



데이터 모델링

3장. 논리 데이터 모델링

1절. 논리 데이터 모델링 이해

2절. 속성 정의

3절. 엔티티 상세화

4절. 이력관리 정의

1-1 논리 데이터 모델링 정의

- ❖ 비즈니스 정보의 구조와 규칙을 명확히 표현하는 기법
- ❖ 데이터 모델링이 최종적으로 완료된 상태
- ❖ 물리적 스키마 설계를 하기 전 단계의 데이터 모델 상태
- ❖ 전산화와는 독립적으로 비즈니스 데이터에 존재하는 사실을 인식·기록하는 기법
- ❖ 데이터 모델링 과정에서 가장 핵심이 되는 부분

1-2 논리 데이터 모델링 목적 및 효과

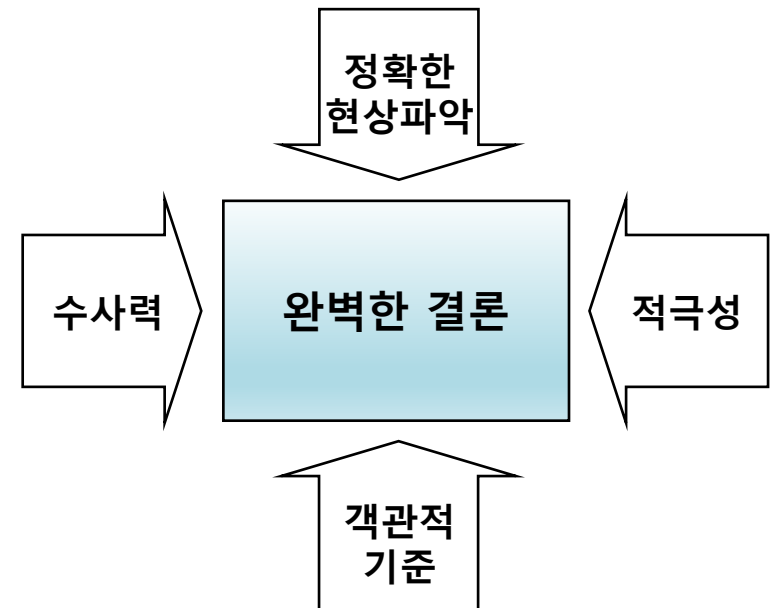
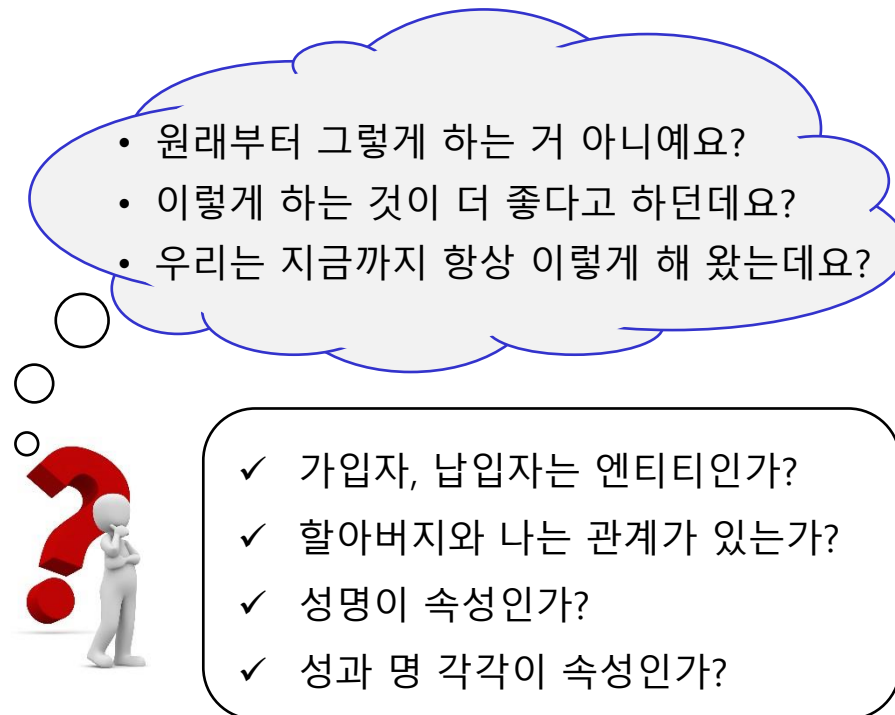
- ❖ 해당 비즈니스에 대한 데이터 관점에서의 명확한 이해
- ❖ 전사적인 통합 데이터 체계 확립
- ❖ 데이터의 일관성 및 정확성 유지를 위한 규칙 도출
- ❖ 안정적인 데이터베이스 설계의 토대 마련
- ❖ 사용자와의 명확한 의사소통을 위한 수단으로 활용

1-3 논리 데이터 모델링 필수 성공요소

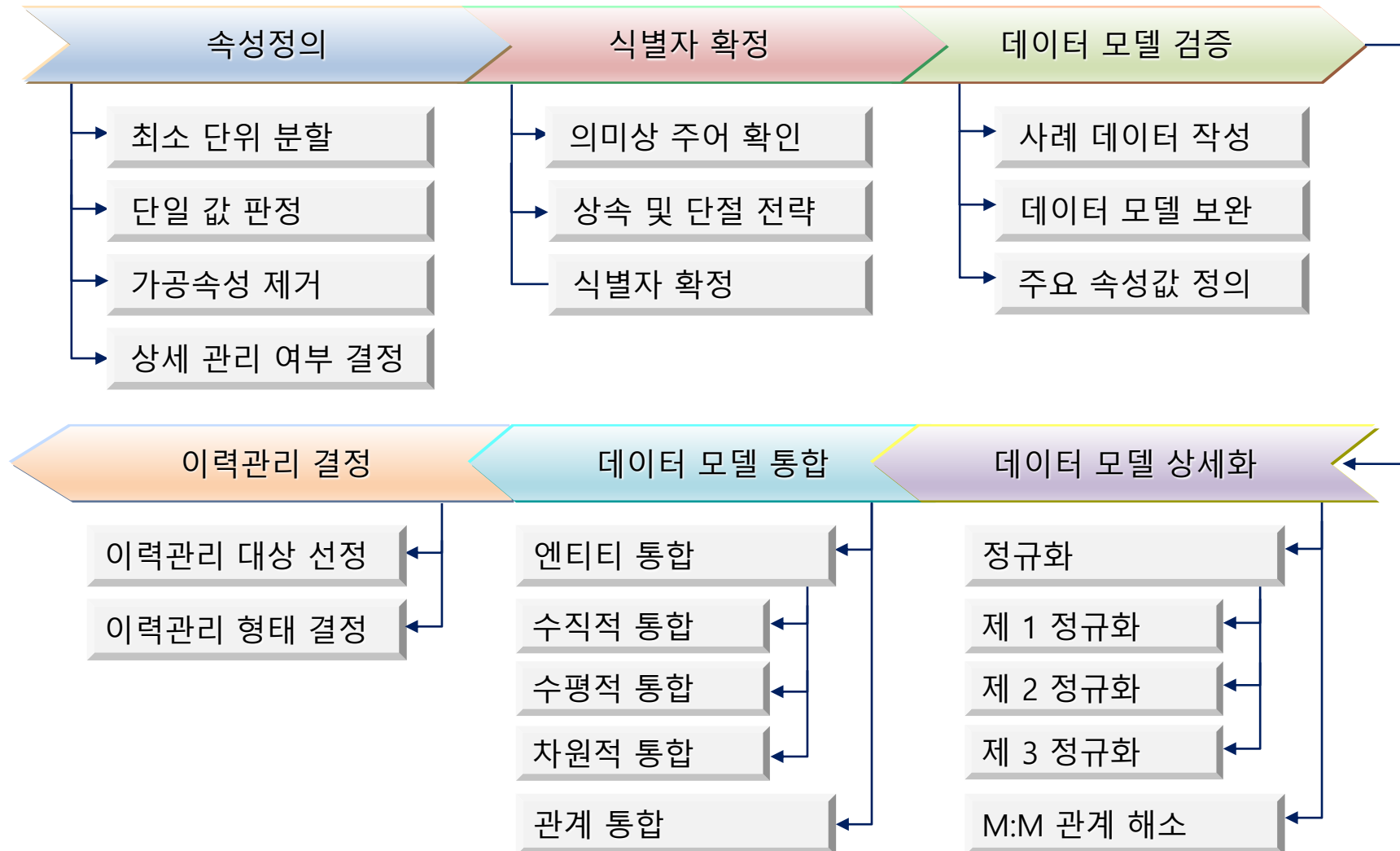
- ❖ 업무에 능통한 현업 사용자와 함께 데이터 모델링을 진행하라
 - ✓ 업무를 데이터 관점에서 체계화하는 작업
- ❖ 절차(Procedure)보다는 데이터에 초점을 두고 모델링을 진행하라
 - ✓ 데이터 모델에서 순서와 시간, 흐름의 개념이 들어가면, 많은 데이터의 중복과 정합성을 훼손
- ❖ 데이터의 구조(Structure)와 무결성(Integrity)을 함께 고려하라
- ❖ 개념화(Conceptualization)와 정규화(Normalization) 기법을 적용하라
- ❖ 가능하면 다이어그램(Diagram)을 이용하여 업무를 표현하라
- ❖ 데이터 모델링을 지원하는 데이터 사전을 구축하라

1-4 데이터 모델링의 객관화

- ❖ 데이터 모델링은 실수를 찾아주는 컴파일러가 없다.
- ❖ 개발 및 테스트 단계에 가서야 문제가 드러난다.
- ❖ 자신의 결정에 확신을 가지기 어렵다.
- ❖ 무엇을 모르는 지를 모르는 사람이 많이 있다.



1-5 논리 데이터 모델링 절차



2-1 속성 개념

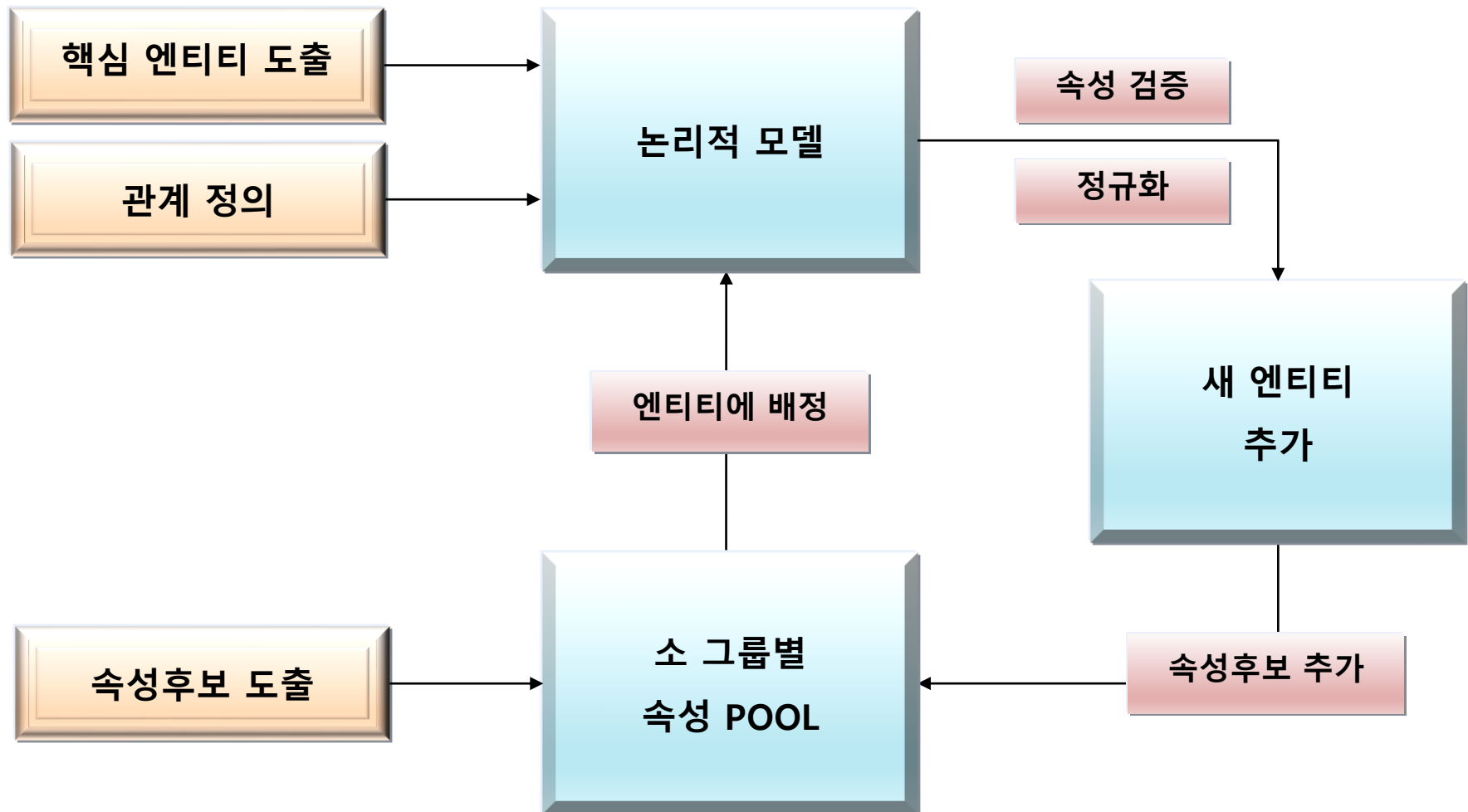
❖ 속성 정의

- ✓ 엔티티에서 관리되는 구체적인 정보 항목으로 더 이상 분리될 수 없는 최소의 데이터 보관 단위
- ✓ 속성의 어원적 의미
 - 가공되지 않은 원천적인 것
 - 고유한 성질로 남의 도움을 받지 않더라도 자기만이 가지고 있는 독자적인 성질이 있어야 함
- ✓ 독자적인 성질이라는 것은 상대적인 의미로서, 어떤 목적으로, 어떤 엔티티에서 사용되는지에 따라 판단이 달라질 수 있다.



