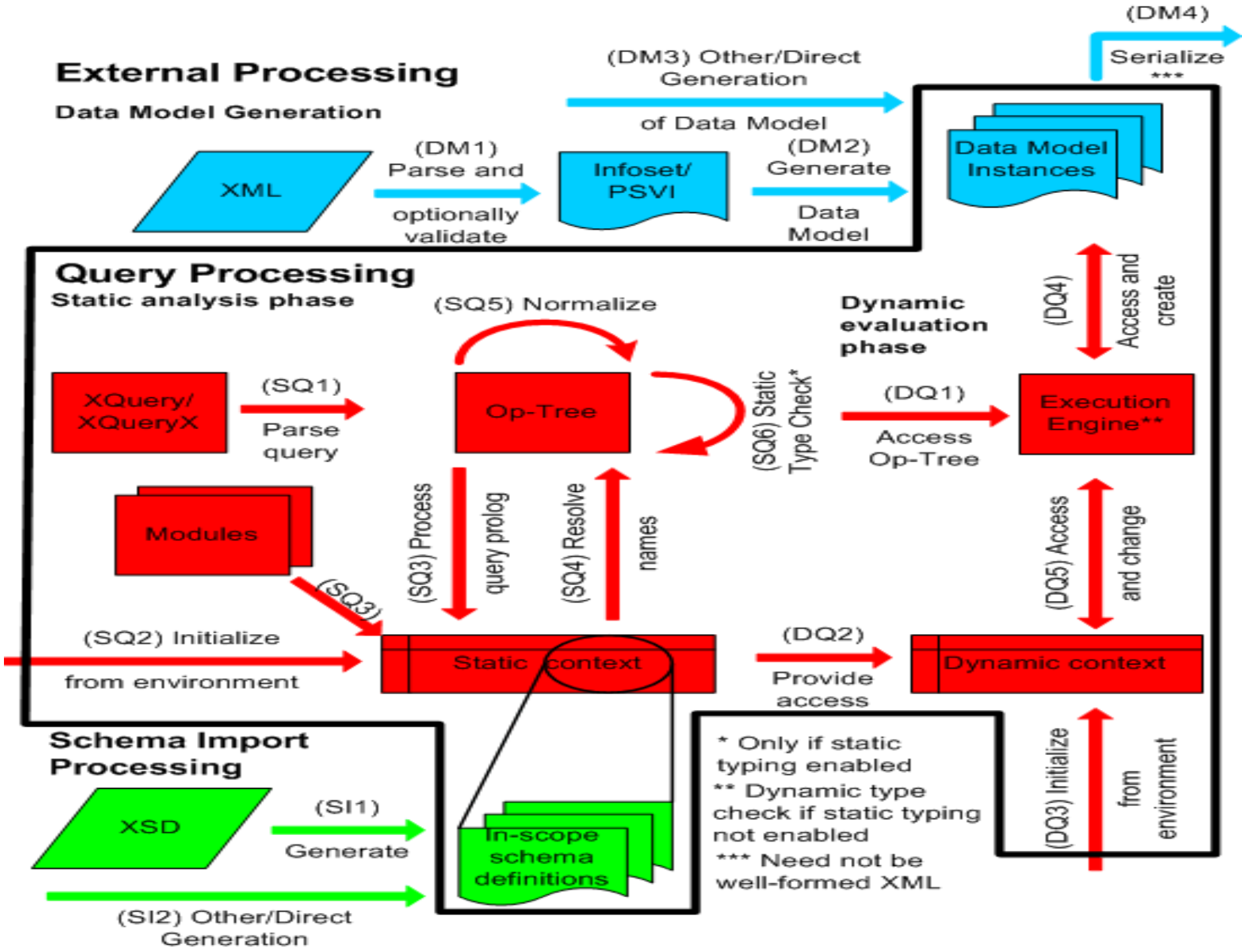
# XQuery

# XQuery specification

- http://www.w3.org/TR/xquery/
- Reference: Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, et al., Web Data Management, Cambridge University Press, 2011, http://webdam.inria.fr/Jorge/
- Or: XQuery from the Experts, A Guide to the W3C XML Query Language, by Don Chamberlin et others, Howard Katz Editor, pages 1-50


