모델링 초기 '미래 사이트 개편 염두에 둬야'

잘못 설계된 모델링은 불필요한 비용 유발 가능성

이현호 교수 / 안양과학대학 컴퓨터정보학부

연 재 순 서

1. 웹환경에서 더욱 위력적인 Oracle8i의 새로운 기능 (이번호)
2. 웹 커뮤니티 시스템 모델링(1)
3. 웹 커뮤니티 시스템 모델링(2)
4. Oracle8i의 새로운 기능을 이용한 웹 커뮤니티 시스템 구현(1)
5. Oracle8i의 새로운 기능을 이용한 웹 커뮤니티 시스템 구현(2)

이번 호에서는, 지금까지의 논의를 통해 어느 정도 분류된 엔터티들을 가지고 몇 가지 영역으로 나누어 부분적 모델링 이슈(issue)들에 대해 설명하고 전체적인 커뮤니티 시스템 ERD와 모델링 결과 테이블을 제시해 보겠다.

지난 호에서는 웹 커뮤니티 시스템의 요구사항을 가정하고, 요구사항을 충실히 반영할 수 있는 효율적인 데이터 모델링을 위한 몇 가지 중요한 개념과 엔터티의 분류 및 선정에 관해서 논의하였다. 이번 호에서는, 지난 호에 이어서 웹커뮤니티 시스템의 데이터모델링에 있어서의 몇 가지 부분적인 모델링 이슈(issue)를 짚어보고, 이를 바탕으로 한 전체적인 커뮤니티 시스템 ERD(Entity Relationship Diagram)와 물리적인 데이터베이스설계를 거친 테이블구조를 결과물로 제시하겠다.

1. 커뮤니티 시스템에서의 몇 가지 부분적인 모델링 이슈(issue)

통합코드 관리방법

본격적인 커뮤니티시스템 모델링을 논의하기에 앞서 통합코드 관리방법에 대하여 논의해 보자. 데이터는 가능하면 일반적인 텍스트 형태보다는 데이터의 내용을 분류하고 각각에 코드를 부여하여 관리하는 것이 바람직하다.

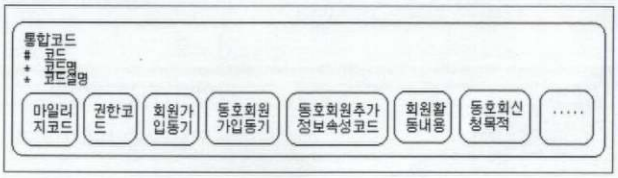
이는 향후 구축될 수 있는 데이터웨어하우스(DW)에서도 필수적으로 요구되는 사항이다. 코드화(형식화)시켜야만 분석이나 의사결정을 위한 의미있는 데이터를 뽑아낼 수 있다. 그러므로, DW의 소스 역할을 하는 OLTP(On-Line Transaction Processing) 시스템에서도 코드화가 가능한 속성은 코드화시켜서 관리하고자하는 요구가 점차 증가하고 있다.

이에 따라, 코드관리 자체도 모델링에서 매우 중요한 부분을 차지한다. 우리가 흔히 하는 실수 중에 하나가 코드를 데이터로서가 아니라 클라이언트 프로그램에서 관리한다는 것이다. 이렇게 하면, 코드의 내용이 한 번 바뀔 때마다 방대한 프로그램을 뒤져서 고쳐주어야 하는 불편함을 초래한다. 코드는 절대적으로 데이터로서 관리되어야 한다.

그런데, 코드의 종류는 많다할지라도 대체로 이와 같은 코드성 데이터는 {코드, 코드명, 코드설명} 형태의 단순하면서도 공통적인 구조를 가지고 있다. 그러므로, 이러한 코드성 데이터는

코드종류마다 테이블을 따로 두어 관리할 것이 아니라 하나의 테이블로 충분히 관리할 수 있기 때문에 모델링 단계에서 통합코드 엔터티로(물리적으로는 하나의 테이블로) 설계하는 것이 데이터 관리의 측면이나 액세스 효율의 측면에서 훨씬 바람직하다.

