Randomized Algorithms for Data Reconciliation in Wide Area Aggregate Query Processing

Fei Xu and Christopher Jermaine
CISE Department
University of Florida
September 26,2007

How to answer an aggregate query over a distributed, “dirty” database

Let us Begin with the Centralized Case

What is the total salary paid by the company?

Salary	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433
Christina	$7633
Steve	$8003
Sean	$9607
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

- If the data is “clean”, we only need to summarize all the salaries in the table.
- But we do have a concern:
  - Whether the two “Michaels” are the same person?
- If we know that the they are the same person, now we have two different salaries.
  - How can we obtain the correct salary?

Prior Researches

- Many prior researches have been done in order to answer the two questions:
  - Deciding whether different records refer to the same entity.
  - Obtaining a clean record from inconsistent duplicates.
- We want to utilize these existing results to answer the two questions:
  - Encode existing techniques into two generic functions.
  - Expect users to provide actual implementation.

Two Generic Functions

- The similarity function *Sim* ():
  - This function tells whether two records refer to the same entity, i.e. whether they are similar.
  - $Sim(r_1, r_2) = true$ iff r_1 and r_2 are similar.

- The reconciliation function *Rec* ():
  - This function takes a set of similar records and reconcile them.
  - It returns a numerical value used for aggregation.

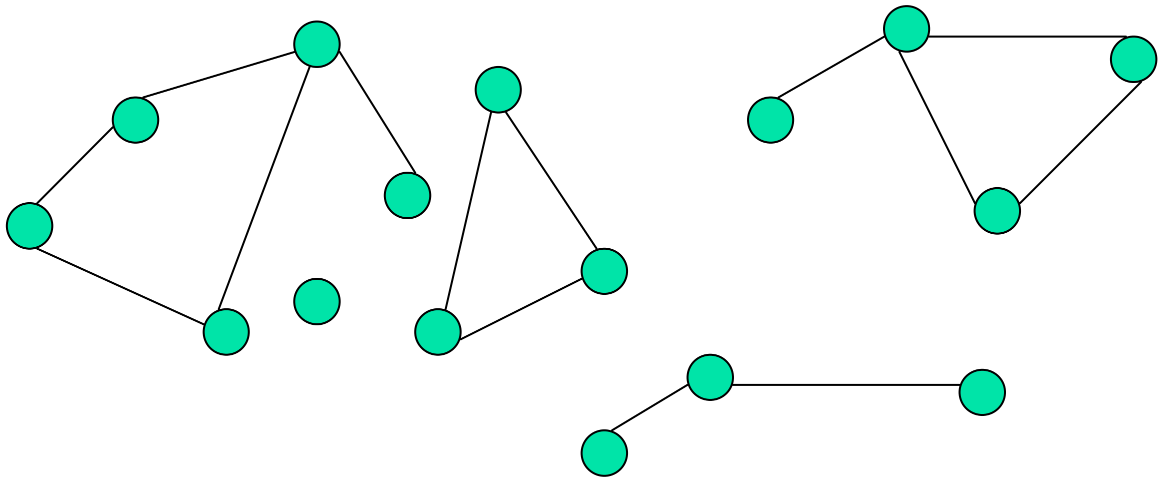
Examples

- The similarity function:
  - Ex: r_1 and r_2 are similar if and only if r_1.Name = r_2.Name.
  - *Sim*((Michael, 10000), (Michael, 9899))=*true*
  - *Sim*((Michael, 10000), (Christina, 7412))=*false*
- The reconciliation function:
  - Ex: Take the average salary.
  - *Rec*({(Michael, 10000), (Michael, 9899)})
    =(10000+9899)/2=9949.5

Utilize the Two Functions to Answer the Aggregate Query.

- Given these two functions, we could answer the aggregate query in the following three steps:
  - Using the similarity function to partition the records in the database into “equivalence classes” (will be discussed soon).
  - Apply the reconciliation function to each of the equivalence classes to reconcile the records.
  - Aggregate the returned values from each reconciliation and return the aggregate value as the answer.

