Machine Learning Systems Design

Data Lifecycle

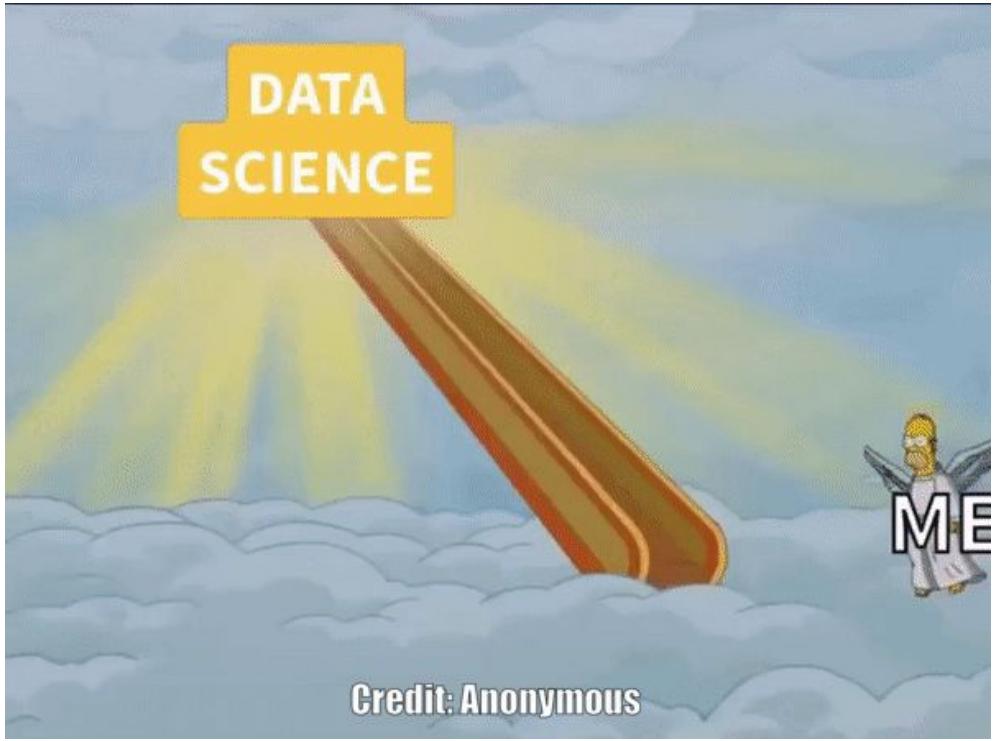
Lecture 4: Data Engineering Fundamentals

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io

Agenda

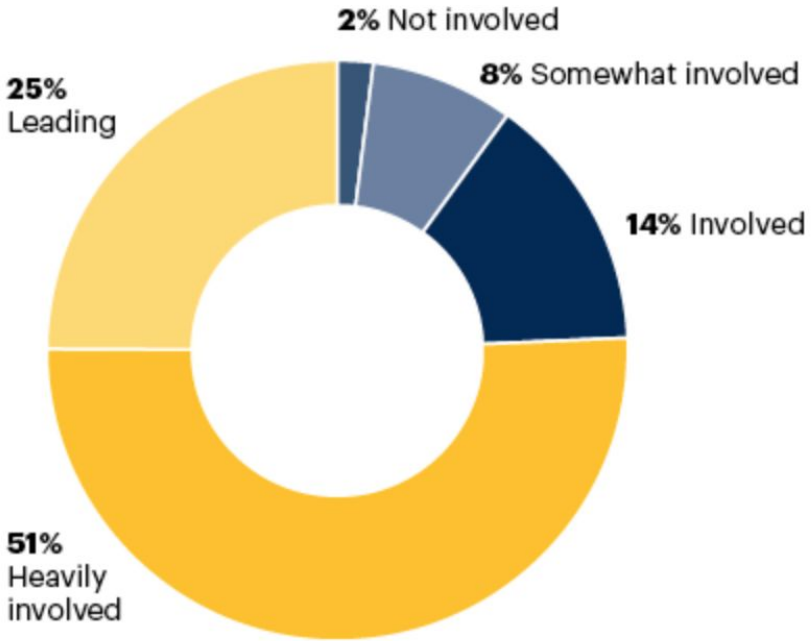
1. What is data engineering?
2. Data sources
3. Data models

Data engineering vs data science



1. What is data engineering?

Leadership of CDOs in digital transformation



Data engineering described

"Data engineering is a set of operations aimed at creating interfaces and mechanisms for the flow and access of information. It takes dedicated specialists—data engineers— to maintain data so that it remains available and usable by others. In short, data engineers set up and ***operate*** *the organization's* ***data infrastructure****,* ***preparing*** *it* ***for*** *further* ***analysis*** *by data analysts and scientists."*

— From "Data Engineering and Its Main Concepts" by *AlexSoft*

Data engineering described

*"The first type of data engineering is **SQL-focused**. The work and primary storage of the data is in relational databases. All of the data processing is done with SQL or a SQL-based language. Sometimes, this data processing is done with an ETL tool. The second type of data engineering is **Big Data–focused**. The work and primary storage of the data is in Big Data technologies like Hadoop, Cassandra, and HBase. All of the data processing is done in Big Data frameworks like MapReduce, Spark, and Flink. While SQL is used, the primary processing is done with programming languages like Java, Scala, and Python."*

