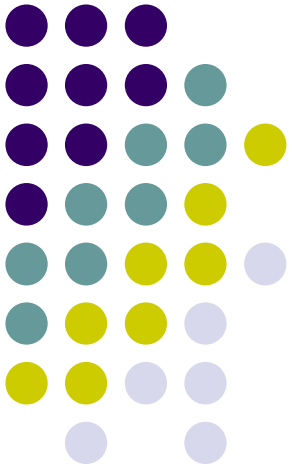
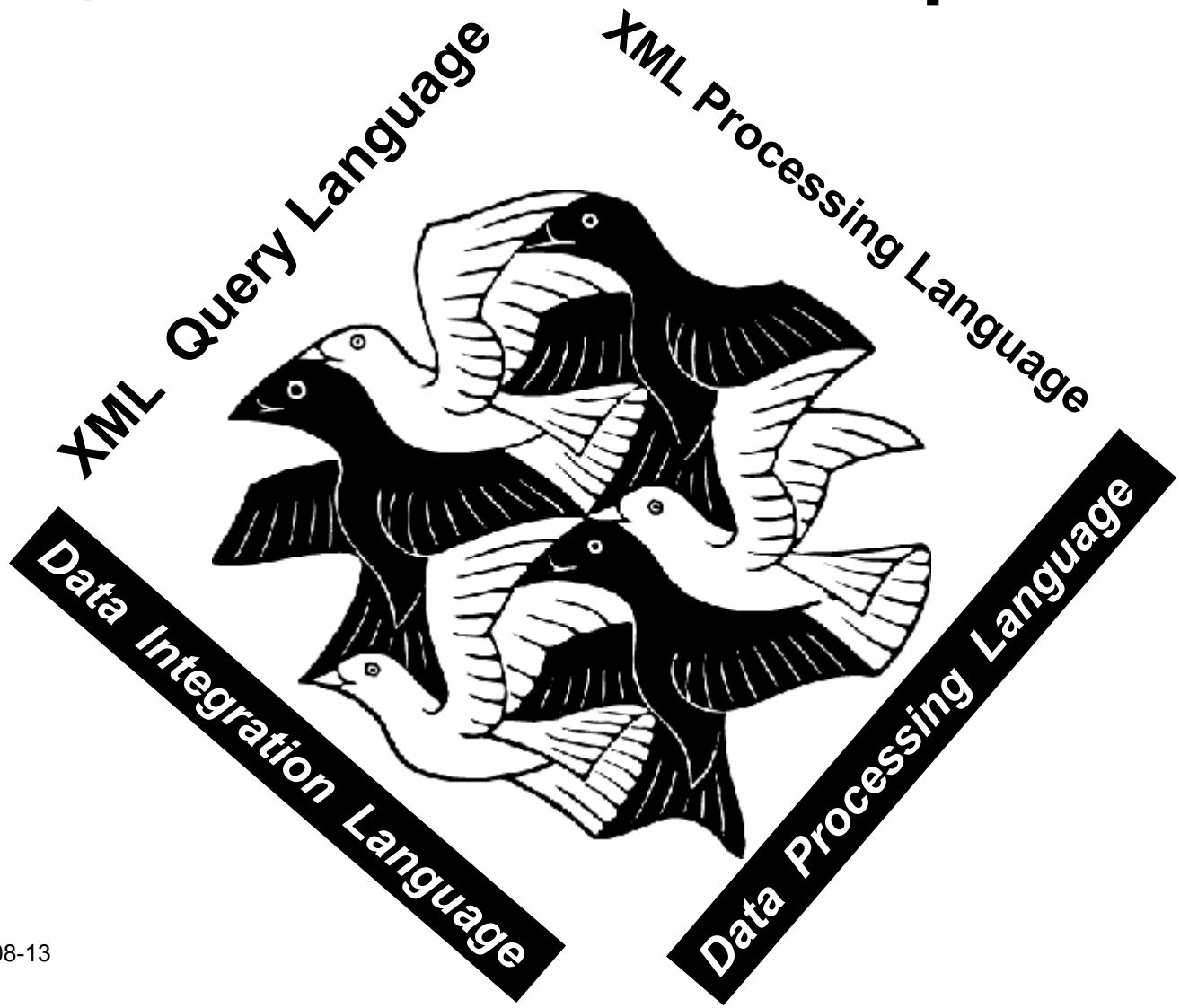
XQuery as a *data integration language*

Assessing the potential.



Hans-Jürgen Rennau, Traveltainment GmbH
Christian Grün, BaseX GmbH
Presented at Balisage 2015, August 13, 2015

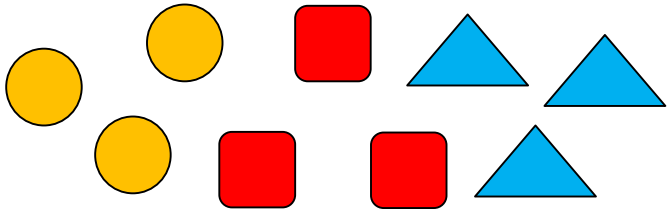
xquerY is WHAT is Xquery



Data integration language ?