- 원자 단위냐?
- 유일하게 존재하냐?
- 근원 값이냐?

2-2 속성 후보 도출 - 반복적인 작업



2-3 속성 검증 및 확정

❖ 1단계 : 최소 단위(Atomic Value)까지 분할하라

- ✓ 집합 개념의 속성은 단순 개념으로 분할한다.
- ✓ 가능한 최소 단위까지 분할한 후 관리 필요에 따라 통합한다.
- ✓ 일자, 시간, 성명, 주민등록번호 등은 일반적으로 분할하지 않는 것이 좋다.
- ✓ 주소와 같은 것은 그냥 두었다가 설계단계에서 필요 시 분할하기도 한다.
- ✓ 분할 및 통합의 기준은 업무의 요구 사항에 따른다.
- ✓ 분할 속성의 대표적 유형
 - 일자(日子) 형태의 속성
 - 전화번호 유형 : 전화, 팩스번호
 - 주소 유형 : 고객 주소

	통합된 속성	분리된 속성		
①	매출일자	매출년도	매출월	매출일
②	처리일시	처리일자	처리시간	
③	우편번호	시군구	읍면동	
④	전화번호	지역번호	국번	개별번호
⑤	전표번호	처리일자	일련번호	
⑥	주소	지역주소	상세주소	
⑦	발생일자	입고일자	출고일자	선적일자
		하역일자	입문일자	출문일자

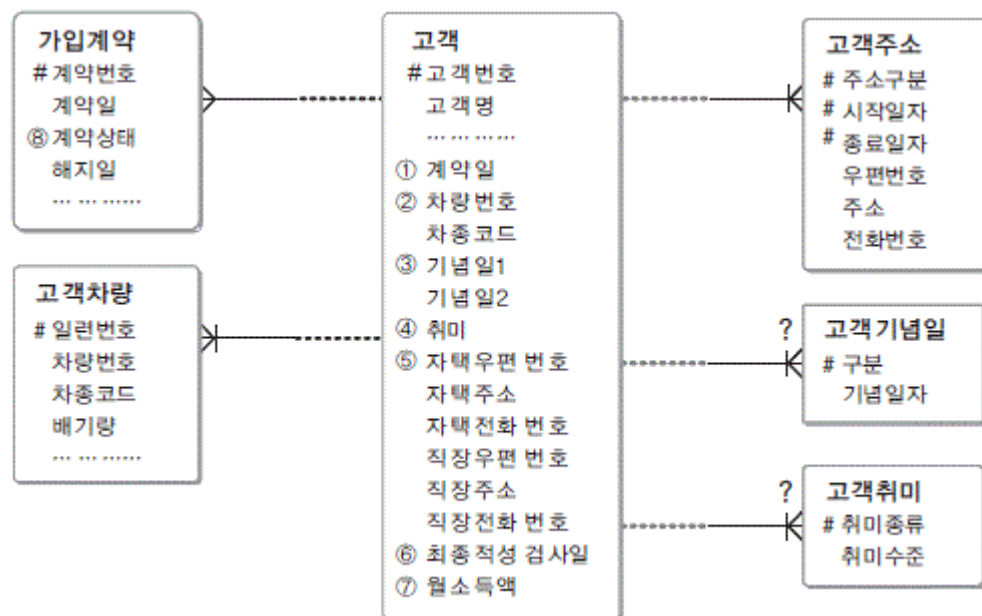
2-3 속성 검증 및 확정

❖ 2단계 : 하나의 값(Single Value)만을 가지는지 검증한다

- ✓ 여러 값을 가지거나 반복되는 속성은 잘못된 속성이다.
- ✓ 반복되는 속성은 새로운 엔티티로 분할해야 할 1차 정규화의 대상이 된다.
- ✓ 속성 의미에 대한 정의에 따라 하나일 수도, 여러 개 일수도 있다.
- ✓ 배타적 관계는 가능한 하나의 속성으로 통합할 것

✓ 대표적 유형

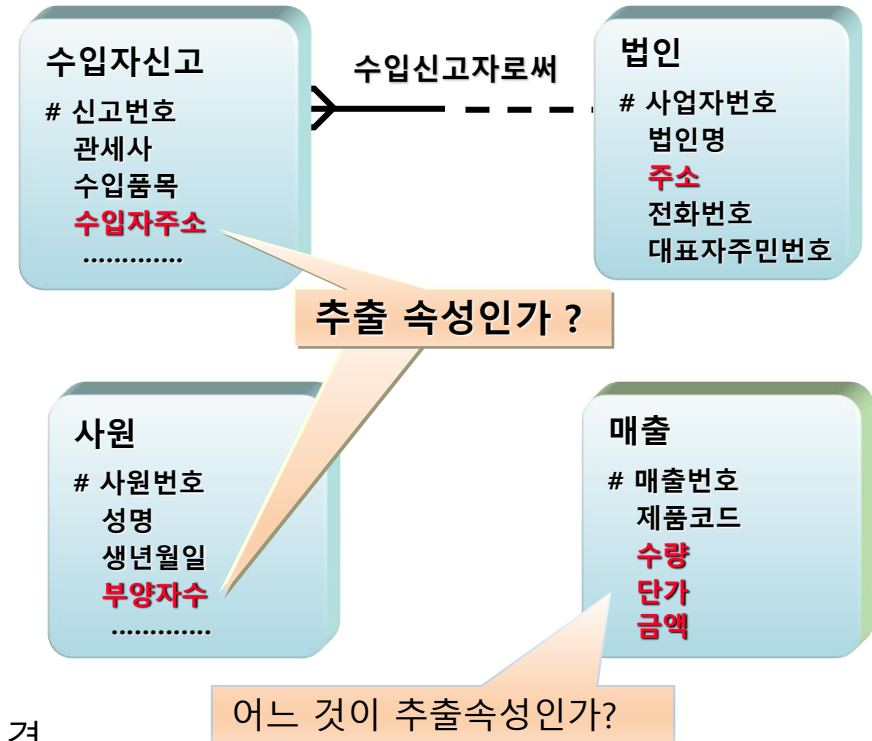
- 계약일
- 차량번호
- 취미



2-3 속성 검증 및 확정

❖ 3단계 : 추출 속성(Derived Attribute)인지 검증한다

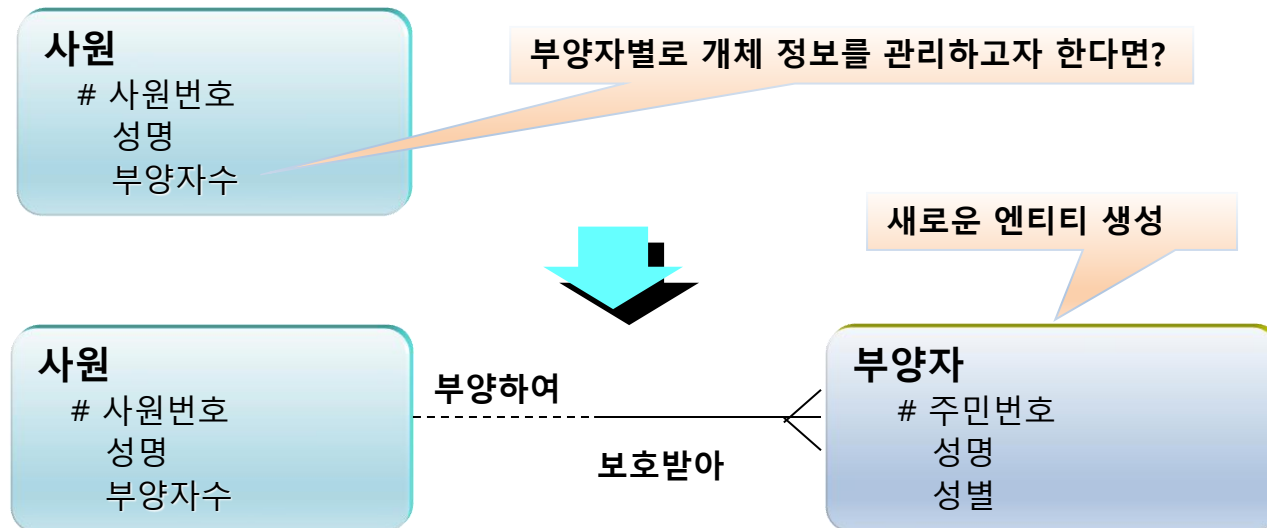
- ✓ 다른 속성에 의해 가공되어서 만들어진 값인지 확인
- ✓ 추출 값이란 원천적인 값을 가지고 언제라도 쉽게 재현할 수 있는 속성
- ✓ 대표적 유형
 - 개수(Count) : 특정범위의 개수, ...
 - 합계(Total) : 특정 기간의 총 매출액, ...
 - 최대/최소/평균(Max/Min/Avg) : 통계정보
 - 기타 계산 : 급여의 10%, 금액(단가*수량)
- ✓ 고려사항
 - 이력관리에 따라 판단이 달라질 수도 있음
 - E-R 모델 내에는 추출값을 포함시키지 말 것
 - 추후 물리 데이터 모델링 시에 검토하라
 - 추출값은 낭비(Redundancy)
 - 추출값은 데이터의 일관성을 저해
 - 추출값의 기본이 되는 속성이 변경되면 같이 변경
 - 꼭 필요한 컬럼은 별도로 정리해 두거나 특별한 표시를 해 둘 것



2-3 속성 검증 및 확정

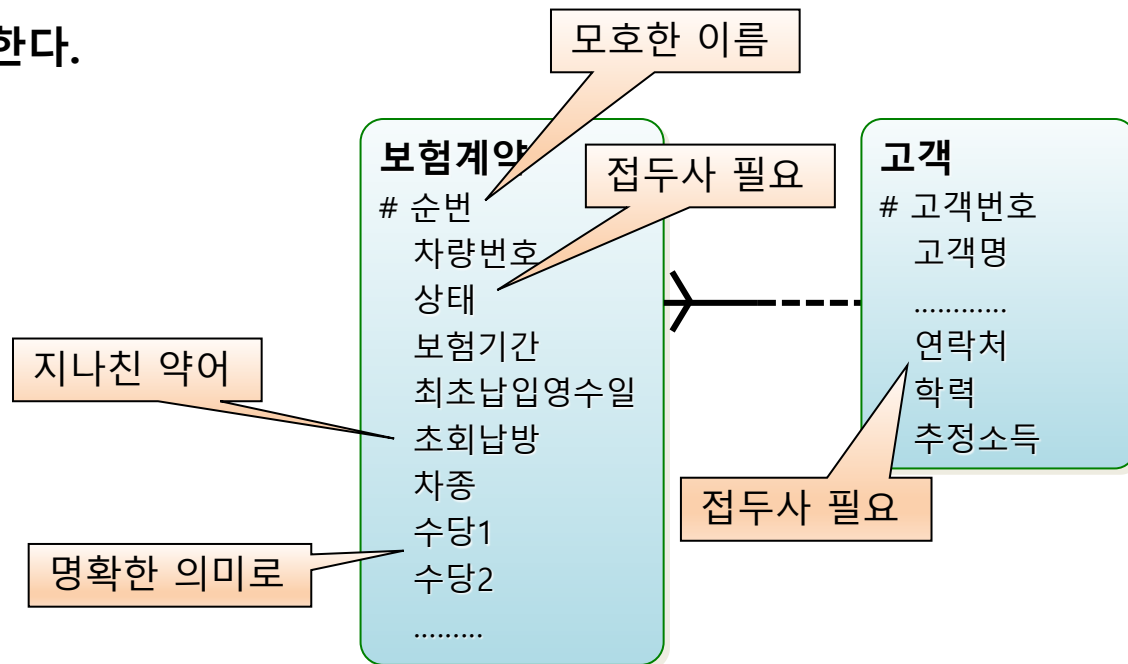
❖ 4단계 : 보다 상세하게 관리할 것이냐?

