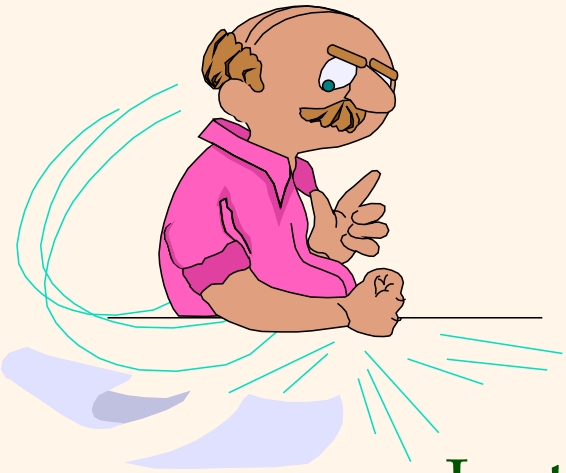


Database System II
Preliminaries

Instructor: Sharma Chakravarthy

sharma@cse.uta.edu

The University of Texas at Arlington

A Few Tips and Suggestions

- Purpose of doing MS or getting a degree and its implications!
  - Get prepared for the next 30+ years of real-world job
- Is your Goal getting A's and a GPA of 4.0? Or learning to evolve, adapt, and deal with whatever comes your way
  - A good GPA does not necessarily mean you are prepared for the job!
    - Answering test questions is different from solving real-world problems!
  - Industry needs problem solvers; not people who can memorize and answer tests and do well on exams
  - Key is: Independent analytical thinking & ability to apply theory/principles to practical problems that you might have not seen before
- You can do BOTH: by understanding the concepts and intuitions/insights behind them
  - And how to think on your toes! Critically needed to ace interviews
- You know how the interview process has changed over the years and what the employers are looking for!
  - Think about how to Succeed in your Career! And learn how to enjoy what you do along the way!
  - Understand the tradeoff between confidence and correctness in interviews!

A Few Tips and Suggestions (2)

- Rather than thinking this as a course in which you learn ONLY DBMS systems details
- Think of various concepts you are learning and how it is relevant to problem solving and data analysis in general.
- Think of
  - Technologies (skill sets) you are acquiring
  - Underpinnings of those technologies
  - How, why, and where they can be used
  - How to match tools to the problem at hand!

- Most people get lost in theory, but never understand where and when it is appropriate and how to apply what you have learnt
- **Why, what, and how** are important in that order
  - **Why** do you want to use this specific approach or solution
  - **What** is the context and can you quantify the relevance
  - **How** to do it in the best possible way
- If you master the above three, you are in business

Preparing for your Career

- You must be aware of how our lives (especially your careers) are going to change in the next 30 to 40 years due to progress in Computer Science (including AI)
- You are not going to replace robots and AI!
- Now, you must also deal with ChatGPT competing with you!

- Robots and AI are here to stay, and you need to get prepared for that!
  - E.g., RazorSQL, tuning tools (both reduced the need for developers and DBAs)
  - ChatGPT can answer a lot of questions, even write code, and will challenge you every step of the way in your career!

- You must figure out a way to SURVIVE AI and automation!

- You need to become good at what computers does not or cannot do
  - Analysis, design, problem solving, independent thinking!
  - Generating new ideas based on needs
  - Matching solutions to problems, thinking in a broader perspective!
  - Not just coding, running a program/package, and generating output/results
- Hence, you need to continuously update your skill sets to stay afloat and figure out how to survive
- I am NOT trying to scare you! you need to be prepared for the reality

This is what I want to accomplish

- Hello Professor,
-
- How are you? I hope you are doing well in this pandemic and are safe. I am doing well and keeping myself safe. It's been more than a year working at Discover and it's been going really well. I have been solving many Natural Language related problems here and learning a lot in the process. **I always remember one thing out of so many things I learnt from you is, anybody can code given an algorithm, but coming up with an algorithm or idea or way of solving a complex problem is important. This has helped me many times. Thank you very much.**
-
- Many students who graduated in 2019 and 2020 took grad walks this year, but I could not do so. I always wanted my parents to attend my grad walk and also meet you at that time, which did not happen. Definitely we would want to meet you whenever my parents are here.
-
- Hoping to hear from you soon.

- Dated 5/16/2021

Applications

- Computation intensive applications
  - E. g., finite element analysis, Computer aided design, wind tunnel testing, simulations, numerical analysis
- Data intensive applications (DBMSs, Data Analytics & Science)
  - E. g., library of congress book management, Walmart point of sales database, Amazon inventory database, airline reservations, Homeland security database and applications, inverted index used by Google, Edgar database. Patent database
- Computation & data intensive applications
  - Data mining, drill down and exploration of data warehouse, dimensional analysis, Google search, stream/sensor data processing, scientific computations…
  - Mining very large volumes of unstructured data: cloud computing, NoSQL processing, Map/Reduce paradigm
  - Neural network algorithms, machine learning

Applications (contd.)

- We are familiar with CPU intensive applications. They keep most/all data in main memory for processing (Java projects, data structures projects).
- The files you used were very small! AND could be stored in main memory!
- We know how to design these algorithms and applications (algorithms and software engineering)
- Design and implementation of data intensive applications is VERY different from main memory applications.
- We deal with designing a database in cse 3330/5330
  - What is the equivalent of UML for data modeling?
  - In fact, EER (1976) came way earlier than UML (Unified Modeling Language) (1994)!

Applications (contd.)

