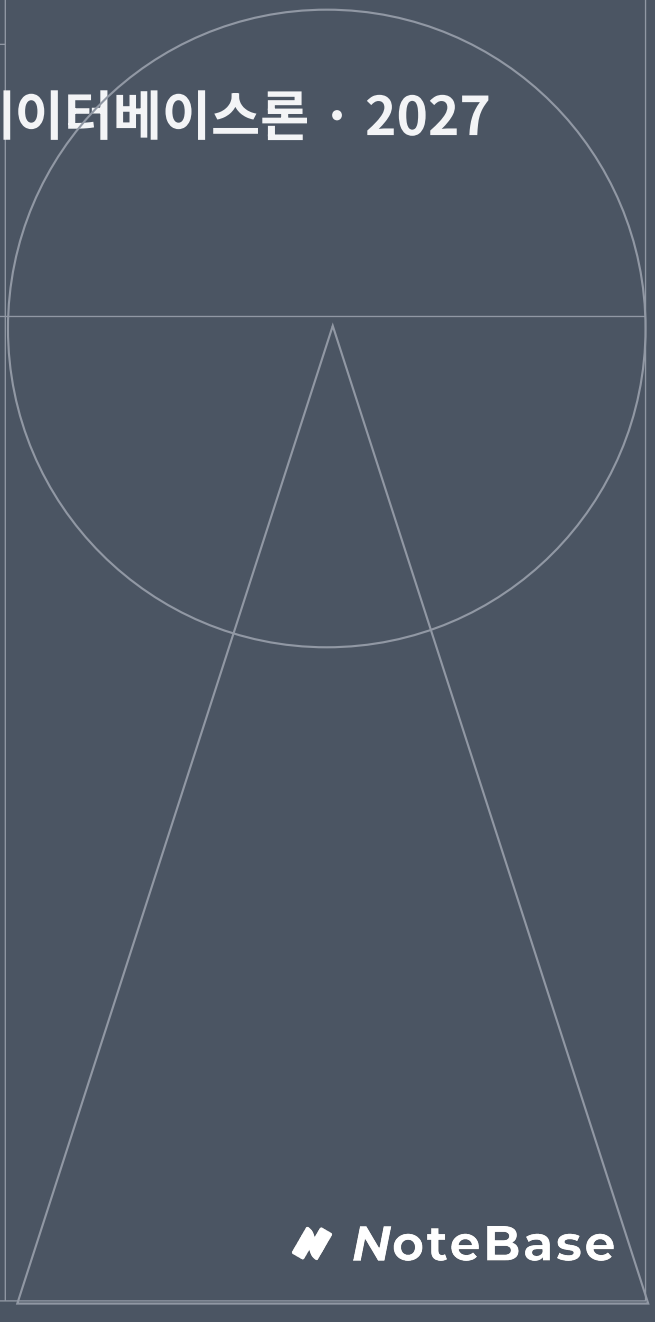
기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2027



NoteBase

기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2027

공무원/경찰/공채(경위) · 데이터베이스론 · 2027

문 1. 다음 상황에 대한 설명으로 가장 적절한 것은?

어느 공공기관의 시스템에서 성능향상을 위해 특정 테이블에 B+트리 인덱스를 추가하였다. 각종 조회, 민원 신청, 회계 관리 프로그램의 SQL 문장은 그대로 유지되었고, 사용자가 보는 속성 구조도 바뀌지 않았다.

① 외부 스키마가 변경되었으므로 논리적 데이터 독립성이 깨졌다.

② 내부 스키마의 접근 경로만 바뀌었으므로 물리적 데이터 독립성이 유지된 사례이다.

③ 개념 스키마가 변경되었으므로 모든 외부 스키마를 다시 정의해야 한다.

④ 저장 구조 변경은 데이터 독립성과 무관하며 DBMS의 보안 기능에 해당한다.

정답 2번

간단한 해설

인덱스 추가는 **내부 스키마의 접근 경로 변경**이다. SQL과 사용자에게 보이는 구조가 유지되었으므로 **물리적 데이터 독립성**이 보장된 사례이다.

NoteBase

핵심 풀이

데이터베이스의 3단계 스키마에서 외부 스키마는 사용자별로 보이는 데이터 구조, 개념 스키마는 데이터베이스 전체의 논리적 구조, 내부 스키마는 저장 방식과 접근 경로를 표현한다.

B+트리 인덱스를 추가한 것은 데이터를 찾는 접근 경로, 즉 **내부 스키마의 물리적 구현을 변경**한 것이다.
테이블의 논리적 속성과 사용자가 보는 구조를 바꾼 것은 아니다.

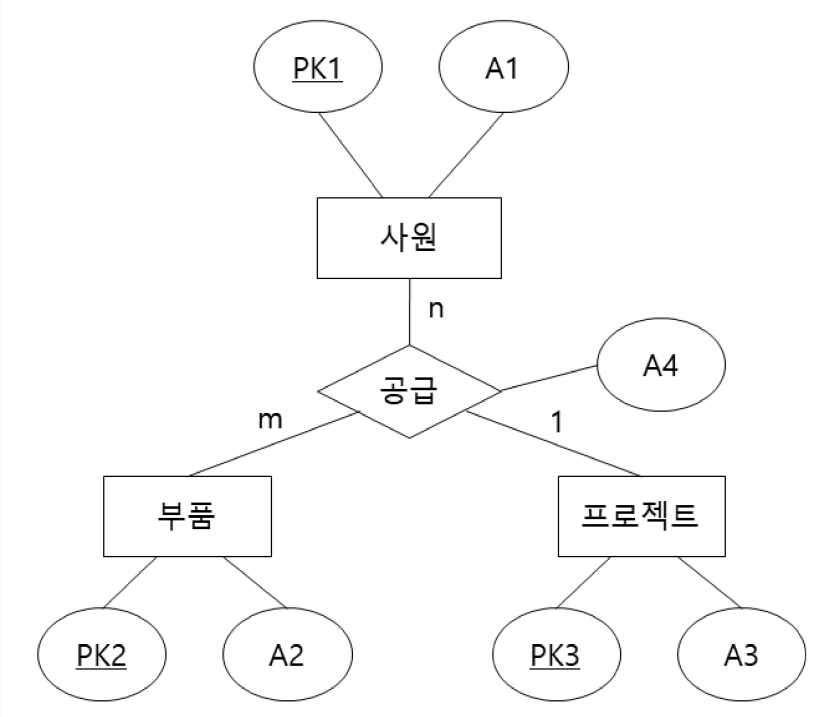
물리적 데이터 독립성은 내부 스키마를 바꾸어도 개념 스키마와 응용 프로그램을 바꾸지 않을 수 있는 성질이다.
이 사례에서는 **인덱스 추가 후에도 SQL과 사용자 관점의 구조가 그대로**이므로 그 성질이 유지되었다.

논리적 데이터 독립성은 개념 스키마의 변경이 외부 스키마와 응용 프로그램에 영향을 주지 않도록 하는 성질이다. 사례에는 외부·개념 스키마가 변경되었다는 조건이 없으므로 ①과 ③의 전제가 맞지 않는다. 또한 저장 구조와 접근 경로를 바꾸는 것은 보안 기능이 아니라 데이터 독립성과 관련되므로 ④도 맞지 않는다.

따라서 내부 스키마의 변경에도 응용 프로그램이 유지된 물리적 데이터 독립성의 사례로 설명한 ②가 가장 적절하다.

NoteBase

문 2. 다음 ERD를 관계형 데이터베이스 스키마로 변환한 것으로 가장 적절한 것은? (단, 밑줄은 기본키이다.)



① 사원(PK1, A1), 부품(PK2, A2), 프로젝트(PK3, A3), 공급(PK1, PK2, PK3, A4)

② 사원(PK1, A1), 부품(PK2, A2), 프로젝트(PK3, A3), 공급(PK1, PK2, PK3, A4)

③ 사원(PK1, A1, PK3), 부품(PK2, A2, PK3), 프로젝트(PK3, A3), 공급(PK1, PK2, A4)

④ 사원(PK1, A1, PK3), 부품(PK2, A2, PK3), 프로젝트(PK3, A3), 공급(PK1, PK2, PK3, A4)

정답 1번

간단한 해설

삼진 관계에서 프로젝트 쪽의 1은 사원·부품의 조합마다 프로젝트가 하나로 정해짐을 뜻한다. 따라서 공급 릴레이션의 기본키는 **(PK1, PK2)**이고, PK3는 프로젝트를 참조하는 외래키로 포함한다.

NoteBase

도식 해설

개체 변환: 사원·부품·프로젝트는 각각 독립된 개체이므로 사원(PK1, A1), 부품(PK2, A2), 프로젝트(PK3, A3) 릴레이션으로 만든다. 그림에서 밑줄 친 PK1, PK2, PK3가 각 개체 릴레이션의 기본키가 된다.

공급 관계 변환: 공급은 사원·부품·프로젝트가 함께 참여하는 **삼진 관계**이므로 별도의 공급(PK1, PK2, PK3, A4) 릴레이션으로 나타낸다. PK1·PK2·PK3는 각각 참여 개체를 참조하는 외래키이고, A4는 공급 관계 자체의 속성이므로 공급 릴레이션에 둔다.

기본키 결정: 프로젝트 쪽의 1은 **사원과 부품의 조합이 정해지면 프로젝트가 하나로 정해진다**는 제약이다. 따라서 공급에서는 PK1과 PK2의 조합으로 PK3와 관계 속성 A4가 결정되며, 기본키는 (PK1, PK2)이다. PK3까지 포함한 세 속성은 유일성을 갖더라도 이 제약하에서는 최소성이 없으므로 기본키로 선택할 필요가 없다.

