Database Design and Implementation

CS 645

Data analytics and systems

Gaining insights from data

type of analytics

descriptive

What happened?

data mining

diagnostic

Why did it happen?

causal reasoning

predictive

What will happen?

simulation & statistics

prescriptive

What should happen?

optimization

data mining

descriptive

What happened?

association rule mining

finding interesting relations between variables in large datasets

examples

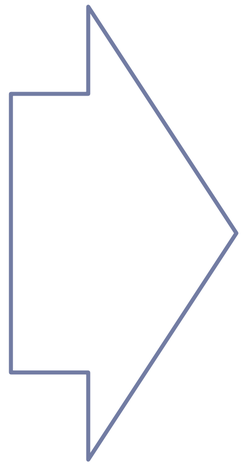
— 98% of customers who purchase tires get automotive services done
— customers who buy burgers also buy mustard and ketchup

association rule mining

finding interesting relations between variables in large datasets

DB of "basket data"

TID	items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5



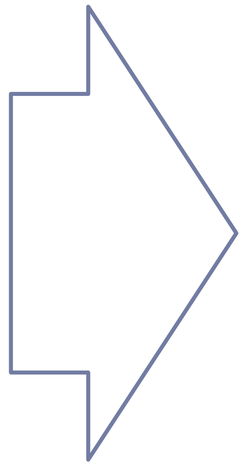
association rules

{1} => {3}
{2,3} => {5}
{2,5} => {3}
⋮

association rule mining

DB of "basket data"

TID	items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5



association rules

{1} => {3}
{2,3} => {5}
{2,5} => {3}
⋮

Association rule: X => Y

- X and Y are disjoint itemsets, called antecedent (LHS) and consequent (RHS)
- **Confidence**: c% of transactions that contain X also contain Y
- **Support**: s% of all transactions contain both X and Y

Goal: Find all rules that satisfy confidence and support thresholds

TID	Cereal	Beer	Bread	Bananas	Milk
1	X		X		X
2	X		X	X	X
3		X			X
4	X			X	
5			X		X
6	X				X
7		X		X	
8			X		

Support (Cereal)

4/8 = 0.5

Support (Cereal => Milk)

3/8 = 0.375

TID	Cereal	Beer	Bread	Bananas	Milk
1	X		X		X
2	X		X	X	X
3		X			X
4	X			X	
5			X		X
6	X				X
7		X		X	
8			X		

Confidence (Cereal => Milk)

3/4 = 0.75

Confidence (Bananas => Bread)

1/3 = 0.333

A-priori algorithm〔1995〕

- $\{i_1, i_2, \ldots, i_m\}$ a set of literals (items)
- $\{T_1, T_2, \ldots, T_n,\}$ a set of transactions, where each transaction is a set of items (itemset)
  - Size of an itemset is the number of its items
  - Itemset of size k is a k-itemset
- We assume that each itemset is in lexicographical order

general strategy

Step I: Find all item sets with minimum support (min_sup)

TID	items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

support	itemsets
0.25	{4}, {1,2}, {1,4}, {1,5}, {3,4}, {1,3,4}, {1,2,3}, {1,2,5}, {1,3,5}, {1,2,3,5}
0.5	{1}, {1,3}, {2,3}, {3,5}, {2,3,5}
0.75	{2}, {3}, {5}, {2,5}

Step II: Generate rules from min_sup - itemsets

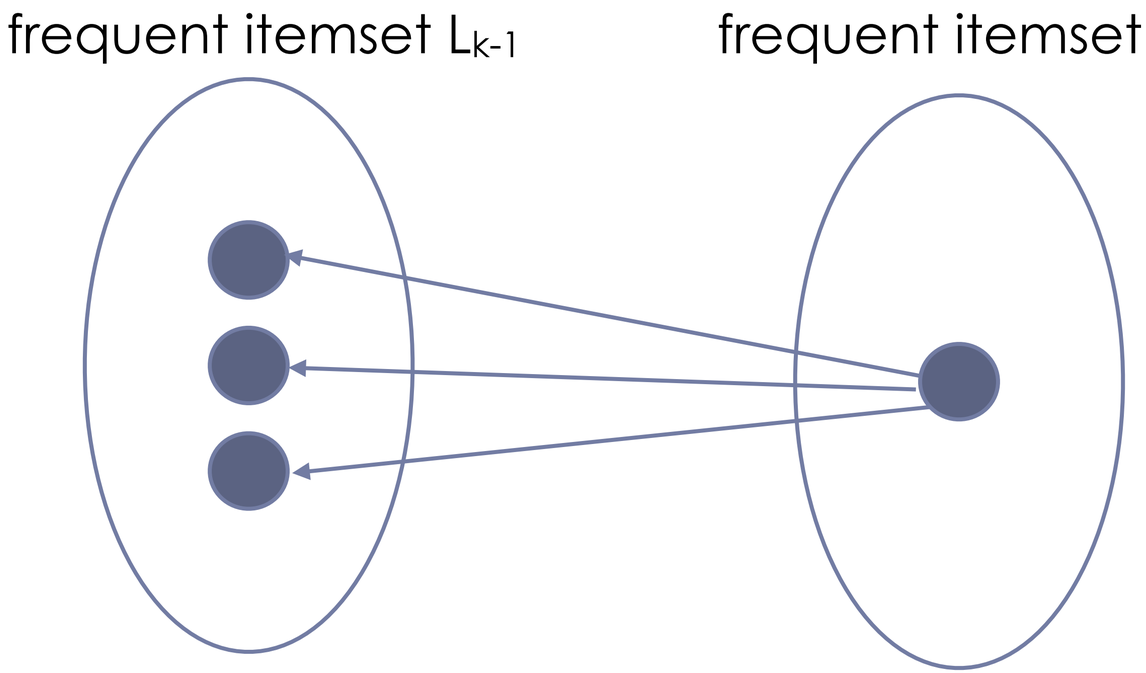
support	confidence	itemsets
0.5	66%	{3}=>{1}, {3}=>{2}, {2}=>{3}, {3}=>{5}, {5}=>{3}, {5}=>{2,3}, {3}=>{2,5}, {2}=>{3,5}, {5,2}=>{3}, {5,3}=>{2}
0.5	100%	{1}=>{3}, {5,3}=>{2}, {2,3}=>{5}
0.75	100%	{5}=>{2}, {2}=>{5}

Step 1: finding min_sup itemsets

- Too complex to test all! (powerset of all literals)

