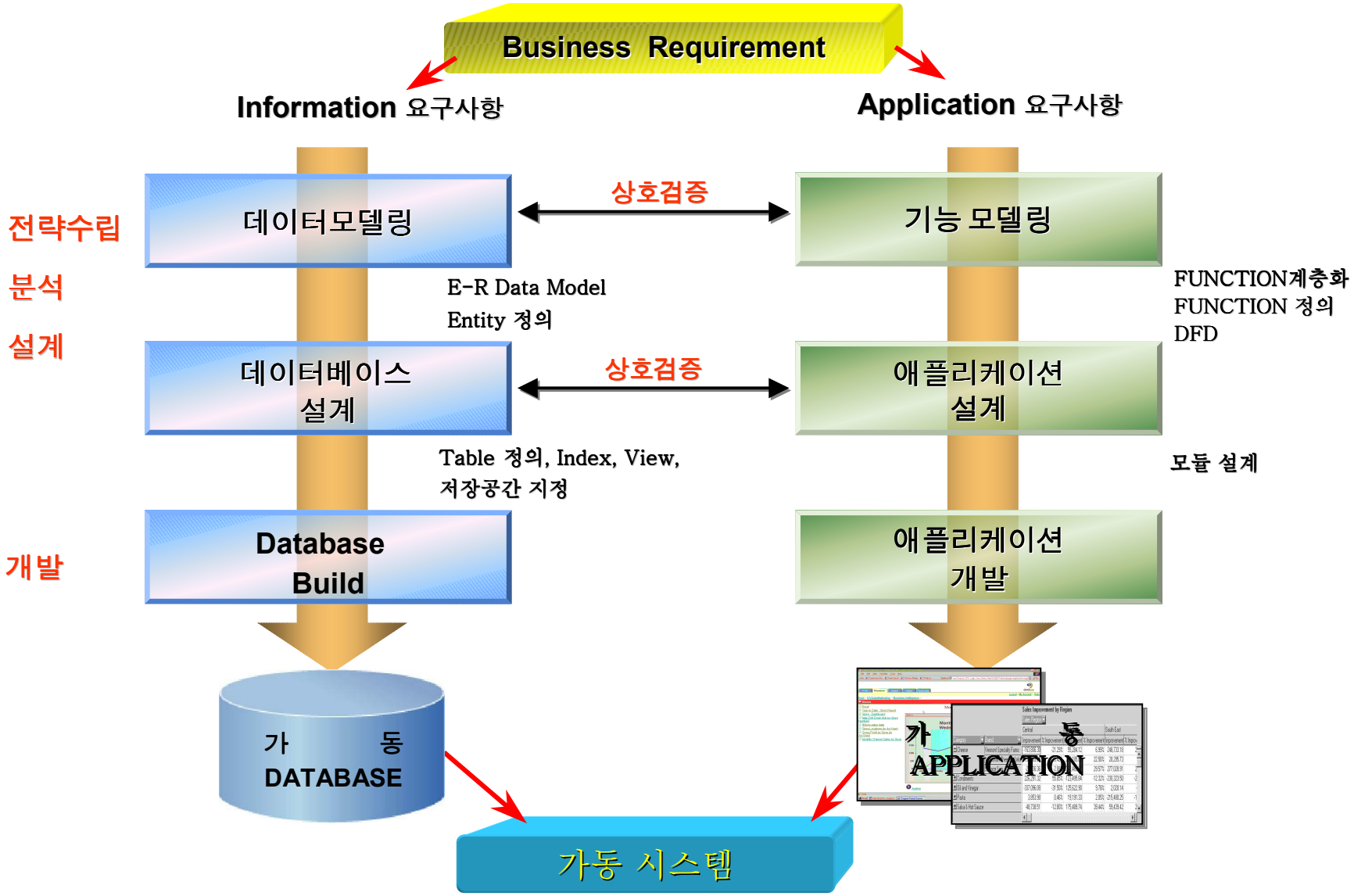
데이터 모델 관리
- Modeler 관점의 품질관리

논리 데이터 모델

명 재 호 이사
엔코아 컨설팅

목차


Business Requirement
Information 요구사항
Application 요구사항
전략수립
분석
설계
개발
데이터모델링
상호검증
기능 모델링
E-R Data Model
Entity 정의
FUNCTION계층화
FUNCTION 정의
DFD
데이터베이스
설계
상호검증
애플리케이션
설계
Table 정의, Index, View,
저장공간 지정
모듈 설계
Database
Build
애플리케이션
개발
가 동
DATABASE
가 동
APPLICATION
가동 시스템

Ⅰ. 데이터모델 개요 – 데이터 아키텍처에서 논리 데이터 모델

개괄적 모델

- 건축물의 조감도, 데이터의 최상위 집합
- 전사적 차원에서 관리
- 기존 엔터티들을 추상화하여 생성
- 아티클을 이용하여 개괄적인 관리항목 정의

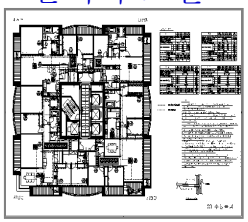
개념적 모델

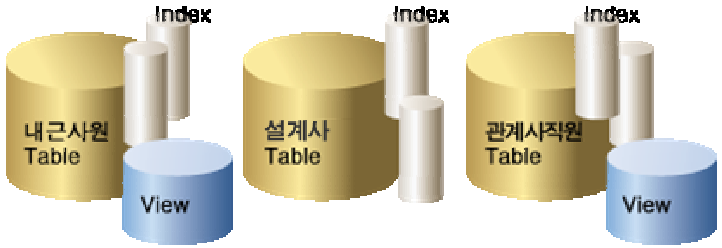
- 주요 키이 엔터티와 핵심 행위 엔터티들로 구성
- 데이터 모델의 골격에 해당
- 세부적인 부분집합을 정의하여 명확화
- 개괄적 모델과 구체적 연결(Alignment)

논리적 모델

- 모든 논리적인 데이터 객체들을 도출
- 최종 식별자가 확정된 모델
- 정규화 및 필요한 반정규화를 실시한 모델
- 데이터 모델 상세화와 통합화를 완료한 모델
- 전략적인 이력관리 방안까지 확정된 모델

물리적 모델

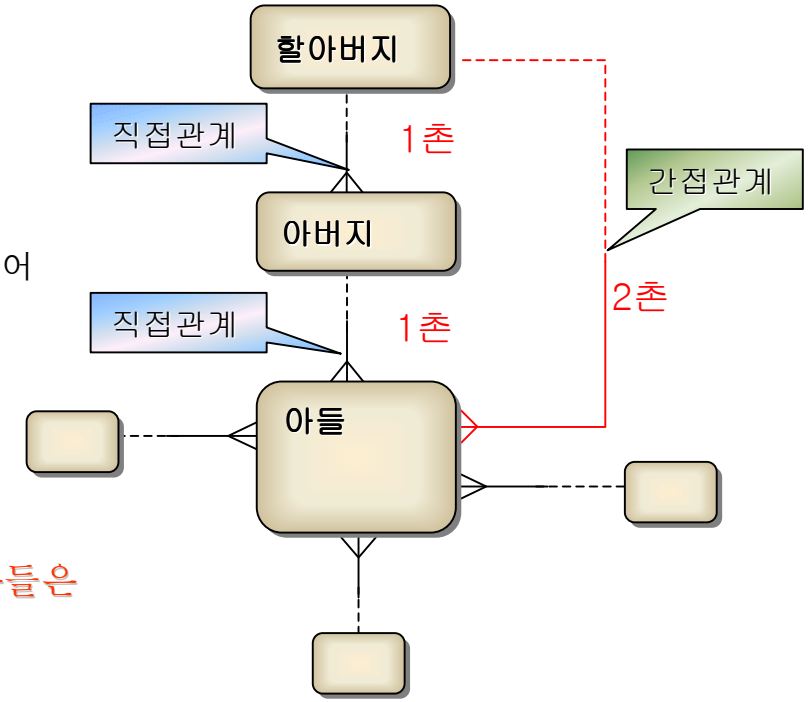


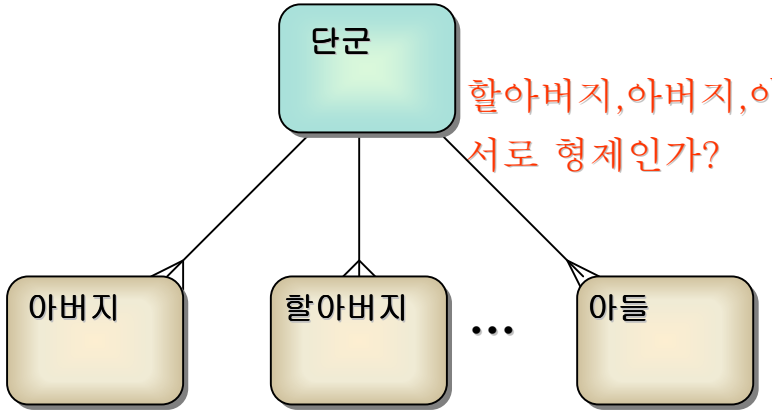


- 논리적 모델과 독립적
- 테이블, 컬럼, 제약요건 정의, 인덱스, 파티션 정의
- 메타 데이터 관리와 연계
- 부가적 단계를 모두 포함

Ⅱ. 데이터 모델의 잘못된 유형 – 형제형 접근

- 모델링은 1촌 관계만을 정의하는 것
- 1촌 관계만 정의해도 모든 촌수를 알 수 있다
- 간접관계는 추후 물리설계에서 감안
- 직접종속이 모두 절대종속은 아니다.
- 나를 위해 정자와 난자를 제공한 것이 절대종속
- 직접종속이고 절대종속이 내 부모 = 의미상의 주어



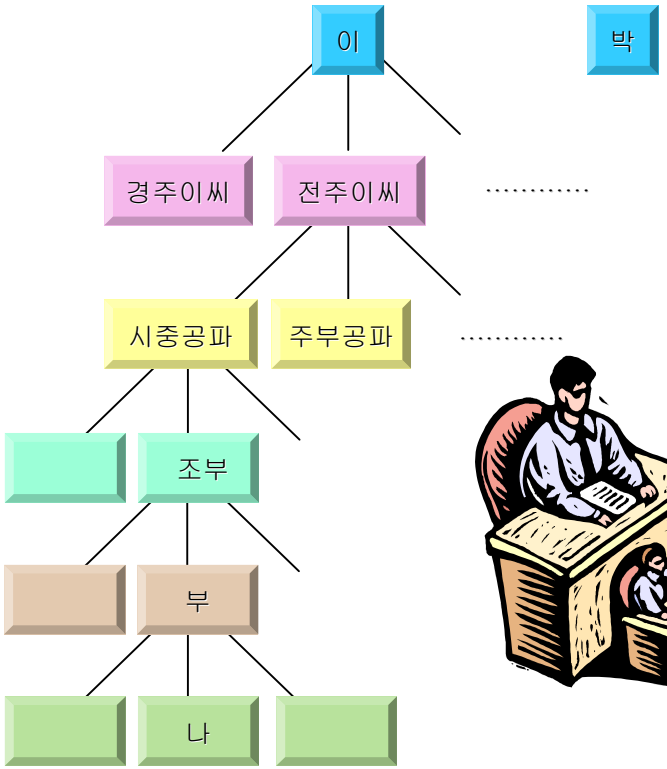


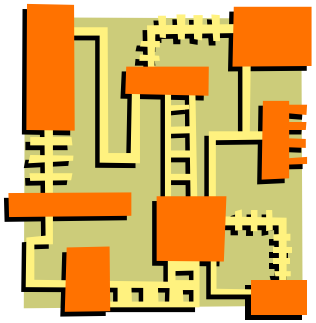
할아버지,아버지,아들은
서로 형제인가?

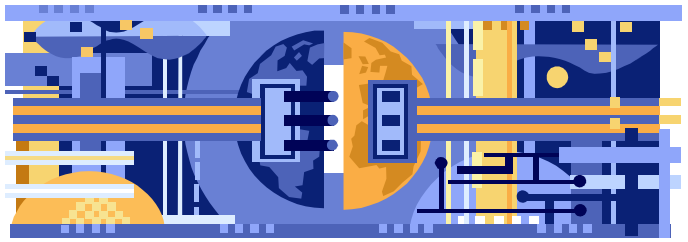
직접종속, 간접종속 : 1촌 관계 여부

절대종속, 상대종속 : 탄생에 미친 영향

Ⅱ. 데이터 모델의 잘못된 유형 –족보형, 네트워크형 접근



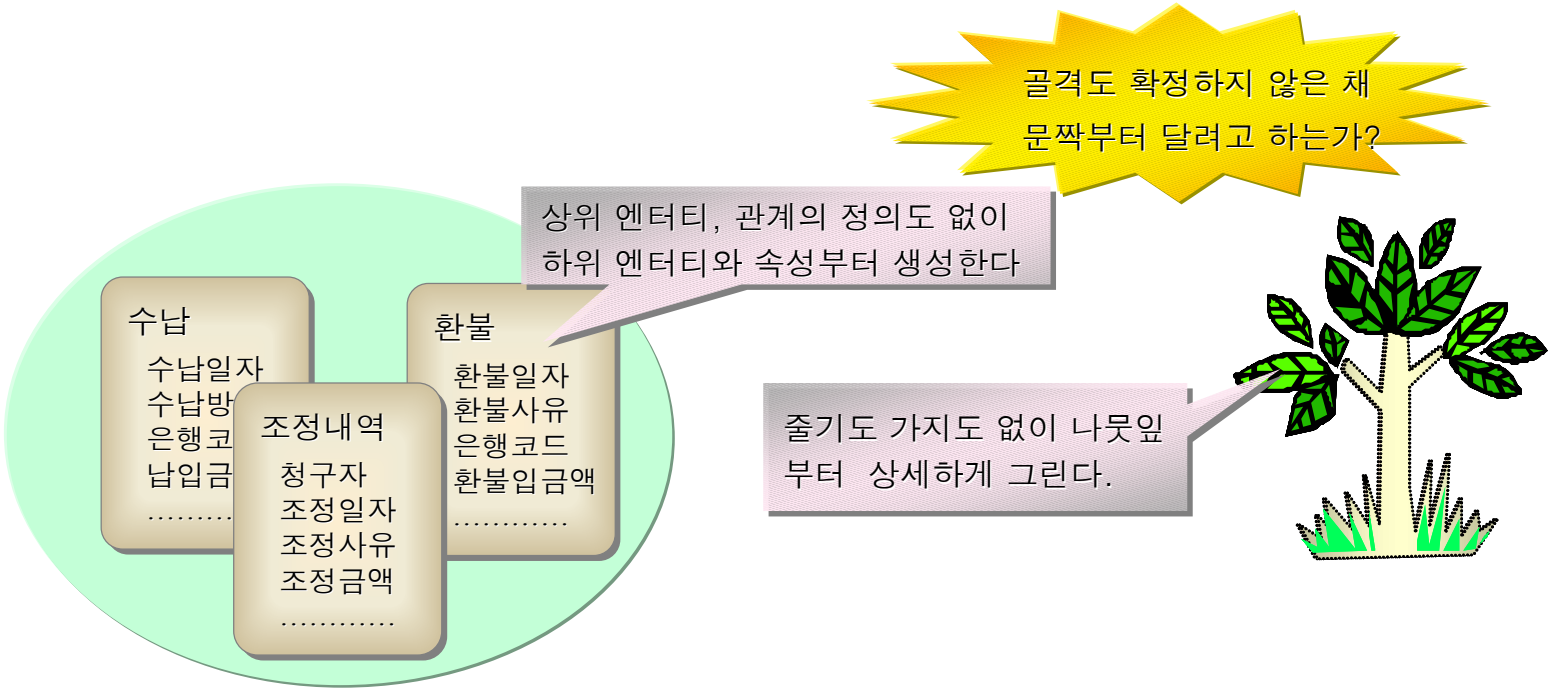




- 우리 몸 속에는 아버지 피만 흐르는가?
- 나는 정말 '이' 씨의 순수 혈통인가?
- 모계는 아무런 영향을 미치지 않았는가?

- ERD가 NETWORK 회로도 인가?
- 데이터 모델에는 흐름이 없다.
- TFD (Table Flow Diagram) ?

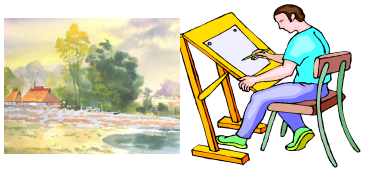
Ⅱ. 데이터 모델의 잘못된 유형 –Bottom up 접근



- 가장 하위 엔터티에서 정의되는 내용은 해당 엔터티의 고유 속성이 아닌 경우가 많다.
- 즉, 다른 엔터티로 부터의 관계(Relationship)인 경우가 대부분이다.
- 이런 경우에 각 Relationship의 주체가 되는 엔터티가 정의되지 않았는데 최하위 엔터티를 정의한다는 것은 골격없이 집을 짓는 것과 같다.
- 골격이 되는 엔터티를 먼저 명확화하고 하위 엔터티를 정의하는 것이 바람직하다.

Ⅱ. 데이터 모델의 잘못된 유형

■ 책상에 앉아 그리는 풍경과



- 업무 담당자와 대화 하면서 결정사항을 결정해 간다.
- 이러한 결정들이 데이터 모델에 바로 반영되어야 한다.
- 결국 최종적인 결론은 데이터 오너(**Owner**)가 하는 것이다.

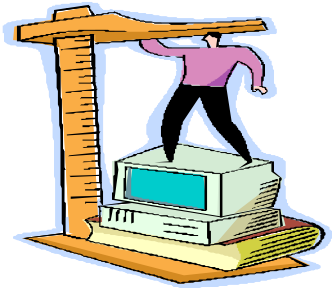
■ 교육을 받는 수사관

- 현업사용자가 설명하고 모델러가 교육받는 모델링은 잘못된 접근이다.
- 모델러는 필요한 정보를 캐내는 사람이다.
- 이러한 정보가 체계적으로 정리되고 설계도면이 되게 하는 역할을 모델러가 해야 한다.

■ 모델러는 예술가

- 현재를 있는 그대로 표현하는 것이 아니다
- 현재를 기반으로 전략적이고 창조적인 작품을 만든다.

Ⅱ. 데이터 모델의 잘못된 유형 – 데이터 모델링의 객관화

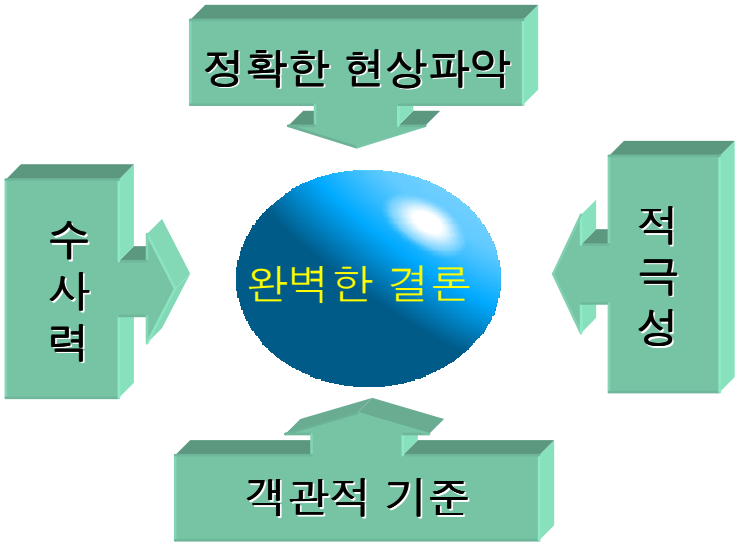


- 데이터 모델링은 실수를 찾아주는 컴파일러가 없다.
- 개발 및 테스트 단계에 가서야 문제가 드러난다.
- 자신의 결정에 확신을 가지기 어렵다.
- 무엇을 모르는 지를 모르는 사람이 많이 있다

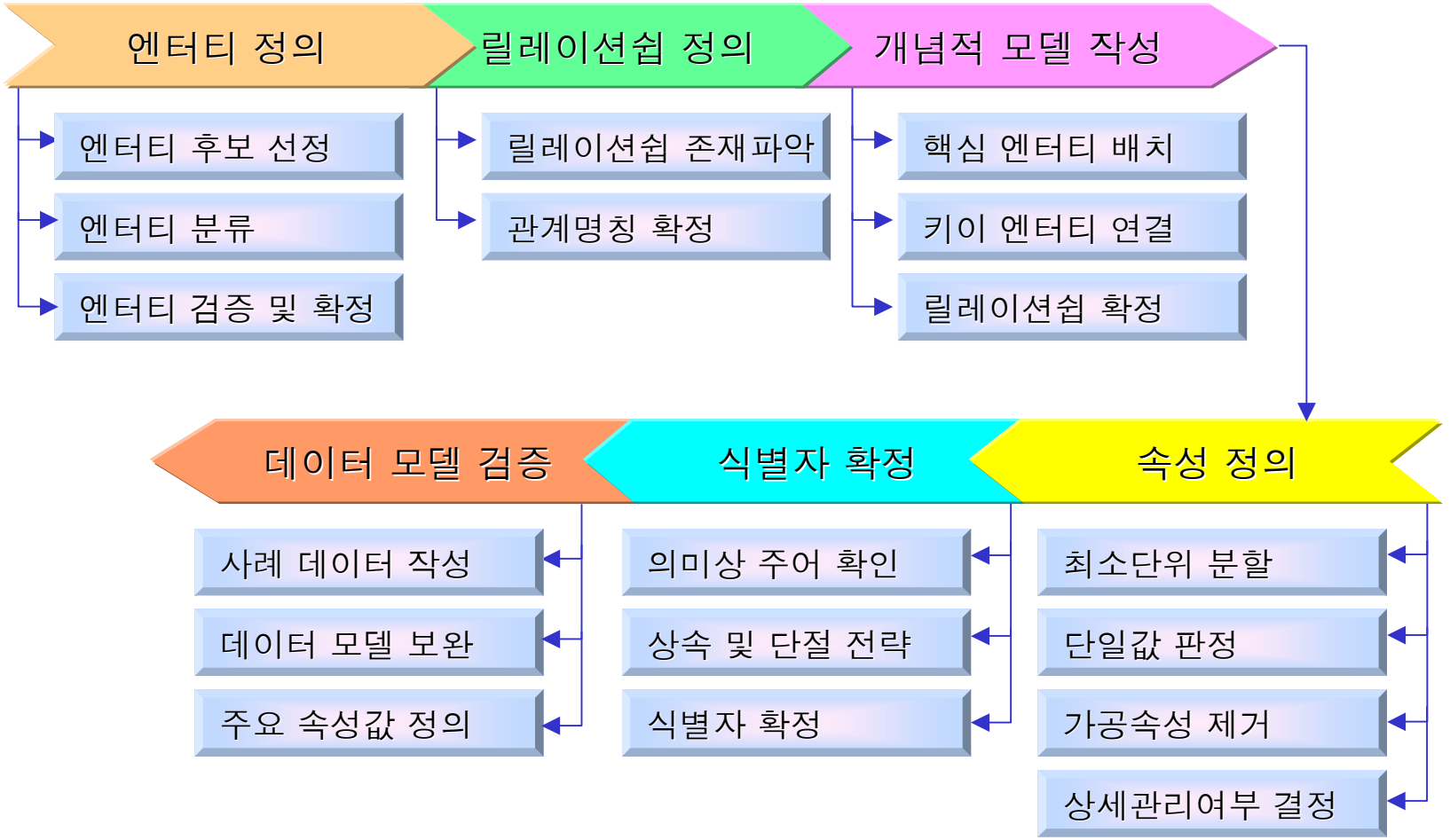
- 원래부터 그렇게 하는 거 아니예요?
- 이렇게 하는 것이 더 좋다고 하던데요
- 우리는 지금까지 항상 이렇게 해 왔는데요

우리에게 과연 객관적이고 구체적인 판단의 잣대가 있는가?

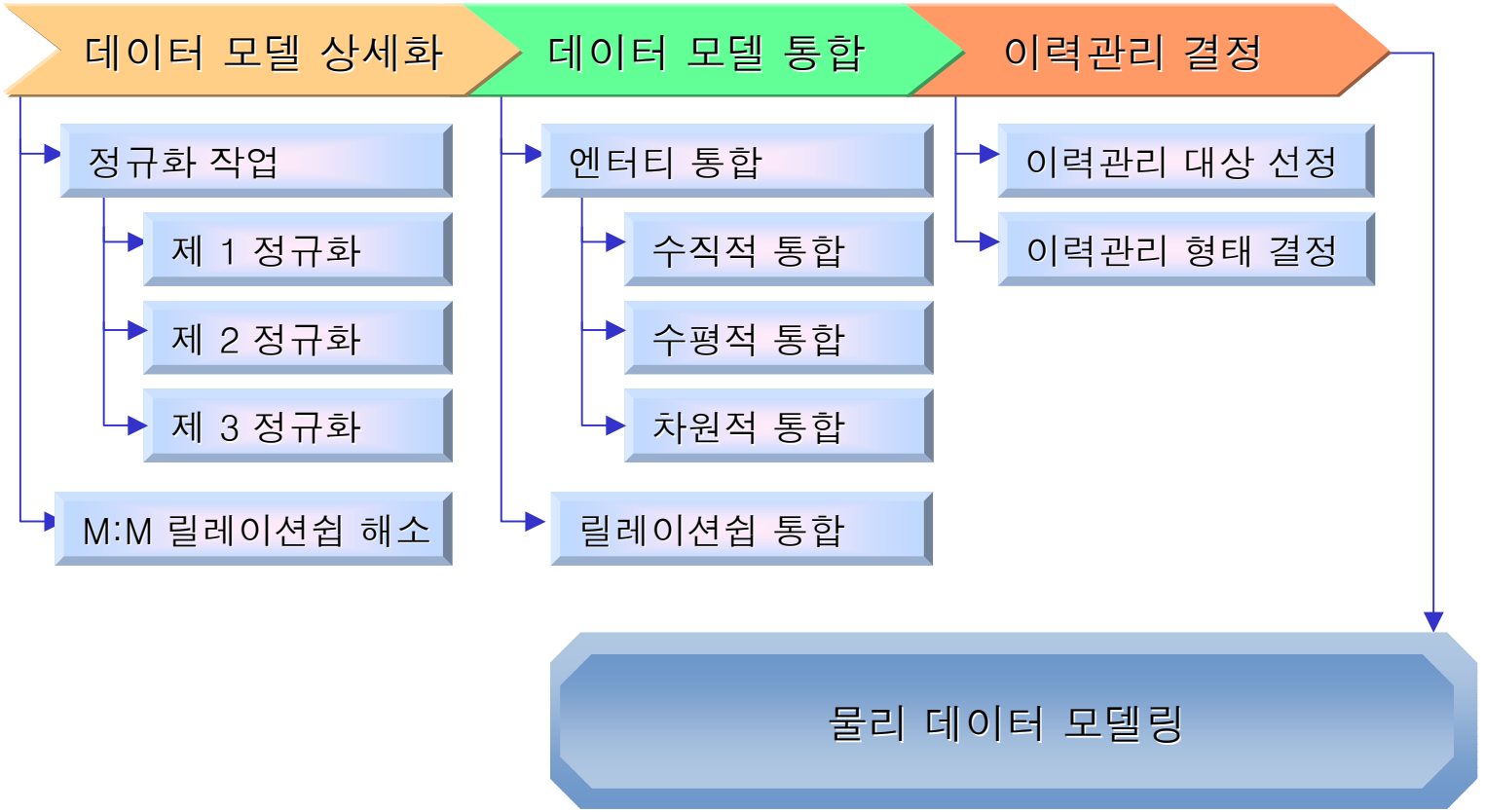
- 가입자, 납입자는 엔터티인가?
- 할아버지와 나는 관계가 있는가?
- 성명이 속성인가?
  성과 명 각각이 속성인가?



