the system crashes while changes are being made.
- Operating systems provide only a password mechanism for security. This is not sufficiently flexible to enforce security policies in which different users have permission to access different subsets of the data

Files vs. DBMS

- Application must stage large datasets between main memory and secondary storage
  - (e.g., buffering, 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

Why a DBMS?

- ✓ Data independence
- ✓ Efficient data access
- ✓ Data integrity and security
- ✓ Uniform data administration
- ✓ Concurrent access, recovery from crashes
- ✓ Reduced application development time

Data Independence

- Application programs should not, ideally, be exposed to details of data representation and storage, The DBMS provides an abstract view of the data that hides such details.

Efficient Data Access

- A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.

Data Integrity and Security

- If data is always accessed through the DBMS, the DBMS can enforce integrity constraints. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded.
- Also, it can enforce *access control* that govern what data is visible to different classes of users.

Data Administration

- When several users share the data, centralizing the administration of data can offer significant improvements.
- Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and for fine-tuning the storage of the data to make retrieval efficient.

Concurrent Access, Crash Recovery

- A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.

Reduced Application Development Time

- The DBMS supports important functions that are common to many applications accessing data in the DBMS.

By Studying Databases...

- Shift from computation to information
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, EOS project
  - ... need for DBMS exploding
- DBMS encompasses most of CS
  - OS, languages, theory, AI, multimedia, logic

Data Models

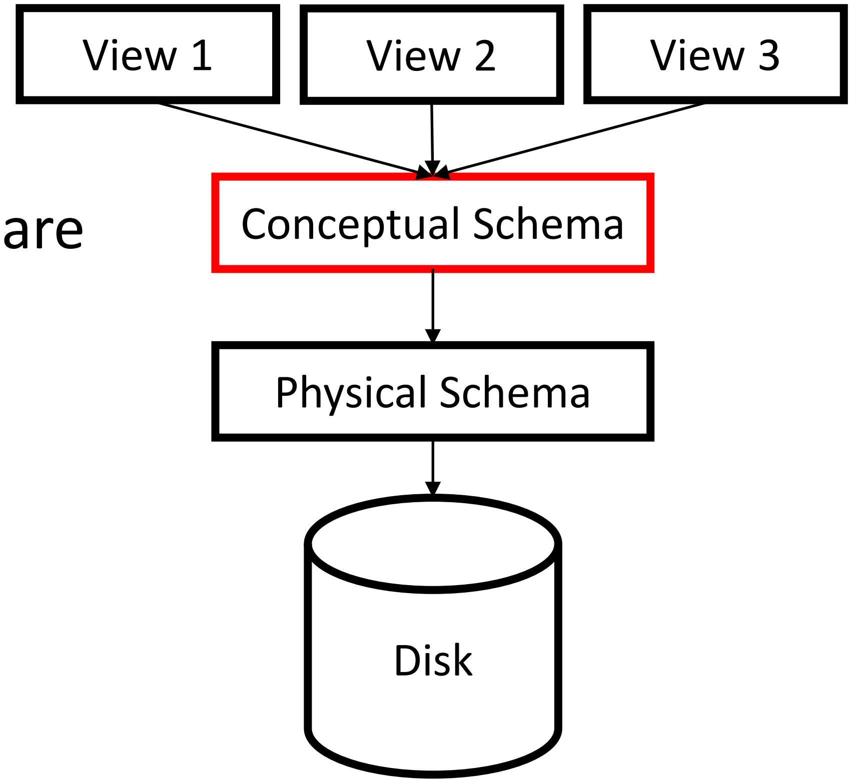
- A data model is a collection of concepts for describing data.
- A schema is a description of a particular collection of data, using the a given data model.
- The relational model of data is the most widely used model today.
  - Main concept: relation, basically a table with rows and columns.
  - Every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

ID	Name	E-mail	Age	GPA
1	Jones	jo@jo.ca	19	3.4
2	Emily	em@em.ca	20	3.5
3	Emily	ly@ly.ca	32	2.3

- Schema
  - Students (ID: integer, name: string, e-mail: string, age: integer, GPA: real)

Levels of Abstraction

- Conceptual Schema
  - Defines logical structure
  - Describes all relations that are stored in the database



Levels of Abstraction

- Conceptual Schema

  Students (ID: string, name: string, e-mail: string, age: integer, GPA: real)