다른 변환과의 차이: 프로젝트 쪽의 1은 사원 하나 또는 부품 하나가 각각 프로젝트 하나만 가진다는 뜻이 아니다. 사원 릴레이션과 부품 릴레이션에 PK3를 넣는 방식은 삼진 관계의 결합 제약을 올바르게 나타내지 못한다. 또한 공급에서 PK3를 생략하면 어느 프로젝트에 공급하는지 보존할 수 없다.

따라서 세 개체 릴레이션은 본래의 속성만 유지하고, 공급(PK1, PK2, PK3, A4)에서 PK1·PK2에만 기본키 밑줄이 있는 ①이 적절하다.

NoteBase

문 3. 다음 고객 릴레이션에 대해 고객번호를 기준으로 나누었을 때, 해당되는 단편화와 위반된 정확성 규칙으로 가장 적절한 것은?

<고객>

고객번호	지역	이름
100	서울	홍길동
400	부산	강감찬
600	대전	이순신

F1: 고객번호 ≤ 500

고객번호	지역	이름
100	서울	홍길동
400	부산	강감찬

F2: 고객번호 ≥ 300

고객번호	지역	이름
400	부산	강감찬
600	대전	이순신

① 수직 단편화, 분리성

② 수직 단편화, 회복성

③ 수평 단편화, 분리성

④ 수평 단편화, 회복성

정답 3번

간단한 해설

같은 속성들을 유지한 채 고객번호 조건으로 행을 나누었으므로 **수평 단편화**이다. 400번 고객이 두 단편에 중복 포함되어 **분리성**을 위반한다.

NoteBase

핵심 풀이

단편화의 방향은 조건에 사용한 속성의 이름이 아니라 무엇을 나누었는지로 구분한다. 두 단편 모두 고객번호·지역·이름이라는 같은 속성을 유지하고, 고객번호 조건으로 **행인 튜플을 나누었으므로 수평 단편화**이다. 수직 단편화는 속성인 열을 나누는 방식이다.

그림에서 F1은 고객번호가 500 이하인 100번과 400번 고객을, F2는 300 이상인 400번과 600번 고객을 포함한다. 따라서 **400번 고객의 동일한 튜플이 두 단편에 중복**되어 있다.

수평 단편화의 분리성은 각 튜플이 서로 다른 단편에 중복 포함되지 않아야 한다는 조건이다. 400번 고객이 두 곳에 있으므로 분리성이 위반되었다.

완전성은 원래의 모든 데이터가 단편들에 포함되어야 한다는 조건이고, 회복성은 단편들로 원래 릴레이션을 재구성할 수 있어야 한다는 조건이다. 원래의 100·400·600번 고객이 모두 남아 있고, 두 단편의 집합 합집합으로 중복을 제거하면 원래 고객 릴레이션을 얻으므로 회복성이 위반된 것은 아니다.

따라서 단편화 방식은 수평 단편화이고 위반된 규칙은 분리성이므로 ③에 해당한다.

NoteBase

문 4. 다음 사례에 해당하는 EER(Enhanced Entity Relationship) 제약 조건으로 모두 옳은 것은?

차량은 승용차와 화물차로 세분화된다. 모든 차량은 반드시 둘 중 하나에 속하고, 한 차량이 동시에 승용차와 화물차가 될 수 없다. 승용차에는 좌석 수, 화물차에는 적재중량 속성이 있다.

① 부분 구체화(partial specialization), 배타 구체화(disjoint specialization)

② 부분 구체화(partial specialization), 중첩 구체화(overlapping specialization)

③ 전체 구체화(total specialization), 배타 구체화(disjoint specialization)

④ 전체 구체화(total specialization), 중첩 구체화(overlapping specialization)

정답 3번

간단한 해설

모든 차량이 반드시 어느 하위 유형에 속하므로 **전체 구체화**이고, 두 유형에 동시에 속할 수 없으므로 **배타 구체화**이다. 두 조건이 함께 적용되어 각 차량은 정확히 한 하위 유형에 속한다.

구성 요소별 해설

참여 범위: 모든 차량이 승용차 또는 화물차에 반드시 속하므로 **전체 구체화**이다. 상위 개체 집합의 모든 개체가 적어도 하나의 하위 개체 집합에 포함되어야 한다는 제약이다. 부분 구체화라면 어느 하위 집합에도 속하지 않는 차량이 허용되지만, 사례에서는 이를 허용하지 않는다.

하위 집합 간 중복: 한 차량이 승용차와 화물차에 동시에 속할 수 없으므로 **배타 구체화**이다. 서로 다른 하위 개체 집합의 중복 소속을 금지하는 제약이다. 중첩 구체화라면 동일 차량이 두 하위 집합에 함께 속하는 것이 가능하다.

하위 개체의 속성: 좌석 수는 승용차의 고유 속성이고, 적재중량은 화물차의 고유 속성이다. 이러한 속성은 하위 개체로 세분화한 이유를 보여 주지만, 전체·부분 여부와 배타·중첩 여부는 각각 모든 차량의 참여 의무와 동시 소속 허용 여부로 판단한다.

전체 구체화와 배타 구체화를 함께 적용하면 모든 차량은 두 하위 집합 중 정확히 하나에 속하게 된다. 두 제약을 모두 나타낸 선택지는 ③이다.

NoteBase

문 5. 관계대수 연산자에 대한 설명으로 가장 적절하지 않은 것은?

① 디비전(division)과 세미조인은 교환법칙이 성립하지 않는다.

② 동등조인과 자연조인은 교환법칙과 결합법칙이 성립한다.

③ 카티션 프로덕트(cartesian product)는 릴레이션이 합병 가능(union compatible)하지 않아도 연산이 가능하다.

④ 복합연산인 디비전은 기본 연산인 실렉트(select), 차집합, 카티션 프로덕트로 대체할 수 있다.

정답 4번

간단한 해설

디비전을 기본 연산으로 표현할 때 필요한 것은 **실렉트가 아니라 프로젝트**이다. 후보 속성을 추출하고, 카티션 프로덕트와 차집합으로 모든 조건을 충족하지 못하는 후보를 제거한다.

선지별 해설

① O: 디비전은 나누어지는 릴레이션과 나누는 릴레이션의 역할이 다르므로 순서를 바꾸면 의미와 결과 구조가 달라지거나 연산 자체가 정의되지 않는다. 세미조인도 조인에 참여하는 왼쪽 릴레이션의 튜플과 속성만 반환하므로 **어느 쪽을 왼쪽에 두는지**에 따라 결과가 달라진다.

② O: 동등조인은 같은 값을 갖는 속성들을 연결하고, 자연조인은 공통 속성의 동등 조건으로 연결한 뒤 중복 속성을 제거한다. 관계대수에서 속성 이름을 일관되게 대응시키고 같은 조인 조건을 유지하면, 두 릴레이션의 순서를 바꾸거나 여러 조인의 묶는 순서를 바꾸어도 결과를 같게 할 수 있다. 이는 외부조인이 아닌 일반적인 내부조인을 전제로 한다.

③ O: 카티션 프로덕트는 두 릴레이션의 모든 튜플 쌍을 결합한다. 속성 수와 대응 도메인이 같아야 하는 **합병 가능 조건을 요구하지 않는다**. 속성 이름이 충돌하면 구별할 수 있도록 이름을 바꾸어 사용한다.

④ X: 디비전은 **프로젝트, 차집합, 카티션 프로덕트**로 표현할 수 있다. R(X, Y)를 S(Y)로 나눌 때 먼저 R에서 X값들의 후보를 프로젝트하고, 후보와 S의 모든 조합 중 R에 없는 조합을 차집합으로 찾는다. 그 누락 조합에서 X값을 프로젝트하여 후보에서 빼면, S의 모든 Y값과 연결된 X값만 남는다. 실렉트는 행을 고르는 연산이므로 필요한 속성을 추출하는 프로젝트를 대신할 수 없다.

NoteBase

문 6. 릴레이션 R과 S에 대한 $R \bowtie_N S$와 동등한 표현으로 적절한 것은 모두 몇 개인가? (단, $\bowtie_N$은 자연조인, $\ltimes$은 세미조인, $\boxtimes$은 완전외부조인 연산이다.)

가. $(R \ltimes S) \bowtie_N S$
나. $(S \ltimes R) \bowtie_N R$
다. $R \bowtie_N (S \ltimes R)$
라. R $\boxtimes$ S

① 1
② 2
③ 3
④ 4

정답 3번

간단한 해설

세미조인은 상대 릴레이션과 연결되는 튜플을 골라 **왼쪽 릴레이션의 속성만** 남긴다. 이 결과를 상대 릴레이션과 다시 자연조인하면 원래 자연조인의 결과가 복원되지만, 완전외부조인은 **연결되지 않는 튜플까지 보존**하므로 일반적으로 동등하지 않다.

NoteBase

보기별 해설

자연조인은 두 릴레이션의 모든 공통 속성값이 같은 튜플을 결합하며, 결과에는 양쪽 속성이 포함되고 공통 속성은 한 번만 나타난다. 반면 $R \ltimes S$는 S에 대응하는 튜플이 있는 R의 튜플만 골라 **R의 속성만 유지**한다. 따라서 세미조인 자체가 자연조인과 같은 것은 아니며, 각 보기에서는 바깥쪽 자연조인까지 수행한 결과를 비교해야 한다.

