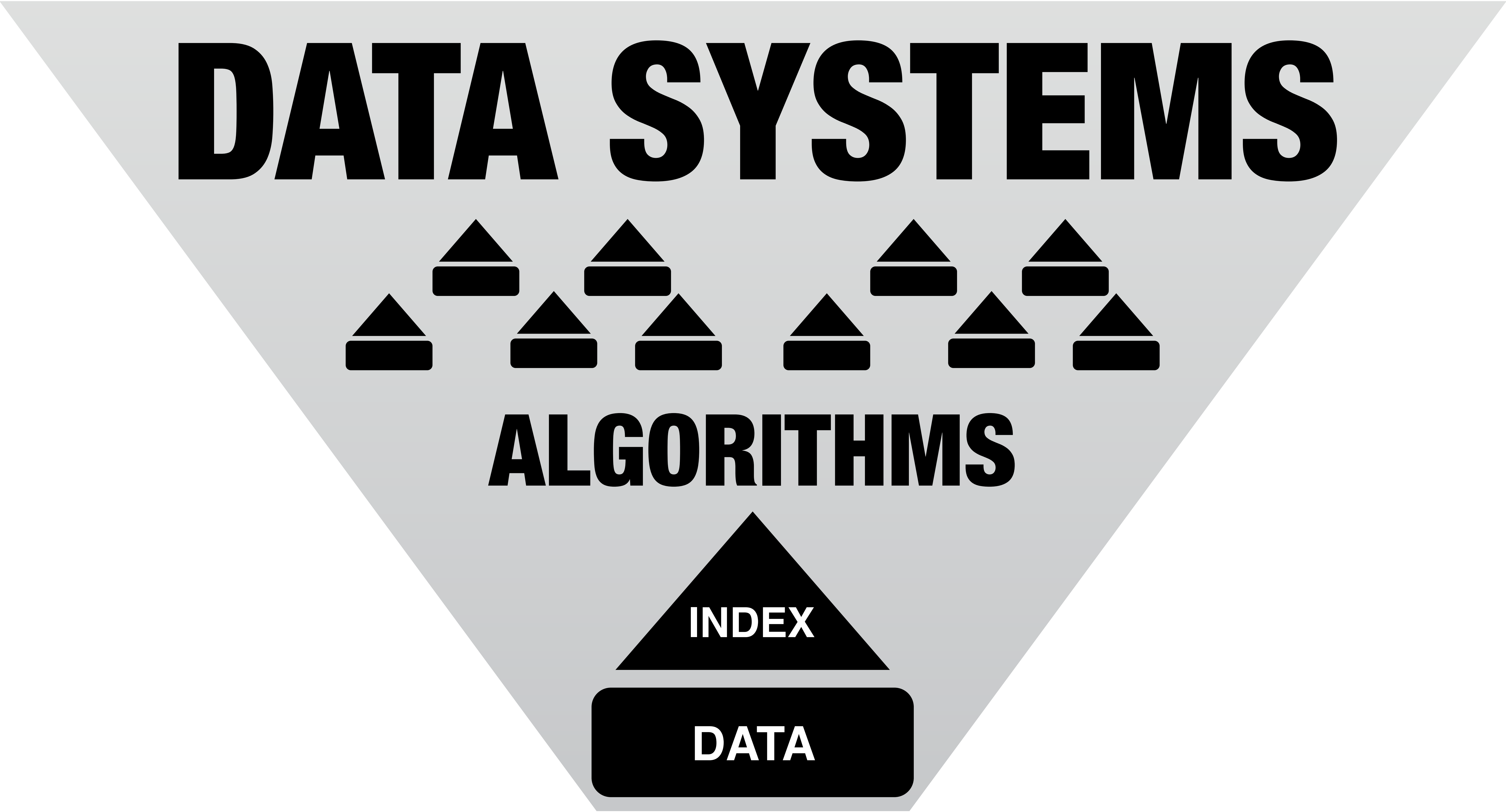
# Advanced Data Management (CSCI 490/680)

## Review

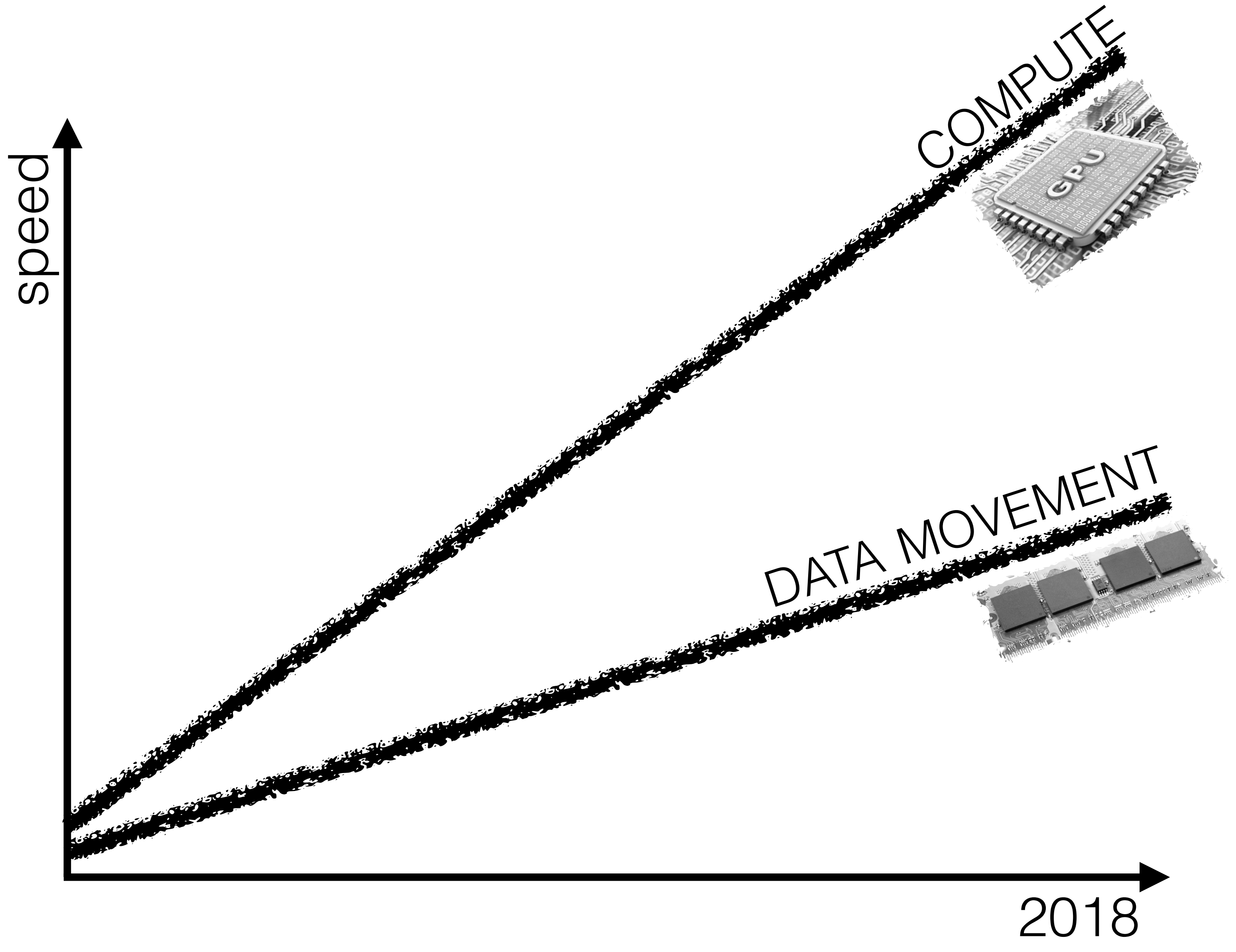
Dr. David Koop

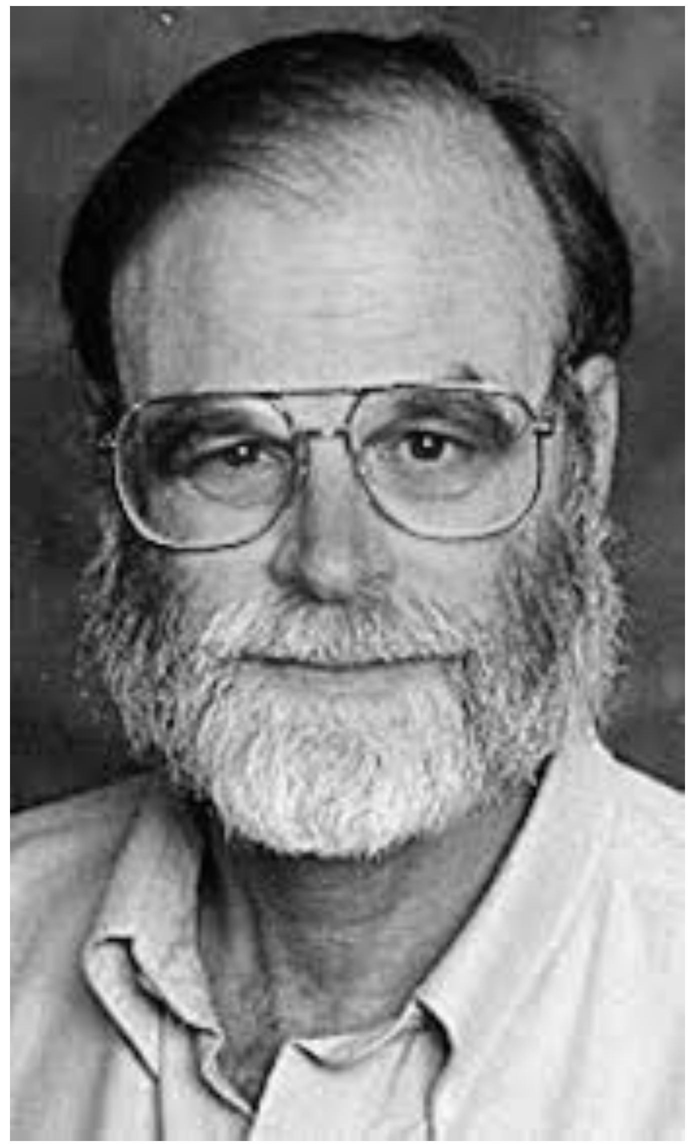
# Data systems rely on algorithms




[S. Idreos, 2019]

# Data structures define performance



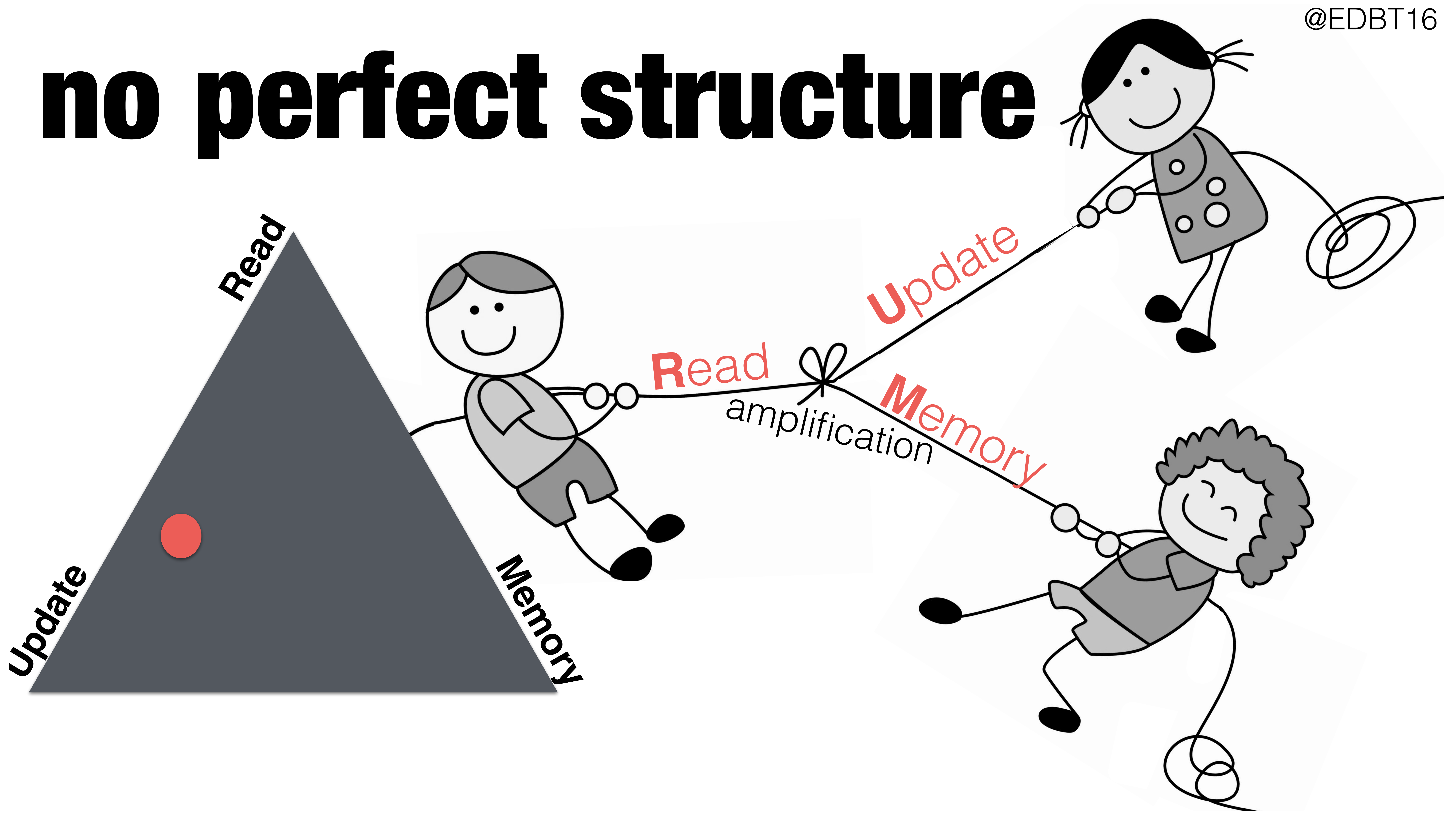



register = this room
caches = this city
memory = nearby city
**disk = Pluto**

Jim Gray, Turing Award 1998

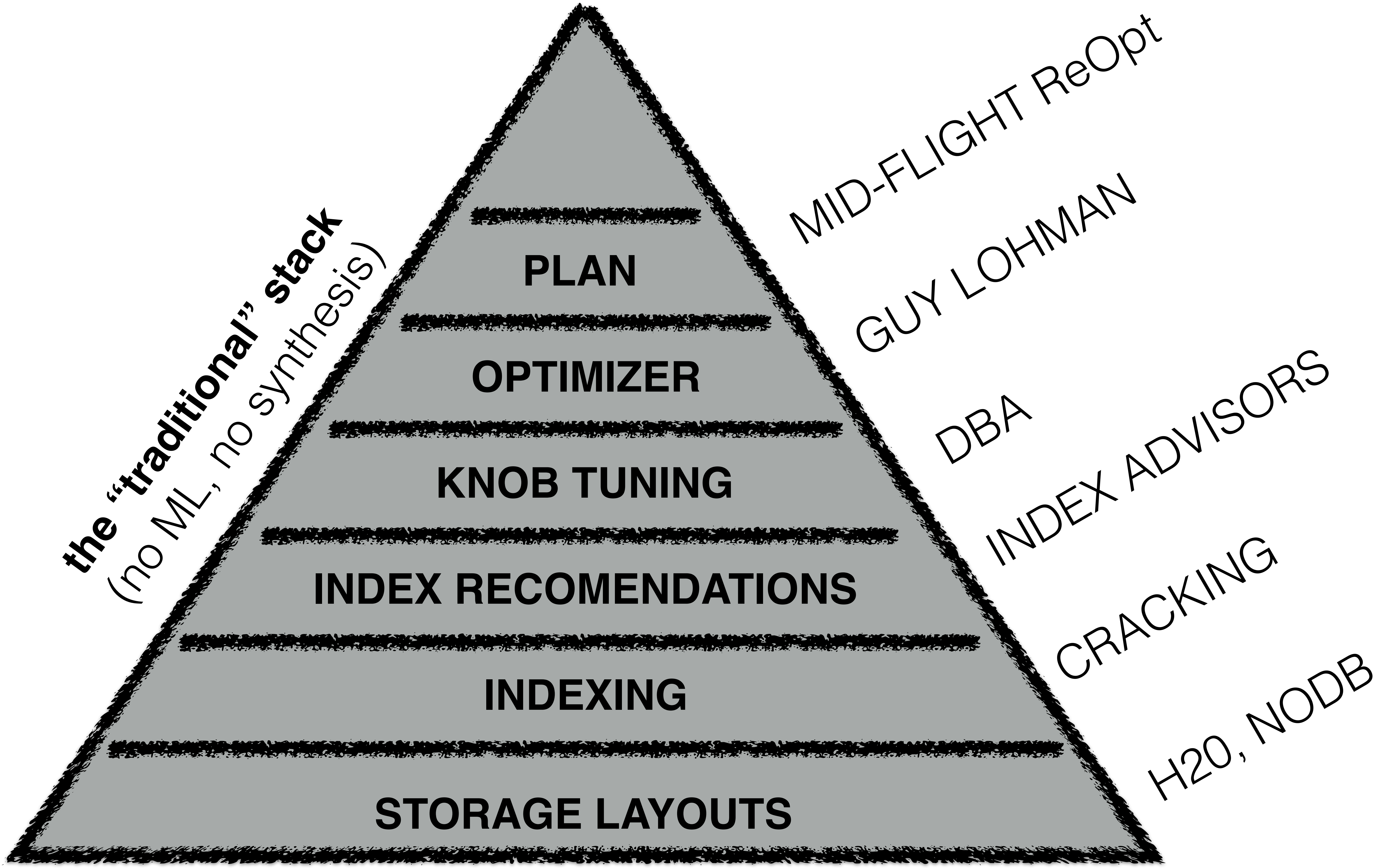
[S. Idreos, 2019]

# Tradeoffs in each structure



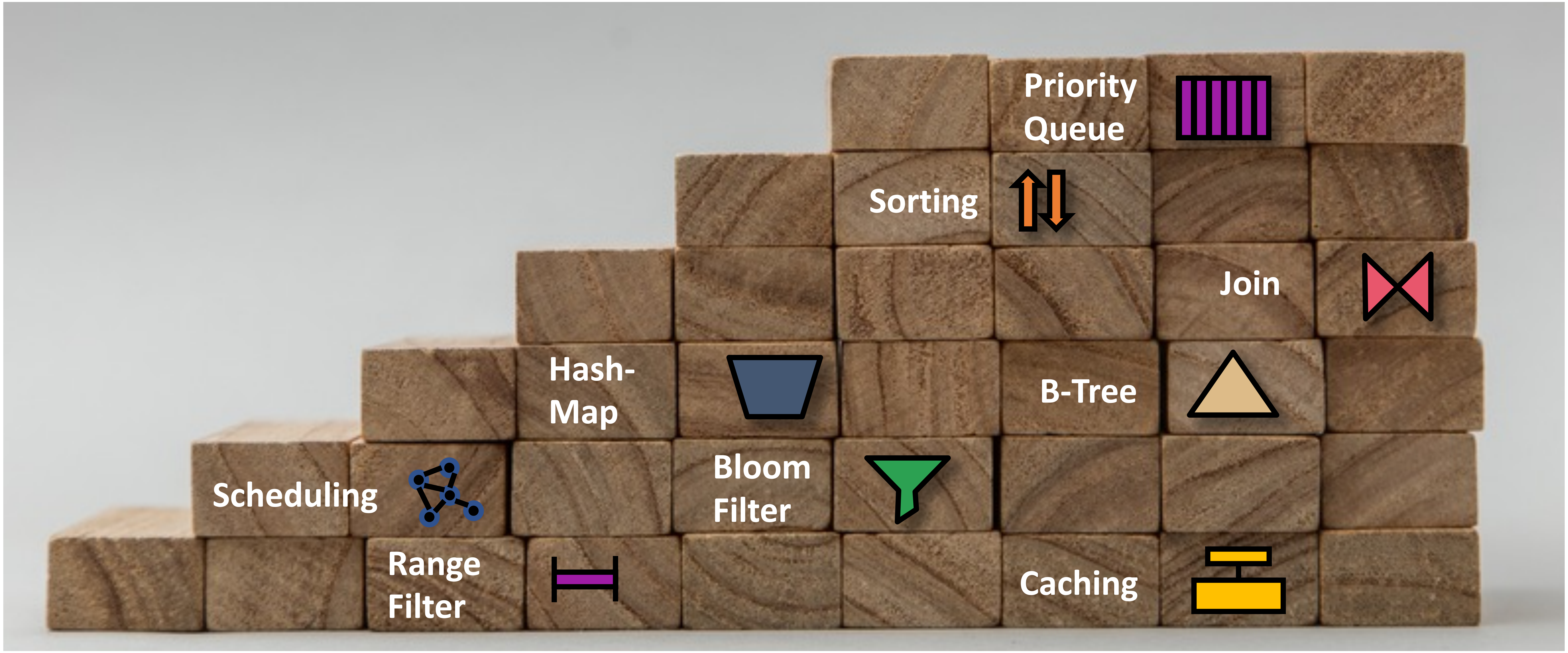

[S. Idreos, 2019]

# "Traditional" Database Research

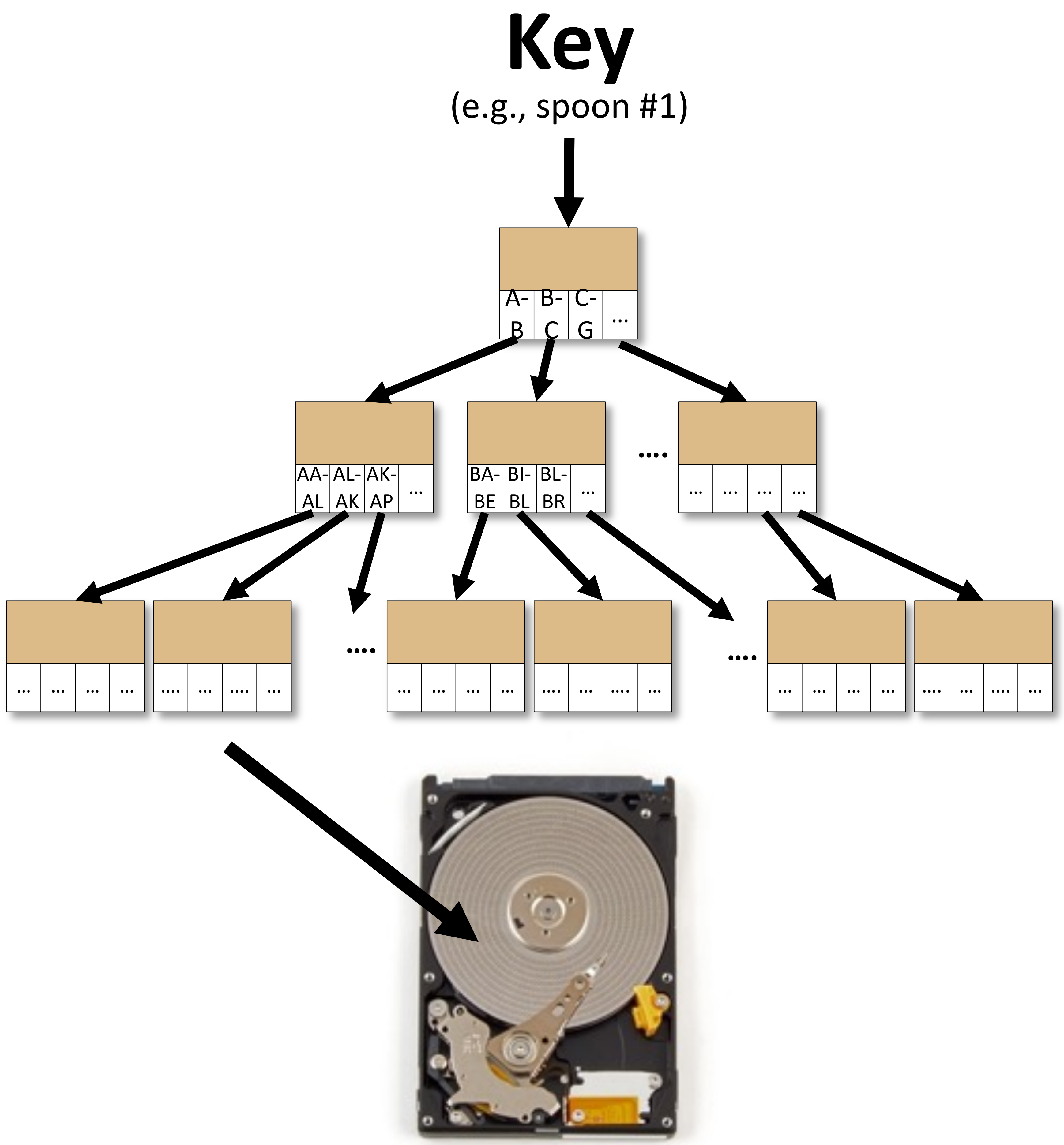


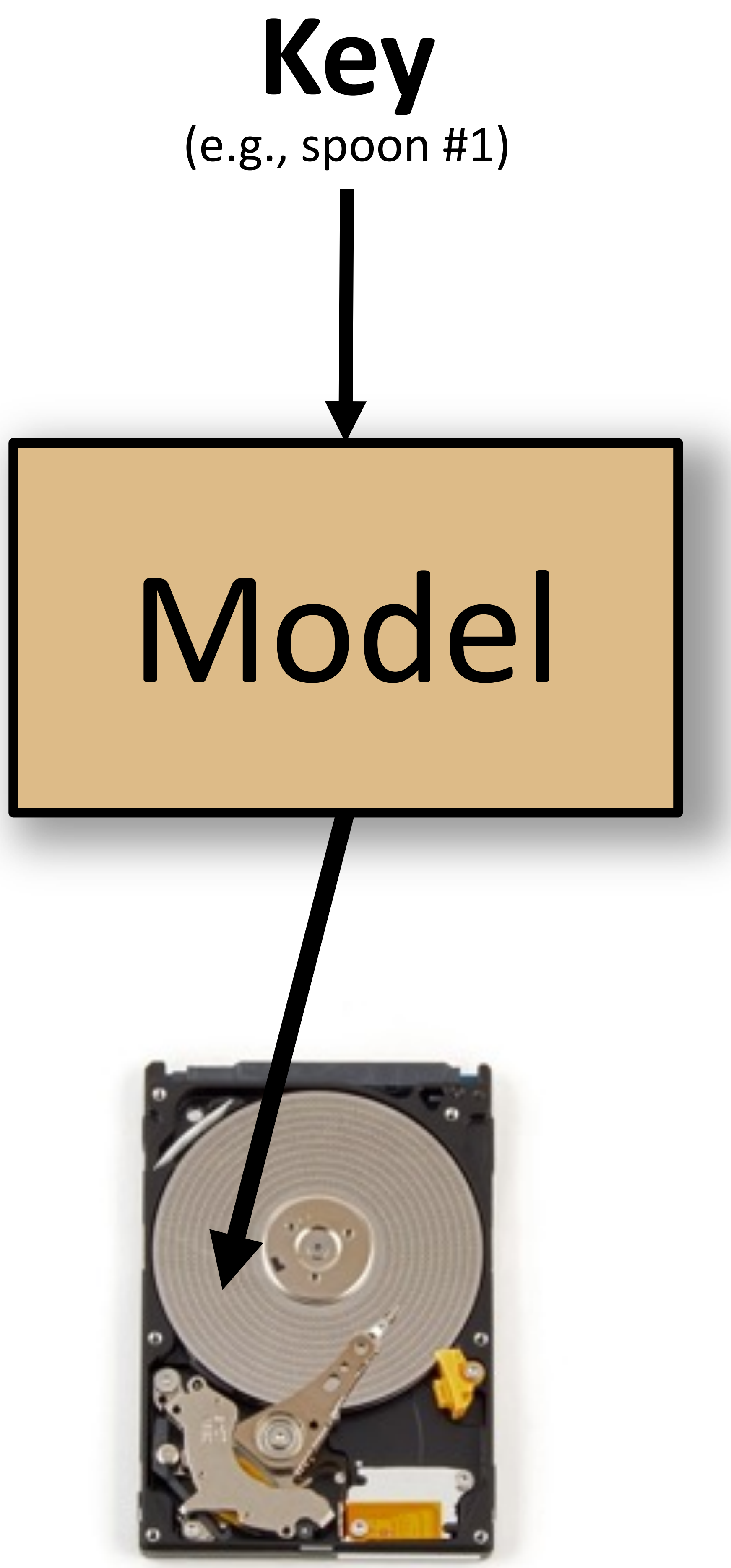

[S. Idreos, 2019]

# Learned Data Structures and Algorithms



# B-Tree

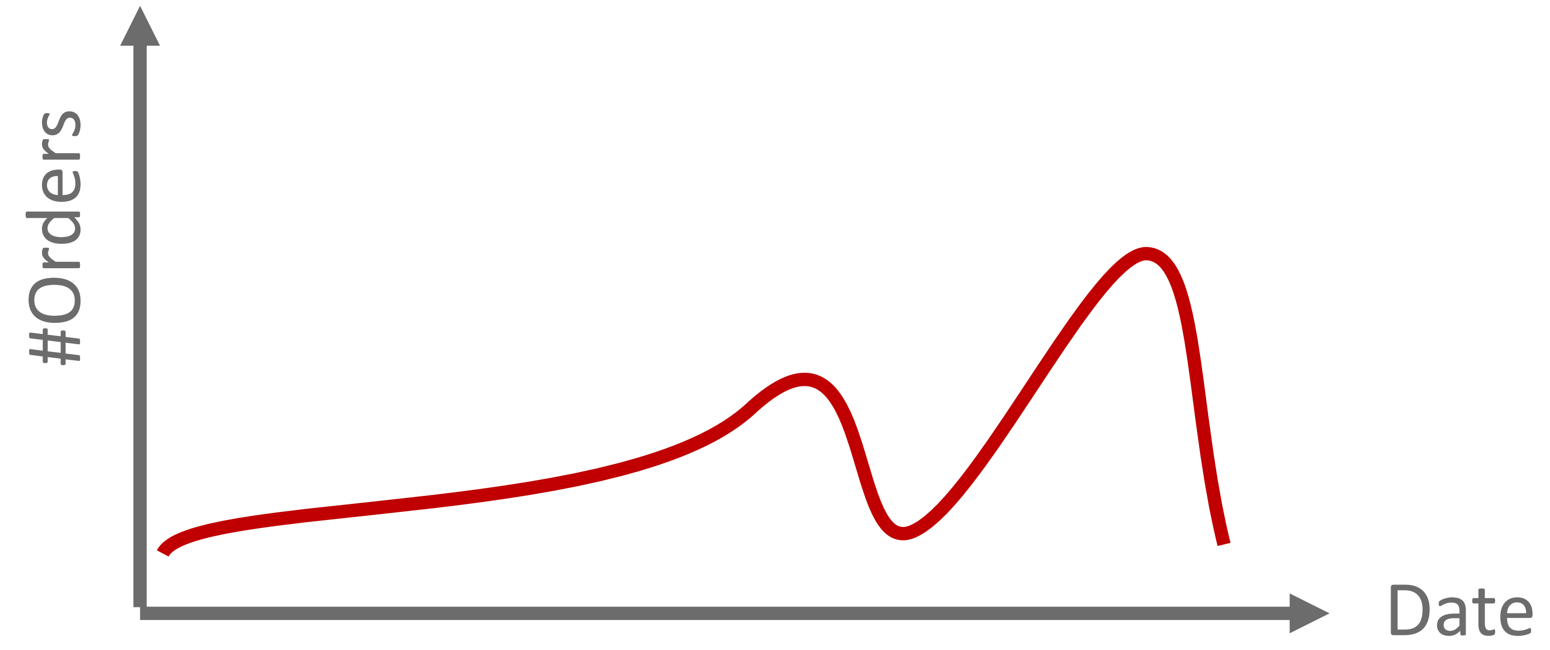


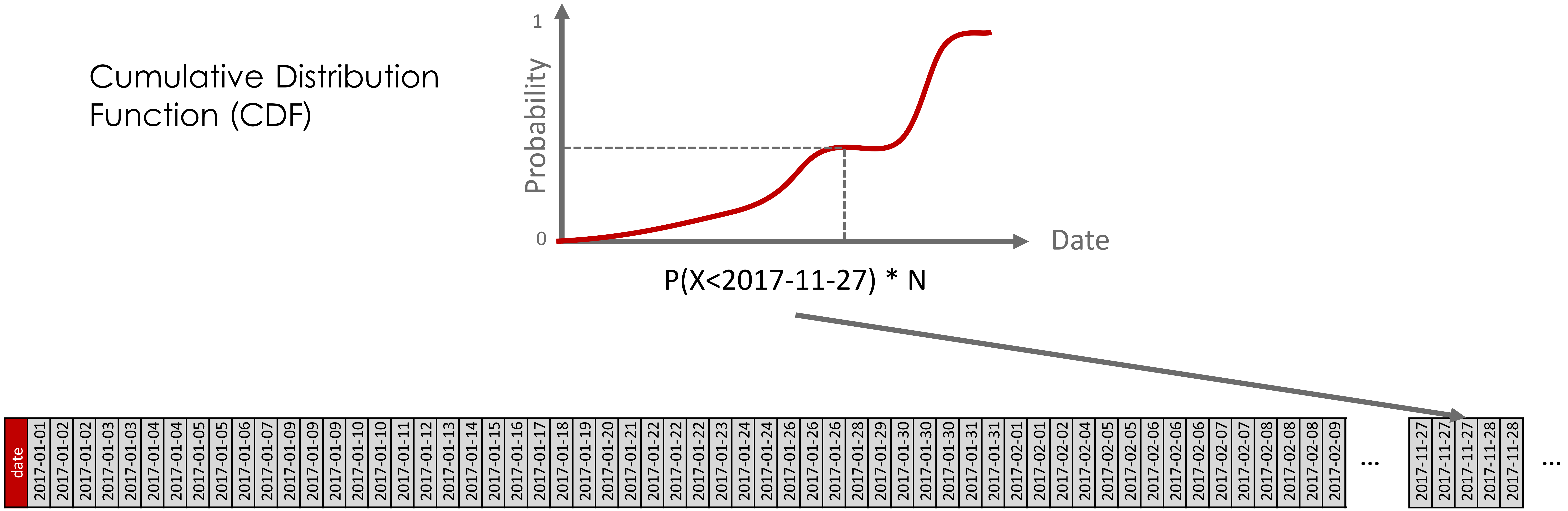

[T. Kraska, 2019]

