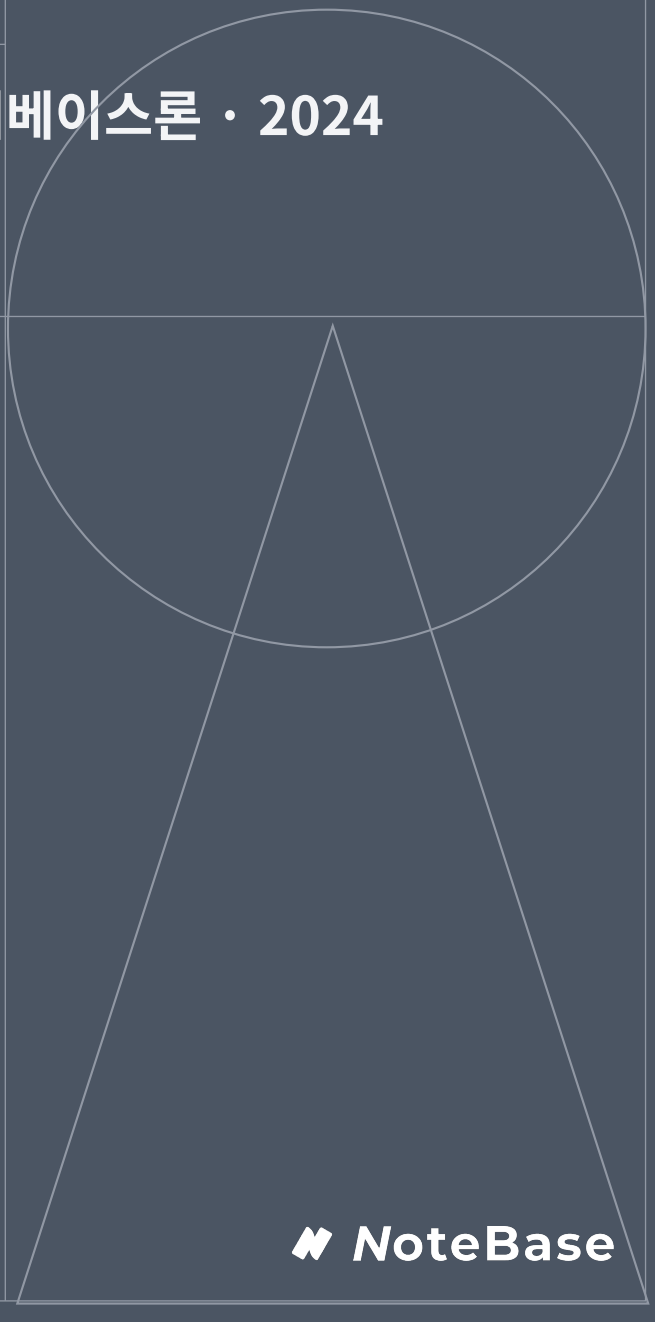
기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2024



NoteBase

기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2024

공무원/서울시/7급 · 데이터베이스론 · 2024

문 1. DBMS 언어에 대한 설명으로 가장 옳지 않은 것은?

① 데이터 조작어(DML)는 고수준 데이터 조작어와 저수준 데이터 조작어로 구분할 수 있다.

② 저장구조 정의어(SDL)를 사용하여 내부 스키마를 표현(지정)할 수 있다.

③ SQL은 데이터 정의 기능, 데이터 조작 기능, 데이터 제어 기능을 모두 지원한다.

④ SQL은 한 번에 한 레코드(one-record-at-a-time) 조작어에 해당한다.

정답 4번

출제 포인트

이 문제는 DBMS 언어의 종류와 SQL의 처리 방식이 집합 단위인지 레코드 단위인지를 구별할 수 있는지를 묻는다.

핵심 개념

DML은 데이터를 검색, 삽입, 삭제, 갱신하는 언어이며 고수준 DML과 저수준 DML로 나눌 수 있다.
SDL은 저장 구조와 접근 경로를 정의하여 내부 스키마를 기술하는 언어이다.
SQL은 DDL, DML, DCL 기능을 포함하는 비절차적 집합 지향 언어이다.

선지별 해설

① O: DML은 데이터 조작을 위한 언어이며 사용자가 원하는 결과를 중심으로 기술하는 고수준 DML과 레코드 단위 처리에 가까운 저수준 DML로 구분할 수 있다.
② O: SDL은 저장 레코드의 형식, 인덱스, 접근 경로 등 물리적 저장 구조를 정의하므로 내부 스키마를 표현할 수 있다.
③ O: SQL은 CREATE, ALTER, DROP 같은 데이터 정의 기능과 SELECT, INSERT, UPDATE, DELETE 같은 데이터 조작 기능, GRANT, REVOKE 같은 데이터 제어 기능을 모두 지원한다.
④ X: 틀린 부분은 "한 번에 한 레코드(one-record-at-a-time) 조작어"라는 표현이다. SQL은 개별 레코드를 하나씩 절차적으로 처리하는 언어가 아니라 여러 튜플의 집합을 한 번에 다루는 집합 지향 언어이다. 올바르게는 "SQL은 한 번에 한 집합(set-at-a-time)을 대상으로 조작하는 언어"이다.

NoteBase

오답 포인트

④에서 one-record-at-a-time이라는 표현을 보고 DML 일반의 설명으로 받아들이면 틀리기 쉽다. 저수준 DML은 레코드 단위 처리에 가깝지만 SQL은 대표적인 고수준 비절차적 집합 지향 언어라는 점이 함정이다.

다시 나오면 이렇게 판단

DBMS 언어 문제는 먼저 언어가 다루는 층위와 처리 단위를 본다. 내부 스키마를 묻으면 SDL인지 확인하고, SQL의 성격을 묻으면 set-at-a-time인지 one-record-at-a-time인지로 바로 판단한다.

암기 포인트

SDL은 내부 스키마, SQL은 set-at-a-time, record-at-a-time은 저수준 DML에 가깝다.

NoteBase

문 2. <보기>에서 설명하는 데이터베이스 시스템을 위한 아키텍처로 가장 옳은 것은?

<보기>
• 많은 웹 응용에서 활용
• 클라이언트-웹 서버-데이터베이스 서버 구조
• 웹 서버는 클라이언트의 요청을 데이터베이스 서버로 전달하기 전에 보안과 관련된 기능이나 기타 역할 수행

① 중앙집중식
② 2-층(2-tier) 클라이언트-서버
③ 3-층(3-tier) 클라이언트-서버
④ P_2P(Peer to Peer)

정답 3번

출제 포인트

이 문제는 데이터베이스 시스템 아키텍처에서 중앙집중식, 2-tier, 3-tier, P2P를 구분할 수 있는지를 묻는 문제이다.

핵심 개념

중앙집중식은 하나의 중앙 시스템에 처리가 집중되는 구조이다.
2-tier는 클라이언트와 데이터베이스 서버가 직접 연결되는 구조이다.
3-tier는 클라이언트, 웹 또는 응용 서버, 데이터베이스 서버로 계층이 분리되는 구조이다.
P2P는 각 노드가 대등하게 자원을 주고받는 구조로 중앙 서버 계층 분리를 전제로 하지 않는다.

선지별 해설

1번 X: 중앙집중식은 처리와 자원이 중앙 컴퓨터에 집중되는 형태이므로 클라이언트-웹 서버-데이터베이스 서버처럼 계층이 나뉜 구조와 다르다. 올바르게는 보기는 중간에 웹 서버가 있는 다계층 구조이다.
2번 X: 2-tier 클라이언트-서버는 보통 클라이언트가 데이터베이스 서버와 직접 통신하는 구조이다. 보기처럼 웹 서버가 별도 계층으로 존재하여 요청을 전달하기 전 보안 기능이나 기타 역할을 수행하면 2-tier가 아니라 3-tier이다.
3번 O: 보기에는 클라이언트-웹 서버-데이터베이스 서버 구조라고 되어 있고, 웹 서버가 요청 중개와 보안 관련 기능을 수행한다고 하였다. 이는 프레젠테이션 계층, 응용 계층, 데이터 계층이 분리된 3-tier 구조의 전형적인 설명이다.
4번 X: P2P는 참여 노드들이 서로 대등한 위치에서 직접 자원을 공유하는 구조이다. 보기처럼 웹 서버와 데이터베이스 서버가 분리되어 클라이언트 요청을 중개하는 서버 중심 구조는 P2P가 아니다. 올바르게는 3-tier 클라이언트-서버 구조이다.

NoteBase

오답 포인트

2번과 3번이 모두 클라이언트-서버라는 표현을 쓰기 때문에 헷갈리기 쉽다. 이 문제의 핵심은 서버가 있다는 사실이 아니라 웹 서버가 독립된 중간 계층으로 존재하는지 여부이다.

다시 나오면 이렇게 판단

아키텍처 문제는 먼저 계층 수를 센다. 그다음 클라이언트와 데이터베이스 사이에 웹 서버나 응용 서버가 별도로 있으면 3-tier로 판단하고, 클라이언트가 데이터베이스와 직접 연결되면 2-tier로 판단한다.

암기 포인트

클라이언트-웹 서버-데이터베이스 서버는 3-tier이고, 중간 웹 또는 응용 서버가 보이면 3-tier로 정리한다.

NoteBase

문 3. RAID(Redundant Array of Independent Disks) 구조에 대한 설명으로 가장 옳지 않은 것은?

① 디스크에 대한 접근 성능과 신뢰성을 높일 수 있다.

② RAID 내의 각 디스크는 각각 독립적으로 동작할 수 있다.

③ 더 빠른 전송속도를 제공하기 위해 데이터 스트라이핑 (data striping) 기법을 사용한다.

④ RAID 레벨 0은 장애 발생에 대비하여 별도의 디스크에 중복 쓰기를 수행한다.

정답 4번

출제 포인트

이 문제는 RAID의 기본 목적과 RAID 0의 스트라이핑, RAID 1의 중복 저장을 구별하는지를 묻는 문제이다.

