Advanced Data Management (CSCI 490/680)

Review

Dr. David Koop
Data systems rely on algorithms

[S. Idreos, 2019]
Data structures define performance

As time goes by, data structures become ever more critical for data driven applications.

Jim Gray, Turing Award 1998

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

[S. Idreos, 2019]
Tradeoffs in each structure

no perfect structure

Every data structure design is simply a point in the design space of possible solutions. There is no perfect design. Every design balances the fundamental tradeoffs of Read, Update, and Memory amplification. For example, Read amplification is defined as the excess data an algorithm needs to read on top of the data it wants to read. Typically a data structure would have some kind of metadata or navigation data that help locate the actual data, e.g., the internal nodes of a B-tree. Reading this navigation data is an excess cost, adding to read amplification. Creating a data structure without any navigation data would suffer update or even more read amplification. For example, we could choose to not have any structure in the data at all. Then every query would have to touch all the data. The other extreme would be to sort all data which effectively provides an implicit structure. But then updates get expensive. Overall, there is no perfect design.

[S. Idreos, 2019]
"Traditional" Database Research

Many efforts in the field have been motivated by the vision of generating tailored systems for a specific scenario. In fact, even traditional databases are architected with this vision in mind. A generic database system can optimize a plan on the fly to match the query needs, it can choose from different storage and indexing options, etc. This is how generic database systems can be used in a wealth of applications! And then recent research has tried to push the boundaries of tailored designs by rethinking parts of the stack of a database system.

[S. Idreos, 2019]
Learned Data Structures and Algorithms
B-Tree

Key
(e.g., spoon #1)

Model

[T. Kraska, 2019]
Model to Predict Data's Location on Disk

Frequency Distribution

Cumulative Distribution Function (CDF)

P(X<2017-11-27) * N

[T. Kraska, 2019]
Traditional model architectures do not work

Frameworks are not designed for nano-second execution

Overfitting can be good

ML+System Co-Design

Challenges

[Overfitting can be good]

[ML+System Co-Design]

[T. Kraska, 2019]
Recursive Model Index (RMI)

2-Stage RMI with Linear Model

\[ \text{pos}_0 = a_0 + b_0 \times \text{key} \]
\[ \text{pos}_1 = m_1[\text{pos}_0].a + m_1[\text{pos}_0].b \times \text{key} \]
\[ \text{record} = \text{local-search}(\text{key}, \text{pos}_1) \]
Sandwiched Bloom Filter

Is This **Key** In My Set?

- **Yes**
  - Model
  - Maybe No
  - No

- **Maybe**
  - Maybe Yes

- **No**

[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

32-bit ints; normal distribution (μ=0, σ=1e6)

Running time (sec.)

0M 10M 20M 30M 40M 50M

std:sort  Radix sort  Timsort
Learned Sort (pre-trained)  Learned Sort

[T. Kraska, 2019]
More...

Tree  Multi-Dim Index  Bloom-Filter  Sorting  Scheduling  Range-Filter  Hash-Map

Data Cubes  DNA-Search  SQL Query Optimizer  Cache Policy  Join  Nearest Neighbor

[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?

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

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

• Just assign expressions to variables, no typing
  
  ```python
  a = 12
  a = "abc"
  b = a + "de"
  ```

• Functions defined using `def`, called using parenthesis:
  
  ```python
def hello(name1="Joe", name2="Jane"):
    print(f"Hello {name1} and {name2}"
  hello(name2="Mary")
  ```

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

Figure 4-1. Indexing elements in a NumPy array

In multidimensional arrays, if you omit later indices, the returned object will be a lower dimensional ndarray consisting of all the data along the higher dimensions. So in the $2 \times 2 \times 3$ array $\text{arr3d}$:

- $\text{arr3d}[0]$ is a $2 \times 3$ array:
  - $\text{old_values} = \text{arr3d}[0].\text{copy}()$
  - $\text{arr3d}[0] = 42$
  - $\text{arr3d}[0]$ = $\text{old_values}$