— From "The Two Types of Data Engineering" by *Jesse Anderson*

Data engineering described

*"In relation to previously existing roles, the data engineering field could be thought of as a **superset of business intelligence and data warehousing** that **brings** more **elements** from **software engineering**. This discipline also integrates specialization around the operation of so-called "**big data**" distributed systems, along with concepts around the extended Hadoop ecosystem, stream processing, and in computation at **scale**."*

— From "The Rise of the Data Engineer" by *Maxime Beauchemin*

Data engineering described

*"Data engineering is all about the movement, manipulation, and **management of data**."*

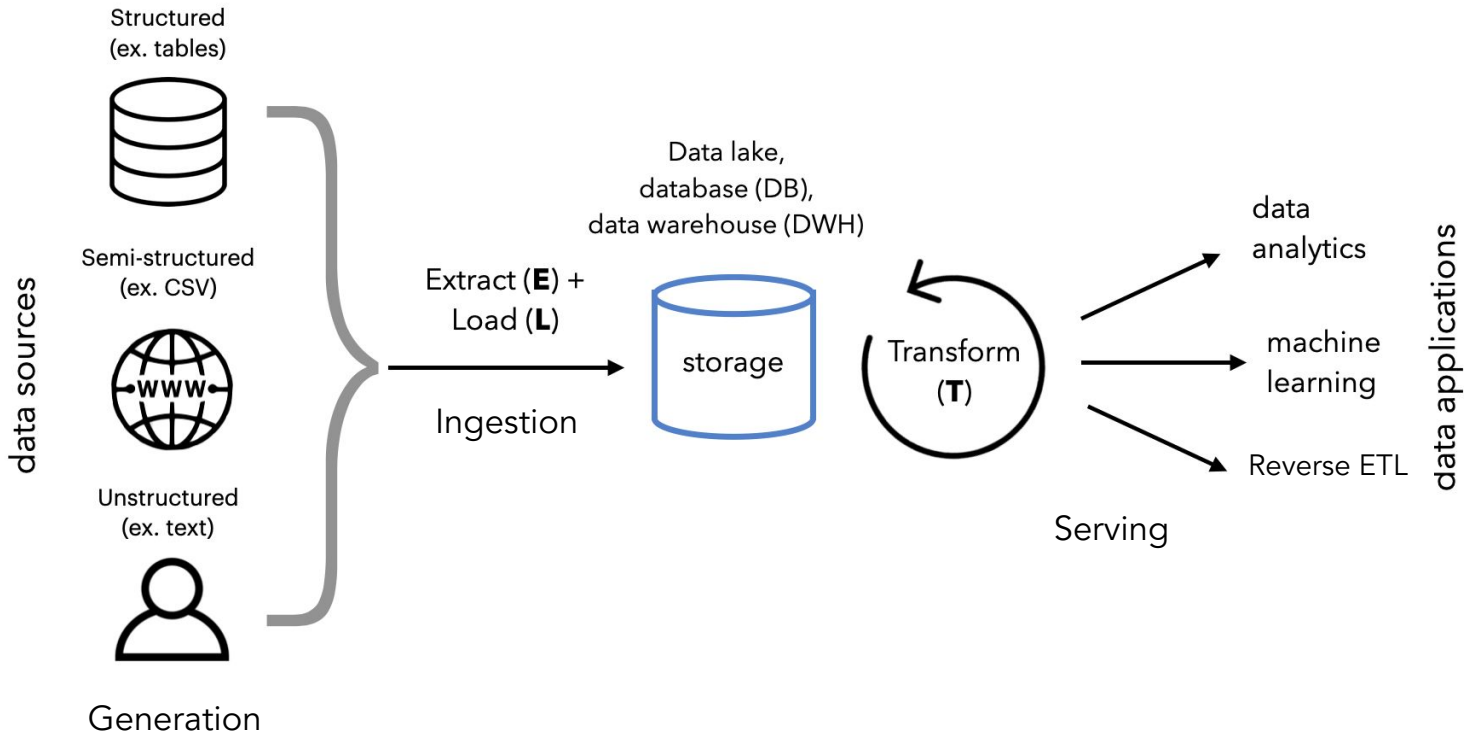
— From "What Is Data Engineering?" by *Lewis Gavin*

Data engineering described

*"Data engineering is the process of **preparing**, transforming, and modeling **data** to make it **suitable for** use in **analytics** and other applications. This typically involves tasks such as **data extraction**, data **cleaning**, data **integration**, data **warehousing**, and data **quality management**. Data engineers work to ensure that data is accurate, accessible, and reliable, and that it can be easily integrated with other systems and technologies. They also develop and maintain the infrastructure and tools needed to manage and process data on a large scale."*

— by **ChatGPT**!

Data engineering lifecycle




ETL
LIGHTYEAR
ETL EVERYWHERE

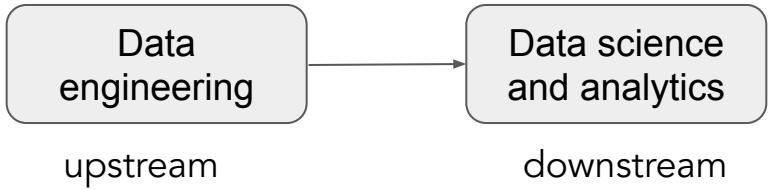
Data engineering lifecycle

- **Data acquisition**: Collecting data from various sources, such as databases, APIs, and file systems.
- **Data integration**: Combining data from different sources and formats into a single, unified dataset.
- **Data transformation**: Cleaning, normalizing, and transforming data to make it consistent and suitable for analysis.
- **Data warehousing**: Storing data in a centralized, optimized location for efficient querying and analysis.
- **Data deployment**: Making data available for use by other systems, applications, and users.

