CS-300: Data-Intensive Systems

Introduction to the course
Why data management?
DBMS Architecture
Entity-Relationship Model

Prof. Anastasia Ailamaki, *Prof. Sanidhya Kashyap*

Data-Intensive Applications and Systems

EPFL

Administrivia

Course Overview

- Book: **Database System Concepts Seventh Edition** by Avi Silberschatz, Henry F. Korth, and S. Sudarshan
- Course organization: Lectures, exercises, project (labs), midterm and final
- Staff email: cs300-staff@groupes.epfl.ch

- Instructors: **Anastasia Ailamaki** and **Sanidhya Kashyap**
- TAs:
  - Antonio Boffa, Ioanna Tsakalidou, Musa Unal, Georgiy Lebedev, Kumar Kartiyeka Dwivedi, Tao Lyu, Gupta Vishal and Yueyang Pan
- AEs:
  - Ilyas Faresse, Amene Gafsi, Tymur Tytarenko, Andrej Kotevski

Schedule

- Lectures: Mondays 11-13 in CE 1 4
- Exercises&Labs: Wednesdays in BCH 2201
- Labs:
  graded programming exercises released during the course
- Midterm/Final exams:
  will be conducted in person on the scheduled exam day

- Lecture slides (and all additional course materials), office hours, detailed schedule will be available on Moodle

- TAs will be available to answer questions during exercise/project sessions and office hours (office hours available on Moodle)

	Lecture (Monday)		Lab session (Wednesday)		Lab release	Lab deadline
	Date	Topic	Date	Topic		
Week 1	Feb 17	Intro, overview: ER	Feb 19	Tutorial 1: SQL 1		
Week 2	Feb 24	Relational Model & Relational Algebra & SQ	Feb 26	Tutorial 2: SQL 2	Lab 1: 9AM Feb 24	
Week 3	Mar 3	Storage, Files, and Indexing	Mar 5	Tutorial 3: SQL ER - Relational model (traslation)		
Week 4	Mar 10	Storage and Buffer Management	Mar 12	Lab session 4		
Week 5	Mar 17	Indexes: B-Tree	Mar 19	Lab session 5		
Week 6	Mar 24	Hashing / Sorting	Mar 26	Lab session 6	Lab 2: 9 AM Mar 24	Lab 1: 9 AM Mar 25
Week 7	Mar 31	Query Operators I (not included in midterm)	Apr 2	Lab session 7		
Week 8	Apr 7	**Midterm**	Apr 9	Lab session 8		
Week 9	Apr 14	Query Operators II	Apr 16	Lab session 9		
Week 10	Apr 21	**Spring break**	Apr 23	Lab session 10		
Week 11	Apr 28	Query Optimization	Apr 30	Lab session 11		
Week 12	May 5	Transactions and Concurrency Control & Concurrency I	May 7	Lab session 12	Lab 3: 9 AM May 5	Lab 2: 9 AM May 5
Week 13	May 12	Concurrency Control and Eventual Consistency & Concurrency II	May 14	Lab session 13		
Week 14	May 19	Parallel and Distributed data systems	May 21	Lab session 14		
Week 15	May 26	Extra	May 28			
	Jun 2		Jun 4			Lab 3: 9 AM Jun 2

Logistics

- Course information: **Moodle**
- Course discussion: **Ed**
- Programming labs: **Moodle/Github classroom**
- Lectures: In-person lectures in **CE 1 4**
  - Attendance is strongly recommended; lectures are not recorded
  - Recordings from previous years are available but some material may change

- Grades: 3 labs (30%) + final exam (70%)
  - Each lab contributes 10%
  - Midterm is not graded

Exams

- Labs (30%): graded programming exercises
  must be submitted on Moodle or GitHub Classroom (more details in lab description documents)

- Exams (70%): in person on the scheduled day, using Moodle for submission
  Questions can be from both lectures and labs

  Midterm: Monday April 7, 11am (not graded)
  (purpose: test your knowledge and have a hands-on exam experience)
  Covers weeks 1–6

  Final (70%): Date / time / place TBD
  Covers ALL lectures and lab content

Time management (suggestion)

- **6 ECTS** points map to **11–13 hours/week**[1]
- Divide-and-conquer between studying and labs
  - 2h - lecture
  - 3h - exercise & lab session
  - 4h - study
  - 2-3h - lab & homeworks

[1] https://www.epfl.ch/education/bachelor/study-programs-structure/faqs/

Academic honesty

Collaborate, **EXCEPT** on labs and exams.

Adhere to the EPFL academic honesty policies :

- Do **not share results, answers, or other material about labs** with others
- Do **not use code, text, results, or other materials generated by another person or software tool** in your solution
- Do **not ask anyone** to provide materials to you

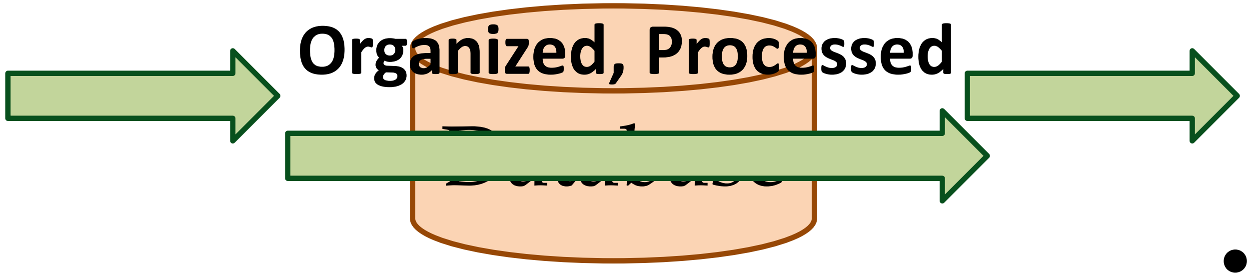
According to the academic integrity policy of EPFL (https://bit.ly/3BmfHJU):

- Cheating and academic integrity violations will be reported
- Each deliverable will be checked for plagiarism

Why data management?

What is data?

Organized, Processed

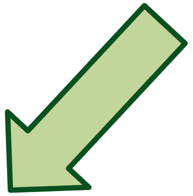


Data

- Facts
- Basis for reasoning/discussion/ calculation
- Useful or irrelevant or redundant
- Must be processed to be meaningful

Information

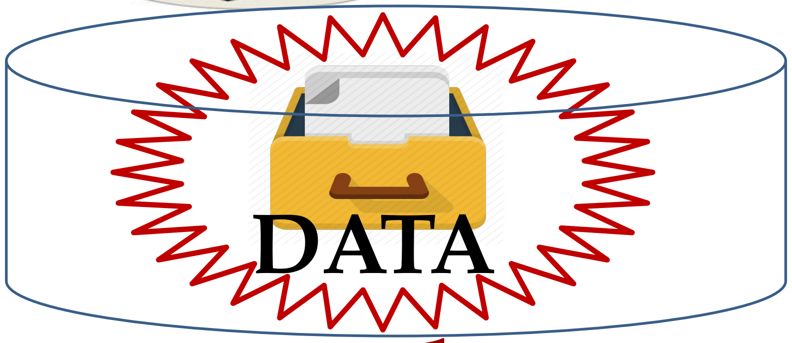
- Has meaning
- Relevant to the problem
- Actionable – leads to a solution



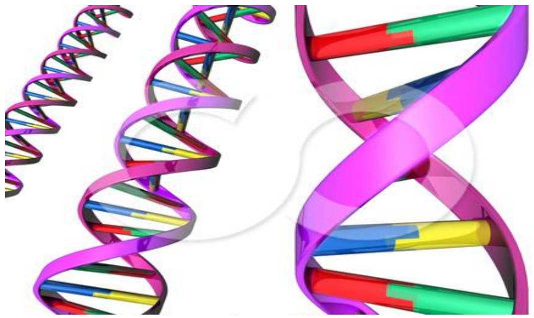
Knowledge

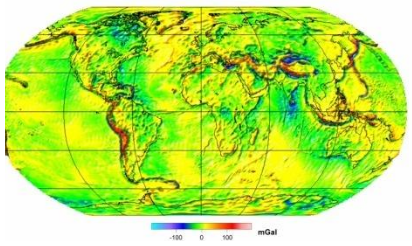
Wisdom

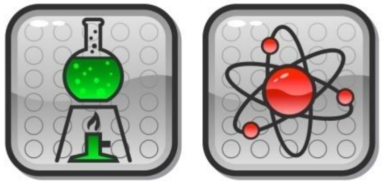
Have you ever “used” a database?



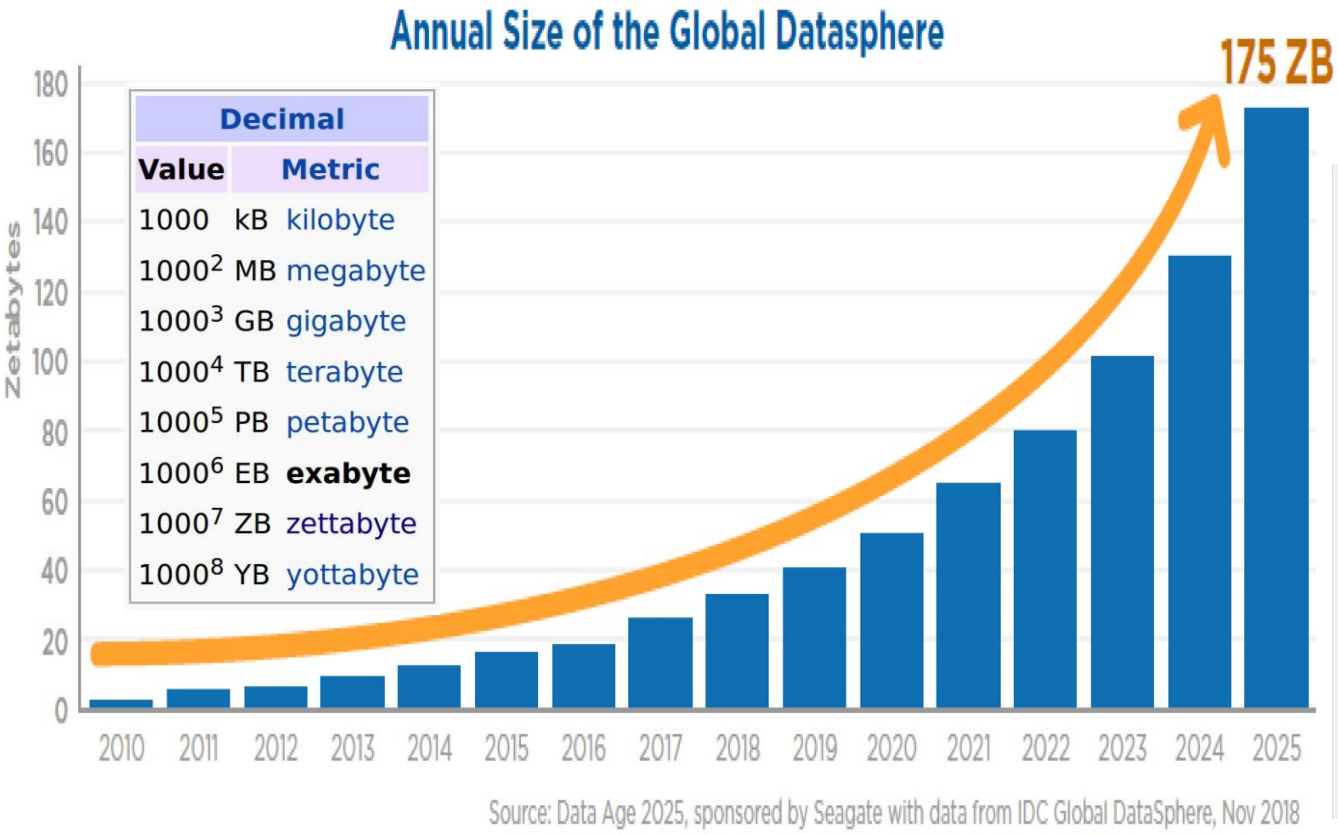
Database





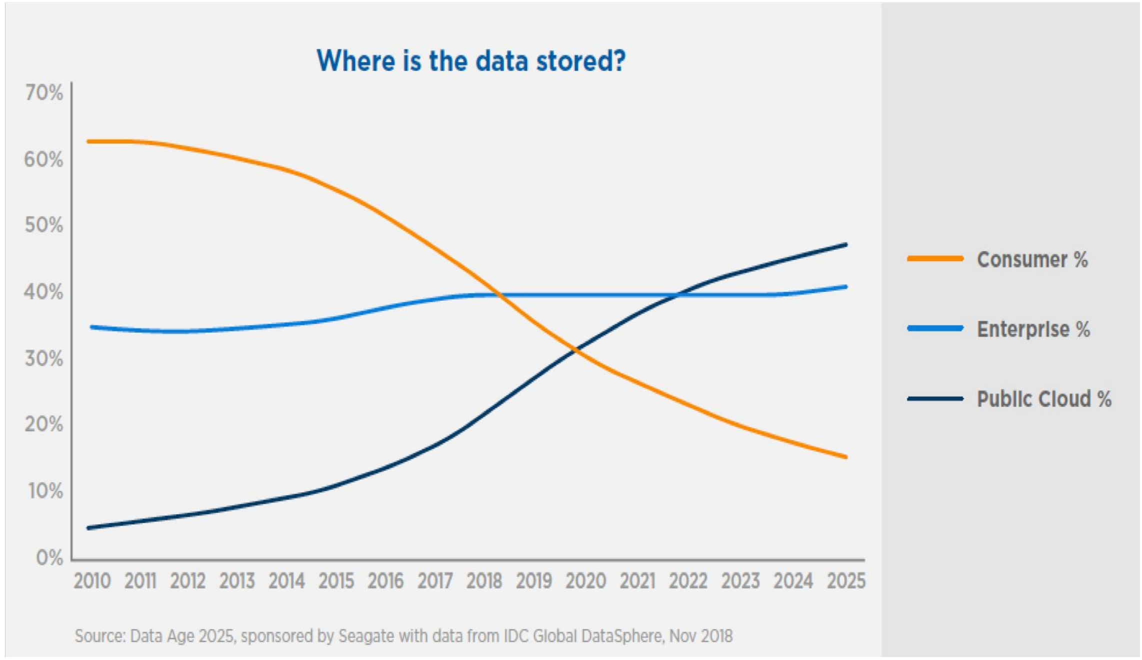


How big is “all” data?



Decimal		
Value	**Metric**	
1000	kB	kilobyte
1000^2	MB	megabyte
1000^3	GB	gigabyte
1000^4	TB	terabyte
1000^5	PB	petabyte
1000^6	EB	**exabyte**
1000^7	ZB	zettabyte
1000^8	YB	yottabyte

Data is moving to the cloud



Data is generated everywhere, all the time!

