데이터 모델링의 이해

모델링의 특징

1. 현실세계를 일정한 형식에 맞추어 표현하는 추상화의 의미를 가짐(추상화)
2. 시스템 구현을 포함한 업무분석 및 업무형상화(시스템 구현만을 위해 진행 X)
3. 복잡한 현실을 제한된 언어나 표기법을 통해 이해하기 쉽게 단순화(단순화)
4. 애매모호함을 배제하고 누구나 이해가 가능하도록 정확하게 현상을 기술(명확성)
5. 데이터베이스를 구축하기 위한 분석/설계 과정

* 데이터모델링은 데이터베이스를 구축하고 업무를 설명하고 분석한다.

데이터 모델링 유의점

1. 중복: 데이터베이스가 여러 장소에 같은 정보를 저장하지 않도록 해야함
2. 비유연성: 데이터의 정의와 사용프로세스를 분리하여 사소한 업무변화에 수시로 변경되지 않도록
3. 비일관성: 사용자가 처리하는 프로세스 혹은 관련된 프로그램과 테이블의 연계성이 높아지면 안됨

데이터 모델링 개념

1. 개념적 데이터 모델링: 전사적 데이터 모델링을 수행할 때 사용, 추상화 수준 높음, 업무 중심적, 포괄적 수준
2. 논리적 데이터 모델링: 시스템으로 구축하고자 하는 업무에 대해 Key, 속성, 관계 등을 정확하게 표현 / 재사용성이 높다.
3. 물리적 데이터 모델링: 실제로 데이터베이스에 이식할 수 있도록, 성능, 저장 등의 물리적인 성격을 고려하여 설계

ERD(Entity-Relationship Model / 피터첸)

1. 작성순서: 엔터티 그리기 => 엔터티 배치 => 엔터티 관계 설정 => 관계명 기술 => 관계 참여도 기술 => 관계 필수여부 기술
2. 관계의 명칭이 중요
3. 배치는 왼쪽 위부터 오른쪽 아래로

데이터베이스 스키마 구조 3단계(3중 스키마)

3중 스키마의 독립성

* 논리적 독립성 : 개념 스키마가 변경되더라도 외부 스키마가 영향을 받지 않는 것
* 물리적 독립성 : 내부 스키마가 변경되더라도 개념 스키마가 영향을 받지 않는 것

1. 외부스키마 : 응용 프로그램이 접근하는 데이터베이스
2. 개념스키마 : 통합관점의 스키마구조를 표현한 것 / 데이터모델링은 통합관점의 뷰를 가지고 있는 개념 스키마를 만들어 가는 과정
3. 내부스키마 : 개발자 관점, 데이터베이스의 물리적 저장 구조

엔터티

* 업무에서 관리해야 하는 데이터 집합
* 저장되고 관리되어야 하는 데이터
* 개념, 사건, 장소 등의 명사

엔터티의 특징

1. 반드시 해당 업무에서 필요하고 관리하고자 하는 정보여야 한다.
2. 유일한 식별자에 의해 식별이 가능해야 한다.
3. 영속적으로 존재하는 두 개 이상의 인스턴스 집합 이어야 한다.
4. 엔터티는 업무 프로세스에 의해 이용되어야 한다.
5. 엔터티는 반드시 속성이 있어야 한다.
6. 엔터티는 다른 엔터티와 최소 한 개 이상의 관계가 있어야 한다.

* 릴레이션 = 테이블 / 인스턴스 = 행의 수

엔터티 종류

물리적 형태의 존재 여부

1. 유형 엔터티 : 업무에서 도출되며 지속적으로 사용되는 엔터티(ex. 고객, 강사, 사원 등)
2. 개념 엔터티 : 유형 엔터티는 물리적 형태가 있지만 개념 엔터티는 없음 (ex.거래소 종목, 코스닥 종목, 생명보험 상품)
3. 사건 엔터티 : 비즈니스 프로세스를 실행하면서 생성되는 엔터티 (ex.주문 체결, 취소주문, 수수료 청구 등)

발생시점에 따른 종류

1. 기본/키엔터티 : 다른 엔터티로부터 주식별자를 상속 X, 자신이 타 엔터티에 부모가 됨 독립적으로 생성되는 엔터티 (ex.고객, 상품, 부서 등)
2. 중심엔터티 : 기본 엔터티와 행위 엔터티 중간 / 기본 엔터티에서 발생, 행위 엔터티에서 실행 (ex. 계좌, 주문, 취소, 체결 등)
3. 행위엔터티 : 2개 이상의 엔터티로부터 발생 (ex. 주문이력, 체결이력 등)

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

속성

* 업무에서 필요한 정보인 엔터티가 가지는 항목
* 관리하고자 하는 의미상 분리되지 않는 최소 데이터
* 인스턴스의 구성 요소

텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진

자동 생성된 설명

속성의 특징

1. 하나의 값만 가짐
2. 주식별자에게 함수적으로 종속(기본키가 변경되면 속성값도 변경된다.)

