관계 데이터로부터 재귀적 구조의 XML 값을 생성하는 SQL 함수

(An SQL Function for the Construction of Recursively Structured XML values from the Relational Data)

박 성 철 † 박 영 철 ††
(Sung Chul Park) (Young Chul Park)

요 약 SQL:2003 표준은 SQL 질의의 결과를 XML 값으로 생성하는 SQL/XML 출판 함수들을 지원하지만 재귀적 질의의 결과를 재귀적 구조의 XML 값으로 생성하는 SQL/XML 출판 함수를 제공하지 않는다. 그러므로, 주어진 조건에 의해 직간접으로 연결된 관계 투플들에 대하여, 적절한 SQL/XML 출판 함수들을 이용하여 그들의 내용과 그들 간의 연결 관계를 XML 값으로 생성하려면, 중첩 SQL 질의를 작성하여야 한다. 그러나 그 질의의 작성은 그 투플들의 연결 깊이를 알 수 있지만 그 깊이가 깊은 경우에는 쉽지 않으며, 그 연결 깊이를 알 수 없는 경우에는 불가능하다. 본 논문은 그 문제를 해결하기 위하여, 재귀적 질의의 결과를 재귀적 구조의 XML 값으로 생성하는 새로운 SQL 함수 XMLNEST를 제안한다.

키워드 : SQL/XML 출판 함수, 재귀적 질의, XMLNEST

Abstract SQL:2003 standard provides SQL/XML publishing functions to publish the result of an SQL query as XML values but it does not provide any SQL/XML publishing function that can publish the result of a recursive query as recursively structured XML values. Therefore, for the relational tuples which are connected directly or indirectly according to given conditions, to publish both the contents of them and the relationship among them as XML values with the use of appropriate SQL/XML publishing functions, we have to write a nested SQL query. Writing that query, however, is not easy provided that the depth of the connections is deep even if we know the depth of them and is not possible once the depth of the connections is not known in advance. In order to resolve that problem, we propose a new SQL function XMLNEST that can publish the result of a recursive query as recursively structured XML values.

Key words : SQL/XML publishing function, recursive query, XMLNEST

1. 서 론

데이터베이스에 저장된 데이터를 XML로 변환하고자 하는 요구에 따라 그와 관련된 연구들이 진행되어 왔다[1,2]. SQL:2003 표준[3]의 XML 관련 명세인 SQL/XML은 SQL 질의의 결과를 XML 값으로 생성할 수 있게 하는 XMLELEMENT, XMLATTRIBUTES, XMLAGG, XMLFOREST, XMLCONCAT 등의 SQL/XML 출판 함수들(publishing functions)[4]을 SQL 함수들로 정의함으로써 데이터베이스 언어인 SQL을 통해 XML 값을 생성하는 방법들을 제공한다.

그러나 SQL:2003 표준은 SQL:1999 표준[5]의 재귀적 질의에 사용되어 그 질의에 의해 식별된 투플들을 재귀적 구조의 XML 값으로 생성하는 SQL/XML 출판 함수를 제공하지 않는다. 그러므로, 주어진 조건에 의해 직간접으로 연결된 관계 투플들에 대하여, 적절한 SQL/XML 출판 함수들을 이용하여 그들의 내용과 그들 간의 연결 관계를 XML 값으로 생성하려면 중첩 SQL 질의를 작성하여야 한다.

예제 1. 테이블 EMPLOYEE를 대상으로 사원번호가 10인 사원과 21인 사원 그리고 그 사원들의 부하사원들 그리고 그 부하사원들의 부하사원들을 대상으로 그 사원들의 정보와 그 사원들 간의 연결 관계를 XML 값으로 생성하시오.

