흥달쌤과 함께하는

정보처리기사 실기

마무리 특강 학습교안

1억뷰 N잡

YouTube 흥달쌤 (https://bit.ly/3KtwdLG) E-Mail hungjik@naver.com

네이버 카페 흥달쌤의 IT 이야기 (https://cafe.naver.com/sosozl/)

01 데이터베이스 구축

Section 1. 데이터베이스 개념

1. 데이터베이스 개념

⑴ 데이터베이스

- 특정 조직의 업무를 수행하는 데 필요한 상호 관련된 데이터들의 모임
- 여러 사람들이 공유하고 사용할 목적으로 통합 관리되는 데이터들의 모임

⑵ 데이터베이스의 정의

정의	설명
통합 데이터 (Integrated Data)	- 검색의 효율성을 위해 중복이 최소화된 데이터의 모임
저장 데이터 (Stored Data)	- 컴퓨터가 접근 가능한 저장 매체에 저장된 데이터
운영 데이터 (Operational Data)	- 조직의 목적을 위해 존재 가치가 확실하고 반드시 필요한 데이터
공유 데이터 (Shared Data)	- 여러 응용 프로그램들이 공동으로 사용하는 데이터

⑶ 데이터베이스의 특징

특징	설명
실시간 접근성 (Real Time Accessibility)	- 사용자의 질의에 대하여 즉시 처리하여 응답
계속적인 변화 (Continuous Evolution)	- 삽입, 삭제, 갱신을 통하여 항상 최근의 정확한 데이터를 유지
동시 공유 (Concurrent Sharing)	- 여러 사용자가 동시에 원하는 데이터를 공유
내용에 의한 참조 (Content Reference)	- 데이터베이스에 있는 데이터를 주소가 아닌 내용에 따라 참조
데이터의 독립성 (Independence)	- 논리적 독립성 : 데이터의 논리적 구조를 변경시키더라도, 응용 프로그램은 변경되지 않음 - 물리적 독립성 : 데이터베이스의 물리적 구조를 변경시켜도 응용 프로그램이나 논리적 구조에는 영향을 미치지 않음

⑷ 데이터 언어

1) DDL(Data Definition Language : 데이터 정의어)

- DB의 구조, 데이터 형식, 접근 방식 등 DB의 구축과 변경 목적으로 사용하는 언어
- 데이터베이스의 논리적, 물리적 구조를 정의 및 변경
- 스키마(Schema)에 사용되는 제약 조건을 정의

2) DML(Data Manipulation Language : 데이터 조작어)

- 데이터 처리를 위한 응용 프로그램과 데이터베이스 관리 시스템 간의 인터페이스를 위한 언어
- 데이터의 검색, 삽입, 삭제, 갱신 연산 등을 포함한 집합

3) DCL(Data Control Language : 데이터 제어어)

- 보안 및 권한 제어, 무결성, 회복, 병행 제어를 위한 언어

⑸ 스키마(Schema)

1) 스키마의 정의

- 데이터베이스의 구조와 제약조건에 관해 전반적인 명세를 기술한 것
- 개체(Entity), 속성(Attribute), 관계(Relation)에 대한 정의와 이들이 유지해야 할 제약조건들을 기술한 것
- 스키마는 데이터 사전(Data Dictionary)에 저장

2) 3계층 스키마

① 외부 스키마(External Schema) - 사용자 뷰

- 데이터베이스의 논리적 구조 정의, 사용자 뷰

② 개념 스키마(Conceptual Schema) - 전체적인 뷰

- 데이터베이스의 전체적인 논리적 구조

③ 내부 스키마(Internal Schema) - 저장 스키마

- 물리적 저장장치의 입장에서 본 데이터베이스 구조

3) 데이터 독립성

① 논리적 독립성

- 개념 스키마가 변경되어도 외부 스키마에는 영향을 미치지 않도록 지원

② 물리적 독립성

- 내부 스키마가 변경되어도 외부/개념 스키마가 영향을 받지 않도록 지원

2. 데이터베이스 관리 시스템(Database Management System)

⑴ DBMS의 정의

- DBMS를 통해 데이터베이스를 관리하여 응용 프로그램들이 데이터베이스를 공유하고, 사용할 수 있는 환경을 제공

⑵ DBMS의 기능

기능	설명
데이터 정의	- 데이터에 대한 형식, 구조, 제약조건들을 명세하는 기능
데이터 조작	- 특정한 데이터를 검색하기 위한 질의, 데이터베이스의 갱신, 보고서 생성 기능
데이터 제어	- 데이터 무결성(Integrity) - 보안(Security) / 권한(Authority) 검사 - 동시성 제어(Concurrency Control)
데이터 공유	- 여러 사용자와 프로그램이 데이터베이스에 동시에 접근하도록 하는 기능
데이터 보호	- 하드웨어나 소프트웨어의 오동작 또는 권한이 없는 악의적인 접근으로부터 시스템을 보호
데이터 구축	- DBMS가 관리하는 기억 장치에 데이터를 저장하는 기능
유지보수	- 시간이 지남에 따라 변화하는 요구사항을 반영할 수 있도록 하는 기능

⑶ DBMS의 종류

1) 계층형(Hierarchical DataBase)
- 데이터 간의 관계가 트리 형태의 구조

2) 네트워크형(Network DataBase)
- 계층형 데이터베이스의 단점을 보완하여 데이터 간 N:N(다대다) 구성이 가능한 망형 모델
- CODASYL이 제안한 것으로, CODASYL DBTG모델이라고도 한다.

3) 관계형(Relational DataBase)
- 키(Key)와 값(Value)으로 이루어진 데이터들을 행(Row)과 열(Column)로 구성된 테이블 구조로 단순화시킨 모델

4) 객체 지향형(Object-Oriented DataBase)
- 객체지향 프로그래밍 개념에 기반하여 만든 데이터베이스 모델
- 비정형 데이터들을 데이터베이스화할 수 있도록 하기 위해 만들어진 모델

5) 객체 관계형(Object-Relational DataBase)
- 관계형 데이터베이스에 객체 지향 개념을 도입하여 만든 데이터베이스 모델

6) NoSQL
- Not Only SQL의 줄임말로 SQL뿐만 아니라 다양한 특성을 지원한다는 의미

7) NewSQL
- RDBMS의 SQL과 NoSQL의 장점을 결합한 관계형 모델

Section 2. 데이터베이스 설계

1. 데이터베이스 설계 단계

⑴ 요구조건 분석

- 데이터베이스의 사용자, 사용목적, 사용범위, 제약조건 등에 대한 내용을 정리하고 명세서를 작성

⑵ 개념적 설계

- 현실세계를 데이터관점으로 추상화 단계
- DBMS에 독립적으로 설계
- 데이터베이스의 개념적 스키마 구성(E-R 다이어그램)

⑶ 논리적 설계

- 자료를 컴퓨터가 이해할 수 있도록 목표 DBMS의 논리적 자료 구조로 변환하는 과정
- 특정 데이터모델(계층형, 관계형, 객체지향형 등)을 적용한 설계
- 데이터베이스의 논리적 스키마 생성
- 관계형 데이터베이스인 경우 이 단계에서 테이블을 설계하는 정규화 과정 수행
- 트랜잭션 인터페이스 설계

⑷ 물리적 설계

- 특정 DBMS의 물리적 구조와 내부적인 저장구조, 분산형태, 데이터타입의 특징, 인덱스의 특징 등을 구체화하는 설계단계
- 레코드 집중의 분석 및 설계
- 오브젝트, 접근방법, 트랜잭션분석, 인덱스, 뷰, 데이터베이스 용량설계 등을 수행
- 데이터베이스의 물리적 스키마 생성
- 트랜잭션 세부 설계

