1 과목 출제과목 변동(2024 년 기준)

변경 전	
주요항목	세부항목
데이터 모델링의 이해	• 데이터모델의 이해
	• 엔터티
	• 속성
	• 관계
	• 식별자
데이터 모델과 성능	• 정규화와 성능
	• 반정규화와 성능
	• 대용량 데이터에 따른 성능
	• DB 구조와 성능
	• 분산 DB 데이터에 따른 성능

⇒

변경 후	
주요항목	세부항목
데이터 모델링의 이해	• 데이터모델의 이해
	• 엔터티
	• 속성
	• 관계
	• 식별자
데이터 모델과 SQL	• 정규화
	• 관계와 조인의 이해
	• 모델이 표현하는 트랜잭션의 이해
	• Null 속성의 이해
	• 본질식별자 vs 인조식별자

엔터티(Entity), 속성(Attribute), 인스턴스(Instance)에 대해서는 아래와 같이 생각하면 쉽습니다.

<학생테이블>

학번	이름	전화번호	주소	학과번호	지도교수번호
1000	홍길동	010-0000-0000	서울시	101	10000
1001	박길동	010-1111-1111	경기도	102	10001
		.....			

컬럼(속성)

행(인스턴스)

테이블(엔터티)

1-1. 데이터 모델의 의해

● 모델링의 개념

- 현실 세계의 비즈니스 프로세스와 데이터 요구 사항을 추상적이고 구조화된 형태로 표현하는 과정
- 데이터베이스의 구조와 관계를 정의하며, 이를 통해 데이터의 저장, 조작, 관리 방법을 명확하게 정의

● 모델링의 특징

1. 단순화(Simplification)

- 현실을 단순화하여 핵심 요소에 집중하고 **불필요한 세부 사항을 제거**
- 단순화를 통해 복잡한 현실 세계를 이해하고 표현하기 쉬워짐

2. 추상화(Abstraction)

- 현실세계를 일정한 형식에 맞추어 **간략하게** 대략적으로 표현하는 과정
- 다양한 현상을 일정한 양식인 표기법에 따라 표현

3. 명확화(Clarity)

- 대상에 대한 애매모호함을 최대한 제거하고 **정확하게 현상을 기술**하는 과정
- 명확화를 통해 모델을 이해하는 이들의 의사소통을 원활히 함

● 데이터 모델링 유의점

1. 중복(Duplication)

- 한 테이블 또는 여러 테이블에 같은 정보를 저장하지 않도록 설계

2. 비유연성(Inflexibility)
- 사소한 업무 변화에 대해서도 잦은 모델 변경이 되지 않도록 주의
- 데이터 정의를 프로세스와 분리

3. 비일관성(Inconsistency)
- 데이터베이스 내의 정보가 모순되거나 상반된 내용을 갖는 상태를 의미
- 데이터간 상호연관 관계를 명확히 정의
- 데이터 품질 관리 필요
- 데이터의 중복이 없더라도 비일관성은 발생할 수 있음

● 데이터 모델링 3 가지 요소
- 대상(Entity) : 업무가 관리하고자 하는 대상(객체)
- 속성(Attribute) : 대상들이 갖는 속성(하나의 특징으로 정의될 수 있는 것)
- 관계(Relationship) : 대상들 간의 관계

● 데이터 모델링의 3 단계

1. 개념적 모델링
- 업무 중심적이고 포괄적(전사적)인 수준의 모델링
- **추상화 수준이 가장 높음**
- **업무를 분석 뒤 업무의 핵심 엔터티(Entity)를 추출하는 단계**
- 도출된 핵심 엔터티(Entity)들과의 관계들을 표현하기 위해 **ERD 작성**

2. 논리적 모델링
- **개념적 모델링의 결과를 토대로 세부속성, 식별자, 관계 등을 표현하는 단계**
- 데이터 구조를 정의하기 때문에 비슷한 업무나 프로젝트에서 동일한 형태의 데이터 사용 시 **재사용** 가능
- 동일한 논리적 모델을 사용하는 경우 쿼리도 재사용 가능
- **데이터 정규화 수행**
- 재사용성이 높은 논리적 모델은 유지보수가 용이해짐

3. 물리적 모델링
- **논리 모델링이 끝나면 이를 직접 물리적으로 생성하는 과정**
- 데이터베이스 성능, 디스크 저장구조, 하드웨어의 보안성, 가용성 등을 고려
- 가장 구체적인 데이터 모델링
- **추상화 수준은 가장 낮음(가장 구체적인 모델링이므로)**

● 데이터 모델의 표기법(ERD : Entity Relationship Diagram)
- 엔터티(Entity)와 엔터티 간의 관계(Relationship)를 시각적으로 표현한 다이어그램
- 1976 년 피터 첸(Peter Chen)이 만든 표기법, 데이터 모델링 표준으로 사용

● **ERD 작성 절차 (6 단계)**

① 엔터티를 도출한 후 그린다
② 엔터티 배치
③ 엔터티 간의 관계를 설정
④ 관계명을 서술
⑤ 관계의 참여도 기술
⑥ 관계의 필수 여부를 확인

1-2. 엔터티

● **엔터티(Entity)의 개념**

- **현실 세계에서 독립적으로 식별 가능한 객체나 사물을 나타냄**
- 엔터티는 업무상 분석해야 하는 **대상(Instance)들로 이루어진 집합**
- 인스턴스는 엔터티의 특정한 속성 값들로 구성되며, 엔터티의 개념을 현실에서 구체적으로 나타낸 것
  예) 엔터티와 속성, 인스턴스 등의 관계

· **엔터티(Entity) :** 학생
· **속성(Attribute) :** 학번, 이름, 학과 등.
· **식별자(Identifier) :** 학번 (고유한 학번으로 각 학생을 식별)
· **인스턴스 : 특정 학생의 데이터**
  - **학번:** 2021001
  - **이름:** 홍길동
  - **학과:** 컴퓨터 공학

● **엔터티(entity)의 특징**

1. 유일한 식별자에 의해 식별 가능
   - 인스턴스가 식별자에 의해 한 개씩만 존재하는 지 검증 필요
   - 유일한 식별자는 그 엔터티의 인스턴스만의 고유 이름
     ex) 이름은 동명이인이 있을 수 있으므로 사번, 학번 등이 고유식별자

2. 해당 업무에 필요하고 관리하고자 하는 정보
   - 설계하는 업무의 시스템 구축에 필요한 정보여야 함
     ex) 학교 시스템 구축 시 학생정보 필요. 다른 업무엔 학생 정보 불필요.

3. 인스턴스들의 집합
   - 영속적으로 존재하는 2 개 이상의 인스턴스의 집합
   - 인스턴스가 한 개 밖에 없는 엔터티는 집합이 아니므로 성립이 안됨.