- ✓ 속성이 자기 소유의 속성을 가지면 엔티티
- ✓ 현재에 만족하지 말고 변화가 예상되는 미래의 관리수준을 감안해라
- ✓ 이 부분을 간과하면 개발 및 테스트, 운영 시에 많은 보완이 발생
- ✓ 모델러의 도덕성, 적극성, 인간미의 문제
- ✓ 한번 더 깊이 생각하지 않으면 우리 눈에 보이지 않는다.
- ✓ 모델링 시에 좀더 깊이 파헤친 것이 시간 및 비용 절약의 원천



2-4 속성 정의 시 유의사항

- ❖ 의미가 명확한 속성 명칭을 부여한다.
- ❖ 유일한 복합명사 사용
- ❖ 단수형으로 속성명을 사용한다
- ❖ 표준 단어 제정
 - ✓ 표준 용어 사전에 사전에 생성
- ❖ 작의적인 전용 금지



※ 전용 발생사유

- 모델러가 속성의 의미를 매우 추상적이고, 모호하게 정의했기 때문에 개발자들이 각기 다르게 이해
- 개발 막바지 또는 이미 사용중인 시스템에서 사용자 요구사항으로 인해 새로운 속성 추가
- ERP 패키지에서 자주 나타나는 형태로서 미리 전용의 용도로 속성을 정의해 두는 경우

실습1. 골프장 관리 모델

❖ 업무처리규정

우리는 골프장 정보가 중요하기 때문에 별도로 관리하려고 한다. 각종 연락처 정보와 약도를 관리하고, 연락처는 부킹용 전화나 팩스를 관리할 필요가 있다. 좀더 자세한 정보를 원하는 회원들 위해 홈페이지주소도 관리한다.

골프장 요약정보로는 홀 규모, 연면적, 대표자, 개장일, 잔디 종류, 연습장유무 등이 있다. 골프장은 정회원제, 정회원퍼블릭, 비정회원퍼블릭으로 구분할 수 있으며 최상급, 상급, 보통으로 골프장 수준을 구분하기도 한다.

우리는 서울, 경기, 강원 등 여러 개의 지역으로 나누어 골프장 위치를 관리하고자 한다.

3-1 식별자(UID, Unique Identifier) 확정

식별자는 엔티티 內 모든 인스턴스의 유일성을 보장하기 위해서 반드시 필요한 것

❖ 본질 식별자

✓ 키 엔티티의 본질 식별자

- 키 엔티티는 부모가 없이 창조된 집합이므로 식별자 또한 창조시켜 주어야 함

✓ 절대 종속 / 상대 종속 의미

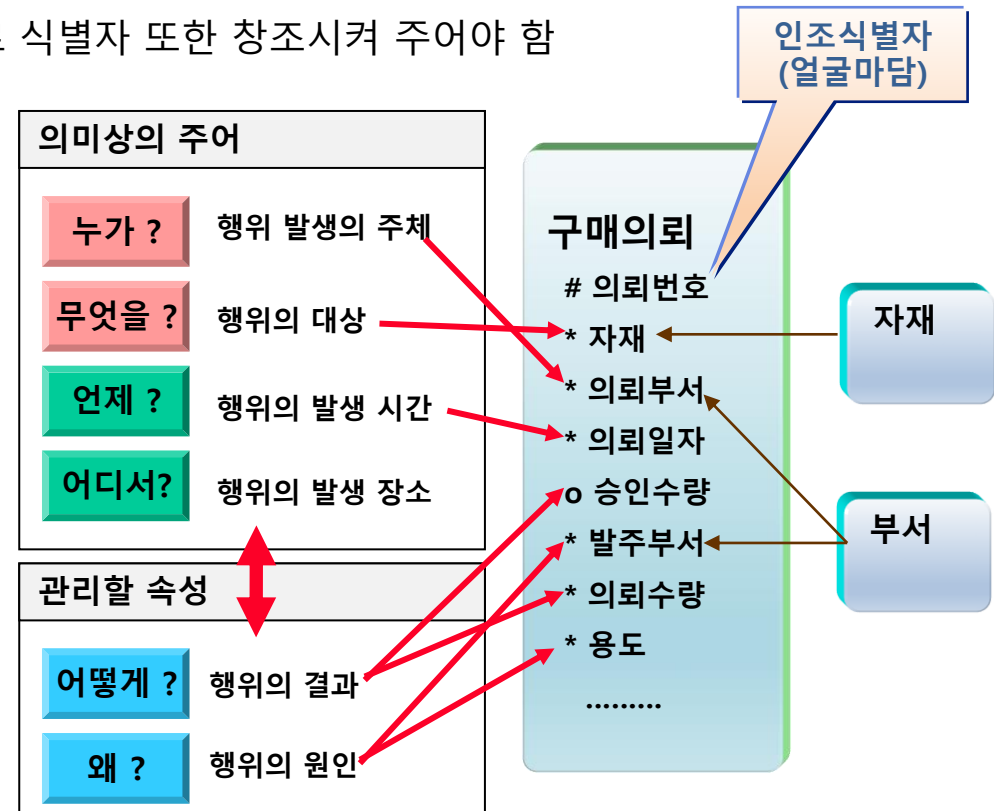
- 나를 태어나게 하는데 절대적인 영향을 주었는지 확인하는 것

✓ 직접 종속 / 간접 종속 의미

- 부모 엔티티와의 관계가 1촌이면 직접 종속이고 1촌 이상이면 간접 종속

✓ 행위 엔티티의 본질 식별자

- 절대종속이면서도 직접종속인 것, 자신을 태어나게 한 부모를 찾는 것



3-1 식별자(UID, Unique Identifier) 확정

❖ 식별자 확정 - UID BAR의 두 가지 의미

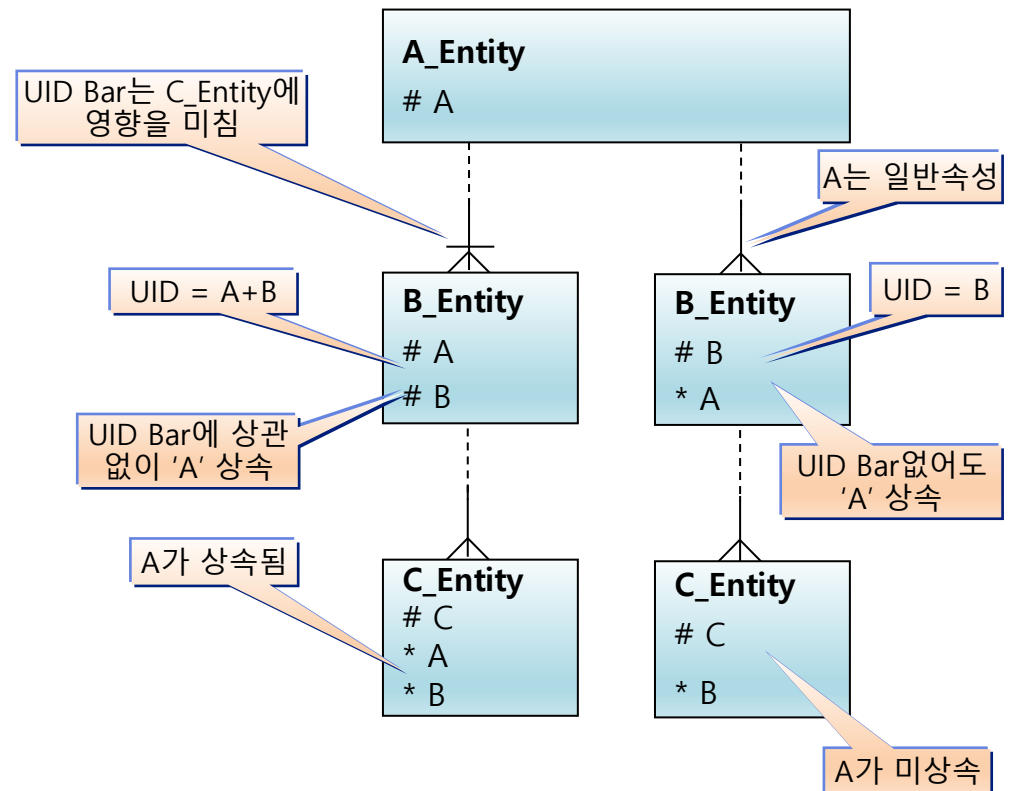
식별자는 자기 엔티티를 위해 생성하는 것처럼 보이나, 나를 참조할 다른 엔티티가 원하는 형태로 결정되어야 하는 것이기 때문에 주변 엔티티의 상황을 종합적으로 살피는 것이 중요

✓ 식별자로서의 역할

엔티티 자신의 입장에서 보면, 자신의 개체들을 다른 것들과 구별될 수 있도록 유일한 값을 만드는데 일조를 한다는 의미

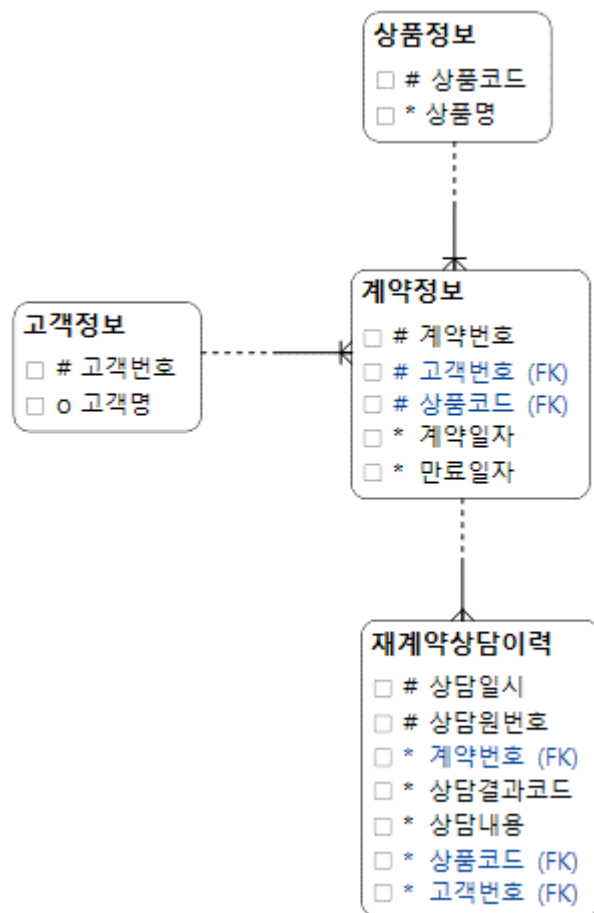
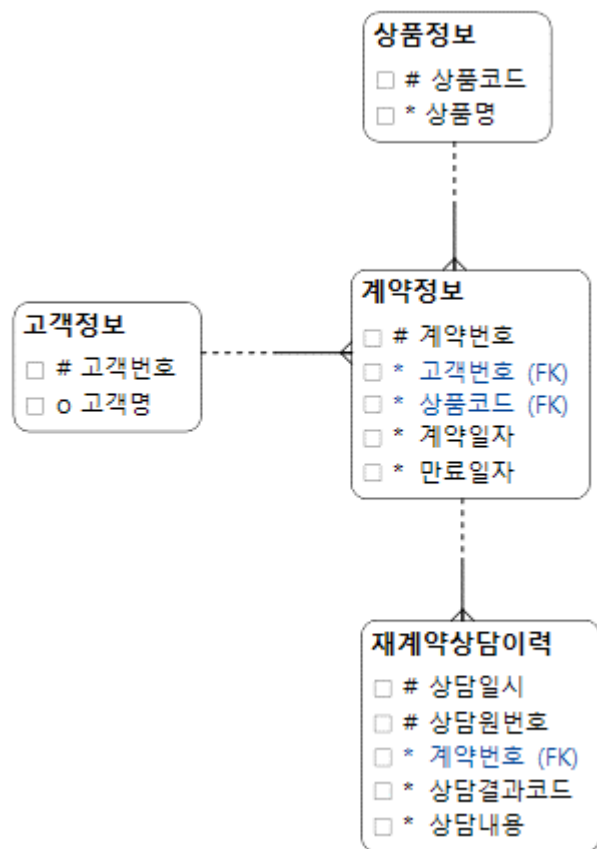
✓ 정보로서의 역할

참조하는 엔티티 입장에서 보면, 상대방 식별자를 상속 받아 자신이 보유할 정보가 증가했다는 의미



실습2. UID BAR

❖ 상속과 단절의 원리



3-1 식별자(UID, Unique Identifier) 확정

❖ 식별자 확정 - 절차

하향식(Top-down) 방식, 즉 상위 엔티티부터 시작해서 하위 엔티티로 순차적으로 결정해 가는 것이 좋다. 식별자 상속이란 상위에서 하위로 이루어지기 때문이다

✓ 키 엔티티 식별자 확정

- 부모를 가지지 않는 최상위 엔티티이므로 독립적으로 식별자를 확정할 수 있고, 식별자 확정 단계에서 주변의 상황과 여건을 감안하여 조정

✓ 메인 엔티티 식별자 확정

- 해당 업무의 근본이 되는 엔티티이므로 하위에 거느리고 있는 수많은 엔티티의 상황을 종합적으로 감안하여 결정한다. 식별자 속성의 개수를 적게하는 것도 중요하므로 인조 식별자를 생성하기도 한다.

