Modeling and Querying Possible Repairs in Duplicate Detection

Based on

„Modeling and Quering Possible Repairs in Duplicate Detection" PVLDB(2)1: 598-609 (2009)

by

George Beskales, Mohamed A. Soliman

Ihab F. Ilyas, Shai Ben-David

Data Cleaning

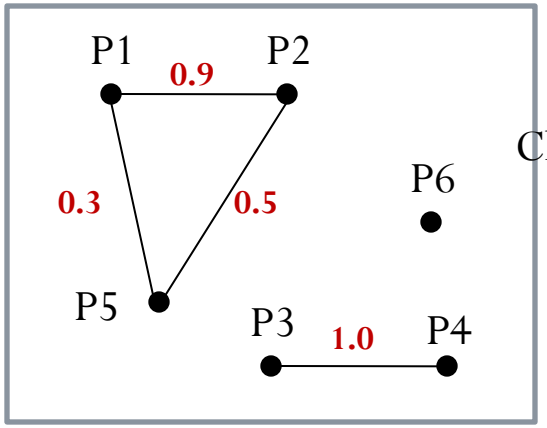
- Real-world data is dirty
- Examples:
  - Syntactical Errors: e.g., Micrsoft
  - Heterogeneous Formats: e.g., Phone number formats
  - Missing Values (Incomplete data)
  - Violation of Integrity Constraints: e.g., FDs/INDs
  - Duplicate Records
- Data Cleaning is the process of improving data quality by removing errors, inconsistencies and anomalies of data