〈그림1〉 통합코드 엔터티



이렇게 하나의 테이블로 설계하였다 하더라도 프로그램에서는 필요에 따라서 독립적인 테이블인 것처럼 사용할 수 있다. 바로 view를 생성하여 적용하는 방법이다.(자세한 내용은 www.en-core.com 사이트의 솔루션웨어하우스란을 참조하기 바람)

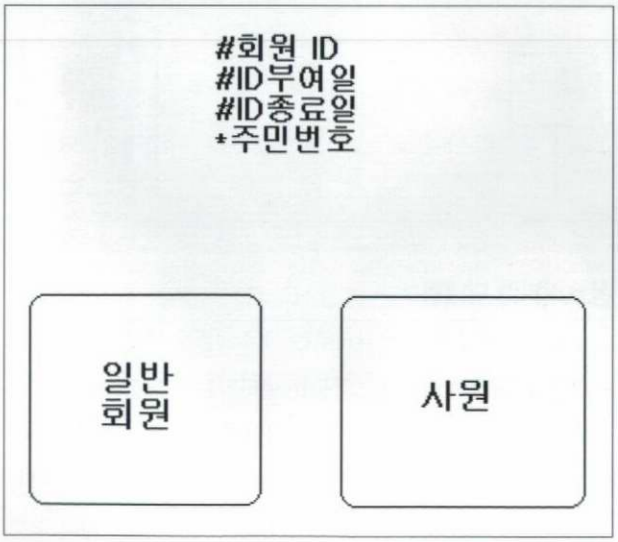
회원관련 모델링

회원, 동호회, 신청, 게시판, 회원등급, 권한, 동호회원, 마일리지 혜택정보, 통계정보, 동호회분류.

위의 논의의 결과로 선정된 엔터티 중 회원관련 엔터티로는 회원, 회원등급, 권한, 마일리지 혜택정보를 꼽을 수 있다. 모델링은 세부적인 요구사항에 따라 조금씩 달라질 수 있기 때문에 여기서는 타당성 있는 가정 하에서 진행시키겠다.

우선, 키 엔터티에 해당하는 회원 엔터티를 살펴보자. 회원은 보통 ID에 의해서 identify된다. 또, 회원에는 일반회원과 사원을 포괄하므로 이는 서브타입으로 정의할 수 있다. 서브타입은 테이블 설계 단계에서 구분자의 역할을 하는 속성으로 추가될 것이다. 그리고, 현재 사용 중이 아닌 ID를 재사용할 수 있도록 하려면 ID부여일과 ID종료일을 두어 이력을 관리해야 할 것이다. 이 외의 속성들은 관리하고 싶은 회원 정보를 결정하여 추가하면 될 것이다. 그러므로, 회원 엔터티의 모양은 〈그림2〉와 같을 것이다.

〈그림2〉 회원 엔터티

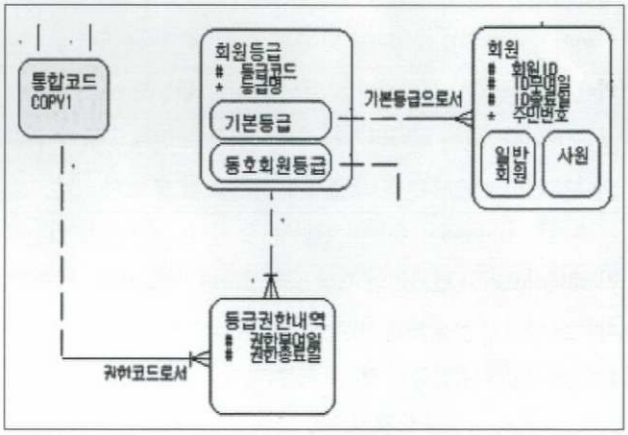


보통 ERD 표기에서 '#'으로 시작하는 속성은 UID 속성에 해당하고 '*'로 시작하는 속성은 NOT NULL 제한조건(constraint)가 있는 일반 속성을 말하며 'o'로 시작하는 속성은 nullable 속성을 나타낸다. 또, 엔터티의 UID는 부모 엔터티에서 상속받은 UID와(ERD의 relation 표기에서 '|'가 표기된 경우. 그림3의 등급권한내역 엔터티 참조) 자신의 UID('#'으로 표시된 속성)을 결합한 것이다.