Both scalar values and arrays can be assigned to $\text{arr3d}[0]$:

Figure 4-2. Two-dimensional array slicing

Suppose each name corresponds to a row in the $\text{data}$ array and we wanted to select all the rows with corresponding name 'Bob'. Like arithmetic operations, comparisons (such as $==$) with arrays are also vectorized. Thus, comparing names with the string 'Bob' yields a boolean array:

- $\text{names} == 'Bob'$
- $\text{data}[\text{names} == 'Bob']$

This boolean array can be passed when indexing the array:

- $\text{data}[\text{names} == 'Bob', 2:]$

The boolean array must be of the same length as the axis it's indexing. You can even mix and match boolean arrays with slices or integers (or sequences of integers, more on this later):

- $\text{data}[\text{names} == 'Bob', 2:,:]$

[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

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

quantitative
ordinal
categorical
Pandas and Data Frames

- 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

```
import pandas as pd
import numpy as np

def read_csv(file_name):
    df = pd.read_csv(file_name)
    return df.head()

df = read_csv('Food_Inspections.csv')
print(df)
```

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

## Preview

<table>
<thead>
<tr>
<th>IMSI</th>
<th>CONTRACT_END</th>
<th>CONTRACT_START</th>
<th>SUBSCRIBER_AGE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3104</td>
<td>Jan 2013 - Dec 2016</td>
<td>Jan 2000 - Dec 2014</td>
<td>0 - 15</td>
<td></td>
</tr>
<tr>
<td>3101</td>
<td>6/4/16</td>
<td>7/29/89</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3102</td>
<td>3/28/15</td>
<td>18/6/13</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3103</td>
<td>9/23/16</td>
<td>1/9/87</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3104</td>
<td>5/29/15</td>
<td>2/14/81</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3105</td>
<td>9/11/15</td>
<td>9/18/10</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3106</td>
<td>8/27/15</td>
<td>3/13/86</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3107</td>
<td>1/16/16</td>
<td>5/11/84</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3108</td>
<td>8/5 - July 2011</td>
<td>9/11/86</td>
<td></td>
<td>INACTIVE</td>
</tr>
<tr>
<td>3109</td>
<td>12/24/15</td>
<td>3/28/81</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3110</td>
<td>3/6/15</td>
<td>7/26/80</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3111</td>
<td>9/25/15</td>
<td>4/4/84</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3112</td>
<td>4/30/16</td>
<td>9/8/84</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3113</td>
<td>11/16/15</td>
<td>11/3/84</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3114</td>
<td>8/13/13</td>
<td>11/23/80</td>
<td></td>
<td>INACTIVE</td>
</tr>
<tr>
<td>3115</td>
<td>8/4/16</td>
<td>18/22/14</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3116</td>
<td>1/22/15</td>
<td>18/19/14</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3117</td>
<td>11/21/15</td>
<td>12/8/14</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3118</td>
<td>27 - Sep - 2011</td>
<td>2/9/89</td>
<td></td>
<td>INACTIVE</td>
</tr>
<tr>
<td>3119</td>
<td>5/29/15</td>
<td>3/29/85</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3120</td>
<td>11/17/16</td>
<td>5/21/87</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3121</td>
<td>9/15/16</td>
<td>7/24/11</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3122</td>
<td>2/27/15</td>
<td>6/29/11</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3123</td>
<td>4/28/16</td>
<td>4/15/84</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3124</td>
<td>2/7/15</td>
<td>3/24/12</td>
<td></td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3125</td>
<td>13 - Jan - 2009</td>
<td>12/18/85</td>
<td></td>
<td>INACTIVE</td>
</tr>
<tr>
<td>3126</td>
<td>18/1/16</td>
<td>18/25/11</td>
<td></td>
<td>ACTIVE</td>
</tr>
</tbody>
</table>

## Pattern Details

**CONTRACT_END**

- **Hide Example Values**
  - 12.65k
  - 9/19/15
  - 6/13/15
  - 5/21/15
  - 12/13/15
  - 1/16/16
  - 5.37k
  - 9/19/15
  - 4/4/15
  - 12/8/16
  - 7/2/14
  - 11/6/15
  - 1.99k

