

# 데이터 입출력 구현

# 데이터 전환

## ❖ 데이터 전환

- ✓ 운영 중인 기존 정보 시스템에 축적되어 있는 데이터를 추출(Extraction)하여 새로 개발할 정보 시스템 에서 운영 가능하도록 변환(Transformation)한 후 적재>Loading)하는 일련의 과정
- ✓ 데이터 이행(Data Migration) 또는 데이터 이관 이라고도 함
- ✓ 데이터 전환 계획서: 데이터 전환이 필요한 대상을 분석하여 데이터 전환 작업에 필요한 모든 계획을 기록하는 문서



# 데이터 검증

## ❖ 데이터 검증

✓ 원천 시스템의 데이터를 목적 시스템의 데이터로 전환하는 과정이 정상적으로 수행되었는지 여부를 확인하는 과정

### ● 검증 방법

- 로그 검증: 데이터 전환 과정에서 작성하는 추출, 전환, 적재 로그를 검증
- 기본 항목 검증: 로그 검증 외에 별도로 요청된 검증 항목에 대해 검증
- 응용 프로그램 검증: 응용 프로그램을 통한 데이터 전환의 정합성을 검증
- 응용 데이터 검증: 사전에 정의된 업무 규칙을 기준으로 데이터 전환의 정합성을 검증
- 값 검증: 숫자 항목의 합계 검증, 코드 데이터의 범위 검증, 속성 변경에 따른 값 검증을 수행

# 데이터 검증

## ❖ 데이터 검증

### ✓ 검증 단계

- 추출: 원천 시스템 데이터에 대한 정합성 확인
- 전환
  - 매핑 정의서에 정의된 내용이 정확히 반영되었는지 확인
  - 매핑 정의서 오류 여부 확인
- DB 적재: SAM 파일을 적재하는 과정에서 발생할 수 있는 오류나 데이터 누락 여부 등 확인
- DB 적재 후: 적재 완료 후 정합성 확인
- 전환 완료 후: 데이터 전환 완료 후 추가 검증 과정을 통해 데이터 전환의 정합성 검증

# 오류 데이터 측정 및 정제

## ❖ 오류 데이터 측정 및 정제

- ✓ 고품질의 데이터를 운영 및 관리하기 위해서 수행
- ✓ 진행 과정
  - 데이터 품질 분석: 오류 데이터를 찾기 위해서 데이터의 적합성을 확인하는 작업
  - 오류 데이터 측정: 정상적인 데이터 와 오류 데이터를 측정해서 관리 목록을 작성하는 작업
  - 오류 데이터 정제: 오류 데이터를 수정하거나 데이터 전환 프로그램 자체를 수정
- ✓ 오류 상태
  - Open: 오류가 보고만 되고 분석되지 않은 상태
  - Assigned: 오류의 영향 분석 및 수정을 위해서 개발자에서 오류를 전달할 상태
  - Fixed: 개발자가 오류를 수정한 상태
  - Closed: 수정된 데이터를 가지고 테스트를 했을 때 오류가 발견되지 않은 상태
  - Deferred: 오류 수정을 연기한 상태
  - Classified: 보고된 오류를 관련자들이 확인했을 때 오류가 아니라고 확인된 상태
- ✓ 데이터 정제 요청서: 데이터 정제와 관련된 전반적인 내용을 문서화한 것
- ✓ 데이터 정제 보고서: 정제된 데이터가 정상적으로 정제되었는지 확인한 결과를 문서화한 것

# DBMS

## ❖ 데이터베이스

- ✓ 데이터베이스는 연관된 데이터의 모임이라 이해할 수 있으며 일반적인 파일 시스템보다 엄격하게 데이터를 일정한 형태로 저장해 놓은 것을 의미
- ✓ 정의
  - 저장 데이터(Stored Data): 컴퓨터를 통해 접근 가능한 저장 매체에 저장된 데이터
  - 통합 데이터(Integrated Data): 중복이 최소화된 데이터
  - 공유 데이터(Shared Data): 여러 응용 프로그램이 공동으로 사용하는 데이터
  - 운영 데이터(Operational Data): 조직의 목적을 위해 존재 가치가 확실하고 반드시 필요한 데이터
- ✓ 과거에는 파일 시스템을 이용하여 데이터를 관리하였는데 파일 시스템을 통하여 데이터의 공유는 가능하였지만 동시에 데이터의 입력, 수정 및 삭제와 같은 조작이 어렵고 동일한 내용이 복사되어 사용되거나 관리해야하는 문제를 해결하기가 어려워서 이와 같은 문제를 해결하기 위하여 파일 시스템을 개선한 데이터 관점의 데이터베이스관리시스템(DBMS)이 탄생

## ❖ DBMS(DataBase Management System): 사용자의 요구에 따라 정보를 생성해주고 데이터베이스를 관리해주는 소프트웨어

- ✓ DBMS의 3대 기능
  - 정의(Definition): 데이터베이스 개체를 생성하고 구조를 변경하고 삭제하는 기능
  - 조작(Manipulation): 데이터를 검색하고 삽입, 수정, 삭제하는 기능
  - 제어(Control): 데이터의 무결성, 보안, 회복, 병행 제어 관련 기능

# DBMS

## ❖ 데이터의 독립성

- ✓ 논리적 독립성: 응용 프로그램과 데이터베이스를 논리적으로 분리해서 구현함으로써 데이터의 논리적 구조를 변경하더라도 응용 프로그램은 영향을 받지 않아야 한다는 성질
- ✓ 물리적 독립성: 응용 프로그램과 실제 데이터 저장소를 분리시킴으로써 데이터의 물리적 구조를 변경하더라도 응용 프로그램은 영향을 받지 않아야 한다는 성질

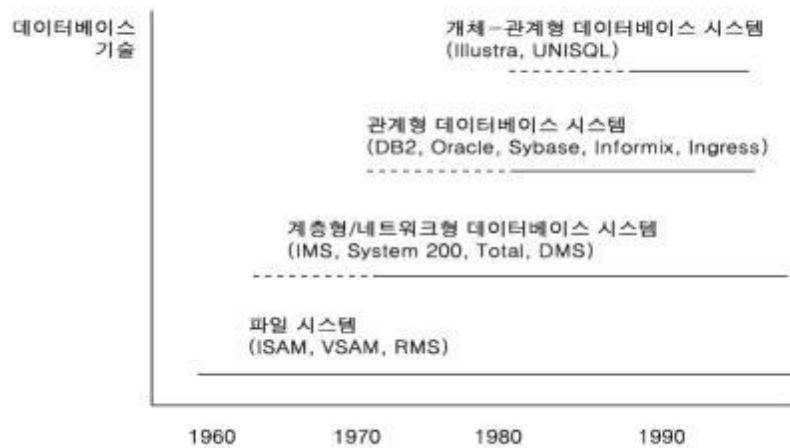
## ❖ 데이터 독립성의 구현 방법(3단계 데이터베이스 구성)

- ✓ 스키마(schema)란 데이터베이스의 구조(개체, 속성, 관계)에 대한 정의와 이에 대한 제약 조건 등을 기술한 것으로 컴파일 되어 데이터 사전에 저장
- ✓ 외부 스키마(external schema)
  - 사용자나 응용 프로그래머가 접근할 수 있는 데이터베이스를 정의 한다.
  - 전체 데이터베이스의 한 논리적인 부분으로 볼 수 있으므로 서브 스키마(subschema)라고도 함
- ✓ 개념 스키마(conceptual schema)
  - 범 기관 적 입장에서 본 데이터베이스의 정의를 기술한 것
  - 모든 응용에 대한 전체적으로 통합된 데이터 구조
  - 접근 권한, 보안 정책, 무결성 규칙을 명세
- ✓ 내부 스키마(internal schema)
  - 물리적 저장 장치의 관점에서 본 데이터베이스의 명세
  - 개념 스키마의 물리적 저장 구조에 대한 정의를 기술한 것

# DBMS

## ❖ DBMS의 역사

1960년대	Flat-File(SAM)
1970년대	Network-DB, Hierarchical-DB
1980년대	관계형-DBMS
1990년대	관계형-DBMS, 객체관계형-DBMS
2000년대	관계형, 객체관계형, 객체지향-DBMS, NoSQL





# DBMS

## ❖ DBMS 종류

- ✓ Hierarchical-DB(계층형 데이터베이스): 디렉토리나 파일 등의 계층 구조로 저장하는 방식의 데이터베이스
- ✓ Network-DB: 계층형 데이터베이스의 데이터 중복 문제를 해결하기 위해 레코드 간의 다양한 관계를 그물처럼 갖는 구조의 데이터베이스
- ✓ Relational-DB(관계형 데이터베이스): 테이블 구조의 데이터베이스
- ✓ 객체 지향 관계형 데이터베이스: 객체 그대로를 데이터베이스의 데이터로 저장하는 것을 목적으로 하는 데이터베이스
- ✓ XML 데이터베이스: 태그 형태로 데이터를 저장하는 방식으로 SQL 대신에 XQuery 라는 전용 명령어를 사용
- ✓ No SQL
  - Key-Value Store: 키와 값의 형식으로 단순하게 데이터를 저장하는 방식의 데이터베이스로 Firebase Realtime Database, Redis 등이 있음
  - Document: 하나의 데이터를 하나의 문서로 취급하는 데이터베이스로 Mongo DB 가 대표적
  - Big Table(Column): Key-Value 형식을 확장한 개념으로 Column Family 라는 개념을 도입한 데이터베이스로 Cassandra, HBase 가 대표적

# 데이터베이스 설계

- ❖ 사용자의 요구 사항을 분석하여 컴퓨터에 저장할 수 있는 구조로 변경한 후 데이터베이스를 구현하여 사용자들이 사용할 수 있도록 하는 것
- ❖ 데이터베이스 설계 시 고려 사항
  - ✓ 무결성(Integrity): 삽입, 삭제, 갱신 등의 연산 후에도 데이터베이스에 저장된 데이터가 정해진 제약 조건을 만족
  - ✓ 일관성(Consistency): 데이터베이스에 저장된 데이터들 사이나, 특정 질의에 대한 응답이 처음부터 끝까지 변함없이 일정
  - ✓ 회복(Recovery): 시스템에 장애가 발생했을 때 장애 발생 직전의 상태로 복구 가능
  - ✓ 보안(Security): 불법적인 데이터의 노출 또는 변경이나 손실로부터 보호
  - ✓ 효율성(Efficiency): 응답 시간의 단축, 시스템의 생산성, 저장 공간의 최적화 등이 가능
  - ✓ 데이터베이스 확장(Extension): 데이터베이스 운영에 영향을 주지 않으면서 지속적으로 데이터를 추가 가능



# 데이터베이스 설계

## ❖ 데이터베이스 설계 단계

- ✓ 요구조건 분석 단계: 사용자로부터 데이터의 용도를 파악하는 작업
- ✓ 개념적 설계 단계: 현실 세계의 데이터를 추상적인 개념의 스키마로 변환하는 작업으로 개념 스키마 모델링(E-R Diagram 작성)과 트랜잭션 모델링을 수행
- ✓ 논리적 설계 단계: 특정 DBMS의 논리적 구조로 변환하는 과정으로 이 과정에서 트랜잭션의 인터페이스를 설계
- ✓ 물리적 설계 단계: 논리적 구조를 물리적 저장장치의 구조로 변환하는 과정 (반응시간, 공간 활용도, 트랜잭션 처리량 등을 고려)
- ✓ 구현 단계



# 데이터 모델

## ❖ 데이터 모델

- ✓ 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형
- ✓ 데이터 모델의 구성 요소
  - 개체(Entity): 데이터베이스에 표현하려는 것으로 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상
  - 속성(Attribute): 데이터의 가장 작은 논리적 단위로서 파일 구조상의 데이터 항목 또는 데이터 필드
  - 관계(Relationship): 개체 간의 관계 또는 속성 간의 논리적인 연결
- ✓ 데이터 모델의 종류
  - 개념적 데이터 모델: 현실 세계에 대한 인간의 이해를 돕기 위해 현실세계에 대한 인식을 추상적 개념으로 표현하는 과정으로 E-R Diagram을 작성하는 것이 개념적 데이터 모델링
  - 논리적 데이터 모델: 개념적 모델링 과정에서 얻은 개념적 구조를 컴퓨터가 이해하고 처리할 수 있는 컴퓨터 세계의 환경에 맞도록 변환하는 과정
  - 물리적 데이터 모델: 실제 컴퓨터에 데이터가 저장되는 방법을 정의하는 물리 데이터베이스 설계 과정

# 데이터 모델

## ❖ 모델링의 이해

### ✓ 모델링의 3가지 관점

#### □ 데이터 관점

- 업무가 어떤 데이터와 관련이 있는지?
- 데이터간의 관계는 무엇인지?
- 비즈니스 프로세스에서 사용하는 데이터 - Data
- What
- 정적 분석과 구조 분석

#### □ 프로세스관점

- 업무(비즈니스 프로세스에서 수행하는 작업)가 실제하고 있는 일이 무엇인지?
- 무엇을 모델링해야 하는지?
- How
- 시나리오 분석, 도메인 분석, 동적 분석

