CS 61:
Database Systems

Introduction

Adapted from Silberschatz, Korth, and Sundarshan unless otherwise noted
Augmented by Claude Opus

You use databases every day, but may not think them about very much

The New York Times

facebook

Databases are a set of programs used to Create, Read, Update, or Delete (CRUD) data through operations called queries

Queries typically use SQL to carry out queries

What characteristics would you like in a database?

Pierson's conjecture

Every non-trivial application has a database component

Introduction

Your background

- Undergrads/grads
- Macs/Windows/Linux

My background

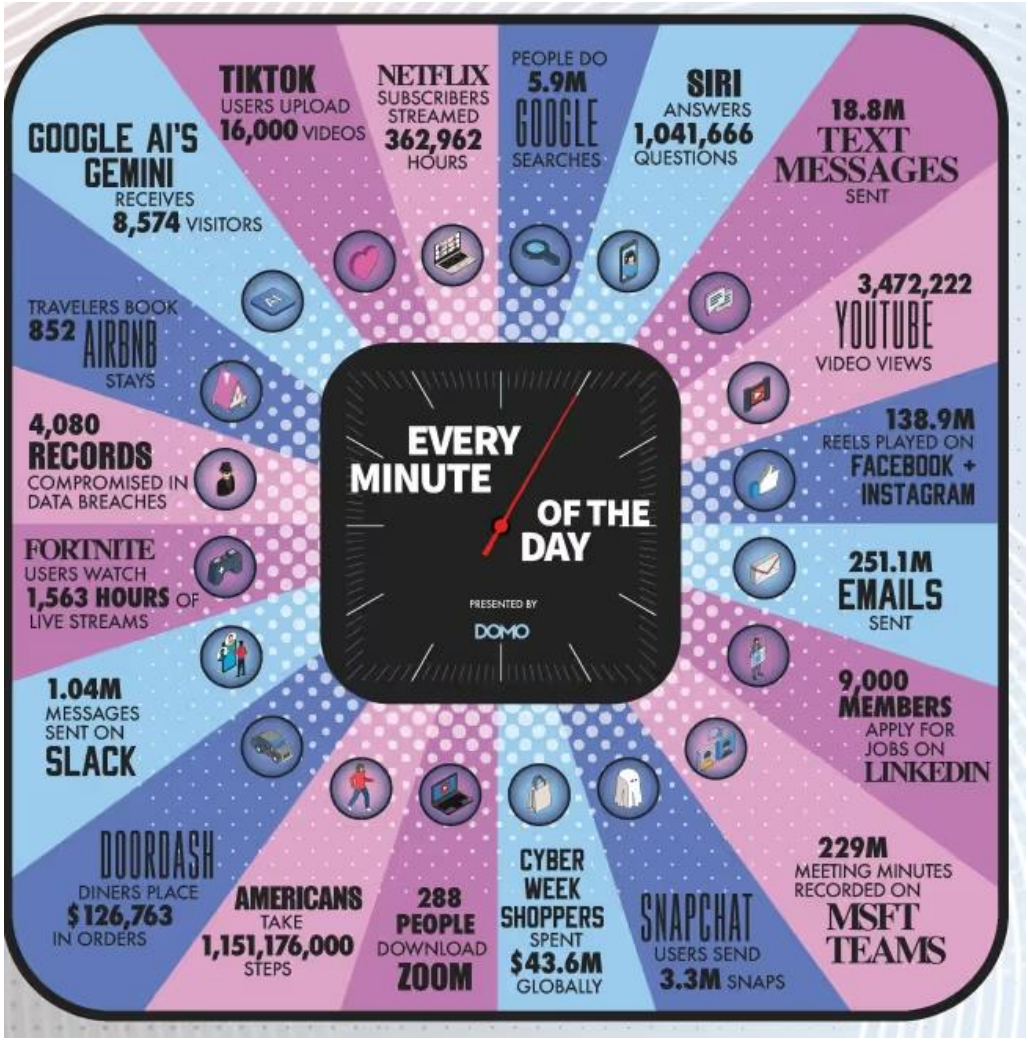
Why are you interested in databases?

- Estimated 19.8 billion devices on the Internet in 2025 (29 billion projected by 2030)[1]
- Devices exchange 402 *quintillion* bytes *every day*[2]
- 90% of world's data created last two years[3]
- Might be a good idea to know something about how this data is processed and stored

[1] https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/
[2] https://explodingtopics.com/blog/data-generated-per-day
[3] https://www.proofpoint.com/sites/default/files/infographics/pfpt-us-ig-a-brief-history-of-data.pdf

Big data is often characterized by the 5 V's

Every minute…



https://www.domo.com/learn/infographic/data-never-sleeps-12

Big data characterized by five V's:

1. **Volume**: quantity of data to be stored, systems can be scaled
   - Vertically : "get a bigger box"
   - Horizontally: "get more boxes"
2. **Velocity**: speed at which data must be processed
   - Stream processing: analyze data as it comes
   - Feedback loop: data generates recommendations, recommendations lead to more data
3. **Variety**: store data in many forms
   - Structured data: fits into predefined data model
   - Unstructured data: does not fit data model
4. **Veracity**: can the data be trusted?
5. **Value**: can we exact value from the data, perhaps by correlating with other data?

Data privacy concerns abound!