핵심 개념

RAID는 여러 개의 독립 디스크를 배열로 묶어 성능 향상이나 신뢰성 향상을 얻는 저장 방식이다. RAID 0은 데이터를 여러 디스크에 분산 저장하는 스트라이핑 중심 구조이므로 속도 향상에는 유리하지만 중복성이 없다. RAID 1은 같은 데이터를 다른 디스크에 그대로 복제하는 미러링 구조이므로 장애 대비가 가능하다.

선지별 해설

① O: RAID는 여러 디스크를 병렬로 활용하므로 접근 성능을 높일 수 있고, RAID 1·5처럼 중복 정보를 두는 방식은 신뢰성도 높일 수 있다.
② O: RAID는 여러 개의 독립 디스크를 묶어 사용하는 구조이므로 각 디스크는 물리적으로 독립 장치이다. 다만 논리적으로는 하나의 배열처럼 동작한다.
③ O: 데이터 스트라이핑은 데이터를 여러 디스크에 나누어 기록해 병렬 입출력을 가능하게 하므로 전송속도 향상에 사용된다.
④ X: 틀린 부분은 'RAID 레벨 0은 장애 발생에 대비하여 별도의 디스크에 중복 쓰기를 수행한다'는 표현이다. RAID 0은 중복 저장 없이 스트라이핑만 수행한다. 장애 대비를 위한 중복 쓰기는 RAID 1의 미러링이나 다른 중복 RAID 수준의 특징이다.

오답 포인트

RAID의 공통 목적과 각 RAID 레벨의 개별 특성을 섞어 놓은 점이 함정이다. 특히 '속도 향상'과 '중복 저장'을 둘 다 RAID라고만 외우면 RAID 0에도 장애 대비 기능이 있다고 오인하기 쉽다.

다시 나오면 이렇게 판단

먼저 해당 선지가 RAID 전체의 일반 목적을 말하는지, 특정 RAID 레벨의 개별 특성을 말하는지 구분하면 된다. 그다음 RAID 0은 스트라이핑 전용, RAID 1은 미러링 전용이라는 기준으로 판단하면 빠르게 정리된다.

NoteBase

암기 포인트

RAID 0은 속도 중심의 스트라이핑이고 중복이 없으며, RAID 1은 복제 중심의 미러링이다.

NoteBase

문 4. <보기>와 같은 순서로 키(key)값을 입력하여, B-트리(B-tree)를 구성하고자 한다. (가) 차수(degree)가 3인 B-트리와 (나) 차수가 5인 B-트리의 루트 노드(root node)에 있는 키값을 옳게 짝지은 것은?

<보기>
키 입력순서: 5, 7, 3, 6, 1, 8, 2

① (가): 3, (나): 5
② (가): 3, (나): 6
③ (가): 5, (나): 5
④ (가): 5, (나): 6

정답 3번

출제 포인트

이 문제는 B-트리의 차수에 따라 한 노드에 들어갈 수 있는 최대 키 수가 달라지고, 키를 순서대로 삽입할 때 루트가 언제 분할되는지를 추적할 수 있는지를 묻는다.

올바른 순서

(가) 차수 3 B-트리는 한 노드가 최대 2개의 키를 가진다. 5, 7까지는 루트에 그대로 들어가고, 3을 넣으면 [3,5,7]이 되어 중간값 5가 루트로 올라간다. 이후 6, 1을 넣으면 각 리프가 [6,7], [1,3]이 된다. 8을 넣으면 오른쪽 리프 [6,7,8]이 분할되어 7이 루트로 올라가 루트가 [5,7]이 된다. 마지막으로 2를 넣으면 왼쪽 리프 [1,2,3]이 분할되어 2가 루트로 올라가고, 루트 [2,5,7]이 다시 분할되어 최종 루트는 5가 된다.
(나) 차수 5 B-트리는 한 노드가 최대 4개의 키를 가진다. 5, 7, 3, 6까지는 루트 하나에 유지된다. 1을 넣으면 [1,3,5,6,7]이 되어 중간값 5가 루트가 되고 좌우 자식은 [1,3], [6,7]이 된다. 이후 8은 오른쪽 자식에, 2는 왼쪽 자식에 들어가도 추가 분할이 없으므로 최종 루트는 계속 5이다.
왜 이 순서: B-트리 삽입은 먼저 리프에 키를 넣고, 허용 키 수를 넘는 순간 중간값이 부모로 승격된다. 최종 루트는 이 승격이 루트까지 전파되는지 여부로 결정된다.

NoteBase

선지별 해설

1번: X. (가)를 3으로 본 것이 틀리다. 차수 3에서는 마지막 2 삽입 때 왼쪽 리프 분할로 2가 루트로 올라가 루트가 [2,5,7]이 되고, 이 루트가 다시 분할되어 최종 루트는 5가 된다. 올바르게는 (가)는 5이다. (나)는 5로 맞다.
2번: X. (가)를 3으로 본 것이 틀리고, (나)를 6으로 본 것도 틀리다. 차수 5에서는 [1,3,5,6,7] 분할 시 중간값 5가 루트가 된 뒤 추가 분할이 없으므로 최종 루트는 6이 아니라 5이다. 올바르게는 (가)는 5이고 (나)도 5이다.
3번: O. (가)는 차수 3에서 분할이 루트까지 한 번 더 전파되어 최종 루트가 5가 되고, (나)는 차수 5에서 처음 분할 때 올라간 5가 끝까지 루트로 유지된다.
4번: X. (가)는 5로 맞지만 (나)를 6으로 본 것이 틀리다. 차수 5에서는 6이 오른쪽 자식의 키일 뿐 루트로 승격되지 않는다. 올바르게는 (나)는 6이 아니라 5이다.

오답 포인트

차수 3과 차수 5의 최대 키 수를 같은 방식으로 보거나, 리프 분할에서 부모로 올라가는 값이 최댓값이라고 착각하면 6이나 3을 고르기 쉽다. 이 문제의 함정은 중간값 승격과 루트 재분할 여부를 끝까지 보지 않는 데 있다.

다시 나오면 이렇게 판단

B-트리 삽입 문제는 차수에서 최대 키 수를 먼저 정한 뒤, 키를 리프에 넣고 초과 시 중간값이 부모로 올라간다는 규칙만 따라가면 된다. 각 삽입마다 분할이 발생했는지와 그 승격이 루트까지 전파되었는지를 확인하면 루트 키를 판단할 수 있다.

암기 포인트

B-트리 분할의 핵심은 중간값 승격이며, 최종 루트는 마지막에 루트까지 승격이 전파되었는지로 결정된다.

NoteBase

문 5. <보기>의 데이터베이스 설계 단계를 순서대로 가장 바르게 나열한 것은?

<보기>
ㄱ. 핵심 정보 개체, 제한 조건 등 특정 데이터베이스 관리 시스템에 독립적이고 추상화된 개체관계 모델 생성
ㄴ. 데이터베이스를 활용하게 되는 도메인, 운영 환경, 처리 조건 등을 분석하여 데이터베이스가 제공해야 하는 핵심적인 정보 대상 및 제한 조건 등 도출
ㄷ. 관계형 데이터 모델과 같이 논리적 데이터베이스 구조에 맞는 테이블 스키마 생성
ㄹ. 논리적 데이터베이스 구조로부터 효율적이고 구현 가능한 물리적 데이터베이스 구조 설계

① ㄱ - ㄴ - ㄷ - ㄹ
② ㄱ - ㄴ - ㄹ - ㄷ
③ ㄴ - ㄱ - ㄷ - ㄹ
④ ㄴ - ㄱ - ㄹ - ㄷ

정답 3번

출제 포인트

이 문제는 데이터베이스 설계의 절차를 요구사항 분석, 개념 설계, 논리 설계, 물리 설계의 순서로 구분하는지를 묻는 문제이다.

올바른 순서

1. ㄴ — 데이터베이스를 사용할 업무 도메인, 운영 환경, 처리 조건을 분석하여 필요한 정보와 제약 조건을 도출하는 요구사항 분석 단계이다.
2. ㄱ — 특정 DBMS와 무관하게 핵심 개체와 관계를 ER 모델로 추상화하는 개념 설계 단계이다.
3. ㄷ — 개념 설계 결과를 관계형 모델의 테이블 스키마처럼 논리적 구조로 바꾸는 논리 설계 단계이다.
4. ㄹ — 논리 구조를 실제 저장 구조와 접근 경로 중심으로 구현 가능하게 만드는 물리 설계 단계이다.

왜 이 순서: 요구사항이 먼저 정리되어야 개체와 관계를 모델링할 수 있고, 개념 모델이 있어야 논리 스키마를 만들 수 있으며, 논리 스키마가 확정되어야 물리 저장 구조를 설계할 수 있다.

NoteBase

선지별 해설

1번 X: ㄱ을 ㄴ보다 앞에 두어 개념 설계를 요구사항 분석보다 먼저 배치한 점이 틀렸다. 개념 설계는 요구사항에서 추출한 정보 대상과 제약 조건을 바탕으로 수행된다. 틀린 표현은 'ㄱ-ㄴ'의 선후관계이고, 올바르게는 'ㄴ-ㄱ-ㄷ-ㄹ'이다.
2번 X: ㄹ을 ㄷ보다 앞에 두어 물리 설계를 논리 설계보다 먼저 배치한 점이 틀렸다. 물리 설계는 논리적 테이블 스키마가 정리된 뒤 그 구조를 저장 방식과 접근 경로로 구체화하는 단계이다. 틀린 표현은 'ㄹ-ㄷ'의 선후관계이고, 올바르게는 'ㄴ-ㄱ-ㄷ-ㄹ'이다.
3번 O: ㄴ-ㄱ-ㄷ-ㄹ은 요구사항 분석, 개념 설계, 논리 설계, 물리 설계의 표준 흐름과 일치한다. 요구사항에서 출발해 ER 모델을 만들고, 이를 관계 스키마로 변환한 뒤 실제 저장 구조를 설계하므로 순서가 맞다.
4번 X: ㄴ-ㄱ까지는 맞지만 ㄹ을 ㄷ보다 먼저 두어 논리 설계를 건너뛴 점이 틀렸다. 물리 설계는 논리적 데이터베이스 구조를 입력으로 삼는다. 틀린 표현은 'ㄹ-ㄷ'의 선후관계이고, 올바르게는 'ㄴ-ㄱ-ㄷ-ㄹ'이다.

