[IT Essential] Project DB 설계 - 프로젝트를 위한 RDBMS 설계방법

> 진행자: 최호근 컨설턴트님
>
> 날짜: 2021-01-20

목차

1. DB 설계의 목적

> 설계를 대충하면 기능 한 개 추가될 때마다 DB와 관련된 이미 개발된 프로그램도 함께 뜯어고쳐야 하는 경우가 발생한다!

- 프로젝트, 명세서 등의 정보 요구사항에 대한 정확한 이해
- 분석자, 개발자, 사용자 간의 원활한 의사소통 수단
- 데이터 중심의 분석 방법
- 현행 시스템만이 아닌 신규 시스템 개발의 기초 제공

2. 설계를 위한 요구사항 분석

- 데이터베이스에 대한 사용자의 요구사항을 수집하고 분석해서 아래와 같이 요구사항(기능) 명세서를 작성한다.
- 예) 항공사 DB

> <요구사항 명세 샘플 - 항공사 DB>
>
> 회원으로 가입하려면 아이디, 비밀번호, 성명, 신용카드 정보를 입력해야 한다
> 회원의 신용카드 정보는 여러 개를 저장할 수 있다 .
> 신용카드번호, 유효기간을 저장할 수 있다.
> 회사가 보유한 비행기에 대해 비행기 번호, 출발 날짜, 출발 시간 정보를 저장하고 있다.
> 비행기 좌석에 대한 좌석 번호, 등급 정보를 저장하고 있다.
> 회원은 좌석을 예약하는데, 회원 한 명은 좌석을 하나만 예약할 수 있고
> , 한 좌석은 회원 한 명만 예약할 수 있다.

3. 개념적 설계

- 작성한 요구사항 명세서에서 데이터베이스를 구성하는데 필요한 개체, 속성, 개체 간의 관계를 추출하여 **ERD(Entity Relationship Diagram)**를 생성한다.

> 1. 개체(Entity)와 속성(Attribute)을 추출한다.
>
>   - 대부분 명사로 선별한다.
> 2. 개체 간의 관계(Relationship)를 추출한다.
>
>   대부분 동사로 선별한다. (개체간의 관계를 나타내는 동사이여야 한다.)
>
>   관계에 속한 속성도 있을 수 있다.
>
>   1:1, 1:N, N:M
>
>   필수적인 참여, 선택적인 참여

- 참고자료
  - ERD란 무엇인가?

3-1. 개체(Entity)와 속성(Attribute) 추출

- 요구사항에서 **개체(Entity)**는 대부분 명사로 이루어져 있지만, **속성(Attribute)**과 구별하여 추출한다.
- 아래 예시에서 개체는 주황색, 속성은 파란색으로 표시되어있다.

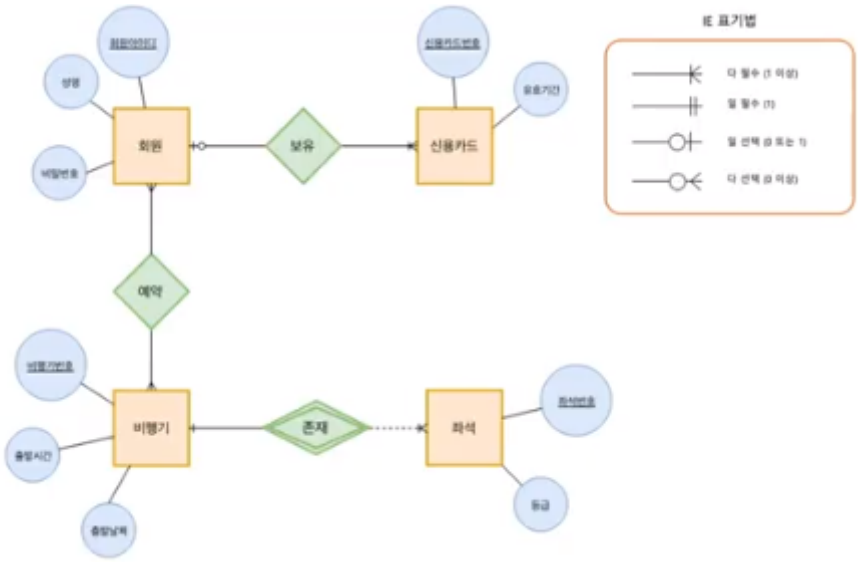
> <요구사항 명세 샘플>
>
> 회원으로 가입하려면 아이디, 비밀번호, 성명, 신용카드 정보를 입력해야 한다
> 회원의 신용카드 정보는 여러 개를 저장할 수 있다 .
> 신용카드번호, 유효기간을 저장할 수 있다.
> 회사가 보유한 비행기에 대해 비행기 번호, 출발 날짜, 출발 시간 정보를 저장하고 있다.
> 비행기 좌석에 대한 좌석 번호, 등급 정보를 저장하고 있다.
> 회원은 좌석을 예약하는데, 회원 한 명은 좌석을 하나만 예약할 수 있고
> , 한 좌석은 회원 한 명만 예약할 수 있다.