Equivalence class



- If we assume each record is a vertex in a graph, and we assume there is an edge between two vertices if they are similar, then an equivalence class is a **connected component** in the graph.
- We use this definition in the paper. But alternative definitions exist. [BBS05][BG04][PD05][PMMRS05]

The Three-Step Solution

Assuming we use the definitions of *Sim*() and *Rec*() given in the example, the three-step solution is shown below:

Salary	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433
Christina	$7633
Steve	$80[illegible]
Sean	$9607
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

Michael	$10000
Michael	$9899
Christina	$7412
Christina	$7633
Daniel	$7864
David	$8433
Steve	$8003
Sean	$9607
Emily	$10822
James	$7322

(10000+ 9899)/2

(7633+ 7412)/2

7864

8433

8003

9607

10822

7322

69523

Extend to the Distributed Environment

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Steve	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

- What is the difficulty:
  - Similar records distributed in different sites.
  - The partition step requires to ship all the data to a coordinator.
  - Each site may contain a large number of records.
  - Such shipments are unaffordable- maybe gigabytes/terabytes of data.

Our Solution

- Use Approximation:
  - Sample from the set of all equivalence classes.
  - Only ship a part of the records from each sites.
  - Several randomized algorithms are proposed.
  - Provide the statistical guarantees.

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
    4. $\hat{M}_i = 0$
    5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
        6. $\hat{M}_i \mathrel{+}= \frac{1}{p_i} Rec(S)$
    7. $\hat{M} \mathrel{+}= \hat{M}_i$
8. Return $\hat{M}$

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
    4. $\hat{M}_i = 0$
    5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
        6. $\hat{M}_i \mathrel{+}= \frac{1}{p_i} Rec(S)$
    7. $\hat{M} \mathrel{+}= \hat{M}_i$
8. Return $\hat{M}$

The *SampleClasses*() Step (Logically)

- Requirements:
  - Probability of selecting an equivalence class of size i is p_i: for some constant.
  - If one record from an equivalence class is selected, all records from that equivalence class are selected.

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
    4. $\hat{M}_i = 0$
    5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
        6. $\hat{M}_i \mathrel{+}= \frac{1}{p_i} Rec(S)$
    7. $\hat{M} \mathrel{+}= \hat{M}_i$
8. Return $\hat{M}$

Michael	$10000
Michael	$9899

Emily	$10822

Daniel	$7864

Steve	$8003

$p_1 = p_2 = 0.5$

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
    4. $\hat{M}_i = 0$
    5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
        6. $\hat{M}_i += \frac{1}{p_i} Rec(S)$
    7. $\hat{M} += \hat{M}_i$
8. Return $\hat{M}$

$i=1$

Michael	$10000
Michael	$9899

Emily	$10822

Daniel	$7864

Steve	$8003

$p_1 = p_2 = 0.5$

$\hat{M}_i += 7864/0.5$

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
    4. $\hat{M}_i = 0$
    5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
        6. $\hat{M}_i += \frac{1}{p_i} Rec(S)$
    7. $\hat{M} += \hat{M}_i$
8. Return $\hat{M}$

$i = 1$

Michael	\$10000
Michael	\$9899

Emily	\$10822

Daniel	\$7864

Steve	\$8003

$p_1 = p_2 = 0.5$

$\hat{M}_i += 8003/0.5$

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
   4. $\hat{M}_i = 0$
   5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
      6. $\hat{M}_i += \frac{1}{p_i} Rec(S)$
   7. $\hat{M} += \hat{M}_i$
8. Return $\hat{M}$

