임베디드 RDBMS로의 효율적인 XML 문서 저장 방법[1)]

조국래°, 임태형, 강원석
대구경북과학기술연구원
kookrae@dgist.ac.kr, THLim@dgist.ac.kr, wskang@dgist.ac.kr

Efficient XML Document Storage Model to Embedded RDBMS

Kookrae Cho°, Tae-Hyung Lim, Won-Seok Kang
Daegu Gyeongbuk Institute of Science and Technology

요 약

멀티미디어 데이터에 대한 효율적인 검색을 지원하기 위해 제정된 MPEG7이나 TV-AnyTime 등의 표준들은 XML을 사용하여 Metadata를 효과적으로 표현할 수 방법을 제시하고 있다. 플랫폼에 독립적으로 사용 가능하다는 XML의 강점에도 불구하고 Text 방식으로 데이터를 저장하기 때문에 보안에 취약할 수 밖에 없으며, 대용량의 자료 처리에도 문제점을 드러낸다. 이를 보완하기 위하여 XML 문서를 RDBMS에 저장하는 방법들이 제안되고 있는데, 기존의 Model Mapping이나 Structure Mapping의 경우 노드 검색을 위해 Traversing을 할 경우 많은 조인이 필요하기 때문에 한정된 메모리와 낮은 처리능력을 갖춘 임베디드 멀티미디어 플랫폼에서는 비효율적일 수 밖에 없다. 본 논문에서는 MPEG7 Metadata를 RDBMS에 저장하고, 이를 검색할 때, 조인의 횟수를 최소화할 수 있는 저장 모델을 제안하고 있다. 부모, 자식, 형제간의 노드를 효율적으로 검색할 수 있도록 path MetaSchema를 제안함으로써, 최소 1번의 검색으로 필요한 노드 정보들을 추출할 수 있도록 하였다. 그리고 Annotation 작성시에 XML의 Attribute와 Element로 삽입하였기에, 기존의 방법에서 Annotation을 분석하기위해 필요했던 Annotation 파싱을 제거하고, SAX나 DOM을 사용할 수 있도록 제안함으로써, Annotation의 정보에 대한 접근이 효율적으로 개선되었다.

1. 개 요

멀티미디어 관리 시스템은 멀티미디어 데이터(오디오, 이미지, 동영상 등)를 효과적으로 검색 및 저장하기 위한 기술로서 이러한 기능을 정의한 표준(MPEG-7, TV-Any Time등)들이 표준화가 완료되었거나 진행 중에 있다. 이러한 표준들은 자료에 대한 정보를 표현하는 Metadata를 효율적으로 표현할 수 있는 방법들을 제시하고 있는데 구현방법으로는 XML의 규약을 따르고 있다.

그림 1과 같은 멀티미디어 관리 시스템에서 플래쉬 메모리에 저장되어 있는 음악, 비디오와 같은 멀티미디어 컨텐츠들의 검색을 용이하게 하기 위하여 Metadata를 활용하게 되는데, 이러한 Metadata를 XML로 작성하여 보관하게 되면, 데이터 표현형식에 일관성을 제공하고, 네트웍을 통해 쉽게 전송할 수 있으며, 플랫폼에 독립적으로 사용할 수 있다는 XML의 장점을 그대로 유지할 수 있다. 하지만 XML 문서는 텍스트 형식으로 작성되기 때문에 보안이 취약할 수밖에 없다.

이러한 문제를 해결하기 위해, XML로 제작된 Metadata를 RDBMS에 저장함으로써, RDBMS가 지원하는 강력한 관리 및 보안기능을 보완할 수 있고, 잠금 메커니즘을 통해 데이터 원본의 무결성을 확보할 수 있다. 그러므로, Metadata는 XML로 구성하고, 이렇게 생성된 XML문서를 RDBMS에 저장하도록 함으로써, XML과 RDBMS의 장점만을 가지는 저장 모델을 설계하고자 한다. 그리고 물리 저장장치가 임베디드 시스템 상에서 Flash Memory 기반으로 구성되어져 있기 때문에, 필요한 자료의 검색시에 조인의 횟수를 최소화하는 등 물리적인 특성을 고려하여 멀티미디어 관리 시스템을 개발하고자 한다.

