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 Management Systems** by Gehrke and Ramakrishnan
- Course organization: Lectures, exercises, project (labs), quiz, midterm and final
- Staff email: cs300-staff@groupes.epfl.ch

- Instructors: **Anastasia Ailamaki** and **Sanidhya Kashyap**
- TAs:
  - Eleni Zapridou, Ioanna Tsakalidou, Musa Unal,
    Nguyne Cao Pham, Tao Lyu, Mahdi Hosseini and Yueyang Pan
- AEs:
  - Michael Koepf, Oussama Gabouj, Kostas Chasialis, Barghorn Jeremy

Schedule

- Lectures: mondays 11-13 in CE 1 4
- Exercises&Labs: wednesdays in CE 1 3

- Quizzes: check understanding of material
- Labs: graded programming exercises
- All exams and deliverables are online
- Detailed descriptions available
    - on moodle
    - during exercises/project
    - during office hours

Mon	Tue	Wed
Feb 19 **LEC 1:** Intro, overview: ER & Relational Model	Feb 20	Feb 21 **TUT 1:** SQL 1 **Lab session 1**
Feb 26 **LEC 2:** Relational Algebra & SQL **Lab 1**: : RDBMS use with SQL	Feb 27	Feb 28 **TUT 2:** SQL 2 **Lab session 2**
Mar 4 **LEC 3:** File Systems & File Layouts (DSM/NSM/PAX)	Mar 5	Mar 6 **TUT 3:** ER - Relational model (translation) **Lab session 3**
Mar 11 **LEC 4:** Storage hierarchy	Mar 12	Mar 13 **Lab session 4**
Mar 18 **LEC 5:** Indexes & Memory	Mar 19	Mar 20 **Lab session 5**
Mar 25 **LEC 6:** Midterm **DUE:** Lab1 **Lab 2**: : Buffer pool	Mar 26	Mar 27 **Lab session 6**
Apr 1 **LEC 7:** Spring break	Apr 2	Apr 3 **Lab session 7**
Apr 8 **LEC 8:** Hashing and Sorting & storage hierarchy	Apr 9	Apr 10 **Lab session 8**
Apr 15 **LEC 9:** Query Operators I	Apr 16	Apr 17 **Lab session 9**
Apr 22 **LEC 10:** Query Operators II	Apr 23	Apr 24 **Lab session 10**
Apr 29 **LEC 11:** Query Optimization **DUE:** Lab2 **Lab 3**: : Index	Apr 30	May 1 **Lab session 11**
May 6 **LEC 12:** Transactions and Concurrency Control & Concurrency I	May 7	May 8 **Lab session 12**
May 13 **LEC 13:** Concurrency Control and Eventual Consistency & Concurrency II	May 14	May 15 **Lab session 13**
May 20 **LEC 14:** No lecture	May 21	May 22 **Lab session 14**
May 27 **LEC 15:** Parallel and Distributed data systems **DUE:** Lab3	May 28	May 29 **Lab session 15**

Logistics

- Course information: **Moodle**
- Course discussion: **Ed**
- Programming labs: **Github classroom**
- Lectures: In-person lectures in **CE4**
  - Attendance is **strongly recommended**
  - Lecture recordings will probably be available for occasional use

- Grades: 3 labs (30%) + midterm (25%) + final exam (40%) + quiz (5%)
  - Each lab contributes 10%
  - Weekly quiz: Best 10 out 12

Exams

- Online quizzes on Moodle
  - Quiz is released after Monday lecture ends
  - Lasts an hour and can be done until Friday 5pm
- Online exams on Moodle
  - Includes questions from both lectures and labs
- Midterm: Monday march 25, 11am
  - Covers weeks 1–5

- Final: Date / time / place TBD
  - Assumes content from weeks 1-5 is known
  - Covers lecture and lab contents from week 7–14

Time management

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

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

Academic honesty

You are allowed to collaborate, **EXCEPT** on labs, quizzes, and exams.

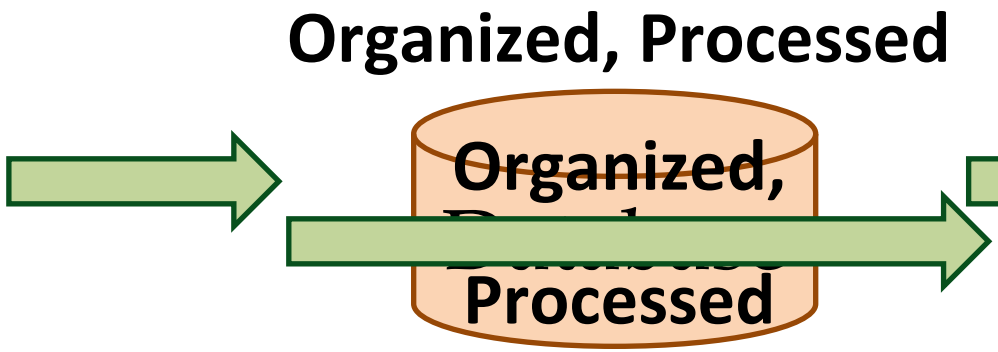
You must adhere to the EPFL academic honesty policies :

- 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?



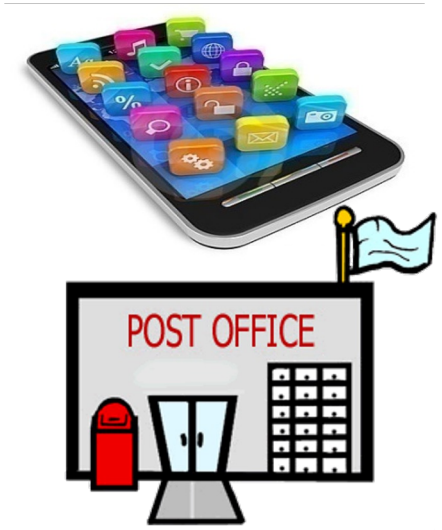
- Facts
- Basis for reasoning/discussion/ calculation
- Useful or irrelevant or redundant
- Must be processed to be meaningful

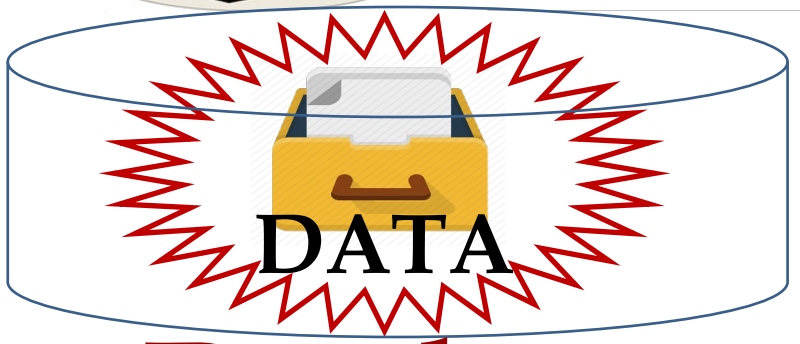
- Has meaning
- Relevant to the problem
- Actionable – leads to a solution

Knowledge

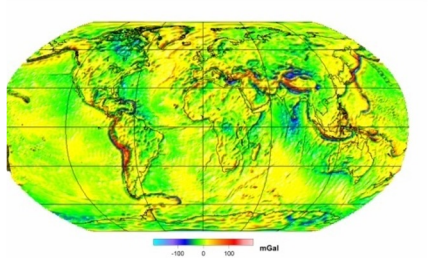
Wisdom

Have you ever “used” a database?

