CSE544
Data Management

Lecture 1: Introduction,

Relational Data Model

Course Staff

- Instructor: Dan Suciu
  - Office hours: Mondays, 2:30-3:20

- TA: Kyle Deeds
  - Office hours: Fridays, 11:30-12:20

Goals of the Class

- Relational Data Model
  - Data models
  - Data independence
  - Declarative query language.
- Relational Database Systems
  - Storage
  - Query execution
  - Query optimization

A Note for Non-Majors

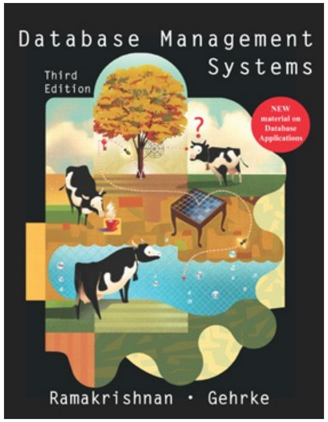
- For the Data Science option: take 414
- For the Advanced Data Science option: take 544

- 544 is an ***advanced*** class, not an introduction.

- Unsure? Look at the short quiz on the website.

Readings

- Lecture notes (the slides)
  - Posted on class website after each lecture
- Background
  - Database Management Systems. **Third Ed**. Ramakrishnan and Gehrke. McGraw-Hill.



- Paper reviews
  - Mix of old seminal papers and new papers
  - Papers are available on class website

Class Resources

Website: lectures, assignments

- http://www.cs.washington.edu/544
- Lectures 1/17 and 2/21 moved to Friday

Canvas: zoom, videos

Ed: discussion board

Other Resources

- [Database course at CMU]
  - Low level: storage, transactions
- [Database theory course at Berkeley]
  - Theory: complexity, information theory
- Our course is in between

Evaluation

- Assignments 40%
- Reviews 10%
- Project 40%
- Intangibles 10%

Assignments – 40%

- **HW1**: Local DBMS (postgres) posted!
- **HW2**: Cloud-based DBMS (snowflake)
- **HW3**: Query Execution and SimpleDB
- **HW4**: Datalog

- See course calendar for deadlines
- Late assignments w/ ***very*** valid excuse

Paper reviews – 10%

- Recommended length: ½ page – 1 page
  - Summary of the main points of the paper
  - Critical discussion of the paper
  - R1 due on 1/10
- Grading: credit/patial-credit/no-credit
- Submit review *before* the lecture

Project – 40%

Work alone, or in a team of 2-3 students

Topic:

- Come up with your own or choose from our list
- Best: related to your research
- Must be about databases / data management
- Must involve some significant engineering
- Open ended

Project – 40%

Dates posted on the calendar page:

- **P1**: Form groups
- **P2**: Project proposal
- **P3**: Milestone report
- **P4**: Poster presentation Wed. March 6
- **P5**: Project final report

Intangibles 10%

- Class participation
- Exceptionally good reviews, or homework, or project
- Etc, etc

How to Turn In

- Homeworks: gitlab
- Project: gitlab
- Reviews: google forms

Now onward to the world of databases!

Database

- **Database** = collection of files storing inter-related data about real world entities and relationships.
- **Entities**: e.g. products, suppliers, customers, employees, warehouses
- **Relationships**: e.g. suppliers-products, customer-products, employee-manages-employee

Database Management System

- **DBMS**: a software system designed to provide data management services

- Examples
  - Oracle, DB2 (IBM), SQL Server (Microsoft)
  - Snowflake, Redshift
  - PostgreSQL, Duckdb, MySQL, Sqlite

Database Example

A database of products and suppliers:

- **Product:** has a name, a price, a color
- **Supplier:** has a name, the products it supplies, city

Flat File Strawman

- Store data in csv files
- Manage your data in python

Flat File Strawman

- Store data in csv files
- Manage your data in python

Product(name, price, color)

```
iPhone, 599, gray
iPhone, 999, black
Gizmo, 399, blue
Pizza, 29, red
```

Flat File Strawman

- Store data in csv files
- Manage your data in python

Product(name, price, color)

```
iPhone, 599, gray
iPhone, 999, black
Gizmo, 399, blue
Pizza, 29, red
```

