CS422
Database systems

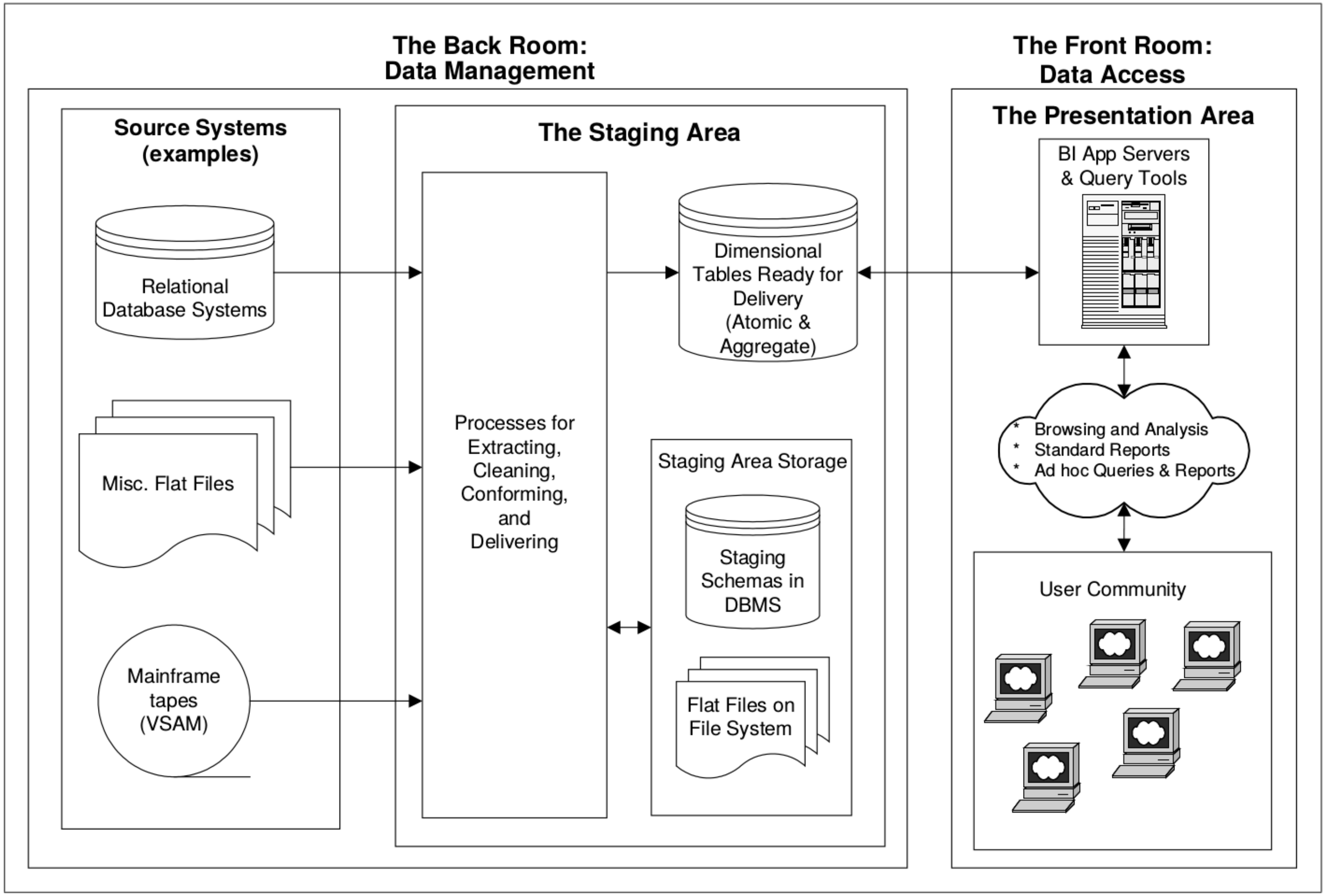
Data Wrangling

Data-Intensive Applications and Systems (DIAS) Laboratory
École Polytechnique Fédérale de Lausanne

Some slides adapted from:

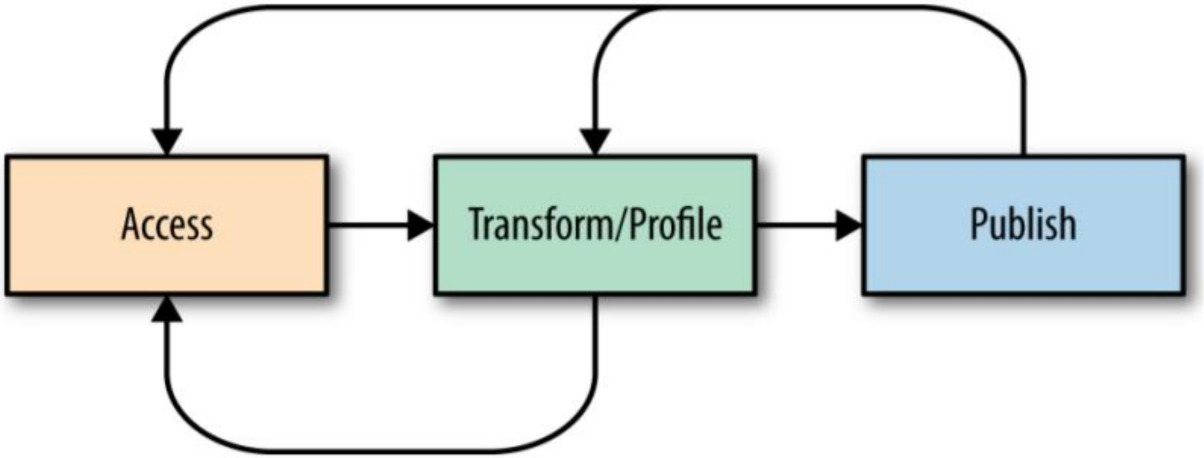
- Joe Hellerstein
- Xu Chu, Ihab Ilyas
- N. Koudas, S. Sarawagi, D. Srivastava

The ETL process



What is data wrangling

- Transforming or preparing data for analysis
- This is how you “get your head in the game”
  - Understand what you have
  - Assess strengths and weaknesses of your data
  - Hypothesize about what to do with your data
  - Get it ready



ETL vs Data wrangling

ETL

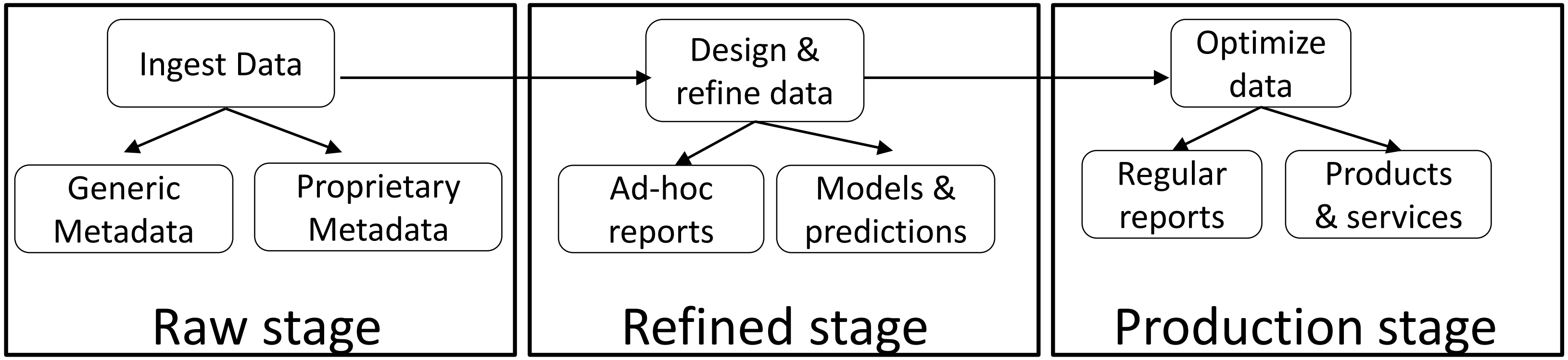
- IT employees build pipelines for their business counterparts
- Well-structured data
- Data-warehousing applications

Data wrangling

- Business analysts who know the data
- Diverse, complex data
- Exploratory analysis use-cases