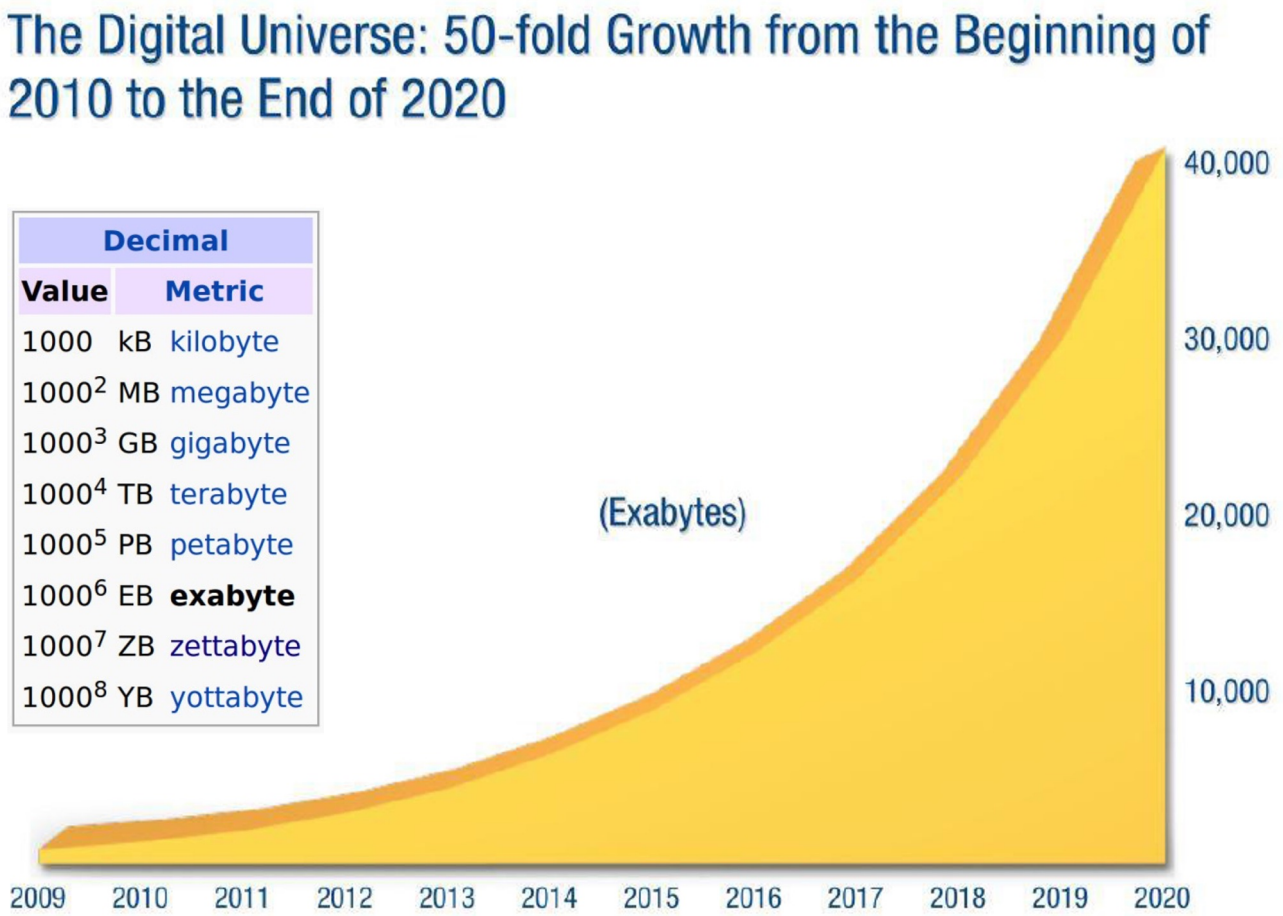




Database

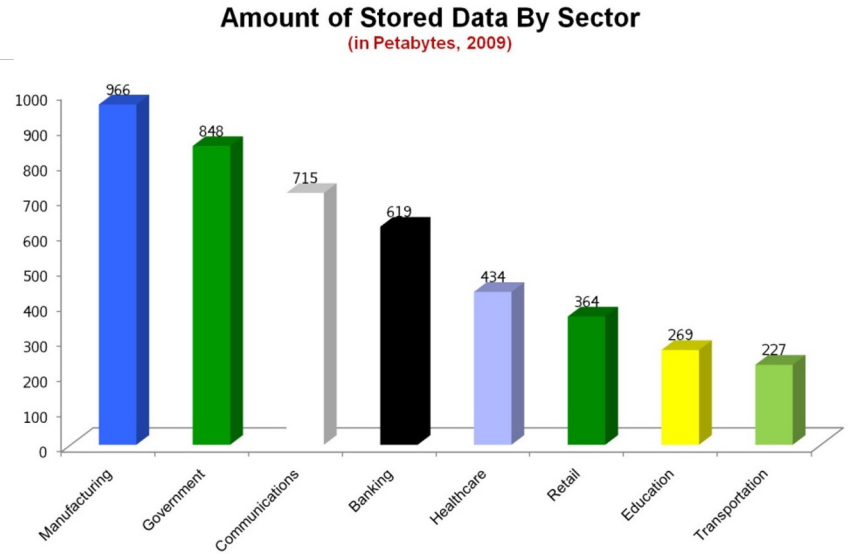


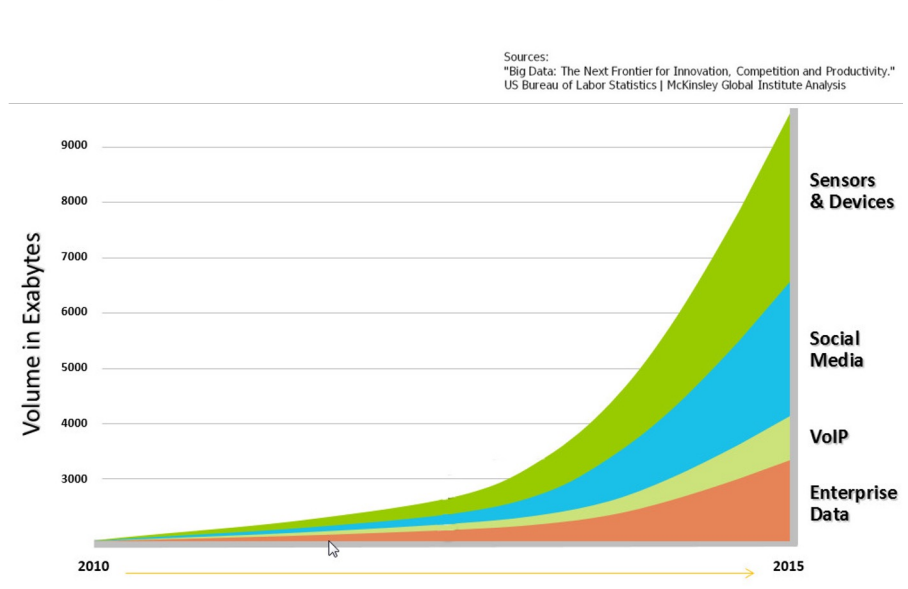
How big is “all” data?



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

This IDC graph predicts exponential growth of data from around 3 zettabytes in 2013 to approximately 40 zettabytes by 2020. An exabyte equals 1,000,000,000,000,000,000 bytes and 1,000 exabytes equals one zettabyte. Source: IDC's Digital Universe Study, December 2012, http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf.





