



XML Databases

II. XML Query languages

II.1 Introduction

Weeks 4 - 6

Outline

II.1 Introduction

- A. Query Language
- B. Requirements
- C. History

Query Language

- “A high-level, human-like language provided by a database management system that enables users to easily extract data and information from a database” (*Management Information System glossary*)
- “Query languages are computer languages used to make queries into databases and information systems.” (*Wikipedia*)

Basic needs of a QL

- Abstraction from the data: definition of a data model

Ex: the relational model

- Abstraction from the implementation: query language are declarative (always?)

Ex: SQL language

QL Example

- The CQL (Cook Query Language)

- Data: ingredients, utensils, ...
- Language: Preheat, Add, Mix, Beat, Fold, Spray
- Example:
Preheat pancake griddle(18 inch) **as** PG;
Sift flour(2 cups) **Into** mixing bowl(3 to 4 quart)
as BMB; **Add** sugar(3 tablespoons) **Into** BMB;

A. Requirements and properties

- From databases
 - ☐ datatypes
 - ☐ selection, filtering, reduction, merging
 - ☐ navigation
 - ☐ insert, update and delete
- From information retrieval (Part IV)
 - ☐ Vague structural constraints
 - ☐ Weighting

A.1. Input/Output

- Output $\not\subseteq$ Input
 - No special property
- Output \subseteq Input
 - Composition
- Output = Input
 - Closure (important for views)

A.2. Query operations

- Selection: choosing document(s) or element(s)
- Extraction: choosing element(s) from selected
- Combination: merging elements
- Reduction: removing selected sub-elements
- Restructuring: construct elements

A.2.1. Selection

document (rank < 10)

```
<manufacturer>
<mn-name>Mercury</mn-name>
<year>1999</year>
<model> <mo-name>Sable
LT</mo-name>
<front-rating>3.84</front-rating>
<side-rating>2.14</side-rating>
<rank>9</rank>
</model>
<model>
...
</model>
...
</manufacturer>
```

document (price < 10)

```
<vehicle>
<vendor>Scott
Thomason</vendor>
<make>Mercury</make>
<model>Sable LT</model>
<year>1999</year>
<color>metallic blue</color>
<option opt="sunroof"/>
<option opt="A/C"/>
<option opt="lthr seats"/>
<price>26800</price>
</vehicle>
```

A.2.2. Extraction

manufacturer

```
<manufacturer>
<mn-name>Mercury</mn-name>
<year>1999</year>
<model> <mo-name>Sable
LT</mo-name>
<front-rating>3.84</front-rating>
<side-rating>2.14</side-rating>
<rank>9</rank>
</model>
<model>
...
</model>
...
</manufacturer>
```

vehicle

```
<vehicle>
<vendor>Scott
Thomason</vendor>
<make>Mercury</make>
<model>Sable LT</model>
<year>1999</year>
<color>metallic blue</color>
<option opt="sunroof"/>
<option opt="A/C"/>
<option opt="lthr seats"/>
<price>26800</price>
</vehicle>
```

A.2.3. Reduction

remove model with rank > 10... remove color and option

```
<manufacturer>
<mn-name>Mercury</mn-name>
<year>1999</year>
<model> <mo-name>Sable
LT</mo-name>
<front-rating>3.84</front-rating>
<side-rating>2.14</side-rating>
<rank>9</rank>
</model>
<model>
...
</model>
...
</manufacturer>
```

```
<vehicle>
<vendor>Scott
Thomason</vendor>
<make>Mercury</make>
<model>Sable LT</model>
<year>1999</year>
<color>metallic blue</color>
<option opt="sunroof"/>
<option opt="A/C"/>
<option opt="lthr seats"/>
<price>26800</price>
</vehicle>
```