Supplier(name, product, city)

```
ACME, iPhone, Seattle
Walmart, iPhone, Renton
Costco, Pizza, Seattle
```

Flat File Strawman

- Store data in csv files
- Manage your data in python

Product(name, price, color)

```
iPhone, 599, gray
iPhone, 999, black
Gizmo, 399, blue
Pizza, 29, red
```

Flat File Strawman

- Store data in csv files
- Manage your data in python

Product(name, price, color)

iPhone	599	gray
iPhone	999	black
Gizmo	399	blue
Pizza	29	red

Find the price of Gizmo:

Flat File Strawman

- Store data in csv files
- Manage your data in python

Product(name, price, color)

```
iPhone, 599, gray
iPhone, 999, black
Gizmo, 399, blue
Pizza, 29, red
```

Find the price of Gizmo:

```
with open('product.csv') as f:
    r = csv.reader(f, delimiter=',')
    for t in r:
        if t[0] == "Gizmo":
            print(t[1])
```

Issues with Flat Files

Need to implement many generic tasks:

- Data integrity
- Efficient implementation
- Concurrency, durability
- Etc.

Flat Files: Data Integrity

- Price should be a number
- Supplier names should be unique
- Suppliers may supply >1 products
- Each supplier in a single city

Flat Files: Implementation

- How do we update/insert/delete?

- Data may be larger than main memory

- A query may traverse multiple files

- Data may be distributed

Flat Files: Concurrency, Durability

- What if multiple applications touch the data?

- What if the system crashes in the middle of an update?

Database Management System

A software system designed to provide data management services:

- Enforce integrity constraints
- Evaluate queries efficiently
- Handle concurrency, recovery
- …

