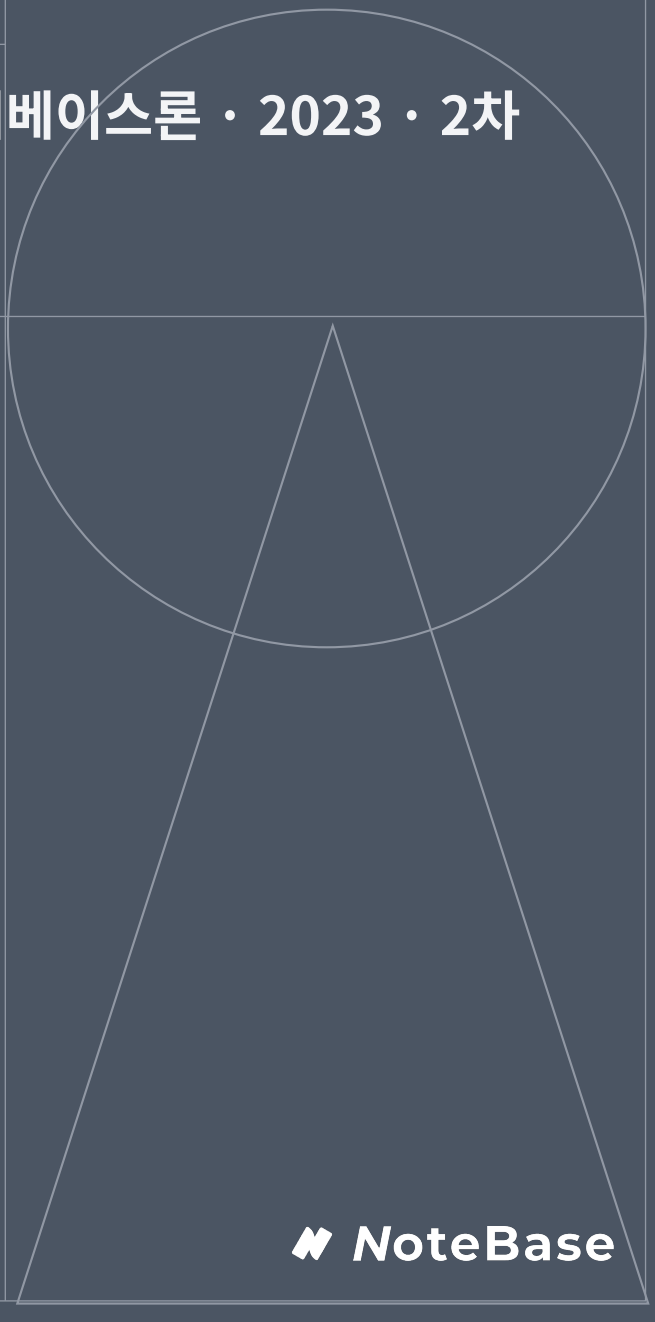
기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2023 · 2차



NoteBase

기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2023 · 2차

공무원/국가직/7급 · 데이터베이스론 · 2023 · 2차

문 1. 데이터 독립성에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 물리적 데이터 독립성은 데이터의 물리적 구조가 변경될 때, 데이터의 논리적 구조도 이에 따라 자동으로 변경되게 한다.
ㄴ. 응용프로그램이나 데이터베이스의 논리적 구조에 독립적으로 데이터의 물리적 구조를 변경할 수 있다.
ㄷ. 논리적 데이터 독립성은 개념 스키마가 변경되어도 외부 스키마나 응용프로그램에 영향을 끼치지 않는 것을 의미한다.
ㄹ. 3단계 데이터베이스 구조의 내부 단계에서는 파일에 데이터를 효율적으로 저장하고 관리하기 위하여 여러 개의 내부 스키마를 가지고 있다.

① ㄱ, ㄴ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄷ, ㄹ

정답 3번

출제 포인트

물리적 데이터 독립성과 논리적 데이터 독립성의 의미, 그리고 3단계 데이터베이스 구조에서 외부 스키마·개념 스키마·내부 스키마의 관계를 구분하는 문제이다.

핵심 개념

물리적 데이터 독립성은 내부 스키마가 바뀌어도 개념 스키마가 영향을 받지 않는 성질이다.
논리적 데이터 독립성은 개념 스키마가 바뀌어도 외부 스키마나 응용프로그램이 영향을 받지 않도록 하는 성질이다.
3단계 데이터베이스 구조에서는 외부 스키마는 여러 개일 수 있고, 개념 스키마와 내부 스키마는 각각 하나이다.

NoteBase

선지별 해설

1번 X: ㄴ은 옳지만 ㄱ이 틀려서 오답이다. 물리적 데이터 독립성은 물리적 구조가 변경되어도 논리적 구조가 함께 바뀌는 것이 아니라, 물리적 구조의 변경이 논리적 구조에 영향을 주지 않는다는 뜻이다. 틀린 표현은 "논리적 구조도 이에 따라 자동으로 변경"이고, 올바르게는 "논리적 구조는 변경되지 않는다"이다.
2번 X: ㄱ과 ㄹ이 모두 틀려서 오답이다. ㄱ의 틀린 표현은 앞과 같고, ㄹ의 틀린 표현은 "여러 개의 내부 스키마"이다. 올바르게는 "내부 스키마는 하나이며, 여러 개일 수 있는 것은 외부 스키마"이다.
3번 O: ㄴ과 ㄷ이 모두 옳아서 정답이다. ㄴ은 응용프로그램이나 데이터베이스의 논리적 구조와 독립적으로 저장 방식, 파일 조직, 접근 경로 같은 물리적 구조를 바꿀 수 있다는 뜻이므로 옳다. ㄷ은 개념 스키마가 변경되어도 외부 스키마나 응용프로그램에 영향을 주지 않는다는 논리적 데이터 독립성의 설명이므로 옳다.
4번 X: ㄷ은 옳지만 ㄹ이 틀려서 오답이다. ㄹ의 틀린 표현은 "여러 개의 내부 스키마"이고, 올바르게는 "여러 개의 외부 스키마, 하나의 내부 스키마"이다.

오답 포인트

데이터 독립성 문제에서는 변경되는 층과 영향을 받지 않는 층을 서로 뒤바꾸어 제시하는 함정이 자주 나온다. ㄱ은 물리적 구조 변경이 논리적 구조를 바꾼다고 서술해 방향을 뒤집었고, ㄹ은 여러 개가 가능한 스키마를 내부 스키마로 바꾸어 놓은 것이 함정이다.

다시 나오면 이렇게 판단

먼저 어느 층이 바뀌는지를 본 뒤, 어떤 층이 영향을 받지 않아야 하는지를 연결해서 판단한다. 내부 스키마 변경에 개념 스키마가 영향을 받지 않으면 물리적 데이터 독립성이고, 개념 스키마 변경에 외부 스키마와 응용프로그램이 영향을 받지 않으면 논리적 데이터 독립성이다. 스키마 개수는 외부 여러 개, 개념 하나, 내부 하나로 정리된다.

암기 포인트

물리 독립성은 내부 변경에도 개념 유지, 논리 독립성은 개념 변경에도 외부 유지, 스키마 수는 외부 다수·개념 1·내부 1이다.

NoteBase

문 2. 다음에서 설명하는 데이터베이스 용어는?

◯ 데이터베이스의 갱신이 발생할 때 DBMS가 자동으로 수행하는 사용자 정의 프로시저이다.
◯ 무한 호출에 따라 실행의 종료를 보장할 수 없는 문제가 있다.
◯ 무결성 제약조건을 유지하기 위하여 데이터베이스 갱신을 모니터링하고, 데이터베이스 갱신을 전파한다.

① assertion
② catalog
③ check
④ trigger

정답 4번

출제 포인트

이 문제는 트리거의 정의와 assertion, catalog, check의 차이를 구별하는지를 묻는 문제이다.

핵심 개념

trigger: 데이터베이스의 갱신 이벤트가 발생할 때 자동으로 실행되는 사용자 정의 프로시저이다. 키워드는 이벤트, 자동 수행, 후속 동작이다.
assertion: 데이터베이스 전체가 항상 만족해야 할 무결성 제약을 선언하는 것이다. 키워드는 조건 선언, 전역 제약이다.
catalog: 테이블, 인덱스, 사용자, 권한 같은 메타데이터를 저장한 데이터 사전이다. 키워드는 메타데이터, 시스템 정보이다.
check: 속성값이나 튜플이 조건식을 만족하는지 검사하는 제약조건이다. 키워드는 조건식 검사, 값 제한이다.

NoteBase

선지별 해설

① X
assertion은 데이터베이스가 만족해야 할 무결성 조건을 선언한 것이다. 자동으로 수행되는 사용자 정의 프로시저가 아니라는 점에서 문제의 설명과 맞지 않는다. 틀린 대응은 assertion이다. 올바르게는 이 설명은 trigger이다.

② X
catalog는 데이터베이스의 구조와 권한 같은 메타데이터를 저장하는 곳이다. 갱신 시 자동 실행되거나 무한 호출 문제가 생기는 절차가 아니다. 틀린 대응은 catalog이다. 올바르게는 이 설명은 trigger이다.

③ X
check는 값이 주어진 조건을 만족하는지 검사하는 제약조건이다. 무결성 유지와 관련은 있지만, 갱신을 모니터링하여 다른 갱신을 전파하는 사용자 정의 프로시저는 아니다. 틀린 대응은 check이다. 올바르게는 이 설명은 trigger이다.

④ O
trigger는 INSERT, UPDATE, DELETE 같은 갱신이 발생할 때 DBMS가 자동으로 수행하는 사용자 정의 프로시저이다. 무결성 제약조건 유지를 위해 갱신을 감시하고 후속 갱신을 전파할 수 있으며, 상호 호출이 반복되면 무한 호출로 종료가 보장되지 않는 문제가 생길 수 있다.

오답 포인트

무결성 관련 용어를 모두 제약조건으로만 묶어 외우면 check, assertion, trigger를 서로 바꾸어 고르기 쉽다. check와 assertion은 조건을 선언하는 개념이고, trigger는 이벤트가 발생했을 때 실행되는 절차라는 점이 핵심 함정이다.

다시 나오면 이렇게 판단

설명에 갱신 발생, 자동 수행, 사용자 정의 프로시저, 후속 동작 전파가 함께 나오면 trigger로 판단한다. 조건식이나 제약의 선언이면 check 또는 assertion이고, 데이터베이스의 구조 정보를 저장하면 catalog이다.

암기 포인트

trigger는 사건이 생기면 발동되는 절차이고, check와 assertion은 조건 선언이며, catalog는 메타데이터 저장소이다.

NoteBase

문 3. 릴레이션 R(A, B, C, D, E, F)에 대한 함수적 종속성(functional dependency)의 집합 F가 다음과 같이 주어졌을 때, 기본 키는?

F = { A→BC, E→CF, B→E, D→E }

① A

② AB

③ AD

④ AE

정답 3번

출제 포인트

이 문제는 함수적 종속성으로 각 속성 집합의 폐포를 구해 모든 속성을 결정하는 최소 집합, 즉 기본 키를 판별하는 문제이다.

핵심 개념

기본 키는 튜플을 유일하게 식별할 수 있는 후보 키 중 대표로 선택된 키이며, 이 문제에서는 모든 속성 A, B, C, D, E, F를 결정하는 최소 속성 집합을 찾으면 된다.
속성 폐포는 어떤 속성 집합으로 함수적 종속성을 반복 적용해 결정할 수 있는 모든 속성의 집합이다.
주어진 종속성에서는 $A \rightarrow BC$, $B \rightarrow E$, $E \rightarrow CF$, $D \rightarrow E$이므로 A가 있으면 B, C, E, F까지 얻고, D가 있어야만 D 자체를 포함한 전체 속성으로 닫힐 수 있다.

NoteBase

선지별 해설

① A: X
A의 폐포는 처음에 A이고, $A \to BC$로 B, C, $B \to E$로 E, $E \to CF$로 F를 얻어 $A^+ = \{A, B, C, E, F\}$가 된다.
틀린 부분은 A만으로 기본 키가 된다는 판단이다. 실제로는 D를 결정할 수 없으므로 A는 기본 키가 아니고, 올바르게는 D를 포함한 AD가 되어야 한다.

② AB: X
AB의 폐포도 A가 이미 B, C, E, F를 만들어 내므로 $AB^+ = \{A, B, C, E, F\}$이다.
틀린 부분은 B를 추가하면 기본 키가 된다고 본 점이다. 실제로는 여전히 D를 결정하지 못하므로 AB는 기본 키가 아니고, 올바르게는 D를 포함한 AD가 되어야 한다.

③ AD: O
AD의 폐포는 $A \to BC$로 B, C, $B \to E$와 $D \to E$로 E, $E \to CF$로 F를 얻어 $AD^+ = \{A, B, C, D, E, F\}$가 된다.
A만으로는 D가 없고 D만으로는 A,B를 만들 수 없으므로 AD는 모든 속성을 결정하면서도 더 줄일 수 없는 최소 집합이다. 따라서 기본 키가 맞다.

