데이터베이스 논리적 모델링 사례 분석

논리적 모델링은 프로그램 코드 작성이나 SQL 작성처럼 일상적인 업무를 통해 실력이 쌓이는 기술은 아니다.
데이터베이스 설계가 거의 프로젝트 초반에 집중적으로 이루어지고, 대부분의 초급 개발자들은 프로젝트 리더가 작성한 테이블과 사용자의 요구를 대조하면서 쿼리를 작성하는 업무에 집중하기 때문이다. 그런데, 논리적 모델링 작업을 이해하거나 기술이 없는 상태에서 고급 엔지니어로 승격될 수 있는가라고 물어보면 다들 그렇지 않다고 대답할 것이다.
프로젝트를 책임지는 위치라면 분명히 고객의 요구를 수용해 실무 개발자들이 작업할 수 있도록 데이터베이스를 설계하는 능력이 필요하기 때문이다.

논리적 모델링의 중요성

현실적으로 실무를 담당하는 개발자들이 데이터베이스 모델링을 학습하는 과정은 주먹구구식이라 해도 무방하다. 프로젝트 관리자가 만들어낸 테이블들을 어깨너머로 보고, 회의록을 뒤져가며 이전 프로젝트에서 만들어진 ER 다이어그램을 역으로 분석하며 학습하게 된다. 아무도 어떤 순서로 테이블을 추출하는지, 관계는 어떻게 만드는지 가르쳐 주지 않는다. 프로젝트를 수행하는 기간에는 데이터베이스 설계를 분석할 시간도 없다.
왜냐하면, 잘못된 테이블 설계인 경우 사용자가 요구하는 자료를 쉽게 추출할 수 없어 복잡한 쿼리를 만드는데 대부분의 시간을 허비하게 되고 개발 기간이 초기 예상보다 몇 배로 늘어나게 마련이다. 테이블 설계를 수정하면 되지 않느냐고 반문할지 모르지만, 개발이 상당부분 진행 중인 상태에서 테이블 설계를 바꾼다면 전체 시스템을 재구축해야 하는 경우가 발생할 수도 있다. 심지어 테이블 설계 과정을 제대로 수행하지 못해, 어떻게 고쳐야할지조차 몰라 테이블 재설계가 불가능한 경우도 빈번하다. 이렇다 보니 새로운 프로젝트에서 똑같은 시행착오를 반복하게 마련이다.
과거의 실패를 되새기고, 수정할 시간이 없기 때문인데, 개발 기간이 부족할 경우 가장 먼저 단축되는 것이 바로 설계 기간이기 때문이다 개발자들은 화면으로

보이는 인터페이스를 먼저 개발하려고 한다. 그런 작업이 전체 업무의 대부분을 차지한다고 생각한다. 하지만, 사용자 인터페이스는 만들기도 쉽고 고치기도 쉽다. 반면 데이터베이스는 만들기는 쉽지만 고치기가 어렵다.
데이터베이스 설계가 잘못되는 경우에는 아예 원하는 기능을 만들지 못하게 됨으로써 프로젝트 자체가 실패하거나, 90%의 업무를 끝마치고도 10%의 기능을 보완하지 못해 연장될 수도 있다.
최악의 사태는 서비스를 오픈한 후 느린 성능으로 인해 도저히 정상적으로 운영하지 못하거나, 재개발에 들어가는 상황이다.
대다수의 개발자들은 시스템 과부하의 원인을 하드웨어의 용량 부족 때문이라고 변명하지만, 제 아무리 좋은 자동차라고 할지라도 1 단 기어만 넣고 엑셀을 밟으면 최대 속도를 낼 수 있겠는가?
데이터베이스 성능 개선 작업에서도 최우선으로 진행하고, 가장 중요하게 여기는 것이 바로 쿼리와 테이블 설계다.
테이블 설계를 잘하고 못하느냐에 따라서 시스템 성능은 무려 수십 배까지 좌우될 수 있다. 무언가 보이기 위해서는 그것이 기록돼 있어야 한다. 데이터베이스에 기록하는 것은 단순한 2 차원 테이블이기 때문에 어렵지 않다는 편견을 가지기 쉽다. 하지만, 사용자가 언제나 단순한 결과만을 보여주길 원하겠는가?
설문 조사를 예로 들어보자. 설문 항목을 보여주고, 사용자의 선택을 입력받는 것은 어렵지 않다. 이런 작업을 의뢰 받으면 개발자들은 입력화면을 만들기만 하면 70%의 일을 마치는 것으로 생각한다. 하지만, 설문을 실시하는 목적이 무언가? 그 결과를 활용하기 위한 것이다.
사용자가 얼마나 상세한 결과를 원하는지 상상해 보아야 한다. 또한 계속적으로 사용자에게 물어보아야 한다. 사용자는 입력 화면을 작성하는 것을 시작으로 생각한다. 그리고 입력된 자료를 이용해 다양한 차트와 보고서를 만들길 원한다.
다양한 차트와 보고서를 쉽게 만들 수 있는 설계를 할 것인가, 아니면 처음부터 다시 개발해야 하는 데이터베이스를 만들 것인가? 그에 대한 해법이 바로 논리적 모델링을 제대로 수행하는 것이다. 그럴 시간이 없다고 변명해서는 안된다. 그런 말은 고급 개발자가 되고 싶지 않다는 말과 동의어인 것이다. 또한 좋은 설계를 함에 따라서 높은 성능과 개발기간 단축이라는 두 마리 토끼를 잡을 수 있게 된다.