#### □ 데이터와 프로세스의 상관 관점

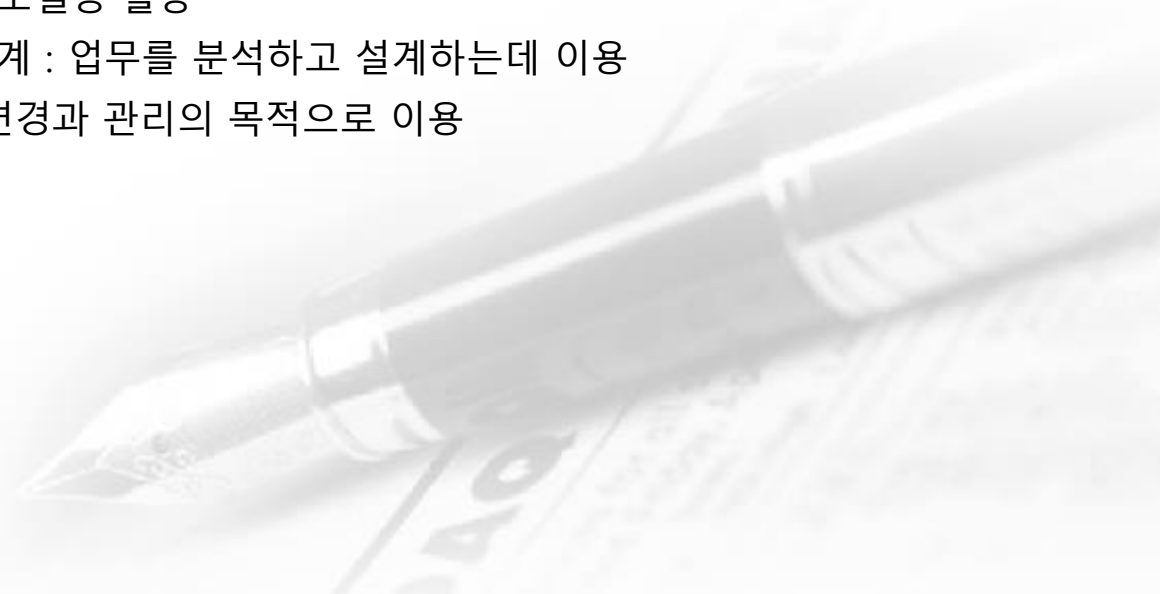
- 업무가 처리하는 일의 방법에 따라 데이터는 어떻게 영향을 받고 있는지?
- Interaction(상호작용)
- CRUD

# 데이터 모델

## ❖ 모델링의 이해

### ✓ 모델링의 특징

- ❑ 추상화(모형화, 가설적): 현실세계를 일정한 형식에 맞추어 표현을 한다는 의미로 다양한 현상을 일정한 양식인 표기법에 의해 표기한다는 것
- ❑ 단순화: 복잡한 현실세계를 약속된 규약에 의해 제한된 표기법이나 언어로 표현하여 쉽게 이해할 수 있도록 하는 개념
- ❑ 명확화: 누구나 이해하기 쉽게 하기 위해 대상에 대한 애매모호함을 제거하고 정확하게 현상을 기술하는 것
- ❑ 모델링의 재정의 : 현실세계를 추상화, 단순화, 명확화하기 위해 일정한 표기법에 의해 표현하는 기법
- ❑ 정보시스템 구축에서의 모델링 활용
  - 계획/분석/설계 단계 : 업무를 분석하고 설계하는데 이용
  - 구축/운영 단계 : 변경과 관리의 목적으로 이용



# 데이터 모델

## ❖ 데이터 모델에 표시할 요소

- ✓ 구조(Structure): 논리적으로 표현된 개체 타입들 간의 관계로서 데이터 구조 및 정적 성질을 표현함
- ✓ 연산(Operation): 데이터베이스에 저장된 실제 데이터를 처리하는 작업에 대한 명세로서 데이터베이스를 조작하는 기본 도구
- ✓ 제약조건(Constraint): 데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약 조건

## ❖ 데이터 사전(data dictionary)과 데이터 디렉토리(data directory)

- ✓ 데이터 사전은 DBA의 도구로서 시스템 내의 모든 개체들에 대한 정의나 명세에 관한 정보(데이터를 위한 데이터, Meta -Data)를 수록하며 시스템 카탈로그(catalog)라고도 함
- ✓ 데이터 디렉토리는 데이터 사전에 수록된 데이터를 참조하는데 필요한 정보를 수록
- ✓ 데이터 사전은 사용자와 시스템이 모두 사용할 수 있는 반면 데이터 디렉토리는 시스템에서만 사용
- ✓ 메타 데이터는 시스템이 사용하고 사용자는 조회만 가능



# 데이터 모델

## ❖ 개체(Entity)

- ✓ 데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체이며 실 세계에 독립적으로 존재하는 유형, 무형의 정보
- ✓ 변별할 수 있는 사물 - Peter Chen (1976)
- ✓ 데이터베이스 내에서 변별 가능한 객체 - C.J Date (1986)
- ✓ 정보를 저장할 수 있는 어떤 것 - James Martin (1989)
- ✓ 정보가 저장될 수 있는 사람, 장소, 물건, 사건 그리고 개념 등 - Thomas Bruce (1992)
- ✓ 엔티티는 데이터의 집합
- ✓ 엔티티는 사람, 장소, 물건, 사건, 개념 등의 명사에 해당
- ✓ 엔티티는 업무 상 관리가 필요한 관심사에 해당
- ✓ 엔티티는 저장이 되기 위한 어떤 것
- ✓ 구성 요소
  - 속성(Attribute): 개체가 가지고 있는 특성
  - 개체 타입(Entity Type): 속성으로만 기술된 개체의 정의
  - 개체 인스턴스(Entity Instance): 개체를 구성하고 있는 각 속성들이 값을 가져 하나의 개체를 나타내는 것으로 개체 어커런스(Occurrence)라고도 함
  - 개체 세트(Entity Set): 개체 인스턴스의 집합



# 데이터 모델

## ❖ Entity 특징

- ✓ 업무에서 필요로 하는 정보: 반드시 해당 업무에서 필요하고, 관리하고자 하는 정보

병원시스템	Entity	인사시스템
O	환자	X
X	토익점수	O

- ✓ 식별이 가능해야 함: 유일한 식별자에 의해 식별이 가능해야 함
- ✓ Instance 의 집합
  - ❑ 연속적으로 존재하는 Instance 의 집합
  - ❑ 2개 이상의 Instance 의 집합
  - ❑ 1개의 Instance 로 이루어진 집합은 Entity 가 아니다.(?)
- ✓ 업무 프로세스에 의해 이용: 업무 프로세스가 반드시 그 Entity를 이용
- ✓ 속성을 포함
  - ❑ Entity 에는 반드시 속성(Attributes)이 포함되어야 함
  - ❑ 식별자만 존재하고 일반 속성이 전혀 없는 객체는 Entity 가 될 수 없음
  - ❑ 단 관계 Entity 의 경우엔 주 식별자 속성만으로도 Entity 로 인정

# 데이터 모델

## ❖ Entity 특징

### ✓ 관계의 존재

- ❑ Entity 는 다른 Entity 와 최소 한 개 이상의 관계가 존재하여야 함
- ❑ 데이터 모델링에서 관계를 생략하여 표현하는 경우
  - 시스템 처리시 내부적으로 필요한 Entity : 로그 테이블
  - 통계를 위한 데이터 : 통계 만을 위한 Read Only Table
  - 코드성 Entity
    - 너무 많은 Entity 들과의 관계로 데이터 모델이 복잡해 짐
    - 일반적으로 코드 테이블에 FK 를 설정하지 않는 경우가 대부분



# 데이터 모델

## ❖ Entity 의 분류

### ✓ 유무(有無)형에 따른 분류

#### □ 유형 엔티티(Tangible Entity)

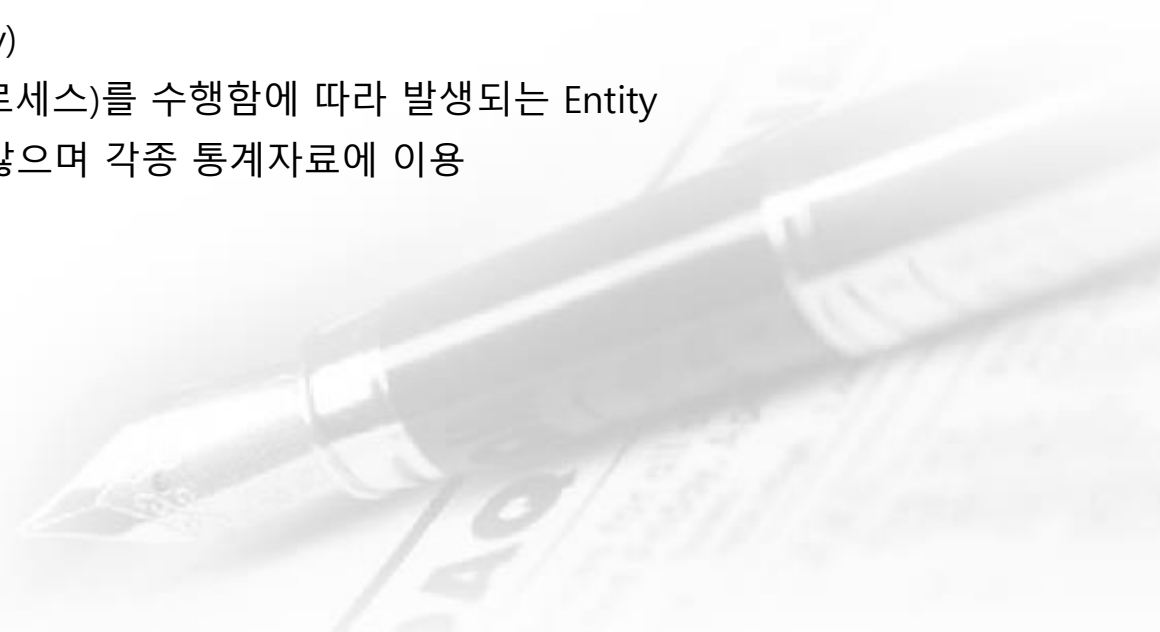
- 물리적 형태가 있고 안정적이며 지속적으로 활용되는 Entity
- 업무에서 도출되며 Entity를 구분하기가 가장 용이한 엔티티
- 사원, 물품, 강사

#### □ 개념 엔티티(Conceptual Entity)

- 물리적 형태는 존재하지 않고 관리해야 할 개념적 정보로 구분이 되는 Entity
- 조직, 보험상품

#### □ 사건 엔티티(Event Entity)

- 업무(비즈니스 프로세스)를 수행함에 따라 발생하는 Entity
- 비교적 발생량이 많으며 각종 통계자료에 이용
- 주문, 청구, 미납



# 데이터 모델

## ❖ Entity 의 분류

### ✓ 발생 시점(發生時點)에 따른 분류

#### □ 기본 엔티티(Basic Entity, Fundamental Entity, Key Entity)

- 업무에 원래 존재하는 정보로서 다른 Entity와 관계에 의해 생성되지 않고 독립적으로 생성 가능
- 다른 Entity로부터 주 식별자를 상속받지 않고 자신의 고유 식별자를 가짐
- 사원, 부서, 고객, 상품, 자재

#### □ 중심 엔티티(Main Entity, Transaction Entity)

- 기본 Entity로부터 발생되고, 그 업무에 있어서 중요한 역할
- 데이터 양이 많이 발생되고 다른 Entity와의 관계를 통해 행위 Entity를 생성
- 계약, 사고, 청구, 주문, 매출

#### □ 행위 엔티티(Active Entity)

- 두개 이상의 Entity로부터 발생되고 자주 내용이 바뀌거나 데이터 양이 증가
- 주문 목록, 사원 변경 이력

#### □ 종속 엔티티(Dependent Entity): 정규화의 결과로 만들어지는 Entity

#### □ 교차 엔티티(Cross Entity): 다 대 다 관계 해소 목적의 Entity

# 데이터 모델

## ❖ 속성(Attribute)

- ✓ 속성은 데이터베이스를 구성하는 가장 작은 논리적 단위로 파일 구조상의 데이터 항목 또는 필드에 해당하며 개체의 특성을 기술
- ✓ 도메인: 속성이 가질 수 있는 값의 범위
  - ❑ 학점은 0.0~4.0 사이의 실수
  - ❑ 주소는 20자리 문자열
- ✓ Degree: 속성의 개수
- ✓ 속성의 분해 여부에 따른 분류
  - ❑ 단일 속성: 하나의 의미로 구성된 것으로 대표적으로 아이디나 이름
  - ❑ 복합 속성: 여러 개의 의미가 있는 것으로 대표적으로 주소 속성이 있음
  - ❑ 다중 값 속성: 여러 개의 값을 가질 수 있는 것으로 리스트 형태들인데 이 속성은 다른 엔티티로 독립 시켜야 함
- ❑ 속성의 개체 구성 방식에 따른 분류
  - ❑ 기본키: Entity 내에서 Row를 구분하기 위한 속성
  - ❑ 외래키: 다른 Entity의 데이터를 조회하기 위한 속성
  - ❑ 일반 속성: Entity 내에 속해있지만 기본키나 외래키에 속하지 않는 속성