본 연구에서는, 2장에서 저장 모델에 대한 선행기술을 조사하고, 3장에서 저장 모델의 기본플랫폼에 대해서 설명한다. 그리고 MPEG-7 XML 스키마를 효율적으로 RDBMS에 저장하는 방법에 대해서 기술하고, 이렇게 설계한 모델에 대한 예제를 보여줌으로써 본 논문을 마치고자 한다.

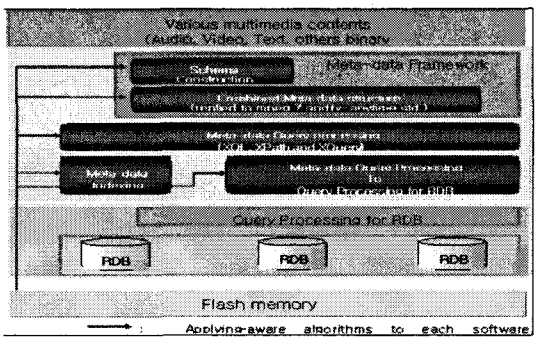

그림 1. 멀티미디어 관리 시스템

1) 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음. [2006-S-040-01, Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술 개발]

2. 저장 모델 선행 기술 분석

MPEG-7은 XML로 구성된 멀티미디어 컨텐츠를 기술하기 위한 표준으로써, XML 스키마에 기반한 Description Definition Language(DDL)의 도움으로 자료 검색을 위한 효율적인 구조로 필요한 정보들을 정의 및 개선할 수 있다. MPEG-7 표준대로 문서를 작성하면, Plain Text로 정보를 교환하기 때문에 서로 다른 이기종간의 데이터 전송에 효과적으로 활용될 수 있다. 하지만, 이러한 장점에도 불구하고 피할수 없는 단점들이 있는데, 그것은 중복되는 정보가 많으며, 파싱과 텍스트 변환에 따른 낮은 접근 능력 등이다. 이러한 문제점들은 MPEG-7을 준수하여 작성된 멀티미디어 정보에 대한 쿼리 및 인덱싱의 처리속도를 저하시키는 주요 원인이되고 있다. 그러므로, 멀티미디어 저장 시스템에 있어서, MPEG-7 Descriptors와 Description 스킴을 어떻게 하면 효율적으로 저장하고 인덱싱 할 수 있는지가 주요 관심사가 되어왔다.

MPEG-7이 데이터중심의 정보 표현 방법이라고 생각하면, 데이터를 효과적으로 저장 및 접근하는 것을 목표로 잘 발달되어 있는 기존의 RDBMS를 이용하여 MPEG-7으로 표현된 자료들을 처리하는 것은 자연스러울 수 있다. 이렇게 MPEG-7 Document 저장 모델을 설계할 때 기존의 데이터베이스를 이용하는 경우에는 MPEG-7만이 가지고 있는 특성을 고려하여야 한다. 첫째, MPEG-7 Description Schemes는 고정되어 있지 않다. 두번째, MPEG-7은 리스트와 같은 많은 complex datatype의 변형된 형태를 제공한다. 세번째, MPEG-7의 많은 데이터들이 단순히 텍스트 형태로 처리되지 않는다.

이러한 MPEG-7만의 특성을 고려하여 MPEG-7 문서들을 RDBMS에 효율적으로 저장하기 위해서는, MPEG-7 Description 스키마를 데이터베이스 스키마에 적절히 mapping 할 수 있는 방법이 필요한데 이를 위해서 사용가능한 기법으로는 LOB 기반 저장 방법, Model Mapping Approach와 Structure Mapping Approach의 MPEG-7 저장모델 방법들이 있다.

LOB 기반의 저장 방법은 간단히 생각할 수 있는 방법인데, 한 컬럼에 XML 문서를 그대로 저장하므로, XML 문서를 분석하거나 여러 테이블에 저장할 필요가 없는 단순한 방법이다. XML 문서를 파일 시스템 상의 파일 형태로 저장해 놓고 필요시에 XML 문서를 추출하여 처리하는데, 장점으로는 mark-up 태그를 파싱 없이 그대로 싱글 셀에 저장하기 때문에 빠른 저장 속도를 가질 수 있다. 하지만, XPath나 XQuery와 같은 사용자 쿼리를 수행하는 경우 필요한 데이터 검색 속도에 상당한 지연이 발생한다. 또한 LOB type은 부분 갱신이라 할지라도 문서 전체를 갱신해야 하기 때문에, 이러한 구조는 Flash 영역의 과도한 쓰기 집중도를 야기할 수 있으며, Garbage Collection을 과도하게 발생시킬수 있는 문제가 예상된다.

