Just-In-Time Data Virtualization: Lightweight Data Management with **ViDa**

Manos Karpathiotakis[*], Ioannis Alagiannis[*], Thomas Heinis[*‡], Miguel Branco[*], Anastasia Ailamaki[*]

AiAS
Data-Intensive Applications and Systems

[‡] Imperial College London

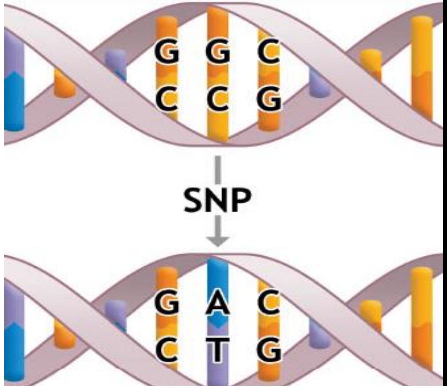
[*] EPFL

Current data analysis does not scale

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

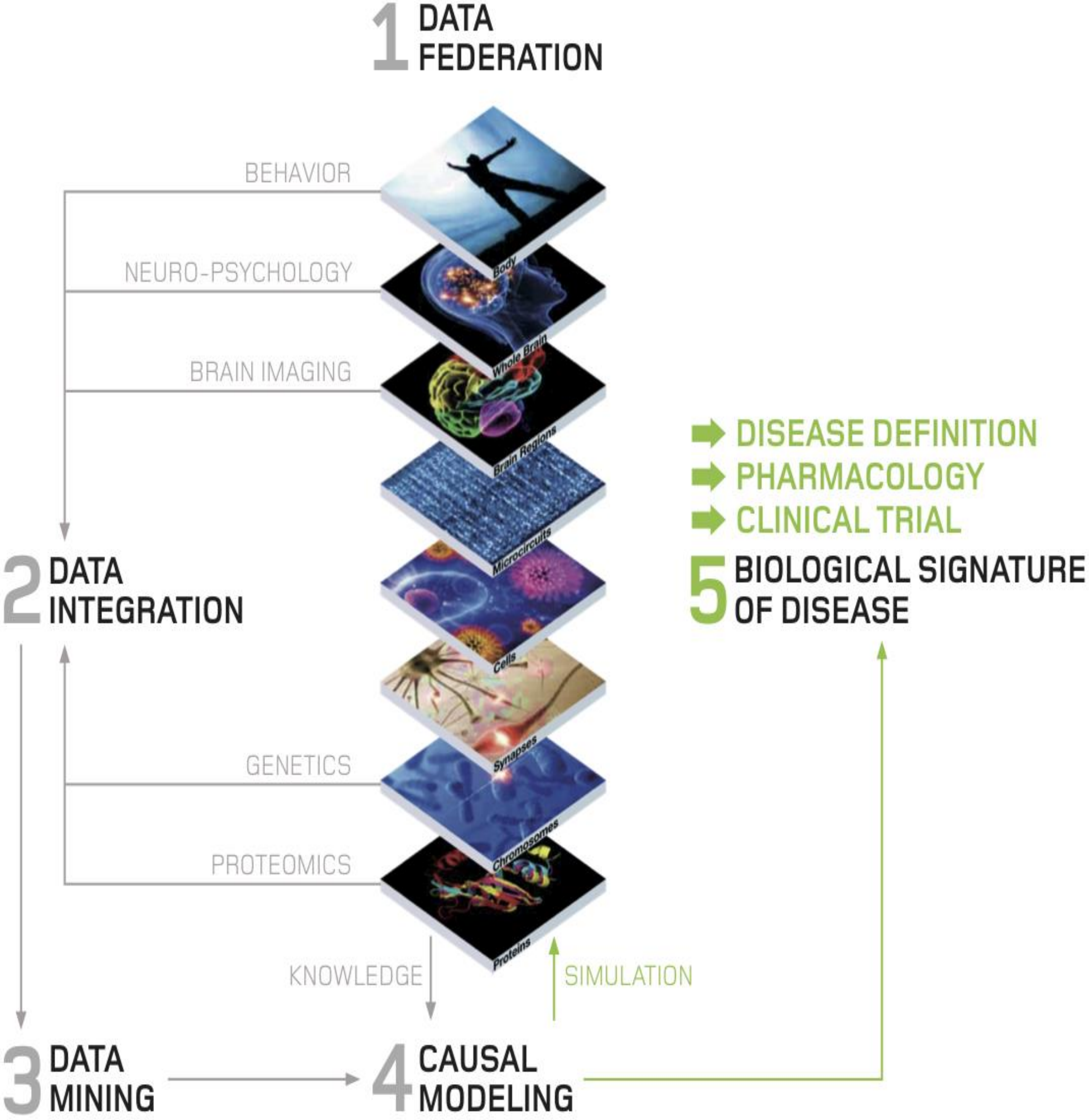
- Growing data
- Growing heterogeneity
- Data movement regulations