속성의 종류

분해 여부에 따른 속성의 종류

1. 단일(값) 속성 : 하나의 의미로 구성된 것 (ex.회원 ID, 이름 등)
2. 복합 속성 : 여러 개의 의미가 있는 것 (ex.주소 => 시, 군, 동 등으로 분해)
3. 다중값 속성 : 속성에 여러 개의 값을 가질 수 있는 것 (ex. 상품 리스트) => 다중 값 속성은 엔터티로 분해됨

속성의 특성에 따른 분류

1. 기본속성 : 원래 가지고 있어야 하는 속성 (ex. 회원ID, 이름, 계좌번호, 주문일자 등)
2. 설계속성 : 데이터 모델링 과정에서 발생하는 속성, 유일한 값 부여 (ex. 상품코드, 지점코드 등)
3. 파생속성 : 원래 속성값을 계산하여 저장할 수 있도록 만든 속성 (ex. 합계, 평균 등)

도메인 => 속성이 가질 수 있는 값의 범위 / 엔터티 내에서 속성에 대한 타입과 크기, 제약사항을 지정하는 것 (ex. 성별이라는 속성의 도메인은 ‘남자’와 ‘여자’이다.)

엔터티, 인스턴스, 속성, 속성값

1. 한 개의 엔터티는 두 개 이상의 인스턴스 집합 이어야함
2. 한 개의 엔터티는 두 개 이상의 속성을 가짐
3. 한 개의 속성은 한 개의 속성값을 가짐

속성의 명칭 부여

해당 업무에서 사용하는 이름 / 서술식 이름 X / 약어사용 X / 전체 데이터 모델에서 유일성 확보

관계 => 엔터티 관의 관련성

1. 관계는 존재적 관계와 행위에 의한 관계로 나누어 볼 수 있다.
2. 관계의 표기법은 관계명, 관계차수, 선택사양의 3가지 개념으로 표현한다.
3. 부서와 사원 엔터티 간의 ‘소속’관계는 존재적 관계 사례
4. 주문과 배송 엔터티 간의 ‘배송근거’관계는 행위에 의한 관계 사례

관계의 분류

1. 존재 관계 : 2개의 엔터티가 존재 여부의 관계가 있는 것

* 엔터티 간의 상태를 의미
* 고객이 은행에 회원가입을 하면, 관리점이 할당되고, 그 할당된 관리점에서 고객 관리

텍스트, 도표, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. 행위 관계 : 2개의 엔터티가 어떤 행위에 의한 관련성이 있는 것

* 엔터티 간의 어떤 행위가 있는 것
* 계좌를 사용해서 주문을 발주하는 관계 / 증권회사는 계좌를 개설하고 주문을 발주

텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

관계 차수

* 두 개의 엔터티 간에 관계에 참여하는 수
* Ex. 1명의 고객은 여러 개의 계좌를 개설할 수 있다. => 1대N관계

관계 차수의 종류

1. 1대1 관계

* 완전 1대1 : 하나의 엔터티에 관계되는 엔터티의 관계가 하나인 경우, 반드시 존재
* 선택적 1대1 : 하나의 엔터티에 관계되는 엔터티의 관계가 하나이거나 없을 수도 있음

1. 1대N 관계

* 엔터티에 행이 하나 있을 때 다른 엔터티의 값이 여러 개 있는 관계
* Ex. 고객은 여러 개의 계좌 가질 수 있음

1. M대N 관계

* 2개의 엔터티가 서로 여러 개의 관계를 가짐
* Ex. 1명이 학생이 여러 개의 과목 수강 가능 + 1개의 과목에 여러 학생 수강 가능

1. 필수적 관계와 선택적 관계

* 필수적 관계는 ‘|’로 표현, 선택적 관계는 ‘O’로 표현
* 필수적 관계 : 반드시 하나가 있어야 하는 관계 (ex. 고객이 있어야 계좌개설 가능)
* 선택적 관계 : 없을 수도 있는 관계 (ex. 고객은 있지만 계좌는 없을 수도 있음)