④ AE: X
AE의 폐포는 A로부터 B, C, B로부터 E, E로부터 C, F를 얻어 $AE^+ = \{A, B, C, E, F\}$가 된다.
틀린 부분은 E가 D를 대신할 수 있다고 본 점이다. 실제로는 어떤 종속성도 E로부터 D를 만들지 못하므로 AE는 기본 키가 아니고, 올바르게는 AE가 아니라 AD가 되어야 한다.

오답 포인트

A가 많은 속성을 결정하므로 ①이나 ②, ④를 고르기 쉽다. 그러나 기본 키 판단에서는 많이 결정하는지보다 빠진 속성이 있는지가 더 중요하다. 이 문항에서는 어떤 선택지든 D를 만들어 낼 수 없으면 탈락이며, D를 포함한다고 해서 자동으로 정답이 되는 것도 아니므로 최소성까지 함께 확인해야 한다.

다시 나오면 이렇게 판단

각 선택지마다 폐포를 직접 계산해 전체 속성이 모두 나오는지 먼저 본다. 전체 속성이 나오면 그다음에 한 속성씩 빼 보아도 여전히 전체가 결정되는지 확인해 최소성을 판단하면 된다. 이 문제처럼 다른 속성으로 절대 생성되지 않는 속성이 있으면 그 속성은 기본 키에 반드시 포함될 가능성이 높다.

암기 포인트

기본 키 판별은 폐포로 전체 속성 확인, 그다음 최소성 확인의 두 단계이다. 이 문항은 A가 B, C, E, F를 만들고 D는 따로 챙겨야 하므로 AD가 핵심이다.

NoteBase

문 4. 다음 릴레이션에서 키에 대한 설명으로 옳지 않은 것은? (단, 밑줄 친 속성은 기본 키이고, 고객 릴레이션에서 주민등록번호 속성의 값은 중복되지 않는다)

고객

고객ID	고객이름	주민등록번호	등급	적립금	추천고객
K123	김탁구	96××××-×××××××	Gold	2000	P492
L243	이승룡	97××××-×××××××	Vip	3000	K123
C524	최무선	93××××-×××××××	Gold	2000	K123
P492	박하얀	95××××-×××××××	Silver	1000	L243

주문

주문번호	주문고객	주문제품	수량	단가	주문일자
1	K123	우동	10	2000	2023 - 01 - 01
2	C524	라면	5	1000	2023 - 01 - 02
3	P492	만두	7	3000	2023 - 01 - 03

① 고객 릴레이션에서 고객ID, 주민등록번호, (고객ID, 고객이름)의 쌍, (고객이름, 주민등록번호)의 쌍은 슈퍼키가 될 수 있다.

② 고객 릴레이션에서 추천고객 속성은 자신의 고객ID 속성을 참조하는 외래 키로 정의가 가능하며, NULL 값을 가질 수 있다.

③ 주문 릴레이션의 주문고객 속성을 고객 릴레이션의 고객ID에 대한 외래 키로 정의할 때, 주문고객의 값으로 ‘S254’를 가질 수 있다.

④ 고객 릴레이션에서 고객ID, 주민등록번호는 후보 키가 될 수 있으나, (고객ID, 고객이름)의 쌍은 후보 키가 될 수 없다.

정답 3번

출제 포인트

이 문제는 슈퍼키와 후보키의 차이, 그리고 자기참조 외래키와 외래키 무결성 제약을 구별할 수 있는지를 묻는다.

NoteBase

핵심 개념

슈퍼키는 한 튜플을 유일하게 식별할 수 있는 속성 집합이다. 후보키는 슈퍼키 중에서 더 이상 속성을 줄일 수 없는 최소 키이다. 외래키는 다른 릴레이션의 기본키나 후보키 값을 참조해야 하므로, NULL이 아니라면 참조 대상에 실제 존재하는 값만 가질 수 있다.

선지별 해설

1번: O. 고객ID는 기본키이고 주민등록번호도 중복되지 않으므로 각각 튜플을 유일하게 식별할 수 있다. 따라서 고객ID, 주민등록번호는 슈퍼키가 된다. 또한 슈퍼키에 다른 속성을 더한 (고객ID, 고객이름)과 주민등록번호를 포함한 (고객이름, 주민등록번호)도 슈퍼키가 될 수 있다.
2번: O. 추천고객은 같은 고객 릴레이션의 고객ID를 가리키는 자기참조 외래키로 정의할 수 있다. 추천한 고객이 아직 없으면 값이 없을 수 있으므로 NULL도 가질 수 있다.
3번: X. 틀린 부분은 '주문고객의 값으로 S254를 가질 수 있다'이다. 주문고객을 고객.고객ID에 대한 외래키로 정의하면 NULL이 아닌 값은 고객 릴레이션에 실제 존재하는 고객ID여야 한다. 제시된 고객ID는 K123, L243, C524, P492뿐이므로 S254는 가질 수 없다. 올바르게는 '주문고객의 값은 고객 릴레이션에 존재하는 고객ID만 가질 수 있다'이다.
4번: O. 고객ID와 주민등록번호는 각각 alone으로 튜플을 유일하게 식별하므로 후보키가 될 수 있다. 반면 (고객ID, 고객이름)은 고객ID만으로 이미 식별이 되므로 최소성이 없어서 후보키가 될 수 없다.

오답 포인트

외래키를 단순히 '다른 테이블과 연결되는 속성' 정도로만 기억하면 3번을 맞는 말처럼 보기 쉽다. 외래키는 연결만 하는 것이 아니라 참조 대상 릴레이션에 실제 존재하는 키값만 허용한다는 무결성 제약까지 함께 본다.

다시 나오면 이렇게 판단

먼저 유일성으로 슈퍼키 여부를 본다. 그다음 불필요한 속성을 제거해도 식별되면 후보키가 아니라고 판단한다. 외래키 선지는 NULL 허용 여부와 함께 '참조 대상에 실제 있는 값만 가능한가'를 확인하면 된다.

암기 포인트

슈퍼키는 유일성, 후보키는 유일성+최소성, 외래키는 참조 대상에 존재하는 값만 허용된다는 점이 핵심이다.

NoteBase

문 5. 다음 릴레이션 R과 S에 대한 관계대수식의 결과로 옳지 않은 것은? (단, 기호 $\bowtie_N$은 자연조인을 의미하고, 릴레이션에서 A, B, C, D는 속성 이름이다)

R		
A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c5

S		
B	C	D
b1	c1	d1
b3	c2	d2
b3	c3	d3
b5	c5	d5

① $\sigma_{B='b3'}(S)$

B	C	D
b3	c2	d2
b3	c3	d3

NoteBase

② $\pi_{B,C}(R \bowtie_N S)$

B	C
b1	c1
b3	c3

③ $\pi_{A,B}(\sigma_{C='c3'}(R))$

A	B
a3	b3

④ $\pi_{A,C}(\sigma_{B='b3'}(R \bowtie_N S))$

A	C
a2	c2

정답 4번

출제 포인트

이 문제는 릴레이션 표에서 선택, 투영, 자연조인의 결과를 정확히 읽는 문제이다.

자료 읽는 순서

1. 각 릴레이션의 속성과 튜플을 확인한다.
2. 자연조인은 공통 속성인 B와 C가 모두 같은 튜플만 결합된다는 점을 확인한다.
3. 선택으로 행을 거른 뒤 투영으로 필요한 열만 남겨 선지 결과와 비교한다.

핵심 근거

R과 S의 자연조인 결과는 공통 속성 B, C가 모두 일치하는 두 튜플뿐이다.
(a1, b1, c1, d1), (a3, b3, c3, d3)
이 결과를 기준으로 각 선지의 선택과 투영 결과를 확인하면 ④만 맞지 않는다.

NoteBase

선지별 해설

① O
S에서 B='b3'인 튜플을 선택하면 (b3, c2, d2), (b3, c3, d3) 두 행이 남으므로 제시된 결과와 같다.

② O
R ⋈N S는 B와 C가 모두 같은 경우만 결합하므로 (a1, b1, c1, d1), (a3, b3, c3, d3)만 나온다.
여기서 B, C만 투영하면 (b1, c1), (b3, c3)이므로 제시된 결과와 같다.

③ O
R에서 C='c3'인 튜플은 (a3, b3, c3) 하나이다.
여기서 A, B만 투영하면 (a3, b3)이므로 제시된 결과와 같다.

④ X
자연조인 결과에서 B='b3'인 튜플은 (a3, b3, c3, d3) 하나이다.
여기서 A, C만 투영하면 (a3, c3)이 되어야 한다.
선지의 (a2, c2)는 잘못된 표현이며, 올바르게는 (a3, c3)이다.

오답 포인트

자연조인을 공통 속성 하나만 보고 판단하면 오답이 된다. 이 문항에서는 B가 같은 값 b3이 있어도 C까지 함께 일치해야 조인되므로, S의 (b3, c2, d2)는 R과 결합되지 않는다.

다시 나오면 이렇게 판단

관계대수식은 선택으로 행을 줄이고, 조인에서 공통 속성 일치 조건을 확인한 뒤, 마지막에 투영으로 열만 남기는 순서로 판단하면 된다. 자연조인은 공통 속성을 모두 비교한다는 점이 핵심이다.

암기 포인트

자연조인 = 이름이 같은 모든 공통 속성이 동시에 일치하는 튜플만 결합이다.

NoteBase

문 6. 크기가 100바이트인 고정길이 레코드 630개로 구성한 정렬된 순차파일(sequential file)에서 이진검색으로 레코드를 검색할 때, 최대로 접근하는 블록의 수는? (단, 블록의 크기는 1,024바이트이며, 하나의 레코드가 두 개 이상의 블록에 나누어 저장되지 않는다고 가정한다)

① 6

② 10

③ 64

④ 320

정답 1번

출제 포인트

이 문제는 블록당 저장 가능한 레코드 수를 먼저 구한 뒤, 전체 블록 수에 대해 이진검색의 최대 접근 횟수를 계산하는 문제이다.

조건 정리

주어진 값:
- 레코드 크기: 100바이트
- 레코드 수: 630개
- 블록 크기: 1,024바이트

구해야 할 값:
- 이진검색 시 최대 접근 블록 수

주의할 조건:
- 레코드는 두 블록에 걸쳐 저장되지 않으므로 한 블록에는 최대 10개만 저장된다.

사용할 공식

블록당 레코드 수는 $\left\lfloor \frac{1024}{100} \right\rfloor = 10$개이다.
전체 블록 수는 $\left\lceil \frac{630}{10} \right\rceil = 63$개이다.
이진검색의 최대 접근 블록 수는 $\lceil \log_2 63 \rceil = 6$이다.
이 공식을 쓰는 이유는 정렬된 순차 파일을 이진검색할 때 비교 대상이 레코드 하나하나가 아니라 블록 단위로 줄어들기 때문이다.

NoteBase

계산 과정

1. 한 블록에 저장 가능한 레코드 수를 구하면 $1024 \div 100 = 10.24$이므로 최대 10개이다.
2. 전체 630개 레코드를 저장하는 데 필요한 블록 수는 $630 \div 10 = 63$이므로 63블록이다.
3. 63블록에 대해 이진검색을 하면 탐색 범위가 $63 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$로 줄어드므로 최대 접근 횟수는 6회이다.

선지별 해설

① O
- 전체 블록 수가 63개이므로 이진검색의 최대 접근 블록 수는 6이다.

② X
- 10은 한 블록에 들어가는 레코드 수이다.
- 틀린 표현은 접근 블록 수를 10으로 본 점이다. 올바르게는 접근 블록 수는 총 63블록에 대한 이진검색 결과인 6이다.

③ X
- 64는 63보다 크거나 같은 가장 가까운 2의 거듭제곱일 뿐, 실제 접근 횟수는 아니다.
- 틀린 표현은 64를 접근 블록 수로 본 점이다. 올바르게는 $2^6 = 64$이므로 최대 접근 횟수는 6회이다.

④ X
- 320은 순차검색에서 중간쯤을 떠올린 값에 가깝고, 이 문항의 이진검색 조건과 맞지 않는다.
- 틀린 표현은 선형적으로 블록을 세는 방식이다. 올바르게는 정렬 파일의 이진검색이므로 탐색 범위를 절반씩 줄여 6블록만 보면 된다.

오답 포인트

블록당 레코드 수 10개와 이진검색 접근 횟수 6회를 서로 혼동하기 쉽다. 또 63블록을 보고 가장 가까운 2의 거듭제곱인 64를 그대로 정답으로 고르는 함정도 자주 나온다.

다시 나오면 이렇게 판단

파일 접근 계산 문제는 블록당 저장 개수 계산, 전체 블록 수 계산, 그다음 탐색 방식에 맞는 접근 횟수 계산의 순서로 판단한다. 이진검색이면 마지막 단계에서 블록 수에 로그를 취한다.

암기 포인트

고정길이 레코드 파일 문제는 블록당 저장 개수부터 구하고, 이진검색 접근 횟수는 전체 블록 수의 $\log_2$값으로 판단한다.

