4. 데이터베이스

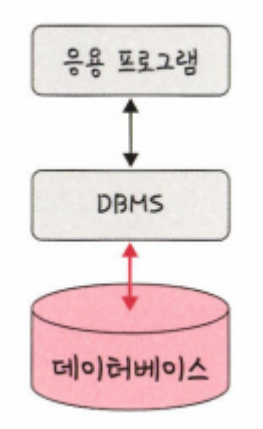
▼ 목차

4-1. 데이터베이스의 기본

데이터베이스는 일정한 규칙 혹은 규약을 통해 구조화되어 저장되는 데이터의 모음.

해당 데이터베이스를 제어, 관리하는 통합 시스템을 DBMS(DataBase Management System)

데이터베이스 안에 있는 데이터들은 특정 DBMS마다 정의된 쿼리 언어를 통해 삽입, 삭제, 수정, 조회 등을 수행할 수 있다. 또한 데이터베이스는 실시간 접근과 동시 공유가 가능하다.



그림처럼 데이터베이스 위에 DBMS 가 있고
그 위에 응용 프로그램이 있으며 이러한 구조를 기반으로 데이터를 주고 받는다.
예를 들어 MySQL이라는 DBMS가 있고 그 위에 응용 프로그램에 속하는 node.js나 php에

해당 데이터베이스 안에 있는 데이터를 끄집어내 해당 데이터 관련 로직을 구축할 수 있는 것이다.

4-1-1. 엔터티

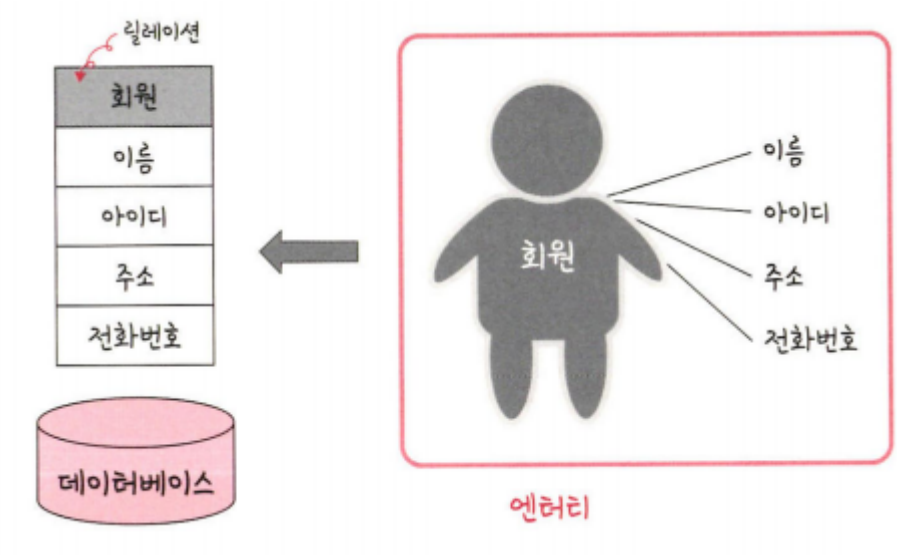
엔터티는 사람 장소 물건 사건 개념 등 여러 개의 속성을 지닌 명사를 의미
예를 들어 회원이라는 엔터티가 있다고 한다면 회원 이름 아이디 주소 전화번호 같은 속성이 있는 것이다.

약한 엔터티와 강한 엔터티

A가 혼자서는 존재하지 못하고 B의 존재 여부에 따라 종속적이라면 A는 약한 엔터티이고 B는 강한 엔터티가 된다.

4-1-2. 릴레이션

데이터베이스에서 정보를 구분하여 저장하는 기본 단위
엔터티에 관한 데이터를 데이터베이스는 릴레이션 하나에 담아서 관리



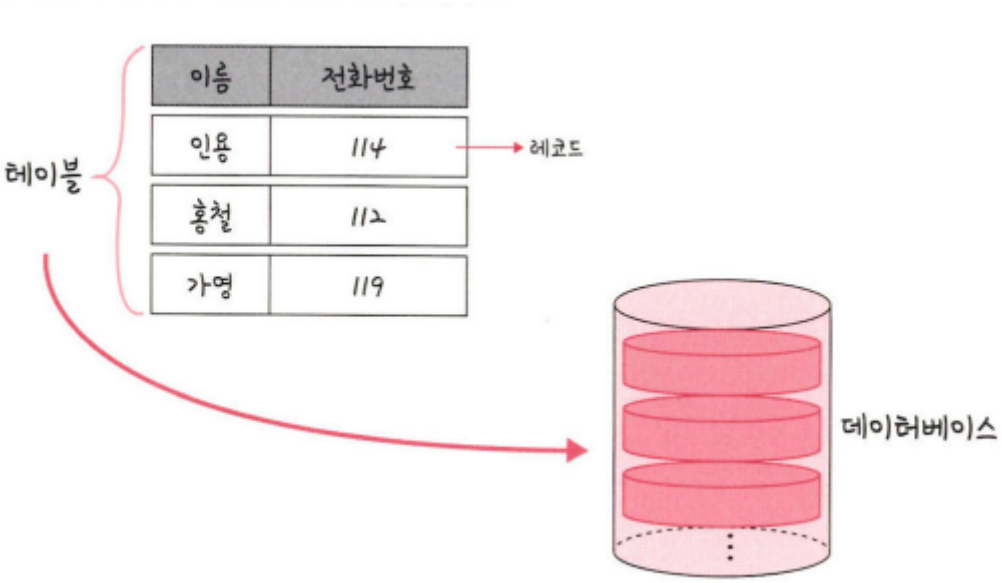
그림처럼 회원이라는 엔터티가 데이터베이스에서 관리될 때 릴레이션으로 변화된 것을 볼 수 있다.
릴레이션은 관계형 데이터베이스에서는 테이블이라고 하며, NoSQL 데이터베이스는 컬렉션이라고 한다

