Database Implementation

Let's Begin!

Avraham Leff

Yeshiva University

avraham.leff@yu.edu

COM3563: Fall 2020

Today's Lecture: Overview

1. Course Overview
2. Rules Of The Game
3. Databases: Motivation
4. Relational Databases: What Makes Them Different

Course Overview

What Is This Course About?

- Many universities offer (at least) two database courses
  1. *Introduction to Databases*
     - "What are databases"?
  2. *Database Implementation*
     - "How do you build a database"?
- We have to (at least) combine these two courses into one ☺
- So: let's go the official document ...

COM 3563: "Why?"

> *Databases are essential to every business. All non-trivial applications depend on data, and thus in turn on well-designed databases. A backend systems engineer needs a deep understanding of the challenges and approaches in implementing a database system in order to be able to build high quality server side systems that use databases to their fullest potential. While not all data will be stored in relational databases, all systems that store data must be designed by computer scientists who understand the challenges and issues that relational databases attempt to solve.*

- Q: so, is this course about relational databases or not?
- A: "yes" ☺ ("hold that thought")

COM 3563: "What?"

Database Implementation – COM 3563

Prerequisites: Design & Analysis of Algorithms (COM 2546), Operating Systems (COM 3610)

WHAT:

This course looks at both the use and implementation of relational databases. It starts with a rigorous but fast-paced introduction to the relational model, schemas, indices, views, SQL, ACID properties, and transactions. The course continues with its primary focus - database implementation - revisiting application-level topics as relevant. Specific topics include:

- The relational model, SQL and relational database management systems
- Entity-Relationship models and relational database normalization
- Database system architecture
- Storage organization, access, and buffer management
- Storage data structures
- Query planning and execution
- Transaction management
- Recovery
- Concurrency control.

COM 3563: "Outcomes"

OUTCOMES:

- Students will understand the architecture, conceptual model, and design of relational databases, and how to use the features of a relational database management system (RDBMS).
- Students will be able to construct a conceptual model (E/R diagram) and a corresponding physical model (relational design model) of a database in which they specify the necessary constraints and normalization to satisfy data integrity and operational requirements.
- Students will understand the importance of, and be familiar with at least one implementation approach to, DBMS system features such as storage organization, buffer management, indices, logging, and query planning & optimization.
- Students will understand and be able to apply the principles of transaction processing (atomicity and isolation) to build a small-scale transactional database system.

Getting back to the original question …

- This course is about the use and implementation of relational databases
  1. Use of RDB as clients
  2. Design of RDB tables and data modeling as business analysts and algorithmists
  3. Implementation of RDBs as back-end system engineers
- The course (see syllabus posted on Piazza) will be roughly broken down into these three components
  - In roughly the above order
  - I plan to intersperse lectures designed to help you achieve the project milestones throughout the semester
  - Unfortunately: breaks the nice tripartite structure, but can't be helped ☹

Our discussion so far:

- The course title is misleading ☹
- I'd rephrase as *"how to intelligently use & implement a (relational) database"*
- Skills that I expect you to learn:
  - How to design a (relational) database, including how to construct an *entity-relationship database model*
  - How to normalize a data-set, and the ability to answer the question of "why bother"?
  - How to query a database, and how to query a database *intelligently* to achieve good performance
  - A good understanding of transactional semantics, and different strategies to implement these semantics
  - A good understanding of the difference between "back-end" *versus* "front-end" implementation issues

Relationship to *Modern Data Management*: I

- Q_1: I already took *Modern Data Management*, and we covered RDBs in the first weeks of that course
- A_1: So . . . ☺
- Q_2: will I be able to sleep through this course?
- A_2: you can try, but it probably won't work well for you ☹
- More seriously:
  - For sure: the implementation portion of this course has no overlap with *MDM*
    - Note: while not all data are stored in RDBs, all of the implementation issues must be addressed (one way or the other) by alternative database architectures
  - The "client" portion of this course does overlap with *MDM*
    - This course will go a level deeper (or at least more thoroughly)

Relationship to *Modern Data Management*: II

- Q_3: will I be able to sleep through the relational database material in next semester's *Modern Data Management* course?
- A_{3a}: first, let's get there ☺
- A_{3b}: that's a very good question!

> Discussion on problems induced by the current scheduling of what should be a seamless two-course sequence ...

