Chapter01. 데이터베이스 기본

01. 데이터베이스의 필요성

1.1 데이터와 정보

- 데이터 : 현실 세계에서 단순히 관찰하거나 측정하여 수집한 사실이나 값(.=. 자료)

- 정보 : 데이터를 의사 결정에 유용하게 활용할 수 있도록 처리하여 체계적으로 조직한 결과물

- 정보 처리 : 데이터에서 정보를 추출하는 과정 또는 방법 -> 데이터를 상황에 맞게 분석하거나 해석하여 데이터 간의 의미 관계를 파악하는 것

1.2 정보 시스템과 데이터베이스

- 정보 시스템 : 조직 운영에 필요한 데이터를 수집하여 저장해두었다가 의사 결정이 필요할 때 유용한 정보를 만들어 주는 수단

02. 데이터베이스의 정의와 특성

2.1 데이터베이스의 정의

- 데이터베이스 : 특정 조직의 여러 사용자가 **공유**하여 사용할 수 있도록 **통합**해서 **저장**한 **운영** 데이터의 집합

1. 공유 데이터 : 데이터베이스는 특정 조직의 여러 사용자가 함께 소유하고 이용할 수 있어야 하는 공용 데이터이다.
2. 통합 데이터 : 데이터베이스는 똑 같은 데이터가 여러 개 존재하는 데이터 중복성을 허용하지 않는다. 하지만 효율성 때문에 중복을 의도적으로 허용하는 경우도 있으므로, 통합데이터는 데이터의 중복을 최소화하고 통제가 가능한 중복만 허용하는 데이터이다.
3. 저장 데이터 : 데이터베이스의 데이터는 주로 컴퓨터가 처리하므로 데이터베이스를 컴퓨터가 접근하는 매체에 저장해야 한다.
4. 운영 데이터 : 데이터베이스는 조직을 운영하고 조직의 주요 기능을 수행하기 위해 꼭 필요하고, 일시적으로 필요한 것이 아니라 지속적으로 유지해야하는 데이터이다.

2.2 데이터베이스의 특성

1. 실시간 접근 : 실시간 처리에서는 사용자의 개인 특성이나 제공되는 서비스 유형에 따라 허용되는 응답 시간이 다르지만 대개 몇 초를 넘지 않는 시간 내에 데이터를 제공할 수 있어야 한다.
2. 계속 변화 : 데이터베이스는 동적인 특성이 있어 데이터를 계속 삽입, 삭제, 수정하여 현재의 정확한 데이터를 유지해야 한다.
3. 동시 공유 : 동시 공유는 사용자가 서로 다른 데이터를 동시에 사용하는 것뿐만 아니라, 같은 데이터를 동시에 사용하는 것도 모두 지원한다는 의미이다. 그런데 데이터베이스가 동시 공유의 개념을 지원하도록 조직하고 관리하기는 쉽지 않다. 특히 같은 데이터를 동시에 사용할 수 있도록 하려면 더욱 까다로운 처리가 필요하다.
4. 내용 기반 참조 : 데이터베이스는 저장된 주소나 위치가 아닌 데이터의 내용 즉 값으로 참조할 수 있다.

Chapter02. 데이터베이스 관리 시스템

01. 데이터베이스 관리 시스템의 등장 배경

- 파일시스템 : 별도의 구입 비용이 들지 않는다는 장점이 있지만 응용 프로그램마다 파일을 따로 유지하는 특성 때문에 문제점들이 있다.

1. 같은 내용의 데이터가 여러 파일에 중복 저장된다.
2. 응용 프로그램이 데이터 파일에 종속적이다.
3. 데이터 파일에 대한 공시 공유, 보안, 회복 기능이 부족하다.
4. 응용 프로그램 개발이 쉽지 않다.

02. 데이터베이스 관리 시스템의 정의

- 데이터베이스 관리 시스템 : 파일 시스템이 가진 데이터 중복과 데이터 종속 문제를 해결하기 위해 제시된 소프트웨어 / 조직에 필요한 데이터를 데이터베이스에 통합하여 저장하고 이에 대한 관리를 집중적으로 담당 / 응용프로그램을 대신하여 데이터베이스에 존재하는 데이터의 검색, 삽입, 삭제, 수정을 가능하게 하고, 모든 응용 프로그램이 데이터베이스를 공유할 수 있게 한다.

1. 정의 기능 : 데이터베이스 구조를 정의하거나 수정할 수 있다.
2. 조작 기능 : 데이터를 삽입, 삭제, 수정, 검색하는 연산을 할 수 있다.
3. 제어 기능 : 데이터를 항상 정확하고 안전하게 유지할 수 있다.

03. 데이터베이스 관리 시스템의 장단점

3.1 데이터베이스 관리 시스템의 장점

1. 데이터 중복을 통제할 수 있다.
2. 데이터 독립성이 확보된다.
3. 데이터를 동시 공유할 수 있다.
4. 데이터 보안이 향상된다.
5. 데이터 무결성을 유지할 수 있다.
6. 표준화 할 수 있다.
7. 장애 발생 시 회복이 가능하다.
8. 응용 프로그램 개발 비용이 줄어든다.

3.2 데이터베이스 관리 시스템의 단점

1. 비용이 많이 든다.
2. 백업과 회복 방법이 복잡하다.
3. 중앙 집중 관리로 인한 취약점이 존재한다.

04. 데이터베이스 관리 시스템의 발전 과정

4.1 1세대 데이터베이스 관리 시스템 : 네트워크 DBMS와 계층 DBMS

4.2 2세대 데이터베이스 관리 시스템 : 관계 DBMS

4.3 3세대 데이터베이스 관리 시스템 : 객체지향 DBMS와 객체관계 DBMS

Chapter03. 데이터베이스 시스템

01. 데이터베이스 시스템의 정의

- 데이터베이스 시스템 : 데이터베이스에 데이터를 저장하고, 저장된 데이터를 관리하여 조직에 필요한 정보를 생성해주는 시스템 -> 다양한 목적의 정보 처리 시스템을 구축하는데 필요한 핵심 요소

02. 데이터베이스의 구조

2.1 스키마

- 스키마 : 데이터베이스에 저장되는 데이터 구조와 제약조건을 정의한 것

- 인스턴스 : 정의된 스키마에 따라 데이터베이스에 실제로 저장된 값

-> 스키마는 한번 정의 되면 자주 변경되지 않지만, 인스턴스는 계속 변하는 특성이 있다.

2.2 3단계 데이터베이스 구조

2.2.1 3단계 데이터베이스 구조의 개념

- 데이터베이스를 쉽게 이해하고 이용할 수 있도록 하나의 데이터베이스를 관점에 따라 세 단계로 나눈 것

2.2.2 외부단계

- 개별 사용자 관점에서 데이터베이스를 이해하고 표현한다.

- 외부 스키마 : 외부 단계에서 사용자에게 필요한 데이터베이스를 정의한 것

- 데이터베이스 하나에는 외부 스키마가 여러 개 존재할 수 있고, 외부 스키마 하나를 사용 목적이 같은 사용자들이 공유할 수 있다.

2.2.3 개념 단계

- 데이터베이스를 이용하는 사용자들의 관점을 통합하여, 데이터베이스를 조직 전체의 관점에서 이해하고 표현한다.

- 개념 스키마 : 데이터베이스 관리 시스템이나 관리자의 관점에서 모든 사용자에게 필요한 데이터를 통합한 전체 데이터베이스의 논리적 구조를 정의한 것

- 데이터베이스 하나에는 개념 스키마가 하나만 존재하고, 각 사용자는 개념 스키마의 일부분을 사용한다.

2.2.4 내부단계

- 데이터베이스를 디스크나 테이프 같은 저장 장치의 관점에서 이해하고 표현한다.

- 내부 스키마 : 전체 데이터베이스가 저장 장치에 실제로 저장되는 방법을 정의한 것

- 데이터베이스 하나에는 하나의 내부스키마만 존재한다.

2.3 데이터 독립성

- 3단계 데이터베이스 구조의 목적은 데이터 독립성을 실현하기 위해서이다.

2.3.1 논리적 데이터 독립성

- 개념 스키마가 변경되더라도 외부 스키마가 영향을 받지 않는 것이다.

- 외부/개념 사상 : 외부 스키마와 개념 스키마의 대응 관계를 정의한 것(응용 인터페이스)

2.3.2 물리적 데이터 독립성

- 내부스키마가 변경되더라도 개념 스키마가 영향을 받지 않는 것이다.

- 개념/내부 사상 : 개념 스키마와 내부 스키마의 대응 관계를 정의한 것(저장 인터페이스)

2.4 데이터 사전(시스템 카탈로그)

- 데이터베이스에 저장되는 데이터에 관한 정보를 저장하는 곳

03. 데이터베이스 사용자

- 사용자 : 데이터베이스를 이용하기 위해 접근하는 모든 사람

3.1 데이터베이스 관리자

- 데이터베이스 시스템을 운영, 관리

- 데이터베이스를 직접 활용하기보다는 조직 내의 사용자를 위해 데이터베이스 설계 및 구축하고, 제대로 서비스할 수 있도록 데이터베이스에 제어한다.

3.2 최종 사용자

- 데이터를 조작하기 위해 데이터베이스에 접근하는 사람들

3.3 응용 프로그래머

- 데이터 언어를 삽입하여 응용 프로그램을 작성한다.

04. 데이터 언어

- 사용자와 데이터베이스 관리 시스템 간의 통신 수단

4.1 데이터 정의어

- 새로운 데이터베이스를 구축하기 위해 스키마를 정의하거나, 기존 스키마의 정의를 삭제 또는 수정하기 위해 사용하는 데이터 언어

4.2 데이터 조작어

- 사용자가 데이터의 삽입, 삭제, 수정, 검색 등의 처리를 데이터베이스 관리 시스템에 요구하기 위해 사용하는 데이터 언어

- 절차적 데이터 조작어 : 사용자가 어떤 데이터를 원하고 해당 데이터를 얻기 위해서는 어떻게 처리해야 하는 지를 설명한다.

- 비절차적 데이터 조작어 : 사용자가 어떤 데이터를 원하는지만 설명한다. 즉, 해당 데이터를 얻으려면 어떻게 처리해야 하는지는 데이터베이스 관리 시스템에 맡긴다.

