(Big) Data Cleaning

Big Data Course

Eduardo Pena
(Invited lecture for Prof. Juliana Freire)
eduardo.pena@nyu.edu.br

Outline

1. Motivation
2. Types of Data Errors
3. Error Detection
4. Data Repair
5. Data Profiling

The Consequences of Poor Data Quality

You're Approaching Data Wrong--and It's Costing the U.S. $3 Trillion Millions of business decisions are driven by data, but few leaders stop to question if the data is actually accurate.

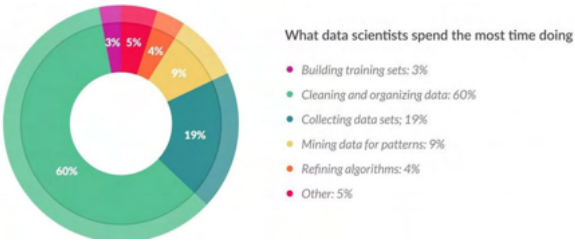
https://www.inc.com/anne-gherini/why-your-bad-data-is-creating-a-3-trillion-problem.html

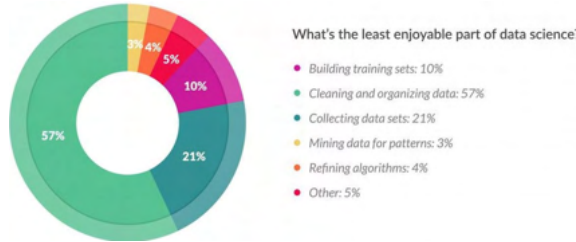
The Sources of Poor Data Quality

- **Data integration issues**: Conversion errors can occur when data is collected from different databases that don't integrate well with the organization's database.
- **Data-capturing inconsistencies**: Variations in formatting processes across different departments can lead to inaccuracies in data capture.
- **Poor data migration**: Moving data from legacy systems to new databases or the cloud can result in missing or irregular data values.
- **Data decay**: Deterioration of data quality over time, often due to outdated information or changes in customer data.
- **Data duplication**: Duplicated data can skew business intelligence and lead to problems if not properly managed.
- ...

Data Cleaning in Reality

Most of the time is spent on "boring" tasks!





"Cleaning Data: Most Time-Consuming, Least Enjoyable Data Science Task", Gil Press, Forbes, March 23rd, 2016 `http://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/`

The Reasons for Poor Data Quality

The Reasons for Poor Data Quality

Data Cleaning: Tasks and Tools

Categories	Available features	Data preparation tools						
		Altair	Paxata	SAP	SAS	Tableau	Talend	Trifacta
Data discovery	Locate missing values (nulls)	✓	✓	✓	✓	✓	✓	✓
	Locate outliers		✓		✓			✓
	Search by pattern	✓	✓	✓	✓	✓	✓	✓
	Sort data	✓	✓	✓	✓	✓	✓	✓
Data validation	Compare values (selection and join)	✓	✓	✓		✓	✓	✓
	Check data range	✓	✓	✓		✓	✓	✓
	Check permitted characters							✓
	Check column uniqueness	✓	✓	✓		✓	✓	✓
	Find type-mismatched data		✓	✓		✓	✓	✓
	Find data-mismatched datatypes		✓				✓	✓
Data structuring	Change column data type		✓	✓	✓	✓	✓	✓
	Delete column			✓	✓	✓	✓	✓
	Detect & change encoding						✓	✓
	Pivot / unpivot					✓		✓
	Rename column						✓	✓
	Split column							✓
	Transform by example [13]							✓
Data enrichment	Assign semantic data type							
	Calculate column using ex							
	Discover & merge extern							
	Duplicate column							
	Generate primary key co							
	Join & union							
	Merge columns							
	Normalize numeric values	✓						✓
Data filtering	Delete/keep filtered rows	✓	✓					✓
	Delete empty and invalid rows	✓	✓	✓				✓
	Extract value parts	✓			✓			✓
	Filter with regular expressions							✓
Data cleaning	Change date & time format	✓	✓	✓	✓	✓	✓	✓
	Change letter case	✓	✓	✓	✓	✓	✓	✓
	Change number format	✓	✓	✓	✓	✓	✓	✓
	Deduplicate data	✓	✓	✓	✓		✓	✓
	Delete by pattern	✓	✓		✓	✓	✓	✓
	Edit & replace cell data	✓	✓	✓	✓	✓	✓	✓
	Fill empty cells	✓	✓				✓	✓
	Remove extra whitespace	✓	✓	✓	✓	✓	✓	✓
	Remove diacritics			✓				
	Standardize strings by pattern		✓	✓	✓	✓	✓	✓
	Standardize values in clusters		✓	✓	✓	✓	✓	✓

Data Cleaning: Tasks and Tools

Tool name	URL
Altair Monarch Data Preparation	https://www.datawatch.com/in-action/monarch-draft/
Alteryx Data Preparation	https://www.alteryx.com/solutions/analytics-need/data-preparation
BigGorilla Data Preparation	https://www.biggorilla.org/
Cambridge Semantics Anzo	https://www.cambridgesemantics.com/
Datameer	https://www.datameer.com/
EasyMorph Data Preparation and Automation	https://easymorph.com/
Erwin	https://erwin.com/
FICO	https://www.fico.com/
Google Cloud Data Prep by Trifacta	https://cloud.google.com/dataprep/
Hitachi-Pentaho Business Analytics	https://www.hitachivantara.com/en-us/products/data-management-analytics.html
IBM Data Refinery	https://www.ibm.com/cloud/data-refinery
INFOGIX	https://www.infogix.com/data3sixty/analyze/
Informatica Enterprise Data Preparation	https://www.informatica.com/products/data-catalog/enterprise-data-prep.html
Looker	https://looker.com/
Lore IO	https://www.getlore.io/
Microsoft Power BI	https://powerbi.microsoft.com/en-us/
MicroStrategy	https://www.microstrategy.com/us/product/analytics/data-visualization
Modak-nabu	https://modakanalytics.com/nabu.html
OpenRefine	http://openrefine.org/
Oracle Analytics Cloud	https://www.oracle.com/business-analytics/analytics-cloud.html
Paxata Self Service Data Preparation	https://www.paxata.com/self-service-data-prep/
Qlik Data Catalyst	https://www.qlik.com/us/products/qlik-data-catalyst
Quest Toad Data Point	https://www.quest.com/products/toad-data-point/
Rapid Insight	https://www.rapidinsight.com/solutions/data-preparation/
RapidMiner Turbo Prep	https://rapidminer.com/products/turbo-prep/
SAP Agile Data Preparation	https://www.sap.com/germany/products/data-preparation.html
SAS Data Preparation	https://www.sas.com/en_us/software/data-preparation.html
Smarten Advanced Data Discovery	https://www.smarten.com/self-serve-data-preparation.html
Solix Common Data Platform	https://www.solix.com/products/solix-common-data-platform/
Sparkflows	https://www.sparkflows.io/data-science
Tableau Prep	https://www.tableau.com/products/prep
Talend Data Preparation	https://www.talend.com/products/data-preparation/
Tamr	https://www.tamr.com/
Teradata Vantage	https://www.teradata.com/Products/Software/Vantage
TIBCO Spotfire Analytics	https://www.tibco.com/products/tibco-spotfire
TMMData	https://www.tmmdata.com/
Trifacta Wrangler	https://www.trifacta.com/products/wrangler-editions/
Unifi Data Platform	https://unifisoftware.com/platform/
Waterline Data	https://www.waterlinedata.com/
Workday-Prism Analytics	https://www.workday.com/en-us/applications/analytics/prism-analytics.html
Yellowfin Data Prep	https://www.yellowfinbi.com/suite/data-prep
Zoho Analytics	https://www.zoho.com/analytics/

Outline

(Unsystematic) Dirty Data

	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	02-2012	300 000	100
t_2	101	R. Geller	Research	02/15/2014	8000	102
t_3	102	D. Brown	Research	02/15/2014	6000	101
t_4	103	H. McCoy	Research	02/15/2015	8000	101
t_5	104	H. McCoy	Research	02/15/2015	8000	101
t_6	105	H. Souza	Marketing	02/15/2015	8000	H. McCoy
t_7	100	H. Silva	Research		8000	101

(Unsystematic) Dirty Data

	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	02-2012	300 000	100
t_2	101	R. Geller	Research	02/15/2014	8000	102
t_3	102	D. Brown	Research	02/15/2014	6000	101
t_4	103	H. McCoy	Research	02/15/2015	8000	101
t_5	104	H. McCoy	Research	02/15/2015	8000	101
t_6	105	H. Souza	Marketing	02/15/2015	8000	H. McCoy
t_7	100	H. Silva	Research		8000	101

Missing Values

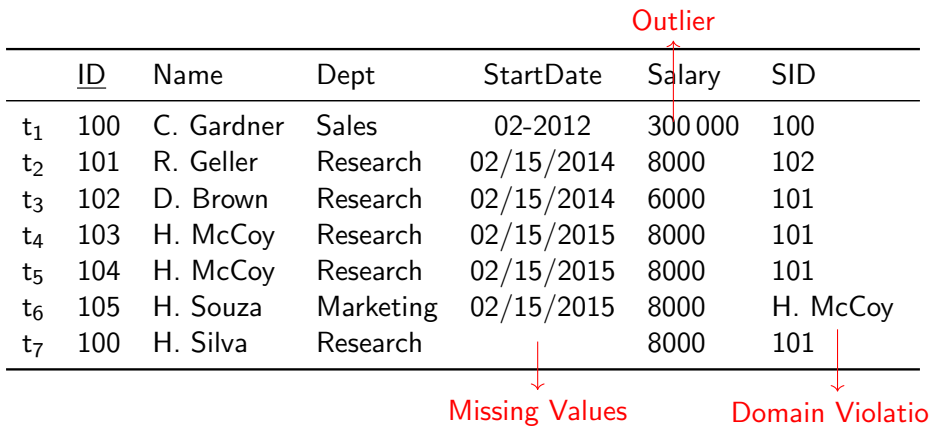
(Unsystematic) Dirty Data

	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	02-2012	300 000	100
t_2	101	R. Geller	Research	02/15/2014	8000	102
t_3	102	D. Brown	Research	02/15/2014	6000	101
t_4	103	H. McCoy	Research	02/15/2015	8000	101
t_5	104	H. McCoy	Research	02/15/2015	8000	101
t_6	105	H. Souza	Marketing	02/15/2015	8000	H. McCoy
t_7	100	H. Silva	Research		8000	101

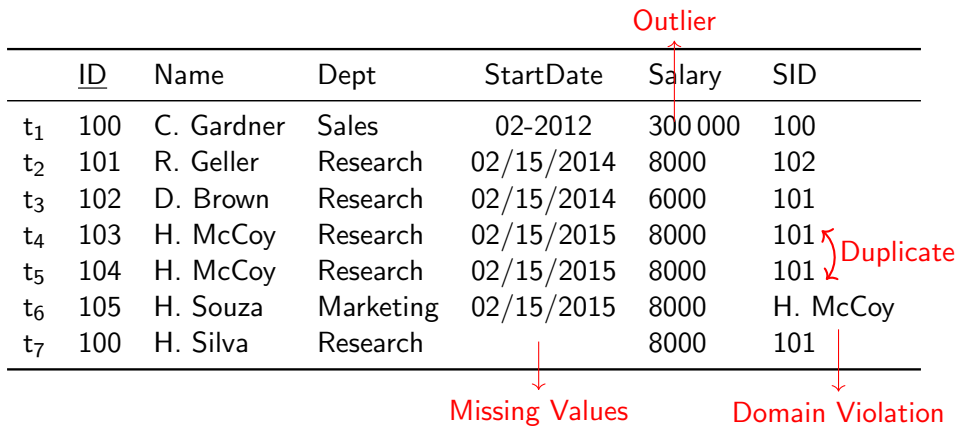
Missing Values

Domain Violation

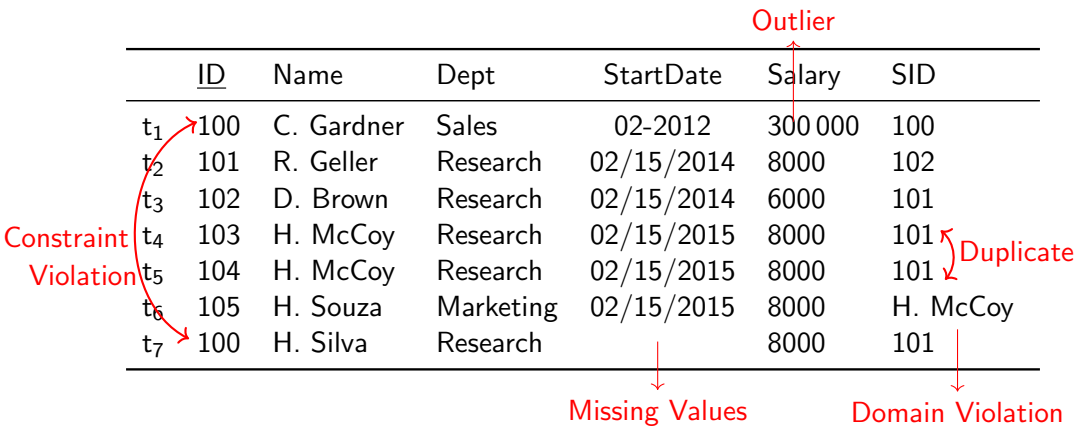
(Unsystematic) Dirty Data



(Unsystematic) Dirty Data

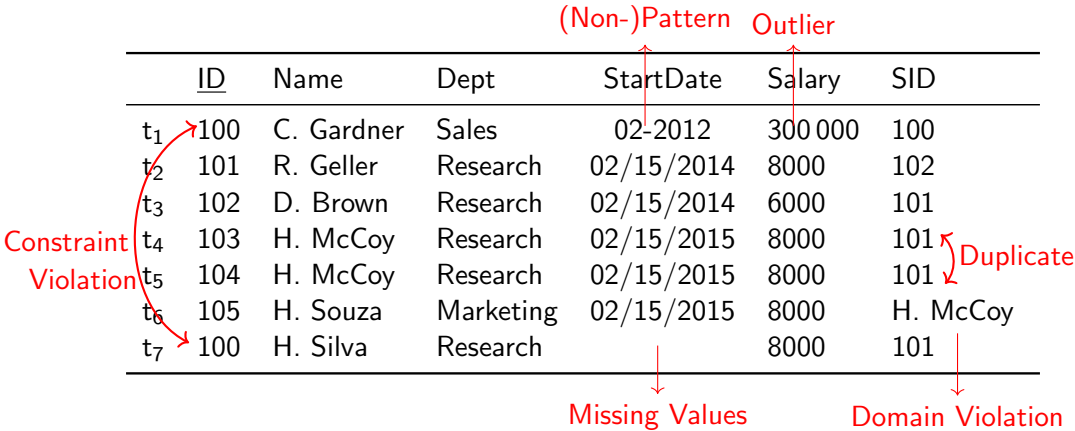


(Unsystematic) Dirty Data



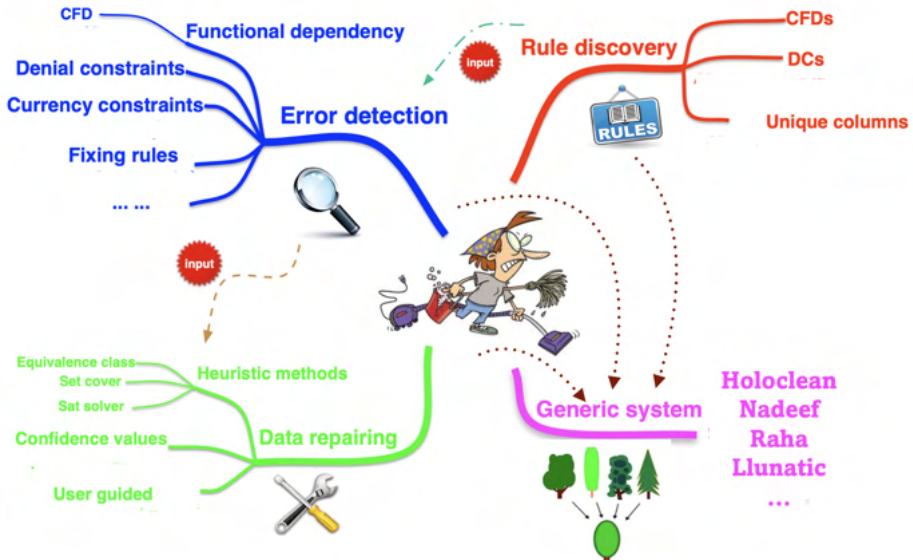
	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	02-2012	300 000	100
t_2	101	R. Geller	Research	02/15/2014	8000	102
t_3	102	D. Brown	Research	02/15/2014	6000	101
t_4	103	H. McCoy	Research	02/15/2015	8000	101
t_5	104	H. McCoy	Research	02/15/2015	8000	101
t_6	105	H. Souza	Marketing	02/15/2015	8000	H. McCoy
t_7	100	H. Silva	Research		8000	101

(Unsystematic) Dirty Data



	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	02-2012	300 000	100
t_2	101	R. Geller	Research	02/15/2014	8000	102
t_3	102	D. Brown	Research	02/15/2014	6000	101
t_4	103	H. McCoy	Research	02/15/2015	8000	101
t_5	104	H. McCoy	Research	02/15/2015	8000	101
t_6	105	H. Souza	Marketing	02/15/2015	8000	H. McCoy
t_7	100	H. Silva	Research		8000	101

Data Cleaning Components



Outline

Types of Data Errors

An error is a deviation from its ground truth value in a dataset.

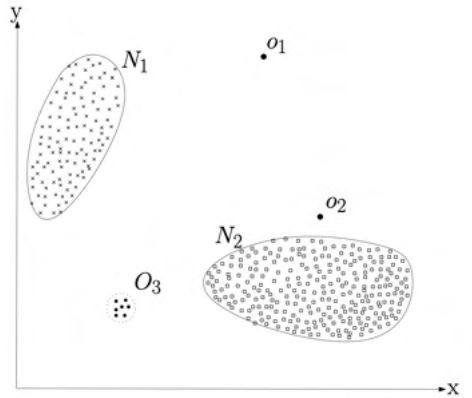
- **Outliers**: Deviations from the distribution of values in a column.
- **Duplicates**: Distinct records referring to the same entity with mismatched attribute values.
- **Rule Violations**: Values violating integrity constraints.
- **Pattern Violations**: Values violating syntactic or semantic constraints.

Note: this categories may overlap, and they're not exhaustive, but they help categorize detectable errors in real-world datasets.

Outliers/Anomalies

Outliers

- **Outliers are patterns in data that deviate from normal behavior.**
- The result from various factors:
  - Malicious activity (e.g., fraud, cyber-intrusion)
  - System breakdowns



Challenges in Outlier Detection

- Unclear boundaries between normal and anomalous behavior
- Evolving nature of normal behavior
- Varied definitions of outliers across domains
- Limited availability of labeled training data
- Similarity between noise and outliers

Outlier Detection

Types of Outliers

- Point outliers: single data points far from the data distribution
- Contextual outliers: systematic outliers in data (i.e., depends on surroundings)
- Sequence outlier: sequence of values with abnormal shape/aggregation
- Univariate vs multivariate analysis

Outlier Detection

Types of Outliers

- Point outliers: single data points far from the data distribution
- Contextual outliers: systematic outliers in data (i.e., depends on surroundings)
- Sequence outlier: sequence of values with abnormal shape/aggregation
- Univariate vs multivariate analysis

Types of Outlier Detection

- Unsupervised: No prior knowledge of data, similar to unsupervised clustering
  - Requires setting distance metrics, # errors
- Supervised: Labeled normal and abnormal data, similar to supervised classification
  - Requires labeled data
- Normal Model: Represent normal behavior, similar to pattern recognition
  - Requires rules/constraints for the model

Chandola, Banerjee, and Kumar. Outlier Detection : A Survey. *ACM Computing Surveys* 2009

Outlier Detection in Relational Data

People

id	name	age
1	Alice	32
2	Bob	35
3	Charlie	42
4	David	39
5	Emily	47
6	Frank	52
7	Grace	50
8	Hannah	38
9	Isaac	41
10	Jane	365

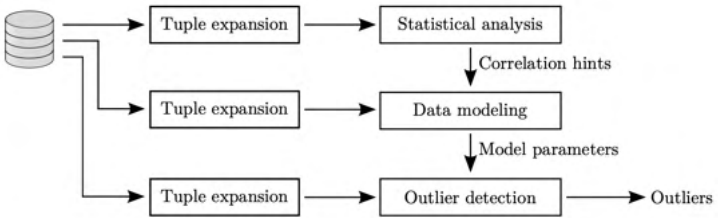
```
SELECT * FROM People
WHERE age
<(SELECT AVG(age) - 2*STDDEV(age) FROM People)
OR age
>(SELECT AVG(age) + 2*STDDEV(age) FROM People)
```

↓

id	name	age
10	Jane	365

Outlier Detection: DBoost

- Automatically creates derived attributes for every tuple using expansion rules.
- These derived attributes are processed by outlier detection models to learn soft constraints and detect soft functional dependencies.
- The method extracts structured attributes from unstructured data, enabling easier detection of inconsistencies.