예제 1을 비롯하여 본 논문에서 제시하는 SQL 질의들에 의해 참조되는 테이블 EMPLOYEE의 구조와 그 테이블의 투플들은 그림 1과 같다.

employee_id	first_name	last_name	salary	supervisor_id	department_id
10	CHRISTINE	HAAS	52750	NULL	1
11	ELIZABATH	QUEEN	32750	10	1
13	JIMAE	IL	37650	14	1
14	CHANHO	PARK	52850	16	2
15	SY	LEE	52050	16	3
16	DONG	SUN	50750	10	2
17	WOL	MAE	45620	14	2
21	BH	KIM	28880	NULL	4
23	JONGBUM	LEE	38200	21	4
24	BIG	CHOI	25480	21	4

그림 1 테이블 EMPLOYEE

· 이 논문은 제35회 추계학술대회에서 '관계 데이터로부터 재귀적 구조의 XML을 생성하는 SQL 함수의 제안'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : 경북대학교 전자전기컴퓨터학부
scpark@knu.ac.kr
†† 종신회원 : 경북대학교 전자전기컴퓨터학부 교수
ycpark@knu.ac.kr

논문접수 : 2008년 12월 15일
심사완료 : 2009년 2월 18일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제4호(2009.4)

예제 1을 위해 SQL/XML 출판 함수들을 이용하여 작성한 질의는 그림 2와 같다. 그림 1의 테이블 EMPLOYEE에 대한 그림 2의 질의의 결과투플들은 그림 3과 같다. 그림 3의 XML 값들의 구조를 XML 스키마 그래프(schema graph)로 표현하면 그림 4와 같다.

```
SELECT XMLELEMENT ("EMPLOYEE", XMLATTRIBUTES (employee_id AS "id"),
       XMLFOREST (first_name as "FIRSTNAME", last_name as "LASTNAME", salary as SALARY, supervisor_id as "MANAGER",
          (SELECT XMLAGG (
                    XMLELEMENT ("EMPLOYEE", XMLATTRIBUTES (employee_id AS "id"),
                       XMLFOREST (first_name as "FIRSTNAME", last_name as "LASTNAME", salary as SALARY,
                                                                    supervisor_id as "MANAGER",
                       (SELECT XMLAGG(
                          XMLELEMENT("EMPLOYEE", XMLATTRIBUTES(employee_id AS "id"),
                             XMLFOREST (first_name as "FIRSTNAME", last_name as "LASTNAME",
                                                    salary as SALARY, supervisor_id as "MANAGER")
                       ) )
                       FROM employee sub
                       WHERE sub.supervisor_id=sub.employee_id
                    ) as "SUBORDINATES" ) ) )
          FROM employee sub
          WHERE sub.supervisor_id = super.employee_id
       ) as "SUBORDINATES" ) ) as result
FROM employee super
WHERE employee_id=10 OR employee_id = 21
```

그림 2 예제 1을 위해 작성된 SQL 질의

```
result
<EMPLOYEE id="10">
  <FIRSTNAME>CHRISTINE</FIRSTNAME><LASTNAME>HAAS</LASTNAME>
  <SALARY>52750</SALARY><MANAGER/>
  <SUBORDINATES>
    <EMPLOYEE id = "11">
      <FIRSTNAME>ELIZABATH</FIRSTNAME><LASTNAME>QUEEN</LASTNAME>
      <SALARY>32750</SALARY><MANAGER>10</MANAGER>
    </EMPLOYEE>
    <EMPLOYEE id="16">
      <FIRSTNAME>DONG</FIRSTNAME><LASTNAME>SUN</LASTNAME>
      <SALARY>50750</SALARY><MANAGER>10</MANAGER>
      <SUBORDINATES>
        <EMPLOYEE id="15">
          <FIRSTNAME>SY</FIRSTNAME><LASTNAME>LEE</LASTNAME>
          <SALARY>52050</SALARY><MANAGER>16</MANAGER>
        </EMPLOYEE>
        <EMPLOYEE id="14">
          <FIRSTNAME>CHANHO</FIRSTNAME><LASTNAME>PARK</LASTNAME>
          <SALARY>52850</SALARY><MANAGER>16</MANAGER>
        </EMPLOYEE>
      </SUBORDINATES>
    </EMPLOYEE>
  </SUBORDINATES>
</EMPLOYEE>
<EMPLOYEE id="21">
  <FIRSTNAME>BH</FIRSTNAME><LASTNAME>KIM</LASTNAME>
  <SALARY>26580</SALARY><MANAGER/>
  <SUBORDINATES>
    <EMPLOYEE id = "23">
      <FIRSTNAME>JONGBUM</FIRSTNAME><LASTNAME>LEE</LASTNAME>
      <SALARY>38200</SALARY><MANAGER>21</MANAGER>
    </EMPLOYEE>
    <EMPLOYEE id="24">
      <FIRSTNAME>BIG</FIRSTNAME><LASTNAME>CHOI</LASTNAME>
      <SALARY>25480</SALARY><MANAGER>21</MANAGER>
    </EMPLOYEE>
  </SUBORDINATES>
</EMPLOYEE>
```