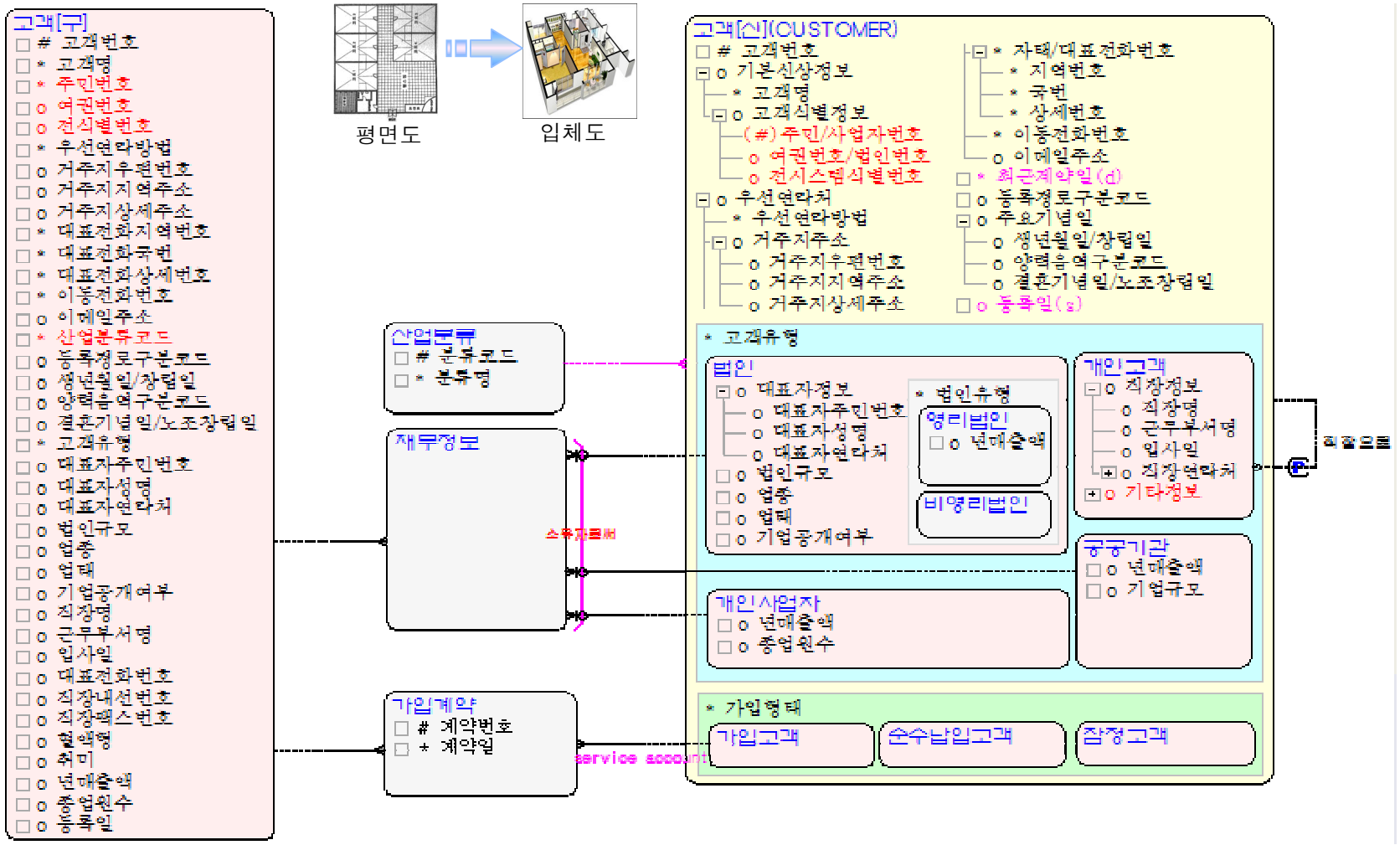
참고. 논리 데이터 모델링 절차



참고. 논리 데이터 모델링 절차

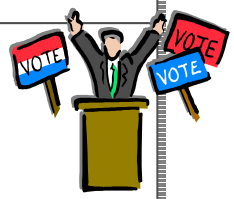


참고. 논리 모델의 구체화


고객[구]
고객번호
* 고객명
* 주민번호
o 여권번호
o 전식별번호
* 우선연락방법
o 거주지우편번호
o 거주지지역주소
o 거주지상세주소
* 대표전화지역번호
* 대표전화국번
* 대표전화상세번호
* 이동전화번호
o 이메일주소
* 산업분류코드
o 등록행정구분코드
o 생년월일/창립일
o 양력음력구분코드
o 결혼기념일/노조창립일
* 고객유형
o 대표자주민번호
o 대표자성명
o 대표자연락처
o 법인규모
o 업종
o 업태
o 기업공개여부
o 직장명
o 근무부서명
o 입사일
o 대표전화번호
o 직장내선번호
o 직장팩스번호
o 혈액형
o 취미
o 년매출액
o 종업원수
o 등록일
평면도
입체도
고객[신](CUSTOMER)
고객번호
o 기본신상정보
* 고객명
o 고객식별정보
(#)주민/사업자번호
o 여권번호/법인번호
o 전시스템식별번호
o 우선연락처
* 우선연락방법
o 거주지주소
o 거주지우편번호
o 거주지지역주소
o 거주지상세주소
* 자택/대표전화번호
* 지역번호
* 국번
* 상세번호
* 이동전화번호
o 이메일주소
* 최근계약일(d)
o 등록행정구분코드
o 주요기념일
o 생년월일/창립일
o 양력음력구분코드
o 결혼기념일/노조창립일
o 등록일(s)
산업분류
분류코드
* 분류명
재무정보
소유자료제
가입계약
계약번호
* 계약일
service account
* 고객유형
법인
o 대표자정보
o 대표자주민번호
o 대표자성명
o 대표자연락처
o 법인규모
o 업종
o 업태
o 기업공개여부
* 법인유형
영리법인
o 년매출액
비영리법인
개인고객
o 직장정보
o 직장명
o 근무부서명
o 입사일
o 직장연락처
o 기타정보
공공기관
o 년매출액
o 기업규모
개인사업자
o 년매출액
o 종업원수
* 가입형태
가입고객
순수납입고객
잠정고객
회원으로

Ⅲ. 엔터티 정의

엔터티 후보 선정



- **엔터티 후보의 수집**
- **엔터티 후보의 식별**
  - 엔터티 후보의 개념정립
  - 관리대상후보의 선정
  - 엔터티 후보의 집합여부 확인

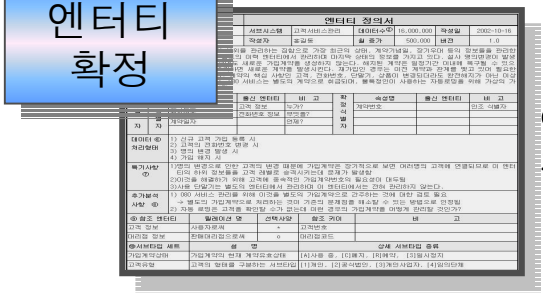
엔터티 형태별 분류

- **우선적용 대상 선별**
- **데이터 영역별 분류**
  - 엔터티의 명확화
  - 데이터 클래스의 정의

엔터티 자격 검증

- **집합의 순수성 확인**
- **동질성 범위 결정**
- **엔터티 명칭 확정**

엔터티 확정



엔터티 정의서 작성

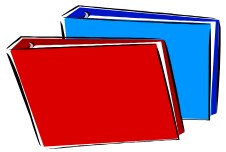
본질식별자 확정

- **키이 엔터티의 본질식별자 정의**
- **행위 엔터티의 본질식별자 정의**

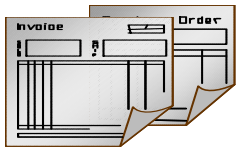
엔터티 형태 확정

- **유사 엔터티 후보와 개념 조정**
- **구체적 서브타입 지정**
- **집합의 독립성 검증**
- **집합의 이합집산**

Ⅲ. 엔터티 정의 – 엔터티 후보의 수집



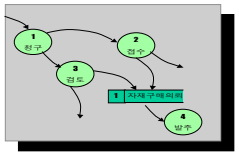
Document



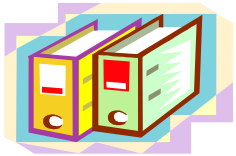
현업 장표

INTERVIEW

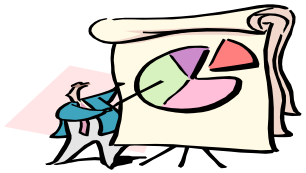
관련전문서적



DFD



타시스템 자료

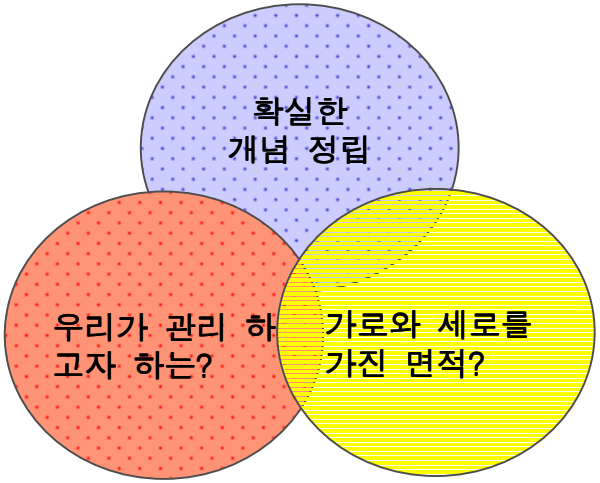


보고서

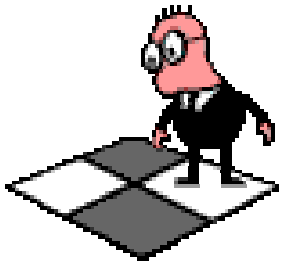
현장조사

Ⅲ. 엔터티 정의 – 엔터티 후보의 식별

- 후보의 최초의 상태는 단지 '단어' 일뿐임
- 애매한 대상을 놓고 검증을 할 수는 없음
  (?는 10도 아니고 10보다 크지도 작지도 않다)
- 사전적 의미가 아니라 어떤 개체로 구성되는 집합?



- 현재 관리하고 있는가?
- 앞으로는 관리해야 하지 않는가?
- 앞으로의 모든 질문의 서두에 포함되어야 함
- 주변의 힘을 빌어 객관적인 증거를 찾을 것



- 가로는 속성
- 세로는 개체
- 선분이 되려면 2개 이상의 점을 가져야 함
- 면적이 되어야 집합, 집합이 되어야 엔터티 후보

Ⅲ. 엔터티 정의 – 엔터티 후보 검증

가로 * 세로 = 면적 = 집합

우리가 관리하고자 하는
가로(속성)가
2개 이상 있는가?

- 현재 관리하는 항목은 ?
- 앞으로는 관리할 항목은?

우리가 관리하고자 하는
세로(개체)가
2개 이상 있는가?

고객

111	710415-1211234	홍길동
112	590518-1411869	박문수
113	651012-2113235	김순자
….	……	……

- 의미상의 주어를 찾아라 !
- 개체 구분을 명확히 하라 !
  - 어떤 경우마다 새로운 개체가 생기는가 ?
  - 절대 종속인가? 상대 종속인가 ?
- 유형별 개체형태를 도식해 보라 !

가로만 선분이면
수평선

2개 이상의 점
을 가져야 선분

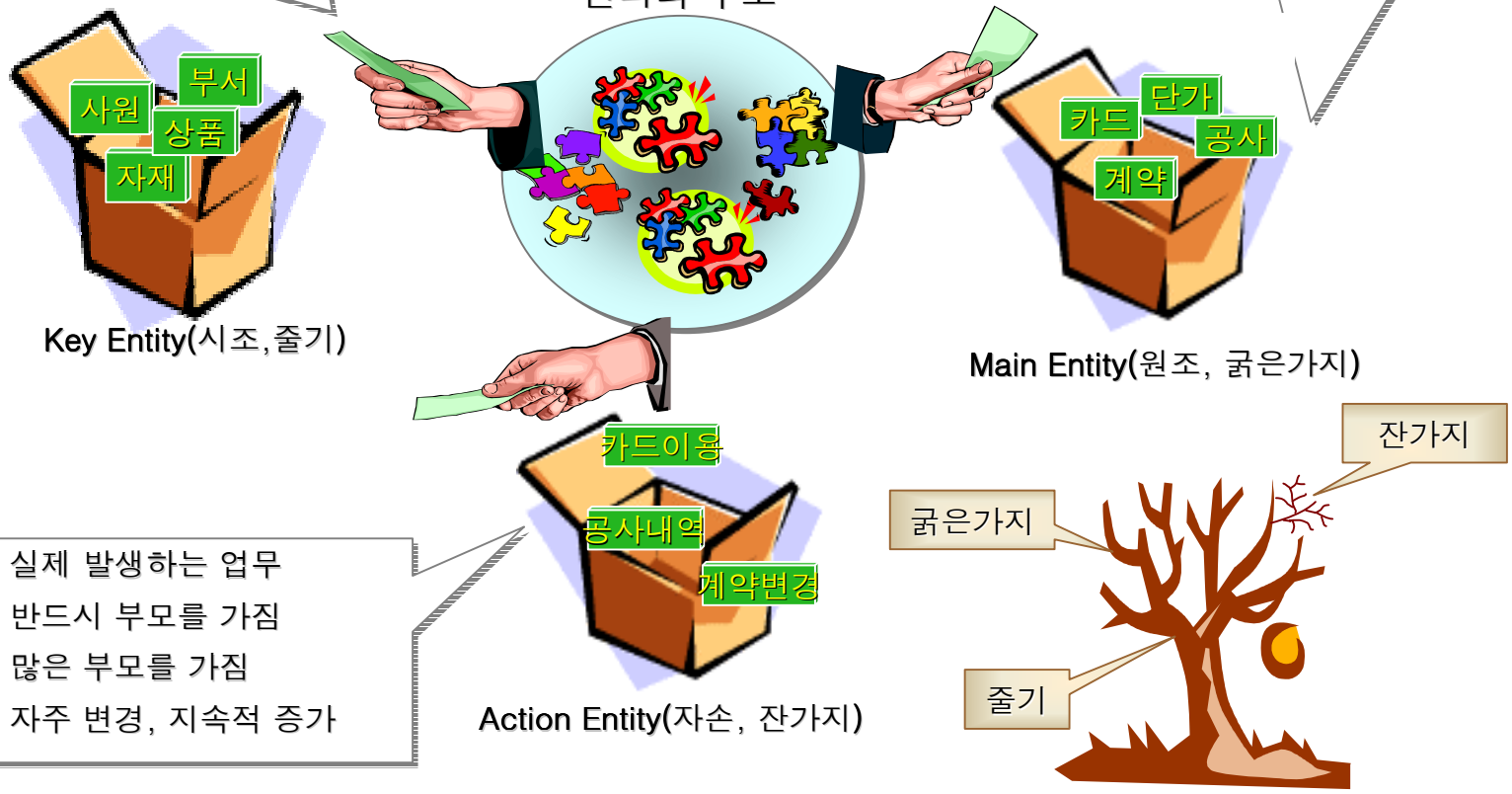
세로만 선분이면
수직선

Ⅲ. 엔터티 정의 – 수집된 엔터티의 분류(우선적용대상)

- 태초부터 창조된 실체
- 잘 변하지 않는 것
- 데이터를 발생의 주체나 목적어
- 자신의 부모를 갖지 아니함

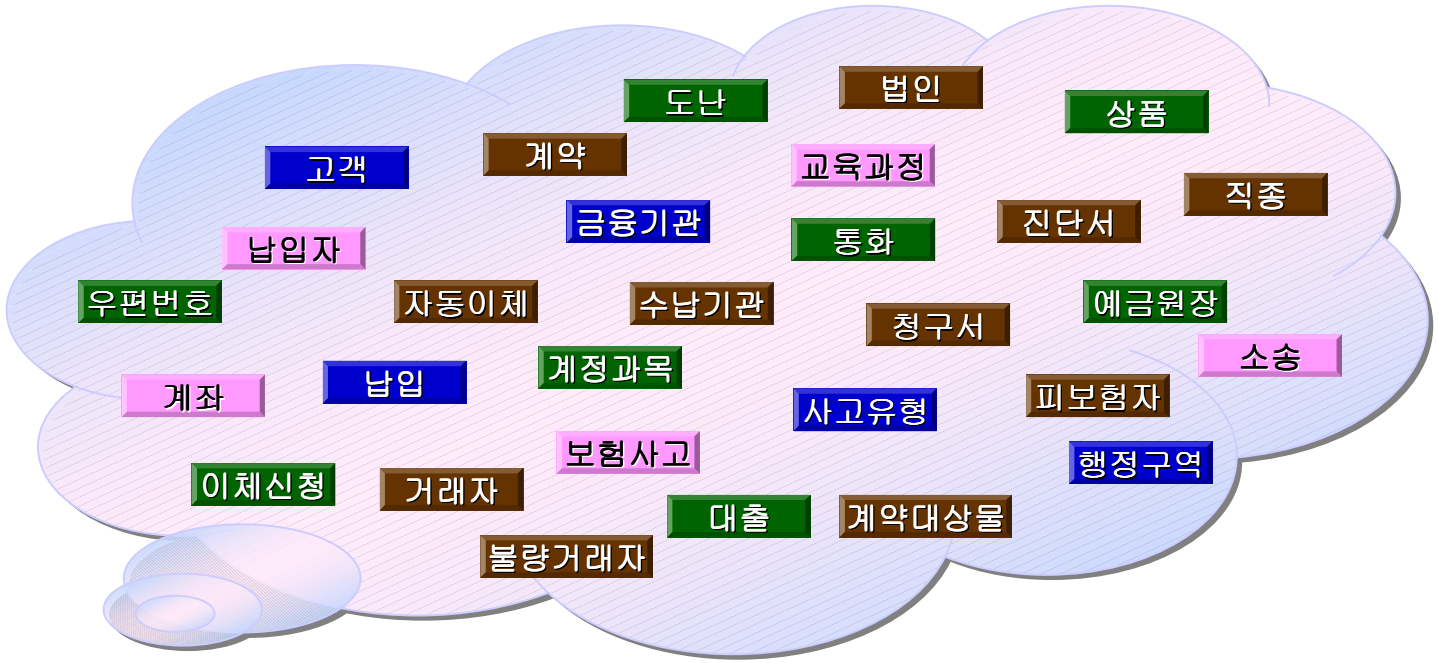
중요한 것부터 먼저 하자!!

- 부모로부터 태어난 실체지만
- 하위에 자손을 거느린 실체
- 많은 자손을 가짐
- 데이터를 발생시키는 주체



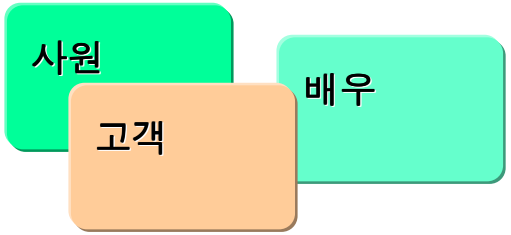
- 실제 발생하는 업무
- 반드시 부모를 가짐
- 많은 부모를 가짐
- 자주 변경, 지속적 증가

Ⅲ. 엔터티 정의 – 실전 연구 (우선적용 대상 분류)



KEY ENTITY	MAIN ENTITY	ACTION ENTITY

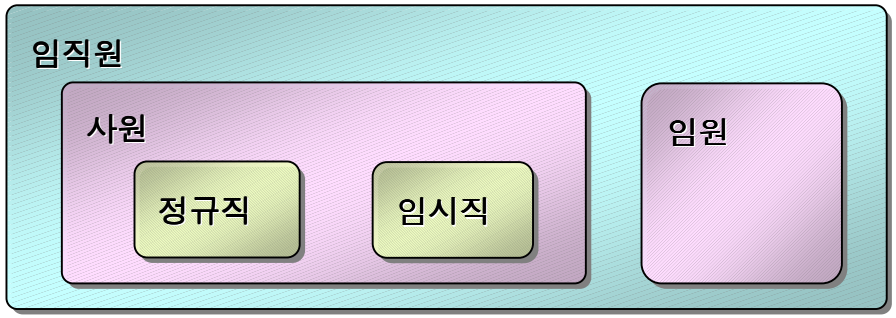
Ⅲ. 엔터티 정의 – 엔터티 명확화 (피상적 認知)

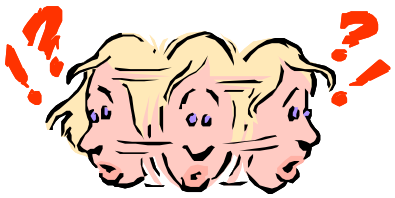


우리는 얼마나 이들을 정확하게 정의하고 있는가?

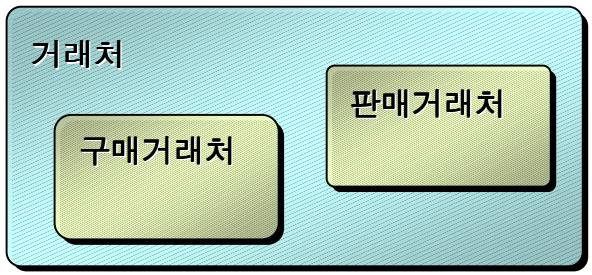
- 임시직, 협력업체, 관계사 직원들도 사원에 포함되는가?
- 연극인도 배우인가? '파트라슈'는 배우인가?
- 외환은행은 우리의 고객인가? '은행' 인가? '거래처' 인가?

- 어느 것을 ENTITY로 해야 하는가 ?





- 어느 것이 ENTITY인가 ?



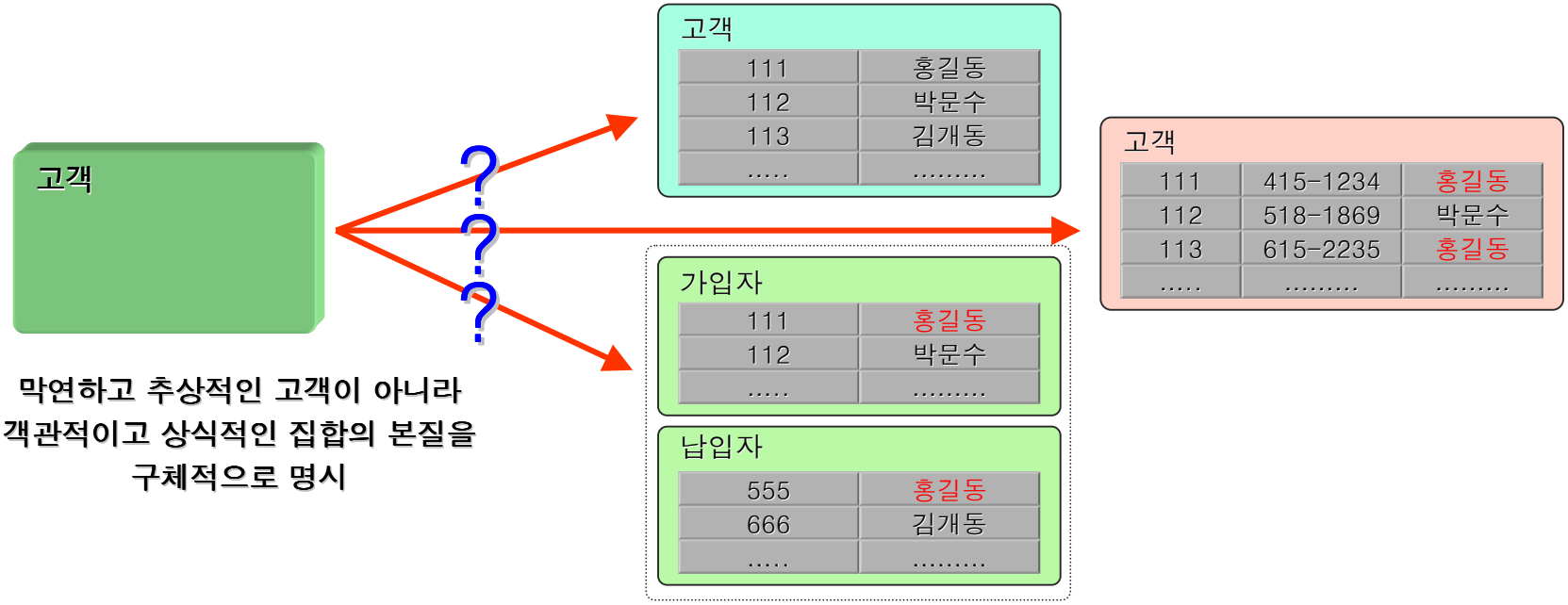
Ⅲ. 엔터티 정의 – 엔터티 명확화 (피상적 인지)

고객

우리가 잘 알고 있다고 생각하는 고객에 대한 질문에 얼마나 확신 있는 답변을 할 수 있는가?

- ✓ 고객은 사람인가? 법인도 고객인가?
- ✓ 법인이 고객이라면 사업자 등록번호를 가지고 있지 않은 것도 법인인가?
- ✓ 우리 상품을 구매(혹은 가입, 계약)하지 않은 것들도 고객인가?
- ✓ 은행 연합회에서 타사에서 관리하는 사람들에 대한 정보를 보유하고 있는데 이들은 우리의 고객인가?
- ✓ 임의 단체(예; 재향군인회, 해병 전우회 등)도 고객인가?
- ✓ 우리가 보유한 고객의 가족정보를 입수했다. 가족은 고객인가?
- ✓ 지킬박사와 하이든씨는 생물학적으로는 하나라고 해서 하나의 고객인가?
- ✓ 만약 여러분이 거래처 정보를 별도로 관리하고 있었다면 거래처는 고객인가?
- ✓ 우리와 체결한 계약을 위해 '보증인' 이 필요한데 이들은 고객인가?
- ✓ 우리 회사 사원들도 계약의 주체가 될 수 있는데 사원은 고객인가?
- ✓ 회원으로 가입했다가 탈퇴 후 다시 가입하면 고객은 하나인가? 둘인가?
- ✓ 삼정그룹은 수많은 계열사를 가지고 있지만 그룹이라는 단체는 공식적인 사업자 등록번호가 없는 단체이다. 이들은 고객인가?
- ✓ 주민번호나 사업자번호를 비교해서 기존 고객인지 신규 고객인지를 구분하고 있는데 만약 번호의 오류로 인해 이미 2건이 존재하고 있었다면 잘못된 한 건은 고객인가? 아닌가?