NoteBase

문 7. 다음과 같은 조건을 모두 만족하는 B^+-트리에 T, W, M, E, F, G, K, C, S, D, B, L 순으로 키 값이 입력되었을 때, 루트(root)에 있는 키 값을 바르게 나열한 것은? (단, $[X_i]$는 포인터 P_i가 가리키는 서브트리의 모든 키 값들을 의미하고, 트리는 비어 있는(empty) 상태이며, 기호 ≡는 동치를 의미한다)

◯ 한 노드에 최대 네 개의 키 값을 가질 수 있다.
◯ 내부노드의 구조는 $< P_1, K_1, ..., P_i, K_i, ..., K_{q-1}, P_q >$이며, K_i는 i번째 키, P_i는 i번째 서브트리를 가리키는 포인터이다.
◯ $i \equiv 1$이면 $[X_1] < K_1$이고, $i \equiv q$이면 $K_{q-1} \leq [X_q]$를 만족한다.
◯ $1 < i < q$이면 $K_{i-1} \leq [X_i] < K_i$를 만족한다.
◯ 노드 분할 시 상위 노드로 올리는 값은 분할대상 키 값들 중 중간 값을 올리는 것으로 가정한다.

① C, K
② D, L
③ F, M
④ G, S

정답 3번

출제 포인트

한 노드에 최대 4개의 키를 두는 B+-트리에서 삽입 순서에 따라 어떤 중간값이 루트로 승격되는지를 추적하는 문제이다.

올바른 순서

1. T, W, M, E까지 삽입하면 루트 리프 노드는 [E, M, T, W]가 된다.
2. F를 삽입하면 [E, F, M, T, W]가 되어 overflow가 발생하고, 중간값 M이 상위로 승격되어 루트는 [M]이 된다.
3. G, K를 계속 삽입하면 M의 왼쪽 리프에 [E, F, G, K]가 쌓인다.
4. C를 삽입하면 [C, E, F, G, K]가 되어 다시 overflow가 발생하고, 중간값 F가 루트로 승격되어 루트는 [F, M]이 된다.
5. S, D, B, L을 차례로 넣어도 하위 노드만 조정될 뿐 루트는 더 이상 분할되지 않으므로 최종 루트 키는 F, M이다.

왜 이 순서: 이 문제는 overflow가 난 노드에서 중간값이 부모로 올라간다는 조건만 추적하면 된다. 루트를 만든 승격값은 먼저 M, 다음 F이다.

NoteBase

선지별 해설

① X
C는 왼쪽 리프 분할 직전에 들어간 값일 뿐 루트로 승격된 값이 아니다. K도 리프에 남는 값이다.
틀린 표현: "C, K"
올바르게는: "F, M"이다.

② X
D와 L은 후반부에 리프에 삽입되는 값이어서 루트 키가 되지 않는다. 루트는 분할 때 상위로 올라간 중간값으로 결정된다.
틀린 표현: "D, L"
올바르게는: "F, M"이다.

③ O
F 삽입으로 첫 분할이 일어날 때 M이 루트로 올라가고, 이후 C 삽입으로 왼쪽 리프가 다시 분할될 때 F가 루트에 추가된다. 따라서 최종 루트 키는 F, M이다.

④ X
G와 S는 각각 중간 리프와 오른쪽 리프에 저장되는 값이다. 둘 다 분할 시 승격된 중간값이 아니므로 루트 키가 아니다.
틀린 표현: "G, S"
올바르게는: "F, M"이다.

오답 포인트

루트 키를 최종적으로 리프에 남아 있는 값과 혼동하면 오답이 된다. 이 문항은 실제 저장 위치보다 분할 순간 부모로 올라가는 중간값이 무엇인지 추적하는지가 핵심 함정이다.

다시 나오면 이렇게 판단

B+-트리 삽입 문제는 모든 키를 끝까지 정렬하려고 하기보다 overflow가 발생한 시점만 표시하고, 그때 부모로 승격되는 중간값만 기록하면 빠르게 판단할 수 있다. 최종 루트는 마지막까지 루트에 올라온 승격값들의 집합으로 판단한다.

암기 포인트

B+-트리 루트 키는 리프의 끝값이 아니라 분할 때 올라간 중간값이다.

NoteBase

문 8. 데이터베이스 관리시스템(DBMS)에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 일반적으로 데이터베이스는 저장된 데이터가 많아 구조가 복잡하고, 동시 공유를 지원하므로 장애 발생에 대응해서 데이터를 일관된 상태로 회복하는 과정이 복잡하다.
ㄴ. DBMS의 필수 기능으로 정의 기능, 조작 기능, 제어 기능, 처리 기능이 있다.
ㄷ. DBMS의 정의 기능은 데이터를 정확하고 안전하게 유지하기 위한 데이터 무결성 유지, 보안 유지, 병행 제어 기능을 포함한다.
ㄹ. DBMS는 데이터의 중복성과 응용프로그램에 대한 종속성 문제 해결에서 파일 시스템 대비 우수한 장점을 가지고 있다.

① ㄱ, ㄴ

② ㄱ, ㄹ

③ ㄴ, ㄷ

④ ㄷ, ㄹ

정답 2번

출제 포인트

이 문제는 DBMS의 기본 기능을 정의 기능·조작 기능·제어 기능으로 구분하고, 파일 시스템 대비 DBMS의 장점을 판별할 수 있는지를 묻는다.

핵심 개념

정의 기능은 데이터베이스 구조, 데이터 형식, 제약조건을 명세하는 기능이다. 조작 기능은 검색, 삽입, 삭제, 갱신처럼 데이터를 실제로 다루는 기능이다. 제어 기능은 무결성 유지, 보안 유지, 병행 제어, 회복처럼 데이터를 정확하고 안전하게 운영하는 기능이다. DBMS는 파일 시스템보다 데이터 중복성과 프로그램-데이터 종속성을 줄이는 장점이 있다.

NoteBase

선지별 해설

1번: X이다. ㄱ은 옳다. 데이터베이스는 대용량 데이터와 동시 공유를 전제로 하므로 장애가 나면 일관된 상태로 회복하는 과정이 복잡해진다. 그러나 ㄴ은 틀리다. 틀린 표현은 "처리 기능"이다. DBMS의 필수 기능은 일반적으로 정의 기능, 조작 기능, 제어 기능이며, 처리 기능을 별도 필수 기능으로 두지 않는다.
2번: O이다. ㄱ은 데이터베이스의 회복 복잡성을 설명한 내용으로 옳고, ㄹ은 DBMS가 파일 시스템보다 데이터 중복성과 응용프로그램 종속성 문제를 더 잘 해결한다는 내용으로 옳다. 따라서 옳은 것만 묶은 선택지는 2번이다.
3번: X이다. ㄴ과 ㄷ이 모두 포함되어 있어 틀리다. ㄴ의 틀린 표현은 "처리 기능"이며, 올바르게는 정의 기능·조작 기능·제어 기능이다. ㄷ도 틀리다. 틀린 부분은 "정의 기능은 … 무결성 유지, 보안 유지, 병행 제어 기능을 포함한다"는 부분이다. 올바르게는 이런 기능들은 제어 기능에 속하고, 정의 기능은 스키마와 제약조건을 정의하는 기능이다.
4번: X이다. ㄹ은 옳지만 ㄷ이 틀리므로 이 조합은 정답이 될 수 없다. ㄷ의 틀린 표현은 정의 기능에 제어 기능의 내용을 넣은 점이다. 올바르게는 무결성 유지, 보안 유지, 병행 제어는 제어 기능에 해당한다.

오답 포인트

ㄷ에서 정의 기능과 제어 기능을 섞어 놓은 표현이 대표적인 함정이다. 또 ㄴ의 "처리 기능"처럼 익숙해 보이는 말을 추가해 세 가지 기본 기능 체계를 흔드는 방식도 자주 출제된다.

다시 나오면 이렇게 판단

DBMS 기능 문제는 먼저 정의 기능인지, 조작 기능인지, 제어 기능인지부터 분류하면 된다. 구조와 제약조건을 정하면 정의 기능, 데이터를 검색·삽입·삭제·갱신하면 조작 기능, 무결성·보안·병행·회복을 다루면 제어 기능으로 판단하면 된다.

암기 포인트

정의는 구조, 조작은 데이터 처리, 제어는 무결성·보안·병행·회복이다. DBMS는 중복성과 종속성을 줄인다.

NoteBase

문 9. 음식점 관리 시스템에서 고객(customer), 주문(order), 음식(food)을 관리하기 위해 릴레이션을 다음과 같이 설계하였다. 이 릴레이션에 대하여, SQL문에 대응하는 관계대수식으로 옳지 않은 것은? (단, 밑줄 친 속성은 기본 키이고, 관계대수 연산자 중에서 ρ는 재명명(rename) 연산자이며, ℑ은 집계함수(aggregate function) 연산을 정의하는 연산자이다)

customer(name, gender, phone, address, year)
order(cname, fname, price, date)
food(name, price, stock)

① SQL:
```
SELECT name
FROM food AS f
WHERE f.stock < 10 OR f.price > 20000;
```
관계대수:

$$\pi_{name}(\sigma_{f.stock<10 \text{OR} f.price>20000}(\rho_f(food)))$$

② SQL:
```
SELECT c.name, c.phone
FROM customer AS c, order AS o, food AS f
WHERE o.fname = f.name AND c.name = o.cname
AND f.price > 10000;
```
관계대수:

$$\pi_{c.name,\,c.phone}(\sigma_{o.fname=f.name \text{AND} c.name=o.cname}(\rho_c(customer) \times \rho_o(order) \times \rho_f(food)))$$

③ SQL:
```
SELECT name, SUM(price) AS orderPrice
FROM customer, order
WHERE name = cname
GROUP BY name;
```
관계대수:

$$\rho_{(name,\,orderPrice)}({}_{name}\Im_{SUM(price)}(customer \bowtie_{name=cname} order))$$

④ SQL:
```
SELECT name, AVG(food.price) AS avgPrice
FROM food, order
GROUP BY name;
```
관계대수:

$$\rho_{(name,\,avgPrice)}({}_{name}\Im_{AVG(food.price)}(food \times order))$$

정답 2번

출제 포인트

이 문제는 SQL의 FROM, WHERE, GROUP BY를 관계대수의 카티션곱, 선택, 조인, 집계 연산으로 정확히 옮길 수 있는지를 묻는 문제이다.

NoteBase

핵심 개념

선택 조건은 관계대수의 선택 연산 σ로 빠짐없이 옮겨야 하고, 조인은 카티션곱 뒤 선택으로 표현하거나 조인 연산으로 직접 표현할 수 있다.
집계가 있으면 GROUP BY 속성과 집계 대상 속성이 모두 집계 연산 $\mathcal{G}$ 안에 반영되어야 한다.

선지별 해설

① O
food를 f로 재명명한 뒤 f.stock < 10 f.price > 20000 조건을 선택하고 name만 투영했으므로 SQL과 대응된다.
② X
SQL에는 `f.price > 10000` 조건이 있는데 제시된 관계대수식의 선택 조건에는 `o.fname = f.name AND c.name = o.cname`만 있다.
틀린 부분은 비싼 음식만 남기는 조건이 빠진 점이다.
올바르게는 _c.name,c.phone(_o.fname=f.name c.name=o.cname f.price>10000(_c(customer) × _o(order) × _f(food)))처럼 `f.price > 10000`이 포함되어야 한다.
③ O
customer와 order를 `name = cname`으로 조인한 뒤 name으로 그룹화하여 price의 합계를 구하고, 결과 속성명을 `(name, orderPrice)`로 재명명했으므로 SQL과 대응된다.
④ O
WHERE 절이 없으므로 SQL의 `FROM food, order`는 카티션곱 `food × order`에 해당한다.
이 상태에서 `name`은 food.name을 가리키고, 그 값으로 그룹화하여 `AVG(food.price)`를 계산하므로 제시된 관계대수식과 대응된다.

오답 포인트

②는 조인 조건 두 개만 보고 맞다고 넘기기 쉽지만, WHERE 절의 나머지 조건인 `f.price > 10000`까지 모두 선택 연산에 들어가야 한다.
④는 조인 조건이 없으면 이상해 보인다는 이유로 틀렸다고 보기 쉬우나, SQL 자체가 카티션곱을 수행하므로 관계대수식도 `food × order`가 맞다.

다시 나오면 이렇게 판단

먼저 FROM 절을 관계들의 곱과 재명명으로 옮기고, WHERE 절의 조건을 하나도 빠뜨리지 않고 선택 연산에 대응시킨다.
그다음 GROUP BY와 집계 함수가 있으면 $\mathcal{G}$로 묶고, 마지막에 SELECT 목록을 투영 π와 재명명 ρ로 정리하면 된다.

