

XQuery: a programming language at the crossroads

- the impact on the compilation and optimization techniques

Daniela Florescu
Oracle Corporation

Plan of the talk

- Introduction on XML and XQuery
- The big picture of XQuery compilation/optimization
- XQuery logical rewritings (30 min)
- XML and XQuery in the Oracle products
- Open research problems in XQuery compilation

2

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

A little bit of history

- *Database world*
  - 1970 relational databases
  - 1990 nested relational model and object oriented databases
  - 1995 semi-structured databases
- *Documents world*
  - 1974 SGML (Structured Generalized Markup Language)
  - 1990 HTML (Hypertext Markup Language)
  - 1992 URL (Universal Resource Locator)

Data + documents = information

1996 **XML** (Extended Markup Language)
URI (Universal Resource Identifier)

3

What is XML?

- The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web.
- Base specifications:
  - XML 1.0, W3C Recommendation Feb '98
  - Namespaces, W3C Recommendation Jan '99

4

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XML Data Example

```
<book year="1967">
   <title>The politics of experience
   </title>
    <author>
        <firstname>Ronald</firstname>
        <lastname>Laing</lastname>
    </author>
</book>
```

- Syntax, no abstract model
- Documents, elements and attributes
- Tree-based, nested, hierarchically organized structure

5

Zooming in….

```
<book year="1967" xmlns:amz="www.amazon.com">
    <title>The politics of experience</title>
    <author>R.D. Laing</author>
   <! Is this the right author ? !>
    <amz:ref amz:isbn="1341-1444-555"/>
    <section>
        The great and true Amphibian, whose nature
  is disposed to…..
        <title>Persons and experience</title> Even
  facts become...
    </section>  …
</book>
```

- Qualified names
- Namespaces
- Mixed content
- Comments, Pis
- Character and entity references
- CDATA sections
- DTDs
- base-uris, xml:base, xml:lang, etc
- Schemas

6

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XML vs. relational data

- Relational data
  - Invented as a mathematically clean *abstract data model*
  - Philosophy: schema first, then data
  - Never had a standard syntax for data
  - Strict rules for data normalization, flat tables
  - Order is irrelevant, textual data supported but not primary goal
- XML
  - First killer application: publishing industry
  - Invented as a *syntax for data, o*nly later an abstract data model
  - Philosophy: data and schemas should not be correlated, data can exist with or without schema, or with multiple schemas
  - No data normalization, flexibility is a must, nesting is good
  - Order *may* be very important, textual data support a primary goal

7

Sources of XML data

1. Inter-application communication data (WS, Rest, etc)
2. Mobile devices communication data
3. Logs
4. Blogs (RSS)
5. Metadata (e.g. Schema, WSDL, XMP)
6. Presentation data (e.g. XHTML)
7. Documents (e.g. Word)
8. Views of other sources of data
   - Relational, LDAP, CSV, Excel, etc.
9. Sensor data
   - *It would be interesting to know the pie-chart and the evolution of each branch !*

8

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Some vertical application domains for XML

- *HealthCare Level Seven http://www.hl7.org/*
- *Geography Markup Language (GML)*
- *Systems Biology Markup Language (SBML) http://sbml.org/*
- *XBRL, the XML based Business Reporting standard http://www.xbrl.org/*
- *Global Justice XML Data Model (GJXDM) http://it.ojp.gov/jxdm*
- *ebXML http://www.ebxml.org/*
- *e.g. Encoded Archival Description Application http://lcweb.loc.gov/ead/*
- *Digital photography metadata XMP*
- *An XML grammar for sensor data (SensorML)*
- *Real Simple Syndication (RSS 2.0)*

Basically everywhere.

9

Processing XML data

- Huge amount of XML information, and growing
- We need to "*process*" it
  - Store it efficiently
  - Verify the correctness
  - *Filter, search, select, join, aggregate*
  - *Create new data*
  - *Update it*
  - Take actions based on the existing data
- No conceptual organization of the data in "databases" (XML applications are too heterogeneous)

10

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Why XQuery ?

- Why a "*query" language* for XML ?
  - Need to process XML data
  - Preserve logical/physical data independence
    - The semantics is described in terms of an *abstract data model*, independent of the physical data storage
  - Declarative programming
    - Such programs should describe the "*what*", not the "*how*"
- Why a *native* query language ? Why not SQL ?
  - We need to deal with the *specificities* of XML (hierarchical, ordered , textual, potentially schema-less structure)
- Why another XML processing language ? Why not XSLT?
  - The template nature of XSLT was not appealing to the database people. Not declarative enough.

11

What is XQuery ?

- A programming language that can express arbitrary XML to XML data transformations
  - Logical/physical data independence
  - "Declarative"
  - "High level"
  - "Side-effect free"
  - "Strongly typed" language
- "An expression language for XML."
- Commonalities with *functional* programming, *imperative* programming and *query* languages
- The "*query*" part might be a misnomer (***)

12

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Advantages of XML

- Schema flexibility
  - Data can exist with or without schemas, or with multiple schemas
  - Schemas can be added after the data
  - Schemas can open and/or complex
- Blurs the distinction between documents and structured data
  - Structured data, Excel spreadsheets, Word documents, RSS channels and XHTML are all XML
- Blurs the distinction between data, metadata and code
  - The data, the schemas and the code (I.e. XSLT and XQuery) are all XML

13

XML Data Model

Remember Lisp ?

- Instance of the data model:
  - a *sequence* composed of zero or more *items*
- The *empty sequence* often considered as the “null value”
- Items
  - *nodes* or *atomic values*
- Nodes

  document | element | attribute | text | namespaces | PI | comment
- Atomic values
  - Instances of all XML Schema atomic types

  `string, boolean, ID, IDREF, decimal, QName, URI, ...`
  - untyped atomic values
- Typed (I.e. schema validated) and untyped (I.e. non schema validated) nodes and values

14

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XML nodes

- 7 types of nodes:
  - document | element | attribute | text | namespaces | PI | comment
- Every node has a unique *node identifier*
  - Scope of node identifier uniqueness is implementation dependent
- Nodes preserve the dual aspect of information: lexical and binary
  - <price xs:type="xs:integer">0042</price>
- Nodes have children and an optional parent
  - conceptual "tree"
- Nodes are ordered based of the topological order in the tree ("document order")

Obvious impact of data storage and indexing

15

XQuery type system

- XQuery has a powerful (and complex!) type system
- XQuery types are imported from XML Schemas
- Every XML data model instance has a dynamic type
- Every XQuery expression has a static type
- Pessimistic static type inference
- The goal of the type system is:
  1. detect statically errors in the queries
  2. infer the type of the result of valid queries
  3. ensure statically that the result of a given query is of a given (expected) type if the input dataset is guaranteed to be of a given type

16

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XQuery type system components

- Atomic types
  - *xdt:untypedAtomic*
  - All 19 primitive XML Schema types
  - All user defined atomic types
- Empty, None
- Type constructors (simplification!)
  - Elements: *element name {type}*
  - Attributes: *attribute name {type}*
  - Alternation : *type1* | *type 2*
  - Sequence: *type1, type2*
  - Repetition: *type**
  - Interleaved product: *type1 & type2*

- $type_1$ *intersect* $type_2$?
- $type_1$ *subtype of* $type_2$?
- $type_1$ *equals* $type_2$?