$i = 1$

Michael	\$10000
Michael	\$9899

Emily	\$10822

Daniel	\$7864

Steve	\$8003

$p_1 = p_2 = 0.5$

$\hat{M}_i$ +=10822/0.5

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
    4. $\hat{M}_i = 0$
    5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
        6. $\hat{M}_i += \frac{1}{p_i} Rec(S)$
    7. $\hat{M} += \hat{M}_i$
8. Return $\hat{M}$

$i = 1$

Michael	\$10000
Michael	\$9899

Emily	\$10822

Daniel	\$7864

Steve	\$8003

$p_1 = p_2 = 0.5$

$\hat{M}$ +=53378

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
    4. $\hat{M}_i = 0$
    5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
        6. $\hat{M}_i + = \frac{1}{p_i} Rec(S)$
    7. $\hat{M} + = \hat{M}_i$
8. Return $\hat{M}$

i =2

Michael	\$10000
Michael	\$9899

Emily	\$10822

Daniel	\$7864

Steve	\$8003

p_1=p_2=0.5

$\hat{M}_i$+=((10000+9899)/2)/0.5

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
    4. $\hat{M}_i = 0$
    5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
        6. $\hat{M}_i += \frac{1}{p_i} Rec(S)$
    7. $\hat{M} += \hat{M}_i$
8. Return $\hat{M}$

$i = 2$

Michael	$10000
Michael	$9899

Emily	$10822

Daniel	$7864

Steve	$8003

$p_1 = p_2 = 0.5$

$\hat{M}$ +=19899

Basic Framework

Function $GetAnswer()$

1. $\mathbf{S}' = SampleClasses()$
2. $\hat{M} = 0$
3. For $i = 1$ to m:
   4. $\hat{M}_i = 0$
   5. For each class $S \in \mathbf{S}'$ where $|S| = i$:
      6. $\hat{M}_i \mathrel{+}= \frac{1}{p_i} Rec(S)$
   7. $\hat{M} \mathrel{+}= \hat{M}_i$
8. Return $\hat{M}$

Michael	\$10000
Michael	\$9899

Emily	\$10822

Daniel	\$7864

Steve	\$8003

$p_1=p_2=0.5$

$\hat{M}$ =73277

Accuracy Guarantees

- The variance of $\hat{M}_i$ is:

$$var(\hat{M}_i) = (\frac{1}{p_i} - 1) \sum_{S \in \mathbf{S} \wedge |S| = i} Rec^2(S)$$

- The variance of $\hat{M}$ is:

$$var(\hat{M}) = \sum var(\hat{M}_i)$$

- An estimator of $var(\hat{M}_i)$ is:

$$v\hat{a}r(\hat{M}_i) = \frac{1}{p_i}(\frac{1}{p_i} - 1) \sum_{S \in \mathbf{S}' \wedge |S| = i} Rec^2(S)$$

- An estimator of $var(\hat{M})$ is:

$$v\hat{a}r(\hat{M}) = \sum_i v\hat{a}r(\hat{M}_i)$$

- Central Limit Theorem based or more conservative Chebyshev confidence bound can be provided.

First Big Question

- How to implement the *SampleClasses* () function?
  - Related to the similarity function.

Case 1:Transitive Similarity Function

$$(Sim(r_1, r_2) = Sim(r_2, r_3) = true) \Rightarrow (Sim(r_1, r_3) = true)$$

- Each connected component is a **clique** if *Sim*() is transitive .
- Canonical form exists for each equivalence class.
  - William: {Will, Bill, William, Wm., Billy, Willy, Willie}
- The canonical form of each equivalence class can be pre-stored at each site.
- Hash Bernoulli algorithm works for this case.

The Hash Bernoulli Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Steve	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p*.

2. Choose a Hash function *H* (*r*), which takes a record and returns a value between 0 and 1.

3. For every record in each site, hash the canonical form of the record.

4. Ship all records whose hash value <= *p*.

The Hash Bernoulli Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Steve	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

$p = 0.5$

1. Choose a probability p.

2. Choose a Hash function $H(r)$, which takes a record and returns a value between 0 and 1.

3. For every record in each site, hash the canonical form of the record.

4. Ship all records whose hash value $<= p$.

The Hash Bernoulli Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Steve	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p*.

