효율적인 RDBMS 기반 XML Transformation을 위한 XML Data Statistics의 확장

이유진° 차재혁 오성교 이성연
한양대학교 정보통신대학원
eu98°@ihanyang.ac.kr, chajh@hanyang.ac.kr, platean@empal.com, yoniatssm@hotmail.com

Extension of XML Data Statistics for efficient XML transformation based RDBMS

Yujin Lee° , Jaehyuk Cha, Sungkyo Oh, Sungyoun Lee
The Graduate School of Informations & Communications, Hanyang University

요 약

XML 문서에 대한 데이터의 통계 정보는 XML 어플리케이션에 유용하다. 특히 XML 어플리케이션에 대해 RDBMS 테이블 형태로 유도하는 방법 중 cost-based approach를 적용할 때 다양한 Schema 변환 중 어플리케이션에 가장 적합한 것을 선택하는 데 사용한다. 본 논문에서는 정확한 통계 정보를 모으기 위해 Shared type과 변환 과정에 생기는 잠재적인 Shared type에 대해 해결한 X2R System을 개발하였고, 효율적으로 통계를 유지하도록 하였다.

1. 서 론

XML이 데이터 표현과 상호교환에 있어 중요한 매체가 됨과 동시에 XML 데이터의 사용량이 증가함에 따라 XML을 RDBMS에 저장하여 RDBMS의 좋은 속성을 재활용하기 위한 많은 매핑 방법들이 있다. 그 중 어플리케이션의 특징과 활용에 최적화된 매핑을 찾는 것은 쉬운 일이 아니기 때문에 어플리케이션의 특징을 가지고 있는 사용자 질의 집합에 대한 비용을 계산하여 최적화된 매핑 방법을 찾는 cost-based approach[1]을 많이 사용하게 되었다. 즉, 사용자 질의 집합에 대해 반환결과 행수를 계산하여 적은 비용의 매핑 방법을 선택하는 것이다. 이를 위해 질의 반환 결과 행수를 예측하는 *Result estimator*와 XML의 각각의 타입(complex type or base type) 대해 최소 값, 최대 값, 총 빈도수와 같은 통계를 정확히 저장하고 있는 것이 중요하다.

XML Result estimator인 StatiX[2]는 XML Schema를 이용하여 엘리먼트 타입 단위로 통계를 모은다. 이렇게 함으로써 변환 후에도 통계의 변환과 유지가 쉽도록 하는 특징을 가지고 있다. 또한 히스토그램을 이용하여 구조적으로 저장하는 특징을 가진다. 그러나 StatiX는 같은 엘리먼트가 동시에 여러 부모를 가지는 *shared type*[3]에 대해 변환 과정에서 정확한 통계를 유지하기 어렵다.

본 연구에서는 통계를 변환 이후에도 정확히 유지하도록 초기 변환 과정에서 Shared type에 대해 미리 처리하고, 이를 유지하기 위한 Unique ID의 사용과 구조화된 통계 저장 방법에 대해 설명하려고 한다.

2. 기존의 StatiX

*StatiX*는 XML *Schema*를 기초로 타입 단위로 통계를 모으고 히스토그램을 이용하여 구조적으로 저장을 하는 Validator 모듈과 테이블로 매핑하기 위한 여러 변환 방법을 적용하는 *Transformation* 모듈로 구성되어 있다. StatiX의 Validator를 통해 만들어진 통계를 Transformation 모듈의 변환을 적용한 뒤 사용자 질의에 대한 반환결과를 예측하여 가장 적은 비용을 가지는 변환 방법을 선택한 후에 테이블로 매핑 하는 시스템이다. 그러나 StatiX는 변환 과정에 생기는 잠재적인 Shared type에 대해 균등 분배를 하기 때문에 정확한 통계를 얻기 어렵다. 적은 비용을 선택하여 매핑방법을 선택해야 하는 cost-based approach의 경우 알맞은 매핑을 선택하지 않을 수도 있다.

2.1 Shared type

Shared Type이란 Schema에서 하나의 타입이 여러 개의 부모를 가지고 있는 것을 말한다. 아래의 그림에서 Aka라는 타입은 Show와 Episode 두 개의 부모를 가진다. StatiX 에서는 이러한 Shared type의 통계를 저장할 때 각각의 부모에 대해 따로 저장한다. 즉 Show를 부모로 하는 Aka의 통계와 Episode를 부모로 하는 Aka의 통계를 구분한다.

```
Type Show =
  show [title [String], Show_year, Aka*, Review*,
        (box_office [Integer] |
        (seasons [Integer], Episode*))]
Type Show_year = year [Integer]
Type Review = review [String]
Type Aka = aka [String]
Type Episode =
  episode [Aka {0,2}, guest_dir [String]*]
```