Important Terminology

- Architecture

- Workloads

Architecture: Single Client

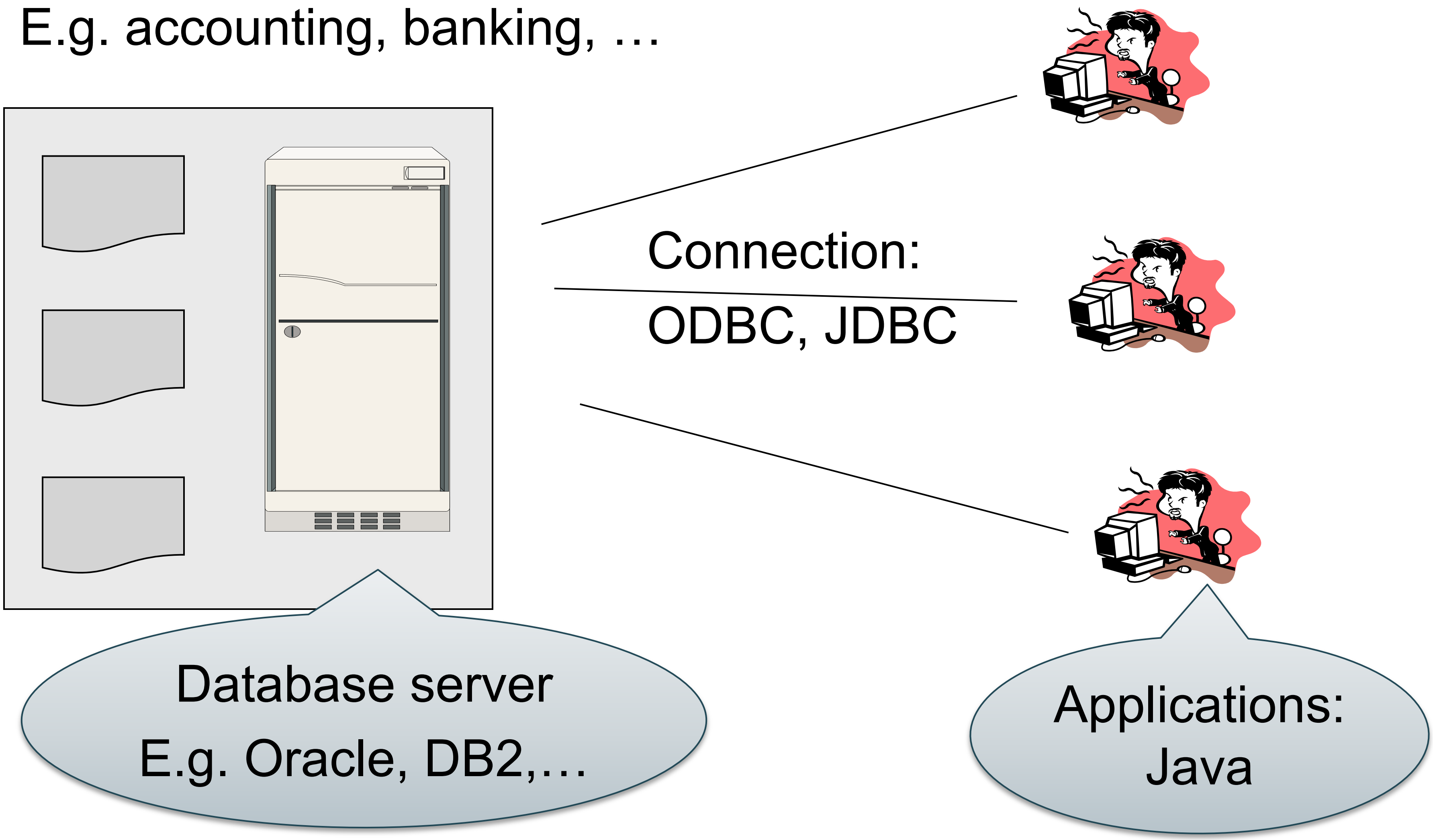
E.g. data analytics

Application and database on the same computer

E.g. sqlite, postgres

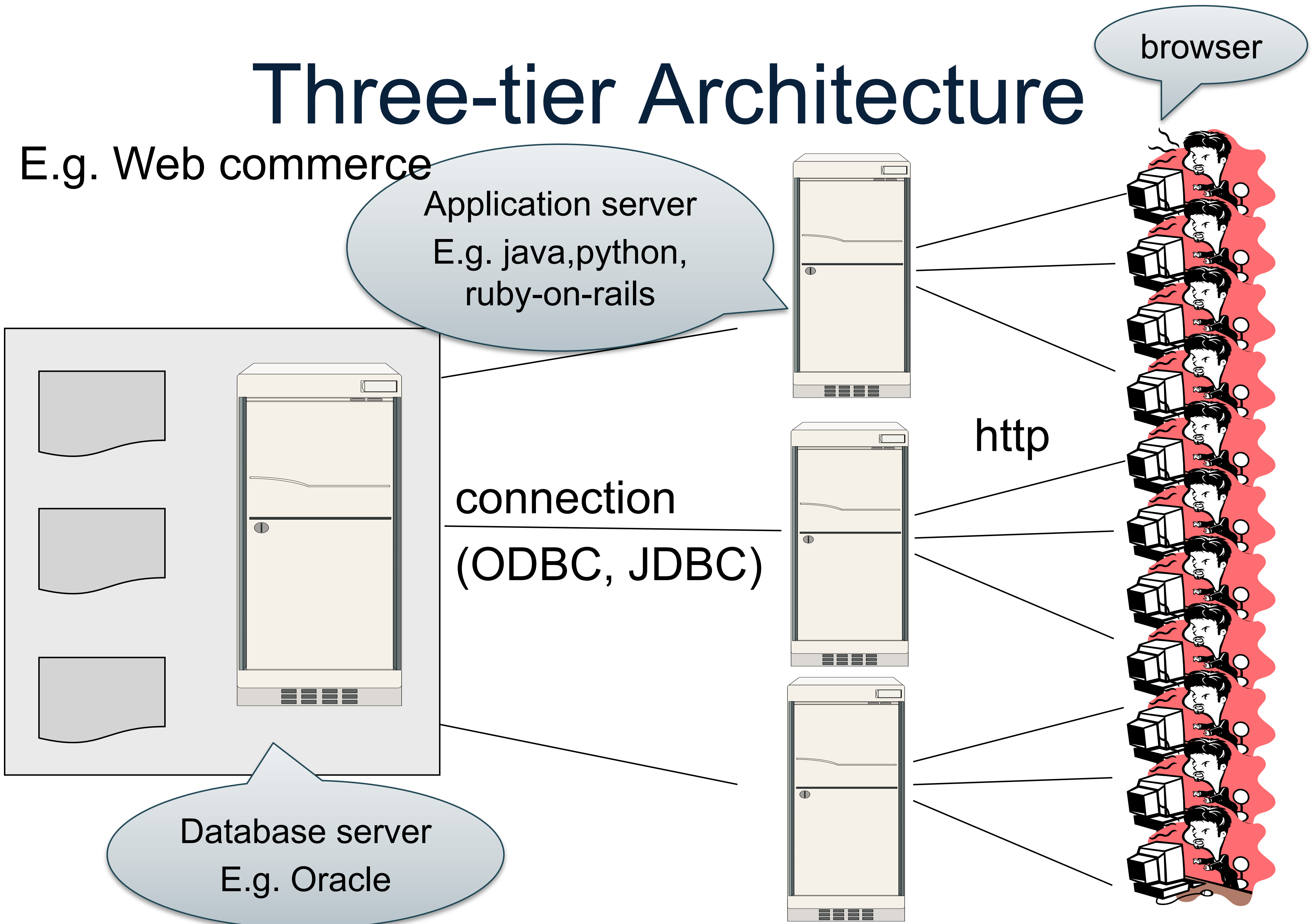
Two-tier Architecture
Client-Server

E.g. accounting, banking, …

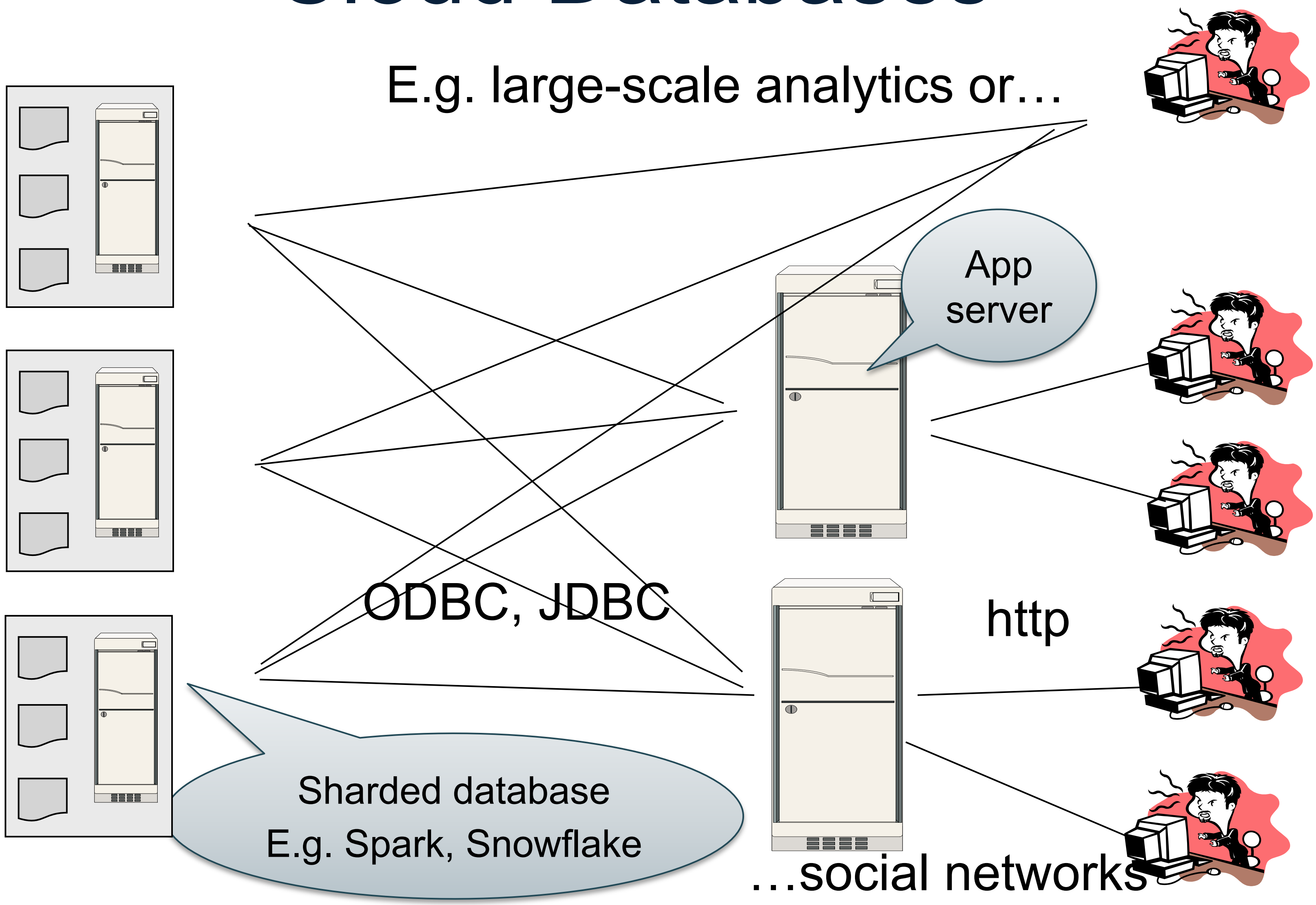


Three-tier Architecture

E.g. Web commerce



Cloud Databases



Types of Workloads

- OLTP – online transaction processing
  - – This is how data is generated
  - – Single point read/update
  - – Concurrency, transactions
- OLAP – online analytics processing
  - – Complex queries w/ aggregages
  - – Data often comes from OLTP
  - – Updates are very rare

This course

Summary

- DBMS: store and manage your data

- Everywhere:
  - On your phone: sqlite
  - On your laptop: sqlite, postgres, duckdb,…
  - In the cloud: snowflake/redshift/bigquery