  Faculty (ID: string, fname: string, sal: real)

  Courses (ID: string, cname: string, credits: integer)

  Rooms (nw: integer, address: string, capacity: integer)

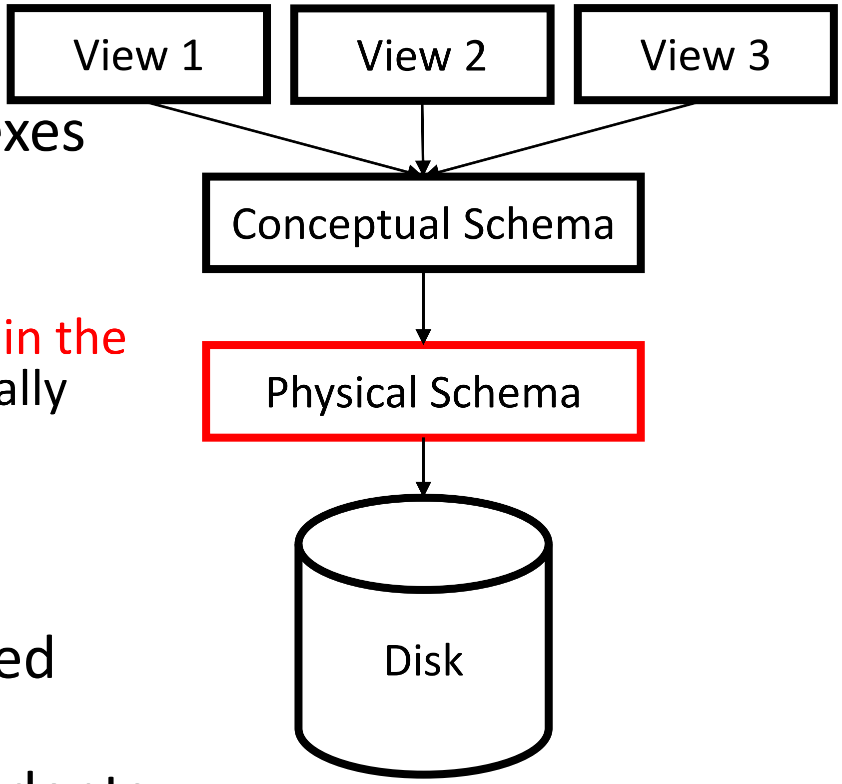
  Enrolled (ID: string, cid: string, grade: string)

  Teaches (ID: string, cid: string)

  Meets_In (cid: string, rno: integer, time: string)