17

XML queries

- An XQuery basic structure:
  - a *prolog* + an *expression*
- Role of the prolog:
  - Populate the context where the expression is compiled and evaluated
- Prologue contains:
  - namespace definitions
  - schema imports
  - default element and function namespace
  - function definitions
  - collations declarations
  - function library imports
  - global and external variables definitions
  - etc

18

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XQuery expressions

XQuery Expr :=Constants | Variable | FunctionCalls | PathExpr |
ComparisonExpr | ArithmeticExpr | LogicExpr |
FLWRExpr | ConditionalExpr | QuantifiedExpr |
TypeSwitchExpr | InstanceofExpr | CastExpr |
UnionExpr | IntersectExceptExpr |
ConstructorExpr | ValidateExpr | NodeConstructors

Expressions can be nested with full generality !

Functional programming heritage.

19

Path expressions

- Second order expression

  expr1 / expr2

- Semantics:
  1. Evaluate *expr1* => sequence of nodes
  2. Bind . to each node in this sequence
  3. Evaluate *expr2* with this binding => sequence of nodes
  4. Concatenate the partial sequences
  5. Eliminate duplicates
  6. Sort by document order
- Implicit iteration

$x/a//b $x/..//c $x/data(.) $x/count(./b)

20

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Simple iteration expression

- Syntax :

```
for  variable in expression1
return expression2
```

- Example

```
for $x in document("bib.xml")/bib/book
return  $x/title
```

- Semantics :
  - bind the variable to each root node of the forest returned by *expression1*
  - for each such binding evaluate *expression2*
  - concatenate the resulting sequences
  - nested sequences are automatically flattened

21

Local variable declaration

- Syntax :

```
let variable := expression1
return expression2
```

- Example :

```
let $x :=document("bib.xml")/bib/book
return count($x)
```

- Semantics :
  - bind the *variable* to the result of the *expression1*
  - add this binding to the current environment
  - evaluate and return *expression2*

22

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

FLW(O)R expressions

- Syntactic sugar that combines FOR, LET, IF



- Example

```
for $x in //bib/book                  /* similar to FROM in SQL */
 let $y := $x/author                  /* no analogy in SQL */
 where $x/title="The politics of experience"
                                      /* similar to WHERE in SQL */
 return count($y)                     /* similar to SELECT in SQL */
```

23

FLWR expression semantics

- FLWR expression:

```
for $x in //bib/book
let $y := $x/author
where $x/title="Ulysses"
return count($y)
```

- Equivalent to:

```
for $x in //bib/book
return  (let $y := $x/author
         return
                 if ($x/title="Ulysses" )
                 then count($y)
                 else  ()
         )
```

24

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

More FLWR expression examples

- Selections

```
for $b in document("bib.xml")//book
where $b/publisher = “Springer Verlag" and
          $b/@year = "1998"
return $b/title
```

- Joins

```
for $b in document("bib.xml")//book,
        $p in //publisher
where $b/publisher = $p/name
return ( $b/title , $p/address)
```

25

Node constructors

- Constructing new nodes:
  - elements
  - attributes
  - documents
  - processing instructions
  - comments
  - text
- Side-effect operation
  - Affects *optimization* and *expression rewriting*
- Element constructors create local scopes for namespaces
  - Affects *optimization* and *expression rewriting*

26

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Literal vs. evaluated element content

```
<result>
   literal text content
</result>


<result>
       {  $x/name    }
</result>


<result>
     some content here {$x/name} and some
more  here
</result>
```

- Braces "{}" used to delineate evaluated content

27

A more complex example

```
<livresFrancais>
   {for $x in fn:doc("input.xml")//book
   where $x/year > 2000 and some $y in $x/author satisfies
   $y/address/country="France"
   return
    <livre annee="{$x/year}">
         <titre>{$x/title/text()}</titre>
        { for $z in $x/( author | editor )
           return
                 if(fn:name($z)="editor)
                 then <editeur>{$z/*}</editeur>
               else  <auteur>{$z/*}</auteur>
        }
    </livre>
    }
    </livresFrancais>
```

28

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Nested scopes

```
declare namespace ns="uri1"

for $x in fn:doc("uri")/ns:a
where $x/ns:b  eq 3
return
    <result xmlns:ns="uri2">
        { for $x in fn:doc("uri")/ns:a
          return $x / ns:b  }
    </result>
```

Local scopes impact optimization and rewriting !

29

XQuery Use Case Scenarios

- XML transformation language in Web Services
  - Large and very complex queries
  - Input message + external data sources
  - Small and medium size data sets (*x*K -> *x*M)
  - Transient and streaming data (no indexes)
  - With or without schema validation
- XML message brokers
  - Simple path expressions, single input message
  - Small data sets
  - Transient and streaming data (no indexes)
  - Mostly non schema validated data
- Semantic data verification
  - Mostly messages
  - Potentially complex (but small) queries
  - Streaming and multiquery optimization required

30

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XQuery Usage Scenarios (cont.)

- Data Integration
  - Complex but smaller queries (FLOWRs, aggregates, constructors)
  - Large, persistent, external data repositories
  - Dynamic data (via Web Services invocations)
- Large volumes of centralized XML data
  - Logs and archives
  - Complex queries (statistics, analytics)
  - Mostly read only
- Large content repositories
  - Large volume of data (books, manuals, etc)
  - With or without schema validation
  - Full text essential, update required

31

XQuery Usage Scenarios (cont.)

- Large volumes of distributed textual data
  - XML search engines
  - High volume of data sources
  - Full text, semantic search crucial
- RSS data
  - High number of input data channels
  - Data is pushed, not pulled
  - Structure of the data very simple, each item bounded size
  - Aggregators using mostly full-text search

32

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Criteria for XQuery usages

1. Type of queries (e.g. simple, complex, construction-intensive, full text search intensive)
2. Volume of queries
3. Native XML or virtual XML views of other forms of data
4. XML Schema validated data or not
5. Volume of data per query
6. Number of data sources
7. Transient data vs. persistent data
8. Transacted vs. non-transacted data
9. Push data vs. pull data
10. Typed vs. untyped data
11. Read only data vs. updatable data
12. Distributed vs. centralized data sets
13. Data compressed/encrypted or not
14. Target architectures
15. Customer expectation

Each scenario requires different processing techniques.

