

# Query Languages for Semistructured Data

---

**Michail Petropoulos**

**CSE DEPARTMENT**

University of California  
**San Diego**



# Outline

---

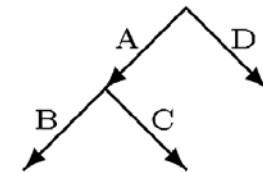
- UnQL
  - Query language for unstructured data
  - UPenn, ACM SIGMOD 96
- Lorel
  - Query language for semistructured data
  - Stanford, JODL 97
- MSL
  - Mediator specification language
  - Stanford, VLDB 96
- WebOQL
  - Query language for documents, databases and webs restructuring
  - UToronto, ICDE 98
- StruQL
  - Language for querying and restructuring semistructured data
  - AT&T Labs, ACM SIGMOD 98

# UnQL

## Data Model

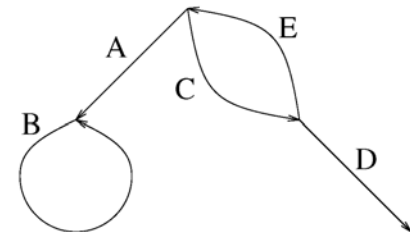
- Rooted edge-labeled directed graph (labeled tree)
  - Fixed-depth, arbitrary-depth, cyclic structures
- Information resides only at labels
- Label types: integers, strings, ..., symbols
- Example (trees):

$$\{A \Rightarrow \{B \Rightarrow \{\}, C \Rightarrow \{\}\}, D \Rightarrow \{\}\}$$



- Example (cyclic structures, *tree markers*):

$$x_1 \text{ where } x_1 = \{A \Rightarrow x_2, C \Rightarrow \{D, E \Rightarrow x_1\}\}, \\ x_2 = \{B \Rightarrow x_2\}$$



# UnQL

---

## Characteristics and Expressiveness

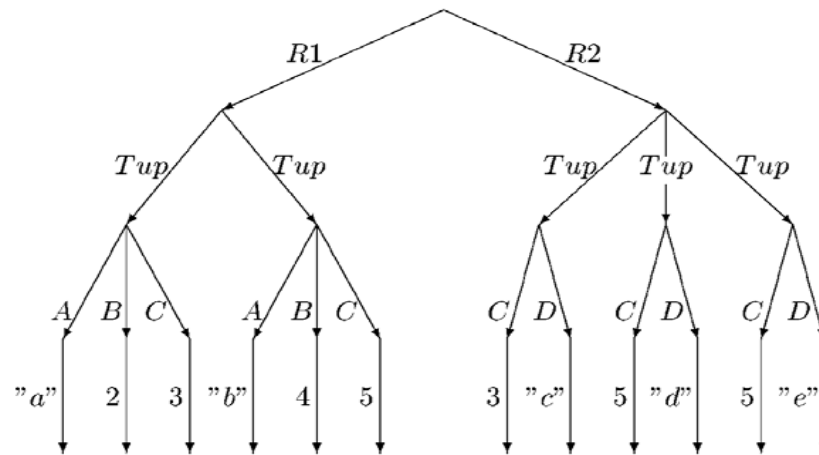
- Tree-traversing pattern-matching-based language
- Two types: Labels and Trees
- A tree is a set of pairs of labels and trees
- Queries defined even on cyclic structures
- Expressive power of relational algebra on fixed-depth structures
- Traversals of arbitrary-depth trees controlled by regular expressions on paths
- Restructuring ability at arbitrary depths
- All queries computable in polynomial time

# UnQL

## Selection Queries

- Example: Group by C column on R2

```
select {x ⇒ (select y where  
                R2 ⇒ Tup ⇒ {C ⇒ x, D ⇒ \y} ← DB) }  
where R2 ⇒ Tup ⇒ C ⇒ \x ⇒ { } ← DB
```

