# Get a Sample for a Discount

## Sampling-Based XML Data Pricing

Ruiming Tang, Antoine Amarilli,
Pierre Senellart, Stéphane Bressan

National University of Singapore

# Introduction

- In most data pricing literature (e.g., [Koutris et al., 2012a, Koutris et al., 2012b, Koutris et al., 2013, Li and Miklau, 2012]), data prices are prescribed and not negotiable, and give access to the best data quality that the provider can achieve.

# Introduction

- In most data pricing literature (e.g., [Koutris et al., 2012a, Koutris et al., 2012b, Koutris et al., 2013, Li and Miklau, 2012]), data prices are prescribed and not negotiable, and give access to the best data quality that the provider can achieve.
- We propose a pricing framework, in which data quality can be traded for discounted prices. We allow a data consumer to propose her own price for the requested data.

# Introduction

- In most data pricing literature (e.g., [Koutris et al., 2012a, Koutris et al., 2012b, Koutris et al., 2013, Li and Miklau, 2012]), data prices are prescribed and not negotiable, and give access to the best data quality that the provider can achieve.
- We propose a pricing framework, in which data quality can be traded for discounted prices. We allow a data consumer to propose her own price for the requested data.
  - If the proposed price is the full price of the requested data, the requested data is returned to the data consumer.
  - If the proposed price is less than the full price, a lower quality version of the requested data is returned.

# Introduction

- In most data pricing literature (e.g., [Koutris et al., 2012a, Koutris et al., 2012b, Koutris et al., 2013, Li and Miklau, 2012]), data prices are prescribed and not negotiable, and give access to the best data quality that the provider can achieve.
- We propose a pricing framework, in which data quality can be traded for discounted prices. We allow a data consumer to propose her own price for the requested data.
  - If the proposed price is the full price of the requested data, the requested data is returned to the data consumer.
  - If the proposed price is less than the full price, a lower quality version of the requested data is returned.
- What are the dimensions to access data quality?

# Introduction

- Data quality dimensions (defined in [Pipino et al., 2002, Wang and Strong, 1996]):

# Introduction

- Data quality dimensions (defined in [Pipino et al., 2002, Wang and Strong, 1996]):
  - intrinsic quality (believability, objectivity, accuracy, reputation)
  - contextual quality (value-added, relevancy, timeliness, ease of operation, appropriate amount of data, completeness)
  - representational quality (interpretability, ease of understanding, concise representation, consistent representation)
  - accessibility quality (accessibility, security)

# Introduction

- Data quality dimensions (defined in [Pipino et al., 2002, Wang and Strong, 1996]):
  - intrinsic quality (believability, objectivity, accuracy, reputation)
  - contextual quality (value-added, relevancy, timeliness, ease of operation, appropriate amount of data, completeness)
  - representational quality (interpretability, ease of understanding, concise representation, consistent representation)
  - accessibility quality (accessibility, security)
- In our previous work [Tang et al., 2013], we proposed a pricing framework for **relational** data, in which **accuracy** can be traded for discounted prices.

# Introduction

- Data quality dimensions (defined in [Pipino et al., 2002, Wang and Strong, 1996]):
  - intrinsic quality (believability, objectivity, accuracy, reputation)
  - contextual quality (value-added, relevancy, timeliness, ease of operation, appropriate amount of data, completeness)
  - representational quality (interpretability, ease of understanding, concise representation, consistent representation)
  - accessibility quality (accessibility, security)
- In our previous work [Tang et al., 2013], we proposed a pricing framework for **relational** data, in which **accuracy** can be traded for discounted prices.
- In this paper, we propose a pricing framework for **XML** data, in which **completeness** can be traded for discounted prices.

NUS National University of Singapore

# Our Framework

- Three main actors in data markets:

# Our Framework

- Three main actors in data markets:
  - Data provider: she has an XML document and sets a price to this document. She also assigns a *weight* to each node.