2. Choose a Hash function *H*(*r*), which takes a record and returns a value between 0 and 1.

3. For every record in each site, hash the canonical form of the record.

4. Ship all records whose hash value <= *p*.

The Hash Bernoulli Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

0.3

*R*2:Chicago	
Name	Salary
Christina	$7633
Steve	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability p.

2. Choose a Hash function $H(r)$, which takes a record and returns a value between 0 and 1.

3. For every record in each site, hash the canonical form of the record.

4. Ship all records whose hash value <= p.

The Hash Bernoulli Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

0.3	
0.7	

*R*2:Chicago	
Name	Salary
Christina	$7633
Steve	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p*.

2. Choose a Hash function *H* (*r*), which takes a record and returns a value between 0 and 1.

3. For every record in each site, hash the canonical form of the record.

4. Ship all records whose hash value <= *p*.

The Hash Bernoulli Algorithm

*R*1:New York		
Name	Salary	
Michael	$10000	0.3
Daniel	$7864	0.7
David	$8433	0.6

*R*2:Chicago		
Name	Salary	
Christina	$7633	0.6
Steve	$8003	0.1
Sean	$9607	0.9

*R*3:Vienna		
Name	Salary	
Emily	$10822	0.2
James	$7322	0.7
Michael	$9899	0.3
Christina	$7412	0.6

1. Choose a probability p.

2. Choose a Hash function $H(r)$, which takes a record and returns a value between 0 and 1.

3. For every record in each site, hash the canonical form of the record.

4. Ship all records whose hash value <= p.

The Hash Bernoulli Algorithm

*R*1:New York		Hash
Name	Salary	
Michael	$10000	0.3
Daniel	$7864	0.7
David	$8433	0.6

*R*2:Chicago		Hash
Name	Salary	
Christina	$7633	0.6
Steve	$8003	0.1
Sean	$9607	0.4

*R*3:Vienna		Hash
Name	Salary	
Emily	$10822	0.2
James	$7322	0.7
Michael	$9899	0.3
Christina	$7412	0.6

1. Choose a probability p.

2. Choose a Hash function $H(r)$, which takes a record and returns a value between 0 and 1.

3. For every record in each site, hash the canonical form of the record.

4. Ship all records whose hash value <= p.

More on Hash Bernoulli Algorithm

*R*1:New York		
Name	Salary	
Michael	$10000	0.3
Daniel	$7864	0.7
David	$8433	0.6

*R*2:Chicago		
Name	Salary	
Christina	$7633	0.6
Steve	$8003	0.1
Sean	$9607	0.4

*R*3:Vienna		
Name	Salary	
Emily	$10822	0.2
James	$7322	0.7
Michael	$9899	0.3
Christina	$7412	0.6

- Records in the same equivalence class have the same hash value.
- Once a record in an equivalence class is sampled out, every record in this class is sampled out.
- The probability to sample each equivalence class is *p*.

Case 2: Non-transitive Similarity Function

- An equivalence class is **not** a clique.
- Canonical form does not exists and cannot be pre-stored.
- Require more complicated algorithms:
  - The uniform-*p* algorithm.
    - Essentially this is a distributed transitive closure algorithm.
  - The diminishing-*p* algorithm.

The Uniform-*p* Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Michael	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p.*

2. For every record in each site, with probability *p*, it is selected as a seed independently.

3. Each site ships all its seeds to the coordinator.

4. The whole seeds set is sent back to every site.

5. Every site returns records that are similar to one of the records in the seed set. These records are added into the seeds set.

