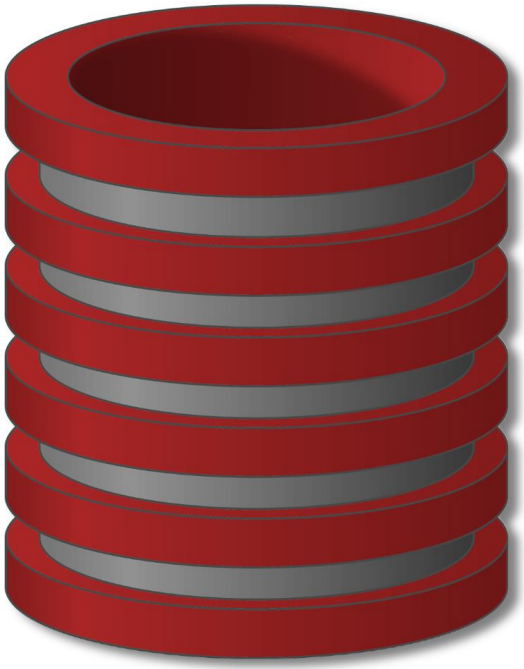


Querying XML

XPath

XPath

Querying XML

Not nearly as mature as Querying Relational

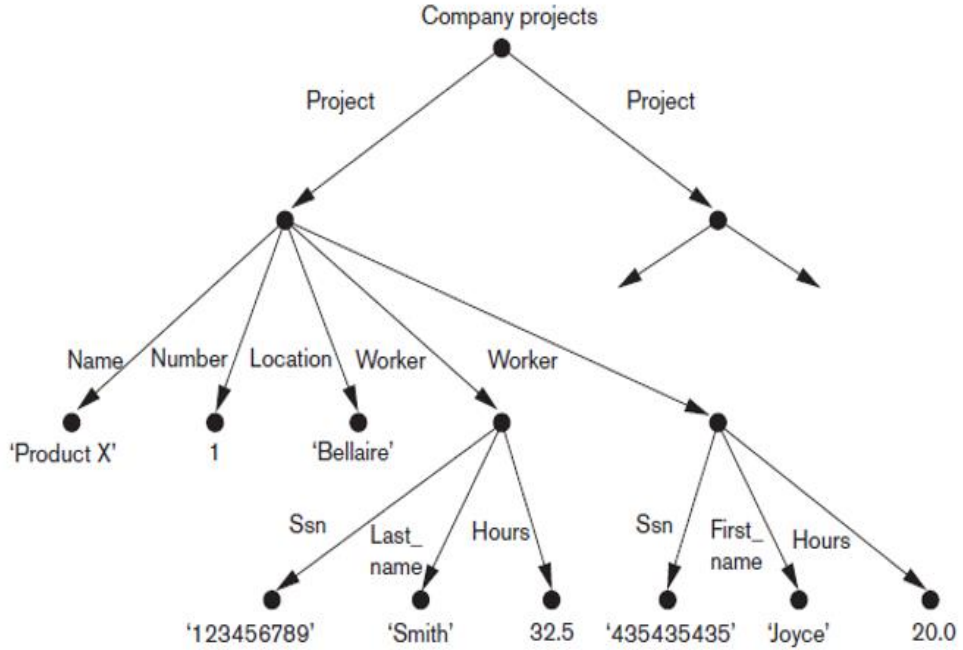
- Newer
- No underlying algebra

Sequence of development

1. XPath: Path expression + conditions
2. XSLT: Xpath transformation, output formatting
3. XQuery: Xpath +full featured QL

XPath = *Path expressions + Conditions*

Think of XML as a tree



Basic Constructs

/: root element separator

Element name X

*: Match any node in the path

@: attribute name

// : any descendent or self wildcard

[C]: conditions

[N]: access to the children nodes based on their positions

/projects/project/worker/*

/projects/project/worker/@Ssn

//Project//workers

/projects/project[@Hours>20]

/projects/project/worker[7]

XPath = *Path expressions + Conditions*

Built-in functions (lots of them)

starts-with() and contains() built-in functions work on string values and can be useful to access elements based on substring matches.

/companyDB/employees/employee[starts-with(lname,"S")]
/companyDB/employees/employee[contains(address,"Philadelphia")]

Navigation “axes” (13 of them):Keywords that allows us to move in multiple directions from current node in path expression

Include self, child, descendent, attribute, parent, ancestor, previous sibling, and next sibling

Parent:: go up the parent, Following-sibling:: match the sibling of the current node, Self:: Descendent::

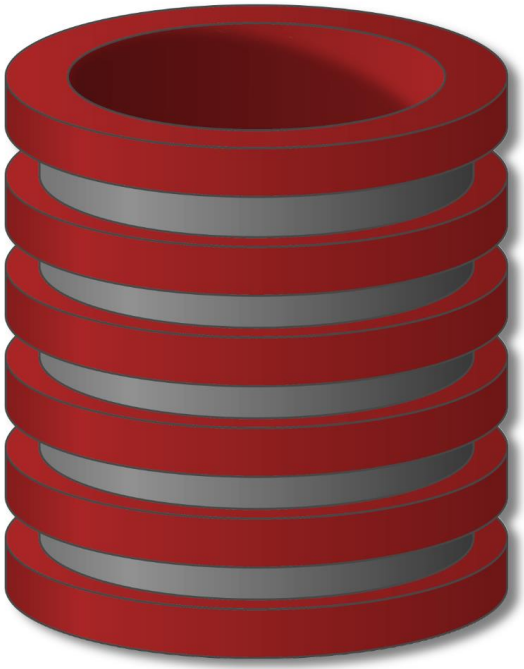
More Details

XPath queries operate on & return *sequence of elements*

- XML document
- XML stream

Sometimes result can be expressed as XML, not always

Demo: XPath examples over bookstore data



Querying XML

XQuery

Querying XML

Not nearly as mature as Querying Relational

- Newer
- No underlying algebra

Sequence of development

1. XPath
2. XSLT
3. XQuery

XQuery

- Expression language (compositional)
- Each expression operates on & returns *sequence of elements*
- XPath is one type of expression

XQuery: FLWOR expression

```
For $var in expr
Let $var := expr
Where condition
Order By expr
Return expr
```

```
FOR <variable bindings to
individual nodes (elements)>
LET <variable bindings to
collections of nodes (elements)>
WHERE <qualifier conditions>
ORDER BY <Ordering
specifications>
RETURN <query result
specification>
```

- All except **Return** are optional
- **For** and **Let** can be repeated and interleaved

Mixing queries and XML

```
<Result> { …query goes here… } </Result>
```

Demo: XQuery examples over bookstore data

Summary

- Three main types of data: structured, semi-structured, and unstructured
- XML standard
  - Tree-structured (hierarchical) data model
  - XML documents and the languages for specifying the structure of these documents
- XPath and XQuery languages
  - Query XML data