오답 포인트

ㄱ과 ㄴ을 모두 초기 단계로 보고 순서를 바꾸거나, ㄷ과 ㄹ을 모두 구현 단계로 보고 순서를 바꾸기 쉽다. 그러나 ㄱ은 요구사항을 모델로 추상화한 결과물이므로 ㄴ 뒤에 와야 하고, ㄹ은 논리 스키마를 실제 저장 구조로 옮기는 단계이므로 ㄷ 뒤에 와야 한다.

다시 나오면 이렇게 판단

데이터베이스 설계 순서 문제는 먼저 무엇을 알아내는 단계인지, 그다음 무엇을 모델로 표현하는 단계인지, 이후 무엇을 스키마로 변환하는 단계인지, 마지막으로 무엇을 저장 구조로 구현하는 단계인지의 흐름으로 판단한다.

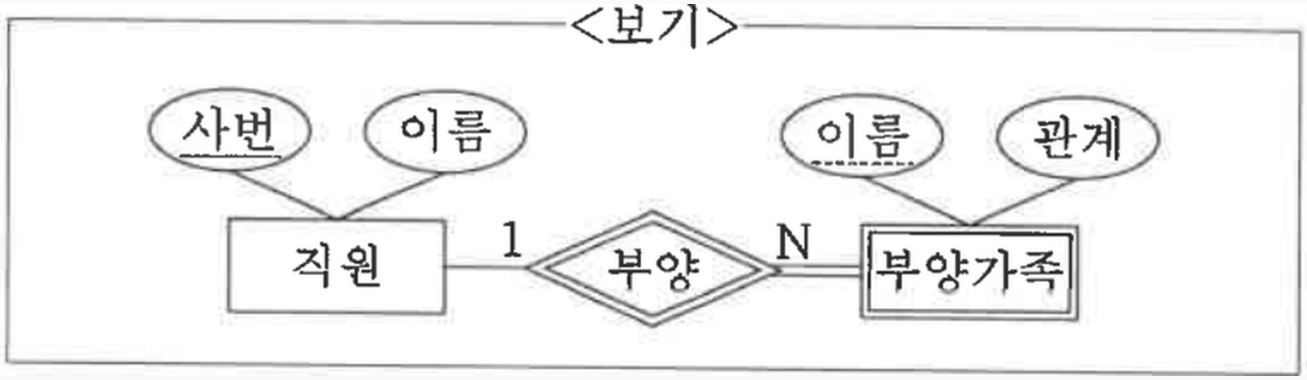
암기 포인트

요구사항 분석 → ER 개념 모델 → 논리 스키마 → 물리 저장 구조의 흐름으로 기억하면 된다.

NoteBase

문 6. <보기>의 개체-관계 다이어그램(E-R Diagram)에 대한 설명으로 가장 옳지 않은 것은?

<보기>



① 모든 부양가족은 자신을 부양하는 직원이 반드시 한 명 있어야 한다.

② 부양가족 개체의 이름 속성은 식별키에 해당한다.

③ 부양가족이 없는 직원도 있다.

④ 직원 개체는 자신과 관련된 부양가족 개체를 소유한다고 말할 수 있다.

정답 2번

출제 포인트

이 문제는 E-R 다이어그램에서 약한 개체, 식별 관계, 부분키 표기와 참여 제약을 읽어 내는지를 묻는 문제이다.

NoteBase

선지별 해설

① O
부양가족은 약한 개체이고 부양 관계에 전체 참여하므로 자신을 부양하는 직원과 반드시 연결되어야 한다.
관계의 1:N 표기상 각 부양가족은 한 직원에 종속된다.

② X
부양가족의 이름은 식별키가 아니라 부분키이다. 약한 개체의 식별은 소유자 개체의 키인 직원의 사번과 부양가족의 부분키인 이름을 함께 써서 이루어진다.
틀린 표현은 "이름 속성은 식별키"이다. 올바르게는 "이름 속성은 부분키이며, 부양가족은 직원의 사번과 이름의 결합으로 식별된다"이다.

③ O
직원 쪽 연결이 단선이므로 직원이 부양 관계에 전체 참여하는 것은 아니다. 따라서 부양가족이 없는 직원도 존재할 수 있다.

④ O
약한 개체는 소유자 개체에 존재 의존하므로 직원 개체가 자신과 관련된 부양가족 개체를 소유한다고 볼 수 있다.
이 도식은 전형적인 owner-weak entity 구조이다.

오답 포인트

점선 밑줄을 일반 키 밑줄과 같은 의미로 오해하면 ②를 맞는 선지로 보기 쉽다. 약한 개체가 이중 사각형이고 관계가 이중 마름모이면 점선 밑줄 속성은 전체 키가 아니라 부분키라는 점을 함께 읽어야 한다.

다시 나오면 이렇게 판단

이중 사각형이 보이면 약한 개체를 먼저 의심하고, 이중 마름모와 점선 밑줄이 함께 나오면 "소유자 키 + 부분키"로 식별된다고 판단한다. 이어서 이중선과 단선으로 전체 참여 여부를 확인하면 참여 제약 선지도 바로 판별된다.

암기 포인트

약한 개체는 부분키만으로 식별되지 않고 소유자 키와 결합되어 식별된다.

NoteBase

문 7. <보기>에서 설명하는 K에 해당하는 키(key)의 명칭으로 가장 옳은 것은?

<보기>
K는 릴레이션 R의 속성들의 부분집합이다. R의 모든 속성들은 K에 대해 함수적 종속성(FD, Functional Dependency)을 가지나, K의 모든 부분집합 K'(단, K' ≠ K)에 대해서는 함수적 종속성을 가지지 않는다.

① 외래키(foreign key)
② 대체키(alternate key)
③ 후보키(candidate key)
④ 슈퍼키(super key)

정답 3번

출제 포인트

이 문제는 모든 속성을 함수적으로 결정하면서도 더 이상 줄일 수 없는 최소 식별자인 후보키를 슈퍼키, 대체키, 외래키와 구별할 수 있는지를 묻는다.

핵심 개념

후보키는 릴레이션의 각 튜플을 유일하게 식별하는 최소 속성 집합이다. 슈퍼키는 튜플을 유일하게 식별하는 속성 집합이지만 불필요한 속성이 포함될 수 있다. 대체키는 후보키 중에서 기본키로 선택되지 않은 후보키이다. 외래키는 다른 릴레이션의 기본키를 참조하여 릴레이션 사이의 연결을 만드는 속성이다.

선지별 해설

1번 X: 외래키는 다른 릴레이션의 기본키를 참조하는 키이다. 틀린 부분은 K가 R의 모든 속성을 함수적으로 결정하고 최소성까지 가진다는 설명을 외래키로 본 점이다. 실제로는 참조 관계를 나타내는 키가 외래키이다.
2번 X: 대체키는 후보키 자체가 아니라 후보키들 가운데 기본키로 채택되지 않은 키이다. 틀린 부분은 최소 식별자 자체를 곧바로 대체키로 본 점이다. 실제로는 후보키가 여러 개일 때 그중 선택되지 않은 후보키가 대체키이다.
3번 O: K는 R의 모든 속성을 함수적으로 결정하므로 유일성 조건을 만족하고, 어떤 진부분집합도 그 성질을 가지지 않으므로 최소성도 만족한다. 유일성과 최소성을 함께 만족하는 키가 후보키이다.
4번 X: 슈퍼키는 모든 튜플을 식별할 수 있으면 되므로 유일성만 있으면 된다. 틀린 부분은 진부분집합이 같은 성질을 가지지 않는다는 최소성까지 슈퍼키의 정의에 넣은 점이다. 실제로 최소성까지 만족하면 후보키이고, 슈퍼키는 불필요한 속성을 포함할 수 있다.

NoteBase

오답 포인트

후보키와 슈퍼키를 같은 것으로 보는 실수가 가장 흔하다. 이 문항의 핵심 함정은 함수적 종속으로 모든 속성을 결정한다는 유일성 조건 뒤에 붙은 진부분집합 불가라는 최소성 조건을 놓치게 만드는 데 있다.

다시 나오면 이렇게 판단

먼저 그 속성 집합이 모든 튜플을 식별하는지 확인하고, 다음으로 속성을 하나라도 빼도 여전히 식별되는지 확인한다. 유일성만 있으면 슈퍼키이고, 유일성과 최소성을 함께 만족하면 후보키이다. 후보키 중 기본키가 아닌 것이 대체키이고, 다른 릴레이션을 참조하면 외래키이다.

암기 포인트

슈퍼키는 식별 가능, 후보키는 식별 가능하면서 최소, 대체키는 선택되지 않은 후보키, 외래키는 다른 릴레이션 참조이다.

NoteBase

문 8. <보기>의 릴레이션 R1, R2, R3에 대해 주어진 관계 대수식(relational algebra) 쌍들 중, (가)와 (나)의 결과가 동일하지 않은 것은?

<보기>

R1=(A, B, C) R2=(D, E) R3=(F, G, H)

① (가) $\Pi_{A,B,H}(R1 \bowtie_{A=F} R3)$, (나) $\sigma_{A=F}(\Pi_{A,B}(R1) \times \Pi_{H}(R3))$

② (가) $\Pi_{B,E}(R1 \bowtie_{A=D} R2)$, (나) $\Pi_{B,E}(\sigma_{A=D}(R1 \times R2))$

③ (가) $(\sigma_{B=5}(R1)) \bowtie_{A=D} R2$, (나) $\sigma_{B=5}(R1 \bowtie_{A=D} R2)$

④ (가) $(R1 \bowtie_{A=D} R2) \bowtie_{D=F} R3$, (나) $(R2 \bowtie_{D=F} R3) \bowtie_{D=A} R1$

정답 1번

출제 포인트

이 문제는 관계대수에서 조인과 카티션 곱+선택의 등가 변환, 선택 연산의 푸시다운, 조인의 결합성을 구별할 수 있는지를 묻는다.

핵심 개념

세타 조인 R ⨝_조건 S 는 σ_조건(R × S)로 바꿀 수 있다.
선택 조건이 한 릴레이션의 속성만 쓰면 조인 전으로 밀어 넣을 수 있다.
투영은 뒤에서 필요한 조인 속성을 남겨 둘 때만 조인 전으로 이동할 수 있다.
조인은 같은 조인 조건을 유지하면 결합 순서를 바꾸어도 같은 결과를 얻는다.