# 데이터 모델

## ❖ 속성(Attribute)

### ✓ 속성의 특성에 따른 구분

#### □ 기본 속성 (Basic Attribute)

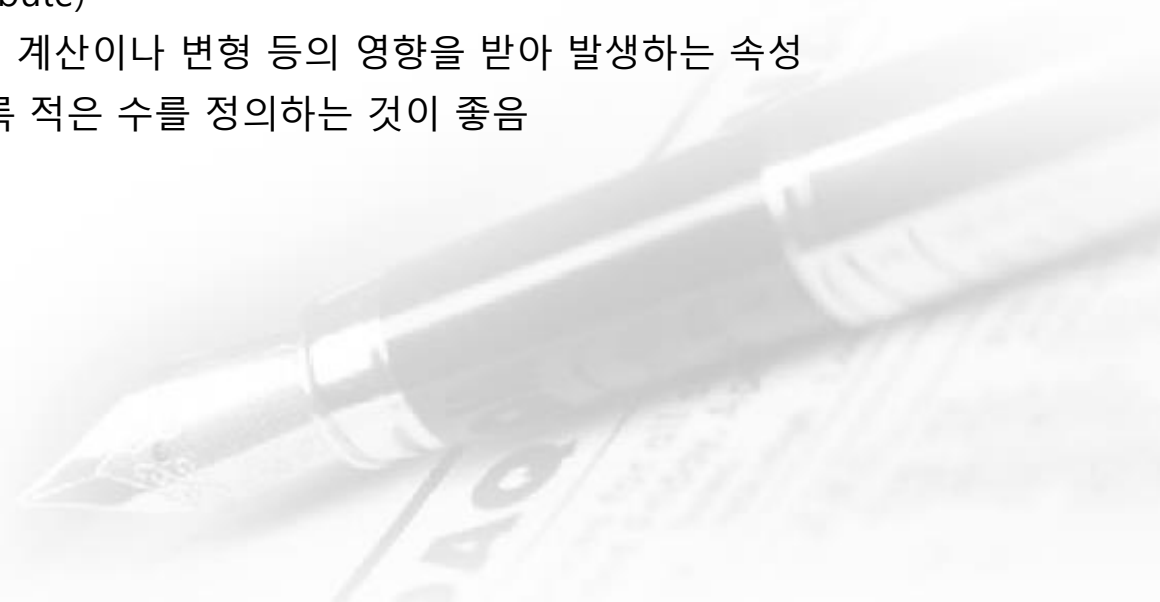
- 업무 분석을 통해 정의한 속성으로 속성 중 가장 많고 일반적
- 업무로부터 분석한 속성이라도 업무상 코드로 정의한 속성은 기본 속성에서 제외

#### □ 설계 속성(Designed Attribute)

- 원래 업무상 존재하지 않고 설계 과정에서 도출해내는 속성
- 업무에 필요한 데이터 외에 데이터 모델링을 위해 업무를 규칙화하려고 속성을 새로 만들거나 변형하여 정의하는 속성

#### □ 파생 속성 (Derived Attribute)

- 다른 속성으로부터 계산이나 변형 등의 영향을 받아 발생하는 속성
- 파생 속성은 되도록 적은 수를 정의하는 것이 좋음



# 데이터 모델

## ❖ 관계 및 관계 타입

- ✓ 관계는 2개 이상의 개체 사이에 존재하는 연관성을 뜻하며 관계 타입은 같은 관계들의 집합
- ✓ 관계는 관계에 참여하는 개체 타입의 개수에 대한 차수(Degree)와 관계에 참여하는 개체 어커런스의 개수에 대한 대응 카디널리티(Mapping Cardinality)를 서유
- ✓ 차수(Degree)에 따른 관계의 종류
  - 단항(Unary) 관계 : 관계에 참여하고 있는 개체 타입이 1개인 관계
  - 이항(Binary) 관계 : 관계에 참여하고 있는 개체 타입이 2개인 관계
  - 삼항(Ternary) 관계 : 관계에 참여하고 있는 개체 타입 이 3개인 관계
  - n항(n-ary) 관계 : 관계에 참여하고 있는 개체 타입이 n개인 관계
- ✓ 대응 카디널리티에 따른 관계의 종류
  - 1:1 관계: 관계에 참여하고 있는 두 개체 타입이 모두 하나씩의 개체 어커런스를 갖는 관계
  - 1:N 관계: 관계에 참여하고 있는 개체 타입 중 한 개체 타입은 여러 개의 개체 어커런스를 가질 수 있고, 다른 한 개체타입은 하나의 개체 어커런스를 갖는 관계
  - N:M 관계: 관계에 참여하고 있는 두 개체 타입 모두 여러 개의 개체 어커런스를 가질 수 있는 관계

# 데이터 모델

## ❖ 관계 및 관계 타입

### ✓ 관계의 종류

- 존재 관계: 엔티티 간의 상태를 의미하는 것으로 학과와 학생처럼 학생이 아무런 행위를 하지 않아도 특정 학과에 소속되는 경우
- 행위 관계: 엔티티 간의 행위가 있는 것으로 특정 행위를 해야만 만들어지는 관계

### ✓ 업무 형태에 따른 관계의 종류

- 종속(Dependent) 관계
  - 두 개체 사이의 주·종 관계를 표현한 것으로, 식별 관계와 비식별 관계가 있음
  - 식별 관계 : 개체 A, B 사이의 관계에서 A 개체의 기본키가 B 개체의 외래키이면서 동시에 기본키가 되는 관계
  - 비식별 관계 : 개체 A, B 사이의 관계에서 A 개체의 기본키가 B 개체의 비기본키 영역에서 외래키가 되는 관계
- 중복(Redundant) 관계 : 두 개체 사이에 2번 이상의 종속 관계가 발생하는 관계
- 재귀(Recursive) 관계: 개체가 자기 자신과 관계를 갖는 것으로, 순환 관계(Recursive Relationship)라고도 함
- 배타(Exclusive)관계
  - 개체의 속성이나 구분자를 기준으로 개체의 특성을 분할하는 관계로, 배타 AND 관계와 배타 OR 관계로 구분
  - 배타 AND 관계는 하위 개체들 중 속성이나 구분자 조건에 따라 하나의 개체만을 선택 할 수 있고, 배타 OR 관계는 하나 이상의 개체를 선택



# 데이터 모델

## ❖ 식별자(Identifier):

- ✓ 대표성 여부에 따른 식별자의 분류
  - 주 식별자(Primary Identifier): 개체를 대표하는 유일한 식별자
    - 유일성, 최소성, 불변성, 존재성을 가짐
  - 보조 식별자(Alternate Identifier): 주 식별자를 대신하여 개체를 식별할 수 있는 속성
- ✓ 스스로 생성되었는지에 따른 식별자의 분류
  - 내부 식별자(Internal Identifier): 개체 내에서 스스로 만들어지는 식별자
  - 외부 식별자(Foreign Identifier): 다른 개체와의 관계(Relationship)에 의해 외부 개체의 식별자를 가져와 사용하는 식별자로 외부 식별자는 자신의 개체에서 다른 개체를 찾아 가는 연결자 역할을 수행
- ✓ 단일 속성으로 이루어 졌는지에 따른 식별자의 분류
  - 단일 식별자(Single Identifier) : 주 식별자가 한 가지 속성으로만 구성된 식별자
  - 복합 식별자(Composit Identifier) : 주 식별자가 두 개 이상의 속성으로 구성된 식별자
- ✓ 기존 식별자를 대체하기 위해 만들어졌는지에 따른 식별자의 분류
  - 원조 식별자(Original Identifier): 업무에 의해 만들어지는 가공되지 않은 원래의 식별자로 본질 식별자라고도 함
  - 대리 식별자(Surrogate Identifier) : 주 식별자의 속성이 두 개 이상인 경우 속성들을 하나의 속성으로 묶어 사용하는 식별자로 인조 식별자
- ✓ 기타 식별자
  - 후보 식별자(Candidate Identifier) : 개체에서 각 인스턴스를 유일하게 식별할 수 있는 속성 또는 속성 집합

# E-R(Entity-Relation)데이터 모델

## ❖ E-R 모델

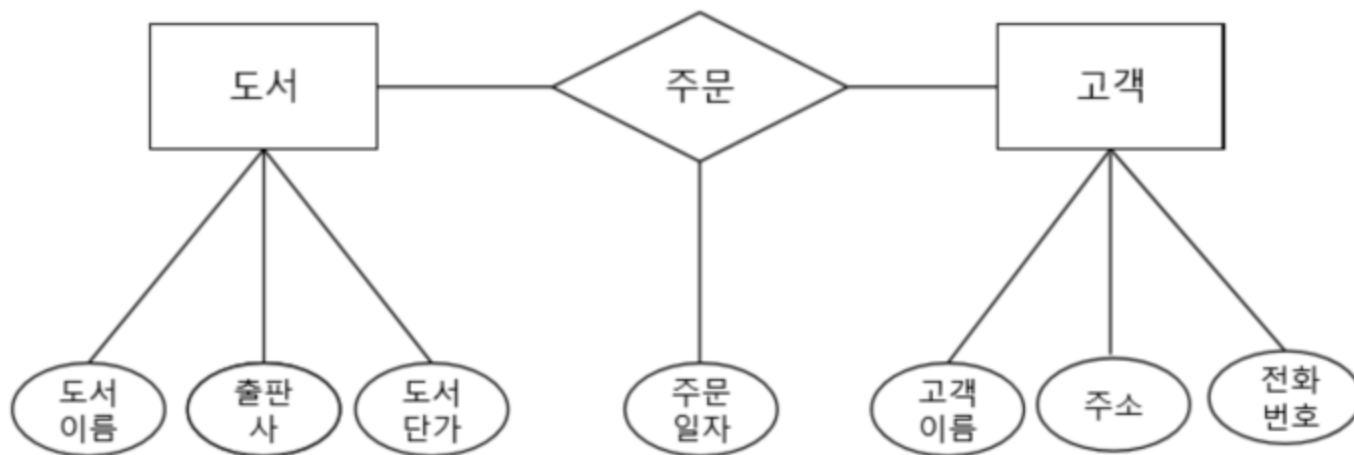
- ✓ 개체와 개체 사이의 관계를 기본 요소로 해서 현실 세계의 데이터를 개념적인 데이터 모델로 변환하기 위한 방법
- ✓ 1976년 피터 첸(Peter Chen)에 의해 제안



# ER Diagram

## ❖ ER Diagram


- ✓ ER diagram 이란 Entity-Relationship Model을 표현하는 것으로 현실세계의 요구사항 (Requirements)들로 부터 Database를 설계하는 과정에서 활용
- ✓ 개념을 모델링하는 것으로 개체(entity)와 속성(attribute), 관계성(relationship)을 표현
- ✓ Peter Cheng's Notation 으로 표현



# ER Diagram

## ❖ ER Diagram

개체(Entities)

 (Regular, Strong) entity type

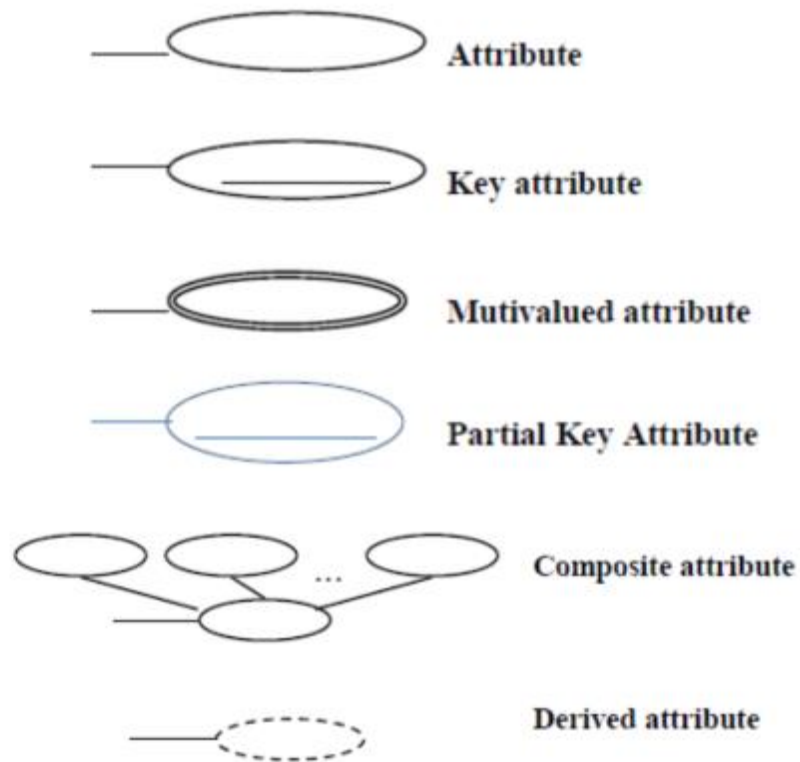
 Weak entity type



# ER Diagram

## ❖ ER Diagram

속성(Attributes)



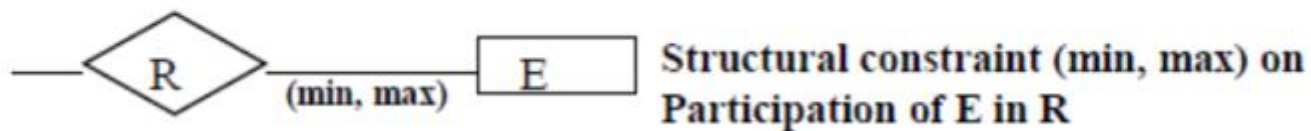
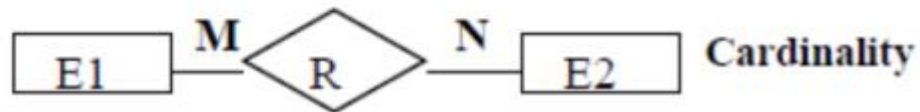
# ER Diagram

## ❖ ER Diagram

기호	의미
-----	<ul style="list-style-type: none"><li>• 비식별자 관계(non-identifying relationship): 강한 개체 타입</li><li>• 부모 개체의 키가 일반 속성으로 포함되는 관계</li></ul>
_____	<ul style="list-style-type: none"><li>• 식별자 관계(identifying relationship): 약한 개체 타입</li><li>• 부모 개체의 키가 주식별자로 포함되는 관계</li></ul>
—————<	<ul style="list-style-type: none"><li>• 일대다(1:N)의 관계: N 쪽에 새발을 표시</li></ul>
—————○	<ul style="list-style-type: none"><li>• 0(선택 참여, 최소 참여가 0일 경우</li></ul>
—————+	<ul style="list-style-type: none"><li>• 1(필수 참여, 최소 참여가 1일 경우</li></ul>

