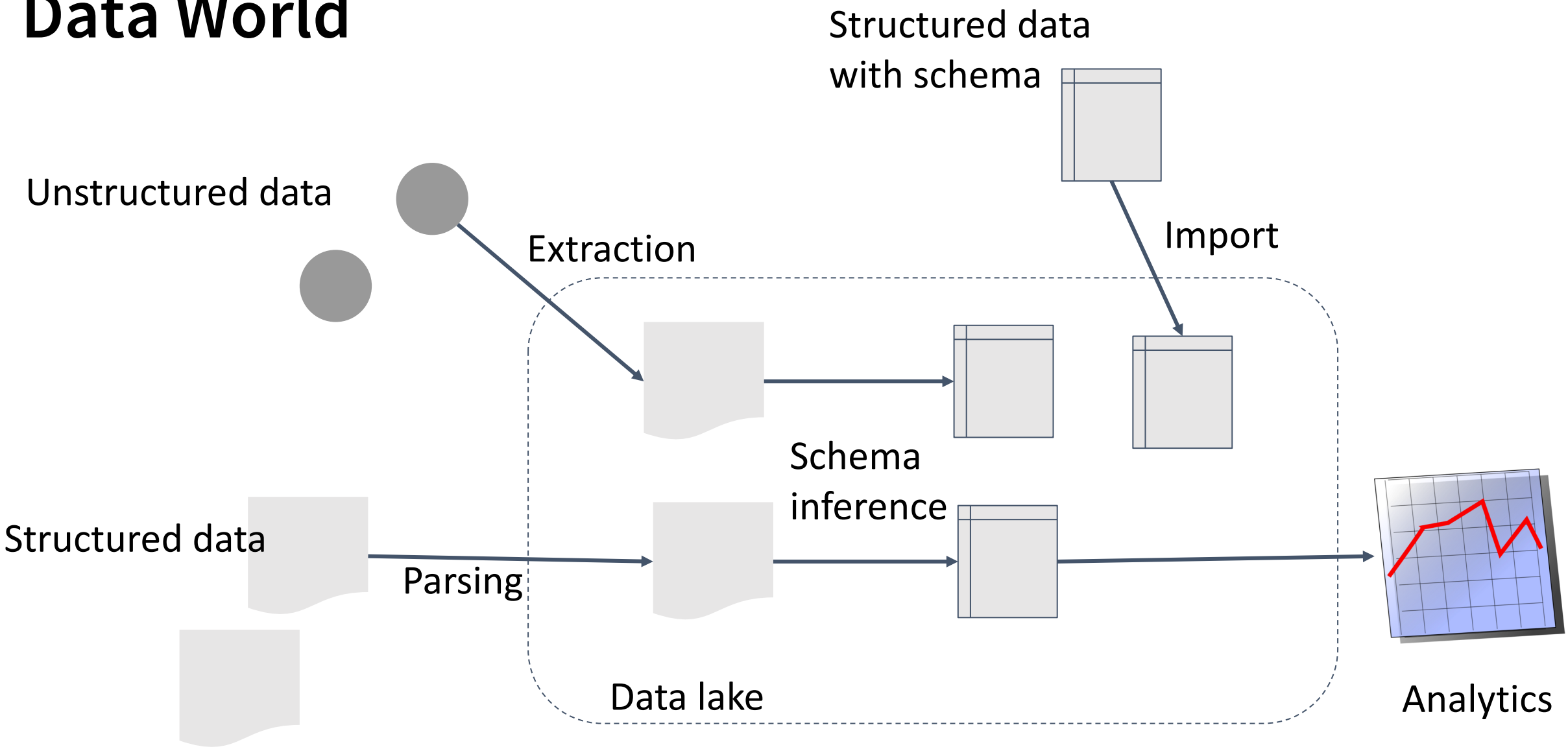
Data Lake Management: Challenges and Opportunities

Fatemeh Nargesian[1], Erkang Zhu[2], Renée J. Miller[3], Ken Q. Pu[4], Patricia C. Arocena[5]

1. University of Rochester, 2. University of Toronto, 3. Northeastern University
4. Ontario Tech University, 5. TD Bank Group

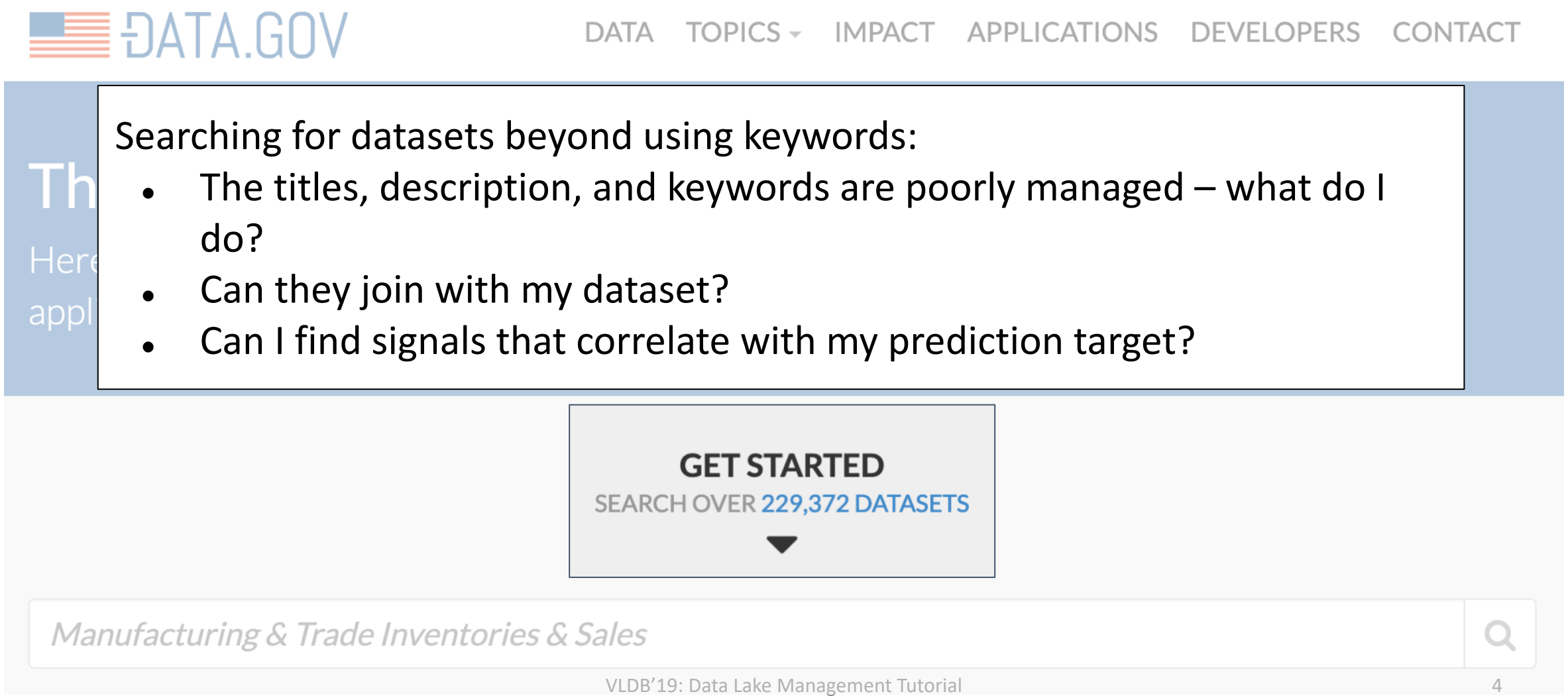
Data World


Structured data
with schema
Unstructured data
Extraction
Import
Schema
inference
Structured data
Parsing
Data lake
Analytics

Data Lakes ≥ Lots of Data

- **Scenario 1**: A global investment bank (>10k employees). More than 100k datasets.
  - During 2008 financial crisis, regulator asked for risk exposure to Lehman Brothers
  - Worked day-and-night scrambled to find relevant datasets scattered across the company to compile the report
- **Data Discovery** is the task of finding relevant datasets for analysis.

Data Lakes ≥ Lots of Data



Data Lakes ≥ Lots of Data

- **Scenario 2**: The same global investment bank, started to build a data lake after the crisis. Further issues were encountered:
  - Metadata of datasets (e.g., description, constraints, annotations, access protocols etc.) is locked in silos without standardization
  - Tried to integrate with Open Data, but many lack any useful metadata and those who do are too chaotic to digest directly
- **Metadata Management** is a common task in data lakes

Data Lakes ≥ Lots of Data

What are “VECTOR” and “COORDINATE”?

Homicides	UOM	UOM_ID	SCALAR_FACTOR	SCALAR_ID	VECTOR	COORDINATE	VALUE	STATUS	SYMBOL	TERMINATED	DECIMALS
Number of homicide victims	Number	223	units	0	v1489206	1.6.1	283				0
Percentage of homicides	Percentage	242	units	0	v1489207	1.6.2	48.05				2
Number of homicide victims	Number	223	units	0	v1489196	1.1.1	76				0
Pe						1.1.2	28.36				2
Nu						1.2.1	180				0
Pe						1.2.2	67.16				2
Nu						1.3.1		..			0
Pe						1.3.2		..			2
Nu						1.4.1	12				0
Pe						1.4.2	4.48				2

Number and percentage of homicide victims, by type of firearm used to commit the homicide

Number and percentage of homicide victims, by type of firearm used to commit the homicide (total firearms; handgun; rifle or shotgun; fully automatic firearm; sawed-off rifle or shotgun; firearm-like weapons; other firearms, type unknown), Canada, 1974 to 2018.

Publisher - Current Organization Name: Statistics Canada

Licence: Open Government Licence - Canada

Not helpful!

https://open.canada.ca/data/en/dataset/be073ee2-a302-4d32-af20-a48f5fbe2e63

Data Lakes can be Dirty

tpep_dropoff_dat...	passenger_count	trip_...	Rate...	store...	PULo...	DOLo...	pay...	fare_amount
2017 Jan 09 11:25:45 AM	1	3.3	1	N	263	161	1	12.5
2017 Jan 09 11:36:01 AM	1	0.9	1	N	186	234	1	5
2017 Jan 09 11:42:05 AM	1	1.1	1	N	164	161	1	5.5
2017 Jan 09 11:57:36 AM	1	1.			36	75	1	6
2017 Jun 20 10:39:16 PM	1	0.			41	141	2	630,461.82
2017 Jan 19 09:29:44 AM	3				39	264	2	625,900.8
2017 Jan 01 02:57:09 AM	1	0	1	N	232	243	3	538,579.2
2017 Apr 01 07:45:43 P...	1	0	1	N	90	264	2	538,481.03
2017 Oct 10 04:33:07 P...	1	1,178.6	1	N	170	170	2	404,093.97

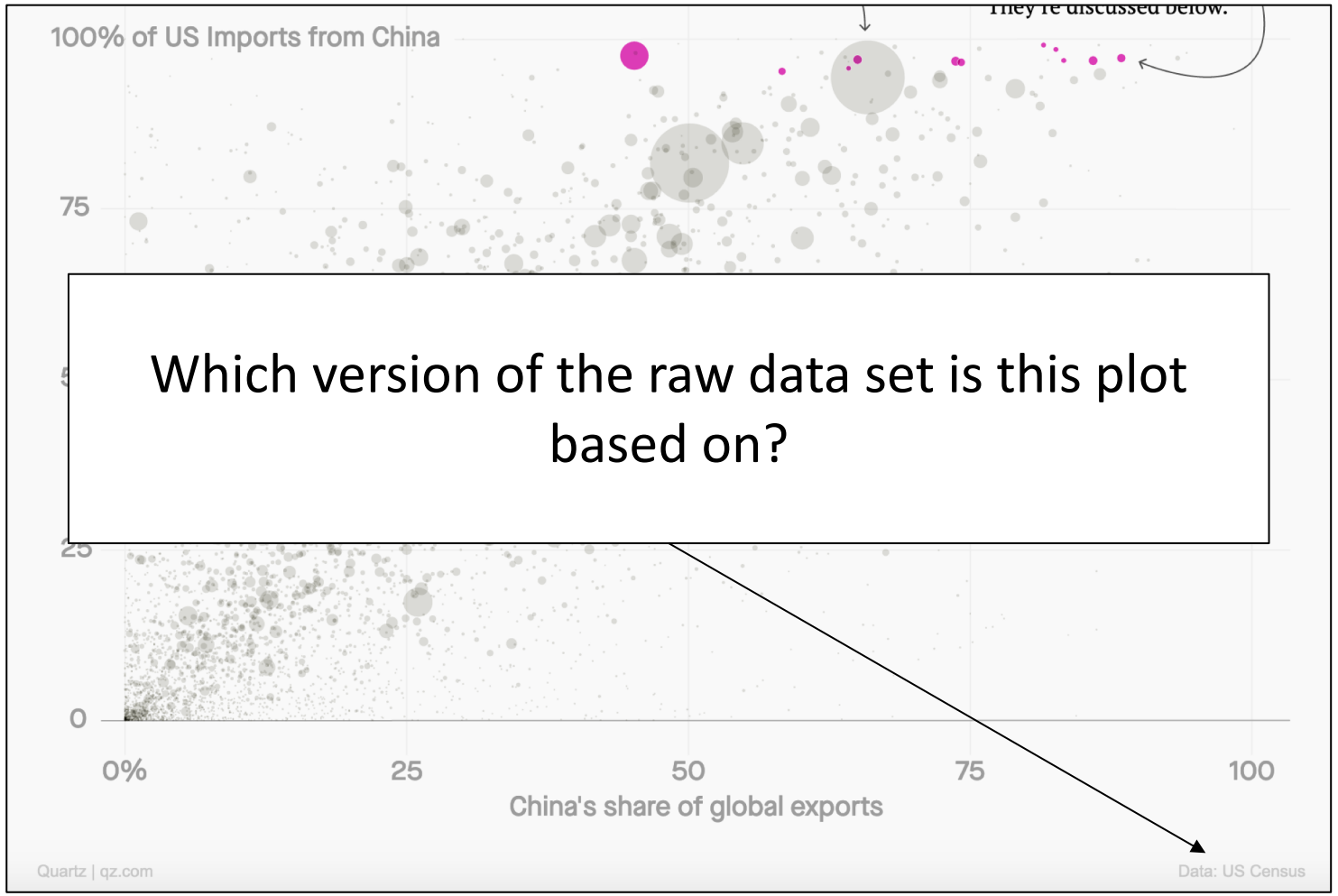
Are these values correct?

https://data.cityofnewyork.us/Transportation/**2017-Yellow-Taxi-Trip-Data**/biws-g3hs

Data Lakes can be Dirty

- **Data Cleaning** is the process of fixing errors and missing values in order to produce reliable analysis.
- **Survey Result 1:** data cleaning is the No. 1 most cited task in data lake, and >85% considered it either major or critical to the business.
  - 80-20 rule of data science – data cleaning is time-consuming and not fun!
  - The challenge for data cleaning in data lake is little schema or type information is available for data users to validate.
  - “Unit-test” for datasets – adopting strategies from software engineering

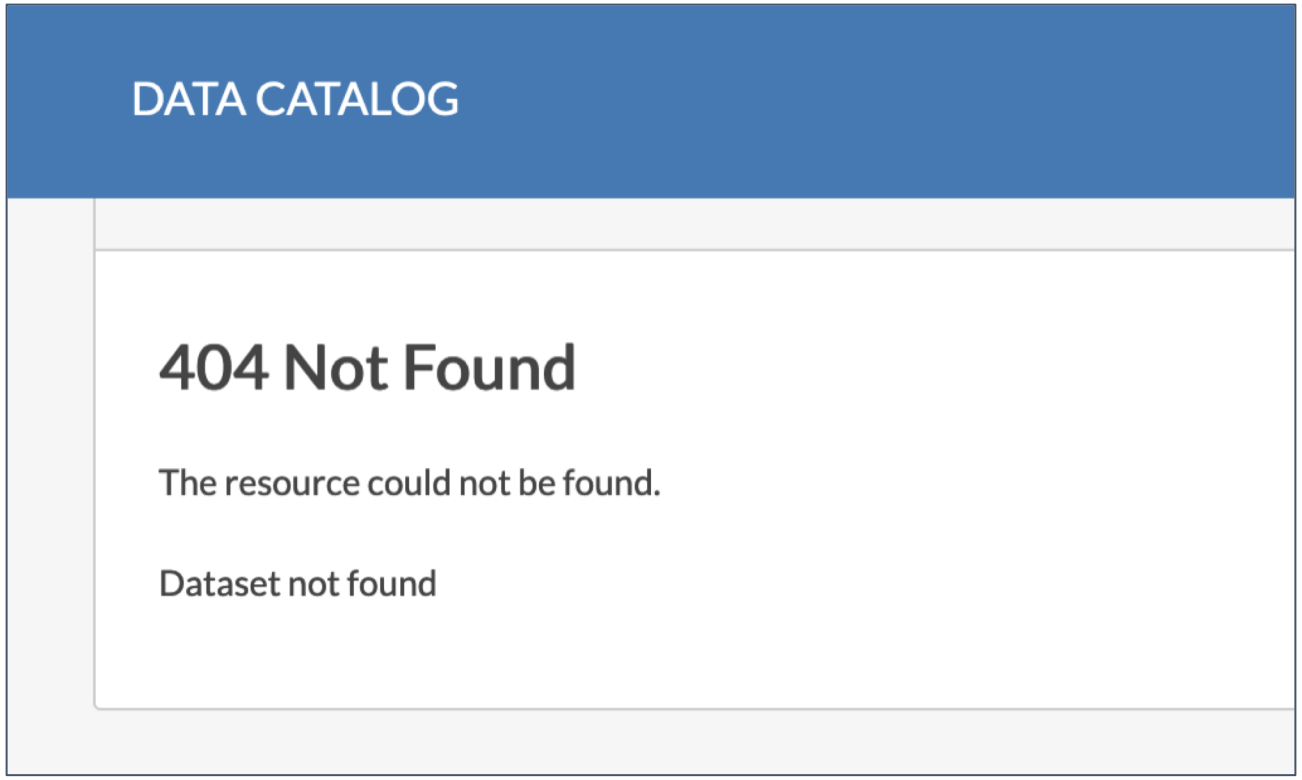
Data Lakes are Evolving



http://tse.export.gov/stateimports/TSIReports.aspx

https://qz.com/1654798/these-are-the-products-the-us-is-most-reliant-on-china-for

Data Lakes are Evolving



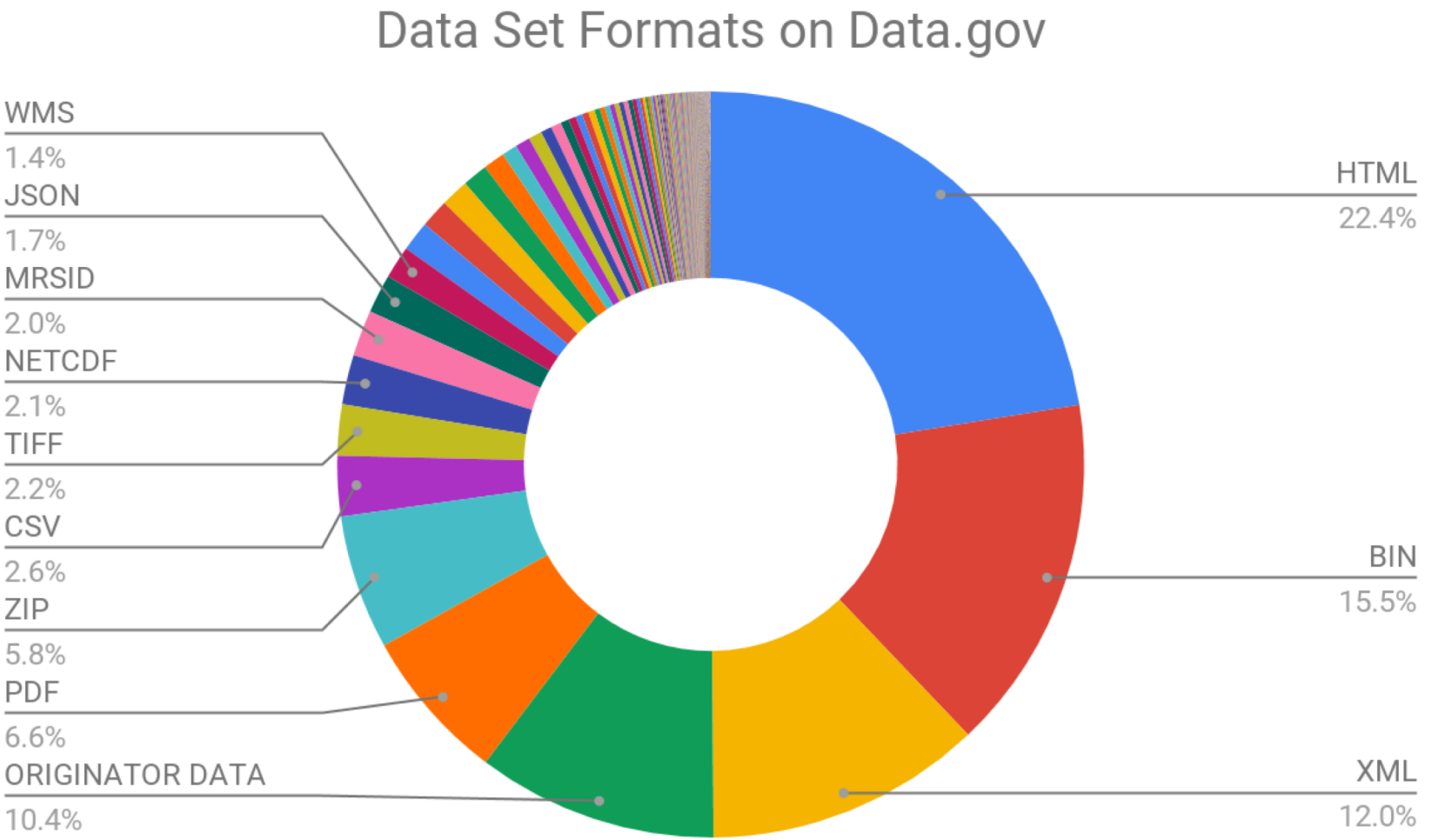
https://catalog.data.gov/dataset/8c0bc869-1189-4ced-af26-2abd2337b886

Data Lakes are Evolving

- **Scenario 3:** a data science research institution (~100 employees). 1000-10k datasets.
  - Datasets are stored in HDFS directories
  - Many duplicates as datasets are often being copied for new project
  - Datasets are constantly being updated, having their schema altered, being derived into new ones, and disappearing/reappearing
- **Dataset Versioning** is to maintain all versions of datasets for storage cost-saving, collaboration, auditing, and experimental reproducibility.

Drawing: Marina Roussou

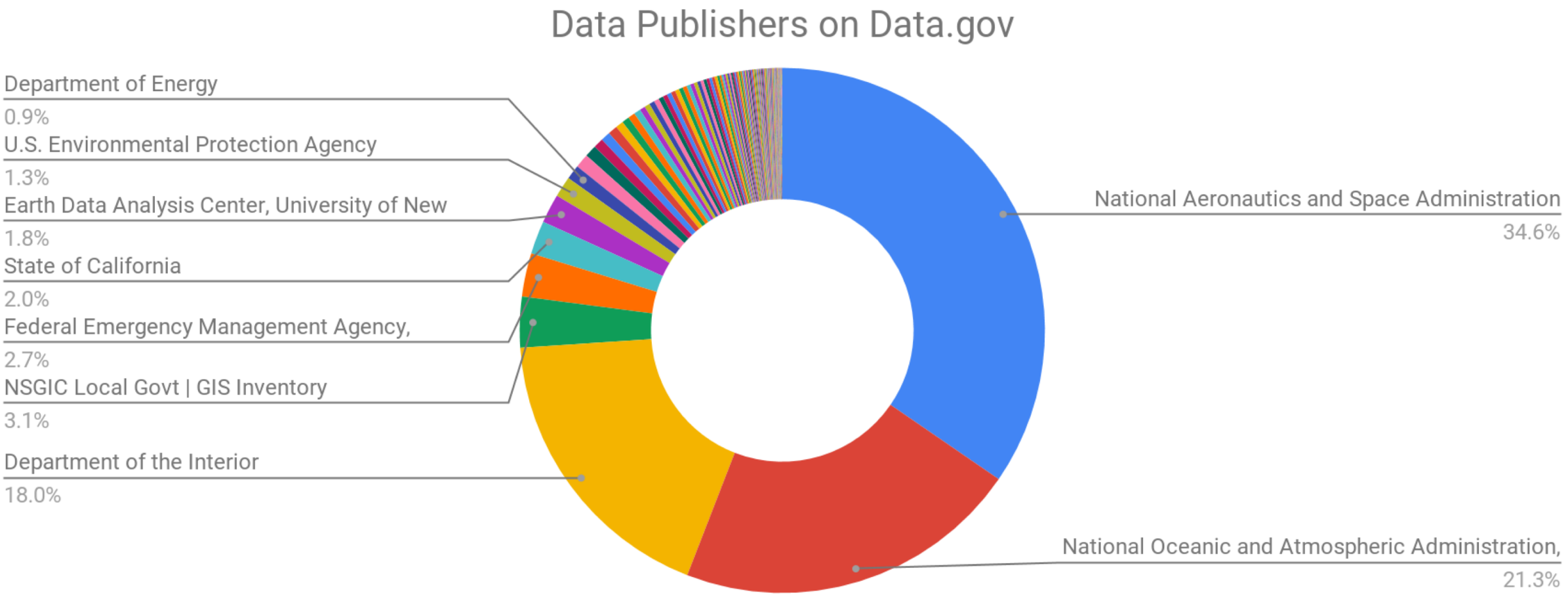
Data Lakes are Diverse



Data Lakes are Diverse

- Dataset formats in the open world can be highly heterogeneous:
- **Ingestion & Extraction** is the task of bringing structured datasets into data lake:
  - Ingest already-structured datasets
  - Extract structured data from unstructured and semi-structured data sources
- Parsing and information extraction techniques are used here.

