자료흐름도를 사용한 테이블 설계순서 추출기의 설계 및 구현†

(Design & Implementation of Extractor for Design Sequence of DB tables using Data Flow Diagrams)

임 은 기*
(Eun-ki Lim)

요 약 현재 운용되고 있는 리가시 시스템에 대한 요구명세서는 대부분 자료흐름도를 사용하고 있어, 시스템의 유지 보수를 위해서는 자료흐름도로부터 획득한 정보에 의존하지 않을 수 없다. 본 논문에서는 자료흐름도로부터 데이터베이스 테이블 설계순서를 추출하는 추출기를 설계, 구현하였다. 추출기는 자료흐름도를 입력 받아 저장하고, 이를 방향그래프로 변환하여 데이터베이스 테이블 설계순서를 추출하여 제시한다. 구현된 추출기는 실제 운용 중인 소프트웨어 시스템에 적용함으로써 실제 적용가능성을 보였다.

핵심주제어 : 구조적 분석, 자료흐름도(DFD), 설계 순서

Abstract Information obtained from DFD(Data Flow Diagram) are very important in system maintenance, because most legacy systems are analyzed using DFD in structured analysis. In our thesis, we design and implement an extractor for design sequence of database tables using DFD. Our extractor gets DFDs as input data, transform them into a directed graph, and extract design sequence of DB tables. We show practicality of our extractor by applying it to a s/w system in operation.

Key Words : Structured Analysis, Data Flow Diagram, Design Sequence

1. 서 론

지금까지 많은 분석 및 설계 기법이 개발되어 소프트웨어 시스템 개발에 적용되어 왔으나, 객체지향 기법이 본격적으로 사용되기 전까지 가장 널리 사용된 것은 구조적 분석 기법[1]이라 할 수 있다. 현재 운용되고 있는 legacy system의 요구명세서는 대부분 자료흐름도를 사용하여 작성되었기 때문에, 이들 시스템의 유지보수를 위해서는 자료흐름도로부터 획득한 정보에 의존하지 않을 수 없다. 이러한 까닭으로 많은 연구자들이 자료흐름도로부터 필요한 정보를 추출하기 위한 연구를 수행하였다. 대표적인 사례로는 Kuo의 DOE 방법[2], Guo의 CAST[3]. Fries의 UML 다이어그램 생성기법[4]을 들 수 있다.

본 논문에서는 자료흐름도로부터 데이터베이스 테이블 설계순서를 추출하는 추출기를 설계, 구현하였다. 소형의 정보시스템은 데이터베이스의 규모가 작기 때문에 테이블 설계상 별다른 어려움이 없으나, 복잡한 정보시스템의 경우에는 테이블의 개수가 수백 개에

† 이 논문은 금오공과대학교 학술연구비에 의하여 연구된 논문입니다.
* 금오공과대학교 컴퓨터소프트웨어공학과

이르고 자료의 생산 및 소비 관계가 복잡하기 때문에 테이블 설계 작업이 매우 어려워진다. 이러한 상황에서 데이터베이스 테이블을 적당하게 설계한다면 여러 번의 시행착오는 물론이고, 설계 작업에 엄청난 시간을 투입해야 할 것이다[5]. 본 논문에서 구현한 추출기는 데이터베이스 테이블을 설계하는 과정에서의 시행착오를 최소화하고 설계 작업을 효율적으로 진행할 수 있도록 한다.

본 논문의 2장에서는 추출기의 설계에 대해서 논의하며, 3장에서는 추출기의 구현 방안에 대하여 상세히 기술한다. 4장에서는 현재 운용 중인 소프트웨어 시스템에서 발췌한 실제 사례에 대해 추출기를 적용하여 테이블 설계순서를 추출한 결과를 제시하며, 5장에서는 연구결과를 요약한 후 차후 연구 과제를 제시한다.