We must be thoughtful about how we use big data

- Each time you shop online, you share information with retailers
- Retailers study patterns closely to determine what you like
- Purchases tied to your credit card/browsing habits
- Also buy information from other sources (demographic, other retailers)
- Target knew a young woman was pregnant before her father knew:
  - Women on Target's baby registry buy lots of lotion around second trimester
  - In first 20 weeks also buy lots of vitamins, unscented soap, and cotton balls (could also have skin infection!)
  - Used 25 products to predict pregnancy, then sent coupons to likely women
  - Buy cocoa-butter lotion, large purse (could double as diaper bag), zinc and magnesium supplements => 87% chance due within four months
- Dad confronted Target suggesting they are encouraging her to get pregnant – found out truth later
- Target now spreads out pregnancy coupons with other ones to not appear creepy
- Did it work? Sales went from $44B to $67B after profiling
- Did Target break any laws?
- What about other companies like Facebook, Twitter, Instagram?

https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did

Agenda

1. Course logistics
2. Data, information, and knowledge
3. Problems with early data management
4. Modern relational database management systems
5. Big picture of relational database design

This class is about database systems

- Four main goals for the course:
  1. Query existing relational databases for insight
  2. Design your own efficient databases
  3. Understand database internals
  4. Describe alternative database technologies

- Most of the time we will focus on traditional Relational Database Management Systems (RDBMS); MySQL in particular

- Toward the end of the term we will look at new technologies such as NoSQL databases (MongoDB in particular)

Class protocol

- I'll assume you've done the reading for the day, in class I'll expand/extend the material from the book, and will not simply repeat the book back to you
- I plan to spend roughly half of each class doing practice exercises
  - After I cover the additional material for the day, I'll post a series of questions to solve
  - I will randomly select one student to present their solution to those questions
  - We will see there are often many ways to efficiently solve a problem, seeing how someone else solved a problem could be useful
- **Come to class on time**

Assessment covers participation, labs, two midterms, and a final project

Participation (5%):

- This is *not* CS10, read the assigned material *before* class
- Come to class, read the course notes and find slides at:
  http://www.cs.dartmouth.edu/~tjp/cs61
- Reading from Database System Concepts, 7th edition, by Silberschatz
- Laptop use in class is ~~encouraged~~ required – Google/LLMs are your friends
- Class participation – "it's your day"

Labs (40%):

- Lab 0: gather information
- Lab 1: 5%
- Lab 2 and Lab 4: 10%
- Lab 3: 15%

Can work with one partner on labs
Both partners submit same solution
See note about AI on course web page

Midterms (2 x 15% each = 30%) – no final

Project (25%)

- Project of your choosing, but must have a transactional component
- Teams of four (neither more, nor fewer)
- Project plan (5%), EERD (5%), final presentation and write up (15%)

We will also be using Canvas and Slack for announcements and help

Online resources

Canvas

- Course announcements
- Lab submissions

Slack

- See the link to Slack on Canvas
- Let me know if you do not have access
- Do NOT post code online

Lab 0 is out now, due by next class

Lab 0

Find it on Canvas

1. Take course survey to understand your background

2. Set up MySQL and MySQL Workbench

3. Connect to a database on your localhost

4. Read and acknowledge course policies

Agenda

1. Course logistics
2. Data, information, and knowledge
3. Problems with early data management
4. Modern relational database management systems
5. Big picture of relational database design

Data versus information versus knowledge

- **Question: what is the difference between data, information, and knowledge?**
- **Data** consists of raw facts
  - Not yet processed to reveal meaning to user
  - Building blocks of information
- **Information** results from processing raw data to reveal its meaning
  - Requires context
  - Bedrock of knowledge
- **Knowledge/insight** body of information and facts about subject
  - Implies familiarity, awareness, and understanding of information
  - Includes experience and judgement

Use these to make better decisions!

Adapted from infogineering

Modern DBMS's use data models to provide users an abstract view of their data

- A **Database Management System (DBMS)** is a collection of interrelated programs to make data persistent, editable, and shareable in a secure way
- **Data models**
  - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints
  - Models are a *logical construct*, do not rely on specific file formats or data locations
  - We will focus on relational database models (at first)
- **Data abstraction**
  - Hide the complexity of data structures used to represent, create, store, update, delete, and retrieve data
  - Physical location of data also not something the user need worry about, database hides this information

Agenda

1. Course logistics
2. Data, information, and knowledge
3. Problems with early data management
4. Modern relational database management systems
5. Big picture of relational database design

In the early days, database applications were built directly on top of file systems

Problems

Sales

Manufacturing

Shipping

Each department keeps records for its own purposes (islands of information) in applications custom written for each group

What could go wrong?

- **Data redundancy and inconsistency**
  - Data is stored in multiple file formats and locations
  - Results in duplication of information in different files
  - Data may become inconsistent between departments as changes are made
  - Eliminating data redundancy will be a big topic for us this term

Stamp out and eradicate superfluous redundancies!

In the early days, database applications were built directly on top of file systems

Problems

Sales

Manufacturing

Shipping

Each department keeps records for its own purposes (islands of information) in applications custom written for each group

What could go wrong?

- Data redundancy and inconsistency
- **Difficulty accessing data**
  - Need to write a new program to carry out each new task
  - Change the file format and break all applications that use it! (no data independence)

In the early days, database applications were built directly on top of file systems

Problems

Sales

Manufacturing

Shipping

Each department keeps records for its own purposes (islands of information) in applications custom written for each group

What could go wrong?

- Data redundancy and inconsistency
- Difficulty accessing data
- **Integrity problems**
  - Integrity constraints (e.g., account balance must be > 0) become "buried" in program code rather than being stated explicitly
  - Difficult to add new constraints or change existing ones, especially across departments

In the early days, database applications were built directly on top of file systems

Problems

Sales

Manufacturing

Shipping

Each department keeps records for its own purposes (islands of information) in applications custom written for each group

What could go wrong?

- Data redundancy and inconsistency
- Difficulty accessing data
- Integrity problems
- **Atomicity of updates**
  - Failures may leave database in an inconsistent state with partial updates carried out (account balance example)

In the early days, database applications were built directly on top of file systems

Problems

Sales

Manufacturing

Shipping

Each department keeps records for its own purposes (islands of information) in applications custom written for each group

What could go wrong?

