Chapter 23: XML

Database System Concepts, 6th Ed.

XML

XML 데이터 구조

XML 문서 스키마

질의 및 변환

```
<university>
    <department>
        <dept_name> Comp. Sci. </dept_name>
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <department>
        <dept_name> Biology </dept_name>
        <building> Watson </building>
        <budget> 90000 </budget>
    </department>
    <course>
        <course_id> CS-101 </course_id>
        <title> Intro. to Computer Science </title>
        <dept_name> Comp. Sci. </dept_name>
        <credits> 4 </credits>
    </course>
    <course>
        <course_id> BIO-301 </course_id>
        <title> Genetics </title>
        <dept_name> Biology </dept_name>
        <credits> 4 </credits>
    </course>
    <instructor>
        <IID> 10101 </IID>
        <name> Srinivasan </name>
        <dept_name> Comp. Sci. </dept_name>
        <salary> 65000 </salary>
    </instructor>
    <instructor>
        <IID> 83821 </IID>
        <name> Brandt </name>
        <dept_name> Comp. Sci. </dept_name>
        <salary> 92000 </salary>
    </instructor>
    <teaches>
        <IID> 10101 </IID>
        <course_id> CS-101 </course_id>
    </teaches>
    <teaches>
        <IID> 83821 </IID>
        <course_id> CS-101 </course_id>
    </teaches>
    <teaches>
        <IID> 76766 </IID>
        <course_id> BIO-301 </course_id>
    </teaches>
</university>
```

```
<!DOCTYPE university [
    <!ELEMENT university (
    (department|course|instructor|teaches)+)>
    <!ELEMENT department ( dept name, building, budget)>
    <!ELEMENT course ( course id, title, dept name, credits)>
    <!ELEMENT instructor (IID, name, dept name, salary)>
    <!ELEMENT teaches (IID, course id)>
    <!ELEMENT dept name( #PCDATA )>
    <!ELEMENT building( #PCDATA )>
    <!ELEMENT budget( #PCDATA )>
    <!ELEMENT course id ( #PCDATA )>
    <!ELEMENT title ( #PCDATA )>
    <!ELEMENT credits( #PCDATA )>
    <!ELEMENT IID( #PCDATA )>
    <!ELEMENT name( #PCDATA )>
    <!ELEMENT salary( #PCDATA )>
]>
```

```
<university-3>
    <department dept_name="Comp. Sci.">
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <department dept_name="Biogoly">
        <building> Watson </building>
        <budget> 90000 </budget>
    </department>
    <course course_id="CS-101" dept_name="Comp. Sci."
                               instructors="10101 83821">
        <title> Intro. to Computer Science </title>
        <credits> 4 </credits>
    </course>
    <course course_id="BIO-301" dept_name="Biology"
                               instructors="76766">
        <title> Genetics </title>
        <credits> 4 </credits>
    </course>
    <instructor IID="10101" dept_name="Comp. Sci.">
        <name> Srinivasan </name>
        <salary> 65000 </salary>
    </Instructor>
    <instructor IID="83821" dept_name="Comp. Sci.">
        <name> Brandt </name>
        <salary> 72000 </salary>
    </Instructor>
    <instructor IID="76766" dept_name="Biology">
        <name> Crick </name>
        <salary> 72000 </salary>
    </Instructor>
</university-3>
```

University DTD with ID and IDREF attribute types.

```
<!DOCTYPE university-3 [
    <!ELEMENT university-3 ( (department|course|instructor)+)>
    <!ELEMENT department ( building, budget )>
    <!ATTLIST department
        dept_name ID #REQUIRED >
    <!ELEMENT course (title, credits )>
    <!ATTLIST course
        course_id ID #REQUIRED
        dept_name IDREF #REQUIRED
        instructors IDREFS #IMPLIED >
    <!ELEMENT instructor ( name, salary )>
    <!ATTLIST instructor
        IID ID #REQUIRED
        dept_name IDREF #REQUIRED >
    . . . declarations for title, credits, building,
        budget, name and salary . . .
]>
```

XML Data의 질의 및 변환

임의의 XML 스키마 정보를 다른 스키마로 바꾸기

XML 데이터에 관한 질의

위의 두 가지 연산은 매우 관련이 깊으며, 같은 툴에 의하여 처리될 수 있다.