- What applications are likely to be data intensive?
  - Online shopping: Netflix user rentals, Amazon, ebay, …
  - Online information: Google, Yahoo, Wiki, …
  - Online services: credit cards, banking, …
  - Others: library book search, comparative shopping, EDGAR database, Facebook database, …
  - Enterprise databases: …

- For these application, we not only have to design the computational part, but we have to design the data to be represented and managed efficiently on **non-volatile** or persistent storage (or disks)!

Applications (Contd.)

- In CPU intensive applications, representation of data was assumed to be simpler of the two (algorithm vs. data storage)

- The choice was in the identification of one of the main-memory data structures (arrays, lists, trees, hash tables, linked lists, etc.)
- You load the data **once.** You may write files as output. **But you do not go back and forth!**

- The focus was on the main-memory algorithm, its correctness and efficiency (or complexity)
- The compiler did a lot of your code optimization!

In Data intensive Applications

- We need to stage data back and forth between persistent storage and memory. (Why?)
- Hence, we focus on DBMS architecture and its components
  - Storage and indexing
    - Buffer Management
  - Transaction management
    - Concurrency control and Recovery (second project)
  - Cloud computing and Map/Reduce (third project)
  - Query Optimization (Homework for grade)
- Projects will be implemented in a realistic setting
- We will also look at new approaches, such as
    - Non-relational or NoSQL DBMSs (time permitting)
    - Their relationship with the others

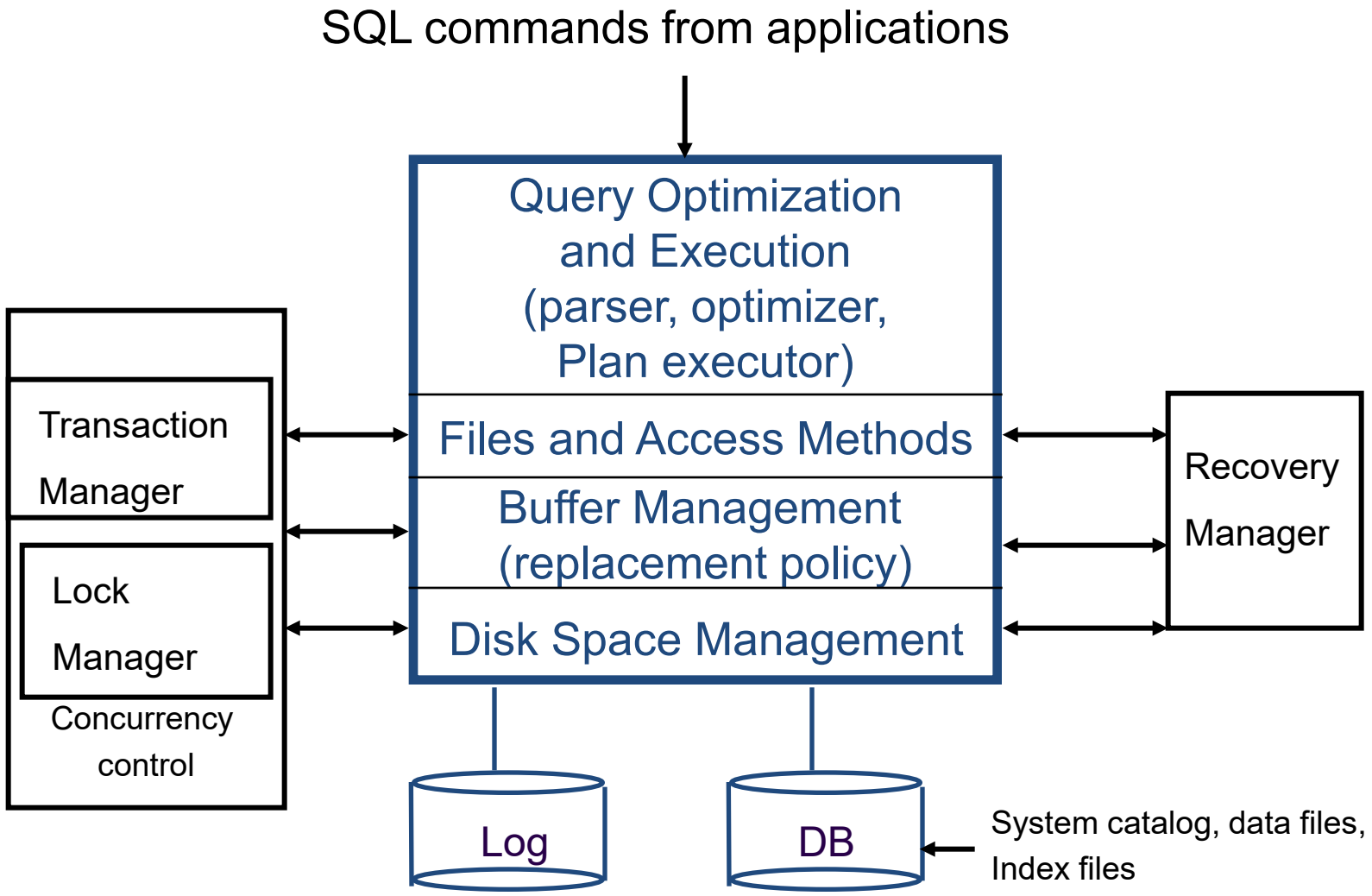
What Is a DBMS?

- Briefly, A very large, integrated, persistent collection of data.