# Model to Predict Data's Location on Disk

Frequency Distribution



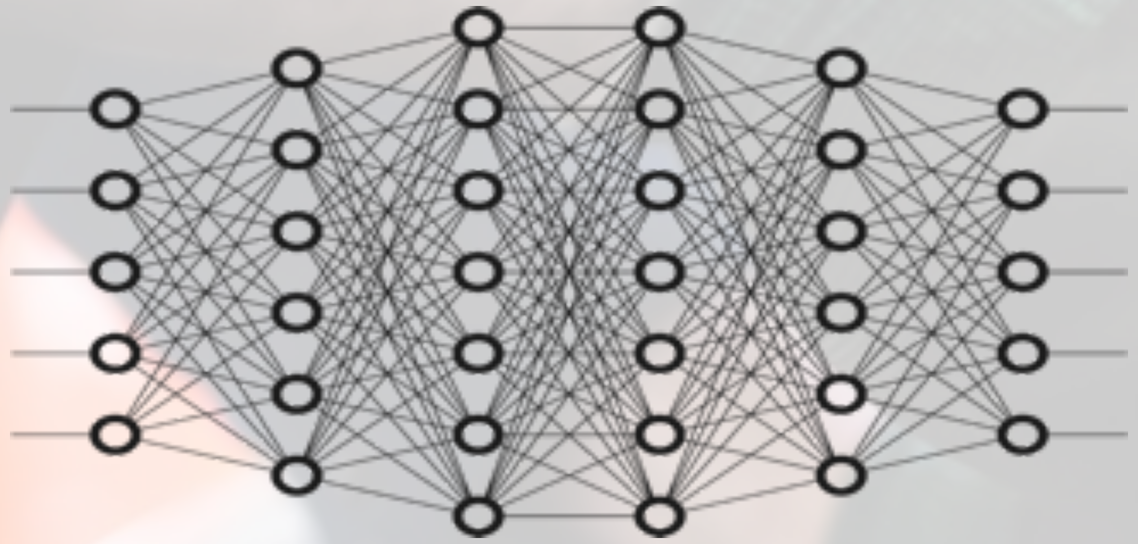

Cumulative Distribution Function (CDF)




[T. Kraska, 2019]

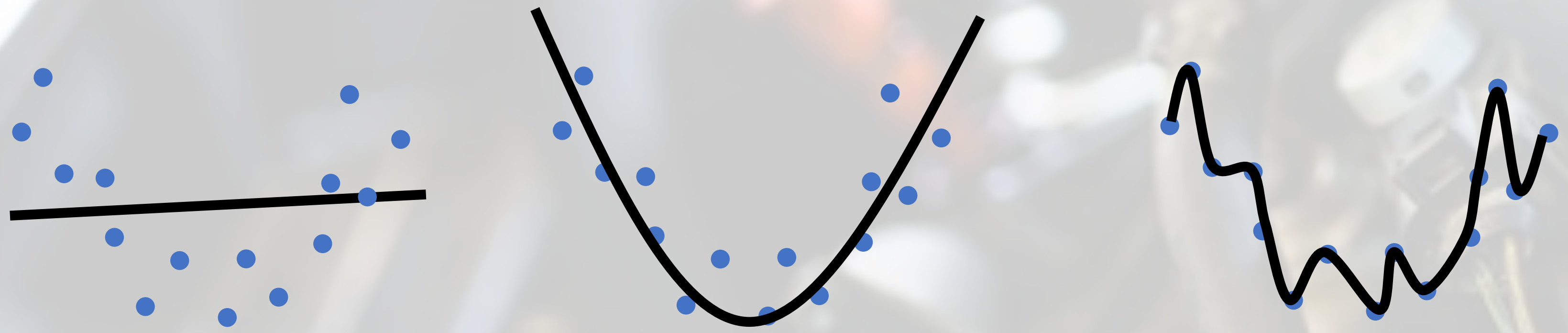
# Challenges

## Traditional model architectures do not work

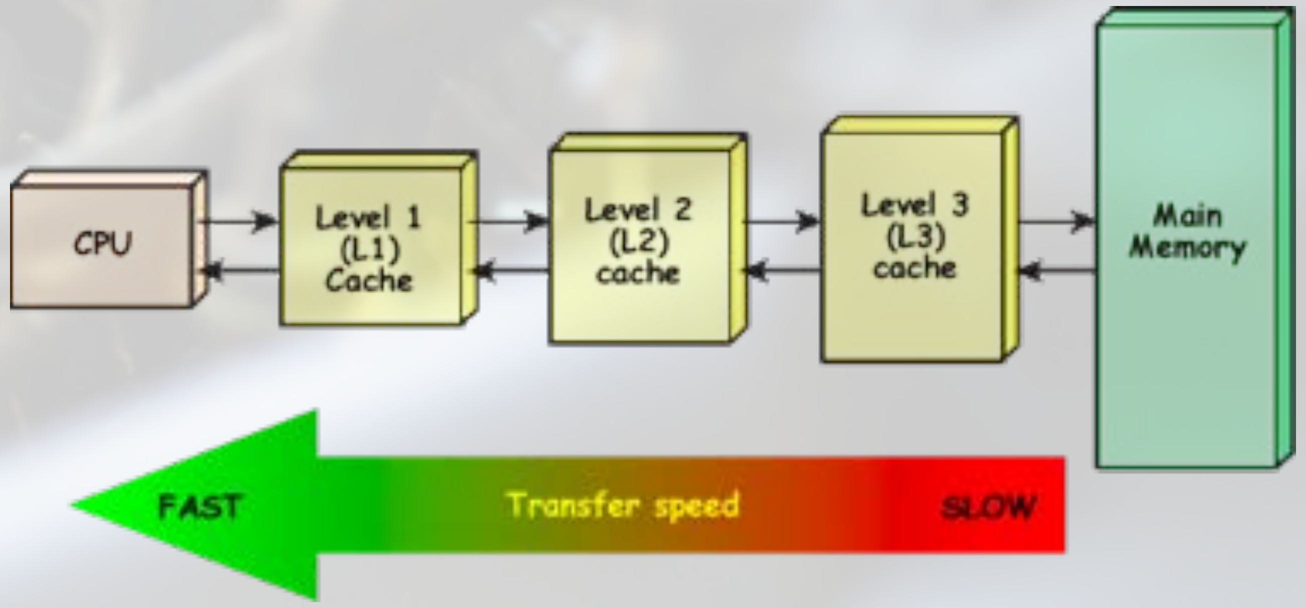


## Frameworks are not designed for nano-second execution

## Overfitting can be good

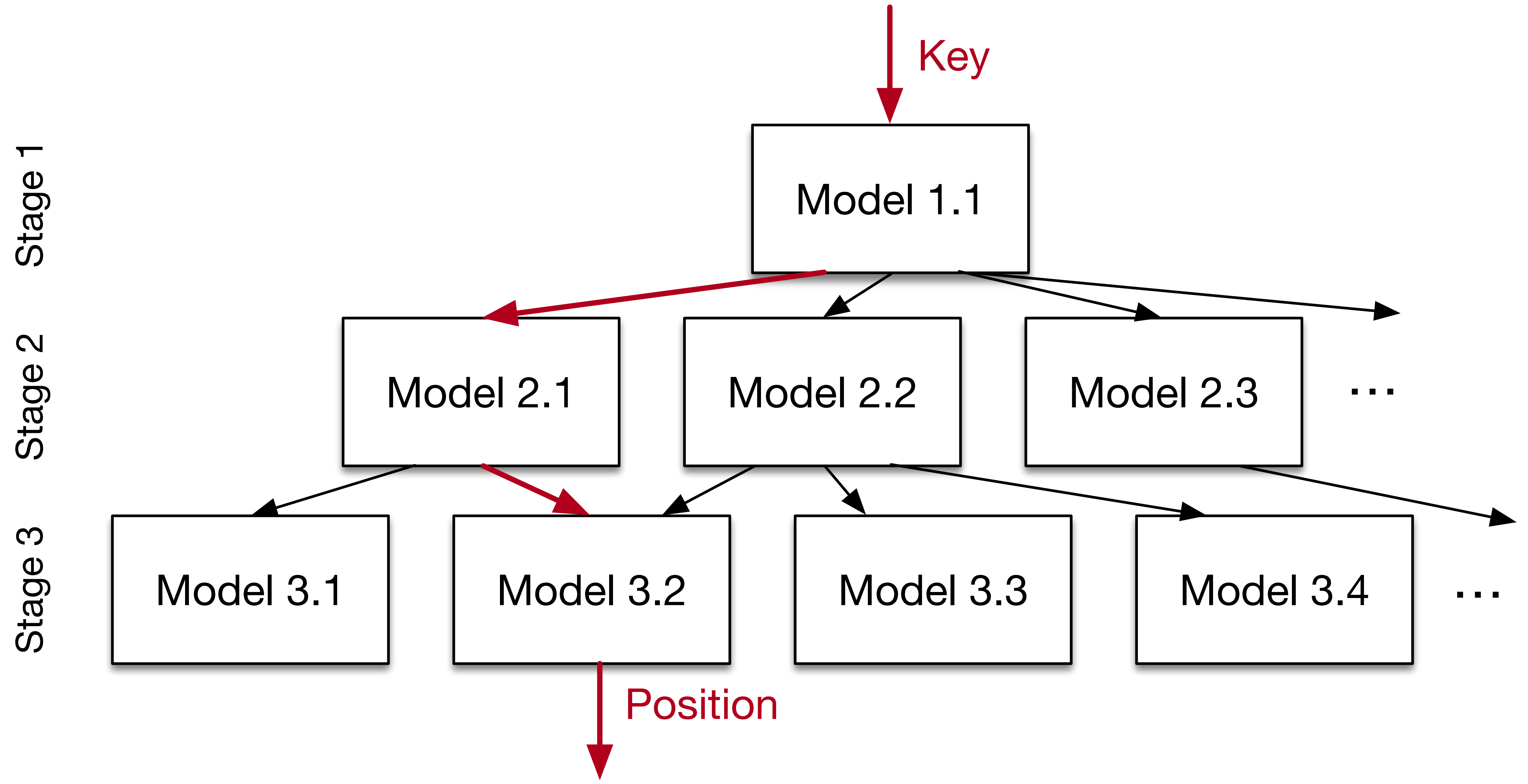



## ML+System Co-Design




[T. Kraska, 2019]

# Recursive Model Index (RMI)

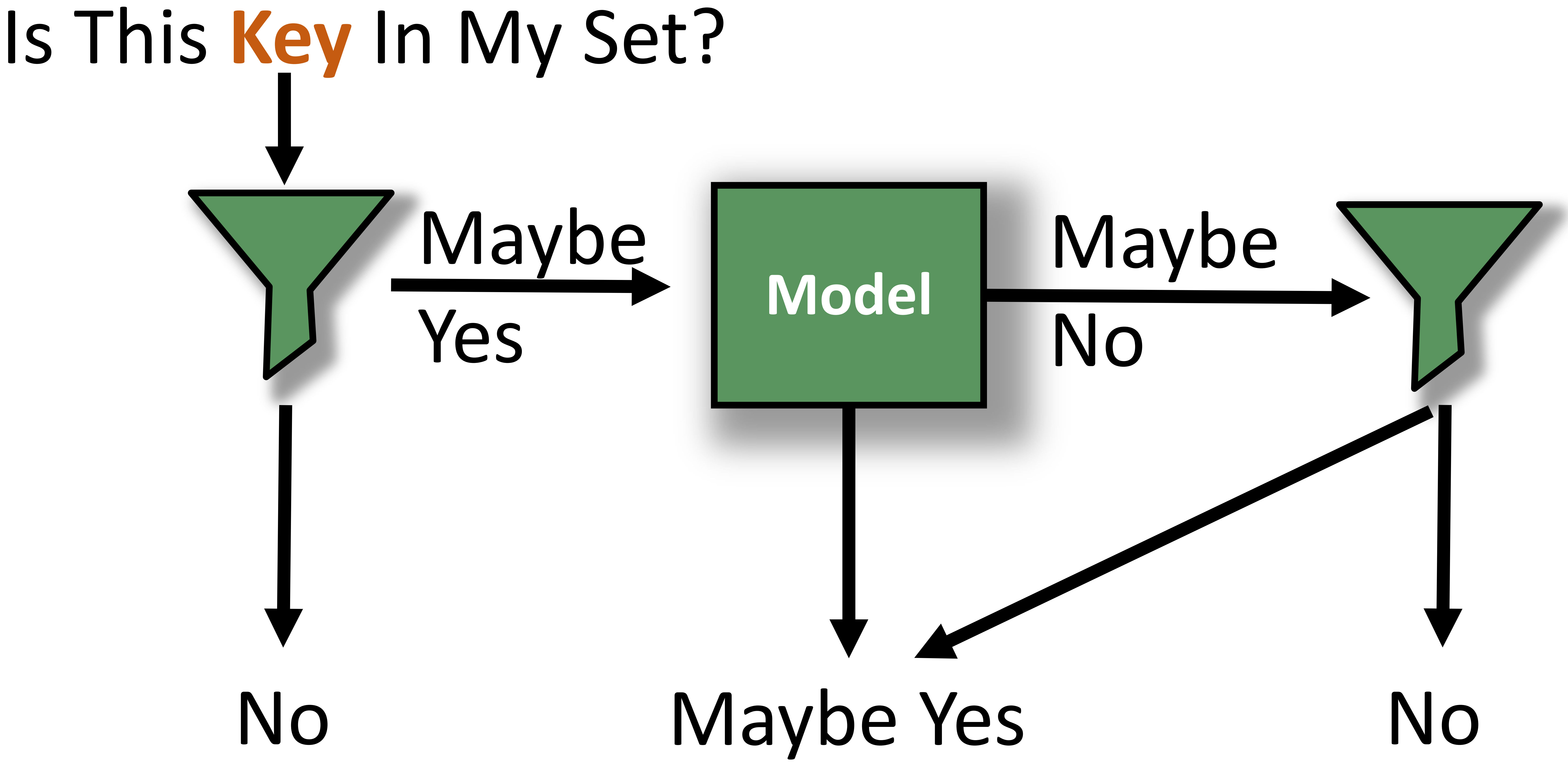



## 2-Stage RMI with Linear Model

```
pos₀ = a₀ + b₀ * key
pos₁ = m₁[pos₀].a + m₁[pos₀].b * key
record = local-search(key, pos₁)
```

[T. Kraska, 2019]

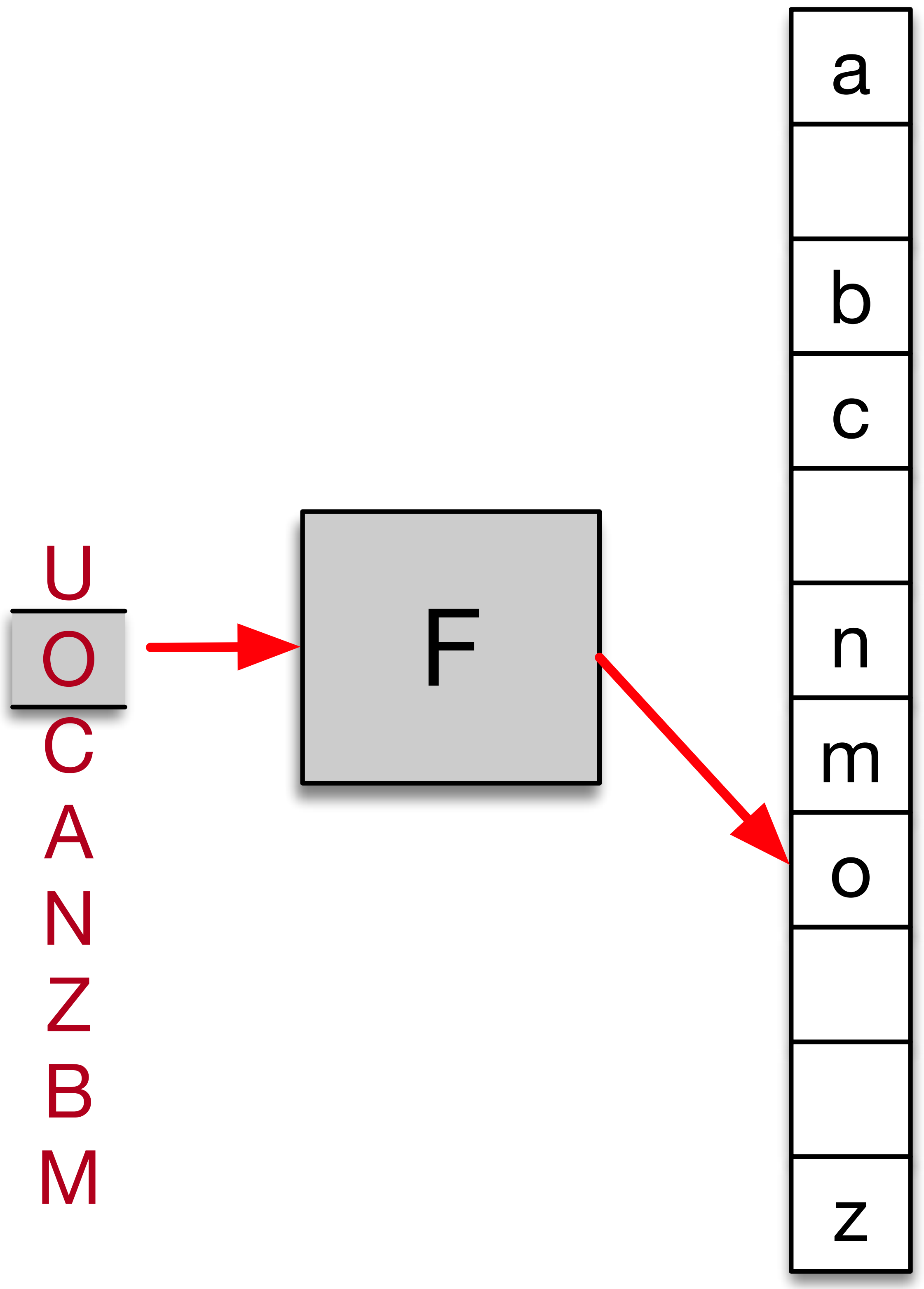
# Sandwiched Bloom Filter



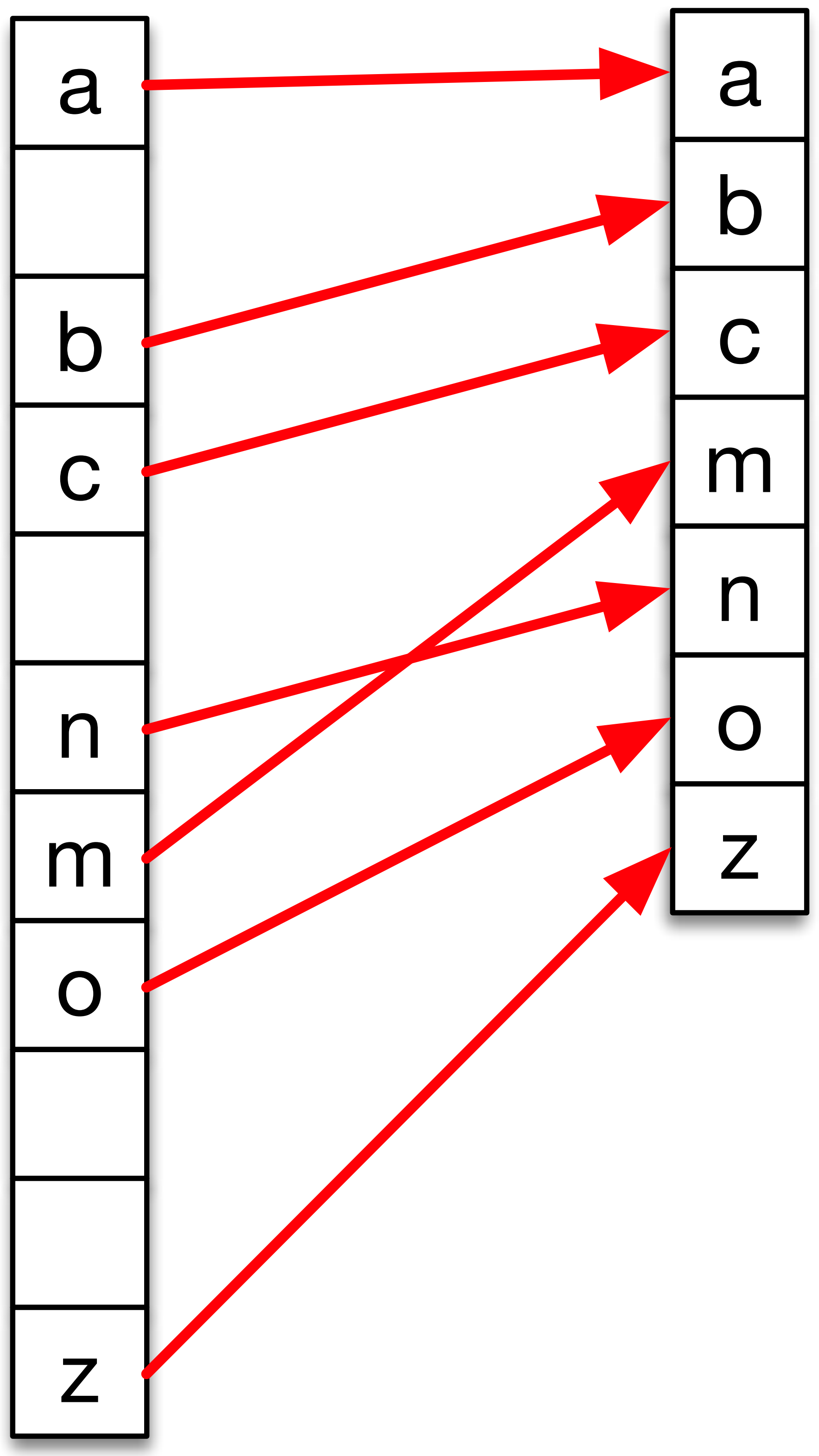

[M. Mitzenmacher, 2018 via T. Kraska, 2019]

# Sorting

(a) CDF Model Pre-Sorts



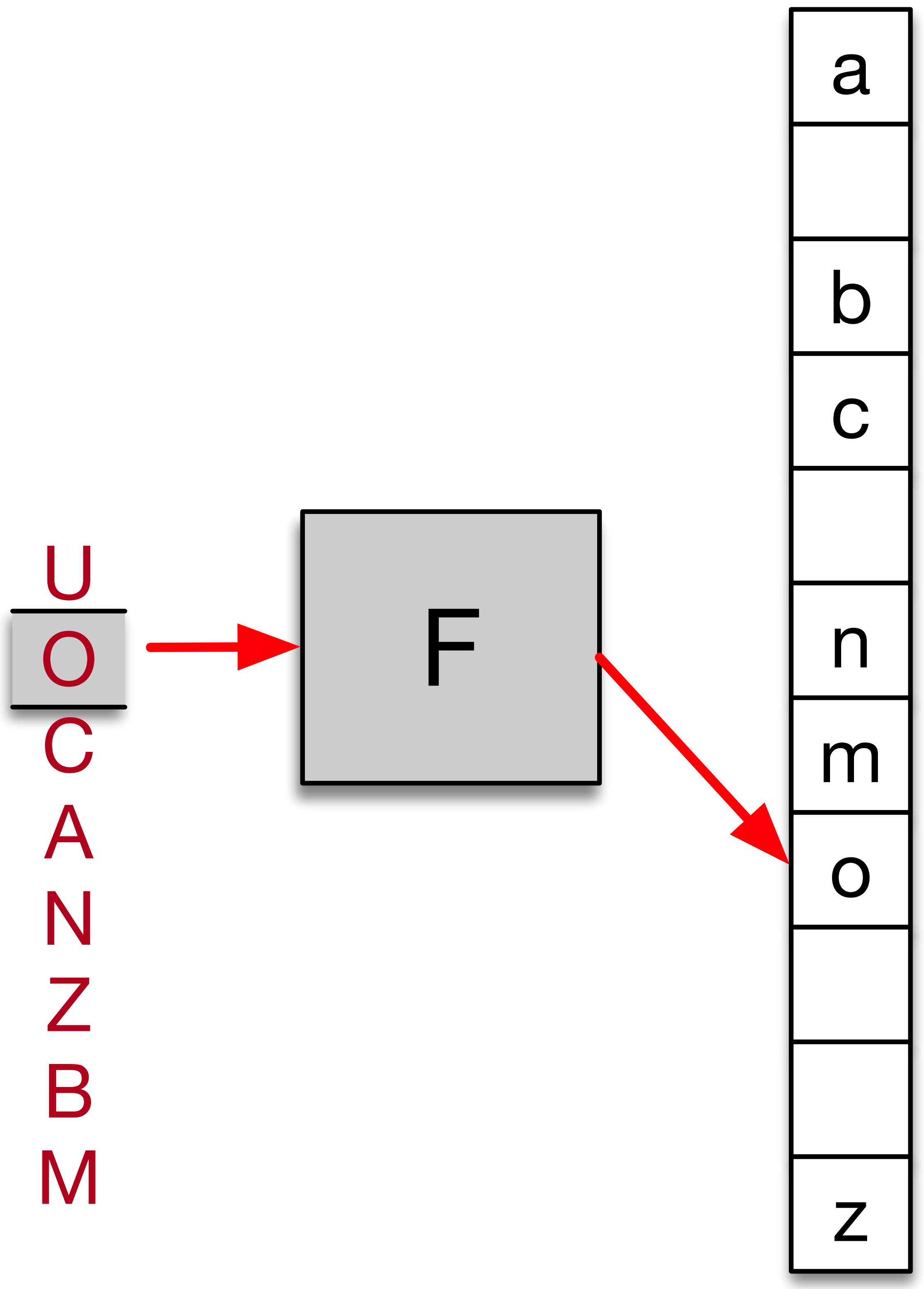

(b) Compact & local sort

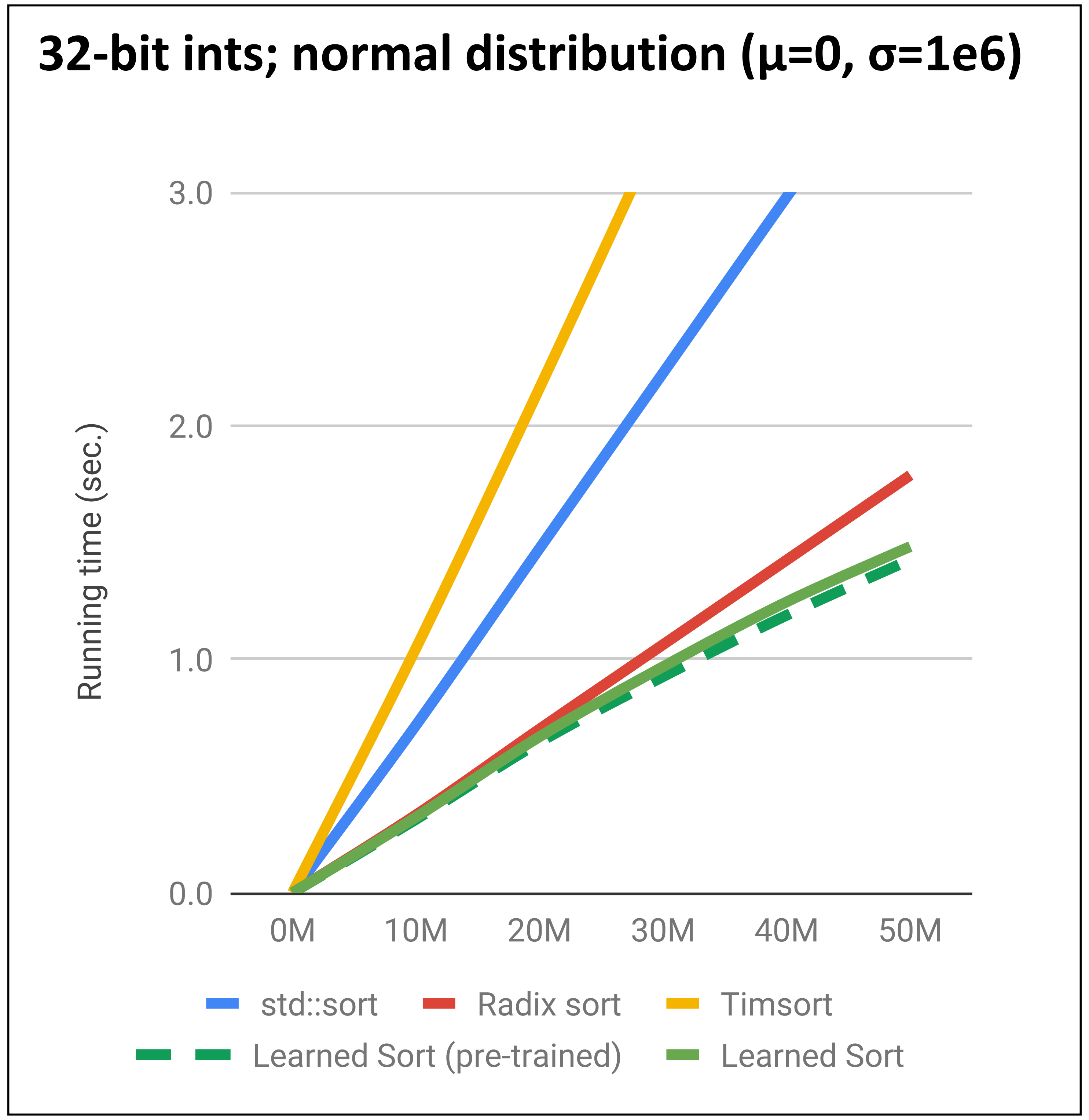



[T. Kraska, 2019]

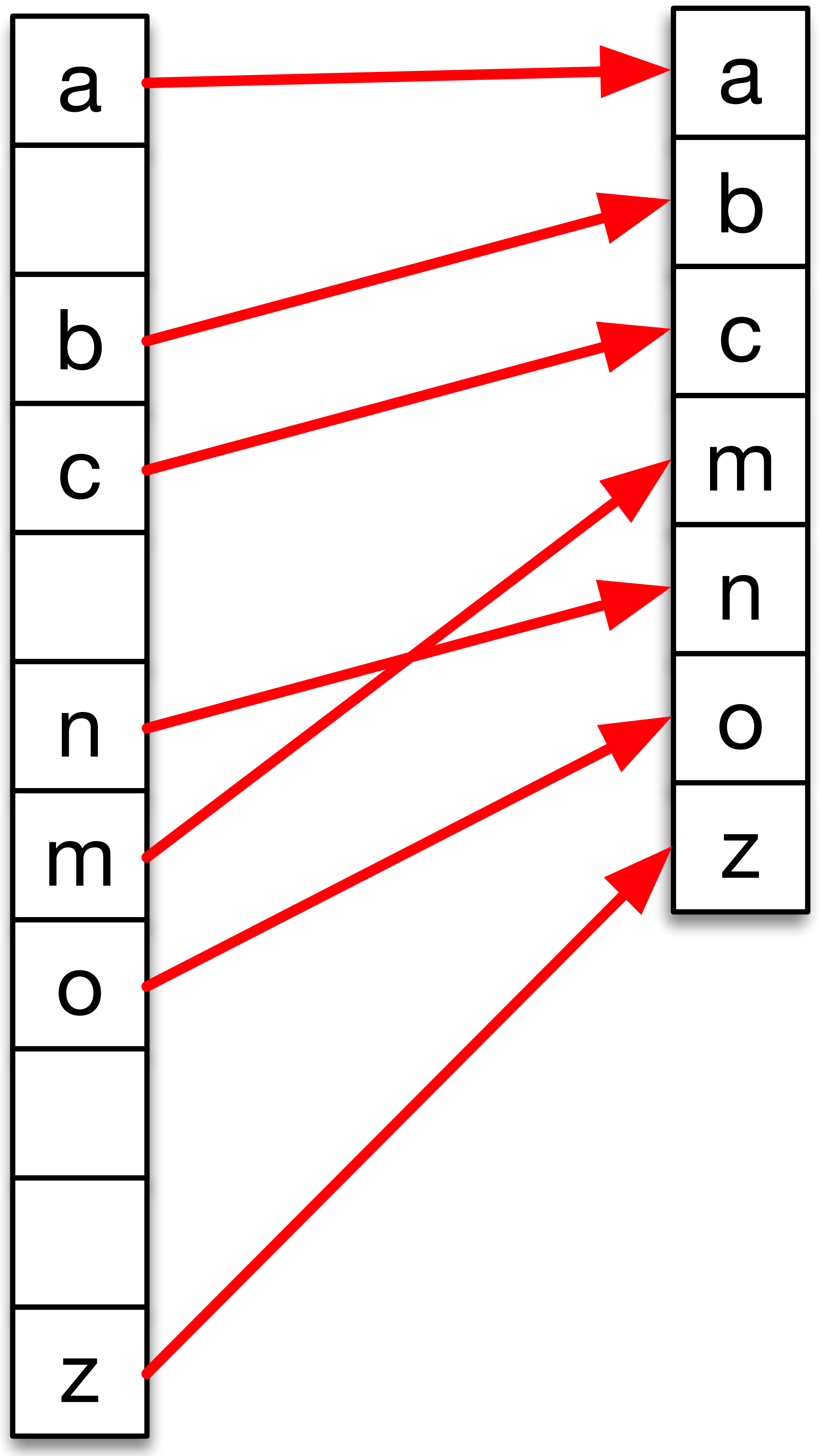
# Sorting

(a) CDF Model Pre-Sorts



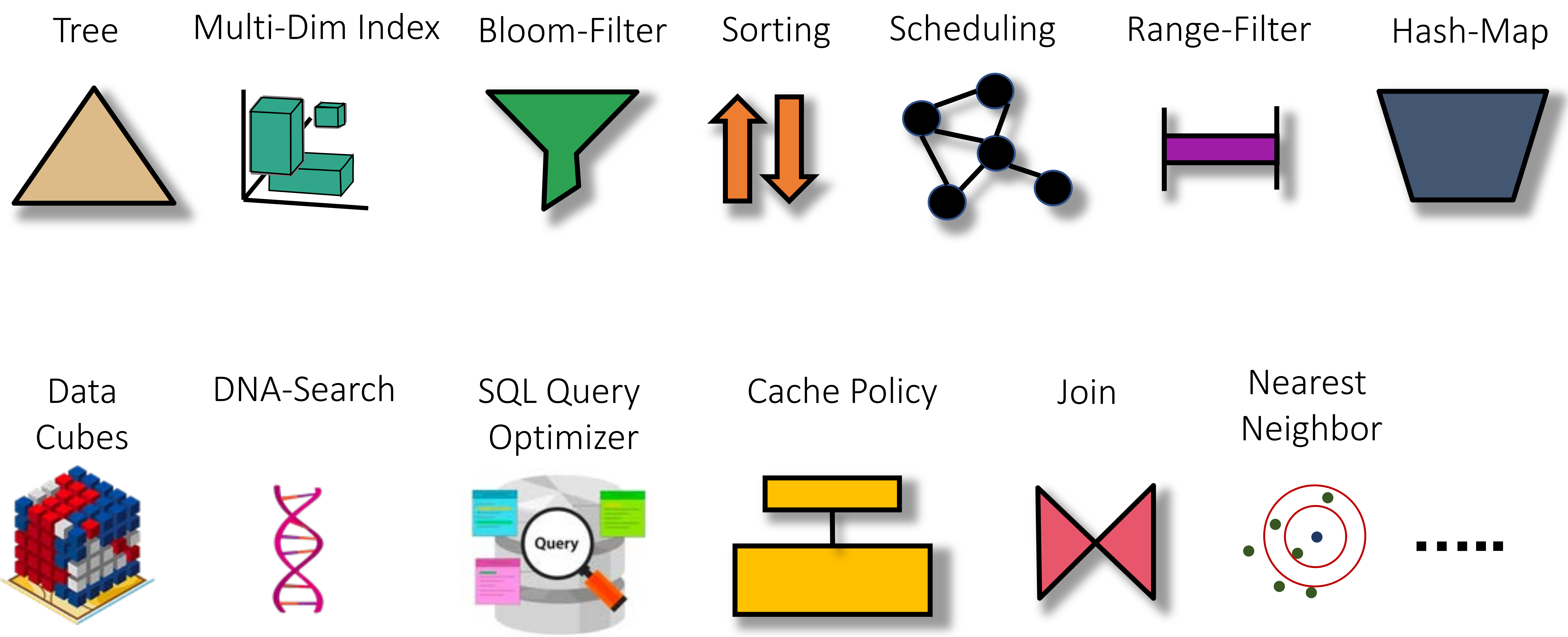




(b) Compact & local sort




[T. Kraska, 2019]

# More…




[T. Kraska, 2019]

# Assignment 5

- Four parts
  - Loading Data
  - Spatial Analysis
  - Graph Analysis
  - Temporal Analysis
- Due tomorrow
- Questions?