⑸ 구현

- 특정 DBMS의 DDL로 기술된 명령문을 컴파일하고, 실행시켜 데이터베이스 스키마 생성

Section 3. 데이터 모델링

1. 데이터모델 개념

(1) 데이터모델 개념

- 현실세계의 요소를 인간과 컴퓨터가 이해할 수 있는 정보로 표현한 것

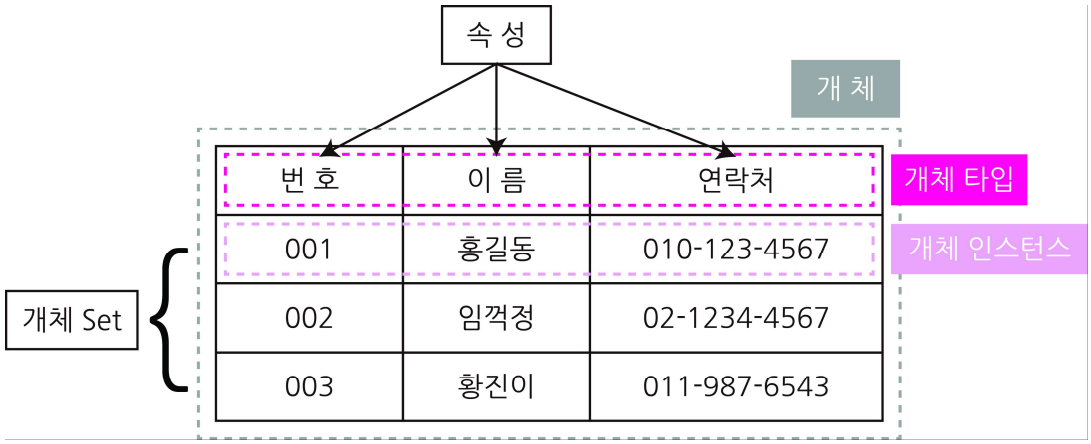
(2) 데이터모델 종류

- 계층형 데이터 모델(Hierarchical Data Model)
- 네트워크(망)형 데이터 모델(Network Data Model)
- 관계형 데이터 모델(Relational Data Model)
- 객체 지향형 데이터 모델(Object Oriented Data Model)

(3) 데이터모델 구분

구분	설명
개념적 데이터 모델	- 현실세계에 대한 인식을 추상적인 개념으로 표현 - 인간이 이해할 수 있는 정보 구조로 표현 - 대표적으로 개체-관계(E-R) 모델
논리적 데이터 모델	- 개념 데이터 모델링의 개념 구조를 컴퓨터가 이해할 수 있도록 변환한 구조 - 필드, 데이터타입 등으로 개념적 모델 구현 - 관계 모델, 계층 모델, 네트워크 모델 등으로 구분
물리적 모델	- 데이터가 저장되는 방법을 표현 - 레코드형식, 레코드 순서, 접근경로, 저장방법에 관한 정의

(4) 데이터모델 구조



구조	설명
개체(Entity)	- 데이터베이스에 데이터로 표현하려고 하는 현실 세계의 대상체 - 저장할 만한 가치가 있는 중요 데이터를 가지고 있는 사람이나 사물 등
개체 타입(Entity type)	- 개체를 구성하고 있는 속성들의 집합
개체 인스턴스 (Entity instance)	- 데이터베이스에 저장되는 구체적인 객체

개체 세트(Entity set)	- 개체 인스턴스의 집합
속성(Attribute)	- 데이터의 가장 작은 논리적 단위 - 개체가 가지고 있는 고유한 특성
관계(Relation)	- 개체와 개체가 맺고 있는 의미 있는 연관성

⑸ 데이터모델 표시해야 할 요소

요소	설명
구조(Structure)	- 데이터베이스에 표현될 대상으로서의 개체 타입과 개체 타입들 간의 관계 - 데이터 구조 및 정적 성질
연산(Operation)	- 데이터베이스에 저장될 실제 데이터를 처리하는 방법
제약조건(Constraint)	- 저장될 수 있는 데이터의 논리적인 제약조건

2. 개체-관계 모델(Entity Relation Model)

⑴ 개체-관계 모델 개념

- 데이터베이스에 대한 요구사항을 그래픽적으로 표현하는 방법
- 산출물로 개체-관계 다이어그램(Entity-Relationship Diagram)이 만들어진다.

⑵ 개체(Entity)

- 현실 세계에서 꼭 필요한 사람이나 사물과 같이 구별되는 모든 것
- ER 다이어그램에서 개체는 사각형으로 표현

⑶ 애트리뷰트, 속성(Attribute)

- 개체나 관계가 가지고 있는 고유의 특성
- DB에 저장할 데이터의 가장 작은 논리적 단위
- ER 다이어그램에서 속성은 기본적으로 원으로 표현, 키 속성은 원에 밑줄 표현, 다중값은 두 개의 원으로 표현, 유도 속성은 원을 점선으로 표현
- 속성의 유형

유형	설명
단일 값 속성	- 값을 하나만 가질 수 있는 속성 (ex. 이름, 학번 등)
다중 값 속성	- 값을 여러 개 가질 수 있는 속성 (ex. 취미 등)
단순 속성	- 의미를 더는 분해할 수 없는 속성 (ex. 성별 등)
복합 속성	- 의미를 분해할 수 있는 속성 (ex. 주소, 생년월일 등)
유도 속성	- 기존의 다른 속성의 값에서 유도되어 결정되는 속성 (ex. 주민번호와 성별)
널 속성	- 아직 결정되지 않은 존재하지 않는 값
키 속성	- 각 개체를 식별하는 데 사용하는 속성

⑷ 관계(Relationship)

- 서로 다른 개체가 맺고 있는 의미 있는 연관성

- 개체 사이의 대응 관계
- ER 다이어그램에서 관계는 마름모로 표현

⑸ E-R 다이어그램 기호

기호	기호 이름	설명
	사각형	- 개체(Entity)
	마름모	- 관계(Relationship)
	타원	- 속성(Attribute)
	밑줄 타원	- 기본키 속성
	이중 타원	- 복합속성
	선 링크	- 개체와 속성 연결

3. 데이터 모델의 품질 기준

기준항목	설명
정확성	데이터 모델이 표기법에 따라 정확하게 표현되었고, 업무영역 또는 요구사항이 정확하게 반영되었음을 의미함
완전성	데이터 모델의 구성 요소를 정의하는 데 있어서 누락을 최소화하고, 요구사항 및 업무영역 반영에 있어서 누락이 없음을 의미함
준거성	제반 준수 요건들이 누락 없이 정확하게 준수되었음을 의미함
최신성	데이터 모델이 현행 시스템의 최신 상태를 반영하고 있고, 이슈사항들이 지체 없이 반영되고 있음을 의미
일관성	여러 영역에서 공통 사용되는 데이터 요소가 전사 수준에서 한 번만 정의되고 이를 여러 다른 영역에서 참조·활용되면서, 모델 표현상의 일관성을 유지하고 있음을 의미함
활용성	작성된 모델과 그 설명 내용이 이해관계자에게 의미를 충분하게 전달할 수 있으면서, 업무 변화 시에 설계 변경이 최소화되도록 유연하게 설계되어 있음을 의미

Section 4. 논리 데이터베이스 설계

1. 논리적 데이터 모델링

⑴ 논리적 모델링

- 개념적 설계에서 추출된 실체와 속성들의 관계를 구조적으로 설계하는 단계
- 데이터 간의 관계를 어떻게 표현하느냐에 따라 관계 모델, 계층 모델, 네트워크 모델로 구분

2. 데이터베이스 정규화(Normalization)

⑴ 정규화의 개념

- 관계형 데이터베이스의 설계에서 중복을 최소화하게 데이터를 구조화

⑵ 정규화의 목적

- 데이터의 중복을 최소화
- 정보의 무손실 : 정보가 사라지지 않아야 함
- 독립적인 관계는 별개의 릴레이션으로 표현
- 정보의 검색을 보다 용이하게 함
- 이상 현상 최소화