Duplicate Elimination Process

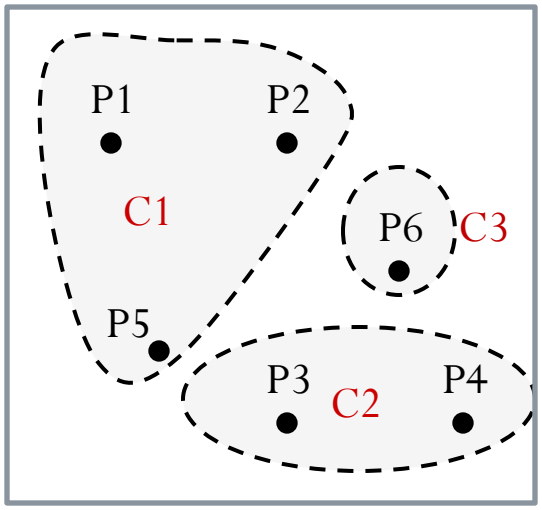
Unclean Relation

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 Pairwise Similarity



Cluster Records

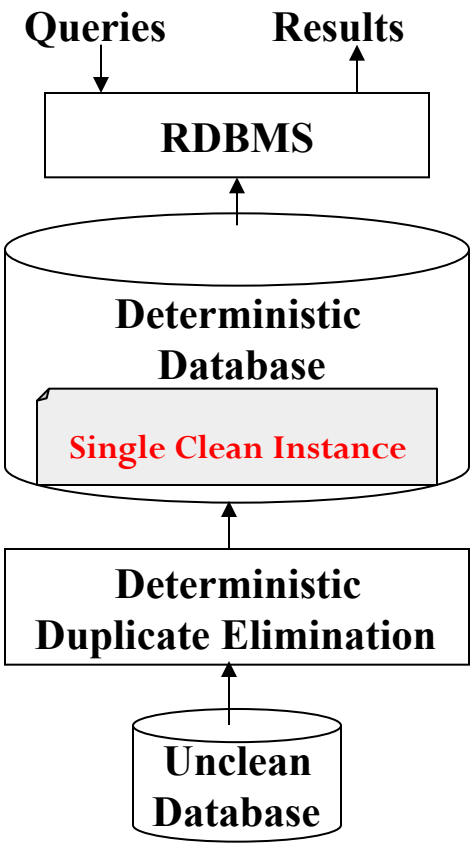


Merge Clusters

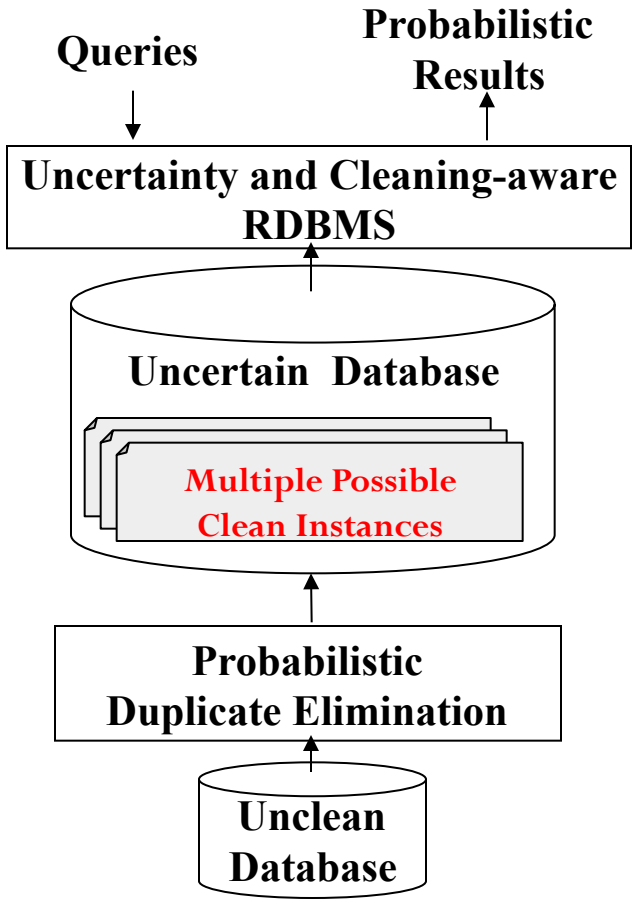
Clean Relation

ID	name	ZIP	Income
C1	Green	51519	39k
C2	Peter	30528	40k
C3	Chuck	51519	30k

Probabilistic Data Cleaning

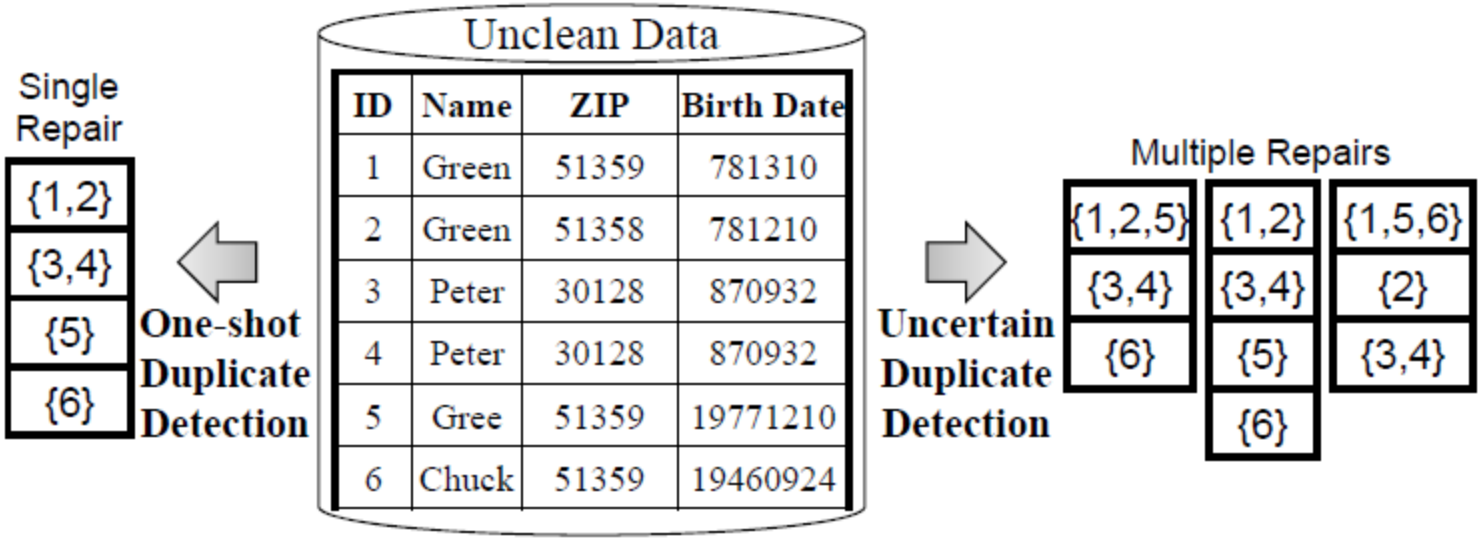


(a) One-Shot Cleaning



(b) Probabilistic Cleaning

Probabilistic Data Cleaning



One-shot Cleaning and Probabilistic Cleaning compared

Motivation

Probabilistic Data Cleaning:

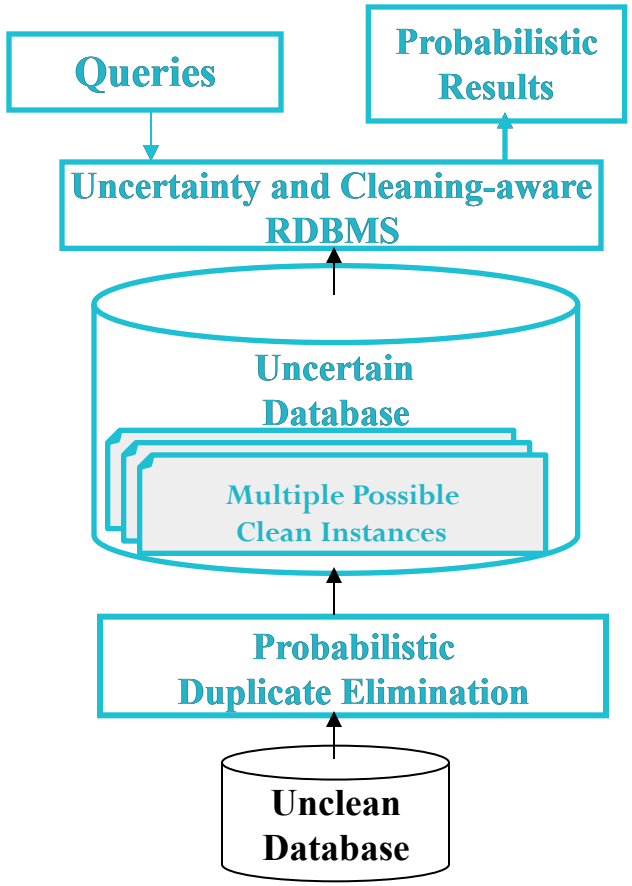
1. Avoid deterministic resolution of conflicts during data cleaning

2. Enrich query results by considering all possible cleaning instances

3. Allows specifying query-time cleaning requirements

Outline

- Generating and Storing the Possible Clean Instances
- Querying the Clean Instances
- Experimental Evaluation

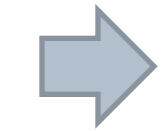


Uncertainty in Duplicate Detection

- **Uncertain Duplicates**: Determining what records should be clustered is uncertain due to noisy similarity measurements
- A **possible repair** of a relation is a clustering (partitioning) of the unclean relation

Person

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



Uncertain Clustering

Possible Repairs

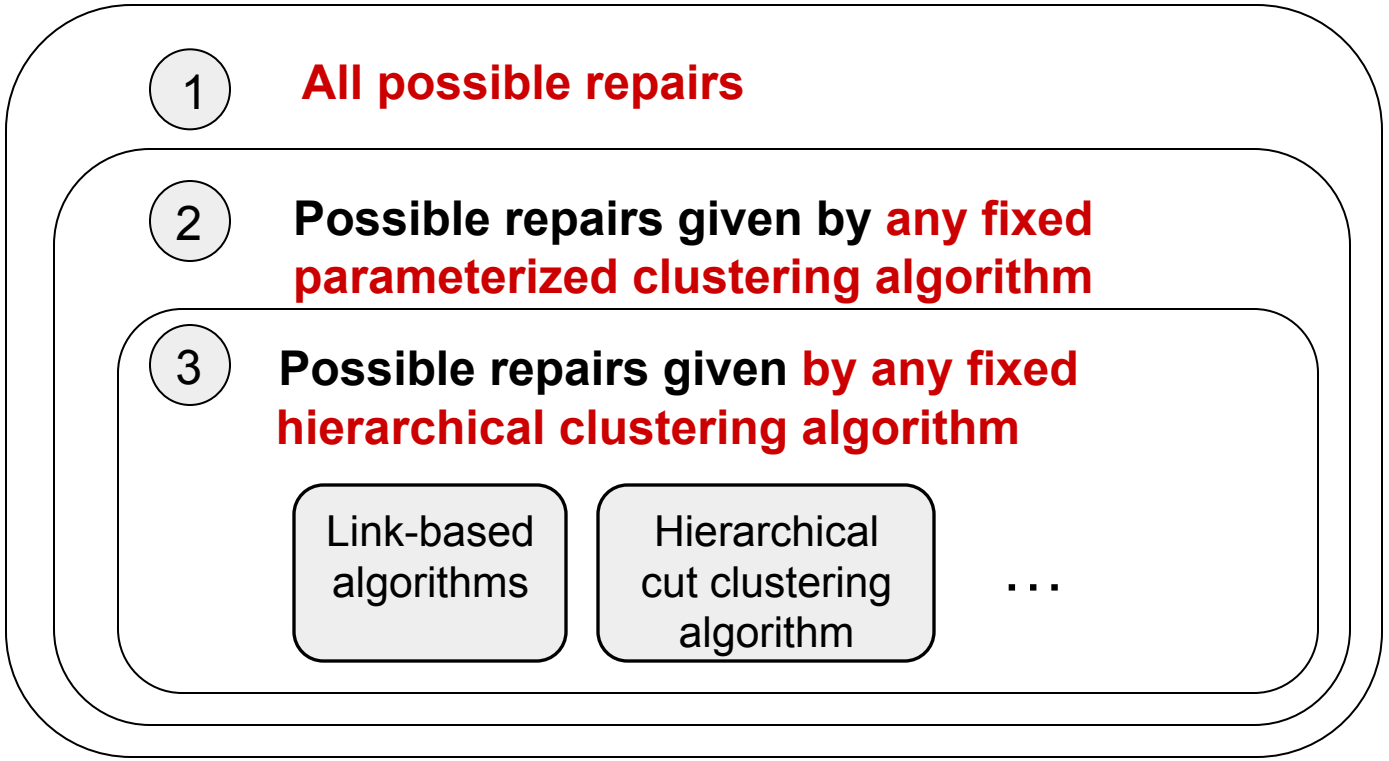
X_1	X_2	X_3
{P1}	{P1,P2}	{P1,P2,P5}
{P2}	{P3,P4}	{P3,P4}
{P3,P4}	{P5}	{P6}
{P5}	{P6}	
{P6}		