LIBRARY OF CONGRESS

- 130 million items
- 10,000 new items added each day
- 530 miles of shelves
- 5 million digital documents
- 20 terabytes of text data

Sprint

- 2.85 trillion database rows.
- 365 million call records processed per day
- At peak, 70,000 new call record per second

at&t

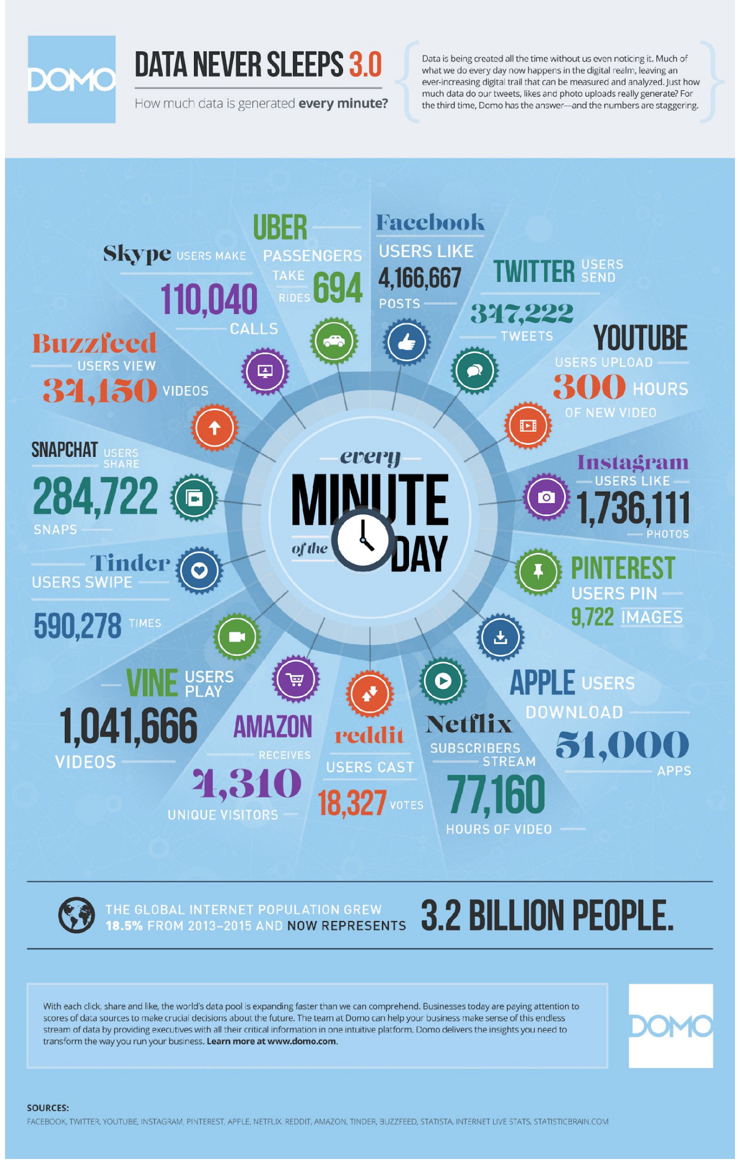
- 323 terabytes of information
- 1.9 trillion phone call records

NERSC
NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

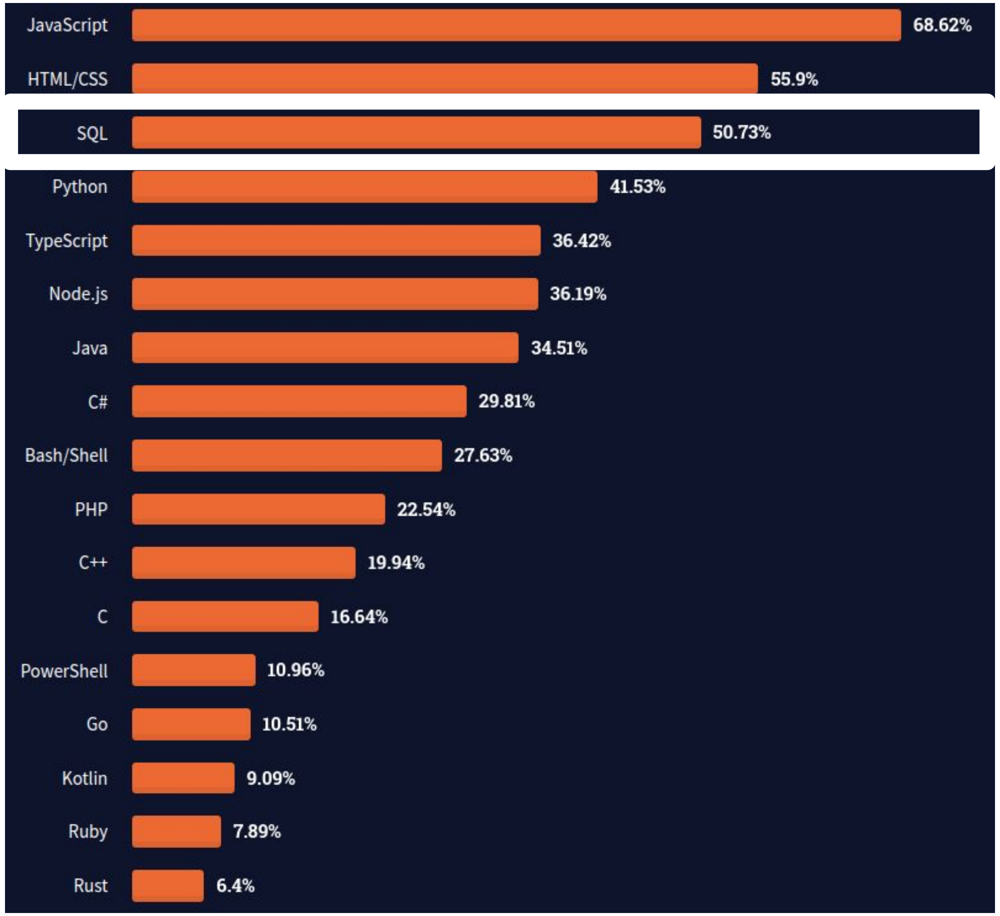
- 2.8 petabytes of data
- Operated by 2,000 computational scientists

WORLD METEOROLOGICAL ORGANIZATION

- 220 terabytes of web data
- 6 petabytes of additional data

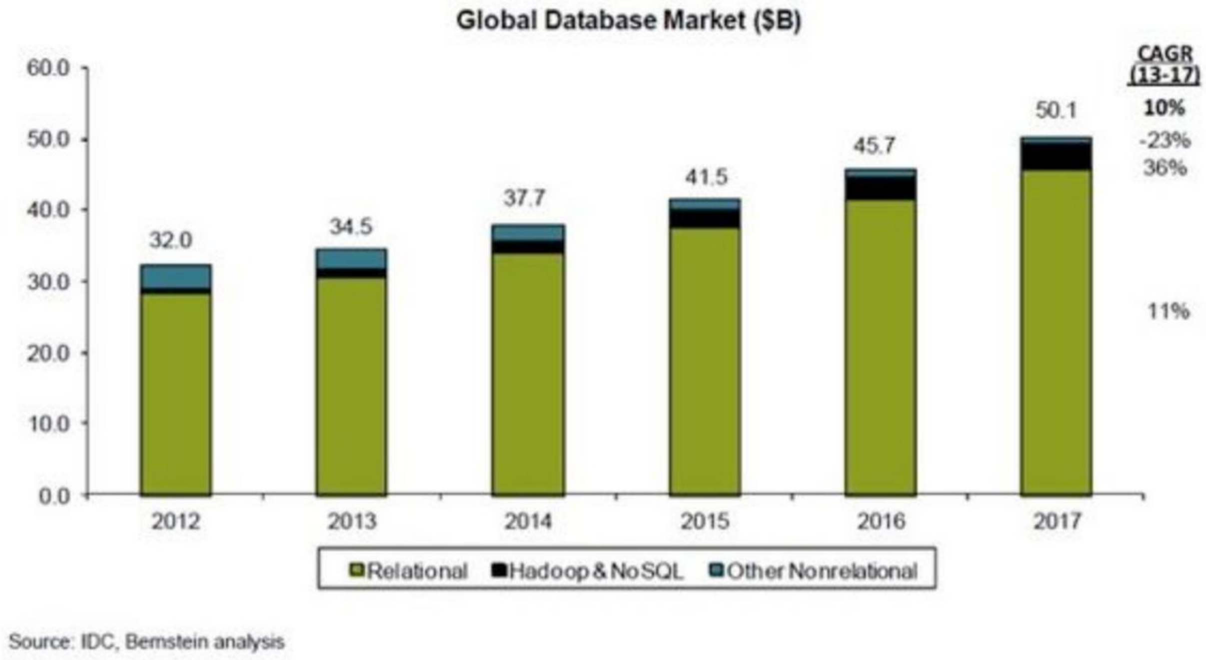


Why study Databases?



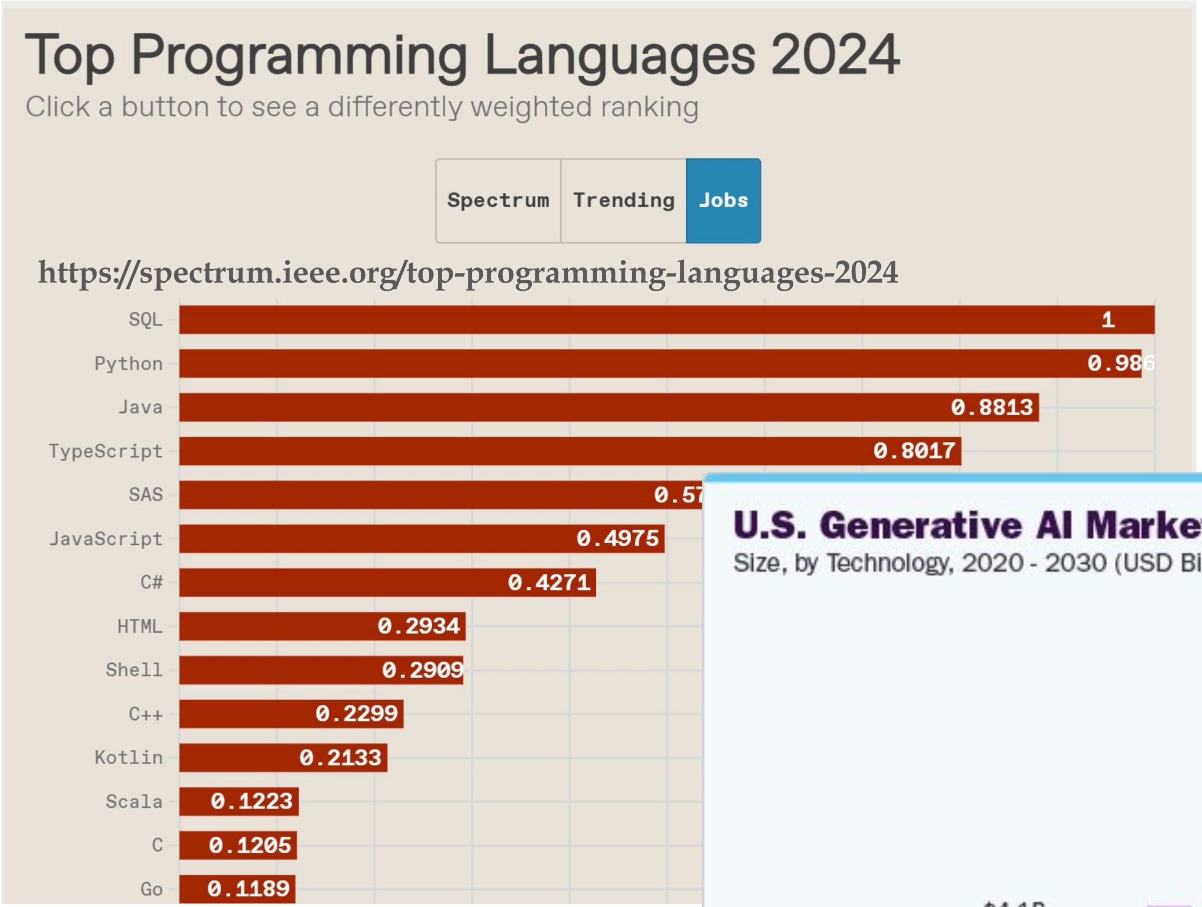
Need for DBMS always high

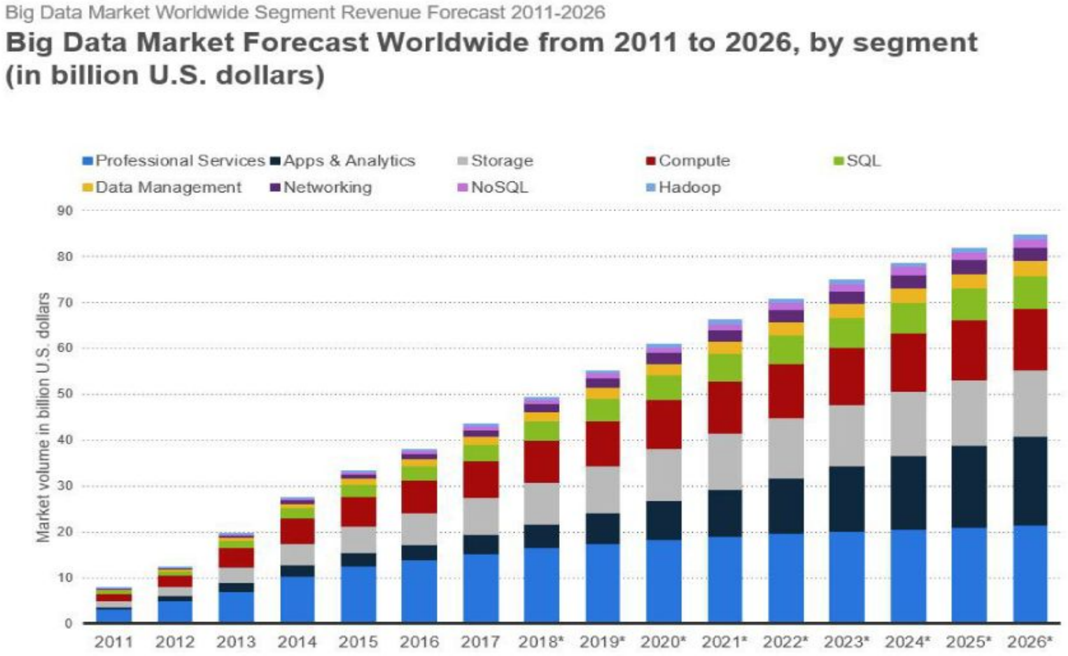
- **Corporate**: "supply chain mgmt", "data analytics", "data science", etc.
- **Scientific**: Digital humanities, Human Brain project, sensor networks

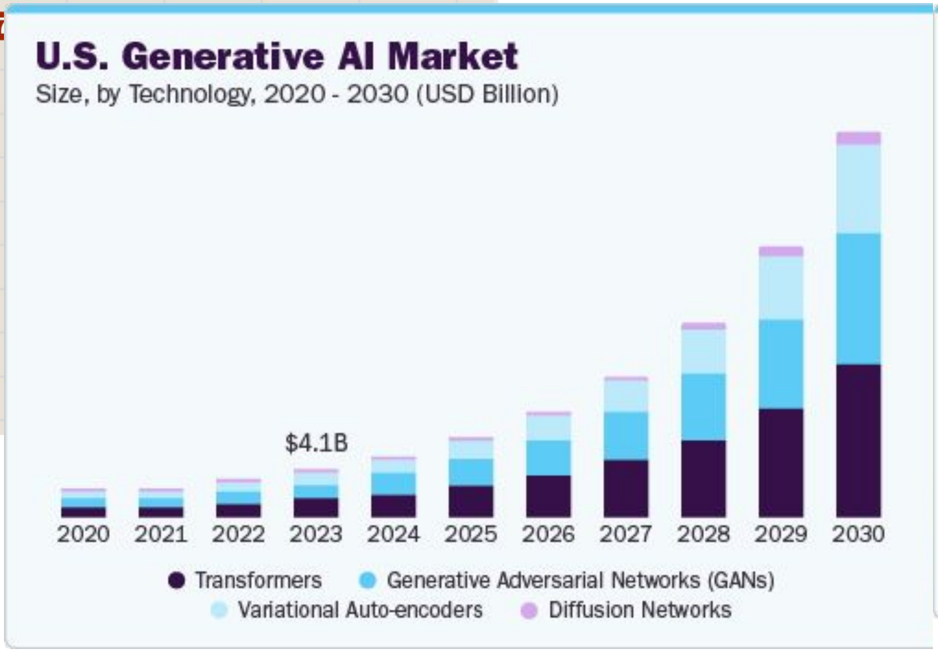


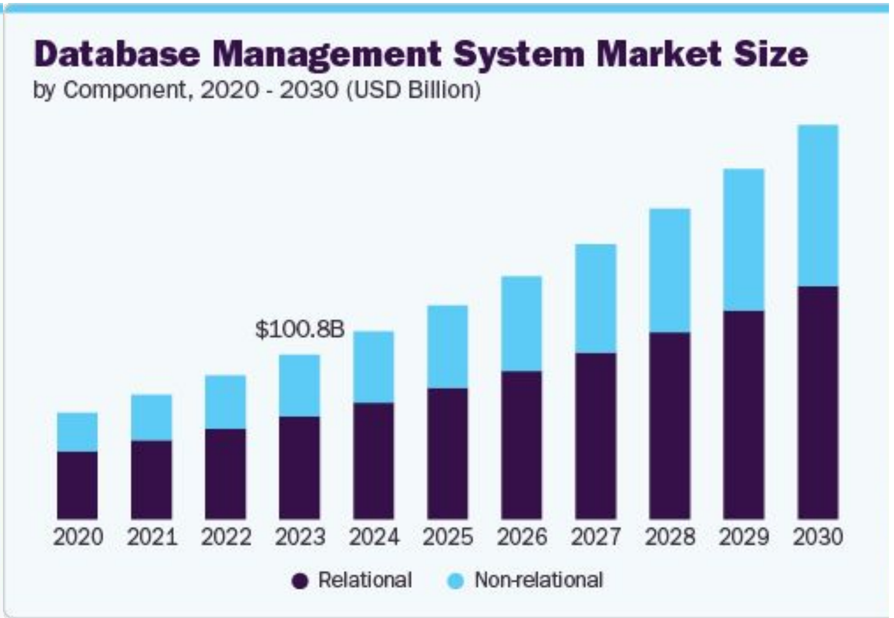
https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language-prof

Big Data markets









What is a database?