논리적 모델링

논리적 모델링이란 무엇인지 다시 한 번 살펴보자.

논리적 모델링 순서

데이터 모델링 기법에는 하향식 분석(Top-down)과 상향식 분석(Botton-up) 등 두 가지 방법이 있다. 하향식 분석 기법은 구축하려는 시스템의 개략적인 업무흐름을 이해한 후에 관리 대상(entity)을 추출하는 것이며, 상향식 분석은 사용자 인터페이스와 이미 만들어진 데이터베이스를 기초해서 모델을 구성하는 것이라고 했다. 우선 하향식 분석 과정을 실행한 후, 상향식 분석으로 보완하는 절차를 설명하도록 하겠다.
하향식 분석의 순서는 다음과 같다.

① 업무 흐름 파악
② 개체(entity) 추출
③ 관계(relation) 설정
④ 업무 흐름을 분석 후 이벤트 개체/관계 추출
⑤ 요약 개체 추출

업무 흐름(Workflow) 파악

데이터베이스 모델링에 앞서 시스템 분석 단계에서 사용자의 요구사항을 문서로 정리하게 되며, 이것이 논리적 모델링을 위한 기초자료가 된다. 좀 더 상세한 스토리 보드(story board)를 작성한다면 더욱 정확한 설계가 가능하다.
논리적 모델링을 위한 사례는 '대학교 홈페이지 구축'을 제시할 것이다. 이런 유형의 업무를 제시하는 이유는 근래의 데이터베이스 개발 업무는 대다수 웹 기반이며, 그중에서 가장 흔하게 접하게 되는 것이 '게시판' 혹은 그와 유사한 업무이기 때문이다.
데이터베이스 모델링 세미나에서 어떤 강사는 '게시판'만 구현할 줄 알면 포털 사이트를 만들 수 있다는 언급을 한 바 있고, 무척 공감하는 이야기이다. 그렇다고, 게시판이 웹 사이트의 전부는 아니기 때문에, 소규모 시스템에서 주로 요구되는 몇 가지 업무들을 설계하는 과정 또한 소개할 것이다. 대학교에서

홈페이지를 개편해달라는 요청을 받고 개발에 착수한 경우, 대학 측의 요구사항 중 중요한 점들을 정리해 보겠다.

① 수요자들의 포털 접근을 유인하기 위한 각종 이벤트를 처리할 수 있는 기능 : 학생관심, 시사문제, 설문 코너 및 온라인 투표 기능
② 웹 페이지 상에서의 업무처리 및 서비스 지원 : 학생이 문의한 내용의 처리 상황을 온라인의 개인 영역에서 확인할 수 있도록 한다(Q&A 상황을 진행 중, 완료 등으로 표시하여 처리 상태에 대한 피드백을 줌).
③ 커뮤니티 강화 : 원활한 쌍방향 커뮤니케이션을 위해 Q&A, 학과 및 동아리 커뮤니티 기능

위와 같은 간단한 요구사항들을 기초해서 모델링을 진행해야 한다는 것이 부담스럽게 여겨질지도 모르지만, 차근차근 접근해 나가는 과정을 통해 이해할 수 있을 것이다.
대부분의 프로젝트는 위에 나열한 수준의 요구사항에 근거해서 설계를 수행하게 된다. 상세한 흐름을 분석하는 과정은 분석가의 경험에 의존하기도 하고, 사용자 측과 요구분석 면담을 추가로 수행하게 된다.