식별관계와 비식별 관계

1. 식별 관계

* 고객과 계좌 엔터티에서 고객은 독립적으로 존재할 수 있는 강한 개체
* 고객 => 독립적 개체, 다른 엔터티와 관계를 가질 때 다른 엔터티에게 기본키 공유, 고객 엔터티의 기본적인 회원ID를 계좌 엔터티에 기본키의 하나로 공유

1. 비식별 관계(점선으로 표현)

* 강한개체의 기본키를 다른 엔터티가 기본키가 아닌 일반 열로 관계를 가지는 것

텍스트, 도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

데이터모델링의 관계

* 데이터모델링 에서는 존재적 관계와 행위에 의한 관계를 구분하는 표기법이 없다.
* UML에서는 연관관계와 의존관계에 대해 다른 표기법을 가지고 표현하게 되어있다.

관계의 표현법

관계명 : 관계의 이름

관계차수 : 1:1, 1:M, M:N

관계선택사양 : 필수관계, 선택관계

엔터티 사이의 관계 체크사항

1. 두 개의 엔터티 사이에 관심 있는 연관규칙이 존재하는가
2. 두 개의 엔터티 사이에 정보의 조합이 발생되는가
3. 업무기술서, 장표에 관계연결을 가능하게 하는 동사(Verb)가 있는가
4. 업무기술서, 장표에 관계연결에 대한 규칙이 서술되어 있는가

엔터티 식별자

식별자 : 엔터티를 대표할 수 있는 유일성을 만족하는 속성(ex. 회원ID, 계좌번호, 주민번호 등)

주식별자의 특징

1. 유일성 : 주식별자에 의해 엔터티내 모든 인스턴스들을 유일하게 구분 함
2. 최소성 : 주식별자를 구성하는 속성의 수는 유일성을 만족하는 최소수가 되어야 함
3. 불변성 : 주식별자가 한 번 특정 엔터티에 지정되면 그 식별자의 값은 변하지 않아야 함
4. 존재성 : 주식별자가 지정되면 반드시 데이터 값이 존재해야 함

식별자의 종류

주식별자(Primary Identifier) / 보조 식별자(AlternateIdentifier) => 엔터티 내에서 대표성을 가지는가에 따라

내부식별자 / 외부식별자(Foreign Identifier) => 엔터티 내에서 스스로 생성되었는지 여부에 따라

단일식별자(Single Identifier) / 복합식별자(Composit Identifier) => 단일 속성으로 식별 되는가

본질식별자 / 인조식별자 => 원래 업무적으로 의미가 있던 식별자 속성을 대체하여 일련번호와 같이 새롭게 만든 식별자(인조식별자)를 구분하기 위해

식별자와 비식별자

1. 식별자 : 강한 연결관계, 종속관계 / 반드시 부모에게 종속
2. 비식별자 : 약간 연결관계, 종속관계 / 자식 주식별자를 독립적으로 구성

데이터 모델과 성능

정규화

* 데이터의 일관성, 최소한의 데이터 중복, 최대한의 데이터 유연성을 위해 테이블을 분해
* 데이터 중복 제거, 독립성 확보
* 정규화를 하면 비즈니스에 변화가 있어도 데이터 모델의 변경을 최소화할 수 있음
* 제1정규화 ~ 제5정규화, 실질적으로는 제3정규화 까지만 함

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명 텍스트, 도표, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

정규화 절차

1. 제1정규화

* 속성의 원자성 확보 / 기본키를 설정

1. 제2정규화

* 기본키가 2개 이상의 속성으로 이루어진 경우, 부분 함수 종속성을 제거한다.

1. 제3정규화

* 기본키를 제외한 칼럼 간의 종속성을 제거한다.
* 즉, 이행 함수 종속성을 제거한다.

1. BCNF

* 기본키를 제외하고 후보키가 있는 경우, 후보키가 기본키를 종속시키면 분해한다.

1. 제4정규화

* 여러 칼럼들이 하나의 칼럼을 종속시키는 경우, 분해하여 다중 값 종속성을 제거한다.

