시험에 나오는 것만 공부한다!

2024 시나공

기출문제집

정보처리기사

필기

7, 8 정규화 과정

12 접근통제 기술

길벗알앤디 지음
(강윤석, 김용갑, 김우경, 김종일)

길벗

FitNesse	웹 기반 테스트케이스 설계, 실행, 결과 확인 등을 지원하는 테스트 프레임워크
NTAF	FitNesse의 장점인 협업 기능과 STAF의 장점인 재사용 및 확장성을 통합한 NHN(Naver)의 테스트 자동화 프레임워크
Selenium	다양한 브라우저 및 개발 언어를 지원하는 웹 애플리케이션 테스트 프레임워크
watir	Ruby를 사용하는 애플리케이션 테스트 프레임워크

핵심 162 APM(Application Performance Management/Monitoring)

2407431

APM은 애플리케이션의 성능 관리를 위해 접속자, 자원 현황, 트랜잭션 수행 내역, 장애 진단 등 다양한 모니터링 기능을 제공하는 도구를 의미한다.

- APM은 리소스 방식과 엔드투엔드(End-to-End)의 두 가지 유형이 있다.
  - 리소스 방식 : Nagios, Zabbix, Cacti 등
  - 엔드투엔드 방식 : VisualVM, 제니퍼, 스카우터 등

3과목 데이터베이스 구축

핵심 163 데이터베이스 설계 순서

2407603

요구 조건 분석	요구 조건 명세서 작성
↓	
개념적 설계	개념 스키마, 트랜잭션 모델링, E-R 모델
↓	
논리적 설계	목표 DBMS에 맞는 논리 스키마 설계, 트랜잭션 인터페이스 설계
↓	
물리적 설계	목표 DBMS에 맞는 물리적 구조의 데이터로 변환
↓	
구현	목표 DBMS의 DDL(데이터 정의어)로 데이터베이스 생성, 트랜잭션 작성

23.2, 22.4

핵심 164 개념적 설계(정보 모델링, 개념화)

2407605

개념적 설계란 정보의 구조를 얻기 위하여 현실 세계의 무한성과 계속성을 이해하고, 다른 사람과 통신하기 위하여 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정이다.

- 개념적 설계 단계에서는 개념 스키마 모델링과 트랜잭션 모델링을 병행 수행한다.
- 개념적 설계 단계에서는 요구 분석 단계에서 나온 결과인 요구 조건 명세를 DBMS에 독립적인 E-R 다이어그램으로 작성한다.
- DBMS에 독립적인 개념 스키마를 설계한다.

22.7, 20.6

핵심 165 논리적 설계(데이터 모델링)

2407606

논리적 설계 단계란 현실 세계에서 발생하는 자료를 컴퓨터가 이해하고 처리할 수 있는 물리적 저장장치에 저장할 수 있도록 변환하기 위해 특정 DBMS가 지원하는 논리적 자료 구조로 변환(mapping)시키는 과정이다.

- 개념 세계의 데이터를 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계로 표현되는 논리적 구조의 데이터로 모델화한다.
- 개념적 설계가 개념 스키마를 설계하는 단계라면 논리적 설계에서는 개념 스키마를 평가 및 정제하고 DBMS에 따라 서로 다른 논리적 스키마를 설계하는 단계이다.
- 트랜잭션의 인터페이스를 설계한다.
- 관계형 데이터베이스라면 테이블을 설계하는 단계이다.

22.4, 22.3, 21.8, 21.5, 21.3, 20.9

2407607

핵심 166 물리적 설계(데이터 구조화)

물리적 설계란 논리적 설계 단계에서 논리적 구조로 표현된 데이터를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정이다.

- 물리적 설계 단계에서는 다양한 데이터베이스 응용에 대해 처리 성능을 얻기 위해 데이터베이스 파일의 저장 구조 및 액세스 경로를 결정한다.
- 저장 레코드의 양식, 순서, 접근 경로, 조회가 집중되는 레코드와 같은 정보를 사용하여 데이터가 컴퓨터에 저장되는 방법을 묘사한다.
- **물리적 설계 시 고려할 사항** : 트랜잭션 처리량, 응답 시간, 디스크 용량, 저장 공간의 효율화 등

23.2, 22.4 , 20.9

2459916

핵심 167 데이터 모델

데이터 모델은 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형이다.

데이터 모델의 구성 요소 읽어보기

개체 (Entity)	데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체
속성 (Attribute)	데이터의 가장 작은 논리적 단위로서 파일 구조상의 데이터 항목 또는 데이터 필드에 해당함
관계 (Relationship)	개체 간의 관계 또는 속성 간의 논리적인 연결을 의미함

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
제약 조건 (Constraint)	데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약 조건

23.7, 22.7

2408201

핵심 168 E-R 모델의 개요

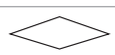
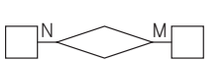
E-R 모델은 개념적 데이터 모델의 가장 대표적인 것으로, 1976년 피터 첸(Peter Chen)에 의해 제안되고 기본적인 구성 요소가 정립되었다.

- E-R 모델은 개체 타입(Entity Type)과 이들 간의 관계 타입(Relationship Type)을 이용해 현실 세계를 개념적으로 표현한다.
- E-R 모델에서는 데이터를 개체(Entity), 관계(Relationship), 속성(Attribute)으로 묘사한다.
- E-R 모델은 특정 DBMS를 고려한 것은 아니다.
- E-R 다이어그램으로 표현하며, 1:1, 1:N, N:M 등의 관계 유형을 제한 없이 나타낼 수 있다.

23.7, 22.7, 22.3, 21.5, 21.3, 20.9, 20.6

2408202

핵심 169 E-R 다이어그램

기호	기호 이름	의미
	사각형	개체(Entity) 타입
	마름모	관계(Relationship) 타입
	타원	속성(Attribute)
	이중 타원	다중값 속성(복합 속성)
	밑줄 타원	기본키 속성
	복수 타원	복합 속성 예 성명은 성과 이름으로 구성
	관계	1:1, 1:N, N:M 등의 개체 간 관계에 대한 대응수를 선 위에 기술함
	선, 링크	개체 타입과 속성을 연결

2408301

핵심 170 관계형 데이터 모델

관계형 데이터 모델은 가장 널리 사용되는 데이터 모델로, 2차원적인 표(Table)를 이용해서 데이터 상호 관계를 정의하는 DB 구조를 말한다.

- 파일 구조처럼 구성한 테이블들을 하나의 DB로 묶어서 테이블 내에 있는 속성들 간의 관계(Relationship)를 설정하거나 테이블 간의 관계를 설정하여 이용한다.
- 기본키(Primary Key)와 이를 참조하는 외래키(Foreign Key)로 데이터 간의 관계를 표현한다.
- 계층 모델과 망 모델의 복잡한 구조를 단순화시킨 모델이다.
- 관계형 모델의 대표적인 언어는 SQL이다.
- 1:1, 1:N, N:M 관계를 자유롭게 표현할 수 있다.

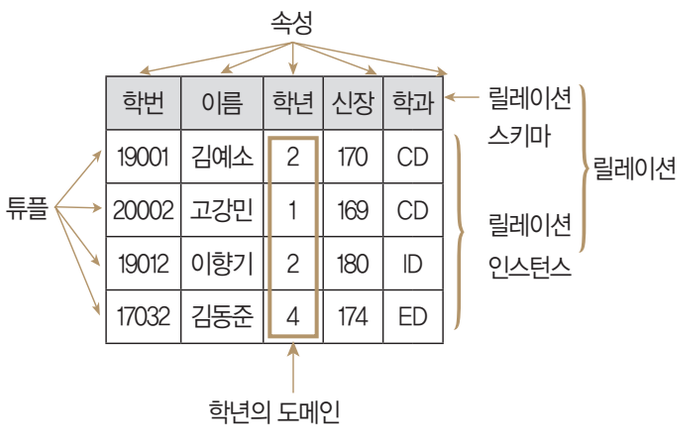
23.2, 22.4, 22.3, 21.5, 21.3, 20.9, 20.8, 20.6

2408402

핵심 171 관계형 데이터베이스의 Relation 구조

릴레이션은 데이터들을 표(Table)의 형태로 표현한 것으로 구조를 나타내는 릴레이션 스키마와 실제 값들인 릴레이션 인스턴스로 구성된다.

〈학생〉 릴레이션



튜플(Tuple)

- 튜플은 릴레이션을 구성하는 각각의 행을 말한다.
- 튜플은 속성의 모임으로 구성된다.
- 파일 구조에서 레코드와 같은 의미이다.
- 튜플의 수를 카디널리티(Cardinality) 또는 기수, 대응수라고 한다.

속성(Attribute)

- 속성은 데이터베이스를 구성하는 가장 작은 논리적 단위이다.
- 파일 구조상의 데이터 항목 또는 데이터 필드에 해당된다.
- 속성은 개체의 특성을 기술한다.
- 속성의 수를 디그리(Degree) 또는 차수라고 한다.

도메인(Domain)

- 도메인은 하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(Atomic)값들의 집합이다.
- 도메인은 실제 애트리뷰트 값이 나타날 때 그 값의 합법 여부를 시스템이 검사하는데에도 이용된다.

예 성별 애트리뷰트의 도메인은 '남'과 '여'로, 그 외의 값은 입력될 수 없다.