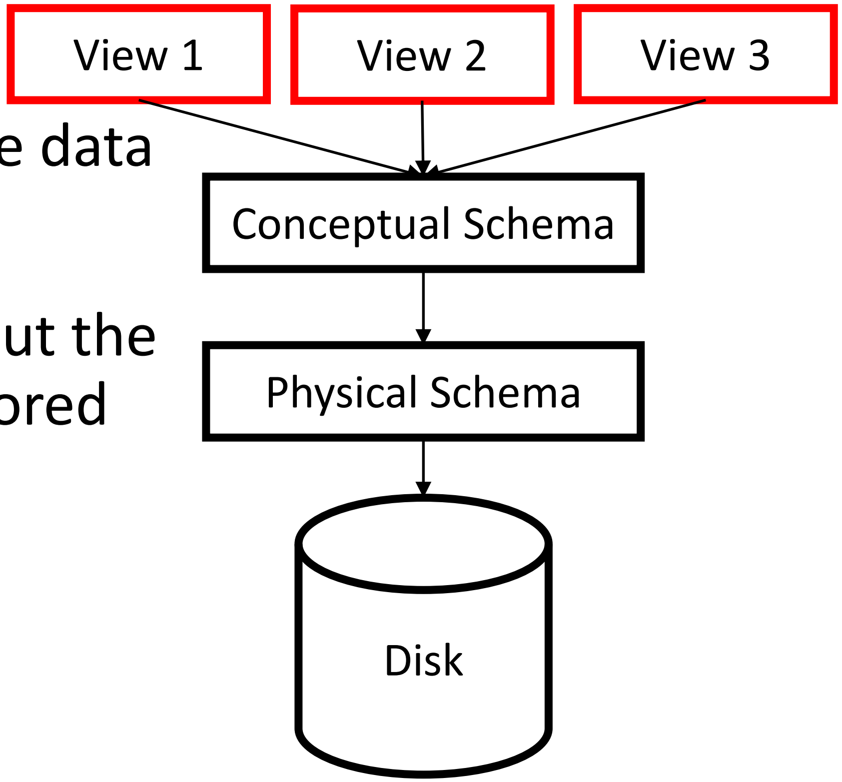
Levels of Abstraction

- Physical Schema
  - Describes the files and indexes used
  - i.e.,
    - how the relations described in the conceptual schema are actually stored on secondary storage devices, such as disks.
- e.g.,
  - relations stored as unordered files.
  - Index on first column of Students



Levels of Abstraction

- External Schema
  - Describes how users see the data
- View
  - Is conceptually a relation, but the records in a view are not stored in the DBMS.



ID	Name	E-mail
1	Jones	jo@jo.ca
2	Emily	em@em.ca
3	Emily	ly@ly.ca

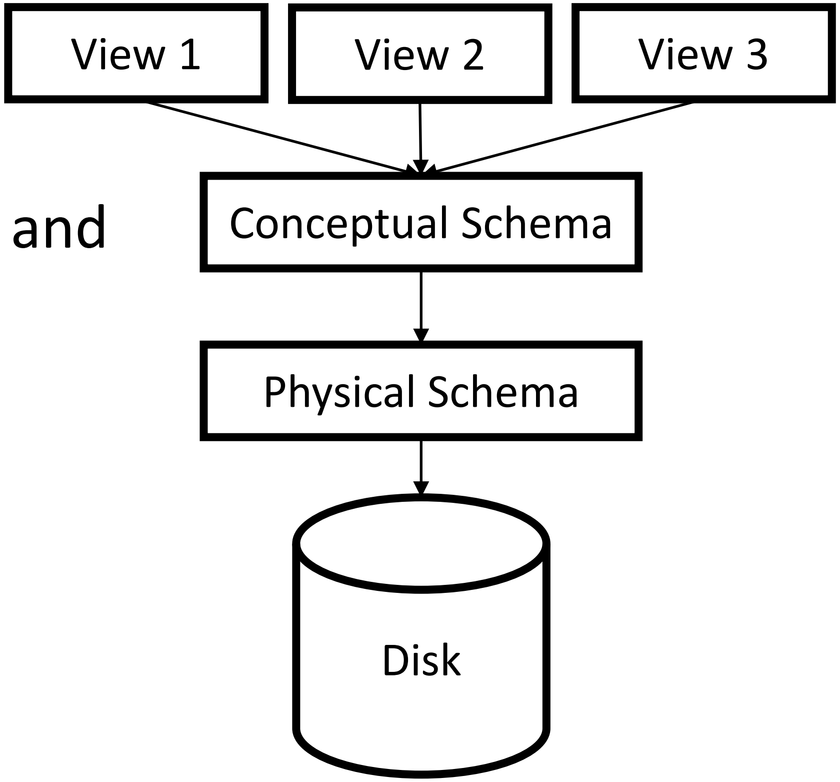
- Schema
  - Students_view1 (ID: integer, name: string, e-mail: string)

ID	Name	E-mail	isTeenager
1	Jones	jo@jo.ca	True
2	Emily	em@em.ca	FALSE
3	Emily	ly@ly.ca	FALSE

- Schema
  - Students_view2 (ID: integer, name: string, e-mail: string, isTeenager: boolean)

Levels of Abstraction

- DDL
  - Data Definition Language
  - Used to define the external and conceptual schemas



Data Independence

- One of the most important benefits of using a DBMS!
- Applications insulated from 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.

Transaction Management

- Scenario,
  - Consider a database that holds information about airline reservations.
  - When several users access (and possibly modify) a database concurrently, the DBMS must order their requests carefully to avoid conflicts.
  - e.g.,
    - when one travel agent looks up Flight 100 on some given day and finds an empty seat
    - another travel agent may simultaneously be making a reservation for that seat

Concurrency Control

- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow, it is important to keep the CPU humming by working on several user programs concurrently.
- 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.