그림 3 그림 2의 SQL 질의의 결과투플들

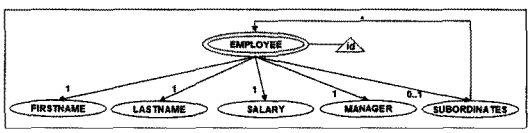

그림 4 XML 스키마 그래프

특정 XML 값의 구조를 XML 스키마 그래프로 표현한 경우, 그 그래프의 노드(node)들과 에지(edge)들이 하나의 사이클(cycle)을 형성한다면, 그 XML 값을 재귀적 구조의 XML 값이라 한다. 본 논문은 하나의 사이클을 가진 XML 스키마 그래프에서 사이클의 시작이 되는 노드를 NEST_ROOT 노드라 하며 특별히 이중 타원으로 표현하고, 그 NEST_ROOT 노드의 자식 노드로서 NEST_ROOT 노드와 사이클을 형성하며 NEST_ROOT 노드를 제외한 어떠한 자식 노드도 가지지 않는 노드를 NEST_SUBORDINATE 노드라 한다. 그림 4가 하나의 사이클을 형성하므로 그림 3의 XML 값들은 재귀적 구조의 XML 값들이다.

예제 1의 요구는 부하사원들의 범위를 두 단계로 제한하기 때문에 그 요구를 적절한 SQL/XML 출판함수들을 포함하는 중첩 SQL 질의로 작성하는 것은 간단하다. 그러나, 그 사원과 그 사원의 10 단계 아래까지의 부하사원들 또는 그 깊이를 예측할 수 없는 그 사원의 말단 부하사원들까지의 사원들에 대한 투플들로써 재귀적 구조의 XML 값을 생성하고자 하는 경우, 그 요구를 그림 2와 같이 중첩 SQL 질의로 작성하는 것은 전자의 경우는 쉽지 않으며 후자의 경우는 불가능하다. 본 논문은 SQL:1999 표준의 재귀적 질의의 select 절에 사용되어 그 질의에 의해 식별된 투플들을 대상으로 재귀적 구조의 XML 값을 생성하는 새로운 SQL 함수 XMLNEST를 제안하고 그 함수의 구문과 의미를 명세한다. 본 논문의 기본 아이디어는 SQL/XML 출판함수들의 선언적 특성을 주어진 조건에 의해 직간접으로 연결된 관계 투플들로부터 재귀적 구조의 XML 값을 생성하는 데에도 그대로 적용하는 것이다.

본 논문의 나머지 부분의 구성은 다음과 같다. 2장에서 관계 투플들을 대상으로 SQL/XML 출판함수들을 이용하지 않고 재귀적 구조의 XML 값을 생성하는 방법들과 그 방법들의 특징을 제시한다. 3장에서 본 논문이 제안하는 SQL 함수 XMLNEST가 기반으로 하는 재귀적 질의의 문법과 의미, 특징, 그리고 그 SQL 함수를 제안하기 위해 필요한 개념들을 정의하며, 4장에서 SQL 함수 XMLNEST를 제안한다. 5장에서 본 논문의 결론을 맺는다.