- Models a real-world *enterprise.*
  - Entities
    - e.g., students, courses, buildings, websites, products
  - Relationships
    - e.g., Billy Joel performs at UTA, students take courses, lenders issue mortgages, an employee manages a department, students submit projects

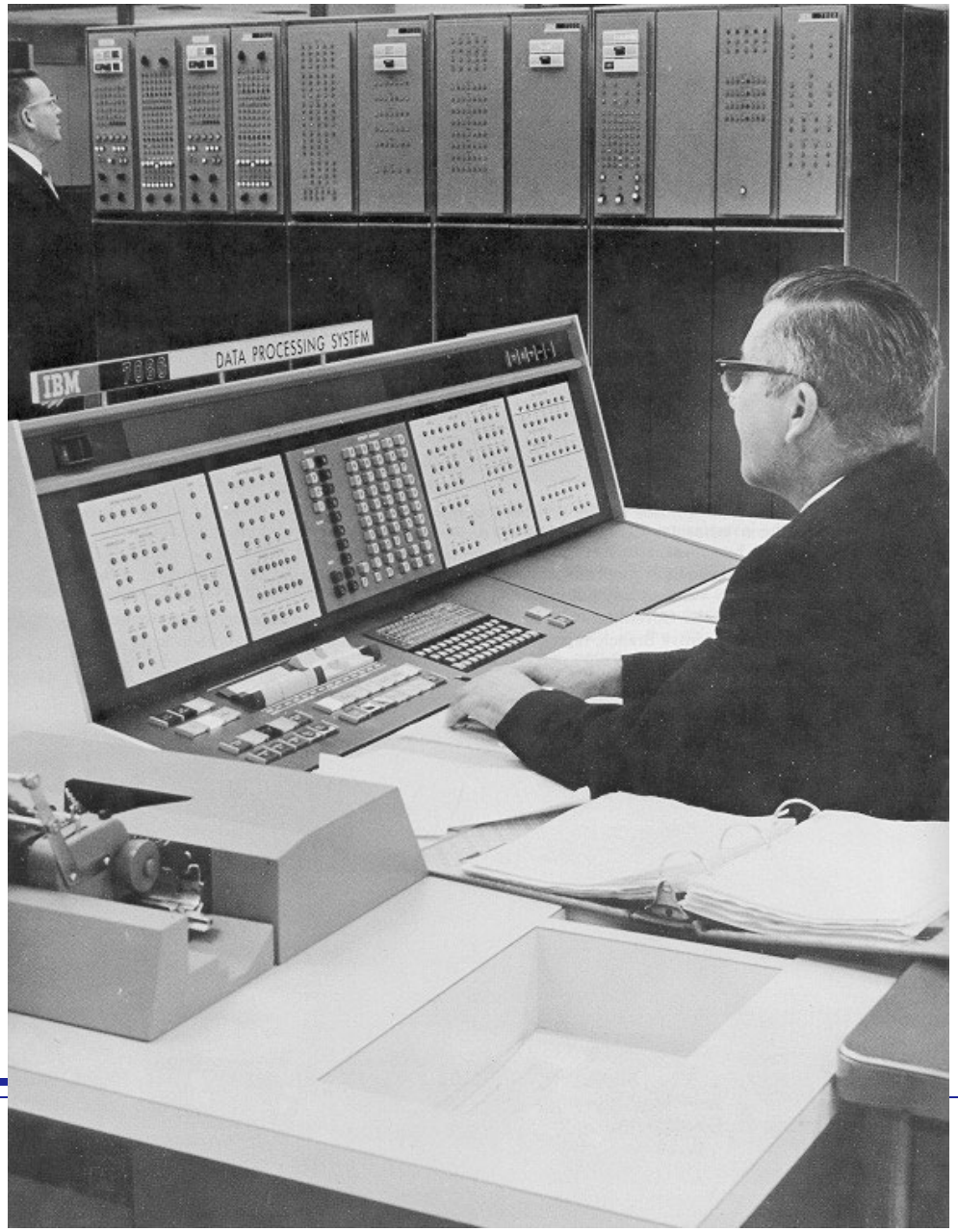
- A *Database Management System (DBMS)* is a software system designed to store, access, and facilitate access to relevant contents (i.e., databases)

DBMS overall Architecture



The University of Texas ARLINGTON

Computer Systems: Then



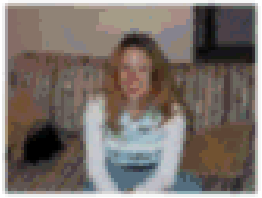
INFORMATION TECHNOLOGY LABORATORY @ UTA-CSE

Database Systems: Today

Show people who are:

- [ ] Male
- [x] Female
- [ ] Unknown gender
- [x] Single
- [ ] In a Relationship
- [ ] Married
- [ ] Open Marriage
- [ ] Unknown status

People matches: **1 - 20 of 203** 1 2 3 4 5 6 7 ... 11 Next

Wendy	Brenda	Melissa
		

From:	 Jonathan You are connected to Jonathan through: You ⇔ Melissa ⇔ Jonathan
Date:	September 1, 2002 3:30 PM
Subject:	Hello!
Message:	Hi Cindy, I'm a friend of Melissa's, and I like your profile. What kind of teaching do you do? Maybe we can play tennis sometime... Jonathan

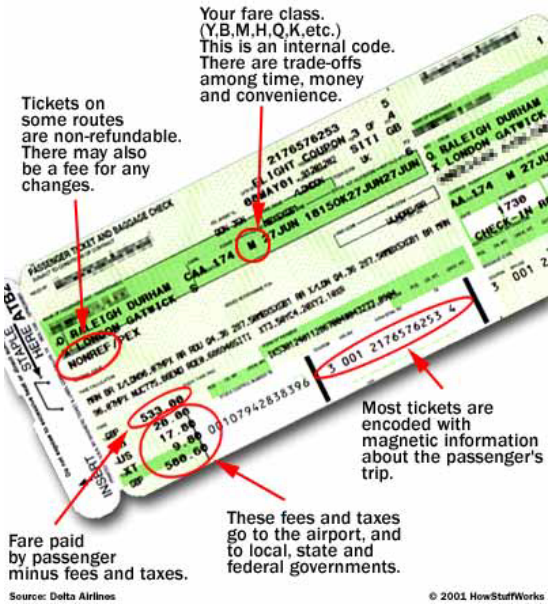
From Friendster.com on-line tour

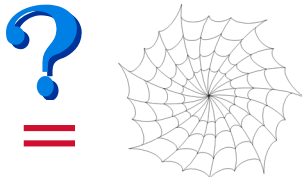
Other Ways Databases Make Life Better?