2. 추출기의 설계

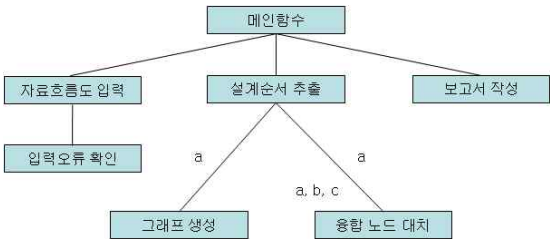
2.1 추출기 설계원칙

추출기는 중형 이상의 정보시스템에서 노련한 개발자가 사용하는 것을 전제로 하므로 다음과 같은 원칙에 따라 설계하였다. 첫째, 빈번한 요구변경에 쉽게 적응할 수 있도록 하기 위해서 자료흐름도를 저장하는 기능을 포함하여야 한다. 둘째, 설계 작업에 필요한 자료를 산출하기 위한 시스템이므로 처리속도에 대한 제약은 거의 없다. 셋째, 추출기의 사용자는 시스템 개발경험이 풍부한 고급 엔지니어이므로 사용자 인터페이스가 다소 복잡하더라도 작업 화면의 개수를 최소화하는 것이 무난하다.

2.2 추출기의 기본 기능

2.1절에서 논의한 설계원칙에 따라 추출기가 장착하여야 할 기능은 다음과 같으며, 추출기의 기능은 <그림 1>의 시스템구조도에 표현하였다.

- 자료흐름도 입력기능
- 테이블 설계순서 추출기능
- 보고서 작성기능



<그림 1> 추출기의 시스템구조도

자료흐름도 입력기능은 추출기가 자료흐름도에 관한 정보를 입력/저장하는 기능이다. 먼저 입력 기능은 자료흐름도의 구성요소인 프로세스, 자료저장소, 자료흐름을 입력하도록 설계하였으며, 자료 출처(data source) 및 자료 도착지(data sink)는 테이블 설계순서를 추출하는데 필요하지 않으므로 입력 대상에서 제외하였다. 자료흐름은 출발점 및 도착점으로 표현 가능하며, 자료흐름도는 요구변경에 의해서 변경될 수 있으므로 데이터베이스에 저장하여야 한다. 자료흐름도를 입력하는 과정에서 오류가 발생할 수 있으므로 입력기능의 하위 모듈로 입력오류 확인 모듈을 포함시켰다.

테이블 설계순서 추출기능은 방향그래프 생성 모듈과 융합노드 대치 모듈을 하위 모듈로 갖는다. 여기서, 융합노드란 위상정렬을 수행하는 과정에서 사이클이 발견된 경우에 사이클을 형성하는 노드들을 대신하는 역할을 담당한다.

보고서 작성기능은 추출한 설계순서 정보를 사용자에게 제시하는 기능이며, 복잡도가 높지 않으므로 별도로 기술하지 않는다.

2.3 자료흐름도 및 설계순서 저장을 위한 테이블 구조

자료흐름도에 대하여 기록해야 하는 구성요소는 프로세스, 자료저장소, 그리고 자료흐름이며, 이들 각각에 관한 정보와 추출한 테이블 설계순서를 저장하는데 사용되는 데이터베이스 테이블의 구조는 <표 1>와 같다.

<표 1> 테이블별 칼럼 구성

테이블명	칼럼 구성
processes	system_id, id, name
data_stores	system_id, id, name
data_flows	system_id, id, name, 출발점, 도착점
design sequence	system_id, 일련번호, id

위에서 '출발점'은 자료흐름이 출발하는 구성요소를 가리키며, '도착점'은 자료흐름이 도착하는 구성요소이다. '출발점'과 '도착점'은 프로세스 또는 자료저장소의 'id'를 값으로 가지며, 2개 모두가 자료저장소인 경우는 없다. '일련번호'는 테이블 설계순서를 가리키며, 동일한 순서를 갖는 테이블이 여러 개 있을 수 있다.

