Advanced Data Management (CSCI 640/490)

Review

Dr. David Koop
Data systems rely on algorithms.

[S. Idreos, 2019]
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
Tradeoffs in each structure

no perfect structure

Read
Update
Memory

Read amplification

Update

Memory amplification

[S. Idreos, 2019]
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.

"Traditional" Database Research

[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

Date

#Orders

Date

Probability

[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

Underfitting | Desired | Overfitting | Desired

T. Kraska, 2019
Recursive Model Index (RMI)

2-Stage RMI with Linear Model

\[
\begin{align*}
pos_0 &= a_0 + b_0 \times \text{key} \\
pos_1 &= m_1[pos_0].a + m_1[pos_0].b \times \text{key} \\
\text{record} &= \text{local-search}(\text{key, } pos_1)
\end{align*}
\]

[T. Kraska, 2019]
Is This **Key** In My Set?

- No
- Maybe
- Yes

Model

- Maybe
- Yes
- No

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 ($\mu=0$, $\sigma=1e6$)

Running time (sec.)

<table>
<thead>
<tr>
<th>0M</th>
<th>10M</th>
<th>20M</th>
<th>30M</th>
<th>40M</th>
<th>50M</th>
</tr>
</thead>
<tbody>
<tr>
<td>std:sort</td>
<td>Radix sort</td>
<td>Timsort</td>
<td>Learned Sort (pre-trained)</td>
<td>Learned Sort</td>
<td></td>
</tr>
</tbody>
</table>

[T. Kraska, 2019]
More...

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

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

[13]

T. Kraska, 2019
Query Optimization

Query Optimization

SQL
Parser
Query Optimizer
TCNN
Reward
Predictions
Execution Engine
User provided
Query plan
External component
Bao

Estimating the run-time for each query plan:

For solving contextual multi-armed bandit problems. This unique application issues many such queries. Thus, Bao is ideally suited to

In summary, the key contributions of this paper are:

• Significantly outperform traditional optimizers while training and

Commercial systems in cost and latency, all while adapting to

changes in workload, data, and schema.

We introduce Bao, a learned system for query optimization

that achieves orders of magnitude better performance than the underlying optimizer. Bao is particularly effective for

workloads that are tail-dominated (e.g., 80% of query processing

is a feature vector). These vector trees are fed into Bao's value model,

but all of the distinct operator

types

are known ahead of time.

If we just wanted to execute

a table

and the selected

plan with the best predicted performance, we would

predictive model improves, and Bao more reliably picks the best set

of hints for each query.

The architecture of Bao is shown in Figure 2.

Generating

A neural network operator that can recognize important patterns in

increased optimization time can be an issue, especially if the ap-

ditional optimization time. However, for very short running queries,

latency of the plan selected by Bao often greatly exceeds the addi-

onal runtime.

4 SYSTEM MODEL

3 SELECTING QUERY HINTS

Assumptions and Limitations

Bao also assumes a user-de

ed upfront. Note that one set of hints could be

and Bao is not always able to

changes in workload, data, and schema.

While some hints can be applied to a single relation or

such hints. While some hints can be applied to a single relation or

Only a single hint set Bao has

case-by-case basis.

On a high-level, Bao combines a tree convolution model

[3626x-2428]

[R. Marcus et al., 2021]
Final Exam

- Wednesday, May 10, **8:00**-9:50am, PM 253
- Similar format
- More comprehensive (questions from topics covered in Test 1 & 2)
- Will also have questions from graph/spatial/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 Acquisition</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>

Fig. 3.1 The Big Data Value Chain as described within (Curry et al., 2014)

[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
Relational Algebra

• Definition: A procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result.

• Six basic operators
  - select: $\sigma$
  - project: $\Pi$
  - union: $\cup$
  - set difference: $-$
  - Cartesian product: $\times$
  - rename: $\rho$

[A. Silberschatz et al.]
Components of SQL

- **Data Definition Language (DDL):** the specification of information about relations, including schema, types, integrity constraints, indices, storage.

- **Data Manipulation Language (DML):** provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.