그림 1 Movie Data의 간략한 XML Schema

통계를 변환과정 마다 XML 문서에 대해 매번 다시 구할 수 없기 때문에 가장 작은 타입 단위로 통계를 저장하고 변환 후에 그에 따라 통계를 유도할 수 있도록 하고 있다. 그러나 merge 변환 후에 split변환을 하게 되면 통계의 정확성을 잃게 된다. 예를 들면, Show의 Aka(count 1)와 Episode의 Aka(count 7)를 merge한다면 Aka의 통계 중 count의 값은 8이 된다. 이 후에 다시 split을 한다면 Show와 Episode에 대해 몇 개의 count인지 알 수 없게 된다. StatiX에서는 균등분배를 원칙으로 하고 있지만 실제 정보와 달라지기 때문에 비용을 계산하여 판별하기에 좋지 않다.

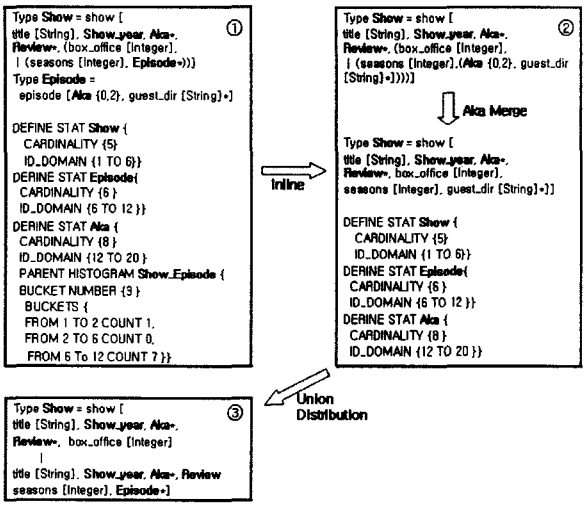

그림 2 Shared type인 Aka의 변환

그림 2는 '|(Choice)'에 대해 Union Distribution을 적용하였다. ②는 ①에서 Episode를 Show에 Inline 한 뒤 Aka를 Merge하였다. 이때의 통계는 ①의 통계를 통해 유도 가능하다. ③은 Show를 Union Distribution을 한 것이다. 이때 Aka의 통계만 살펴보면 Show와 Episode에 해당했던 각각의 통계를 구분 할 수 없다. 또한 box_office는 영화에 관련된 Show이고, season과 Episode는 TV에 관한 Show라 할 수 있다. 이에 대해 변환을 적용하면, Title, Show_year, Aka, Review와 같은 요소들이 box_office와 season, Episode에 대해 공동으로 포함된다. 변환 과정 중에 요소가 분리 되게 되는 이러한 요소를 잠재적인 Shared type이라 한다. 변환 중 통계를 잃어버리는 경우와 잠재적인 Shared type에 대한 처리가 정확한 통계를 저장하는데 고려해야 할 문제가 된다.

3. X2R System의 개발

기존 StatiX 모듈을 포함한 LegoDB와 비슷한 시스템을 구현하여 위와 같은 문제를 해결하였다.

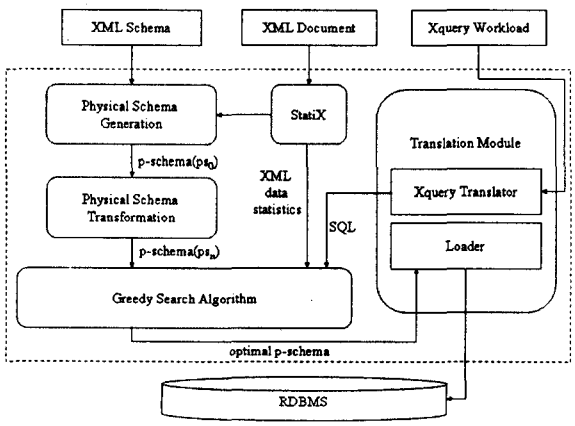

그림 3 X2R System의 아키텍처

XML Schema에 대해 Generator에서 Physical Schema를 생성한다. Physical Schema는 XML Schema를 RDBMS의 테이블로 매핑 하기 위한 중간 단계 Schema로 통계를 포함하고 있고, 변환이 가능 하도록 XML Schema보다 간략한 구조를 가지고 있다. StatiX는 실제 XML 문서의 통계를 이미 만들어진 Physical Schema에 대해 구조적으로 저장하는 기능을 한다. Transformation은 Physical Schema를 변환시키는 모듈로 inline&outline, split&merge, Union Factorization/Union Distribution의 방법을 적용하여 Schema구조를 변환하여 매핑 가능한 구조를 만든다. Translator의 XQuery Translator는 XQuery를 SQL로 변환하여 Greedy Search Algorithm 모듈의 Result estimator가 cardinality를 계산하도록 한다. 그리하여 Query에 대해 가장 적은 비용의 매핑을 선택하여 Translation 모듈의 Loader가 RDBMS의 테이블로 매핑 한다.