표준 XML querying/translation 언어

XPath

- 경로식(path expression)으로 이루어진 간단한 언어

XSLT

- 임의의 XML 형태에서 다른 XML 형태 혹은 XML 에서 HTML로의 번역(변환)을 지원하는 언어

XQuery

- 다양한 기능을 가진 XML 질의 언어

XML Data의 트리 모델

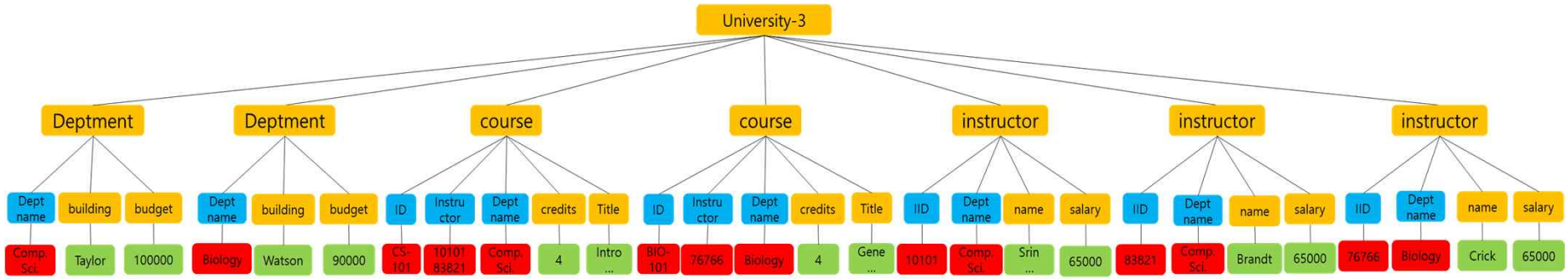
질의/변환 언어는 XML 데이터의 트리 모델을 이용한다.

임의의 XML 문서는 엘리먼트와 애트리뷰트를 나타내는 노드로 이루어진 트리 구조로 나타낼 수 있다.

- 엘리먼트는 애트리뷰트 혹은 서브 엘리먼트를 자식 노드로 갖는다.
- 엘리먼트 내의 텍스트는 텍스트 노드라는 자식 노드로 표현된다.
- 자식 노드들은 XML 문서 내에서의 출현 순서에 따라 순서를 갖는다.
- 루트 노드를 제외한 모든 엘리먼트와 애트리뷰트 노드들은 단일의 부모 노드를 갖는다.
- 루트 노드는 문서의 루트 엘리먼트에 해당하는 단 하나의 지식 노드를 갖는다.

```
<university-3>
    <department dept_name="Comp. Sci.">
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <department dept_name="Biogoly">
        <building> Watson </building>
        <budget> 90000 </budget>
    </department>
    <course course_id="CS-101" dept_name="Comp. Sci."
                               instructors="10101 83821">
        <title> Intro. to Computer Science </title>
        <credits> 4 </credits>
    </course>
    <course course_id="BIO-301" dept_name="Biology"
                               instructors="76766">
        <title> Genetics </title>
        <credits> 4 </credits>
    </course>
```

```
    <instructor IID="10101" dept_name="Comp. Sci.">
        <name> Srinivasan </name>
        <salary> 65000 </salary>
    </Instructor>
    <instructor IID="83821" dept_name="Comp. Sci.">
        <name> Brandt </name>
        <salary> 72000 </salary>
    </Instructor>
    <instructor IID="76766" dept_name="Biology">
        <name> Crick </name>
        <salary> 72000 </salary>
    </Instructor>
</university-3>
```



XPath

Xpath 언어는 경로식을 이용하여 문서의 일부분을 선택하여 내는데 사용된다.