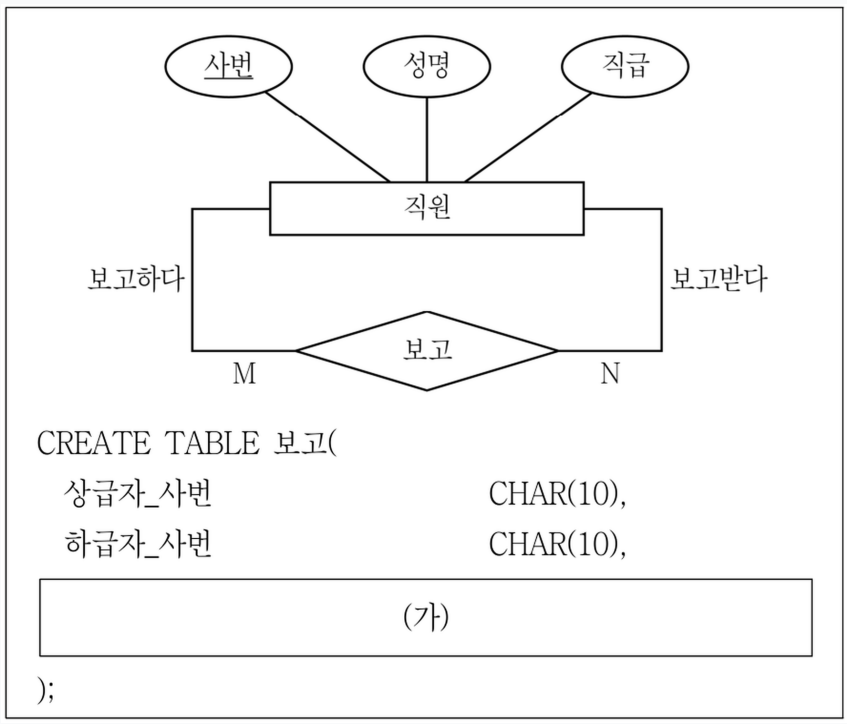
NoteBase

암기 포인트

FROM은 곱, WHERE는 σ, SELECT는 π, 이름 변경은 ρ, GROUP BY+집계는 $\mathcal{G}$로 대응된다고 기억한다.

NoteBase

문 10. 다음 ERD(entity-relationship diagram)에 대응하는 보고 테이블을 생성하고자 한다. 보고 테이블을 생성하는 SQL문의 (가)에 들어갈 내용으로 옳은 것은? (단, 밑줄 친 속성은 기본 키이고, 직원 개체(entity)는 테이블로 생성되어 있다고 가정한다)



① PRIMARY KEY (상급자_사번, 하급자_사번),
FOREIGN KEY (상급자_사번) REFERENCES 직원(사번),
FOREIGN KEY (하급자_사번) REFERENCES 직원(사번)

② PRIMARY KEY (상급자_사번),
FOREIGN KEY (상급자_사번) REFERENCES 직원(사번),
FOREIGN KEY (하급자_사번) REFERENCES 직원(사번)

③ PRIMARY KEY (상급자_사번, 하급자_사번),
FOREIGN KEY (상급자_사번, 하급자_사번) REFERENCES 직원(사번)

④ PRIMARY KEY (하급자_사번),
FOREIGN KEY (하급자_사번) REFERENCES 직원(사번)

정답 1번

출제 포인트

이 문제는 자기참조 M:N 관계인 ERD를 관계 스키마로 변환할 때 교차 테이블의 기본 키와 외래 키를 어떻게 잡는지를 묻는 문제이다.

NoteBase

핵심 개념

직원 개체가 직원 개체와 다시 관계를 맺는 재귀 관계라도 M:N이면 관계 자체를 별도 테이블로 만든다. 이때 관계 테이블에는 참여하는 두 역할의 기본 키를 각각 외래 키로 두고, 두 값을 묶어 기본 키로 잡는다.

선지별 해설

① O
- 보고 관계는 직원과 직원 사이의 자기참조 M:N 관계이므로 보고 테이블을 별도로 만들고 상급자_사번, 하급자_사번을 모두 둔다.
- 각 열은 직원(사번)을 참조하는 외래 키가 되고, 한 보고 관계를 유일하게 식별하려면 두 값을 함께 묶어야 하므로 PRIMARY KEY (상급자_사번, 하급자_사번)이 맞다.

② X
- M:N 관계인데 PRIMARY KEY를 상급자_사번 하나만 두면 한 상급자가 여러 하급자와 맺는 관계를 구분할 수 없다.
- 틀린 표현은 "PRIMARY KEY (상급자_사번)"이다. 올바르게는 "PRIMARY KEY (상급자_사번, 하급자_사번)"이어야 한다.

③ X
- 상급자_사번과 하급자_사번을 묶어 기본 키로 두는 점은 맞지만, 두 열을 한 번에 묶은 FOREIGN KEY (상급자_사번, 하급자_사번) REFERENCES 직원(사번)은 직원 테이블의 단일 기본 키 사번과 대응되지 않는다.
- 틀린 표현은 복합 외래 키 선언이다. 올바르게는 "FOREIGN KEY (상급자_사번) REFERENCES 직원(사번), FOREIGN KEY (하급자_사번) REFERENCES 직원(사번)"처럼 각각 따로 참조해야 한다.

④ X
- PRIMARY KEY를 하급자_사번 하나만 두면 한 하급자가 여러 상급자와 연결되는 관계를 표현할 수 없어서 M:N 조건을 만족하지 못한다.
- 틀린 표현은 "PRIMARY KEY (하급자_사번)"이다. 올바르게는 "PRIMARY KEY (상급자_사번, 하급자_사번)"이어야 하며 상급자_사번도 직원(사번)을 참조하는 외래 키로 있어야 한다.

오답 포인트

자기참조 관계라는 이유만으로 1:N 조직도처럼 단순화하면 함정에 걸린다. 이 문제의 핵심은 관계선 양쪽에 M과 N이 모두 표시되어 있어 상급자 하나만이나 하급자 하나만으로는 관계를 식별할 수 없다는 점이다. 또한 같은 직원 테이블을 참조하더라도 역할이 둘이면 외래 키도 둘로 나누어 선언해야 한다.

NoteBase

다시 나오면 이렇게 판단

ERD를 테이블로 옮길 때 먼저 관계의 카디널리티를 본다. M:N이면 관계를 별도 테이블로 만들고, 참여 개체의 기본 키들을 각 역할별 외래 키로 넣은 뒤 그 조합을 기본 키로 판단하면 된다.

암기 포인트

재귀 M:N 관계는 역할별 FK 2개, PK는 두 FK의 조합이다.

NoteBase

문 11. 다음은 사원 정보를 나타내기 위한 EMP 테이블이다. EMP 테이블에서 상사가 없는 사원의 SUPERID가 NULL 처리될 때, 아래 SQL문 실행 결과의 차수(degree)와 카디널리티(cardinality)는? (단, 밑줄 친 속성은 기본 키이다)

EMP

ID	NAME	SALARY	SUPERID
1001	Kim	8000	NULL
1002	Lee	5000	1003
1003	Park	7000	NULL
1004	Cho	3000	1001

```
SELECT *
FROM EMP S RIGHT OUTER JOIN EMP E
ON S.ID = E.SUPERID;
```

① 차수: 4, 카디널리티: 2

② 차수: 4, 카디널리티: 8

③ 차수: 8, 카디널리티: 2

④ 차수: 8, 카디널리티: 4

정답 4번

출제 포인트

이 문제는 자기 조인에서 RIGHT OUTER JOIN이 어느 테이블의 튜플을 보존하는지와 SELECT * 수행 시 결과 릴레이션의 차수와 카디널리티를 함께 계산하는 문제이다.

조건 정리

EMP S와 EMP E는 같은 테이블을 별칭만 달리한 자기 조인이다. RIGHT OUTER JOIN이므로 오른쪽 테이블 E의 모든 튜플이 결과에 남는다. S.ID는 기본 키이므로 E.SUPERID 하나가 매칭되면 S의 행은 최대 1개만 붙는다.

NoteBase

사용할 공식

차수는 결과에 포함되는 속성 수이고, 카디널리티는 결과 튜플 수이다. 이 문제에서는 SELECT * 이므로 S의 4개 속성과 E의 4개 속성이 모두 포함되어 차수는 8이다. 카디널리티는 E의 각 행을 기준으로 매칭 수를 세면 된다.

계산 과정

E의 각 행을 기준으로 보면 Kim은 SUPERID가 NULL이라 매칭되는 S가 없어도 외부 조인으로 1행이 남는다. Lee는 SUPERID가 1003이므로 S의 Park와 매칭되어 1행이 나온다. Park는 SUPERID가 NULL이라 1행이 남는다. Cho는 SUPERID가 1001이므로 S의 Kim과 매칭되어 1행이 나온다. 따라서 카디널리티는 4이고, 차수는 S 4개 속성 + E 4개 속성으로 8이다.

선지별 해설

① X: 차수 4가 틀리다. SELECT * 는 자기 조인이라도 S와 E의 모든 속성을 그대로 가져오므로 차수는 8이다. 카디널리티 2도 틀리다. SUPERID가 NULL인 Kim, Park도 RIGHT OUTER JOIN 때문에 결과에 남으므로 총 4행이다. 올바르게는 차수 8, 카디널리티 4이다. ② X: 차수 4가 틀리다. 조인 결과에서 같은 이름의 속성이 자동으로 합쳐지지 않으므로 8개 속성이 유지된다. 카디널리티 8도 틀리다. 각 E 행은 기본 키 S.ID와 최대 1번만 매칭되므로 4개의 E 행에서 결과도 4행이다. 올바르게는 차수 8, 카디널리티 4이다. ③ X: 차수 8은 맞지만 카디널리티 2가 틀리다. NULL인 SUPERID 두 건을 탈락시킨 내부 조인처럼 계산한 오류이다. 외부 조인에서는 이 두 행도 유지되므로 카디널리티는 4이다. 올바르게는 차수 8, 카디널리티 4이다. ④ O: SELECT * 로 S와 E의 속성 8개가 모두 출력되므로 차수는 8이다. RIGHT OUTER JOIN은 E의 4개 행을 모두 보존하므로 카디널리티는 4이다.

오답 포인트

RIGHT OUTER JOIN을 내부 조인처럼 생각해 SUPERID가 NULL인 행을 제외하면 카디널리티를 2로 잘못 계산한다. 또 자기 조인이라고 해서 같은 이름의 속성이 하나로 합쳐진다고 오해하면 차수를 4로 잘못 본다.

다시 나오면 이렇게 판단

외부 조인 문제는 먼저 어느 쪽 테이블이 보존되는지 확인한 뒤 그 테이블의 각 행이 결과에서 몇 번 나타나는지 센다. 차수는 SELECT 절이 *인지, 특정 열만 고른 것인지부터 확인해 속성 개수를 따로 계산한다.

암기 포인트

RIGHT OUTER JOIN은 오른쪽 테이블 행을 끝까지 살리고, SELECT * 자기 조인의 차수는 양쪽 속성 수를 그대로 더한 값이다.

NoteBase

문 12. 다음 고객 테이블에 대한 SQL 구문에서 실행 결과가 다른 하나는? (단, 밑줄 친 속성은 기본 키이다)

고객

고객ID	고객이름	등급	적립금
k123	김상중	Gold	3000
l018	이영우	Vip	4000
c389	최영	Gold	3000
p273	박구연	Silver	2000
h328	홍연수	Vip	5000
j451	정영상	Silver	3000
y524	윤성진	Gold	2000

①
```
SELECT 고객이름
FROM 고객
WHERE 적립금 = (SELECT MAX(적립금) FROM 고객);
```

②
```
SELECT 고객이름
FROM 고객
WHERE 적립금 >= ALL (SELECT 적립금 FROM 고객);
```

③
```
SELECT 고객이름
FROM 고객
WHERE 적립금 > ANY (SELECT 적립금 FROM 고객
                    WHERE 등급 = 'Vip');
```

④
```
SELECT 고객이름
FROM 고객
WHERE 적립금 IN (SELECT 적립금 FROM 고객
                 WHERE 등급 = 'Vip');
```

정답 4번

출제 포인트

이 문제는 고객 테이블의 적립금 데이터를 기준으로 MAX, >= ALL, > ANY, IN 연산의 결과 집합이 어떻게 달라지는지를 묻는 문제이다.

자료 읽는 순서

1. Vip 등급 고객의 적립금이 4000, 5000이라는 점을 확인한다.
2. 전체 고객의 최대 적립금이 5000이라는 점을 확인한다.
3. 각 SQL이 홍연수 1명만 조회하는지, 4000과 5000을 모두 조회하는지 비교한다.

NoteBase

선지별 해설

① X
MAX(적립금)은 5000이므로 적립금이 5000인 홍연수만 조회된다. 따라서 ①은 실행 결과가 다른 쿼리가 아니라 ②, ③과 같은 결과이다. 틀린 표현은 '실행 결과가 다른 하나'로 고른 점이며, 올바르게는 '홍연수만 조회되므로 같은 결과'이다.

② X
적립금 >= ALL(전체 적립금 집합)은 모든 값 이상이라는 뜻이므로 결국 최대값 5000과 같은 의미가 된다. 따라서 홍연수만 조회된다. 틀린 표현은 '실행 결과가 다른 하나'로 고른 점이며, 올바르게는 '최대 적립금 행만 조회되므로 같은 결과'이다.

③ X
Vip 고객의 적립금은 4000, 5000이다. 적립금 > ANY(4000, 5000)은 두 값 중 하나보다만 크면 되므로 4000보다 큰 값이면 만족하고, 실제로는 5000만 해당한다. 따라서 홍연수만 조회된다. 틀린 표현은 '실행 결과가 다른 하나'로 고른 점이며, 올바르게는 '최솟값 4000보다 큰 값만 남아 홍연수만 조회된다'이다.

④ O
Vip 고객의 적립금을 서브쿼리로 구하면 4000, 5000이 나온다. 적립금 IN (4000, 5000)은 그 값과 같은 행을 모두 찾으므로 이영우와 홍연수 2명이 조회된다. ①, ②, ③은 홍연수만 조회하지만 ④는 두 명을 조회하므로 실행 결과가 다르다.