4. 엔터티는 반드시 속성을 가짐
   - 각 엔터티는 2 개 이상의 속성을 가짐
   - 하나의 인스턴스는 각각의 속성들에 대한 1 개의 속성 값만을 가짐
     ex) 학생 엔터티에서 한 학생의 데이터(인스턴스)의 이름(속성) 정보에는 반드시 한 값만 저장됨

5. 엔터티는 업무 프로세스에 의해 이용
   - 업무적으로 필요해 선정했지만 실제 사용되지 않으면 잘못 설계된 것
   - 모델링 시 발견하기 어려운 경우 데이터 모델 검증이나 상관 모델링 시 단위 프로세스 교차점검으로 문제 도출
   - 누락된 프로세스의 경우 추후 해당 프로세스 추가
   - 반대로 사용되지 않는 고립 엔터티는 제거 필요

6. 다른 엔터티와 최소 1 개 이상의 관계 성립
   - 엔터티는 업무적 연관성을 갖고 다른 엔터티와 연관의 의미를 가짐
   - 관계가 없는 엔터티 도출은 부적절한 엔터티이거나 적절한 관계를 찾지 못한것

● **엔터티의 분류**

1) 유형과 무형에 따른 분류

① **유형엔터티**
- **물리적 형태가 있음(실체가 있는 대상)**
- 안정적이며 지속적으로 활용되는 엔터티
- 업무로부터 구분하기가 가장 용이한 엔터티

  ex) 사원, 물품, 감사 등

② **개념엔터티**
- **물리적인 형태 없음**
- 관리해야 할 개념적 정보로부터 구분되는 엔터티

  ex)조직, 보험상품 등

③ **사건엔터티**
- **업무를 수행에 따라 발생하는 엔터티**
- 발생량이 많고 각종 통계자료에 이용

  ex) 주문, 청구, 미납 등

2) 발생 시점에 따른 분류

① **기본엔터티**
- **그 업무에 원래 존재하는 정보**
- 다른 엔터티와 관계에 의해 생성되지 않고 독립적으로 생성
- 타 엔터티의 부모 역할을 하는 엔터티
- 다른 엔터티로부터 주식별자를 상속받지 않고 **자신의 고유한 주식별자를 가짐**

  ex) 사원, 부서, 고객, 상품 등

② **중심엔터티**
- **기본엔터티로부터 발생되고 그 업무에서 중심적인 역할**
- 많은 데이터가 발생되고 다른 엔터티와의 관계를 통해 많은 행위 엔터티를 생성

  ex) 계약, 사고, 청구, 주문, 매출 등

③ **행위엔터티**
- **2 개 이상의 부모엔터티로부터 발생**
- 자주 내용이 바뀌거나 데이터 양이 증가
- 분석 초기 단계보다는 상세 설계 단계나 프로세스와 상관모델링을 진행하면서 도출

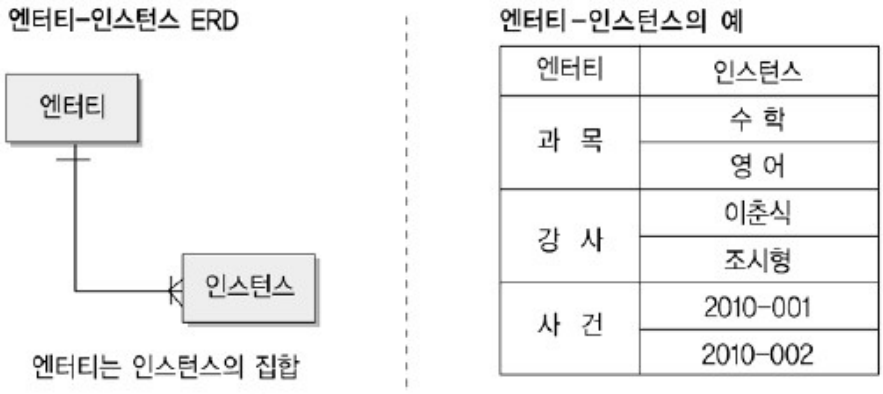
  ex) 주문(고객과 상품 엔터티로 부터 발생하므로 행위엔터티이기도 함), 사원변경이력, 이력 등

● **엔터티의 명명**

① 현업에서 사용하는 용어 사용
② 가능하면 약자 사용은 자제
③ 단수 명사 사용
④ 모든 엔터티에서 유일하게 이름 부여
⑤ 엔터티 생성 의미대로 이름 부여

● **엔터티와 인스턴스 표기법**

- 엔터티는 사각형으로 표현, 속성은 조금씩 다름



[그림 Ⅰ-1-15] 엔터티와 인스턴스

출처 : [SQLD : Ⅰ. 데이터 모델링의 이해] 1-2. 엔터티와 인스턴스, 표기법 (tistory.com)

[그림 Ⅰ-1-16] 엔터티에 대한 표기법

출처 : [SQLD : Ⅰ. 데이터 모델링의 이해] 1-2. 엔터티와 인스턴스, 표기법 (tistory.com)

1-3. 속성

● 속성(Attribute)의 개념

- 속성은 **업무에서 필요로 하는 고유한 성질, 특징을 의미(관찰 대상) -> 컬럼으로 표현할 수 있는 단위!**
- 업무상 인스턴스로 관리하고자 하는 **더 이상 분리되지 않는 최소의 데이터 단위**
- 인스턴스의 구성 요소

  ex) 학생 엔터티에 이름, 학번, 학과번호 등이 속성이 될 수 있음

● 엔터티, 인스턴스, 속성, 속성값의 관계

- 한 개의 엔터티는 2 개 이상의 인스턴스의 집합이어야 한다(하나의 테이블은 두 개 이상의 행을 가짐)
- 한 개의 엔터티는 2 개 이상의 속성을 갖는다(하나의 테이블은 두 개 이상의 컬럼으로 구성됨)
- 한 개의 속성은 1 개의 속성값을 갖는다(각 컬럼의 값은 하나씩만 삽입 가능)
- 속성은 엔터티에 속한 엔터티에 대한 자세하고 구체적인 정보를 나타냄, **각 속성은 구체적인 값을 가짐**

● 속성의 특징

- 반드시 해당 업무에서 필요하고 관리하고자 하는 정보여야 한다
- **정해진 주식별자에 함수적 종속성을 가져야 한다**
- 하나의 속성은 한 개의 값만을 가진다.(한 컬럼의 값은 각 인스턴스마다 하나씩만 저장)
- 하나의 속성에 여러 개의 값이 있는 다중값일 경우 별도의 엔터티를 이용하여 분리한다
- 하나의 인스턴스는 속성마다 반드시 하나의 속성값을 가진다

=> 각 속성이 하나의 값을 갖고 있음을 의미(속성의 **원자성**)

**** 원자성이란**

- 데이터모델에서 각 엔터티의 **인스턴스가 해당 속성에 대해 단일하고 명확한 값을 가지는 것을 의미**

● 함수적 종속성

- 한 속성의 값이 다른 속성의 값에 종속적인 관계를 갖는 특징을 말함
- 즉, **어떤 속성 A 의 값에 의해 다른 속성 B 도 유일하게 결정된다면**, B 는 A 에 함수적으로 종속됐다 하고,
- 이를 수식으로 나타내면 A → B 라고 표현함

1) **완전 함수적 종속**

- 특정 컬럼이 기본키에 대해 완전히 종속될 때를 말함
- PK 를 구성하는 컬럼이 2 개 이상일 경우 PK 값 모두에 의한 종속관계를 나타낼 때 완전 함수 종속성 만족

  ex) (주문번호 + 제품번호)에 의해 수량 컬럼의 값이 결정됨

주문 PK

주문번호	제품번호	수량
1	100	4
1	101	1
2	100	3
2	101	2

2) **부분 함수적 종속**

- 기본키 전체가 아니라, 기본키 일부에 대해 종속될 때를 말함

ex) 수강기록 테이블에서 학생번호와 과목이 PK 라고 가정할 때, 과목에 의해서도 교수가 결정되면 부분 함수적 종속 관계!