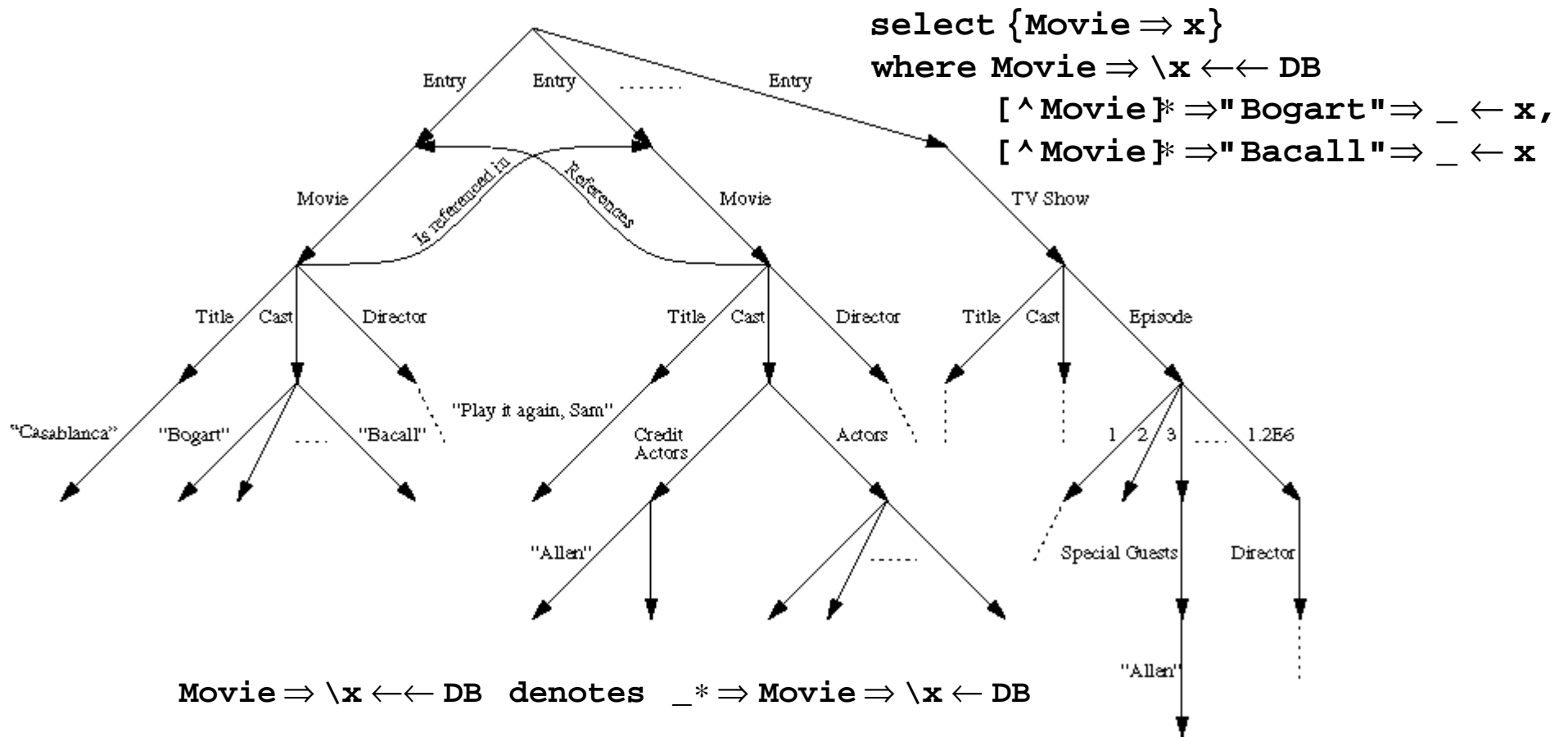


{3 ⇒ {"c"}, 5 ⇒ {"d", "e"}}

# UnQL

## Deep Queries

- Example: Find all movies involving "Bogart" and "Bacall"