- Data redundancy and inconsistency
- Difficulty accessing data
- Integrity problems
- Atomicity of updates
- **Concurrent access by multiple users**
  - Want multiple users accessing same data at same time, without performance degradation

In the early days, database applications were built directly on top of file systems

Problems

Sales

Manufacturing

Shipping

Each department keeps records for its own purposes (islands of information) in applications custom written for each group

What could go wrong?

- Data redundancy and inconsistency
- Difficulty accessing data
- Integrity problems
- Atomicity of updates
- Concurrent access by multiple users
- **Security**
  - Hard to provide user access to some, but not all, data

In the early days, database applications were built directly on top of file systems

Problems

Sales

Manufacturing

Shipping

Each department keeps records for its own purposes (islands of information) in applications custom written for each group

What could go wrong?

- Data redundancy and inconsistency
- Difficulty accessing data
- Integrity problems
- Atomicity of updates
- Concurrent access by multiple users
- Security

Modern database management systems (attempt to) solve all these problems

Agenda

1. Course logistics
2. Data, information, and knowledge
3. Problems with early data management
4. Modern relational database management systems
5. Big picture of relational database design

Relational database systems store data in relations (tables) and also store metadata

Relational database

attributes

Instructor relation

Relation instances

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

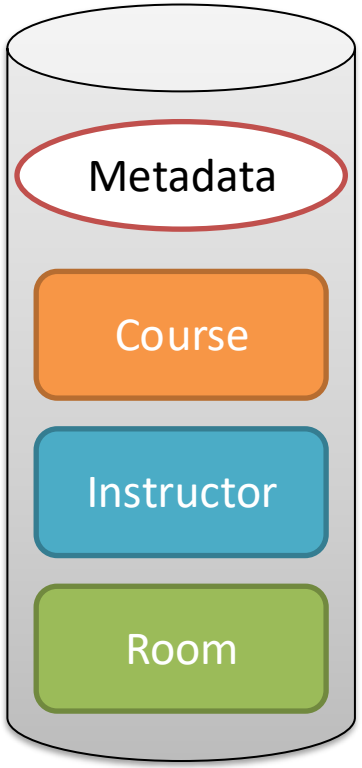
Data in a relational database

- Data stored in **relations** (tables)
- Relations are made up of **relation instances** (rows or tuples)
- Relation instances are made up of a fixed number of **attributes** (fields) of fixed type
- Related relations are contained in a **schema**
- Database may store multiple schemas

Metadata

- Metadata is data about the data stored in the database
- Describes things such as each field's name, data type, if ok to be NULL
- Also describes the relationships between data
- Metadata kept in **data dictionary**

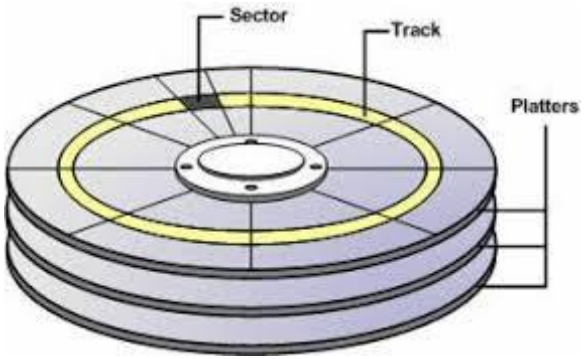
A **database instance** is a snapshot of the database at a point in time



College database schema

How and where data is stored is abstracted (hidden) from users

Simplified database architecture

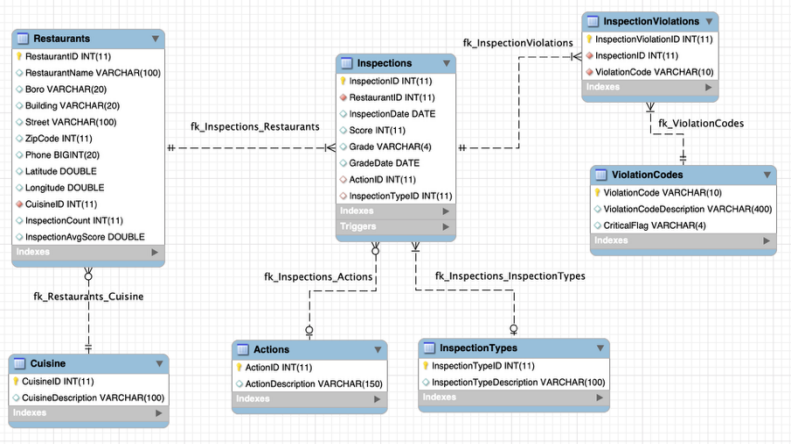


Physical level

- Data structures used to represent, create, store, update, delete, and retrieve data
- Indicates where (e.g., where and on which disks) data is stored
- **Physical schema** physical layout of database

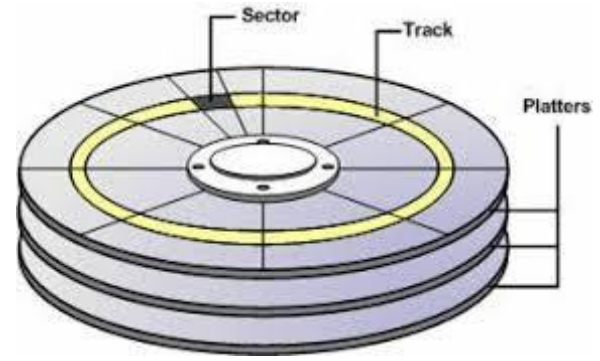
How and where data is stored is abstracted (hidden) from users

Simplified database architecture



Logical level

- Describes data and relationships between relations at a high level
- **Logical schema** is the overall conceptual database design
- Hides physical layer details; makes data physically independent from applications (like an ADT)