가. O: $R \ltimes S$에서 제거되는 것은 S와 결합할 수 없는 R의 튜플뿐이다. 이 튜플들은 원래의 자연조인에도 참여하지 않는다. 남은 R의 튜플을 S와 자연조인하면 대응하는 S의 튜플이 모두 결합되고 S의 속성도 포함되므로, $R \bowtie_N S$와 같은 결과가 된다.

나. O: $S \ltimes R$는 R에 대응하는 튜플이 있는 S의 튜플만 남기며, 속성은 S의 속성만 유지한다. 이를 R과 자연조인하면 원래 S와 R 사이에서 가능했던 모든 결합이 만들어진다. 자연조인은 **피연산자의 순서를 바꾸어도 같은 릴레이션**을 얻으므로, 이 결과는 $R \bowtie_N S$와 같다.

다. O: 안쪽의 $S \ltimes R$에서 빠지는 S의 튜플은 R의 어떤 튜플과도 결합하지 못하는 튜플이다. 따라서 R을 이 세미조인 결과와 자연조인해도 원래 자연조인에 필요한 S의 튜플은 모두 남아 있다. R과 S의 속성을 결합한 최종 결과는 $R \bowtie_N S$와 같다.

라. X: 완전외부조인은 결합되는 튜플뿐 아니라 **어느 쪽에서도 대응하지 못한 튜플까지** 결과에 남기고, 상대 릴레이션의 속성값이 없는 자리는 NULL로 채운다. 예를 들어 공통 속성 B에 대해 R에는 B = 1인 튜플만, S에는 B = 2인 튜플만 있다면 자연조인 결과는 공집합이지만 완전외부조인에는 양쪽 튜플이 남는다. 따라서 일반적으로 동등하지 않으며, 두 결과가 같으려면 양쪽 모두에 대응하지 못한 튜플이 없어야 한다.

동등한 표현은 가·나·다의 3개이므로, 3개를 나타내는 ③에 해당한다.

NoteBase

문 7. 릴레이션 R(A, B, C)의 유일한 후보키가 {A}일 때, R이 가질 수 있는 슈퍼키(superkey)의 총 개수는?

① 3

② 4

③ 7

④ 8

정답 2번

간단한 해설

유일한 후보키가 {A}이므로 슈퍼키에는 **A가 반드시 포함**된다. 나머지 B와 C를 각각 넣거나 빼는 경우는 2 × 2이므로 총 **4개**이다.

계산 과정

슈퍼키는 릴레이션의 튜플을 유일하게 식별할 수 있는 속성 집합이다. 후보키는 이 중 불필요한 속성을 제거할 수 없는 최소의 슈퍼키이므로, **후보키를 포함하는 모든 속성 집합은 슈퍼키**이다.

유일한 후보키가 {A}이므로 모든 슈퍼키는 A를 포함해야 한다. 만약 A가 없는 슈퍼키가 존재한다면, 그 안에서 불필요한 속성을 제거하여 A를 포함하지 않는 또 다른 후보키를 얻을 수 있어 유일하다는 조건에 모순된다.

A를 반드시 넣은 뒤 B와 C는 각각 포함하거나 포함하지 않을 수 있다. 따라서 가능한 경우의 수는 **2 × 2 = 4**이다.

실제로 가능한 슈퍼키는 {A}, {A, B}, {A, C}, {A, B, C}의 네 개이다. 따라서 총 개수는 4이며 선택지 ②에 해당한다.

NoteBase

문 8. 참조 무결성(referential integrity) 제약조건에 대한 설명으로 가장 적절하지 않은 것은?

① NOT NULL이 아닌 외래키 컬럼은 NULL 값을 가질 수 있다.

② ON DELETE CASCADE는 부모 행 삭제 시 이를 참조하는 자식 행도 삭제한다.

③ 외래키는 반드시 기본키만 참조할 수 있고 UNIQUE 키는 참조할 수 없다.

④ 외래키는 같은 테이블의 키를 참조하는 자기참조 구조로도 정의할 수 있다.

정답 3번

간단한 해설

외래키는 기본키뿐 아니라 **UNIQUE 제약으로 유일성이 보장된 키**도 참조할 수 있다. 참조 대상의 유일성이 중요하므로 기본키만 참조할 수 있다는 제한은 잘못이다.

선지별 해설

① O: 외래키 제약은 참조 대상과의 값 일치를 검사하지만, 그 자체가 NULL을 금지하지는 않는다. 외래키 컬럼에 NOT NULL 등 NULL을 막는 별도 제약이 없다면 **NULL 값을 가질 수 있다**. 참조 관계가 아직 없거나 정해지지 않은 경우를 표현할 수 있다.

② O: ON DELETE CASCADE는 부모 테이블의 참조 대상 행을 삭제할 때 **그 행을 참조하는 자식 행도 함께 삭제**하는 연쇄 삭제 동작이다. 따라서 부모가 사라진 뒤 그 부모를 가리키는 자식 행이 남는 것을 방지한다.

③ X: 외래키가 참조할 수 있는 대상은 기본키에만 한정되지 않는다. **UNIQUE 제약으로 유일성이 보장되는 키도 참조할 수 있다**. 참조 값으로 대상 행을 식별할 수 있어야 한다는 것이 핵심이며, 기본키라는 명칭만이 허용 조건은 아니다.

④ O: 외래키의 참조 대상은 다른 테이블일 수도 있고 같은 테이블일 수도 있다. 예를 들어 사원 테이블의 상사번호가 같은 테이블의 사원번호를 참조하면 **자기참조 외래키**가 된다. 하나의 테이블 안에서 사원과 상사의 관계를 표현하는 구조이다.

NoteBase

문 9. SQL의 분류에 대한 설명으로 가장 적절한 것은?

① CREATE, DROP은 데이터 조작어(DML)에 속한다.

② SELECT, INSERT, UPDATE, DELETE는 데이터 정의어(DDL)에 속한다.

③ COMMIT, ROLLBACK은 데이터 정의어(DDL)에 속한다.

④ GRANT, REVOKE는 데이터 제어어(DCL)에 속한다.

정답 4번

간단한 해설

GRANT와 REVOKE는 데이터베이스의 **사용 권한을 부여하거나 회수**하는 명령이다. 따라서 데이터 제어어(DCL)로 분류한다.

선지별 해설

① X: CREATE와 DROP은 테이블 등의 데이터베이스 객체를 생성하거나 제거하는 명령으로 **데이터 정의어(DDL)**에 속한다. 객체에 저장된 개별 행을 조회·변경하는 데이터 조작어와 구별된다.

② X: SELECT는 데이터 조회, INSERT는 행 삽입, UPDATE는 값 수정, DELETE는 행 삭제를 수행하므로 일반적인 분류에서는 **데이터 조작어(DML)**에 속한다. SELECT를 데이터 질의어(DQL)로 별도 구분하기도 하지만 데이터 정의어로 분류하지는 않는다.

③ X: COMMIT은 트랜잭션의 변경을 확정하고, ROLLBACK은 확정되지 않은 변경을 취소한다. 이는 **트랜잭션 제어**에 관한 명령이며 스키마를 정의하는 DDL이 아니다. 트랜잭션 제어어(TCL)로 별도 분류하거나 넓은 제어어 범주에 포함하여 설명한다.

④ O: GRANT는 사용자나 역할에 권한을 부여하고, REVOKE는 부여된 권한을 회수한다. 데이터베이스에 대한 접근·사용 권한을 제어하므로 **데이터 제어어(DCL)**에 속한다.

NoteBase

문 10. 다음과 같은 순서로 작업이 수행될 때 시스템 카탈로그에 저장되는 정보로만 묶인 것으로 가장 적절한 것은?

```
CREATE TABLE 계좌(계좌번호 CHAR(12) PRIMARY KEY, 잔액 INTEGER);
CREATE INDEX IDX_계좌_잔액 ON 계좌(잔액);
INSERT INTO 계좌 VALUES('A001', 10000);
```

① 계좌번호 CHAR(12), IDX_계좌_잔액, 기본키 제약조건

② 계좌번호 A001, 잔액 10000, 기본키 제약조건

③ 계좌번호 A001, 인덱스 물리 블록 주소, 사용자 화면 배치 좌표

④ INSERT 수행 시각, 잔액 10000, 화면 버튼 이름

정답 1번

간단한 해설

시스템 카탈로그에는 **속성의 자료형, 인덱스 정의, 기본키 제약조건** 같은 메타데이터가 저장된다. A001과 10000은 카탈로그 정보가 아니라 계좌 테이블에 삽입되는 실제 데이터이다.

핵심 풀이

시스템 카탈로그는 DBMS가 관리하는 데이터 사전으로, **데이터베이스 객체의 구조와 제약조건에 관한 메타데이터**를 저장한다. 실제 업무 데이터의 값과 데이터의 구조를 설명하는 정보를 구별해야 한다.

CREATE TABLE을 수행하면 계좌 테이블의 속성 이름, 계좌번호의 자료형 CHAR(12), 잔액의 자료형 INTEGER, 계좌번호에 설정된 기본키 제약조건 등이 카탈로그에 기록된다.

CREATE INDEX를 수행하면 IDX_계좌_잔액이라는 인덱스의 존재와 그 대상 테이블·속성인 계좌의 잔액에 관한 정보가 등록된다.

INSERT로 추가한 **A001과 10000은 계좌 테이블에 저장되는 실제 튜플의 값**이다. 따라서 이 값들을 포함하는 ②·③·④는 카탈로그 정보만의 묶음이 아니다. 사용자 화면 배치 좌표나 버튼 이름도 일반적인 DBMS 시스템 카탈로그의 스키마 정보가 아니다.