경로식은 "/" 로 구분된 일련의 시퀀스로 볼 수 있음.

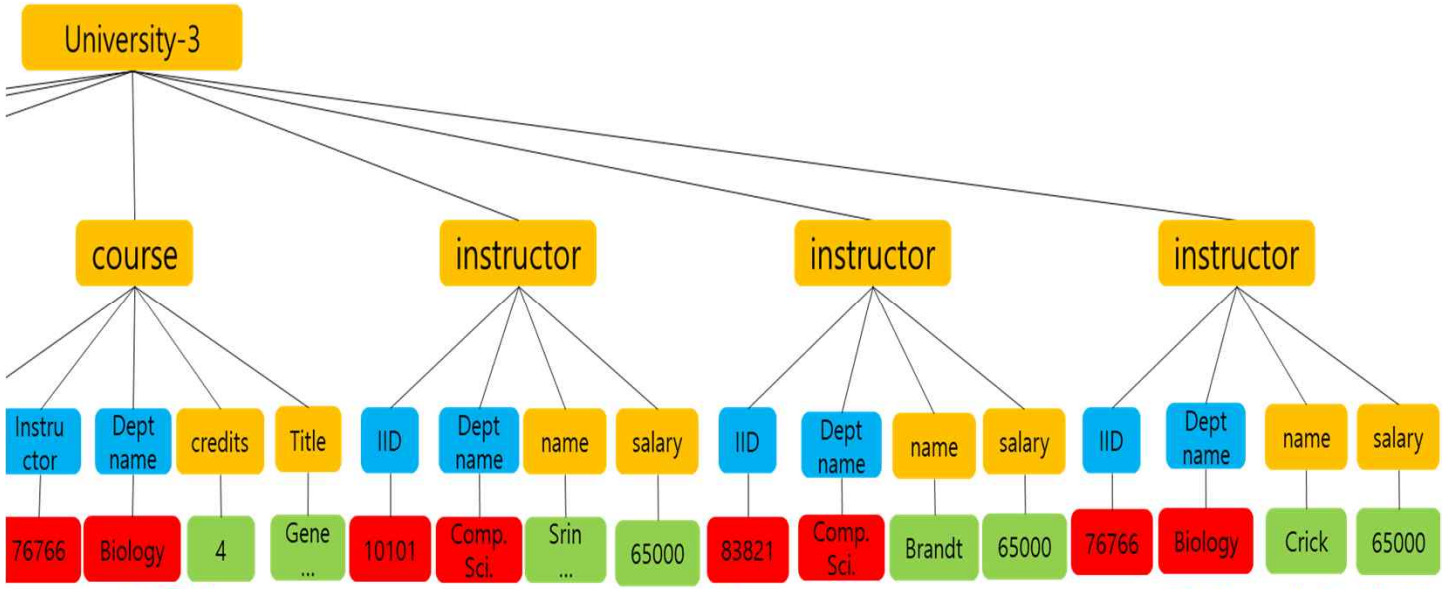
디렉토리 구조 상의 파일명 지정 방식과 같음

경로식의 결과: 지정된 경로식에 해당하는 엘리먼트/애트리뷰트를 포함한 값들의 집합

E.g. /university-3/instructor/name 의 결과:

<name>Srinivasan</name>
<name>Brandt</name>
<name>Crick</name>

E.g. /university-3/instructor/name/text( ) : 테그 정보가 빠진 위와 동일한 결과를 얻음.



XPath (Cont.)