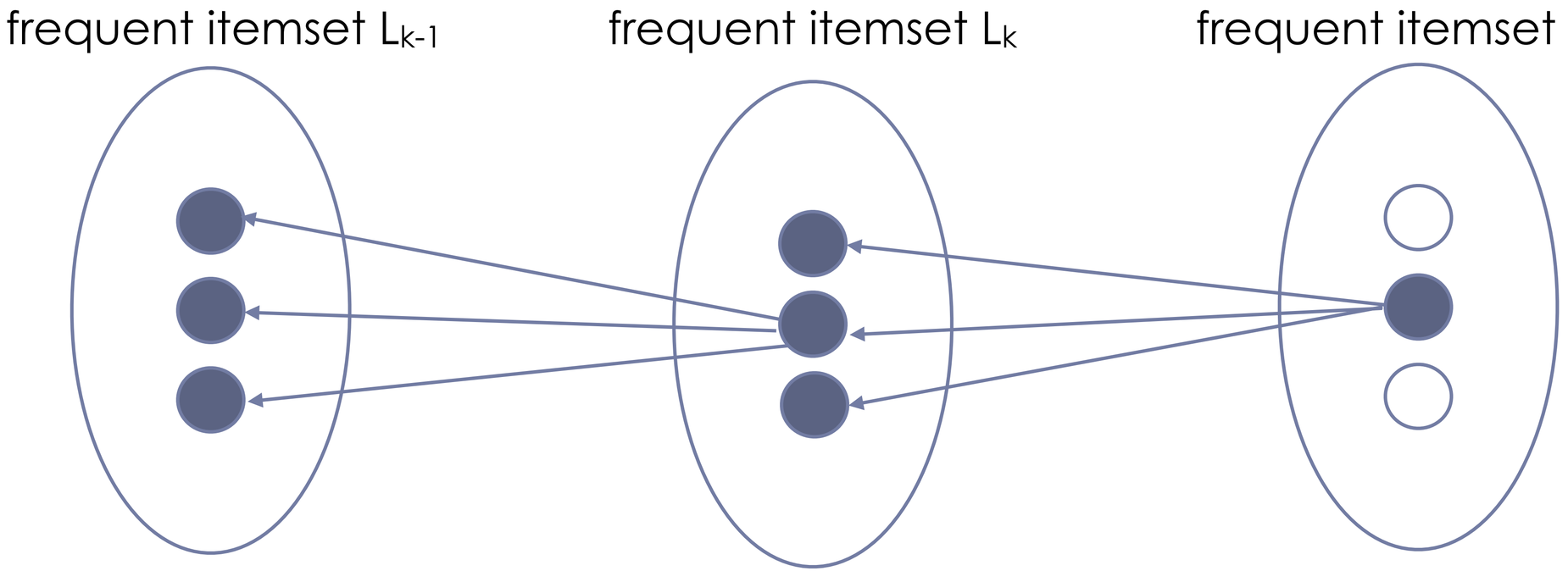
- Exploit *anti-monotonicity*
  - Adding items to an itemset does not increase its support
  - If an itemset is frequent, its subsets are also frequent
  - If an itemset is infrequent, its supersets are also infrequent

anti-monotonicity



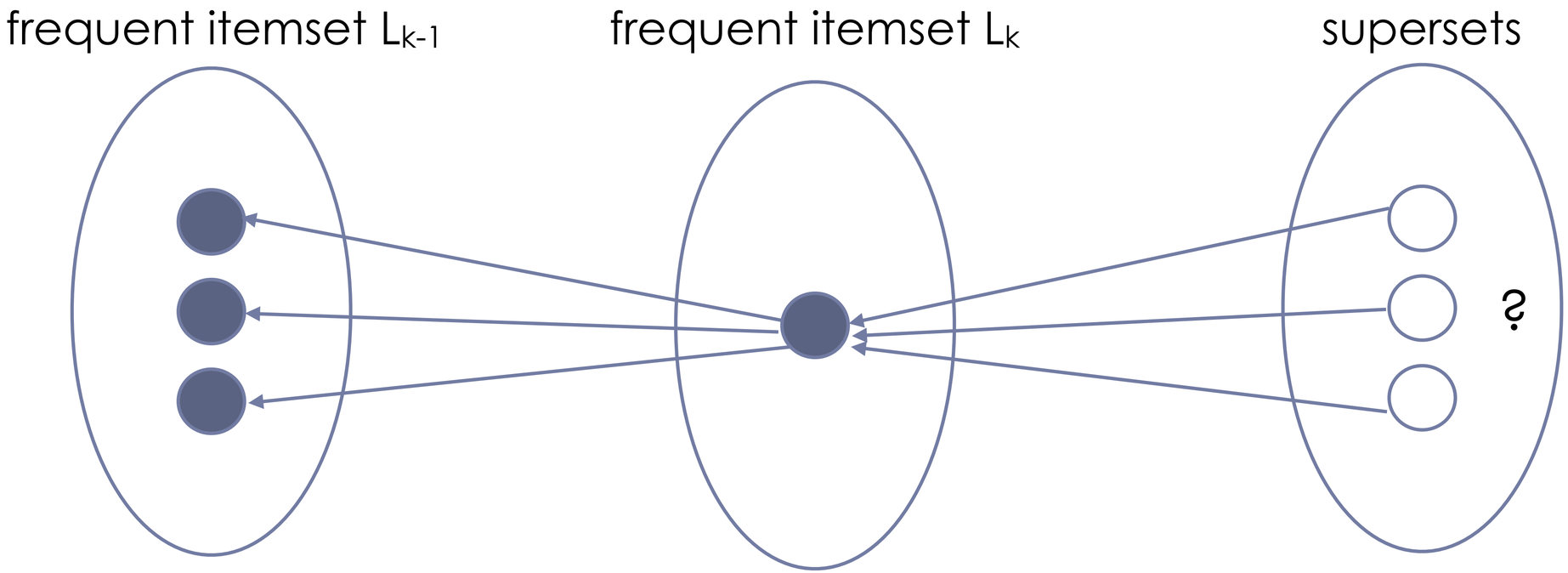
- Adding items to an itemset does not increase its support
- If an itemset is frequent, its subsets are also frequent
- If an itemset is infrequent, its supersets are also infrequent

anti-monotonicity



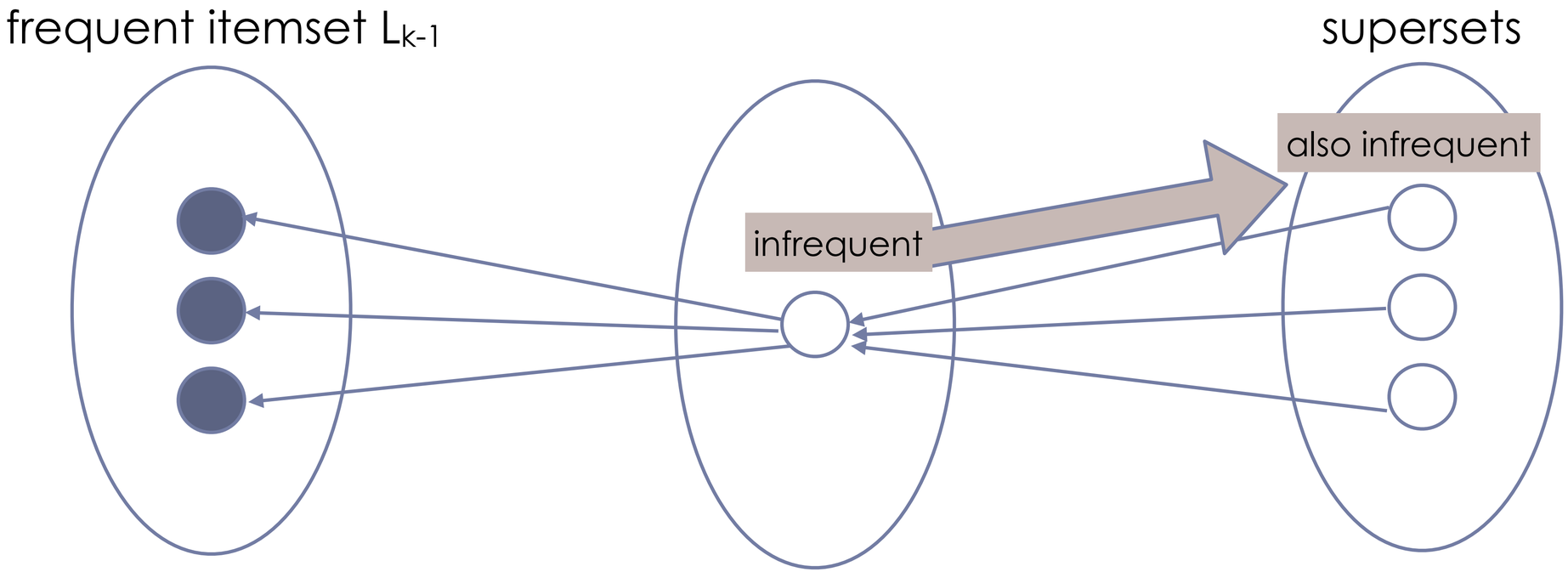
- Adding items to an itemset does not increase its support
- If an itemset is frequent, its subsets are also frequent
- If an itemset is infrequent, its supersets are also infrequent

anti-monotonicity



- Adding items to an itemset does not increase its support
- If an itemset is frequent, its subsets are also frequent
- If an itemset is infrequent, its supersets are also infrequent

anti-monotonicity



- Adding items to an itemset does not increase its support
- If an itemset is frequent, its subsets are also frequent
- If an itemset is infrequent, its supersets are also infrequent