①의 계좌번호 CHAR(12), IDX_계좌_잔액, 기본키 제약조건은 모두 구조·객체·제약조건에 관한 메타데이터이므로 가장 적절하다.

NoteBase

문 11. 판매 테이블에 대한 다음 질의문의 결과에서 상품 A의 COUNT(*)와 COUNT(수량)의 값으로 가장 적절한 것은?

<판매>

상품	지역	수량
A	서울	10
A	부산	NULL
B	서울	5

```
SELECT 상품, COUNT(*), COUNT(수량)
FROM 판매
GROUP BY 상품;
```

① COUNT(*): 1, COUNT(수량): 1

② COUNT(*): 1, COUNT(수량): 2

③ COUNT(*): 2, COUNT(수량): 1

④ COUNT(*): 2, COUNT(수량): 2

정답 3번

간단한 해설

상품 A의 행은 두 개지만 수량 값이 있는 행은 하나뿐이다. COUNT(*)는 **NULL 여부와 관계없이 행 수**를 세고, COUNT(수량)은 NULL을 제외하므로 각각 2와 1이 된다.

NoteBase

계산 과정

상품별 그룹 구성: GROUP BY 상품은 상품 값이 같은 행끼리 묶는다. 상품 A 그룹에는 (A, 서울, 10)과 (A, 부산, NULL)의 두 행이 들어가며, 상품 B의 행은 별도 그룹이므로 A의 집계에 포함되지 않는다.

COUNT(*) 계산: COUNT(*)는 그룹에 속한 **모든 행의 수**를 센다. 수량이 NULL인 부산 행도 하나의 행으로 존재하므로 A 그룹의 COUNT(*)는 2이다.

COUNT(수량) 계산: COUNT(수량)은 해당 컬럼 값이 **NULL이 아닌 행의 수**만 센다. A 그룹에서는 서울 행의 수량 10만 집계 대상이고 부산 행의 NULL은 제외되므로 결과는 1이다. 수량을 합산하는 것이 아니므로 값 10 자체가 결과가 되지는 않는다.

따라서 상품 A의 결과는 COUNT(*) = 2, COUNT(수량) = 1이며 ③과 일치한다.

NoteBase

문 12. DEPT, EMP 테이블에 대한 다음 질의문의 결과로 가장 적절한 것은?

<DEPT>

did
10
20
30

<EMP>

eid	did
1	10
2	10
3	20
4	NULL

```
SELECT D.did, COUNT(E.eid) AS cnt
FROM DEPT AS D
LEFT OUTER JOIN EMP AS E ON D.did = E.did
GROUP BY D.did
HAVING COUNT(E.eid) >= 1
ORDER BY D.did;
```

①

did	cnt
10	1
20	1
30	1

②

did	cnt
10	2
20	1

NoteBase

③

did	cnt
10	2
20	1
30	NULL

④

did	cnt
10	2
20	1
30	0

정답 2번

간단한 해설

COUNT(E.eid)는 NULL을 제외하므로 부서별 인원은 10번 2명, 20번 1명, 30번 0명이다. **HAVING 조건이 30번 부서를 제거**하여 최종 결과는 (10, 2), (20, 1)이다.

계산 과정

LEFT OUTER JOIN은 왼쪽 DEPT의 행을 모두 보존한다. 부서 10은 사원 1·2와 연결되고, 부서 20은 사원 3과 연결된다. 부서 30은 일치하는 사원이 없으므로 EMP 쪽 속성이 NULL인 행으로 남는다. EMP의 사원 4는 did가 NULL이어서 어떤 부서와도 동등 조건으로 연결되지 않으며, 오른쪽의 불일치 행이므로 결과에 보존되지 않는다.

GROUP BY D.did로 부서별 그룹을 만든 뒤 COUNT(E.eid)를 계산한다. **COUNT(속성)은 NULL을 세지 않으므로** 부서 10은 2, 부서 20은 1, 부서 30은 0이다. 부서 30의 외부조인 보충 행이 한 개 있다는 이유로 1을 세는 COUNT(*)와 혼동하지 않아야 한다.

HAVING COUNT(E.eid) >= 1은 집계값이 1 이상인 그룹만 남긴다. 따라서 **집계값이 0인 부서 30은 제거**되고 부서 10과 20만 남는다.

ORDER BY D.did는 부서번호를 오름차순으로 정렬한다. 최종 결과의 (did, cnt)는 (10, 2), (20, 1)이며, 이에 해당하는 표는 ②이다.

NoteBase

문 13. 릴레이션 R(A, B, C, D)에 함수종속 F = {A → B, B → C, C → D, D → A}가 성립한다. R의 후보키(candidate key) 총 개수는?

① 1

② 2

③ 3

④ 4

정답 4번

간단한 해설

함수종속이 A → B → C → D → A의 순환을 이루므로 **어느 한 속성에서도 모든 속성을 결정**할 수 있다. 네 단일 속성 집합이 각각 최소성을 만족하므로 후보키는 4개이다.

계산 과정

후보키의 조건: 후보키는 함수종속을 이용하여 모든 속성을 결정할 수 있는 슈퍼키 가운데 **어느 속성도 더 제거할 수 없는** 최소 속성 집합이다. 각 속성의 폐포를 구하여 전체 속성 A, B, C, D를 얻는지 확인한다.

A의 폐포: A → B, B → C, C → D를 차례로 적용하면 A로부터 B·C·D를 모두 얻는다. 따라서 $A^+ = \{A, B, C, D\}$이고 {A}는 후보키이다.

B의 폐포: B → C → D → A를 따라가면 모든 속성을 얻는다. 따라서 $B^+ = \{A, B, C, D\}$이고 {B}도 후보키이다.

C의 폐포: C → D → A → B를 적용하므로 $C^+ = \{A, B, C, D\}$이다. 따라서 {C}도 후보키이다.

D의 폐포: D → A → B → C를 적용하므로 $D^+ = \{A, B, C, D\}$이다. 따라서 {D}도 후보키이다.

최소성과 개수: 빈 집합으로는 주어진 함수종속을 적용하여 속성을 얻을 수 없으므로 위의 단일 속성 키들은 최소성을 만족한다. 두 개 이상의 속성 집합은 이미 그 안의 단일 속성 하나만으로 전체를 결정하므로 최소성이 없어 후보키가 아니다. 후보키는 {A}, {B}, {C}, {D}의 총 4개이며 ④와 일치한다.

NoteBase

문 14. 릴레이션 R(A, B, C, D)의 후보키가 {A, B}이고 함수종속이 F = {AB → C, AB → D, B → D}일 때, R이 만족 가능한 최고 정규형으로 가장 적절한 것은?

① 1NF

② 2NF

③ 3NF

④ BCNF

정답 1번

간단한 해설

비주요 속성 D가 복합 후보키 {A, B}의 일부인 B에 종속되어 **부분 함수종속**이 존재한다. 따라서 2NF를 만족하지 못하므로 만족 가능한 최고 정규형은 1NF이다.

계산 과정

주요 속성과 비주요 속성: 후보키가 {A, B}이므로 A와 B는 후보키를 구성하는 주요 속성이고, C와 D는 비주요 속성이다. AB → C와 AB → D에 따라 후보키 전체로 C와 D를 결정할 수 있다.

부분 함수종속 확인: B → D가 성립하므로 비주요 속성 D는 후보키 {A, B} 전체가 아니라 **그 진부분집합인 B에 종속**된다. 따라서 AB → D는 완전 함수종속이 아니라 부분 함수종속이다.

제2정규형 판정: 2NF는 1NF를 만족하면서 모든 비주요 속성이 각 후보키에 완전 함수종속되어야 한다. D에 부분 함수종속이 존재하므로 **2NF를 만족하지 못한다**. 3NF와 BCNF도 이를 넘어서는 조건이므로 만족할 수 없다.

최고 정규형 도출: 주어진 함수종속은 속성값이 원자값인 1NF의 성립을 막지 않지만, 부분 함수종속 때문에 2NF 이상은 불가능하다. 따라서 만족 가능한 최고 정규형은 1NF이며 ①과 일치한다.

NoteBase

문 15. 판매 테이블에 대한 질의 결과로 가장 적절한 것은?

< 판매 >

제품	지역	매출
P1	서울	100
P2	서울	200
P1	부산	150
P2	부산	50

```
SELECT SUM(CASE WHEN 지역='서울' THEN 매출 ELSE 50 END) AS 보정매출 FROM 판매;
```

① 보정매출: 200

② 보정매출: 300

③ 보정매출: 350

④ 보정매출: 400

정답 4번

간단한 해설

서울 행의 매출 100과 200은 그대로 사용하고, 부산의 두 행은 각각 **ELSE 값 50**으로 바꿔 합산한다. 보정매출은 100 + 200 + 50 + 50 = **400**이다.

계산 과정

CASE는 각 행마다 지역을 확인하여 서울이면 그 행의 매출을 반환하고, 서울이 아니면 **실제 매출과 관계없이 50**을 반환한다. WHERE 조건이 없으므로 부산 행도 합산 대상에서 제외되지 않는다.

서울의 P1 행은 매출 100을, 서울의 P2 행은 매출 200을 반환한다. 두 서울 행의 합은 100 + 200 = 300이다.

부산의 P1 행은 원래 매출이 150이지만 ELSE에 따라 50을 반환한다. 부산의 P2 행도 50을 반환하므로 두 부산 행의 합은 50 + 50 = 100이다.

SUM은 CASE가 반환한 네 값을 모두 합산하므로 보정매출은 **100 + 200 + 50 + 50 = 400**이다. GROUP BY가 없으므로 전체에 대한 집계 결과 한 행이 나오며, 선택지 ④에 해당한다.