---

D. Koop, CSCI 680/490, Spring 2021
Foofah: Programming by Example

1. **Transformations Targeted:**
   - Example
     - L

2. **Background Knowledge for Data Transformation**
   - Our goal: minimize depth of data transformation knowledge from the user.
   - Most of the users will not need to understand the underlying data transformation algorithms.
   - Most of the users do not want to view the output of the transformation process.
   - Most of the users do not want to view the output of the transformation process but rather demonstrate correct output.

3. **Data Transformation Tasks:**
   - Raw Data
   - Transformed Data
   - Synthesized Data Transformation Program
   - Example Input
   - Example Solution
   - Batching:
     - Example Solution
     - Example Solution
   - Most data transformation operations can be seen as many small operators:
     - Add/Remove/Move/Transform
     - Distance

4. **A* Search:**
   - A* search: iteratively explore the space of possible solutions.
   - The algorithm maintains a set of candidate solutions, ranking them by an estimate of the cost to reach the goal.
   - At each step, the algorithm selects the node with the lowest estimated cost to reach the goal.
   - The estimated cost is calculated as the sum of the cost to reach the node (g(n)) and an estimate of the cost to reach the goal from the node (h(n)).
   - The algorithm then expands the selected node and repeats the process.

5. **Related Work:**
   - Z. Jin et al., 2017
   - Smith et al., 2018
   - Barowy et al., 2019

6. **Comparison:**
   - Task completion time: Wrangler vs. Foofah
   - Performance metrics:
     - Success rate
     - Failure rate
     - Efficiency (execution time)
     - Performance gain over previous methods

7. **Conclusion:**
   - Foofah provides a user-friendly interface for data transformation tasks.
   - It allows users to focus on the desired output without needing to understand the underlying algorithms.
   - Foofah is particularly useful for tasks that are repetitive or involve complex data manipulation.
   - Future work includes integrating more advanced data transformation algorithms and improving the user interface.
TDE: Transform Data by Example

In a separate scenario, suppose one would like identified

This data in Figure 1 is clearly not ready for analysis

The overarching goal of TDE works like a search engine, which

In a separate scenario, suppose one would like identified

The first two values are provided as output examples to produce city, state, and zip-code. Note that some of

Figure 3 shows additional examples for trans-

Figure 2: (Left): transformation for names. The first three values in column-D are provided as output examples. The desired first-names and last-names are marked in bold for ease of reading. A composed

Figure 2: (Left): transformation for names. The first three values in column-D are provided as output examples. The desired first-names and last-names are marked in bold for ease of reading. A composed

Figure 2: (Left): transformation for names. The first three values in column-D are provided as output examples. The desired first-names and last-names are marked in bold for ease of reading. A composed

Figure 2: (Left): transformation for names. The first three values in column-D are provided as output examples. The desired first-names and last-names are marked in bold for ease of reading. A composed

Figure 2: (Left): transformation for names. The first three values in column-D are provided as output examples. The desired first-names and last-names are marked in bold for ease of reading. A composed

Figure 2: (Left): transformation for names. The first three values in column-D are provided as output examples. The desired first-names and last-names are marked in bold for ease of reading. A composed

Figure 2: (Left): transformation for names. The first three values in column-D are provided as output examples. The desired first-names and last-names are marked in bold for ease of reading. A composed

Figure 2: (Left): transformation for names. The first three values in column-D are provided as output examples. The desired first-names and last-names are marked in bold for ease of reading. A composed
# Tidy Data

The experimental design also determines whether or not missing values can be safely imputed every combination of attributes.

The experimental design tells us more about the structure of the observations. In this experiment, every combination of attributes creates an observation.

- **height**: Height of a person.
- **temperature**: Temperature of a day.
- **duration**: Duration of a race.