# Our Framework

- Three main actors in data markets:
  - Data provider: she has an XML document and sets a price to this document. She also assigns a *weight* to each node.
  - Data consumer: she proposes a price for the document. The proposed price may be lower than the price of the document because
    - she has a limited budget
    - she wants to explore the document before the full purchase

# Our Framework

- Three main actors in data markets:
  - Data provider: she has an XML document and sets a price to this document. She also assigns a *weight* to each node.
  - Data consumer: she proposes a price for the document. The proposed price may be lower than the price of the document because
    - she has a limited budget
    - she wants to explore the document before the full purchase
  - Data market owner:
    - she negotiates with the data provider a *pricing function* to decide the *completeness* of a sample that should be returned, according to a proposed price
    - she samples a rooted subtree of the requested document with the decided completeness **uniformly at random**.

# Weight and Completeness of a rooted subtree

- A **rooted subtree** $t'$ of a tree $t$ is (1) a subtree of $t$ and (2) $\text{root}(t') = \text{root}(t)$.

# Weight and Completeness of a rooted subtree

- A **rooted subtree** $t'$ of a tree $t$ is (1) a subtree of $t$ and (2) $\text{root}(t') = \text{root}(t)$.
- The **weight of a tree** ($\text{weight}(t)$) is intuitively, the sum weight of all the nodes.

# Weight and Completeness of a rooted subtree

- A **rooted subtree** $t'$ of a tree $t$ is (1) a subtree of $t$ and (2) $\text{root}(t') = \text{root}(t)$.
- The **weight of a tree** ($\text{weight}(t)$) is intuitively, the sum weight of all the nodes.
- **Completeness** of $t'$ with respect to a tree $t$ ($t'$ is a rooted subtree of $t$) is $c_t(t') = \frac{\text{weight}(t')}{\text{weight}(t)}$.

NUS National University of Singapore

# Pricing Function

- The **pricing function** of a tree $t$ is a function $\phi_t : [0, 1] \rightarrow \mathbb{Q}^+$. Its input is the completeness of a rooted subtree $t'$ and it returns the price of $t'$, as a non-negative rational.

# Pricing Function

- The **pricing function** of a tree $t$ is a function $\phi_t : [0, 1] \to \mathbb{Q}^+$. Its input is the completeness of a rooted subtree $t'$ and it returns the price of $t'$, as a non-negative rational.
- Non-decreasing. The more complete a rooted subtree is, the more expensive it should be, i.e., $c_1 \geq c_2 \Rightarrow \phi_t(c_1) \geq \phi_t(c_2)$.

NUS National University of Singapore

# Pricing Function

- The **pricing function** of a tree $t$ is a function $\phi_t : [0, 1] \to \mathbb{Q}^+$. Its input is the completeness of a rooted subtree $t'$ and it returns the price of $t'$, as a non-negative rational.
- Non-decreasing. The more complete a rooted subtree is, the more expensive it should be, i.e., $c_1 \geq c_2 \Rightarrow \phi_t(c_1) \geq \phi_t(c_2)$.
- Arbitrage-free. Buying a rooted subtree of completeness $c_1 + c_2$ should not be more expensive than buying two subtrees with respective completeness $c_1$ and $c_2$, i.e., $\phi_t(c_1) + \phi_t(c_2) \geq \phi_t(c_1 + c_2)$. This property is useful when considering repeated requests.

# Pricing Function

- Minimum and maximum bound. We should have $\phi_t(0) = pr_{\min}$ and $\phi_t(1) = pr_t$, where $pr_{\min}$ is the minimum cost that a data consumer has to pay using the data market and $pr_t$ is the price of the whole tree $t$.

# Pricing Function

- Minimum and maximum bound. We should have $\phi_t(0) = pr_{\min}$ and $\phi_t(1) = pr_t$, where $pr_{\min}$ is the minimum cost that a data consumer has to pay using the data market and $pr_t$ is the price of the whole tree $t$.
- All these properties can be satisfied, for instance, by functions of the form $\phi_t(c) = (pr_t - pr_{\min})c^p + pr_{\min}$ where $p \leq 1$.