NoteBase

문 16. 학생, 수강 테이블에 대해 다음 질의문의 결과에 포함되는 학번으로 가장 적절한 것은?

<학생>

학번
1
2
3

<수강>

학번
1
NULL

```
SELECT 학번
FROM 학생
WHERE 학번 NOT IN (SELECT 학번 FROM 수강);
```

① 1

② 2, 3

③ 1, 2, 3

④ 없음

정답 4번

간단한 해설

부질의 결과에 NULL이 있어 학번 2·3의 NOT IN 판단은 **TRUE가 아니라 UNKNOWN**이 된다. 학번 1은 FALSE이고 WHERE는 TRUE인 행만 선택하므로 결과에 포함되는 학번은 없다.

NoteBase

계산 과정

부질의 결과: 수강 테이블에서 학번을 조회하면 1과 NULL이 나온다. 따라서 각 학생의 학번에 대해 NOT IN (1, NULL)을 평가한다. NULL은 다른 특정 학번이 아니라 **값을 알 수 없음을 나타내는 표지**이다.

학번 1: 목록에 같은 값 1이 있으므로 IN의 결과는 TRUE이고 이를 부정한 NOT IN의 결과는 FALSE이다. 따라서 학번 1은 WHERE 조건을 통과하지 못한다.

학번 2: 2는 1과 다르지만 NULL과의 비교 결과는 UNKNOWN이다. NOT IN은 목록의 모든 값과 다르다고 확인되어야 TRUE인데, NULL과의 비교를 확정할 수 없으므로 결과는 **UNKNOWN**이다. WHERE는 UNKNOWN인 행을 포함하지 않는다.

학번 3: 3 역시 1과는 다르지만 NULL과의 비교가 UNKNOWN이므로 NOT IN의 결과가 UNKNOWN이다. 따라서 학번 3도 결과에서 제외된다.

최종 결과: WHERE는 조건이 TRUE인 행만 남기며 FALSE와 UNKNOWN은 모두 제외한다. 세 학번 가운데 조건이 TRUE인 행이 없으므로 결과는 빈 집합이고, 포함되는 학번이 없다는 ④가 적절하다.

NoteBase

문 17. 릴레이션 R의 카디널리티가 20,000이다. R의 속성 A가 균등 분포한다고 가정하고 서로 다른 값 40개를 가질 때, $\sigma_{A=k}(R)$에 대한 결과 튜플 수의 추정치로 가장 적절한 것은? (단, k는 서로 다른 값 40개 중 하나이다.)

① 40

② 500

③ 800

④ 20,000

정답 2번

간단한 해설

A의 40개 값이 균등하게 분포하므로 한 값에 대응하는 튜플 수는 전체의 **1/40**로 추정한다. 20,000 ÷ 40 = **500개**이다.

계산 과정

릴레이션의 카디널리티는 튜플 수이므로 전체 튜플은 20,000개이다. A의 서로 다른 값은 40개이고, 균등 분포한다는 조건은 **각 값이 나타나는 비율을 동일하게 추정**한다는 뜻이다.

A = k라는 동등 조건은 40개 값 중 하나를 선택한다. 따라서 선택도, 즉 전체 튜플 중 조건에 해당하는 비율은 1/40이다. k가 실제 존재하는 값이라는 조건도 주어져 있다.

예상 결과 튜플 수는 전체 튜플 수에 선택도를 곱하여 구한다. $20{,}000 \times \frac{1}{40} = 500$개이므로 선택지 ②에 해당한다.

NoteBase

문 18. 다음 응답 지연을 해결하기 위해 검토할 방안으로 가장 적절한 것은?

운영 DB의 특정 SQL이 전체 응답 지연의 대부분을 차지한다.
실행 계획을 보니 고객 테이블 500만 행을 매번 전체 스캔하고,
조건은 고객등급 = 'VIP' AND 가입일자 >= '2026-01-01'이다.
VIP 비율은 1%이고 최근 가입자 조건도 선택도가 높다.

① 고객 테이블의 보안 레벨을 낮춘다.

② 고객 테이블의 정규화 수준을 강화한다.

③ DBMS를 사용하지 않고 CSV 파일로 전환한다.

④ 고객등급과 가입일자를 고려한 복합 인덱스 후보를 검토한다.

정답 4번

간단한 해설

반복되는 전체 스캔을 줄이려면 **고객등급의 동등 조건과 가입일자의 범위 조건**을 활용할 접근 경로가 필요하다.
두 속성을 고려한 복합 인덱스가 탐색 범위를 줄이는지 검토하는 것이 적절하다.

NoteBase

핵심 풀이

문제에서 확인된 병목은 특정 SQL이 고객 500만 행을 반복해서 전체 스캔하는 것이다. 따라서 먼저 검토할 대상은 보안이나 데이터 저장 형식의 변경이 아니라 **검색 조건에 맞는 접근 경로**이다.

고객등급 = VIP는 동등 조건이고 가입일자 >= 2026-01-01은 범위 조건이다. 일반적인 B+트리 계열 인덱스라면 **동등 조건인 고객등급을 앞에, 범위 조건인 가입일자를 뒤에 둔 복합 인덱스**를 후보로 검토할 수 있다. VIP 구간을 찾은 뒤 해당 구간의 최근 가입일 범위로 탐색을 좁힐 수 있기 때문이다.

VIP는 전체의 1%이므로 등급 조건만으로도 후보가 약 5만 행으로 줄어든다. 최근 가입일 조건까지 함께 만족하는 행을 탐색하면 전체 스캔보다 읽는 양을 줄일 가능성이 있다. 다만 두 조건의 상관관계와 실제 반환 행 수, 조회 열, 인덱스 유지 비용 등을 확인해야 하므로 무조건 빨라진다고 단정하지 않고 실행 계획과 성능을 비교해야 한다.

보안 수준을 낮추는 것은 확인된 스캔 병목을 해결하는 방법이 아니다.

정규화 강화는 데이터 중복과 이상 현상을 다루는 설계 방안으로, 이 검색 조건의 접근 경로를 직접 개선하지 않는다.

CSV 전환도 인덱스와 질의 최적화 기능을 활용할 기회를 없애므로 적절한 우선 대책이 아니다.

따라서 고객등급과 가입일자를 고려한 복합 인덱스 후보를 검토하는 ④가 가장 적절하다.

NoteBase

문 19. 다음 상품 테이블에 가~라의 레코드를 입력할 때, 허용되는 것을 모두 고른 것으로 가장 적절한 것은?

```
CREATE TABLE 상품 ( 상품번호 INT PRIMARY KEY,
                    가격 INT CHECK (가격 > 0) );
```

가. (1, 100)
나. (2, 0)
다. (3, -50)
라. (4, 1)

① 가, 라
② 나, 다
③ 가, 나, 라
④ 가, 나, 다, 라

정답 1번

간단한 해설

CHECK (가격 > 0)은 가격이 **0보다 큰 값**일 때 충족된다. 가격 100과 1은 허용되지만, 0과 -50은 조건을 위반한다.

NoteBase

보기별 해설

상품번호에는 기본키 제약이 있고 가격에는 CHECK (가격 > 0)이 있다. 제시된 상품번호는 모두 서로 다른 정수이므로, 네 보기의 허용 여부는 가격이 0보다 큰지를 확인하여 판단한다.

가. O: 가격 100은 **0보다 크므로** CHECK 조건을 만족한다. 상품번호 1도 다른 보기와 중복되지 않으므로 (1, 100)은 입력할 수 있다.

나. X: 가격 0은 0보다 크지 않으므로 CHECK 조건을 위반한다. 조건이 가격 ≥ 0이 아니라 **가격 > 0**이므로, 허용되려면 가격을 양의 정수로 바꾸어야 한다.

다. X: 가격 -50은 음수이므로 가격 > 0 조건을 만족하지 못한다. 상품번호 3에 문제가 없어도 가격 제약 위반 때문에 입력할 수 없으며, 가격은 양의 정수여야 한다.

라. O: 가격 1은 0보다 크므로 CHECK 조건을 만족한다. 상품번호 4도 다른 보기와 중복되지 않으므로 (4, 1)은 입력할 수 있다.

허용되는 레코드는 가와 라이며 ①과 일치한다.

NoteBase

문 20. 다음 분산 질의 처리에 대한 설명으로 가장 적절한 것은?

전역 질의가 서울 노드의 주문 테이블과 부산 노드의 부산고객 테이블을 고객ID로 조인한다. 네트워크 전송 비용을 줄이기 위해 서울에서 주문한 고객ID만 추출해 부산으로 보내고, 부산 노드에서 매칭된 부산 고객ID만 다시 서울로 보내 조인을 축소하려 한다.

① 네스티드 루프 조인 수행

② 세미 조인 수행

③ 외부 소트 머지 조인 수행

④ 해시 조인 수행

정답 2번

간단한 해설

테이블 전체 대신 **조인에 필요한 고객ID만 교환**하고, 일치하는 고객ID로 조인 대상을 미리 줄이는 것은 분산 질의에서 활용하는 **세미 조인** 방식이다.

공통 해설

세미 조인은 상대 테이블에 조인 조건을 만족하는 행이 존재하는지 확인하여 **한쪽 테이블의 행만 남기는 연산**이다. 두 테이블의 모든 속성을 결합하기 전에, 실제로 조인될 수 있는 행을 추려 내는 데 활용할 수 있다.