A dataset is a collection of values that measure the same underlying attribute. A variable contains all values that measure the same underlying attribute. A variable contains all values that measure the same underlying attribute (like height, temperature, duration) across units. An observation contains all values measured on the same unit (like a person, or a day, or a race) across attributes.

Most statistical datasets are rectangular tables made up of columns and three rows, and both rows and columns are labelled.

There are many ways to structure the same underlying data. Table 1 shows the initial data, Table 2 shows the same data after it has been transposed. The data is the same, but the way that the values, variables and observations are described is different. Our vocabulary of rows and columns is simply not rich enough to describe why the two tables represent the same data. In addition to appearance, we need a way to describe the underlying semantics, or meaning, of the values displayed in table.

## Data structure

A dataset contains 18 values representing three variables and six observations. The variables are: name, treatmenta, treatmentb, result. Each type of observational unit forms a table. Each observation forms a row.

### Table 1: Typical presentation dataset.

<table>
<thead>
<tr>
<th>Name</th>
<th>treatmenta</th>
<th>treatmentb</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td>—</td>
<td>2</td>
</tr>
<tr>
<td>Jane Doe</td>
<td>16</td>
<td>11</td>
</tr>
<tr>
<td>Mary Johnson</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 2: The same data as in Table 1, but the rows and columns have been transposed. The data is the same, but the way that the values, variables and observations are described is different.

<table>
<thead>
<tr>
<th>Name</th>
<th>trt</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td>a</td>
<td>—</td>
</tr>
<tr>
<td>Jane Doe</td>
<td>a</td>
<td>16</td>
</tr>
<tr>
<td>Mary Johnson</td>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>John Smith</td>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>Jane Doe</td>
<td>b</td>
<td>11</td>
</tr>
<tr>
<td>Mary Johnson</td>
<td>b</td>
<td>1</td>
</tr>
</tbody>
</table>

### Initial Data

<table>
<thead>
<tr>
<th>John Smith</th>
<th>Jane Doe</th>
<th>Mary Johnson</th>
</tr>
</thead>
<tbody>
<tr>
<td>treatmenta</td>
<td>—</td>
<td>16</td>
</tr>
<tr>
<td>treatmentb</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>result</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

### Tidy Data

<table>
<thead>
<tr>
<th>name</th>
<th>trt</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td>a</td>
<td>—</td>
</tr>
<tr>
<td>Jane Doe</td>
<td>a</td>
<td>16</td>
</tr>
<tr>
<td>Mary Johnson</td>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>John Smith</td>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>Jane Doe</td>
<td>b</td>
<td>11</td>
</tr>
<tr>
<td>Mary Johnson</td>
<td>b</td>
<td>1</td>
</tr>
</tbody>
</table>

[H. Wickham, 2014]
MultiIndex Row Access and Slicing

- 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

<table>
<thead>
<tr>
<th>Id</th>
<th>Location</th>
<th>Date</th>
<th>Home</th>
<th>Away</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Boston</td>
<td>9/2</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>Boston</td>
<td>9/9</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Cleveland</td>
<td>9/16</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>San Diego</td>
<td>9/23</td>
<td>21</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>wId</th>
<th>City</th>
<th>Date</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Boston</td>
<td>9/2</td>
<td>72</td>
</tr>
<tr>
<td>1</td>
<td>Boston</td>
<td>9/3</td>
<td>68</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>Boston</td>
<td>9/9</td>
<td>75</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>21</td>
<td>Boston</td>
<td>9/23</td>
<td>54</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>36</td>
<td>Cleveland</td>
<td>9/16</td>
<td>81</td>
</tr>
</tbody>
</table>

No data for San Diego
## Inner Strategy

### Merged

<table>
<thead>
<tr>
<th>Id</th>
<th>Location</th>
<th>Date</th>
<th>Home</th>
<th>Away</th>
<th>Temp</th>
<th>wId</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Boston</td>
<td>9/2</td>
<td>1</td>
<td>15</td>
<td>72</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Boston</td>
<td>9/9</td>
<td>1</td>
<td>7</td>
<td>75</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Cleveland</td>
<td>9/16</td>
<td>12</td>
<td>1</td>
<td>81</td>
<td>36</td>
</tr>
</tbody>
</table>

