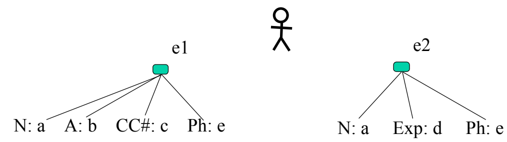
Generic Entity Resolution: Identifying Real-World Entities in Large Data Sets

Steven Whang

Stanford University

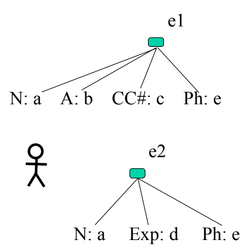
Work with: Hector Garcia-Molina, David Menestrina, Omar Benjelloun, Qi Su, Jennifer Widom, Tyson Condie, Nicolas Pombourcq, Tait Larson, Johnson Gong, Makoto Tachibana, Sutthipong Thavisomboon, Georgia Koutrika, Martin Theobald

Entity Resolution



Applications

- comparison shopping
- mailing lists
- classified ads
- customer files
- counter-terrorism



Outline

- Why is ER challenging?
- How is ER done?
- Some ER work at Stanford
  - Generic ER
  - Distributed ER
  - Iterative Blocking

Challenges (1)

- No keys!
- Value matching
  - "Kaddafi", "Qaddafi", "Kadafi", "Kaddaffi"...
- Record matching

- Nm: Tom
- Ad: 123 Main St
- Ph: (650) 555-1212
- Ph: (650) 777-7777

- Nm: Thomas
- Ad: 132 Main St
- Ph: (650) 555-1212

Challenges (2)

- Merging records

- Nm: Tom
- Ad: 123 Main St
- Ph: (650) 555-1212
- Ph: (650) 777-7777

- Nm: Thomas
- Ad: 132 Main St
- Ph: (650) 555-1212
- Zp: 94305

- Nm: Tom
- Nm: Thomas
- Ad: 123 Main St
- Ph: (650) 555-1212
- Ph: (650) 777-7777
- Zp: 94305

Challenges (3)

- Chaining

Nm: Tom
Ad: 123 Main
BD: Jan 1, 85
Wk: IBM

Nm: Thomas
Ad: 123 Maim
Oc: lawyer

Nm: Tom
Wk: IBM
Oc: laywer
Sal: 500K

Nm: Tom
Ad: 123 Main
BD: Jan 1, 85
Wk: IBM
Oc: lawyer

Nm: Tom
Ad: 123 Main
BD: Jan 1, 85
Wk: IBM
Oc: lawyer
Sal: 500K

Challenges (4)

- Un-merging

Nm: Tom
Ad: 123 Main
BD: Jan 1, 85
Wk: IBM
Oc: lawyer
Sal: 500K

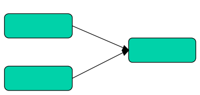
too young to make
500K at IBM!!

Challenges (5)

- Confidences in data

(0.8) Nm: Tom (0.9)
Ad: 123 Main St (1.0)
Ph: (650) 555-1212 (0.6)
Ph: (650) 777-7777 (0.8)

- In value matching, match rules, merge:



Taxonomy

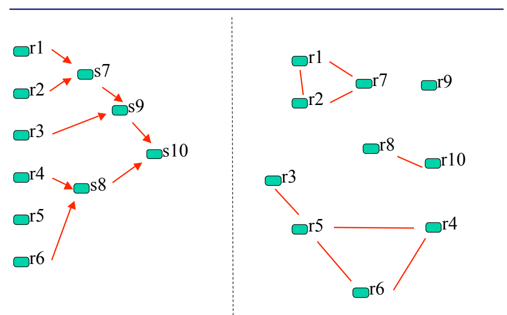
- Pairwise snaps vs. clustering
- De-duplication vs. fidelity enhancement
- Schema differences
- Relationships
- Exact vs. approximate
- Generic vs application specific
- Confidences

Schema Differences

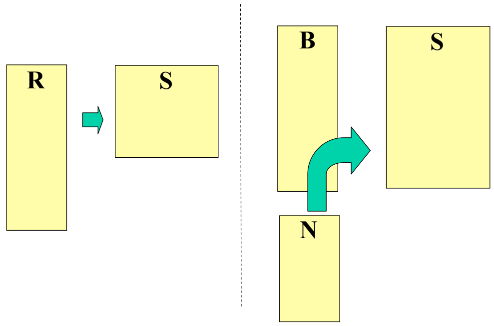
Name: Tom
Address: 123 Main St
Ph: (650) 555-1212
Ph: (650) 777-7777