U.S. banks currently have 1 Exabyte of stored data



Available data blocks business & scientific analytics

Discovering *disease signatures*



1 DATA FEDERATION

BEHAVIOR

NEURO-PSYCHOLOGY

BRAIN IMAGING

GENETICS

PROTEOMICS

2 DATA INTEGRATION

3 DATA MINING

KNOWLEDGE

SIMULATION

4 CAUSAL MODELING

➡ DISEASE DEFINITION

➡ PHARMACOLOGY

➡ CLINICAL TRIAL

5 BIOLOGICAL SIGNATURE OF DISEASE

✗ Move data

✗ Copy data

✗ Transform data

Clinical+Genetic+Imaging Data → Signature

Patients (CSV)

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

Brain_GrayMatter (Binary)

	0	1	…	n
0	0.45	0.75	…	0.1
1	0.33	0.3	…	0.38
…	…	…	…	…
m	0.12	0	…	0.47

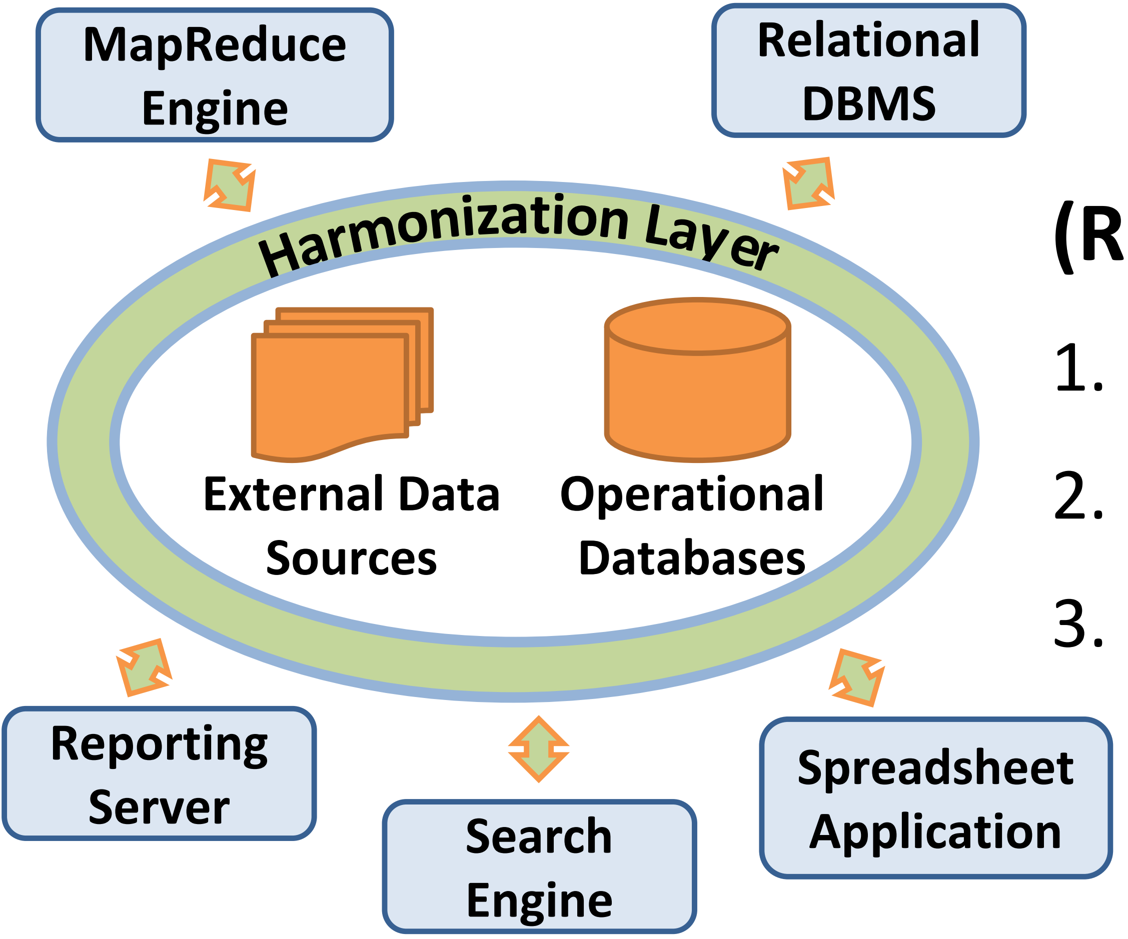
Signature:
age > 50
AND
amygdala.Vol > 0.3
AND
AACT < 1

