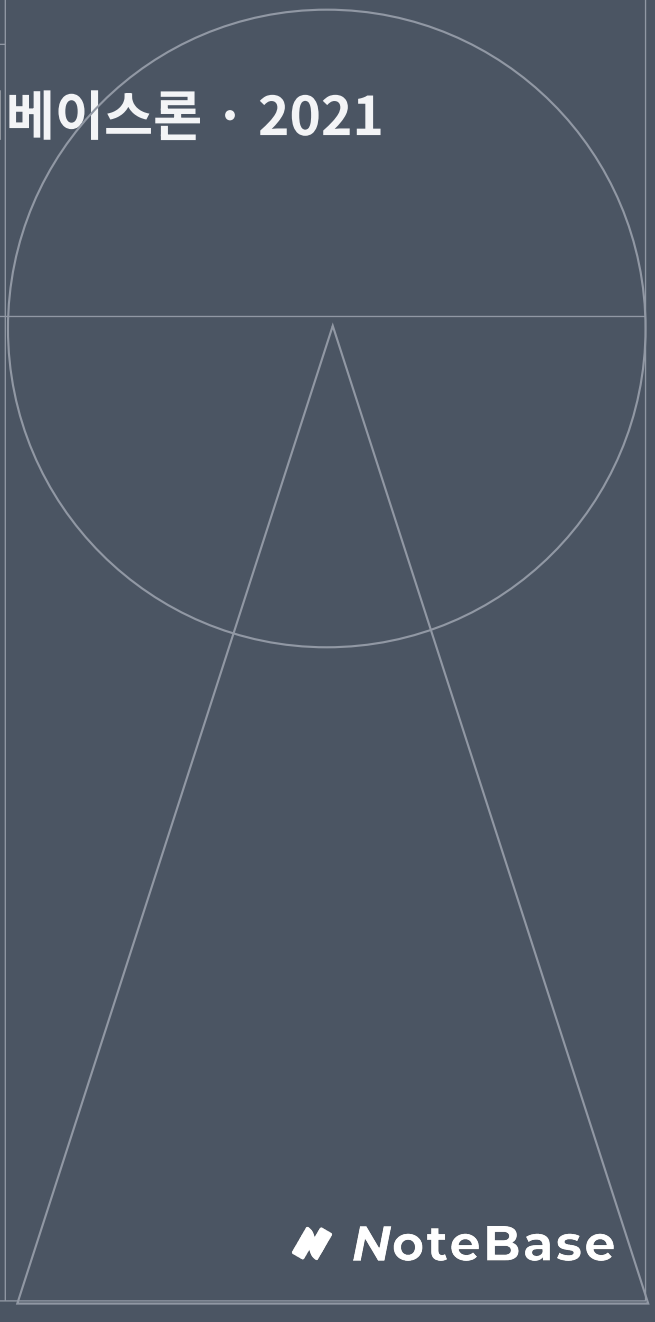
기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2021



NoteBase

기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2021

공무원/서울시/7급 · 데이터베이스론 · 2021

문 1. 트랜잭션의 ACID 특성 중 DBMS의 회복(recovery) 기능과 가장 밀접한 관련이 있는 특성을 옳게 짝지은 것은?

① 원자성(atomicity), 일관성(consistency)

② 일관성(consistency), 격리성(isolation)

③ 격리성(isolation), 영속성(durability)

④ 영속성(durability), 원자성(atomicity)

정답 4번

출제 포인트

이 문제는 ACID 네 특성 중 장애 발생 시 되돌리기와 재실행으로 구현되는 회복 기능이 원자성과 영속성에 연결된다는 점을 묻는 문제이다.

핵심 개념

원자성은 트랜잭션의 연산이 전부 수행되거나 전부 취소되어야 한다는 성질이다.
일관성은 트랜잭션 전후에 데이터베이스가 무결성 제약을 만족하는 올바른 상태를 유지해야 한다는 성질이다.
격리성은 동시에 수행되는 트랜잭션들이 서로의 중간 결과에 간섭하지 않아야 한다는 성질이다.
영속성은 성공적으로 완료된 트랜잭션의 결과가 장애가 발생해도 보존되어야 한다는 성질이다.
회복은 실패한 트랜잭션을 undo하여 원자성을 보장하고, commit된 결과를 redo하여 영속성을 보장하는 기능과 가장 직접적으로 연결된다.

NoteBase

선지별 해설

① X: 원자성은 회복과 직접 관련되지만 일관성은 회복 기능의 1차 대응 대상이 아니다. 틀린 부분은 "일관성"이다. 실제로는 "원자성, 영속성"의 짝이 회복과 가장 밀접하다.
② X: 일관성과 격리성은 각각 무결성 유지와 동시성 제어의 핵심 개념이다. 틀린 부분은 "일관성, 격리성"이다. 실제로 회복은 undo와 redo를 통해 원자성과 영속성을 보장한다.
③ X: 영속성은 회복과 직접 관련되지만 격리성은 주로 동시 실행 간섭을 막는 성질이다. 틀린 부분은 "격리성"이다. 실제로는 격리성 대신 원자성이 와야 한다.
④ O: 영속성은 commit된 결과를 장애 후에도 유지하는 성질이고, 원자성은 실패 시 전부 취소되게 하는 성질이므로 둘 다 회복 기능과 가장 밀접하다.

오답 포인트

회복을 장애 처리 전반으로 넓게 이해하면 일관성이나 격리성도 관련 있어 보이지만, DBMS 회복 기법의 직접 대상은 undo와 redo이다. ③처럼 영속성 하나만 보고 격리성을 함께 고르면 동시성 제어와 회복을 혼동한 것이다.

다시 나오면 이렇게 판단

문장에서 회복, 로그, undo, redo, commit 이후 보존, 실패 시 전부 취소가 보이면 먼저 영속성과 원자성을 떠올리면 된다. 반대로 간섭 방지이면 격리성, 무결성 유지이면 일관성으로 분리해 판단한다.

암기 포인트

회복은 undo로 원자성, redo로 영속성을 보장한다.

NoteBase

문 2. <보기>에서 학생 테이블과 학생 테이블을 참조하는 테이블을 함께 삭제하기 위한 SQL문에 들어갈 단어를 옳게 짝지은 것은?

<보기>
㉠_____ TABLE 학생 ㉡_____;

① ㉠: DROP, ㉡: RESTRICT
② ㉠: DROP, ㉡: CASCADE
③ ㉠: DELETE, ㉡: RESTRICT
④ ㉠: DELETE, ㉡: CASCADE

정답 2번

출제 포인트

이 문제는 테이블 삭제 명령인 DROP TABLE과 참조 관계 처리 옵션인 RESTRICT, CASCADE의 차이를 묻는 문제이다.

핵심 개념

DROP TABLE은 테이블 정의 자체를 삭제하는 명령이다.
DELETE는 테이블 안의 행만 삭제하고 테이블 구조는 남기는 명령이다.
RESTRICT는 다른 테이블이 현재 테이블을 참조하고 있으면 삭제를 허용하지 않는 옵션이다.
CASCADE는 참조 관계에 걸린 종속 객체나 참조 관계를 함께 정리하면서 삭제를 진행하는 옵션이다.

선지별 해설

① X: DROP은 테이블 삭제 명령이라는 점에서는 맞지만, RESTRICT가 틀린 부분이다. RESTRICT는 다른 테이블이 학생 테이블을 참조하고 있으면 삭제를 막는다. 틀린 표현은 "RESTRICT"이고, 올바르게는 "CASCADE"이다.
② O: DROP TABLE 학생 CASCADE는 학생 테이블을 삭제하면서 그 테이블을 참조하는 종속 관계도 함께 처리하는 형태이므로 문제의 조건에 맞는다.
③ X: 틀린 부분은 "DELETE"이다. DELETE는 테이블 자체를 없애는 명령이 아니라 행을 삭제하는 명령이므로 TABLE과 함께 이런 형태로 쓰지 않는다. 올바르게는 "DROP TABLE 학생 RESTRICT" 또는 "DROP TABLE 학생 CASCADE" 같은 DROP 구문을 사용한다.
④ X: 틀린 부분은 "DELETE"이다. CASCADE가 함께 있더라도 DELETE는 테이블 구조 자체를 삭제하는 명령이 아니다. 올바르게는 테이블을 함께 삭제할 때 "DROP TABLE 학생 CASCADE"를 사용한다.

NoteBase

오답 포인트

DELETE와 DROP을 같은 삭제 개념으로 뭉뚱그리면 틀리기 쉽다. DELETE는 데이터 행 삭제이고, DROP은 객체 정의 삭제이다. 또 RESTRICT와 CASCADE는 참조 관계가 있을 때 삭제를 막는지 함께 정리하는지를 가르는 단어라는 점이 함정이다.

다시 나오면 이렇게 판단

문장에서 "테이블을 삭제"한다고 하면 먼저 DROP인지 DELETE인지부터 구분한다. 그다음 "참조하는 테이블이 있음"이라는 조건이 보이면 RESTRICT는 삭제 불가, CASCADE는 함께 처리라고 판단하면 된다.

암기 포인트

DROP은 구조 삭제, DELETE는 행 삭제, RESTRICT는 막음, CASCADE는 함께 삭제이다.

NoteBase

문 3. 데이터베이스 설계 단계를 순서대로 바르게 나열한 것은?