http://www.tech-dynamics.com/wp-content/uploads/2014/02/BigDataChart.png

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

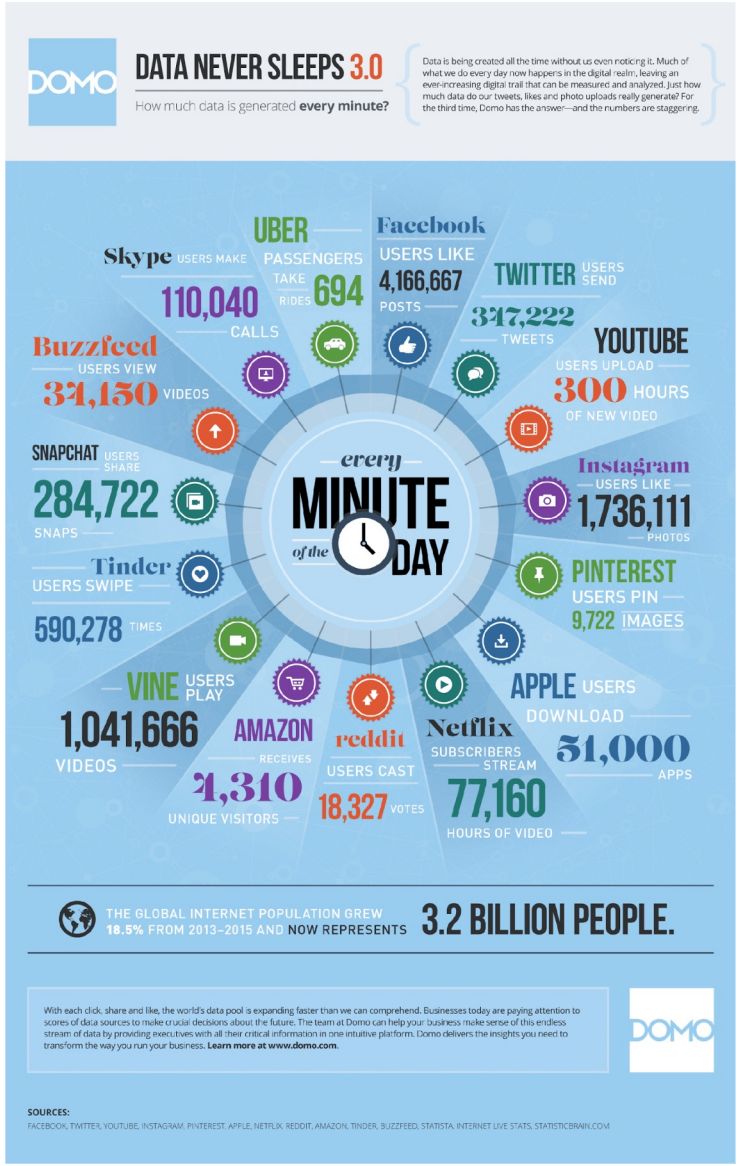
at&t

- 323 terabytes of information
- 1.9 trillion phone call records

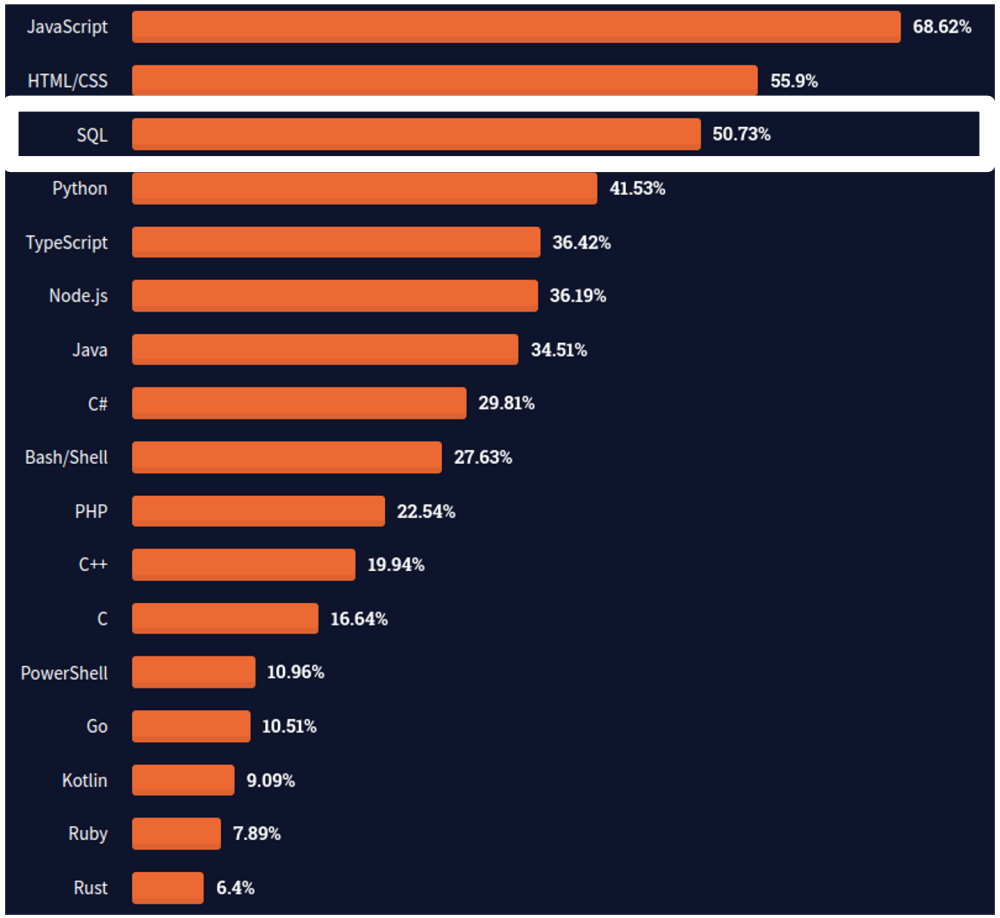
NERSC
NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

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

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

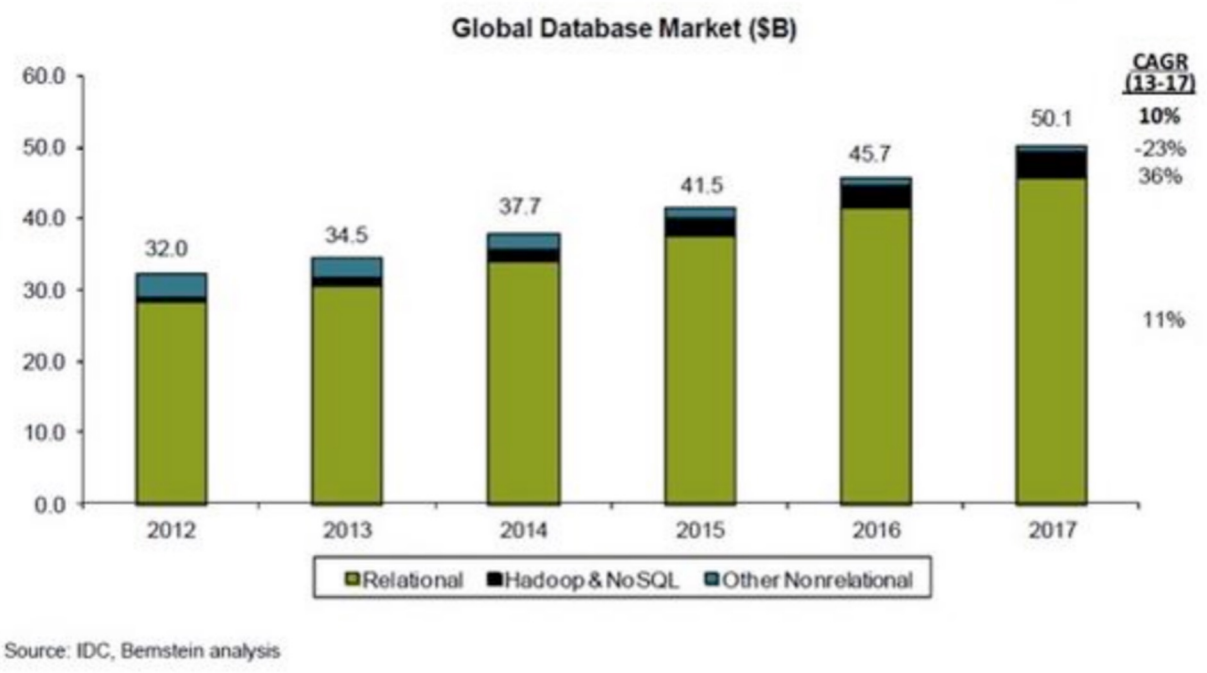


Why study Databases?



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

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 Exabytes everyday

Data outlives the programs that operate on it

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

Entities

Relationships

Question: Is the web a DBMS?

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

- However, currently
  - 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)

Question: 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) ???**

Question: Is your file system a DBMS?

- Thought experiment 1:
  - You and your project partn… th…
  - You h…
  - Wh…

A… ???

- Th…
  - The po…
  - Which of your changes will surv…

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

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

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***
- **Small matter of programming (SMOP)? Not really …**
  - As we will see in this semester
  - 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?

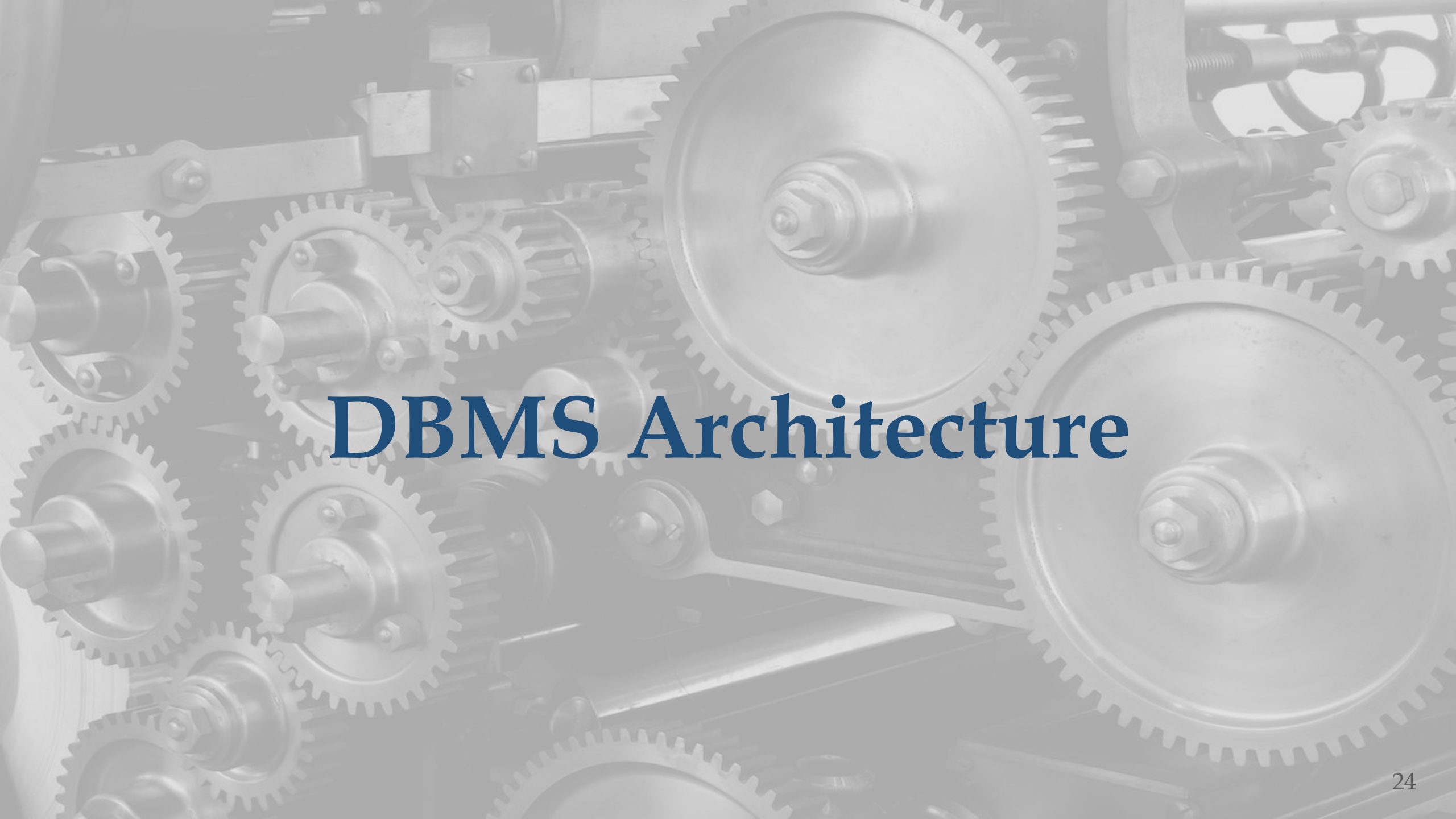
- **Representation information**
  - Data modeling
- **Languages and systems for querying data**
  - Complex queries with *real semantics*
  - Over massive amount of data
- **Concurrency control for data manipulation**
  - Controlling concurrent accesses
  - Ensuring transactional semantics
- **Reliable 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