FirstName: Tom
StreetName: Main St
StreetNumber: 123
Tel: (650) 777-7777

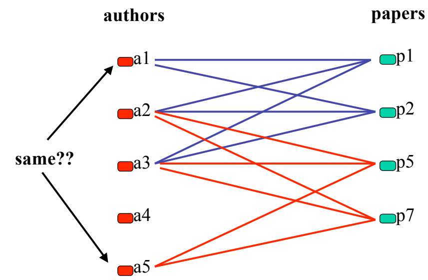
Pair-Wise Snaps vs. Clustering



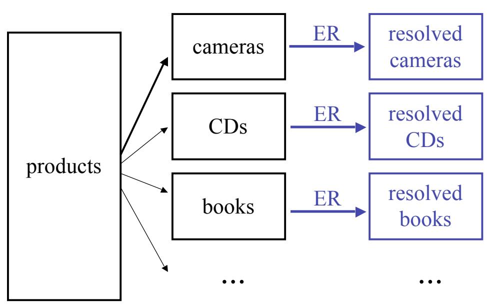
De-Duplication vs. Fidelity Enhancement



Relationships



Exact vs Approximate ER



Generic vs Application Specific

- Match function M(r, s)
- Merge function <r, s> => t

Taxonomy

- Pairwise snaps vs. clustering
- De-duplication vs. fidelity enhancement
- Schema differences
- Relationships
- Exact vs. approximate
- Generic vs application specific
- Confidences

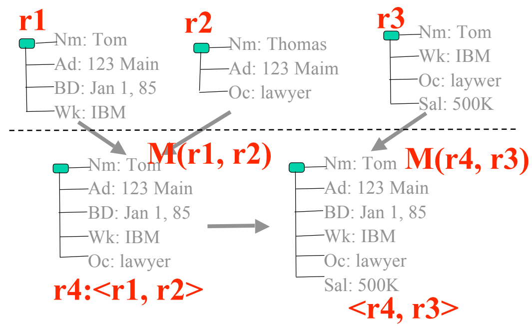
Outline

- Why is ER challenging?
- How is ER done?
- Some ER work at Stanford
  - Generic ER
  - Distributed ER
  - Iterative Blocking

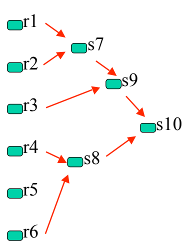
Taxonomy

- Pairwise snaps vs. clustering
- De-duplication vs. fidelity enhancement
- Schema differences No
- Relationships No
- Exact vs. approximate
- Generic vs application specific
- Confidences No

Model



Correct Answer



ER(R) = All derivable records.....

Minus “dominated” records

Question

- What is best sequence of match, merge calls that give us right answer?

Brute Force Algorithm

- Input R:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c: 4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]

Brute Force Algorithm

- Input R:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c: 4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
- Match all pairs:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c: 4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
  - r12 = [a:1, b:2, c:4, e:5]

Brute Force Algorithm

- Match all pairs:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c: 4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
  - r12 = [a:1, b:2, c:4, e:5]
- Repeat:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c: 4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
  - r12 = [a:1, b:2, c:4, e:5]
  - r123 = [a:1, b:2, c:4, e:5, f:6]

25

Question # 1

Brute Force Algorithm

- Match all pairs:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c: 4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
  - r12 = [a:1, b:2, c:4, e:5]
- Repeat:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c: 4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
  - r12 = [a:1, b:2, c:4, e:5]
  - r123 = [a:1, b:2, c:4, e:5, f:6]

Can we avoid comparisons?

26

Question # 2

Brute Force Algorithm

- Input R:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c: 4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
- Match all pairs:
  - r1 = [a:1, b:2]
  - r2 = [a:1, c: 4, e:5]
  - r3 = [b:2, c:4, f:6]
  - r4 = [a:7, e:5, f:6]
  - r12 = [a:1, b:2, c:4, e:5]

Can we delete r1, r2?

27

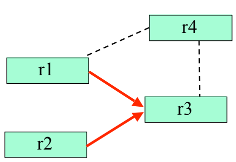
ICAR Properties

- Idempotence:
  - M(r1, r1) = true; <r1, r1> = r1
- Commutativity:
  - M(r1, r2) = M(r2, r1)
  - <r1, r2> = <r2, r1>
- Associativity
  - <r1, <r2, r3>> = <<r1, r2>, r3>

28

More Properties

- Representativity
  - If <r1, r2> = r3, then for any r4 such that M(r1, r4) is true we also have M(r3, r4) = true.