Data Lakes are Diverse

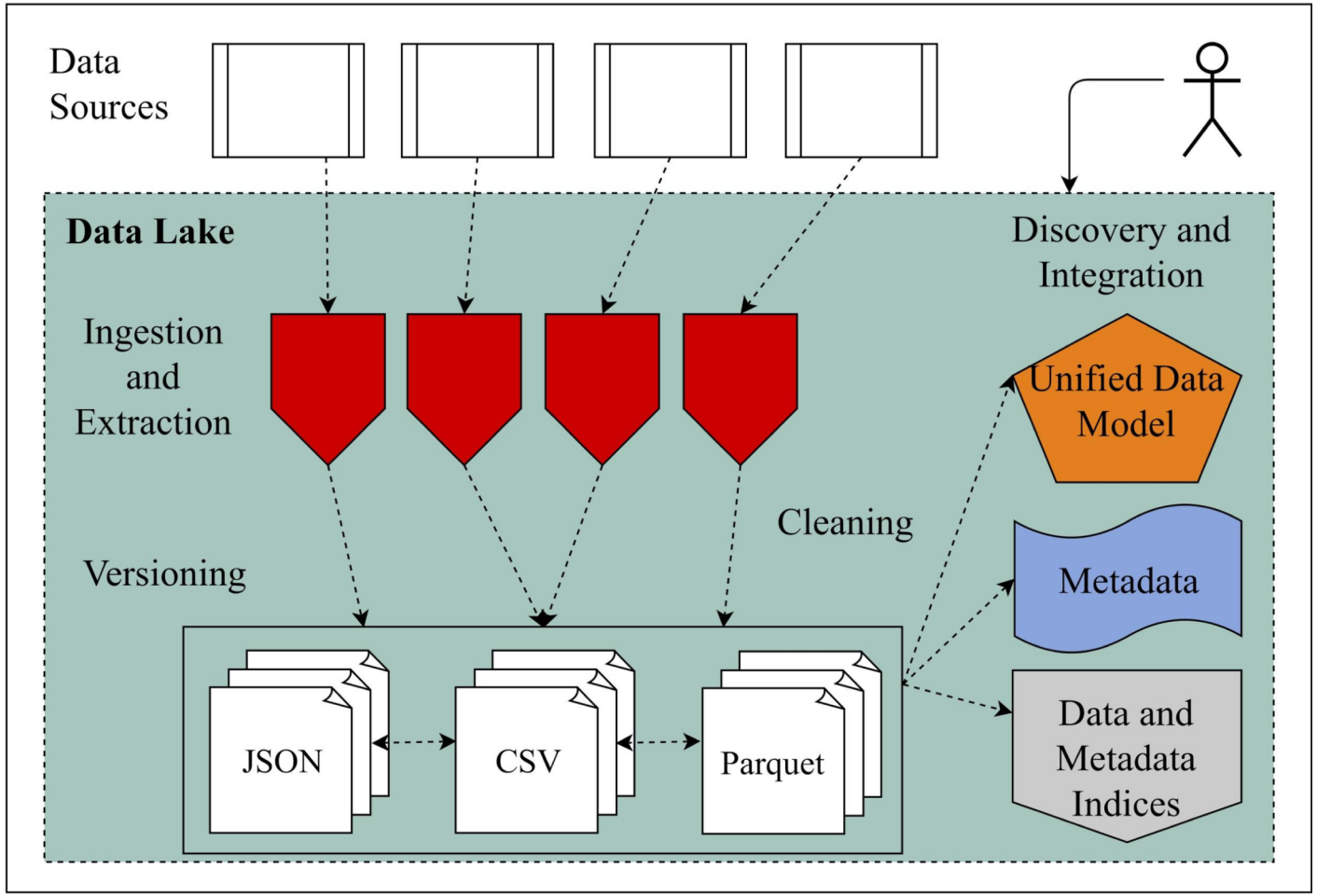


Data Lakes are Diverse

- **Scenario 4:** A large hospital (>10k employees). More than 1000 datasets.
  - Enrich Electronic Health Records (EPR) using data from various non-standard personal health record datasets for better predicting health risks
  - Joining or “unioning” tables from different datasets and sources
- **Data Integration** is the task of finding joinable or unionable tables or of on-demand population a schema with all data from the lake that conforms to the schema

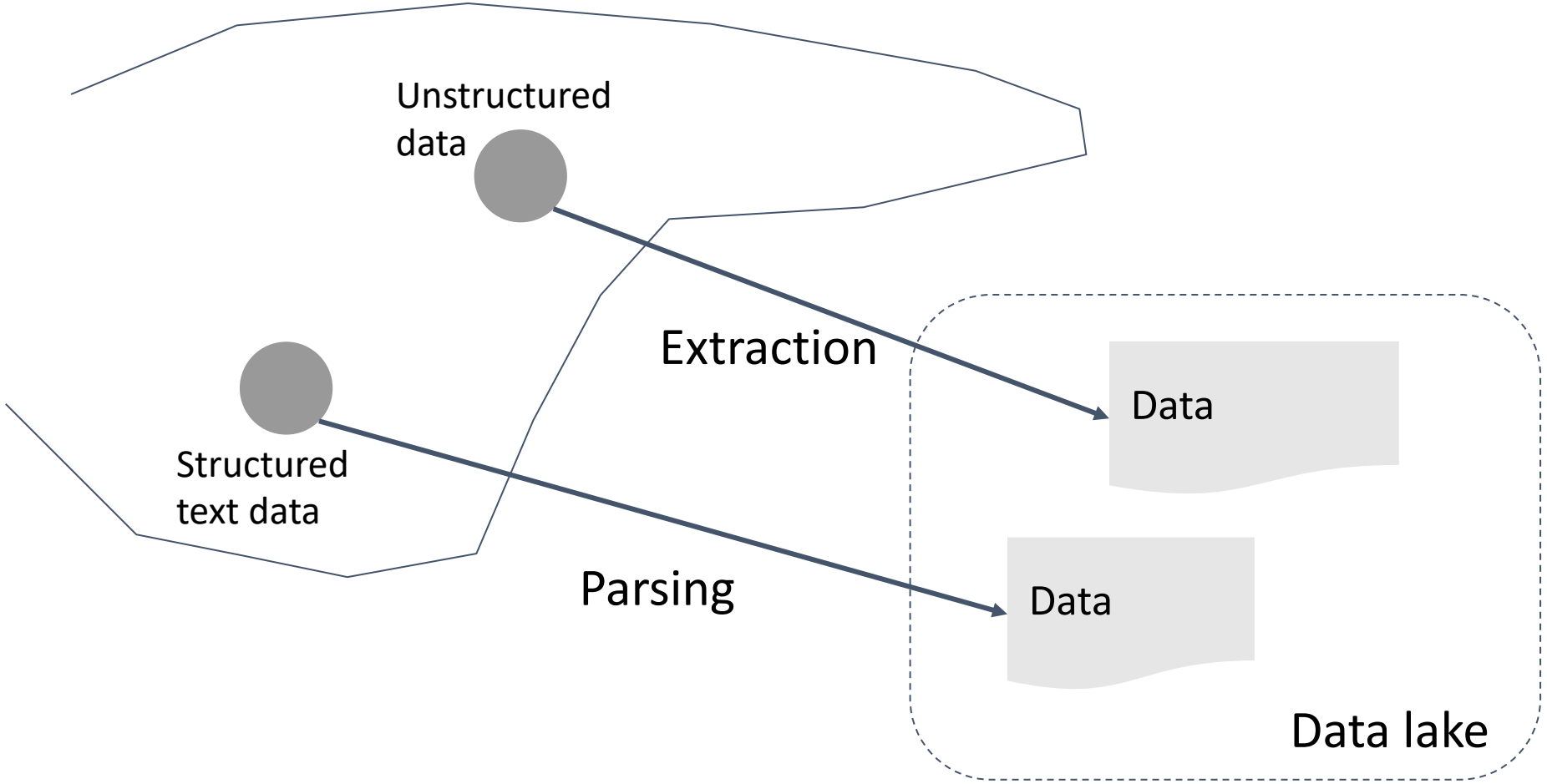
Common Tasks in Data Lakes

1. Ingestion
2. Extraction (Type Inference)
3. Metadata Management
4. Cleaning
5. Integration
6. Discovery
7. Versioning

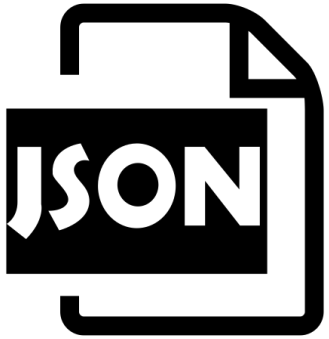


Ingestion

Ingestion


Unstructured
data
Extraction
Data
Structured
text data
Parsing
Data
Data lake

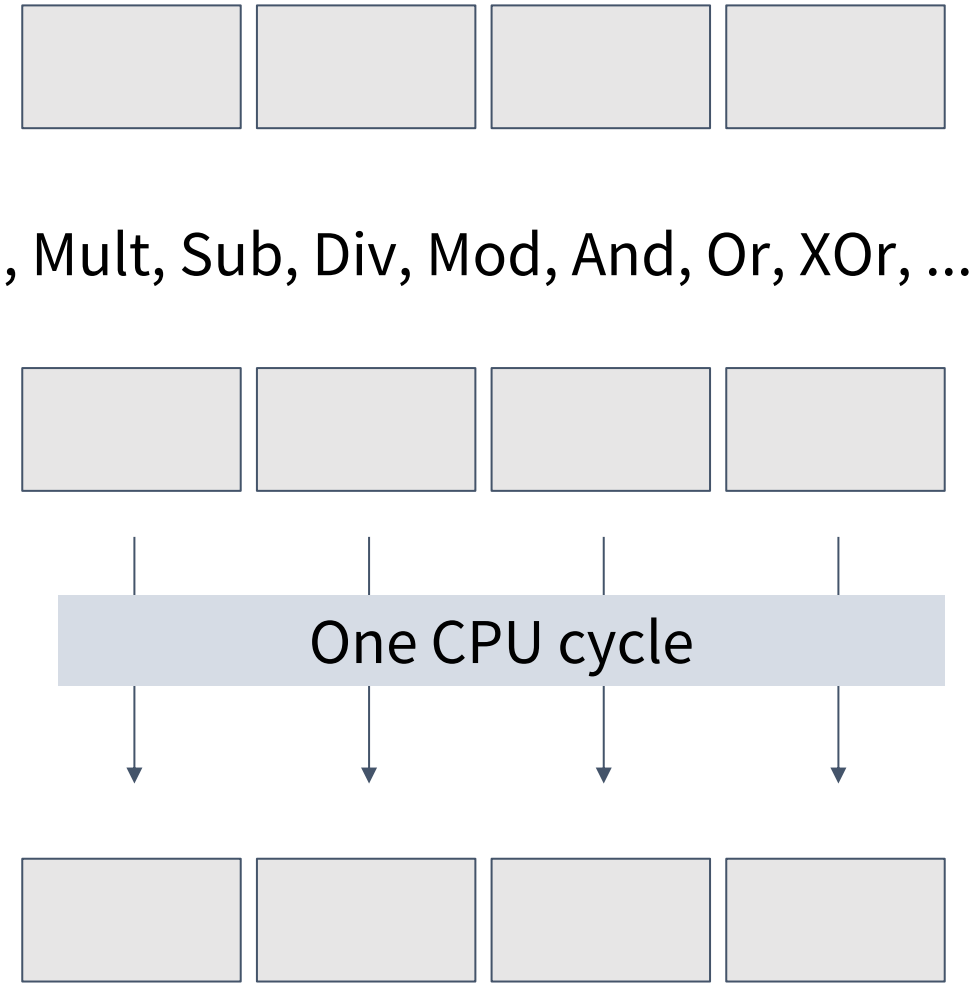
Parsing



- Fast parsing
- Modern hardware

Hardware Acceleration

- Single-Instruction-Multiple-Data (SIMD)
- Allows arithmetic operations on vectors of length 4 of words to be done in a single CPU cycle.
- Intel AVX-512 instructions can operate on 4 X 128-bit = 512-bit.
- Modern CPUs can operate on 4 X 64-bit integer operations.

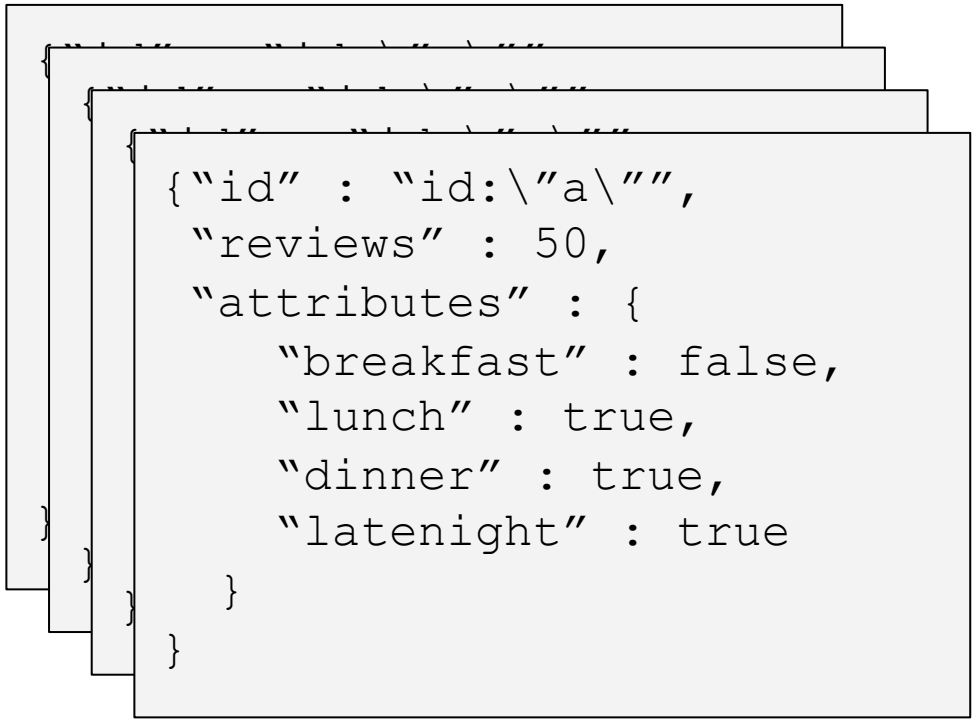


Mison

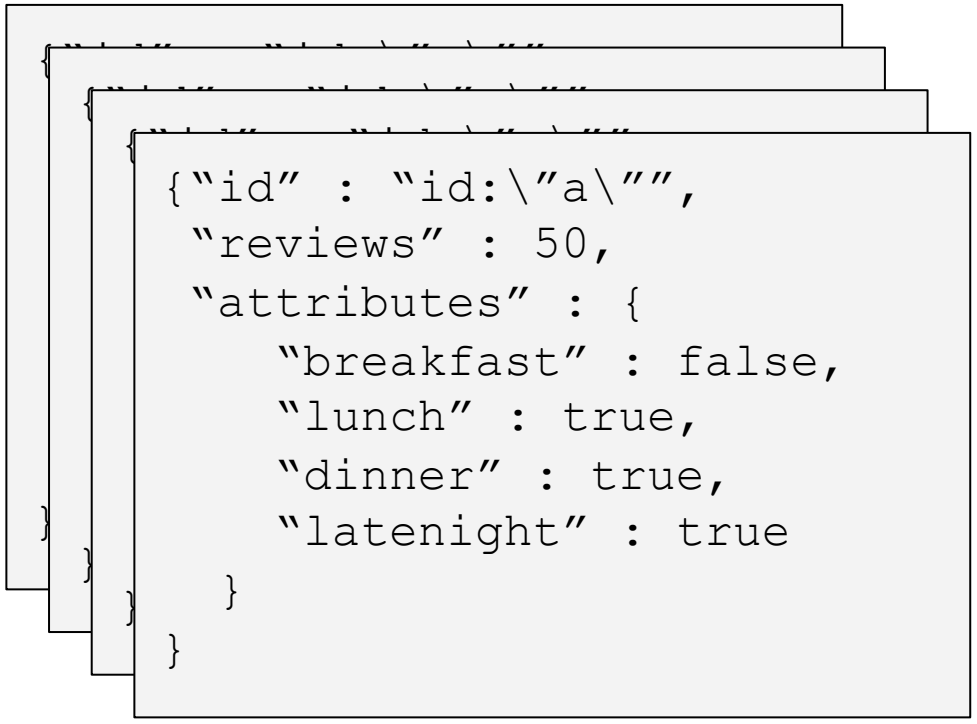
- Performs fast parsing of JSON records in two passes by vectorized SIMD processing.
- Evaluates path queries during parsing.
- Builds multi-level bitmap index to identify field boundaries.
- Speculate the ordinal position of queried fields using gathered statistics.

Li, Yinan, et al. "Mison: a fast JSON parser for data analytics." VLDB 2017.

Mison


{“id” : “id:\”a\””,
“reviews” : 50,
“attributes” : {
“breakfast” : false,
“lunch” : true,
“dinner” : true,
“latenight” : true
}
}

Mison



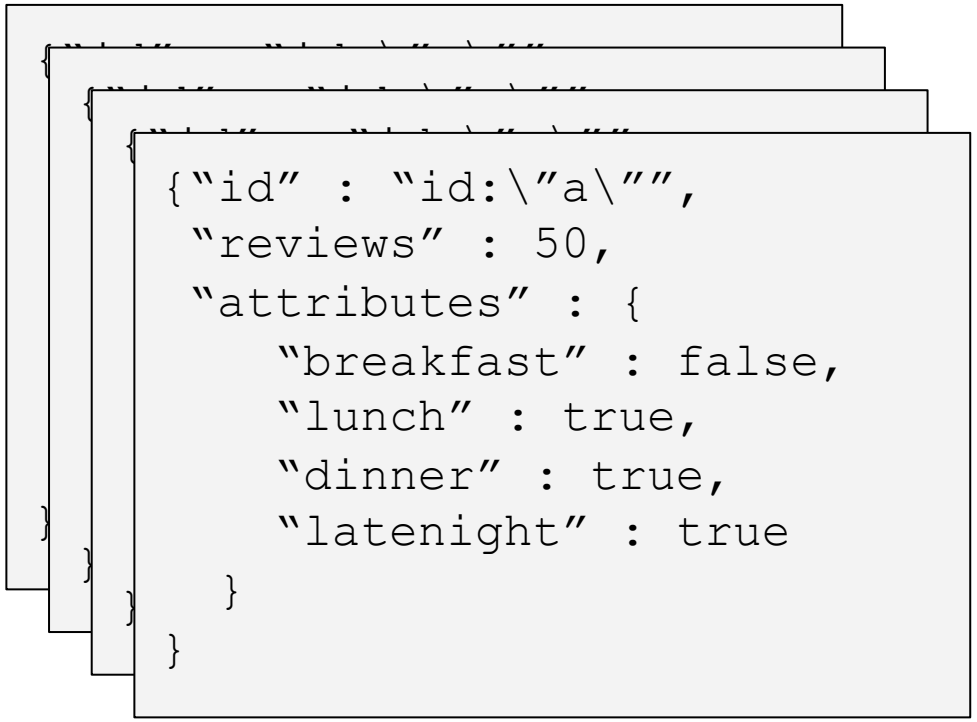
{"id" : "id:\"a\"", "reviews" : 50, "a

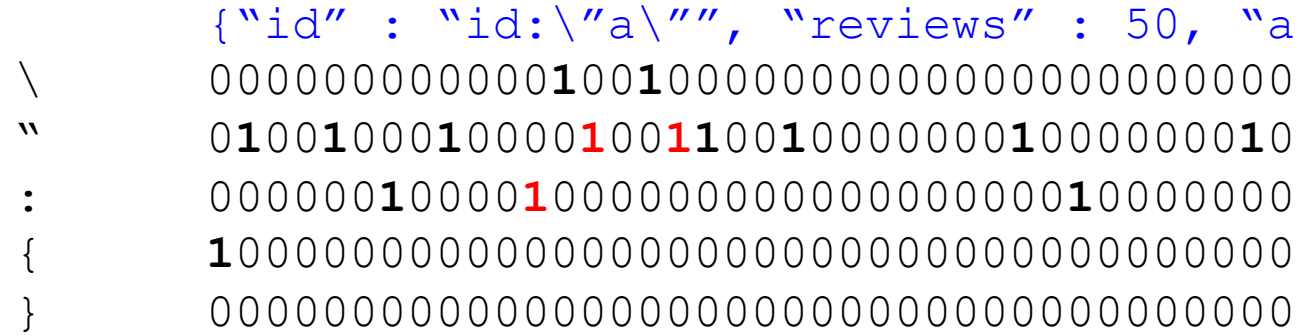
Byte stream

\
"
:
{
}

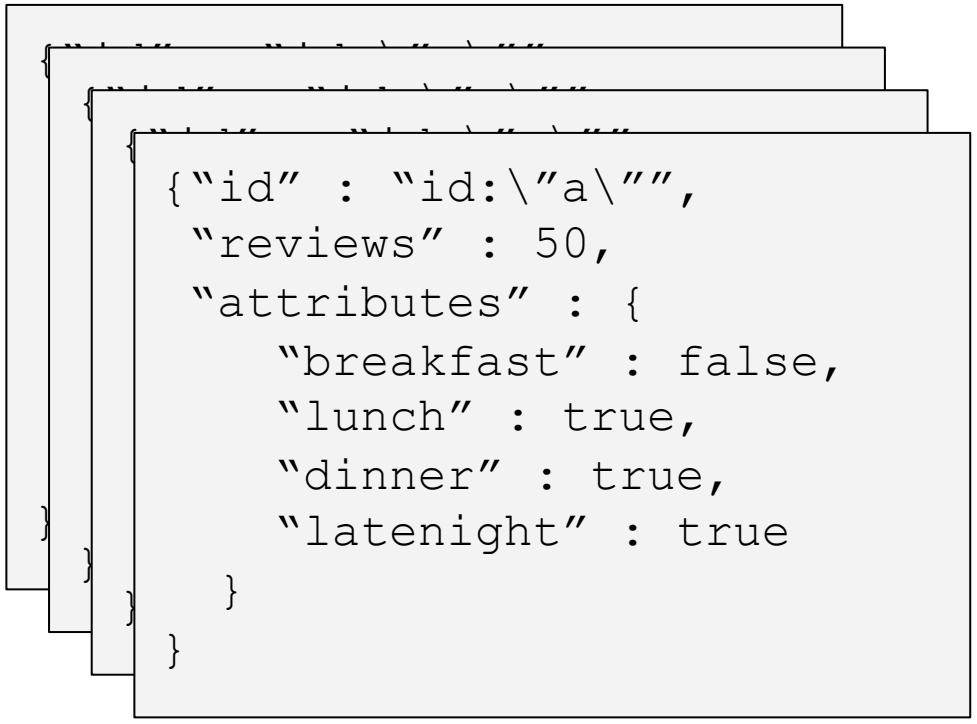
Structural characters

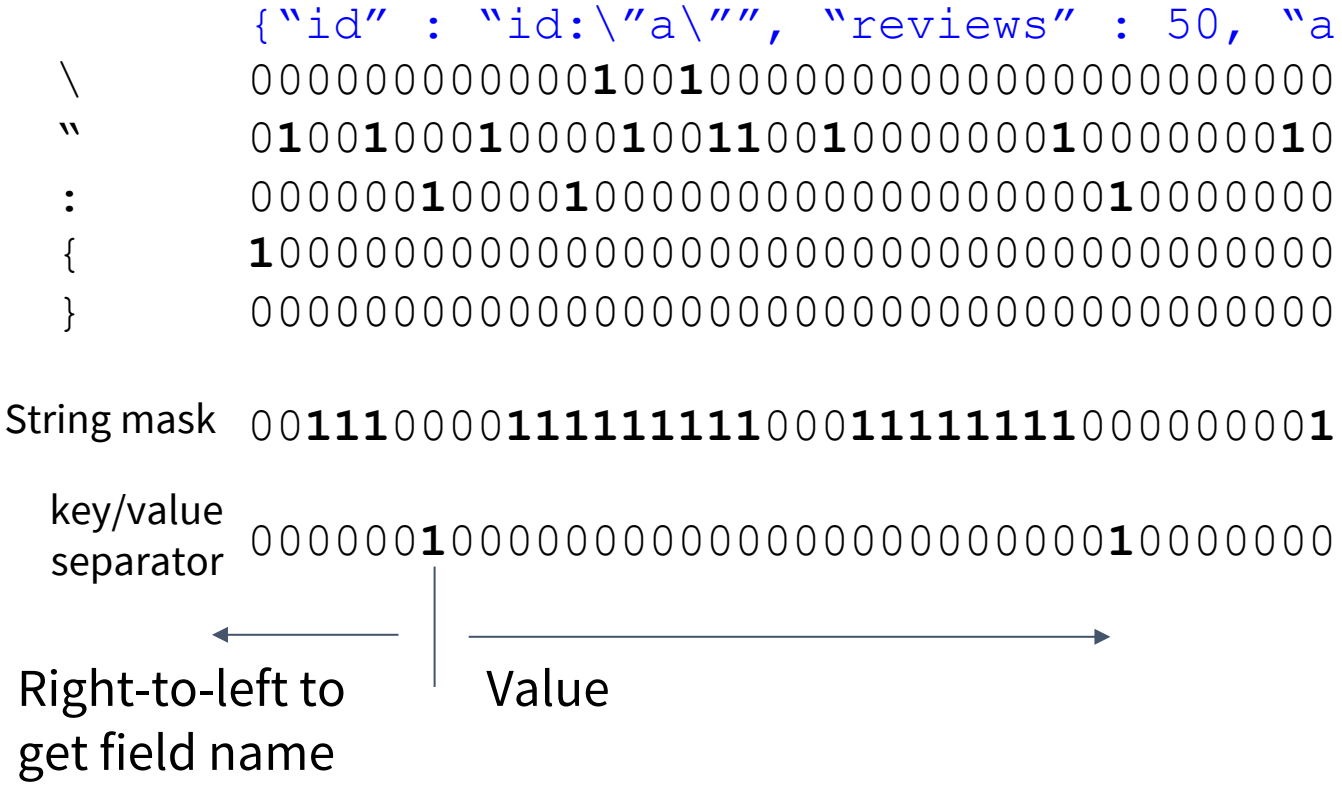
Mison


{“id” : “id:\”a\””,
“reviews” : 50,
“attributes” : {
“breakfast” : false,
“lunch” : true,
“dinner” : true,
“latenight” : true
}
}


{“id” : “id:\”a\””, “reviews” : 50, “a
\ 0000000000001001000000000000000000000
“ 0100100010000100110010000000100000010
: 0000001000010000000000000000010000000
{ 1000000000000000000000000000000000000
} 0000000000000000000000000000000000000

Mison


{“id” : “id:\”a\””,
“reviews” : 50,
“attributes” : {
“breakfast” : false,
“lunch” : true,
“dinner” : true,
“latenight” : true
}
}


{“id” : “id:\”a\””, “reviews” : 50, “a
\ 0000000000001001000000000000000000000
“ 0100100010000100110010000000100000010
: 0000001000010000000000000000010000000
{ 1000000000000000000000000000000000000
} 0000000000000000000000000000000000000
String mask 0011100001111111100011111110000000 1
key/value separator 0000001000000000000000000010000000
Right-to-left to get field name
Value

Mison

```
          {“id” : “id:\”a\””, “reviews” : 50, “a
\         0000000000001001000000000000000000000
“         0100100010000100110010000000100000000
:         0000001000010000000000000000010000000
{         1000000000000000000000000000000000000
}         0000000000000000000000000000000000000

String mask  0011100001111111100011111110000000 1

key/value separator  0000001000000000000000000010000000
```

Performed by SIMD accelerated bit operations

Sample query with two levels: c**ountry . capital**

Build multi-level index to answer the query.

```
“country” : { “population” : false, “capital” : true
0000000000100000000000000000000000000000000000000000
0000000000000000000000000100000000000000001000000
```

Mison

- Multi-level bitmap index of key/value separators allows **fast index-based scan** of JSON.
- Speculatively assumes position index of query field is the same for all JSON records.
- Verifies that the field name is correct. If not, then the speculation is incorrect.
  - Uses a **bloom filter** to check if query field exists.
  - **Fallback to sequential scan** to find the query field.

Only preserved the queried field values.
We may need to preserve the entire JSON for ad-hoc data analytics.