Model Mapping Approach의 경우 XML 문서 스키마(DTD 혹은 XML schema)의 도움없이 고정된 데이터베이스 스키마를 사용하여 XML 문서의 구조 및 정보들을 테이블에 저장하고 있다. XML 문서의 완전한 구조가 저장되기 때문에 XPath기반의 쿼리들을 충분히 지원할 수 있으며, XML 형태로 자료를 간단히 재구성하는 장점을 가지고 있다. 하지만, 스트링으로 데이터값을 싱글 칼럼에 저장하기 때문에 IDREFS나 NMTOKENS 등과 같은 리스트 데이터 등에 대한 인덱싱 정보를 저장하기 힘들고, 이러한 데이터 저장형태는 모든 종류의 데이터 타입을 반영할 수 없다는 단점을 가지고 있다. 게다가 XML 문서의 정보를 완벽하게 유지하기 위한 여러 개의 테이블이 존재 할 수 있는데, 임베디드 DBMS, 특히 Flash Memory를 기반으로 하는 저장장치에서 Query 사용시 Temporary Storage 공간이 필요하고, Join Operation이 늘어날 수 있는 단점이 존재한다. 관련 연구로는 Edeg Approach[13], XRel[1], XParent[2] 등이 있다.

Structure Mapping Approach의 경우 DTD나 XML 스키마의 이해를 바탕으로 데이터베이스 스키마를 구성한다. 각 엘리먼트 타입간의 관계를 정의하고, 엘리먼트내에서 부모 자식간의 관계 정립을 위하여 Primary Key와 Foreign Key를 사용한다. 장점으로는 Built-in 데이터베이스 인덱스를 사용할 수 있으므로, 효율적인 인덱싱이 가능하기 때문에 문서 쿼리를 신속하게 처리할 수 있다. 하지만, MPEG-7 응용 프로그램에 있어서 XML 문서의 재구성 기능은 필수적으로 존재해야 함에도 불구하고 original XML 문서에 대한 전체 구조를 완벽하게 저장하는 것이 아니므로, 원래의 XML 문서를 재구성할 수가 없다. 게다가 XPath 기반의 쿼리는 사용할 수 없다는 단점과 세부노드별 라우팅 패스 설정이 힘들다는 어려움이 존재한다. 관련 연구로는 X-Ray[4], XML-DBMS[5], Cost-based Approach[6], ShreX[12] 등이 있다.

3. 제안하는 Metadata 저장 모델

3.1 저장 모델 기본 플랫폼

MPEG-7에서는 멀티미디어 컨텐츠의 검색을 위해 다양한 방법들의 표준이 제안되고 있는데, 여기서는 한정된

메모리와 제한된 데이터 처리능력을 갖추고 있는 Embedded 멀티미디어 장치의 환경에 적합하도록, MPEG-7 스키마 중에서 필요한 부분들만을 추출하여 개발자들에게 제공하고자 한다. 개발자들은 이중에서 자신들이 실제 사용할 부분들에 Annotation을 작성하여 Annotation MetaSchema를 생성한다. 이렇게 생성한 Annotation MetaSchema를 바탕으로 XML로 저장되어 있는 Metadata를 RDBMS에 저장하기 위해 필요한 테이블과 칼럼을 RDBMS에 생성한다. 그리고 Annotation MetaSchema를 사용해서 XML 엘리먼트와 속성을 RDBMS에 저장되어 있는 테이블과 칼럼으로 맵핑할 수 있는 Path MetaSchema를 작성한다.

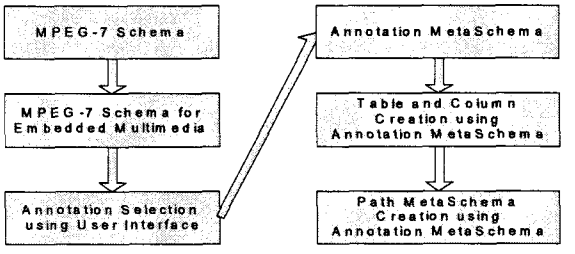

그림 2. 기본 요소 생성 순서도

XML Document를 입력하거나, RDBMS에 저장되어 있는 XML Instance를 검색하기 위해서 Annotation MetaSchema를 활용할 경우, 트리의 깊이만큼 traversing하는 것이 필요하지만, Path MetaSchema를 활용할 경우 사용하는 인덱싱 알고리즘에 따라서 차이는 있겠지만, 한 두 번의 검색만을 통해서 찾고자하는 XML Instance가 저장되어 있는 위치를 알아낼 수 가 있다.