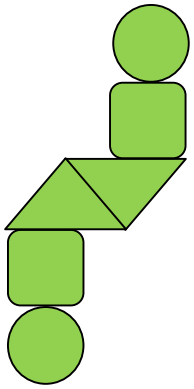
- Integration – facing

  multiplicity and heterogeneity



- Data integration language:

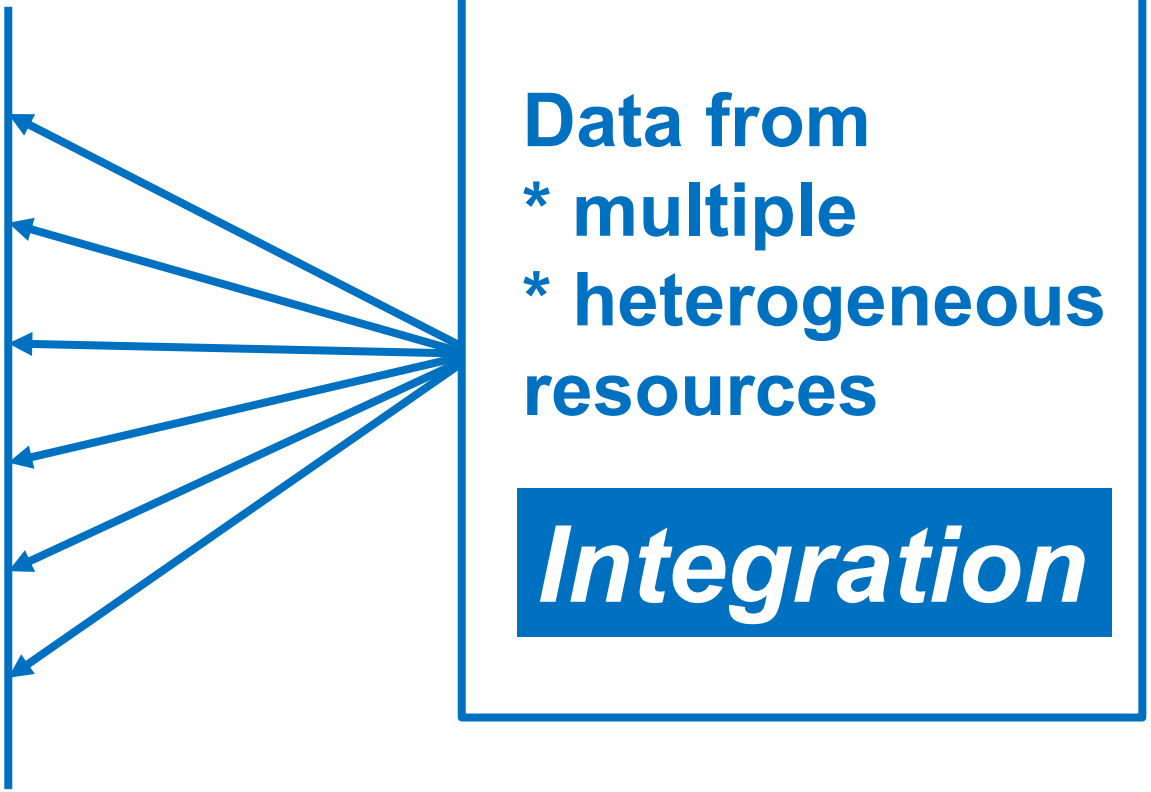
  making integration *simpler*



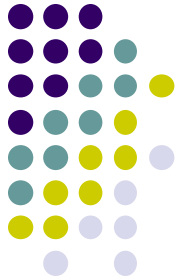
Integration – core operations

Data ...

- Selection
- Construction
- Modification
- Transformation
- Exploration
- Validation



Divide and conquer



How to assess the integration capabilities?

Investigation strategy

- XML integration capabilities

  (capabilities in an XML-only environment)
- Access to non-XML resources
- XML integration capabilities applicable to non-XML?