BrainRegions (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



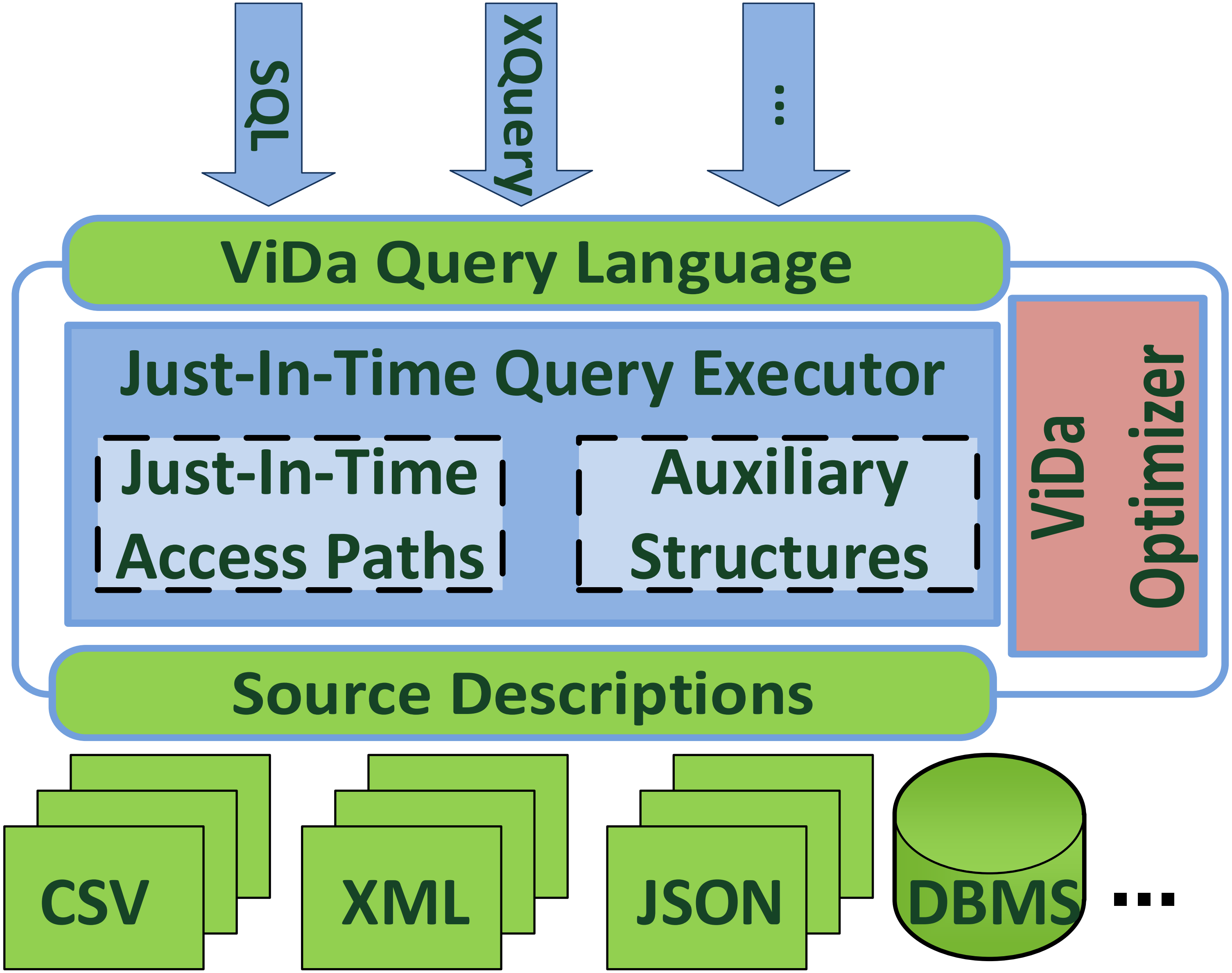
(Raw) Data:

1. “Golden” repository
2. Manipulate it freely
3. Adapt to it & to queries

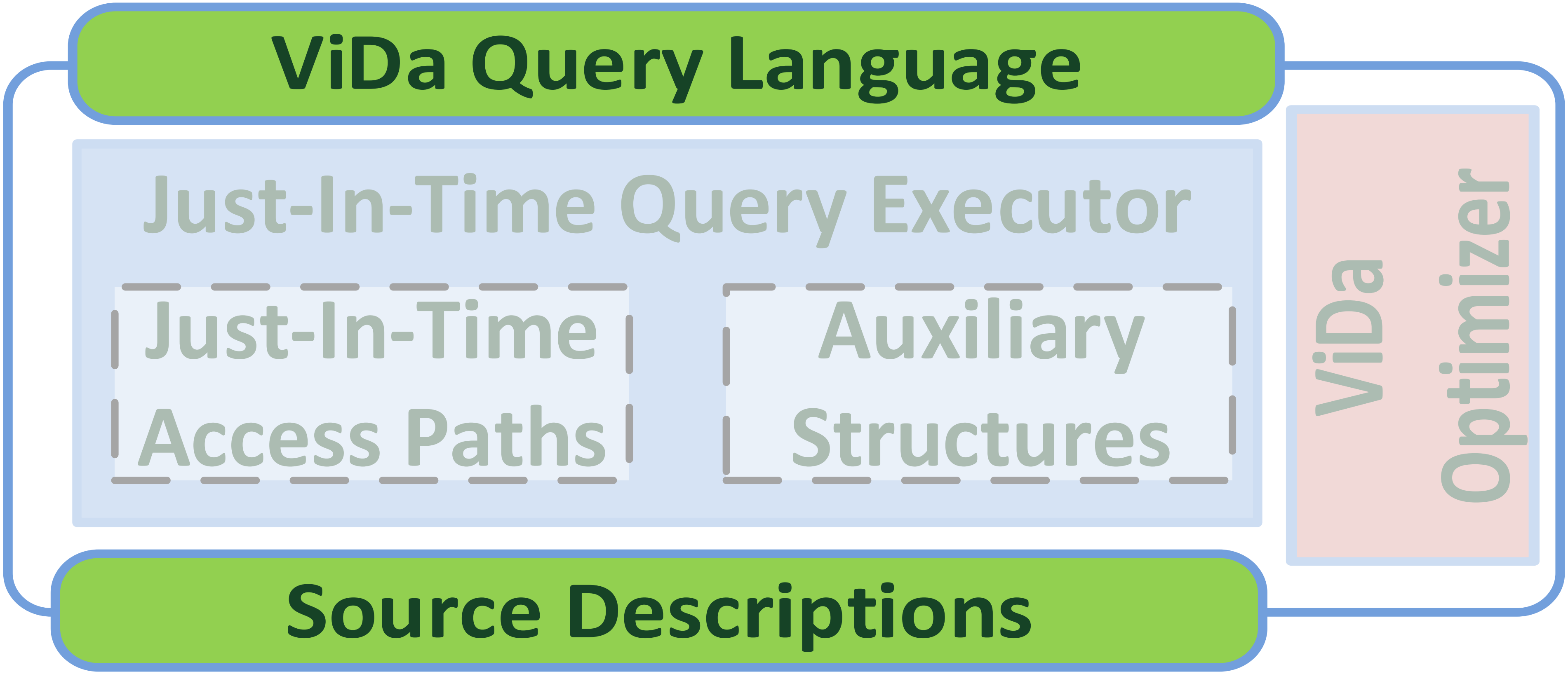
Key: Data Virtualization

No Static Decisions!

ViDa Architecture



Queries over heterogeneous datasets



Queries translated to monoid comprehensions

Monoids:

- Abstraction for “aggregates” computation

Monoid Comprehensions*:

*Fegaras [TODS 2000]

- Operations between monoids

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

Sum/Bag/Set/Top-K/…

Support multiple data models as input & output

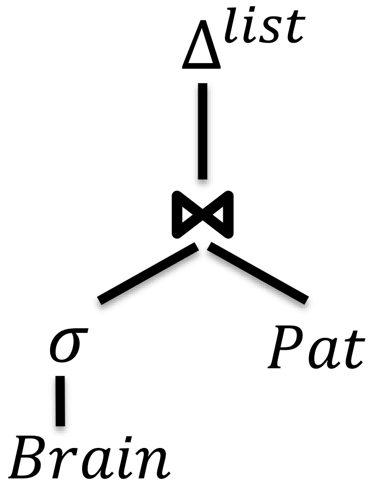
"SQL++" → Comprehensions → Algebra

```
SELECT r.age
FROM Patients p
JOIN BrainRegions r
ON (p.id = r.id)
WHERE r.amygdala.Vol > 0.2
```