테이블과 컬렉션

데이터베이스의 종류는 크게 관계형 데이터베이스와 NoSQL 비관계형 데이터베이스로 나눔

이 중 대표적인 관계형 데이터베이스인 MySQL과
대표적인 NoSQL 데이터베이스인 MongoDB를 예로 들면,

MySQL의 구조는 레코드-테이블-데이터베이스로 이루어져 있고
NoSQL 데이터베이스의 구조는 도큐먼트-컬렉션-데이터베이스로 이루어져 있다.



그림처럼 레코드가 쌓여서 테이블이 되고 테이블이 쌓여서 데이터베이스가 되는 것이다.

4-1-3. 속성

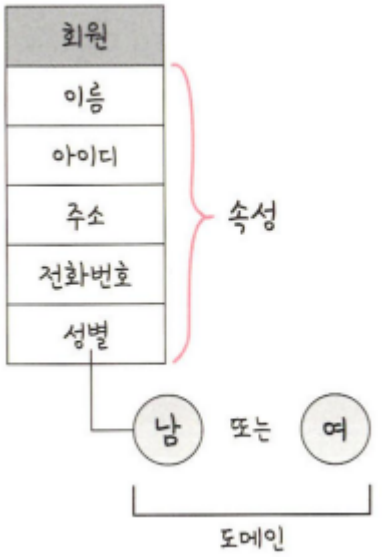
속성은 릴레이션에서 관리하는 구체적이며 고유한 이름을 갖는 정보

예를 들어 '차'라는 엔터티의 속성을 뽑아보자. 차 넘버, 바퀴 수, 차 색깔, 차종 등이 있겠다. 이 중에서 서비스의 요구 사항을 기반으로 관리해야 할 필요가 있는 속성들만 엔터티의 속성이 된다.

4-1-4. 도메인

릴레이션에 포함된 각각의 속성들이 가질 수 있는 값의 집합을 말한다

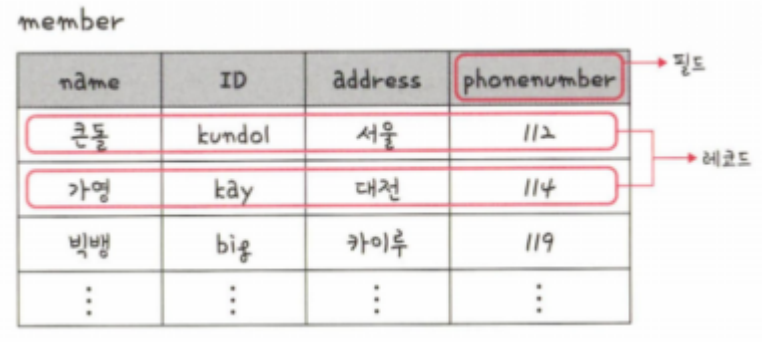
예를 들어 성별이라는 속성이 있다면 이 속성이 가질 수 있는 값은 (남, 여)라는 집합이 된다



그림처럼 회원이라는 릴레이션에 이름 아이디 주소 전화번호 성별이라는 속성이 있고 성별은 (남, 여)라는 도메인을 가지는 것을 알 수 있다.

4-1-5. 필드와 레코드

앞에서 설명한 것들을 기반으로 데이터베이스에서 필드와 레코드로 구성된 테이블을 만들 수 있다



회원이란 엔터티는 member라는 테이블로 속성인 이름, 아이디 등을 가지고 있으며 name id address 등의 필드를 가진다. 그리고 이 테이블에 쌓이는 행 단위의 데이터 레코드라고 한다. 또한 레코드를 튜블이라고도 한다.

필드 타입

필드는 타입을 갖는다. 예를 들어 이름은 문자열이고 전화번호는 숫자다. 이러한 타입들은 DBMS마다 다르다. 아래는 MySQL 을 기준으로 설명한 것이다.

숫자 타입

숫자 타입으로는 TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT 등이 있습니다.

