


CS 61: Database Systems

Introduction

Zoom poll

Where are you located?

Agenda

- 
1. Course logistics
 2. Data, information, and knowledge
 3. Problems with early data management
 4. Modern relational database management systems

We are meeting online this term, that may cause issues...

- We will use Zoom for class meetings during normal class hours (I will record and post each class session)
- We will see what works as we go, but here are some rules for starters:
 - Start with your camera off and microphone muted (there are a lot us!)
 - If you would like to ask a question, use the “Raise hand” feature in Zoom, then when called on, turn on your video camera and ask
 - We will make additional rules as we need them...
- 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 had planned to spend roughly half of each class doing group exercises
 - I will try to do that online with Zoom’s break out rooms
 - After I cover the additional material for the day, I’ll post a series of questions
 - Zoom will assign you to a break-out to work out answers with other students
 - I will randomly select one student to present their solution to the class
 - If you are selected, but not online live, please post your solution on Canvas
 - We will see there are often many ways to efficiently solve a problem, seeing how someone else solved a problem could be useful

This class is about database systems

- Four main goals for the course:
 1. Query existing databases for insight into the data they contain
 2. Design your own efficient databases
 3. Understand what goes on under the hood
 4. Describe new and developing 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) and blockchains
- Guest speakers will augment the experience

Material will be covered in lecture, labs, one midterm, and a term-long project

Lectures (10%):

- 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 is your friend
- Class participation – “it’s your day”

Labs (30%):

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

Midterm (20%) – no final

Project (40%)

- Project of your choosing, but must have a transactional component
- Teams of four (neither more, nor fewer)
- Project plan (5%), EERD (10%), final presentation and write up (25%)

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

Online resources

Canvas

- Course announcements
- Lab submissions

Slack


- I will post a link to join the channel after class
- Let the Almas (Grad TA) know if you do not have access
- We will use Slack in place of Piazza
- You can post code related to your project, but not related to the labs or the midterm
- You can post code for in-class problems after the class period ends

Lab 0 is out now, due by next class

Lab 0

- Find it on Canvas
- Take course survey to understand your background
- Set up MySQL and MySQL Workbench
- Connect to a database on your localhost
- 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

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



**The
New York
Times**



facebook



Virtually all non-trivial applications have a database component

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?

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!**

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

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 with other departments as changes are made
 - Eliminating data redundancy will be a big thread for us this term

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
systems solve
these problems**

Agenda

1. Course logistics
2. Data, information, and knowledge
3. Problems with early data management
-  4. Modern relational database management systems

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

Relational database

Instructor relation

Relation instances

attributes			
<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

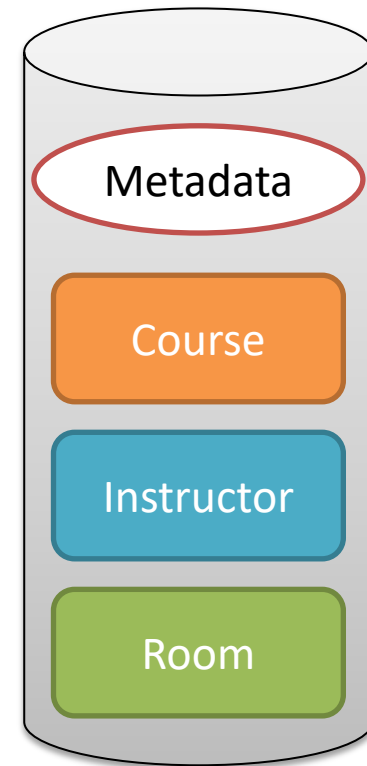
Data in a relational database

- 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**

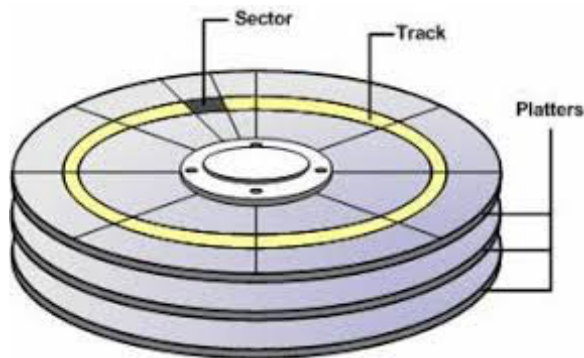
A **database instance** is a snapshot of the database at a point in time