- A **large**, **integrated**, **structured collection** of data
- Usually intended to model some real-world enterprise
- Example: University
  - Courses
  - Students
  - Professors
  - Enrollment
  - Teaching

Courses, Students, Professors → **Entities**

Enrollment, Teaching → **Relationships**

What is a Database Management System (DBMS)?

- A software system designed to **store**, **manage,** and **facilitate access** to databases
- **DBMS** = Interrelated data (database) + set of programs to access it (software)

What does a DBMS do?

Protects data from failures: h/w, s/w, power; malicious users

Thousands of queries / updates per second

24X7 availability

Physical data independence, declarative high-level query languages

Provides **efficient**, **reliable**, **convenient**, and **safe** **multi-user** storage of and access to **massive** amounts of **persistent** data.

Concurrency control

Extremely large (often Exabytes every day)

Data outlives the programs that operate on it

Is your 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 are updating a file
  - The power goes out!
  - Which of your changes will survive?

A) All **B) None** **C) All since last save** **D) ???**

Is your file system a DBMS?

- Thought experiment 1:
  - You and your project partner … the
  - You bo…
  - Wh…

A) … ???

- Th…
  - The po…
  - Which of your … sur…

How do you write programs over a platform that promises “???” ?

A) All B) None C) All since last save D) ???

Is the web a DBMS?

- Fairly sophisticated search available
  - Crawler *indexes* pages for faster search

- However…
  - Data is mostly **unstructured** and **untyped**
  - Correct answer to a search query is NOT well-defined
    - NO guarantee of completeness
  - Cannot manipulate data
  - Few guarantees provided for data freshness, consistency across data items, fault tolerance …
  - Websites typically have a **DBMS** in the background for requests:
    - Ex: nba.com (SAP HANA), facebook.com (MySQL and others)

Is ChatGPT a DBMS???

The scope of DBMS

- **What more could we want than a file system?**
  - Simple, efficient *ad-hoc*[1] queries
  - Concurrency control
  - Recovery
  - benefits of good ***data modeling***

- **Simple programming ? Not really…**
  - things become very complex very fast: many queries, users, transactions, resources…

- **Can the OS offer services like memory management etc? Again – not really …**
  - In fact, the OS often comes in the way!

ad-hoc: formed or used for specific or immediate problems or needs

What is the intellectual contribution?

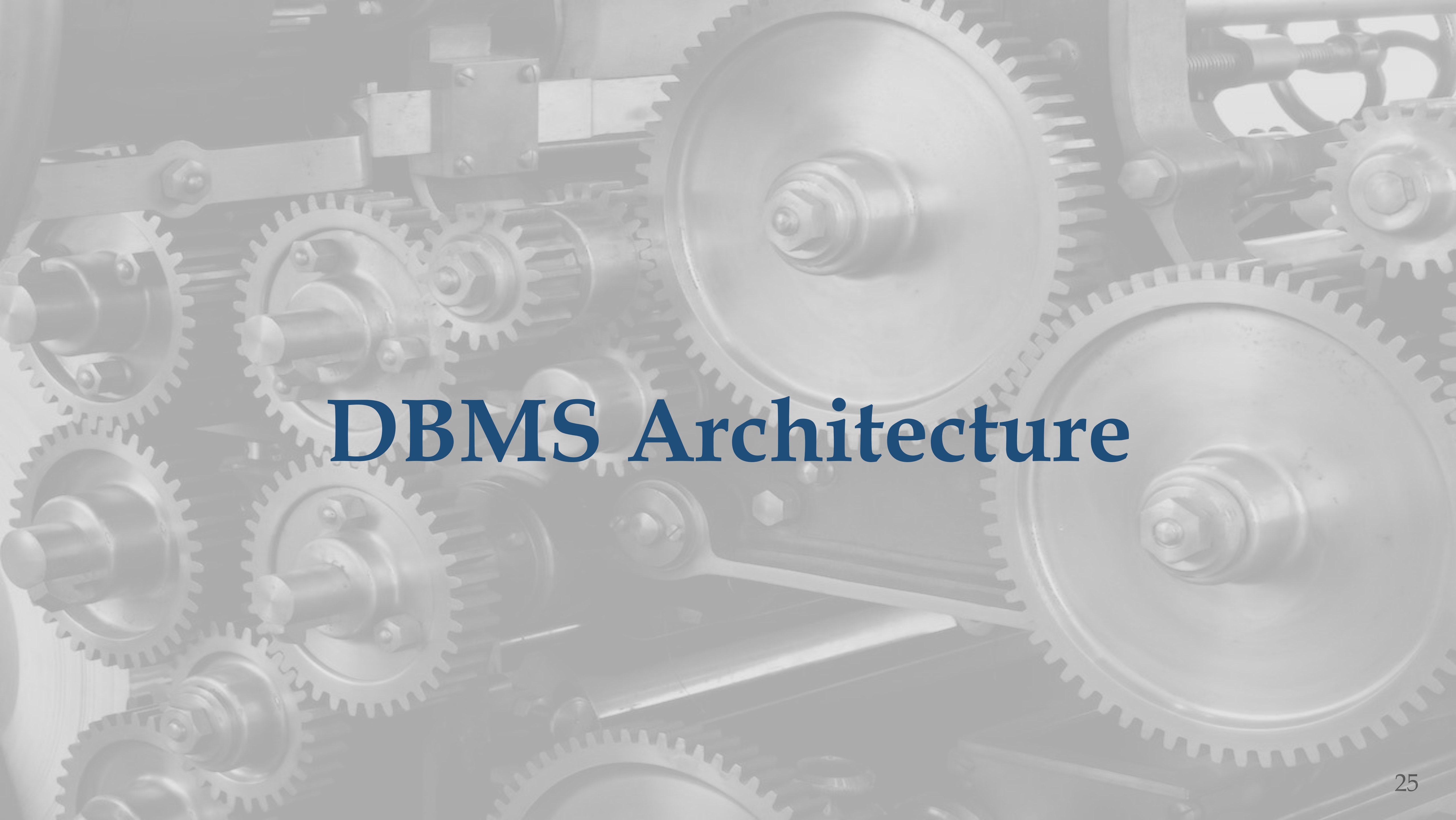
- **Representation information**
  - Data modeling
- **Languages and systems for querying data**
  - Complex queries with *real semantics*
  - Over massive amount of data
- **Concurrency control for data manipulation**
  - Controlling concurrent accesses
  - Ensuring transactional semantics
- **Reliable data storage**
  - Maintains data semantics even after pulling the plug (i.e., power off)

- # How to design and build a data-intensive application?

Application "sits" on top of a DBMS!

A detailed look "under the hood" of a DBMS is key:

- The best application writers & database administrators understand DBMS internals
- Intellectual content relevant to other contexts (e.g. web, OS, file systems)
- DBMS technology still very much evolving
    - Distributed/map-reduce databases, NoSQL movement
    - Column stores, row stores
    - Scientific databases, vector databases, embeddings
    - ML for DB, DB for ML



DBMS Architecture

Describing data

- A **data model** is a collection of concepts for describing data:
  - Higher-level: Hides lots of low-level storage details
  - Relational, hierarchical, graph, object-oriented …
- **Relational data model**
  - Set of records
  - **Relation:** Table with rows and columns
  - **Schema:** Describe the structure (columns) of a relation
- **Nested data model**
  - *Not all data fits naturally in tables!*
  - Hierarchy, arrays, etc.
- **Schema vs. Data**
  - Type vs variable
  - Description of a particular collection of data, using a given model

Example: Schema of a University database

Students

sid	string
name	string
login	string
age	integer
gpa	real

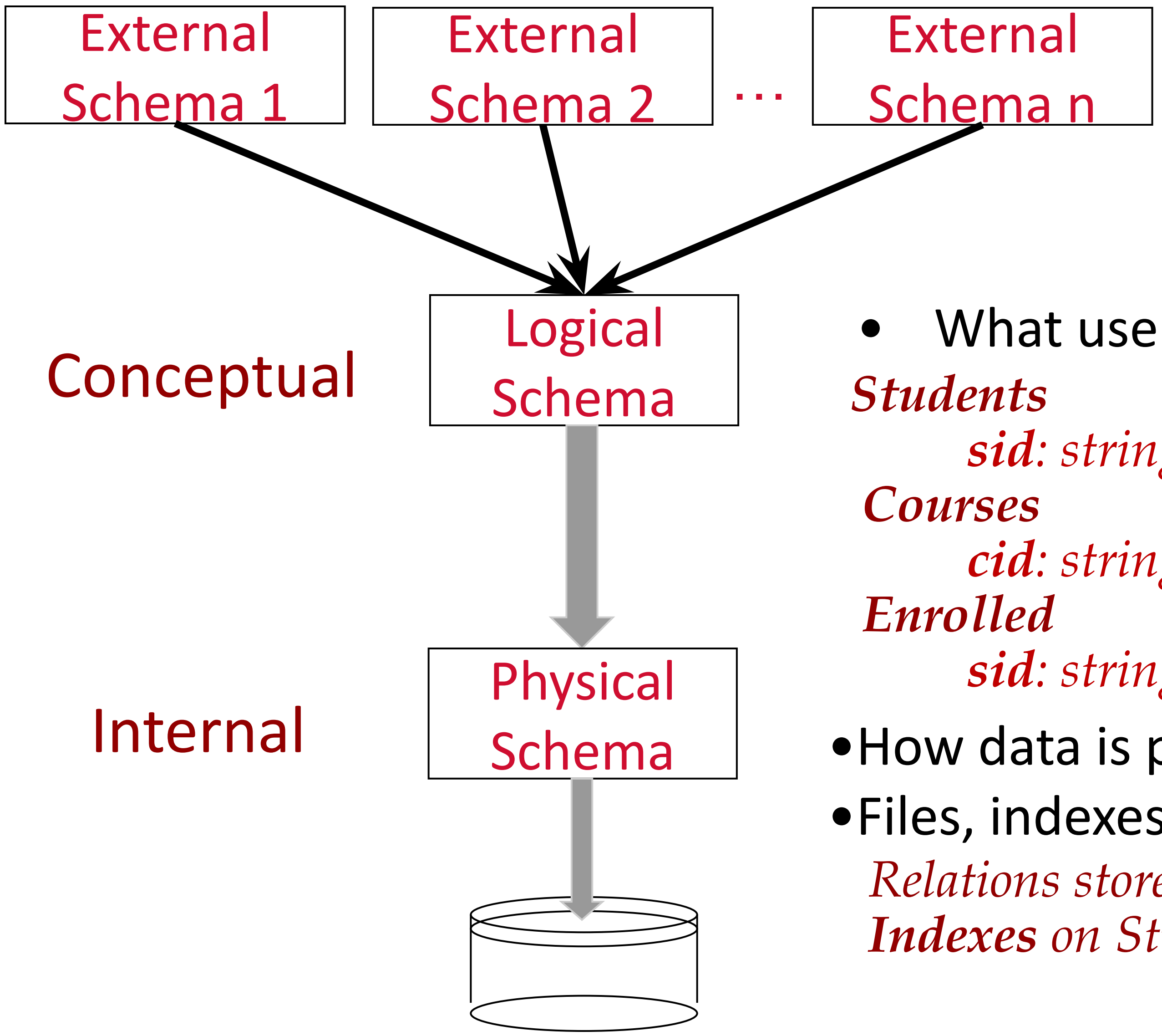
Enrolled

sid	string
cid	string
grade	string

Courses

cid	string
cname	string
credits	string

Describing data: Levels of abstraction



- User access control

  Students_Info
  - ***sid:*** *string*, ***name:*** *string*

  Course_Enrollment
  - ***cid:*** *string*, ***#enrolled:*** *integer*

- What users, application programs see

  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*

- How data is physically stored on disk
- Files, indexes…

  Relations stored as ***unordered files***

  Indexes *on Students.sid, Courses.cid*

DBMS cares about data independence types

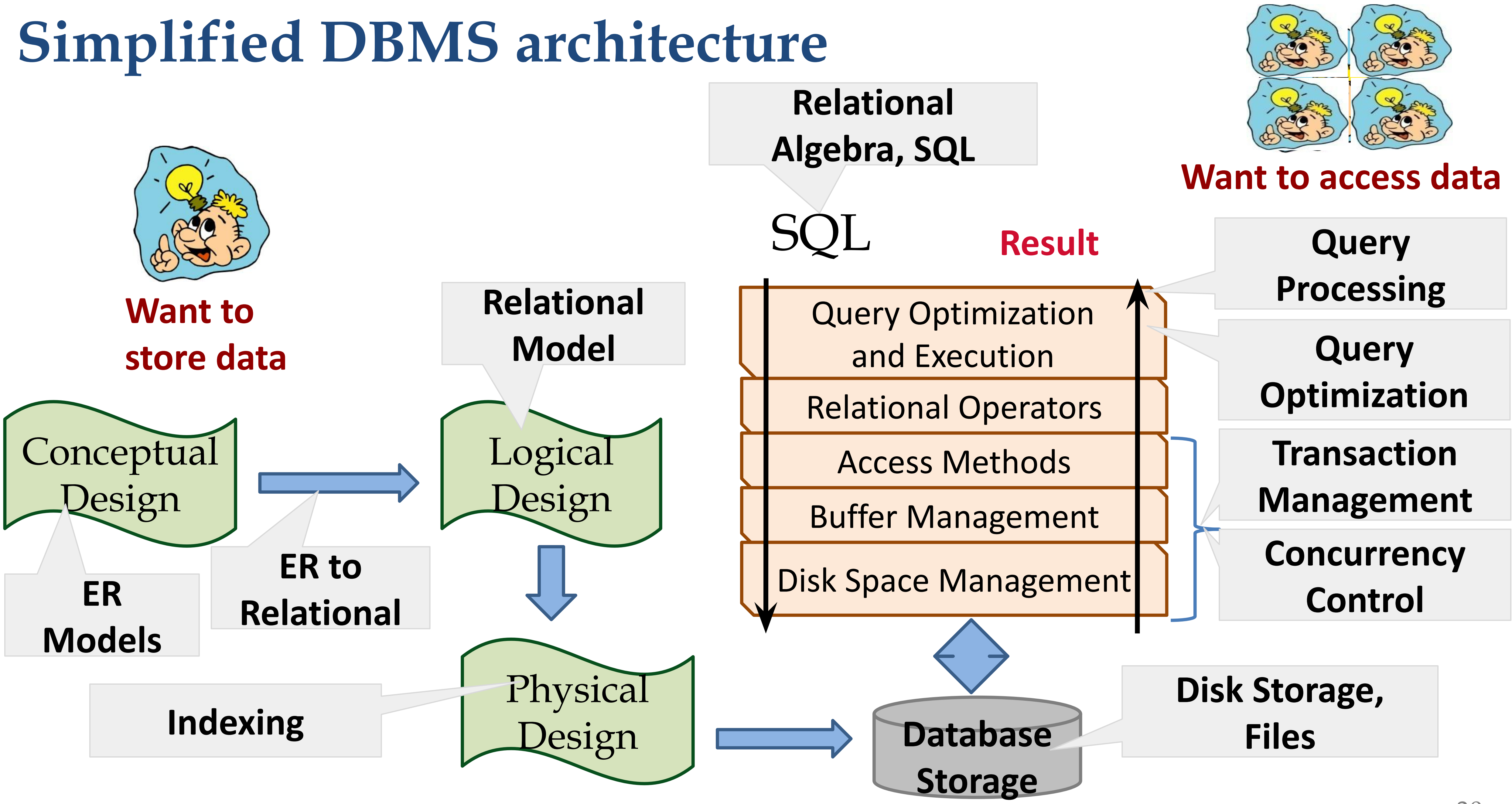
- **Data independence**: The ability to change the schema at one level of the database system without changing the schema at the next higher level