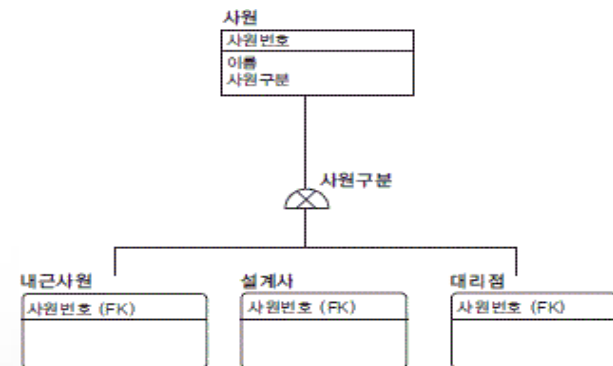
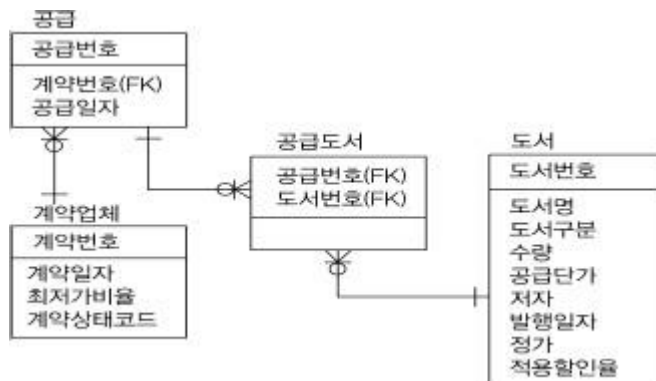
# ER Diagram

## ❖ ER Diagram



# ER Diagram

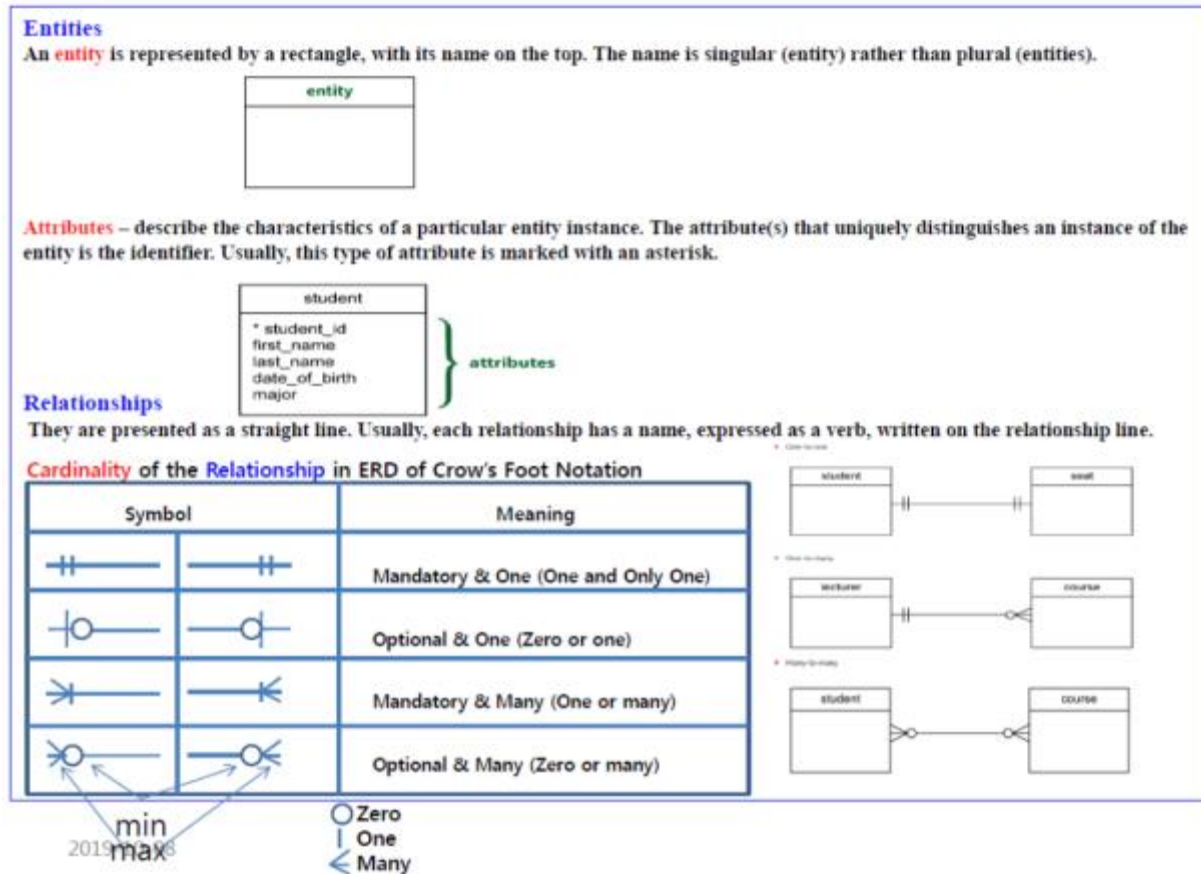
- ❖ ER Diagram: 정보 공학(IE – 까마귀 발)표기법
  - ✓ 제임스 마틴(James Martin)이 주창한 Information Engineering에서 사용하는 Data 모델 표기법
  - ✓ 까마귀발 모양과 같은 형태의 관계 표현 때문에 Crow's Foot Model로 불리기도 함
  - ✓ 가장 널리 사용되는 표기법의 하나이지만 서브 타입과 같은 개체 집합의 상세 표현에 있어서 공간을 많이 차지하는 단점이 있음





# ER Diagram

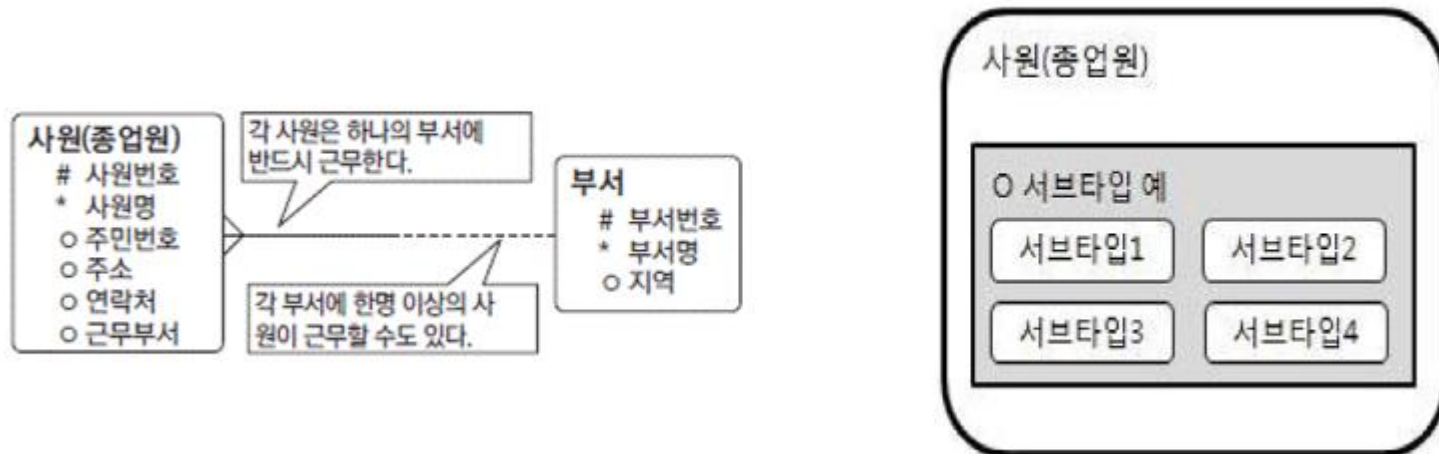
## ❖ ER Diagram: 정보 공학(IE – 까마귀 발)표기법



# ER Diagram

## ❖ ER Diagram: Barker's Notation

- ✓ 1986년, 영국 컨설팅 회사 CACI에 근무하던 Richard Barker, Lan Palmer, Harry Ellis 등에 의해 처음 개발
- ✓ Original Chen style 에서 사용되던 'Crow's Foot' style을 변형하여 개발
- ✓ 후에 오라클로 이직한 리차드 바커에 의해 오라클에서 Case Method로 채택하여 사용되고, Oracle CASE modeling tools에 적용
- ✓ 오늘날 널리 사용되는 표기법의 하나로 개체 집합과 그 관계의 표현에 있어서 직관적 이해성이 뛰어나고 구체적이고 상세한 표현성 및 공간 활용의 장점 등으로 인해 사용자가 지속적으로 증가



# ER Diagram

## ❖ ER Diagram: Barker's Notation

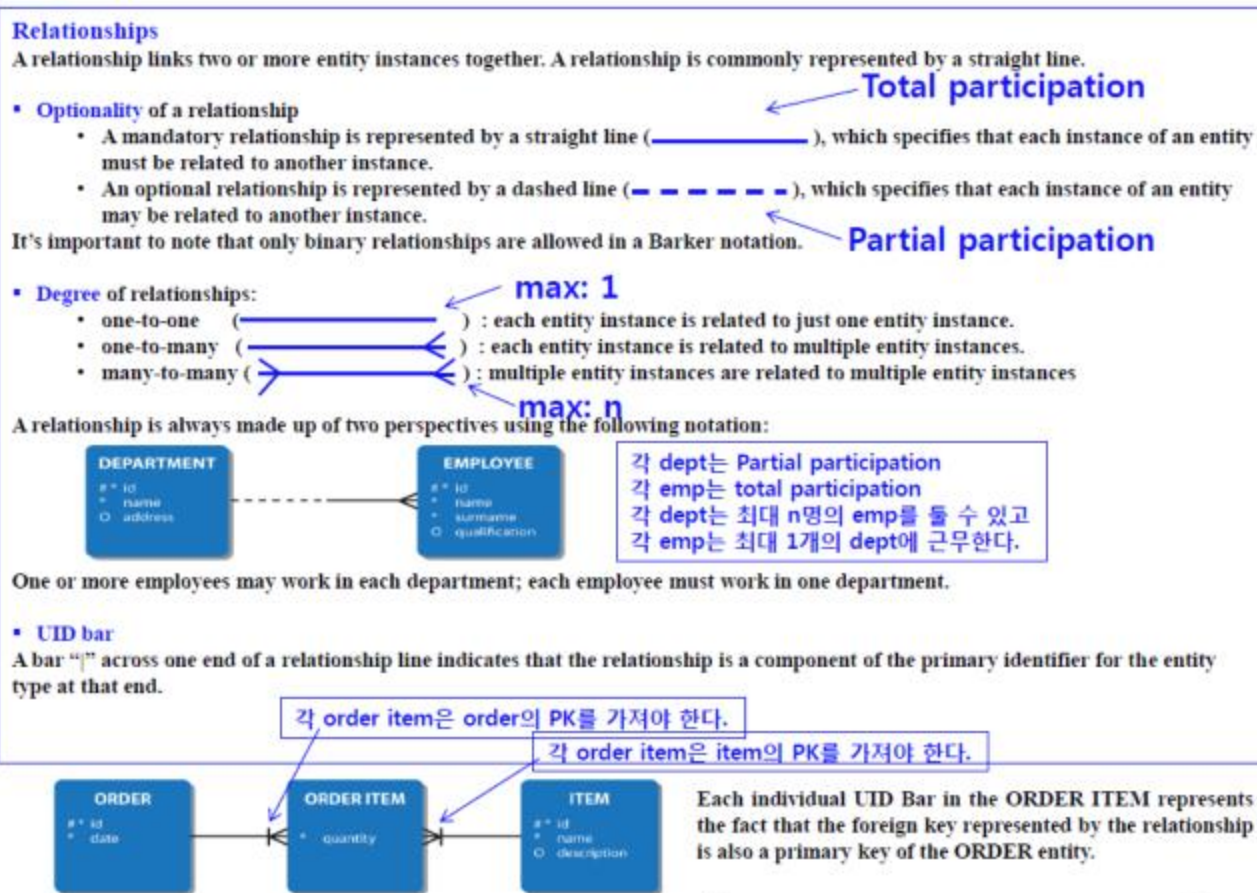
Attributes – describe the characteristics of a particular entity instance. An attribute can be of three types:

- Unique Identifier – uniquely identifies an entity instance
- Mandatory – its value cannot be null
- Optional – its value can be null











# ER Diagram

## ❖ ER Diagram: Barker's Notation



# ER Diagram

## ❖ ERD 관계 관계 차수와 선택성 표시

관계	선택성	IE 표기법	Barker 표기법
1 : 1	필수		
1 : 1	선택		
1 : n	필수		
1 : n	선택		



# 관계형 데이터베이스

## ❖ 관계형 데이터베이스

### ✓ 개요

- 테이블의 집합으로 데이터를 표현하는 데이터베이스
- 1970년 IBM의 근무하던 E.F.Codd에 의해서 제안
- 간결하고 보기 편리하며 많은 종류의 DBMS가 있음
- 성능이 떨어지는 단점이 있음
- 모든 관계 표현이 가능



# 관계형 데이터베이스

## ❖ 관계형 데이터베이스

- ✓ 릴레이션: 데이터들을 표(Table)의 형태로 표현한 것으로 구조를 나타내는 릴레이션 스키마와 실제 값들인 릴레이션 인스턴스로 구성
  - 릴레이션을 구성하는 속성 사이에는 순서가 없다.
  - 릴레이션을 구성하는 튜플 사이에는 순서가 없다.
  - 속성의 이름은 중복될 수 없다.
  - 속성의 값은 더 이상 쪼갤 수 없는 원자값이어야 한다.
  - 튜플을 구별할 수 있는 키가 있어야 한다.