Challenges

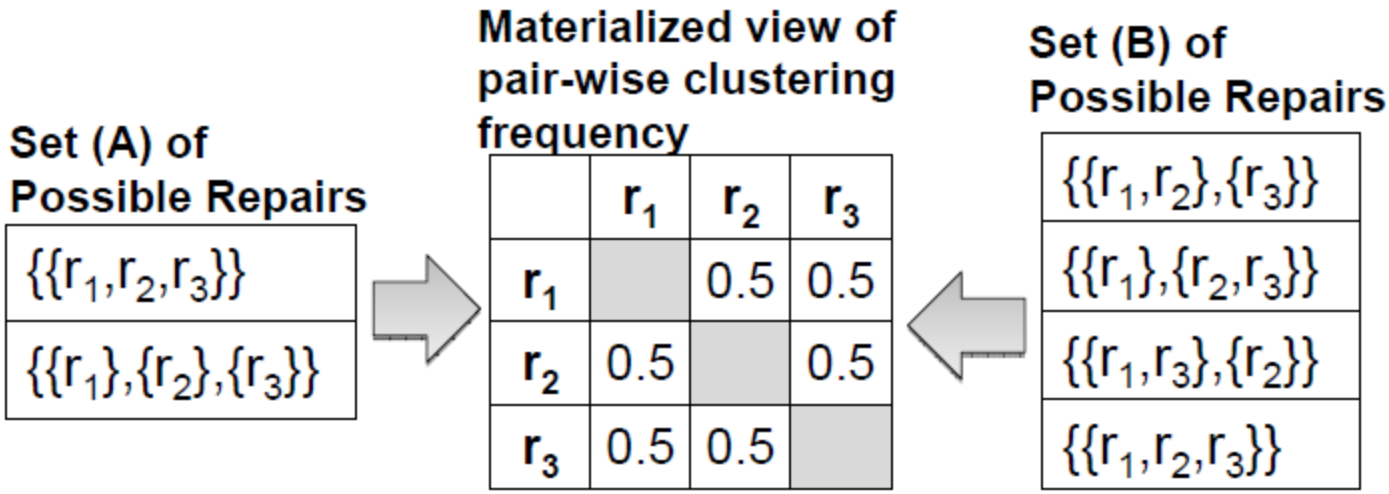
1. The space of all possible clusterings (repairs) is exponentially large

2. How to efficiently and reasonably generate, store and query the possible repairs?

The Space of Possible Repairs



Cleaning Algorithm



- The model should store possible repairs in lossless way

Cleaning Algorithm

- Allow efficient answering of important queries (e.g. queries frequently encountered in applications)

- Provide materializations of the results of costly operations (e.g. clustering procedures) that are required by most queries

- Small space complexity to allow efficient construction, storage and retrieval of the possible repairs, in addition to efficient query processing

Algorithm – Dependent Model

A – Algorithm; R – starting unclean relation; P – set of possible parameters for A

τ - continuous random variable for A from interval $[\tau^l, \tau^u]$
f_τ – probability density function of τ (given by user or learned)

Applying A to R using parameter $t \in [\tau^l, \tau^u]$ generates possible clustering (i.e. repair) denoted as A(R, t)

$\mathcal{X}$ – set of all possible repairs, defined as $\{A(R,t) : t \in [\tau^l, \tau^u]\}$