23.7, 22.7, 22.4, 21.5, 20.8

2408403

핵심 172 릴레이션의 특징

- 한 릴레이션에는 똑같은 튜플이 포함될 수 없으므로 릴레이션에 포함된 튜플들은 모두 상이하다.

예 〈학생〉 릴레이션을 구성하는 김예소 레코드는 김예소에 대한 학적 사항을 나타내는 것으로 〈학생〉 릴레이션 내에서는 유일하다.

- 한 릴레이션에 포함된 튜플 사이에는 순서가 없다.

예 〈학생〉 릴레이션에서 김예소 레코드와 고강민 레코드의 위치가 바뀌어도 상관없다.

- 튜플들의 삽입, 삭제 등의 작업으로 인해 릴레이션은 시간에 따라 변한다.

예 〈학생〉 릴레이션에 새로운 학생의 레코드를 삽입하거나 기존 학생에 대한 레코드를 삭제함으로써 테이블은 내용 면에서나 크기 면에서 변하게 된다.

- 릴레이션 스키마를 구성하는 속성들 간의 순서는 중요하지 않다.
  - 예 학번, 이름 등의 속성을 나열하는 순서가 이름, 학번 순으로 바뀌어도 데이터 처리에는 아무런 영향을 미치지 않는다.
- 속성의 유일한 식별을 위해 속성의 명칭은 유일해야 하지만, 속성을 구성하는 값은 동일한 값이 있을 수 있다.
  - 예 각 학생의 학년을 기술하는 속성인 '학년'은 다른 속성명들과 구분되어 유일해야 하지만 '학년' 속성에는 2, 1, 2, 4 등이 입력된 것처럼 동일한 값이 있을 수 있다.
- 릴레이션을 구성하는 튜플을 유일하게 식별하기 위해 속성들의 부분집합을 키(Key)로 설정한다.
  - 예 〈학생〉 릴레이션에서는 '학번'이나 '이름'이 튜플들을 구분하는 유일한 값인 키가 될 수 있다.
- 속성의 값은 논리적으로 더 이상 쪼갤 수 없는 원자값만을 저장한다.
  - 예 '학년'에 저장된 1, 2, 4 등은 더 이상 세분화할 수 없다.

23.7, 23.5, 23.2, 22.7, 22.4, 22.3, 21.8, 20.9, 20.6

핵심 173 키(Key)

2459917

키(Key)는 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 튜플들을 서로 구분할 수 있는 기준이 되는 애트리뷰트를 말한다.

학번,주민번호 후보키 (Candidate Key)	• 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용하는 속성들의 부분집합, 즉 기본키로 사용할 수 있는 속성들을 말함 • 후보키는 릴레이션에 있는 모든 튜플에 대해서 유일성과 최소성을 만족시켜야 함
학번,주민번호 기본키 (Primary Key)	• 후보키 중에서 특별히 선정된 주키(Main Key)로 중복된 값을 가질 수 없음 • 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성 • 기본키는 NULL 값을 가질 수 없다. 즉 튜플에서 기본키로 설정된 속성에는 NULL 값이 있어서는 안 됨
학번이 기본키면 주민번호가 대체키 대체키 (Alternate Key)	• 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키를 의미함 • 보조키라고도 함
학번+주민번호 슈퍼키 (Super Key)	• 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키로서 릴레이션을 구성하는 모든 튜플들 중 슈퍼키로 구성된 속성의 집합과 동일한 값은 나타나지 않음 • 슈퍼키는 릴레이션을 구성하는 모든 튜플에 대해 유일성은 만족시키지만, 최소성은 만족시키지 못함
외래키 (Foreign Key)	• 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합을 의미함 • 한 릴레이션에 속한 속성 A와 참조 릴레이션의 기본키인 B가 동일한 도메인 상에서 정의되었을 때의 속성 A를 외래키라고 함

23.2, 22.7, 22.4, 21.8, 21.5, 21.3, 20.8, 20.6

핵심 174 무결성(Integrity)

2459918

무결성이란 데이터베이스에 저장된 데이터 값과 그것이 표현하는 현실 세계의 실제값이 일치하는 정확성을 의미한다.

- 개체 무결성(Entity Integrity, 실체 무결성) : 기본 테이블의 기본키를 구성하는 어떤 속성도 Null 값이나 중복값을 가질 수 없다는 규정
- 도메인 무결성(Domain Integrity, 영역 무결성) : 주어진 속성 값이 정의된 도메인에 속한 값이어야 한다는 규정
- 참조 무결성(Referential Integrity) : 외래키 값은 Null이거나 참조 릴레이션의 기본키 값과 동일해야 한다. 즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다는 규정
- 사용자 정의 무결성(User-Defined Integrity) : 속성 값들이 사용자가 정의한 제약조건에 만족해야 한다는 규정

22.7, 21.8, 21.5, 21.3, 20.9, 20.8

핵심 175 관계대수의 개요

2459919

관계대수는 관계형 데이터베이스에서 원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 절차적인 언어이다.

- 관계대수는 릴레이션을 처리하기 위해 연산자와 연산규칙을 제공하는 언어로 피연산자가 릴레이션이고, 결과도 릴레이션이다.
- 질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다.

- 관계대수에는 관계 데이터베이스에 적용하기 위해 특별히 개발한 순수 관계 연산자와 수학적 집합 이론에서 사용하는 일반 집합 연산자가 있다.
- 순수 관계 연산자
  - Select
  - Project
  - Join
  - Division
- 일반 집합 연산자
  - UNION(합집합)
  - INTERSECTION(교집합)
  - DIFFERENCE(차집합)
  - CARTESIAN PRODUCT(교차곱)

23.7, 23.2, 22.3, 21.3, 20.8, 20.6

2408702

핵심 176 순수 관계 연산자

Select	릴레이션에 존재하는 튜플 중에서 선택 조건을 만족하는 튜플의 부분집합을 구하여 새로운 릴레이션을 만드는 연산 • 릴레이션의 행(가로)에 해당하는 튜플을 구하는 것이므로 수평 연산이라고도 함 • 연산자의 기호는 그리스 문자 시그마(σ)를 사용함
Project	주어진 릴레이션에서 속성 리스트(Attribute List)에 제시된 속성 값만을 추출하여 새로운 릴레이션을 만드는 연산이다. 단 연산 결과에 중복이 발생하면 중복이 제거됨 • 릴레이션의 열(세로)에 해당하는 Attribute를 추출하는 것이므로 수직 연산자라고도 함 • 연산자의 기호는 그리스 문자 파이(π)를 사용함
Join	공통 속성을 중심으로 두 개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만드는 연산 • 연산자의 기호는 ⋈를 사용함
Division	X⊃Y인 두 개의 릴레이션 R(X)와 S(Y)가 있을 때, R의 속성이 S의 속성값을 모두 가진 튜플에서 S가 가진 속성을 제외한 속성만을 구하는 연산 • 연산자의 기호는 ÷를 사용함

23.7, 23.5, 21.8, 21.5

2408706

핵심 177 일반 집합 연산자

연산자	기능 및 수학적 표현	카디널리티
합집합 UNION ∪	두 릴레이션에 존재하는 튜플의 합집합을 구하되, 결과로 생성된 릴레이션에서 중복되는 튜플은 제거되는 연산	합집합의 카디널리티는 두 릴레이션 카디널리티의 합보다 크지 않음
교집합 INTERSECTION ∩	두 릴레이션에 존재하는 튜플의 교집합을 구하는 연산	교집합의 카디널리티는 두 릴레이션 중 카디널리티가 적은 릴레이션의 카디널리티보다 크지 않음
차집합 DIFFERENCE −	두 릴레이션에 존재하는 튜플의 차집합을 구하는 연산	차집합의 카디널리티는 릴레이션 R의 카디널리티 보다 크지 않음
교차곱 CARTESIAN PRODUCT ×	두 릴레이션에 있는 튜플들의 순서쌍을 구하는 연산	교차곱의 디그리는 두 릴레이션의 디그리를 더한 것과 같고, 카디널리티는 두 릴레이션의 카디널리티를 곱한 것과 같음

23.7, 23.2, 22.7, 22.3

2408707

핵심 178 관계해석(Relational Calculus)

관계해석은 관계 데이터 모델의 제안자인 코드(E. F. Codd)가 수학의 Predicate Calculus(술어 해석)에 기반을 두고 관계 데이터베이스를 위해 제안했다.

- 관계해석은 관계 데이터의 연산을 표현하는 방법으로, 원하는 정보를 정의할 때는 계산 수식을 사용한다.
- 관계해석은 원하는 정보가 무엇이라는 것만 정의하는 비절차적 특성을 지닌다.
- 튜플 관계해석과 도메인 관계해석이 있다.
- 기본적으로 관계해석과 관계대수는 관계 데이터베이스를 처리하는 기능과 능력면에서 동등하며, 관계대수로 표현한 식은 관계해석으로 표현할 수 있다.
- 질의어로 표현한다.
- 주요 논리기호

기호	구성 요소	설명
∀	전칭 정량자	가능한 모든 튜플에 대하여(For All)
∃	존재 전량자	하나라도 일치하는 튜플이 있음(There Exists)

23.7, 23.5, 22.7, 21.8, 20.9

핵심 179 정규화의 개요

2408801

정규화란 함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정이다. 테이블 나눔

- 하나의 종속성이 하나의 릴레이션에 표현될 수 있도록 분해해가는 과정이라 할 수 있다.
- 정규형에는 제1정규형, 제2정규형, 제3정규형, BCNF형, 제4정규형, 제5정규형이 있으며, 차수가 높아질수록 만족시켜야 할 제약 조건이 늘어난다.
- 정규화는 데이터베이스의 논리적 설계 단계에서 수행한다.
- 정규화는 논리적 처리 및 품질에 큰 영향을 미친다.
- 정규화된 데이터 모델은 일관성, 정확성, 단순성, 비중복성, 안정성 등을 보장한다.