# 관계형 데이터베이스

## ❖ 관계형 데이터베이스

### ✓ 속성(Attribute)

- 릴레이션을 구성하는 각각의 열(Column)
- 데이터베이스를 구성하는 가장 작은 논리적 단위
- 파일 구조 상의 데이터 항목 또는 데이터 필드에 해당
- 개체의 특성을 기술
- 속성의 수 = 디그리(Degree) = 차수

### ✓ 튜플(Tuple)

- 릴레이션을 구성하는 각각의 행
- 속성의 모임으로 구성
- 파일 구조에서 레코드(Record)와 같은 의미
- 튜플의 수 = 카디널리티(Cardinality) = 기수 = 대응

### ✓ 도메인(Domain)

- 하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(Atomic)값들의 집합
- 실제 애트리뷰트 값이 나타날 때 그 값의 합법 여부를 시스템이 검사하는 데에도 이용

### ✓ 릴레이션 인스턴스(Relation Instance): 데이터 개체를 구성하고 있는 속성들에 데이터 타입이 정의되어 구체적인 데이터 값을 갖고 있는 것을 말한



# 관계형 데이터베이스

## ❖ 키(Key)

- ✓ 키(Key)는 중복된 값이 없는 속성 또는 속성의 집합
- ✓ 키의 종류
  - 슈퍼키(Super Key): 유일성(Unique)은 만족하지만 최소성 (Minimality)은 만족하지 못하는 키
  - 후보키(Candidate Key): 유일성 + 최소성
  - 기본키(Primary Key): 후보키 중에서 특별히 선정된 키로 튜플을 구별하기 위한 유일한 키이며 릴레이션 내에서 유일한 식별자
  - 대체키(Alternate Key): 후보키 중에서 기본키를 제외한 나머지 후보키
  - 외래키(Foreign Key): 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합을 의미하며 릴레이션 간 의 관계를 표현할 때 사용



# 관계형 데이터베이스

## ❖ 무결성(Integrity)

- ✓ 데이터베이스에 저장된 데이터 값과 그것이 표현하는 현실 세계의 실제 값이 일치하는 정확성을 의미
- ✓ 무결성 제약조건 종류
  - 개체 무결성(Entity Integrity, 실체 무결성): 기본키를 구성하는 어떤 속성도 Null 값이나 중복 값을 가질 수 없음
  - 참조 무결성(Referential Integrity): 외래키 값은 Null이거나 참조 릴레이션의 기본키 값과 동일해야 한다. 즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다는 규정
  - 도메인 무결성(Domain Integrity, 영역 무결성): 주어진 속성 값이 정의된 도메인에 속한 값이어야 한다.
  - 키(Key) 무결성 : 하나의 릴레이션에는 적어도 하나의 키가 존재해야 한다는 규정
  - NULL 무결성
  - Unique 무결성
  - 관계 무결성
- ✓ 무결성 강화 방법
  - Application Validation Check: 응용 프로그램에서 데이터의 유효성 검사를 수행
  - 데이터베이스의 trigger 이용: DML 수행 전이나 후에 수행할 작업을 절차형 SQL로 작성
  - 데이터베이스의 무결성 제약조건 설정

# 테이블

## ❖ 테이블

- ✓ 관계형 데이터베이스의 가장 기본적인 개체
- ✓ 테이블을 저장하면 논리적으로는 테이블 스페이스에 저장되고, 물리적으로는 해당 테이블과 연관된 데이터 파일(Data File)에 저장됨
- ✓ 데이터베이스를 테이블, 테이블 스페이스, 데이터 파일로 나눠 관리하면 논리적 구성이 물리적 구성에 종속되지 않아 투명성이 보장됨

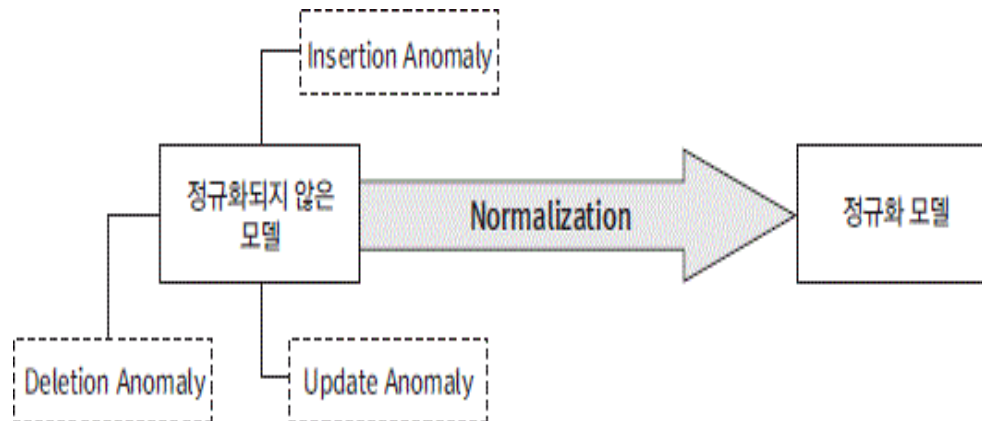
# 이상

## ❖ 이상(Anomaly)

- ✓ 테이블에서 일부 속성들의 종속으로 인해 데이터의 중복(Redundancy)이 발생하고 이 중복으로 인해 테이블 조작 시 문제가 발생하는 현상
- ✓ 이상의 종류
  - 삽입 이상(Insertion Anomaly): 테이블에 데이터를 삽입 할 때 의도와는 상관없이 원하지 않은 값들로 인해 삽입할 수 없게 되는 현상
  - 삭제 이상(Deletion Anomaly): 테이블에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 현상
  - 갱신 이상(Update Anomaly): 테이블에서 튜플에 있는 속성 값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 불일치성(Inconsistency)이 생기는 현상

# 이상 현상

- ❖ 데이터의 중복으로 인한 이상 현상
  - ✓ 변경 이상 (Modification Anomaly)
  - ✓ 삽입 이상 (Insertion Anomaly)
  - ✓ 삭제 이상 (Deletion Anomaly)



# 이상 현상

## ❖ 이상 현상(Anomaly)

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

고객아이디	이벤트번호	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!

# 이상 현상

## ❖ 이상 현상(Anomaly)

- ✓ 삭제 이상: 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
<del>orange</del>	<del>E004</del>	<del>N</del>	<del>김용욱</del>	<del>silver</del>

← 데이터 손실 발생!

© 2000

# 이상 현상

## ❖ 이상 현상(Anomaly)

- ✓ 갱신 이상: 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!



# 함수적 종속(FD)

## ❖ 함수적 종속(Functional Dependency)

- ✓ 어떤 테이블 R에서 X와 Y를 각각 R의 속성 집합의 부분 집합이라 할 때 속성 X의 값 각각에 대해 속성 Y의 값이 오직 하나만 연관되어 있을 때 Y는 X에 함수적 종속 또는 X가 Y를 함수적으로 결정한다고 하고  $X \rightarrow Y$ 로 표기
- ✓  $X \rightarrow Y$ 의 관계를 갖는 속성 X와 Y에서 X를 결정자(Determinant)라 하고 Y를 종속자(Dependent)라고 함
- ✓ 완전 함수적 종속: 어떤 테이블 R에서 2개 이상의 속성(X, Y) 모두가 있는 경우에만 하나의 속성 Z를 종속하는 경우를 완전 함수적 종속이라고 함
- ✓ 부분 함수적 종속: 어떤 테이블 R에서 2개 이상의 속성(X, Y) 중 일부분만으로 Z를 종속하는 경우를 부분 함수적 종속이라고 함
- ✓ 이행적 함수 종속:  $A \rightarrow B$  이고  $B \rightarrow C$  이면  $A \rightarrow C$

# 함수적 종속

## ❖ 함수적 종속(Functional Dependency)

학번	이름	나이	성별	전공코드	전공명
110011	박지현	26	여성	AAA1	국문학과
110011	박지현	26	여성	C0B7	컴퓨터공학과
131001	김민석	25	남성	C0A5	전기전자공학과
120006	홍현희	25	여성	B1027	무용과
150705	한태민	23	남성	C0A5	전기전자공학과
171024	설화영	22	여성	B01K2	공예과

- ✓ '학번'을 알면 '이름', '나이', '성별' 속성을 식별할 수 있으며, '학번'이 다르면 그에 따른 값도 다름
- ✓ '이름', '나이', '성별' 속성은 '학번'에 함수적인 종속관계
- ✓ 전공 속성 또한 '전공코드'에 함수적인 종속관계
- ✓ 학번→이름, 학번→나이, 학번→성별

# 함수적 종속

## ❖ 함수적 종속(Functional Dependency)

- ✓ 완전 함수적 종속(Full Functional Dependency): 완전 함수적 종속이란 종속자가 기본키에 종속되며, 기본키가 여러 속성으로 구성되어 있을 경우 기본키를 구성하는 모든 속성이 포함된 기본키의 부분집합에 종속된 경우

회원번호	이름	나이	거주지역
A001	송민지	17	서울
A002	박아람	15	부산
A003	이예은	16	대전

- 이 릴레이션에서는 기본키가 '회원번호' 속성으로 구성되어 있는데 여기서 '이름', '나이', '거주지역' 속성은 기본키인 '회원번호'를 알아야 식별 가능하기 때문에 '이름', '나이', '거주지역'은 '회원번호'에 완전 함수 종속된 관계

# 함수적 종속

## ❖ 함수적 종속(Functional Dependency)

### ✓ 완전 함수적 종속(Full Functional Dependency)

<u>고객ID</u>	<u>상품코드</u>	주문상품	수량	가격
AAAA01	T001	티셔츠	2	12000
AAAA01	B110	청바지	1	11000
AAAA02	B110	청바지	2	22000
AAAA03	T091	와이셔츠	1	15000
AAAA03	O100	원피스	1	19000

- 해당 릴레이션의 기본키는 '고객ID'와 '상품코드' 속성으로 구성되어 있는데 '수량' 속성은 기본키를 구성하는 '고객ID', '상품코드' 속성을 모두 알아야 식별할 수 있기 때문에 '수량'은 완전 함수 종속된 관계

# 함수적 종속

## ❖ 함수적 종속(Functional Dependency)

- ✓ 부분 함수적 종속(Partial Functional Dependency): 기본키가 여러 속성으로 구성되어 있을 경우 기본키를 구성하는 속성 중 일부에 종속되는 경우

<u>고객ID</u>	<u>제품코드</u>	주문상품	수량	가격
AAAA01	T001	티셔츠	2	12000
AAAA01	B110	청바지	1	11000
AAAA02	B110	청바지	2	22000
AAAA03	T091	와이셔츠	1	15000
AAAA03	O100	원피스	1	19000

- 기본키가 '고객ID'와 '상품코드' 속성으로 구성된 위의 릴레이션에서 '주문상품'은 기본키 중 '상품코드'만 알아도 식별할 수 있는데 '주문상품' 속성은 기본키에 부분 함수 종속된 관계

# 함수적 종속

## ❖ 함수적 종속(Functional Dependency)

- ✓ 이행적 함수 종속(Transitive Functional Dependency): 릴레이션에서  $X, Y, Z$ 라는 3 개의 속성이 있을 때  $X \rightarrow Y, Y \rightarrow Z$  이란 종속 관계가 있을 경우,  $X \rightarrow Z$ 가 성립될 때 이행적 함수 종속이라고 하는데  $X$ 를 알면  $Y$ 를 알고 그를 통해  $Z$ 를 알 수 있는 경우

<u>회원번호</u>	생년월일	나이	주소
A001	1995.1.3	26	군포 산본
A002	1996.1.16	25	강남 청담
A003	1997.2.11	24	오كل랜드
A004	1997.3.27	23	부리람

- 회원번호를 알면 생년월일을 알 수 있고 생년월일을 알면 나이를 알 수 있으면 회원번호를 알면 나이를 알 수 있는데 이러한 관계가 이행적 함수 종속

# 정규화

## ❖ 정규화(Normalization)

- ✓ 테이블의 속성들이 상호 종속적인 관계를 갖는 특성을 이용하여 테이블을 무손실 분해하는 과정
- ✓ 데이터의 일관성, 최소한의 데이터 중복, 최대한의 데이터 유연성을 위한 방법으로 데이터를 분해하는 과정
- ✓ 데이터 모델의 독립성을 확보하기 위한 방법
- ✓ 정규화를 수행하면 비즈니스의 변경에 유연하게 대처해서 데이터 모델의 변경을 최소화 할 수 있음
- ✓ 정규화의 장점
  - 중복 값이 줄어든다
  - NULL 값이 줄어든다
  - 복잡한 코드로 데이터 모델을 보완할 필요가 없다
  - 새로운 요구 사항의 발견 과정을 돕는다
  - 업무 규칙의 정밀한 포착을 보증한다
  - 데이터 구조의 안정성을 최대화한다

