Entity Resolution

Susan B. Davidson and Zachary G. Ives
University of Pennsylvania
CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-31-22

Named Entities

- So far we've seen how to extract (potential) entities from text
- How do we know when they mean the same thing

https://tinyurl.com/cis545-lecture-2-2-22

Resolving Named Entities

We can use approximate string match, but it can be ambiguous or misleading:

- "Hep A" = "Hep B"
  or "Hepatitis A"?

Context and entity type help

- "Cal" = "calories"
  or "California"
  or "Univ. of California"

https://tinyurl.com/cis545-lecture-2-2-22

Entity Resolution for Certain Kinds of Data

Brand names (companies) are relatively easy

- Need to deal with abbreviations and spelling mistakes

Product models are more complex

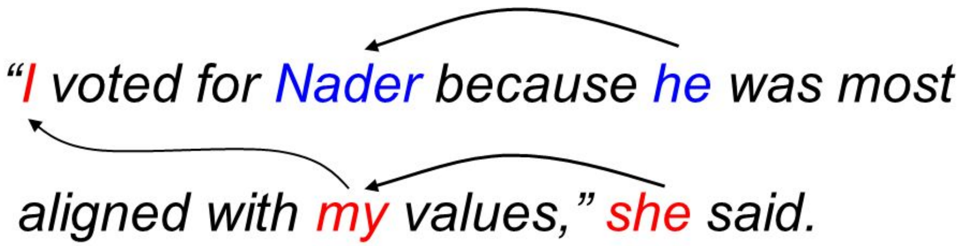
- Variations in writing styles
  - Honda Civic could be written as "Honda Civic"; "Civic"; "Honda Civic LS"; "Honda Civic LE"; "LE"; "H. Civic"; "Hondah Sivik"
  - Model numbers can be written as: 5, V, Five
  - "Asics Speedstar (both I and II), I love the I and II's and can't wait for the III's"
  - Model can be referred to as numbers but numbers do not always refer to models (e.g., "1010 for New Balance 1010", but $1010)

City names ambiguous: Cambridge, Rochester, San Jose, Portland

https://tinyurl.com/cis545-lecture-2-2-22

Exactly the "record linking" problem we saw with our Wikipedia data wrangling example!

Coreference Resolution



https://nlp.stanford.edu/projects/coref.shtml

Determining when different segments of the text are referring to the same entity

more than entity matching: pronouns, paraphrases, etc.

https://tinyurl.com/cis545-lecture-2-2-22

Coreference Resolution

Can be complicated, but relatively simple methods work OK.

- Locate all noun phrases
- Identify their properties or variations
  - singular/plural, …
- Cluster them in starting with the highest-confidence rules and moving to lower-confidence ones
  - Check first for pronominal/generic-nominal references
  - Then do closest first

Lee, Peirsman et al. 2011

https://tinyurl.com/cis545-lecture-2-2-22

Co-reference resolution example

Microsoft announced it plans to acquire Visio. The company said it will finalize its plans within a week.

Mark said that he used Symlin and it caused him to get a rash. He said that it bothered him.

https://tinyurl.com/cis545-lecture-2-2-22

Summary of Entity Resolution

- A variant of the entity matching / record linking problem, and can use many of the same techniques
- General approaches work better on some domains than others
- Coreference resolution within text is more complicated due to prepositions, paraphrases – heavily based on heuristics

https://tinyurl.com/cis545-lecture-2-2-22

Brief Review

https://canvas.upenn.edu/courses/1636888/quizzes/2771543 (05D)

- Why is approximate string match tricky to use for entity resolution?

a. abbreviations may not match closely against full words
b. text doesn't approximately match
c. string similarity is only defined for structured data

- Co-reference resolution looks at whether

a. strings are similar
b. items are appropriately cited
c. different words or phrases represent the same thing

https://tinyurl.com/cis545-lecture-2-2-22

Relation Extraction and Part I Wrap-up

Susan B. Davidson and Zachary G. Ives
University of Pennsylvania

CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-31-22

Relation Extraction

- Ultimately we want to learn more from text than the nouns
- How do they relate, can we use these to derive new facts?

https://tinyurl.com/cis545-lecture-2-2-22

Template-based IE for relation extraction

Write or learn templates to extract entities and relations between them

- X "was acquired by" Y
- X “in” Y
- <person> , .* inventor .* of Y

Open IE = Machine Reading

- Automatically learn templates for new relationships

https://tinyurl.com/cis545-lecture-2-2-22

Extract Relations

http://www.nltk.org/book/ch07.html

```
# Regular expression: . means single wildcard character,
# .* means any sequence of wildcard characters, \b = blank,
#
! means negatio
IN = re.compile
table = []
for doc in nltk
for rel in nltk
  corpus='ieer',
    simple_dict



      table.
```

WHYY	ORGANIZATION	in	Philadelphia	LOCATION
McGlashan & Sarrail	ORGANIZATION	firm in	San Mateo	LOCATION
Freedom Forum	ORGANIZATION	in	Arlington	LOCATION
Brookings Institution	ORGANIZATION	, the research group in	Washington	LOCATION

https://tinyurl.com/cis54

Relation Extraction

Use templates to extract relations

- For ***Acquisition*(*Company*, *Company*)** :
  - NP2 "was acquired by" NP1
  - NP1 "'s acquisition of" NP2
- For ***MayorOf*(*City*, *Person*)**:
  - NP ", mayor of" <city>
  - <city> "'s mayor" NP
  - <city> "mayor" NP

NP = Noun Phrase

KnowItAll (Etzioni, Cafarella et al. 2005).

Impossible to guess all the possible templates – use ML!

https://tinyurl.com/cis545-lecture-2-2-22

IE is hard

Language is complex

- Synonyms and Orthonyms
  - Bush, HEK
- User-generated text is rarely grammatical
- Complex structure
  - The first time I bought your product, I tried it on my dog, who became very unhappy and almost ate my cat, who my daughter dearly loves, and then when I tried it on her, she turned blue!

https://tinyurl.com/cis545-lecture-2-2-22

Really Effective IE is Hard

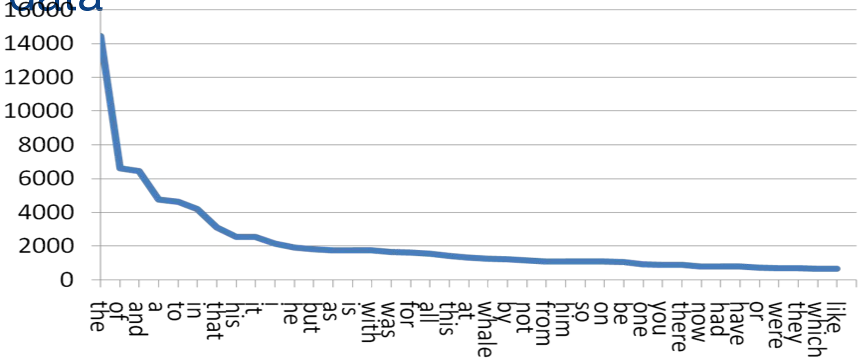
Hand-built systems give poor coverage

- Can’t manually list all patterns
- Zipf's law ensures that most words are rare

Statistical methods need training data

- Expensive to manually label data

http://searchengineland.com/the-long-tail-of-search-12198



https://tinyurl.com/cis545-lecture-2-2-22

“Adequate” IE May Be Relatively Easy

Accuracy and coverage are OK

- Typically 80% to 90% accurate
- Typically finds less than half of all mentions

Since many facts occur hundreds of times on the web, finding popular facts is easy