3.1 X2R System의 Generator

X2R System의 Generator는 Shared type에 대한 문제를 개선하도록 설계하였다. '|(Choice)'에 대해 Union Distribution을 미리 적용하여 잠재적인 Shared type에 대한 통계를 유지할 수 있도록 하였다. 아래 그림 4는 Medline이라는 미국 국립의학도서관 의학 및 생명의학 관련 서지 데이터 이다. 왼쪽의 XML Schema를 간략하게 표현한 것이다. 이것을 Generator가 Physical Schema으로 만든 것이다. 여기서는 Article의 Journal과 Book이 '|(Choice)'이다. 이를 Union Distribution을 적용하여 오른쪽과 같이 ArticlePart1, ArticlePart2로 변환하였다. Author와 Pagination의 자식 중 '|(Choice)' 또한 같은 변환을 적용하였다.

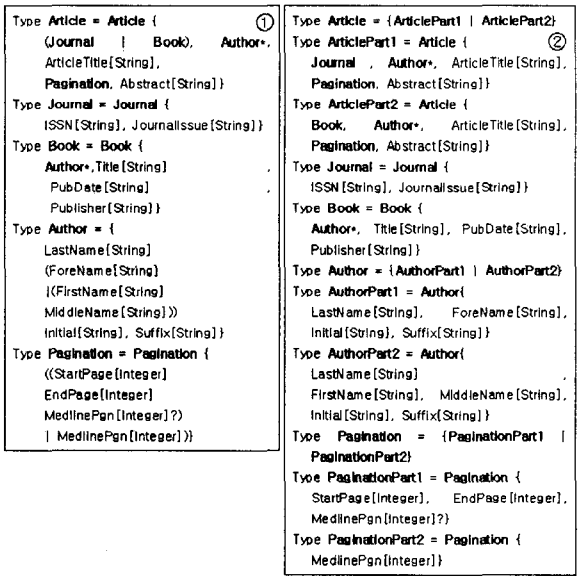

그림 4 Medline Schema와 Medline Physical Schema

3.2 X2R System의 StatiX Summary

Generator를 통해 만들어진 Physical Schema에 대해 샘플 XML 문서의 통계를 구한다. 통계는 최소 단위인 타입에 대해 모은다. 기존의 StatiX의 부모에 대한 히스토그램은 이미

Shared type에 대해 처리를 했기 때문에 필요가 없다. 또한 부모에 대한 통계를 알고 싶다면 ID를 이용하여 부모의 통계를 찾아볼 수 있다. 아래 그림 5와 같이 샘플 XML 문서 안에 몇 개 있는지 Count와 문자 데이터인 경우 길이를 나타내는 Size, 부모에 대해 복수 개 존재 할 경우 최대 몇 개, 최소 몇 개인지를 나타내는 Occur, 값에 대한 통계인 Value그리고 문자 데이터에 대한 빈도를 나타내는 OccurByValue를 통해 나타낸다.

element LastName {
ID : 1.1.2.1.1
Count [28]
Size [15]
Occur (minOccur[-1],maxOccur[4]),
Value (minValue[], maxValue[])
OccurByValue ([Deakin = 1],[Wu = 1], [Cope =1], [Oldfield = 1], [Licko = 1], [Rashevsky = 1], [Alinat = 1], [Perreau-Bertran = 1].....)}

그림 5 LastName에 대한 통계 예

3.3 X2R System의 ID 생성 및 유지

ID는 숫자와 점으로 구성되어 있다. 부모 자식간의 관계에 대해 점 하나를 추가하고 숫자는 부모의 숫자 뒤에 형제 순서대로 만든다. 아래 그림에서 ArticlePart1의 ID는 1.1이고 이것의 자식인 Journal은 1.1.1, 형제인 Author는 1.1.2이다. ID를 부여함으로써 인식하기 쉽고, ID와 통계와의 연관을 부모의 통계를 찾아보고자 할 때 용이하게 이용할 수 있다. 즉, 변환 이후 자신의 ID를 통해 부모의 통계 정보를 찾아보기 쉽다. 또한 테이블에 매핑 할 때 키 생성이나 XML 데이터를 입력할 편리하게 이용할 수 있다.