College database schema

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

Simplified database architecture

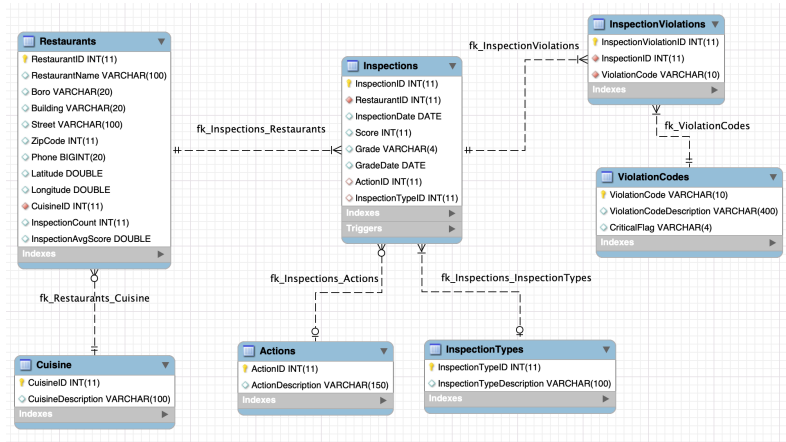


Physical level

- 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

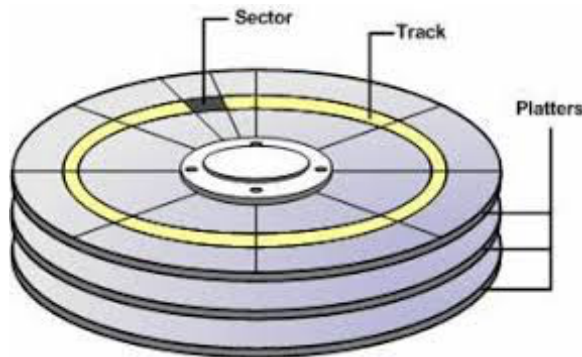


Logical level

- 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

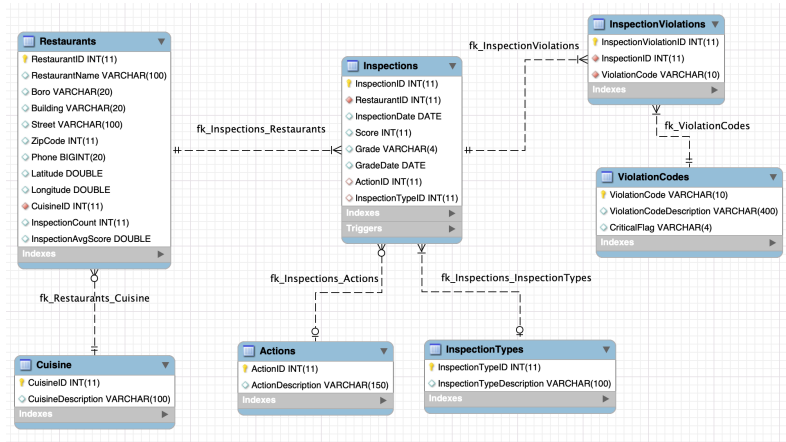
Simplified database architecture

View 1

View 2

View n

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

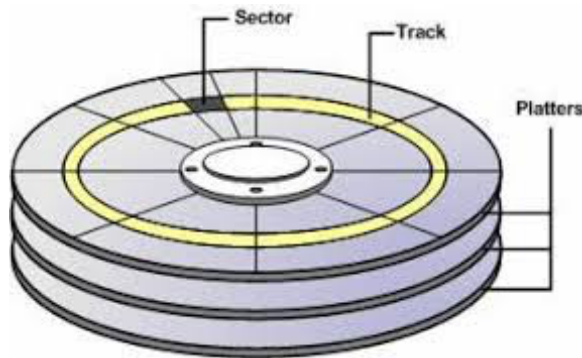


Logical level

- 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)