- **"Players could finally sign up for the Star Wars Galaxies game last week as Sony opened up registration to the public."**
- **"Once players got in to the game they found that the game servers were offline because of database problems."**
- **"Some players spent hours tuning their in-game characters only to find that crashes deleted all their hard work."**
- **Source: BBC News Online, July 1, 2003.**

Other databases you may use

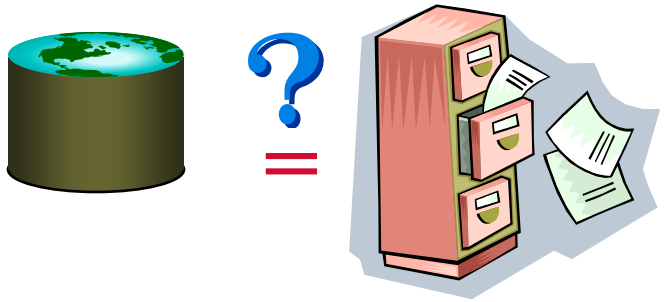
amazon.com



Is the WWW a DBMS?

- **Fairly sophisticated search available**
  - crawler *indexes* pages on the web
  - Keyword-based search for pages
- **But, currently**
  - data is mostly unstructured and untyped
  - search only:
    - can't modify the data
    - can't get summaries, complex combinations of data
  - few guarantees provided for freshness of data, consistency across data items, fault tolerance, …
  - Web sites typically have a DBMS in the **background** to provide these functions.
- **The picture is changing**
  - New standards e.g., XML, Semantic Web can help data modeling
  - Research groups (e.g., at Berkeley) are working on providing some of this functionality *across multiple web sites.*



Is a File System a DBMS?

- **Thought Experiment 1:**
  - You and your project partner are editing the same file.
  - You both save it at the same time.
  - Whose changes survive?

A) Yours B) Partner's C) Both D) Neither E) ???

- **Thought Experiment 2:**
  - You're updating a file.
  - The power goes out.
  - Which of your changes survive?

A) All B) None C) All Since Last Save D) ???

Q: How do you write programs over a subsystem when it promises you only "???" ?

A: Very, very carefully!!

Why Use a DBMS?

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.
- Persistence, scalability, portability

Why Study Databases??

- Shift from *computation* to *information management to* Big Data analytics
  - at the "low end": ad hoc data, spread sheet
  - at the "high end": scientific data management
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, EOS project, multi-media dbms
  - ... need for DBMS is exploding
- DBMS study encompasses all aspects of CS
  - OS, languages, theory, AI, multimedia, logic

Current Commercial Outlook

- A major part of the software industry:
  - Oracle, IBM, Microsoft, Sybase
  - Also, Informix (now IBM), Teradata
  - smaller players: java-based dbms, devices, OO, …
- Well-known benchmarks (esp. TPC)
- Lots of related industries
  - data warehouse, document management, storage, backup, reporting, business intelligence, app integration
- Relational products dominant and evolving
  - adapting for extensibility (user-defined types), adding native XML support.
- Open Source coming on strong
  - MySQL, PostgreSQL, BerkeleyDB (has been acquired by Oracle)