- Not so good if something shows up once or twice

https://tinyurl.com/cis545-lecture-2-2-22

Learning from the Web Is Tricky

Everything on the web is NOT true

… And it's very hard to use statistical methods to combine claims

Lots of ongoing research on copy detection, fact checking, claim provenance, …

https://tinyurl.com/cis545-lecture-2-2-22

Brief Review

https://canvas.upenn.edu/courses/1636888/quizzes/2771515 (05E)

- Relation extraction depends on templates to figure out

a. descriptions for how entities relate
b. how to match relational tables
c. how seed pairs relate
d. how to make text grammatical

- Information extraction depends on what to address the issue that it has low recall (misses many mentions)?

a. redundant information in the text
b. fake news
c. coregistration
d. pronouns and antecedents

https://tinyurl.com/cis545-lecture-2-2-22

Data and Concept Representation

Susan B. Davidson and Zachary G. Ives

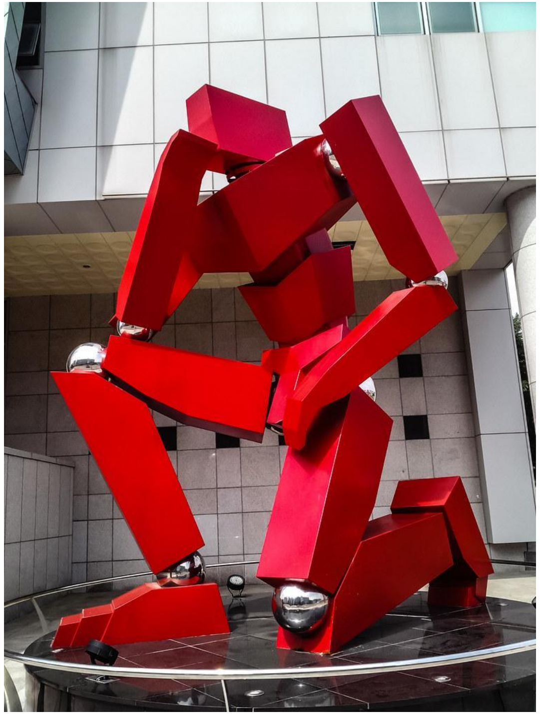
University of Pennsylvania

CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-31-22

Welcome to Part 2: Representing Data Logically & Physically



https://tinyurl.com/cis545-lecture-2-2-22

We've seen several ways of using data:

- **semi-structured** HTML and **unstructured** text, and information extraction
- **tabular data**
- operations over tabular data
  - **projection**, **selection**, **apply**
  - **merge/join**, outerjoin

Now let's dive into more detail on **designing data**:

- How to encode data? What are the implications?
- How do operations affect **performance** and **scale**?

A Key Question: How Do We Capture What We Know?

We know attributes and values – from tuples, dictionaries, objects, …

But alone, these don't capture certain other notions, e.g.:

- A **student** is a special kind of a **person** who is enrolled in classes
- **Atmospheric pressure** reduces as we go to higher altitudes
- Once you **buy** an item, you are its **owner**

Is there a general way of capturing such knowledge?

https://tinyurl.com/cis545-lecture-2-2-22

Class Membership and Relationships Expressed in Logic

Most of modeling in computer science can be described using local constraints or *predicates*

person(maya).

childOf(nan,wenqin).

We can also describe rules for *inferring* new relationships:

childOf(*x*,*y*) □ descendantOf(*x*,*y*)
childOf(*x*,*y*) ^ descendantOf(*y*,*z*) □ descendantOf(*x*,*z*)

https://tinyurl.com/cis545-lecture-2-2-22

This is part of *Knowledge Representation*

and is one aspect of AI (and databases)

https://tinyurl.com/cis545-lecture-2-2-22

Module Overview

- How might we capture knowledge, both general and specific?
  - Logical predicates and expressions
  - Classes, entities and relationships in a graph
- Encoding hierarchy and graphs in relations
- Hierarchical storage and NoSQL
- Going between trees, graphs, and relations

https://tinyurl.com/cis545-lecture-2-2-22

Representing Knowledge with Logic and Queries

Susan B. Davidson and Zachary G. Ives

University of Pennsylvania

CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-31-22

Classes Let Us *Infer* Properties and Characteristics!

The famous example from logic and philosophy, about the inventor of this approach:

- All people (men) are mortal
- Socrates is a man
- Therefore, **Socrates is mortal**

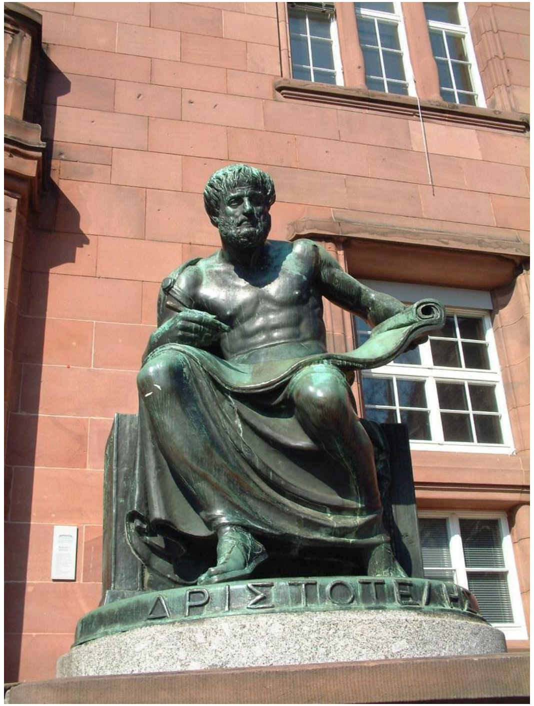
May also want to infer properties / features:

- All mortal things have dates of birth and death
- Therefore, Socrates has a birth date and death date

Note this takes general knowledge and makes inference

Data design is about trying to codify the above!

https://tinyurl.com/cis545-lecture-2-2-22



Lessons from Artificial Intelligence: Thinking about Knowledge Using Logic

Classes / concepts: named, categorized collections of items

"All people are mortal" IsMortal(person).

Classes have specializations or subclasses:

"Men are people" IsSubclass(man, person).

Classes have instances:

"Aristotle is a man" IsInstance(Aristotle, man)

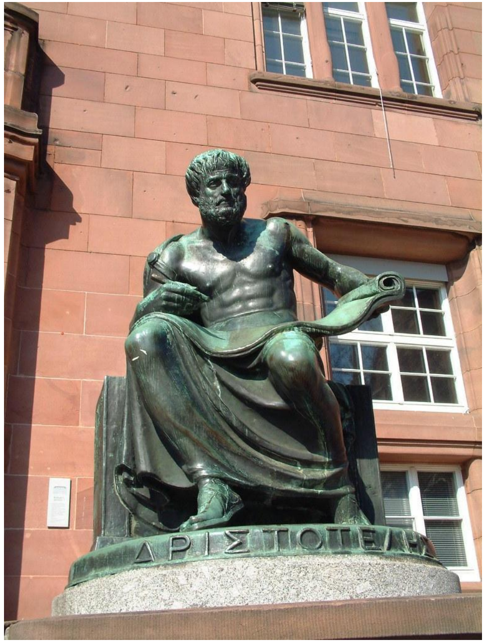
Knowledge representation has inference *rules*:

IsMortal(x) ^ IsSubclass(y, x) □ IsMortal(y)

IsMortal(x) ^ IsInstance(y, x) □ IsMortal(y)