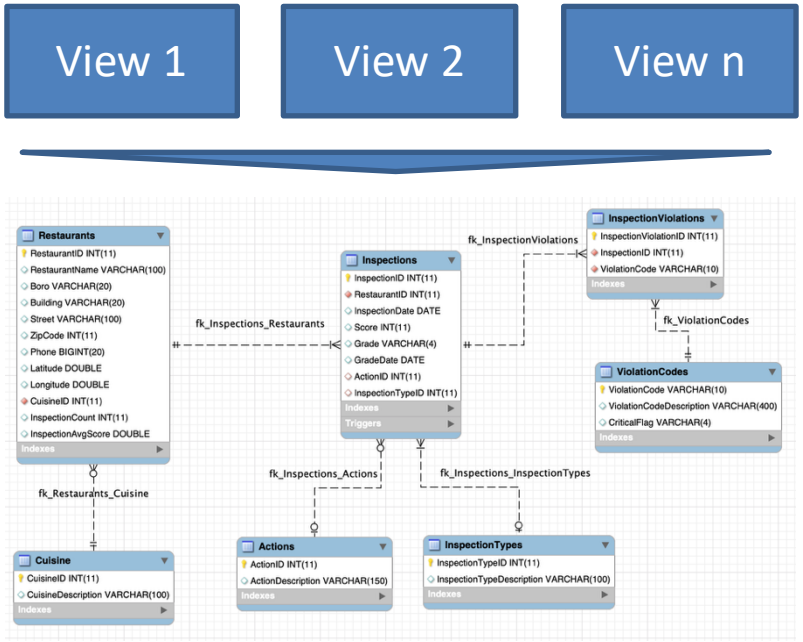


Physical level

- Data structures used to represent, create, store, update, delete, and retrieve data
- Indicates where (e.g., where and on which disks) data is stored
- **Physical schema** physical layout of database

How and where data is stored is abstracted (hidden) from users

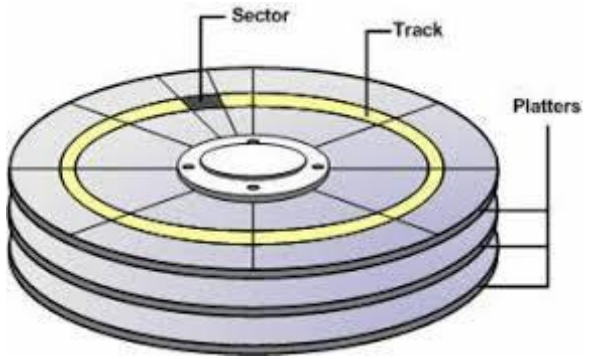
Simplified database architecture



- **Views** provide data needed for applications
- Can hide data (such as salary) for security or confidentiality reasons

Logical level

- Describes data and relationships between relations at a high level
- **Logical schema** is the overall conceptual database design
- Hides physical layer details; makes data physically independent from applications (like an ADT)



Physical level

- Data structures used to represent, create, store, update, delete, and retrieve data
- Indicates where (e.g., where and on which disks) data is stored
- **Physical schema** physical layout of database

SQL allows us to create a database schema, then use that schema to manipulate data

Structured Query Language, aka SQL, aka 'sequel'

Create/manage database schema → Use data (CRUD)

Data Definition Language (DDL)

- Notation for defining and managing the database's logical and physical schemas
- DDL creates **data dictionary** containing:
  - Database schema
  - Integrity constraints (what values attributes can take on)
  - Security (who can access what)

Data Manipulation Language (DML)

- Language for accessing and updating the data
- DML called **query language** allowing
  - Create (Insert)
  - Read (Select)
  - Update (Update)
  - Delete (Delete)

Structured Query Language (S-Q-L or Sequel) does both!

SQL will our primary means of interacting with the database

Structured Query Language (SQL)

- SQL query language is nonprocedural
- A query takes as input one or more tables and always returns a single table
- Example to find all instructors in Comp. Sci. dept

  SELECT *
  FROM *instructor*
  WHERE *dept_name* = 'Comp. Sci.'

- SQL is **NOT** a Turing machine equivalent language – there are some things it cannot compute
- To be able to compute complex functions SQL is usually interfaced with some higher-level language (e.g., Python, Java, PHP, JavaScript)
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL (less common today)
  - **Application Program Interface** (API) which allow SQL queries to be sent to a database on behalf of an application; API then returns result

Today applications (and users) normally access a database through an API

Three-tiered architecture

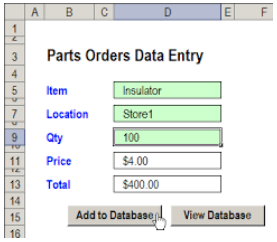
Smart phone apps

Web browser

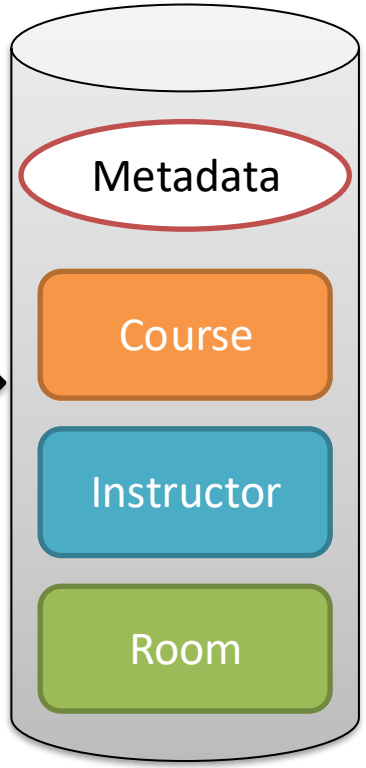
Network

API

Database Management System (DBMS)



"Thick client" apps



College database schema

Today applications (and users) normally access a database through an API

Three-tiered architecture

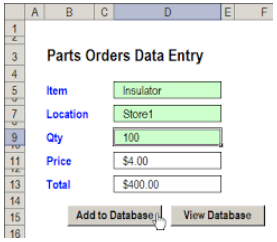
Smart phone apps

Web browser

Network

API

Database Management System (DBMS)



