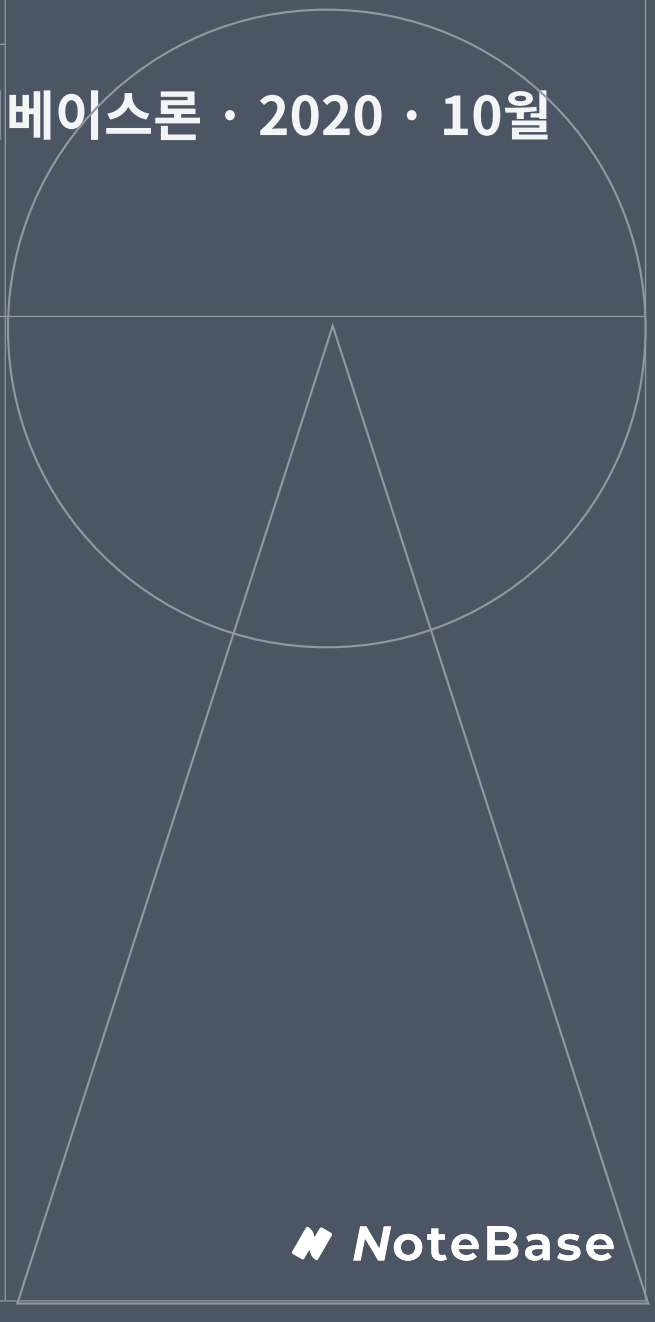
기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2020 · 10월



NoteBase

기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2020 · 10월

공무원/서울시/7급 · 데이터베이스론 · 2020 · 10월

문 1. 데이터베이스 설계에서 물리적 설계에 해당하지 않는 것은?

① 접근 경로(access path) 설계

② 저장 레코드(stored record)의 양식(format) 설계

③ 스키마의 평가와 정제(evaluation & refinement)

④ 레코드 집중(clustering)의 분석 및 설계

정답 3번

출제 포인트

데이터베이스 설계 단계 중 물리적 설계의 작업과 논리적 설계의 작업을 구분하는 문제이다.

핵심 개념

물리적 설계는 논리 스키마를 실제 저장 장치에 구현하는 단계로, 레코드 형식, 접근 경로, 파일 조직, 클러스터링처럼 저장과 접근 방식을 정하는 작업이다.
논리적 설계는 스키마를 관계 구조로 정리하고 평가·정제하는 단계로, 정규화나 스키마 개선이 여기에 해당한다.
구별 기준은 저장 방식과 접근 효율을 정하면 물리적 설계이고, 스키마 구조 자체를 다듬으면 논리적 설계라는 점이다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2020 · 10월

선지별 해설

① X
접근 경로 설계는 인덱스, 해시, 탐색 경로처럼 데이터를 어떻게 빨리 찾을지를 정하는 작업이므로 물리적 설계에 해당한다.
틀린 부분은 이 선지를 물리적 설계에 해당하지 않는 것처럼 고른 점이다. 올바르게는 '접근 경로 설계는 물리적 설계에 해당한다'이다.

② X
저장 레코드의 양식 설계는 레코드를 디스크에 어떤 형식으로 저장할지 정하는 작업이므로 물리적 설계에 해당한다.
틀린 부분은 이 선지를 비해당 항목으로 보는 점이다. 올바르게는 '저장 레코드의 양식 설계는 물리적 설계에 해당한다'이다.

③ O
스키마의 평가와 정제는 중복과 이상을 줄이도록 스키마 구조를 점검하고 개선하는 작업이므로 논리적 설계 단계의 내용이다.
따라서 물리적 설계에 해당하지 않는 것을 묻는 이 문제의 정답이 된다.

④ X
레코드 집중은 관련 레코드를 가까이 저장하여 접근 효율을 높이는 저장 구조 설계이므로 물리적 설계에 해당한다.
틀린 부분은 이 선지를 물리적 설계에 해당하지 않는 것으로 본 점이다. 올바르게는 '레코드 집중의 분석 및 설계는 물리적 설계에 해당한다'이다.

오답 포인트

③의 '스키마의 평가와 정제'를 저장 구조 최적화와 비슷하게 보고 물리적 설계로 착각하기 쉽다. 그러나 '스키마'를 다듬는 작업은 구조 개선이고, 접근 경로·레코드 형식·클러스터링처럼 저장과 접근을 정하는 작업만 물리적 설계이다. 또한 발문이 '해당하지 않는 것'이라는 부정형이라는 점도 함께 걸림돌이 된다.

다시 나오면 이렇게 판단

선지의 핵심어가 저장, 접근 경로, 레코드 배치, 파일 조직이면 물리적 설계로 판단할 수 있다. 반대로 스키마 평가, 정제, 정규화처럼 구조 자체를 다듬는 말이면 논리적 설계로 판단할 수 있다.

암기 포인트

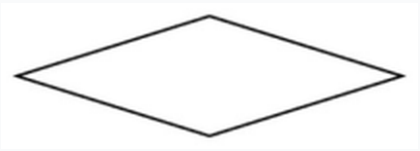
저장·접근·집중은 물리, 평가·정제는 논리이다.

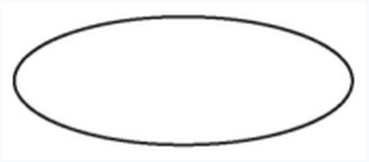
NoteBase

문 2. ER 다이어그램에서 주민등록번호와 같은 기본키가 존재하지 않는 피부양자와 부양자의 관계를 표현하는 관계 기호에 해당하는 것은?

① 

② 

③ 

④ 

정답 1번

출제 포인트

이 문제는 ER 다이어그램에서 기본키가 없는 약한 개체와 소유자 개체를 연결하는 식별 관계를 이중 마름모로 구분하는지를 묻는 문제이다.

NoteBase

선지별 해설

① O
이중 마름모는 약한 개체와 강한 개체 사이의 식별 관계를 나타낸다. 피부양자는 주민등록번호와 같은 독자적 기본키가 없는 약한 개체로 볼 수 있으므로 부양자와의 관계는 이 기호로 표현된다.

② X
사각형은 개체 타입을 나타내는 기호이다. 관계 기호를 고르는 발문과 맞지 않는다. 올바르게는 피부양자와 부양자의 관계는 이중 마름모로 표시된다.

③ X
단일 마름모는 일반 관계를 나타낸다. 약한 개체를 식별해 주는 관계는 일반 관계가 아니라 식별 관계이므로 이 기호는 맞지 않는다. 올바르게는 이 경우 이중 마름모를 사용한다.

④ X
타원은 속성을 나타내는 기호이다. 주민등록번호와 같은 기본키 유무는 속성 설명이지만, 발문은 관계 기호를 묻고 있으므로 타원은 맞지 않는다. 올바르게는 관계를 이중 마름모로 표현한다.

오답 포인트

약한 개체 자체의 모양과 약한 개체를 식별하는 관계의 모양을 섞어 보면 틀리기 쉽다. 이 문항은 피부양자라는 개체가 아니라 피부양자와 부양자의 관계 기호를 묻기 때문에 사각형이나 타원이 아니라 마름모 계열에서 판단해야 한다.

다시 나오면 이렇게 판단

ER 문제에서 독자적 기본키가 없는 개체가 나오면 먼저 약한 개체 여부를 판단하고, 그 개체를 소유자 개체와 연결하는 관계가 식별 관계인지 확인한다. 식별 관계이면 이중 마름모이고 일반 관계이면 단일 마름모이다.

암기 포인트

약한 개체의 식별 관계는 이중 마름모이고 일반 관계는 단일 마름모이다.

NoteBase

문 3. <보기>의 관계 데이터베이스 설계의 함수적 종속성과 정규형에 대한 설명 중 괄호에 들어갈 용어는?

<보기>
릴레이션 R이 BCNF에 속하고 모든 ( )이 함수 종속(functionally dependent)이면 릴레이션 R은 4NF에 속한다.

