데이터베이스 역공학을 이용한 데이터 구조 추출에 관한 연구

황태희, 김미화, 배석찬
군산대학교 컴퓨터과학과
e-mail:thwang@cs.kunsan.ac.kr

A Study on The Data Structure Extraction using Database Reverse Engineering

Taehee Hwang, Mihwa Kim, Seokchan Bae
Dept. of Computer Sciences, Kunsan University

요약

데이터베이스 역공학은 소프트웨어의 융통성을 개선하여 과거의 개발들을 재사용하고 유지 비용을 감소하는데 목적이 있다. 파일과 데이터베이스 구조의 의미 기술을 회복하는 것은 역공학의 중요한 측면이다. 이것은 데이터베이스의 정확한 데이터 구조와 무결성 제약을 발견하는 데이터 구조 추출을 포함한다. 본 논문에서는 관계 데이터베이스에 초점을 두고 소프트웨어 유지 보수를 용이하게 하기 위하여 역공학을 이용한 데이터 구조 추출 방법을 제안한다

1. 서론

데이터베이스 역공학(DBRE)은 응용의 파일과 데이터베이스를 이해하고 재문서화하기 위한 노력을 통한 소프트웨어 공학 과정이다. 이 과정은 처리 구성요소와 사용자 인터페이스를 포함하는 전체 응용의 역공학에서 첫 번째 단계에 제안될 수 있다[1].

DBRE는 데이터베이스의 완전한 논리 스키마와 개념 스키마를 산출하는 것이다. 이 스키마들은 현재 데이터베이스 방법론에서 발견되므로 데이터 추상화에 있어서 표준이다[2].

논리 스키마에 의해, 그들은 데이터 매니저에 의해 구현되고, 응용 프로그래머에 의해 보이는 데이터 구조의 기술을 의미한다.

데이터 구조를 이해하는 문제는 오래되고, 불완전하게 설계되고, 빈약하게 문서화된 응용을 다룰 때 특히 복잡하다. 데이터 구조 추출과정은 모든 명시적이고 암시적인 구조와 속성들을 문서화하기 위해 완전한 논리 스키마를 보강하는 것에 목적이 있다. 이 과정의 주된 문제는 많은 구조와 속성들이 프로그램의 절차 절에서 제어되고 관리되지만, 명시적으로 선언되지 않고 암시적이라는 사실이다.

본 논문에서는 관계 데이터베이스에서 역공학을 이용하여 데이터 구조를 추출해 내는 방법을 제안한다. 본 논문의 구성은 다음과 같다. 2장은 관련연구를 기술한다. 3장은 데이터베이스 역공학을 통해 데이터 구조를 추출하는 과정을 제안하고 4장은 본 논문의 결론과 앞으로의 연구방향에 대하여 언급한다.

2. 관련연구

데이터베이스에서 관계를 추출하는 것과 개개의 데이터 설정을 분석하는 것은 상당히 큰 문제이다[3].

Kett[4]는 데이터베이스에서 엔티티들의 클러스터를 생성시키는 과정을 처리하기 위해 객체-관계 모델에서 일대다 관계를 사용한다. 이 작업은 정확한 데이터 모델을 이용하는 것을 가정한다.

Davi[5]은 개체-관계 모델로 관계 데이터베이스 스키마의 역공학에 대한 체계적인 접근을 제안한다. 그들의 목적은 관계 데이터베이스 스키마에서 개체-관계 스키마로의 전환에 있다. 이 접근은 상속을 무시하고 입력 관계 데이터베이스 스키마는 3정규형이어야만 하며 어떤 동의어와 동음이의어도 존재하지 않는다는 제한이 있다.

데이터베이스 모델을 분석하는 문제는 넓게 연구되어진다. 하나의 데이터베이스 모델을 다른 것에 사상하는 방법은 데이터베이스의 기초적인 구조를 확인하는 것을 수반하기 때문에 지식 발견에 유용하다. 그러므로 이용할 수 있는 소스에서 필요한 정보를 찾아 그것의 타입에 개의치 않고 처리할 수 있는 이상적인 데이터베이스 역공학 방법이 요구된다.