IsMortal(person) ^ IsSubclass(man, person) □ IsMortal(man)

IsMortal(man) ^ IsInstance(Aristotle, man) □IsMortal(Aristotle)



https://tinyurl.com/cis545-lecture-2-2-22

Connecting this to SQL Via Derived Tables or *Views*

```
CREATE VIEW mortal(id) AS
SELECT id
FROM person
```

Property holds for all instances of the class

```
CREATE VIEW person(id) AS
  SELECT id
  FROM man
UNION
  SELECT id
  FROM woman
UNION …
```

Parent class is a superset of all subclasses

https://tinyurl.com/cis545-lecture-2-2-22

Brief Review

https://canvas.upenn.edu/courses/1636888/quizzes/2771487 (06B)

- To represent that Aristotle is mortal using logical constraints, we might use which predicate?
  a. IsMortal(aristotle)
  b. IsMortal -> aristotle
  c. IsMan(aristotle)
  d. aristotle -> IsMortal
- To compute all members of a superclass, we can:
  a. union together the members of all subclasses
  b. join together the members of all subclasses
  c. union together the members of all superclasses
  d. join together the members of all superclasses

https://tinyurl.com/cis545-lecture-2-2-22

Knowledge and Entity-Relationship Graphs

Susan B. Davidson and Zachary G. Ives

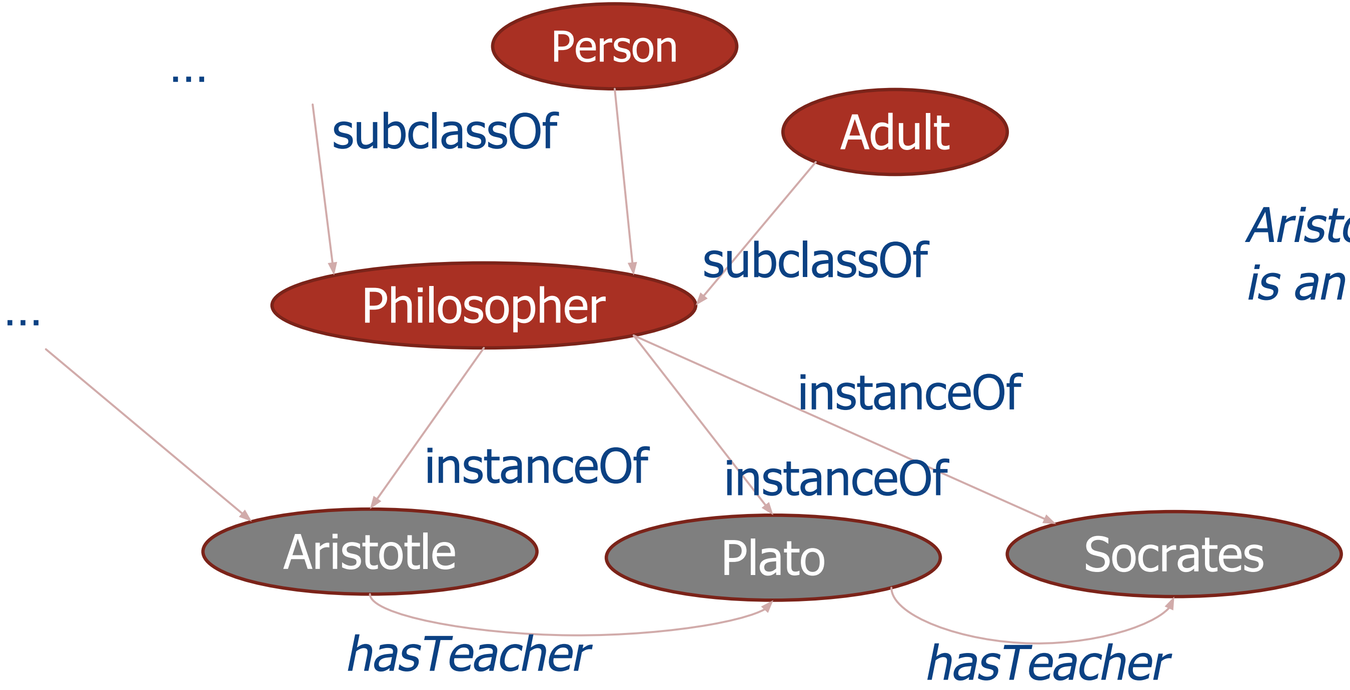
University of Pennsylvania

CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-31-22

Thinking of this as a “Knowledge Graph” of Relationships



Aristotle is an adult

https://tinyurl.com/cis545-lecture-2-2-22

Real Knowledge Graphs for the Web

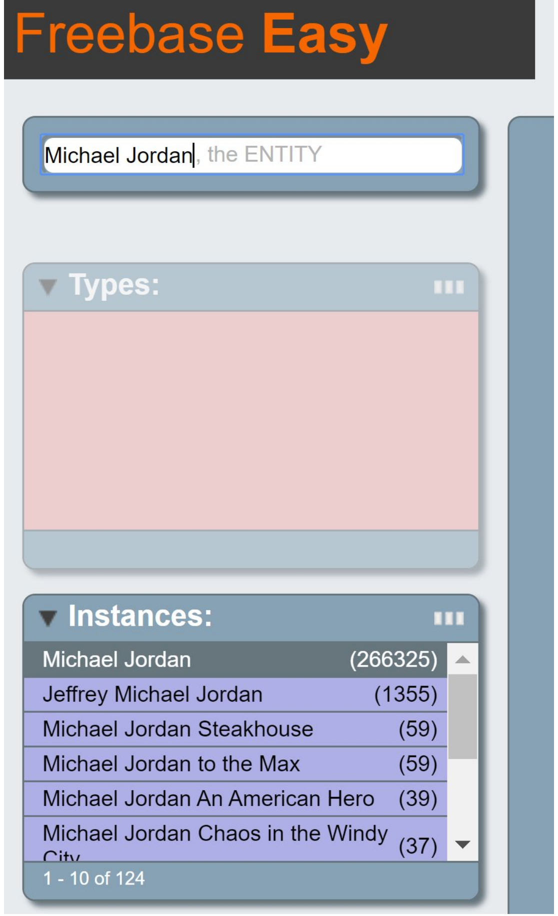
Freebase: the basis of the **Google Knowledge Graph**

- No longer updated externally, data available at http://freebase-easy.cs.uni-freiburg.de/browse/

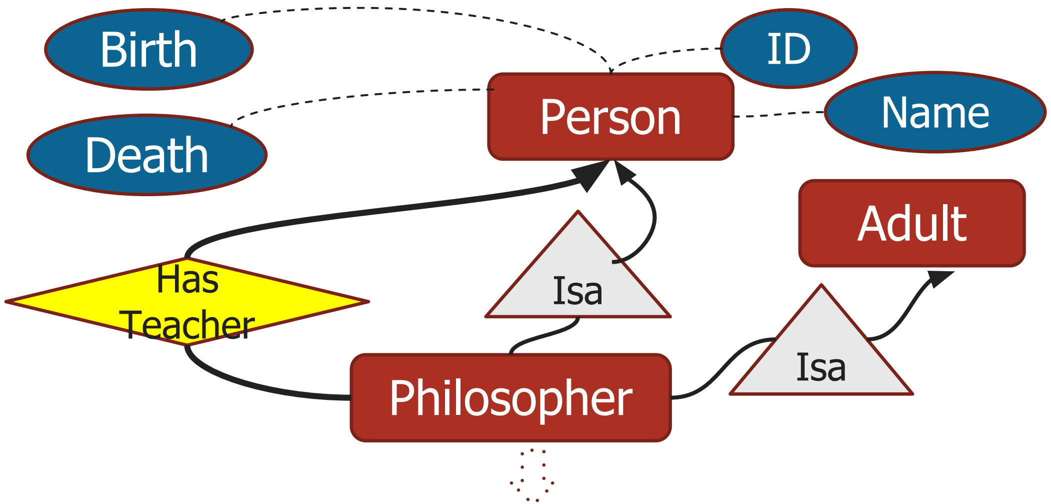
DBpedia: a crawl and extraction of Wikipedia, wiki.dbpedia.org