Simdjson: Parsing Gigabytes of JSON per Second

- Performs two pass parsing of JSON documents
- First pass indexes the structures with a bitmap.
  - SIMD accelerated, and branching free
  - Structural validation
- The second pass builds the complete parse tree on *tapes.*
  - SIMD accelerated
  - UTF-8 validation
  - Tape consists of two arrays that encode the JSON parse tree.
- This is the fastest JSON parser design.

Langdale, Geoff, and Daniel Lemire. "Parsing Gigabytes of JSON per Second." *arXiv preprint arXiv:1902.08318* (2019).

Speculative Distributed CSV Parsing

- Accelerates CSV parsing using parallel processing of non-overlapping chunking of the CSV file.

- Single pass (mostly) using speculative parsing in parallel.

Ge, Chang, et al. "Speculative Distributed CSV Data Parsing for Big Data Analytics." SIGMOD 2019

Speculative Distributed CSV Parsing

```
XXXX , XX , XXXXX XXXX , “XXXXX”
XXX  , XX , XXXXX XXXX , “XXXXX”
XXX  , XX , XXXXX XXXX , “XXXXX”
XXX  , XX , XXXXX XXXX , “X XXXX”
XXXX , XX , XXXXX XXXX , “XXXXX”
XXX  , XX , XXXXX XXXX , “XXXXX”
```

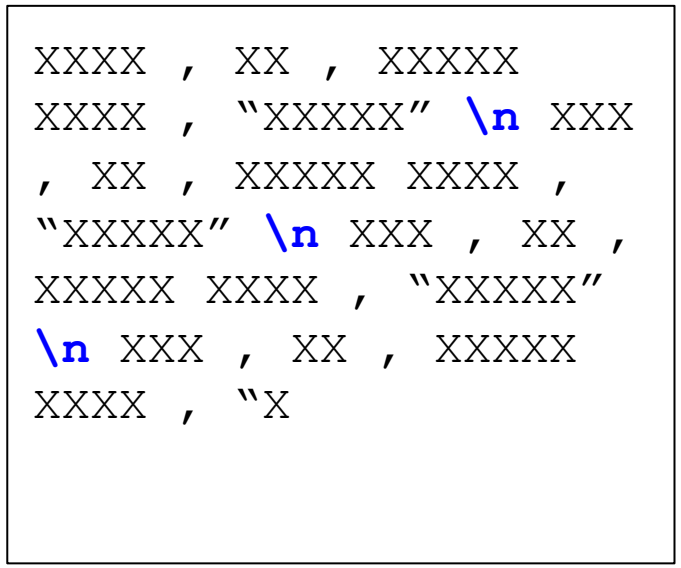
Logical layout

RFC-4180 standard

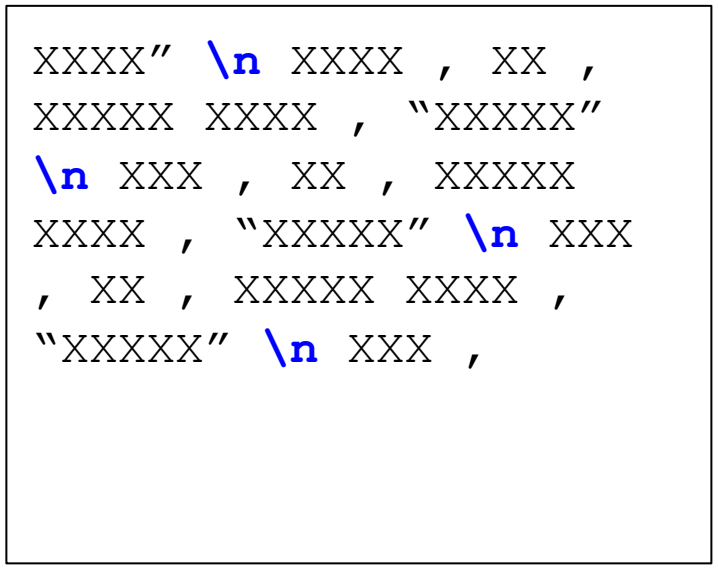
```
XXXX , XX , XXXXX XXXX ,
“XXXXX” \n XXX , XX , XXXXX
XXXX , “XXXXX” \n XXX , XX ,
XXXXX XXXX , “XXXXX” \n XXX ,
XX , XXXXX XXXX , “X XXXX” \n
XXXX , XX , XXXXX XXXX ,
“XXXXX” \n XXX , XX , XXXXX
XXXX , “XXXXX” \n XXX , XX ,
XXXXX XXXX , “XXXXX” \n XXX ,
XXXX, “XXXXX” \n XXXX , XX ,
XXXXX XXXX , “XXXXX” \n XXX ,
XX , XXXXX XXXX , “XXXXX” \n
XXX , XX , XXXXX XXXX , “XXXXX”
\n XXX , XX , XXXXX XXXX ,
“XXXXX” \n
```

Speculative Distributed CSV Parsing

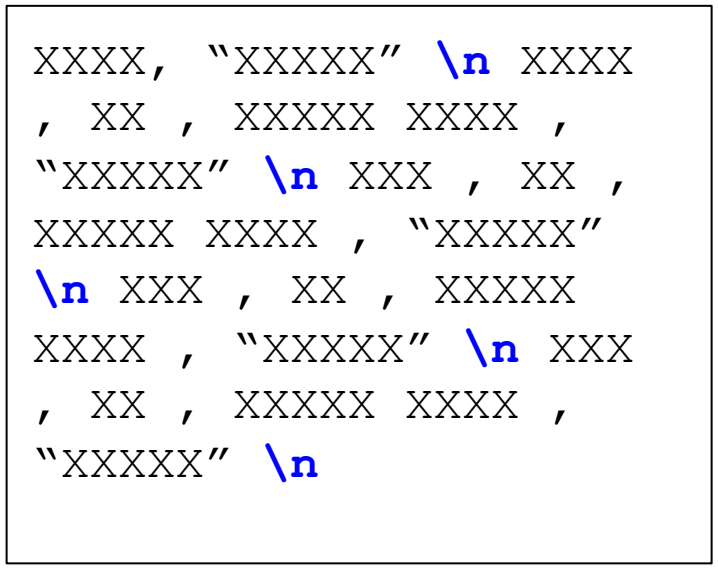
Chunk 0



Chunk 1



Chunk 2



CSV is assumed to be distributed in chunks.

Speculative Distributed CSV Parsing

```
XXXX , XX , XXXXX
XXXX , "XXXXX" \n XXX
, XX , XXXXX XXXX ,
"XXXXX" \n XXX , XX ,
XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX
XXXX , "X
```

Chunk1 with incomplete record

```
XXXX" \n   XXXX , XX ,
XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX
XXXX , "XXXXX" \n XXX
, XX , XXXXX XXXX ,
"XXXXX" \n XXX ,
```

Chunk2 with incomplete record

```
XXXX, "XXXXX"\n  XXXX
, XX , XXXXX XXXX ,
"XXXXX" \n XXX , XX ,
XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX
XXXX , "XXXXX" \n XXX
, XX , XXXXX XXXX ,
"XXXXX" \n
```

Speculative Distributed CSV Parsing

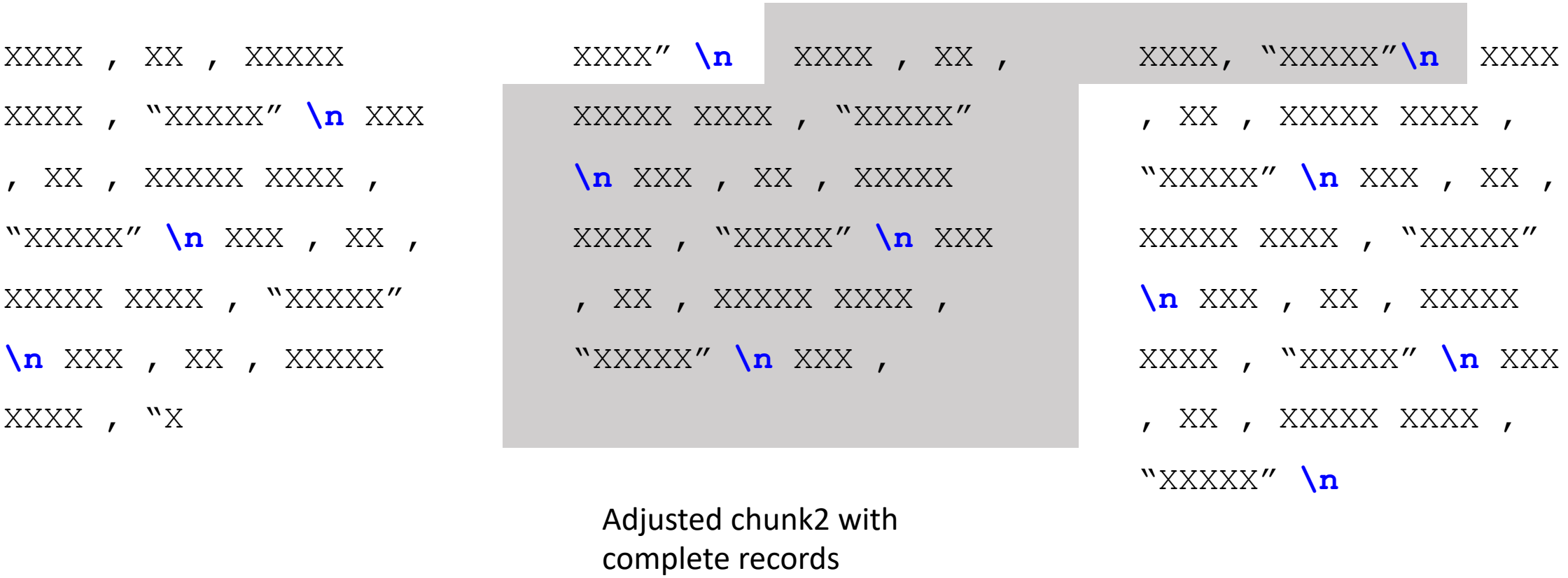
```
XXXX , XX , XXXXX
XXXX , "XXXXX" \n XXX
, XX , XXXXX XXXX ,
"XXXXX" \n XXX , XX ,
XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX
XXXX , "X
```

```
XXXX" \n  XXXX , XX ,
XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX
XXXX , "XXXXX" \n XXX
, XX , XXXXX XXXX ,
"XXXXX" \n XXX ,
```

```
XXXX, "XXXXX"\n  XXXX
, XX , XXXXX XXXX ,
"XXXXX" \n XXX , XX ,
XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX
XXXX , "XXXXX" \n XXX
, XX , XXXXX XXXX ,
"XXXXX" \n
```

Adjusted chunk1 with complete records

Speculative Distributed CSV Parsing



Adjusted chunk2 with complete records

Speculative Distributed CSV Parsing

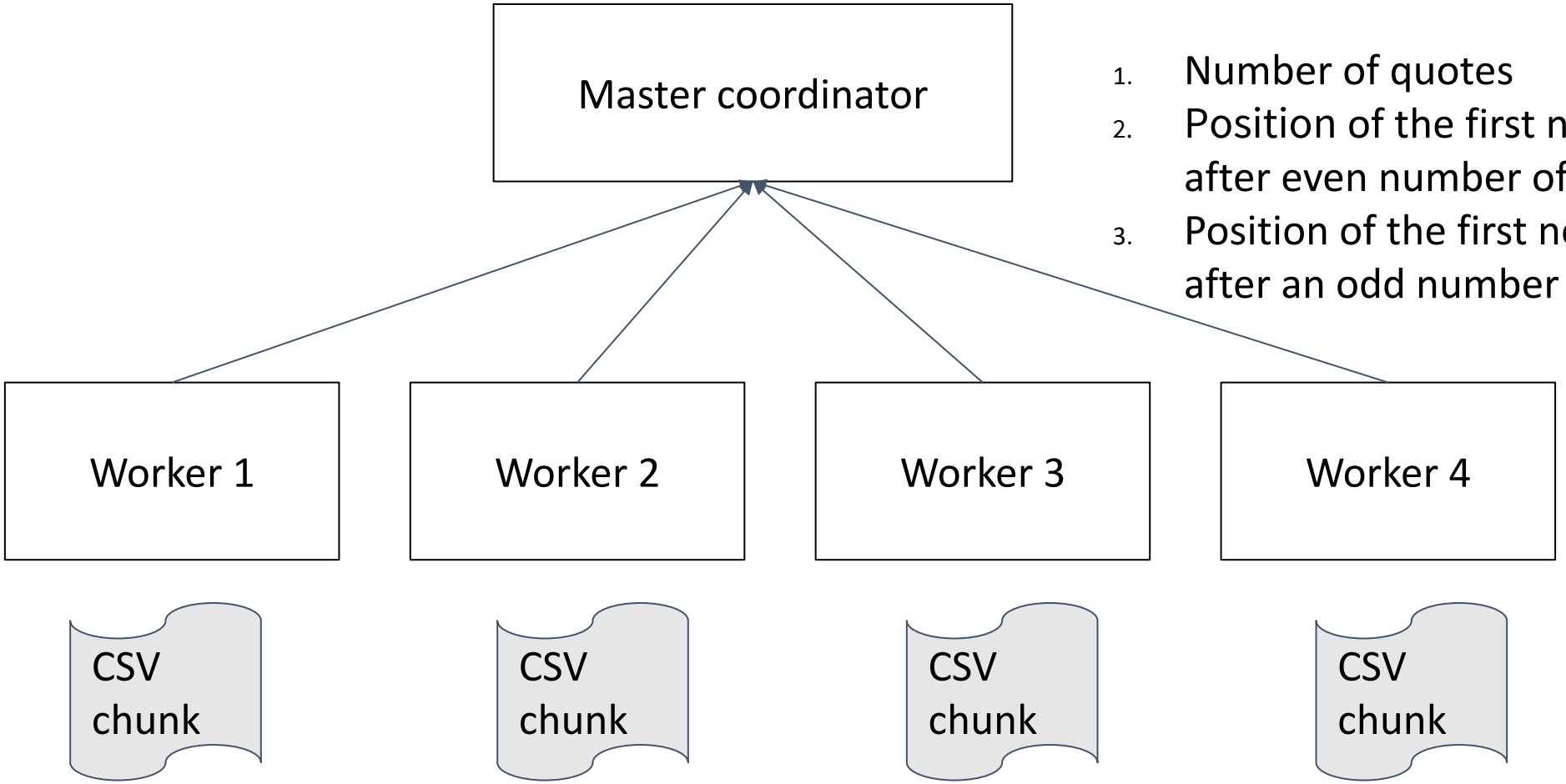
```
XXXX , XX , XXXXX
XXXX , "XXXXX" \n XXX
, XX , XXXXX XXXX ,
"XXXXX" \n XXX , XX ,
XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX
XXXX , "X
```

```
XXXX" \n   XXXX , XX ,
XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX
XXXX , "XXXXX" \n XXX
, XX , XXXXX XXXX ,
"XXXXX" \n XXX ,
```

```
XXXX, "XXXXX"\n  XXXX
, XX , XXXXX XXXX ,
"XXXXX" \n XXX , XX ,
XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX
XXXX , "XXXXX" \n XXX
, XX , XXXXX XXXX ,
"XXXXX" \n
```

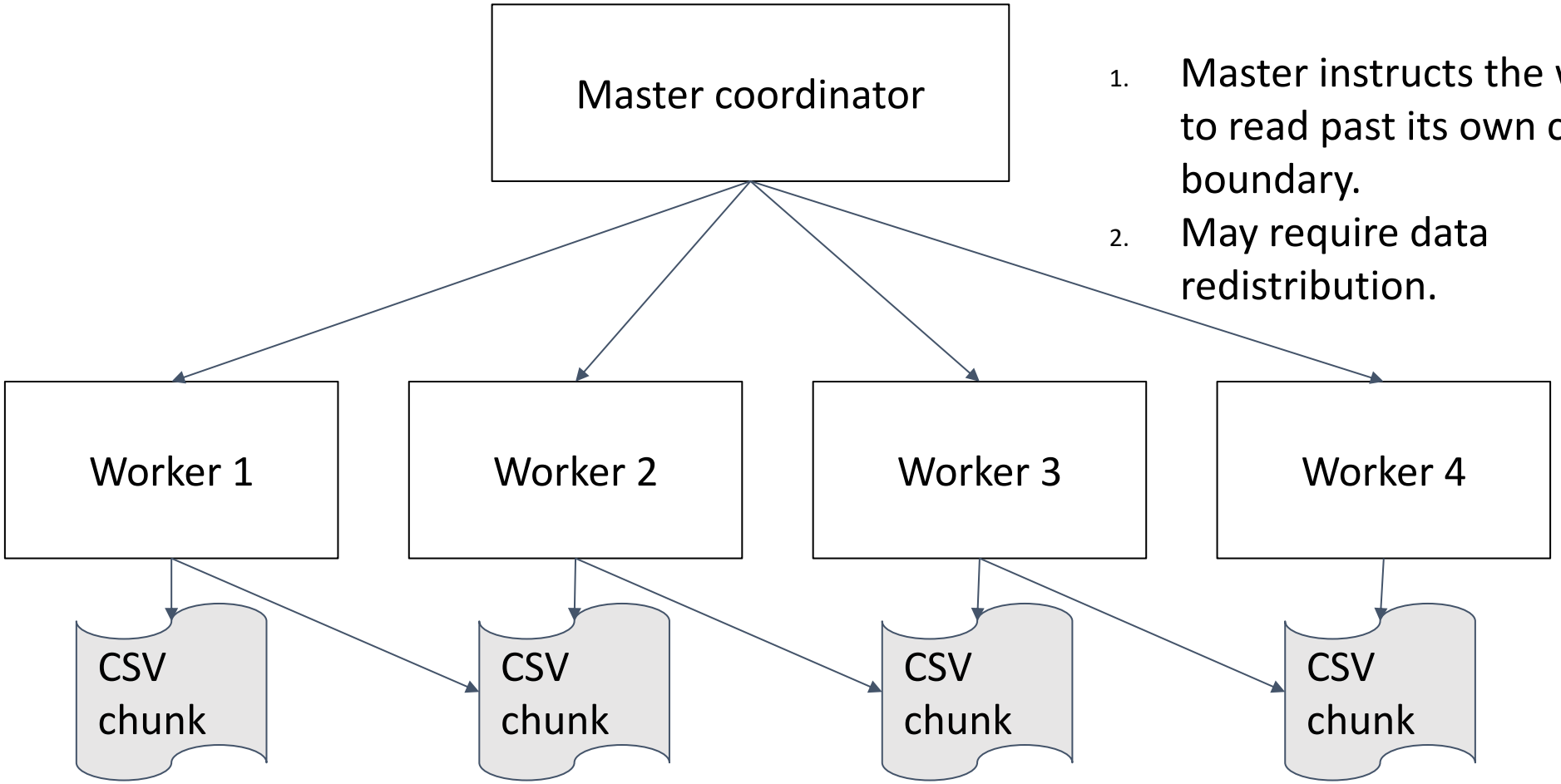
Adjusted chunk3 with complete records

Speculative Distributed CSV Parsing



1. Number of quotes
2. Position of the first newline after even number of quotes
3. Position of the first newline after an odd number

Speculative Distributed CSV Parsing



1. Master instructs the workers to read past its own chunk boundary.
2. May require data redistribution.

Speculative Distributed CSV Parsing

Parsing & adjustment in one pass using speculation.

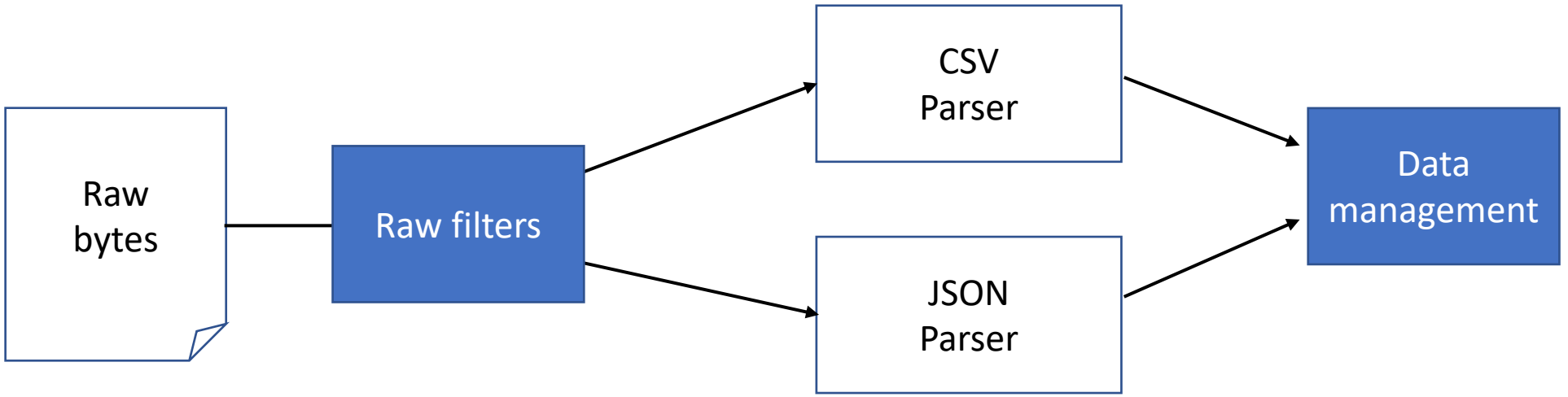
```
xxxx" \n xxxx , xx ,
xxxxx xxxx , "xxxxx"
\n xxx , xx , xxxxx
xxxx , "xxxxx" \n xxx
, xx , xxxxx xxxx ,
"xxxxx" \n xxx ,
```

Worker parses the chunks **before** the adjustments are computed. The worker speculates the start of its chunk.

The master performs verification.

If verification fails, the worker needs to perform parsing again on the adjusted chunk.

Pre-parsing filtering

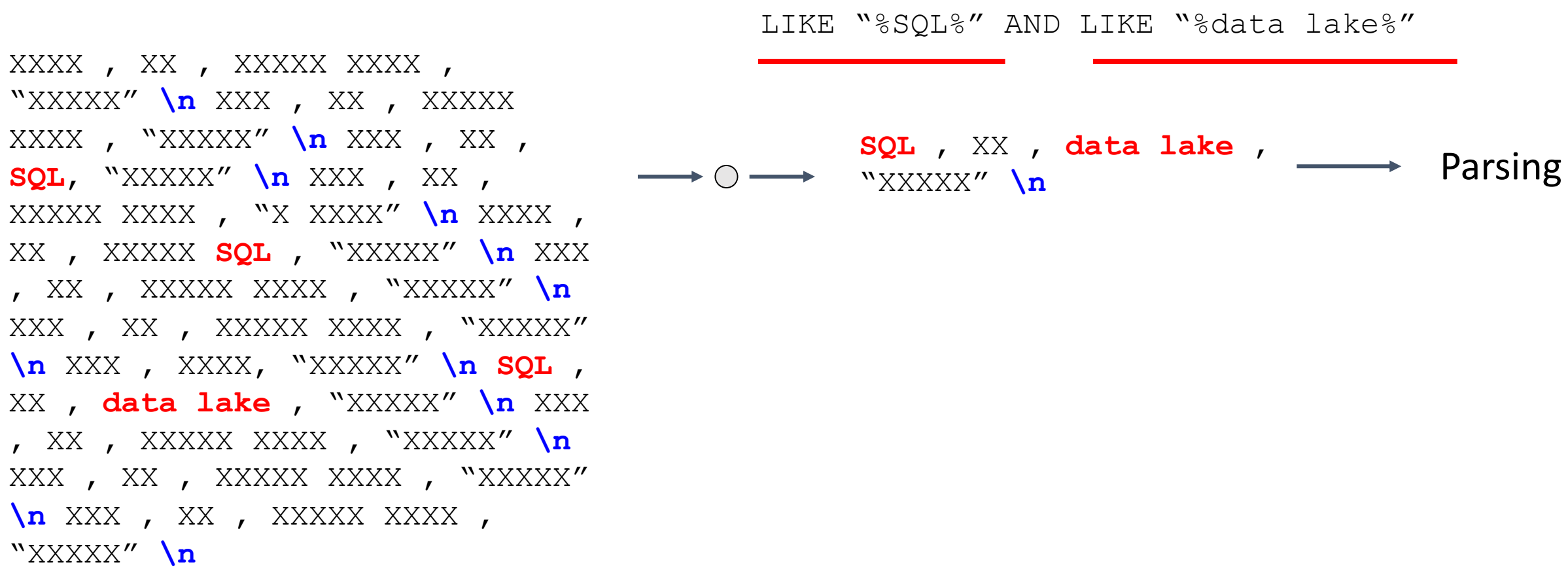


Sparser

- Performs safe filtering on the raw byte stream that guarantees to:
  - **Achieve 100% recall**
  - **Preserves data format**
- Raw filters (RF) can be composed in cascade to improve the pruning power.
- Sparser optimizes the RF cascade by cost estimation.

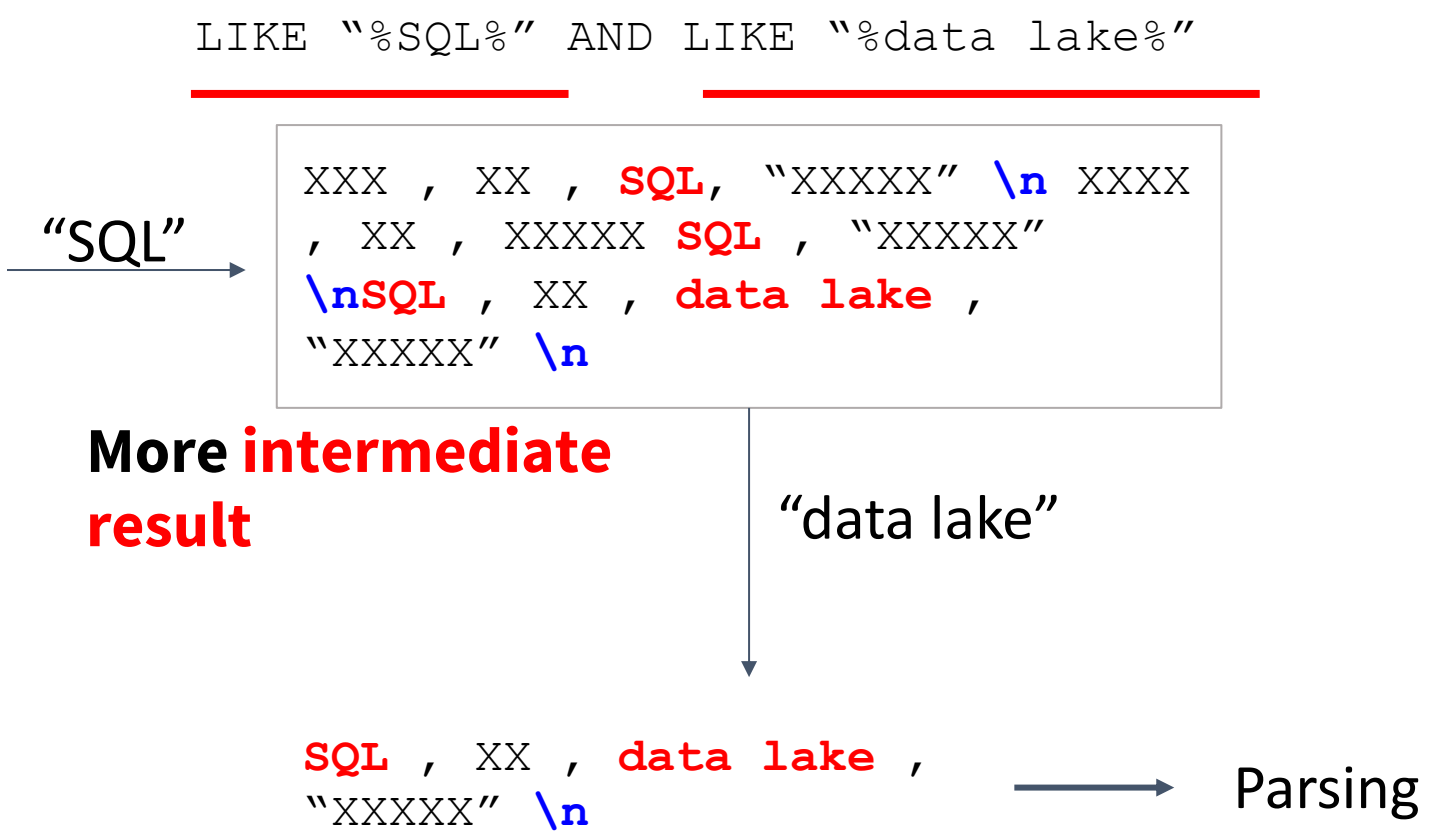
Palkar, Shoumik, et al. "Filter before you parse: Faster analytics on raw data with sparser." Proceedings of the VLDB Endowment 11.11 (2018): 1576-1589.