# Sampling Problem

- Given a proposed price $pr_0$, once a completeness value $c \in \phi_t^{-1}(pr_0)$ is chosen, the weight of the returned rooted subtree is fixed as $c \times \text{weight}(t)$.

# Sampling Problem

- Given a proposed price $pr_0$, once a completeness value $c \in \phi_t^{-1}(pr_0)$ is chosen, the weight of the returned rooted subtree is fixed as $c \times \text{weight}(t)$.
- We consider the problem of uniform sampling a rooted subtree with prescribed weight (instead with prescribed completeness). The problem of **sampling a rooted subtree**, given a tree $t$ and a weight $k$, is to sample a rooted subtree $t'$ of $t$, such that $\text{weight}(t') = k$, *uniformly at random*.

# Sampling Problem

- Given a proposed price $pr_0$, once a completeness value $c \in \phi_t^{-1}(pr_0)$ is chosen, the weight of the returned rooted subtree is fixed as $c \times \text{weight}(t)$.
- We consider the problem of uniform sampling a rooted subtree with prescribed weight (instead with prescribed completeness). The problem of **sampling a rooted subtree**, given a tree $t$ and a weight $k$, is to sample a rooted subtree $t'$ of $t$, such that $\text{weight}(t') = k$, *uniformly at random*.
- Why "uniformly at random"? To be fair to the data consumer, there should be an equal chance to explore every possible part of the XML document that is worth the proposed price.

# Tractability of the Sampling Problem

- Given a tree $t$ and a weight $x$, it is NP-hard to sample a rooted subtree of $t$ of weight $x$ uniformly at random.

# Tractability of the Sampling Problem

- Given a tree $t$ and a weight $x$, it is NP-hard to sample a rooted subtree of $t$ of weight $x$ uniformly at random.
- Tractable cases:
  - Unweighted Sampling: $\mathrm{w}(n) = 1$ for all $n$.
  - $0/1$-weights Sampling.: $\mathrm{w}(n) \in \{0, 1\}$ for all $n$.

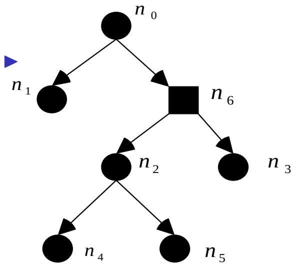
# Unweighted Sampling for Binary Trees

- We first present a sampling algorithm for binary trees and extend the algorithm to any unranked trees.

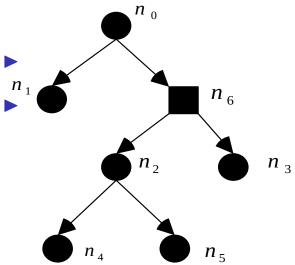
# Unweighted Sampling for Binary Trees

- We first present a sampling algorithm for binary trees and extend the algorithm to any unranked trees.
- **First phase: Subtree Counting.** We start by computing a matrix $D$ such that, for every node $n_i$ of the input tree $t$ and any value $0 \leq k \leq \text{size}(t)$, $D_i[k]$ is the number of subtrees of size $k$ rooted at node $n_i$.

# Unweighted Sampling for Binary Trees

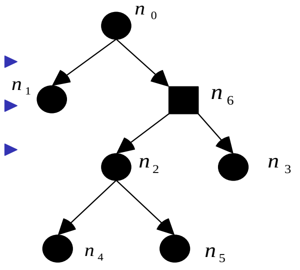


# Unweighted Sampling for Binary Trees

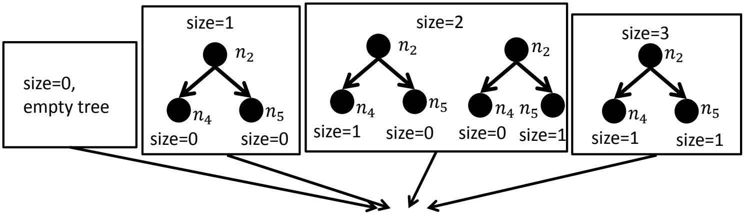



For leaf nodes, there is one rooted subtree of size 1 and one rooted subtree of size 0, therefore $D_1 = D_4 = D_5 = D_3 = (1,1)$