① 다치 종속성(multivalued dependency)
② 이행 종속성(transitive dependency)
③ 부분 종속성(partial dependency)
④ 조인 종속성(join dependency)

정답 1번

출제 포인트

이 문제는 BCNF, 4NF, 5NF가 각각 어떤 종속성을 기준으로 정의되는지를 구분할 수 있는지를 묻는 문제이다.

핵심 개념

함수 종속성은 어떤 속성값이 다른 속성값을 하나로 결정하는 관계이다.
다치 종속성은 하나의 결정자에 대해 서로 독립적인 여러 값의 집합이 대응되는 관계이다.
조인 종속성은 릴레이션이 여러 투영의 자연조인으로 손실 없이 복원되는 성질이다.
정규형 대응은 2NF-부분 종속성 제거, 3NF-이행 종속성 제거, BCNF-모든 함수 종속성의 결정자가 슈퍼키, 4NF-모든 다치 종속성의 결정자가 슈퍼키, 5NF-조인 종속성 처리이다.

선지별 해설

① O: 4NF는 다치 종속성을 기준으로 정의된다. BCNF에 속하면서 모든 다치 종속성이 사실상 함수 종속성으로 환원되면, 비자명한 다치 종속성의 결정자가 모두 슈퍼키가 되어 4NF를 만족한다.
② X: 틀린 부분은 "이행 종속성"이다. 이행 종속성은 주로 3NF에서 제거 대상이 되는 개념이며, 4NF의 기준이 아니다. 올바르게는 3NF는 이행 종속성을 제거한 정규형이다.
③ X: 틀린 부분은 "부분 종속성"이다. 부분 종속성은 복합키의 일부분에만 종속되는 경우로 2NF의 판단 기준이다. 올바르게는 2NF는 부분 종속성을 제거한 정규형이다.
④ X: 틀린 부분은 "조인 종속성"이다. 조인 종속성은 5NF와 연결되는 개념이며, 4NF를 설명하는 용어가 아니다. 올바르게는 5NF는 조인 종속성과 관련된 정규형이다.

NoteBase

오답 포인트

정규형과 종속성의 대응 관계를 한 묶음으로 기억하지 않으면 4NF와 5NF, 2NF와 3NF를 서로 바꾸기 쉽다. 이 문항은 괄호 뒤의 "함수 종속이면 릴레이션 R은 4NF에 속한다"라는 문장을 보고 4NF의 기준이 다치 종속성임을 떠올렸는지를 확인한다.

다시 나오면 이렇게 판단

문장에서 목표 정규형이 4NF이면 먼저 다치 종속성을 찾고, 5NF이면 조인 종속성, 3NF이면 이행 종속성, 2NF이면 부분 종속성을 대응시켜 판단하면 된다.

암기 포인트

2NF-부분, 3NF-이행, 4NF-다치, 5NF-조인이다.

NoteBase

문 4. 2단계 로킹(two-phase locking)에 대한 설명으로 가장 옳지 않은 것은?

① 모든 트랜잭션이 이를 준수하면 직렬 가능(conflict serializable)하다.

② 모든 트랜잭션이 이를 준수하면 교착 상태(deadlock)에 빠지지 않는다.

③ 확장(growing) 단계에서는 lock만 수행할 수 있고 unlock은 수행할 수 없다.

④ 축소(shrinking) 단계에서는 unlock만 수행할 수 있고 lock은 수행할 수 없다.

정답 2번

출제 포인트

이 문제는 2단계 로킹의 보장 범위와 확장 단계·축소 단계의 동작을 구분하는 문제이다.

핵심 개념

2단계 로킹은 트랜잭션의 락 연산을 확장 단계와 축소 단계로 나누는 동시성 제어 기법이다.
확장 단계는 lock만 획득하고 unlock은 하지 않는 단계이다.
축소 단계는 unlock만 수행하고 새로운 lock은 획득하지 않는 단계이다.
2단계 로킹은 충돌 직렬가능성은 보장하지만, 교착 상태까지 자동으로 방지하지는 않는다.

선지별 해설

① O
2단계 로킹을 모든 트랜잭션이 지키면 스케줄은 충돌 직렬가능하다. 락의 획득과 해제가 두 단계로 분리되어 직렬화 가능한 순서를 만들기 때문이다.
② X
틀린 부분은 "교착 상태(deadlock)에 빠지지 않는다"이다.
실제로는 2단계 로킹만으로도 교착 상태가 발생할 수 있다. 서로 상대방이 가진 lock의 해제를 기다리면 교착 상태가 생기므로, deadlock 방지나 탐지는 별도로 다루어야 한다.
③ O
확장 단계에서는 필요한 lock을 획득만 할 수 있고 unlock을 시작할 수 없다. 따라서 growing 단계에서 unlock은 수행할 수 없다는 설명은 맞다.
④ O
축소 단계에서는 이미 획득한 lock을 해제만 할 수 있고 새로운 lock은 더 이상 얻지 못한다. 2단계 로킹의 shrinking 단계 설명으로 옳다.

오답 포인트

2단계 로킹이 직렬가능성을 보장한다는 사실을 deadlock 방지까지 보장하는 것으로 과대해석하기 쉽다. 이 문항은 "직렬가능성 보장"과 "교착 상태 방지"를 일부러 섞어 놓은 점이 함정이다.

NoteBase

다시 나오면 이렇게 판단

동시성 제어 기법 문제에서는 그 기법이 무엇을 보장하는지와 무엇을 보장하지 않는지를 먼저 분리한다. 2단계 로킹은 충돌 직렬가능성은 O, deadlock 방지는 X, growing은 lock만 O, shrinking은 unlock만 O로 판단하면 된다.

암기 포인트

2PL은 직렬가능성은 보장하고 deadlock은 막지 못한다. growing은 lock만, shrinking은 unlock만이다.

NoteBase

문 5. 릴레이션 R(A, B, C, D)에 대한 함수적 종속성이 <보기>와 같이 주어졌을 때 슈퍼키(super key)가 아닌 것은?

<보기>
ABC → D D → A

① ABC
② BCD
③ D
④ ABCD

정답 3번

출제 포인트

이 문제는 함수적 종속성으로 각 속성 집합의 폐포를 구해 슈퍼키인지 아닌지를 판별하는 문제이다.

핵심 개념

슈퍼키는 어떤 속성 집합의 폐포가 릴레이션의 모든 속성을 포함하는 키이다.
폐포는 주어진 함수적 종속성을 반복 적용하여 그 속성 집합이 결정할 수 있는 모든 속성의 집합이다.
이 문제의 판단 기준은 각 선택지의 폐포가 $ABCD$ 전체가 되는지 여부이다.

NoteBase

선지별 해설

① X

$ABC^+ = ABCD$이다. 주어진 $ABC \rightarrow D$에 의해 D를 얻고, 이미 A, B, C를 가지고 있으므로 전체 속성을 모두 결정한다.

따라서 ①은 '슈퍼키가 아닌 것'이 아니라 슈퍼키이다.

② X

BCD^+에서 먼저 D가 있으므로 $D \rightarrow A$를 적용해 A를 얻는다. 그러면 B, C, D, A를 모두 가지게 되어 $BCD^+ = ABCD$이다.

따라서 ②도 '슈퍼키가 아닌 것'이 아니라 슈퍼키이다.

③ O

$D^+ = DA$이다. $D \rightarrow A$만 적용할 수 있고 B, C는 얻을 수 없다.

전체 속성 $ABCD$를 모두 결정하지 못하므로 D는 슈퍼키가 아니다.

④ X

ABCD는 릴레이션의 모든 속성을 이미 다 포함한 집합이다. 따라서 자기 자신인 $ABCD$를 결정하므로 슈퍼키이다.

따라서 ④는 '슈퍼키가 아닌 것'이 아니라 슈퍼키이다.

오답 포인트

③에서 D가 A를 결정하므로 키처럼 보인다는 점이 함정이다. 그러나 슈퍼키 판단은 한두 속성을 더 얻는지가 아니라 최종적으로 모든 속성 $ABCD$를 얻는지로 본다. ②도 처음에는 A가 없어서 아니라고 보기 쉽지만 D가 있으므로 $D \rightarrow A$를 반드시 한 번 더 적용해야 한다.

다시 나오면 이렇게 판단

선택지를 보면 각 속성 집합의 폐포를 구한다. 함수적 종속성을 더 이상 적용할 수 없을 때까지 반복한 뒤, 폐포에 릴레이션의 모든 속성이 들어 있으면 슈퍼키이고 하나라도 빠지면 슈퍼키가 아니다.

암기 포인트

슈퍼키 판단은 폐포가 전체 속성을 덮는지로 끝낸다. $X^+ = R$이면 슈퍼키이고 $X^+ \neq R$이면 슈퍼키가 아니다.

NoteBase

문 6. 일반적인 DBMS에서 문자열을 저장하기에 가장 적합한 속성의 데이터 타입은?

① SMALLINT

② VARCHAR(n)

③ REAL

④ TIME

정답 2번

출제 포인트

이 문제는 DBMS의 기본 데이터 타입 중 문자열 저장에 쓰는 문자형 타입을 구별하는 문제이다.

핵심 개념

SMALLINT는 작은 범위의 정수를 저장하는 수치형 타입이다.
VARCHAR(n)은 길이가 가변적인 문자열을 최대 n자까지 저장하는 문자형 타입이다.
REAL은 실수를 저장하는 수치형 타입이다.
TIME은 시각 값을 저장하는 날짜·시간형 타입이다.