4.3 데이터 제어어

- 데이터베이스에 저장된 데이터를 여러 사용자가 무결성과 일관성을 유지하며 문제없이 공유할 수 있도록, 내부적으로 필요한 규칙이나 기법을 정의하는데 사용하는 데이터 언어

- 무결성 : 데이터베이스에 정확하고 유효한 데이터만 유지한다.

- 보안 : 허가받지 않는 사용자가 데이터에 접근한다는 것을 차단하거나, 허가된 사용자가 접근 권한이 있는 데이터에만 접근할 수 있게 한다.

- 회복 : 장애가 발생해도 데이터의 일관성을 유지한다.

- 동시성 : 여러 사용자가 같은 데이터에 동시에 접근하여 처리할 수 있게 한다.

05. 데이터베이스 관리 시스템의 구성

5.1 질의 처리기

- 사용자의 데이터 처리 요구를 해석하여 처리하는 역할을 담당

- DDL 컴파일러 : 데이터 정의어로 작성된 스카마의 정의를 해석한다.

- DML 프리 컴파일러 : 응용 프로그램에 삽입된 데이터 조작어를 추출하여 DML컴파일러에 전달한다.

- DML 컴파일러 : 데이터 조작어로 작성된 데이터의 처리 요구를 분석하여 런타임 데이터베이스 처리기가 이해할 수 있도록 해석한다.

- 런타임 데이터베이스 처리기 : 저장 데이터 관리자를 통해 데이터베이스에 접근하여, DML 컴파일러로부터 전달받은 데이터 처리 요구를 데이터베이스에서 실제로 실행한다.

- 트랜잭션 관리자 : 데이터베이스에 접근하는 과정에서 사용자의 접근 권한이 유효한지를 검사하고, 데이터베이스 무결성을 유지하기 위한 제약조건 위반 여부를 확인한다.

5.2 저장 데이터 관리자

- 디스크에 저장되어 있는 사용자 데이터베이스와 데이터 사전을 관리하고, 여기에 실제로 접근하는 역할을 담당한다.

Chapter04. 데이터 모델링

01. 데이터 모델링과 데이터 모델의 개념

- 데이터 모델링 : 현실 세계에 존재하는 데이터를 컴퓨터 세계의 데이터베이스로 옮기는 변환 과정

- 개념적 모델링 : 중요 데이터를 추출하여 개념 세계로 옮기는 작업

- 논리적 모델링 : 개념 세계의 데이터를 데이터베이스에 저장할 구조를 결정하고 이 구조로 표현하는 작업

- 데이터 모델 : 데이터 모델링을 쉽게 할 수 있도록 도와주는 도구

- 개념적 데이터 모델 : 현실 세계를 개념적 데이터 모델링하여 데이터베이스의 개념적 구조로 표현하는 도구

- 논리적 데이터 모델 : 개념적 구조를 논리적 데이터 모델링하여 데이터베이스의 논리적 구조로 표현하는 도구

- 데이터 구조 : 현실세계를 개념 세계로 추상화했을 때 어떤 요소로 이루어져 있는지 표현하는 개념적 구조 + 데이터를 어떤 모습으로 저장할 것인지를 표현하는 논리적 구조 -> 정적

- 연산 : 데이터 구조에 따라 개념 세계나 컴퓨터 세계에서 실제로 표현된 값들을 처리하는 작업 -> 동적

- 제약 조건 : 구조적 측면의 제약 사항과 연산을 적용하는 경우 허용할 수 있는 의미적 측면의 제약 사항

02. 개체-관계 모델

- 개체-관계 모델 : 현실 세계를 개체와 개체간의 관계를 이용해 개념적 구조로 표현하는 방법

- 개체-관계 다이어그램 : 현실 세계를 개체-관계 모델을 이용해 개념적으로 모델링하여 그림으로 표현한 것

2.1 개체

- entity : 저장할 만한 가치가 있는 중요 데이터를 가지고 있는 사람이나 사물 등 / 다른 개체와 구별되는 이름을 가지고 있고, 각 개체만의 고유한 상태, 즉 속성을 하나 이상 가지고 있다.

- entity type : 개체를 고유한 이름과 속성들로 정의한 것

- entity instance : 개체를 구성하고 있는 속성이 실제 값을 가짐으로써 실체화된 개체

2.2 속성

- attribute : 개체가 가지고 있는 고유의 특성

2.2.1 단일 값 속성과 다중 값 속성

- single-valued attribute : 특정 개체를 구성하는 속성의 값이 하나

- multi-valued attribute : 속성의 값을 여러 개 가짐

2.2.2 단순 속성과 복합 속성

- simple attribute : 의미를 더는 분해할 수 없는 속성 / 의미가 하나

- composite attribute : 의미를 분해할 수 있어 값이 여러 개의 의미를 포함

2.2.3 유도 속성

- derived attribute : 값이 별도로 저장되는 것이 아니라 기존의 다른 속성의 값에서 유도되어 결정되는 속성

- stored attribute : 실제로 값을 저장하고 있는 것

2.2.4 널 속성

- null : 아직 결정되지 않거나 모르는 값 또는 존재하지 않는 값

- null attribute : 널 값이 허용되는 속성

2.2.5 키 속성

- key attribute : 개체를 구성하는 속성들 중에서 특별한 역할을 하는 속성 -> 모든 개체 인스턴스의 키 속성 값이 다르므로 키 속성은 개체 집합에 존재하는 각 개체 인스턴스들을 식별하는데 사용

2.3 관계

- relationship : 개체와 개체가 맺고 있는 의미 있는 연관성

- relationship type : 관계를 여러 개체 사이에서 정의됨

- relationship instance : 실제 속성 값으로 구성되어 있는 특정 개체 인스턴스들 간에 맺어진 실제적인 관계

2.3.1 관계의 유형

- 1:1 관계 : 개체 A의 각 개체 인스턴스가 개체 B의 개체 인스턴스 하나와 관계를 맺을 수 있고, 개체 B의 각 개체 인스턴스도 개체 A의 개체 인스턴스 하나와 관계를 맺음

- 1:N 관계 : 개체 A의 각 개체 인스턴스는 개체 B의 개체 인스턴스 여러 개와 관계를 맺을 수 있지만, 개체 B의 각 개체 인스턴스는 개체 A의 개체 인스턴스 하나와만 관계를 맺음

- N:M 관계 : 개체 A의 각 개체 인스턴스가 개체 B의 개체 인스턴스 여러 개와 관계를 맺을 수 있고, 개체 B의 각 개체 인스턴스도 개체 A의 개체 인스턴스 여러 개와 관계를 맺음

2.3.2 관계의 참여 특성

- 개체 A와 B 사이의 관계에서, 개체 A의 모든 개체 인스턴스가 관계에 반드시 참여해야 된다면개체 A가 관계에 ‘필수적 참여한다’ 또는 ‘전체 참여한다’라고 한다. 그리고 개체 A의 개체 인스턴스 중 일부만 관계에 참여해도 되면 개체 A가 관계에 ‘선택적 참여한다’ 또는 ‘부분 참여한다’라고 한다.

2.3.3 관계의 종속성

- 개체 B가 독자적으로 존재할 수 없고 다른 개체 A의 존재 여부에 의존적이라면, 개체 B가 개체 A에 종속되어 있다고 한다.

- existence dependence : 개체 B가 개체 A에 종속되면, 이는 개체 A가 존재해야 개체 B가 존재할 수 있고 개체 A가 삭제되면 개체 B도 함께 삭제되어야 함을 의미한다.

- weak entity : 다른 개체의 존재 여부에 의존적인 개체 B

- strong entity : 다른 개체의 존재 여부를 결정하는 개체 A

2.4 E-R 다이어그램

03. 논리적 데이터 모델

3.1 논리적 데이터 모델의 개념과 특성

- 논리적 데이터 모델 : 선택한 데이터베이스 관리 시스템에 따라 사용자 입장에서 E-R 다이어그램으로 표현된 개념적 구조를 데이터베이스에 저장할 형태로 표현한 논리적인 구조

3.2 계층 데이터 모델

- hierarchical data model : 데이터베이스의 논리적 구조가 트리형태 / 두 개체 사이에 관계를 하나만 정의할 수 있어 관계에 이름을 붙여 구별할 필요가 없다.

3.3 네트워크 데이터 모델

- network data model : 데이터베이스의 논리적 구조가 그래프 또는 네트워크 형태 / 두 개체 간의 관계를 여러 개 정의할 수 있어 관계를 이름으로 구별한다.

Chapter05. 관계 데이터 모델

01. 관계 데이터 모델의 개념

1.1 관계 데이터 모델의 기본 용어

- 일반적으로 관계 데이터 모델에서는 하나의 개체에 관한 데이터를 릴레이션 하나에 담아 데이터베이스에 저장한다.

1.1.1 속성

- attribute : 릴레이션의 열 / 각 속성은 서로 다른 이름을 이용해 구별한다.

1.1.2 투플

- tuple : 릴레이션의 행

1.1.3 도메인

- domain : 속성 하나가 가질 수 있는 모든 값의 집합

1.1.4 널 값

- null : 릴레이션에 있는 특정 투플의 속성값을 모르거나, 적합한 값이 없는 경우

1.1.5 차수

- degree : 하나의 릴레이션에서 속성의 전체 개수

1.1.6 카디널리티

- cardinality : 하나의 릴레이션에서 투플의 전체 개수

1.2 릴레이션과 데이터베이스의 구성

- 관계 데이터 모델에서 릴레이션은 릴레이션 스키마와 릴레이션 인스턴스로 구성되어 있다.

1.2.1 릴레이션 스키마

- relation schema : 릴레이션의 이름과 릴레이션에 포함된 모든 속성의 이름으로 정의하는 릴레이션의 논리적 구조

1.2.2 릴레이션 인스턴스

- relation instance : 어느 한 시점에 릴레이션에 존재하는 투플들의 집합 / 릴레이션 인스턴스에 포함된 투플은 릴레이션 스키마에서 정의하는 각 속성에 대응하는 실제 값으로 구성되어 있다.