⑶ 정규화의 장/단점

장점	- 데이터 중복의 최소화 - 저장 공간의 효율적 사용 - 릴레이션에서 발생 가능한 이상 현상 제거
단점	- 처리 명령의 복잡 - 실행 속도 저하 - 분리된 두 릴레이션 간 참조 무결성 유지를 위한 노력 필요 - 분리된 여러 개의 테이블에서 정보를 취합하기 위한 JOIN 연산이 필요

⑷ 이상 현상(Anomaly)

- 데이터 중복으로 인해 릴레이션 조작 시 예상하지 못한 곤란한 현상이 발생
- 이상의 종류

① 삽입 이상 : 데이터를 삽입할 때 불필요한 데이터가 함께 삽입되는 현상

② 삭제 이상 : 한 튜플을 삭제할 때 연쇄 삭제 현상으로 인해 정보 손실

③ 갱신 이상 : 튜플의 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상

⑸ 함수적 종속(Functional Dependency)

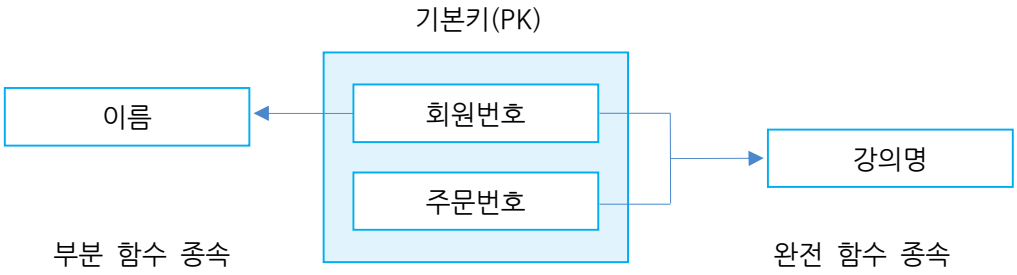
1) 함수적 종속의 개념

- 어떤 릴레이션 R의 X와 Y를 각각 속성의 부분집합이라고 가정했을 때
  - X의 값을 알면 Y의 값을 바로 식별할 수 있고, X의 값에 Y의 값이 달라질 때, Y는 X에 함수적 종속이라고 함
  - 이를 기호로 표현하면 X→Y

2) 함수적 종속 관계

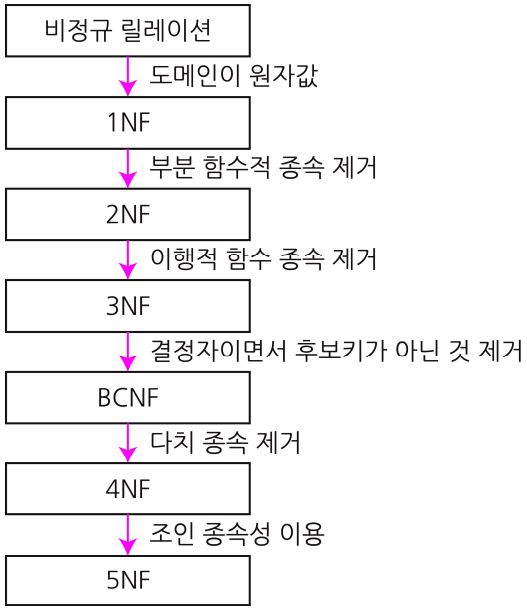
- 완전 함수적 종속(Full Functional Dependency)
  - 종속자가 기본키에만 종속되며, 기본키가 여러 속성으로 구성되어 있을 때, 기본키를 구성하는 모든 속성이 포함된 기본키의 부분집합에 종속된 경우

- 부분 함수적 종속(Partial Functional Dependency)
  - 기본키가 여러 속성으로 구성되어 있을 때, 기본키를 구성하는 속성 중 일부만 종속되는 경우



- 이행적 함수 종속(Transitive Functional Dependecy)
  - X→Y, Y→Z 이러한 종속 관계가 있을 경우, X→Z가 성립되는 경우

⑹ 정규화 과정



1) 제 1정규형(1NF)
- 어떤 릴레이션에 속한 모든 도메인이 원자값만으로 되어 있다.

2) 제 2정규형(2NF)
- 부분 함수적 종속을 모두 제거하여 완전 함수적 종속으로 만든다.

3) 제 3정규형(3NF)
- 이행적 함수 종속을 없앤다.

4) 보이스/코드(BCNF) 정규형
- 결정자 중 후보키가 아닌 것들을 제거

5) 제 4정규형(4NF)
- 다치 종속을 제거

6) 제 5정규형(5NF)
- 조인 종속 이용

Section 5. 물리 데이터베이스 설계

1. 물리 데이터베이스 설계 과정

- 사용자 DBMS 결정
- 데이터 타입 크기 결정
- 데이터 용량, 설계 및 업무 프로세스 분석
- 역정규화(반정규화)
- 인덱스 정의
- 데이터베이스 생성

2. 반정규화

⑴ 반정규화의 개념

- 시스템의 성능향상과 개발 편의성 등을 위해 정규화에 위배되는 중복을 허용하는 기법

⑵ 반정규화 시 고려사항

- 데이터의 중복이 발생하여 데이터 수정 시 무결성이 깨질 수 있다.
- 읽기 속도는 향상되지만, 삽입/삭제/수정 속도는 느려짐
- 저장 공간의 효율이 떨어짐
- 테이블이 크고 복잡해져 유지보수가 어려움
- 과도한 반정규화는 오히려 성능을 저하시킴

⑶ 반정규화의 적용순서

순서	설명
반정규화 대상 조사	- 자주 사용하는 테이블에 접근하는 프로세스의 수가 많고 항상 일정 범위만을 조회하는 경우 - 테이블에 대량의 데이터가 있고, 대량의 데이터 범위를 자주 처리하는 경우 - 통계 정보를 필요로 할 때 - 지나치게 많은 조인이 걸려 있을 때
다른 방법으로 유도	- 성능을 고려한 뷰를 생성하고, 뷰를 통해 접근하게 함으로 성능저하 위험 예방 - 인덱스나 클러스터링을 통한 성능 향상 - 파티셔닝 고려 - 애플리케이션의 로직을 변경하여 성능 향상
반정규화 수행	- 반정규화를 수행한다.

3. 데이터베이스 이중화

⑴ 데이터베이스 이중화 구성

- 장애발생 시 데이터베이스를 보호하기 위한 방법으로 동일한 데이터베이스를 중복시켜 동시에 갱신하여 관리하는 방법

⑵ 데이터베이스 이중화의 목적

- 장애 또는 재해 시 빠른 서비스 재개를 위함
- 원활한 서비스의 성능을 보장하기 위함

⑶ 데이터베이스 이중화의 분류

분류	설명
Eager 기법	- 트랜잭션 수행 중에 발생한 변경은 발생 즉시 모든 이중화서버로 전달하여 변경 내용 반영
Lazy 기법	- 트랜잭션의 수행이 완전히 완료된 후에 변경 사실에 대한 새로운 트랜잭션을 작성하여 각 노드에게 전달하는 기법

⑷ 데이터베이스 이중화의 종류

1) Active-Active
- 다중화된 장비가 모두 가동되는 방식

2) Active-Standby
- 두 대 중 하나는 가동이 되고, 하나는 장애 상황의 경우를 대비해서 준비 상태로 대기
- Active-Standby 타입

Hot Standby	- Standby 장비가 가동되었을 때 즉시 사용가능
Warm Standy	- Standby 장비가 가동되었을 때 설정에 대한 준비가 필요함
Cold Standby	- Standby 장비를 평소에는 정지시켜두며 필요에 따라서 직접 켜서 구성을 함