선지별 해설

① X: SMALLINT는 정수 저장용 수치형 타입이므로 문자열 저장에 적합하지 않다. 틀린 표현은 문자열 저장이라는 판단이고, 올바르게는 작은 정수값 저장에 적합하다.
② O: VARCHAR(n)은 문자 수가 일정하지 않은 문자열을 저장하는 대표적인 가변 길이 문자형 타입이므로 가장 적합하다.
③ X: REAL은 소수점을 포함한 실수 저장용 타입이므로 문자열을 저장하는 타입이 아니다. 틀린 표현은 문자열용 타입으로 보는 점이고, 올바르게는 실수 데이터 저장에 사용된다.
④ X: TIME은 시:분:초와 같은 시간 정보를 저장하는 타입이므로 문자열 전용 타입이 아니다. 틀린 표현은 일반 문자열 저장에 적합하다는 점이고, 올바르게는 시간값 저장에 사용된다.

오답 포인트

DBMS 데이터 타입 문제는 타입 이름만 보고 막연히 고르면 혼동하기 쉽다. SMALLINT와 REAL은 숫자형이고 TIME은 시간형인데, 문자열을 저장한다는 발문을 끝까지 확인하지 않으면 ①이나 ④처럼 의미상 익숙한 타입을 잘못 고를 수 있다.

NoteBase

다시 나오면 이렇게 판단

먼저 저장하려는 값이 문자열, 정수, 실수, 시간 중 무엇인지 데이터의 성격을 분류한 뒤 타입군을 대응시키면 된다. 문자형이면 CHAR 또는 VARCHAR, 숫자형이면 정수·실수 타입, 시간 정보면 DATE/TIME 계열로 판단한다.

암기 포인트

문자열은 VARCHAR, 정수는 SMALLINT, 실수는 REAL, 시간은 TIME으로 구분한다.

NoteBase

문 7. <보기 1>의 내용에 따라 <보기 2>의 로크 호환성 테이블을 옳게 채운 것은?

<보기 1>
공유 로크(shared lock)와 배타 로크(exclusive lock)만 가지는 표준 로킹 기법에서, 공유 로크와 배타 로크의 관계는 로크 호환성 테이블로 나타낼 수 있다. <보기 2>는 어떤 트랜잭션 T가 항목 X에 대해 열 제목에 명시된 로크를 보유하고 있고, 다른 트랜잭션 T'이 같은 항목 X에 대해 행 제목에 명시된 로크를 요청할 때, T'이 로크를 획득할 수 있는지 여부를 나타낸다.

<보기 2>

		T 보유	
		공유 로크	배타 로크
T′ 획득 요청	공유 로크	㉠	㉡
	배타 로크	㉢	㉣

① ㉠: Yes, ㉡: Yes, ㉢: Yes, ㉣: Yes

② ㉠: Yes, ㉡: Yes, ㉢: Yes, ㉣: No

③ ㉠: Yes, ㉡: Yes, ㉢: No, ㉣: No

④ ㉠: Yes, ㉡: No, ㉢: No, ㉣: No

정답 4번

출제 포인트

이 문제는 공유 로크와 배타 로크의 호환 여부를 표에 대응시키는 문제이다.

핵심 개념

공유 로크는 여러 트랜잭션이 같은 항목을 함께 읽을 수 있게 하는 읽기 잠금이다 / 결정 차이축: 같은 항목에 다른 로크를 함께 허용하는가 / 키워드: 읽기, S-S 가능
배타 로크는 한 트랜잭션이 항목을 단독으로 사용하게 하는 쓰기 잠금이다 / 결정 차이축: 다른 공유 로크와 배타 로크를 모두 막는가 / 키워드: 쓰기, 단독, X 포함 조합 불가

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2020 · 10월

선지별 해설

① X
- 틀린 부분은 'ㄴ Yes'이다.
- T가 배타 로크를 보유 중이면 T'의 공유 로크 요청은 허용되지 않으므로 올바르게는 'ㄴ No'이다.

② X
- 틀린 부분은 'ㄴ Yes, ㄷ Yes'이다.
- T가 배타 로크를 보유 중인 상태에서는 공유 로크 요청도 불가하고, T가 공유 로크를 보유 중인 상태에서는 다른 트랜잭션의 배타 로크 요청도 불가하므로 올바르게는 'ㄴ No, ㄷ No'이다.

③ X
- 틀린 부분은 'ㄴ Yes'이다.
- 공유 로크끼리는 호환되지만 배타 로크가 이미 보유된 상태에서 추가 공유 로크를 허용할 수 없으므로 올바르게는 'ㄴ No'이다.

④ O
- 공유 로크와 공유 로크만 함께 허용되므로 ㄱ은 Yes이다.
- 배타 로크가 포함되는 ㄴ, ㄷ, ㄹ 조합은 모두 충돌하므로 No가 맞다.

오답 포인트

공유 로크는 읽기 잠금이므로 항상 허용된다고 오해하면 ㄴ을 Yes로 채우기 쉽다. 이 표는 요청 로크만 보는 것이 아니라 이미 보유 중인 로크와의 충돌 여부를 함께 보는 표이다.

다시 나오면 이렇게 판단

표를 외우기보다 'S-S만 가능하고 X가 한쪽이라도 끼면 모두 불가'라는 규칙으로 판단하면 된다. 그다음 행은 요청 로크, 열은 기존 보유 로크로 읽어 각 칸에 대응시키면 된다.

암기 포인트

공유끼리만 Yes, 배타가 끼면 전부 No이다.

NoteBase

문 8. <보기>의 분산 데이터베이스의 단편화 투명성(fragmentation transparency)에 대한 설명에서, (가)와 (나) 각각에 들어갈 용어로 가장 옳은 것은?

<보기>
• (가)는 한 릴레이션을 서브 릴레이션으로 분산하며, 서브 릴레이션은 원래 릴레이션의 튜플(행)들의 부분집합이다.
• (나)는 한 릴레이션을 애트리뷰트들의 부분집합으로 이루어진 두 개 이상의 릴레이션들로 나눈다.

① (가): 수평 단편화, (나): 수직 단편화
② (가): 수직 단편화, (나): 수평 단편화
③ (가): 튜플 단편화, (나): 애트리뷰트 단편화
④ (가): 애트리뷰트 단편화, (나): 튜플 단편화

정답 1번

출제 포인트

이 문제는 분산 데이터베이스의 단편화에서 튜플 기준 분할은 수평 단편화, 애트리뷰트 기준 분할은 수직 단편화라는 대응을 묻는 문제이다.

핵심 개념

수평 단편화는 한 릴레이션을 튜플(행)의 부분집합들로 나누는 방식이다. 결정 차이축은 행 기준 분할 여부이다. 키워드는 튜플, 행, 부분집합이다.
수직 단편화는 한 릴레이션을 애트리뷰트(열)의 부분집합들로 나누는 방식이다. 결정 차이축은 열 기준 분할 여부이다. 키워드는 애트리뷰트, 열, 속성 분할이다.

선지별 해설

① O: (가)는 원래 릴레이션의 튜플(행)들의 부분집합으로 서브 릴레이션을 만든다고 했으므로 수평 단편화이다. (나)는 애트리뷰트들의 부분집합으로 두 개 이상의 릴레이션으로 나눈다고 했으므로 수직 단편화이다.
② X: 틀린 부분은 (가)와 (나)를 서로 바꾼 점이다. 튜플 기준 분할은 수평 단편화이고, 애트리뷰트 기준 분할은 수직 단편화이다.
③ X: 틀린 부분은 정식 용어를 튜플 단편화, 애트리뷰트 단편화로 적은 점이다. 이 설명에 대응하는 표준 분류명은 각각 수평 단편화와 수직 단편화이다.
④ X: 틀린 부분은 (가)를 애트리뷰트 단편화, (나)를 튜플 단편화로 둔 점이다. 실제로는 (가)가 수평 단편화, (나)가 수직 단편화이다.

NoteBase

오답 포인트

행과 열의 기준을 뒤바꾸면 ②, ④에 걸리기 쉽다. 또 튜플과 애트리뷰트라는 설명어를 그대로 단편화의 정식 명칭으로 착각하면 ③으로 흔들릴 수 있다.

다시 나오면 이렇게 판단

단편화 문제는 무엇을 기준으로 나누는지부터 본다. 튜플(행)의 부분집합이면 수평 단편화이고,
애트리뷰트(열)의 부분집합이면 수직 단편화이다.

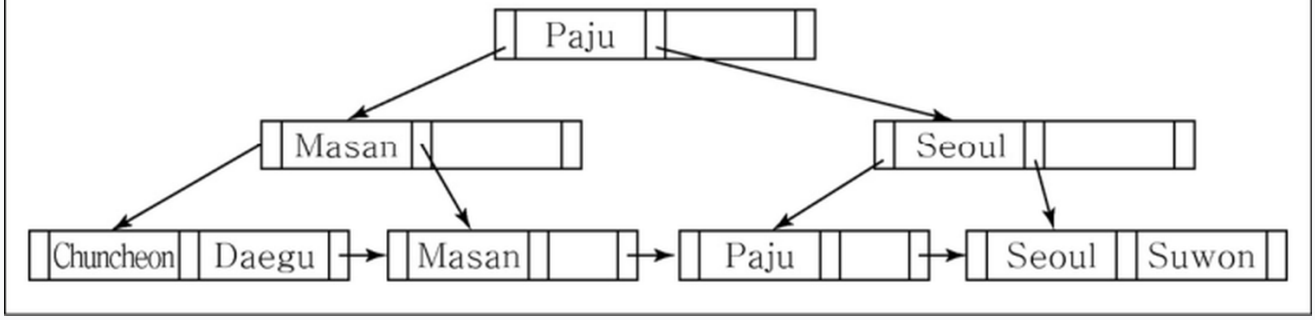
암기 포인트

행을 나누면 수평, 열을 나누면 수직이다.