개체(entity) 추출

하향식 논리 모델링에서 가장 먼저 수행하는 작업은 데이터베이스에 저장할 대상을 추출하는 작업이다. - 대상이라는 단어는 영어로 object 인데, object 라는 단어에는 다른 것과 구별되는 독립된 어떤 것이라는 의미가 포함되어 있어서, entity 라는 단어와 혼용된다. 일단, 데이터베이스에서는 entity 라는 단어를 주로 사용한다.
개체를 추출하는 가장 손쉬운 방법은 요구 사항에서 명사들을 추출하는 것이다. 개체라는 시스템의 관리 대상으로서 사람, 물건 등의 실체나 개념을 의미한다고 하였다. 그러나 모든 명사가 저장해야할 개체가 되는 것은 아니라는 점을 명심해야 한다.
홈페이지라는 업무에서 개체들을 모두 나열하다보면 많은 개체들이 복잡하게 얽혀서 전체를 들여다보기 어려워질 수도 있다. 따라서 큰 업무를 다시 쪼개어 분석하는 과정을 거친다.

큰 업무를 작은 업무로 다시 분할한 것을 영역 혹은 도메인(domain)이라고 한다. 위에 예시한 학교 홈페이지 서비스의 경우에는 이벤트, Q&A, 커뮤니티 등 3 개의 영역으로 분할하여 설계하도록 하겠다.

● **이벤트**

학생관심, 시사문제라는 표현은 홈페이지 기능 혹은 메뉴를 구상하기 위한 제시어(혹은 방향 설정)라고 볼 수 있으며, 개체라고 보기는 어렵다.

설문 코너와 온라인 투표는 기능에 대한 요구이면서 학생들이 입력한 내용을 저장해야 하므로 개체라고 판단 해야 하며, 설문과 투표라는 2 개의 개체를 후보로 선정하도록 한다.

그리고 설문과 투표는 개체인가 아니면 다시 분리될 수 있는 가를 생각해 봐야 한다.

우선 투표의 경우, 학생들의 선택을 기록하기 위한 개체가 필요할 것이고, 더불어 여러 가지 문제에 대한 투표 결과를 기록해야할 필요성이 있을 것이다.

그렇다면 투표 주제를 관리할 수 있는 개체를 분리하는 것은 어떤지 생각해 보아야 한다.

투표 주제와 결과를 하나의 2 차원 표에 표시할 수 있는가를 떠올려 보자. 여러 개의 투표 결과에 대해서 하나의 주제나 표시되는데, 두개의 정보를 하나의 표에 나타내는 것보다 나누어서 표시하는 것이 자연스럽다.

따라서 투표 기능에서는'투표 주제'와'투표 결과' 라는 2 개의 개체를 추출할 수 있다. 고객의 요구사항에서는 투표 주제가 여러 가지로 존재할 수 있다는 언급은 없었다.

하지만, 그런 것까지 예측할 수 있어야만 개발 단계에서 설계를 변경하는 심각한 위험(risk)을 피할수 있게 되는 것이다.

더불어 이런 설계 단계에서 밝혀낸 요소가 정말 필요한 것인지를 확인하기 위해 고객에게 설계 결과를 설명하는 과정을 거치는 것이 좋다.

설문의 경우에도'설문 주제'와'설문 결과'라는 개체를 도출 할 수 있지만, 설문이 투표와 다른 점은 질문 항목이 복수라는 점 이다. 따라서'설문 문항'을 기록할 수 있는 개체가 필요할 것이다.

주제와 결과, 그리고 문항을 분리하는 이유는 앞서 설명한 바와 같은데 하나의 주제에 대해서 여러 개의 문항이 존재하며, 마찬가지로 각 문항에 대해 다수의 답변이 존재한다.