회원의 등급과 권한에 관해서 살펴보기 위해 몇 가지 가정을 세워보자. 회원등급은 사이트 일반회원에 적용되는 기본등급과 동호회원에게 적용되는 동호회원등급으로 구분되고, 기본등급과 동호회원등급 모두 관리자등급을 포함한 몇 개의 등급으로 나누어진다.

회원권한은 회원등급에 따라 주어지고, 권한의 내용은 시간이 지남에 따라 바뀔 수 있다라고 가정한다. 그렇다면, 회원등급은 1,2,3,4,… 또는 A,B,C,D,… 등급으로 정의될 수 있으며, 회원권한은 이벤트 참여권한, 동호회 가입권한, 게시판 둘러보기권한, 글쓰기권한, 글수정권한, 글삭제권한 등의 여러가지 권한내용을 분류하여 정의될 수 있고, 이 권한집합의 부분집합이 특정 회원등급에 부여되는 식이 될 것이다.

〈그림3〉 회원 엔터티, 회원등급 엔터티, 등급권한내역 엔터티



그러므로, 권한은 원자화(atomic)된 권한으로 세분하 각각에 코드를 부여하여 관리하고, 등급 엔터티와 권한 엔터티를 1:M 관계의 relation으로 연결시키며, 권한 엔터티에 '권한부여일'과 '권한종료일'을 두어 시간이 지남에 따라 변할 수 있는 각 등급에 부여되는 권한내용의 이력을 관리하는 방식으로 모델링할 수 있다. 결과는 〈그림3〉과 같은 모양이 될 것이다.

마일리지 관련 엔터티로서는 마일리지 혜택정보라는 엔터티를 선정했었다. 그런데, 좀 더 일반화시켜서 생각하면, 마일리지는 회원활동을 점수화해서 관리하겠다는 의도로서, 마일리지 측면에서의 회원활동에는 크게 마일리지 포인트를 가산시키는 활동, 마일리지 포인트를 감산시키는 활동, 마일리지 포인트와 관계없는 활동으로 분류할 수 있다.

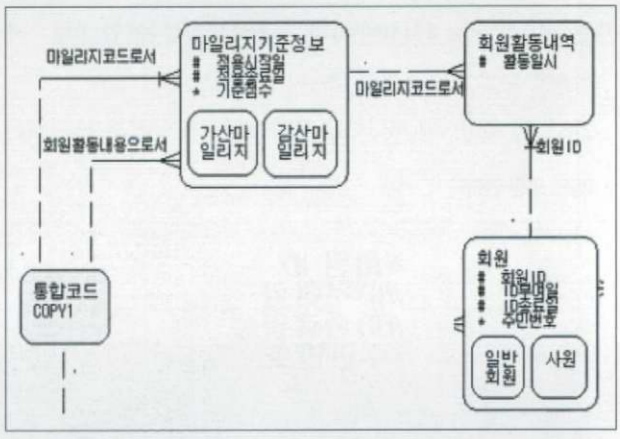
마일리지 포인트와 관계없는 활동은 우리의 관심 밖이므로 논의에서 제외하면, 마일리지 포인트를 버는 활동도 쓰는 활동으로 구분된다. 이는 데이터적인 측면에서는 포인트가 증가냐 감소냐의 차이만 존재할 뿐, 동질의 데이터로 볼 수 있다.

즉, 한 엔터티에서 충분히 관리될 수 있다. 그러므로, 마일리지 혜택정보라는 엔터티의 개념을 확장하여, 마일리지 기준정보라는 엔터티를 상정하고, 여기에서 마일리지 가산정보와 감산정보를 한꺼번에 관리하는 것이 바람직하다.

또, 회원의 마일리지 포인트에 대한 근거정보로서 회원활동내역을 관리할 수 있다. 이는 마일리지 관련 회원활동이력을 관리한다는 측면에서도 충분한 가치가 있다. 한가지 더 고려해봄직한 사항은 각 회원들의 마일리지 누적포인트를 어떻게 관리할 것인가?라는 것이다.