4. 데이터베이스 백업

⑴ 데이터베이스 백업 개념
- 정전, 사이버 공격 및 다른 중단 사태에 대비하여 복구를 진행할 수 있도록 데이터를 주기적으로 복사하는 것을 의미

⑵ 백업 방식

1) 전체 백업(Full Backup)
- 선택된 폴더의 Data를 모두 백업하는 방식

2) 증분 백업(Incremental Backup)
- Full 백업 이후 변경/추가된 Data만 백업하는 방식

3) 차등 백업(Differential Backup)
- Full 백업 이후 변경/추가된 Data를 모두 포함하여 백업

4) 실시간 백업(RealTime Backup)
- 즉각적으로 모든 변경사항을 분리된 스토리지 디바이스에 복사

5) 트랜잭션 로그 백업(Transaction Log Backup)
- 데이터베이스에서 실행되는 모든 SQL문을 기록한 로그
- REDO(다시 실행), UNDO(원상태로 복구)로 복원

6) 합성 백업
- 기존의 전체 백업본과 여러 개의 증분 백업을 합하여 새로운 전체 백업을 만드는 작업

⑶ 복구 시간 목표/복구 시점 목표

1) 복구 시간 목표(RTO)
- 서비스 중단 시점과 서비스 복원 시점 간에 허용되는 최대 지연 시간
- 서비스를 사용할 수 없는 상태로 허용되는 기간

2) 복구 시점 목표(RPO)

- 마지막 데이터 복구 시점 이후 허용되는 최대 시간
- 마지막 복구 시점과 서비스 중단 시점 사이에 허용되는 데이터 손실량

5. 데이터베이스 암호화

⑴ 데이터베이스 암호화 개념

- 데이터베이스의 내용을 암호화하는 것

⑵ 데이터베이스 암호화 방식

방식	설명
API 방식	- 애플리케이션에서 데이터의 암/복호화를 수행
Plug-in 방식	- 데이터베이스 서버에 제품을 설치→암/복호화 수행
TDE(Transparent Data Encryption)방식	- DBMS에 내장 또는 옵션으로 제공되는 암호화 기능을 이용하는 방식 - DB 내부에서 암/복호 처리를 하는 방식 - 응용 프로그램에 대한 수정이 없고 인덱스의 경우 DBMS 자체 인덱스 기능과 연동이 가능하다.
파일암호화 방식	- 데이터뿐만 아니라 비정형 데이터 암호화 적용가능
하드웨어 방식	- 별도의 하드웨어 장비를 외부에 설치

Section 6. 데이터베이스 물리속성 설계

1. 파티셔닝

⑴ 파티셔닝 개념

- 데이터베이스를 여러 부분으로 분할하는 것

⑵ 샤딩(Sharding)

- 하나의 거대한 데이터베이스나 네트워크 시스템을 여러 개의 작은 조각으로 나누어 분산 저장하여 관리하는 것

⑶ 파티셔닝의 종류

1) 수평 분할(Horizontal Partitioning)
   - 하나의 테이블의 각 행들을 분할
2) 수직 분할(Vertical Partitioning)
   - 테이블의 일부를 컬럼을 기준으로 분할

⑷ 분할 기준

1) 범위 분할(Range Partitioning)
   - Partition Key의 연속된 범위로 파티션을 정의
   - ex) 월별, 분기별 등
2) 목록 분할(List Partitioning)
   - 특정 Partition에 저장될 Data에 대한 명시적 제어
   - ex) [한국, 일본, 중국 → 아시아] [노르웨이, 스웨덴, 핀란드 → 북유럽]
3) 해시 분할(Hash Partitioning)
   - 파티션 키값에 해시 함수를 적용하고, 거기서 반환된 값으로 파티션 매핑
4) 라운드 로빈 분할(Round Robin Partitioning)
   - 데이터를 균일하게 분배해서 저장하는 방식
5) 합성 분할(Composite Partitioning)
   - 위의 기술들을 복합적으로 사용하는 방법

2. 클러스터 설계

⑴ 클러스터의 개념

- 디스크로부터 데이터를 읽어오는 시간을 줄이기 위해서 조인이나 자주 사용되는 테이블의 데이터를 디스크의 같은 위치에 저장시키는 방법

⑵ 클러스터 대상 테이블

- 분포도가 넓은 테이블
- 대량의 범위를 자주 조회하는 테이블
- 입력, 수정, 삭제가 자주 발생하지 않는 테이블
- 자주 JOIN되어 사용되는 테이블
- ORDER BY, GROUP BY, UNION이 빈번한 테이블

3. 인덱스(Index)

⑴ 인덱스의 개념

- 추가적인 저장 공간을 활용하여 데이터베이스 테이블의 검색 속도를 향상시키기 위한 자료구조

⑵ 인덱스의 종류

1) 클러스터 인덱스
- 테이블당 1개만 허용되며, 해당 컬럼을 기준으로 테이블이 물리적으로 정렬

2) 넌클러스터 인덱스
- 테이블당 약 240개의 인덱스 생성 가능
- 레코드의 원본은 정렬되지 않고, 인덱스 페이지만 정렬

3) 밀집 인덱스
- 데이터 레코드 각각에 대해 하나의 인덱스가 만들어짐

4) 희소 인덱스
- 레코드 그룹 또는 데이터 블록에 대해 하나의 인덱스가 만들어짐

⑶ 인덱스의 구조

1) 트리 기반 인덱스
- 대부분의 상용 DBMS에서는 트리 구조를 기반으로 하는 B+ 트리 인덱스를 주로 사용

2) 비트맵 인덱스
- 비트를 이용하여 컬럼값을 저장하고 이를 이용하여 주소를 자동으로 생성하는 인덱스

3) 함수 기반 인덱스
- 함수(Function)나 수식(Expression)으로 계산된 결과에 대해 B+ 트리 인덱스나 Bit-Map Index를 생성, 사용할 수 있는 기능을 제공

4) 비트맵 조인 인덱스
- 비트맵 조인 인덱스의 물리적인 구조는 비트맵 인덱스와 완전히 동일

5) 도메인 인덱스
- 개발자가 자신이 원하는 인덱스 타입을 생성할 수 있게 함으로써 인덱스 시스템에 많은 확장 가능

4. 뷰(View)

⑴ 뷰의 개념
- 하나 이상의 기본 테이블로부터 유도된, 이름을 가지는 가상 테이블

⑵ 뷰의 장/단점

장점	- 논리적 데이터 독립성을 제공한다. - 동일 데이터에 대해 동시에 여러 사용자의 상이한 요구를 지원해 준다. - 사용자의 데이터관리를 간단하게 해준다. - 접근 제어를 통한 자동 보안이 제공된다.
단점	- 독립적인 인덱스를 가질 수 없다. - ALTER VIEW문을 사용할 수 없다. - 뷰로 구성된 내용에 대한 삽입, 삭제, 갱신 연산에 제약이 따른다.

5. 시스템 카탈로그

⑴ 시스템 카탈로그
- 데이터베이스에 저장되어 있는 모든 데이터 개체들에 대한 정의나 명세에 대한 정보가 수록되어 있는 시스템 테이블

- 시스템 카탈로그를 데이터 사전(Data Dictionary)이라고도 한다.

⑵ 시스템 카탈로그의 내용

- 릴레이션 관련 정보
- 인덱스 관련 정보
- 뷰 관련 정보
- 통계 관련 정보
- 사용자 관련 정보

⑶ 시스템 카탈로그의 특징

- 시스템 카탈로그 자체도 시스템 테이블로 구성되어 있어 사용자가 SQL문을 이용하여 내용을 검색해 볼 수 있다.
- 시스템 카탈로그는 데이터베이스 관리 시스템에 의해 생성되고 유지된다.