3. 데이터 구조 추출

데이터 구조 추출의 과정은 그림 1과 같다.

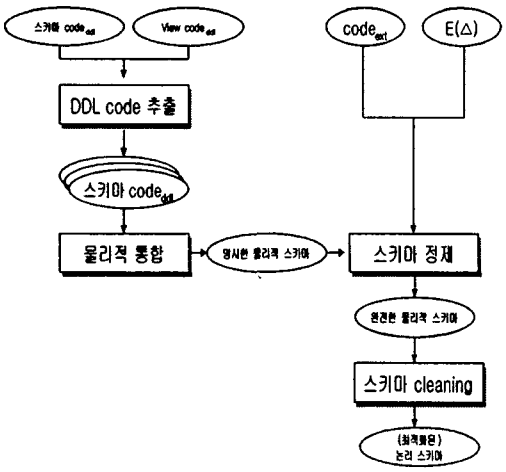

그림 1. 데이터 구조 추출 과정

이 과정은 모든 암시적이고 명시적인 구조와 제약을 포함하는 완전한 데이터 관리 시스템 스키마를 회복하는 것이다. 읽을 수 있고 처리할 수 있는 형태에서, 제대로 된 데이터베이스 시스템은 일반적으로 이 스키마의 기술(data dictionary contents, DDL texts, 등)을 보완한다. 문제는 대부분의 경우에 그들 구조의 컴퓨터화된 설명이 존재하지 않기 때문에 표준화된 파일에 비해 훨씬 복잡하다는 것이다. 각각 소스 프로그램의 분석은 단지 파일과 레코드 구조의 부분적인 뷰만 제공한다. 대부분의 실세계 응용에서 분석은 프로그램들에 선언된 레코드 구조의 단순한 검출 이상이어야 한다.

본 논문에서는 세 개의 파일을 사용하는 작은 COBOL 프로그램을 원시 데이터로 사용하여 COBOL 파일의 집합이 관계 테이블로 변환되는 사례 연구를 기술한다.

3.1 DDL code 분석

이 과정은 파일과 레코드 형을 추출하는 COBOL 파서에 의해 수행되고, CASE 도구의 저장소에서 입력된 그대로의 물리적 스키마로서 나타낸다. 이는 스키마 스크립트와 응용 프로그램에서 포함하는 데이터 구조 선언 문장을 분석하는 것에 본질이 있다. 이 과정은 파일제어, 파일 정의와 기본 식별자를 산출하는 Environment Division과 레코드 타입 구조를 제공하는 Data Division의 File Section을 분석한다. 이것은 first-cut 논리 스키마를 산출한다(그림 2).

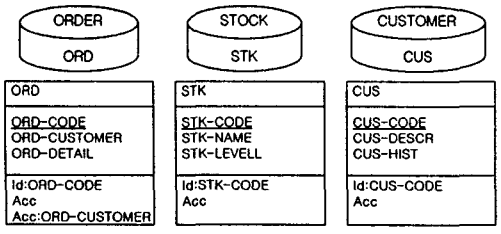

그림 2 명시적 파일과 레코드 스키마

스키마의 필드의 타입과 길이를 보여주는 소스이다(그림 3).

```
Schema CUST-ORD/Raw physical

collection CUSTOMER
    CUS
collection ORDER
    ORD
collection STOCK
    STK

entity type CUS in CUSTOMER
                CUS-CODE char(12)
                CUS-DESCR char(80)
                CUS-HIST char(1000)
                id:CUS-CODE, access key

entity type ORD in ORDER
                ORD-CODE num(12)
                ORD-CUSTIMER char(12)
                ORD-DETAIL char(200)
                id:ORD-CODE, access key
                access key:ORD-CUSTOMER

entity type STK in STOCK
                STK-CODE num(5)
                STK-NAME char(100)
                STK-LEVEL num(5)
                Id:STK-CODE, access key
```

그림 3. 필드의 타입과 길이

각각 레코드 타입은 물리적 엔티티 타입에 의해, 그리고 각각 필드는 물리적 속성에 의해 표시된다. 레코드 키는 명확하게 유일한 제약일 때 식별자에 의해 그리고 명확한 인덱스 일 때 액세스 키에 의해 표현한다. 파일은 물리적 엔티티 집합으로 표현한다.