NoteBase

문 9. <보기>의 트리 구조에 대한 설명으로 가장 옳은 것은?

<보기>



① B^+-트리로서, 삭제 과정에서 재분배가 필요 없는 장점이 있다.

② B-트리로서, 삭제 과정에서 재분배가 필요 없는 장점이 있다.

③ B^+-트리로서, 루트 노드를 제외한 모든 노드는 절반 이상의 사용률(utilization)을 보장한다.

④ B-트리로서, 루트 노드를 제외한 모든 노드는 절반 이상의 사용률(utilization)을 보장한다.

정답 3번

출제 포인트

이 문제는 리프 노드가 순차 포인터로 연결되고 내부 노드에는 탐색용 키만 있는 도식을 보고 B+-트리의 구조와 최소 사용률 조건을 구별하는 문제이다.

NoteBase

선지별 해설

① X
- 그림은 리프 노드들이 왼쪽에서 오른쪽으로 연결되고 실제 데이터 엔트리가 리프에 모여 있으므로 B+-트리이다. 그러나 B+-트리도 삭제 과정에서 언더플로가 발생하면 형제 노드와의 재분배나 병합이 필요하다. 틀린 표현은 "재분배가 필요 없는"이고, 올바르게는 "삭제 과정에서 재분배가 필요할 수 있다"이다.

② X
- B-트리는 내부 노드와 리프 노드 모두가 데이터 엔트리를 가질 수 있지만, 그림은 내부 노드가 분기 키만 가지고 리프 노드가 서로 연결되어 있으므로 B-트리가 아니다. 삭제 시 재분배가 필요 없다는 설명도 틀리다. 틀린 표현은 "B-트리"와 "재분배가 필요 없는"이고, 올바르게는 "B+-트리이며 삭제 과정에서 재분배가 필요할 수 있다"이다.

③ O
- 그림의 내부 노드는 탐색용 분기 키만 가지고 리프 노드가 순차 연결되어 있으므로 B+-트리이다. B+-트리는 루트 노드를 제외한 모든 내부 노드와 리프 노드가 최소 반 이상 차도록 유지되므로 절반 이상의 사용률을 보장한다.

④ X
- "루트 노드를 제외한 모든 노드는 절반 이상의 사용률을 보장한다"는 내용 자체는 균형 다진 탐색트리의 성질과 맞지만, 이 그림의 구조는 B-트리가 아니라 B+-트리이다. 틀린 표현은 "B-트리로서"이고, 올바르게는 "B+-트리로서"이다.

오답 포인트

출제자는 절반 이상의 사용률이라는 일반 성질만 기억한 수험생이 그림의 구조를 먼저 확인하지 않고 ④를 고르도록 유도한다. 이 문제의 핵심 함정은 사용률 조건이 아니라 리프 노드 연결과 내부 노드의 역할을 보고 B-트리와 B+-트리를 먼저 구별해야 한다는 점이다.

다시 나오면 이렇게 판단

도식 문제가 나오면 먼저 내부 노드에 실제 데이터가 들어 있는지, 리프 노드들이 순차 포인터로 연결되어 있는지를 본다. 내부는 분기 키만 두고 데이터는 리프에 모으며 리프가 연결되어 있으면 B+-트리로 판단한 뒤, 삭제 시 재분배 가능성과 최소 반 채움 조건을 적용하면 된다.

암기 포인트

내부는 길잡이 키, 데이터는 리프, 리프 순차 연결이면 B+-트리이며 루트 외 노드는 반 이상 찬다.

NoteBase

문 10. <보기>의 개체-관계(Entity-Relationship) 모델링에서 EMPLOYEE와 SKILL 간의 관계는? (단, EMPLOYEE는 종업원에 대한 정보가, SKILL은 기술에 대한 정보가 저장되어 있다.)

<보기>
한 개의 EMPLOYEE 튜플은 많은 SKILL 튜플들과 연관될 수 있으며, 한 개의 SKILL 튜플은 많은 EMPLOYEE 튜플들과 연관될 수 있다.

① 1 대 1 관계
② 1 대 다 관계
③ 다 대 1 관계
④ 다 대 다 관계

정답 4번

출제 포인트

이 문제는 ER 모델에서 문장으로 주어진 양방향 연결 가능성을 읽고 EMPLOYEE와 SKILL의 카디널리티를 판별하는지를 묻는다.

핵심 개념

1 대 1 관계는 양쪽 엔터티가 서로 하나의 튜플에만 연결되는 관계이다.
1 대 다 관계는 한쪽 엔터티 하나가 다른 쪽 여러 튜플과 연결되지만 반대쪽 각 튜플은 한 튜플에만 연결되는 관계이다.
다 대 1 관계는 1 대 다 관계를 반대 방향에서 본 관계이다.
다 대 다 관계는 양쪽 엔터티가 모두 여러 튜플과 연결될 수 있는 관계이다.

선지별 해설

① X: 보기에서 한 EMPLOYEE 튜플은 많은 SKILL 튜플들과 연결되고, 한 SKILL 튜플도 많은 EMPLOYEE 튜플들과 연결된다고 했다. 틀린 표현은 "1 대 1"이다. 올바르게는 양쪽 모두 복수 연결이 가능한 다 대 다 관계이다.
② X: 1 대 다 관계는 한쪽만 여러 개와 연결될 때 성립한다. 이 보기에서는 EMPLOYEE도 여러 SKILL과 연결되고 SKILL도 여러 EMPLOYEE와 연결된다. 틀린 표현은 "1 대 다"이다. 올바르게는 다 대 다 관계이다.
③ X: 다 대 1 관계는 여러 EMPLOYEE가 하나의 SKILL에 연결되는 구조처럼 한쪽만 단일 연결일 때 성립한다. 이 보기에서는 한 EMPLOYEE도 많은 SKILL과 연결될 수 있다. 틀린 표현은 "다 대 1"이다. 올바르게는 다 대 다 관계이다.
④ O: 한 EMPLOYEE 튜플이 많은 SKILL 튜플들과 연관되고, 한 SKILL 튜플도 많은 EMPLOYEE 튜플들과 연관된다고 명시되어 있으므로 양방향 모두 복수 참여가 가능한 다 대 다 관계가 맞다.

NoteBase

오답 포인트

보기의 첫 문장만 읽고 "한 개의 EMPLOYEE 튜플은 많은 SKILL 튜플들과 연관"이라는 부분 때문에 1 대 다로 고르기 쉽다. 그러나 바로 뒤의 "한 개의 SKILL 튜플은 많은 EMPLOYEE 튜플들과 연관"까지 읽으면 반대 방향도 다수 연결이므로 다 대 다가 된다.

다시 나오면 이렇게 판단

ER 관계 문제는 한 방향만 보지 않고 양쪽 엔터티 각각이 몇 개와 연결되는지를 따로 확인하면 된다. 양쪽 모두 여러 개면 다 대 다, 한쪽만 여러 개면 1 대 다 또는 다 대 1, 양쪽 모두 하나면 1 대 1이다.

암기 포인트

카디널리티는 양쪽을 모두 본다. 양쪽 모두 많으면 다 대 다 관계이다.

NoteBase

문 11. 무결성 규정(integrity rule)에 대한 설명으로 가장 옳지 않은 것은?

① 널(NULL) 무결성 제약조건이란 릴레이션의 특정 속성값이 널(NULL)이 될 수 없다는 규정이다.

② 엔티티(entity) 무결성 제약조건이란 어떠한 기본키(primary key) 값도 널(NULL)이 될 수 없다는 규정이다.

③ 참조 무결성 제약조건이란 외래키 값은 널(NULL)이거나 참조 릴레이션의 기본키(primary key) 값과 달라야 한다는 규정이다.

④ 키 무결성 제약조건이란 하나의 테이블에는 적어도 하나의 키가 존재해야 한다는 규정이다.

정답 3번

출제 포인트

이 문제는 널 무결성, 엔티티 무결성, 참조 무결성, 키 무결성의 정의를 정확히 구분하는지를 묻는 문제이다.

핵심 개념

널 무결성은 특정 속성에 NULL을 허용하지 않는 제약이다.
엔티티 무결성은 기본키가 NULL이 될 수 없다는 제약이다.
참조 무결성은 외래키가 NULL이거나 참조되는 릴레이션의 기본키 값과 일치해야 한다는 제약이다.
키 무결성은 각 릴레이션이 튜플을 식별할 수 있는 키를 가져야 한다는 취지의 제약이다.

선지별 해설

① O: 널 무결성 제약조건은 특정 속성값에 NULL을 허용하지 않는다는 뜻이다. NOT NULL 제약의 의미를 설명한 것으로 볼 수 있으므로 옳다.
② O: 엔티티 무결성은 기본키가 개체를 식별해야 하므로 기본키 값에 NULL을 허용하지 않는 규정이다.
③ X: 틀린 표현은 "참조 릴레이션의 기본키 값과 달라야 한다"이다. 참조 무결성은 외래키 값이 NULL이거나 참조 릴레이션의 기본키 값과 같아야 성립한다.
④ O: 키 무결성은 릴레이션에서 각 튜플을 식별할 수 있는 키가 존재해야 한다는 취지의 설명이므로 옳다.