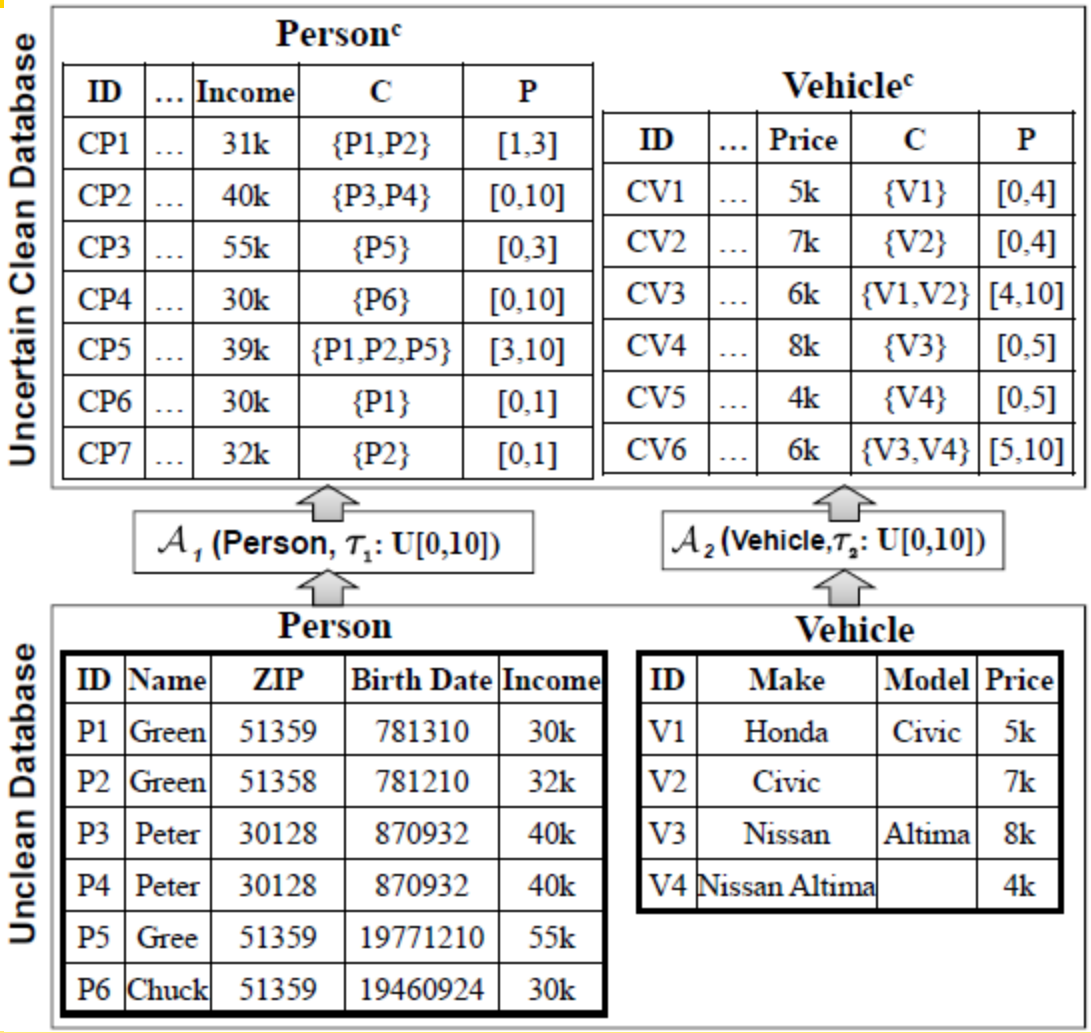
Algorithm – Dependent Model

Probability of a specific repair X $\in \mathcal{X}$:

$$\Pr(X) = \int_{\tau^l}^{\tau^u} f_\tau(t) \cdot h(t, X)\, dt$$

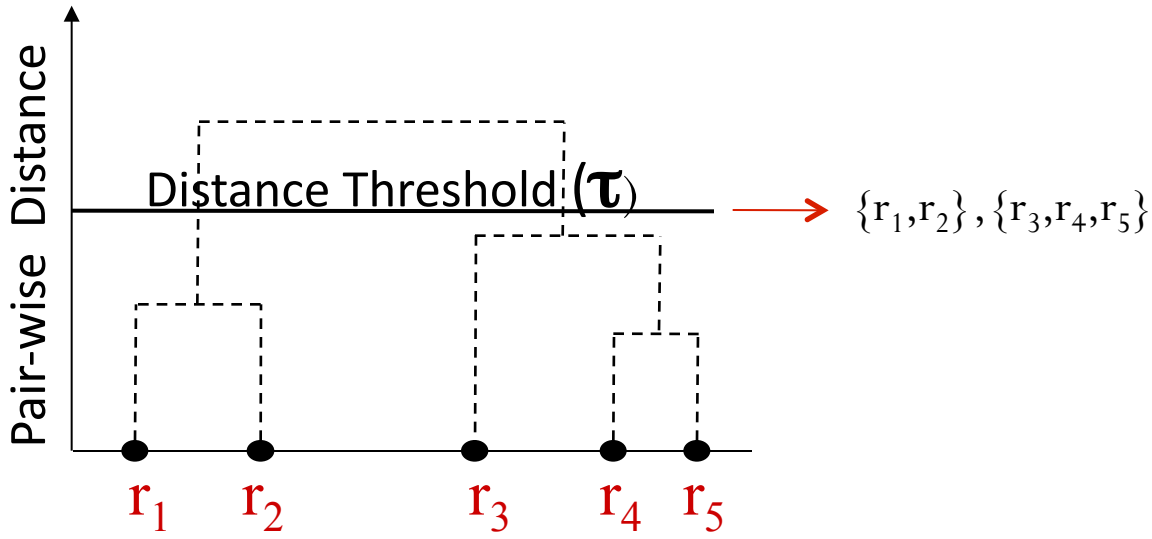
Where h(t, X) = 1 if A(R,t) = X, and 0 otherwise.

Creating U-clean Relations



Hierarchical Clustering Algorithms

- Records are clustered in a form of a hierarchy: all singletons are at leaves, and one cluster is at root
- **Example**: Linkage-based Clustering Algorithm

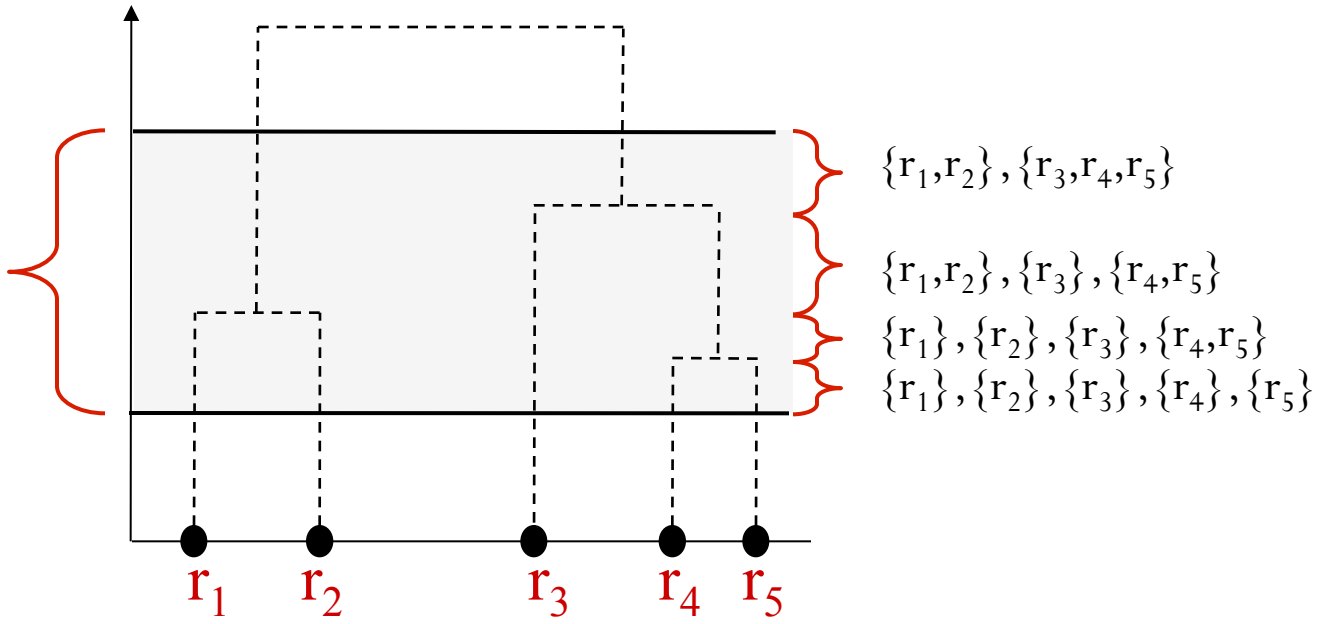


Uncertain Hierarchical Clustering

- Hierarchical clustering algorithms can be modified to accept uncertain parameters
- The number of generated possible repairs is linear