수강기록 PK

학생번호	과목	강사
1001	수학	홍쌤
1002	수학	홍쌤
1001	과학	최쌤
1003	영어	김쌤

● **속성의 분류**

1) 속성의 특성에 따른 분류

① 기본 속성
- 업무로부터 추출된 모든 속성
- 엔터티에 가장 일반적으로 많이 존재하는 속성

ex) 원금, 예치기간 등

② 설계 속성
- 기본 속성 외에 업무를 규칙화하기 위해 새로 만들어지거나 기본 속성을 변형하여 만들어지는 속성

ex) 상품코드, 지점코드, 예금분류 등

③ 파생 속성
- 다른 속성에 의해 만들어지는 속성
- 일반적으로 계산된 값들이 해당
- 데이터 정합성을 유지하기 위해 가급적 적게 정의하는 것이 좋음

ex) 합계, 평균, 이자 등

2) 엔터티 구성방식에 따른 분류

① PK(Primary Key, 기본키)
- 인스턴스를 식별할 수 있는 속성

② FK(Foreign Key, 외래키) 속성
- 다른 엔터티와의 관계에서 포함된 속성

③ 일반 속성
- 엔터티에 포함되어 있고 PK/FK 에 포함되지 않는 속성

3) 분해 여부에 따른 속성

① 단일 속성
- 하나의 의미로 구성된 경우

ex) 회원 ID, 이름 등

② 복합 속성
- 여러개의 의미로 구성된 경우
  ex) 주소(시, 구, 동 등으로 분해 가능) 등

③ 다중값 속성
- 속성에 여러 개의 값을 가질 수 있는 경우
- 다중값 속성은 엔터티로 분해
  ex) 상품 리스트 등

● **속성의 명명규칙**
① 해당 업무에서 사용하는 이름을 부여
② 서술식 속성명은 사용하지 않음
③ 약어의 사용은 가급적 제한
④ 전체 데이터 모델에서 유일한 명칭

● **도메인(Domain)**
- 도메인은 각 속성이 가질 수 있는 값의 범위를 의미함
- 엔터티 내에서 속성에 대한 데이터 타입과 크기, 제약사항을 지정하는 것이다

1-4. 관계

● **관계(Relationship)의 개념**

- **관계는 엔터티간의 연관성을 나타낸 개념**
- 관계를 정의할 때는 인스턴스(각 행 데이터)간의 논리적인 연관성을 파악하여 정의
- 엔터티를 어떻게 정의하느냐에 따라 변경되기도 함

● **관계의 종류**

1) **존재적 관계**

- 한 엔터티의 존재가 다른 엔터티의 존재에 영향을 미치는 관계
- 엔터티 간의 연관된 상태를 의미

ex) 부서 엔터티가 삭제되면 사원 엔터티의 존재에 영향을 미침

2) **행위적 관계**

- 엔터티 간의 어떤 행위가 있는 것을 의미

ex) 고객 엔터티의 행동에 의해 주문 엔터티가 발생

※ ERD 에서는 존재관계와 행위관계를 구분하지 않는다.

● **관계의 구성**

1. 관계명
2. 차수(Cardinality)
3. 선택성(Optionality)

● **관계의 차수 (Cardinality)**

- **한 엔터티의 레코드(인스턴스)가 다른 엔터티의 레코드(인스턴스)와 어떻게 연결되는지를 나타내는 표현**
- 주로 1:1, 1:N, N:M 등으로 표현

1) 1 대 1 관계

· **완전 1 대 1 관계**

- 하나의 엔터티에 관계되는 엔터티가 반드시 하나로 존재하는 경우

ex) 사원은 반드시 소속 부서가 있어야 함

· **선택적 1 대 1 관계**

- 하나의 엔터티에 관계되는 엔터티가 하나이거나 없을 수 있는 경우

ex) 사원은 하나의 소속 부서가 있거나 아직 발령전이면 없을 수 있음

2) 1 대 N 관계

- 엔터티에 하나의 행에 다른 엔터티의 값이 여러 개 있는 관계

ex) 고객은 여러 개 계좌를 소유할 수 있음.

3) M 대 N 관계

- 두 엔터티가 다대다의 연결 관계 가지고 있음
- 이 경우 조인 시 카테시안 곱이 발생하므로 **두 엔터티를 연결하는 연결엔터티의 추가로 1 대 N 관계로 해소할 필요가 있음**

  ex) 한 학생이 여러 강의를 수강할 수 있고, 한 강의 기준으로도 여러 학생이 보유할 수 있음

  => 이 두 엔터티의 연결엔터티로는 구매이력 엔터티가 필요함

● **관계의 페어링**

- **엔터티 안에 인스턴스가 개별적으로 관계를 가지는 것**
- 관계란 페이링의 집합을 의미함

**** 관계와 차수, 페어링 차이**

- 학생과 강의 엔터티는 관계를 가짐
- 한 학생은 여러 강의를 수강할 수 있고, 한 강의도 여러 학생에게 수강될 수 있으므로 M 대 N 관계이며, 이 때 차수는 M:N 가 됨
- 인스턴스의 관계를 보면 "학생 A 가 강의 B 를 2023 년 1 학기에 수강했고 성적은 'A+'를 받았다"와 같은 특정한 페어링이 형성
- 이런식으로 관계의 **차수는 하나의 엔터티와 다른 엔터티 간의 레코드 연결 방식**을 나타내는 반면, **관계의 페어링은 두 엔터티 간의 특정 연결을 설명**하고 추가 정보를 제공하는 용도로 사용.

1-5. 식별자

● 식별자 개념

- 하나의 엔터티에 구성된 여러 개의 속성 중에 엔터티를 대표할 수 있는 속성을 나타냄
- 하나의 유일한 유일한 식별자가 존재해야 함
- 식별자는 논리 모델링에서 사용하는 용어, 물리 모델링에서는 키(key)라고 표현

  ex) 학생 엔터티의 주식별자는 학생번호 속성 => 학생 테이블의 기본키는 학생번호 컬럼

  (논리 모델링) (물리 모델링)

● 주식별자 특징

① 유일성 : 주식별자에 의해 모든 인스턴스를 유일하게 구분함

ex) 학생 엔터티에서 이름 속성은 동명이인이 발생할 수 있으므로 모든 인스턴스를 완벽하게 구분할 수 없으므로 학생번호와 같은 유일한 식별자를 주식별자로 사용

② 최소성 : 주식별자를 구성하는 속성은 유일성을 만족하는 최소한의 속성으로 구성

ex) 학생 엔터티의 주식별자는 학생번호만으로 충분한데, 학생번호 + 이름으로 구성할 필요 없음

③ 불변성: 주식별자가 한번 특정 엔터티에 지정되면 그 식별자의 값은 변하지 않아야 함

(항상 고유값으로 존재해야 함)

ex) 학생 엔터티에 주식별자인 학생번호가 때에 따라 변경되서는 안됨

④ 존재성 : 주식별자가 지정되면 반드시 값이 존재해야 하며 NULL 은 허용 안 됨

● 식별자 분류

1) 대표성 여부에 따른 식별자의 종류

주식별자	보조식별자
• 유일성과 최소성을 만족하면서 엔터티를 대표하는 식별자 • 엔터티 내에서 각 인스턴스를 유일하게 구분할 수 있는 식별자 • 타 엔터티와 참조관계를 연결할 수 있는 식별자	• 엔터티 내에서 각 인스턴스를 구분할 수 있는 구분자지만, 대표성을 가지지 못해 참조 관계 연결을 할 수 없는 식별자 • 유일성과 최소성은 만족하지만 대표성을 만족하지 못하는 식별자