2.4 설계순서 추출을 위한 자료구조

설계순서 추출 모듈에서 사용하는 주요자료는 방향그래프 및 융합노드 목록이다. 설계순서 추출은 위상정렬 알고리즘을 골격으로 사용하므로 방향그래프는 인접 리스트로 표현하였다. 융합노드는 사이클을 형성하는 노드를 대치하는 역할을 담당하므로 다양한 자료구조를 선택할 수 있으나, 본 논문에서는 연결리스트를 사용하였으며, 사용한 자료구조는 다음과 같다.

· head node for adjacency list

type	deleted	p_count	node_id	first

* type->1: 프로세스, 2:자료저장소, 3:융합노드
* node_id->프로세스 또는 자료저장소의 id.
융합노드는 UL의 인덱스의 음수 값.

· edge node for adjacency list

node_id	next

· directed graph G
-> array of head nodes

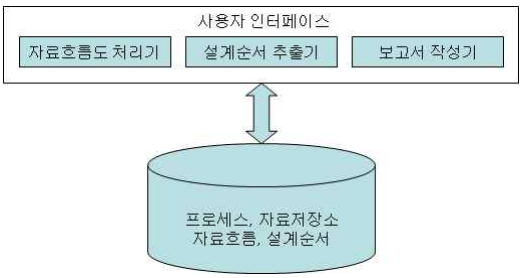
· unified node list UL
-> array of pointers to edge node

3. 추출기의 구현

3.1 추출기의 구성

추출기는 <그림 2>와 같이 구성되어 있다. 사용자 인터페이스, 자료흐름도를 입력하는 '자료흐름도 처리기', 그리고 보고서 작성기능을 담당하는 '보고서 작성기'는 'My Builder'로 구현하였고, 데이터베이스 접근을 위하여 'Oracle 9i'를 사용하였다. 테이블 설계순서를 추출하는 '설계순서 추출기'는 C언어로 구현하여, 'MyBuilder'에서 OLE 객체로 호출하도록 하였다.

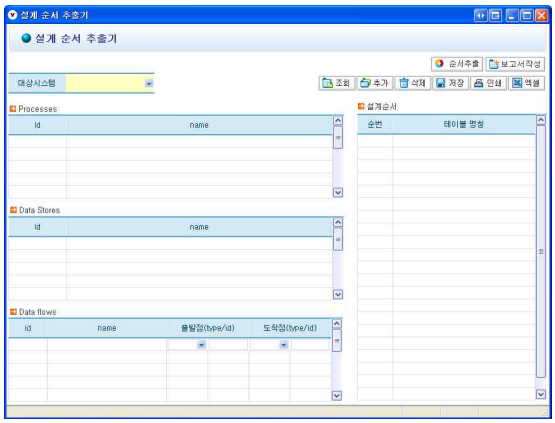
'자료흐름도 처리기'에서 '입력오류 확인' 모듈은 3.3절에서 기술하였으며, 나머지는 단순한 입력 기능이므로 상세한 기술은 생략하였다. '설계순서 추출기'는 테이블 설계순서 추출 알고리즘(부록 #1 참조)[5]을 기반으로 구현하였으며, 하위 모듈인 '그래프 생성' 모듈과 '융합노드 대치' 모듈은 각각 3.4절 및 3.5절에서 기술하였다. '보고서 작성기'는 단순한 출력 기능이므로 상세한 기술은 생략하였다.



<그림 2> 추출기의 구성

3.2 사용자 인터페이스

사용자 인터페이스는 <그림 3>과 같이 구성하였다. '설계순서 추출기' 및 '보고서 작성기'는 명령 버튼으로 처리하였고, 시스템별로 처리할 수 있도록 하기 위해서 처리대상 시스템을 선택하는 콤보박스를 설치하였다. 화면의 좌단에는 자료흐름도에 관한 정보를 처리하는 영역을 설정하였으며, 화면의 우단에는 보고서 작성기에 출력될 내용을 제시하도록 설계하였다. 화면의 전반적인 구성은 가장 보편적인 형태를 유지하였다.