# Final Exam

- Monday, April 26, 4:00-5:50pm, Online (Blackboard)
- Similar format
- More comprehensive (questions from topics covered in Test 1 & 2)
- Will also have questions from temporal data, provenance, reproducibility, machine learning

Questions?

# Review

## What did we do this semester?

# What's involved in dealing with data?

| Data Acquisition | Data Analysis | Data Curation | Data Storage | Data Usage |
|---|---|---|---|---|
| • Structured data<br>• Unstructured data<br>• Event processing<br>• Sensor networks<br>• Protocols<br>• Real-time<br>• Data streams<br>• Multimodality | • Stream mining<br>• Semantic analysis<br>• Machine learning<br>• Information extraction<br>• Linked Data<br>• Data discovery<br>• 'Whole world' semantics<br>• Ecosystems<br>• Community data analysis<br>• Cross-sectorial data analysis | • Data Quality<br>• Trust / Provenance<br>• Annotation<br>• Data validation<br>• Human-Data Interaction<br>• Top-down/Bottom-up<br>• Community / Crowd<br>• Human Computation<br>• Curation at scale<br>• Incentivisation<br>• Automation<br>• Interoperability | • In-Memory DBs<br>• NoSQL DBs<br>• NewSQL DBs<br>• Cloud storage<br>• Query Interfaces<br>• Scalability and Performance<br>• Data Models<br>• Consistency, Availability, Partition-tolerance<br>• Security and Privacy<br>• Standardization | • Decision support<br>• Prediction<br>• In-use analytics<br>• Simulation<br>• Exploration<br>• Visualisation<br>• Modeling<br>• Control<br>• Domain-specific usage |

[Big Data Value Chain, Curry et al., 2014]

# Python!

- Just assign expressions to variables, no typing

```
a = 12
a = "abc"
b = a + "de"
```

- Functions defined using def, called using parenthesis:

```
def hello(name1="Joe", name2="Jane"):
    print(f"Hello {name1} and {name2}")
hello(name2="Mary")
```

- Always indent blocks (if-else-elif, while, for, etc.):

```
 z = 20
if x > 0:
    if y > 0:
        z = 100
else:
    z = 10
```

# Python Containers

- List: `[1,"abc",12.34]`
- Tuple: `(1, "abc", 12.34)`
- Indexing/Slicing:
  - `x[0], x[:-1], x[1:2], x[::2]`
- Set: `{1, "abc", 12.34}`
- Dictionary: `{'x': 1, 'y': "abc", 'z': 12.34}`
- Mutable vs. Immutable
- Stored by reference
- Iterators: objects that traverse containers, just know how to get next element
- You cannot index/slice an iterator (`d.values()[-1]` doesn't work)

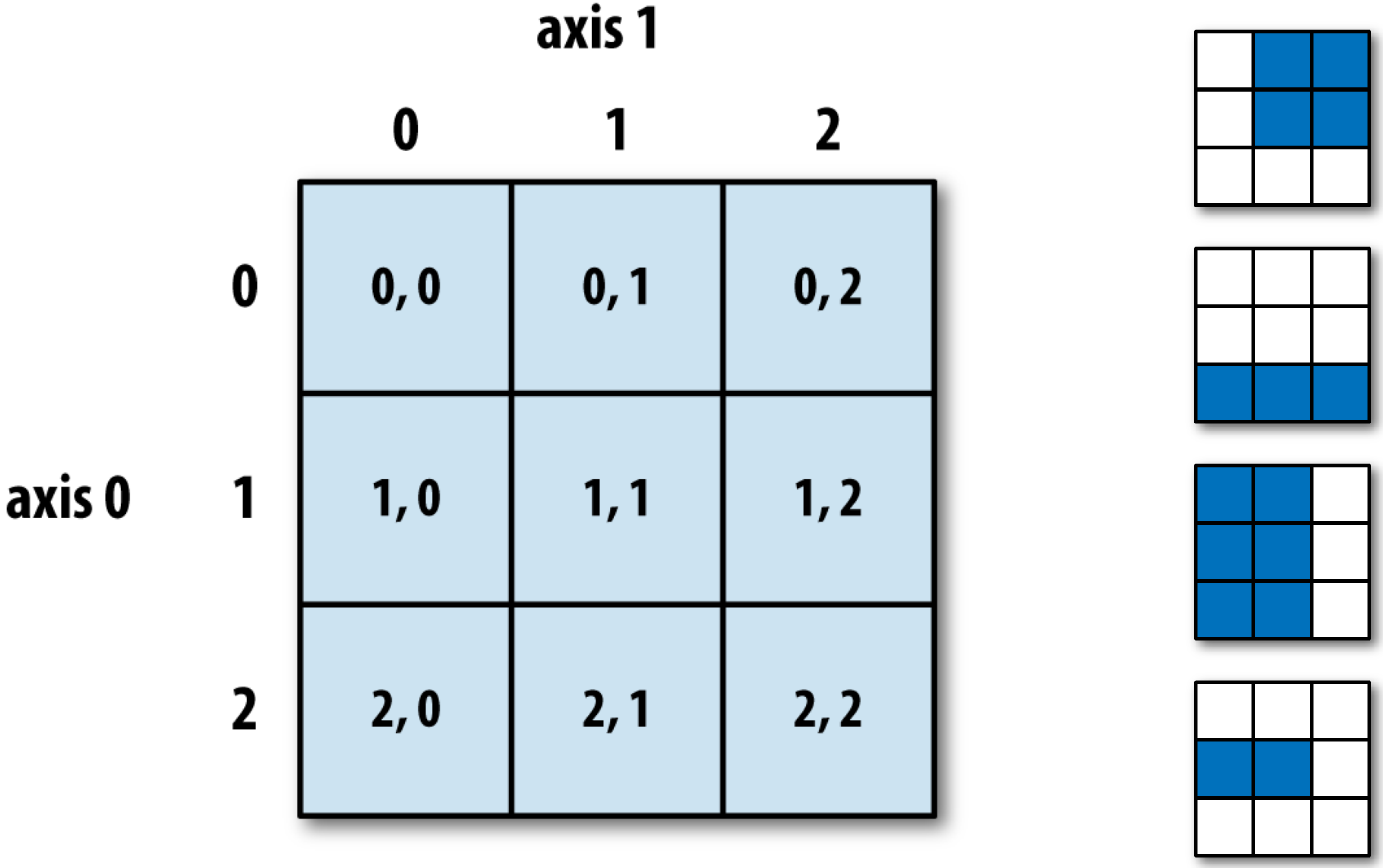
# Comprehensions

- List Comprehensions:
  - `squares = [i**2 for i in range(10)]`
- Dictionary Comprehensions:
  - `squares = {i: i**2 for i in range(10)}`
- Set Comprehensions:
  - `squares = {i**2 for i in range(10)}`
- Comprehensions allow filters:
  - `squares = [i**2 for i in range(10) if i % 2 == 0]`

# JupyterLab

- An interactive, configurable programming environment
- Supports many activities including notebooks
- Runs in your web browser
- Notebooks:
  - Originally designed for Python
  - Supports other languages, too
  - Displays results (even interactive maps) inline
  - You decide how to divide code into executable cells
  - Shift+Enter to execute a cell

# NumPy arrays and slicing




| Expression | Shape |
|---|---|
| `arr[:2, 1:]` | (2, 2) |
| `arr[2]` | (3,) |
| `arr[2, :]` | (3,) |
| `arr[2:, :]` | (1, 3) |
| `arr[:, :2]` | (3, 2) |
| `arr[1, :2]` | (2,) |
| `arr[1:2, :2]` | (1, 2) |

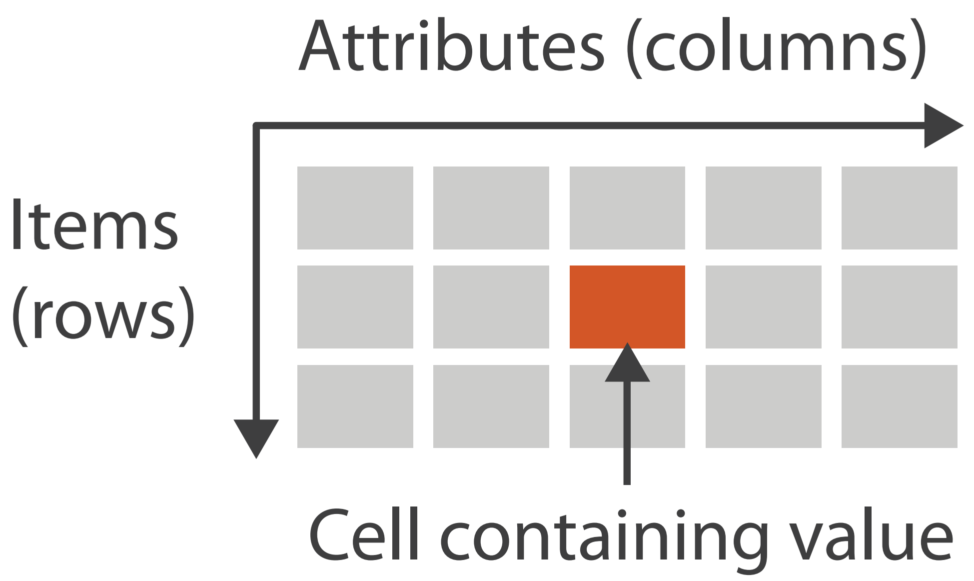
[W. McKinney, Python for Data Analysis]

# Boolean Indexing

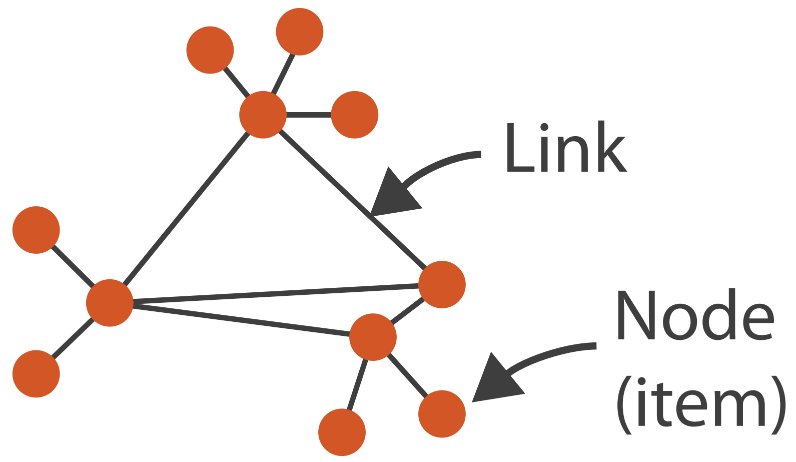
- `names == 'Bob'` gives back booleans that represent the element-wise comparison with the array `names`
- Boolean arrays can be used to index into another array:
  - `data[names == 'Bob']`
- Can even mix and match with integer slicing
- Can do boolean operations (`&`, `|`) between arrays (just like addition, subtraction)
  - `data[(names == 'Bob') | (names == 'Will')]`
- Note: `or` and `and` do not work with arrays
- We can set values too! `data[data < 0] = 0`

# What is Data?

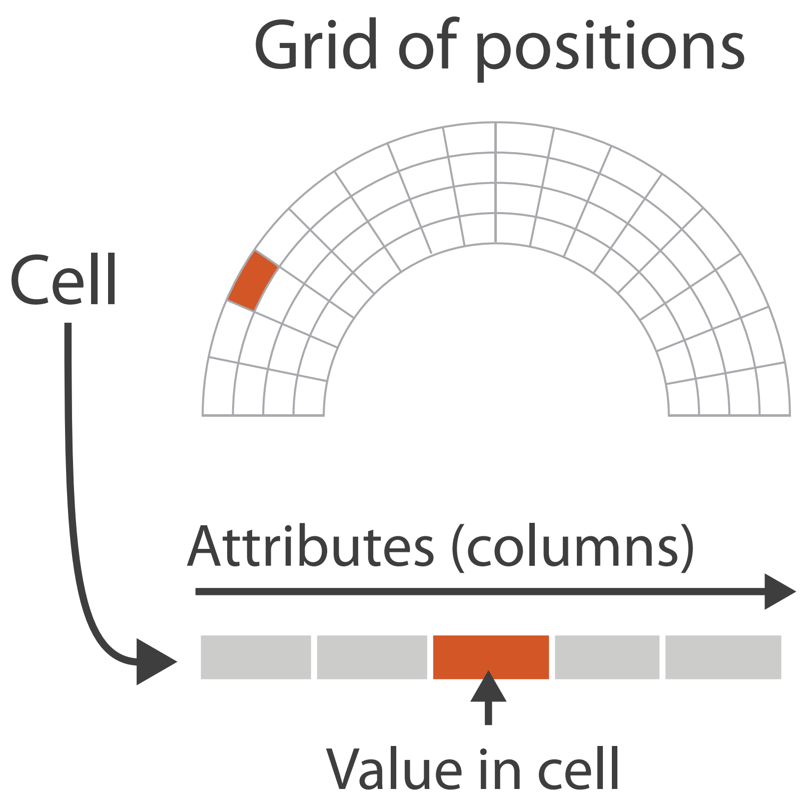
- ➔ Tables



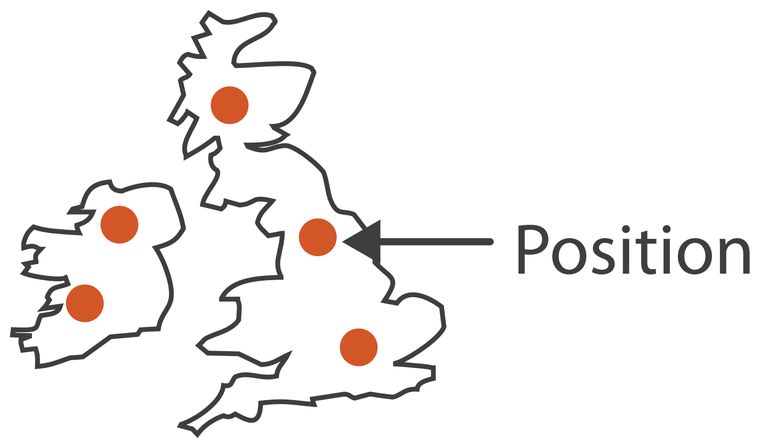

- ➔ Networks



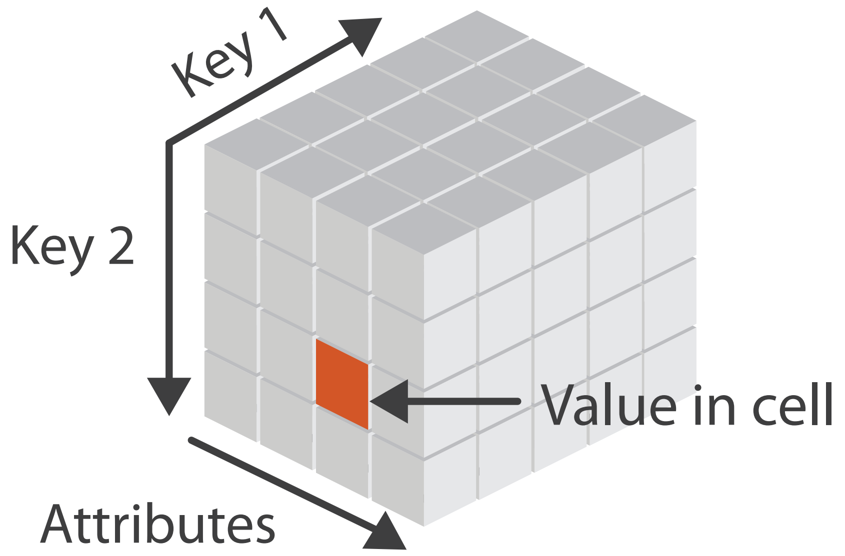

- ➔ Fields (Continuous)



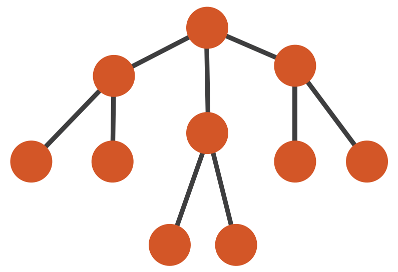

- ➔ Geometry (Spatial)




- ➔ *Multidimensional Table*




- ➔ *Trees*



[Munzner (ill. Maguire), 2014]

# Categorial, Ordinal, and Quantitative

| A | B | C | S | T | U |
|---|---|---|---|---|---|
| Order ID | Order Date | Order Priority | Product Container | Product Base Margin | Ship Date |
| 3 | 10/14/06 | 5-Low | Large Box | 0.8 | 10/21/06 |
| 6 | 2/21/08 | 4-Not Specified | Small Pack | 0.55 | 2/22/08 |
| 32 | 7/16/07 | 2-High | Small Pack | 0.79 | 7/17/07 |
| 32 | 7/16/07 | 2-High | Jumbo Box | 0.72 | 7/17/07 |
| 32 | 7/16/07 | 2-High | Medium Box | 0.6 | 7/18/07 |
| 32 | 7/16/07 | 2-High | Medium Box | 0.65 | 7/18/07 |
| 35 | 10/23/07 | 4-Not Specified | Wrap Bag | 0.52 | 10/24/07 |
| 35 | 10/23/07 | 4-Not Specified | Small Box | 0.58 | 10/25/07 |
| 36 | 11/3/07 | 1-Urgent | Small Box | 0.55 | 11/3/07 |
| 65 | 3/18/07 | 1-Urgent | Small Pack | 0.49 | 3/19/07 |
| 66 | 1/20/05 | 5-Low | Wrap Bag | 0.56 | 1/20/05 |
| 69 | 6/4/05 | 4-Not Specified | Small Pack | 0.44 | 6/6/05 |
| 69 | 6/4/05 | 4-Not Spec |  | 0.6 | 6/6/05 |
| 70 | 12/18/06 | 5-Low |  | 0.59 | 12/23/06 |
| 70 | 12/18/06 | 5-Low |  | 0.82 | 12/23/06 |
| 96 | 4/17/05 | 2-High |  | 0.55 | 4/19/05 |
| 97 | 1/29/06 | 3-Medium |  | 0.38 | 1/30/06 |
| 129 | 11/19/08 | 5-Low |  | 0.37 | 11/28/08 |
| 130 | 5/8/08 | 2-High | Small Box | 0.37 | 5/9/08 |
| 130 | 5/8/08 | 2-High | Medium Box | 0.38 | 5/10/08 |
| 130 | 5/8/08 | 2-High | Small Box | 0.6 | 5/11/08 |
| 132 | 6/11/06 | 3-Medium | Medium Box | 0.6 | 6/12/06 |
| 132 | 6/11/06 | 3-Medium | Jumbo Box | 0.69 | 6/14/06 |
| 134 | 5/1/08 | 4-Not Specified | Large Box | 0.82 | 5/3/08 |
| 135 | 10/21/07 | 4-Not Specified | Small Pack | 0.64 | 10/23/07 |
| 166 | 9/12/07 | 2-High | Small Box | 0.55 | 9/14/07 |
| 193 | 8/8/06 | 1-Urgent | Medium Box | 0.57 | 8/10/06 |
| 194 | 4/5/08 | 3-Medium | Wrap Bag | 0.42 | 4/7/08 |

**quantitative**
**ordinal**
**categorical**

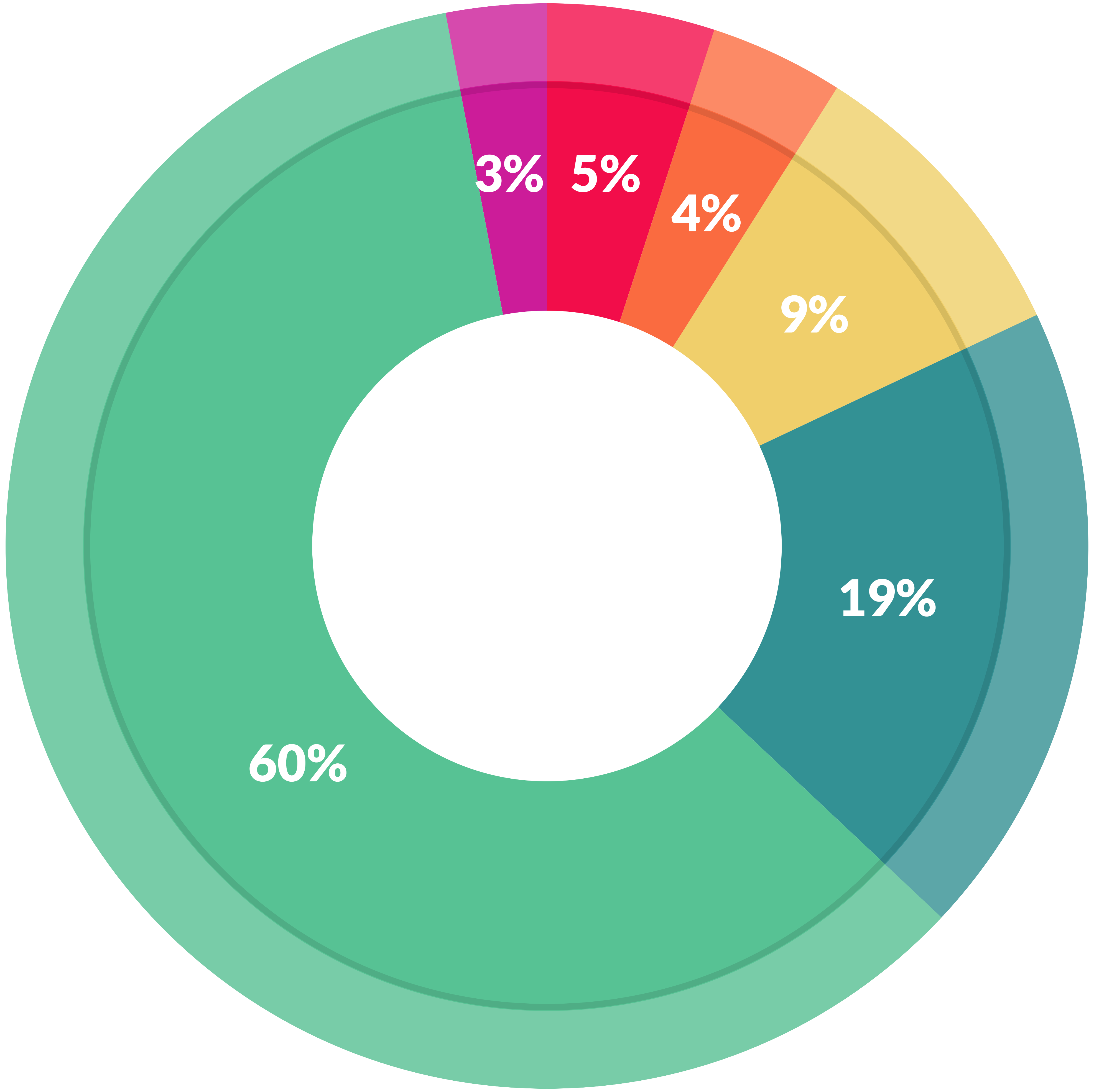
# Pandas and Data Frames

| | Inspection ID | DBA Name | AKA Name | License # | Facility Type | Risk | Address | City | State | Zip | Inspection Date | Inspection Type | Results |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2356580 | UNCOOKED LLC | UNCOOKED LLC | 2709319.0 | NaN | All | 210 N CARPENTER ST | CHICAGO | IL | 60607.0 | 01/13/2020 | License | Not Ready |
| **1** | 2356551 | MOJO 33 NORTH LASALLE LLC | MOJO 33 NORTH LASALLE LLC | 2689550.0 | Restaurant | Risk 1 (High) | 33 N LA SALLE ST | CHICAGO | IL | 60602.0 | 01/13/2020 | License Re-Inspection | Pass |
| **2** | 2356492 | LA BIZNAGA #2 | LA BIZNAGA #2 | 2708992.0 | NaN | Risk 1 (High) | 2949 W BELMONT AVE | CHICAGO | IL | 60618.0 | 01/10/2020 | License | Not Ready |
| **3** | 2356432 | LAS TABLAS | LAS TABLAS | 1617900.0 | Restaurant | Risk 1 (High) | 4920 W IRVING PARK RD | CHICAGO | IL | 60641.0 | 01/09/2020 | Canvass | Pass |
| **4** | 2356423 | GIORDANO'S OF BEVERLY | GIORDANO'S OF BEVERLY | 2074456.0 | Restaurant | Risk 1 (High) | 9613 S WESTERN AVE | CHICAGO | IL | 60643.0 | 01/09/2020 | Canvass | Pass |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **199687** | 112321 | PANDA EXPRESS #236 | PANDA EXPRESS #236 | 1801495.0 | Restaurant | Risk 1 (High) | 77 W JACKSON BLVD | CHICAGO | IL | 60604.0 | 02/18/2010 | Suspected Food Poisoning | Pass |
| **199688** | 74300 | KENNYS RIBS & CHICKEN | UNCLE JOE'S | 81030.0 | Restaurant | Risk 1 (High) | 1453 E HYDE PARK BLVD | CHICAGO | IL | 60615.0 | 02/08/2010 | Complaint | Pass |
| **199689** | 70314 | Cafe Marbella | Cafe Marbella | 2016764.0 | Restaurant | Risk 1 (High) | 5527-5531 N Milwaukee AVE | CHICAGO | IL | 60630.0 | 01/28/2010 | License Re-Inspection | Pass |
| **199690** | 78309 | WALGREENS # 07876 | WALGREENS # 07876 | 2004292.0 | Grocery Store | Risk 3 (Low) | 7544 S STONY ISLAND AVE | CHICAGO | IL | 60649.0 | 02/18/2010 | TASK FORCE LIQUOR 1474 | Pass |
| **199691** | 150209 | YSABEL'S FILIPINO CUISINE | YSABEL'S GRILL ASIAN CUISINE | 2013419.0 | Restaurant | Risk 1 (High) | 4908 W Irving Park RD | CHICAGO | IL | 60641.0 | 01/12/2010 | License Re-Inspection | Pass |

199692 rows × 17 columns

- Data Frames are tables with many database-like operations
- Index shared across all columns
- Can select, project, merge (join), and more
- Read and write many file formats

# How do data scientists spend their time?

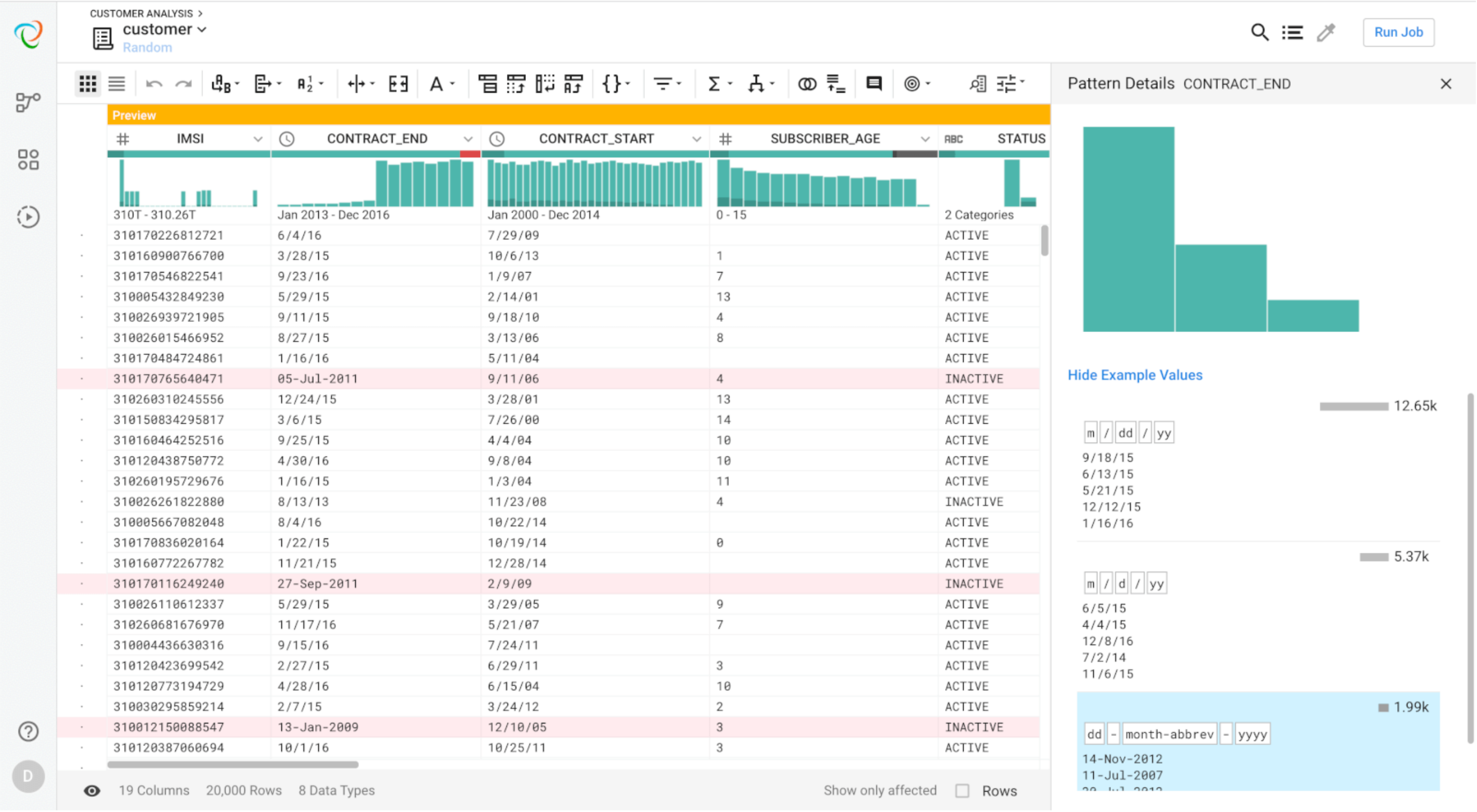



## What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

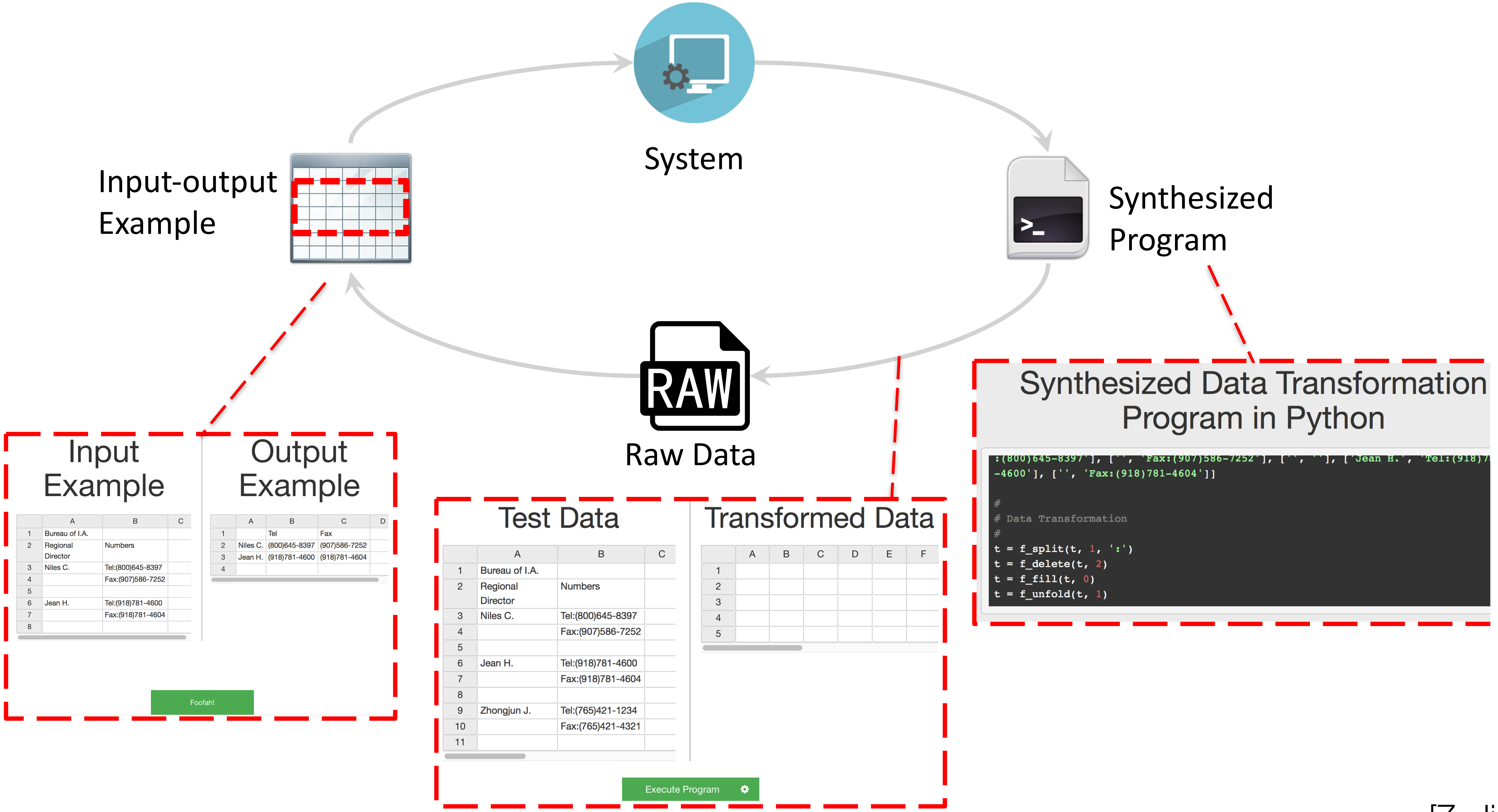
[CrowdFlower Data Science Report, 2016]