① 요구사항 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현

② 요구사항 분석 → 개념적 설계 → 물리적 설계 → 논리적 설계 → 구현

③ 요구사항 분석 → 논리적 설계 → 물리적 설계 → 개념적 설계 → 구현

④ 요구사항 분석 → 논리적 설계 → 개념적 설계 → 물리적 설계 → 구현

정답 1번

출제 포인트

이 문제는 데이터베이스 설계의 표준 진행 순서에서 개념적 설계, 논리적 설계, 물리적 설계의 선후관계를 묻는 문제이다.

올바른 순서

1. 요구사항 분석 - 사용자 요구와 데이터 처리 요구를 수집한다.
2. 개념적 설계 - 요구사항을 바탕으로 ER 모델 등 개념 수준의 구조를 만든다.
3. 논리적 설계 - 개념 모델을 관계 스키마 등 DBMS 독립적인 논리 구조로 변환한다.
4. 물리적 설계 - 저장 구조, 인덱스, 접근 경로처럼 실제 저장 방식을 정한다.
5. 구현 - 설계 결과를 DBMS에 반영해 테이블과 제약조건 등을 생성한다.

왜 이 순서인가: 논리적 설계는 개념적 설계의 결과를 받아 스키마로 바꾸는 단계이고, 물리적 설계는 그 논리 스키마를 실제 저장 방식으로 구체화하는 단계이기 때문이다.

NoteBase

선지별 해설

① O
요구사항 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현의 순서가 맞다. 개념 모델이 먼저 있어야 이를 논리 스키마로 변환할 수 있고, 논리 스키마가 있어야 물리 저장 구조를 정할 수 있다.

② X
틀린 부분: 물리적 설계가 논리적 설계보다 앞에 배치되어 있다.
올바르게는: 요구사항 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현이다. 물리적 설계는 논리 구조가 확정된 뒤에 저장 장치와 인덱스를 정하는 단계이다.

③ X
틀린 부분: 논리적 설계를 개념적 설계보다 먼저 두었다.
올바르게는: 요구사항 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현이다. 요구사항으로부터 바로 논리 스키마를 만들기보다 먼저 현실 세계를 추상화한 개념 모델을 수립한다.

④ X
틀린 부분: 논리적 설계와 개념적 설계의 순서가 뒤바뀌어 있다.
올바르게는: 요구사항 분석 다음에는 개념적 설계가 오고, 그 다음이 논리적 설계이다. 개념적 설계는 엔터티와 관계를 정리하는 단계이고, 논리적 설계는 이를 테이블 구조로 옮기는 단계이다.

오답 포인트

개념적 설계와 논리적 설계를 같은 단계처럼 외우면 ③, ④처럼 순서를 뒤집기 쉽다. 또 논리적 설계와 물리적 설계를 혼동하면 ②처럼 저장 구조 결정을 너무 앞당기게 된다.

다시 나오면 이렇게 판단

데이터베이스 설계 순서 문제는 추상화 수준이 높은 단계에서 낮은 단계로 내려간다고 판단하면 된다. 요구 파악 후 개념 모델을 만들고, 이를 논리 스키마로 바꾼 뒤, 마지막에 실제 저장 방식과 구현으로 내려간다.

암기 포인트

요구 분석 → 개념 모델 → 논리 스키마 → 물리 저장 → 구현의 흐름이다.

NoteBase

문 4. 키의 관계로 가장 옳지 않은 것은?

① 후보키 ⊃ 기본키

② 후보키 ⊃ 대체키

③ 대체키 ⊃ 기본키

④ 슈퍼키 ⊃ 후보키

정답 3번

출제 포인트

이 문제는 기본키, 후보키, 대체키, 슈퍼키의 포함 관계를 정확히 구분하는지를 묻는 문제이다.

핵심 개념

후보키는 튜플을 유일하게 식별할 수 있으면서 더 이상 속성을 줄일 수 없는 최소 슈퍼키이다.
기본키는 여러 후보키 중에서 대표로 선택한 후보키이다.
대체키는 후보키 중 기본키로 선택되지 않은 나머지 후보키이다.
슈퍼키는 튜플을 유일하게 식별할 수 있는 속성 집합 전체를 말하며, 후보키보다 범위가 넓다.

선지별 해설

① O: 기본키는 후보키들 중 하나를 선정한 것이므로 기본키는 후보키에 포함된다. 따라서 후보키의 범주가 기본키보다 넓다고 볼 수 있다.
② O: 대체키는 기본키로 선택되지 않은 후보키이므로 모든 대체키는 후보키에 포함된다. 따라서 후보키가 대체키를 포함한다.
③ X: 틀린 부분은 "대체키 ⊃ 기본키"라는 표현이다. 대체키는 후보키 중 기본키를 제외한 나머지이므로 기본키를 포함하지 않는다. 올바르게는 "후보키 ⊃ 대체키" 또는 "대체키와 기본키는 서로 겹치지 않는다"이다.
④ O: 후보키는 최소성을 만족하는 슈퍼키이므로 모든 후보키는 슈퍼키에 포함된다. 슈퍼키는 후보키보다 더 넓은 개념이다.

오답 포인트

대체키를 기본키의 상위 개념처럼 착각하면 ③을 고르게 된다. 대체키는 기본키와 같은 후보키 계열이지만, 정의 자체가 "기본키로 선택되지 않은 후보키"이므로 기본키를 포함할 수 없다.

다시 나오면 이렇게 판단

먼저 후보키를 중심에 두고 분류하면 판단이 쉬워진다. 후보키 중 대표로 뽑힌 것이 기본키이고, 탈락한 나머지가 대체키이며, 후보키를 포함해 더 넓게 유일성만 만족하면 슈퍼키라고 정리하면 된다.

NoteBase

암기 포인트

후보키 = 최소 슈퍼키, 기본키 = 선택된 후보키, 대체키 = 선택되지 않은 후보키이다.

NoteBase

문 5. 파일 처리 시스템과 비교하여 데이터베이스 시스템이 갖는 장점으로 옳은 것들을 <보기>에서 모두 고른 것은?

<보기>
ㄱ. 데이터의 일관성(consistency)을 유지하는 것이 용이하다.
ㄴ. 데이터의 무결성(integrity)을 유지하는 것이 용이하다.
ㄷ. 다수의 사용자들이 동시에, 동일한 데이터에 접근하는 상황을 잘 처리한다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 4번

출제 포인트

이 문제는 파일 처리 시스템과 DBMS를 비교할 때 일관성, 무결성, 동시성 처리 능력이 모두 DBMS의 장점인지 묻는 문제이다.

핵심 개념

일관성은 여러 사용자와 여러 프로그램이 데이터를 다루어도 데이터 사이의 모순이 없도록 유지하는 성질이다.
무결성은 데이터가 정확하고 허용된 규칙을 만족하도록 유지하는 성질이다.
동시성 처리는 여러 사용자가 같은 데이터에 동시에 접근해도 충돌과 비정상 결과를 줄이도록 제어하는 기능이다.
DBMS는 제약조건, 트랜잭션, 동시성 제어 기능을 제공하므로 파일 처리 시스템보다 이 세 요소를 체계적으로 관리하기 쉽다.

NoteBase

선지별 해설

① X
- ①은 ㄱ, ㄴ만 포함하고 ㄷ을 빠뜨렸다.
- ㄷ도 옳다. DBMS는 동시성 제어와 트랜잭션 처리 기능으로 다수 사용자의 동시 접근을 잘 처리한다.
- 올바르게는 ㄱ, ㄴ, ㄷ을 모두 포함한 ④이다.

② X
- ②는 ㄱ, ㄷ만 포함하고 ㄴ을 빠뜨렸다.
- ㄴ도 옳다. DBMS는 제약조건과 통합 관리로 데이터 무결성을 유지하기 쉽다.
- 올바르게는 ㄱ, ㄴ, ㄷ을 모두 포함한 ④이다.

③ X
- ③은 ㄴ, ㄷ만 포함하고 ㄱ을 빠뜨렸다.
- ㄱ도 옳다. DBMS는 데이터 중복을 줄이고 통합된 데이터를 관리하므로 일관성 유지가 쉽다.
- 올바르게는 ㄱ, ㄴ, ㄷ을 모두 포함한 ④이다.

④ O
- ㄱ, ㄴ, ㄷ이 모두 맞으므로 ④가 옳다.
- 파일 처리 시스템보다 DBMS는 일관성 유지, 무결성 유지, 다중 사용자 동시 접근 처리에서 모두 유리하다.

오답 포인트

일관성과 무결성을 같은 뜻으로 보거나, 동시성 처리를 운영체제의 일반 기능 정도로만 생각하면 ㄷ을 제외한 조합을 고르기 쉽다. 이 문항은 DBMS가 데이터 품질과 동시 접근을 함께 관리한다는 점을 묻는다.

다시 나오면 이렇게 판단

파일 처리 시스템과 DBMS 비교 문제에서는 중복 감소에 따른 일관성, 제약조건에 따른 무결성, 트랜잭션과 락에 따른 동시성 처리를 먼저 대응시키면 조합형 선택지를 빠르게 판단할 수 있다.

암기 포인트

DBMS는 일관성, 무결성, 동시성 처리를 통합 관리하는 시스템이다.

NoteBase

문 6. <보기 1>의 두 릴레이션 C, O에 대한 연산 후 <보기 2>의 결과 릴레이션을 얻기 위한 연산자로 가장 옳은 것은?