33

XQuery: a language at the cross-roads

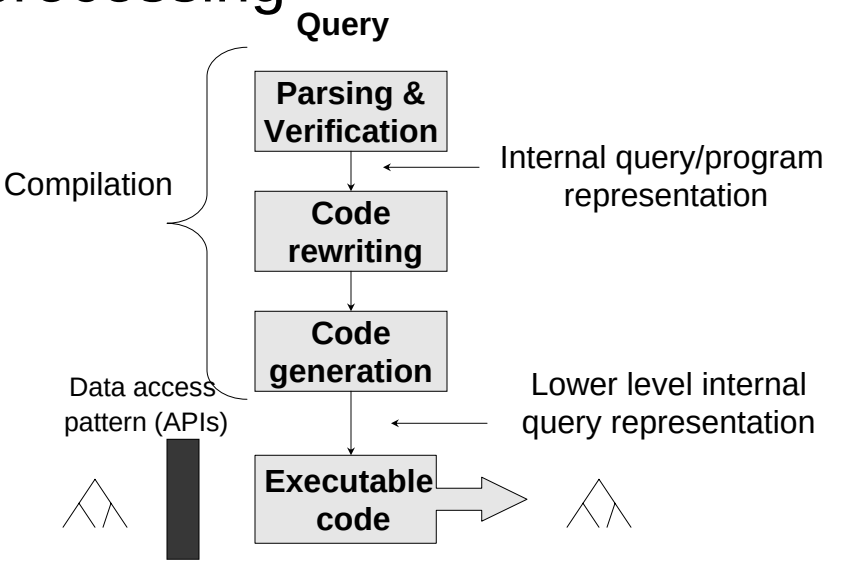
- Query languages
- Functional programming languages
- Object-oriented languages
- Procedural languages
- Some new features : context sensitive semantics

- Processing XQuery has to learn from all those fields, plus innovate

34

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Major steps in XML Query processing



35

Code rewriting

- Code rewritings *goals*
  1. Reduce the *level of abstraction*
  2. Reduce the *execution cost*
- Code rewriting *concepts*
  - Code representation
    - db: algebras
  - Code transformations
    - db: rewriting rules
  - Cost transformation policy
    - db: search strategies
  - Code cost estimation

36

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Code representation

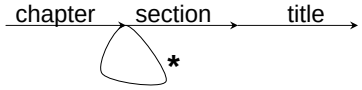
- Is "*algebra*" the right metaphor ? Or *expressions* ? *Annotated expressions* ? *Automata* ?
- *Standard* algebra for XQuery ?
- *Redundant* algebra or not ?
  - Core algebra in the XQuery Formal Semantics
- *Logical* vs. *physical* algebra ?
  - What is the "physical plan" for 1+1 ?
- Additional structures, e.g. *dataflow graphs* ? *Dependency graphs* ?

See Compiler transformations for High-Performance computing
Bacon, Graham, Sharp

37

Automata representation

- Path expressions
  $x/chapter//section/title




```
<book>
   <chapter>
      <section>
         <title/>
      </section>
   </chapter>
</book>
```

```
begin book
begin chapter
begin section
begin title
end title
end section
end chapter
end book
```

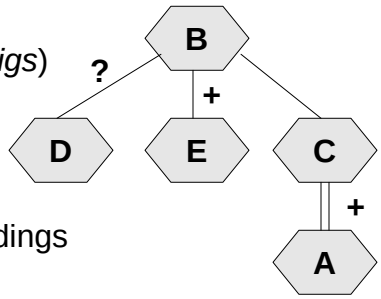
- *[Yfilter'03, Gupta'03, etc]*
- NFA vs. DFA vs. AFA
- one path vs. a set of paths
- Problems
  - Not extensible to full XQuery
  - Better suited for push execution, pull is harder
  - Lazy evaluation is hard

38

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

TLC Algebra
(Jagadish et al. 2004)

- XML Query tree patterns (called *twigs*)
- Annotated with predicates
- Tree matching as basic operation
  - Logical and physical operation
- Tree pattern matching => tuple bindings (i.e. relations)
- Tuples combined via classical relational algebra
  - Select, project, join, duplicate-elim., …