Sparser


XXXX , XX , XXXXX XXXX ,
"XXXXX" \n XXX , XX , XXXXX
XXXX , "XXXXX" \n XXX , XX ,
SQL, "XXXXX" \n XXX , XX ,
XXXXX XXXX , "X XXXX" \n XXXX ,
XX , XXXXX SQL , "XXXXX" \n XXX
, XX , XXXXX XXXX , "XXXXX" \n
XXX , XX , XXXXX XXXX , "XXXXX"
\n XXX , XXXX, "XXXXX" \n SQL ,
XX , data lake , "XXXXX" \n XXX
, XX , XXXXX XXXX , "XXXXX" \n
XXX , XX , XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX XXXX ,
"XXXXX" \n
LIKE "%SQL%" AND LIKE "%data lake%"
SQL , XX , data lake ,
"XXXXX" \n
Parsing

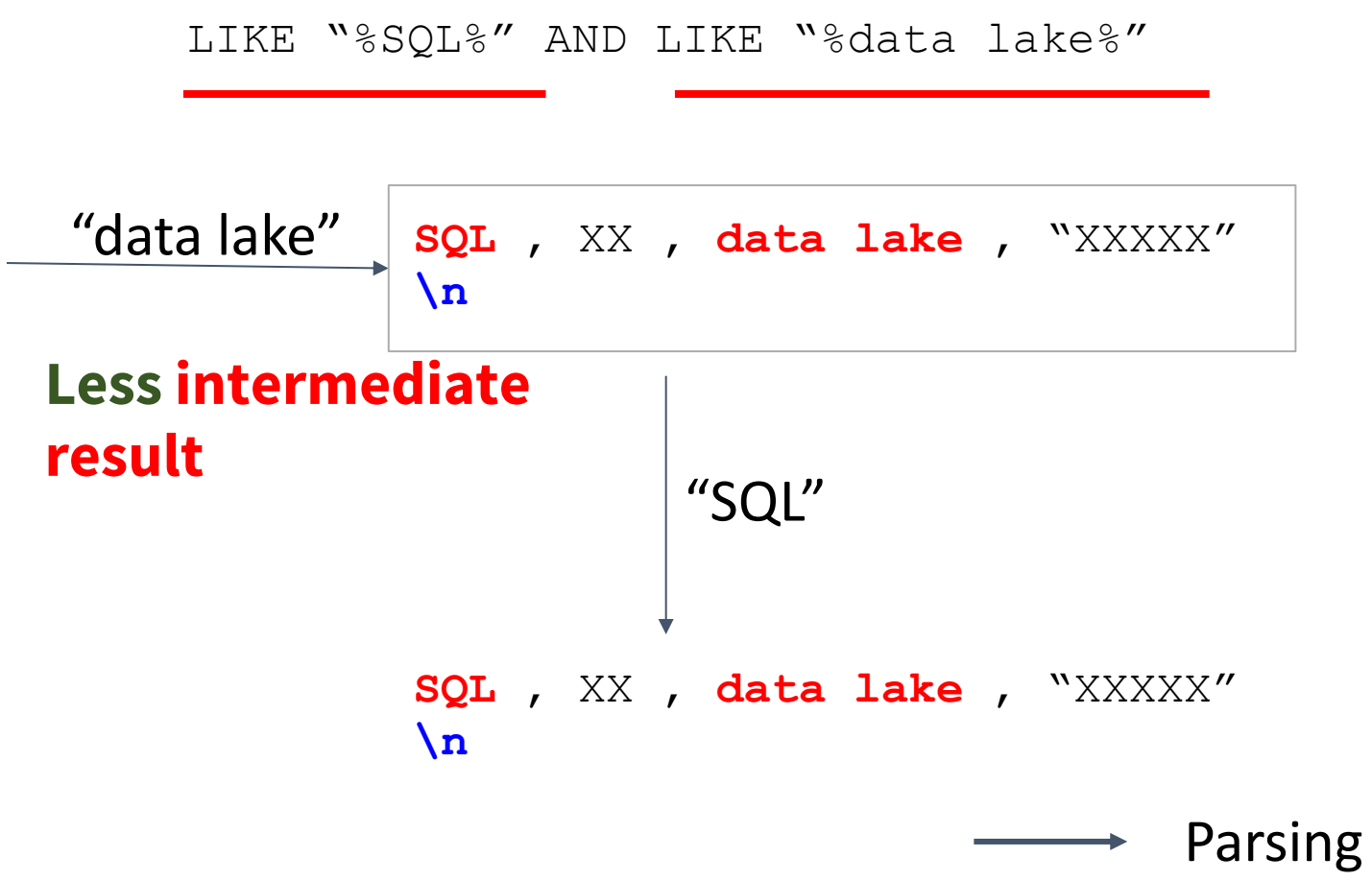
Sparser

```
XXXX , XX , XXXXX XXXX ,
"XXXXX" \n XXX , XX , XXXXX
XXXX , "XXXXX" \n XXX , XX ,
SQL, "XXXXX" \n XXX , XX ,
XXXXX XXXX , "X XXXX" \n XXXX ,
XX , XXXXX SQL , "XXXXX" \n XXX
, XX , XXXXX XXXX , "XXXXX" \n
XXX , XX , XXXXX XXXX , "XXXXX"
\n XXX , XXXX, "XXXXX" \n SQL ,
XX , data lake , "XXXXX" \n XXX
, XX , XXXXX XXXX , "XXXXX" \n
XXX , XX , XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX XXXX ,
"XXXXX" \n
```



Sparser

```
XXXX , XX , XXXXX XXXX ,
"XXXXX" \n XXX , XX , XXXXX
XXXX , "XXXXX" \n XXX , XX ,
SQL, "XXXXX" \n XXX , XX ,
XXXXX XXXX , "X XXXX" \n XXXX ,
XX , XXXXX SQL , "XXXXX" \n XXX
, XX , XXXXX XXXX , "XXXXX" \n
XXX , XX , XXXXX XXXX , "XXXXX"
\n XXX , XXXX, "XXXXX" \n SQL ,
XX , data lake , "XXXXX" \n XXX
, XX , XXXXX XXXX , "XXXXX" \n
XXX , XX , XXXXX XXXX , "XXXXX"
\n XXX , XX , XXXXX XXXX ,
"XXXXX" \n
```



Sparser

1. Builds a set of RF based on the query predicates.
2. Estimates the cost and passthrough rates based on a small sample of the raw records. The statistics also capture possible correlations in co-occurrences.
3. Select the RF and search for optimal cascade based on the estimated cost and passthrough rates.

PADS/ML: generic parsing

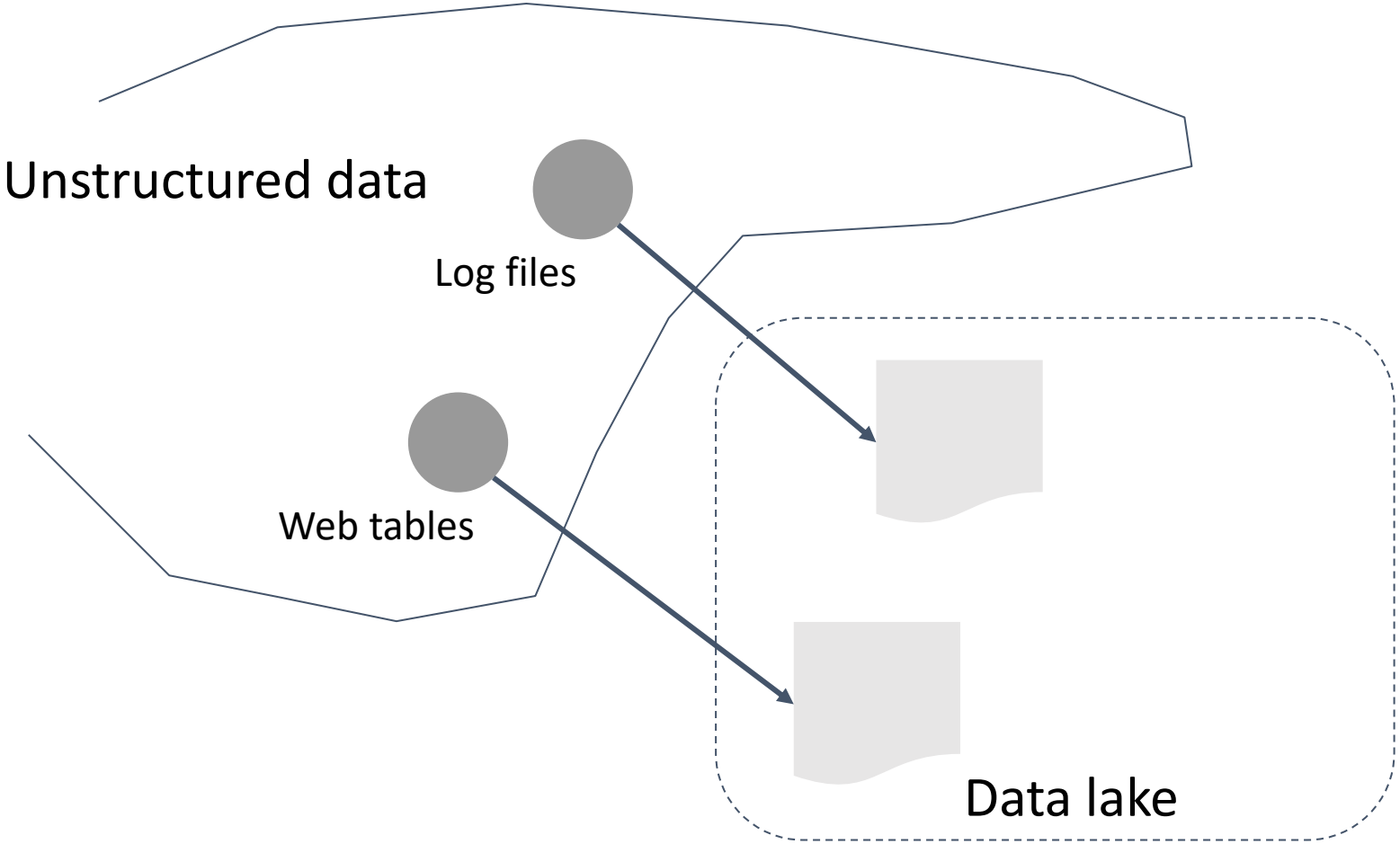
- More specialized and complex structures would require manually specified parser.
- Systems such as PADS/ML can generate the parser based on grammar-based data description.

Mandelbaum, Yitzhak, et al. "PADS/ML: A functional data description language." *ACM SIGPLAN Notices*. Vol. 42. No. 1. ACM, 2007.

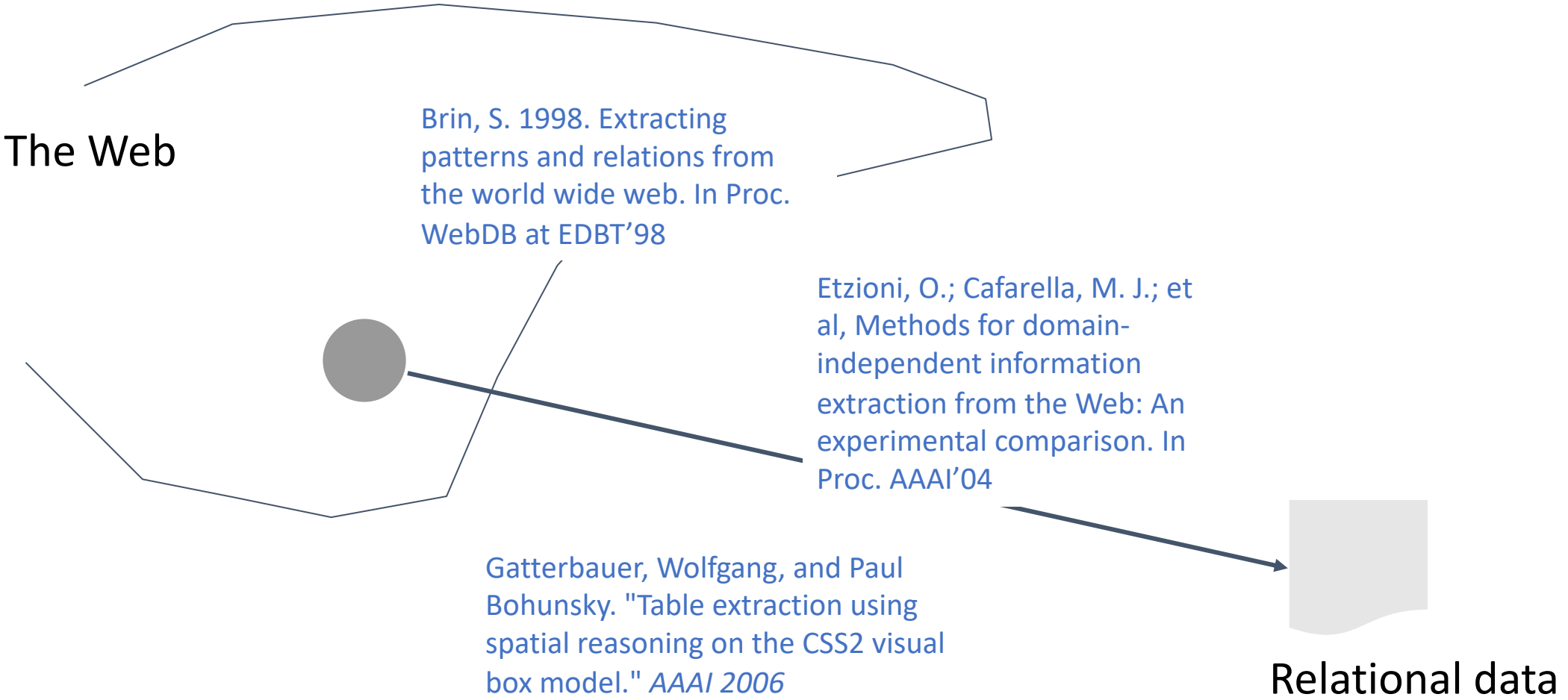
Zhu, Kenny Q., Kathleen Fisher, and David Walker. "Learnpads++: Incremental inference of ad hoc data formats." *International Symposium on Practical Aspects of Declarative Languages*. 2012.

Extraction

Extraction



Extraction


The Web
Brin, S. 1998. Extracting patterns and relations from the world wide web. In Proc. WebDB at EDBT'98
Etzioni, O.; Cafarella, M. J.; et al, Methods for domain-independent information extraction from the Web: An experimental comparison. In Proc. AAAI'04
Gatterbauer, Wolfgang, and Paul Bohunsky. "Table extraction using spatial reasoning on the CSS2 visual box model." *AAAI 2006*
Relational data

Web Tables: The Power of Tables on the Web

- Analyzes the **Google Web Crawl** (2008).
- Extracts data tables in (billions) HTML:

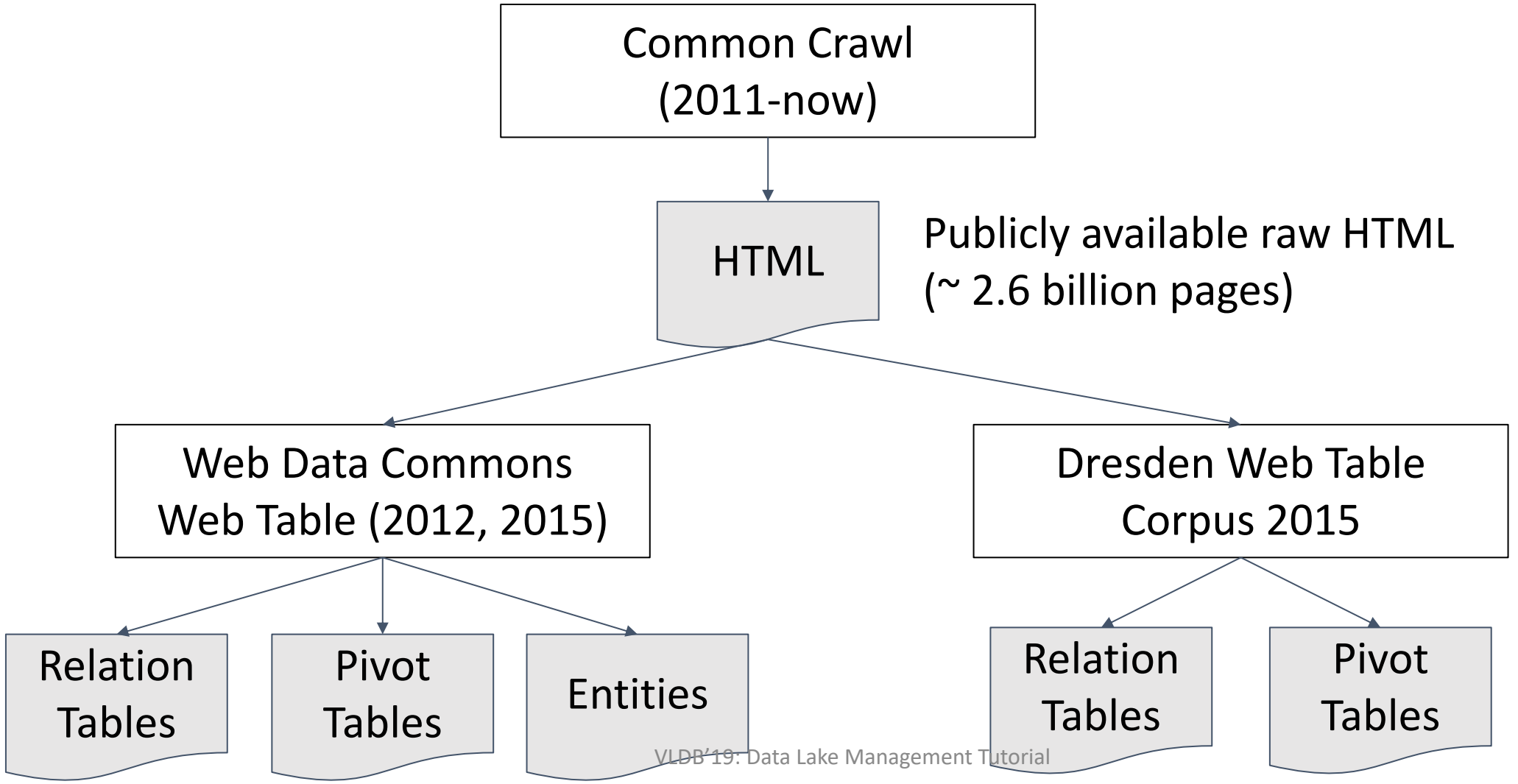
  154 million data tables (with clean attributes)
- Attribute occurrence statistics are collected and stored in attribute correlation statistics database (ACSDb).
- Demonstrates interesting applications:
  - Schema auto-complete
  - Attribute synonyms
  - Join graph traversal by attribute labels

Cafarella, Michael J., et al. "Webtables: exploring the power of tables on the web." VLDB 2008

Data on the Web

Publicly accessible Web Data?

Public Access of Web Tables



WDC Web Tables

- 233 million Web tables (2015)
  - 90 million relational tables
  - 140 million entities
  - 3 million matrices (pivot tables)
- Corpus also provides metadata containing:
  - Orientation of the records (row vs column)
  - Title of HTML page
  - Text surrounding the extracted table

Lehmberg, Oliver, et al. "A large public corpus of web tables containing time and context metadata." WWW 2016.

Dresden Web Table Corpus

- 125 million tables
  - Horizontal, vertical and matrix (pivot) tables
- The data table classifier is learned from a training set.
- The classifier
  - 127 features are generated for each HTML <table>.
  - Max and average cell count per row and column.
  - Max and average cell length in characters.
  - Frequency of cells containing <TH>, <A>, <IMG> …
- The training data with 2000 data tables is manually curated.
- Correlation based feature selection (CFS) is used to perform feature selection, producing 30 useful features.

Eberius, Julian, et al. "Building the dresden web table corpus: A classification approach." Big Data Computing (BDC) 2015.

DATAMARAN – Logs to Structure

- Designed to extract **structured records** from log files with unknown schema.
- Assume a single record can span over multiple lines.
  - Designed to:
    - Identify field separation and record separation
    - Identify records from noise
    - Represents the record structure as a (restricted) regular expression
- Objectives:
  - Regularity scoring function
  - Coverage

Gao, Yihan, Silu Huang, and Aditya Parameswaran. "*Navigating the data lake with datamaran: automatically extracting structure from log datasets.*" SIGMOD 2018.

DATAMARAN – Logs to Structure

```
"id" : "1"
"zip": 10200
"lat": 34.539
"long": 67.382

"id" : "2"
"zip": 95201
"lat": 15.283
"long": 100.509
```

```
# HTTP traffic log
IP: 127.0.0.1
IP: 10.0.9.1
IP: 192.68.0.2

# SSH traffic log
IP: 127.0.0.1
IP: 10.0.9.1
IP: 192.68.0.2
```

```
[01:01:00] fc(7,9)
[01:02:00] fc(7,2)
[01:03:00] fc(1,9)
[01:04:00] fc(7,9,0)
```

DATAMARAN – Logs to Structure

```
"id" : "1"
"zip": 10200
"lat": 34.539
"long": 67.382
```

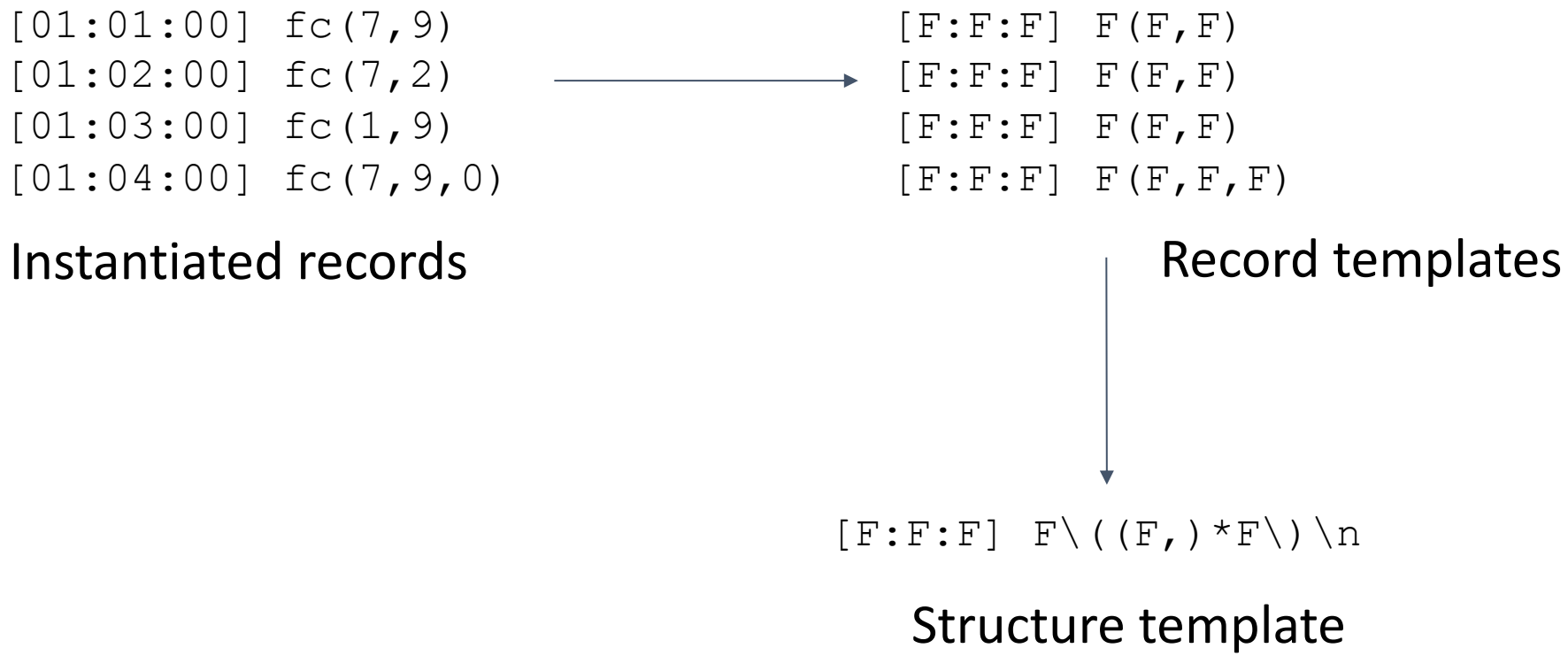
```
"id" : "2"
"zip": 95201
"lat": 15.283
"long": 100.509
```

```
# HTTP traffic log
IP: 127.0.0.1
IP: 10.0.9.1
IP: 192.68.0.2
```

```
# SSH traffic log
IP: 127.0.0.1
IP: 10.0.9.1
IP: 192.68.0.2
```

```
[01:01:00] fc(7,9)
[01:02:00] fc(7,2)
[01:03:00] fc(1,9)
[01:04:00] fc(7,9,0)
```

DATAMARAN – Logs to Structure



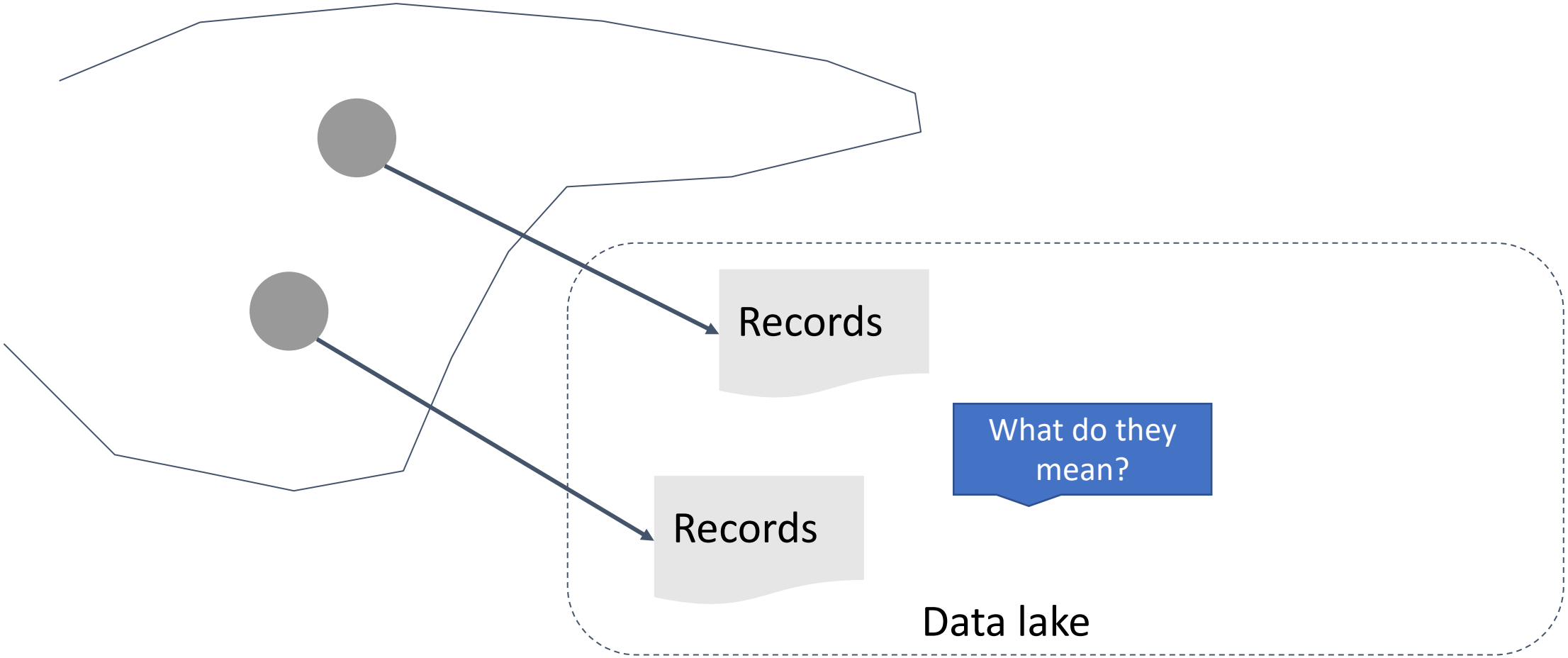
DeepDive – Extracting from Dark Data

- Extracting relational data from unstructured and text data.
  - Human is in the loop.
  - Text is preprocessed into tokens by NLP preprocessor.
  - A specification language, DDlog, based on datalog is used to specify *patterns* for record extraction.
  - DeepDive will:
    - Generate candidate records.
    - Generate a set of features used for extraction.
  - User provide **correct** records as training.
  - DeepDive generalizes the training extractions by
    - Train a graphical model
    - Use the graphical model to infer the validity of all candidate records.

