Northeastern University
London

Introduction to Database Design

This course

After this class, you should be able to…

- Use a relational database (via SQL and code)
- Design a secure, normalized, efficient database
- Understand (some of) how a DBMS works
- Understand database transactions, security and recovery.

Expectations from you

a) Work hard (really hard, it'll be worth it!)

b) Use resources (read book/materials before class, attend class, etc.)

c) Always complete the labs/assignments

Course Overview

Week No.	Database Design
1	Introduction to Database Design
2	Conceptual Database Design
3	Logical Design
4	Normalization
5	SQL& Relational algebra part 1
6	SQL& Relational algebra part 2
7	*MID TERM*
8	SQL and Web Programming
9	SQL Query Optimizations ***Set Exercises (70%)*** **AE1 Submission 14/11/2025**
10	Database Security and Administration
11	Database Transactions and Concurrency Part I
12	Autumn Break
13	Database Transactions and Concurrency Part I ***Written Assignment(Group & Individual)*** **AE2 Submission 8/12/2025**

labs

- Labs/Assignments & Feedbacks
- Will help with AE1 & AE 2
- Will be used for check-in on the group assignments

Outline

1. What is a Database? A DBMS?
2. Why use a DBMS?
3. Databases in Context
4. Design and Implementation Process
5. Relational Data Model *(Intro)*
6. *DBMS in the Real World - Industry Research & Presentation* ***(lab)***
7. *Software installation* ***(lab)***

What is Database?

A collection of related data, most often…

- reflects some aspect of the real world
- logically coherent with inherent meaning
- designed, built, and populated with data for a specific purpose
  - intended group of users
  - some preconceived applications with which these users are interested
  - application requirements in terms of performance, security, redundancy, concurrency, etc.

Database Management System (DBMS)

A collection of programs that enables users to create and maintain a database

- Supports specifying the data types, structures, and constraints of the data
- Stores the data on some medium under control of the DBMS
- Supports querying and updating the database
- Protects data against malfunction and unauthorized access

Why use a DBMS?

Common tradeoff in CS:

A. Code from scratch
   - Pros: you know your problem best (so fast, customized)
   - Cons: slow, labor intensive, need to add/change features?

B. Find a library/tool that solves [part of] your problem
   - Pros: fast via bootstrapping, better designed?
   - Cons: understand the tool, may not be efficient, support?

DBMSs adopt some set of limiting assumptions in order to <u>efficiently</u> support a <u>useful</u> feature set over a wide class of possible databases

Example: Student Records

- Given a school with MANY students (NEU: ~25k, UM: ~45k), each with some data (name, ID, DOB, classes)

- Write a program that can <u>efficiently</u>…
    - Retrieve a random student
    - Retrieve the first/last student, according to…
        - Last name
        - DOB
    - Retrieve a student by…
        - ID
        - Name (with *'s)
    - Retrieve a class roster (all students in class X)
    - Handles adding/removing/editing students/classes
    - Handles multiple simultaneous reads/writes
    - Provides differing access rights
    - Handles OS faults/power outages

…

Many Kinds of DBMSs (1)

- Graph databases
  - Create nodes, edges, labels
  - Query about relationships and paths
    - Find your friends
    - Find someone that can help you learn databases

- Spatial databases
  - Data pertaining to space occupied by objects
  - Objects in 2D/3D
  - Query locations, relations
    - Collision detection

Many Kinds of DBMSs (2)

- Key-Value stores
  - Associative array
  - Scalable, fault-tolerant
  - Query

FOUNDATIONDB

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

- Document stores
  - Create dynamic documents
  - Query about contents
    - Find by author, title, content, etc. patterns

mongoDB

{name:"mongo", type:"DB"}

Relational DBMS

We focus on **relational** databases

Based on the relational *data model*

- Researched ~45 years, widely used
  - Free/paid implementations for personal use, embedded systems, small/large enterprise

Relational Databases (1)

Table Name

Table or "Relation"