Different users, different data, different use-cases

Stages of Wrangling



- Raw: Data ingestion & discovery (“unboxing”)
  - What: Exploratory ad hoc analysis
  - Who: The individual wrangler
- Refined: Curating data for reuse
  - What: Data warehousing, canonical models
  - Who: Data curators, IT engineers
- Production: Ensuring feeds and workflows
  - What: Recurrent, automated use cases
  - Who: SW engineers and IT/ops folks

Rough Guide to Wrangling Issues

- Structure: the “shape” and granularity of a data file
- Faithfulness: how well does data capture “reality”
- Temporality: how is the data situated in time
  - Scope: how (in)complete is the data

Outline

- **Data structuring**
- Data accuracy
  - Integrity constraints
  - Outliers
  - Duplicates
- Temporality

Structuring data

- *Intrarecord* structuring
  - Reorder record fields (moving columns)
  - Creating new record fields through extracting values
  - Combining multiple record fields into a single record field

- *Interrecord* structuring
  - Filter dataset by removing records
  - Shift granularity through aggregation and pivots

Intrarecord structuring

- Positional extraction

Date
03102015
03302015
03022015

→

Day
10
30
02

- Pattern extraction

Contribution
P/R DEDUCTION ($296.67 MONTHLY)
P/R DEDUCTION ($326.67 MONTHLY)

→

Monthly Contribution
$296.67
$326.67

- Complex structure extraction
  - JSON array: [“Sally”,“Bob”,“Alon”,“Georgia”]
  - JSON map: {“product”:“Trifacta Wrangler”, “price”:“free”}

Interrecord structuring - Aggregations

contribution data from the US presidential election

Id	Contribution
C00406	750
C00406	1000
C00253	225
C00253	50

Compute:

- average contribution
- sum of contributions
- the number of contributions

Id	Sum Contribution	Mean Contribution	Count Contribution
C00406	1750	875	2
C00253	275	137.5	2

Interrecord structuring - Pivots

- Unpivoting, denormalization

Region	2015	2016
East	2300	2453
West	9866	8822
Midwest	2541	2575

Restructure data
Each row contains sales for a unique combination of region year

Region	Year	Sales
East	2015	2300
East	2016	2453
West	2015	9866
West	2016	8822
Midwest	2015	2541
Midwest	2016	2575

Outline

- Data structuring
- **Data accuracy**
  - Integrity constraints
  - Outliers
  - Duplicates
- Temporality

Assessing faithfulness

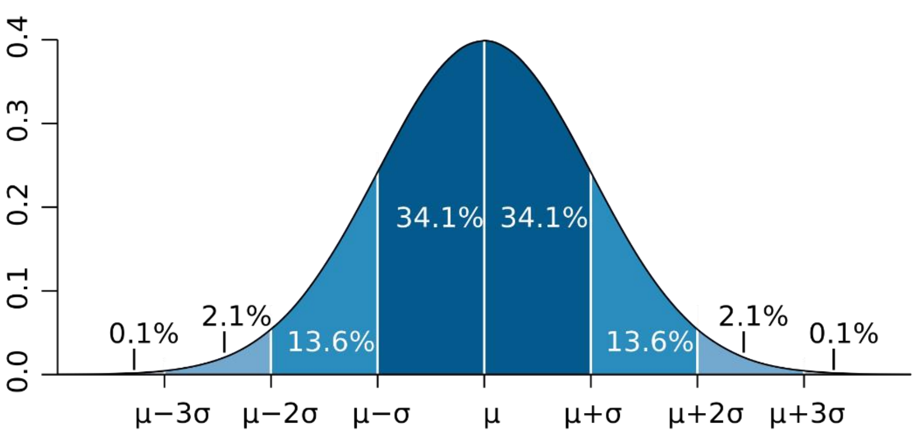
- The faithfulness of a record can only be evaluated in context
  - Application context
  - Context in your data set
    - Across records