# Unweighted Sampling for Binary Trees




For leaf nodes, there is one rooted subtree of size 1 and one rooted subtree of size 0, therefore $D_1 = D_4 = D_5 = D_3 = (1,1)$

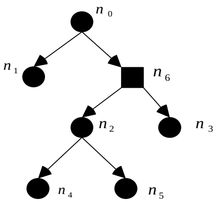



$D_2 = (1,1,2,1)$

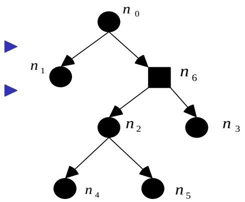
# Unweighted Sampling for Binary Trees

- **Second phase: Uniform Sampling.** We sample a rooted subtree from $t$ in a recursive top-down manner, based on the matrix $D$ computed.
- The basic idea is that to sample a rooted subtree at node $n_i$, we decide on the size of the subtrees rooted at each child node, biased by the number of outcomes as counted in $D$, and then sample rooted subtrees of the desired sizes recursively.

# Unweighted Sampling for Binary Trees

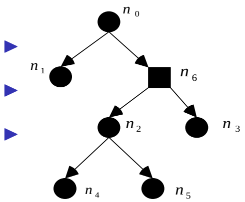
- 

# Unweighted Sampling for Binary Trees




$$D_1 = D_4 = D_5 = D_3 = (1,1)$$
$$D_2 = (1,1,2,1), D_6 = (1,1,2,3,3,1)$$
$$D_0 = (1,1,2,3,5,6,4,1)$$

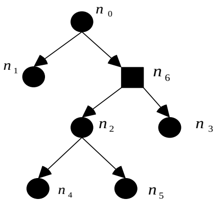
# Unweighted Sampling for Binary Trees




$$D_1 = D_4 = D_5 = D_3 = (1,1)$$
$$D_2 = (1,1,2,1), D_6 = (1,1,2,3,3,1)$$
$$D_0 = (1,1,2,3,5,6,4,1)$$

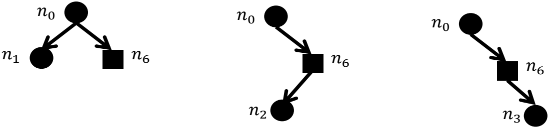
Assume we want to sample a rooted subtree of size 3. From $D_0$, we know that there are 3 such rooted subtrees.

# Unweighted Sampling for Binary Trees

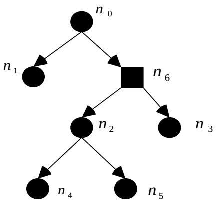



$$D_1 = D_4 = D_5 = D_3 = (1,1)$$
$$D_2 = (1,1,2,1), D_6 = (1,1,2,3,3,1)$$
$$D_0 = (1,1,2,3,5,6,4,1)$$

Assume we want to sample a rooted subtree of size 3. From $D_0$, we know that there are 3 such rooted subtrees.

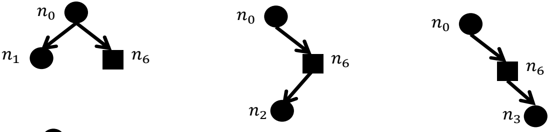


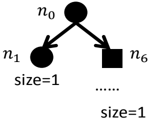
# Unweighted Sampling for Binary Trees




$$D_1 = D_4 = D_5 = D_3 = (1,1)$$
$$D_2 = (1,1,2,1), D_6 = (1,1,2,3,3,1)$$
$$D_0 = (1,1,2,3,5,6,4,1)$$

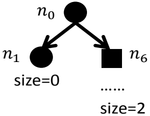
Assume we want to sample a rooted subtree of size 3. From $D_0$, we know that there are 3 such rooted subtrees.