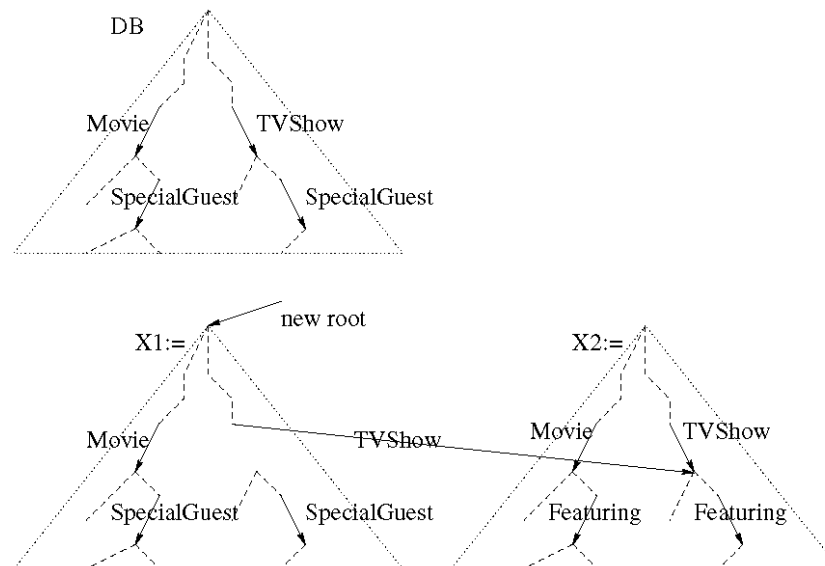


# UnQL

## Restructuring Queries

- Example: Replace every *SpecialGuest* edge with a *Featuring* edge, but only in *TVShows*

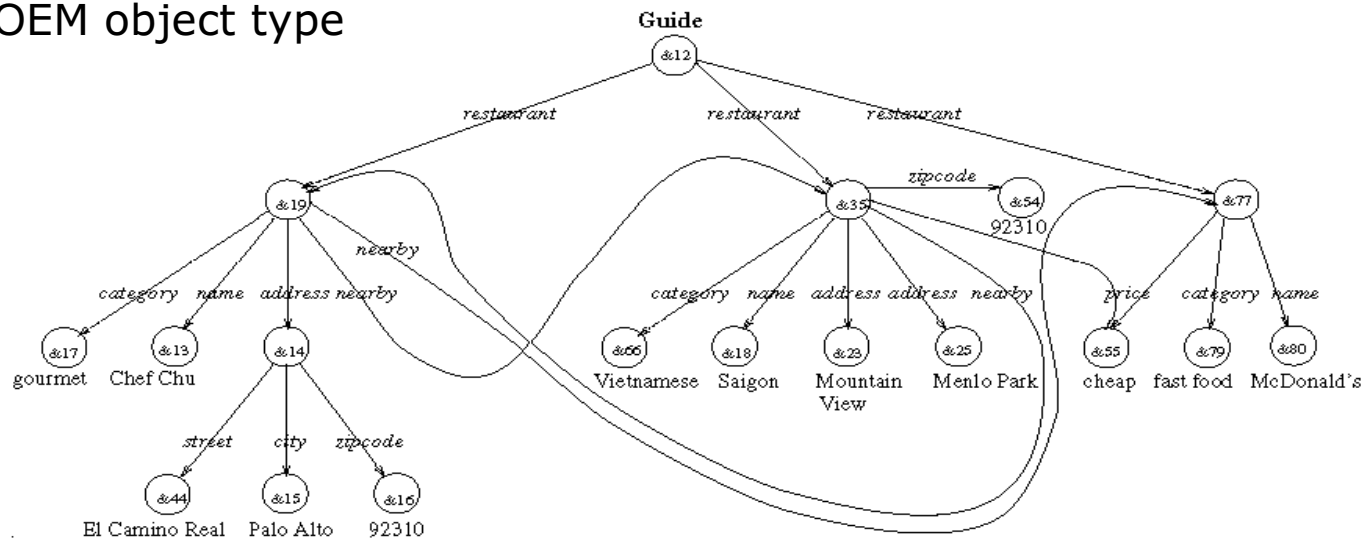
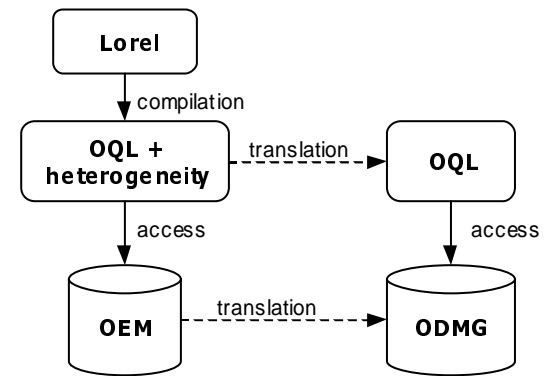
```
traverse DB giving X1, X2
  case TVShow ⇒ _
  case SpecialGuest ⇒ _
  case \1 ⇒ _
    then X1 := {TVShow ⇒ X2}
    then X2 := {Featuring ⇒ X2}
    then X1 := {1 ⇒ X1}, X2 := {1 ⇒ X2}
```