29

ICAR Properties ➨ Efficiency

- Commutativity
- Idempotence
- Associativity
- Representativity

- Can discard records
- ER result independent of processing order

30

Swoosh Algorithms

- R-Swoosh
  - Merges records as soon as they match
  - Optimal in terms of record comparisons
- F-Swoosh
  - Remembers values seen for each feature
  - Avoids redundant value comparisons

31

R-Swoosh

R	R’
r1 r2 r3 r4	

M(r1,r2)=true

M(r12,r3)=true

32

R-Swoosh

R	R’
r2 r3 r4	r1

M(r1,r2)=true

M(r12,r3)=true

33

R-Swoosh

R	R’
r12 r3 r4	

M(r1,r2)=true

M(r12,r3)=true

34

R-Swoosh

R	R’
r3 r4	r12

M(r1,r2)=true

M(r12,r3)=true

35

R-Swoosh

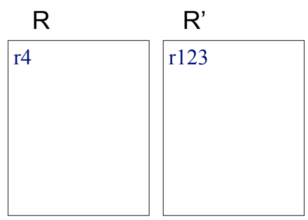
R	R’
r123 r4	

M(r1,r2)=true

M(r12,r3)=true

36

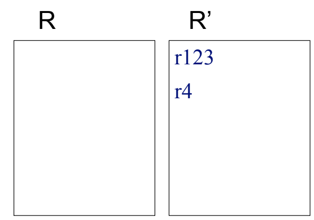
R-Swoosh



M(r1,r2)=true

M(r12,r3)=true

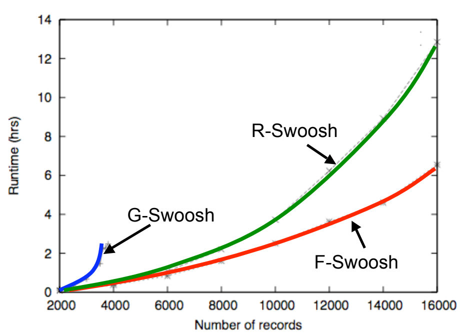
R-Swoosh



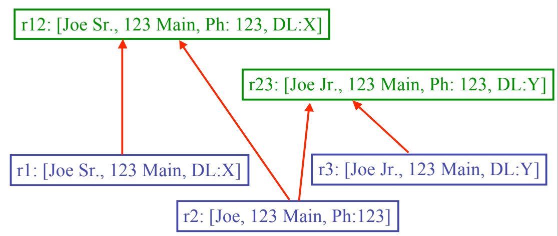
M(r1,r2)=true

M(r12,r3)=true

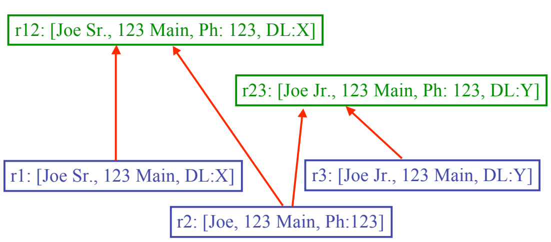
Scalability



If ICAR Properties Do Not Hold?

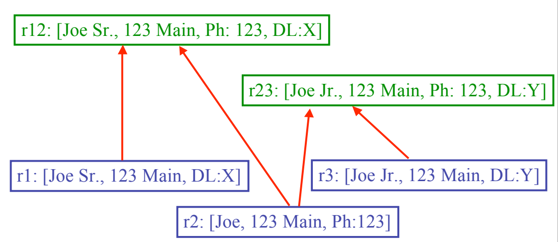


If ICAR Properties Do Not Hold?



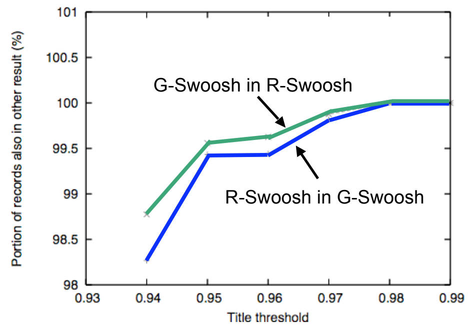
Full Answer: ER(R) = {r12, r23, r1, r2, r3}
Minus Dominated: ER(R) = {r12, r23}

If ICAR Properties Do Not Hold?