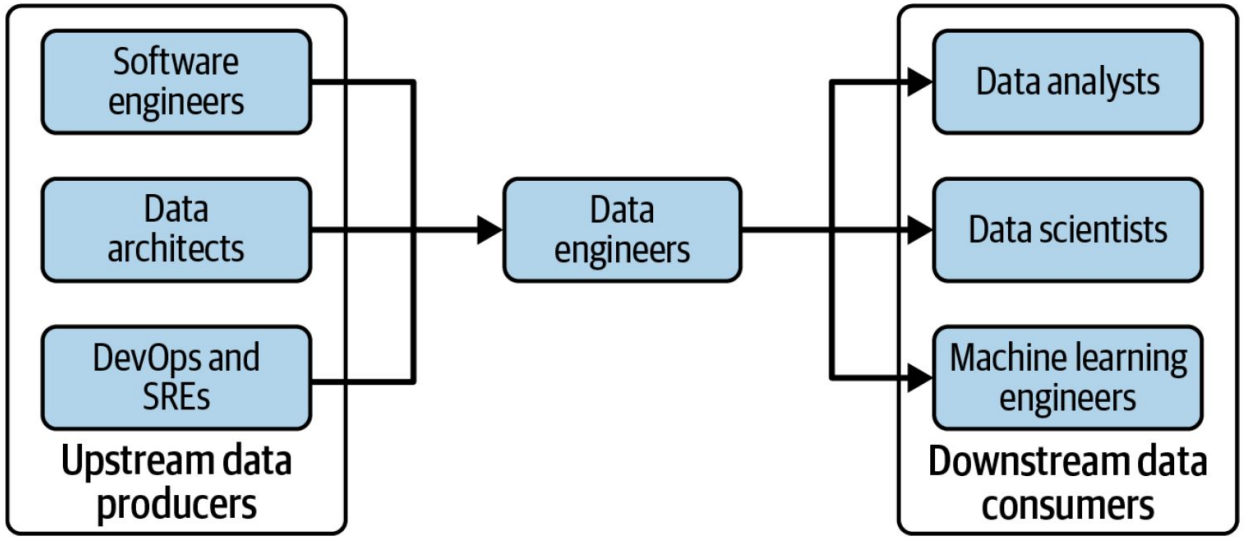
Data engineering lifecycle

- **Data governance**: Managing and maintaining data quality, security, and compliance.
- **Data monitoring**: Tracking and analyzing data usage, performance, and trends to identify issues and opportunities for improvement.
- **Data maintenance**: Keeping data up-to-date, accurate and accessible.

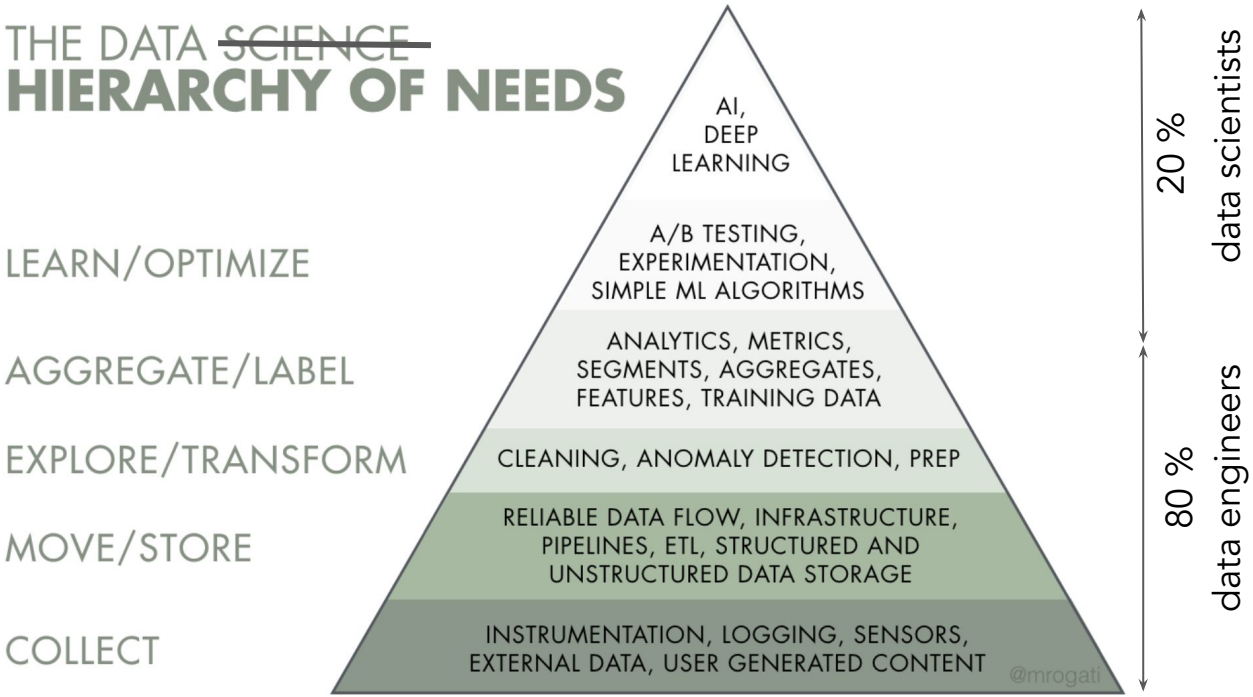
Data engineers vs data scientist



Data engineers vs data scientist



Data engineers vs data scientist



2. Data sources

How is data created?

