XML & RDBMS’

Introduction

- XML stands for Extensible Markup Language.
- It is designed to describe data and focus on what data is.
- It is used to structure store and to send information.
- It is easy to understand and is self describing.

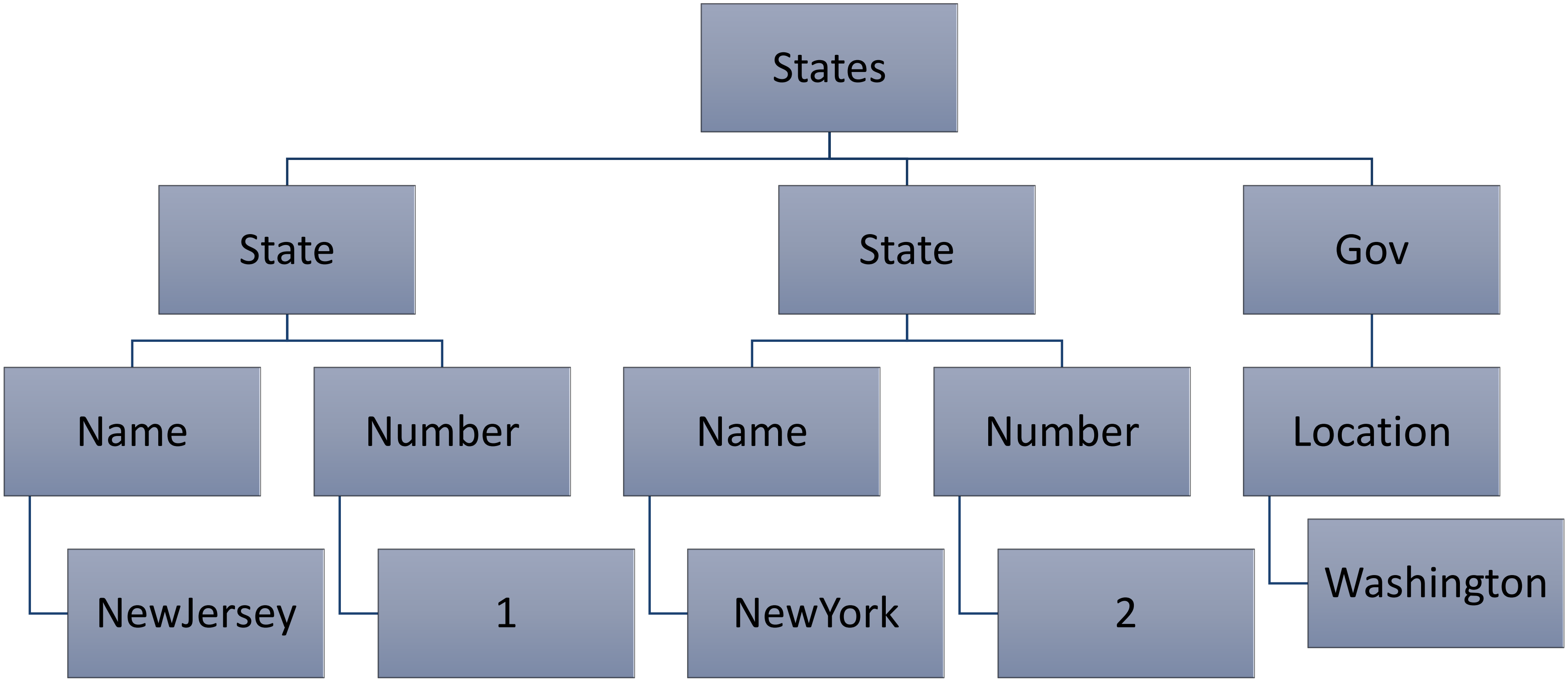
Rules

- The first tag is the root of the tree. There must be a single root
- Every other matching pair of tags becomes one node. If a pair of tags is contained in another pair, the contained pair becomes a child of the containing pair. Children have a defined order
- Text becomes a child of the node corresponding to the tag that encloses the text. This is always a leaf node
- XML allows single tag. Single tag always become leaves with a box
- XML Tags are case sensitive and they must be always properly nested

XML Example

```
<States>
    <State>
            <Name>NewJersey</Name>
            <Number>1</Number>
    </State>
    <State>
            <Name>NewYork</Name>
            <Number>2<Number>
    </State>
    <Gov>
            <Location>Washington</Location>
    </Gov>
</States>
```

Tree Representation of XML



DTD

- A valid XML document is a “well formed” XML document, which also conforms to the rules of a Document Type Definition(DTD).
- A DTD is like a database schema for XML files.

Example of DTD

DTD

```
<?XML version = “1.0”?>
<!DOCTYPE note[
<!Element note(to,from,heading,notebody)>
<!Element to(#PCDATA)>
<!Element from(#PCDATA)>
<!Element heading(#PCDATA)>
<!Element notebody(#PCDATA)>
]>
```

Example XML

```
<note>
    <to>CS 731</to>
    <from>21456687</from>
    <heading>presentation</heading>
    <notebody>XML introduction</notebody>
</note>
```

Interpretation of DTD

- !ELEMENT note defines the note element as having four elements:”to,from,heading,notebody”. In this order.
- <!ELEMENT to(#PCDATA)> defines the “to” element is of type “#PCDATA”.
- PCDATA Parsed Character Data: a character string

Interpretation of DTD Cont..

- Element with children (sequence)

```
<!Element note(to,from,heading ,body)>
```

- Declaring minimum one occurrence of the same element(one or more)

```
<!ELEMENT note (message +)>
```

- Declaring zero or more occurrences of the same element

```
<!ELEMENT note (message *)>
```

- Declaring zero or one occurrences of the same element

```
<!ELEMENT note (message ?)>
```

Advantages of XML

- XML is an open standard.
- It is human readable and not cryptic like a machine language.
- XML processing is easy.
- It can be used to integrate complex web based systems (using XML as communication).

Importance of XML

- Extensible Markup Language (XML) is fast emerging as the dominant standard for representing data on the Internet.
- Most organizations use XML as a data communication standard.
- All commercial development frameworks are XML oriented (.NET, Java).
- All modern web systems architecture is designed based on XML.

Storing and Querying XML in Databases

XML data can be stored in following ways

- Relational database
- File system
- Object-oriented database (e.g., Excelon), or
- a special-purpose (or semi-structured) system such as Lore (Stanford), Lotus Notes, or Tamino (Software AG).