- **Logical data independence:** The capacity to change the conceptual schema without changing the user views

- **Physical data independence** The capacity to change the internal schema without having to change the conceptual schema or user views

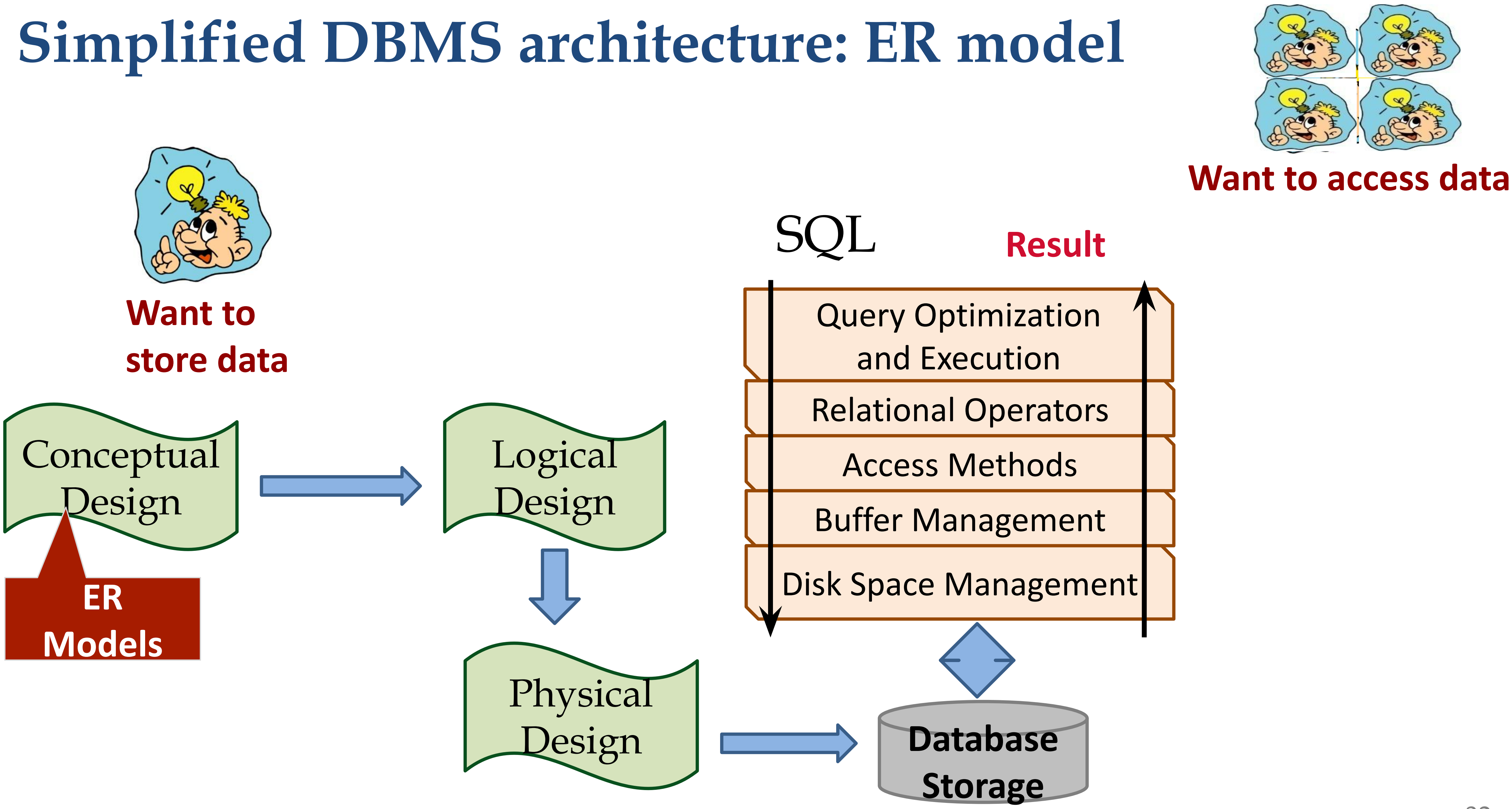
Q: Why is this particularly important for a DBMS?

Simplified DBMS architecture



Conceptual Design: Entity-Relationship Model

Simplified DBMS architecture: ER model



The ER model

- **Basic ER modeling concepts**
  Readings: Chapter 1.1-1.5 6.1-6.2
- Constraints
- Complex relationships
- Conceptual Design

(recap) Describing data: Data model

- A **data model** is a collection of concepts for describing data:
  - Higher-level: Hides lots of low-level storage details
  - Relational, hierarchical, graph, object-oriented …

- **Relational data model**
  - Set of records
  - **Relation:** Table with rows and columns
  - **Schema:** Describe the structure (columns) of a relation

- **Schema vs. Data**
  - Type vs variable
  - Description of a particular collection of data, using a given model

Relational data model

- Rows and columns
- Keys and foreign keys to link relations

Enrolled

SID	CID	Grade
53666	Carnatic101	5
53666	Raggae203	5.5
53650	Topology112	6
53666	History105	5

Students

SID	Name	Login	Age	GPA
53666	Jones	jones@cs	18	5.4
53688	Smith	smith@eecs	18	4.2
53650	Smith	smith@math	19	4.8

- How do we design the schema?

Relational data model

- Rows and columns
- Keys and foreign keys to link relations

Enrolled

sid	cid	grade
53666	Carnatic101	5
53666	Reggae203	5.5
53650	Topology112	6
53666	History105	5

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	5.4
53688	Smith	smith@eecs	18	4.2
53650	Smith	smith@math	19	4.8

36

- How do we design the schema?

Database design requirements

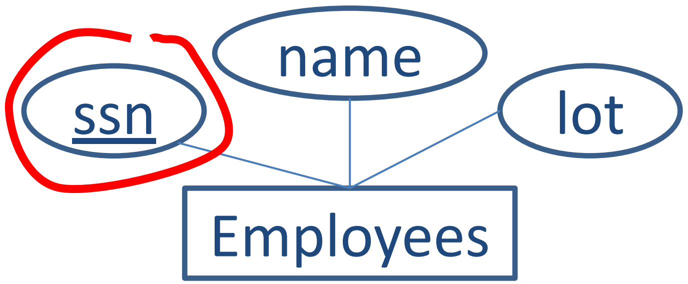
- **Requirements analysis**
  - User needs; what must database do?
- **Conceptual design**
  - High-level description (often done with ER model)
- **Logical design**
  - Translate ER into DBMS data model
- **Schema refinement**
  - Consistency, normalization
- **Physical design**
  - Indexes, disk layout
- **Security design**
  - Who accesses what

What is a conceptual design

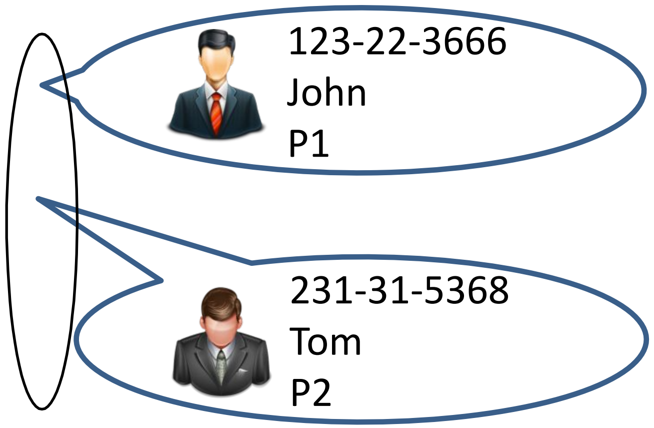
- What are the *entities* and *relationships* in the enterprise?
- What information about entities and relationships should we store in the database?
- What are the *integrity constraints* that hold?
- A database "schema" in the ER Model can be represented pictorially: ***ER diagrams***
- Can map an ER diagram into a relational schema

Basics of ER model

- **Entity**
  - A real-world object, distinguishable from other objects
  - An entity is described (in database) using a set of **attributes**
- **Entity Set**
  - A collection of similar entities. E.g., all employees
  - All entities in an entity set have the same set of attributes (Until we consider hierarchies, anyway!)
  - Each entity set has a ***key*** (underlined)
  - Each attribute has a ***domain***

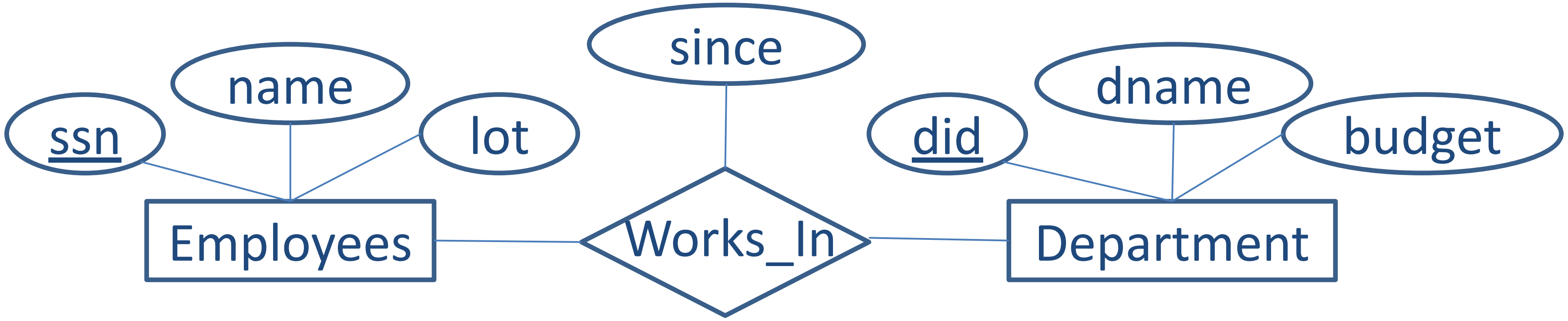


- **Instance of an entity set**



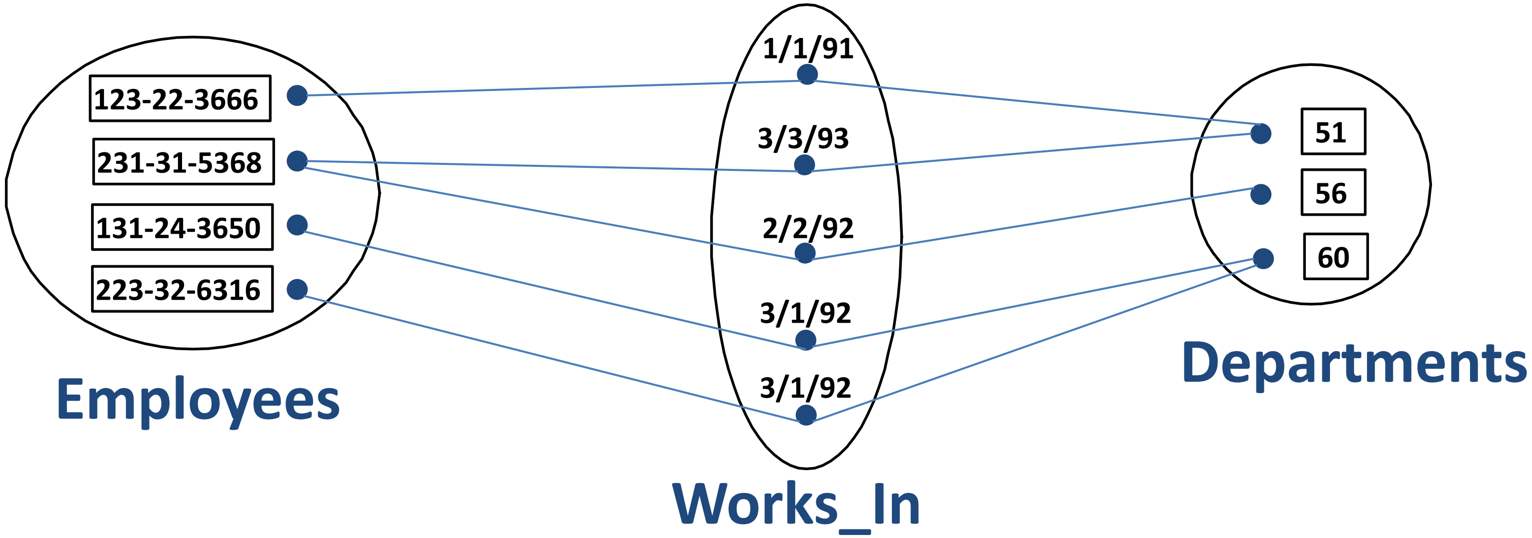
Basics of ER model (contd.)

- **Relationship**
  - Association among two or more entities. E.g., Fred works in the Pharmacy department
  - Can have their own attributes
- **Relationship set**
  - A collection of similar relationships. E.g., all employees working in some department



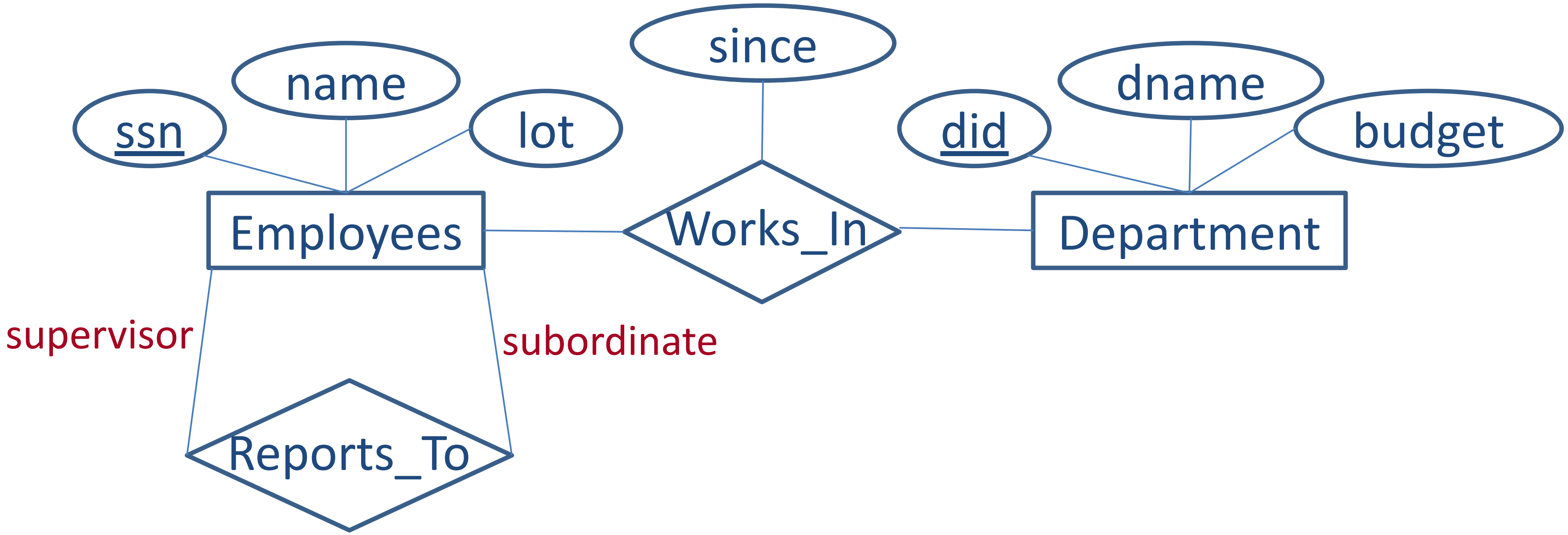
Basics of ER model (contd.)