우선, 마일리지 누적포인트를 속성으로서 관리해야 할 것인가?의 측면이다. 누적포인트는 마일리지 기준정보와 회원활동내역에서 필요할 때 뽑아쓸 수 있다. 그러나, 누적포인트가 자주 회원에게 보여져야 한다면, 보일 때마다 관련 테이블을 액세스하여 계산을 하는 것이 부담스러울 수 있으므로, 회원 엔터티에 '누적마일리지포인트' 속성을 추가하여 관리할 수도 있다. (이러한 속성을 derived attribute라고 한다.)

〈그림4〉 회원 엔터티, 마일리지기준정보 엔터티, 회원활동내역 엔터티



동호회관련 모델링

동호회에 관련한 키엔터티로는 동호회분류 엔터티와 동호회 엔터티를 들 수 있다. 동호회분류 엔터티는 코드성 엔터티로서 {분류코드,분류코드명,코드설명} 형식의 구조를 가진다. 그런데, 동호회분류는 대분류/중분류/소분류 형식으로 다양한 분류체계를 가질 수 있기 때문에 엔터티 자체내에 순환관계를 포함

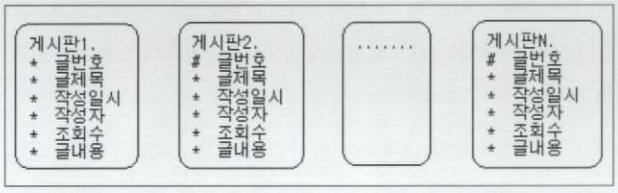
한다. 순환관계는 '상위분류코드' 속성을 추가해서 구현할 수 있다.

동호회 엔터티는 동호회 관련 모든 트랙잭션 엔터티의 조상이 되는 키 엔터티이다. 여기에서, 동호회ID, 동호회명, 생성일, 폐기일 등의 속성이 관리된다.

그리고, 동호회의 내용적 근간을 이루는 게시판 엔터티의 부모가 된다. 여기서 한가지 고려해야 할 것은 관련동호회 부분이다. 앞부분에서 관련동호회는 엔터티가 아니라 관계(Relationship)이나 M:M 관계를 가지므로 교차엔터티로 설계됨이 바람직하다라고 밝힌 바 있다.

게시판 엔터티는 2NF(제2정규화)에 의해서 게시판 자체의 내용정보를 담는 게시판HEADER 엔터티와 게시판에 올라온 글정보를 관리하는 게시판BODY 엔터티로 분리되어 관리되어야 한다. 우리가 게시판 엔터티를 정의함에 있어서 흔히 저지를 수 있는 잘못은 게시판이 새로 만들어질 때마다 그에 해당하는 테이블을 생성하는 방식을 택한다는 것이다.

〈그림5〉 잘못 설계된 게시판 엔터티

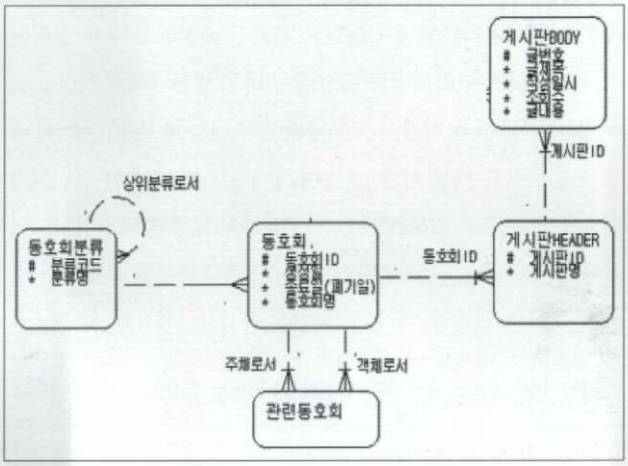


이는 절대로 피해야될 모델링이다. 게시판은 수시로 생성/폐기 될 수 있는데, 게시판 당 테이블 하나씩을 할당하여 관리한다면 이러한 생성/폐기에 엄청난 오버헤드를 가져올 수 있다.(때로는 프로그램을 고쳐야 하는 경우도 발생할 수 있다.) 이는 시스템의 유지/보수를 굉장히 어렵게 한다. 또한, 이는 엔터티들 간의 독립성을 확보해야 한다는 모델링 원칙에도 위배된다.