Possible Thresholds

$[\tau_l, \tau_u]$



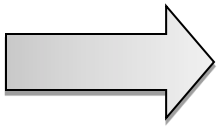
Uncertain Hierarchical Clustering

Algorithm 1 `U_Cluster` $(R(A_1, \ldots, A_m), \tau^l, \tau^u)$

Require: $R(A_1, \ldots, A_m)$: The unclean relation
Require: τ^l: Minimum threshold value
Require: τ^u: Maximum threshold value
1: Define a new singleton cluster C_i for each record $r_i \in R$ (i.e., C_i contains a unique identifier of r_i)
2: $\mathcal{C} \leftarrow \{C_1, \ldots, C_{|R|}\}$
3: **for each** $r_i \in R$ **do**
4: Add $(r_i[A_1], \ldots, r_i[A_m], C_i, [\tau^l, \tau^u])$ to R^c
5: **end for**
6: **while** ($|\mathcal{C}| > 1$ and distance between the closest pair of clusters (C_i, C_j) in $\mathcal{C}$ is less than τ^u) **do**
7: $C_k \leftarrow C_i \cup C_j$
8: Replace C_i and C_j in $\mathcal{C}$ with C_k
9: $r_k \leftarrow$ `get_representative_record`(C_k) {See Section 3.3}
10: Add $(r_k[A_1], \ldots, r_k[A_m], C_k, [dist(C_i, C_j), \tau^u])$ to R^c
11: **if** $(dist(C_i, C_j) < \tau^l)$ **then**
12: Remove the c-records corresponding to C_i and C_j from R^c
13: **else**
14: Set the upper bounds of parameter settings of the c-records corresponding to C_i and C_j to $dist(C_i, C_j)$
15: **end if**
16: **end while**
17: **return** R^c

Probabilities of Repairs

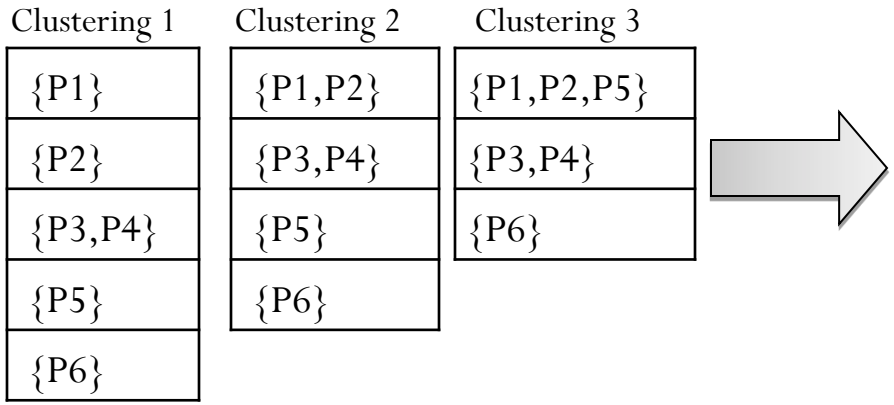
ID	Name	ZIP	Income
P1	Green	51519	30k
P2	Green	51518	32k
P3	Peter	30528	40k
P4	Peter	30528	40k
P5	Gree	51519	55k
P6	Chuck	51519	30k



$\tau \sim U(0,10)$

Clustering 1	Clustering 2	Clustering 3
{P1}	{P1,P2}	{P1,P2,P5}
{P2}	{P3,P4}	{P3,P4}
{P3,P4}	{P5}	{P6}
{P5}	{P6}	
{P6}		
$0 \leq \tau < 1$	$1 \leq \tau < 3$	$3 \leq \tau < 10$
Pr = 0.1	Pr = 0.2	Pr = 0.7

Representing the Possible Repairs



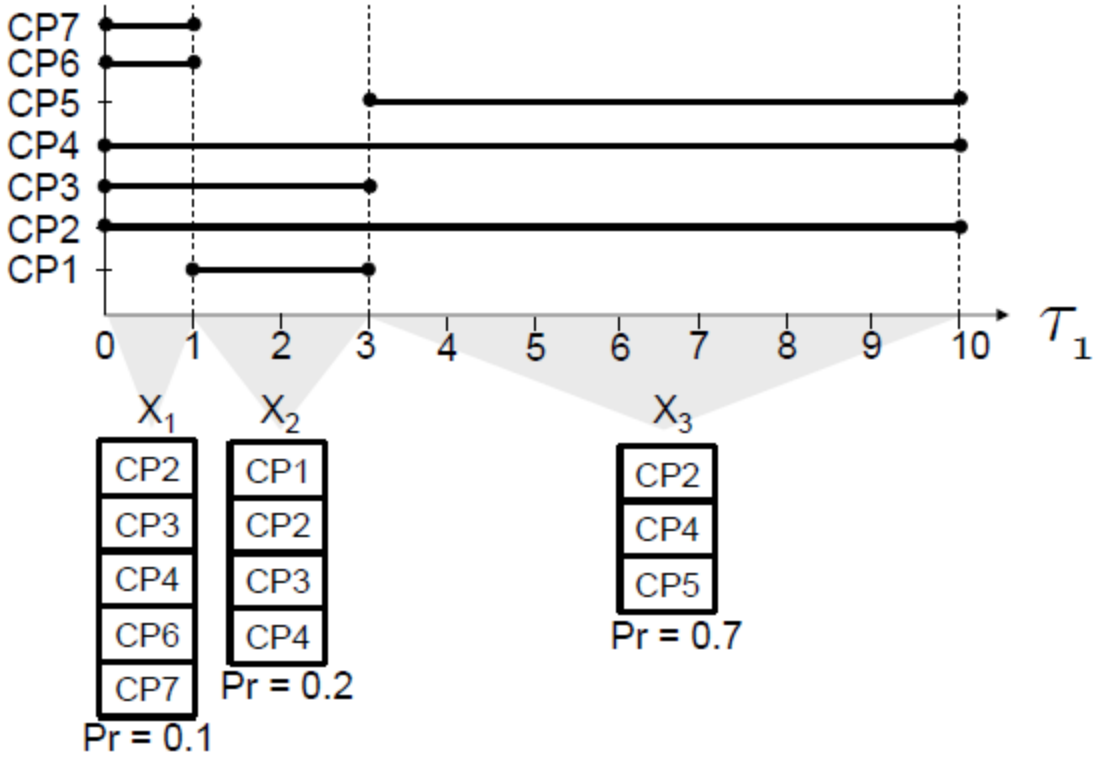
$0 \leq \tau < 1$ $\quad$ $1 \leq \tau < 3$ $\quad$ $3 \leq \tau < 10$