이런 경우에는 전체를 하나의 2 차원 테이블에 표현할 수 없기 때문에 각각을 분리하는 것이 좋다. 즉, 개체라고 여겨지는 것을 2 차원 테이블에 집어넣을 수 없다면 분리하는 것을 고려해야 한다.
이와 같이 데이터베이스의 논리적 개체를 추출할 때는 개체가 분리될 수 있는 것인지를 들여다보아야 한다.
가급적 잘게 쪼개어 더 이상 나눌 수 없는 단계까지 쪼개보는 것이 좋다.
다시 결합할 필요가 있을지 몰라도 우선은 분리한 후에 재조합하는 편이 실패를 적게 하는 요령이다.
그렇다면 더 이상 나눌 수 없는 단계라는 것은 어떻게 판단할 것인가?
대상 혹은 개체를 분리하려고 시도했을 때, 값(수치, 문자열)만을 가지게 된다면 그것은 더 이상 해체할 수 없는 것이다.

● 커뮤니티

커뮤니티라는 단어 자체는 개체가 될 수 없다.
즉흥적으로 생각해 보아도 거대한 자료의 집합체라는 것을 이해할 수 있기 때문이다. 따라서 커뮤니티가 가지는 기능들을 세밀하게 묘사하고 기능을 분할한 후에 그 안에서 개체를 찾는 것이 유리할 것이다.
커뮤니티는 일반적으로 카페나 동호회의 집합을 의미하기 하기 때문에 상식적인 개념을 적용하기로 하고, 동호회는 다시 여러 개의 게시판으로 이루어진다는 것을 알 수 있다. 그런데, 커뮤니티 전체를 모델링하는 것은 소규모 모델링이라는 주제에서 크게 벗어나게 되므로 하나의 게시판을 어떻게 모델링하는지에 대해서만 언급하기로 한다.
게시판이라는 것은 사용자(학생, 직원, 외부인)들이 작성한 글들의 집합이다.
우선 하나의 글을 하나의 개체로 볼 수 있는가를 따져보도록 하자.
그렇다면 사용자가 글을 쓸 때마다 새롭게 개체 혹은 테이블이 생겨야 할 것이다.
일단 게시물들의 집합 즉 게시판은 하나의 개체로 볼 수 있다는 것을 확인할 수 있다. 또, 게시판을 분할하면 제목, 작성자, 일자, 내용 등 단일 데이터로 분리되므로 앞선 판단의 보충 근거가 될 수 있다.
그런데 게시물의 하위 요소로서 추가할 수 있는 것들이 있다. 첨부 파일(attach), 답글(reply), 덧글(comment)과 같은 것들이다. 우선 답글을 생각해보자. 답글은 일반 게시물과 다른 점은 원문 게시물의 아래에 표시된다는 것이며 그 외에는

동일한 항목들을 포함하고 있다. 따라서 답글은 게시판의 일부로 봐야 할 것이며, 별개의 개체로 표현해서는 안 된다.
첨부 파일과 덧글은 하나의 게시물에 복수가 연결될 수 있다.
이렇듯 어떤 두 개의 대상 혹은 개체가 일대다의 관계(1:N)를 가지면서, 종속적인 경우에는 명백히 별개의 개체로 표현해야 한다.
일대일 대응이면서 종속적인 경우에는 굳이 개체로 분리하지 않아도 된다. 즉, 게시물에 파일을 하나만 첨부할 수 있다면 첨부파일과 게시판은 하나로 표현해도 된다.
이번 설계에서는 첨부파일 수를 복수로 가정하기로 한다.
정리하자면 게시판은 '게시문서', '첨부파일', '덧글' 등 3 개의 개체로 분리될 수 있다.
이와 같은 논리적 설계에서는 고려하지 않은 것이 있는데 다수의 게시판을 어떻게 표현할 것인가 하는 점이다.
우리는 하나의 게시판이라는 개념에서 하향식으로 모델링하였기 때문에 다수의 게시판을 표현할 방법이 없다는 문제에 직면하게 되었다.
하향식 설계에서는 복잡한 시스템을 손쉽게 모델링하기 위해 특정 영역에 한정하여 모델링하는 경우가 있고, 그런 관계로 상위 개념의 개체를 누락하는 문제가 발생할 우려가 있다.
때문에 어느 정도 설계를 진행한 후에 전체를 다시 내려다보는 과정이 꼭 필요하다.
복수의 게시판을 관리하기 위해서는 게시판 개체를 관리하는 상위 개념의 '게시판 목록(유형)'이라는 개체를 모델에 추가하기만 하면 된다.
또한, 게시판들의 집합에 대한 상위 집합인 카페를 모델링하고 싶다면, 게시판 목록의 상위 개념인 '카페'를 다시 추가하면 된다.
즉, 설계 단계에서 누락된 개체를 추가하거나 모델을 확장하기는 비교적 쉽다는 것이다.
그러나 점차 개발에 가까워지거나 진행하는 단계에서 개체를 추가하는 것은 어려워진다.
상세한 항목과 수많은 개체들이 세밀하게 나열된 상태에서 틀을 고치는 것은 인간의 사고 능력에 많은 부담을 주고, 고쳐야할 문서와 소스가 점점 늘어나기 때문이다.