- An SQL relation is defined using the create table command:

```
create table r (A_1 D_1, A_2 D_2, ..., A_n D_n, (C_1), ..., (C_k))
```

- A typical SQL query has the form:

```
select A_1, A_2, ..., A_n
from r_1, r_2, ..., r_m
where P
```

- $A_i$ is an attribute
- $D_i$ is the data type
- $r_i$ represents a relation
- $P$ is a predicate

[A. Silberschatz et al.]
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 $\text{arr3d}$:

```
arr3d = np.array([[[1, 2, 3], [4, 5, 6]],
                  [[7, 8, 9], [10, 11, 12]]])
```

```
In [77]: arr3d[0]
Out[77]:
array([[1, 2, 3],
       [4, 5, 6]])
```

```
In [78]: arr3d[0][0]
Out[78]:
array([[1]])
```

```
In [79]: arr3d[0][0] = 42
```

```
In [80]: arr3d[0][0]
Out[80]:
array([42])
```

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

```
old_values = arr3d[0].copy()
```

```
In [81]: arr3d[0] = 42
```

```
In [82]: arr3d[0]
Out[82]:
array([42, 42, 42])
```

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:

```
In [87]: names == 'Bob'
Out[87]:
array([ True, False, False, True, False, False, False], dtype=bool)
```

This boolean array can be passed when indexing the array:

```
In [88]: data[names == 'Bob']
Out[88]:
array([[ 0.1913,  0.4544,  0.4519,  0.5535],
       [ 0.5994,  0.8174, -0.9297, -1.2564]])
```

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):

```
In [89]: data[names == 'Bob', 2:]
Out[89]:
array([[ 0.5535],
       [ 0.8174]])
```

[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:

```
In [87]: names == 'Bob'
Out[87]:
array([ True, False, False, True, False, False, False], dtype=bool)
```

This boolean array can be passed when indexing the array:

```
In [88]: data[names == 'Bob']
Out[88]:
array([[ 0.1913,  0.4544,  0.4519,  0.5535],
       [ 0.5994,  0.8174, -0.9297, -1.2564]])
```

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):

```
In [89]: data[names == 'Bob', 2:]
Out[89]:
array([[ 0.5535],
       [ 0.8174]])
```

[Figure 4-2. Two-dimensional array slicing]
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**
  - Attributes (columns)
  - Items (rows)
  - Cell containing value

- **Networks**
  - Link
  - Node (item)

- **Fields (Continuous)**
  - Grid of positions
  - Cell
  - Attributes (columns)
  - Value in cell

- **Geometry (Spatial)**
  - Position

- **Multidimensional Table**
  - Key 1
  - Key 2
  - Attributes
  - Value in cell

- **Trees**
Categorial, Ordinal, and Quantitative

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
<th>T</th>
<th>U</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>Small Pack</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>Large Box</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>Medium Box</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>Small Box</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>Medium Box</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>Large Box</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>5/1/06</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>166</td>
<td>9/12/07</td>
<td>2-High</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
FINDINGS

we got about the future of data science, the most salient takeaway was how excited our respondents were about the evolution of the field. They cited things in their own practice, how they saw their jobs getting more interesting and less repetitive, all while expressing a real and broad enthusiasm about the value of the work in their organization.

As data science becomes more commonplace and simultaneously a bit demystified, we expect this trend to continue as well. After all, last year's respondents were just as excited about their work (about 79% were “satisfied” or better).

How a Data Scientist Spends Their Day

Here’s where the popular view of data scientists diverges pretty significantly from reality. Generally, we think of data scientists building algorithms, exploring data, and doing predictive analysis. That’s actually not what they spend most of their time doing, however.

As you can see from the chart above, 3 out of every 5 data scientists we surveyed actually spend the most time cleaning and organizing data. You may have heard this referred to as “data wrangling” or compared to digital janitor work. Everything from list verification to removing commas to debugging databases—that time adds up and it adds up immensely. Messy data is by far the more time-consuming aspect of the typical data scientist’s workflow. And nearly 60% said they simply spent too much time doing it.

Data scientist job satisfaction

[Chart showing job satisfaction levels]

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

What data scientists spend the most time doing

[Legend: 60%, 19%, 9%, 4%, 3%, 5%]

How do data scientists spend their time?

[Chart showing time distribution]

[CrowdFlower Data Science Report, 2016]
Data Wrangling

- Automated Transformation Suggestions
- Editable Natural Language Explanations
- Visual Transformation Previews
- Transformation History

[S. Kandel et al., 2011]
### TDE: Transform Data by Example

**Table:**

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transaction Date</strong></td>
<td><strong>output</strong></td>
</tr>
<tr>
<td>Wed, 12 Jan 2011</td>
<td>2011-01-12-Wednesday</td>
</tr>
<tr>
<td>Thu, 15 Sep 2011</td>
<td>2011-09-15-Thursday</td>
</tr>
<tr>
<td>Mon, 17 Sep 2012</td>
<td></td>
</tr>
<tr>
<td>2010-Nov-30 11:10:41</td>
<td></td>
</tr>
<tr>
<td>2011-Jan-11 02:27:21</td>
<td></td>
</tr>
<tr>
<td>2011-Jan-12</td>
<td></td>
</tr>
<tr>
<td>2010-Dec-24</td>
<td></td>
</tr>
<tr>
<td>9/22/2011</td>
<td></td>
</tr>
<tr>
<td>7/11/2012</td>
<td></td>
</tr>
<tr>
<td>2/12/2012</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