23.2, 21.8, 20.8

핵심 180 정규화의 목적

2408802

- 데이터 구조의 안정성 및 무결성을 유지한다.
- 어떠한 릴레이션이라도 데이터베이스 내에서 표현 가능하게 만든다.
- 효과적인 검색 알고리즘을 생성할 수 있다.
- 데이터 중복을 배제하여 이상(Anomaly)의 발생 방지 및 자료 저장 공간의 최소화가 가능하다.
- 데이터 삽입 시 릴레이션을 재구성할 필요성을 줄인다.
- 데이터 모형의 단순화가 가능하다.
- 속성의 배열 상태 검증이 가능하다.
- 개체와 속성의 누락 여부 확인이 가능하다.
- 자료 검색과 추출의 효율성을 추구한다.

21.8, 21.5, 21.3, 20.8

핵심 001 이상(Anomaly)의 개념 및 종류

2408803

정규화를 거치지 않으면 데이터베이스 내에 데이터들이 불필요하게 중복되어 릴레이션 조작 시 예기치 못한 곤란한 현상이 발생하는데, 이를 이상(Anomaly)이라 하며 삽입 이상, 삭제 이상, 갱신 이상이 있다.

삽입 이상 (Insertion Anomaly)	릴레이션에 데이터를 삽입할 때 의도와는 상관없이 원하지 않은 값들도 함께 삽입되는 현상
삭제 이상 (Deletion Anomaly)	릴레이션에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 연쇄가 일어나는 현상
갱신 이상 (Update Anomaly)	릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상

23.7, 23.5, 23.2, 22.7, 22.4, 22.3, 21.8, 21.5, 21.3, 20.9, 20.8, 20.6

핵심 182 정규화 과정 책보기 P 309

2408805

비정규 릴레이션
↓ 도메인이 원자값
1NF
↓ 부분적 함수 종속 제거
2NF
↓ 이행적 함수 종속 제거
3NF
↓ 결정자이면서 후보키가 아닌 것 제거
BCNF
↓ 다치 종속 제거
4NF
↓ 조인 종속성 이용
5NF

정규화 단계 암기 요령

두부를 좋아하는 정규화가 두부가게에 가서 가게에 있는 두부를 다 달라고 말하니 주인이 깜짝 놀라며 말했다.

두부이걸다줘? ≒ 도부이결다조

도메인이 원자값

부분적 함수 종속 제거

이행적 함수 종속 제거

결정자이면서 후보키가 아닌 것 제거

다치 종속 제거

조인 종속성 이용

22.3, 21.8, 20.8, 20.6

2408831

핵심 183 이행적 종속 / 함수적 종속

이행적 종속(Transitive Dependency) 관계

A → B이고 B → C일 때 A → C를 만족하는 관계를 의미한다.

함수적 종속(Functional Dependency)

- 함수적 종속은 데이터들이 어떤 기준값에 의해 종속되는 것을 의미한다.
- 예를 들어 〈수강〉 릴레이션이 (학번, 이름, 과목명)으로 되어 있을 때, '학번'이 결정되면 '과목명'에 상관없이 '학번'에는 항상 같은 '이름'이 대응된다. '학번'에 따라 '이름'이 결정될 때 '이름'을 '학번'에 함수 종속적이라고 하며 '학번 → 이름'과 같이 쓴다.

23.7, 20.9

2408901

핵심 184 반정규화의 개념

반정규화란 시스템의 성능 향상, 개발 및 운영의 편의성 등을 위해 정규화된 데이터 모델을 통합, 중복, 분리하는 과정으로, 의도적으로 정규화 원칙을 위배하는 행위이다.

- 반정규화를 수행하면 시스템의 성능이 향상되고 관리 효율성은 증가하지만 데이터의 일관성 및 정합성이 저하될 수 있다.
- 과도한 반정규화는 오히려 성능을 저하시킬 수 있다.
- 반정규화를 위해서는 사전에 데이터의 일관성과 무결성을 우선으로 할지, 데이터베이스의 성능과 단순화를 우선으로 할지를 결정해야 한다.
- 반정규화 방법에는 테이블 통합, 테이블 분할, 중복 테이블 추가, 중복 속성 추가 등이 있다.

20.6

2459920

핵심 185 반정규화 방법

테이블 통합	• 두 개의 테이블이 조인(Join)되는 경우가 많아 하나의 테이블로 합쳐 사용하는 것이 성능 향상에 도움이 될 경우 수행함 • 두 개의 테이블에서 발생하는 프로세스가 동일하게 자주 처리되는 경우, 두 개의 테이블을 이용하여 항상 조회를 수행하는 경우 테이블 통합을 고려함
테이블 분할	• 테이블을 수직 또는 수평으로 분할하는 것 • 수평 분할(Horizontal Partitioning) : 레코드(Record)를 기준으로 테이블을 분할하는 것 • 수직 분할(Vertical Partitioning) : 하나의 테이블에 속성이 너무 많을 경우 속성을 기준으로 테이블을 분할하는 것
중복 테이블 추가	• 여러 테이블에서 데이터를 추출해서 사용해야 하거나 다른 서버에 저장된 테이블을 이용해야 하는 경우 중복 테이블을 추가하여 작업의 효율성을 향상시킬 수 있음 • 중복 테이블 추가 방법 : 집계 테이블의 추가, 진행 테이블의 추가, 특정 부분만을 포함하는 테이블의 추가
중복 속성 추가	조인해서 데이터를 처리할 때 데이터를 조회하는 경로를 단축하기 위해 자주 사용하는 속성을 하나 더 추가하는 것

23.7, 22.7, 22.4, 21.5, 21.3

2459921

핵심 186 시스템 카탈로그

시스템 카탈로그는 시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스이다.

- 시스템 카탈로그 내의 각 테이블은 사용자를 포함하여 DBMS에서 지원하는 모든 데이터 객체에 대한 정의나 명세에 관한 정보를 유지 관리하는 시스템 테이블이다.
- 카탈로그들이 생성되면 데이터 사전(Data Dictionary)에 저장되기 때문에 좁은 의미로는 카탈로그를 데이터 사전이라고도 한다.
- 시스템 카탈로그에 저장된 정보를 메타 데이터(Meta-Data)라고 한다.
- 카탈로그 자체도 시스템 테이블로 구성되어 있어 일반 이용자도 SQL을 이용하여 내용을 검색해 볼 수 있다.
- INSERT, DELETE, UPDATE문으로 카탈로그를 갱신하는 것은 허용되지 않는다.
- 데이터베이스 시스템에 따라 상이한 구조를 갖는다.
- 카탈로그는 DBMS가 스스로 생성하고 유지한다.

- 카탈로그의 갱신 : 사용자가 SQL문을 실행시켜 기본 테이블, 뷰, 인덱스 등에 변화를 주면 시스템이 자동으로 갱신함

※ Data Directory

– 데이터 사전에 수록된 데이터를 실제로 접근하는 데 필요한 정보를 관리 유지하는 시스템이다.

– 시스템 카탈로그는 사용자와 시스템 모두 접근할 수 있지만 데이터 디렉터리는 시스템만 접근할 수 있다.

23.2, 22.7, 22.4, 22.3, 21.8

2409301

핵심 187 트랜잭션

트랜잭션은 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.

- 트랜잭션은 데이터베이스 시스템에서 병행 제어 및 회복 작업 시 처리되는 작업의 논리적 단위로 사용된다.
- 트랜잭션은 사용자가 시스템에 대한 서비스 요구 시 시스템이 응답하기 위한 상태 변환 과정의 작업 단위로 사용된다.

23.7, 22.7, 22.4, 22.3

2459922

핵심 188 트랜잭션의 상태

- 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) : 트랜잭션을 모두 성공적으로 실행한 후 Commit 연산이 실행되기 직전인 상태
- 완료(Committed) : 트랜잭션을 모두 성공적으로 실행한 후 Commit 연산을 실행한 후의 상태

23.7, 23.5, 23.2, 22.7, 22.4, 21.8, 21.3, 20.9, 20.8, 20.6

2409302

핵심 189 트랜잭션의 특성

Atomicity (원자성)	• 트랜잭션의 연산은 데이터베이스에 모두 반영되도록 완료(Commit)되든지 아니면 전혀 반영되지 않도록 복구(Rollback)되어야 함 • 트랜잭션 내의 모든 명령은 반드시 완벽히 수행되어야 하며, 모두가 완벽히 수행되지 않고 어느 하나라도 오류가 발생하면 트랜잭션 전부가 취소되어야 함
Consistency (일관성)	• 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환함 • 시스템이 가지고 있는 고정 요소는 트랜잭션 수행 전과 트랜잭션 수행 완료 후의 상태가 같아야 함
Isolation (독립성, 격리성, 순차성)	• 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행중에 다른 트랜잭션의 연산이 끼어들 수 없음 • 수행중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조할 수 없음
Durability (영속성, 지속성)	성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 함

22.7, 20.9

2409303

핵심 190 CRUD 분석

CRUD는 '생성(Create), 읽기(Read), 갱신(Update), 삭제(Delete)'의 앞 글자만 모아서 만든 용어이며, CRUD 분석은 데이터베이스 테이블에 변화를 주는 트랜잭션의 CRUD 연산에 대해 CRUD 매트릭스를 작성하여 분석하는 것이다.