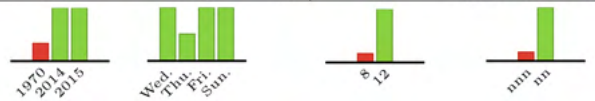


Pit-Claudel, Mariet, Harding, Madden. Outlier Detection in Heterogeneous Datasets using Automatic Tuple Expansion. *Tech Report* 2016

Outlier Detection: DBoost

- Automatically creates derived attributes for every tuple using expansion rules.
- These derived attributes are processed by outlier detection models to learn soft constraints and detect soft functional dependencies.
- The method extracts structured attributes from unstructured data, enabling easier detection of inconsistencies.

	Original Data		Expansions of Reg. Date (not materialized)			Expansions of SSN (not materialized)		
XID	Reg. Date	SSN	Year	Weekday	...	Length	Strip Numbers	...
1	1416497422	783-345-2351	2014	Thursday	...	12	<num>-<num>-<num>	...
2	1418201134	773-746	2014	Wednesday	...	**8**	**<num>-<num>**	...
3	1420359855	773-289-5552	2015	Sunday	...	12	<num>-<num>-<num>	...
4	1421575392	849-843-2729	2015	Sunday	...	12	<num>-<num>-<num>	...
5	01302015	773-387-9201	**1970**	Friday	...	12	<num>-<num>-<num>	...
6	1424866716	821-322-1857	2015	Wednesday	...	12	<num>-<num>-<num>	...
7	1425059692	822-971-1892	2015	Friday	...	12	<num>-<num>-<num>	...

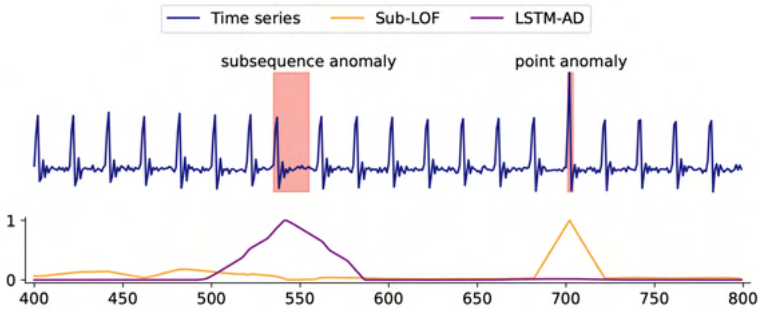


Pit-Claudel, Mariet, Harding, Madden. Outlier Detection in Heterogeneous Datasets using Automatic Tuple Expansion. *Tech Report* 2016

Time Series Anomaly Detection

Time series (TS) are ordered sequences of data points describing properties of objects or processes, often measured over time.

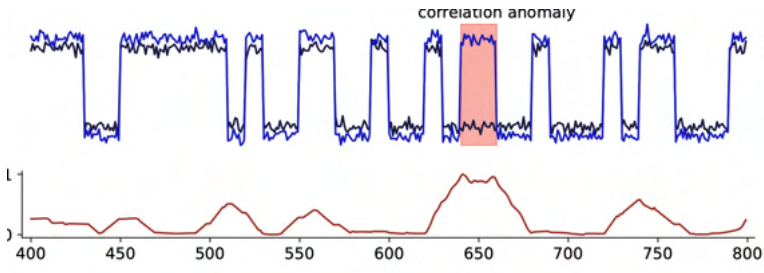
- TS anomalies are points or sequences deviating from regular patterns.
- TS anomalies can occur in univariate or multivariate time series, affecting individual channels or correlations between channels.
- TS anomalies may vary in length and can reappear within the same time series.



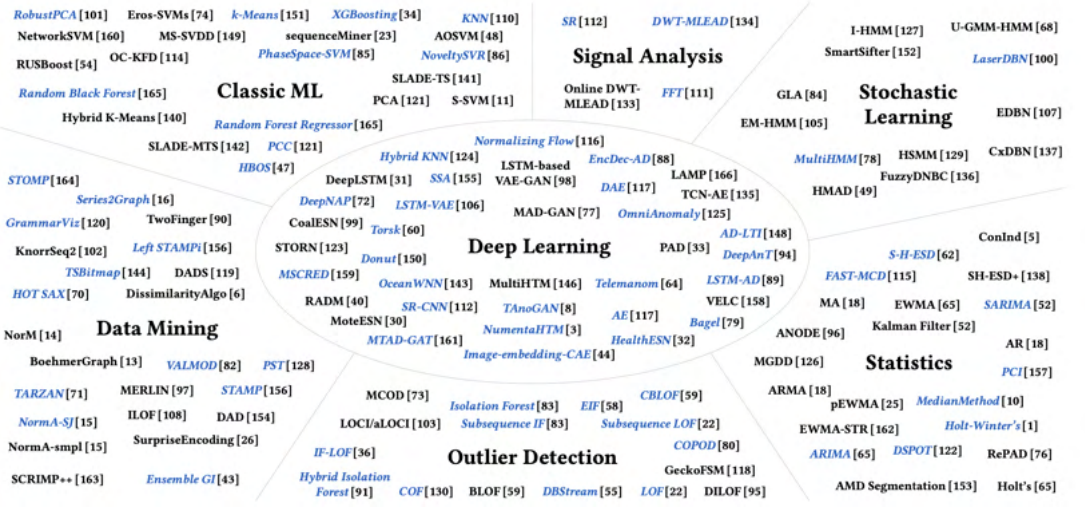
Time Series Anomaly Detection

Time series (TS) are ordered sequences of data points describing properties of objects or processes, often measured over time.

- TS anomalies are points or sequences deviating from regular patterns.
- TS anomalies can occur in univariate or multivariate time series, affecting individual channels or correlations between channels.
- TS anomalies may vary in length and can reappear within the same time series.



Approaches for Time Series Anomaly Detection



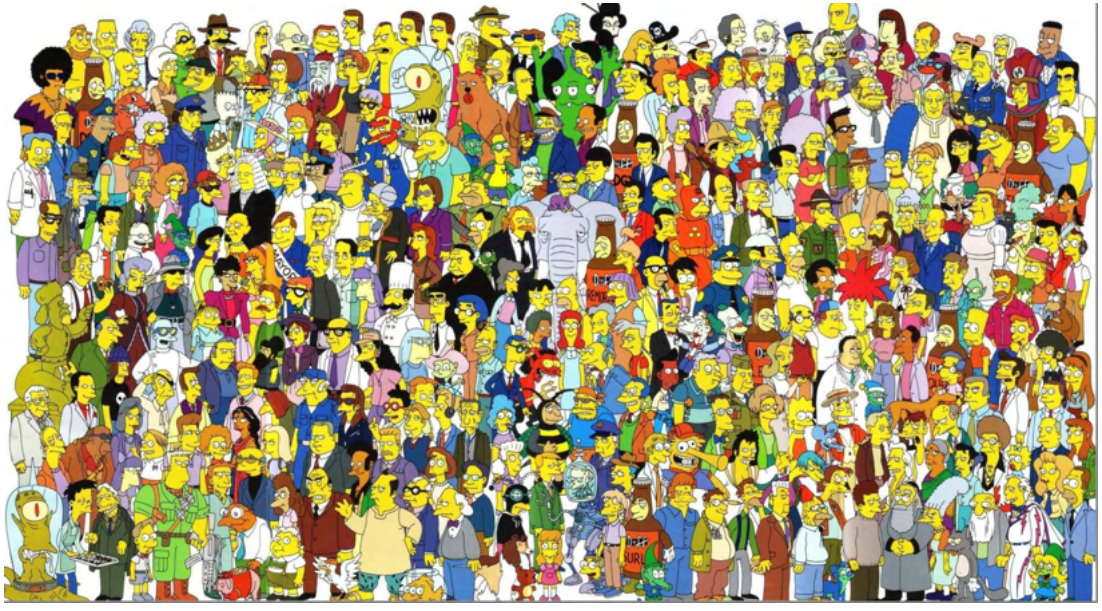
Approaches for Time Series Anomaly Detection

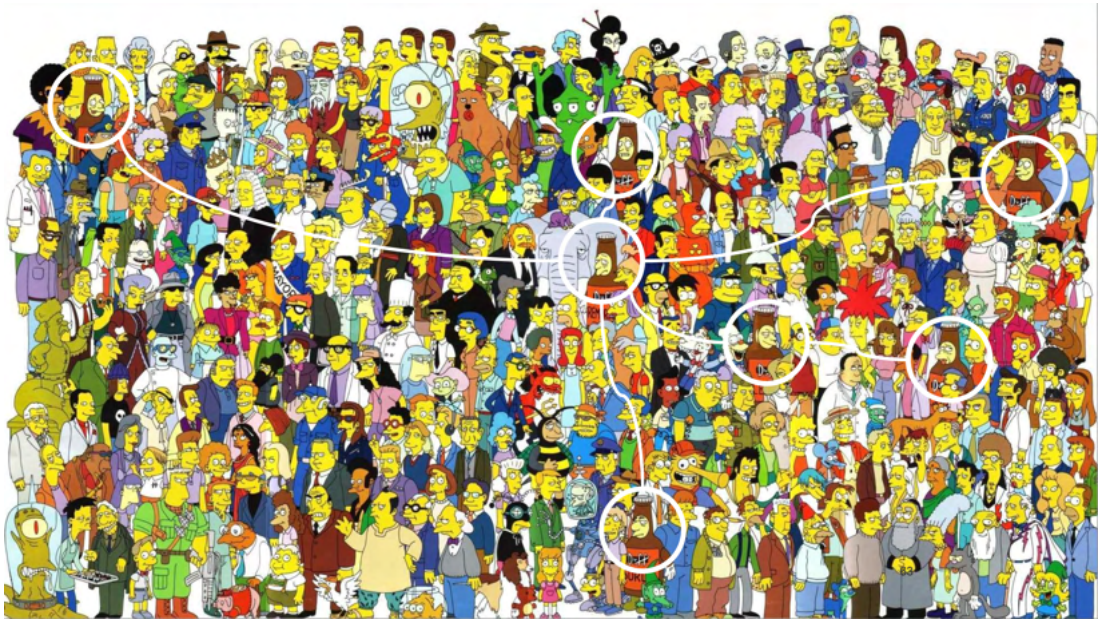
Main Results

- Deep learning approaches are not yet competitive, despite their higher processing effort on training data;
- Simple methods yield performance almost as good as more sophisticated methods;
- No single algorithm clearly outperforms others, highlighting the need for **further research**.
- Overall performance calls for research in flexibility, reliability, and simplicity.
  - Flexibility: Holistic and hybrid Outlier detection systems combining existing strengths for detecting diverse anomalies in time series.
  - Reliability: Further research on the robustness and scalability of time series Outlier detection algorithms is needed.
  - Simplicity: Most algorithms were sensitive to parameter settings, indicating the need for research on auto-configuring and self-tuning algorithms.

Schmidl, Wenig, Papenbrock. Anomaly Detection in Time Series: A Comprehensive Evaluation. *VLDB* 2022

Duplicates

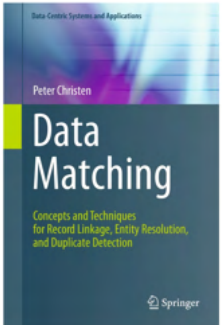


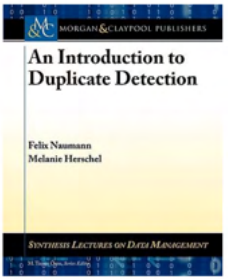


Duplicate Detection

Fun fact: "Duplicate Detection" has various duplicates (alternative names).

- Record Linkage
- Entity Resolution
- Object Identification
- Doubles
- Object Consolidation
- Entity Clustering
- Fuzzy Match
- Approximate Match
- Reference Reconciliation
- Reference Matching
- Merge/Purge





Duplicate Detection in Relational data

Product	Company	Year	Memory	Screen
iPhone 10	Apple	2017	64GB	5.8 inches
iPhone 11	Apple Inc	2019	128GB	6.1 inches
iPhone X	Apple	2017	64GB	5.8-inch
Duplicates:				
iPhone 10	Apple	2017	64GB	5.8 inches
iPhone X	Apple	2017	64GB	5.8-inch

Duplicate Detection in Relational data

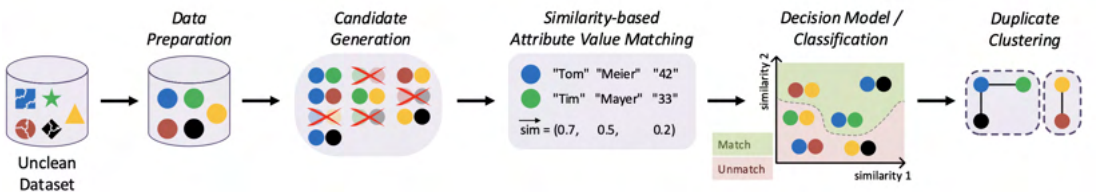
Product	Company	Year	Memory	Screen
iPhone 10	Apple	2017	64GB	5.8 inches
iPhone 11	Apple Inc	2019	128GB	6.1 inches
iPhone X	Apple	2017	64GB	5.8-inch
Duplicates:				
iPhone 10	Apple	2017	64GB	5.8 inches
iPhone X	Apple	2017	64GB	5.8-inch

We can see duplicate detection as a **special kind of join**:

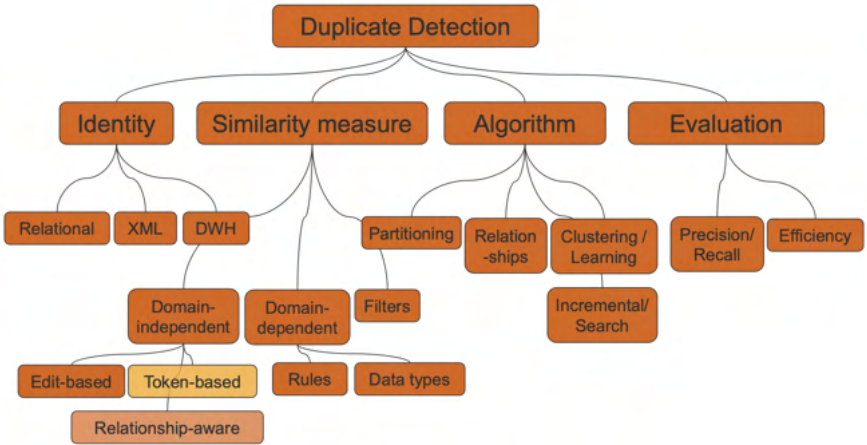
```
SELECT P1.* ,P2.*
FROM Product AS P1, Product AS P2
WHERE SIM(P1,P2) > θ
```

Duplicate Detection Research

The five steps of a typical duplicate detection pipeline based on pairwise record comparisons:



Duplicate Detection Research



Token-based example: Jaccard Coefficient

Jaccard Coefficient: compares two sets P and Q with the following formula:

$$\text{Jaccard}(P, Q) = \frac{|P \cap Q|}{|P \cup Q|}.$$

Token-based example: Jaccard Coefficient

Jaccard Coefficient: compares two sets P and Q with the following formula:

$$\text{Jaccard}(P, Q) = \frac{|P \cap Q|}{|P \cup Q|}.$$

We compute the Jaccard coefficient of two strings s_1 and s_2 as:

$$\text{StringJaccard}(s_1, s_2) = \frac{|\text{tokenize}(s_1) \cap \text{tokenize}(s_2)|}{|\text{tokenize}(s_1) \cup \text{tokenize}(s_2)|}.$$

Token-based example: Jaccard Coefficient

Jaccard Coefficient: compares two sets P and Q with the following formula:

$$\text{Jaccard}(P, Q) = \frac{|P \cap Q|}{|P \cup Q|}.$$

We compute the Jaccard coefficient of two strings s_1 and s_2 as:

$$\text{StringJaccard}(s_1, s_2) = \frac{|\text{tokenize}(s_1) \cap \text{tokenize}(s_2)|}{|\text{tokenize}(s_1) \cup \text{tokenize}(s_2)|}.$$

Assuming a tokenization function based on spaces:

$$\text{tokenize}(\text{Thomas Sean Connery}) = \{\text{Thomas}, \text{Sean}, \text{Connery}\},$$
$$\text{tokenize}(\text{Sir Sean Connery}) = \{\text{Sir}, \text{Sean}, \text{Connery}\}.$$

Token-based example: Jaccard Coefficient

Jaccard Coefficient: compares two sets P and Q with the following formula:

$$\text{Jaccard}(P, Q) = \frac{|P \cap Q|}{|P \cup Q|}.$$

We compute the Jaccard coefficient of two strings s_1 and s_2 as:

$$\text{StringJaccard}(s_1, s_2) = \frac{|\text{tokenize}(s_1) \cap \text{tokenize}(s_2)|}{|\text{tokenize}(s_1) \cup \text{tokenize}(s_2)|}.$$

Assuming a tokenization function based on spaces:

$$\text{tokenize}(\text{Thomas Sean Connery}) = \{\text{Thomas}, \text{Sean}, \text{Connery}\},$$
$$\text{tokenize}(\text{Sir Sean Connery}) = \{\text{Sir}, \text{Sean}, \text{Connery}\}.$$

Then:

$$\text{StringJaccard}(\text{Thomas Sean Connery}, \text{Sir Sean Connery}) = \frac{2}{4}.$$

Edit-based example: Levenshtein distance

Levenshtein distance: measures the minimum number of operations required to transform one string into another.

- Operations typically include: insertion, deletion, or substitution of a single character.

Edit-based example: Levenshtein distance

Levenshtein distance: measures the minimum number of operations required to transform one string into another.

- Operations typically include: insertion, deletion, or substitution of a single character.

- For example, let's compute the edit distance between the words "cat" and "hat".
  - Initial state: "cat" $\rightarrow$ "hat"
  - Operations:
    1. Substitute 'h' for 'c': "cat" $\rightarrow$ "hat"
  - The minimum number of operations required is 1. Therefore, the edit distance between "cat" and "hat" is 1.

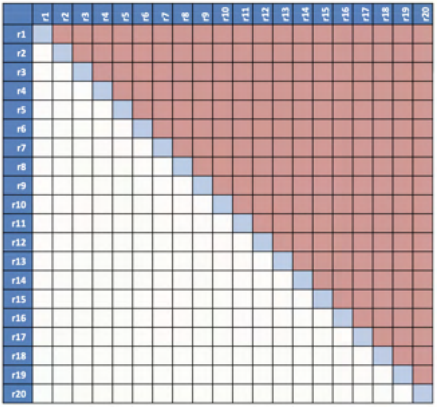
Pairwise Comparisons Algorithms

Instead of comparing all pairs of records $\rightarrow$ partition the records into smaller subsets.
If we can assume that duplicate records appear only within the same partition, it is sufficient to compare all record-pairs within each partition.

Two competing approaches:

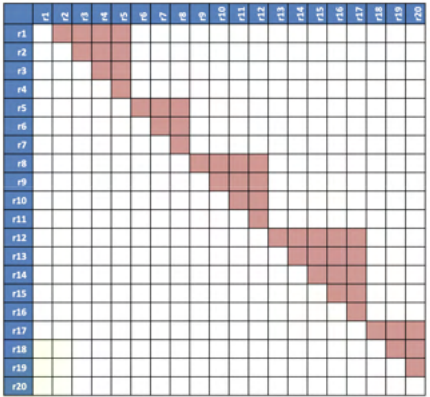
- **Blocking methods:** Strictly partition records into disjoint subsets, for instance, using zip codes as partitioning key.
- **Windowing methods:** In particular, the Sorted-Neighborhood method, sorts the data according to some key, such as zip code, and then slides a window of fixed size across the sorted data and compares pairs only within the window.
- **Both methods can be enhanced by running multiple partitioning/windowing passes over the data.**

Pairwise Comparisons Algorithms: Intuition



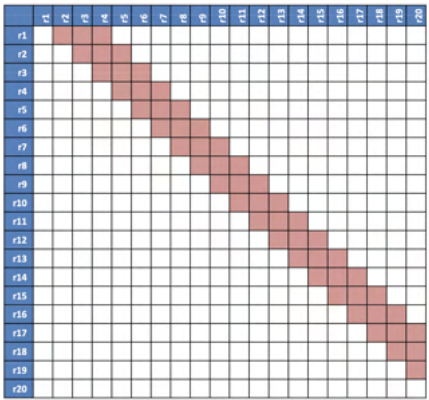
Matrix of duplicate candidates

Pairwise Comparisons Algorithms: Intuition



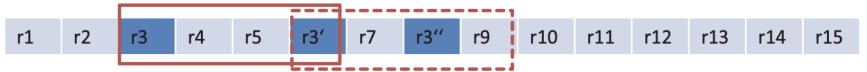
Matrix of duplicate candidates with blocking algorithm.

Pairwise Comparisons Algorithms: Intuition



Matrix of duplicate candidates with the Sorted-Neighborhood algorithm.