A-priori: augment itemset size inductively

- Begin with min_sup itemsets of size 1 (L_1)

repeat

- Generate candidate sets of size k (C_k) from L_{k-1} (without looking at the data)
- Remove sets from C_k that contain unsupported subsets
- Check the rest against the DB to produce L_k

- Generate candidate sets of size k (C_k) from L_{k-1} (without looking at the data)

Naive way: extend itemsets with all possible items

A-priori: join L_{k-1} with itself to add a single item

E.g.: L_{k-1} has {1,2,3}, {1,2,4}, {1,3,4}, {1,3,5}, {2,3,4}

Candidates: {1,2,3,4}, {1,2,3,5}, {1,3,4,5}

- Remove sets from C_k that contain unsupported subsets

  A-priori: join L_{k-1} with itself to add a single item

  E.g.: L_{k-1} has {1,2,3}, {1,2,4}, {1,3,4}, {1,3,5}, {2,3,4}

  Candidates: {1,2,3,4}, {1,2,3,5}, {1,3,4,5}

  {2,3,5} is unsupported

  {1,4,5} is unsupported

- Check the rest against the DB to produce L_k

type of analytics

descriptive

What happened?

data mining

diagnostic

Why did it happen?

causal reasoning

predictive

What will happen?

simulation & statistics

prescriptive

What should happen?

optimization

diagnostic

Why did it happen?

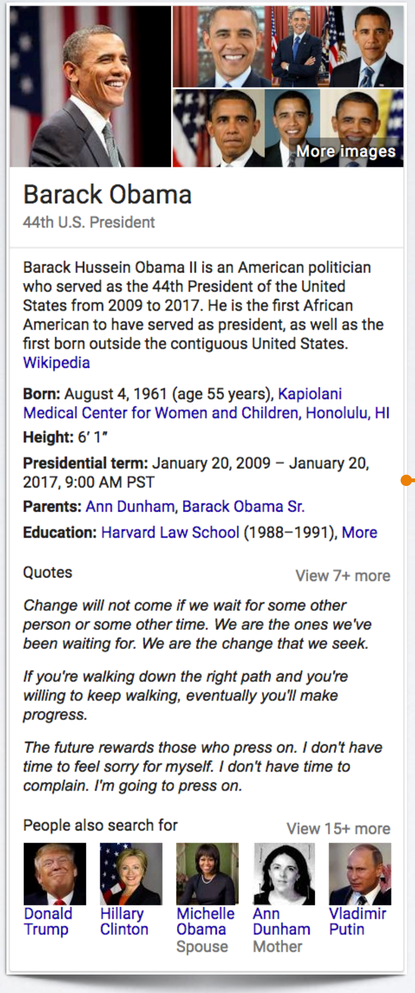
causal reasoning

explanations of results

diagnosis of data errors

diagnosis of system errors

[SIGMOD 2015]



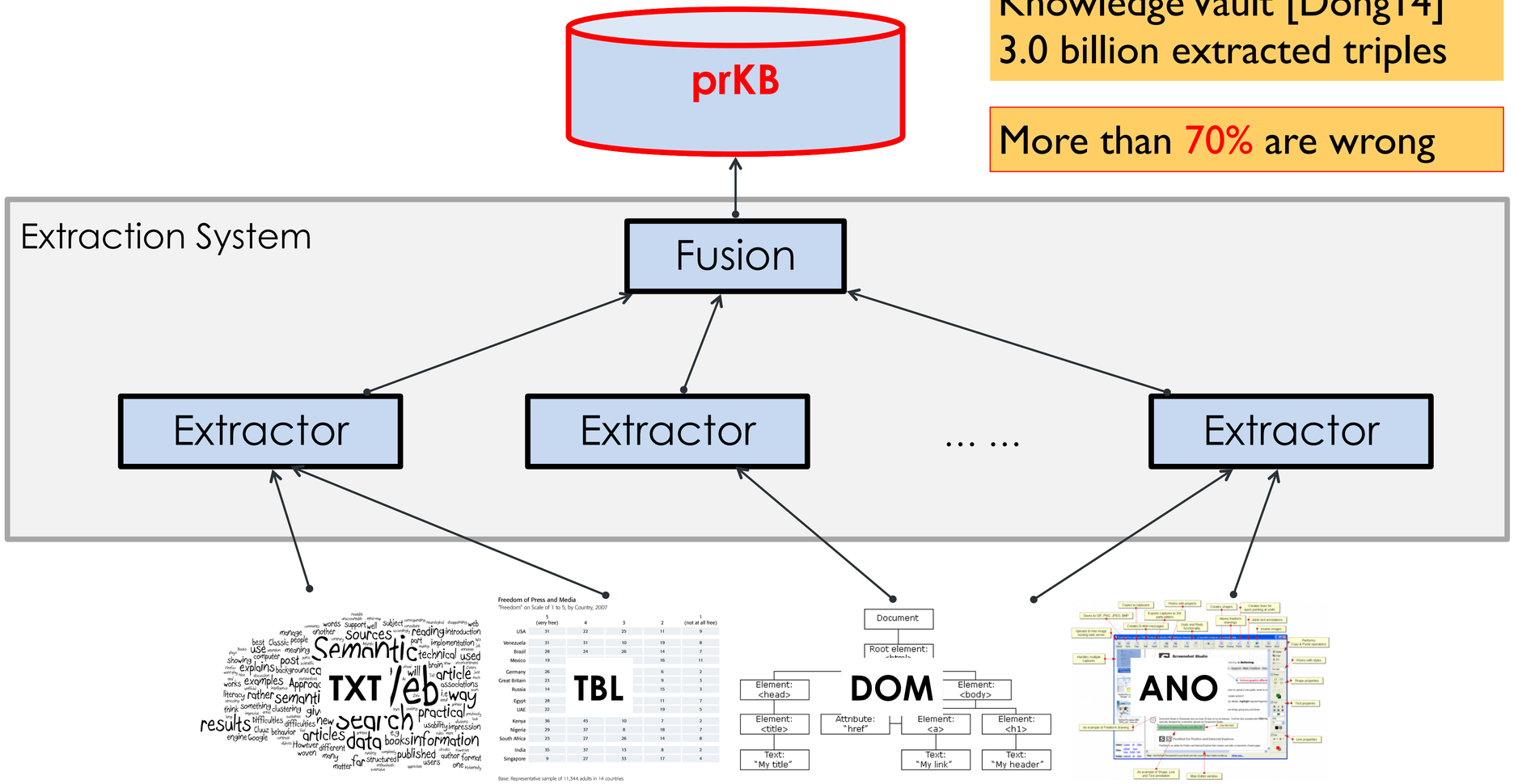
Structured information retrieved from unstructured web data