✓ 하위 엔티티 식별자 확정

- 인조 속성을 많이 사용하지 않는 것이 바람직하다. 정보로서의 가치가 현저하게 감소하기 때문이다.

실습3. 구매 발주 모델

❖ 업무처리규정

각 부서는 필요한 품목들에 대해 구매 발주를 한다.

한 건의 발주는 한 거래처에 이루어지고, 발주일자와 납품요청일이 관리된다. 그리고 한 건의 발주에는 여러 품목들이 있으며 단가, 수량, 금액, 합계 등이 포함된다.

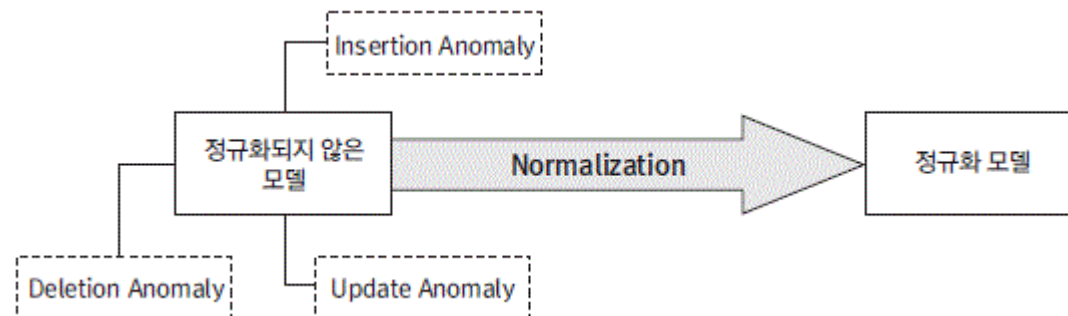
품목은 별도로 관리되며, 품목 관리 시에 가격비교를 위해 최근에 구매한 단가 정보도 관리한다.

3-2 정규화(Normalization)

논리 데이터 모델을 일관성이 있고 안정성 있는 자료구조로 만드는 단계

❖ 정규화의 의미

- ✓ 엔티티에 데이터를 삽입, 수정, 삭제할 때 오류가 발생할 개연성을 가지고 있으며, 이를 변경 이상 (Modification Anomaly)이라 하며 구체적으로 삽입 이상(Insertion Anomaly), 수정 이상(Update Anomaly), 삭제 이상 (Deletion Anomaly) 등이 있다.
- ✓ 변경 이상이 발생하는 데이터가 신뢰할 수 없는 값들로 채워지고 데이터의 일관성, 무결성을 해친다.
- ✓ 정규화 과정을 통해서 변경 이상의 엔티티를 정규화된 엔티티로 변환하게 된다.



3-2 정규화(Normalization)

❖ 정규화의 의미 – 입력 이상(insertion anomaly)

- ✓ 릴레이션에 새 데이터를 삽입하기 위해 원치 않는 불필요한 데이터도 함께 삽입해야 하는 문제

<u>고객아이디</u>	<u>이벤트번호</u>	당첨여부	고객이름	등급
apple	E001	Y	정소화	gold
apple	E005	N	정소화	gold
apple	E010	Y	정소화	gold
banana	E002	N	김선우	vip
banana	E005	Y	김선우	vip
carrot	E003	Y	고명석	gold
carrot	E007	Y	고명석	gold
orange	E004	N	김용욱	silver
melon	NULL	NULL	성원용	gold

← 삽입 불가!

3-2 정규화(Normalization)

❖ 정규화의 의미 – 삭제 이상(delete anomaly)

- ✓ 릴레이션에서 데이터를 삭제하면 꼭 필요한 데이터까지 함께 삭제하여 데이터가 손실되는 연쇄 삭제 현상

<u>고객아이디</u>	<u>이벤트번호</u>	당첨여부	고객이름	등급
apple	E001	Y	정소화	gold
apple	E005	N	정소화	gold
apple	E010	Y	정소화	gold
banana	E002	N	김선우	vip
banana	E005	Y	김선우	vip
carrot	E003	Y	고명석	gold
carrot	E007	Y	고명석	gold
orange	E004	N	김용욱	silver

← 데이터 손실 발생!

NAVER

3-2 정규화(Normalization)

❖ 정규화의 의미 – 갱신 이상(update anomaly)

- ✓ 릴레이션의 중복된 데이터들 중 일부만 수정하여 데이터가 불일치하게 되는 모순이 발생하는 것

고객아이디	이벤트번호	당첨여부	고객이름	등급
apple	E001	Y	정소화	vip
apple	E005	N	정소화	vip
apple	E010	Y	정소화	gold
banana	E002	N	김선우	vip
banana	E005	Y	김선우	vip
carrot	E003	Y	고명석	gold
carrot	E007	Y	고명석	gold
orange	E004	N	김용욱	silver

← 데이터 불일치 발생!

3-2 정규화(Normalization)

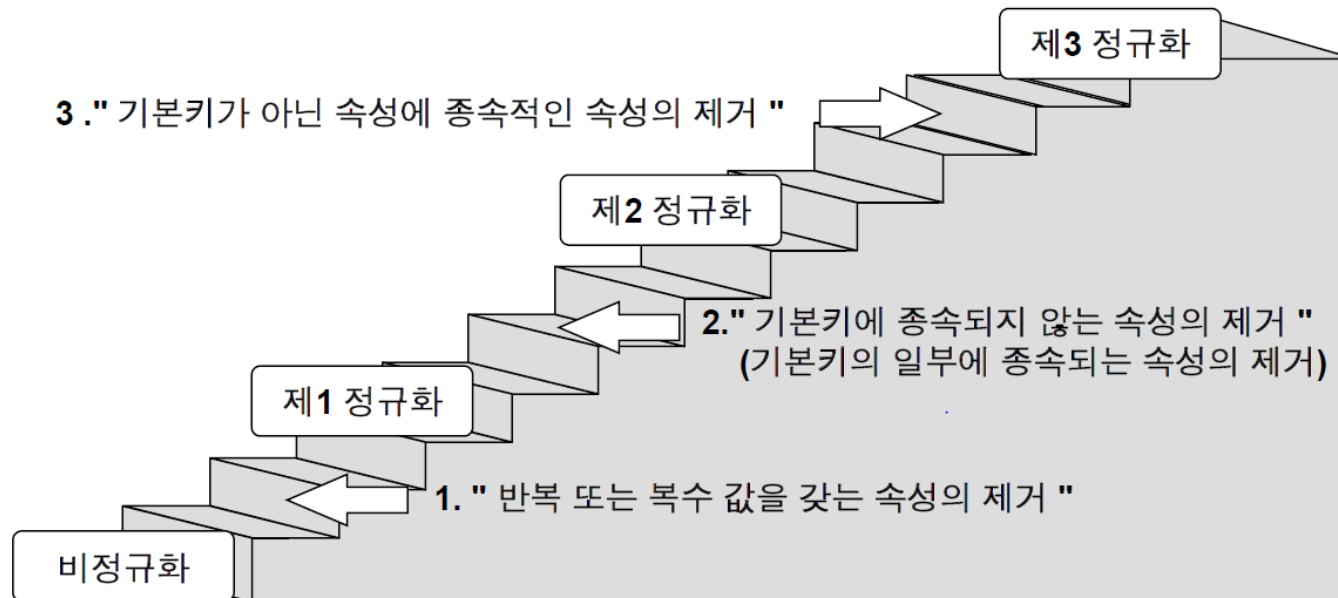
❖ 정규화의 장점

- ✓ **중복값이 줄어든다**
 - 정규화의 최대 성과는 칼럼 간, 레코드 간, 테이블 간에 중복되는 데이터들을 최소화하는 것
- ✓ **NULL 값이 줄어든다**
 - 전체적으로 NULL 값의 사용이 줄어들게 된다.
- ✓ **복잡한 코드로 데이터 모델을 보완할 필요가 없다**
 - 중복된 값이 적고, 그 부모가 누구인지가 명시되어 있다면, 데이터의 무결성을 지키기 위한 복잡한 변환 과정과 조인 질의, NULL 값 처리들이 필요가 없어진다.
- ✓ **새로운 요구 사항의 발견 과정을 돕는다**
 - 업무 담당자와의 협의를 통해서 현재뿐만 아니라 미래까지도 고려한 요구 사항의 발견 과정에서 많은 엔티티 혹은 속성들이 태어나게 된다.
- ✓ **업무 규칙의 정밀한 포착을 보증한다**
 - 체계화되고, 규칙의 Value화에도 도움을 주게 된다.
- ✓ **데이터 구조의 안정성을 최대화한다**
 - 중복된 값이 최소화되고 모든 정보들이 자기가 있어야 할 자리에 존재하게 되기 때문에 향후 발생 하게 될 모델 변화에도 유연하게 대처할 수 있다.

3-2 정규화(Normalization)

❖ 정규화 과정

- ✓ 정규화는 1차 정규화부터 BCNF(Boyce-Codd Normal Form)을 포함한 5차 정규화까지로 구성
- ✓ 일반적으로 중복이 최소로 발생하는 제 3 정규화까지 진행



3-2 정규화(Normalization)

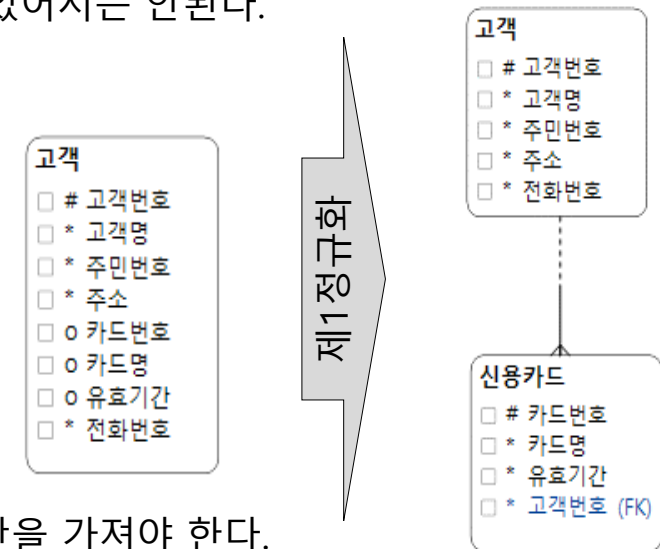
❖ 정규화 단계 - 1차 정규형 (1NF, First Normal Form)

✓ 정의

- 모든 속성은 반드시 하나의 값을 가져야 한다. 즉, 반복 형태가 있어서는 안된다.
- 각 속성의 모든 값은 동일한 형식이어야 한다.
- 각 속성들은 유일한 이름을 가져야 한다.
- 레코드들은 서로 간에 식별 가능해야 한다.