3.2 스키마 정제(Refinement)

이 과정은 여러 가지 정보 소스가 암시적이거나 분실한 구조의 증거를 검사하는 복잡한 작업이다. 지금까지 획득한 명시적 물리 스키마는 이 구조로 풍부하게 되며, 완전한 물리 스키마를 이끌어 낸다.

스키마는 프로그램이 데이터를 사용하고, 관리하는 방법의 철저한 점검을 통하여 정제한다. 이 과정을 통해서

레코드 타입과 필드의 매우 작은 구조를 찾는다.

암시적인 선택사항, 혼합되거나 다중치 필드를 찾는다.

다중 필드와 레코드 구조를 찾는다(변경과 재정의).

순차적이고 상대 파일에서 잘못된 레코드 식별자를 찾는다.

복합 다중치 필드의 식별자를 찾는다.

다중치 필드로 내장된 것을 포함하는 외래키를 찾는다.

선택 필드에 따라서 존재하는 제약을 찾는다.

다중치 필드의 최소-최대 Cardinality를 정제한다.

여분의 것을 확인한다.

값 도메인에 제약과 열거된 값 도메인을 찾는다.

여기서 필드 구조, 외래 키, 그리고 필드 Cardinality의 3가지 중요한 구조를 고려한다.

3.2.1 필드 구조

몇몇 필드는 매우 길다(CUS-DESCR, CUS-HIST, ORD-DETAIL, STK-NAME). 그것을 더 정제하기 위하여 첫 번째 CUS-DESCR을 고려한다. CUS-DESCR에 관하여 데이터 흐름을 요약하는, 변수 종속 관계 그래프를 만든다:

CUS.CUS-DESCR <-->DESCRIPTION

이 그래프는 CUS-DESCR과 DESCRIPTION이 같은 값을 공유하고 같은 구조를 갖는 것을 의미한다(그림 4).

```
01 DESCRIPTION
  02 NAME PIC X(20).
  02 ADDRESS PIC X(40).
  02 FUNCTION PIC X(10).
  02 REC-DATE PIC X(10).
```

그림 4. DESCRIPTION 구조

이 구조는 논리적 스키마에서 필드 CUS-DESCR과 연관되며 CUS-HIST, ORD-DETAIL 그리고 STK-NAME에 대하여 같은 방법으로 계속 분석한다(그림 5).

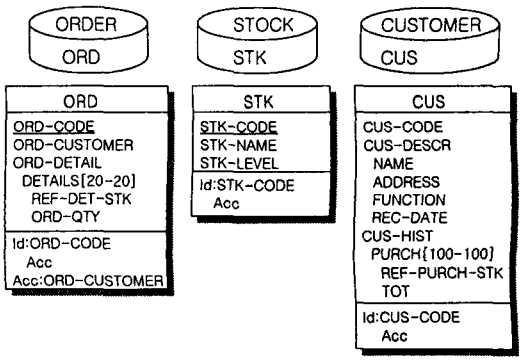

그림 5. 필드 정제 과정의 결과

3.2.2 외래 키 유도

참조 링크가 레코드 타입 중에 존재해야 한다. 예를 들어 필드 ORD-CUSTOMER을 조사한다:

· 그것의 이름은 파일의 이름을 포함한다(CUSTOMER).

· 그것은 CUSTOMER의 레코드 키와 같은 타입과 길이를 가진다.

· 그것은 액세스 키에 의해 지원된다(즉, INDEX).

· 그것의 종속관계 그래프는 CUSTOMER의 레코드 키로부터 그 값을 받는다.