Storing XML in Databases

The primary ways to store XML data can be classified as:

- *Structure-Mapping approach*

  In the Structure Mapping approach the design of database schema is based on the understanding of DTD (Document Type Descriptor) that describes the structure of XML documents.

- *Model-Mapping approach.*

  In the Model Mapping approach no DTD information is required for data storage. A fixed database schema is used to store any XML documents without assistance of DTD.

Storing XML in Databases…

The advantages of the model mapping approaches are:

1. it is capable of supporting any sophisticated XML applications that are considered either as static (the DTDs are not changed) or dynamic (the DTDs vary from time to time)
2. it is capable of supporting well-formed but non-DTD XML applications
3. it does not require extending the expressive power of database models, in order to support XML documents. It is possible to store large XML documents in off-the-shelf DBMS

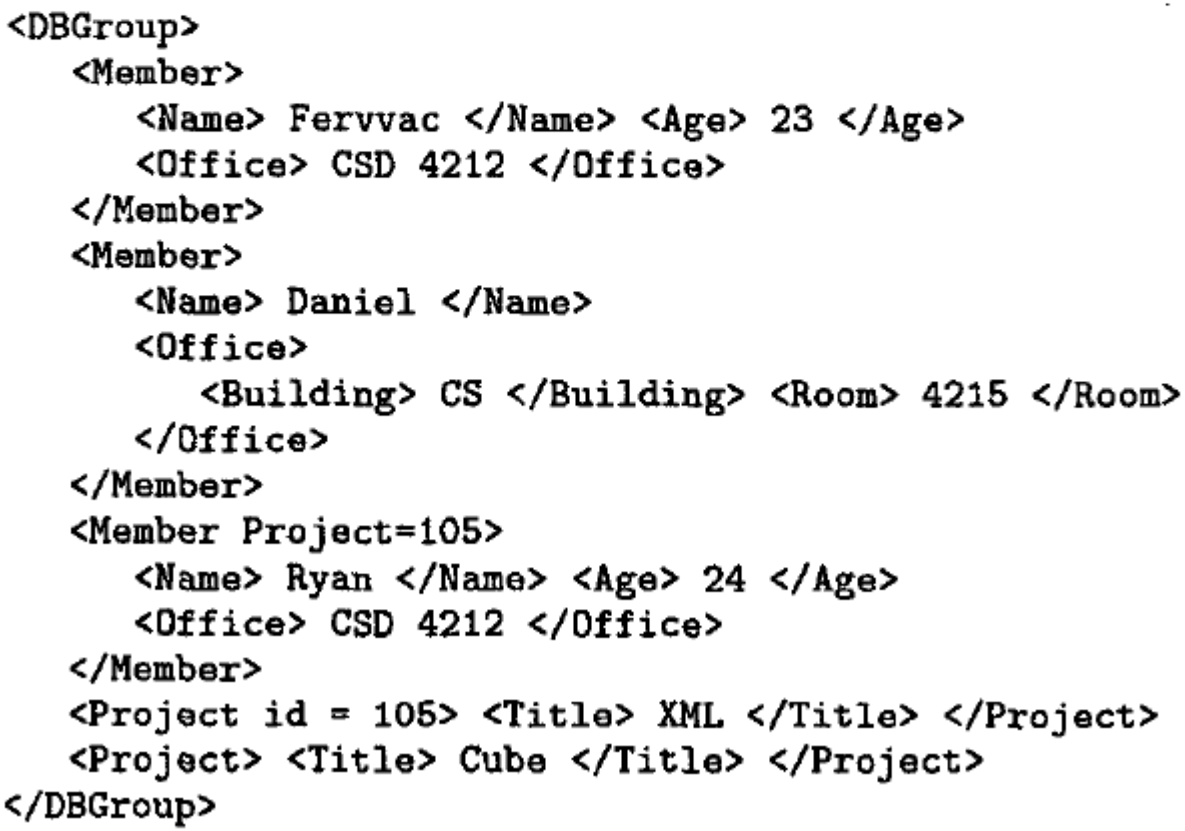
Model Mapping Approaches

- Edge
  All the edges of XML document are stored in a single table.
- Monet
  It Partitions the edge table according to all possible label paths.
- XParent.
  Based on LabelPath, DataPath, Element and Data.
- XRel.
  XML data stored based on Path, Element, Text, and Attribute.