Section 7. 관계 데이터베이스 모델

1. 관계 데이터 모델

⑴ 관계 데이터 모델 개념

- 데이터의 논리적 구조가 릴레이션, 즉 테이블 형태의 평면 파일로 표현되는 데이터 모델

⑵ 관계 데이터 릴레이션의 구조

〈학생 릴레이션〉

구조	설명
속성(Attribute)	- 릴레이션의 각 열을 속성 또는 Attribute라고 한다. - 데이터를 구성하는 가장 작은 논리적인 단위 - 속성의 수 = 디그리(Degree) = 차수
튜플(Tuple)	- 릴레이션의 행을 튜플(Tuple)이라고 한다. - 튜플의 수 = 카디널리티(Cardinality) = 기수
도메인(Domain)	- 하나의 속성이 가질 수 있는 값의 범위
차수(Degree)	- 하나의 릴레이션에서 속성의 전체 개수
카디널리티(Cardinality)	- 하나의 릴레이션에서 튜플의 전체 개수

⑶ 릴레이션

- 데이터들을 2차원 테이블의 구조로 저장한 것

1) 릴레이션의 구성

- 릴레이션 스키마 : 릴레이션 이름과 릴레이션에 포함된 모든 속성의 이름으로 정의하는 릴레이션의 논리적인 구조
- 릴레이션 인스턴스 : 릴레이션 스키마에 실제로 저장된 데이터의 집합

2) 릴레이션의 특징

- 튜플의 유일성 : 릴레이션 안에는 똑같은 튜플이 존재할 수 없음
- 튜플의 무순서성 : 튜플 사이에는 순서가 없음
- 속성의 무순서성 : 속성 사이에는 순서가 없음
- 속성의 원자성 : 속성은 더 이상 분해할 수 없는 원자값만 가진다.
- 튜플들의 삽입, 갱신, 삭제작업이 실시간으로 일어나므로 릴레이션은 수시로 변한다.

2. 관계데이터 언어(관계대수, 관계해석)

⑴ 관계 대수의 개념

- 원하는 데이터를 얻기 위해 데이터를 어떻게 찾는지에 대한 처리 과정을 명시하는 절차적인 언어

⑵ 순수 관계 연산자

1) SELECT

- 릴레이션에서 주어진 조건을 만족하는 튜플을 선택하는 연산자
- 기호 : σ(시그마)
- 표기법 : σ<조건>(R)
- 조건에서는 =, ≠, <, ≤, >, ≥ 등의 기호를 사용한 비교 연산이 허용, AND(∧), OR(∨), NOT(¬) 등의 논리 연산자를 사용
- 예시1 : σ 성적 > 90 (학생)
- 예시2 : σ 성적 >= 90 ^ 학과 = '컴퓨터' (학생)

2) PROJECT

- 주어진 릴레이션에서 속성 리스트에 제시된 속성 값만을 추출하는 연산자
- 기호 : π(파이)
- 표기법 : π<리스트>(R)
- 예시1 : π 학번, 성적 (학생)
- 예시2 : π 학번, 이름, 성적 (σ 성적 >= 90 (학생))

3) JOIN

- 두 개의 릴레이션으로부터 연관된 튜플들을 결합하는 연산자
- 기호 : ⋈(보타이)
- 표기법 : R⋈<조건>S
- 예시 : (학생) ⋈ 학번=학번 (수강과목)

4) DIVISION

- 릴레이션 S의 모든 튜플과 관련이 있는 릴레이션 R의 튜플들을 반환
- 기호 : ÷(나누기)
- 표기법 : R÷S
- 예시1 : (R) ÷ (S1)
- 예시2 : (R) ÷ (S2)

⑶ 일반 집합 연산자

1) 합집합(Union)

- 두 릴레이션에 존재하는 튜플의 합집합을 구하되, 결과로 생성된 릴레이션에서 중복되는 튜플은 제거
- 표기법 : ∪
- 예시 : A과목 ∪ B과목

2) 교집합(Intersection)

- 두 릴레이션에 존재하는 튜플의 교집합을 구하는 연산
- 표기법 : ∩
- 예시 : A과목 ∩ B과목

3) 차집합(Difference)

- 두 릴레이션에 존재하는 튜플의 차집합을 구하는 연산
- 표기법 : -
- 예시 : A과목 - B과목

4) 교차곱(Cartesian Product)

- 두 릴레이션에 있는 튜플들의 순서쌍을 구하는 연산
- 표기법 : X
- 예시 : 학생 X 과목

⑷ 관계해석

- 관계 데이터 모델의 제안자인 코드(E. F. Codd)가 수학의 Predicate Calculus(술어 해석)에 기반을 두고 관계 데이터베이스를 위해 제안
- 관계해석은 원하는 정보가 무엇이라는 것만 정의하는 비절차적 특성
- 튜플 관계해석과 도메인 관계해석이 있다.
- 연산자

구분	기호	설명
연산자	∨	- OR 연산
	∧	- AND 연산
	¬	- NOT 연산
정량자	∀	- 모든 가능한 튜플 “For All”
	∃	- 어떤 튜플 하나라도 존재

Section 8. 키와 무결성 제약조건

1. 속성(컬럼)

(1) 속성의 개념

- 릴레이션에서 정보를 나타내는 최소 단위로, 각 열의 상태나 특성을 나타내는 항목을 말한다.

(2) 속성의 특징

- 시스템을 구축하려는 업무 프로세스에 필요한 정보로 구성되어야 한다.

(3) 속성의 분류

분류	설명
기본속성	- 업무로부터 추출한 모든 속성
설계속성	- 코드성 데이터, 릴레이션 식별용 일련번호
파생속성	- 다른 속성에 영향을 받아 발생하는 속성 - 계산값, 합계, 재고 등

(4) 세부 의미에 따른 분류

분류	설명
단순 속성 (Simple Attribute)	- 나이, 성별같이 다른 속성들로 구성될 수 없는 단순한 속성
복합 속성 (Composite Attribute)	- 주소와 같이 시, 구, 동처럼 여러 세부 속성들로 구성될 수 있는 속성

(5) 구성방식 따른 분류

분류	설명
PK(Primary Key) 속성	- 릴레이션에서 튜플을 유일하게 구분할 수 있는 속성
FK(Foreign Key) 속성	- 다른 릴레이션과의 관계에서 참조하고 있는 속성
일반 속성	- 릴레이션에 포함된 속성 중, PK와 FK가 아닌 속성

(6) 도메인

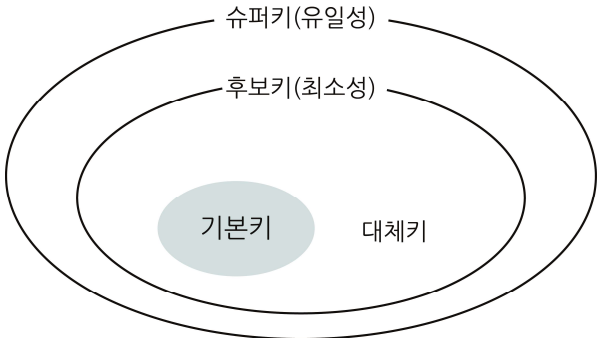
- 속성이 가질 수 있는 값의 범위

2. 키 종류

⑴ 키(Key)의 개념

- 릴레이션에서 다른 튜플들과 구별할 수 있는 유일한 기준이 되는 컬럼

⑵ 키(Key)의 종류



1) 후보키(Candidate Key)
   - 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별할 수 있는 속성들의 부분집합
   - 튜플에 대한 유일성과 최소성을 만족시켜야 한다.
2) 기본키(Primary Key)
   - 후보키 중에서 선택한 주키(Main Key)
   - 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성
3) 대체키(Alternate Key)
   - 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키
4) 슈퍼키(Super Key)
   - 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키
   - 튜플에 대한 유일성은 만족하지만, 최소성은 만족시키지 못한다.
5) 외래키(Foreign Key)
   - 관계(Relation)를 맺고 있는 릴레이션 R1, R2에서 릴레이션 R1이 참조하고 있는 릴레이션 R2의 기본키와 같은 R1 릴레이션의 속성

3. 데이터베이스 무결성

⑴ 데이터베이스 무결성 개념

- 데이터의 정확성, 일관성, 유효성이 유지되는 것

⑵ 데이터베이스 무결성 종류

1) 개체 무결성(Entity Integrity)
   - 모든 릴레이션은 기본 키(Primary Key)를 가져야 한다.
   - 기본키는 중복되지 않은 고유한 값을 가져야 한다.
   - 릴레이션의 기본키는 NULL 값을 허용하지 않는다.
2) 참조 무결성(Referential Integrity)
   - 외래키 값은 NULL이거나 참조하는 릴레이션의 기본키 값과 동일해야 한다.
   - 각 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다.