2) 생성 여부에 따른 식별자의 종류

내부식별자	외부식별자
• 다른 엔터티 참조 없이 엔터티 내부에서 스스로 생성되는 식별자	• 다른 엔터티와 관계로 인하여 만들어지는 식별자 (외래키)

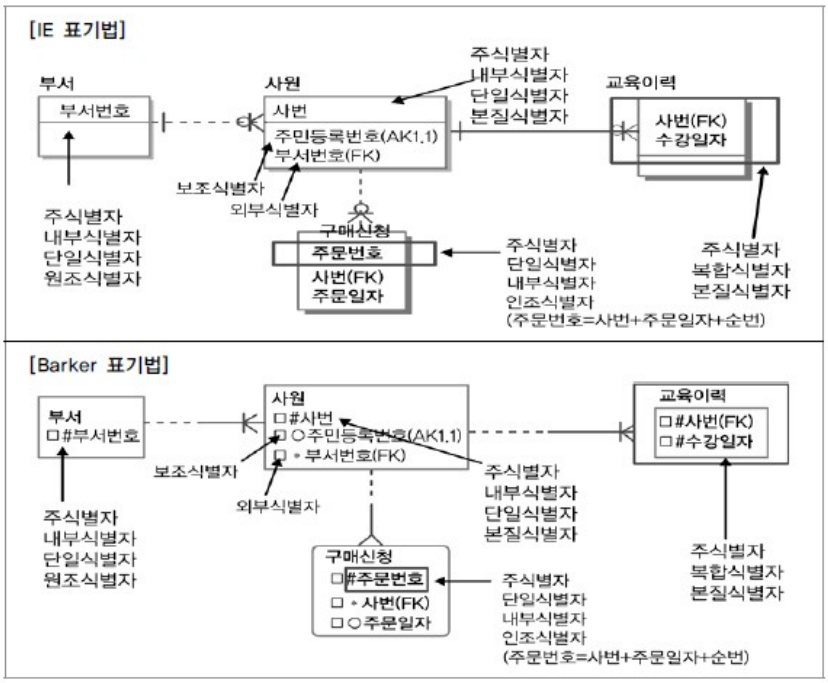
3) 속성 수에 따른 식별자 종류

단일식별자	복합식별자
• 하나의 속성으로 구성	• 2 개 이상의 속성으로 구성

4) 대체 여부에 따른 식별자의 종류

본질식별자	인조식별자
• 비즈니스 프로세스에서 만들어지는 식별자	• 인위적으로 만들어지는 식별자 • 자동 증가하는 일련번호 같은 형태

● 식별자 표기법



[그림 Ⅰ-1-42] 식별자의 분류-데이터 모델

● 주식별자 도출기준

1) 해당 업무에서 자주 이용되는 속성을 주식별자로 지정한다.

- 같은 식별자 조건을 만족하더라도 업무적으로 더 많이 사용되는 속성을 주식별자로 지정

ex) 학생번호와 주민번호 중에 학생번호가 주식별자, 주민번호는 보조식별자

2) 명칭이나 내역등과 같은 이름은 피함
  - 이름 자체를 주식별자로 사용하는 행위를 피함
    ex) 부서명 보다는 부서코드를 부여하여 부서코드로 주식별자로 사용

3) 속성의 수를 최대한 적게 구성
  - 주식별자를 너무 많은 속성으로 구성 시, 조인으로 인한 성능저하 발생 우려
  - 일반적으로 7~8 개 이상의 주식별자 구성은 새로운 인조식별자를 생성하여 모델을 단순화 시키는 것이 좋음
    ex) 주문 엔터티에 대해 주문일자 + 주문상품코드 + 고객번호 + ... 등으로 구성 => 주문번호 속성 추가!

● 관계간 엔터티 구분

1) 강한 개체
  - 독립적으로 존재할 수 있는 엔터티
    ex) 고객과 계좌 엔터티 중, 고객은 독립적으로 존재할 수 있음

2) 약한 개체
  - 독립적으로 존재할 수 없는 엔터티
    ex) 고객과 계좌 엔터티 중, 계좌은 독립적으로 존재할 수 없음(고객에 의해 파생되는 엔터티)

● 식별 관계와 비식별관계

1) 식별관계(Identification Relationship)
  - 하나의 엔터티의 기본키를 다른 엔터티가 기본키의 하나로 공유하는 관계
  - 식별관계는 ERD 에서 실선으로 표시
    ex) 사원과 교육이력 엔터티에서 양쪽 모두 기본키 중 일부가 사원번호임

사원	교육이력
# 사원번호	# 사원번호(FK)
	# 수강일자

2) 비식별관계(Non-identification Relationship)
  - 강한 개체의 기본키를 다른 엔터티의 기본키가 아닌 일반 속성으로 관계를 가지는 것
  - 비식별관계는 ERD 에서 점선으로 표시
    ex) 부서와 사원의 관계에서 부서의 부서번호(기본키)를 사원 엔터티에서는 일반키로 가짐
    (사원에서는 사원번호가 기본키)

1-6. 정규화

모델링 시 최대한 중복 데이터를 허용하지 않아야 저장공간의 효율적 사용과 업무 프로세스의 성능을 기대할 수 있다. 이러한 중복 데이터를 허용하지 않는 방식으로 테이블을 설계하는 방식을 정규화라고 한다.

● **정규화(DB Normalization)의 개념**

- 하나에 엔터티에 많은 속성을 넣게 되면, 해당 엔터티를 조회할 때 마다 많은 양의 데이터가 조회될 것이므로 **최소한의 데이터만을 하나의 엔터티에 넣는식으로 데이터를 분해하는 과정**을 정규화라고 한다
- 데이터의 일관성, 최소한의 데이터 중복, 최대한의 데이터 유연성 위한 과정이라고 볼 수 있음
- 데이터의 **중복을 제거하고 데이터 모델의 독립성을 확보**
- 데이터 **이상현상을 줄이기 위한** 데이터 베이스 설계 기법
- 엔터티를 상세화하는 과정으로 **논리 데이터 모델링 수행 시점**에서 고려됨
- 제 1 정규화부터 제 5 정규화까지 존재, 실질적으로는 제 3 정규화까지만 수행

● **이상현상(Abnormality)**

- **정규화를 하지 않아 발생하는 현상(삽입이상, 갱신이상, 삭제이상)**
- 특정 인스턴스가 삽입 될 때 정의되지 않아도 될 속성까지도 반드시 입력되어야 하는(**삽입이상**) 현상이 발생함
  ex) 만약 사원 + 부서 엔터티를 합쳐 놓고 사원번호, 사원이름, 전화번호, 부서번호, 부서명, 부서위치의 속성이 존재할 때 새로운 사원 값이 추가될 때 정해지지 않은 부서정보(부서번호, 부서명, 부서위치) 모두 임의값 또는 NULL 삽입되야함. 반대로 부서가 새로 추가 될 경우 소속 사원이 없어도 사원과 관련된 모든 속성이 불필요하게 값이 입력되어야 함
- 불필요 한 값까지 입력해야 되는 현상을 삽입이상, 그 외 갱신이상, 삭제이상이 발생할 수 있음
  ex) 부서 정보만 삭제하면 되는데 관련된 사원 정보까지도 함께 삭제되는 현상(**삭제이상**)

● **정규화 단계**

1. 제 1 정규화(1NF)
   - 테이블이 컬럼이 원자성(한 속성이 하나의 값을 갖는 특성)을 갖도록 테이블을 분해하는 단계
   - 쉽게 말해 하나의 행과 컬럼의 값이 반드시 한 값만 입력되도록 행을 분리하는 단계

   예시) 구매 테이블의 제 1 정규화
   상품에 여러 값이 있으므로 이를 여러 인스턴스로 분해

이름	구매상품
홍길동	샴푸, 린스
박길동	우유,치즈
최길동	세제
김길동	우유

→

이름	구매상품
홍길동	샴푸
홍길동	린스
박길동	우유
박길동	치즈
최길동	세제
김길동	우유

□ ○ 구매품목 1
□ ○ 구매품목 2
□ ○ 구매품목 3
□ ○ 구매품목가격 1
□ ○ 구매품목가격 2
□ ○ 구매품목가격 3

   ☞ 홍길동과 박길동은 구매상품이 두 값이 입력되어 있으므로 이를 각각 두 행으로 분리하는 작업을 거쳐야 함!