1.2.3. 데이터베이스 스키마와 데이터베이스 인스턴스

- 데이터베이스 스키마 : 데이터베이스를 구성하는 릴레이션들의 스키마를 모아놓은 것

- 데이터베이스 인스턴스 : 데이터베이스를 구성하는 모든 릴레이션의 인스턴스를 모아놓은 것

1.3 릴레이션의 특성

1. 투플의 유일성 : 하나의 릴레이션에는 동일한 투플이 존재할 수 없다.
2. 투플의 무순서 : 하나의 릴레이션에서 투플 사이의 순서는 무의미하다.
3. 속성의 무순서 : 하나의 릴레이션에서 속성 사이의 순서는 무의미하다.
4. 속성의 원자성 : 속성 값으로 원자 값만 사용할 수 있다.

1.4 키의 종류

1.4.1 수퍼키

- super key : 유일성의 특성을 만족하는 속성 또는 속성들의 집합

- uniqueness : 키가 갖추어야 하는 기본 특성으로, 하나의 릴레이션에서 키로 지정된 속성의 값은 투플마다 달라야 한다는 의미

1.4.2 후보키

- candidate key : 유일성과 최소성을 만족하는 속성 또는 속성들의 집합

- minimality : 키를 구성하고 있는 여러 속성 중에서 하나라도 없으면 투플을 유일하게 구별할 수 없는, 꼭 필요한 최소한의 속성들로만 키를 구성하는 특성

1.4.3 기본키

- primary key : 후보키 중에서 기본적으로 사용하기 위해 선택한 키

- 널 값을 가질 수 있는 속성이 포함된 후보키는 기본키로 부적합 하다.

- 값이 자주 변경될 수 있는 속성이 포함된 후보키는 기본키로 부적합하다.

- 단순한 후보키를 기본키로 선택한다.

1.4.4 대체키

- alternate key : 기본키로 선택되지 못한 후보키

1.4.5 외래키

- foreign key : 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합

02. 관계 데이터 모델의 제약

- 무결성 : 데이터를 정확하고 유효하게 유지하는 것

- 무결성 제약조건의 목적 : 데이터베이스에 저장된 데이터의 무결성을 보장하고, 데이터베이스의 상태를 일관되게 유지하는 것

- integrity constraint : 어느 시점에 데이터베이스에 저장된 데이터를 의미하는 데이터베이스 상태 또는 데이터베이스 인스턴스가 항상 지켜야 하는 중요한 규칙

2.1 개체 무결성 제약조건

- entity integrity constraint : 기본키를 구성하는 모든 속성은 널 값을 가질 수 없다.

2.2 참조 무결성 제약조건

- referential integrity constraint : 외래키는 참조할 수 없는 값을 가질 수 없다.

Chapter06. 관계 데이터 연산

01. 관계 데이터 연산의 개념

- 관계 데이터 모델에서 연산은 원하는 데이터를 얻기 위해 릴레이션에 필요한 처리 요구를 수행하는 것으로, 데이터베이스 시스템의 구성 요소 중 데이터 언어의 역할을 한다.

2.1 관계 대수의 개념과 연산자

- 관계 대수 : 원하는 결과를 얻기 위해 릴레이션을 처리하는 과정으로 순서대로 기술하는 언어

2.2 일반 집합 연산자

- 일반 집합 연산자는 연산을 위해 피연산자가 두 개 필요하다.

- 합집합, 교집합, 차집합은 피연산자인 두 개의 릴레이션이 합병 가능해야 한다.

1. 두 릴레이션의 차수가 같다.(속성 개수가 같다.)
2. 두 개의 릴레이션에서 서로 대응되는 속성의 도메인이 같다. 단, 도메인이 같으면 속성의 이름은 달라도 된다.

2.2.1 합집합

- 릴레이션 R과 S의 합집합을 반환

2.2.2 교집합

- 릴레이션 R과 S의 교집합을 반환

2.2.3 차집합

- 릴레이션 R과 S의 차집합을 반환

2.2.4 카티션 프로덕트

- 릴레이션 R의 각 투플과 릴레이션 S의 각 투플을 모두 연결하여 만들어진 새로운 투플을 반환

2.3 순수 관계 연산자

2.3.1 셀렉트

- 릴레이션 R에서 조건을 만족하는 투플들을 반환

2.3.2 프로젝트

- 릴레이션 R에서 주어진 속성들의 값으로만 구성된 투플들을 반환

2.3.3. 조인

- 공통 속성을 이용해 릴레이션 R과 S의 투플들을 연결하여 만들어진 새로운 투플들을 반환

2.3.4 디비전

- 릴레이션 S의 모든 투플들과 관련이 있는 릴레이션 R의 투플들을 반환

2.4 관계 대수를 이용한 질의 표현

2.5 확장된 관계 대수 연산자

2.5.1 세미조인

- 릴레이션 S의 조인 속성으로만 구성한(프로젝트한) 릴레이션을 릴레이션 R에 자연 조인하는 것

2.5.2 외부조인

- 두 릴레이션에 자연 조인 연산을 수행할 때 조인 속성 값이 같은 투플이 상대 릴레이션에 존재하지 않아 조인 연산에서 제외된 모든 투플을 결과 릴레이션에 포함

03. 관계 해석

- 관계 해석 : 처리를 원하는 데이터가 무엇인지만 기술하는 비절차 언어

- 수학의 프레디킷 해석에 기반을 둔다.

- 투플 관계 해석과 도메인 관계 해석으로 분류한다.

Chapter07. 데이터베이스 언어 SQL

01. SQL의 소개

- SQL : 관계 데이터베이스를 위한 표준 질의어로 많이 사용되는 언어 -> 사용자가 처리를 원하는 데이터가 무엇인지만 제시하고 데이터를 어떻게 처리해야 하는지를 언급할 필요가 없어 비절차적 데이터 언어의 특징을 띤다.

- 데이터 정의어(DDL) : 테이블을 생성하고 변경/제거 하는 기능을 제공

- 데이터 조작어(DML) : 테이블에 새 데이터를 삽입하거나, 테이블에 저장된 데이터를 수정/삭제/검색하는 기능을 제공

- 데이터 제어어(DCL) : 보안을 위해 데이터에 대한 접근 및 사용 권한을 사용자별로 부여하거나 취소하는 기능을 하는 언어

02. SQL을 이용한 데이터 정의

2.1 SQL의 데이터 정의 기능

- 테이블 생성, 생성된 테이블 구조의 변경, 테이블 제거

2.2 테이블의 생성

- CREATE TABLE : 생성할 테이블을 구성하는 속성들의 이름과 데이터 타입 및 제약 사항에 대한 정의, 기본키/대체키/외래키의 정의, 데이터 무결성을 위한 제약조건의 정의 등을 포함

CREATE TABLE 테이블\_이름(

1. 속성\_이름 데이터\_타입 [NOT NULL] [DEFAULT 기본\_값]

-> 테이블을 구성하는 각 속성의 이름과 데이터 타입과 기본적인 제약사항 정의

1. [PRIMARY KEY(속성\_리스트)]

-> 기본키로 테이블에 하나만 존재

1. [UNIQUE(속성\_리스트)]

-> 대체키로 테이블에 여러 개 존재

1. [FOREIGN KEY(속성\_리스트) REFERENCES 테이블\_이름(속성\_리스트)]

[ON DELETE 옵션] [ON UPDATE 옵션]

-> 외래키로 테이블에 여러 개 존재

1. [CONSTRAINT 이름] [CHECK(조건)]

-> 데이터 무결성을 위한 제약조건으로 테이블에 여러 개 존재

2.2.1 속성의 정의

- 테이블을 구성하는 각 속성의 데이터 타입을 선택한 후에는 속성의 널 값 허용 여부와 기본 값 필요 여부를 결정해야 한다. CREATE TABLE 문으로 생성되는 테이블을 구성하는 속성은 기본적으로 널 값이 허용된다.

- 기본키를 구성하는 속성이 아니더라도 값을 꼭 입력해야 된다고 판단되는 속성은 NOT NULL 키워드를 표기한다.

- 속성에 기본 값을 지정하려면 DEFAULT 키워드를 사용한다. 속성에 기본값을 저장해두지 않으면 사용자가 속성에 값을 입력하지 않았을 때 해당 속성에 널 값이 기본으로 저장된다.

2.2.2 키의 정의

- CREATE TABLE 문으로 테이블을 정의할 때는 기본키, 대체키, 외래키를 지정할 수 있다.

- PRIMARY KEY : 기본키가 없어도 테이블을 정의할 수 있지만 각 투플을 식별할 수 있는 기본키는 가능한 한 선택하는 것이 좋다. 모든 테이블에서 기본키는 반드시 하나만 지정할 수 있고, 여러 개의 속성으로 구성할 수도 있다.

- UNIQUE : 대체키는 기본키와 같이 각 투플을 유일하게 식별하는 특성이 있다. 대체키는 기본키와 같이 각 투플을 유일하게 식별하는 특성이 있다. 대체키로 지정된 속성의 값은 테이블에서 중복되면 안 되고 유일성을 가져야 한다. 하지만 기본키로 지정된 속성과 달리 널 값을 가질 수 있다. 대체키는 한 테이블에서 여러 개 지정할 수 있다.

- FOREIGN KEY : 외래키가 어떤 테이블의 무슨 속성을 참조하는지 REFERENCES 키워드 다음에 명확히 제시해야 한다. 이는 참조 무결성 제약조건을 유지하기 위함이다.

1. ON DELETE NO ACTION : 투플을 삭제하지 못하게 함
2. ON DELETE CASCADE : 관련 투플을 함께 삭제함
3. ON DELETE SET NULL : 관련 투플의 외래키 값을 NULL로 변경함
4. ON DELETE SET DEFAULT : 관련 투플을 외래키 값을 미리 지정한 기본 값으로 변경함
5. ON UPDATE NO ACTION : 투플을 변경하지 못하도록 함
6. ON UPDATE CASCADE : 관련 투플에서 외래키 값을 함께 변경함
7. ON UPDATE SET NULL : 관련 투플의 외래키 값을 NULL로 변경함
8. ON UPDATE SET DEFAULTE : 관련 투플의 외래키 값을 미리 지정한 기본 값으로 변경함

2.2.3 데이터 무결성 제약조건의 정의

- CREATE TABLE 문으로 테이블을 정의할 때 CHECK 키워드를 사용해 특정 속성에 대한 제약조건을 지정할 수 있다. 테이블에는 CHECK 키워드로 지정한 제약조건을 만족하는 투플만 존재할 수 있다. 테이블에 새로운 투플을 삽입하거나 기존 투플을 수정할 때도 이 제약조건을 반드시 지켜야 한다.