<그림 3> 추출기의 사용자 인터페이스

3.3 입력오류 확인 모듈

자료흐름도를 입력하는 과정에서 발생 가능한 오류는 다음의 3가지 경우로 한정된다.

- 프로세스가 자료흐름에 전혀 사용되지 않는 경우
- 자료저장소가 자료흐름에 전혀 사용되지 않는 경우
- 자료흐름의 출발점과 도착점이 모두 자료저장소인 경우

이러한 형태의 입력오류를 확인하는 모듈의 함수 헤더와 처리 알고리즘은 다음과 같다.

- 함수명: check_input_error()
- input parameter: none
- return value: none

```
// 프로세스 및 자료저장소 각각에 대해서
// 자료흐름에 사용되는 횟수를 계산하여,
// 사용되는 횟수가 0이면 오류메시지 출력
for each record in processes {
  select count(*) from data_flows into n
  where 출발점=id or 도착점=id;
  if (n = 0) display error message;
}

for each record in data_stores {
  select count(*) from data_flows into n
  where 출발점=id or 도착점=id;
  if (n = 0) display error message;
}

// 자료흐름 각각에 대해서 출발점과
// 도착점이 모두 자료저장소이면
// 오류 메시지 출력
for each record in data_flows {
  select count(*) from data_stores into n1
  where id=출발점;
  select count(*) from data_stores into n2
  where id=도착점;

  if (n1>0 and n2>0)
    display error message;
}
```

3.4 방향그래프 생성 모듈

데이터베이스 테이블에 저장되어 있는 자료를 읽어 방향 그래프를 생성하는 모듈의 함수 헤더와 처리 알고리즘은 다음과 같다.

- 함수명: construct_graph()
- input parameter: directed graph G
- return value: count of stored nodes

```
// 프로세스 및 자료저장소 각각에 대해서
// 방향그래프 G의 head node 1개씩을
// 할당하고 head node를 초기화
i=0;
for each record in processes {
  // 노드 G[i]를 초기화
  // i 값을 증가
}

for each record in data_stores {
  // 노드 G[i]를 초기화
  // i 값을 증가
}

// 자료흐름 각각에 대해서 다음 처리를 한다
for each record in data_flows {
  // edge node 생성
```

```
  new = make_edge_node();
  new->node_id='도착점';
  new->next=NULL;

  // 'node_id=출발점'인 G의 원소를 찾아서
  // new node를 삽입
  i = find_index_of_G(출발점);
  new->next=G[i].first;
  G[i].first=new;

  // 'node_id=도착점'인 G의 원소를 찾아서
  // p_count의 값을 증가
  j = find_index_of_G(도착점);
  G[j].p_count++';
}

// G에서 p_count=0인 node를 스택에 삽입
for (i=0; i<node_count; i++) {
  if (G[i].p_count=0)
    push it into stack;
}

// head nodes의 개수를 반환
return count of head nodes;
```

3.5 융합노드 대치 모듈

'설계순서 추출기'는 설계순서를 추출하는 과정에서 사이클이 존재하면, 사이클을 형성하는 노드를 연결리스트에 저장한다. 이것을 융합노드 목록인 'UL'에 등록한 후, 융합노드 대치 모듈을 호출하여 사이클을 형성하는 노드를 융합노드로 대치시킨다. 융합노드 대치 모듈의 함수 헤더와 처리 알고리즘은 다음과 같다.