# Lorel

## Data Model

- OEM (Object Exchange Model)
  - Labeled Directed Graph
  - Objects at vertices, Labels at edges
  - Atomic and complex objects
  - Object names
- Extension of the ODMG data model
  - OEM object type





# Lorel

---

## Characteristics and Expressiveness

- User-friendly, pattern-matching-based language
- Extension of OQL
- Extensive use of coercion (relax strong typing)
- Declarative navigational access using path expressions
- Four types: atomic values, atomic objects, set of objects, complex object

# Lorel

## Coercion

- Example: Find the addresses of all restaurants with zipcode 92310

```
select X.address
from Guide.restaurant X, X.zipcode Y
where Y = 92310
```

- Rules provided for atomic types, predicates and functions
- Comparing values and atomic objects

<i>arg1</i>	<i>arg2</i>	<i>string</i>	<i>real</i>	<i>int</i>
<i>string</i>	–		<i>string</i> → <i>real</i>	<i>both</i> → <i>real</i>
<i>real</i>			–	<i>int</i> → <i>real</i>
<i>int</i>				–

- Comparing objects and sets of objects
  - Two types of equality: "=" and "=="

<i>arg1</i>	<i>arg2</i>	<i>value</i>	<i>atomic object</i>	<i>set of objects</i>	<i>complex object</i>
<i>value</i>		<i>coerce</i>	<i>dereference</i>	<i>existential with ==</i>	<i>false</i>
<i>atomic object</i>			<i>value =</i>	<i>existential with ==</i>	<i>false</i>
<i>set of objects</i>				<i>existential with == on both sides</i>	<i>false</i>
<i>complex object</i>					<i>value =</i>

# Lorel

## Simple Path Expressions

- Can be translated to OQL queries
- Example: Select prices more than 25

```
select Guide.restaurant.price
from Guide.restaurant
where Guide.restaurant.price > 25
```

```
select (select P from R.price P)
from Guide.restaurant R
where exist Q in R.price : Q > 25
```

WRONG

```
select (select P from R.price P where P > 25)
from Guide.restaurant R
where exist Q in R.price : Q > 25
```

CORRECT

```
Guide &12
restaurant &19
  category &17 "gourmet"
  name &13 "Chef Chu"
  address &14
    street &44 "El Camino Real"
    city &15 "Palo Alto"
    zipcode &16 92310
  nearby_eating_place &35
  nearby_eating_place &77
restaurant &35
  category &66 "Vietnamese"
  name &18 "Saigon"
  address &23 "Mountain View"
  address &25 "Menlo Park"
  nearby_eating_place &19
  zipcode &54 "92310"
  price &55 "cheap"
restaurant &77
  category &79 "fast food"
  name &80 "McDonald's"
  price &55
  price &10
```

# Lorel

---

## General Path Expressions

- Cannot be translated to OQL queries
- Allow regular expressions and label completion
- Examples:

```
Guide.restaurant(.address)?.zipcode  
Guide.restaurant.#@P.comp%.name  
Guide.restaurant(.nearby)*{R}.name
```