[SIGMOD 2015]

a look at Knowledge Bases

Example:
Knowledge Vault [Dong14]
3.0 billion extracted triples

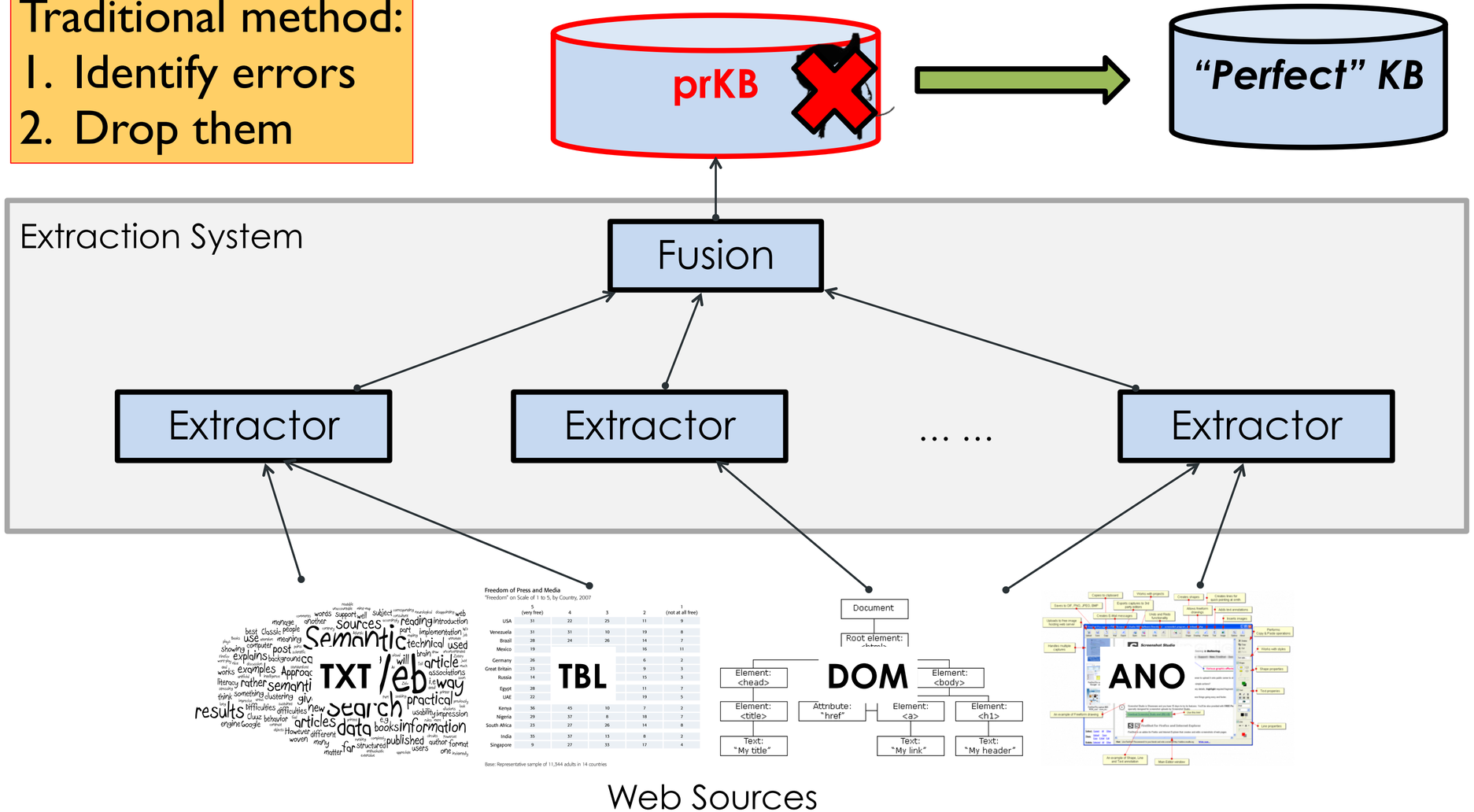
More than 70% are wrong



[SIGMOD 2015]