Edge Oriented Approaches

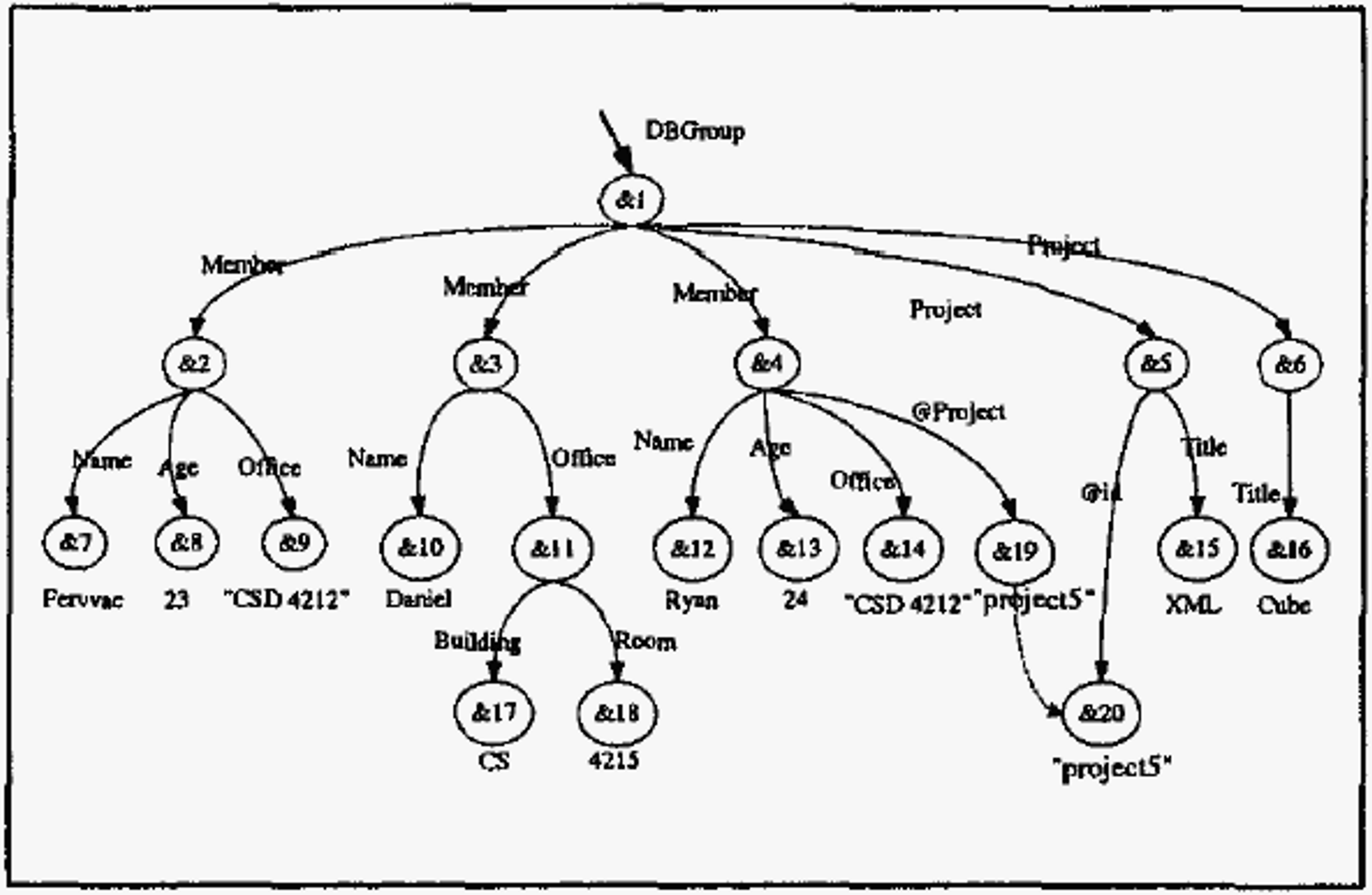
Node Oriented Approach

XML Document



```
<DBGroup>
   <Member>
      <Name> Fervvac </Name> <Age> 23 </Age>
      <Office> CSD 4212 </Office>
   </Member>
   <Member>
      <Name> Daniel </Name>
      <Office>
         <Building> CS </Building> <Room> 4215 </Room>
      </Office>
   </Member>
   <Member Project=105>
      <Name> Ryan </Name> <Age> 24 </Age>
      <Office> CSD 4212 </Office>
   </Member>
   <Project id = 105> <Title> XML </Title> </Project>
   <Project> <Title> Cube </Title> </Project>
</DBGroup>
```

Data Graph


DBGroup
&1
Member
Member
Member
Project
Project
&2
&3
&4
&5
&6
Name
Age
Office
Name
Office
Name
Age
Office
@Project
@id
Title
Title
&7
&8
&9
&10
&11
&12
&13
&14
&19
&15
&16
Peruvac
23
"CSD 4212"
Daniel
Ryan
24
"CSD 4212"
"project5"
XML
Cube
Building
Room
&17
&18
&20
CS
4215
"project5"

Key Terms

- **ORDINAL**: The ordinal of an element is the order of this element among all siblings that share the same parent.

  Ex. The ordinals for the elements &4 and &5 are 3 and 1 respectively.

- A **LABEL-PATH** in an XML data graph is a dot separated sequence of edge labels.

  Ex. DBGroup.Member.Name.

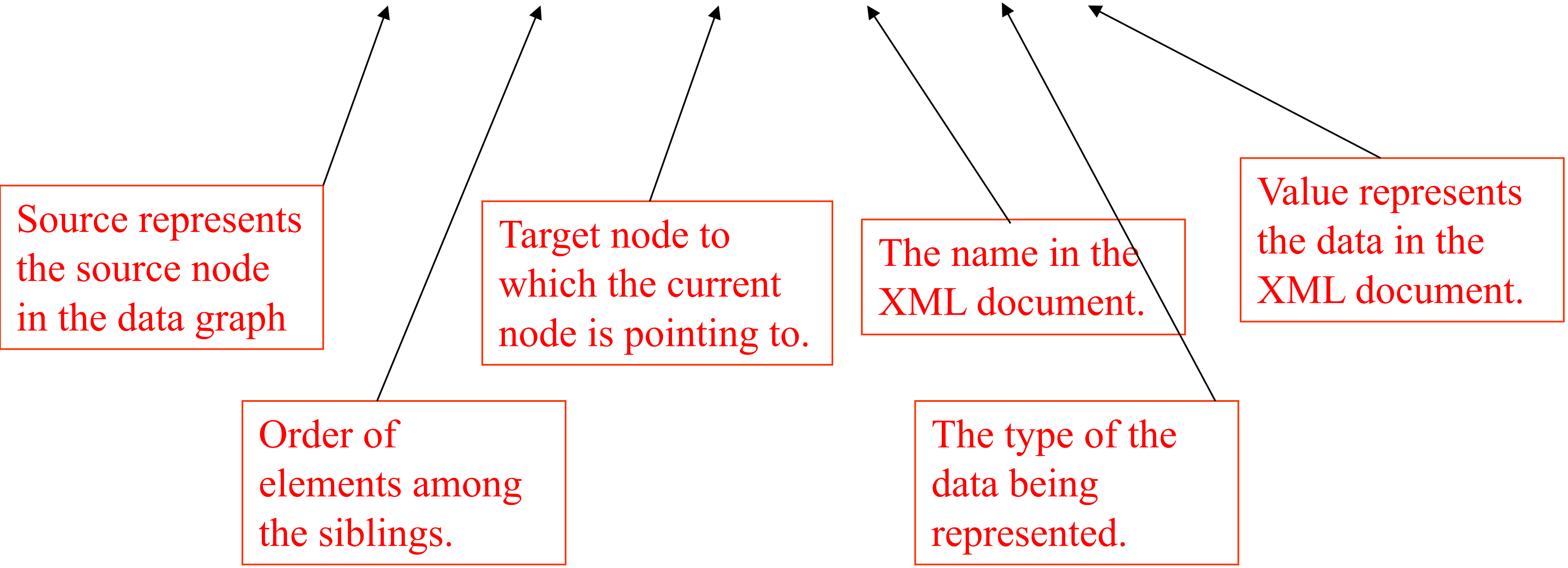
- A **DATA-PATH** is a dot-separated alternating sequence of element nodes.

  Ex. &1.&2.&7

Edge Approach

The Edge table can be represented as

Edge(Source,Ordinal,Target,Label,Flag,Value)



Source represents the source node in the data graph

Order of elements among the siblings.