Pairwise Comparisons Algorithms: Intuition

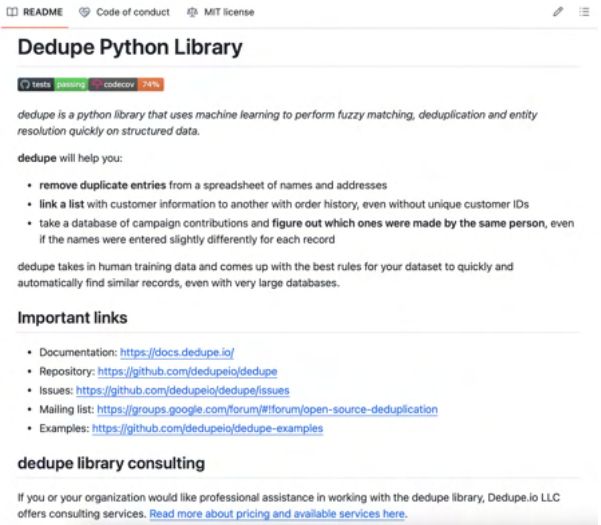


Transitive closure for duplicate candidates in Sorted-Neighborhood algorithm:

But algorithms need to compromise, as:

If $\text{sim}(A, B) < \theta$ and $\text{sim}(B, C) < \theta$, does not necessarily imply $\text{sim}(A, C) < \theta$.

Duplicate Detection Tool



README Code of conduct MIT license

Dedupe Python Library

tests passing codecov 74%

dedupe is a python library that uses machine learning to perform fuzzy matching, deduplication and entity resolution quickly on structured data.

dedupe will help you:

- **remove duplicate entries** from a spreadsheet of names and addresses
- **link a list** with customer information to another with order history, even without unique customer IDs
- take a database of campaign contributions and **figure out which ones were made by the same person**, even if the names were entered slightly differently for each record

dedupe takes in human training data and comes up with the best rules for your dataset to quickly and automatically find similar records, even with very large databases.

Important links

- Documentation: https://docs.dedupe.io/
- Repository: https://github.com/dedupeio/dedupe
- Issues: https://github.com/dedupeio/dedupe/issues
- Mailing list: https://groups.google.com/forum/#!forum/open-source-deduplication
- Examples: https://github.com/dedupeio/dedupe-examples

dedupe library consulting

If you or your organization would like professional assistance in working with the dedupe library, Dedupe.io LLC offers consulting services. Read more about pricing and available services here.

https://github.com/dedupeio/dedupe

Pattern Violations

Pattern Violation Detection in Relational Data

employeeid	name	email
1	John Doe	john@example.com
2	Jane Smith	jane@example
3	Bob Johnson	bob@example.com
4	Alice Johnson	alice@example.net

↓

```
SELECT * FROM employees
WHERE NOT REGEXP_MATCHES(email,
'^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}')
```

↓

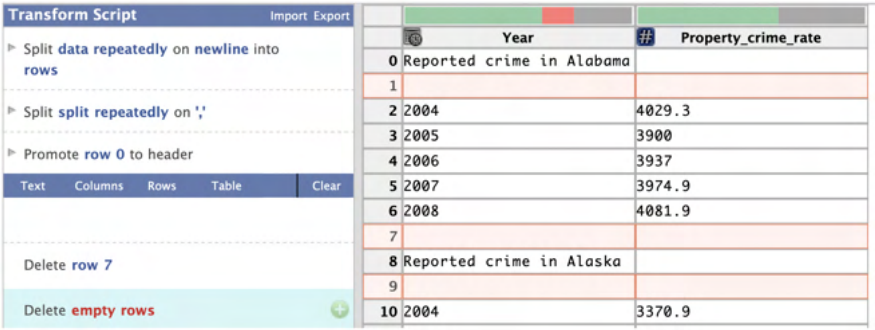
employeeid	name	email
1	Jane Smith	jane@example

Data Wrangling

- Transformation suggestions based on interaction context.
- Programming-by-demonstration to assist with complex criteria like regular expressions.
- Ranking transforms considering user input and other factors.
- Recording transformations for reuse and data provenance.

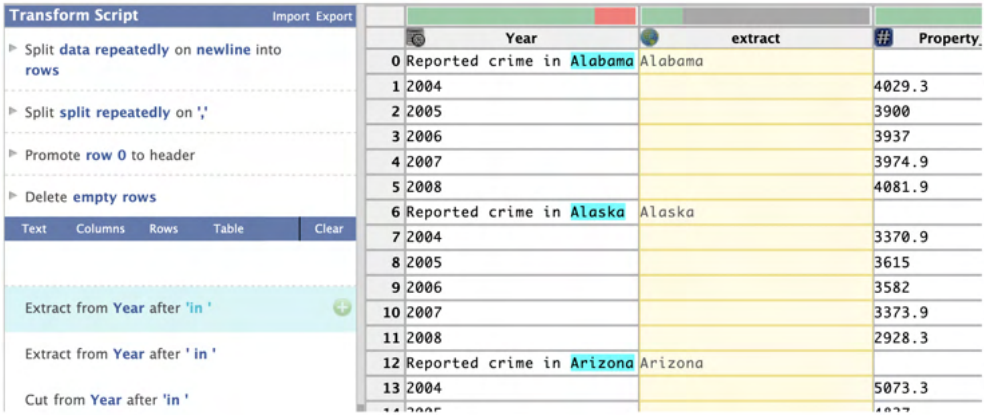
Kandel, Paepcke, Hellerstein, and Heer. Wrangler: Interactive visual specification of data transformation scripts. *CHI* 2011

Data Wrangling



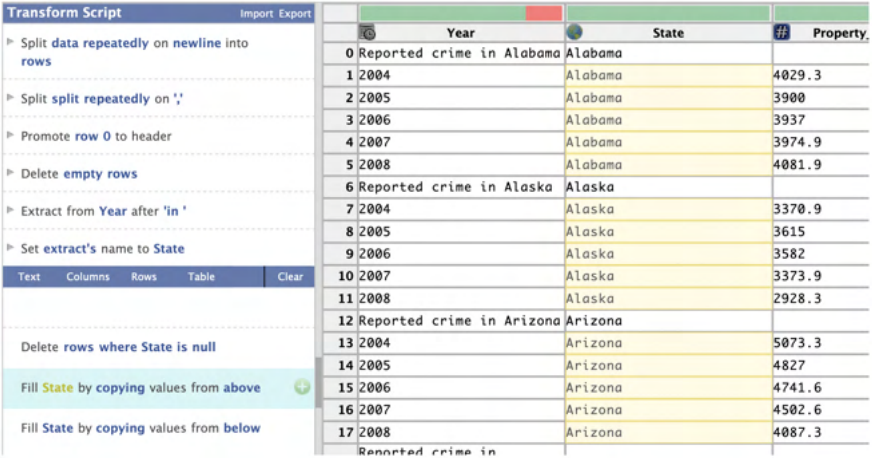
Kandel, Paepcke, Hellerstein, and Heer. Wrangler: Interactive visual specification of data transformation scripts. *CHI* 2011

Data Wrangling



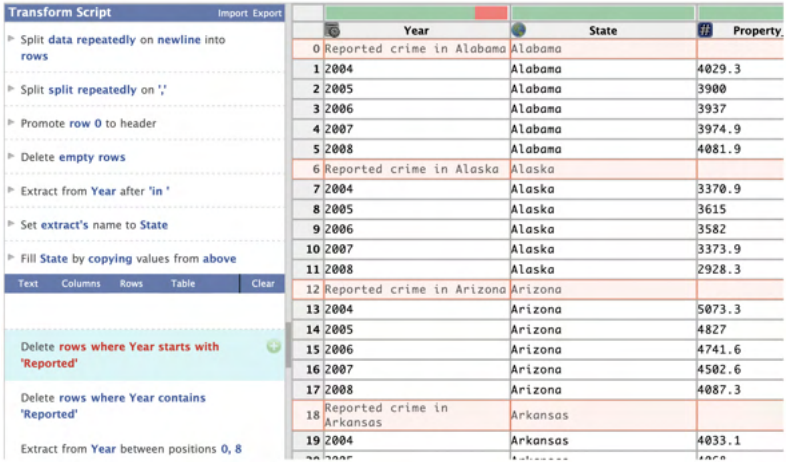
Kandel, Paepcke, Hellerstein, and Heer. Wrangler: Interactive visual specification of data transformation scripts. *CHI* 2011

Data Wrangling



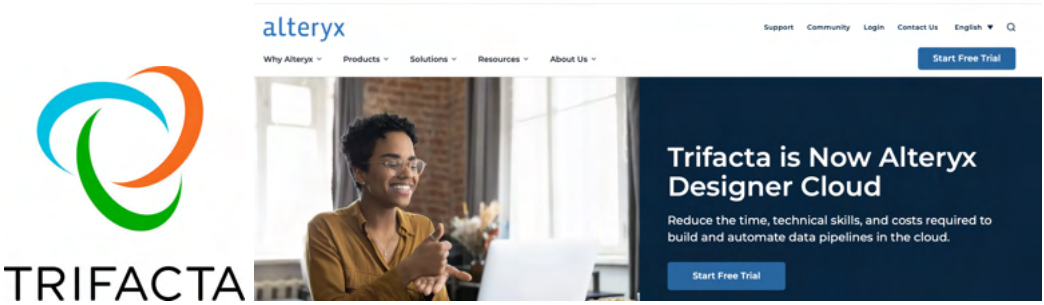
Kandel, Paepcke, Hellerstein, and Heer. Wrangler: Interactive visual specification of data transformation scripts. *CHI* 2011

Data Wrangling



Kandel, Paepcke, Hellerstein, and Heer. Wrangler: Interactive visual specification of data transformation scripts. *CHI* 2011

Data Wrangling in the Market



https://en.wikipedia.org/wiki/Trifacta

Pattern Violation Tools: Katara

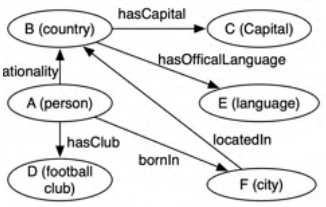
	A	B	C	D	E	F	G
t_1	Rossi	Italy	Rome	Verona	Italian	Proto	1.78
t_2	Klate	S. Africa	Pretoria	Pirates	Afrikaans	P. Eliz.	1.69
t_3	Pirlo	Italy	Madrid	Juve	Italian	Flero	1.77

- Cells are verified against a knowledge base.
- If a cell is not in the knowledge base, it is declared as an error.
- Manual construction of knowledge bases is necessary for each dataset due to domain specificity.

Chu et. al. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. *SIGMOD* 2015

Pattern Violation Tools: Katara

	A	B	C	D	E	F	G
t_1	Rossi	Italy	Rome	Verona	Italian	Proto	1.78
t_2	Klate	S. Africa	Pretoria	Pirates	Afrikaans	P. Eliz.	1.69
t_3	Pirlo	Italy	Madrid	Juve	Italian	Flero	1.77

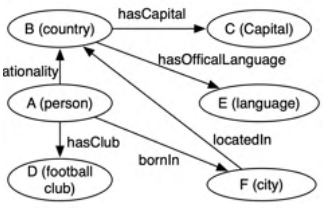


(a) A table pattern φ_s

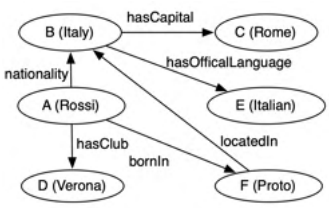
Chu et. al. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. *SIGMOD* 2015

Pattern Violation Tools: Katara

	A	B	C	D	E	F	G
t_1	Rossi	Italy	Rome	Verona	Italian	Proto	1.78
t_2	Klate	S. Africa	Pretoria	Pirates	Afrikaans	P. Eliz.	1.69
t_3	Pirlo	Italy	Madrid	Juve	Italian	Flero	1.77



(a) A table pattern φ_s

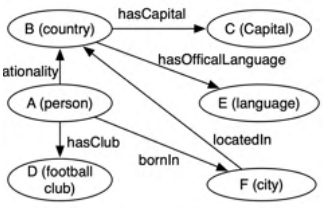


(b) t_1: validated by KB

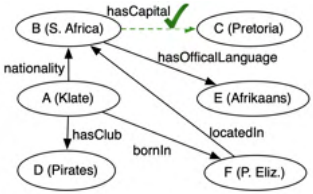
Chu et. al. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. *SIGMOD* 2015

Pattern Violation Tools: Katara

	A	B	C	D	E	F	G
t_1	Rossi	Italy	Rome	Verona	Italian	Proto	1.78
t_2	Klate	S. Africa	Pretoria	Pirates	Afrikaans	P. Eliz.	1.69
t_3	Pirlo	Italy	Madrid	Juve	Italian	Flero	1.77



(a) A table pattern φ_s

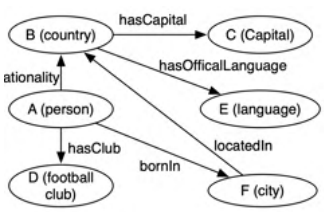


(c) t_2: validated by KB&crowd

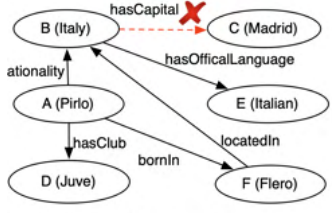
Chu et. al. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. *SIGMOD* 2015

Pattern Violation Tools: Katara

	A	B	C	D	E	F	G
t_1	Rossi	Italy	Rome	Verona	Italian	Proto	1.78
t_2	Klate	S. Africa	Pretoria	Pirates	Afrikaans	P. Eliz.	1.69
t_3	Pirlo	Italy	Madrid	Juve	Italian	Flero	1.77



(a) A table pattern φ_s



(d) t_3: Erroneous tuple

Chu et. al. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. *SIGMOD* 2015

Constraint Violations

Modeling Constraints

emp

	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

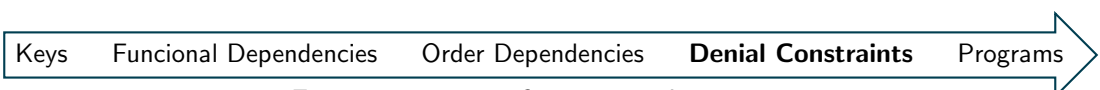
– Data Quality Rules:

- Each employee has a unique value of ID.
- Employees cannot supervise their own supervisors (SID).
- **For any two employees from the same department, the employee with the most years of service should not earn the lowest salary.** (Our running example.)

Modeling Constraints

emp

	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101



Expressive power of constraint languages

Modeling Constraints

emp

	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

– Data quality rules as **Denial Constraints (DCs)**:

- $\varphi_1 \colon \forall t, t' \in \text{emp}\ \neg(t.\text{ID} = t'.\text{ID})$
- $\varphi_2 \colon \forall t, t' \in \text{emp}\ \neg(t.\text{ID} = t'.\text{SID} \wedge t.\text{SID} = t'.\text{ID})$
- $\varphi_3 \colon \forall t, t' \in \text{emp}\ \neg(t.\text{Dept} = t'.\text{Dept} \wedge t.\text{StartDate} < t'.\text{StartDate} \wedge t.\text{Salary} < t'.\text{Salary})$

Error detection in DC-based Data Cleaning Systems

emp

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

$$\forall t, t' \in \text{emp}\ \neg(t.\text{Dept} = t'.\text{Dept} \wedge t.\text{StartDate} < t'.\text{StartDate} \wedge t.\text{Salary} < t'.\text{Salary})$$

Error detection in DC-based Data Cleaning Systems

emp

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

$$\forall t, t' \in \text{emp}\ \neg(t.\text{Dept} = t'.\text{Dept} \wedge t.\text{StartDate} < t'.\text{StartDate} \wedge t.\text{Salary} < t'.\text{Salary})$$

```
SELECT  t.tid ,t'.tid
FROM    emp t, emp t'
WHERE   t.Dept = t'.Dept
AND     t.StartDate < t'.StartDate
AND     t.Salary < t'.Salary
```

t_3, t_4

Error detection in DC-based Data Cleaning Systems

emp

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

$$\forall t, t' \in \text{emp}\ \neg(t.\text{Dept} = t'.\text{Dept} \wedge t.\text{StartDate} < t'.\text{StartDate} \wedge t.\text{Salary} < t'.\text{Salary})$$

Even for small datasets

No results after hours

```
SELECT  t.tid ,t'.tid
FROM    emp t, emp t'
WHERE   t.Dept = t'.Dept
AND     t.StartDate < t'.StartDate
AND     t.Salary < t'.Salary
```

t_3, t_4

Error detection in DC-based Data Cleaning Systems

emp

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

$$\forall t, t' \in \text{emp}\ \neg(t.\text{Dept} = t'.\text{Dept} \wedge t.\text{StartDate} < t'.\text{StartDate} \wedge t.\text{Salary} < t'.\text{Salary})$$

Even for small datasets

No results after hours

Some DBMSs

Run out of memory

```
SELECT  t.tid ,t'.tid
FROM    emp t, emp t'
WHERE   t.Dept = t'.Dept
AND     t.StartDate < t'.StartDate
AND     t.Salary < t'.Salary
```

t_3, t_4

Limitations of Previous Works

- SQL-based (from DBMSs with different engines)
  - Inefficient join strategies (e.g., nested loops)
  - Inappropriate materialization strategies (e.g., full materialization of Cartesian products)
  - Tuning execution for multiple types of DCs is hard (e.g., finding optimal set of indexes)

Rekatsinas, Chu, Ilyas, and Ré
HoloClean: Holistic Data Repairs with Probabilistic Inference
PVLDB 2017

Geerts, Mecca, Papotti, and Santoro
Cleaning data with Llunatic
VLDB Journal 2020

- Standalone approaches
  - Suboptimal representation of intermediates
  - Join algorithms face performance degradation for some predicates
  - Selection of predicate order using sampling-based selectivities (often off)

Bleifuß, Kruse, Naumann
Efficient Denial Constraint Discovery with Hydra
VLDB 2017

A Faster Error Detection System

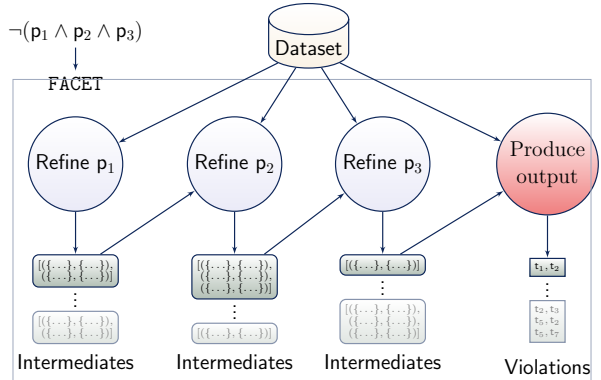
– *FAst Constraint-based Error DeTector* (FACET)

1. Algorithms and optimizations for the various (complex) types of DC predicates
2. Hybrid data structures to handle intermediates
3. Heuristic algorithms based on column sketches for robust evaluation planning
4. Multi-constraint execution using prefix trees (not discussed in this talk)

Pena, Almeida, and Naumann. Fast detection of denial constraint violations.. *PVLDB* 2021.

FACET ...

- ... maps DCs into pipelines of *refinements*, which are logical operators that:
  - operate on compact representations of pairs of tuples;
  - implement custom-designed algorithms for the different predicate structures of DCs;



Refinement operator example

$$\neg(t.Dept = t'.Dept \wedge t.StartDate < t'.StartDate \wedge t.Salary < t'.Salary)$$

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

Refinement operator example

$$\neg(t.Dept = t'.Dept \wedge t.StartDate < t'.StartDate \wedge t.Salary < t'.Salary)$$

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

$\Downarrow$

Step	Refinement	Compressed Intermediates
1	$t.Dept = t'.Dept$	$(\{t_2, t_3, t_4\}, \{t_2, t_3, t_4\})$
2	$t.StartDate < t'.StartDate$	$(\{t_3\}, \{t_2\}, \{t_4\})$
3	$t.Salary < t'.Salary$	$(\{t_4\}, \{t_2\}, \{t_3\})$

Refinement operator example

$$\neg(t.Dept = t'.Dept \wedge t.StartDate < t'.StartDate \wedge t.Salary < t'.Salary)$$

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

$\Downarrow$

Step	Refinement	Compressed Intermediates
1	$t.Dept = t'.Dept$	$(\{t_2, t_3, t_4\}, \{t_2, t_3, t_4\})$
2	$t.StartDate < t'.StartDate$	$(\{t_3\}, \{t_2, t_4\})$
3	$t.Salary < t'.Salary$	$(\{t_3\}, \{t_4\})$

Compact representations of tuple pairs

Refinement operator example

$$\neg(t.Dept = t'.Dept \wedge t.StartDate < t'.StartDate \wedge t.Salary < t'.Salary)$$

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

$\Downarrow$

Step	Refinement	Compressed Intermediates
1	$t.Dept = t'.Dept$	$(\{t_2, t_3, t_4\}, \{t_2, t_3, t_4\})$
2	$t.StartDate < t'.StartDate$	$(\{t_2, t_3\}, \{t_4\})$
3	$t.Salary < t'.Salary$	$(\{t_3\}, \{t_4\})$

Refinement operator example

$$\neg(t.Dept = t'.Dept \wedge t.StartDate < t'.StartDate \wedge t.Salary < t'.Salary)$$

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

$\Downarrow$

Step	Refinement	Compressed Intermediates
1	$t.Dept = t'.Dept$	$(\{t_2, t_3, t_4\}, \{t_2, t_3, t_4\})$
2	$t.StartDate < t'.StartDate$	$(\{t_2, t_3\}, \{t_4\})$
3	$t.Salary < t'.Salary$	$(\{t_3\}, \{t_4\})$

Refinement operator example

$$\neg(t.Dept = t'.Dept \wedge t.StartDate < t'.StartDate \wedge t.Salary < t'.Salary)$$

tid	ID	Name	Dept	StartDate	Salary	SID
t_1	100	C. Gardner	Sales	2012	3000	100
t_2	101	R. Geller	Research	2014	8000	102
t_3	102	D. Brown	Research	2014	6000	101
t_4	103	H. McCoy	Research	2015	8000	101

$\Downarrow$

Step	Refinement	Compressed Intermediates
1	$t.Dept = t'.Dept$	$(\{t_2, t_3, t_4\}, \{t_2, t_3, t_4\})$
2	$t.StartDate < t'.StartDate$	$(\{t_2, t_3\}, \{t_4\})$
3	$t.Salary < t'.Salary$	$(\{t_3\}, \{t_4\}) \rightarrow (t_3, t_4)$

Refinement algorithms of FACET

- Two phases:
  1. Read input and build auxiliary data structures (hash tables or permutation arrays);
  2. Iterate these structures to emit the results incrementally.