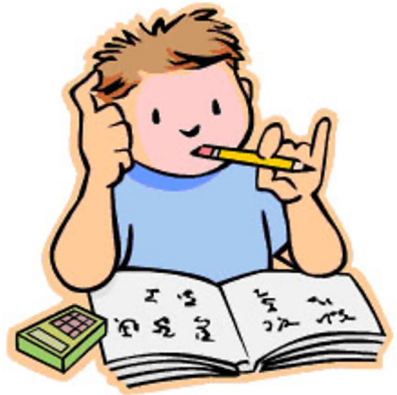


DBMS Architecture

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

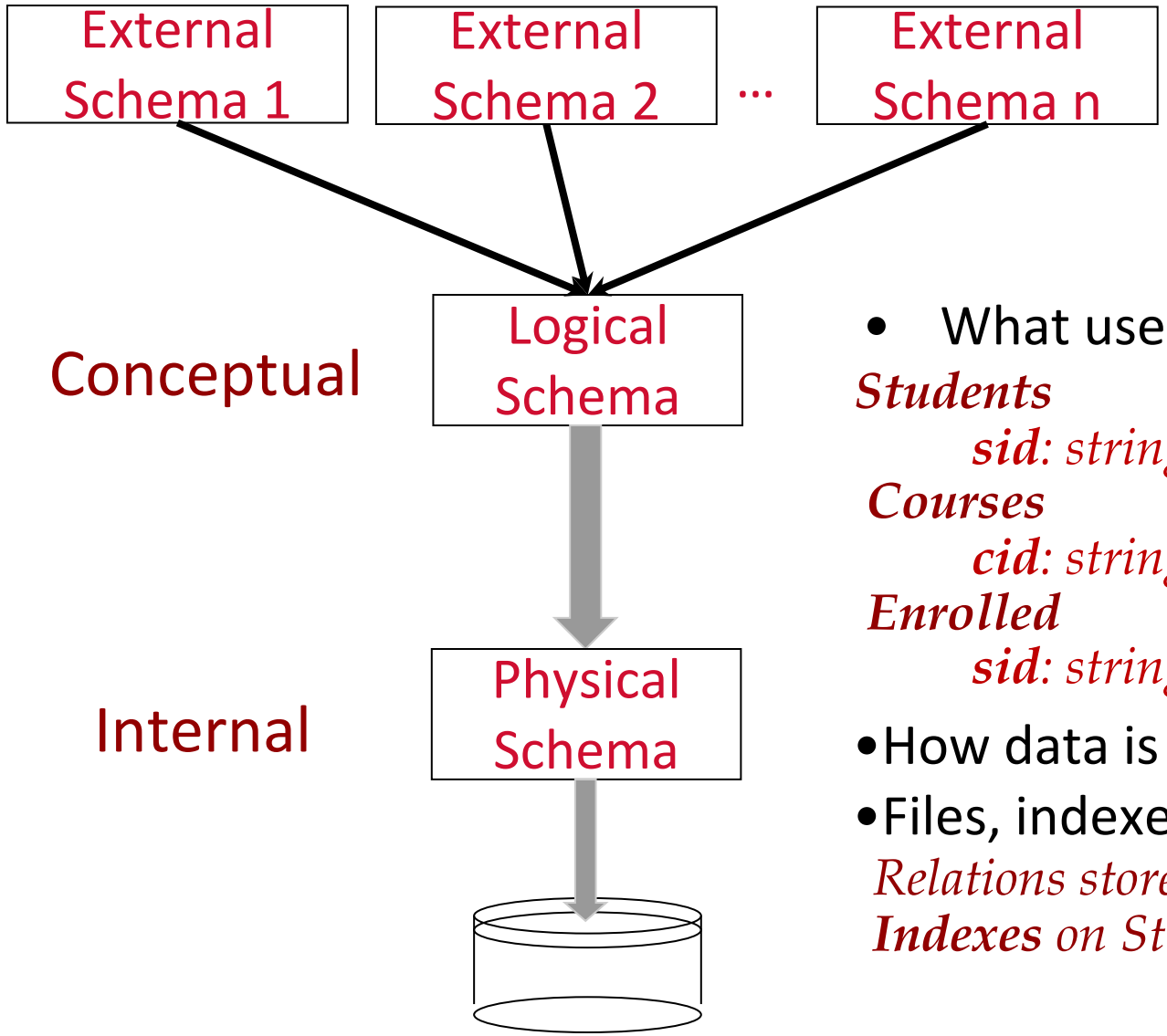
Enrolled

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

Want to store data

Conceptual Design

ER Models

ER to Relational

Relational Model

Logical Design

Indexing

Physical Design

Relational Algebra, SQL

SQL

Result

Query Optimization and Execution

Relational Operators

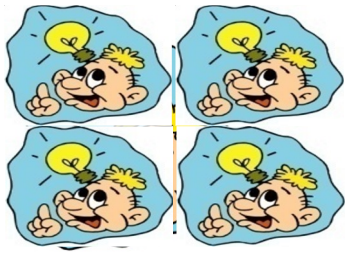
Access Methods

Buffer Management

Disk Space Management

Database Storage

Disk Storage, Files



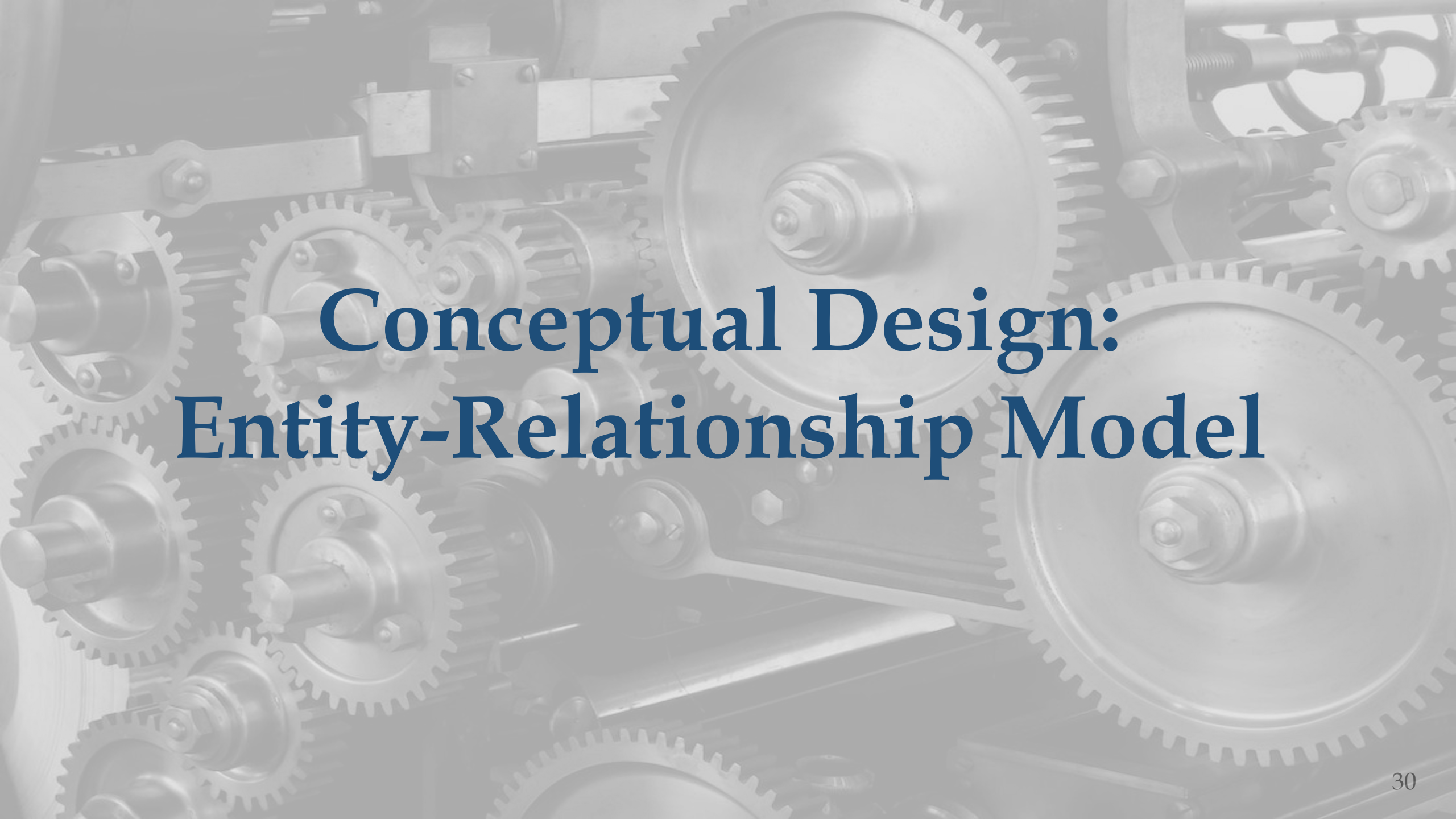
Want to access data

Query Processing

Query Optimization

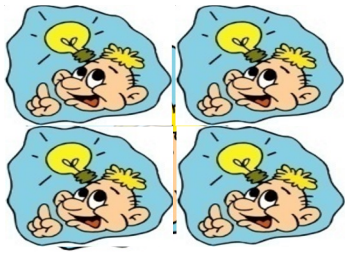
Transaction Management

Concurrency Control



Conceptual Design: Entity-Relationship Model

Simplified DBMS architecture: ER model



Want to access data

Want to store data

SQL

Result

Conceptual Design

ER Models

Logical Design

Physical Design

Query Optimization and Execution

Relational Operators

Access Methods

Buffer Management

Disk Space Management

Database Storage

The ER model

- **Basic ER modeling concepts**
  Readings: Chapters 2.1-2.3
- 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	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

- How do we design the schema?

