Announcements

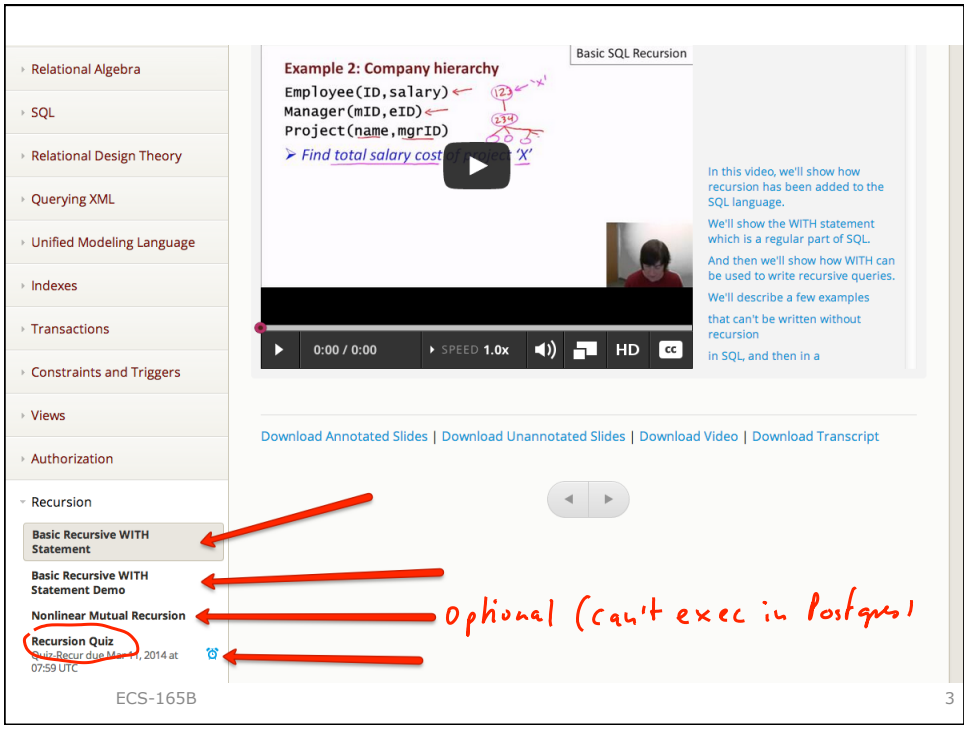
- There was a discussion this morning
- Instructor's Office Hours moved **this** week
  - Tuesday (8:30am) ➔ Thursday (10am)

- Wrapping up Recursion
- Starting up XML processing!

Wrapping Up Recursion

- Datalog "rules" when it comes to recursion
- … but you should also know how to do it in SQL!
- One more time: Check out **DB-class.org**!
  - Specifically check out the material on recursion!
  - If you haven't attended the discussion and/or still have questions about recursion, make sure you have …
  - 1. … watched the videos
  - 2. … understood the examples
  - 3. … done the online quiz!

```
WEEK 2
======
* M 1/13
- Slide Handouts: [165B-04.pdf]
* W 1/15
- 9am Lecture      [165B-05.pdf]
- 3:10pm Lecture   [165B-06.pdf]
* F 1/17 CANCELLED
- Readings: Chapters on Datalog/Deductive Databases: [GMUW09, Ch 5.3-5.6], [AHV95, Ch 12-13]
- Additional Reading: Datalog @ Wikipedia, Chapters 12-13 on Datalog (Alice Book)
- Further Material on Recursion: Visit DB-class.org, and do the recursion module;
  see also the recursion folder.
```



XML Data

Querying Semistructured Data

Semistructured Data

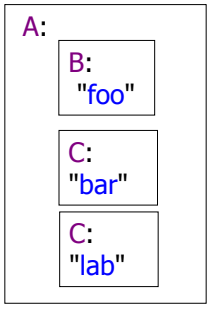
- Relational Data Model
  - Very mature theory, engineering, & practice
  - PODS, ICDT, SIGMOD, VLDB, ICDE, EDBT, …
  - IBM DB2, Oracle, MS SQL Server, Postgres, …
  - De facto standard in industry (SAP, …)
- But also need (e.g. "web data") for:
  - Less structured ("semistructured") data
  - Nested data collections
- Main trends:
  - XML, JSON, (… YAML, …)
  - NoSQL

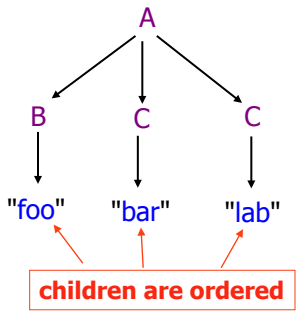
XML data ~ labeled, ordered trees

- XML data model:
  - nested containers ("boxes within boxes")
  - labeled ordered trees (=a semistructured data model)
  - relational, object-oriented, other data: easy to encode

XML Syntax

```
<A>
  <B>foo</B>
  <C>bar</C>
  <C>lab</C>
</A>
```



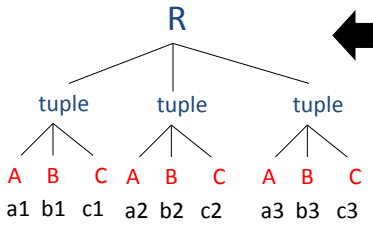


Relational Structures vs. XML

R → tuple*
tuple → A, B, C?