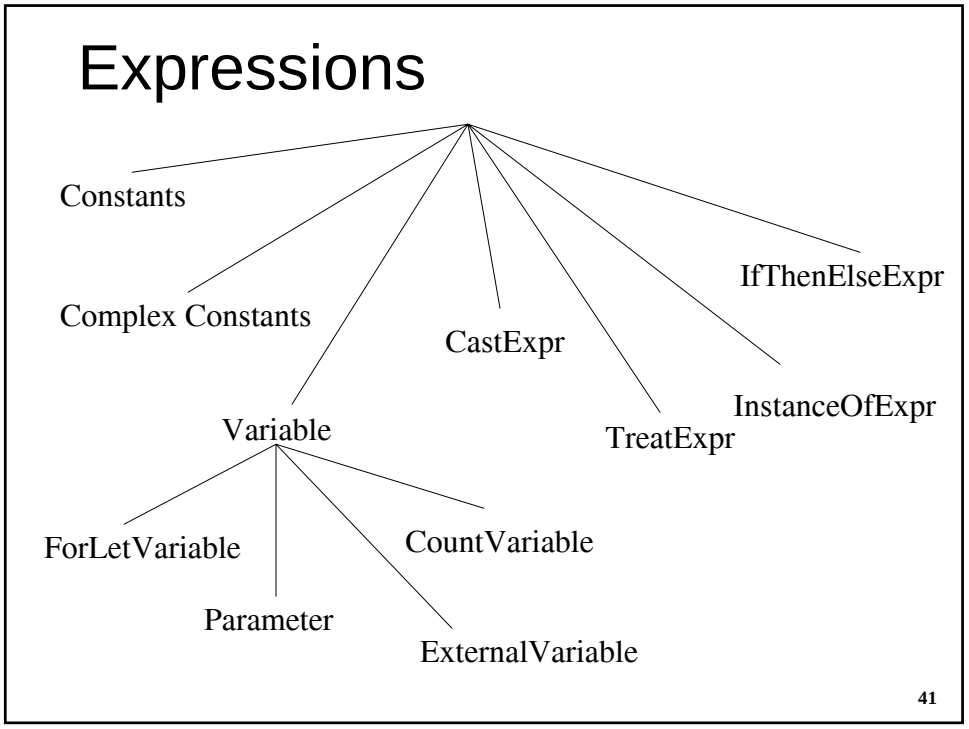
39

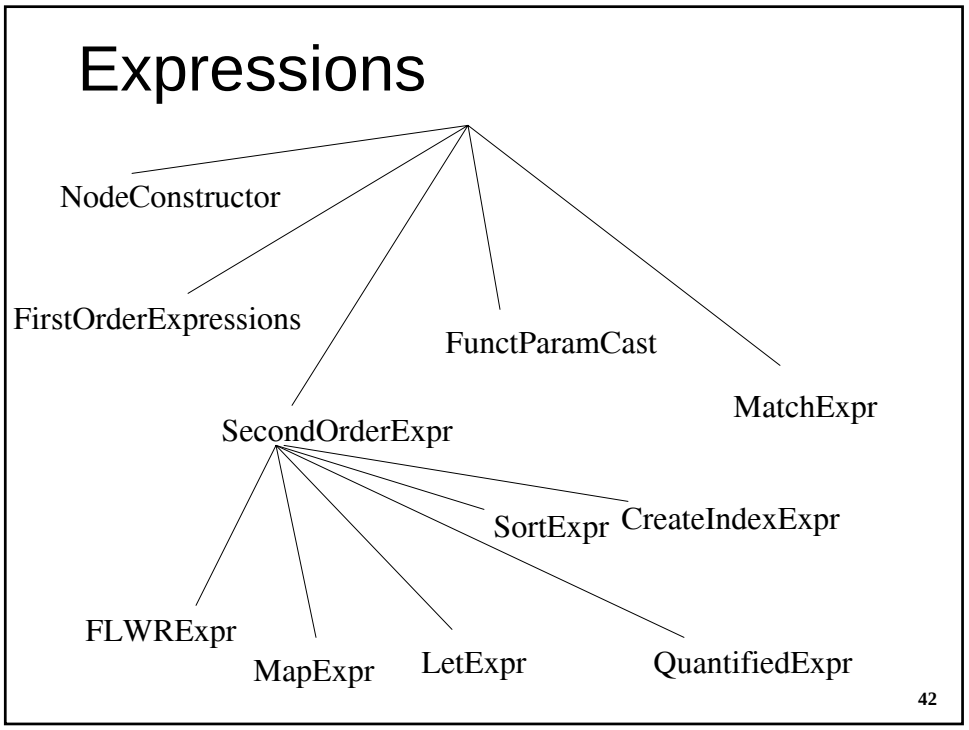
XQuery Expressions (BEA implementation)

- Expressions built during parsing
- (almost) 1-1 mapping between expressions in XQuery and internal ones
  - Differences: Match (expr, NodeTest) for path expressions
- Annotated expressions
  - *E.g. unordered* is an annotation
  - Annotations exploited during optimization
- Redundant algebra
  - E.g. general FLWR, but also LET and MAP
  - E.g. typeswitch, but also instanceof and conditionals
- Support for dataflow analysis is fundamental

40

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.





2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Expression representation example

for $line in $doc/Order/OrderLine
where $line/SellersID eq 1
return <lineItem>{$line/Item/ID}</lineItem>

for $line in $doc/Order/OrderLine
where xs:integer(fn:data($line/SellersID)) eq 1
return <lineItem>{$line/Item/ID}</lineItem>

Original **Normalized**

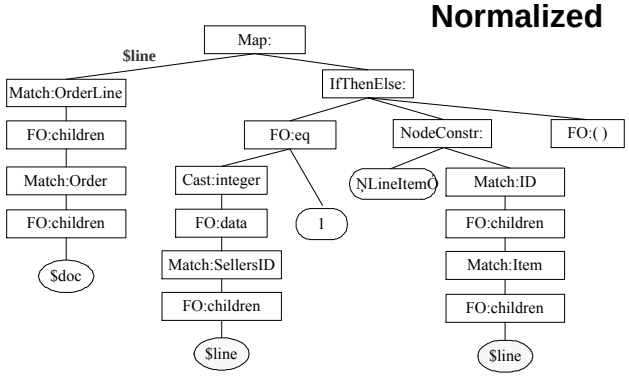


Fig. 3-1. Expression tree of query Q1.

43

XQuery logical rewritings

- Algebraic properties of comparisons
- Algebraic properties of Boolean operators
- LET clause folding and unfolding
- Function inlining
- FLWOR nesting and unnesting
- FOR clauses minimization
- Constant folding
- Common sub-expressions factorization
- Type based rewritings
- Navigation based rewritings
- “Join ordering”

44

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Algebraic properties of comparisons

- General comparisons not reflexive, transitive
  - (1,3) = (1,2) *(but also !=, <, >, <=, >= !!!!!)*
  - Reasons
    - implicit existential quantification, dynamic casts
- Negation rule does not hold
  - fn:not($x = $y) is not equivalent to $x != $y
- General comparison not transitive, not reflexive
- Value comparisons are *almost* transitive
  - Exception:
    - xs:decimal due to the loss of precision

Impact on grouping, hashing, indexing, caching !!!

45

XML data equivalence

- XQuery has multiple notions of data “equality”
  - “=“, “*eq*”, “*is*”, “*fn:deep-equal()”*
- Expected properties:
  - *Transitivity*, *reflexivity* and *symmetry*
  - Necessary for grouping, indexing and hashing
- Additional property:
  - if ($data_1$ equal $data_2$) then (**f**($data_{1)}$ equal **f**($data_2$))
  - Necessary for memoization, caching
- None of the equality relationships above (except “is”) satisfies those properties
- The “*is*” relationship only applies to nodes
- Careful implementations for *indexes*, *hashing*, *caches*

46

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Properties of Boolean operators

- *Among of the most useful logical rewritings: PCNF and PDNF*
- *And*, *or* are commutative & allow short-circuiting
  - For optimization purposes
- But are non-deterministic
  - Surprise for some programmers :(
    - If (($x castable as xs:integer) and (($x cast as xs:integer) eq 2)) …..
- 2 value logic
  - () is converted into fn:false() before use
- Conventional distributivity rules for *and*, *not*, *or* do hold

47

LET clause folding

- Traditional FP rewriting

```
let $x := 3                    3+2
return $x +2
```

- Not so easy !

```
let $x := <a/>          (<a/>, <a/> )      NO. Side effects. (Node identity)
return ($x, $x )

declare namespace ns="uri1"                NO. Context sensitive
let $x := <ns:a/>                           namespace processing.
return <b xmlns:ns="uri2">{$x}</b>

            declare namespace ns:="uri1"
            <b xmlns:ns="uri2">{<ns:a/>}</b>
```

XML does not allow cut and paste

48

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

LET clause folding (cont.)