오답 포인트

ANY를 ALL과 비슷하게 읽거나, IN을 최대값 비교로 착각하면 오답이 된다. 특히 > ANY는 '집합의 최솟값보다 크면 참'으로 해석되고, IN은 서브쿼리 결과값 각각과의 동등 비교라는 점이 함정이다.

다시 나오면 이렇게 판단

서브쿼리 결과가 여러 값이면 먼저 그 집합을 숫자로 적어 두고 해석한다. 그다음 = (MAX), >= ALL, > ANY, IN을 각각 최대값 비교인지, 최솟값 기준 비교인지, 집합 포함 여부인지로 바꾸어 보면 결과 집합을 빠르게 판단할 수 있다.

암기 포인트

>= ALL은 최대값 조건, > ANY는 최솟값 초과 조건, IN은 집합 포함 조건이다.

NoteBase

문 13. 데이터베이스 시스템의 데이터 언어에 대한 설명으로 옳지 않은 것은?

① 데이터 언어는 사용자가 데이터베이스를 구축하고, 이에 접근하기 위해 데이터베이스 관리시스템과 통신하는 수단이다.

② 비절차적 데이터 조작어는 사용자가 어떤 데이터를 원하고 해당 데이터를 얻으려면 어떻게 처리해야 하는지에 대한 절차를 구체적으로 명세한다.

③ 데이터 제어어는 데이터베이스에 저장된 데이터를 여러 사용자가 무결성과 일관성을 유지하며 문제없이 공유할 수 있도록, 내부적으로 필요한 규칙이나 기법을 정의하는 데 사용하는 데이터 언어이다.

④ 데이터 정의어는 새로운 데이터베이스를 구축하기 위해 스키마를 정의하거나 기존 스키마의 정의를 삭제하거나 수정하기 위해 사용하는 데이터 언어이다.

정답 2번

출제 포인트

이 문제는 데이터 언어의 종류별 역할, 특히 비절차적 데이터 조작어와 데이터 제어어의 차이를 묻는 문제이다.

핵심 개념

데이터 정의어(DDL)는 데이터베이스의 구조와 스키마를 정의·변경·삭제하는 언어이다.
데이터 조작어(DML)는 저장된 데이터를 검색·삽입·삭제·갱신하는 언어이다.
비절차적 DML은 원하는 결과가 무엇인지만 기술하고, 그 결과를 얻는 접근 절차는 DBMS가 결정한다.
데이터 제어어(DCL)는 권한 부여, 보안, 무결성 유지, 회복, 병행 제어처럼 데이터 사용을 통제하는 언어이다.

선지별 해설

① O: 데이터 언어는 사용자가 데이터베이스를 정의하거나 조작하고, 그 과정에서 DBMS와 상호작용하는 수단이라는 설명이므로 맞다.
② X: 틀린 부분은 "해당 데이터를 얻으려면 어떻게 처리해야 하는지에 대한 절차를 구체적으로 명세한다"는 표현이다. 비절차적 데이터 조작어는 처리 절차를 상세히 쓰지 않고 원하는 데이터만 선언한다. 올바르게는 "어떤 데이터를 원하는지만 기술하고, 접근 절차는 DBMS가 결정한다"이다.
③ O: 데이터 제어어는 여러 사용자가 데이터를 공유할 때 권한, 무결성, 일관성, 회복, 병행 제어와 관련된 통제 기능을 다루므로 맞다.
④ O: 데이터 정의어는 새로운 스키마를 정의하고 기존 스키마를 수정하거나 삭제하는 데 사용하는 언어이므로 맞다.

오답 포인트

절차적 데이터 조작어와 비절차적 데이터 조작어를 뒤집어 놓는 함정이 핵심이다. 선지 ②처럼 "어떻게 처리할지"를 구체적으로 쓴다는 표현은 절차적 언어의 특징이지 비절차적 언어의 특징이 아니다.

NoteBase

다시 나오면 이렇게 판단

선지에 처리 절차를 사용자가 직접 명세한다고 나오면 절차적 언어로 본다. 원하는 결과만 기술한다고 나오면 비절차적 언어로 판단하고, 스키마는 DDL, 데이터의 검색·변경은 DML, 권한·보안·무결성 통제는 DCL로 구분한다.

암기 포인트

DDL은 구조, DML은 데이터, DCL은 통제이다. 비절차적 DML의 핵심은 "무엇"이지 "어떻게"가 아니다.

NoteBase

문 14. 다음은 분산 데이터베이스에 대한 설명이다. (가), (나)에 들어갈 내용을 바르게 연결한 것은?

(가)는 각 조각 스키마의 인스턴스가 물리적으로 저장되어야 하는 지역을 정의한다.
(나)는 분산 데이터베이스에 저장할 모든 데이터 구조와 제약조건을 정의하는 것으로서, 관계 데이터 모델의 관점에서 보면 데이터베이스 안에 존재하는 모든 릴레이션 스키마의 집합이다.

① (가): 지역 스키마, (나): 전역 개념 스키마
② (가): 단편화 스키마, (나): 지역 스키마
③ (가): 단편화 스키마, (나): 할당 스키마
④ (가): 할당 스키마, (나): 전역 개념 스키마

정답 4번

출제 포인트

이 문제는 분산 데이터베이스에서 할당 스키마와 전역 개념 스키마의 정의를 구별하는 문제이다.

핵심 개념

할당 스키마는 각 조각 스키마의 인스턴스를 어느 지역 사이트에 물리적으로 저장할지를 정하는 스키마이다.
키워드는 저장 위치와 사이트 배치이다.
전역 개념 스키마는 분산 데이터베이스 전체의 데이터 구조와 제약조건을 통합적으로 정의한 스키마이다. 관계 데이터 모델 관점에서는 데이터베이스에 존재하는 모든 릴레이션 스키마의 집합이다.
단편화 스키마는 전체 릴레이션을 어떤 조각으로 나눌지를 정의하는 스키마이다.
지역 스키마는 특정 지역이나 사이트에서 보이는 데이터의 논리적 구조를 정의하는 스키마이다.

NoteBase

선지별 해설

① X: (가)는 틀렸다. 각 조각 스키마의 인스턴스를 물리적으로 어느 지역에 저장할지를 정하는 것은 지역 스키마가 아니라 할당 스키마이다. (나)는 맞다. 모든 데이터 구조와 제약조건, 모든 릴레이션 스키마의 집합은 전역 개념 스키마를 뜻한다.
② X: (가)가 틀렸다. 단편화 스키마는 릴레이션을 수평 분할이나 수직 분할 등으로 어떻게 나눌지를 정의할 뿐, 저장 지역을 정하지 않는다. 올바르게는 할당 스키마이다. (나)도 틀렸다. 지역 스키마는 특정 사이트의 논리 구조이지 데이터베이스 전체의 모든 릴레이션 스키마 집합이 아니다. 올바르게는 전역 개념 스키마이다.
③ X: (가)가 틀렸다. 단편화 스키마는 조각의 분할 방식을 정의하는 스키마이므로 저장 지역을 설명하는 문장과 맞지 않는다. 올바르게는 할당 스키마이다. (나)도 틀렸다. 할당 스키마는 조각을 어느 사이트에 둘지를 정하는 물리적 배치 개념이지 전체 데이터 구조와 제약조건을 정의하는 개념이 아니다. 올바르게는 전역 개념 스키마이다.
④ O: (가)는 할당 스키마로 보는 것이 맞다. 각 조각 스키마의 인스턴스가 물리적으로 저장될 지역을 정의하기 때문이다. (나)는 전역 개념 스키마로 보는 것이 맞다. 분산 데이터베이스 전체의 논리적 구조와 제약조건, 모든 릴레이션 스키마를 통합적으로 정의하기 때문이다.

오답 포인트

출제자는 단편화와 할당을 바꾸어 놓고, 지역 스키마와 전역 개념 스키마도 서로 섞어 놓았다. '어떻게 나누는가'와 '어디에 저장하는가'를 구분하지 못하면 ②나 ③으로 흔들리기 쉽다.

다시 나오면 이렇게 판단

문장에 조각을 나눈다는 뜻이 있으면 단편화 스키마를, 조각 인스턴스를 어느 사이트에 저장하는지가 나오면 할당 스키마를 고른다. 데이터베이스 전체의 논리 구조와 모든 릴레이션 스키마를 묻으면 전역 개념 스키마를, 특정 지역의 논리 구조를 묻으면 지역 스키마를 고르면 된다.

암기 포인트

나누는 것은 단편화, 놓는 것은 할당, 전체 논리 구조는 전역 개념, 특정 사이트 관점은 지역 스키마로 기억한다.

NoteBase

문 15. NoSQL에 대한 설명으로 옳지 않은 것은?

① NoSQL 시스템으로 Ingres 등이 있다.

② 가용성을 높이기 위해 일치성을 약하게 보장하는 궁극적 일치성(eventual consistency)을 제공하는 경우가 많다.

③ 미리 정의된 스키마를 사용하지 않아도 된다.

④ 다수의 컴퓨터에 데이터를 분산, 저장, 처리하는 것이 가능한 데이터베이스 시스템이다.

정답 1번

출제 포인트

이 문제는 NoSQL의 대표적 특징과 전통적 관계형 DBMS의 차이를 묻는 문제이다.

핵심 개념

NoSQL은 대규모 분산 환경에서 확장성과 가용성을 높이기 위해 사용되는 비관계형 데이터 저장 방식의 총칭이다.
관계형 DBMS는 테이블, 고정 스키마, SQL 기반 질의를 중심으로 데이터를 관리하는 방식이다.
NoSQL은 유연한 스키마, 분산 저장, 수평 확장, 궁극적 일치성 같은 키워드로 정리할 수 있다.

선지별 해설

① X: 틀린 부분은 "Ingres"이다. Ingres는 전통적인 관계형 DBMS이며 NoSQL 시스템의 예가 아니다. 올바르게는 MongoDB, Cassandra, HBase, Redis 등이 NoSQL 시스템의 예이다.
② O: NoSQL은 분산 환경에서 가용성과 확장성을 높이기 위해 강한 일관성 대신 궁극적 일치성을 채택하는 경우가 많다.
③ O: NoSQL은 미리 엄격하게 정의된 고정 스키마 없이도 데이터를 저장할 수 있는 경우가 많아 구조가 자주 바뀌는 데이터에 유리하다.
④ O: NoSQL은 여러 컴퓨터에 데이터를 분산 저장하고 처리하는 구조를 전제로 하는 경우가 많아 대용량 데이터와 수평 확장에 적합하다.

오답 포인트

"SQL을 쓰지 않는다"라는 이름 때문에 제품명만 보고 NoSQL로 오해하기 쉽다. 이 문항의 핵심은 예시 제품의 분류와 NoSQL의 공통 특징을 함께 구분하는 데 있다.

다시 나오면 이렇게 판단

제품명이 보이면 먼저 관계형 DBMS인지 비관계형 NoSQL인지 분류한다. 그다음 분산 처리, 유연한 스키마, 수평 확장, 궁극적 일치성 같은 키워드와 맞는지 확인하면 답을 판단할 수 있다.

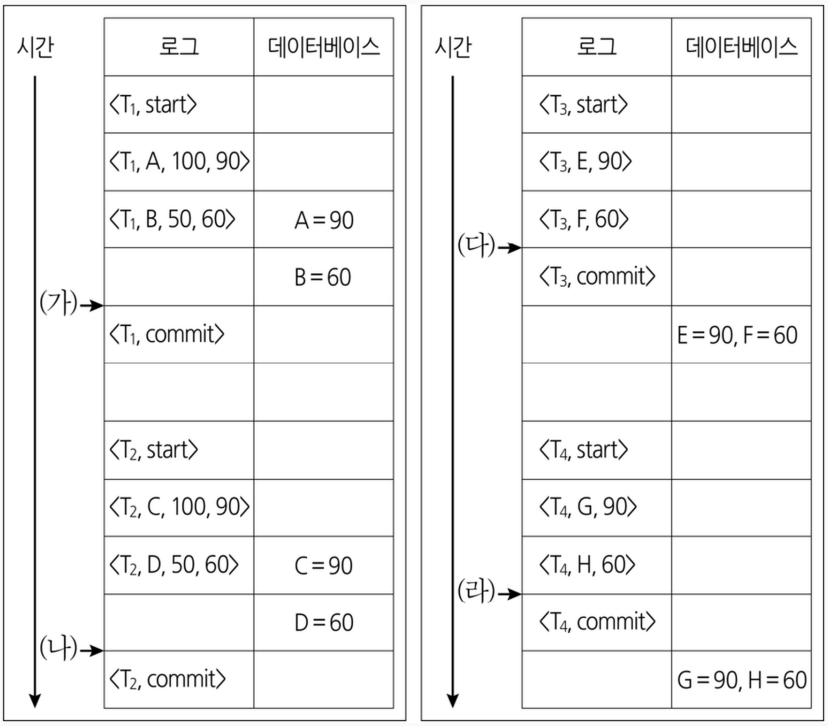
NoteBase

암기 포인트

Ingres는 관계형 DBMS이고, NoSQL은 분산 저장·유연한 스키마·수평 확장이 핵심이다.