[Image of TDE: Transform Data by Example]

[Y. He et al., 2018]
Transform by Pattern: Automating Unify/Repair

- **Auto-Unify**

- **Auto-Repair**

---

<table>
<thead>
<tr>
<th>Date</th>
<th>Opponents</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 12, 1997</td>
<td>Venezuela</td>
</tr>
<tr>
<td>February 12, 1997</td>
<td>Peru</td>
</tr>
<tr>
<td>April 2, 1997</td>
<td>Colombia</td>
</tr>
<tr>
<td>1997-06-04</td>
<td>United States</td>
</tr>
<tr>
<td>1997-06-11</td>
<td>Chile</td>
</tr>
<tr>
<td>1997-06-14</td>
<td>Ecuador</td>
</tr>
</tbody>
</table>

**Table 1** shows a list of example TBP programs (we will not cover here).

**Figure 6** gives a high-level overview of the architecture.

- **Auto-Repair**

---

- **Table 1** shows a list of example TBP programs (we will not cover here).

**Figure 6** gives a high-level overview of the architecture.

- **Auto-Repair**

---

- **Table 1** shows a list of example TBP programs (we will not cover here).

**Figure 6** gives a high-level overview of the architecture.
TBP: Learning from Tables

Pair & Link Related Table-Cols

Invoke TBE to learn programs

Global optimization of TBP Graph

Table Corpus

T1

<table>
<thead>
<tr>
<th>Name</th>
<th>Born</th>
<th>Died</th>
</tr>
</thead>
<tbody>
<tr>
<td>Washington, George</td>
<td>02/22/1732</td>
<td>02/14/1799</td>
</tr>
<tr>
<td>Adams, John</td>
<td>10/30/1735</td>
<td>07/04/1826</td>
</tr>
<tr>
<td>Jefferson, Thomas</td>
<td>03/16/1751</td>
<td>06/28/1836</td>
</tr>
<tr>
<td>Madison, James</td>
<td>04/28/1758</td>
<td>07/04/1851</td>
</tr>
<tr>
<td>Monroe, James</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

T2

<table>
<thead>
<tr>
<th>Date of birth</th>
<th>President</th>
<th>Birthplace</th>
<th>State of birth</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 22, 1732</td>
<td>George Washington</td>
<td>Westmoreland County</td>
<td>Virginia</td>
</tr>
<tr>
<td>October 30, 1735</td>
<td>John Adams</td>
<td>Braintree</td>
<td>Massachusetts</td>
</tr>
</tbody>
</table>

T3

<table>
<thead>
<tr>
<th>Name</th>
<th>Term</th>
<th>State of birth</th>
<th>Born</th>
<th>Died</th>
</tr>
</thead>
<tbody>
<tr>
<td>George Washington</td>
<td>Nat-Rep</td>
<td>Va.</td>
<td>22/02/1732</td>
<td>14/12/1799</td>
</tr>
<tr>
<td>John Quincy Adams</td>
<td>Dem-Rep</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thomas Jefferson</td>
<td>Dem-Rep</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>James Madison</td>
<td>Dem-Rep</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>James Monroe</td>
<td>Dem-Rep</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

T4

<table>
<thead>
<tr>
<th>President</th>
<th>State of birth</th>
<th>Birthdate</th>
<th>Birthplace</th>
<th>Deathdate</th>
<th>Location of Death</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thomas Jefferson</td>
<td>Virginia</td>
<td>Apr 13, 1743</td>
<td></td>
<td>July 4, 1826</td>
<td></td>
</tr>
<tr>
<td>James Madison</td>
<td>Virginia</td>
<td>Mar 16, 1751</td>
<td></td>
<td>June 28, 1836</td>
<td></td>
</tr>
<tr>
<td>John Adams</td>
<td>Massachusetts</td>
<td>July 11, 1767</td>
<td></td>
<td>Feb 23, 1848</td>
<td></td>
</tr>
</tbody>
</table>