- 참조 무결성 제약조건

제약조건	설명
제한(Restrict)	- 문제가 되는 연산을 거절
연쇄(Cascade)	- 부모 릴레이션에서 튜플을 삭제하면 자식 릴레이션에서 이 튜플을 참조하는 튜플도 함께 삭제
널값(Nullify)	- 부모 릴레이션에서 튜플을 삭제하면 자식 릴레이션에서 이 튜플을 참조하는 튜플들의 외래키에 NULL 등록
기본값(Default)	- Null을 넣는 대신에 디폴트 값을 등록

3) 도메인 무결성(Domain Integrity)
- 속성들의 값은 정의된 도메인에 속한 값이어야 한다.
- 성별이라는 컬럼에는 '남', '여'를 제외한 데이터는 제한되어야 한다.

4) 고유 무결성(Unique Integrity)
- 릴레이션의 특정 속성에 대해 각 튜플이 갖는 속성 값들이 서로 달라야 한다.

5) 키 무결성(Key Integrity)
- 하나의 릴레이션에는 적어도 하나의 키가 존재해야 한다.

6) 릴레이션 무결성(Relation Integrity)
- 삽입, 삭제, 갱신과 같은 연산을 수행하기 전과 후에 대한 상태의 제약

Section 9. 물리데이터 모델 품질 검토

1. CRUD 분석

⑴ CRUD의 개념

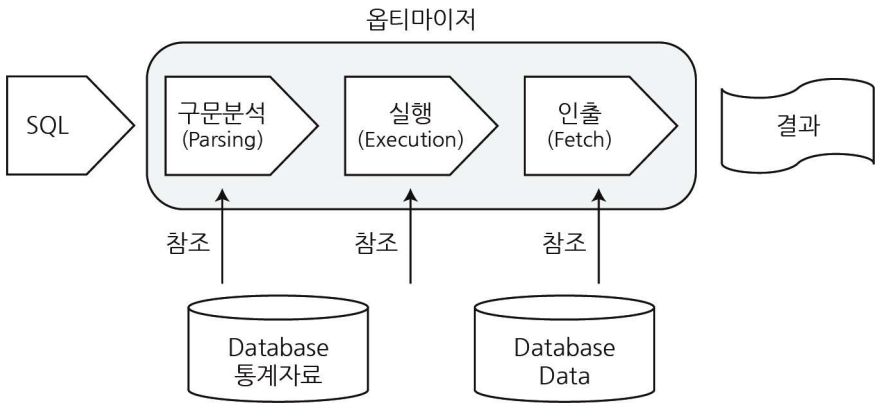
- 데이터 처리 기능인 Create(생성), Read(읽기), Update(갱신), Delete(삭제)를 묶어서 표현한 말이다.
- 시스템 구축 시 프로세스와 DB에 저장되는 데이터 사이의 의존관계를 표현하는 표

⑵ CRUD의 필요성

1) 모델링 작업검증
2) 중요 산출물
3) 테스트 시 사용
4) 인터페이스 현황 파악

2. 옵티마이저

⑴ SQL 처리 흐름



1) 구문분석 단계

- 사용자가 요청한 SQL문이 데이터베이스에서 사용된 적이 있는지 공유 풀 영역을 검색하여 확인
- 이미 사용했다면 구문분석 작업을 하지 않고, 처음 사용되었다면 구문분석 작업을 수행
- SQL 문이 문법에 따라 정상적으로 작성되었는지 분석하고, SQL내에 포함된 테이블, 뷰 등이 데이터베이스에 존재하는지 확인

2) 실행 단계

- SQL문에서 사용된 데이터가 버퍼캐시 영역에 존재하는지 검색
- 데이터버퍼 캐시영역에 존재한다면, 테이블의 해당 데이터 파일로부터 테이블을 읽지 않고 캐시영역의 데이터를 그대로 추출
- 존재하지 않는다면 정의된 테이블의 해당 데이터 파일로부터 테이블을 읽어서 데이터버퍼 캐시영역에 저장

3) 추출 단계

- 실행단계가 끝나면 서버 프로세스는 데이터버퍼 캐시영역에서 관련 테이블 데이터를 읽어서 사용자가 요청한 클라이언트로 전송

⑵ 옵티마이저 개념

- 사용자가 질의한 SQL문에 대해 최적의 실행 방법을 결정하는 역할을 수행
- 실행계획을 결정한다.

- 옵티마이저의 구분

구분	설명
규칙기반 옵티마이저 (Rule Based Optimizer)	- 규칙(우선순위)를 가지고 실행 계획을 생성 - 인덱스 유무, 연산자, 객체 등을 참조하여 우선순위를 부여
비용기반 옵티마이저 (Cost Based Optimizer)	- SQL문을 처리하는 데 필요한 비용이 가장 적은 실행계획을 선택하는 방식 - 테이블, 인덱스, 컬럼 등의 다양한 객체 통계정보와 시스템 통계정보 활용 - 통계정보가 정확하지 않을 경우 정확한 비용 예측이 불가능하여 비효율적인 실행계획이 생성된다.

3. SQL 성능 튜닝

⑴ 튜닝의 개념

- SQL문을 최적화하여 빠른 시간 내에 원하는 결과값을 얻기 위한 작업
- 주어진 H/W 환경을 통해 처리량과 응답속도를 개선하기 위해 수행

⑵ 튜닝 영역

튜닝 영역	설명
데이터베이스 설계튜닝	- 데이터베이스 설계 단계에서 성능을 고려하여 설계
데이터베이스 환경	- 성능을 고려하여 메모리나 블록 크기 등을 지정
SQL 문장 튜닝	- 성능을 고려하여 SQL 문장을 작성

⑶ SQL 성능 최적화를 위한 유틸리티

- SQL Trace
- TKPROF(Trace Kernel PROFile)
- EXPLAIN PLAN

⑷ Row Migration / Row Chaining

1) Row Migration
   - 데이터가 수정되면서 데이터가 더 커져서 기존 블록(Block)에 못 들어가는 경우
   - 다른 블록에 데이터를 넣고, 기존 블록 위치에는 링크를 남긴다.
2) Row Chaining
   - 컬럼이 너무 길어서 DB BLOCK 사이즈보다 길어진 경우 블록 두 개에 이어서 한 Row가 저장