$Pr = 0.1$ $\quad$ $Pr = 0.2$ $\quad$ $Pr = 0.7$

ID	…	Income	C	P
CP1	…	31k	{P1,P2}	[1,3)
CP2	…	40k	{P3,P4}	[0,10)
CP3	…	55k	{P5}	[0,3)
CP4	…	30k	{P6}	[0,10)
CP5	…	39k	{P1,P2,P5}	[3,10)
CP6	…	30k	{P1}	[0,1)
CP7	…	32k	{P2}	[0,1)

U-clean Relation $Person^C$

Constructing Probabilistic Repairs



Re-creating Probabilistic repairs from U-clean Relations

NN-based clustering

- Tuples represented as points in d-dimensional space
- Columns are dimensions
- Dimensions ordering is very important choice
- Clusters are created by coalescing points that are “near” to each other
- With growing t points that are further and further away are being put together into mutual clusters, also various clusters can be united
- Algorithm stops when there is only one cluster and all points belong to it or maximum value of t is reached.

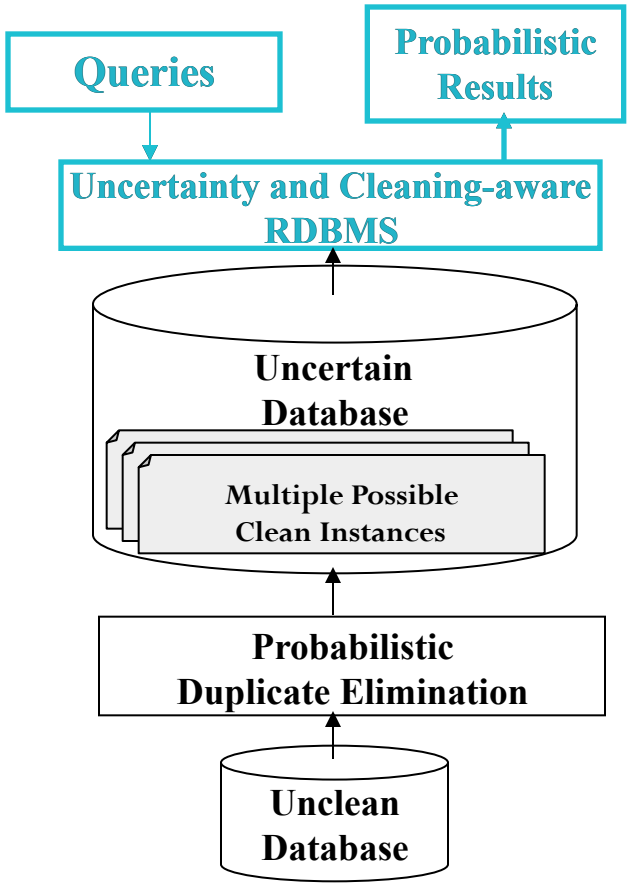
Time and Space Complexity

- Hierarchical clustering arranges records in N-ary tree, records being the leaves
- Maximum possible number of nodes of a tree that has n leaves (assuming each non-leaf node has at least 2 children) is 2n-1
- Therefore number of possible clusters is bounded by 2n-1, so it is linear w.r.t. to number of starting tuples.

- In general above algorithms have asymptotic complexity exactly as the original ones on which they build, since only constant amount of work is added to each iteration

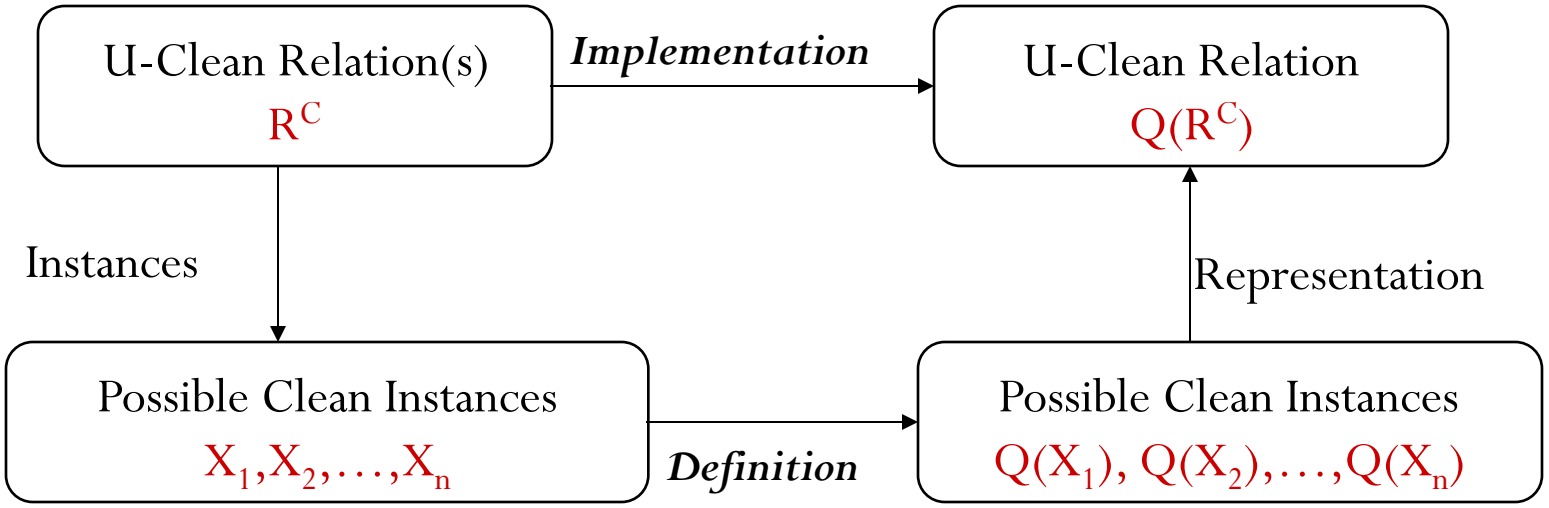
Outline

- Generating and Storing the Possible Clean Instances
- Querying the Clean Instances
- Experimental Evaluation



Queries over U-Clean Relations

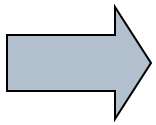
- We adopt the *possible worlds semantics* to define queries on U-clean relations



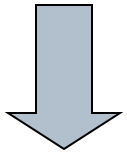
Example: Selection Query

Person^{C}

ID	…	Income	C	P
CP1	…	31k	{P1,P2}	[1,3)
CP2	…	40k	{P3,P4}	[0,10)
CP3	…	55k	{P5}	[0,3)
CP4	…	30k	{P6}	[0,10)
CP5	…	39k	{P1,P2,P5}	[3,10)
CP6	…	30k	{P1}	[0,1)
CP7	…	32k	{P2}	[0,1)