Attributes/Columns

STUDENT

Name	SSN	Phone	Address	Age	GPA
Ben Bayer	305-61-2435	555-1234	1 Foo Lane	19	3.21
Chung-cha Kim	422-11-2320	555-9876	2 Bar Court	25	3.53
Barbara Benson	533-69-1238	555-6758	3 Baz Blvd	19	3.25

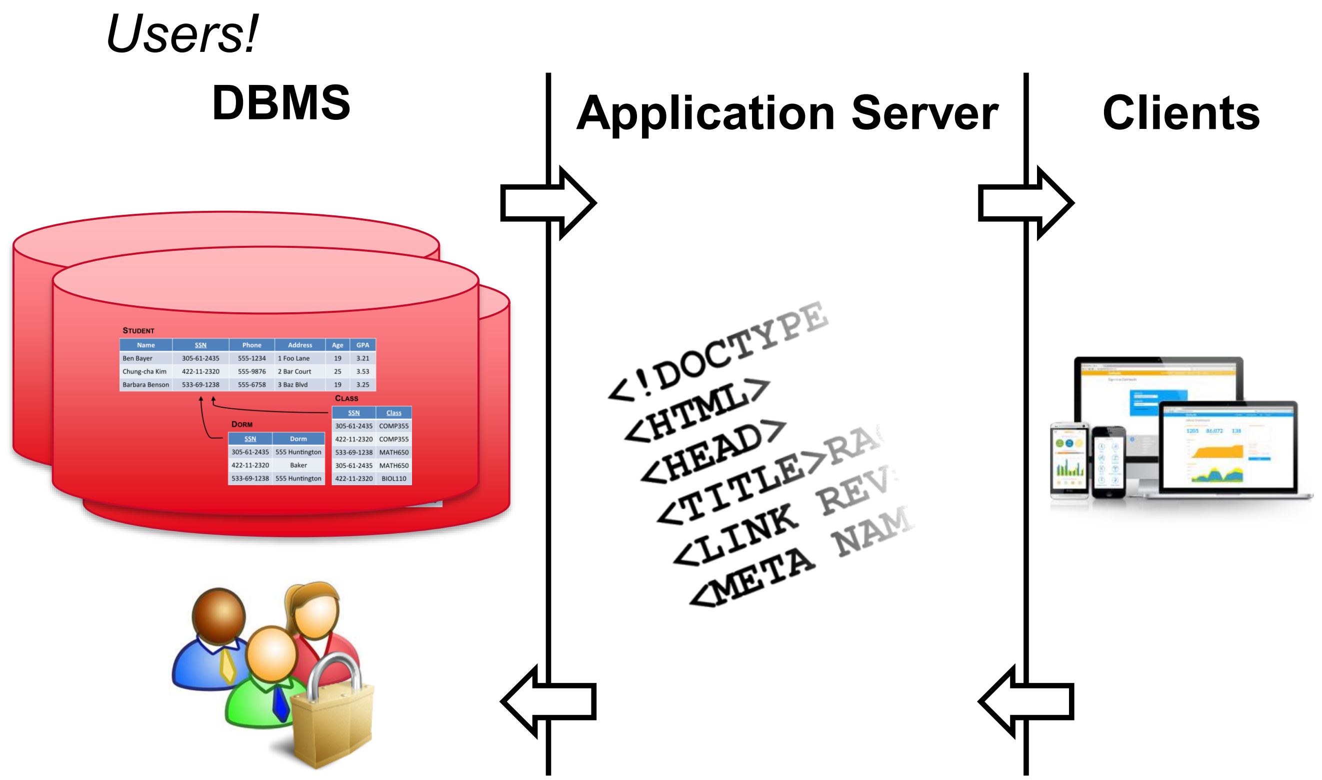
Constraints

Has to be unique

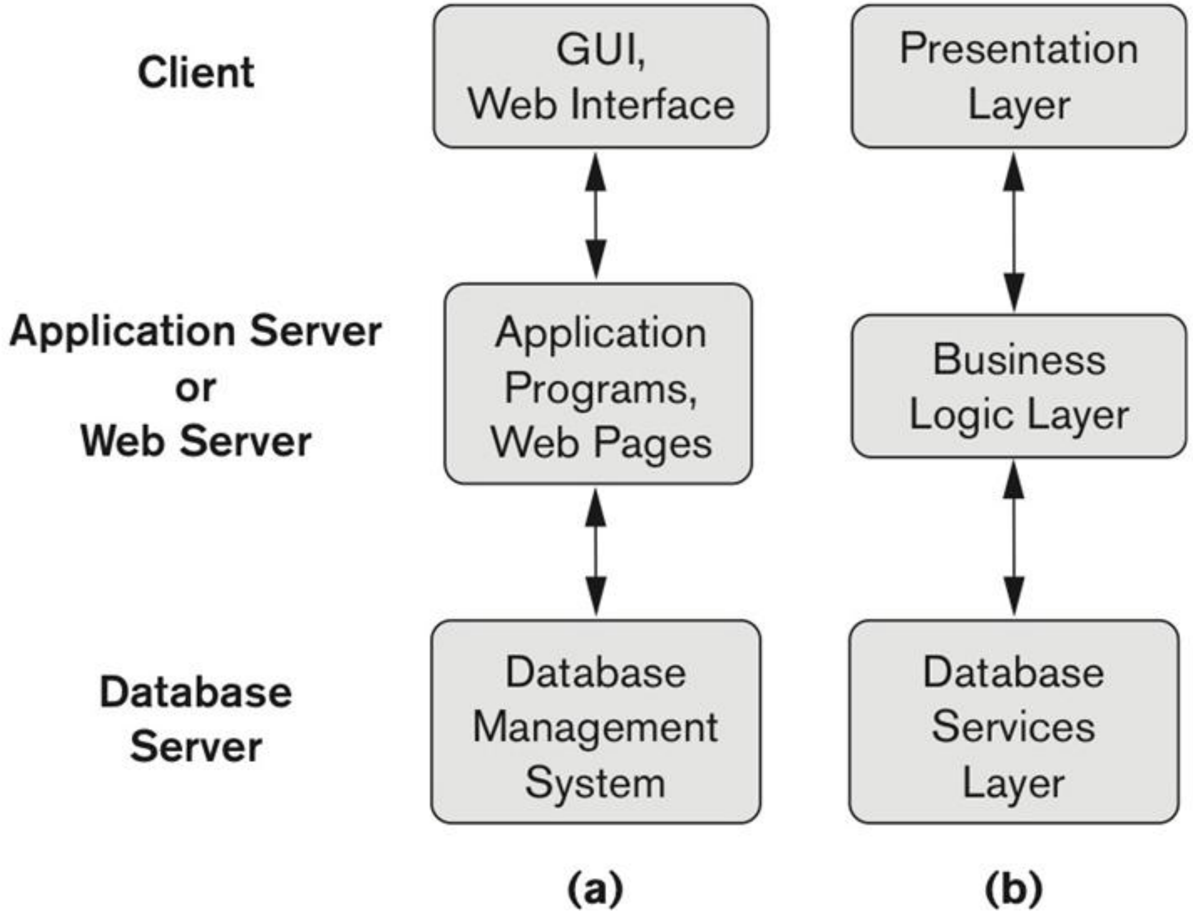
Must be an integer > 0

Must be a number [0,4]

Relational Database (2)