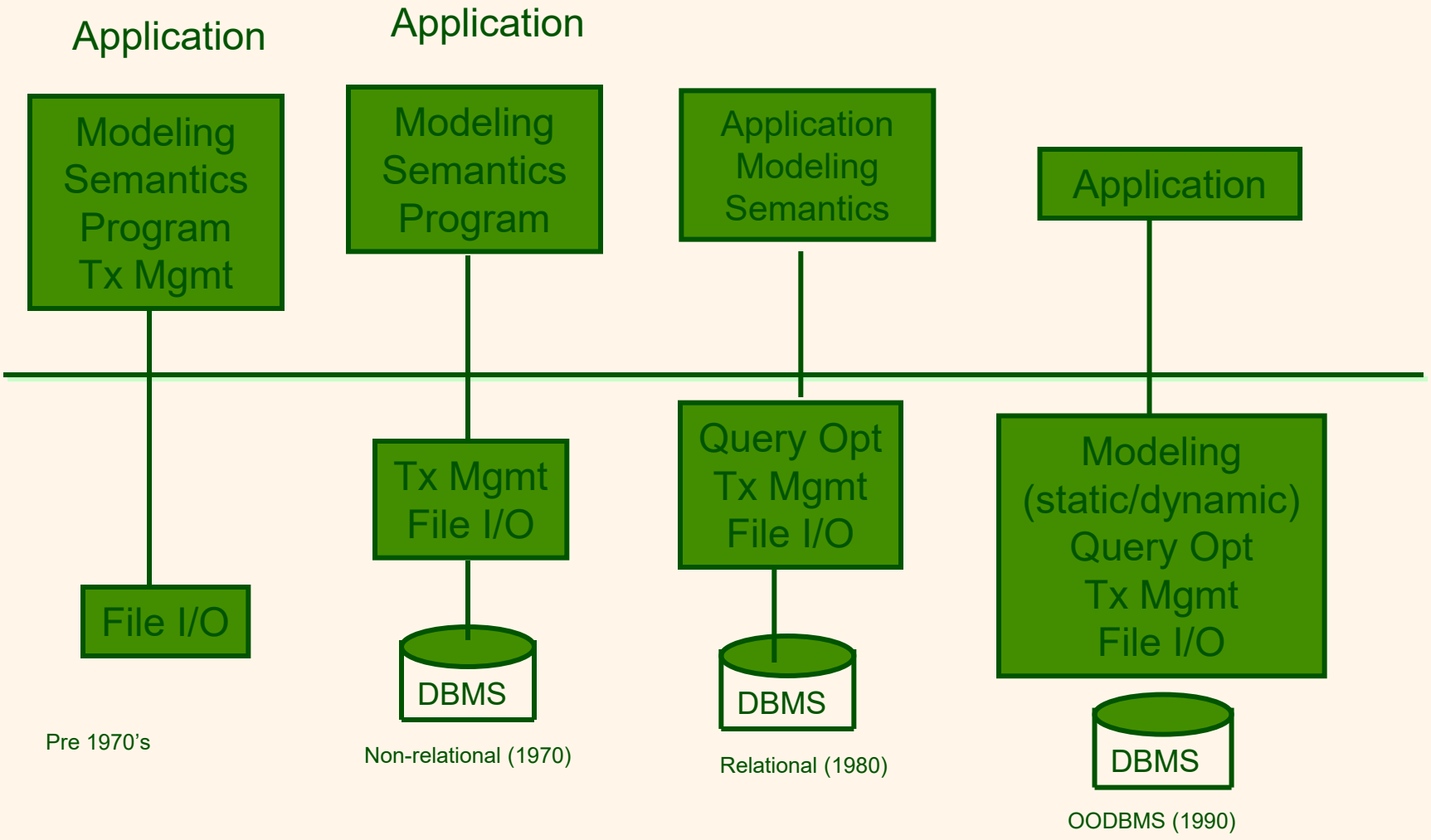
What's the intellectual content?

- representing information
  - data modeling
- languages and systems for querying data
  - complex queries with real *semantics**
  - over massive data sets
- *concurrency control* for data manipulation
  - controlling concurrent access
  - ensuring *transactional semantics*
- reliable data storage (recovery)
  - maintain data semantics even if you **pull the plug**

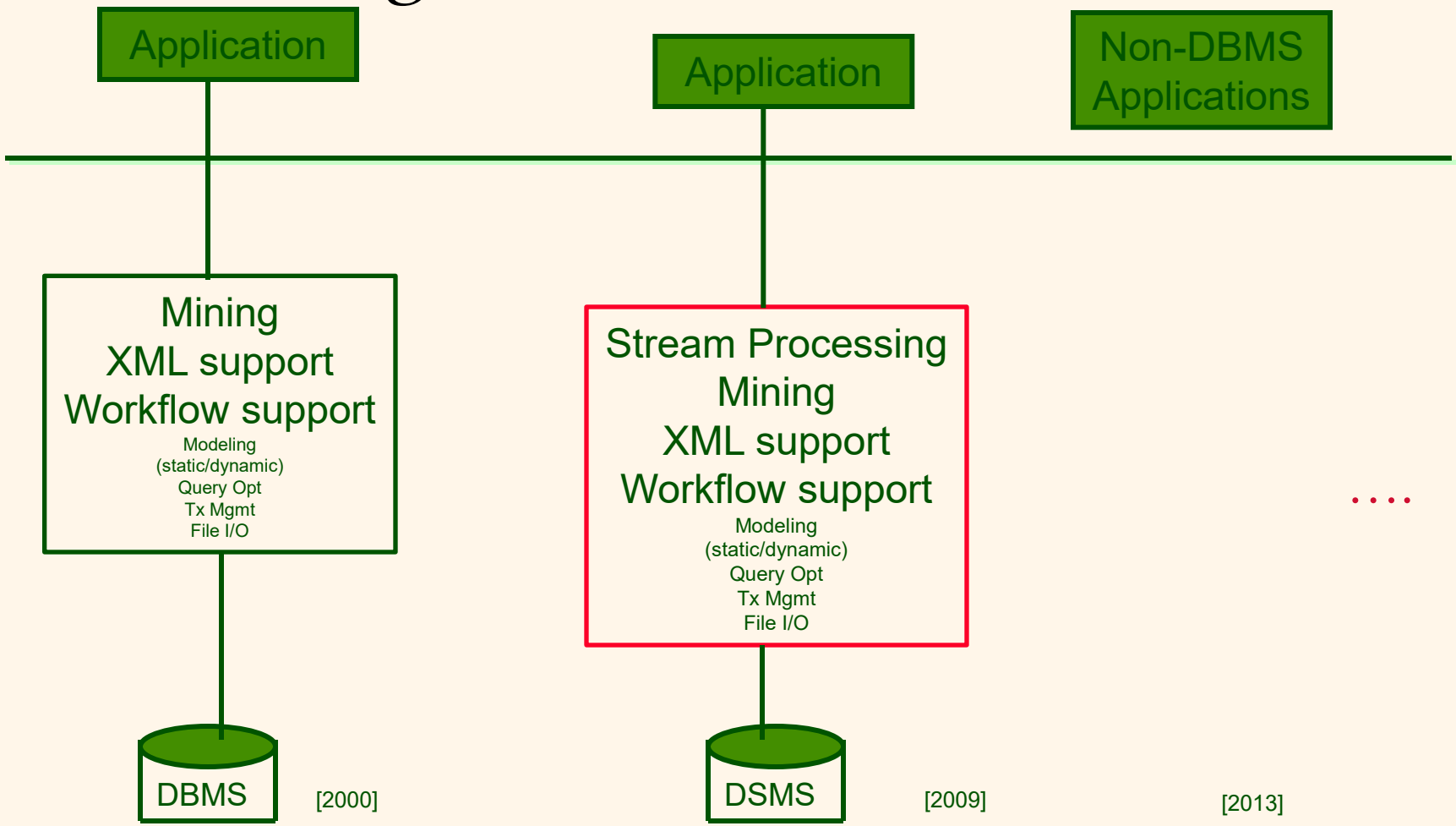
* semantics: the meaning or relationship of meanings of a sign or set of signs