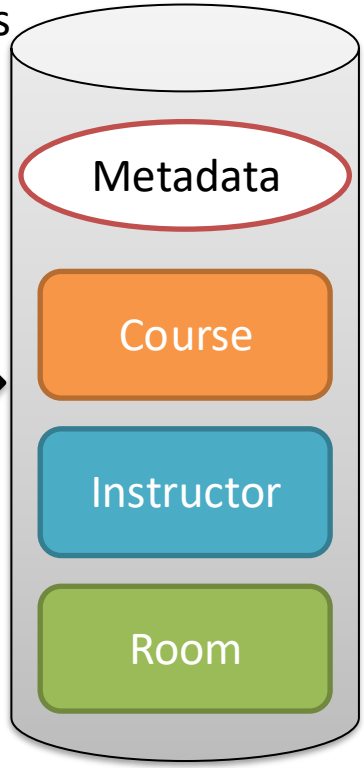
"Thick client" apps

Advantages

1. Allows data to be shared between multiple applications
2. Creates a single repository of knowledge
3. Manages security

Tier 1: DBMS

- Manages database structure
- Controls access to database
- Allows data to be shared



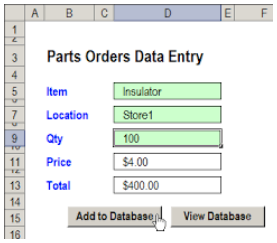
College database schema

Today applications (and users) normally access a database through an API

Three-tiered architecture

Smart phone apps

Web browser



"Thick client" apps

Network

Advantages

1. Abstracts data access
2. Data storage can be changed without changing all user applications

API

Tier 2: API

- Provides access to database via web services
- May also be web server for web pages

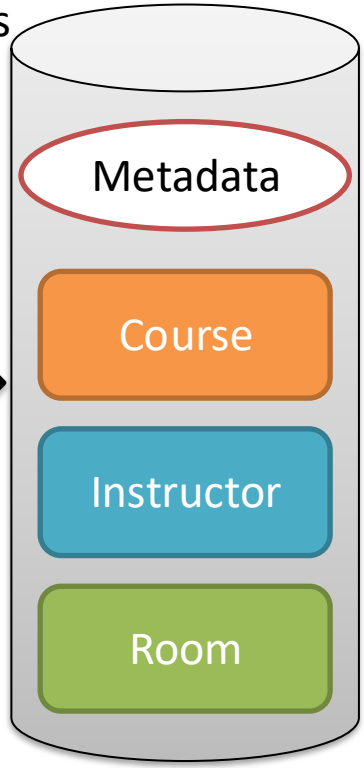
Advantages

1. Allows data to be shared between multiple applications
2. Creates a single repository of knowledge
3. Manages security

Database Management System (DBMS)

Tier 1: DBMS

- Manages database structure
- Controls access to database
- Allows data to be shared



College database schema

Today applications (and users) normally access a database through an API

Three-tiered architecture

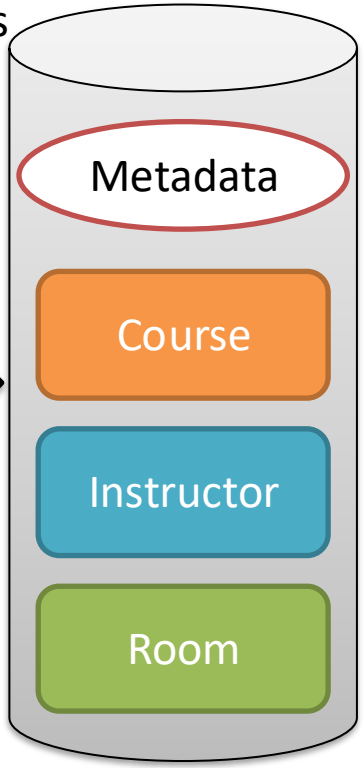
Smart phone apps

Web browser

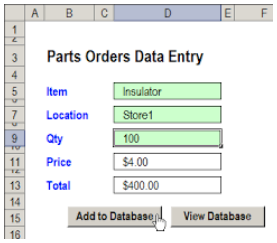
Network

API

Database Management System (DBMS)



College database schema



"Thick client" apps

Tier 3: Applications

Advantages

1. Abstracts data access
2. Data storage can be changed without changing all user applications

Tier 2: API

- Provides access to database via web services
- May also be web server for web pages

Advantages

1. Allows data to be shared between multiple applications
2. Creates a single repository of knowledge
3. Manages security

Tier 1: DBMS

- Manages database structure
- Controls access to database
- Allows data to be shared

There are several types of databases and metrics to distinguish between them

Use

- **Online Transaction Processing** (OLTP) – production databases
- **Online Analytical Processing** (OLAP) – reporting databases, use historical data, "Business intelligence"
- **Data warehouse** – data optimized for decision support, may use data from external sources
- Pros? Cons?

Location

- **Centralized** – database located in one location (our main focus)
- **Distributed** – many connected "mini databases" each may hold only a shard of the entire data (end of class)
- Cloud or onsite
- Pros? Cons?

Database type

- General purpose (our focus in CS61)
- Discipline specific (GIS, graph/vector databases)
- **Structured** vs. **unstructured** data
- Relational vs. NoSQL

DBMS's handle several important functions

Database management functions

1. **Data dictionary management**
   - Data dictionary: stores definitions of data elements and their relationships (metadata)
   - DBMS looks up data elements and relationships, so you don't have to!
   - Any changes made to structure of database update data dictionary
   - Many times applications will not need to be updated after changes to database structure (data independence)
2. **Data storage management**
   - You deal with logical organization of data; DBMS handles physical storage for you
   - Performance tuning ensures efficient performance
3. **Data transformation and presentation**
   - Data is formatted to conform to logical expectations (e.g., date handled according to location, US vs. UK)

DBMS's handle several important functions

Database management functions

4. **Security management**
   - Enforces user security and data privacy
   - Only authorized users able to view or alter database
5. **Multiuser access**
   - Sophisticated algorithms ensure that multiple users can access the database concurrently without compromising its integrity
   - Accept end-user requests via multiple, different network environments
6. **Backup and recovery management**
   - Enables recovery of the database after a failure
7. **Data integrity management**
   - Minimizes redundancy and maximizes consistency

Can’t we just do this with a spreadsheet?