- CHECK 키워드를 사용해 지정한 제약조건에 CONSTRAINT 키워드와 함께 고유의 이름을 부여할 수도 있다. 제약조건을 여러 개 지정할 때 고유의 이름을 부여하면 테이블이 생성된 이후에 제약조건을 수정하거나 제거할 때 식별하기가 쉽다.

2.2.4 테이블 생성의 예

2.3 테이블의 변경

- 테이블은 ALTER TABLE 문으로 변경할 수 있다.

2.3.1 새로운 속성의 추가

- ALTER TABLE 테이블\_이름

ADD 속성\_이름 데이터\_타입 [NOT NULL] [DEFAULTE 기본\_값];

2.3.2 기존 속성의 삭제

- ALTER TABLE 테이블\_이름 DROP 속성\_이름 CASCADE | RESTRICT;

- ALTER TABLE 문을 작성할 때는 삭제할 속성과 관련된 제약조건이 존재하거나 이 속성을 참조하는 다른 속성이 존재하는 경우에 처리하는 방법을 선택할 수 있다.

- 관련된 제약조건이나 참조하는 다른 속성을 함께 삭제하기 위해 CASCADE를 지정

- 관련된 제약조건이나 참조하는 다른 속성이 존재하면 삭제가 수행되지 않도록 RESTRICT를 지정

2.3.3 새로운 제약조건의 추가

- ALTER TABLE 테이블\_이름 ADD CONSTRAINT 제약조건\_이름 제약조건\_내용;

2.3.4 기존 제약조건의 삭제

- ALTER TABLE 테이블\_이름 DROP CONSTRAINT 제약조건\_이름;

2.4 테이블의 제거

- DROP TABLE 테이블\_이름 CASCADE | RESTRICT;

- DROP TABLE 문을 작성할 때 삭제할 테이블을 참조하는 다른 테이블도 함께 삭제하려면 CASCADE를 지정한다. 반대로 삭제할 테이블을 참조하는 테이블이 있으면 삭제가 수행되지 않도록 하려면 RESTRICT를 지정한다.

03. SQL을 이용한 데이터 조작

3.1 SQL의 데이터 조작 기능

- 데이터 검색, 새로운 데이터 삽입, 데이터 수정, 데이터 삭제

3.2 데이터의 검색

- SELECT : 다양한 검색 유형을 지원하며, 일반 사용자들이 가장 많이 사용

3.2.1 기본 검색

- SELECT [ALL | DISTINCT] 속성\_리스트

FROM 테이블\_리스트;

3.2.3 조건 검색

- SELECT [ALL | DISTINCT] 속성\_리스트

FROM 테이블\_리스트

[WHERE 조건];

- AND : 모든 조건을 만족해야 검색한다.

- OR : 여러 조건 중 한 가지만 만족해도 검색한다.

- NOT : 조건을 만족하지 않는 것만 검색한다.

3.2.4 LIKE를 이용한 검색

- 검색 조건을 정확히 몰라 부분적으로 일치하는 데이터를 검색

- % : 0개 이상의 문자(문자의 내용과 개수는 상관 없음)

- \_ : 한 개의 문자(문자의 내용은 상관 없음)

3.2.5 NULL을 이용한 검색

- 검색 조건에서 특정 속성의 값이 널 값인지를 비교하려면 IS NULL 키워드를 사용한다. 마찬가지로 특정 속성의 값이 널 값이 아닌지를 비교하려면 IS NOT NULL 키워드를 사용한다.

3.2.6 정렬 검색

- SELECT [ALL | DISTINCT] 속성\_리스트

FROM 테이블\_리스트

[WHERE 조건]

[ORDER BY 속성\_리스트 [ASC | DESC]];

- SELECT 문의 검색 결과 테이블은 일반적으로 DBMS가 정한 순서로 출력된다. 결과 테이블의 내용을 사용자가 원하는 순서로 출력하려면 ORDER BY 키워드를 사용한다.

- ORDER BY 키워드와 함께 정렬 기준이 되는 속성을 지정하고, 오름차순 정렬이면 ASC, 내림차순 정렬이면 DESC로 표현한다. 기본 정렬 방법은 오름차순이므로 특별히 지정하지 않으면 오름차순으로 기본 정렬한다.

3.2.7 집계 함수를 이용한 검색

- 특정 속성 값을 통계적으로 계산한 결과를 검색

- 집계 함수는 널인 속성 값은 제외하고 계산

- 집계 함수는 WHERE 절에서는 사용할 수 없고 SELECT 절이나 HAVING 절에서만 사용 가능

- COUNT : 속성 값의 개수 – 모든 데이터

- MAX : 속성 값의 최댓값 – 모든 데이터

- MIN : 속성 값의 최솟값 – 모든 데이터

- SUM : 속성 값의 합계 – 숫자 데이터

- AVG : 속성 값의 평균 – 숫자 데이터

3.2.8 그룹별 검색

- 테이블에서 특정 속성의 값이 같은 투플을 모아 그룹을 만들고, 그룹별로 검색을 하기 위해 GROUP BY 키워드를 사용한다. 그룹에 대한 조건을 추가하려면 GROUP BY 키워드를 HAVING 키워드와 함께 사용하면 된다.

- SELECT [ALL | DISTINCT] 속성\_리스트

FROM 테이블\_리스트

[WHERE 조건]

[GROUP BY 속성\_리스트 [HAVING 조건]]

[ORDER BY 속성\_리스트 [ASC | DESC]];

3.2.9 여러 테이블에 대한 조인 검색

- 조인 검색 : 여러 개의 테이블을 연결하여 데이터를 검색하는 것

- 테이블을 연결하려면, 조인 속성의 이름은 달라도 되지만 도메인은 반드시 같아야 한다.

- 조인 검색을 위한 SQL문은 FROM 절에 검색에 필요한 모든 테이블을 나열하고, WHERE 절에는 조인 속성의 값이 같아야 함을 의미하는 조인 조건을 제시한다.

3.2.10 부속 질의문을 이용한 검색

- sub query : 다른 SELECT 문 안에 내포된 SELECT 문

- main query : 다른 SELECT 문을 포함하는 SELECT 문

- IN : 부속 질의문의 결과 값 중 일치하는 것이 있으면 검색조건이 참

- NOT IN : 부속 질의문의 결과 값 중 일치하는 것이 없으면 검색조건이 참

- EXISTS : 부속 질의문의 결과 값이 하나라도 존재하면 검색조건이 참

- NOT EXISTS : 부속 질의문의 결과 값이 하나도 존재하지 않으면 검색조건이 참

- ALL : 부속 질의문의 결과 값 모두와 비교한 결과가 참이면 검색조건을 만족(비교연산자와 함께 사용)

- ANY 또는 SOME : 부속 질의문의 결과 값 중 하나라도 비교한 결과가 참이면 검색조건을 만족(비교연산자와 함께 사용)

3.3 데이터의 삽입

3.3.1 데이터 직접 삽입

- INSERT

INTO 테이블\_이름[(속성\_리스트)]

VALUES(속성값\_리스트);

- INTO 키워드와 함께 투플을 삽입할 테이블의 이름을 제시한 후, 속성의 이름을 나열하는데 이 나열 순서대로 VALUES 키워드 다음의 속성 값들이 차례로 삽입된다.

- INTO 절의 속성 이름과 VALUES 절의 속성 값은 순서대로 일대일 대응되고 개수도 같아야 한다.

- INTO 절에서 속성 이름의 리스트는 생략할 수 있는데, 생략한 경우에는 테이블을 정의할 때 지정한 속성의 순서대로 VALUES 절의 속성 값이 삽입된다.

- VALUES 절에 나열되는 속성 값은 문자나 날짜 타입의 데이터인 경우에는 작은 따옴표로 묶어야 한다.

3.3.2 부속 질의문을 이용한 데이터 삽입

- INSERT

INTO 테이블\_이름[(속성\_리스트)]

SELECT 문;

3.4 데이터의 수정

- UPDATE 테이블\_이름

SET 속성\_이름1 = 값1, 속성\_이름2 = 값2, …

[WHERE 조건];

- UPDATE 문은 테이블에 저장된 투플에서 특정 속성의 값을 수정한다. 값을 어떻게 수정할 것인지는 SET 키워드 다음에 지정한다.

- WHERE 절에 제시된 조건을 만족하는 투플만 속성 값을 수정하는데, WHERE 절을 생략하면 테이블에 존재하는 모든 투플을 대상으로 SET 절에서 지정한대로 속성 값을 수정한다.

3.5 데이터의 삭제

- DELETE

FROM 테이블\_이름

[WHERE 조건];

- DELETE 문은 WHERE 절에 제시된 조건을 만족하는 투플만 삭제한다. WHERE 절을 생략하면 테이블에 존재하는 모든 투플을 삭제하여 빈 테이블이 된다.

04. 뷰

4.1 뷰의 개념

- view : 다른 테이블을 기반으로 만들어진 가상 테이블 / 기본 테이블을 들여다 볼 수 있는 창 역할 / 뷰를 통해 기본 테이블의 내용을 쉽게 검색할 수는 있지만 기본 테이블의 내용을 변화시키는 작업은 제한적으로 이루어진다.

4.2 뷰의 생성

- CREATE VIEW 뷰\_이름[(속성\_리스트)]

AS SELECT 문

[WITH CHECK OPTION];

- CREATE VIEW 명령어와 함께 새로 생성할 뷰의 이름과, 뷰를 구성하는 속성의 이름을 괄호 안에 나열한다. 그리고 AS 키워드와 함께 기본 테이블에 대한 SELECT 문을 제시한다.

- SELECT 문은 생성하고자 하는 뷰의 정의를 담고 있는데, ORDER BY를 사용할 수 없다는 점만 제외하면 일반 SELECT 문과 동일하다.

- 뷰를 구성하는 속성의 이름 리스트는 생략할 수 있는데, 생략하면 SELECT 절에 나열된 속성의 이름을 뷰에서도 그대로 사용한다.

- WITH CHECK OPTION은 생성한 뷰에 삽입이나 수정 연산을 할 때 SELECT 문에서 제시한 뷰의 정의 조건을 위반하면 수행되지 않도록 하는 제약조건을 제시한다.