· 그것이 저장되기 위한 ORD 레코드로 이동하기 전에, 그 사용 패턴은 ORD-CUSTOMER 값이 저장된 CUS 레코드를 확인하는 프로그램을 조사한다(그림 6).

```
NEW-ORD.
 . . .
 MOVE 1 TO END-FILE.
 PERFORM READ-CUS-CODE UNTIL END-FILE = 0.
 . . .
 MOVE CUS-CODE TO ORD-CUSTOMER.
 . . .
 WRITE ORD INVALID KEY DISPLAY "ERROR".
 . . .
READ-CUS-CODE.
 ACCEPT CUS-CODE.
 MOVE 0 TO END-FILE.
 READ CUSTOMER INVALID KEY
 DISPLAY "NO SUCH CUSTOMER"
 MOVE 1 TO END-FILE
```

그림 6.CUS 레코드를 확인하는 소스

이것은 ORD-CUSTOMER가 외래 키라고 확신하는 다섯 가지 명백한 증거이다. 같은 방법에 의하여, ORD-DETAIL.DETAILS.REF-DET-STK는 STOCK을 위한 다중 값을 갖는 외래 키이다. 이름의 REF 부분은 필드의 참조를 의미하고, CUS-HIST.PURCH.REF-PURCH -STK는 STOCK을 위한 다중치 외래 키 이다. 이 스키마는 그림 7과 같다.

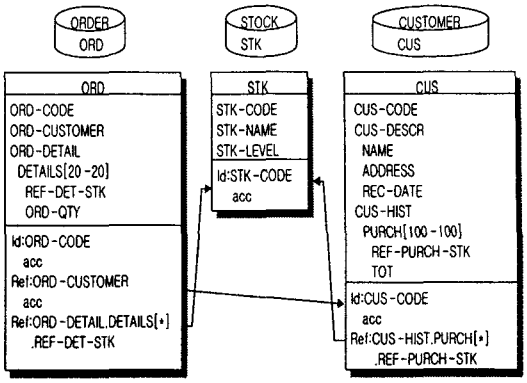

그림 7. 명시한 외래키

3.2.3 다중치 속성의 cardinality 세분

다중치 필드는 occurs구로부터 추론한 주어진 cardinality 제약을 가진다. 최대 cardinality는 주어지지만, 최소 cardinality에 관해서는 주어지지 않는다. 일반적으로 새로운 CUS 레코드를 이야기하는 것은 CUS-HIST.PURCH을 포함한, 각각 필드에 초기치를 설정하는 것이다. 이것은 REF-DET-STK에 0을 설정하는, INIT-HIST를 통해서 한다. 게다가, 이 LIST의 스캐닝은 0의 값이 주어질 때 정지한다. 결과적으로 LIST에 0에서 100까지 요소가 있다. 그림 8의 유사한 분석은 ORD-DETAIL.DETAILS의 cardinality를 정제하는 것이며 그림 9의 스키마이다.

```
MAIN.
  PERFORM PROCESS
  UNTIL CHOICE = 0.

PROCESS.
  PERFORM NEW-ORD.

NEW-ORD.
  SET IND-DET TO 1.
  PERFORM READ-DETAIL
    UNTIL END-FILE = 0
    OR IND-DET = 21.
  WRITE ORD
    INVALID KEY DISPLAY "ERROR".

READ-DETAIL.
  PERFORM READ-PROD-CODE.
READ-PROD-CODE.
  PERFORM UPDATE-ORD-DETAIL.
UPDATE-ORD-DETAIL.
  MOVE 1 TO NEXT-DET.
  PERFORM UNTIL
    REF-DET-STK(NEXT-DET)
      = PROD-CODE
  OR IND-DET = NEXT-DET
    ADD 1 TO NEXT-DET
END-PERFORM.
  IF IND-DET = NEXT-DET
  MOVE PROD-CODE
  TO REF-DET-STK(IND-DET)
  SET IND-DET UP BY 1
  ELSE
  DISPLAY "ERROR : ...".
```