처음 "/" 는 최상위 태그 위에 존재하는 문서의 루트를 나타낸다

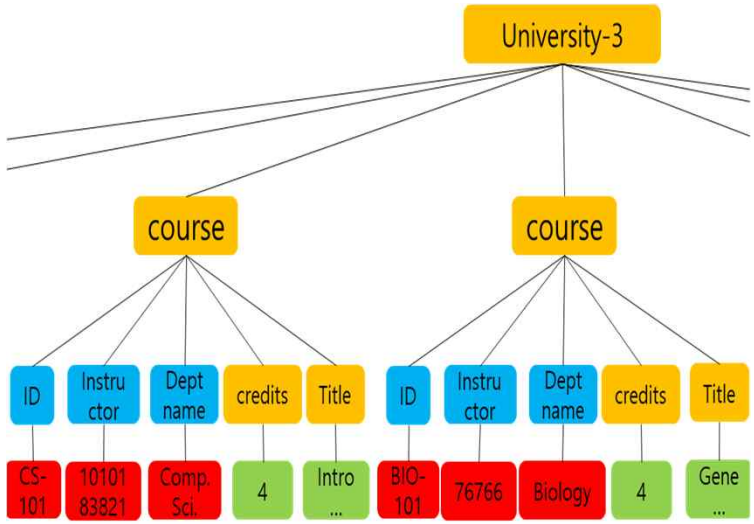
경로식은 왼쪽에서 오른쪽 순서로 평가된다.

각 단계에서는 직전 단계에서 얻은 결과의 집합에 대하여 연산을 수행하게 된다.

각 단계에서 괄호 [ ] 안에 선택 조건을 기술할 수 있다.

/university-3/course[credits >= 4] : 4 이상의 credits 서브 엘리먼트 값을 갖는 모든 course 엘리먼트를 결과로 얻는다.

/university-3/course[credits] : credits 를 서브 엘리먼트로 갖는 모든 course 엘리먼트를 결과로 얻는다.



각 애트리뷰는 "@" 를 이용하여 참조할 수 있다.

/university-3/course[credits >= 4]/@course_id

- 4 이상의 credits 서브 엘리먼트 값을 갖는 모든 course 엘리먼트의 course_id 애트리뷰트를 결과로 얻는다.

IDREF 속성은 위의 방식으로 참조하지 못한다 (뒤에 다시 기술함)

Functions in XPath

Xpath는 다양한 함수를 제공한다

각 경로의 마지막에 count() 함수를 붙여, 경로 해석의 결과로 얻어진 모든 엘리먼트의 개수를 셀 수 있다.

- E.g. /university-2/instructor[count(./teaches/course)> 2]:
- 2 이상의 course를 가르치고 있는 instructor 엘리먼트를 결과로서 반환한다.

자식 노드의 순서를 반환하는 함수도 있다

And, or, not() 등을 선택 조건 기술에 사용할 수 있다.

IDREFs 타입을 갖는 애트리뷰트 값은 id() 함수를 이용하여 참조 가능하다

id() 함수는 IDREFS 와 같은 참조 타입의 집합 값에도 적용가능하며, 동시에 blank로 나누어 열거된 참조 타입들에도 적용가능하다.