4.3 뷰의 활용

- CREATE VIEW 문으로 생성된 뷰는 일반 테이블처럼 원하는 데이터를 검색할 수 있다.

- 뷰가 데이터를 실제로 저장하고 있지 않는 가상 테이블임에도 SELECT 문을 이용해 데이터를 검색할 수 있는 것은, 뷰에 대한 SELECT 문이 내부적으로는 기본 테이블에 대한 SELECT 문으로 변환되어 수행되기 때문이다.

- 기본 테이블의 기본키를 구성하는 속성이 포함되어 있지 않은 뷰는 변경할 수 없다.

- 기본 테이블에 있던 내용이 아니라 집계 함수로 새로 계산된 내용을 포함하고 있는 뷰는 변경할 수 없다.

- DISTINCT 키워드를 포함하여 정의한 뷰는 변경할 수 없다.

- GROUP BY 절을 포함하여 정의한 뷰는 변경할 수 없다.

- 여러 개의 테이블을 조인하여 정의한 뷰는 변경할 수 없는 경우가 많다.

1. 질의문을 좀 더 쉽게 작성할 수 없다.
2. 데이터의 보안 유지에 도움이 된다.
3. 데이터를 좀 더 편리하게 관리할 수 있다.

4.4 뷰의 삭제

- DROP VIEW 뷰\_이름 CASCADE | RESTRICT;

- 뷰를 삭제하더라도 기본 테이블은 영향을 받지 않는다. 삭제할 뷰를 이용해 만들어진 다른 뷰가 존재하면 삭제가 수행도지 않도록 하려면 DROP VIEW 문에서 RESTRICT를 지정한다. 반면, 삭제할 뷰를 이용하는 다른 뷰를 모두 함께 삭제하려면 CASCADE를 지정한다.

05. 삽입 SQL

5.1 삽입 SQL의 개념과 특징

1. 삽입 SQL 문은 프로그램 안에서 일반 명령문이 위치할 수 있는 곳이라면 어디든 삽입할 수 있다.
2. 프로그램 안의 일반 명령문과 구별하기 위해 삽입 SQL 문 앞에 EXEC SQL을 붙인다.
3. 프로그램에 선언된 일반 변수를 삽입 SQL 문에서 사용할 수 있다. 단, SQL 문에서 일반 변수를 사용할 때는 앞에 콜론(:)을 붙여 테이블 이름이나 속성의 이름과 구분한다.

- 커서 : 수행 결과로 반환된 행들을 한 번에 하나씩 가리키는 포인터

5.2 커서가 필요 없는 삽입 SQL

- SQL 문을 실행했을 때 특별히 결과 테이블을 반환하지 않는 CREATE TABLE 문, INSERT 문, DELETE 문, UPDATE 문, 결과로 행 하나만 반환하는 SELECT 문은 커서가 필요 없다.

5.3 커서가 필요한 삽입 SQL

- SELECT 문의 실행 결과로 여러 행이 검색되는 경우에는 한 번에 한 행씩 차례로 접근

- EXEC SQL DECLARE 커서\_이름 CURSOR FOR SELECT 문; -> 선언

- EXEC SQL OPEN 커서\_이름; -> SELECT 문 실행

- EXEC SQL FETCH 커서\_이름 INTO 변수\_리스트; -> 커서 이동

- EXEC SQL CLOSE 커서\_이름; -> 커서 사용 X

Chapter08. 데이터베이스 설계

01. 데이터베이스 설계 단계

- 1단계 : 요구 사항 분석 -> 조직의 구성원들이 데이터베이스를 사용하는 용도를 파악

결과물 : 요구 사항 명세서

- 2단계 : 개념적 설계 -> 개발에 사용할 DBMS에 독립적인 개념적 구조 설계

결과물 : 개념적 스키마(E-R 다이어그램)

- 3단계 : 논리적 설계 -> 개발에 사용할 DBMS에 적합한 논리적 구조 설계

결과물 : 논리적 스키마(릴레이션 스키마)

- 4단계 : 물리적 설계 -> DBMS로 구현이 가능한 물리적 구조 설계

결과물 : 물리적 스키마

- 5단계 : 구현 -> SQL 문을 작성한 후 이를 DBMS에서 실행하여 데이터베이스 생성

02. 요구 사항 분석

- 데이터베이스에 대한 사용자들의 요구 사항을 수집하고 분석하여, 개발할 데이터베이스의 용도를 명확히 파악해 요구 사항 명세서를 작성한다.

- 데이터베이스를 사용할 주요 사용자의 범위부터 결정해야 한다.

- 사용자의 범위가 결정되면 해당 사용자가 조직에서 수행하는 업무를 분석해야 한다.

- 요구 사항을 수집하기 위해서는 사용자들과의 면담, 설문지 배포, 업무 관련 문서의 분석 등과 같은 방법이 주로 사용된다.

03. 개념적 설계

- 요구 사항 분석 단계의 결과물을 개념적 데이터 모델을 이용하여 표현한다.

3.1 개체와 속성 추출

- 개체 : 현실 세계에서 어떤 조직을 운영하는데 꼭 필요한 사람, 사물과 같이 구별되는 모든 것

- 요구 사항 명세서에 있는 명사에 주목

- E-R 다이어그램에서 개체는 사각형으로, 속성은 타원형으로 표현한다. 키 속성은 이름에 밑줄을 그어 표시한다.

3.2 관계 추출

- 관계 : 개체 간의 의미 있는 연관성

- 요구 사항 명세서에 있는 동사에 주목

- E-R 다이어그램에서 관계는 마름모로 표현하고, 개체를 표현하는 사각형과 선으로 연결한다. 1:1, 1:N, N:M 관계는 선 위에 레이블로 표시한다. 필수적으로 참여하는 개체는 개체와 관계를 이중선으로 연결한다.

3.3 E-R 다이어그램 작성

04. 논리적 설계

- DBMS 독립적인 개념적 스키마로부터 사용할 DBMS가 처리할 수 있는 데이터베이스의 논리적 구조를 설계하는 것

4.1 릴레이션 스키마 변환 규칙

4.1.1 규칙1 : 모든 개체는 릴레이션으로 변환한다.

- E-R 다이어그램의 각 개체를 하나의 릴레이션으로 변환한다. 개체의 이름을 릴레이션의 이름으로 하고, 개체가 가진 속성도 릴레이션의 속성으로 그대로 변환한다. 단, 개체가 가지고 있는 속성이 복합 속성인 경우에는 복합 속성을 구성하고 있는 단순 속성만 릴레이션의 속성으로 변환한다. 개체가 가지고 있는 키 속성은 릴레이션의 기본키로 변환한다.

4.1.2 규칙2 : 다대다 관계는 릴레이션으로 변환한다.

- E-R 다이어그램에 있는 다대다 관계를 하나의 릴레이션으로 변환한다. 관계의 이름을 릴레이션의 이름으로 하고, 관계의 속성도 릴레이션의 속성으로 그대로 변환한다. 단, 관계를 맺고 있는 개체가 무엇인지 중요하므로, 관계를 맺고 있는 개체들을 규칙1에 따라 변환한 후 이 릴레이션들의 기본키를 관계 릴레이션의 기본키로 지정한다. 개체를 변환한 릴레이션의 기본키를 외래키로 지정할 때는 가져온 기본키들의 이름이 같으면 하나는 이름을 변경해야 한다.

4.1.3 규칙3: 일대다 관계는 외래키로 표현한다.

- E-R 다이어그램에 있는 일대다 관계는 릴레이션으로 변환하지 않고 외래키로만 표현한다.

- 규칙3-1 : 일반적인 일대다 관계는 외래키로 표현한다. -> 관계를 맺고 있는 개체들을 규칙1에 따라 변환한 릴레이션 중에서, 일대다 관계의 1측 개체 릴레이션의 기본키를 가져와 n측 개체 릴레이션에 포함시키고 외래키로 지정한다.

- 규칙3-2 : 약한 개체가 참여하는 일대다 관계는 외래키를 포함해서 기본키로 지정한다. -> 일반 개체들이 참여하는 일대다 관계처럼 관계를 맺고 있는 개체들을 규칙1에 따라 변환한 릴이션들 중에서, 일대다 관계의 1측 개체 릴레이션의 기본키를 가져와 n측 개체 릴레이션에 포함시키고 외래키로 지정한다.

4.1.4 규칙4 : 일대일 관계를 외래키로 표현한다.

- E-R 다이어그램에 있는 일대일 관계도 일대다 관계처럼 릴레이션으로 변환하지 않고 외래키로만 표현한다.

- 규칙4-1 : 일반적인 일대일 관계는 외래키를 서로 주고 받는다. -> 관계를 맺고 있는 개체들을 규칙1에 따라 변환한 릴레이션들이 서로의 기본키를 주고 받아 이를 외래키로 지정한다.

- 규칙4-2 : 일대일 관계에 필수적으로 참여하는 개체의 릴레이션만 외래키를 받는다. -> 관계에 필수적으로 참여하는 개체에 해당하는 릴레이션이 선택적으로 참여하는 개체에 해당하는 릴레이션의 기본키를 받아 외래키로 지정한다.

- 규칙4-3 : 모든 개체가 일대일 관계에 필수적으로 참여하면 릴레이션 하나로 합친다. -> 관계의 이름을 릴레이션의 이름으로 사용하고, 관계에 참여하는 두 개체의 속성들도 관계 릴레이션에 모두 포함시킨다. 그리고 두 개체에 릴레이션의 키 속성을 조합하여 관계 릴레이션의 기본키로 지정한다.

4.1.5 규칙5 : 다중 값 속성은 릴레이션으로 변환한다.

- E-R 다이어그램에 있는 다중 값 속성은 그 속성을 가지고 있는 개체에 해당하는 릴레이션이 아닌 별도의 릴레이션을 만들어 포함시킨다.

- 새로 만들어진 릴레이션에는 E-R 다이어그램에서 다중 값 속성으로 표현된 속성뿐만 아니라 그 속성을 가지고 있는 개체에 해당하는 릴레이션의 기본키를 가져와 포함시키고 이를 외래키로 지정한다.