그림에서 보는바와 같이, XML Document가 입력되면, XML-RDBMS Processor는 Path MetaSchema에서 현재 Document의 Path 정보에 해당하는 테이블과 칼럼 정보를 추출하고, XML-RDBMS Processor는 이 정보를 바탕으로 RDBMS에 접근한다. 그리고 자료의 저장 및 검색 등의 필요한 작업들을 해당 테이블이나 칼럼에 요구한다. RDBMS에 저장되어 있는 자료를 질의할 때, RDBMS는 SQL 질의어를 사용하지만, XML 문서를 질의할 때는 XQuery나 XQL 등과 같은 특정 질의어를 사용해야 한다. 그러므로 사용자가 XQuery를 사용하여 XML 문서를 질의 하고자 할 때, 이를 RDBMS의 질의어인 SQL로 바꾸어 주는 쿼리 변환기가 필요하다.

3.2 Annotation 구성 요소 및 기술 방법

3.2.1 Annotation 기본 요소

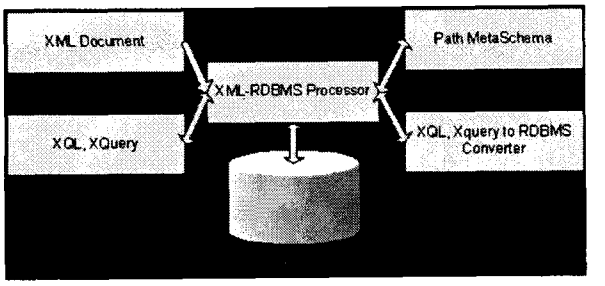

그림 3. 저장 모델 기본 플랫폼

Embedded Multimedia 기기에 사용되는 Annotation은 태그와 Action 함수로 구분할 수 있다. Annotation 태그에는 <sqlite_Table>, <sqlite_Element>, <sqlite_Header>, <sqlite_Attribute>가 있고, 해당 Annotation이 존재하는 위치에서 테이블과 칼럼의 생성을 암시한다. 이러한 태그내에 createColumn, columnIntoTable, createPrimaryKey, createForeignKey와 같은 Annotation Action 함수를 삽입하여, 지정하는 위치에서의 특정 칼럼생성을 나타낸다. 여기서는 Annotation 태그들에 대해서 알아보고, 태그 속에서 동작하는 함수들의 종류와 그 사용방법에 대해서 설명한다.

3.2.1.1 Annotation 태그의 종류와 역할

- **<sqlite_Table Tablename="name", createPrimaryKey="action", createForeignKey="action" </sqlite_Table>** : 테이블 생성에 대한 Annotation을 나타내고, Tablename이 가리키는 name 테이블을 생성하라는 의미이다. createPrimaryKey와 create ForeignKey에 있는 action에 따라서 PrimaryKey와 Foreign Key를 각각 생성한다.

- **<sqlite_Element ElementName="name", columnCreation= "action" </sqlite_Element>** : Element와 관련한 Annotation 을 나타낸다. ElementName은 실제 XML Instance에서의 Element 이름을 뜻하고, columnCreation의 Action 부분에서는 실제 수행할 함수 이름을 나타낸다.

- **<sqlite_Header ElementName="name", columnCreation= "action" </sqlite_Header>** : sqlite_Element와 동일한 동작을 수행하는데, 현재 Element가 속성과 content를가지고 있는 경우 해당 Attribute가 현재의 Element에 속한 Attribute임을 명확히 하기 위해서 사용된다.

- **<sqlite_Attribute AttributeName="name", columnCreation= "action" </sqlite_Attribute>** : 속성과 관련한 Annotation을 의미하고, AttributeName은 실제XML Instance에서의 속성 이름을 말한다. Action 부분에는 실제 수행할 함수 이름을 나타낸다.