- CRUD 분석으로 테이블에 발생되는 트랜잭션의 주기별 발생 횟수를 파악하고 연관된 테이블들을 분석하면 테이블에 저장되는 데이터의 양을 유추할 수 있다.

22.4, 21.8, 21.3 2409401

핵심 191 인덱스(Index)

인덱스는 데이터 레코드를 빠르게 접근하기 위해 〈키 값, 포인터〉 쌍으로 구성되는 데이터 구조이다.

- 인덱스는 데이터가 저장된 물리적 구조와 밀접한 관계가 있다.
- 인덱스는 레코드가 저장된 물리적 구조에 접근하는 방법을 제공한다.
- 인덱스를 통해서 파일의 레코드에 대한 액세스를 빠르게 수행할 수 있다.
- 레코드의 삽입과 삭제가 수시로 일어나는 경우에는 인덱스의 개수를 최소로 하는 것이 효율적이다.
- 데이터 정의어(DDL)를 이용하여 사용자가 생성, 변경, 제거할 수 있다.

21.8 2459923

핵심 192 인덱스의 종류

트리 기반 인덱스	인덱스를 저장하는 블록들이 트리 구조를 이루고 있는 것으로, 상용 DBMS에서는 트리 구조 기반의 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 생성하여 사용함 • 데이터를 입력하거나 수정할 때 함수를 적용해야 하므로 부하가 발생할 수 있음
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)라고도 함

23.7, 22.4, 22.3, 21.3, 20.9, 20.8, 20.6 2459924

핵심 193 뷰(View)

뷰는 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된, 이름을 가지는 가상 테이블이다.

- 뷰는 저장장치 내에 물리적으로 존재하지 않지만, 사용자에게는 있는 것처럼 간주된다.
- 뷰는 데이터 보정 작업, 처리 과정 시험 등 임시적인 작업을 위한 용도로 활용된다.

뷰(View)의 특징

- 뷰는 기본 테이블로부터 유도된 테이블이기 때문에 기본 테이블과 같은 형태의 구조를 사용하며, 조작도 기본 테이블과 거의 같다.
- 뷰는 가상 테이블이기 때문에 물리적으로 구현되어 있지 않다.
- 데이터의 논리적 독립성을 제공할 수 있다.
- 필요한 데이터만 뷰로 정의해서 처리할 수 있기 때문에 관리가 용이하고 명령문이 간단해진다.
- 뷰를 통해서만 데이터에 접근하게 하면 뷰에 나타나지 않는 데이터를 안전하게 보호하는 효율적인 기법으로 사용할 수 있다.
- 기본 테이블의 기본키를 포함한 속성(열) 집합으로 뷰를 구성해야만 삽입, 삭제, 갱신 연산이 가능하다.
- 일단 정의된 뷰는 다른 뷰의 정의에 기초가 될 수 있다.
- 뷰를 정의할 때는 CREATE문, 제거할 때는 DROP문을 사용한다.

뷰(View)의 장 · 단점

장점	• 논리적 데이터 독립성을 제공함 • 동일 데이터에 대해 동시에 여러 사용자의 상이한 응용이나 요구를 지원해 줌 • 사용자의 데이터 관리를 간단하게 해줌 • 접근 제어를 통한 자동 보안이 제공됨
단점	• 독립적인 인덱스를 가질 수 없음 • 뷰의 정의를 변경할 수 없음 • 뷰로 구성된 내용에 대한 삽입, 삭제, 갱신 연산에 제약이 따름

23.2, 22.7, 21.5, 20.8

핵심 194 파티션(Patition)

2459925

데이터베이스에서 파티션은 대용량의 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것을 말한다.

- 대용량 DB의 경우 중요한 몇 개의 테이블에만 집중되어 데이터가 증가되므로, 이런 테이블들을 작은 단위로 나눠 분산시키면 성능 저하를 방지할 뿐만 아니라 데이터 관리도 쉬워진다.

파티션의 종류

범위 분할 (Range Partitioning)	지정한 열의 값을 기준으로 범위를 지정하여 분할함 예 일별, 월별, 분기별 등
해시 분할 (Hash Partitioning)	• 해시 함수를 적용한 결과 값에 따라 데이터를 분할함 • 특정 파티션에 데이터가 집중되는 범위 분할의 단점을 보완한 것으로, 데이터를 고르게 분산할 때 유용함 • 특정 데이터가 어디에 있는지 판단할 수 없음 • 고객번호, 주민번호 등과 같이 데이터가 고른 컬럼에 효과적임
조합 분할 (Composite Partitioning)	• 범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할하는 방식 • 범위 분할한 파티션이 너무 커서 관리가 어려울 때 유용함
목록 분할 (List Partitioning)	지정한 열 값에 대한 목록을 만들어 이를 기준으로 분할함 예 '국가'라는 열에 '한국', '미국', '일본'이 있는 경우 '미국'을 제외할 목적으로 '아시아'라는 목록을 만들어 분할함
라운드 로빈 분할 (Round Robin Partitioning)	• 레코드를 균일하게 분배하는 방식 • 각 레코드가 순차적으로 분배되며, 기본키가 필요 없음

22.4, 22.3

핵심 195 분산 데이터베이스 정의 및 구성 요소

2459926

분산 데이터베이스는 논리적으로는 하나의 시스템에 속하지만 물리적으로는 네트워크를 통해 연결된 여러 개의 컴퓨터 사이트(Site)에 분산되어 있는 데이터베이스를 말한다.

- 분산 데이터베이스의 구성 요소

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

23.5, 22.7, 22.3, 20.8, 20.6

핵심 196 분산 데이터베이스의 목표

2409904

- 위치 투명성(Location Transparency) : 액세스하려는 데이터베이스의 실제 위치를 알 필요 없이 단지 데이터베이스의 논리적인 명칭만으로 액세스할 수 있음
- 중복 투명성(Replication Transparency) : 동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고, 시스템은 자동으로 여러 자료에 대한 작업을 수행함
- 병행 투명성(Concurrency Transparency) : 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않음
- 장애 투명성(Failure Transparency) : 트랜잭션, DBMS, 네트워크, 컴퓨터 장애에도 불구하고 트랜잭션을 정확하게 처리함

23.7, 22.7

핵심 197 분산 데이터베이스의 장 · 단점

2409905

장점	• 지역 자치성이 높음 • 자료의 공유성이 향상됨 • 분산 제어가 가능함 • 시스템 성능이 향상됨 • 중앙 컴퓨터의 장애가 전체 시스템에 영향을 끼치지 않음 • 효용성과 융통성이 높음 • 신뢰성 및 가용성이 높음 • 점진적 시스템 용량 확장이 용이함
단점	• DBMS가 수행할 기능이 복잡함 • 데이터베이스 설계가 어려움 • 소프트웨어 개발 비용이 증가함 • 처리 비용이 증가함 • 잠재적 오류가 증가함

21.3

핵심 198 암호화(Encryption)

2459927

암호화는 데이터를 보낼 때 송신자가 지정한 수신자 이외에는 그 내용을 알 수 없도록 평문을 암호문으로 변환하는 것이다.

- **암호화(Encryption) 과정** : 암호화되지 않은 평문을 정보 보호를 위해 암호문으로 바꾸는 과정
- **복호화(Decryption) 과정** : 암호문을 원래의 평문으로 바꾸는 과정

개인키 암호 방식(Private Key Encryption) = 비밀키 암호 방식

비밀키 암호화 기법은 동일한 키로 데이터를 암호화하고 복호화한다.

- 비밀키 암호화 기법은 대칭 암호 방식 또는 단일키 암호화 기법이라고도 한다.
- 비밀키는 제3자에게는 노출시키지 않고 데이터베이스 사용 권한이 있는 사용자만 나누어 가진다.
- **종류** : 전위 기법, 대체 기법, 대수 기법, 합성 기법 (DES, LUCIFER)

공개키 암호 방식(Public Key Encryption)

공개키 암호화 기법은 서로 다른 키로 데이터를 암호화하고 복호화한다.

- 데이터를 암호화할 때 사용하는 키(공개키, Public Key)는 데이터베이스 사용자에게 공개하고, 복호화할 때의 키(비밀키, Secret Key)는 관리자가 비밀리에 관리 하는 방법이다.
- 공개키 암호화 기법은 비대칭 암호 방식이라고도 하며, 대표적으로 RSA(Rivest Shamir Adleman)가 있다.

23.5, 22.4, 21.8, 21.3, 20.9

핵심 199 접근통제 기술

2410201

임의 접근통제(DAC; Discretionary Access Control)

- 임의 접근통제는 데이터에 접근하는 사용자의 신원에 따라 접근 권한을 부여하는 방식이다.
- 데이터 소유자가 접근통제 권한을 지정하고 제어한다.
- 객체를 생성한 사용자가 생성된 객체에 대한 모든 권한을 부여받고, 부여된 권한을 다른 사용자에게 허가할 수도 있다.
- 임의 접근통제에 사용되는 SQL 명령어에는 GRANT와 REVOKE가 있다.