선지별 해설

1번: O. (가)는 R1과 R3를 A=F로 먼저 조인한 뒤 A, B, H만 남긴 식이다. (나)는 Π_H(R3)만 취해 F 속성을 없앤 뒤 σ_A=F를 적용하려는 형태이므로 조인 조건에 필요한 F를 사용할 수 없다. 틀린 부분은 Π_H(R3)로 조인 속성 F를 제거한 점이다. 올바르게는 F를 남긴 뒤 조인하거나, 먼저 조인을 수행한 다음 필요한 속성만 투영해야 같은 결과가 된다.
2번: X. R1 ⨝_A=D R2 는 정의상 σ_A=D(R1 × R2)와 같다. 그 뒤 B, E만 투영하면 두 식의 결과가 동일하다. 틀린 판단은 두 식이 동일하지 않다는 결론이다. 올바르게는 두 식의 결과가 동일하다.
3번: X. 조건 B=5는 R1의 속성 B만 사용하므로 조인 전의 R1에 먼저 적용해도 되고, 조인 뒤 전체 결과에 적용해도 된다. 따라서 두 식의 결과가 동일하다. 틀린 판단은 두 식이 동일하지 않다는 결론이다. 올바르게는 선택 연산을 조인 아래로 밀어 넣을 수 있으므로 동일하다.
4번: X. 두 식 모두 A=D와 D=F 조건을 만족하는 R1, R2, R3의 조인이다. 조인 순서만 바뀌었을 뿐 사용한 조인 조건은 같으므로 결과는 동일하다. 틀린 판단은 두 식이 동일하지 않다는 결론이다. 올바르게는 조인의 결합 순서를 바꾸어도 같은 결과가 된다.

NoteBase

오답 포인트

1번에서 많은 수험생이 조인을 카티션 곱과 선택으로 바꿀 수 있다는 사실만 보고 투영 위치까지 무심코 옮긴다. 그러나 조인 전에 투영할 때는 뒤에서 사용할 조인 속성 F를 반드시 남겨 두어야 한다는 점이 함정이다.

다시 나오면 이렇게 판단

관계대수식의 동치 여부는 먼저 조인 조건에 쓰이는 속성이 중간 투영에서 살아남는지 확인하고, 그다음 조인을 σ(R×S)로 바꿀 수 있는지, 선택 조건이 한 릴레이션에만 걸리는지, 마지막으로 조인 순서만 달라졌는지를 차례대로 보면 된다.

암기 포인트

조인 속성을 잃는 투영은 동치가 아니고, 조인은 σ(R×S)로 바뀌며, 단일 릴레이션 선택은 아래로 내려가고, 조인은 조건만 같으면 순서를 바꿀 수 있다.

NoteBase

문 9. <보기>의 두 릴레이션 '부서'와 '프로젝트'에 대해서 조인(JOIN) 연산을 수행한 결과로 옳지 않은 것은?

<보기>

부서

ID	부서명
1	인사
2	재무
3	총무

프로젝트

ID	프로젝트명
1	채용
2	회계
4	세무
7	총무

① 내부 조인

ID: 1, 부서명: 인사, 프로젝트명: 채용

ID: 2, 부서명: 재무, 프로젝트명: 회계

ID: 3, 부서명: 총무, 프로젝트명: 세무

ID: 7, 부서명: NULL, 프로젝트명: 총무

② 왼쪽 외부 조인

ID: 1, 부서명: 인사, 프로젝트명: 채용

ID: 2, 부서명: 재무, 프로젝트명: 회계

ID: 3, 부서명: 총무, 프로젝트명: NULL

③ 오른쪽 외부 조인

ID: 1, 부서명: 인사, 프로젝트명: 채용

ID: 2, 부서명: 재무, 프로젝트명: 회계

ID: 4, 부서명: NULL, 프로젝트명: 세무

ID: 7, 부서명: NULL, 프로젝트명: 총무

④ 완전 외부 조인

ID: 1, 부서명: 인사, 프로젝트명: 채용

ID: 2, 부서명: 재무, 프로젝트명: 회계

ID: 3, 부서명: 총무, 프로젝트명: NULL

ID: 4, 부서명: NULL, 프로젝트명: 세무

ID: 7, 부서명: NULL, 프로젝트명: 총무

NoteBase

정답 1번

출제 포인트

이 문제는 두 릴레이션을 ID 속성으로 조인할 때 내부 조인, 왼쪽 외부 조인, 오른쪽 외부 조인, 완전 외부 조인의 결과 행이 어떻게 달라지는지를 구분하는 문제이다.

선지별 해설

1번: X. 내부 조인은 양쪽 릴레이션에 모두 존재하는 ID만 남긴다. 따라서 결과는 ID 1의 인사-채용, ID 2의 재무-회계 두 행만 있어야 한다. 제시된 표처럼 ID 3의 총무-세무나 ID 7의 NULL-총무가 포함되면 내부 조인이 아니다. 틀린 표현은 내부 조인 결과에 공통 ID가 아닌 3과 7이 포함된 부분이고, 올바르게는 ID 1과 2만 남아야 한다.
2번: O. 왼쪽 외부 조인은 왼쪽 릴레이션인 부서의 모든 행을 유지한다. 따라서 ID 1, 2는 프로젝트와 매칭되고, ID 3은 대응 프로젝트가 없으므로 프로젝트명이 NULL로 표시된다.
3번: O. 오른쪽 외부 조인은 오른쪽 릴레이션인 프로젝트의 모든 행을 유지한다. 따라서 ID 1, 2는 매칭되고, 부서에 없는 ID 4와 7은 부서명이 NULL로 표시된다.
4번: O. 완전 외부 조인은 양쪽 릴레이션의 모든 행을 유지한다. 따라서 공통 ID 1, 2와 부서에만 있는 ID 3, 프로젝트에만 있는 ID 4, 7이 모두 포함되어 제시된 결과가 맞다.

오답 포인트

프로젝트명에 총무가 있다고 해서 부서의 총무와 연결된다고 보면 틀린다. 조인 기준은 이름이 아니라 ID이다. 내부 조인은 공통 ID만 남기고, 외부 조인은 한쪽 릴레이션의 고아 행까지 남긴다는 점을 섞어 버리면 1번 같은 오답을 고르지 못한다.

다시 나오면 이렇게 판단

먼저 조인 속성이 무엇인지 확인한 뒤 양쪽에 공통으로 존재하는 키를 찾는다. 그다음 내부 조인은 공통 키만, 왼쪽 외부 조인은 왼쪽 전체, 오른쪽 외부 조인은 오른쪽 전체, 완전 외부 조인은 양쪽 전체를 남긴다고 대입하면 된다.

암기 포인트

내부 조인은 교집합, 왼쪽 외부 조인은 왼쪽 기준, 오른쪽 외부 조인은 오른쪽 기준, 완전 외부 조인은 합집합에 가깝다.

NoteBase

문 10. 릴레이션 처리에서 데이터 중복으로 인해 발생하는 이상(anomaly) 현상으로 가장 옳지 않은 것은?

① 삭제 이상(deletion anomaly)

② 검색 이상(search anomaly)

③ 삽입 이상(insertion anomaly)

④ 갱신 이상(update anomaly)

정답 2번

출제 포인트

이 문제는 데이터 중복으로 생기는 대표적 이상 현상의 표준 분류를 정확히 구별하는 문제이다.

핵심 개념

삽입 이상은 어떤 사실을 저장하려면 불필요한 다른 값까지 함께 입력해야 하는 현상이다. 삭제 이상은 한 튜플을 삭제할 때 따로 보관해야 할 정보까지 함께 사라지는 현상이다. 갱신 이상은 같은 사실이 여러 튜플에 중복되어 일부만 수정되면 데이터 불일치가 생기는 현상이다. 정규화 문맥에서 대표적 이상 현상은 삽입 이상, 삭제 이상, 갱신 이상으로 정리한다.

선지별 해설

① O 삭제 이상은 중복 저장된 데이터에서 한 행을 지우는 과정에서 필요한 다른 정보까지 함께 소실되는 현상이므로 대표적 이상 현상에 해당한다.
② X 틀린 부분은 "검색 이상"이다. 실제로는 데이터 중복으로 인한 대표적 이상 현상은 검색 이상이 아니라 삽입 이상, 삭제 이상, 갱신 이상이다.
③ O 삽입 이상은 특정 사실만 저장하고 싶어도 다른 속성값이 없으면 삽입이 불가능하거나 불필요한 값까지 넣어야 하는 현상이므로 대표적 이상 현상에 해당한다.
④ O 갱신 이상은 같은 내용이 여러 튜플에 반복되어 일부만 수정되면 서로 다른 값이 남는 현상이므로 대표적 이상 현상에 해당한다.

오답 포인트

보기들이 모두 anomaly라는 영어 표현을 붙이고 있어 ②도 그럴듯해 보인다는 점이 함정이다. 그러나 정규화에서 데이터 중복으로 인한 기본 이상 현상은 삽입, 삭제, 갱신의 세 가지로 정리한다.

다시 나오면 이렇게 판단

문제에 데이터 중복과 이상 현상이 함께 나오면 먼저 삽입 이상, 삭제 이상, 갱신 이상의 세 가지를 기준으로 본다. 여기에 없는 낯선 용어가 보이면 표준 분류인지부터 의심하는 방식으로 판단한다.

NoteBase

암기 포인트

중복이 낳는 이상은 삽입, 삭제, 갱신의 세 가지이며 검색 이상은 표준 분류가 아니다.

NoteBase

문 11. <보기 1>의 테이블 스키마는 저자, 책, 저자-책 관계 정보를 나타내고 밑줄 친 속성은 기본키이다. <보기 2>는 동일한 책을 공동으로 저술한 모든 저자들의 이름과 그 책의 book_id를 함께 출력하기 위한 질의문이다. <보기 2> 질의문의 빈칸에 들어갈 것을 가장 옳게 짝지은 것은? (단, 동일한 책에 대한 공동저자들의 이름은 순서에 관계없이 한 번씩만 출력되어야 하고, 3인 이상 공동 저술한 책은 없다.)