6. Repeat 4 and 5 until no more records is returned.

The Uniform-*p* Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Michael	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

p=0.5

1. Choose a probability *p*.

2. For every record in each site, with probability *p*, it is selected as a seed independently.

3. Each site ships all its seeds to the coordinator.

4. The whole seeds set is sent back to every site.

5. Every site returns records that are similar to one of the records in the seed set. These records are added into the seeds set.

6. Repeat 4 and 5 until no more records is returned.

The Uniform-*p* Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Michael	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p.*

2. For every record in each site, with probability *p*, it is selected as a seed independently.

3. Each site ships all its seeds to the coordinator.

4. The whole seeds set is sent back to every site.

5. Every site returns records that are similar to one of the records in the seed set. These records are added into the seeds set.

6. Repeat 4 and 5 until no more records is returned.

The Uniform-*p* Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Michael	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p*.

2. For every record in each site, with probability *p*, it is selected as a seed independently.

3. Each site ships all its seeds to the coordinator.

4. The whole seeds set is sent back to every site.

5. Every site returns records that are similar to one of the records in the seed set. These records are added into the seeds set.

6. Repeat 4 and 5 until no … ds is returned.

Seeds

The Uniform-*p* Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Michael	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p*.

2. For every record in each site, with probability *p*, it is selected as a seed independently.

3. Each site ships all its seeds to the coordinator.

4. The seeds set is sent back to every site.

5. Eve... r... si... f the records in the seed set. These rec... ed into the seeds set.

6. Repeat 4 and 5 until no ... ds is returned.

Seeds

James	$7322
Michael	$10000
Christina	$7633
Emily	$10822
Sean	$9607

The Uniform-*p* Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Michael	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p*.

2. For every record in each site, with probability *p*, it is selected as a seed independently.

3. Each site ships all its seeds to the coordinator.

4. The seeds set is sent back to every site.

5. Eve… the records in the seed set. These rec… ed into the seeds set.

6. Repeat 4 and 5 until no… ds is returned.

James	$7322
Emily	$10822
Michael	$10000
Christina	$7633
Sean	$9607

Seeds

The Uniform-p Algorithm

$R1$:New York	
Name	Salary
Michael	\$10000
Daniel	\$7864
David	\$8433

$R2$:Chicago	
Name	Salary
Christina	\$7633
Michael	\$8003
Sean	\$9607

$R3$:Vienna	
Name	Salary
Emily	\$10822
James	\$7322
Michael	\$9899
Christina	\$7412

1. Choose a probability p.

2. For every record in each site, with probability p, it is selected as a seed independently.

3. Each site ships all its seeds to the coordinator.

4. The seeds set is sent back to every site.

5. Eve[illegible] in the seed set. These records are added into the seeds set.

6. Repeat 4 and 5 until n[illegible]ds is returned.

Seeds

James	\$7322
Michael	\$8003
Emily	\$10822
Michael	\$10000
Christina	\$7633
Sean	\$9607
Christina	\$7412

The Uniform-*p* Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Michael	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p.*

2. For every record in each site, with probability *p*, it is selected as a seed independently.

3. Each site ships all its seeds to the coordinator.

4. The seeds set is sent back to every site.

5. Eve … si … in the seed set. These rec … ed into the seeds set.

James	$7322
Michael	$8003
Emily	$10822
Michael	$10000
Christina	$7633
Sean	$9607
Christina	$7412

Seeds

6. Repeat 4 and 5 until n … ds is returned.

The Uniform-*p* Algorithm

*R*1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

*R*2:Chicago	
Name	Salary
Christina	$7633
Michael	$8003
Sean	$9607

*R*3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

1. Choose a probability *p*.

2. For every record in each site, with probability *p*, it is selected as a seed independently.

3. Each site ships all its seeds to the coordinator.

4. The seeds set is sent back to every site.

5. Eve… in the seed set. These rec… ed into the seeds set.

James	$7322	Michael	$8003	Emily	$10822	Michael	$9899
Michael	$10000	Christina	$7633	Sean	$9607	Christina	$7412