Students

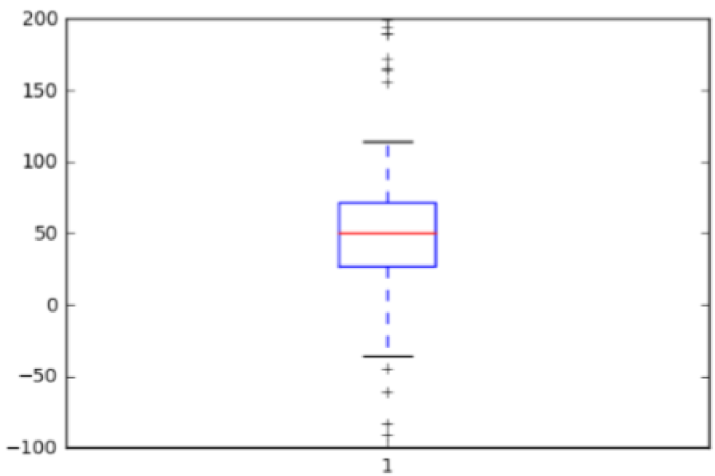
id: integer	DOB: date	GPA: float	Risk: float
123457	01/16/1997	3.2	465
123458	01/24/2017	2.7	28
123457	01/16/2002	5.0	27
123459	03/22/1996	3.6	31
123460	06/13/1997	2.2	43

Faithfulness across records: Outliers

- What is an "outlier"?
  - A value that is "far" from the "center"
- Distribution-based definition
  - Center (e.g. average, median)
  - Spread (e.g. standard deviation, IQR)

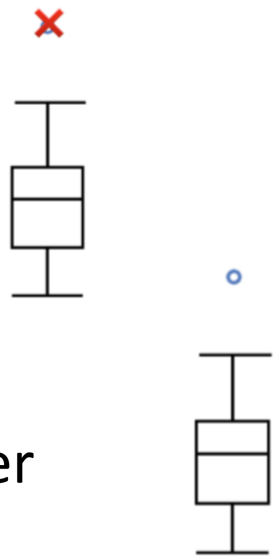


source: wikipedia



What to do with outliers?

- Delete ("trimming")
- Set to a default
  - E.g., the nearest non-outlier
- Good Hygiene:
  - Leave the original column
  - Derive an indicator column to flag presence of outlier
  - Derive a clean column for your use



Correlations within records

- Dependence (i.e. lack of independence!) between 2 random variables
- Think of the attributes in a relational schema
  - An instance of that relation was generated from some real-world process
  - Each column of that relation is a "random variable" generated by the process
- A Functional Dependency is a "deterministic" correlation

Functional Dependencies (FDs)

- Generalization of Keys
- Attribute A determines Attribute B
  - customerId -> age
  - i.e. age= f(customerId)
- A set of columns determines another set of columns
  - transactionTime, customerId -> age, residenceArea
- Primary Keys are special FDs
  - Right-hand-size is the set of all attributes in the relation

Conditional Functional Dependencies

- (X → Y, Tp)
- An FD defined on a subset of the data
- Example: ZIP → Street is valid on subset of the data where Country = "England"
- Example: AC = 020 where City = London

CFD example

({name, type, country} → {price, tax}, Tp)

TID	Name	Type	Country	price	tax
t1	Harry Potter	book	France	10	0
t2	Harry Potter	book	France	10	0
t3	Harry Potter	book	France	10	0.05
t4	Terminator	DVD	Italy	25	0.08
t5	Terminator	DVD	Italy	25	0.05
t6	Spiderman	DVD	UK	19	0

Name	Type	Country	price	tax
-	book	France	-	-
-	-	UK	-	-

CFD needs to hold only over the tuples matching the tableau

Matching Dependencies

Tran:

FN	LN	Street	City	AC	Post	Phone	Item
Robert	Brady	5 Wren St	London	020	WC1H	3887834	Watch
Robert	Brady	Null	London	020	WC1H	3887834	necklace

Master: Card

FN	LN	Street	City	AC	Zip	Tel
Robert	Brady	5 Wren St	London	020	WC1H	3887644

- MD: Tran[LN, City, Street, Post] = card[LN, City, St, Zip] ^
  Tran[FN] ≈ Card[FN] → Tran[FN, Phone] ↔ Card[FN, Tel]

More complex integrity constraints

Employees

ID	FN	LN	Role	City	State	Salary
105	Anne	Nash	M	NYC	NY	110
211	Mark	White	E	SJ	CA	80
386	Mark	Lee	E	NYC	AZ	75
235	John	Smith	M	NYC	NY	1200