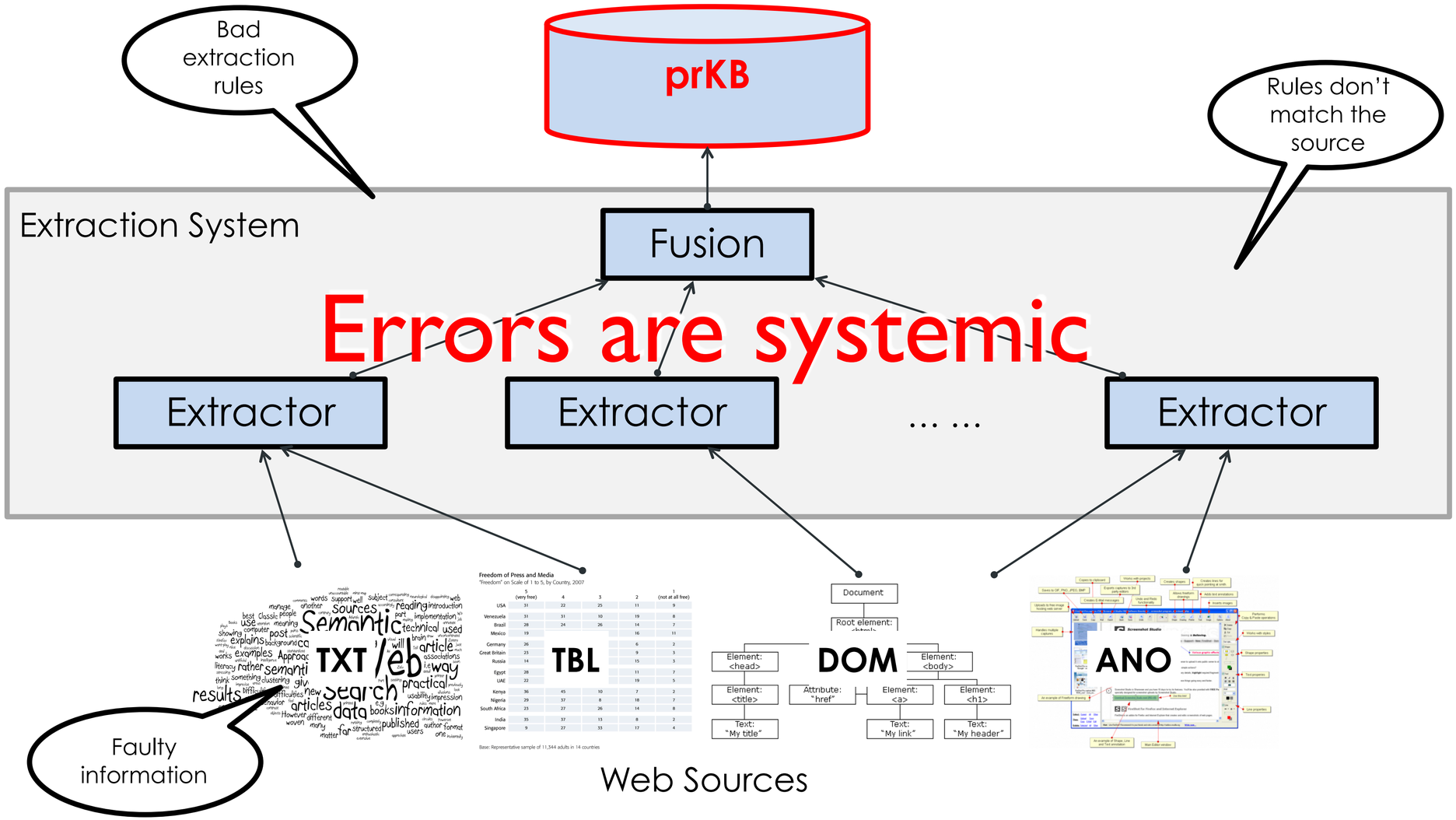
traditional data cleaning

Traditional method:
1. Identify errors
2. Drop them



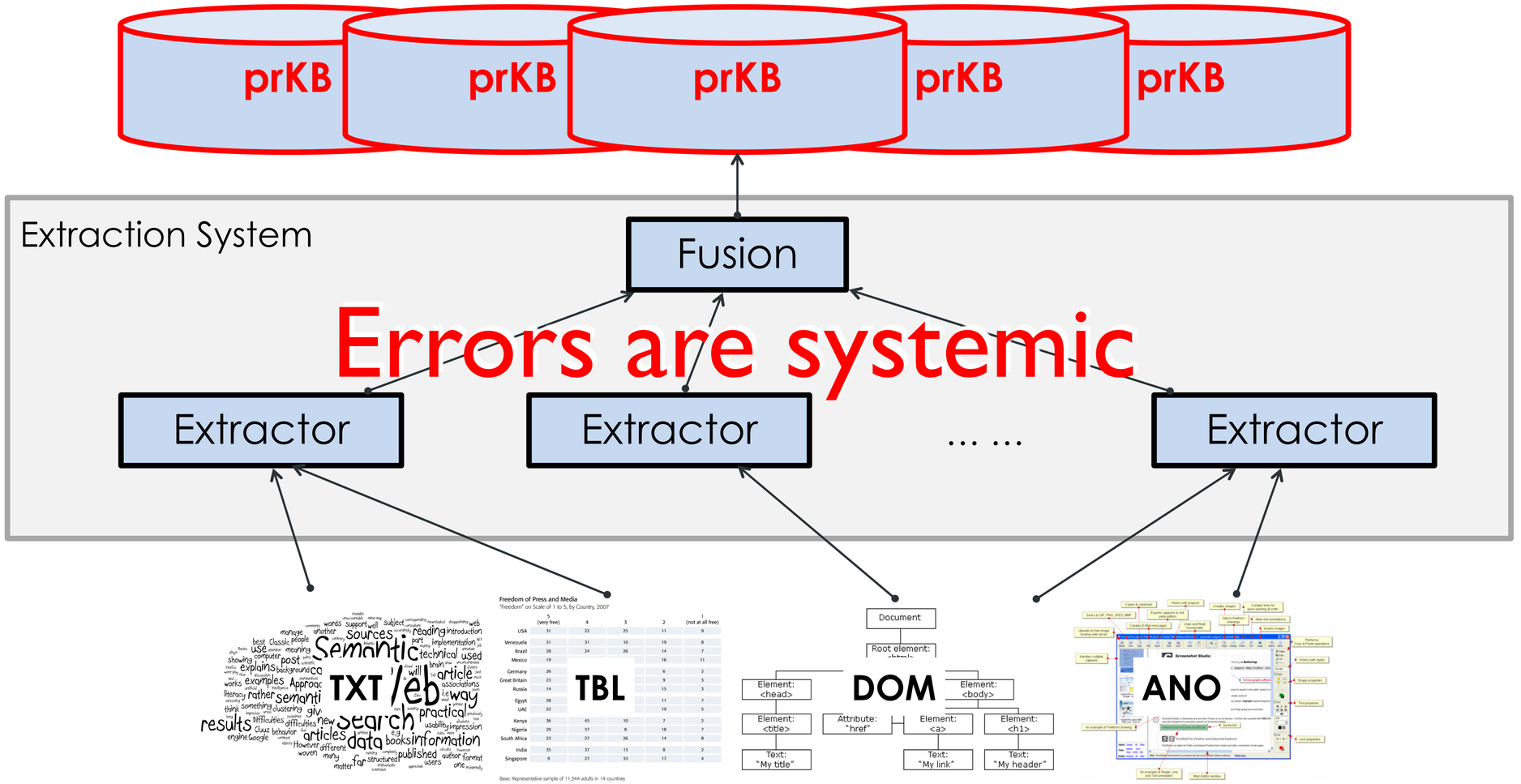
[SIGMOD 2015]

but the errors are deeper…



[SIGMOD 2015]

…continue producing bad data



[SIGMOD 2015]

CHALLENGES

- Massive scale

  Sampling loses statistical strength and misses a lot of mistakes

- System complexity

  Complex processes, thousands extraction patterns

- High error rates

DATA X-RAY [SIGMOD 2015]

Works on simple meta-data
Parallelizable in MapReduce

Example: Default value error

(besoccer.com, date_of_birth, 1986-02-18)

Triples 630

Error Rate 100%

Context: Date of birth of athletes extracted from besoccer.com is set to default value 1986-02-18, due to copied html segments

Is this feature selection?

No! The objective is different.

NPGSQL INTERMITTENT FAILURE: CONCURRENCY BUG

npgsql / npgsql

Used by 10.7k | Watch 175 | Star 2k | Fork 628

Code | Issues 167 | Pull requests 33 | Actions | Security 0 | Insights

Race condition in PoolManager.TryGetValue #2485

New issue

Closed **thetranman** opened this issue on May 29, 2019 · 3 comments

thetranman commented on May 29, 2019 — Contributor

Steps to reproduce

I've created a test that can reproduce the issue. All you have to do is fill in the values for the connection string. The test is VolatileTest as seen here:
https://github.com/thetranman/npgsql/pull/1/files

The issue

Could be related to: #2146

In our production code, we are running into issues when trying to create a new Postgres connection (Specifically when we call: var connection = new NpgsqlConnection(ConnectionString);).

This can intermittently occur when we are trying to start our service on a server which can contain

Assignees
thetranman

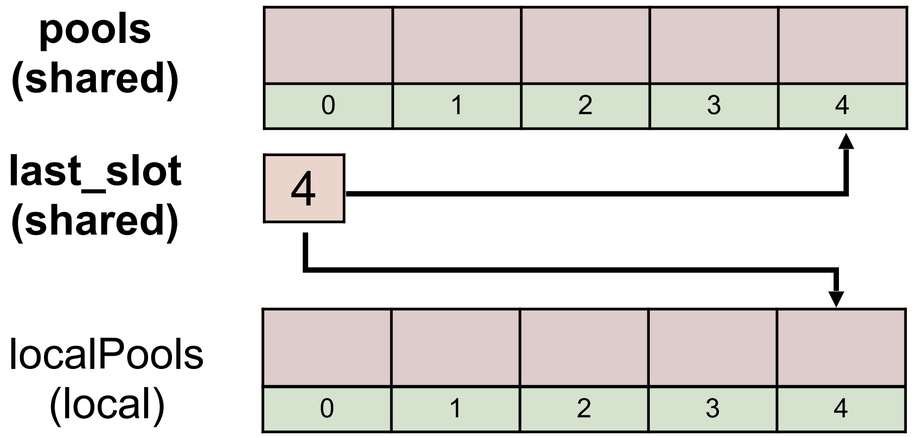
Labels
bug

Projects
None yet

Milestone
4.0.8

Linked pull requests

NPGSQL INTERMITTENT FAILURE: CONCURRENCY BUG



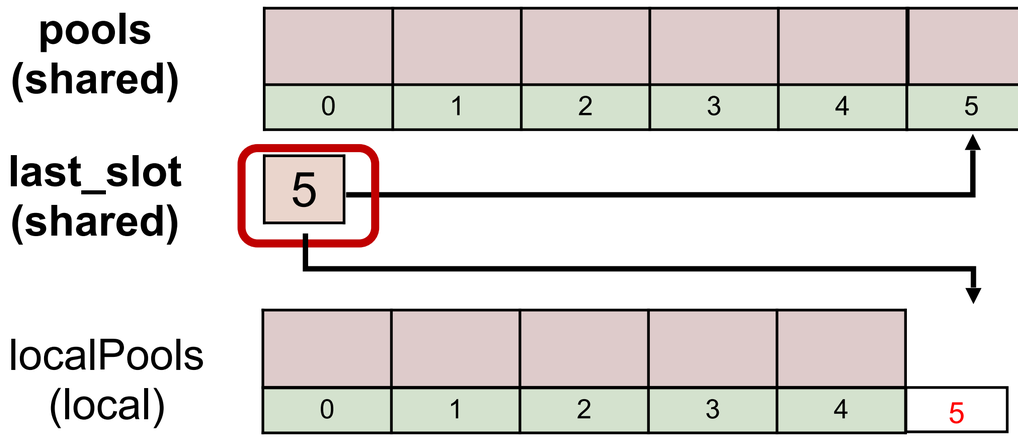
Thread 1

```
Find(key):
1   localPools = pools
```