Zhang, Ce, et al. "*Extracting databases from dark data with DeepDive.*" SIGMOD 2016

Column Type Inference of Ingested Data

Column Type Inference



Traditional Schema Matching

- Match attributes between two datasets.
  - Structural similarity
  - Data similarity
  - Semantic similarity (embeddings)
  - Preserves data integrity constraints
- Generic to data model
  - Relational
  - Semi-structured

Challenges of data lake:

- Massive number of tables
- Lack of ground truth
- Semantic understanding

Madhavan, Jayant, Philip A. Bernstein, and Erhard Rahm. "Generic schema matching with cupid." VLDB 2001.

Rahm, Erhard, and Philip A. Bernstein. "A survey of approaches to automatic schema matching." *the VLDB Journal* 10.4 (2001): 334-350.

Erhard Rahm, Eric Peukert: Large-Scale Schema Matching. Encyclo pedia of Big Data Technologies 2019

Ontology based annotation

a set of concepts and categories in a subject area or domain that shows their properties and the relations between them.

Yago

- Extracted from Wikipedia, WordNet and GeoNames
- As of 2019:
  - 17 million entities
  - 120 million facts on entities
  - 350,000 classes

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. "Yago: A Core of Semantic Knowledge." WWW 2007

Rebele, Thomas, et al. "Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames." *International Semantic Web Conference*. Springer, Cham, 2016.

Aumueller, David, et al. "Schema and ontology matching with COMA++." SIGMOD 2005.

Cardoso, Silvio Domingos, et al. "Leveraging the impact of ontology evolution on semantic annotations." *European Knowledge Acquisition Workshop*, 2016.

Annotating Web Tables

- Utilizing ontology (e.g., Yago) to annotate relational tables:
  - Cell values with entities in ontology.
  - Columns with types in ontology.
  - Binary relations between columns.
- Annotation is posed as a **statistical inference problem**:
  - Create **hidden variables** to represent entities, column types and binary relations.
  - Generate **observed features** based on syntactic similarity between cell values and values in ontology.
  - Annotation is done by inference on the hidden variables.

Limaye, Girija, Sunita Sarawagi, and Soumen Chakrabarti. "*Annotating and searching web tables using entities, types and relationships.*" VLDB 2010

Annotating Web Tables

- Training
  - Features between a cell and an entity in ontology are based on their string values.
  - The model weights are trained using training data.
- Inference
  - Binary relation variables can be used to encode graphs, allowing one to encode the k-colorability problem.
  - Inference is NP-hard.

Accuracy

- Type annotation: ~56%
- Entity annotation: ~83%
- Relation annotation: ~68%

The statistical model is to cope with the incomplete coverage of the Yago ontology.

Recovering Semantics of Tables on the Web

- A data driven approach based on **Google Web crawl**.
- Building an “is-a” database from Web crawls using patterns.
  - **1.5 M** (web extracted) vs **185K** (YAGO)
- Label tables using the “is-a” database
- Incorporate the semantic label into table search

Accuracy (table search based on annotation): ~ 79%

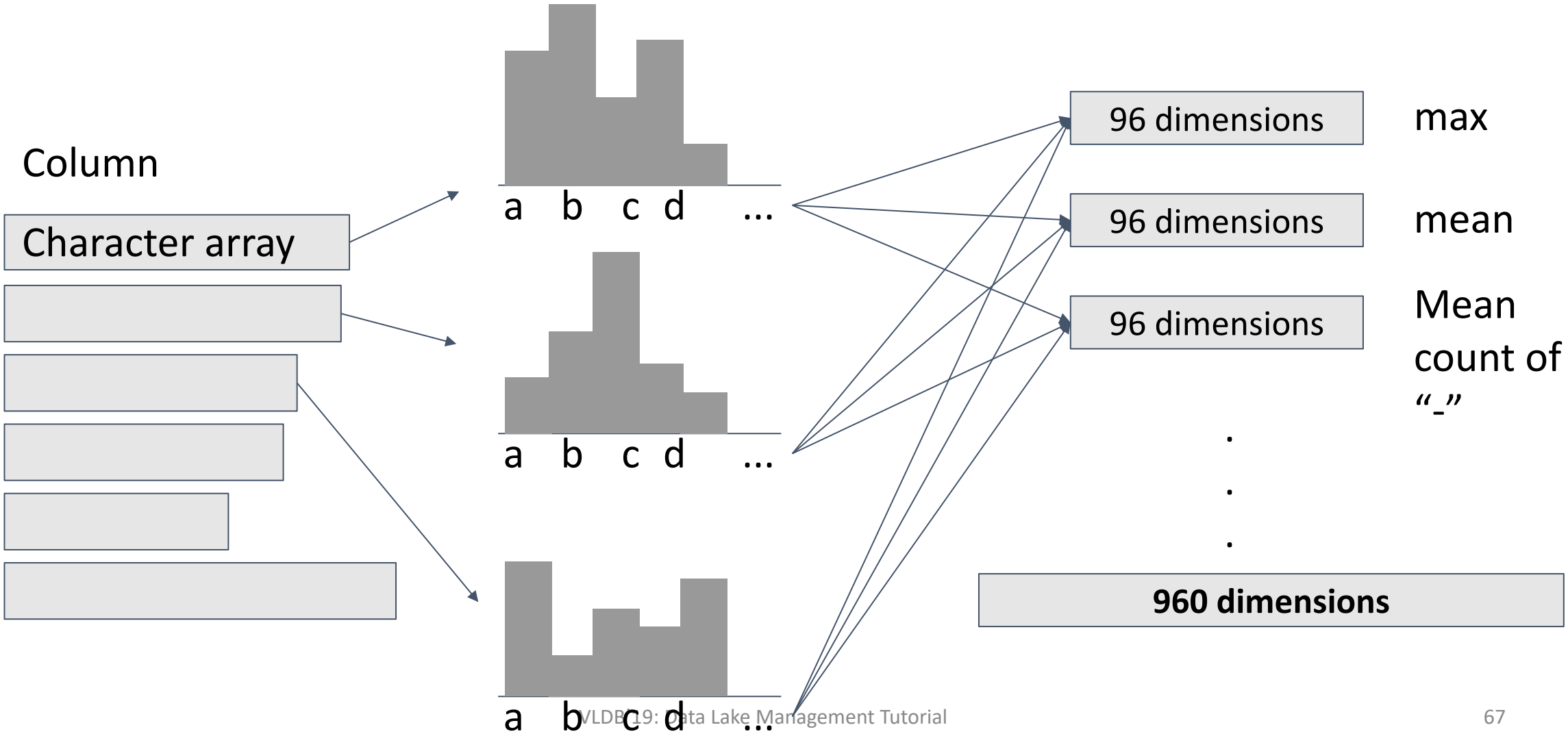
Venetis, Petros, et al. "*Recovering semantics of tables on the web*." VLDB 2011

Sherlock - Semantic Data Type Detection

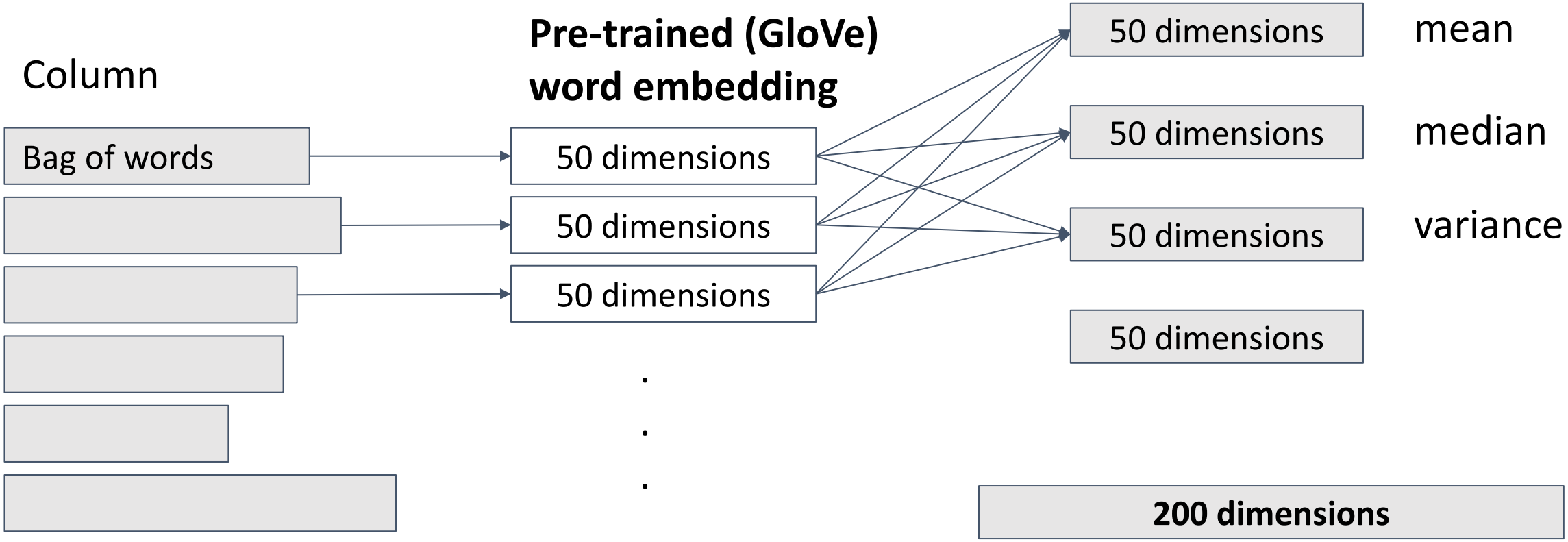
- A neural network approach to column type annotation.
- Each column is mapped to a 1588 dimensional feature vector.
- A 4-layer feedforward network is used to the type.
- Training data
  - DBPedia entity properties
  - Selected VizNet columns that appear in DBPedia as well.
  - ~ 680,000 columns in training data

Hulsebos, Madelon, et al. "*Sherlock: A Deep Learning Approach to Semantic Data Type Detection*." SIGKDD (2019).

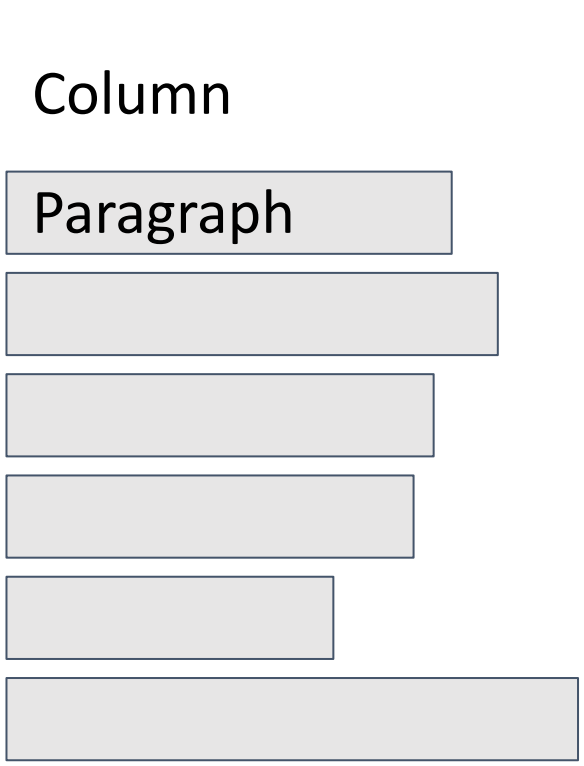
Sherlock - Semantic Data Type Detection

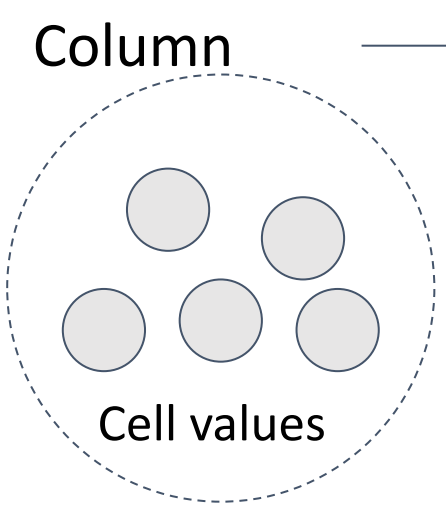


Sherlock - Semantic Data Type Detection



Sherlock - Semantic Data Type Detection

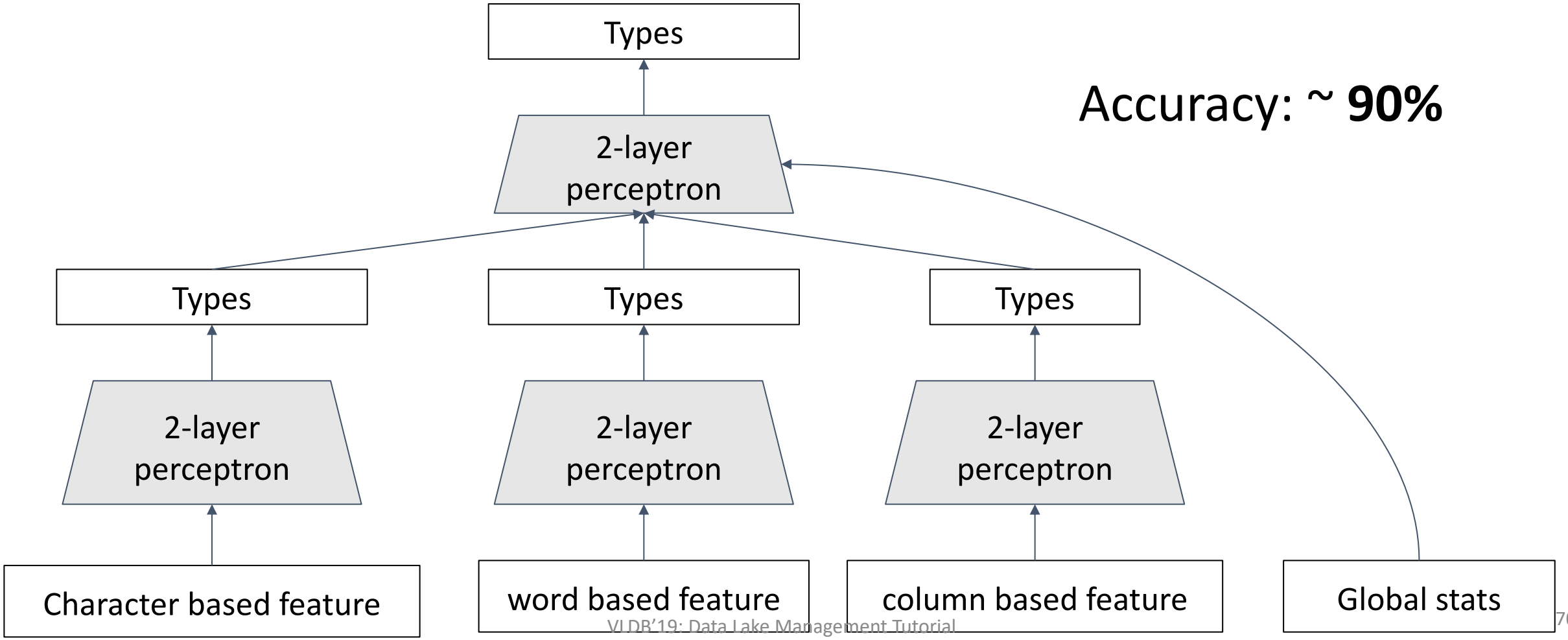




400 dimensions

Distributed bag-of-word paragraph vector

Sherlock - Semantic Data Type Detection



Accuracy: ~ **90%**

Metadata Management

Enterprise Metadata Management

- Schemas (models) and mappings between schemas are first class citizens.
- Operators on models and mappings
  - Compose mappings
  - Invert mapping
- Focus is on a few (possibly complex) models and mappings.
- Data profiling

Ronald Fagin et al. "Schema Mapping Evolution Through Composition and Inversion." Schema Matching and Mapping, 2011.

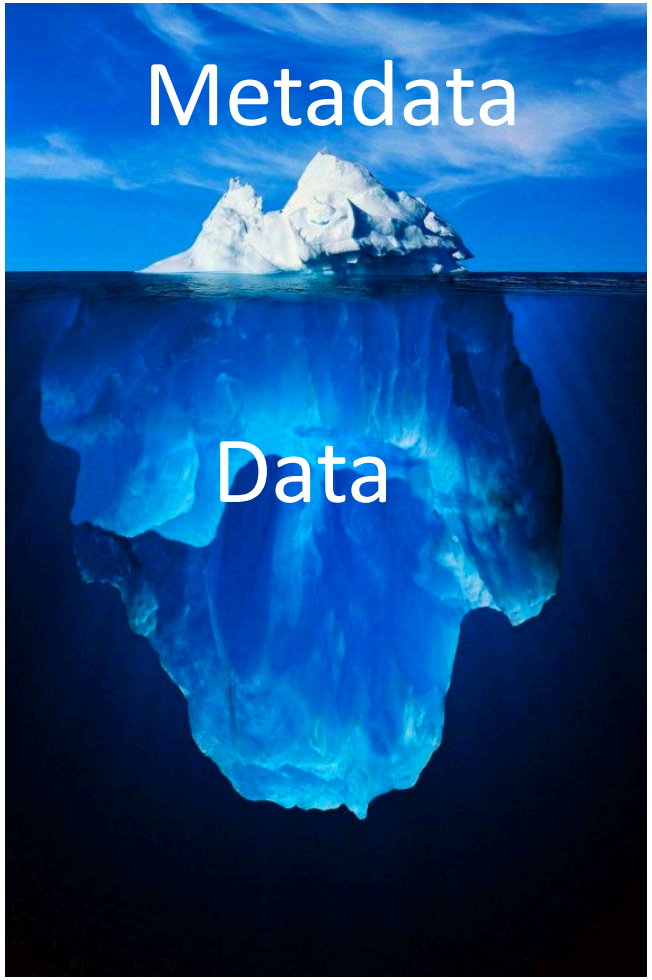
Philip A. Bernstein, Howard Ho, "Model Management and Schema Mappings: Theory and Practice" VLDB 2007.

Yannis Velegrakis et al. "Preserving mapping consistency under schema changes" VLDB Journal, 2004.

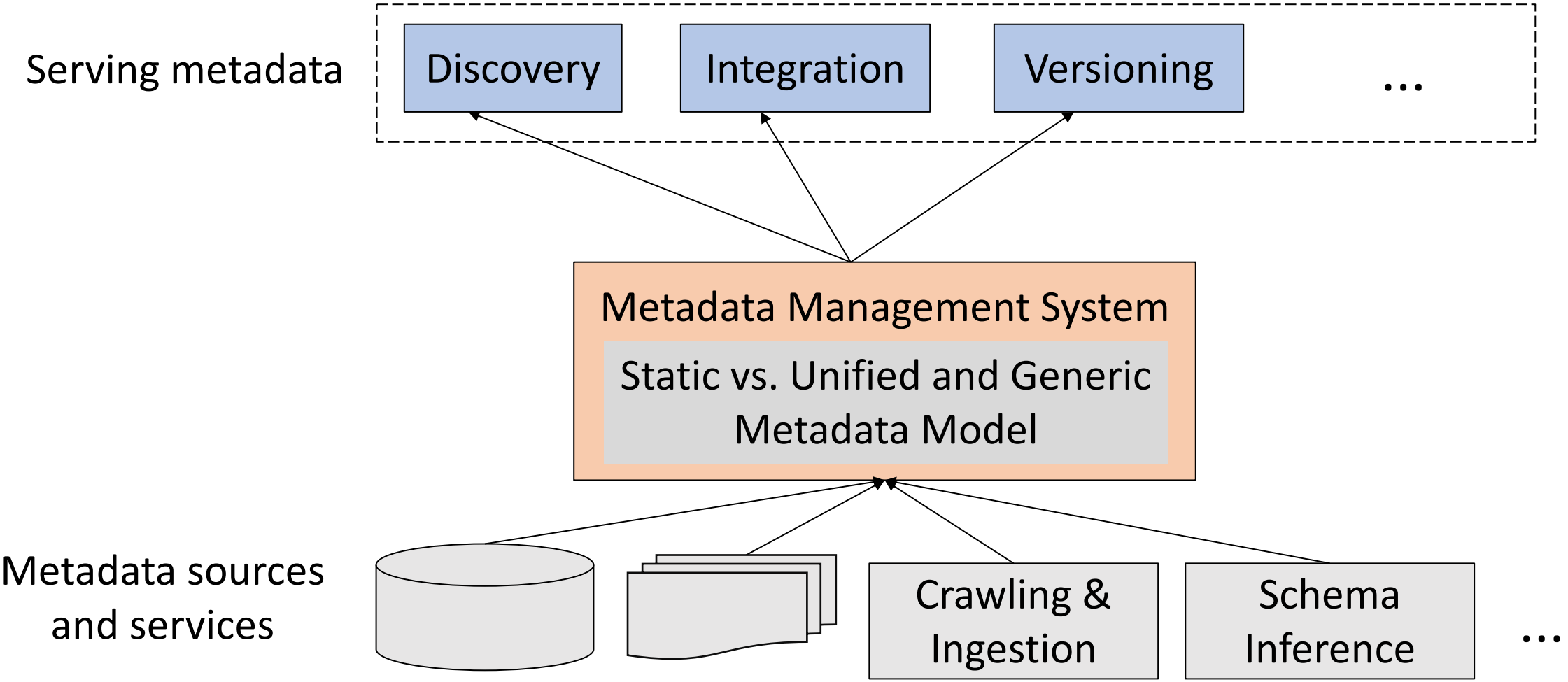
Abedjan, Ziawasch, et al. "Data profiling: a tutorial." SIGMOD, 2017.

Challenges of Data Lake Metadata Management

- The number and size of datasets
  - All-pair problem essential to metadata inference
- Variety of metadata types
  - Complexity of finding a uniform way to represent metadata and access datasets
- Evolving data
  - Capturing temporal changes to content and schema of datasets



Generic Architecture



General Approaches

- Constance
  - enriching data and metadata with semantic information
  - its upstream application is template-based query answering on metadata

Hai, Rihan, et al. "Constance: An intelligent data lake system." SIGMOD, 2016.

- Skulma
  - extracts deeply embedded metadata and contextual metadata
  - allows topic-based discovery

Skluzacek, Tyler J., et al. "Skluma: An extensible metadata extraction pipeline for disorganized data." IEEE e-Science, 2018.

- Labbook
  - Collects metadata about data, users, and queries in collaborative visual analytics environments.

Kandogan, Eser, et al. "Labbook: Metadata-driven social collaborative data analysis." IEEE International Conference on Big Data, 2015.

GOODS: Enterprise-specific Metadata Model

- Googles' internal data lake consists of tens of billions of datasets some with gigabytes and terabytes size.
- An entry for each dataset contains
  - Basic metadata
  - Provenance
  - Schema
  - Content summary
  - …
- Metadata sources: content samples, logs, source code repository, crowdsourcing, and knowledge bases

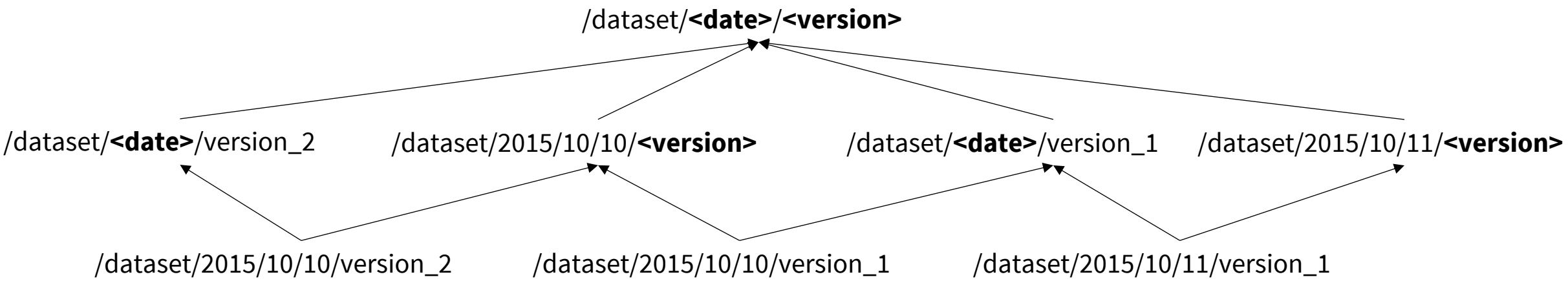
Path/Identifier	Metadata					
	size	…	provenance	…	schema	…
/bigtable/foo/bar	100 G		written by job A		Proto:foo.bar	
/gfs/nlu/foo	10G		written by job B read by job C		Proto:nlu.foo	

Halevy, Alon, et al. "Goods: Organizing google's datasets ” SIGMOD, 2016.