1. 제5정규화

* 조인에 의해서 종속성이 발생하는 경우 분해한다.

함수의 종속성

1. 제1정규화

* 정규화는 함수적 종속성을 근거로 한다.
* 함수적 종속성: X->Y이면 Y는 X에 함수적으로 종속됨.

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* 계좌 테이블 <X>가 <Y>의 칼럼들을 함수적으로 종속하고 있다.
* <X>는 계좌번호 하나만으로는 유일성을 만족하지 못한다고 가정.
* So 계좌번호와 회원ID를 기본키로 잡았다.
* 이처럼 기본키를 잡는 것이 제1정규화

1. 제2정규화

* 부분 함수 종속성은 기본키가 2개 이상의 칼럼으로 이루어진 경우 발생
* 기본키가 하나의 컬럼이면 제2정규화 생략

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* 기본키에 있는 회원 ID가 변경되면 이름이 변경된다. => 회원ID가 이름을 종속하고 있다
* 이러한 경우를 부분 함수 종속성이라고 한다. => 부분 함수 종속성이 발생하면 분해

도표, 라인, 텍스트, 평면도이(가) 표시된 사진

자동 생성된 설명

* 부분 함수 종속성을 제거하면 위와 같다.
* 새로운 테이블이 도출되고, 회원ID가 기본키가 된다.

1. 제3정규화

* 제3정규화는 이행 함수 종속성을 제거한다.
* 이행 함수 종속성: 기본키를 제외하고 칼럼 간 종속성이 발생하는 것
* 제 3정규화는 제1정규화와 제2정규화를 수행한 다음에 수행

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* 위 그림이 이행 함수 종속성의 예시

텍스트, 도표, 라인, 직사각형이(가) 표시된 사진

자동 생성된 설명

* 제3정규화를 수행하면 <관리점> 테이블이 도출되고, 관리점 코드가 기본키가 된다.

1. BCNF

* BCNF는 복수의 후보키가 있고, 후보키들이 복합 속성이어야 하며, 서로 중첩되어야 함

텍스트, 도표, 직사각형, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* 기본키(학번, 과목번호)가 교수를 함수적으로 종속하고 있다.
* 이때 교수가 후보키(최소성과 유일성 만족)이고, 교수가 과목번호를 함수적으로 종속하는 경우 분해가 일어난다.
* 즉 위와 같은 경우 <교수>테이블을 새롭게 만들고, 기본키는 교수로 하고 칼럼은 과목번호가 된다.
* 이러한 작업이 BCNF

정규화 예제

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

* 결과

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

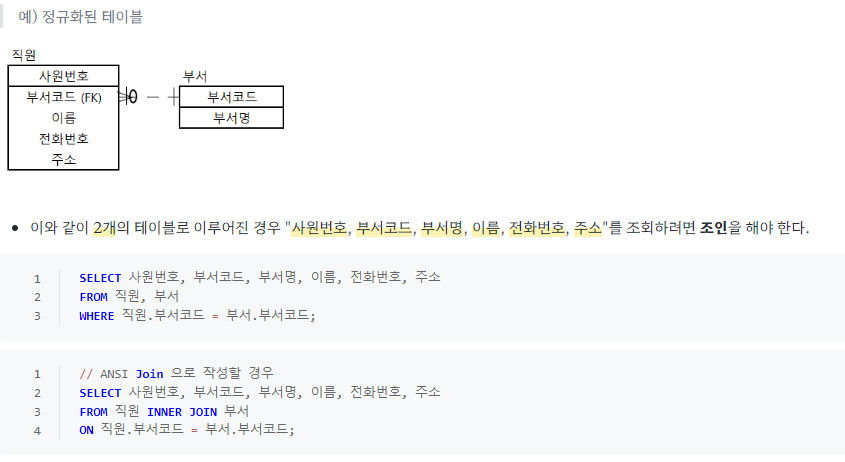
* 결과

0텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

정규화의 문제점

* 정규화는 테이블을 분해해서 데이터 중복을 제거하기 때문에 데이터 모델의 유연성 높임
* 정규화는 데이터 조회시에 조인을 유발하기 때문에 CPU와 메모리를 많이 사용



* <직원>과 <부서> 테이블에서 부서코드가 같은 것을 찾을 때 2중 for문을 써야 함
* 이러한 문제를 해결하기 위해서 반정규화를 하여 하나에 테이블에 저장한다.
* But 반정규화도 문제가 있다.