Ⅲ. 엔터티 정의 – 엔터티 명확화 (구체화 요구)

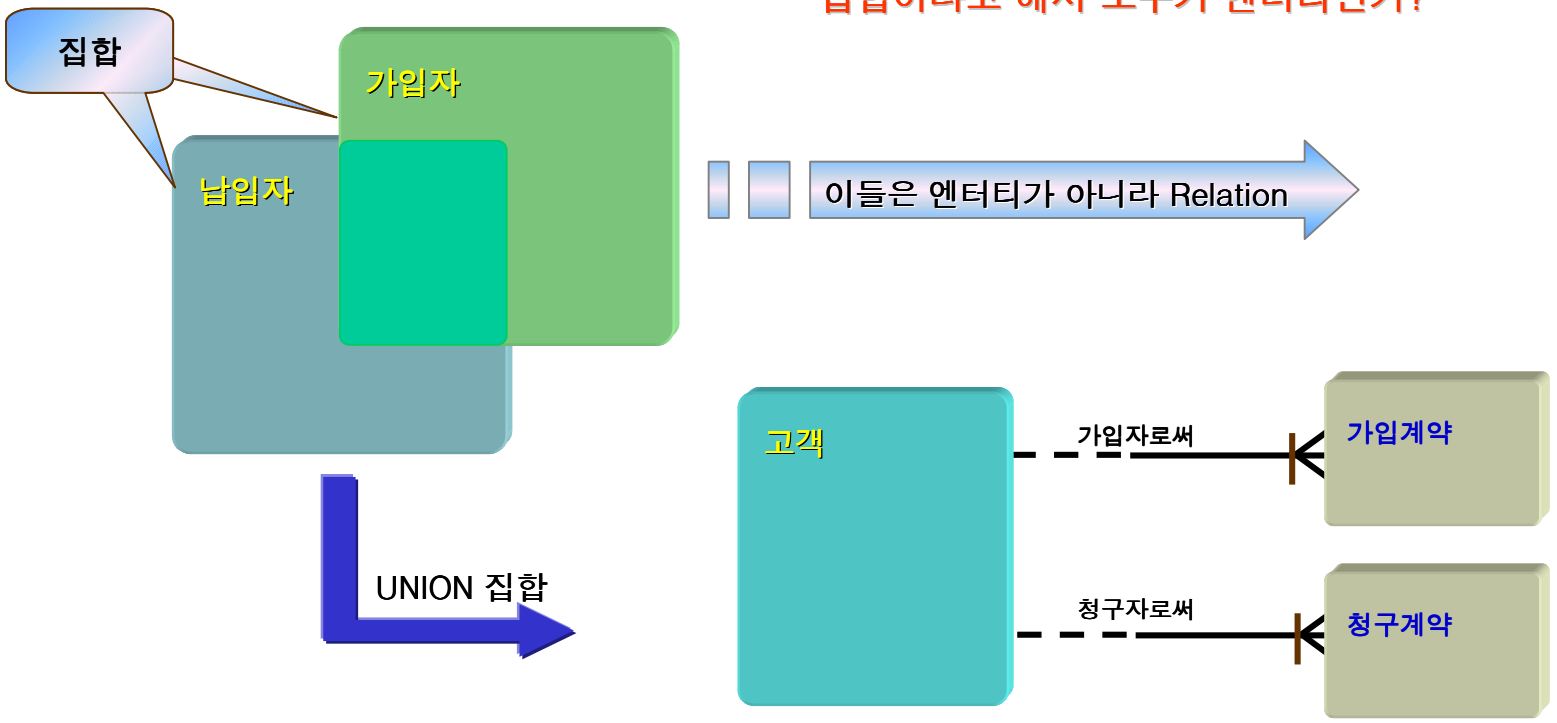


- 매우 일반적인 용어로 정의되어 있더라도 오류는 발생할 수 있음
- 반드시 어떠한 요소들이 포함되는지를 명확하게 정의
- Entity가 명확하지 않으면 추후 많은 혼란 발생
  - ✓ 관계를 맺고 있는 다른 Entity와 관계형태의 혼란
  - ✓ 배타적 논리합 관계 다수 발생
  - ✓ 부모를 가지지 않는 데이터 발생
  - ✓ Entity간의 부정확한 관계, 매우 복잡한 관계로 나타나게 됨
  - ✓ 추후 애플리케이션에서 복잡한 IF 처리를 해야 함

ENTITY 정의가
명확해졌으면 가장 내용에
부합되는 명칭 부여

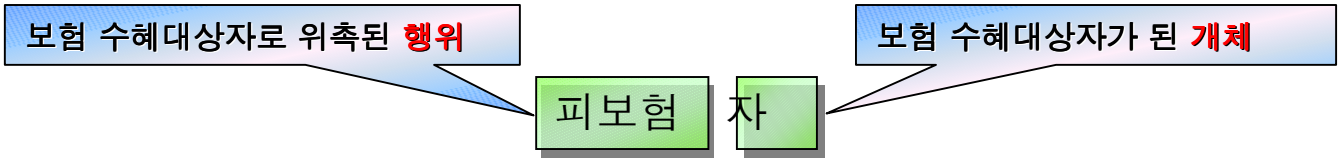
Ⅲ. 엔터티 정의 – ENTITY의 자격 검증

가로 * 세로 = 면적 = 집합



홍길동이가 가입자도 되고 청구자도 된다고 홍길동이가 두 명인가?

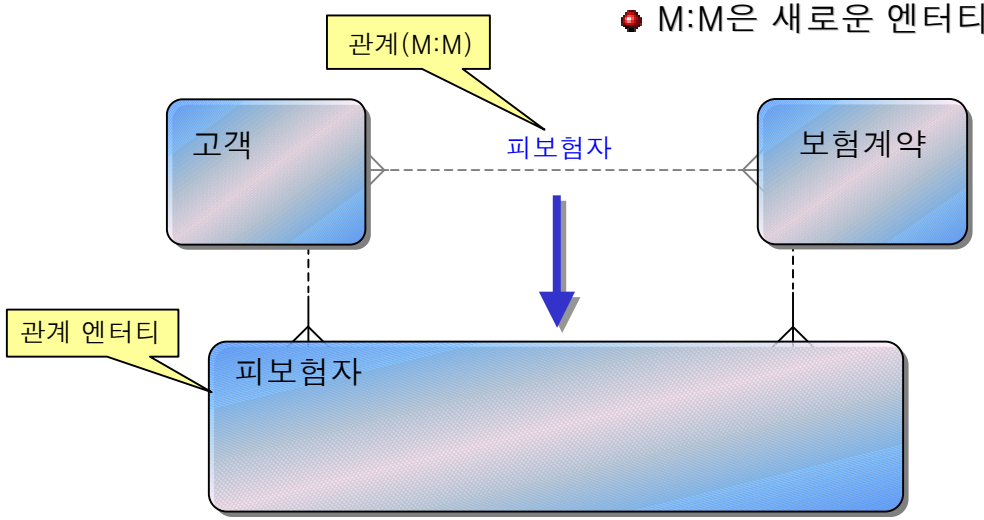
Ⅲ. 엔터티 정의 – 집합의 순수성 검증



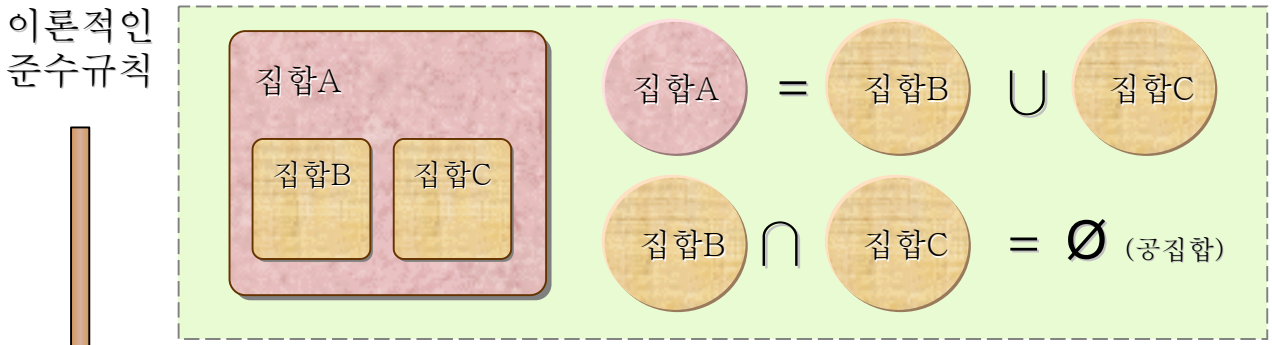
행위와 개체의 결합이므로 분명히 릴레이션쉽

RELATION 이지만 M:M이면 ENTITY ➔ RELATION & ENTITY
이 엔터티를 RELATION ENTITY라 부른다

- 1:M이나 1:1은 RELATION으로 존재
- M:M은 새로운 엔터티로 생성



Ⅲ. 엔터티 정의 – 구체적인 서브타입 지정



그러나 아직 그것을
따질 게재가 아니므로

어떤 것으로
구성되었느냐의
측면에서

사원 정보
사원번호
* 사원명
* 성별(남,여)
o 주소

서브타입세트1

* 사원구분

내근사원	설계사	대리점
o 직급	o 자격등급	o 사업자번호
* 호봉	o 간부급수	* 대표자명
o 입사경로		o 대리점등급

구성 / 유형

대리점 사원
직원번호
* 성명
* 주민등록번호

가능하다면
여러 각도에서
조망

서브타입세트2

* 직무

경영자	관리자	사무직	생산직	...
임원구분				

역할 / 행위

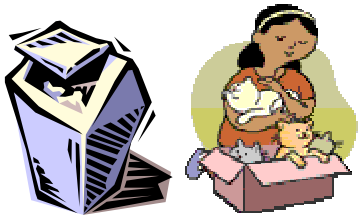
어떤 행위를
하고 있느냐의
측면에서

Ⅲ. 엔터티 정의 – 엔터티 형태 결정의 대원칙

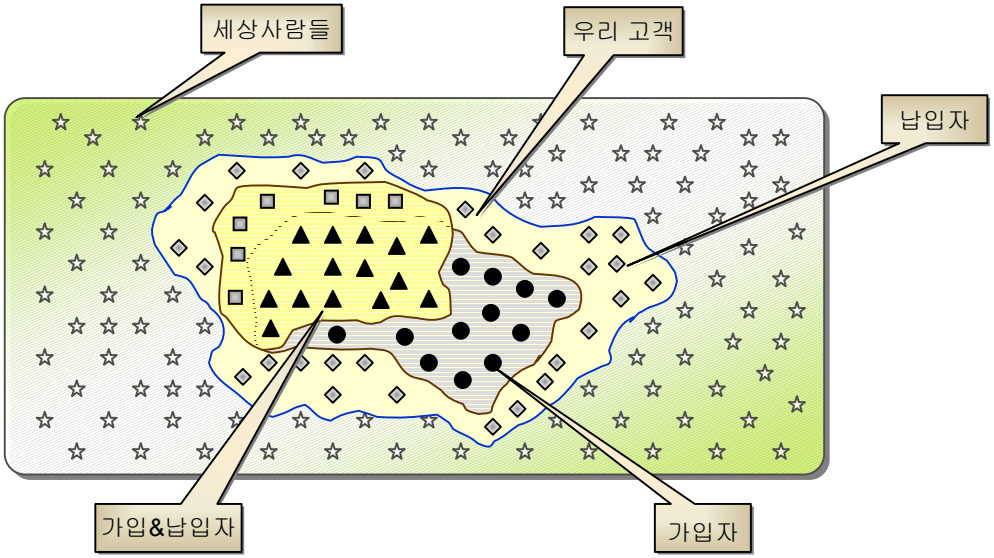
어느 것이 우리가 관리 해야 할 실체인가?

논리적으로 존재하는 집합은 매우 다양

잣대가 필요



버릴 집합 취할 집합



엔터티의 조건을 만족했더라도 크게 하나로 할 것인지, 필요에 따라 여러 개로 할 것인지 결정하는 일은 매우 어렵다

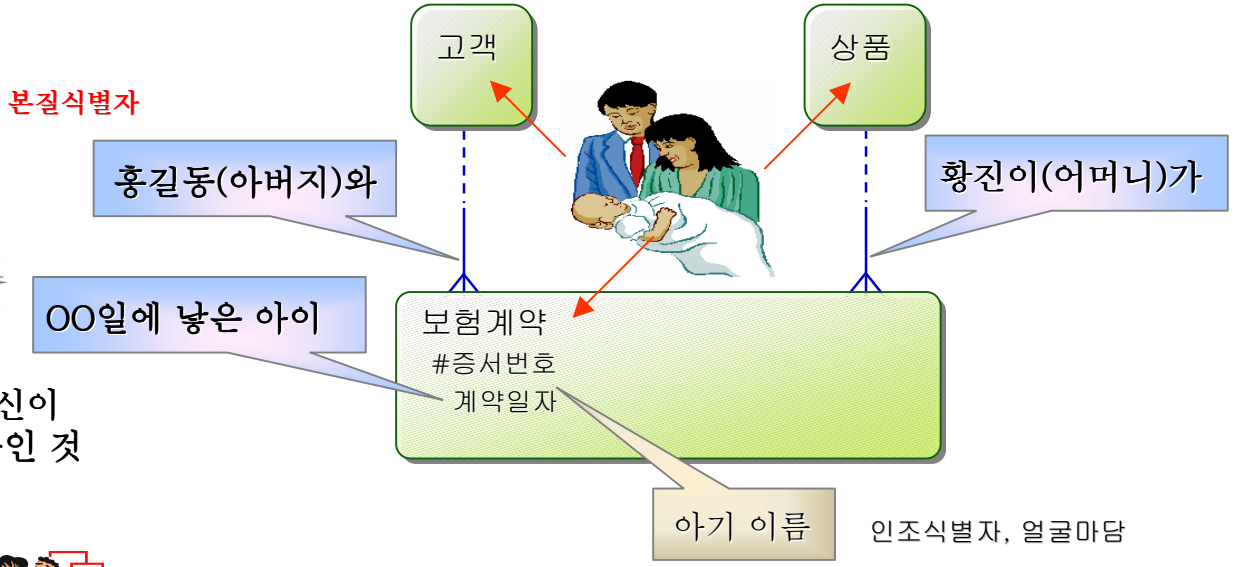
Ⅲ. 엔터티 정의 – 의미상의 주어(본질 식별자)

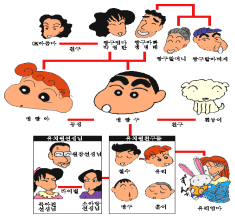
그것, 저것,
거시기?

진주어(眞主語) : 내용상의 실질적인 주어,
가주어(假主語) : 지시대명사, 문법상의 주어

의미상의 주어

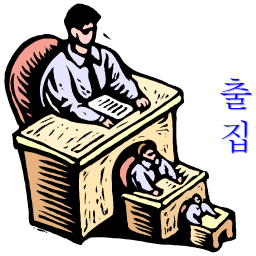
본질 식별자란 그가 없다면 자신이
결코 태어날 수 없는 절대 종속인 것





이름을 확정하는 것보다
자신의 출생의 비밀을
밝히는 것이 훨씬 우선적

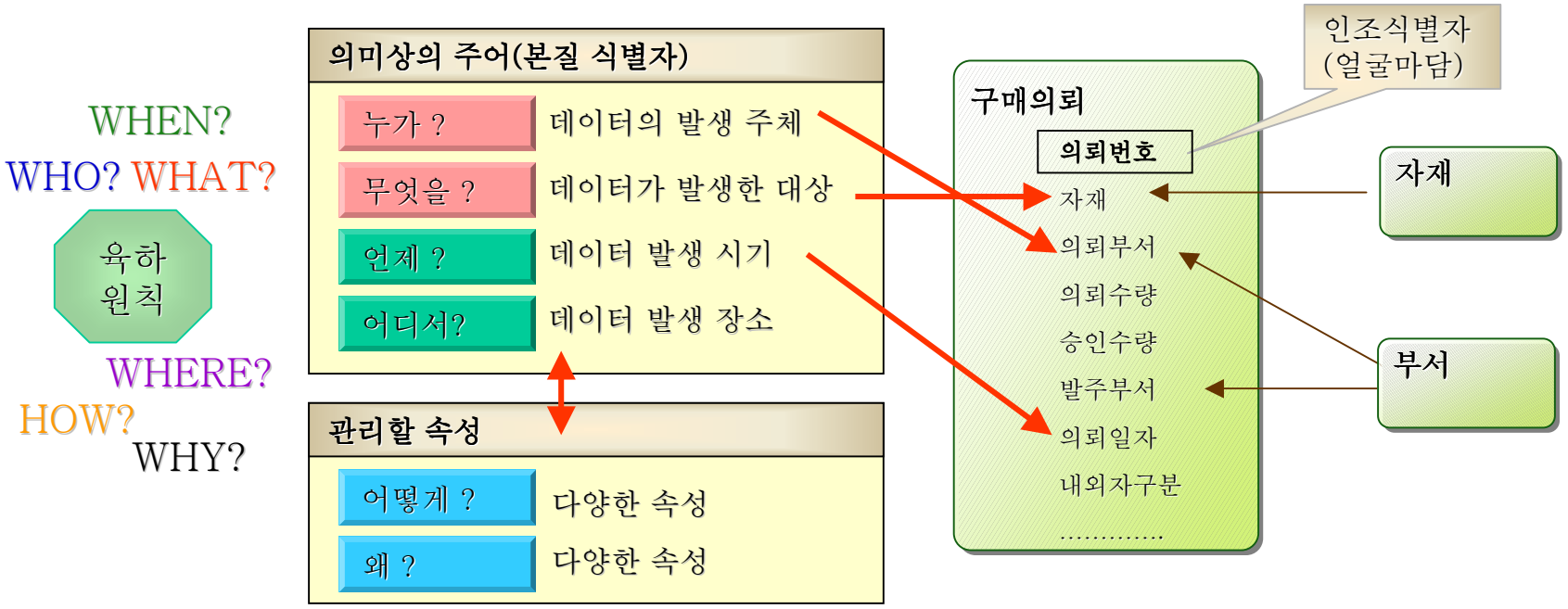
본질 식별자를 정확히 밝혀내는 것은
엔터티 정의를 명확히 하는 중요한 과정



출생의 비밀을 모르고서는
집합이 명확해질 수 없다

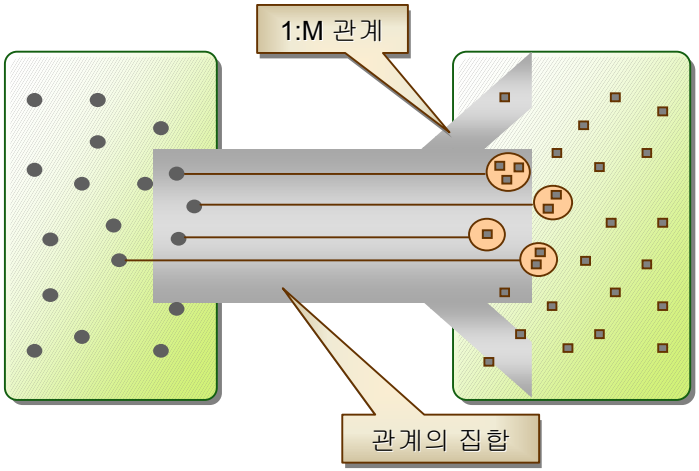
Ⅲ. 엔터티 정의 – 행위 엔터티의 상향식 접근

- 부모없이 태어난 자손은 없다.
- 의미상의 주어란 ? ENTITY 탄생에 직접적인 역할을 한 속성들
- 인조속성이 아닌 의미상의 주어(본질식별자)를 찾아라.
  - ✓ 절대 종속인가? 상대 종속인가?
  - ✓ 직접 종속인가? 간접 종속인가?
- 놓쳐 버린 Entity가 있더라도 각 단계에 충실하면 결국에는 다시 나타난다.
- 그러나 Key Entity를 놓치면 시행착오를 겪게 된다.

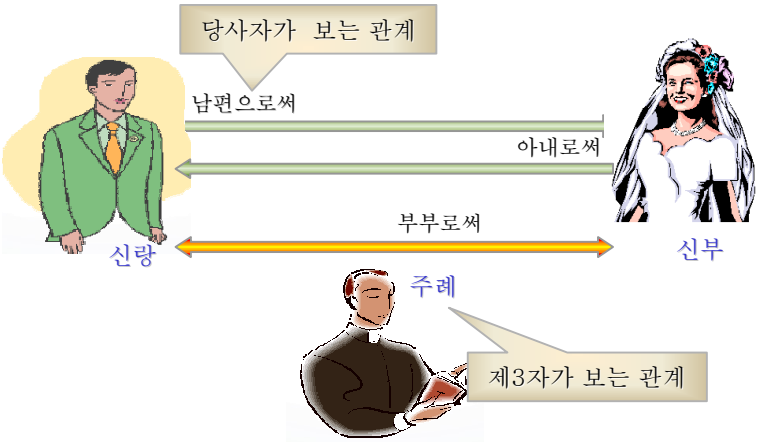


Ⅳ. 관계 정의 – 관계의 진정한 의미

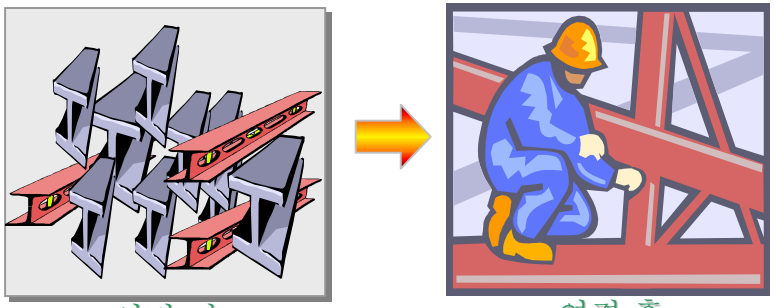
- 릴레이션쉽도 집합이다.



- 관계에는 당사자 간의 관계와 제3자가 보는 관계가 있다

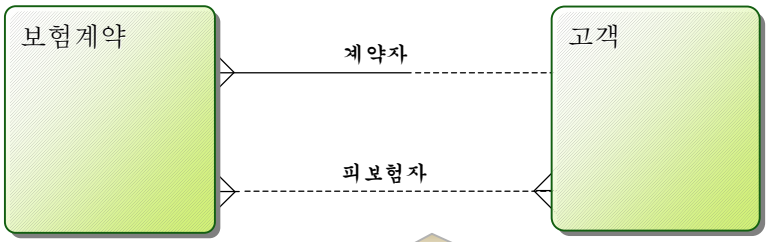


- 간접 릴레이션쉽은 무수히 존재하지만 직접 릴레이션쉽은 많지 않다



연결 전에는 어느 것과도 연결될 수 있을 것처럼
보이지만 연결 후에 보면 단지 몇 개와 직접 연결될 뿐임

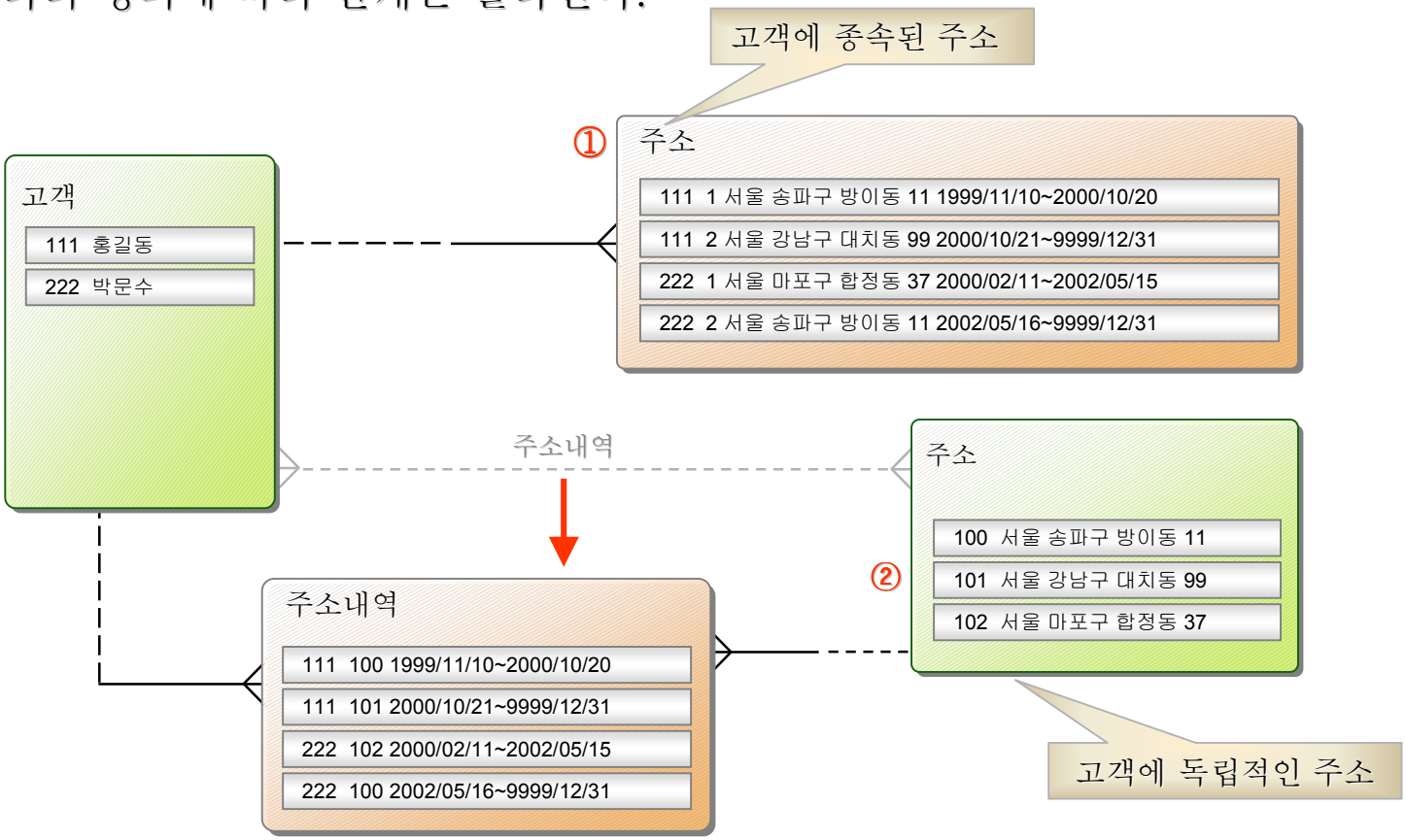
- 관계란 정의하기에 따라 다양하게 존재한다



정의하기에 따라 여러 종류의
릴레이션쉽이 존재

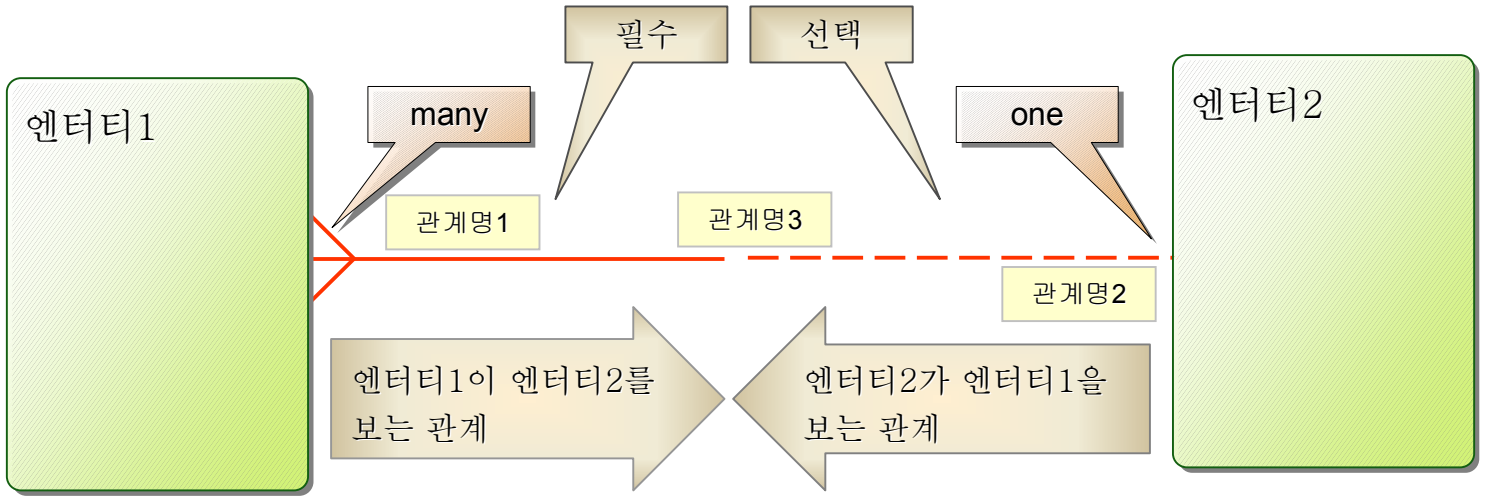
Ⅳ. 관계 정의 – 관계의 진정한 의미

- 엔터티 정의에 따라 관계는 달라진다.



'주소' 엔터티의 정의에 따라
전혀 다른 릴레이션쉽 발생.