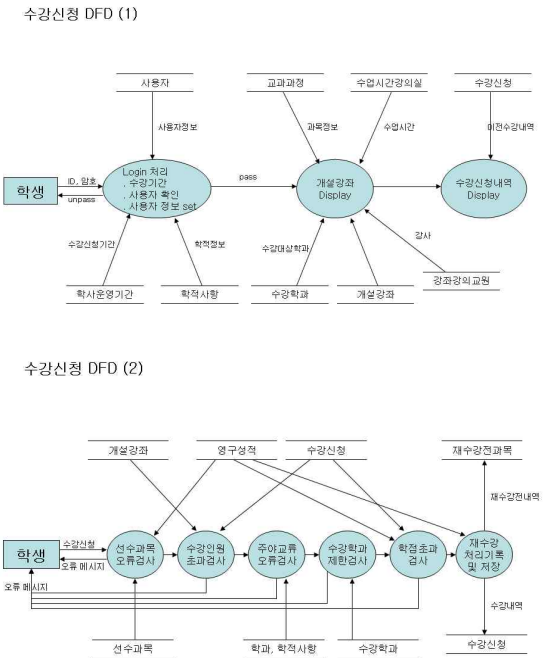
- 함수명: replace_cycle()
- input parameter: directed graph G,
  linked list of cycle nodes C,
  index for unified node UID
- return value: directed graph G

```
// 융합노드를 G에 등재
G[n].type=3;
G[n].deleted='N';
G[n].p_count=0;
G[n].node_id=-UID;
G[n].first=NULL;
n++;                  // n: G에 등록된 노드 개수

// 사이클 노드 리스트 C의 각 노드에 대해서
// 다음의 처리를 한다.
ptr = C;
while (ptr != NULL) {
  // 'node_id=출발점'인 G의 원소를 찾아서
  // 삭제 처리
  i = find_index_of_G(ptr->node_id);
  G[i].deleted = 'Y';

  // G[i]의 edge list에 있는 노드를
  // 융합노드 G[n]의 edge list에 삽입
  ptr1 = G[i].first;
  while (ptr1 != NULL) {
    j = ptr1->node_id;

    // node j가 사이클 노드 리스트 C에도
    // 없고 융합노드 G[n]의 edge list에도
    // 없으면 G[n]의 edge list에 삽입
    if (is_in_C(j)='N' && is_in_G[n](j)='N')
      insert j into list of G[n];

    // G[i]의 다음 노드로 진행
    ptr1 = ptr1->next;
  }

  // 사이클 리스트의 다음 노드로 진행
  ptr = ptr->next;
}

// 방향그래프 G의 헤드 노드에 각각에
// 대해서 다음의 처리를 한다.
for (i=0; i<n; i++) {
  // 삭제 처리된 노드는 skip
  if (G[i].deleted = 'Y') continue;

  // G[i]의 edge list에 있는 노드의
```

```
  // 융합노드의 node id인 n으로 변경
  ptr = G[i].first;
  while (ptr != NULL) {
    j = ptr->node_id;
    if (is_in_C(j)='Y')
      ptr->node_id = n;

    ptr = ptr->next;
  }
}

// 방향그래프 G의 헤드 노드에 대해서
// p_count 값을 재계산
for (i=0; i<n; i++)
  G[i].p_count=0;
for (i=0; i<n; i++) {
  if (G[i].deleted = 'Y') continue;

  ptr = G[i].first;
  while (ptr != NULL) {
    j = find_index_if_G(ptr->node_id);
    G[j].p_count++;
    ptr=ptr->next;
  }
}

// p_count=0인 node를 스택에 삽입
for (i=0; i<node_count; i++) {
  if (G[i].deleted = 'Y') continue;
  if (G[i].p_count=0)
    push it into stack;
}
```

4. 적용 사례

금오공과대학교의 학사관리시스템은 데이터베이스 테이블 970개를 포함하고, 소스 코드가 450,000 라인에 달하는 대형 소프트웨어이다. 본 논문에서 제안한 설계순서 추출 알고리즘을 실제 상황에 적용하기 위해서 수강신청 처리 업무를 기술한 자료흐름도(<그림 4>)를 발췌하였으며, 이를 입력하여 추출한 테이블 설계순서는 다음의 <표 2>와 같다.