Equalities like t.A = t′.B

- Akin to hash-joins
- Some twists for tuple pair results
- Sketches to decide on build/probe sides
- Optimizations for reflexive intermediates (e.g., $(\{t_2, t_3, t_4\}, \{t_2, t_3, t_4\})$)

Non-Equalities like t.A ≠ t′.B

- Similar to equalities
- Bitwise `AndNot` to produce the results

Inequalities like t.A < t′.B

Three complementary approaches:

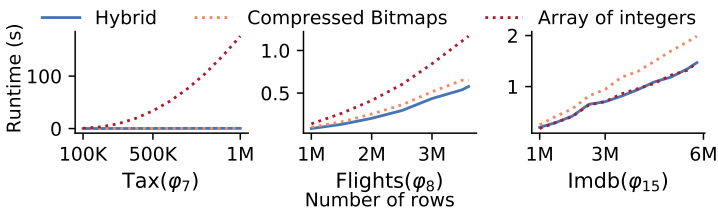
- IEJoin
  - Sorting + permutation arrays + marked bitmaps
  - Optimized for inequality pairs
- Hash-Sort-Merge (HSM)
  - Hashing + bitmap indexes + logical operations
  - Optimized for low-cardinality columns
- Binning-Hash-Sort-Merge (BHSM)
  - HSM + column values binning
  - Optimized for moderate-cardinality columns

Intermediate representation in FACET

- May change throughout the refinement pipeline
- Hybrid structures based on the computation pattern of each refinement algorithm
  - Equalities use simple fetches $\rightarrow$ Simple arrays
  - Non-equalities and inequalities use many logical operations $\rightarrow$ Compressed bitmaps

Intermediate representation in FACET

- May change throughout the refinement pipeline
- Hybrid structures based on the computation pattern of each refinement algorithm
  - Equalities use simple fetches → Simple arrays
  - Non-equalities and inequalities use many logical operations → Compressed bitmaps



Impact of the intermediate representation on runtime

Execution Planning in FACET

- The costs of refinements are strongly connected to:
  - Predicate structure
  - Distinct values of column or column combinations (cardinality)
    - We obtain accurate estimates with Sketch-Corrected Estimators

Predicate Order

– Cardinality as a proxy for costs

– Predicate Push Down

– GreedyHLL Algorithm

- Favors predicate pairs with small intermediates and low evaluation costs

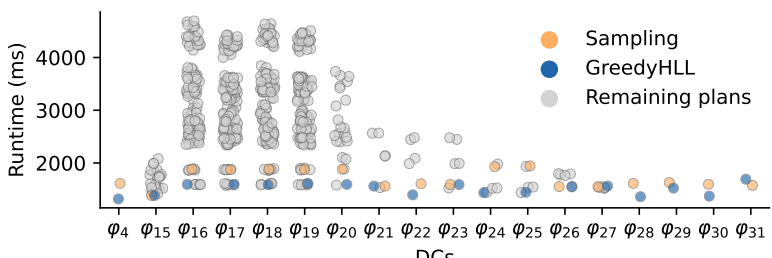
Selection of Inequality Algorithms

– Joint cardinality of the columns to decide between algorithms

– Ascending order of the column cardinalities when using HSM/BHSM

Execution Planning in FACET

- The costs of refinements are strongly connected to:
  - Predicate structure
  - Distinct values of column or column combinations (cardinality)
    - We obtain accurate estimates with Sketch-Corrected Estimators



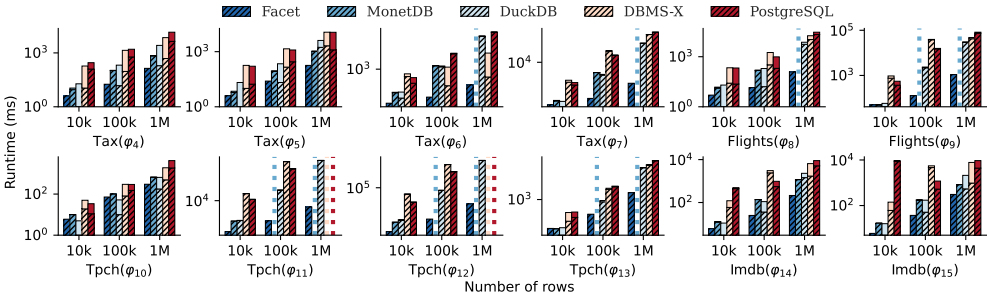
Runtime of FACET using GreedyHLL vs. runtime of FACET using all other possible plans.

Experimental Evaluation of FACET

Dataset	Number of rows	DC number	DC expression
Tax	10M	φ_4	$\neg$(t.AreaCode = t′.AreaCode $\wedge$ t.Phone = t′.Phone)
Tax	10M	φ_5	$\neg$(t.ZipCode = t′.ZipCode $\wedge$ t.City $\neq$ t′.City)
Tax	10M	φ_6	$\neg$(t.State = t′.State $\wedge$ t.HasChild = t′.HasChild $\wedge$ t.ChildExemp $\neq$ t′.ChildExemp)
Tax	10M	φ_7	$\neg$(t.State = t′.State $\wedge$ t.Salary > t′.Salary $\wedge$ t.Rate < t′.Rate)
Flights	3.6M	φ_8	$\neg$(t.Origin = t′.Dest $\wedge$ t.Dest = t′.Origin $\wedge$ t.Distance $\neq$ t′.Distance)
Flights	3.6M	φ_9	$\neg$(t.Orgin = t′.Origin $\wedge$ t.Dest = t′.Dest $\wedge$ t.Flights > t′.Flights $\wedge$ t.Passengers < t′.Passengers)
TPC-H	6M	φ_{10}	$\neg$(t.Customer = t′.Supplier $\wedge$ t.Supplier = t′.Customer)
TPC-H	6M	φ_{11}	$\neg$(t.Receiptdate $\geq$ t′.Shipdate $\wedge$ t.Shipdate $\leq$ t′.Receiptdate)
TPC-H	6M	φ_{12}	$\neg$(t.ExtendedPrice > t′.ExtendedPrice $\wedge$ t.Discount < t′.Discount)
TPC-H	6M	φ_{13}	$\neg$(t.Qty = t′.Qty $\wedge$ t.Tax = t′.Tax $\wedge$ t.ExtendedPrice > t′.ExtendedPrice $\wedge$ t.Discount < t′.Discount)
IMDB	2.5M	φ_{14}	$\neg$(t.Title = t′.Title $\wedge$ t.ProductionYear = t′.ProductionYear $\wedge$ t.Kind $\neq$ t′.Kind)
IMDB	5.8M	φ_{15}	$\neg$(t.Title = t′.Title $\wedge$ t.Name = t′.Name $\wedge$ t.CharName = t′.CharName $\wedge$ t.Role = t′.Role)

- Comparisons with PostgreSQL, MonetDB, DuckDB, and DBMS-X
  - Different query engines
  - All columns indexed + updated statistics

Runtime comparison of FACET with different DBMSs



* The dotted lines are cases when the respective DBMS either exceeded the time limit of four hours or ran out of memory.
* The dashed areas represent the violation detection time, whereas the solid areas represent the indexing construction time (only for the DBMSs)

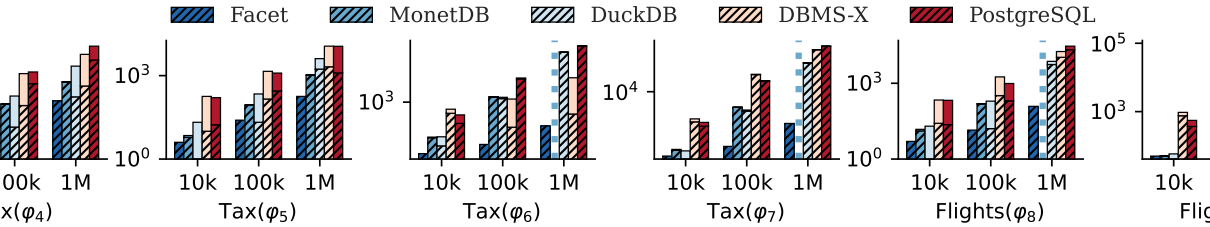
DBMSs

- Work well for DCs including only equalities (e.g., keys)
- Quadratic increase in runtime (DCs with inequalities)
- Varied query plans and performance between DBMSs
- Varied performance for different dataset scales

FACET

- Speedups of orders of magnitude for many DCs
  - FACET takes a few seconds to finish where the DBMSs often take a few hours or reach the time limit

Runtime comparison of FACET with different DBMSs



* The dotted lines are cases when the respective DBMS either exceeded the time limit of four hours or ran out of memory.
The dashed areas represent the violation detection time, whereas the solid areas represent the indexing construction time (only for the DBMSs)

DBMSs

- Work well for DCs including only equalities (e.g., keys)
- Quadratic increase in runtime (DCs with inequalities)
- Varied query plans and performance between DBMSs
- Varied performance for different dataset scales

FACET

- Speedups of orders of magnitude for many DCs
  - FACET takes a few seconds to finish where the DBMSs often take a few hours or reach the time limit

Updates in the DBMS Market

Range Joins in DuckDB

Richard Wesley
2022-05-27

TL;DR: DuckDB has fully parallelised range joins that can efficiently join millions of range predicates.

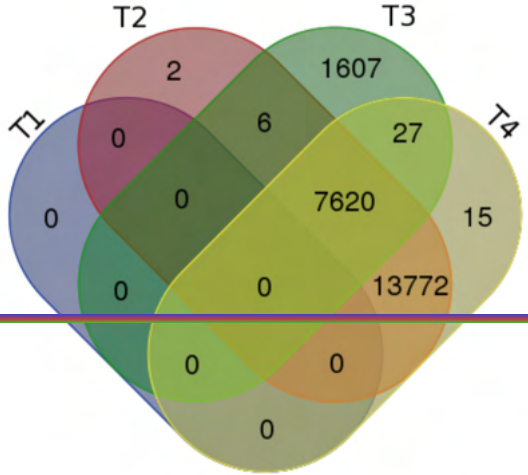
Range intersection joins are an important operation in areas such as temporal analytics, and occur when two inequality conditions are present in a join predicate. Database implementations often rely on slow `O(N^2)` algorithms that compare every pair of rows for these operations. Instead, DuckDB leverages its fast sorting logic to implement two highly optimized parallel join operators for these kinds of range predicates, resulting in 20-30x faster queries. With these operators, DuckDB can be used effectively in more time-series-oriented use cases.

Inequality Join (IEJoin)

For two range conditions (like the combat pay query), there are even faster algorithms available. We have recently added a new join called IEJoin, which sorts on two predicates to really speed things up.

https://duckdb.org/2022/05/27/iejoin.html

Different Error Detectors in Action

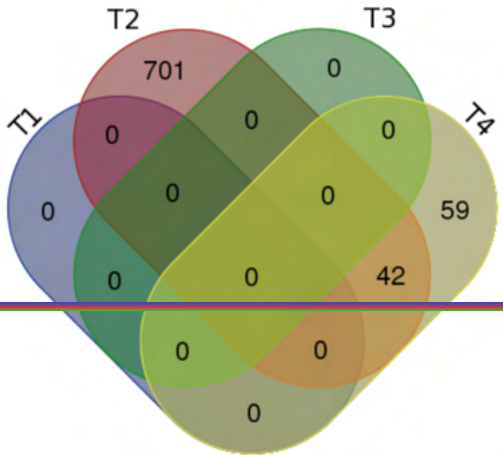


(a) Merck: 23,049 out of 27,208 errors

Overlaps of error types:
T1: Duplicates,
T2: Constraint Violations,
T3: Outliers,
T4: Pattern Violations

Abedjan et. al. Detecting data errors: where are we and what needs to be done?. *PVLDB* 2016

Different Error Detectors in Action

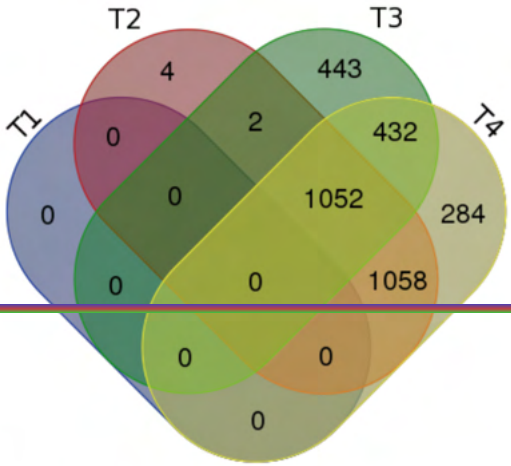


(b) Animal: 802 out of 1,394 errors

Overlaps of error types:
T1: Duplicates,
T2: Constraint Violations,
T3: Outliers,
T4: Pattern Violations

Abedjan et. al. Detecting data errors: where are we and what needs to be done?. *PVLDB* 2016

Different Error Detectors in Action



(c) Rayyan Bib: 3275 out of 3853 errors

Overlaps of error types:

T1: Duplicates,

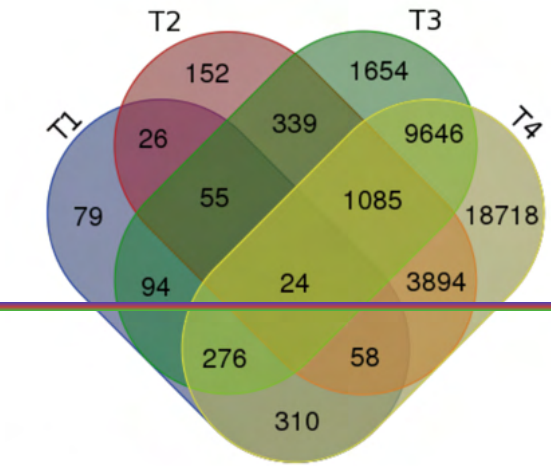
T2: Constraint Violations,

T3: Outliers,

T4: Pattern Violations

Abedjan et. al. Detecting data errors: where are we and what needs to be done?. *PVLDB* 2016

Different Error Detectors in Action

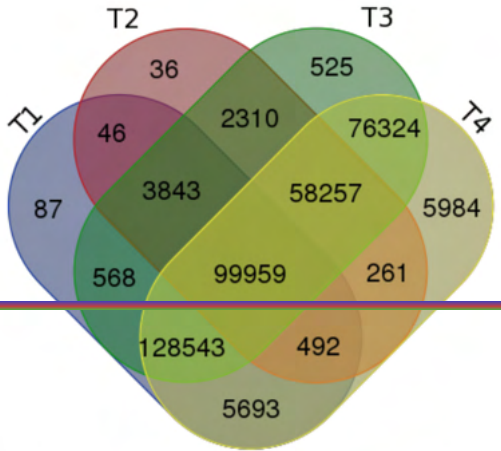


(d) MIT VPF: 36,410 out of 39,158 errors

Overlaps of error types:
T1: Duplicates,
T2: Constraint Violations,
T3: Outliers,
T4: Pattern Violations

Abedjan et. al. Detecting data errors: where are we and what needs to be done?. *PVLDB* 2016

Different Error Detectors in Action

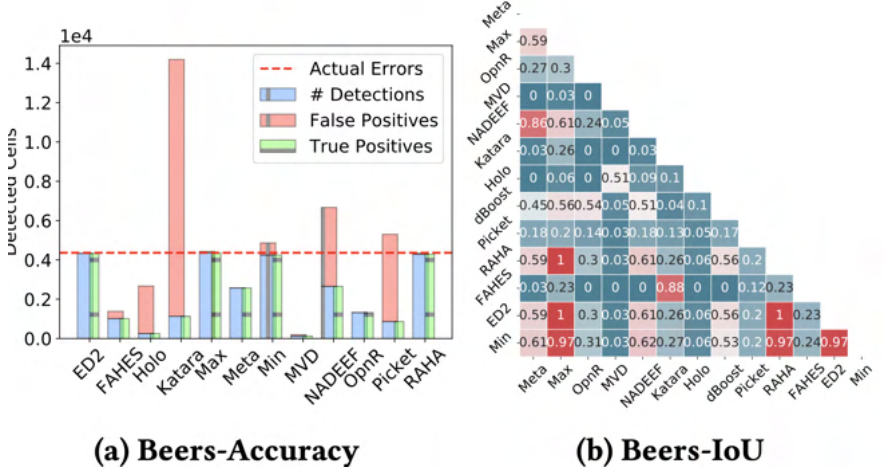


(e) BlackOak: 382,928 out of 383,003 errors

Overlaps of error types:
T1: Duplicates,
T2: Constraint Violations,
T3: Outliers,
T4: Pattern Violations

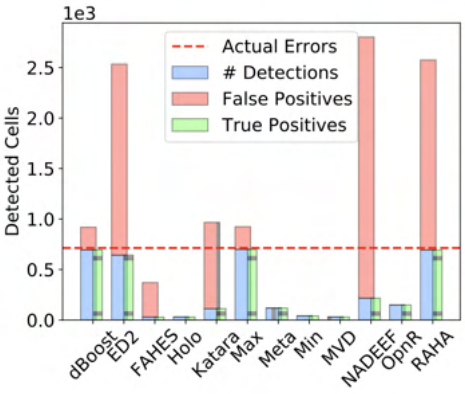
Abedjan et. al. Detecting data errors: where are we and what needs to be done?. *PVLDB* 2016

Different Error Detectors in Action

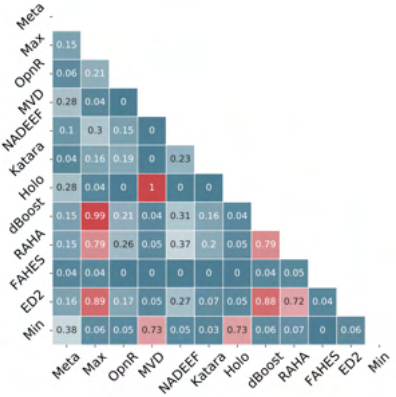


Abdelaal, Hammacher, and Schoning. REIN: A Comprehensive Benchmark Framework for Data Cleaning Methods in ML Pipelines. *EDBT* 2023

Different Error Detectors in Action



(k) Nasa-Accuracy



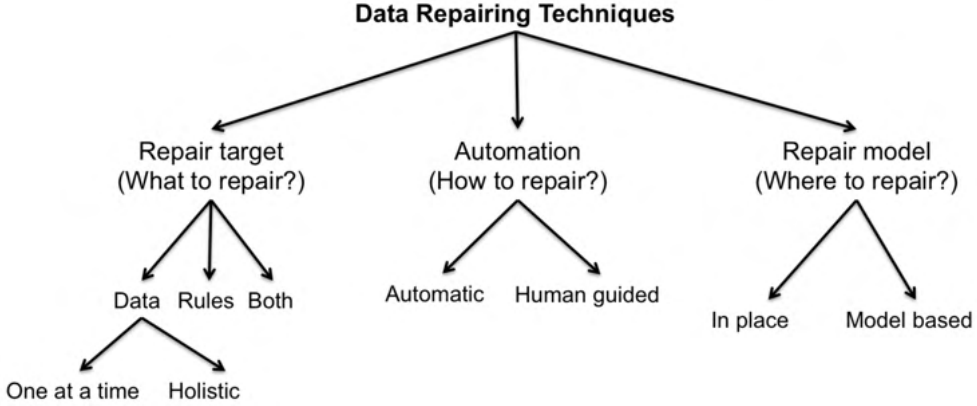
(l) Nasa-IoU

Abdelaal, Hammacher, and Schoning. REIN: A Comprehensive Benchmark Framework for Data Cleaning Methods in ML Pipelines. *EDBT* 2023

Outline

Data repairing

Given a database instance I and a set of data quality requirements, data repairing refers to the process of finding another database instance I' that conforms to these requirements.



Ilyas and Chu. Trends in Cleaning Relational Data: Consistency and Deduplication. *Foundations and Trend in Databases* 2012

Data repairing for constraints

- A **repair** of a database D_{dirty} relative to a set Σ of constraints is a database D_{rep} such that:
  - D_{rep} satisfies all the constraints in Σ; and
  - D_{rep} is close to D_{dirty}.
- A **repair model** indicates what kind of operations are allowed to modify the dirty database: e.g., tuple deletions, insertions, value modifications.
- A **cost function** is used to ensure that a repair minimally differs from the original database, e.g., edit distance, ...