A.2.4. Combination

`<mn-name> = <make>`, `<mo-name> = <model>` and `<year> = <year>`

```

<manufacturer>
<mn-name>Mercury</mn-name>
<year>1999</year>
<model> <mo-name> Sable
LT</mo-name>

<vehicle>
<vendor>Scott
Thomason</vendor>
<make>Mercury</make>
<model> Sable LT</model>
<year>1999</year>

<rank>9</rank>
</model>

<price>26800</price>
</vehicle>

</manufacturer>
  
```

A.2.5 Restructuring

`<mn-name>` = `<make>`, `<mo-name>` = `<model>` and `<year>` = `<year>`

`<manufacturer>`

`<mn-name>Mercury</mn-name>`

`<year>1999</year>`

`<model>` `<mo-name>Sable
LT</mo-name>`

`<rank>9</rank>`

`</model></manufacturer>`

`<vehicle>`

`<vendor>Scott`

`Thomason</vendor>`

`<make>Mercury</make>`

`<model>Sable LT</model>`

`<year>1999</year>`

`<price>26800</price>`

`</vehicle>`

`<car>`

`<make>Mercury</make>`

`<model>Sable LT</model>`

`<vendor>Scott Thomason</vendor>`

`<rank>9</rank>`

`<price>26800</rank>`

`</car>`

A.3. Schema

- No Schema Required

Usable on (XML) data when there is no schema (DTD, XML Schema, ...) known in advance

- Exploit Available Schema

Possible inference on type and early error detection

A.4. Semantics

- Precise (to allow reasoning)
 - Result structure, equivalence and containment
- Compositional
 - Stable meaning: an expression always means the same whatever the context
 - “Expressions with equal result types should be allowed to appear in the same contexts”
 - Not true in SQL
 - Interest: replacing a typed value by an expression with the same type

A.5. Misc

- In a document, XML elements are:
 - naturally ordered (ex. authors of an article)
 - nested
- XLink, XPointers and namespace aware
- Representation:
 - Simple (can be manipulated by programs)
 - XML (to use existing tools)
 - Mutually embedding with XML (to build views)
- Locators, collection and stream processing

A.6. Summary of properties

- Composition
- Query operation: Selection, Extraction, Reduction, Combination, Restructuring
- Schema and datatypes
- Semantics
- XML data model
- XML technologies (XLink, namespace, ...)
- Allows updating

B. Locating information

- Region Algebra
 - Ex. Proximal Nodes
 - Generation + Operations
- Tree Patterns
 - Related to QBE
 - Query = XML document with bindings
- Path Expression
 - Related to OQL
 - Navigation in the structure(s) aka “steps”

B.1. Region algebra

■ Generation

- An element, a text portion = region (usually a segment)
- Ex. “red cat” returns all regions which contains exactly “red cat”

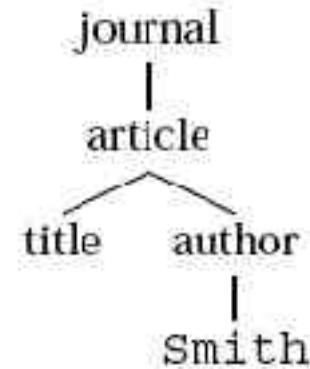
■ Operations

- Set: union, intersection, ...
- Hierarchical: contained, contains, parent, ...