Seeds

6. Repeat 4 and 5 until no… ds is returned.

More on Uniform-*p* Algorithm

*R*1:New York	
Name	Salary
Michael	\$10000
Daniel	\$7864
Michael	\$8433

*R*2:Chicago	
Name	Salary
Christina	\$7633
Michael	\$8003
Michael	\$9607

*R*3:Vienna	
Name	Salary
Michael	\$10822
Michael	\$7322
Michael	\$9899
Christina	\$7412

- Obvious Optimization:
  - Every time, for each site, only “new” records need to be sent.
- Drawbacks:
  - Large equivalence classes have high probability to sample out: $p_i=1-(1-p)^i$
  - If there is a super larger equivalence class, sampling becomes meaningless (too many records sampled out).

The Diminishing-*p* Algorithm

- Purpose:
  - Reduce the sampling probability of large equivalence classes.
  - Try to avoid sampling super large equivalence classes.
- Solution:
  - Take seeds as shown in the uniform-*p* algorithm.
  - Large equivalence classes need to wait for more time to complete.
    - In the first run, only partial equivalence classes of size one are sent back, in the second run, only partial equivalence classes of size two are sent back,…
  - In the beginning of every run, randomly killing some seeds.
  - If all seeds of an equivalence class are killed, the whole class is dropped the sample.
  - See the paper for details.

Can we do better:

Increasing the Accuracy While Retaining the Same Sample Size?

- More complicated methods is possible to build to improve the accuracy if the reconciliation function satisfies the size-ratio property:

$$\frac{\sum_{r \in S} Rec(\{r\})}{Rec(S)} = \rho(|S|)$$

- Example:
  - *Rec*(*S*): take the average salary.
  - *S*={(Michael, 10000), (Michael, 9899)}
  - $\rho(x)=x$
  - *Rec*(*S*) =(10000+9899)/2=9949.5
  - Rec({(Michael, 10000)})+Rec({(Michael, 9899)}
    =10000+9899=19899
  - 19899/9949.5=2=$\rho(|S|)$

Free Extra Information: the "Dirty" Sum

- Intuition:
  - The "dirty" sum is not the correct answer, but it still contains useful information.
- Basic trick: take the whole sum, try to estimate the error incurred due to the duplicates.
- Require Lagrangian multiplier optimization.
- See the paper for details.

R1:New York	
Name	Salary
Michael	$10000
Daniel	$7864
David	$8433

R2:Chicago	
Name	Salary
Christina	$7633
Steve	$8003
Sean	$9607

R3:Vienna	
Name	Salary
Emily	$10822
James	$7322
Michael	$9899
Christina	$7412

26297 25243 35455

Coordinator

Each site calculates the partial "dirty" sum while hashing / seeding and sends this value along with the records to the coordinator.