R

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3




```
<R>          <relation name = "R">
  <tuple>
    <A> a1 </A>
    <B> b1 </B>
    <C> c1 </C>
  </tuple>
  <tuple>
    <A> a2 </A>
    <B> b2 </B>
    <C> c2 </C>
  </tuple>
  ...
</R>         </relation>
```

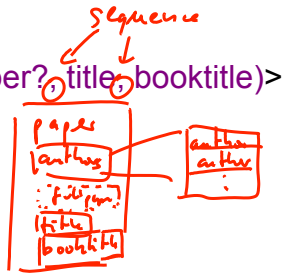
Relational Model vs XML

	Relational	XML
Structure	Tables	Hierarchical/labeled Tree, graph/ ordered trees
Schema	Fixed in advance	Flexible "self-describing" (part of the schema embedded)
Queries	Simple, nice langs. ☺	Less so. 😐
Ordering	None.	Implied.
Implementation	Native.	Add-on.

Source: DB-class.org (Jennifer Widom, Stanford)

XML Document Type Descriptions as Grammars

XML DTD (SGML heritage shows)




```
<!element bibliography paper*>
<!element paper        (authors, fullPaper?, title, booktitle)>
<!element authors      author+>
<!element author       #PCDATA>
```

Grammar

bibliography ⟶ paper*
paper ⟶ authors, fullPaper?, title, booktitle
authors ⟶ author+
author ⟶ string

A → (B | C)*
A → (B, C)*

lhs = *element (name)*
rhs = ***regular expression*** *over elements + strings (PCDATA)*

DTDs

- Declaring the Cardinality of Elements
  - [none] - element must appear exactly once. Default cardinality whenever no cardinality indicator is used
  - ? - **optional**, element may appear once or not at all
  - + - element may appear **one or more** times
  - * - element may appear **zero or more** times

A → B*, C*
"or"
A → (B | C)*

- Specifying Sequences and Choices
  - In addition to cardinality, the DTD uses a notation to specify **sequences** and **choices**
    - A , B (comma) - Both A and B occur, in that order
    - A | B (vertical bar) - either A or B occur, but not both
    - ( ) (paragraphs) are used for grouping and may be nested.

Validation: XML Instance and DTD (Schema)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- recordings.xml by Ed Gellenbeck -->
<!DOCTYPE Recordings SYSTEM "recordings.dtd">
<Recordings>
 <Compactdisc>
  <Artist type="individual">Van Morrison</Artist>
  <Title numberOfTracks="4">Too Long in Exile</Title>
  <Tracks>
    <Track>Big Time Operators</Track>
    <Track>Gloria</Track>
    <Track>Close Enough For Jazz</Track>
    <Track>I'll Take Care Of You</Track>
  </Tracks>
  <Price>$12.99</Price>
 </Compactdisc>
 <Compactdisc>
...
 </Compactdisc>
</Recordings>
```

recordings.xml

```
<!ELEMENT Recordings ( Compactdisc* ) >
<!ELEMENT Compactdisc ( Artist, Title, Tracks, Price? ) >
<!ATTLIST Artist type CDATA #REQUIRED >
<!ELEMENT Artist ( #PCDATA ) >
<!ATTLIST Title numberOfTracks CDATA #REQUIRED >
<!ELEMENT Title ( #PCDATA ) >
<!ELEMENT Tracks ( Track+ ) >
<!ELEMENT Track ( #PCDATA ) >
<!ELEMENT Price ( #PCDATA ) >
```

recordings.dtd

Data Modeling with DTDs

- XML element types ~ "object types"
- content model for children elements ~ "subobject structure"
- recursive types (container analogy!?)
  - <!ELEMENT A (B|C)> *"an A can contain a B..."*
  - <!ELEMENT B (A|C)> *"... which contains an A!"*
  - <!ELEMENT C (#PCDATA)>
  - – found in doc world: document **DIV**ision (=generic block-level container)
- loose typing
  - – <!ELEMENT A ANY> *"so what's in the box, please??"*
- **no** context-sensitive types:

  DTDs cannot distinguish between the publisher in
  - – <journal> <publisher>... </publisher> </journal>
  - – <website> <publisher> ... </publisher> </website>

  => renaming "hack" <j_pub> and <w_pub> (or use a "union type")

  => DTD extensions (**XML SCHEMA**)

Where is the Data??

- Actual data can go into leaf elements and/or attributes
- Common/good practice (!?):
  - XML element ~ container (object)
  - XML element type (tag) ~ container (object) type
  - XML attribute ~ properties of the container as a whole ("metadata")
  - XML leaf elements ~ contain actual data
- Problems with DTDs:
  - no data types
  - no specialization/extension of types
  - no "higher level" modeling (classes, relationships, constraints, …)

Processing XML: DOM

- The XML DOM is:
  - A standard object model for XML
  - A standard programming interface for XML
  - Platform- and language-independent
  - A W3C standard
- The XML DOM
  - defines the objects and properties of all XML elements, and the methods (interface) to access them
  - is a standard for how to get, change, add, or delete XML elements
- XML DOM Tutorial:
  - http://www.w3schools.com/dom

XML DOM Model

- Node Parents, Children, and Siblings
  - The nodes in the node tree have a **hierarchical** relationship to each other.
  - The terms **parent**, **child**, and **sibling** are used to describe the relationships.
    - Parent nodes have children.
    - Children on the same level are called siblings (brothers or sisters).
  - In a node tree, the top node is called the **root**
  - Every node, except the root, has **exactly one parent** node
  - A node can have **any number of children**
  - A **leaf** is a node with no children
  - **Siblings** are nodes with the same parent
- For the CS theoretician:
  - XML ~ labeled, ordered trees