혹자는 게시판은 액션 엔터티로서 데이터량이 방대해질 수 있기 때문에 할 수 없이 이런식으로 관리할 수밖에 없다라고 항변할 수 있다. 그러나, 이러한 문제는 partition table 개념을 도입하거나 인덱스의 적절한 설계 등의 물리적인 모델링 단계에서 풀어야 할 문제이지, 엔터티 즉 집합 자체의 정의를 흔들면서까지 논리적(개념적)인 모델링 단계에서 풀어야 할 문제가 아니라는 것을 명심해야 할 것이다. 논리적인 모델링의 원칙을 지키면, 게시판HEADER 엔터티와 게시판BODY 엔터티 두 개로서 충분히 구현될 수 있다.

이와 같은 내용을 종합하면, 동호회 관련 모델링 부분은 〈그림6〉과 같은 모양이 될 것이다.

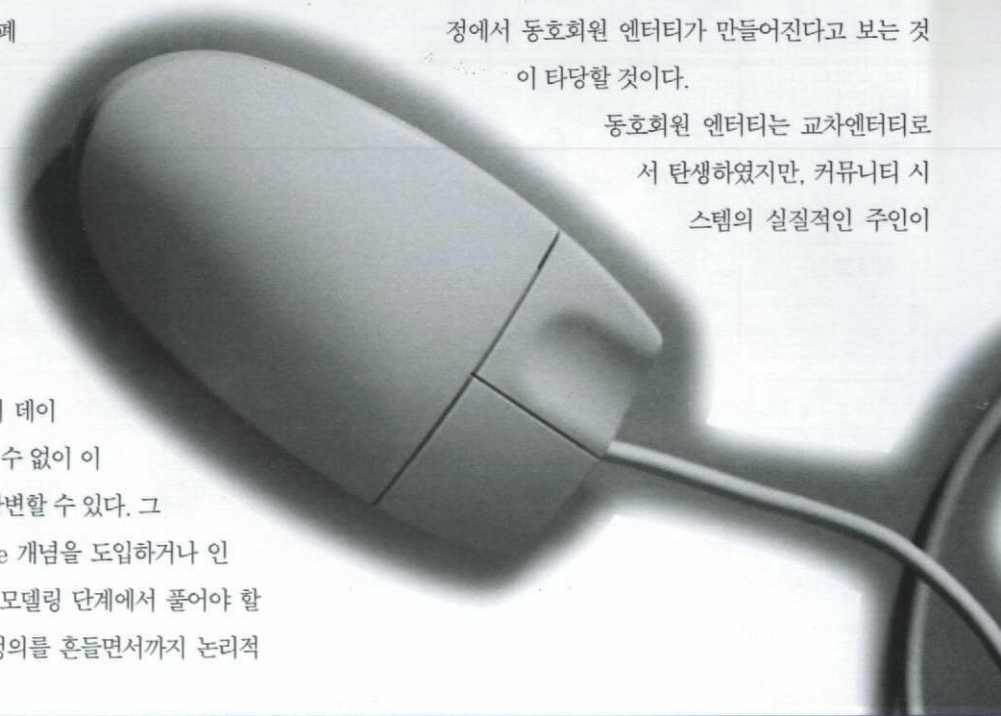
〈그림6〉 동호회분류, 동호회, 게시판HEADER, 게시판BODY, 관련동호회 엔터티



회원과 동호회 간의 관계

회원과 동호회의 관계(relationship)에서 탄생하는 엔터티가 바로 동호회원 엔터티이다. 일반적으로 한 회원은 여러 동호회에 가입할 수 있고, 동호회는 여러 회원으로 구성되기 때문에 두 엔터티는 M:M 관계를 가진다. 이 M:M 관계를 해소하는 과정에서 동호회원 엔터티가 만들어진다고 보는 것이 타당할 것이다.