YAGO, https://github.com/yago-naga/yago3

https://tinyurl.com/cis545-lecture-2-2-22



Entity-Relationship Graphs Codify Sets of Entities, Properties, and Relationships



"Isa":
subclass inherits all *properties* of superclass

Superclass: includes all *members* of subclasses

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

https://tiny

Philosopher is an entity set with many philosophers, who are *also* people

Recap

- We have two main (often, interchangeable) ways of modeling relationships
  - Logical constraints, sometimes as SQL queries
  - Paths in a graph

- E-R graphs codify entity sets, relationship sets, properties, instances

https://tinyurl.com/cis545-lecture-2-2-22

Brief Review

https://canvas.upenn.edu/courses/1636888/quizzes/2771509 (06C)

- To determine whether an instance is a member of a class in a knowledge graph, we can:
  - find a connection from the class to subclasses
  - find a connection between instances
  - find a connection from the class to the instance
  - find a path from the class, possibly through subclasses, to the instance
- With an IS-A relationship, the derived entity set (sub-type):
  - includes all properties of the supertype or parent entity set
  - includes all properties and instances of the supertype or parent entity set
  - includes all instances of the supertype or parent entity set
  - includes all properties of the subtype or child entity set

https://tinyurl.com/cis545-lecture-2-2-22

Storing Entities and Relationships

Susan B. Davidson and Zachary G. Ives

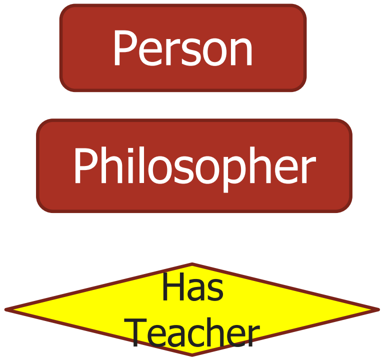
University of Pennsylvania

CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-31-22

How Do We Store Entities & Relationships?



Entity set: represents all of the entities of a type, and their properties

- Person: ID, name, birth, death
- Philosopher: inherits the same fields, possibly adds new ones

Relationship set: represents a link between people

- *HasTeacher*(teacher: ID of Person, student: ID of Person)

Person (Also: Philosopher)

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

Key

Foreign keys

HasTeacher

Teacher	Student
1233	1234
1232	1233

https://tinyurl.com/cis545-lecture-2-2-22

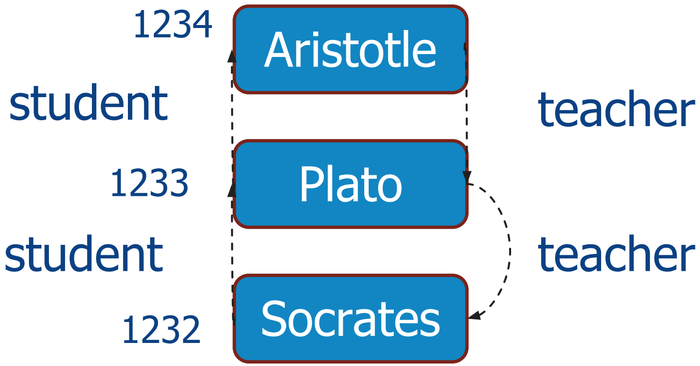
The Tables Represent a Graph of Connections

Person

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

HasTeacher

Teacher	Student
1233	1234
1232	1233



https://tinyurl.com/cis545-lecture-2-2-22

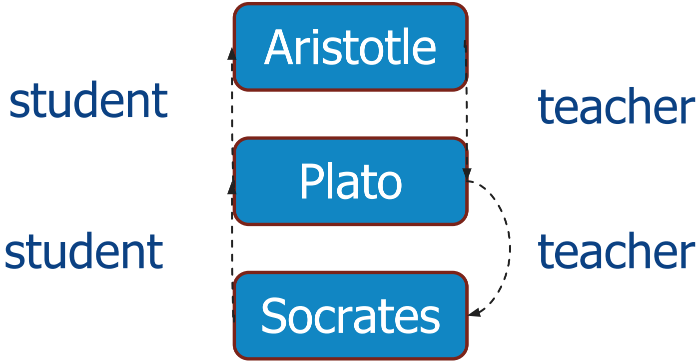
Who Is the Teacher of Aristotle's Teacher?

Person

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

HasTeacher

Teacher	Student
1233	1234
1232	1233



SELECT
FROM
WHERE

https://tinyurl.com/cis545-lecture-2-2-22

Who Is the Teacher of Aristotle's Teacher?

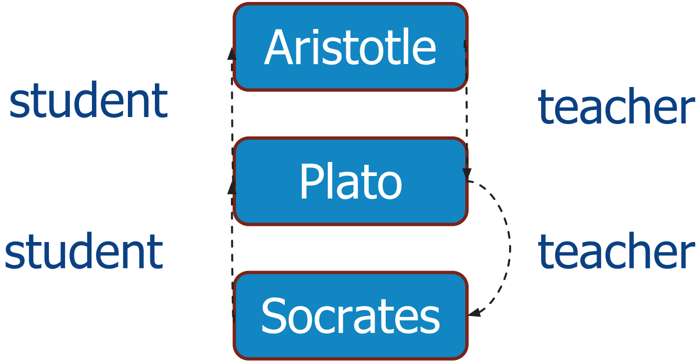
Person

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

A

HasTeacher

Teacher	Student
1233	1234
1232	1233



SELECT
FROM Person A
WHERE A.name='Aristotle'

https://tinyurl.com/cis545-lecture-2-2-22

Who Is the Teacher of Aristotle's Teacher?

Person

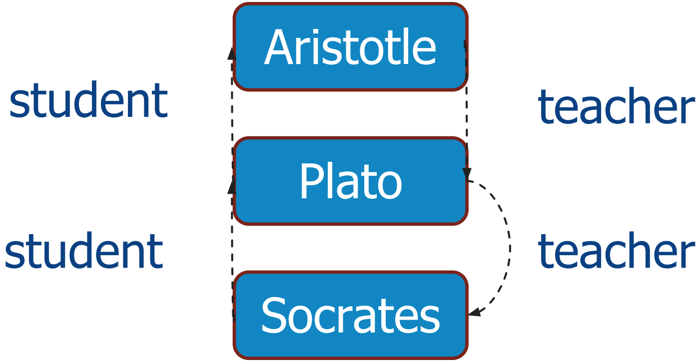
ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

A

HasTeacher

Pl

Teacher	Student
1233	1234
1232	1233




```
SELECT
FROM Person A JOIN HasTeacher Pl
      ON ID=Student
WHERE A.name='Aristotle'
```

https://tinyurl.com/cis545-lecture-2-2-22

Who Is the Teacher of Aristotle's Teacher?