No San Diego entry
## Outer Strategy

### Merged

<table>
<thead>
<tr>
<th>Id</th>
<th>Location</th>
<th>Date</th>
<th>Home</th>
<th>Away</th>
<th>Temp</th>
<th>wId</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Boston</td>
<td>9/2</td>
<td>1</td>
<td>15</td>
<td>72</td>
<td>0</td>
</tr>
<tr>
<td>NaN</td>
<td>Boston</td>
<td>9/3</td>
<td>NaN</td>
<td>NaN</td>
<td>68</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>Boston</td>
<td>9/9</td>
<td>1</td>
<td>7</td>
<td>75</td>
<td>7</td>
</tr>
<tr>
<td>NaN</td>
<td>Boston</td>
<td>9/10</td>
<td>NaN</td>
<td>NaN</td>
<td>76</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>NaN</td>
<td>Cleveland</td>
<td>9/2</td>
<td>NaN</td>
<td>NaN</td>
<td>61</td>
<td>22</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>Cleveland</td>
<td>9/16</td>
<td>12</td>
<td>1</td>
<td>81</td>
<td>36</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>San Diego</td>
<td>9/23</td>
<td>21</td>
<td>1</td>
<td>NaN</td>
<td>NaN</td>
</tr>
</tbody>
</table>
Data Integration

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

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

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

D. Koop, CSCI 680/490, Spring 2021
Information Integration

![Diagram showing information integration processes]

Source A

Source B

Schema Mapping

Data Transformation

Duplicate Detection

Data Fusion

[<pub> <Titel> Federated Database Systems </Titel> <Autoren> <Autor> Amit Sheth </Autor> <Autor> James Larson </Autor> </Autoren> </pub>]

[<publication> <title> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </title> <author> Scheth & Larson </author> <year> 1990 </year> </publication>]

[L. Dong and F. Naumann, 2009]
Information Integration

Source A

Source B

Schema Mapping

Data Transformation

Duplicate Detection

Data Fusion

[L. Dong and F. Naumann, 2009]
Information Integration

Source A

Source B

Transformation queries or views

Schema Mapping

Data Transformation

Duplicate Detection

Data Fusion

[L. Dong and F. Naumann, 2009]
Information Integration

Source A

<Titel> Federated Database Systems </Titel>

<Autor> Amit Sheth </Autor>

<autor> James Larson </author>

</pub>

Source B

<publication>

<title> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </title>

<author> Scheth & Larson </author>

<year> 1990 </year>
</publication>

Schema Mapping
Data Transformation
Duplicate Detection
Data Fusion

[Source A]

[Source B]

[L. Dong and F. Naumann, 2009]
Information Integration

- **Source A**
  - Federated Database Systems

- **Source B**
  - Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases

- **Preserve lineage**

- **Diagram arrows**
  - Schema Mapping
  - Data Transformation
  - Duplicate Detection
  - Data Fusion

[Source: L. Dong and F. Naumann, 2009]
Figure 1: An overview of the Dataset Search components. Google crawler collects the metadata from the Web; Dataset Search backend normalizes and reconciles the metadata; we then index the reconciled metadata and rank results for user queries.

We then look for the triples that use our vocabularies of interest, Schema.org and DCAT. Specifically, we collect all the triples for all the pages that have elements of specific types: http://schema.org/Dataset, http://schema.org/DataCatalog, and http://www.w3.org/ns/dcat#Dataset.

For a set of triples from each page, we traverse the graph to collect all the properties and related objects for each dataset in a protocol buffer. A dataset record can point to other records such as organizations that provided a dataset or a record describing the distribution of a dataset. A single Web page can have multiple dataset records on it.

The specification of the graph traversal captures the mapping from Schema.org and DCAT vocabularies to the corresponding elements in the protocol buffer definition (e.g., example fields in Figure 2). The schema of the protocol buffer for the metadata largely corresponds to http://schema.org/Dataset and therefore the transformation of metadata at this stage is rather small.