동호회원 엔터티는 교차엔터티로서 탄생하였지만, 커뮤니티 시스템의 실질적인 주인이



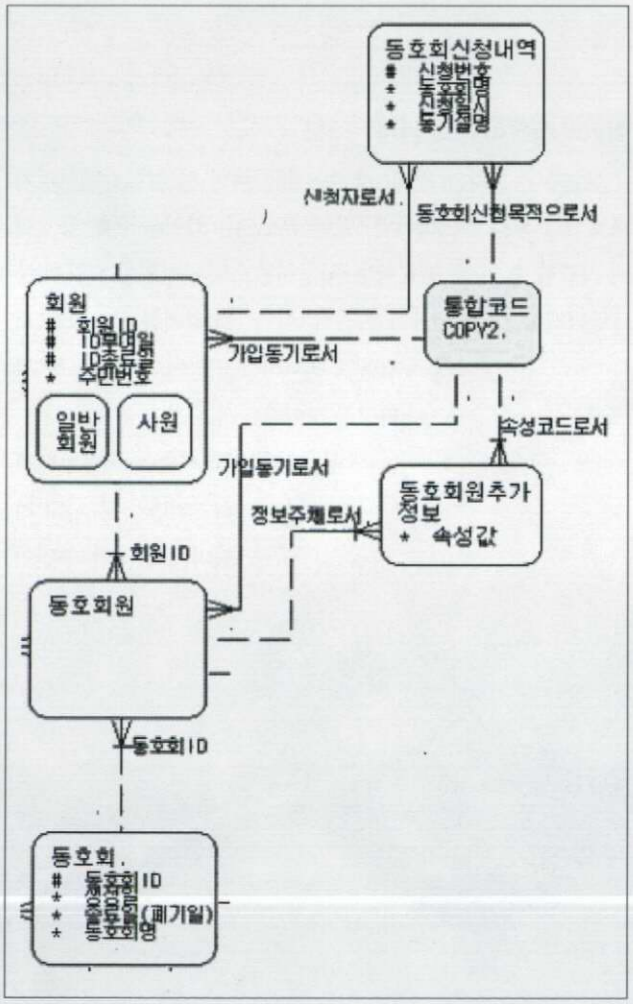
고 시스템 요구사항에 따라 여러 자식을 거느리는 부모의 역할을 할 수도 있으므로 액션 엔터티라기 보다는 메인 엔터티로 분류하는 것이 타당할 것이다.

여기서 동호회원 추가정보에 대해서 잠깐 살펴보자. 동호회마다 회원들의 특징적인 정보들이 더 필요할 수도 있다. 이러한 정보들은 성격상 동호회원 엔터티의 속성이지 독립적인 엔터티는 아니다라고 이미 밝힌 바 있다. 그런데, 문제는 동호회마다 관리해야할 속성의 종류가 많이 다를 수 있다는 것이다.

동호회원 엔터티에 추가정보에 관련된 속성들을 모두 정의하여 사용한다면 관리해야할 속성들이 너무 많은데다가 NULL값이 많이 분포하여 저장공간의 낭비를 초래할 수 있다. 또한, 새로운 동호회의 탄생 시에 그 동호회가 필요한 속성이 정의되지 않았을 경우, 동호회원 엔터티를 수정하지 않으면 안된다.

이와 같은 문제를 해결하기 위해서, 추가정보를 동호회원 엔터티와 M:1 관계를 가지고 {속성명,속성값} 형식의 속성을 갖는 엔터티로 정의하는 방법이 있을 수 있다. 예를 들어, 스키 동호회에서 동호회원의 스키보유 유무를 관리하고자 한다면, 추가정보 엔터티에서 속성명 attribute에 '스키보유유무', 속성값 attribute에 'Y' or 'N'의 값을 저장하여 관리하는 방식이다. 이 방식을 채택하면, 동호회에서 새로운 속성이 필요하다고 할지라도 테이블 구조를 변경시키지 않고 레코드 하나만 추가시키는 것으로 충분히 관리된다. 다만, 속성명은 가능하면 코드화시켜서 관리하는 것이 바람직해 보인다.