정규화를 사용한 성능 튜닝

* 조인으로 인하여 성능이 저하되는 문제를 반정규화로 해결할 수 있다.
* But, 반정규화는 데이터를 중복시키기 때문에 또 다른 문제점을 발생시킨다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* 위 그림의 경우 조인이 최소화되면 조회는 빠르게 가능하다.
* 하지만 칼럼이 너무 많아지면 1개 행의 크기가 데이터베이스 관리 시스템의 입출력 단위인 블록의 크기를 넘어서게 된다.
* 그렇게 되면 1개 행을 읽기위해 여러 블록을 읽어야 하고 디스크 입출력up, 성능down

반정규화

* 데이터베이스의 성능 향상을 위해, 데이터 중복을 허용하고 조인을 줄이는 방법
* 반정규화는 조회(SELECT)속도를 향상시키지만, 데이터 모델의 유연성은 낮아진다.

반정규화를 수행하는 경우

* 정규화에 충실하면 종속성, 활용성은 향상되지만 수행속도가 느려지는 경우
* 다량의 범위를 자주 처리해야 하는 경우
* 특정 범위의 데이터만 자주 처리하는 경우
* 요약/집계 정보가 자주 요구되는 경우

반정규화 절차

1. 대상 조사 및 검토

* 데이터 처리 범위, 통계성 등을 확인해서 반정규화 대상을 조사한다.

1. 다른 방법 검토

* 반정규화를 수행하기 전에 다른 방법이 있는지 검토한다.
* Ex. 클러스터링, 뷰, 인덱스 튜닝, 응용프로그램, 파티션 등을 검토

1. 반정규화 수행

* 테이블, 속성, 관계 등을 반정규화 한다.
* 클러스터링
* 클러스터링 인덱스: 인덱스 정보를 저장할 때 물리적으로 정렬해서 저장하는 방법
* 따라서 조회 시에 인접 블록을 연속적으로 읽기 때문에 성능이 향상된다.

반정규화 기법

1. 계산된 칼럼 추가

* 배치 프로그램으로 총판매액, 평균잔고, 계좌평가 등을 미리 계산하고, 특정칼럼에 추가

텍스트, 도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 테이블 수직 분할

* 하나의 테이블을 2개 이상의 테이블로 분할한다.
* 칼럼을 분할하여 새로운 테이블을 만든다.

스크린샷, 라인, 번호, 도표이(가) 표시된 사진

자동 생성된 설명

1. 테이블 수평 분할

* 하나의 테이블에 있는 값을 기준으로 테이블을 분할하는 방법

텍스트, 도표, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

* 파티션 기법
* 데이터베이스에서 파티션을 사용하여 테이블을 분할할 수 있다.
* 파티션을 사용하면 논리적으로는 하나의 테이블이지만 여러 개의 데이터 파일에 분산되어 저장된다.
* 종류
* Range Partition: 데이터 값의 범위를 기준으로 파티션 수행
* List Partition: 특정한 값을 지정하여 파티션 수행
* Hash Partition: 해시 함수를 적용하여 파티션 수행
* Composite Partition: 범위와 해시를 사용하여 파티션 수행
* 장점
* 데이터 조회 시에 엑세스 범위가 줄어들기 때문에 성능 향상
* 데이터가 분할되어 있기 때문에 I/O(Input/Output)의 성능이 향상
* 각 파티션을 독립적으로 백업 및 복구가 가능

1. 테이블 병합

* 1대1 관계의 테이블을 하나의 테이블로 병합해서 성능을 향상
* 1대N 관계의 테이블을 병합하여 성능을 향상
  + 하지만 많은 양의 데이터 중복이 발생
* 슈퍼타입과 서브타입 관계가 발생하면 테이블을 통합하여 성능을 향상
* Ex. <고객>엔터티는 <개인고객>과 <법인고객>으로 분류된다.
  + <고객>: 슈퍼타입(부모)
  + <개인고객>, <법인고객>: 서브타입(자식)
* 관계 종류
  + 베타적 관계
  + <고객>이 <개인고객>이거나 <법인고객>인 경우
  + 포괄적 관계
  + <고객>이 <개인고객>이 될 수도 있고 <법인고객>일 수도 있는 경우

도표, 스케치, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

* 슈퍼타입 및 서브타입 변환 방법
* OneToOne Type
  + 슈퍼타입과 서브타입을 개별 테이블로 도출한다.
  + 테이블의 수가 많아서 조인이 많이 발생하고 관리가 어렵다
* Plus Type
  + 슈퍼타입과 서브타입 테이블로 도출한다.
  + 조인이 발생하고 관리가 어렵다.
* Single Type
  + 슈퍼타입과 서브타입을 하나의 테이블로 도출한다.
  + 조인 성능이 좋고 관리가 편리하지만, 입출력 성능이 나쁘다.