A Quick Aside On Non-Relational Databases

- When I went to school, relational databases were the "new kid on the block"
  - Network & hierarchical databases were the established players
  - We are not going to discuss those technologies ☺
- Now it's "plus ca change, plus c'est la meme chose"
  - Relational technology is the established player
  - Variety of so-called NOSQL databases have emerged
- I have strong opinions on the subject: but this course will only touch tangentially on this very important topic
- Take *Modern Data Management* if you want to learn more ☺

The Database System For This Course

- We'll be using POSTGRESQL, which you'll install on your laptop
- Free!
- Details in the requirements document for the first assignment (see Piazza)

Textbook

- Required: Database System Concepts, 7_{th} Edition by **Silberschatz, Korth, Sudarshan**
  - Amazon
  - McGraw Hill ISBN-13: 978-0078022159
- I am using the paper edition, "your responsibility" if you get another version (e.g., Kindle edition)
- Note: I did consider using an alternative textbook
  - Decided to go with a "less formal, less mathematics" approach
  - Please let me know (mid-semester or later!) whether you think Silberschatz is "too easy"

How Textbook Will Be Used

- I view my lectures as both an introduction to the textbook material …
  - So: you're required to read thoroughly after the lecture
- And: lectures give an alternative viewpoint or supplement the textbook
- Either way: read the textbook!
- Assignments and exam assume that you've read the textbook …
- Also: (most) non-programming assignments will come from the textbook

Course Is an Evolving Work

- Second time that it's being taught at YU
- Second time that I'm teaching it
- I'll try to minimize the bumps
- But be prepared to be flexible ...

Rules Of The Game

Piazza & Zoom URLs

- Piazza
  - Signup Link: piazza.com/yu/fall2020/com3563
  - Class Link: piazza.com/yu/fall2020/com3563/home
- Zoom: https://yeshiva-university.zoom.us/my/avrahamleff
  - Pass-code: 330709

“Asynchronous Learning” (I)

- Our department has traditionally insisted that students attend class in person, closed laptops & phones, etc
- As you know, we’ll be changing this model this semester: we are teaching remotely and have made provision for asynchronous learning
  - Lectures will be recorded and available for download on your schedule
  - Your call as to whether you attend lecture over Zoom or download the video
- My opinion: you’ll learn better when you (or other students) engage interactively with the Professor (that’s me ☺)
  - But remains your decision …

“Asynchronous Learning” (II)

- IMNSHO there is a serious risk that “remote learning” in general, and “asynchronous learning” in particular will lead to your treating the course less seriously than you should ☹
- Note: it is your responsibility to learn this very important material and to stay “on top” of the assignments (see next slides)
  - I do not plan to relax standards with respect to assignment deadlines!
  - I do not plan to relax standards with respect to assignment quality!
- It is your responsibility to monitor Piazza discussions
- That said: do not hesitate to reach out to me if you think that you’re falling behind!

Juniors & Seniors: Have Your Resume Ready To Go

You may not be aware of this:

> In order to apply for a summer job next year, you must send out your resume in the first week of this semester

- Register with the career office to get relevant notices about on-campus recruitment events
- Work on your LinkedIn profile
- Connect with YU CS alumni on LinkedIn and get their feedback on your profile

Pandemic: Implications For Exams

- I'll be blunt: there is no effective way to administer exams remotely in a manner that enforces academic integrity effectively ☹
- Implication: although there will be a final exam, it will be weighted much less than usual
- That's the good news ☺
- OTOH: assignments & project will be worth more and graded with less leniency
- Key point: important for you to learn the material

Wrapping This Up

- BEH, we will get through this crisis together, and we'll have a successful course this semester!
- (Note: I plan to teach remotely even after the *Yomim Tovim*)

Assignments

Be sure to read carefully! the *Homework Assignments: Requirements & Policies* document (on Piazza)

- There will be a mix of small-scale programming and non-programming assignments
  - You **are required** to typeset your non-programming assignments
  - My recommendation: LaTeX
  - See available downloads for Windows, Mac, and Linux
  - I use text editor + *pdflatex* + **TexShop**
  - See LyX – The Document Processor if you want a graphical interface
  - Or TeXlipse, and Eclipse plugin for LaTeX support
  - Speak to students from "Design & Analysis"
- Use whatever you want: just no hand-written submissions!