4.1.6 기타 고려 사항

- 기본 변환 규칙에서는 다대다 관계만 릴레이션으로 변환하였지만 일대일, 일대다 관계도 릴레이션으로 변환할 수 있다. 특히, 속성이 많은 관계는 관계의 유형에 상관없이 릴레이션으로의 변환을 고려할 수 있다.

4.2 릴레이션 스키마 변환 규칙을 이용한 논리적 설계

- 논리적 설계를 위해 E-R 다이어그램을 릴레이션 스키마로 변환할 때는 변환 규칙을 순서대로 적용하면 된다. 해당되지 않는 규칙은 제외하고 다음으로 넘어간다.

05. 물리적 설계와 구현

- 하드웨어나 운영체제의 특성을 고려하여 필요한 인덱스의 구조나 내부 저장 구조 등에 대한 물리적인 구조를 설계한다. 그리고 마지막으로 DBMS를 이용해 SQL 문을 작성하고 이를 실행시켜서 데이터베이스를 실제로 생성하면 데이터베이스 개발이 완료된다.

Chapter 09. 정규화

01. 정규화의 개념과 이상 현상

1.1 정규화의 개념

- anomaly : 불필요한 데이터 중복으로 인해 발생하는 부작용들 / 함수 종속 관계에 여러 개를 하나의 릴레이션에 표현하는 경우에 주로 발생

1.2 이상현상의 종류

1.2.1 삽입 이상

- insert anomaly : 새 데이터를 삽입하기 위해 불필요한 데이터도 함께 삽입해야 하는 문제

1.2.2 갱신 이상

- update anomaly : 중복 투플 중 일부만 변경하여 데이터가 불일치하게 되는 모순의 문제

1.2.3 삭제이상

- delete anomaly : 투플을 삭제하면 꼭 필요한 데이터까지 함께 삭제되는 데이터 손실의 문제

1.3 정규화의 필요성

- 이상 현상이 발생하는 이유 : 관련이 없는 데이터, 즉 속성들을 하나의 릴레이션에 모아두고 있기 때문이다.

- 정규화 : 이상 현상이 발생하지 않도록, 릴레이션을 관련이 있는 속성들로만 구성하기 위해 릴레이션을 분해하는 과정

- 함수적 종속성 : 정규화 과정에서 고려해야 하는 속성들 간의 관련성

02. 함수 종속

- X가 Y를 함수적으로 결정한다 = Y가 X에 함수적으로 종속되어 있다 : X(결정자)->Y(종속자)

- 현제 시점에 릴레이션에 포함된 속성 값만으로 판단하면 안된다. 릴레이션에서 속성 값은 계속 변화할 수 있기 때문에 속성 자체가 가지고 있는 특성과 의미를 기반으로 판단해야 한다.

- 완전 함수 종속 : 릴레이션에서 속성 집합 Y가 속성 집합 X에 함수적으로 종속되어 있지만, 속성 집합 X의 전체가 아닌 일부분에는 종속되지 않음

- 부분 함수 종속 : 속성 집합 Y가 속성 집합 X의 전체가 아닌 일부분에도 함수적으로 종속됨

- 릴레이션에 존재하는 함수 종속 관계에서는 결정자와 종속자가 같거나, 결정자가 종속자를 포함하는 것처럼 당연한 함수 종속 관계는 고려하지 않는다.

03. 기본 정규형과 정규화 과정

3.1 정규화의 개념과 종류

- normalization : 이상 현상이 발생하지 않는 올바른 릴레이션으로 만들어 나가는 과정

- 목표 : 관련이 없는 함수 종속성은 별개의 릴레이션으로 표현하는 것

- Normal Form : 릴레이션이 정규화된 정도

3.2 제 1정규형

- 1NF : 릴레이션에 속한 모든 속성의 도메인이 원자 값으로만 구성되어 있으면 제 1정규형에 속한다.

- 릴레이션이 제 1정규형에 속하기 위해서는 릴레이션에 속한 모든 속성이 더는 분해되지 않는 원자 값을 가져야 한다.

3.3 제 2정규형

- 2NF : 릴레이션이 제 1정규형에 속하고, 기본키가 아닌 모든 속성이 기본키에 완전 함수 종속되면 제 2정규형에 속한다.

- 제 1정규형에 속하는 릴레이션이 제 2정규형을 만족하게 하려면, 부분 함수 종속을 제거하고 모든 속성이 기본키에 완전 함수 종속되도록 릴레이션을 분해하는 정규화 과정을 거쳐야 한다.

- 분해된 릴레이션들을 자연 조인하여 분해 전의 릴레이션으로 다시 복원할 수 있어야 한다.

- 제 2정규형에 속하더라도 릴레이션에 이상 현상이 발생할 수 있다.

3.4 제 3정규형

- 3NF : 릴레이션이 제 2정규형에 속하고, 기본키가 아닌 모든 속성이 기본키에 이행적 함수 종속이 되지 않으면 제 3정규형에 속한다.

- X->Y, Y->Z => X->Z : 속성 집합 Z가 속성 집합 X에 이행적으로 함수 종속되었다.

- 제 2정규형을 만족하더라도 하나의 릴레이션에 함수 종속 관계가 여러 개 존재하고, 논리적으로 이행적 함수 종속 관계가 유도되면 이상 현상이 발생할 수 있다. 릴레이션에서 이행적 함수 종속을 제거해서, 모든 속성이 기본키에 이행적 함수 종속이 되지 않도록 릴레이션을 분해하는 정규화 과정을 거쳐야 제 3정규형을 만족할 수 있다.

3.5 보이스/코드 정규형

- BCNF : 릴레이션의 함수 종속 관계에서 모든 결정자가 후보키이면 보이스/코드 정규형에 속한다.

3.6 제 4정규형과 제 5정규형

- 4NF : 릴레이션이 보이스/코드 정규형을 만족하면서, 함수 종속이 아닌 다치 종속을 제거해야 만족할 수 있다.

- 5NF : 릴레이션이 제 4정규형을 만족하면서 후보키를 통하지 않는 조인 종속을 제거해야 만족할 수 있다.

3.7 정규화 과정 정리

- 비정규형 릴레이션 -----(속성의 도메인이 원자 값으로만 구성되도록 분해)-----> 제1정규형 릴레이션(모든 속성이 도메인이 원자 값으로만 구성) -----(부분 함수 종속 제거)-----> 제2정규형 릴레이션(모든 속성이 기본키에 완전 함수 종속) -----(이행적 함수 종속 제거)-----> 제3정규형 릴레이션(모든 속성이 기본키에 이행적 함수 종속이 아님) -----(후보키가 아닌 결정자 제거)-----> 보이스/코드 정규형 릴레이션(모든 결정자가 후보키)

Chapter 10. 회복과 병행 제어

01. 트랜잭션

1.1 트랜잭션의 개념

- transaction : 하나의 작업을 수행하기 위해 필요한 데이터베이스의 연산들을 모아놓은 것 -> 데이터베이스에서 논리적인 작업의 단위 / 장애가 발생했을 때 데이터를 복구하는 작업의 단위

1.2 트랜잭션의 특성

1.2.1 원자성

- atomicity : 트랜잭션을 구성하는 연산들이 모두 정상적으로 실행되거나 하나도 실행되지 않아야 한다는 all-or-nothing 방식

1.2.2 일관성

- consistency : 트랜잭션이 성공적으로 수행된 후에도 데이터베이스가 일관성 있는 상태를 유지해야 함

1.2.3 격리성

- isolation : 현재 수행 중인 트랜잭션이 완료될 때까지 트랜잭션이 생성한 중간 연산 결과에 다른 트랜잭션들이 접근할 수 없음

1.2.4 지속성

- durability : 트랜잭션이 성공적으로 완료된 후 데이터베이스에 반영한 수행 결과는 어떠한 경우에도 손실되지 않고 영구적이어야 함

1.2.5 트랜잭션의 특성을 지원하는 DBMS의 기능

- 원자성 : 회복 기능

- 일관성 : 병행 제어 기능

- 격리성 : 병행 제어 기능

- 지속성 : 회복 기능

1.3 트랜잭션의 연산

- commit 연산 : 트랜잭션이 성공적으로 수행되었음을 선언(작업 완료)

- rollback 연산 : 트랜잭션이 수행을 실패했음을 선언(작업 취소)

1.4 트랜잭션의 상태

- 활동 상태 : 트랜잭션이 수행을 시작하여 현재 수행 중인 상태 / 상황에 따라 부분 완료 상태나 실패 상태가 된다.

- 부분 완료 상태 : 트랜잭션의 마지막 연산이 실행된 직후의 상태

- 완료 상태 : 트랜잭션이 성공적으로 완료되어 commit 연산을 실행한 상태

- 실패 상태 : 하드웨어, 소프트웨어의 문제나, 트랜잭션 내부의 오류 등의 이유로 장애가 발생하여 트랜잭션의 수행이 중단된 상태

- 철회 상태 : 트랜잭션의 수행이 실패하여 rollback 연산을 실행한 상태

02. 장애와 회복

- recovery : 장애가 발생했을 때 데이터베이스를 장애가 발생하기 전의 일관된 상태로 복구시키는 것

2.1 장애의 유형

- failure : 시스템이 제대로 동작하지 않는 상태

- 트랜잭션 장애 : 트랜잭션 수행 중 오류가 발생하여 정상적으로 수행을 계속할 수 없는 상태

- 원인 : 트랜잭션의 논리적 오류, 잘못된 데이터 입력, 시스템 자원의 과다 사용 요구, 처리 대상 데이터의 부재 등

- 시스템 장애 : 하드웨어의 결함으로 정상적으로 수행을 계속할 수 없는 상태

- 원인 : 하드웨어 이상으로 메인 메모리에 저장된 정보가 손실되거나 교착 상태가 발생한 경우 등

- 미디어 장애 : 디스크 장치의 결함으로 디스크에 저장된 데이터베이스의 일부 혹은 전체가 손상된 상태

- 원인 : 디스크 헤드의 손상이나 고장 등

2.2 데이터베이스의 저장 연산

- 휘발성 저장 장치(소멸성) : 장애가 발생하면 저장된 데이터가 손실됨

- 비휘발성 저장 장치(비소멸성) : 장애가 발생해도 저장된 데이터가 손실되지 않음 단, 디스크 헤더 손상 같은 저장 장치 자체에 이상이 발생하면 데이터가 손실될 수 있음