# 정규화

## ❖ 정규화(Normalization)

### ✓ 종류





# 정규화

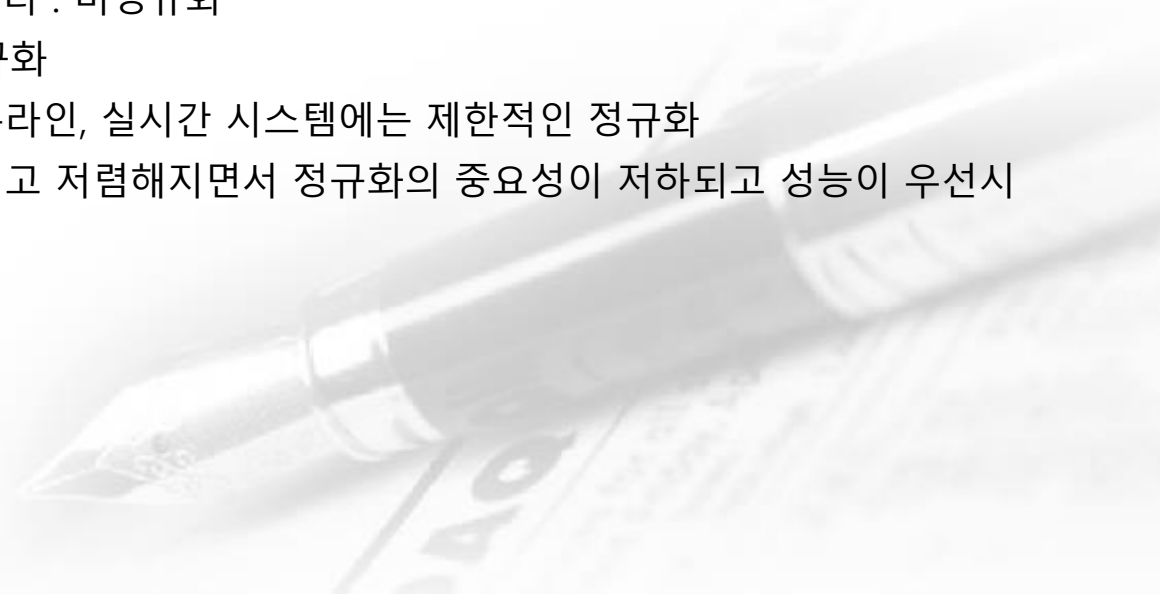
## ❖ 정규화의 문제점 과 해결방안

### ✓ 정규화의 문제점

- ❑ 빈번한 Join 연산의 증가 : 시스템 성능 저하
- ❑ 부자연스러운 DB Semantic 초래
- ❑ 조회/검색 위주의 응용시스템에 부적합

### ✓ 문제점 해결방안 및 업계 동향

- ❑ 정규화 완료 후 업무 특성과 성능 향상을 위해 De-normalization 수행으로 유연성 확보
- ❑ 업무특성 별 정규화 수준
  - 온라인 처리 : 소규모 정규화
  - 단순 조회 용 데이터 : 비정규화
  - 배치 처리 : 비정규화
- ❑ 응답시간이 요구되는 온라인, 실시간 시스템에는 제한적인 정규화
- ❑ 기억장치가 대용량화 되고 저렴해지면서 정규화의 중요성이 저하되고 성능이 우선시 됨



# 반정규화(Denormalization)

## ❖ 반정규화(Denormalization)

- ✓ 시스템의 성능 향상, 개발 및 운영의 편의성 등을 위해 정규화된 데이터 모델을 통합, 중복, 분리하는 과정으로 의도적으로 정규화 원칙을 위배하는 행위로 역 정규화
- ✓ 구현 방법
  - 테이블 통합: 두 개의 테이블이 조인(Join)되는 경우가 많으면 하나의 테이블로 합쳐 사용하는 것이 성능 향상에 도움이 될 경우 수행
  - 테이블 분할
    - 수평 분할(Horizontal Partitioning): 레코드 (Record)를 기준으로 테이블을 분할하는 것으로 레코드 별로 사용 빈도의 차이가 큰 경우 사용 빈도에 따라 테이블을 분할
    - 수직 분할(Vertical Partitioning): 하나의 테이블 에 속성이 너무 많을 경우 속성을 기준으로 테이블을 분할
  - 중복 테이블 추가
    - 여러 테이블에서 데이터를 추출해서 사용해야 하거나 다른 서버에 저장된 테이블을 이용해야 하는 경우 중복 테이블을 추가하여 작업의 효율성을 향상시킬 수 있음
    - 추가 방법 : 집계 테이블의 추가, 진행 테이블 의 추가, 특정 부분만을 포함하는 테이블의 추가
  - 중복 속성 추가: 조인해서 데이터를 처리할 때 데이터를 조회하는 경로를 단축하기 위해 자주 사용하는 속성을 하나 더 추가하는것

# 반정규화(Denormalization)

## ❖ 반정규화를 통한 성능향상 전략

### ✓ 반정규화

- ❑ 반정규화는 정규화된 엔티티,속성,관계를 시스템의 성능향상 및 개발과 운영의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링 기법
- ❑ 시스템의 성능 향상, 개발 및 운영의 편의성 등을 위해 정규화된 데이터 모델을 통합, 중복, 분리하는 과정으로 의도적으로 정규화 원칙을 위배하는 행위로
- ❑ 디스크 I/O량이 많아서 조회 시 성능이 저하되거나, 테이블끼리의 경로가 너무 멀어 조인으로 인한 성능저하가 예상되거나, 컬럼을 계산하여 읽을 때 성능이 저하될 것이 예상되는 경우 반정규화를 수행
- ❑ 업무적으로 조회에 대한 처리 성능이 중요하다고 판단될 때 부분적으로 반정규화를 고려하는데 반정규화를 수행하게 되면 모델의 유연성은 떨어지게 됨
- ❑ 설계단계에서 반정규화를 적용하게 되며 반정규화 미수행시에는 다음과 같은 현상이 발생할 수 있음
  - 성능이 저하된 데이터베이스가 생성될 수 있음
  - 구축 단계나 시험 단계에서 반정규화를 적용할 때 수정에 따른 노력 비용이 많이 소모

# 반정규화

## ❖ 반정규화 기법

### ✓ 테이블 반정규화

#### □ 테이블 추가

- 중복 테이블 추가: 다른 업무이거나 서버가 다른 경우 동일한 테이블구조를 중복하여 원격 조인을 제거하여 성능을 향상
- 통계 테이블 추가: SUM, AVG 등을 미리 수행하여 계산해 둬으로써 조회시 성능을 향상
- 이력 테이블 추가: 이력 테이블 중에서 마스터 테이블에 존재하는 레코드를 중복하여 이력 테이블에 존재시키는 방법
- 부분 테이블 추가: 하나의 테이블을 전체 컬럼 중 자주 이용하는 집중화된 컬럼이 있을 경우, 디스크 I/O를 줄이기 위해 해당 컬럼들을 모아놓은 별도의 반정규화된 테이블을 생성



# 반정규화

## ❖ 반정규화 기법

### ✓ 테이블 반정규화

#### □ 테이블 병합

- 1:1 관계 테이블 병합 - 1:1 관계를 통합하여 성능향상
- 1:M 관계 테이블 병합 - 1:M 관계를 통합하여 성능향상
- 슈퍼/서브타입 테이블 병합 - 슈퍼/서브 관계를 통합하여 성능향상
  - Super type: 여러 Sub Type이 공통으로 가지는 내용을 소유하고 있는 타입
  - Sub Type: 자신만의 가져야 하는 내용을 소유하고 있는 타입



# 반정규화

## ❖ 반정규화 기법

### ✓ 테이블 반정규화

#### □ 테이블 분할 - 파티셔닝

##### ○ 장점

- 관리적 측면 : partition 단위 백업, 추가, 삭제, 변경
  - 전체 데이터를 손실할 가능성이 줄어들어 데이터 가용성이 향상
  - partition별로 백업 및 복구가 가능
  - partition 단위로 I/O 분산이 가능하여 UPDATE 성능이 향상
- 성능적 측면 : partition 단위 조회 및 DML수행
  - 데이터 전체 검색 시 필요한 부분만 탐색해 성능이 증가
    - Full Scan에서 데이터 Access의 범위를 줄여 성능 향상
    - 필요한 데이터만 빠르게 조회할 수 있기 때문에 쿼리 자체가 가벼워짐

##### ○ 단점

- table간 JOIN에 대한 비용이 증가
- table과 index를 별도로 파티셔닝할 수 없음
- table과 index를 같이 파티셔닝해야 하

# 반정규화

## ❖ 반정규화 기법

### ✓ 테이블 반정규화

#### □ 대량 데이터발생에 따른 테이블 분할

##### ▪ 대량 데이터가 발생하는 테이블의 문제점

- 설계가 잘 되어 있는 데이터 모델이라도 대량의 데이터가 하나의 테이블에 집약되어 있고 하나의 하드웨어 공간에 저장되어 있으면 성능 저하를 피하기 어려움
- 인덱스도 또한 트리가 커지고 깊이가 깊어져, 조회 성능에 영향을 미침
- 입력/수정/삭제의 트랜잭션인 경우도 인덱스의 특성상 일의 양이 증가하여, 더 많은 성능저하를 유발
- 컬럼이 많아지게 되면 물리적인 디스크의 여러 블록에 걸쳐 데이터가 저장되게 되며, 로우 길이가 너무 길어서 로우 체이닝과 로우 마이그레이션이 많아지게 되어 성능이 저하

##### ▪ 한 테이블에 많은 수의 컬럼을 가지고 있는 경우

- 200개의 컬럼을 가진 도서 정보 테이블이 있다고 가정하고, 하나의 로우 길이가 10K이고 블록이 2K단위로 쪼개져 있으면, 로우는 대략 5개의 블록에 걸쳐 저장
- 여러 블록에 컬럼이 흩어져 있다면, 디스크 I/O가 많이 발생
- 트랜잭션 발생시 어떤 컬럼에 대해 집중적으로 발생되는지 분석하여 테이블 분할을 하면 디스크 I/O가 감소하여 성능을 개선할 수 있음

# 반정규화

## ❖ 반정규화 기법

### ✓ 테이블 반정규화

#### □ 테이블 분할 - 파티셔닝

##### ○ 종류

- 수직 분할 - 컬럼 단위의 테이블을 디스크 I/O를 분산처리하기 위해 테이블을 1:1로 분리하여 성능 향상(트랜잭션의 처리되는 유형 파악이 선행)
- 수평 분할(Sharding) - 로우 단위로 집중 발생하는 트랜잭션을 분석하여 디스크 I/O 및 데이터 접근의 효율성을 높여 성능을 향상하기 위해 로우 단위로 테이블을 쪼갬(관계가 없음)
- 수직/수평 분할 절차
  - 데이터 모델링을 완성
  - 데이터베이스 용량을 산정
  - 대량 데이터가 처리되는 테이블에 대해서 트랜잭션 처리 패턴을 분석
  - 컬럼 단위로 집중화된 처리가 발생하는지, 로우 단위로 집중화된 처리가 발생하는지 분석하여 집중화된 단위로 테이블을 분리하는 것을 검토
- 컬럼 수가 너무 많은 경우는 테이블을 1:1형태로 분리할 수 있는지 검증하고 컬럼 수가 적지만 데이터 용량이 많아 성능저하가 예상되는 경우 테이블에 대해 파티셔닝 전략을 고려



# 반정규화

## ❖ 반정규화 기법

### ✓ 테이블 반정규화

#### □ 테이블 분할 - 파티셔닝

##### ○ 분할 기준

- 범위 분할 (range partitioning): 분할 키 값이 범위 내에 있는지 여부로 구분하는데 우편 번호를 분할 키로 수평 분할하는 경우
- 목록 분할 (list partitioning): 값 목록에 파티션을 할당 분할 키 값을 그 목록에 비추어 파티션을 선택하는 경우로 Country 라는 컬럼의 값이 Iceland, Norway, Sweden, Finland, Denmark 중 하나에 있는 행을 빼낼 때 북유럽 국가 파티션을 구축 할 수 있음
- 해시 분할 (hash partitioning): 해시 함수의 값에 따라 파티션에 포함할지 여부를 결정하는 것으로 4개의 파티션으로 분할하는 경우 해시 함수는 0-3의 정수를 리턴
- 합성 분할 (composite partitioning): 여러 기술을 결합하는 것을 의미하며, 예를 들면 먼저 범위 분할하고, 다음에 해시 분할 같은 것을 추가하는 방법으로 컨시스턴트 해시법은 해시 분할 및 목록 분할의 합성으로 간주 될 수 있고 키 공간을 해시 축소함으로써 일람할 수 있도록 함

# 반정규화

## ❖ 반정규화 기법

### ✓ 테이블 반정규화

□ 테이블 분할 - 오라클의 경우 LIST PARTITION, RANGE PARTITION, HASH PARTITION, COMPOSITE PARTITION 등이 가능

#### ○ RANGE PARTITION

- 요금정보처럼 항상 월단위로 데이터를 처리하는 특성을 가진 경우, PK인 요금일자의 년+월을 이용하여 12개의 파티션테이블을 만들어서 성능 개선을 유도
- 대상 테이블의 컬럼의 값이 날짜 또는 숫자 값으로 분리가 가능하고 각 영역 별로 트랜잭션이 분리된다면 적용
- 데이터 보관 주기에 따라 테이블에 데이터를 쉽게 지우는 것이 가능하므로 (파티션테이블을 DROP) 테이블 관리가 용이

#### ○ LIST PARTITION

- 지점, 사업소, 사업장, 핵심적인 코드 값 등으로 PK가 구성되어 있는 테이블 이라면, 값 각각에 의해 파티셔닝이 되는 LIST PARTITION을 적용
- 특정 값에 따라 분리 저장할 수는 있으나, RANGE PARTITION과 같이 데이터 보관 주기에 따라 쉽게 삭제하는 기능은 제공될 수 없음

#### ▪ HASH PARTITION 적용

- HASH조건에 따라 해시알고리즘이 적용되어 테이블이 분리되므로 설계자는 데이터가 어떤 테이블에 어떻게 들어갔는지 알 수 없으며 보관주기에 따라 쉽게 삭제하는 기능은 제공 될 수 없음