2. 기존 방법들

SQL/XML 출판함수들을 이용하지 않고 관계 투플들을 대상으로 재귀적 구조의 XML 값을 생성하는 방법들은 두 가지 유형으로 분류될 수 있다. 첫 번째 유형은 주어진 요구에 따라 재귀적 질의를 작성하고 그 질의에 대해 커서를 선언하며 그 커서를 통해 인출된 각 결과투플의 칼럼값들에 태그(tag)를 부착하여 XML 엘리먼트들을 생성하고 생성된 그 XML 엘리먼트들을 그 결과투플들 간의 연결 관계에 따라 재귀적 구조로 구조화하는 일련의 과정을 데이터베이스 응용 프로그램으로 구현하는 것이다. 이 유형의 방법들은 결과투플들 간의 부모-자식 관계와 결과투플들의 반환 순서를 고려해야 하며 XML 값들의 복잡한 구조화 작업을 구현해야 하는 특징을 가지므로 프로그램 작성에 많은 불편과 수고를 수반한다. 그리고 DBMS는 관계 투플들로부터 재귀

적 구조의 XML 생성이라는 관점에서 최적화된 질의 실행 계획을 생성할 수 없다.

재귀적 구조의 XML 값을 생성하는 기존 방법들의 두 번째 유형은 XQuery 질의를 이용하는 것이다. 이 경우, 개발자는 SQL:2006 표준의 XQuery 함수들[6]과 XQuery 언어를 사용할 수 있어서 생성할 XML 값의 내용과 구조에 따라 재귀적 사용자 정의 함수들을 작성하고 그 함수들의 호출을 포함하는 XQuery의 표현식(expression)을 작성해야 한다. 재귀적 사용자 정의 함수는 (1) 대상이 되는 관계 테이블들에 대해 묵시적 XML 뷰(implicit XML view)[1,7]들을 설정하고, (2) 그 뷰들을 대상으로 XQuery 표현식을 통해 직간접으로 연결된 투플들에 해당하는 XML 엘리먼트들을 식별하고, (3) 식별된 그 XML 엘리먼트들을 기반으로 의도하는 XML 값의 구조를 가지는 새로운 XML 엘리먼트들을 생성하며, (4) 그 표현식에서 그 함수 자신을 호출하여야 한다. 이 유형의 방법들을 이용한 재귀적 구조의 XML 값의 생성은 SQL 개발자들에게는 쉽지 않은 작업이다. 이 방법의 경우, DBMS는 XML 엘리먼트들의 내용뿐만 아니라 그들 간의 계층 구조를 기반으로 XML 엘리먼트들을 식별해야 하므로 평면적인 구조의 관계 투플들을 식별하는 경우에 비해 더 많은 질의 처리 비용과 수행 시간이 소요되며 묵시적 XML 뷰들에 대한 저장 비용이 요구된다.

3. 재귀적 질의

SQL:1999 표준의 재귀적 질의는 하나의 select 문이다. 그 select 문은 하나의 with 절로써 이행폐포(transitive closure)의 내용을 담을 임시 테이블들을 선언하고 정의하며 하나의 부질의로써 그 임시 테이블들과 그 외의 테이블들을 대상으로 최종 결과투플들을 생성한다. 다음은 SQL:1999 표준의 재귀적 질의를 설명하기 위하여 본 논문에서 사용되는 용어들이다.

용어_1. 이행폐포의 내용을 담을 재귀적 질의의 with 절의 임시 테이블을 재귀적 임시 테이블이라 하며 그 테이블의 투플들을 재귀적 임시 결과투플들(recursive temporary result tuples, RTRTs)이라 한다.

용어_2. 재귀적 임시 테이블의 내용의 정의는 루트 식별 부질의들과 자식 식별 부질의들로 구성된다.

용어_3. 루트 식별 부질의들은 이행폐포의 시작 투플들을 식별하여 재귀적 임시 테이블을 초기화하는 것으로서 "이행폐포의 루트조건들"을 포함한다.

용어_4. 자식 식별 부질의들은 재귀적 임시 테이블의 기존 RTRT들에 연결된 새로운 투플들을 반복적으로 그 재귀적 임시 테이블에 추가함으로써 그 재귀적 임시 테이블을 완성하는 것으로서 "이행폐포의 부모-자식 조건들"을 포함한다.