2. 제 2 정규화(2NF)

- 제 1 정규화를 진행한 테이블에 대해 완전 함수 종속을 만들도록 테이블을 분해
- **완전 함수 종속이란**, 기본키를 구성하는 모든 컬럼의 값이 다른 컬럼을 결정짓는 상태
- 기본키의 부분 집합이 다른 컬럼과 1:1 대응 관계를 갖지 않는 상태를 의미
- 즉, PK(Primary Key)가 2 개 이상일 때 발생하며 PK 의 일부와 종속되는 관계가 있다면 분리한다.

예시) 수강이력 테이블의 제 2 정규화
기본키(학생번호 + 강의명)중, 강의명에 의해 강의실이 결정 -> 완전 함수 종속성 위배
(부분 함수 종속성을 가짐)
-> PK 와 부분 함수 종속성을 갖는 컬럼을 각각 다른 테이블로 분해!

수강이력

학생번호	강의명	강의실	성적
1000	컴퓨터공학	자연관	A
1001	컴퓨터공학	자연관	B
1002	기초통계	본관	A+
1000	데이터베이스	자연관	B+
1001	경영학	본관	C
1003	경영학	본관	A

수강이력

학생번호	강의명	성적
1000	컴퓨터공학	A
1001	컴퓨터공학	B
1002	기초통계	A+
1000	데이터베이스	B+
1001	경영학	C
1003	경영학	A

강의실

강의명	강의실
컴퓨터공학	자연관
기초통계	본관
데이터베이스	자연관
경영학	본관

☞ 수강이력에서는 한 학생이 여러 강의를 수강할 수 있기 때문에 주식별자는 학생번호로만는 불가능(유일성 불만족 때문) 따라서 학생번호와 강의명과 결합되어 주식별자가 되어야 한다(한 학생이 같은 강의는 수강할 수 없다고 가정) 이 때, 주식별자의 부분집합인 **강의명에 의해 강의실이 달라지는 1 대 1 대응관계**를 갖는것을 완전 함수 종속성 위배, 같은 말로 부분 함수 종속 관계라고 하는데. **제 2 정규화는 이러한 부분 함수 종속성을 깨는 것을 목표**로 한다. 따라서 주식별자를 분리할 수 없으니 주식별자는 수강이력에 그대로 있고, 문제가 되는 강의실 컬럼을 주식별자와 분리!

3. 제 3 정규화(3NF)

- 제 2 정규화를 진행한 테이블에 대해 이행적 종속을 없애도록 테이블을 분리
- **이행적 종속성이란 A -> B, B-> C 의 관계가 성립할 때, A -> C 가 성립되는 것을 말함**
- **(A,B)와 (B,C)로 분리하는 것이 제 3 정규화**

예시) 구매 테이블 제 3 정규화

고객번호에 의해 상품명이 결정, 상품명에 의해 가격이 결정되는데

고객번호에 의해서도 구매 가격이 결정됨(고객이 상품을 결정하면 그에 매칭되는 가격이 결정되는 구조이므로)

따라서 (고객번호 + 상품명)과 (상품명 + 가격)으로 분리하는 것이 제 3 정규화!

구매

고객번호	상품명	가격
1001	우유	2500
1002	치즈	3500
1003	소시지	4000
1004	우유	2500

구매

고객번호	상품명
1001	우유
1002	치즈
1003	소시지
1004	우유

상품

상품명	가격
우유	2500
치즈	3500
소시지	4000

☞ 이 경우 테이블을 분리하지 않으면, 구매 테이블에서 상품명을 변경해야 하는 상황이 발생할 경우 그 때마다 구매 테이블에서도 가격을 변경해야 한다. 하지만 제 3 정규화를 진행하여 테이블을 분리하게 되면, 구매 테이블에서의 상품명만 변경하면 되므로 업데이트에 비효율성이 줄어든다!

예시) 학생 테이블의 제 3 정규화

학번은 과목의 결정자이며, 과목은 교수의 결정자이다. 이 때, 학번이 달라지면 그 학번에 의한 교수가 달라지므로 학번 역시 교수의 결정자라고 얘기 할 수 있다. 따라서 전공과 교수 컬럼을 분리해야 함.

학생 테이블에서 교수정보가 삭제되고, 따로 과목테이블이 생기면서 **교수**의 결정자인 **전공**과 함께 들어간다.

학생

학번	전공	교수
1001	통계학	홍교수
1002	수학	김교수
1003	경제학	최교수
1004	경영학	박교수

학생

학번	전공
1001	통계학
1002	수학
1003	경제학
1004	경영학

과목

전공	교수
통계학	홍교수
수학	김교수
경제학	최교수
경영학	박교수

예시) 계좌번호 제 3 정규화

계좌 테이블(분리전)에서 계좌번호가 관리점코드의 결정자이며, 관리점코드 역시 관리점의 결정자인 상태에서 계좌번호에의해 관리점도 달라지므로 계좌번호 역시 관리점에 대한 결정자이다. 이 때는 PK 외 두 속성을 분리, 따라서 관리점이 계좌 테이블에서 삭제되고, 따로 관리점 테이블로 분리되면서 이의 결정자인 관리점코드가 따라감

계좌

계좌번호	예수금	관리점코드	관리점
100111	100	1000	서울점
100222	200	1001	경기점
100333	300	1002	인천점
100444	400	1003	제주점

계좌

계좌번호	예수금	관리점코드
100111	100	1000
100222	200	1001
100333	300	1002
100444	400	1003

관리점

관리점코드	관리점
1000	서울점
1001	경기점
1002	인천점
1003	제주점

※ **결정자와 종속관계**

만약 A 속성이 B 속성의 값을 결정하게 되면, 이 때 A는 B의 결정자라고 하며, 반대로 B는 A에 종속된다 표현함. 따라서 위 예제에서는 고객번호가 상품명을 결정자이며, 상품명 역시 가격의 결정자이다.