Exercise

See day1.xlsx

We will fix these problems soon

Data anomalies

- *Consistency anomalies* – entering same data, but with different name
- *Update anomalies* – If data stored in multiple rows, must update all rows (ex. If update CourseName, must update all rows for that Course)
- *Insertion anomalies* – If an instructor exists, but is not assigned to a class, they will not appear in the spreadsheet
- *Deletion anomalies* – If instructor teaches only one class, but you delete that class, the instructor disappears

Good database design is critical!

Good design vs. poor design

- Well-designed database: facilitates data management and generates accurate and valuable information
- Poorly designed database causes difficult-to-trace errors that may lead to poor decision making
- **Good applications can't overcome bad database designs**
- The existence of a DBMS does not guarantee good data management, nor does it ensure that the database will be able to generate correct and timely information
- Ultimately, the end user and the database designer decide what data will be stored in the database

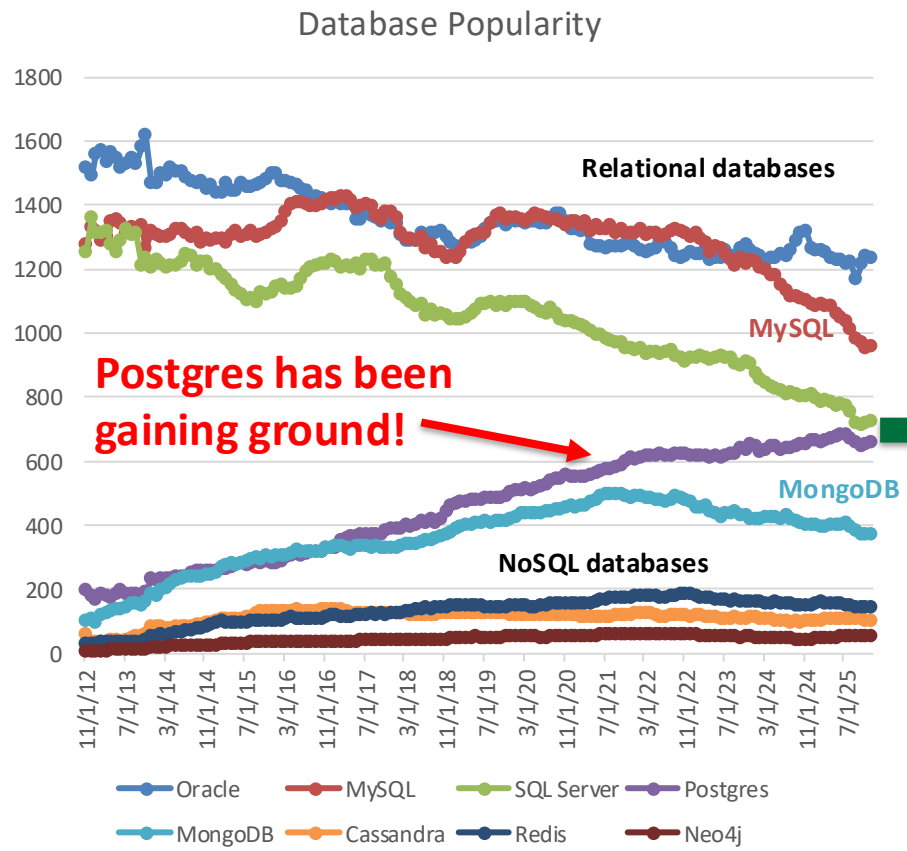
There are some disadvantages to database systems

Database disadvantages

- Increased costs
- Management/training complexity
- Maintaining software currency
- Vendor dependence
- Frequent upgrade/replacement cycles

There are number of popular DBMS's in use today, we will use MySQL and Mongo

Popular DBMS



https://db-engines.com/en/ranking_trend

Relational → Focus in CS61

- *MySQL/MariaDB (open source, but owned by Oracle, MariaDB is fork)*
- Oracle (king of the hill, but expensive)
- Microsoft SQL Server (also Access, easy to use compared with Oracle)

NoSQL

- *Mongo (most popular NoSQL, has security concerns?)*
- Redis (in-memory data structure store, used as a database, cache and message broker)
- Cassandra (hybrid key-value & column-oriented DB)
- Neo4j graph database

Agenda

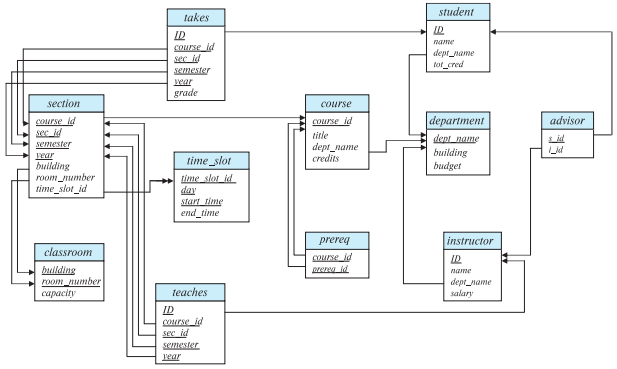
1. Course logistics
2. Data, information, and knowledge
3. Problems with early data management
4. Modern relational database management systems
5. Big picture of relational database design

Big picture of relational database design

Relational Database Management System

- Normally represented graphically as a cylinder
- Holds data in relations (tables)