NoteBase

문 16. 다음 두 종류의 로그 및 데이터베이스 기록 내용에서 (가), (나), (다), (라) 시점에 장애가 발생하였을 때, 수행되어야 할 회복 연산이 바르게 연결된 것은? (단, A, C, E, G의 초깃값은 100이며, B, D, F, H의 초깃값은 50이고, (가) ~ (라)의 각 시점에 발생하는 장애는 상호 영향을 주지 않는다)



① (가): 회복작업 없음, (나): Redo(T1), (다): Undo(T3), (라): Redo(T3), Undo(T4)

② (가): 회복작업 없음, (나): Redo(T1), (다): 회복작업 없음, (라): Redo(T3)

③ (가): Undo(T1), (나): Redo(T1), Undo(T2), (다): 회복작업 없음, (라): Redo(T3)

④ (가): Undo(T1), (나): Redo(T1), Undo(T2), (다): Undo(T3), (라): Redo(T3), Undo(T4)

정답 3번

출제 포인트

이 문제는 로그 형식과 데이터베이스 반영 시점을 보고 즉시 갱신형 회복과 지연 갱신형 회복을 구분할 수 있는지를 묻는다.

자료 읽는 순서

1. 로그가 이전값과 이후값을 함께 기록하는지, 이후값만 기록하는지를 본다.
2. 데이터베이스 값이 커밋 전에 반영되는지, 커밋 후에 반영되는지를 본다.
3. 장애 시점마다 커밋된 트랜잭션과 미커밋 트랜잭션을 나누어 Redo와 Undo 필요 여부를 결정한다.

NoteBase

핵심 근거

(가), (나)의 왼쪽 표는 로그에 이전값과 이후값이 모두 있고 데이터베이스 값이 커밋 전에 바뀌므로 즉시 갱신형이다. 따라서 커밋된 트랜잭션은 Redo, 미커밋 트랜잭션은 Undo 대상이다.
(다), (라)의 오른쪽 표는 로그에 이후값만 있고 데이터베이스 값이 커밋 뒤에 반영되므로 지연 갱신형이다. 따라서 커밋된 트랜잭션만 Redo 대상이고, 커밋 전 장애가 나면 아직 데이터베이스에 반영된 값이 없으므로 Undo가 필요하지 않다.

선지별 해설

1번 X: (가)에서 T1은 커밋 전인데 A=90, B=60이 이미 데이터베이스에 반영되어 있으므로 Undo(T1)이 필요하다. "회복작업 없음"이 틀린 표현이고 올바르게는 Undo(T1)이다. (나)는 즉시 갱신형이므로 커밋된 T1은 Redo, 미커밋 T2는 Undo 대상이어서 "Redo(T1)"만으로는 부족하다. 올바르게는 Redo(T1), Undo(T2)이다. (다)의 "Undo(T3)"도 틀리다. T3은 커밋 전이지만 데이터베이스 반영 전이므로 회복작업이 없다. (라)의 "Redo(T3), Undo(T4)"도 틀리다. T4는 아직 데이터베이스에 반영되지 않아 Undo가 필요 없고, 올바르게는 Redo(T3)이다.
2번 X: (가)의 "회복작업 없음"이 틀리다. 즉시 갱신형에서 미커밋 T1이 데이터베이스를 바꾸었으므로 Undo(T1)이 필요하다. (나)의 "Redo(T1)"도 틀리다. 미커밋 T2를 되돌리는 Undo(T2)가 추가되어야 한다. (다)의 회복작업 없음과 (라)의 Redo(T3)는 맞지만, 전체 연결은 옳지 않다.
3번 O: (가)는 즉시 갱신형에서 미커밋 T1이 이미 DB를 갱신한 상태이므로 Undo(T1)이다. (나)는 즉시 갱신형에서 커밋된 T1은 Redo, 미커밋 T2는 Undo 대상이므로 Redo(T1), Undo(T2)이다. (다)는 지연 갱신형에서 T3이 커밋 전이고 DB 반영도 없으므로 회복작업이 없다. (라)는 지연 갱신형에서 커밋된 T3만 Redo 대상이고 T4는 DB 미반영 미커밋 상태이므로 Redo(T3)이다.
4번 X: (가)와 (나)는 맞다. 그러나 (다)의 "Undo(T3)"가 틀리다. 지연 갱신형에서는 커밋 전 장애 시 DB 반영이 없으므로 회복작업이 없다. (라)의 "Redo(T3), Undo(T4)"도 틀리다. T4는 아직 DB에 쓰이지 않았으므로 Undo 대상이 아니다. 올바르게는 Redo(T3)이다.

오답 포인트

왼쪽 표와 오른쪽 표에 같은 회복 규칙을 적용하면 틀리기 쉽다. 특히 미커밋 트랜잭션이면 항상 Undo가 필요하다고 보면 (다), (라)를 틀린다. 반대로 커밋된 트랜잭션이면 이미 끝났다고 보고 Redo를 빼면 즉시 갱신형의 (나)를 틀린다.

다시 나오면 이렇게 판단

로그에 이전값과 이후값이 함께 있고 커밋 전에 DB가 바뀌면 즉시 갱신형으로 보고 커밋 완료는 Redo, 미커밋은 Undo로 판단한다. 로그에 이후값만 있고 커밋 후에만 DB가 바뀌면 지연 갱신형으로 보고 커밋 완료만 Redo, 미커밋은 회복작업 없음으로 판단한다.

NoteBase

암기 포인트

즉시 갱신형은 Undo와 Redo를 함께 보고, 지연 갱신형은 Redo만 본다.

NoteBase

문 17. 관계형 데이터베이스의 상태에 대한 제약조건으로 옳지 않은 것은?

① 참조되는 릴레이션에 새로운 투플이 삽입되는 과정에서 참조무결성 제약조건의 위반이 발생하면, 제약조건 위반의 원인이 되는 연산을 거부하도록 해야 한다.

② 릴레이션 스키마 R의 임의의 두 투플 t_1, t_2에 대하여 $t_1[SK] \neq t_2[SK]$이 성립한다면, 속성 집합 SK를 R의 슈퍼키라고 한다.

③ 참조하는 릴레이션 R1의 외래 키는 참조되는 릴레이션 R2의 기본 키와 동일한 도메인이어야 하며, 기본 키 값 중 하나이거나 NULL 값이어야 한다.

④ 데이터베이스는 개체무결성과 참조무결성 제약조건을 만족해야 한다.

정답 1번

출제 포인트

이 문제는 참조무결성, 슈퍼키, 개체무결성의 정의와 적용 시점을 구분하는 문제이다.

핵심 개념

참조무결성은 외래 키 값이 참조 릴레이션의 기본 키 값이거나 NULL이어야 한다는 제약이다.
슈퍼키는 릴레이션의 각 튜플을 유일하게 식별할 수 있는 속성 집합이다.
개체무결성은 기본 키 값이 NULL이 될 수 없다는 제약이다.

선지별 해설

① X. 참조무결성 위반은 참조하는 릴레이션이 존재하지 않는 기본 키 값을 외래 키로 가지거나, 참조되는 릴레이션의 기본 키 튜플이 삭제·변경될 때 발생한다. 참조되는 릴레이션에 새 튜플을 삽입하는 것은 참조 대상을 늘리는 일이므로 위반 원인이 아니다. 틀린 표현은 "참조되는 릴레이션에 새로운 튜플이 삽입되는 과정에서"이다. 올바르게는 "참조하는 릴레이션에 잘못된 외래 키가 삽입되거나 참조되는 릴레이션의 기본 키가 삭제·변경되는 과정에서"이다.
② O. 임의의 두 튜플 t_1, t_2에 대하여 $t_1[SK] \neq t_2[SK]$가 성립하면 속성 집합 SK는 각 튜플을 유일하게 구분하므로 슈퍼키이다.
③ O. 외래 키는 참조되는 릴레이션의 기본 키와 대응 가능한 도메인을 가져야 하며, 값은 참조되는 기본 키 값 중 하나이거나 NULL일 수 있다. 이는 참조무결성의 기본 정의에 맞는다.
④ O. 관계형 데이터베이스는 기본 키의 NULL 금지를 뜻하는 개체무결성과 외래 키의 참조 일관성을 뜻하는 참조무결성을 만족해야 한다.

NoteBase

오답 포인트

참조되는 릴레이션과 참조하는 릴레이션의 역할을 뒤바꾸면 ①을 맞는 문장처럼 오해하기 쉽다. 참조무결성은 부모 릴레이션에 행을 추가하는 상황보다 자식 릴레이션의 외래 키 입력, 부모 릴레이션의 삭제·변경 상황에서 깨진다는 점이 함정이다.

다시 나오면 이렇게 판단

외래 키 제약 문제는 먼저 참조하는 릴레이션이 자식, 참조되는 릴레이션이 부모라는 관계를 잡으면 된다. 그다음 "부모를 새로 추가"하는 연산인지, "자식이 없는 부모를 지우거나 자식이 잘못된 부모를 가리키는" 연산인지를 구분하면 정오를 빠르게 판단할 수 있다.

암기 포인트

슈퍼키는 유일 식별, 개체무결성은 기본 키 NULL 금지, 참조무결성은 외래 키가 부모 기본 키를 따라야 한다는 규칙이다.

NoteBase

문 18. SQL의 부속질의(subquery)에 대한 설명으로 옳지 않은 것은?

① 부속질의는 하나의 SQL문 안에 다른 SQL문이 중첩된 질의를 말한다.

② 연산자 ANY는 부속질의가 단일 행을 반환할 때만 적용할 수 있다.

③ 스칼라(scalar) 부속질의는 결과 값을 단일한 스칼라 값으로 반환한다.

④ 인라인 뷰(inline view)는 FROM 절에서 사용되는 부속질의이다.

정답 2번

출제 포인트

부속질의를 반환 행 수와 사용 위치에 따라 구분하고, 단일행·다중행 연산자의 대응 관계를 아는지를 묻는 문제이다.

핵심 개념

단일행 부속질의는 한 행만 반환하며 단일행 비교 연산자와 함께 사용된다.
다중행 부속질의는 여러 행을 반환하며 IN, ANY, ALL, EXISTS 같은 다중행 연산자와 함께 사용된다.
스칼라 부속질의는 1행 1열의 단일 값 하나를 반환한다.
인라인 뷰는 FROM 절에 넣어 임시 테이블처럼 사용하는 부속질의이다.

선지별 해설

① O
부속질의는 하나의 SQL문 안에 다른 SQL문이 포함된 중첩 질의를 뜻하므로 설명이 맞다.

② X
틀린 부분은 "단일 행을 반환할 때만"이다.
ANY는 여러 행을 반환하는 다중행 부속질의에서 반환값들 중 하나 이상과 비교 조건을 만족하는지를 판단할 때 사용하는 연산자이다.
올바르게는 "연산자 ANY는 부속질의가 여러 행을 반환할 때 적용할 수 있다"이다.

③ O
스칼라 부속질의는 결과가 값 하나인 부속질의이므로 단일한 스칼라 값으로 반환한다는 설명이 맞다.

④ O
인라인 뷰는 FROM 절에서 사용하는 부속질의이므로 설명이 맞다.

NoteBase

오답 포인트

ANY를 단일행 부속질의 연산자로 착각하면 ②를 맞는 설명처럼 보게 된다. 이 문제는 부속질의의 이름이 아니라 반환 행 수와 연결되는 연산자를 구분하는지를 노린 문항이다.

다시 나오면 이렇게 판단

먼저 부속질의가 몇 행 몇 열을 반환하는지 본 뒤 그 결과 형태에 맞는 연산자와 사용 절을 연결하면 된다. 여러 행이면 ANY·ALL·IN 같은 다중행 연산자를 떠올리고, FROM 절에 있으면 인라인 뷰로 판단한다.

암기 포인트

ANY는 다중행, 스칼라는 1행 1열, 인라인 뷰는 FROM 절이다.

NoteBase

문 19. 다음 릴레이션 R_1, R_2, R_3에 대한 관계대수식의 의미가 다른 것은?

$R_1(A, B, C)\ R_2(D, E)\ R_3(F, G, H)$

① $\sigma_{B=5}(R_1 \bowtie_{A=D} R_2) \bowtie_{E=F} R_3$

② $\sigma_{B=5\ \text{AND}\ A=D}(\pi_{A,B}(R_1) \times R_2) \bowtie_{E=F} R_3$

③ $\sigma_{A=D}(\sigma_{B=5}(R_1) \times R_2) \bowtie_{E=F} R_3$

④ $\sigma_{B=5}((R_2 \bowtie_{E=F} R_3) \bowtie_{A=D} R_1)$

정답 2번

출제 포인트

이 문제는 선택 연산의 푸시다운, 세타 조인의 결합성, 그리고 투영 연산이 결과 스키마를 바꾼다는 점을 비교하는 문제이다.

핵심 개념

선택 σ은 조건에 쓰인 속성이 남아 있으면 조인 전후로 이동할 수 있다.
세타 조인은 같은 조건을 유지하면 결합 순서를 바꾸어도 같은 튜플 집합을 만든다.
투영 π은 지정한 속성만 남기므로, 원래 릴레이션의 다른 속성은 결과에서 사라진다.