Ⅳ. 관계 정의 – 릴레이션쉽의 표현

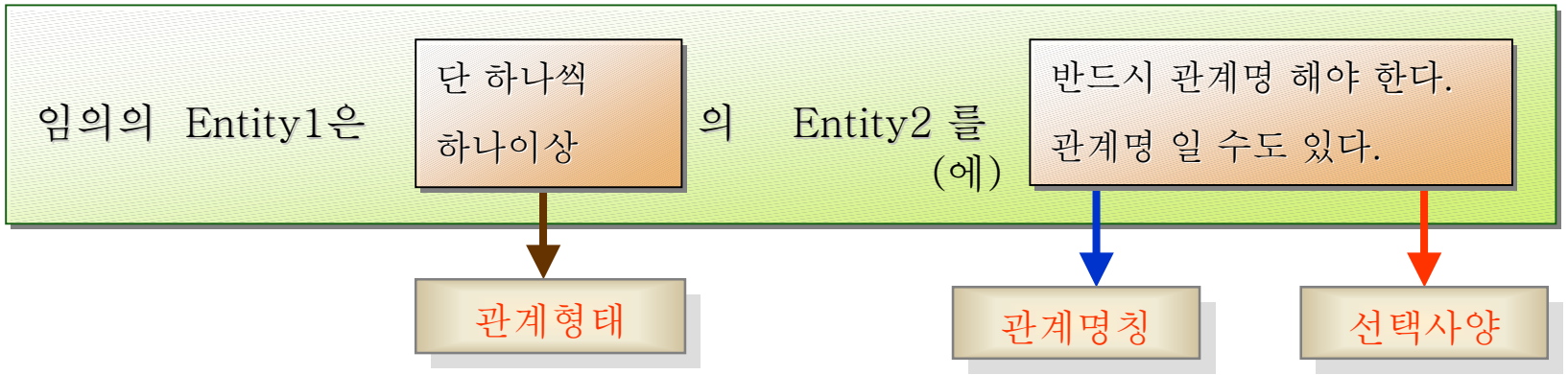


- 관계형태(degree)
  - Many : 까마귀발가락 (⟩—)
  - One : 한 줄 실선 (—)
- 선택사양(optionality)
  - Mandatory : 실선 (———)
  - Optional : 점 (- - - -)
- 관계명칭
  - 제1자로 표현
  - 제3자로 표현

- 관계명은 가능한 제1자의 입장에서 규정
- 다른 관계와 차별적성 있게 구체적으로 지정
- 제3자의 입장에서 지정하는 것이 분명한 경우도 있음
- 각 엔터티의 역할보다는 관계의 내용을 분명히 하고자 하는 경우에는 제3자 입장에서 지정
- 관계를 규명할 때는 반드시 제1자 입장에서 생각할 것

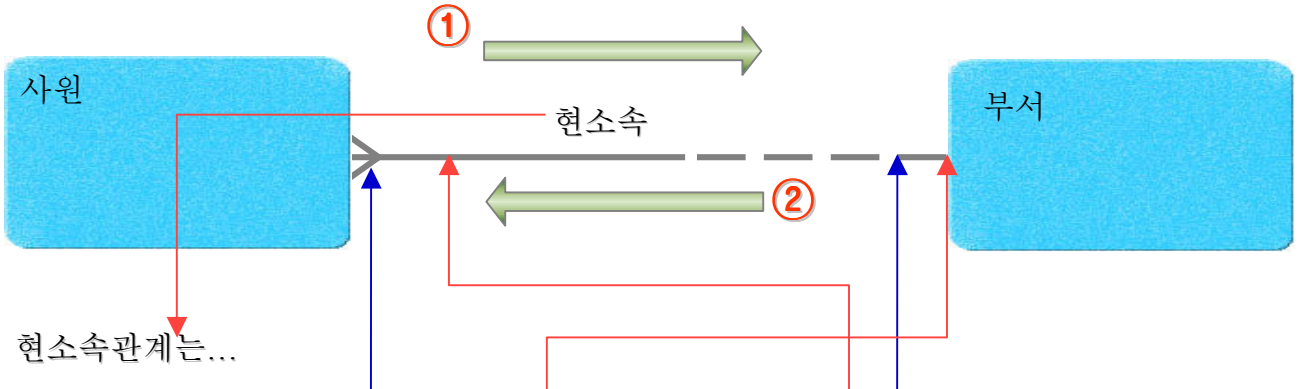
Ⅳ. 관계 정의 – 구체적인 관계를 규명하는 절차

- 두개의 ENTITY나 그 자신과의 특정관계를 양방향으로 표현
- 현재의 관계나 장래 유용한 관계만 한정적으로 표시
- 각 방향의 관계에는
  - ✓ 관계의 명칭(부사형으로)
  - ✓ 선택사양(Optionality)
  - ✓ 관계형태(Degree) 를 표시

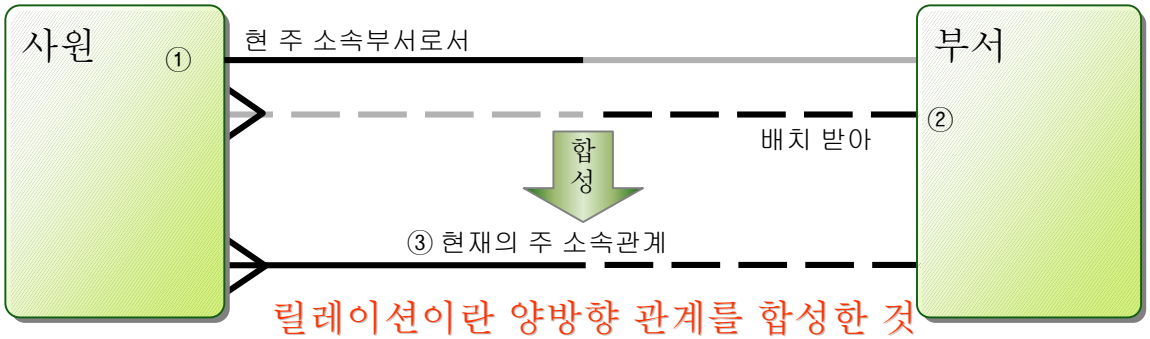


- 관계명은 구체적이어야 한다. (속하여, 참조하여 등은 피할 것)
- 주는 쪽(One) : 특정 집합만 관계를 가질 때는 해당 집합을 표현하는 것이 좋음
- 받는 쪽(One or Many) : 최대한 관계 내용을 구체적으로 표현할 것
- 전후 상황을 보아 관계가 너무 분명한 경우나 지극히 일반적인 경우는 관계명을 생략할 수도 있으며 제3자의 관계로 표현해도 무방함
- 보편타당성을 유지 (특별한 설명이 없더라도 이해할 수 있는 일반적이고 객관적인 용어 사용)

Ⅳ. 관계 정의 – 관계규명의 실전적 접근 방법

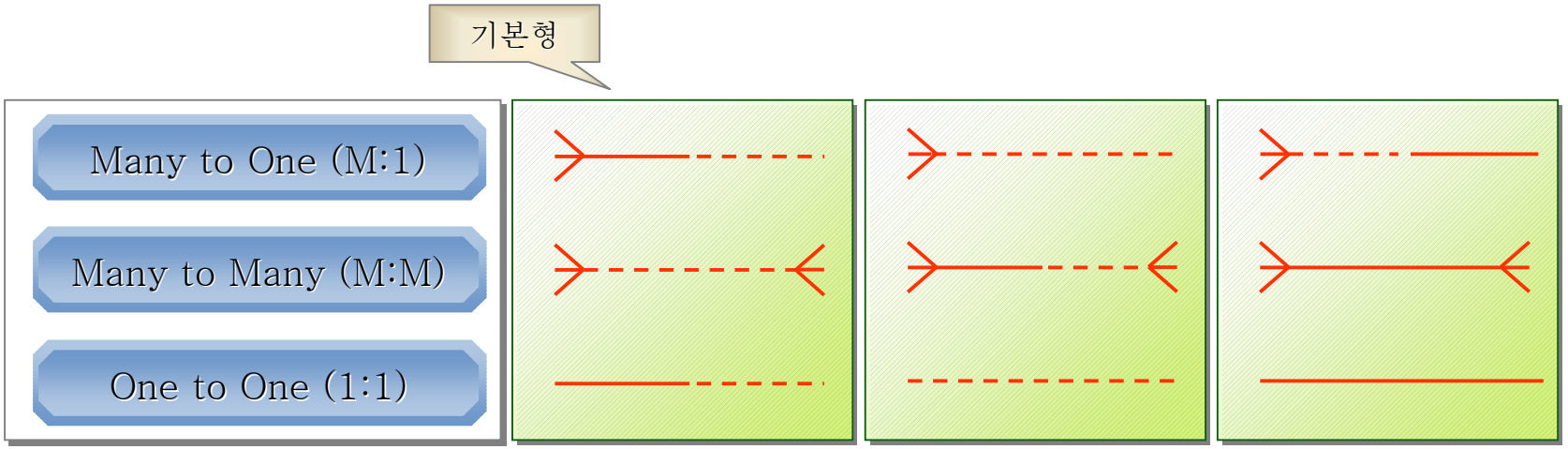


	주어부	관계비	목적부	선택사양
①	각각의 사원은	단 하나의	부서를	반드시 가져야 한다
②	각각의 부서는	하나이상의	사원을	가질 수도 있다



릴레이션이란 양방향 관계를 합성한 것

Ⅳ. 관계 정의 – 관계형태(relationship type)

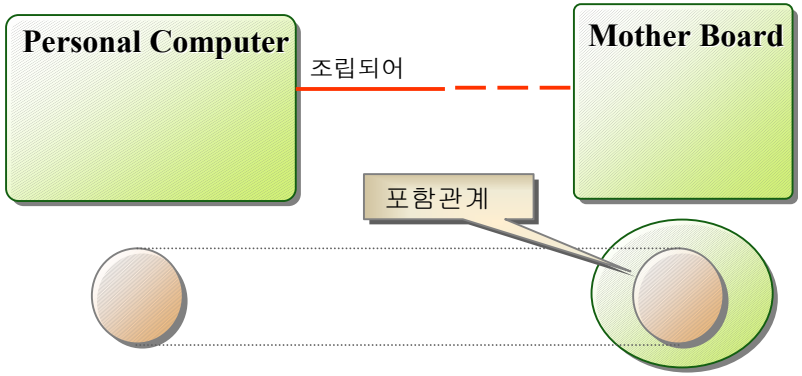


- 대부분은 기본형이다 (70~80%)
- 먼저 관계형태부터 확인한 후 기본형을 그린다
- 당사자 간의 구체적인 선택사양을 검증한다
- 구체적 검증 시에 관계형태가 달라질 수도 있다

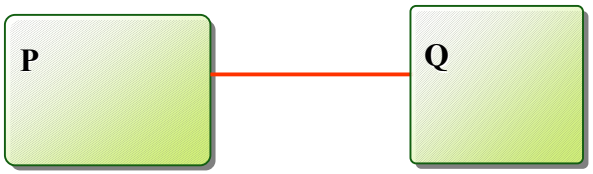
Ⅳ. 관계 정의 - 관계형태(1:1 관계)

- 양쪽 방향 모두 단 하나씩(one and only one)
- 드물게 발생되는 형태
- 양방향 모두 반드시(must be) 가 되는 경우는 아주 드물다
- 1 : 1 관계는 실제로는 동일한 ENTITY 일 경우가 많다
- 1 : 1 관계가 많이 나타난다면 Entity가 명확하게 정의되지 않았음을 의미함

필수-선택 관계



필수-필수 관계

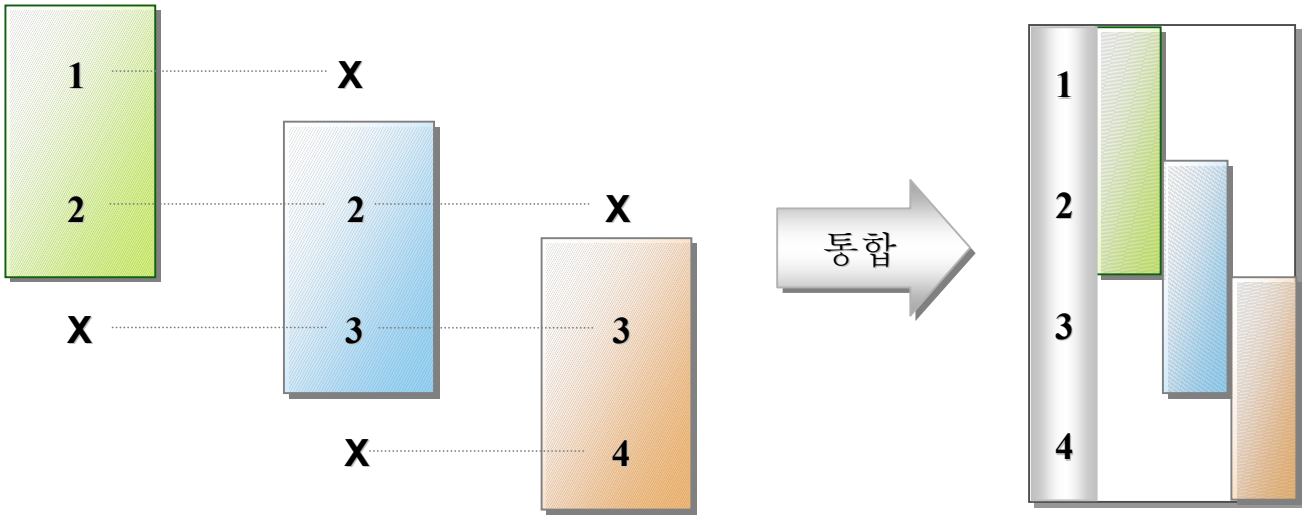


$P \Rightarrow Q$이고, $Q \Rightarrow P$이면, $P \Leftrightarrow Q$

필요충분조건을 만족하면 서로 동치(同値)

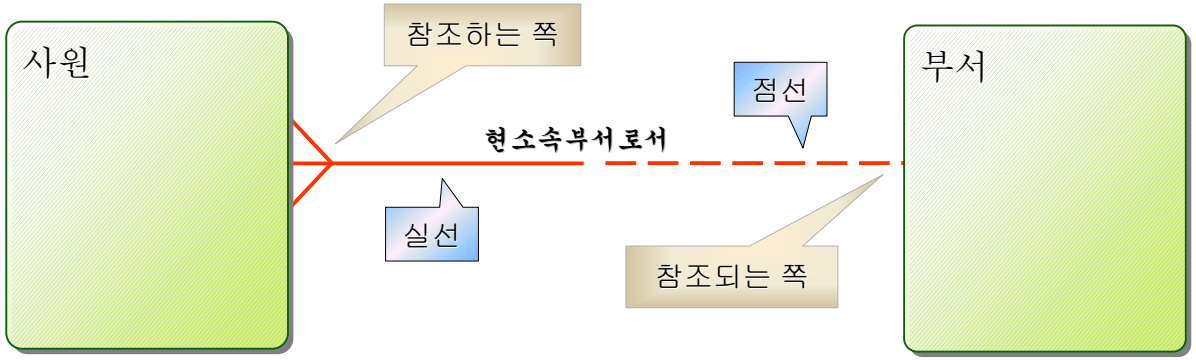
Ⅳ. 관계 정의 – 관계형태(1:1 관계)

필수-필수 관계



- ✓ 수행속도를 염려해 지나친 수직분할을 함으로써 주로 발생
- ✓ 양측 Outer Join이 발생함
- ✓ 애플리케이션이 복잡해지고 수행속도가 나빠짐
- ✓ 데이터 일관성이 훼손될 수 있음
- ✓ 이들이 자식 엔터티를 가지면 더 복잡해짐

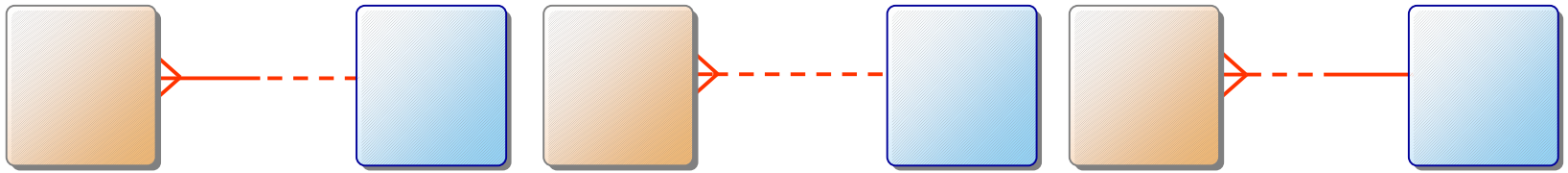
Ⅳ. 관계 정의 – 관계형태(M:1 관계)



현실세계는 수평관계보다 수직관계가 더 많다

수평관계는 이미 같은 엔터티가 되었을 가능성이 있음

- 한쪽 방향은 하나이상(one or more)
- 다른 방향은 단하나씩(one and only one)
- 가장 일반적인 형태



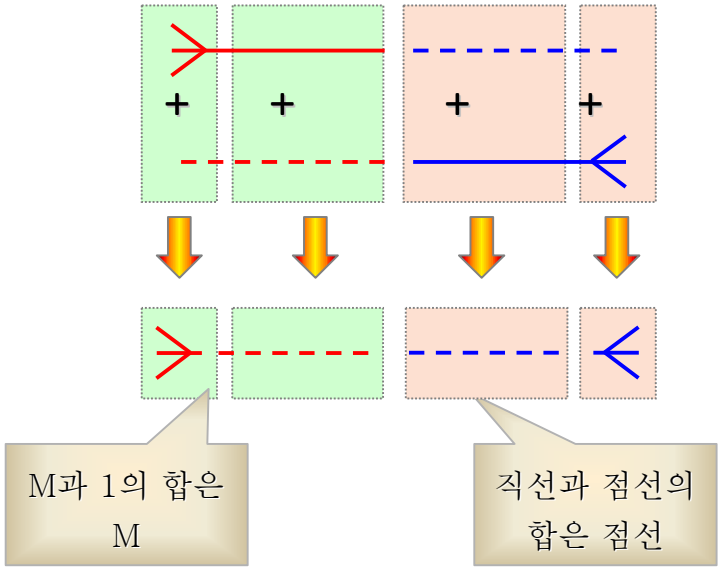
- ✓ 자식은 부모 없이 함부로 탄생불가
- ✓ 부모는 경우에 따라 자식을 갖지 않을 수도 있다
- ✓ 본질식별자라면 당연히 이렇게 되어야 상식적이고 보편 타당한 형태

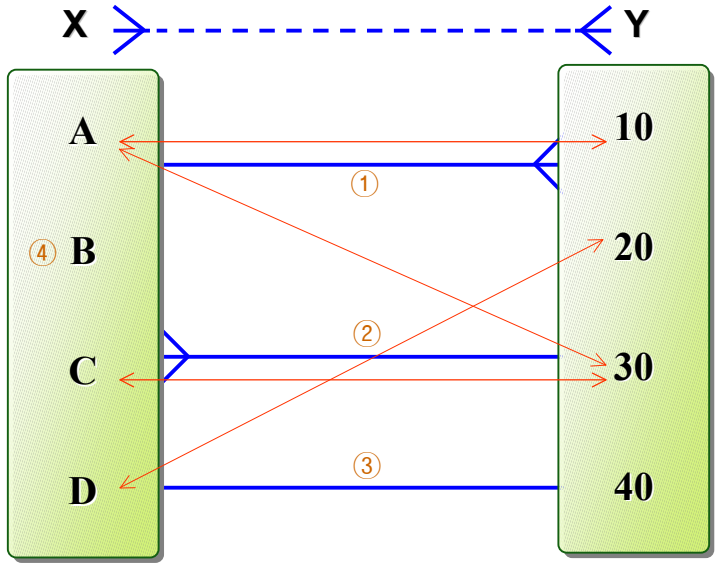
- ✓ 참조할 필요가 있는 것들만 연결
- ✓ 주는 쪽은 가만히 있고 받는 쪽에서 가져가는 것이 일반적
- ✓ 일반적인 참조인 경우에는 이러한 형태가 더욱 보편적

- ✓ 자식이 필요에 따라 그룹핑하여 부모를 옹립할 때 발생되는 형태
- ✓ 부모는 자식이 옹립해 주지 않으면 발생할 수 없다
- ✓ 실무에서 가끔은 발생하는 형태

Ⅳ. 관계 정의 – 관계형태(M:M 관계)

- ✓ 양쪽 방향 모두 하나이상(one or more)
- ✓ 자주 발생되는 형태
- ✓ 단계적으로 분할된다
- ✓ M:M은 아직 덜 풀려진 상태





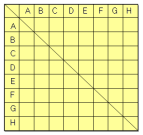
① A의 입장에서 10,30을 보는 관계

② 30의 입장에서 A,C를 보는 관계

③ D입장에서 20을 보는 관계

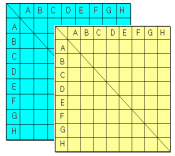
④ B, 40은 관계를 가지지 않는 경우

Ⅳ. 관계 정의 – 릴레이션쉽의 정의절차



✓릴레이션쉽 매트릭스(matrix)를 활용한다.

✓엔터티 수가 많다면 분할해서 정복한다. (divide & conquer)

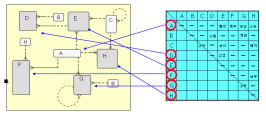


✓제3자 입장에서 직접관계의 유무(有無)만 판단한다.

✓관계가 있다고 판단된 것들에 대해 구체적인 관계명을 정의한다.

✓작성된 매트릭스에서 활동성이 강한 엔터티를 찾아 구도를 잡는다.

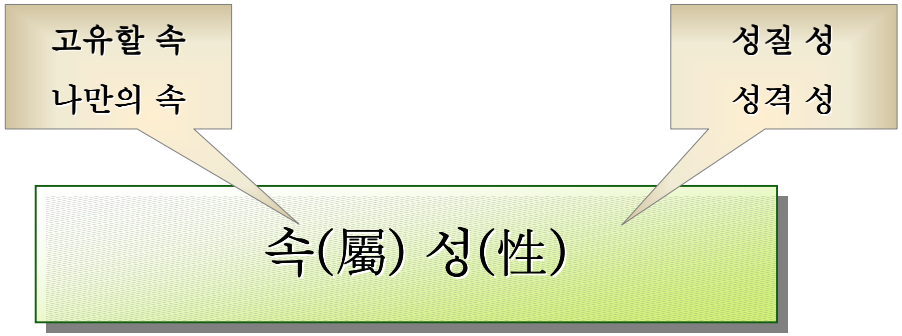
✓나머지 엔터티들을 적절한 위치에 넣고 기본적인 릴레이션을 맺는다.



✓당사자 입장에서 릴레이션쉽을 구체적으로 검증하여 확정한다.

✓표현할 관계명을 최종적으로 확정한다.

Ⅴ. 속성 정의 – 속성(ATTRIBUTE)의 어원(語原)분석



사물의 본질을 이루는 고유한 특성이나 성질

1) 원자 단위냐?

본질

본질 = 근본, 원천
중간 과정의 것이 아니라
그 보다 더 근원적인 것,
즉, 분자(分子)가 아니라 원자(原子)

고유

2) 유일하게 존재?

고유 = 단 하나
분자인 수소(H2)와
물(H2O)에는 수소가 2개씩
있지만 원자는 단 한가지씩일 뿐

3) 근원 값이냐?

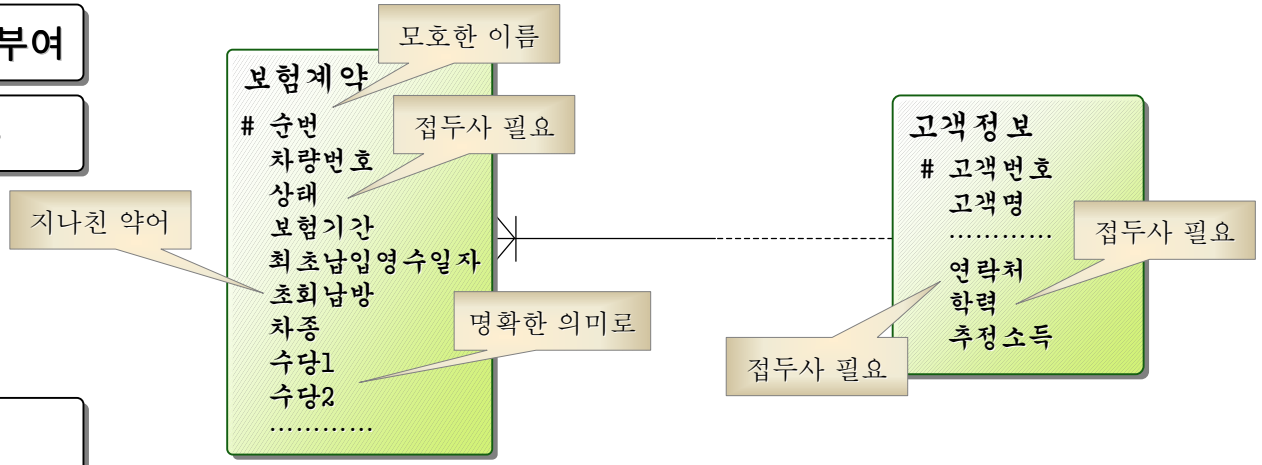
혼자서도 독립적인 의미를 갖는가?

남의 도움 없이 자기만의 독자적인 존재가치를 가지고 있는가?

Ⅴ. 속성 정의 – 속성 정의 시의 유의사항

의미가 명확한 속성명칭 부여

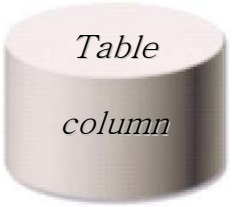
최대한 복합명사 사용



표준 단어 제정

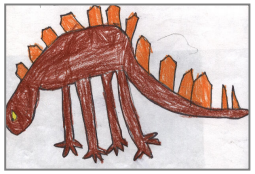
표준용어

복합명사 표준



컬럼명칭 표준

작의적 사용(전용) 금지



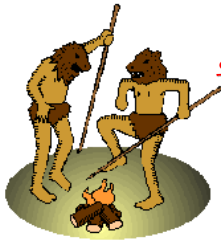
대충 실시한 설계로 인해 개발자 임의로 사용

본질이 훼손된 것이 전용

정보의 가치 하락의 원흉

Ⅴ. 속성 정의 – 속성후보 선정 원칙

원시(source) 속성으로 보이는 후보는 절대 버리지 마라



원시속성이란?

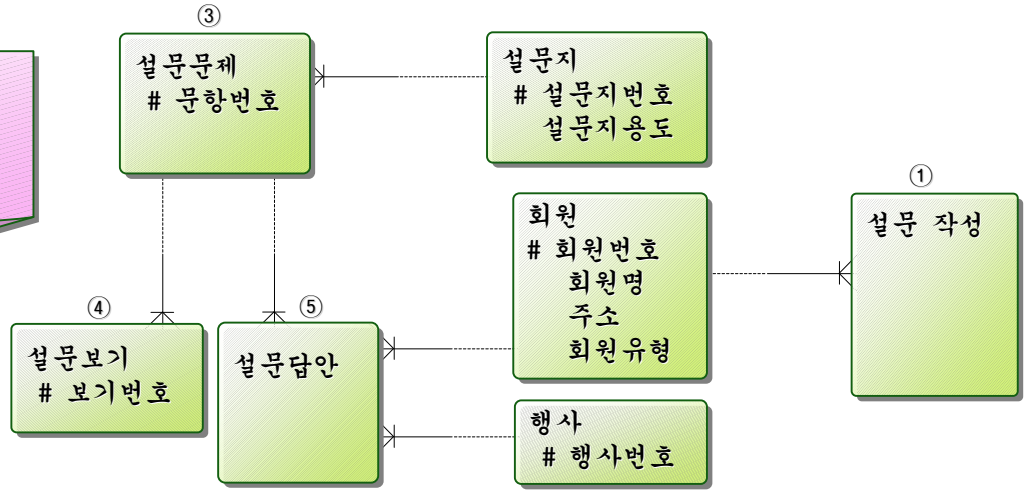
다른 속성에 의해 다시 재현할 수 없는 속성
이 속성이 없다면 다시는 재현할 수가 없는 속성

과연 재현할 수 있는가?