- Impact of unordered{..} /* context sensitive*/

```
let $x := ($y/a/b)[1]            the c's of a specific b parent
return unorderded { $x/c }       (in no particular order)
```

not equivalent to

```
unordered {($y/a/b)[1]/c }     the c's of "some" b
                                 (in no particular order)
```

49

LET clause folding : fixing the node construction problem

- Sufficient conditions

```
(: before LET :)              (: before LET :)
let $x := expr1               (: after LET :)
(: after LET :)               return expr2'
return expr2
                              where expr2' is expr2 with substitution {$x/expr1}
```

  - Expr1 does never generate new nodes in the result
  - OR $x is used (a) only once and (b) not part of a loop and (c) not input to a recursive function
  - Dataflow analysis required

50

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

LET clause folding: fixing the namespace problem

- Context sensitivity for namespaces
  1. Namespace resolution during query analysis
  2. Namespace resolution during evaluation
- (1) is not a problem if:
  - Query rewriting is done *after* namespace resolution
- (2) could be a serious problem (***)
  - XQuery avoided it for the moment
  - Restrictions on context-sensitive operations like string -> Qname casting

51

LET clause unfolding

- Traditional rewriting

```
for $x := (1 to 10)                 let $y := ($input+2)
return ($input+2)+$x                for $x in (1 to 10)
                                    return  $y+$x
```

- Not so easy!
  - Same problems as above: side-effects, NS handling and unordered/ordered{..}
  - Additional problem: *error handling*

```
for $x in (1 to 10)                 let $y := ($input idiv 0)
return  if($x lt 1)                 for $x in (1 to 10)
        then  ($input idiv 0)       return if ($x lt 1)
        else  $x                            then $y
                                            else  $x
```

Guaranteed only if runtime implements consistently lazy evaluation

Otherwise dataflow analysis ***and*** error analysis required.

52

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Function inlining

- Traditional FP rewriting technique

```
define function f($x as xs:integer) as xs:integer        2+1
{$x+1}
f(2)
```

- Not always!
  - Same problems as for LET (NS handling, side-effects, unordered {…})
  - Additional problems: *implicit operations (atomization, casts)*

```
define function f($x as xs:double) as xs:boolean
{$x instance of xs:double}
f(2)

(2 instance of xs:double)          NO
```

- Make sure this rewriting is done *after* normalization

53

Join ordering

- In SQL the FROM clauses do not have an order
  - Formally: Cartesian product of bags: commutative
  - The SQL decides the order of accessing the relations ("join ordering")
- In XQuery the FOR clauses have a fixed order
  - Formally: Cartesian products of lists: *non-commutative*
  - Additional complexity: dependent joins

```
for $x in (1,2)
    $y in ("a", "b")
return <result>{$x, $y}</result>
```

- (1,"a"), (1, "b"), (2,"a"), (2, "b")

```
for $x in //departments
    $y in $x/employees
return  <result>
            {$x/name, $y/name}
        </result>
```

54

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Join ordering (2)

- Dataflow analysis to detect cases when the order of the FOR clauses can be changed
  - When unordered {…} is used in the query
  - When the result is later re-ordered (implicitly or explicitly)
  - When the result is passed as input to an operation that ignores the order (e.g. count)
  - When there is no data dependency

55

FLWR unnesting

- Traditional database technique

```
for $x in (for $y in $input/a/b
           where $y/c eq 3
           return $y/d)
where $x/e eq 4
return $x
```

```
for $y in $input/a/b,
    $x in $y/d
where ($x/e eq 4) and ($y/c eq 3)
return $x
```

- Problem simpler than in OQL/ODMG
  - No nested collections in XML
- Order-by, count variables and unordered{…} limit the limits applicability

56

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

FLWR unnesting (cont.)

- Another traditional database technique

```
for $x in $input/a/b
where $x/c eq 3
return (for $y in $x/d)
        where $x/e eq 4
         return $y)
```

```
for $x in $input/a/b,
    $y in $x/d
where ($x/e eq 4) and ($y/c eq 3)
return $y
```

- Same comments apply

57

FOR clauses minimization

- Yet another useful rewriting technique

```
for $x in $input/a/b,
    $y in $input/c
where ($x/d eq 3)
return $y/e
```

```
for $x in $input/a/b
where ($x/d eq 3)
return $input/c/e
```

```
for $x in $input/a/b,
    $y in $input/c
where $x/d eq 3  and $y/f eq 4
return $y/e
```

```
for $x in $input/a/b
where $x/d eq 3 and $input/c/f eq 4
return $input/c/e
```

NO

```
for $x in $input/a/b
     $y in $input/c
where ($x/d eq 3)
return <e>{$x, $y}</e>
```

```
for $x $input/a/b
where ($x/d eq 3)
return <e>{$x, $input/c}</e>
```

NO

58

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Constant folding

- Yet another traditional technique

```
for $x in  (1 to 10)                  for  $x in (1 to 10)
where $x eq 3                         where $x eq 3            YES
return $x+1                           return (3+1)

for $x in $input/a                    for $x in $input/a
where $x eq 3                         where $x eq 3              NO
return <b>{$x}</b>                    return <b>{3}</b>

for $x in (1.0,2.0,3.0)               for $x in (1.0,2.0,3.0)    NO
where $x eq 1                         where $x eq 1
return ($x instance of xs:integer)    return (1 instance of xs:integer)
```

59

Common sub-expression factorization

- Preliminary questions
  - *Same* expression ?
  - *Same* context ?
  - *Error* “equivalence” ?
  - Create the same *new nodes*?

```
for $x in $input/a/b                  let $y := (1 idiv 0)
where $x/c lt 3                       for $x in $input/a/b
return if ($x/c lt 2)                 where $x/c lt 3
       then if ($x/c eq 1)             return if($x/c lt 2)
            then (1 idiv 0)                  then if ($x/c eq 1)
            else $x/c+1                           then $y
       else   if($x/c eq 0)                       else $x/c+1
              then (1 idiv 0)                 else if($x/c eq 0)
              else $x/c+2                           then $y
                                                    else $x/c+2
```

60

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Type-based rewritings

- Type-based optimizations:
  - Increase the advantages of lazy evaluation
    - $input/a/b/c (((($input/a)[1]/b[1])/c)[1]
  - Eliminate the need for expensive operations (sort, dup-elim)
    - $input//a/b $input/c/d/a/b
  - Static dispatch for overloaded functions
    - e.g. min, max, avg, arithmetics, comparisons
    - Maximizes the use of indexes
  - Elimination of no-operations
    - e.g. casts, atomization, boolean effective value
  - Choice of various run-time implementations for certain logical operations

61

Dealing with backwards navigation

- Replace backwards navigation with forward navigation

```
for $x in $input/a/b
return <c>{$x/.., $x/d}</c>
```

```
for $y in $input/a,
    $x in $y/b
return <c>{$y, $x/d}</c>
```

YES

```
for $x in $input/a/b
return <c>{$x//e/..}</c>
```

??

- Enables streaming

62

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

More compiler support for efficient execution

- Streaming vs. data materialization
- Node identifiers handling
- Document order handling
- Scheduling for parallel execution
- Projecting input data streams

63

When should we materialize?