- All use the Relational Data Model

Relational Data Model

Data Model

Mathematical formalism that models data.

Several exist:

- Relational
- Key/value / Document / XML / Json
- Graph
- Arrays / matrices / tensors
- Hierarchical, network…

Data Model

Mathematical formalism that models data.

Several exist:

- Relational
- Key/value / Document / XML / Json
- Graph
- Arrays / matrices / tensors
- Hierarchical, network…

Most DBMS

This class

Data Model

Mathematical formalism that models data.

Several exist:

- Relational
- Key/value / Document / XML / Json
- Graph
- Arrays / matrices / tensors
- Hierarchical, network…

NoSQL

Data Model

Mathematical formalism that models data.

Several exist:

- Relational
- Key/value / Document / XML / Json
- Graph
- Arrays / matrices / tensors
- Hierarchical, network…

Specialized

Data Model

Mathematical formalism that models data.

Several exist:

- Relational
- Key/value / Document / XML / Json
- Graph
- Arrays / matrices / tensors
- Hierarchical, network…

Machine Learning

Data Model

Mathematical formalism that models data.

Several exist:

- Relational
- Key/value / Document / XML / Json
- Graph
- Arrays / matrices / tensors
- Hierarchical, network…

Legacy…

Relational Data Model

- Database: collection of relations
- Relation (aka Table): set of tuples
- Tuple (aka row, record): $\mathrm{t} \in \mathrm{Dom}_1 \times \cdots \times \mathrm{Dom}_\mathrm{n}$

Relational Data Model

Product(name, price, color)

name	**price**	**color**
iPhone	599	Gray
iPhone	999	Black
Gizmo	399	Blue
Pizza	29	red

Relational Data Model

Product(name, price, color)

name	**price**	**color**
iPhone	599	Gray
iPhone	999	Black
Gizmo	399	Blue
Pizza	29	red

Supplier(name, product, city)

name	**product**	**city**
ACME	iPhone	Seattle
Walmart	iPhone	Renton
Costco	Pizza	Seattle

Discussion

Data independence!