- Path Variables and path distinguishers
- Example: Select restaurants with two distinct paths to the same nearby restaurant

```
select R  
from Guide.restaurant R  
where R(.#.nearby)@P = R(.#.nearby)@Q and P <> Q
```

# MSL

---

## Data Model

- OEM
  - Schema-less, self-describing, object-oriented model
  - Same as Lorel, but labels are migrated to nodes
  - OEM objects have object-id, label, type and value
  - Three types: Atomic value, atomic object, set of objects

```
<&r1, report, set, {&r1n, &r1a, &r1t, &r1y, &r1r}>
  <&r1n, report_num, string, 'AB-123-456'>
  <&r1a, authors, set, {&b1a1}>
    <&r1a1, author, string, 'John Patriot'>
  <&r1t, title, string, 'UN Conspiracies'>
  <&r1y, year, string, '1995'>
  <&r1r, related, set, {&r2}>
    :
  <&r2, report, set, {&r2n, &r2a, &r2t, &r2y, &r2r}>
```

# MSL

---

## Characteristics and Expressiveness

- Mediator specified by a set of non-procedural rules
- Each rule maps a set of objects at a source
- Additional sources require more rules for a specification
- Assignment of semantically meaningful object ids at the mediator using Skolem functions
- Fusion operations specified non-procedurally
- Removal of redundancies and resolution of inconsistencies between sources
- Integration of nested and cross-referenced objects (Web)

# MSL

---

## Object Fusion

- Example: Fuse reports that have the same report number from sources s1 and s2

```
(MS1) (R1.1) <trep(RN) techreport {<title T >}>@simple :-  
        <report {<report_num RN > <title T >}>@s1  
      (R1.2) <trep(RN) techreport {<postscript P >}>@simple :-  
        <report {<report_num RN > <postscript P >}>@s2
```

- Rules can incrementally and independently insert information into a semantically identified mediator object
- Example: Retrieve all data of techreports's with object-id `trep('123')`

```
(Q1) <trep('123') techreport V > :- <trep('123') techreport V >@simple
```

# MSL

---

## Removing Redundancies

- Example: Group all information about reports into techreports objects

```
(MS2) (R2.1) < trep(RN) techreport {< rnOID(RN) report_num RN > < O1 L1 X1 >} > @nored :-  
    < report {< report_num RN > < O1 L1 X1 >} > @s1 AND NOT L1 = report_num  
(R2.2) < trep(RN) techreport {< rnOID(RN) report_num RN > < O2 L2 X2 >} > @nored :-  
    < report {< report_num RN > < O2 L2 X2 >} > @s2 AND NOT L2 = report_num
```

- The structure of the source reports is not required



# MSL

---

## Resolving Inconsistencies

- Blocking sources at various levels of granularity
- Example: s1 is accessed for free, s2 is not

```
(MS3) (R3.1) <trep(RN) techreport V >} >@save :-  
        <report V :{<report_num RN >} >@s1  
      (R3.2) provides(RN) :- <report {<report_num RN > <author A >} >@s1  
      (R3.3) <trep(RN) techreport V >@save :-  
        NOT provides(RN) AND <report V :{<report_num RN >} >@s2
```

- Dynamic priority assignment
- Fine-grained blocking at subobjects level

# MSL

---

## Handling References

- Solution 1 (inexpensive, weak):
  - Subobjects that contain references must be known

```
(MS4) (R4.1) < trep(RN) techreport {L X} > @ref :-
```

```
    < report {< report_num RN > < L X >} @s1 AND NOT L = related
```

```
(R4.2) < trep(RN) techreport {related {< trep(REL) techreport {} >} >} > @ref :-
```

```
    < report {< report_num RN > < related {< report_num REL >} >} >} > @s1
```

- Solution 2 (expensive, powerful):
  - Copies needed
  - If references between sources exist, more copies are needed

```
(MS5) (R5.1) < trep(RN) techreport {L X} > @ref :-
```

```
    < report {< report_num RN > < L X >} @s1
```

```
(R5.2) < O techreport V : {report_num RN} > @ref :-
```

```
    AND < O report {< report_num RN >} > @s1
```