3.2.1.2 Annotation Action 함수의 종류와 역할

- **createColumn <Columnname/>, <Type/>, <IndexType/>** : 3개의 태그로 구성되어 있는데, Columnname의 이름을 가지고 Type의 데이터유형을 가지는 칼럼을 테이블에 생성하라는 것을 의미한다. 해당 칼럼이 삽입될 장소는 현재 Element나 Attribute의 가장 가까운 조상이 생성한 테이블이다.

- **columnIntoTable <Columnname/>, <Tablename/>, <Type/>, <IndexType/>** : 4개의 태그로 구성되어 있으며, Columnname의 이름을 가지고, Type의 데이터 유형을 가지는 칼럼을 생성하라는 것은 createColumn()과 유사하지만, 칼럼을 생성할 테이블을 TableName에 지정할 수 있는 것이 다른 점이다..

- **createPrimaryKey <pri_Tablename/>, <pri_PrimaryKeyname/>, <pri_Type/>, <pri_IndexType/>** : pri_Tablename의 테이블에 pri_PrimaryKeyname의 이름을 가진 pri_PrimaryKey를 pri_Type의 데이터 유형으로 생성한다. 해당 칼럼에 Indexing 알고리즘을 적용하고자 한다면, pri_IndexType에 표현하면 된다.

- **createForeignKey <for_SourceTablename/>, <for_Foreign Keyname/>, <for_Type/>, <for_DestinationTablename/>, <for_PrimaryKeyname/>, <for_IndexType/>** : for_DestinationTable name 테이블의 for_PrimaryKeyname를 가리키는 for_Foreign Keyname 칼럼을 for_Type의 데이터유형으로 for_Source Tablename 테이블에 생성한다. 이때, Indexing Algorithm을 for_IndexType으로 지정한다.

3.2.2 Annotation 위치와 방법

- Annotation 위치 : Annotation 정보는 Element에서 <sqlite_ Table>, <sqlite_Element>, <sqlite_Attribute>, <sqlite_Header>로써 삽입한다. 테이블 생성과 관련한 정보를 나타내는 <sqlite_Table>는 문서내의 어느 Element에도 존재할 수 있지만, 테이블에 Element와 속성 정보를 삽입하는 태그인 <sqlite_Element>와 <sqlite_Attribute>의 경우에는 기본형이나 기본형에서 파생된 XML 스키마 데이터 타입인 경우에만 나타난다. 즉 Element의 데이터 타입이 여러 개의 Element를 가지는 Complex 데이터 타입이 아니라, 현재 데이터가 최종 데이터 타입인 경우에만 <sqlite_Element>를 삽입하여 해당 테이블에 칼럼을 생성하는 동작을 수행한다. . <sqlite_Attribute>는 속성을 나타내는 엘리먼트의 시작과 끝 태그 사이에 삽입한다. <sqlite_Header>는 <sqlite_ Element>를 사용해서 칼럼을 생성하고자 할 때, 해당 엘리먼트가 컨텐츠와 속성을 가지고 있는 경우, 컨텐츠의 생성을 명확히 표현하기 위해 사용한다. <sqlite_Attribute>와 같이 해당 엘리먼트의 시작과 끝 태그 사이에 생성한다.

3.3 Path MetaSchema 생성

XML 문서로 되어있는 Metatdata를 RDBMS에 저장하거나, RDBMS에 저장되어 있는 Metadata를 검색하기 위해서는 사용자가 쿼리하고자 하는 엘리먼트나 속성들이 저장되어 있는 RDBMS 테이블과 칼럼 이름이 필요하다. 이렇게 엘리먼트나 속성들을 테이블과 칼럼에 연관시켜 주는 테이블을 Path MetaSchema라 부른다.

XML-RDBMS Processor는 XML Document를 저장하거나, 검색, 수정 등의 XQuery가 들어오면, Path MetaSchema를 이용하여 해당하는 엘리먼트의 RDBMS 위치, 즉 테이블과 칼럼을 찾아낼 수 있다. 그리고 XQuery 질의를 RDBMS의 문법(SQL)으로 바꿔주는 질의 변환기를 통과하여, 해당 RDBMS내의 테이블과 칼럼에 대해 원하는 동작을 수행한다. 사용자 질의응답에 대한 처리속도를 원활하게하기 위해서, Path MetaSchema를 메모리에 올려놓고 질의문을 처리할 수도 있다.