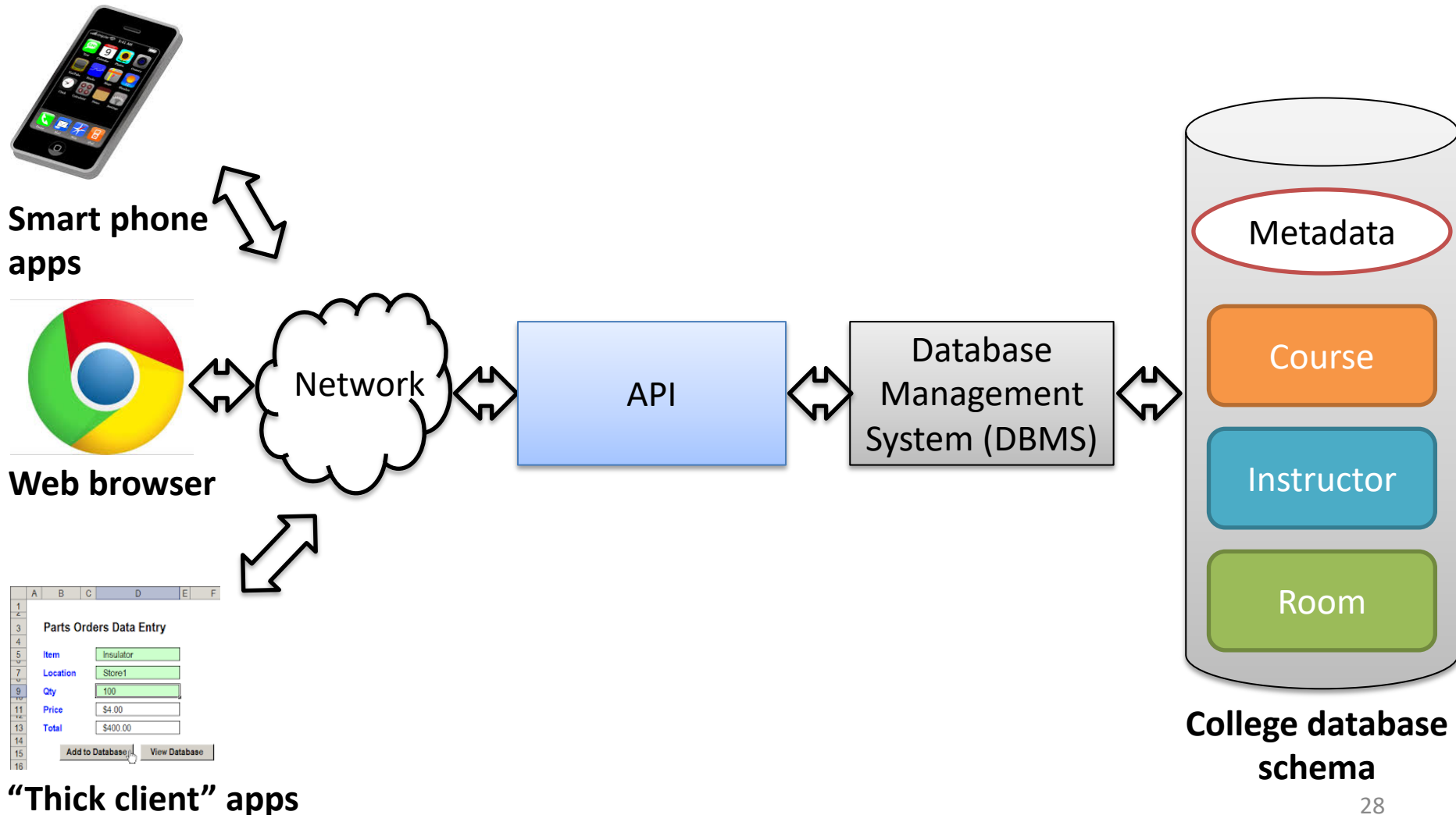
- SQL query language is nonprocedural
- A query takes as input one or more tables and always returns a single table
- Example to find all instructors in Comp. Sci. dept

```
SELECT name  
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



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

Three-tiered architecture



Smart phone apps



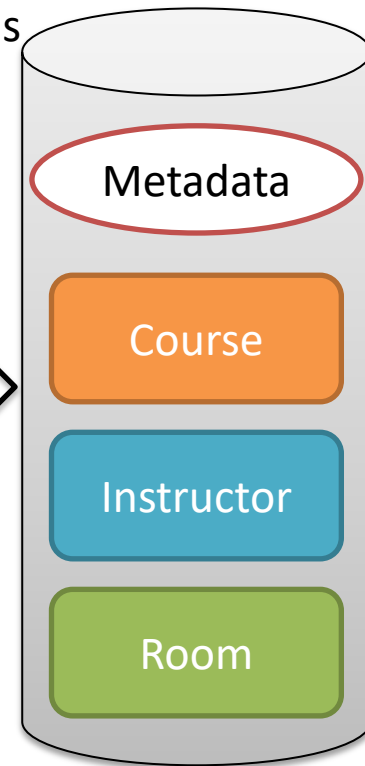
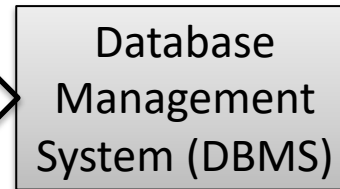
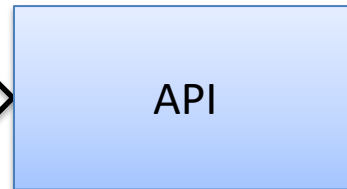
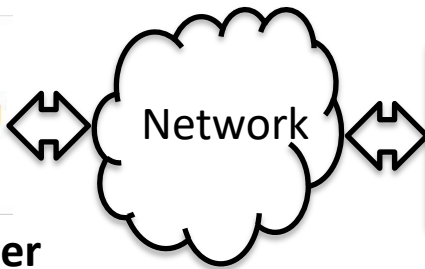
Web browser

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						

Parts Orders Data Entry

Item	<input type="text" value="Insulator"/>
Location	<input type="text" value="Store1"/>
Qty	<input type="text" value="100"/>
Price	<input type="text" value="\$4.00"/>
Total	<input type="text" value="\$400.00"/>

“Thick client” apps



College database schema

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

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

Three-tiered architecture



Smart phone apps

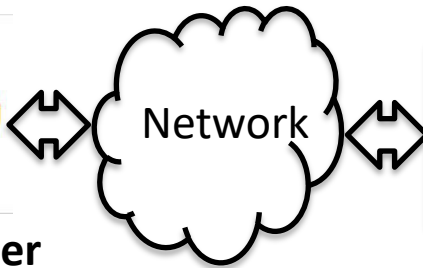


Web browser

A	B	C	D	E	F
1					
2					
3					
4					
5	Item		Insulator		