4. BCNF(Boyce-Codd Normal Form) 정규화
   - 모든 결정자가 후보키가 되도록 테이블을 분해하는 것(결정자가 후보키가 아닌 다른 컬럼에 종속되면 안됨)

5. 제 4 정규화
   - 여러 컬럼들이 하나의 컬럼을 종속시키는 경우 분해하여 다중값 종속성을 제거

6. 제 5 정규화
   - 조인에 의해서 종속성이 발생되는 경우 분해

● **반정규화=역정규화(De-Normalization)의 개념**

- 데이터베이스의 성능 향상을 위해 데이터 중복을 허용하고 조인을 줄이는 데이터베이스 성능 향상 방법
- 시스템의 성능 향상, 개발 및 운영의 단순화를 위해 정규화된 데이터 모델을 중복, 통합, 분리하는 데이터 모델링 기법
- 조회(SELECT) 속도를 향상시키지만, 데이터 모델의 유연성은 낮아짐

※ 비정규화는 정규화를 수행하지 않음을 의미

● **반정규화 수행 케이스**

- 정규화에 충실하여 종속성, 활용성은 향상되지만 수행 속도가 느려지는 경우
- 다량의 범위를 자주 처리해야 하는 경우
- 특정 범위의 데이터만 자주 처리하는 경우
- 요약/집계 정보가 자주 요구되는 경우

1-7. 관계와 조인의 이해

● **관계(Relationship)의 개념**

- **엔터티의 인스턴스 사이의 논리적인 연관성**
- 엔터티의 정의, 속성 정의 및 관계 정의에 따라서도 다양하게 변할 수 있음
- **관계를 맺는다는 의미는 부모의 식별자를 자식에 상속하고, 상속된 속성을 매핑키(조인키)로 활용**
  -> 부모, 자식을 연결함

사원 테이블

사번	이름	부서번호
1000	홍길동	100
1001	일지매	101
1002	강감찬	201

부서 테이블

부서번호	부서명	위치
100	영업부	서울
101	관리부	서울
201	생산부	부산

● **관계의 분류**

- 관계는 존재에 의한 관계와 행위에 의한 관계로 분류
- **존재 관계는 엔터티 간의 상태를 의미**
  ex) 사원 엔터티는 부서 엔터티에 소속

- **행위 관계는 엔터티 간의 어떤 행위가 있는 것을 의미**
  ex) 주문은 고객이 주문할 때 발생

● **조인의 의미**

- 결국 데이터의 중복을 피하기 위해 테이블은 정규화에 의해 분리된다. 분리되면서 두 테이블은 서로 관계를 맺게 되고, 다시 이 두 테이블의 데이터를 동시에 출력하거나 관계가 있는 테이블을 참조하기 위해서는 데이터를 연결해야 하는데 이 과정을 조인이라고 함

계좌

계좌번호	예수금	관리점코드	관리점
100111	100	1000	서울점
100222	200	1001	경기점
100333	300	1002	인천점
100444	400	1003	제주점

계좌

계좌번호	예수금	관리점코드
100111	100	1000
100222	200	1001
100333	300	1002
100444	400	1003

관리점

관리점코드	관리점
1000	서울점
1001	경기점
1002	인천점
1003	제주점

☞ **계좌 테이블은 제 3 정규화에 의해 계좌 + 관리점으로 분리됨. 이 때 관리점 코드를 같이 공유함. 만약 계좌 정보와 함께 관리점 정보를 함께 출력(예. 관리점 별로 거래 계좌의 수 확인)할 경우 두 데이터를 조인하여 출력함. 즉, 양 테이블의 데이터를 함께 출력하거나 참조하기 위해 두 데이터를 연결하는 과정을 조인, 연결키를 조인키라고 함**

조인 예) 계좌번호 100111 의 관리점이 어딘지를 찾으려면?

계좌

계좌번호	예수금	관리점코드	관리점
100111	100	1000	서울점
100222	200	1001	경기점
100333	300	1002	인천점
100444	400	1003	제주점

계좌

계좌번호	예수금	관리점코드
100111	100	1000
100222	200	1001
100333	300	1002
100444	400	1003

관리점

관리점코드	관리점
1000	서울점
1001	경기점
1002	인천점
1003	제주점

① 계좌번호 테이블에서 계좌번호가 100111 데이터 확인
② 계좌번호 테이블에서 계좌번호가 100111 데이터의 관리점코드(1000)를 확인
③ 관리점코드(1000)를 관리점 테이블에 전달하여 관리점 확인(서울점)

SQL 작성)

```
SELECT A.계좌번호, B.관리점
FROM 계좌 A, 관리점 B
WHERE A.관리점코드 = B.관리점코드
AND A.계좌번호 = '100111'
```

● **계층형 데이터 모델**

- **자기 자신끼리 관계가 발생**. 즉, 하나의 엔터티 내의 인스턴스끼리 계층 구조를 가지는 경우를 말함
- 계층 구조를 갖는 인스턴스끼리 연결하는 조인을 셀프조인이라함(같은 테이블을 여러 번 조인)

아래 EMP 테이블은 직원테이블로 순서대로 사번, 이름, 직무, 매니저번호, 입사일, 급여, 보너스, 부서번호 컬럼으로 구성. 이 때, 매니저번호(MGR)는 매니저의 사원번호를 의미하므로 사원번호(EMPNO) 컬럼과 관련이 있다.
즉, SMITH 의 매니저 이름을 확인하는 과정을 보면, SMITH 의 매니저번호를 확인(7902),
해당 번호의 사번을 갖는 데이터를 찾으면 FORD 라는것을 확인할 수 있음

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	1980/12/17 00:00:00	800		20
2	7499	ALLEN	SALESMAN	7698	1981/02/20 00:00:00	1600	300	30
3	7521	WARD	SALESMAN	7698	1981/02/22 00:00:00	1250	500	30
4	7566	JONES	MANAGER	7839	1981/04/02 00:00:00	2975		20
5	7654	MARTIN	SALESMAN	7698	1981/09/28 00:00:00	1250	1400	30
6	7698	BLAKE	MANAGER	7839	1981/05/01 00:00:00	2850		30
7	7782	CLARK	MANAGER	7839	1981/06/09 00:00:00	2450		10
8	7788	SCOTT	ANALYST	7566	1987/04/19 00:00:00	3000		20
9	7839	KING	PRESIDENT		1981/11/17 00:00:00	5000		10
10	7844	TURNER	SALESMAN	7698	1981/09/08 00:00:00	1500	0	30
11	7876	ADAMS	CLERK	7788	1987/05/23 00:00:00	1100		20
12	7900	JAMES	CLERK	7698	1981/12/03 00:00:00	950		30
13	7902	FORD	ANALYST	7566	1981/12/03 00:00:00	3000		20
14	7934	MILLER	CLERK	7782	1982/01/23 00:00:00	1300		10

위 예제를 SQL 로 표현하면,

SQL1 *

```
1  SELECT E1.ENAME AS 사원이름,
2         E2.ENAME AS 매니저이름
3    FROM EMP E1, EMP E2
4   WHERE E1.MGR = E2.EMPNO;
```

Result

Grid Result | Server Output | Text Output

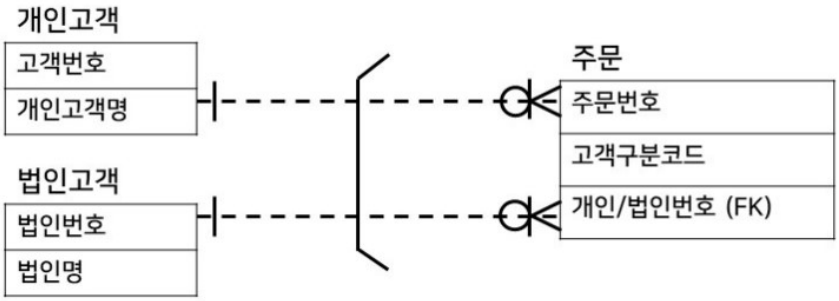
	사원이름	매니저이름
1	FORD	JONES
2	SCOTT	JONES
3	TURNER	BLAKE
4	ALLEN	BLAKE
5	WARD	BLAKE
6	JAMES	BLAKE
7	MARTIN	BLAKE
8	MILLER	CLARK
9	ADAMS	SCOTT
10	BLAKE	KING
11	JONES	KING
12	CLARK	KING
13	SMITH	FORD

● **상호배타적 관계**

- 두 테이블 중 하나만 가능한 관계를 말함

ex) 주문 엔터티에는 개인 또는 법인번호 둘 중 하나만 상속될 수 있음 => 상호배타적 관계
즉, 주문은 개인고객이거나 법인고객 둘 중 하나의 고객만이 가능

IE 표기법



출처 : SQL 전문가 가이드

1-8. 모델이 표현하는 트랜잭션의 이해

● **트랜잭션이란**

- **하나의 연속적인 업무 단위를 말함**
- 트랜잭션에 의한 관계는 **필수적인 관계 형태**를 가짐
- 하나의 트랜잭션에는 여러 SELECT, INSERT, DELETE, UPDATE 등이 포함될 수 있음

※ **계좌이체**를 예를 들면)

A 고객이 B 고객에게 100 만원을 이체하려고 한다고 가정하자.