SELECT ID, Income
FROM Person^{c}
WHERE Income>35k



ID	Income	C	P
CP2	40k	{P3,P4}	[0,10)
CP3	55k	{P5}	[0,3)
CP5	39k	{P1,P2,P5}	[3,10)

Example: Projection Query

Person^C

ID	…	Income	C	P
CP1	…	31k	{P1,P2}	[1,3)
CP2	…	40k	{P3,P4}	[0,1)
CP3	…	55k	{P5}	[0,3)
CP4	…	30k	{P6}	[3,10)
CP5	…	40k	{P1,P2,P5}	[3,10)
CP6	…	30k	{P1}	[0,1)
CP7	…	32k	{P2}	[0,1)

SELECT DISTINCT Income
FROM Person^c

Income	C	P
30k	{P1} v {P6}	[0,1) v [3,10)
31k	{P1,P2}	[1,3)
32k	{P2}	[0,1)
40k	{P3,P4} v {P1,P2,P5}	[0,1) v [3,10)
55k	{P5}	[0,3)

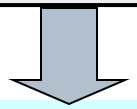
Example: Join Query

Person^c

ID	…	Income	C	P
CP1	…	31k	{P1,P2}	τ_1 :[1,3)
CP2	…	40k	{P3,P4}	τ_1:[0,10)
…	…	…	…	…

Vehicle^c

ID	Price	C	P
CV5	4k	{V4}	τ_2 :[3,5)
CV6	6k	{V3,V4}	τ_2 :[5,10)
…	….	….	….



SELECT Income, Price
FROM Person^c , Vehicle^c
WHERE Income/10 >= Price

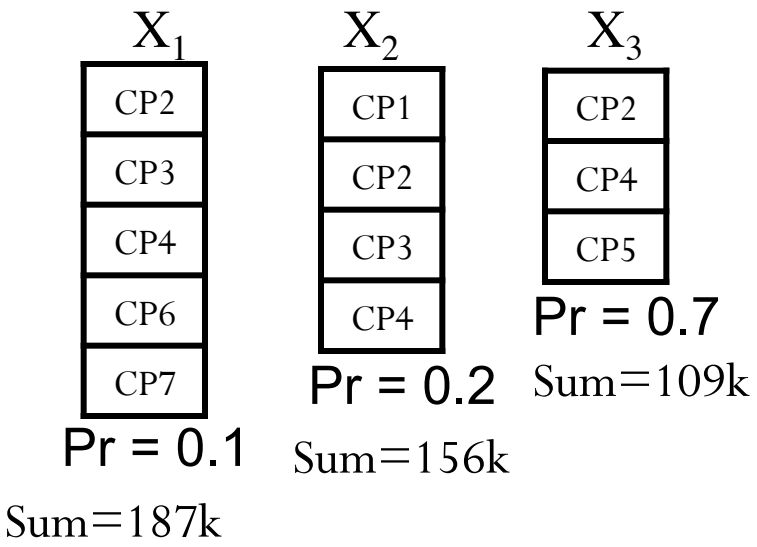
Income	Price	C	P
40k	4k	{P3,P4} ^ {V4}	τ_1 :[0, 10) ^ τ_2 :[3,5)
…	…	…	…

Aggregation Queries

Person^c

ID	…	Income	C	P
CP1	…	31k	{P1,P2}	[1,3)
CP2	…	40k	{P3,P4}	[0,10)
CP3	…	55k	{P5}	[0,3)
CP4	…	30k	{P6}	[0,10)
CP5	…	39k	{P1,P2,P5}	[3,10)
CP6	…	30k	{P1}	[0,1)
CP7	…	32k	{P2}	[0,1)

SELECT Sum(Income)
FROM Person^c



Other Meta-Queries

1. Obtaining the most probable clean instance
2. Obtaining the α-certain clusters
3. Obtaining a clean instance corresponding to a specific parameters of the clustering algorithms
4. Obtaining the probability of clustering a set of records together

Outline

- Generating and Storing the Possible Clean Instances
- Querying the Clean Instances
- Experimental Evaluation

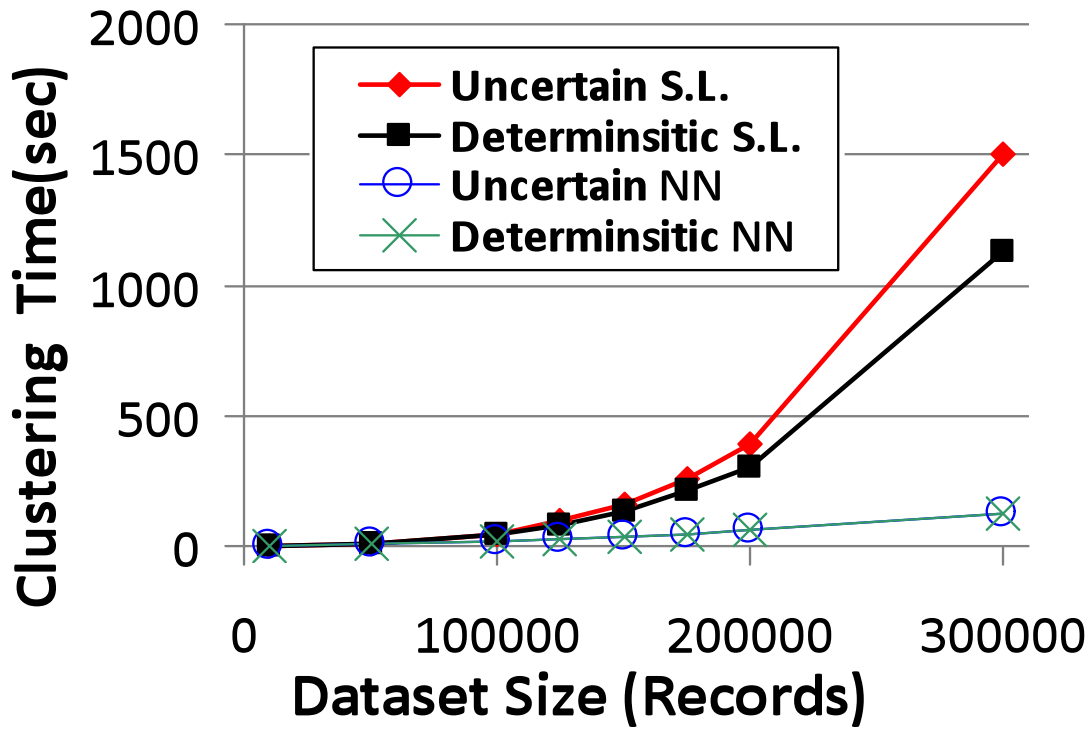
Experimental Evaluation

- A prototype as an extension of PostgreSQL
- Synthetic data generator provided by Febrl (freely extensible biomedical record linkage)
- Two hierarchical algorithms:
  - Single-Linkage (S.L.)
  - Nearest-neighbor based clustering algorithm (N.N.) *[Chaudhuri et al., ICDE'05]*