T5

<table>
<thead>
<tr>
<th>Name and (party)</th>
<th>Term</th>
<th>State of birth</th>
<th>Born</th>
<th>Died</th>
<th>Religion</th>
<th>Age at inauguration</th>
<th>Age at death</th>
</tr>
</thead>
<tbody>
<tr>
<td>Washington (F)</td>
<td>1789–1797</td>
<td>Va.</td>
<td>2/22/1732</td>
<td>12/14/1799</td>
<td>Episcopal</td>
<td>57</td>
<td>67</td>
</tr>
<tr>
<td>J. Adams (F)</td>
<td>1797–1801</td>
<td>Mass.</td>
<td>10/30/1735</td>
<td>7/4/1826</td>
<td>Unitarian</td>
<td>61</td>
<td>90</td>
</tr>
</tbody>
</table>

T6

<table>
<thead>
<tr>
<th>President</th>
<th>Birthdate</th>
<th>Birthplace</th>
<th>Deathdate</th>
<th>Location of Death</th>
</tr>
</thead>
</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>

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>16</td>
<td>3</td>
</tr>
<tr>
<td>treatmentb</td>
<td>2</td>
<td>11</td>
</tr>
</tbody>
</table>

Transpose

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

- Automate "Complex" Data Preparation steps
- Focus on frame transformations (not per-cell transformations)
- Learn from Jupyter Notebooks
- Two Types of Predictions:
  - Single-Operator Prediction
  - Next-Operator Prediction
Data Cleaning: SampleClean

A recent survey on duplicate detection has argued that the result estimation can be very challenging. For example, sample to estimate the result of aggregate queries. Similarly, crowdsourced matching is used to get humans to match each tuple pair as duplicate or non-duplicate. This problem has been extensively studied for several entities. This problem has been extensively studied for several decades. Most deduplication approaches consist of two phases: detection and sampling. What is surprising about SampleClean is that reducing the data cleaning effort makes it feasible to manually or algorithmically clean the data. Consider our example dirty dataset of publications. For deduplication, the system will propose potential matches for each publication in the sample based on a blocking technique and the user can accept or reject these matches. Finally, the clean sample with the deduplication information is loaded back into the dataset.

To query this clean sample, we need to apply a data cleaning technique, and then estimate the result. Another quantity of interest is how much the dirty data makes it feasible to manually or algorithmically clean the data. Consider our example dirty dataset of publications. For deduplication, the system will propose potential matches for each publication in the sample based on a blocking technique and the user can accept or reject these matches. Finally, the clean sample with the deduplication information is loaded back into the dataset.

The greater the number of duplicates, the higher probability that papers with a higher citation count tend to have more duplicates. Assume that the data has duplication errors and that papers are from the cleaned data. We can estimate the mean difference based on comparing the dirty and cleaned samples. We formalize these issues and propose the SampleClean framework. The result estimation is similar to SAQP, however, we retain more highly cited papers, leading to an over-estimated confidence interval. On the other hand, SampleClean is quite different from typical SAQP frameworks. NormalizedSC estimates the number of potential duplicates to manually check. Next, for deduplication, the system will propose potential matches for each publication in the sample based on a blocking technique and the user can accept or reject these matches. Finally, the clean sample with the deduplication information is loaded back into the dataset.
Data Cleaning: HoloClean

The HoloClean Framework

1. Error detection module
2. Automatic compilation to a probabilistic graphical model
3. Repair via statistical learning and inference

Proposed Cleaned Dataset

<table>
<thead>
<tr>
<th>DBAName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

Marginal Distribution of Cell Assignments

<table>
<thead>
<tr>
<th>Cell</th>
<th>Possible Values</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>t2.Zip</td>
<td>60609</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>60609</td>
<td>0.16</td>
</tr>
<tr>
<td>t4.City</td>
<td>Chicago</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>Chicago</td>
<td>0.05</td>
</tr>
<tr>
<td>t4.DBAName</td>
<td>John Veliotis Sr.</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>Johnnyo's</td>
<td>0.01</td>
</tr>
</tbody>
</table>

[T. Rekatsinas et al., 2017]
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>Game</th>
<th>Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Location</td>
</tr>
<tr>
<td>0</td>
<td>Boston</td>
</tr>
<tr>
<td>1</td>
<td>Boston</td>
</tr>
<tr>
<td>2</td>
<td>Cleveland</td>
</tr>
<tr>
<td>3</td>
<td>San Diego</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></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>wld</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.