▼ 표 4-1 MySQL 숫자 타입

타입	용량(바이트)	최솟값(부호 있음)	최솟값(부호 없음)	최댓값(부호 없음)	최댓값(부호 있음)
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-263	0	263-1	264-1

날짜 타입

날짜 타입으로는 DATE, DATETIME, TIMESTAMP 등이 있습니다.

DATE

날짜 부분은 있지만 시간 부분은 없는 값에 사용됩니다. 지원되는 범위는 1000-01-01~9999-12-31입니다. 3바이트의 용량을 가집니다.

DATETIME

날짜 및 시간 부분을 모두 포함하는 값에 사용됩니다. 지원되는 범위는 1000-01-01 00:00:00에서 9999-12-31 23:59:59입니다. 8바이트의 용량을 가집니다.

TIMESTAMP

날짜 및 시간 부분을 모두 포함하는 값에 사용됩니다. 1970-01-01 00:00:01에서 2038-01-19 03:14:07까지 지원합니다. 4바이트의 용량을 가집니다.

문자 타입

문자 타입으로는 CHAR, VARCHAR, TEXT, BLOB, ENUM, SET이 있습니다.

CHAR와 VARCHAR

CHAR 또는 VARCHAR 모두 그 안에 수를 입력해서 몇 자까지 입력할지 정합니다. 예를 들어 CHAR(30)이라면 최대 30글자까지 입력할 수 있습니다.

CHAR는 테이블을 생성할 때 선언한 길이로 고정되며 길이는 0에서 255 사이의 값을 가집니다. 레코드를 저장할 때 무조건 선언한 길이 값으로 '고정'해서 저장됩니다.

VARCHAR는 가변 길이 문자열입니다. 길이는 0에서 65,535 사이의 값으로 지정할 수 있으며, 입력된 데이터에 따라 용량을 가변시켜 저장합니다. 예를 들어 10글자의 이메일을 저장할 경우 10글자에 해당하는 바이트 + 길이기록용 1바이트로 저장하게 됩니다. VARCHAR(10000)으로 선언했음에도 말이죠.

그렇기 때문에 지정된 형태에 따라 저장된 CHAR의 경우 검색에 유리하며, 검색을 별로 하지 않고 유동적인 길이를 가진 데이터는 VARCHAR로 저장하는 것이 좋습니다.

TEXT와 BLOB

두 개의 타입 모두 큰 데이터를 저장할 때 쓰는 타입입니다.

TEXT는 큰 문자열 저장에 쓰며 주로 게시판의 본문을 저장할 때 씁니다.

BLOB은 이미지, 동영상 등 큰 데이터 저장에 씁니다. 그러나 보통은 아마존의 이미지 호스팅 서비스인 S3를 이용하는 등 서버에 파일을 올리고 파일에 관한 경로를 VARCHAR로 저장합니다.

▼ 그림 4-8 아마존의 S3

ENUM과 SET

ENUM과 SET 모두 문자열을 열거한 타입입니다.

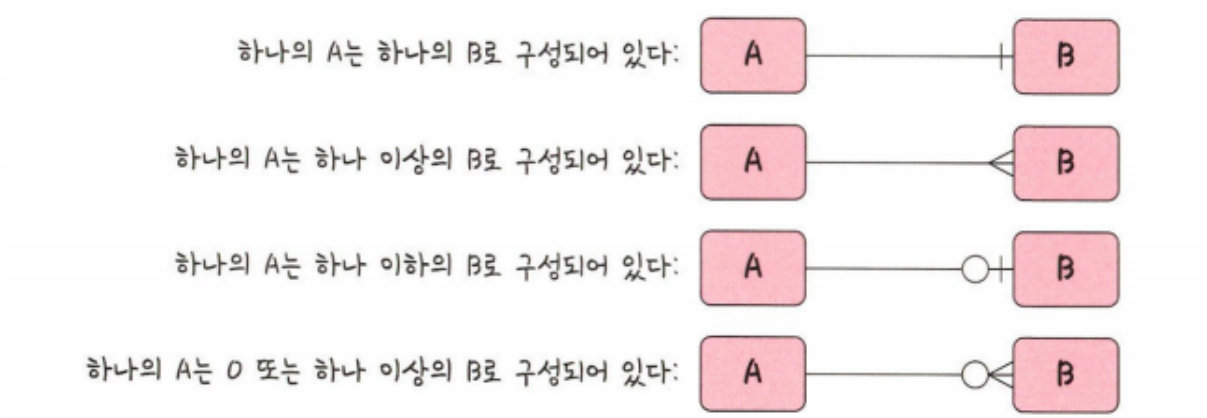
ENUM은 ENUM('x-small', 'small', 'medium', 'large', 'x-large') 형태로 쓰이며, 이 중에서 하나만 선택하는 단일 선택만 가능하고 ENUM 리스트에 없는 잘못된 값을 삽입하면 빈 문자열이 대신 삽입됩니다. ENUM을 이용하면 x-small 등이 0, 1 등으로 매핑되어 메모리를 적게 사용하는 이점을 얻습니다. ENUM은 최대 65,535개의 요소들을 넣을 수 있습니다.