Some especially useful “symbol sheets”

- List of LaTeX mathematical symbols
- Relational Algebra Symbols in LaTeX

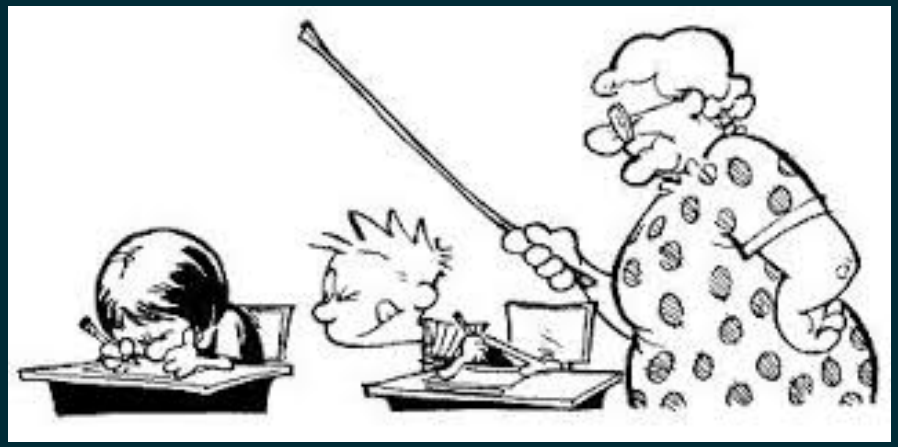
Assignments & “Late Policy”

- Assignments are due (checked into Git) at 10am of the Piazza due date
- `assignmentLate ? 0 : assignmentGrade`
- You have two “lateness credits” that you can apply at your discretion
  - Typically: the new due date is 10am two days later (so procrastination will still hurt you)
  - I’ll let you know the exact extension date

- One implication: don’t ask me for additional “late days”
  - That’s unfair to other students!
- To use a “lateness credit”, you must send me an email before the cutoff time

- This policy doesn’t apply to a genuine emergency (*chas v’shalom*)
- Does apply to “I have a tough exam in another class”

Cheating



- Read the Homework document (on Piazza) for details
- Key point:
  - The department takes cheating very seriously
  - Steady stream of prosecuted and penalized students ☹
  - We maintain a database of “no recommendation for these students” ☹

New Students (Any Present)?

- Are there any “new students” (those without a YU Github repository)?
- If so: please send an email to `avraham.leff@yu.edu` immediately!
- Include
  - Your name (how you wish to be called)
  - A picture
- I need to setup a Git repository for you as soon as possible
- You need to become competent with Git as soon as possible

Course Project (I)

- This course is called Database Implementation
  - So: you'll have to implement a database ☺
- Two lectures from now: an official introduction to the course project
- Summary:
  - I want you to have the satisfaction of "owning all the code"
    - This goal is impossible if the goal is to create even a minimal relational database
    - I may change my mind on this in next course iteration ☺
  - The project is designed to make you think through "serious" database implementation issues
  - Your solutions won't have to be optimal: but they will have to solve those issues
  - The project is designed such that you can start no knowledge about databases

Course Project (II)

- Read the project requirements document (on Piazza) thoroughly!
  - It contains important background information as well as the “requirements”

> Because the milestones and requirements are published way in advance, there will be no late pass options available!

- You’re good programmers but this will likely be a lot of work
  - I urge you to get started immediately

Other Workload

- No midterm exam: work on your project instead ☺
- Yes, there is a final exam …
- Because of the issues related to remote exam administration, the final will be weighted considerably less than usual
- I haven't yet determined the precise grade weightings

I Want Your Feedback!

- Don't be shy!
  - Too fast? Too slow?
  - Clear? Unclear?
  - Anything that can improve your learning this material
- Don't wait until end of the semester: too late to help you then!

Databases: Motivation

Why Bother With Databases At All?

Remainder of today's lecture: make sure you understand why we need databases

What Is a Database System?

- Database: A collection of related data
  - Typically: “very large”
- Database Management System (or DBMS): software that stores, manages, retrieves, and transforms database data
  - As clients we interact with the “database” through the services of the DBMS
- Database System: the data, the DBMS, and “server-side” applications that access the data through the DBMS
  - As opposed to “client-side” applications (e.g., a web-page using JDBC to populate a form)
  - Not ready to argue about the subtleties of “server-side” *versus* “client-side” applications ☺
- These definitions raise the question: “why bother with a database system?”

Why Bother With Database Systems?

- Consider this flat-file (e.g., in csv format)
- What's wrong with storing my data in a file, and use a "text editor" to manage it?
- After all: the OS already provides a cheap, simple, file system!

```
employeeNumber, lastName, firstName, extension, email

1619,'King','Tom','x103','tking@classicmodelcars.com'
1621,'Nishi','Mami','x101','mnishi@classicmodelcars.com'
1625,'Kato','Yoshimi','x102','ykato@classicmodelcars.com'
1702,'Gerard','Martin','x2312','mgerard@classicmodelcars.com'
```

Key points:

- Database systems provide far, far more benefits than persistence
- Managing your file data can work, but will not be scalable ☹

File Scalability Issues: I

Q: How many problems can you think of for a "flat file" solution?