[Adapted from Peter Lyngbaek:OOPSLA:1991]

Evolution of DBMSs

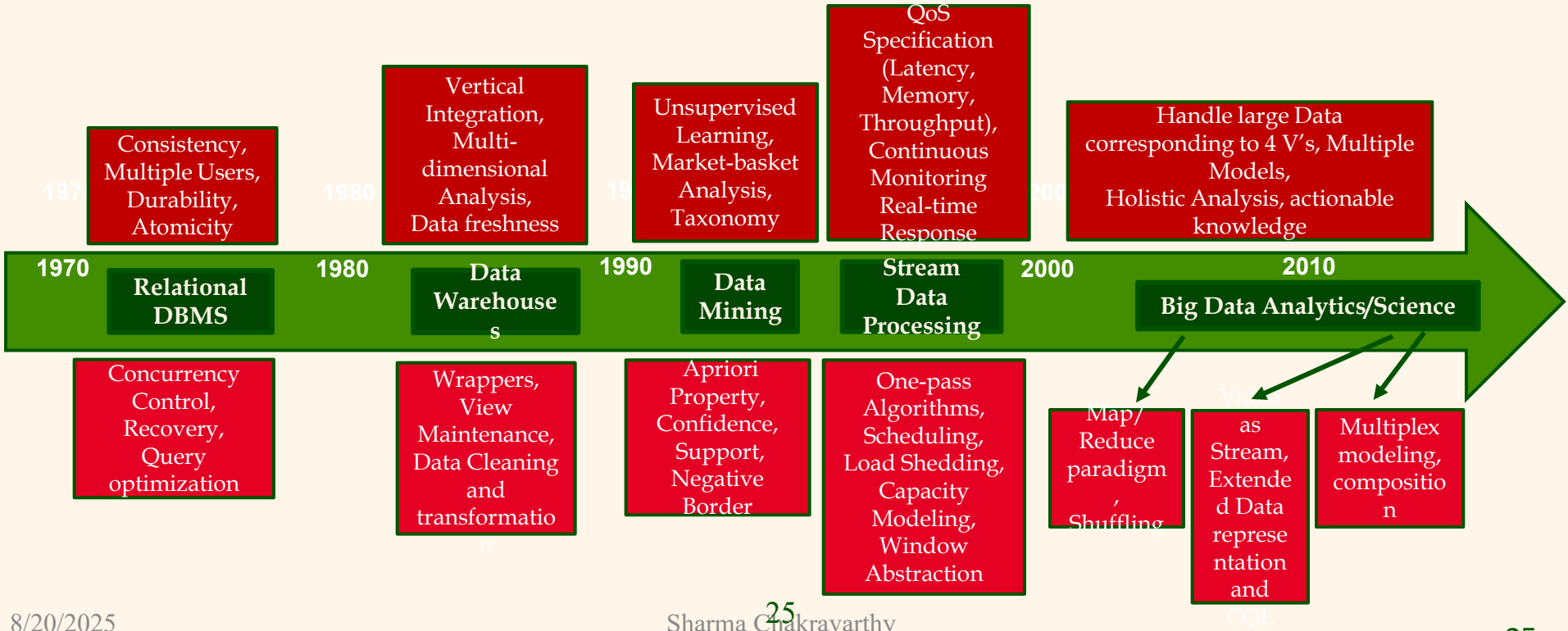


Moving On …


Application
Application
Non-DBMS Applications
Mining
XML support
Workflow support
Modeling (static/dynamic)
Query Opt
Tx Mgmt
File I/O
Stream Processing
Mining
XML support
Workflow support
Modeling (static/dynamic)
Query Opt
Tx Mgmt
File I/O
. . . .
DBMS
[2000]
DSMS
[2009]
[2013]

Where are we headed?

- **Without understanding the past, it is very difficult to appreciate the present and plan for the future!**
- **Technology provides solutions; it does NOT necessarily solve problems!**



Impedance mismatch

- If you are studying DBMS, you have to understand this very clearly
- CPU is way faster than disk access. In fact, it is a Million or more times faster
  - CPU accesses are in nano secs and disk accesses are still in milli secs!
- Because of this speed difference, it takes much longer to bring data from a disk
- So, CPU is waiting/idling most of the times for data (if not architected properly)!

- This is called impedance mismatch!