3-2. 개체 간의 관계(Relationship) 추출

- **개체 간의 관계(Relationship)**는 여러가지로 분류해서 정의된다.
- 아래 예시에서 관계는 초록색으로 표시되어있다.

> <요구사항 명세 샘플>
>
> 회원으로 가입하려면 아이디, 비밀번호, 성명, 신용카드 정보를 입력해야 한다.
> 회원의 신용카드 정보는 여러 개를 저장 할 수 있다 .
> 신용카드번호, 유효기간을 저장할 수 있다.
> 회사가 보유한 비행기에 대해 비행기 번호, 출발 날짜, 출발 시간 정보를 저장하고 있다.
> 비행기 좌석에 대한 좌석 번호, 등급 정보를 저장하고 있다.
> 회원은 좌석을 예약하는데, 회원 한 명은 좌석을 하나만 예약할 수 있고,
> 한 좌석은 회원 한 명만 예약할 수 있다.

3-3. 개념 설계 기반으로 ERD 생성



- 주황색은 개체, 파란색은 속성, 초록색은 관계를 나타낸다.
- **IE 표기법**을 사용해서 관계를 표기한다.
- 참고자료
  - IE 표기법/까마귀발 표기법

4. 논리적 설계

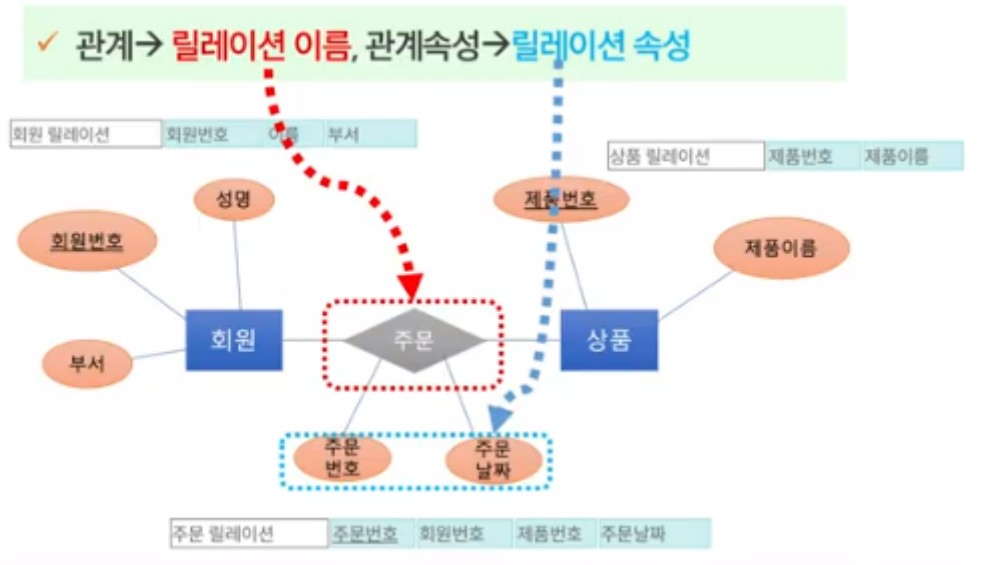
- 모든 개체는 릴레이션(Table)으로 변환
- N:M 관계는 릴레이션(Table)으로 변환
- 1:N 관계는 외래키로 표현
- 1:1 관계는 외래키로 표현
- 다중값 속성은 독립 릴레이션(Table)으로 변환

4-1. 모든 개체는 릴레이션(Table)으로 변환

- ER 다이어그램의 각 개체를 **릴레이션(Table)**으로 변환한다.
- 개체는 테이블, 속성은 테이블의 속성으로 변환한다.

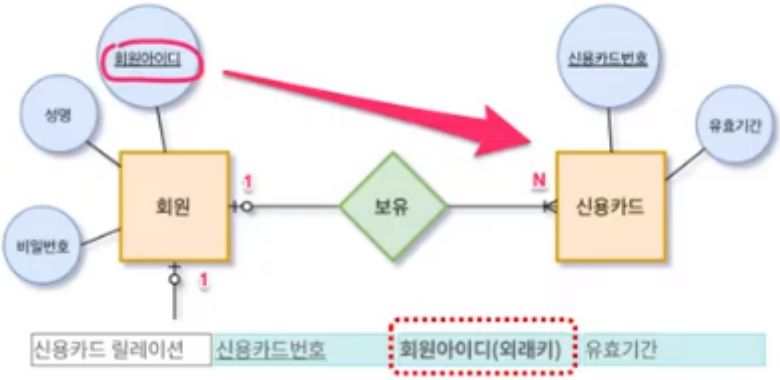
4-2. N:M 관계는 릴레이션(Table)으로 변환

- **N:M 관계(다대다 관계)**에서 관계는 릴레이션 이름, 관계 속성은 릴레이션 속성으로 변환한다.