Target node to which the current node is pointing to.

The name in the XML document.

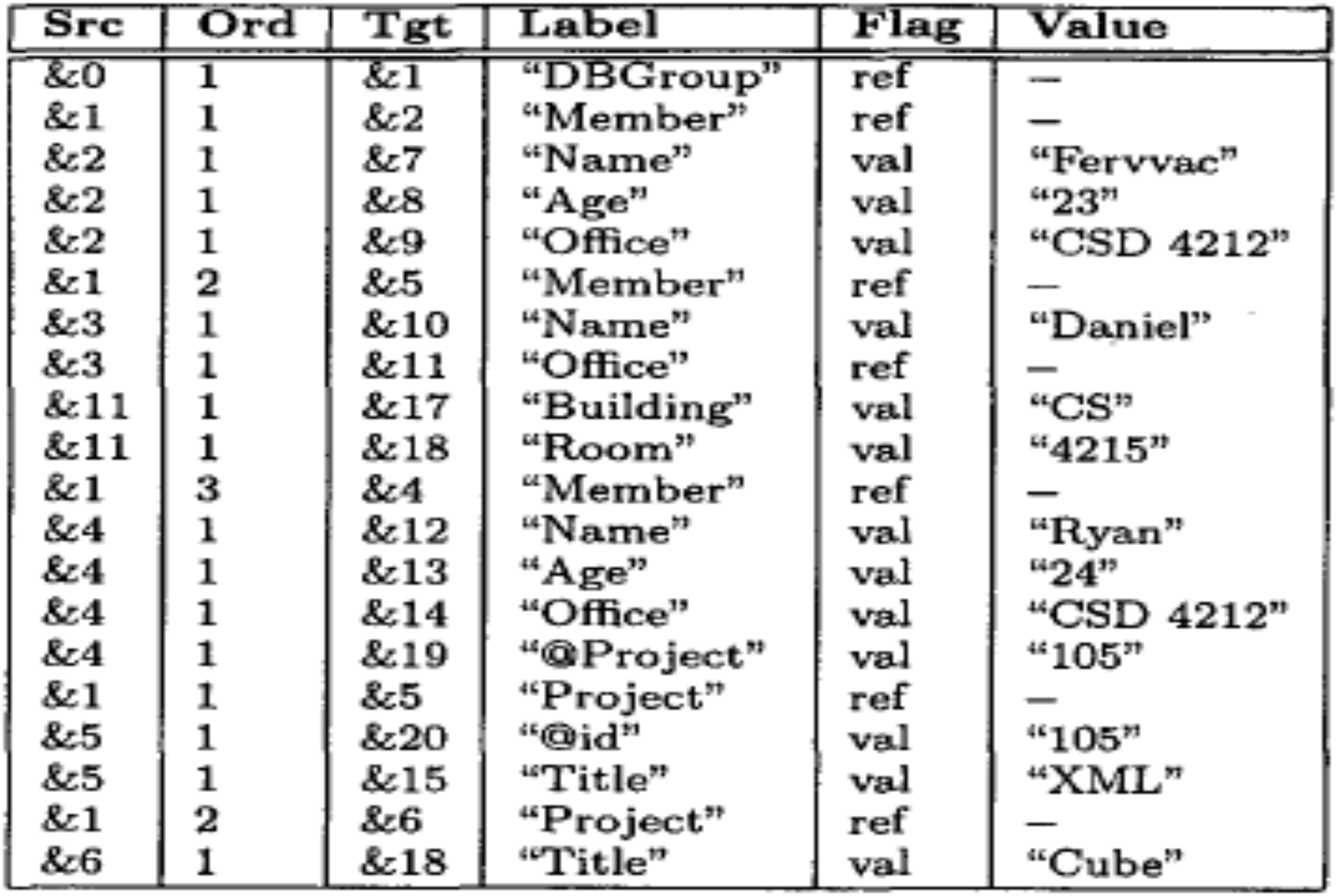
The type of the data being represented.

Value represents the data in the XML document.

Edge Approach…

- Edge is specified by two node identifiers Source and Target.
- The label attribute keeps the edge label of an edge.
- The Ordinal attribute records the ordinal of the edge among its siblings.
- A Flag value indicates ref or value.
- Value is the data stored in the XML document.

Data in Edge Table



Src	Ord	Tgt	Label	Flag	Value
&0	1	&1	"DBGroup"	ref	—
&1	1	&2	"Member"	ref	—
&2	1	&7	"Name"	val	"Fervvac"
&2	1	&8	"Age"	val	"23"
&2	1	&9	"Office"	val	"CSD 4212"
&1	2	&5	"Member"	ref	—
&3	1	&10	"Name"	val	"Daniel"
&3	1	&11	"Office"	ref	—
&11	1	&17	"Building"	val	"CS"
&11	1	&18	"Room"	val	"4215"
&1	3	&4	"Member"	ref	—
&4	1	&12	"Name"	val	"Ryan"
&4	1	&13	"Age"	val	"24"
&4	1	&14	"Office"	val	"CSD 4212"
&4	1	&19	"@Project"	val	"105"
&1	1	&5	"Project"	ref	—
&5	1	&20	"@id"	val	"105"
&5	1	&15	"Title"	val	"XML"
&1	2	&6	"Project"	ref	—
&6	1	&18	"Title"	val	"Cube"

Monet Approach

- Monet stores XML data in multiple tables.
- Partitions the Edge table on all possible label-paths (No of Tables = No of distinct label-paths)
- Tables are classified as
  - Element Node (Source, Target, Ordinal)

    The combination represents unique edge in XML data graph.
  - Text Node (ID, Value)

    The type of the value is implicit in the table name.