- **Instance of a relationship**



Basics of ER model (contd.)

- Same entity set can participate in different relationship sets or in different "roles" in the same set

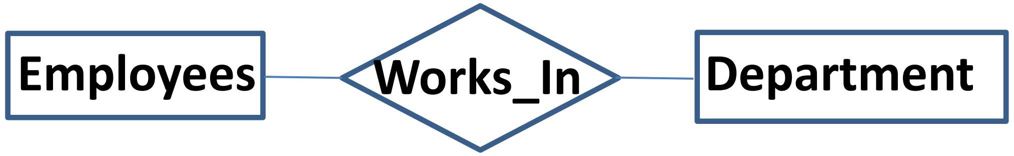


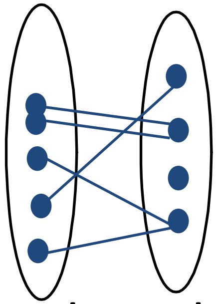
The ER model

- Basic ER modeling concepts
- **Constraints**

  Readings: Chapters 6.4
- Complex relationships
- Conceptual Design

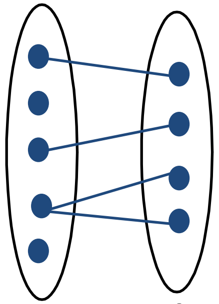
Key constraints





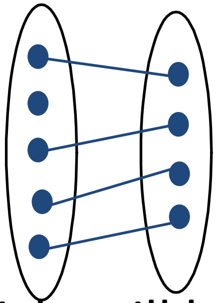
Many-to-Many

An employer can work in **many** departments; a department can have **many** employees



One-to-Many

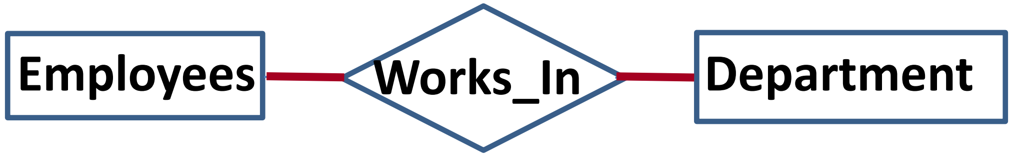
Each department has **at most one** manager, according to the **key constraint** on Manages



One-to-One

Each driver can drive **at most** vehicle and each vehicle will have **at most one** driver

Participation constraints



Total Participation

- Every Employee works in at least one department
- Every Department has at least one employee

- There can be Employees who are not managers
- An Employee can manage at most one department
- Every Department has exactly one manager

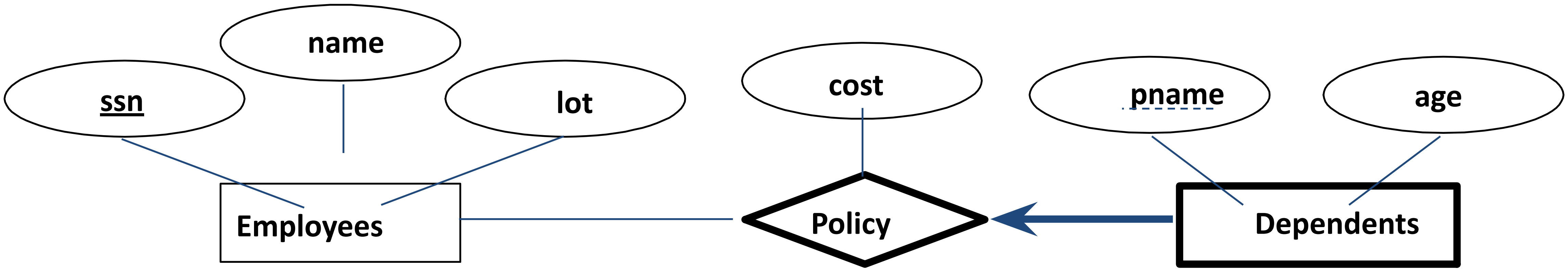
- There can be Customers who do not buy Products
- There can be Products not bought by any Customers

Partial Participation

Weak entities

A ***weak entity*** can be identified uniquely only by considering the primary key of another (*owner*) entity

- Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities)
- Weak entity set must have total participation in this ***identifying*** relationship set



Weak entities have only a "partial key" (dashed underline)

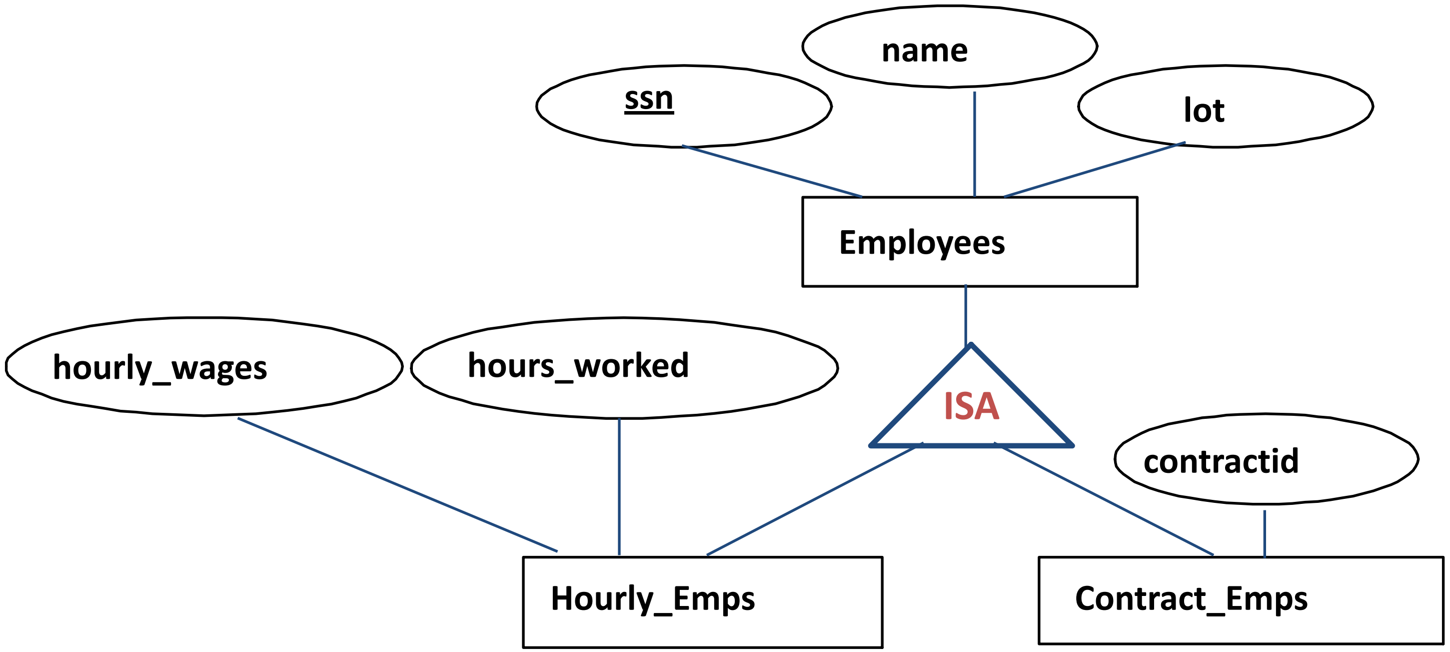
The ER model

- Basic ER modeling concepts
- Constraints
- **Complex relationships**

  Readings: Chapters 6.3, 6.8
- Conceptual Design

ISA ("is a") hierarchies

- As in C++, or other PLs, attributes are inherited
- If we declare A **ISA** B, every A entity is also considered to be a B entity

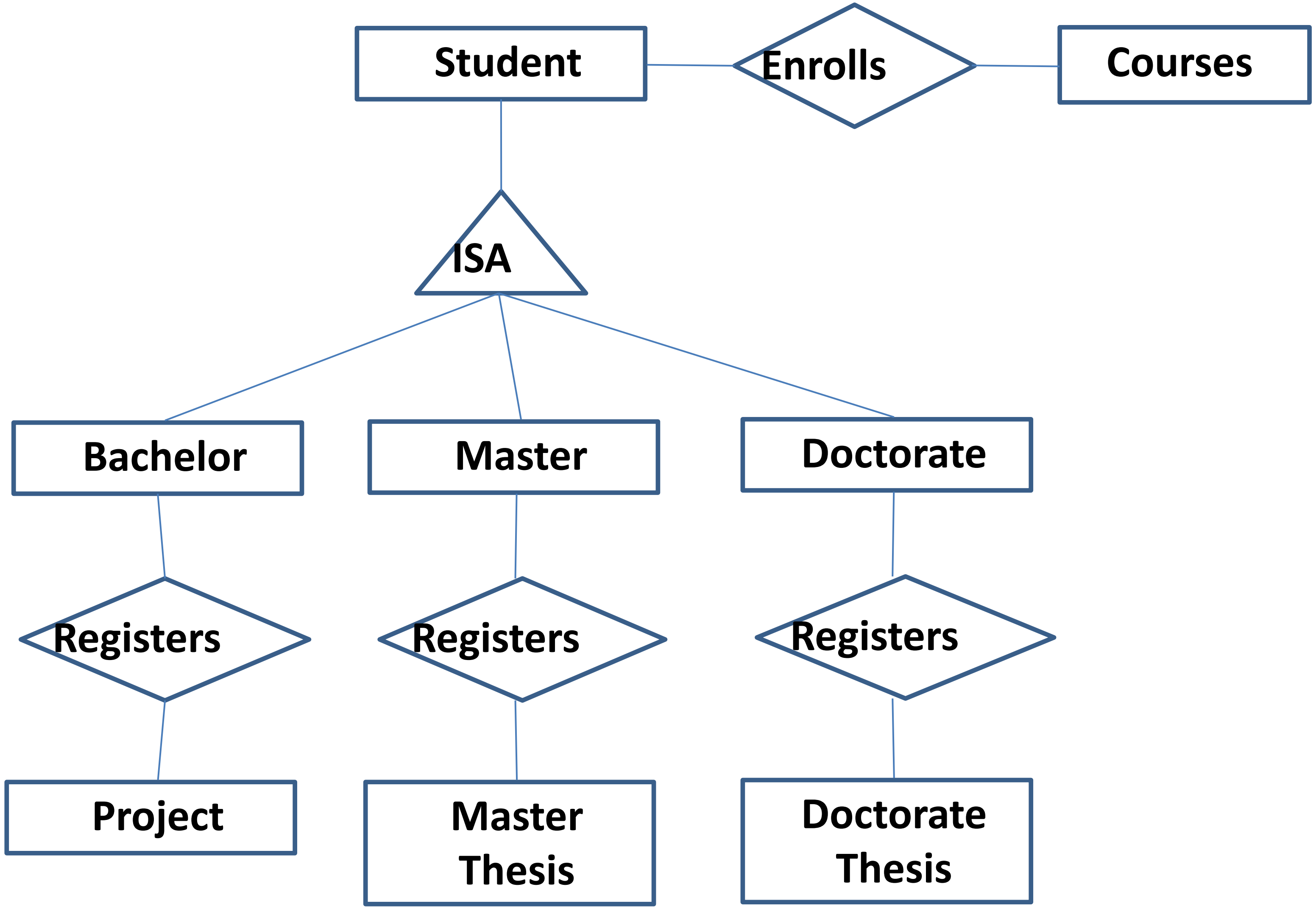


ISA ("is a") hierarchies

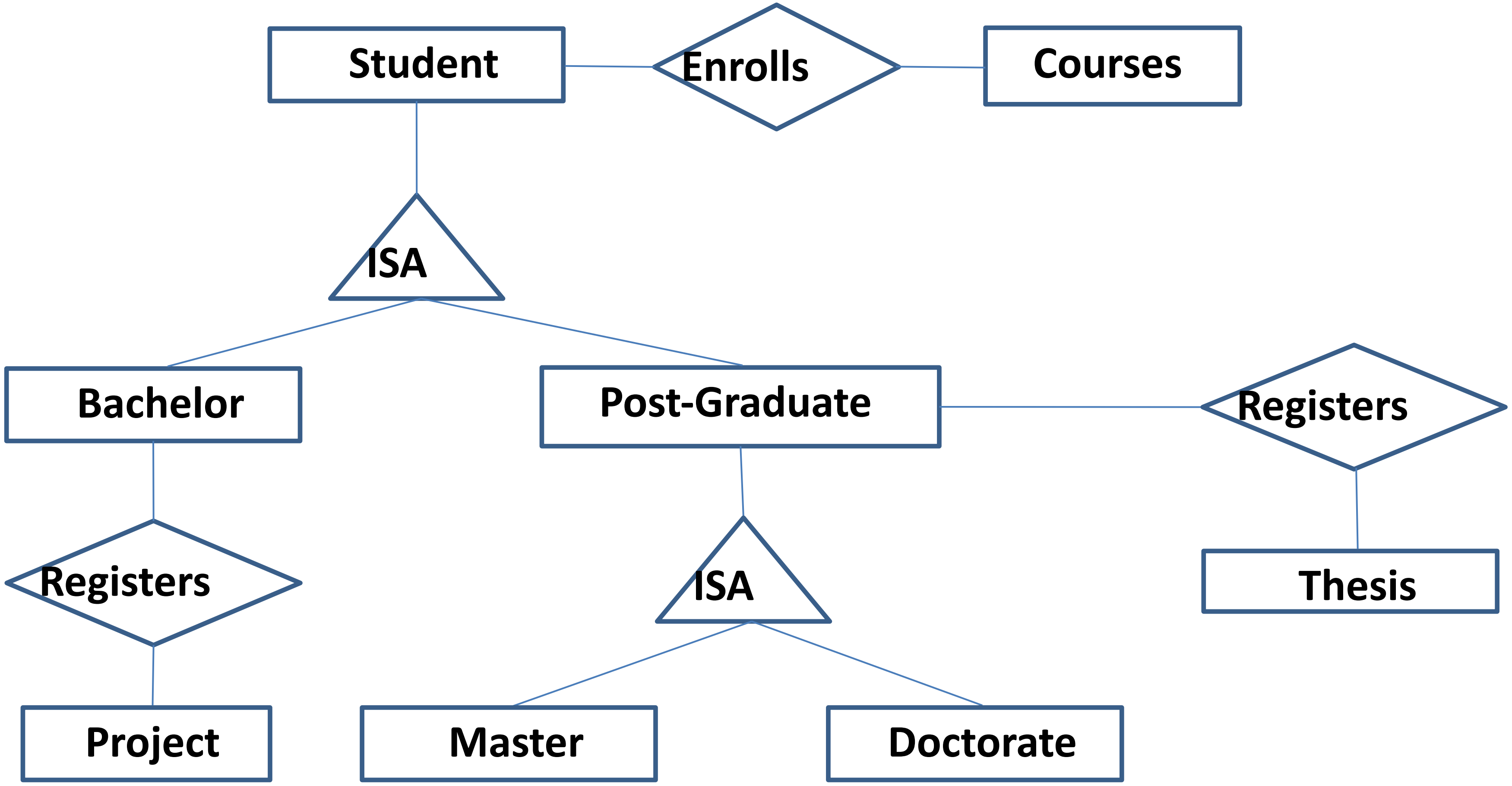
- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/Disallowed*)
- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/No)*

- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass. (i.e., not appropriate for all entities in the superclass)
  - To identify entities that participate in a particular relationship (i.e., not all superclass entities participate)