# 반정규화

## ❖ 반정규화 기법

### ✓ 컬럼 반정규화

- ❑ 중복 컬럼 추가: 조인시 성능저하를 예방하기 위해,중복된 컬럼을 위치시킴
- ❑ 파생 컬럼 추가: 트랜잭션이 처리되는 시점에 계산에 의해 발생하는 성능저하를 예방하기 위해, 미리 계산하여 컬럼에 보관
- ❑ 이력 테이블 컬럼 추가: 대량의 이력 데이터 처리시 불특정 일 조회나 최근 값을 조회 할 때 나타날 수 있는 성능저하를 예방하기 위해 기능성 컬럼(최근 값여부, 시작 일자,종 료 일자)을 추가함
- ❑ PK에 의한 컬럼 추가: 복합 의미를 갖는 PK를 단일 속성으로 구성했을 때 발생되며, PK 안에 데이터가 존재하지만 성능 향상을 위해 일반속성으로 포함하는 방법
- ❑ 응용 시스템 오작동을 위한 컬럼 추가: 업무적으로는 의미가 없으나, 데이터 처리시 오류로 인해 원래값으로 복구하길 원하는 경우 이전 데이터를 임시적으로 중복 보관하는 방법

### ✓ 관계 반정규화

- ❑ 중복 관계 추가: 여러 경로를 거쳐 조인이 가능하지만 성능 저하를 예방하기 위해 추가적인 관계를 맺는 방법

# 데이터베이스 용량 설계

## ❖ 데이터베이스 용량 설계

- ✓ 데이터가 저장될 공간을 정의하는 것
- ✓ 데이터베이스 용량을 설계할 때는 테이블에 저장할 데이터 양과 인덱스, 클러스터 등이 차지하는 공간 등을 예측하여 반영
- ✓ 데이터베이스의 용량을 정확히 산정하여 디스크의 저장 공간을 효과적으로 사용하고 확장성 및 가용성을 높여야 함
- ✓ 데이터 접근성을 향상시키는 설계 방법
  - 테이블의 테이블 스페이스와 인덱스의 테이블 스페이스를 분리하여 구성
  - 테이블 스페이스와 임시 테이블 스페이스를 분리하여 구성
  - 테이블을 마스터 테이블과 트랜잭션 테이블로 분류

# 데이터베이스 용량 설계

## ❖ 데이터베이스 용량 설계

### ✓ 테이블 종류

- 일반 테이블
  - 현재 사용되는 대부분의 DBMS에서 표준 테이블로 사용되는 테이블
- 클러스터드 인덱스 테이블 (Clustered Index Table)
  - 기본키(Primary Key)나 인덱스 키의 순서에 따라 데이터가 저장되는 테이블
  - 일반적인 인덱스를 사용하는 테이블에 비해 접근 경로가 단축
- Partitioning Table
  - 대용량의 테이블을 작은 논리적 단위인 파티션(Partition)으로 나눈 테이블
  - 대용량의 데이터를 효과적으로 관리할 수 있지만 파티션 키를 잘못 구성하면 성능 저하 등의 역효과를 초래
- 외부 테이블 (External Table)
  - 데이터베이스에서 일반 테이블처럼 이용할 수 있는 외부 파일
  - Data Warehouse에서 ETL(Extraction, Transformation, Loading) 등의 작업에 유용하게 사용
- Temporary Table
  - 트랜잭션이나 세션 별로 데이터를 저장하고 처리할 수 있는 테이블
  - 저장된 데이터는 트랜잭션이 종료되면 삭제

# Transaction

## ❖ 트랜잭션(Transaction)

- ✓ 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위
- ✓ 한꺼번에 모두 수행되어야 할 일련의 연산
- ✓ 성질(ACID)
  - Atomicity(원자성): 트랜잭션의 연산은 데이터베이스에 모두 반영 되도록 완료(Commit)되든지 아니면 전혀 반영 되지 않도록 복구(Rollback)되어야 함 – All or Nothing
  - Consistency(일관성): 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환
  - Isolation(격리성, 일관성): 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행 중에 다른 트랜잭션의 연산이 끼어들 수 없음
  - Durability(영속성): 성공적으로 완료된 트랜잭션의 결과는 영구적으로 반영되어야 함
- ✓ 동작
  - Commit: 트랜잭션이 정상적으로 완료되어 작업 내용을 데이터베이스 원본에 반영
  - Rollback: 트랜잭션이 정상적으로 완료되지 못해서 작업 내용을 취소
  - Savepoint: 트랜잭션이 Rollback 할 위치를 생성하는 것
- ✓ CRUD 분석: 프로세스와 테이블 간의 CRUD 매트릭스를 만들어서 트랜잭션을 분석하는 것
- ✓ 트랜잭션 분석: CRUD 매트릭스를 기반으로 테이블에 발생하는 트랜잭션의 양을 분석해서 데이터베이스 용량 산정 및 구조의 최적화를 수행하는 것

# Index

## ❖ 인덱스(Index)

- ✓ 데이터를 빠르게 접근하기 위해 <키 값, 포인터> 쌍으로 구성되는 데이터 구조
- ✓ 인덱스 키의 순서에 따라 데이터가 정렬되어 저장되는 방식인 클러스터드 인덱스(Clustered Index)와 인덱스의 키 값만 정렬되어 있을 뿐 실제 데이터는 정렬되지 않는 방식인 언클러스터드 인덱스(Non-Clustered Index)가 있음
- ✓ Index-Organized Table: 인덱스 안에 테이블 데이터를 직접 삽입하여 저장함으로써 주소를 얻는 과정이 생략되어 더욱 빠른 조회가 가능
- ✓ 인덱스 종류
  - 트리 기반 인덱스: 인덱스를 저장하는 블록들이 트리 구조를 이루고 있는 것으로 B+ 트리 인덱스를 주로 활용함
  - 비트맵 인덱스: 인덱스 컬럼의 데이터를 Bit 값인 0 또는 1로 변환하여 인덱스 키로 사용하는 방법
  - 함수 기반 인덱스: 컬럼의 값 대신 컬럼에 특정 함수(Function)나 수식(Expression)을 적용하여 산출된 값을 사용하는 것으로 B+ 트리 인덱스 또는 비트맵 인덱스를 생성
  - 비트맵 조인 인덱스: 다수의 조인된 객체로 구성된 인덱스로, 단일 객체로 구성된 일반적인 인덱스와 액세스 방법이 다름
  - 도메인 인덱스: 개발자가 필요한 인덱스를 직접 만들어 사용하는 것으로 확장형 인덱스(Extensible Index)

# Index

## ❖ 인덱스의 장점

- ✓ 검색 속도가 빨라짐
- ✓ 시스템에 걸리는 부하를 줄여서 시스템 전체 성능을 향상시킴

## ❖ 인덱스의 단점

- ✓ 오라클에서의 인덱스의 내부 구조는 B\* 트리 형식으로 구성
- ✓ 컬럼에 인덱스를 설정하면 이를 위한 B\* 트리도 생성되어야 하기 때문에 인덱스를 생성하기 위한 시간도 필요하고 인덱스를 위한 추가적인 공간이 필요
- ✓ 인덱스가 생성된 후에 새로운 행을 추가하거나 삭제할 경우 인덱스로 사용된 컬럼 값도 함께 변경되는 경우가 발생
- ✓ 인덱스로 사용된 컬럼 값이 변경되는 이를 위한 내부 구조(B\* 트리) 역시 함께 수정
- ✓ 이 작업은 오라클 서버에 의해 자동으로 일어나는데 그렇기 때문에 인덱스가 없는 경우 보다 인덱스가 있는 경우에 DML 작업이 훨씬 무거워짐





# Index

## ❖ 클러스터 기반 인덱스

✓ SQL Server의 인덱스 종류는 저장 구조에 따라 클러스터형, 비클러스터형 인덱스로 구분

✓ 클러스터 형 인덱스의 중요한 특징

### □ 인덱스 리프 페이지가 곧 데이터 페이지

- 테이블 탐색에 필요한 레코드 식별자가 리프 페이지에 없음
- 클러스터형 인덱스의 리프 페이지를 탐색하면 해당 테이블의 모든 칼럼 값을 곧바로 얻을 수 있음
- 클러스터형 인덱스를 사전에 비유
- 예를 들면, 영한사전은 알파벳 순으로 정렬되어 있으며 각 단어 바로 옆에 한글 설명이 붙어있음
- 책 끝 부분에 있는 찾아보기(색인)가 페이지 번호만 알려주는 것과 비교하면 그 차이를 알 수 있음

### □ 페이지를 모든 로우(데이터)는 인덱스 키 칼럼 순으로 물리적으로 정렬되어 저장

- 테이블 로우는 물리적으로 한 가지 순서로만 정렬될 수 있다. 그러므로 클러스터형 인덱스 테이블당 한 개만 생성 할 수 있다
- 전화번호부 한 권을 상호와 인명으로 동시에 정렬할 수 없는 것과 마찬가지로

# View

## ❖ 뷰(View)

- ✓ 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 가상 테이블
- ✓ 저장장치 내에 물리적으로 존재하지 않지만 사용자는 존재하는 테이블 처럼 사용
- ✓ CREATE문으로 정의하고 DROP문으로 제거하지만 ALTER를 이용해서 구조 변경하는 것은 안됨
- ✓ 데이터의 논리적 독립성을 제공
- ✓ 보안을 향상 시킬 수 있고 데이터베이스 성능을 향상 시킬 수 있다.

# 분산 데이터베이스

## ❖ 분산 데이터베이스

- ✓ 여러 곳으로 분산되어 있는 데이터베이스를 하나의 가상 시스템으로 사용할 수 있도록 한 데이터베이스
- ✓ 논리적으로 동일한 시스템에 속하나 네트워크를 통해 물리적으로 분산되어 있는 데이터들의 모임
- ✓ 데이터베이스를 연결하는 빠른 네트워크환경을 이용하여 데이터베이스를 여러 지역 및 노드로 위치시켜 사용성/성능을 극대화시킨 데이터베이스
- ✓ 데이터 입/출력 부하를 감소시켜 성능을 향상시키고 확장성 및 가용성을 높일 수 있지만 비용이 증가하고 보안이 취약해 질 수 있음

## ❖ 분산 데이터베이스의 투명성(Transparency)

- ✓ 분할 투명성(단편화) : 하나의 논리적 Relation이 여러 단편으로 분할되어 각 단편의 사본이 여러 사이트에 저장
- ✓ 위치 투명성 : 사용하려는 데이터의 저장 장소를 명시할 필요가 없음. 위치정보는 System Catalog에 유지되어야 함
- ✓ 지역 사상 투명성: 지역DBMS와 물리적 DB사이의 Mapping 보장 - 각 지역 시스템 이름과 무관한 이름 사용가능
- ✓ 중복 투명성 : DB객체가 여러 사이트에 중복되어 있는지 알 필요가 없는 성질
- ✓ 장애 투명성 : 구성요소(DBMS, Computer)의 장애에 무관한 트랜잭션의 원자성 유지
- ✓ 병행 투명성 : 다수 트랜잭션 동시 수행 시 결과의 일관성 유지, TimeStamp, 분산 2단계 Locking을 이용하여 구현

# 파일 시스템

## ❖ 데이터 저장을 위한 파일 시스템

- ✓ DAM(Direct Access Method) : 해싱을 이용하여 직접적으로 액세스할 수 있는 방식
- ✓ SAM(Sequential Access Method) : 순차적으로 액세스할 수 있는 방식
- ✓ ISAM(Index Sequential Access Method) : 인덱스를 이용하여 데이터를 순차적 또는 랜덤적으로 액세스할 수 있는 방식으로 Index의 주소는 Hardware적인 방법으로 처리하는 정적 Index 방법을 사용
- ✓ VSAM(Virtual Sequential Access Method) : VSAM 파일은 Index내의 주소를 소프트웨어적으로 처리하는 동적 Index 방법을 사용한다는 점에서 ISAM 파일과 다름
- ✓ NFS(Network File System)
  - 1984년에 썬 마이크로시스템즈가 개발한 프로토콜
  - 클라이언트 컴퓨터의 사용자가 네트워크 상의 파일을 직접 연결된 스토리지에 접근하는 방식과 비슷한 방식으로 접근

# 데이터베이스 이중화

## ❖ 데이터베이스 이중화

✓ 데이터베이스 이중화는 동일한 데이터베이스를 복제하여 관리하는 것

### ✓ 분류

- Eager 기법: 트랜잭션 수행 중 데이터 변경이 발생하면 이중화 된 모든 데이터베이스에 즉시 전달하여 변경 내용이 즉시 반영되도록 하는 기법
- Lazy 기법: 트랜잭션의 수행이 종료되면 변경 사실을 새로운 트랜잭션에 작성하여 각 데이터베이스에 전달하는 기법으로 각각의 데이터베이스는 새로운 트랜잭션을 수행하는 것으로 간주

### ✓ 구성 방법

- 활동 - 대기 방법: 하나의 데이터베이스가 활성 상태로 서비스 중이면 나머지 데이터베이스는 대기하고 있다가 서비스 중인 데이터베이스에 장애가 발생하면 대기 중인 데이터베이스를 사용하는 방식
- 활동 - 활동 방법: 모든 데이터베이스가 활성 상태로 존재

### ✓ 클러스터링

- 두대 이상의 서버를 하나의 서버처럼 운영하는 기술
- 고가용성 클러스터링과 병렬 처리 클러스터링으로 구분

✓ RTO(Recovery Time Objective): 장애가 발생한 시점부터 복구되어 가동될 때 까지 소요 시간

✓ RPO(Recovery Point Objective): 장애가 발생한 이후 데이터를 복구할 수 있는 기준점

# 데이터베이스 보안

## ❖ 데이터베이스 보안

- ✓ 데이터베이스의 일부 또는 전체에 대하여 접근을 못하게 하거나 권한이 없는 사용자가 액세스하는 것을 금지하기 위한 기술
- ✓ 암호화: 지정한 수신자 이외에는 내용을 알 수 없도록 하는 방법
  - ❑ Encrytion: 평문을 암호문으로 변경하는 것
  - ❑ Decryption: 암호문을 평문으로 변경하는 것
  - ❑ 암호화 방법
    - 개인키 암호화
    - 공개키 암호화