Functional Dependency: City → State

Business Rule:
Two employees of the same role, the one who lives in NYC cannot earn less than the one who does not live in NYC

Denial Constraints (DCs)

$$\forall t1, t2, \ldots, tk \ \neg(p(x1) \land p(x2) \land \cdots \land p(xn))$$

- A universal constraint dictates that a set of predicates cannot be true together
- Each predicate expresses a relationship between two cells, or a cell and a constant

FDs and CFDs are subcategories of DCs

Denial Constraints: Example

- FD: City → State:

$$\forall t1, t2 \in Employee,$$
$$\neg\left((t1.city = t2.city) \wedge (t1.State \neq t2.State)\right)$$

- Two employees of the same role, the one who lives in NYC cannot earn less than the one who does not live in NYC

$$\forall t1, t2 \in Employee, \neg\left((t1.Role = t2.Role) \wedge (t1.city = "NYC") \wedge (t2.city \neq "NYC") \wedge (t1.salary < t2.salary)\right)$$

DCs are expressive enough to support arbitrary data quality rules

Data Deduplication

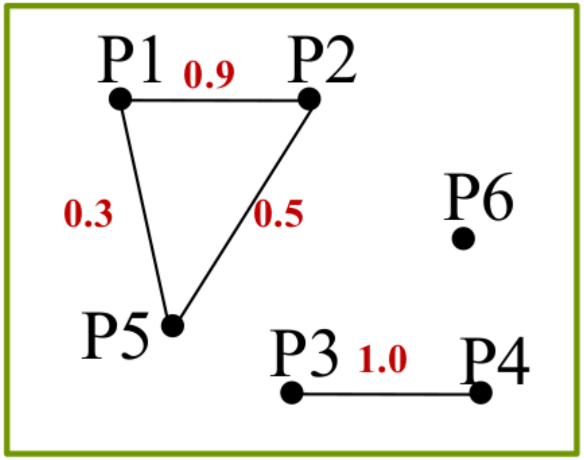
- Similarity measures
- Machine learning for classifying pairs as duplicates or not (unsupervised, supervised, and active)
- Clustering and handling of transitivity
- Merging and consolidation of records

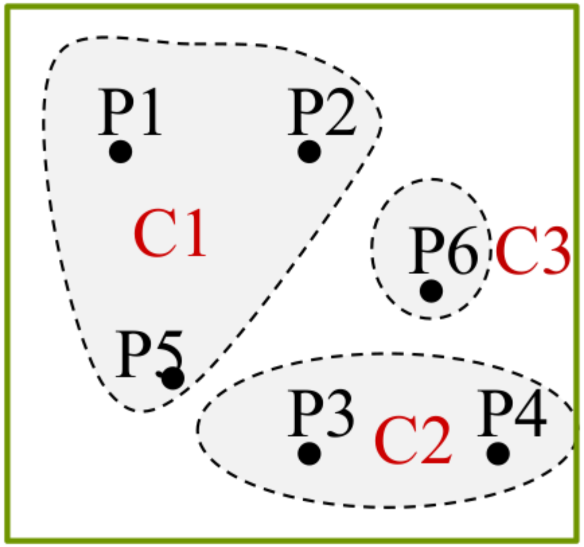
Duplicate elimination with clustering

Unclean relation

ID	Name	ZIP	Income
P1	Green	51519	30K
P2	Green	51518	32K
P3	Peter	30528	40K
P4	Peter	30528	40K
P5	Gree	51519	55K
P6	Chuck	51519	30K

Compute pair-wise similarity





Merge clusters

Clean relation

ID	Name	ZIP	Income
C1	Green	51519	39K
C2	Peter	30528	40K
C3	Chuck	51519	30K

Possible Repairs

- A possible repair is a clustering (partitioning) of the input tuples

ID	Name	ZIP	Income
P1	Green	51519	30K
P2	Green	51518	32K
P3	Peter	30528	40K
P4	Peter	30528	40K
P5	Gree	51519	55K
P6	Chuck	51519	30K