그림 8. ORD-DETAIL.DETAILS의 cardinality를 정제

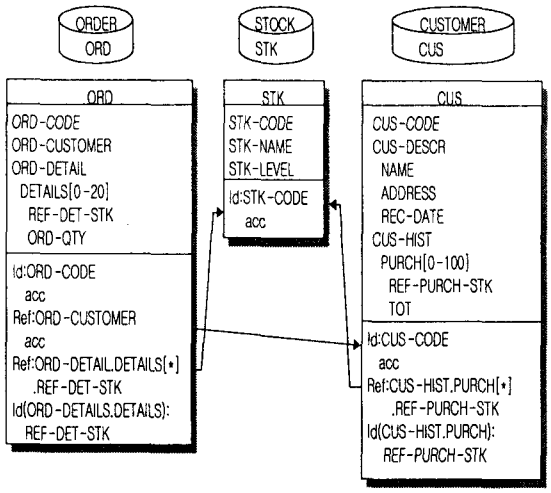

그림 9. 정제된 COBOL-논리적 스키마

3.3 스키마 클리닝

물리적 구성, 즉 파일과 액세스 키는 더 이상 유용하지 않고 삭제한다(그림 10).

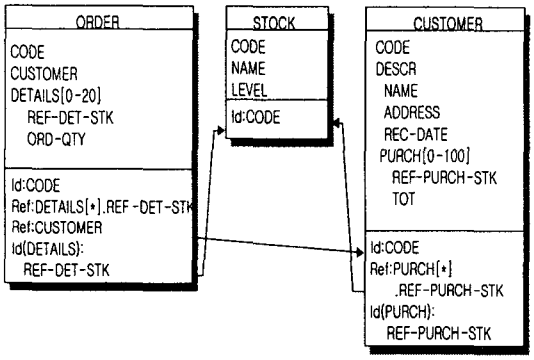

그림 10. 완전한 COBOL 논리 스키마

4. 결론

데이터베이스 역공학은 유연성 있고 대화식 접근이 필요하며, 후보키와 외래키의 결정은 역공학에 있어서 중요한 단계이다. 비표준화된 프로그래밍 스타일과 데이터 관리 시스템의 기술상 한계 때문에 데이터에 관한 구조상 정보의 중요한 부분이 프로그램들의 절차 코드와 사용자 인터페이스에서 그리고 데이터 자신에서 매설될 수 있다. 본 논문은 데이터베이스 역공학을 이용한 데이터 구조 추출과정에서 명시적이고 암시적인 구조에서 외래키를 추출하는 것을 제안하였다. 외래키를 찾기 위한 가설을 제안하는 대부분의 도구는 "필드가 레코드 타입 R의 식별자에 같은 이름을 갖는다면, 그 R을 외래 키로 선언한다"는 식의 원시적인 방법을 사용한다. 이 규칙은 대부분의 외래키를 무시할 수 있고, 우연히 같은 이름의 필드가 외래 키로 선언될 수 있다. 이러한 결과는 불완전한 논리 스키마를 산출할 수 있고 불완전한 개념상의 스키마로 번역될 수 있다. 그러므로 앞으로는 본 논문에서 제안된 데이터 구조 추출에서, 특히 외래키 추출을 역공학 도구에 이용하여 분석작업과 역공학 도구의 지원을 암시적인 구조까지 확장하는 연구가 요구된다.

참고문헌

[1] Hainaut, J-L, Henrard, J., Roland, D., Englebert, V., Hick J-M, "Structure Elicitation in Database Reverse Engineering", Proc. of the IEEE Working Conf. on Reverse Engineering, Monterey, Nov. 1996. IEEE Computer Society Press, 1996.

[2] Batini, C., Ceri, S., Navathe, S., "Conceptural Database Design An Entity-Relationship Approach", Benjamin/Cummings, 1992.

[3] Piatetsky-Shapiro, G., Frawley, W.J. "Knowledge Discovery in Database", AAAI/MIT Press,1991.

[4] Ketterlin, A., Gancarski, P., Korczak J. J. "Conceptual Clustering in Structured Database: A Practical Approach", in Proceeding of the First International Conference on Knowledge Discovery and Data Mining(KDD-95), U.M.Fayyad, R. Uthurusamy (eds), AAAI Press, pp.63-68,1995.

[5] Kathi Hogshead Davis and Adarsh K. Arora. "Converting a relational database model into an Entity-Relationship mode". Proceedings of the Sixth International Conference on Entity-Relationship Approach, November 9-11, 1987.