용어_5. 재귀적 질의에서 재귀적 임시 테이블들과 그 외의 테이블들을 대상으로 최종 결과투플들을 생성하는 하나의 부질의를 마감 부질의라 한다.

용어_6. 재귀적 질의의 마감 부질의에 의해 with 절에서 정의된 특정 재귀적 임시 테이블로부터 최종적으로 식별된 투플들을 재귀적 결과투플들(recursive result tuples, RRTs)이라 한다.

본 논문은 SQL:1999 표준의 재귀적 질의의 일반성을 해치지 않는 범위 내에서 재귀적 질의의 개념과 제안하는 SQL 함수 XMLNEST의 개념을 간략히 제시하기 위해 재귀적 질의의 마감 부질의에 대해 다음과 같이 가정한다. 재귀적 질의의 마감 부질의에 의해 참조되는 테이블들 중에는 재귀적 임시 테이블이 반드시 그리고 오직 하나가 존재하며 그 재귀적 임시 테이블의 임의의 RTRT는 그 부질의에 의해 최대 하나의 RRT로 생성된다. 그러한 재귀적 질의의 마감 부질의를 "RRT 생성" 양식의 부질의라 한다.

정의 1. RTRT들 간의 부모-자식 관계. 특정 재귀적 임시 테이블의 임의의 RTRT t_p에 대하여 그 테이블의 정의의 "이행폐포의 부모-자식 조건들"을 적용함으로써 직접적으로 그 재귀적 임시 테이블에 포함된 RTRT들의 집합을 SC라 할 때, 투플 t_p를 집합 SC의 투플들의 부모 투플이라 하고, 집합 SC의 투플들을 투플 t_p의 자식 투플들이라 하며, 투플 t_p와 집합 SC의 투플들의 관계를 부모-자식 관계라 한다.

정의 2. RTRT와 RRT 간의 관계. 재귀적 질의의 "RRT 생성" 양식의 부질의에서 하나의 RTRT t_a에 의해 생성된 하나의 RRT를 t_a'이라 할 때 t_a를 t_a'의 기원 투플이라 하고 t_a'을 t_a의 환생 투플이라 한다.

정의 3. RRT들 간의 부모-자식 관계. 부모-자식 관계의 임의의 RTRT들인 t_p와 t_c에 대하여 t_p가 부모 투플이고 t_c가 자식 투플일 때, t_p의 환생 투플이 t_p'이고 t_c의 환생 투플이 t_c'이면 t_p'과 t_c'은 부모-자식 관계에 있으며 t_p'을 t_c'의 부모 투플이라 하고 t_c'을 t_p'의 자식 투플이라 한다.

정의 4. RTRT-tree와 RRT-tree. 특정 재귀적 임시 테이블의 RTRT들을 대상으로 그리고 특정 재귀적 질의의 RRT들을 대상으로 그 투플들을 노드들로 표현하고, 부모-자식 관계의 투플들에 해당하는 노드들을 부모 노드로부터 자식 노드로 향하는 방향 에지로 연결하여 표현함으로써 형성되는 트리들을 각각 재귀적 임시 결과투플 트리들(recursive temporary result tuple trees, RTRT-trees)과 재귀적 결과투플 트리들(recursive result tuple trees, RRT-trees)이라 한다.

정의 5. "독자환생" RRT-tree. 임의의 "RRT 생성"

양식의 부질의에서 하나의 RTRT-tree가 정확히 하나의 RRT-tree를 생성한 경우, 생성된 그 RRT-tree를 그 RTRT-tree의 "독자환생" RRT-tree라 한다.

정의 6. "루트유지" RRT-tree. 임의의 "RRT 생성" 양식의 부질의에서 임의의 RTRT-tree에 대하여 그 트리의 루트 노드에 해당하는 RTRT를 기원 투플로 가지는 RRT가 존재할 경우, 그 RRT에 해당하는 노드를 루트로 가지는 RRT-tree를 그 RTRT-tree의 "루트유지" RRT-tree라 한다.