강제 접근통제(MAC; Mandatory Access Control)

- 강제 접근통제는 주체와 객체의 등급을 비교하여 접근 권한을 부여하는 방식이다.
- 시스템이 접근통제 권한을 지정한다.
- 데이터베이스 객체별로 보안 등급을 부여할 수 있고, 사용자별로 인가 등급을 부여할 수 있다.
- 주체는 자신보다 보안 등급이 높은 객체에 대해 읽기, 수정, 등록이 모두 불가능하고, 보안 등급이 같은 객체에 대해서는 읽기, 수정, 등록이 가능하고, 보안 등급이 낮은 객체는 읽기가 가능하다.

역할기반 접근통제(RBAC; Role Based Access Control)

- 역할기반 접근통제는 사용자의 역할에 따라 접근 권한을 부여하는 방식이다.
- 중앙관리자가 접근 통제 권한을 지정한다.
- 임의 접근통제와 강제 접근통제의 단점을 보완하였으며, 다중 프로그래밍 환경에 최적화된 방식이다.
- 중앙관리자가 역할마다 권한을 부여하면, 책임과 자질에 따라 역할을 할당받은 사용자들은 역할에 해당하는 권한을 사용할 수 있다.

21.5

핵심 200 강제 접근통제(MAC)의 보안 모델

2410231

벨 라파듈라 모델(Bell-LaPadula Model)

- 군대의 보안 레벨처럼 정보의 기밀성에 따라 상하 관계가 구분된 정보를 보호하기 위해 사용한다.
- 보안 취급자의 등급을 기준으로 읽기 권한과 쓰기 권한이 제한된다.
- 자신의 보안 레벨 이상의 문서를 작성할 수 있고, 자신의 보안 레벨 이하의 문서를 읽을 수 있다.

비바 무결성 모델(Biba Integrity Model)

- 벨 라파듈라 모델을 보완한 수학적 모델로, 무결성을 보장하는 최초의 모델이다.
- 비인가자에 의한 데이터 변형을 방지한다.

클락-윌슨 무결성 모델(Clark-Wilson Integrity Model)

무결성 중심의 상업용 모델로, 사용자가 직접 객체에 접근할 수 없고 프로그램에 의해 접근이 가능한 보안 모델이다.

만리장성 모델(Chinese Wall Model)

서로 이해 충돌 관계에 있는 객체 간의 정보 접근을 통제하는 모델이다.

23.7, 22.3, 20.9

핵심 201 DAS (Direct Attached Storage)

2410402

DAS는 서버와 저장장치를 전용 케이블로 직접 연결하는 방식으로, 일반 가정에서 컴퓨터에 외장하드를 연결하는 것이 여기에 해당된다.

- 서버에서 저장장치를 관리한다.
- 저장장치를 직접 연결하므로 속도가 빠르고 설치 및 운영이 쉽다.
- 초기 구축 비용 및 유지보수 비용이 저렴하다.
- 직접 연결 방식이므로 다른 서버에서 접근할 수 없고 파일을 공유할 수 없다.
- 확장성 및 유연성이 상대적으로 떨어진다.
- 저장 데이터가 적고 공유가 필요 없는 환경에 적합하다.

핵심 202 NAS (Network Attached torage)

2410403

- NAS는 서버와 저장장치를 네트워크를 통해 연결하는 방식이다.
- 별도의 파일 관리 기능이 있는 NAS Storage가 내장된 저장장치를 직접 관리한다.
- Ethernet 스위치를 통해 다른 서버에서도 스토리지에 접근할 수 있어 파일 공유가 가능하고, 장소에 구애받지 않고 저장장치에 쉽게 접근할 수 있다.
- DAS에 비해 확장성 및 유연성이 우수하다.
- 접속 증가 시 성능이 저하될 수 있다.

22.7, 21.5

핵심 203 SAN (Storage Area Network)

2410404

SAN은 DAS의 빠른 처리와 NAS의 파일 공유 장점을 혼합한 방식으로, 서버와 저장장치를 연결하는 전용 네트워크를 별도로 구성하는 방식이다.

- 광 채널(FC) 스위치를 이용하여 네트워크를 구성한다.
- 광 채널 스위치는 서버나 저장장치를 광케이블로 연결하므로 처리 속도가 빠르다.
- 저장장치를 공유함으로써 여러 개의 저장장치나 백업 장비를 단일화시킬 수 있다.
- 확장성, 유연성, 가용성이 뛰어나다.
- 높은 트랜잭션 처리에 효과적이나 기존 시스템의 경우 장비의 업그레이드가 필요하고, 초기 설치 시에는 별도의 네트워크를 구축해야 하므로 비용이 많이 든다.

23.7, 23.2, 22.7, 21.8, 21.5, 20.6

핵심 204 DDL(Data Define Language, 데이터 정의어)

2459928

DDL은 SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다.

- 논리적 데이터 구조와 물리적 데이터 구조의 사상을 정의한다.
- 데이터베이스 관리자나 데이터베이스 설계자가 사용한다.
- DDL(데이터 정의어)의 세 가지 유형

명령어	기능
CREATE	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의함
ALTER	TABLE에 대한 정의를 변경하는 데 사용함
DROP	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 삭제함

↳ DROP은 테이블을 삭제하기 때문에 조심해야한다.

23.5, 22.4, 20.8, 20.6

핵심 205 DML(Data Manipulation Language, 데이터 조작어)

2459929

DML은 데이터베이스 사용자가 응용 프로그램이나 질의어를 통하여 저장된 데이터를 실질적으로 처리하는 데 사용되는 언어이다.

- 데이터베이스 사용자와 데이터베이스 관리 시스템 간의 인터페이스를 제공한다.
- DML(데이터 조작어)의 네 가지 유형

명령어	기능
SELECT	테이블에서 조건에 맞는 튜플을 검색함
INSERT	테이블에 새로운 튜플을 삽입함
DELETE	테이블에서 조건에 맞는 튜플을 삭제함
UPDATE	테이블에서 조건에 맞는 튜플의 내용을 변경함

23.7, 22.4, 22.3, 20.8, 20.6

핵심 206 DCL(Data Control Language, 데이터 제어어)

2459930

DCL은 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정의하는 데 사용되는 언어이다.

- 데이터베이스 관리자가 데이터 관리를 목적으로 사용한다.
- DCL(데이터 제어어)의 종류

명령어	기능
COMMIT	명령에 의해 수행된 결과를 실제 물리적 디스크로 저장하고, 데이터베이스 조작 작업이 정상적으로 완료되었음을 관리자에게 알려줌
ROLLBACK	데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래의 상태로 복구함
GRANT	데이터베이스 사용자에게 사용 권한을 부여함
REVOKE	데이터베이스 사용자의 사용 권한을 취소함

23.5, 23.2, 22.3

핵심 207 CREATE TABLE

407204

CREATE TABLE은 테이블을 정의하는 명령문이다.

표기 형식

```
CREATE TABLE 테이블명
        (속성명 데이터_타입 [DEFAULT 기본값] [NOT NULL], …
        [, PRIMARY KEY(기본키_속성명, …)]
        [, UNIQUE(대체키_속성명, …)]
        [, FOREIGN KEY(외래키_속성명, …)]
                [REFERENCES 참조테이블(기본키_속성명, …)]
                [ON DELETE 옵션]
                [ON UPDATE 옵션]
        [, CONSTRAINT 제약조건명] [CHECK (조건식)]);
```

- 기본 테이블에 포함될 모든 속성에 대하여 속성명과 그 속성의 데이터 타입, 기본값, NOT NULL 여부를 지정한다.
- PRIMARY KEY : 기본키로 사용할 속성 또는 속성의 집합을 지정함
- UNIQUE : 대체키로 사용할 속성 또는 속성의 집합을 지정하는 것으로 UNIQUE로 지정한 속성은 중복된 값을 가질 수 없음
- FOREIGN KEY ~ REFERENCES ~
  - 참조할 다른 테이블과 그 테이블을 참조할 때 사용할 외래키 속성을 지정함
  - 외래키가 지정되면 참조 무결성의 CASCADE 법칙이 적용됨
  - ON DELETE 옵션 : 참조 테이블의 튜플이 삭제되었을 때 기본 테이블에 취해야 할 사항을 지정함. 옵션에는 NO ACTION, CASCADE, SET NULL, SET DEFAULT가 있음
  - ON UPDATE 옵션 : 참조 테이블의 참조 속성 값이 변경되었을 때 기본 테이블에 취해야 할 사항을 지정한다. 옵션에는 NO ACTION, CASCADE, SET NULL, SET DEFAULT가 있음
- CONSTRAINT : 제약 조건의 이름을 지정함. 이름을 지정할 필요가 없으면 CHECK절만 사용하여 속성 값에 대한 제약 조건을 명시함
- CHECK : 속성 값에 대한 제약 조건을 정의함

21.3, 20.9

핵심 208 ALTER TABLE

407207

ALTER TABLE은 테이블에 대한 정의를 변경하는 명령문이다.

표기 형식

```
ALTER TABLE 테이블명 ADD 속성명 데이터_타입 [DEFAULT '기본값'];
ALTER TABLE 테이블명 ALTER 속성명 [SET DEFAULT '기본값'];
ALTER TABLE 테이블명 DROP COLUMN 속성명 [CASCADE];
```

- ADD : 새로운 속성(열)을 추가할 때 사용함
- ALTER : 특정 속성의 Default 값을 변경할 때 사용함
- DROP COLUMN : 특정 속성을 삭제할 때 사용함

예제 1 〈학생〉 테이블에 최대 3문자로 구성되는 '학년' 속성 추가하시오.

```
ALTER TABLE 학생 ADD 학년 VARCHAR(3);
```

예제 2 〈학생〉 테이블의 '학번' 필드의 데이터 타입과 크기를 VARCHAR(10)으로 하고 NULL 값이 입력되지 않도록 변경하시오.

```
ALTER TABLE 학생 ALTER 학번 VARCHAR(10) NOT NULL;
```

23.2, 22.3, 21.5, 20.6

핵심 209 DROP

407208

DROP은 스키마, 도메인, 기본 테이블, 뷰 테이블, 인덱스, 제약 조건 등을 제거하는 명령문이다.