분산 데이터베이스

* 중앙 집중형 데이터베이스: 데이터베이스 시스템 구축 시, 1대의 물리적 시스템에 데이터베이스 관리 시스템을 설치하고 여러 명의 사용자가 데이터베이스 관리 시스템에 접속하여 데이터베이스를 사용하는 구조
* 분산 데이터베이스: 물리적으로 떨어진 데이터베이스에 네트워크로 연결하여 단일 데이터베이스 이미지를 보여주고, 분산된 작업 처리를 수행하는 데이터베이스
  + 분산 데이터베이스를 사용하는 고객은 시스템이 네트워크로 분산되어 있는지의 여부를 인식하지 못하면서, 자신만의 데이터베이스를 사용하는 것처럼 사용 가능
  + 이처럼 데이터베이스는 투명성을 제공해야 한다.

분산 데이터베이스의 투명성 종류

* 분할 투명성
* 고객은 하나의 논리적 릴레이션이 여러 단편으로 분할되어 각 단편의 사본이 여러 시스템에 저장되어 있음을 인식할 필요가 없다.
* 위치 투명성
* 고객이 사용하려는 데이터의 저장 장소를 명시할 필요가 없다.
* 고객은 데이터가 어느 위치에 있더라도 동일한 명령을 사용하여 데이터에 접근할 수 있어야 한다.
* 지역 사상 투명성
* 지역 DBMS에서 물적 데이터베이스 사이의 사상이 보장됨에 따라 각 지역 시스템 이름과 무관한 이름이 사용 가능하다.
* 중복 투명성
* 데이터베이스 객체가 여러 시스템에 중복되어 존재함에도 고객과는 무관하게 데이터의 일관성이 유지된다.
* 장애 투명성
* 데이터베이스가 분산되어 있는 각 지역의 시스템이나 통신망에 이상이 발생해도, 데이터 무결성은 보장된다.
* 병행 투명성
* 여러 고객의 응용 프로그램이 동시에 분산 데이터베이스에 대한 트랜잭션을 수행하는 경우에도 결과에 이상이 없다.

분산 데이터베이스 설계 방식

1. 상향식 설계 방식

* 지역 스키마 작성 후 향후 전역 스키마를 작성하여 분산 데이터베이스를 구축한다.

1. 하향식 설계 방식

* 전역 스키마 작성 후 해당 지역 사상 스키마를 작성하여 분산 데이터베이스를 구축한다.

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* 분산 데이터베이스를 구축할 경우
  + 하향식 접근 방식
  + 기업 전체의 전사 데이터 모델을 수렴하여 전역 스키마를 생성하고, 각 지역별로 지역 스키마를 생성하여 구축
  + 상향식 접근 방식
  + 지역별로 데이터베이스를 구축한 후에 전역 스키마로 통합하여 구축
* 분산 데이터베이스를 구축하거나 운영할 때, 동일한 데이터베이스 관리 시스템으로 분산 데이터베이스를 구축하는 것은 크게 어렵지 않다.
* But, 기업에 여러 종류의 데이터베이스 관리 시스템이 있으면 이기종 데이터베이스 관리 시스템으로 연동해야 한다.
* 이기종 데이터베이스 시스템으로 연동하기 위해서는 데이터베이스 미들웨어를 사용해야 한다.
* 분산 데이터베이스의 장점과 단점
* 장점
  + 데이터베이스의 신뢰성과 가용성이 높다
  + 분산 데이터베이스가 병렬 처리를 수행하기 때문에 빠른 응답이 가능하다.
  + 분산 데이터베이스를 추가하여 시스템 용량 확장이 쉽다
* 단점
  + 데이터베이스가 여러 네트워크를 통해서 분리되어 있기 때문에 관리와 통제가 어렵다.
  + 보안관리가 어렵다.
  + 데이터 무결성 관리가 어렵다.
  + 데이터베이스 설계가 복잡하다.