데이터베이스 리버스를 통한 소프트웨어 유지보수의 사례연구

나학연[O], 최용락, 류성열
숭실대학교 컴퓨터학부

hacyunna@selab.soongsil.ac.kr, ylchoi@selab. soongsil.ac.kr,
syrhew@computing.soongsil.ac.kr

A Case Study of Software Maintenance by Database Reverse

Hac Yun Na[O], Yong Lak Choi, Sung Rul Rhew
Dept. of Computing, SoongSil University

요 약

시스템에 사용되는 데이터베이스 또한 소프트웨어이고 이를 유지보수 하기위해 데이터베이스 역 공학을 사용하고자 한다. 데이터베이스 역 공학 기법은 기존 시스템에서 사용되는 데이터베이스의 스크립트 파일들을 기준으로 현 데이터베이스의 구조를 역으로 도출하는 기법이다. 본 논문은 소프트웨어 유지보수를 위해 위와 같이 도출된 구조를 대상으로 검증 과정을 실시하여 문제점을 파악하고 해결 방법을 제시한다.

1. 서 론

대부분의 시스템에는 대규모의 데이터베이스가 구축되어 있다. 구축되어진 데이터베이스에 문제가 있을 때 이러한 문제점들을 해결하기위한 공정을 검토하고, 효율적인 절차를 확립할 필요가 있다. 기존에 구축된 데이터베이스의 역 도출 기법이 필요하겠고, 도출된 자료를 분석하여 문제점을 발견하는 과정도 필요하다. 최종적으로 문제점이 해결된 데이터베이스 모델을 그리는 과정이 있다. 이러한 과정을 도식화한 공정으로 표현함으로써 유지보수 환경을 보다 효율적으로 관리할 수 있게 한다. 관련 연구로 역 공학과 ERD 분석을 정리하여 본 논문에 적용한다.

끝으로 사례연구에서 본 공정을 통한 데이터베이스 유지보수를 수행하고, 향후의 연구 과제를 살펴 본다.

2. 관련 연구

2.1 ERD((Entity Relationship Diagram)분석

ERD 는 데이터베이스의 모델이다. 이것이 필요한 이유는 기업의 정보구조를 실체와 관계를 중심으로 명확하게 체계적으로 표현하고 문서화하기 위함이다.

여기에는 Logical ERD 와 Physical ERD 가 있다.

각각의 ERD 에서 데이터베이스 모델을 분석하는 기법을 정리하면 다음과 같다.

- Logical ERD 분석

논리적 구조의 파악은 실체/관계/속성의 정합성을 먼저 파악하고 Logical ERD 에 적용한 일반적인 규칙을 파악한다. 업무 흐름을 고려한 실체와 관계와의 의미 있는 연결 등을 검토하고 각 속성이 적절히 표현되어 있는 가와 Data 의 중복, 정규화 등을 검토한다.

- Physical ERD 분석

물리적 구조의 파악은 실제 Database 생성 상태와 현 운영 상태(Data 의 분산/분할,집중적 부하) 등을 고려하여 파악하는 것이 원칙이다. 이러한 과정 상의 준비가 소홀할 경우 즉 Document 만을 근거로 분석할 때 Table/FK/Column/Index 등만을 고려하여 검토 할 수 있다.

3. 데이터베이스 리버스

본 절에서는 데이터베이스 리버스 분석 단계를 설명한다. 그리고 사례연구를 통해 이 분석 모델을 적용해 나간다. 여기서 얻어진 결과를 좀더 체계적으로 관리하기 위해 문제점들을 목록화 한다. 그리고 이러한 문제점들에 대한 해결 방법과 유를 문서화 한다. 다음 그림 1 은 데이터베이스 리버스 분 단계를 도식화 한 것이다.

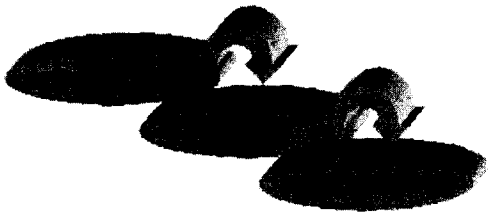

그림 1. 데이터베이스 리버스 분석 단계

그림 1 의 데이터베이스 분석 단계를 각 단계별 기능으로 세분하여 설명하면 다음과 같다.

단계 1. Script File 을 통한 DB 구조 도출

SQL(Structured Query Language)로 작성된 Database Script File 을 리버스 자동화 도구인 ERWin 을 사용하여, ERD 를 생성한다. 그림 2 는 문제점을 갖고 있는 데이터베이스의 Script File 이다.