<보기 1>

C

ID	NAME	AGE	GRADE
AAA	TOM	22	S
BBB	JAMES	23	A
CCC	AMY	21	B
DDD	ANNE	20	B

O

NO	ID	ORDER	AMOUNT
10001	AAA	BOOK	2
10002	CCC	PIZZA	1
10003	BBB	SNACK	10

<보기 2>

결과 릴레이션

ID	NAME	AGE	GRADE	NO	ORDER	AMOUNT
AAA	TOM	22	S	10001	BOOK	2
BBB	JAMES	23	A	10003	SNACK	10
CCC	AMY	21	B	10002	PIZZA	1

① σ

② ⊠

③ ÷

④ π

정답 2번

NoteBase

출제 포인트

이 문제는 두 릴레이션의 공통 속성인 ID를 기준으로 튜플을 결합해 결과 릴레이션을 만드는 조인 연산을 묻는 문제이다.

자료 읽는 순서

1. 두 릴레이션 C와 O에 공통으로 있는 속성이 ID인지 확인한다.
2. 결과 릴레이션이 C의 속성(NAME, AGE, GRADE)과 O의 속성(NO, ORDER, AMOUNT)을 함께 갖는지 확인한다.
3. 결과 행이 AAA, BBB, CCC만 남고 DDD가 빠진 이유가 O에 일치하는 ID가 있는 튜플만 결합되었기 때문인지 연결한다.

핵심 근거

결과 릴레이션은 C의 속성과 O의 속성을 한 행에 함께 보여 준다. 또한 공통 속성 ID가 같은 AAA, BBB, CCC만 남고 O에 대응 튜플이 없는 DDD는 사라진다. 이는 공통 속성을 기준으로 두 릴레이션을 결합하는 조인 연산의 결과와 일치한다.

선지별 해설

① X
σ는 한 릴레이션에서 조건에 맞는 행만 선택하는 연산이다. C나 O에서 일부 튜플을 고를 수는 있어도 두 릴레이션의 속성을 합쳐 새 릴레이션을 만들 수는 없다. 틀린 표현은 '결합 결과를 만든다'라는 전제이며, 올바르게는 'σ는 단일 릴레이션에서 조건에 맞는 튜플을 추출한다'이다.

② O
⨝는 공통 속성 값을 기준으로 두 릴레이션을 결합하는 조인 연산이다. 결과 릴레이션은 ID가 같은 튜플끼리 묶여 C의 NAME, AGE, GRADE와 O의 NO, ORDER, AMOUNT가 함께 나타난다. O에 대응 행이 없는 DDD가 제외된 점도 조인 결과와 맞는다.

③ X
÷는 어떤 집합의 모든 값과 관계를 만족하는지를 구할 때 쓰는 나눗셈 연산이다. 현재 결과처럼 속성 수가 늘어나고 상세 주문 정보가 붙는 형태는 나눗셈으로 나오지 않는다. 틀린 표현은 '매칭된 튜플 결합'이며, 올바르게는 '÷는 포함 관계나 전칭 조건을 만족하는 값을 찾는 연산'이다.

④ X
π는 필요한 열만 남기는 투영 연산이다. 투영을 하면 속성 수가 줄어들 뿐 C와 O의 열이 합쳐지지 않는다. 틀린 표현은 '새로운 속성을 붙인다'라는 전제이며, 올바르게는 'π는 기존 릴레이션에서 지정한 속성만 추출한다'이다.

NoteBase

오답 포인트

선택과 투영은 각각 행과 열을 줄이는 연산인데, 둘 다 결과 모양이 달라진다는 이유만으로 조인과 혼동하기 쉽다. 이 문항의 핵심 함정은 결과 릴레이션의 열 수가 늘었고 공통 ID가 같은 행만 남았다는 조인 신호를 놓치게 만드는 데 있다.

다시 나오면 이렇게 판단

결과 릴레이션에 두 릴레이션의 속성이 함께 나타나면 먼저 조인을 의심한다. 그다음 공통 속성으로 연결되었는지와 일치하지 않는 튜플이 제거되었는지를 확인하면 selection, projection, division과 구별할 수 있다.

암기 포인트

행만 고르면 σ, 열만 고르면 π, 두 릴레이션을 붙이면 ⨝이다.

NoteBase

문 7. <보기>의 내용을 표현한 ER 다이어그램으로 가장 옳은 것은?

<보기>
모든 기계는 최소 한 종류 이상의 물품을 생산한다.
모든 종류의 물품은 기계를 사용하지 않고 생산되거나, 최대 하나의 기계에서만 생산된다.

① 

② 

③ 

④ 

정답 2번

출제 포인트

이 문제는 ER 다이어그램에서 카디널리티 비율과 전체 참여/부분 참여를 함께 읽어, 기계는 반드시 생산 관계에 참여하고 물품종류는 없어도 되거나 최대 한 기계에만 연결된다는 조건을 도식으로 옮길 수 있는지를 묻는 문제이다.

NoteBase

선지별 해설

① X
기계 쪽 카디널리티 1, 물품종류 쪽 카디널리티 N은 맞지만, 물품종류 쪽이 이중선이라 모든 물품종류가 반드시 어떤 기계에 의해 생산되어야 한다는 뜻이 된다. 보기에서는 모든 종류의 물품이 기계를 사용하지 않고 생산되거나 최대 하나의 기계에서만 생산된다고 했으므로 물품종류는 부분 참여가 맞다. 틀린 표현은 물품종류의 전체 참여 표시이고, 올바르게는 기계 쪽만 전체 참여 이중선이어야 한다.

② O
기계 쪽이 이중선이므로 모든 기계가 최소 한 번은 생산 관계에 참여한다는 뜻이 되어 "모든 기계는 최소 한 종류 이상의 물품을 생산한다"와 맞다. 또한 기계 쪽 1, 물품종류 쪽 N은 한 기계가 여러 물품종류를 생산할 수 있고 각 물품종류는 최대 한 기계와만 연결된다는 뜻이므로 보기의 최대 하나의 기계에서만 생산된다는 조건과도 일치한다.

③ X
물품종류 쪽 이중선이라 모든 물품종류가 반드시 어떤 기계와 연결되어야 하므로 "기계를 사용하지 않고 생산되거나"라는 조건을 반영하지 못한다. 또한 기계 쪽 N, 물품종류 쪽 1은 한 기계가 최대 한 종류만 생산하고 한 물품종류는 여러 기계가 생산할 수 있다는 뜻이 된다. 틀린 표현은 카디널리티 방향과 물품종류의 전체 참여 표시이며, 올바르게는 기계 쪽 1, 물품종류 쪽 N이고 기계 쪽만 이중선이다.

④ X
기계 쪽 이중선은 맞지만, 기계 쪽 N과 물품종류 쪽 1은 한 기계가 최대 한 종류만 생산하고 한 물품종류를 여러 기계가 생산할 수 있다는 의미가 된다. 보기에서는 한 기계가 최소 한 종류 이상, 즉 여러 종류를 생산할 수 있고 각 물품종류는 최대 한 기계에서만 생산된다고 했으므로 카디널리티가 반대로 표시되었다. 틀린 표현은 N:1 비율이고, 올바르게는 1:N 비율이다.

오답 포인트

"모든 기계"와 "모든 종류의 물품"을 둘 다 전체 참여로 읽는 실수가 잦다. 이 문제에서 전체 참여는 기계에만 걸리고, 물품종류는 "기계를 사용하지 않고 생산되거나"라는 문구 때문에 관계에 참여하지 않아도 되는 부분 참여라는 점이 함정이다.

다시 나오면 이렇게 판단

문장을 두 부분으로 나누어 읽으면 된다. "모든 A는 최소 하나 이상 B와 연결된다"는 A 쪽 전체 참여 이중선으로, "모든 B는 없거나 최대 하나의 A와 연결된다"는 B 쪽 부분 참여 단일선과 최대 카디널리티 1로 옮기면 된다.

암기 포인트

전체 참여는 반드시 연결, 카디널리티 1은 최대 하나 연결이라는 뜻이다.

NoteBase

문 8. 데이터 사전 및 데이터 디렉터리에 대한 설명으로 가장 옳은 것은?

① 데이터 사전은 DBMS가 스스로 생성하고 유지한다.

② 데이터 사전은 데이터에 접근하는 데 필요한 위치 정보를 관리한다.

③ 데이터 사전에 DBMS는 접근할 수 있지만, 일반 사용자는 접근할 수 없다.

④ 데이터 디렉터리에는 일반 사용자가 접근할 수 있다.

정답 1번

출제 포인트

데이터 사전과 데이터 디렉터리의 역할 차이, 즉 메타데이터 관리와 물리적 위치 정보 관리의 구분을 묻는 문제이다.

핵심 개념

데이터 사전은 DBMS가 생성하고 유지하는 메타데이터 저장소이다. 테이블명, 속성명, 자료형, 제약조건 같은 데이터의 정의와 구조를 담는다.
데이터 디렉터리는 실제 데이터에 접근하기 위한 저장 위치와 접근 경로 같은 물리적 위치 정보를 관리하는 시스템용 정보이다.
두 개념의 차이는 데이터의 의미와 구조를 설명하느냐, 데이터의 위치와 경로를 설명하느냐에 있다.

NoteBase

선지별 해설

① O
데이터 사전은 시스템 카탈로그로서 DBMS가 자동으로 생성하고 유지하는 메타데이터 집합이다. 따라서 DBMS가 스스로 생성하고 유지한다는 설명이 맞다.

② X
틀린 부분은 "데이터 사전은 ... 위치 정보를 관리한다"는 표현이다.
올바르게는 데이터 디렉터리가 데이터 접근에 필요한 위치 정보와 접근 경로를 관리하고, 데이터 사전은 데이터의 정의와 구조를 관리한다.