Scaling Metadata Extraction

- Sampling and sketching
- Metadata abstraction dimensions
  - abstraction dimensions: timestamps, data-center names, machine names, versions
  - Metadata aggregation and propagation within clusters

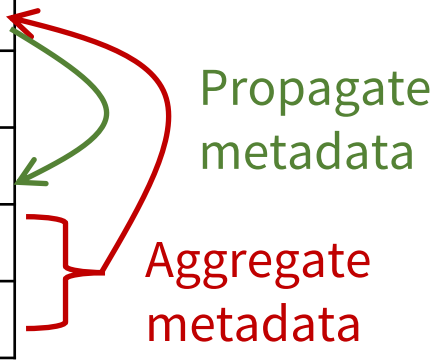
Lattice of date and version dimensions



Scaling Metadata Extraction

- Sampling and sketching
- Metadata abstraction dimensions
  - abstraction dimensions: timestamps, data-center names, machine names, versions
  - Metadata aggregation and propagation within clusters

Path	Other Metadata	Schema
/dataset/<date>/<version>		prod.model
/dataset/2015/10/13/version_3		prod.model
/dataset/2015/10/12/version_2		prod.model
/dataset/2015/10/11/version_1		prod.model



Ground: Generic Metadata Model

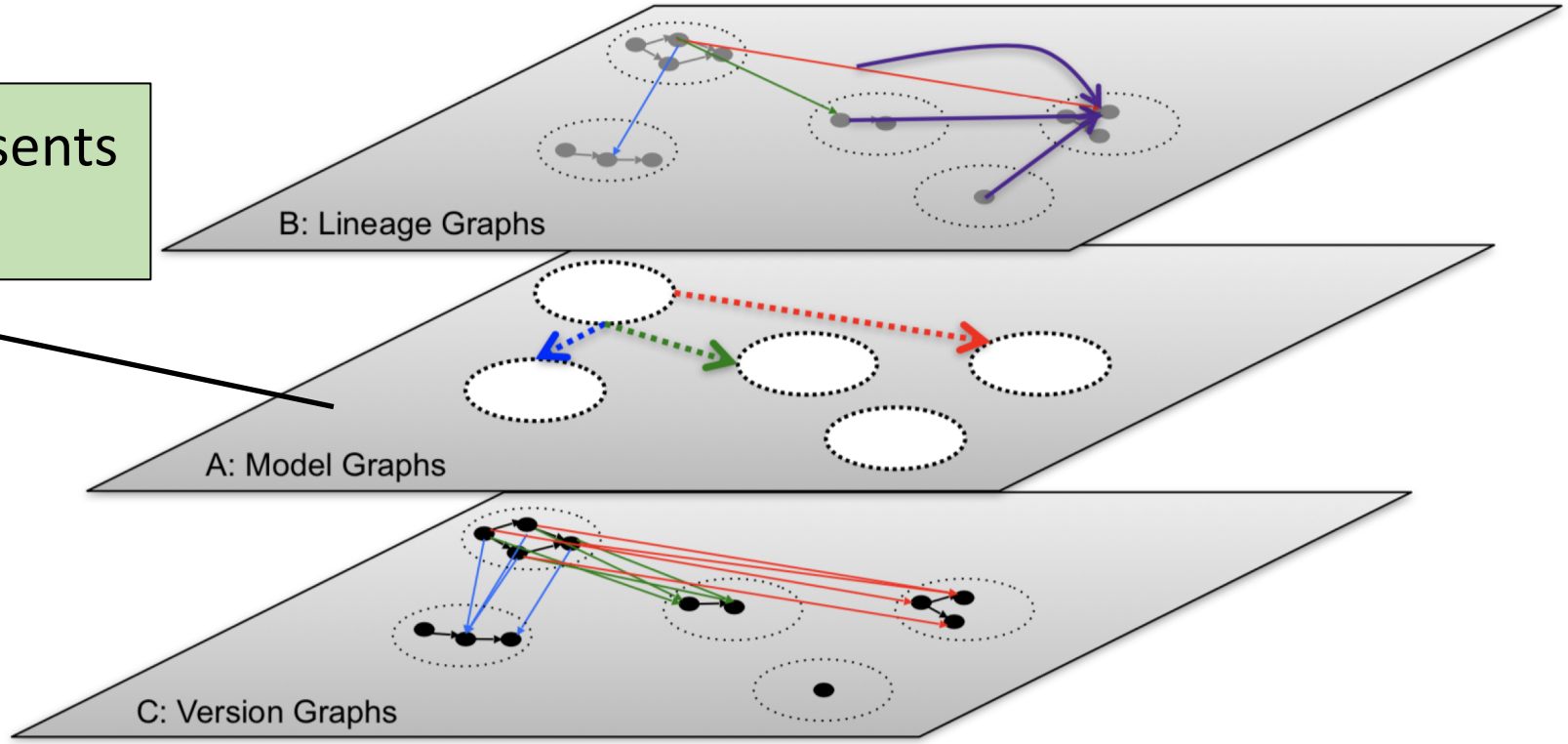
- Open-source data context service
- Ground captures all the information that informs the use of data: data, code, and users
- Generic and expandable metadata model
- Data context characteristics
  - Model-agnostic
  - Immutable

Hellerstein, Joseph, et al. "Ground: A Data Context Service." CIDR. 2017.

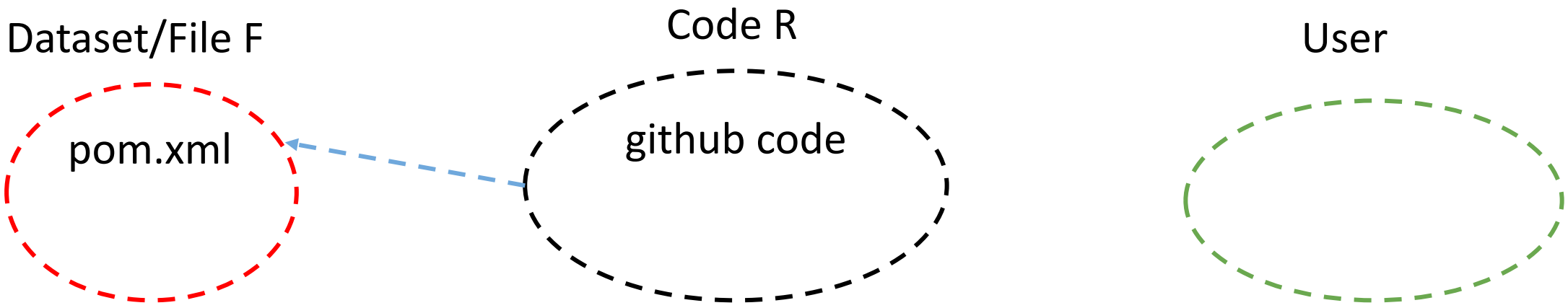
Ground Metamodel

- The ABCs of data context: Application, Behavior, and Change
- Layered graph structure

Model Graph Layer represents application metadata.



Ground: Example



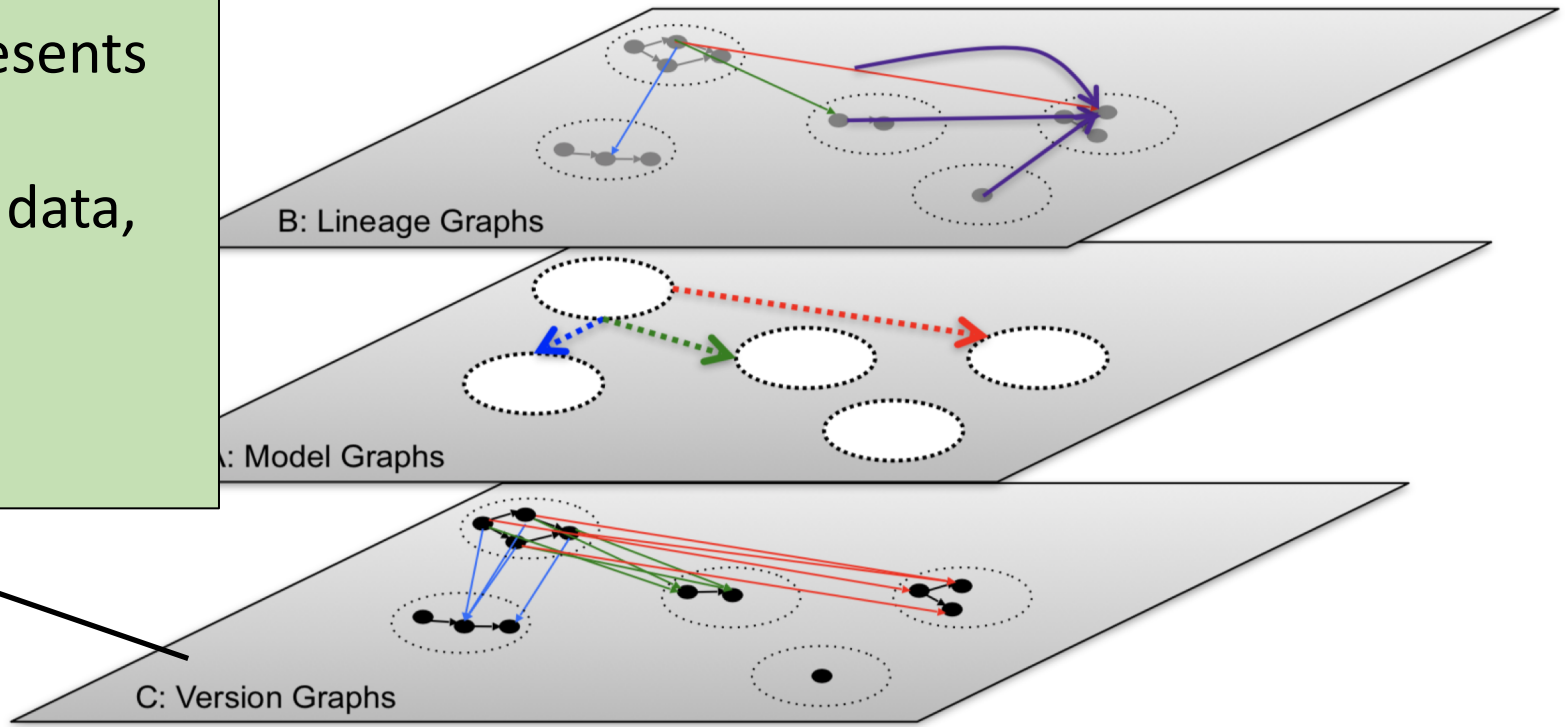
A: Model Graph

Ground Metamodel

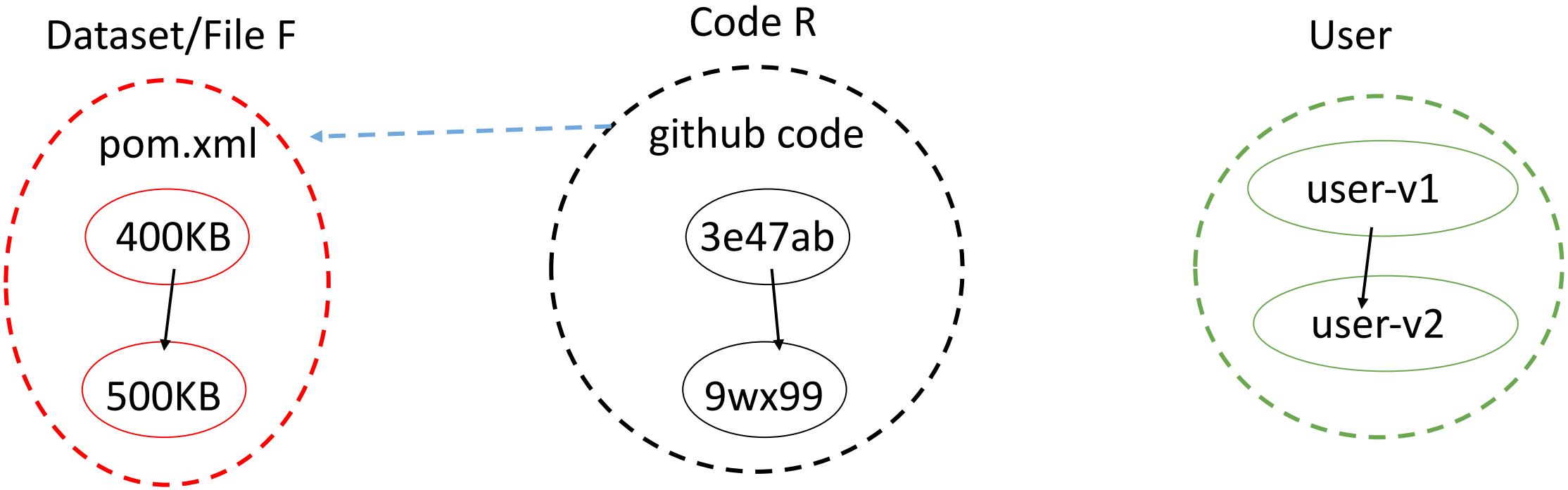
- The ABCs of data context: Application, Behavior, and Change
- Layered graph structure

Version Graph Layer represents changes.

- A node is a version of data, code, or user.
- An edge indicates a successor version.



Ground: Example



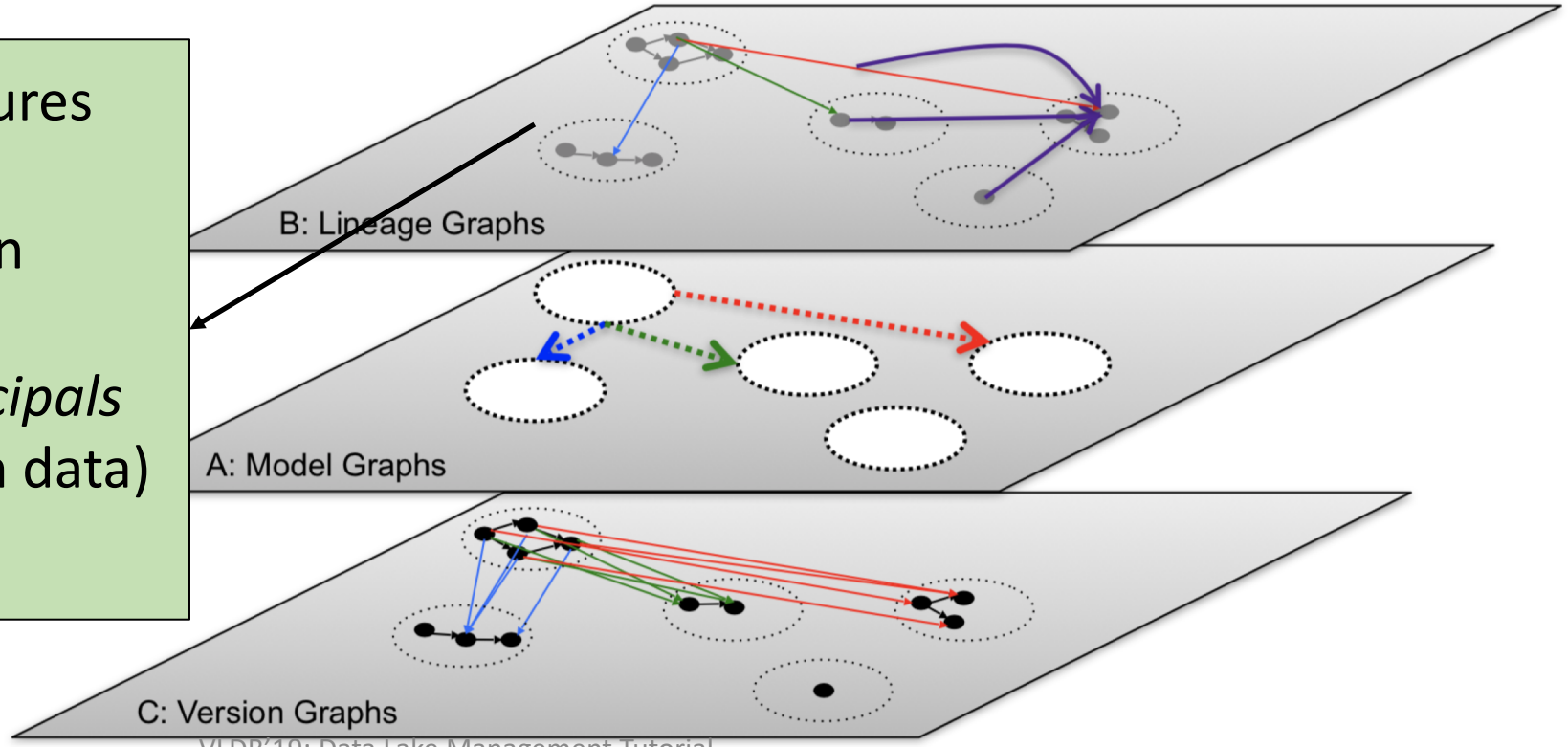
A: Model Graph
C: Version Graph

Ground Metamodel

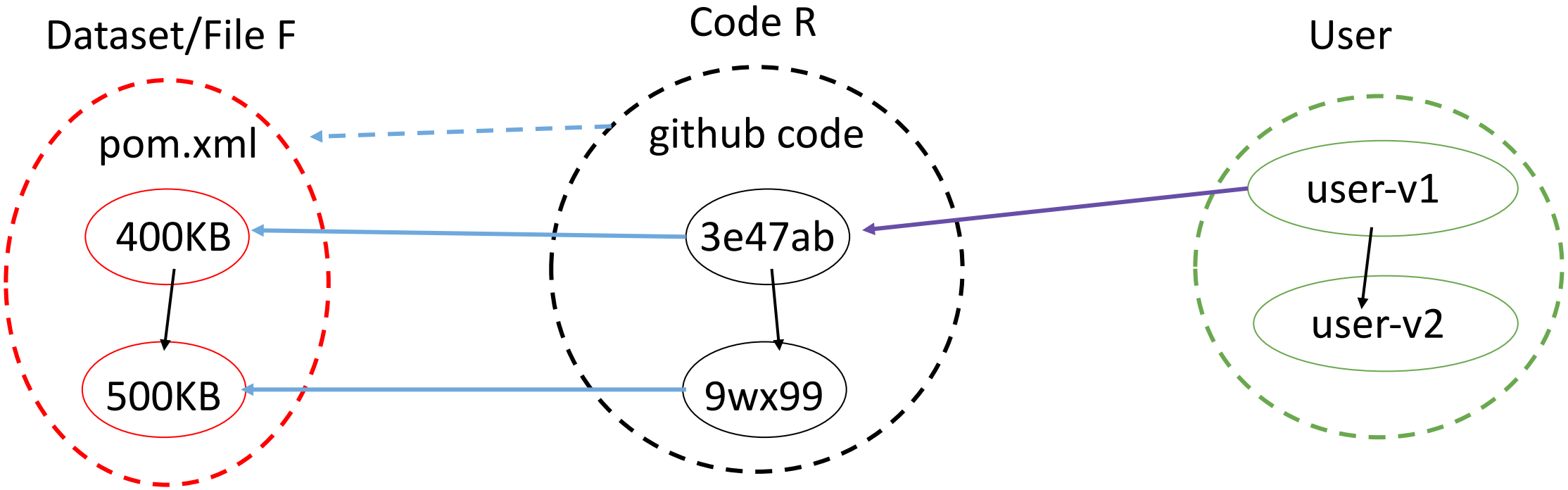
- The ABCs of data context: Application, Behavior, and Change
- Layered graph structure

Lineage Graph Layer captures usage informatin.

- Relationships between versions.
- Lineage includes *principals* (actors that work with data) and *workflows*.



Ground: Example



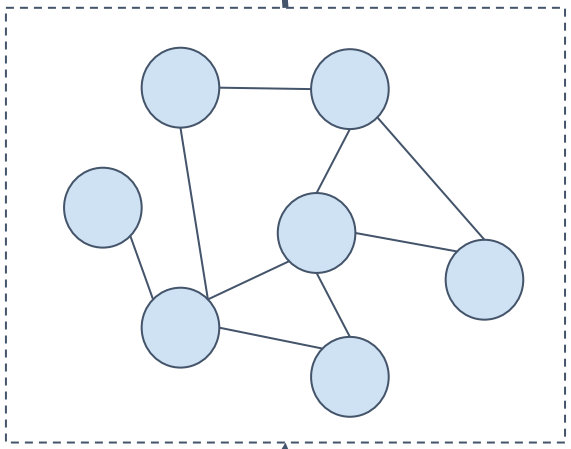
A: Model Graph
C: Version Graph
B: Lineage Graph

AURUM: Inter-datasets Metadata

Serving

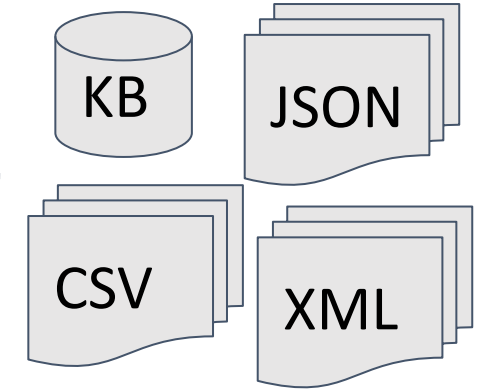
Data Discovery using Source Retrieval Query Language (SRQL)

Metadata Model: Enterprise Knowledge Graph (EKG)



Sources

EKG Builder builds and maintains the EKG.



Fernandez, Raul Castro, et al. "Aurum: A data discovery system." ICDE, 2018.

Fernandez, Raul Castro, et al. "Seeping semantics: Linking datasets using word embeddings for data discovery." ICDE, 2018.

Aurum Metamodel: Enterprise Knowledge Graph

- A node is an attribute or a dataset
- An edge indicates
  - PK/FK relationship
  - Syntactic relationship: value-based similarity (Jaccard) of attributes
  - Semantic relationship: syntactic and semantic similarity of attribute names
- EKG construction is an all-pair problem
  - Leveraging LSH and parallelism
- Source Retrieval Query Language (SRQL) for discovering similar contents

match(columns: drs) = **contentSim**(columns)
AND **attributeNameSim**(columns)
table = "Profits3Term2017"
results = **match**(**columns**(table))

SRQL Execution Engine

- SRQL query answering involves the traversal of a large graph
- Implementing SRQL query answering on RDBMS
  - Leveraging the indexing capabilities to efficiently process discovery queries
- Scales to enterprise data lakes containing hundreds of tables.

How to perform inter-datasets metadata inference and querying?

Cleaning

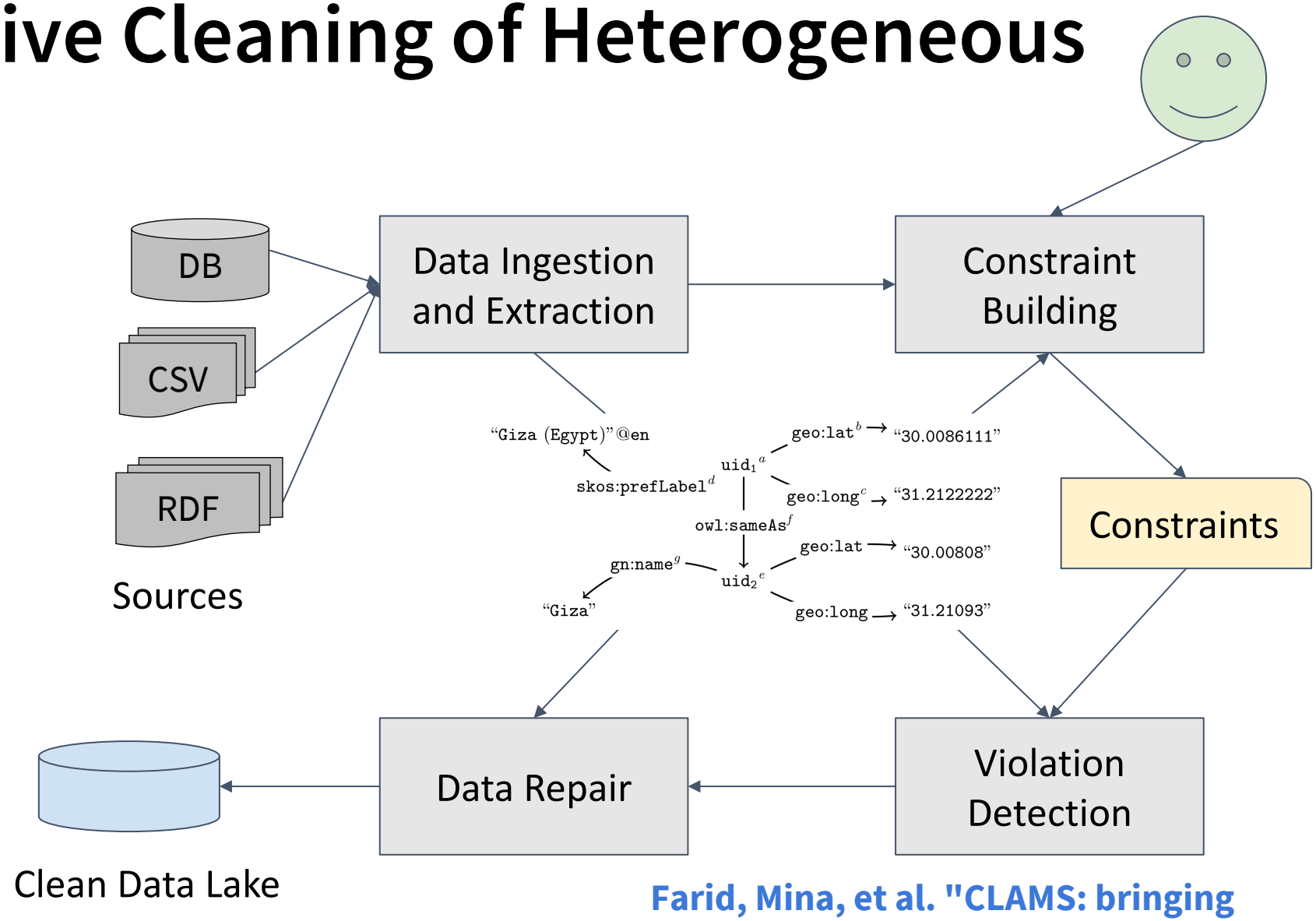
Challenges of Lake Data Cleaning

- Load-first-schema-later paradigm
  - Some of cleaning approaches require integrity constraints
  - The lack of a global schema and limited integrity constraints
  - Cleaning often follows ingestion.
- Heterogeneous data
  - Existing algorithms focus on relational data.
  - No support for integrity constraints and data quality checks for all data formats
- Collective cleaning
  - Statistical approaches typically consider one table.
  - Existing solutions for relational data do not consider collective cleaning of a large number of datasets

Prokoshyna, Nataliya, et al. "Combining quantitative and logical data cleaning." PVLDB, 2015.

