Just-In-Time Data Virtualization:
Lightweight Data Management with \textbf{ViDa}

Manos Karpathiotakis\textsuperscript{*}, Ioannis Alagiannis\textsuperscript{*}, Thomas Heinis\textsuperscript{**\textdagger}, Miguel Branco\textsuperscript{*}, Anastasia Ailamaki\textsuperscript{*}
Current data analysis does not scale

"Most firms estimate that they are only analyzing 12% of the data that they already have" [Forrester 2014]

- Growing data
- Growing heterogeneity
- Data movement regulations

Available data blocks business & scientific analytics
Discovering *disease signatures*

1. **DATA FEDERATION**
   - Behavior
   - Neuro-psychology
   - Brain imaging

2. **DATA INTEGRATION**
   - Genetics
   - Proteomics
   - Knowledge

3. **DATA MINING**

4. **CAUSAL MODELING**

5. **BIOLOGICAL SIGNATURE OF DISEASE**
   - Disease definition
   - Pharmacology
   - Clinical trial

× Move data
× Copy data
× Transform data
Clinical + Genetic + Imaging Data → Signature

Patients (CSV)

<table>
<thead>
<tr>
<th>id</th>
<th>Protein: AACT</th>
<th>Age</th>
<th>Phenotype</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.4</td>
<td>45</td>
<td>Trauma</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>55</td>
<td>Chronic Symptoms</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>0.2</td>
<td>56</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Brain_GrayMatter (Binary)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>...</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.45</td>
<td>0.75</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>0.33</td>
<td>0.3</td>
<td></td>
<td>0.38</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>0.12</td>
<td>0</td>
<td></td>
<td>0.47</td>
</tr>
</tbody>
</table>

Signature:
age > 50
AND
amygdala.Vol > 0.3
AND
AACT < 1

BrainRegions (JSON)

```json
[{
    "id": 1,
    "amygdala": {"X":15,"Y":20, "Vol": 0.5},
    "hippocampus": {"X":17, "Y":10, "Vol":0.2},
},
{"id": 2, ...},
{"id": 3, ...}]
```

Challenge: Physical integration & diverse queries
Diverse applications over diverse datasets

(Raw) Data:
1. “Golden” repository
2. Manipulate it freely
3. Adapt to it & to queries

Key: Data Virtualization
No Static Decisions!
ViDa Architecture

ViDa Query Language

Just-In-Time Query Executor

Just-In-Time Access Paths

Auxiliary Structures

ViDa Optimizer

Source Descriptions

CSV
XML
JSON
DBMS
Queries over heterogeneous datasets

ViDa Query Language

Source Descriptions

ViDa Optimizer

Just-In-Time Query Executor

Auxiliary Structures

Just-In-Time Access Paths
Queries translated to monoid comprehensions

Monoids:
• Abstraction for “aggregates” computation

Monoid Comprehensions*:
• Operations between monoids

```plaintext
for { 
    p <- Patients, r <- BrainRegions, 
    p.id = r.id, r.amygdala.Vol > 0.2 
} yield list p.age
```

Support multiple data models as input & output  

*Fegaras [TODS 2000]*
```
SELECT r.age 
FROM Patients p 
JOIN BrainRegions r 
ON (p.id = r.id) 
WHERE r.amygdala.Vol > 0.2
```
Query execution in ViDa

ViDa Query Language

Just-In-Time Query Executor

Just-In-Time Access Paths

Auxiliary Structures

ViDa Optimizer

Source Descriptions
Creating a query executor *just-in-time*

Adapt to data and queries just-in-time
ViDa access paths

• Access paths generated *Just-in-time*

• Adapting to schema of data

• File-format-specific opportunities

• Position caches for textual formats *

• Data caches

Reduce access costs by adapting to underlying data

*RAW [VLDB 2014]

* NoDB [SIGMOD 2012]
Just-in-time operators

- Query operators generated **Just-in-time**

- “Hard-coded”, fine-grained operators

- Adapting data layout of caches to
  - query requirements
  - data format, model

Reduce processing costs by adapting to queries
Query optimization in ViDa

ViDa Query Language

Just-In-Time Query Executor

Just-In-Time Access Paths

Auxiliary Structures

Source Descriptions

ViDa Optimizer
Optimizing a just-in-time database

- Choosing appropriate layout
- Lazy vs. Speculative Execution
- Fixing "wrong" decisions at runtime
Experimental Setup

- Intel(R) Xeon(R) CPU E5-2660 @ 2.20GHz
- 128 GB RAM
- 7500 RPM SATA

<table>
<thead>
<tr>
<th>Relation name</th>
<th>Tuples</th>
<th>Attributes</th>
<th>Size</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patients</td>
<td>41718</td>
<td>156</td>
<td>29 MB</td>
<td>CSV</td>
</tr>
<tr>
<td>Genetics</td>
<td>51858</td>
<td>17832</td>
<td>1.8 GB</td>
<td>CSV</td>
</tr>
<tr>
<td>BrainRegions</td>
<td>17000</td>
<td>20446</td>
<td>5.3 GB</td>
<td>JSON</td>
</tr>
</tbody>
</table>

```sql
SELECT val1, ..., valN
FROM Patients p
JOIN Genetics g ON (p.id = g.id)
JOIN BrainRegions b ON (g.id=b.id)
WHERE pred1 AND ... AND predN
for { p <- Patients,
    g <- Genetics,
    b <- BrainRegions,
    p.id=g.id, g.id=b.id,
    pred1, ..., predN
} yield val1,...,valN
```
ViDa vs State-of-the-art

150 analytics queries on CSV & JSON data

ViDa: Competitive without loading/transforming
ViDa enables lightweight data management

- Decouple query language used from data layout
- Adapt to datasets and queries just-in-time
- Flexible and competitive with state of the art