Database design requirements

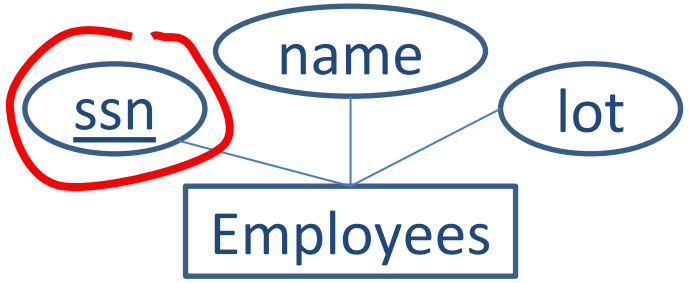
- **Requirements analysis**
  - User needs; what must database do?
- **Conceptual design**
  - High-level description (often done with ER model)
- **Logical design**
  - Translate ER into DBMS data model
- **Schema refinement**
  - Consistency, normalization
- **Physical design**
  - Indexes, disk layout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

What is a conceptual design

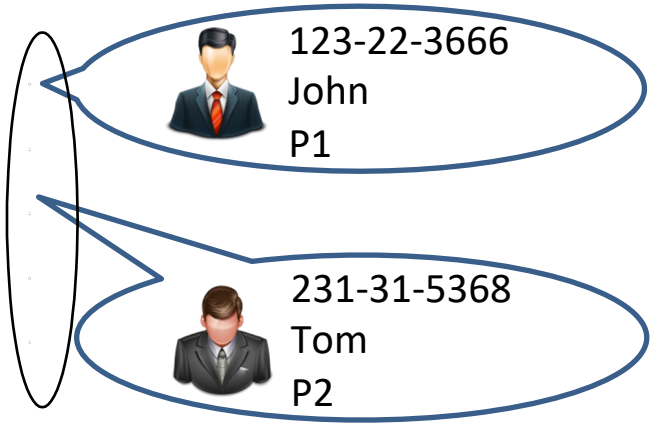
- What are the *entities* and *relationships* in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* that hold?
- A database "schema" in the ER Model can be represented pictorially: ***ER diagrams***
- Can map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

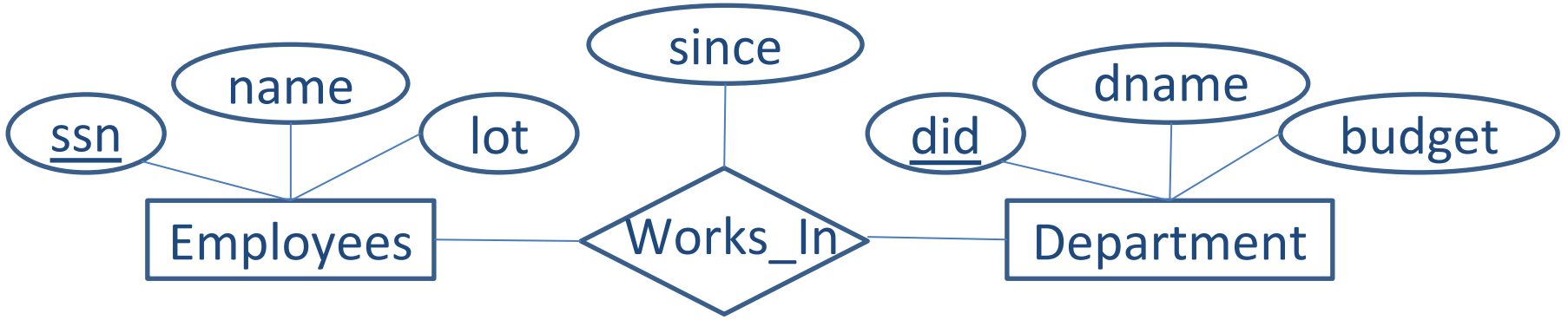


- **Instance of an entity set**



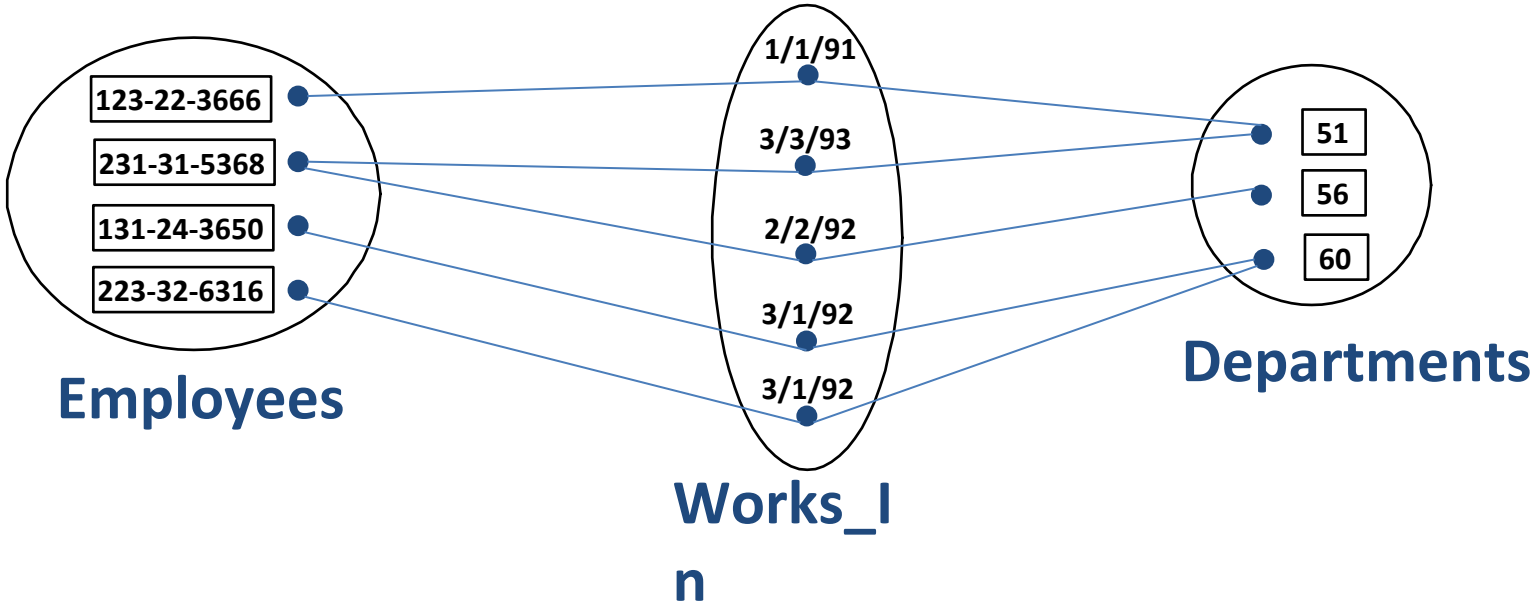
Basics of ER model (contd.)

- **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



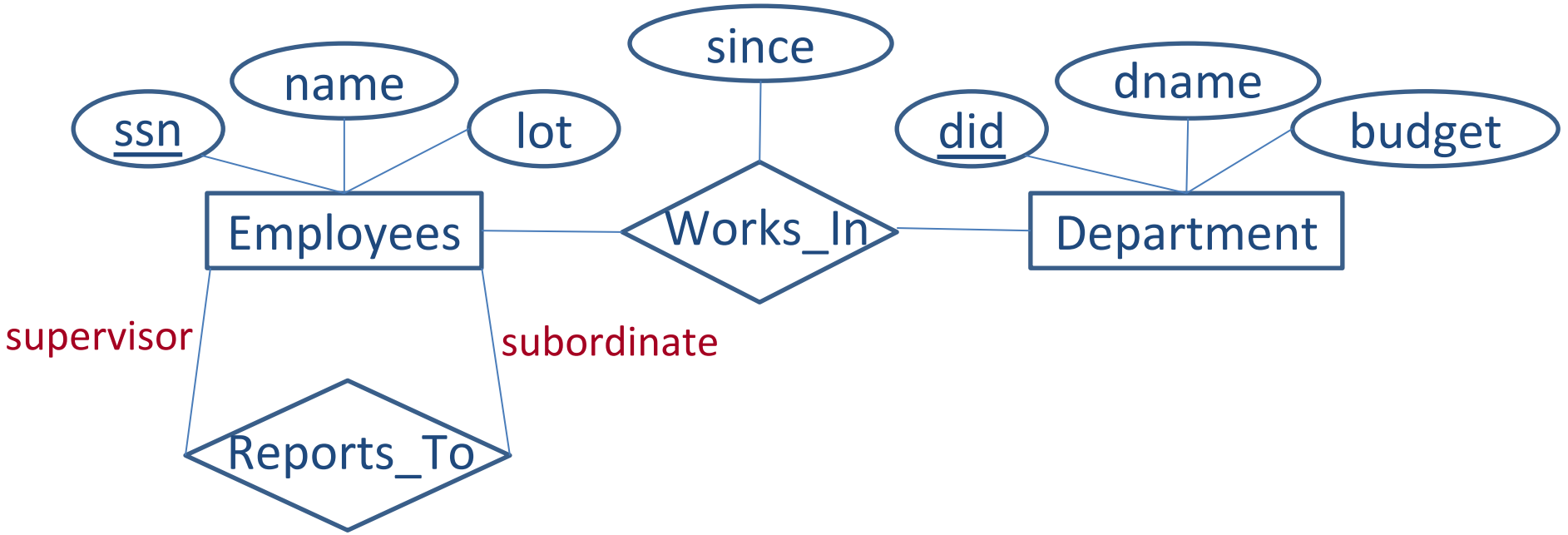
Basics of ER model (contd.)