✓ 정규화 작업

- 고객 엔티티의 카드번호 속성의 값이 여러 건이 된다면 1차 정규형을 위배하는 것이 된다.
- 어떤 속성이 다수의 값을 가지고 있다면 M:1 관계의 새로 엔티티를 추가한다.
- 관계형 모델에서는 관계(Relation) 정의상 한 속성이 하나의 값만을 가져야 한다.
- 비정규형 관계가 관계로서의 모습을 갖추기 위해선 여러 개의 복합적인 의미를 가지고 있는 속성이 분해되어 하나의 의미만을 표현하는 속성들로 분해되어야 한다. 즉 속성수가 늘어나야 한다.
- 비정규형 관계가 관계로서의 모습을 갖추기 위해선 하나의 속성이 하나의 값을 가질 수 있어야 하며, 이 조건을 만족시키기 위해선 로우(Row)가 늘어나야 한다. 또는 다른 관계로 분리되어야 한다.
- 분석 또는 모델링 진행 과정에서 발생하며, 최종적인 모델링 완성 단계에서는 나올 수 없다. 그러나 분석(모델링) 초기 단계에는 상세히 분해된 속성보다는 위와 같은 레벨의 속성 추출이 복잡도를 줄일 수 있으므로 실전에서는 효율적이고 유리하게 이용될 수도 있다.



실습4. 제1 정규형

사원

- ☐ # 사원번호
- ☐ * 사원명
- ☐ * 주민번호
- ☐ * 주소
- ☐ * 자격증

3-2 정규화(Normalization)

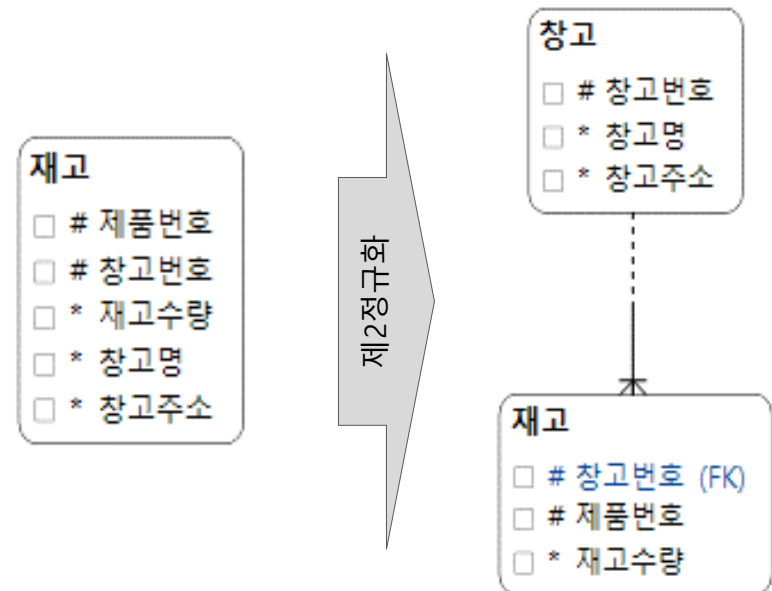
❖ 정규화 단계 - 2차 정규형 (2NF, Second Normal Form)

✓ 정의

- 식별자가 아닌 모든 속성들은 식별자 전체 속성에 완전 종속되어야 한다.
- 기본키가 아닌 모든 칼럼들이 기본키에 종속적이어야 2차 정규형을 만족할 수 있다.

✓ 정규화 작업

- 식별자가 제품번호+창고번호로 이루어진 재고 엔티티에서 창고번호 속성에 창고명, 창고주소 속성들이 종속적이기 때문에 이것은 2차 정규형을 위반하고 있는 것이다.
- 의미상의 주어 즉, 본질식별자를 알아야 식별자 부분 종속인지를 구분할 수 있다.
- 어떤 속성이 식별자 전체에 종속되어 있지 않으면 잘못된 위치이며 새로운 엔티티 즉, **상위 부모 엔티티를 생성하고 UID BAR를 상속받게 된다.**



실습5. 제2 정규형

가맹사업자

- ☐ # 가맹본부번호
- ☐ # 가맹지사번호
- ☐ * 공급마진
- ☐ * 가맹지사명
- ☐ * 가맹지사위치
- ☐ * 가맹본부명

실습6. 제2 정규형

이벤트참여

- ☐ # 고객ID
- ☐ # 이벤트번호
- ☐ * 당첨여부
- ☐ * 고객이름
- ☐ * 등급
- ☐ * 할인율

3-2 정규화(Normalization)

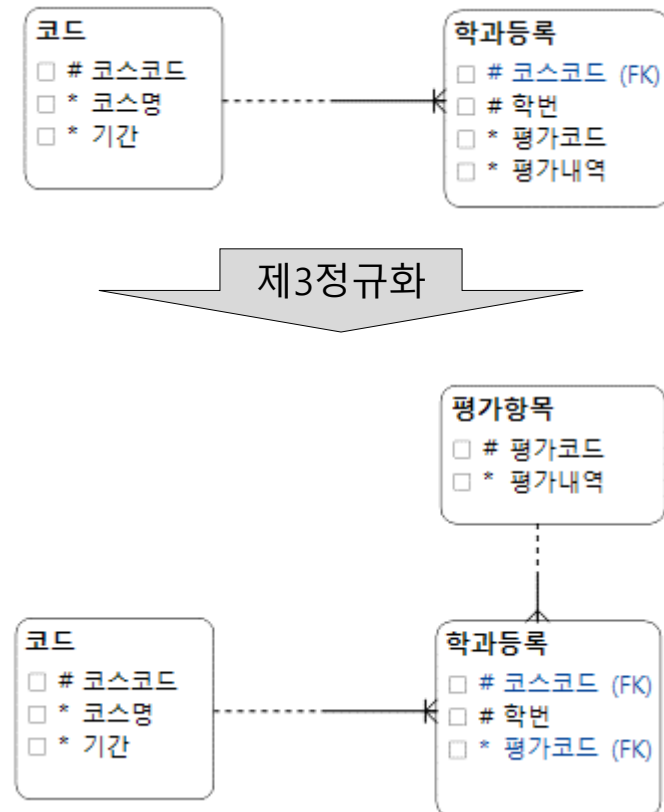
❖ 정규화 단계 - 3차 정규형 (3NF, Third Normal Form)

✓ 정의

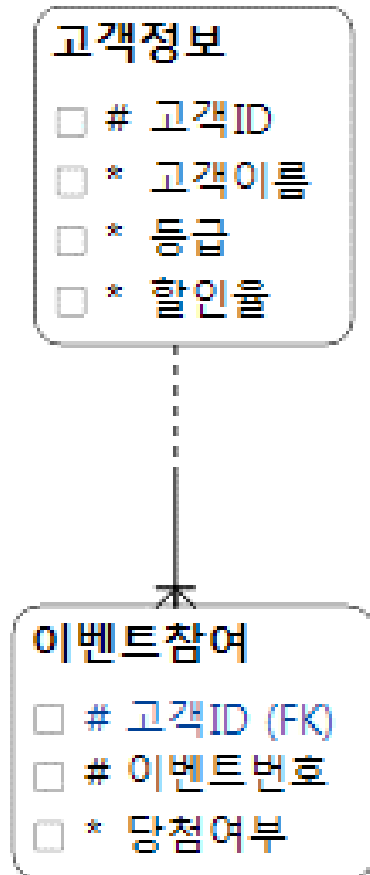
- 2차 정규형을 만족하고 식별자를 제외한 나머지 속성들 간의 종속이 존재하면 안된다. 이것이 3차 정규형을 만족하는 것이다.

✓ 정규화 작업

- 학과등록 엔티티에서 평가코드, 평가내역 속성들이 서로 간에 종속적이다. 즉, 평가내역 속성은 평가코드 속성에 종속적이다. 그렇기 때문에 이것은 3차 정규형을 위반하고 있는 것이다.
- 3차 정규형을 위반하고 있을 시에는 평가항목 엔티티처럼 부모 엔티티가 생성되고 그 부모 엔티티로부터 UID Bar가 없는 관계를 상속받게 된다.



실습7. 제3 정규형

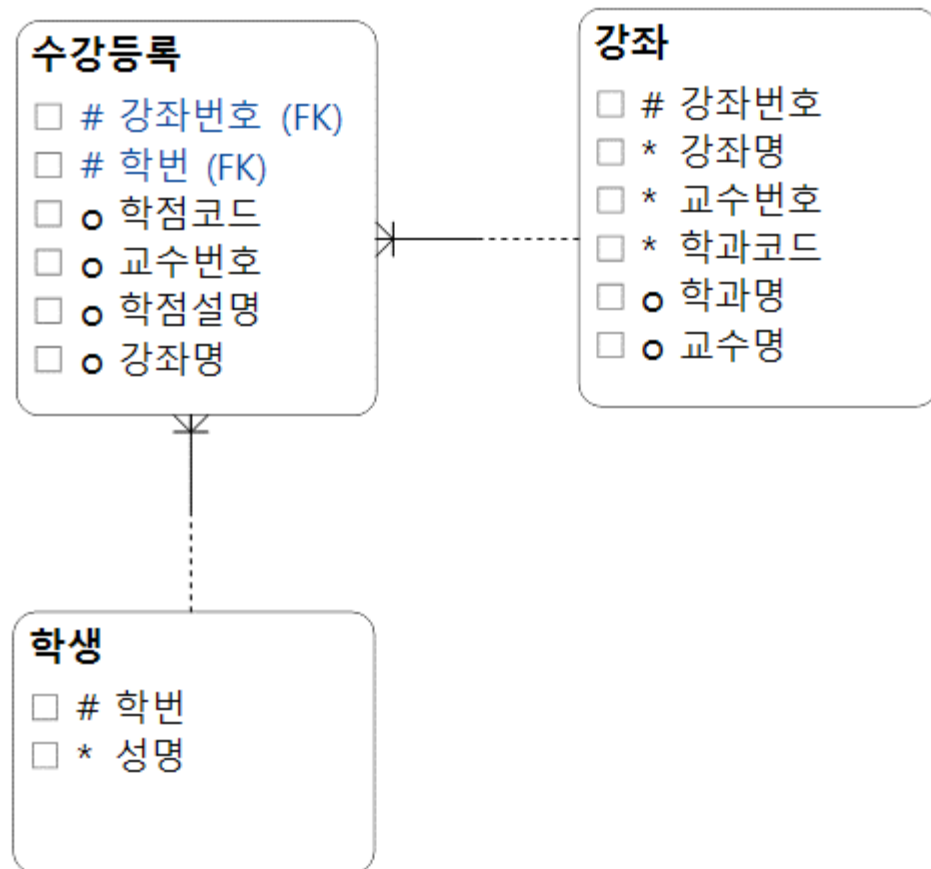


실습8. 제3 정규형

주문

- ☐ # 주문번호
- ☐ * 주문가격
- ☐ * 선적일자
- ☐ o 지불방법
- ☐ * 고객번호
- ☐ * 고객명
- ☐ * 주민번호
- ☐ * 주소