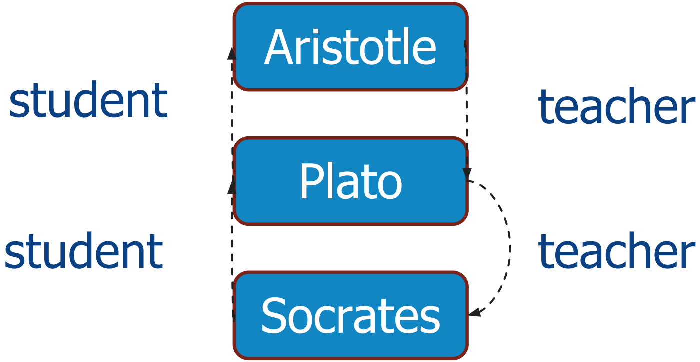
Person

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

A

HasTeacher

	Teacher	Student
Pl	1233	1234
So	1232	1233




```
SELECT
FROM Person A JOIN HasTeacher Pl
      ON A.ID=Student JOIN HasTeacher So
      ON Pl.teacher = So.student
WHERE A.name='Aristotle'
```

https://tinyurl.com/cis545-lecture-2-2-22

Who Is the Teacher of Aristotle's Teacher?

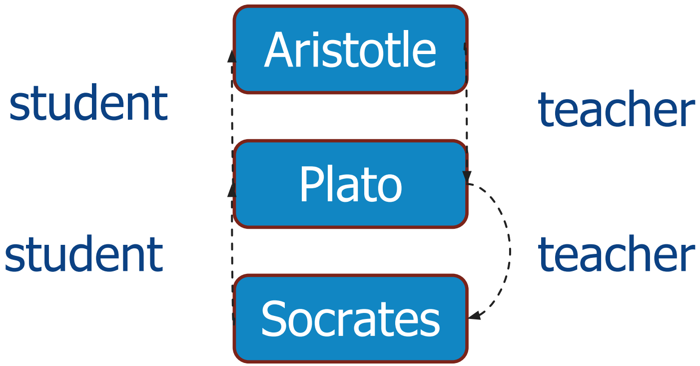
Person

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

A

HasTeacher

	Teacher	Student
Pl	1233	1234
So	1232	1233




```
SELECT So.teacher
FROM Person A JOIN HasTeacher Pl
       ON A.ID=Student JOIN HasTeacher So
       ON Pl.teacher = So.student
WHERE A.name='Aristotle'
```

https://tinyurl.com/cis545-lecture-2-2-22

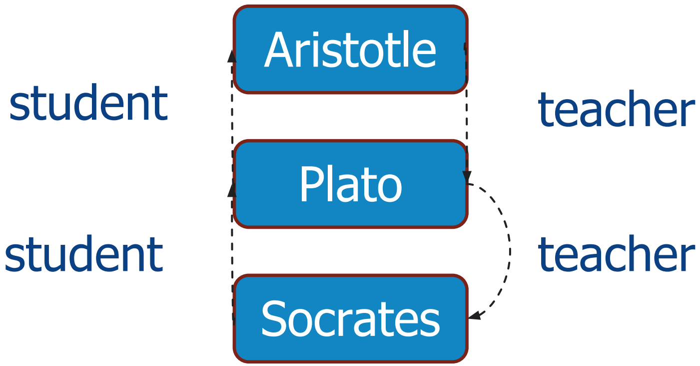
Who Are Socrates' Academic Descendants?

Person

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

HasTeacher

Teacher	Student
1233	1234
1232	1233



SELECT
FROM
WHERE

https://tinyurl.com/cis545-lecture-2-2-22

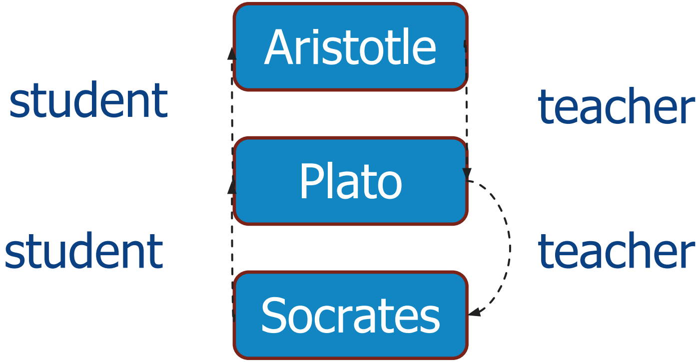
Who Are Socrates' Academic Descendants?

Person

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

HasTeacher

Teacher	Student
1233	1234
1232	1233



SELECT
FROM Person So
WHERE So.Name='Socrates'

https://tinyurl.com/cis545-lecture-2-2-22

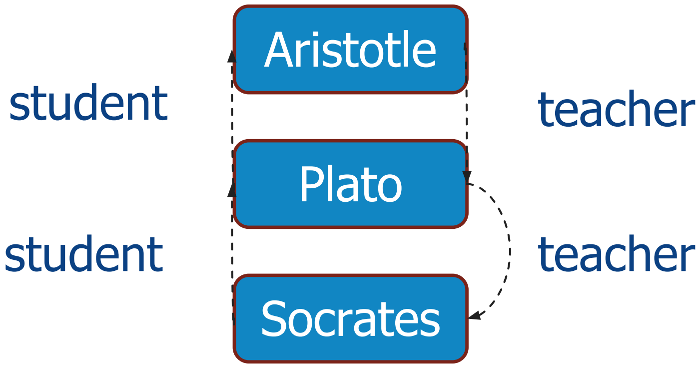
Who Are Socrates' Academic Descendants?

Person

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

HasTeacher

Teacher	Student
1233	1234
1232	1233




```
SELECT Pl.Student
FROM Person So JOIN HasTeacher Pl
   ON So.ID = Pl.teacher
WHERE So.Name='Socrates'
```

https://tinyurl.com/cis545-lecture-2-2-22

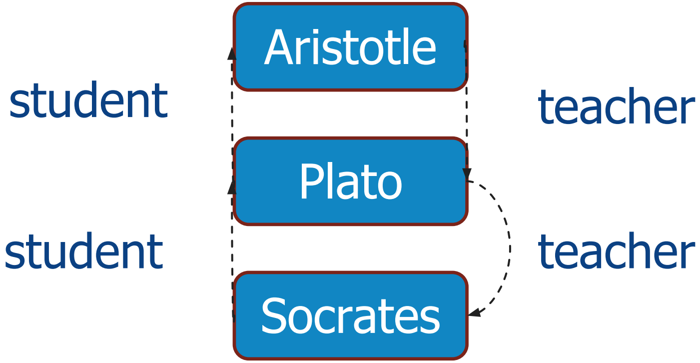
Who Are Socrates' Academic Descendants?