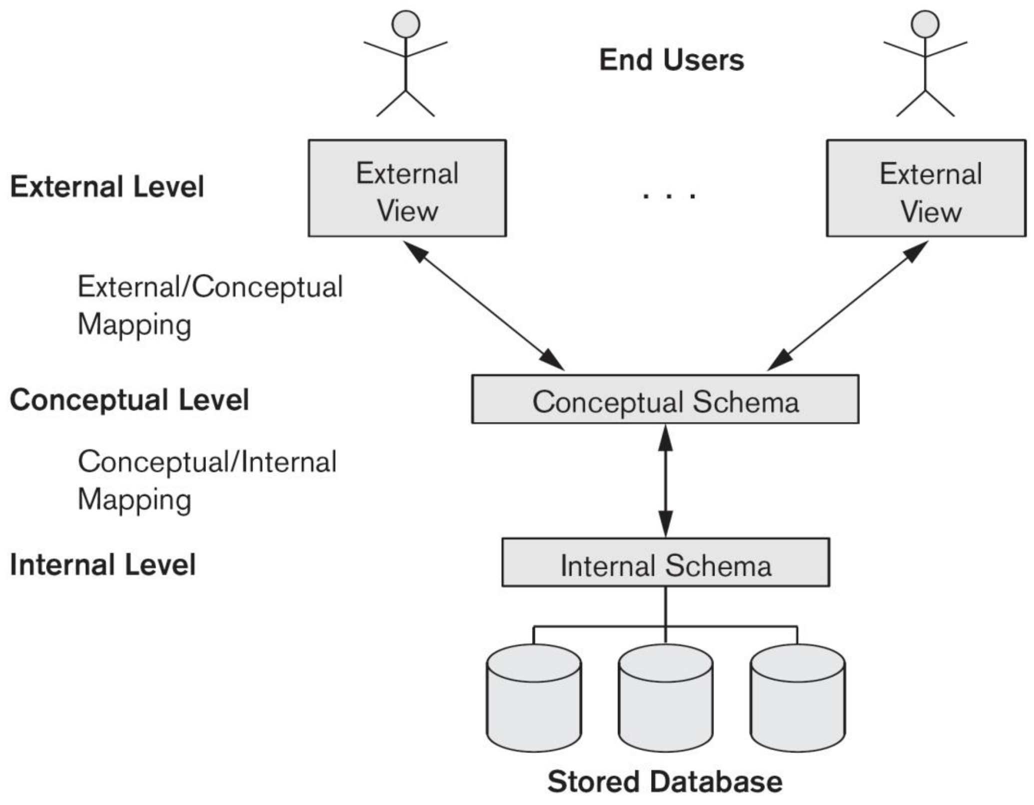


Databases in Context



Relational Database features (1)

- Data *independence* via **data models**
  - Conceptual representation independent of underlying storage or operation implementation



ER Diagrams

Relations

Relational DBMS Features (2)

- Operation abstraction via…
    - Declarative languages
        - Structured Query Language (SQL)
            - Data… definition, manipulation, query
    - Programmatic APIs
        - Function libraries (focus), embedded languages, stored procedures, etc.

Relational DBMS Features (3)

- Reliable concurrent transactions
  - (**A**)tomicity: “all or nothing”
  - (**C**)onsistency: valid -> valid’
  - (**I**)solation: parallel execution, serial result
  - (**D**)urability: once it is written, it is so

- High performance
  - Buffering, caching, locking (like a mini OS)
  - Query optimization, redundant data structures (e.g. indexes, materialized views)

Relational DBMS Features (4)