# Data Wrangling




[Trifacta]

# Foofah: Programming by Example



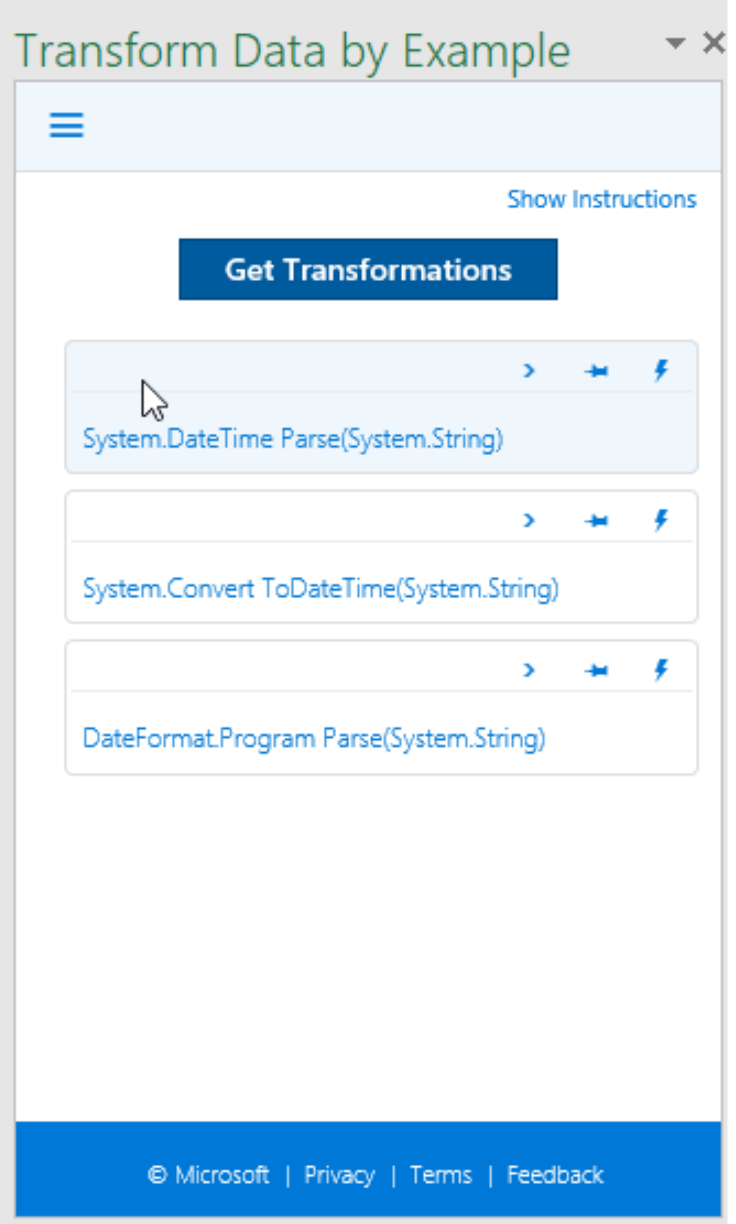

[Z. Jin et al., 2017]

# TDE: Transform Data by Example

| C | D |
|---|---|
| **Transaction Date** | **output** |
| Wed, 12 Jan 2011 | 2011-01-12-Wednesday |
| Thu, 15 Sep 2011 | 2011-09-15-Thursday |
| Mon, 17 Sep 2012 | |
| 2010-Nov-30 11:10:41 | |
| 2011-Jan-11 02:27:21 | |
| 2011-Jan-12 | |
| 2010-Dec-24 | |
| 9/22/2011 | |
| 7/11/2012 | |
| 2/12/2012 | |

→

| C | D |
|---|---|
| **Transaction Date** | **output** |
| Wed, 12 Jan 2011 | 2011-01-12-Wednesday |
| Thu, 15 Sep 2011 | 2011-09-15-Thursday |
| Mon, 17 Sep 2012 | 2012-09-17-Monday |
| 2010-Nov-30 11:10:41 | 2010-11-30-Tuesday |
| 2011-Jan-11 02:27:21 | 2011-01-11-Tuesday |
| 2011-Jan-12 | 2011-01-12-Wednesday |
| 2010-Dec-24 | 2010-12-24-Friday |
| 9/22/2011 | 2011-09-22-Thursday |
| 7/11/2012 | 2012-07-11-Wednesday |
| 2/12/2012 | 2012-02-12-Sunday |




[Y. He et al., 2018]

# Tidy Data

| | treatmenta | treatmentb |
|---|---|---|
| John Smith | — | 2 |
| Jane Doe | 16 | 11 |
| Mary Johnson | 3 | 1 |

Initial Data

| | John Smith | Jane Doe | Mary Johnson |
|---|---|---|---|
| treatmenta | — | 16 | 3 |
| treatmentb | 2 | 11 | 1 |

Transpose

| name | trt | result |
|---|---|---|
| John Smith | a | — |
| Jane Doe | a | 16 |
| Mary Johnson | a | 3 |
| John Smith | b | 2 |
| Jane Doe | b | 11 |
| Mary Johnson | b | 1 |

Tidy Data

[H. Wickham, 2014]

# MultiIndex Row Access and Slicing

- `df.loc[("Boston", 2007)]` or sometimes `df.loc["Boston", 2007]`
- Remember that `loc` uses the index values, `iloc` uses integers
- **Note**: `df.iloc[0]` gets the first row, **not** `df.iloc[0,0]`
- Can get a subset of the data using partial indices
  - `df.loc["Boston"]` returns both 2007 and 2008 data
- What about slicing?
  - `df.loc["Boston":"Cleveland"]` → ERROR! (Need sorted data)
  - `df = df.sort_index()`
  - `df.loc["Boston":"Cleveland"]` → inclusive!
  - `df.loc[(slice("Boston","Cleveland"),2007),:]`

# Merges (aka Joins)

- Need to merge data from one DataFrame with data from another DataFrame
- Example: Football game data merged with temperature data

**Game**

| Id | Location | Date | Home | Away |
|---|---|---|---|---|
| 0 | Boston | 9/2 | 1 | 15 |
| 1 | Boston | 9/9 | 1 | 7 |
| 2 | Cleveland | 9/16 | 12 | 1 |
| 3 | San Diego | 9/23 | 21 | 1 |

**Weather**

| wId | City | Date | Temp |
|---|---|---|---|
| 0 | Boston | 9/2 | 72 |
| 1 | Boston | 9/3 | 68 |
| … | … | … | … |
| 7 | Boston | 9/9 | 75 |
| … | … | … | … |
| 21 | Boston | 9/23 | 54 |
| … | … | … | … |
| 36 | Cleveland | 9/16 | 81 |

No data for San Diego

# Inner Strategy

**Merged**

| Id | Location | Date | Home | Away | Temp | wId |
|---|---|---|---|---|---|---|
| 0 | Boston | 9/2 | 1 | 15 | 72 | 0 |
| 1 | Boston | 9/9 | 1 | 7 | 75 | 7 |
| 2 | Cleveland | 9/16 | 12 | 1 | 81 | 36 |

No San Diego entry

# Outer Strategy

**Merged**

| Id | Location | Date | Home | Away | Temp | wId |
|---|---|---|---|---|---|---|
| 0 | Boston | 9/2 | 1 | 15 | 72 | 0 |
| NaN | Boston | 9/3 | NaN | NaN | 68 | 1 |
| … | … | … | … | … | … | … |
| 1 | Boston | 9/9 | 1 | 7 | 75 | 7 |
| NaN | Boston | 9/10 | NaN | NaN | 76 | 8 |
| … | … | … | … | … | … | … |
| NaN | Cleveland | 9/2 | NaN | NaN | 61 | 22 |
| … | … | … | … | … | … | … |
| 2 | Cleveland | 9/16 | 12 | 1 | 81 | 36 |
| … | … | … | … | … | … | … |
| 3 | San Diego | 9/23 | 21 | 1 | NaN | NaN |

# Data Integration

```
select title, startTime
from Movie, Plays
where Movie.title=Plays.movie AND
          location="New York"  AND
          director="Woody Allen"
```

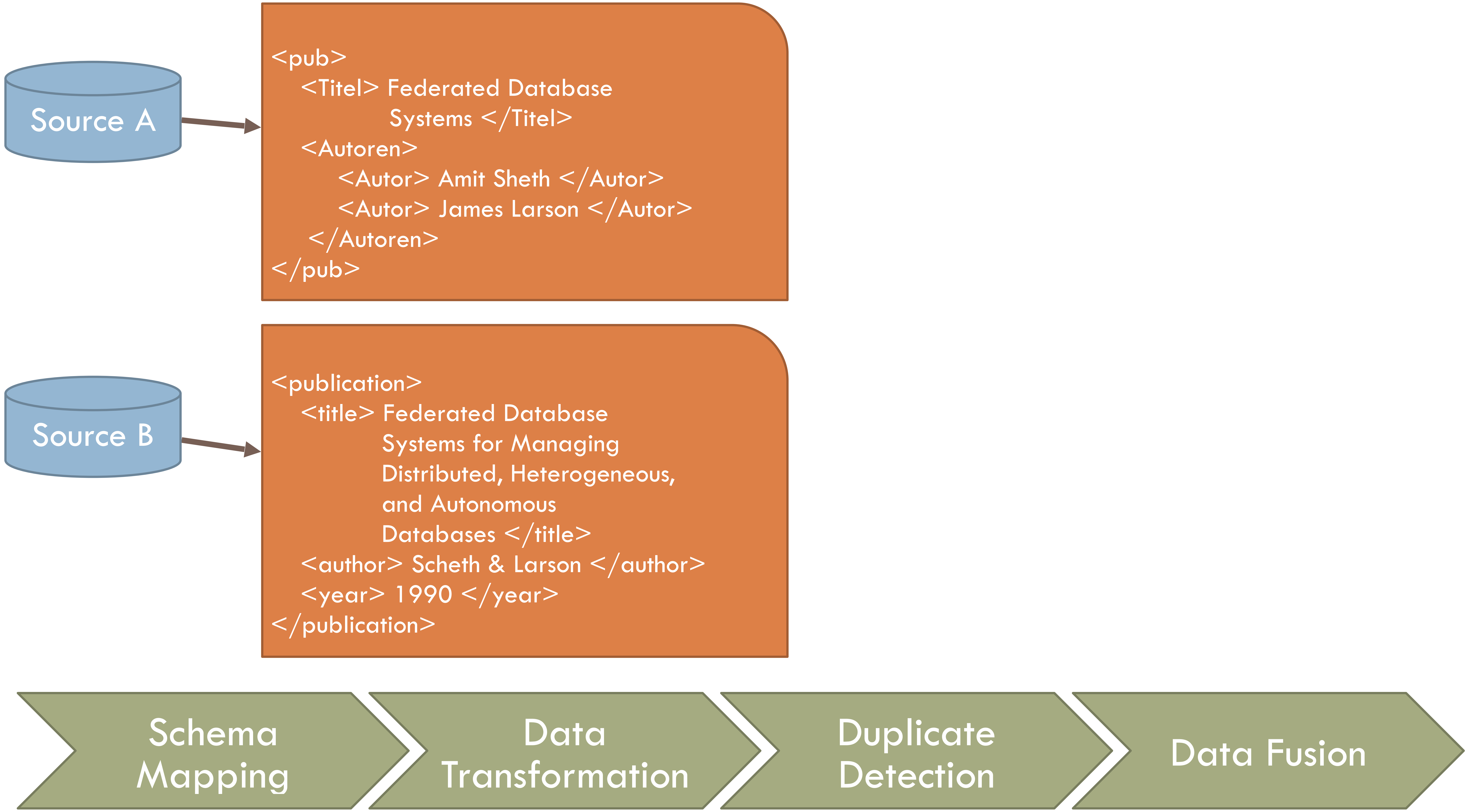
**Movie**: Title, director, year, genre
**Actors**: title, actor
**Plays**: movie, location, startTime
**Reviews**: title, rating, description

Sources S1 and S3 are relevant, sources S4 and S5 are irrelevant, and source S2 is relevant but possibly redundant.

| S1 | S2 | S3 | S4 | S5 |
|---|---|---|---|---|
| Movies: name, actors, director, genre | Cinemas: place, movie, start | Cinemas in NYC: cinema, title, startTime | Cinemas in SF: location, movie, startingTime | Reviews: title, date grade, review |

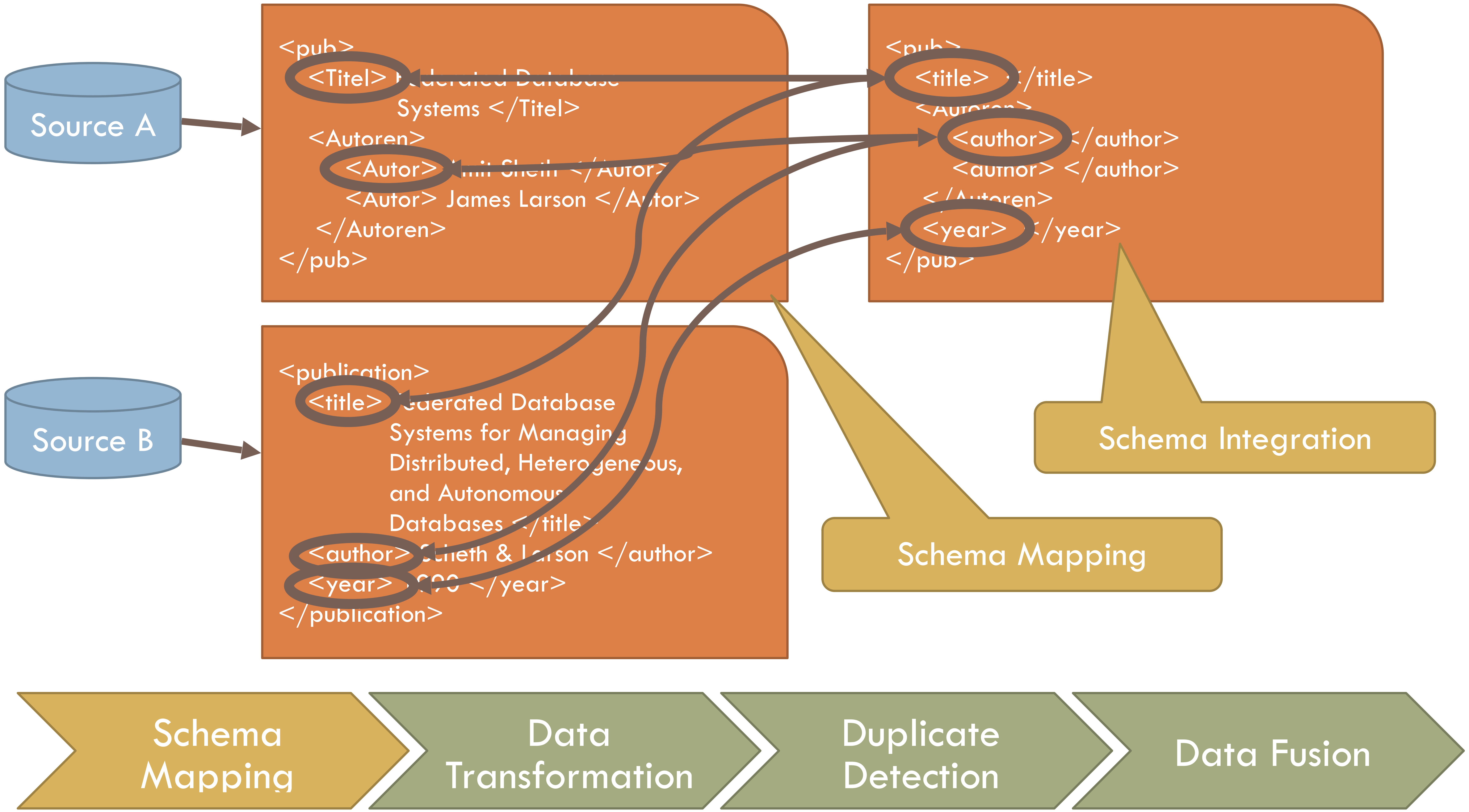
[AH Doan et al., 2012]

# Information Integration



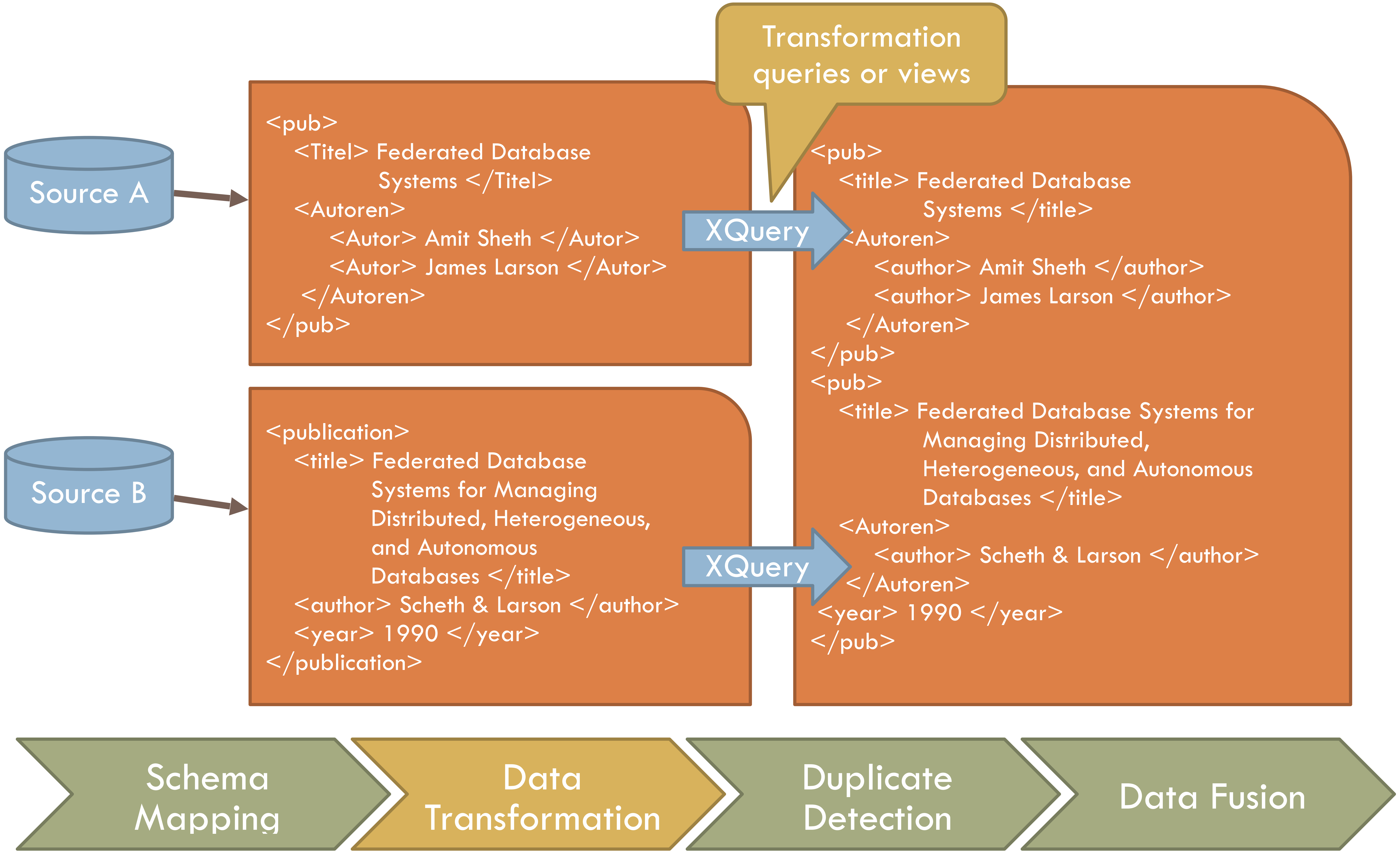

[L. Dong and F. Naumann, 2009]

# Information Integration



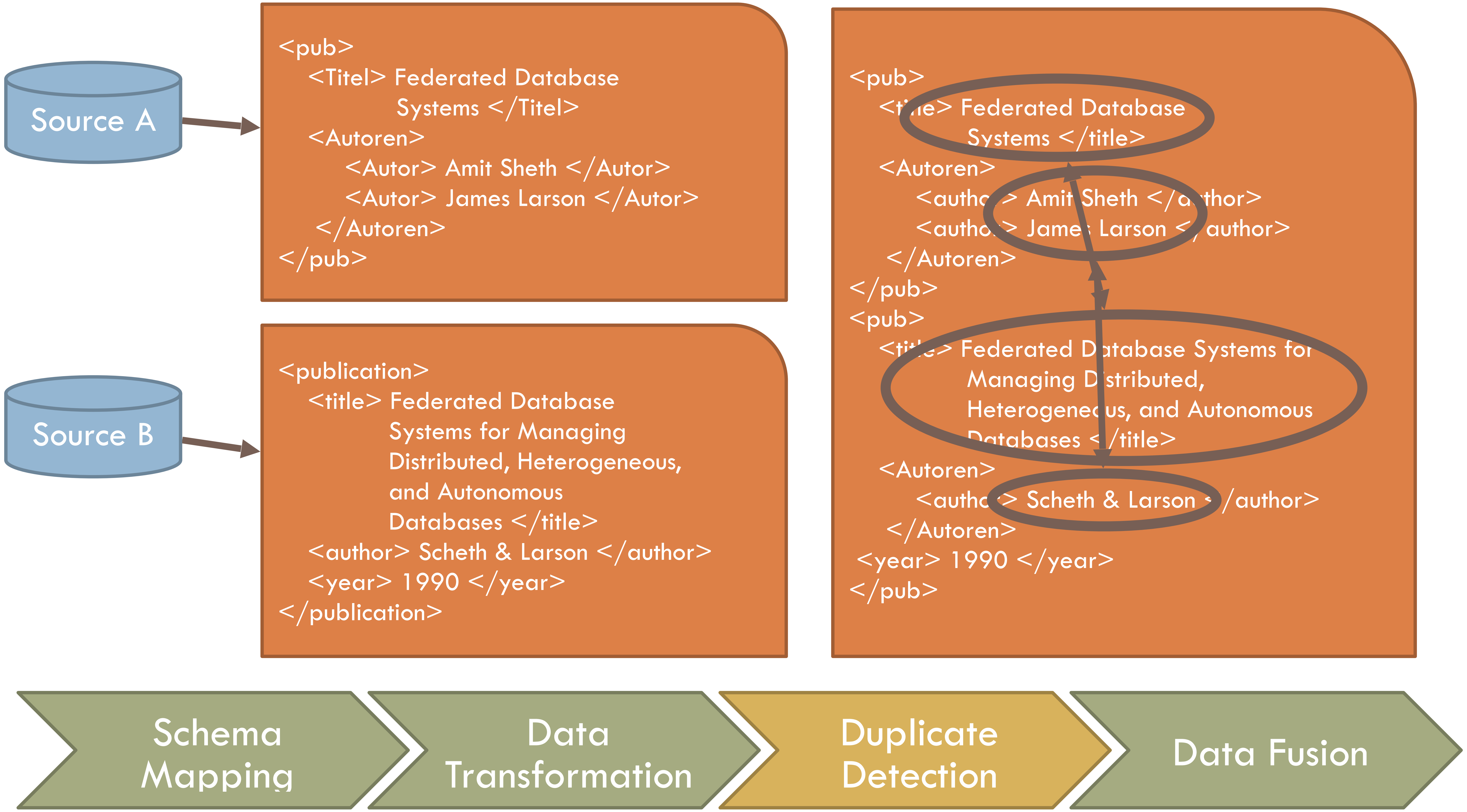

[L. Dong and F. Naumann, 2009]

# Information Integration



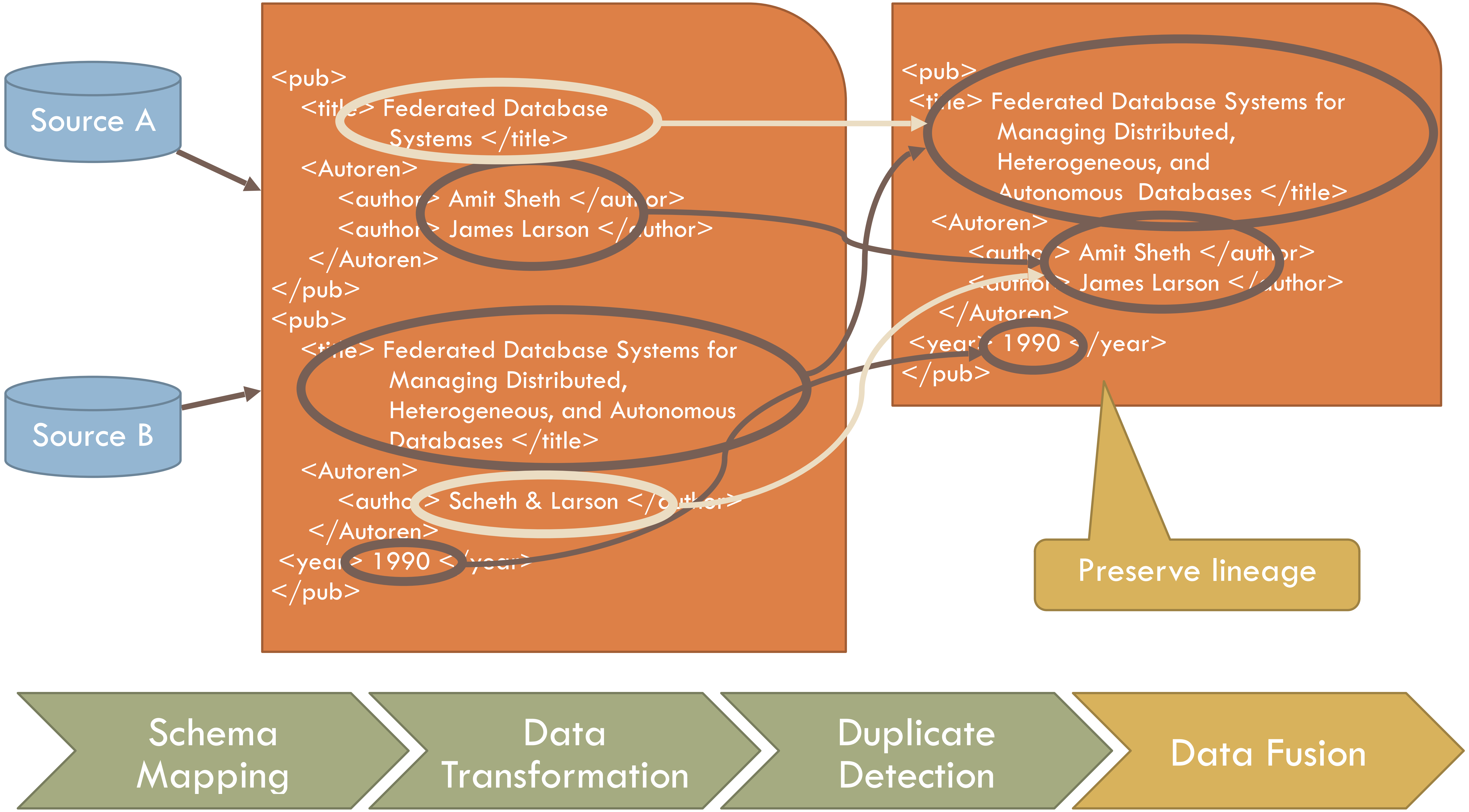

[L. Dong and F. Naumann, 2009]

# Information Integration



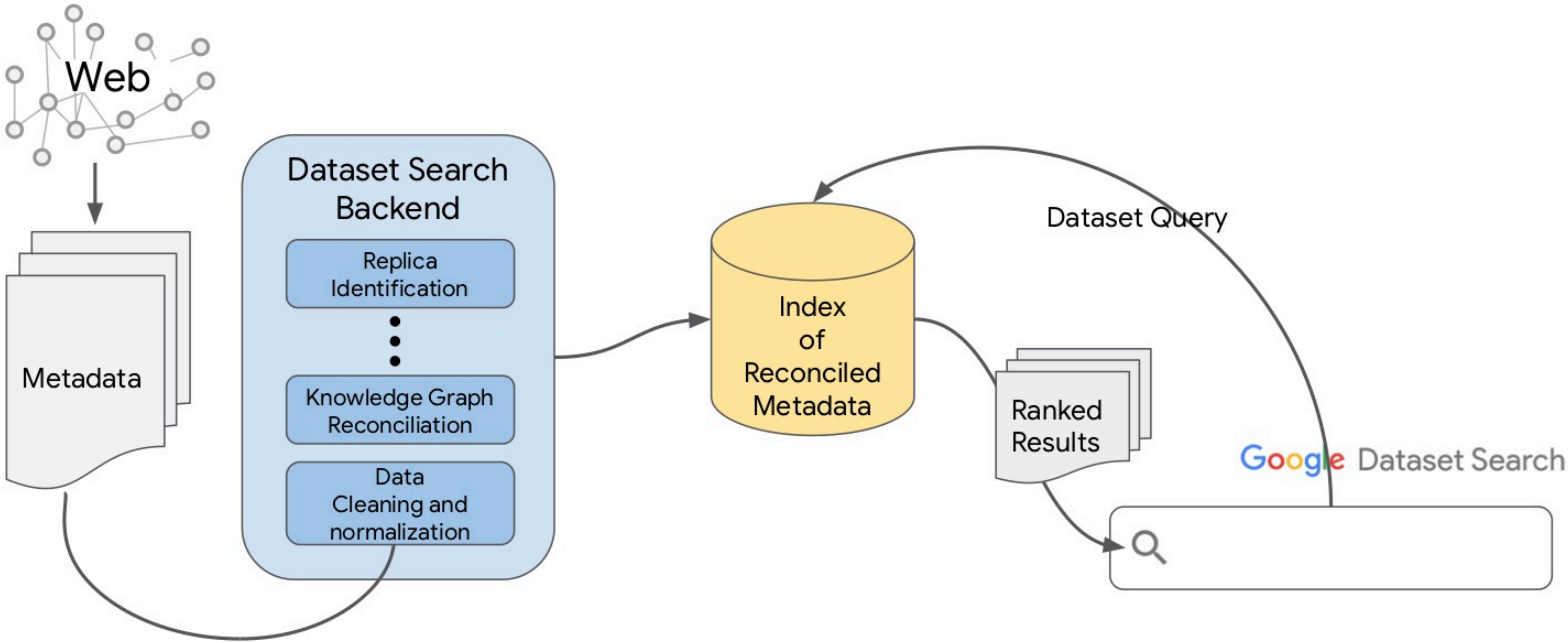

[L. Dong and F. Naumann, 2009]

# Information Integration



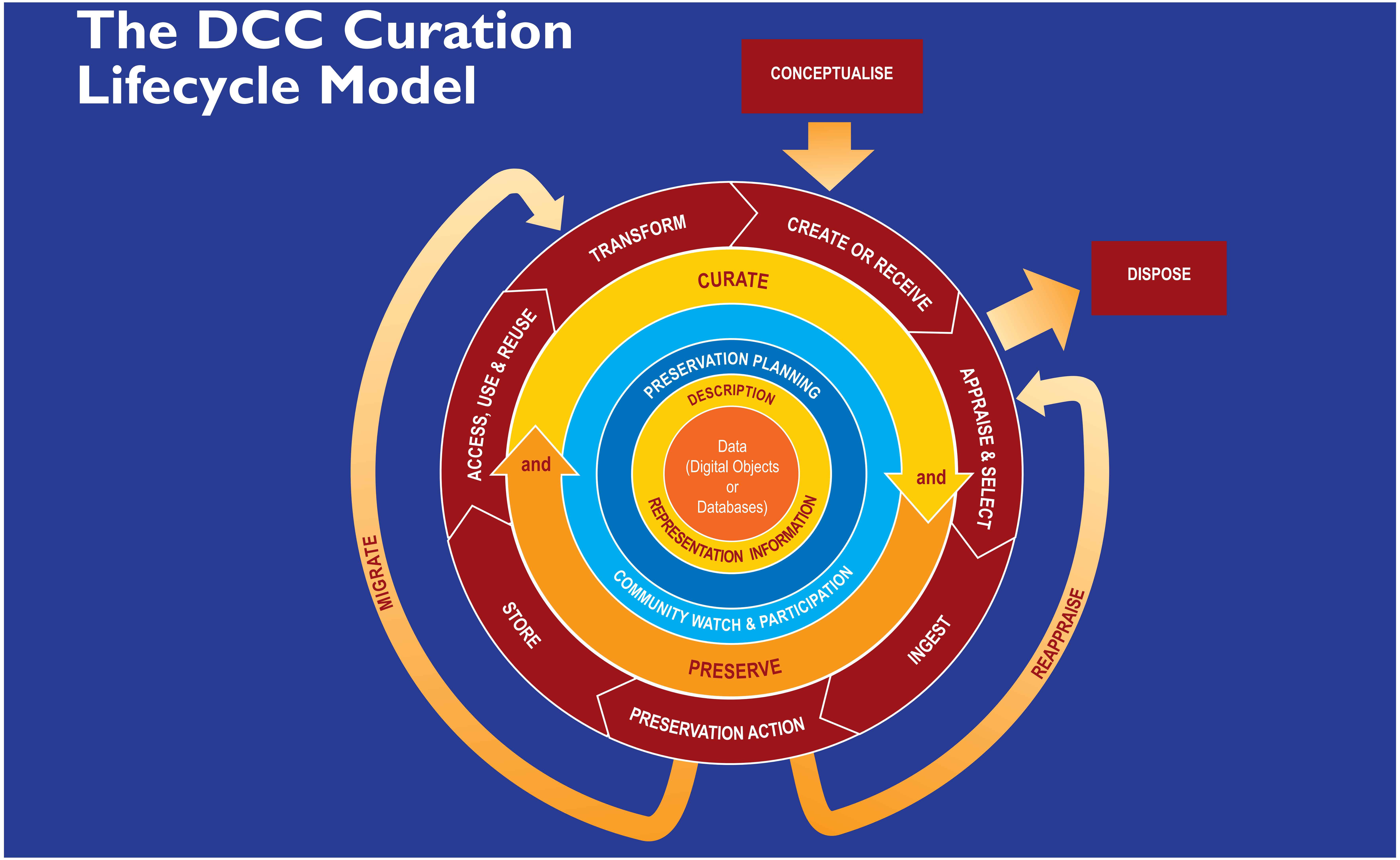

[L. Dong and F. Naumann, 2009]