그림 2. Database Script File

```
drop table cmUSERID;

create table cmUSERID (
    UserId          char(6)  not null,
    Password        char(6)  not null,
    UserName        char(20) not null,
    UserSabun       char(20) default null,
    UserSosok       char(20) default null,
    UserJikwhi      char(20) default null,
    UserPhone       char(20) default null,
    Permission      decimal(8,0)     not null,
    MalsoGb         char(1)  default '0',
Updatelija      decimal(8,0)     default 0,
    UpdateSigan     decimal(8,0)     default 0,
    SystemMgr       char(1)  default 'N',
    StockTradeMgr   char(1)  default 'N',
. . .
. . .
    HostSendMgr     char(1)  default 'N',
    primary key(UserId,Password)
);
..
..
..
revoke all on cmUSERID from public;
 grant select on cmUSERID to public;
. . .
. . .
```

단계 2. 문제점 파악

Logical ERD 분석과 Physical ERD 분석을 통해 문제점을 파악한다. 문제점으로 지적할 수 있는 사항들을 목록화 하고, 그 이유와 해결 방안을 정리 한다. 이러한 문서화 작업은 차후 관리에 도움을 준다. 이 Script File 을 ERWin 에서 ERD Reverse 를
위한 입력으로 사용하여 나온 산출물은 그림 3 과 같다.

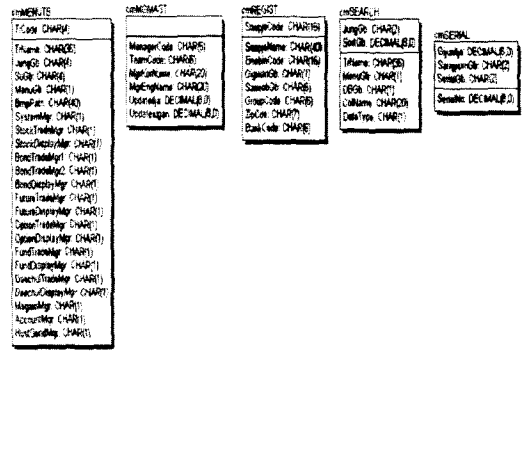

그림 3. ERWin 을 이용한 ERD

현 ERD 를 분한 결과는 표 1 에서 보는 바와 같이 표준화 설정의 미비, 중복된 Table, Table 크기의 비대, 관계 설정의 미비, Key 설정이 미흡, Domain Integrity 문제로 구성되어 있다.

표 1. ERD 분석 표 구성

표준화 설정의 미비
중복된 Table
Table 크기의 비대
관계(Relation)설정의 미비
Key 설정이 미흡
Domain Integrity 문제

표 1 에서 제시된 문제점들을 좀 더 자세히 설명하면 다음과 같다.

- **표준화 설정의 미비** : 데이터베이스 설계 기법인 ERD 의 목적 중 제일 중요한 것은 효율적인 정보 전달인데, 현재의 Naming 규칙은 정보 전달 차원으로 이해하기에 부적절하다. 이는 개발의 생산성 차원에서도 신중히 고려해야 할 사항이다.[4]

- **중복된 Table** : 현재의 ERD 는 같은 종류 및 성격의 Table 이 여러 개 존재하는 것으로 나타나 있다. 이는 성능(Performance)을 위한 DeNormalized 차원에서는 고려해 볼 수 있었으나, 실제 Database 에서 Data 의 불일치 제거, Data 무결성 확보 차원 등에 지대한 영향을 미치므로 이는 반드시 시정되어야 한다.[4]

- **Table 크기의 비대** : 전반적으로 Table 의 크기가 너무 크게 설계되었다.. 이는 Performance 에 지대한 영향을 미치므로 반드시 고려해야 할 사항이다.[5]

- **관계(Relation)설정의 미비** : 전체 Database 에서 Entity 간의 관계가 전혀 없다. 이는 관계형 DBMS(Data Base Management System)의 가장 큰 장점인 관계가 없는 것으로 관계형 DBMS 를 File 구조의 시스템으로 사용하는 것과 같은 효과를 나타낸다. 따라서 Data 의 무결성 유지에도 문제가 있으며 Data 를 Join 할 때 성능에도 문제점이 많이 발생한다. 관계형 DBMS 의 기본 3 요소는 Entity /Relationship/ Attribute 이며 이들 구조의 정합성을 고려하여 잘 설계하는 것이 매우 중요하다.[5]