$$\boxed{D_1[1] \times D_6[1] = 1}$$

$$p_1 = \frac{1}{3}$$




$$\boxed{D_1[0] \times D_6[2] = 2}$$

$$p_2 = \frac{2}{3}$$

# Unweighted Sampling for Binary Trees

- Complexity of Phase 1 (the size of $t$ is $n$) is $O(n^3)$.
  - Each $D_i$ has size at most $n$. Therefore computing the convolution sum of two such $D_i$'s is in $O(n^2)$.
  - We need to compute at most $n$ convolution sums. Therefore the overall complexity is $O(n^3)$.

# Unweighted Sampling for Binary Trees

- Complexity of Phase 1 (the size of $t$ is $n$) is $O(n^3)$.
  - Each $D_i$ has size at most $n$. Therefore computing the convolution sum of two such $D_i$'s is in $O(n^2)$.
  - We need to compute at most $n$ convolution sums. Therefore the overall complexity is $O(n^3)$.
- Complexity of Phase 2 (the size of $t$ is $n$) is $O(n^2)$.
  - At each node $n_i$, the number of possibilities to consider is $O(n)$, since $n_i$ has at most two children.
  - There are $n$ nodes, hence the overall complexity is $O(n^2)$.

# Unweighted Sampling for Unranked Trees

- The sampling algorithm for binary trees can be adapted for unranked trees, thanks to the associativity of the convolution sum.

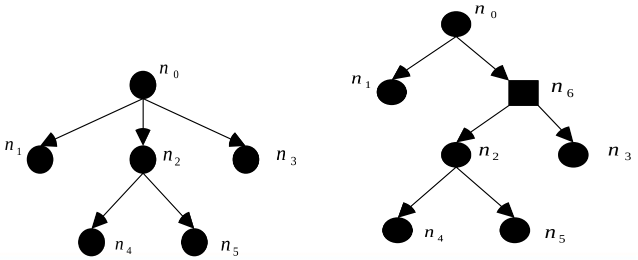
# Unweighted Sampling for Unranked Trees

- Based on this idea, we introduce "dummy node" to represent a set of (more than two) nodes. An unranked tree can be transformed to a binary tree by adding such "dummy nodes".

# Unweighted Sampling for Unranked Trees

- Based on this idea, we introduce "dummy node" to represent a set of (more than two) nodes. An unranked tree can be transformed to a binary tree by adding such "dummy nodes".
- The change in the Phase 1 and Phase 2: when we are reaching a "dummy node", this node has weight (namely size) 0.

# Unweighted Sampling for Unranked Trees

- Based on this idea, we introduce "dummy node" to represent a set of (more than two) nodes. An unranked tree can be transformed to a binary tree by adding such "dummy nodes".
- The change in the Phase 1 and Phase 2: when we are reaching a "dummy node", this node has weight (namely size) 0.
- For instance, $n_6$ is a "dummy node".
- 

# 0/1-weights Sampling

- 0/1-weights sampling problem can be resolved by adapting the sampling algorithm of unweighted sampling for unranked trees, since nodes of 0 weight are similar to dummy nodes. (Details refer to the paper)

NUS
National University of Singapore

# 0/1-weights Sampling

- 0/1-weights sampling problem can be resolved by adapting the sampling algorithm of unweighted sampling for unranked trees, since nodes of 0 weight are similar to dummy nodes. (Details refer to the paper)
- We have presented the algorithms of sampling problem for a single request. Next, we are going to discuss the case of multiple requests from a data consumer.

# Repeated Requests

- Motivation: after having bought incomplete data, the data consumer may realize that she needs additional data, in which case she would like to obtain more incomplete data that is not redundant with what she already has.