CLAMS: Collective Cleaning of Heterogeneous Lakes

1. Transforming data sources into RDF triples
2. Loading triples into a unified model
3. Human-in-the-loop constraint detection and data repair



Farid, Mina, et al. "CLAMS: bringing quality to Data Lakes." SIGMOD, 2016.

Integrity Constraints on Heterogeneous Data

A Conditional Denial Constraint (CDC) is a Denial Constraint over a relational view of a non-relational

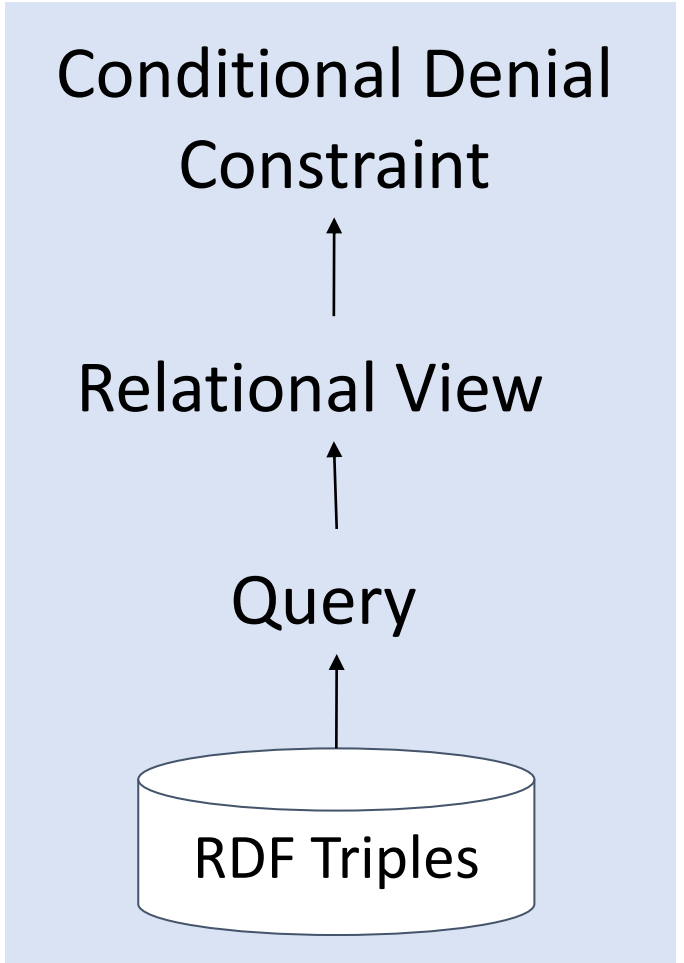
- CDC c
  - Q is a
    mod
  - φ is a
  - Data

with data

"Two identical locations must have the same latitude coordinates."

Q = SELECT ?loc1 ?loc2 ?lat1 ?lat2
WHERE { ?loc1 owl:sameAs ?loc2 .
?loc1 geo:lat ?lat1 .
?loc2 geo:lat ?lat2 }

$\varphi = \{\forall t \in Q(D), \neg(t.la1 \neq t.lat2)\}$



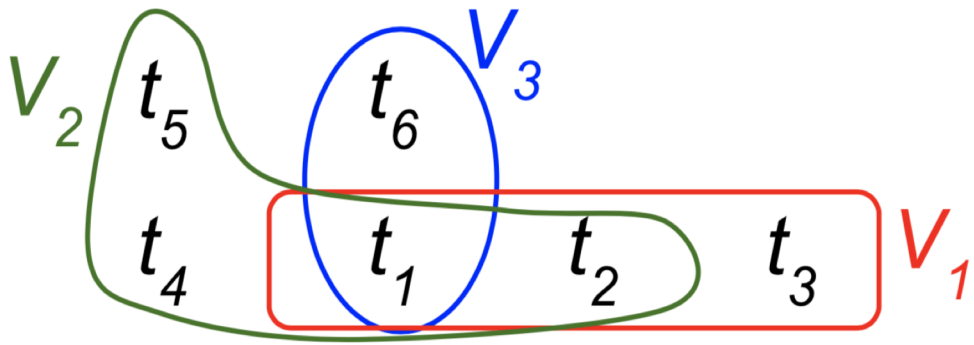
Arenas, Marcel
Chomicki. "Co
databases." PODS. Vol. 99. 1999.

Chu, Xu, Ihab F. Ilyas, and Paolo Papotti. "Discovering denial constraints." PVLDB, 2013.

Violation Detection and Repair

- CDC discovery by limiting the query to conjunctive queries
- A violation of a CDC is a minimal set of triples that cannot co-exist.
- Human-in-the-loop Repair
  - Ranking erroneous tuples by the number of violations
  - Providing the lineage of violating data
  - Repairing by deletion

Violation Hypergraph



High throughput and scalable collective cleaning in lakes?

Deequ: Scalable Single-Dataset Verification

- An open-source library for automating the verification of data quality of a single dataset at scale.
- Declarative and flexible data quality rules
  - combining quality constraints and user-defined verification code
  - leveraging external data and custom code
- Scaling verification to large datasets
- Incremental verification on growing datasets

Schelter, Sebastian, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. "Automating large-scale data quality verification." PVLDB, 2018.

Schelter, Sebastian, Felix Biessmann, Dustin Lange, Tammo Rukat, Phillipp Schmidt, Stephan Seufert, Pierre Brunelle, and Andrey Taptunov. "Unit Testing Data with Deequ." SIGMOD, 2019.

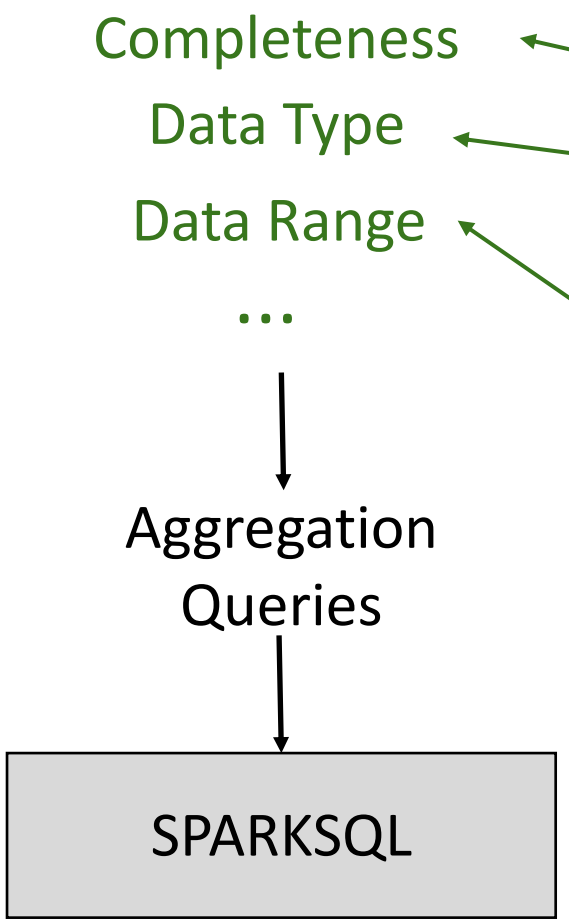
Unit Tests for Data

```
CREATE EXTERNAL TABLE amazon_reviews_parquet(
  marketplace string,
  customer_id string,
  review_id string,
  product_id string,
  product_parent string,
  product_title string,
  star_rating int,
  helpful_votes int,
  total_votes int,
  vine string,
  verified_purchase string,
  review_headline string,
  review_body string,
  review_date bigint,
  year int)
PARTITIONED BY (product_category string)
ROW FORMAT SERDE ...
```

Declarative definition of data quality constraints

```
Check(Level.Error)
  .isComplete("customer_id")
  .hasDataType("customer_id", Integral)
  .isComplete("product_category")
  .isContainedIn("product_category",
    Array("Beauty", "Shoes", "Jewelry"))
  .isContainedIn("vine", Array("N"), _ >= 0.99)
  .isNonNegative("total_votes")
...
```

Scalable Unit Test Execution

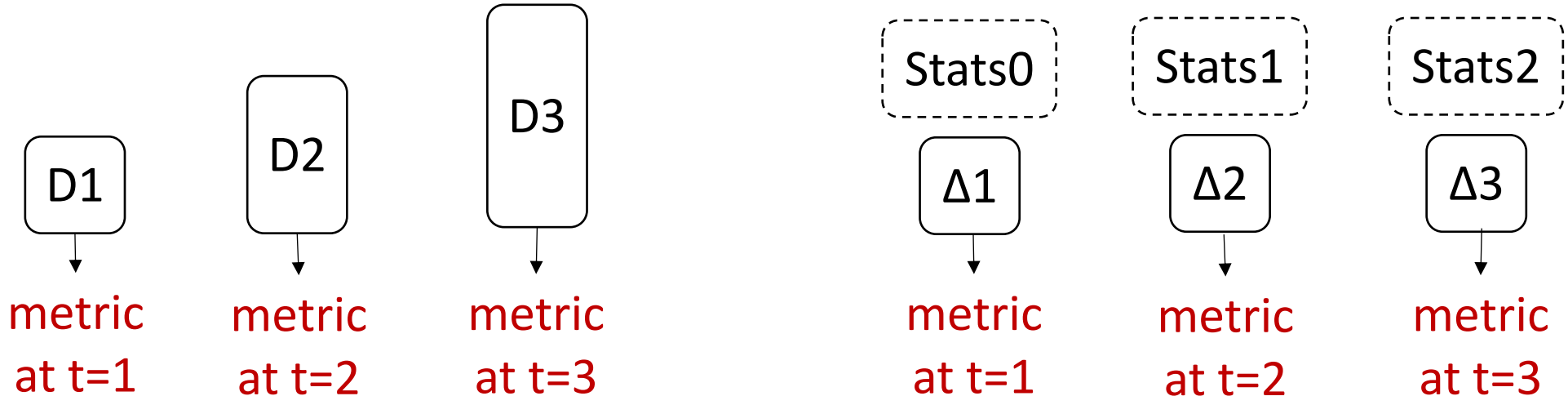



```
Check(Level.Error)
  .isComplete("customer_id")
  .hasDataType("customer_id", Integral)
  .isComplete("product_category")
  .isContainedIn("product_category",
    Array("Beauty", "Shoes", "Jewelry"))
  .isContainedIn("vine", Array("N"), _ >= 0.99)
  .isNonNegative("total_votes")
...
```

work-load sharing among metrics

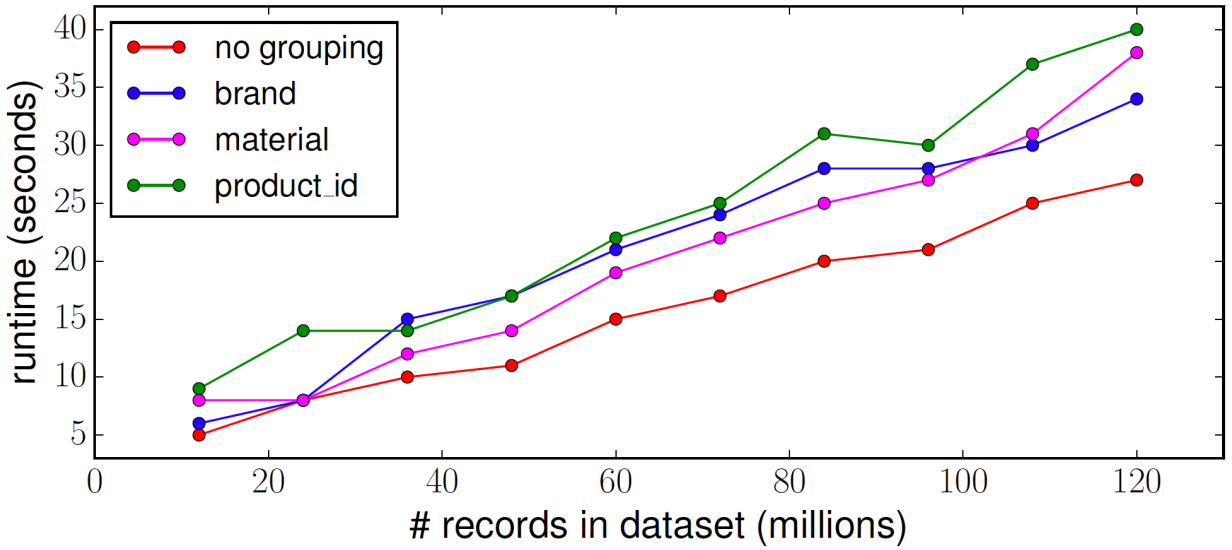
Incremental Validation of Growing Data

- Updating metrics without having to access the previous data.



Scalable Verification of Growing Datasets

- Linearly increasing runtime for quality verification on growing datasets.



Data growth is one aspect of change. What happens when data records are modified or deleted?

Error Diagnosis of Data Lake Staging

- Ingestion and extraction methods have inherent imperfections.
- Data X-Ray diagnoses the underlying conditions which cause erroneous data.
  - Profiling data
  - Representing data points in a hierarchy of features
  - Finding the features that best represent erroneous data points

Lourenço, Raoni, et al. "Debugging Machine Learning Pipelines." DEEM, 2019.

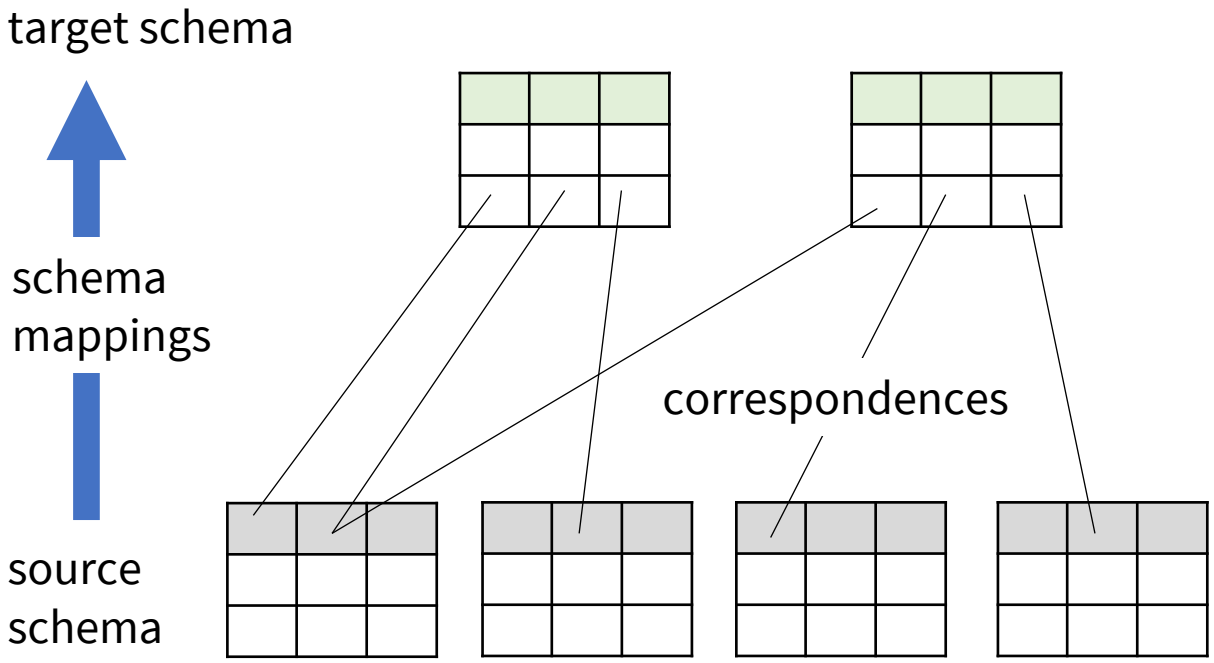
Wang, Xiaolan, et al. "Data x-ray: A diagnostic tool for data errors." SIGMOD, 2015.

Integration

Traditional Data Integration

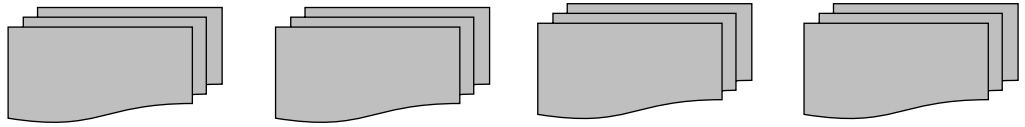
Fagin, Ronald, et al. "Data exchange: semantics and query answering." Theoretical Computer Science 336.1 (2005): 89-124.

Maurizio Lenzerini: Data Integration: A Theoretical Perspective. PODS 2002: 233-246.



Ten Cate, Balder, et al. "Schema mappings and data examples." EDBT, 2013.

PCMember(name, institution, conference, year)



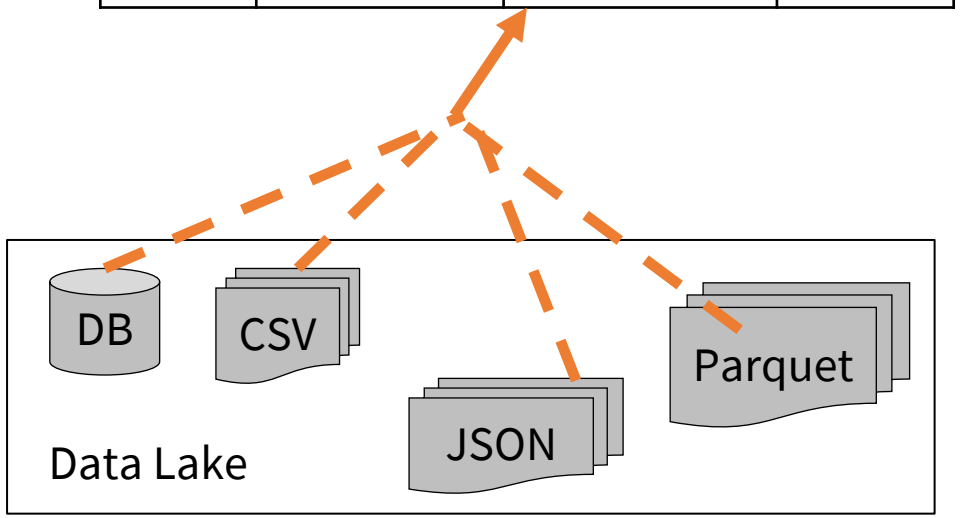
Data Lake

PCMember(name, institution, 'VLDB', 2018) :-
VLDB18 (name, institution),
PCMember(name, institution, 'VLDB', 2019) :-
VLDB19 (name, institution)

On-demand Integration

- Input is a description of data (e.g. schema).
- Output is an assembled dataset from a lake that satisfies the schema.

name	institution	conference	year
?	?	?	?
?	?	?	?



Challenges of Data Lake Integration

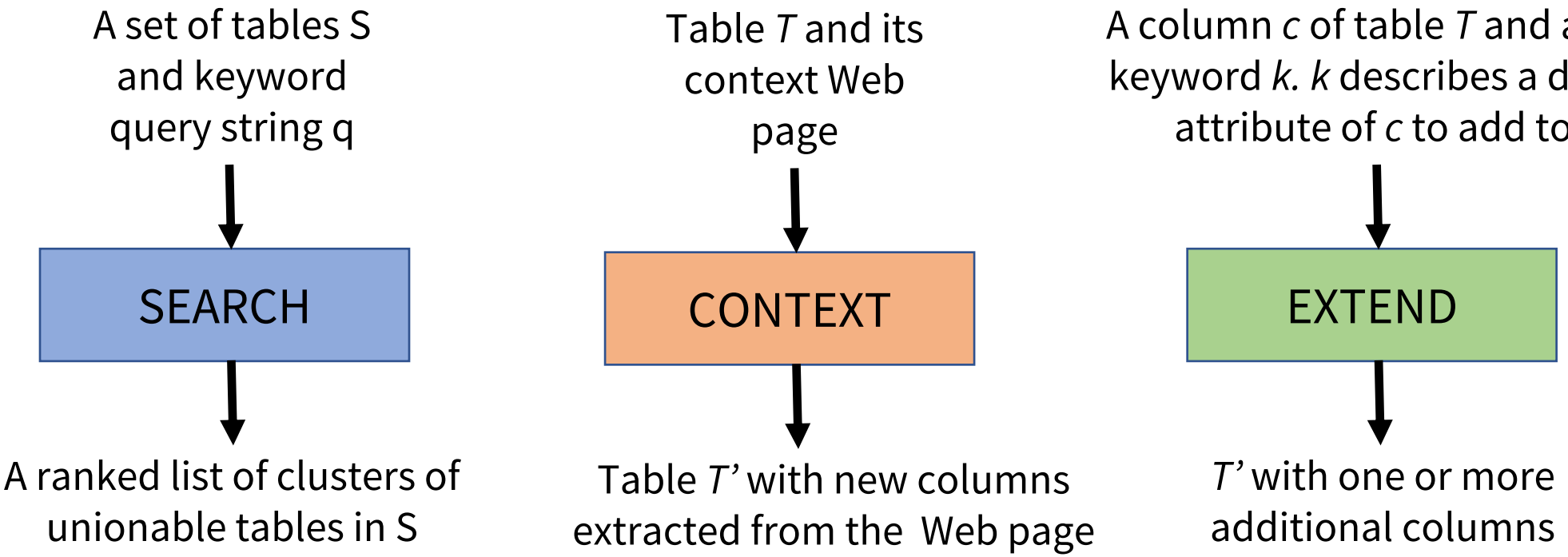
- Specifying what is to be integrated
  - Using a simple keyword or topic
  - Using a table or schema specification (as is done in data federation/exchange)
  - Using a query table with both schema and data
  - Using a full schema with multiple datasets
- Efficient discovery of relevant data sources
- The schema of lake is incomplete
  - Column matching and data matching (entity resolution) is not straightforward.
- Uncertainty in integration

Altowim, Yasser, and Sharad Mehrotra. "Parallel progressive approach to entity resolution using mapreduce." 2017 IEEE 33rd International Conference on Data Engineering (ICDE). IEEE, 2017.

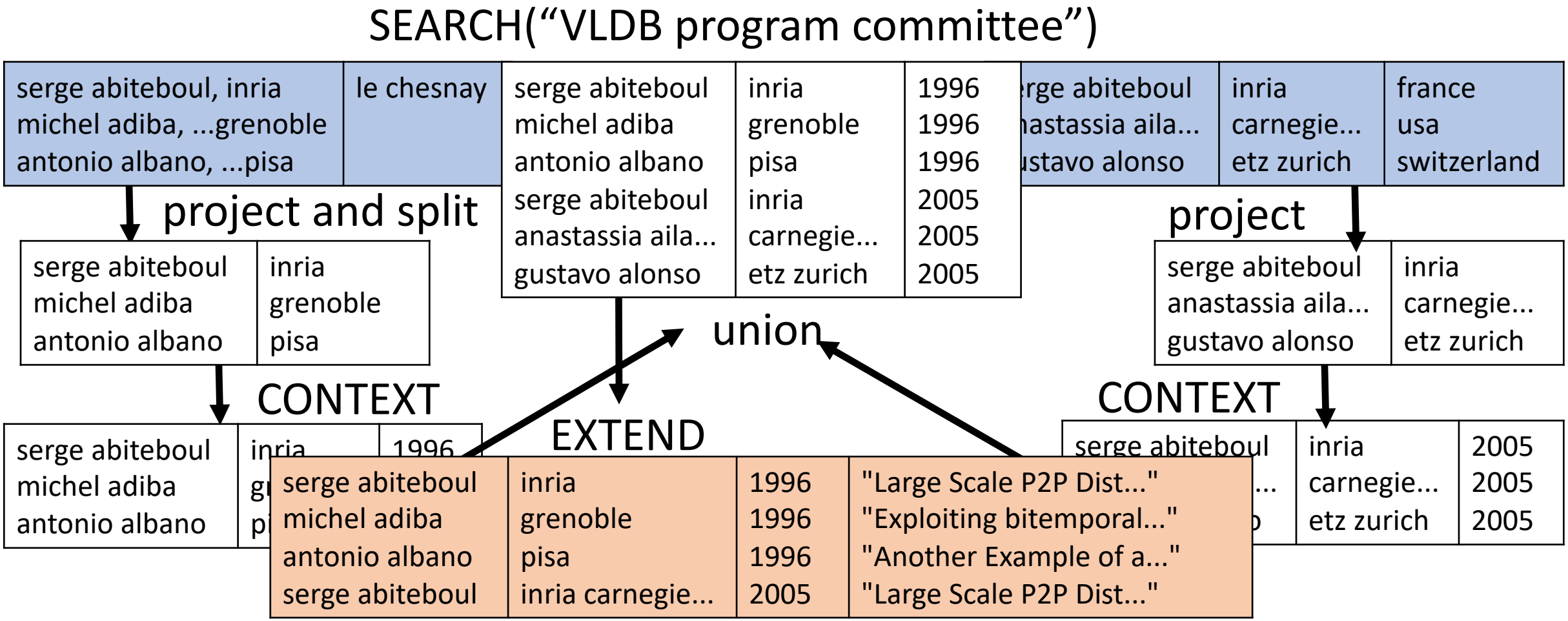
Octopus

Cafarella, Michael J., et al. "Data integration for the relational web." PVLDB (2009): 1090-1101.

A set of integration operators for the corpus of Web tables that allows users to build datasets in a human-in-the-loop manner.

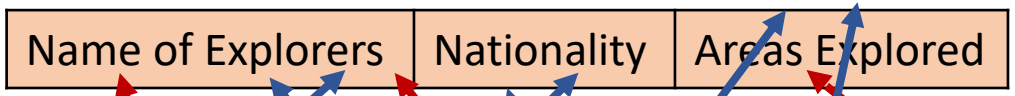


Octopus: Query by a Keyword and Column



The cardinality of attributes in a lake can be large and skewed. Recent work address this issue (coming up in discovery section).

WWT: Query by Keywords



Column Mapper: mapping each table column to a query column (or none).

Finding candidate Web Tables

Consolidator: finding the similarity of the content of columns across tables

Answer Table

Number of Explorers	Nationality	Areas Explored
Vasco da Gama	Portuguese	Sea route to India
Abel Tasman	Dutch	Oceania
Christopher Columbus	British	Caribbean
…		

Pimplikar, Rakesh, and Sunita Sarawagi. "Answering table queries on the web using column keywords." PVLDB, 2012.

WebTable 1: List of explorers

Name	Nationality	Main areas explored
Abel Tasman	Dutch	Oceania
Vasco da Gama	Portuguese	Sea route to India
Alexander Mackenzie	British	Canada
…		

WebTable 2: Explorations in history

Exploration	Who
Sea route to India	Vasco da Gama
Caribbean	Christopher Columbus
Oceania	Abel Tasman
…	

WebTable 3: Forest reserves under Act …

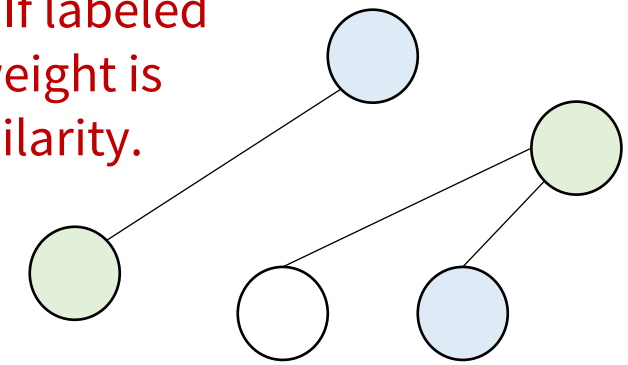
Forest reserves ID	Name	Area
7	Shakespeare Hills	2236
9	Plains Creek	880
13	Welcome Swamp	168
…		

Web Tables

WWT: Collective Matching and Mapping

- Collective Mapping is formulated a graphical model.