- 안전 저장 장치 : 비휘발성 저장 장치를 이용해 여러 개 데이터 복사본을 만드는 방법으로, 어떤 장애가 발생해도 데이터가 손실되지 않고 데이터를 영구적으로 저장할 수 있음

- 디스크와 메인 메모리 간의 데이터 이동 연산

- input(X) : 디스크 블록에 저장되어 있는 데이터 X를 메인 메모리 버퍼 블록으로 이동시키는 연산

- output(X) : 메인 메모리 버퍼 블록에 있는 데이터 X를 디스크 블록으로 이동시키는 연산

- 메인 메모리의 버퍼 블록과 프로그램 변수 간의 데이터 이동 연산

- read(X) : 메인 메모리 버퍼 블록에 저장되어 있는 데이터 X를 프로그램의 변수로 읽어오는 연산

- write(X) : 프로그램의 변수 값을 메인 메모리 버퍼 블록에 있는 데이터 X에 기록하는 연산

2.3 회복 기법

2.3.1 회복을 위한 연산

- 회복을 위해 복사본을 만드는 방법

- dump : 데이터베이스 전체를 다른 저장 장치에 주기적으로 복사하는 방법

- log : 데이터베이스에서 변경 연산이 실행될 때마다 데이터를 변경하기 이전 값과 변경한 이후의 값을 별도의 파일에 기록하는 방법

- 회복 연산

- redo(재실행) : 가장 최근에 저장한 데이터베이스 복사본을 가져온 후 로그를 이용해 복사본이 만들어진 이후에 실행된 모든 변경 연산을 재실행하여 장애가 발생하기 직전의 데이터베이스 상태로 복구

- undo(취소) : 로그를 이용해 지금까지 실행된 모든 변경 연산을 취소하여 데이터베이스를 원래의 상태로 복구

- 로그 레코드의 종류

- <Ti, start> : 트랜잭션 Ti가 수행을 시작했음을 기록

- < Ti, X, old\_value, new\_value> : 트랜잭션 Ti가 데이터 X를 이전 값에서 새로운 값으로 변경하는 연산을 실행했음을 기록

- <Ti, commit> : 트랜잭션 Ti가 성공적으로 완료되었음을 기록

- <Ti, abort> : 트랜잭션 Ti가 철회되었음을 기록

2.3.2 로그 회복 기법

- 즉시 갱신 회복 기법 : 트랜잭션 수행 중에 데이터를 변경한 연산의 결과를 데이터베이스에 즉시 반영

- 지연 갱신 회복 기법 : 트랜잭션이 수행되는 도중 데이터 변경 연산의 결과를 데이터베이스에 즉시 반영하지 않고 로그 파일에만 기록해두었다가, 트랜잭션이 부분 완료된 후에 로그에 기록된 내용을 이용해 데이터베이스에 한번에 반영

2.3.3 검사 시점 회복 기법

- 로그 회복 기법과 같은 방법으로 로그 기록을 이용하되, 일정 시간 간격으로 검사 시점을 만들어둔다.

2.3.4 미디어 회복 기법

- 전체 데이터베이스의 내용을 일정 주기마다 다른 안전한 저장 장치에 복사해두는 덤프를 이용한다.

03. 병행 제어

3.1 병행 수행과 병행 제어

- concurrency : 여러 개의 트랜잭션이 동시에 수행 -> 트랜잭션들이 차례로 번갈아 수행되는 인터리빙 방식으로 진행

- concurrency control : 여러 개의 트랜잭션이 병행 수행되면서 같은 데이터에 접근하여 연산을 실행하더라도, 문제가 발생하지 않고 정확한 수행 결과를 얻을 수 있도록 트랜잭션의 수행을 제어하는 것

3.2 병행 수행의 문제

3.2.1 갱신 분실

- lost update : 하나의 트랜잭션이 수행한 데이터 변경 연산의 결과를 다른 트랜잭션이 덮어써 변경 연산이 무효화되는 것

3.2.2 모순성

- inconsistency : 하나의 트랜잭션이 여러 개의 데이터 변경 연산을 실행할 때 일관성 없는 상태의 데이터베이스에서 데이터를 가져와 연산을 실행함으로써 모순된 결과가 발생하는 것

3.2.3 연쇄 복귀

- cascading rollback : 트랜잭션이 완료되기 전에 장애가 발생하여 rollback 연산을 수행하면, 이 트랜잭션이 장애 발생 전에 변경한 데이터를 가져가 변경 연산을 실행한 또 다른 트랜잭션에도 rollback 연산을 연쇄적으로 실행해야 한다는 것

3.3 트랜잭션 스케줄

- 트랜잭션에 포함되어 있는 연산들을 수행하는 순서

3.3.1 직렬 스케줄

- serial schedule : 인터리빙 방식을 이용하지 않고 각 트랜잭션별로 연산들을 순차적으로 실행시키는 것

3.3.2 비직렬 스케줄

- nonserial schedule : 인터리빙 방식을 이용하여 트랜잭션들을 병행해서 수행시키는 것

3.3.3 직렬 가능 스케줄

- serializable schedule : 직렬 스케줄과 같이 정확한 결과를 생성하는 비직렬 스케줄

3.4 병행 제어 기법

- 여러 트랜잭션을 병행 수행하면서도 정확한 결과를 얻을 수 있는 직렬 가능성을 보장받기 위해 사용

3.4.1 로킹 기법의 개념

- locking : 한 트랜잭션이 먼저 접근한 데이터에 대한 연산을 모두 마칠 때까지, 해당 데이터에 다른 트랜잭션이 접근하지 못하도록 상호 배제하여 직렬 가능성을 보장하는 것

- lock 연산 : 트랜잭션이 사용할 데이터에 대한 독점권을 가지기 위해 사용

- unlock 연산 : 트랜잭션이 데이터에 대한 독점권을 반납하기 위해 사용

- lock 연산

- shared lock : 트랜잭션이 데이터에 대해 공용 lock 연산을 실행하면, 해당 데이터에 read 연산을 실행할 수 있지만 write 연산은 실행할 수 없다. 그리고 해당 데이터에 다른 트랜잭션도 공용 lock 연산을 동시에 실행할 수 있다. (데이터에 대한 사용권을 여러 트랜잭션이 함께 가질 수 있음)

- exclusive lock : 트랜잭션이 데이터에 전용 lock 연산을 실행하면 해당 데이터에 read 연산과 write 연산을 모두 실행할 수 있다. 그러나 해당 데이터에 다른 트랜잭션은 공용이든 전용이든 어떤 lock 연산도 실행할 수 없다. (전용 lock 연산을 실행한 트랜잭션만 해당 데이터에 대한 독점권을 가질 수 있음)

3.4.2 2단계 로킹 규약

- 기본 로킹 규약의 문제를 해결하고 트랜잭션의 직렬 가능성을 보장하기 위해 lock과 unlock연산의 수행 시점에 대한 새로운 규약을 추가한 것

- 확장 단계 : 트랜잭션이 lock 연산만 실행할 수 있고, unlock 연산은 실행할 수 없는 단계

- 축소 단계 : 트랜잭션이 unlock 연산만 실행할 수 있고, lock 연산은 실행할 수 없는 단계

Chapter11. 보안과 관리

01. 보안

- 데이터베이스 보안의 유형

1. 물리적 환경에 대한 보안 : 물리적 손실을 발생시키는 위험으로부터 데이터베이스를 보호
2. 권한 관리를 통한 보안 : 접근이 허락된 사용자만 권한 내에서 데이터베이스를 사용하도록 보호
3. 운영 관리를 통한 보안 : 무결성 제약조건을 정의하고 이 제약조건을 위반하지 않도록 통제

02. 권한 관리

2.1 권한 관리의 기념

- 데이터베이스 관리 시스템은 데이터베이스의 보안을 유지하기 위해, 계정이 발급된 사용자가 로그인에 성공했을 경우에만 데이터베이스에 접근이 가능하도록 하는 접근 제어 기능을 기본으로 제공한다.

2.2 권한의 부여

- GRANT 권한 ON 객체 TO 사용자 [WITH GRANT OPTION];

2.3 권한의 취소

- REVOKE 권한 ON 객체 FROM 사용자 CASCADE | RESTRICT

2.4 역할의 부여와 취소

- role : 여러 사용자에게 동일한 권한들을 부여하고 취소하는 번거로운 작업을 편리하게 수행할 수 있도록 도움을 주는 것

- CREATE ROLE 롤이름; -> 역할 생성

- GRANT 권한 ON 객체 TO 롤이름; -> 역할에 필요한 권한 부여

- GRANT 롤이름 TO 사용자; -> 역할을 사용자에게 부여하는 것

- REVOKE 롤이름 FROM 사용자; -> 사용자에게 부여한 역할의 취소

- DROP ROLE 롤이름; ->역할 제거

Chapter 12. 데이터베이스 응용 기술

01. 객체지향 데이터베이스

- 객체 지향 개념에 기분을 둔 객체지향 데이터 모델을 지원하는 데이터베이스

1.1 객체지향 데이터 모델

- 객체와 객체 식별자, 속성과 메소드, 클래스, 클래스 계층 및 상속, 복합 객체 등을 지원하는 객체지향 개념에 기반을 둔 데이터 모델

1.1.1 객체와 객체 식별자

- object : 현실세계에 존재하는 개체를 추상적으로 표현한 것

- Object Identifier : 각 객체를 유일하게 식별

1.1.2 속성과 메소드

- attribute : 객체의 상태나 특성

- method : 객체에 수행할 수 있는 연산으로 객체의 속성 값을 조작

1.1.3 클래스

- class : 속성과 메소드를 공유하는 유사한 성질의 객체들을 하나로 그룹화한 것

- class instance(=object instance) : 클래스의 구성원

1.1.4 클래스 계층과 상속

- 클래스를 단계적으로 세분화하면 클래스 간의 계층 관계가 발생하여 결과적으로 클래스 계층이 하나 형성된다.