Data Repair Example

Dirty Database

StdId	StdName
123	John
123	Steve
456	Anna
789	Geoff

Σ: Key constraint $StdId \rightarrow StdName$.

Repair Model: Tuple Deletion

Cost Function: Number of Deleted Tuples

Result: Two possible repairs:

1.

StdId	StdName
123	John
456	Anna
789	Geoff

2.

StdId	StdName
123	Steve
456	Anna
789	Geoff

Different Approaches to Data Repairing

- We have seen that a repair is not unique.
- When one wants to query repairs, one distinguishes between the following two approaches:

Consistent Query Answering

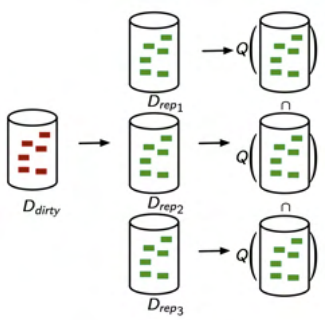
- Avoid selecting a repair.
- At query time only return query answers that are common to all repairs.

Data Repairing

- Select the best possible repair.
- Which is subsequently queried.

Consistent Query Answering

The problem of querying a database that is inconsistent, i.e., that fails to satisfy integrity constraints, in such a way that **the answers returned by the database are consistent** with those integrity constraints.



Challenge: How to compute certain answers without computing all repairs.

Leopoldo Bertossi. Database Repairing and Consistent Query Answering. *Synthesis Lectures on Data Management*. 2011

Data Repairing

Obtains a "clean" version of the database, which is then used for querying.

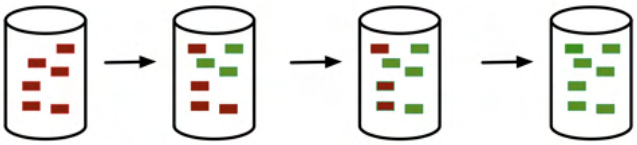
Types of Repair Methods

1. **Chase-based repair methods:**
   - Repair is obtained by "firing" constraints until all constraints are satisfied.
2. **Holistic repair methods:**
   - Repair is obtained by taking a "global" view of all errors involved.
3. **Probabilistic repair methods:**
   - Use probabilistic inference to find good repair, possibly not satisfying all constraints.

Chase-based Repair Methods

Decide how to repair based on a local manner:

- Repair $Error(\varphi, D_{\text{dirty}})$ for each constraint φ, one at a time.
- Keep doing so, until (hopefully) a repair is obtained.



This process is also known as the **chase**: one chases the constraints.

Geerts, Mecca, Papotti, and Santoro. The LLUNATIC data-cleaning framework. *VLDB* 2013

Chase-based data repair in Llunatic

SSN	Name	Phone	STR	CITY	#CC
222	L. Lennon	122-1874	null	SF	7842554
222	L. Lennon	102-111	Fry	SF	7842545
111	J. White	110-1000	Maple	NY	1010101

and a key constraint: SSN $\rightarrow$ Name, Phone, STR, CITY, #CC

In the chase, changes must be made to errors of φ such that φ is satisfied.

- But which changes?
  - Is the phone number of L. Lennon 122-1874 or 102111?
  - Is the street of L. Lennon Fry?
  - What is his credit card number?
  - Perhaps they are different Lennons? So SSN is incorrect?

Chase-based data repair in Llunatic

Collect extra information about the values in the database:

- Each cell is extended with a preference level.
- Preference levels are related by means of a partial order.

SSN	Name	Phone	STR	CITY	#CC
$\langle p_\mu, 222\rangle$	$\langle p_{c1}$, L. Lennon$\rangle$	$\langle p_{0.9}, 122-1874\rangle$	$\langle p_\perp$, null$\rangle$	$\langle p_{c6}$, SF$\rangle$	$\langle p_{c10}, 7842554\rangle$
$\langle p_\mu, 222\rangle$	$\langle p_{c2}$, L. Lennon$\rangle$	$\langle p_{0.1}, 102-111\rangle$	$\langle p_{c4}$, Fry$\rangle$	$\langle p_{c7}$, SF$\rangle$	$\langle p_{c11}, 7842545\rangle$
$\langle p_\mu, 111\rangle$	$\langle p_{c3}$, J. White$\rangle$	$\langle p_{1.0}, 110-1000\rangle$	$\langle p_{c5}$, Maple$\rangle$	$\langle p_{c9}$, NY$\rangle$	$\langle p_{c12}, 1010101\rangle$

and a key constraint: SSN $\rightarrow$ Name, Phone, STR, CITY, #CC

- p_{c1}–p_{c12}: no additional information.
- p_μ: user certified
- p_x: the confidence degree of x.
- $p_\perp$: a null value.

Chase-based data repair in Llunatic

SSN	Name	Phone	STR	CITY	#CC
$\langle p_\mu, 222\rangle$	$\langle p_{c1}$, L. Lennon$\rangle$	$\langle p_{0.9}, 122-1874\rangle$	$\langle p_\perp$, null$\rangle$	$\langle p_{c6}$, SF$\rangle$	$\langle p_{c10}, 7842554\rangle$
$\langle p_\mu, 222\rangle$	$\langle p_{c2}$, L. Lennon$\rangle$	$\langle p_{0.1}, 102-111\rangle$	$\langle p_{c4}$, Fry$\rangle$	$\langle p_{c7}$, SF$\rangle$	$\langle p_{c11}, 7842545\rangle$
$\langle p_\mu, 111\rangle$	$\langle p_{c3}$, J. White$\rangle$	$\langle p_{1.0}, 110-1000\rangle$	$\langle p_{c5}$, Maple$\rangle$	$\langle p_{c9}$, NY$\rangle$	$\langle p_{c12}, 1010101\rangle$

and a key constraint: SSN $\rightarrow$ Name, Phone, STR, CITY, #CC

Chasing with constraint means that relevant cells are merged in order to satisfy the constraints

- In each step, cells gather information in the form of preference levels and values.
- The result is a database with sets of preference level-attribute pairs.
- Termination occurs when cells grow in each step, and there is an upper bound on the information that can be collected.

Chase-based data repair in Llunatic

SSN	Name	Phone	STR	CITY	#CC
$\langle p_\mu, 222\rangle$	$\langle p_{c1}$, L. Lennon$\rangle$	$\langle p_{0.9}, 122-1874\rangle$	$\langle p_\perp$, null$\rangle$	$\langle p_{c6}$, SF$\rangle$	$\langle p_{c10}, 7842554\rangle$
$\langle p_\mu, 222\rangle$	$\langle p_{c2}$, L. Lennon$\rangle$	$\langle p_{0.1}, 102-111\rangle$	$\langle p_{c4}$, Fry$\rangle$	$\langle p_{c7}$, SF$\rangle$	$\langle p_{c11}, 7842545\rangle$
$\langle p_\mu, 111\rangle$	$\langle p_{c3}$, J. White$\rangle$	$\langle p_{1.0}, 110-1000\rangle$	$\langle p_{c5}$, Maple$\rangle$	$\langle p_{c9}$, NY$\rangle$	$\langle p_{c12}, 1010101\rangle$

and a key constraint: SSN $\rightarrow$ Name, Phone, STR, CITY, #CC

Preferred values:

- $\{\langle p_{c1}$, L. Lennon$\rangle, \langle p_{c2}$, L. Lennon$\rangle\}$: L. Lennon
- $\{\langle p_{0.1}, 102-111\rangle, \langle p_{0.9}, 122-1874\rangle\}$: $122-1874$
- $\{\langle p_\perp$, null$\rangle, \langle p_{c4}$, Fry$\rangle\}$: Fry
- $\{\langle p_{c6}$, SF$\rangle, \langle p_{c7}$, SF$\rangle\}$: SF
- $\{\langle p_c10, 7842554\rangle, \langle$pc11, 7842545$\rangle\}$: llun.

Chase-based data repair in Llunatic

Preferred values are put in cells: Repair.

SSN	Name	Phone	STR	CITY	#CC
222	L. Lennon	122-1874	Fry	SF	Llun
111	J. White	110-1000	Maple	NY	1010101

and a key constraint: SSN $\rightarrow$ Name, Phone, STR, CITY, #CC

A variety of preference levels encoding:

- User: Preference levels set by the user.
- Data from master data: Preference levels derived from the master data.
- Constants in constraints: Preference levels assigned to constants used in constraints.

Fine-grained control of the chase by:

- Using cost functions to only perform changes that minimize the cost function.
- Allowing left-hand side repairs, effectively disabling constraints.

Holistic data cleaning

Conflict graph-based repairing

1. Consider a set of denial constraints.
2. Encode all violations of constraints as a conflict hypergraph.
3. Find a minimal vertex cover of this hypergraph.
4. Collect information from the cover to resolve conflicts.
5. Repair the data accordingly; if new conflicts arise, repeat the process.

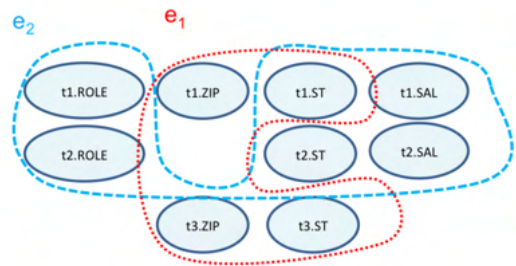
Chu, Ilyas, and Papotti. Holistic data cleaning: Putting violations into context.. *ICDE*. 2013

Holistic data cleaning

given the relation instance:

TID	FN	LN	ROLE	ZIP	ST	SAL
t_1	Anne	Nash	E	85376	NY	110
t_2	Mark	White	M	90012	NY	80
t_3	Mark	Lee	E	85376	AZ	75

gives the conflict hypergraph:



and two data quality rules:

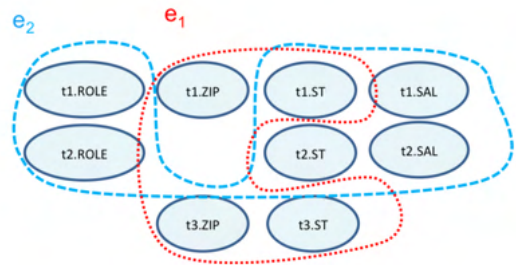
- φ_1: $\neg(\text{t.ZIP} = \text{t}'.\text{ZIP} \wedge \text{t.ST} \neq \text{t}'.\text{ST})$
- φ_2: $\neg(\text{t.ST} = \text{t}'.\text{ST} \wedge \text{t.ROLE} = M \wedge \text{t.Salary} < \text{t}'.\text{Salary})$

Holistic data cleaning

given the relation instance:

TID	FN	LN	ROLE	ZIP	ST	SAL
t_1	Anne	Nash	E	85376	NY	110
t_2	Mark	White	M	90012	NY	80
t_3	Mark	Lee	E	85376	AZ	75

gives the conflict hypergraph:



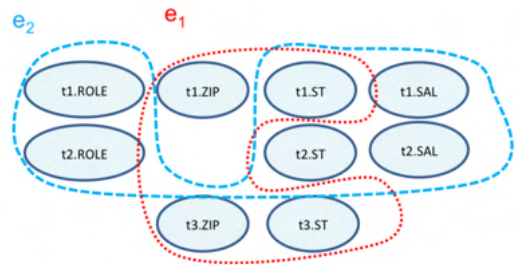
- $t1.ST$ is a minimal cover.
- A repair can be found by changing the value of nodes in the minimal cover.
  - change $t1.ST$ into $t3.ST$
  - change $t1.ST$ into something different than $t2.ST$
- This ensures the minimal number of changes required.

Holistic data cleaning

given the relation instance:

TID	FN	LN	ROLE	ZIP	ST	SAL
t_1	Anne	Nash	E	85376	NY	110
t_2	Mark	White	M	90012	NY	80
t_3	Mark	Lee	E	85376	AZ	75

gives the conflict hypergraph:



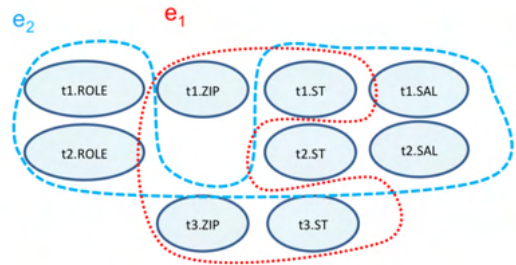
- In general, inspect constraints and identify constraints that would **eliminate all current hyperedges** (no previous conflicts) and sets of assignments satisfying the conditions.
- If it's not possible to find such a value, introduce a **fresh constant** different from anything else.

Holistic data cleaning

given the relation instance:

TID	FN	LN	ROLE	ZIP	ST	SAL
t_1	Anne	Nash	E	85376	NY	110
t_2	Mark	White	M	90012	NY	80
t_3	Mark	Lee	E	85376	AZ	75

gives the conflict hypergraph:



- If the change incurs new conflicts, repeat the whole procedure.
- Termination is guaranteed since in the worst case all values have new "fresh" constants.
- **Most constraint formalisms won't detect errors on datasets with only unique values.**

Probabilistic Repairing

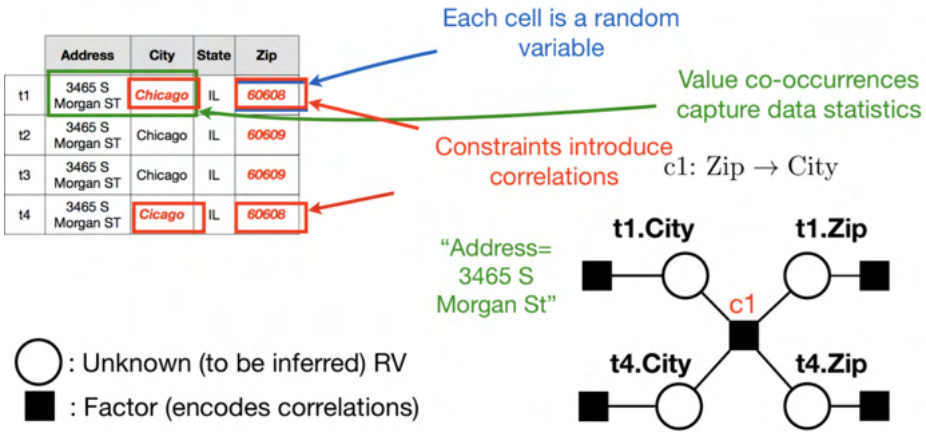
Key ideias:

1. Associate random variables to cells in instances
2. Describe constraints (and other background knowledge) and relax these with weights
3. Probability distribution on possible worlds
4. Most probable possible world=best repair

Rekatsinas, Chu, Ilyas, Ré. Holistic data repairs with probabilistic inference. *PVLDB*. 2017

Probabilistic Repairing

Holistic data cleaning framework: combines a variety of heterogeneous signals (e.g., integrity constraints, external knowledge, quantitative statistics)



	Address	City	State	Zip
t1	3465 S Morgan ST	Chicago	IL	60608
t2	3465 S Morgan ST	Chicago	IL	60609
t3	3465 S Morgan ST	Chicago	IL	60609
t4	3465 S Morgan ST	Cicago	IL	60608

Holoclean: Weighted Possible Worlds

– Let V be a set of random variables and α be an assignment of variables to domain values.
– Let D_α denote the corresponding possible world.
– Given a set of weighted constraints (Σ, w), the weight of D_α is defined as:

$$W(D_\alpha, \Sigma, w) = \sum_{\phi \in \Sigma} W(D_\alpha, \phi, w)$$

where, intuitively,

$$W(D_\alpha, \phi, w) = w \times f(\text{size of the set of satisfying tuples})$$

This leads to a probability distribution on possible worlds:

$$\text{Prob}_{\Sigma,w}(D_\alpha) = \frac{1}{Z} e^{W(D_\alpha, \Sigma, w)}$$

Each assignment of variables results in a **weighted possible world**.

Holoclean: Weighted Possible Worlds

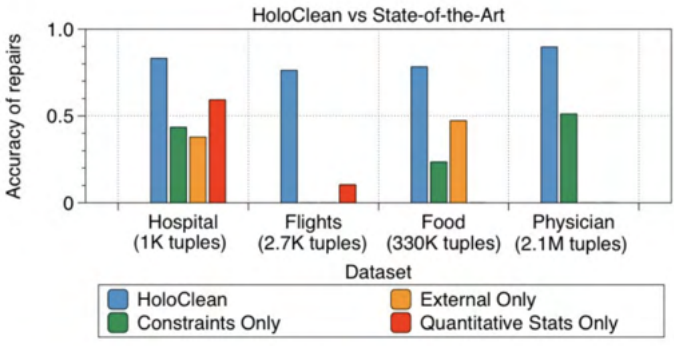
Problem statement:

- Find the assignment α such that $\text{Prob}_{\Sigma,w}(D_\alpha)$ is **maximized**.
- **Intractable problem:** search is over all possible worlds.
- HoloClean: lots of effort to make this scalable in practice.

Solving this problem:

- Efficient probabilistic inference techniques, factor graphs, etc.
- Sampling, approximation, etc.
- See `http://www.holoclean.io/`.

Data Repairing: HoloClean



HoloClean: our approach combining all signals and using inference
Holistic[Chu,2013]: state-of-the-art for constraints & minimality
KATARA[Chu,2015]: state-of-the-art for external data
SCARE[Yakout,2013]: state-of-the-art ML & qualitative statistics

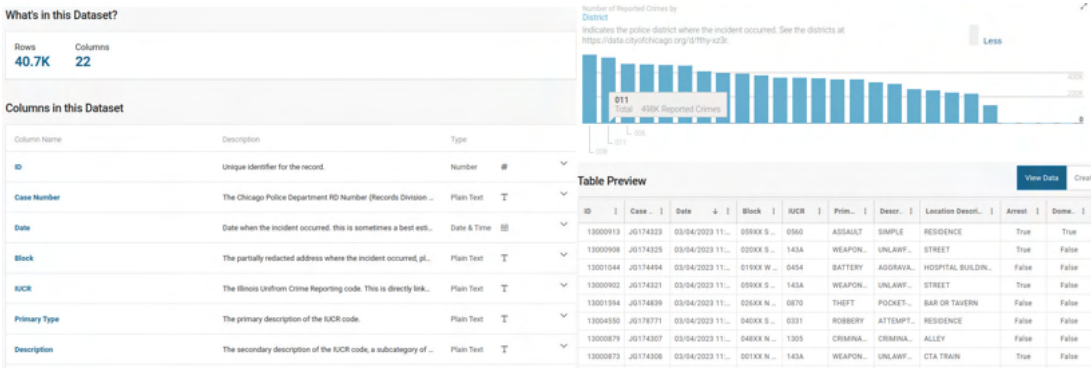
Outline

Data Profiling extracts **metadata from datasets**.

Such metadata can help in several tasks, for example:

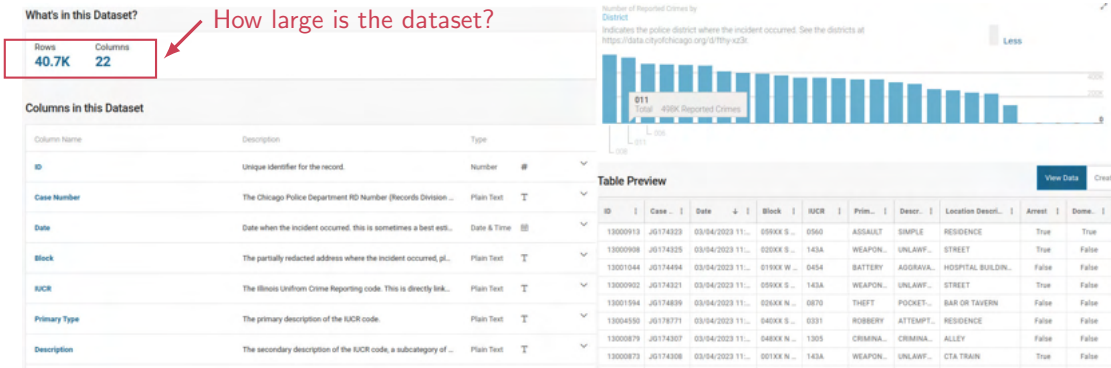
- **Data Cleaning**
- **Data Preparation**
- Query Optimization
- Data Integration
- Data Discovery
- Data analytics
- Scientific Data Management
- Data Privacy
- ...