Full Answer: ER(R) = {r12, r23, r1, r2, r3}
Minus Dominated: ER(R) = {r12, r23}
R-Swoosh Yields: ER(R) = {r12, r3} or {r1, r23}

Swoosh Without ICAR Properties



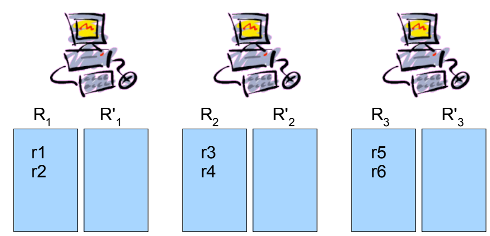
Generic ER Summary

- Entity resolution is critical
- Generic approach yields reusable techniques
- Efficient resolution is important

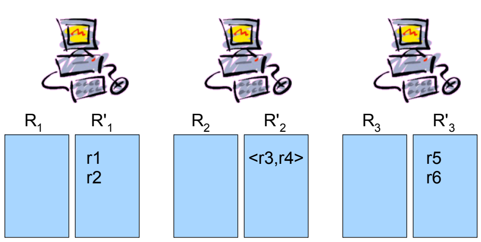
Outline

- Why is ER challenging?
- How is ER done?
- Some ER work at Stanford
  - Generic ER
  - Distributed ER ⬅
  - Iterative Blocking

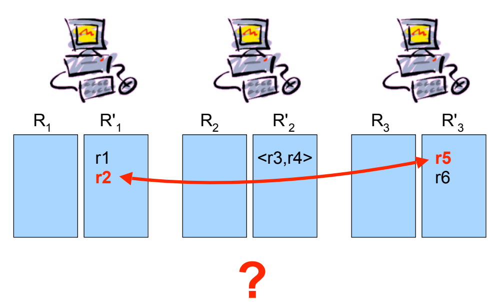
Partition Input Data



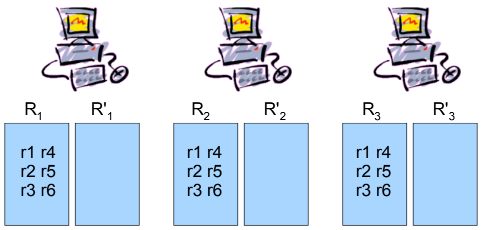
Partition Input Data



Partition Input Data

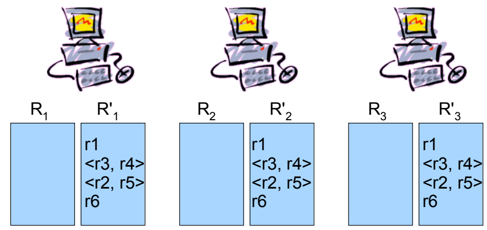


Partition Input Data



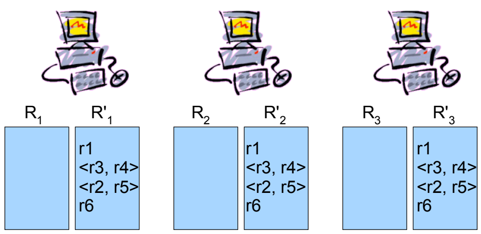
49

Partition Input Data



50

Partition Input Data



Redundant work!

51

Scope and Responsible

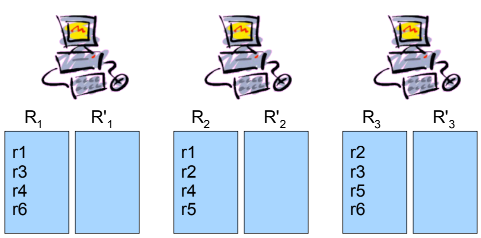
- *scope*(r) – Returns list of processors that receive r
- *resp*(p_k, r_i, r_j) – true if processor p_k should perform comparison of r_i and r_j

Coverage property:

For any pair of matching records r, r', there exists at least one processor P_k such that $P_k \in$ *scope*(r)∩*scope*(r') and *resp*(P_k, r, r') = true.