〈그림7〉 회원, 동호회, 동호회원, 동호회원추가정보, 동호회신청내역 엔터티



또 한가지 엔터티로서 상정할 수 있는 것은 동호회신청내역 엔터티이다. 회원의 동호회신청 행위를 데이터로서 관리하고자 한다면 필요한 엔터티이다. 동호회신청은 회원이면 누구나 할 수 있고 신청행위자체를 관리하므로 이 엔터터는 회원 엔터티를 부모로 하는 액션 엔터티로 보는 것이 타당하다. 한가지 더 살펴봄직한 것은 이 엔터티와 동호회 엔터티 간의 관계이다.

동호회 신청을 통해서 하나의 동호회가 탄생하는 것은 사실이지만, 이 엔터티와 동호회 엔터티는 데이터적으로 직접적인 관련은 없다고 보는 것이 타당하다. 왜냐하면, 관리하고자 하는 데이터의 성격이 다르다.(동호회 엔터티는 동호회 자체정보를, 동호회신청 엔터티는 신청행위정보를 각각 관리한다.) 또한, ERD가 원인과 결과 내지는 데이터의 흐름을 기술하는 다이어그램이 아니라 엔터티 간의 관계를 기술하는 다이어그램이기 때문이다.

이와 같은 내용을 종합하면, 회원과 동호회 간의 관계부분은 〈그림7〉과 같은 모양이 될 것이다.

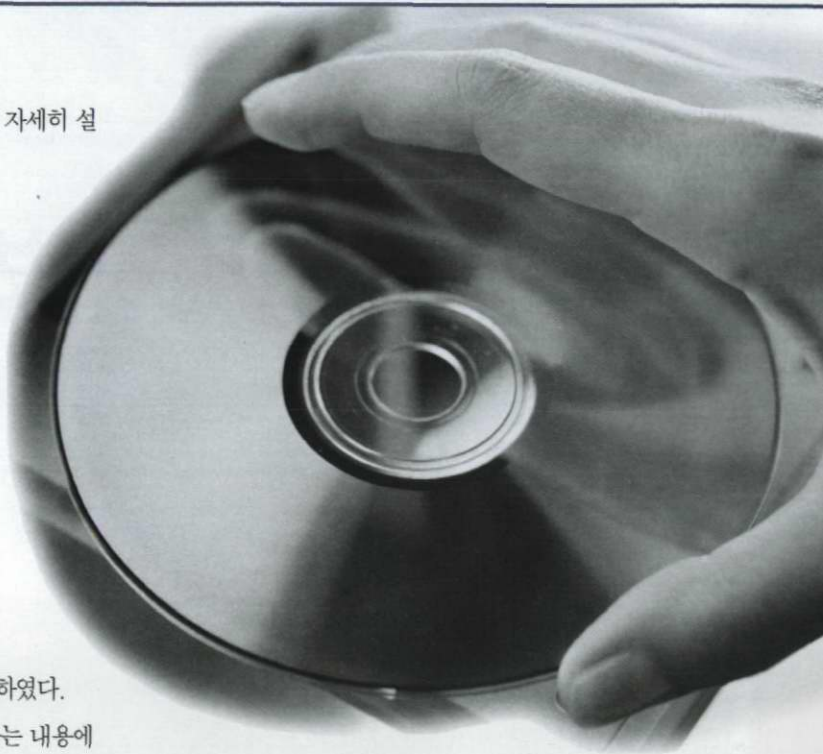
통계정보

총회원수, 동호회별 동호회원수, 월별 동호회별 글등록수/글조회수 등 통계정보는 사이트를 운영하는 주체의 이해와 요구에 따라 다양하게 정의될 수 있다. 그러나, 이러한 통계정보는 OLTP 시스템 상에서 독자적인 데이터 발생영역으로 존재하는 것이 아니라, OLTP 시스템에 존재하는 여러 테이블이 가지고 있는 데이터를 가공하여 추출하는 것이다.

이러한 테이블을 집계테이블이라고 한다. 집계테이블은 OLTP 상의 어떤 테이블과도 직접적인 관계(relationship)을 가지지 않는다. 이러한 집계테이블은 구현하기에 아주 적합한 도구가 오라클 8.1.6에서 도입된 materialized view이다. Materialized view에 대해서는 이전회에서 상세히 설명했으니 참고 바란다.