Data Profiling: a Daily Duty of Data Practitioners



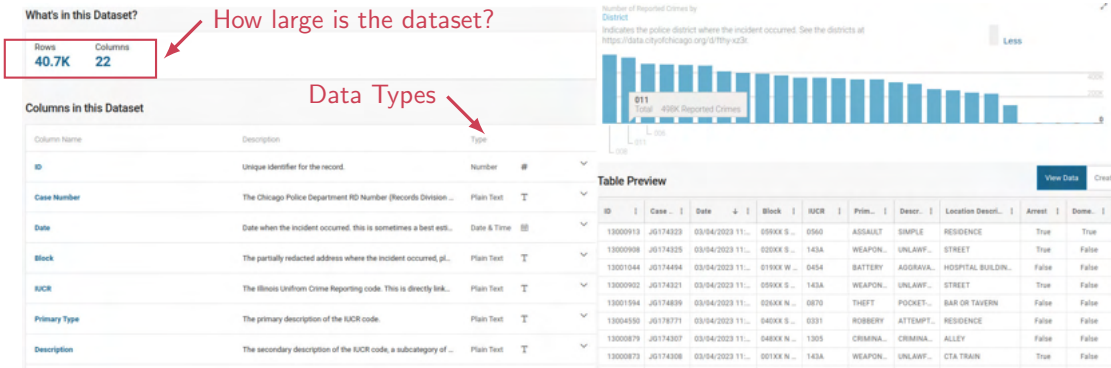
Tasks become easier if there is (at least some) metadata at hand.

Data Profiling: a Daily Duty of Data Practitioners



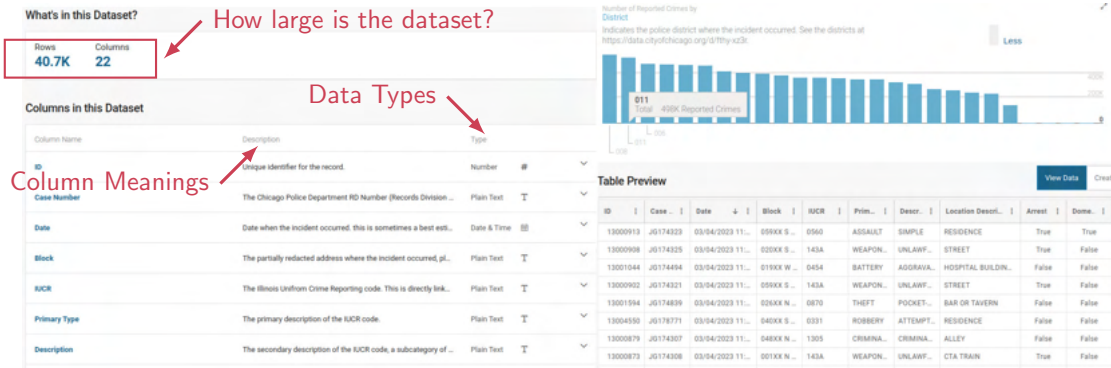
Tasks become easier if there is (at least some) metadata at hand.

Data Profiling: a Daily Duty of Data Practitioners



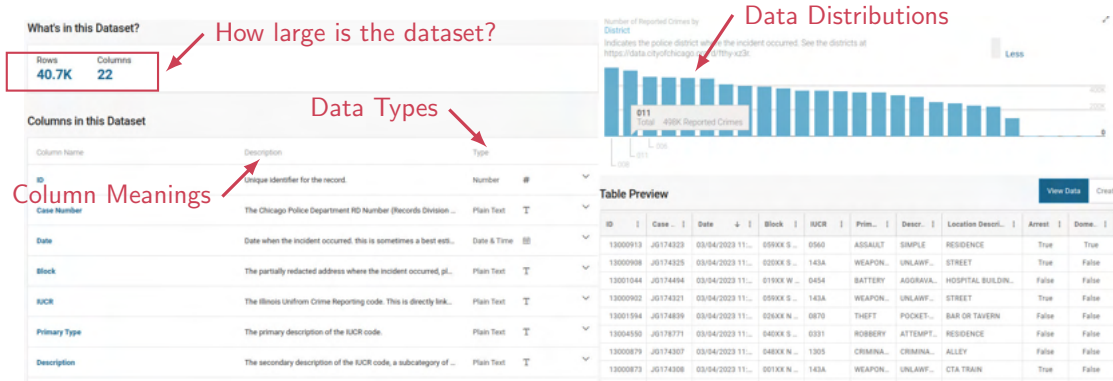
Tasks become easier if there is (at least some) metadata at hand.

Data Profiling: a Daily Duty of Data Practitioners



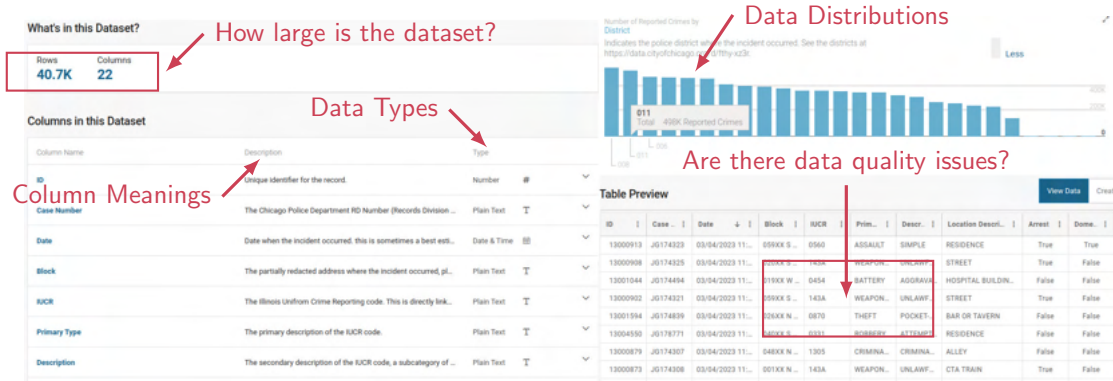
Tasks become easier if there is (at least some) metadata at hand.

Data Profiling: a Daily Duty of Data Practitioners



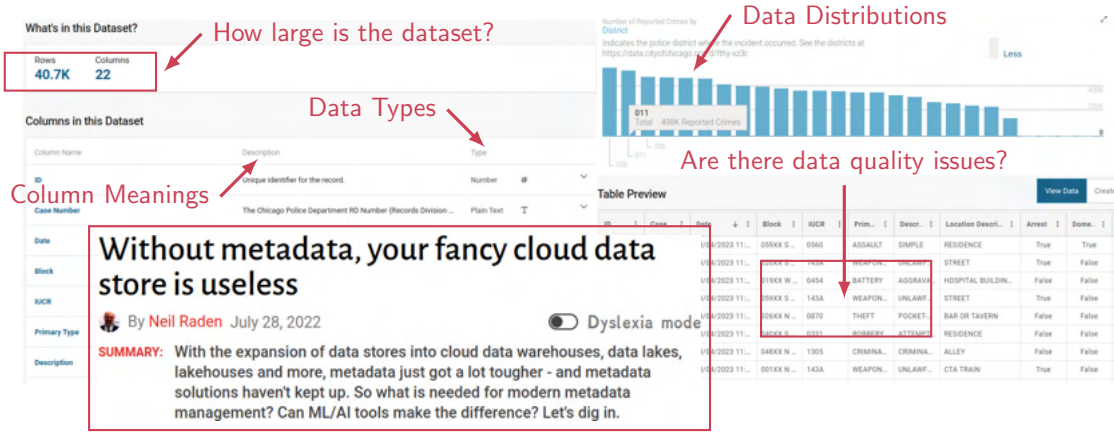
Tasks become easier if there is (at least some) metadata at hand.

Data Profiling: a Daily Duty of Data Practitioners



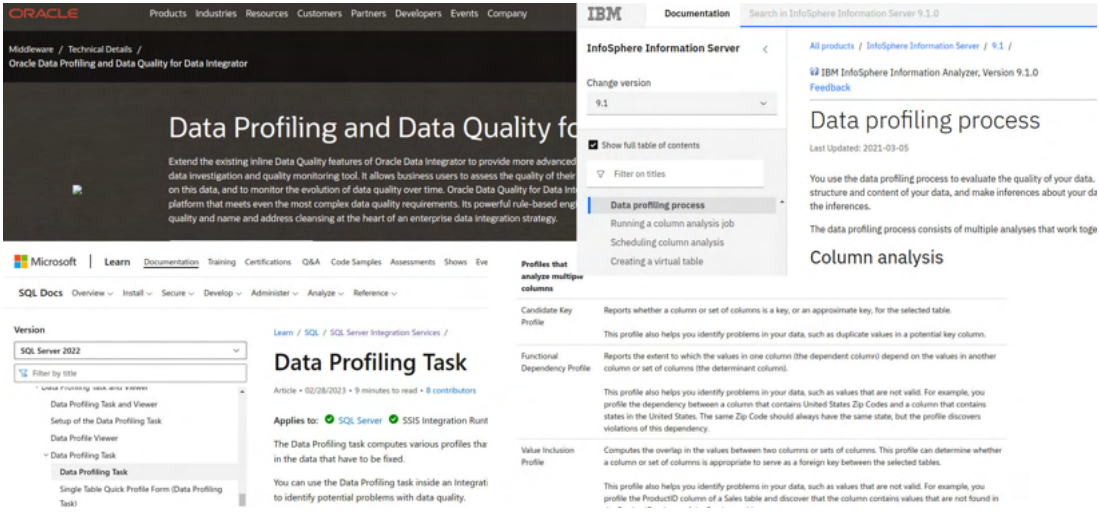
Tasks become easier if there is (at least some) metadata at hand.

Data Profiling: a Daily Duty of Data Practitioners



Tasks become easier if there is (at least some) metadata at hand.

Data Profiling Tools in Industry



Data Profiling Tools in Industry – Uses Cases

Tool	Statistics	Patterns	Data types	Uniques	Column dependency	Data dependency
Attacama, DQ Analyzer	✓	✓		✓		
IBM, InfoSphere Information Analyzer	✓	✓		✓	✓	
Microsoft SQL Server Data Profiling Task	✓	✓			✓	
Oracle Enterprise Data Quality	✓	✓				
Paxata Adaptive Preparation	✓					
SAP Information Steward	✓	✓	✓		✓	
Splunk Enterprise/Hunk		✓				✓
Talend Data Profiler	✓	✓			✓	
Trifacta	✓	✓	✓			
Tamr	✓			✓		
OpenRefine	✓	✓	✓			

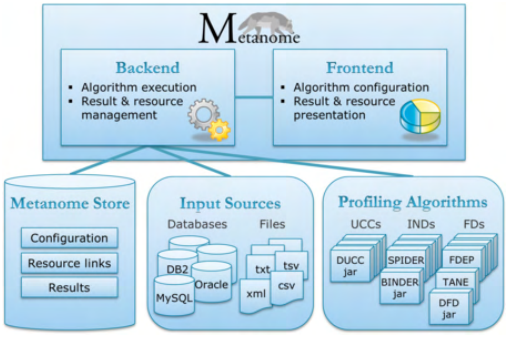
Restricted data types

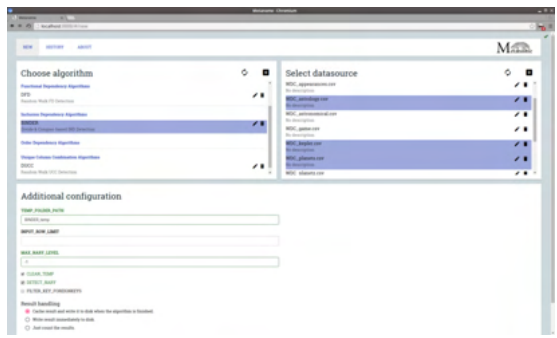
Restricted number of columns

Data Profiling Tools in Industry – Shortcomings

- Usability
- No single tool covers all metadata types
- No management of large metadata sets
  - No metadata summarizing
  - No metadata ranking based on relevance
- Focus on "easy" problems:
  - Statistics
  - Single column or "few" column dependencies
  - "Checking" vs. "discovery"
- SQL instead of optimized algorithms
  - Many queries / no early abort

Data Profiling Tools in Research





Open source framework, tool plus many algorithms:
www.metanome.de

Data Profiling Tools in Research – Uses Cases

Tool	Main purpose	Statistics	Patterns	Data types	Uniques	Dependencies	Data Mining
Bellmann	Data quality browser	✓			✓		
Potter's Wheel	ETL tool	✓	✓				
Data Auditor	Rule discovery						
RuleMiner	Dependency discovery					✓	
MADLib	Machine learning	✓				✓	
Metanome	Data profiling	✓			✓	✓	
ProLOD++	Profiling and Mining	✓	✓		✓	✓	✓

Data Profiling Tools in Research – Shortcomings

- Usability
- No single tool covers all metadata types (**Yet**)
- No management of large metadata sets
  - No metadata summarizing
  - ~~No metadata ranking based on relevance~~
- ~~Focus on "easy" problems:~~
  - ~~Statistics~~
  - ~~Single column or "few" column dependencies~~
  - ~~"Checking" vs. "discovery"~~
- ~~SQL instead of optimized algorithms~~
  - ~~Many queries / no early abort~~

Data Profiling Tasks

Single-Column Tasks

- Statistics
- Cardinalities
- Patterns
- Data Types
- Distributions
- Domains
- ...

Multi-Column Tasks (Much Harder)

- Uniqueness (UCCs)
- Inclusion dependencies (INDs)
- Functional dependencies (FDs)
- Order dependencies (ODs)
- Denial constraints (DCs)
- ...

Data profiling refers to the activity of creating small but informative summaries of a database.

"Ted Johnson, Encyclopedia of Database Systems"

Single-Column Example – Cardinality Estimation

Problem: Find the **number of distinct values** in a column.

Many applications:

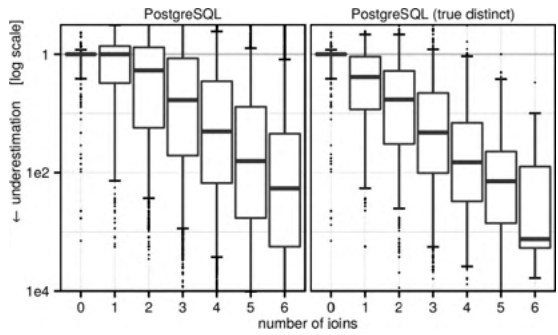
- **Query processing;**
- **Query optimization;**
- Network monitoring;
- Data streams;
- Search engines;
- Online data mining.

Single-Column Example – Cardinality Estimation

Problem: Find the **number of distinct values** in a column.

Many applications:

- **Query processing;**
- **Query optimization;**
- Network monitoring;
- Data streams;
- Search engines;
- Online data mining.



PostgreSQL cardinality estimates based on the default distinct count estimates **(Left)**, and the true distinct counts **(Right)**.

How Good Are Query Optimizers, Really? Leis et al. PVLDB, 2015.

Single-Column Example – Cardinality Estimation

Exact Solution: Bitmaps, Sorting or Hashing

- ✓ Perfect accuracy;
- ✘ Expensive in both size (memory) and runtime;
- ✘ Impractical for current big datasets.

Estimation Solution: Sampling, Sketches

- ✓ High accuracy;
- ✓ Low costs (memory and runtime);
- ✓ Practical for very large datasets.

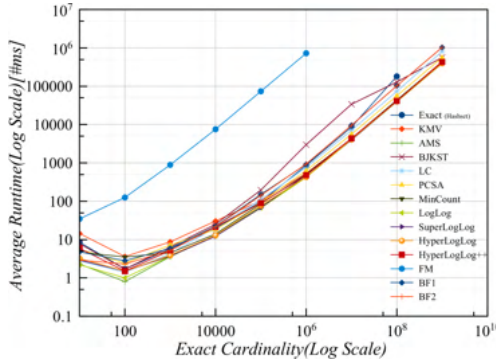
Single-Column Example – Cardinality Estimation

Exact Solution: Bitmaps, Sorting or Hashing

- ✓ Perfect accuracy;
- ✘ Expensive in both size (memory) and runtime;
- ✘ Impractical for current big datasets.

Estimation Solution: Sampling, Sketches

- ✓ High accuracy;
- ✓ Low costs (memory and runtime);
- ✓ Practical for very large datasets.



Runtime behavior of the twelve cardinality estimation algorithms on synthetic and real-world datasets.

Cardinality Estimation: An Experimental Survey. Harmouch and Naumann. PVLDB, 2017.

Multi-Column – Example of Problems

User Perspective : Find more **features** for my model.
Data Profiling Perspective: Find **tables** that can be joined with a query table.

Multi-Column – Example of Problems

User Perspective : Find more **features** for my model.
Data Profiling Perspective: Find **tables** that can be joined with a query table.

User Perspective : Find the **data errors** in my dataset.
Data Profiling Perspective: Find **data quality rules** that identify data errors.

Multi-Column – Example of Problems

User Perspective : Find more **features** for my model.
Data Profiling Perspective: Find **tables** that can be joined with a query table.

User Perspective : Find the **data errors** in my dataset.
Data Profiling Perspective: Find **data quality rules** that identify data errors.

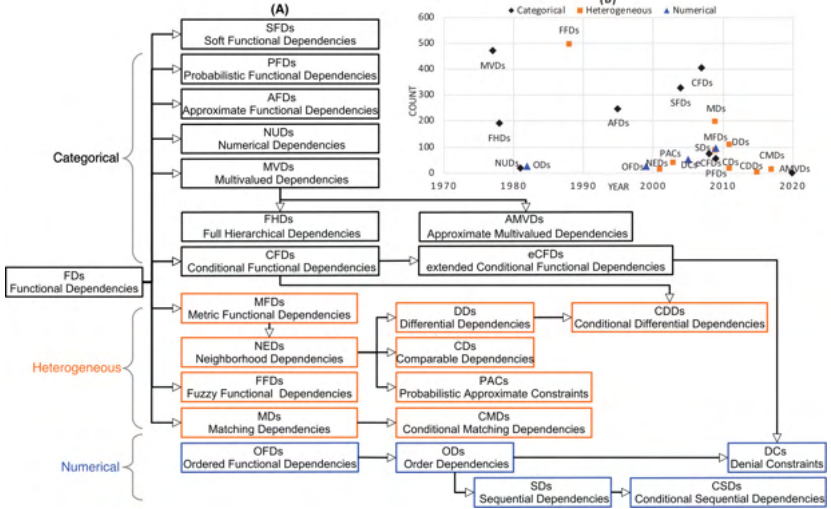
User Perspective : ...
Data Profiling Perspective: ...
... **many other examples (not covered in this talk).**

Great solutions based on:

Data Dependencies

Extending the Relational Model Since 1970.

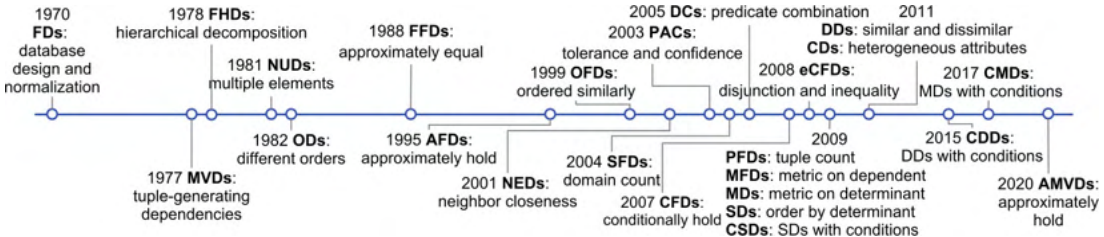
Data Dependencies Soup



A (not complete) Taxonomy.

Data Dependencies Extended for Variety and Veracity: A Family Tree. Song et al.TKDE, 2022.

Data Dependencies Soup



A (not complete) Timeline.

Data Dependencies Extended for Variety and Veracity: A Family Tree. Song et al.TKDE, 2022.

Data Dependencies Soup

Applications of Data Dependencies

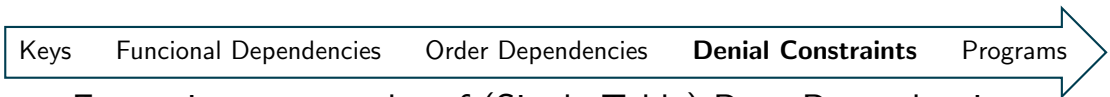
Applications	Categorical	Heterogeneous	Numerical
Violation detection	FDs [7], PFDs [104], CFDs [25], [40], eCFDs [14]	MFDs [64], CDs [92], CDDs [66], PACs [63]	ODs [28], DCs [8], [9], SDs [48], CSDs [48]
Data repairing	FDs [7], CFDs [25], eCFDs [14], MVDs [80]	NEDs [4], DDs [96], [95], [93], [94], CDDs [66], MDs [38], [41], CMDs[110]	DCs [70], [20], [98], ODs [28]
Query optimization	SFDs [55], [60], AFDs [111], NUDs [22], AMVDs[59]	DDs [86], CDs [92], PACs [63], FFDs [56]	ODs [100], [28]
Consistent query answering	FDs [7]	OFDs [75], DCs [8], [9]	
Data deduplication	CFDs [40]	DDs [86], CDs [92], FFDs [13], MDs [38], [41], CMDs [110]	
Data partition		DDs [86], MDs [37]	
Schema normalization	FDs [24], PFDs [104], MVDs [30], FHDs [27], [52]		
Model fairness	MVDs [80]		

Data dependencies have been proposed for various reasons.

Data Dependencies Extended for Variety and Veracity: A Family Tree. Song et al.TKDE, 2022.

Data Dependencies Soup

In summary ...