# Google Dataset Search Overview



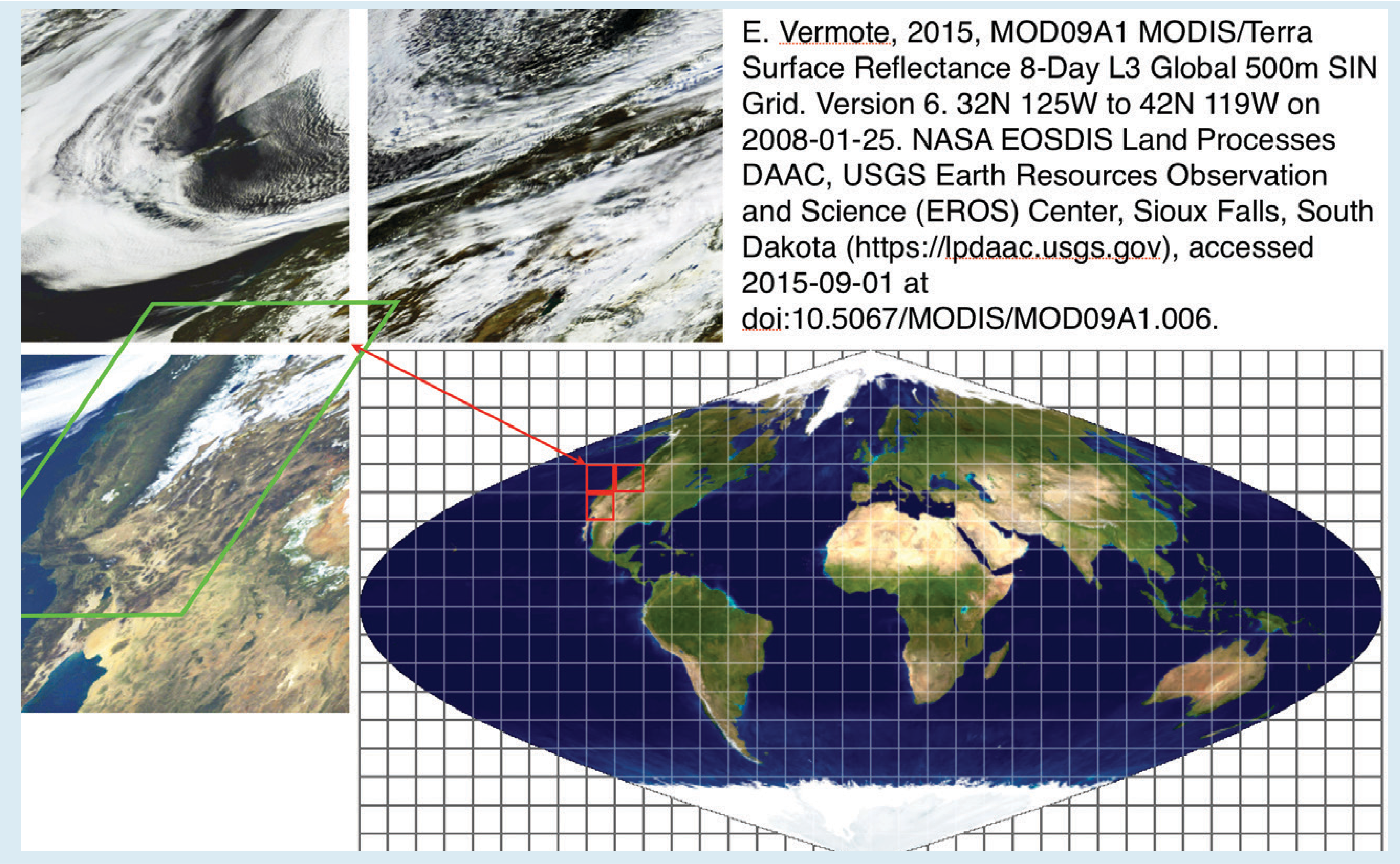

[N. Noy et al., 2019]

# Data Curation




[DCC]

# Computational Data Citation (MODIS)



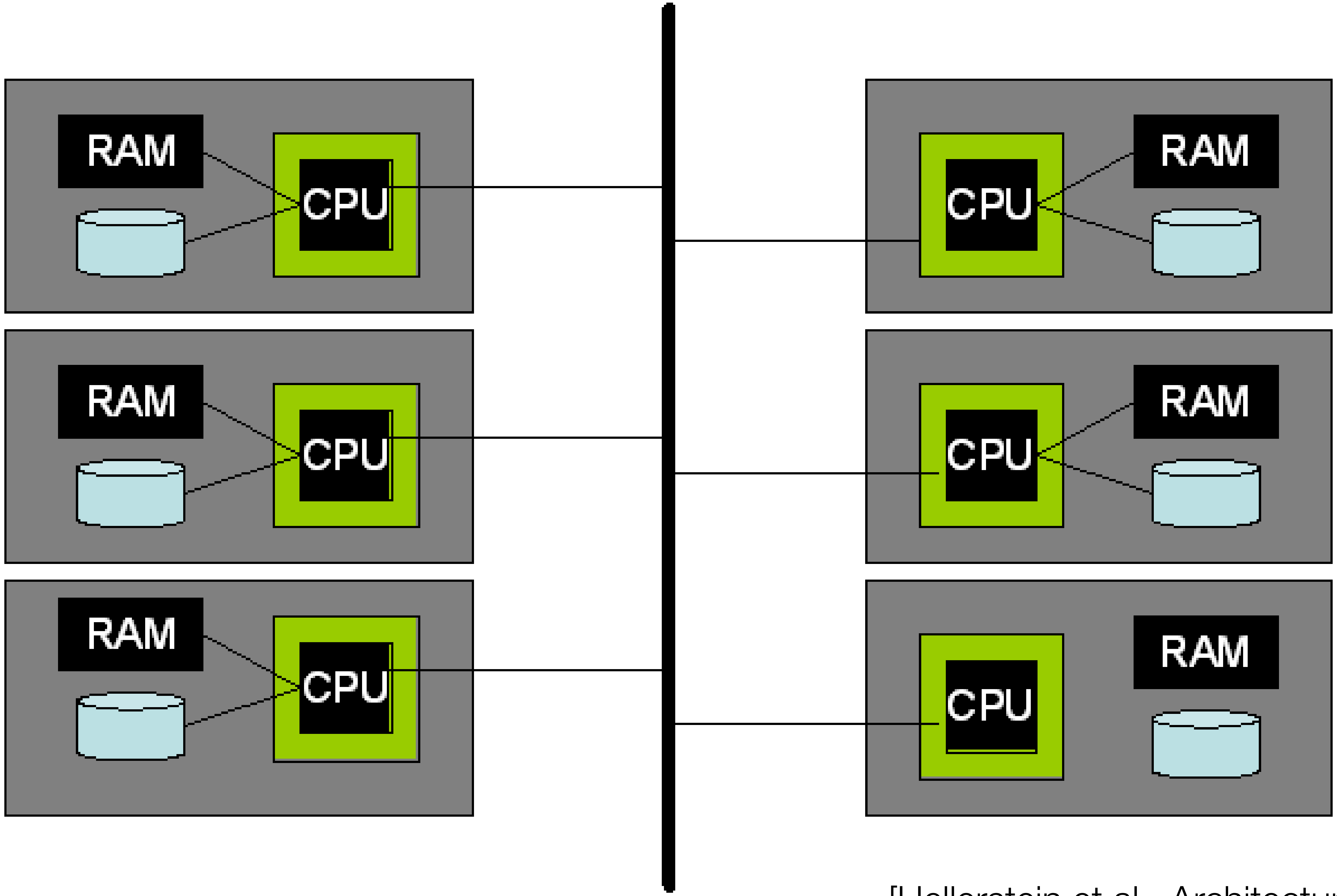

[Buneman et al., 2016]

# FAIR Principles

- Findable: Metadata and data should be easy to find for both humans and computers
- Accessible: Users need to know how data can be accessed, possibly including authentication and authorization
- Interoperable: Can be integrated with other data, and can interoperate with applications or workflows for analysis, storage, and processing
- Reusable: Optimize the reuse of data. Metadata and data should be well-described so they can be replicated and/or combined in different settings

[GO FAIR]

# Parallel DB Architecture: Shared Nothing




[Hellerstein et al., Architecture of a Database System]

# Column Stores

row id = 1

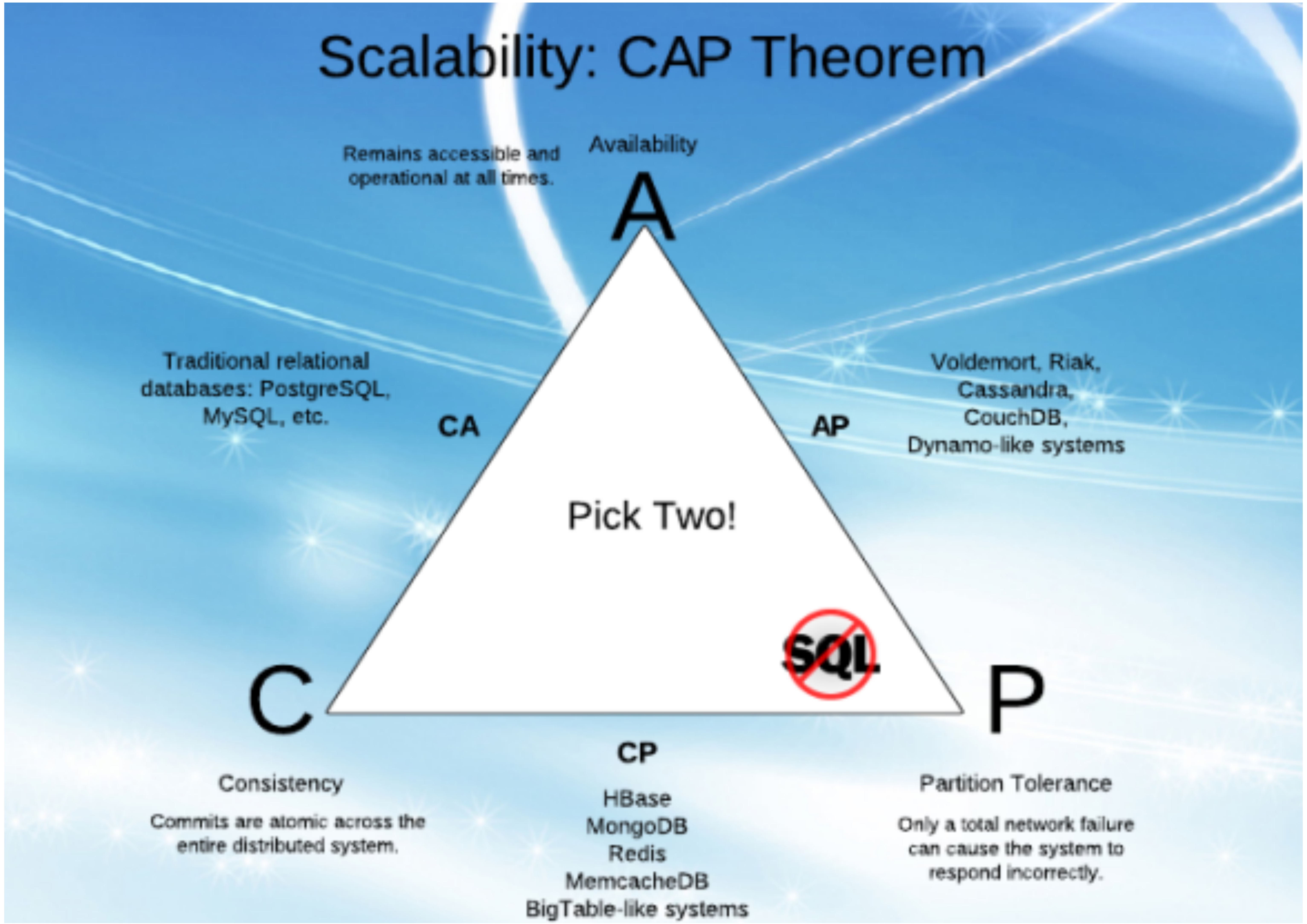
row id = 6

| id | Title | Person | Genre |
|---|---|---|---|
| 1 | Mrs. Doubtfire | Robin Williams | Comedy |
| 2 | Jaws | Roy Scheider | Horror |
| 3 | The Fly | Jeff Goldblum | Horror |
| 4 | Steel Magnolias | Dolly Parton | Drama |
| 5 | The Birdcage | Nathan Lane | Comedy |
| 6 | Erin Brokovitch | Julia Roberts | Drama |

Each column has a file or segment on disk

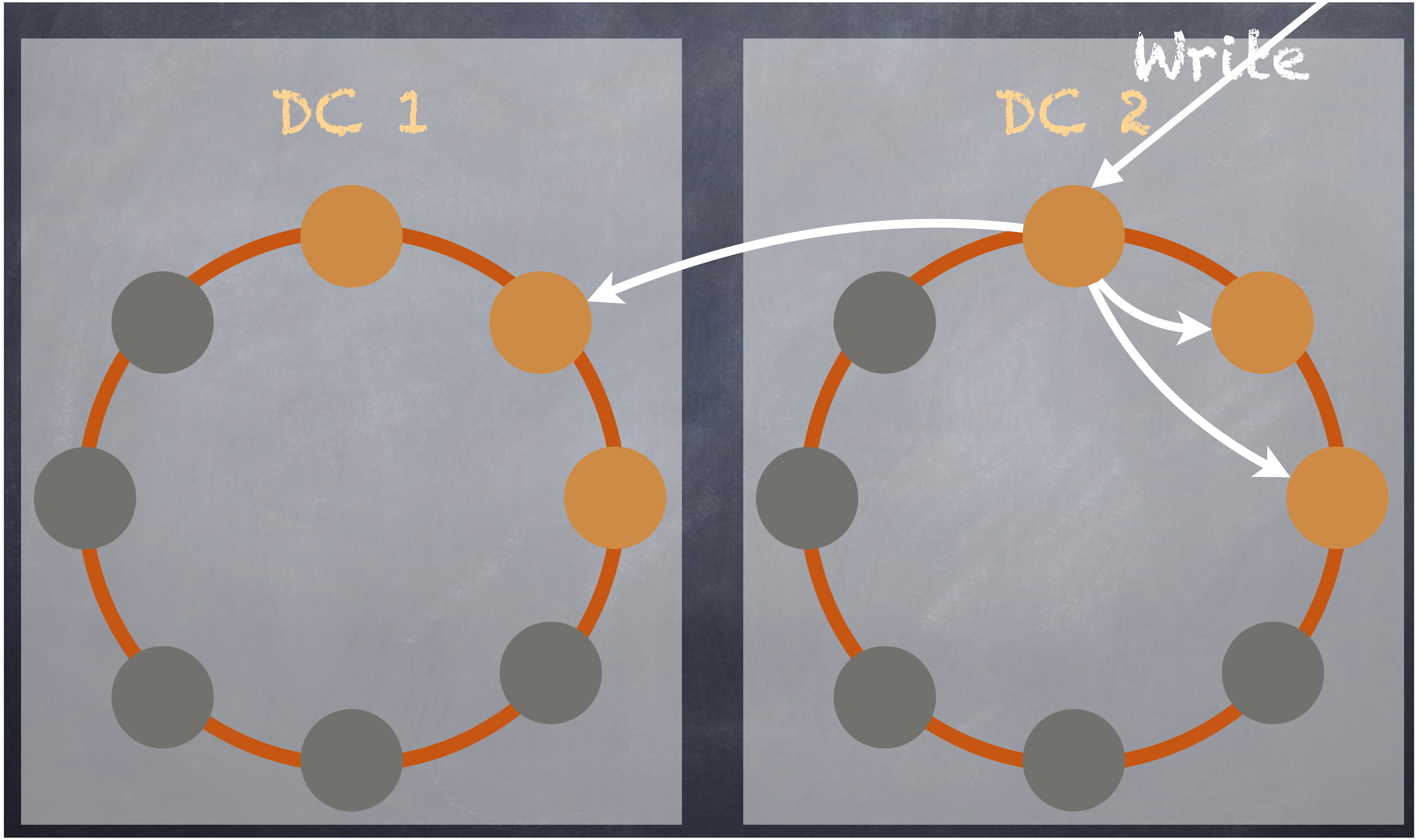
[J. Swanhart, Introduction to Column Stores]

# CAP Theorem



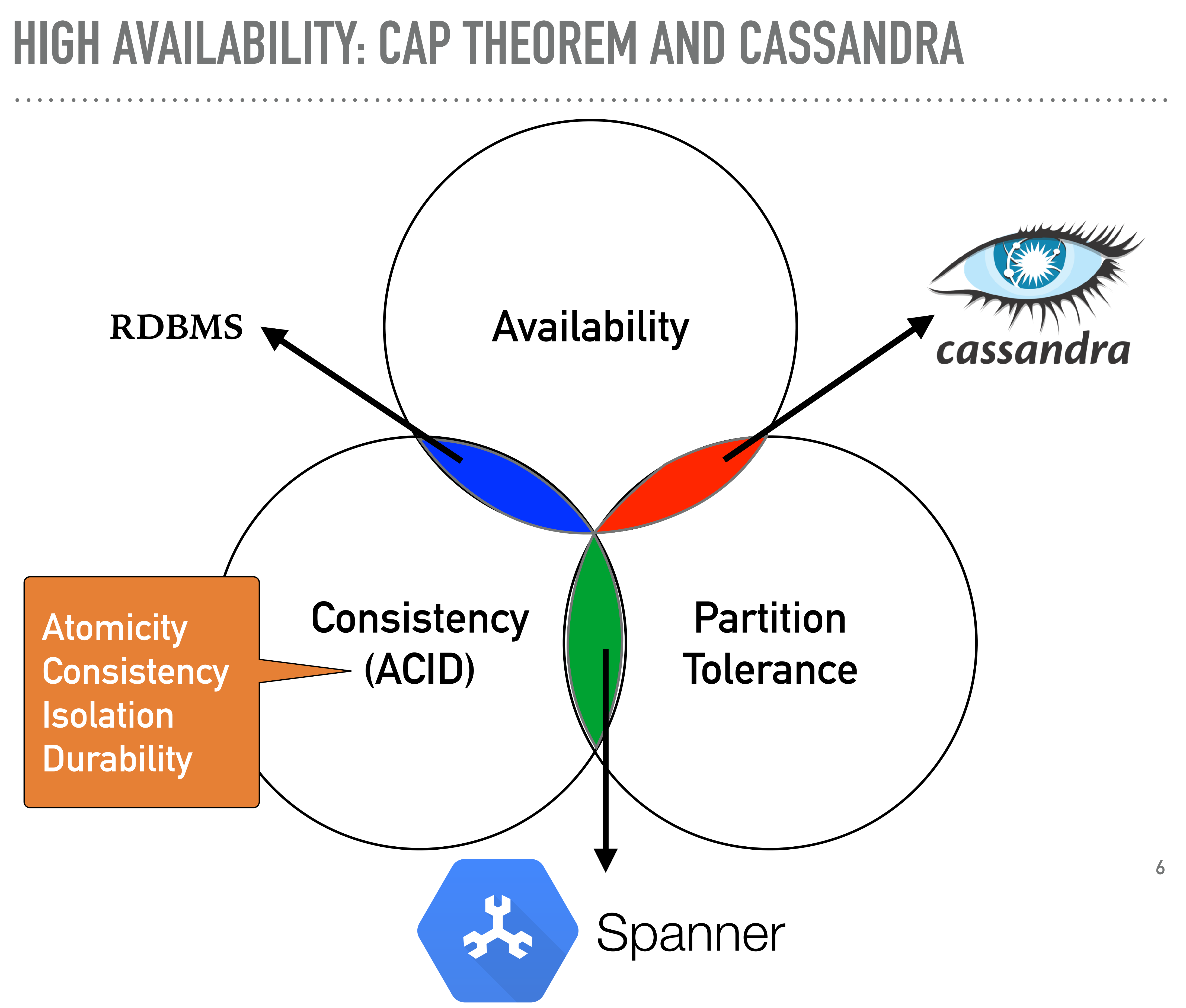

[E. Brewer]

# Cassandra: Replication and Consistency




[R. Stupp]

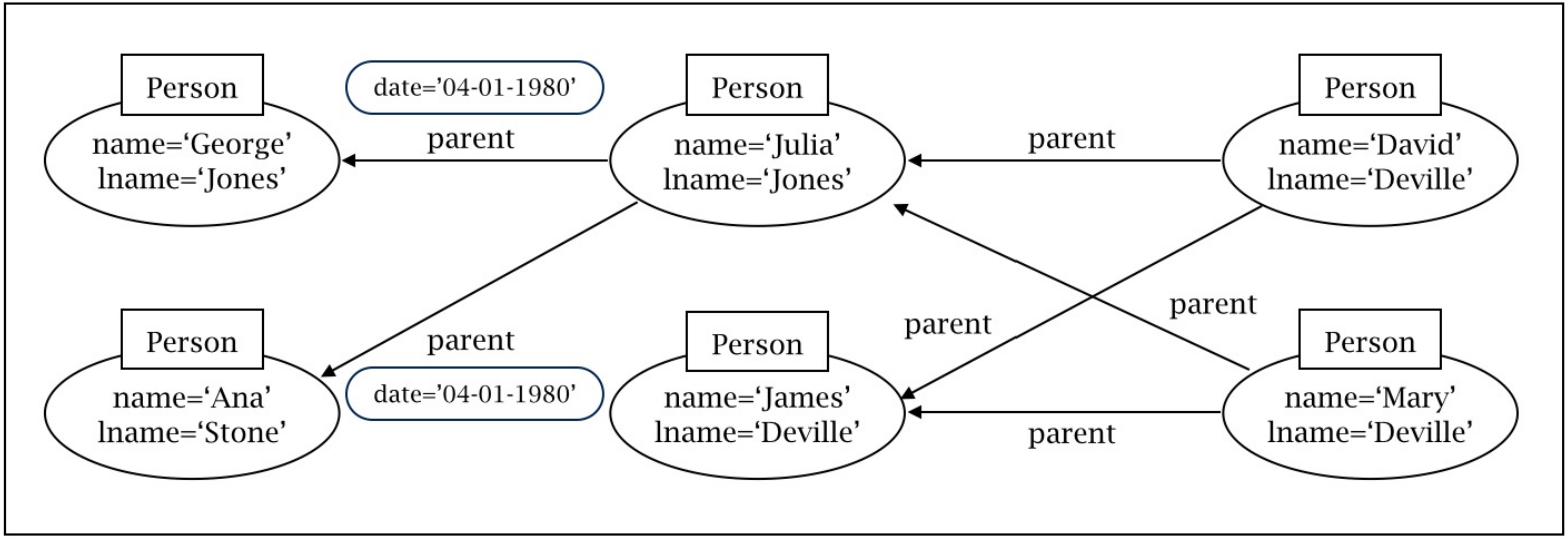
# Spanner: Google's NewSQL Cloud Database




- Which type of system is Spanner?
  - C: consistency, which implies a single value for shared data
  - A: 100% availability, for both reads and updates
  - P: tolerance to network partitions
- Which two?
  - CA: close, but not totally available
  - So actually **CP**

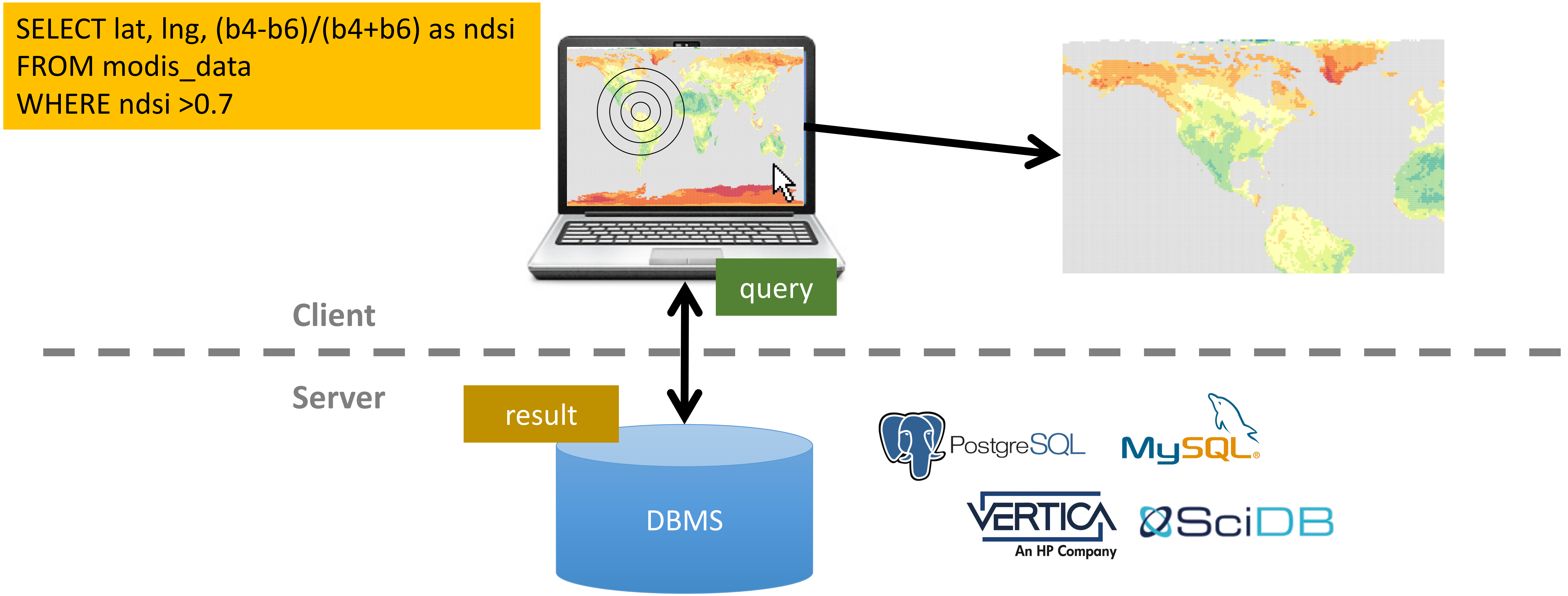
# Graph Databases focus on relationships

- Directed, labelled, attributed multigraph
- Properties are **key/value pairs** that represent metadata for nodes and edges




[R. Angles and C. Gutierrez, 2017]

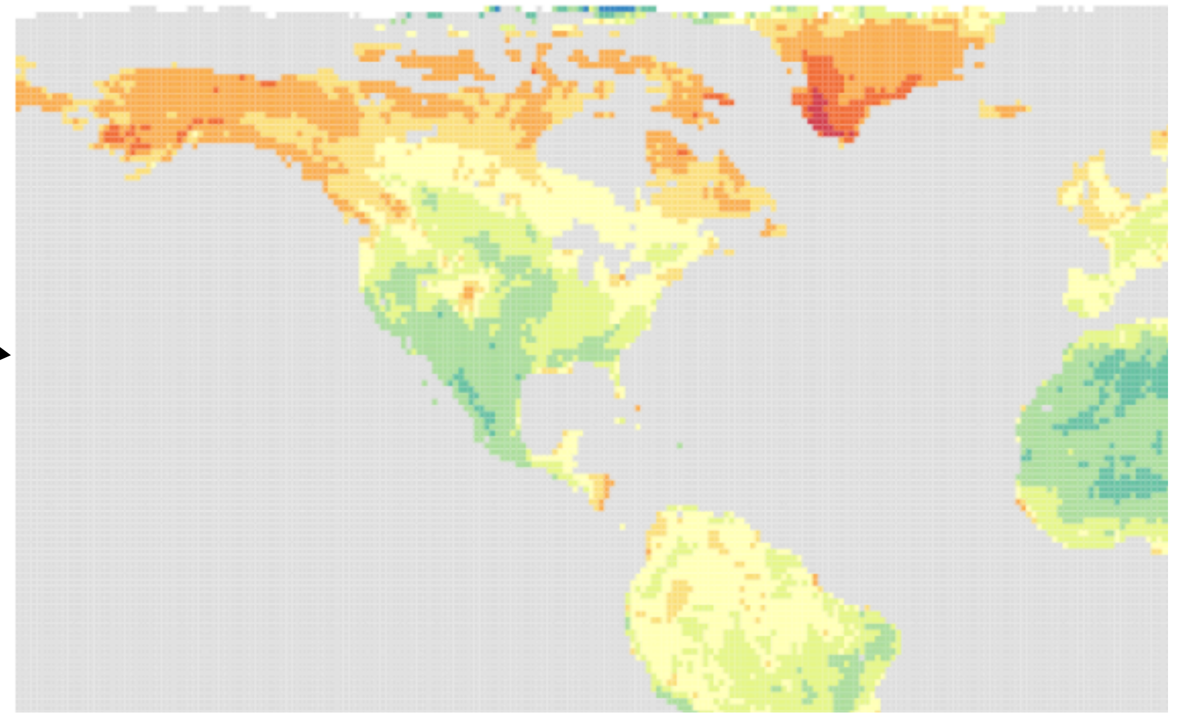
# Interactive Exploration of Spatial Data




[L. Battle, 2017]

# Interactive Exploration of Spatial Data

SELECT lat, lng, (b4-b6)/(b4+b6) as ndsi
FROM modis_data
WHERE ndsi >0.7



query

Client

Server

result

**SLOW** →

DBMS

PostgreSQL

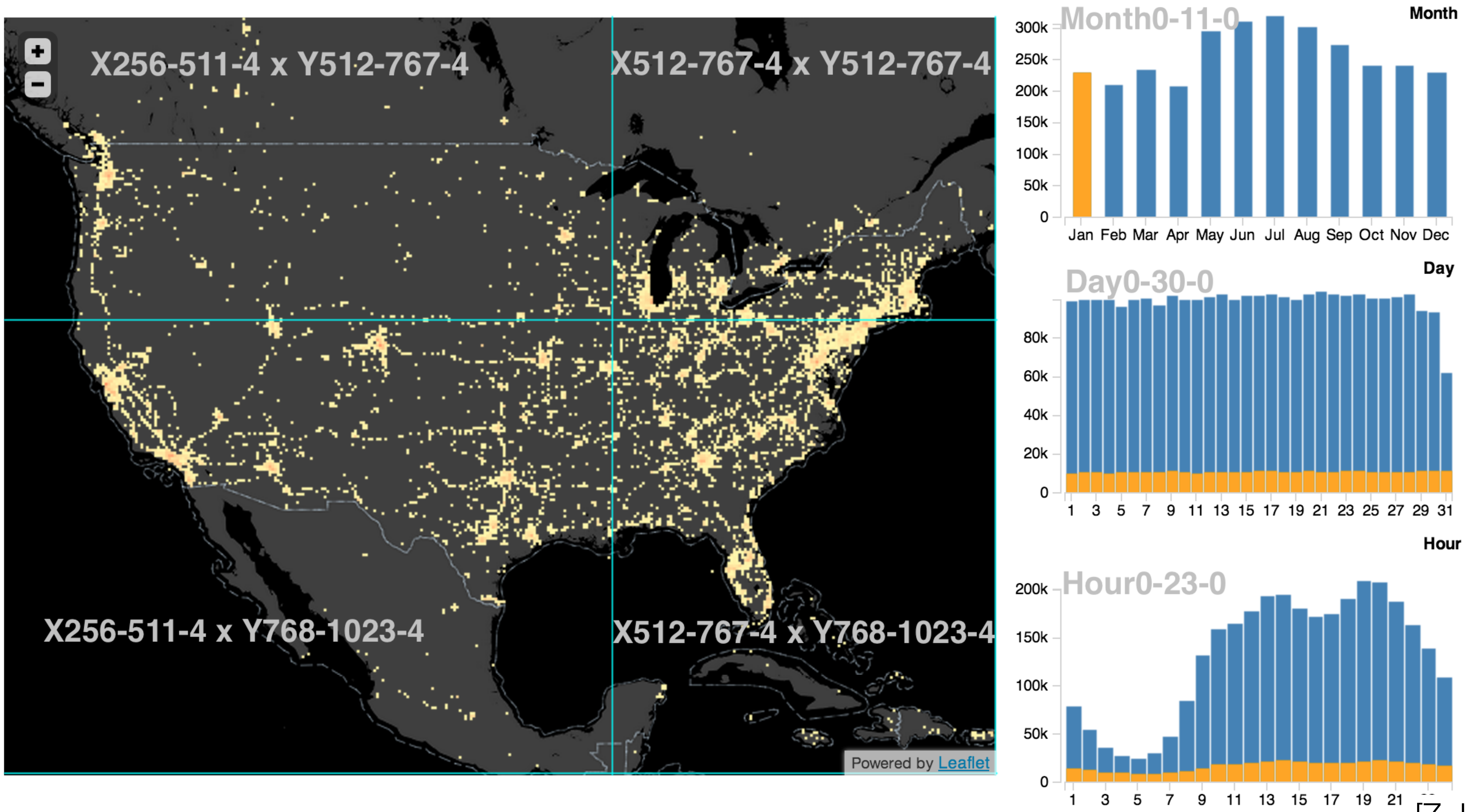
MySQL

VERTICA An HP Company

SciDB

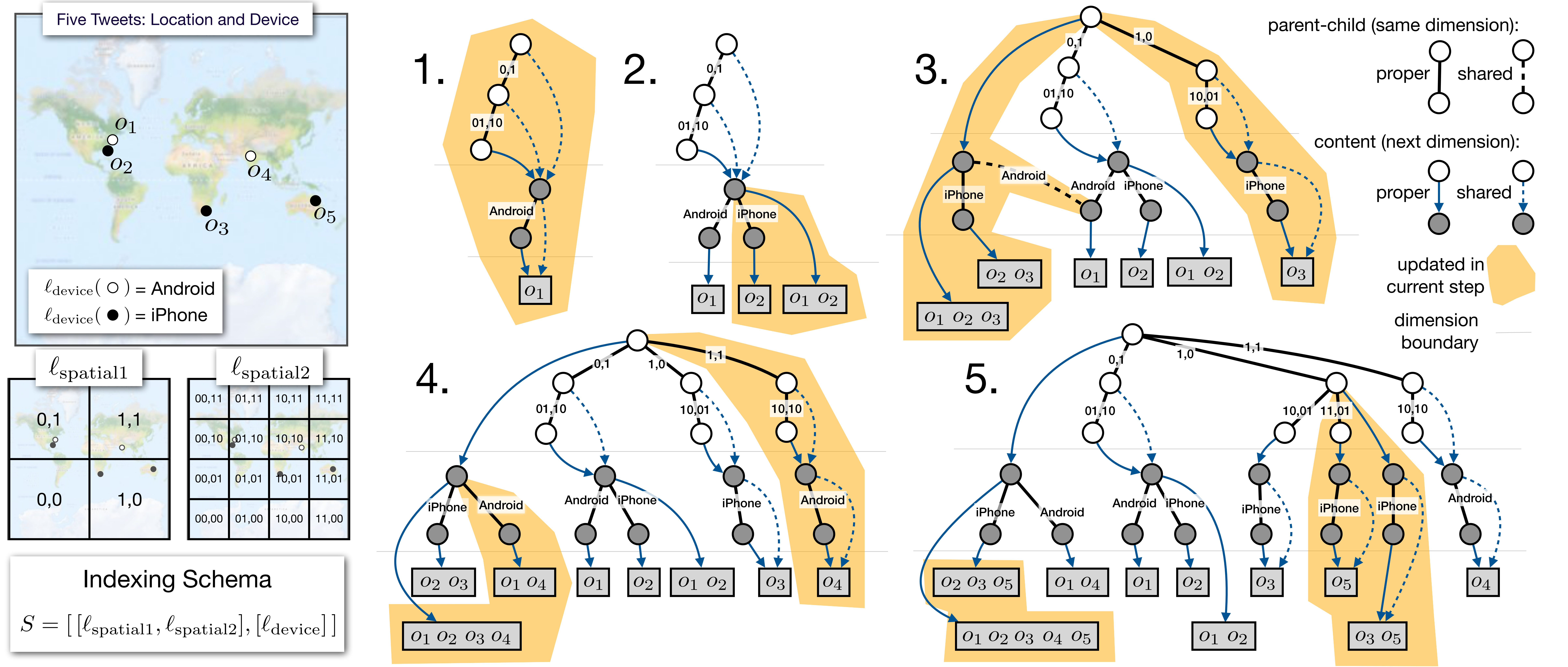
[L. Battle, 2017]