Thread 2

```
Add(key):
```

NPGSQL INTERMITTENT FAILURE: CONCURRENCY BUG



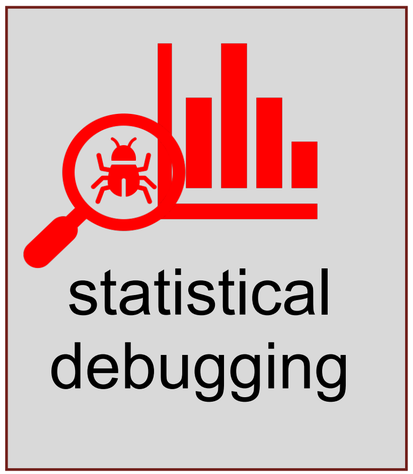
Thread 1

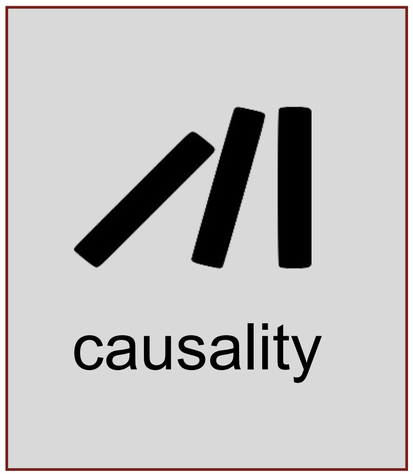
```
Find(key):

1   localPools = pools

3   for i in range(0,last_slot+1):
        if (localPools[i] == key)
            return i

    return null
```

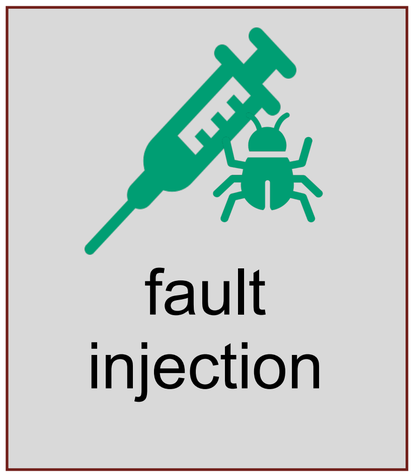
Thread 2

```
Add(key):

    if pools_is_filled:
        pools = ResizeDouble(pools)

    last_slot ++
    pools[last_slot] = key
```

2







AID: Adaptive Interventional Debugging

[SIGMOD 2020]

type of analytics

descriptive

What happened?

data mining

diagnostic

Why did it happen?

causal reasoning

predictive

What will happen?

simulation & statistics

prescriptive

What should happen?

optimization

Constrained Decision-Making

optimization

prescriptive

What should happen?

Cheapest diet plan satisfying nutrition requirements

Most effective collection and storage strategy for perishable blood supply

Most efficient plan for radiotherapy treatments in limited-capacity facility

Minimal delinquent consumer credit loss under limits on collection effort

Maximal expected return under upper bound on acceptable risk

Flight routes and crew assignments that minimize delays under limits on budget and available routes

Most profitable product bundles for given manufacturing cost

And many more...

[example from Wikipedia]

```
set Plants;
set Markets;

# Capacity of plant p in cases
param Capacity{p in Plants};

# Demand at market m in cases
param Demand{m in Markets};

# Distance in thousands of miles
param Distance{Plants, Markets};

# Freight in dollars per case per thousand miles
param Freight;

# Transport cost in thousands of dollars per case
param TransportCost{p in Plants, m in Markets} := Freight * Distance[p, m] / 1000;

# Shipment quantities in cases
var shipment{Plants, Markets} >= 0;

# Total transportation costs in thousands of dollars
minimize cost: sum{p in Plants, m in Markets} TransportCost[p, m] * shipment[p, m];

# Observe supply limit at plant p
s.t. supply{p in Plants}: sum{m in Markets} shipment[p, m] <= Capacity[p];

# Satisfy demand at market m
s.t. demand{m in Markets}: sum{p in Plants} shipment[p, m] >= Demand[m];

data;

set Plants := seattle san-diego;
set Markets := new-york chicago topeka;

param Capacity :=
                seattle 350
                san-diego 600;
```

```
param Distance{Plants, Markets};

# Freight in dollars per case per thousand miles
param Freight;

# Transport cost in thousands of dollars per case
param TransportCost{p in Plants, m in Markets} := Freight * Distance[p, m] / 1000;

# Shipment quantities in cases
var shipment{Plants, Markets} >= 0;

# Total transportation costs in thousands of dollars
minimize cost: sum{p in Plants, m in Markets} TransportCost[p, m] * shipment[p, m];

# Observe supply limit at plant p
s.t. supply{p in Plants}: sum{m in Markets} shipment[p, m] <= Capacity[p];

# Satisfy demand at market m
s.t. demand{m in Markets}: sum{p in Plants} shipment[p, m] >= Demand[m];

data;

set Plants := seattle san-diego;
set Markets := new-york chicago topeka;

param Capacity :=
                seattle 350
                san-diego 600;

param Demand :=
                new-york 325
                chicago 300
                topeka 275;

param Distance : new-york chicago topeka :=
                seattle 2.5 1.7 1.8
                san-diego 2.5 1.8 1.4;

param Freight := 90;
```

[example from Wikipedia]