- User generated
- Systems generated
- Internal databases: users, inventory, customer relationships
- Third-party data

Data sources

Users generated data	Systems generated data
User inputs	Logs, metadata, predictions
Easily mal-formatted	Easier to standardize
Need to be processed ASAP	OK to process periodically (unless to detect problems ASAP)
	Can grow very large very quickly ● Many tools to process & analyze logs: Logstash, DataDog, Logz, etc. ● OK to delete when no longer useful

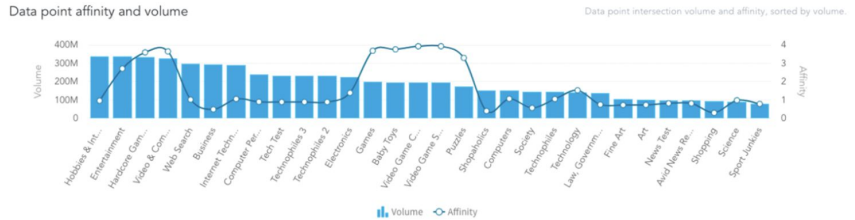
Users' behavioral data (clicks, time spent, etc.) is often system-generated but is considered **user data**

Third-party data: creepy but fascinating

- Types of data
  - social media, income, job
- Demographic group
  - men, age 25-34, work in tech
- More available with Mobile Advertiser ID
- Useful for learning features
  - people who like A also like B

Top interests

They love computing and electronic entertainment. If you want to reach players, try targeting at their top interests.



61 M profiles

Remote working

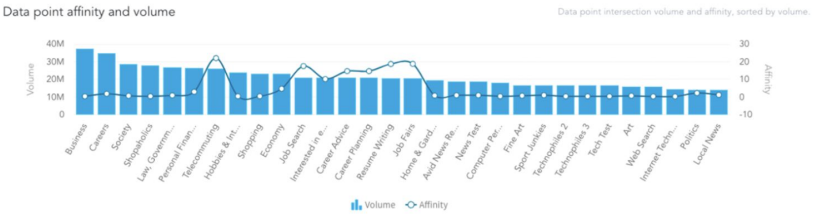
Millions of people decided to #stayhome and work remotely to limit the spread of coronavirus. Use our Remote working segment to easily reach them and show software or products that will help them stay effective.

How did we build the segment?

Our segment includes profiles of users who recently read articles, watched videos or used mobile apps which refers to:

- remote working
- effective ways of working from home
- tools for remote workers
- homeschooling and e-learning

If you want to reach remote workers, try to extend your target group by selecting the top interests, which include Telecommuting, Career Planing or Personal Finance.



Data sources

- Files and unstructured data
  - images, audio and video
  - Excel, CSV, TXT, JSON, and XML
  - Parquet, ORC, and Avro
- Application databases
- Messages and Streams
- APIs
- Logs of applications and databases

4. Data models

Data models

- Describe how data is represented
- Two main paradigms:
  - Relational model
  - NoSQL
    - Document model
    - Graph model
    - Search
    - etc

Relational model

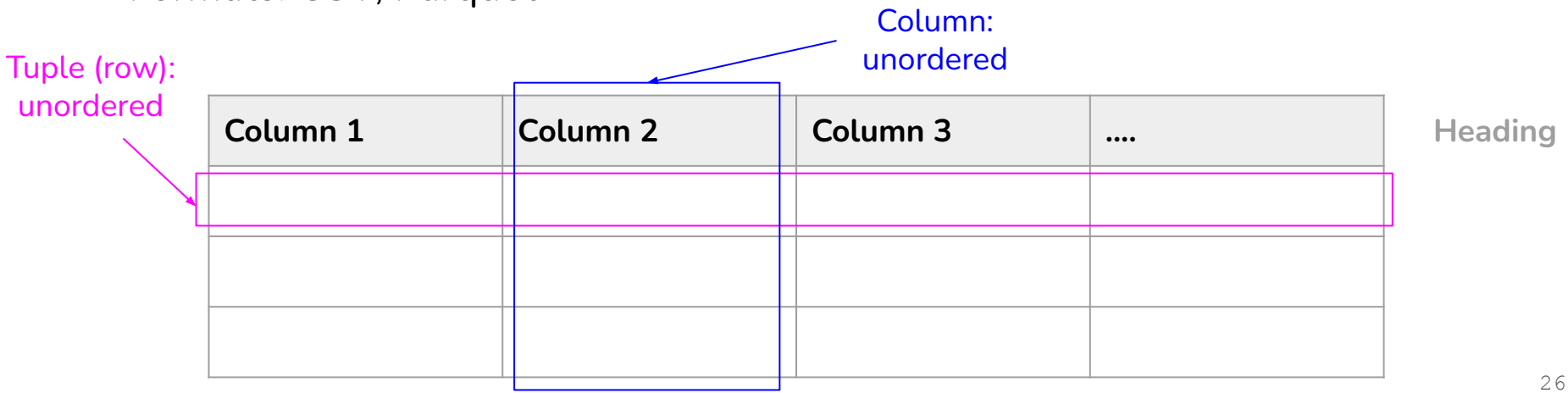
The roots of relational databases lie in *business data processing*:

- ***transaction*** *processing* (entering sales or banking transactions, airline reservations, stock-keeping in warehouses)
- ***batch*** *processing* (customer invoicing, payroll, reporting)

but, turned out to generalize very well, beyond their original scope of business data processing, to a broad variety of use cases.