Person

ID	Name	Birth	Death
1234	Aristotle	384 BC	322 BC
1233	Plato	428 BC	348 BC
1232	Socrates	470 BC	399 BC

HasTeacher

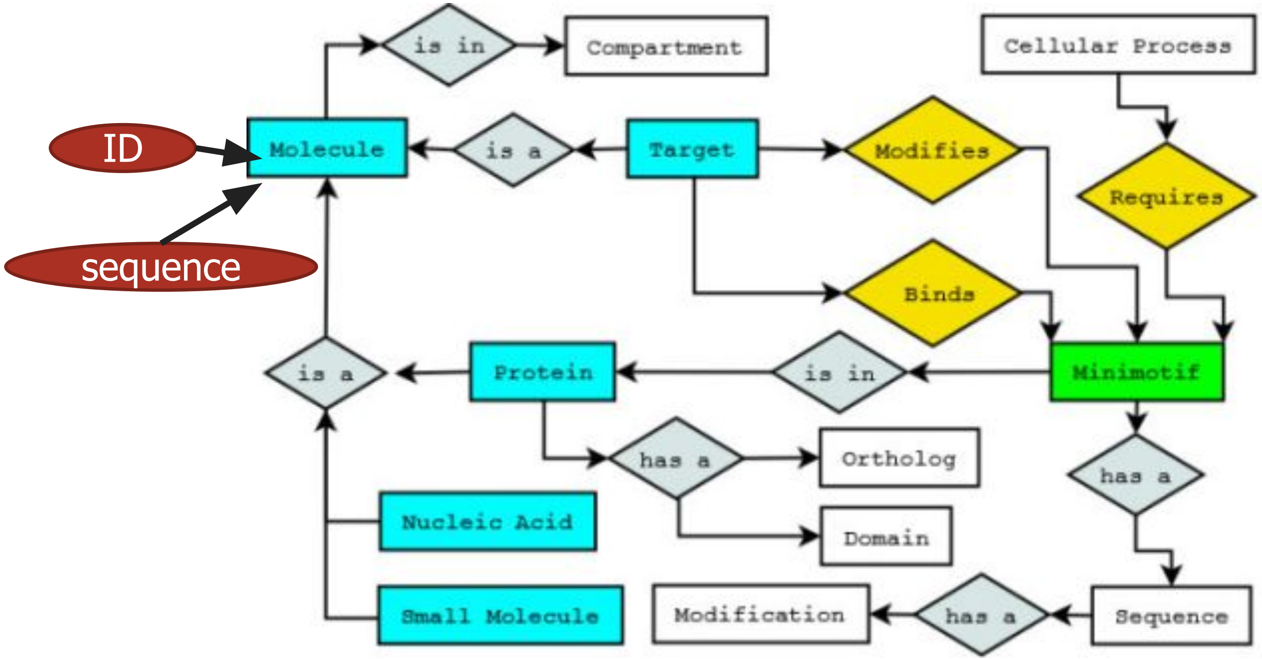
Teacher	Student
1233	1234
1232	1233




```
SELECT Pl.Student
FROM Person So JOIN HasTeacher Pl
   ON So.ID = Pl.teacher
WHERE So.Name='Socrates'
UNION
SELECT Ar.Student
FROM Person So JOIN HasTeacher Pl
   ON So.ID = Pl.teacher JOIN HasTeacher Ar
   ON Ar.teacher = Pl.student
WHERE So.Name='Socrates'
```

https://tinyurl.com/cis545-lecture-2-2-22

ER is a General Model: A Graph of Entities & Relationships



https://tinyurl.com/cis545-lecture-2-2-22

Vyas et al, BMC Genomics 2009, *A proposed syntax for Minimotif Semantics, version 1*

General Database Design

Deciding on the entities, relationships, and constraints is part of *database design*

- There are ways to do this to minimize the errors in the database, and make it easiest to keep *consistent*

- See CIS 450/550 for details

For this class: we'll assume we do simple E-R diagrams with properties

… and that each node becomes a Dataframe

https://tinyurl.com/cis545-lecture-2-2-22

Recap: Basic Concepts in Data Modeling

Knowledge represented as **concepts or classes**, which can correspond to tables

- But there is also a notion of **subclassing** (inheriting fields)
- And of **instances** (rows in the tables)

Knowledge representation often describes these relationships as constraints

We can capture knowledge using graphs with nodes (entity sets, concepts) and edges (relationship sets)

- Entity-relationship diagrams show this
- Entity sets and relationship sets can both become tables!
- Graphs + queries can be used to capture any kind of data and relationships (not always conveniently)

https://tinyurl.com/cis545-lecture-2-2-22

Brief Review

https://canvas.upenn.edu/courses/1636888/quizzes/2771559 (06D)

- A foreign key:
  a. takes on a value from a key in another table
  b. is a C++ pointer
  c. must be unique within its own table
  d. has multiple values per row
- The data in a relational database can be modeled as a graph of tuple relationships, as we saw in the slides. How do we traverse edges in this graph?
  a. filtering / selection
  b. applymap
  c. joins
  d. unions

https://tinyurl.com/cis545-lecture-2-2-22

Hierarchical Data and NoSQL

Susan B. Davidson and Zachary G. Ives

University of Pennsylvania

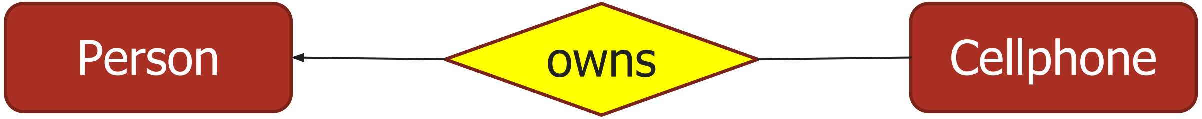
CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-31-22

Hierarchy vs Relations (“NoSQL” vs “SQL”)

Sometimes it’s convenient to take data we could codify as a graph:



And instead save it as a *tree* or *forest*:

[{‘person’: {‘name’: ‘jai’, phones: [{‘mfr’: ‘Apple’, ‘model’: …},
{‘mfr’: ‘Samsung’, ‘model’: …}},
{‘person’: {‘name’: ‘kai’, phones: [{‘mfr’: ‘Apple’, ‘model’: …}]}]

This is what NoSQL databases do!

https://tinyurl.com/cis545-lecture-2-2-22

Let's Now Look at a Working Example: *Social Network Analysis*

https://tinyurl.com/cis545-notebook-03

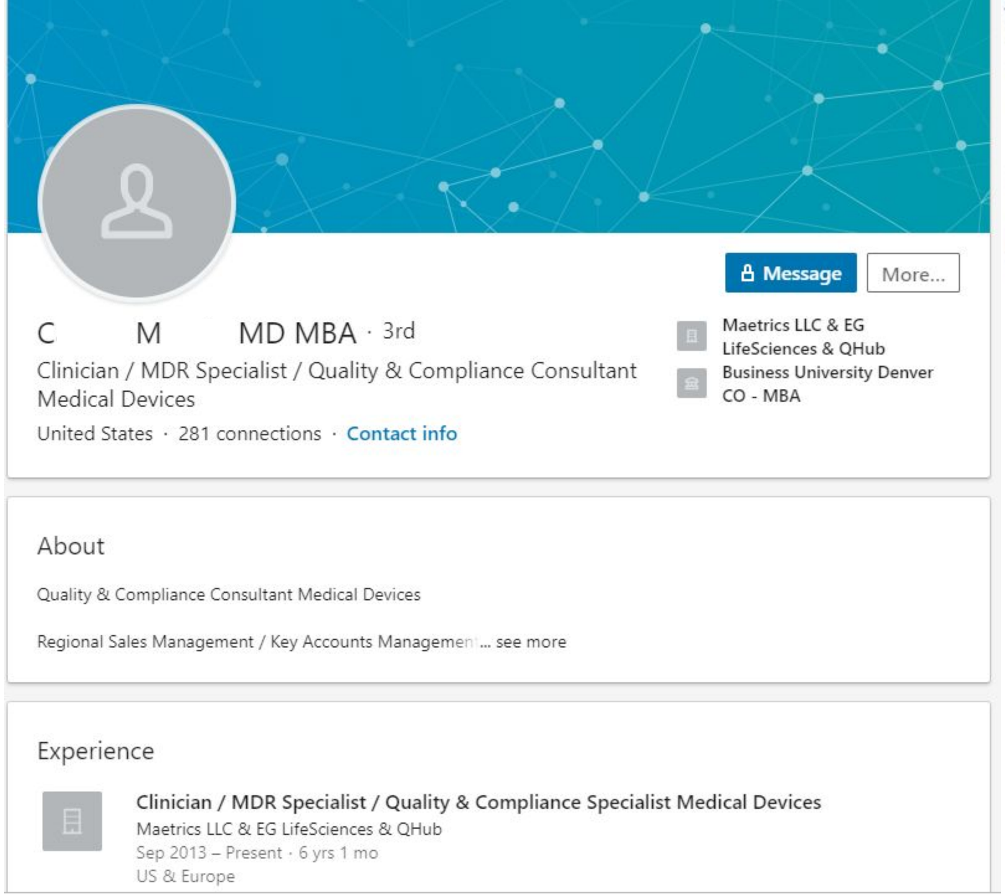
Extracted data from LinkedIn, was in https://www.kaggle.com/linkedindata/linkedin-crawled-profiles-dataset