Data in Monet Tables

DBGroup→Member =

{<&1, &2, 1>, <&1, &3, 2>,<&1, &4, 3>}

DBGroup→Member→Name =

{<&2, &7, 1>, <&3, &10, 1>, <&4, &12, 1>}

DBGroup→Member→Name→String =

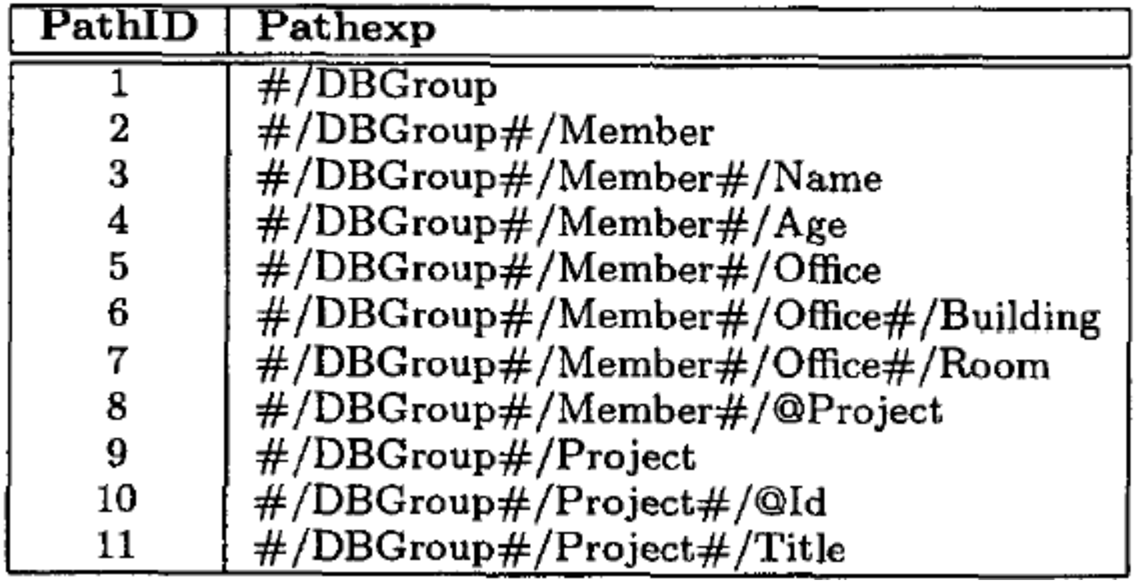
{<&7, Fervvac, 1>, <&7, Daniel, 1>, <&7, Ryan, 1>}

and so on….(18 tables)

XRel Approach

- Node oriented approach - maintains nodes individually.
- XRel Stores XML data in four tables:
  - Path (PathID, Pathexp)

    This table maintains the simple path expression identifier (PathID) and path expression(Pathexp).

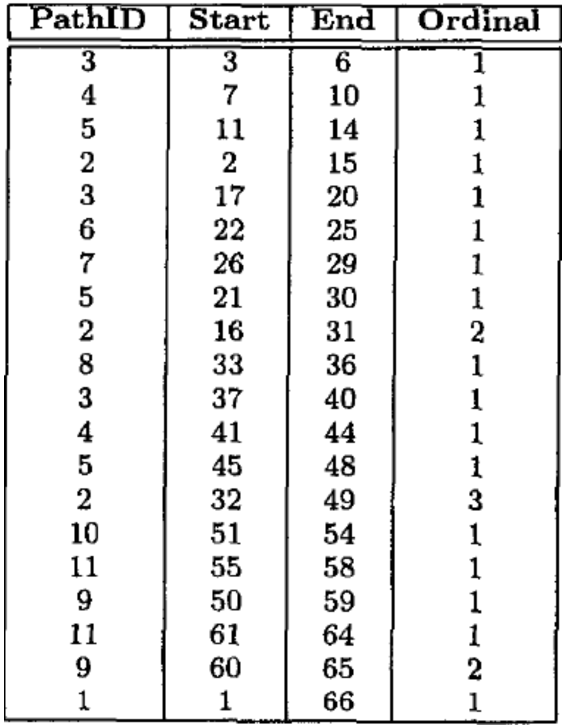


PathID	Pathexp
1	#/DBGroup
2	#/DBGroup#/Member
3	#/DBGroup#/Member#/Name
4	#/DBGroup#/Member#/Age
5	#/DBGroup#/Member#/Office
6	#/DBGroup#/Member#/Office#/Building
7	#/DBGroup#/Member#/Office#/Room
8	#/DBGroup#/Member#/@Project
9	#/DBGroup#/Project
10	#/DBGroup#/Project#/@Id
11	#/DBGroup#/Project#/Title

XRel Approach

Element (PathID, Start, End, Ordinal)

- This table contains the start position of a region, end position of a region for a given PathId.
- Region of node is the start and end positions of this node in XML Document



PathID	Start	End	Ordinal
3	3	6	1
4	7	10	1
5	11	14	1
2	2	15	1
3	17	20	1
6	22	25	1
7	26	29	1
5	21	30	1
2	16	31	2
8	33	36	1
3	37	40	1
4	41	44	1
5	45	48	1
2	32	49	3
10	51	54	1
11	55	58	1
9	50	59	1
11	61	64	1
9	60	65	2
1	1	66	1

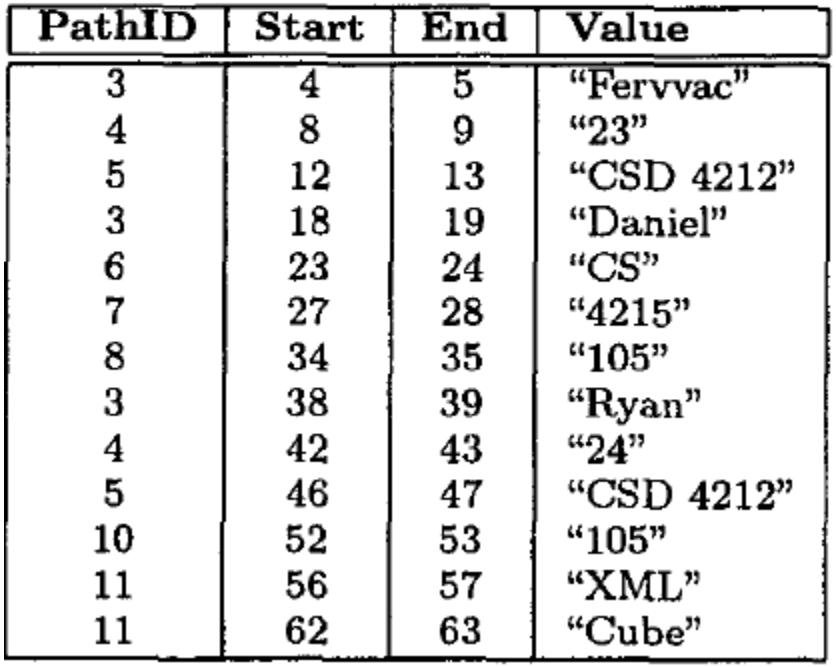
XRel Approach

- Text (PathID, Start, End, Value)

  This table contains the start position of a region, end position of a region, value of the element for a given PathId.