To improve scalability, we use the graph query independently on the triples from each individual page rather than try to extract information from a graph that includes all metadata triples on the Web. Because the links across different pages must specify objects on another page directly through a URL (e.g., a provider of this dataset on page A is described on page B), we can do this reconciliation post-hoc. So, essentially, each page corresponds to its own, possibly disconnected graph. At the same time, doing graph traversal only for a single page is dramatically more scalable.

The information that we extract through graph traversal constitutes the raw metadata, metadata that closely mimics the structure of Schema.org properties on the original page. In the next few steps, we describe how we create reconciled metadata for each dataset, accounting for the different levels of quality and variety of the modeling patterns used.

5.2 Normalizing and cleaning the metadata

As we mentioned in Section 4.1, we must assume that we will encounter every possible misuse and mis-interpretation of Schema.org properties when we operate at the scale of the whole Web. Thus, we perform a number of operations to normalize and clean up the metadata.

First, for the properties where we observe different patterns on the Web, we analyze the common patterns used and try to account for all of them. For instance Figure 2 shows the different patterns that we observed for defining downloads and distribution. In the figure, the first example of raw metadata defines the format of the dataset (CSV) at the level of the dataset itself and stores the download URL as the value of the http://schema.org/distribution property. Other examples in the figure deal with these two pieces of information differently. All these patterns are commonly used in our corpus. We mine these patterns by traversing either the initial graph or the resulting protocol buffer. Once we identify the patterns, we write adapters to convert all of them into the same modeling pattern in the reconciled metadata record. The right-hand side of Figure 2 shows this reconciled result.

Similarly, we have developed adapters for other metadata fields: We understand a lot more representations of dates than the ISO standard required by the Schema.org specification (Section 4.1. We will pick up digital object identifiers (DOIs) for a dataset from a variety of fields, and not just http://schema.org/identifier. We will use a uniform field, provider, for the many different fields that dataset providers used to identify this property. As we collect more metadata, our set of such adapters grows. Our decisions in these steps are guided by two factors: (1) the frequent usage patterns that we observed in the data; and (2) our understanding of what we expect the users to see in Dataset Search results.
The DCC Curation Lifecycle Model

Preservation Planning
Community Watch and Participation
Curate and Preserve
Conceptualise
Create or Receive
Appraise and Select
Ingest
Preservation Action
Store
Access, Use and Reuse
Transform
Preservation Planning
Description
Appraise & Select
Preservation Action
Ingest
Store
Access, Use & Reuse
Transform
CURATE
Data (Digital Objects or Databases)
REAPPRaise & SELECT
PRESERVE
PREServation Action
STORE
ACCESS, USE & REUSE
TRANSFORM
CONCEPTUALISE
DISPOSE

Assign administrative, descriptive, technical, structural and preservation metadata, using appropriate standards, to ensure adequate description and control over the long-term. Collect and assign representation information required to understand and render both the digital material and the associated metadata.

Plan for preservation throughout the curation lifecycle of digital material. This would include plans for management and administration of all curation lifecycle actions.

Maintain a watch on appropriate community activities, and participate in the development of shared standards, tools and suitable software.

Be aware of, and undertake management and administrative actions planned to promote curation and preservation throughout the curation lifecycle.

Conceive and plan the creation of data, including capture method and storage options.

Create data including administrative, descriptive, structural and technical metadata. Preservation metadata may also be added at the time of creation.

Receive data, in accordance with documented collecting policies, from data creators, other archives, repositories or data centres, and if required assign appropriate metadata.

Evaluate data and select for long-term curation and preservation. Adhere to documented guidance, policies or legal requirements.

Transfer data to an archive, repository, data centre or other custodian. Adhere to documented guidance, policies or legal requirements.

Undertake actions to ensure long-term preservation and retention of the authoritative nature of data. Preservation actions should ensure that data remains authentic, reliable and usable while maintaining its integrity. Actions include data cleaning, validation, assigning preservation metadata, assigning representation information and ensuring acceptable data structures or file formats.