● **질의응답(Q&A)**

질의응답이라는 기능을 떠올려 보면 외형적으로는 게시판과 다를 것이 없다. 질의 내용을 작성하고, 목록을 출력하고, 질의 내용과 답변을 보여주면 되는 것이니 '답변형 게시판'이라고 정의해도 그다지 틀린 것은 아니다.

그러나 대학의 요구사항을 읽어보면 학생들이 질의한 내용이 처리되고 있는 상황(과정)을 조회할 수 있어야 한다고 명시되어 있다.

즉, 이것은 게시판의 기본 기능 이외에 추가적인 기능을 요구하는 것이다.

따라서 업무 흐름을 좀 더 상세히 묘사한 후 추가로 기록해야 할 사항들을 찾아낼 필요가 있다.

업무흐름(workflow)을 분석하면서 개체를 추출하는 과정은 다음 단계에서 설명하도록 하겠다.

일단, 질의응답은 게시판과 유사한 업무이므로 게시판에서 추출한 개체와 동일한 '게시문서', '첨부파일', '덧글' 등 3 개의 개체를 기본적으로 포함하는 것으로 정리하겠다.

● 공통 개체

정리된 요구 사항에 기초해서 설계를 진행해도 누락되는 부분이 발생한다.

그 이유는 설계자의 실수에서 비롯되는 것도 있고 사용자가 너무나 당연하게 생각해 언급하지 않은 결과 벌어지는 문제도 있다.

앞서 대학교에서 요구한 내역에서는 직원이나, 학생들이 홈페이지에 로그인해야 하는지 여부를 밝히고 있지 않다.

그렇다고 해서 누구나 글을 쓸 수 있게 한다면, 질의에 대한 응답을 직원만이 가능하도록 제한하는 등의 필수적인 조치를 취할 수 없게 된다.

사용자 측에서 너무나 당연하다고 여기기 때문에 언급하지 않는 것들은 주로 사용자(신상정보), 조직(부서), 권한, 코드 체계 등이 있다. .

사용자와 부서는 각기 다른 개체라는 것을 쉽게 이해할 수 있으니 각 개체를 모델링에 포함시키도록 하자.

그렇다면, 사용자의 접근 권한을 어떻게 다룰 것인가 생각해 보아야 한다.

즉, 개체로 볼 것인가 아닌가?

이런 의문을 제기하는 것은 개발 대상 업무의 특징 혹은 사용자의 요구수준에 따라서 개체로 정의할 수도 아닐 수도 있기 때문이다.

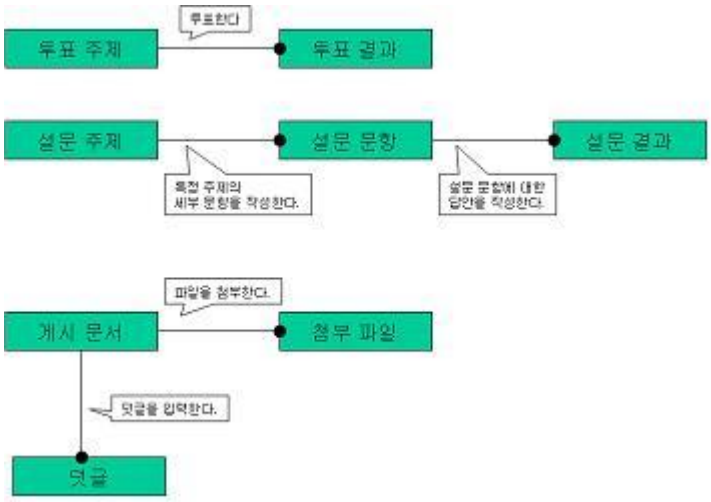
만약 사용자 개체 내에 재학생, 교직원 등의 구분 항목(또는 유형)을 포함하고, 글쓰기 권한을 유형에 따라 다르게 부여한다면 굳이 권한 개체를 추가할 필요가 없다.
설계라는 작업에 임하는데 있어서 어떤 업무에서는 어떤 설계가 나온다는 공식을 적용하려고 고집해서는 안 된다.