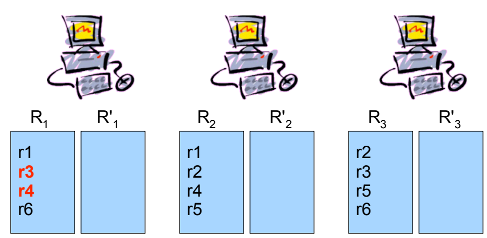
52

D-Swoosh



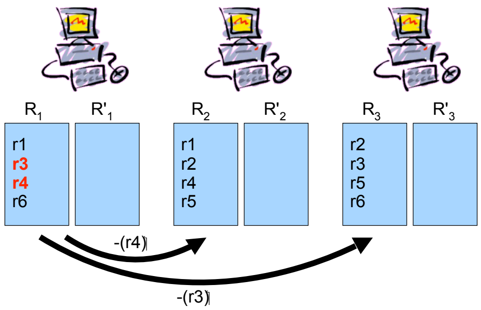
53

D-Swoosh

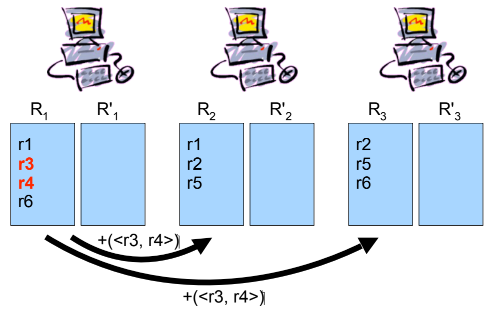


54

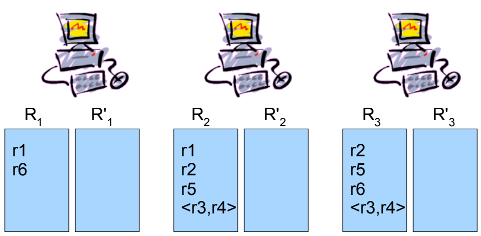
D-Swoosh



D-Swoosh



D-Swoosh



Defining *scope and resp*

- What are good *scope* and *resp* functions?
- How do different functions compare and scale?
- How can we exploit semantic knowledge?

Full Replication

p1	p2	p3
r1	r1	r1
r2	r2	r2
r3	r3	r3
r4	r4	r4
r5	r5	r5
r6	r6	r6

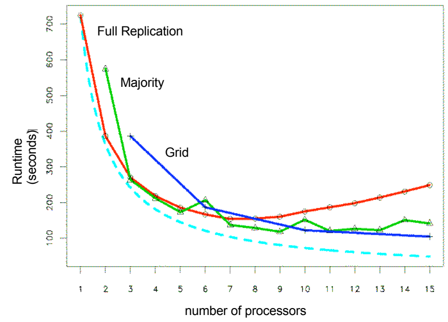
Majority

p1	p2	p3
r1	r1	
	r2	r2
r3		r3
r4	r4	
	r5	r5
r6		r6

Grid

	B1	B2	B3
B1	p_1		
B2	p_2	p_3	
B3	p_4	p_5	p_6

Experimental Results



Distributed ER Summary

- Entity Resolution is fundamentally expensive
- Reduce processing time by:
  - Distributing data
  - Eliminating redundant work
- ER benefits greatly from distributed computing

Outline

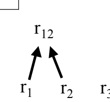
- Why is ER challenging?
- How is ER done?
- Some ER work at Stanford
  - Generic ER
  - Distributed ER
  - Iterative Blocking ⬅

Entity Resolution

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

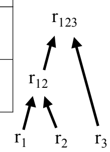
Entity Resolution

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo
r_{12}	John Doe	{12345, 94305}	jdoe@yahoo



Entity Resolution

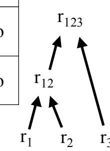
Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo
r_{12}	John Doe	{12345, 94305}	jdoe@yahoo
r_{123}	{John Doe, J. Foe}	{12345, 94305}	jdoe@yahoo



67

Entity Resolution

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo
r_{12}	John Doe	{12345, 94305}	jdoe@yahoo
r_{123}	{John Doe, J. Foe}	{12345, 94305}	jdoe@yahoo



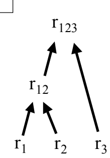
ER Solution: $\{r_{123}\}$

68

Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_1	r_2 , r_3

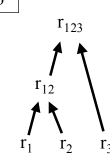


69

Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_1	r_2 , r_3
1^{st} char last name	r_1 , r_2	r_3

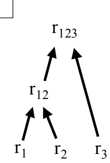


70

Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_1	r_2 , r_3
1^{st} char last name	r_{12}	r_3

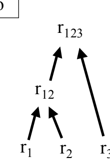


71

Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_1	r_2 , r_3
1^{st} char last name	r_{12}	r_3