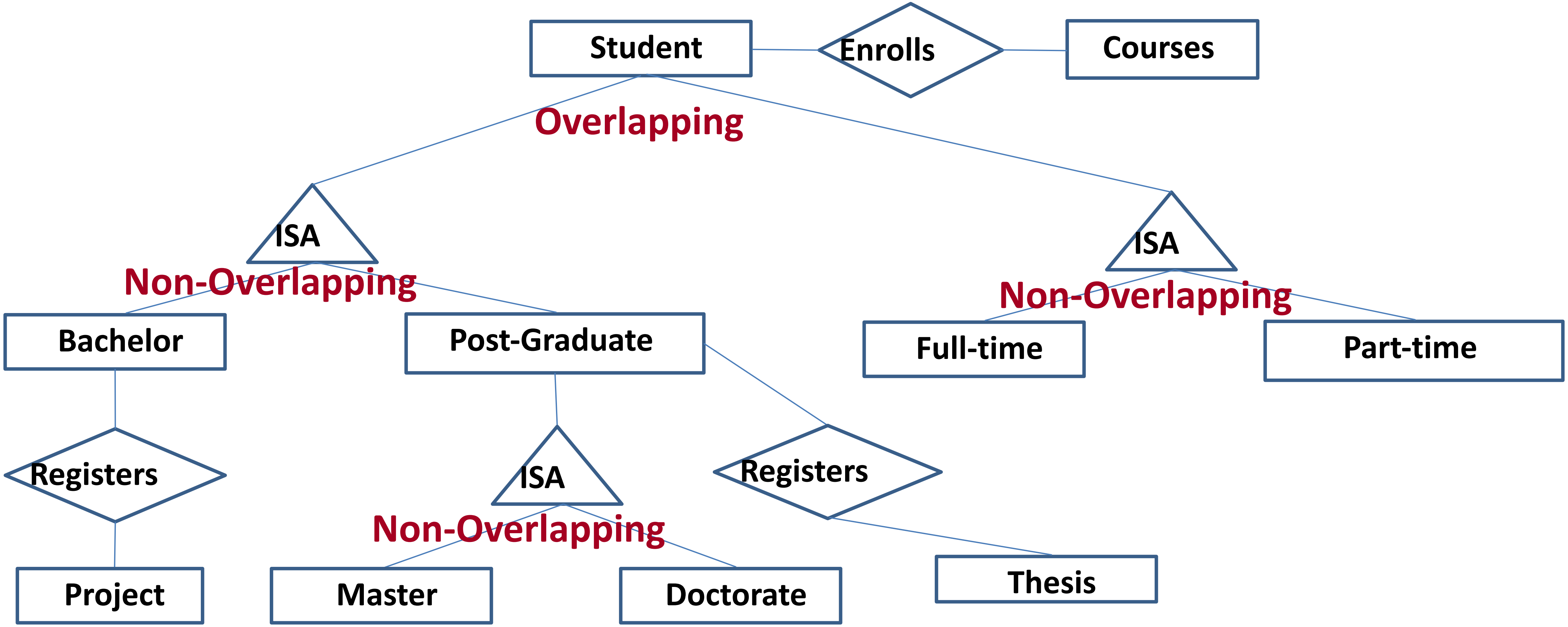
ISA ("is a") hierarchies



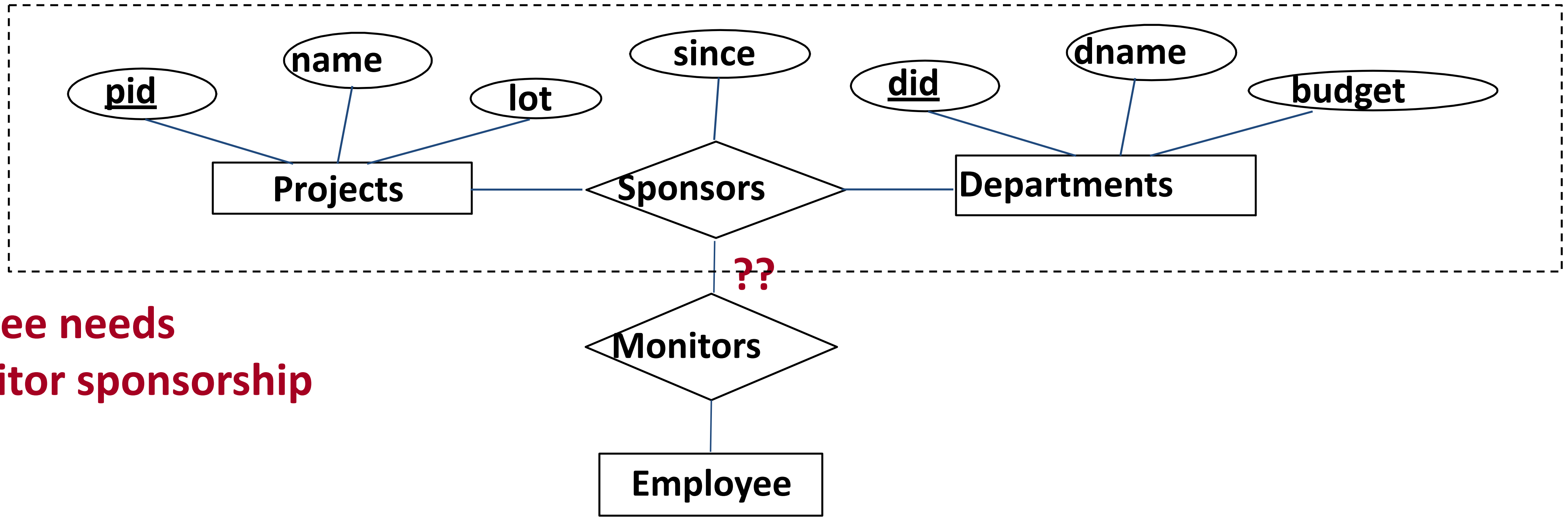
ISA (“is a”) hierarchies



ISA (“is a”) hierarchies



Aggregation



Employee needs to monitor sponsorship

- Used to model a relationship involving a *relationship set*
- Allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships

The ER model

- Basic ER modeling concepts
- Constraints
- Complex relationships
- **Conceptual Design**

Readings: Chapter 6.9

Conceptual design using the ER model

- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?

- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured
  - But some constraints cannot be captured in ER diagrams

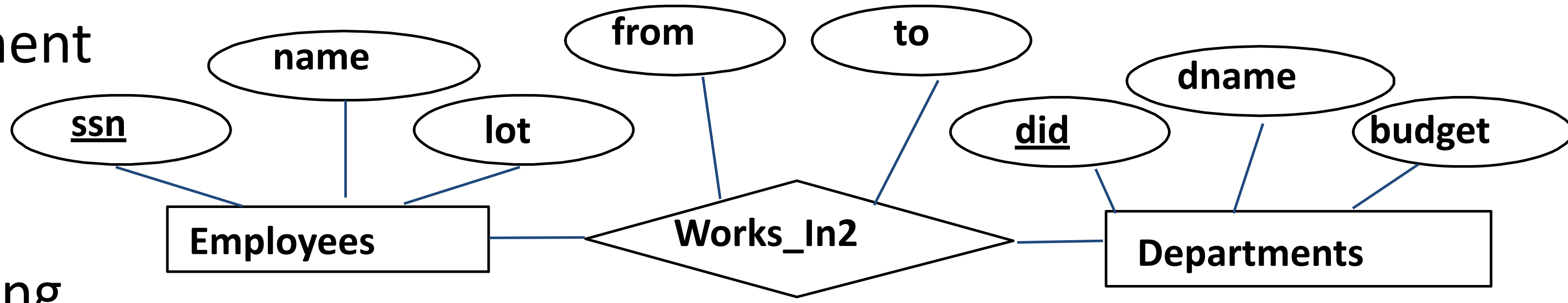
Entity vs attribute

- Should *address* be an attribute of Employees or an entity (related to Employees)?

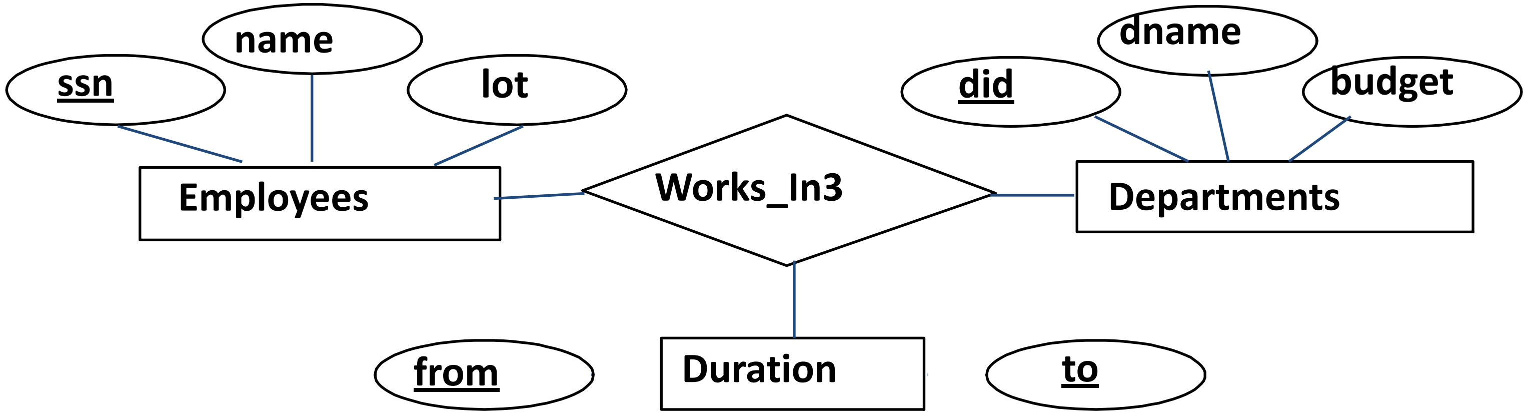
- Depends upon how we want to use address information, and the semantics of the data:
  - If we have several addresses per employee, the *address* must be an entity (since attributes cannot be set-valued)
  - If the structure (city, street, etc.) is important, the *address* must be modeled as an entity (since attribute values are atomic)

Entity vs attribute (contd.)

- Works_In2 does not allow an employee to work in a department for two or more periods

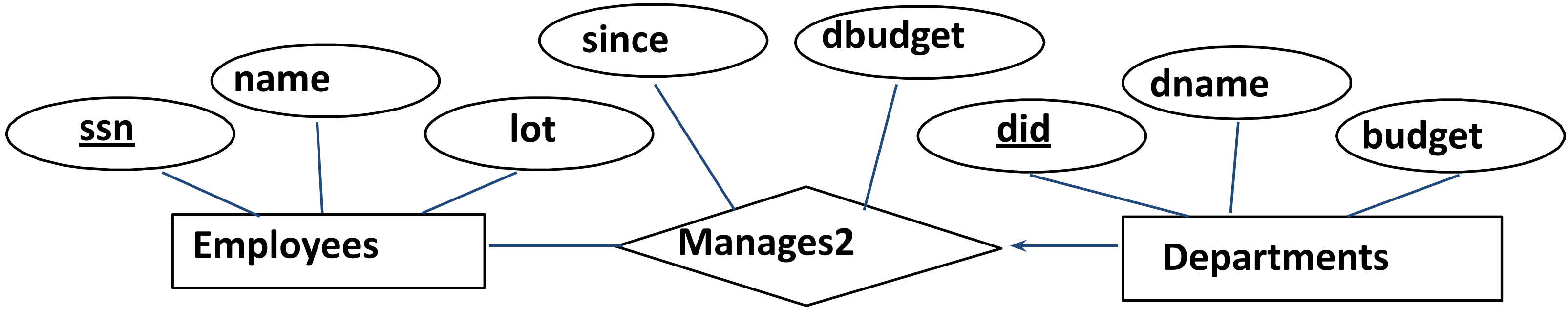


- Similar to the problem of wanting to record several addresses for an employee: we want to record *several values of the descriptive attributes for each instance of this relationship*

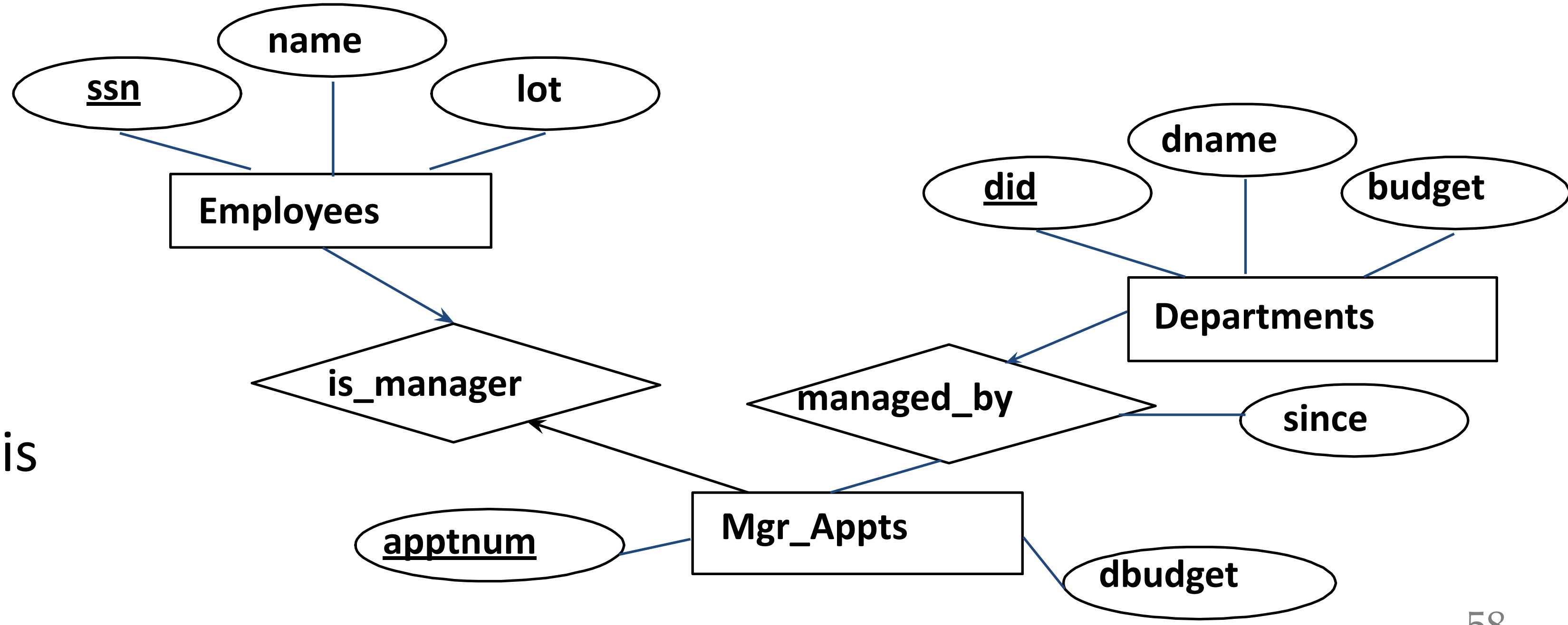


Entity vs. relationship

- OK as long as a manager gets a separate discretionary budget (*dbudget*) for each department



- What if the manager's *dbudget* covers *all* managed departments? (can repeat value, but such redundancy is problematic)



ER design notes

- ER design is *subjective*. There are often many ways to model a given scenario!

- Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, aggregation.

- Many types of constraints (notably, *functional dependencies*) cannot be expressed, although constraints play an important role in determining the best database design for an enterprise
- Other modeling languages available, e.g. UML

Summary

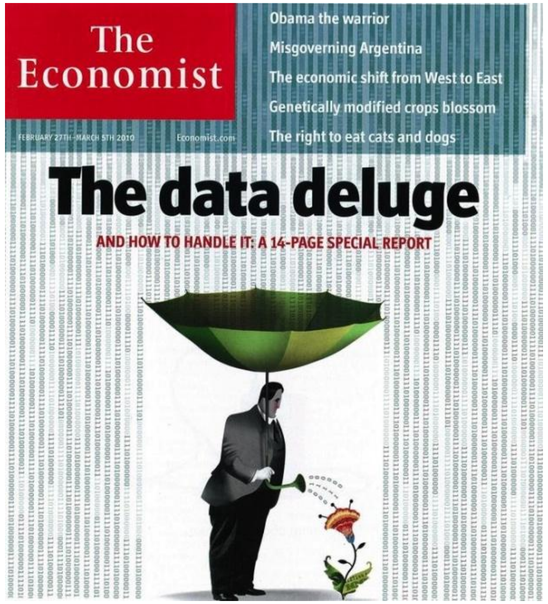
- A **DBMS** maintains, queries large data sets
  - Can manipulate data and exploit *semantics*
- Other benefits include: recovery, concurrency, quick application development, and date integrity and security
- Levels of abstraction provide data independence

Reading Material: Chapter 1 from textbook

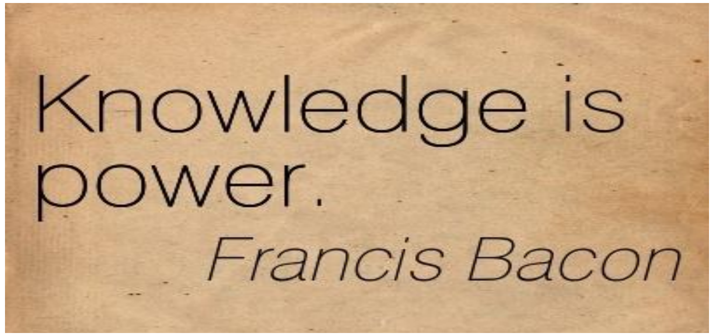
- ER models are commonly used for conceptual design in DB
  - Constructs are expressive, close to the way people think about their applications
- Basic constructs: entities, relationships, and attributes (of entities and relationships)
- Some additional constructs
  - Weak entities, ISA hierarchies, and aggregation

Backup Slides

Is **data** the new God?



