Fast, just-in-time queries on heterogeneous raw data

Anastasia Ailamaki
EPFL and RAW Labs SA

With Manos Karpathiotakis, Stella Giannakopoulou, Matt Olma, and the DIAS lab
most firms estimate that they are only analyzing 12% of the data that they already have

- growing data
- growing heterogeneity
- data movement restrictions

available data *impedes* business & scientific analytics

*Forrester, 2014*
The Digital Universe: 50-fold Growth from the Beginning of 2010 to the End of 2020
(The Digital Universe, EMC/IDC 2014)

store interesting data

explore data efficiently

Processingechnology grows much slower than data

(WinterCorp Survey)
When you have a hammer...

1. Write software tools to analyze data
2. Clean, extract, transform, load ALL data
3. (finally!) Run analytics

90% of the data is never used.
build database to run queries

information

process

- cache
- access
- store
- collect

⇒ bloated database software
new: one DB per app/data pair

80% of analysts’ time goes to data preparation and configuration

Main-memory DBMS

Column stores

NoSQL systems

Stream DBMS

OLTP

OLAP

Large-scale embarrassingly parallel
Databases will be extinct
The way forward

• Data model:
  – Support variety (complex structured and unstructured data)
  – Col-store/Row-store are only two of many possible layouts

• Storage model:
  – Don’t store!
  – Run in situ and cache based on actual needs/usage

• Execution model:
  – Generate engine based on query, available caches, history

Fundamentally rethink DB stack
A lean and agile engine

- Adaptive Query Processing
  - A database per query and dataset
  - SQL++ to query and clean all data
- Adaptive data access
  - Tune database dynamically
- The Human Brain Project
  - An inspiring use case
A lean and agile engine

- Adaptive Query Processing
  - A database per query and dataset
  - SQL++ to query and clean all data
- Adaptive data access
  - Tune database dynamically
- The Human Brain Project
  - An inspiring use case
Detecting active spambots

- **Flexibility**: Ad-hoc queries over diverse data formats
- **Performance**: Fast queries regardless of data format

Symantec data

Classify / Cluster

Transform & Load

Detecting active spambots

<table>
<thead>
<tr>
<th>class</th>
<th>JSON</th>
<th>SPAM</th>
<th>SPAM</th>
<th>SPAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transform &amp; Load</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Format</td>
<td>JSON</td>
<td>CSV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symantec data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Detecting active spambots

- **Flexibility**: Ad-hoc queries over diverse data formats
- **Performance**: Fast queries regardless of data format

Symantec data

Classify / Cluster

Transform & Load

Detecting active spambots

<table>
<thead>
<tr>
<th>class</th>
<th>JSON</th>
<th>SPAM</th>
<th>SPAM</th>
<th>SPAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transform &amp; Load</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Format</td>
<td>JSON</td>
<td>CSV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symantec data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
fast queries on heterogeneous data

- diverse formats
- legacy software
- privacy limitations
- data “owned” by one database

RAW: interface to raw data
With extended SQL code-generated engine

key: data virtualization
Adapting a query engine to data

Generate plug-in per data source

Query original data formats, files, and scripts

Treat each source as *native storage format*
How to build a just-in-time data base

SELECT bot, country, ...
FROM SpamEmail e, SpamCategories c
WHERE e.id == c.id AND
  e.lang = ‘English’ AND ...
How to build a just-in-time data base

SELECT bot, country, ...
FROM SpamEmail e, SpamCategories c
WHERE e.id == c.id AND
  e.lang = 'English' AND ...

Code Generate the Access Paths

Code Generate the Query

Build Position and Data Caches
Monoids:
• Abstraction for “aggregates” computation

Monoid Comprehensions*:
• Operations between monoids

Support multiple data models as input & output

Queries → Monoid comprehensions

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

*Fegaras [SIGMOD95, TODS 2000,...]
“SQL++” → Comprehensions → Algebra

\[
\text{SELECT } r.\text{age} \\
\text{FROM Patients } p \\
\text{JOIN BrainRegions } r \\
\text{ON } (p.\text{id} = r.\text{id}) \\
\text{WHERE } r.\text{amygdala.Vol} > 0.2
\]

if-else
record construction
function application
(nested) comprehension
...

for \{ 
  p \leftarrow \text{Patients}, 
  r \leftarrow \text{BrainRegions}, 
  p.\text{id} = r.\text{id}, 
  r.\text{amygdala.Vol} > 0.2 
\} 
\text{yield } \Delta \text{bag } r.\text{age}

Algebraic form amenable to relational optimizations
Data cleaning using monoid comprehensions

for(o←orders) yield list(split(o.ship_date,"/"))

dataGroup := for (o←orders)
yield cluster(o.item,kmeans)
dictGroup := for (d←dict)
yield cluster(d.item,kmeans)

for(d₁←dataGroup,
d₂←dictGroup,
d₁.center = d₂.center,
similar(metric,d₁.item,d₂.item,θ))
yield group (d₁.item)
SQL-like extensions for data cleaning

Functional Dependencies:
orderno, item → quantity

```
SELECT o.orderno, o.item, * 
FROM Orders o  
FD((o.orderno, o.item), o.quantity)
```

Data Deduplication:
```
SELECT <projections> 
FROM <dataset>  
DEDUP([<metric>], [<theta>,] <attributes>)
```
Symantec Spam Email Analysis

95GB Binary - 22GB JSON – 22GB CSV
50 queries

Flexible and fast by specializing & adapting
A lean and agile engine

- Adaptive Query Processing
  - A database per query and dataset
  - SQL++ to query and clean all data