Data navigation

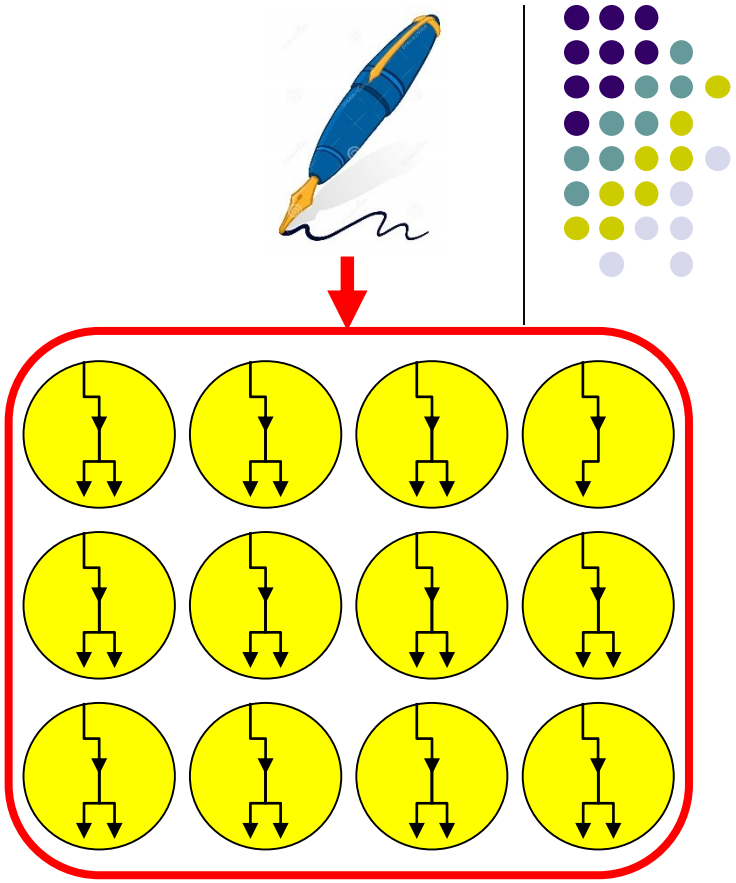
Precision
Expressiveness
Naturalness

```
//@flightNumber
```

```
//flights
/flight
/@flightNumber
```

```
//flights
/flight [arrival   /@airport = 'IAD']
        [departure/@airport = doc('/usr/data/airports.xml')
                              //airport[@cty!='US']/@code]
/@flightNumber
```

Bulk navigation

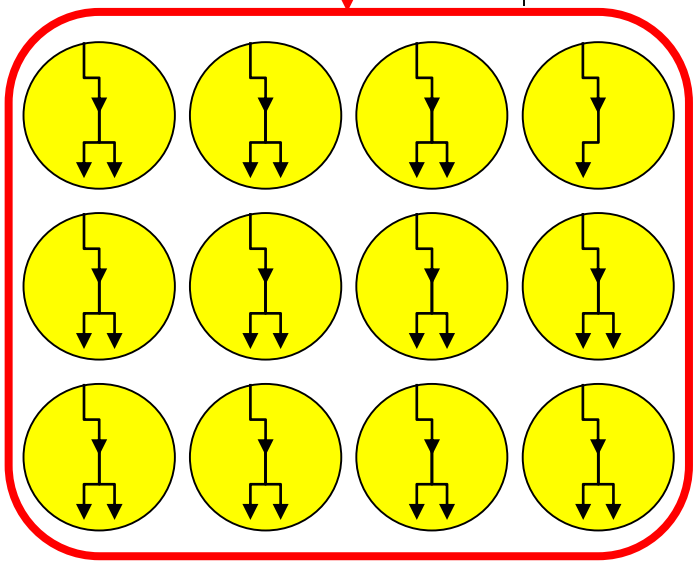


```
//@flightNumber
```

```
//flights
/flight
/@flightNumber
```

```
//flights
/flight [arrival   /@airport = 'IAD']
        [departure/@airport = doc('/usr/data/airports.xml')
                              //airport[@cty!='US']/@code]
/@flightNumber
```

Bulk navigation



```
(doc('f1.xml'), doc('f2.xml'))
//@flightNumber
```

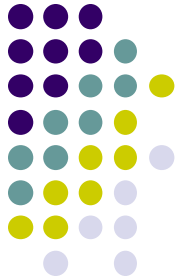
document pump

```
doc('airports.xml') //@href /doc(.)
//flights
/flight
/@flightNumber
```

document pump

```
file:list($dir, true(), 'f*.xml') !concat($dir,'/', .) !doc(.)
//flights
/flight [arrival   /@airport = 'IAD']
        [departure/@airport = doc('/usr/data/airports.xml')
                              //airport[@cty!='US']/@code]
/@flightNumber
```

Document pump – a pattern



Original: PATH

Bulk version: UFACTORY ! doc(.) ! PATH

Some expression emitting URIs

UFACTORY „name list“

```
unparsed-text-lines('doclist.txt')
```

UFACTORY „file list“

```
file:list($dir, true(),'*.xml')!concat($dir,'/', .)
```

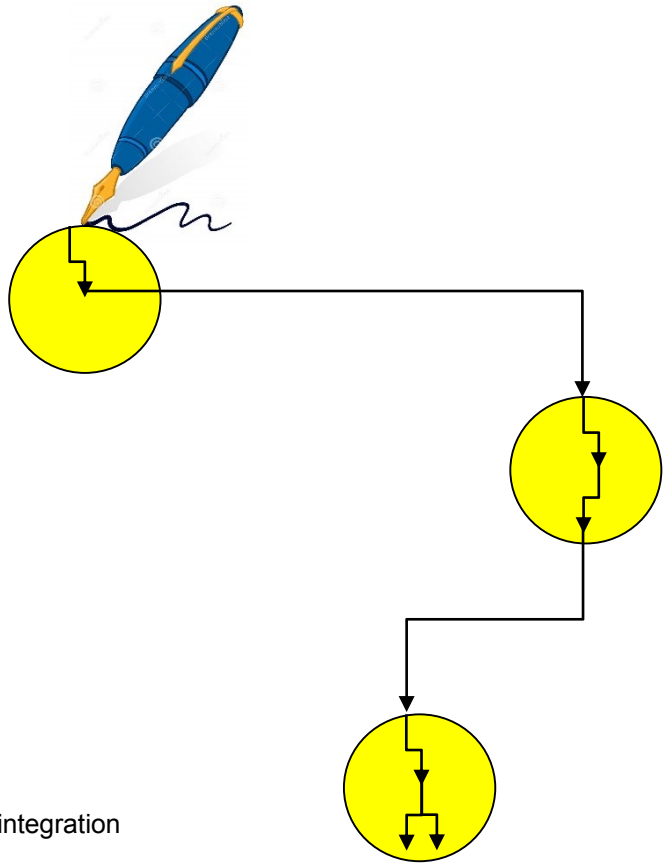
UFACTORY „document catalog“

```
doc('mycatalog.xml')//@href
```