Expressive power order of (Single Table) Data Dependencies

Uniqueness and Keys

- **Unique Column Combination (UCC)**
  - Sets of columns that contain only unique values.
  - Minimality: no column subset is unique
- Key candidates: UCCs without `NULL`s
- **Primary Keys**: up to experts to decide
  - Being a UCC is prerequisite

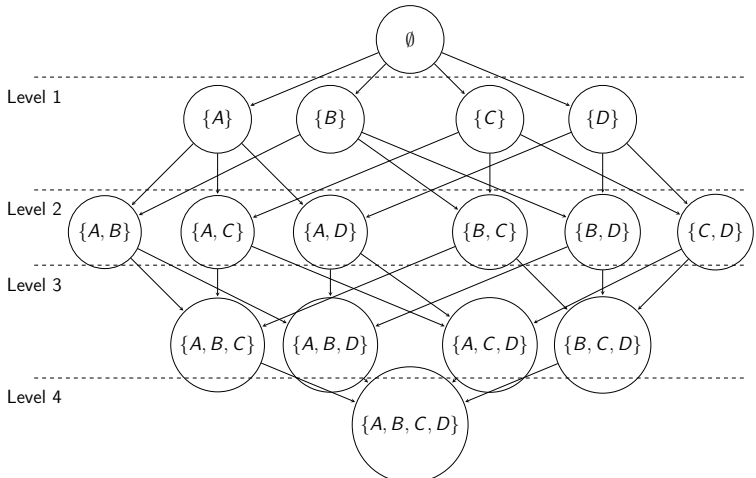
ID (A)	**Name (B)**	**Phone (C)**	**City (D)**
101	Chuck Norris	123-4567	Houston
102	Donald D.	987-6543	New York
103	Donald D.	555-1212	Chicago
104	Captain Hook	222-3333	Houston
105	Mr. Bean	444-5555	Seattle

Minimal UCCs:

- {**ID**}, {Phone}, {Name, City}

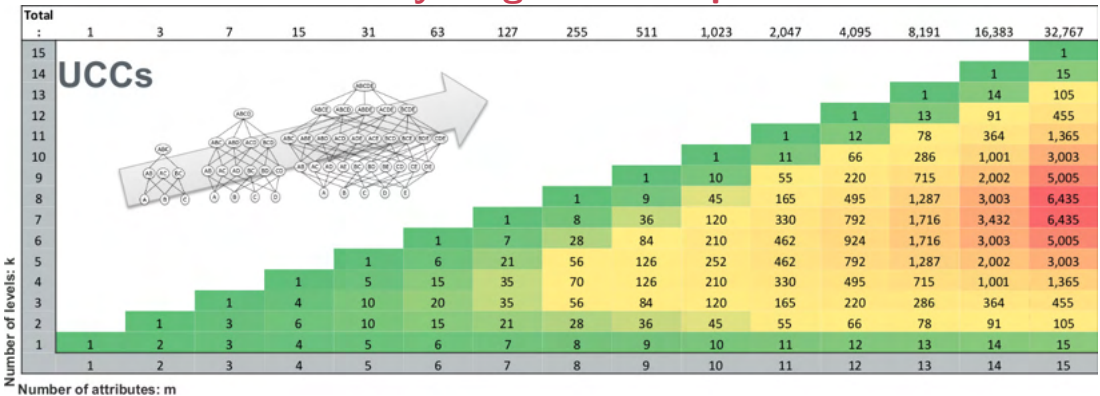
UCC Discovery – The "Easier" Problem

A Very Large Search Space



UCC Discovery – The "Easier" Problem

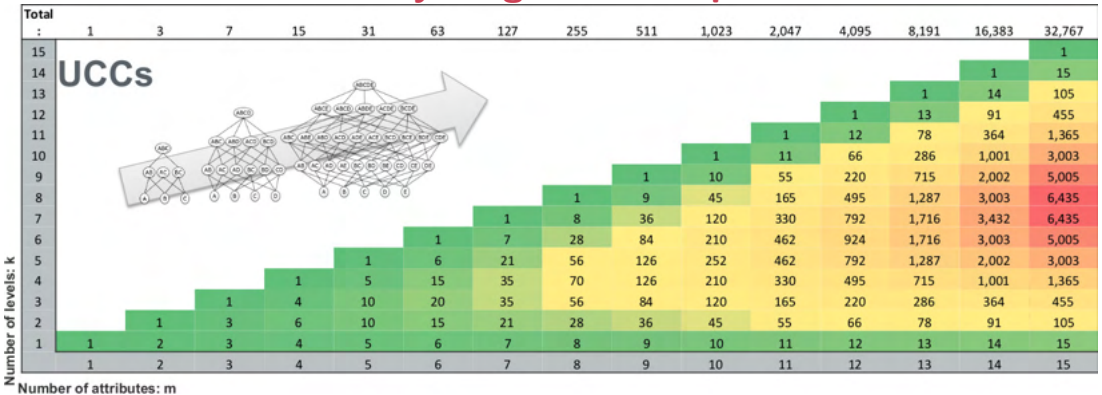
A Very Large Search Space



:	1	3	7	15	31	63	127	255	511	1,023	2,047	4,095	8,191	16,383	32,767
15															1
14														1	15
13													1	14	105
12												1	13	91	455
11											1	12	78	364	1,365
10										1	11	66	286	1,001	3,003
9									1	10	55	220	715	2,002	5,005
8								1	9	45	165	495	1,287	3,003	6,435
7							1	8	36	120	330	792	1,716	3,432	6,435
6						1	7	28	84	210	462	924	1,716	3,003	5,005
5					1	6	21	56	126	252	462	792	1,287	2,002	3,003
4				1	5	15	35	70	126	210	330	495	715	1,001	1,365
3			1	4	10	20	35	56	84	120	165	220	286	364	455
2		1	3	6	10	15	21	28	36	45	55	66	78	91	105
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

UCC Discovery – The "Easier" Problem

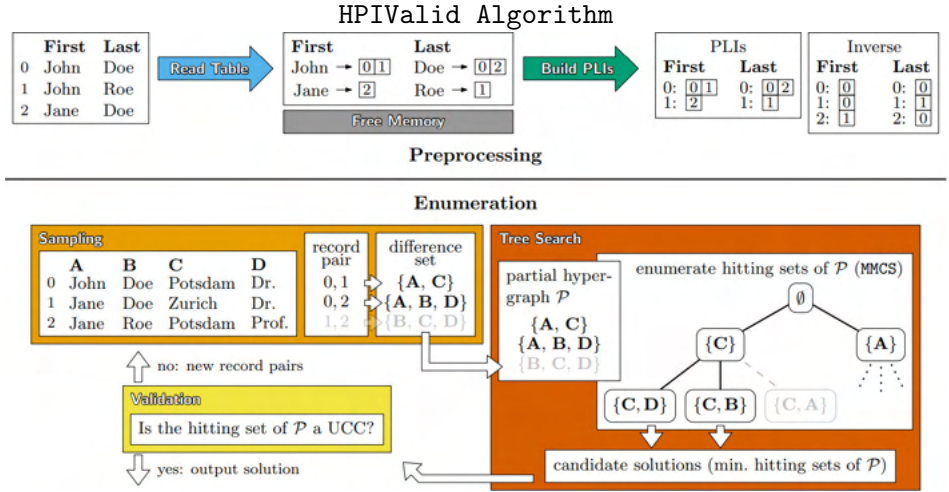
A Very Large Search Space



:	1	3	7	15	31	63	127	255	511	1,023	2,047	4,095	8,191	16,383	32,767
15															1
14														1	15
13													1	14	105
12												1	13	91	455
11											1	12	78	364	1,365
10										1	11	66	286	1,001	3,003
9									1	10	55	220	715	2,002	5,005
8								1	9	45	165	495	1,287	3,003	6,435
7							1	8	36	120	330	792	1,716	3,432	6,435
6						1	7	28	84	210	462	924	1,716	3,003	5,005
5					1	6	21	56	126	252	462	792	1,287	2,002	3,003
4				1	5	15	35	70	126	210	330	495	715	1,001	1,365
3			1	4	10	20	35	56	84	120	165	220	286	364	455
2		1	3	6	10	15	21	28	36	45	55	66	78	91	105
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

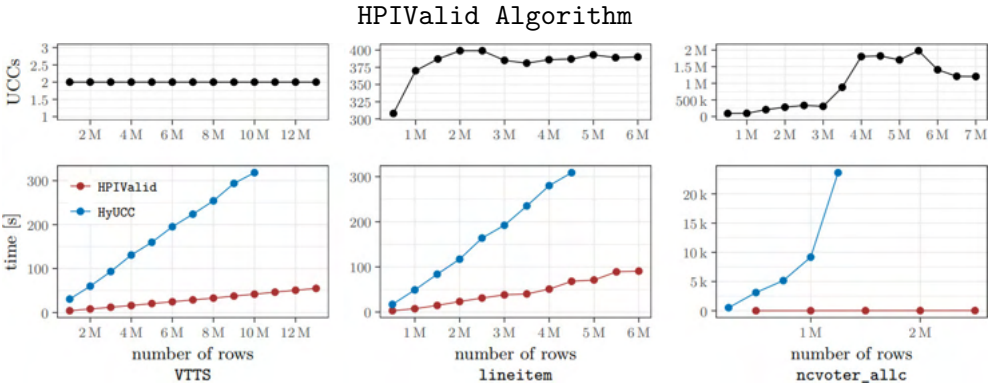
Fortunately, data profiling provides smart algorithms that avoid the exponential enumeration and validation trap.

UCC Discovery – The "Easier" Problem



Hitting Set Enumeration with Partial Information for Unique Column Combination Discovery. Birnick et al. PVLDB, 2020.

UCC Discovery – The “Easier” Problem



Discovers the UCCs from tables with millions of records in a matter of seconds/a few minutes.

Hitting Set Enumeration with Partial Information for Unique Column Combination Discovery. Birnick et al. PVLDB, 2020.

Inclusion Dependencies (INDs)

INDs are statements about a relational dataset indicating that all values of a certain attribute-combination are also contained in the values of another attribute-combination.

- **Unary**
  - $R[A] \subseteq S[B]$
- **n-ary INDs**
  - $R[ABC] \subseteq S[DEF]$

Showings

Screen	Movie	Length
1	Titanic	194
2	Titanic	194
3	Shrek	90
1	Ben-Hur	212

Movies

Title	Stars	Time
Titanic	7.8	194
Shrek	7.9	90
Ben-Hur	8.1	212
Gladiator	8.5	155

$$Showings[\text{Movie}] \subseteq Movies[\text{Title}]$$

$$Showings[\text{Length}] \subseteq Movies[\text{Time}]$$

Inclusion Dependencies (INDs)

INDs are statements about a relational dataset indicating that all values of a certain attribute-combination are also contained in the values of another attribute-combination.

- **Unary**
  - $R[A] \subseteq S[B]$
- **n-ary INDs**
  - $R[ABC] \subseteq S[DEF]$

Showings

Screen	Movie	Length
1	Titanic	194
2	Titanic	194
3	Shrek	90
1	Ben-Hur	212

Movies

Title	Stars	Time
Titanic	7.8	194
Shrek	7.9	90
Ben-Hur	8.1	212
Gladiator	8.5	155

$Showings[\text{Movie}] \subseteq Movies[\text{Title}]$

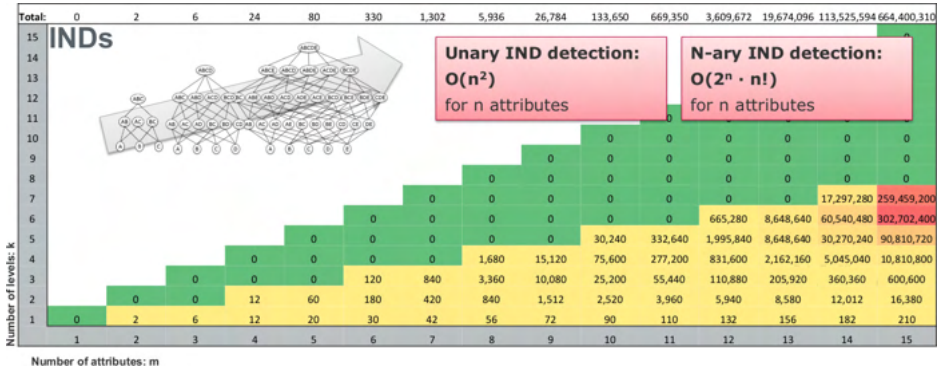
$Showings[\text{Length}] \subseteq Movies[\text{Time}]$

Discovering INDs can help with the problem:

- **User Perspective :** Find more **features** for my model.
- **Data Profiling Perspective:** Find **tables** that can be joined with a query table.

IND Discovery

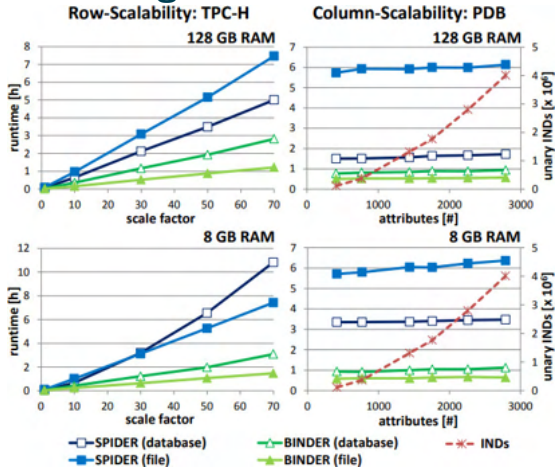
Again, a Very Large Search Space



Number of levels: k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Total:	0	2	6	24	80	330	1,302	5,936	26,784	133,650	669,350	3,609,672	19,674,096	113,525,594	664,400,310
15															0
14															0
13															0
12															0
11															0
10										0	0	0	0	0	0
9									0	0	0	0	0	0	0
8								0	0	0	0	0	0	0	0
7							0	0	0	0	0	0	0	17,297,280	259,459,200
6						0	0	0	0	0	0	665,280	8,648,640	60,540,480	302,702,400
5					0	0	0	0	0	30,240	332,640	1,995,840	8,648,640	30,270,240	90,810,720
4				0	0	0	0	1,680	15,120	75,600	277,200	831,600	2,162,160	5,045,040	10,810,800
3			0	0	0	120	840	3,360	10,080	25,200	55,440	110,880	205,920	360,360	600,600
2		0	0	12	60	180	420	840	1,512	2,520	3,960	5,940	8,580	12,012	16,380
1	0	2	6	12	20	30	42	56	72	90	110	132	156	182	210

IND Discovery

Again, fast algorithms have been devised.



Fantastic DCs and Where to Find Them

Denial Constraints (DCs): sets of **predicates that cannot be true together**.

– Generalize many other dependency types (UCCs, FDs, ODs, CFDs, etc)

	Name	Phone	Position	Salary	Hired
t_0	W. Jones	202-222	Developer	$2.000	2012
t_1	B. Jones	202-222	Developer	$3.000	2010
t_2	J. Miller	202-333	Developer	$4.000	2010
t_3	D. Miller	202-333	DBA	$8.000	2010
t_4	W. Jones	202-555	DBA	$7.000	2010
t_5	W. Jones	202-222	Developer	$1.000	2012

Exact DCs hold in the entire data:

$$\neg(t_x.\text{Position} = t_y.\text{Position} \wedge t_x.\text{Hired} < t_y.\text{Hired} \wedge t_x.\text{Salary} < t_y.\text{Salary})$$

Approximate DCs and Where to Find Them

- Often, acquiring 100% correct data to mine DCs is not possible;
- We can still find **approximate DCs**
  - which allow "a few" violations

	Name	Phone	Position	Salary	Hired
t_0	W. Jones	202-222	Developer	\$2.000	2012
t_1	B. Jones	202-222	Developer	\$3.000	2010
t_2	J. Miller	202-333	Developer	\$4.000	2010
t_3	D. Miller	202-333	DBA	\$8.000	2010
t_4	W. Jones	202-555	DBA	\$7.000	2010
t_5	W. Jones	202-222	Developer	\$1.000	2012

$$\neg(t_x.\text{Name} = t_y.\text{Name} \wedge t_x.\text{Phone} = t_y.\text{Phone})$$

Discovering DCs is challenging

A table with |R| columns:

Name	Dept	...
Ana	Research	...
Carlos	Sales	...
...	...	...

Generates |P| predicates:

$p_1 : t.Name = t'.Name$
$p_2 : t.Name \neq t'.Name$
$p_3 : t.Dept = t'.Dept$
$p_4 : t.Dept \neq t'.Dept$
...

... which can include:

- Operators: $\{=, \neq, <, \leq, >, \geq\}$
- Comparison over columns and column pairs (e.g., $t.A = t'.B$)
- Constants (not covered here)

Discovering DCs is challenging

A table with |R| columns:

Name	Dept	...
Ana	Research	...
Carlos	Sales	...
...	...	...

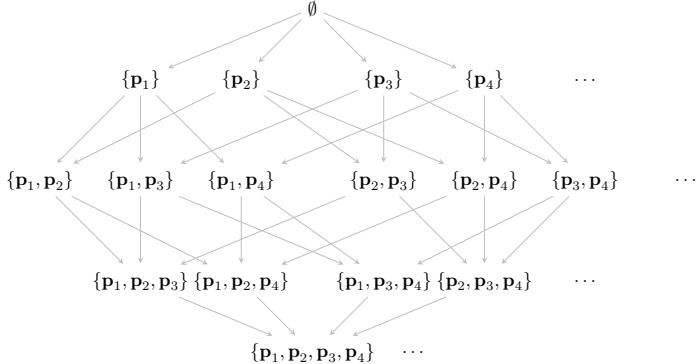
Generates |P| predicates:

$p_1 : t.Name = t'.Name$
$p_2 : t.Name \neq t'.Name$
$p_3 : t.Dept = t'.Dept$
$p_4 : t.Dept \neq t'.Dept$
...

... which can include:

- Operators: $\{=, \neq, <, \leq, >, \geq\}$
- Comparison over columns and column pairs (e.g., $t.A = t'.B$)
- Constants (not covered here)

A Very Large Search Space!



- $2^{|P|}$ DC candidates ($|P| \gg |R|$).
- Checking candidates with complex predicates (e.g., ranges) is computationally expensive.

The three-phase DC Discovery Framework

1. Predicate space building → straightforward
   - Check data types and shared values on columns to define a predicate space P.

2. Evidence set building → **quadratic in the number of records**
   - Evidence set E → data structure used to validate DC candidates quickly.
   - Compute the evidence $e_{t,t'} = \{p \mid p \in P, t, t' \models p\}$ for all t, t' of the input.

3. DC enumeration → **exponential in the number of predicates**
   - Equivalent to enumerating all (minimal) predicate sets $\{p_1 \wedge \ldots \wedge p_m\}$ such that $\nexists e \in E\colon \{p_1 \wedge \ldots \wedge p_m\} \subseteq e$.
   - Equivalent to minimal cover search/hitting set enumeration.

Framework proposed in "Chu, Ilyas, and Papotti. Discovering denial constraints. *PVLDB* 2013".

- More efficient algorithms for phases 1 and 2 in subsequent works.

Contributions of this Work

Evidence set building

2.

- Novel intermediate representations;
- Column indexes and algorithms specialized for the complex DC predicates;
- Pipeline approach that runs in parallel threads;

DC enumeration

3.

- Techniques that integrate and speedup all previous DC enumeration algorithms:
  - Inverted index;
  - Pruning strategies;
  - Lighter validation scheme;
  - Parallel execution;

Pena, Porto, and Naumann

Fast Algorithms for Denial Constraint Discovery

PVLDB 2022

☞ **Next, we discuss only a few of these. Please, check out the paper for details.**

Evidence Contexts

	ID	Name	Salary
t_1	#1	Caruso	10 000
t_2	#2	Zhang	5500
t_3	#3	Schneider	6000
t_4	#4	Smith	11 000
t_5	#5	Caruso	6000

$p_1 : t.ID = t'.ID$
$p_2 : t.ID \neq t'.ID$
$p_3 : t.Name = t'.Name$
$p_4 : t.Name \neq t'.Name$
$p_5 : t.Salary = t'.Salary$
$p_6 : t.Salary \neq t'.Salary$
$p_7 : t.Salary < t'.Salary$
$p_8 : t.Salary \leq t'.Salary$
$p_9 : t.Salary > t'.Salary$
$p_{10} : t.Salary \geq t'.Salary$

Evidence Contexts

	ID	Name	Salary
t_1	#1	Caruso	10 000
t_2	#2	Zhang	5500
t_3	#3	Schneider	6000
t_4	#4	Smith	11 000
t_5	#5	Caruso	6000

$p_1 : t.ID = t'.ID$
$p_2 : t.ID \neq t'.ID$
$p_3 : t.Name = t'.Name$
$p_4 : t.Name \neq t'.Name$
$p_5 : t.Salary = t'.Salary$
$p_6 : t.Salary \neq t'.Salary$
$p_7 : t.Salary < t'.Salary$
$p_8 : t.Salary \leq t'.Salary$
$p_9 : t.Salary > t'.Salary$
$p_{10} : t.Salary \geq t'.Salary$

Previous works compute, iterate or hash the evidence individually.