Store the data in a secure manner adhering to relevant standards.

Ensure that data is accessible to both designated users and reusers, on a day-to-day basis. This may be in the form of publicly available published information. Robust access controls and authentication procedures may be applicable.

Create new data from the original, for example-
- By migration into a different format.
- By creating a subset, by selection or query, to create newly derived results, perhaps for publication.
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
3.2 Shared-Nothing

A shared-nothing parallel system (Figure 3.2) is made up of a cluster of independent machines that communicate over a high-speed network interconnect or, increasingly frequently, over commodity networking components. There is no way for a given system to directly access the memory or disk of another system. Shared-nothing systems provide no hardware sharing abstractions, leaving coordination of the various machines entirely in the hands of the DBMS. The most common technique employed by DBMSs to support these clusters is to run their standard process model on each machine, or node, in the cluster. Each node is capable of accepting client SQL requests.

[Parallel DB Architecture: Shared Nothing

[Fig. 3.2 Shared-nothing architecture.

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

<table>
<thead>
<tr>
<th>id</th>
<th>Title</th>
<th>Person</th>
<th>Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mrs. Doubtfire</td>
<td>Robin Williams</td>
<td>Comedy</td>
</tr>
<tr>
<td>2</td>
<td>Jaws</td>
<td>Roy Scheider</td>
<td>Horror</td>
</tr>
<tr>
<td>3</td>
<td>The Fly</td>
<td>Jeff Goldblum</td>
<td>Horror</td>
</tr>
<tr>
<td>4</td>
<td>Steel Magnolias</td>
<td>Dolly Parton</td>
<td>Drama</td>
</tr>
<tr>
<td>5</td>
<td>The Birdcage</td>
<td>Nathan Lane</td>
<td>Comedy</td>
</tr>
<tr>
<td>6</td>
<td>Erin Brokovich</td>
<td>Julia Roberts</td>
<td>Drama</td>
</tr>
</tbody>
</table>

Each column has a file or segment on disk

[J. Swanhart, Introduction to Column Stores]
CAP Theorem

Scalability: CAP Theorem

Remains accessible and operational at all times.

Availability

Traditional relational databases: PostgreSQL, MySQL, etc.

CA

Pick Two!

AP

Voldemort, Riak, Cassandra, CouchDB, Dynamo-like systems

C

Consistency
Commits are atomic across the entire distributed system.

CP

HBase
MongoDB
Redis
MemcacheDB
BigTable-like systems

P

Partition Tolerance
Only a total network failure can cause the system to respond incorrectly.
Cassandra: Replication and Consistency

[Diagram showing replication between DC 1 and DC 2]
Spanner: Google's NewSQL Cloud Database

High Availability: CAP Theorem and Cassandra

- 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
Interactive Exploration of Spatial Data

SELECT lat, lng, (b4-b6)/(b4+b6) as ndsi
FROM modis_data
WHERE ndsi > 0.7
Interactive Exploration of Spatial Data

SELECT lat, lng, (b4-b6)/(b4+b6) as ndsi
FROM modis_data
WHERE ndsi > 0.7
Spatial Data: Minimize Latency

Figure 3: Panning and zooming in a binned plot: initial view (left), zooming in (middle), panning to the lower-left (right).

Figure 4: Spatial data for dynamic visualization, not pre-rendered images.

Z. Liu et al., 2013
Spatial Data: Precompute Optimized Storage

- **Fig. 2. An illustration of how to build a nanocube for five points**

**Indexing Schema**

\[ S = (\ell_{\text{spatial1}}, \ell_{\text{spatial2}}, \ell_{\text{device}}) \]

- **\( \ell_{\text{device}}(\bigcirc) = \text{Android} \)**
- **\( \ell_{\text{device}}(\bullet) = \text{iPhone} \)**