- superclass : 상위에 위치하는 클래스

- subclass : 하위에 위치하는 클래스

1.1.5 복합 객체

- composite object : 시스템에서 기본으로 제공하지 않는 사용자 정의 클래스를 도메인으로 하는 속성을 가진 객체

1.2 객체지향 질의 모델

- 질의 대상은 클래스, 질의 결과는 클래스에 속하는 객체 집합

02. 객체관계 데이터베이스

- 관계 데이터 모델에 객체지향 개념을 적용한 객체관계 데이터 모델을 기반으로 하는 데이터베이스

- 객체지향 개념을 지원하는 표준 SQL을 사용할 수 있다.

03. 분산 데이터베이스 시스템

- 물리적으로 분산된 데이터베이스 시스템을 네트워크로 연결해 사용자가 논리적으로 하나의 중앙 집중식 데이터베이스 시스템처럼 사용할 수 있도록 한 것

3.1 분산 데이터베이스 시스템의 구성

- 분산 처리기 : 물리적으로 분산되어 지역별로 필요한 데이터를 처리

- 분산 데이터베이스 : 물리적으로 분산된 지역 데이터베이스

- 통신 네트워크 : 특정 통신 규약에 따라 데이터를 전송하고 수신

3.2 분산 데이터베이스 시스템의 주요 목표

- 분산 데이터 독립성 : 데이터베이스가 분산되어 있음을 사용자가 인식하지 못하게 하는 것

3.2.1 위치 투명성

- location transparency : 데이터베이스가 지역적으로 분산되어 있지만, 사용자가 접근하려는 데이터의 실제 저장 위치를 알 필요 없이 데이터베이스의 논리적인 이름만으로 데이터에 접근 가능

3.2.2 중복 투명성

- replication transparency : 동일한 데이터가 여러 지역에 중복 저장되더라도 사용자가 중복을 인식하지 못하고, 하나의 데이터베이스 시스템에 데이터가 저장된 것처럼 사용하는 것

- 데이터 중복의 장점 : 한 지역에서 문제가 발생해도 동일한 데이터가 저장되어 있는 다른 지역에서 작업을 계속 수행할 수 있으므로 신뢰성과 가용성이 높아진다. / 동일한 데이터가 저장된 여러 지역에서 병렬 처리를 수행할 수 있어 데이터 처리 성능이 향상된다. / 데이터 처리 요청이 한 지역에 집중되지 않고 여러 지역에 분산되므로 처리 부담을 줄일 수 있다.

- 데이터 중복의 단점 : 동일한 데이터를 중복 저장하므로 저장 공간을 많이 사용한다. / 데이터를 변경하려면 중복 저장된 데이터를 모두 함께 변경해야 하므로 비용이 증가하고, 변경 도중에 문제가 생겨 데이터 불일치가 발생할 수도 있다.

3.2.3 단편화 투명성

- fragmentation : 하나의 릴레이션을 더 작은 조각으로 나누고 각 조각을 별개의 릴레이션으로 처리하는 것

- 단편화 수행 조건

- 완전성 : 전체 릴레이션의 모든 데이터는 어느 한 조각에는 꼭 속해야 한다.

- 회복성 : 단편화된 조각들로부터 원래의 전체 릴레이션을 회복할 수 있어야 한다.

- 분리성 : 전체 릴레이션의 모든 조각을 서로 중복되지 않게 분리해야 한다.

- 수평적 단편화 : 릴레이션을 수평적으로 단편화하는 것 – 릴레이션을 투플(행) 단위로 나눈다.

- 수직적 단편화 : 릴레이션을 수직적으로 단편화하는 것 – 릴레이션을 속성(열) 단위로 나눈다.

- 혼합 단편화 : 수평적 단편화와 수직적 단편화를 모두 사용하여 릴레이션을 나눈다.

3.2.4 병행 투명성

- concurrency transparency : 분산 데이터베이스와 관련된 트랜잭션이 동시에 수행되더라도 결과는 항상 일관성을 유지하는 것

3.2.5 장애 투명성

- failure transparency : 특정 지역에서 문제가 발생하더라도 전체 시스템이 작업을 계속 수행할 수 있는 것

3.3 분산 데이터베이스의 구조

- 전역 개념 스키마 : 분산 데이터베이스에 저장할 모든 데이터 구조와 제약조건을 정의

- 단편화 스키마 : 전역 개념 스키마를 분할하는 방법

- 할당 스키마 : 각 조각 스키마의 인스턴스를 물리적으로 저장해야 되는 지역을 정의

- 지역 스키마 : 지역별로 저장하고 있는 데이터 구조와 제약조건의 정의

3.4 분산 데이터베이스의 질의 처리

- 디스크 접근 횟수뿐만 아니라, 네트워크에서 데이터를 전송하는 비용과 하나의 질의문을 분해하여 여러 지역에서 병렬 처리함으로써 얻는 성능상의 이점도 고려

3.5 분산데이터베이스 시스템의 장단점

3.5.1 분산데이터베이스 시스템의 장점

- 신뢰성과 가용성의 증대, 지역 자치성과 효율성 증대, 확장성 증대

3.5.2 분산데이터베이스 시스템의 단점

- 개발 및 관리 비용이 많이 소모

04. 멀티미디어 데이터베이스 시스템

- 숫자나 문자 데이터와 같은 일반 데이터를 처리하는 기능뿐만 아니라 영상, 음향 및 애니메이션과 같은 멀티미디어 데이터도 효과적으로 저장하고 처리하는 기능도 제공

4.1 멀티미디어 데이터의 특성

- 미디어 : 데이터의 각 타입

- 멀티미디어 데이터 : 여러 미디어의 조합으로 이루어진 데이터

- 멀티미디어 데이터의 유형

- 텍스트 : 문자로 구성된 데이터

- 그래픽 : 수학 공식을 기반으로 제작된 벡터 이미지 데이터

- 이미지 : 정적 이미지나 사진과 같이 픽셀 단위로 표현되는 비트맵 이미지

- 비디오 : 동영상, 애니메이션

- 오디오 : 음성, 소리, 음악

- 멀티미디어 데이터의 특성

- 대용량 데이터

- 검색방법이 복잡한 데이터

- 구조가 복잡한 데이터

4.2 멀티미디어 데이터베이스의 발전 과정

4.2.1 관계 데이터베이스에서의 멀티미디어 데이터 처리

- 관계 데이터베이스에 멀티미디어 데이터를 위한 새로운 데이터 타입을 추가하여 멀티미디어 데이터를 저장하고 처리하는 방법

4.2.2 객체지향 데이터베이스에서의 멀티미디어 데이터 처리

- 다양한 관계의 표현, 데이터 추상화와 캡슐화, 상속 등 멀티미디어 데이터를 처리하는데 필요한 다양한 기능을 제공

4.3 멀티미디어 데이터베이스 관리 시스템의 구성

- 파일 시스템을 이용하는 방식

- 관계 데이터베이스 관리 시스템을 이용하는 방식

- 확장된 관계 데이터베이스 관리 시스템을 이용하는 방식

- 객체 지향 데이터베이스 관리 시스템을 이용하는 방식

4.4 멀티미디어 데이터의 질의

4.4.1 멀티미디어 데이터의 질의 유형

1. 텍스트 질의 : 사용자가 제시한 키워드를 포함하는 문서를 검색하는 질의가 대부분이다.
2. 이미지 질의 : 사용자가 제시한 키워드와 관련 있는 이미지를 검색하는 내용 검색이나, 사용자가 제시한 이미지와 유사한 이미지를 검색하는 유사도 검색 질의가 많다.
3. 비디오 질의 : 비디오는 장면이라고 하는 여러 정지 화면으로 구성되므로, 장면을 대상으로 하는 검색 질의가 많다.
4. 공간 질의 : 주어진 범위 조건에 맞는 특정 위치를 검색하는 질의가 많다.

4.4.2 멀티미디어 데이터의 질의 처리

1. 매칭 기법 : 수학 함수로 저장된 데이터와 질의 조건으로 주어진 데이터 간의 유사도를 수학 함수로 계산하여, 유사도가 높은 데이터를 검색한다.
2. 랭킹 기법 : 검색 결과를 질의 조건과의 관련 정도에 따라 정렬하여, 관련성이 높은 결과부터 제공한다.
3. 필터링 기법 : 질의 조건과의 관련성이 적은 데이터를 단계적으로 제거하여 검색 범위를 줄여가면서 검색한다.
4. 인덱스 기법 : 인덱스 구조를 이용해 질의 조건에 적합한 데이터를 검색한다.

05. 기타 데이터베이스 응용 기술

5.1 웹 데이터베이스

- 웹 서비스의 특성과 데이터베이스 시스템의 데이터 관리 기능을 통합한 것으로 다양한 웹 서비스 분야에서 활용할 수 있다.

5.2 데이터 웨어하우스

- 데이터베이스 시스템에서 의사 결정에 필요한 데이터를 미리 추출하여, 이를 원하는 형태로 변환하고 통합한 읽기 전용의 데이터 저장소

- subject-oriented

- integrated

- time-variant

- nonvolatile

5.3 빅 데이터

5.3.1 빅 데이터의 개념

- 기존의 데이터베이스가 저장하고 관리할 수 있는 범위를 넘어서는 대규모의 다양한 데이터로, 이를 저장 및 관리하는 기술과 가치 있는 정보를 만들기 위해 분석하는 기술까지 포함한다.

5.3.2 빅 데이터의 특징

- Volume : 테라바이트 단위 이상의 대량 데이터

- Velocity : 데이터의 수집 및 분석의 실시간 처리

- Variety : 정형/반정형/비정형 데이터

5.3.3 빅 데이터의 기술

- 저장 기술 : 하둡, NoSQL

- 분석 기술

- 텍스트 마이닝 : 반정형 또는 비정형 텍스트에서 자연어 처리 기술을 기반으로 가치 있는 정보를 추출하고 가공한다.

- 오피니언 마이닝 : SNS, 블로그, 게시판 등에 기록된 사용자들의 의견을 수집하고 분석하여, 제품이나 서비스에 대한 긍정, 부정, 중립 등의 선호도를 추출한다.

- 소셜 네트워크 분석 : 소셜 네트워크의 연결 구조나 강도 등을 바탕으로 소셜 네트워크에서의 영향력, 관심사, 성향, 행동 패턴 등을 추출한다.

- 군집 분석 : 데이터 간의 유사도를 측정한 후 이를 바탕으로 특성이 비슷한 데이터를 합쳐가면서 최종적으로 유사 특성의 데이터 집합을 추출한다.

- 표현 기술 : R 언어