STEP1) A 고객의 잔액이 100 만원 이상인지 확인
STEP2) 이상이면, A 고객 잔액을 -100 UPDATE
STEP3) B 고객 잔액에 +100 UPDATE

이 때, 2 번과 3 번 과정이 동시에 수행되어야 한다. 즉 **모두 성공하거나 모두 취소돼야 함(All or Nothing)**

☞ **이런 특성을 갖는 연속적인 업무 단위를 트랜잭션이라고 한다.**

※ 주의

1. A 고객 잔액 차감과 B 고객 잔액 가산이 서로 독립적으로 발생하면 안됨
   -> 각각의 INSERT 문으로 개발되면 안됨
2. 부분 COMMIT 불가
   -> 동시 COMMIT 또는 ROLLBACK 처리

● **필수적, 선택적 관계와 ERD**

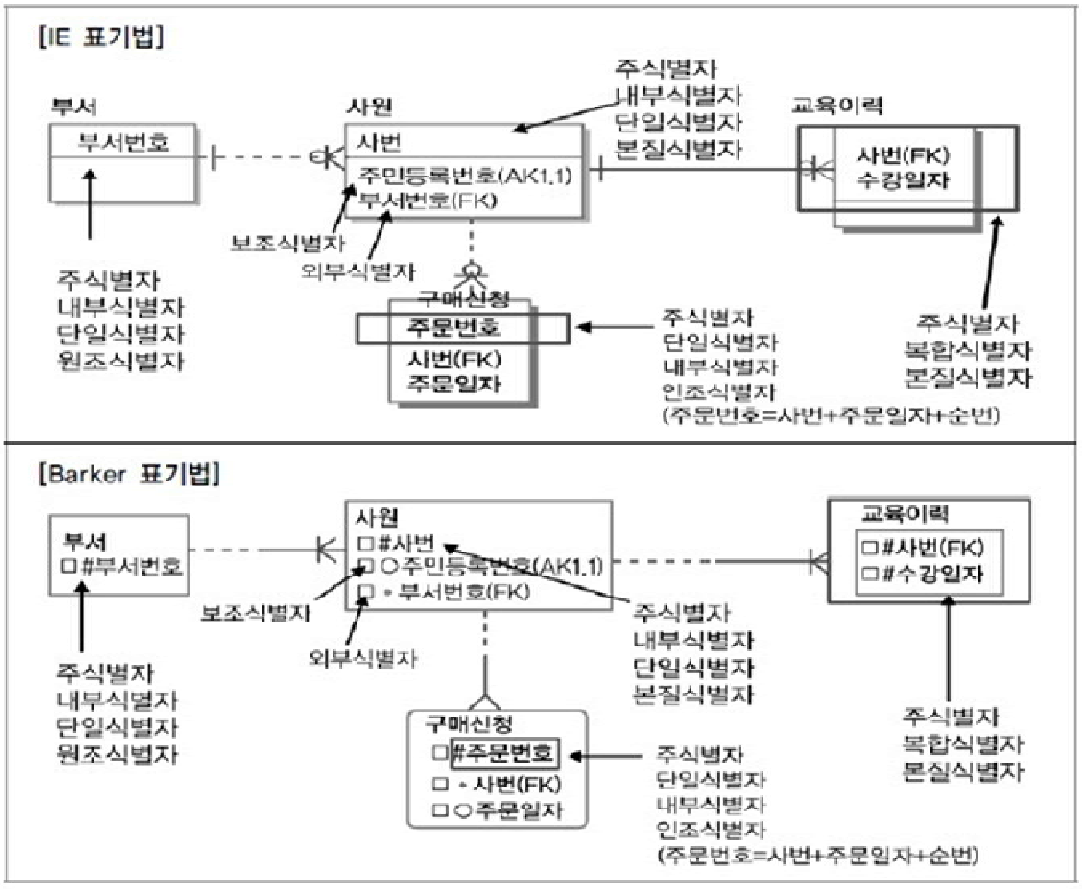
- 두 엔터티의 관계가 서로 필수적일 때 하나의 트랜잭션을 형성
- 두 엔터티가 서로 독립적 수행이 가능하다면 선택적 관계로 표현

IE 표기법)

- **원을 사용하여** 필수적 관계와 선택적 관계를 **구분**
- **필수적 관계**에는 **원을 그리지 않는다.**
- **선택적 관계에는** 관계선 끝에 **원을 그린다.**

바커표기법)

- **실선과 점선으로 구분**
- **필수적 관계**는 관계선을 **실선으로** 표기
- **선택적 관계**는 관계선을 **점선**으로 표기



[그림 Ⅰ-1-42] 식별자의 분류-데이터 모델

1.9 null 속성의 이해

● **NULL 이란**

- DBMS 에서 **아직 정해지지 않은 값을 의미**
- 0 과 빈문자열('')과는 다른 개념
- 모델 설계 시 각 컬럼별로 NULL 을 허용할 지를 결정(Nullable Column)

● **NULL 의 특성**

1. NULL 을 포함한 연산 결과는 항상 NULL

SQL1 *

```
1  SELECT ENAME, SAL, COMM, SAL + COMM
2    FROM EMP;
```

	ENAME	SAL	COMM	SAL+COMM
1	SMITH	800		
2	ALLEN	1600	300	1900
3	WARD	1250	500	1750
4	JONES	2975		
5	MARTIN	1250	1400	2650
6	BLAKE	2850		
7	CLARK	2450		
8	SCOTT	3000		
9	KING	5000		
10	TURNER	1500	0	1500
11	ADAMS	1100		
12	JAMES	950		
13	FORD	3000		
14	MILLER	1300		

☞ COMM 컬럼에 공백으로 보이는것들이 NULL 이다(물론 빈문자열일 수 있지만 해당 데이터에서는 NULL 임)
이 때, NULL 을 포함한 COMM 과 SAL 과의 연산결과는 NULL 이 리턴된다.
-> NULL 을 사전에 치환한 후 연산 필

※ NULL 치환 후 연산 결과

```
SQL1 *
1  SELECT ENAME, SAL, COMM, SAL + NVL(COMM, 0)
2    FROM EMP;
```

	ENAME	SAL	COMM	SAL+NVL(COMM,0)
1	SMITH	800		800
2	ALLEN	1600	300	1900
3	WARD	1250	500	1750
4	JONES	2975		2975
5	MARTIN	1250	1400	2650
6	BLAKE	2850		2850

2. 집계함수는 NULL 을 제외한 연산 결과 리턴

※ sum, avg, min, max 등의 함수는 항상 NULL 을 무시한다.