SET은 ENUM과 비슷하지만 여러 개의 데이터를 선택할 수 있고 비트 단위의 연산을 할 수 있으며 최대 64개의 요소를 집어넣을 수 있다는 점이 다릅니다.

참고로 ENUM이나 SET을 쓸 경우 공간적으로 이점을 볼 수 있지만 애플리케이션의 수정에 따라 데이터베이스의 ENUM이나 SET에서 정의한 목록을 수정해야 한다는 단점이 있습니다.

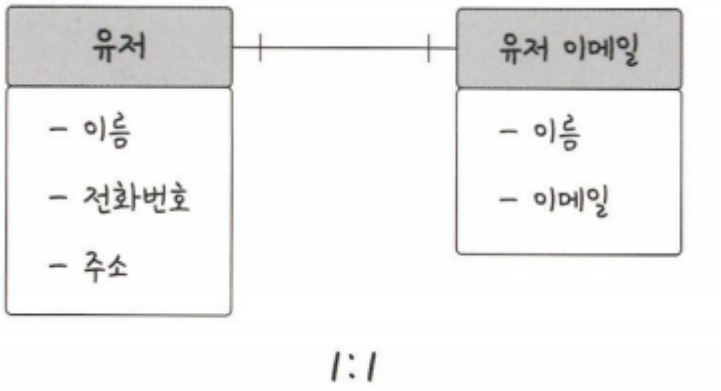
4-1-6. 관계

데이터베이스에 테이블은 하나만 있는 것이 아니다
여러 개의 테이블이 있고 이러한 테이블은 서로의 관계가 정의되어 있다. 이러한 관계를 관계화살표로 나타낸다



1:1 관계

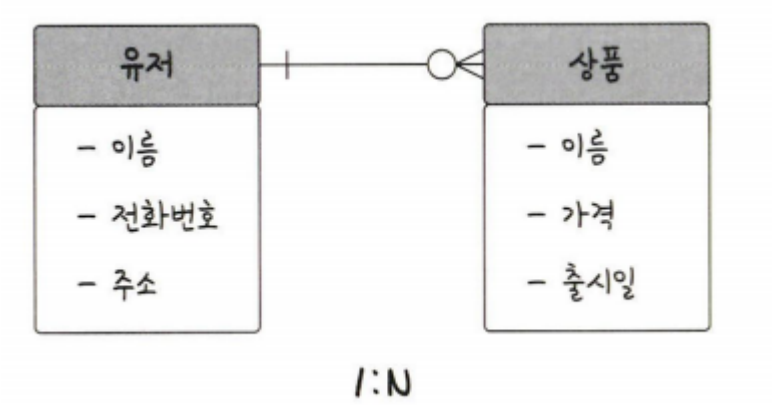
예를 들어 유저당 유저 이메일은 한 개씩 있을 것. 이 경우 1:1 관계가 된다



1:1 관계는 테이블을 두 개의 테이블로 나눠 테이블의 구조를 더 이해하기 쉽게 만들어 준다

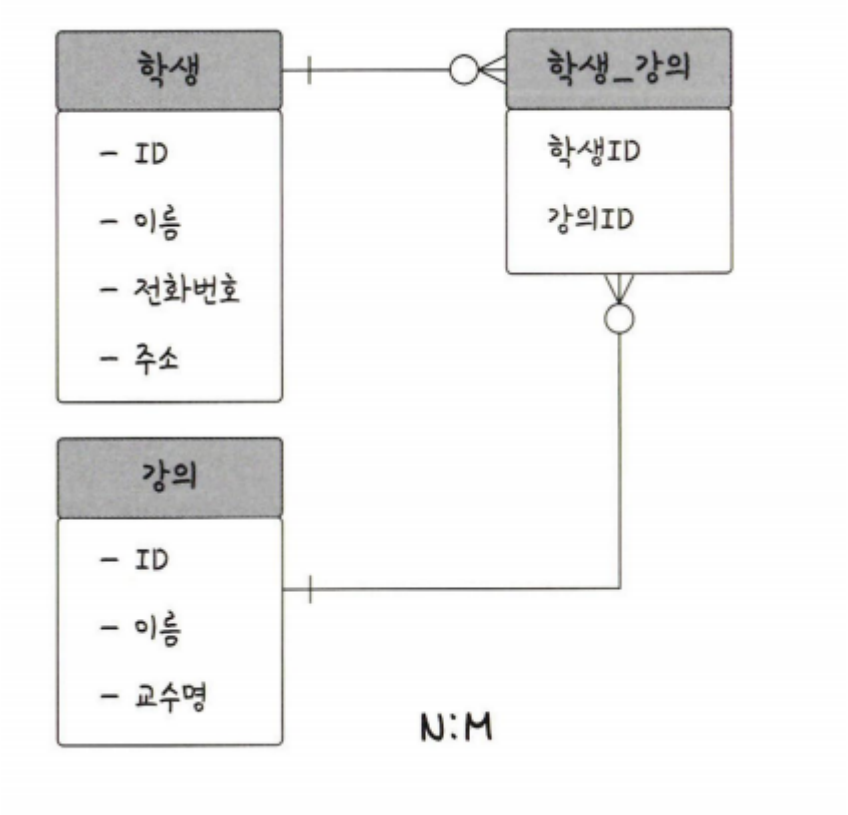
1:N 관계

쇼핑몰을 운영한다고 하면 한 유저당 여러 개의 상품을 장바구니에 넣을 수 있다
이 경우가 1:N 관계가 된다. 한 개체가 다른 많은 개체를 포함하는 관계를 말한다.