Relational model

- Proposed by Edgar Codd in 1970
- Data is organized into relations (called tables in SQL), where each relation is an unordered collection of tuples (rows in SQL)
- Similar to SQL model
- Formats: CSV, Parquet



Relational model: normalization

What if we change “Banana Press” to “Pineapple Press”?

Title	Author	Format	Publisher	Country	Price
Harry Potter	J.K. Rowling	Paperback	Banana Press	UK	$20
Harry Potter	J.K. Rowling	E-book	Banana Press	UK	$10
Sherlock Holmes	Conan Doyle	Paperback	Guava Press	US	$30
The Hobbit	J.R.R. Tolkien	Paperback	Banana Press	US	$30
Sherlock Holmes	Conan Doyle	Paperback	Guava Press	US	$15

Original Book Relation

Relational model: normalization

Title	Author	Format	Publisher ID	Price
Harry Potter	J.K. Rowling	Paperback	1	$20
Harry Potter	J.K. Rowling	E-book	1	$10
Sherlock Holmes	Conan Doyle	Paperback	2	$30
The Hobbit	J.R.R. Tolkien	Paperback	1	$30
Sherlock Holmes	Conan Doyle	Paperback	2	$15

Updated Book Relation

Publisher ID	Publisher	Country
1	Banana Press	UK
2	Guava Press	US

Publisher Relation

Relational model: normalization

Title	Author	Format	Publisher ID	Price
Harry Potter	J.K. Rowling	Paperback	1	$20
Harry Potter	J.K. Rowling	E-book	1	$10
Sherlock Holmes	Conan Doyle	Paperback	2	$30
The Hobbit	J.R.R. Tolkien	Paperback	1	$30
Sherlock Holmes	Conan Doyle	Paperback	2	$15

Publisher ID	Publisher	Country
1	Banana Press	UK
2	Guava Press	US

Pros:

- **Less mistakes (standardized spelling)**
- **Easier to update**
- **Easier localization**
- **Better search**
- **Avoids deduplication**

Cons:

- **Slow to join across multiple large tables**

Relational Model & SQL Model

- SQL model slightly differs from relational model
  - e.g. SQL tables can contain row duplicates. True relations can’t.
- SQL is a query language
  - How to specify the data that you want from a database
- SQL is declarative
  - You tell the data system what you want
  - It’s up to the system to figure out how to execute
    - Query optimization

SQL

- SQL is an essential data scientists’ tool, it's the language of a data scientist.

LEARN SQL!

Problems with SQL

- What if we add a new column?
- What if we change a column type?
- What if tabel size increase?

The Birth of NoSQL

In the 2010s, NoSQL is the latest attempt to overthrow the relational model's dominance.