NoteBase

선지별 해설

① O

$R_1 \bowtie_{A=D} R_2$를 만든 뒤 $B = 5$를 선택하고 다시 $E = F$로 R_3와 조인한 형태이다.

여기서 $B = 5$는 R_1의 속성만 사용하므로 실제 의미는 R_1, R_2, R_3를 $A = D, E = F$로 연결하면서 $B = 5$인 튜플만 남기는 것이다.

② X

$\pi_{A,B}(R_1)$가 먼저 수행되어 R_1의 속성 C가 결과에서 제거된다.

그 뒤 $\sigma_{B=5 \text{ AND } A=D}(\pi_{A,B}(R_1) \times R_2)$와 R_3의 조인은 조건 자체는 ①과 비슷하지만, 최종 결과 스키마에 C가 없다.

틀린 부분은 ①, ③, ④와 같은 의미라고 보는 점이다.

올바르게는 ②는 C를 버린 뒤 조인하므로 결과 속성 집합이 달라져 다른 의미가 된다.

③ O

$\sigma_{B=5}(R_1) \times R_2$에서 먼저 R_1에 $B = 5$를 적용한 뒤, $\sigma_{A=D}$로 필요한 튜플만 남기면 $R_1 \bowtie_{A=D} R_2$ 후에 $B = 5$를 거는 것과 같다.

이후 $E = F$로 R_3와 조인하므로 ①과 같은 의미이다.

④ O

먼저 $R_2 \bowtie_{E=F} R_3$를 수행하고, 그 결과를 $A = D$ 조건으로 R_1과 조인한 뒤 $B = 5$를 선택한 형태이다.

조인의 결합 순서만 바뀌었을 뿐 조건은 같고, $B = 5$도 여전히 R_1의 속성에 대한 선택이므로 ①과 같은 의미이다.

오답 포인트

많이 틀리는 지점은 조인 조건의 모양만 보고 네 식이 모두 같다고 판단하는 점이다. 이 문항의 함정은 ②에서만 투영이 먼저 일어나 C 속성이 사라진다는 점이며, 튜플 조건이 같아 보여도 결과 스키마가 달라지면 관계대수식의 의미도 달라진다.

다시 나오면 이렇게 판단

먼저 각 식의 최종 속성 집합이 같은지 확인한다. 그다음 선택 조건이 한 릴레이션의 속성만 쓰는지 보고 조인 전으로 밀어 넣을 수 있는지 판단한다. 마지막으로 조인은 결합 순서를 바꾸어도 되는지 확인하면 동치 여부를 빠르게 판별할 수 있다.

암기 포인트

선택은 밀어 넣을 수 있고 조인은 묶는 순서를 바꿀 수 있지만, 투영은 속성을 지우므로 의미를 가장 쉽게 바꾼다.

NoteBase

문 20. 두 개의 트랜잭션 T1과 T2를 수행하는 과정에서 다음과 같은 현상이 발생하였다. 이 현상이 발생하는 격리 수준은?

T1에서 1번 상품의 가격을 조회하였을 때 10만 원으로 나타났고, T2에서 1번 상품의 가격을 15만 원으로 변경하고 커밋(commit)을 수행하였다. 그 후에 T1에서 1번 상품의 가격을 조회하니 15만 원으로 나타났다.
T2에서 1번 상품의 가격을 10만 원에서 15만 원으로 변경하다가 커밋을 수행하지 않고 롤백(rollback)하는 과정에서 T1에서 1번 상품의 가격을 조회하였으나 허용되지 않았다.
T1에서 등록된 상품을 조회하였는데 0건이 조회되었고, T2에서 1번 상품을 등록하고 커밋을 수행하였다.
T1에서 등록된 상품을 다시 조회하였더니 1개 상품이 나타났다.

① Read Committed
② Serializable
③ Repeatable Read
④ Read Uncommitted

정답 1번

출제 포인트

이 문제는 격리 수준별로 더티 리드, 비반복 읽기, 팬텀 리드가 각각 허용되는지 구분하는 문제이다.

핵심 개념

Read Uncommitted는 커밋되지 않은 변경도 읽을 수 있는 수준이므로 더티 리드가 허용된다.
Read Committed는 커밋된 데이터만 읽을 수 있으므로 더티 리드는 막지만, 같은 행을 다시 읽었을 때 값이 달라지는 비반복 읽기와 범위 재조회 결과가 달라지는 팬텀 리드는 발생할 수 있다.
Repeatable Read는 같은 행의 재조회 값 변경은 막으므로 비반복 읽기는 허용하지 않지만, 범위에 새 행이 끼어드는 팬텀 리드는 남을 수 있다.
Serializable은 트랜잭션을 직렬 실행한 것과 같은 효과를 보장하므로 더티 리드, 비반복 읽기, 팬텀 리드를 모두 막는다.

NoteBase

선지별 해설

① O
1번 사례는 T1이 같은 상품 가격을 두 번 읽었는데 T2의 커밋 이후 값이 10만 원에서 15만 원으로 바뀐 비반복 읽기이고, 2번 사례는 커밋되지 않은 값 읽기가 허용되지 않아 더티 리드가 차단되며, 3번 사례는 재조회 시 새 행이 나타나는 팬텀 리드이다. 더티 리드는 막고 비반복 읽기와 팬텀 리드는 허용하는 수준은 Read Committed이다.

② X
틀린 부분은 격리 수준을 Serializable로 본 점이다.
Serializable은 비반복 읽기와 팬텀 리드까지 모두 차단하므로 1번과 3번 같은 현상이 나타날 수 없다.
올바르게는 이 현상 조합은 Read Committed이다.

③ X
틀린 부분은 격리 수준을 Repeatable Read로 본 점이다.
Repeatable Read는 같은 행을 다시 읽을 때 값이 바뀌는 비반복 읽기를 허용하지 않는다. 이 문항에서는 1번 사례에서 같은 상품 가격이 10만 원에서 15만 원으로 바뀌었으므로 Repeatable Read가 아니다. 올바르게는 Read Committed이다.

④ X
틀린 부분은 격리 수준을 Read Uncommitted로 본 점이다.
Read Uncommitted는 커밋되지 않은 변경도 읽을 수 있으므로 2번 사례처럼 T2가 롤백할 값을 T1이 읽지 못하게 막는 상황과 맞지 않는다. 올바르게는 더티 리드를 차단하는 Read Committed이다.

오답 포인트

Repeatable Read와 Read Committed를 팬텀 리드만으로 구분하려 하면 1번 사례를 놓치기 쉽다. 이 문항의 핵심 함정은 2번 사례가 더티 리드 차단을 보여 주고, 1번 사례가 비반복 읽기 허용을 동시에 보여 준다는 점이다.

다시 나오면 이렇게 판단

먼저 커밋 전 값을 읽었는지로 더티 리드 허용 여부를 판단한다. 그다음 같은 행 값이 다시 읽을 때 달라지면 비반복 읽기 허용이고, 범위 조회 결과에 새 행이 끼어들면 팬텀 리드 허용으로 본다. 더티 리드는 막고 나머지 둘이 보이면 Read Committed로 판단한다.

암기 포인트

Read Committed는 더티 리드는 막고 비반복 읽기와 팬텀 리드는 허용하는 격리 수준이다.

NoteBase

문 21. 릴레이션 R과 S에 대한 조인 연산의 구현 방법을 설명한 것으로 옳지 않은 것은?

① 중첩 루프 조인(nested loop join): R과 S의 레코드들에 대한 별도의 인덱스나 정렬을 필요로 하지 않는다.

② 단일 루프 조인(single loop join): R과 S 중 하나의 릴레이션에 있는 레코드들은 조인 속성 값에 따라 물리적으로 정렬되어 있어야 한다.

③ 정렬-합병 조인(sort-merge join): R과 S의 레코드들이 조인 속성 값에 따라 물리적으로 정렬되어 있을 때 가장 효율적이다.

④ 해시 조인(hash join): R과 S의 레코드들은 조인 속성을 해시 키로 하고 동일한 해시 함수를 이용해 해시되어 있어야 한다.

정답 2번

출제 포인트

이 문제는 중첩 루프 조인, 단일 루프 조인, 정렬-합병 조인, 해시 조인의 전제 조건과 적용 상황을 구분할 수 있는지를 묻는다.

핵심 개념

중첩 루프 조인은 한 릴레이션의 각 튜플마다 다른 릴레이션 전체를 차례로 대조하는 가장 기본적인 조인 방식이다.
단일 루프 조인은 한쪽 릴레이션을 순차 탐색하고 다른 쪽 릴레이션은 조인 속성의 인덱스를 이용해 필요한 튜플만 찾는 방식이다.
정렬-합병 조인은 두 릴레이션이 조인 속성 순으로 정렬되어 있을 때 정렬 순서를 따라 병합하듯 매칭하는 방식이다.
해시 조인은 두 릴레이션을 같은 해시 함수로 조인 속성 기준 버킷에 분할한 뒤 같은 버킷끼리 비교하는 방식이다.

선지별 해설

1번 O: 중첩 루프 조인은 각 튜플 쌍을 직접 비교하는 기본 방식이므로 별도의 인덱스나 사전 정렬이 없어도 구현할 수 있다. 인덱스가 있으면 성능 개선에는 도움이 되지만 필수 조건은 아니다.
2번 X: 단일 루프 조인은 한쪽 릴레이션이 조인 속성 값에 따라 물리적으로 정렬되어 있어야 하는 방식이 아니다. 틀린 표현은 "물리적으로 정렬되어 있어야 한다"이다. 올바르게는 한쪽 릴레이션의 조인 속성에 인덱스가 있어 순차 탐색 중 다른 쪽의 일치 튜플을 빠르게 찾는 방식이다.
3번 O: 정렬-합병 조인은 두 릴레이션이 조인 속성 값 순으로 정렬되어 있을 때 정렬 순서를 따라 한 번 훑으며 매칭할 수 있어 매우 효율적이다. 따라서 정렬 상태에서 효율적이라는 설명은 맞다.
4번 O: 해시 조인은 조인 속성을 해시 키로 삼아 두 릴레이션에 동일한 해시 함수를 적용하고 같은 버킷에 들어간 레코드끼리 비교한다. 따라서 같은 해시 기준으로 분할해 조인한다는 설명은 타당하다.

NoteBase

오답 포인트

단일 루프 조인을 정렬-합병 조인과 혼동하면 틀린다. 정렬이 필요한 것은 정렬-합병 조인이고, 단일 루프 조인의 핵심은 한쪽 순차 탐색과 다른 쪽 인덱스 탐색의 결합이다.

다시 나오면 이렇게 판단

각 조인 방식의 전제 조건을 먼저 대응시키면 된다. 별도 구조가 없어도 되면 중첩 루프, 한쪽 인덱스를 활용하면 단일 루프, 양쪽이 정렬되어 있으면 정렬-합병, 같은 해시 버킷으로 분할하면 해시 조인이다.

암기 포인트

중첩은 전수 비교, 단일 루프는 한쪽 인덱스, 정렬-합병은 양쪽 정렬, 해시는 같은 해시 버킷 대응이다.

NoteBase

문 22. 회복 기법과 그 특징이 바르게 짝 지어진 것은?

① 지연 갱신에 기반을 둔 기법: UNDO/REDO

② 즉시 갱신에 기반을 둔 기법: NO-UNDO/NO-REDO

③ 그림자 페이지 기법: UNDO/NO-REDO

④ ARIES 기법: 분석 → REDO → UNDO

정답 4번

출제 포인트

이 문제는 회복 기법별 갱신 시점과 UNDO/REDO 필요 여부를 정확히 구별하는지를 묻는다.

핵심 개념

지연 갱신은 커밋 전에는 데이터베이스 자체를 갱신하지 않고 로그만 기록해 두는 방식이므로 보통 NO-UNDO/REDO와 연결된다.
즉시 갱신은 커밋 전에도 데이터베이스 페이지가 갱신될 수 있는 방식이므로 미완료 트랜잭션 취소와 완료 트랜잭션 재실행을 위해 UNDO/REDO가 필요하다.
그림자 페이지 기법은 원본 페이지를 보존한 채 새 페이지를 만들어 교체하므로 보통 NO-UNDO/NO-REDO의 성격을 가진다.
ARIES는 장애 복구 시 분석 → REDO → UNDO 순서로 진행하는 로그 기반 회복 기법이다.

선지별 해설

1번 X. 지연 갱신에서 틀린 부분은 "UNDO/REDO"이다. 지연 갱신은 커밋 전 데이터베이스 반영을 미루므로 UNDO가 필요하지 않고, 커밋 후 반영 전 장애에 대비한 REDO가 중심이 된다. 올바르게는 "지연 갱신에 기반을 둔 기법: NO-UNDO/REDO"이다.
2번 X. 즉시 갱신에서 틀린 부분은 "NO-UNDO/NO-REDO"이다. 즉시 갱신은 커밋 전 변경이 실제 데이터베이스에 반영될 수 있으므로 미완료 트랜잭션을 취소하는 UNDO와 완료 트랜잭션을 재실행하는 REDO가 모두 필요하다. 올바르게는 "즉시 갱신에 기반을 둔 기법: UNDO/REDO"이다.
3번 X. 그림자 페이지 기법에서 틀린 부분은 "UNDO/NO-REDO"이다. 그림자 페이지는 원본 페이지를 보존하고 새 사본을 교체하는 방식이라 기존 상태로 되돌리는 UNDO가 따로 필요하지 않다. 올바르게는 "그림자 페이지 기법: NO-UNDO/NO-REDO"이다.
4번 O. ARIES는 복구 시 먼저 분석 단계에서 로그를 훑어 트랜잭션 상태와 더티 페이지를 파악하고, 그다음 REDO로 이력을 재적용한 뒤, 마지막에 UNDO로 미완료 트랜잭션을 되돌린다. 따라서 "분석 → REDO → UNDO"가 맞다.