③ X
틀린 부분은 "일반 사용자는 접근할 수 없다"는 표현이다.
올바르게는 데이터 사전은 DBMS가 관리하는 메타데이터 저장소이지만, 사용자나 DBA가 조회하여 데이터 구조를 확인할 수 있는 대상이다.

④ X
틀린 부분은 "일반 사용자가 접근할 수 있다"는 표현이다.
올바르게는 데이터 디렉터리는 데이터의 물리적 위치 정보를 담는 시스템 내부 정보이므로 일반 사용자가 직접 접근하는 대상이 아니다.

오답 포인트

데이터 사전과 데이터 디렉터리를 모두 데이터에 관한 정보라고만 외우면 ②와 ④를 헷갈리기 쉽다. 사전은 메타데이터, 디렉터리는 위치 정보라는 역할 차이를 잡아야 한다.

다시 나오면 이렇게 판단

선지에 테이블 정의, 속성, 자료형, 제약조건처럼 구조와 의미를 나타내는 말이 나오면 데이터 사전으로 판단한다.
저장 위치, 주소, 접근 경로처럼 물리적 위치를 나타내는 말이 나오면 데이터 디렉터리로 판단한다.

암기 포인트

데이터 사전은 의미와 구조, 데이터 디렉터리는 위치와 경로를 다룬다.

NoteBase

문 9. 데이터베이스 스키마(database schema)가 <보기>와 같을 때, 문법적으로 가장 옳지 않은 관계 대수(relational algebra) 식은?

<보기>
professor(id, name, department, tel)
teach(id, course_id, section_id, semester, year)
section(course_id, section_id, semester, year, classroom_no)

① $\pi_{name}(\sigma_{department = 'Biology'}(professor))$

② $\pi_{id, name, tel}(professor)$

③ _course_id(_semester='Fall' year=2020(section)) _course_id, section_id(_semester='Spring' year=2021(section))

④ $\sigma_{professor.id = teach.id}(professor \times teach)$

정답 3번

출제 포인트

이 문제는 선택, 투영, 카티션곱 뒤 선택은 정상적인지와 합집합의 합병 가능 조건을 구별할 수 있는지를 묻는다.

핵심 개념

선택은 조건에 맞는 튜플만 고르는 연산이고, 투영은 필요한 속성만 남기는 연산이다.
합집합은 두 결과 릴레이션의 차수와 대응 속성 구성이 같아야 하는 합병 가능 관계에서만 가능하다.
카티션곱 뒤에 선택 조건을 붙이면 세타 조인과 같은 형태로 쓸 수 있다.

선지별 해설

1번 O: professor에서 department='Biology'인 튜플을 선택한 뒤 name만 투영한 식이다. 선택 결과의 속성 중 name이 존재하므로 문법적으로 자연스럽다.
2번 O: professor에서 id, name, tel만 투영한 식이다. 세 속성 모두 professor 스키마에 있으므로 올바른 투영식이다.
3번 X: 왼쪽은 course_id만 남겨 1개 속성 릴레이션이 되고, 오른쪽은 course_id와 section_id를 남겨 2개 속성 릴레이션이 된다. 합집합은 두 피연산자의 속성 수와 대응 구조가 같아야 하는데 여기서는 다르므로 옳지 않다. 틀린 표현은 두 투영 결과를 그대로 합집합한 부분이고, 올바르게는 두 쪽 모두 π_{course_id}로 맞추거나 두 쪽 모두 $\pi_{course_id, section_id}$로 맞춘 뒤 합집합해야 한다.
4번 O: professor × teach로 카티션곱을 만든 뒤 professor.id=teach.id 조건으로 선택한 식이다. 이는 조인 조건을 선택으로 표현한 정상적인 관계대수식이다.

NoteBase

오답 포인트

3번은 각 부분식이 따로 보면 그럴듯해서 지나치기 쉽다. 그러나 합집합 기호가 보이면 먼저 두 결과의 속성 개수와 대응 속성이 같은지부터 봐야 한다.

다시 나오면 이렇게 판단

관계대수식의 문법을 묻는 문제에서는 연산자별 입력 조건을 먼저 확인한다. 특히 합집합, 교집합, 차집합은 두 결과가 합병 가능인지부터 보고, 투영 뒤 남는 속성 수가 달라지지 않는지를 바로 점검한다.

암기 포인트

합집합은 내용보다 먼저 형태를 맞춰야 한다. 차수와 대응 속성이 다르면 합칠 수 없다.

NoteBase

문 10. 2단계 로킹 규약(2PLP: two-phase locking protocol)을 준수하여 <보기>의 연산을 실행하기 위하여 ㉠, ㉡, ㉢, ㉣에 들어갈 용어를 가장 옳게 짝지은 것은? (단, lock-X는 배타로크이고, lock-S는 공유로크이다.)

<보기>
㉠

————————
read(B)
B = B + 100
write(B)
㉡

————————
read(A)
A = A + 100
write(A)
㉢

————————
㉣

————————

① ㉠: lock-X(B), ㉡: lock-X(A), ㉢: unlock(B), ㉣: unlock(A)

② ㉠: lock-S(B), ㉡: lock-X(A), ㉢: unlock(B), ㉣: unlock(A)

③ ㉠: lock-X(B), ㉡: unlock(B), ㉢: lock-S(A), ㉣: unlock(A)

④ ㉠: lock-S(B), ㉡: unlock(B), ㉢: lock-S(A), ㉣: unlock(A)

정답 1번

출제 포인트

이 문제는 2단계 로킹 규약에서 배타락과 공유락의 구분, 그리고 성장 단계에서는 락만 획득하고 축소 단계에 들어간 뒤에야 해제한다는 순서를 묻는 문제이다.

NoteBase

올바른 순서

1. ㄱ: lock-X(B) — B를 읽고 값을 변경한 뒤 write(B)까지 수행하므로 처음부터 배타락이 필요하다.
2. ㄴ: lock-X(A) — B에 대한 락을 아직 풀지 않은 상태에서 A를 갱신하기 위한 배타락을 추가로 획득한다.
3. ㄷ: unlock(B) — 필요한 락을 모두 얻은 뒤 축소 단계로 들어가면서 B의 락을 해제한다.
4. ㄹ: unlock(A) — 마지막으로 A의 락을 해제한다.

왜 이 순서: 2PL은 락을 획득하는 성장 단계와 락을 해제하는 축소 단계가 분리된다. 따라서 A에 대한 락을 얻기 전에 B의 락을 먼저 해제하면 2PL을 위반한다.

선지별 해설

① O

B와 A 모두 값을 변경하는 연산이므로 각각 배타락이 필요하다. 또 두 데이터 항목의 락을 모두 획득한 뒤에 unlock을 시작하므로 2PL의 성장 단계와 축소 단계를 지킨다.

② X

틀린 부분: ㄱ의 lock-S(B)

올바르게는: B는 read(B)만 하는 것이 아니라 B=B+100 후 write(B)까지 수행하므로 lock-X(B)이어야 한다.

③ X

틀린 부분: ㄴ의 unlock(B)

올바르게는: B의 락을 먼저 풀지 말고 A에 대한 lock-X(A)를 먼저 획득한 뒤, 그 다음에 unlock(B)를 해야 한다. 중간에 unlock이 나오면 이후 새 락을 다시 얻을 수 없어 2PL에 어긋난다.

④ X

틀린 부분: ㄱ의 lock-S(B), ㄴ의 unlock(B), ㄷ의 lock-S(A)

올바르게는: B와 A는 모두 write 연산이 있으므로 둘 다 lock-X가 필요하고, unlock은 모든 필요한 락을 얻은 뒤에 시작해야 한다.

오답 포인트

read가 먼저 보인다는 이유로 B에 공유락을 거는 경우가 많다. 그러나 뒤에 write(B)가 있으면 처음부터 배타락이 필요하다. 또 한 항목 작업이 끝났다고 바로 unlock(B)를 넣기 쉽지만, 그 뒤에 A에 대한 새 락을 획득해야 하므로 그렇게 하면 2PL의 핵심인 성장 단계 종료 전 unlock 금지 원칙을 깨게 된다.

다시 나오면 이렇게 판단

절차형 락 문제는 각 데이터 항목에 write가 있는지부터 확인한 뒤 lock-S와 lock-X를 정한다. 그 다음 전체 연산에서 필요한 락을 모두 모으는 구간이 끝났는지 보고, 새 락 획득 뒤에는 unlock이 나오고 unlock 뒤에는 새 lock이 나오지 않는지 판단한다.

NoteBase

암기 포인트

2PL은 먼저 모두 잠그고 나중에 푼다. write가 있으면 공유락이 아니라 배타락이다.

NoteBase

문 11. 데이터 마이닝(data mining)의 연관규칙(association rules)에 대한 <보기>의 설명에서, (가)와 (나)가 설명하는 용어로 가장 옳은 것은?

<보기>
(가) 전체 거래 중에서 항목집합 X와 Y를 모두 포함하는 거래의 비율
(나) 항목집합 X를 포함하는 거래 중에서 항목집합 Y도 포함하는 거래의 비율

① (가): 지지도(support), (나): 향상도(lift)
② (가): 지지도(support), (나): 신뢰도(confidence)
③ (가): 신뢰도(confidence), (나): 지지도(support)
④ (가): 신뢰도(confidence), (나): 향상도(lift)

정답 2번

출제 포인트

연관규칙의 대표 지표인 지지도와 신뢰도의 정의를 구분하는 문제이다.