<보기 1>

테이블 스키마	설명
author(author_id INT, name VARCHAR(10))	저자 정보 테이블
book(book_id INT, title VARCHAR(50))	책 정보 테이블
writes(author_id INT, book_id INT)	저자와 책 간의 관계 정보 테이블

<보기 2>

```
SELECT  a1.name, a2.name, w1.book_id
FROM    writes w1, writes w2, author a1, author a2
WHERE   w1.book_id (가) and w1.author_id (나)
        and w1.author_id = a1.author_id and (다) a2.author_id;
```

① (가): != w2.book_id, (나): > w2.author_id, (다): w2.author_id =

② (가): = w2.book_id, (나): != w2.author_id, (다): w2.author_id =

③ (가): = w2.book_id, (나): != w2.author_id, (다): a1.author_id =

④ (가): = w2.book_id, (나): < w2.author_id, (다): w2.author_id =

정답 4번

출제 포인트

이 문제는 writes 테이블의 자기 조인으로 같은 책의 공동저자를 한 번만 출력하도록 조인 조건을 구성할 수 있는지를 묻는다.

핵심 개념

공동저자 추출은 writes를 자기 조인하여 같은 book_id를 가진 두 행을 짝짓는 방식이다.
중복 제거는 두 저자를 단순히 다르다고만 두는 것이 아니라 author_id에 방향을 주어 한쪽만 남기는 방식이다.
저자명 출력은 w1은 a1과, w2는 a2와 각각 author_id로 연결해야 한다.

NoteBase

선지별 해설

1번: X. w1.book_id != w2.book_id 이면 같은 책의 공동저자가 아니라 서로 다른 책의 저자쌍을 묶게 된다. 틀린 표현은 '!= w2.book_id'이다. 올바르게는 '= w2.book_id'이다.
2번: X. w1.book_id = w2.book_id 와 w2.author_id = a2.author_id 는 맞지만, w1.author_id != w2.author_id 는 같은 저자쌍을 (A,B)와 (B,A)로 모두 남겨 이름이 중복 출력된다. 틀린 표현은 '!= w2.author_id'이다. 올바르게는 '< w2.author_id' 또는 '> w2.author_id'처럼 한 방향 비교를 둔다.
3번: X. w1.book_id = w2.book_id 와 저자 구분 조건 자체는 가능하지만, a1.author_id = a2.author_id 는 두 author 별칭끼리만 비교할 뿐 w2를 a2에 연결하지 못한다. 틀린 표현은 'a1.author_id ='이다. 올바르게는 'w2.author_id = a2.author_id'이다.
4번: O. w1.book_id = w2.book_id 로 같은 책을 쓴 두 행을 묶고, w1.author_id < w2.author_id 로 한 방향만 남겨 공동저자 이름을 한 번씩만 출력한다. 또 w1.author_id = a1.author_id 와 w2.author_id = a2.author_id 로 각 writes 행을 해당 저자 정보와 정확히 연결한다.

오답 포인트

공동저자를 구할 때 '서로 다른 저자' 조건만 넣으면 충분하다고 생각하기 쉽다. 그러나 이 경우 (A,B)와 (B,A)가 모두 조회된다. 이 문항의 함정은 '다르다'와 '한 번만 나온다'를 같은 뜻으로 오해하게 만드는 데 있다.

다시 나오면 이렇게 판단

자기 조인 문제는 먼저 같은 대상을 묶는 동치 조건이 무엇인지 확인하고, 다음으로 중복 제거가 필요하면 < 또는 > 같은 방향 비교를 두며, 마지막으로 각 별칭이 어떤 원본 테이블 별칭과 연결되는지 확인하면 된다.

암기 포인트

공동저자 자기 조인은 '같은 책은 =, 한 번만 출력은 < 또는 >, 저자명 연결은 각 writes 별칭이 자기 author 별칭과 연결'로 정리된다.

NoteBase

문 12. DBMS는 사용자가 제시한 SQL 질의를 동등한 관계대수식으로 번역한 후 다양한 방식으로 최적화를 수행한다. SQL 문장 혹은 이를 번역한 관계대수식에 대한 최적화와 관련된 설명으로 가장 옳지 않은 것은?

① 주기억장치보다 더 큰 데이터에 대해 order by 구문은 외부 정렬 알고리즘에 의해 정렬(sort)-합병(merge) 전략으로 수행시킨다.

② where SSN='1234' 구문에서, SSN이 기본키이고 그 기본키에 기본 인덱스가 생성되어 있다면 이진 탐색(Binary Search) 방식으로 동작한다.

③ where 절에 논리합 조건이 있는 경우는 논리곱 조건보다 실행하거나 최적화하기가 더 어려운 경우가 많다.

④ $R \bowtie_{A=B} S$와 같은 동등 조인 연산은 중첩 루프 조인 (nested loop join) 방식으로 처리 가능하다.

정답 2번

출제 포인트

이 문제는 SQL 질의 최적화에서 정렬, 선택 연산 접근 방식, 논리합 조건 처리, 조인 알고리즘의 기본 성격을 구분하는 문제이다.

핵심 개념

외부 정렬은 주기억장치에 다 올리지 못하는 데이터를 디스크 기반 sort-merge로 정렬하는 방식이다.
이진 탐색은 정렬된 파일 자체에서 범위를 반씩 줄여 원하는 값을 찾는 방식이다.
인덱스 탐색은 인덱스 구조를 따라 원하는 레코드 위치를 찾는 방식으로, 정렬된 파일에 대한 단순 이진 탐색과 구별된다.
논리곱 조건은 선택 조건을 순차 적용하거나 인덱스를 결합하기 쉬운 편이고, 논리합 조건은 결과 합집합과 중복 처리 때문에 더 까다로운 편이다.
중첩 루프 조인은 바깥 릴레이션의 각 튜플마다 안쪽 릴레이션을 검사하는 조인 방식으로, 동등 조인에도 적용할 수 있다.

NoteBase

선지별 해설

① O
주기억장치보다 큰 데이터의 ORDER BY는 대표적으로 외부 정렬을 사용한다. 외부 정렬의 기본 아이디어가 정렬된 런을 만든 뒤 병합하는 sort-merge 전략이므로 설명이 맞다.
② X
틀린 부분은 "기본 인덱스가 생성되어 있다면 이진 탐색 방식으로 동작한다"는 표현이다. 이진 탐색은 정렬된 파일 자체를 반씩 줄여 찾는 접근 방식이고, 기본 인덱스가 있으면 일반적으로 인덱스를 따라 원하는 블록이나 레코드를 찾는다. 올바르게는 "정렬된 파일이면 이진 탐색을 사용할 수 있고, 기본 인덱스가 있으면 인덱스 탐색을 이용한다"이다.
③ O
WHERE 절의 논리합 조건은 각 조건 결과를 합집합으로 결합하고 중복도 처리해야 하므로 논리곱 조건보다 실행 계획과 최적화가 더 어려운 경우가 많다.
④ O
동등 조인은 nested loop join으로 처리할 수 있다. 동등 조인은 hash join이나 sort-merge join에도 잘 맞지만, 가능 여부만 보면 중첩 루프 조인으로도 충분히 수행 가능하다.

오답 포인트

②는 "기본키"와 "기본 인덱스"라는 말 때문에 빠른 탐색을 곧바로 이진 탐색으로 연결하기 쉬운 점이 함정이다. 탐색 대상이 정렬된 데이터 파일인지, 인덱스 구조인지부터 구분해야 한다.

다시 나오면 이렇게 판단

선지가 접근 경로를 말하면 먼저 대상이 파일 자체인지 인덱스인지 구분한다. 정렬된 파일 자체를 반씩 줄여 찾으면 이진 탐색이고, 인덱스 엔트리를 따라 찾으면 인덱스 탐색이다. 조인 알고리즘 문제는 "가능한가"와 "가장 효율적인가"를 구분해서 판단한다.

암기 포인트

정렬된 파일은 이진 탐색, 인덱스가 있으면 인덱스 탐색, 큰 ORDER BY는 외부 정렬, 동등 조인은 중첩 루프 조인도 가능하다.

NoteBase

문 13. <보기>는 부서와 그 부서에서 수행하는 프로젝트에 대한 데이터를 저장하는 테이블 스키마다. 이 스키마에서 (가) 출력하고자 하는 내용과 (나) 이를 출력하기 위한 SQL문을 짝지은 것으로 가장 옳지 않은 것은? (단, 밑줄 친 속성이 기본키며, 목표 외의 다른 내용은 출력되지 않아야 한다.)

<보기>
부서(ID, 부서이름, 부서위치, 설립일자, 부서장)
프로젝트(ID, 프로젝트명, 담당부서ID, 예산, 프로젝트타입, 프로젝트시작일)

① (가) 예산이 10,000 이상인 프로젝트의 이름과 예산 출력
(나) SELECT 프로젝트명, 예산
FROM 프로젝트
WHERE 예산 >= 10000;

② (가) 부서별로 수행 중인 모든 소관 프로젝트의 평균 예산을 부서이름과 함께 출력하되, 프로젝트가 없는 부서는 제외
(나) SELECT 부서.부서이름, AVG(프로젝트.예산) AS avg_budget
FROM 부서 JOIN 프로젝트 ON 부서.ID = 프로젝트.담당부서ID
GROUP BY 부서.부서이름;

③ (가) 모든 부서의 이름과 해당 부서에서 수행 중인 프로젝트의 수를 출력하되, 프로젝트가 없는 부서도 포함
(나) SELECT 부서.부서이름, COUNT(프로젝트.ID) AS project_count
FROM 부서
LEFT JOIN 프로젝트 ON 부서.ID = 프로젝트.담당부서ID
GROUP BY 부서.부서이름;

④ (가) 특정 부서(ID = 2)에서 수행 중인 프로젝트 중 예산이 가장 높은 프로젝트의 이름을 출력
(나) SELECT 프로젝트명
FROM 프로젝트
WHERE 예산 = (SELECT MAX(예산) FROM 프로젝트 WHERE
담당부서ID = 2);

정답 4번

출제 포인트

이 문제는 조건 검색, 조인 방식, 집계 함수, 서브쿼리의 적용 범위가 출력 목표와 정확히 일치하는지를 묻는다.