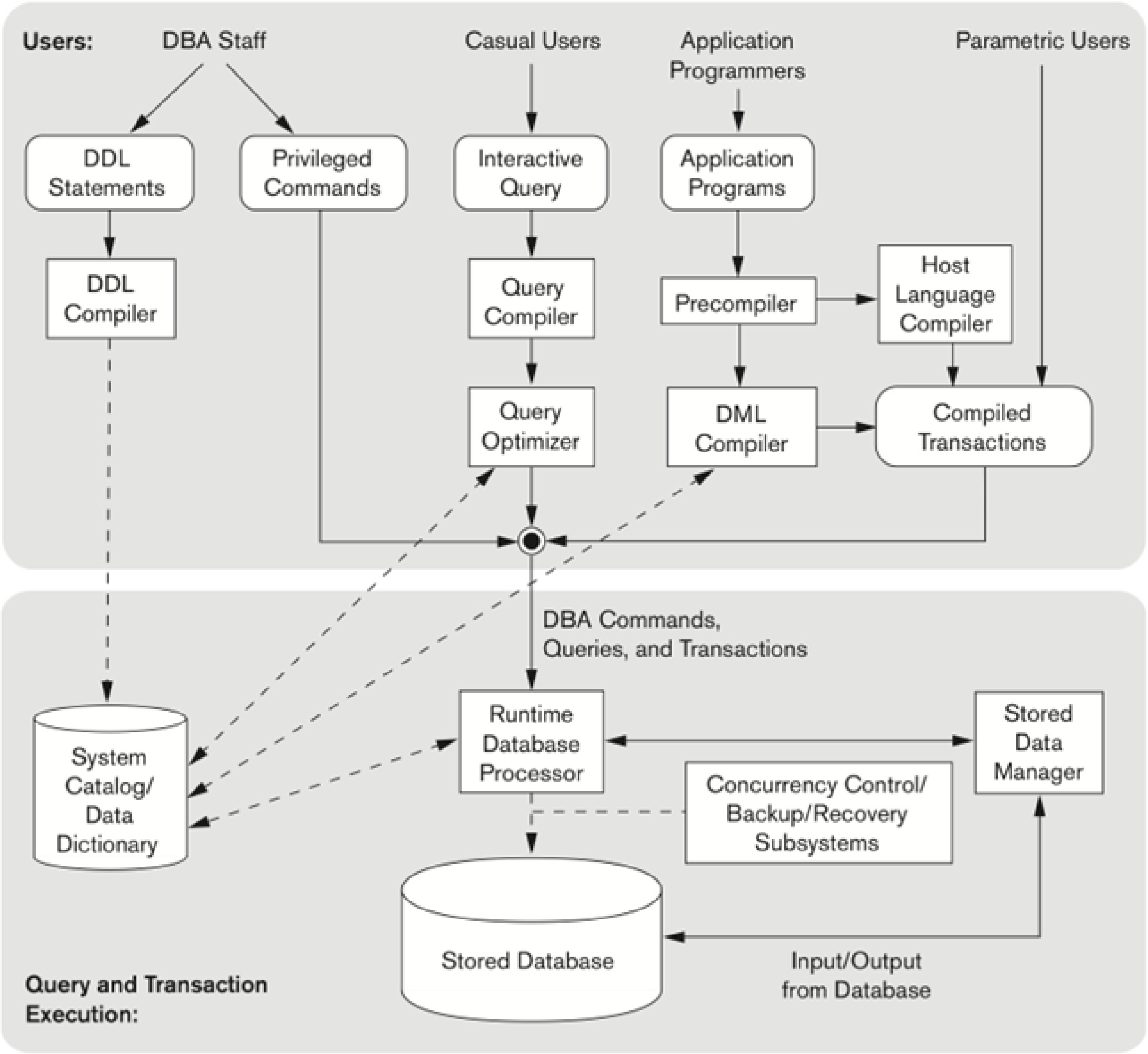
- Authentication and authorization
  - Discussed in context of other security concerns/techniques
- Backup and recovery
  - Logging, replication, migration