NoSQL = Not only SQL

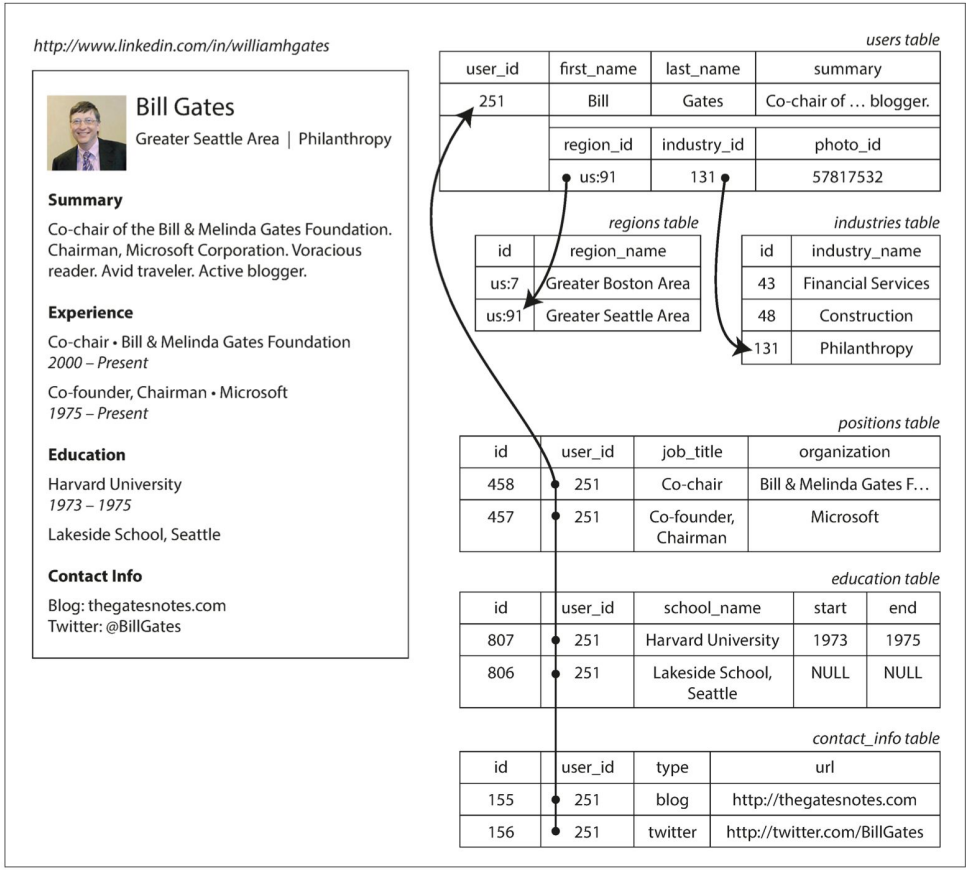
The Birth of NoSQL

Driving forces behind the adoption of NoSQL:

- Greater scalability
- Flexible data model
- Open source preference
- Sometime closer to application data structure

different applications have different requirements

Example: resume data model



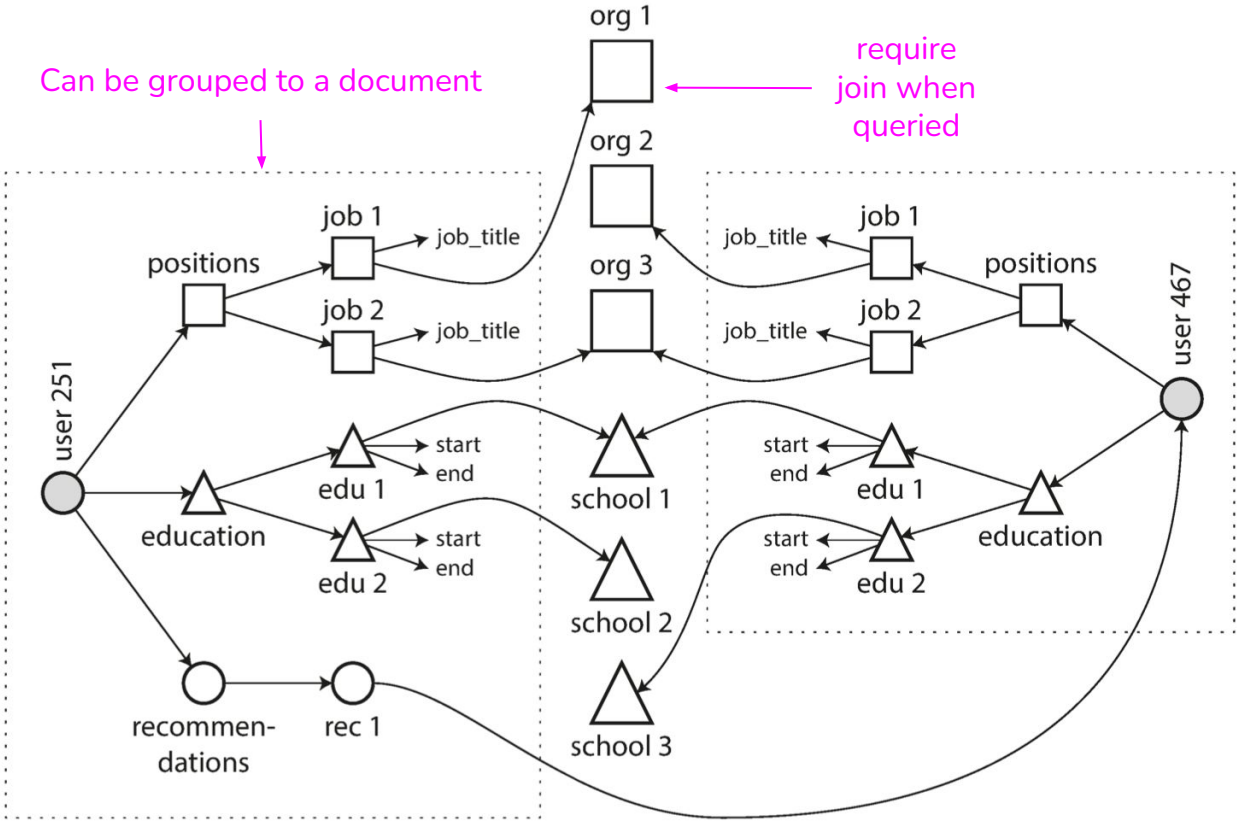
Object-Relational Mismatch

Example: resume data model

JSON representation has better locality

```
{
  "user_id":     251,
  "first_name":  "Bill",
  "last_name":   "Gates",
  "summary":     "Co-chair of the Bill & Melinda Gates... Active blogger.",
  "region_id":   "us:91",
  "industry_id": 131,
  "photo_url":   "/p/7/000/253/05b/308dd6e.jpg",
  "positions": [
    {"job_title": "Co-chair", "organization": "Bill & Melinda Gates Foundation"},
    {"job_title": "Co-founder, Chairman", "organization": "Microsoft"}
  ],
  "education": [
    {"school_name": "Harvard University",       "start": 1973, "end": 1975},
    {"school_name": "Lakeside School, Seattle", "start": null, "end": null}
  ],
  "contact_info": {
    "blog":    "http://thegatesnotes.com",
    "twitter": "http://twitter.com/BillGates"
  }
}
```