- Traditional operators (e.g. sort)
- Other conditions:
  - Whenever a variable is used multiple times
  - Whenever a variable is used as part of a loop
  - Whenever the content of a variable is given as input to a recursive function
  - In case of backwards navigation
- Those are the ONLY cases
- In most cases, materialization can be *partial* and *lazy*
- Compiler can detect those cases via dataflow analysis

64

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

How can we minimize the use of node identifiers ?

- Node identifiers are required by the XML Data model but onerous (time, space)
- Solution:
  1. Decouple the node construction operation from the node id generation operation
  2. Generate node ids *only* if *really* needed
     - Only if the query contains (after optimization) operators that need node identifiers (e.g. sort by doc order, is, parent, <<) OR node identifiers are required for the result
- Compiler support: dataflow analysis

65

How can we deal with path expressions ?

- Sorting by document order and duplicate elimination required by the XQuery semantics but very expensive
- Semantic conditions
  - $document / a / b / c
    - Guaranteed to return results in doc order and not to have duplicates
  - $document / a // b
    - Guaranteed to return results in doc order and not to contain duplicates
  - $document // a / b
    - NOT guaranteed to return results in doc order but guaranteed not to contain duplicates
  - $document // a // b $document / a / .. / b
    - Nothing can be said in general

66

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Parallel execution

ns1:WS1($input)+ns2:WS2($input)
for $x in (1 to 10)
return ns:WS($i)

- Obviously certain subexpressions of an expression can (and should...) be executed in parallel
  - Scheduling based on data dependency
- Horizontal and vertical partitioning
- Interraction between errors and paralellism

See David J. DeWitt, Jim Gray:
Parallel Database Systems: The Future of High Performance Database Systems.

67

XQuery expression analysis and inferencing for compilation

- How many times does an expression use a variable ?
- Is an expression using a variable as part of a loop ?
- Is an expression a *map* on a certain variable ?
- Is an expression guaranteed to return results in doc order ?
- Is an expression guaranteed to return (node) distinct results?
- Is an expression a “function” ?
- Is an expression an injective function on a certain variable ?
- Can the result of an expression contain newly created nodes ?
- Is the evaluation of an expression context-sensitive ?
- Can an expression raise user errors ?
- Is a sub expression of an expression guaranteed to be executed ?
- Etc.

68

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Errors and optimization

- Inherent *tension* between the determinism of errors and ability to do code rewritings (aka optimizations) and lazy evaluation.

(1 div 0) + (2 div 0) () + (1 div 0)

  - Imperative programming: errors are all deterministic, but very little rewriting and re-organization of the code is allowed. Also, no lazy evaluation either.
  - Declarative programming: rewriting and lazy evaluation are important goals, but errors are non-deterministic as a result
- XQuery: at the crossroads. Covers the gray area.
  - Semantics allows non-deterministic errors in certain conditions to facilitate optimization
  - However, this impacts the usability and the possibility to debug the programs
  - Most decent implementations have flexible compilers

69

XQuery vs. SQL compilation

- Data model and language very different
- Use case scenarios, applications very different
- Some important differences for compilation
  - Context sensitive semantics
  - Error handling
  - Order preservation
  - Lazy evaluation more important

70

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Some open problems

1. XQuery equivalence
2. XQuery subsumption
3. Answering queries using views
4. Memoization for XQuery
5. Caching for XQuery
6. Partial and lazy indexes for XML and XQuery
7. XQueries independent of updates
8. Xqueries independent of schema changes
9. Reversing an XML transformation
10. Data lineage through XQuery
11. Keys and identities on the Web

71

Some open problems (cont.)

11. Declarative description of data access patterns; query optimization based on such descriptions
12. Integrity constraints and assertions for XML
13. Query reformulation based on XML integrity constraints
14. XQuery and full text search
15. Parallel and asynchronous execution of XQuery
16. Distributed execution of XQuery in a peer-to-peer environment
17. Automatic testing of schema verification
18. Optimistic XQuery type checking algorithm
19. Debugging and explaining XQuery behavior
20. XML diff-grams
21. Automatic XML Schema mappings

72

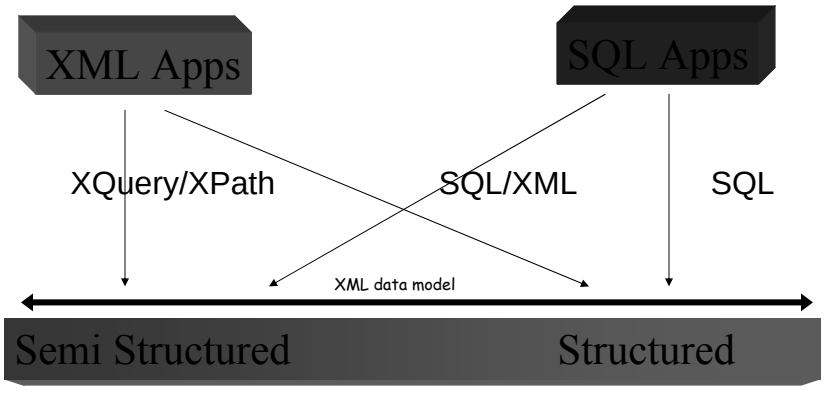
2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Oracle's XQuery support