# Spatial Data: Minimize Latency




[Z. Liu et al., 2013]

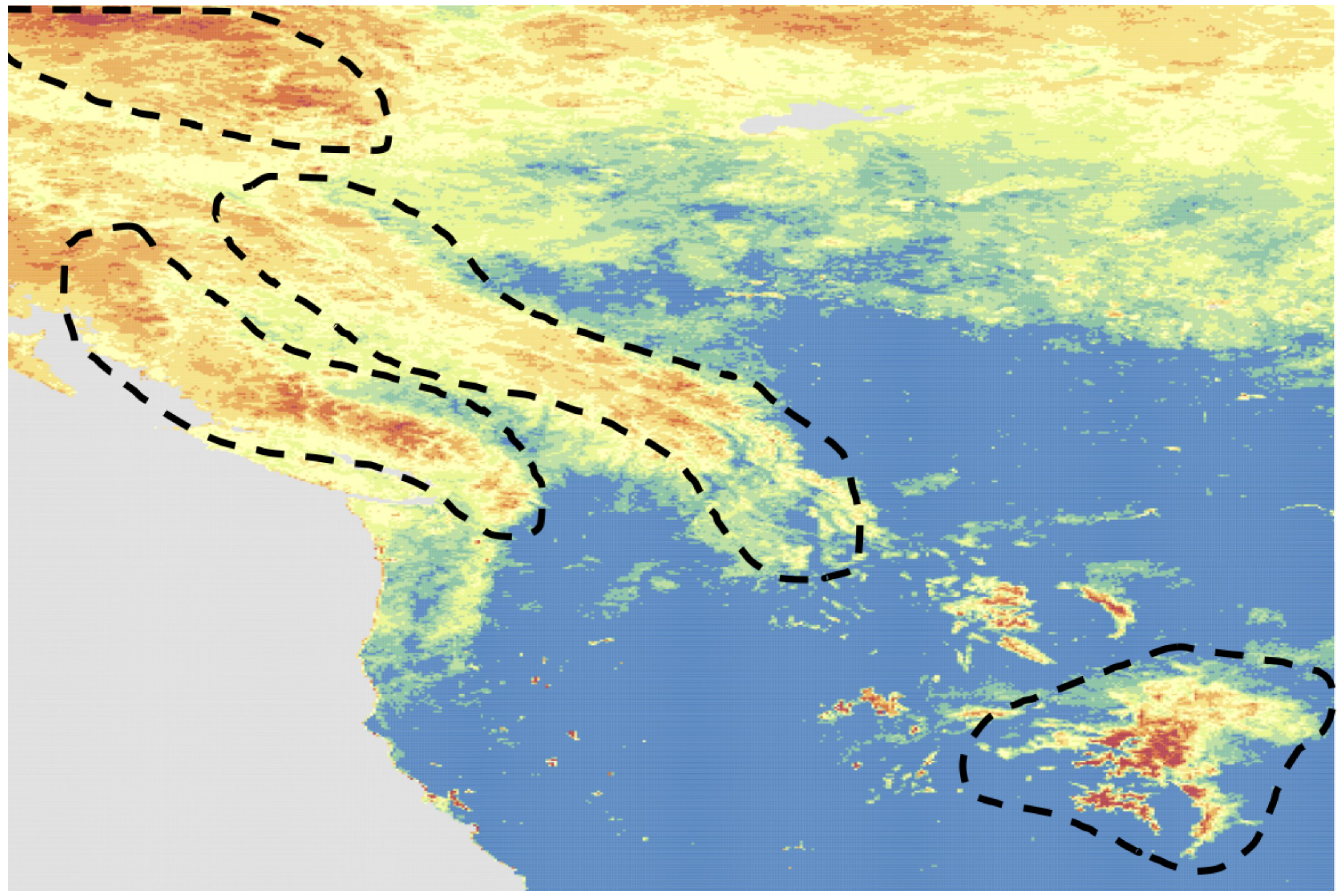
# Spatial Data: Precompute Optimized Storage

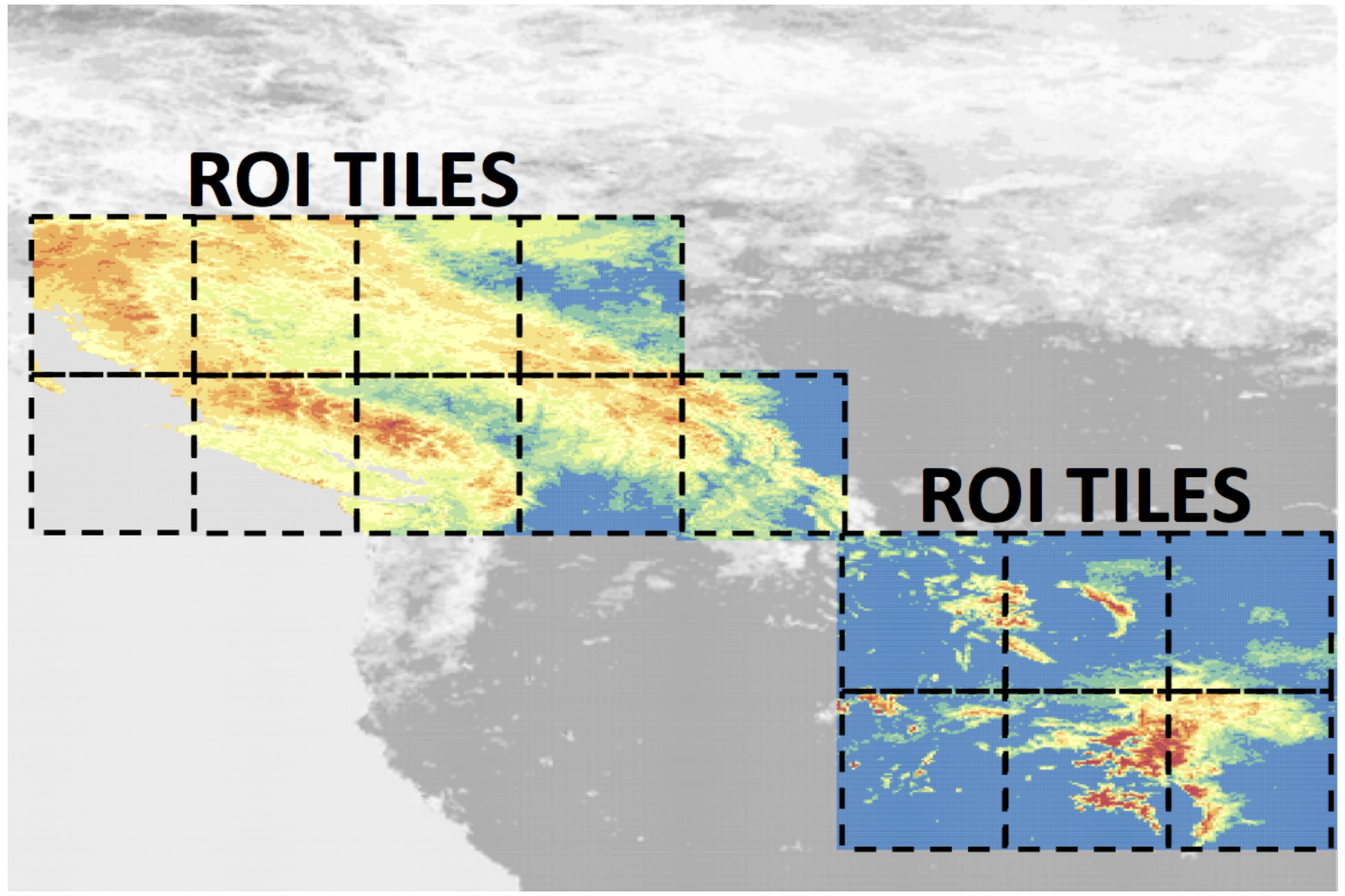



[Lins et. al, 2013]

# Spatial Data: Prefetching

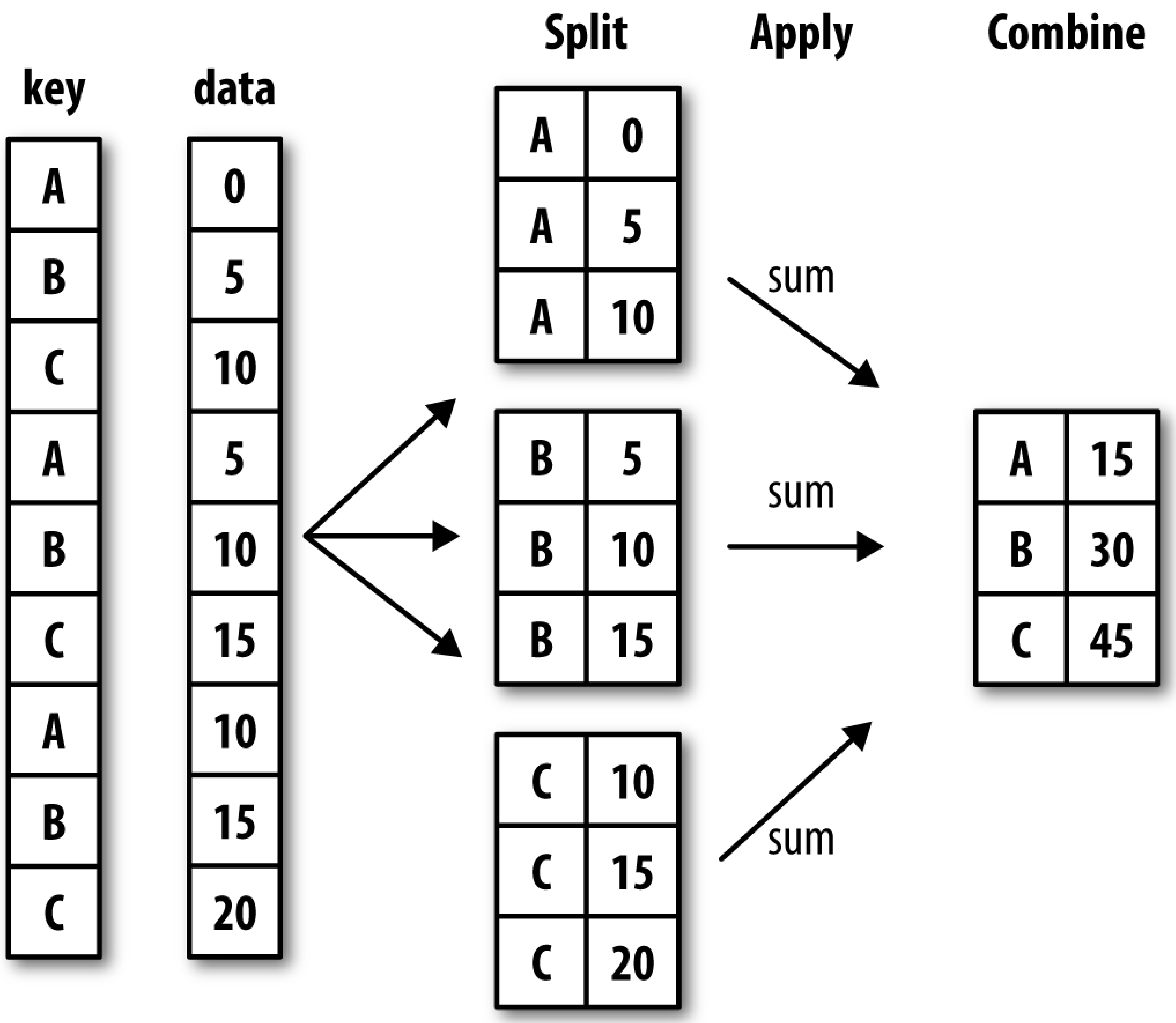
- Predict which tiles a user will need next and prefetch those
  - Use common patterns (zoom, pan)
  - Use regions of interest (ROIs)





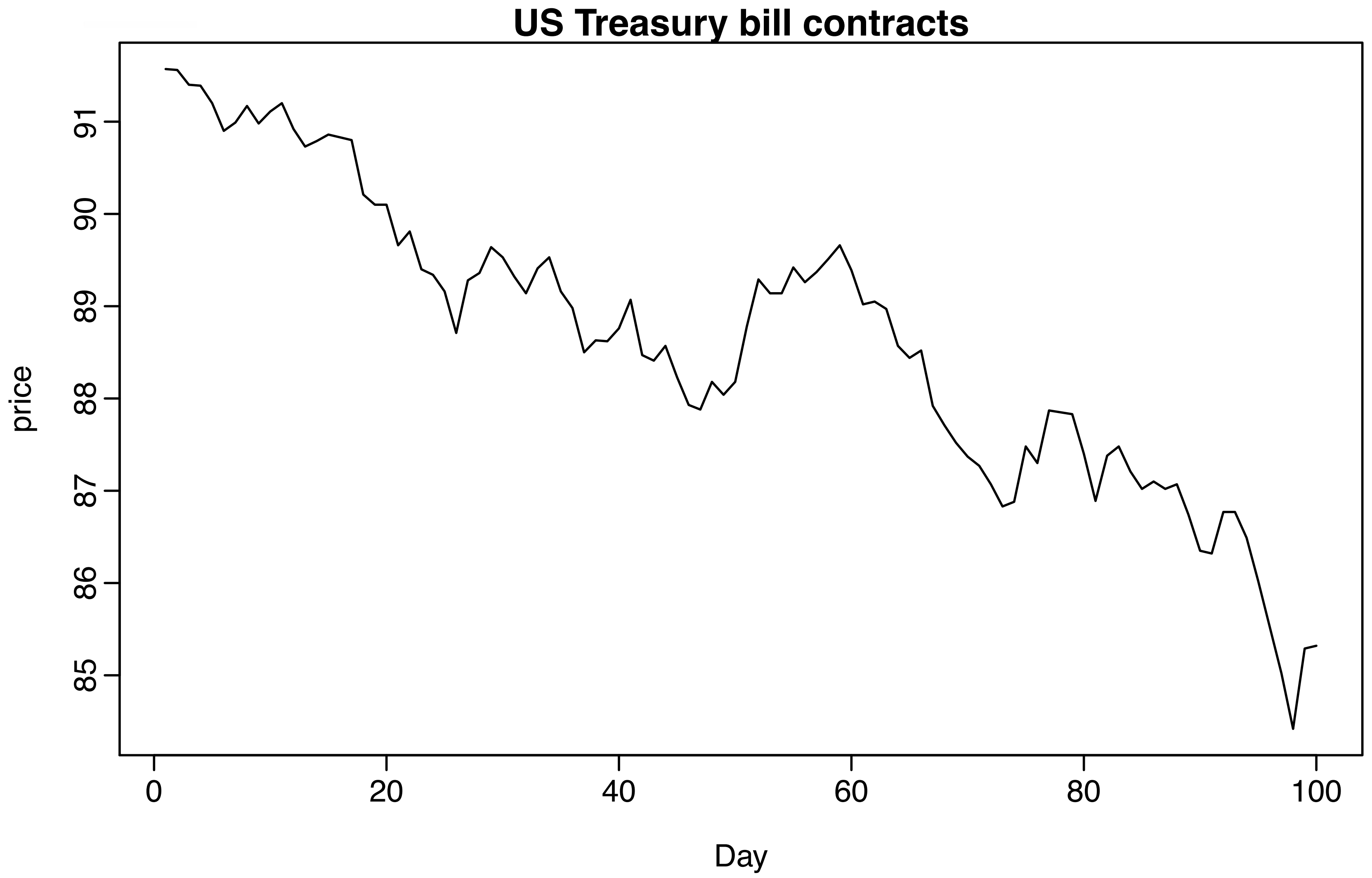

[Battle et al., 2016]

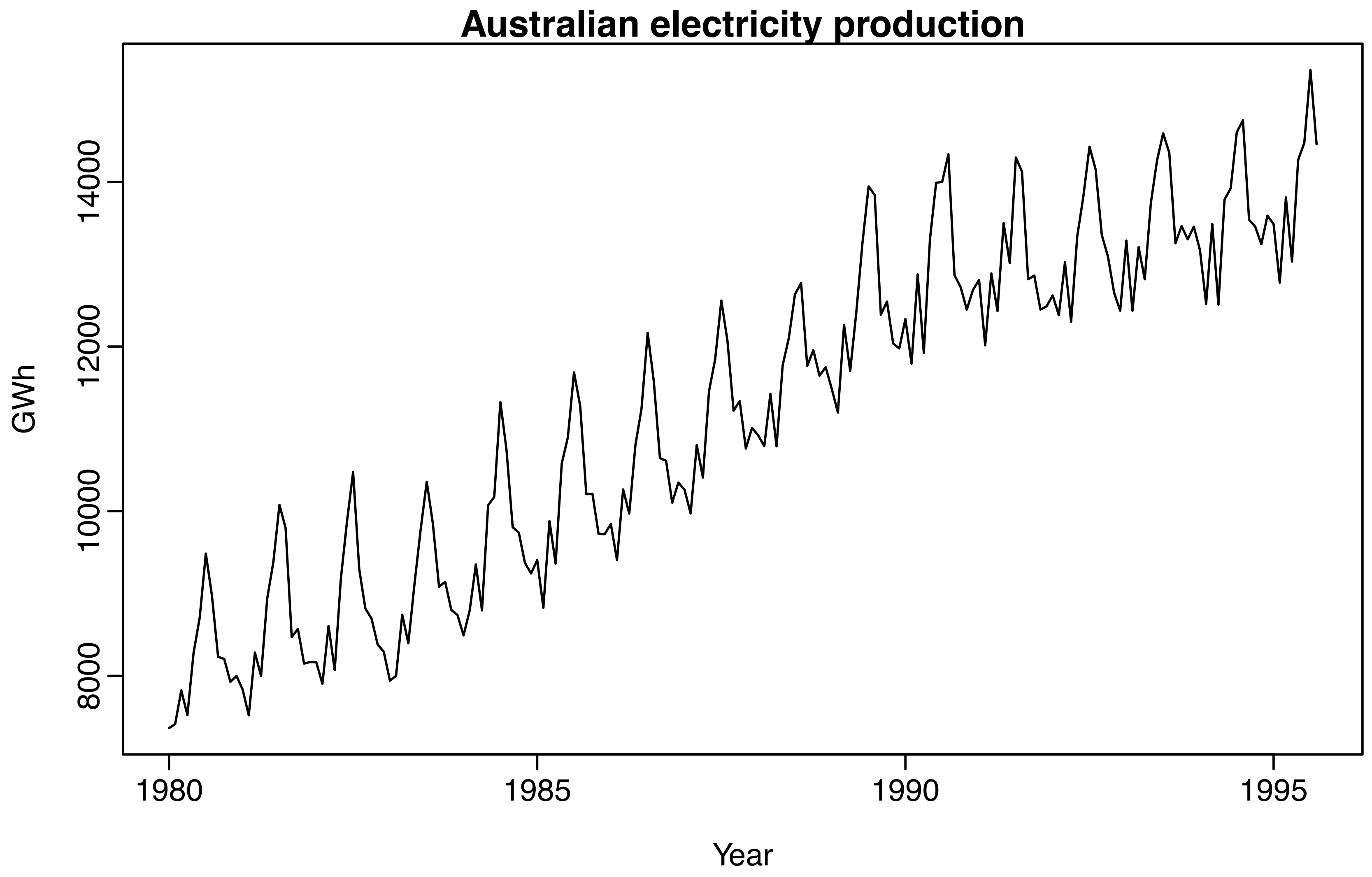
# Split-Apply-Combine

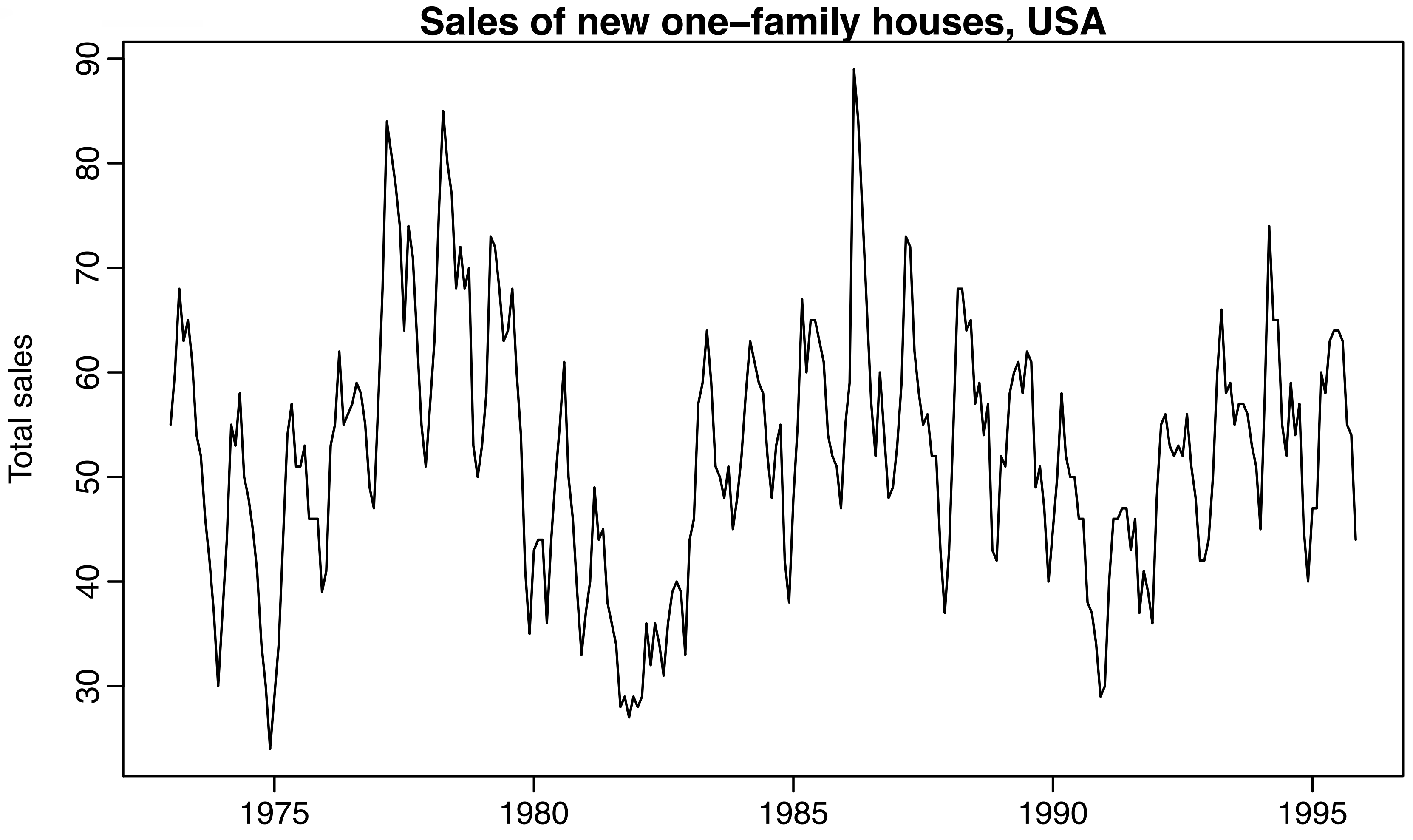


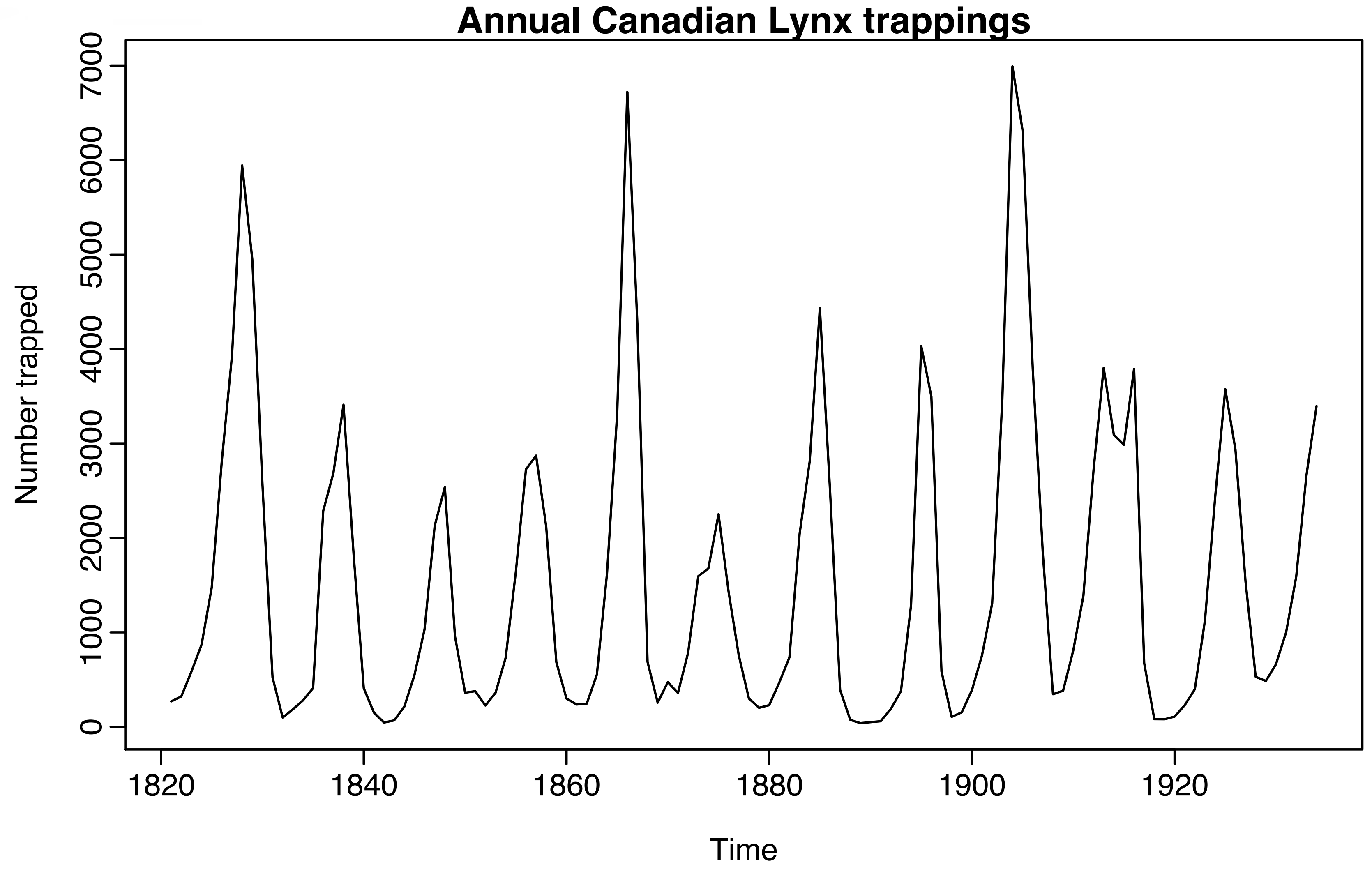

[W. McKinney, Python for Data Analysis]