Path MetaSchema를 생성하는 방법은 Annotation MetaSchema에서 <sqlite_Table>등과 같은 Annotation을 탐색하고, 해당 엘리먼트나 속성의 Path정보와 그들이 속해있는 Table 및 Column 이름을 저장한다. 엘리먼트의 구분은 '/'를 이용하고, 속성정보를 나타낼때는 '@'를 표시한다. <sqlite_Table> Annotation으로 생성된 Element에 자식노드가 있는 경우 columnName 칼럼에 '~childname'을 삽입한다. 이때 자식이 여러명일 경우 '~childname, ~childname, ... ,~childname'으로 나타낸다. 이렇게 함으로써, 부모노드들은 자식노드들은 자신들의 형제노드 검색을 효율적으로 처리할 수 있다.

3.4 저장 모델 예제

이제 앞서 설명한 저장 모델 기법을 바탕으로 어떻게 Annotation MetaSchema 를 작성하는지와 이를 바탕으로 생성되는 테이블과 칼럼, 그리고 path MetaSchema의 예제를 보여주고자 한다.

3.2.2 Annotation MetaSchema

```
<MPEG7>
  <Description>
    <sqlite_Table tableName="Description" primary="createPrimaryKey"
                  foreign="NULL">
      <pri_tableName>description</pri_tableName>
      <pri_primaryKeyName>description_PID</pri_primaryKeyName>
      <pri_columnType>Integer</pri_columnType>
      <pri_indexType>Btree</pri_indexType>
    </sqlite_Table>
```

```
<DescriptionMetadata>
 <sqlite_Element elementName="Confidence" columnCreation="createColumn">
    <columnName>Confidence</columnName>
    <columnType>Real</columnType>
  </sqlite_Element>
  <create_double>
    <sqlite_Table tableName="create_double"
          primary="createPrimaryKey" foreign="createForeignKey">
      <pri_tableName>createdouble_table</pri_tableName>
      <pri_primaryKeyName>createdouble_PID</pri_primaryKeyName>
      <pri_columnType>Integer</pri_columnType>
      <pri_indexType>Btree</pri_indexType>
      <for_sourceTableName>createdouble_table</for_sourceTableName>
      <for_foreignKeyName>description_FID</for_foreignKeyName>
      <for_columnType>Integer</for_columnType>
      <for_destinationTableName>description</for_destinationTableName>
      <for_primaryKeyName>desccription_PID</for_primaryKeyName>
      <for_indexType>Btree</for_indexType>
    </sqlite_Table>
  </create_double>
</DescriptionMetadata>
<end>
 <sqlite_Header elementName="end" columnCreation="createColumn">
   <columnName>end_1</columnName>
   <columnType>Text</columnType>
 </sqlite_Header>
  <sqlite_Attribute elementName="Attr1" columnCreation="createColumn">
   <columnName>Attr_1</columnName>
   <columnType>Text</columnType>
   <indexType>Btree</indexType>
  </sqlite_Attribute>
 </end>
</Description>
</MPEG7>
```

3.4.2 테이블 및 칼럼 생성

위 3.4.1의 Annotation MetaSchema를 파싱하면, 다음과 같이 2개의 tableName, createdouble_table이 생성된다. contents_id는 모든 테이블에 삽입되는 default값으로써, 하나의 컨텐츠는 다른 컨텐츠와 구별하기위한 id를 부여받게된다. 각 칼럼은 칼럼 이름, 칼럼 속성, 그리고, 인덱스가 있다면, 인덱스 정보들로 구성된다. 각각의 인덱스는 해당하는 칼럼에 'a_idx'덧붙여서 생성한다.

tableName				
description_PID	Confidence	end_1	attr_1	contents_id
(integer:Btree)	(real)	(text)	(text:Btree)	(integer)

createdouble_table		
createdouble_PID	description_FID	contents_id
(integer:Btree)	(integer:Btree)	(integer)

3.4.3 Path MetaSchema 생성

Path MetaSchema는 path, tableName, columnName의 칼럼으로 구성되어 있다. 3.4.1의 예제로 생성된 path Meta Schema는 다음과 같다. Path MetaSchema에 있는 path 정보를 사용하여 해당 엘리먼트가 저장되어 있는 테이블과 칼럼을 효율적으로 찾을 수 가 있다. 예를 들어, Path MetaSchema 테이블을 살펴보면, MPEG/Description/descriptionmetadata/Confidence가 Description 테이블의 Confidence 칼럼에 저장되어 있음을 쉽게 알 수 있다. 그리고 MPEG/Description/Descriptionmetadata의 자식노드를 찾고자 한다면, columnName에서 구분자를 '~'로 파싱하면, Confidence와 create_double의 자식 노드를 가지고 있음을 알 수 있다.