<그림 4> 수강신청 처리 업무 자료흐름도

<표 2> 테이블 설계순서

순번	테이블 명칭
1	학과학적사항
2	영구성적
3	선수과목
4	순수과목 오류검사
5	강의교원
6	개설강좌
7	수강학과
8	학적사항
9	학사운영기간
10	수업시간강의실
11	교과과정
12	사용자
13	login
14	개설강좌display
15	수강인원초과검사, 주야교류오류검사, 수강학과제한검사, 수강신청, 학점초과검사, 재수강처리
16	재수강전 과목
17	수강신청내역

5. 결 론

본 논문에서는 리가시 정보시스템의 유지 보수 단계에서 필요한 테이블 설계순서 추출기를 설계, 구현하였다. 제안한 추출기는 사용자 인터페이스, 자료흐름도를 입력하는 '자료흐름도 처리기', 테이블 설계순서를 추출하는 '설계순서 추출기', 그리고 보고서 작성기능을 담당하는 '보고서 작성기'로 구성하였다. 사용자 인터페이스는 1개의 화면만을 사용하였으며, 화면의 전반적인 구성은 가장 보편적인 형태를 유지하였다. 각각의 모듈은 'My Builder' 및 C언어로 구현하였다. 향후 연구를 통하여 자료흐름도를 기술한 파일을 일괄 로더하여 저장할 수 있는 처리기를 구현한다면 방대한 자료 입력 량을 획기적으로 감소시킬 수 있게 되어 추출기의 실용성은 더욱 커질 것으로 판단된다.

참 고 문 헌

[1] T. DeMarco, Structured Analysis and Specification, Yourdon Inc., New York, 1978.

[2] Feng-Yang Kuo, "A Methodology for Deriving an Entity-Relationship Model Based on a Data Flow Diagram," J. Systems Software Vol. 24, pp. 139-154, 1994.

[3] Minyi Guo, "Automatic Transformation from Data Flow Diagram to Structure Chart," Software Engineering Notes, Vol. 22, No. 4, pp. 44-49, 1997.

[4] Terrence P. Fries, "A Framework for Transforming Structured Analysis and Design Artifacta to UML," SIGDOC '06, October, pp. 105-112, 2006.

[5] 임은기, "자료흐름도로부터 데이터베이스 테이블 설계순서를 추출하는 알고리즘," 한국정보기술학회 논문지 제5권 제4호, pp 226-233, 2007.

[6] A. V. Aho, John E. Hopcroft, and Jeffrey D. Ullman, Data Structures and Algorithms, Addison-Wesley, p221, 1983.

부록 #1 <설계순서 추출 알고리즘>

```
01 construct digraph G from DFD data;
02 for(i=0; i<n; i++) {// Let n be the # of
                        nodes in G
03     if(every node has a predecessor) {
04         detect a cycle starting from a node
             with minimum in-degree;
05         replace the cycle by a unified node;
06         count # of predecessors
             for a unified node;
07         continue;
08     }
09     pick a node v that has no predecessors;
10     output v according to its type;
11     delete v and its outgoing edges from G;
12 }
```

※ 위에서 밑줄 친 부분은 하위 모듈을 호출하는 것을 의미한다.

임 은 기 (Eun-ki Lim)

- 1977년 서울대학교 수학과(학사)
- 1988년 한국과학기술원 전산학과(석사)
- 1993년 한국과학기술원 전산학과 박사과정 수료
- 1989년~현재 금오공과대학교 컴퓨터소프트웨어공학과 교수
- 관심분야 : 데이터베이스 설계, S/W시스템 분석 및 설계, 컴퓨터 교육

논 문 접 수 일 : 2012년 02월 01일
1차수정완료일 : 2012년 02월 26일
2차수정완료일 : 2012년 04월 18일
게 재 확 정 일 : 2012년 04월 23일