73

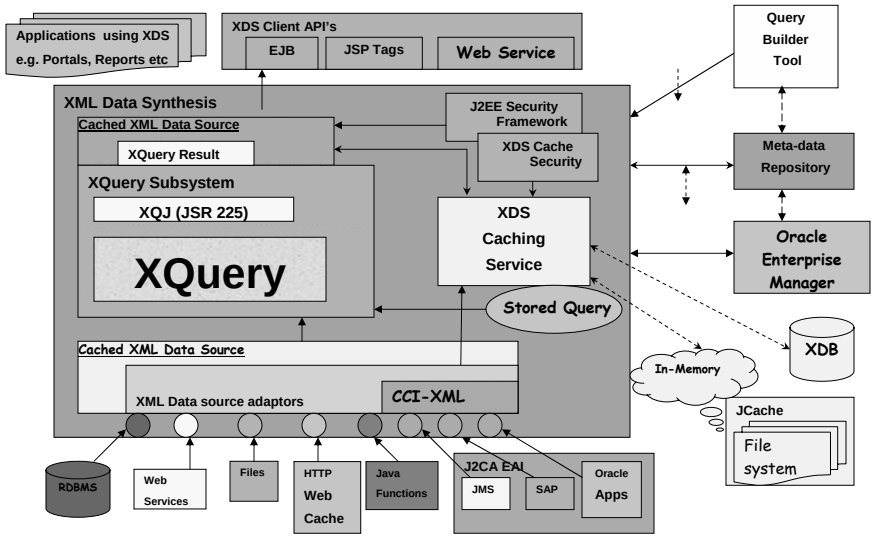
Enterprise Information Management



74

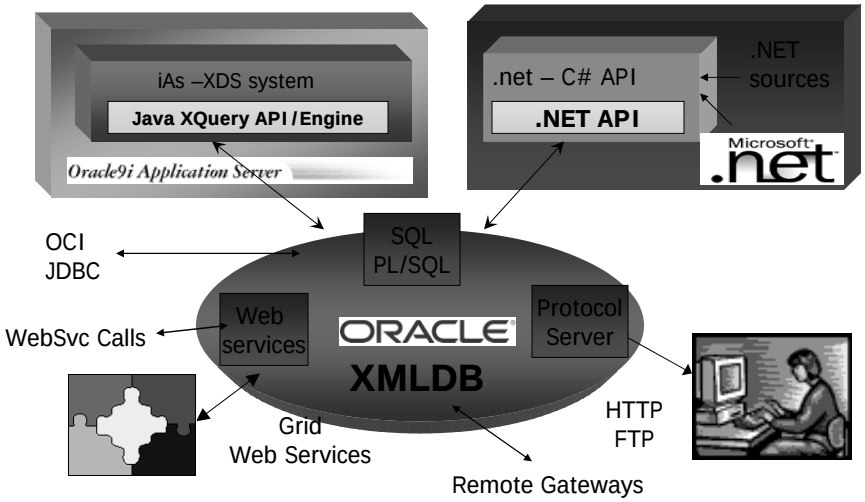
2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Enterprise Information Integration



75

XML Access



76

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Accessing XML

- SQL – SQL/XML standards
  - XMLType – Type, Storage
  - Publishing Functions (SQL/XML 2003)
  - XQuery - XMLQuery/XMLTable
  - XSL-T - XMLTransform()
  - Schema validation – isValid(), schemaValidate()
- PL/SQL
  - Packages for generation, DOM, parsing, transformation etc.
- C/C++ - OCI/OCCI
  - Access XMLType through SQL
  - C-based DOM, parsers, validators

77

Accessing XML

- Java
  - Java based parsers, DOM, schema validators, SQL utilities for generation/shredding of XML, XML servlets
  - XSL-T engine
  - XQuery engine with XQJ – JSR 225 API access
  - JDBC access to XMLType with unified DOM access in Java tier
    - DOM from server is lazilly manifested
    - Binary XML from server is supported in Java DOM
- .NET
  - XMLType APIs in .Net using OCI
  - Ability to generate XML, store and retrieve XMLType in .NET

78

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Java standards

- JSR 225 (XQJ) - Public draft available
  - Define standard for access in Java
  - Same API for XQuery in mid-tier & backend.
    - Provide Java classes for data model
    - Provide all types of access (SAX, StaX)
    - Metadata and type support
    - Output of one query is input of another
- JDBC 4.0 – Support SQL/XML XMLType data model in JDBC

79

Example – JSR 225

Draft API – subject to change

```
import javax.xml.xquery;

XQConnection con = Datasource.getconnection(jdbcco);
XQExpression expr = con.prepareExpr("for $i in doc(..");
XQSequence seq = expr.executeQuery();

while (seq.next()) {
  if (seq.getItemType() == XQITemType.XS_INT)
    int intVal = items.getInt();
}

con.close();
```

80

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

SQL/XML standards

- XML datatype will support XQuery data model
  - XML(Content) – old XML infoset model
  - XML(Sequence) – holds sequences
- XMLQuery() function to create XML content using XQuery
- XMLTable function to explode XML to relational values – using XQuery
- Provide mapping between SQL & XQuery data model

81

XMLQuery expression

- SQL Expression – use in select for constructing XML

```
select XMLQuery(
  'for $i in ./PurchaseOrder
   where $i/PoNo = $j/val
   return $i//Item '
  passing pocol ,
          xmlelement("val",2100) as "j"
    returning content)
from purchaseorder
```

Pocol maps to default item

XMLElement value maps to $j

```
<Item itemno="21"><Quantity>200</Quantity>..</Item>
<Item itemno="22"><Quantity>22</Quantity>..</Item>
```

82

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XMLTable construct

- Used in FROM clause
- Splits up result into SQL columns

```
select items.pos, items.itemno, items.quantity
from purchaseorder p,
   XMLTable('for $i in /PurchaseOrder//Items
               where $i/Quantity > 200
               return $i'  passing pocol
             columns pos for ordinality,
                     itemno number path 'ItemNo'
                   quantity number path 'Quantity' DEFAULT 0
  ) items;
```

Relational columns returned in result

Ordinality returns sequential position

```
POS  ITEMNO   QUANTITY
------  ----------  ----------
1      21          21
2      22          0
```

Default value is used If path does not return value

83

XMLTable contd..

- Can be chained (master-detail drill down)

```
select items.pono, items.itemno, locs.city
from purchaseorder p,
   XMLTable('for $i in /PurchaseOrder//Items
               where $i/Quantity > 200
               return $i'  passing pocol
             columns  pono   number path '../PurchaseOrder/@PoNo'
                      itemno number path 'ItemNo'
                       locs   xmltype path 'Locations' ) items,
  XMLTable('for $i in /Location where $i/State = "CA" return $i'
             passing items.locs
              columns   city varchar2(20) path 'City' ) locs;
```

```
PONO   ITEMNO   CITY
-------  ----------  ----------
100       21          Redwood City
100       21          Foster City
100       22          Berkeley
101       30          Los Angeles
```

84

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XQuery Support in Oracle

- Standalone Java query engine
  - 100% Java
  - Integrated into Oracle App Server -XDS
  - Interoperates with XSLT/XPath
- XMLDB integrated database engine
  - SQL/XML standard support
  - Optimized queries – rewrite to relational

85

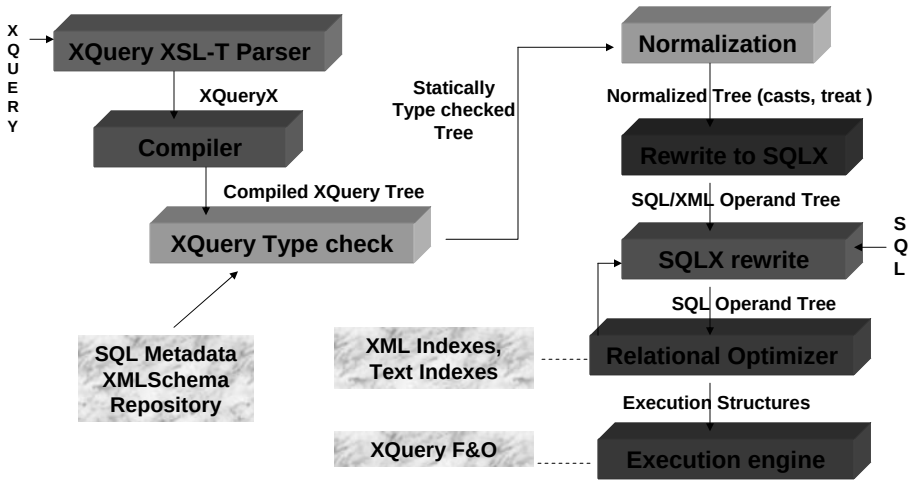
XQuery database support

- Production in Oracle Database 10gr2
- Supports XMLQuery and XMLTable construct
- Native compilation into SQL/XML structures
- Returns XMLType(Content)
- Can query over relational, O-R, XMLType data
- fn:doc - Maps to XDB Repository on server
- SQLPlus provides *xquery* command to execute XQuery
- XSL-T will also get compiled in to XQuery