오답 포인트

③은 참조 무결성의 핵심 단어를 반대로 바꾼 선지이다. 외래키는 참조 대상 기본키와 "달라야" 하는 것이 아니라 "같아야" 한다는 점을 혼동하면 오답이 된다.

다시 나오면 이렇게 판단

무결성 규정 문제는 먼저 기본키인지, 외래키인지, NULL 허용 여부인지로 나눈다. 기본키의 NULL 금지는 엔티티 무결성, 외래키의 일치 여부는 참조 무결성으로 판단하면 된다.

NoteBase

암기 포인트

엔티티 무결성은 기본키 NULL 금지이고, 참조 무결성은 외래키가 NULL 또는 참조 기본키와 일치이다.

NoteBase

문 12. 질의 최적화 방법으로 가장 옳지 않은 것은?

① σ(select)는 최대한 빨리 수행한다.

② Π(project)는 최대한 빨리 수행한다.

③ 수행 비용이 같다면 결과 테이블의 크기가 작은 연산을 먼저 수행한다.

④ ×(cartesian product)는 최대한 빨리 수행한다.

정답 4번

출제 포인트

이 문제는 관계대수 연산의 일반적인 질의 최적화 원칙, 특히 선택·사영의 조기 수행과 카티션 프로덕트의 지연 수행을 구분하는 문제이다.

핵심 개념

선택 σ는 조건에 맞는 튜플만 남겨 중간 결과의 행 수를 줄이는 연산이다.
사영 Π는 필요한 속성만 남겨 중간 결과의 열 수를 줄이는 연산이며, 조인에 필요한 속성은 함께 유지해야 한다.
카티션 프로덕트 ×는 두 릴레이션의 모든 조합을 만드는 연산이어서 중간 결과를 급격히 키우므로 가능한 한 늦게 수행하거나 조인 조건과 결합해 대체하는 것이 원칙이다.

선지별 해설

① O
선택 연산은 조건에 맞지 않는 튜플을 초기에 제거하므로 이후 연산이 처리할 행 수를 줄인다. 질의 최적화의 대표 원칙이 선택의 조기 수행이므로 옳다.

② O
사영 연산은 필요한 속성만 남겨 중간 결과의 열 폭을 줄인다. 실제 최적화에서는 이후 조인이나 조건 평가에 필요한 속성은 남겨 둔 채 가능한 한 빨리 수행하므로 옳다.

③ O
수행 비용이 같다면 결과 릴레이션의 크기가 작은 연산을 먼저 수행하는 편이 이후 연산의 부담을 줄인다. 중간 결과를 작게 유지하는 것이 최적화의 기본 방향이므로 옳다.

④ X
틀린 부분은 "카티션 프로덕트는 최대한 빨리 수행한다"는 표현이다.
카티션 프로덕트는 불필요하게 큰 중간 결과를 만들기 쉬우므로 가능한 한 늦게 수행하는 것이 원칙이다.
실제로는 선택 조건을 먼저 적용하고, 가능하면 카티션 프로덕트 대신 조인 형태로 변환해 처리한다.

NoteBase

오답 포인트

②와 ④가 모두 "최대한 빨리 수행한다"는 형태라서 같은 최적화 원칙처럼 보이지만, 선택과 사영은 중간 결과를 줄이는 연산이고 카티션 프로덕트는 중간 결과를 폭증시키는 연산이라는 점이 핵심 차이이다.

다시 나오면 이렇게 판단

질의 최적화 선지를 보면 먼저 중간 결과를 줄이는 연산인지, 키우는 연산인지를 구분하면 된다. 선택과 사영은 앞당기고, 카티션 프로덕트는 미루며, 같은 비용이면 더 작은 결과를 만드는 쪽을 먼저 택하는 방식으로 판단한다.

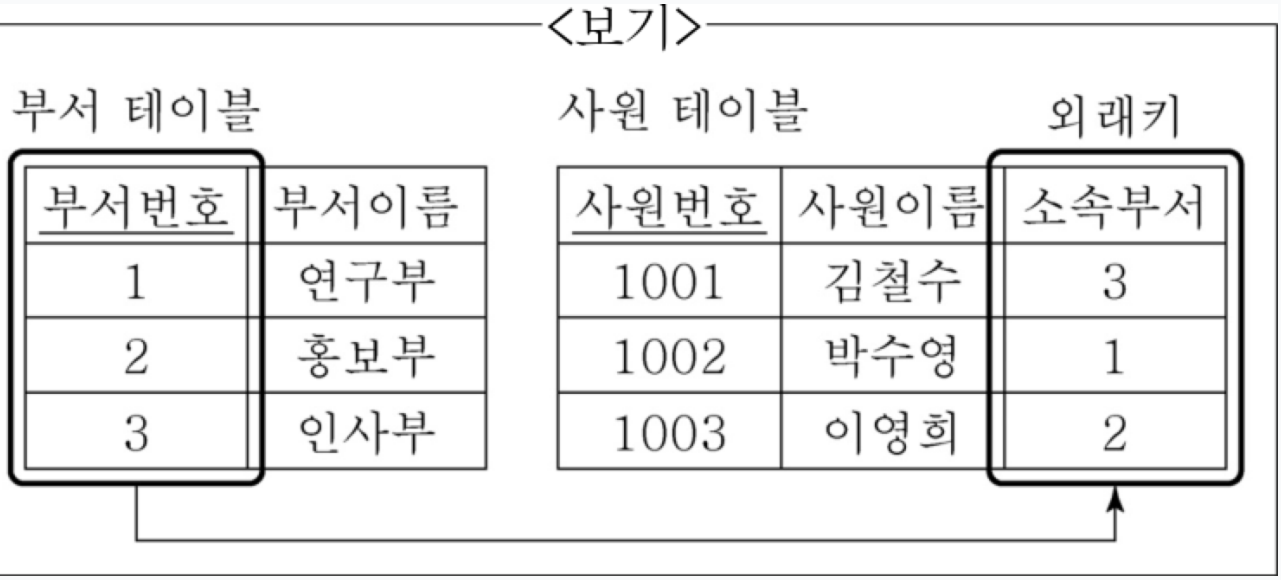
암기 포인트

선택·사영은 앞, 카티션 프로덕트는 뒤가 기본 원칙이다.

NoteBase

문 13. <보기>에서 사원 테이블의 소속부서 속성은 부서 테이블의 부서번호 속성을 참조하는 외래키이다. 사원 테이블을 정의하는 CREATE TABLE을 작성할 때 외래키에 ON DELETE NO ACTION을 지정하였으면, 부서 테이블에서 홍보부 튜플을 삭제하려고 할 때 발생하는 일로 가장 옳은 것은?

<보기>



부서 테이블

부서번호	부서이름
1	연구부
2	홍보부
3	인사부

사원 테이블

사원번호	사원이름	소속부서
1001	김철수	3
1002	박수영	1
1003	이영희	2

① 부서 테이블의 홍보부 튜플이 삭제되지 않는다.

② 사원 테이블에서 이영희 사원에 대한 튜플도 함께 삭제된다.

③ 사원 테이블에서 이영희 사원 튜플의 소속부서 속성값이 널(NULL)로 변경된다.

④ 사원 테이블에서 이영희 사원 튜플의 소속부서 속성값이 미리 지정한 기본값으로 변경된다.

정답 1번

출제 포인트

이 문제는 외래키의 참조 무결성 옵션 중 ON DELETE NO ACTION이 부모 튜플 삭제 시 어떻게 동작하는지를 사례에 적용할 수 있는지를 묻는다.

사례 정리

부서 테이블의 부서번호는 부모 키이고, 사원 테이블의 소속부서는 이를 참조하는 외래키이다.
홍보부의 부서번호는 2이고, 사원 테이블에서 이영희의 소속부서 값도 2이다.
즉 홍보부 튜플은 현재 사원 테이블의 이영희 튜플에 의해 참조되고 있다.

핵심 개념

ON DELETE NO ACTION은 부모 키를 참조하는 자식 튜플이 남아 있으면 부모 튜플 삭제를 허용하지 않는 옵션이다.
참조 중인 부모 행을 지우려면 먼저 자식 행의 외래키 값을 변경하거나 자식 행을 삭제해야 한다.

NoteBase

적용 기준

삭제 대상 부모 행이 다른 테이블의 외래키로 참조되는지부터 확인한다.
참조되고 있는데 옵션이 NO ACTION이면 부모 행 삭제가 거부되고 기존 데이터는 그대로 유지된다.

선지별 해설

① O: 홍보부 부서번호 2를 이영희 튜플의 소속부서 값 2가 참조하고 있으므로, ON DELETE NO ACTION에서는 부서 테이블의 홍보부 튜플을 삭제할 수 없다.
② X: 이 설명은 부모 행 삭제 시 자식 행도 함께 삭제되는 ON DELETE CASCADE의 동작이다. 틀린 표현은 "이영희 사원에 대한 튜플도 함께 삭제된다"이고, 올바르게는 "NO ACTION에서는 이영희 튜플은 삭제되지 않고 부모 행 삭제가 거부된다"이다.
③ X: 이 설명은 부모 행 삭제 시 외래키 값을 NULL로 바꾸는 ON DELETE SET NULL의 동작이다. 틀린 표현은 "소속부서 속성값이 널(NULL)로 변경된다"이고, 올바르게는 "NO ACTION에서는 NULL로 바뀌지 않고 삭제 자체가 허용되지 않는다"이다.
④ X: 이 설명은 부모 행 삭제 시 외래키 값을 기본값으로 바꾸는 ON DELETE SET DEFAULT의 동작이다. 틀린 표현은 "미리 지정한 기본값으로 변경된다"이고, 올바르게는 "NO ACTION에서는 기본값으로 바뀌지 않고 부모 행 삭제가 거부된다"이다.