- **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

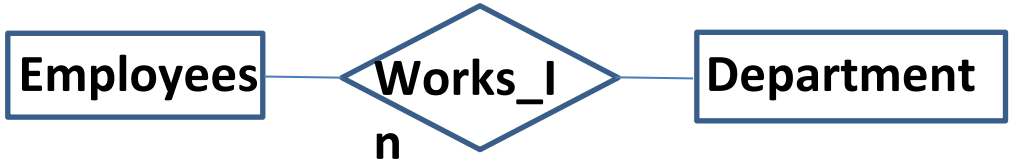


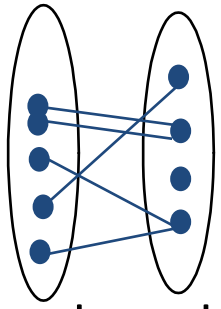
The ER model

- Basic ER modeling concepts
- **Constraints**

  Readings: Chapters 2.4-2.4.3, 2.5.3
- Complex relationships
- Conceptual Design

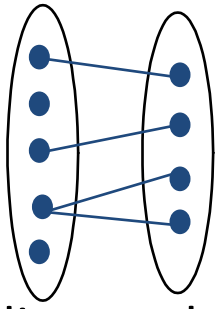
Key constraints





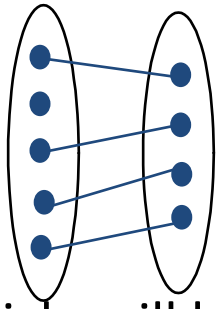
Many-to-Many

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



One-to-Many

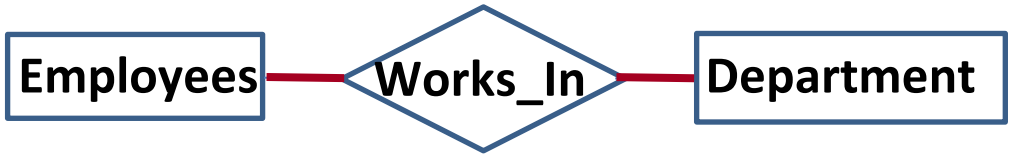
Each department has **at most one** manager, according to the **key constraint** on Manages



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 should work in at least one department
- Every Department should have at least one employee

- There could be some Employees who are not managers
- An Employee can manage at most one department.
- Every Department should have exactly one manager

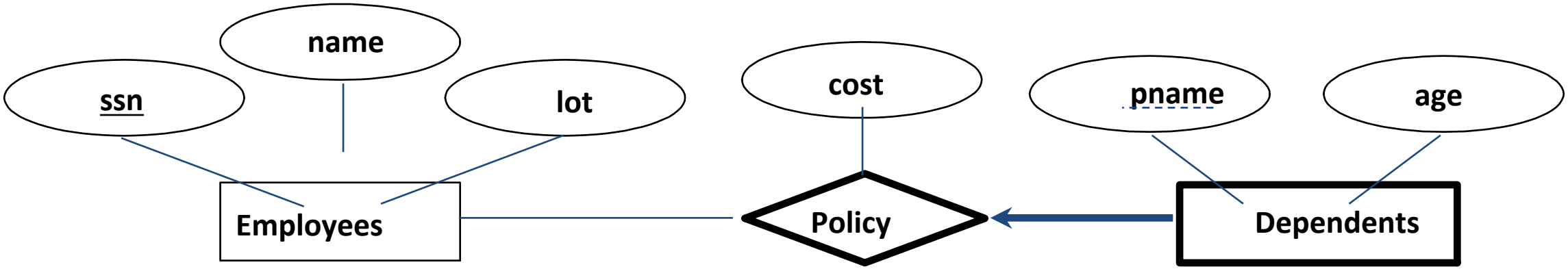
Partial Participation

- There could be some Customers who do not buy any Products
- There could be some Products which are not bought by any Customers

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



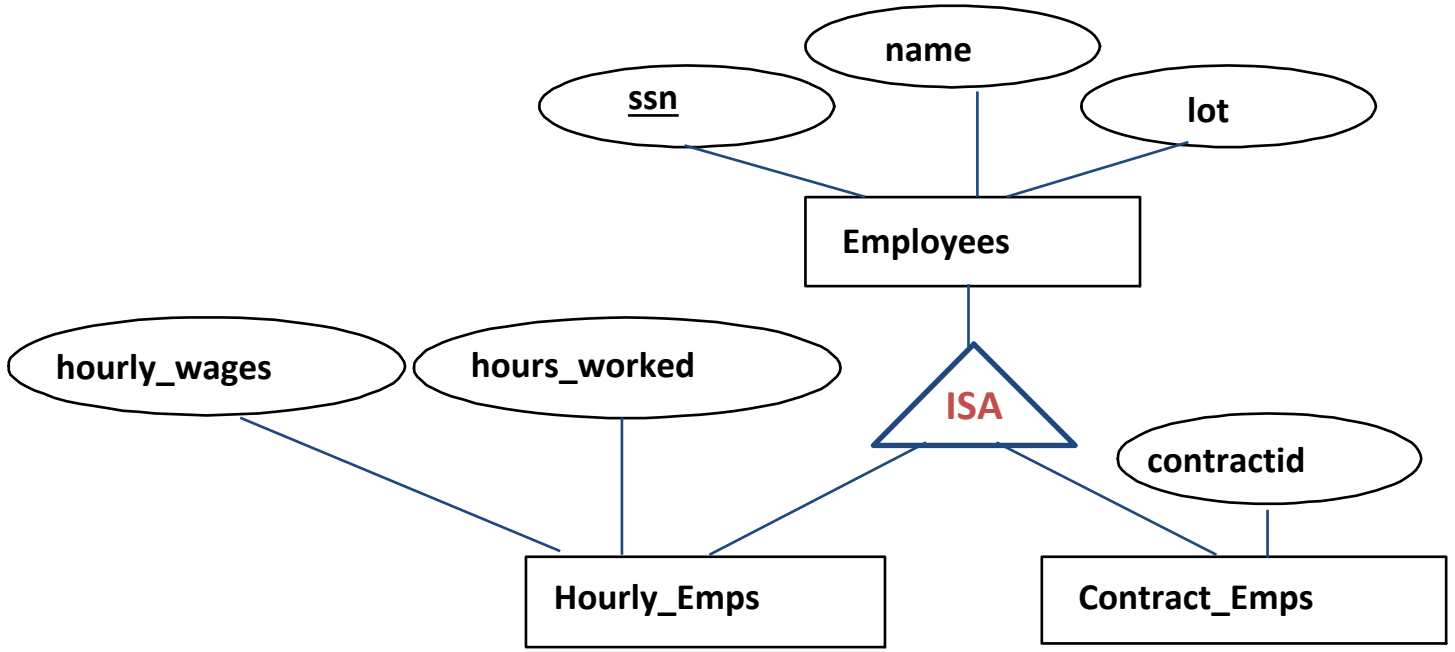
Weak entities have only a “partial key” (dashed underline)

The ER model

- Basic ER modeling concepts
- Constraints
- **Complex relationships**
  Readings: Chapters 2.4.4-2.4.5
- 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