Internal Calculus

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

Optimizable Algebra



if-else

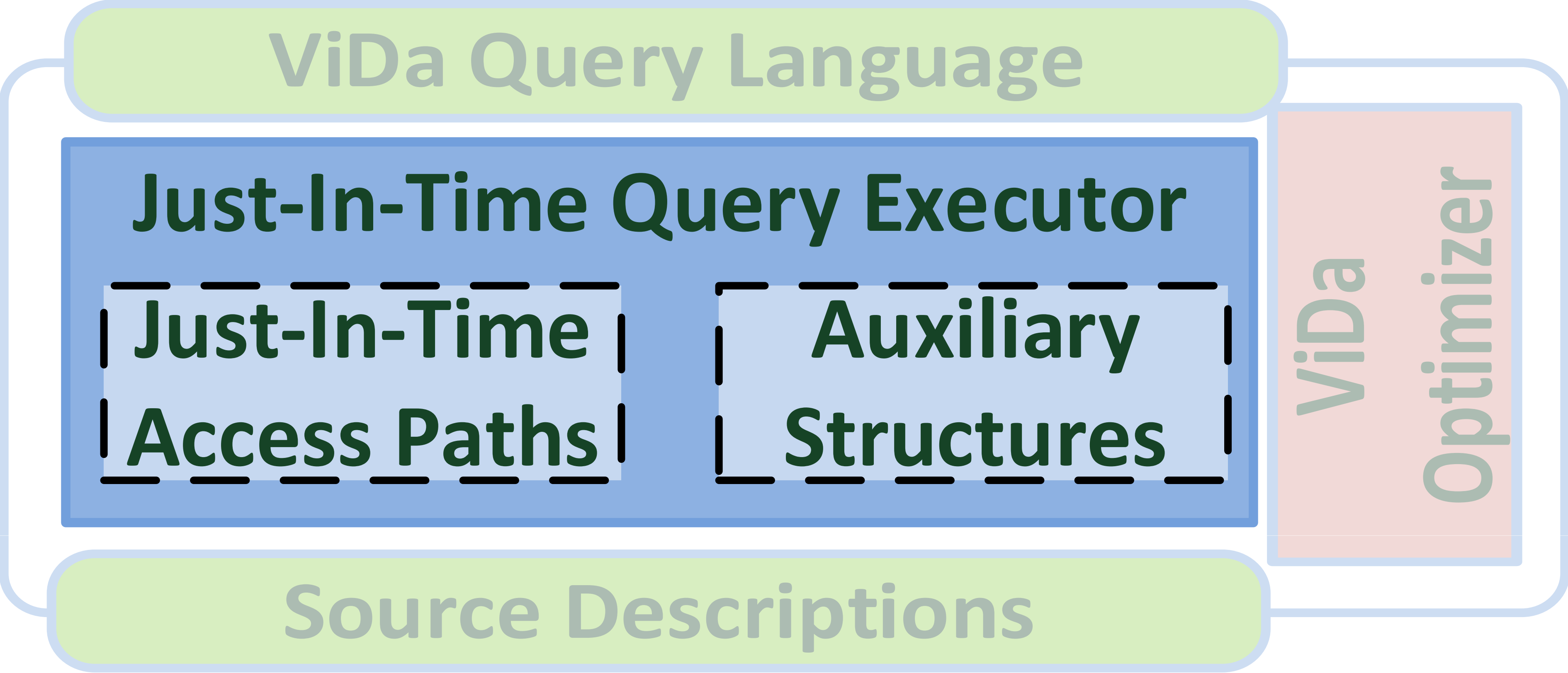
record construction

function application

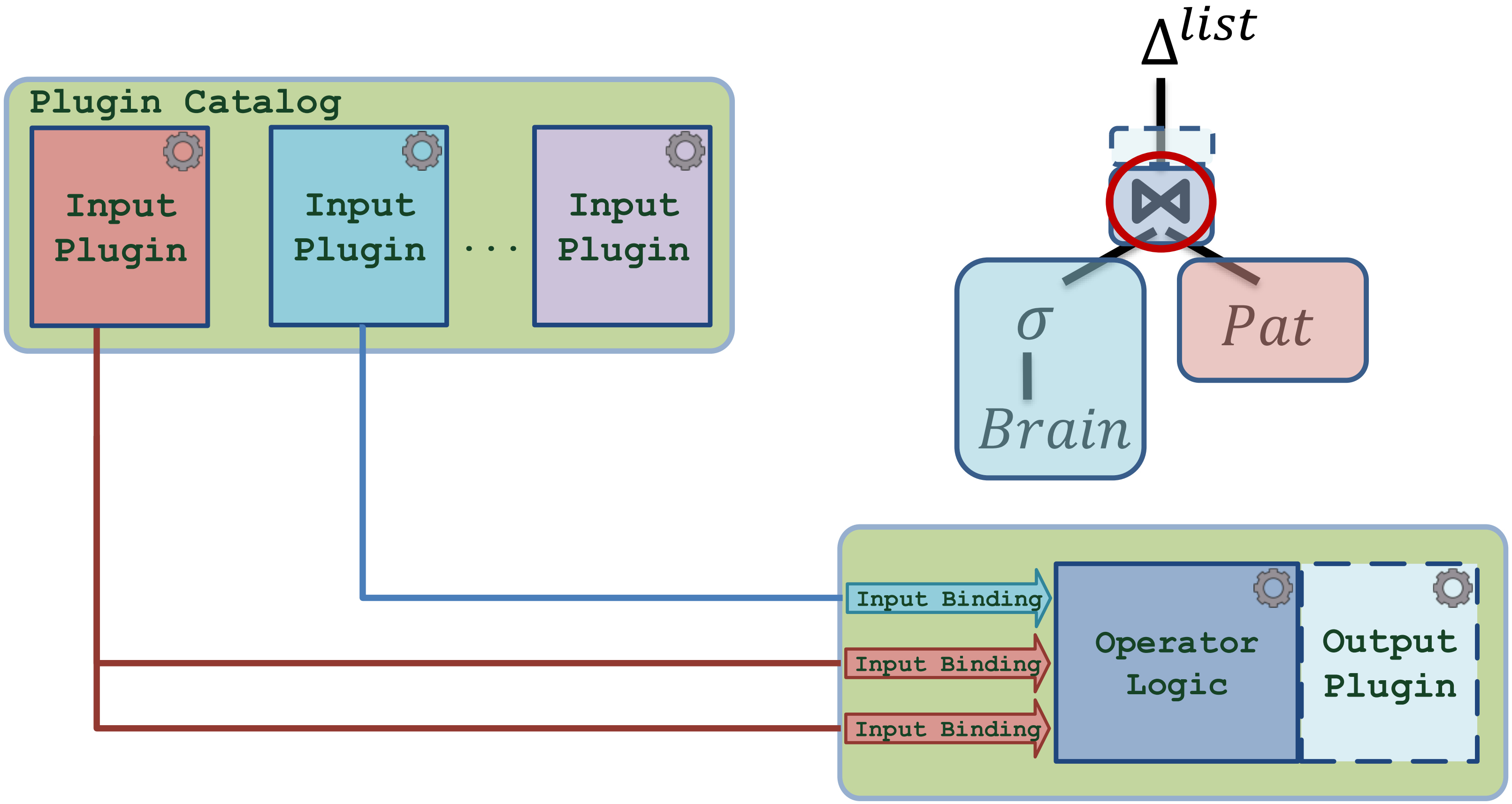
(nested) comprehension

…

Query execution in **ViDa**



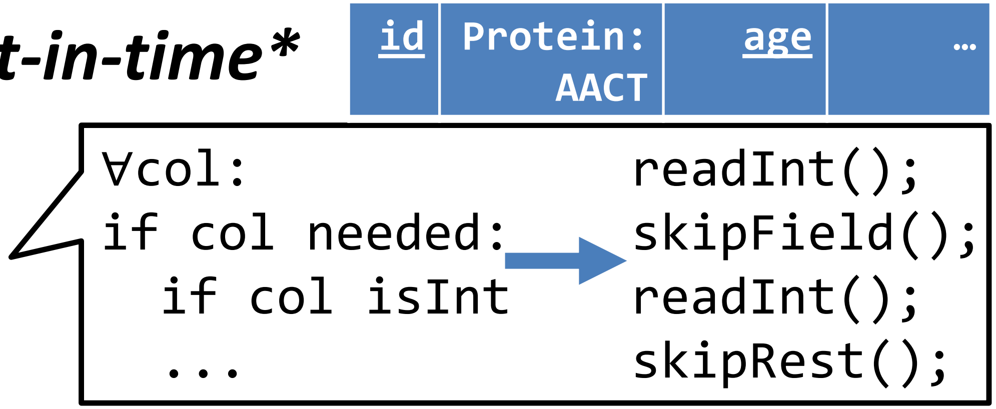
Creating a query executor *just-in-time*



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 ǂ
- Data caches



<u>id</u>	Protein: AACT	<u>age</u>	…

```
∀col:
if col needed:
   if col isInt
   ...
```

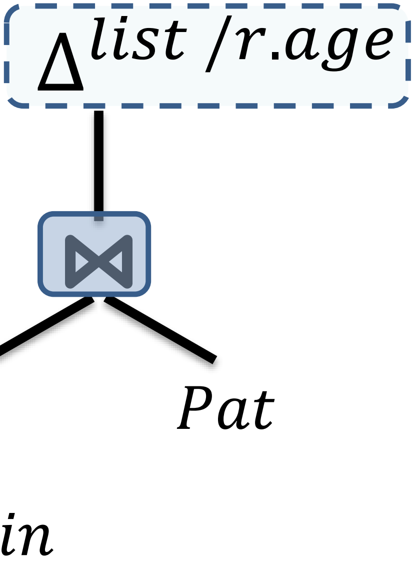
```
readInt();
skipField();
readInt();
skipRest();
```

*RAW [VLDB 2014]

ǂ NoDB [SIGMOD 2012]