Example: resume data model



Data has a tendency of becoming more interconnected as features are added to applications:

- Organizations and schools as entities
- Recommendations

NoSQL data models

- Document databases (MongoDB)
- Key-value stores (Redis)
- Column-oriented databases (Cassandra)
- Graph databases (Neo4J)
- Other
  - Search databases (Elastic)
  - Time series databases (InfluxDB)

Document model: example

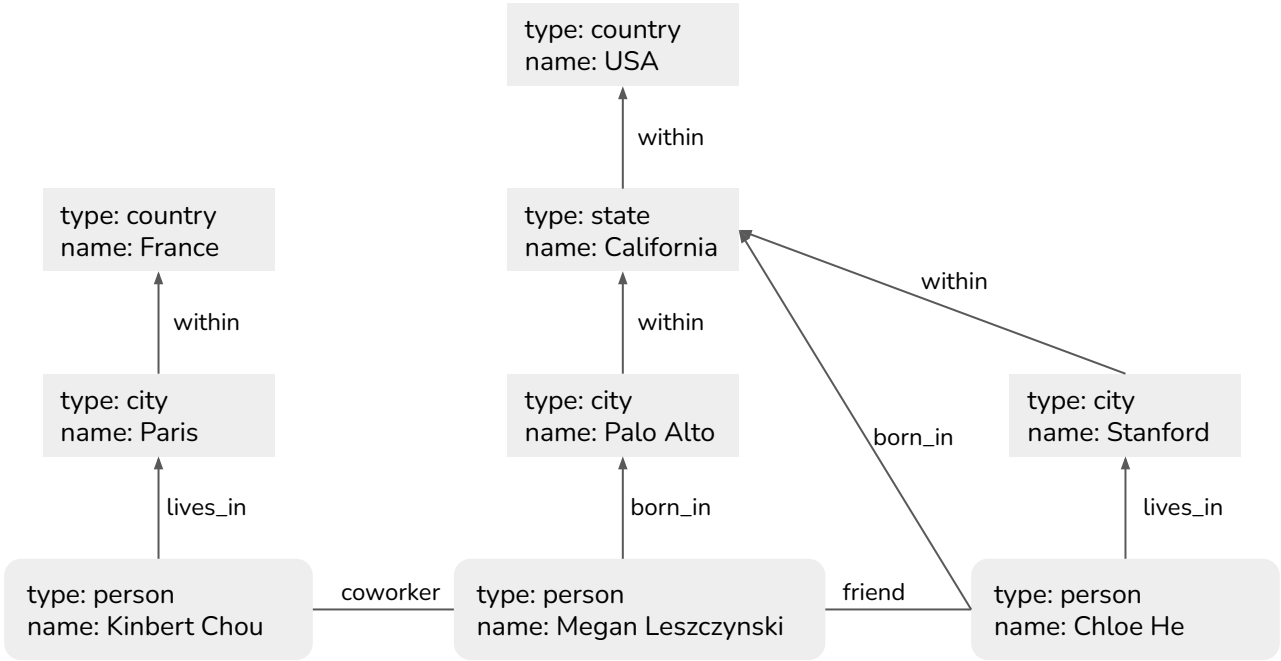
- Book data in the document model
- Each book is a document

```
# Document 1: harry_potter.json
{
    "Title": "Harry Potter",
    "Author": "J.K. Rowling",
    "Publisher": "Banana Press",
    "Country": "UK",
    "Sold as": [
        {"Format": "Paperback", "Price": "$20"},
        {"Format": "E-book", "Price": "$10"}
    ]
}

# Document 2: sherlock_holmes.json
{
    "Title": "Sherlock Holmes",
    "Author": "Conan Doyle",
    "Publisher": "Guava Press",
    "Country": "US",
    "Sold as": [
        {"Format": "Paperback", "Price": "$30"},
        {"Format": "E-book", "Price": "$15"}
    ]
}

# Document 3: the_hobbit.json
{
    "Title": "The Hobbit",
    "Author": "J.R.R. Tolkien",
    "Publisher": "Banana Press",
    "Country": "UK",
    "Sold as": [
        {"Format": "Paperback", "Price": "$30"},
    ]
}
```