N:M 관계

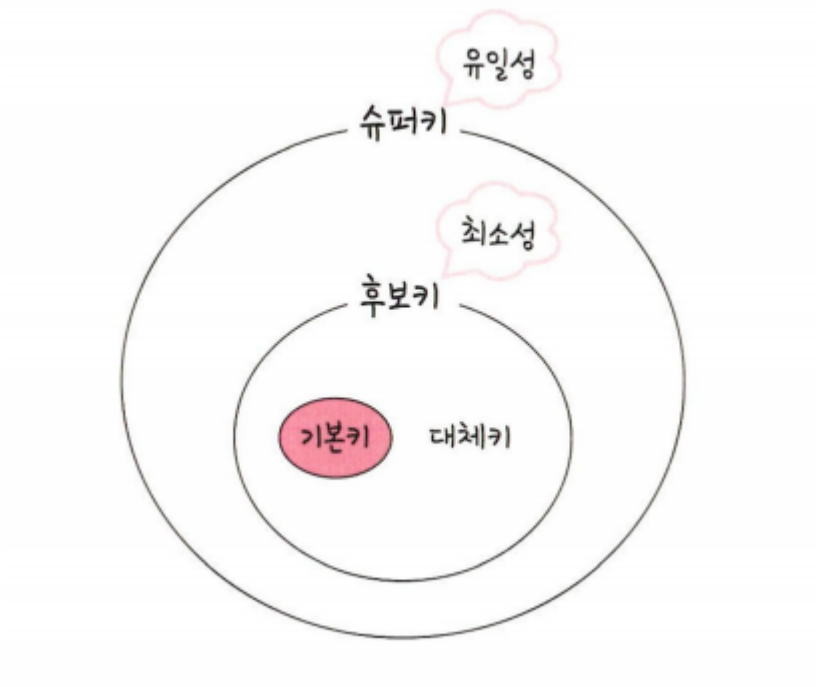
학생과 강의의 관계를 정의한다면 학생도 강의를 많이 들을 수 있고 강의도 여러 명의 학생을 포함할 수 있다. 이러한 관계가 N:M 관계이다



중간에 학생_강의 라는 테이블이 끼어 있다. N:M은 테이블 두 개를 직접적으로 연결해서 구축하지 않고 1:N, 1:M 이라는 관계를 갖는 테이블 두 개로 나눠서 설정한다

4-1-7. 키

테이블 간의 관계를 조금 더 명확하게 하고 테이블 자체의 인덱스를 위해 설정된 장치로 기본키, 외래키, 후보키, 슈퍼키, 대체키가 있다



키들은 앞의 그림과 같은 관계를 가진다.
슈퍼키는 유일성이 있고, 그 안에 포함된 후보 키는 최소성까지 갖춘 키이다. 후보키 중에서 기본키로 선택되지 못한 키는 대체키가 된다. 유일성은 중복되는 값은 없으며, 최소성은 필드를 조합하지 않고 최소 필드만 써서 키를 형성할 수 있는 것을 말한다.

기본키

기본키는 줄여 PK 또는 프라이머리키라고 많이 부르며, 유일성과 최소성을 만족하는 키이다

기본적으로 중복되는 값은 기본키 값이 될 수 없다

ID	name
1	주홍철
2	주홍철
3	최범석
4	양기영

아이디와 이름이라는 복합키를 기본키로 설정할 수 있지만 그렇게 되면 최소성을 만족하지 않는다
기본키는 자연키 또는 인조키 중에 골라 설정한다.