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000



Relations

- Each relation holds data about people, places, things or events (nouns)
- Tables consist of rows and columns
- Each row (tuple) represents one person, place, thing, or event
- Each column represents one attribute about a person, place, thing, or event (e.g., name)
- A column (FK) can refer to a column (PK) in another table, creating a relationship between tables

Database schema

- Logical collection of tables and relationships
- Minimizes storing multiple copies of data
- Look up additional data in another table if needed using key

Relations in a relational database must conform to eight rules

Table characteristics

6 rows (tuples) with 3 columns (attributes) for each row

Department table

DepartmentID	DepartmentName	DepartmentBuilding
1	Computer Science	ECSC
2	Biology	Life Sciences Center
3	English	Sanborn
4	Chemistry	Burke
5	Government	Silsby
6	Engineering	Thayer

1. Each table is perceived as a two-dimensional structure of rows and columns

Relations in a relational database must conform to eight rules

Table characteristics

Each row describes one department

Department table

DepartmentID	DepartmentName	DepartmentBuilding
1	Computer Science	ECSC
2	Biology	Life Sciences Center
3	English	Sanborn
4	Chemistry	Burke
5	Government	Silsby
6	Engineering	Thayer

1. Each table is perceived as a two-dimensional structure of rows and columns
2. Each row (tuple) represents a single entity occurrence within the entity set

Relations in a relational database must conform to eight rules

Table characteristics

Each column represents a different attribute of a department (e.g., ID, Name, Building) and each column has a different name

Department table

DepartmentID	DepartmentName	DepartmentBuilding
1	Computer Science	ECSC
2	Biology	Life Sciences Center
3	English	Sanborn
4	Chemistry	Burke
5	Government	Silsby
6	Engineering	Thayer

1. Each table is perceived as a two-dimensional structure of rows and columns
2. Each row (tuple) represents a single entity occurrence within the entity set
3. Each column represents an attribute, and each column has distinct name

Relations in a relational database must conform to eight rules

Table characteristics

Single entry in each cell

Department table

DepartmentID	DepartmentName	DepartmentBuilding
1	Computer Science	ECSC
2	Biology	Life Sciences Center
3	English	Sanborn
4	Chemistry	Burke
5	Government	Silsby
6	Engineering	Thayer

1. Each table is perceived as a two-dimensional structure of rows and columns
2. Each row (tuple) represents a single entity occurrence within the entity set
3. Each column represents an attribute, and each column has distinct name
4. Each intersection of a row and column represents a single data value

Relations in a relational database must conform to eight rules

Table characteristics

In column 1 all entries are numeric, in other columns each entry is character data

Department table

DepartmentID	DepartmentName	DepartmentBuilding
1	Computer Science	ECSC
2	Biology	Life Sciences Center
3	English	Sanborn
4	Chemistry	Burke
5	Government	Silsby
6	Engineering	Thayer

1. Each table is perceived as a two-dimensional structure of rows and columns
2. Each row (tuple) represents a single entity occurrence within the entity set
3. Each column represents an attribute, and each column has distinct name
4. Each intersection of a row and column represents a single data value
5. All values in a column must conform to the same data format

Relations in a relational database must conform to eight rules

Table characteristics

Domain is positive integers for column 1, alphanumeric characters for others

Department table

DepartmentID	DepartmentName	DepartmentBuilding
1	Computer Science	ECSC
2	Biology	Life Sciences Center
3	English	Sanborn
4	Chemistry	Burke
5	Government	Silsby
6	Engineering	Thayer

1. Each table is perceived as a two-dimensional structure of rows and columns
2. Each row (tuple) represents a single entity occurrence within the entity set
3. Each column represents an attribute, and each column has distinct name
4. Each intersection of a row and column represents a single data value
5. All values in a column must conform to the same data format
6. Each column has a specific range of values known as the **attribute domain**

Relations in a relational database must conform to eight rules

Table characteristics

Departments not ordered in any particular fashion, except CS is first ;-)

Department table

DepartmentID	DepartmentName	DepartmentBuilding
1	Computer Science	ECSC
2	Biology	Life Sciences Center
3	English	Sanborn
4	Chemistry	Burke
5	Government	Silsby
6	Engineering	Thayer

1. Each table is perceived as a two-dimensional structure of rows and columns
2. Each row (tuple) represents a single entity occurrence within the entity set
3. Each column represents an attribute, and each column has distinct name
4. Each intersection of a row and column represents a single data value
5. All values in a column must conform to the same data format
6. Each column has a specific range of values known as the **attribute domain**
7. The order of the rows and columns is immaterial to the DBMS

Relations in a relational database must conform to eight rules

Table characteristics

DepartmentID is a *Primary Key* (PK), it can uniquely identify each row

No two rows can be *exactly* the same

Department table

DepartmentID	DepartmentName	DepartmentBuilding
1	Computer Science	ECSC
2	Biology	Life Sciences Center
3	English	Sanborn
4	Chemistry	Burke
5	Government	Silsby
6	Engineering	Thayer

1. Each table is perceived as a two-dimensional structure of rows and columns
2. Each row (tuple) represents a single entity occurrence within the entity set
3. Each column represents an attribute, and each column has distinct name
4. Each intersection of a row and column represents a single data value
5. All values in a column must conform to the same data format
6. Each column has a specific range of values known as the **attribute domain**
7. The order of the rows and columns is immaterial to the DBMS
8. Each table must have an attribute or combination of attributes that uniquely identifies each row

NOTE: a value of NULL means the value is not known or empty; Primary keys cannot be null

Highlander theory of database design: "There can be only one (copy of the data)!"