NoteBase

핵심 개념

INNER JOIN은 매칭되는 행만 남기므로 관련 프로젝트가 없는 부서는 자동으로 제외된다.
LEFT JOIN은 왼쪽 테이블의 모든 행을 남기며, COUNT(프로젝트.ID)처럼 NULL이 아닌 값만 세면 프로젝트가 없는 부서도 0건으로 표현된다.
서브쿼리로 최댓값을 구할 때는 그 값을 비교하는 외부 질의에도 같은 대상 조건이 있어야 원하는 집합 안에서의 최대값 행만 반환된다.

선지별 해설

1번 O: 프로젝트 테이블에서 프로젝트명, 예산만 선택하고 WHERE 예산 >= 10000 조건을 주었으므로 예산이 10,000 이상인 프로젝트의 이름과 예산만 출력한다.
2번 O: 부서와 프로젝트를 JOIN하면 프로젝트가 있는 부서만 남으므로 프로젝트가 없는 부서는 제외된다. AVG(프로젝트.예산)로 부서별 평균 예산을 구하고 부서이름과 함께 출력하므로 요구와 맞는다.
3번 O: 모든 부서를 포함해야 하므로 LEFT JOIN이 맞다. COUNT(프로젝트.ID)는 실제로 매칭된 프로젝트 ID만 세므로 프로젝트가 없는 부서는 0으로 집계된다.
4번 X: 틀린 부분은 외부 질의에 담당부서ID = 2 조건이 없다는 점이다. 현재 SQL은 부서 2의 최대 예산값과 같은 예산을 가진 다른 부서 프로젝트도 함께 반환할 수 있다. 올바르게는 SELECT 프로젝트명 FROM 프로젝트 WHERE 담당부서ID = 2 AND 예산 = (SELECT MAX(예산) FROM 프로젝트 WHERE 담당부서ID = 2); 이다.

오답 포인트

4번은 서브쿼리 안에 담당부서ID = 2가 있으므로 전체 결과도 자동으로 부서 2로 제한된다고 착각하기 쉽다. 하지만 서브쿼리는 최대 예산값만 계산할 뿐이고, 실제로 어떤 행을 출력할지는 외부 WHERE 절이 따로 결정한다.

다시 나오면 이렇게 판단

SQL 문제에서는 먼저 출력 대상 집합이 무엇인지 정한 뒤, 조인이 그 집합을 맞게 만들었는지, 집계 함수가 빈 행을 어떻게 처리하는지, 서브쿼리의 조건이 외부 질의에도 필요한지 순서대로 확인하면 된다. 특히 MAX, MIN 서브쿼리는 외부 질의에도 동일한 대상 조건이 있는지를 본다.

암기 포인트

INNER JOIN은 없는 행을 빼고, LEFT JOIN + COUNT(컬럼)은 없는 행을 0으로 남긴다. 서브쿼리의 조건과 외부 질의의 대상 조건은 서로 별개이다.

NoteBase

문 14. 트랜잭션의 동시성 제어(concurrency control)를 위한 로킹(locking)에 대한 설명으로 가장 옳지 않은 것은? (단, lock은 배타로크이고 shared-lock은 공유로크이다.)

① 트랜잭션 T는 다른 트랜잭션에 의해 이미 lock이 걸려 있는 데이터 아이템 x에 대해 다시 lock(x)를 실행시키지 못한다.

② 트랜잭션 T가 데이터 아이템 x에 대해 실행한 lock(x)에 대해서는 T가 모든 실행을 종료하기 전에 반드시 unlock(x)를 실행해야 한다.

③ 트랜잭션 T가 데이터 아이템 x에 대해 read(x)나 write(x)를 하려면 반드시 먼저 lock(x)를 실행해야 한다.

④ 트랜잭션 T가 데이터 아이템 x에 대해 shared-lock을 걸고 난 후 T는 x에 대해 판독과 기록을 할 수 있다.

정답 4번

출제 포인트

이 문제는 배타로크와 공유로크의 접근 권한 차이와 lock-unlock의 기본 규칙을 구별할 수 있는지를 묻는다.

핵심 개념

배타로크는 한 트랜잭션만 해당 데이터 아이템을 사용하게 하는 잠금으로 읽기와 쓰기를 모두 할 수 있다.
공유로크는 여러 트랜잭션이 함께 가질 수 있는 잠금으로 읽기는 가능하지만 쓰기는 할 수 없다.

선지별 해설

1번 O: 다른 트랜잭션이 이미 데이터 아이템 x에 로크를 걸어 두었다면, 트랜잭션 T는 그 로크가 해제될 때까지 배타로크 lock(x)를 다시 획득할 수 없다. 배타로크는 다른 트랜잭션의 접근과 양립하지 않는다.
2번 O: 트랜잭션 T가 x에 대해 획득한 로크는 작업이 끝난 뒤 반드시 unlock(x)로 해제되어야 한다. 로크를 해제하지 않으면 다른 트랜잭션이 해당 데이터에 접근할 수 없게 된다.
3번 O: 데이터 아이템에 접근하기 전에는 먼저 잠금을 획득해야 한다는 locking의 기본 절차를 설명한 선지이다. write(x)에는 배타로크가 필요하고, read(x)도 잠금 없이 바로 수행하지 않는다.
4번 X: shared-lock은 공유로크이므로 판독만 허용하고 기록은 허용하지 않는다. 틀린 표현은 "판독과 기록을 할 수 있다"이다. 올바르게는 "shared-lock을 걸고 난 후에는 x를 판독할 수만 있고, 기록을 하려면 배타로크 lock(x)를 걸어야 한다"이다.

오답 포인트

공유로크도 잠금이므로 쓰기까지 가능하다고 섞어 외우면 4번에 걸린다. 이 문항의 함정은 공유로크의 허용 범위를 읽기 전용으로 제한한다는 점을 놓치게 만드는 데 있다.

NoteBase

다시 나오면 이렇게 판단

선지에 lock과 shared-lock이 함께 나오면 먼저 배타로크인지 공유로크인지 구분한 뒤, 읽기만 가능한지 읽기와 쓰기가 모두 가능한지를 대입해 판단한다.

암기 포인트

공유로크는 읽기 전용, 배타로크는 읽기와 쓰기 가능이라는 기준으로 정리된다.

NoteBase

문 15. 대한 설명으로 가장 옳지 않은 것은?

① 기록 단계(write phase) - 판독 단계(read phase) - 확인 단계(validation phase)를 거쳐 처리된다.

② 트랜잭션을 실행하는 동안 모든 갱신은 트랜잭션을 위해 유지되는 작업 구역의 지역 사본에만 반영된다.

③ 지역 사본에 반영된 트랜잭션의 실행 결과를 데이터베이스에 그대로 반영할 때 직렬 가능성 위반 여부를 확인한다.

④ 아무런 검사를 시행하지 않고 트랜잭션 실행을 시작하는 것으로 트랜잭션이 수행하는 갱신은 다 종료될 때까지 데이터베이스에 직접 반영되지 않는다.

정답 1번

출제 포인트

이 문제는 검증 기반 병행 제어의 단계 순서와 갱신 반영 시점을 묻는다.

핵심 개념

검증 기반 병행 제어는 충돌을 미리 막지 않고 트랜잭션이 읽기와 계산을 끝낸 뒤 직렬 가능성을 검사한 다음 결과를 데이터베이스에 반영하는 낙관적 병행 제어 기법이다. 단계 순서는 판독 단계(read phase) - 확인 단계(validation phase) - 기록 단계(write phase)이다.

선지별 해설

1번 X: 틀린 부분은 "기록 단계(write phase) - 판독 단계(read phase) - 확인 단계(validation phase)"라는 순서이다. 검증 기반 병행 제어의 실제 순서는 판독 단계에서 지역 사본을 사용해 연산하고, 확인 단계에서 직렬 가능성을 검사한 뒤, 마지막에 기록 단계에서 데이터베이스에 반영하는 것이다.
2번 O: 검증 기반 병행 제어에서는 트랜잭션 수행 중 갱신 내용을 데이터베이스에 바로 쓰지 않고 작업 구역의 지역 사본에만 유지한다. 그래서 충돌 검사는 마지막 검증 시점까지 미뤄진다.
3번 O: 지역 사본의 결과를 실제 데이터베이스에 반영하기 전에 다른 트랜잭션과 함께 보았을 때 직렬 가능성이 깨지는지를 확인한다. 이 검사가 validation phase의 핵심이다.
4번 O: 이 기법은 시작 시 잠금 같은 선행 검사를 하지 않아도, 갱신 결과를 즉시 데이터베이스에 반영하지는 않는다. 트랜잭션의 읽기와 연산이 끝나고 검증까지 통과한 뒤에만 write phase에서 실제 데이터베이스에 기록된다.

오답 포인트

검증 기반 병행 제어는 이름 때문에 write phase가 먼저 나온다고 착각하기 쉽다. 그러나 이 기법의 핵심은 먼저 지역 사본에서 읽기와 갱신을 수행하고, 마지막에만 실제 데이터베이스에 쓰는 점이다.

NoteBase

다시 나오면 이렇게 판단

선지에 낙관적 병행 제어나 validation phase가 보이면 먼저 지역 사본 사용 여부를 확인하고, 다음으로 단계 순서가 read-validation-write인지, 실제 데이터베이스 반영이 마지막인지 차례로 판단한다.

암기 포인트

검증 기반 병행 제어는 먼저 읽고 검증한 뒤 마지막에 쓴다.

NoteBase

문 16. <보기>는 트랜잭션 처리 과정에서 남은 로그 기록이며, 마지막 줄까지 로그를 기록한 후 시스템 장애가 발생하였다. <보기>의 로그 기록을 사용하여 복구(recovery) 과정을 거친 후의 A, B, C, D 값을 옳게 짝지은 것은?