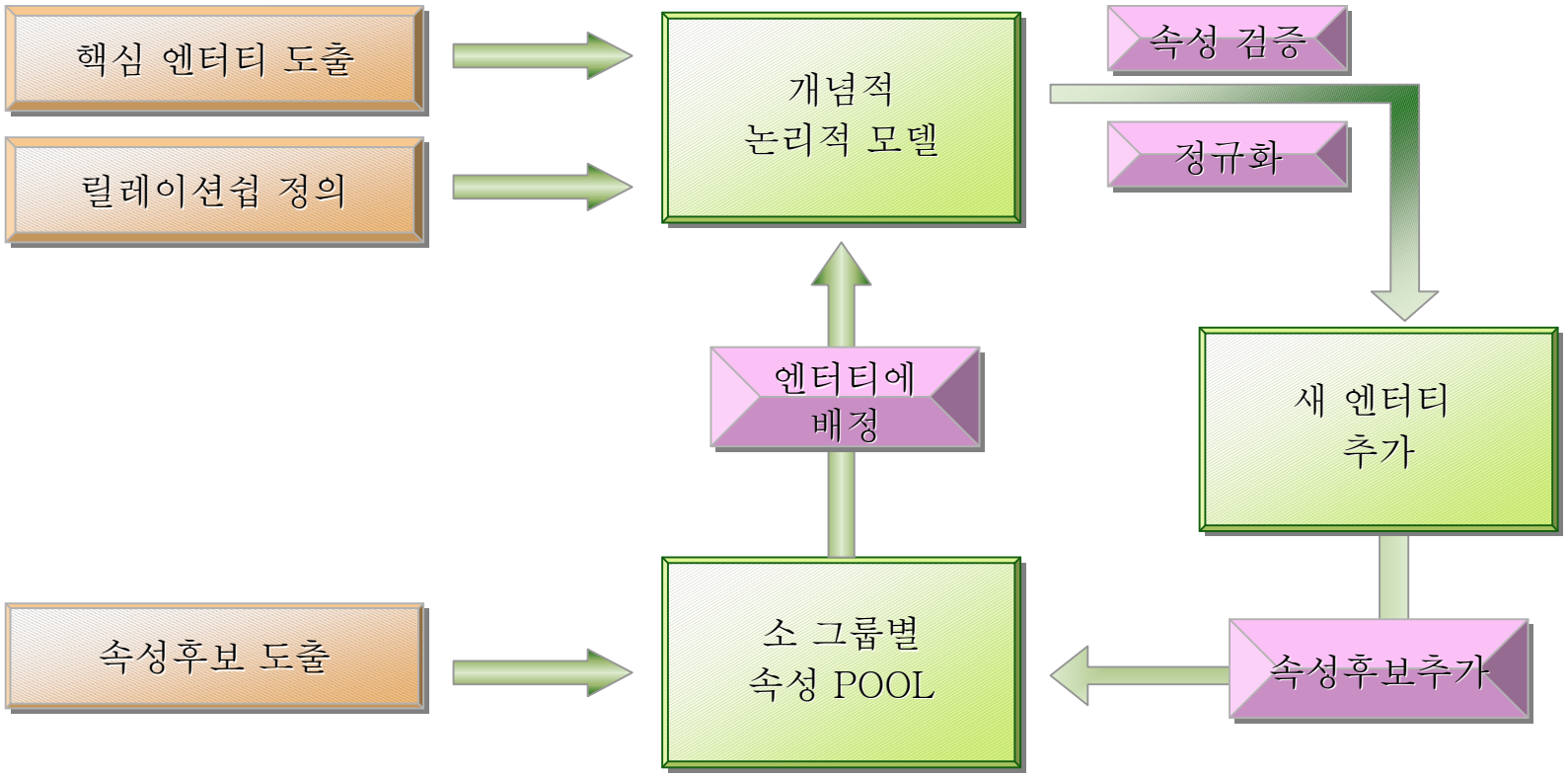
1) 다른 속성에 의해 자신이 만들어질 수 있는가?
2) 자신이 없어졌을 때 존재 불가능한 속성이 있는가?

소그룹별로 후보군을 만들고 가장 근접한 엔터티에 소속

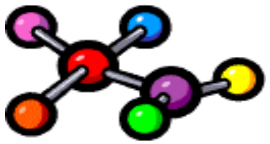
설문지유형, 질문내용, 답변내용(text), 선택답안,
설문지작성일, 설문작성행사, 보기내용, 보기항목,
복수응답가능유무, 주관식/객관식구분, 설문지용도,
설문지생성일, 관련문항번호, ……



Ⅴ. 속성 정의 – 속성후보 선정 원칙(엔터티↔속성)

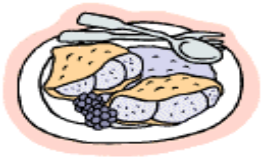


Ⅴ. 속성 정의 – 속성 검증 및 확정



- 최소 단위 분할 → 원자 단위냐?

- 유일값 존재 검증 → Single Value 냐?



- 가공값 유무 판정 → 추출값이냐?

- 관리수준 상세화 검토 → 보다 상세하게 관리할 것이냐?

Ⅴ. 속성 정의 – 속성 검증 1 단계 (원자단위 검증)

① 성 명
- 홍 길동
- 김 길동
- Edgar Allan Poe

② 성 / 명
- 홍 / 길동
- 김 / 길동

③ forename / family name
- Edgar Allan / Poe
- John Fitzgerald / Kennedy
- Gil dong / Hong

	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

최소단위 까지 분할

- 집합 개념의 속성은 단순개념으로 분할
- 일단 최소 단위까지 분할한 후 필요에 따라 통합
- 일자,시간,성명,주민등록번호,우편번호 등은 일반적으로 분할하지 않는 것이 좋다
- 주소와 같은 것은 그냥 두었다가 설계단계에서 필요 시 분할하기도 한다
- 분할 및 통합의 기준은 업무의 요구사항 영향

Ⅴ. 속성 정의 – 속성 검증 1 단계 (원자단위 검증)

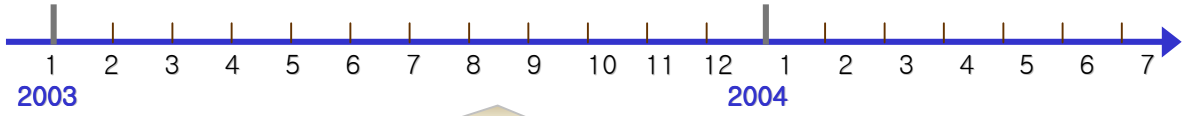
잘못된 속성 분할 사례

YYYY = '2003' and
MM between '03' and '07'

O

X

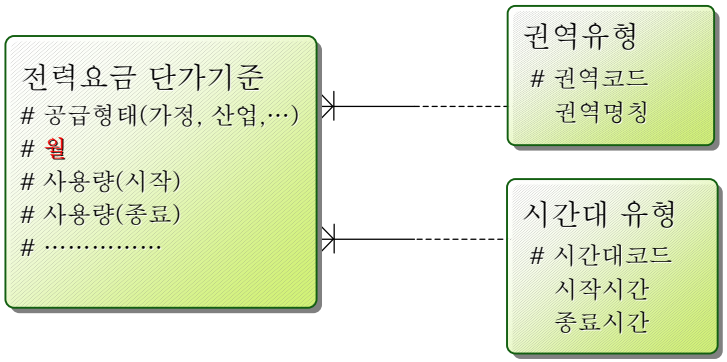
YYYY between '2003' and '2004' and
MM between '11' and '03'



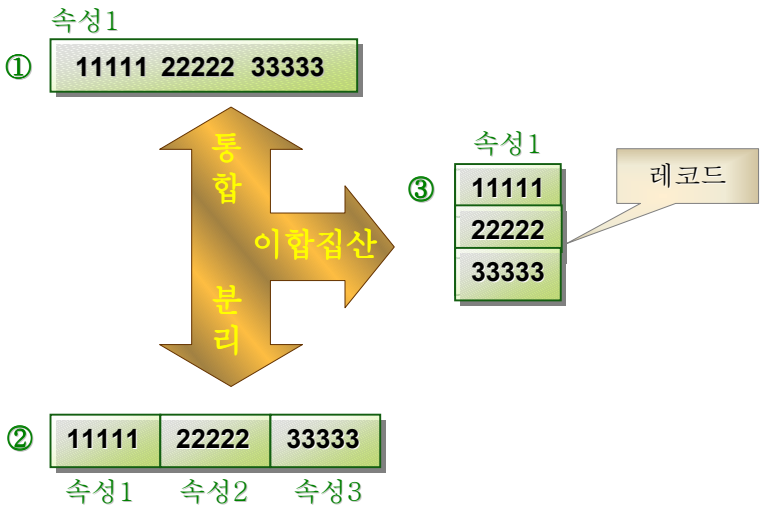
YYYY||MM between '200311' and '200403'

△

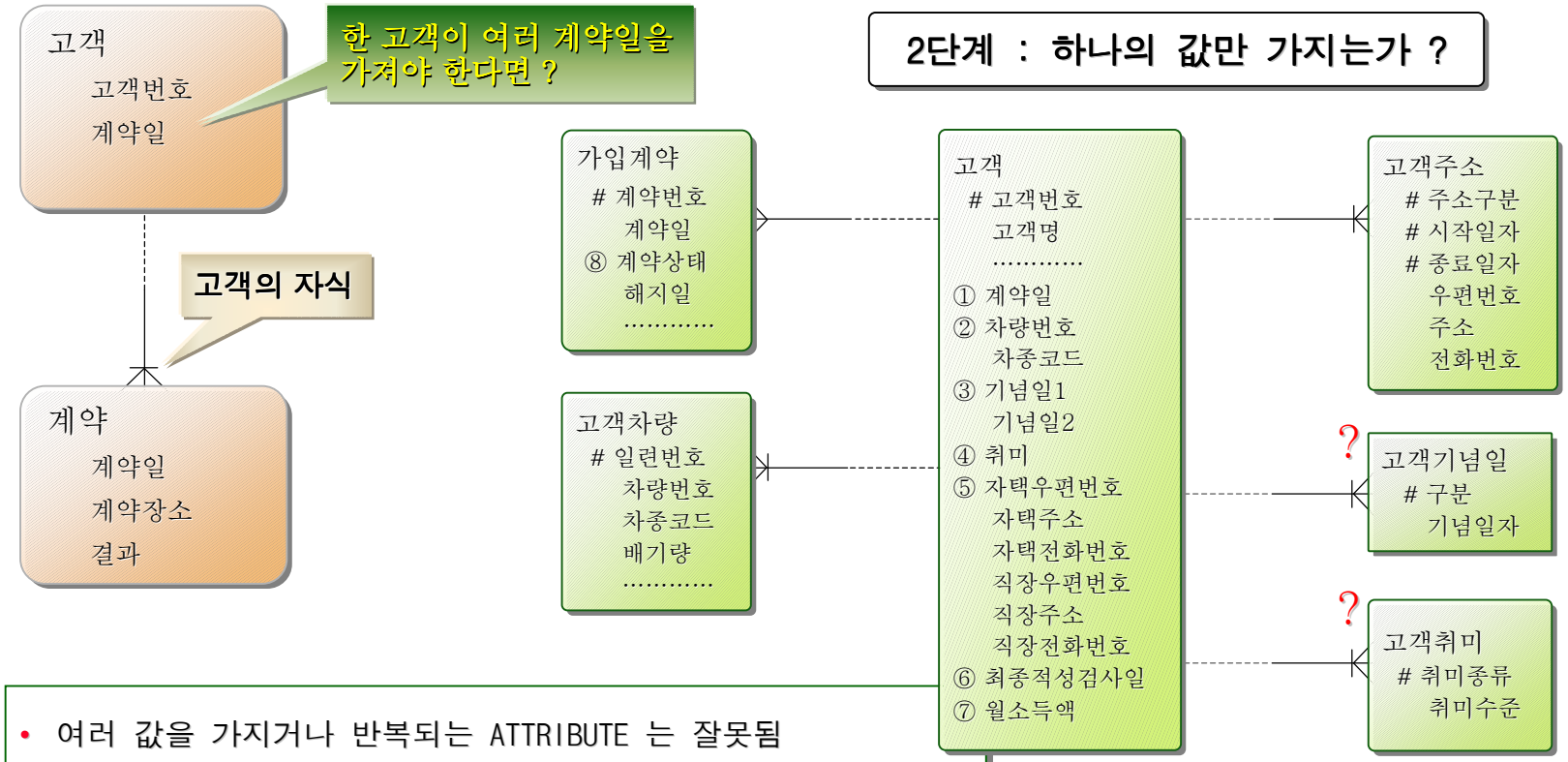
MM이 독립성을 갖는 사례



속성의 이합집산

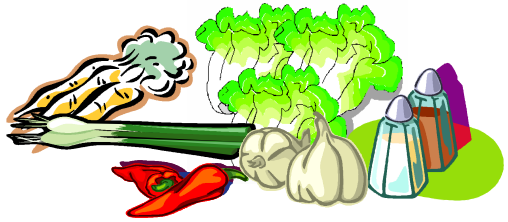


V. 속성 정의 - 속성 검증 2단계 (유일값 검증)



- 여러 값을 가지거나 반복되는 ATTRIBUTE 는 잘못됨
- 반복되는 경우 새로운 ENTITY로 분할
- 속성 의미에 대한 정의에 따라 하나 혹은 여러 개 일 수 있음
- 배타적 관계는 가능한 하나의 속성으로 통합할 것

Ⅴ. 속성 정의 – 속성 검증 3단계 (추출값 검증)



배추, 무우, 고추, 마늘, 소금

김치

김치찌게

김치는 원재료인가? 가공식품인가?

재현성(再現性)

버렸다고 가정했을 때
쉽게 재현할 수 있다면 추출값,
그렇지 않으면 원천값

'쉽게' 라는 것은
어느 정도 ?

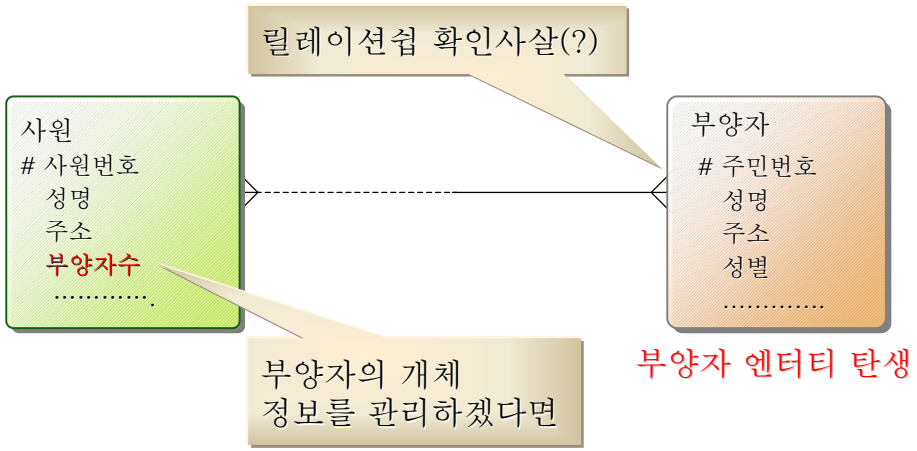
논리적 데이터 모델링에서는

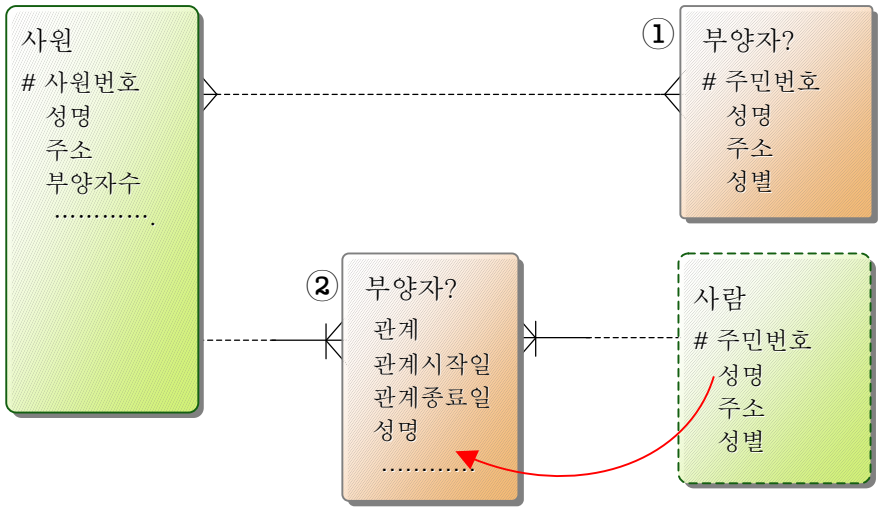
재현 비용이 지나치게 많이 소요될 것이 너무나
명백한 경우에 한해서만 원천값

차이가 미묘한 경우는

실용적 모델링 단계나 물리적 설계 단계에서 감안

Ⅴ. 속성 정의 – 속성 검증 4단계 (상세화 결정)

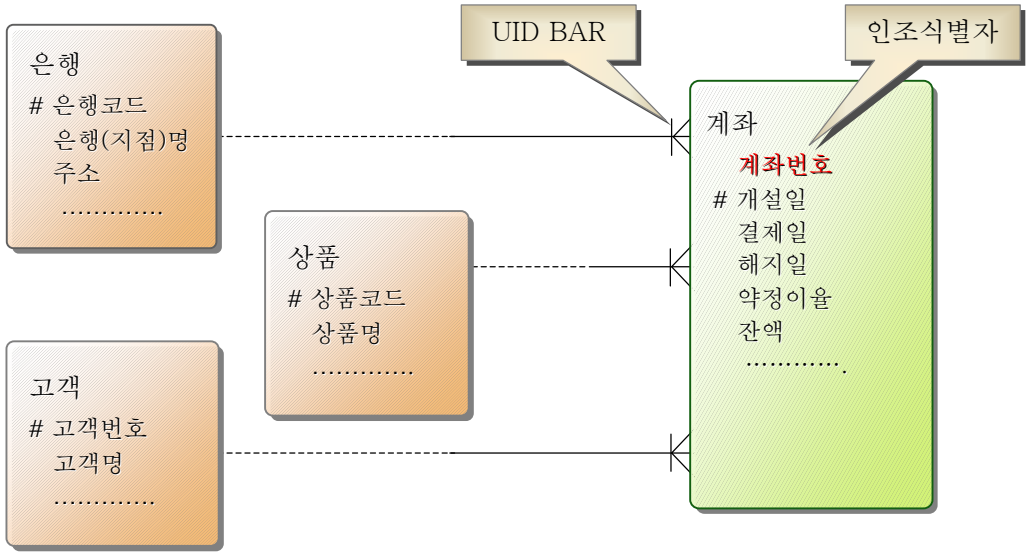




- 속성이 자기 소유의 속성을 가지면 엔터티
- 현재에 만족하지 말고 미래의 관리수준을 감안하라!
- 이 부분을 간과하면 개발 및 테스트, 운영 시에 많은 보완이 발생
- 모델러의 도덕성, 적극성, 인간미의 문제
- 한번 더 깊이 생각하지 않으면 우리 눈에 보이지 않는다.
- 모델링 시에 좀더 깊이 파 헤친 것이 시간 및 비용 절약의 원천

Ⅴ. 속성 정의 – 식별자(UID) 지정

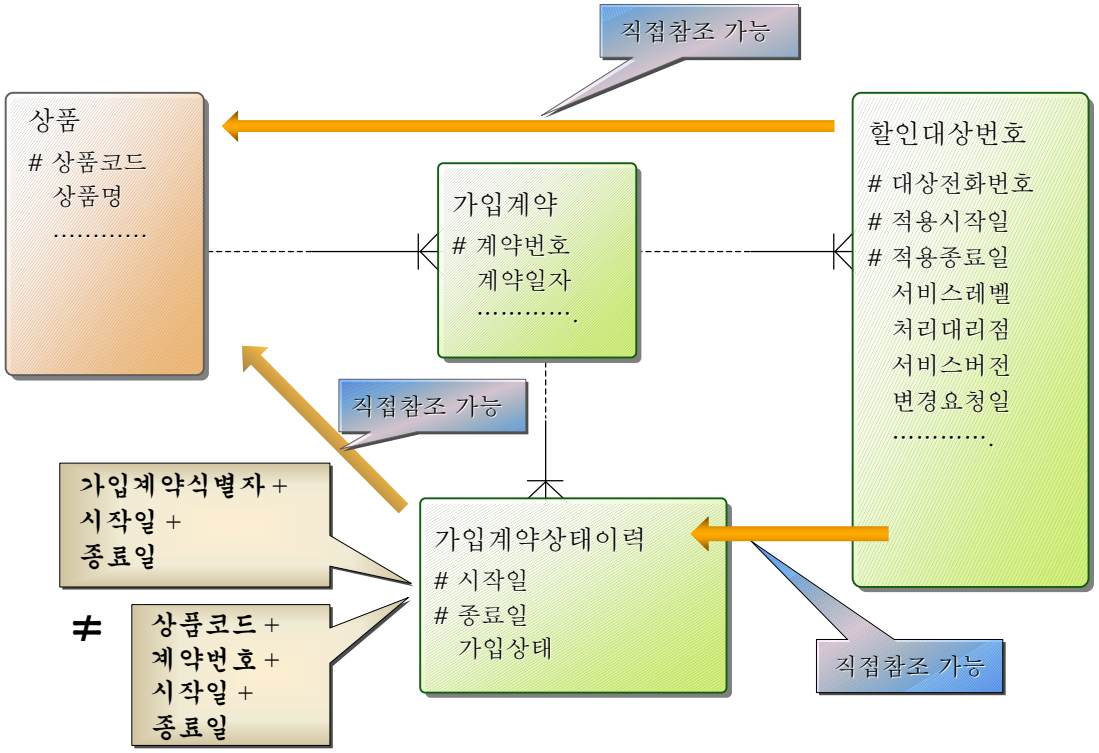
- UID : Unique IDentifier
- 하나, 혹은 하나이상의 ATTRIBUTE로 구성
- 모든 ENTITY는 반드시 UID를 가져야한다.
- UID를 가지지 못하면 ENTITY가 아니다.
- UID를 구성하는 모든 ATTRIBUTE는 반드시 존재해야 한다. (mandatory)
- UID Attribute에는 #* 를 표시
- 대부분의 키이 엔터티는 하나의 ATTRIBUTE로 구성됨
- 메인(Main) 엔터티와 액션(Action) 엔터티는 의미상의 주어가 UID
- 그러나 상황에 따라 의미상의 주어는 인조키(얼굴마담)로 대체될 수 있다.
- 무조건 인조키를 만들거나 상속받은 속성만으로 UID를 구성하는 것은 아님
- 전략적인 판단이 요구됨



V. 속성 정의 – UID BAR의 진정한 의미

릴레이션쉽은 '부모의 식별자' 라는 논리속성이다.

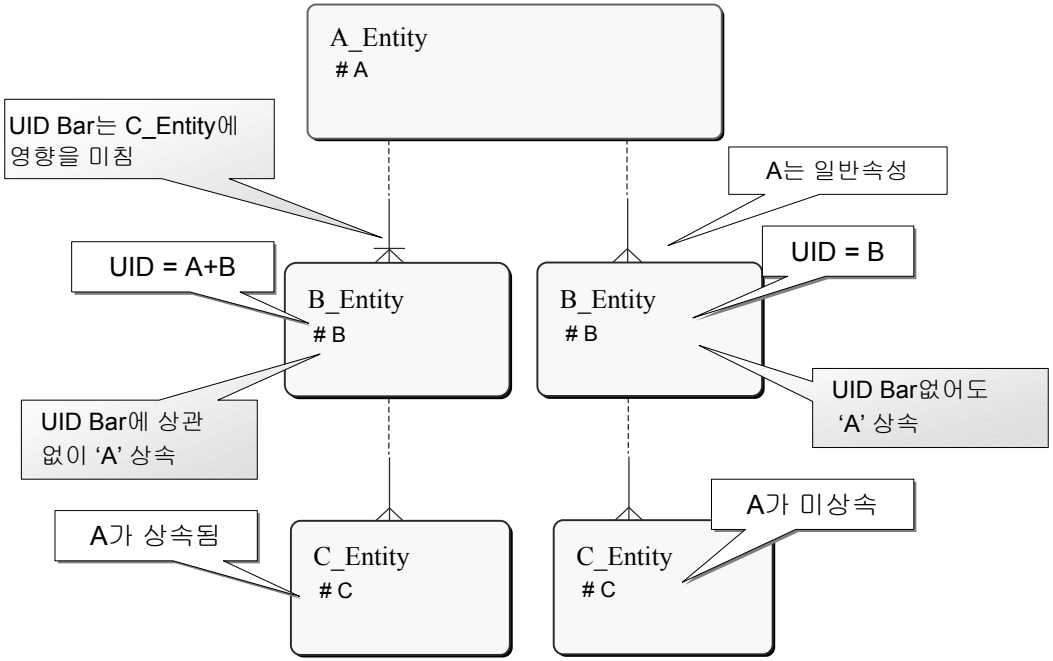
UID BAR는 단지 이 논리속성이 식별자의 한 부분이라는 것



- 변수도 하나의 숫자
- 상수만 숫자라고 생각하면 한계를 벗어나지...

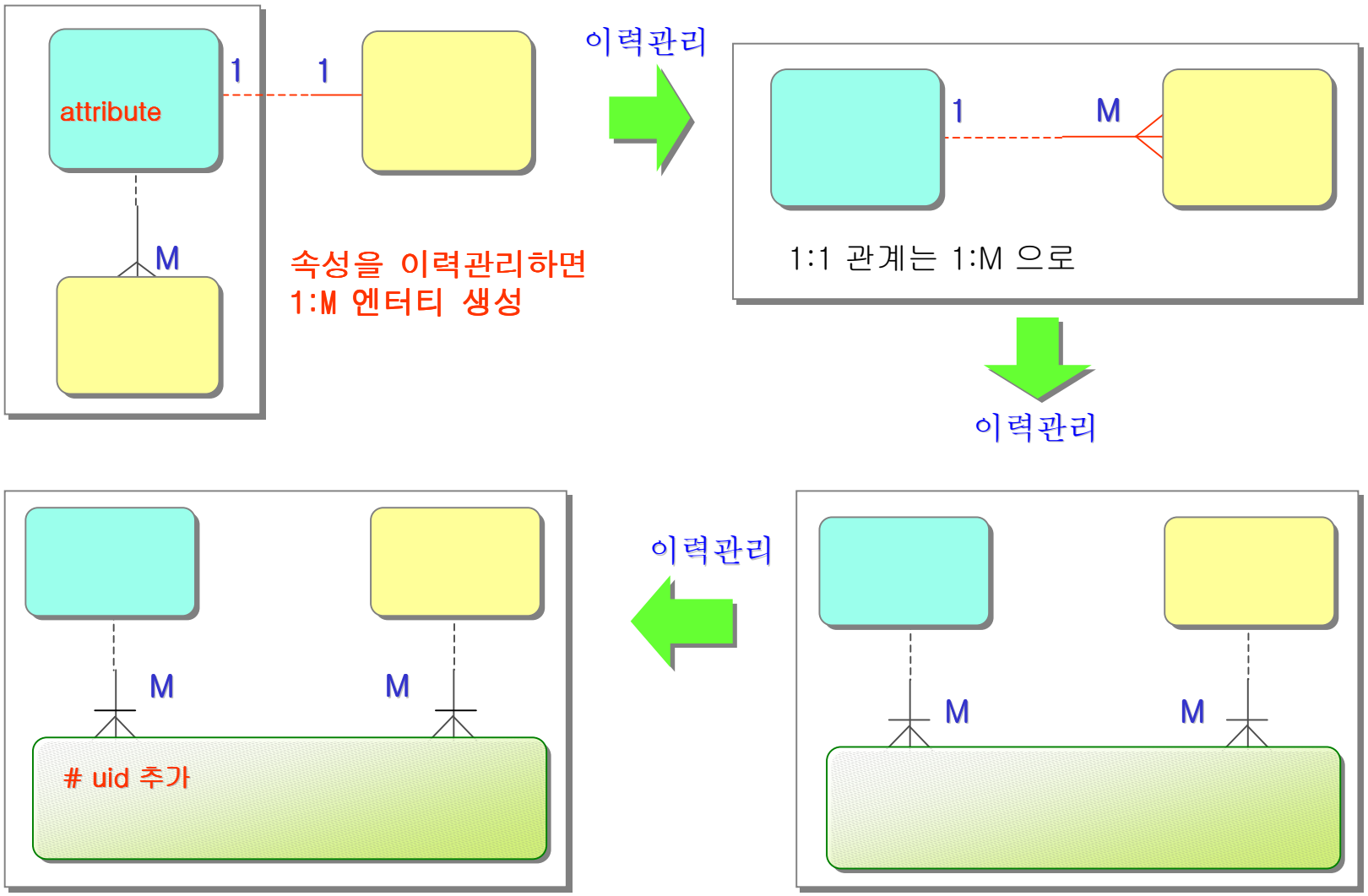
Ⅴ. 속성 정의 – 식별자의 확정 (UID의 2가지 역할)

- **식별자 역할**
- **정보로써의 가치**



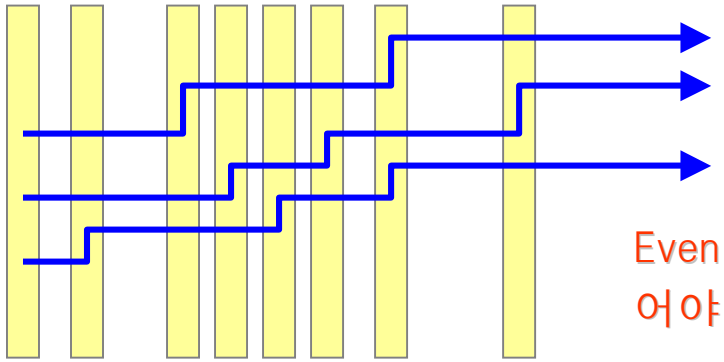
- 'C' 엔터티는 조부 엔터티인 'A'의 UID가 정보로써 필요한가?
- 'B' 엔터티는 얼마나 많은 자식 엔터티를 거느리고 있는가?
- 얼마나 많은 자식 엔터티들이 조부 엔터티와 친밀한가?
- 'C'나 그 이하의 자식 엔터티들이 얼마나 조부 엔터티의 UID를 정보로써 원하는가?
- 자식 엔터티 중 일부만이 조부 엔터티의 UID를 정보로써 원한다면 상속을 단절(UID Bar 없이)시키고 물리적 단계에서 접근경로 단축

Ⅴ. 속성 정의 – 이력관리 시의 관계 변화



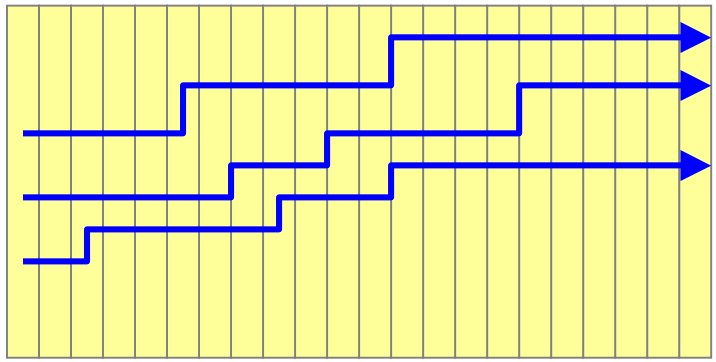
참고. 전통적인 이력관리 방법

- EVENT 발생 시 마다 생성



Event가 발생할 때마다 사진을 찍어 두어야 이력을 관리할 수 있는가?