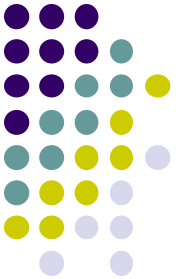
Cross-document navigation

Navigation can cross document boundaries any number of times.

```
//flights
/flight
/termsAndConditions
/@URL
/doc(.)
//payments
/ccProvider
/@href
/doc(.)
//contact
```



Boundless navigation



In summary, XPath navigation is *boundless*:

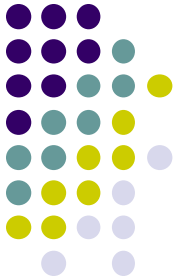
- Can have starting / intermediate / end points distributed over several documents
- Can cross document boundaries at any time

The navigation model fuses all accessible documents into a single space of information

=

info space

Construction ...



Data integration requires data construction – rearranging bits of distributed information

Data construction powered by navigation:

- Embedded expressions
- Update facility – combines navigation / modification

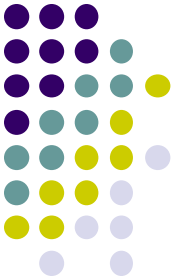
... powered by navigation

```
<routes>

   <national>{
      //flight[not(addInfo/@international)]
      /<route>{@*, * except addInfo}</route>
   }</national>

   <international>{
      //flight[addInfo/@international]
      /<route>{@*, * except addInfo}</route>
   }</international>

</routes>
```

NAV#1 => data source

NAV#2 => new contents

Modification

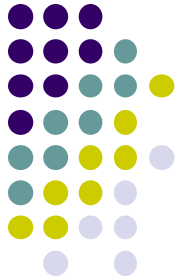


- Special case of construction: modification
- XQuery Update Facility
- Modification powered by navigation

Update Expressions:

- Sub expression #1: *selects* the target node
- Sub expression #2: *changes* the target node

Bulk modification



document pump

```
doc('airlines.xml') //flights/@href /doc(.)
```

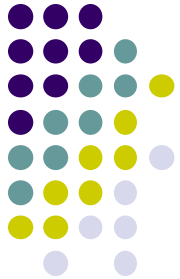
navigator

```
//flights
/flight [arrival   /@airport = 'IAD']
        [departure/@airport = doc('/usr/data/airports.xml')
                              //airport[@cty!='US']/@code]
```

```
/(insert node attribute international {true()} into addInfo)
```

modifier

Resource exploration



age
. 10
. 17
. 18
. 19
. 20

antigenSequence
. NSFKNNYEKALKQYNSTGDYRSHAVDKIQNTLHCCGVTDYRDWTD ...

cellLine
. A-431
. A549
. AN3-CA
. BEWO
. CACO-2

cellType
. Langerhans
. Leydig cells
. Purkinje cells
. adipocytes
. bile duct cells

Example output:
names and values of simple-content elements

Resource exploration

```
declare variable $limit as xs:int external := 5;

for $elem in //*[not(*)][text()]
group by $ename := local-name($elem)
let $values :=
  sort(distinct-values($elem))
  [position() le $limit]
order by $ename
return
  ($ename, $values!concat('. ',.), '')
)
```

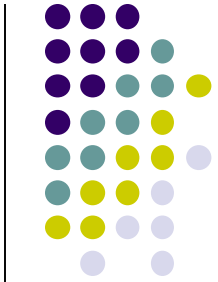
Collect all data elements

... group them by name

... extract values

Example code:
names and values of simple-content elements

Validation



- Data integration often requires data validation:
  - Grammar-based – e.g. XSD validation
  - Rule-based – e.g. Schematron
- Presently, only extension functions available

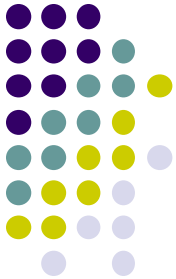
- XQuery assets:
  - Bulk validation
  - Expression-based validation

Bulk validation

```
let $xsds := UFACTORY1 ! doc(.)
let $docs := UFACTORY2 ! doc(.)
let $reports :=
  for $doc in $docs
  let $name := $doc/local-name(*)
  let $namespace := $doc/namespace-uri(*)
  let $xsd :=
    $xsds[$name      = */xs:element/@name]
         [$namespace = string(*/@targetNamespace)]
  let $msgs := validate:xsd-info($doc, $xsd)
  return
    <report doc="{document-uri($doc)}">{
         $msgs ! <msg>{.}</msg>
    }</report>
return
  <reports>{$reports}</reports>
```