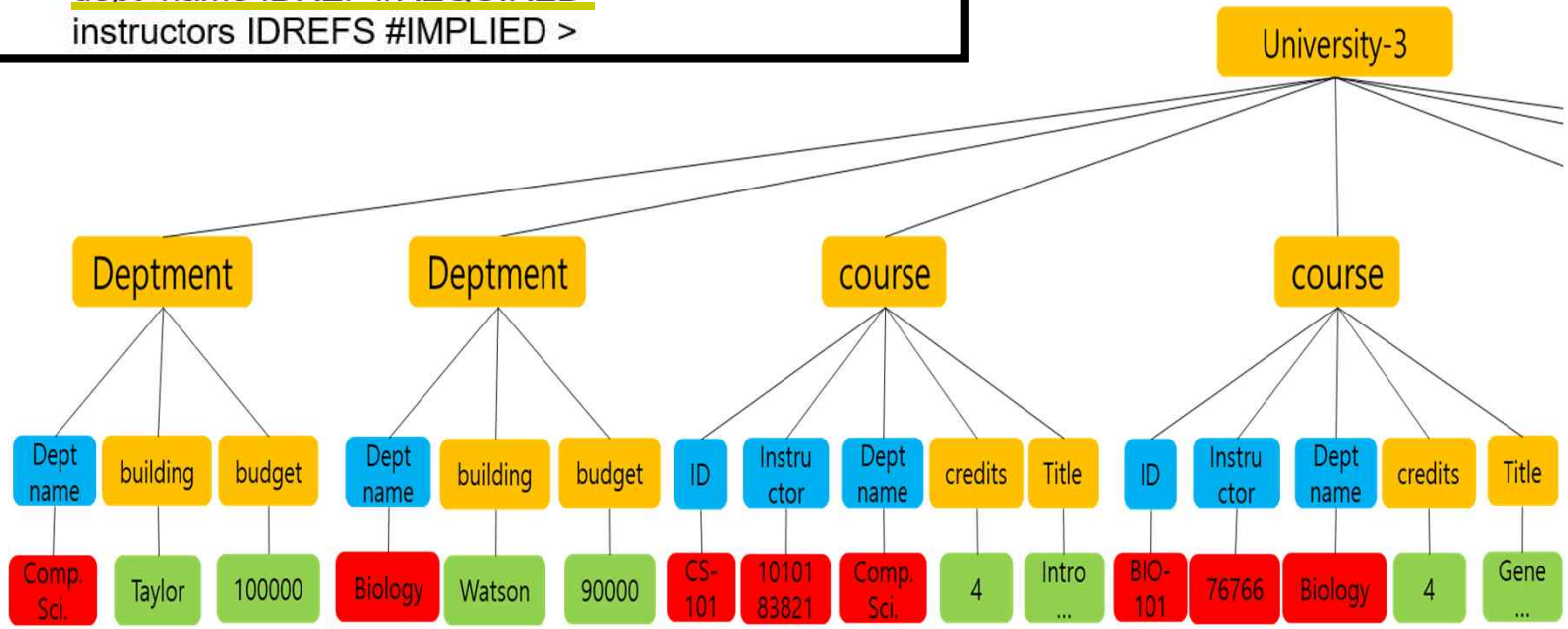
E.g. /university-3/course/id(@dept_name)

- course 엘리먼트의 dept_name 애트리뷰트에 의하여 참조된 모든 department 엘리먼트를 결과로 반환한다.

Functions in XPath

E.g. /university-3/course/id(@dept_name): course 엘리먼트의 dept_name 애트리뷰트에 의하여 참조된 모든 department 엘리먼트를 결과로 반환한다.

```
University DTD with ID and IDREF attribute types.
<!DOCTYPE university-3 [
    <!ELEMENT university-3 ( (department|course|instructor)+)>
    <!ELEMENT department ( building, budget )>
    <!ATTLIST department
        dept_name ID #REQUIRED >
    <!ELEMENT course (title, credits )>
    <!ATTLIST course
        course_id ID #REQUIRED
        dept_name IDREF #REQUIRED
        instructors IDREFS #IMPLIED >
```



More XPath Features

"|" 연산자는 합집합 연산을 나타낸다.

E.g. /university-3/course[@dept name="Comp. Sci"] |
/university-3/course[@dept name="Biology"]

- "Comp. Sci." 와 "Biology" 두 course의 합집합 결과를 반환한다.
- 그러나 "|" 연산자는 다른 연산자에 네스팅된 구조로 사용될 수 없다.

"//" 연산자는 다단계 노드를 한번에 스킵하는데 사용될 수 있다.

E.g. /university-3//name

- /university-3 노드의 임의의 하위 노드로 존재하는 name 엘리먼트를 반환한다.

경로식은 해당 노드로부터 상위 혹은 하위 노드로 자유로이 이동할 수 있다.

"//" 는 모든 자손 노드를 나타낸다.

".." 는 부모 노드를 나타낸다.

doc(name) 은 name에 기술된 문서의 루트 노드를 반환한다.