Why study Databases?



Inter-disciplinary

- Shift from **computation** to **information**
  - always true for corporate computing
  - Web made this point for personal computing
  - more and more true for scientific computing

- Need for DBMS has exploded in the last years
  - Corporate: “supply chain mgmt”, “data analytics”, etc.
  - Scientific: Digital humanities, Human Brain

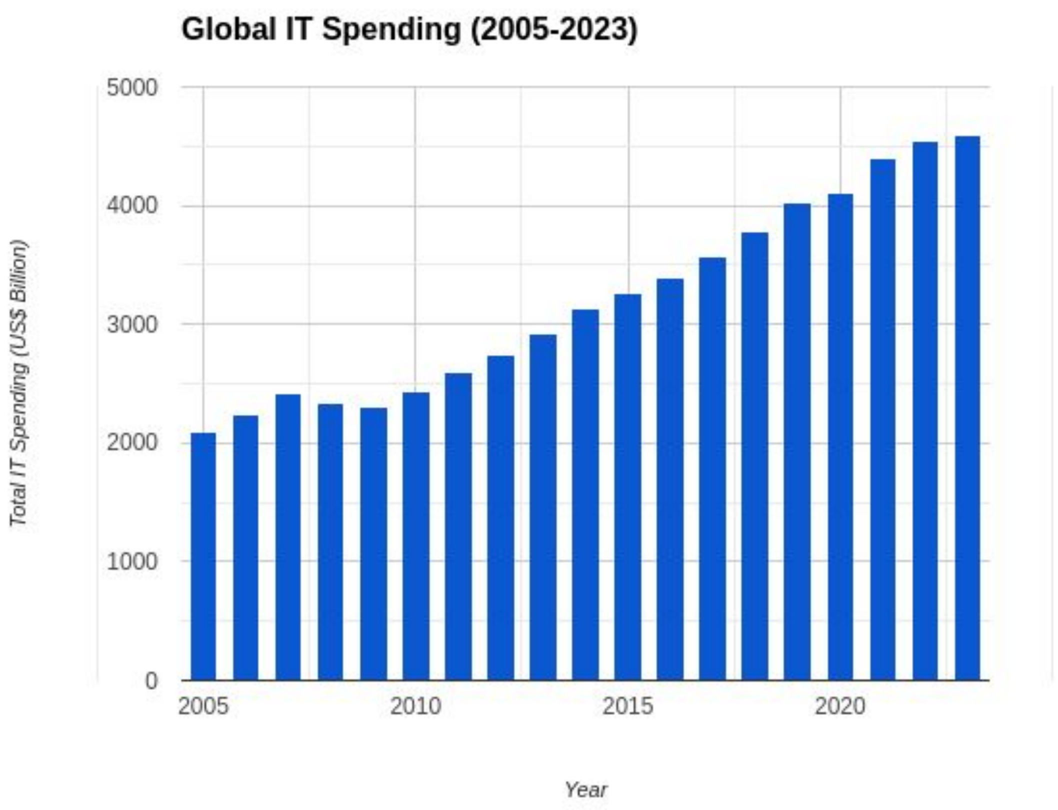
Why study Databases?

- Intellectually challenging
- DBMS encompasses much of CS in a practical discipline
  - OS, languages, theory, multimedia, logic
  - Focus on real-world apps
- It is fun!

- Lot of career opportunities
  - Oracle, IBM, Microsoft, SAP, Teradata,…
  - $24 billion earned by database companies (2011)
  - Larry Ellison, CEO of Oracle
    - 2014 Forbes 6th richest person - $48 billion

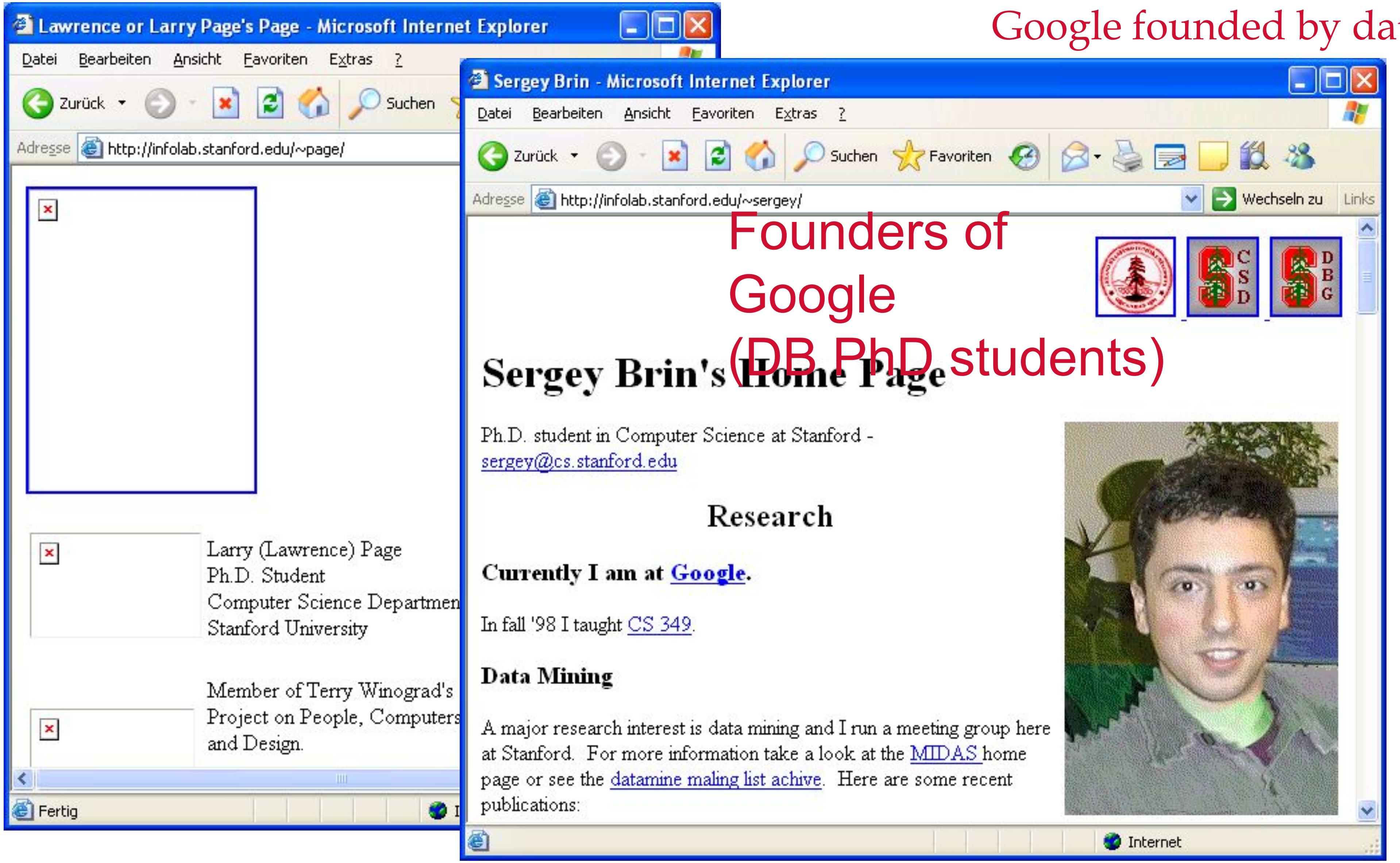
Global IT spending

Spending on database systems is just the tip of the iceberg: add database application software, hardware infrastructure, consulting, … = IT



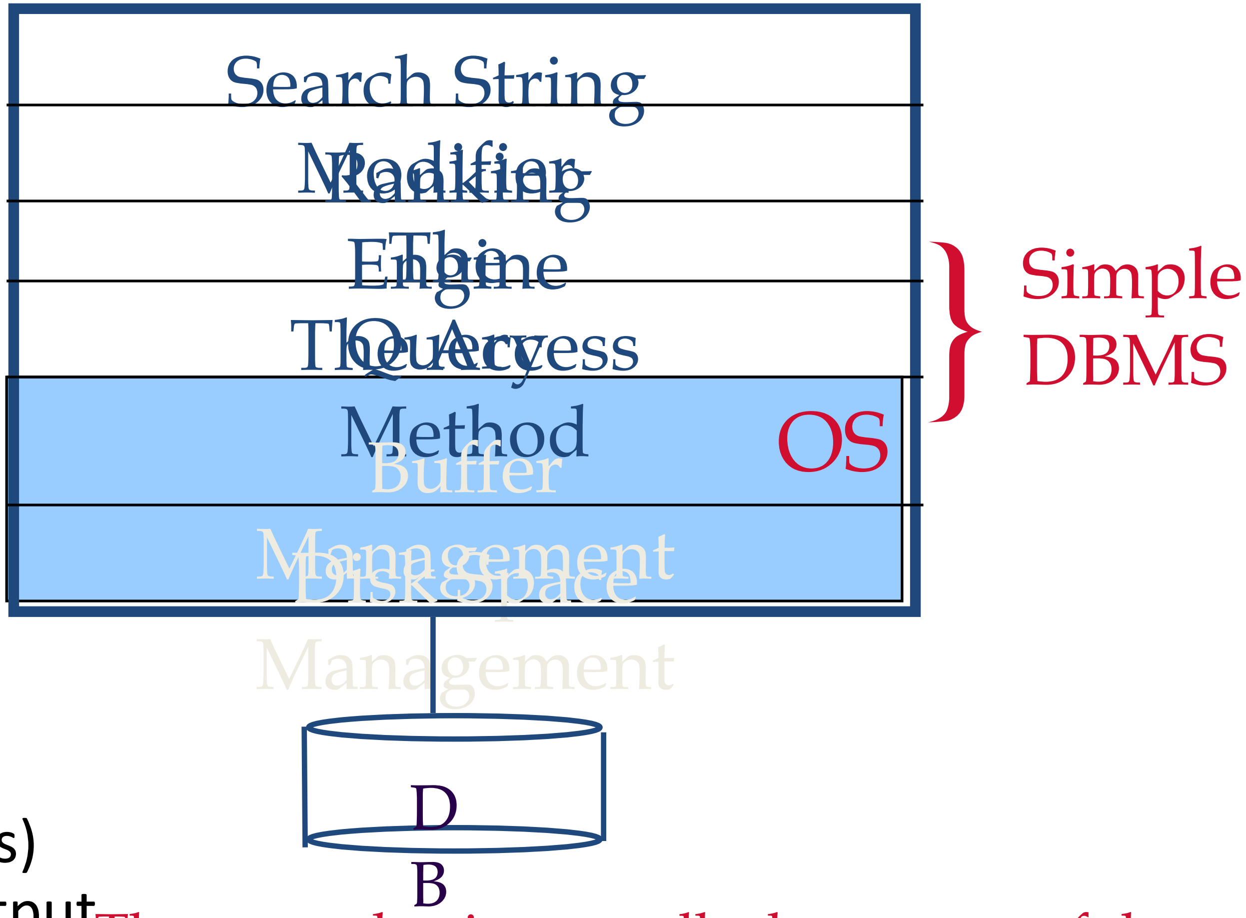
Why study Databases?

Google founded by database PhD students



FYI: A text search engine

- Less "system" than DBMS
  - Uses OS files for storage
  - Just one access method
  - One hardwired query
    - regardless of search string
- Typically no concurrency or recovery management
  - Read-mostly
  - Batch-loaded, periodically
  - No updates to recover
  - OS a reasonable choice
- Smarts: text tricks
  - Search string modifier (e.g. "stemming" and synonyms)
  - Ranking Engine (sorting the output, e.g. by word or document popularity)
  - no semantics: WYGIWIGY



There may be time to talk about some of these text tricks in this class, but it won't be a focus.

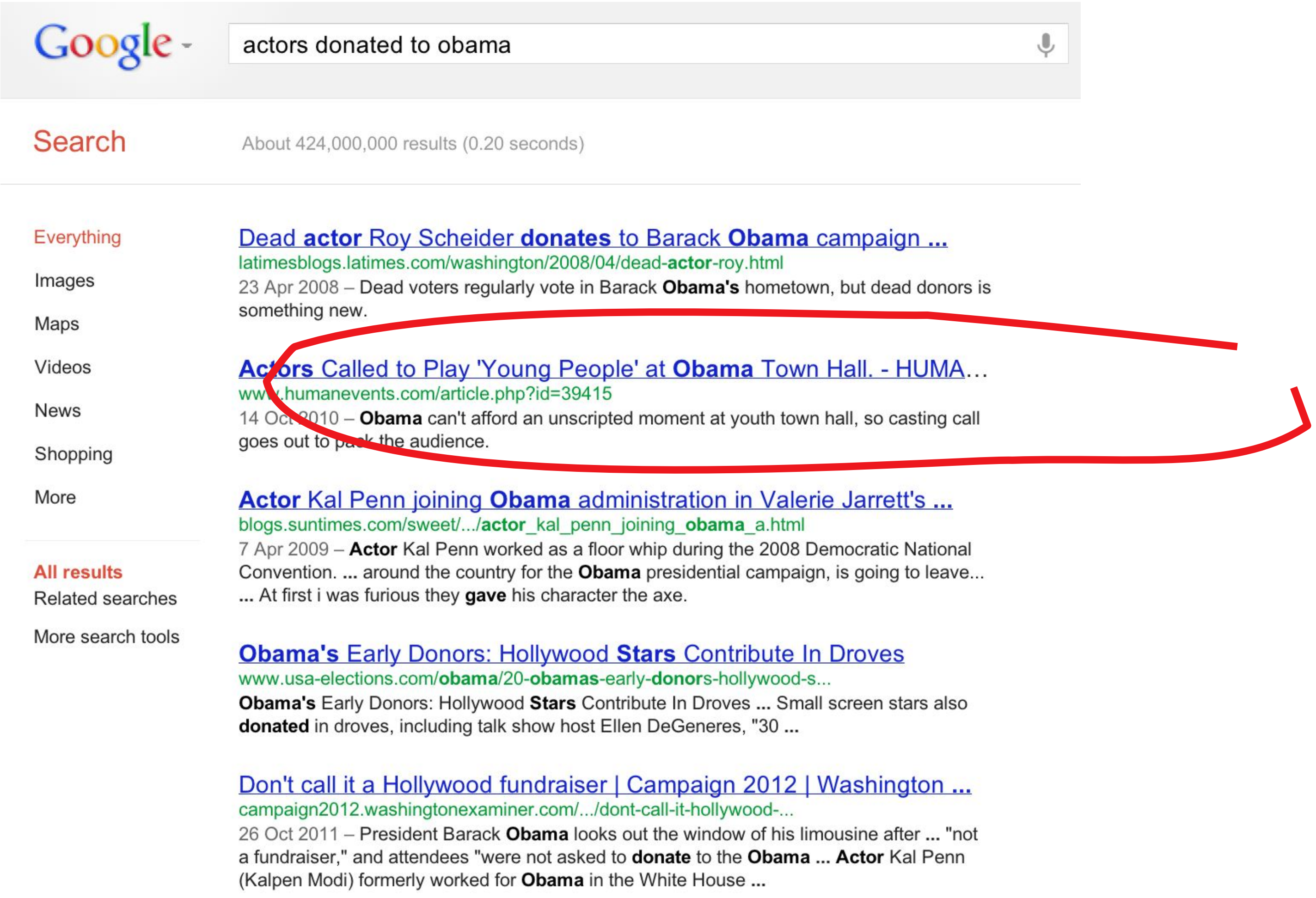
About the course – Who will benefit?