Performance Issues

- There are many sources of performance problems
  - Improving existing abstraction/adding new functionality
  - Increase in data volume without any other changes
  - Or both of the above!
- In the absence of the above, we can continue to improve performance (improvements are offset by the increase in data or additional software needed for supporting the abstraction)

Solutions

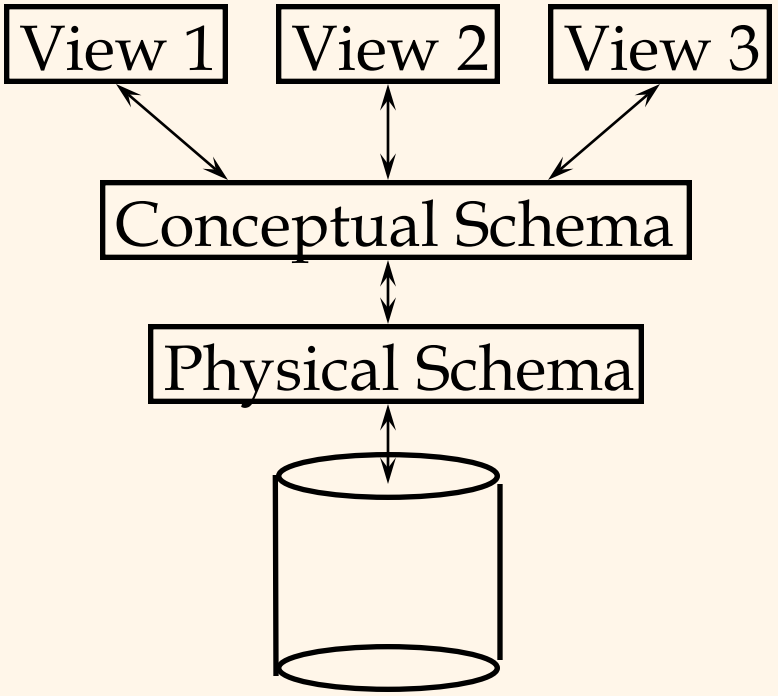
- There are several solutions
  - Doubling of CPU speed every 12 months or sooner (Moore's Law) Have you heard of Neuromorphic computing?
  - Increase in disk i/o throughput
  - Use of multiple processors (Map/Reduce leverages this on a large scale)
- The above CANNOT completely address the problems mentioned earlier
- New algorithms/techniques, optimization are critical (e.g., data structures, indexes, concurrent processing, …)
- The introduction of SSDs (solid state disks) are warranting a re-examination of some of the above issues
  - No impedance mismatch!!

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.

Levels of Abstraction

- Many *views*,
- single *conceptual (logical) schema* and *physical schema*.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.



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

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)*
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):
  - *Course_info(cid:string,enrollment:integer)*

Data Independence

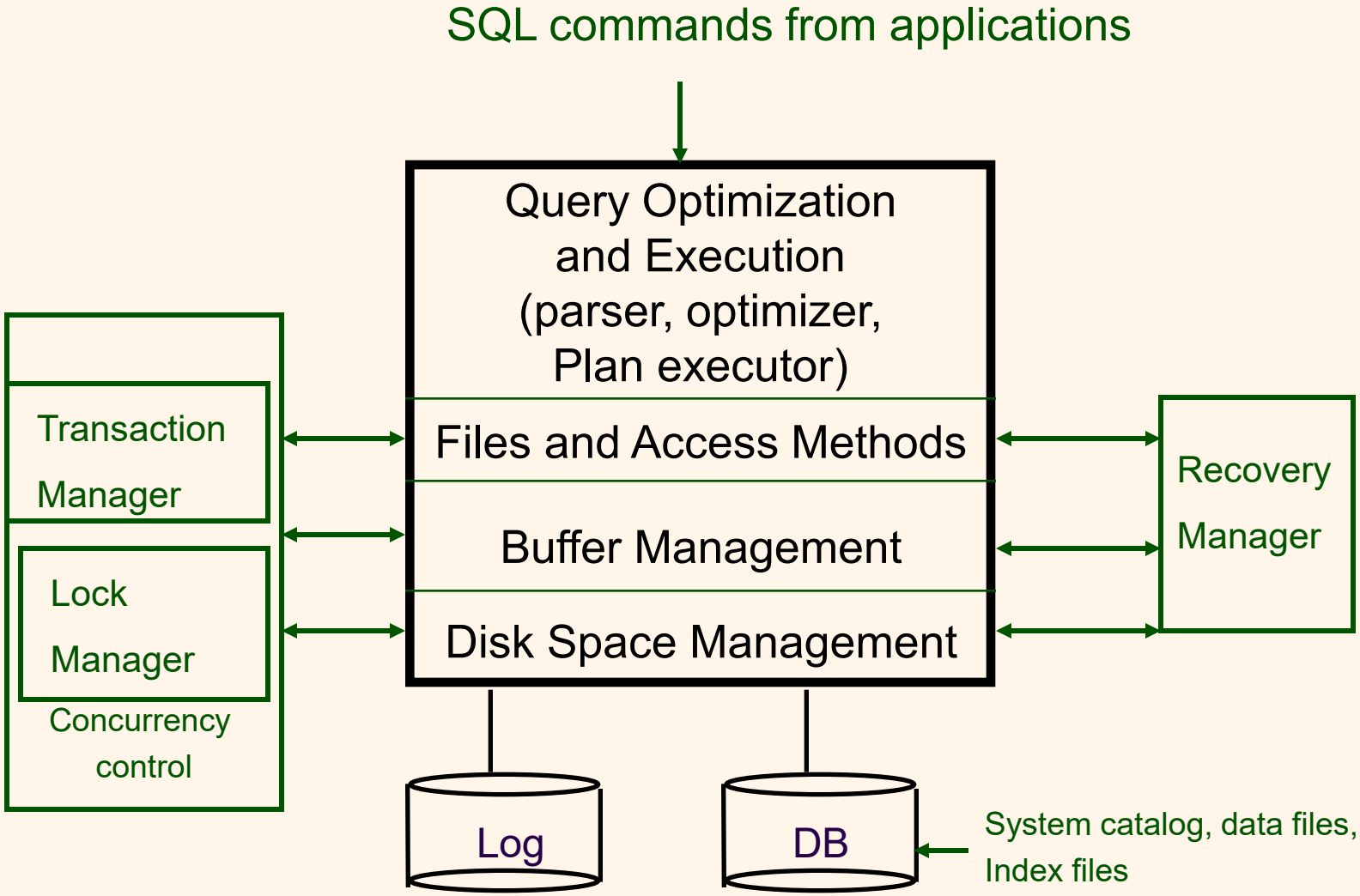
- Applications insulated from how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data.
- *Physical data independence*: Protection from changes in *physical* structure of data.