Possible repairs

X1	X2	X3
{P1}	{P1,P2}	{P1,P2,P5}
{P2}	{P3,P4}	{P3,P4}
{P3,P4}	{P5}	{P6}
{P5}	{P6}	
{P6}		

Temporality

- Often two kinds of time in data
  - Time of data entry
  - Time of a recorded phenomenon being “true”
    - E.g. A physical time of an event happening
    - E.g. An “effective” time, e.g. date that a subscription will start
- Often more
- Time is tricky!
  - Periodicities (recurring patterns in Days of the week)
  - Non-uniform hierarchy of units (# days in a month, # of days in a year, etc.)
  - Time zones are complex
  - Clocks can be skewed
  - Relativity: true perception of event may vary


OTHERSIDE
RED HOT CHILI PEPPERS

Outline

- **Data repairing techniques**
- Dealing with similarity comparisons

Data repairing techniques

Repair target
(what to repair?)

Automation
(how to repair?)

Data

Rules

Both

Automatic

Human guided

Trust the rules

Trust the Data

- **Schema evolution**
- **Obsolete rules**

Data repairing automation

- Most automatic repairing techniques adopt the “minimality” of repairs principle
  - **Minimal repairs principle**: the distance between the original database and the modified database is minimized

- Repairing techniques in practice are:
  - predominantly manual and
  - semi-automatic at best

Data repairing requires ground truth to infer the correct value of an erroneous cell

Data repairing FD violations

- I is a dirty database if $I \not\models \Sigma$ and I_j is a repair for I if $I_j \models \Sigma$
- For a repair I_j , $\Delta(I_j)$ is the set of changed cells

I

	A	B
t_1	1	2
t_2	1	3
t_3	1	3
t_4	4	5

$\Sigma = \{A \rightarrow B\}$

I_1

A	B
1	3
1	3
1	3
4	5

$\Delta(\boldsymbol{I_1}) = \{t_1[B]\}$

Minimal repair

I_2

A	B
1	2
1	2
1	2
4	5

$\Delta(\boldsymbol{I_2}) = \{t_2[B], t_3[B]\}$

Outline

- Data repairing techniques
- **Dealing with similarity comparisons**

Fuzzy join

- A fuzzy join of $R_1(A_1,\ldots,A_n)$ and $R_2(B_1,\ldots,B_m)$ is:
  - A subset of the cartesian product of R_1 and R_2
  - "Matching" specified attributes $A_{i1}, \ldots, A_{ik}$ with $B_{i1}, \ldots, B_{ik}$
  - Labeled with a similarity score $> t > 0$
- Naïve method: for each record pair, compute similarity score
  - I/O and CPU intensive, not scalable to millions of records
  - Goal: reduce $O(n^2)$ cost to $O(n*w)$, where $w << n$
  - Reduce number of pairs on which similarity is computed
  - Take advantage of efficient relational join methods

Q-gram set join

- Goal: compute thresholded similarity distance join on string attributes
- Methodology: domain-independent similarity
  - Extract set of all overlapping q-grams Q(s) from string s
  - $Dist(s1,s2) \leq d \rightarrow |Q(s1) \cap Q(s2)| \geq max(|s1|,|s2|) - (d-1)*q - 1$
  - Cheap filters (length, count, position) to prune non-matches
  - Pure SQL solution: cost-based join methods

Lesson: reduce fuzzy join to aggregated set intersection

Q-gram set join in action

ID	Name
r1	Srivastava
r2	Shrivastava
r3	Shrivastav

ID	Name	3-grams
r1	Srivastava	##s, **#sr, sri**, riv, iva, vas, ast, sta, tav, ava, va$, a$$
r2	Shrivastava	##s, **#sh, shr, hri**, riv, iva, vas, ast, sta, tav, ava, va$, a$$
r3	Shrivastav	##s, **#sh, shr, hri**, riv, iva, vas, ast, sta, tav, av$, v$$

Edit Distance (ED):

- $ED(s1,s2) \leq d \rightarrow |Q(s1) \cap Q(s2)| \geq \max(|s1|,|s2|) - (d-1)*q - 1$
- $ED(r1, r2) = 1$, $|Q(r1) \cap Q(r2)| = 10$
- $ED(r1, r3) = 1$, $|Q(r1) \cap Q(r2)| = 7$