XQuery

XQuery 는 XML 데이터를 위한 일반 질의 언어이다.

World Wide Web Consortium (W3C)에 의하여 개발/표준화된 언어이다.

현재 교과서 내용은 2007년 1월 표준안에 근거하여 기술되었으나, 가장 최신 버전도 이 표준안과 크게 다르지 않다.

XQuery 기본 구문은 다음과 같다

for … let … where … order by …return …

for ⇔ SQL **from**
where ⇔ SQL **where**
order by ⇔ SQL **order by**
return ⇔ SQL **select**

let 임시 변수 선언에 사용됨.

FLWOR Syntax in XQuery

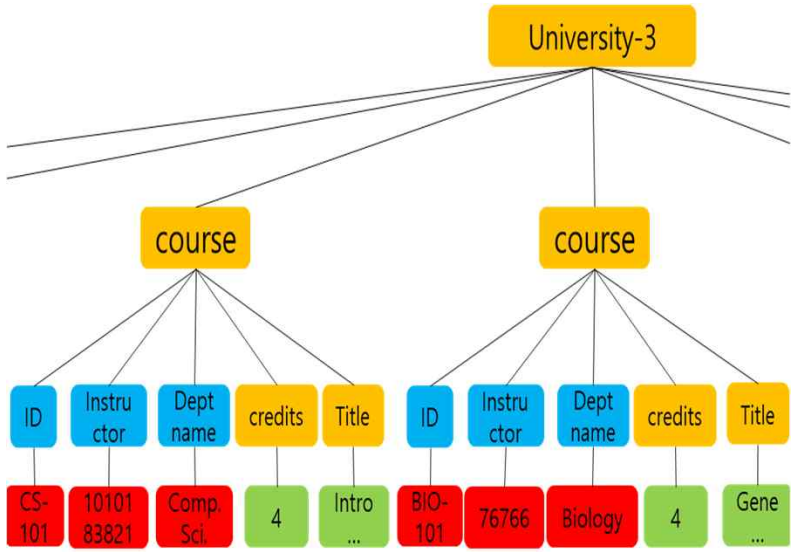
For 절에 XPath 표현식이 사용되며, for 절의 변수는 XPath 표현식에 의하여 반환되는 집합 값을 그 값으로 갖는다.

간단한 FLWOR 표현식의 예

credits > 3의 조건을 만족하는 모든 course를 찾아, 각 결과를 <course_id> .. </course_id> 의 태그로 묶어 기술하시오.

```
for  $x in /university-3/course
let   $courseId := $x/@course_id
where $x/credits > 3
return <course_id> { $courseId } </course id>
```

```
<course_id  course_id="CS-101"  />
<course_id  course_id="BIO-101" />
```



return 절에 기술된 항목들은 중괄호 { }로 둘러싸인 부분을 제외하고는 모두 일반 XML 텍스트로 인식된다. 단 중괄호 { }로 둘러싸인 부분은 평가되어 그 결과가 반환된다.

위 질의는 아래와 같이 간략히 기술될 수 있다:

```
for $x in /university-3/course[credits > 3]
 return <course_id> { $x/@course_id } </course_id>
```

조인

SQL의 경우와 유사하게 join 표현식을 기술할 수 있다.

```
for $c in /university/course,
    $i in /university/instructor,
    $t in /university/teaches
where $c/course_id= $t/course id and $t/IID = $i/IID
return <course_instructor> { $c $i } </course_instructor>
```