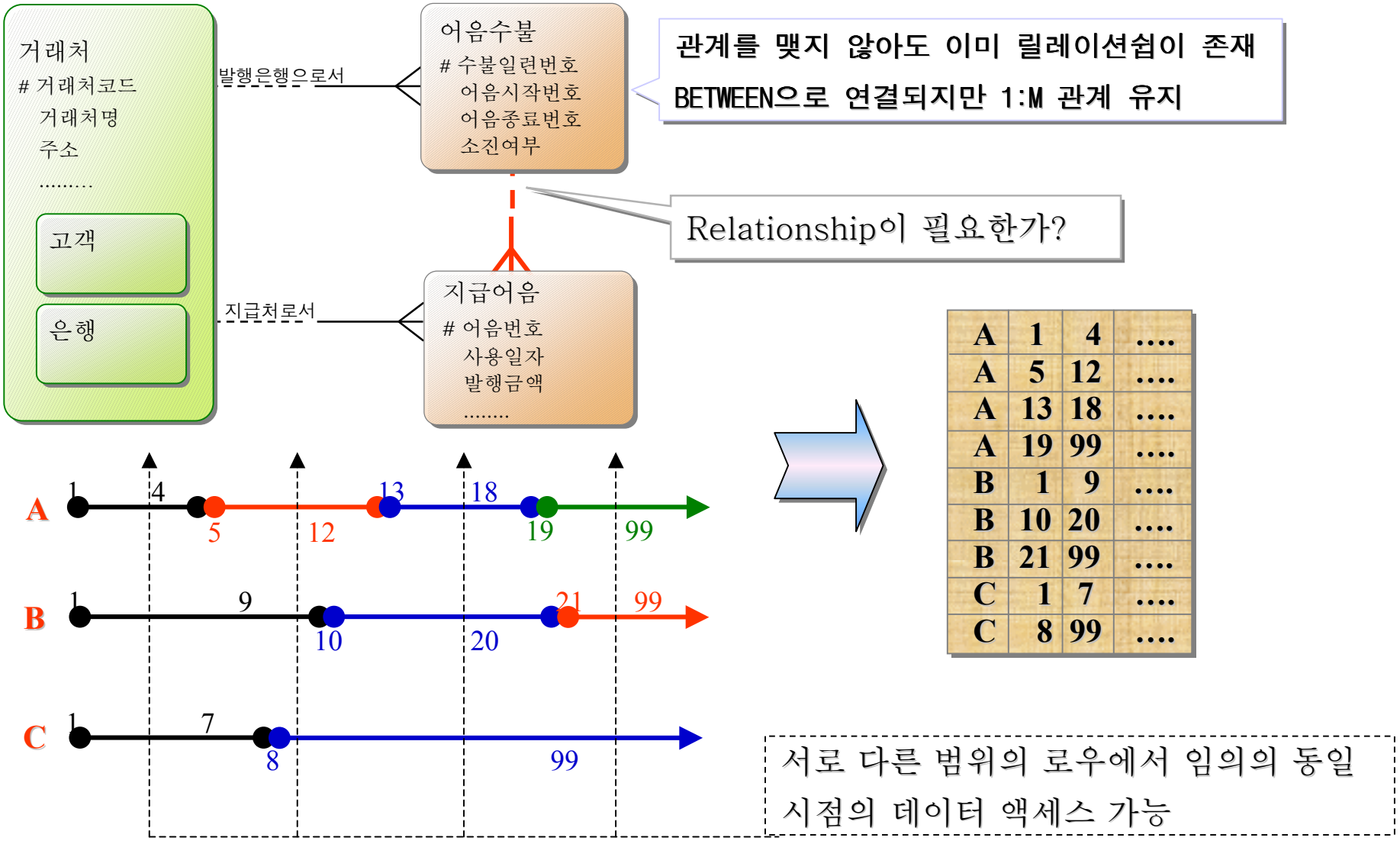
- DAILY 마다 생성



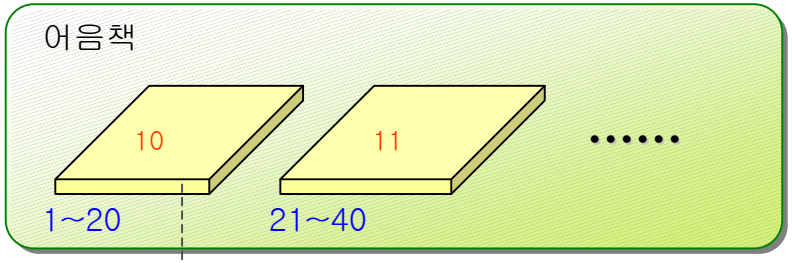
아무 변화가 없는 경우도 데이터 생성

그러나 완벽한 이력관리는 불가능

참고. BETWEEN Relationship



참고. BETWEEN RELATION의 개념



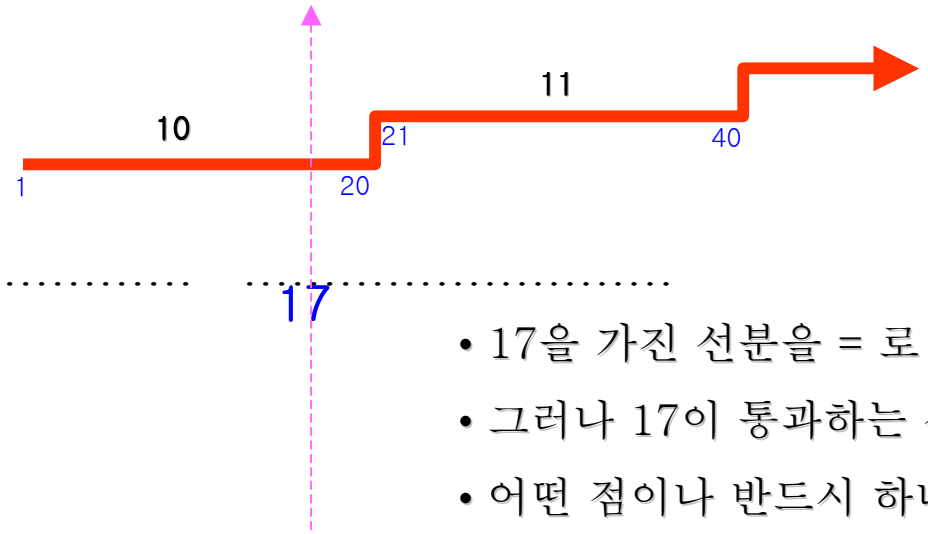
? 낱장의 어음이 반드시 하나의 어음책에 들어 있다고 해서 1:M 관계를 부여해야 하는가?

어음 1장마다 1레코드인 집합

Relation을 준다는 것은 결국 어음책번호를 지급어음에 갖다 두겠다는 의미

그러나 Relation을 없애더라도 관계는 존재 (between relation)

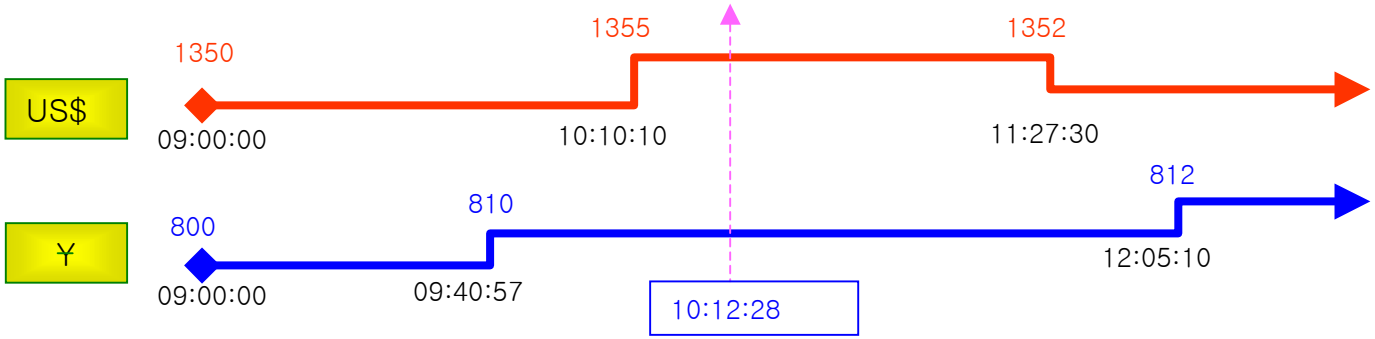
참고. 이력선분의 기하학적 고찰



- 17을 가진 선분을 = 로 찾을 수는 없다 !
- 그러나 17이 통과하는 선분을 찾아보자 !!!
- 어떤 점이나 반드시 하나의 선분을 통과한다.
- 선분이 아무리 길어도 레코드는 하나

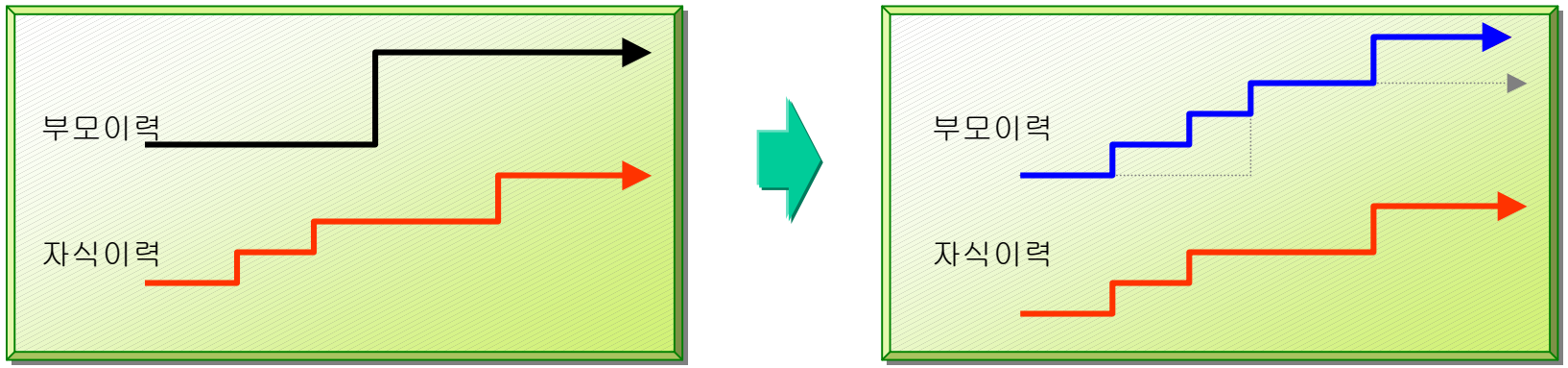
- 점을 통과하는 선분 찾는 방법
  - 부등식 표현 : 시작점 <= 17 <= 종료점
  - 연산자 표현 : 17 Between 시작점 and 종료점

참고. 이력 선분의 사용 예



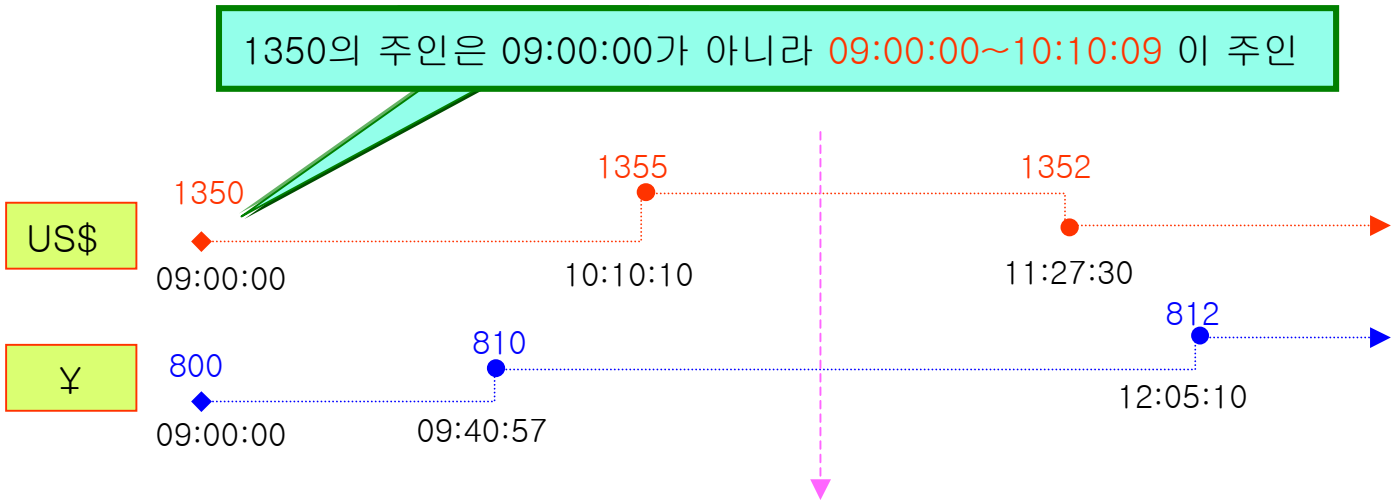
- US$: 1355
- Y : 810

◆ 모 유명 Package의 이력관리 방법



= 로 읽기 위해 부모의 이력을 복제

참고. 이력(선분)과 EVENT(점)의 차이



특정 시점의 데이터를 찾으려면 <= 은 것들을 모두 읽어 MAX를 구해야 함

선분을 표시하기 위한 두 점에서 하나를 빼면 선분이 없어진다.

Q & A

Contact Info : jhmyung@en-core.com

데이터 모델 관리
- Modeler 관점의 품질관리

데이터 모델 관리

명 재 호 이사
엔코아 컨설팅

한국데이터베이스진흥센터

목차

Ⅰ. 데이터 모델 관리 개요 – 프레임워크의 등장배경


경영환경의 정보화시대


인수.합병


급속한 국제화

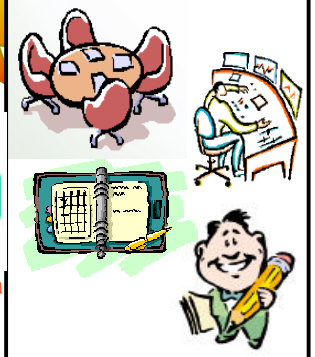

소비자요구의 고급화

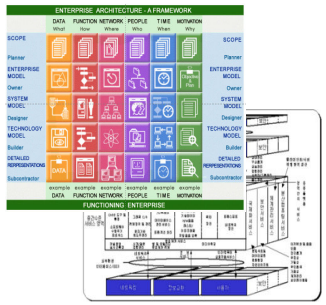

효율과 효과 지향으로
경영의 패러다임이 전환

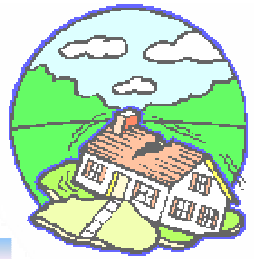
정보 기술 환경의 다변화

업무변화
조직변화
기술변화


Disconnection


BUSINESS


목표의 효과적 달성과
전략적 우위 선점의 목적을 위해
정보기술 아키텍처의 도입 절실


경영 최대화의 기반인
정보시스템을 근본적
으로 불안하게 함


빈번하게 아키텍처를
조정하는 악순환 반복

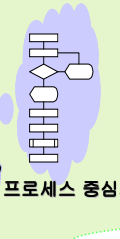

주로 문서적인 연결


사수와 조수에 의
한 업무 계승


Business Rule이 프로그램 소
스코드에 숨어있다


업무담당자 시각에서
단편적인 요구 난무


프로그래머는 많지만
설계전문가는 적다



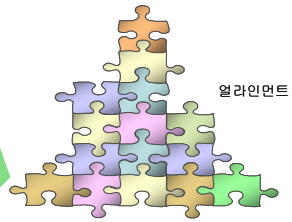
프로세스 중심의 접근


유지.보수 위주의 경력

Ⅰ. 데이터 모델 관리 개요 – 모델 데이터의 필수요건

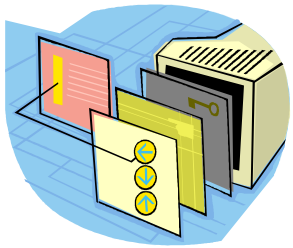
구조적으로 접근할 수 있어야...

전 단계 지원 가능

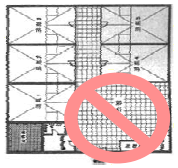


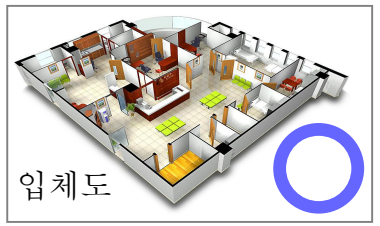
서로 유기적인 결합

진행과정을 구체적으로 정의할 수 있어야...



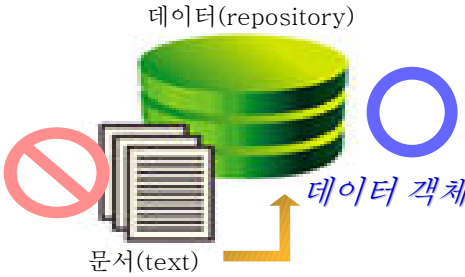
진행과정을 지원





매우 상세하고 입체적으로

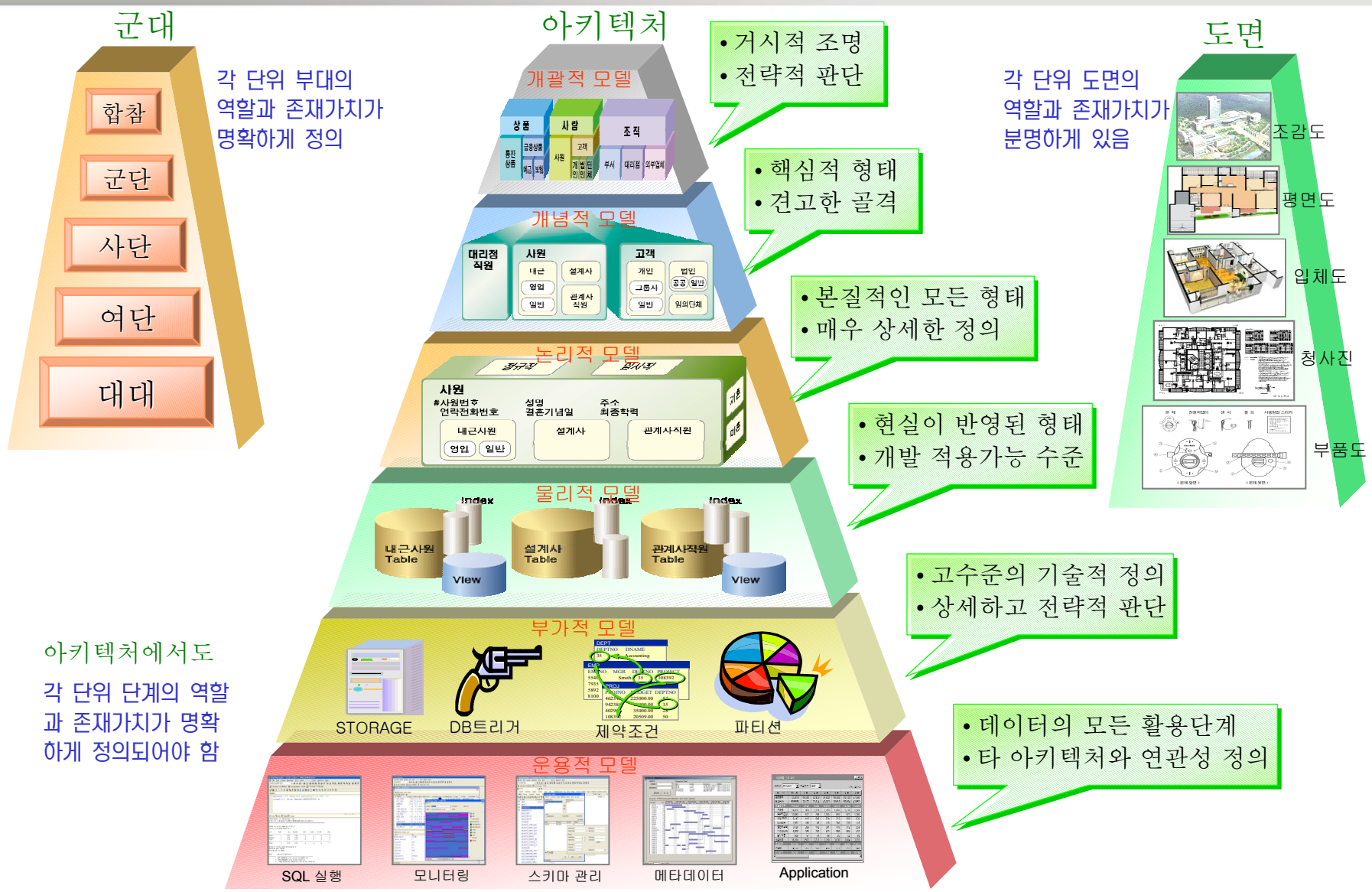
데이터적으로 표현할 수 있어야...



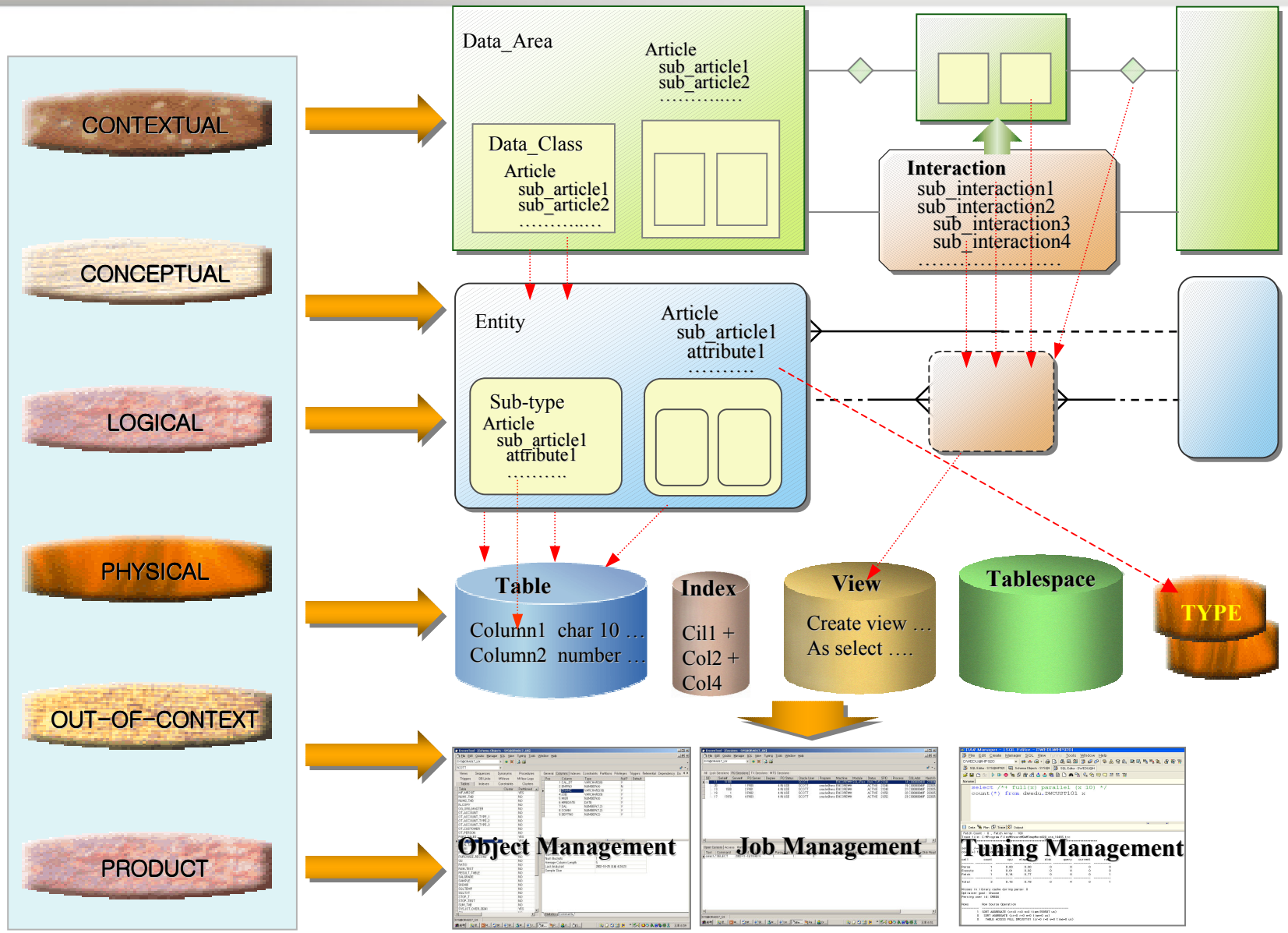
시스템적으로 관리할 수 있어야...

전문가 집단에 의해 주관되어야...