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

DBMS Components

- Architecture – typically client/server
- DDL, DML parsing, processing
- Physical DB design (files, indexing, access methods)
- Query processing, optimization
- Buffer management
- Concurrency control
- Recovery
- Utilities – backup, archiving, exporting, OLAP, report generator, physical design wizards, access to multiple DBMSs, designer tools, tuning of parameters, etc. etc.

DBMS Architecture



Query Processing

- Need an expressive Query Language over a simple data model
- Non-procedural query language
  - Burden of optimization on the system (not on the user as in hierarchical and network databases)
  - Query representation should not affect query optimization
- Need for report generation

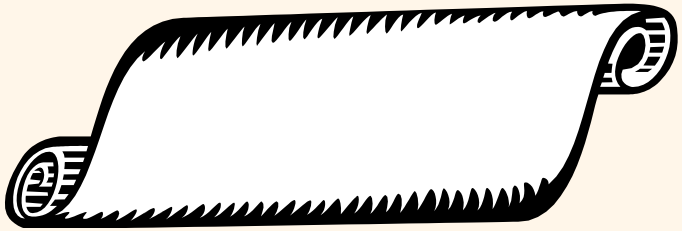
Concurrency Control

- 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 do not arise: users can pretend they are using a single-user system.

Transaction: An Execution of a DB Program

- Key concept is *transaction*, which is an *atomic* (either all or nothing) 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.

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 Xact id, so it's easy to undo a specific Xact (e.g., to resolve a deadlock).
- Log is often *archived* on "stable" storage.
- All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.

Databases make these folks happy ...

- End users and DBMS vendors
- DB application programmers
  - E.g. smart webmasters
- *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!

Why Study the Relational Model?

- Most widely used model.
  - Vendors: IBM (Informix), Microsoft, Oracle, Sybase, etc.
- "Legacy systems" in older models
  - e.g., IBM's IMS
  - Network models are not used today
- More recently, NoSQL systems
  - Based on different data models
  - Key-store, document store, graph
  - Tradeoffs in functionality ACID vs. CAP (or BASE)

Competitors

- Earlier Competitor: Object-Oriented model
  - ObjectStore, Versant, Ontos
  - a synthesis: *object-relational model*
    - Informix Universal Server, UniSQL, O2, Oracle

- More recent competitor: Map/reduce, NoSQL
  - Works on unstructured data
  - Useful for one-time processing as opposed to data management
  - Overhead is low, however, programming level is high
  - Less expensive, use of commodity machines!!

Relational Database: Definitions

- *Relational database:* a set of *relations*.
- *Relation:* made up of 2 parts:
  - *Instance* : a *table*, with rows and columns. #rows = *cardinality*, #fields = *degree / arity*
  - *Schema* : specifies name of relation, plus name and type of each column.
    - E.g. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)
- Can think of a relation as a *set* of rows or *tuples*. (i.e., all rows are distinct)

Example Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Cardinality = 3, degree = 5 , all rows distinct
- Do all columns in a relation instance have to be distinct?

Creating Relations in SQL

- Creates the Students relation. Observe that the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.
- As another example, the Enrolled table holds information about courses that students take.

```
CREATE TABLE Students
        (sid: CHAR(20),
         name: CHAR(20),
         login: CHAR(10),
         age: INTEGER,
         gpa: REAL)
```

```
CREATE TABLE Enrolled
        (sid: CHAR(20),
         cid: CHAR(20),
         grade: CHAR(2))
```

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES ('53688', 'Smith', 'smith@ee', 18, 3.2)
```

- Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

- *Powerful variants of these commands are available; more later!*

Integrity Constraints (ICs)

- IC: condition that must be true for *any* instance of the database; e.g., *domain constraints.*
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.
- A *legal* instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances.
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!

Relational Query Languages

- A major strength of the relational model: supports simple, powerful *querying* of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
  - The key: precise semantics for relational queries.
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

The SQL Query Language

- The most widely used relational query language. Current standard is SQL-2011.
- To find all 18 year old students, we can write:

```
SELECT *
FROM Students S
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- To find just names and logins, replace the first line:

```
SELECT S.name, S.login
```

Relational Model: Summary

- A tabular representation of data.
- Simple and intuitive, currently the most widely used.
- Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
  - Two important ICs: primary and foreign keys
  - In addition, we *always* have domain constraints.
- Powerful and natural query languages exist.

Relational Model: Summary (contd.)

- ACID properties are important
  - Atomicity, consistency, isolation, and durability
- Concurrency control and Recovery together guarantee ACID properties

Other data models

- The older hierarchical and network models are not used much (although they are in use)
- However, several non-relational data models have been proposed and currently used in specific contexts
- They are called NoSQL DBMSs
  - Key-value store
  - Document
  - graph

Thank You !