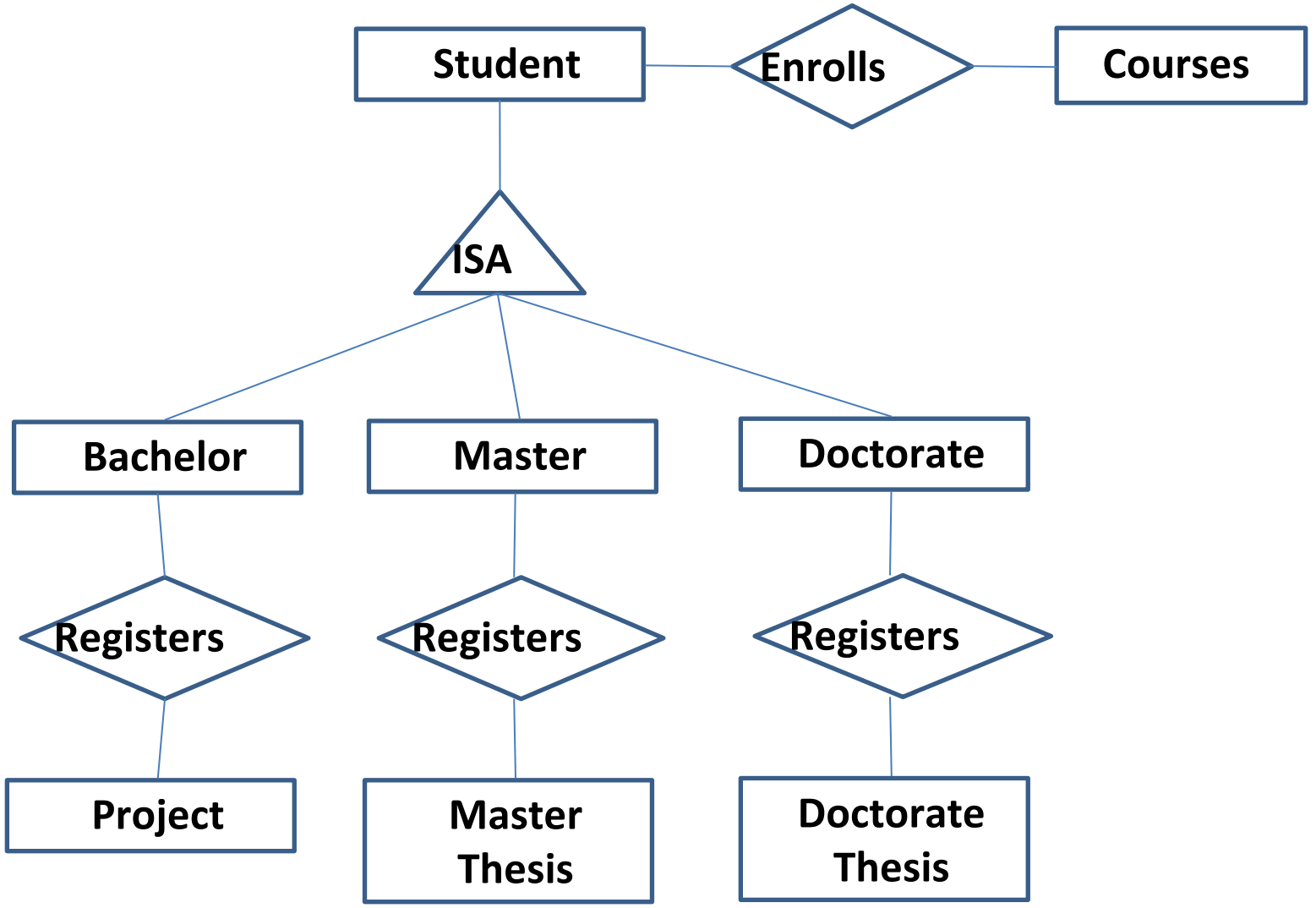


ISA ("is a") hierarchies

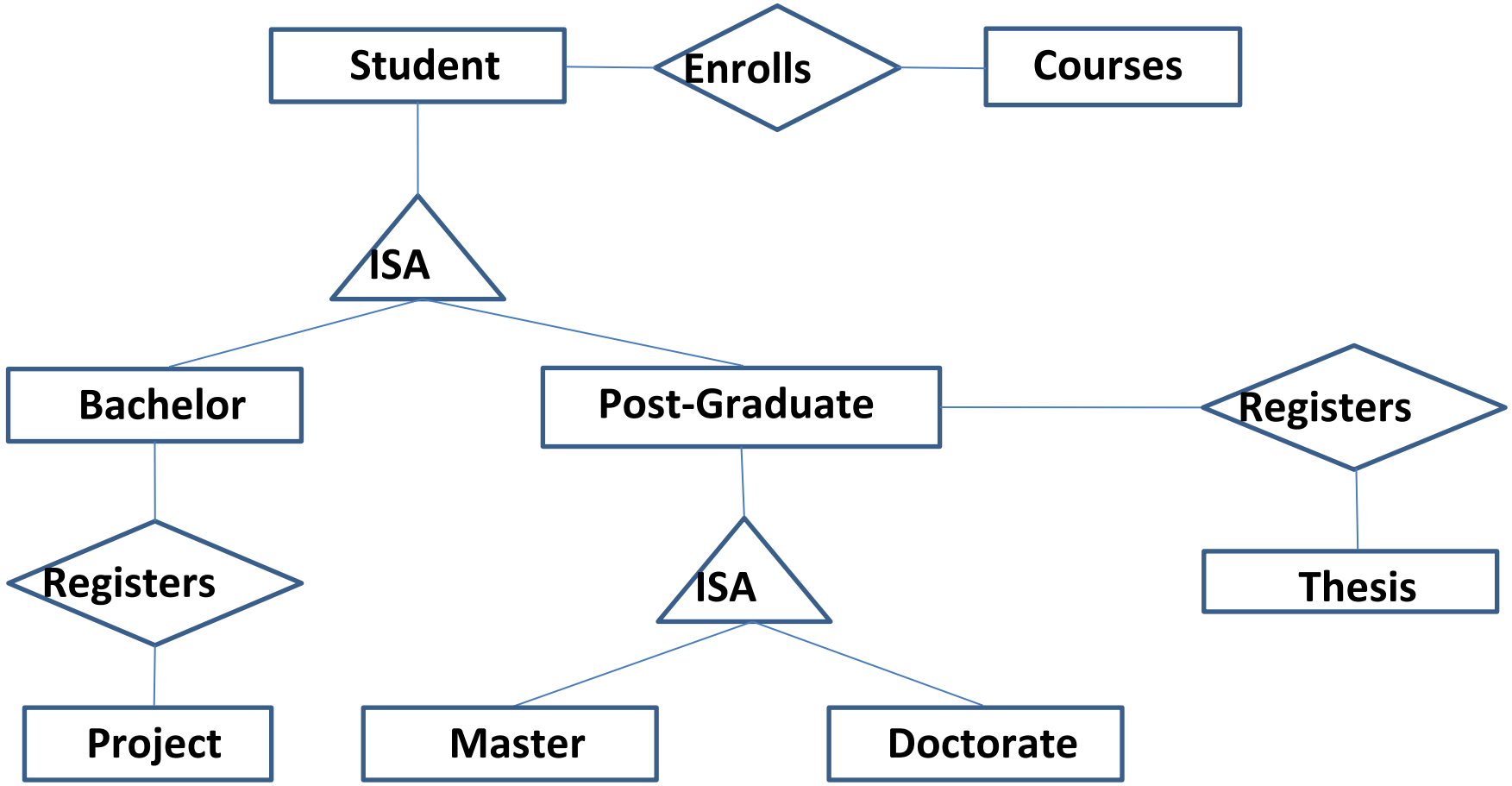
- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/Disallowed*)
- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/No)*

- 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)