오답 포인트

삭제 옵션 이름을 보지 않고 결과만 감으로 연결하면 CASCADE, SET NULL, SET DEFAULT를 혼동하기 쉽다. 이 문제의 함정은 "삭제를 시도할 때 자식 행에 어떤 변화가 생기는가"를 묻는 것처럼 보이지만, 실제 핵심은 NO ACTION이면 부모 행 삭제가 성립하지 않는다는 점이다.

다시 나오면 이렇게 판단

외래키 삭제 옵션 문제는 먼저 부모 행이 자식 행에 의해 참조되는지 확인하고, 그다음 옵션별 결과를 대응시키면 된다. NO ACTION은 삭제 거부, CASCADE는 자식도 삭제, SET NULL은 NULL 대입, SET DEFAULT는 기본값 대입으로 판단한다.

암기 포인트

참조 중인 부모 행에서 NO ACTION은 아무 조치를 하지 않는 것이 아니라 삭제를 막는 옵션이다.

NoteBase

문 14. 시스템이 교착 상태(deadlock)에 빠졌으면 교착 상태를 야기한 트랜잭션들 중의 일부를 철회해야 한다. 철회시킬 트랜잭션을 선택하는 것을 희생자 선택(victim selection)이라 하며, 희생자 선택을 위한 올바른 방안을 <보기>에서 모두 고른 것은?

<보기>
ㄱ. 먼저 시작한 트랜잭션을 선택
ㄴ. 나중에 시작한 트랜잭션을 선택
ㄷ. 많은 갱신을 수행한 트랜잭션을 선택
ㄹ. 적은 갱신을 수행한 트랜잭션을 선택

① ㄱ, ㄷ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄴ, ㄹ

정답 4번

출제 포인트

이 문제는 교착상태 해소를 위한 희생자 선택에서 롤백 비용을 최소화하는 기준을 묻는 문제이다.

핵심 개념

희생자 선택은 교착상태를 풀기 위해 철회할 트랜잭션을 고르는 절차이다.
기준은 시스템 전체 손실을 최소화하는 데 있으므로, 일반적으로 더 최근에 시작한 트랜잭션과 갱신량이 적은 트랜잭션이 우선 희생자가 된다.
먼저 시작했거나 이미 많은 갱신을 수행한 트랜잭션은 지금까지 투입한 작업량이 커서 철회 비용이 더 크다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2020 · 10월

선지별 해설

① X
ㄱ의 "먼저 시작한 트랜잭션"이 틀렸다. 오래 실행된 트랜잭션은 이미 수행한 작업이 많을 가능성이 커서 철회 비용이 크다.
ㄷ의 "많은 갱신을 수행한 트랜잭션"도 틀렸다. 많은 갱신은 되돌릴 작업이 많다는 뜻이므로 희생자 선택 기준으로 불리하다.
틀린 표현은 "먼저 시작한", "많은 갱신을 수행한"이고, 올바르게는 "나중에 시작한", "적은 갱신을 수행한"이다.

② X
ㄱ의 "먼저 시작한 트랜잭션"이 틀렸으므로 이 선택지는 성립하지 않는다.
ㄹ의 "적은 갱신을 수행한 트랜잭션"은 맞지만, 짝이 되는 기준이 잘못되었다.
틀린 표현은 "먼저 시작한"이고, 올바르게는 "나중에 시작한"이다.

③ X
ㄴ의 "나중에 시작한 트랜잭션"은 맞다. 최근에 시작한 트랜잭션은 누적 작업량이 상대적으로 적어 철회 부담이 작다.
하지만 ㄷ의 "많은 갱신을 수행한 트랜잭션"이 틀렸다. 갱신량이 많을수록 롤백 비용이 커진다.
틀린 표현은 "많은 갱신을 수행한"이고, 올바르게는 "적은 갱신을 수행한"이다.

④ O
ㄴ의 "나중에 시작한 트랜잭션"과 ㄹ의 "적은 갱신을 수행한 트랜잭션"은 모두 희생자 선택의 올바른 기준이다.
둘 다 철회 시 손실되는 작업량과 복구 부담을 줄이는 방향이므로 교착상태 해소 비용을 최소화한다.

오답 포인트

희생자 선택을 "오래 기다린 트랜잭션을 먼저 살린다"라는 직관으로만 보면 ㄱ을 고르기 쉽다. 하지만 기준은 공정성이 아니라 롤백 비용 최소화이므로, 오래 실행했거나 많이 갱신한 트랜잭션은 오히려 철회 대상에서 뒤로 밀린다.

다시 나오면 이렇게 판단

교착상태 희생자 선택 문제는 "누구를 취소해야 되돌릴 비용이 가장 작은가"로 판단한다. 시작 시점은 더 최근인 쪽, 작업량은 더 적은 쪽을 고르면 정답 방향이 맞다.

암기 포인트

희생자는 최근 시작, 적게 갱신이다.

NoteBase

문 15. <보기>와 같은 함수적 종속성 집합 F를 가지는 릴레이션 R(A, B, C, D, E)을 여러 릴레이션으로 분해하고자 할 때 이에 대한 설명으로 가장 옳지 않은 것은?

<보기>

F = {A → B, C → D, D → AE}

① R에서 모든 속성은 C에 함수적으로 종속될 수 있다.

② R1(A, B), R2(C, D, E), R3(D, A)로 분해할 경우, R1, R2, R3는 제3정규형을 만족한다.

③ R1(D, A, B), R2(C, D, E)로 분해할 경우, R1, R2는 제2정규형을 만족한다.

④ R1(C, D), R2(D, A, E), R3(A, B)로 분해할 경우, R1, R2, R3는 BCNF를 만족한다.

정답 2번

출제 포인트

이 문제는 함수적 종속성으로 후보키를 구하고, 분해된 릴레이션이 제2정규형·제3정규형·BCNF를 만족하는지 구별하는 문제이다.

핵심 개념

후보키는 모든 속성을 함수적으로 결정하는 최소 속성 집합이다. 이 문제에서는 $C \to D, D \to A, E, A \to B$이므로 $C^+ = \{A, B, C, D, E\}$가 되어 C가 후보키이다.
제2정규형은 부분 함수 종속이 없는 상태이고, 기본키가 단일 속성이면 부분 함수 종속 문제가 생기지 않는다.
제3정규형은 비자명 함수 종속 $X \to Y$에서 X가 슈퍼키이거나 Y가 주요 속성이어야 한다. BCNF는 이보다 강해서 모든 결정자 X가 반드시 슈퍼키여야 한다.

NoteBase

선지별 해설

① O
$C \to D, D \to A, E, A \to B$를 연쇄 적용하면 C가 A, B, D, E를 모두 결정한다. 따라서 C는 R의 모든 속성을 함수적으로 결정할 수 있다.

② X
R1(A, B)에서는 $A \to B$이고 A가 키이므로 제3정규형을 만족한다. R3(D, A)에서는 $D \to A$이고 D가 키이므로 역시 제3정규형을 만족한다.
그러나 R2(C, D, E)에서는 $C \to D$와 $D \to E$가 성립한다. 여기서 D는 슈퍼키가 아니고 E는 주요 속성도 아니므로 $D \to E$가 제3정규형을 위반한다.
틀린 표현은 "R1, R2, R3는 제3정규형을 만족한다"이다.
올바르게는 "R1과 R3는 제3정규형이지만 R2는 제3정규형을 만족하지 않는다"이다.

③ O
R1(D, A, B)은 D가 단일 키이므로 부분 함수 종속이 없어 제2정규형을 만족한다. $A \to B$는 이행 종속이라 제3정규형 판단에는 영향을 주지만, 제2정규형 위반 사유는 아니다.
R2(C, D, E)도 C가 단일 후보키이므로 부분 함수 종속이 없어 제2정규형을 만족한다.

④ O
R1(C, D)는 $C \to D$에서 C가 키이므로 BCNF이다. R2(D, A, E)는 $D \to A, E$에서 D가 키이므로 BCNF이다. R3(A, B)도 $A \to B$에서 A가 키이므로 BCNF이다.

오답 포인트

②는 R2(C, D, E)에서 남는 $D \to E$를 놓치면 쉽게 속는다. 많은 수험생이 단일 키이면 곧바로 제3정규형까지 만족한다고 착각하거나, 이행 종속과 부분 함수 종속을 섞어서 판단한다.

다시 나오면 이렇게 판단

분해 문제는 먼저 각 릴레이션에 투영되는 함수적 종속성을 적고 후보키를 구한 뒤 정규형을 판정하면 된다.
제2정규형은 부분 함수 종속만, 제3정규형은 이행 종속까지, BCNF는 모든 결정자가 키인지까지 본다.

암기 포인트

단일 키 릴레이션은 제2정규형이 쉽게 성립하고, 제3정규형은 이행 종속을, BCNF는 결정자 전부가 키인지 점검하는 단계이다.

NoteBase

문 16. <보기>의 두 트랜잭션 T_1, T_2를 적절한 제어 없이 동시에 수행할 때 발생할 수 있는 상황에 대한 설명으로 가장 옳지 않은 것은?