다음과 같이 간략히 기술할 수 있다:

```
for $c in /university/course,
    $i in /university/instructor,
    $t in /university/teaches[ $c/course_id= $t/course_id
                               and $t/IID = $i/IID]
return <course_instructor> { $c $i } </course_instructor>
```

```
<university>
    <department>
        <dept_name> Comp. Sci. </dept_name>
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <department>
        <dept_name> Biology </dept_name>
        <building> Watson </building>
        <budget> 90000 </budget>
    </department>
    <course>
        <course_id> CS-101 </course_id>
        <title> Intro. to Computer Science </title>
        <dept_name> Comp. Sci. </dept_name>
        <credits> 4 </credits>
    </course>
    <course>
        <course_id> BIO-301 </course_id>
        <title> Genetics </title>
        <dept_name> Biology </dept_name>
        <credits> 4 </credits>
    </course>
    <instructor>
        <IID> 10101 </IID>
        <name> Srinivasan </name>
        <dept_name> Comp. Sci. </dept_name>
        <salary> 65000 </salary>
    </instructor>
    <instructor>
        <IID> 83821 </IID>
        <name> Brandt </name>
        <dept_name> Comp. Sci. </dept_name>
        <salary> 92000 </salary>
    </instructor>
    <teaches>
        <IID> 10101 </IID>
        <course_id> CS-101 </course_id>
    </teaches>
    <teaches>
        <IID> 83821 </IID>
        <course_id> CS-101 </course_id>
    </teaches>
    <teaches>
        <IID> 76766 </IID>
        <course_id> BIO-301 </course_id>
    </teaches>
</university>
```

```
for $c in /university/course,
    $i in /university/instructor,
    $t in /university/teaches
where $c/course_id= $t/course id and $t/IID = $i/IID
return <course_instructor> { $c $i } </course_instructor>
```

```
<course_instructor>
  <course>
    <course-id> CS-101</course-id>
    …..
  </course>
  <instructor>
    <IID> 10101 </IID>
      ….
  </instructor>
</course_instructor>
<course_instructor>
  <course>
    <course-id> CS-101</course-id>
    …..
  </course>
  <instructor>
    <IID> 83821 </IID>
      ….
  </instructor>
</course_instructor>
```

중첩 질의

다음 질의는 플랫 구조를 갖는 university 정보를 네스팅 구조를 갖는 university-1 정보로 변환한다.

```
<university-1>
{    for $d in /university/department
     return <department>
                { $d/* }
                { for $c in /university/course[dept name = $d/dept name]
                  return $c }
            </department>
}
{    for $i in /university/instructor
     return  <instructor>
                { $i/* }
                { for $c in /university/teaches[IID = $i/IID]
                  return $c/course id }
             </instructor>
}
</university-1>
```

$c/* 는 $c 변수가 나타내는 모든 노드 (혹은 노드들의 집합)의 모든 자식을 의미한다.,

```
<university>
    <department>
        <dept_name> Comp. Sci. </dept_name>
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <department>
        <dept_name> Biology </dept_name>
        <building> Watson </building>
        <budget> 90000 </budget>
    </department>
    <course>
        <course_id> CS-101 </course_id>
        <title> Intro. to Computer Science </title>
        <dept_name> Comp. Sci. </dept_name>
        <credits> 4 </credits>
    </course>
    <course>
        <course_id> BIO-301 </course_id>
        <title> Genetics </title>
        <dept_name> Biology </dept_name>
        <credits> 4 </credits>
    </course>
    <instructor>
        <IID> 10101 </IID>
        <name> Srinivasan </name>
        <dept_name> Comp. Sci. </dept_name>
        <salary> 65000 </salary>
    </instructor>
    <instructor>
        <IID> 83821 </IID>
        <name> Brandt </name>
        <dept_name> Comp. Sci. </dept_name>
        <salary> 92000 </salary>
    </instructor>
    <teaches>
        <IID> 10101 </IID>
        <course_id> CS-101 </course_id>
    </teaches>
    <teaches>
        <IID> 83821 </IID>
        <course_id> CS-101 </course_id>
    </teaches>
    <teaches>
        <IID> 76766 </IID>
        <course_id> BIO-301 </course_id>
    </teaches>
</university>
```

```
<university-1>
{   for $d in /university/department
    return <department>
              { $d/* }
              { for $c in /university/course[dept name = $d/dept
name]
                return $c }
           </department>
}
{   for $i in /university/instructor
    return  <instructor>
              { $i/* }
              { for $c in /university/teaches[IID = $i/IID]
                return $c/course id }
            </instructor>
}
</university-1>
```

```
<university-1>
  <department>
    <dept_name> Comp. Sci. </dept_name>
    <building> Taylor </building>
    <budget> 100000 </budge>
    <course>
        <course-id> CS-101</course-id>
        …..
    </course>
  </department>
  <department>
    <dept_name> Biology </dept_name>
    <building> Watson </building>
    <budget> 90000 </budge>
    <course>
        <course-id> BIO-301</course-id>
        …..
    </course>
  </department>
```

```
  <instructor>
      <IID> 10101 </IID>
      <name> Srinivasan </name>
      <dept_name> Comp. Sci.  </dept_name>
      <salary> 65000 </salary>
      <course-id> CS-101</course-id>
  </instructor>
  <instructor>
      <IID> 83821 </IID>
      <name> Brandt </name>
      <dept_name> Comp. Sci.  </dept_name>
      <salary> 92000 </salary>
      <course-id> CS-101</course-id>
   </instructor>
</university-1>
```