<그림 1>투표 결과, 주제 개체 추출

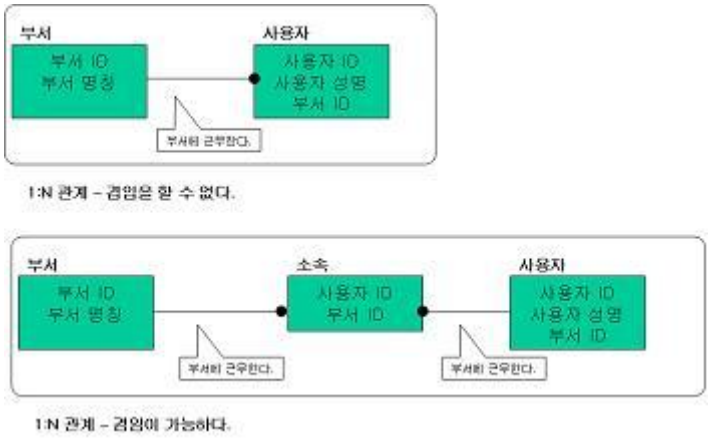
관계(relation) 추출

관계라는 것은 개체 간의 의존성 혹은 관련성을 표시하는 것이다.
요구사항으로부터 개체들이 추출하였다면 다음으로 개체간의 관계를 파악해야 한다.
관계의 유형의 일 대 일(1:1), 일 대 다(1:N), 다 대 다(N:N) 등이 있다.
이중에서 N:N 관계에서는 새로운 개체가 추가되기도 한다.
앞서 언급한 개체들 간의 관계는 거의 1:N 관계이다. 이런 관계는 매우 빈번하고 데이터를 검색 및 관리하는 과정에서도 큰 어려움이 없기에 간단히 그림으로 표현하도록 한다.
새로운 개체가 추가되는 상황을 설명하기 위해서 사용자와 조직(부서) 개체의 관계에 대해서 언급해 보도록 하겠다.
질의응답 기능에서 학생이 작성한 질의에 대해 직원이 답변을 쓰는 경우 작성자의 성명뿐만 아니라 소속 부서도 의미 있는 정보가 된다.
소속명(혹은 부서명)을 신상정보 개체에 문자열 속성으로 추가한다면 간단히 해결되는 문제일 수도 있다. 이것은 손쉬운 해결책이라는 장점이 있지만 부서명이 바뀌거나, 통폐합 및 신설되는 경우에 사용자의 소속명이 정확하게 유지될 것이라는 보장을 못하게 된다.
이런 문제를 보완하기 위해서는 부서와 사용자 간의 관계를 1:N 으로 표현할 수 있다. 즉, 하나의 부서에는 여러 명의 직원이 근무한다고 표현하며, 이로서

사용자의 소속 정보는 부서라는 개체에 의존되어 있음을 명시하면 된다. 그렇다고 해서 문제가 완벽히 해결되었다고 보기 어렵다. 드물게 한 직원이 둘 이상의 부서에 속해 있는 경우가 있다. 대학의 예를 들어 보자면, 컴퓨터공학과의 교수가 정보센터의 소장을 겸임하고 있는 경우가 있을 수 있다.
이런 경우는 사용자와 부서 간의 관계가 N:N 관계가 되어 버린다.
N:N 관계를 표현하기 위해서는 2 개의 개체를 연결하는 개체를 추가로 모델에 표시하면 된다.
N:N 관계에 대한 해법은 공식과 같은 것이기에 N:N 관계인 것으로 판단되면 주저 없이 개체를 하나 추가하면 된다 (<그림 3> 참조).