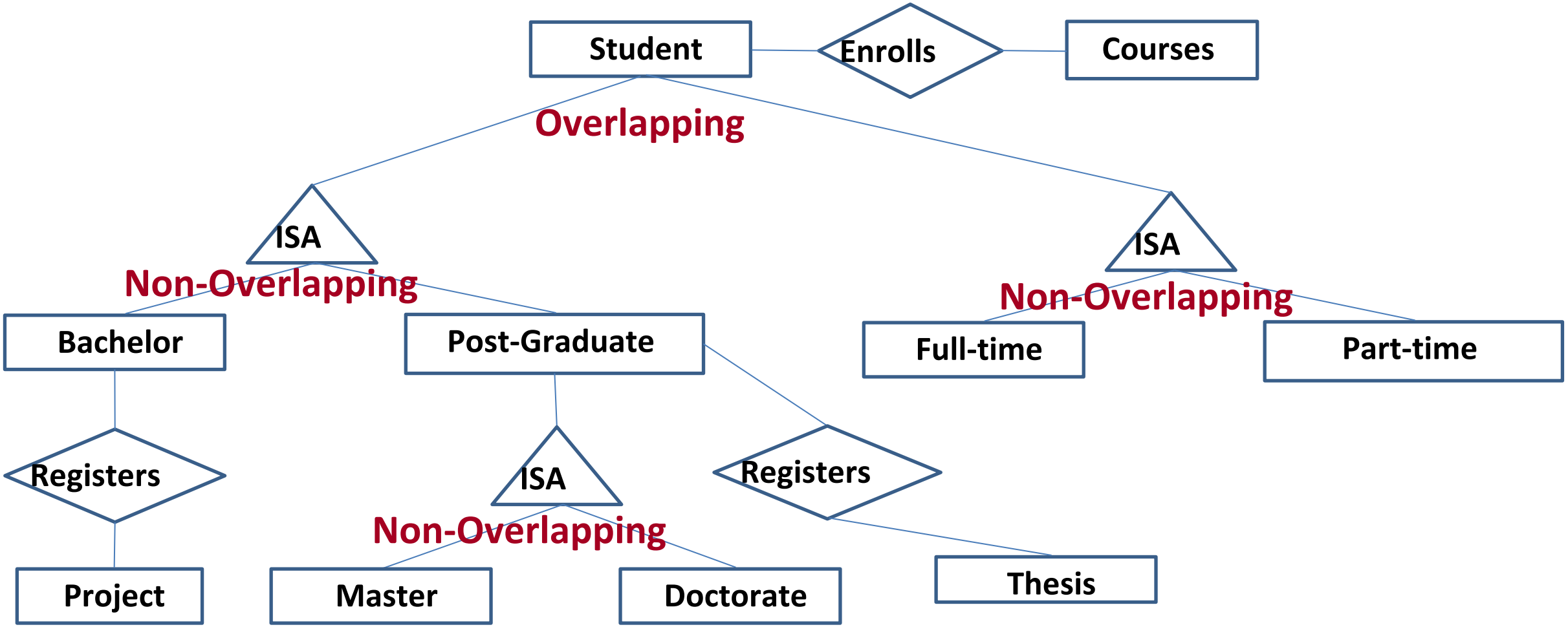
ISA (“is a”) hierarchies



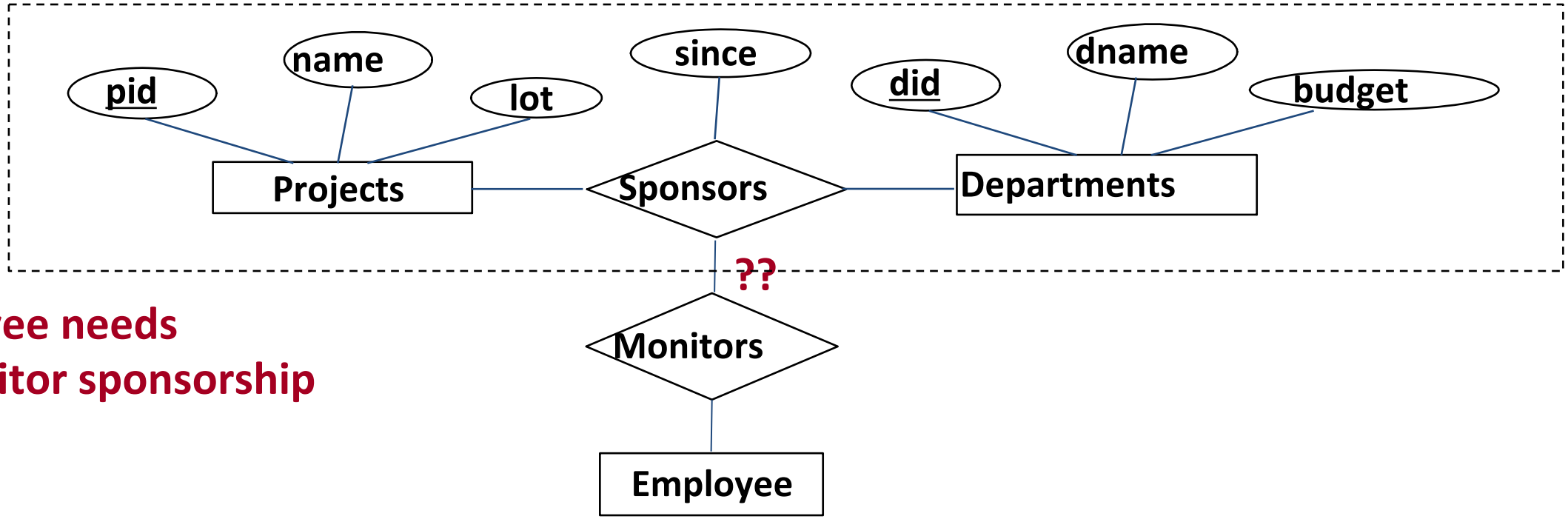
ISA (“is a”) hierarchies



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 2.5

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

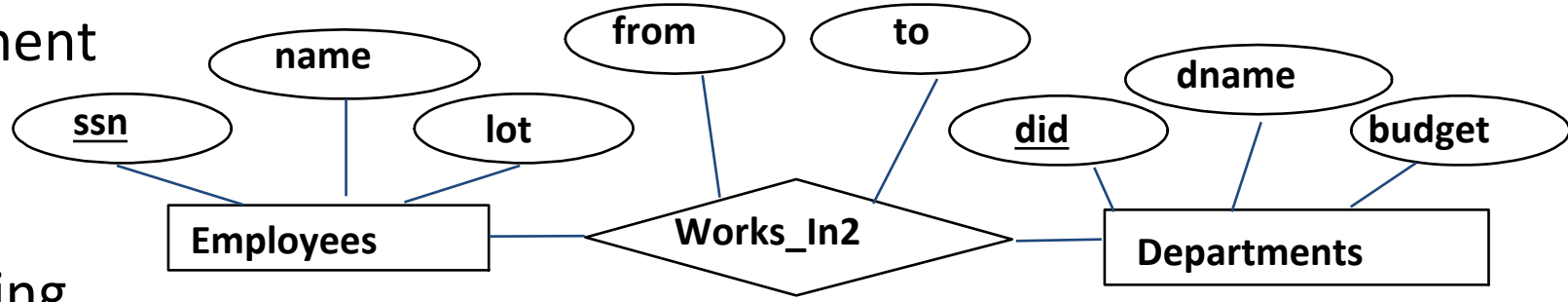
Entity vs attribute

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

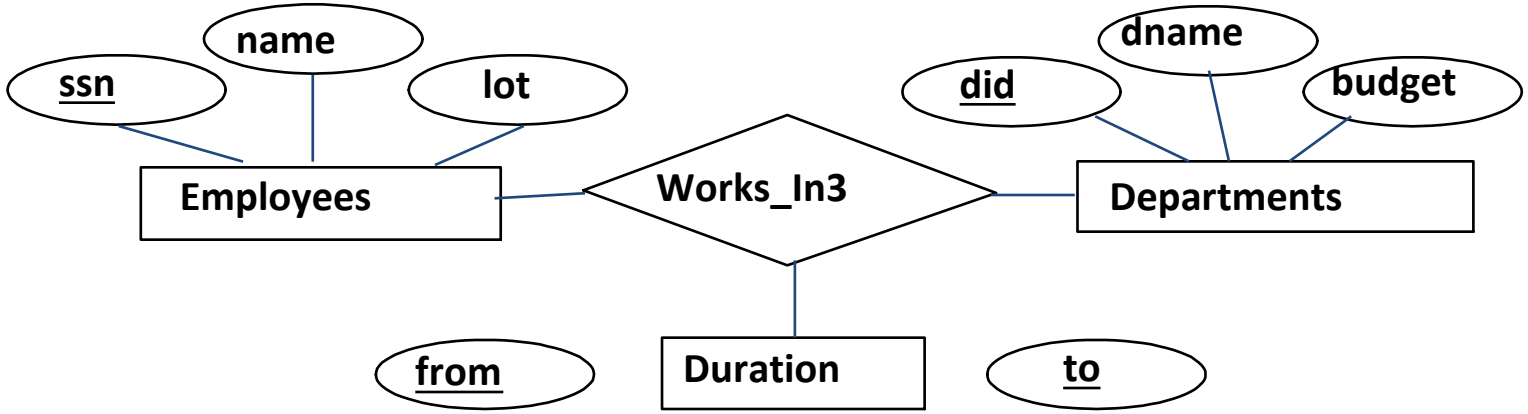
- Depends upon how we want to use address information, and the semantics of the data:
    - If we have several addresses per employee, the *address* must be an entity (since attributes cannot be set-valued)
    - If the structure (city, street, etc.) is important, the *address* must be modeled as an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

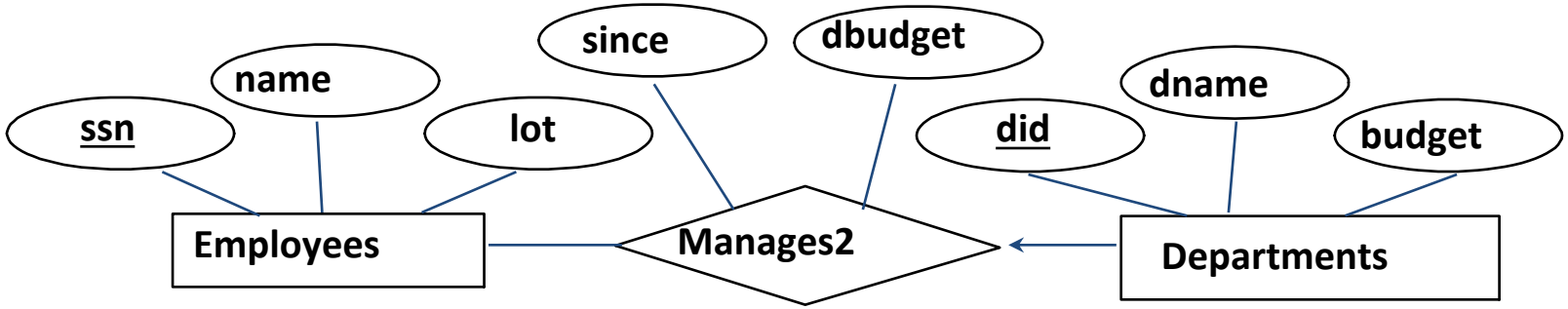


- Similar to the problem of wanting to record several addresses for an employee: we want to record *several values of the descriptive attributes for each instance of this relationship*

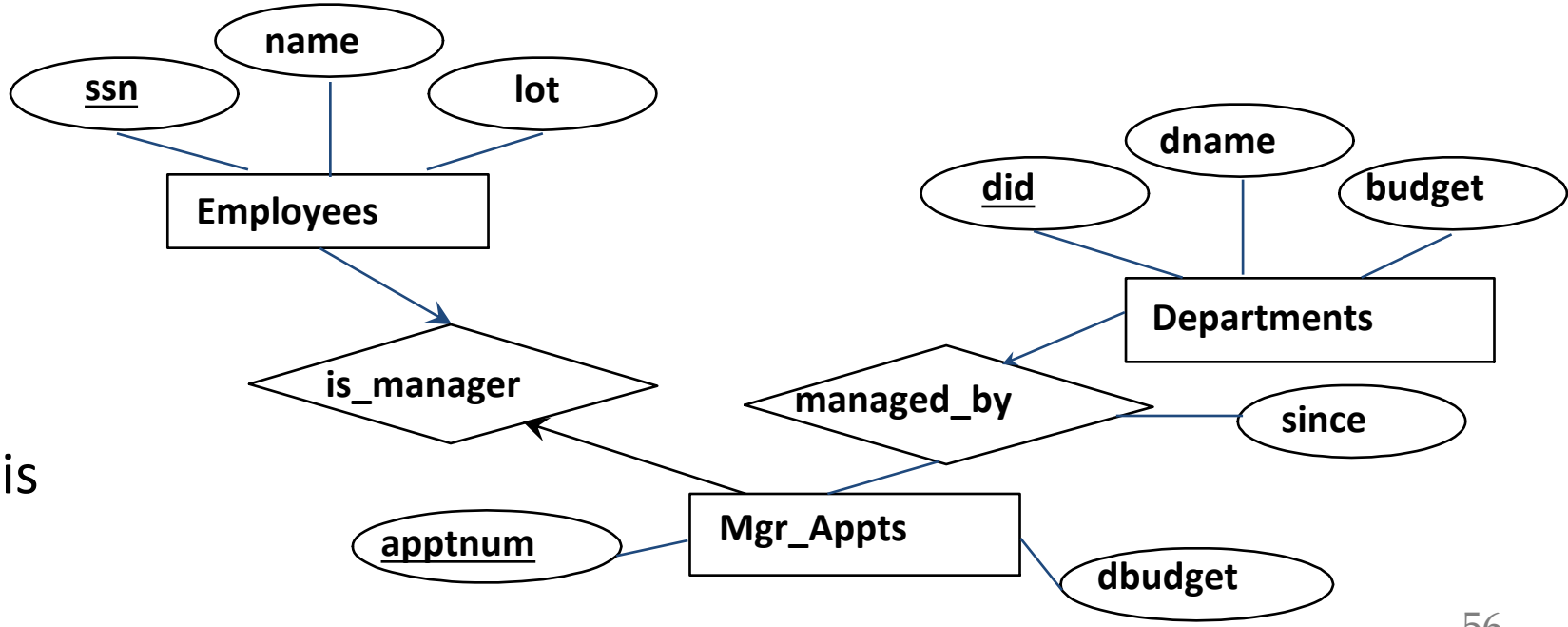


Entity vs. relationship

- 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