Reduce access costs by adapting to underlying data

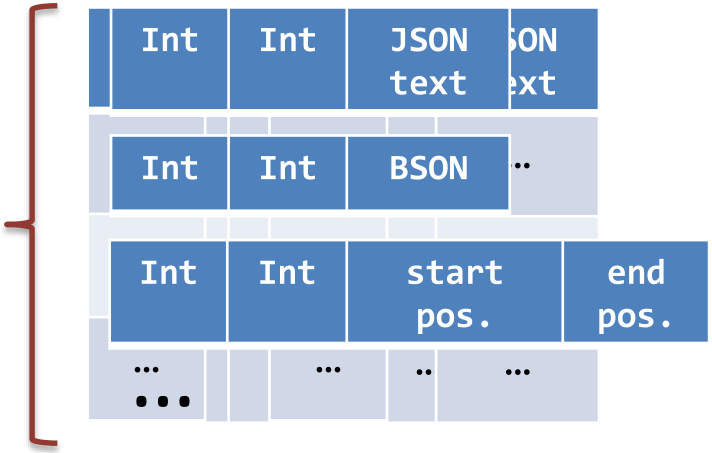
Just-in-time operators



- Query operators generated ***Just-in-time***
- "Hard-coded", fine-grained operators

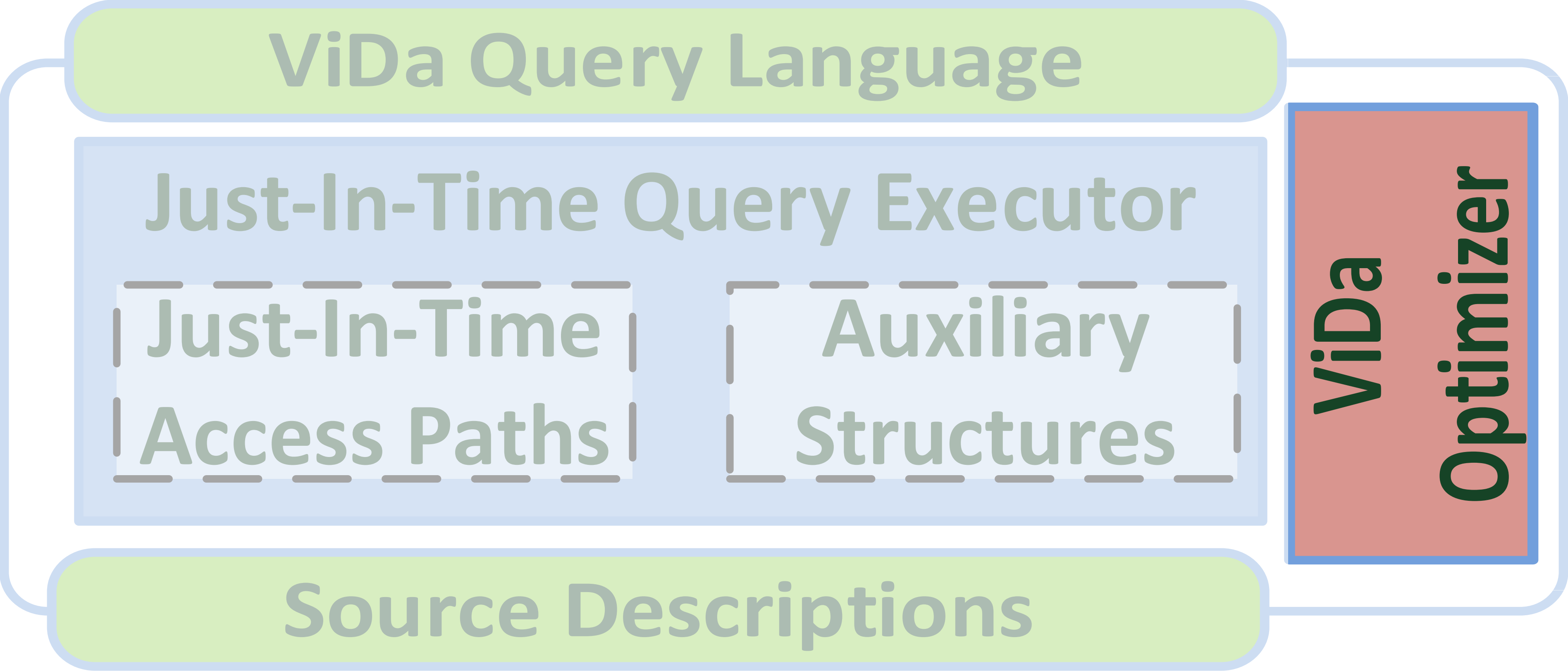
$$\Delta_{p}^{\oplus/e}(X)$$ → **`outputBindings(format);`**

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



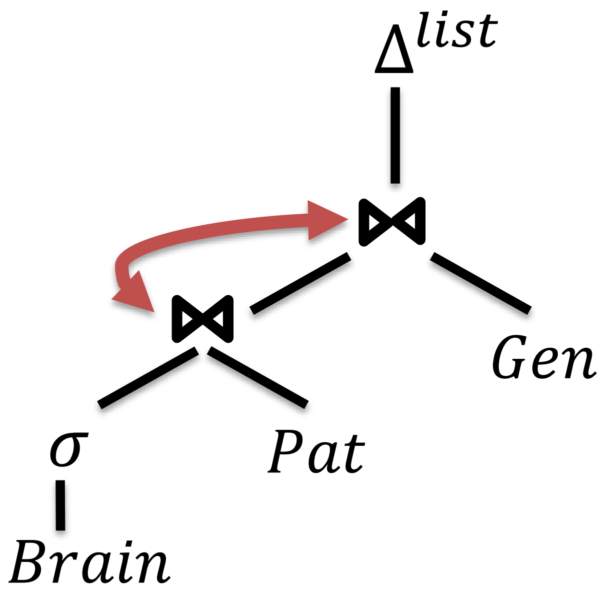
Reduce processing costs by adapting to queries