[AH Doan et al., 2012]
Information Integration

Source A

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

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

Source A

Federated Database Systems
Amit Sheth
James Larson
1990

Source B

Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases
Scheth & Larson
1990

Schema Mapping
Data Transformation
Duplicate Detection
Data Fusion

Schema Integration

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

Source A

Schema Mapping

Data Transformation

Duplicate Detection

Data Fusion

Source B

Transformation queries or views

XQuery

XQuery

Federated Database Systems

Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases

[Amit Sheth & James Larson, 1990]

[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

Federated Database Systems

Source B

Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases

Schema Mapping
Data Transformation
Duplicate Detection
Data Fusion

Preserve lineage

[L. Dong and F. Naumann, 2009]
### Challenges in Data Fusion when Sources Copy

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stonebraker</td>
<td>MIT</td>
<td>Berkeley</td>
<td>MIT</td>
<td>MIT</td>
<td>MS</td>
</tr>
<tr>
<td>Dewitt</td>
<td>MSR</td>
<td>MSR</td>
<td>UWisc</td>
<td>UWisc</td>
<td>UWisc</td>
</tr>
<tr>
<td>Bernstein</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
</tr>
<tr>
<td>Carey</td>
<td>UCI</td>
<td>AT&amp;T</td>
<td>BEA</td>
<td>BEA</td>
<td>BEA</td>
</tr>
<tr>
<td>Halevy</td>
<td>Google</td>
<td>Google</td>
<td>UW</td>
<td>UW</td>
<td>UW</td>
</tr>
</tbody>
</table>

[X L Dong et al., 2009]
### Challenges in Data Fusion when Sources Copy

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stonebraker</td>
<td>MIT</td>
<td>Berkeley</td>
<td>MIT</td>
<td>MIT</td>
<td>MS</td>
</tr>
<tr>
<td>Dewitt</td>
<td>MSR</td>
<td>MSR</td>
<td>UWisc</td>
<td>UWisc</td>
<td>UWisc</td>
</tr>
<tr>
<td>Bernstein</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
</tr>
<tr>
<td>Carey</td>
<td>UCI</td>
<td>AT&amp;T</td>
<td>BEA</td>
<td>BEA</td>
<td>BEA</td>
</tr>
<tr>
<td>Halevy</td>
<td>Google</td>
<td>Google</td>
<td>UW</td>
<td>UW</td>
<td>UW</td>
</tr>
</tbody>
</table>

[X L Dong et al., 2009]
Challenges in Data Fusion when Sources Copy

2. With only a snapshot it is hard to decide which source is a copier.

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stonebraker</td>
<td>MIT</td>
<td>Berkeley</td>
<td>MIT</td>
<td>MIT</td>
<td>MS</td>
</tr>
<tr>
<td>Dewitt</td>
<td>MSR</td>
<td>MSR</td>
<td>UWisc</td>
<td>UWisc</td>
<td>UWisc</td>
</tr>
<tr>
<td>Bernstein</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
</tr>
<tr>
<td>Carey</td>
<td>UCI</td>
<td>AT&amp;T</td>
<td>BEA</td>
<td>BEA</td>
<td>BEA</td>
</tr>
<tr>
<td>Halevy</td>
<td>Google</td>
<td>Google</td>
<td>UW</td>
<td>UW</td>
<td>UW</td>
</tr>
</tbody>
</table>

[X L Dong et al., 2009]
Challenges in Data Fusion when Sources Copy

1. Sharing common data does not in itself imply copying.

2. With only a snapshot it is hard to decide which source is a copier.

3. A copier can also provide or verify some data by itself, so it is inappropriate to ignore all of its data.

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stonebraker</td>
<td>MIT</td>
<td>Berkeley</td>
<td>MIT</td>
<td>MIT</td>
<td>MS</td>
</tr>
<tr>
<td>Dewitt</td>
<td>MSR</td>
<td>MSR</td>
<td>UWisc</td>
<td>UWisc</td>
<td>UWisc</td>
</tr>
<tr>
<td>Bernstein</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
<td>MSR</td>
</tr>
<tr>
<td>Carey</td>
<td>UCI</td>
<td>AT&amp;T</td>
<td>BEA</td>
<td>BEA</td>
<td>BEA</td>
</tr>
<tr>
<td>Halevy</td>
<td>Google</td>
<td>Google</td>
<td>UW</td>
<td>UW</td>
<td>UW</td>
</tr>
</tbody>
</table>

[1] X L. Dong et al., 2009
Source Dependence: Iteration on Truth and Sources

Truth Discovery

Source-accuracy Computation

Dependence Detection

[X L Dong et al., 2009]
Source Dependence: Iteration on Truth and Sources

Step 1

Dependence Detection

Step 2

Truth Discovery

Step 3

Source-accuracy Computation

[X L Dong et al., 2009]
NoSQL Motivation

Scalability

User-generated data, Request load

Impedance Mismatch

ID
Customer
Line Item 1: ...
Line Item 2: ...
Payment: Credit Card, ...