실습9. 제3 정규형

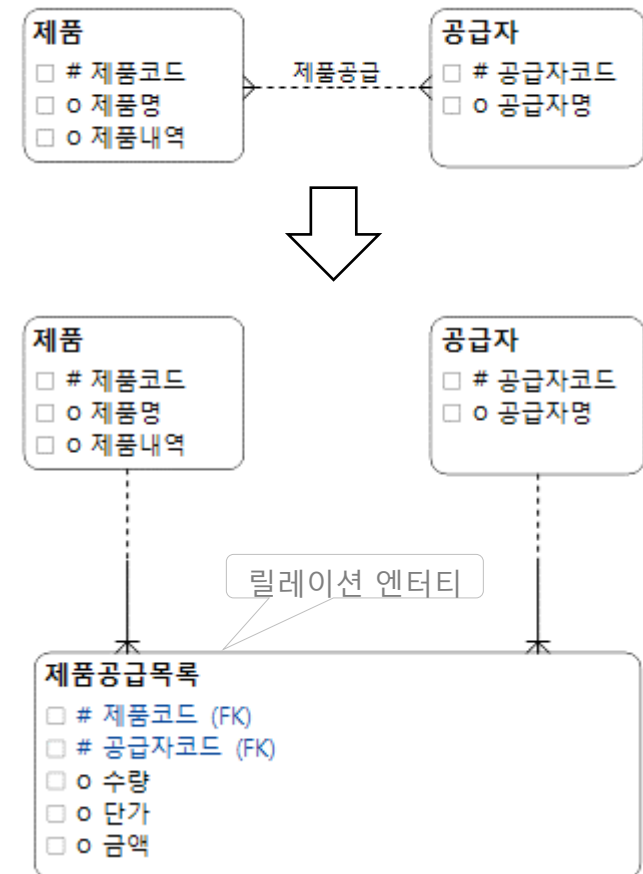


3-3 M:M 관계 해소

❖ M:M 관계의 의미

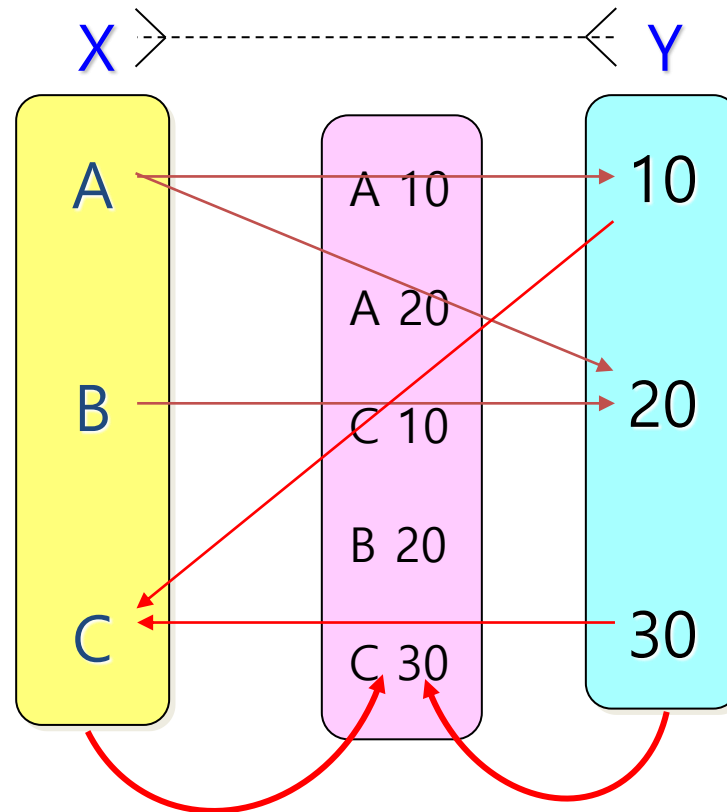
M:M 관계는 논리 데이터 모델링 과정에서 많이 나타나며, 최종적으로 완성된 데이터 모델에는 존재할 수 없는 형태라고 할 수 있다.

- ✓ 실제 업무 중 대부분은 M:M 관계이다. 기업이 관리하고 있는 많은 데이터가 M:M 관계로 표현되고 향후에 모델링이 더 진행됨에 따라 이것이 해소된다.
- ✓ 키 엔티티와 키 엔티티 간에는 대부분 M:M 관계이다. 그래서 데이터 모델 상세화 단계에서 이러한 M:M 관계가 해소되면서 액션 엔티티가 새롭게 도출되기도 한다.
- ✓ 지속적으로 발생하는 대다수의 엔티티는 M:M 관계이다. 즉, 기업의 업무 내용을 관리하는 데이터는 대부분 이러한 관계에 의해서 생겨난 엔티티라고 볼 수 있다.
- ✓ 제품 엔티티와 공급자 엔티티 간에는 M:M 관계가 존재한다. 이 관계는 제품 공급이라는 업무 내용에 의해서 두 엔티티 간에 생겨난 M:M 관계이다.
- ✓ 이 관계가 해소된 결과로 공급제품목록이라는 새로운 엔티티가 만들어지고 또한 각각의 두 엔티티 즉 제품, 공급자와 1:M의 관계를 부모로서 가지게 된다.



3-3 M:M 관계 해소

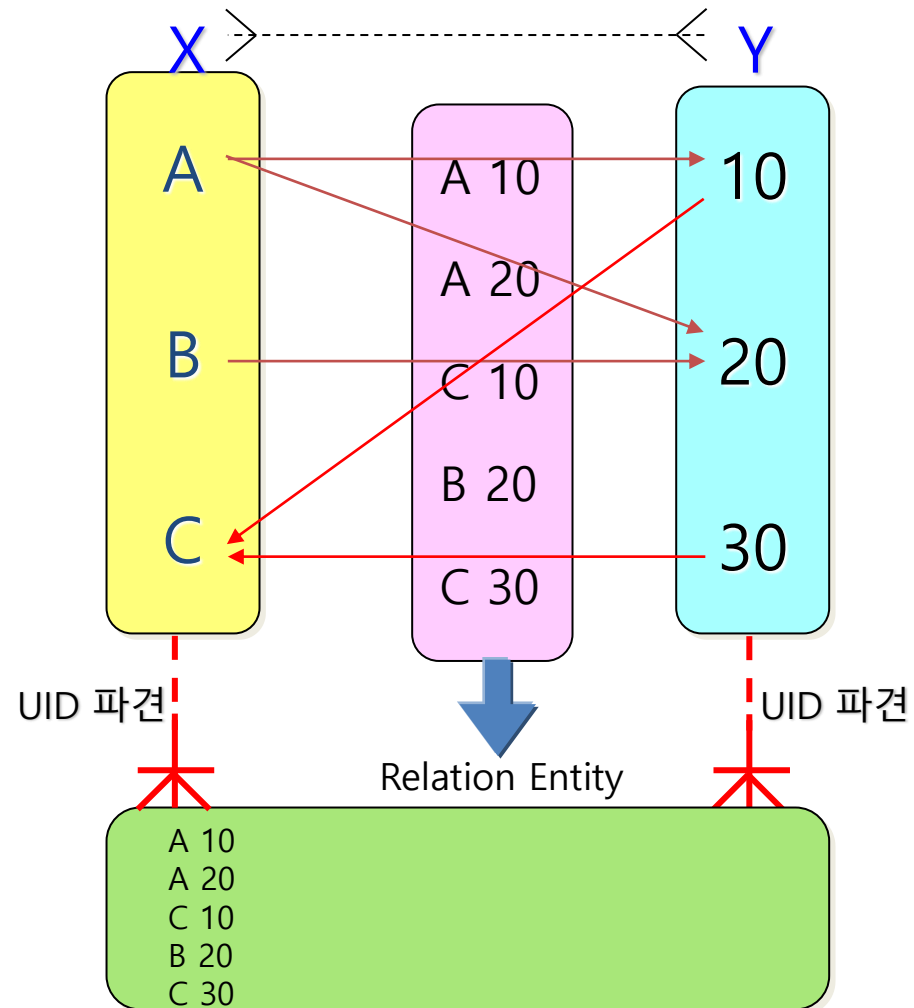
❖ M:M 관계 해결 원리



X쪽의 A가 Y쪽의 무엇과
대응되는지 정보로써 필요

3-3 M:M 관계 해소

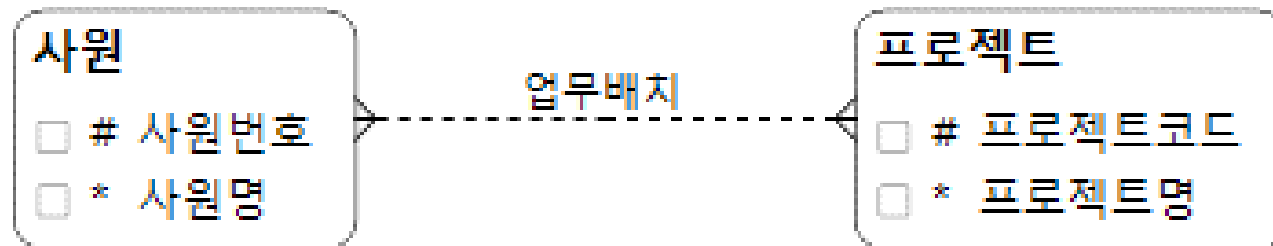
❖ M:M 관계 해결 원리



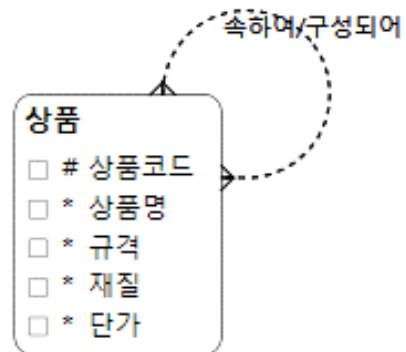
실습10. M:M 관계 해소



실습11. M:M 관계 해소



실습12. M:M 순환관계(BOM) 해소



4-1 이력 관리란?

- ❖ 현재는 단지 하나의 점에 불과하지만 과거란 엄청난 개수의 점이 모여 있는 형상이다. 이력은 선분이고 현재의 순간은 점이므로 선분을 관리해야 한다.

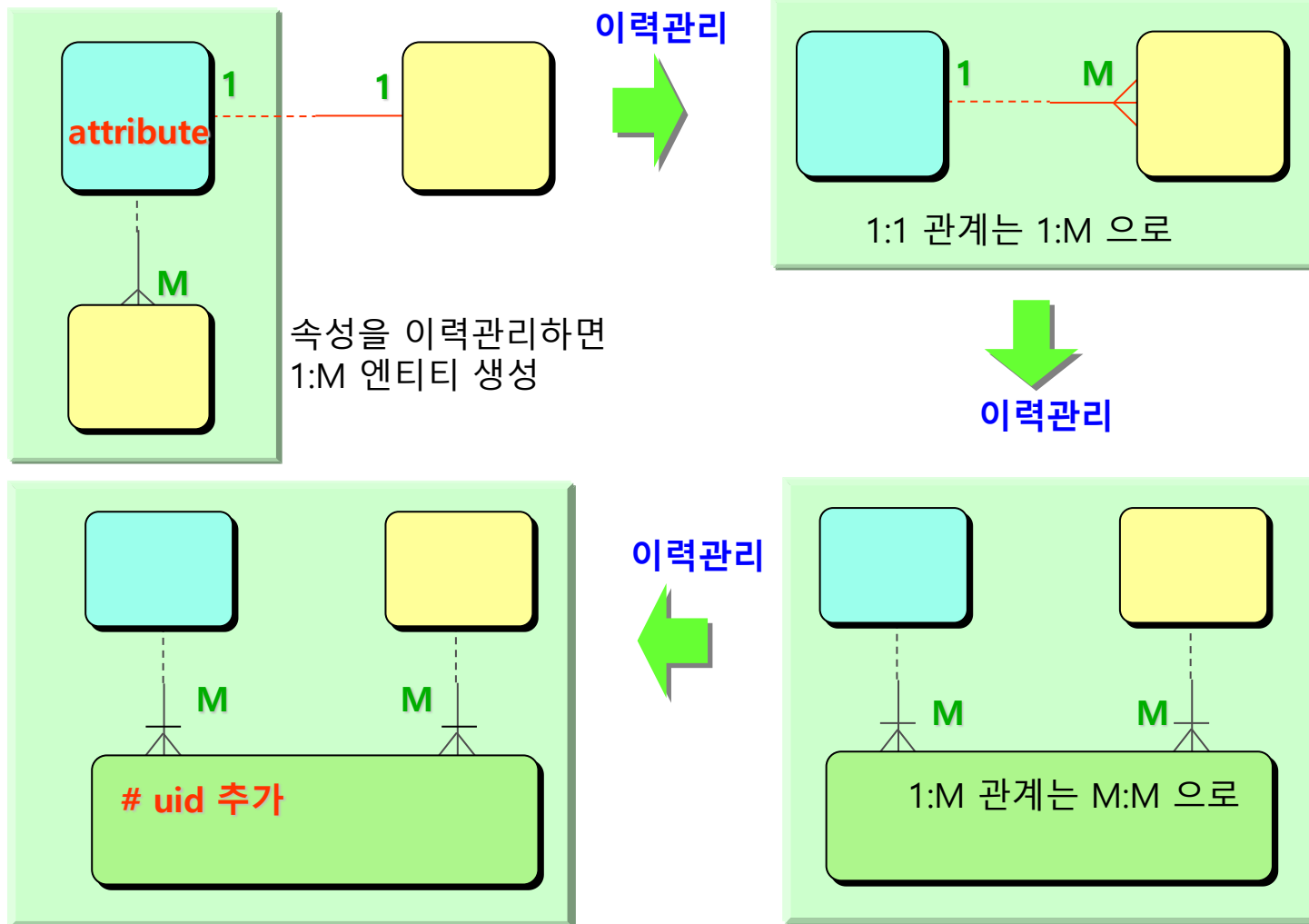
❖ 사용자 조사

데이터의 이력을 관리한다는 것은 많은 비용이 들어가므로 사용자의 검증 과정이 필요하다.