<그림 2>관계추출



<그림 3>N:N 관계 모델링

업무 흐름상의 이벤트 개체 및 관계 추출

앞서 개체 추출 단계에서 요구분석 내용을 읽고 다수의 개체를 확인하였지만, 질의응답 업무의 흐름을 파악하면 추가할 개체가 드러나게 된다.

질의응답 업무가 어떻게 진행되는지 대학 측과 논의한 시나리오를 정리해 보도록 하겠다.
학생들은 학교 홈페이지의 질의응답 코너를 통해 다양한 장학제도, 수강신청, 휴학 및 복학 절차 등의 질문을 올릴 수 있다.
질문들에 답변해야할 부서와 담당자가 다르기 때문에 질의응답 게시판 관리자는 각 질문에 대한 담당부서 및 담당자를 지정할 수 있어야 한다.
답변 작성자로 지정된 직원은 인계된 질문에 대한 답변을 정리해서 게시할 수 있으며, 만일 본인의 소관이 아니거나 타 부서의 추가 답변이 필요한 경우는 다시 해당 질의를 인계할 수 있다.
학생들은 자신이 질문에 대한 처리 일시와 담당자, 그리고 처리 결과를 개인화된 페이지에서 확인할 수 있어야 하고, 관리자는 모든 질의사항에 대한 처리 현황을 조회할 수 있어야 한다.
이렇듯 업무 흐름을 보면 기록해야 할 정보가 더 있음을 파악할 수 있다.
하나의 질의에 대해서 복수의 응답이 있을 수도 있으며, 각 응답을 누가, 언제 작성했는가라는 정보를 기록해야 한다.
답변이 완료되었는지 상태 여부도 필요하다면 기록해야 하는 것이다.
이러한 정보는 화면 인터페이스나 기능 요구 사항에 표현되지 않을 수도 있으므로, 가급적 스토리 보드(story board), 시나리오 혹은 플로우차트 등을 작성하면서 찾아내야 한다.
업무가 진행되는 특정 시점에 사건(event)으로 인해 발생하므로, '이벤트 개체'라고 표현한다.
질의 사항에 대한 담당자, 부서 지정과 응답 내용을 관리하기 위해 '답변 기록'이라고 하는 개체를 추가하며, 질의에 종속적이므로 '질의' 개체와의 관계를 '처리한다'라고 설정하며, 1:N 관계임을 표시한다.

속성 정의

개체와 관계가 정리되었다면 남은 작업은 개체의 속성을 정의하는 것이다. 앞서 도출한 개체들에 상세한 속성들을 추가하면 되는데, '질의처리 상태' 등 관리가 필요하나 개체로 표현하지 않은 것들을 빠짐없이 적절한 개체에 추가해야 한다. 논리적 모델링 단계에서는 속성의 데이터타입이나 크기 등은 생각하지 않는 것이 좋다. 또한, 속성을 추가하면서 해당 속성이 복수로 발생하지는 않는지 주의 깊게

살펴봐야 한다. 무심코 단 하나의 값만을 저장할 수 있게끔 설계한 후에 구현 단계에서 상당한 곤란을 겪게 되는 일들이 빈번하기 때문이다.

요약 엔티티(속성) 추가

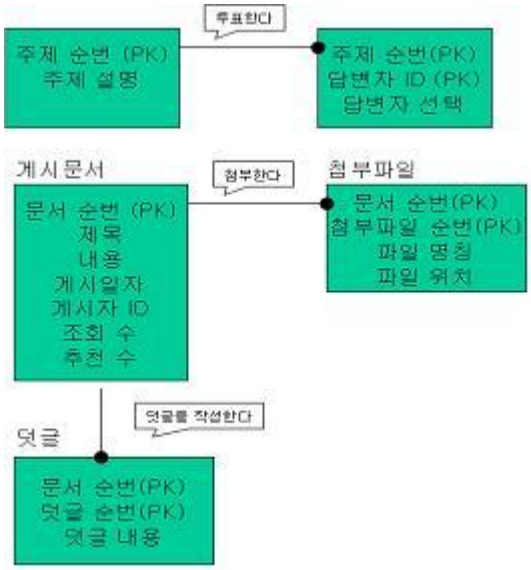
다시금 질의응답 업무에 대해서 검토해 보도록 하자. 학생들이 질의한 목록을 출력하는 화면에서 각 질의에 대한 '답변완료' 여부를 표시해야 한다는 요구사항이 있을 수 있다. 이런 요구를 수용하기 위해서는 앞서 설계한 개체들을 어떻게 조합(혹은 쿼리)해야 할까? '게시문서' 개체와 '답변내역' 개체는 1:N 관계이므로 두개의 개체를 조인(join)한다면 목록을 출력할 경우에, 동일한 게시물이 두 번 이상 나타나게 된다.