### Five Tweets: Location and Device

<table>
<thead>
<tr>
<th>Location</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>Android</td>
</tr>
<tr>
<td>O2</td>
<td>iPhone</td>
</tr>
<tr>
<td>O3</td>
<td>iPhone</td>
</tr>
<tr>
<td>O4</td>
<td>iPhone</td>
</tr>
<tr>
<td>O5</td>
<td>iPhone</td>
</tr>
</tbody>
</table>

**\( \ell_{\text{spatial1}} \)**

<table>
<thead>
<tr>
<th>Location</th>
<th>( x )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>O2</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>O3</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>O4</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>O5</td>
<td>1.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**\( \ell_{\text{spatial2}} \)**

<table>
<thead>
<tr>
<th>Location</th>
<th>( x )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>0.1</td>
<td>1.1</td>
</tr>
<tr>
<td>O2</td>
<td>0.1</td>
<td>1.1</td>
</tr>
<tr>
<td>O3</td>
<td>0.1</td>
<td>1.1</td>
</tr>
<tr>
<td>O4</td>
<td>0.1</td>
<td>1.1</td>
</tr>
<tr>
<td>O5</td>
<td>0.1</td>
<td>1.1</td>
</tr>
</tbody>
</table>

- **1.**
  - **iPhone**
  - **Android**

- **2.**
  - **iPhone**
  - **Android**

- **3.**
  - **iPhone**
  - **Android**

- **4.**
  - **iPhone**
  - **Android**

- **5.**
  - **iPhone**
  - **Android**

**[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)

[Image of maps with ROI tiles highlighted]
Split-Apply-Combine

Aggregation of time series data, a special use case of groupby, is referred to as resampling in this book and will receive separate treatment in Chapter 10.

GroupBy Mechanics

Hadley Wickham, an author of many popular packages for the R programming language, coined the term split-apply-combine for talking about group operations, and I think that's a good description of the process. In the first stage of the process, data contained in a pandas object, whether a Series, DataFrame, or otherwise, is split into groups based on one or more keys that you provide. The splitting is performed on a particular axis of an object. For example, a DataFrame can be grouped on its rows (axis=0) or its columns (axis=1). Once this is done, a function is applied to each group, producing a new value. Finally, the results of all those function applications are combined into a result object. The form of the resulting object will usually depend on what's being done to the data. See Figure 9-1 for a mockup of a simple group aggregation.

Figure 9-1. Illustration of a group aggregation

Each grouping key can take many forms, and the keys do not have to be all of the same type:

- A list or array of values that is the same length as the axis being grouped
- A value indicating a column name in a DataFrame
Time Series Data

US Treasury bill contracts

Australian electricity production

Sales of new one-family houses, USA

Annual Canadian Lynx trappings

[R. J. Hyndman]
Time Series Data

**US Treasury bill contracts**

**Australian electricity production**

**Sales of new one-family houses, USA**

**Annual Canadian Lynx trappings**

[R. J. Hyndman]
Time Series Data

- **US Treasury bill contracts**
- **Australian electricity production**
- **Sales of new one-family houses, USA**
- **Annual Canadian Lynx trappings**

**Trend**

**Trend + Seasonality**
Time Series Data

- **US Treasury bill contracts**
  - Trend

- **Australian electricity production**
  - Trend + Seasonality

- **Sales of new one-family houses, USA**
  - Seasonality + Cyclic

- **Annual Canadian Lynx trappings**

---

[R. J. Hyndman]
Time Series Data

- **US Treasury bill contracts**
  - Trend

- **Australian electricity production**
  - Trend + Seasonality

- **Sales of new one-family houses, USA**
  - Seasonality + Cyclic

- **Annual Canadian Lynx trappings**
  - Stationary

[R. J. Hyndman]
Provenance

Data Management

Visualization

Computation

Publishing

Data

Data

Provenance

(\textit{Provenance for computational and visualizing workflows})

\textit{Provenance for computational and visualizing workflows}
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

Reproducibility Spectrum

- Publication only
- Code
- Code and data
- Linked and executable code and data
- Full replication

Not reproducible

Gold standard

[R. D. Peng]
Machine Learning and Databases
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