In-Database Analytics

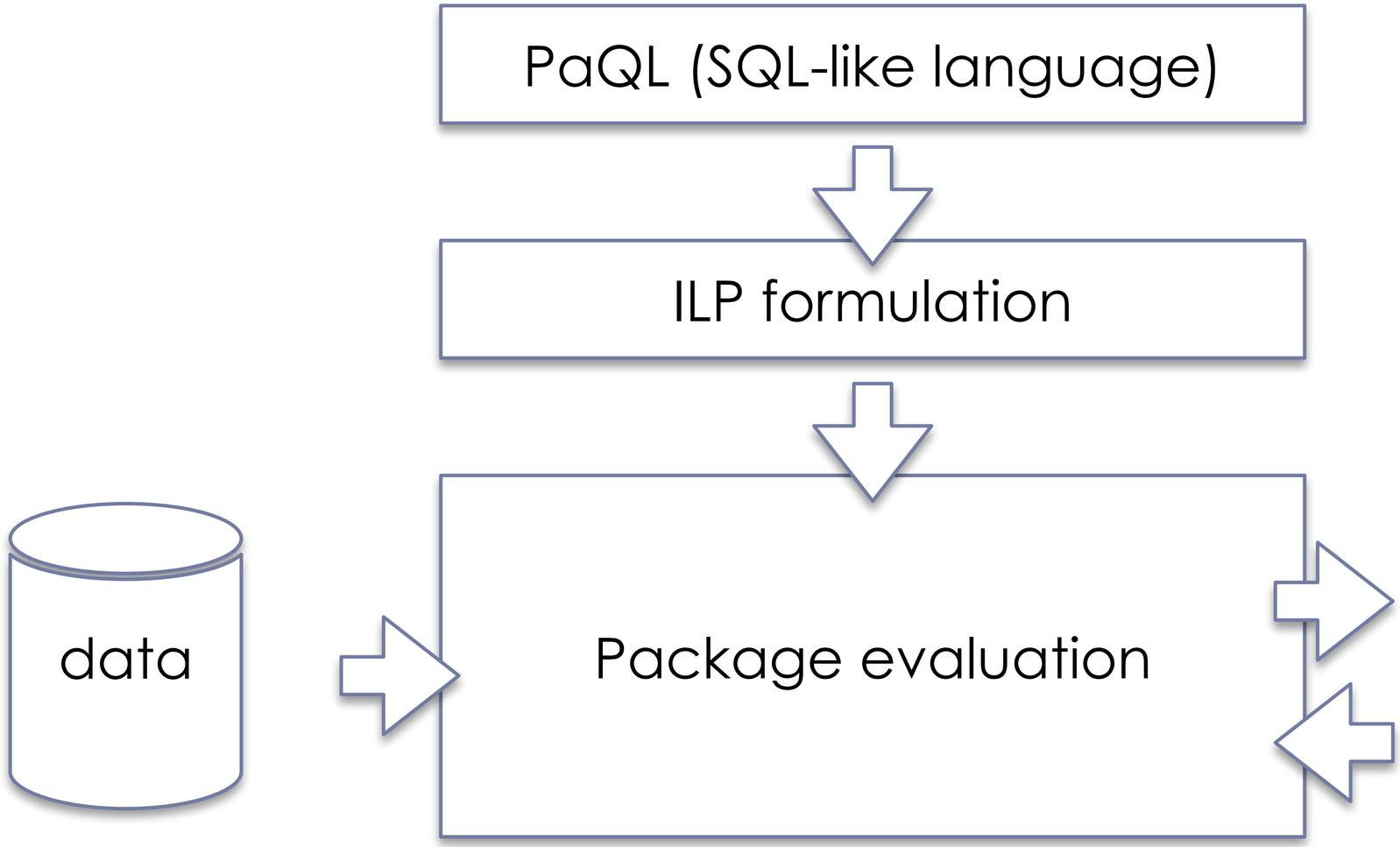
Disconnect between Data and Analytics Tools

- Cumbersome, error-prone data movement between tools (optimization solvers, prediction models) and DBMS
- Solutions are problem-specific and do not generalize

Advantages of In-Database Analytics

- Efficient, seamless, and agile analytics workflow
- Database functionality "for free"
  - Consistency, integrity, access control, efficient retrieval, …
- Applicable to a wide range of problems

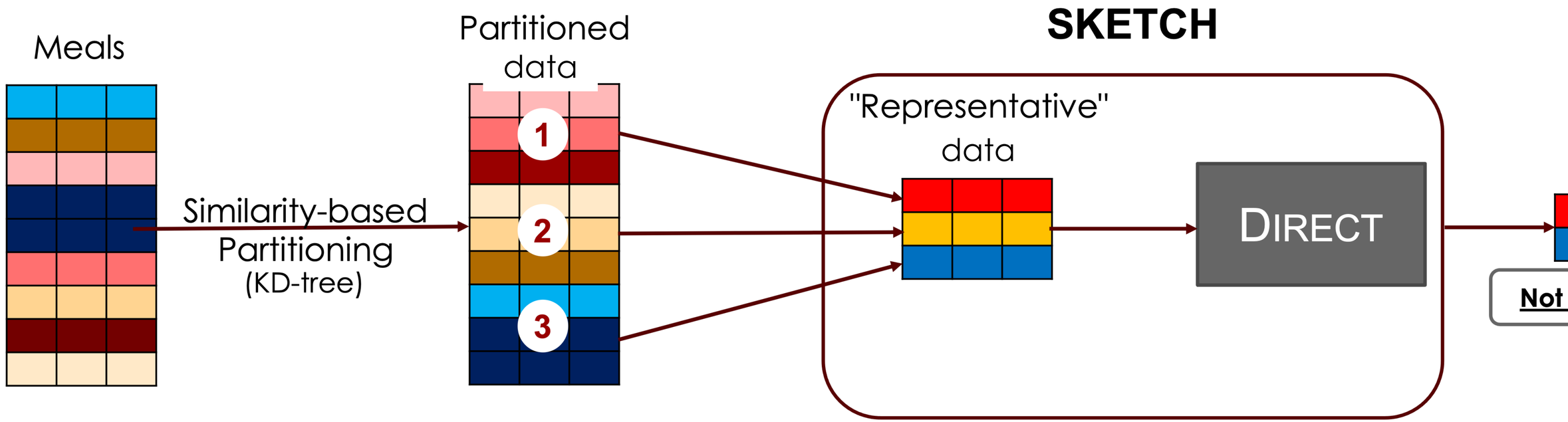
package queries



GUROBI
OPTIMIZATION

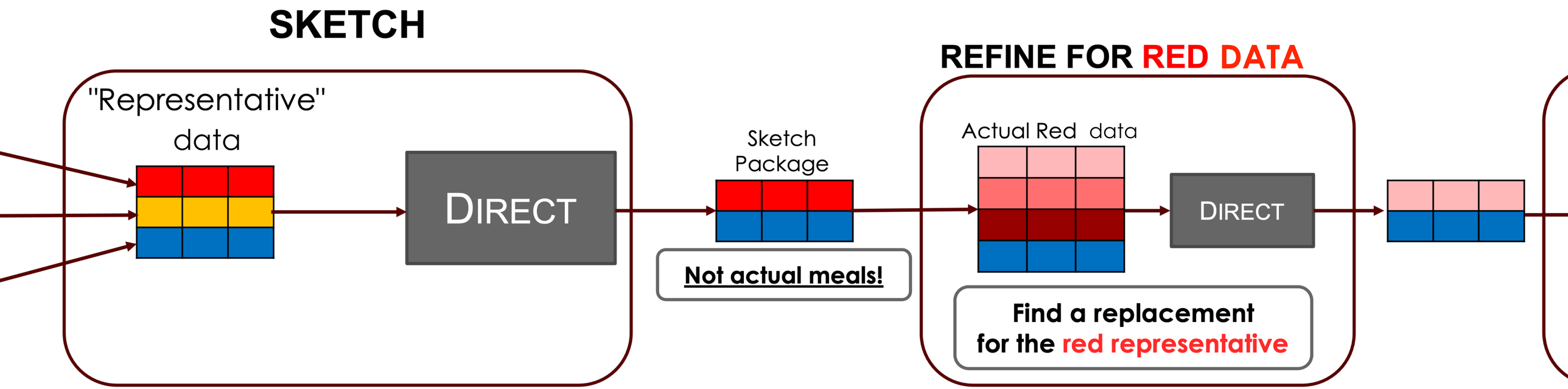
SketchRefine for scalable PaQL evaluation