자연키

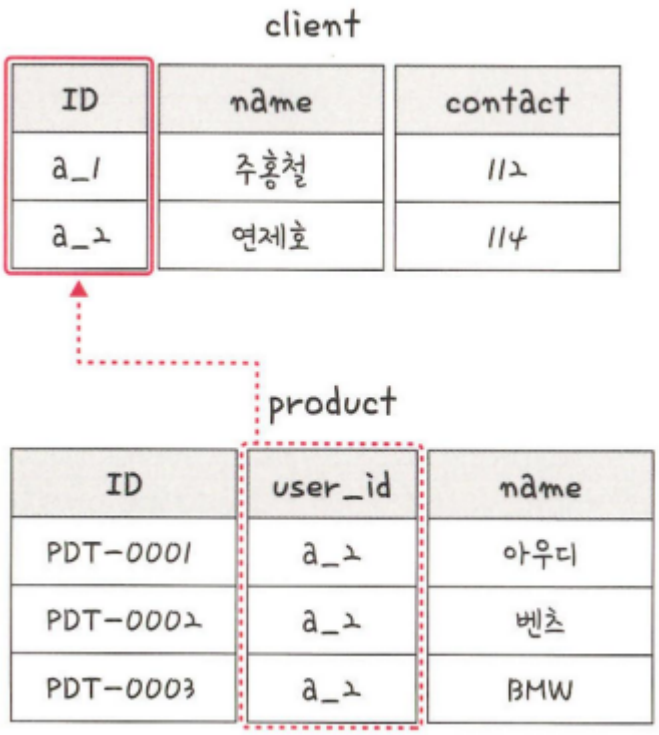
유저 테이블을 만든하고 가정하면 주민번호, 이름, 성별 등의 속성이 있다. 이 중 이름, 성별 등은 중복된 값이 들어올 수 있으므로 부적절하고 남는 것은 주민번호이다. 이런 식으로 중복된 값들을 제외하며 중복되지 않는 것을 자연스레 뽑다가 나오는 키를 자연키라고 한다.
자연키는 언젠가는 변하는 속성을 가진다.

인조키

유저 테이블을 만든다고 했을 때 회원 테이블을 생성한다고 가정하면 주민 번호 이름 성별 등의 속성이 있다. 여기에 인위적으로 유저 아이디를 부여한다. 이를 통해 고유 식별자가 생겨난다. 오라클은 sequence mysql은 auto increment 등으로 설정한다. 이렇게 인위적으로 생성한 키를 인조키라고 한다.
자연키와는 대조적으로 변하지 않는다. 따라서 보통 기본키는 인조키로 설정한다.

외래키

외래키는 FK라고도 하며, 다른 테이블의 기본키를 그대로 참조하는 값으로 개체와의 관계를 식별하는데 사용한다.



외래키는 중복되어도 괜찮다. 앞의 그림을 보면 client 라는 테이블의 기본키인 id가 product 라는 테이블의 user_id 라는 외래키로 설정될 수 있음을 보여준다. 또한 user_id는 a_2라는 값이 중복되는 것을 볼 수 있다.

후보키

기본키가 될 수 있는 후보들이며 유일성과 최소성을 동시에 만족하는 키

대체키

후보키가 두 개 이상일 경우 어느 하나를 기본키로 지정하고 남은 후보키들을 말한다

슈퍼키

각 레코드를 유일하게 식별할 수 있는 유일성을 갖춘 키

4-2. ERD와 정규화 과정

ERD (Entity Relationship Diagram)는 데이터베이스를 구축할 때 가장 기초적인 뼈대 역할을 하며, 릴레이션 간의 관계들을 정의한 것이다.
만약 서비스를 구축한다면 가장 먼저 신경 써야할 부분이며 이부분을 신경 쓰지 않고 서비스를 구축한다면 단단하지 않은 골조로 건물을 짓는 것이나 다름 없다

4-2-1. ERD의 중요성

ERD는 시스템의 요구 상항을 기반으로 작성되며 이 ERD를 기반으로 데이터베이스를 구축한다.

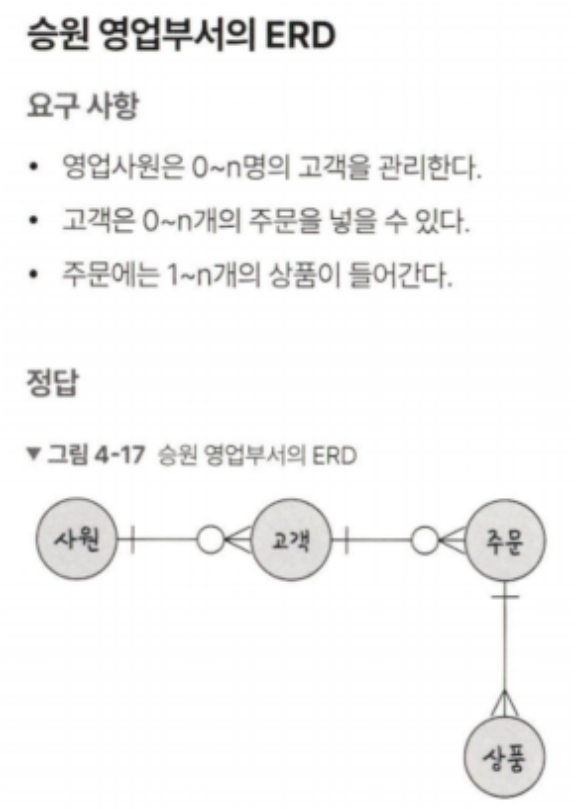
데이터베이스를 구축한 이후에도 디버깅 또는 비즈니스 프로세스 재설계가 필요한 경우에 설계도 역할을 담당하기도 한다.