Simple bulk navigation picks the appropriate XSD

Expression-based validation



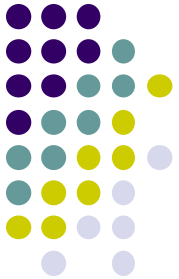
```
<rules>
   <rule msg="No product found."
         expr="empty(//*:Product)"/>
   <rule msg="Travel start date &gt; end date."
         expr="//travel[@start gt @end]"/>
</rules>
```

Rule descriptors (expression + message)

```
<report>{
  let $docs := UFACTORY ! doc(.)
  for $doc in $docs return
    <errors uri="{document-uri($doc)}">{
        doc('rules.xml')
        //rule[xquery:eval(@expr, map{'':$doc})]
        /<error msg="{@msg}"/>
    }</errors>
}</report>
```

Error list produced by a navigation expression !

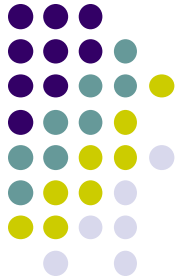
XML data integration



XQuery excellently suited

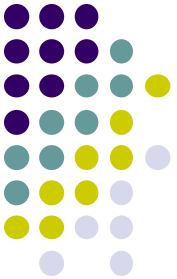
- Language foundation: boundless navigation
- Data selection – implemented by navigation
- Data construction - powered by navigation
- Data modification - powered by navigation
- Transformation – supported by navigation
- Resource exploration supported by navigation
- Resource validation supported by navigation

XQuery - the Great Transition



- Proved: XQuery suitable for XML integration
- Q: suitable for data integration in general?
  - XQuery 1.0: only XML
  - XQuery 3.0: add plain text resources
  - XQuery 3.1: add JSON
  - EXPath, vendor-specific extension functions:
    HTML, CSV, SQL, HTTP POST, archives, RDF

Resource access



- Resource representation = XDM value

  Examples: node, map, array, string

- Resource access: an XQuery function

  returning a resource representation

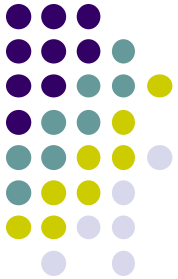
  Examples:

  `doc(URI)`

  `unparsed-text(URI)`

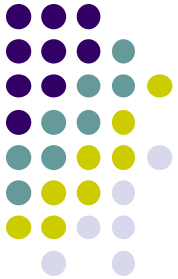
  `json-doc(URI)`

Resource access patterns



- Input (resource identification)
  - Access by URI `doc (URI)`
  - Access by text `parse-json (string)`
  - Access by message `http:send-request (msg)`
  - Access by query `sql:execute (query)`
- Output (resource representation)
  - XML node tree `json-to-xml (string)`
  - map/array tree `parse-json (string)`
  - string `unparsed-text (URI)`