- Rows in a relation:
  - Ordering immaterial (a relation is a set)
  - All rows are distinct – sets
  - Query answers may have duplicates – bags
  - Relation is flat: no lists, collections, arrays in a relation
- Attributes of a record:
  - Ordering is immaterial (mostly…)
  - Applications refer to columns by their names
- Domain of each column is a primitive type

Instance v.s. Schema

- Relation schema:
  - Relation name
  - Name of each field/column/attribute
  - Domain/type of each field

- Relational instance:
  - A set of records

Primary Key

- A key is an attribute, or a set of attributes whose value uniquely identify the record

- There may be more than one: we choose one that we call primary key

Primary Key

Product(name, price, color)

name	**price**	**color**
iPhone	599	Gray
iPhone	999	Black
Gizmo	399	Blue
Pizza	29	red

Supplier(name, product, city)

name	**product**	**city**
ACME	iPhone	Seattle
Walmart	iPhone	Renton
Costco	Pizza	Seattle

Primary key = name

Primary Key

Product(name, price, color)

name	**price**	**color**
iPhone	599	Gray
iPhone	999	Black
Gizmo	399	Blue
Pizza	29	red

Primary key = ???

Supplier(<u>name</u>, product, city)

<u>name</u>	**product**	**city**
ACME	iPhone	Seattle
Walmart	iPhone	Renton
Costco	Pizza	Seattle

Primary key = name

Primary Key

Product(name, price, color)

name	**price**	**color**
iPhone	599	Gray
iPhone	999	Black
Gizmo	399	Blue
Pizza	29	red

Primary key = name, price color

Supplier(<u>name</u>, product, city)

<u>name</u>	**product**	**city**
ACME	iPhone	Seattle
Walmart	iPhone	Renton
Costco	Pizza	Seattle

Primary key = name

Primary Key

Product(pid, name, price, color)

pid	name	price	color
p001	iPhone	599	Gray
p002	iPhone	999	Black
p003	Gizmo	399	Blue
p004	Pizza	29	red

Primary key = pid

Supplier(name, product, city)

name	product	city
ACME	iPhone	Seattle
Walmart	iPhone	Renton
Costco	Pizza	Seattle

Primary key = name

Good practice to have a single attribute as primary key

Foreign Key

- An attribute whose values are keys of another relation is called a foreign key
- Also called “semantic pointer”

Foreign Key

Product(pid, name, price, color)

pid	**name**	**price**	**color**
p001	iPhone	599	Gray
p002	iPhone	999	Black
p003	Gizmo	399	Blue
p004	Pizza	29	red

Supplier(name, product, city)

name	**product**	**city**
ACME	iPhone	Seattle
Walmart	iPhone	Renton
Costco	Pizza	Seattle

Product is ambiguous. Why?

Foreign Key

Product(<u>pid</u>, name, price, color)

<u>pid</u>	name	price	color
p001	iPhone	599	Gray
p002	iPhone	999	Black
p003	Gizmo	399	Blue
p004	Pizza	29	red

Supplier(<u>name</u>, pid, city)

<u>name</u>	pid	city
ACME	p002	Seattle
Walmart	p002	Renton
Costco	p004	Seattle

pid is a foreign key

Foreign Key

Product(pid, name, price, color)

pid	name	price	color
p001	iPhone	599	Gray
p002	iPhone	999	Black
p003	Gizmo	399	Blue
p004	Pizza	29	red

Supplier(name, pid, city)

name	pid	city
ACME	p002	Seattle
Walmart	p002	Renton
Costco	p004	Seattle

pid is a foreign key

What if Walmart sells both iPhones?

Relational Data Model

Product(pid, name, price, color)

pid	**name**	**price**	**color**
p001	iPhone	599	Gray
p002	iPhone	999	Black
p003	Gizmo	399	Blue
p004	Pizza	29	red

Supplier(name, city)

name	**city**
ACME	Seattle
Walmart	Renton
Costco	Seattle

Supply(pid, name)

pid	**name**
p002	ACME
p001	Walmart
p002	Walmart
p004	Costco

Summary

Relational data model:

- Data is stored in flat relations
- No prescription of the physical storage
- Access to the data through high-level declarative language:
  - SQL (next lecture)
  - Relational Algebra