# 데이터베이스 보안

## ❖ 접근 통제

✓ 데이터 저장된 개체와 사용자 사이의 정보 흐름을 제한하는 것

✓ 3요소

□ 접근 통제 정책

□ 접근 통제 메커니즘

□ 접근 통제 보안 모델

✓ 접근 통제 기술

□ 임의 접근 통제

○ 사용자의 신원에 따라 접근 권한을 부여하는 방식

○ 데이터 소유자가 접근 통제 권한을 지님

□ 강제 접근 통제

○ 사용자 와 개체에 따라 접근 권한을 부여하는 방식

○ 시스템이 접근 통제 권한을 지님

□ 역할 기반 접근 통제

○ 사용자의 역할에 따라 접근 권한을 부여하는 방식

○ 중앙 관리자가 접근 통제 권한을 지

# 데이터베이스 보안

## ❖ 접근 통제

### ✓ 접근 통제 정책

- ❑ 신분 기반 정책: 사용자의 신원에 따라 접근 권한을 제한
- ❑ 규칙 기반 정책: 사용자의 권한에 근거하여 접근 권한을 제한
- ❑ 접근 통제 보안 모델: 신분 기반 정책의 변형으로 주체가 맡은 역할에 근거해서 개체의 접근을 제한

### ✓ 접근 통제 메커니즘: 접근 통제를 위한 기술적인 방법으로 접근 통제 목록, 권한 리스트, 보안 등급, 비밀번호, 암호화 등

### ✓ 접근 통제 보안 모델: 보안 정책을 구현하기 위한 정형화된 모델

- ❑ 기밀성 모델: 군사적인 목적으로 개발된 것으로 기밀성 보장이 최우선
- ❑ 무결성 모델: 기밀성 모델에 대해서 불법적인 변경을 방지하기 위해 개발된 모델
- ❑ 접근 통제 모델: 접근 통제 메커니즘을 보안 모델로 발전 시킨 것으로 Matrix를 이용

### ✓ 접근 통제 보안 조건: 접근 통제 메커니즘의 취약점을 보완하기 위하여 접근 통제 정책에 부가하여 적용할 수 있는 조건

- ❑ 값 종속 통제: 개체에 저장된 값에 따라 다르게 접근 통제를 허용
- ❑ 다중 사용자 통제: 지정된 객체에 다수의 사용자가 동시에 접근을 요구하는 경우에 사용
- ❑ 컨텍스트 기반 통제: 특정 시간, 네트워크 주소, 접근 경로, 인증 수준 등에 근거하여 접근을 제어하는 방법

### ✓ 감사 추적: 사용자나 애플리케이션이 접근하여 수행한 모든 활동을 기록하는 것



# 데이터베이스 회복

❖ 회복: 데이터베이스에 장애가 발생했을 때 장애 발생 이전 데이터로 돌아가는 것

✓ 로그 파일

- ❑ 데이터베이스 처리 내용이나 이용 상황 등 상태 변화를 시간의 흐름에 따라 기록한 파일
- ❑ 트랜잭션 시작 시점, rollback 시점, 데이터 입력, 수정, 삭제 시점 등에서 기록

✓ 로그 기반의 회복 기법

- ❑ 지연 갱신(Deferred Update) 회복 기법: 트랜잭션이 부분 완료 상태에 이르기까지 발생한 모든 변경 내용을 로그 파일에만 저장하고 데이터베이스에는 commit이 발생할 때 까지 저장을 지연하는 방법
- ❑ 즉시 갱신(Immediate Update) 회복 기법: 트랜잭션 수행 도중 데이터를 변경하면 변경 정보를 로그 파일에 저장하고 트랜잭션이 부분 완료 되기 전이라도 모든 변경 내용을 즉시 데이터베이스에 반영하는 기법

✓ 기타 회복 기법

- ❑ Shadow Paging: 갱신 이전의 데이터베이스를 일정 크기의 페이지 단위로 구성하여 각 페이지마다 복사본인 그림자 페이지를 별도 보관해놓고 실제 페이지를 대상으로 갱신 작업을 수행하다가 장애가 발생하여 트랜잭션 작업을 Rollback 시킬 때는 갱신 이후의 실제 페이지 부분을 그림자 페이지로 대체하여 회복시키는 기법
- ❑ Check Point: 트랜잭션 실행 중 특정 단계에서 재실행할 수 있도록 갱신 내용이나 시스템에 대한 상황들에 관한 정보와 함께 검사점을 로그에 보관해두고 장애 발생 시 트랜잭션 전체를 철회하지 않고 검사점부터 회복 작업을 수행하여 회복시간을 절약하도록 하는 기법

# 데이터베이스 회복

- ❖ 백업: 데이터 회복(복구)을 위하여 데이터의 복제 본을 만드는 작업
  - ✓ 물리적 백업
    - ❑ 데이터베이스 파일을 백업
    - ❑ 백업 속도가 빠르고 작업이 단순하지만 문제 해결이 어려움
  - ✓ 논리적 백업
    - ❑ 데이터베이스 논리적 개체를 백업
    - ❑ 회복 시 문제 해결이나 원인을 파악하는 것이 수월하지만 백업이나 복구 시 시간이 많이 소모

# 데이터베이스 병행 제어

## ❖ 병행 제어

- ✓ 동시에 실행되는 트랜잭션들이 데이터베이스의 일관성을 파괴하지 않도록 트랜잭션 간의 상호 작용을 제어하는 것
- ✓ 병행 제어 없이 트랜잭션들이 데이터베이스에 동시에 접근하도록 허용할 경우 갱신 분실, 비완료 의존성, 모순성, 연쇄 복귀 등의 문제가 발생
- ✓ 병행 제어 기법의 종류
  - ❑ Locking: Lock을 이용
  - ❑ Time Stamp Ordering: 트랜잭션과 트랜잭션이 읽거나 갱신한 데이터에 대해 트랜잭션이 실행을 시작하기 전에 시간표를 부여하여 부여된 시간에 따라 트랜잭션 작업을 작업을 수행하는 기법
  - ❑ 최적 병행 수행: 병행 수행하고자 하는 대부분의 트랜잭션이 판독 전용 트랜잭션일 경우 트랜잭션 간의 충돌률이 매우 낮아서 병행 제어 기법을 사용하지 않고 실행되어도 이 중의 많은 트랜잭션은 시스템의 상태를 일관성 있게 유지한다는 점을 이용한 기법
  - ❑ 다중 버전 기법: 다중 버전 타임 스탬프 기법이라고도 하는데 갱신될 때 마다 버전을 부여하여 관리

# Storage

- ❖ 많은 양의 데이터를 저장하기 위한 서버와 저장장치를 연결하는 기술
- ❖ 방법
  - ✓ DAS(Direct Attached Storage): 서버와 저장장치를 전용 케이블로 연결하는 방식
  - ✓ NAS(Network Attached Storage): 서버와 저장장치를 네트워크로 연결하는 방식
  - ✓ SAN(Storage Area Network): 서버와 저장장치를 연결하는 전용 네트워크를 별도로 구축하는 방식

# 논리 데이터 모델 -> 물리 데이터 모델

## ❖ 논리 데이터 모델의 물리 데이터 모델로 변환

- ✓ 엔티티(Entity)를 테이블로 변환: 논리 데이터 모델에서 정의된 엔티티를 물리 데이터 모델의 테이블로 변환
- ✓ 슈퍼 타입 기준 테이블 변환: 서브 타입을 슈퍼 타입에 통합하여 하나의 테이블로 변환
- ✓ 서브 타입 기준 테이블 변환: 슈퍼 타입 속성들을 각각의 서브타입에 추가하여 서브타입들을 개별적인 테이블로 변환
- ✓ 개별 타입 기준 테이블 변환: 슈퍼 타입과 서브타입들을 각각의 개별적인 테이블로 변환
- ✓ 속성을 컬럼으로 변환: 논리 데이터 모델에서 정의한 속성을 물리 데이터 모델의 컬럼으로 변환
- ✓ 관계를 외래키로 변환: 논리 데이터 모델에서 정의된 관계는 기본키와 이를 참조하는 외래키로 변환

# 데이터 모델

## ❖ 데이터 모델

### ✓ 논리적 데이터 모델의 품질 검증

- 개체 품질 검증 항목
- 속성 품질 검증 항목
- 관계 품질 검증 항목
- 식별자 품질 검증 항목
- 전반적인 품질 검증 항목



# 데이터 모델

## ❖ 데이터 모델

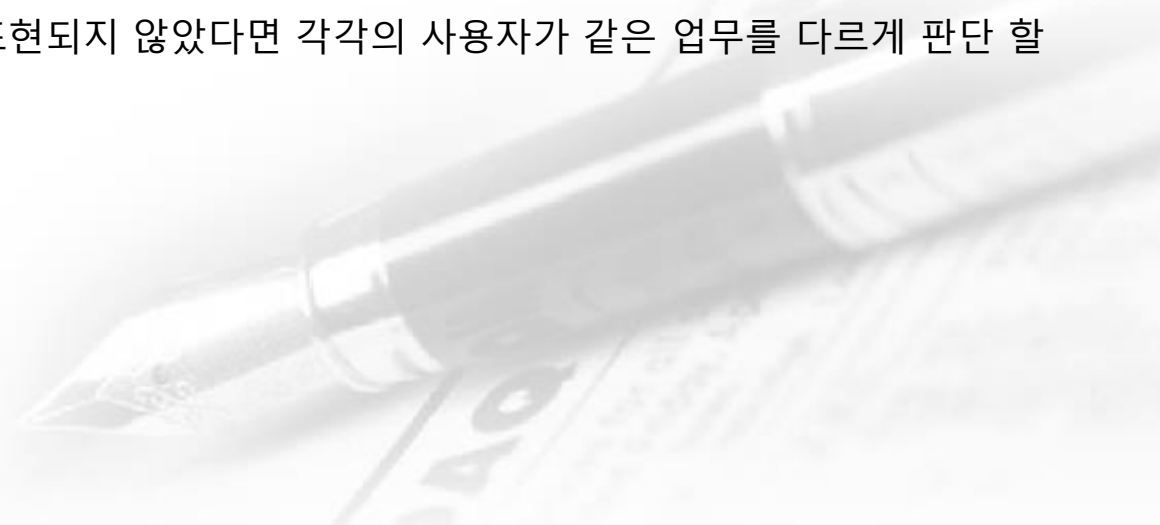
### ✓ 물리 데이터 모델 품질 기준

- 정확성: 데이터 모델이 요구사항이나 업무 규칙, 표기법에 따라 정확하게 표현
- 완전성: 모델이 데이터 모델의 구성 요소를 누락 없이 정의하고 요구사항이나 업무 영역을 누락 없이 반영
- 준거성: 데이터 모델이 데이터 표준, 표준화 규칙, 법적 요건 등을 정확하게 준수
- 최신성: 데이터 모델이 최근의 이슈나 현행 시스템을 반영
- 일관성: 데이터 모델이 표현상의 일관성을 유지
- 활용성: 작성된 모델과 설명을 사용자가 충분히 이해할 수 있고, 업무 변화에 따른 데이터 구조의 변경 이 최소화될 수 있도록 설계되었음을 의미한다.

# 데이터 모델

## ❖ 좋은 데이터 모델의 요소

- ✓ 완전성(Completeness): 업무에서 필요로 하는 모든 데이터를 모델에 정의
- ✓ 중복 배제(Non-Redundancy)
  - ❑ 하나의 데이터베이스에 동일한 사실은 반드시 한번만 기록
  - ❑ 중복 시 문제점
    - 저장공간의 낭비
    - 일관성 유지를 위한 추가 비용 발생
- ❑ 업무 규칙(Business Rules)
  - 업무 규칙(Business Rules)을 데이터 모델링에 표현하고, 모든 사용자가 공유
  - 모든 사용자(개발자, 관리자)가 해당 규칙에 대해 동일하게 판단하고 데이터를 조작할 수 있도록 모델링
  - 업무 규칙이 명확하게 표현되지 않았다면 각각의 사용자가 같은 업무를 다르게 판단 할 수 있음





# 데이터 모델

## ❖ 좋은 데이터 모델의 요소

### ✓ 데이터 재사용(Data Reusability)

#### □ 통합성

- 전사적 관점에서 공통 데이터를 도출하고 이를 전 영역에서 사용하기 적절한 형태로 설계

#### □ 독립성

- 데이터가 어플리케이션에 독립적으로 설계되어야 만 데이터 재 사용성이 향상

#### □ 확장성, 유연성

- 정보시스템은 비즈니스 변화에 대해 최적의 적응을 요구
- 비즈니스 변화에 유연하게 대처하고 확장이 용이한 데이터 설계가 필요
- 확장성, 유연성이 떨어질 경우 작은 업무 변경에도 시스템 기반이 흔들림
- 합리적 균형이 있으면서도 단순하게 분류하는 것

### ✓ 의사소통(Communication)

□ 데이터 모델은 대상 업무를 데이터 관점에서 분석하고 설계하여 나오는 최종 산출물

□ 분석과정에서 도출되는 수많은 업무 규칙들은 최대한 자세하게 표현

□ 모든 관련자들이 데이터 모델을 통해 의사소통을 할 수 있도록 자세하게 기술

### ✓ 통합성(Integration)