6	Location		Store1		
7	Qty		100		
8	Price		\$4.00		
9	Total		\$400.00		
10					
11					
12					
13					
14					
15					
16					

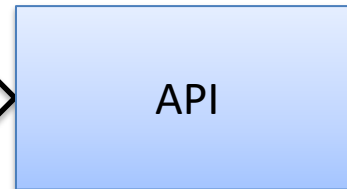
Add to Database View Database

“Thick client” apps



Advantages

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

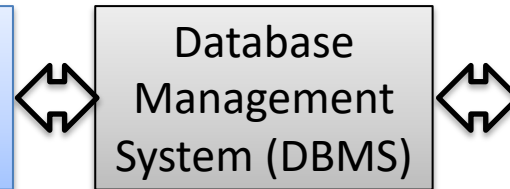


Tier 2: API

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

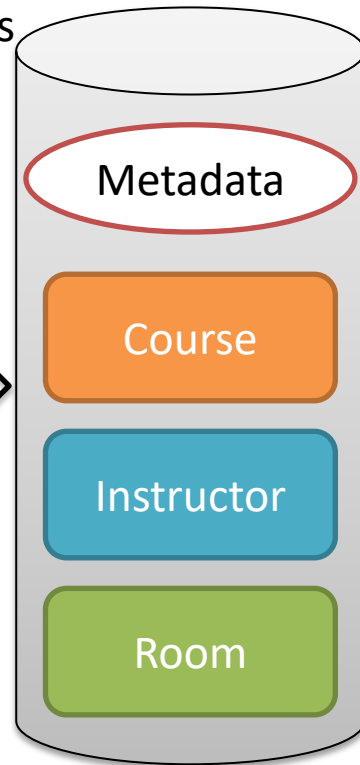
Advantages

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



College database schema

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

Three-tiered architecture



Smart phone apps

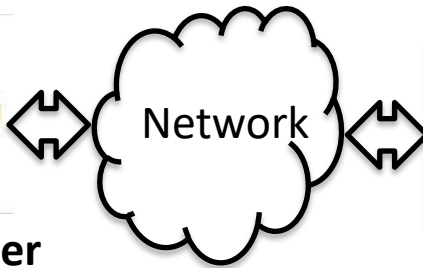


Web browser

	A	B	C	D	E	F
1						
2						
3						
4						
5	Item			Insulator		
6	Location			Store1		
7	Qty			100		
8	Price			\$4.00		
9	Total			\$400.00		
10						
11						
12						
13						
14						
15						
16						

Add to Database View Database

“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

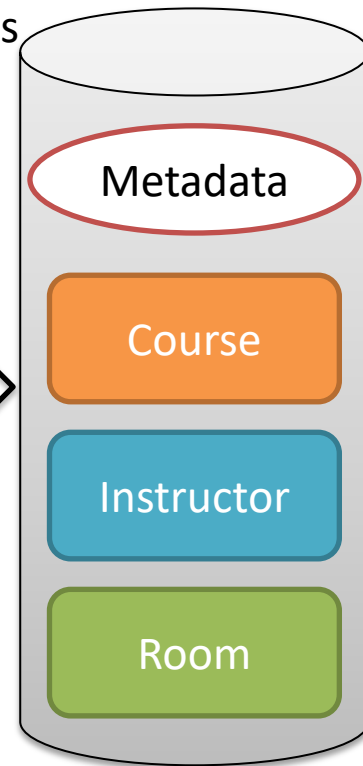
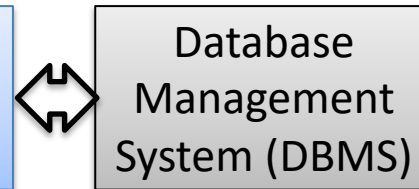
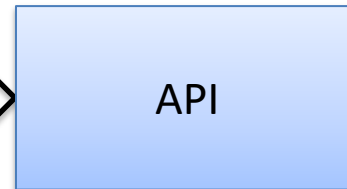
- Provides access to database via web services
- May also be web server for web pages