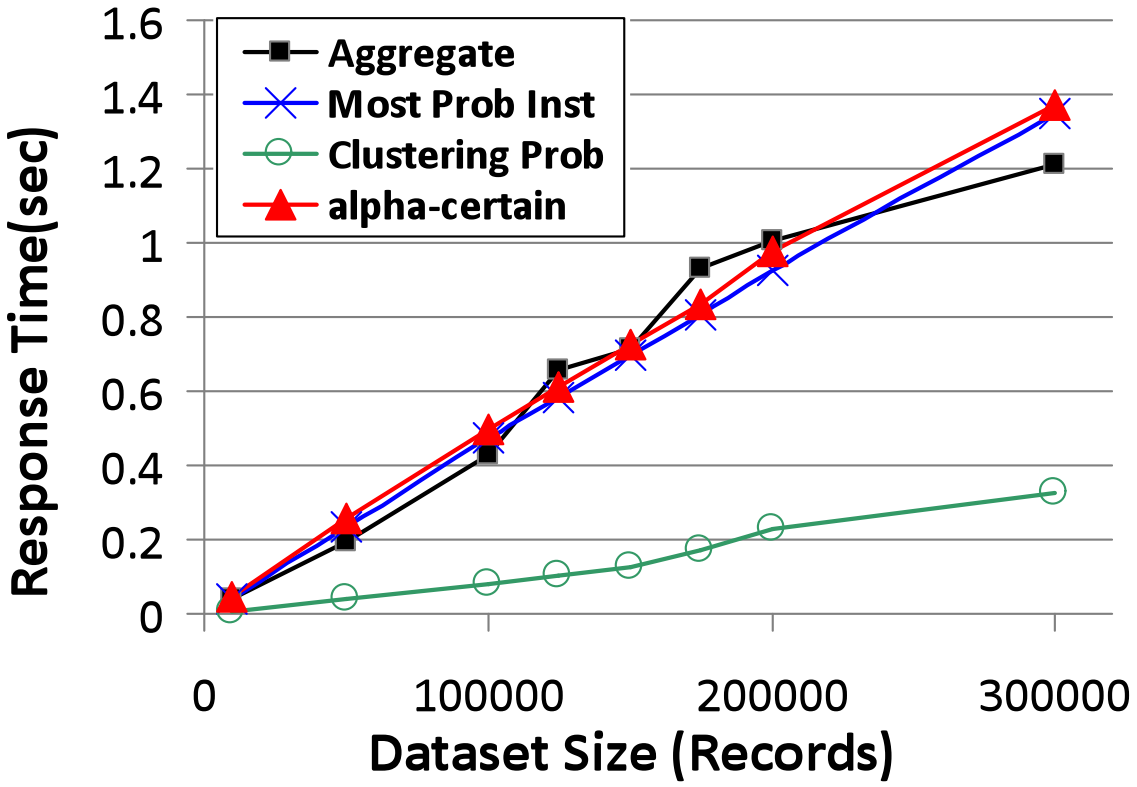
Experimental Evaluation

- Machine used: SunFire X4100, Dual Core 2.2GHz, 8GB RAM
- 10% of records were duplicates
- Each query executed 5 times and average time taken

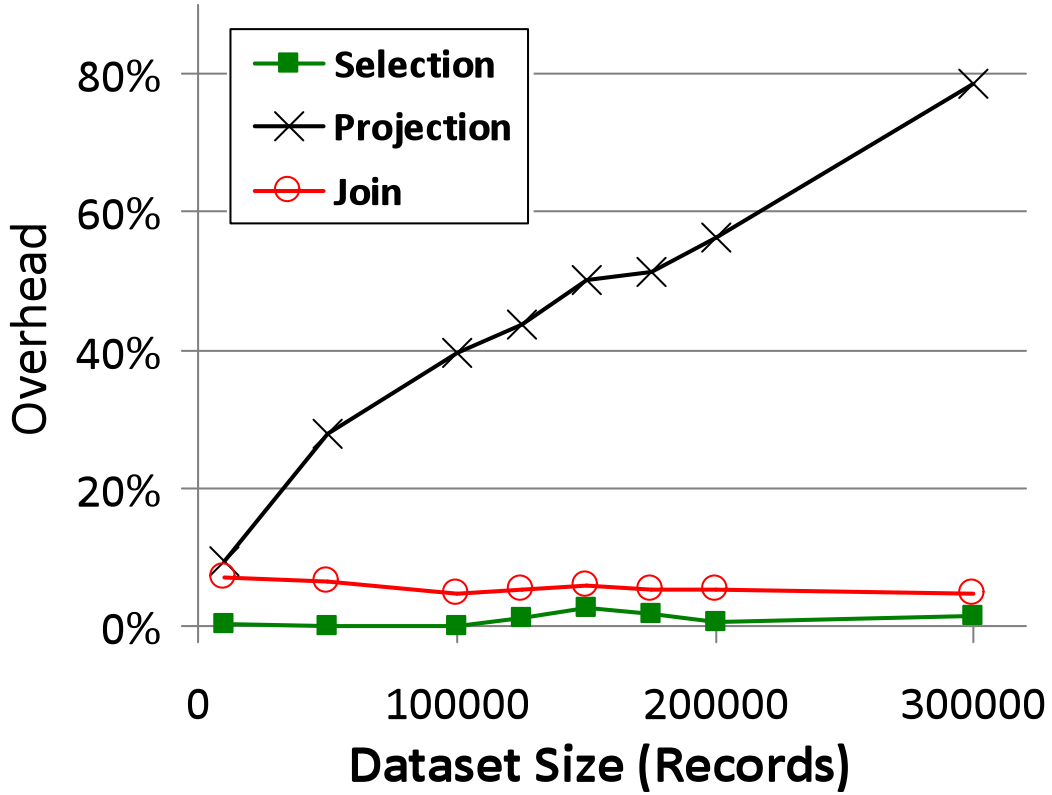
Experimental Evaluation



Experimental Evaluation



Experimental Evaluation



Experimental Evaluation

- Exucution time overhead for S.L. Is 30%, and for NN less than 5% , and it is negligibly correlated with percentage of duplicates
- Space overhead is 8.35%
- Extracting clean instance for 100,000 records requeries only 1.5sec, which means this approach is more efficient than restarting deduplication algorithm whenever a new parameter setting is requested

Conclusion

- We allow representing and querying multiple possible clean instances
- We modified hierarchical clustering algorithms to allow generating multiple repairs
- We compactly store the possible repairs by keeping the lineage information of clusters in special attributes
- New (probabilistic) query types can be issued against the population of possible repairs