4-3. 1:N 관계는 외래키(FK)로 표현

- 일반적으로 **1:N 관계(일대다 관계)**에서 1측 개체의 기본키를 N측 릴레이션에 포함시키고 **외래키(FK, Foreign Key)**로 지정한다.

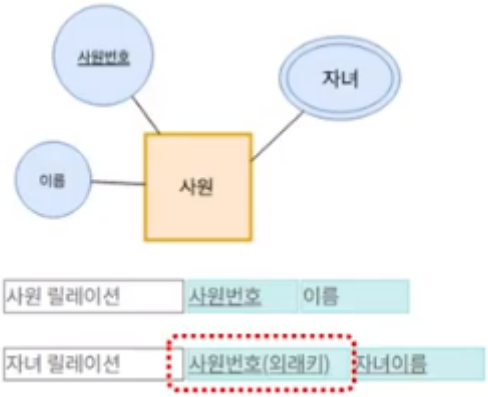


4-4. 1:1 관계는 외래키(FK)로 표현

- 일반적 **1:1 관계(일대일 관계)**는 외래키(FK)를 주고 받는다.

4-5. 다중값 속성은 독립 릴레이션(Table)으로

- 릴레이션에서는 **다중 값 속성**을 가질 수 없으므로 다중 값 속성은 별도의 릴레이션으로 생성해야 한다.
- 예) 한 사원이 자녀를 5명 두고 있다고 할 때, 자녀 정보를 담기 위해 사원 릴레이션(table)에 자녀 컬럼을 다섯개 추가 할 수는 없으니까 별도의 자녀 릴레이션을 생성하고 외래키로 참조한다.



5. 물리적 스키마 및 구현

- ERD를 실제 테이블로 생성한다. (Workbench 같은 DB Tool이나 SQL스크립트 사용으로도 가능해야 한다.)

- 참고자료
  - ERD 툴 종류

6. 반정규화

- 반정규화(역정규화)는 정규화 된 엔티티타입, 속성, 관계를 **시스템의 성능향상, 개발과 운영의 단순화를 위해** 모델을 통합하는 프로세스를 말한다.
- 정규화 모델 vs 반정규화 모델

<정규화 모델>
이상적인 논리모델은 모든 엔티티타입,속성,관계가 반드시 한 개만 존재하며 따라서 입력,수정,삭제도 한군데에서만 발생하므로 데이타 값이 변질되거나 이질화 될 가능성이 없다. 반면 여러 테이블이 생성되어야 하므로 SQL작성이 용이하지 않고 과다한 테이블 조인이 발생하여 성능이 저하될 가능성이 높다.

<반정규화 모델>
반대로 반정규화를 하면 여러 개의 테이블이 단순해지므로 SQL작성이 용이하고 성능이 향상될 가능성이 많다. 그러나 같은 데이터가 여러 테이블에 걸쳐 존재하므로 무결성이 깨질 우려가 있다.

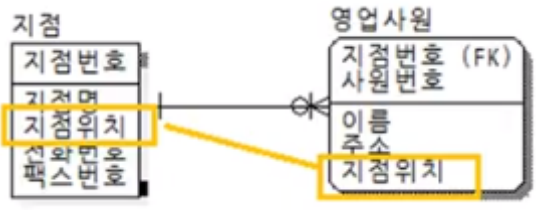
6-1. 테이블 반정규화 방법

- 1:1 관계의 테이블 병합
- 1:N 관계의 테이블 병합
- 수퍼/서브 타입 테이블 병합
- 수직 분할(집중화 된 일부 컬럼을 분리)
- 수평 분할(행으로 구분하여 구간별 분리)
- 테이블 추가 (중복테이블, 통계테이블, 이력케이블, 부분테이블)

- 참고자료
  - 반정규화 (비디오)