Edge potential: If labeled the same, the weight is the content similarity.



Webtable column

Node potential: query column labels + {no match, table not relevant}

- Find query keyword labels for columns of corpus tables such that the following is maximized:

polynomial solution for finding optimal column mapping

$$\sum_{web\ columns} node\ potential + \sum_{web\ columns} \sum_{web\ columns} edge\ potential$$

Representing Query Dataset

- Sample-driven mapping: MWEAVER
  - Query is a table schema and a few tuples.
  - Solution for tens of sources and hundreds of columns
- Multi-resolution mapping: PRISM
  - Query is the same as sample-driven and is relaxed to allow defining constraints at various resultion.
- Infogather:
  - Query is a table and the description of a column.

Qian, Li, et al. "Sample-driven schema mapping." SIGMOD, 2012.
Shen, Yanyan, et al. "Discovering queries based on example tuples." SIGMOD, 2014.

name	director	producer
Avatar	James…	Lightstorm Co.
Harry Potter	David Y…	
Christopher ..	Tim Bur…	
…		

Jin, Zhongjun, et al. "Demonstration of a Multiresolution Schema Mapping System." CIDR, 2019.

lake	location	area
Nevada || California	Lake Tahoe	DataType==‘decimal” ’ AND MinValue>=‘0”

Yakout, Mohamed, et al. "Infogather: entity augmentation and attribute discovery by holistic matching with web tables." SIGMOD, 2012.

model	brand
S80	
A10	
GX-1S	

S80	Nikon
A10	Canon
GX-1S	

Discovery

Data Discovery

- The objective is to find relevant datasets for analytic tasks
- Two ways to search:
  - Search by keywords
  - Search by dataset
    - Joinable table search
    - Unionable table search
    - Attribute search

Brickley, Dan, et al. "Google Dataset Search: Building a search engine for datasets in an open Web ecosystem." WWW, 2019.

Covered here

Nargesian, Fatemeh, et al. "Table union search on open data." *PVLDB* 11.7 (2018): 813-825.

Yakout, Mohamed, et al. "Infogather: entity augmentation and attribute discovery by holistic matching with web tables." *SIGMOD*, 2012.

LSH Ensemble

Zhu, Erkang, et al. "LSH ensemble: internet-scale domain search." *PVLDB* (2016): 1185-1196.

- **Objective**: given a query column Q and a threshold parameter, t^*, find any column, X, in the data lake such that its containment, $\frac{|Q \cap X|}{|Q|} \geq t^*$
- Containment is more suitable for joinable table search than Jaccard ($\frac{|Q \cap X|}{|Q \cup X|}$)

Assume $|Q \cap X|$ constant, when $|X| \nearrow$, $\frac{|Q \cap X|}{|Q \cup X|} \to 0$, but $\frac{|X \cap Q|}{|Q|}$ is unchanged

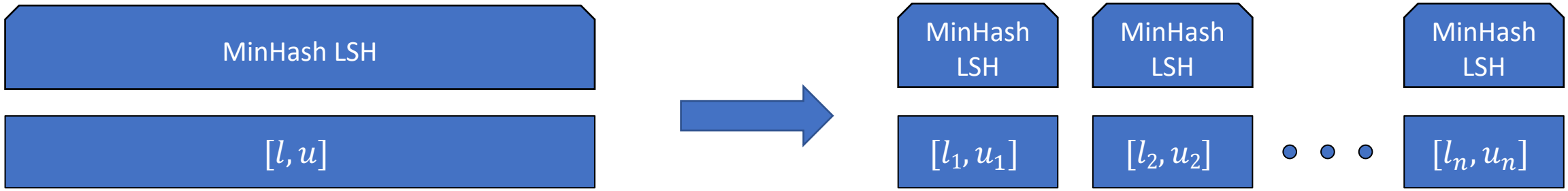
- MinHash LSH is scalable for data lakes but only supports Jaccard

We want to use containment because Jaccard has recall issue when $|X| > |Q|$

- **Our Contribution**: utilize the MinHash LSH for scalable containment search

Reduce False Positives - Partitions

- Partition the sets into n disjoint partitions with increasing set sizes:



- Reduce false positive at the rate of $\frac{1}{(u-u_i)^2}$ by using u_i instead of u for each partition!

- An optimal partitioning for data lakes that minimize overall false positive is approximately equi-depth.

LSH Ensemble

- Novel way to use MinHash LSH for containment search
- Optimal partitioning to minimize overall false positives
- Parallelize query over partitions
- Scale to massive number of columns
  - Tested on WDC English Web Tables > 50 million tables
- Open source: https://github.com/ekzhu/datasketch

Lazo

Fernandez, Raul Castro, et al. "Lazo: A Cardinality-Based Method for Coupled Estimation of Jaccard Similarity and Containment." ICDE, 2019

- An improvement to MinHash is One-Permutation Hashing (OPH)
  - More computationally efficient
- Lazo is an LSH index that uses OPH, supports both containment and Jaccard, and returns similarity scores
- **Contribution**: computational efficiency in index building, better accuracy at high thresholds

Li, Ping, et al. "One permutation hashing." *NeurIPS*, 2012.

JOSIE

Zhu, Erkang, et al. "JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes." *SIGMOD*, 2019.

- **Objective**: exact top-k search for finding joinable tables
- Given a query column Q, find top-k columns $(X_1 \ldots X_k)$ from the data lakes with the highest intersection size $|Q \cap X|$
- JOISE is a drop-in component utilizing inverted index for finding joinable tables

Inverted Index

- Posting list union: read all posting lists of a query column Q, then rank column IDs by their occurrences
  - Issue: $|Q|$ can be very large – think typical data table, thousands of rows

- Prefix filtering: use a top-k heap; eagerly read columns (as sets) to increase the minimum $|Q \cap X|$ in the heap, reducing the need for reading too many posting lists
  - Issue: column $|X|$ can be huge as well

JOSIE: A Hybrid Approach

- Read posting lists in batch, and use the occurrences of column IDs to estimate their intersection size with Q
- Estimate which operation has higher net progress in determining the final top-k:
  - Read the next batch of posting lists (a chunked posting list union)
  - Read the column with the highest estimated intersection size (a prioritized prefix filter)
- Adaptive to different data lake characteristics without tuning. E.g.: long posting lists and small columns, short posting lists and large columns

JOSIE

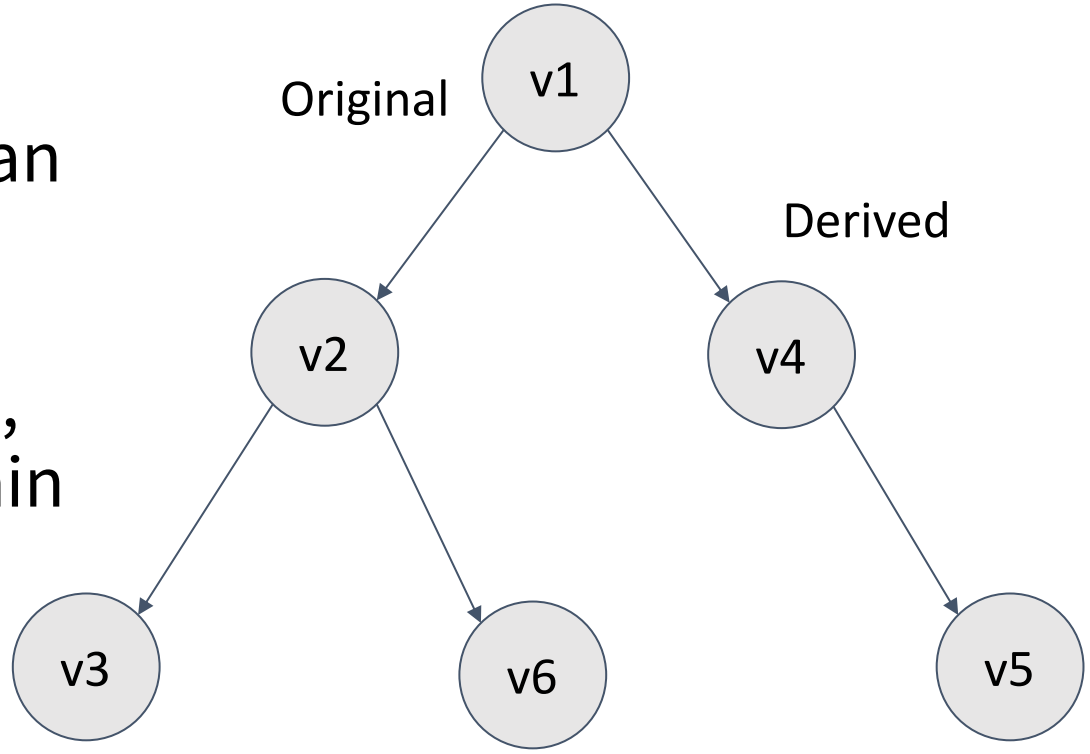
- Exact computation of top-k containment queries
- Scales to > 50 Million tables
- Outperforms previous approximate approaches for top-k queries of up to several thousand values on real data

Versioning

The Data Set Versioning Problem

Goal: allow access (e.g., query) to different versions of data sets, branching -- more than a linear chain of versions

Storage–recreation trade-off well explored, but many other aspects of versioning remain underexplored



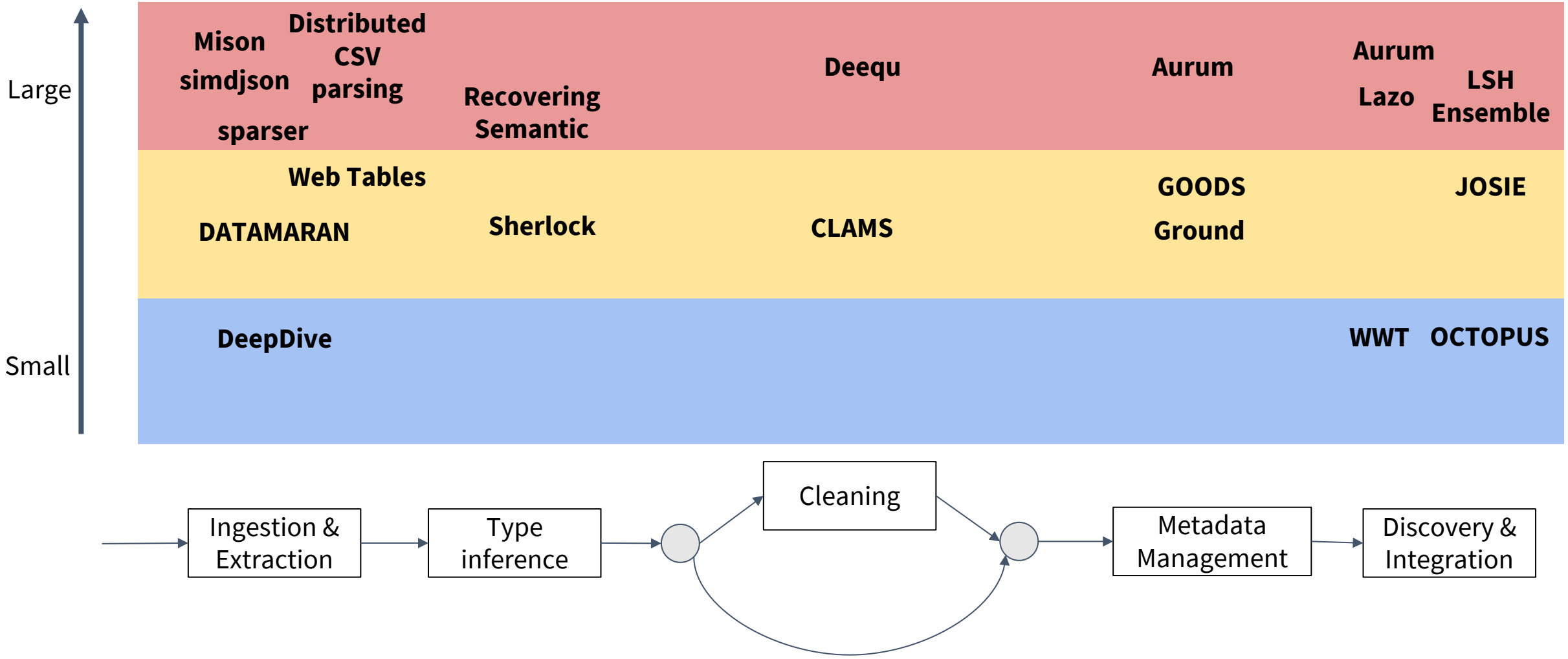
Bhattacherjee, Souvik, et al. "Principles of dataset versioning: Exploring the recreation/storage tradeoff." *PVLDB* 8.12 (2015): 1346-1357.

Bhardwaj, Anant, et al. "Collaborative data analytics with DataHub." *PVLDB* 8.12 (2015): 1916-1919.

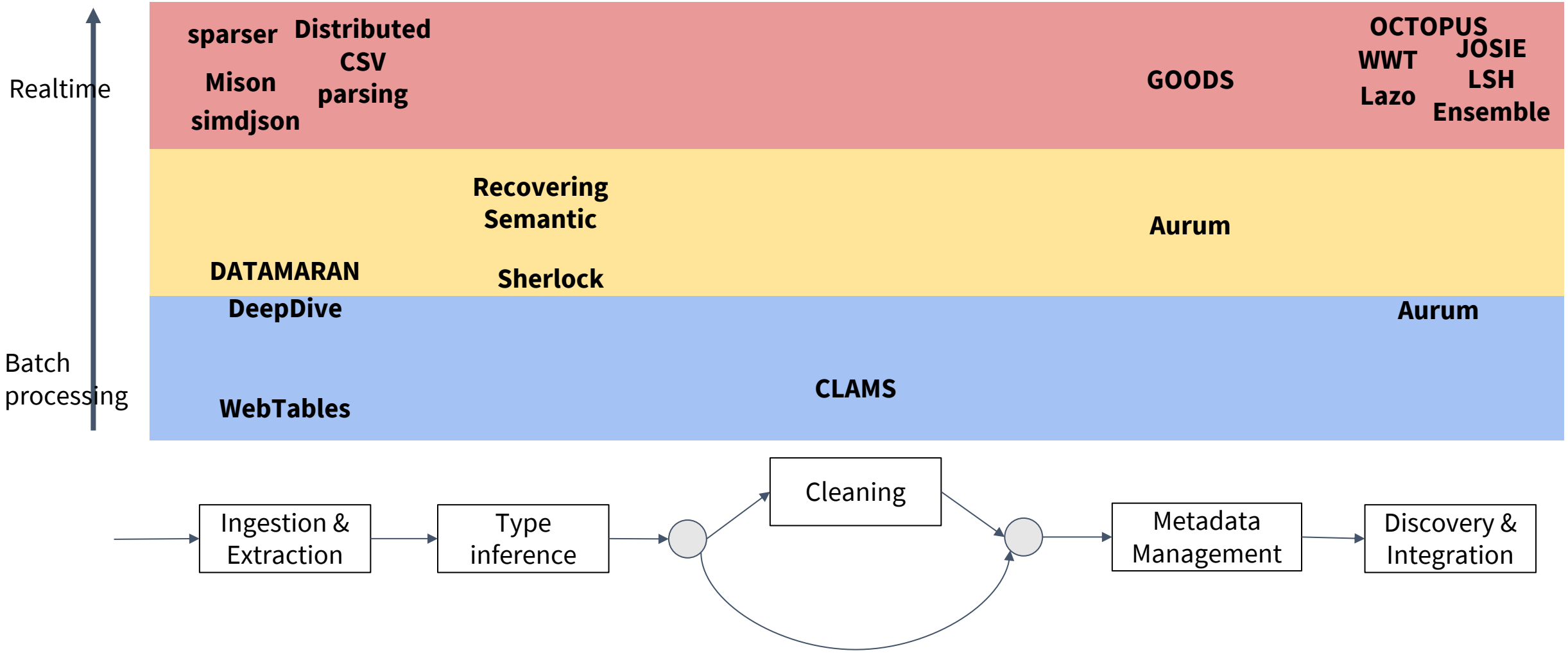
Huang, Silu, et al. "Orpheus DB: bolt-on versioning for relational databases." *PVLDB* 10.10 (2017): 1130-1141.

Conclusion

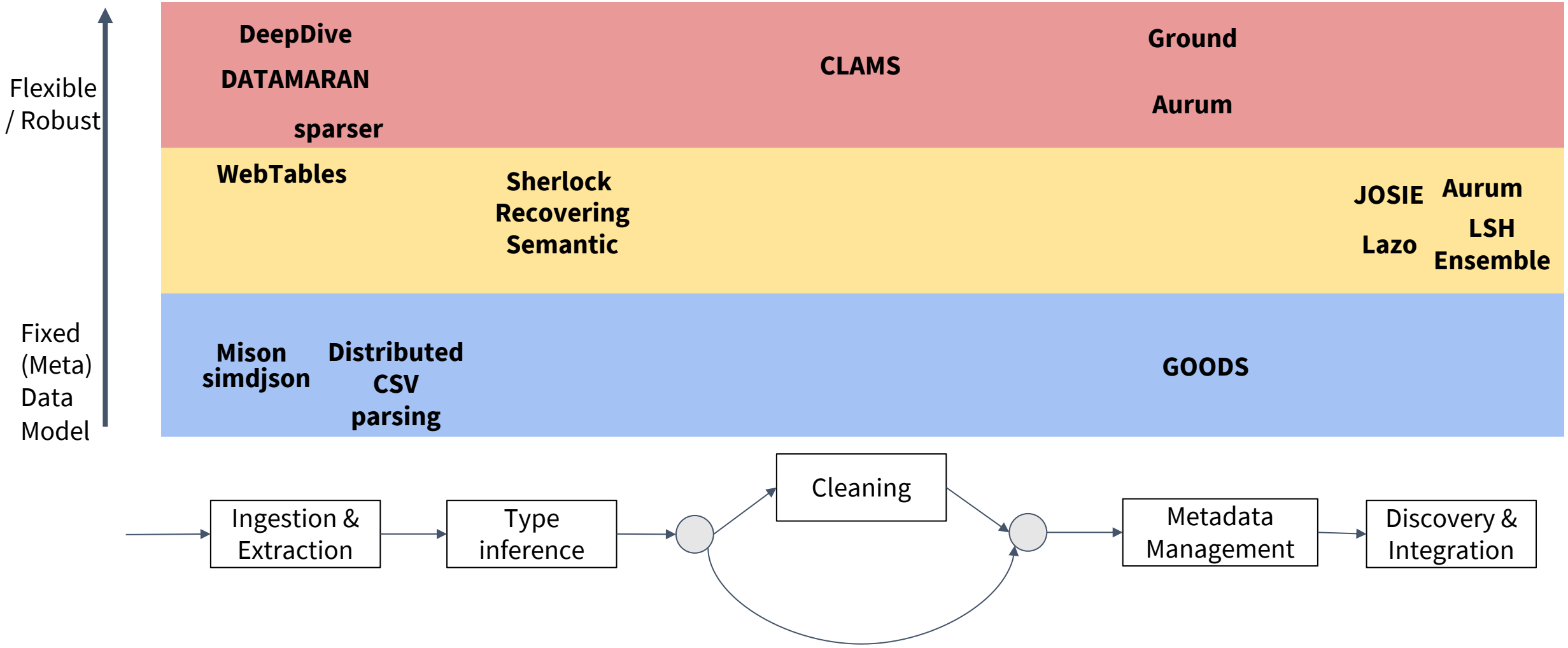
Dataset Size (Scalability over Single Dataset)



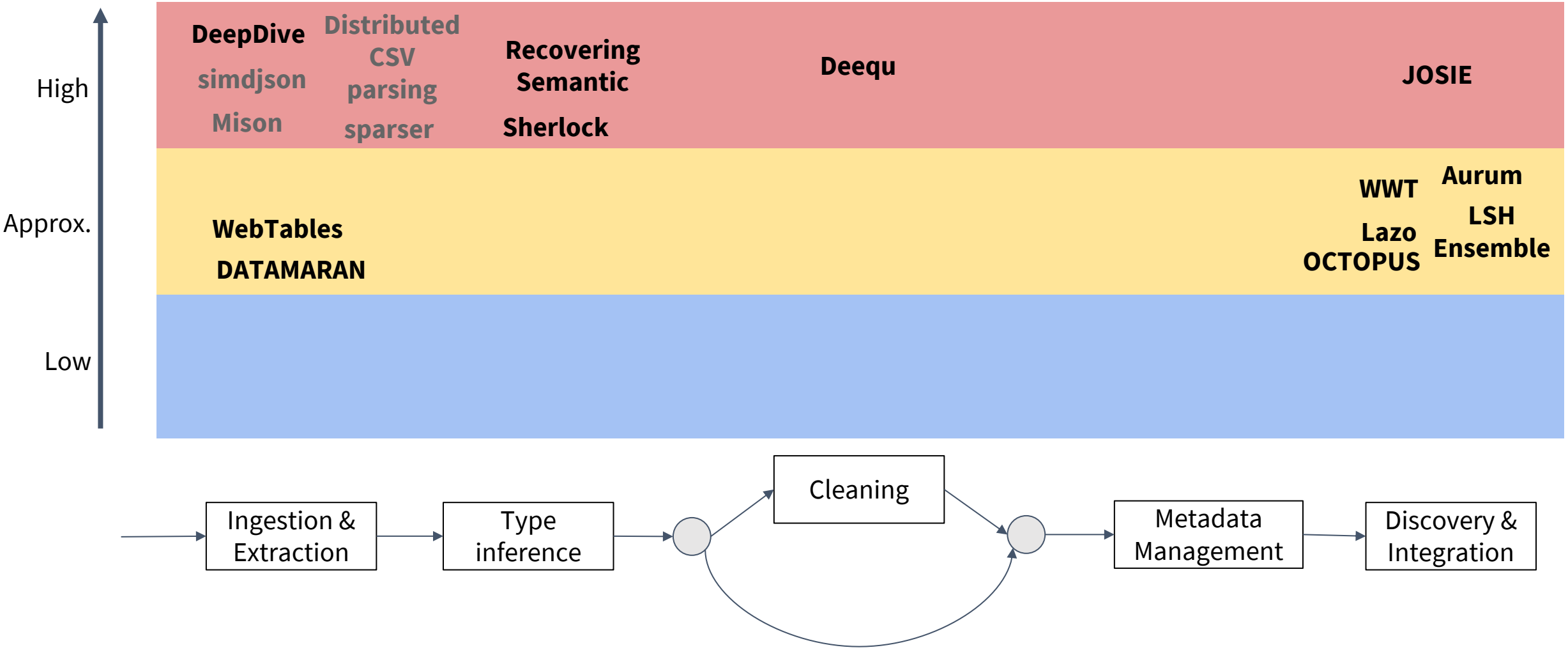
Throughput of all Processing over Lake



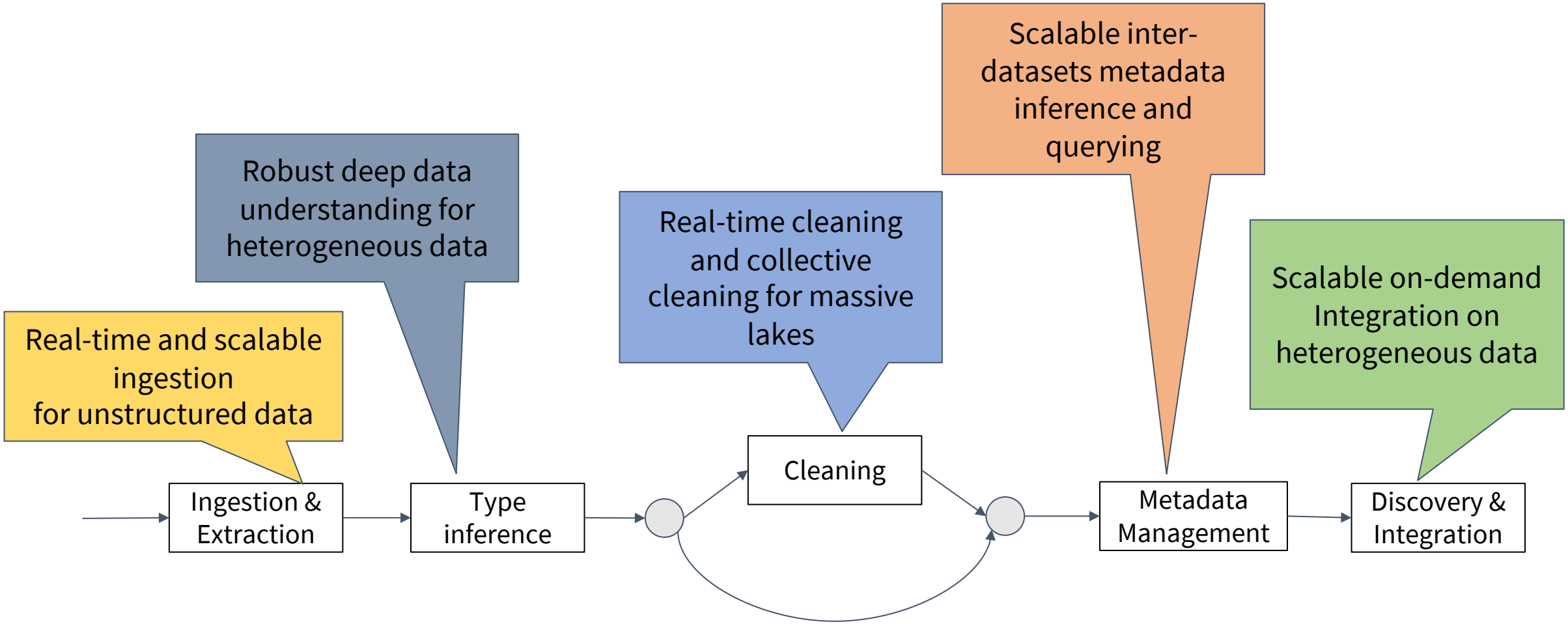
Robustness to Heterogeneity



Accuracy


High
Approx.
Low
DeepDive
simdjson
Mison
Distributed CSV parsing
sparser
Recovering Semantic
Sherlock
Deequ
JOSIE
WWT
Aurum
LSH Ensemble
Lazo
OCTOPUS
WebTables
DATAMARAN
Ingestion & Extraction
Type inference
Cleaning
Metadata Management
Discovery & Integration

Open Areas


Scalable inter-datasets metadata inference and querying
Robust deep data understanding for heterogeneous data
Real-time cleaning and collective cleaning for massive lakes
Scalable on-demand Integration on heterogeneous data
Real-time and scalable ingestion for unstructured data
Cleaning
Ingestion & Extraction
Type inference
Metadata Management
Discovery & Integration