- Here are (only some) of the possible problems ...
- A_1: where does the meta-data get stored?
  - Example: What do each of the fields represent?
  - Example: What "type of data" do these fields represent?
- A_2: processing time, since the file has to be scanned and parsed each time you need data
- A_3: data redundancy and inconsistency
  - How do you prevent people from creating duplicate information in multiple files?
    - And how do you keep that data from becoming inconsistent?
  - Worse: each file can have its own file structure!
    - "first name" before "last name" ... or after

File Scalability Issues: II

- Note: the issues raised on the previous slide are all scalability issues
- They can be solved “by hand” for a small set of files & a small set of users
- We need a database system as soon as the data-sets or user-sets grow

But Databases Provide So Much More

Note: these can also be viewed as “scalability issues”

- Data retrieval
- Data integrity
- Data security

Data Retrieval

Consider these example queries:

- Find all employee email addresses
- Find all employees whose last name starts with 'K'

- Flat files force us to write a new program for every query
- Each program would have to read & parse the whole set of files

Ideally we want the "retrieval" process to be

- Easy to write
- Execute efficiently

Important: You're all smart & experienced enough to be thinking "indices", "store the previous results", "metadata" ...
Great: you've started to write a database system ☺

Data Integrity & Security

Flat files provide (very) poor support for the following services …

- Transaction isolation: how do you manage simultaneous (and different) modifications?
  - Classic example: two users withdraw funds from the same bank account
- Transaction atomicity: what if the system crashes in the middle of the data update?
  - Classic example: transfer of funds from one account to another should either complete or not happen at all
- How do you prevent "data entry" issues?
  - Example: enter a String instead of a Number?
- How do you enforce security & permissions?
  - Only HR should have access to my employee record
  - Oh …and my manager …
  - Oh …and me (but only to read – *not* to update – salary) ☺

OK: “Raw Files” Aren’t the Answer

- The points just discussed should have convinced you that we need to do a lot better than the “direct to file system” approach
- What is the answer?
- A database system!
- Database systems provide specialized services for managing data through a set of (conceptual) “modules” or “components”
- Some of the most important:
  - The storage manager component
  - The query processor component
  - The transaction management component

> What follows is a high-level overview of topics we’ll be addressing in this course ☺

Storage Manager

- A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- Responsible for:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- Sub-modules include:
  - Authorization and integrity manager
  - Transaction manager
  - File manager
  - Buffer manager

Data & Meta-Data

- At the lowest level (conceptually) of the system, the storage manager manages various data-structures
- "Data files": these store the database itself
- "Data dictionary": stores metadata about the structure of the database
  - Example: the database schema (see the "University schema" in your first assignment)
- "Indices": provide fast access to data
  - Think of an index as providing "pointers" to a set of data with common values