$(t_4, t_1) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$
$(t_4, t_2) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$
$(t_4, t_3) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$
$(t_4, t_5) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$

Evidence Contexts

	ID	Name	Salary
t_1	#1	Caruso	10 000
t_2	#2	Zhang	5500
t_3	#3	Schneider	6000
t_4	#4	Smith	11 000
t_5	#5	Caruso	6000

$p_1 : t.ID = t'.ID$
$p_2 : t.ID \neq t'.ID$
$p_3 : t.Name = t'.Name$
$p_4 : t.Name \neq t'.Name$
$p_5 : t.Salary = t'.Salary$
$p_6 : t.Salary \neq t'.Salary$
$p_7 : t.Salary < t'.Salary$
$p_8 : t.Salary \leq t'.Salary$
$p_9 : t.Salary > t'.Salary$
$p_{10} : t.Salary \geq t'.Salary$

Previous works compute, iterate or hash the evidence individually.

$(t_4, t_1) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$
$(t_4, t_2) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$
$(t_4, t_3) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$
$(t_4, t_5) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$

Much redundancy, e.g., $> 99\%$

Evidence Contexts

	ID	Name	Salary
t_1	#1	Caruso	10 000
t_2	#2	Zhang	5500
t_3	#3	Schneider	6000
t_4	#4	Smith	11 000
t_5	#5	Caruso	6000

$p_1 : t.ID = t'.ID$
$p_2 : t.ID \neq t'.ID$
$p_3 : t.Name = t'.Name$
$p_4 : t.Name \neq t'.Name$
$p_5 : t.Salary = t'.Salary$
$p_6 : t.Salary \neq t'.Salary$
$p_7 : t.Salary < t'.Salary$
$p_8 : t.Salary \leq t'.Salary$
$p_9 : t.Salary > t'.Salary$
$p_{10} : t.Salary \geq t'.Salary$

Previous works compute, iterate or hash the evidence individually.

$(t_4, t_1) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$
$(t_4, t_2) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$
$(t_4, t_3) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$
$(t_4, t_5) \rightarrow \{p_2, p_4, p_6, p_9, p_{10}\}$

Much redundancy, e.g., $> 99\%$

We propose **evidence contexts**:

$\langle t, tids, e \rangle \rightarrow$ "t combined with $t' \in$ tids produces e"

$\langle t_4, \{t_1, t_2, t_3, t_5\}, \{p_2, p_4, p_6, p_9, p_{10}\} \rangle$

- **Fewer integers**
- **Great compression potential**
- **Evidence multiplicity (for ADCs)**

Reconciling Evidence Contexts

ID	Name	Salary	
#1	Caruso	10 000	t_1
#2	Zhang	5500	t_2
#3	Schneider	6000	t_3
#4	Smith	11 000	t_4
#5	Caruso	6000	t_5

$p_1 : t.ID = t'.ID$ ✔
$p_2 : t.ID \neq t'.ID$ ✔
$p_3 : t.Name = t'.Name$ ✘
$p_4 : t.Name \neq t'.Name$ ✘
$p_5 : t.Salary = t'.Salary$
$p_6 : t.Salary \neq t'.Salary$
$p_7 : t.Salary < t'.Salary$
$p_8 : t.Salary \leq t'.Salary$
$p_9 : t.Salary > t'.Salary$
$p_{10} : t.Salary \geq t'.Salary$

$\langle t_1, \{t_2, t_3, t_4, t_5\}, \{p_2, p_4, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✘

Reconciling Evidence Contexts

ID	Name	Salary	
#1	Caruso	10 000	t_1
#2	Zhang	5500	t_2
#3	Schneider	6000	t_3
#4	Smith	11 000	t_4
#5	Caruso	6000	t_5

$p_1 : t.ID = t'.ID$ ✔
$p_2 : t.ID \neq t'.ID$ ✔
$p_3 : t.Name = t'.Name$ ✔
$p_4 : t.Name \neq t'.Name$ ✔
$p_5 : t.Salary = t'.Salary$
$p_6 : t.Salary \neq t'.Salary$
$p_7 : t.Salary < t'.Salary$
$p_8 : t.Salary \leq t'.Salary$
$p_9 : t.Salary > t'.Salary$
$p_{10} : t.Salary \geq t'.Salary$

$\langle t_1, \{t_2, t_3, t_4, t_5\}, \{p_2, p_4, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✘

↓

$\langle t_1, \{t_5\}, \{p_2, p_3, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✔

$\langle t_1, \{t_2, t_3, t_4\}, \{p_2, p_4, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✔

Reconciling Evidence Contexts

ID	Name	Salary	
#1	Caruso	10 000	t_1
#2	Zhang	5500	t_2
#3	Schneider	6000	t_3
#4	Smith	11 000	t_4
#5	Caruso	6000	t_5

$p_1 : t.ID = t'.ID$ ✔
$p_2 : t.ID \neq t'.ID$ ✔
$p_3 : t.Name = t'.Name$ ✔
$p_4 : t.Name \neq t'.Name$ ✔
$p_5 : t.Salary = t'.Salary$
$p_6 : t.Salary \neq t'.Salary$
$p_7 : t.Salary < t'.Salary$
$p_8 : t.Salary \leq t'.Salary$
$p_9 : t.Salary > t'.Salary$
$p_{10} : t.Salary \geq t'.Salary$

$\langle t_1, \{t_2, t_3, t_4, t_5\}, \{p_2, p_4, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✘

↓

$\langle t_1, \{t_5\}, \{p_2, p_3, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✔

$\langle t_1, \{t_2, t_3, t_4\}, \{p_2, p_4, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✔

↓

Operations are more likely to require less data movement due to predicate selectivity!

Reconciling Evidence Contexts

ID	Name	Salary	
#1	Caruso	10 000	t_1
#2	Zhang	5500	t_2
#3	Schneider	6000	t_3
#4	Smith	11 000	t_4
#5	Caruso	6000	t_5

$p_1 : t.ID = t'.ID$ ✔
$p_2 : t.ID \neq t'.ID$ ✔
$p_3 : t.Name = t'.Name$ ✔
$p_4 : t.Name \neq t'.Name$ ✔
$p_5 : t.Salary = t'.Salary$
$p_6 : t.Salary \neq t'.Salary$
$p_7 : t.Salary < t'.Salary$
$p_8 : t.Salary \leq t'.Salary$
$p_9 : t.Salary > t'.Salary$
$p_{10} : t.Salary \geq t'.Salary$

$\langle t_1, \{t_2, t_3, t_4, t_5\}, \{p_2, p_4, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✘

↓

$\langle t_1, \{t_5\}, \{p_2, p_3, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✔
$\langle t_1, \{t_2, t_3, t_4\}, \{p_2, p_4, p_6, p_9, p_{10}\}\rangle \rightarrow$ ✔

↓

Operations are more likely to require less data movement due to predicate selectivity!

☞ **How to perform the reconciliation fast?**

Evidence Context Pipeline (ECP)

– Reconciliation for each predicate group G (predicates sharing a column set).

Input: Context set EC **Output:** Context set EC′ corrected w.r.t. G

– Algorithms and data structures specialized for DC predicates.

- We consider only two cases due to the evidence ahead principle and inference:

Evidence Context Pipeline (ECP)

– Reconciliation for each predicate group G (predicates sharing a column set).

Input: Context set EC **Output:** Context set EC′ corrected w.r.t. G

– Algorithms and data structures specialized for DC predicates.

- We consider only two cases due to the evidence ahead principle and inference:

Equalities like $t.A = t'.B$

- Akin to hash-joins
- Bitmap indexes and logical operations
- Fewer computations required as most tuple pairs satisfy the $t.A \neq t'.B$ counterpart

Inequalities like $t.A < t'.B$

– Extend the equality case:

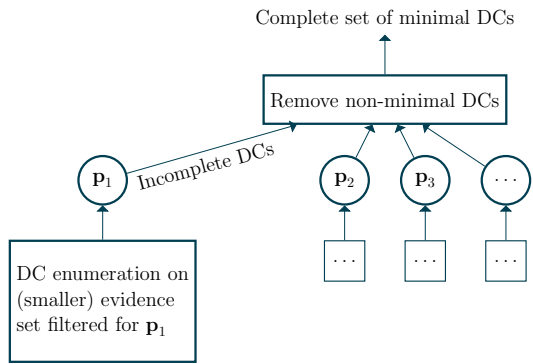
- Sorted bitmap indexes and logical operations
- Two-layered bitmap index with binning for high-cardinality
- Heuristics to improve compression and performance

Solutions for DC enumeration - Overview

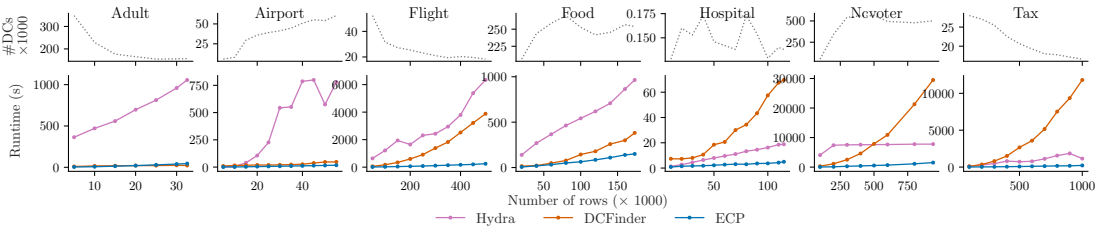
– Predicate subspaces:
  - Discover DCs containing $\mathbf{p}_1$, then $\mathbf{p}_2$, ...
  - Each path $\rightarrow$ a parallel thread
  - Late minimality check with subset lookups.

– Inverted index for filtering evidence.

– **New Hybrid Algorithms:**
  - Indexed Negative Cover Search Parallel (INCS-P)
  - Hybrid Evidence Inversion Parallel (HEI-P) $\rightarrow$ (often) the fastest
  - Hybrid Minimal-to-Maximal Conversion Search with pruning Parallel (HMMCS-P)

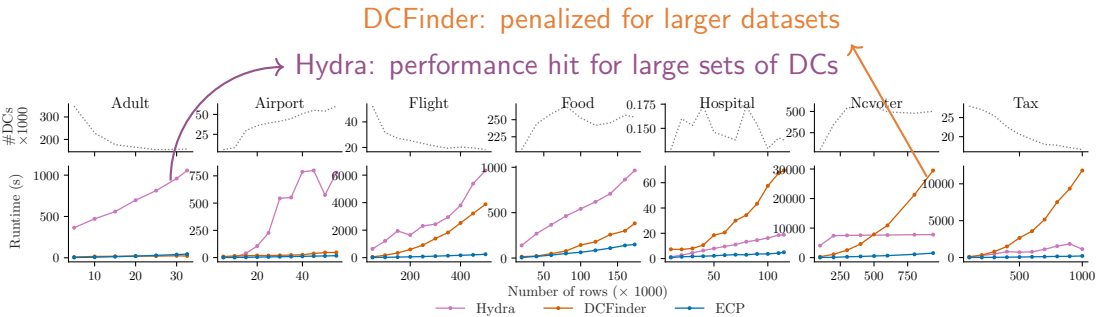


Experimental Evaluation - Evidence Set Building



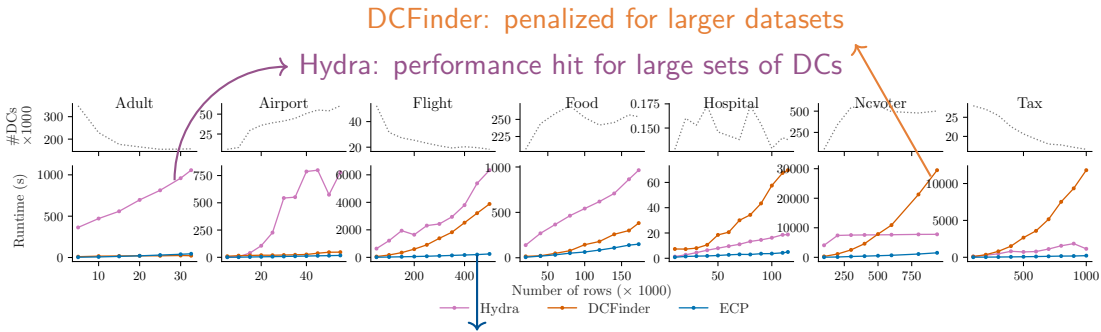
* Compared to the fastest algorithms that existed at the time
* Hydra does not support discovery of approximate DCs

Experimental Evaluation - Evidence Set Building



* Compared to the fastest algorithms that existed at the time
* Hydra does not support discovery of approximate DCs

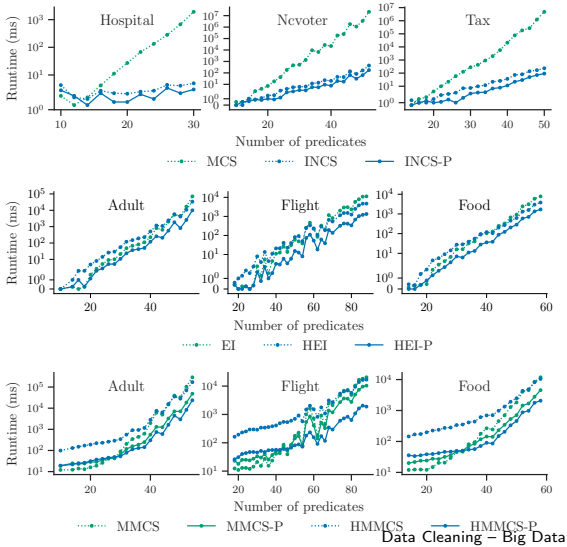
Experimental Evaluation - Evidence Set Building



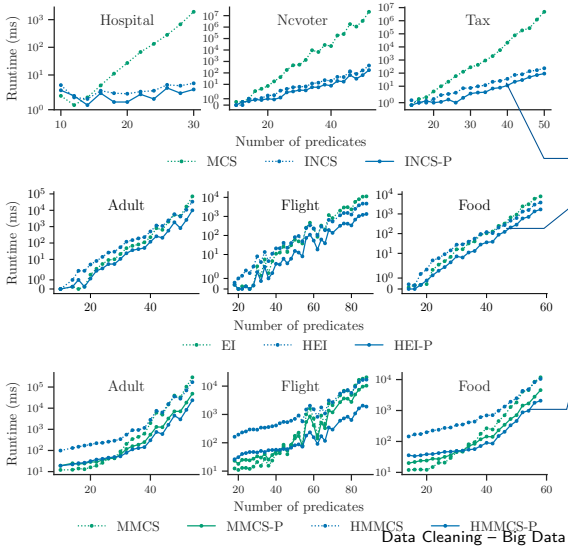
Evidence Context Pipeline (ECP):
More stable and often orders of magnitude faster!

* Compared to the fastest algorithms that existed at the time
* Hydra does not support discovery of approximate DCs

Experimental Evaluation - DC Enumeration



Experimental Evaluation - DC Enumeration



Runtime improvements of (up to) orders of magnitude!

New Directions – Incremental Profiling

Research Question:

`Can we efficiently maintain metadata for dynamic data?`

- Invalidate existing dependencies
  - ✗ Check all dependencies?
- Discover new dependencies
  - ✗ Rerun entire algorithm?

	Name	Phone	Position	Salary	Hired
t_0	W. Jones	202-222	Developer	$2.000	2012
t_1	B. Jones	202-222	Developer	$3.000	2010
t_2	J. Miller	202-333	Developer	$4.000	2010
t_3	D. Miller	202-333	DBA	$9.000	2010
t_4	W. Jones	202-555	DBA	$7.000	2010
~~t_5~~	~~W. Jones~~	~~202-222~~	~~Developer~~	~~$1.000~~	~~2012~~
...	...	...	...	...	...

New Directions – Incremental Profiling

Research Question:

Can we efficiently maintain metadata for dynamic data?

- Invalidate existing dependencies
  - ✗ Check all dependencies?
- Discover new dependencies
  - ✗ Rerun entire algorithm?

	Name	Phone	Position	Salary	Hired
t_0	W. Jones	202-222	Developer	$2.000	2012
t_1	B. Jones	202-222	Developer	$3.000	2010
t_2	J. Miller	202-333	Developer	$4.000	2010
t_3	D. Miller	202-333	DBA	$9.000	2010
t_4	W. Jones	202-555	DBA	$7.000	2010
~~t_5~~	~~W. Jones~~	~~202-222~~	~~Developer~~	~~$1.000~~	~~2012~~
...	...	...	...	...	...

New Directions – Genuine Dependencies

Research Question:

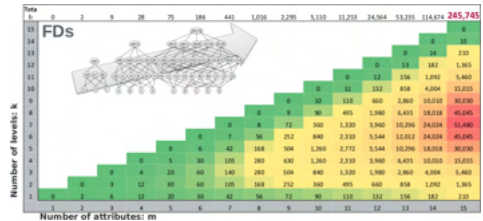
`Can we effectively identify genuine dependencies?`

- Most dependencies are spurious
  - ✓ Ranking (coverage, simplicity, etc)
  - ✗ Top-5? Top-10?
- Features for "Good" dependencies?

Accidental?

Useful?

Reliable?



New Directions – User-in-the-loop

Research Question:

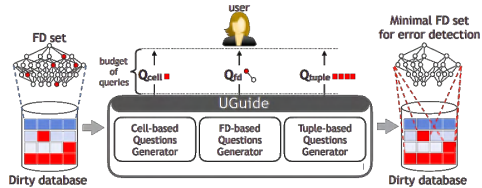
How to effectively leverage user input for profiling ?

- Too many results
  - ✗ Let the user label everything

Questions?

Feedback?

Validation?



New Directions – Visualization

Research Question:

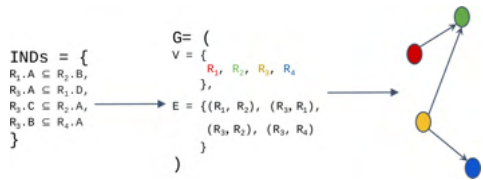
How to effectively visualize data profiling results?

- Too many results
  - ✗ Show the user everything as a list

Ranked List?

Table?

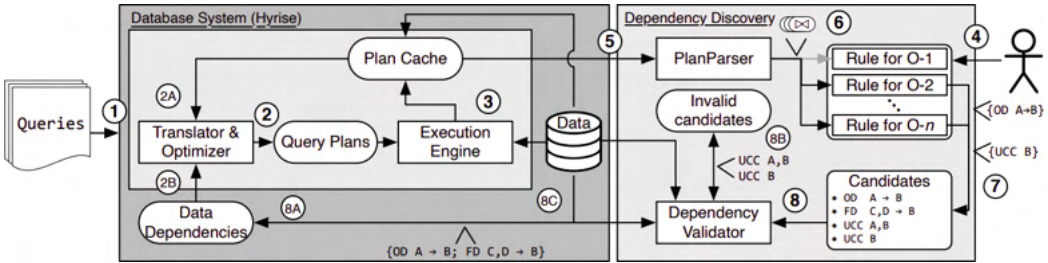
Iterative Graph?



New Directions – Application

Research Question:

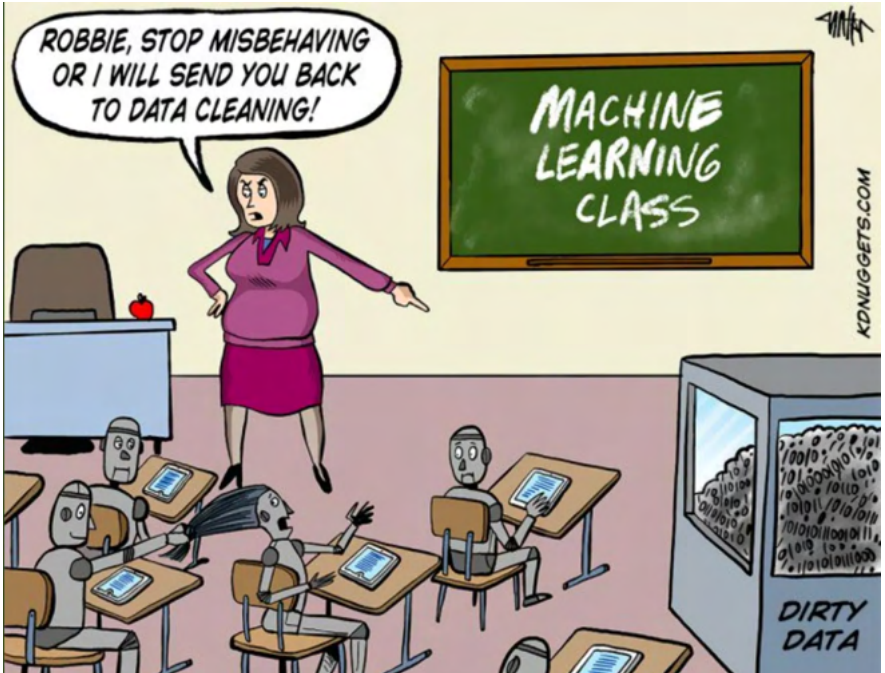
How to effectively use the application to narrow data profiling results?



Workload-driven, Lazy Discovery of Data Dependencies for Query Optimization. Kossmann et al. CIDR, 2022.

Summary

1. Motivation
2. Types of Data Errors
3. Error Detection
4. Data Repair
5. Data Profiling


ROBBIE, STOP MISBEHAVING OR I WILL SEND YOU BACK TO DATA CLEANING!
MACHINE LEARNING CLASS
KDNUGGETS.COM
DIRTY DATA