핵심 개념

지지도는 전체 거래를 분모로 하여 항목집합 X와 Y를 함께 포함하는 거래의 비율이다.
신뢰도는 X를 포함한 거래를 분모로 하여 그중 Y도 함께 포함하는 거래의 비율이다.
향상도는 신뢰도를 Y의 지지도로 나눈 값으로, X와 Y의 독립성 대비 연관 강도를 본다.

선지별 해설

① X: (가)를 지지도라고 본 점은 맞지만, (나)를 향상도라고 본 점이 틀리다. (나)는 X가 주어졌을 때 Y가 함께 나타날 조건부 비율이므로 신뢰도이다. 틀린 표현은 "향상도"이고, 올바르게는 "신뢰도"이다.
② O: (가)는 전체 거래 중 X와 Y를 모두 포함하는 거래의 비율이므로 지지도이다. (나)는 X를 포함하는 거래 중 Y도 포함하는 거래의 비율이므로 신뢰도이다.
③ X: (가)를 신뢰도라고 본 점이 틀리다. (가)는 전체 거래를 기준으로 하므로 지지도이다. 틀린 표현은 "신뢰도"이고, 올바르게는 "지지도"이다. 또한 (나)를 지지도라고 본 점도 틀리며, 올바르게는 "신뢰도"이다.
④ X: (가)를 신뢰도라고 본 점이 틀리고, (나)를 향상도라고 본 점도 틀리다. (가)는 전체 거래 기준이므로 지지도이고, (나)는 X를 포함한 거래 기준의 조건부 비율이므로 신뢰도이다.

오답 포인트

지지도와 신뢰도는 모두 비율이라서 분모를 놓치면 쉽게 뒤바뀐다. 이 문항의 함정은 (가)는 전체 거래가 분모이고, (나)는 X를 포함한 거래가 분모라는 점을 구별하지 못하게 만드는 데 있다. 향상도는 조건부 비율 자체가 아니라 신뢰도를 독립 기준과 비교한 값이라는 점도 자주 혼동된다.

NoteBase

다시 나오면 이렇게 판단

분모를 먼저 확인하면 된다. 전체 거래가 분모이면 지지도이고, X를 포함한 거래가 분모이면 신뢰도이다. 다른 사건의 기준 확률과 비교하는 비율이 나오면 그때 향상도를 검토한다.

암기 포인트

지지도는 전체 분모, 신뢰도는 X 분모, 향상도는 신뢰도 대비 독립 기준이다.

NoteBase

문 12. 릴레이션 R(A, B, C, D, E)에 대해서 <보기>와 같이 함수적 종속성(functional dependency)이 있을 때, 릴레이션 R을 릴레이션 R1과 R2로 나눈 것 중 무손실 분해(lossless decomposition) 성질을 만족하는 것은?

<보기>

A → CD

① R1(A, C, D), R2(A, B, E)

② R1(A, C, D), R2(B, C, E)

③ R1(A, D, E), R2(A, B, C)

④ R1(A, D, E), R2(B, C, D)

정답 1번

출제 포인트

이 문제는 두 릴레이션으로 분해할 때 공통 속성이 한쪽 릴레이션 전체를 함수적으로 결정하면 무손실 분해가 된다는 조건을 묻는다.

핵심 개념

무손실 분해는 분해한 릴레이션들을 자연조인했을 때 원래 릴레이션이 그대로 복원되는 분해이다. 두 릴레이션 분해에서는 공통 속성 $R1 \cap R2$에 대하여 $R1 \cap R2 \rightarrow R1$ 또는 $R1 \cap R2 \rightarrow R2$가 성립하면 된다. 이 문항에서는 $A \rightarrow CD$만 주어졌으므로 교집합이 무엇인지와 그 교집합이 한쪽 릴레이션 전체를 결정하는지를 보면 된다.

선지별 해설

① O. 공통 속성은 A이다. 주어진 함수 종속성 $A \rightarrow CD$와 자명한 종속성으로부터 $A \rightarrow ACD$가 성립하므로 공통 속성 A가 R1(A, C, D) 전체를 결정한다. 따라서 무손실 분해 조건을 만족한다.
② X. 공통 속성은 C이다. 주어진 조건으로는 $C \rightarrow A$, $C \rightarrow B$, $C \rightarrow E$를 얻을 수 없으므로 C가 어느 한 릴레이션 전체도 결정하지 못한다. 올바르게는 교집합이 A가 되도록 R1(A, C, D), R2(A, B, E)처럼 분해해야 한다.
③ X. 공통 속성은 A이다. 그러나 A로부터 D와 C는 구할 수 있어도 E나 B는 구할 수 없으므로 $A \rightarrow ADE$도 아니고 $A \rightarrow ABC$도 아니다. 올바르게는 A가 한쪽 릴레이션 전체를 결정하도록 R1(A, C, D), R2(A, B, E)처럼 분해해야 한다.
④ X. 공통 속성은 D이다. 주어진 조건은 $A \rightarrow CD$뿐이므로 D만으로 A, B, C, E를 결정할 수 없다. 따라서 D가 어느 한 릴레이션 전체도 결정하지 못한다. 올바르게는 교집합이 A이고 A가 한쪽 릴레이션을 결정하는 분해를 선택해야 한다.

NoteBase

오답 포인트

교집합에 A가 들어 있다는 이유만으로 ③을 무손실 분해로 보기 쉽다. 핵심은 교집합의 존재 자체가 아니라 그 교집합이 실제로 한쪽 릴레이션 전체를 결정하는지 여부이다.

다시 나오면 이렇게 판단

두 릴레이션 분해 문제는 먼저 교집합을 찾고, 그 교집합으로 각 릴레이션 전체를 함수적으로 결정할 수 있는지 본다. 교집합이 X이면 $X \rightarrow R1$ 또는 $X \rightarrow R2$가 성립하는 선택지만 남기면 된다.

암기 포인트

무손실 분해는 교집합이 한쪽 릴레이션을 결정할 때 성립한다. 이 문항에서는 A가 CD를 결정하므로 A를 교집합으로 갖고 R1(A, C, D)을 결정하는 분해가 정답이다.

NoteBase

문 13. 관계형 데이터베이스 관리 시스템(RDBMS)에 해당하지 않는 것은?

① Oracle

② IMS

③ MySQL

④ DB2

정답 2번

출제 포인트

이 문제는 관계형 데이터베이스 관리 시스템과 계층형 데이터베이스 관리 시스템을 구분하는 문제이다.

핵심 개념

관계형 DBMS는 데이터를 테이블 형태로 저장하고 SQL로 조작하는 시스템이다.
IMS는 대표적인 계층형 DBMS이고, Oracle·MySQL·DB2는 대표적인 관계형 DBMS이다.

선지별 해설

① O
Oracle은 테이블 기반의 대표적인 관계형 DBMS이므로 RDBMS에 해당한다.
② X
IMS는 IBM의 대표적인 계층형 DBMS이므로 관계형 DBMS에 해당하지 않는다.
틀린 표현은 "RDBMS에 해당"이다.
올바르게는 "IMS는 계층형 DBMS에 해당한다"이다.
③ O
MySQL은 테이블과 SQL을 기반으로 데이터를 처리하는 대표적인 관계형 DBMS이다.
④ O
DB2는 IBM의 관계형 DBMS 제품이므로 RDBMS에 해당한다.

오답 포인트

IMS와 DB2가 모두 IBM 계열 제품이라는 점 때문에 같은 유형의 DBMS로 오해하기 쉽다. 이 문항의 함정은 제조사가 아니라 데이터 모델을 기준으로 분류해야 한다는 점이다.

다시 나오면 이렇게 판단

제품명을 보면 먼저 데이터 저장 구조를 확인한다. 테이블과 SQL 중심이면 관계형이고, 부모-자식 트리 구조 중심이면 계층형으로 판단할 수 있다.

NoteBase

암기 포인트

IMS는 계층형, Oracle·MySQL·DB2는 관계형이라는 구분이 핵심이다.

NoteBase

문 14. <보기 1>의 customer 릴레이션에서 <보기 2>의 SQL 질의를 수행할 때 예상되는 출력 결과는?

<보기 1>

customer

id	name	field_interest	point
1	Sally	Comp. Sci.	7,000
3	Mozart	Music	1,000
8	Nick	History	2,200
12	Brook	Fiction	5,800
13	Katz	Finance	6,000
16	Jenny	Health	5,300
19	Einstein	Physics	3,100
20	Crick	Music	2,500
24	Ella	Comp. Sci.	8,000
27	James	History	4,200
31	Hailey	Fiction	3,700
34	El Said	Finance	1,100

<보기 2>

```
select count (distinct id) from customer
where field_interest like '%ic%';
```

① 1

② 3

③ 5

④ 7

정답 3번

출제 포인트

이 문제는 LIKE '%ic%' 패턴에 매칭되는 field_interest 값을 찾고, 그 행들의 id 개수를 count(distinct id)로 세는지를 묻는다.

NoteBase

자료 읽는 순서

1. WHERE 조건인 field_interest like '%ic%'를 field_interest 값마다 적용한다.
2. 'ic'가 들어 있는 값만 남기면 Music, Fiction, Physics가 해당된다.
3. 해당 행의 id는 3, 12, 19, 20, 31이고 모두 서로 달라 count(distinct id)의 결과는 5가 된다.