본 커뮤니티 시스템의 통계정보를 materialized view로 구현

하는 것에 대해서는 추후 시스템 구현부분을 다룰 때 자세히 설명하기로 한다.



2. 전체적인 커뮤니티 시스템 ERD와 모델링 결과 테이블

이상에서 서술한 내용을 종합하여 전체적인 ERD를 그리면 다음과 같다.

- 통합코드COPY1과 통합코드COPY2는 통합코드 엔터티의 복사본을 뜻함.
- 통계정보의 MV는 materialized view임을 의미.

엔터티의 속성들은 주요한 속성들 위주로만 표기하였다. 세부적인 속성들은 각각의 엔터티에서 관리하고자 하는 내용에 따라 결정될 것이다.

ERD가 완성되면 물리적인 데이터베이스설계 단계로 들어간다. 이는 엔터티를 실제적인 테이블로 설계하는 단계로서 서브타입의 처리, 수평분할, 수직분할, 반정규화 등의 여러가지 고려해야할 작업들이 있으나 이에 관해서는 언급하지 않겠다. (자세한 내용은 www.en-core.com 사이트의 모델링 관련자료를 참조하기 바란다.)

통합코드	{ 코드타입(subtype), 코드, 코드명, 코드설명, … }
회원	{ 회원ID, ID부여일, ID종료일, 주민번호, 회원등급, 회원분류(subtype), … }
회원등급	{ 등급코드, 등급명, 등급구분(subtype), … }
등급권한내역	{ 회원등급, 권한코드, 권한부여일, 권한종료일, … }
마일리지기준정보	{ 마일리지코드, 적용시작일, 적용종료일, 회원활동내용, 기준점수, 가감산분류(subtype), … }
회원활동내역	{ 회원ID, 활동일시, 마일리지코드, … }
동호회분류	{ 분류코드, 분류명, 상위분류코드, … }
동호회	{ 동호회ID, 생성일, 폐기일, 동호회명, … }
관련동호회	{ 주체동호회ID, 객체동호회ID, … }
게시판HEADER	{ 동호회ID, 게시판ID, 게시판명, … }
게시판BODY	{ 동호회ID, 게시판ID, 글번호, 글제목, 작성일시, 작성자, 조회수, 글내용, … }
동호회원	{ 동호회ID, 회원ID, 동호회원등급, 가입동기, … }
동호회원추가정보	{ 동호회원ID, 속성코드, 속성값, … }
동호회신청내역	{ 신청번호, 동호회명, 신청일시, 신청자, 신청목적, 동기설명, … }

* 밑줄 친 속성은 primary key를 나타냄.

다만, 본 커뮤니티 시스템에서의 데이터베이스설계의 원칙은 하나의 엔터티는 하나의 테이블로 설계하고, 서브타입(subtype)은 속성(attribute)으로 처리한다는 것이다. 이와 같은 원칙하에 위 ERD의 물리적인 테이블은 좌측과 같이 정의될 수 있다.

이상으로 커뮤니티 시스템의 모델링에 대해서 살펴보았다. 모델링은 대충 넘어갈 수 있는 부분이 아님을 다시 한 번 강조한다. 모델의 차이는 구현단계에서 더 확연하게 드러날 것이고, 유지/보수의 승패를 좌우한다고 해도 과언이 아님을 명심해야 할 것이다.

다음 회에서는 완성된 모델을 바탕으로 첫 회에서 소개한 오라클 8i에 도입된 새로운 기능인 -analytic function, function-based index, materialized view-를 이용한 웹 커뮤니티 시스템을 구현에 대해서, '게시판의 부분범위처리', '집계성 데이터의 효율적인 관리' 등의 핵심적인 기능을 중심으로 소개하고자 한다.

참고문헌 및 참고사이트

(1) DATA MODELING & DATABASE DESIGN, DB 모델링 교육용 교재, 엔코아정보컨설팅, 1997.

(2) 엔코아정보컨설팅 홈페이지(www.en-core.com) 솔루션웨어하우스(solution warehouse)