Databases in Context

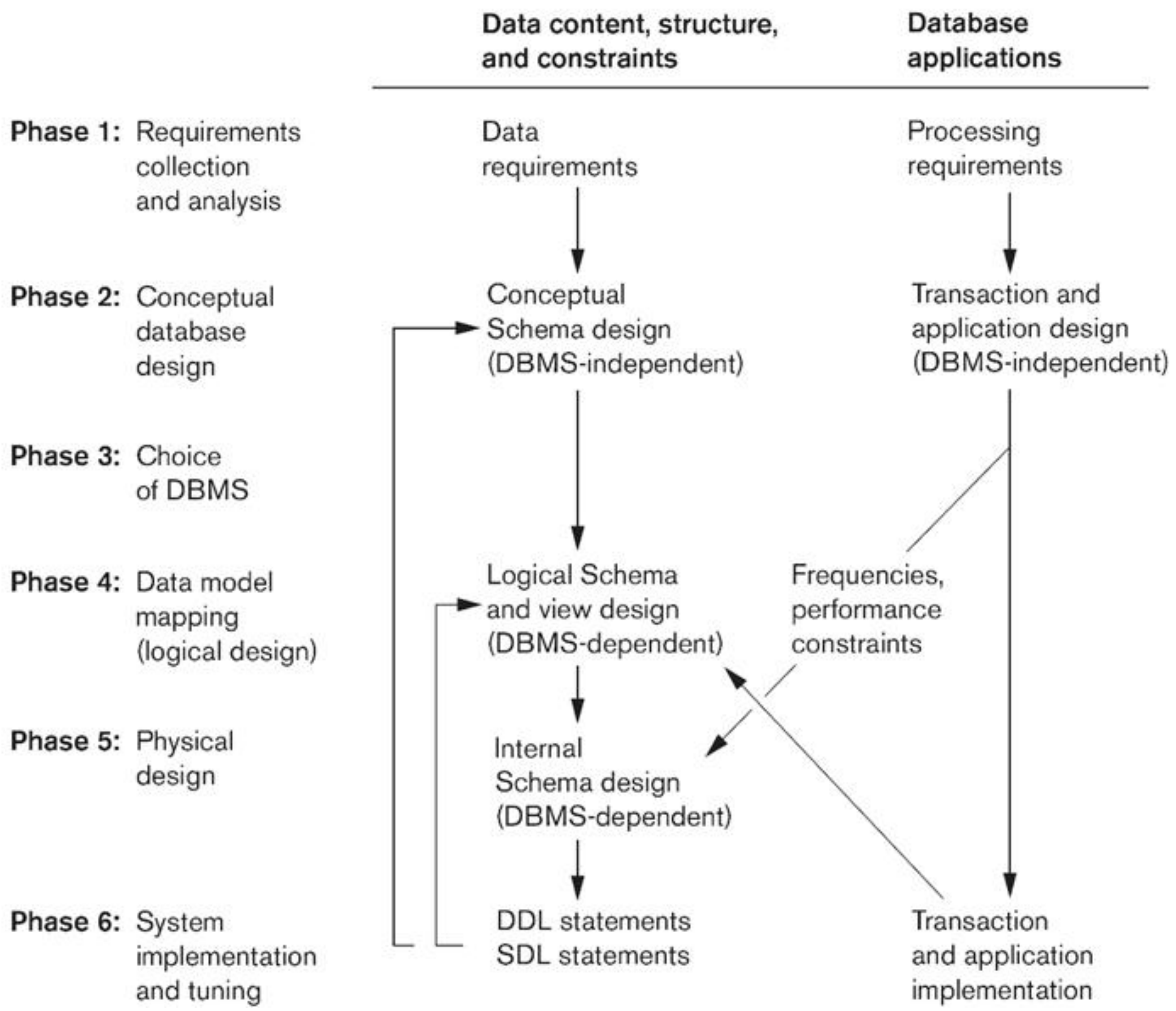
People

1. **Database designers**
2. **System analysts & application programmers**
3. Database administrators
4. End users
5. Back-end
   a. DBMS designer/implementer
   b. Tool developers
   c. SysAdmins