# WebOQL

---

## Data Model

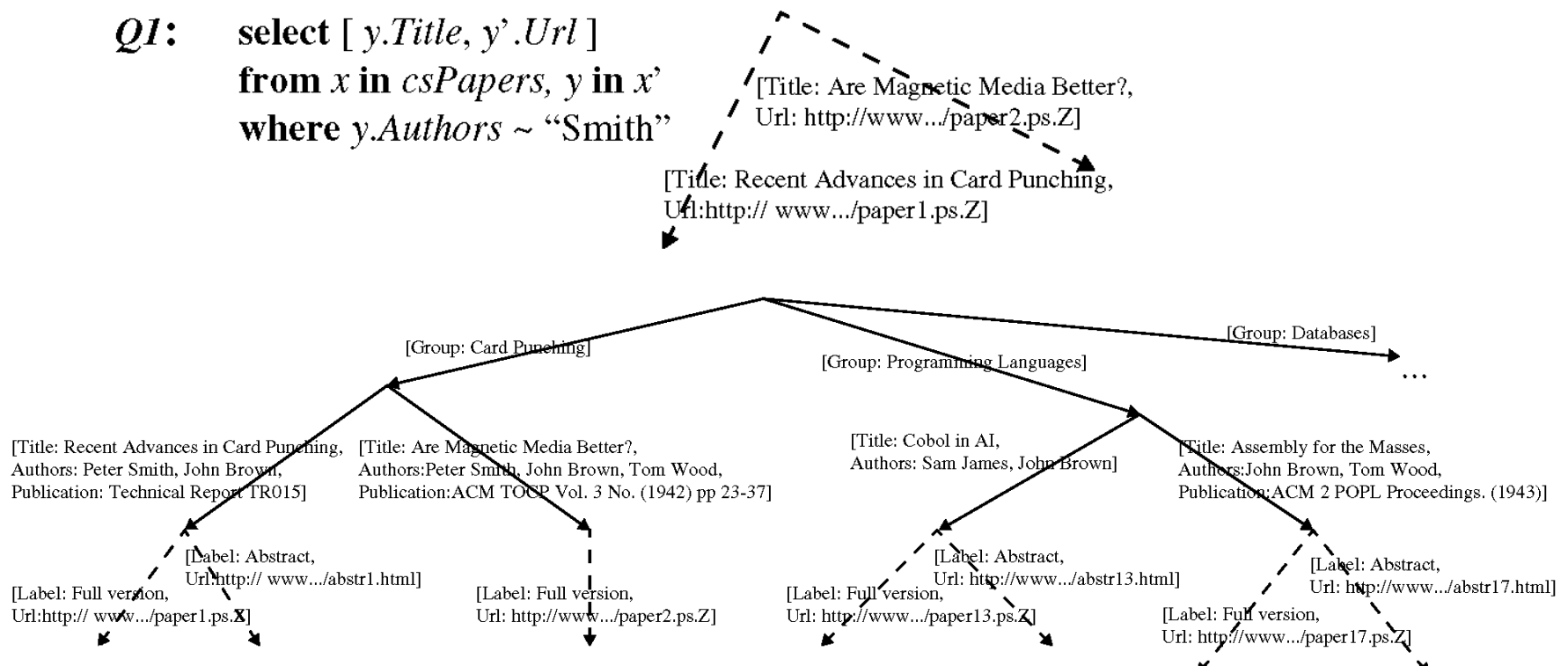
- Hypertrees
  - Ordered arc-labeled trees
  - Contains internal (structured objects) and external (references) arcs
  - Represents a document or a relation
  - Three abstractions: collections, nesting, ordering, references
- Webs
  - Models a set of related hypertrees
  - Consists of a hypertree  $t$  (schema) and a function  $F$  that maps URLs to hypertrees (browsing function)
  - The browsing function defines a graph of hypertrees (pages)
  - The schema provides entry points to the graph
  - URL dereferencing using the browsing function ( $F(u)$  is the content of the page,  $u$  is a URL)