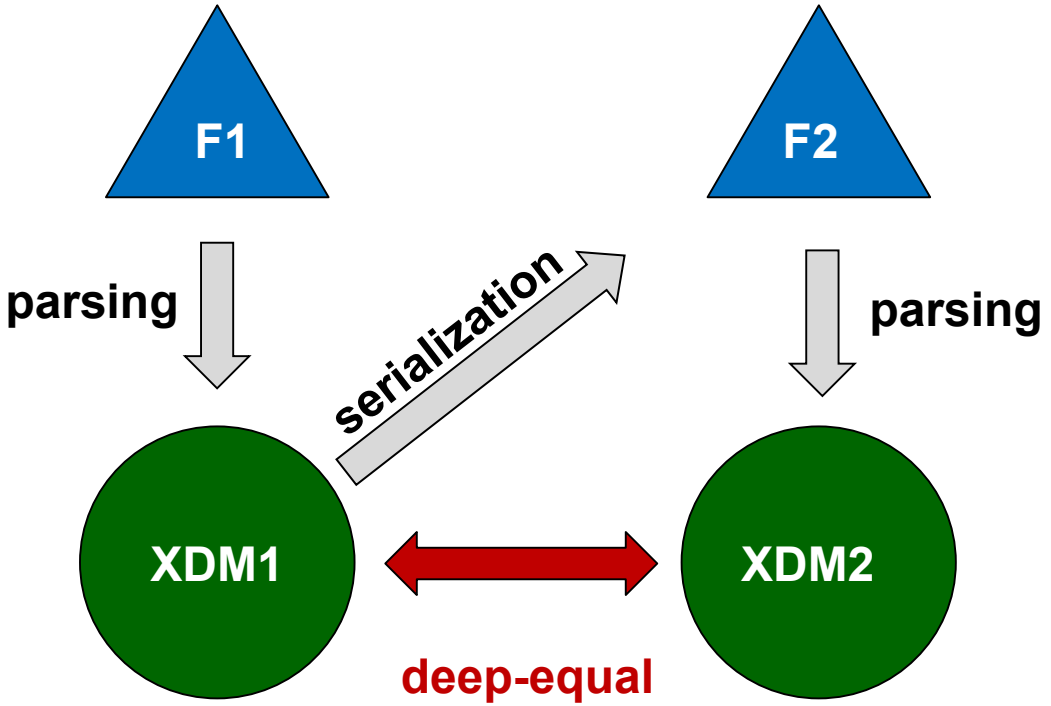
Resource representation - formally



DEFINITION. XDM binding

- **Parsing function P**
- **Serialization function S**
- **Constraint:**

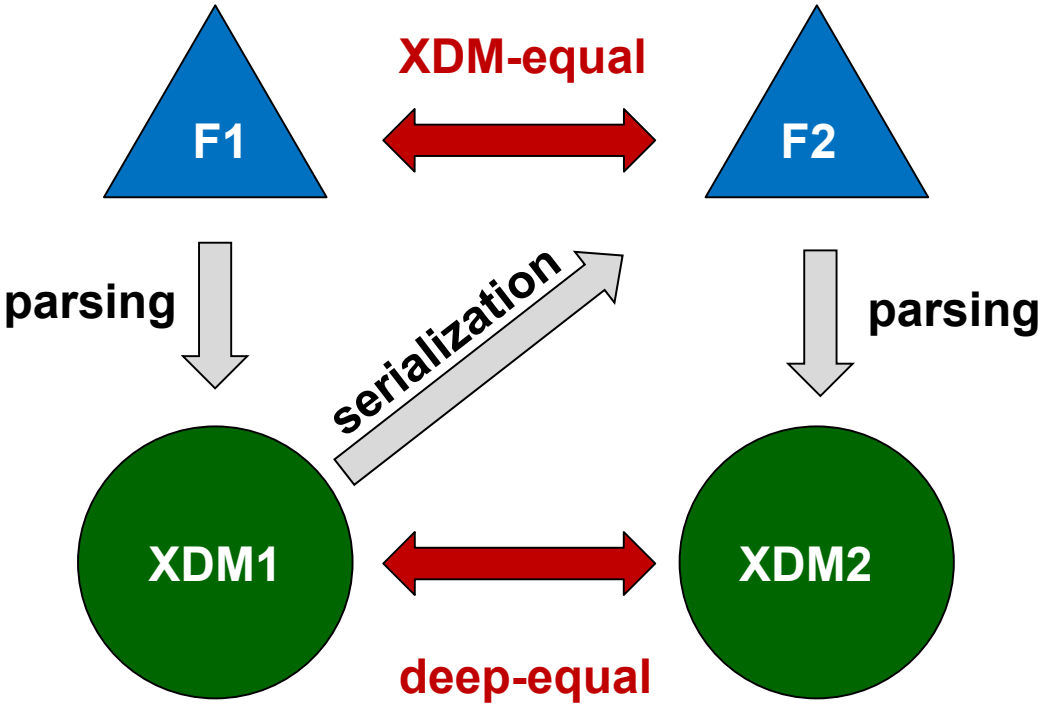
deep-equal(
P(F), P(S(P(F)))
)