Relational DBMS



Database Design and Implementation Process



Requirements Collection & Analysis

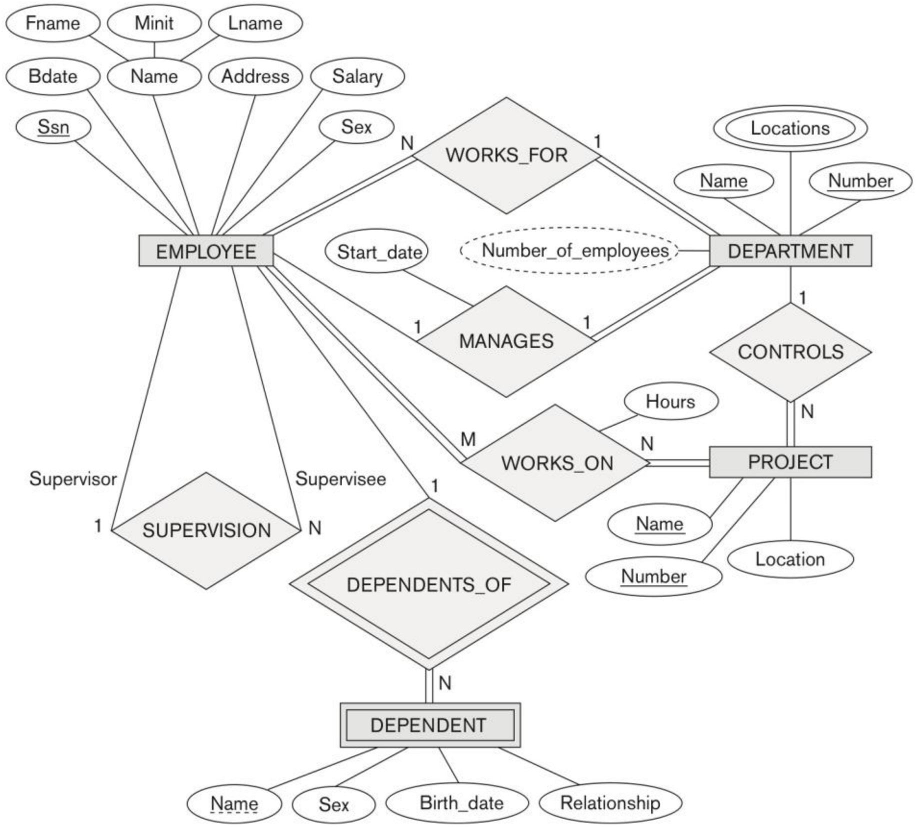
- Data/Constraints

  "The company is organized into departments. Each department has a unique name, number, and a particular employee who manages the department. We keep track…"

- Functional Needs
  - Operations/queries/reports
    - Frequency
  - Performance, security, etc.

Conceptual Design

Data



Application

- Software
  - UML
  - Form design
- Database
  - Transaction design
  - Report design

Logical Design

Data

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Application

- Supporting code (that does not depend upon database)

Normalization

Data

- Schema Refinement

Application

- Supporting code (that does not depend upon database)

Physical Design

Data

- Index, materialized view selection and analysis

Application

- Implementing operations as queries
- Implementing constraints as keys, triggers, views
- Implementing multi-user security as grants

Implementing and Tuning

Data

- DDL statements
- De-normalization, updating indexes/materialized views

Application

- Query integration
- Profiling queries/operations
- Security, concurrency, performance, etc. analysis

Relational Data Model

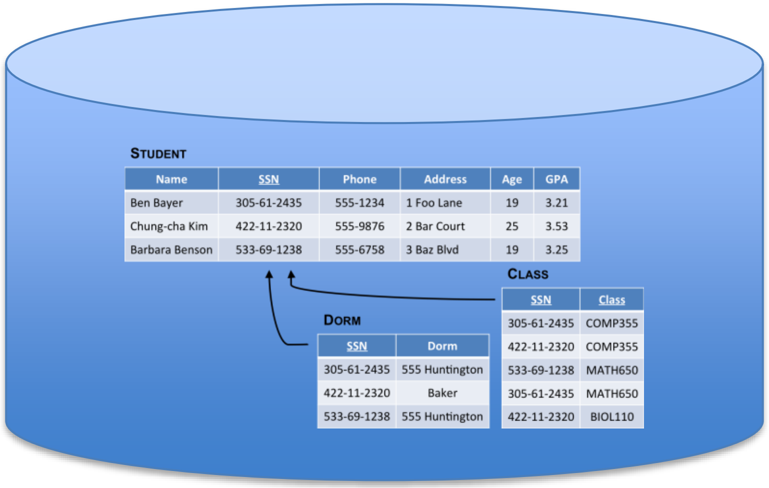
A database consists of…

i. a set of ***relations*** (tables)

ii. a set of ***integrity constraints***

Pop Quiz:
What is a **set**?