86

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Architecture



87

Example SQL integration

- SQL tables (get all 200K+ employees)

```
select XMLQuery('
  for $i in ora:view("SCOTT","EMP")/ROW
 where $i/SALARY  > 200000
 return   $i/EMPNO' returning content)
from dual;

<EMPNO>2100</EMPNO>
<EMPNO>344</EMPNO>
```



88

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Example – XMLType integration

- XMLType tables as input

```
select p.XQuery('<PO pono="{/PoNo}"/>')
 from purchaseorder p
 where p.XQuery('/PurchaseOrder[ShipAddr/City="Fresno"]')
          is not null;
```

- **After rewrite**

```
select XMLElement("PO",
             XMLAttributes(p.xmldata."PONo"))
from purchaseorder p
where p.xmldata."ShipAddr"."City" = 'Fresno'
```



89

Example – Repository integration

- Query files from repository
  - doc() queries files from repository (rewrite)
  - collection() maps to directories

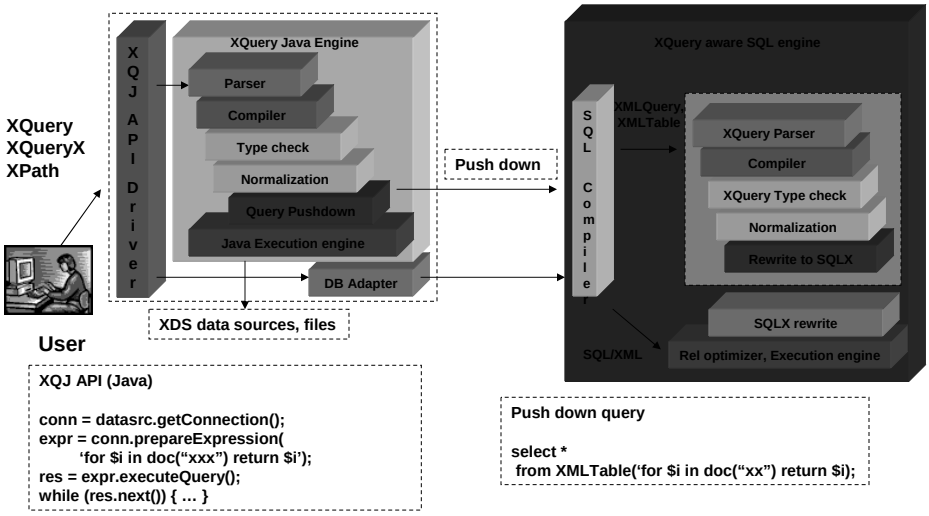
```
select XQuery(' for $i in doc("/public/foo.xml")
                    return $i')
from dual;

<FOO> ....</FOO>
```

90

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Java Layer Architecture




```
XQJ API (Java)

conn = datasrc.getConnection();
expr = conn.prepareExpression(
       'for $i in doc("xxx") return $i');
res = expr.executeQuery();
while (res.next()) { ... }
```

```
Push down query

select *
 from XMLTable('for $i in doc("xx") return $i);
```

91

XQuery Java implementation

- XQuery or XQueryX input
- Extensible function implementation
- Compiles into rowsource like structures
- Optimization – push XQuery to XMLDB
- XQJ API driver – for accessing mid tier/backend
- Shared data model with XSL/XPath
- Shared F&O – pre-defined & external
  - Standard Function implementation interfaces
  - Write Java func once – use it in XQuery/XSLT

92

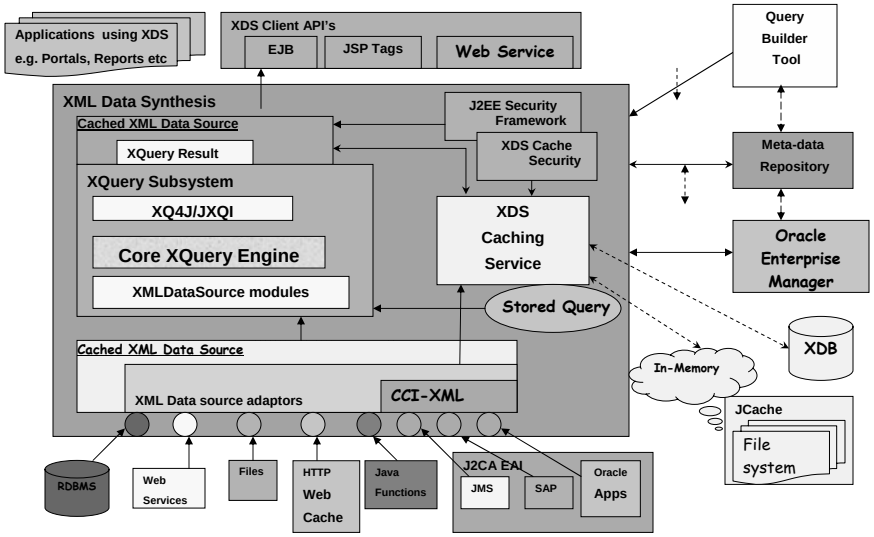
2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XDS – XML Data Synthesis

- XDS provides common XQuery services
  - Caching – cache in Java/XMLDB
  - View text storage, functional definitions
  - Web services for XQuery
  - Common XML Adapter framework
- XQuery – the integration language
- Caching in XMLDB – query pushdown
- JNDI tree used to discover datasource

93

XQuery in XDS



94

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Example – XDS usage

- User registers webservice as datasource
- XDS creates an XQuery module automatically
- User Query for querying webservice

```
import module namespace wss="datasrc/stockws";
for $i in wss:getcompanies()
return wss:get_stock_price($i/name)
```

wss – namespace prefix for the loaded module
JNDI lookup to get datasource implementation
XDS adapters implement datasource

95

Conclusion

- XQuery: a language at the crossroads
  - Programming language in it's own right
  - Declarative XML to XML mapping language
  - Borrows features from query languages, functional languages, imperative languages and OO languages
  - Has some unique features
- Usage scenarios for XQuery are very diverse
  - Databases are only one scenario among many others
- Compilation of XQuery is obviously impacted
  - Needs techniques from *all* the fields above
- Just the beginning, lots of open research problems
- XQuery isn't SQL with </> :-)

96

2005 Microsoft Corporation. All rights reserved.
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.