- Attribute (PathID, Start, End, Value)

  This table contains the start position of a region, end position of a region, value of the attribute for a given PathId

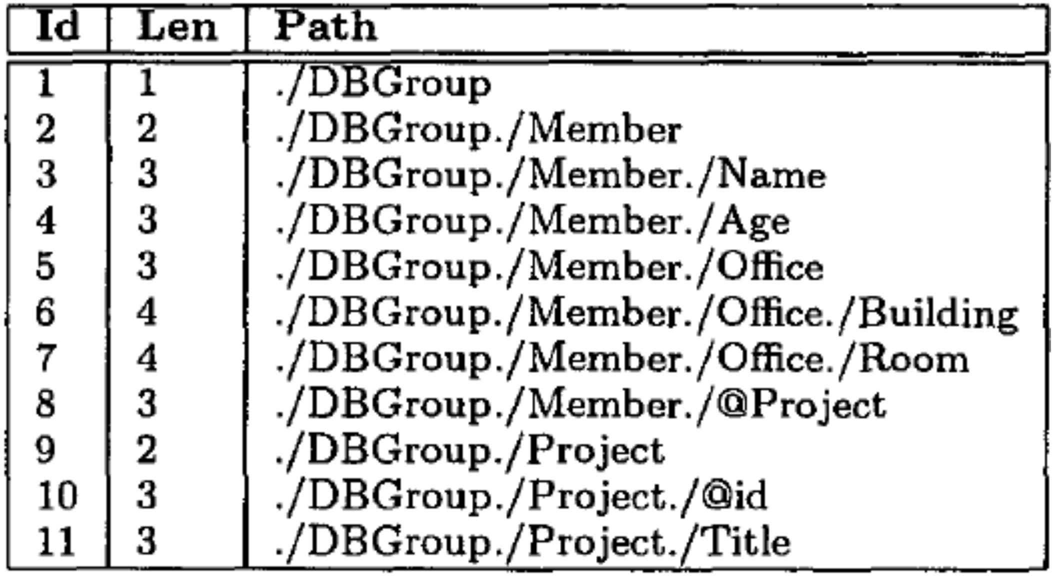


PathID	Start	End	Value
3	4	5	"Fervvac"
4	8	9	"23"
5	12	13	"CSD 4212"
3	18	19	"Daniel"
6	23	24	"CS"
7	27	28	"4215"
8	34	35	"105"
3	38	39	"Ryan"
4	42	43	"24"
5	46	47	"CSD 4212"
10	52	53	"105"
11	56	57	"XML"
11	62	63	"Cube"

XParent Approach

- Edge oriented approach
- XParent has four tables

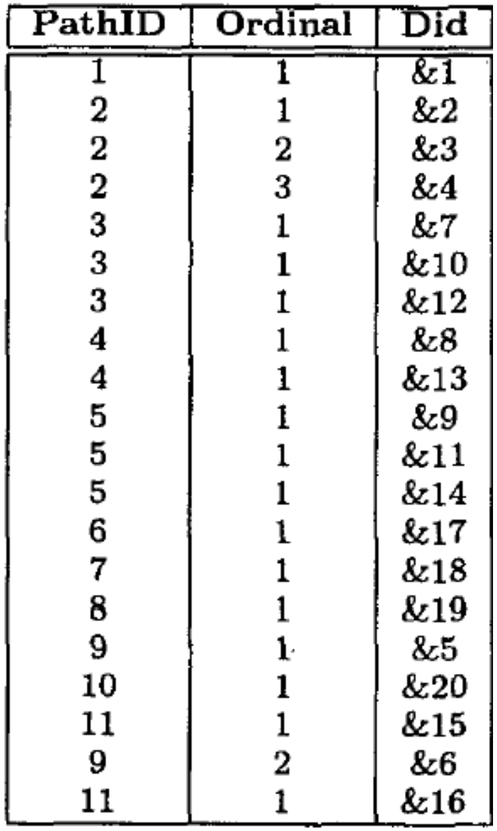
LabelPath (ID, Len, Path)



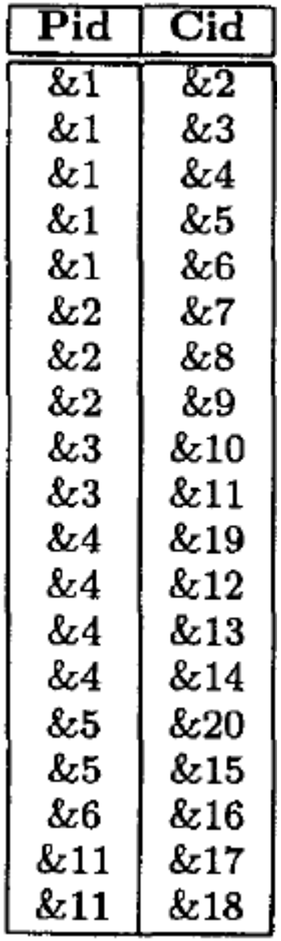
Id	Len	Path
1	1	./DBGroup
2	2	./DBGroup./Member
3	3	./DBGroup./Member./Name
4	3	./DBGroup./Member./Age
5	3	./DBGroup./Member./Office
6	4	./DBGroup./Member./Office./Building
7	4	./DBGroup./Member./Office./Room
8	3	./DBGroup./Member./@Project
9	2	./DBGroup./Project
10	3	./DBGroup./Project./@id
11	3	./DBGroup./Project./Title

XParent Approach

- DataPath (Pid, Cid)
- Element (pathID, Ordinal, Did)