A database is in a **valid state** if it satisfies all integrity constraints (else **invalid state**)



A Relation

A relation consists of…

i. its ***schema***, describing structure
ii. its ***state***, or current populated data

Schema

STUDENT

Name	<u>SSN</u>	Phone	Address	Age	GPA
Ben Bayer	305-61-2435	555-1234	1 Foo Lane	19	3.21
Chung-cha Kim	422-11-2320	555-9876	2 Bar Court	25	3.53
Barbara Benson	533-69-1238	555-6758	3 Baz Blvd	19	3.25

State

Relational Schema

- Relation name
  STUDENT
- Ordered list of *n* ***attributes*** (columns; degree *n* or *n*-ary)
  Each with a corresponding ***domain*** (set of valid ***atomic*** values)
  dom(SSN) = "###-##-####"
  dom(GPA) = [0, 4]

  Notation: NAME(A_1, A_2, … A_n)
  STUDENT(Name, SSN, Phone, Address, Age, GPA)

What is the degree of STUDENT?

STUDENT

Name	<u>SSN</u>	Phone	Address	Age	GPA

Relation State

A set of *n*-**tuples** (rows)
Each has a value in the domain of every corresponding attribute (or **NULL**)
Notation: *r*(NAME)

Mathematically, a subset of the Cartesian product of the attribute domains; related to the closed-world assumption

$$r(STUDENT) \subseteq (dom(Name) \times dom(SSN) \times \ldots dom(GPA))$$

Ben Bayer	305-61-2435	555-1234	1 Foo Lane	19	3.21
Chung-cha Kim	422-11-2320	555-9876	2 Bar Court	25	3.53
Barbara Benson	533-69-1238	555-6758	3 Baz Blvd	19	3.25

Exercise

Diagrammatically produce a relation HAT according to the following schema; the relation state should have at least three tuples

HAT(Team, Size, Color)
dom(Team) = { `RedSox, Bruins, Celtics, Patriots, Revolution` }
dom(Size) = { `S, M, L, XL` }
dom(Color) = { `Black, Blue, White, Red, Green, Yellow` }

How many tuples are possible in this relation?

NULL

- **NULL** is a special value that may be in the attribute domain

- Used to represent the values of attributes that may be unknown or may not apply to a tuple.
- Several possible meanings
  e.g. unknown, not available, does not apply, undefined, …

- Best to avoid
- Else deal with caution

Value Structure in Tuples

- Each value should be **atomic** – no *composite* or *multi-valued* attributes
  - Composite: “one column, many parts”
  - Multi-valued: “one column, multiple values”

- Convention called 1NF (*first normal form*)
  - More on this later in the course

Violation of 1NF: Composite

DORM

SSN	Dorm Info	
	Dorm	Room
305-61-2435	555 Huntington	1
422-11-2320	Baker	2
533-69-1238	555 Huntington	3

vs.

DORM

SSN	Dorm	Room
305-61-2435	555 Huntington	1
422-11-2320	Baker	2
533-69-1238	555 Huntington	3

Violation of 1NF: Multi-Valued

CLASS

SSN	Class
305-61-2435	COMP355, MATH 650
422-11-2320	COMP355, BIOL110
533-69-1238	MATH650

vs.

CLASS

SSN	Class
305-61-2435	COMP355
422-11-2320	COMP355
533-69-1238	MATH650
305-61-2435	MATH650
422-11-2320	BIOL110

Summary

- A **database** is a collection of related data that reflects some aspect of the real world; is logically coherent with inherent meaning; and is designed, built, and populated with data for a specific purpose

- A **database management system** (DBMS) is a collection of programs that enables users to create and maintain a database

- There are many types – we will focus on **relational** databases (RDBMS) and a bit of NoSQL Databases

- The typical database design process is an iterative process of requirements collection/analysis, conceptual design, logical design, physical design, and system implementation/tuning

Friday Lab

- The lab is available on canvas
  - Please try to go through before Friday.

Acknowledgement

- The major part of the teaching agenda for this course is based on material developed by Nate Derbinsky (https://derbinsky.info/)