표기 형식

```
DROP SCHEMA 스키마명 [CASCADE | RESTRICT];
DROP DOMAIN 도메인명 [CASCADE | RESTRICT];
DROP TABLE 테이블명 [CASCADE | RESTRICT];
DROP VIEW 뷰명 [CASCADE | RESTRICT];
DROP INDEX 인덱스명 [CASCADE | RESTRICT];
DROP CONSTRAINT 제약조건명;
```

- CASCADE : 제거할 요소를 참조하는 다른 모든 개체를 함께 제거함. 즉 주 테이블의 데이터 제거 시 각 외래키와 관계를 맺고 있는 모든 데이터를 제거하는 참조 무결성 제약 조건을 설정하기 위해 사용됨
- RESTRICT : 다른 개체가 제거할 요소를 참조중일 때는 제거를 취소함

예제 〈학생〉 테이블을 제거하되, 〈학생〉 테이블을 참조하는 모든 데이터를 함께 제거하시오.

```
DROP TABLE 학생 CASCADE;
```

핵심 210 DCL(Data Control Language, 데이터 제어어)의 개념

407301

DCL(데이터 제어어)는 데이터의 보안, 무결성, 회복, 병행 제어 등을 정의하는 데 사용하는 언어이다.

- DCL은 데이터베이스 관리자(DBA)가 데이터 관리를 목적으로 사용한다.
- DCL에는 GRANT, REVOKE, COMMIT, ROLLBACK, SAVEPOINT 등이 있다.

22.7, 22.4, 22.3, 20.9

핵심 211 GRANT / REVOKE

407302

데이터베이스 관리자가 데이터베이스 사용자에게 권한을 부여하거나 취소하기 위한 명령어이다.

- GRANT : 권한 부여를 위한 명령어
- REVOKE : 권한 취소를 위한 명령어
- 사용자등급 지정 및 해제

```
– GRANT 사용자등급 TO 사용자_ID_리스트 [IDENTIFIED BY 암호];
– REVOKE 사용자등급 FROM 사용자_ID_리스트;
```

예제 1 사용자 ID가 "NABI"인 사람에게 데이터베이스 및 테이블을 생성할 수 있는 권한을 부여하는 SQL문을 작성하시오.

```
GRANT RESOURCE TO NABI;
```

예제 2 사용자 ID가 "STAR"인 사람에게 단순히 데이터베이스에 있는 정보를 검색할 수 있는 권한을 부여하는 SQL문을 작성하시오.

```
GRANT CONNECT TO STAR;
```

- 테이블 및 속성에 대한 권한 부여 및 취소

```
– GRANT 권한_리스트 ON 개체 TO 사용자 [WITH GRANT OPTION];
– REVOKE [GRANT OPTION FOR] 권한_리스트 ON 개체 FROM 사용자 [CASCADE];
```

- 권한 종류 : ALL, SELECT, INSERT, DELETE, UPDATE, ALTER 등
- WITH GRANT OPTION : 부여받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한을 부여함
- GRANT OPTION FOR : 다른 사용자에게 권한을 부여할 수 있는 권한을 취소함
- CASCADE : 권한 취소 시 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄적으로 취소함

예제 3 사용자 ID가 "NABI"인 사람에게 〈고객〉 테이블에 대한 모든 권한과 다른 사람에게 권한을 부여할 수 있는 권한까지 부여하는 SQL문을 작성하시오.

```
GRANT ALL ON 고객 TO NABI WITH GRANT OPTION;
```

예제 4 사용자 ID가 "STAR"인 사람에게 부여한 〈고객〉 테이블에 대한 권한 중 UPDATE 권한을 다른 사람에게 부여할 수 있는 권한만 취소하는 SQL문을 작성하시오.

```
REVOKE GRANT OPTION FOR UPDATE ON 고객 FROM STAR;
```

22.3

핵심 212 COMMIT

407303

트랜잭션이 성공적으로 끝나면 데이터베이스가 새로운 일관성(Consistency) 상태를 가지기 위해 변경된 모든 내용을 데이터베이스에 반영하여야 하는데, 이때 사용하는 명령이 COMMIT이다.

- COMMIT 명령을 실행하지 않아도 DML문이 성공적으로 완료되면 자동으로 COMMIT되고, DML이 실패하면 자동으로 ROLLBACK이 되도록 Auto Commit 기능을 설정할 수 있다.

22.3, 21.5

핵심 213 ROLLBACK

407304

ROLLBACK은 아직 COMMIT되지 않은 변경된 모든 내용들을 취소하고 데이터베이스를 이전 상태로 되돌리는 명령어이다.

- 트랜잭션 전체가 성공적으로 끝나지 못하면 일부 변경된 내용만 데이터베이스에 반영되는 비일관성(Inconsistency)인 상태를 가질 수 있기 때문에 일부분만 완료된 트랜잭션은 롤백(Rollback)되어야 한다.

23.7, 23.5

핵심 214 삽입문(INSERT INTO~)

407402

삽입문은 기본 테이블에 새로운 튜플을 삽입할 때 사용한다.

일반 형식

```
INSERT INTO 테이블명([속성명1, 속성명2,…])
VALUES (데이터1, 데이터2,… );
```

- 대응하는 속성과 데이터는 개수와 데이터 유형이 일치해야 한다.
- 기본 테이블의 모든 속성을 사용할 때는 속성명을 생략할 수 있다.
- SELECT문을 사용하여 다른 테이블의 검색 결과를 삽입할 수 있다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	성산동	80
황진이	편집	07/21/75	연희동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	망원동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	합정동	110
강호동	인터넷	12/11/80		90

예제 1 〈사원〉 테이블에 (이름 – 홍승현, 부서 – 인터넷)을 삽입하시오.

```
INSERT INTO 사원(이름, 부서) VALUES ('홍승현', '인터넷');
```

예제 2 〈사원〉 테이블에 (장보고, 기획, 05/03/73, 홍제동, 90)을 삽입하시오.

```
INSERT INTO 사원 VALUES ('장보고', '기획', #05/03/73#, '홍제동', 90);
```

예제 3 〈사원〉 테이블에 있는 편집부의 모든 튜플을 편집부원(이름, 생일, 주소, 기본급) 테이블에 삽입하시오.

```
INSERT INTO 편집부원(이름, 생일, 주소, 기본급)
SELECT 이름, 생일, 주소, 기본급
FROM 사원
WHERE 부서 = '편집';
```

23.2, 22.3

핵심 215 삭제문(DELETE FROM~)

407403

삭제문은 기본 테이블에 있는 튜플들 중에서 특정 튜플(행)을 삭제할 때 사용한다.

일반 형식

```
DELETE
FROM 테이블명
[WHERE 조건];
```

- 모든 레코드를 삭제할 때는 WHERE절을 생략한다.
- 모든 레코드를 삭제하더라도 테이블 구조는 남아 있기 때문에 디스크에서 테이블을 완전히 제거하는 DROP과는 다르다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	성산동	80
황진이	편집	07/21/75	연희동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	망원동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	합정동	110
강호동	인터넷	12/11/80		90

예제 〈사원〉 테이블에서 "임꺽정"에 대한 튜플을 삭제하시오.

```
DELETE
FROM 사원
WHERE 이름 = '임꺽정';
```

23.7, 22.3, 20.9

핵심 216 갱신문(UPDATE~ SET~)

407404

갱신문은 기본 테이블에 있는 튜플들 중에서 특정 튜플의 내용을 변경할 때 사용한다.

일반 형식

```
UPDATE 테이블명
SET 속성명 = 데이터[, 속성명=데이터, …]
[WHERE 조건];
```

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	성산동	80
황진이	편집	07/21/75	연희동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	망원동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	합정동	110
강호동	인터넷	12/11/80		90

예제 〈사원〉 테이블에서 "황진이"의 '부서'를 "기획부"로 변경하고 '기본급'을 5만 원 인상시키시오.

```
UPDATE 사원
SET 부서 = '기획', 기본급 = 기본급 + 5
WHERE 이름 = '황진이';
```

407431

핵심 217 데이터 조작문의 네 가지 유형

- SELECT(검색) : SELECT~ FROM~ WHERE~
- INSERT(삽입) : INSERT INTO~ VALUES~
- DELETE(삭제) : DELETE~ FROM~ WHERE~
- UPDATE(변경) : UPDATE~ SET~ WHERE~