예제 2. 테이블 EMPLOYEE에서 상사를 가지지 않는 각 사원에 대하여 그 사원과 그 사원의 모든 직간접 부하 사원들에 해당하는 투플들을 인출하시오. 단, 급여가 32000 미만이거나 급여가 38000 이상이고 52800 이하의 범위에 속하는 사원들만이 결과에 포함되도록 하시오.

예제 2를 위한 질의는 그림 5와 같다. 그림 6은 그림 5의 질의의 RTRT-tree들과 RRT-tree들을 나타낸다.

```
WITH empHierarchy(emp_id, first_name, last_name, salary, spv_id) AS (
        SELECT employee_id, first_name, last_name, salary, supervisor_id
        FROM employees WHERE supervisor_id IS NULL
        UNION ALL
        SELECT e.employee_id, e.first_name, e.last_name, e.salary, e.supervisor_id
        FROM employees e, empHierarchy eh WHERE e.supervisor_id = eh.emp_id
)
SELECT emp_id, first_name, last_name, salary, spv_id
FROM empHierarchy WHERE salary < 32000 OR (salary >= 38000 AND salary <= 52800)
```

그림 5 예제 2를 위한 질의

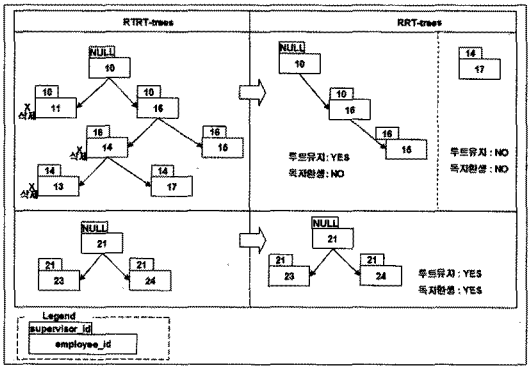

그림 6 그림 5의 질의의 RTRT-tree들과 RRT-tree들

4. XMLNEST

SQL 함수 XMLNEST는 "RRT 생성" 양식의 부질의의 select 절에 명시될 수 있다. SQL 함수 XMLNEST는 그 부질의가 형성하는 각 RRT-tree에 대하여 그 RRT-tree의 모든 노드들을 대상으로 하나의 XML 값을 생성하여 그 값을 그 RRT-tree의 값으로 설정한다. SQL 함수 XMLNEST는 그 함수가 포함된 임의의 "RRT 생성" 양식의 부질의에서 그 부질의의 모든 RRT-tree들이 "루트유지" RRT-tree이면서 "독자환생" RRT-tree인 경우에 한하여 의미를 가지도록 제한된다. 예를 들어, 그림 6의 RRT-tree들을 생성하는 그림 5의 재귀적 질의에 XMLNEST를 사용하면 실행시간 오류로 처리된다.

하나의 사이클을 가진 XML 스키마 그래프에서 NEST_ROOT 노드가 나타내는 XML 엘리먼트를 NEST_ROOT 엘리먼트라 정의하고, NEST_SUBORDINATE 노드가 나타내는 XML 엘리먼트를 NEST_SUBORDINATE 엘리먼트라 정의하자. 하나의 RRT-tree에 대해 하나의 재귀적 구조의 XML 값을 생성하는 기본 아이디어는 다음과 같다.

(1) RRT-tree의 각 노드에 대해 하나의 NEST_ROOT 엘리먼트가 생성된다.

(2) RRT-tree의 각 노드에 대해 NEST_ROOT 노드의 자식 노드들에 해당하는 XML 엘리먼트들이 그 노드를 위한 NEST_ROOT 엘리먼트의 자식 엘리먼트들로 생성된다.