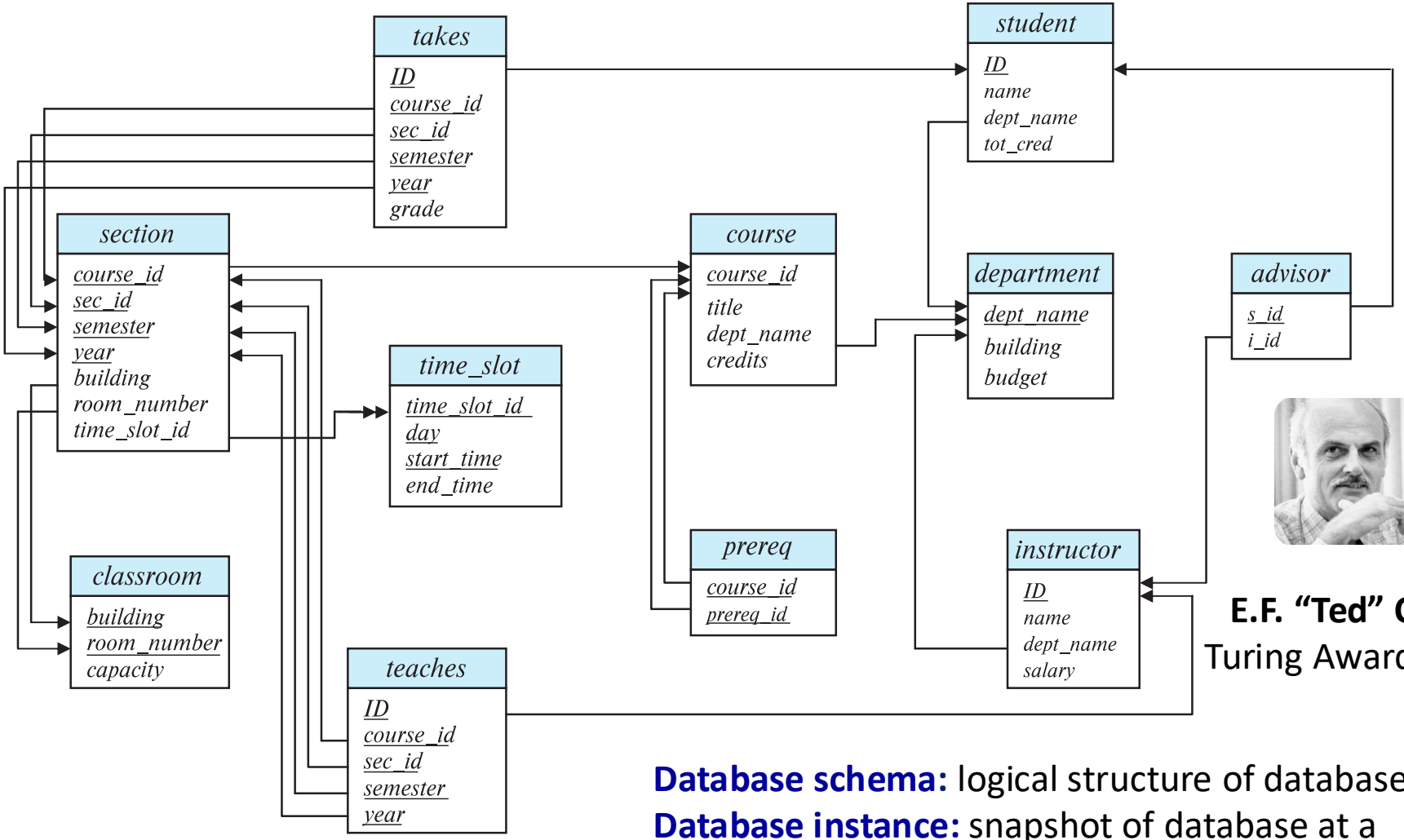
Avoid storing the same data multiple times, store it once!

- We will discuss this idea further when we cover normalization
- For now, tables hold data about one type of entity (e.g., customer), each row in the table is an instance of that thing (e.g., Alice Jones)

- Each table holds data about a type of entity: a person, place, thing or event
- Avoid storing the same data in multiple tables!
- Example:
  - Do not store a customer's address in multiple tables
  - Instead create one table that represents customers and store their address as columns in that single table
  - Other tables that need the customer's address look it up in this table
  - If address changes, only one update needed

Look up data in other tables when needed

Database schema diagram



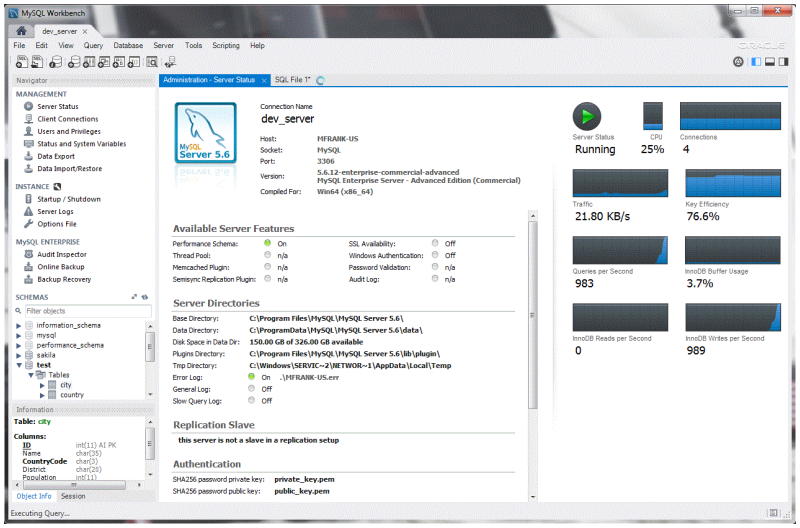
E.F. "Ted" Codd
Turing Award 1981

Database schema: logical structure of database
Database instance: snapshot of database at a point in time

Install software

MySQL database

MySQL workbench



Before next class

1. Complete Lab 0

2. Read textbook (parts of chapter 2) for next class as shown on schedule (also skim chapter 1)