CSE544
Data Management

Lecture 1: Introduction, Relational Data Model

Course Staff

- Instructor: Dan Suciu
  - Office hours: Mondays, 11:30-12:20
- TA: Moe Kayali
  - Office hours: Wednesdays, 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
- Miscellaneous

A Note for Non-Majors

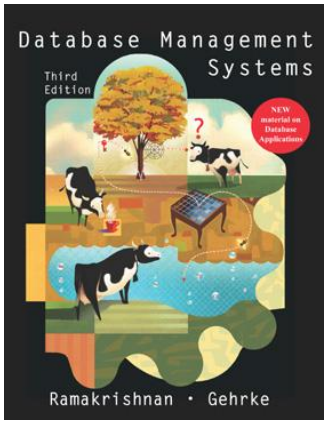
- For the Data Science option: take 414
- For the Advanced Data Science option: take 544

- 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
- Homework assignments

Canvas:

- Zoom
- Grades

Ed: discussion board

Other Resources

- Fall 2025: 590q Reading Seminar
- Winter 2026: 599ds Query Optimization
- Database course at CMU
  - Low level: storage, transactions
- Database theory course at Berkeley
  - Theory: complexity, information theory

Evaluation

- Assignments 50%
- Reviews 20%
- Project 30%

Assignments – 50%

- **HW1**: Local DBMS (postgres) posted!
- **HW2**: Cloud-based DBMS (snowflake)
- **HW3**: Query Execution and SimpleDB
- **HW4**: Datalog
- **HW5**: Theory

Late assignments w/ ***very*** valid excuse

Paper reviews – 20%

- Recommended length: ½ page – 1 page
  - Short, critical discussion of the paper
  - See guidelines on the Web
- Submit by 9:30am on the due date
  - Use the google form to submit
- Credit / partial-credit / no-credit

- R1 due on Monday, October 6

Project – 30%

Teams of 1-3 students

Topics:

- Come up with your own or choose from our list
- Best: related to your research
- Must be about databases / data management

Project – 30%

Dates posted on the calendar page:

- **P0**: Form groups
- **P1**: Project proposal
- **P2**: Project milestone
- **P3**: Poster presentation Wed. December 3rd
- **P4**: Final project report

Gitlab

- gitlab.cs.washington.edu
- Use it to submit homework assignments and project reports
- See instructions on the website:
  - Under Homework assignments
  - Set up your git repo today

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, BigQuery
  - 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
- Other issues: concurrency, durability

Let's discuss them one by one

Flat Files: Data Integrity

Give examples of data integrity conditions

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 Files: Data Integrity

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

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: Other Issues

- Concurrency: what if multiple applications touch the data?

- Durability: 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 or Embedded DB