DML Versus DDL

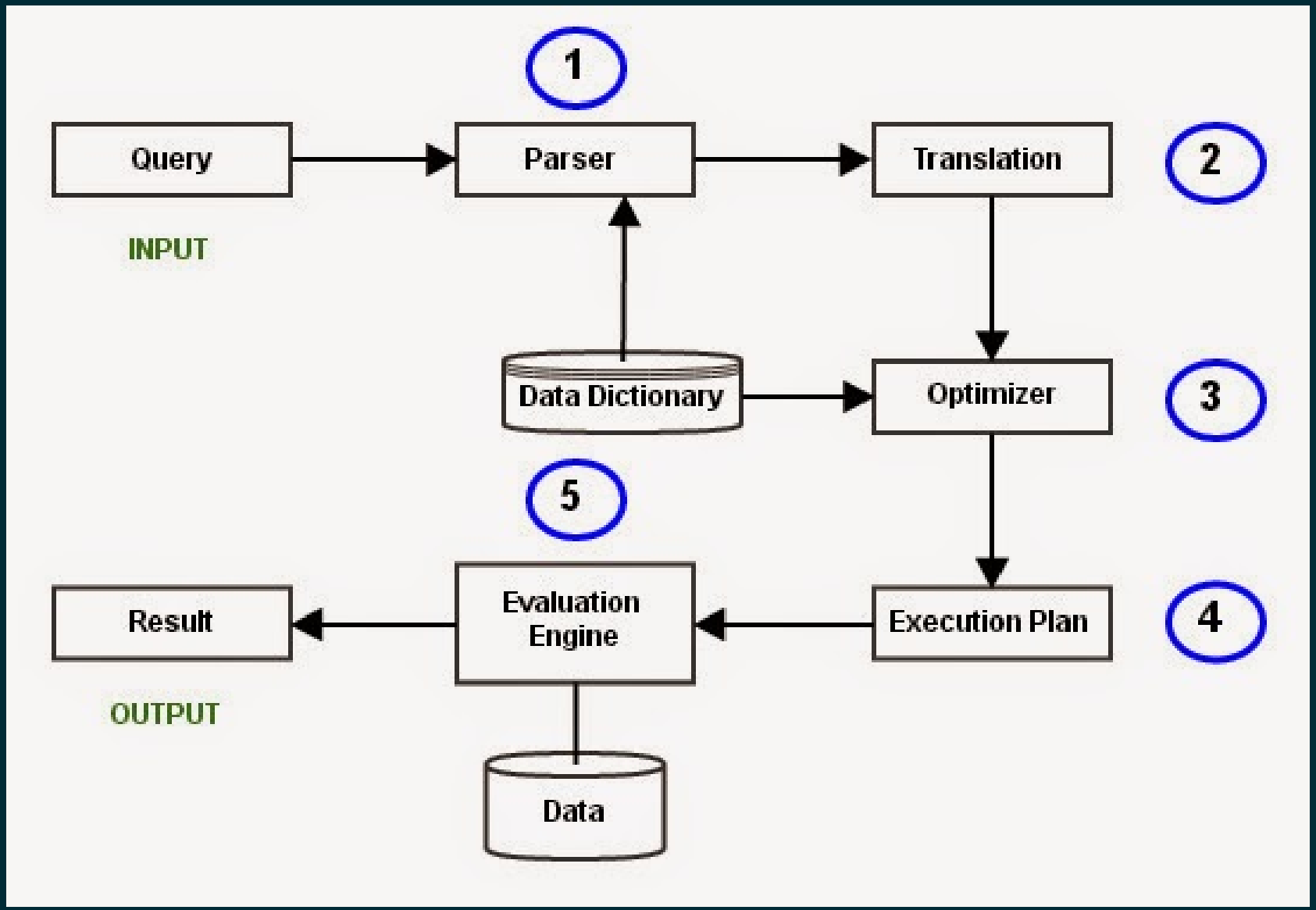
You'll need to know some terminology, not giving examples now ...

- DDL: Data Definition Language
  - Relational examples: "create" a table, "drop" (remove) a table, "alter" (modify) a table's definition
  - Defines meta-data for the system
- DML: Data Manipulation Language
  - Access & manipulate the data in the database
  - Not just a query language
  - SQL is the most widely used commercial language

Query Processor

- The query processor includes the following important sub-components
- DDL interpreter: interprets DDL ("data definition language") statements and records the definitions in the data dictionary
  - Example: `create table customers`
- DML compiler: translates DML ("data manipulation language") statements into an evaluation plan consisting of low-level instructions that the query evaluation engine understands
  - Example: `select id, name from customers`
  - The DML compiler performs query optimization
  - It picks the lowest cost plan from among the various alternatives
- Query evaluation engine: executes low-level instructions generated by the DML compiler