Section 10. 분산 데이터베이스

1. 분산 데이터베이스

⑴ 분산 데이터베이스(Distribute Database)의 정의

- 여러 곳으로 분산되어있는 데이터베이스를 하나의 가상 시스템으로 사용할 수 있도록 한 데이터베이스

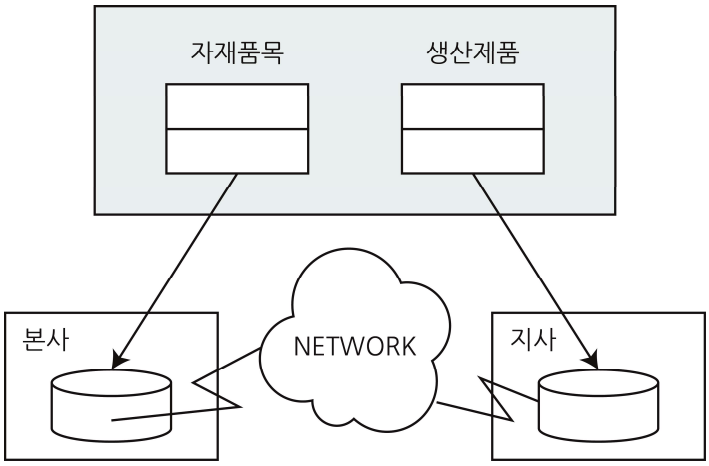
⑵ 분산 데이터베이스 구성요소

구성요소	설명
분산 처리기	- 자체적으로 처리 능력을 가지며, 지리적으로 분산되어 있는 컴퓨터 시스템
분산 데이터베이스	- 지리적으로 분산되어 있는 데이터베이스로서 해당 지역의 특성에 맞게 데이터베이스가 구성
통신 네트워크	- 분산 처리기들을 통신망으로 연결하여 논리적으로 하나의 시스템처럼 작동할 수 있도록 하는 통신 네트워크

⑶ 분산 데이터베이스의 적용 기법

1) 테이블 위치 분산

- 설계된 테이블의 위치를 각각 다르게 위치시키는 것
- 테이블의 구조가 변하지 않고, 데이터베이스에 중복되어 생성되지 않는다.



2) 테이블 분할(Fragmentation) 분산

- 각각의 테이블을 쪼개어 분산하는 방법
- 종류

수평분할 (Horizontal Fragmentation)	- 테이블을 특정 칼럼의 값을 기준으로 로우(Row)를 분리 - 컬럼은 분리되지 않음
수직분할 (Vertical Fragmentation)	- 테이블 칼럼을 기준으로 컬럼(Column)을 분리

3) 테이블 복제(Replication) 분산

- 동일한 테이블을 다른 지역이나 서버에서 동시에 생성하여 관리하는 유형

• 종류

부분복제 (Segment Replication)	- 마스터 데이터베이스에서 테이블의 일부의 내용만 다른 지역이나 서버에 복제
광역복제 (Broadcast Replication)	- 마스터 데이터베이스의 테이블의 내용을 각 지역이나 서버에 복제

4) 테이블 요약(Summarization) 분산

• 지역 간 또는 서버 간에 데이터가 비슷하지만 서로 다른 유형으로 존재

• 종류

분석요약 (Rollup Replication)	- 각 지역별로 존재하는 요약정보를 마스터에 통합하여 다시 전체에 대해서 요약정보를 산출하는 분산방법
통합요약 (Consolidation Replication)	- 각 지역별로 존재하는 다른 내용의 정보를 마스터에 통합하여 다시 전체에 대해서 요약정보를 산출하는 분산방법

⑷ 투명성 조건

투명성 조건	설명
위치 투명성 (Location)	- 액세스하려는 데이터베이스의 실제 위치를 알 필요 없이 단지 데이터베이스의 논리적인 명칭만으로 액세스할 수 있음 - 데이터가 물리적으로 저장되어 있는 곳을 알 필요 없이 논리적인 입장에서 데이터가 모두 자신의 사이트에 있는 것처럼 처리하는 특성
분할 투명성 (Division)	- 하나의 논리적 테이블이 여러 단편으로 분할되어 각 단편의 사본이 여러 위치에 저장
지역사상 투명성 (Local Mapping)	- 지역DBMS와 물리적 DB사이의 Mapping 보장 - 각 지역시스템 이름과 무관한 이름 사용 가능
중복 투명성 (Replication)	- 동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고, 시스템은 자동으로 여러 자료에 대한 작업을 수행함
병행 투명성 (Concurrency)	- 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않음
장애 투명성 (Failure)	- 트랜잭션, DBMS, 네트워크, 컴퓨터 장애에도 불구하고 트랜잭션을 정확하게 처리함

⑸ CAP 이론

1) 개념

• 어떤 분산 환경에서도 일관성(C), 가용성(A), 분단 허용성(P) 세 가지 속성 중, 두 가지만 가질 수 있다는 것

2) 특징의 의미

① 일관성(Consistency)

② 가용성(Availability)

③ 분단 허용성(Partition Tolerance)

2. 트랜잭션

(1) 트랜잭션의 개념

- 데이터베이스의 상태를 변환시키는 하나의 논리적인 기능을 수행하는 작업 단위

(2) 트랜잭션의 성질

성질	설명
원자성(Atomicity)	- 트랜잭션의 연산은 데이터베이스에 모두 반영되든지 아니면 전혀 반영되지 않아야 한다. - Commit과 Rollback 명령어에 의해 보장받는다.
일관성(Consistency)	- 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환한다.
독립성, 격리성(Isolation)	- 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행 중에 다른 트랜잭션의 연산이 끼어들 수 없다.
영속성(Durability)	- 성공적으로 완료된 트랜잭션의 결과는 시스템이 고장이 나더라도 영구적으로 반영되어야 한다.

(3) 트랜잭션의 상태

상태	설명
활동(Active)	트랜잭션이 실행 중인 상태
실패(Failed)	트랜잭션 실행에 오류가 발생하여 중단된 상태
철회(Aborted)	트랜잭션이 비정상적으로 종료되어 Rollback 연산을 수행한 상태
부분 완료 (Partially Committed)	트랜잭션의 마지막 연산까지 실행했지만, Commit 연산이 실행되기 직전의 상태
완료(Committed)	트랜잭션이 성공적으로 종료되어 Commit 연산을 실행한 후의 상태

02 SQL 활용

Section 1. 기본 SQL 작성

1. SQL(Structured Query Language)

⑴ SQL의 개념

- 데이터베이스 시스템에서 자료를 처리하는 용도로 사용되는 구조적 데이터 질의 언어

⑵ SQL 문법의 종류

1) Data Definition Language (DDL) - 데이터 정의어
   - 데이터가 저장되는 테이블이나 각종 개체들을 정의하는 데 사용되는 명령
   - CREATE, ALTER, DROP, RENAME, TRUNCATE
2) Data Manipulation Language (DML) - 데이터 조작어
   - 데이터베이스 내의 데이터를 조작(추출, 생성, 수정, 삭제)하는 명령
   - SELECT, INSERT, UPDATE, DELETE
3) Data Control Language (DCL) - 데이터 제어어
   - 데이터베이스에 접근하고, 객체들을 사용하도록 권한을 주고 회수하는 명령
   - GRANT, REVOKE
4) Transaction Control Language (TCL) - 트랜잭션 제어어
   - 논리적인 작업의 단위를 묶어 이에 의해 조작된 결과를 작업 단위별로 제어하는 명령어
   - COMMIT, ROLLBACK, SAVEPOINT

Section 4. 절차형 SQL

1. 저장 프로시저(Stored Procedure)

⑴ 저장 프로시저의 개념

- 일련의 쿼리를 마치 하나의 함수처럼 실행하기 위한 쿼리의 집합

⑵ 저장 프로시저의 구조

```
CREATE OR REPLACE PROCEDURE 프로시저명
( 변수1 IN 변수타입, 변수2 OUT 변수타입, 변수3 IN OUT 변수타입.... )
IS
    변수 처리부
BEGIN
    처리내용
EXCEPTION
    예외처리부
END;
```

2. 트리거

⑴ 트리거의 개념

- 테이블에 대한 이벤트에 반응해 자동으로 실행되는 작업

⑵ 트리거의 유형

1) 행 트리거
   - 테이블 안의 영향을 받은 행 각각에 대해 실행
   - 데이터 변화가 생길 때마다 실행
   - FOR EACH ROW 옵션 사용
2) 문장 트리거
   - INSERT, UPDATE, DELETE문에 대해 단 한 번만 실행

⑶ 트리거의 실행 시기

- BEFORE : 이벤트 전
- AFTER : 이벤트 후

3. 사용자 정의 함수

⑴ 사용자 정의 함수의 개념

- 프로시저와 사용자 정의 함수 모두 호출하게 되면 미리 정의해 놓은 기능을 수행하는 모듈
- 파라미터는 입력 파라미터만 가능하고, 리턴값이 하나이다.