하지만 ERD는 관계형 구조로 표현할 수 있는 데이터를 구성하는 데 유용할 수 있지만 비정형 데이터를 충분히 표현할 수 없다는 단점이 있다

- 비정형 데이터
  비구조화 데이터를 말하며, 미리 정의된 데이터 모델이 없거나 미리 정의된 방식으로 정리되지 않은 정보를 말한다.

4-2-2. 예제로 배우는 ERD

다음은 예제의 서비스 요구 사항과 답을 기반으로 ERD를 작성하며 공부해 보자
참고로 정답 ERD의 테이블 필드 타입은 생략



4-2-3. 정규화 과정

정규화 과정은 릴레이션 간의 잘못쇤 종속 관계로 인해 데이터베이스 이상 현상이 일어나서 이를 해결하거나, 저장 공간을 효율적으로 사용하기 위해 릴레이션을 여러 개로 분리하는 과정

데이터베이스 이상 현상이란 회원이 한 개의 등급을 가져야 하는데 세 개의 등급을 갖거나 삭제할 때 필요한 데이터가 같이 삭제되고, 데이터를 삽입해야 하는데 하나의 필드 값이 NULL이 되면 안 되어서 삽입하기 어려운 현상을 말한다.

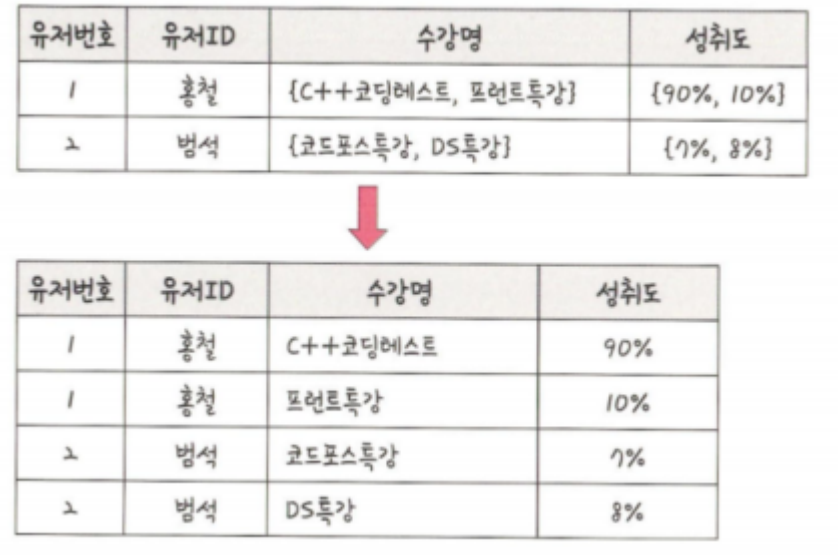
정규화 과정은 정규형 원칙을 기반으로 정규형을 만들어가는 과정이며 정규화된 정도는 정규형으로 표현한다. 기본 정규형인 제1정규형 제2정규형 제3정규형. 보이스/코드 정규형이 있고 고급 정규형인 제4정규형 제5정규형이 있다. 이 중 기본 정규형인 제 123 정규형 보이스/코드 정규형을 알아보자

정규형 원칙

정규형의 원칙이란 같은 의미를 표현하는 릴레이션이지만 좀 더 ㄷ좋은 구조로 만들어야 하고 자료의 중복성은 감소해야 하고, 독립적인 관계는 별개의 릴레이션으로 표현해야 하며 각각의 릴레이션은 독립적인 표현이 가능해야 하는 것을 말한다.

제 1 정규형

릴레이션의 모든 도메인이 더 이상 분해될 수 없는 원자 값만으로 구성되어야 한다. 릴레이션의 속성 값 중에서 한 개의 기본키에 대해 두 개 이상의 값을 가지는 반복 집합이 있어서는 안된다. 만약에 반복 집합이 있다면 제거해야 한다.



유저번호	유저ID	수강명	성취도
1	홍철	{C++코딩테스트, 프런트특강}	{90%, 10%}
2	범석	{코드포스특강, DS특강}	{7%, 8%}

유저번호	유저ID	수강명	성취도
1	홍철	C++코딩테스트	90%
1	홍철	프런트특강	10%
2	범석	코드포스특강	7%
2	범석	DS특강	8%

그림처럼 홍철이라는 id에 수강명이 {C++코딩테스트, 프런트특강} 이었는데 이것을 나눠서 반복 집합을 제거하는 것을 볼 수 있다.

제 2 정규형

릴레이션이 제 1 정규형이며 부분 함수의 종속성을 제거한 형태를 말한다.

부분 함수의 종속성 제거란 기본키가 아닌 모든 속성이 기본키에 완전 함수 종속적인 것을 말한다.