~3M people, stored as a ~9GB list of lines made up of JSON

(For the Colab version we've cut to 10,000 lines so it executes quickly enough.)

JSON is nested dictionaries and lists – i.e., NoSQL-style !

https://tinyurl.com/cis545-lecture-2-2-22




```
{
"_id":"in-00000001",
"name":{
  "family_name":"M          ",
  "given_name":"Dr C       "
},
"locality":"United States",
"skills":[
  "Key Account Development",
  "Strategic Planning",
  "Market Planning",
  "Team Leadership",
  "Negotiation",
  "Forecasting",
  "Key Account Management",
  "Sales Management",
  "New Business Development",
  "Business Planning",
  "Cross-functional Team Leader
  "Budgeting",
  "Strategy Development",
  "Business Strategy",
  "Consultative Selling",
  "Medical Devices",
  "Customer Relations",
  "Contract Negotiation"
```


https://tinyurl.com/cis545-lecture-2-2-22

NoSQL Databases
(We'll See Details in a Bit)

- Originally, indeed stood for "no-SQL", now "not-only-SQL"
- Typically store **nested objects**, or possibly binary objects, by IDs or keys

Note that a nested object can be captured in relations, via multiple tables!

Some well-known NoSQL systems:

- MongoDB: stores JSON, i.e., lists and dictionaries
- Google Bigtable: stores tuples with irregular properties
- Amazon S3: stores binary files by key

Major differences from SQL databases:

- Querying is often much simpler, eg they often don't do joins!
- They support limited notions of **consistency** when you update [we'll discuss later]

https://tinyurl.com/cis545-lecture-2-2-22

Parsing Even Not-So-Big Data Is Painfully Slow!

```
%%time
# 100,000 records from linkedin
linked_in = open('linkedaa')

people = []

for line in linked_in:
    person = json.loads(line)
    people.append(person)

people_df = pd.DataFrame(people)
people_df[people_df['industry'] == 'Medical Devices']
```

```
CPU times: user 58.2 s, sys: 1min 57s, total: 2min 55s
Wall time: 3min 19s
```

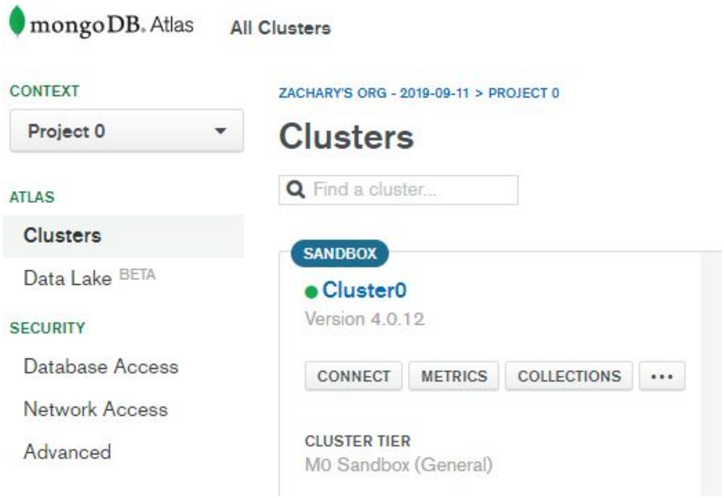
	_id	name	locality	skills	industry	summary
0	in-00000001	{'family_name': 'Mazalu MBA', 'given_name': 'D...	United States	[Key Account Development, Strategic Planning, ...	Medical Devices	SALES MANAGEMENT / BUSINESS DEVELOPMENT / PROJ...
161	in-13806219531	{'family_name': 'Gao', 'given_name': 'Tony'}	China	[ISO 13485, Medical Devices]	Medical Devices	NaN

https://tinyurl.com/cis545-lecture-2-2-22

Can We Do Better?

Maybe save the data in a way that doesn't require parsing of strings?

https://cloud.mongodb.com



https://tinyurl.com/cis545-lecture-2-2-22

MongoDB NoSQL DBMS
Lets Us Store + Fetch Hierarchical Data

```
client =
MongoClient('mongodb+srv://cis545:1course4all@cluster0-cy1yu.mongodb.
net/test?retryWrites=true&w=majority')

linkedin_db = client['linkedin']
linked_in = open('linkedin.json')

for line in linked_in:
    person = json.loads(line)
    linkedin_db.posts.insert_one(person)
```

https://tinyurl.com/cis545-lecture-2-2-22

Data in MongoDB


```
_id: "in-00001"
> education: Array
> group: Object
> name: Object
  overview_html: "<dl id="overview"><dt id="overview-summary-current-title" class="summa..."
  locality: "Antwerp Area, Belgium"
> skills: Array
  industry: "Pharmaceuticals"
  interval: 20
v experience: Array
  > 0: Object
  > 1: Object
  v 2: Object
      org: "Columbia University"
      title: "Associate Research Scientist"
      start: "August 2006"
      desc: "Work on peptide to restore wt p53 function in cancer."
  > 3: Object
  > 4: Object
  summary: "Ph.D. scientist with background in cancer research, translational medi..."
  url: "http://be.linkedin.com/in/00001"
> also_view: Array
  specilities: "Biomarkers in Oncology, Cancer Genomics, Molecular Profiling of Cancer..."
> events: Array
```


https://tinyurl.com/cis545-lecture-2-2-22

Finding Things,
in a Dataframe vs in MongoDB

```
def find_skills_in_list(skill):
    for post in list_for_comparison:
        if 'skills' in post:
            skills = post['skills']
            for this_skill in skills:
                if this_skill == skill:
                    return post
    return None

def find_skills_in_mongodb(skill):
    return linkedin_db.posts.find_one({'skills': skill})
```

Similar to an XPath
posts[skills=mySkill]

https://tinyurl.com/cis545-lecture-2-2-22

The Story So Far

For hierarchical data, it's often useful to use a NoSQL database

- Natural mapping for nested dictionaries, JSON, etc.

Such systems often support queries that are in terms of nested content

https://tinyurl.com/cis545-lecture-2-2-22

Brief Review

https://canvas.upenn.edu/courses/1636888/quizzes/2771582 (06E)

- We split hierarchical data into tables when
  a. the data has multiple values per parent item
  b. the data has a separate column name
  c. the data is a nested dictionary
  d. the data can be represented as an ER diagram
- We may want to use left outerjoins to reassemble hierarchical data because
  a. there may be parent items with no children
  b. outerjoin is faster than innerjoin
  c. outerjoin is hierarchical
  d. outerjoin returns a subset of the answers of innerjoin

https://tinyurl.com/cis545-lecture-2-2-22

Hierarchy and Relations

Susan B. Davidson and Zachary G. Ives

University of Pennsylvania

CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-31-22

How Do We Convert Hierarchical Data to Dataframes?