Resource representation - formally

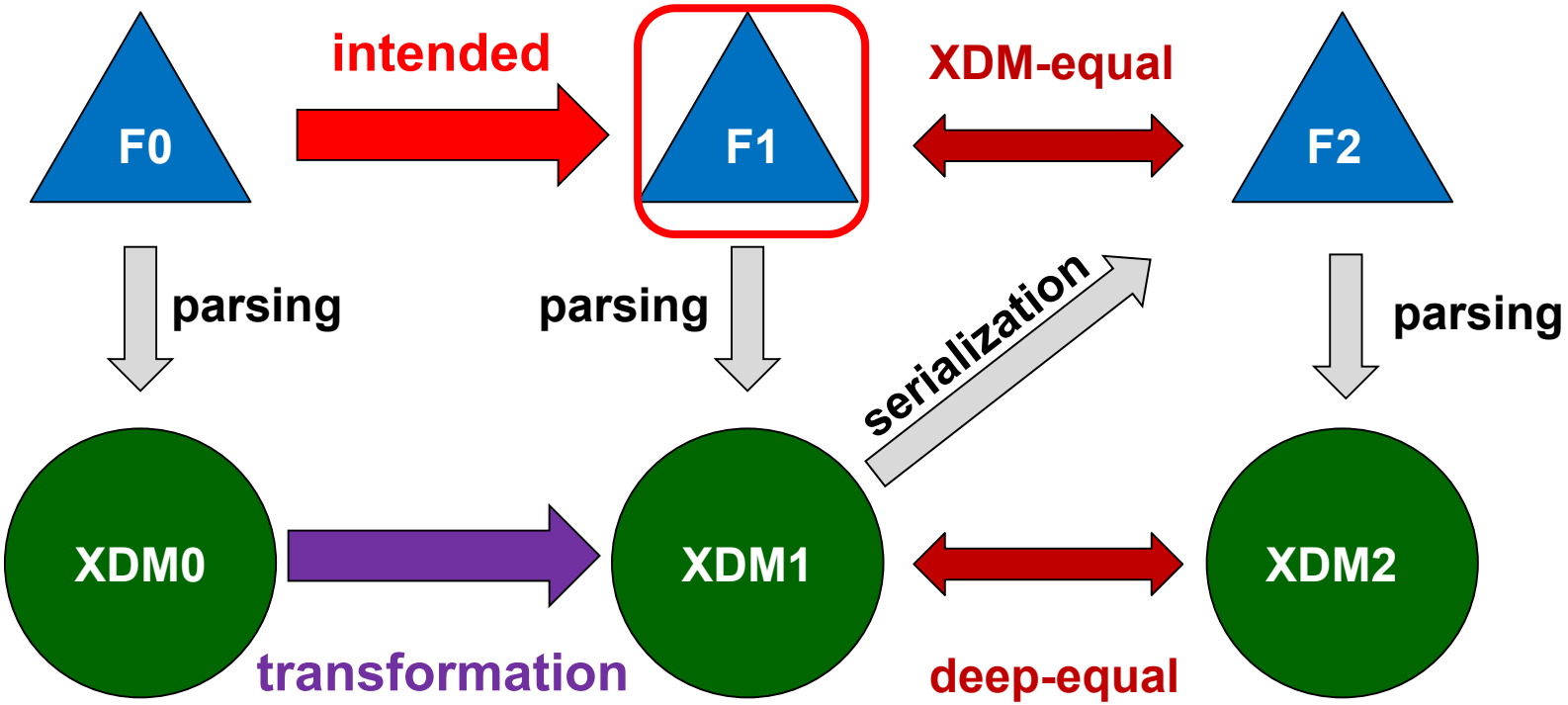
DEFINITION. XDM equal

Two format instances
F1, F2 are XDM equal
⇔
deep-equal(P(F1), P(F2))



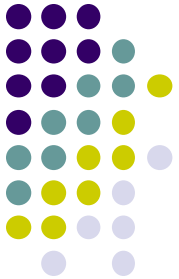
Non-XML transformation

format instances



XDM values

XML binding



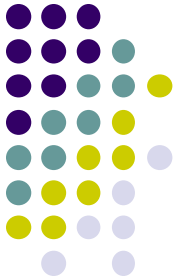
[DEFINITION. XML binding]

An XDM binding whose parsing function returns an **XML node tree**.

Nota bene.

XQuery‘s navigation capabilities apply to XML node trees – not to map/array trees.

XDM binding - examples



- Format: JSON

- Three XDM bindings:
  - (1) json#parse (XDM=XML) (BaseX)
  - (2) fn#json-to-xml (XDM=XML) (standard)
  - (3) fn#json-doc (XDM=map/array) (standard)

Bindings identified by a URI (namespace#name of parsing function)

- Q: How to navigate à la: `//booking/bookingID`

Navigation - XML binding

- Case 1: json:parse (XDM=XML (BaseX))

```
//booking/(., _)/bookingID
```

- Case 2: fn:json-to-xml (XDM=XML (W3C))

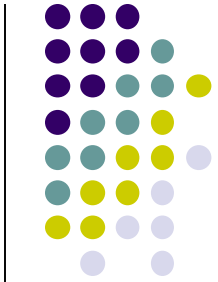
```
//*[@key eq 'booking']
/descendant-or-self::j:map[1]
/*[@key eq 'bookingID']
```

Navigation - non-XML binding

- Case 3: fn:json-doc (XDM=map/array)

```
declare function local:descendant($item as item(), $fieldName as xs:string) {
    typeswitch($item)
    case map(*) return (
      let $field := $item($fieldName)
      return
        if ($field instance of array(*)) then $field?* else $field,
      map:keys($item)[. ne $fieldName] !
        local:descendant($item(.), $fieldName)
    )
    case array(*) return $item?* ! local:descendant(., $fieldName)
    default return ()
};
local:descendant($data, 'booking')?bookingID
```

Map/array navigation issues



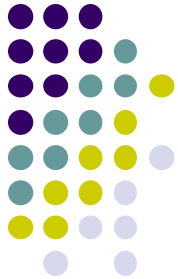
Primitive navigation: `a?b?c`

- Only child axis – no deep, no upward navigation
  - No deep navigation — no `//foo`
  - No upward navigation — no `ancestor::booking`
- Result heterogeneous (map, array, string, ...)
- Result anonymous — no `$res/name()`
- Knowledge of array-steps required

```
flights?*?departure?airport
```

- No modification — try `$j?a?b!map:put(.,'c','X')`

XML-binding ⇔ integration



- XML-binding available

  => format within scope of XPath navigation

  => within scope of integration capabilities

- XML-binding not available

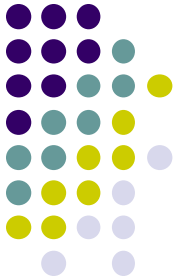
  => format not within scope of XPath navigation

  => *out of scope of integration capabilities ?*

  ... or conditionally within scope ?