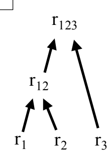
Blocking Solution: $\{r_{12}, r_3\}$

72

Iterative Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_1	r_2 , r_3
1st char last name	r_1 , r_2	r_3

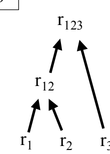


73

Iterative Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_1	r_2 , r_3
1st char last name	r_{12}	r_3

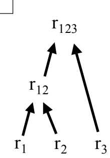


74

Iterative Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_1 , r_{12}	r_2 , r_3, r_{12}
1st char last name	r_{12}	r_3

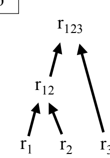


75

Iterative Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_{12}	r_{123}
1st char last name	r_{12}	r_3

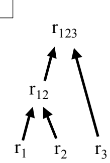


76

Iterative Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_{123}	r_{123}
1st char last name	r_{123}	r_{123}

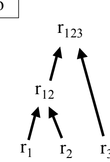


77

Iterative Blocking

Record	Name	Zip	Email
r_1	John Doe	12345	jdoe@yahoo
r_2	John Doe	94305	
r_3	J. Foe	94305	jdoe@yahoo

Partition by	Block#1	Block#2
zip	r_{123}	r_{123}
1st char last name	r_{123}	r_{123}



Iterative Blocking Solution: $\{r_{123}\}$

78

Overview

- Model
- Algorithms
- Experimental Results

79

Iterative Blocking Model

- Result is the fixed-point state of applying a "core" ER algorithm on blocks and re-distributing new records
- Can plug in any core ER algorithm that partitions records

80

In-memory Algorithm (Lego)

- Maintains "maximal records" for efficient block updates
- Maintains a queue of blocks to process

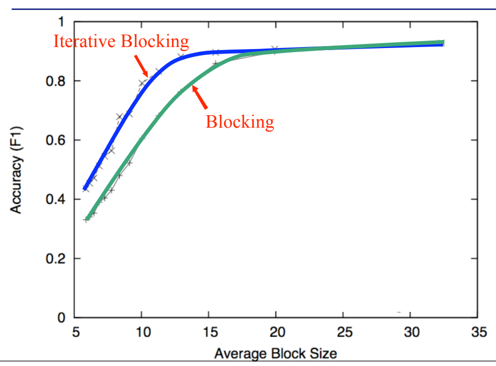
81

Disk-based Algorithm (Duplo)

- Reads into memory and processes "N blocks at a time" using segments
- Maintains a queue of segments to process
- Updates segments by scanning a merge log on disk
  - Uses timestamps to avoid a full scan for each segment read

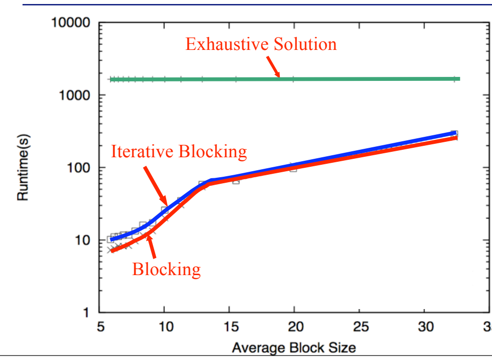
82

Accuracy



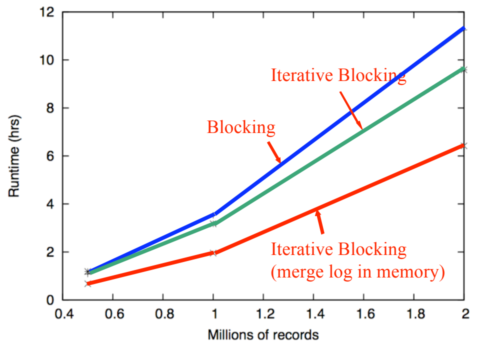
83

Performance



84

Scalability



Iterative Blocking Summary

- Proposed model & efficient algorithms (in-memory, disk) for iterative blocking
- Showed that iterative blocking can be more accurate and scalable than simple blocking

ER in the InfoLab

- Generic ER [VLDB J. 09]
- Distributed ER [ICDCS 07]
- Iterative Blocking [SIGMOD 09]
- Negative Rules [VLDB J. 09]
- Confidences [VLDB CleanDB 06]
- Evolving Rules
- Joint ER
- ER Measures
- ER in Probabilistic Databases
- Privacy and Information Leakage