**XQuery Class Outline**

* Introduction and goals of the class
  + Show what XQuery is and what it does
  + Get class to write a simple XQuery script
  + Give class a starting point for later exploration
* What is XQuery? How is it used?
  + W3C standard
  + Designed for users without formal programming background
  + Designed to extract, transform, and manipulate XML data
  + mySQL for XML data
* Introduction to BaseX GUI
  + Example of a successful query **[use EADsummary.xq]**
  + Example of a failed query and how to find errors
  + Saving and loading scripts and documents
* Structure of and XQuery script and basic rules
  + Prolog and Body
  + Odd punctuation
  + Variables
  + FLOWR expressions
  + **Detail what FOR, etc. does**
  + XPATH in XQuery, @attributes
  + Operators
  + IF expressions
  + Functions
  + Formatting results in XML/HTML
* Knowing your XML source
  + introduce league.xml
* Get class to write a basic XQuery script (class now follows along)
  + Simple For/Return:

**for $player in doc("national.xml")/LEAGUE/PLAYER**

**return $player**

* + $Variables
  + Editing and undo (CTRL+Z and CTRL+Y)
  + XPath use (/ and // and ../)
  + Declare variable in prolog:

**declare variable $xml := doc (‘national.xml');**

* + Let

**let $qualify := 500**

* + Order by
    - Return player names in order of AB

**order by $player/AB descending**

* + - difference between integers and strings

**order by $player/NAME**

* + - Who has the highest AVG?
    - Which players led the league in SO?

**order by number($player/SO) descending**

* + Where
    - Who has the highest AVG with over 500 AB?
    - Return players with over 500 AB

**where $player/AB > 500**

**where $player/AB >= $qualify**

* + - use of operators
  + Comments  **(: comments :)**
    - Comment out qualifying for the batting title
  + Functions
    - How many players play in the league?

**let $number := count($xml/LEAGUE/PLAYER)**

**return $number**

* + - How many home runs were his last season?

**let $number := sum($xml/LEAGUE/PLAYER)**

**return $number**

* + IF expression
    - Return the names of players who hit more than 25 home runs

**for $player in doc("national.xml")/LEAGUE/PLAYER**

**return**

**if ($player/HR >= 25)**

**then $player/NAME**

**else ()**

* + Formatting results in XML
    - Return a XML file of the batting title leaders with the batters name and average
    - Need a root node just like an XML file

**for $player in doc("national.xml")/LEAGUE/PLAYER**

**let $qualify := 500**

**where $player/AB > $qualify**

**order by number($player/AVG) descending**

**return**

**<batter>**

**<name>{$player/NAME}</name>**

**<average>{$player/AVG}</average>**

**</batter>**

* return only the data not nodes

**<name>{data($player/NAME)}</name>**

**<average>{data($player/AVG)}</average>**

* + - Return with one element per batter

**<batter avg="{data($player/AVG)}">{data($player/NAME)}</batter>**

**}</leaders>**

* + IF expression within FLOWR
    - Return a list of AVG leaders but disqualify players who hit 25 or more HR

**<leaders>{**

**for $player in doc("national.xml")/LEAGUE/PLAYER**

**let $qualify := 500**

**where $player/AB > $qualify**

**order by number($player/AVG) descending**

**return**

**if ($player/HR >= 25)**

**then <batter avg="DISQUALIFIED">{data($player/NAME)}</batter>**

**else <batter avg="{data($player/AVG)}">{data($player/NAME)}</batter>**

**}</leaders>**

* + - Let’s not show the disqualified players at all ()
  + Return an HTML table of qualified leaders(AB<500) that lists: NAME, AVG, HR, SO/BB, WAR sort by SO/BB

**xquery version "3.0";**

**declare variable $xml := collection('baseball');**

**<html>**

**<head>**

**<link rel="stylesheet" type="text/css" href="table.css"/>**

**</head>**

**<body>**

**<table>**

**<tr>**

**<th>Name</th>**

**<th>AVG</th>**

**<th>HR</th>**

**<th>K/BB</th>**

**<th>WAR</th>**

**</tr>**

**{**

**for $player in $xml/LEAGUE/PLAYER**

**let $qualify := 100**

**where $player/AB > $qualify**

**let $k := $player/SO**

**let $walk := $player/BB**

**let $ktowalk := $walk div $k**

**order by $ktowalk descending**

**return**

**<tr>**

**<td>{data($player/NAME)}</td>**

**<td>{data($player/AVG)}</td>**

**<td>{data($player/HR)}</td>**

**<td>{data($ktowalk)}</td>**

**<td>{data($player/WAR)}</td>**

**</tr>**

**}</table>**

**</body>**

**</html>**

* + - reduce the K/BB ratio to two decimal places
* What XQuery can teach you about XML
  + XML is very flexible
  + Hard to predict how data will be used until you use it
  + Breaks document-centric thinking
  + Query and manipulate not reformat
  + Further separate data storage and display
* In-Class exercises:
  + Hardest: use the EAD files in the EAD folder to make a HTML table of collections, listing collection level information: title, unitdate, extent, and author
  + Medium: From the baseball collection, return a HTML table listing player name, team, hits, RBIs, and WAR, sorted by hits
  + Easier: from the baseball collection, return a basic XML file that lists the name, team, and RBIs of each player that had over 90 RBIs