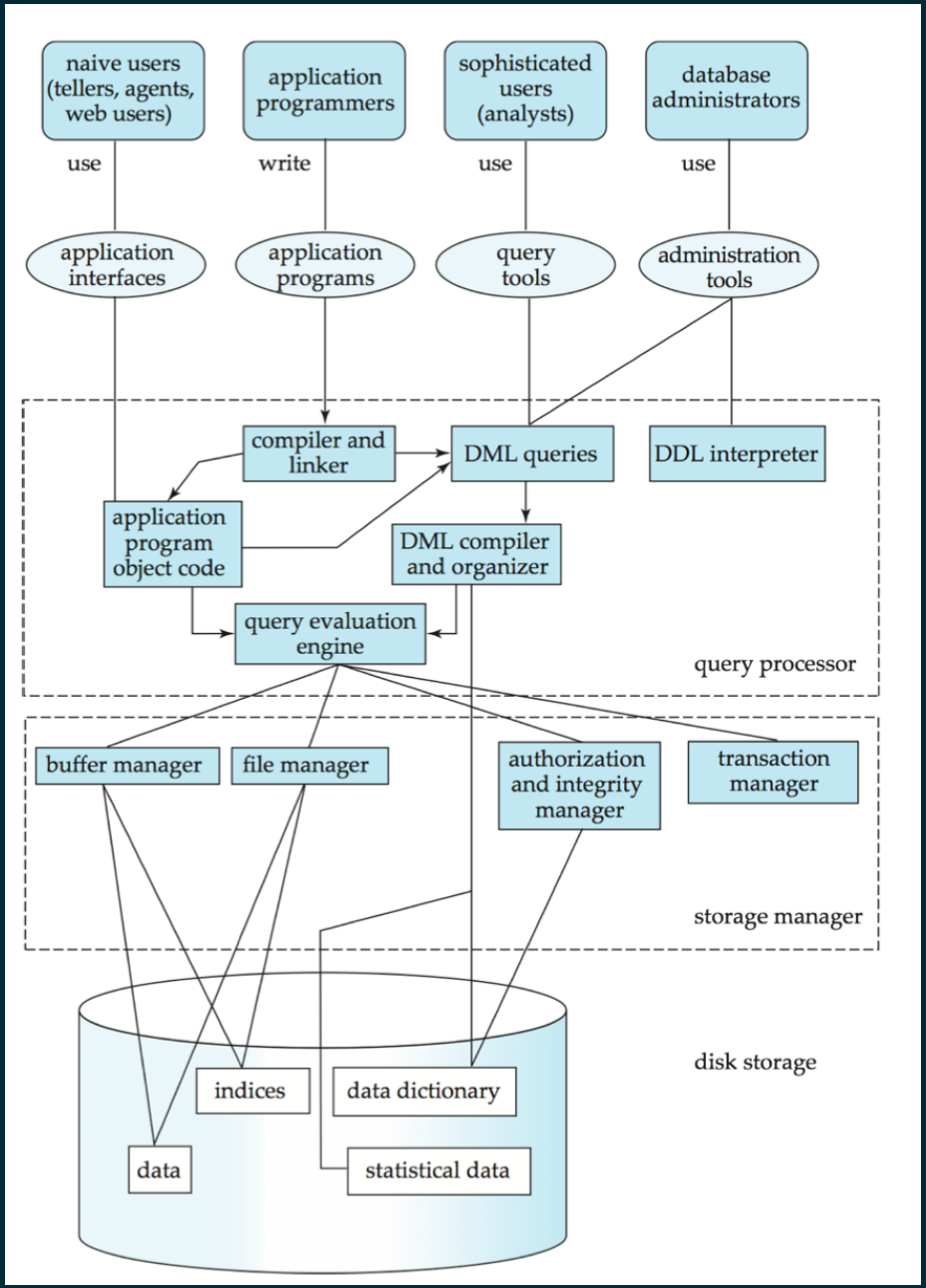
Query Processing: Some Key Players



Transaction Management

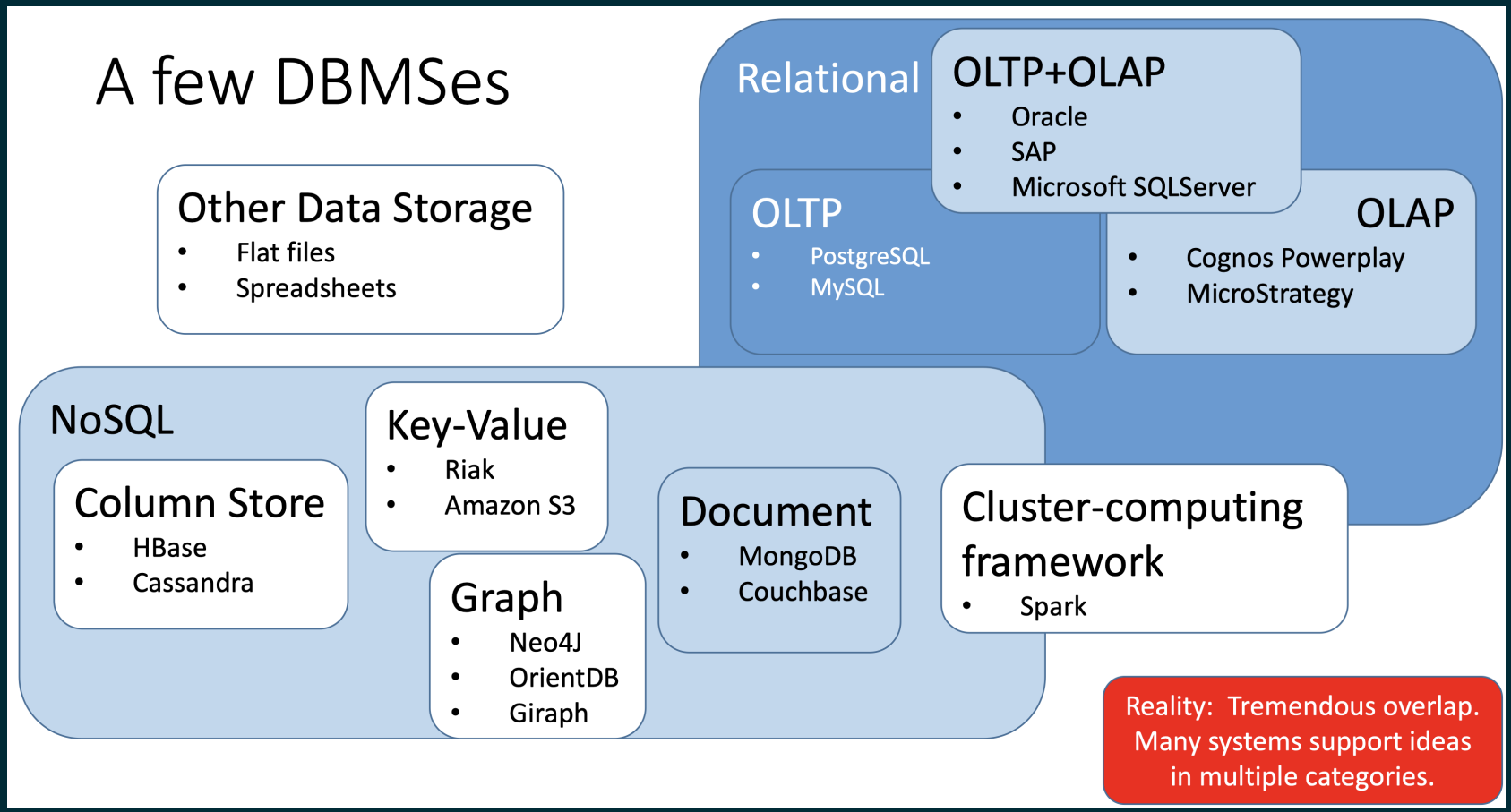
- A transaction is a collection of operations that performs a single logical function in a database application
  - Enforces the "single logical" semantics despite the fact that there are actually multiple functions being executed
- Transaction-management: ensures that the database remains in a consistent (correct) state despite
  - System failures (e.g., power failures and operating system crashes)
  - Transaction failures
- Concurrency-control manager: controls the interaction among concurrent transactions, to ensure the consistency of the database

Putting It All Together



Relational Databases: What Makes Them Different

Are Relational Databases Different?



- Q: OK, you've motivated the uses of "databases" in general but what's special about relational databases?
- A: let's take a quick look at some of the special RDB features
  - And: you may want to take *Modern Data Management* to learn more ☺

Relational Databases (Relative To Other Types)

- Data is highly structured as mathematical relations
- Structure is rigorously defined & enforced!
- Typically: "design before implementation"
- Data integrity is of critical importance!
  - System enforces data consistency properties
  - System controls access permissions
- System optimizes storage & operations (can do so because it controls everything rigorously)