- ✓ 변경 내역을 감시할 필요가 있는가?
- ✓ 시간의 경과에 따라 데이터가 변할 수 있는가?
- ✓ 시간의 경과에 따라 관계가 변할 수 있는가?
- ✓ 과거의 데이터를 조회할 필요가 있는가?
- ✓ 과거 버전을 보관할 필요가 있는가?

4-2 이력 관리란?

❖ 이력 관리에 따른 관계 형태의 변화



4-2 이력 관리 대상 선정

❖ 이력 데이터의 종류

✓ 발생 이력(Occurrence History) 데이터

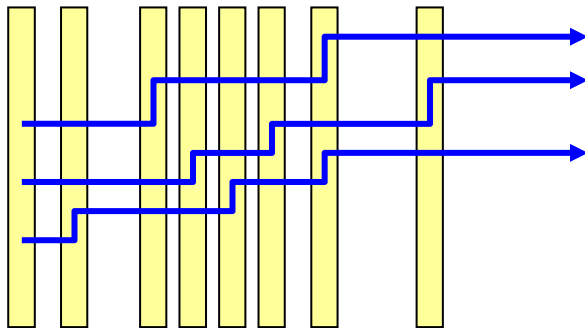
어떤 데이터가 발생할 때마다 이력 정보를 남기는 것 (예 : 고객의 접속 기록)

✓ 변경 이력 (Modification History) 데이터

데이터가 변경될 때마다 변경 전과 후의 차이를 확인하기 위해 남기는 이력 (예 : 주문 정보 변경)

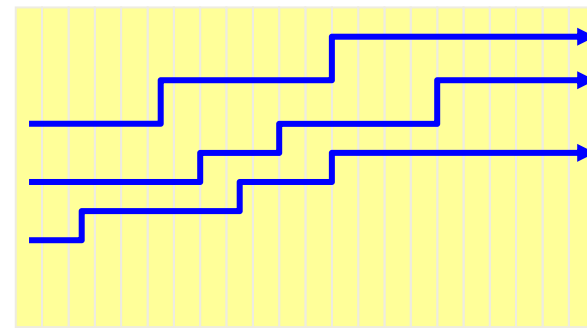
✓ 진행 이력 (Progress History) 데이터

업무의 진행에 따라 이 데이터를 이력 정보로 남기는 것(예 : 주문의 업무 처리)



EVENT 발생 시 마다 생성

Event가 발생할 때마다 사진을 찍어 두어야 이력을 관리할 수 있는가?

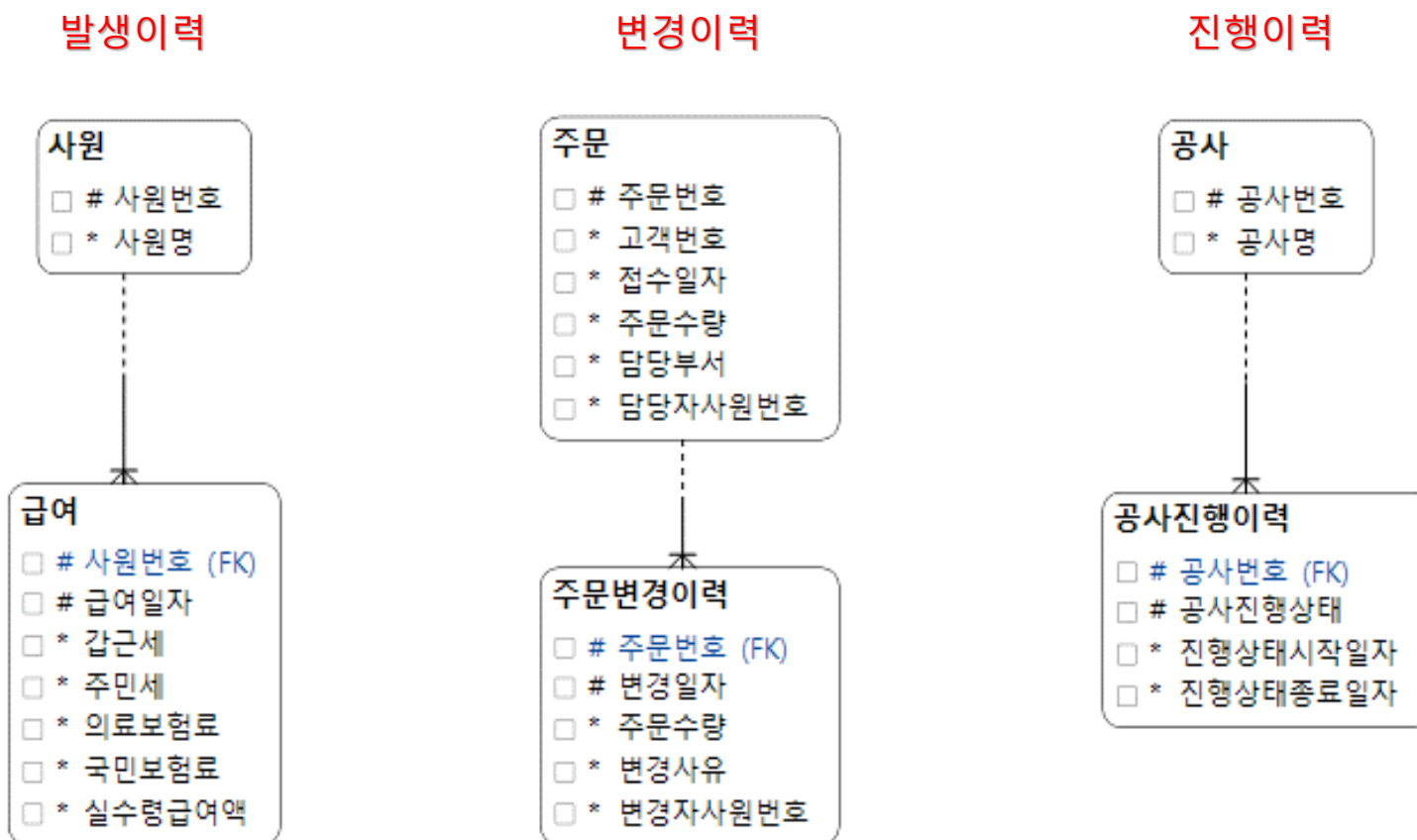


DAILY 마다 생성

아무 변화가 없는 경우도 데이터 생성
그러나 완벽한 이력관리는 불가능

4-2 이력 관리 대상 선정

❖ 이력 데이터의 종류



4-2 이력 관리 대상 선정

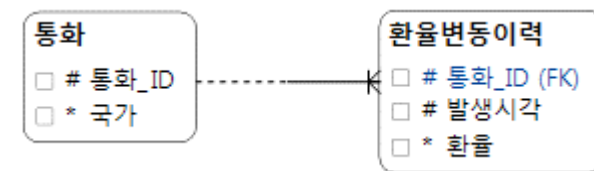
❖ 이력 관리 형태

✓ 시점(Event) 이력

- 데이터의 변경이 발생한 시각만을 관리
- 로그성 데이터로 이력을 쌓아두고자 할 경우(임의 시점에 대한 조회 요구사항이 없음)
- 이력 데이터를 저장하고 이해하기에는 용이하지만, 활용상에서 비효율이 많이 나타난다.
- 과거 임의의 시점에서의 이력을 재현하기에 비효율이 많이 발생
- 특정 통화의 환율이 변경되면 어느 시점에 얼마의 값으로 변경되었다라는 정보를 관리한다. 이러한 방식은 특정한 시점의 데이터를 추출하고자 할 경우에 불필요한 작업을 수행하게 된다

- 시점 이력 활용 예

```
SELECT 환율
FROM 환율변동이력
WHERE 발생시각 = (SELECT MAX(발생시각)
                  FROM 환율변동이력
                  WHERE 발생시각 <= '20020521' AND 통화_ID = 'USD')
AND 통화_ID = 'USD';
```



4-2 이력 관리 대상 선정

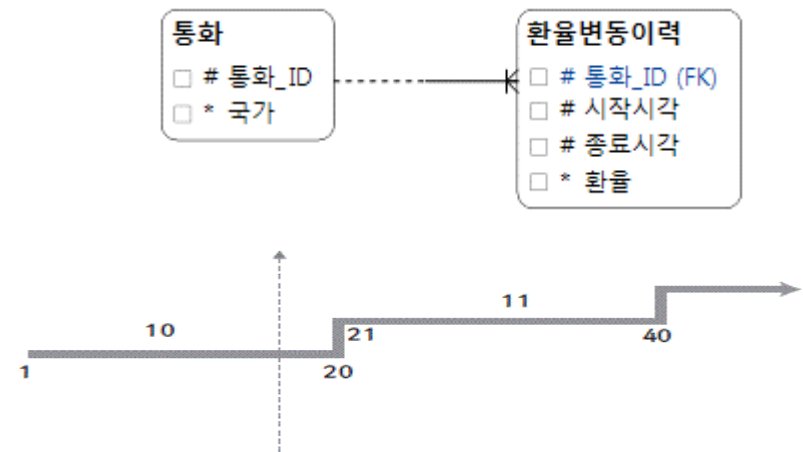
❖ 이력 관리 형태

✓ 선분 이력

- 데이터 변경의 시작 시점부터 그 상태의 종료 시점까지 관리
- 이력관리 대상 속성들이 의미하는 정보가 일정 시점의 정보만을 나타내는 것이 아니라 일정 기간동안 해당 정보가 유효하다는 의미
- 임의의 시점의 데이터를 찾고자 하는 요구가 있는 경우에 유용
- 이력 데이터를 저장하고 이해하는데 있어서 점이력 방식보다 불리하지만, 과거 임의의 시점에서의 이력을 재현하고 활용하는데 매우 효율적임
- 각 통화의 특정 기간 동안 유효한 환율을 관리

- 선분 이력 활용 예

```
SELECT 환율
FROM 환율변동이력
WHERE '20020521' between 시작시각 and 종료시각
AND 통화_ID = 'USD';
```



점을 통과하는 선분 찾는 방법

- 부등식 표현 : 시작점 <= 17 <= 종료점
- 연산자 표현 : 17 Between 시작점 and 종료점

4-2 이력 관리 대상 선정

❖ 선분 이력에서 식별자 결정 시 고려사항

선분 이력을 관리하는 엔티티의 UID를 확정하는 것은 향후에 테이블로 만들어지고 사용될 때의 성능 측면도 고려되어야 한다. 의미적으로 Unique을 점검할 수 있는 조치가 필요하다.

통화_ID + 시작일 + 종료일

시작일~종료일은 겹쳐진 것이 없다.

어느 것을 PRIMARY KEY로 해야 하는가?

X

통화_ID + 시작일

Unique ?

X

통화_ID + 종료일

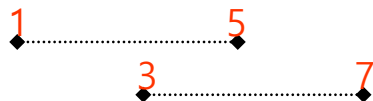
Unique ?

O

통화_ID + 시작일 + 종료일

Unique ?

점을 비교하면 Unique 한 것처럼 보이지만 DBMS가 check 해주는 Constraints는 아무 의미가 없다.

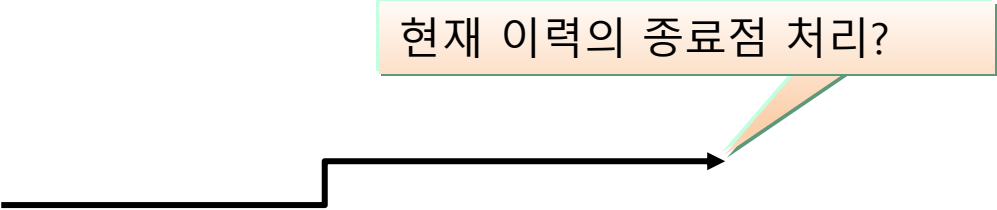


- 점을 비교하면 Unique 하지만 선분은 겹쳐 있다.
- 종료점이 없으면 선분이 겹치는 지를 확인할 수 없다.