Experiment: Basic Setup

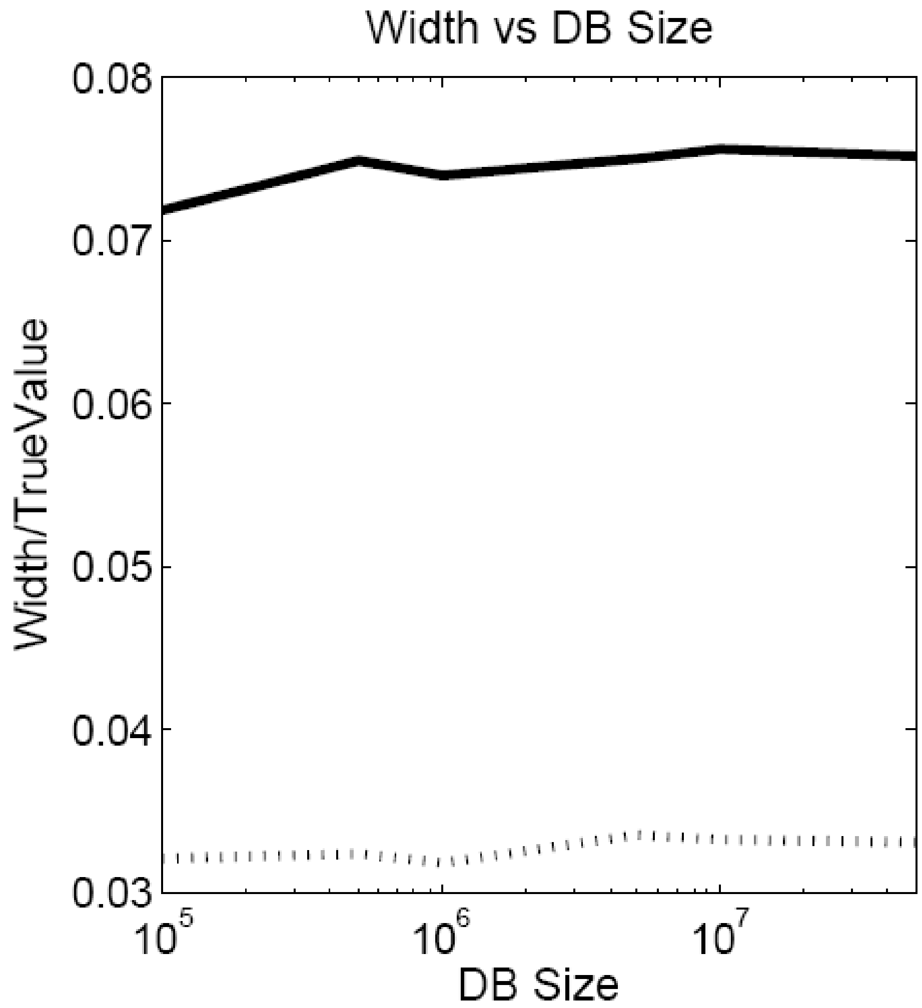
- Until the data set generated is of size *s*, the following steps are repeated:
  - An equivalence class size is generated by taking a random sample from a gamma distribution with shape parameter *sh* and scale parameter *sc*.
  - A equivalence class mean μ_1 is generated from a normal distribution with mean μ and variance σ^2.
  - For each record in this equivalence class, an aggregate value is generated using a sample from another normal distribution with mean μ_1 and variance σ^2.
  - Finally, each record in the equivalence class is randomly sent to one of five data sites.
- The data generation are determined by *s, sh, sc,* μ , σ^2.
- A sample fraction *p* is chosen to take samples.

Experiment: Estimators work correctly.

- We have 6 different estimators, 3 simple ones and three optimized ones in the test.
- We generated data using the following setting
  - $S=10^7$, $sh=1$, $sc=4$, $\mu=1$, $\sigma^2=1$
- For each estimator, we took 1% samples and provided 95% confidence bound, and checked whether the bound contained the answer.
- We repeated this procedure for 500 time and counted how many times each estimator provided a confidence bound that contained the answer.
- It turned every estimator worked well, the rate that the bound contained the query answer differed from 94% to 97%.