<보기>
로그 기록
<T1, start>
<T2, start>
<T1, B, 6, 12>
<T3, start>
<T1, commit>
<T3, C, 21, 8>
<checkpoint, active (T2, T3)>
<T4, start>
<T3, commit>
<T4, B, 12, 17>
<T5, start>
<T2, A, 45, 16>
<T4, abort>
<T5, D, 16, 21>
<T5, commit>
-- 시스템 장애 발생 --

① A: 16, B: 12, C: 8, D: 21

② A: 16, B: 12, C: 21, D: 21

③ A: 45, B: 12, C: 8, D: 21

④ A: 45, B: 12, C: 21, D: 21

정답 3번

출제 포인트

이 문제는 체크포인트가 있는 즉시 갱신 로그에서 redo 대상과 undo 대상을 구분하여 복구 후 A, B, C, D 값을 판단하는 문제이다.

NoteBase

올바른 순서

로그 기록은 <트랜잭션, 데이터, 이전값, 이후값> 형식이며, 즉시 갱신에서 undo는 이전값(before-image, 세 값 중 첫 번째)으로 되돌리고 redo는 이후값(after-image, 두 번째)을 다시 반영한다.
1. 체크포인트의 active 목록인 T2, T3를 보고 crash 시점까지의 미완료 후보를 잡는다.
2. 체크포인트 이후 commit된 T3와 T5는 redo 대상으로 본다. T3의 <T3, C, 21, 8>은 이후값 8을, T5의 <T5, D, 16, 21>은 이후값 21을 재반영하여 C=8, D=21이 된다.
3. crash 시 commit되지 않은 T2는 undo 대상이다. <T2, A, 45, 16>을 이전값으로 되돌려 A=16이 아니라 A=45가 된다.
4. T4는 abort 기록이 있으므로 <T4, B, 12, 17>이 undo되어 이전값 12로 되돌아간다. T4가 갱신한 대상은 A가 아니라 B이므로 A에는 영향을 주지 않는다.
T1은 체크포인트 이전에 <T1, B, 6, 12>로 B=12를 쓰고 commit하였으므로 이미 반영된 상태이며, 이후 T4의 B 갱신이 abort undo로 취소되어 최종 B=12가 된다.

선지별 해설

① X: A가 틀렸다. T2의 <T2, A, 45, 16>은 commit되지 않았으므로 recovery에서 undo되어, 이후값 16이 아니라 이전값(before-image)인 45로 되돌아간다. 따라서 A는 16이 아니라 45이다. 올바르게는 A=45, B=12, C=8, D=21이다.
② X: A와 C가 틀렸다. A는 T2 undo로 45가 되어야 하고, C는 T3의 <T3, C, 21, 8>이 commit된 redo 대상이므로 21이 아니라 8이 된다. 올바르게는 A=45, B=12, C=8, D=21이다.
③ O: T2는 undo되어 A=45가 되고, T1의 반영값과 T4 abort 결과로 B=12가 유지되며, T3 redo로 C=8, T5 redo로 D=21이 된다.
④ X: C가 틀렸다. C는 T3 commit 이후 redo되어 21이 아니라 8이 된다. 올바르게는 A=45, B=12, C=8, D=21이다.

오답 포인트

체크포인트가 있다고 해서 active 목록의 T2를 그대로 남겨 둔다고 오해하거나, abort된 T4의 B=17을 최종값으로 착각하기 쉽다. 체크포인트 이후 commit된 트랜잭션은 redo하고, crash 시 commit되지 않은 트랜잭션은 undo한다는 구분이 핵심이다.

다시 나오면 이렇게 판단

로그 복구 문제는 마지막 체크포인트의 active 트랜잭션을 기준으로 시작하여 commit 완료 트랜잭션은 redo, crash 시 미완료 트랜잭션은 undo, abort 완료 트랜잭션은 이미 원상복구된 것으로 판단하면 된다.

암기 포인트

복구는 완료 redo, 미완료 undo, abort는 원상복구이다.

NoteBase

문 17. <보기>는 SQL에 정의된 고립도(Isolation Level)에 따라 발생 가능한 위반 유형을 나타내고 있다. (가)~(라)에 들어갈 것을 옳게 짝지은 것은? (단, Yes는 위반이 가능함을 나타내고, No는 위반이 가능하지 않음을 나타낸다.)

<보기>

고립도 (Isolation Level)	위반 유형		
	오손 읽기 (Dirty Read)	반복할 수 없는 읽기 (Nonrepeatable Read)	팬텀 (Phantom)
(가)	Yes	Yes	Yes
(나)	No	Yes	Yes
(다)	No	No	Yes
(라)	No	No	No

① (가) READ UNCOMMITTED (나) READ COMMITTED
(다) REPEATABLE READ (라) SERIALIZABLE

② (가) READ UNCOMMITTED (나) NO DIRTY READ
(다) REPEATABLE READ (라) SERIALIZABLE

③ (가) READ UNCOMMITTED (나) NO DIRTY READ
(다) REPEATABLE READ (라) NO PHANTOM READ

④ (가) DIRTY READ (나) NO DIRTY READ
(다) REPEATABLE READ (라) NO PHANTOM READ

정답 1번

출제 포인트

이 문제는 SQL 표준의 고립도별로 허용되는 이상 현상인 오손 읽기, 반복할 수 없는 읽기, 팬텀을 정확히 대응시키는지를 묻는다.

NoteBase

핵심 개념

READ UNCOMMITTED는 커밋 전 데이터까지 읽을 수 있는 최저 고립도이므로 오손 읽기, 반복할 수 없는 읽기, 팬텀이 모두 가능하다.
READ COMMITTED는 커밋된 데이터만 읽으므로 오손 읽기는 막지만 반복할 수 없는 읽기와 팬텀은 가능하다.
REPEATABLE READ는 같은 행을 다시 읽을 때 값이 달라지는 반복할 수 없는 읽기는 막지만, 조건에 맞는 새 행이 나타나는 팬텀은 가능하다.
SERIALIZABLE은 직렬 실행과 같은 수준이므로 세 가지 이상 현상을 모두 막는다.

선지별 해설

1번: O. (가) READ UNCOMMITTED, (나) READ COMMITTED, (다) REPEATABLE READ, (라) SERIALIZABLE의 대응이 표의 Yes/No 패턴과 정확히 일치한다.
2번: X. 틀린 표현은 '(나) NO DIRTY READ'이다. 표의 (나)는 오손 읽기가 불가능하고 나머지 두 현상은 가능한 고립도이므로 올바르게는 READ COMMITTED이다.
3번: X. 틀린 표현은 '(나) NO DIRTY READ'와 '(라) NO PHANTOM READ'이다. (나)는 현상 설명이 아니라 고립도 이름인 READ COMMITTED가 들어가야 하고, (라)는 세 현상이 모두 불가능한 SERIALIZABLE이 들어가야 한다.
4번: X. 틀린 표현은 '(가) DIRTY READ', '(나) NO DIRTY READ', '(라) NO PHANTOM READ'이다. 고립도 칸에는 위반 현상명이 아니라 고립도 이름이 와야 하므로 각각 READ UNCOMMITTED, READ COMMITTED, SERIALIZABLE로 고쳐야 한다. (다) REPEATABLE READ만 맞다.

오답 포인트

오손 읽기 가능 여부와 고립도 이름을 섞어 놓은 선지에 흔들리기 쉽다. 특히 REPEATABLE READ가 팬텀까지 막는다고 착각하거나, READ COMMITTED를 'NO DIRTY READ' 같은 현상 부정 표현으로 바꾸면 틀린다.

다시 나오면 이렇게 판단

먼저 각 행이 고립도 이름인지 현상 이름인지부터 구분한 뒤, READ UNCOMMITTED는 세 가지 모두 허용, READ COMMITTED는 오손 읽기만 차단, REPEATABLE READ는 팬텀만 남음, SERIALIZABLE은 모두 차단의 순서로 대입하면 된다.

암기 포인트

RU는 다 허용, RC는 Dirty만 차단, RR은 Phantom만 남고, Serializable은 전부 차단이다.

NoteBase

문 18. 분산 데이터베이스의 특징에 대한 설명으로 가장 옳지 않은 것은?

① 분산 데이터베이스는 컴퓨터 네트워크를 통해 분산된, 논리적으로 상호연관된 다수의 데이터베이스의 모임이다.

② 수평 단편화란 한 릴레이션의 투플들의 부분집합을 여러 사이트에 할당하는 방식이다.

③ 여러 사이트에 데이터가 완전히 중복된 분산 데이터베이스를 구축할 경우, 사용자 검색 질의나 갱신 요구를 효율적으로 처리할 수 있다.

④ 2단계 완료 프로토콜(2PC)은 봉쇄형 프로토콜이라는 단점이 있어서, 3단계 완료 프로토콜(3PC)을 사용하기도 한다.

정답 3번

출제 포인트

이 문제는 분산 데이터베이스의 기본 개념과 단편화, 완전 중복, 2PC·3PC의 성질을 구별할 수 있는지를 묻는다.

핵심 개념

분산 데이터베이스는 네트워크로 연결된 여러 사이트에 데이터를 분산 저장하되 논리적으로는 하나의 데이터베이스처럼 관리하는 구조이다. 수평 단편화는 한 릴레이션의 투플을 행 단위로 나누는 방식이고, 완전 중복은 같은 데이터를 여러 사이트에 복제해 두는 방식이다. 2PC는 모든 참여자가 응답할 때까지 대기해야 하므로 봉쇄 가능성이 있고, 3PC는 이를 완화하려는 프로토콜이다.

선지별 해설

1번 O: 분산 데이터베이스의 표준적 정의에 해당한다. 여러 사이트에 물리적으로 분산되어 있어도 논리적으로는 상호 연관된 하나의 시스템처럼 관리된다는 점이 핵심이다.
2번 O: 수평 단편화는 릴레이션의 행, 즉 투플들의 부분집합을 나누어 여러 사이트에 배치하는 방식이다. 열 단위 분할은 수직 단편화이므로 구별해야 한다.
3번 X: 틀린 부분은 "검색 질의나 갱신 요구를 효율적으로 처리할 수 있다"는 묶음 표현이다. 완전 중복은 읽기 질의는 유리할 수 있지만, 갱신은 모든 복제본에 반영해야 하므로 오히려 비용과 일관성 유지 부담이 커진다. 올바르게는 "완전 중복은 검색 질의 처리에는 유리할 수 있으나 갱신 요구 처리에는 비효율이 커질 수 있다"이다.
4번 O: 2PC는 코디네이터나 참여자 장애 시 대기 상태가 길어질 수 있는 봉쇄형 프로토콜이다. 이 단점을 줄이기 위해 3PC를 사용하는 경우가 있다는 설명은 타당하다.