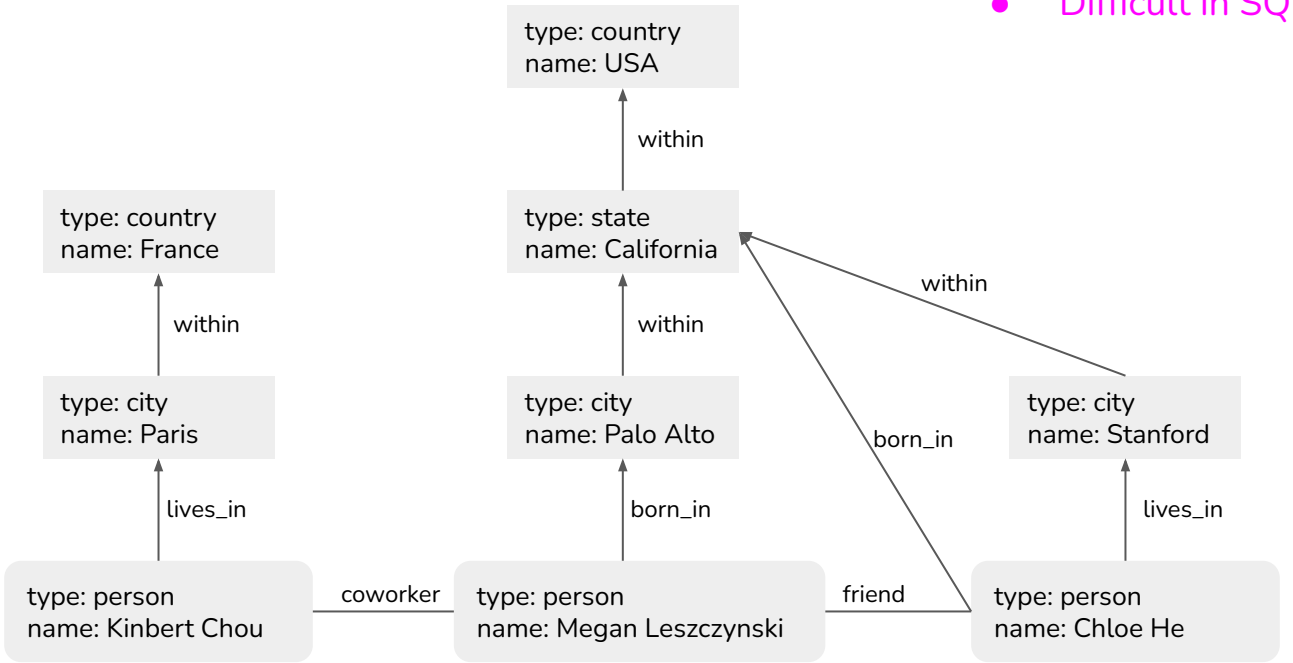
Graph model


type: country
name: USA
within
type: country
name: France
type: state
name: California
within
within
within
type: city
name: Paris
type: city
name: Palo Alto
type: city
name: Stanford
born_in
lives_in
born_in
lives_in
type: person
name: Kinbert Chou
coworker
type: person
name: Megan Leszczynski
friend
type: person
name: Chloe He

Graph model

Query: show me everyone who was born in the USA?

- Easy in graph
- Difficult in SQL



Consider SQL databases when...

- Your data is **highly structured**, and that structure doesn't change frequently
- You support **transaction-oriented systems** such as accounting or financial applications
- You require a high degree of **data integrity** and security
- You don't require the **scale-out** capabilities that NoSQL offers

Consider NoSQL databases when...

- You're working with large amounts of **unstructured** or semi-structured data that doesn't fit the relational model
- You require the **flexibility** of a dynamic **schema** or want more choice over the data model
- You require a database system that can be **scaled horizontally**, perhaps across multiple geographic locations
- You want to streamline development and avoid the **overhead** of a more structured approach
- Your applications don't require the level of **data integrity** offered by SQL databases

Query languages for data

```
function getSharks() {
    var sharks = [];
    for (var i = 0; i < animals.length; i++) {
        if (animals[i].family === "Sharks") {
            sharks.push(animals[i]);
        }
    }
    return sharks;
}
```

Usual approach to query for sharks (**imperative**)

Query Languages for Data

$$\text{sharks} = \sigma_{\text{family} = \text{“Sharks”}} (\text{animals})$$

In relational algebra

```
SELECT * FROM animals WHERE family = 'Sharks';
```

SQL introduced a new way to query data (**declarative**)

Query languages for data

In **imperative** language you tell the computer to perform certain operations in a certain order.

In **declarative** language you just specify the pattern of the data you want but not how to achieve that goal → *hides implementation details, performance improvement without change in code, parallel execution*

Query languages for data

Imagine you are a marine biologist, and you add an observation record to your database every time you see animals in the ocean:

```
{
    observationTimestamp: Date.parse("Mon, 25 Dec 1995 12:34:56 GMT"),
    family:      "Sharks",
    species:     "Carcharodon carcharias",
    numAnimals: 3
}
{
    observationTimestamp: Date.parse("Tue, 12 Dec 1995 16:17:18 GMT"),
    family:      "Sharks",
    species:     "Carcharias taurus",
    numAnimals: 4
}
```

Query languages for data

How many sharks you have sighted per month, in PostgreSQL:

```
SELECT date_trunc('month', observation_timestamp) AS observation_month,
       sum(num_animals) AS total_animals
FROM observations
WHERE family = 'Sharks'
GROUP BY observation_month;
```

truncate timestamp to nearest month

Query languages for data

in MongoDB's **MapReduce**:

```
db.observations.mapReduce(
    function map() { ❷
        var year  = this.observationTimestamp.getFullYear();
        var month = this.observationTimestamp.getMonth() + 1;
        emit(year + "-" + month, this.numAnimals); ❸
    },
    function reduce(key, values) { ❹
        return Array.sum(values); ❺
    },
    {
        query: { family: "Sharks" }, ❶
        out: "monthlySharkReport" ❻
    }
);
```

Query languages for data

MapReduce is a fairly low-level programming model for **distributed** execution on a cluster of machines.

Machine Learning Systems Design

Data Lifecycle

Next Lecture: Data Engineering Fundamentals (cont.)

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io