4-2 이력 관리 대상 선정

❖ 선분 이력에서 종료점 처리 시 주의사항

- ✓ 종료점이 미정이므로 NULL
 - 논리적으로는 타당하지만 비교가 불가능
 - 인덱스를 사용하지 못하므로 수행 속도 저하
- ✓ 수렴하므로 최대치 부여
 - 아직 종료되지 않았으므로 무한히 계속되는 것으로 간주
 - 최대치 부여 (예; 일자라면 9999/12/31)
 - 가능한 TABLE creation 시 Default constraints 부여
 - 수행 속도에 유리



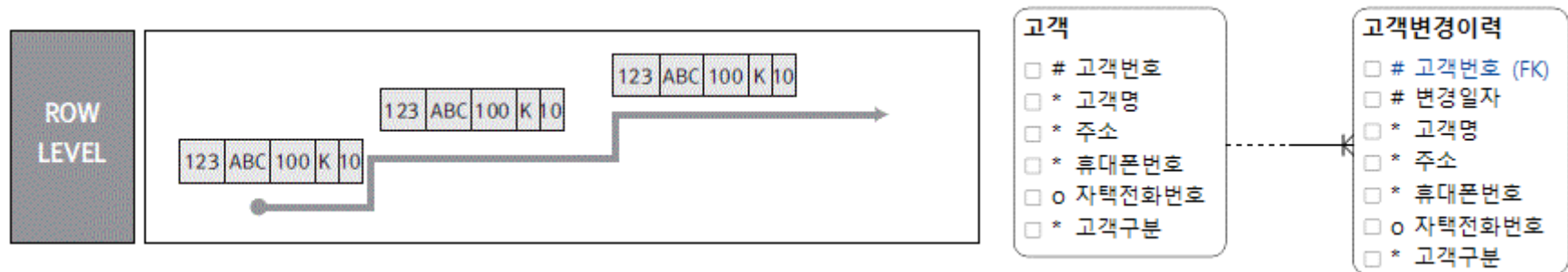
현재 이력의 종료점 처리?

4-2 이력 관리 대상 선정

❖ 이력 관리 유형

✓ 인스턴스 레벨 이력 관리

하나의 인스턴스의 어떤 변경이라도 발생하면 전체 인스턴스를 새롭게 생성하는 방식의 이력 관리 유형



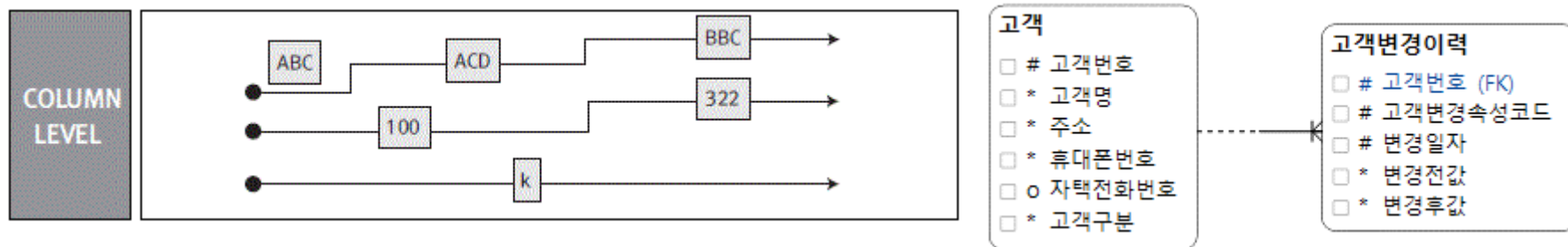
- 한번의 액세스를 스냅샷을 참조하는 것이 가능하다. 즉, 한번의 액세스를 해당 시점의 모든 데이터를 참조하는 것이 가능
- 로그성 데이터를 저장할 목적인 경우 적당한 방법이다.
- 다른 이력 관리 유형에 비해서 저장하기가 쉽다.
- 가장 큰 단점 중에 하나는 하나 이상의 칼럼에 변경이 발생했을 때 이벤트가 모호해진다는 점이다.
- 만약 이벤트가 지식 정보를 가지게 된다면 매우 치명적이다.
- 이력 관리의 다른 유형들에 비해서 저장 공간의 낭비를 초래할 수 있다.
- 실제 어떤 데이터가 변경된 것인지를 찾기 위해서는 과거의 데이터를 Merge해서 비교를 해야만 가능할 수 있다.
- 특정 순간의 스냅샷만 보는 게 아니라면 처리가 복잡해지는 경향이 있다.
- 변화가 빈번하게 발생하는 상황이라면 고려해 볼 수 있는 유형이다.

4-2 이력 관리 대상 선정

❖ 이력 관리 유형

✓ 속성 레벨 이력 관리

이력을 관리할 대상 속성에서 변화가 생길 때만 이력을 생성하는 방식



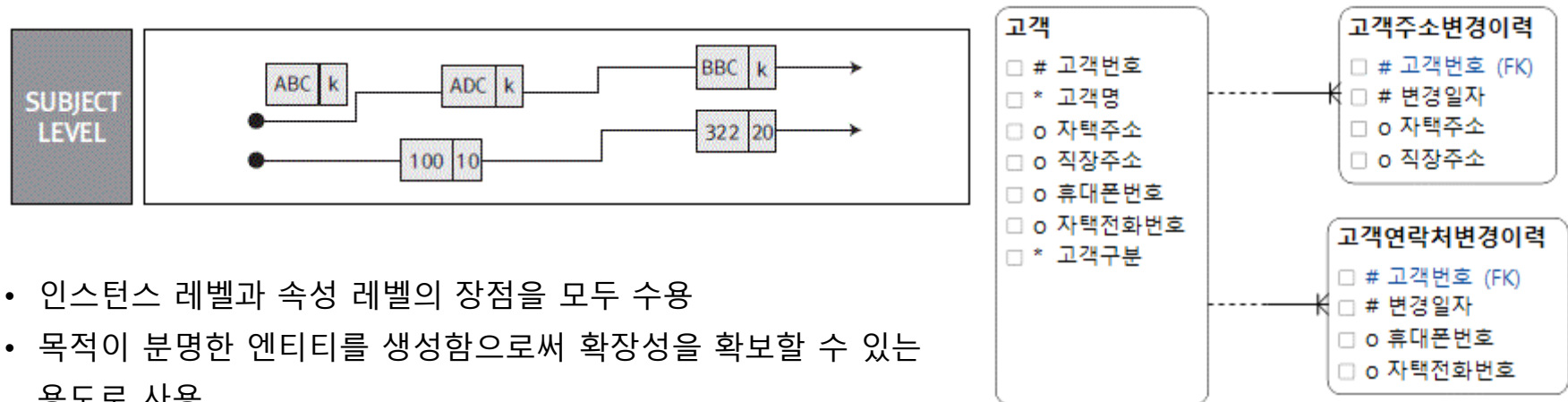
- 실제 어떤 데이터가 변경되었는지가 분명
- 하나의 이력 관리 엔티티에서 다른 엔티티와 통합 이력 관리가 가능
- 변경된 것만 처리하기 때문에 독립적 처리가 가능
- 변화 발생 가능성은 매우 낮으면서 대상 속성은 매우 많은 경우에 사용하는 것이 유리
- 특정 속성들에 변화가 집중되는 경우에 해당 속성에 대해서만 이력을 관리할 수 있기 때문에 유리
- 여러 속성에 대한 이력이 필요할 때 많은 Merge가 발생
- 이력 관리의 다른 유형들에 비해서 향후 사용될 데이터 액세스 쿼리에서 조건 검색이 조금 어려움
- 변화가 너무 많은 경우에는 적용이 곤란

4-2 이력 관리 대상 선정

❖ 이력 관리 유형

✓ 주제 레벨 이력 관리

내용이 유사하거나 연동될 확률이 높은 것별로 인스턴스 레벨 이력을 관리하는 방법



- 인스턴스 레벨과 속성 레벨의 장점을 모두 수용
- 목적이 분명한 엔티티를 생성함으로써 확장성을 확보할 수 있는 용도로 사용
- 변경 부분만 처리가 가능 (독립적 처리 가능)
- 다른 엔티티와 통합 이력 관리가 가능
- 속성 레벨의 단점을 해소
- 전체를 참조할 때 인스턴스 레벨에 비해 Merge가 발생하는 문제점이 존재
- 부문에 따라 변경 정도의 차이가 심한 경우 유리

실습13. 고객연락처 정규화

❖ 정규화 과정을 통해서 변경 이상이 발생할 개연성을 가지고 있는 엔터티를 정규화된 엔터티로 변환하시오.

고객정보

고객번호	고객명	주민번호	전화번호
170001	정*룡	901110-1*****	010-1861-2025
170003	전*철	920822-1*****	010-2403-1659, 02-776-4100, 031-256-1659
180005	오*숙	950420-2*****	010-1808-9633

실습14. 상품색상 정규화

❖ 정규화 과정을 통해서 변경 이상이 발생할 개연성을 가지고 있는 엔터티를 정규화된 엔터티로 변환하시오.

상품정보

상품번호	상품명	성별구분	단가	색상
B0001	블라우스	여성	20,000	red, white, blue
T0007	티셔츠	공용	10,000	white, blue
H0002	반바지	여성	15,000	red, blue

실습15. 주문관리 정규화

❖ 1~3차 정규화 과정을 통해서 변경 이상이 발생할 개연성을 가지고 있는 엔티티를 정규화된 엔티티로 변환하시오.

주문목록

- ☐ # 제품번호
- ☐ * 제품명
- ☐ * 주문번호
- ☐ * 주문일자
- ☐ * 고객번호
- ☐ o 사업자번호
- ☐ * 우선순위
- ☐ * 연락처
- ☐ * 주문수량
- ☐ * 주문단가

주문목록

제품 번호	제품명	주문 번호	주문 일자	고객 번호	사업자 번호	우선 순위	연락처	주문 수량	주문 단가
1001	모니터	A0001	05/11	4520	398201	1	0953	150	90,000
1001	모니터	A0002	05/14	2341	200212	3	0861	600	70,000
1007	마우스	A0001	05/11	4520	398201	1	0953	100	5500
1007	마우스	A0002	05/14	2341	200212	3	0861	300	5000
1007	마우스	B2011	05/16	2341	200212	3	0861	200	5000
1201	스피커	B2012	05/18	8320	563892	8	5300	70	10,000

실습16. 출고관리 정규화

❖ 1~3차 정규화 과정을 통해서 변경 이상이 발생할 개연성을 가지고 있는 엔터티를 정규화된 엔터티로 변환하시오.

출고

- | | | |
|------------------------------------|----------------------------------|----------------------------------|
| <input type="checkbox"/> # 출고번호 | <input type="checkbox"/> o 대리점주소 | <input type="checkbox"/> o 단가2 |
| <input type="checkbox"/> o 출고일자 | <input type="checkbox"/> o 제품번호1 | <input type="checkbox"/> o 수량2 |
| <input type="checkbox"/> o 출고부서 | <input type="checkbox"/> o 제품명1 | <input type="checkbox"/> o 금액2 |
| <input type="checkbox"/> o 부서명 | <input type="checkbox"/> o 단가1 | <input type="checkbox"/> o 제품번호3 |
| <input type="checkbox"/> o 출고사원번호 | <input type="checkbox"/> o 수량1 | <input type="checkbox"/> o 제품명3 |
| <input type="checkbox"/> o 사원명 | <input type="checkbox"/> o 금액1 | <input type="checkbox"/> o 단가3 |
| <input type="checkbox"/> o 출고대리점번호 | <input type="checkbox"/> o 제품번호2 | <input type="checkbox"/> o 수량3 |
| <input type="checkbox"/> o 대리점명 | <input type="checkbox"/> o 제품명2 | <input type="checkbox"/> o 금액3 |

실습17. 대학관리

❖ 업무처리규정

- ✓ 교수는 아이디, 이름, 나이, 직위, 연구분야를 가진다.
- ✓ 학과에는 학과번호, 학과이름, 학과사무실이 있다.
- ✓ 대학원생은 아이디, 이름, 나이, 학위과정(석사/박사)을 가진다.
- ✓ 과제는 과제번호, 지원기관, 개시일, 종료일, 예산액이 있다.
- ✓ 학과마다 그 학과를 운영하는 교수(학과장)가 한 명씩 있다.
- ✓ 한 교수가 여러 학과에서 근무할 수 있는데, 이때 각 학과별로 참여백분율이 기록된다.
- ✓ 대학원생에게는 학위 과정을 밟을 전공학과가 하나씩 있다.
- ✓ 대학원생에게는 어떤 과목을 들으면 좋을지 조언해주는 선임 대학원생(학생조언자)이 있다.
- ✓ 과제는 한 교수(연구책임자)에 의해 관리된다.
- ✓ 과제는 한 사람 이상의 교수(공동연구책임자)에 의해 수행된다.
- ✓ 한 과제는 한 명 이상의 대학원생(연구조교라고 한다)에 의해 수행된다.