22.4, 22.3, 21.5, 20.8, 20.6

407501

핵심 218 SELECT 1 – 일반 형식

```
SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭][, [테이블명.]속성명, …]
[, 그룹함수(속성명) [AS 별칭]]
[, Window함수 OVER (PARTITION BY 속성명1, 속성명2, …
                    ORDER BY 속성명3, 속성명4, …)]
FROM 테이블명[, 테이블명, …]
[WHERE 조건]
[GROUP BY 속성명, 속성명, …]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

- SELECT절
  - PREDICATE : 불러올 튜플 수를 제한할 명령어를 기술함
    - ▶ ALL : 모든 튜플을 검색할 때 지정하는 것으로, 주로 생략함
    - ▶ DISTINCT : 중복된 튜플이 있으면 그 중 첫 번째 한 개만 검색함
    - ▶ DISTINCTROW : 중복된 튜플을 제거하고 한 개만 검색하지만 선택된 속성의 값이 아닌, 튜플 전체를 대상으로 함
  - 속성명 : 검색하여 불러올 속성(열) 또는 속성을 이용한 수식을 지정함
    - ▶ 기본 테이블을 구성하는 모든 속성을 지정할 때는 '*'를 기술함
    - ▶ 두 개 이상의 테이블을 대상으로 검색할 때는 '테이블명.속성명'으로 표현함
  - AS : 속성 및 연산의 이름을 다른 제목으로 표시하기 위해 사용됨
- FROM절 : 질의에 의해 검색될 데이터들을 포함하는 테이블명을 기술함
- WHERE절 : 검색할 조건을 기술함
- ORDER BY절 : 특정 속성을 기준으로 정렬하여 검색할 때 사용함
  - 속성명 : 정렬의 기준이 되는 속성명을 기술함
  - [ASC | DESC] : 정렬 방식으로서 'ASC'는 오름차순, 'DESC'는 내림차순. 생략하면 오름차순으로 지정됨

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트 댄스	5
일지매	태껸	15
임꺽정	씨름	8

예제 1 〈사원〉 테이블에서 '주소'만 검색하되 같은 '주소'는 한 번만 출력하시오.

```
SELECT DISTINCT 주소
FROM 사원;
```

〈결과〉

주소
대흥동
망원동
상암동
서교동
연남동
합정동

예제 2 〈사원〉 테이블에서 "기획" 부서에 근무하면서 "대흥동"에 사는 사람의 튜플을 검색하시오.

```
SELECT *
FROM 사원
WHERE 부서 = '기획' AND 주소 = '대흥동';
```

〈결과〉

이름	부서	생일	주소	기본급
성춘향	기획	02/20/64	대흥동	100

21.8

핵심 219 조건 연산자 / 연산자 우선순위

407531

조건 연산자

- 비교 연산자

연산자	의미
=	같다
〈 〉	같지 않다
〉	크다
〈	작다
〉=	크거나 같다
〈=	작거나 같다

- 논리 연산자 : NOT, AND, OR
- LIKE 연산자 : 대표 문자를 이용해 지정된 속성의 값이 문자 패턴과 일치하는 튜플을 검색하기 위해 사용됨

대표 문자	의미
%	모든 문자를 대표함
_	문자 하나를 대표함
#	숫자 하나를 대표함

연산자 우선순위

종류	연산자	우선순위
산술 연산자	×, /, +, −	왼쪽에서 오른쪽으로 갈수록 낮아짐
관계 연산자	=, 〈 〉, 〉, 〉=, 〈, 〈=	모두 같음
논리 연산자	NOT, AND, OR	왼쪽에서 오른쪽으로 갈수록 낮아짐

※ 산술, 관계, 논리 연산자가 함께 사용되었을 때는 산술 〉 관계 〉 논리 연산자 순서로 연산자 우선순위가 정해집니다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트 댄스	5
일지매	태껸	15
임꺽정	씨름	8

예제 〈사원〉 테이블에서 성이 '김'인 사람의 튜플을 검색하시오.

```
SELECT *
FROM 사원
WHERE 이름 LIKE "김%";
```

〈결과〉

이름	부서	생일	주소	기본급
김선달	편집	10/22/73	망원동	90

23.5, 22.4, 21.3, 20.9, 20.6

핵심 220 하위 질의

407505

하위 질의는 조건절에 주어진 질의를 먼저 수행하여 그 검색 결과를 조건절의 피연산자로 사용한다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트 댄스	5
일지매	태껸	15
임꺽정	씨름	8

예제 1 '취미'가 "나이트댄스"인 사원의 '이름'과 '주소'를 검색하시오.

```
SELECT 이름, 주소
FROM 사원
WHERE 이름 = (SELECT 이름 FROM 여가활동 WHERE 취미 =
              '나이트댄스') ;
```

〈결과〉

이름	주소
성춘향	대흥동

예제 2 취미활동을 하는 사원들의 부서를 검색하시오.

```
SELECT 부서
FROM 사원
WHERE EXISTS (SELECT 이름 FROM 여가활동 WHERE 여가활
              동.이름 = 사원.이름);
```

〈결과〉

부서
인터넷
편집
기획
기획

218

407601

핵심 221 SELECT 2 – 일반 형식

```
SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭][, [테이블명.]속
       성명, …]
[, 그룹함수(속성명) [AS 별칭]]
[, WINDOW함수 OVER (PARTITION BY 속성명1, 속성명2, …
                    ORDER BY 속성명3, 속성명4, …) [AS 별칭]]
FROM 테이블명[, 테이블명, …]
[WHERE 조건]
[GROUP BY 속성명, 속성명, …]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

- **그룹함수** : GROUP BY절에 지정된 그룹별로 속성의 값을 집계할 함수를 기술함
- **WINDOW 함수** : GROUP BY절을 이용하지 않고 속성의 값을 집계할 함수를 기술함
  - PARTITION BY : WINDOW 함수가 적용될 범위로 사용할 속성을 지정함
  - ORDER BY : PARTITION 안에서 정렬 기준으로 사용할 속성을 지정함
- **GROUP BY절** : 특정 속성을 기준으로 그룹화하여 검색할 때 사용함. 일반적으로 GROUP BY절은 그룹 함수와 함께 사용됨
- **HAVING절** : GROUP BY와 함께 사용되며, 그룹에 대한 조건을 지정함

〈상여금〉

부서	이름	상여내역	상여금
기획	홍길동	연장근무	100
기획	일지매	연장근무	100
기획	최준호	야간근무	120
기획	장길산	특별근무	90
인터넷	강건달	특별근무	90
인터넷	서국현	특별근무	90
인터넷	박인식	연장근무	30
편집	김선달	특별근무	80
편집	황종근	연장근무	40
편집	성춘향	야간근무	80
편집	임꺽정	야간근무	80
편집	황진이	야간근무	50

예제 〈상여금〉 테이블에서 '상여금'이 100 이상인 사원이 2명 이상인 '부서'의 튜플 수를 구하시오.

```
SELECT 부서, COUNT(*) AS 사원수
FROM 상여금
WHERE 상여금 >= 100
GROUP BY 부서
HAVING COUNT(*) >= 2;
```

〈결과〉

부서	사원수
기획	3

20.8

핵심 222 그룹 함수

407631

GROUP BY절에 지정된 그룹별로 속성의 값을 집계할 때 사용된다.

- COUNT(속성명) : 그룹별 튜플 수를 구하는 함수
- SUM(속성명) : 그룹별 합계를 구하는 함수
- AVG(속성명) : 그룹별 평균을 구하는 함수
- MAX(속성명) : 그룹별 최대값을 구하는 함수
- MIN(속성명) : 그룹별 최소값을 구하는 함수
- STDDEV(속성명) : 그룹별 표준편차를 구하는 함수
- VARIANCE(속성명) : 그룹별 분산을 구하는 함수

23.5, 23.2, 22.3, 21.5

핵심 223 집합 연산자를 이용한 통합 질의

407604

집합 연산자를 사용하여 2개 이상의 테이블의 데이터를 하나로 통합한다.

표기 형식

```
SELECT 속성명1, 속성명2, …
FROM 테이블명
UNION | UNION ALL | INTERSECT | EXCEPT
SELECT 속성명1, 속성명2, …
FROM 테이블명
[ORDER BY 속성명 [ASC | DESC]];
```

- 두 개의 SELECT문에 기술한 속성들은 개수와 데이터 유형이 서로 동일해야 한다.
- 집합 연산자의 종류(통합 질의의 종류)

UNION	• 두 SELECT문의 조회 결과를 통합하여 모두 출력함 • 중복된 행은 한 번만 출력함
UNION ALL	• 두 SELECT문의 조회 결과를 통합하여 모두 출력함 • 중복된 행도 그대로 출력함
INTERSECT	두 SELECT문의 조회 결과 중 공통된 행만 출력함
EXCEPT	첫 번째 SELECT문의 조회 결과에서 두 번째 SELECT문의 조회 결과를 제외한 행을 출력함

〈사원〉

사원	직급
김형석	대리
홍영선	과장
류기선	부장
김현천	이사

〈직원〉

사원	직급
신원섭	이사
이성호	대리
홍영선	과장
류기선	부장

예제1 〈사원〉 테이블과 〈직원〉 테이블을 통합하는 질의문을 작성하시오. (단, 같은 레코드가 중복되어 나오지 않게 하시오.)

```
SELECT *
FROM 사원
UNION
SELECT *
FROM 직원;
```

〈결과〉

사원	직급
김현천	이사
김형석	대리
류기선	부장
신원섭	이사
이성호	대리
홍영선	과장

예제2 〈사원〉 테이블과 〈직원〉 테이블에 공통으로 존재하는 레코드만 통합하는 질의문을 작성하시오.

```
SELECT *
FROM 사원
INTERSECT
SELECT *
FROM 직원;
```

〈결과〉

사원	직급
류기선	부장
홍영선	과장

21.5

핵심 224 INNER JOIN

1412302

INNER JOIN은 일반적으로 EQUI JOIN과 NON-EQUI JOIN으로 구분된다.