⑵ 사용자 정의 함수의 구조

```
CREATE OR REPLACE FUNCTION 함수명
( 매개변수1, 매개변수2, 매개변수3,..... )
RETURN 데이터 타입
IS
    변수 처리부
BEGIN
    처리내용
    RETURN 반환값;
EXCEPTION
    예외처리부
END;
```

03 병행제어와 데이터전환

Section 1. 병행제어와 회복

1. 병행제어

⑴ 병행제어의 개념

- 여러 트랜잭션들이 동시에 실행되면서도 데이터베이스의 일관성을 유지할 수 있게 하는 기법

⑵ 병행제어의 문제점

1) 갱신 분실(Lost Update)

- 두 개 이상의 트랜잭션이 같은 자료를 공유하여 갱신할 때 갱신 결과의 일부가 없어지는 현상

데이터	트랜잭션 1	트랜잭션 2
1000	READ(1000)	
1000		READ(1000)
1000	ADD(1000)	
1000		ADD(2000)
2000	STORE(2000)	
3000		STORE(3000)

2) 비완료 의존성(Uncommitted Dependency)

- 하나의 트랜잭션 수행이 실패한 후 회복되기 전에 다른 트랜잭션이 실패한 갱신 결과를 참조하는 현상

데이터	트랜잭션 1	트랜잭션 2
1000	READ(1000)	
1000	ADD(1000)	
2000	STORE(2000)	READ(2000)
1000	장애로 ROLLBACK	

3) 모순성(Inconsistency)

- 두 개의 트랜잭션이 병행수행될 때 원치 않는 자료를 이용함으로써 발생하는 문제
- 갱신 분실과 비슷해 보이지만 여러 데이터를 가져올 때 발생하는 문제

4) 연쇄 복귀(Cascading Rollback)

- 병행수행 된 트랜잭션들 중 어느 하나에 문제가 생겨 Rollback하는 경우 다른 트랜잭션도 함께 Rollback되는 현상

데이터	트랜잭션 1	트랜잭션 2
1000	READ(1000)	
1000	ADD(1000)	
2000	STORE(2000)	

2000		READ(2000)
2000		ADD(2000)
4000		STORE(4000)
1000	장애로 ROLLBACK	

⑶ 병행제어 기법

1) 로킹(Locking)

- 트랜잭션이 어떤 데이터에 접근하고자 할 때 로킹 수행
- 로킹 단위: 필드, 레코드, 파일, 데이터베이스 모두 로킹 단위가 될 수 있음
- 로킹 단위에 따른 구분

구분	로크 수	병행성	오버헤드
로킹 단위가 크면	적어짐	낮아짐	감소
로킹 단위가 작으면	많아짐	높아짐	증가

2) 2단계 로킹 규약(Two-Phase Locking Protocol)

- Lock과 Unlock이 동시에 이루어지면 일관성이 보장되지 않으므로 Lock만 가능한 단계와 Unlock만 가능한 단계를 구분
- 확장단계: 새로운 Lock은 가능하고 Unlock은 불가능하다.
- 축소단계: Unlock은 가능하고 새로운 Lock은 불가능하다.

3) 타임스탬프(Time Stamp)

- 데이터에 접근하는 시간을 미리 정해서 정해진 시간(Time Stamp)의 순서대로 데이터에 접근하여 수행

4) 낙관적 병행제어(Optimistic Concurrency Control)

- 트랜잭션 수행 동안은 어떠한 검사도 하지 않고, 트랜잭션 종료 시에 일괄적으로 검사

5) 다중 버전 병행제어(Multi-version, Concurrency Control)

- 여러 버전의 타임스탬프를 비교하여 스케줄상 직렬가능성이 보장되는 타임스탬프를 선택

2. 회복(Database Recovery)

⑴ 회복의 개념

- 트랜잭션들을 수행하는 도중 장애로 인해 손상된 데이터베이스를 손상되기 이전의 정상적인 상태로 복구시키는 작업

⑵ 장애의 유형

유형	설명
트랜잭션 장애	트랜잭션의 실행 시 논리적인 오류로 발생할 수 있는 에러 상황
시스템 장애	H/W 시스템 자체에서 발생할 수 있는 에러 상황
미디어 장애	디스크 자체의 손상으로 발생할 수 있는 에러 상황

⑶ Undo와 Redo

Undo	트랜잭션 로그를 이용하여 오류와 관련된 모든 변경을 취소하여 복구 수행
Redo	트랜잭션 로그를 이용하여 오류가 발생한 트랜잭션을 재실행하여 복구 수행

⑷ 회복 기법

1) 로그 기반 회복 기법

① 지연갱신 회복 기법(Deferred Update)

- 트랜잭션의 부분 완료 상태에선 변경 내용을 로그 파일에만 저장
- 커밋이 발생하기 전까진 데이터베이스에 기록하지 않음
- 중간에 장애가 생기더라도 데이터베이스에 기록되지 않았으므로 UNDO가 필요 없음(미실행된 로그 폐기)

② 즉시갱신 회복 기법(Immediate Update)

- 트랜잭션 수행 도중에도 변경 내용을 즉시 데이터베이스에 기록
- 커밋 발생 이전의 갱신은 원자성이 보장되지 않는 미완료 갱신이므로 장애 발생 시 UNDO 필요

2) 검사점 회복 기법(Checkpoint Recovery)

- 장애 발생 시 검사점(Checkpoint) 이전에 처리된 트랜잭션은 회복에서 제외하고 검사점 이후에 처리된 트랜잭션은 회복 작업 수행

3) 그림자 페이징 회복 기법(Shadow Paging Recovery)

- 트랜잭션이 실행되는 메모리상의 Current Page Table과 하드디스크의 Shadow Page Table 이용

4) 미디어 회복 기법(Media Recovery)

- 디스크와 같은 비휘발성 저장 장치가 손상되는 장애 발생을 대비한 회복 기법

5) ARIES 회복 기법(Algorithms for Recovery and Isolation Exploiting Semantics)

- 주요 3단계

단계	설명
분석단계	- 붕괴가 발생한 시점에 REDO가 시작되어야 하는 로그의 위치를 결정
REDO 단계	- REDO 시작 위치의 로그로부터 로그가 끝날 때까지 REDO를 수행 - REDO 된 로그 레코드의 리스트를 관리하여 불필요한 REDO 연산이 수행되지 않도록 한다.
UNDO 단계	- 로그를 역순으로 읽으면서 진행 트랜잭션의 연산을 역순으로 UNDO

Section 2. 데이터 전환

1. ETL(Extraction, Transformation, Loading)

⑴ ETL 개념

- 기존의 원천 시스템에서 데이터를 추출(Extraction)하여 목적 시스템의 데이터베이스에 적합한 형식과 내용으로 변환(Transformation)한 후, 목적 시스템에 적재(Loading)하는 일련의 과정

⑵ ETL 기능

기능	설명
추출(Extraction)	하나 또는 그 이상의 데이터 소스로부터 데이터 획득
변환(Transformation)	데이터 클렌징, 형식 변환 및 표준화, 데이터 통합
적재(Load)	변형 단계의 처리가 완료된 데이터를 목표 시스템에 적재