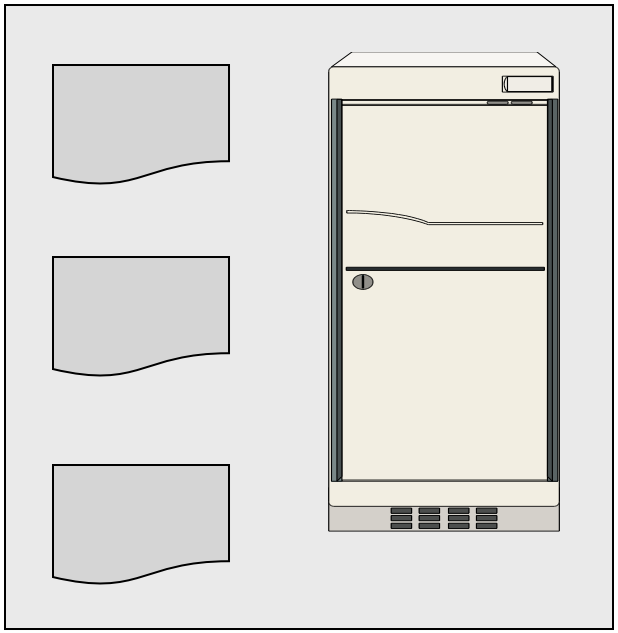
E.g. data analytics

App and DB on same computer

E.g. sqlite, postgres, duckdb

Two-tier or Client-Server

E.g. accounting, banking, …



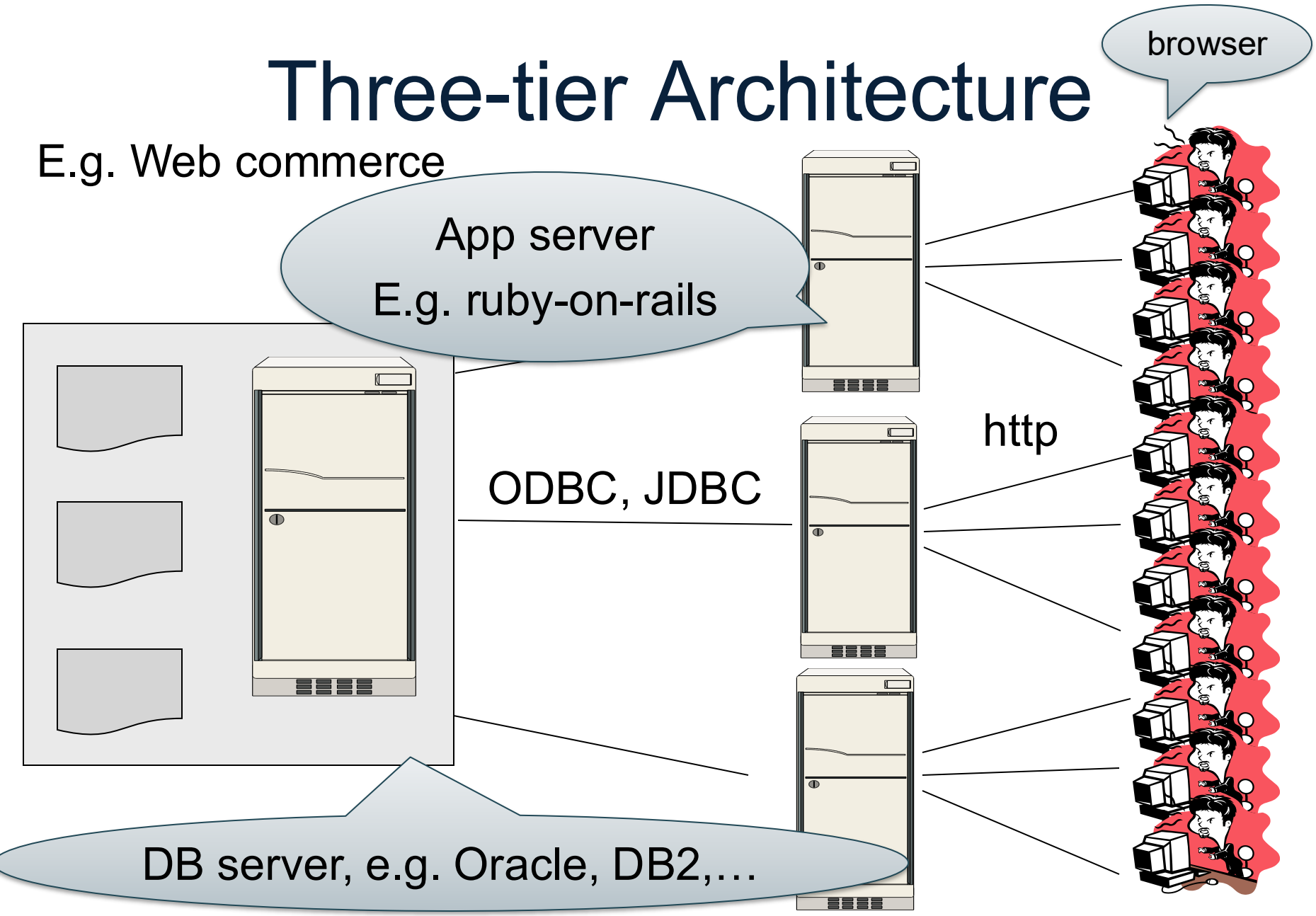
ODBC, JDBC

DB server, e.g. Oracle, DB2,…

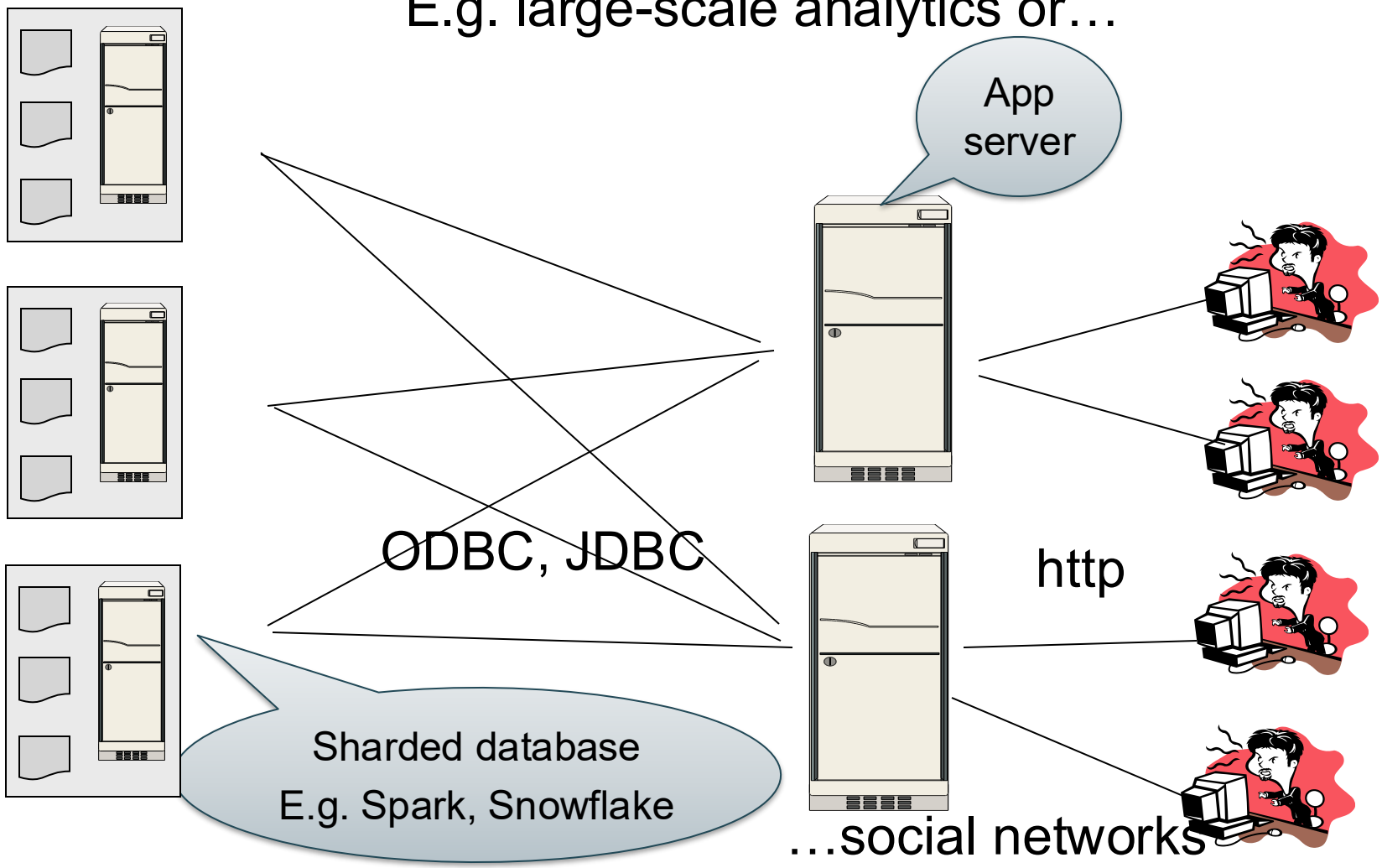
APP eg Java,…

Three-tier Architecture

E.g. Web commerce



Cloud Databases



Types of Workloads

- OLTP – online transaction processing
  - Point queries, read/write/update
  - Concurrency, transactions

- OLAP – online analytics processing
  - Complex queries, data analytics

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

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

Data Model

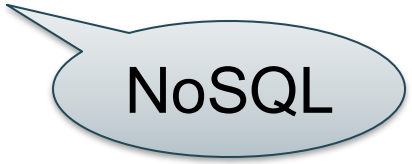
Mathematical formalism that models data

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

Data Model

Mathematical formalism that models data

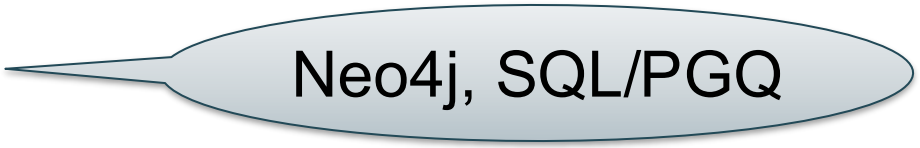
- Relational
- Key/value / Document / XML / Json
- Graph
- Arrays / matrices / tensors
- Hierarchical, network…