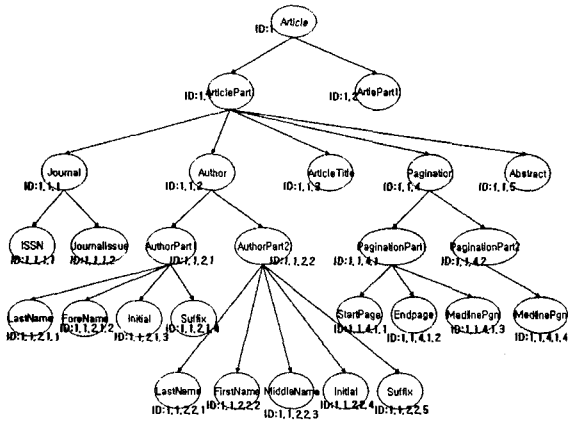

그림 6 ID를 포함하여 나타낸 Medline Physical Schema

3.4 X2R System의 Transformation

Generator를 통해 만들어진 Physical Schema를 테이블에 알맞도록 변환 시키는 모듈이다. 각 element에 대해 타입 네임(type name)을 주면 하나의 테이블로 구분된다. 반대로 타입 네임을 제거하면 테이블이 되지 않는다. 위의 Physical Schema중에 Article에 대한 테이블은 다음과 같이 된다.

Table Article [Article_PK]
Table ArticlePart1 [ArticlePart1_PK, Article_FK, ArticleTitle .....]
Table ArticlePart2 [ArticlePart2_PK, Article_FK, ArticleTitle .....]

Physical Schema를 기존의 Transformation 방법을 적용한 것은 다음과 같다.

Type ArticlePart1 = Article { ①
Journal , Author*, ArticleTitle[String],
Pagination, Abstract[String]}
Type Journal = Journal {
ISSN[String], JournalIssue[String]}
Online Journal →
Type ArticlePart1 = Article {
ISSN[String], JournalIssue[String],
Author*,ArticleTitle[String]
Pagination, Abstract[String]}

Type ArticlePart1 = Article { ②
Journal , Author*, ArticleTitle[String],
Pagination, Abstract[String]}
Split Repetition Article →
Type ArticlePart1 = Article {
Journal, Author?, Author*
ArticleTitle[String]
Pagination, Abstract[String]}
Type Author = Author {
LastName[String]
ForeName[String].....}

Type Author ={AuthorPart1 | AuthorPart2}
Type AuthorPart1 = Author{ ③
LastName[String], ForeName[String],
Initial[String], Suffix[String]}
Type AuthorPart2 = Author{
LastName[String]
FirstName[String]
MiddleName[String],Initial[String],
Suffix[String]}
Union Factorization Author →
Type Author = Author{
LastName[String], ForeName[String],
Initial[String], Suffix[String]
|
LastName[String]
FirstName[String], MiddleName[String],
Initial[String], Suffix[String]}
→
Type Author = Author{
LastName[String],(ForeName[String],
|
FirstName[String], MiddleName[String]
), Initial[String], Suffix[String]}
Merge Author →
Type Author = Author{
LastName[String], ForeName[String],
FirstName[String], MiddleName[String]
Initial[String],Suffix[String]}

그림 7 여러 Transformation 적용한 예

①은 Online을 적용한 것으로 Journal의 자식들을 ArticlePart1에 포함시켰다. ②는 복수개의 Author를 Author?와 Author*으로 변환하였다. ③은 Union Factorization과 Merge를 적용한 것으로 이미 작은 단위로 통계를 구해 변환을 하기 때문에 대부분 Split변환보다 Merge를 하는 방향으로 진행된다.

4. 결 론

X2R System을 개발을 통해 정확한 XML 데이터 통계를 모으는 방법을 고안하였다. 특히 Shared type과 변환 과정에 생기는 잠재적인 Shared type에 대해 미리 처리하여 최소 단위의 타입에 대해 통계를 모으기 때문에 변환 적용 후에도 통계가 정확하게 유지된다. 또한 ID는 처음 Generator에 의해 생성된 Physical Schema에 부여된 것으로 변환 이후에도 부모와 형제타입을 ID를 통해 알 수 있기 때문에 여러 변환을 적용 하더라도 쉽게 통계를 유도 할 수 있다.

5. 참고문헌

[1]Phil Bohannon, et. al, From XML Schema to Relations: A Cost-Based Approach to XML Storage Proc. of the 18th Int'l Conference on Data Engineering (ICDE'02) Feb 26-01. pp 64, 2002

[2]J. Freire et.al, Statix: Making XML count. Proc. of SIGMOD, Feb. pp 181-191, 2002

[3]Maya Ramanath, et. al, Searching for Efficient XML-to-Relational, Proc. of the VLDB pp.19-36. 2003