<보기>

T_1	T_2
read_item(X);	read_item(X);
X = X − 5;	X = X * 2;
read_item(Y);	write_item(X);
write_item(X);	read_item(Y);
Y = Y + 5;	read_item(X);
write_item(Y);	

① 두 트랜잭션이 끝난 후 Y의 값에 오류가 발생할 가능성은 없다.

② 초기 주어진 X의 값이 10이라면 T_1의 read_item(X)에서 읽어 들인 X의 값은 10일 수도 있고 20일 수도 있다.

③ 초기 주어진 X의 값이 10이라면 두 트랜잭션이 끝난 후 X의 값은 10 또는 15가 될 것이다.

④ 초기 주어진 Y의 값이 10이라면 T_2의 read_item(Y)에서 읽어 들인 Y의 값은 10일 수도 있고 15일 수도 있다.

정답 3번

출제 포인트

이 문제는 동시성 제어 없이 두 트랜잭션이 인터리빙될 때 각 데이터 항목의 최종값과 읽기 결과가 어떻게 달라지는지를 사례에 대입해 판단할 수 있는지를 묻는다.

사례 정리

T1은 X에서 5를 빼고 그 5를 Y에 더하는 이체 형태의 트랜잭션이다.
T2는 X를 읽어 2배로 만든 뒤 X에 다시 기록하고, 그 후 Y와 X를 다시 읽는다.
X는 T1과 T2가 모두 읽고 쓰지만, Y는 T1만 쓰고 T2는 읽기만 한다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2020 · 10월

핵심 개념

동시성 제어가 없으면 같은 항목 X에 대한 두 트랜잭션의 write 순서에 따라 갱신 손실이 발생할 수 있다.
반면 어떤 항목을 한 트랜잭션만 갱신한다면 다른 트랜잭션이 그 항목을 읽기만 하는 경우 최종 저장값 자체는 흔들리지 않는다.

적용 기준

각 선지는 누가 그 항목을 write하는지와 read 시점이 write 전인지 후인지를 나누어 보면 판단된다.
최종값을 묻는 선지는 마지막 write가 누구 것인지 확인하고, 읽은 값을 묻는 선지는 해당 read_item이 어느 write 앞뒤에 놓일 수 있는지를 확인하면 된다.

선지별 해설

① O: Y는 T1만 write_item(Y) 한다. T2는 Y를 읽기만 하므로 인터리빙이 어떻게 되더라도 두 트랜잭션 종료 후 최종 Y는 항상 초기값보다 5 큰 값이 된다.
② O: 초기 X가 10일 때 T1의 read_item(X)가 T2의 write_item(X) 전에 실행되면 10을 읽는다. 반대로 T2가 먼저 X를 20으로 기록한 뒤 T1이 읽으면 20을 읽을 수 있다.
③ X: 틀린 표현은 '두 트랜잭션이 끝난 후 X의 값은 10 또는 15가 될 것이다'이다. 올바르게는 write_item(X)의 실행 순서에 따라 최종 X는 5, 10, 15, 20이 모두 가능하다. 예를 들어 T1이 10을 읽고 T2도 10을 읽은 뒤 T2가 20을 쓰고 마지막에 T1이 5를 쓰면 최종 X는 5가 된다. T1이 먼저 5를 쓴 뒤 T2가 그 5를 읽어 10을 쓰면 최종 X는 10이 된다. T2가 먼저 20을 쓴 뒤 T1이 읽어 15를 쓰면 최종 X는 15가 된다. T1과 T2가 모두 10을 읽고 마지막에 T2가 20을 쓰면 최종 X는 20이 된다.
④ O: 초기 Y가 10일 때 T2의 read_item(Y)가 T1의 write_item(Y) 전에 실행되면 10을 읽고, T1이 Y를 15로 기록한 뒤 실행되면 15를 읽을 수 있다.

오답 포인트

③은 X의 가능한 최종값을 너무 좁게 본 것이 함정이다. X는 두 트랜잭션이 모두 write하는 항목이므로 read 시점뿐 아니라 마지막 write의 주체까지 함께 따져야 한다.

다시 나오면 이렇게 판단

동시 수행 문제는 항목별로 읽기 전용인지, 읽기 후 쓰기인지, 누가 마지막으로 쓰는지를 분리해 본다. 한 항목을 둘 다 쓰면 가능한 최종값의 경우의 수를 열어 두고, 한쪽만 쓰면 읽기 시점만 따져도 된다.

암기 포인트

동시에 둘 다 쓰는 항목은 최종값이 흔들리고, 한쪽만 쓰는 항목은 최종 저장값보다 읽기 시점 차이가 핵심이다.

NoteBase

문 17. NoSQL 개발 동기가 된 기존의 관계형 데이터베이스 관리시스템(RDBMS)의 한계에 대한 설명으로 가장 옳지 않은 것은?

① 애초에 테이블을 분할(partition)하여 저장하도록 설계되지 않아 분산처리에 어려움이 있을 수 있다.

② 원본 데이터 구조가 테이블 형태가 아닐 경우 변환에 어려움이 있을 수 있다.

③ SQL 언어는 테이블 형태가 아닌 데이터에 적용하기에 어려움이 있을 수 있다.

④ 일반적으로 RDBMS는 일관성(consistency) 유지 기능이 부족하다.

정답 4번

출제 포인트

이 문제는 RDBMS와 NoSQL의 차이 중에서 관계형 데이터베이스의 실제 한계와 강점을 정확히 구분하는지를 묻는다.

핵심 개념

RDBMS는 정형 데이터를 테이블과 SQL로 관리하며 트랜잭션과 일관성 보장을 강하게 지원하는 시스템이다. / 결정 차이축: 데이터 모델과 트랜잭션 보장 / 키워드: 테이블, SQL, ACID, 일관성
NoSQL은 대규모 분산 환경에서 확장성과 유연한 스키마를 중시하며 비정형·반정형 데이터 처리에 유리한 시스템이다. / 결정 차이축: 분산 확장성과 스키마 유연성 / 키워드: 분산, 파티셔닝, 유연한 스키마

NoteBase

선지별 해설

① O
애초의 전통적 RDBMS는 단일 서버 중심의 테이블 저장과 트랜잭션 처리를 전제로 발전했으므로, 데이터를 처음부터 여러 노드에 수평 분할해 저장하는 구조에는 불리할 수 있다. 따라서 분산처리에 어려움이 있을 수 있다는 설명은 타당하다.

② O
RDBMS는 데이터를 행과 열의 테이블 구조에 맞추어 저장하므로, 문서형·그래프형·키-값형처럼 원본 구조가 테이블이 아닌 경우 관계형 스키마로 변환해야 하는 부담이 생긴다. 따라서 변환의 어려움이 있을 수 있다는 설명은 맞다.

③ O
SQL은 관계형 테이블 질의를 중심으로 설계된 언어이므로, 중첩 문서나 비정형 구조를 그대로 표현하고 조작하는 데에는 제약이 있다. 따라서 테이블 형태가 아닌 데이터에 적용하기 어려울 수 있다는 설명은 옳다.

④ X
틀린 부분은 "일반적으로 RDBMS는 일관성(consistency) 유지 기능이 부족하다"는 표현이다. 실제로는 RDBMS가 ACID 트랜잭션을 통해 일관성 유지에 강점을 가지며, 오히려 많은 NoSQL 시스템이 확장성과 가용성을 위해 일관성을 일부 완화한다.

오답 포인트

분산처리의 어려움과 일관성 보장 능력을 같은 축으로 혼동하면 ④에 속기 쉽다. RDBMS의 대표적 약점은 수평 확장성과 비정형 데이터 처리 유연성이지, 일관성 유지 능력의 부족이 아니다.

다시 나오면 이렇게 판단

RDBMS의 한계를 묻는 문제에서는 테이블 중심 구조, 수평 확장성 부족, 비정형 데이터 변환 부담은 약점으로 본다. 반대로 트랜잭션과 일관성 보장은 RDBMS의 강점이므로, 이 강점을 약점처럼 바꿔 적은 선지는 틀린 선지로 판단한다.

암기 포인트

RDBMS는 테이블·SQL·ACID, NoSQL은 분산·확장성·유연 스키마로 정리된다.

NoteBase

문 18. (맥주→기저귀)의 관계에서 장바구니 분석(연관 규칙)의 지지도와 신뢰도를 구하는 데 필요한 정보가 아닌 것은?

① 맥주가 포함된 거래 횟수

② 맥주와 기저귀가 함께 거래된 횟수

③ 기저귀가 포함된 거래 횟수

④ 전체 거래 횟수

정답 3번

출제 포인트

이 문제는 연관 규칙에서 지지도와 신뢰도의 정의를 구분하고, 각 식에 어떤 거래 횟수가 들어가는지를 묻는 문제이다.

핵심 개념

지지도는 전체 거래 중 맥주와 기저귀가 함께 나타난 비율이므로 support(맥주→기저귀)=(count(맥주 기저귀))/(N)이다.
신뢰도는 맥주가 포함된 거래 중 기저귀도 함께 나타난 비율이므로 confidence(맥주→기저귀)=(count(맥주 기저귀))/(count(맥주))이다.
따라서 필요한 값은 전체 거래 횟수, 맥주 포함 거래 횟수, 맥주와 기저귀의 동시 거래 횟수이다.