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



College database schema

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 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 (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 act on database

5. **Multuser 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

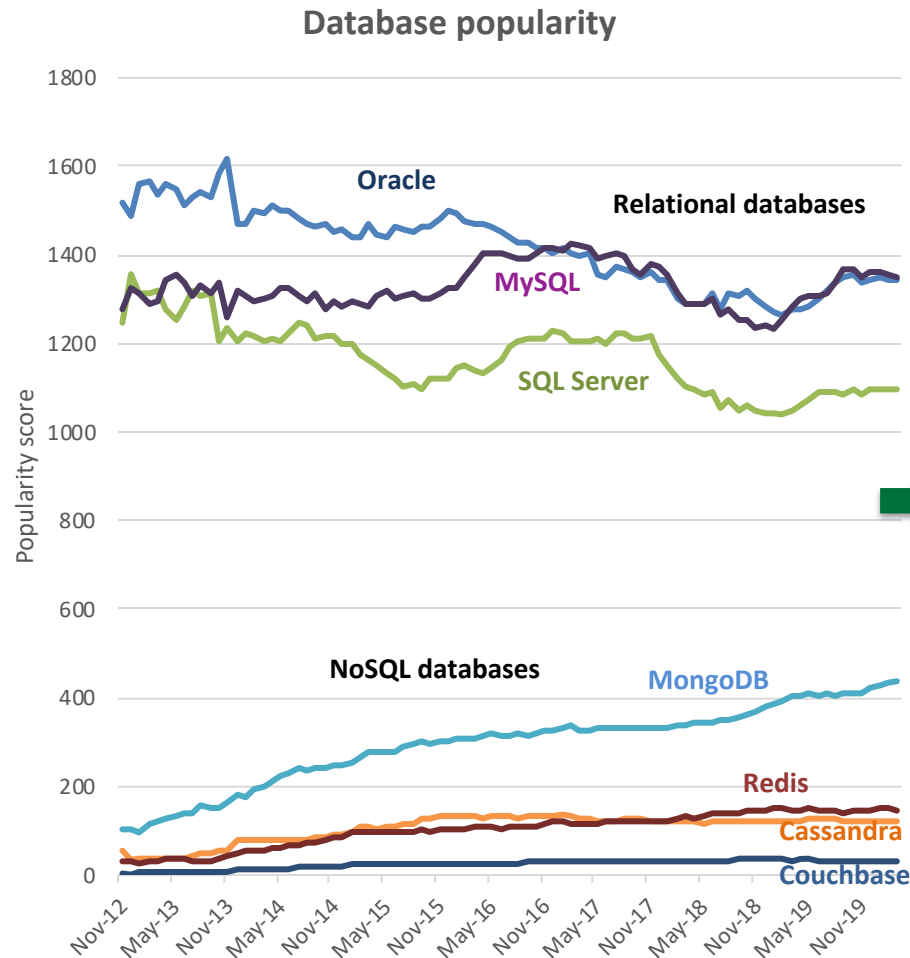
There are some disadvantages to database systems

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



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)
- Couchbase (key/value store)

Last thought: database skills are in high demand

TOP TECH SKILLS

2019 Rank	Occupation	Change in Rank from 2018
1	SQL	—
2	Java	—
3	JavaScript	▲ 1
4	Project Management	▼ 1
5	Python	▲ 1
6	Linux	▼ 1
7	Oracle	—
8	Microsoft C#	—
9	Scrum	▲ 1
10	Quality Assurance and Control	▼ 1

Job placement firm Dice analyzed 6 million job postings for most frequently sought tech skills in 2020

Employers are looking to hire people with database skills (e.g., you in 10 weeks)!

Before next class

1. Complete Lab 0
2. Read textbook (parts of chapter 2 and 3) for next class as shown on schedule (also skim chapter 1)