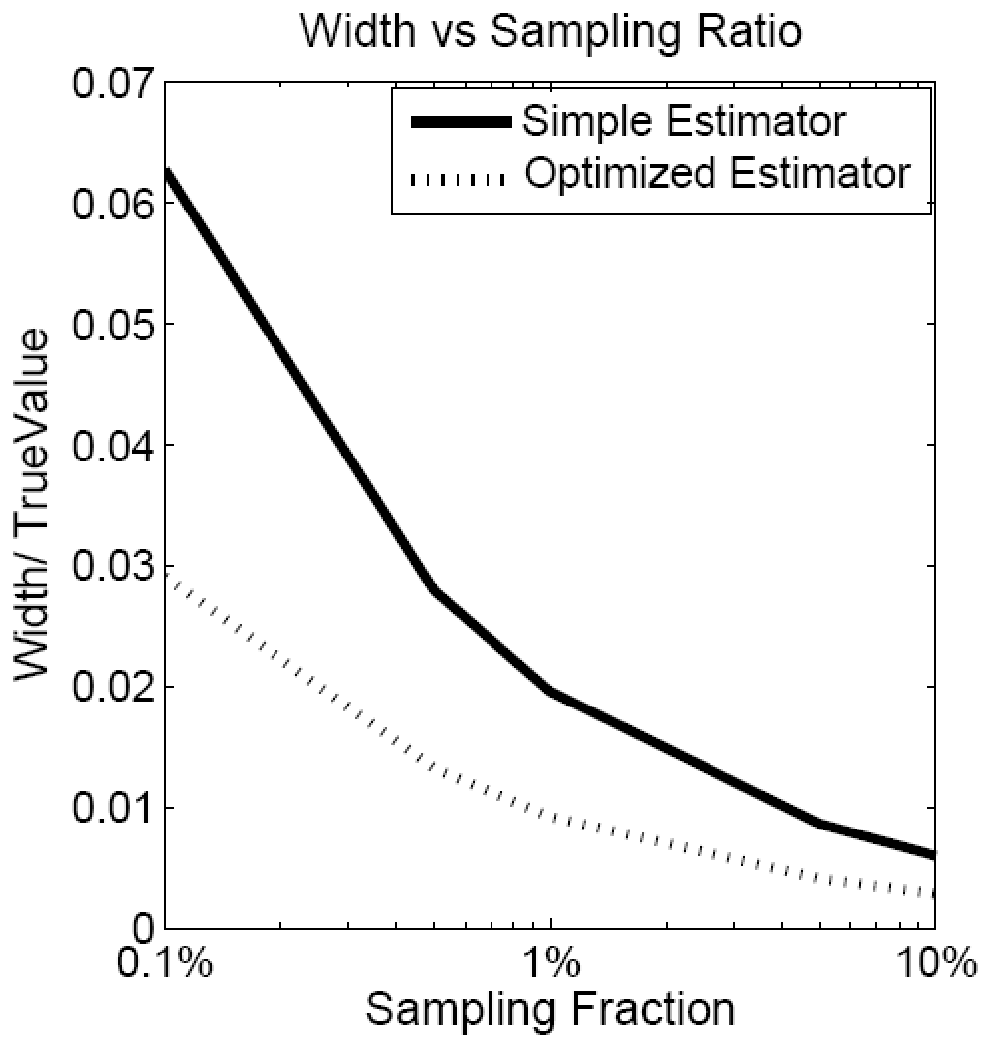
Experiment: Accuracy only Relies on Sample Size

- Fixed the sample size while increasing the database size and observed how the width of confidence bounds changed.
- Conclusion: accuracy only relies on the sample size.
- This is theoretically expected.



Experiment: Optimized Estimators Performed Better.

- For different sample ratio, execute both the simple estimator and the optimized one. Record both confidence interval widths.
- Conclusion: Optimized estimators performed better, especially when the sample ratio is low.



Experiment: The Cost of the Estimators.

- For three simple estimators, record the cost to reach a particular accuracy (in terms of confidence interval width) in two cases:
  - In the first case: 95% of the equivalence classes are sized fewer than 12, and 22% of the classes are size one.
  - In the second case: 95% of the classes have fewer than six elements and 44% of the classes are size one.
- Conclusion:
  - The cost of the Hash Bernoulli algorithm is always the smallest one.
  - The diminishing-*p* algorithm outperforms the Uniform-*p* algorithm when there are more large-size equivalence classes.

Conclusion

- We solve the problem to answer an aggregate query over a distributed, “dirty” database environment.
- We proposed several randomized algorithms.
- We integrate existing data cleaning techniques into our solution:
  - Two generic function are used to encapsulate existing data cleaning techniques.

Thank You.

Questions?