B.2. Tree patterns

- A query = Tree
 - Annotated edges?
 - Variable bindings
- Example: OQL

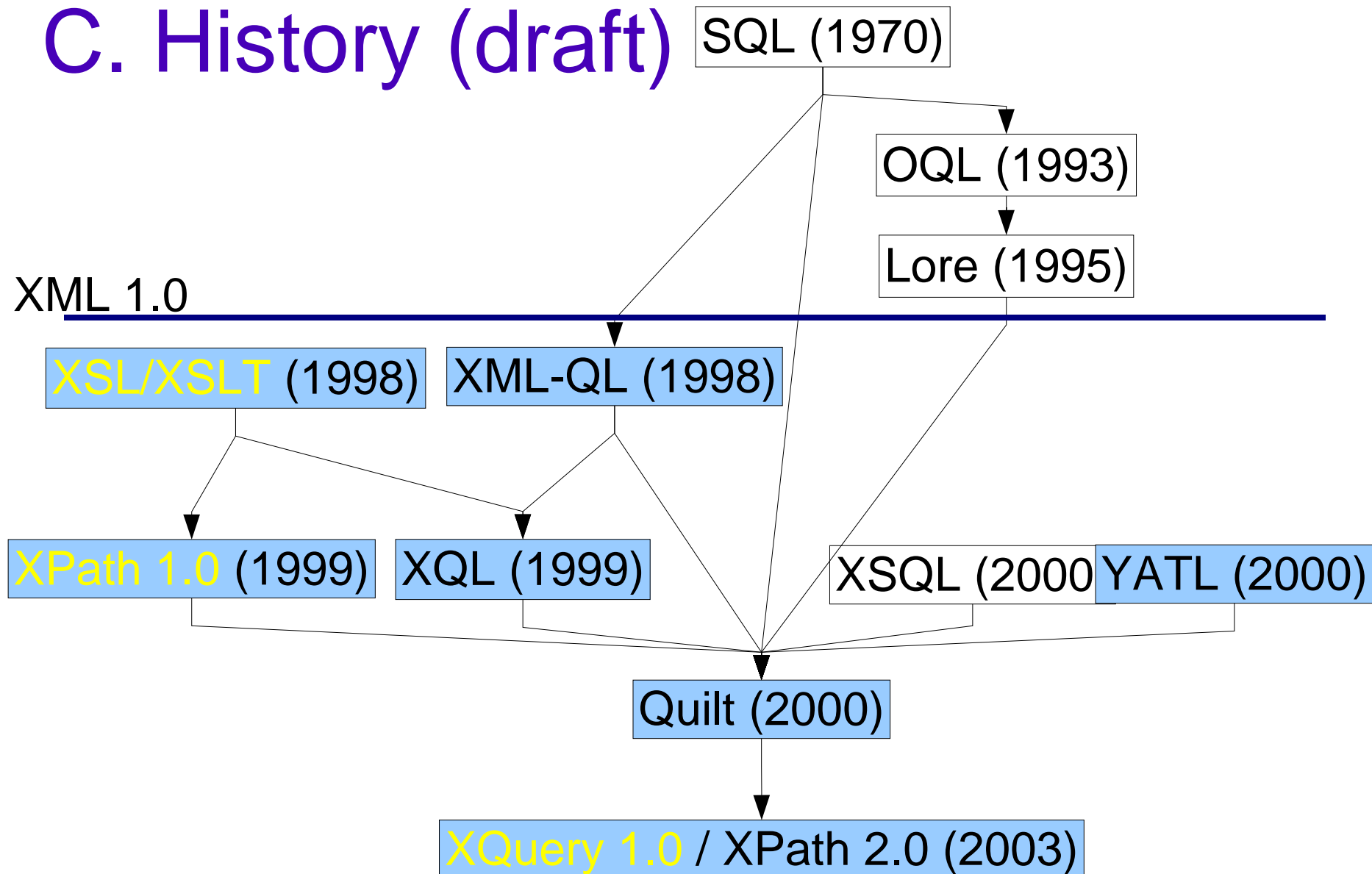
```
<$p>  
  <title> $t </title>  
  <year> 1995 </>  
  <$e> Smith </>  
</> IN "www.a.b.c/bib.xml",  
$e IN {author, editor}
```



B.3 Regular Path Expression

- Gives an indication on the navigation
 - Example: take the document A, then one of its section, then one of the section paragraph...
- Example: XPath 1.0 = sequence of steps. A step:
 - An optional axe (child, parent, ...)
 - A filtering expression
 - ex. node name
 - elaborated boolean predicates
 - Ex. `//sec/p[@id = "3"]`