지문에서는 서울의 주문 테이블에서 고객ID만 추출해 부산으로 보낸다. 부산에서는 이 고객ID와 일치하는 부산고객의 고객ID만 서울로 돌려보낸다. 서울은 돌아온 고객ID로 주문 테이블을 걸러, 부산고객과 연결될 수 없는 주문을 조인 대상에서 제외할 수 있다. 이처럼 **조인 속성값으로 일치 여부를 확인해 대상을 축소**하는 것이 세미 조인을 이용한 처리이다. 테이블의 모든 열과 행을 먼저 전송하는 대신 필요한 속성값을 교환하여 네트워크 전송량을 줄이려는 것이다.

네스티드 루프, 외부 소트 머지, 해시는 조인을 물리적으로 수행하는 방식에 관한 후보이다. 지문에는 반복 탐색·정렬·해시 중 무엇으로 일치 여부를 검사하는지 제시되어 있지 않다. 따라서 이 사례를 식별하는 기준은 특정 물리적 알고리즘이 아니라, 조인할 수 있는 행만 미리 남기는 세미 조인의 원리이다.

NoteBase

문 21. 다음 릴레이션 R을 R1과 R2로 분해한 뒤 자연 조인으로 복원할 때, 그 결과 릴레이션의 튜플 수로 가장 적절한 것은?

< R >

A	B	C
a1	b1	c1
a2	b1	c2

< R1 >

A	B
a1	b1
a2	b1

< R2 >

B	C
b1	c1
b1	c2

① 0

② 2

③ 4

NoteBase

④ 6

정답 3번

간단한 해설

두 릴레이션의 모든 튜플이 공통 속성 B = b1을 가지므로 각 릴레이션의 두 튜플이 서로 모두 결합한다. 결과는 **2 × 2 = 4개**이며, 원래 R에 없던 조합도 생성된다.

계산 과정

조인 기준 확인: R1의 속성은 A·B이고 R2의 속성은 B·C이므로 자연 조인은 공통 속성 B의 값이 같은 튜플끼리 결합한다. 결과에는 B를 한 번만 남겨 A·B·C를 구성한다.

일치하는 튜플 수: R1에는 B = b1인 (a1, b1), (a2, b1)의 두 튜플이 있고, R2에도 B = b1인 (b1, c1), (b1, c2)의 두 튜플이 있다. 따라서 R1의 각 튜플은 R2의 두 튜플 모두와 결합하며, 결합 수는 **2 × 2 = 4**이다.

결과 튜플: 결합 결과는 (a1, b1, c1), (a1, b1, c2), (a2, b1, c1), (a2, b1, c2)이다. 네 튜플은 A 또는 C가 서로 달라 중복 제거로 줄어들지 않는다.

원래 R과의 차이: 원래 존재한 (a1, b1, c1), (a2, b1, c2) 외에 (a1, b1, c2), (a2, b1, c1)이 새로 생긴다.
분해하면서 A와 C 사이의 원래 짝 관계가 보존되지 않아 **허위 튜플이 발생**한 것이므로 원본의 두 행만 남는다고 볼 수 없다.

자연 조인 결과의 튜플 수는 총 4개이며 ③과 일치한다.

NoteBase

문 23. 다음 조건의 자기 디스크에서 임의의 블록 하나를 읽는 데 걸리는 평균 접근 시간(ms)으로 가장 적절한 것은?

◯ 디스크 회전 속도 : 6,000 RPM
◯ 평균 탐구 시간(average seek time) : 5 ms
◯ 블록 하나의 전송 시간(transfer time) : 1 ms

① 6
② 10
③ 11
④ 16

정답 3번

간단한 해설

한 바퀴 회전 시간은 10 ms이므로 평균 회전 지연은 **반 바퀴인 5 ms**이다. 탐구 5 ms와 전송 1 ms를 더하면 평균 접근 시간은 **11 ms**이다.

계산 과정

블록 하나를 읽는 평균 시간은 헤드를 해당 트랙으로 옮기는 평균 탐구 시간, 목표 블록이 헤드 아래로 올 때까지의 평균 회전 지연 시간, 블록 전송 시간을 더하여 구한다. 별도로 주어지지 않은 대기열 시간이나 제어장치 지연은 추가하지 않는다.

6,000 RPM은 1분에 6,000회 회전한다는 뜻이다. 1분은 60,000 ms이므로 한 바퀴 회전 시간은 60,000 ÷ 6,000 = 10 ms이다.

임의의 블록을 기다리는 회전 지연은 평균적으로 반 바퀴에 해당한다. 따라서 **평균 회전 지연 시간은 10 ÷ 2 = 5 ms**이다. 평균 회전 지연에는 한 바퀴 전체가 아닌 반 바퀴의 회전 시간을 적용한다.

평균 접근 시간은 5 ms(탐구) + 5 ms(회전 지연) + 1 ms(전송) = **11 ms**이며, 선택지 ③에 해당한다.

NoteBase

문 24. 다음과 같은 데이터 접근 상황에서 적합한 파일 조직을 연결한 것으로 가장 적절한 것은?

가. 대량 로그 레코드를 파일 끝에 계속 덧붙여 저장만 하고, 개별 레코드 검색은 거의 하지 않는다.
나. 특정 키 값에 대한 등호(=) 검색이 대부분이고, 범위 검색은 거의 발생하지 않는다.
다. 날짜 범위에 대한 검색과 정렬된 순서의 순차 출력이 매우 빈번하다.

① 가: 순차 파일, 나: 해시 파일, 다: 힙 파일
② 가: 해시 파일, 나: 힙 파일, 다: 순차 파일
③ 가: 힙 파일, 나: 순차 파일, 다: 해시 파일
④ 가: 힙 파일, 나: 해시 파일, 다: 순차 파일

정답 4번

간단한 해설

정렬을 유지하지 않고 추가 저장하는 데에는 **힙 파일**, 특정 키의 등호 검색에는 **해시 파일**이 적합하다. 날짜 범위 검색과 정렬 순서의 출력에는 날짜를 기준으로 정렬된 순차 파일이 적합하다.

대응 관계 해설

가: 힙 파일이 적합하다. 힙 파일은 검색 키 순서로 레코드를 정렬하지 않고 빈 공간이나 파일 끝에 저장한다. 따라서 새 로그를 계속 추가할 때 **키 순서를 유지하기 위한 재배치 부담이 작다**. 개별 검색 시 전체를 훑는 비용이 들 수 있지만, 사례에서는 그러한 검색이 거의 없으므로 삽입이 단순하다는 장점이 중요하다.

나: 해시 파일이 적합하다. 키 값에 해시 함수를 적용하여 해당 레코드가 저장될 버킷을 찾으므로 **특정 키의 등호 검색**에 유리하다. 반면 가까운 키 값이 인접한 위치에 저장된다는 보장이 없어 범위 검색에는 불리하지만, 사례에서는 범위 검색이 거의 없으므로 적합하다.

다: 날짜를 정렬 키로 하는 순차 파일이 적합하다. 레코드를 키 순서로 저장하므로 필요한 날짜 구간을 찾은 뒤 그 범위의 레코드를 순서대로 읽기 쉽다. **범위 검색과 정렬 순서의 연속 출력**이 빈번하다는 조건에 맞는다.

가에는 힙 파일, 나에는 해시 파일, 다에는 순차 파일을 대응시켜야 하므로 ④가 적절하다.

NoteBase

문 25. 다음 인덱스에 대한 설명으로 적절한 것은 모두 몇 개인가?

가. 보조(secondary) 인덱스는 데이터 파일이 정렬되어 있지 않을 수 있으므로 반드시 밀집(dense) 인덱스이어야 한다.
나. 희소(sparse) 인덱스는 데이터 파일이 인덱스 대상 필드로 정렬되어 있을 때에만 사용할 수 있다.
다. 클러스터링(clustering) 인덱스는 하나의 테이블에 여러 개 만들 수 있다.
라. 기본(primary) 인덱스는 데이터 파일의 각 블록마다 하나의 엔트리를 두는 희소 인덱스로 구성할 수 있다.

① 1
② 2
③ 3
④ 4

정답 3번

간단한 해설

클러스터링 인덱스는 데이터의 **물리적 배열 순서와 연결되므로 하나만 둘 수 있다**. 보조 인덱스의 밀집성, 정렬된 파일에서의 희소 인덱스 사용, 기본 인덱스의 블록 단위 구성은 적절한 설명이다.

NoteBase

보기별 해설

이 문항은 데이터 파일의 정렬 순서와 연결하여 구분하는 기본·보조·클러스터링 인덱스의 전통적인 분류를 전제로 한다.

가 O: 보조 인덱스는 데이터 파일의 물리적 정렬 순서와 다른 필드를 대상으로 하므로, 일부 값의 위치만 알아서는 나머지 레코드의 위치를 순차적으로 찾을 수 없다. 따라서 **모든 검색키 값에 접근할 수 있는 밀집 인덱스**가 필요하다. 중복 값이 있으면 한 엔트리가 그 값의 여러 레코드 위치를 가리키는 방식도 가능하다.

나 O: 희소 인덱스는 검색키 값 중 일부에만 엔트리를 둔다. 해당 값 이하의 가까운 엔트리부터 데이터 파일을 따라가며 목표를 찾으려면 **데이터가 그 검색키 순서로 정렬**되어 있어야 한다.

다 X: 클러스터링 인덱스는 검색키 값에 따라 관련 레코드가 물리적으로 모이도록 한 데이터 배치를 바탕으로 한다. 한 데이터 파일의 물리적 배열 순서는 하나이므로, 이 분류에서 **하나의 테이블에 여러 클러스터링 인덱스를 둘 수 없다**. 별도의 보조 인덱스는 여러 개 만들 수 있다.