External Processing
Data Model Generation
XML
(DM1) Parse and optionally validate
Infoset/ PSVI
(DM2) Generate Data Model
(DM3) Other/Direct Generation of Data Model
Data Model Instances
(DM4) Serialize ***
Query Processing
Static analysis phase
XQuery/ XQueryX
(SQ1) Parse query
Op-Tree
(SQ5) Normalize
(SQ6) Static Type Check*
Dynamic evaluation phase
(DQ1) Access Op-Tree
Execution Engine**
(DQ4) Access and create
Modules
(SQ3)
(SQ3) Process query prolog
(SQ4) Resolve names
(DQ5) Access and change
(SQ2) Initialize from environment
Static context
(DQ2) Provide access
Dynamic context
Schema Import Processing
XSD
(SI1) Generate
In-scope schema definitions
(SI2) Other/Direct Generation
* Only if static typing enabled
** Dynamic type check if static typing not enabled
*** Need not be well-formed XML
(DQ3) Initialize from environment

# XQuery: For Where Return

- **for** $x **in** doc("books.xml")/bookstore/book
  **where** $x/price>30
  **return** $x/title
- doc("books.xml")/bookstore/book[./price > 30]/title

# FLWR

- **for** $auth **in** doc("books.xml")//author/text()
  **let** $books **:=**
  doc("books.xml")//book[./author=$auth][./year>2000]
  **where** fn:count($books) < 10
  **return** $auth

# Node construction

- **for** $auth **in** doc("books.xml")//author/text()
  **let** $books **:=** doc("books.xml")
  //book[./author=$auth][./year>2000]
  **where** fn:count($books) < 10
  **return** <author name="Dr. {$auth}">
  {$books}
  </author>
- The nodes returned by $books are cloned and inserted in the new element
- The new element is validated at run-time

# Nested loops in FLWR

- **for** $b **in** doc("books.xml")//book
  **for** $a **in** $b//author
  **return** <pair> { $b $a } </pair>
- <!DOCTYPE books
  **[** <!ELEMENT books (book*)>
  <!ELEMENT book (title author*)> …
  ]
  >
- books
  book*
  title
  author*
- pair*
  book
  author

# Nested loops in FLWR

- **for** $b **in** doc("books.xml")//book,
  $a **in** $b//author
  **return** <pair> { $b/title, $a } </pair>
- books
  book* (title  author*)
- pair*
  title
  author

# Nested loops in FLWR

- In: …

  book* (author*)
- Out: libro* (autore*)
- **for** $b **in** doc("books.xml")//book

  **return** <libro> {

  **for** $a **in** $b/author

  **return** <autore> { $a/text() } </autore>}

  </libro>

# Nested loops in FLWR

- Out:

  author*

    title*

- **for** $a **in** fn:distinct-values(doc("books.xml")//author)

  **return** <author> {

    <name> {$a} </name> ,

    **for** $b **in** doc("books.xml")//book

    **where** $b//author = $a

    **return** <title> {$b/title/text()} </title>}

  </author>

# Quantifiers

- **for** $auth **in** doc("books.xml")//author/text()
  **let** $books **:=** doc("books.xml")//book[./author=$auth]
  **where**
  **every** $book **in** $books
  **satisfies** $book/year > 2000
  **return** <author name="{$auth}">
  {$books}
  </author>
- Existential: some / in / satisfies
- **some**|**every** $var **in** expr (**,** $var **in** expr)*
  **satisfies** expr

# Dynamic typing

- Expr **instance of** Type
- **typeswitch** Expr
  **case** $x **as** Type **return** Expr
  **default** $x **return** Expr
- Expr **castable as** AtomicType
- Expr **cast as** AtomicType
- Expr **treat as** Type
- **validate** {Expr}

# Function declaration

- **declare function** local:summary
  ($emps **as** element(employee)*)
  **as** element(dept)*
  **{** **for** $d **in** fn:distinct-values($emps/deptno)
  **let** $e **:=** $emps**[**deptno **=** $d**]**
  return
  <dept>
  <deptno>**{**$d**}**</deptno>
  <headcount> **{**fn:count($e)**}** </headcount>
  <payroll> **{**fn:sum($e/salary)**}** </payroll>
  </dept>
  **};**