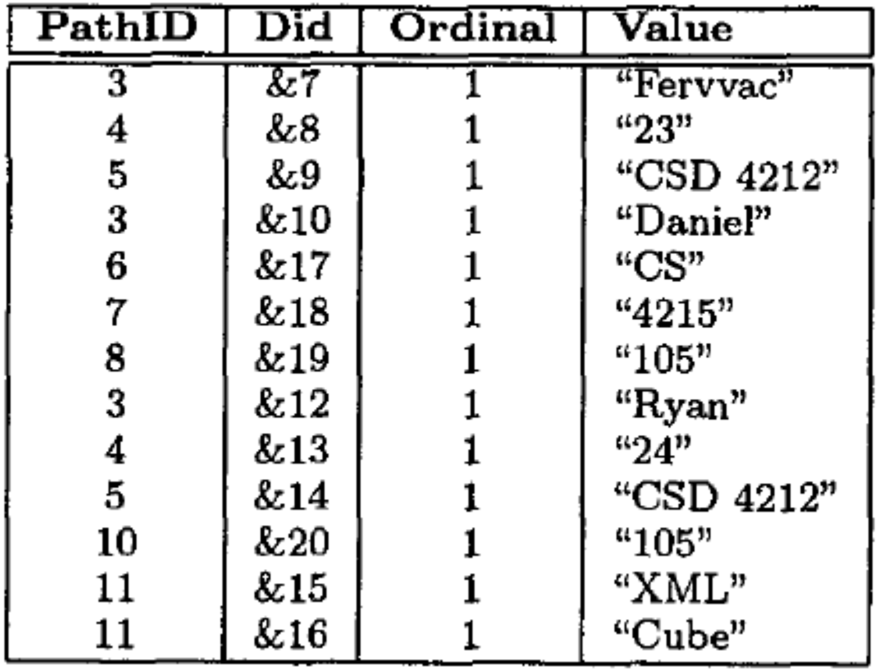
PathID	Ordinal	Did
1	1	&1
2	1	&2
2	2	&3
2	3	&4
3	1	&7
3	1	&10
3	1	&12
4	1	&8
4	1	&13
5	1	&9
5	1	&11
5	1	&14
6	1	&17
7	1	&18
8	1	&19
9	1	&5
10	1	&20
11	1	&15
9	2	&6
11	1	&16



Pid	Cid
&1	&2
&1	&3
&1	&4
&1	&5
&1	&6
&2	&7
&2	&8
&2	&9
&3	&10
&3	&11
&4	&19
&4	&12
&4	&13
&4	&14
&5	&20
&5	&15
&6	&16
&11	&17
&11	&18

XParent Approach

Data (PathID, Did, Ordinal, Value)



PathID	Did	Ordinal	Value
3	&7	1	“Fervvac”
4	&8	1	“23”
5	&9	1	“CSD 4212”
3	&10	1	“Daniel”
6	&17	1	“CS”
7	&18	1	“4215”
8	&19	1	“105”
3	&12	1	“Ryan”
4	&13	1	“24”
5	&14	1	“CSD 4212”
10	&20	1	“105”
11	&15	1	“XML”
11	&16	1	“Cube”

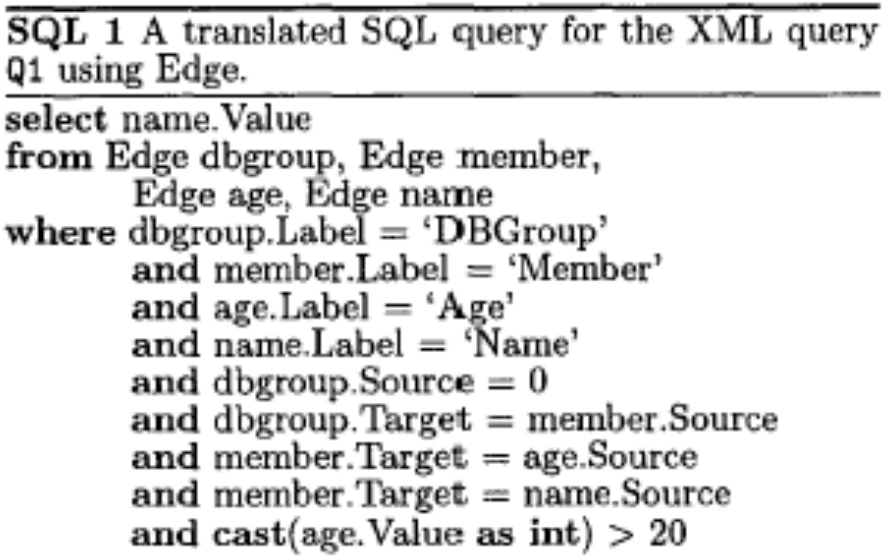
Querying XML Data

Select the names of all members whose ages are greater than 20.

- Xpath: /DBGroup/member[Age>20]/Name

Edge Query:

involves 6 selections and
3 equi joins



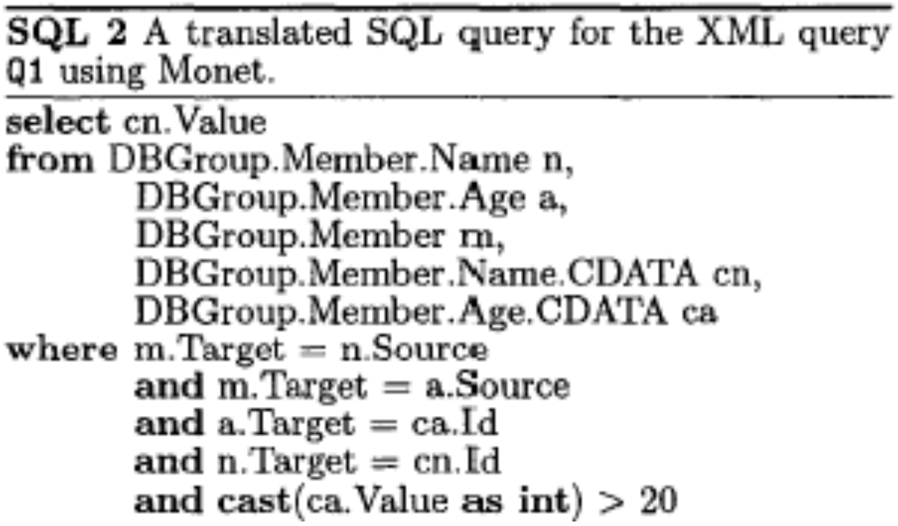
SQL 1 A translated SQL query for the XML query Q1 using Edge.

```
select name.Value
from Edge dbgroup, Edge member,
        Edge age, Edge name
where dbgroup.Label = 'DBGroup'
        and member.Label = 'Member'
        and age.Label = 'Age'
        and name.Label = 'Name'
        and dbgroup.Source = 0
        and dbgroup.Target = member.Source
        and member.Target = age.Source
        and member.Target = name.Source
        and cast(age.Value as int) > 20
```