(3) RRT-tree의 각 노드에 대해, 그 노드를 RRT_NODE_i라 하고 RRT_NODE_i의 자식 노드들의 집합을 CS_i라 하면, CS_i의 모든 노드들을 위한 NEST_ROOT 엘리먼트들은 RRT_NODE_i를 위한 NEST_SUBORDINATE 엘리먼트의 자식 엘리먼트들이 될 수 있으며 그 NEST_ROOT 엘리먼트들 간의 순서는 명시될 수 있다.

(4) RRT-tree의 각 노드에 대해 그 노드를 위한 NEST_ROOT 엘리먼트와 NEST_SUBORDINATE 엘리먼트를 각각 엘리먼트R과 엘리먼트S라 하면, 1) 엘리먼트R의 속성 값들과 문자 데이터, 2) 엘리먼트S와 엘리먼트S의 후손 엘리먼트들을 제외한 엘리먼트R의 후손 엘리먼트들의 속성 값들과 문자 데이터, 그리고 3) 엘리먼트S의 속성값들은 그 노드에 대응되는 RRT의 칼럼 값들을 기반으로 생성된다.

그 기본 아이디어를 구현하기 위한 SQL 함수 XMLNEST의 명세는 그림 7과 같다. 그림 7의 명세는 SQL/XML 출판함수들에 대한 SQL:2003 표준의 표기 방법을 따른다. 그림 7에서 (1) <XML nest_root element spec>은 NEST_ROOT 엘리먼트의 이름, 그 엘리먼트의 XML 이름공간들, 그리고 그 엘리먼트의 XML 속성들에 대한 명세이며, (2) <XML element content>는 NEST_SUBORDINATE 엘리먼트를 제외한 NEST_ROOT 엘리먼트의 각 자식에 대한 명세이며, (3) <XML nest_subordinate element spec>은 NEST_SUBORDINATE 엘리먼트의 이름과 선택사양(option), 그 엘리먼트의 XML 속성들, 그리고 NEST_SUBORDINATE 엘리먼트의 자식들로 생성되는 NEST_ROOT 엘리먼트들 간의 정렬 기준에 대한 명세이다. NEST_SUBORDINATE 엘리

먼트의 선택사양으로 "MANDATORY"가 명시되면 모든 NEST_ROOT 엘리먼트들이 NEST_SUBORDINATE 엘리먼트를 가짐을 의미하며, "OPTIONAL"이 명시되면 단말 노드를 위해 생성되는 NEST_ROOT 엘리먼트는 NEST_SUBORDINATE 엘리먼트를 가지지 않음을 의미한다. 그 선택사양의 디폴트(default)는 "MANDATORY"이다.

그림 7의 명세에서 기술되지 않은 상세 문법들은 그 명세에 대한 SQL/XML의 그것들과 동일하다. 참고로, RTRT-tree들의 깊이를 제한하는 기능은 재귀적 질의 자체에서 지원될 수 있기 때문에 그림 7의 명세는 그 기능을 포함하고 있지 않다.

```
<XML nest>::=
        XMLNEST <left paren>
        <XML nest_root element spec>
        [{<comma> <XML element content>}...]
        <comma> <XML nest_subordinate element spec>
        [{<comma> <XML element content>}...] <right paren>

< XML nest_root element spec > ::=
        [NAME] <XML element name>
        [<comma> <XML namespace declaration>]
        [<comma> <XML attributes>]

< XML nest_subordinate element spec > ::=
        XMLNESTSPEC <left paren>
        [[NAME] <XML element name> [<XML nest_subordinate element option>]
        [<comma> <XML attributes>]]
        [ORDER CHILDREN BY <sort specification list>]
        <right paren>

<XML nest_subordinate element option> ::= OPTIONAL | MANDATORY
```

그림 7 XMLNEST의 명세

SQL 함수 XMLNEST를 포함한 select 절과 group by 절은 하나의 부질의에 함께 사용되지 않아야 한다. 그러나, SQL의 기존 집단함수들인 SUM, COUNT, AVG, MIN, MAX, 그리고 XMLAGG는 XMLNEST와 함께 동일 부질의의 select 절에 명시될 수 있다. 그 경우, 그 기존 집단함수들은 그 부질의가 형성하는 각 RRT-tree에 대하여 그 RRT-tree의 모든 노드들을 대상으로 계산한 하나의 값을 그 RRT-tree의 값으로 설정해야 한다.