# Time Series Data



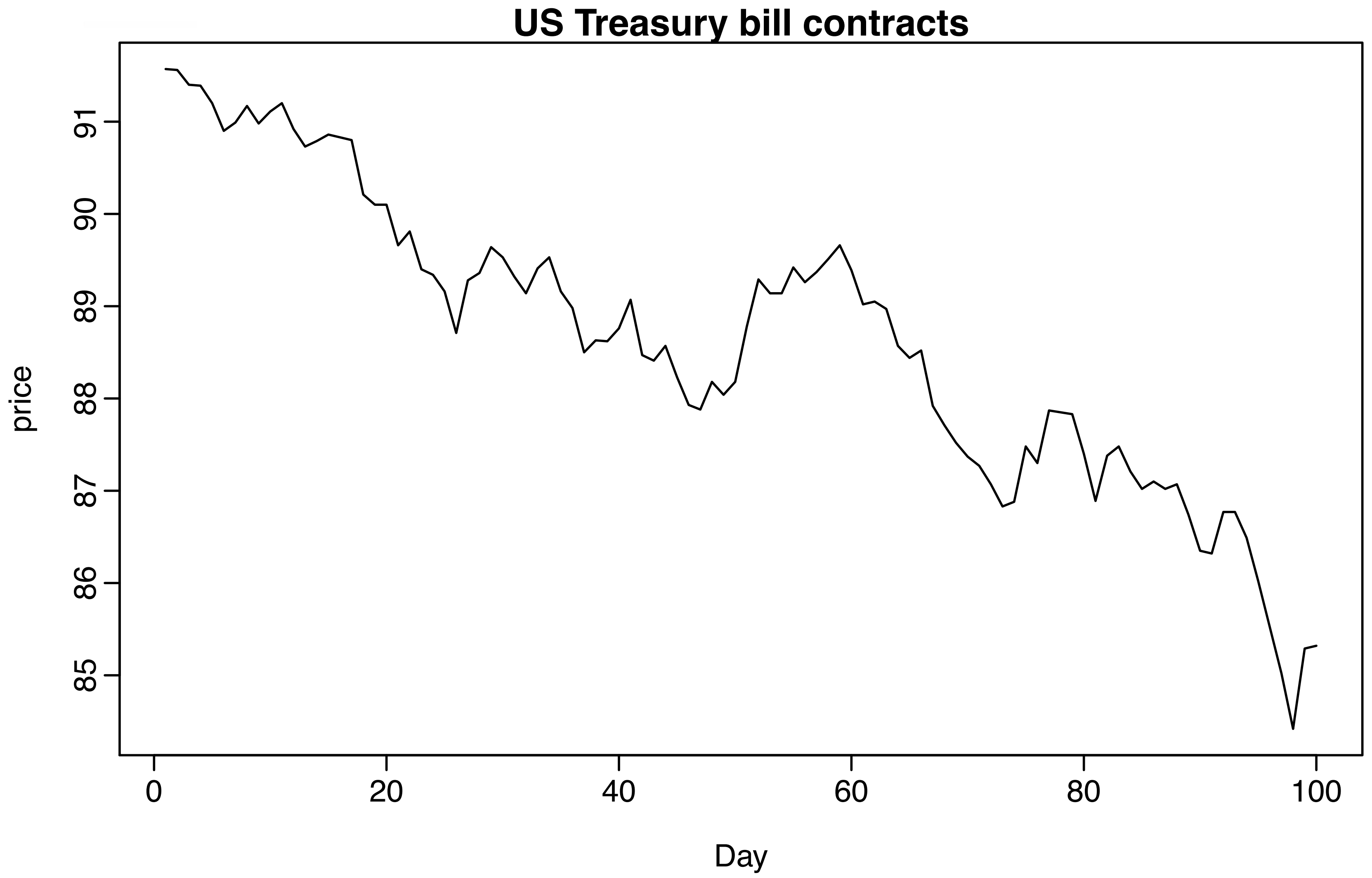



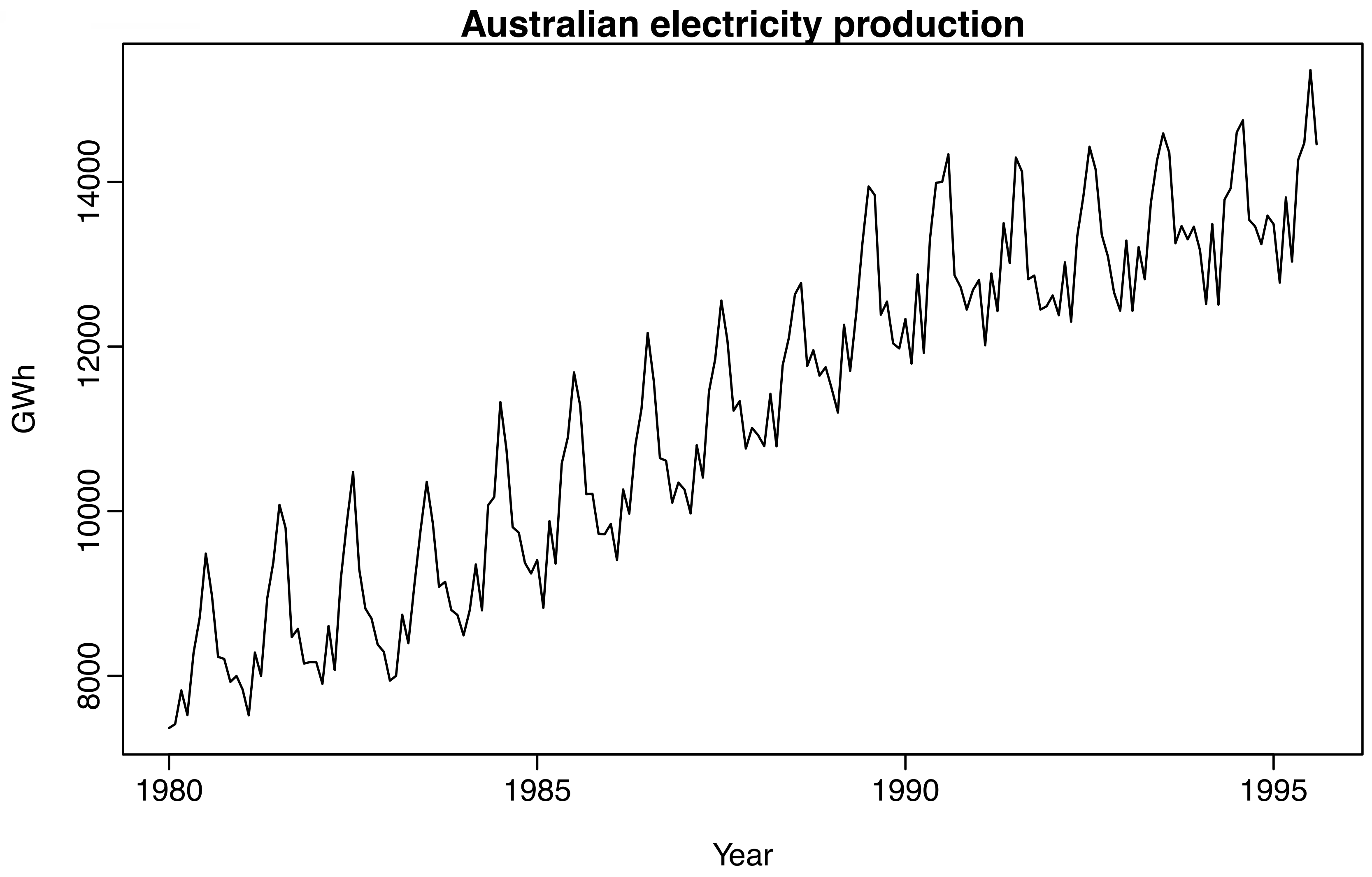



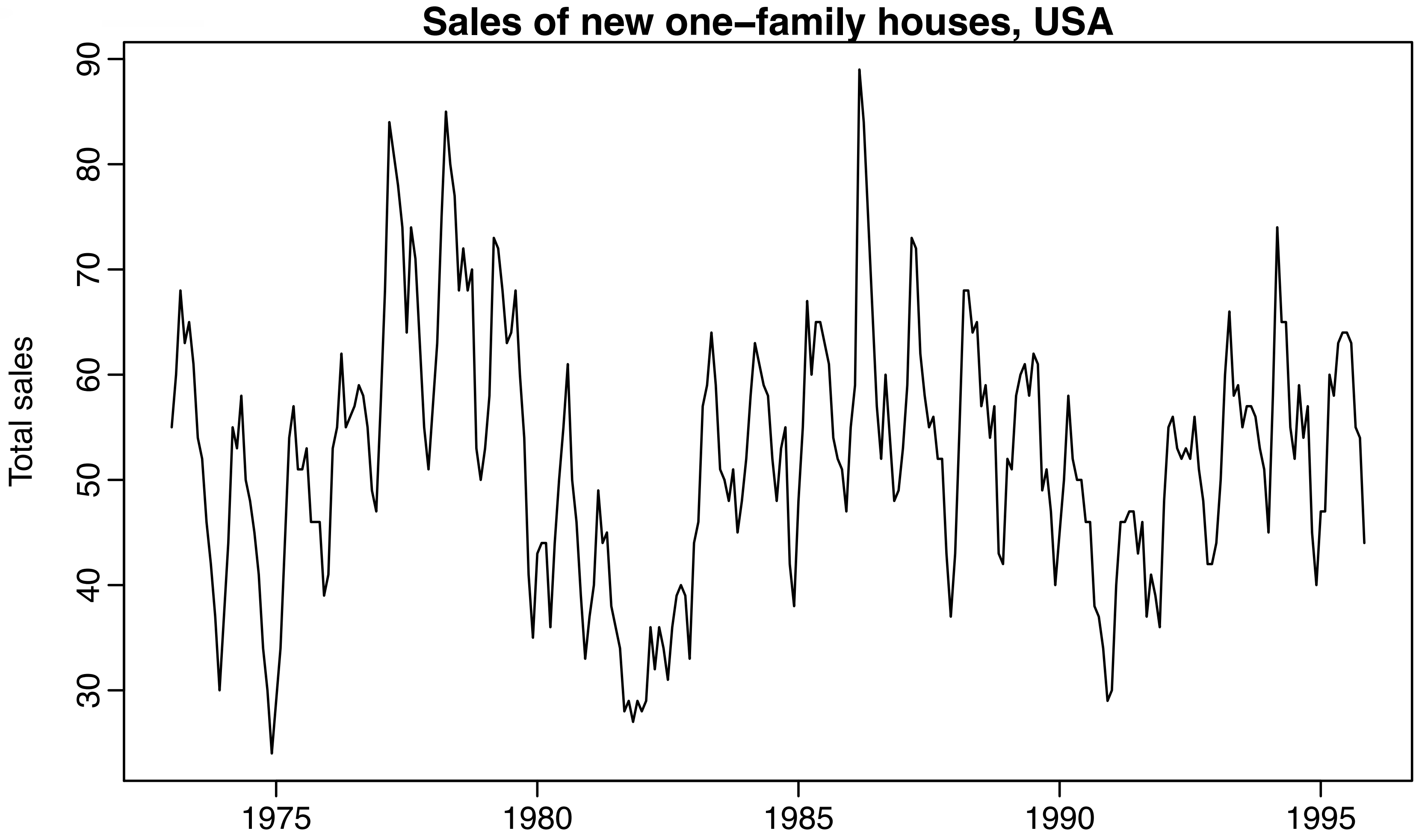

[R. J. Hyndman]

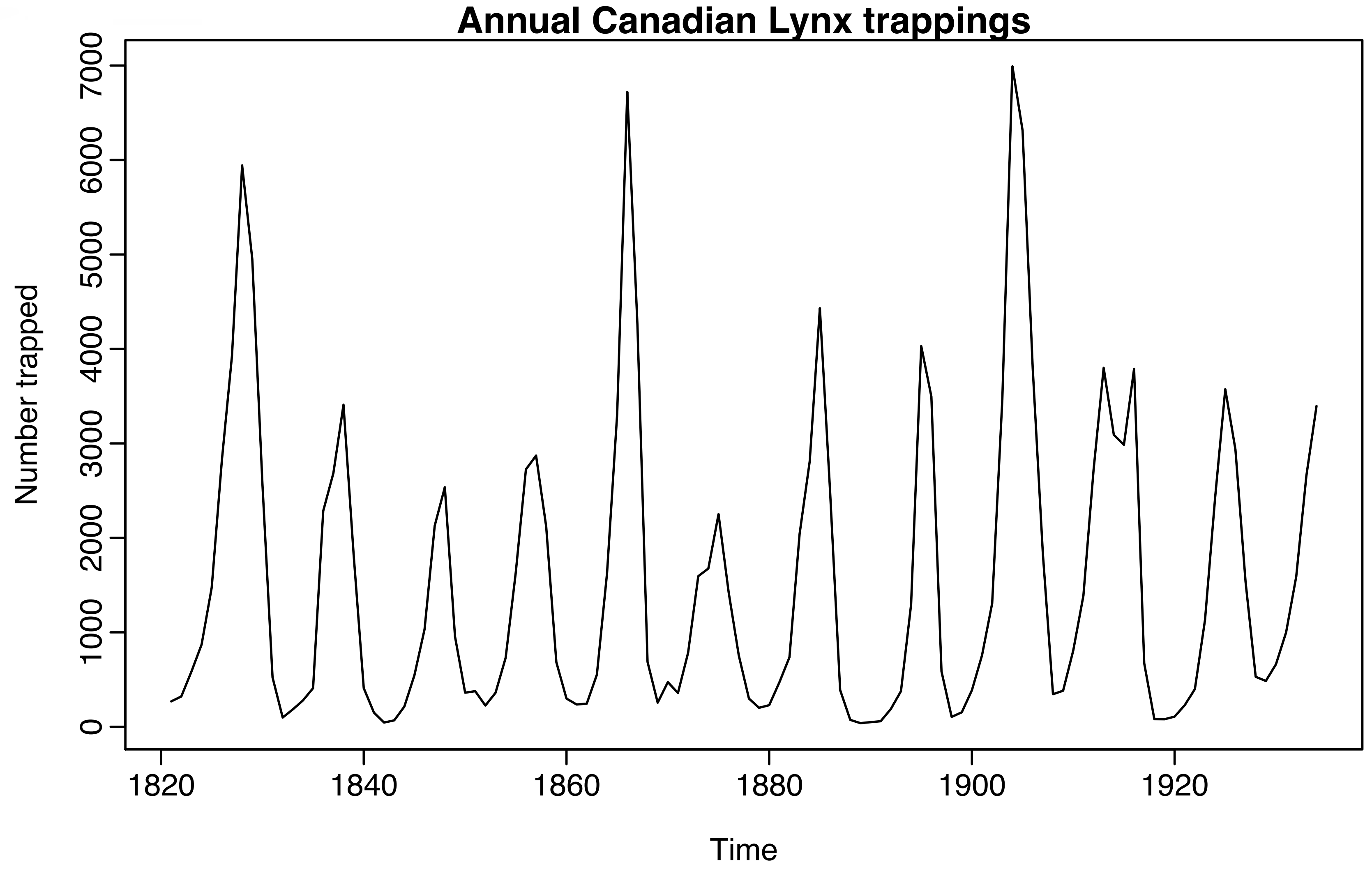
# Time Series Data

Trend



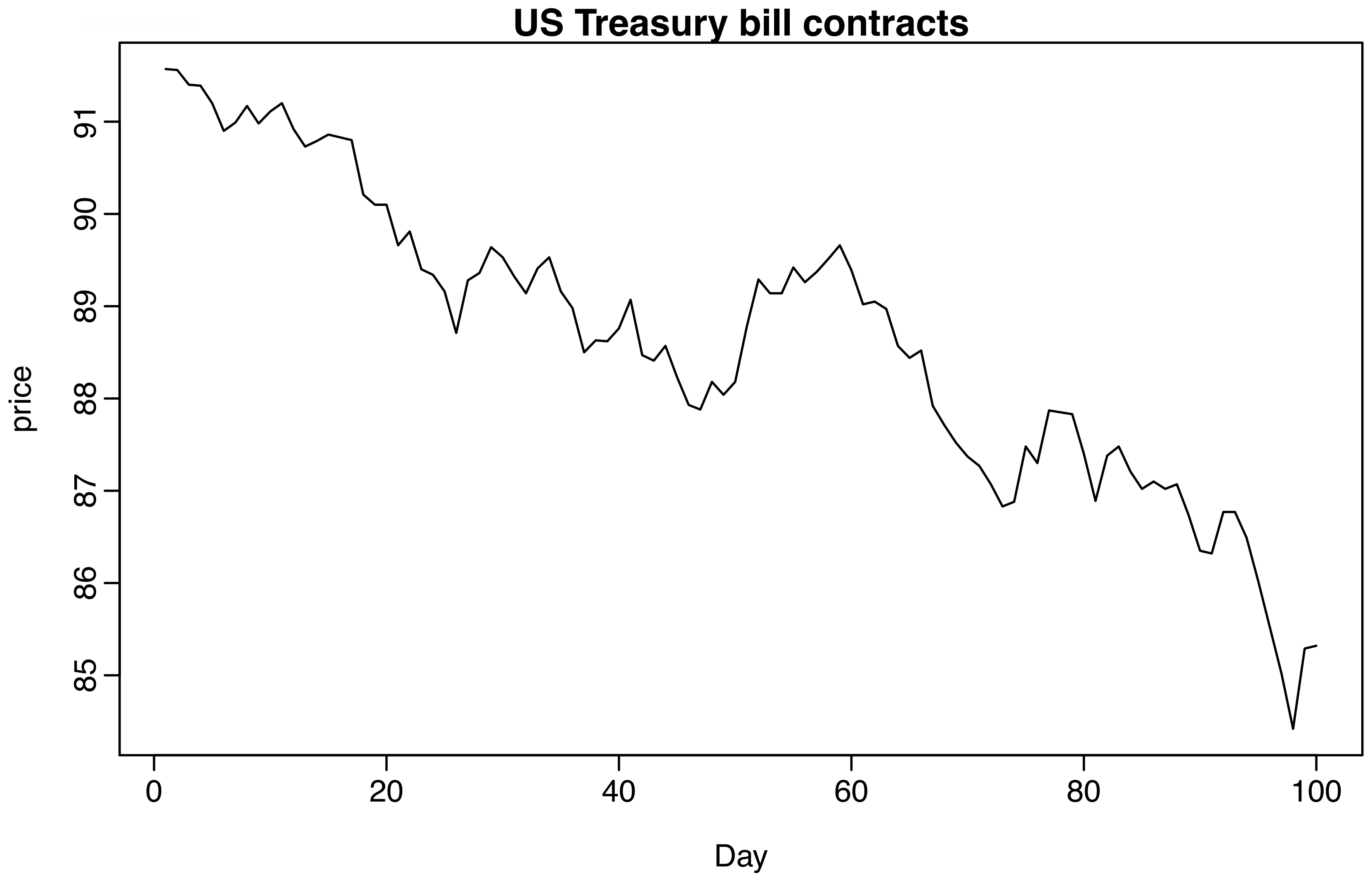



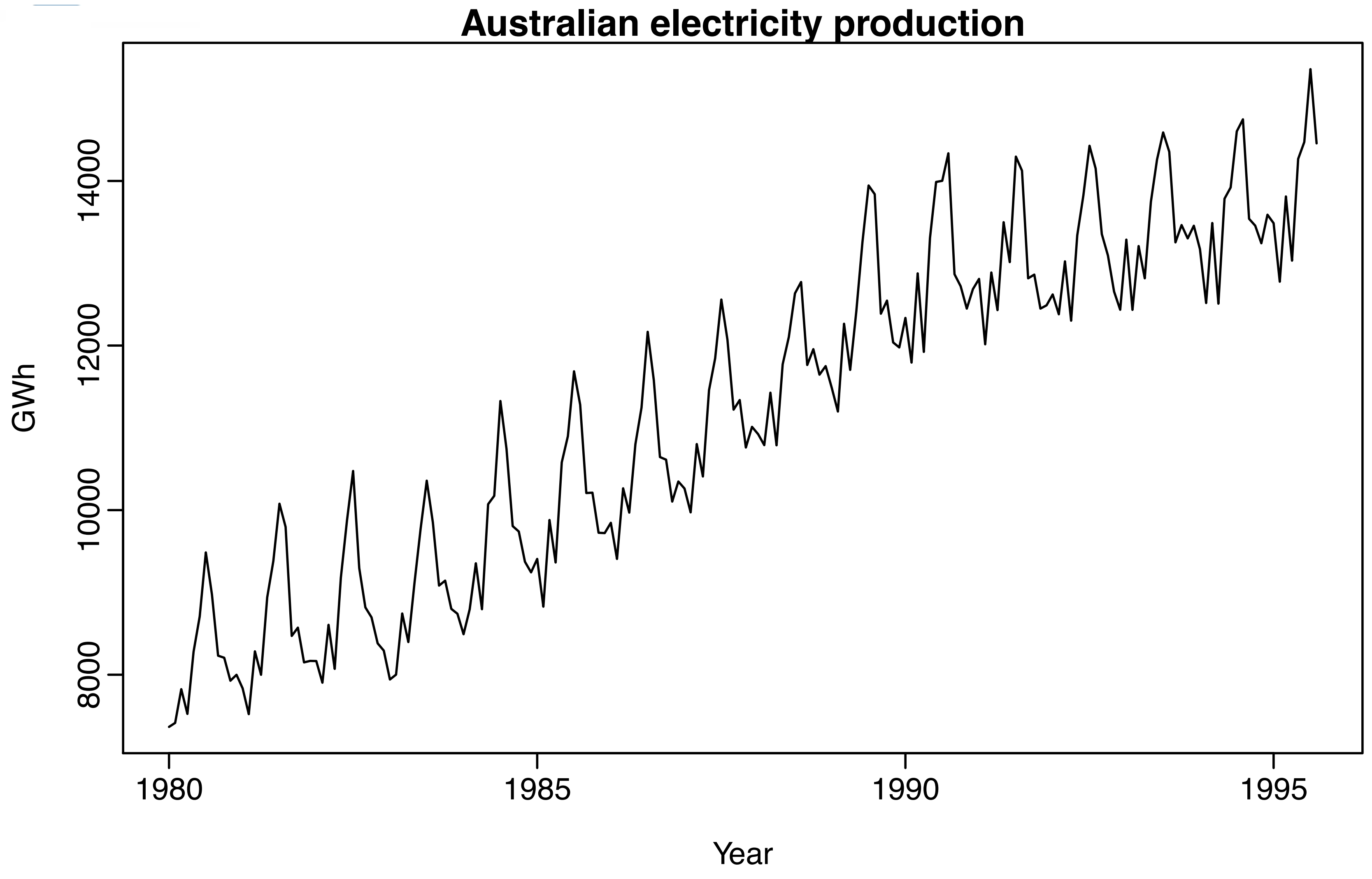




[R. J. Hyndman]

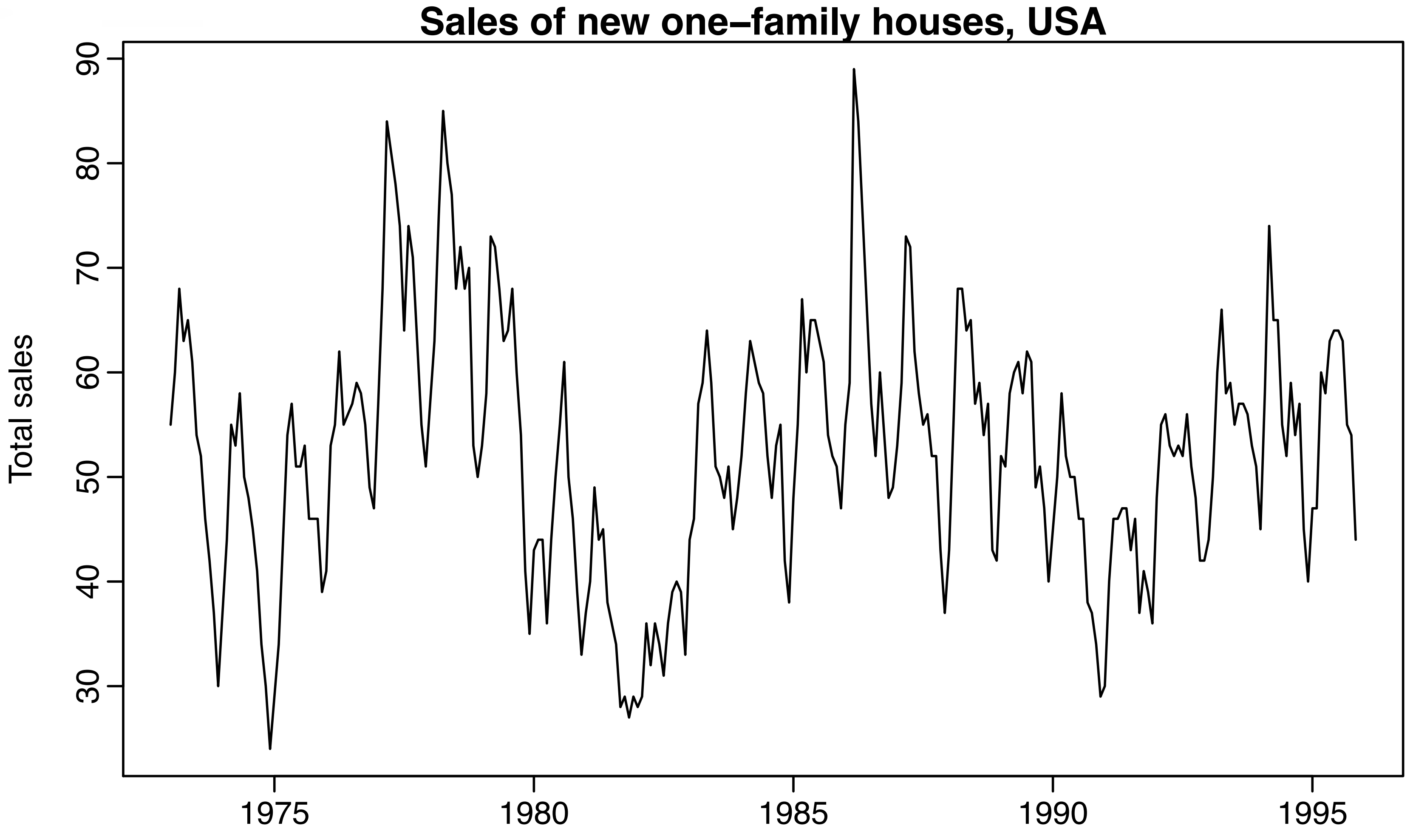
# Time Series Data

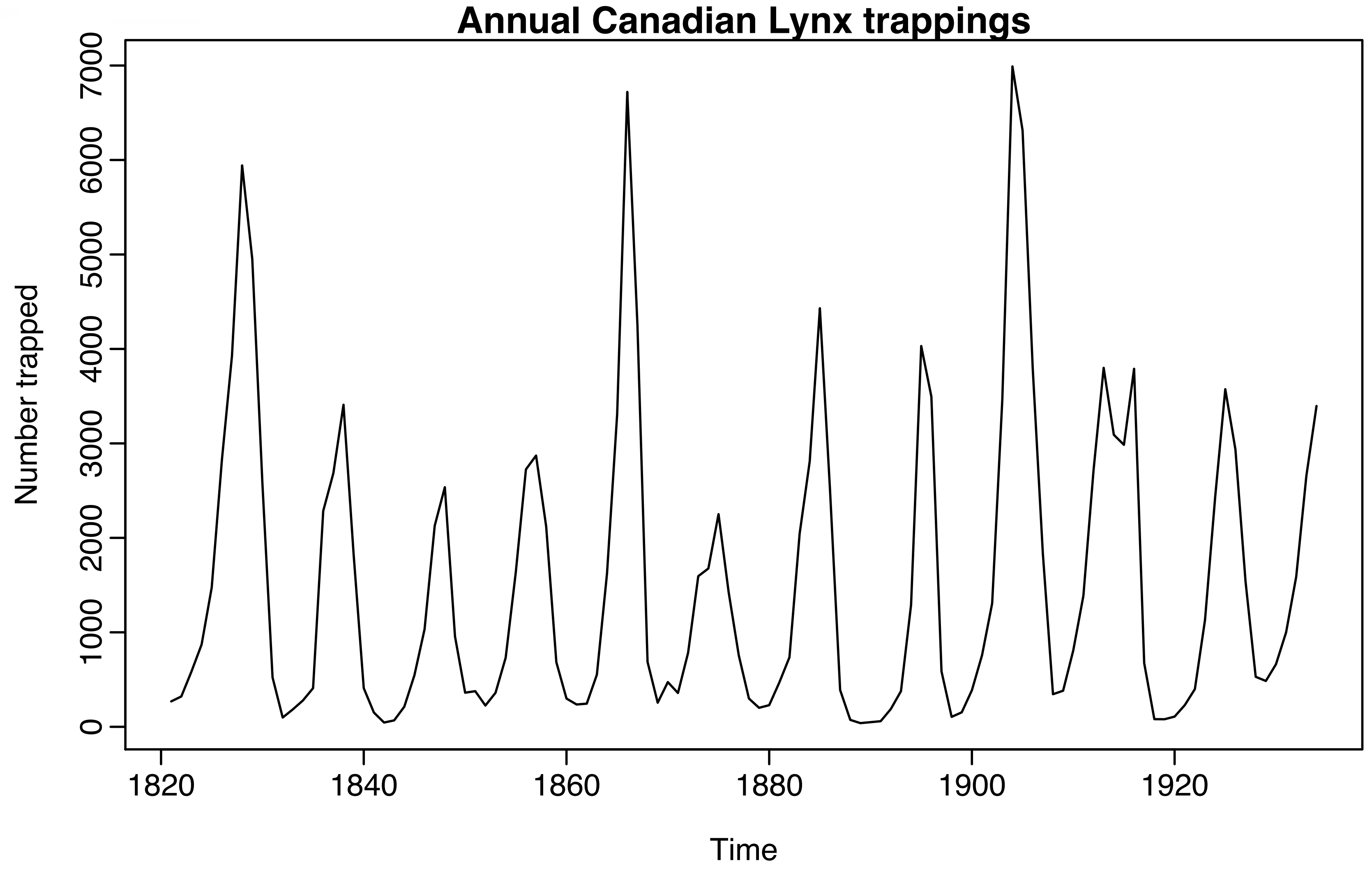
Trend




Trend + Seasonality

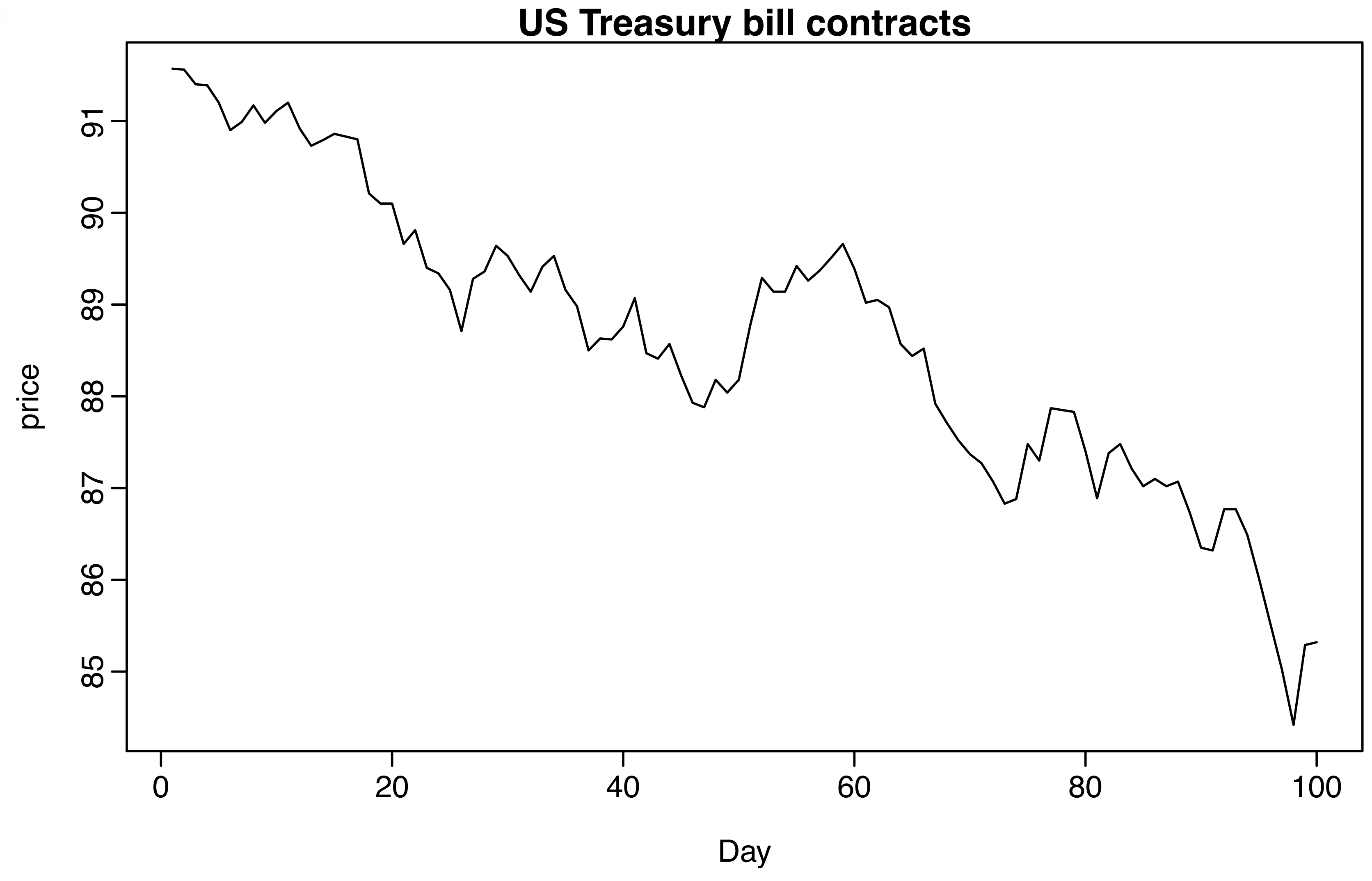


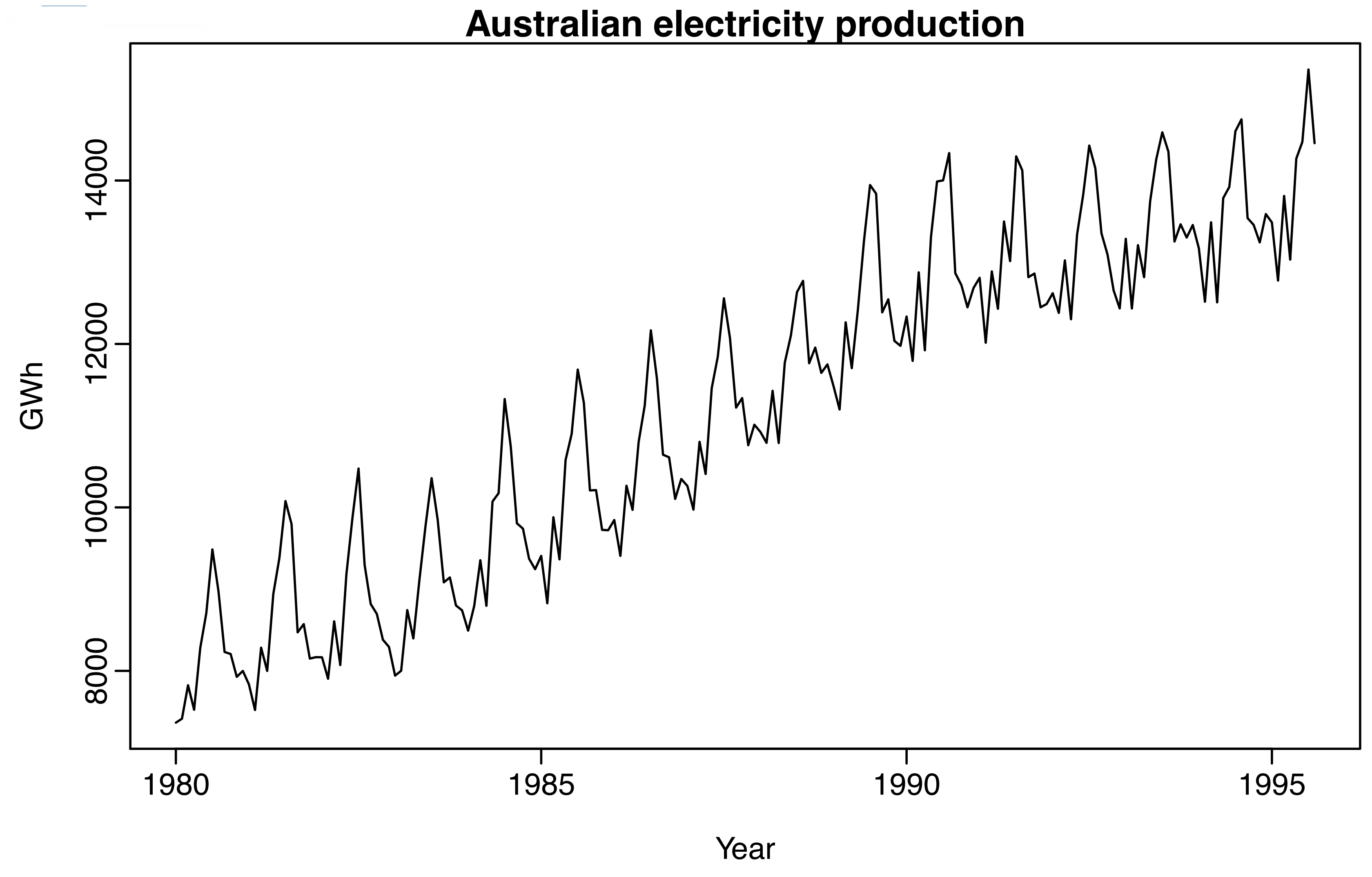

[R. J. Hyndman]