Q-gram set join in action

ID	Name
r1	Srivastava
r2	Shrivastava
r3	Shrivastav

Q

ID	Qg
r1	##s
r1	#sr
r1	sri
r1	riv
r1	iva
r1	vas
r1	ast
r1	sta
r1	tav
r1	ava
r1	va$
r1	a$$

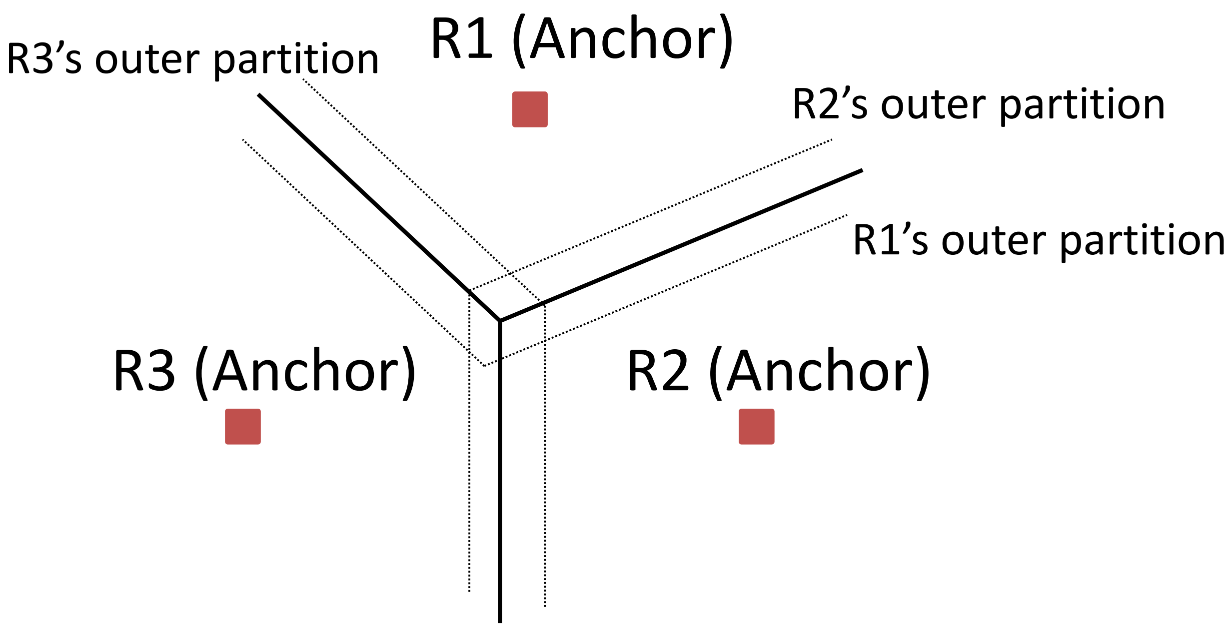
ID	Qg
r3	##s
r3	#sh
r3	shr
r3	hri
r3	riv
r3	iva
r3	vas
r3	ast
r3	sta
r3	tav
r3	av$
r3	v$$

```
SELECT Q1.ID, Q2.ID
FROM Q AS Q1, Q AS Q2
WHERE Q1.Qg = Q2.Qg
GROUP BY Q1.ID, Q2.ID
HAVING COUNT(*) > T
```

Scaling out similarity joins

ClusterJoin, 2014

Sample anchor points



Assign values to the closest anchors

- Take into consideration neighboring partitions

Conclusion

- Data transformations
  - Update structure and granularity
- Data accuracy
  - Error detection using rules or similarity comparisons
  - Data repairing is expensive and requires human guidance

Reading material

- I. F. Ilyas and X. Chu. Trends in cleaning relational data: Consistency and deduplication. Foundations and Trends in Databases, 5(4):281–393, 2015
- A. D. Sarma, Y. He, and S. Chaudhuri. ClusterJoin: A Similarity Joins Framework using Map-Reduce. PVLDB, 7(12):1059–1070, 2014
- T. Rattenbury, J.M. Hellerstein, J. Heer, S. Kandel, and C. Carreras, Principles of Data Wrangling: Practical Techniques for Data Preparation. O'Reilly Media, 2017