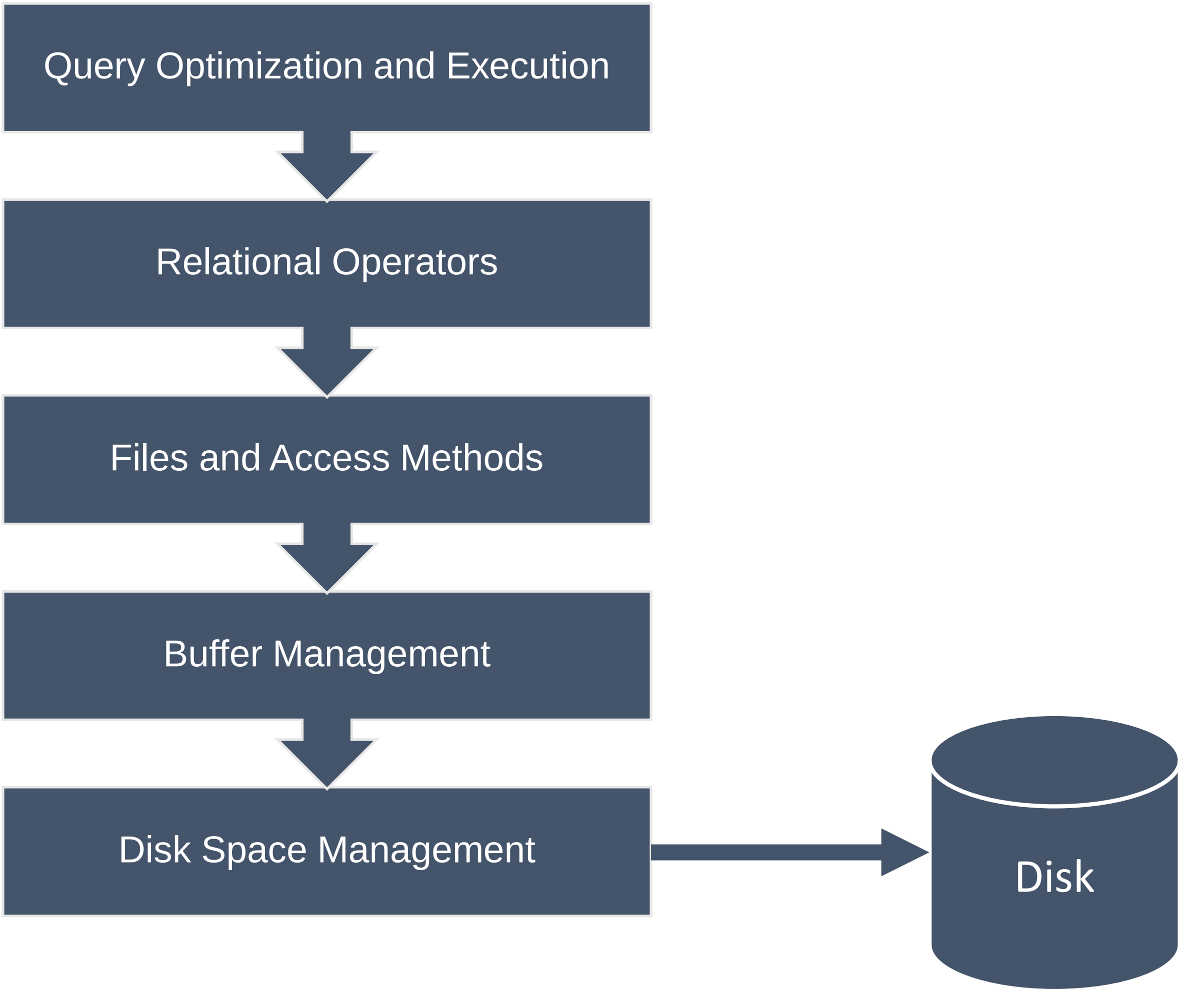
Transaction

- Transaction 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.
    - e.g., it does not understand how the interest on a bank account is computed.
  - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user's responsibility!

A Solution: Database Lock

- Consider two transactions *T1* and *T2* such that
  - *T1* wants to modify a data object
  - *T2* wants to read the same object
- Intuitively, if *T1's* request for an exclusive lock on the object is granted first, *T2* cannot proceed until *T1* releases this lock, because *T2's* request for a shared lock will not be granted by the DBMS until then.
- Thus, all of *T1's* actions will be completed before any of *T2's* actions are initiated.

Structure of a DBMS



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.