NoteBase

오답 포인트

지연 갱신과 즉시 갱신은 갱신 시점이 비슷하게 느껴져 UNDO 필요 여부를 뒤바꾸기 쉽고, 그림자 페이지 기법도 로그 기반 기법처럼 UNDO가 있다고 착각하면 틀리기 쉽다.

다시 나오면 이렇게 판단

먼저 커밋 전에 실제 데이터베이스 페이지가 바뀌는지 본다. 커밋 전 반영이 없으면 지연 갱신으로 보고 NO-UNDO 쪽으로, 커밋 전 반영이 있으면 즉시 갱신으로 보고 UNDO가 필요하다고 판단한다. 그림자 페이지는 원본 보존 여부를, ARIES는 분석 → REDO → UNDO 순서를 바로 연결하면 된다.

암기 포인트

지연=NO-UNDO/REDO, 즉시=UNDO/REDO, 그림자=NO-UNDO/NO-REDO,
ARIES=분석→REDO→UNDO이다.

NoteBase

문 23. 데이터베이스 보안에 대한 설명으로 옳지 않은 것은?

① 권한 관리를 이용하여 접근 권한이 없는 사용자의 접근으로부터 데이터베이스를 보호한다.

② 사용자가 소유한 데이터베이스 객체에 대한 권한 부여는 반드시 데이터 관리자에 의해 수행되어야 한다.

③ 접근이 허락된 사용자가 실수 등의 이유로 데이터 무결성을 위반하지 않도록 정의된 제약조건 등을 이용하여 통제할 수 있어야 한다.

④ 보안을 위한 데이터 단위는 데이터베이스 전체로부터 특정 테이블의 특정 행과 열 수준까지 정의될 수 있다.

정답 2번

출제 포인트

이 문제는 데이터베이스 보안에서 접근 권한 관리, 객체 소유자의 권한 부여 가능 범위, 무결성 제약조건의 역할을 구분하는 문제이다.

핵심 개념

권한 관리: 승인된 사용자에게만 데이터 접근 권한을 부여하고 비인가 접근을 차단하는 보안 기능이다. / 결정 차이축: 누가 어떤 객체에 접근할 수 있는가 / 키워드: 접근 통제
무결성 제약조건: 허용된 사용자의 입력이나 갱신이 데이터의 정확성과 일관성을 해치지 않도록 제한하는 규칙이다. / 결정 차이축: 접근 문제가 아니라 값의 적합성 문제인가 / 키워드: 무결성
객체 소유자: 자신이 소유한 데이터베이스 객체에 대해 다른 사용자에게 권한을 부여할 수 있는 주체이다. / 결정 차이축: 권한 부여 권한이 DBA에게만 전속되는가 / 키워드: 객체 소유자, GRANT

NoteBase

선지별 해설

① O
권한 관리는 접근 권한이 없는 사용자의 접근을 차단하는 기능이므로 옳다.

② X
틀린 부분은 "반드시 데이터 관리자에 의해"이다.
실제로는 데이터베이스 객체의 소유자도 자신이 소유한 객체에 대한 권한을 다른 사용자에게 부여할 수 있다.
따라서 올바른 문장은 "사용자가 소유한 데이터베이스 객체에 대한 권한 부여는 데이터 관리자 또는 해당 객체의 소유자에 의해 수행될 수 있다."이다.

③ O
접근이 허락된 사용자라도 실수로 잘못된 값을 입력하거나 갱신할 수 있으므로, 정의된 제약조건을 이용해 데이터 무결성을 통제할 수 있어야 한다. 이는 보안과 무결성이 연결되는 지점이므로 옳다.

④ O
데이터베이스 보안의 적용 단위는 데이터베이스 전체, 테이블, 뷰, 특정 행, 특정 열 등 다양한 수준으로 설정할 수 있다. 따라서 특정 테이블의 특정 행과 열 수준까지 정의될 수 있다는 설명은 옳다.

오답 포인트

②의 "반드시"가 함정이다. 권한 부여 주체를 DBA 한 사람으로만 외우면 객체 소유자의 권한 부여 기능을 놓치기 쉽다. 또한 ③은 보안과 무결성을 별개로만 외우면 틀리기 쉬운데, 허가된 사용자에 대한 제약조건 통제도 넓은 의미의 보안 관리와 연결된다.

다시 나오면 이렇게 판단

먼저 문장이 "누가 접근할 수 있는가"를 말하는지, "어떤 값이 허용되는가"를 말하는지 구분한다. 전자는 권한 관리, 후자는 무결성 제약조건으로 판단한다. 그리고 "반드시 DBA만"처럼 주체를 단정하는 표현이 나오면 객체 소유자의 권한 부여 가능 여부를 함께 점검한다.

암기 포인트

보안은 접근 통제, 무결성은 값 통제, 권한 부여는 DBA 전속이 아니라 객체 소유자도 가능하다.

NoteBase

문 24. 데이터 웨어하우스(data warehouse)에 대한 설명으로 옳지 않은 것은?

① 데이터 웨어하우스는 의사결정에 필요한 주제에 따라 데이터를 관리하는 주제 지향성(subject orientation) 특징을 가진다.

② 데이터 웨어하우스는 데이터가 일관된 형식으로 정의되는 통합성(integration) 특징을 가진다.

③ 데이터 웨어하우스는 데이터에 접근하는 현재 시간을 기준으로 최신의 값을 유지하는 시불변성(time invariance) 특징을 가진다.

④ 데이터 웨어하우스는 읽기 전용의 데이터를 유지하는 비휘발성(nonvolatilization) 특징을 가진다.

정답 3번

출제 포인트

이 문제는 데이터 웨어하우스의 핵심 특성인 주제 지향성, 통합성, 시계열성, 비휘발성을 정확히 구별할 수 있는지를 묻는 문제이다.

핵심 개념

데이터 웨어하우스는 의사결정을 지원하기 위해 여러 운영 시스템의 데이터를 분석 주제별로 통합하여 장기간 저장하는 저장소이다.
주제 지향성은 업무 처리 절차가 아니라 고객, 상품, 매출 같은 분석 주제를 중심으로 데이터를 구성하는 성질이다.
통합성은 서로 다른 원천 데이터의 이름, 코드, 형식, 단위를 일관된 기준으로 정리해 저장하는 성질이다.
시계열성은 현재값만 남기는 것이 아니라 과거부터 축적된 이력 데이터를 함께 저장하여 시간에 따른 변화를 분석할 수 있는 성질이다.
비휘발성은 적재된 데이터가 주로 조회용으로 유지되어 빈번한 수정이나 삭제가 거의 없다는 성질이다.

NoteBase

선지별 해설

① O
주제 지향성에 대한 설명으로 옳다. 데이터 웨어하우스는 주문 처리 같은 운영 기능 중심이 아니라 의사결정에 필요한 주제 중심으로 데이터를 관리한다.

② O
통합성에 대한 설명으로 옳다. 여러 시스템에서 수집한 데이터를 일관된 형식으로 정제하여 하나의 기준으로 저장한다.

③ X
틀린 부분은 "현재 시간을 기준으로 최신의 값을 유지하는 시불변성"이라는 표현이다. 데이터 웨어하우스는 최신값만 유지하는 저장소가 아니라 과거 이력까지 함께 보관하는 시계열성 또는 시간 가변성의 특징을 가진다. 올바르게는 "시간의 흐름에 따른 데이터를 누적 저장하여 과거와 현재를 비교할 수 있는 시계열성(time variant) 특징을 가진다"라고 정리된다.

④ O
비휘발성에 대한 설명으로 옳다. 데이터 웨어하우스의 데이터는 읽기 전용에 가깝게 유지되며, 운영 데이터베이스처럼 빈번한 갱신을 전제로 하지 않는다.

오답 포인트

③은 "시간 관련 특성"이라는 큰 틀만 맞고 세부 내용이 뒤바뀌어 있어 헷갈리기 쉽다. 데이터 웨어하우스의 시간 특성은 최신 현재값 유지가 아니라 과거 이력 축적이며, 최신 상태만 빠르게 반영하는 쪽은 운영 데이터베이스의 성격에 가깝다.

다시 나오면 이렇게 판단

데이터 웨어하우스의 시간 관련 선지가 나오면 먼저 현재값만 유지하는지, 과거 이력을 누적하는지를 구분한다. 과거와 현재를 함께 분석할 수 있다고 하면 데이터 웨어하우스의 시계열성이고, 최신 거래값을 즉시 갱신한다고 하면 운영 데이터베이스의 성격이다.

암기 포인트

데이터 웨어하우스는 주제 지향성, 통합성, 시계열성, 비휘발성이 핵심이며, 최신 현재값 중심은 운영 데이터베이스에 가깝다.

NoteBase

문 25. 데이터베이스에서 동시성 제어(concurrency control)를 위한 로킹(locking) 기법과 이를 이용하여 제어할 수 있는 상황이 바르게 짝 지어진 것은?

ㄱ. 직렬가능(serializable) 스케줄 보장
ㄴ. 교착상태(deadlock) 방지
ㄷ. 복구가능한(recoverable) 스케줄 보장

① 2PL(two-phase locking) - ㄷ
② 엄격한(strict) 2PL - ㄱ, ㄷ
③ 엄중한(rigorous) 2PL - ㄱ, ㄴ
④ 엄중한(rigorous) 2PL - ㄱ, ㄴ, ㄷ

정답 2번

출제 포인트

이 문제는 2PL, strict 2PL, rigorous 2PL이 각각 어떤 스케줄 성질을 보장하는지와 교착상태 방지와는 별개라는 점을 구분하는 문제이다.

핵심 개념

2PL은 잠금 획득 단계와 잠금 해제 단계가 분리되는 프로토콜로 충돌 직렬가능 스케줄을 보장한다.
strict 2PL은 배타 잠금을 트랜잭션 종료 시점까지 유지하므로 직렬가능성과 복구가능성을 보장하고 연쇄 취소를 막는다.
rigorous 2PL은 공유 잠금과 배타 잠금을 모두 종료 시점까지 유지하는 더 강한 형태로 strict 2PL보다 제약이 강하지만, 교착상태를 자동으로 방지하는 기법은 아니다.
교착상태 방지는 2PL 계열 자체의 성질이 아니라 wait-die, wound-wait, 타임아웃, 탐지/회복 같은 별도 기법의 영역이다.

NoteBase

선지별 해설

① X
2PL은 ㄱ의 직렬가능 스케줄 보장은 맞지만 ㄷ의 복구가능 스케줄 보장은 일반적으로 보장하지 않는다.
틀린 부분은 '2PL - ㄷ'이다.
올바르게는 '2PL - ㄱ'으로 보는 것이 맞다.

② O
strict 2PL은 직렬가능 스케줄을 보장하므로 ㄱ이 맞다.
또한 배타 잠금을 종료 시점까지 유지하므로 다른 트랜잭션이 미완료 갱신 결과를 읽지 못해 복구가능 스케줄이 보장되므로 ㄷ도 맞다.
교착상태 방지 성질은 포함되지 않으므로 ㄱ, ㄷ의 조합이 정확하다.

③ X
rigorous 2PL은 모든 잠금을 종료 시점까지 유지하므로 ㄱ의 직렬가능성은 맞지만 ㄴ의 교착상태 방지는 틀리다.
틀린 부분은 'ㄴ'이다.
올바르게는 'rigorous 2PL - ㄱ, ㄷ'이다.

④ X
rigorous 2PL은 ㄱ과 ㄷ은 보장하지만 ㄴ의 교착상태 방지는 보장하지 않는다.
틀린 부분은 'ㄴ'을 포함한 점이다.
올바르게는 'rigorous 2PL - ㄱ, ㄷ'이다.

오답 포인트

출제자가 노린 함정은 '잠금을 오래 잡고 있으면 교착상태도 막을 것 같다'는 직관이다. 실제로는 strict 2PL과 rigorous 2PL은 복구 안정성을 높이는 방식이지 교착상태를 예방하는 방식이 아니다. 또한 기본 2PL의 직렬가능 보장과 strict 계열의 복구가능 보장을 섞으면 ① 같은 오답에 걸리기 쉽다.

다시 나오면 이렇게 판단

2PL 계열 문제는 먼저 '직렬가능성 보장 여부'와 '복구가능성 보장 여부'를 나누어 본다. 그다음 '교착상태 방지'가 나오면 2PL 자체의 성질이 아니라 별도 예방·탐지 기법인지부터 확인하면 된다. strict나 rigorous가 붙으면 복구 측면은 강해지지만 교착상태 방지로 바로 연결되지는 않는다.

암기 포인트

2PL은 직렬가능, strict 2PL은 직렬가능+복구가능, rigorous 2PL은 더 엄격한 잠금 유지이지만 교착상태 방지는 아니다.

NoteBase