- Database Designers
- Application Developers
  - Use database as a black box
- Database Administrators
  - Maintain & repair
- Database Developers
  - Design & build

"Search" vs. Query

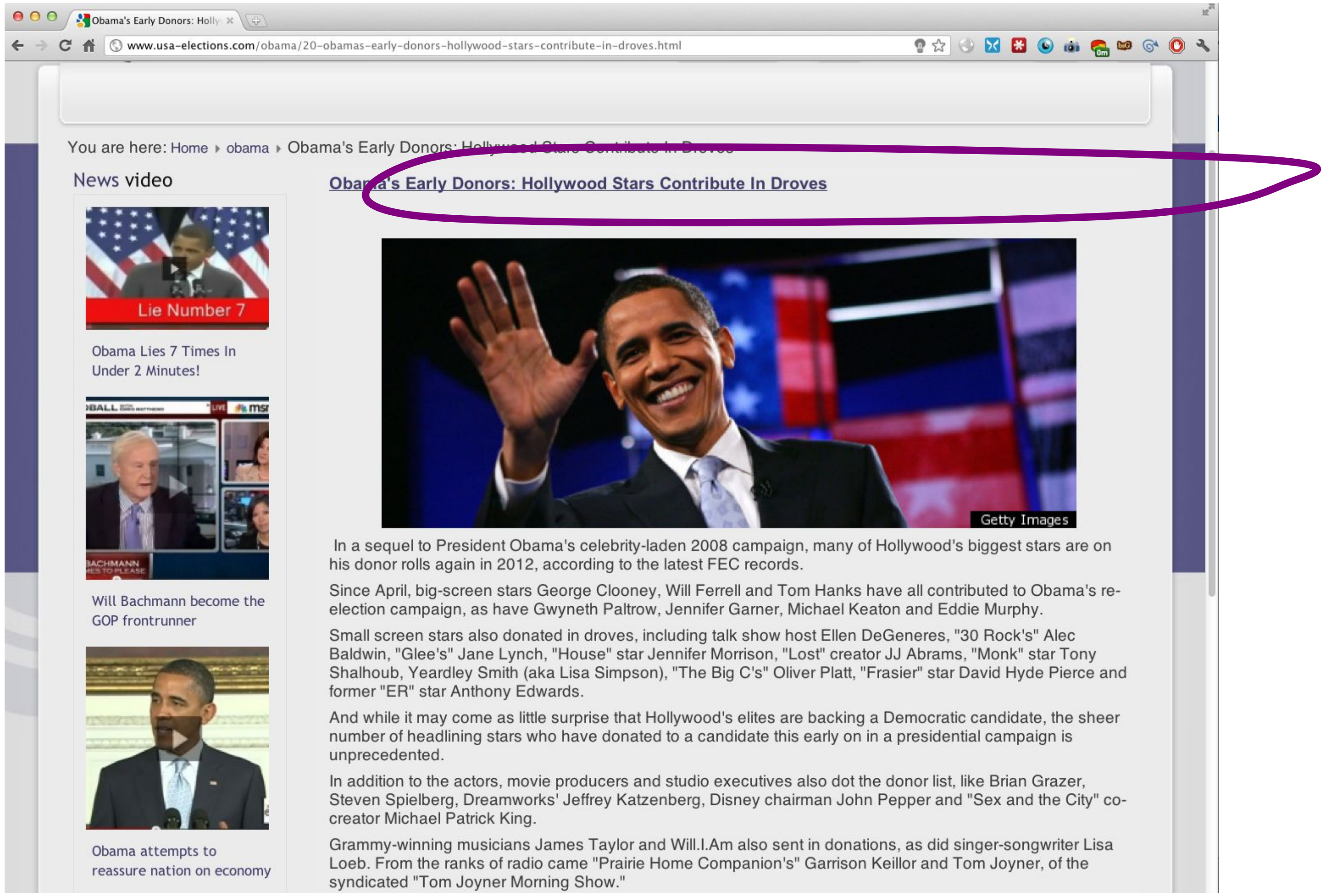
- What if you wanted to find out which actors donated to Barrack Obama's presidential campaign?

- Try "actors donated to obama" in your favorite search engine.

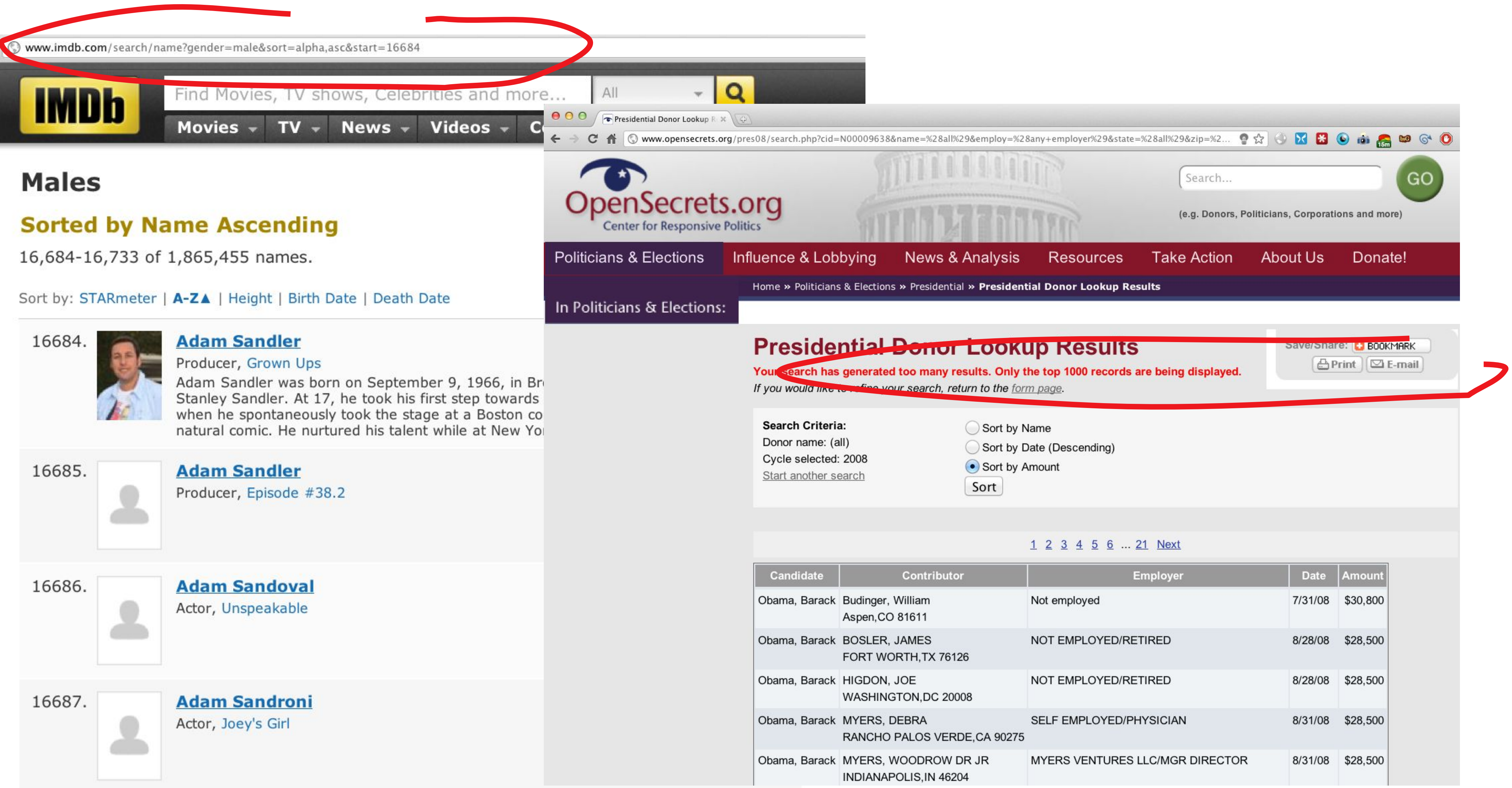


“Search” vs. Query

- “Search” can return only what’s been “stored”
- E.g., best match at Google:



A “Database Query” Approach



"IMDB Actors" JOIN "OpenSecrets"

Contributor	Employer	Date	Amount
ROCK, CHRIS MR NEW YORK,NY 10019	ACTOR	4/20/07	$9,200
DOUGLAS, MICHAEL UNIVERSAL CITY,CA 91608	ACTOR/ PRODUCER	3/30/07	$4,600
DOUGLAS, MICHAEL UNIVERSAL CITY,CA 91608	ACTOR/ PRODUCER	3/30/07	$2,300
ROCK, CHRIS MR NEW YORK,NY 10019	ACTOR	4/20/07	$2,300
CARIDES, GEORGIA NEW YORK,NY 10017	ACTOR	5/18/07	$1,000
CARTER COVINGTON, CLAUDIA CHARLOTTE,NC 28207	ACTORS THEATRE PART TIME/ACTOR/NEW	5/20/08	$1,000
FOX, RICK ENCINO,CA 91316	ACTOR/PRODUCER	6/16/08	$1,000
HILDRETH, THOMAS W LOS ANGELES,CA 90068	ACTOR	9/29/08	$1,000
RENNER, CARL BEVERLY HILLS,CA 90210	ACTOR/BESSONE@ROADRUNNER.COM	8/28/08	$1,000
SIMMONS, HENRY WEST HOLLYWOOD,CA 90046	ACTOR	6/4/07	$1,000

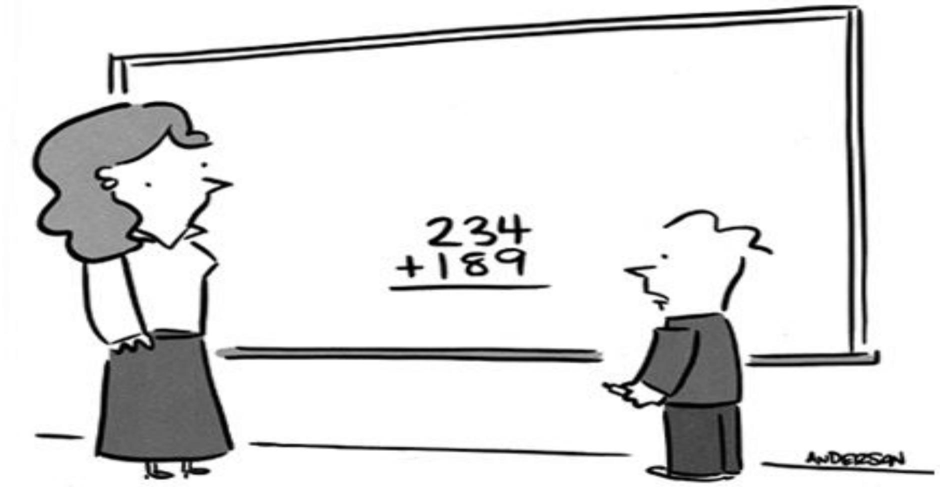
Is the "classic" DBMS all there is?

Not at all…

- Semi-structured and unstructured data, images
- Standards like XML, RDF help with data modeling
- DBMS-like functionality *across multiple web sites*
- Data streams
- NoSQL movement: No transactions, focus on queries
- Querying raw data before even seeing it – NoDB
- Database queries in large datacenters

…exciting times!

Questions?/Comments!



"Does this count as big data?"

Coming up next…

Conceptual Modeling: Entity-Relation Models

DATA-INTENSIVE APPLICATIONS AND SYSTEMS

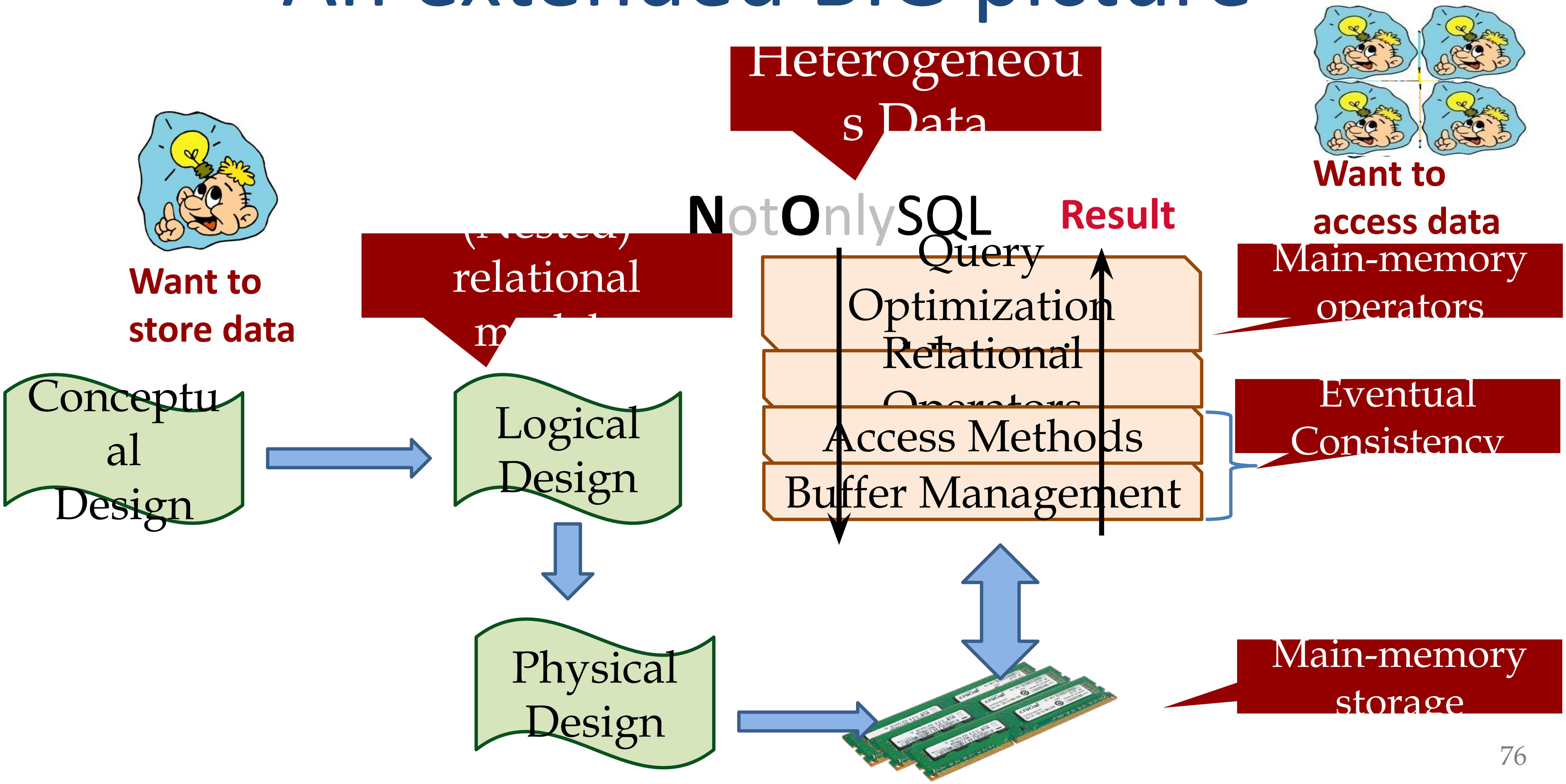
EPFL

OS Support for Data Management

- Data can be stored in RAM
  - this is what every programming language offers!
  - RAM is fast, and random access
  - Isn’t this heaven?

- Every OS includes a File System
  - manages *files* on a magnetic disk
  - allows *open, read, seek, close* on a file
  - allows protections to be set on a file

An extended BIG picture



250+ venture capital backs new DB startups!