- **Key 설정이 미흡** : 일반적인 PK(Primary Key)상태는 대체로 크지 않고 잘 설정되어 있으나, 일부 Table 의 PK 는 너무 크게 설계되었다. 예를 들어 cmUSERiD 는 패스워드를 PK 로 설정하였다. 이는 PK 의 길이에도 문제가 있지만, 사용자의 패스워드 변경 시에는 PK 가 수정되는 상태이므로 시스템에 많은 영향을 미치게 되고, 시스템 무결성 유지에도 나쁜 영향을 미친다. 또한 일반적으로 업무의 유형이나 Join 의 유형을 분석한 AK(Alternate Key) 또는 FK(Foreign Key)등이 거의 없는 것으로 설계되어 있다.[6]

- **Domain Integrity 문제** : 제약 조건의 하나인 영역 무결성 규칙 유지를 위한 설계가 없다. 일반 프로그램과 시스템 프로그램은 성능에 많은 차이가 있으므로 관계형 DBMS 가 제공하는 Utility 기능을 활용하는 것이 유리하다.[6]

단계 3. 문제점을 제거한 ERD 도출

발견된 문제점을 해결하기위한 방법을 제시한다. 해결 방법 또한 문서화 한다. 최종적으로 문제점을 제거하면서 새로운 ERD 를 도출한다. 문제점을 해결하는 방안은 다음과 같다.

- **표준화 설정의 미비**
  Table name 에 대한 표준을 만드는 것이 중요한데, 한글의 영어식 표현은 되도록 피한다.

- **중복된 Table**
  같은 종류 및 성격의 Table 은 서로 병합하여 중복을 없게 만든다.

- **Table 크기의 비대**
  활용도를 분석해서 Table 의 크기를 줄이거나, Table 을 좀 더 작게 분할 하여 Database buffer block 의 이용도를 높여야 한다.

- **관계(Relation)설정의 미비**
  실체와 관계를 설계할 때에는 Data 의 발생 순서와 주 실체와 종 실체를 정확하게 설계해야 한다.

- **Key 설정이 미흡**
  PK, AK, FK 의 적절한 사용을 해야 하는데, 전반적으로 Key 에 대한 재조정을 수행해야 한다.

- **Domain Integrity 문제**
  영역 무결성은 stored procedure 나 Trigger 를 이용하여 구현을 해야 한다.

그림 4 는 문제점을 제거하고 새로 도출한 ERD 이다.

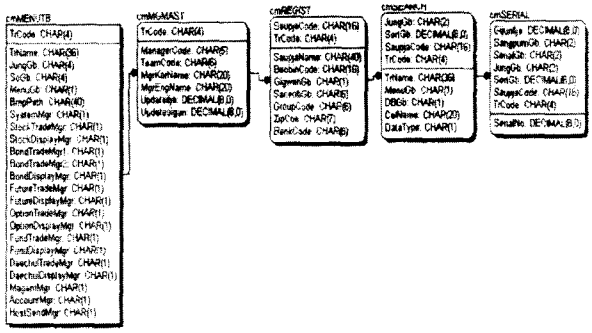

그림4. 문제점을 제거한 ERD

4. 결론 및 향후 연구 과제

본 논문에서 제시한 방법을 이용하여, 데이터베이스 리버스 과정을 사례연구로 소개하였다. 이번 Database 검증에서는 확인 할 수 없었던 Table 의 분산/분할을 해야 한다. 그리고 Disk 의 활용 등을 고려한 Disk Volume 의 재조정 및 재배치를 해야 할 것이다. 특히, 데이터베이스의 설계에 대한 관심이 높아지고 있기 때문에 더 많은 문제점 파악에 대한 연구가 필요 할 것이다. 축적된 자료들 또한 데이터베이스화 하고 이를 문제 파악 단계에 활용함으로써 데이터베이스 리버스 분석 단계를 프로세스로 확립하는 연구를 계속해 나갈 것 이다.

5. 참 고 문 헌

[1] Scott Tilley, "A Reverse-Engineering Environment Framework, Carnegie Mellon Software Engineering Institute," April 1998.

[2] Rene R.Klösch, "Reverse Engineering: Why and How to Reverse Engineer Software, Proceedings of the California Software Symposium," 1996.

[3] Jean-MarcPetit, et al, "Towards the Reverse Engineering of Denormalized Relational Databases," ICDE '96

[4] David W. Cheung, et al, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Techniques," ICDE '96

[5] Chengwen Liu, Hao Chen, and Warren Krueger, "A Distributed Query Processing Strategy Using Placement Dependency," ICDE'96