오답 포인트

3번은 "완전 중복 = 무조건 효율적"이라는 인상을 이용한 함정이다. 복제는 조회 성능과 가용성을 높일 수 있지만, 갱신 비용과 동기화 부담은 커진다는 점을 같이 보아야 한다.

NoteBase

다시 나오면 이렇게 판단

분산 데이터베이스 문제에서는 먼저 개념을 정의로 구분하고, 그다음 읽기 성능과 갱신 비용이 서로 어떻게 달라지는지 확인하면 정답을 가를 수 있다. 복제 방식이 나오면 조회 이점과 갱신 부담을 분리해서 판단한다.

암기 포인트

수평 단편화는 행 분할, 완전 중복은 조회 유리·갱신 불리, 2PC는 봉쇄 가능성이 있다.

NoteBase

문 19. 분산 데이터베이스에서 <보기 1>의 테이블 스키마에 대해 <보기 2>의 질의를 처리한 결과(RESULT)를 사이트(site) 3으로 전송하고자 한다. 데이터 전송량을 줄이기 위해 세미조인을 이용하여 분산 질의 처리를 수행한다고 할 때, 이에 대한 설명으로 가장 옳지 않은 것은?

<보기 1>

테이블 스키마	설명
EMP (ID, Ename, Dno)	• 직원 정보 테이블로 사이트 1에 존재함 • 직원 1,000명, ID와 Dno 속성의 크기는 각 4바이트, Ename 속성의 크기는 10바이트임
DEPT (Dno, Dname, MgrID)	• 부서 정보 테이블로 사이트 2에 존재함 • MgrID는 EMP 테이블의 직원번호 속성인 ID를 참조하는 외래키로 부서 관리자의 ID를 나타내며, 중복된 값이나 NULL 값을 가질 수 없음 • 부서 10개, Dno와 MgrID 속성의 크기는 각 4바이트, Dname 속성의 크기는 10바이트임

<보기 2>

$RESULT \leftarrow \pi_{MgrID,\ Ename,\ Dname}(EMP \bowtie_{ID\ =\ MgrID} DEPT)$

① 사이트 1에서 사이트 2로 $\pi_{Dno}(EMP)$을 전송해야 한다.

② 사이트 2에서 사이트 1로 $\pi_{MgrID}(DEPT)$을 전송해야 한다.

③ 사이트 2에서 사이트 3으로 240바이트를 전송해야 한다.

④ 사이트 1에서 사이트 2로 140바이트를 전송해야 한다.

정답 1번

출제 포인트

이 문제는 세미조인으로 전송 튜플 수를 줄일 때 어느 속성을 어느 방향으로 보내야 하는지와 각 단계의 전송량을 계산하는 문제이다.

조건 정리

EMP는 사이트 1에 있고 1000튜플이다. EMP에서 ID는 4바이트, Ename은 10바이트이다. DEPT는 사이트 2에 있고 10튜플이다. DEPT에서 MgrID는 EMP.ID를 참조하며 중복과 NULL이 없으므로 MgrID 값은 10개이다. 최종 결과는 부서마다 관리자 1명씩 매칭되어 10튜플이다.

NoteBase

사용할 공식

세미조인은 먼저 작은 쪽의 조인 속성만 보내 상대 사이트에서 필요 튜플만 걸러낸 뒤, 걸러진 튜플의 필요한 속성만 다시 보내는 방식이다. 전송량은 튜플 수 × 전송 속성 크기로 계산한다. 이 문제에서는 DEPT가 10튜플뿐이므로 사이트 2에서 사이트 1로 $\pi_{MgrID}(DEPT)$를 보내 EMP를 10명의 관리자 튜플로 줄이는 것이 가장 효율적이다.

계산 과정

1. 사이트 2에서 사이트 1로 $\pi_{MgrID}(DEPT)$를 보낸다. 크기는 $10 \times 4 = 40$바이트이다.
2. 사이트 1에서 EMP를 세미조인하면 관리자 10명의 튜플만 남는다.
3. 사이트 1에서 사이트 2로 세미조인 결과의 $\pi_{ID,\,Ename}$를 보낸다. 크기는 $10 \times (4 + 10) = 140$바이트이다.
4. 사이트 2에서 DEPT와 조인한 뒤 RESULT의 속성 $MgrID$, $Ename$, $Dname$만 남기면 한 튜플 크기는 $4 + 10 + 10 = 24$바이트이다.
5. 최종 RESULT는 10튜플이므로 사이트 2에서 사이트 3으로 보내는 크기는 $10 \times 24 = 240$바이트이다.

선지 판단

1번: X. 잘못된 표현은 "사이트 1에서 사이트 2로 $\pi_{ID}(EMP)$를 전송"이라는 부분이다. $\pi_{ID}(EMP)$는 1000개 ID이므로 $1000 \times 4 = 4000$바이트가 되어 전송량 절감 효과가 작다. 올바르게는 사이트 2에서 사이트 1로 $\pi_{MgrID}(DEPT)$ 40바이트를 먼저 보내고, 그 후 걸러진 관리자 10명에 대한 $\pi_{ID,\,Ename}$ 140바이트를 사이트 2로 보낸다.
2번: O. DEPT의 MgrID는 10개이며 중복이 없으므로 사이트 2에서 사이트 1로 $\pi_{MgrID}(DEPT)$를 보내는 것이 세미조인의 첫 단계가 된다. 전송량은 40바이트이다.
3번: O. 최종 RESULT는 부서 10개에 대해 $MgrID$, $Ename$, $Dname$만 가지므로 한 튜플은 24바이트이고 전체는 240바이트이다. 따라서 사이트 2에서 사이트 3으로 240바이트를 전송한다.
4번: O. 세미조인으로 EMP에서 관리자 10명만 남긴 뒤 필요한 속성은 ID와 Ename뿐이다. 따라서 사이트 1에서 사이트 2로 보내는 데이터는 $10 \times (4 + 10) = 140$바이트이다.

오답 포인트

EMP가 1000튜플이라는 사실만 보고 $\pi_{ID}(EMP)$를 먼저 보내는 것으로 착각하기 쉽다. 세미조인은 전체 조인 속성을 보내는 것이 아니라 더 작은 관계의 조인 속성으로 큰 관계를 먼저 줄이는 방식이라는 점을 놓치면 1번을 옳다고 보기 쉽다.

다시 나오면 이렇게 판단

분산 질의 전송량 문제는 먼저 어느 관계가 더 작은지, 조인 속성 투영을 어느 방향으로 보내면 상대 관계를 크게 줄일 수 있는지부터 본다. 그 다음 걸러진 튜플 수와 필요한 속성 크기만 다시 계산하면 된다.

NoteBase

암기 포인트

세미조인은 작은 쪽 조인 속성을 먼저 보내 큰 쪽을 줄인 뒤, 걸러진 결과의 필요한 속성만 전송하는 방식이다.

NoteBase

문 20. <보기>에서 설명하는 데이터 마이닝(data mining) 분석 기법으로 가장 옳은 것은?

<보기>
• 비지도학습(unsupervised learning)의 일종이다.
• 유사한 구매 패턴(pattern)을 가진 고객들의 집단을 알아내는 데 사용할 수 있다.

① 연관 분석
② 군집 분석
③ 분류 분석
④ 회귀 분석

정답 2번

출제 포인트

이 문제는 비지도학습에 해당하는 데이터 마이닝 기법 중에서 유사한 특성을 가진 대상을 묶는 군집 분석의 개념을 묻는 문제이다.

핵심 개념

군집 분석은 정답 레이블 없이 비슷한 속성을 가진 데이터를 여러 집단으로 나누는 기법이다. 연관 분석은 함께 나타나는 항목 사이의 동시발생 규칙을 찾는 기법이다. 분류 분석은 미리 주어진 범주를 기준으로 새 데이터를 어느 범주에 넣을지 판단하는 지도학습 기법이다. 회귀 분석은 수치형 값을 예측하는 지도학습 기법이다.

선지별 해설

1번 X: 연관 분석은 상품 A를 사면 상품 B도 함께 사는 경향처럼 항목 간 관계를 찾는 기법이다. 틀린 부분은 고객들의 집단을 알아낸다는 취지이다. 올바르게는 함께 구매되는 품목의 연관 규칙을 찾는 데 사용한다.
2번 O: 군집 분석은 비지도학습의 대표 기법이며, 구매 패턴이 비슷한 고객들을 같은 군집으로 묶어 세분화할 수 있다. 보기의 두 문장을 모두 만족한다.
3번 X: 분류 분석은 정답이 있는 학습 데이터로 범주를 예측하는 지도학습이다. 틀린 부분은 비지도학습이라는 전제와 고객 집단을 스스로 찾아낸다는 취지이다. 올바르게는 기존 범주를 기준으로 고객이 어느 집단에 속하는지 분류한다.
4번 X: 회귀 분석은 매출액, 수요량처럼 연속형 수치를 예측하는 기법이다. 틀린 부분은 유사한 구매 패턴의 고객 집단을 찾는다는 점이다. 올바르게는 고객 집단화가 아니라 수치 예측에 사용한다.

NoteBase

오답 포인트

비지도학습이라는 문장을 보고 연관 분석과 군집 분석을 함께 떠올리면 헷갈리기 쉽다. 이 문항은 둘 다 비지도 성격을 가질 수 있다는 막연한 인상보다, 대상들을 비슷한 특성으로 묶는가 아니면 항목 사이의 동시구매 관계를 찾는가를 구분하는지가 핵심 함정이다.

다시 나오면 이렇게 판단

문장에서 고객들을 몇 개의 집단으로 나눈다, 비슷한 패턴끼리 묶는다고 하면 군집 분석으로 판단한다. 품목 간 함께 발생하는 규칙이면 연관 분석, 미리 정해진 범주 예측이면 분류, 연속형 값 예측이면 회귀로 구분한다.

암기 포인트

군집은 비슷한 대상을 묶고, 연관은 함께 나타나는 항목을 찾고, 분류는 범주를 맞히고, 회귀는 수치를 예측한다.

NoteBase