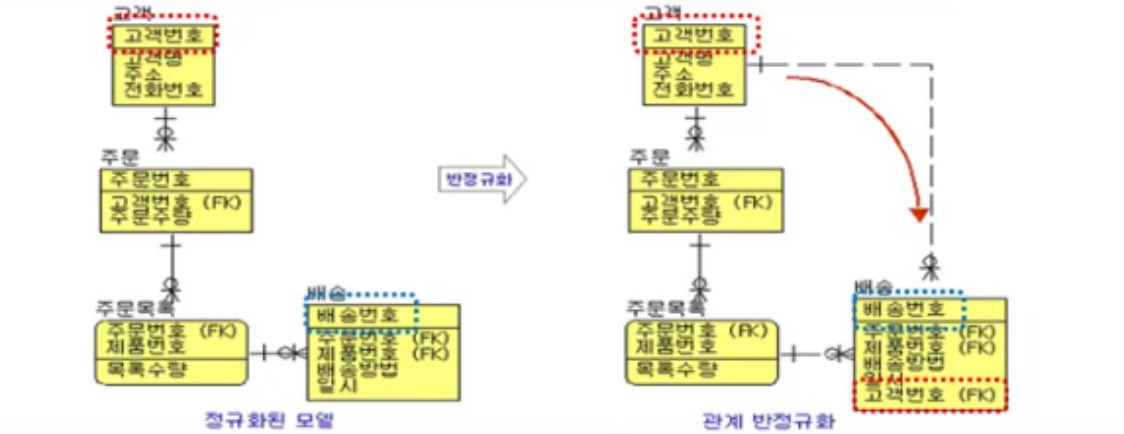
6-2. 대표적 반정규화 - 컬럼 반정규화

- 중복컬럼 추가 (자주 조회하는 컬럼이 있는 경우)
- 파생 컬럼 추가 (미리 계산한 값)
- PK에 의한 컬럼 추가
- 응용시스템 오작동을 위한 컬럼 추가 (이전 데이터 임시 보관)



6-3. 대표적 반정규화 - 관계 반정규화

- 중복 관계 추가 (이미 A테이블에서 C테이블의 정보를 읽을 수 있는 관계가 있음에도 관계를 중복하여 조회(Read) 경로를 단축)
- 아래의 예시에서 왼쪽 같은 경우는 고객번호를 가지고 배송번호를 조회하려면 여러 번의 조인이 일어나야하지만, 오른쪽과 같이 반정규화를 통해 배송 테이블에 고객번호를 컬럼으로 넣으면 조회 경로를 단축 시킬 수 있다.



7. 두 줄 요약

- 정규화가 좋은가 반정규화가 좋은가?
  - 정답은 정해져 있지 않고 **그때그때 다르다.** 개념을 잘 알고 있어야 적절하게 쓸 수 있다.

8. Q&A

1. 정규화에도 단계가 있나요?
   - 제1 정규형부터 제5 정규형까지 있으며 정규화를 할 때 단계별로 순차적으로 진행합니다.
   - 참고자료
     - 정규화 (Normalization) 단계

2. 테이블을 설계 할 때 컬럼명을 테이블명을 구분할 수 있도록 지으면 반정규화가 어려워지나요?
   - 명칭은 상관이 없으나, 구분이 가능하도록 명칭을 명확하게 지으면 좋습니다.

3. 테이블을 만들 때 쓰는 SQL 스크립트는 다 외우고 있어야 하나요?
   - DB 생성, 테이블 생성, 컬럼 추가, 테이블 변경 등 기본적인 CRUD 정도는 알고 있으면 좋습니다.
   - 참고자료
     - SQL 쿼리 연습

4. RDBMS와 NoSQL 설계는 많이 다른가요?

- NoSQL은 key와 value 형태로 되어있어 정규화를 위반하는 경우도 많아 RDBMS와 NoSQL의 설계 방법은 많이 다릅니다. 저장할 값의 종류와 호출 방법에 따라 RDBMS와 NoSQL 중 어떤 것을 사용하는 것이 좋을지 고민 후 선택해야 합니다.
- 참고자료
  - SQL vs NoSQL

5. 외부키 유무가 데이터베이스 효율에 큰 영향을 끼치나요?
   - 효율은 어떤 형태로 쿼리가 이루어지는지, 외부키가 얼마나 자주 사용되는지와 관련이 있기 때문에 단순히 외부키의 존재 유무만으로는 효율성을 평가하기는 어렵습니다.

6. 언제 NoSQL을 사용하고 언제 RDBMS를 사용하는지 좀 더 구체적으로 알고싶습니다.
   - CRUD가 많이 일어날 때는 RDBMS를 많이 사용하고, 단순한 입력과 조회가 많이 일어날 때는 NoSQL을 많이 사용합니다.

7. 어떻게 설계된 DB를 볼 때 매력적이라고 느끼시나요?
   - 개발자들이 개발하기 편한 DB가 매력적인 DB라고 생각합니다.

8. DB 성능을 고려해야 할 데이터 개수의 기준이 있을까요?
   - 요즘에는 데이터베이스의 성능이 많이 좋아져서 설계만 잘하면 억 단위의 데이터도 문제 없이 다룰 수 있습니다.