선지별 해설

① X: 결과를 1로 보는 것은 '%ic%'에 맞는 값을 하나로 잘못 본 경우이다. 표에서 Music은 id 3, 20의 2행이고, Fiction은 id 12, 31의 2행이며, Physics는 id 19의 1행이므로 총 5행이다. 틀린 표현은 결과가 1이라는 판단이고, 올바르게는 결과가 5이다.
② X: 결과를 3으로 보는 것은 Music, Fiction, Physics의 종류 수만 센 경우이다. SQL은 field_interest의 distinct 개수가 아니라 distinct id의 개수를 세므로 3이 아니다. 틀린 표현은 매칭된 관심분야 종류 수를 답으로 본 점이고, 올바르게는 매칭된 행의 서로 다른 id 수 5를 센다.
③ O: '%ic%'에 매칭되는 행은 Music 2개, Fiction 2개, Physics 1개로 총 5개이고, 이들 id는 3, 12, 19, 20, 31로 모두 다르다. 따라서 count(distinct id)의 결과는 5가 된다.
④ X: 결과를 7로 보는 것은 Comp. Sci.나 Finance까지 포함해 'ic' 문자열 판정을 잘못한 경우이다. Comp. Sci.에는 'ic'가 연속으로 들어 있지 않고 Finance에도 'ic'가 없다. 틀린 표현은 7개가 매칭된다는 판단이고, 올바르게는 5개가 매칭된다.

오답 포인트

Comp. Sci.를 Science 전체처럼 보고 'ic'가 있다고 착각하거나, count(distinct id)를 매칭된 값의 종류 수로 오해하면 틀리기 쉽다. 이 표에서는 id가 모두 유일하므로 distinct는 개수를 줄이지 않는다는 점도 함정이다.

다시 나오면 이렇게 판단

LIKE 문제는 먼저 패턴에 실제로 들어맞는 문자열만 표시하고, 그다음 count 대상 컬럼이 무엇인지 확인한다. distinct가 붙으면 그 컬럼의 중복만 제거하되, 이 문항처럼 id가 원래 모두 다르면 단순 행 개수와 같아진다.

암기 포인트

LIKE는 문자열 매칭, count(distinct 컬럼)은 그 컬럼의 서로 다른 값 개수이다.

NoteBase

문 15. 빅데이터(big data)를 저장, 처리 및 관리하기 위해 사용되는 기술을 <보기>에서 모두 고른 것은?

<보기>
ㄱ. HDFS(Hadoop Distributed File System)
ㄴ. MapReduce
ㄷ. ZooKeeper

① ㄱ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 4번

출제 포인트

이 문제는 빅데이터 플랫폼에서 저장, 처리, 관리 역할을 맡는 대표 기술인 HDFS, MapReduce, ZooKeeper를 함께 구분하는 문제이다.

핵심 개념

HDFS는 대용량 데이터를 분산 저장하는 하둡의 파일 시스템이다.
MapReduce는 분산된 데이터를 병렬 처리하는 프로그래밍 모델이자 처리 방식이다.
ZooKeeper는 분산 시스템의 설정 정보, 이름 서비스, 동기화, 클러스터 조정을 맡는 관리 기술이다.
따라서 저장은 HDFS, 처리는 MapReduce, 관리는 ZooKeeper로 연결해서 보면 된다.

NoteBase

선지별 해설

① X
ㄱ인 HDFS만 고른 선택지이다. HDFS는 저장 기술로는 맞지만, 빅데이터 처리 기술인 MapReduce와 분산 환경 관리 기술인 ZooKeeper가 빠져 있다. 틀린 부분은 'ㄱ만 고른 것'이다. 올바르게는 'ㄱ, ㄴ, ㄷ 모두'이다.

② X
ㄱ, ㄷ을 골라 저장과 관리는 포함했지만 처리 기술인 ㄴ MapReduce가 빠져 있다. 틀린 부분은 'ㄴ이 빠진 조합'이다. 올바르게는 'HDFS, MapReduce, ZooKeeper를 모두 포함한 조합'이다.

③ X
ㄴ, ㄷ을 골라 처리와 관리는 포함했지만 저장 기술인 ㄱ HDFS가 빠져 있다. 틀린 부분은 'ㄱ이 빠진 조합'이다. 올바르게는 '저장용 HDFS까지 포함한 ㄱ, ㄴ, ㄷ'이다.

④ O
ㄱ HDFS는 분산 저장, ㄴ MapReduce는 분산 처리, ㄷ ZooKeeper는 분산 환경의 조정과 관리에 사용된다. 문제에서 묻는 저장, 처리 및 관리 기술을 모두 포함하므로 옳다.

오답 포인트

ZooKeeper를 저장소나 처리 엔진이 아니라는 이유로 아예 제외하면 틀리기 쉽다. 이 문항은 '빅데이터를 저장, 처리 및 관리하기 위해 사용되는 기술'을 묻기 때문에, 관리 역할의 ZooKeeper도 포함된다는 점이 함정이다.

다시 나오면 이렇게 판단

빅데이터 관련 기술 문제가 나오면 각 기술의 역할을 먼저 기능별로 나눈다. 저장은 HDFS, 처리는 MapReduce, 조정·관리는 ZooKeeper처럼 역할 축으로 판단하면 조합형 선지를 빠르게 고를 수 있다.

암기 포인트

HDFS는 저장, MapReduce는 처리, ZooKeeper는 관리이다.

NoteBase

문 16. 데이터베이스 시스템의 회복(recovery)에 대한 설명으로 가장 옳지 않은 것은?

① 장애로 인해 손상된 데이터베이스를 손상되기 이전의 정상적 상태로 복구하는 것이다.

② SW 오류는 물론 HW 오류까지 대비한다.

③ 검사점(checkpoint) 기법의 목적은 복구 작업에 소요되는 시간을 줄이기 위한 것이다.

④ WAL(Write–Ahead Logging) 기법은 데이터를 갱신한 후에, 로그에 기록을 남기는 방법이다.

정답 4번

출제 포인트

이 문제는 데이터베이스 회복의 기본 개념과 checkpoint, WAL의 핵심 원리를 구별하는 문제이다.

핵심 개념

회복은 장애로 손상된 데이터베이스를 일관된 정상 상태로 되돌리는 기능이다. checkpoint는 로그의 일정 지점을 기준으로 복구 범위를 줄여 복구 시간을 단축하는 기법이다. WAL은 데이터 페이지를 디스크에 쓰기 전에 관련 로그를 먼저 안정 저장 장치에 기록하는 원리이다.

선지별 해설

① O
장애로 인해 깨진 데이터베이스를 장애 이전의 일관된 정상 상태로 복구하는 것이 회복의 목적이므로 옳다.

② O
데이터베이스 회복은 트랜잭션 처리 중의 소프트웨어 오류뿐 아니라 시스템 장애나 저장장치 장애 같은 하드웨어 오류에도 대비하는 기능이므로 옳다.

③ O
checkpoint는 그 시점까지의 변경 내용을 기준점으로 삼아 장애 후 다시 조사해야 할 로그 범위를 줄인다. 따라서 복구 작업에 걸리는 시간을 줄이는 목적이 있으므로 옳다.

④ X
틀린 부분은 "데이터를 갱신한 후에, 로그에 기록을 남기는"이다. WAL은 데이터베이스 페이지를 디스크에 기록하기 전에 해당 변경 로그를 먼저 기록하는 방식이다. 올바르게는 "로그를 먼저 안정 저장 장치에 기록한 뒤 데이터 페이지를 기록하는 방법"이다.

NoteBase

오답 포인트

④는 'Write-Ahead'의 ahead를 놓치면 쉽게 속는다. '나중에 로그를 남긴다'가 아니라 '데이터 페이지보다 로그를 앞서 기록한다'는 순서가 핵심이다.

다시 나오면 이렇게 판단

회복 문제에서는 각 기법의 목적과 기록 순서를 먼저 본다. checkpoint는 복구 시간 단축, WAL은 로그 선기록, recovery는 장애 후 일관성 회복으로 연결하면 빠르게 판단할 수 있다.

암기 포인트

checkpoint는 복구 범위 축소, WAL은 로그 선기록, recovery는 정상 상태 복원이다.

NoteBase

문 17. <보기>에서 설명하는 생년월일, 주소의 속성 종류는?

<보기>
• 고객 개체의 생년월일 속성은 연, 월, 일로 의미를 세분화할 수 있다.
• 고객 개체의 주소 속성은 시(도), 구(군), 동, 우편번호 등으로 의미를 세분화할 수 있다.

① 단일 값 속성
② 다중 값 속성
③ 단순 속성
④ 복합 속성

정답 4번

출제 포인트

이 문제는 단순 속성과 복합 속성의 차이를 구별할 수 있는지를 묻는 문제이다.

핵심 개념

단순 속성은 더 이상 의미 있는 하위 요소로 나누기 어려운 속성이다.
복합 속성은 하나의 속성을 연, 월, 일이나 시, 구, 동처럼 여러 하위 속성으로 분해할 수 있는 속성이다.
다중 값 속성은 한 개체가 같은 속성값을 여러 개 가질 수 있는 속성이고, 단일 값 속성은 한 개체가 그 속성값을 하나만 가지는 속성이다.