Querying XML Data

Monet Query:

involves 1 selection and 4 joins

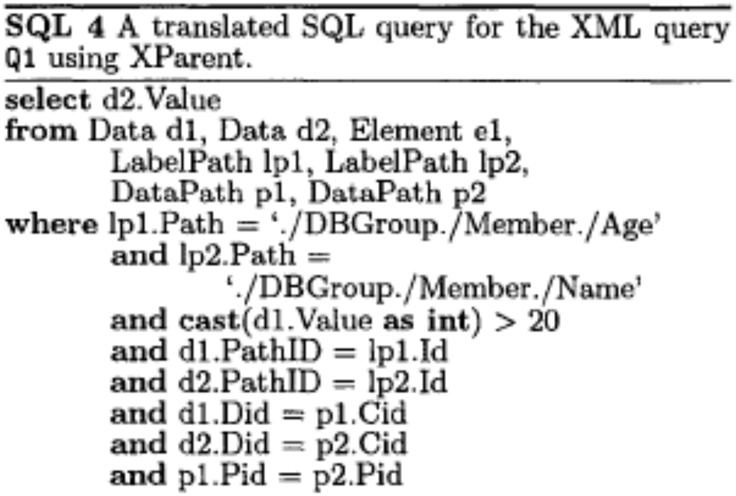


SQL 2 A translated SQL query for the XML query Q1 using Monet.

```
select cn.Value
from DBGroup.Member.Name n,
        DBGroup.Member.Age a,
        DBGroup.Member m,
        DBGroup.Member.Name.CDATA cn,
        DBGroup.Member.Age.CDATA ca
where m.Target = n.Source
        and m.Target = a.Source
        and a.Target = ca.Id
        and n.Target = cn.Id
        and cast(ca.Value as int) > 20
```

Xparent Query:

Involves 3 selections and 5 equi joins

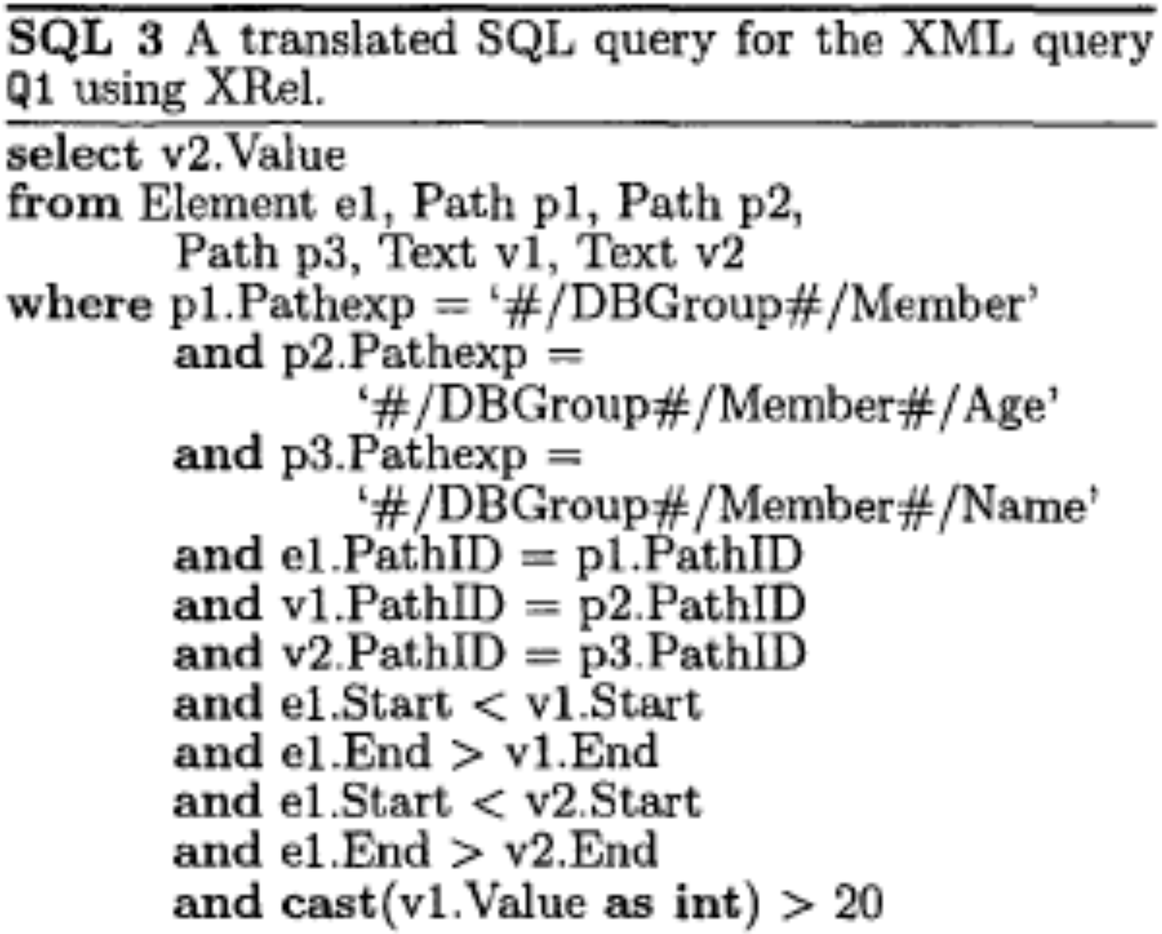


SQL 4 A translated SQL query for the XML query Q1 using XParent.

```
select d2.Value
from Data d1, Data d2, Element e1,
        LabelPath lp1, LabelPath lp2,
        DataPath p1, DataPath p2
where lp1.Path = './DBGroup./Member./Age'
        and lp2.Path =
                './DBGroup./Member./Name'
        and cast(d1.Value as int) > 20
        and d1.PathID = lp1.Id
        and d2.PathID = lp2.Id
        and d1.Did = p1.Cid
        and d2.Did = p2.Cid
        and p1.Pid = p2.Pid
```

Querying XML Data

XRel Query - involves 4 selections and 7 joins



SQL 3 A translated SQL query for the XML query Q1 using XRel.

```
select v2.Value
from Element e1, Path p1, Path p2,
          Path p3, Text v1, Text v2
where p1.Pathexp = '#/DBGroup#/Member'
          and p2.Pathexp =
                    '#/DBGroup#/Member#/Age'
          and p3.Pathexp =
                    '#/DBGroup#/Member#/Name'
          and e1.PathID = p1.PathID
          and v1.PathID = p2.PathID
          and v2.PathID = p3.PathID
          and e1.Start < v1.Start
          and e1.End > v1.End
          and e1.Start < v2.Start
          and e1.End > v2.End
          and cast(v1.Value as int) > 20
```

Conclusions

- XRel and XParent outperform Edge
- XRel and XParent outperform Edge in complex queries.
- Edge performs better when using simple queries.
- Label-paths help in reducing querying time.