해결 방안은 두 가지가 있다. 첫 번째는 게시문서 목록을 쿼리한 후에 각각의 게시문서에 대한 답변 내역을 다시금 쿼리하는 것이다. 이런 방식은 성능을 떨어뜨릴 뿐 아니라, 개발 시간까지 늘어나게 된다. 두 번째는 게시문서 개체에 처리 상태를 나타내는 속성을 추가하거나 처리 상태를 기록하는 개체를 별도로 추가하는 방법이다. 당연히 두 번째 방식이 최선이라고 할 수 있다. 그러면 첫 번째 방식은 틀린 것일까? 필요한 정보를 모두 도출해냈으니 첫 번째 방식이 틀렸다고 볼 수만은 없다. 그러나 모든 프로젝트는 정해진 기한 내에 종료해야 하며 동시에 충분한 성능이 보장되어야 한다. 설계자가 신중히 검토하지 않음으로서 발생하는 문제들은 결국 시스템을 구현하는 개발자의 부담으로 고스란히 넘겨지게 된다.

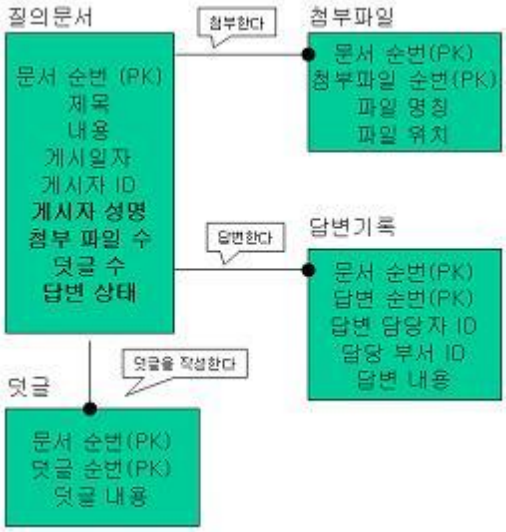
요약 속성이 필요한 것은 질의응답 게시판 뿐만이 아니다. 일반 커뮤니티 게시판에서도 덧글의 수를 목록에 함께 표현해야 한다면, 게시문서 개체에 덧글수를 포함시키는 것이 좋다. 요약 속성을 추가하는 작업은 엄밀히 말하자면 하향식 분석이라기보다는 상향식 분석 작업이라고 볼 수 있다. 즉, 실제 구현되는 화면이나 보고서 화면을 검토함으로써 알 수 있는 것들이다.

누누이 당부하지만, 설계에서 하나의 속성을 추가하는 것이 개발 단계에서 추가하는 것보다 몇 배의 시간을 절약하게 된다. 관념적으로는 개발 완료된 시스템이라고 할지라도 항목 하나 추가하는 것을 쉽게 생각할 수 있다. 하지만, 개발된 시스템을 수정하게 되면 소스, 설계 문서, 화면설계서 등 고쳐야 할 범위가 상당히 늘어나며 안정성을 위한 테스트 작업도 추가 진행해야 한다. 설계에서의 10 분은 개발 단계에서의 10 시간이라고 여기는 것이 좋다.

홈페이지 구축이라는 사례를 통해 논리적 모델링 과정을 설명해 보았다. 논리적 모델링은 설계자의 경험과 업무의 특성에 따라 각기 다르게 만들어지는 것이다. 따라서 어떤 모델이 표준이고 좋은 모델인지 단정하기 어려운 점이 있다. 세밀하게 모델링한 결과 사용하지도 않을 개체까지 포함시키거나 지나치게 복잡한 모델이 만들어지는 경우도 있다. 개발기간을 단축하기 위해 모델링했지만, 결과적으로 개발기간이 늘어나게 됐다. 하나의 좋은 패턴을 학습하더라도 그것을 모든 업무에 적용하는 것은 금물이다.



<그림 4>개체 속성 정의



<그림 5>요약속성추가