Orders
Payment
Customers

Line Items

Two main motivations:
User-generated data, Request load

D. Koop, CSCI 640/490, Spring 2023

[F. Gessert et al., 2017]
## Column Stores

Each column has a file or segment on disk

<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>

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

Scalability: CAP Theorem

- **Availability**
  - Remains accessible and operational at all times.

- **Consistency**
  - Commits are atomic across the entire distributed system.

- **Partition Tolerance**
  - Only a total network failure can cause the system to respond incorrectly.

Pick Two!

- **C** (Consistency)
  - Traditional relational databases: PostgreSQL, MySQL, etc.

- **AP** (Partition Tolerance)
  - Voldemort, Riak, Cassandra, CouchDB, Dynamo-like systems

- **CA** (Availability)

[E. Brewer]
Cassandra: Replication and Consistency
Three Types of NewSQL Systems

- **New Architectures**
  - New codebase without architectural baggage of legacy systems
  - Examples: VoltDB, Spanner, Clustrix

- **Transparent Sharding Middleware:**
  - Transparent data sharding & query redirecting over cluster of single-node DBMSs
  - Examples: citusdata, ScaleArc (usually support MySQL/postgres wire)

- **Database-as-a-Service:**
  - Distributed architecture designed specifically for cloud-native deployment
  - Examples: xeround, GenieDB, FathomDB (usually based on MySQL)
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
Dataframe Data Model

- Combines parts of matrices, databases, and spreadsheets
- Ordered, but not necessarily sorted
  - Rows and columns
- No predefined schema necessary
  - Types can be induced at runtime
- Typed Row/column labels
  - Labels can become data
- Indexing by label or row/column number
  - “Named notation” or “Positional notation”
Differences between Databases & Dataframes

<table>
<thead>
<tr>
<th>Convenience</th>
<th>Flexible</th>
<th>Versatility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entire query at once</td>
<td>Strict schema</td>
<td>SFW or bust</td>
</tr>
<tr>
<td>Incremental + inspection</td>
<td>Mixed types, R/C and data/metadata equiv.</td>
<td>600+ functions</td>
</tr>
</tbody>
</table>

[D. Petersohn, 2022]
Modin as a Way to Scale Dataframes

New Data Source → Prototyping → Testing → Production

New spec → Exploring

New requirements → Laptop/Workstation → Small Cluster

Large Cluster → Feedback

[D. Petersohn]
The current landscape is a fragmented jungle! PySpark, Nvidia RAPIDS, and Dask are some of the tools in this jungle.

**Higher-level Abstractions**
- MODIN
- Koalas
- Ibis

**APIs**
- pandas
- Vaex Dataframe
- DASK Dataframe
- Cuda Dataframe
- Ray Programs
- PySpark Dataframe
- SQL + User Defined Functions
- SQL + Built-in Functions
- SQL Extensions

**Data Layer**
- NumPy Arrays
- APACHE ARROW
- Distributed
- Relational Tables

**Backends**
- Native Python
- Dask
- Nvidia RAPIDS
- Ray
- PySpark
- Azure Synapse Analytics
- Apache Spark
- PostgreSQL
- Microsoft SCOPE
- Apache MADlib
- Google BigQuery
- Microsoft SQL Server

Extending Python ecosystem

Extending SQL databases

[A. Jindal et al., 2021]
Magpie Goals

**Pythonic Environment**

- Familiar Python surface

**Unified Dataframe API**

- Ongoing standardization

**Magpie Middleware**

- Batching Pandas into large query expressions
- Backend selection using past workloads
- Cache commonly seen dataframes

**Polyengines & Mappers**

- Multi-backend environments and libraries

**Database Backends**

- Cloud backends
Time Series Data

US Treasury bill contracts

Australian electricity production

Sales of new one-family houses, USA

Annual Canadian Lynx trappings
Time Series Data

- **US Treasury bill contracts**
  - Trend

- **Australian electricity production**
  - Seasonal or cyclic?

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

- **Annual Canadian Lynx trappings**
  - Number trapped
Time Series Data

US Treasury bill contracts

Australian electricity production

Sales of new one-family houses, USA

Annual Canadian Lynx trappings

Trend

Trend + Seasonality

[Source: 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
- Cyclic

[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]
Gorilla Time Series Data Compression

Figure 2: Visualizing the entire compression algorithm. For this example, 48 bytes of values and time stamps are compressed to just under 21 bytes/167 bits.