# WebOQL

## Restructuring Simple Trees

- Example: Retrieve the title and URL of the full version of papers authored by "Smith"

**Q1:** `select [ y.Title, y.Url ]  
from x in csPapers, y in x'  
where y.Authors ~ "Smith"`

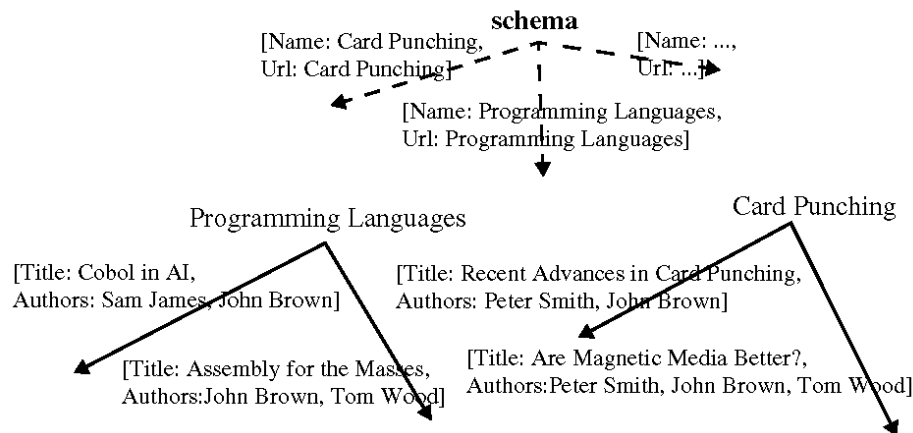


# WebOQL

## Restructuring Webs

- How to map one web into another
- New schema, redefined browsing function
- Example: Create a new web containing one page per group (title,author/publication) and an index page

```
Q11: newWeb ← select unique  
           [Name:x.Group,Url:x.Group] as schema,  
           [ y.Title, y.Authors ] as x.Group  
           from x in csPapers, y in x'
```