라 O: 기본 인덱스는 기본키 순서로 정렬된 데이터 파일을 대상으로 한다. 각 블록의 대표 키와 블록 위치만 기록해도 그 블록 안에서 목표 레코드를 찾을 수 있으므로, **블록당 하나의 엔트리를 두는 희소 형태**로 구성할 수 있다.

적절한 설명은 가, 나, 라의 3개이므로 선택지 ③에 해당한다.

NoteBase

문 26. 다음 조건의 B+트리에서 내부 노드가 가질 수 있는 최대 차수, 즉 자식 포인터의 최대 개수 n을 구한 것으로 가장 적절한 것은? (단, 내부 노드는 $n-1$개의 키와 n개의 포인터를 가진다.)

◯ 노드(블록) 크기 : 4,096 바이트
◯ 탐색 키 값 하나의 길이 : 12 바이트
◯ 트리 포인터 하나의 길이 : 8 바이트

① 204
② 205
③ 348
④ 512

정답 2번

간단한 해설

내부 노드는 n개의 포인터와 n−1개의 키를 가지므로 **8n+12(n-1)≤4,096**을 만족해야 한다. n≤205.4에서 정수인 최대 차수는 **205**이다.

계산 과정

문제의 차수 n은 자식 포인터의 개수이다. 내부 노드에는 **포인터가 n개, 키가 n−1개** 들어가므로 두 종류의 개수를 같게 놓으면 안 된다.

키 하나는 12바이트이고 포인터 하나는 8바이트이므로 필요한 공간은 $12(n-1)+8n=20n-12$ 바이트이다. 이 공간이 노드 크기 4,096바이트 이하여야 하므로 20n-12≤4,096이다.

부등식을 정리하면 20n≤4,108, 즉 n≤205.4이다. 포인터 개수는 정수이므로 가능한 최댓값은 **205개**이다.

검산하면 n=205일 때 204×12 + 205×8 = 4,088바이트로 들어가지만, n=206일 때는 205×12 + 206×8 = 4,108바이트로 초과한다. 따라서 최대 차수는 205이며 선택지 ②에 해당한다.

NoteBase

문 27. 트랜잭션 T_1이 변수 X에 배타 잠금을 보유한 상태에서 트랜잭션 T_2가 X에 공유 잠금을 요청하였다. 엄격한 2단계 잠금 규약을 따를 때, T_2의 요청에 대한 처리로 가장 적절한 것은?

① 즉시 허용된다.

② T_1이 배타 잠금을 해제할 때까지 대기한다.

③ T_2가 T_1의 배타 잠금을 강제로 공유 잠금으로 낮춘다.

④ 공유 잠금은 읽기용이므로 배타 잠금과 항상 호환된다.

정답 2번

간단한 해설

같은 데이터 항목에 대한 **배타 잠금과 다른 트랜잭션의 공유 잠금은 호환되지 않는다**. 엄격한 2단계 잠금에서는 T_1이 종료할 때까지 배타 잠금을 유지하므로 T_2는 그 잠금이 해제될 때까지 기다린다.

핵심 풀이

잠금 호환성: 공유 잠금은 여러 트랜잭션의 읽기를 함께 허용하지만, 배타 잠금은 다른 트랜잭션의 공유·배타 잠금과 모두 충돌한다. 따라서 같은 X에 대해 T_1의 배타 잠금이 유지되는 동안 T_2의 공유 잠금은 **호환되지 않아 즉시 허용되지 않는다**. 읽기 요청이라고 해서 다른 트랜잭션의 배타 잠금을 무시할 수는 없다.

엄격한 2단계 잠금의 효과: 엄격한 2단계 잠금 규약에서는 배타 잠금을 트랜잭션이 커밋하거나 중단할 때까지 유지한다. 이로써 다른 트랜잭션이 아직 확정되지 않은 변경을 읽거나 덮어쓰는 것을 막는다. 주어진 상황에서 T_2는 **T_1이 배타 잠금을 해제할 때까지 대기**한다.

잠금 강제 변경과의 구별: T_2가 다른 트랜잭션인 T_1의 배타 잠금을 임의로 공유 잠금으로 낮추지는 않는다. 공유 잠금끼리는 호환되지만 공유 잠금과 배타 잠금은 호환되지 않으므로, 강제 강등이나 항상 호환된다는 설명도 적용되지 않는다.

따라서 기존 배타 잠금의 해제를 기다린다는 ②가 적절하다.

NoteBase

문 28. RAID 1로 2TB 디스크 6개를 구성한다. 모든 디스크는 동일 용량이고 미러링 쌍을 구성한다고 할 때 사용 가능한 데이터 용량과 한 쌍 내 디스크 1개 장애 시 읽기 가능 여부로 가장 적절한 것은?

① 용량(TB): 6, 읽기: 가능

② 용량(TB): 6, 읽기: 불가능

③ 용량(TB): 12, 읽기: 가능

④ 용량(TB): 12, 읽기: 불가능

정답 1번

간단한 해설

6개 디스크를 두 개씩 미러링하면 3쌍이므로 실제 데이터 용량은 **3 × 2 TB = 6 TB**이다. 한 쌍에서 하나가 고장 나도 **남은 복사본으로 읽을 수 있다**.

계산 과정

미러링은 두 디스크에 같은 데이터를 저장하는 방식이다. 문제에서 여섯 디스크가 미러링 쌍을 구성한다고 했으므로, **6 ÷ 2 = 3쌍**이며 모든 디스크를 하나의 여섯 중 복사본으로 묶는 경우가 아니다.

한 쌍의 물리 용량은 2 TB × 2 = 4 TB이지만 같은 내용을 중복 저장하므로 실제 데이터 용량은 2 TB이다. 세 쌍의 사용 가능한 총 데이터 용량은 **3 × 2 TB = 6 TB**이다.

한 쌍에서 디스크 하나가 고장 나도 다른 디스크에 같은 데이터가 남아 있으므로 그 복사본에서 읽을 수 있다. 다만 그 쌍의 남은 디스크까지 고장 나면 해당 데이터의 가용성이 유지된다고 보장할 수 없다.

따라서 사용 가능한 데이터 용량은 6 TB이고 한 쌍 내 디스크 한 개의 장애 시 읽기가 가능하므로 선택지 ①에 해당한다.

NoteBase

문 29. 레코드를 블록 경계에서 분할하지 않는 비신장(unspanned) 방식으로 파일을 저장할 때, 다음 조건에서 필요한 최소 블록 수로 가장 적절한 것은?

◯ 레코드 개수 : 40,000개
◯ 레코드 하나의 길이 : 100 바이트(고정 길이)
◯ 블록 크기 : 1,024 바이트

① 3,907
② 4,000
③ 4,100
④ 4,167

정답 2번

간단한 해설

레코드를 분할할 수 없으므로 한 블록에는 **100바이트 레코드 10개**만 들어간다. 40,000개를 10개씩 저장하면 필요한 블록은 4,000개이다.

계산 과정

저장 조건: 비신장 방식에서는 레코드 하나가 두 블록에 걸쳐 저장될 수 없다. 따라서 총 바이트 수만 블록 크기로 나누는 것이 아니라, 먼저 한 블록에 들어가는 **완전한 레코드의 개수**를 구해야 한다.

블록당 레코드 수: 블록 크기는 1,024바이트이고 레코드 길이는 100바이트이다. 따라서 블록당 저장 가능한 수는 $\left\lfloor \frac{1,024}{100} \right\rfloor = 10$개이다. 10개를 저장하면 1,000바이트를 사용하고 24바이트가 남지만, 이 공간에는 100바이트 레코드를 더 넣을 수 없다.

필요 블록 수: 전체 레코드 40,000개를 블록당 10개씩 저장하므로 최소 블록 수는 $\left\lceil \frac{40,000}{10} \right\rceil = 4,000$개이다. 이번에는 정확히 나누어떨어지므로 마지막에 별도의 부분 사용 블록을 추가할 필요가 없다.

총 데이터 크기를 블록 크기로 바로 나누면 각 블록에 남는 24바이트를 서로 이어 사용할 수 있다고 가정하게 된다. 그러나 **레코드를 분할하지 않는 조건**에서는 그렇게 계산할 수 없으므로 4,000개를 고른 ②가 적절하다.

NoteBase

문 30. 계좌 테이블에 잔액이 음수가 되면 오류를 발생시키는 BEFORE UPDATE 트리거가 있다. 한 트랜잭션이 잔액 50인 계좌에 대해 "잔액 = 잔액 - 70"을 수행하였다. 트리거가 정상 동작할 때의 설명으로 가장 적절한 것은?

① 갱신이 거부되고 트랜잭션은 오류 처리 대상이 된다.

② 잔액은 -20으로 저장되고 트리거는 다음 체크포인트 때 실행된다.

③ 트리거는 SELECT 문에서만 실행되므로 아무 영향이 없다.

④ BEFORE UPDATE 트리거는 갱신 완료 후 로그만 삭제한다.

정답 1번

간단한 해설

새 잔액은 50−70 = **−20**으로 음수이다. BEFORE UPDATE 트리거가 저장 전에 이를 검사해 오류를 발생시키므로 **해당 갱신은 거부**된다.

계산 과정

갱신하려는 새 잔액을 계산하면 50−70 = −20이다. 이 값은 **0보다 작으므로 트리거의 오류 발생 조건**에 해당한다.

BEFORE UPDATE 트리거는 갱신 내용이 적용되기 전에 실행된다. 문제에서 음수 잔액을 오류로 처리하도록 했으므로, 새 잔액 −20을 저장하려는 갱신은 거부되고 오류가 발생한다.