path	tableName	columnName
MPEG/	NULL	~Description
MPEG/Description	Description	~DescriptionMetadata ~end
MPEG/Description/Desccriptionmetadata	NULL	~Confidence ~create_double
MPEG/Description/Desccriptionmetadata/Confidence	Description	Confidence
MPEG/Description/Desccriptionmetadata/createdouble_table	createdouble_table	NULL

4. 결 론

본 논문에서는 멀티미디어 관리 시스템에서 멀티미디어 데이터(오디오, 이미지, 동영상 등)을 효과적으로 검색할 수 있도록 MPEG7 Metadata를 저장할 수 있는 저장 모델을 제안하고 있다. XML로 구성되어 있는 Metadata를 RDBMS에 저장하고 **XQuery**를 사용하여 저장된 정보들을 활용할때, 부모, 자식, 형제간의 노드를 효율적으로 검색할 수 있도록 path MetaSchema를 제안함으로써, 기존의 Model Mapping이나 Structure Mapping에서 필요한 노드 검색 시 발생하였던 많은 조인의 횟수를 효율적으로 감소시킨 장점을 가지고 있다. 그리고 Annotation 작성시에 Element의 값으로 삽입하였던 기존의 방법과는 달리, XML의 Attribute와 Element로 Annotation을 작성하였다. 따라서 기존의 방법에서 Annotation을 분석하기위해 필요했던 Annotation 파싱이 여기서는 DOM을 사용하여 직접 Annotation의 필요한 정보에 접근할 수 있기 때문에 보다 효율적으로 개선되었다.

* 참고 문헌

[1] Masatoshi Yoshikawa and Toshiyuki Amagasa. *XRel: A path-based approach to storage and retrieval of XML documents using relational databases.* ACM Transactions on Internet Technology, 1(1). 2001.

[2] Haifeng Jiang, Hongjun Lu, Wei Wang and Jeffrey Xu Yu. *XParent: An Efficient RDBMS-Based XML Database*

System. Proceedings of the 18th International Conference on Data Engineering (ICDE'02), 2002.

[3] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt and Jeffrey Naughton. *Relational Databases for Querying XML Documents: Limitations and Opportunities*. The 25th VLDB Conference, Edinburgh, Scotland, 1999.

[4] Gerti Kappel, Elisabeth Kapsammer and Werner Retschitzegger. *X-Ray – Towards Integrating XML and Relational Database Systems*. The International Conference on Conceptual Modeling, 2000.

[5] R. Bourret, C. Bornhövd, A. Buchmann. *A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases*. 2nd Int. Workshop on Advanced Issues of EC and Web-based Information Systems (WECWIS), San Jose, California, June, 2000.

[6] Philip Bohannon, Juliana Freire, Prasan Roy, Jérôme Siméon and Bell Laboratories. *From XML Schema to Relations:A Cost-Based Approach to XML Storage*. The 18th International Conference on Data Engineering (ICDE'02),2002.

[7] B. S. manjunath, Philippe Salembier and Thomas Sikora, *Introduction to MPEG-7 : Multimedia Content Description Interface*, WILEY, ISBN : 0-471-48678-7, 2002.

[8] ISO/IEC 15938-2, *Information technology – Multimedia content description interface – Part 2 : Description definition language.*

[9] ISO/IEC 15938-3, *Information technology – Multimedia content description interface – Part 3 : Visual.*

[10] ISO/IEC 15938-4, *Information technology – Multimedia content description interface – Part 4 : Audio.*

[11] ISO/IEC 15938-5, *Information technology – Multimedia content description interface – Part 5 : Multimedia description schemes.*

[12] Fang Du, Sihem Amer-Yahia, Juliana Freire. *ShreX : Managing XML Documents in Relational Databases*. Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

[13] Daniela Florescu, Donald Kossmann. *A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Databases*. INRIA Rocquencourt.

[14] Yang Chu, Liang-Tien Chia, Sourav S. Bhowmick. *Looking at Mapping, Indexing & Querying of MPEG-7 Descriptors in RDBMS with SM3*. MMDB'04, Washington, USA.