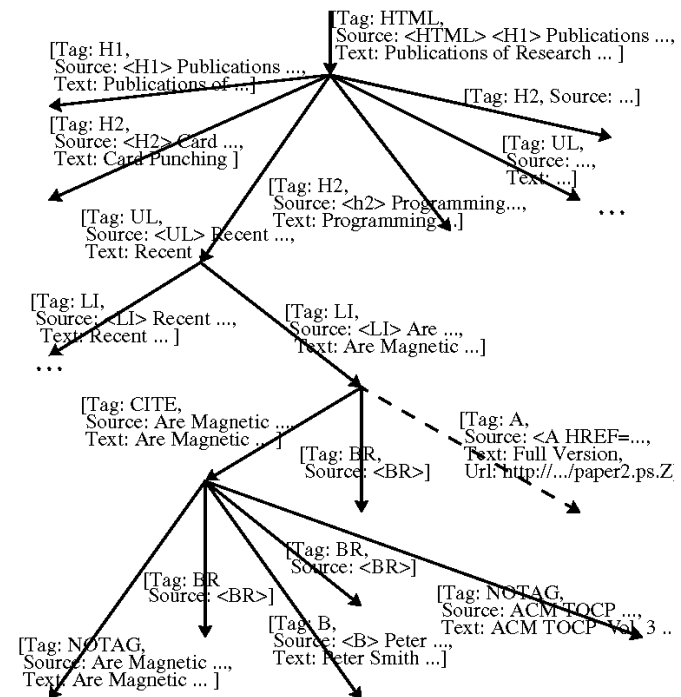
# WebOQL

## Annotated Abstract Syntax Trees (ASTs)

- HTML documents are modeled with their annotated abstract syntax trees (ASTs)

```
<HTML>
<H1>Publications of Research Groups at CS Department</H1>
<H2> Card Punching </H2>
<UL>
<LI>
<CITE> Recent Advances in Card Punching <BR>
<B> Peter Smith, John Brown </B> <BR>
Technical Report TR015 </CITE> <BR>
<A HREF="http://.../paper1.ps.Z"> Full version </A>
<A HREF="http://.../abstr1.html"> Abstract </A> <BR>
</LI>
<LI>
<CITE> Are Magnetic Media Better? <BR>
<B> Peter Smith, John Brown, Tom Wood </B> <BR>
ACM TOCP Vol. 3 No.1 (1942) pp 23-37</CITE> <BR>
<A HREF="http://.../paper2.ps.Z"> Full version </A>
</LI>
</UL>

<H2> Programming Languages </H2>
<UL>
<LI>
<CITE> Cobol in AI <BR>
<B> Sam James, John Brown </B> <CITE> <BR>
<A HREF="http://.../paper13.ps.Z"> Full version </A>
<A HREF="http://.../abstr13.html"> Abstract </A> <BR>
</LI>
...
<H2> Databases </H2>
...
</HTML>
```



# WebOQL

---

## Restructuring Documents

- Navigation patterns (NPs)
  - Variables can range over linked hypertrees
  - Example: Extract the names of all research groups (tagged with H2)

```
Q13:  select [ x.Text ]  
        from x in “papers.html” via ^*[Tag = “H2”]
```

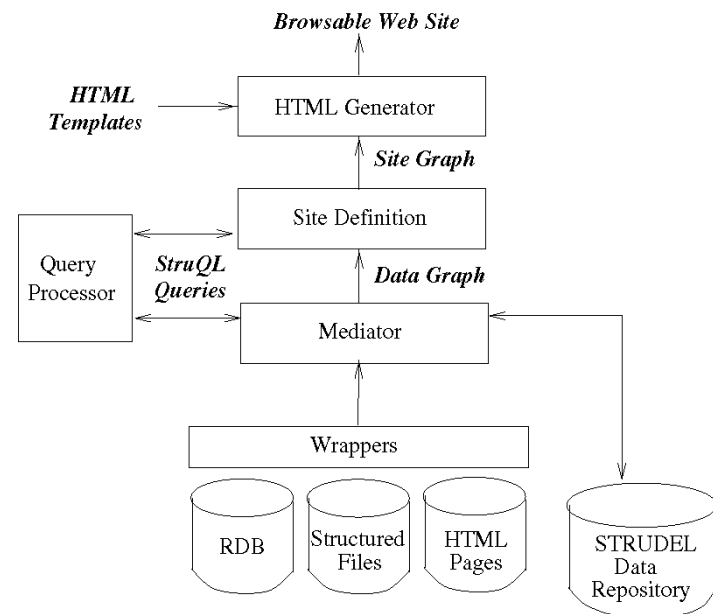
- Tail Variables
  - Example: Restructure the list of papers for a group into an HTML ordered list

```
Q15:  [ Tag: “OL” /  
        select [ Tag: “LI” / X&3 ]  
        from X in browse(“Card Punching.html”)!  
        where X.Tag = “H3”  
        ]
```

# StruQL

## Characteristics

- Strudel Web-Site Management System
  - Three independent phases
  - Labeled directed graph data model
- StruQL
  - Declarative specification of a web site's content (datagraph) and structure (site-definition query)
  - Restructuring operations on graphs





# References

---

- "*A Query Language and Optimization Techniques for Unstructured Data*", P. Buneman, S. Davidson, G. Hillebrand, D. Suciu (ACM SIGMOD 96)
- "*The Lorel Query Language for Semistructured Data*", S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Wiener (JODL 97)
- "*Object Fusion in Mediator Systems*", Y.Papakonstantinou, S. Abiteboul, H. Garcia-Molina (VLDB 96)
- "*WebOQL: Restructuring Documents, Databases and Webs*", G. Arocena, A. Mendelzon (ICDE 98)
- "*Catching the boat with Strudel: Experiences with a Web-Site Management System*", M. Fernandez, D. Florescu, J. Kang, A. Levy, D. Suciu (ACM SIGMOD 98)