선지별 해설

① X: '필요하지 않은 정보'라고 보면 틀리다. 올바르게는 신뢰도의 분모에 들어가는 필요한 정보이다.
② X: '필요하지 않은 정보'라고 보면 틀리다. 올바르게는 지지도와 신뢰도의 공통 분자에 들어가는 필요한 정보이다.
③ O: 기저귀가 포함된 거래 횟수만 따로 아는 값은 지지도와 신뢰도 식에 직접 들어가지 않는다. 이 값은 향상도처럼 다른 연관성 지표를 구할 때는 필요할 수 있지만, 이 문제에서 묻는 두 지표만으로는 필요하지 않다.
④ X: '필요하지 않은 정보'라고 보면 틀리다. 올바르게는 지지도의 분모에 들어가는 필요한 정보이다.

오답 포인트

신뢰도를 막연한 관련성으로 외우면 ③도 필요하다고 착각하기 쉽다. 신뢰도는 '맥주를 산 거래'를 기준으로 조건부 비율을 보는 값이므로, 후행항인 기저귀의 단독 거래 횟수는 직접 쓰지 않는다는 점이 함정이다.

다시 나오면 이렇게 판단

연관 규칙 문제는 먼저 지표의 분자와 분모를 식으로 적는다. 그 식에 직접 들어가지 않는 거래 횟수는 해당 지표를 구하는 데 필요하지 않은 정보이다.

NoteBase

암기 포인트

지지도는 전체 대비 동시구매, 신뢰도는 선행항 대비 동시구매이다.

NoteBase

문 19. <보기>의 릴레이션 R과 S에 대하여 주어진 관계 대수식을 수행하려고 한다. 관계 대수식의 수행 결과로 가장 옳지 않은 것은?

<보기>

R

sid	name	address	year
1	문재훈	광양	7
2	김형진	대구	6
3	유제현	서울	9
4	박진우	서울	8
5	김현호	서울	2
6	정승완	서울	5
7	박성욱	부산	1

S

sid	score	dept
1	80	100
2	80	200
3	90	300
4	70	400
6	100	500

① _address='서울'(R) - _year≥ 6(R)

sid	name	address	year
5	김현호	서울	2
6	정승완	서울	5

② $\pi_{\text{name, score}}(\sigma_{\text{score} \geq 80 \text{ and } \text{year} > 5}(R \bowtie S))$

name	score
문재훈	80
김형진	80
유제현	90
정승완	100

③ $\pi_{\text{name, address, year, score}}(\sigma_{\text{year} \geq 5}(R \bowtie S))$

name	address	year	score
문재훈	광양	7	80
김형진	대구	6	80
유제현	서울	9	90
박진우	서울	8	70
정승완	서울	5	100

NoteBase

④ $\pi_{\text{score, dept}}(\sigma_{\text{score} \geq 90}(S))$

score	dept
90	300
100	500

정답 2번

출제 포인트

이 문제는 자연조인 결과를 만든 뒤 선택, 사영, 차집합을 순서대로 적용하여 결과 릴레이션이 맞는지 판별하는 문제이다.

자료 읽는 순서

1. R과 S를 `sid`로 자연조인하면 공통 `sid`인 1, 2, 3, 4, 6만 남는다.
2. 각 식의 선택 조건을 확인한다. `year>5`와 `year≥5`는 결과가 다르고, `score≥80` 또는 `score≥90` 조건도 함께 봐야 한다.
3. 마지막에 사영이나 차집합을 적용하여 제시된 결과표와 일치하는지 비교한다.

선지별 해설

① O: `σ_address='서울'(R)`의 결과는 sid 3, 4, 5, 6이고, `σ_year≥6(R)`의 결과는 sid 1, 2, 3, 4이다. 차집합을 취하면 sid 5, 6만 남으므로 제시된 결과와 일치한다.
② X: `R⋈S` 후 `score≥80 and year>5`를 적용하면 sid 1, 2, 3만 남는다. sid 6은 score 100이지만 year가 5이므로 `year>5`를 만족하지 못한다. 틀린 표현은 결과에 `정승완 100`이 포함된 부분이다. 올바르게는 `문재훈 80`, `김형진 80`, `유제현 90`만 남는다.
③ O: `R⋈S` 후 `year≥5`를 적용하면 sid 1, 2, 3, 4, 6이 남는다. 이를 `name, address, year, score`로 사영하면 제시된 다섯 행과 정확히 같다.
④ O: S에서 `score≥90`인 튜플은 sid 3과 6뿐이다. 이를 `score, dept`로 사영하면 `(90, 300)`, `(100, 500)`이 되어 제시 결과와 일치한다.

오답 포인트

②는 `year>5`를 `year≥5`처럼 읽으면 쉽게 속는다. 또한 조인을 먼저 수행한 뒤 선택을 적용해야 하는데, 이 순서를 흐리게 보면 sid 6을 잘못 포함시키기 쉽다.

다시 나오면 이렇게 판단

관계대수식은 `조인 또는 원본 릴레이션 결정 → 선택 조건 적용 → 사영 또는 차집합 적용` 순서로 중간 결과를 직접 적어 보면 된다. 특히 경계값이 있는 조건은 `>`와 `≥`를 먼저 구분하면 오답을 줄일 수 있다.

NoteBase

암기 포인트

자연조인 후 선택을 적용하고, 경계값 조건은 `>`와 `≥`를 먼저 구분한다.

NoteBase

문 20. <보기 1>은 학생들에 대한 정보를 담고 있는 WORK_INFO 테이블이다. <보기 2>의 SQL 질의를 수행할 때 예상되는 출력 결과는?

<보기 1>

Name	D_name	Salary
심종화	engineering_building	250
이준석	information_building	240
김성희	engineering_building	260
한상유	engineering_building	210
김태의	science_building	310
유상민	computer_building	290
김소희	information_building	220
김은빈	science_building	180
조성국	computer_building	300

<보기 2>

```
CREATE VIEW DEPT_INFO
            (Dept_name, No_of_emps, Total_sal)
AS SELECT D_name, COUNT(*), SUM(Salary)
FROM WORK_INFO
GROUP BY D_name
```

① DEPT_INFO

COUNT(*)	SUM(Salary)
3	720
2	460
2	490
2	590

NoteBase

② DEPT_INFO

D_name	No_of_emps	Total_sal
engineering_building	3	720
information_building	2	460
science_building	2	490
computer_building	2	590

③ DEPT_INFO

Dept_name	COUNT(*)
engineering_building	3
information_building	2
science_building	2
computer_building	2

④ DEPT_INFO

Dept_name	No_of_emps	Total_sal
engineering_building	3	720
information_building	2	460
science_building	2	490
computer_building	2	590

정답 4번

출제 포인트

이 문제는 CREATE VIEW에서 지정한 열 이름과 GROUP BY 집계 결과를 함께 읽을 수 있는지를 묻는다.

자료 읽는 순서

1. WORK_INFO를 D_name별로 묶어 부서별 행 수와 Salary 합계를 계산한다.
2. CREATE VIEW DEPT_INFO (Dept_name, No_of_emps, Total_sal) 구문에서 최종 열 이름이 따로 지정되었는지 확인한다.
3. 집계값과 열 이름이 모두 맞는 결과표를 고른다.

NoteBase

선지별 해설

1번: X. engineering_building은 3명 720, information_building은 2명 460, science_building은 2명 490, computer_building은 2명 590으로 집계값 자체는 맞다. 그러나 결과표에서 부서명을 나타내는 열이 빠져 있어 GROUP BY 결과의 첫 번째 열인 부서 구분값이 누락되었다. 틀린 표현은 부서명 없이 COUNT(*)와 SUM(Salary)만 출력된 부분이며, 올바르게는 부서명 열까지 포함되어야 한다.
2번: X. 행별 집계값은 맞지만 VIEW 정의에서 열 이름을 (Dept_name, No_of_emps, Total_sal)로 지정했으므로 최종 열 이름은 D_name, COUNT(*), SUM(Salary)가 아니라 Dept_name, No_of_emps, Total_sal이다. 틀린 표현은 첫 번째 열 이름을 D_name으로 둔 부분이며, 올바르게는 Dept_name이어야 한다.
3번: X. VIEW 정의에서 세 개의 열을 만들고 SELECT도 D_name, COUNT(*), SUM(Salary)의 세 열을 반환하므로 결과표도 세 열이어야 한다. 틀린 표현은 Total_sal 열이 빠진 부분이며, 올바르게는 부서명, 인원 수, 급여 합계의 세 열이 모두 있어야 한다.
4번: O. D_name별 그룹화 결과가 engineering_building 3명 720, information_building 2명 460, science_building 2명 490, computer_building 2명 590으로 계산되고, VIEW의 열 이름도 Dept_name, No_of_emps, Total_sal로 재지정되므로 이 표가 맞다.

오답 포인트

집계 함수 COUNT(*)와 SUM(Salary)만 보고 값이 맞으면 정답으로 고르기 쉽다. 그러나 CREATE VIEW 뒤의 열 이름 목록이 SELECT 결과 열 이름을 덮어쓴다는 점과 GROUP BY 결과에는 그룹 기준 열도 함께 나온다는 점을 놓치면 ②나 ①로 흔들린다.

다시 나오면 이렇게 판단

GROUP BY 문제는 먼저 그룹 기준 열, 집계 열의 개수, 집계값을 계산하고, 그다음 CREATE VIEW나 별칭에서 최종 열 이름이 바뀌는지 확인한다. 값과 열 이름이 모두 맞아야 정답이다.

암기 포인트

CREATE VIEW의 열 목록은 최종 열 이름을 정하고, GROUP BY 결과는 그룹 기준 열과 집계 열이 함께 나온다.

NoteBase