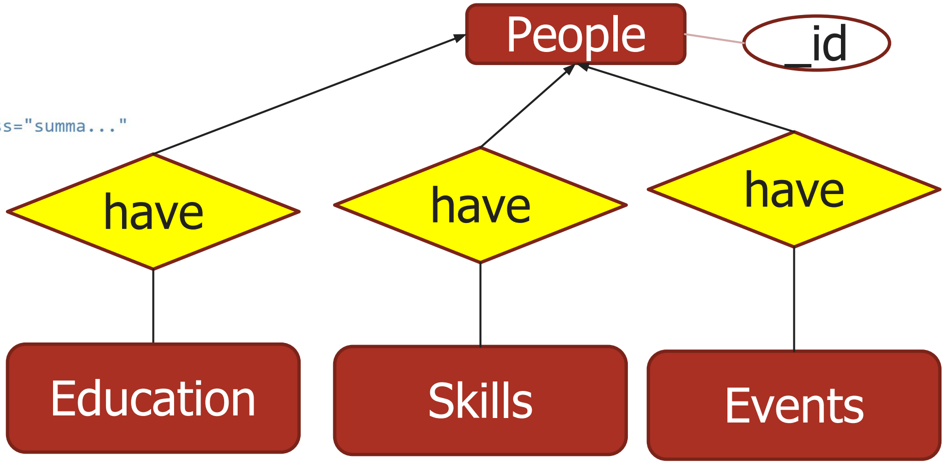
Hierarchical data doesn’t work well for data visualization or machine learning


```
_id: "in-00001"
> education: Array
> group: Object
> name: Object
  overview_html: "<dl id="overview"><dt id="overview-summary-current-title" class="summa..."
  locality: "Antwerp Area, Belgium"
> skills: Array
  industry: "Pharmaceuticals"
  interval: 20
v experience: Array
  > 0: Object
  > 1: Object
  v 2: Object
      org: "Columbia University"
      title: "Associate Research Scientist"
      start: "August 2006"
      desc: "Work on peptide to restore wt p53 function in cancer."
  > 3: Object
  > 4: Object
  summary: "Ph.D. scientist with background in cancer research, translational medi..."
  url: "http://be.linkedin.com/in/00001"
> also_view: Array
  specilities: "Biomarkers in Oncology, Cancer Genomics, Molecular Profiling of Cancer..."
> events: Array
```


https://tinyurl.com/cis545-lecture-2-2-22

The Basic Idea:
Split into Tables When There Isn’t 1:1


```
  _id: "in-00001"
> education: Array
> group: Object
> name: Object
  overview_html: "<dl id="overview"><dt id="overview-summary-current-title" class="summa..."
  locality: "Antwerp Area, Belgium"
> skills: Array
  industry: "Pharmaceuticals"
  interval: 20
v experience: Array
  > 0: Object
  > 1: Object
  v 2: Object
      org: "Columbia University"
      title: "Associate Research Scientist"
      start: "August 2006"
      desc: "Work on peptide to restore wt p53 function in cancer."
  > 3: Object
  > 4: Object
  summary: "Ph.D. scientist with background in cancer research, translational medi..."
  url: "http://be.linkedin.com/in/00001"
```

https://tinyurl.com/cis545-lecture-2-2-22

The Basic Idea: Nesting Becomes Links ("Key/Foreign Key")


```
_id: "in-00001"
> education: Array
> group: Object
> name: Object
  overview_html: "<dl id="overview"><dt id="overview-summary-cu
  locality: "Antwerp Area, Belgium"
> skills: Array
  industry: "Pharmaceuticals"
  interval: 20
v experience: Array
  > 0: Object
  > 1: Object
  v 2: Object
      org: "Columbia University"
      title: "Associate Research Scientist"
      start: "August 2006"
      desc: "Work on peptide to restore wt p53 function in ca
  > 3: Object
  > 4: Object
  summary: "Ph.D. scientist with background in cancer research,
  url: "http://be.linkedin.com/in/00001"
> also_view: Array
  specilities: "Biomarkers in Oncology, Cancer Genomics, Molecu
> events: Array
```


people

_id	Overview_html	locality	industry	…
in-00001	<dl id=…	Antwerp Area	Pharmaceu	

experience

person	org	title	start	desc
in-00001	Columbia	Assoc…	August	Wor…
in-00001	…	…	…	…

https://tinyurl.com/cis545-lecture-2-2-22

Reassembling through (Left Outer) Joins

```
pd.read_sql_query("select _id, org" +\
                  " from people left join experience on _id=person ",\
                  conn)
```

_id	org
in-00001	Albert Einstein Medical Center
in-00001	Columbia University
in-00001	Johnson and Johnson

```
pd.read_sql_query("select _id, \'[\' + group_concat(org) + \']\'" +\
                  " from people left join experience on _id=person "+\
                  " group by _id", conn)
```

_id	experience
in-00000001	None
in-00001	Albert Einstein Medical Center,Columbia Univer...
in-00006	UCSF,Wyss Institute for Biologically Inspired ...

https://tinyurl.com/cis545-lecture-2-2-22

Views

Sometimes we use a query enough that we want to give its results a name, and make it essentially a table

```
conn.execute("create view people_experience as " +\
             " select _id, group_concat(org) as experience " +\
             " from people left join experience on _id=person group by _id")

pd.read_sql_query('select * from people_experience', conn)
```

https://tinyurl.com/cis545-lecture-2-2-22

Occasional Data Storage Considerations: Access and Consistency

Sometimes we may need to allow for failures and "undo"…

- We saw "BEGIN TRANSACTION … COMMIT"
- There is also "ROLLBACK"

Relational DBMS typically provide atomic **transactions** for this; most NoSQL DBMSs don't

A second consideration when the data is shared: what happens when multiple users are editing and querying at the same time?

- **Concurrency control** (how do we handle concurrent updates) and **consistency** (when do I see changes)
- The focus of other courses, e.g. CIS 450/550…

https://tinyurl.com/cis545-lecture-2-2-22

Brief Review

https://canvas.upenn.edu/courses/1636888/quizzes/2771563 (06F)

- NoSQL systems are distinct from relational database management systems in that they:
  a. provide improved consistency
  b. typically support key-based lookup of values
  c. emphasize joins
  d. cannot support SQL

- Hierarchical JSON data ("forests") may be faster to access from MongoDB, versus parsing a file into a dataframe and querying, because
  a. we can query for individual JSON trees
  b. it's faster to parse everything in MongoDB
  c. the network is slow
  d. MongoDB operates faster servers

https://tinyurl.com/cis545-lecture-2-2-22

Recap

We can model hierarchical data in relations

- Using separate relations for each 1:many or many:many nesting level

Need to (left outer)join it to reassemble the hierarchy

- *Views* let us give a name to the reassembled results

If data isn't static, we should consider **transactions** and **concurrency**

https://tinyurl.com/cis545-lecture-2-2-22

Summary of Data Modeling

Representing data's classes and properties is essential

- We can do this via logical constraints (including queries)
- And by diagrams

Two main kinds of data models for databases

- NoSQL – largely hierarchical
- Relational

With joins, each can encode graphs – thus they are equivalent in what they capture, but the convenience of querying differs!

https://tinyurl.com/cis545-lecture-2-2-22