C. History (draft)

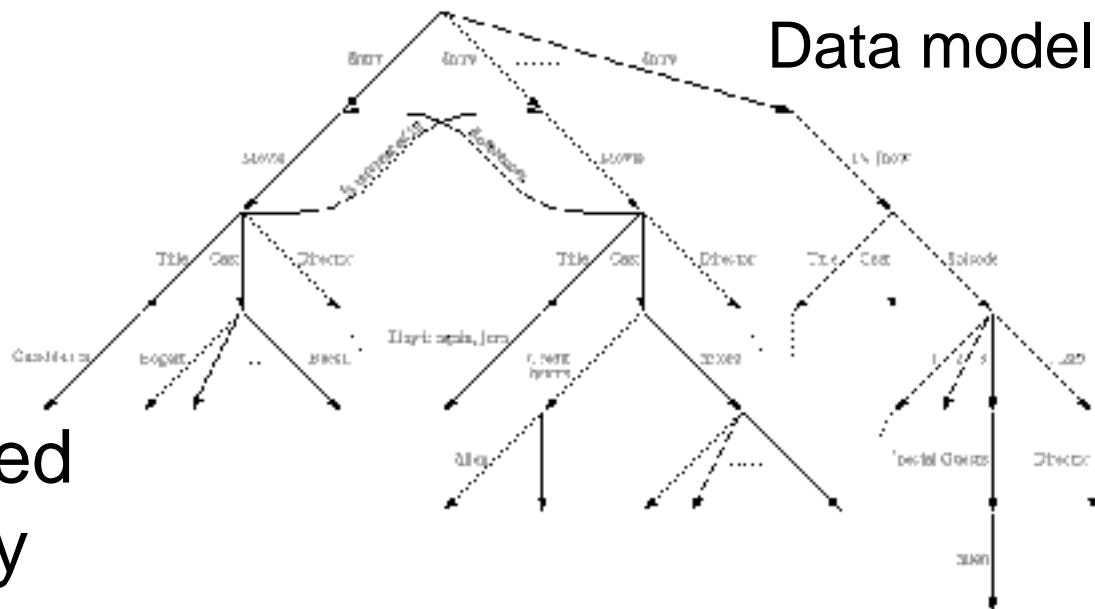


C.1. UnQL

- **Model:**
 - Rooted, edge-labelled
 - Data on edges only
 - Joins
- **Example:** Return a page labeled "Result:" pointing to all the pages in the graph (reachable from the root) containing the string "John":


```
select "Result:".p where __.* > __.p in
http://www.huji.ac.il; p matches "John";
```

More information: UnQL: a query language and algebra for semistructured data...



C.2. Lore (1995)

- Model: Graph with labelled edges (evolved to XML data)
- Regular path expression, joins, update
- Query language close to OQL
- Example:

```
SELECT $R.Name, $R.Phone  
FROM Repertory.Hotel $H, Guide.Restaurant $R  
WHERE $H.Adress.Town.Street=$R.Adress.Town.Street  
AND $H.Name="Blue fish"
```


C.3. XML-QL (1998)

- XML Data Model (but no schema support)
- Process
 - Selection (tree-patterns + path expression)
 - Filtering (predicates)
 - Construction

■ Example

WHERE <\$e> <title> \$t </><year> \$y </> </> CONTENT_A \$p IN "
www.a.b.c/bib.xml", \$y > 0

CONSTRUCT <result ID=ResultID(\$p)> <title> \$t </> </>
 { WHERE \$e = "journal-paper", <month> \$m </> IN \$p
 CONSTRUCT <result ID=ResultID(\$p)> <month> \$m </> </> }
 { WHERE \$e = "book", <publisher>\$q </> IN \$p
 CONSTRUCT <result ID=ResultID(\$p)> <publisher>\$q </> </> }

More information: <http://www.research.att.com/~mff/files/final.html>

C.4. XQL (1998)

- Ancestor of XPath
- Extension of URL
 - XML specific (ex: @ denotes an attribute)
 - Axes (descendant)
 - Predicates
- Example:

`//authors/author/address[@type='email']`

More information: <http://www.ibiblio.org/xql/xql-proposal.html>