# Time Series Data

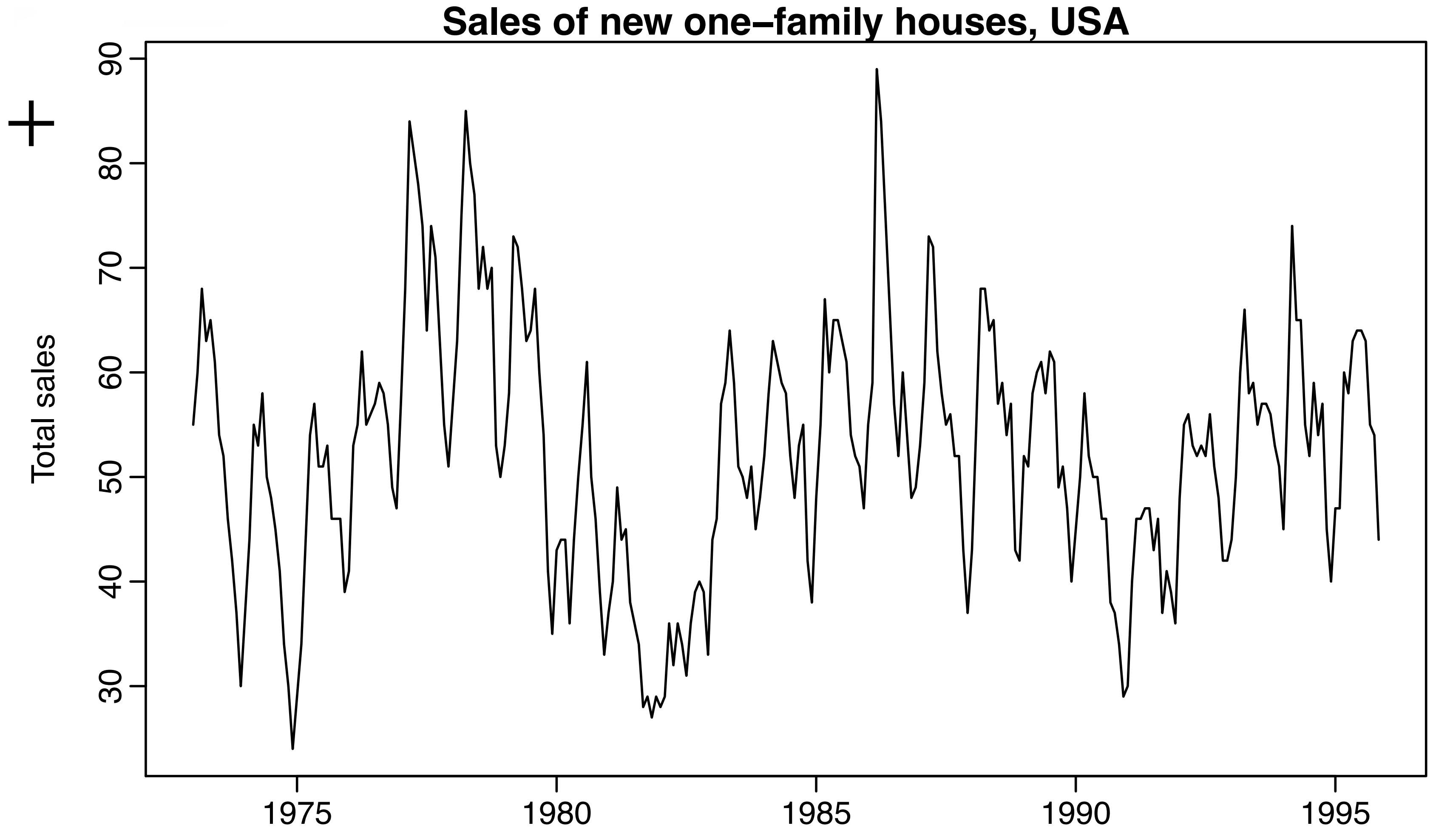
Trend

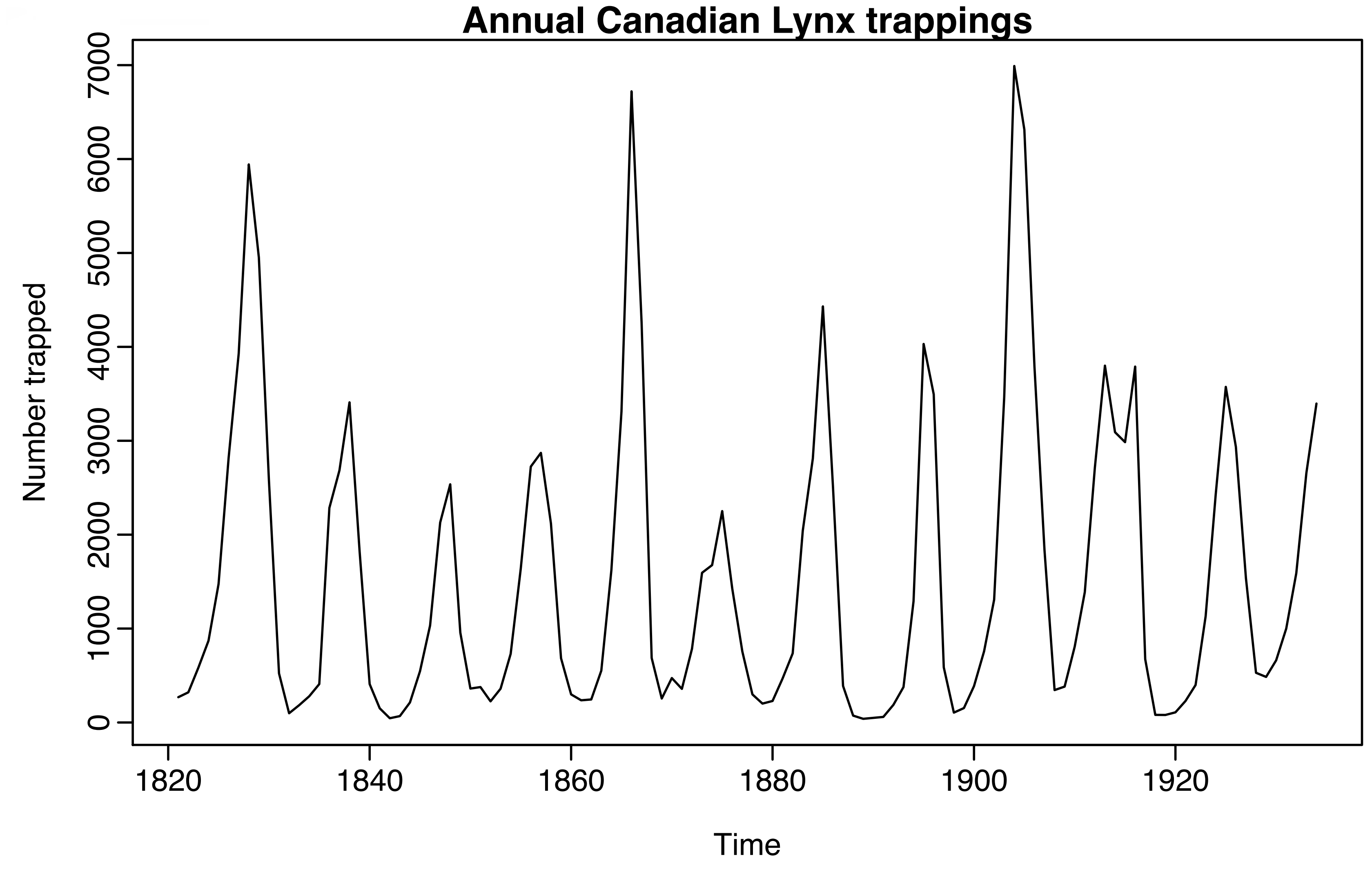



Trend + Seasonality




Seasonality + Cyclic

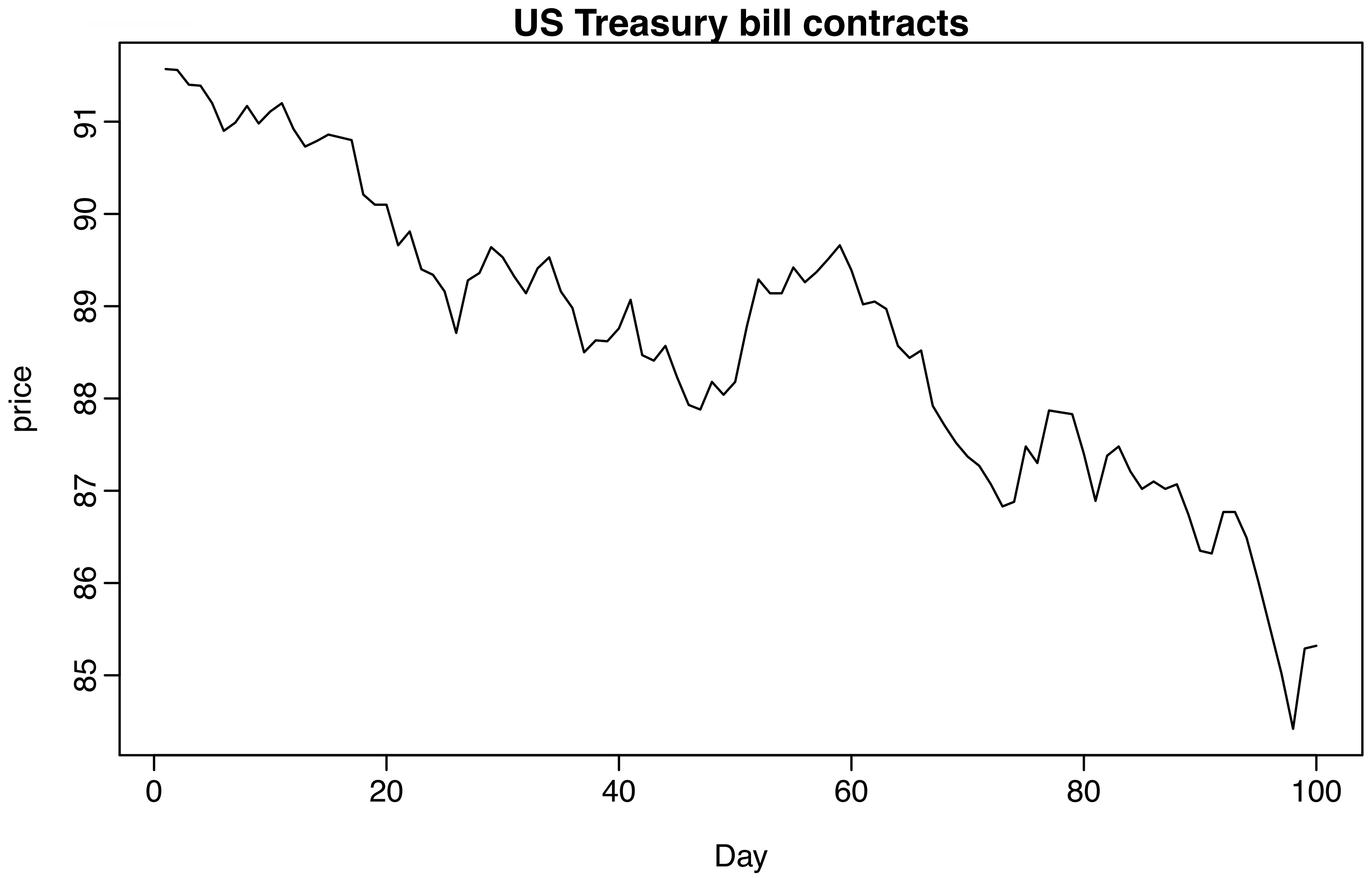


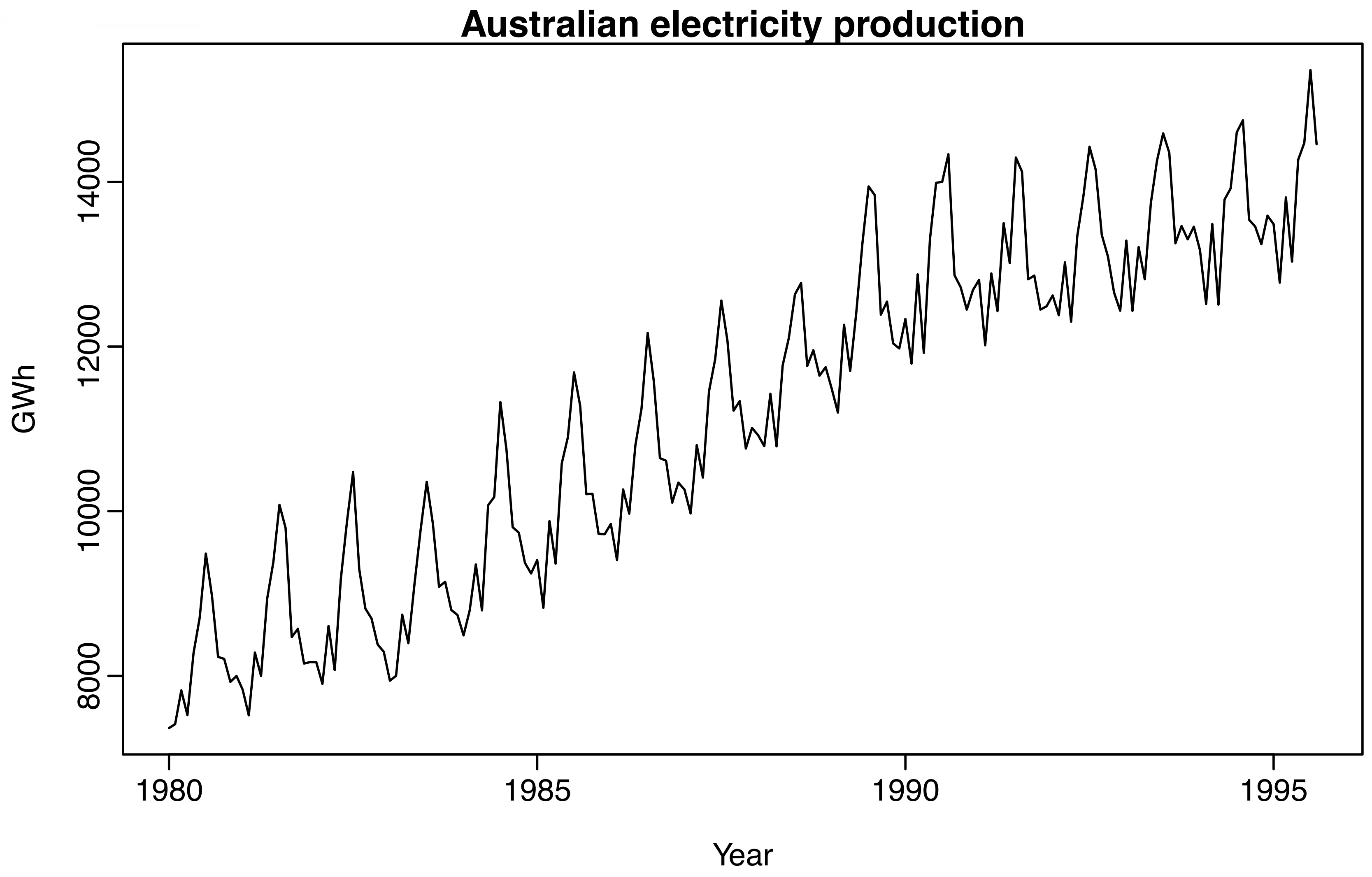




[R. J. Hyndman]

# Time Series Data

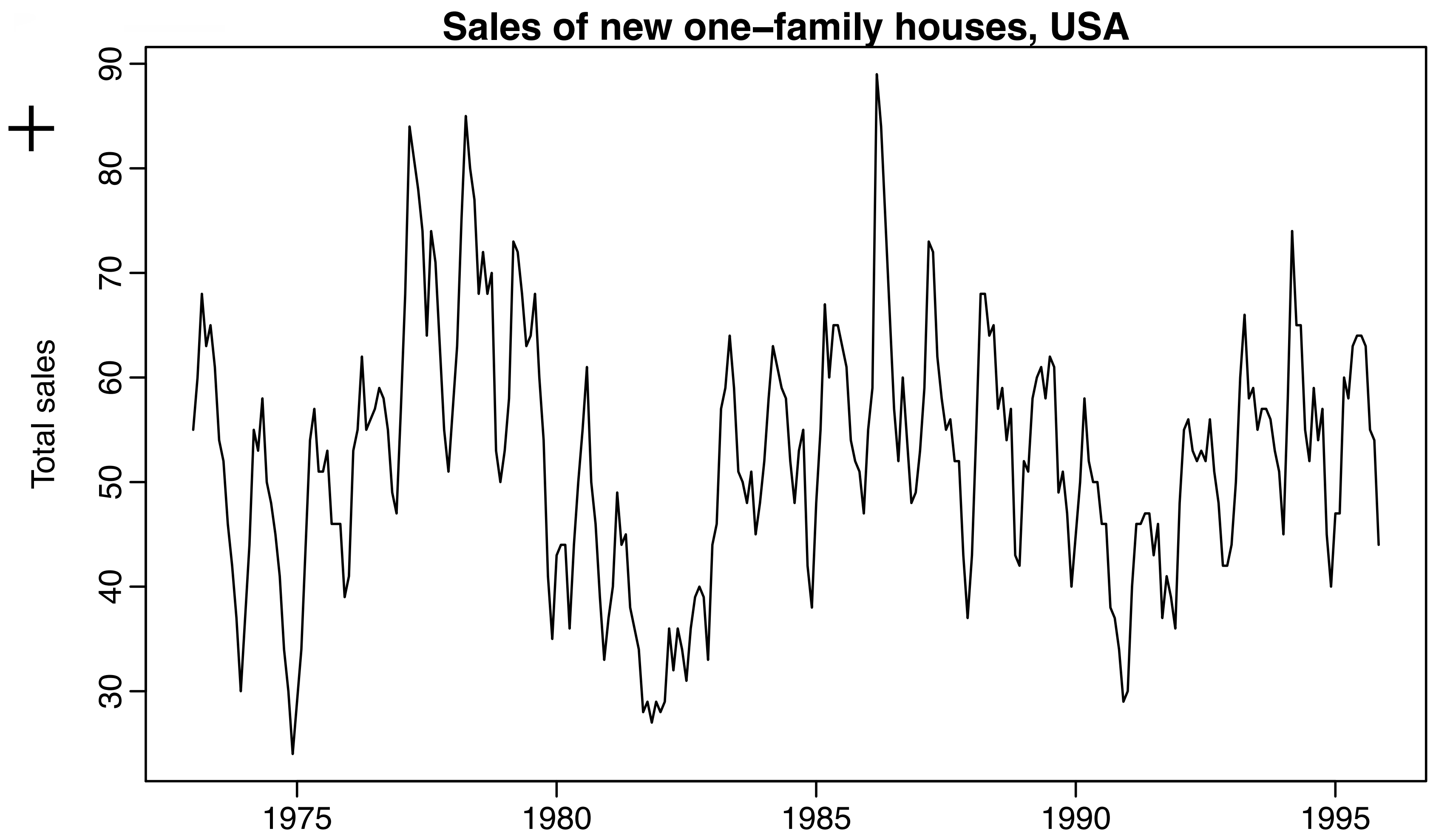
Trend

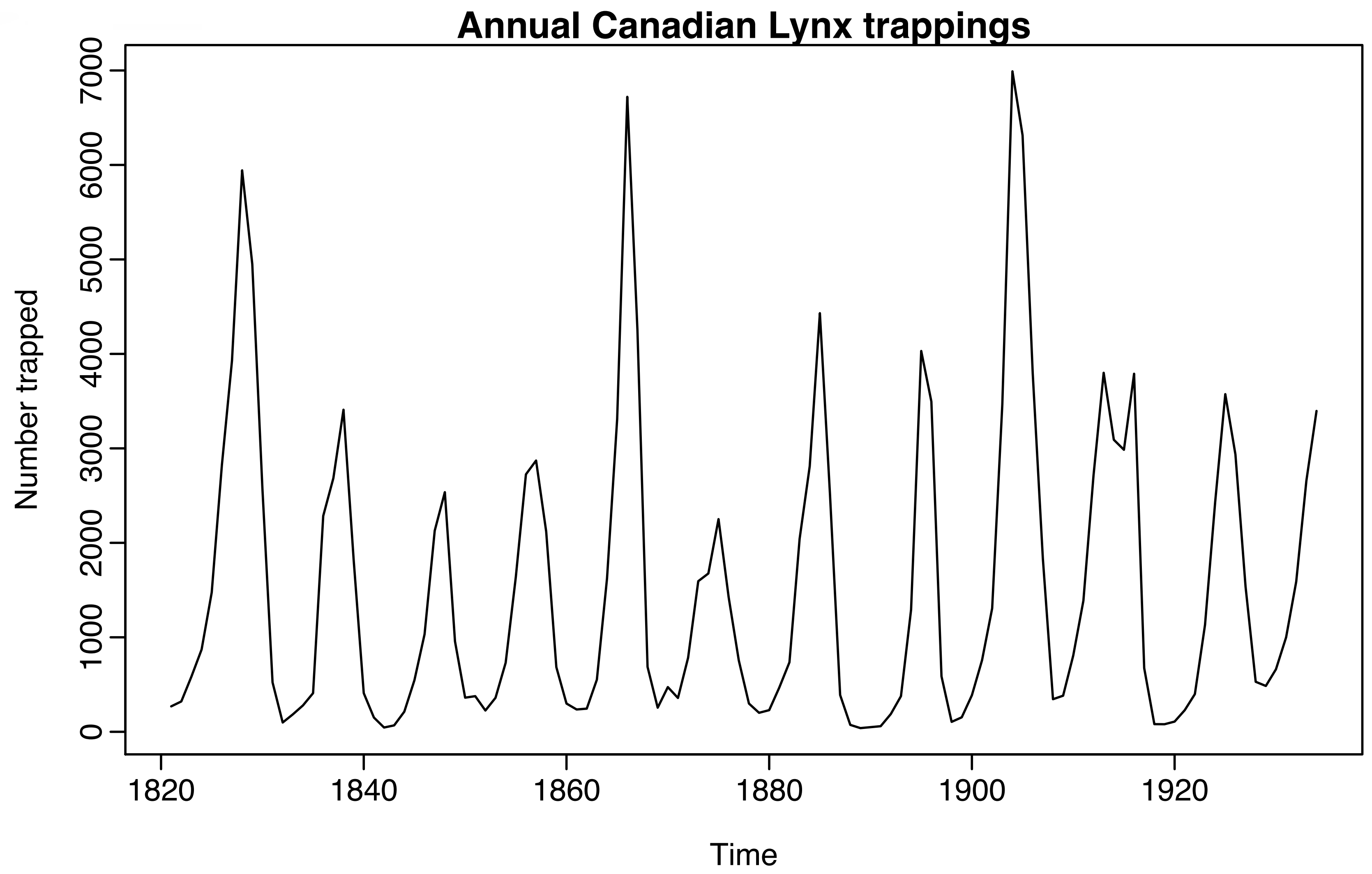



Trend +
Seasonality

Seasonality +
Cyclic

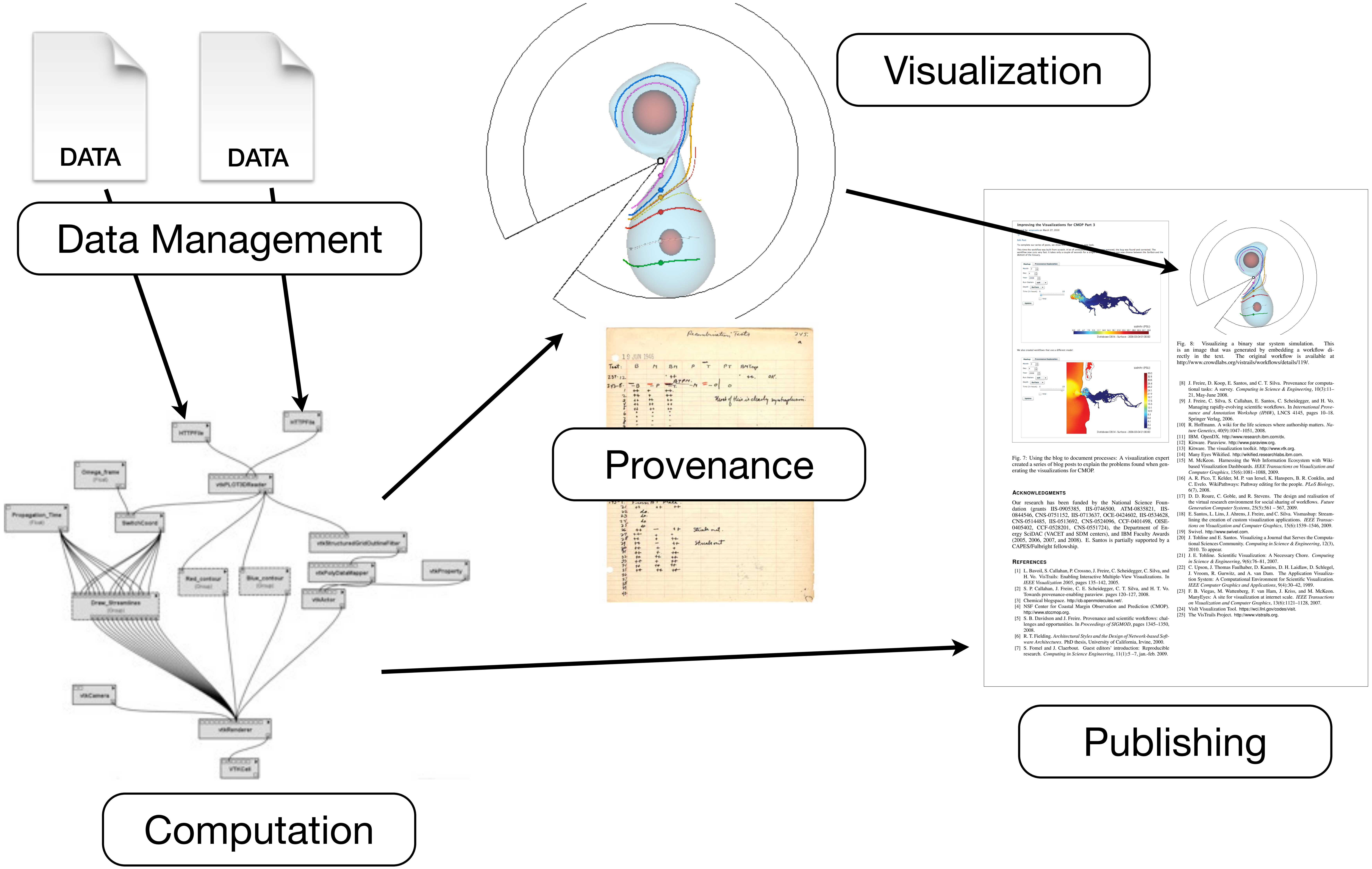



Stationary

[R. J. Hyndman]

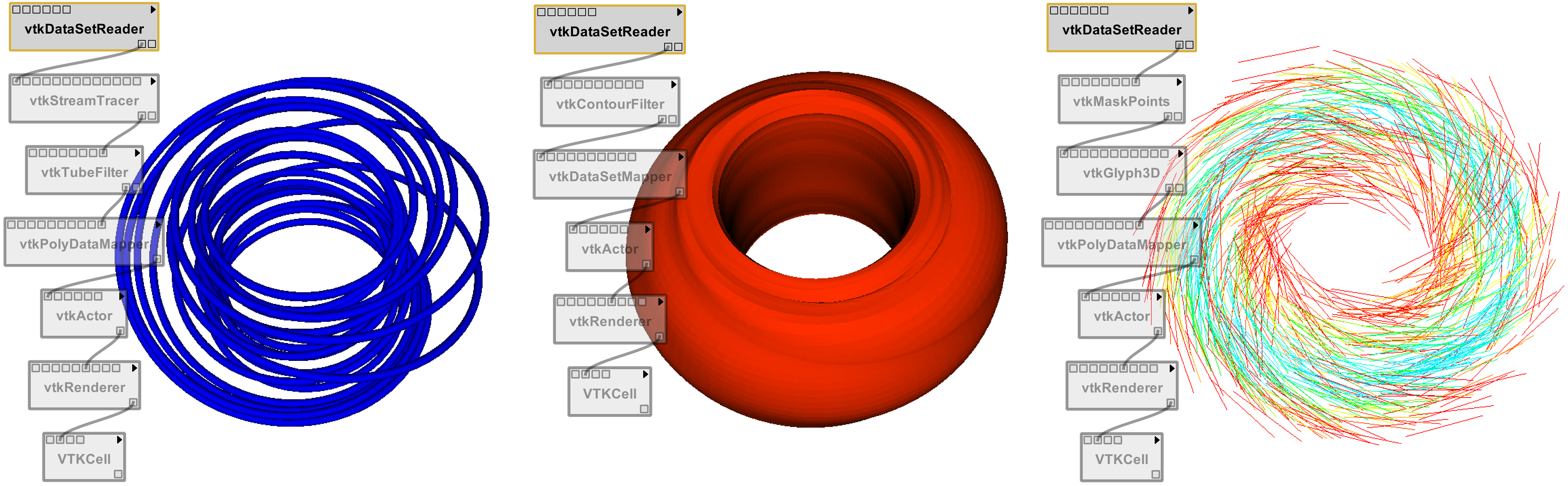
# Provenance


DATA
DATA
Data Management
Visualization
Provenance
Computation
Publishing

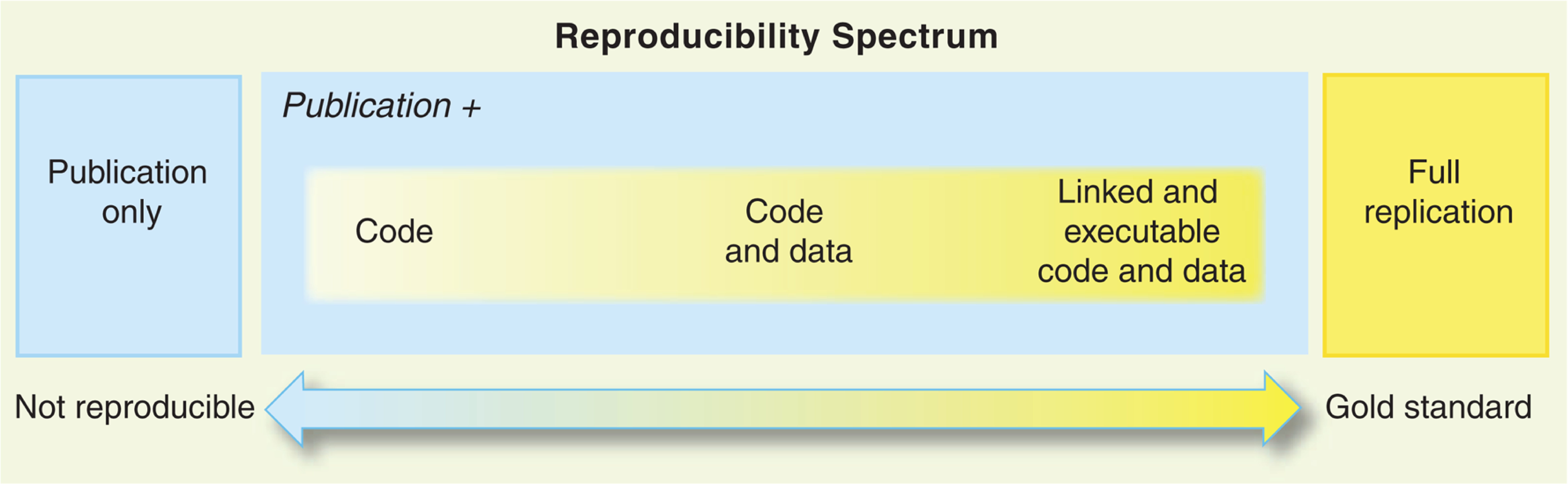
# Prospective and Retrospective Provenance

- Recipe for baking a cake versus the actual process & outcome
- Prospective provenance is what was specified/intended
  - a workflow, script, list of steps
- Retrospective provenance is what actually happened
  - actual data, actual parameters, errors that occurred, timestamps, machine information
- **Do not need** prospective provenance to have retrospective provenance!

# Using Provenance

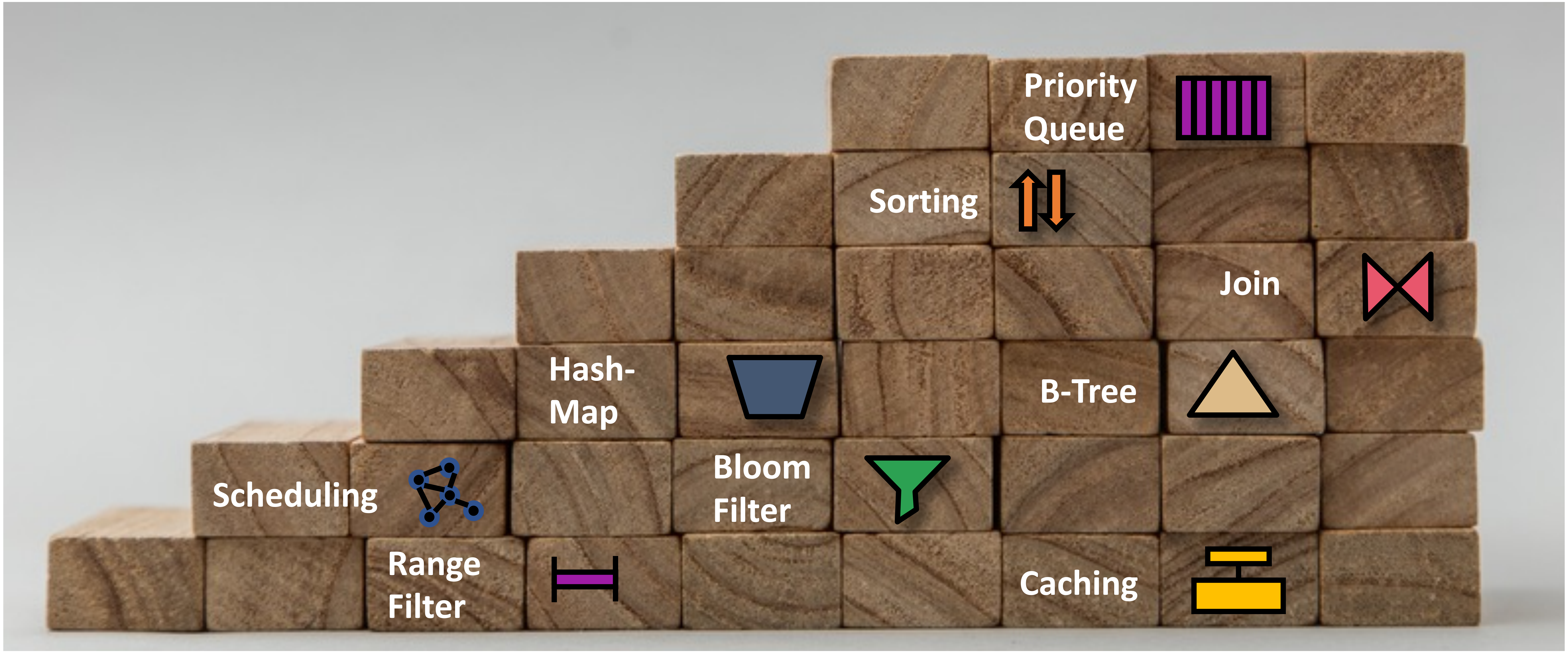


# Reproducibility




[R. D. Peng]

# Machine Learning and Databases




[T. Kraska, 2019]

Questions?

# Final Exam

- Monday, April 26, 4:00-5:50pm, Online (Blackboard)
- Similar format
- More comprehensive (questions from topics covered in Test 1 & 2)
- Will also have questions from temporal data, provenance, reproducibility, machine learning