VLDB 16 — SIGMOD Record 17 — VLDBJ 18 — CACM 19



SketchRefine for scalable PaQL evaluation

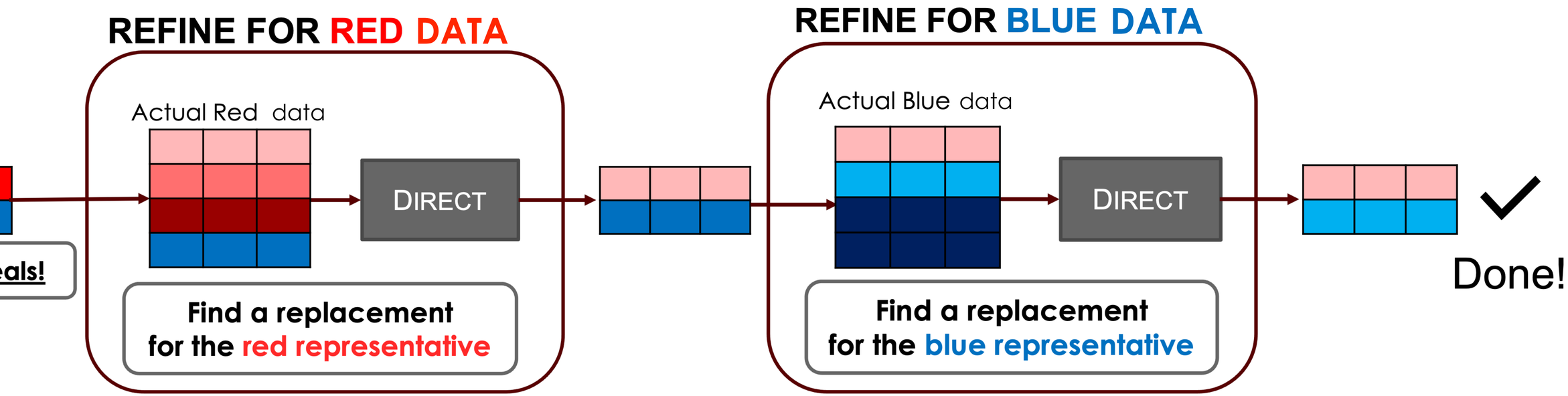
VLDB 16 — SIGMOD Record 17 — VLDBJ 18 — CACM 19



SketchRefine for scalable PaQL evaluation

VLDB 16 — SIGMOD Record 17 — VLDBJ 18 — CACM 19

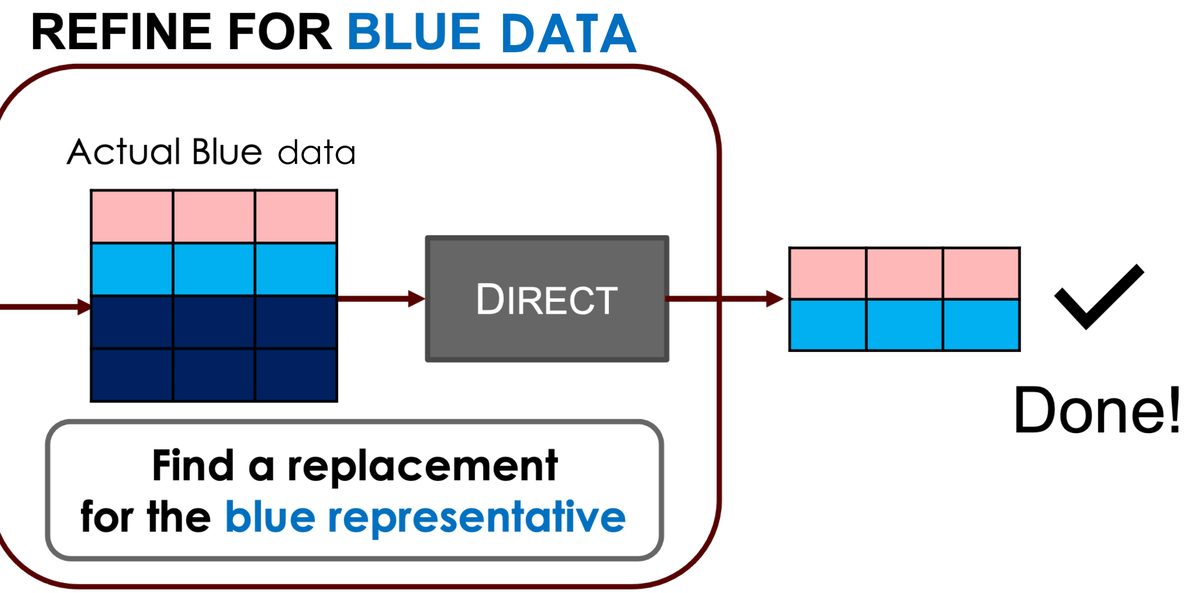
Solution is (1+ε)-approximate wrt. DIRECT over input table
(if partitioning obeys size and diameter constraints)

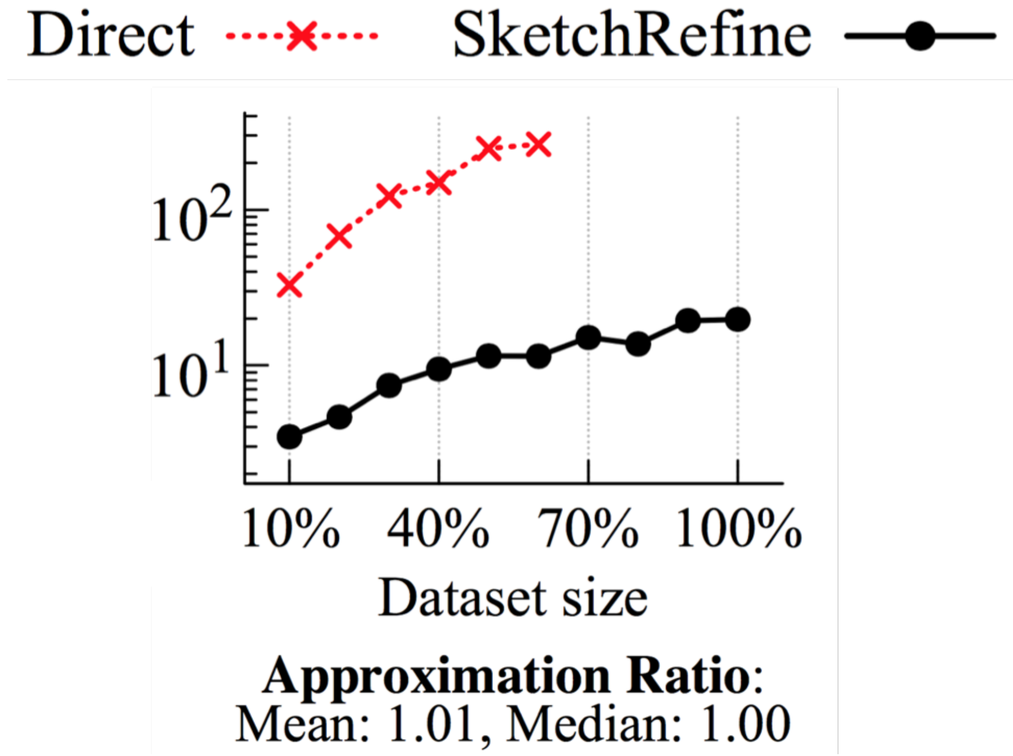


SketchRefine for scalable PaQL evaluation

VLDB 16 — SIGMOD Record 17 — VLDBJ 18 — CACM 19

Solution is (1+ε)-approximate wrt. DIRECT over input table
(if partitioning obeys size and diameter constraints)





Systems for analytics

- Need to scale to big data!
- Parallel DBs
- MapReduce

acyclic data flow; not efficient for reusing working set of data

- Spark

need for iterative and interactive applications

What is Apache Spark

- Parallel execution engine for big data
  - Implements BSP (Bulk Synchronous Processing) model
- Data abstraction: Resilient Distributed Datasets (RDDs)
  - Sets of objects partitioned & distributed across a cluster
  - Stored in RAM or on Disk
- Automatic recovery based on lineage of bulk transformations

Fill in your SRTIs!

http://owl.umass.edu/partners/courseEvalSurvey/uma/