# Repeated Requests

- Motivation: after having bought incomplete data, the data consumer may realize that she needs additional data, in which case she would like to obtain more incomplete data that is not redundant with what she already has.
- Sampling problem: A rooted subtree, including nodes that she already has and a set of new nodes whose sum weight is $k$, is sampled uniformly at random.

# Repeated Requests

- Motivation: after having bought incomplete data, the data consumer may realize that she needs additional data, in which case she would like to obtain more incomplete data that is not redundant with what she already has.
- Sampling problem: A rooted subtree, including nodes that she already has and a set of new nodes whose sum weight is $k$, is sampled uniformly at random.
  - This problem is NP-hard.

# Repeated Requests

- Motivation: after having bought incomplete data, the data consumer may realize that she needs additional data, in which case she would like to obtain more incomplete data that is not redundant with what she already has.
- Sampling problem: A rooted subtree, including nodes that she already has and a set of new nodes whose sum weight is $k$, is sampled uniformly at random.
  - This problem is NP-hard.
  - The unweighted version of this problem is tractable. We can set the weights of nodes that a data consumer already has to 0. Then this sampling problem is similar to 0/1-weights sampling problem with slight adaption. (Details refer to the paper)

NUS National University of Singapore

# Conclusion and Future Work

- We proposed a framework for a data market in which data quality can be traded for a discount.

# Conclusion and Future Work

- We proposed a framework for a data market in which data quality can be traded for a discount.
- We studied the case of XML documents with completeness as the quality dimension.

NUS National University of Singapore

# Conclusion and Future Work

- We proposed a framework for a data market in which data quality can be traded for a discount.
- We studied the case of XML documents with completeness as the quality dimension.
- The data consumer proposes a price but may get only a sample if the proposed price is lower than that of the entire document.

# Conclusion and Future Work

- We proposed a framework for a data market in which data quality can be traded for a discount.
- We studied the case of XML documents with completeness as the quality dimension.
- The data consumer proposes a price but may get only a sample if the proposed price is lower than that of the entire document.
- A sample is a rooted subtree of prescribed weight, as determined by the proposed price, sampled uniformly at random.

# Conclusion and Future Work

- We proved that if nodes in the XML document have arbitrary non-negative weights, the sampling problem is intractable.

# Conclusion and Future Work

- We proved that if nodes in the XML document have arbitrary non-negative weights, the sampling problem is intractable.
- We identified tractable cases, namely the unweighted sampling problem and 0/1-weights sampling problem, for which we devised P-TIME algorithms. We also considered repeated requests and provided P-TIME solutions to the unweighted cases.

# Conclusion and Future Work

- The more general issue that we are currently investigating is that of sampling rooted subtrees uniformly at random under more expressive conditions than size restrictions or $0/1$-weights. In particular, we intend to identify the tractability boundary to describe the class of tree statistics for which it is possible to sample rooted subtrees in P-TIME under a uniform distribution.

# References I

Koutris, P., Upadhyaya, P., Balazinska, M., Howe, B., and Suciu, D. (2012a).
Query-based data pricing.
In *PODS*.

Koutris, P., Upadhyaya, P., Balazinska, M., Howe, B., and Suciu, D. (2012b).
QueryMarket demonstration: Pricing for online data markets.
*PVLDB*, 5(12).

Koutris, P., Upadhyaya, P., Balazinska, M., Howe, B., and Suciu, D. (2013).
Toward practical query pricing with QueryMarket.
In *SIGMOD*.

# References II

Li, C. and Miklau, G. (2012).
Pricing aggregate queries in a data marketplace.
In *WebDB*.

Pipino, L., Lee, Y. W., and Wang, R. Y. (2002).
Data quality assessment.
*Commun. ACM*, 45(4).

Tang, R., Shao, D., Bressan, S., and Valduriez, P. (2013).
What you pay for is what you get.
In *DEXA (2)*.

# References III

Wang, R. Y. and Strong, D. M. (1996).
Beyond accuracy: What data quality means to data consumers.
*J. of Management Information Systems*, 12(4).

# Thank you! Questions?