**XML Storage Alternatives**

- Plain Text
- Trees with Navigation
- Tuples (i.e., mapping to RDBMS)

**Plain Text**

- Use XML standards to encode data
- Advantages:
  - simple, universal
  - indexing possible
- Disadvantages:
  - need to re-parse (re-validate) all the time
  - no compliance with XQuery data model (collections)
  - not an option for XQuery processing
Trees

- XML data model uses tree semantics
  - use Trees/Forests to represent XML instances
  - annotate nodes of tree with data model info
- Examples:
  - Document Object Model (DOM)
    http://www.w3.org/DOM/
  - Object Exchange Model (OEM)

---

DataGuides [Goldman & Widom 97]

- Schema-based environments
  - Schema generates Data
  - Schema formulates Queries
  - Queries execute against Data

---

DataGuides [Goldman & Widom 97]

- Schema-free environments:
  - don’t know the schema in advance.
  - semantic heterogeneity (i.e. a mix of schemas)
  - DataGuides Summarized into App-specific Templates
  - DataGuides formulate Data
  - Data generate Queries
  - Queries execute against Data
Schema vs. DataGuides

- A DataGuide only includes info that exists in a DB.
- A schema can be a superset of any DB that conforms to it.
- So, a schema defines a superset of a DataGuide.
- Issues addressed in the paper:
  - Summarize data into DataGuides;
  - Use them for query formulation and optimization.

Object Exchange Model (OEM)

- Each object has an id (oid) and a value (atomic or a set of subobjects).
- Each edge links an object to one of its subobjects with a label; a subobject may have multiple parents.

- **Label path**: a seq. of labels
- **Data path**: an alternating seq. of labels and oids
- **Target set**: a set of all objects reached by traversing a label path

Definition of a DataGuide

- **Conciseness**: a DataGuide describes every unique label path of a source exactly once
- **Accuracy**: a DataGuide does not encode any label path that does not appear in the source
- **Convenience**: represented as an OEM model, like the data
- A DataGuide reflects the structure of a DB; it contains no atomic values.
From Data to DataGuides

- Creating a DataGuide is equivalent to converting an NFA to DFA.
  - Consider a label path (query) as a string to be accepted by the data source and the DataGuide.
  - Intuition: The data source has multiple matches, so execution is non-deterministic. But the DataGuide has only one path, so execution is deterministic.

Cost of creation
- Source DB is a tree: linear
- Worst-case: exponential in # of objects and edges in the source
- Empirical results: average performance for certain datasets is quite encouraging

Multiple DataGuides

- An OEM source may have multiple DataGuides
  - A single NFA may have many equivalent DFAs.
- Minimal DataGuide
  - Can be created using DFA minimization
- Minimality may not always be desirable
  - Hard to maintain as the data source changes—well known problem with DFA.
  - Does not allow annotations.

Annotations

- Annotation: a property of the target set of a label path in the data source
  - Statistical information: e.g. # occurrences of l in s
  - Pointers to objects reachable via l
  - ...
- Issue with minimality

  Annotation for A.C
  Annotation for B.C

   ```
   11
   ```
**Strong DataGuides**

- Each set of label paths that share a node in the DataGuide is the set of label paths that share the same target set in the source.
  - Label paths can be merged in the DataGuide if they lead to the same target set.
- There is one-to-one correspondence between source target sets and DataGuide objects.
- Creation from the data source
  - A DFS algorithm that examines source target sets reachable by all possible label paths...
- Maintenance uses a similar set of data structures...

**Query Formulation & Optimization**

- Query formulation
  - Query by example: click buttons to select a path and add value filters
  - Blurs the distinction between formulating a query and browsing a query result
- Query optimization
  - Uses the DataGuide for structural matching (e.g. A.B.C) and retrieves the target set
  - Uses value indexes (e.g. B+trees) for value filters for a specific label (e.g. C.price>100)
  - Intersects the two resulting sets of objects

**XML Data Stored as Tuples**

- Motivation: Use an RDBMS infrastructure to store and process the XML data
  - query optimization
  - scalability
  - richness and maturity of RDBMS
- Alternative relational storage approaches:
  - Map XML schema to relational schema
  - Generic shredding of the data (edge, binary, …)
  - New XML storage integrated tightly with the relational processor