예) NULL 을 포함한 컬럼의 집계함수 결과 1

```
SQL1 *
1  SELECT COUNT(*), COUNT(SAL), COUNT(COMM)
2    FROM EMP;
3
```

Result

Grid Result | Server Output | Text Output | Explain Pl

	COUNT(*)	COUNT(SAL)	COUNT(COMM)
1	14	14	4

☞ COUNT 는 행의 수를 리턴하는 함수인데, COUNT 에 * 전달 시 모든 컬럼을 체크하여 NULL 일 경우는 COUNT 제외. COMM 의 경우 NULL 이 다수 포함되어 있는데, COUNT 시 NULL 을 제외. 즉 NOT NULL 인 행만 세어 리턴하므로 전체 행의 수보다 적은 4 의 값이 출력됨

예) NULL 을 포함한 컬럼의 집계함수 결과 2

```
SQL1 *
1  SELECT SUM(COMM), MIN(COMM), MAX(COMM)
2    FROM EMP;
3
```

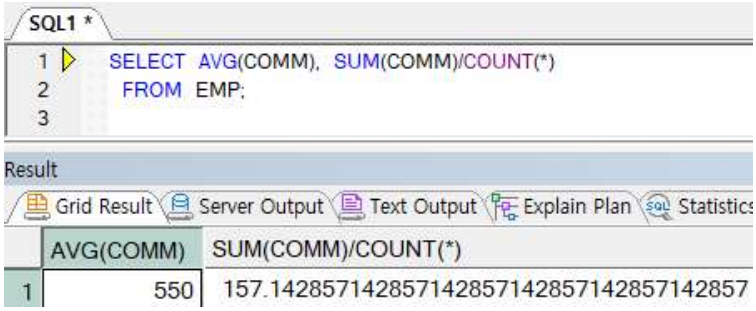
Result

Grid Result | Server Output | Text Output | Explain Plar

	SUM(COMM)	MIN(COMM)	MAX(COMM)
1	2200	0	1400

☞ SUM, MIN, MAX 연산 결과도 모두 NULL 을 무시하여 연산한다.

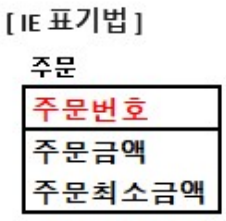
예) NULL 을 포함한 컬럼 평균연산

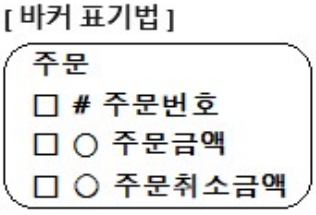


☞ AVG 연산 결과는 NULL 을 무시한 평균을 리턴하므로, NULL 아닌 4 개의 데이터들의 평균을 리턴. 두 번째 수식은 평균을 직접 구한 것으로, COMM 의 총 합을 총 행이 수인 14 로 나눈 값이다. 따라서 이 두 연산결과는 COMM 이 NULL 을 포함할 경우 항상 다르게 리턴된다. NULL 을 무시한 평균을 얻고자 함인지, 전체 14 명에 대한 평균을 계산하고자 함인지에 따라 적절히 선택하여 사용!

● **NULL 의 ERD 표기법**

- IE 표기법에서는 NULL 허용여부를 알 수 없음
- 바커 표기법에서는 속성 앞에 동그라미가 NULL 허용 속성을 의미함





1-10. 본질식별자

● **식별자 구분(대체 여부에 따른)**

1) **본질식별자**

- 업무에 의해 만들어지는 식별자(**꼭 필요한 식별자**)

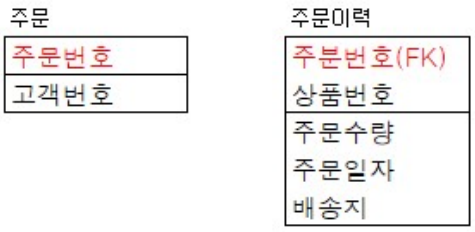
2) **인조식별자**

- 인위적으로 만들어지는 식별자(**꼭 필요하지 않지만 관리의 편이성 등의 이유로 인위적으로 만들어지는 식별자**)
- 본질식별자가 복잡한 구성을 가질때 인위적으로 생성
- 주로 각 행을 구분하기 위한 기본키로 사용되며 자동으로 증가하는 일련번호 같은 형태임

예제) 주문과 주문상세에 대한 엔터티 설계 과정을 예를 들어보자.

주문이 들어오면 주문 엔터티에는 (주문번호 + 고객번호)를 저장, 이 때 PK 는 주문번호이다.

주문상세에는 각 주문별로 어떤 상품이, 언제, 몇 개 주문됐는지 등을 기록한다.



※ 주문상세 테이블 설계 시 다음과 같은 식별자를 고려할 수 있다.

1. PK : 주문번호 + 상품번호로 설계
   - 주문을 하면 주문번호와 상품번호가 필요하므로 **본질식별자(주문번호 + 상품번호)**가 된다
   - 하지만 PK 가 주문번호 + 상품번호이면 **하나의 주문번호로 같은 상품의 주문 결과를 저장할 수 없게 된다**.

☞ 실제로 쇼핑을 하다보면 동일한 장바구니에 A 상품을 5 개 주문했는데, 뒤에 또 다시 A 상품을 3 개 추가로 주문하기도 함

2. PK : 주문번호 + 주문순번(주문순번이라는 컬럼을 생성)
   - 하나의 주문에 여러 상품에 대한 주문 결과 저장 가능 -> 주문순번으로 인해 구분함

주문이력

주문번호(FK)
주문순번
상품번호
상품명
주문일자
배송지

주문번호	주문순번	상품번호	상품명	주문일자	배송지
0000001	1	100	사과	2024-01-01	서울시
0000001	2	100	사과	2024-01-01	경기도
0000001	3	100	사과	2024-01-01	제주도

☞ 매 주문마다 동일한 상품 주문 시 주문순번을 정하기 위해 상품의 주문 횟수를 세야한다는 점이 매우 불편!
즉, 사과를 총 3번 구매 하였으니 주문순번은 1,2,3 순서대로 입력돼야 함

3. PK : 주문상세번호(인조식별자 생성)
   - 주문상세번호로 각 주문이력을 구분하기 때문에 같은 주문의 같은 상품이력이 저장될 수 있음
   - 주문상세번호만이 주식별자이므로 나머지 정보들이 불필요하게 중복 저장될 위험 발생
   - 실제 업무와 상관없는 주문상세번호를 주식별자로 생성하면 쓸모없는 index가 생성됨(PK 생성 시 자동 unique index 생성)

주문이력

주문상세번호
주분번호(FK)
상품번호
상품명
주문일자
배송지

주문상세번호	주문번호	상품번호	상품명	주문일자	배송지
1	0000001	100	사과	2024-01-01	서울시
2	0000001	100	사과	2024-01-01	서울시
3	0000001	100	사과	2024-01-01	제주도

※ 따라서 **인조식별자는 다음의 단점을 가지게 된다.**
1. **중복 데이터 발생** 가능성 -> 데이터 품질 저하
2. **불필요한 인덱스** 생성 -> 저장공간 낭비 및 DML 성능 저하

** 인덱스는 원래 조회 성능을 향상시키기 위한 객체이며, 인덱스는 DML(INSERT/UPDATE/DELETE)시 INDEX SPLIT 현상으로 인해 성능이 저하된다.