예제 3. 예제 1을 위한 질의를 XMLNEST를 이용한 재귀적 질의로 작성하시오. 단, 예제 1과 달리 형제 NEST_ROOT 엘리먼트들을 사원들의 급여에 따라 정렬하시오.

예제 3을 위한 질의는 그림 8과 같다. 그 질의의 결과는 사원번호가 14와 15인 XML 엘리먼트들의 순서를 제외하면 그림 2와 동일하다. 그림 1의 질의는 부질의 중첩(subquery-nesting)을 통해 부하사원 수준에 제한을 가하지만 그림 8의 질의는 각 RTRT가 칼럼 depth를 가지도록 하고 그 칼럼에 적절한 값이 설정되도록 하며, 부모 RTRT가 될 수 있는 RTRT들의 칼럼 depth의 값을 특정 값들로 한정함으로써 부하사원 수준에 제한을 가하고 있다.

```
WITH empHierarchy(employee_id, first_name, last_name, salary, supervisor_id, depth) AS (
        SELECT employee_id, first_name, last_name, salary, supervisor_id, 0
        FROM employee
        WHERE employee_id = 10 or employee_id = 21
        UNION ALL
        SELECT e.employee_id, e.first_name, e.last_name, e.salary, e.supervisor_id, eh.depth+1
        FROM employee e, empHierarchy eh
        WHERE eh.employee_id = e.supervisor_id and eh.depth <= 1 )

SELECT XMLNEST("EMPLOYEE", XMLATTRIBUTES(employee_id as "id"),
                XMLELEMENT("FIRSTNAME", first_name),
                XMLELEMENT("LASTNAME", last_name),
                XMLELEMENT("SALARY", salary),
                XMLELEMENT("MANAGER", supervisor_id),
                XMLNESTSPEC("SUBORDINATES" OPTIONAL ORDER CHILDREN BY salary)
        ) as result
FROM empHierarchy
```

그림 8 예제 3을 위해 작성된 SQL 질의

그림 8의 질의와 같이 XMLNEST를 이용한 방법은 질의 작성의 편의성, 질의 처리 비용과 속도, 그리고 저장 비용에서 2장의 방법들에 비해 장점을 가진다.

5. 결 론

본 논문은 SQL/XML 출판함수들의 선언적 특성을 재귀적 질의의 결과를 재귀적 구조의 XML 값으로 생성하는 데에도 그대로 적용하기 위하여 SQL 함수인 XMLNEST를 제안하였다. SQL 함수 XMLNEST를 이용함으로써 응용 프로그램 또는 XQuery 질의를 작성할 필요가 없다. 다만, 생성하고자 하는 XML 값의 재귀적 구조와 형제 XML 엘리먼트들 간의 순서를 XMLNEST에 기술하면 된다.

참 고 문 헌

[1] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, B. Reinwald, "Efficiently Publishing Relational Data as XML Documents," VLDB Journal 10(2-3), 2001.

[2] A. Eisenberg, J. Melton, "Advancements in SQL/XML," SIGMOD Record, Vol.33, No.3, pp. 79-86, 2004.

[3] ISO, "ISO/IEC 9075-14:2003(E) Information technology - Database Languages - SQL - Part 14: XML-Related Specifications (SQL/XML)," 2003.

[4] J. Melton, S. Buxton, "QUERYING XML XQuery, XPath, and SQL/XML in Context," Morgan Kaufmann Publishers, 2006.

[5] ISO, "ISO/IEC 9075-2-1999 for Information Technology - Database Languages - SQL - Part 2: Foundation (SQL/Foundation)," 1999.

[6] ISO, "ISO/IEC 9075-14:2006(E) Information technology - Database languages - SQL - Part 14: XML-Related Specifications (SQL/XML)," 2006.

[7] ORACLE, "SQL Language Reference 11g Release 1 (11.1)," 2008.