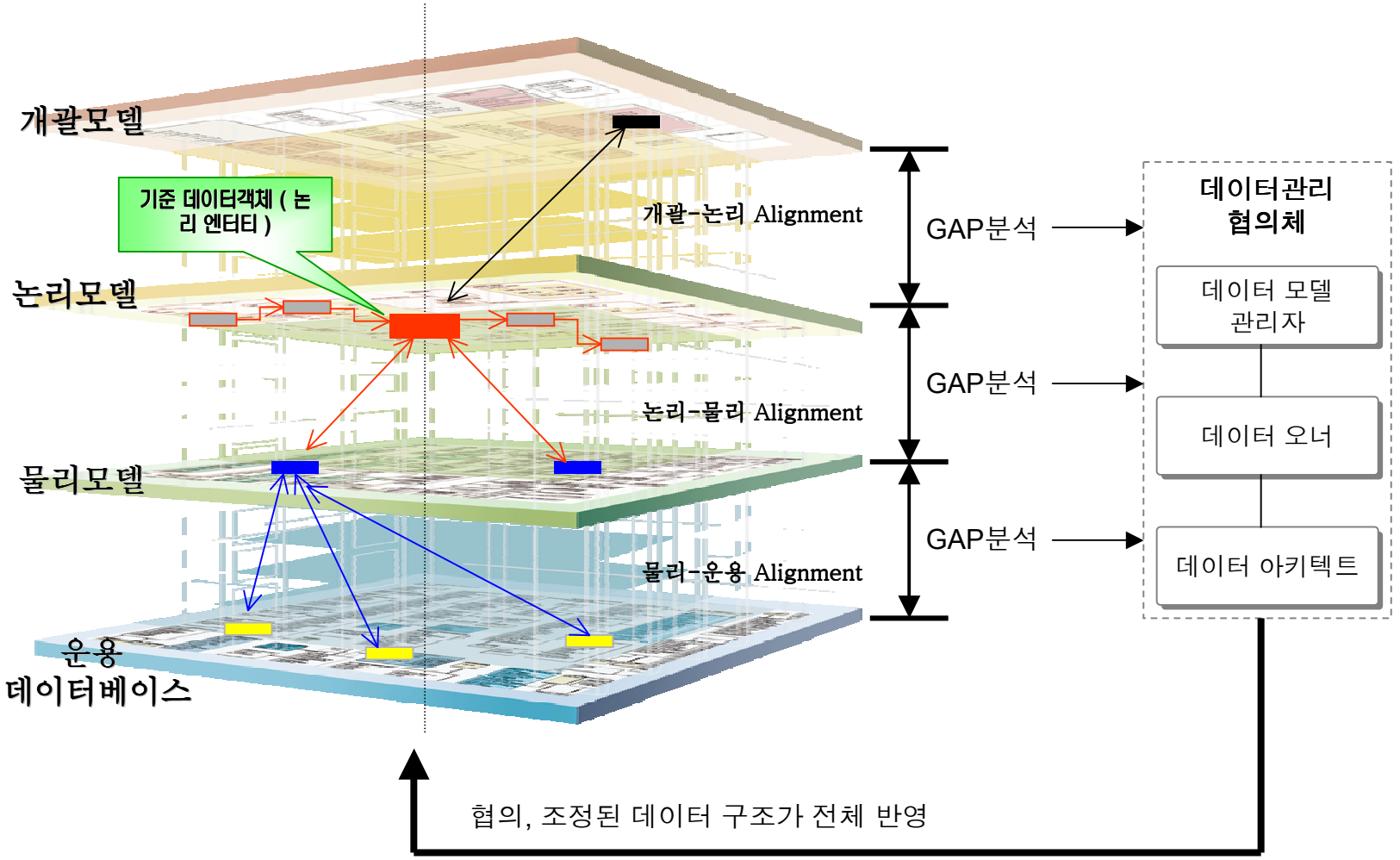
Ⅰ. 데이터 모델 관리 개요 – 데이터 모델 (전체 프레임워크)



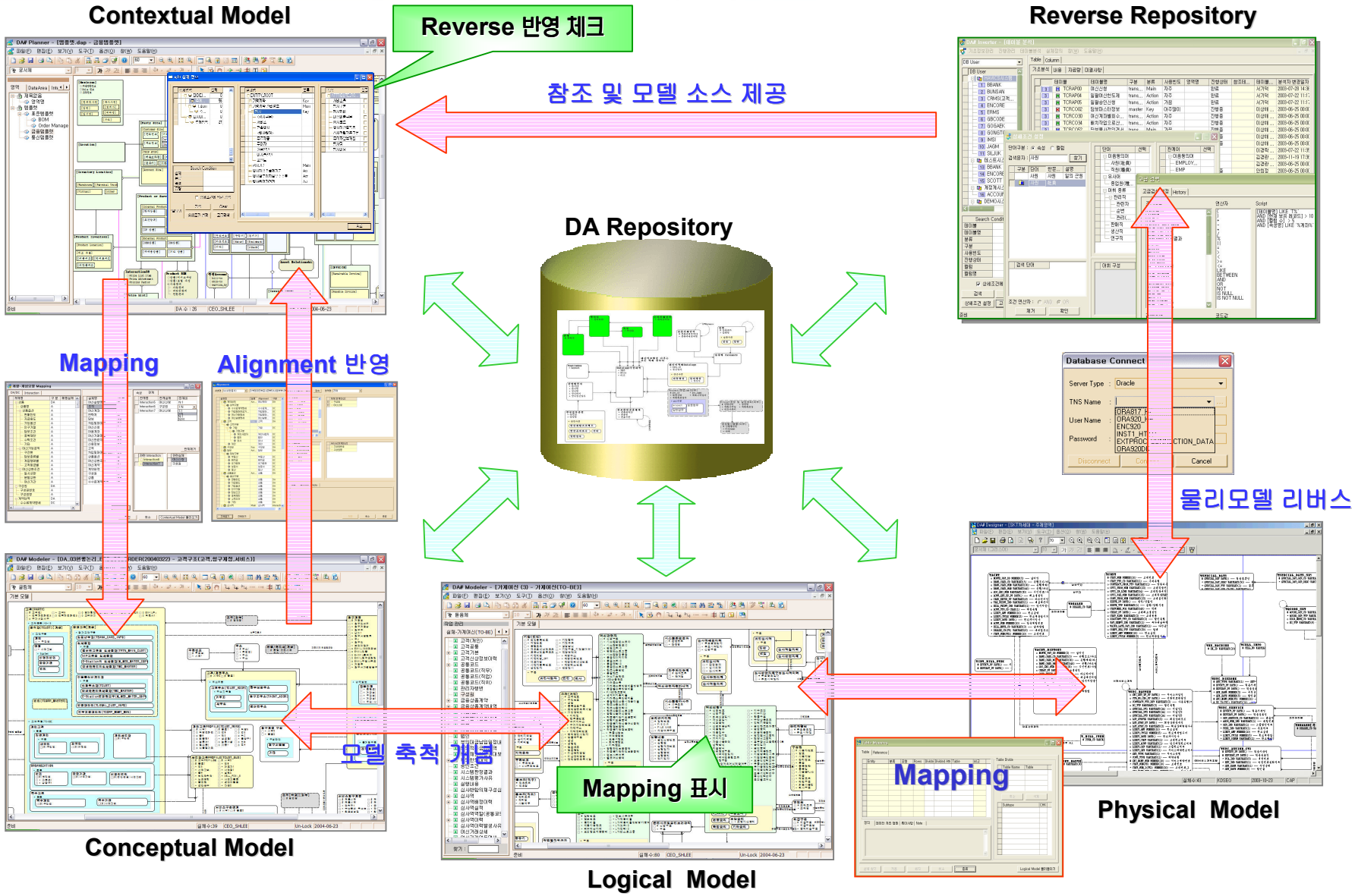
Ⅱ. 데이터 모델 관리 : Alignment



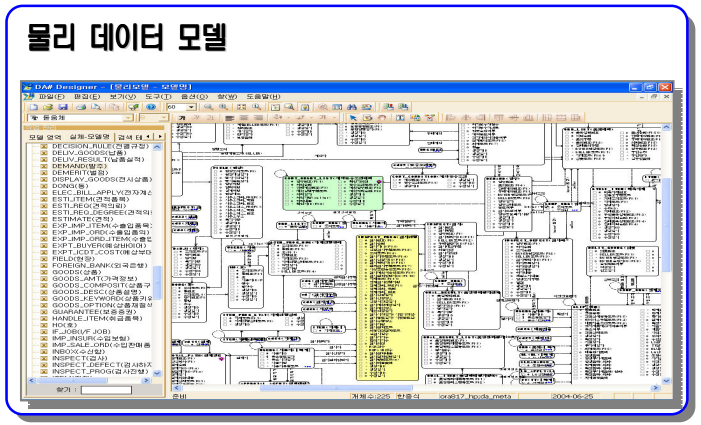
Ⅱ. 데이터 모델 관리 : Alignment



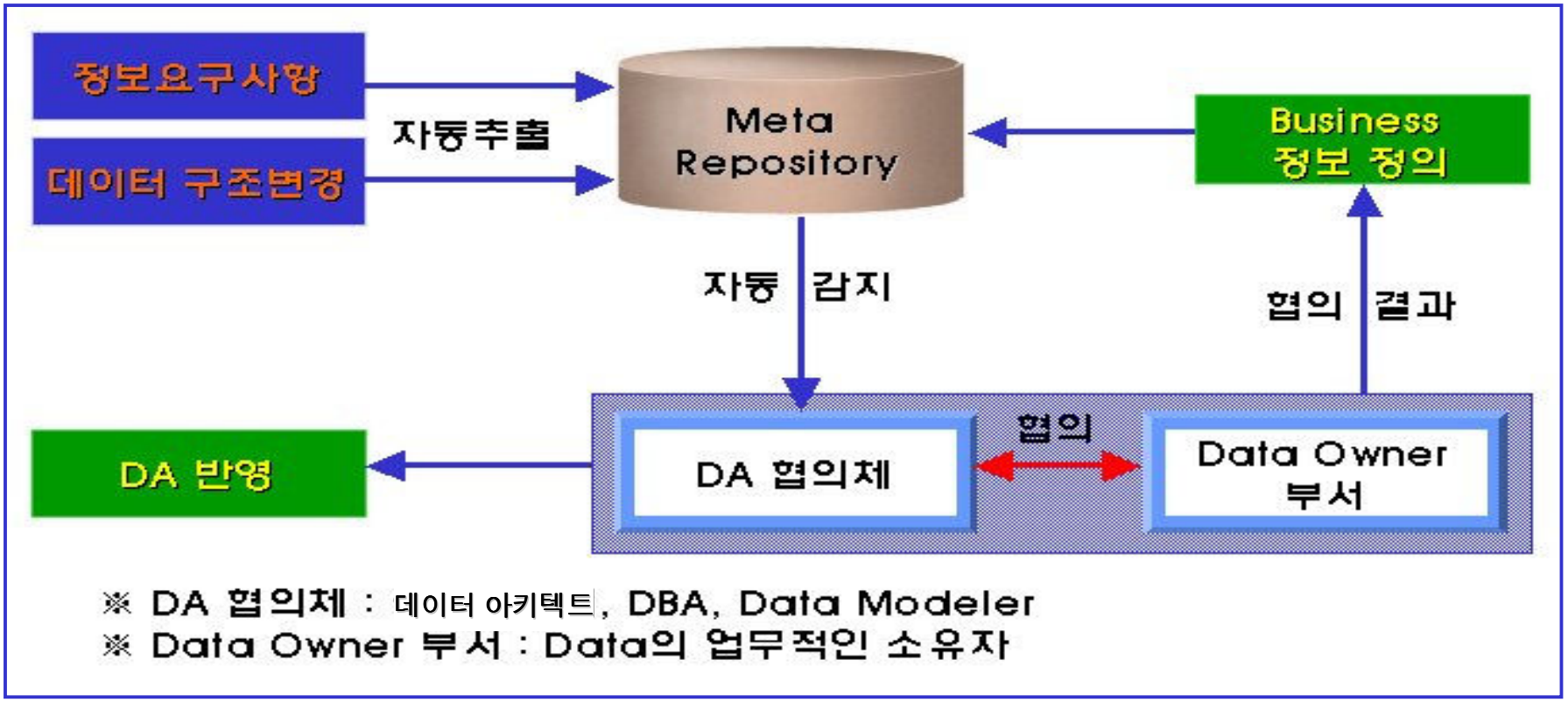
Ⅱ. 데이터 모델 관리 : Alignment – 사례



참고. 데이터 모델 관리 : 물리 모델, DBMS 비교 분석



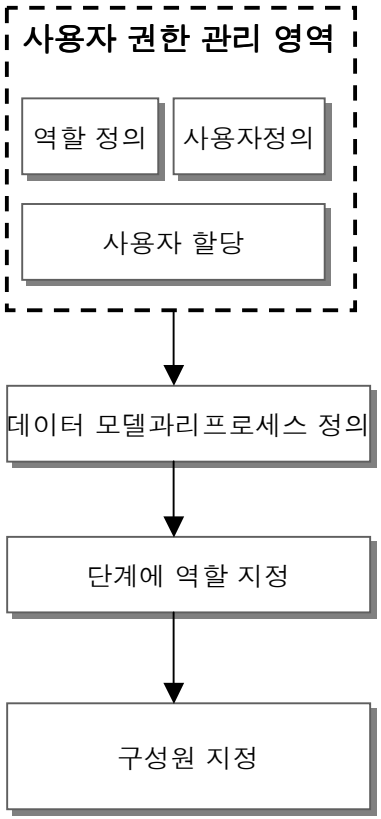
Ⅲ. 데이터 모델 관리 : 프로세스



- ❑ 전사 데이터 모델(구조,체계) 관리를 위한 표준 절차 수립
- ❑ 위의 수립된 내용을 근거로 시스템화
- ❑ Data Modeler, DBA, 기타 의사 결정권자로 이루어진 상시 협의 기구 생성
- ❑ 전사 DA를 관리하고 생성하기 위한 Third Party Tool 검토 및 도입

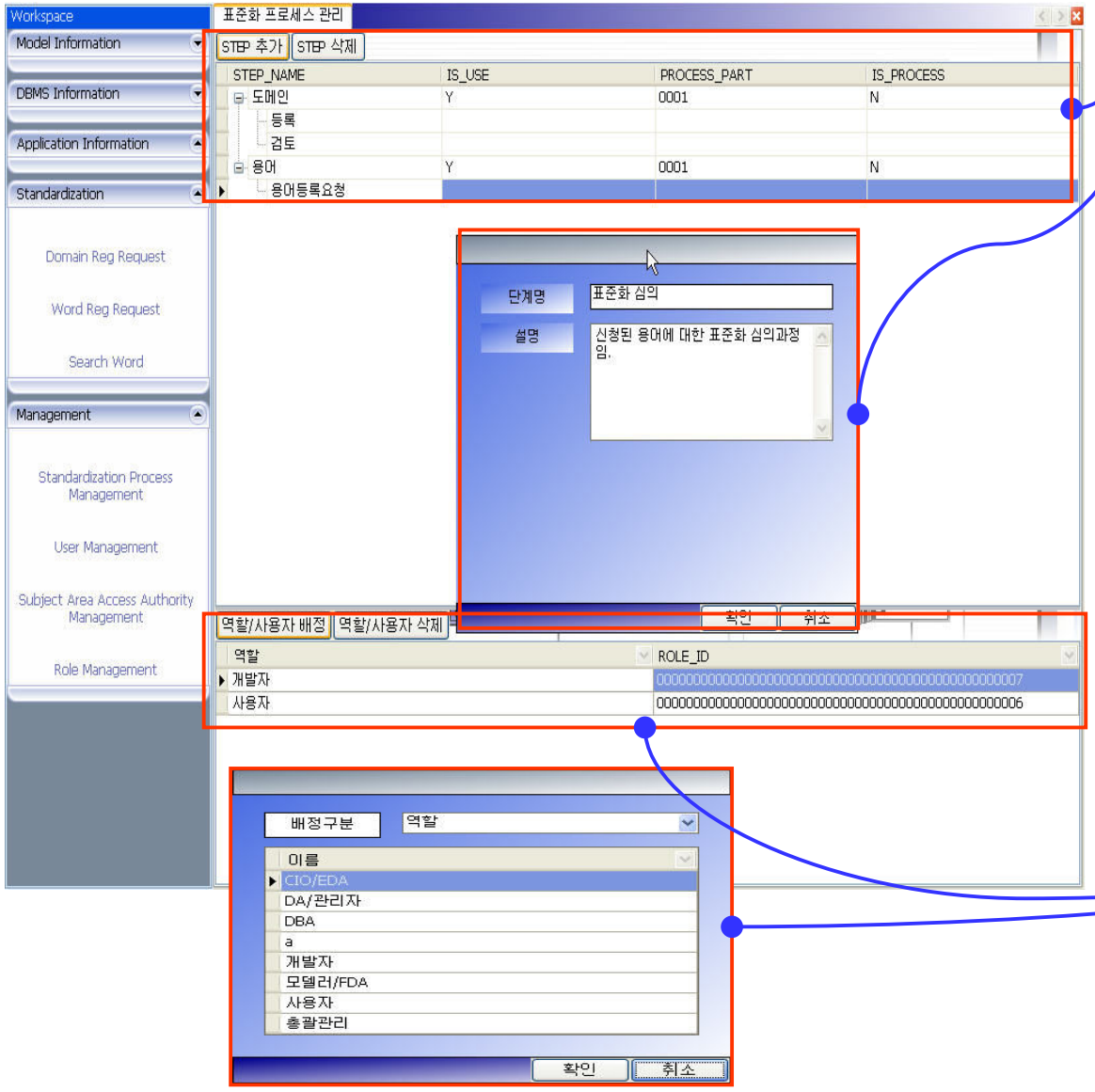
Ⅲ. 데이터 모델 관리 : 프로세스 – 프로세스 정의

데이터 관리 프로세스를 수행하기 위한 조직을 정의하고 각 조직의 구성원들과 그들의 권한을 정의하는 과정이다. 또한 각 조직들이 데이터 모델 관리에 대한 각 역할을 어떤 절차로 수행할 것인지를 정의하는 것이 필요하다. 획일화된 프로세스가 아닌 회사의 실정에 맞도록 정의해서 사용하는 것이 필요하다.



- 데이터 모델 관련 모든 역할 (Role) 들을 정의 한다.
- 정보 시스템 사용자들의 모두 위에서 정의한 역할이 할당한다.
- 데이터 모델 관리 프로세스 각 단계를 정의한다.
- 단계를 수행할 권한을 가진 역할을 지정한다.
- 해당 역할 중에서 특정 구성원이 할 수 있다고 지정할 수 있다.
  Default는 해당 역할에 구성되어 있는 모든 구성원이 수행할 수 있다.

Ⅲ. 데이터 모델 관리 : 프로세스 – 프로세스 정의(사례)



✓ 기업의 현실에 맞는 데이터 표준화 프로세스를 정의 한다.
✓ 먼저 표준화의 각 단계를 지정한다.

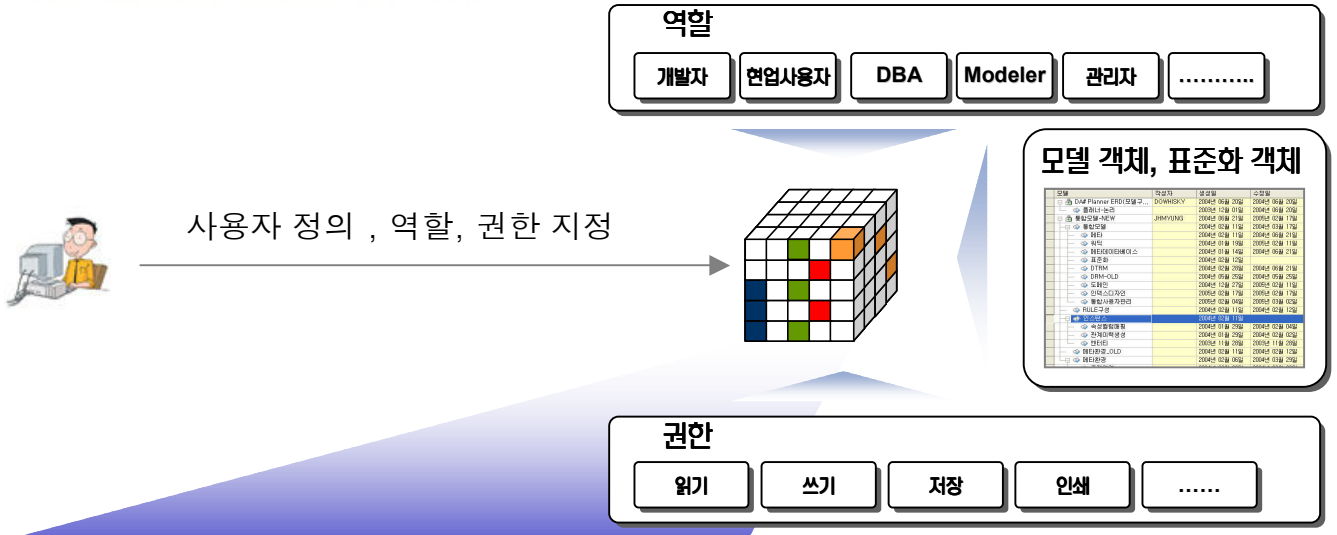
✓ 위에서 정의한 각 단계,프로세스 마다 해당 프로세스를 수행할 수 있는 역할을 지정한다.
✓ 즉, 표준용어 요청을 누가 할 수 있는가를 지정하는 것이다.
✓ 기업의 실정에 맞게 적은 조직에서는 데이터 표준화 프로세스의 대부분의 단계를 한사람, 또는 한 두사람이 전부 수행하는 경우도 있을 수 있는 것이다.
✓ 여기에서 지정한 역할에 해당하는 사람은 특정 단계가 진행되면 자동적으로 본인들이 수행해야 하는 단계(프로세스)를 처음 로그인 화면에서 할 수 있도록 유도한다.

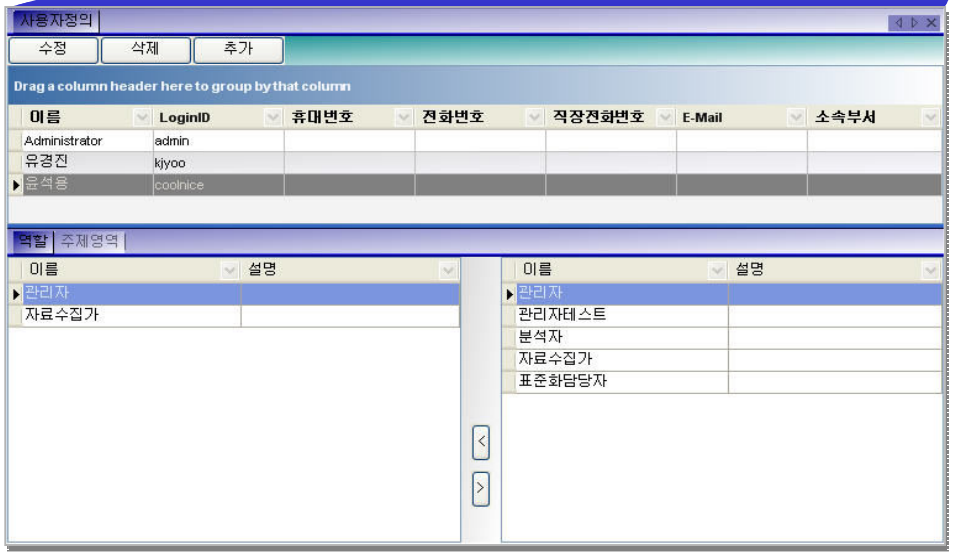
Ⅳ. 데이터 모델 관리 : 조직 , 권한, 사용자 – 조직

조직/역할	명 칭	설 명
CIO/EDA	◆EDA(Enterprise Data Architect) ◆CIO	➢ 전사 데이터관리 총괄 ➢ 데이터관리 정책 및 지침 마련 ➢ 데이터관리자간 이슈사항 조정
DA	◆DA(Data Architect) ◆관리자	➢ 표준 개발 및 형상관리, 검증/표준화 절차 수립, 운영 ➢ 전사 데이터 모델 통합 지휘 ➢ 데이터 요구사항에 대한 정리 및 FDA 지원
모델러	◆FDA (Functional Data Architect) ◆모델러 (Modeler)	➢ 해당 기능 영역의 데이터 요구사항 및 이슈사항 조정과 통합 ➢ 해당 기능영역의 비즈니스 요건을 토대로 데이터 모델링 수행 ➢ 표준확인 및 적용
DBA	◆DBA (DataBase Administrator)	➢ 데이터베이스 디자인, 데이터 표준화 준수 의무를 가짐 ➢ 데이터베이스와 데이터의 형상관리 수행 ➢ 데이터베이스의 모니터링 및 튜닝, 보안관리
사용자	◆일반사용자 (User) ◆현업사용자	➢ 서비스되는 데이터 및 운영/분석 데이터에 대한 활용 ➢ 데이터에 대한 추가요건 요청
개발자	◆개발자(Developer)	➢ 각 기능을 해당 데이터모델을 활용하는 Application으로 생성 ➢ 논리 데이터 모델에 대한 이해를 바탕으로 개발 필요 ➢ 개발 표준화 준수 의무

각 역할 중에서 여러 명칭 중에서 회사의 표준에 맞게 지정하여 사용할 수 있다.
역할을 새롭게 정의/ 재정의하여 사용할 수도 있음.

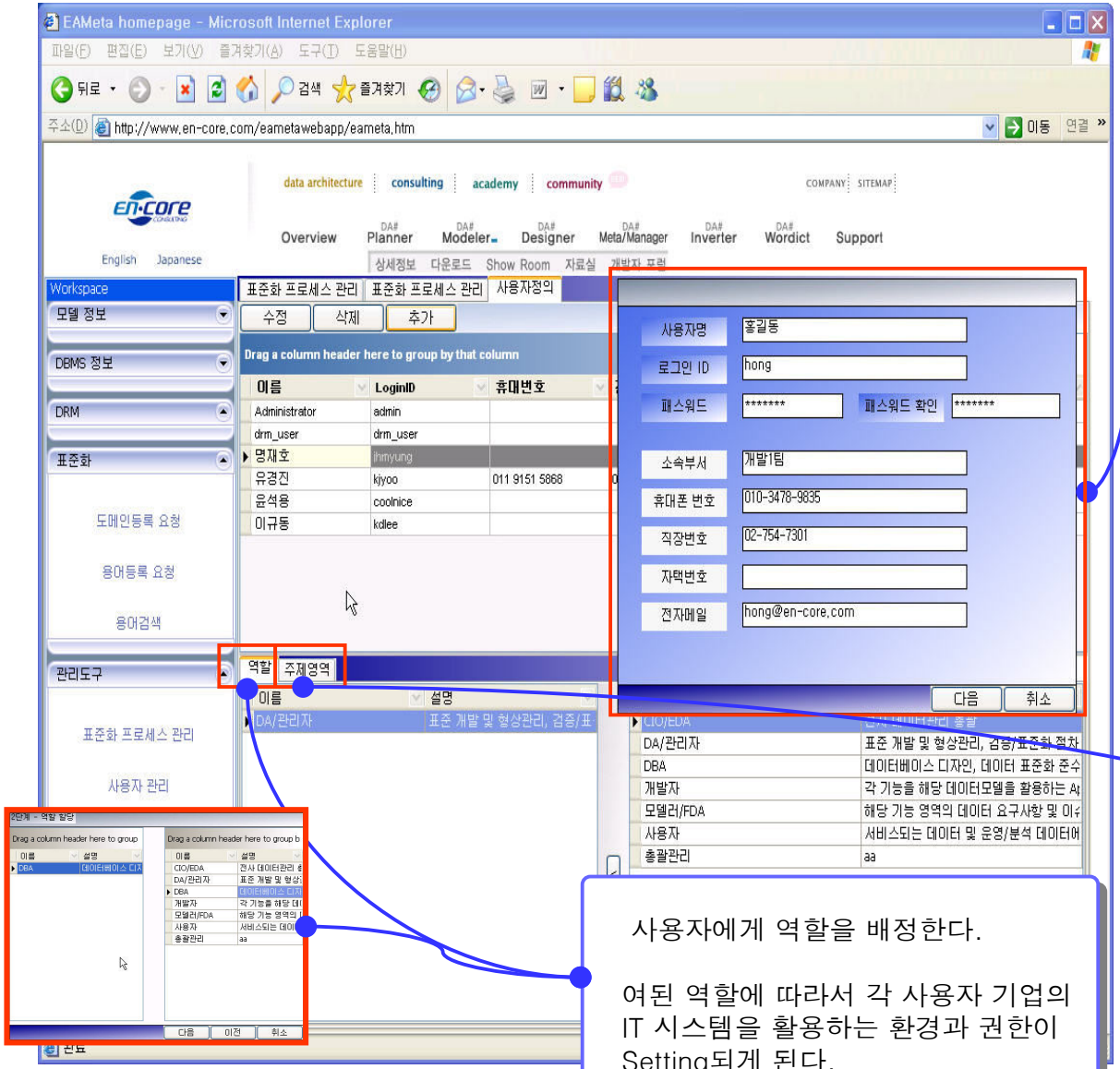
Ⅳ. 데이터 모델 관리 : 조직 , 권한, 사용자 – 권한





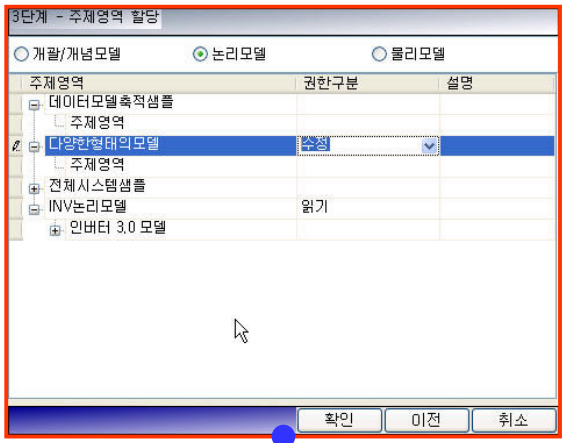
- 정보 시스템을 사용하는 사용자를 정의한다.
- 해당 사용자의 역할을 지정한다. (개발자, DA, DBA, Modeler, 관리자,.....)
- 각 사용자에 데이터 구조 (모델)에 대한 객체 권한을 부여한다.
- 각 사용자에 표준화 객체에 대한 권한 (용어 등록 신청 권한, 승인 권한, 심의 참여권한(의무) 등등)을 지정한다.
- 각 사용자는 해당 역할과 권한에 따라서 데이터 표준화 관리 체제하에서 활동하게 된다.