유저번호	유저ID	수강명	성취도
1	홍철	C++코딩테스트	90%
1	홍철	프런트특강	10%
2	범석	코드포스특강	7%
2	범석	DS특강	8%

유저번호	유저ID
1	홍철
2	범석

유저ID	수강명	성취도
홍철	C++코딩테스트	90%
홍철	프런트특강	10%
범석	코드포스특강	7%
범석	DS특강	8%

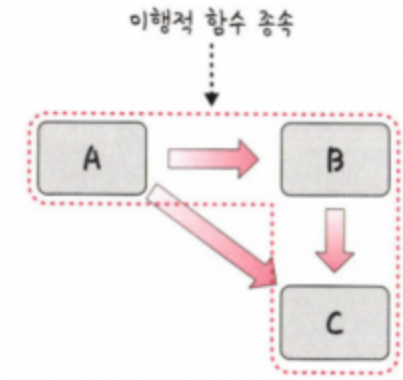
앞의 그림을 보면 기본키인 {유저ID, 수강명}과 완전 종속된 유저번호 릴레이션과 '{유저ID, 수강명}에 따른 성취도' 릴레이션으로 분리된 것을 볼 수 있습니다.

이때 주의할 점은 릴레이션을 분해할 때 동등한 릴레이션으로 분해해야 하고, 정보 손실이 발생하지 않는 무손실 분해로 분해되어야 한다는 것입니다.

제 3 정규형

제 2 정규형이고 기본키가 아닌 모든 속성이 이행적 함수 종속(transitive FD)을 만족하지 않는 상태를 말한다

이행적 함수 종속
이행적 함수 종속이란 A → B와 B → C가 존재하면 논리적으로 A → C가 성립하는데, 이때 집합 C가 집합 A에 이행적으로 함수 종속이 되었다고 한다



예를 들어 무무쇼핑몰이 있다고 해봅시다. 유저ID와 등급, 할인율이 정해져 있는 테이블을 다음과 같이 분해하는 것을 말합니다.

▼ 그림 4-22 제3정규형

유저ID	등급	할인율
홍철	플래티넘	30%
범수	다이아	50%
가영	마스터	70%

유저ID	등급
홍철	플래티넘
범수	다이아
가영	마스터

등급	할인율
플래티넘	30%
다이아	50%
마스터	70%

보이스/코드 정규형

보이스/코드 정규형(BCNF)은 제 3 정규형이고, 결정자가 후보키가 아닌 함수 종속 관계를 제거하여 릴레이션의 함수 종속 관계에서 모든 결정자가 후보키인 상태를 말한다.

- 결정자
  함수 종속 관계에서 특정 종속자를 결정짓는 요서 X→Y일 때 X는 결정자 Y는 종속자

요구 사항은 다음과 같다고 해봅시다.

- 각 수강명에 대해 한 학생은 오직 한 강사의 강의만 수강한다.
- 각 강사는 한 수강명만 담당한다.
- 한 수강명은 여러 강사가 담당할 수 있다.

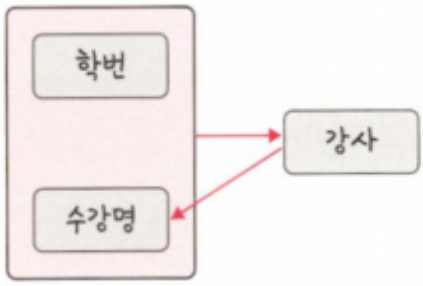
▼ 그림 4-23 학번-수강명-강사 릴레이션

학번	수강명	강사
12010	코딩테스트	큰돌
12010	MEVN	재엽
12011	코딩테스트	큰돌
12011	MEVN	가영
NULL	롤	범석

삽입 이상

앞의 릴레이션을 보면 {학번, 수강명} 또는 {학번, 강사}가 후보키가 되며, 만약 범석이라는 강사가 '롤'이라는 수강명을 담당한다고 했을 때 이를 삽입하면 학번이 NULL이 되는 문제점이 발생합니다. 또한, 이 릴레이션은 다음과 같은 함수 종속 다이어그램을 가집니다.

▼ 그림 4-24 학번-강사-수강명 함수 종속



즉, 강사 속성이 결정자이지만 후보키가 아니므로 이 강사 속성을 분리해야 합니다.

▼ 그림 4-25 보이스/코드 정규형을 만족한 릴레이션

학번	수강명	강사
12010	코딩테스트	큰돌
12010	MEVN	재엽
12011	코딩테스트	큰돌
12011	MEVN	가영
NULL	롤	범석

삽입 이상

학번	강사
12010	큰돌
12010	재엽
12011	큰돌
12011	가영

수강명	강사
코딩테스트	큰돌
MEVN	재엽
MEVN	가영
롤	범석

앞의 그림처럼 롤-범석이 제대로 들어갔으며 학번-강사/수강명-강사로 잘 분해된 모습을 볼 수 있죠?

참고로 이렇게 정규형 과정을 거쳐 테이블을 나눈다고 해서 성능이 100% 좋아지는 것은 아닙니다. 성능이 좋아질 수도 나빠질 수도 있습니다. 테이블을 나누게 되면 어떠한 쿼리는 조인을 해야 하는 경우도 발생해서 오히려 느려질 수도 있기 때문에 서비스에 따라 정규화 또는 비정규화 과정을 진행해야 합니다.