4.1 Time series compression

In evaluating the feasibility of building an in-memory time series database, we considered several existing compression schemes to reduce the storage overhead. We identified techniques that applied solely to integer data which didn't meet our requirement of storing double precision floating point values. Other techniques operated on a complete dataset but did not support compression over a stream of data as was stored in Gorilla [7, 13]. We also identified lossy time series approximation techniques used in data mining to make the problem set more easily fit within memory [15, 11], but Gorilla is focused on keeping the full resolution representation of data.

Our work was inspired by a compression scheme for floating point data derived in scientific computation. This scheme leveraged XOR comparison with previous values to generate a delta encoding [25, 17].

Gorilla compresses data points within a time series with no additional compression used across time series. Each data point is a pair of 64 bit values representing the time stamp and value at that time. Timestamps and values are compressed separately using information about previous values. The overall compression scheme is visualized in Figure 2, showing how time stamps and values are interleaved in the compressed block.

Figure 2.a illustrates the time series data as a stream consisting of pairs of measurements (values) and time stamps. Gorilla compresses this data stream into blocks, partitioned by time. After a simple header with an aligned time stamp (starting at 2 am, in this example) and storing the first value in a less compressed format, Figure 2.b shows that time stamps are compressed using delta-of-delta compression, described in more detail in Section 4.1.1. As shown in Figure 2.b the time stamp delta of delta is 2. This is stored with a two bit header ('10'), and the value is stored in seven bits, for a total size of just 9 bits. Figure 2.c shows floating-point values are compressed using XOR compression, described in more detail in Section 4.1.2. By XORing the floating point value with the previous value, we find that there is only a single meaningful bit in the XOR. This is then encoded with a two bit header ('11'), encoding that there are eleven leading zeros, a single meaningful bit, and the actual value ('1'). This is stored in fourteen total bits.
Graph Databases focus on relationships

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

- **performance**
  - Slow loading speeds
  - Query speeds over magnitude slower than RDBMS
- **scalability**
  - Low datasize limit, typically << RAM
  - Little benefit from parallelism
- **reliability**
  - Loads never terminate
  - Query run out of memory or crash
  - Bugs
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: L. Battle, 2017]
Interactive Exploration of Spatial Data

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

L. Battle, 2017

D. Koop, CSCI 640/490, Spring 2023
Visualization: Minimize Latency

While each binned chart type in the previous section visualizes one or two aggregated dimensions, more data resolution is required for finer grained bins, as in Figure 3.

Interaction is essential to exploratory visual analysis (left), zooming in (middle), panning to the lower-left (right).

Brushing & linking, in which selections in one view highlight corresponding data in other views, requires comfortable rendering as well as querying. Given a set of data tiles and the possible brushing and linking scenarios shown in Figure 5(a), the data can be treated as a 5-dimensional cube, as illustrated in Figure 5(b).

For example, four 3-dimensional cubes can cover all 15 dimensions needed to support brushing & linking. For a pair of 1D or 2D binned plots, the maximum number of dimensions is four, effectively addressing this issue. We decompose the full cube into sub-cubes, as illustrated in Figure 4, with at most four dimensions.

A full data cube is often too big to fit in memory. We use brushing & linking to form a 5-dimensional data cube (Figure 3), which we apply binned aggregation to, deriving a heatmap of checkins in the month of January.

Brightkite data has five dimensions: User, Month, Day, Hour, and Lon/Lat. To zoom in to see fine-grained details, requiring roll-up queries and rendering projected data. Figure 4 shows four linked visualizations depicting one or two aggregated dimensions, more data resolution prohibiting large if the bin count is high. In some plots, we may wish to zoom in to see fine-grained details, requiring finer grained bins, as in Figure 3.

Applying binned aggregation to heatmap shows only checkins in the month of January. However, multiple clients might overload the server. To prevent this, we compute the filtered aggregation (or "roll-up"). Sending queries require partially de-aggregated data over which to put computing aggregates filtered by an initial data selection. These operations are performed in real-time. The size of a cube is notated by its backing data dimensions and indices.

Visualization systems such as Google Maps and Hotmap to form geographic tile boundaries highlighted in cyan. We label each checkin distribution derived from the Brightkite data set as a running example. The raw data for dynamic visualization, not pre-rendered images, is stored on disk and loaded as needed. The primary contributor to data cube size is the combination of dimensions that the data supports. For example, four 3-dimensional cubes can cover all 15 dimensions needed to support brushing & linking.

Second, they contain multidimensional data to support visual querying as well as rendering. Given a set of data tiles and the possible brushing and linking scenarios shown in Figure 5(a), the data can be treated as a 5-dimensional cube, as illustrated in Figure 5(b).

