**DSE 250 10.14.2017**

QBE (query by example) -> multi-stage query ie. nested query in SQL requires intermediate results

* Can return either all of the row or only the key

In OQL cannot do d.fas\_faculty.name would need to specify what you want via a nested query

**Introduction Web Data Management & Distribution 🡪 XML**

XML is generalized HTML (annotated text)

Semi-structured data model, based on a graph/tree

Self-describing data ie. association lists (label-value pairs), natural extension (values may themselves be other structures), further extension (allow duplicate labels)

Nodes can be identified even if data structure is irregular

What SQL does for relational dbs, XQuery does for tree structures.

XML doc is labeled (some annotation, the label, is attached to each node), unranked (means there is no apriori bound on the # of children of a node), ordered (there is an order btw. children of each node)

Unlike elements, attributes are not ordered, & there cannot be 2 attributes w/the same name in an element.

The 1st line of the serialized form must always be the prologue if there is one & the document content must always be enclosed in a single opening/ending tag called the element root

Literal sections – use entities for all special symbols; or prevent parsing w/a literal section

Kind of typing -> possibly irregular, partial, tolerant, flexible; possibly evolving; possibly very large & complex; ignored by some applications ie. keyword search

* Typing is not compulsory

Interpreting labels: Namespaces (a particular labels ie. job may denote diff. notions in diff. contexts)

**XPath**

Node types:

Document -> the root node of the XML doc

Element -> element nodes; has a name but no value

Attribute -> attribute nodes, rep. as children of an Element node; has both a name & a value

Text -> text nodes, ie. leaves of the XML tree; has a value (character string) but no name

Path that begins with / -> absolute path expression; w/o is a relative path expression

Ie. /A/B/@att1[.>2] denotes all the Attribute nodes @att1 whose value is greater than 2

* . is a special step, which refers to the context node

To evaluate /A/B/@att1 in SQL:

SELECT

FROM edge A, edge B, edge attr1

WHERE A.src = 0

AND B.src = A.tgt

AND A.tgtlabel = ‘A’

AND B.tgtlabel = ‘B’

AND attr1.src = B.tgt

AND attr.tgtlabel = ‘att1’

Axes = set of nodes determined from the context node & an ordering of the sequence

Ie. child, parent, attribute, descendent, ancestor, following, etc.

Predicate evaluation requires several rules for converting nodes & node sets to the appropriate type

Ie. //B[@att1=1] nodes B having an attribute att1 with value 1

Examples

Child::A/descendant::B -> A//B

Child::\*/child::B -> \*/B

Child::B[position()=last()] -> B[last()]

Child::B[child::C] -> B[C]

/descendent::B[@att1 or @att2] -> //B[@att1 or @att2]

In XPath 2.0 have path expressions

Ie. element(\*, xs:person) -> any element of type xs:person

* Any expression that returns a sequence of nodes can be used as a step ie. /book/(author | editor)/name

**XQuery**

A declarative language, good at efficiently retrieving some content from large (collections of) documents

There exist 7 kinds of nodes:

Document -> doc root

Element -> named, mark the structure of the doc

Attributes -> named & valued, associated to an Element

Text -> unnamed & values

Comment

ProcessingInstruction

Namespace

There is no distinction btw. an item & a sequence of length 1 -> everything is a sequence

-a sequence may be empty, contain heterogenous items, are ordered, cannot be nested

XQuery allows the construction of new elements, whose content may freely mix literal tags, literal values, & results of XQuery expressions

* An expression e must be surrounded by curly braces {} in order to be recognized & processed

A variable is a name that refers to a value, it can be used in any expression (ie. identity) in its scope

The most powerful expressions in XQuery are a FLWOR expression:

Iterates over sequences (for, will iterate through variables defined by let

Defines and binds variables (let)

Apply predicates (where)

Sort the result (order)

Construct a result (return)

* Not all FLWR expressions can be rewritten w/XPath

For and let

For -> successively binds each item from the input sequence

Ie. for $x in /company/employee -> binds each employee to $x, for each item in the company sequence

Let -> binds the whole input sequence

Ie. let $x := /company/employee -> binds $x to all the employees in company

For + return = an expression