- Adaptive data access
  - Tune database dynamically

- The Human Brain Project
  - An inspiring use case
Querying raw data w/o index: Diminishing returns

Scanning all data is slow

Indexing/tuning non-trivial for ad hoc data

60GB smart meter data, selectivity 1%, 128GB RAM, 1 thread
Invest in popular data subsets

Refine partitions over the data => Skip if useless for query

Tune indexes over popular partitions => Minimize data accesses
Adapt to data: Logical partitioning

Set the “ground” for reducing data access

Enable data skipping
Fine-grained access path selection

Capture implicit clustering
Iteratively partition dataset

Homogeneous Query-based

Increase disjointness: Reduce distinct values
Remove tails: Reduce excess kurtosis

1) Collect data statistics at runtime
2) Calculate number of sub-partitions

\[ Q_1, \ldots, Q_n \]
Adapt to queries: index tuning

Maximize gain: build cost vs performance

Index tuning on partition level

Choose what & when to build

What
- Value-Existence (i.e., Bloom filters)
- Value-Position (i.e., B+ Trees)

When
- Based on randomized algorithm
- Cost of scan vs. cost of build + gain

Build and drop based on budget
Append & in-place updates

Minimize update overhead

- Store partition state
- Calculate hash value (MD5)
- Monitor file for modifications
- Recognize updated partitions
- Fix modified partitions
  - Drop/Re-build cache/index

(attr1 - attrN)

(Bf) (B+)

Q_m

...
Slalom: Adaptive indexing over raw data

- Incremental logical partitioning
  - Based on data distribution

- Adaptive partition indexing
  - Based on access patterns

Combining Online Tuning with Adaptive Indexing

Adapt data access to queries and data at runtime
From raw data to results

59GB uniform dataset, 128GB RAM, cold caches
1000 point & range queries interchange on 2 attributes, sel: 0.5%-5%

In-situ adaptive indexing achieves interactive access
A lean and agile engine

- Adaptive Query Processing
  - A database per query and dataset
  - SQL++ to query and clean all data

- Adaptive data access
  - Tune database dynamically

- The Human Brain Project
  - An inspiring use case
human brain project

integrate clinical and simulation data

data -> knowledge

patterns

rules

unifying models

synapses

neurons

circuits

whole brain

cognition

human brain project

molecules

data

unifying models

patterns

rules

knowledge

neurons

synapses

circuits

whole brain

cognition

integrate clinical and simulation data
patient data: medical informatics

health services

epidemiology data visualization

hypothesis testing

clinical trials
biological disease signatures

the coupling of **clinical measurements** with **validated biomarkers**

**Example:** Alzheimer’s disease

<table>
<thead>
<tr>
<th>Clinical - Phenotype</th>
<th>Proteomic Biomarkers</th>
<th>Genomic Biomarkers</th>
<th>Structural Biomarkers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional capacity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General physical health</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

memory loss/tau level/APOE e4e4/hippocampus atrophy/amyloid
medical informatics platform

- visualization
- analysis
- diagnostics

unified portal

federation

 PRIVACY
local data mirror

scans, clinic, labs

hospital 1

 PRIVACY
local data mirror

scans, clinic, labs

hospital N

create a local store
image preprocessing (Lausanne)

MRI scan (DICOM) → NIFTI conversion → Segmentation → Identified tissue types → Normalization → Feature Extraction → Data store Features

script converts DICOM to NIFTI format (3d volume)

3d image of the brain

Image processing to discover the type of tissue each voxel corresponds to

data normalized to same standards, format

extract interesting features from normalized data

intermediate data also available

Identified tissue types

Normalized and mapped to brain regions
two-level anonymization

Federation and data map

- query `filter`
- response cleaner

database

- personal identifier eraser

- each hospital’s responsibility

Exclude fields
Aggregate values
Remove keywords

Anon headers
Match words
Mask images
access to private hospital data

no move, no copy
clinical+genetic+imaging data $\rightarrow$ signature

Patients (CSV)

| id | Protein: AACT | Age | Phenotype       | ...
|----|--------------|-----|-----------------|-----
| 1  | 1.4          | 45  | Trauma          | ...
| 2  | 2            | 55  | Chronic Symptoms| ...
| 3  | 0.2          | 56  |                 | ...

signature:

age $> 50$

amygdala.Vol $> 0.3$

AACT $< 1$

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>

BrainRegions (JSON)

```
[{
  "id": 1,
  "amygdala": {
    "X": 15, "Y": 20, "Vol": 0.5
  },
  "hippocampus": {
    "X": 17, "Y": 10, "Vol": 0.2
  }
},
{
  "id": 2,
},
{
  "id": 3,
}]
```
How **RAW** works

1. Ask a question
2. Generate the needed software tools
3. Discover interesting data

Data is accessed and integrated in real time
As queries run, RAW remembers information on data accessed and generated code. Its “database” is only the useful data.
deployment: hospital data mirror

Lab Results
Clinical data
Scans (MRI, CT)
Other Data

data federation
query Client

RAW query engine

extract
anonymize
convert

analysis

raw data
CSV
EXCEL

Features
Nifti
Medical Data

CSV
DICOM
raw data
What we learned

• currently data management cost grows with data owned

• impossible to pre-cook a database system suitable for all data

• from manual ingestion to automatic adaptation: rethinking DB stack with just-in-time queries and storage
Just ask.
dias.epfl.ch
raw-labs.com