For example, four 3-dimensional cubes can cover all 15 dimensions needed to support brushing & linking. For a pair of 1D or 2D binned plots, the maximum number of dimensions is four, effectively addressing this issue. We decompose the full cube into sub-cubes, as illustrated in Figure 4, with at most four dimensions.

A full data cube is often too big to fit in memory. We use brushing & linking to form a 5-dimensional data cube (Figure 3), which we apply binned aggregation to, deriving a heatmap of checkins in the month of January. However, multiple clients might overload the server. To prevent this, we compute the filtered aggregation (or "roll-up"). Sending queries require partially de-aggregated data over which to put computing aggregates filtered by an initial data selection. These operations are performed in real-time. The size of a cube is notated by its backing data dimensions and indices.
Visualization: Task-Prioritized Prefetching

Brushes in the precomputed view serves requests from a data cube


Interacts with a new view

Query for new data cubes
Visualization: Prefetching

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

![Image of ROI tiles](image-url)
Spatial Data: Beast Architecture

**The On-top Approach**
- Spatial Modules
- User Programs
- SQL
- Spark Java/Scala APIs
- Job Monitoring and Scheduling
- RDD Runtime
- Storage (HDFS)

**From Scratch Approach**
- (Spatial) User Program + RDD APIs + Job Monitoring and Scheduling + RDD Runtime + Storage + ...

**The Built-in Approach (Beast)**
- User Programs
- Spatial Language
- SQL
- Spark Java/Scala APIs
- Spatial Operators
- Early Pruning
- RDD Runtime
- Spatial Indexing
- Storage (HDFS)

[A. Eldawy, 2021]
Spatial Data: Partitioning/Indexing & Range Query

- Read a sample
- Partition the sample using an R-tree index
- Use MBR of leaf nodes as partition boundaries for all the data

Scan matching partitions in parallel to find matching records

[A. Eldawy, 2021]
The DCC Curation Lifecycle Model

Data Curation

The DCC Curation Lifecycle Model provides a graphical high level overview of the stages required for successful curation and preservation of data from initial conceptualisation or receipt. The model can be used to plan activities within an organisation or consortium to ensure that all necessary stages are undertaken, each in the correct sequence. The model enables granular functionality to be mapped against it; to define roles and responsibilities, and build a framework of standards and technologies to implement. It can help with the process of identifying additional steps which may be required, or actions which are not required by certain situations or disciplines, and ensuring that processes and policies are adequately documented.

Data, any information in binary digital form, is at the centre of the Curation Lifecycle. This includes:

- Simple Digital Objects are discrete digital items; such as textual files, images or sound files, along with their related identifiers and metadata.
- Complex Digital Objects are discrete digital objects, made by combining a number of other digital objects, such as websites, structured collections of records or data stored in a computer system.

Full Lifecycle Actions

Sequential Actions

- Conceptualise
- Create or Receive
- Appraise and Select
- Ingest
- Preservation Action
- Preserve
- Transform
- Access, Use & Reuse
- Store
- Migration
- Migrate
- Reappraise & Select
- Reappraise
- Dispose

Occasional Actions

- Dispose
- Reappraise
- Migrate

Data (Digital Objects or Databases)

- Preservation Planning
- Description
- Representation Information
- Community Watch & Participation
- Ingest
- Transform
- Access, Use & Reuse
- Store
- Migration
- Migrate
- Reappraise

Data Curation

D. Koop, CSCI 640/490, Spring 2023
Data Curation: 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
Our research has been funded by the National Science Foundation. We have created a series of blog posts to explain the problems found when generating visualizations for scientific workflows. "Towards provenance enabling paraview pages"" and "Publishing the visualizations for (MOPx solutions" are among the topics covered. The blog provides a way to document processes and visualization expert knowledge. For more information, visit our website at http://www.stccmop.org.
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!
Reproducibility

Reproducibility Spectrum

- **Publication only**
  - Code
  - Not reproducible

- **Publication +**
  - Code and data

- **Linked and executable code and data**
  - Full replication
  - Gold standard

---

[R. D. Peng]
Machine Learning and Databases

[Image: Blocks representing different database operations such as Sorting, Hash-Map, B-Tree, Priority Queue, Join, Scheduling, Bloom Filter, Range Filter, and Caching.]

[T. Kraska, 2019]
Questions?
Final Exam

- Wednesday, May 10, **8:00**-9:50am, PM 253
- Similar format
- More comprehensive (questions from topics covered in Test 1 & 2)
- Will also have questions from graph/spatial/temporal data, provenance, reproducibility, machine learning