Query optimization in **ViDa**



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

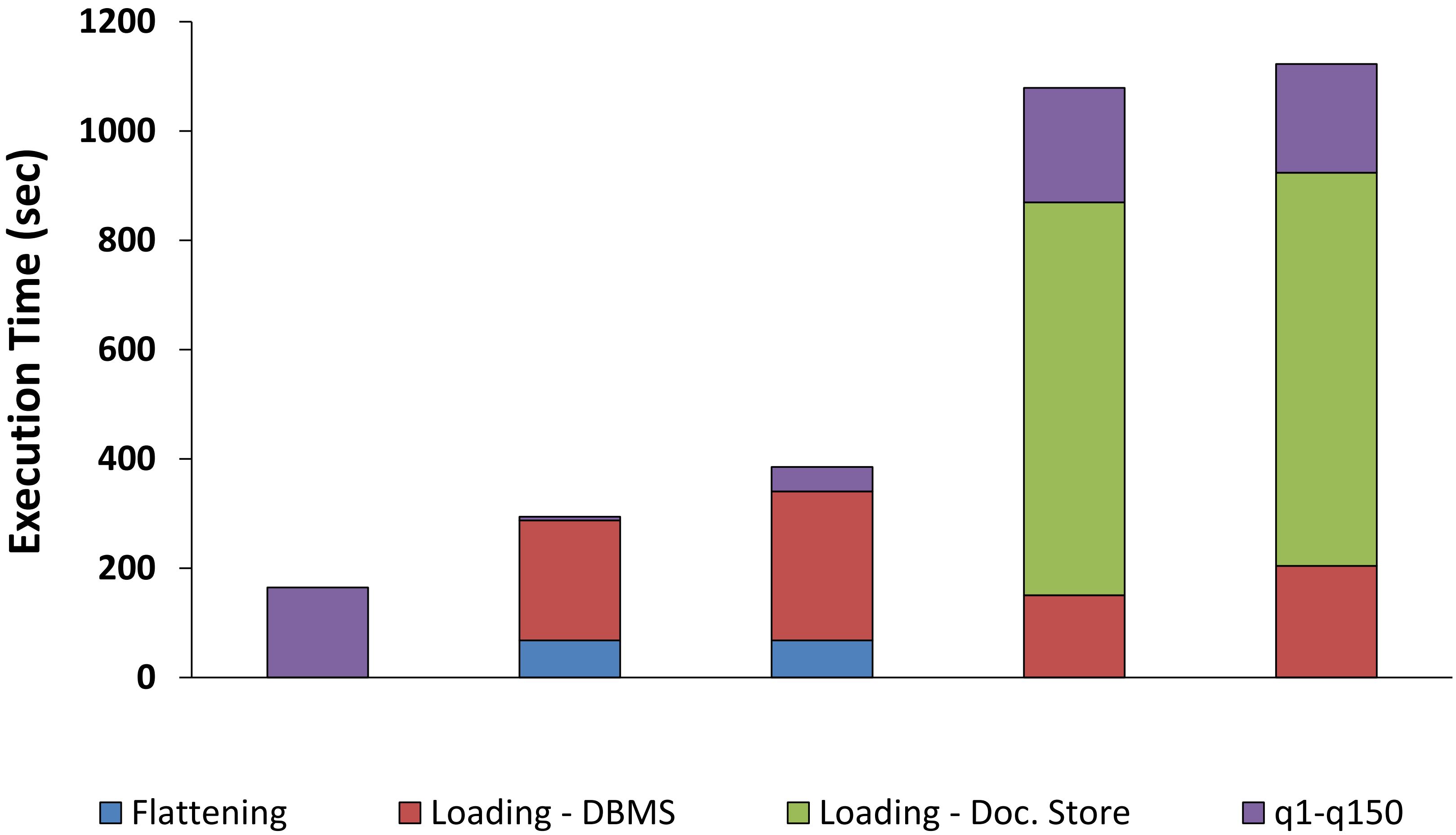
Relation name	Tuples	Attributes	Size	Type
Patients	41718	156	29 MB	CSV
Genetics	51858	17832	1.8 GB	CSV
BrainRegions	17000	20446	5.3 GB	JSON

```
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