선지별 해설

① X: 단일 값 속성은 한 개체에 대해 값이 하나만 존재하는 속성이라는 뜻이다. 보기의 생년월일과 주소는 값의 개수가 아니라 내부를 여러 부분으로 나눌 수 있다는 점이 핵심이므로 틀리다. 틀린 표현은 '단일 값'이고, 올바르게는 '복합 속성'이다.
② X: 다중 값 속성은 전화번호처럼 한 개체가 여러 값을 가질 수 있는 속성이다. 보기의 설명은 여러 값을 가진다는 뜻이 아니라 연, 월, 일 또는 시, 구, 동으로 세분화된다는 뜻이므로 틀리다. 틀린 표현은 '다중 값'이고, 올바르게는 '복합 속성'이다.
③ X: 단순 속성은 더 이상 분해되지 않는 원자적 속성이다. 보기의 생년월일과 주소는 각각 연·월·일, 시·구·동·우편번호로 나눌 수 있으므로 단순 속성이 아니다. 틀린 표현은 '단순'이고, 올바르게는 '복합'이다.
④ O: 복합 속성은 하나의 속성을 의미 있는 하위 속성들로 분해할 수 있는 속성이다. 생년월일을 연, 월, 일로 나누고 주소를 시, 구, 동, 우편번호 등으로 나누는 설명은 복합 속성에 정확히 해당한다.

NoteBase

오답 포인트

복합 속성과 다중 값 속성을 자주 혼동한다. 이 문항은 값이 여러 개인지보다 속성 내부를 하위 항목으로 분해할 수 있는지를 묻는다.

다시 나오면 이렇게 판단

속성이 여러 개의 값으로 반복되는지 먼저 보고, 아니라면 그 속성을 연·월·일처럼 의미 있는 구성 요소로 쪼갤 수 있는지 본다. 쪼갤 수 있으면 복합 속성, 쪼개기 어렵고 값만 하나면 단순 속성 또는 단일 값 속성으로 판단한다.

암기 포인트

값이 여러 개면 다중 값 속성이고, 속성 내부가 여러 항목으로 나뉘면 복합 속성이다.

NoteBase

문 18. <보기>의 트랜잭션의 무제어 동시 공용 사례 (가)와 (나)에서, 각각에 해당하는 문제점을 옳게 짝지은 것은?

<보기>

(가)

T_1	T_2
read(x) x ← x + 300	
	read(x) x ← x * 5
write(x)	
	write(x)

(나)

T_1	T_2
read(x) x ← x + 300 write(x)	
	read(x) x ← x * 5 write(x)
read(y) rollback T_1	

① (가): 갱신 분실 (lost update), (나): 모순성 (inconsistency)

② (가): 모순성 (inconsistency), (나): 연쇄 복귀 (cascading rollback)

③ (가): 갱신 분실 (lost update), (나): 연쇄 복귀 (cascading rollback)

④ (가): 연쇄 복귀 (cascading rollback), (나): 갱신 분실 (lost update)

정답 3번

출제 포인트

이 문제는 트랜잭션 스케줄의 읽기·쓰기·롤백 순서를 보고 무제어 동시 공용의 이상 현상을 식별할 수 있는지를 묻는 문제이다.

사례 정리

(가)는 T1이 x를 읽어 300을 더한 뒤 아직 반영하지 않은 상태에서 T2도 같은 x를 읽어 5배로 계산하고, 이후 T1의 write(x) 다음에 T2의 write(x)가 덮어쓴 사례이다.
(나)는 T1이 x를 갱신하여 write(x)까지 한 값을 T2가 읽어 다시 write(x)한 뒤, 나중에 T1이 rollback되는 사례이다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2021

핵심 개념

갱신 분실은 두 트랜잭션이 같은 데이터를 각각 갱신할 때 나중의 쓰기가 먼저의 갱신 결과를 덮어써 한쪽 갱신이 사라지는 현상이다.
연쇄 복귀는 어떤 트랜잭션이 아직 commit되지 않은 다른 트랜잭션의 값을 읽은 뒤, 원래 트랜잭션이 rollback되면서 그 값을 읽은 트랜잭션도 함께 rollback되어야 하는 현상이다.

적용 기준

(가)는 같은 x를 두 트랜잭션이 각각 계산한 뒤 마지막 write가 앞선 write 결과를 덮어쓰는지가 기준이다.
(나)는 T2가 T1의 미완료 갱신값을 읽었는지, 그리고 그 뒤 T1의 rollback 때문에 T2도 영향을 받는지가 기준이다.

선지별 해설

① X
(가)를 갱신 분실로 본 것은 맞지만, (나)를 모순성으로 본 것이 틀리다. (나)는 rollback된 T1의 미확정 값을 T2가 읽었으므로 연쇄 복귀이다. 틀린 표현은 '(나) 모순성'이며, 올바르게는 '(나) 연쇄 복귀'이다.

② X
(가)를 모순성으로 본 것이 틀리다. (가)는 두 트랜잭션이 같은 x를 각각 갱신하고 마지막 write가 먼저의 갱신을 덮어쓰므로 갱신 분실이다. (나)를 연쇄 복귀로 본 것은 맞다. 틀린 표현은 '(가) 모순성'이며, 올바르게는 '(가) 갱신 분실'이다.

③ O
(가)는 T1의 x+300 결과가 write된 뒤 T2의 write(x)가 이를 덮어써 T1의 갱신 효과가 사라지므로 갱신 분실이다. (나)는 T2가 T1의 write(x) 결과를 읽은 뒤 T1이 rollback되므로 T2도 함께 취소될 수 있어 연쇄 복귀이다.

④ X
(가)를 연쇄 복귀로 본 것이 틀리고, (나)를 갱신 분실로 본 것도 틀리다. (가)에는 rollback 연쇄가 없고 단순히 마지막 쓰기가 앞선 갱신을 덮어쓴다. (나)는 미확정 데이터 읽기로 인한 연쇄 복귀이다. 틀린 표현은 '(가) 연쇄 복귀, (나) 갱신 분실'이며, 올바르게는 '(가) 갱신 분실, (나) 연쇄 복귀'이다.

오답 포인트

(나)에서 T2도 write(x)를 했다는 이유만으로 갱신 분실로 오인하기 쉽다. 그러나 이 사례의 핵심은 마지막 쓰기 자체보다 T2가 commit 전의 T1 값을 읽었다는 점이며, rollback T1이 등장하는 순간 연쇄 복귀 가능성을 먼저 떠올려야 한다.

NoteBase

다시 나오면 이렇게 판단

동시 공용 이상 현상은 먼저 같은 항목을 둘 다 갱신하는지, 다음으로 한 트랜잭션이 다른 트랜잭션의 미확정 값을 읽는지를 본다. 마지막 write가 앞선 결과를 덮어쓰면 갱신 분실이고, 미확정 값을 읽은 뒤 원 트랜잭션이 rollback되면 연쇄 복귀이다.

암기 포인트

덮어쓰면 갱신 분실, 미확정값을 읽고 따라 취소되면 연쇄 복귀이다.

NoteBase

문 19. OLAP(online analytical processing)시스템이 갖추어야 할 요건으로 가장 옳지 않은 것은?

① 사용자는 OLAP 데이터에 대해 다차원 뷰를 가질 수 있어야 한다.

② 사용자는 차원의 수에 관계없이 같은 연산을 실행시킬 수 있어야 한다.

③ 주제 지향적이고, 통합적이고, 비소멸성이며, 시간에 따라 변화하는 데이터베이스로서 의사결정 지원을 제공해야 한다.

④ 차원의 수가 증가하더라도 검색 성능이 저하되어서는 안 된다.

정답 3번

출제 포인트

이 문제는 OLAP의 기능적 요구사항과 데이터 웨어하우스의 정의를 구별하는 문제이다.

핵심 개념

OLAP는 의사결정 지원을 위해 다차원 데이터에 대해 빠르고 일관된 분석 연산을 제공하는 처리 시스템이다.
데이터 웨어하우스는 주제 지향적, 통합적, 비소멸적, 시변적인 데이터를 저장하는 의사결정 지원용 데이터베이스이다.
즉 데이터 웨어하우스는 분석 대상 데이터를 저장하는 저장소이고, OLAP는 그 데이터를 다차원으로 분석하는 도구라는 점이 핵심 차이이다.

NoteBase

선지별 해설

① O
사용자가 OLAP 데이터에 대해 다차원 뷰를 가질 수 있어야 한다는 설명은 맞다. OLAP의 본질이 차원별로 데이터를 자르고, 묶고, 회전하며 보는 다차원 분석에 있기 때문이다.

② O
차원의 수가 달라져도 같은 분석 연산을 수행할 수 있어야 한다는 설명은 맞다. OLAP는 특정 차원 개수에 종속되지 않고 drill-down, roll-up, slice, dice 같은 연산을 일반적으로 지원해야 한다.

③ X
틀린 부분은 "주제 지향적이고, 통합적이고, 비소멸성이며, 시간에 따라 변화하는 데이터베이스"라는 규정으로 OLAP 시스템 자체를 설명한 점이다. 이는 OLAP 요구사항이 아니라 데이터 웨어하우스의 정의이다. 올바르게는 OLAP는 이러한 데이터 웨어하우스의 데이터를 기반으로 다차원 분석과 의사결정 지원 기능을 제공하는 시스템이라고 보아야 한다.

④ O
차원의 수가 증가하더라도 검색 성능이 급격히 저하되지 않아야 한다는 설명은 맞다. OLAP는 대용량 다차원 분석을 전제로 하므로 차원 수나 데이터 크기가 늘어나도 일관된 응답 성능을 유지하는 것이 요구된다.

오답 포인트

③은 의사결정 지원이라는 공통 표현 때문에 OLAP와 데이터 웨어하우스를 같은 개념으로 섞어 놓은 선지이다. 저장소의 성격을 묻는 문장인지, 분석 시스템의 기능을 묻는 문장인지를 구분하지 못하면 ③을 맞는 말처럼 보기 쉽다.