- 조건이 없는 INNER JOIN을 수행하면 CROSS JOIN과 동일한 결과를 얻을 수 있다.
- EQUI JOIN
  - EQUI JOIN은 JOIN 대상 테이블에서 공통 속성을 기준으로 '='(equal) 비교에 의해 같은 값을 가지는 행을 연결하여 결과를 생성하는 JOIN 방법이다.
  - WHERE절을 이용한 EQUI JOIN의 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1, 테이블명2, …
WHERE 테이블명1.속성명 = 테이블명2.속성명;
```

  - NATURAL JOIN절을 이용한 EQUI JOIN의 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1 NATURAL JOIN 테이블명2;
```

  - JOIN ~ USING절을 이용한 EQUI JOIN의 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1 JOIN 테이블명2 USING(속성명);
```

〈학생〉

학번	이름	학과코드	선배	성적
15	고길동	com		83
16	이순신	han		96
17	김선달	com	15	95
19	아무개	han	16	75
37	박치민		17	55

〈학과〉

학과코드	학과명
com	컴퓨터
han	국어
eng	영어

〈성적등급〉

등급	최저	최고
A	90	100
B	80	89
C	60	79
D	0	59

예제 〈학생〉 테이블과 〈학과〉 테이블에서 '학과코드' 값이 같은 튜플을 JOIN하여 '학번', '이름', '학과코드', '학과명'을 출력하는 SQL문을 작성하시오.

- SELECT 학번, 이름, 학생.학과코드, 학과명
  FROM 학생, 학과
  WHERE 학생.학과코드 = 학과.학과코드;
- SELECT 학번, 이름, 학생.학과코드, 학과명
  FROM 학생 NATURAL JOIN 학과;
- SELECT 학번, 이름, 학생.학과코드, 학과명
  FROM 학생 JOIN 학과 USING(학과코드);

〈결과〉

학번	이름	학과코드	학과명
15	고길동	com	컴퓨터
16	이순신	han	국어
17	김선달	com	컴퓨터
19	아무개	han	국어

22.7, 20.6

핵심 225 트리거(Trigger)의 개요

2411501

트리거는 데이터베이스 시스템에서 데이터의 삽입(Insert), 갱신(Update), 삭제(Delete) 등의 이벤트(Event)가 발생할 때마다 관련 작업이 자동으로 수행되는 절차형 SQL이다.

- 트리거는 데이터베이스에 저장되며, 데이터 변경 및 무결성 유지, 로그 메시지 출력 등의 목적으로 사용된다.
- 트리거의 구문에는 DCL(데이터 제어어)을 사용할 수 없으며, DCL이 포함된 프로시저나 함수를 호출하는 경우에도 오류가 발생한다.
- 트리거에 오류가 있는 경우 트리거가 처리하는 데이터에도 영향을 미치므로 트리거를 생성할 때 세심한 주의가 필요하다.

핵심 226 DBMS 접속 기술

2411702

DBMS 접속 기술은 DBMS에 접근하기 위해 사용하는 API 또는 API의 사용을 편리하게 도와주는 프레임워크 등을 의미한다.

JDBC(Java DataBase Connectivity)

- JDBC는 Java 언어로 다양한 종류의 데이터베이스에 접속하고 SQL문을 수행할 때 사용되는 표준 API이다.
- 1997년 2월 썬 마이크로시스템에서 출시했다.
- 접속하려는 DBMS에 대한 드라이버가 필요하다.

ODBC(Open DataBase Connectivity)

- ODBC는 데이터베이스에 접근하기 위한 표준 개방형 API로, 개발 언어에 관계없이 사용할 수 있다.
- 1992년 9월 마이크로소프트에서 출시했다.
- ODBC도 접속하려는 DBMS에 맞는 드라이버가 필요하지만, 접속하려는 DBMS의 인터페이스를 알지 못하더라도 ODBC 문장을 사용하여 SQL을 작성하면 ODBC에 포함된 드라이버 관리자가 해당 DBMS의 인터페이스에 맞게 연결해 주므로 DBMS의 종류를 몰라도 된다.

MyBatis

- MyBatis는 JDBC 코드를 단순화하여 사용할 수 있는 SQL Mapping 기반 오픈 소스 접속 프레임워크이다.
- JDBC로 데이터베이스에 접속하려면 다양한 메소드를 호출하고 해제해야 하는데, MyBatis는 이를 간소화 했고 접속 기능을 더욱 강화하였다.
- MyBatis는 SQL 문장을 분리하여 XML 파일을 만들고, Mapping을 통해 SQL을 실행한다.
- MyBatis는 SQL을 거의 그대로 사용할 수 있어 SQL 친화적인 국내 환경에 적합하여 많이 사용된다.

핵심 227 ORM(Object-Relational Mapping)의 개요

2411901

ORM은 객체지향 프로그래밍의 객체(Object)와 관계형 데이터베이스(Relational Database)의 데이터를 연결(Mapping)하는 기술을 의미한다.

- ORM은 객체지향 프로그래밍에서 사용할 수 있는 가상의 객체지향 데이터베이스를 만들어 프로그래밍 코드와 데이터를 연결한다.
- ORM으로 생성된 가상의 객체지향 데이터베이스는 프로그래밍 코드 또는 데이터베이스와 독립적이므로 재사용 및 유지보수가 용이하다.
- ORM은 SQL 코드를 직접 입력하지 않고 선언문이나 할당 같은 부수적인 코드가 생략되기 때문에 직관적이고 간단하게 데이터를 조작할 수 있다.

핵심 228 쿼리 성능 최적화의 개요

2459931

문쿼리 성능 최적화는 데이터 입 · 출력 애플리케이션의 성능 향상을 위해 SQL 코드를 최적화하는 것이다.

- 쿼리 성능을 최적화하기 전에 성능 측정 도구인 APM을 사용하여 최적화 할 쿼리를 선정해야 한다.
- 최적화 할 쿼리에 대해 옵티마이저가 수립한 실행 계획을 검토하고 SQL 코드와 인덱스를 재구성한다.

※ RBO vs CBO

RBO(Rule Based Optimizer)는 규칙 기반 옵티마이저이고, CBO(Cost Based Optimizer)는 비용 기반 옵티마이저로서 다음과 같은 차이점이 있다.

	RBO	CBO
최적화 기준	규칙에 정의된 우선순위	액세스 비용
성능 기준	개발자의 SQL 숙련도	옵티마이저의 예측 성능
특징	실행 계획 예측이 쉬움	성능 통계치 정보 활용, 예측이 복잡함
고려사항	개발자의 규칙 이해도, 규칙의 효율성	비용 산출 공식의 정확성

핵심 229 데이터 전환의 정의

2412101

데이터 전환이란 운영 중인 기존 정보 시스템에 축적되어 있는 데이터를 추출(Extraction)하여 새로 개발할 정보 시스템에서 운영 가능하도록 변환(Transformation)한 후, 적재(Loading)하는 일련의 과정을 말한다.

- 데이터 전환을 ETL(Extraction, Transformation, Load), 즉 추출, 변환, 적재 과정이라고 한다.
- 데이터 전환을 데이터 이행(Data Migration) 또는 데이터 이관이라고도 한다.

2412401

핵심 230 데이터 검증

데이터 검증이란 원천 시스템의 데이터를 목적 시스템의 데이터로 전환하는 과정이 정상적으로 수행되었는지 여부를 확인하는 과정을 말한다.

- 데이터 전환 검증은 검증 방법과 검증 단계에 따라 분류할 수 있다.

2412504

핵심 231 오류 데이터 정제

오류 데이터 정제는 오류 관리 목록의 각 항목을 분석하여 원천 데이터를 정제하거나 전환 프로그램을 수정하는 것이다.

오류 데이터 분석

- 오류 관리 목록의 오류 데이터를 분석하여 오류 상태, 심각도, 해결 방안을 확인 및 기재한다.
- 상태

Open	오류가 보고만 되고 분석되지 않은 상태
Assigned	오류의 영향 분석 및 수정을 위해 개발자에게 오류를 전달한 상태
Fixed	개발자가 오류를 수정한 상태
Closed	수정된 오류에 대해 테스트를 다시 했을 때 오류가 발견되지 않은 상태
Deferred	오류 수정을 연기한 상태
Classified	보고된 오류를 관련자들이 확인했을 때 오류가 아니라고 확인된 상태

4과목 프로그래밍 언어 활용

20.8

2413001

핵심 232 배치 프로그램

배치 프로그램은 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하는 것을 의미한다.

- 배치 프로그램이 갖추어야 하는 필수 요소는 다음과 같다.

대용량 데이터	대량의 데이터를 가져오거나, 전달하거나, 계산하는 등의 처리가 가능해야 함
자동화	심각한 오류가 발생하는 상황을 제외하고는 사용자의 개입 없이 수행되어야 함
견고성	잘못된 데이터나 데이터 중복 등의 상황으로 중단되는 일 없이 수행되어야 함
안정성/신뢰성	오류가 발생하면 오류의 발생 위치, 시간 등을 추적할 수 있어야 함
성능	다른 응용 프로그램의 수행을 방해하지 않아야 하고, 지정된 시간 내에 처리가 완료되어야 함

23.5, 23.2, 20.8

400102

핵심 233 C/C++의 데이터 타입 크기 및 기억 범위

종류	데이터 타입	크기
문자	char	1Byte
부호없는 문자형	unsigned char	1Byte
정수	short	2Byte
	int	4Byte
	long	4Byte
	long long	8Byte
실수	float	4Byte
	double	8Byte
	long double	8Byte