These characteristics are manifested in the relational data model (stay tuned!)

OLTP Versus OLAP

OLTP

- Online Transaction Processing
- Frequent updates
- Flexible queries
- Performance requirement: “do everything well”
- OLTP is the focus of this course!

OLAP

- Online Analytical Processing
- Analyze historical data
- Periodic updates
- “Canned” queries
- Performance requirement: optimize for specific canned queries

OLTP primarily uses SQL

OLTP Versus OLAP: Different Usage Scenarios

- OLTP example
  - Company charges the customer for a purchase
  - Results has better be exactly correct!
- OLAP example
  - Database is used for "business intelligence" such as data mining or machine learning
- Key difference: OLAP results do not have to be "completely correct" ☺
  - Example: company wants to know how many books at $< \$10$ each were sold in New Jersey during July 2017
- And . . . being "completely correct & fast" is too expensive

“Data Model” Concept

- The concept of a data model is important, but hard to define precisely ☹
- A data model is an abstraction that describes
  - Data (e.g., the structure and sub-structure “type” of an employee record)
  - Data relationships (e.g., employee ⇔ departments)
  - Data constraints
  - Data semantics (e.g., employee’s zipcode can only be from a restricted set of values)
- Example: compare & contrast Java’s notion of a data-type (very precise, very constrained) to JavaScript’s (very fluid)
- Databases, like programming languages, have data models