White doves ? *or both ?* ***Black doves ?***

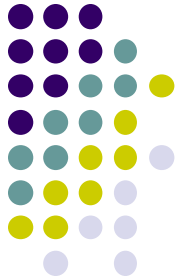
Wrapping up



- Everything within reach of XQuery‘s navigation capabilities is within reach of easy construction, modification, transformation and aggregation => within reach of integration.

- To consider: extend access to non-XML resources systematically (HTML, CSV, SQL, archives, ...). Make sure to support XML bindings whenever possible.

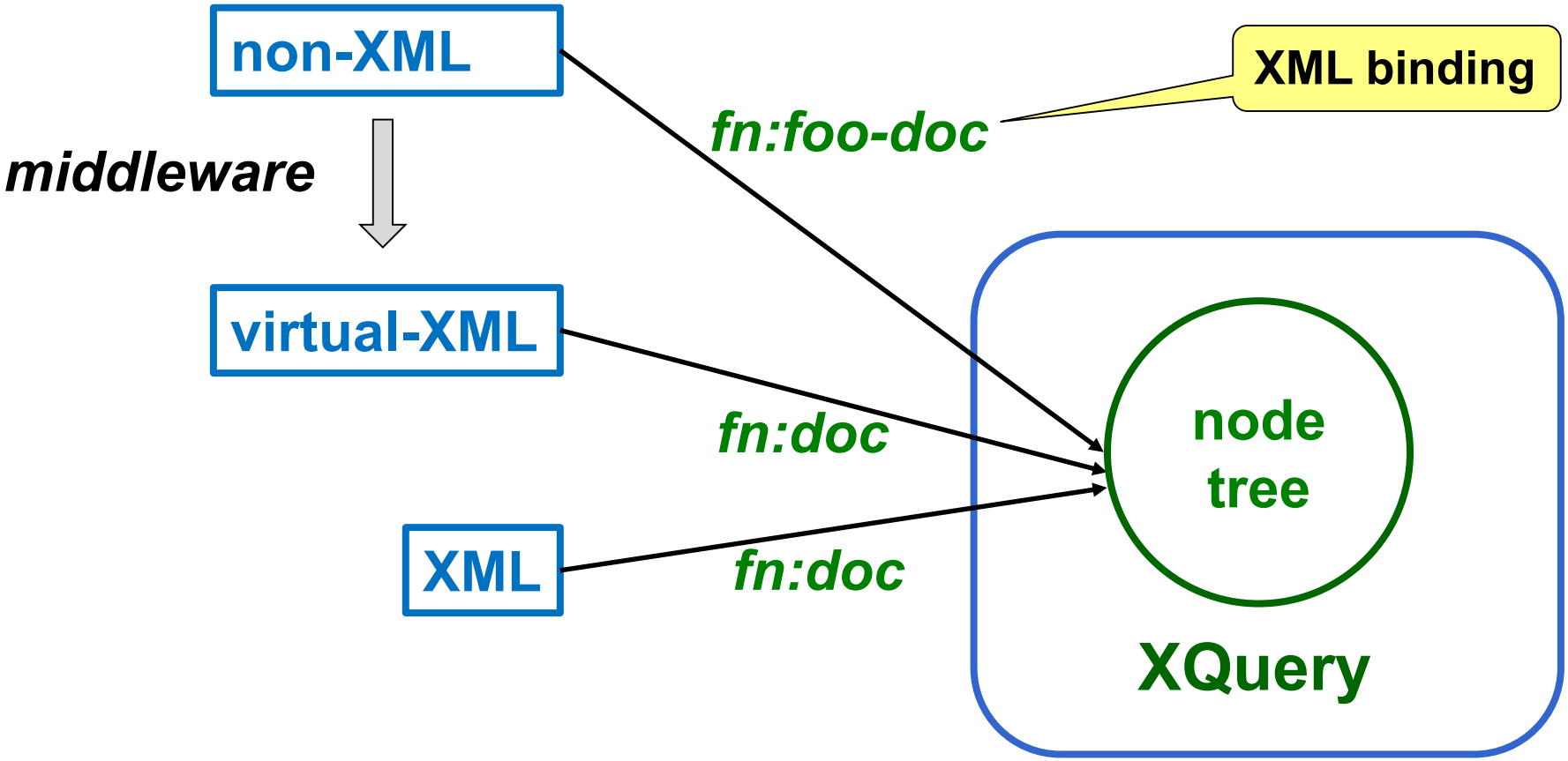
XQuery spec – first sentences



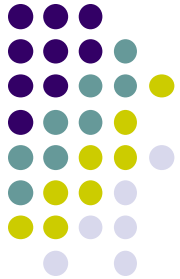
… A query language that uses the structure of XML intelligently can express queries across all these kinds of data, whether physically stored in XML or viewed as XML via middleware.

This specification describes a query language … designed to be broadly applicable across many types of XML data sources.

Integration via XML binding



A conceivable extension



… A query language that uses the structure of XML intelligently can express queries across all these kinds of data, whether physically stored in XML or viewed as XML via middleware, or bound to a node tree representation within XQuery itself.

This specification describes a query language … designed to be broadly applicable across many types of XML and XML-bindable data sources.

Thank you for looking!