Data Model

Mathematical formalism that models data

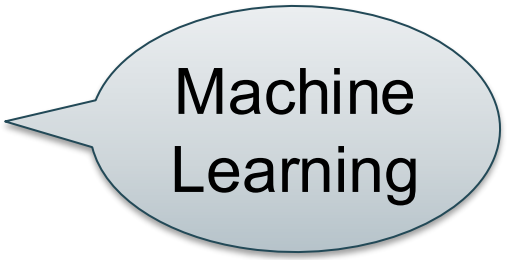
- Relational
- Key/value / Document / XML / Json
- Graph
- Arrays / matrices / tensors
- Hierarchical, network…



Data Model

Mathematical formalism that models data

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



Data Model

Mathematical formalism that models data

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

Relational Data Model

- Database: collection of relations

$$R_1, R_2, \ldots, R_m$$

- Relation (aka Table): set of tuples

$$R = \{t_1, t_2, \ldots, t_n\}$$

- Tuple (aka row, record):

$$\mathrm{t} \in \mathrm{Dom}_1 \times \cdots \times \mathrm{Dom}_\mathrm{k}$$

Relational Data Model

Product

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

Relational Data Model

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

Relational Data Model

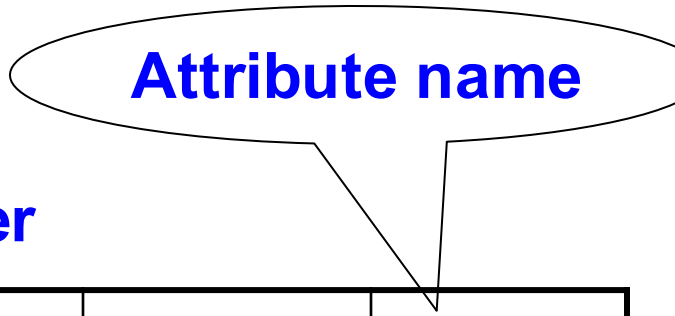
Product

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

Relational Data Model



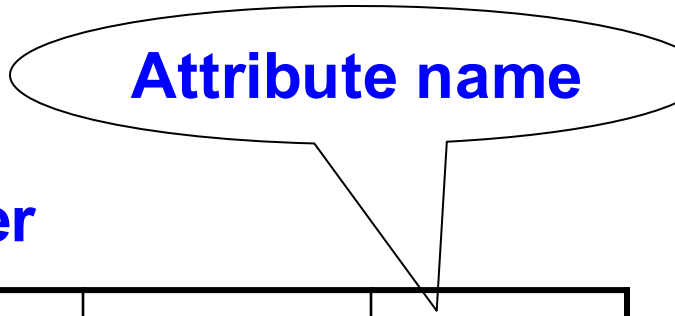
Product

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

Relational Data Model

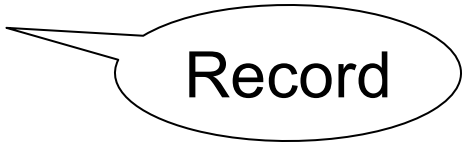


Product

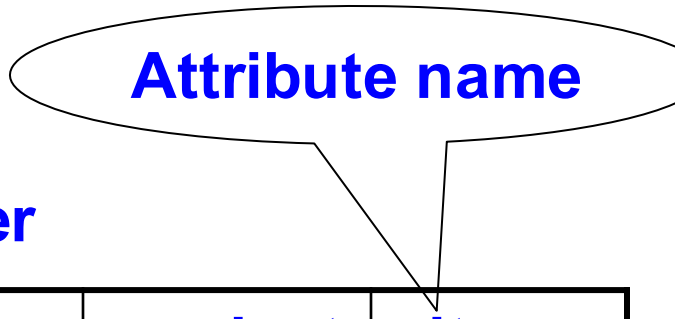
name	price	color
iPhone	599	Gray
iPhone	999	Black
Gizmo	399	Blue
Pizza	29	red

Supplier

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

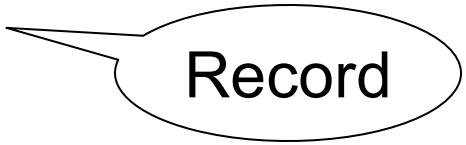


Relational Data Model



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

Database schema v.s. Database Instance

Discussion

- Rows in a relation:
  - Ordering immaterial (a relation is a set)
  - All rows are distinct – sets
  - Sometimes we allow for duplicates – bags
  - Relation is flat: no lists, collections, arrays in a relation

Discussion

- Rows in a relation:
  - Ordering immaterial (a relation is a set)
  - All rows are distinct – sets
  - Sometimes we allow for 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

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, color ?

Supplier(name, product, city)

name	**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.

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

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?

Many-many Relationship

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 via declarative language: SQL

Next Lectures: SQL

What you should do: Gitlab setup, start reading HW1