다시 나오면 이렇게 판단

선지에 주제 지향성, 통합성, 비소멸성, 시변성이 나오면 먼저 데이터 웨어하우스의 4대 특성인지 확인한다. 반대로 다차원 뷰, 차원 간 연산, 성능 유지가 나오면 OLAP의 기능 요구사항으로 판단할 수 있다.

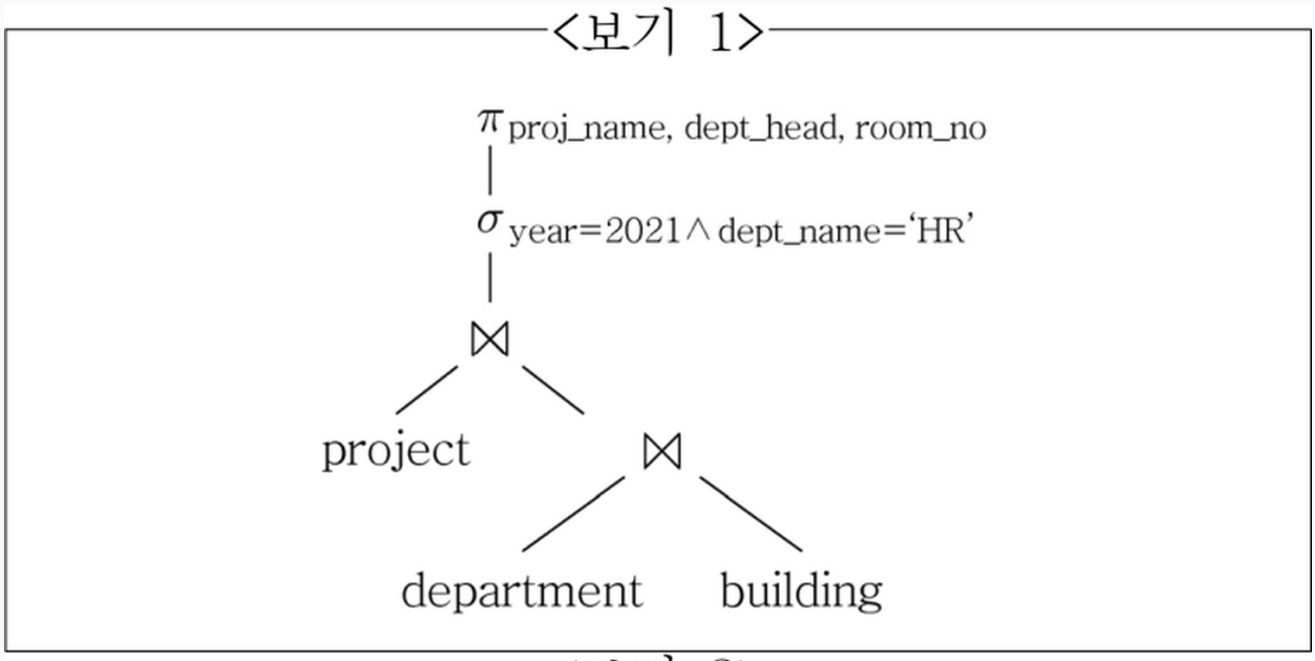
암기 포인트

데이터 웨어하우스는 저장, OLAP는 분석이다.

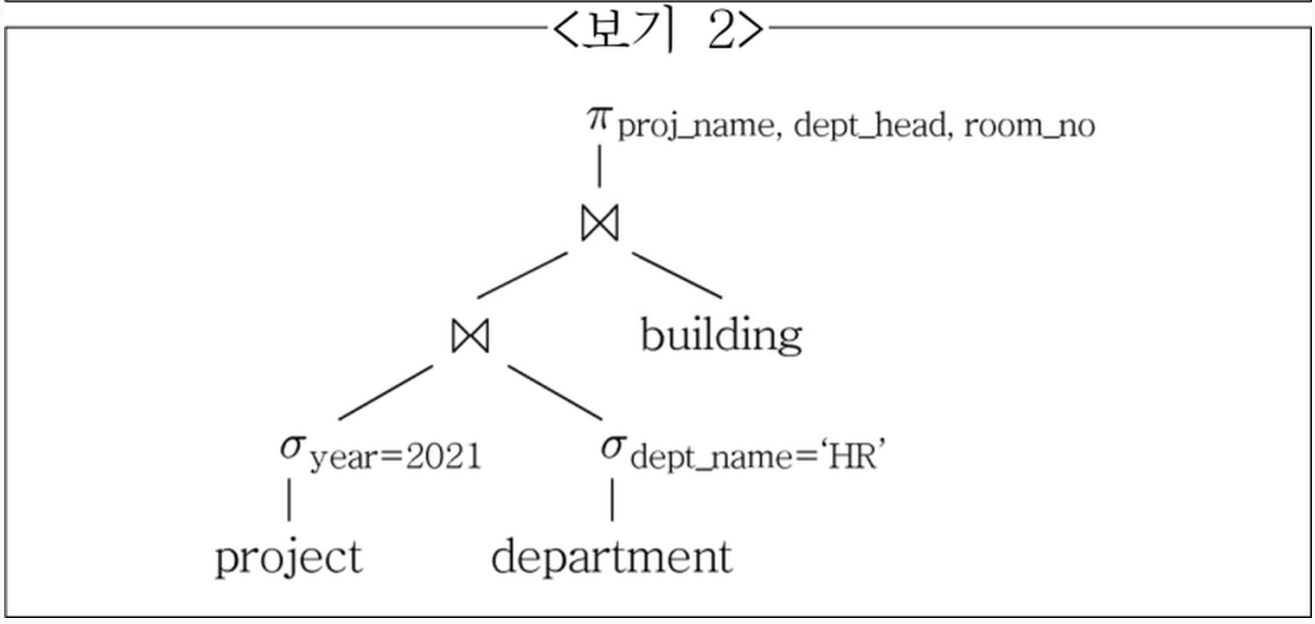
NoteBase

문 20. <보기 1>과 <보기 2>는 데이터베이스 시스템 내에서 질의를 수행할 때 사용되는 수행계획(query evaluation plan)을 나타낸 것이다. 가장 옳지 않은 설명은?

<보기 1>



<보기 2>



① <보기 1>과 <보기 2>의 수행계획은 같은 결과를 출력한다.

② <보기 1>의 수행계획은 조인(join) 연산을 먼저 수행한 다음 선택(selection) 연산을 수행하고, <보기 2>의 수행계획은 선택 연산을 먼저 수행하고 조인 연산을 수행한다.

③ <보기 1>의 수행계획은 테이블에 색인이 있을 경우 조인 과정에서 이를 활용할 수 있다.

④ <보기 2>의 수행계획은 선택 연산의 조건을 만족하는 레코드들로 중간결과 테이블을 만든 다음 조인을 수행해야 하기 때문에 일반적으로 <보기 1>의 수행계획에 비해서 성능저하가 크다.

NoteBase

정답 4번

출제 포인트

이 문제는 질의 수행계획 트리에서 선택 연산을 조인 전에 미리 밀어 넣는 선택 푸시다운과 조인 순서가 중간 결과 크기와 성능에 어떤 영향을 주는지를 판단하는 문제이다.

선지별 해설

① O
<보기 1>은 project와 department, building을 먼저 조인한 뒤에 year=2021, dept_name='HR' 조건을 적용하고, <보기 2>는 project와 department에 각각 선택 조건을 먼저 적용한 뒤 조인한다. 선택 조건이 각 릴레이션의 속성에만 걸려 있으므로 결과 집합은 관계대수의 선택 푸시다운 규칙에 따라 동일하다.

② O
<보기 1>의 트리에서는 조인 연산 두 번이 먼저 수행되고 그 위에서 선택 연산이 수행된다. 반대로 <보기 2>의 트리에서는 project에 year=2021, department에 dept_name='HR' 선택을 먼저 적용한 뒤 그 결과를 조인하므로 설명이 맞다.

③ O
<보기 1>은 조인을 먼저 수행하는 계획이므로 조인 시 사용할 접근 경로가 중요하다. 조인 속성에 인덱스가 있으면 인덱스 중첩 루프 조인처럼 인덱스를 활용하는 조인 방식이 가능하므로 이 설명은 성립한다.

④ X
틀린 부분은 '일반적으로 <보기 1>의 수행계획에 비해서 성능저하가 크다'이다. <보기 2>는 선택 조건을 먼저 적용해 project와 department의 후보 레코드 수를 줄인 뒤 조인하므로 중간 결과가 작아져 일반적으로 <보기 1>보다 효율적이다. 올바르게는 '선택 연산을 먼저 수행하는 <보기 2>가 일반적으로 <보기 1>보다 성능상 유리하다'이다.

오답 포인트

④는 '중간결과 테이블을 만든다'는 표현 때문에 비효율적으로 보이도록 유도한 선지이다. 그러나 질의 최적화에서는 중간 결과를 무조건 만든다는 표현보다 조인 전에 튜플 수를 줄였는지가 핵심이며, 선택 푸시다운은 보통 조인 비용을 줄인다.

다시 나오면 이렇게 판단

수행계획 트리가 나오면 먼저 선택 조건이 어느 릴레이션의 속성인지 확인하고, 그 선택이 조인 전에 내려갈 수 있으면 중간 결과가 줄어드는 방향인지 판단한다. 그다음 조인을 먼저 하는 계획보다 선택을 먼저 하는 계획이 보통 유리한지 비교하면 된다.

NoteBase

암기 포인트

선택은 가능한 한 아래로 내리고 조인은 줄어든 결과로 수행할수록 일반적으로 유리하다.

NoteBase