Ⅳ. 데이터 모델 관리 : 조직 , 권한, 사용자 – 사용자(사례)



사용자를 등록한다.

기존의 기업에서 사용하고 있는 사용자 정보와의 호환가능

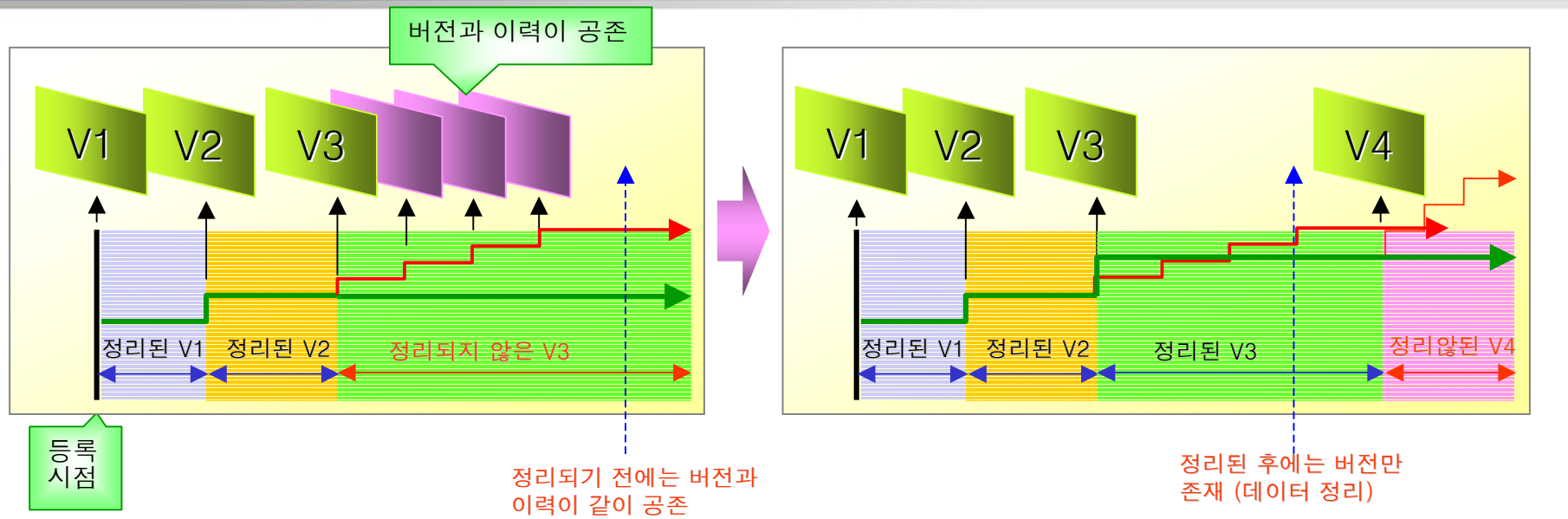


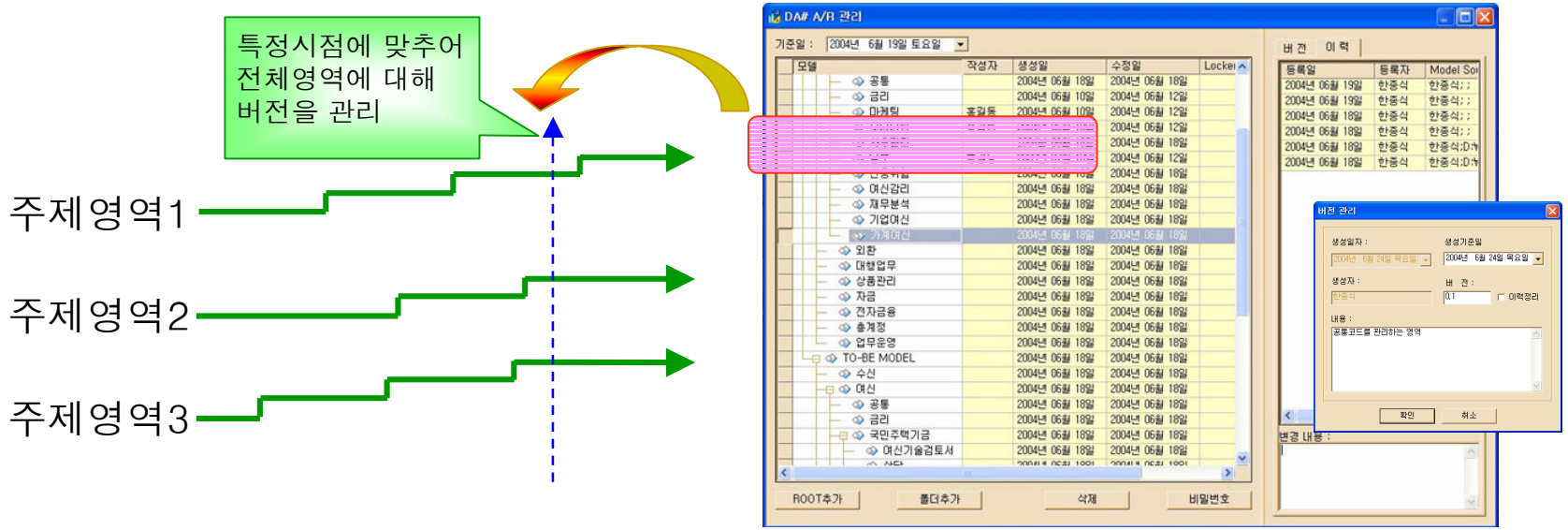
사용자에게 역할을 배정한다.

여된 역할에 따라서 각 사용자 기업의 IT 시스템을 활용하는 환경과 권한이 Setting되게 된다.

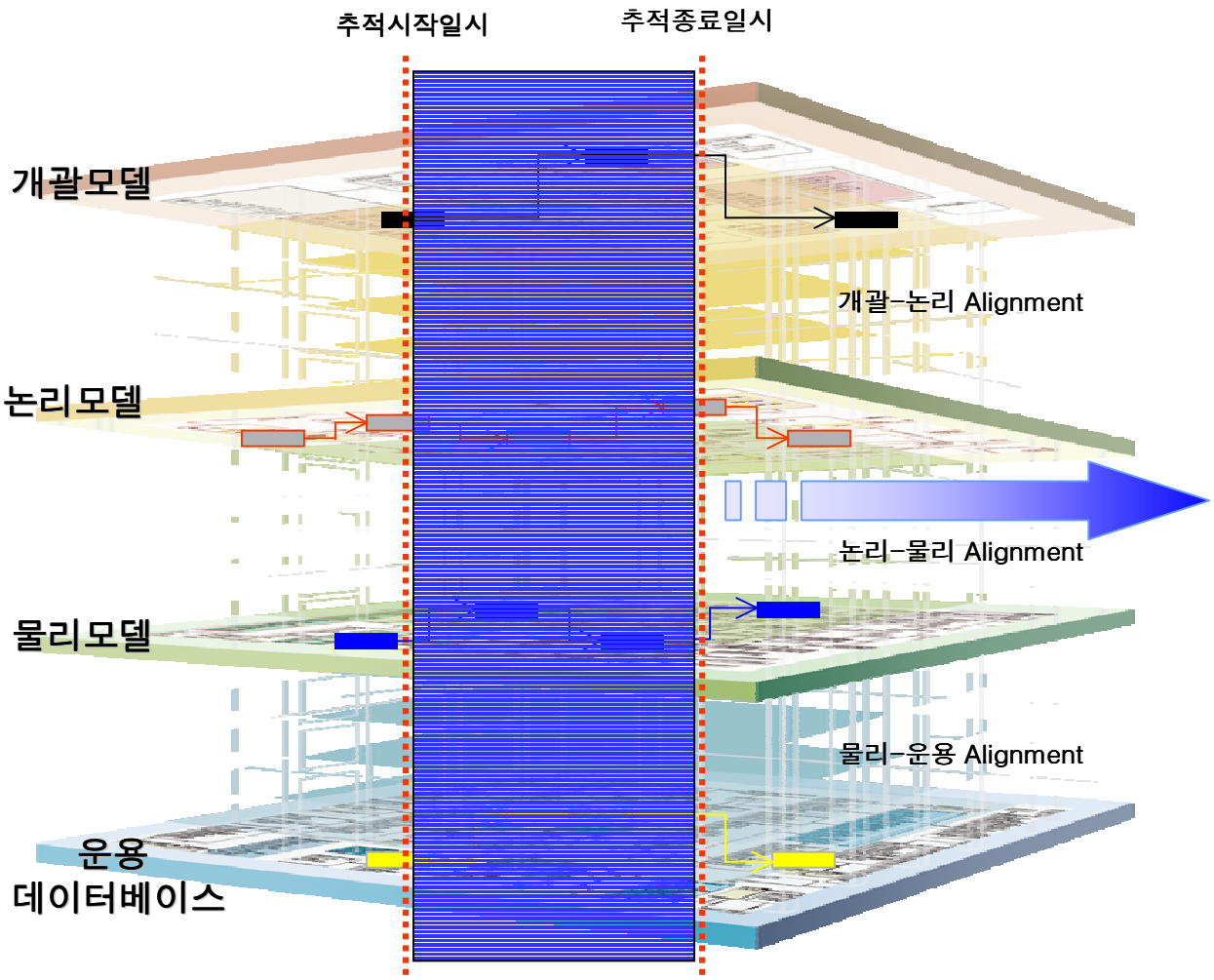
사용자에게 기업이 가지고 있는 모델 구조에 대한 권한을 배정한다.

Ⅴ. 데이터 모델 관리 : 형상 관리





Ⅴ. 데이터 모델 관리 : 형상 관리 – 변경사항 추적

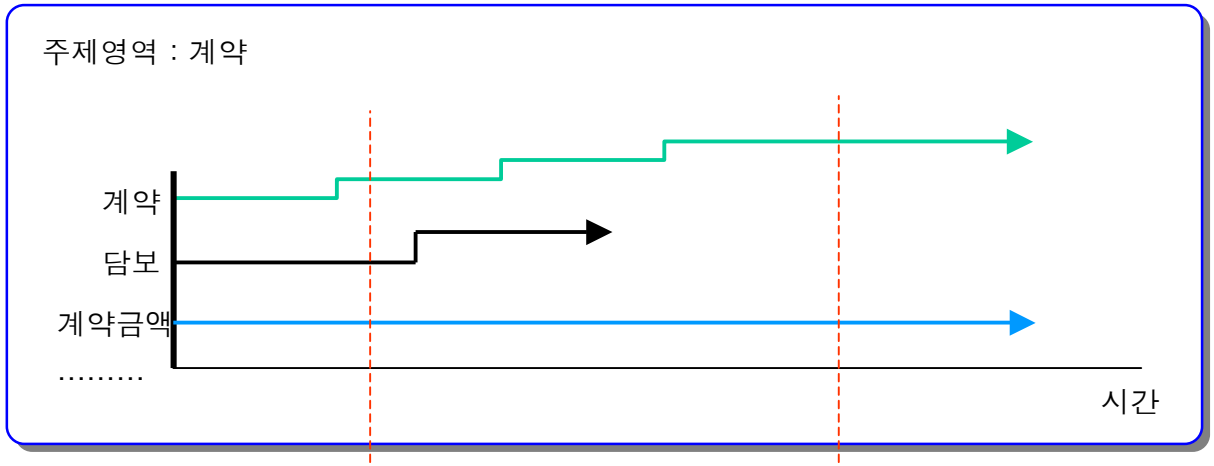


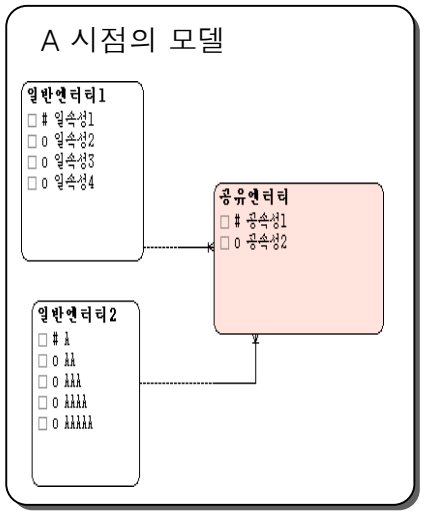
변경사항 추적

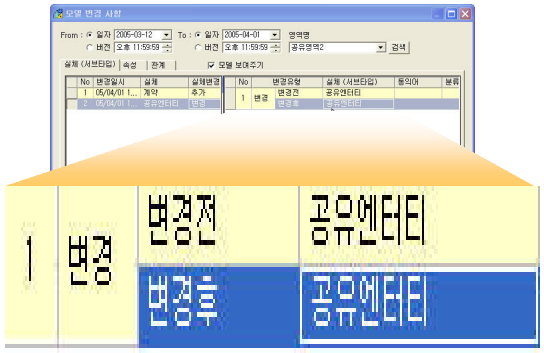
- 변경 데이터객체
- 변경자
- 변경일자
- 변경내용
- 영향도 분석

.
.

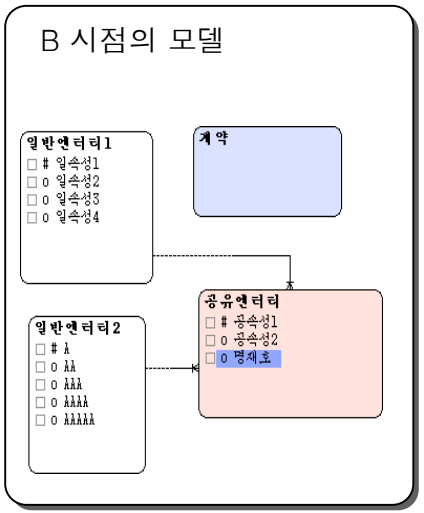
Ⅴ. 데이터 모델 관리 : 형상 관리 – 모델 버전(시점)별 비교







두 시점을 조회하면서 변경사항 확인 가능
과거 시점을 완벽하게 재현 (초 단위 까지 가능)
변경사항을 색으로 구분

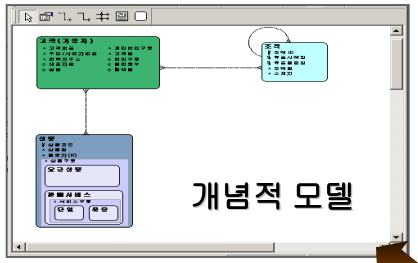


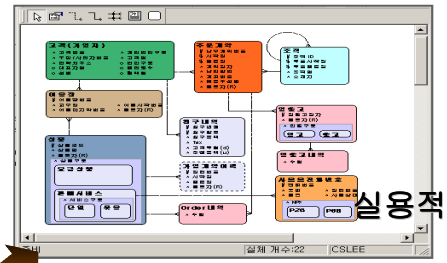
개념적, 본질적, 실용적 모델을 One View로 관리

핵심 엔터티 구성된 데이터의 골격

골격에 본질적인 것만 추가

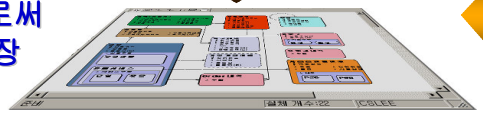
현실을 반영한 최적화





Modeler

툴의 기능으로 각 데이터 모델을 제공함으로써 모델의 일관성 보장

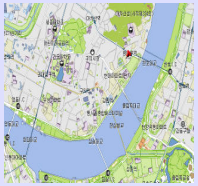


참조레벨

Repository에 One 모델만 저장

개괄적 모델에서 출발

하향식



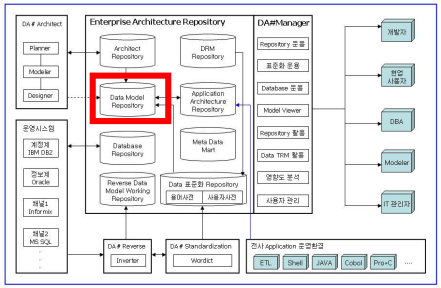
상향식

물리적 테이블에서 출발

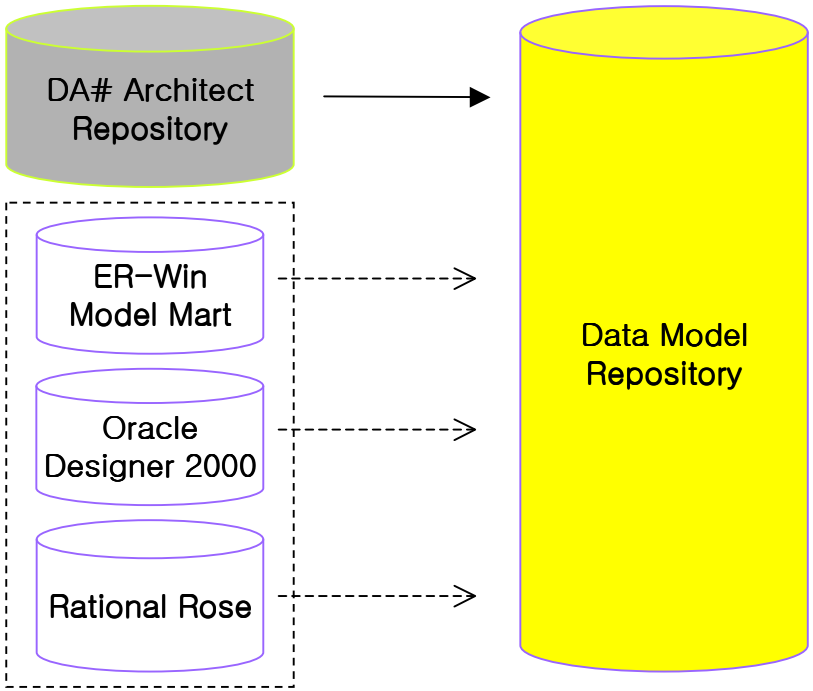
하향식으로 접근하는 경우 수순이 준수 되어야만 오류를 감소 시킨다.

상향식으로 접근하는 경우 하위 모델이확정되면 상위 모델 자동 도출

Ⅵ. 데이터 모델 관리 : 활용 – 리파지토리 관리



- 업무 시스템으로 말하면 DA# Architect Repository는 계정계 데이터 모델이다. 반면 Data Model Repository는 정보계(DW) 데이터 모델이라고 할 수 있다.
- 다양한 형태의 Data Model 관리 툴들로 부터의 Input을 받아 들일 수 있어야 한다.
- 궁극적으로는 기업의 데이터 아키텍쳐를 관리 운용하는 전문적인 툴과는 별개로 데이터 아키텍쳐를 전사의 권한을 가진 사용자들이 쉽게 활용할 수 있는 환경을 구축하는 것이다.



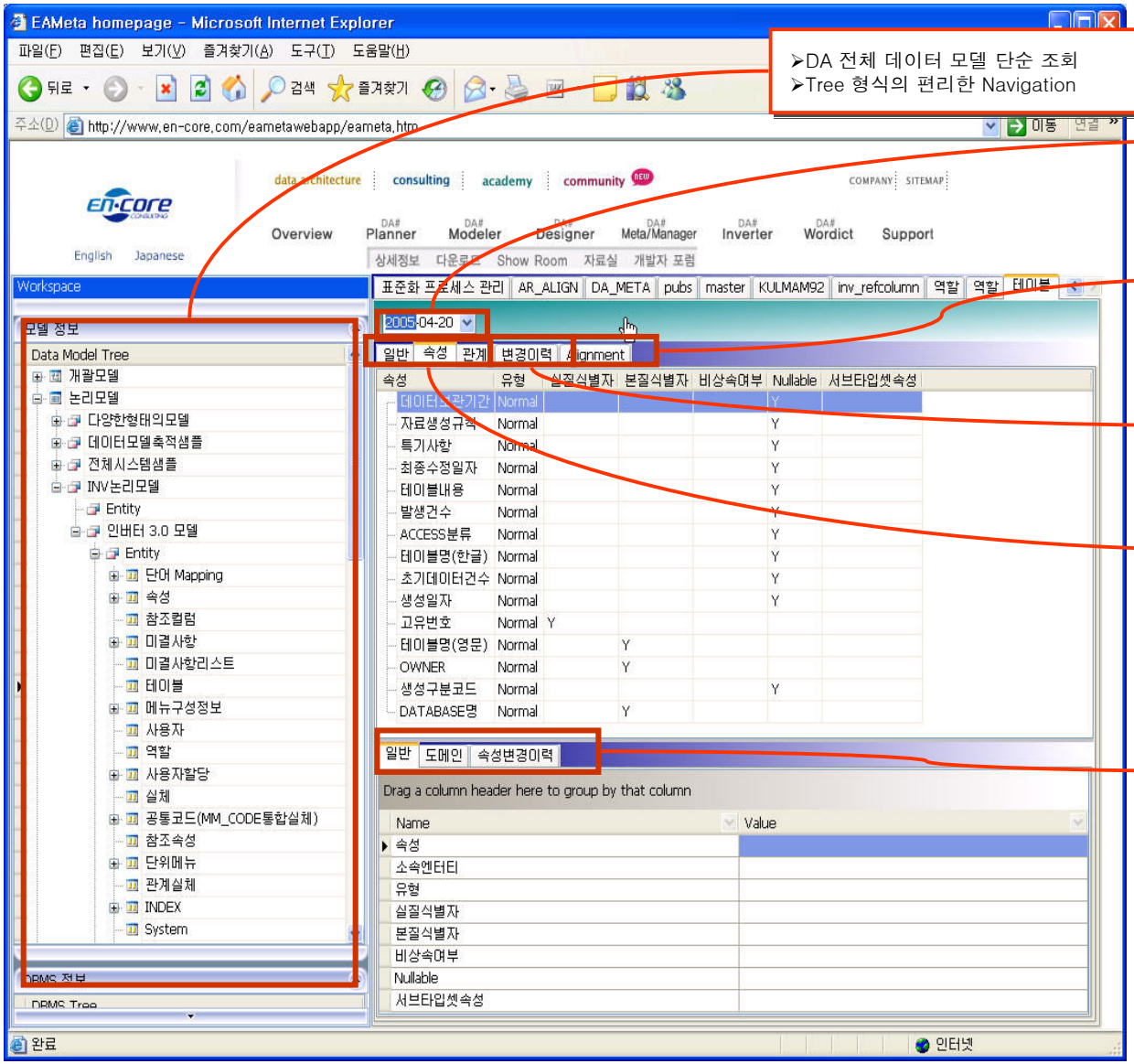
- 데이터 모델의 정보화
- 데이터 모델 각 Layer 간의 완벽한 Alignment
- 초 단위까지의 변경사항에 대한 완벽한 이력관리 (Record Level 이력관리 + 부가적인 정보)
- DA# Architect 뿐만 아니라 다른 외부 툴과의 Interface를 고려한 확장성 확보
- 다양한 종류의 DBMS와의 Alignment를 고려한 Repository Model

물리모델조회 | **논리모델조회** | **개념모델조회**

모델 Alignment 검증 | **모델 변경이력 조회**

전용 Viewer를 통한 모델 활용 효용성 극대화

Ⅵ. 데이터 모델 관리 : 활용 – 배포 및 활용



➢DA 전체 데이터 모델 단순 조회
➢Tree 형식의 편리한 Navigation

➢과거 어느 시점 기준이건 조회 가능
➢초 단위까지의 형상 관리

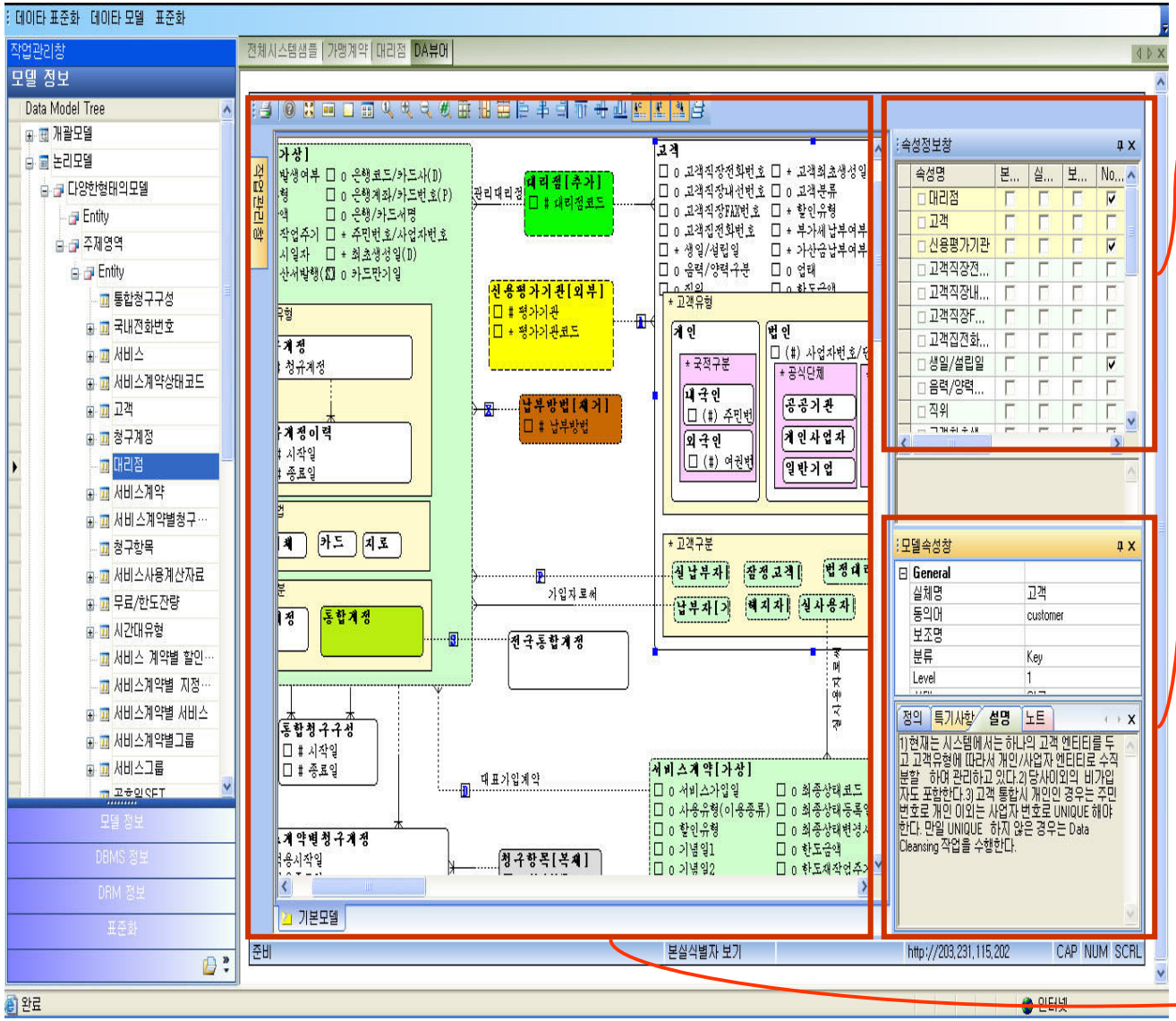
➢자신을 기준으로 자신의 상위,하위 오브젝트 연관관계 조회
➢이러한 연관관계는 Database 오브젝트 레벨까지 확장가능함.

➢하나의 오브젝트를 기준으로 탄생부터 기준일 현재까지의 변경 Histoty 조회

➢엔터티의 속성 정보, 관계 정보를 한눈에 파악 할 수 있음.

➢엔터티 뿐만 아니라 속성에 대한 내용도 다양한 정보를 활용할 수 있음.
➢속성에 대칭된 데이터 표준화의 도메인 정보도 확인
➢속성에 대한 변경사항도 확인함.

VI. 데이터 모델 관리 – 활용(사례)



- 논리 모델의 속성 정보, 물리모델의 컬럼 정보를 조회
- 개발자들이 실제로 참고 하면서 프로그램을 작성함.

- 엔터티 정의 , 테이블 정의
- 업무규칙
- 데이터 모델링시에 생성했던 개발시의 고려, 특기 사항들의 개발자들이 실제 조회하면서 프로그램을 개발

- DA# Modeler가 가지고 있는 모든 획기적인 Navigation 기능을 가지고 있음.
- 위치에 구해받지 않고 데이터 모델을 공유하는 체제 구축
- 데이터 모델을 Paper로 관리, 배포함으로써 발생하는 비효율을 한꺼번에 완벽 해소
- 주제영역별 권한 관리 통해 보완 완벽
- 확대 축소, 다양한 보기 기능 제공

Q & A

Contact Info : jhmyung@en-core.com