Relational Data Model

- In this course, we'll focus on the relational data model
- But also study the entity-relationship (or E-R) model:
  - Describes data in terms of "entities" and the "relationships" between entities
  - Example: "1 → 1", "1 → many", "many → 1", "many → many"
  - We use E-R to design a database
  - Once we've defined the entities and their relationships, we implement an RDB by "translating" E-R to relational model

> Deferring exploration of the relational data model for later lecture

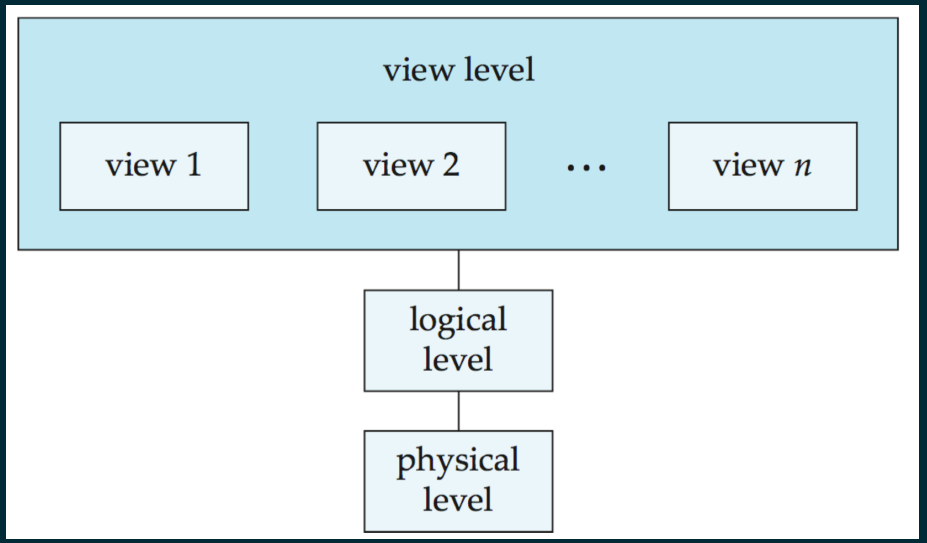
Other Data Models: Less Important For This Course

- Object-based models: use your application program's "object oriented" data model in the database as well!
  - Can extend E-R model to have OO features
  - Can meld (usually badly ☹) OO with relational model
- Semi-structured model: fewer constraints than relational model
  - Example: allow data instances (of the same "type") to have different sets of attributes
  - Examples: XML , JSON
  - Intrinsically a "bad fit" for an RDB, but recently RDBs have been forced to accommodate (if they want to keep their customers ☺)

Multiple Database Levels

- Physical level: describes how a record (e.g., employee) is stored as a single block of storage. Describes how multiple records are stored and associated with each other
  - Programming language analogy: “bits & bytes”, “array layout”, “virtual method tables”
- Logical level: describes what data is stored in database, and the relationships among the data
  - Programming language analogy: “employee object” (or “struct”)
- View level: describes a subset of the logical level.
  - Can subset within a record instance: e.g., hide employee salary information
  - Can subset within sets of instances: e.g., the set of employees in a given department
  - Views provide database security and simplicity

Data Abstraction



- Key point: data abstraction provides physical data independence
  - Ability to modify the physical *schema* without changing the logical schema
- Applications depend ("code to") the logical schema
  - The DBMS can change the physical level and schema without (seriously) affecting higher level

Views: Multiple "Virtual Constructs" On Same Data

- Different "user classes" may have different usage requirements
- Example: "need to know" motivation
  - Payroll department must see salaries but no need to see diseases
  - Health department must see diseases but no need to see salaries
- Example: convenience motivation
  - Payroll department may prefer to see salary presented in different currency
  - Health department may prefer to see "current age" rather than "date of birth"
- Database views are tables that are computed from the actual database base tables

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley (Michael Stonebraker) begins Ingres prototype
    - Oracle releases first commercial relational database
  - High-performance (for the era) transaction processing

History of Database Systems (Cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
    - Wisconsin, IBM, Teradata
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce

History of Database Systems (Cont.)

- 2000s
  - Big data storage systems
    - Google BigTable, Yahoo PNuts, Amazon,
    - “NoSQL” systems.
  - Big data analysis: beyond SQL
    - Map reduce and friends
- 2010s
  - SQL reloaded
    - SQL front end to Map Reduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases

Going Forward: Short-Term Plan

- Readings: Read Chapter 1
- Next two lectures: introduction to course project
- Afterwards: focus on relational model

Today's Lecture: Wrapping it Up

Course Overview

Rules Of The Game

Databases: Motivation

Relational Databases: What Makes Them Different