정렬

order by 절을 이용하여 결과를 정렬하여 얻을 수 있다.

```
 for $i in /university/instructor
order by $i/name
return <instructor> { $i/* } </instructor>
```

order by $i/name **descending** : 내림차순으로 정렬

네스팅의 다단계에 걸쳐 정렬하는 경우의 예:

```
<university-1> {
  for $d in /university/department
  order by $d/dept name
  return
      <department>
        { $d/* }
        { for $c in /university/course[dept name = $d/dept name]
          order by $c/course id
          return <course> { $c/* } </course> }
      </department>
} </university-1>
```

함수와 다른 특성

다양한 내장 함수를 지원하며, 동시에 사용자 정의 함수를 지원한다.

```
declare function local:dept_courses($iid as xs:string)
        as element(course)*
{
    for $i in /university/instructor[IID = $iid],
        $c in /university/courses[dept_name = $i/dept name]
    return $c
}
```

함수의 입력 값과 반환값의 타입 지정은 선택적이므로 생략가능하다.

* (예: decimal*) 는 해당 타입을 가진 값들의 집합을 나타낸다.

XQuery 에서 If-then-else 구문을 사용할 수 있다.

Application Program Interface

두 가지의 표준 API가 존재한다:

SAX (Simple API for XML)

- 응용 프로그램 개발자는 각 이벤트에 대하여 핸들러 함수를 생성/등록한다. SAX 파서가 문서를 읽을 때 이벤트가 발생하면 핸들러 함수가 호출되어, 작업을 수행한다.
  - E.g. start of element, end of element

DOM (Document Object Model)

- **XML** 데이터는 트리 구조로 파싱된다
- DOM 트리 탐색을 위한 다양한 함수가 제공된다.
- E.g.: Java DOM API provides Node class with methods
  getParentNode( ), getFirstChild( ), getNextSibling( )
  getAttribute( ), getData( ) (for text node)
  getElementsByTagName( ), …
- DOM tree 업데이트 함수도 제공된다.

XML 자료의 publishing과 shredding

university-1 스키마에 적용한 예의 일부

department(*id*, *dept_name*, *building*, *budget*)
course(*parent id*, *course_id*, *title*, *credits*)

Publishing: 릴레이션 데이터를 XML 포맷으로 변환하는 작업

Shredding: XML 문서 내용을 분해하여, 하나 이상의 릴레이션에 삽입할 튜플들로 변환하는 작업

XML을 지원하는 상용 데이터베이스는 자동 publishing/shredding 기능을 제공한다.

또한 많은 시스템에서는 **xml** data type 으로 원래의 XML 데이터를 그대로 저장하는 기능을 제공한다. 또한 효율을 고려하여 특수한 내부 데이터 구조와 인덱스 구조를 사용하고 있다.

End of Chapter 23

Database System Concepts, 6th Ed.