해당 트랜잭션은 오류 처리 대상이 되며, 구체적인 문장 취소·트랜잭션 처리 범위는 DBMS와 오류 처리 방식에 따를 수 있다. 여기서 확정할 수 있는 것은 **트리거가 정상 동작하면 음수 잔액의 갱신이 성공하지 않는다는 점**이다.

체크포인트는 트리거의 실행 시점을 정하는 절차가 아니며, 이 트리거는 SELECT가 아니라 UPDATE에 반응한다.

BEFORE UPDATE가 갱신 후 로그를 삭제한다는 설명도 맞지 않는다.

따라서 갱신 거부와 오류 처리를 설명한 ①이 가장 적절하다.

NoteBase

문 31. 다음 상황에서 발생할 수 있는 직렬성 위반 문제로 가장 적절한 것은?

트랜잭션 T_1이 변수 A의 값을 100에서 50으로 변경하고 아직 커밋하지 않았다. 트랜잭션 T_2가 A의 값으로 50을 읽어 보고서를 생성한 직후 T_1이 롤백되었다.

① 오손 읽기(dirty read)

② 반복 불가능 읽기(unrepeatable read)

③ 팬텀 읽기(phantom read)

④ 갱신 손실(lost update)

정답 1번

간단한 해설

T_2는 T_1이 **아직 커밋하지 않은 값 50**을 읽었다. 이처럼 다른 트랜잭션의 미확정 변경값을 읽는 현상이 오손 읽기이며, 이후 T_1의 롤백으로 보고서는 취소된 값에 근거하게 된다.

공통 해설

오손 읽기(dirty read)는 다른 트랜잭션이 변경했지만 **아직 커밋하지 않은 데이터를 읽는 현상**이다. 커밋은 변경을 확정하는 것이고, 롤백은 변경을 취소하여 이전 상태로 되돌리는 것이다.

T_1은 A를 100에서 50으로 변경한 뒤 커밋하지 않았으므로, 50은 아직 확정되지 않은 값이다. T_2는 이 값을 읽어 보고서를 생성했고, 그 직후 T_1이 롤백되어 A는 100으로 돌아간다. 따라서 T_2의 보고서는 취소된 변경값 50에 근거하게 된다. 이 사례를 오손 읽기로 식별하는 기준은 T_2가 읽는 시점에 T_1의 변경이 미커밋 상태였다는 점이다.

NoteBase

문 32. 다음과 같이 트랜잭션 실행 중 시스템 장애가 발생하였다. 회복 기법이 성공적으로 수행되었을 때 변수 X, Y의 값으로 적절한 것은? (단, WAL(write-ahead logging) 규약과 steal/no-force 방식으로 캐시를 관리한다고 가정하고, 변경 로그 레코드 형식은 <트랜잭션 식별자, 변수, 이전값, 새값> 이다.)

<START_TRANSACTION, T_1>
<T_1, X, 100, 130>
<START_TRANSACTION, T_2>
<T_2, Y, 200, 240>
<COMMIT T_1>
<CRASH>

① X: 100, Y: 200

② X: 100, Y: 240

③ X: 130, Y: 200

④ X: 130, Y: 240

정답 3번

간단한 해설

커밋한 T_1의 X 변경은 보존하고, 커밋하지 않은 T_2의 Y 변경은 취소한다. 따라서 회복 후 값은 **X = 130, Y = 200**이다.

계산 과정

장애 직전 로그에서 T_1에는 COMMIT 기록이 있지만 T_2에는 없다. 따라서 **T_1의 변경은 보존하고 T_2의 미완료 변경은 취소**해야 한다.

no-force 방식에서는 커밋한 변경도 아직 디스크에 반영되지 않았을 수 있으므로 REDO가 필요하다. T_1의 로그는 X를 100에서 130으로 변경했음을 나타내므로, 회복 후 X = 130이 되어야 한다.

steal 방식에서는 커밋하지 않은 변경도 디스크에 기록될 수 있으므로 UNDO가 필요하다. T_2의 로그는 Y를 200에서 240으로 변경했음을 나타내지만 커밋하지 못했으므로, 이전 값으로 되돌려 Y = 200이 된다.

WAL은 변경 데이터보다 관련 로그를 먼저 영속화하여 이러한 REDO와 UNDO의 근거를 확보하는 규약이다. 실제 디스크에 어느 변경이 이미 반영되었는지와 관계없이, 성공적인 회복의 최종 결과는 **X = 130, Y = 200**이므로 ③이다.

NoteBase

문 33. 다음 상황에 해당하는 설명으로 가장 적절한 것은?

2단계 커밋에서 조정자가 모든 참여자로부터 YES를 받은 후 COMMIT 메시지를 보내는 중 장애가 발생했다. 참여자 일부는 아직 최종 결정을 받지 못했다.

① 참여자는 독자적으로 항상 COMMIT을 결정할 수 있다.

② 참여자가 불확실 상태에서 조정자의 결정을 기다릴 수 있어 블로킹이 발생할 수 있다.

③ 2단계 커밋에서는 로그를 사용하지 않는다.

④ YES 투표 후에는 어떤 장애도 발생하지 않는다.

정답 2번

간단한 해설

YES를 보낸 참여자도 최종 결정을 받지 못하면 전체 트랜잭션과 다른 결과를 독자적으로 확정할 수 없다. 조정자 장애로 결정을 확인하지 못한 채 기다리는 **블로킹**이 발생할 수 있다.

핵심 풀이

① X: 2단계 커밋의 준비 단계에서 참여자는 로컬 처리를 커밋할 준비가 되면 YES를 보낸다. 이는 참여자 자신의 준비 완료를 뜻할 뿐, 전체 트랜잭션의 **최종 커밋 결정을 전달받았다는 뜻은 아니다**. YES를 보낸 참여자는 전체 결정과 다른 결과를 임의로 확정해서는 안 된다. 결정을 모르는 참여자가 언제나 독자적으로 COMMIT할 수 있다는 설명도 원자적 완료 원칙에 맞지 않는다.

② O: 조정자가 COMMIT을 일부에게만 전달하고 중단되면, 아직 결정을 받지 못한 참여자는 다른 참여자의 처리 결과와 일치시켜야 한다. 최종 결정을 확인할 수 없는 동안에는 자원이나 잠금을 유지한 채 **결정 통보 또는 복구를 기다리는 블로킹**이 발생할 수 있다. 다른 참여자로부터 확정된 결정을 알아낼 수 있다면 진행할 수 있지만, 항상 가능한 것은 아니다.

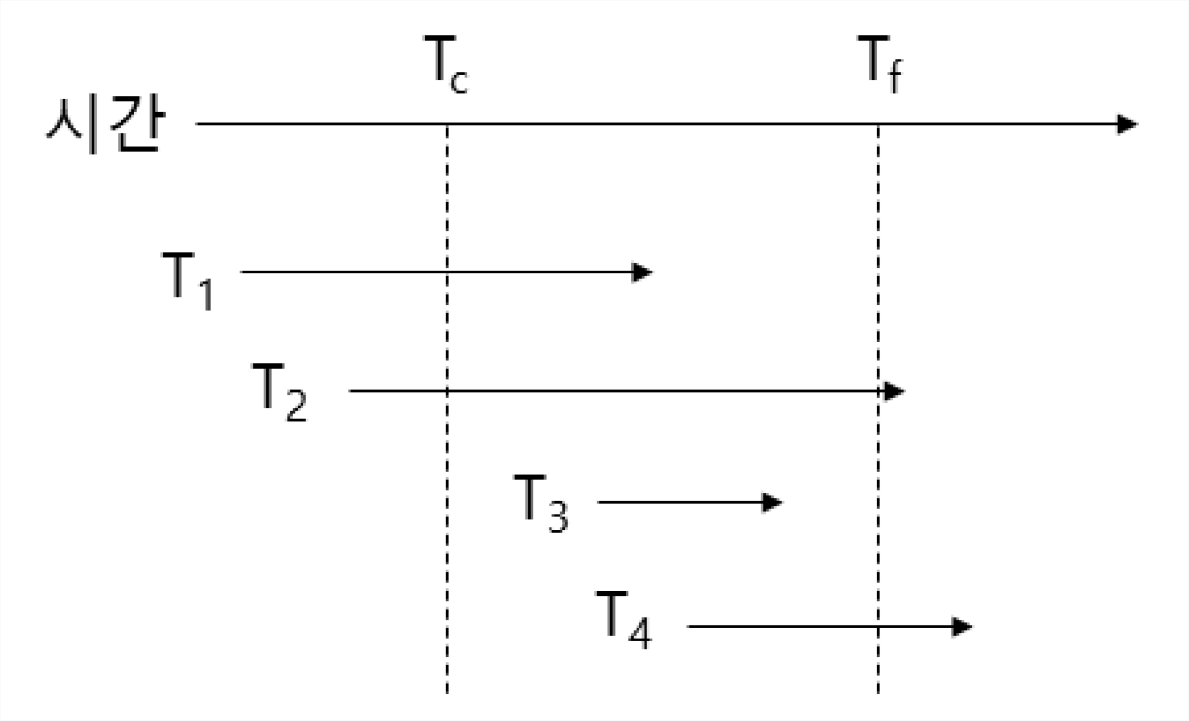
③ X: 2단계 커밋은 준비 상태와 최종 결정 등을 로그에 남겨 장애 후 복구에 사용한다. 따라서 로그를 사용하지 않는다는 설명은 틀리다.

④ X: YES 투표는 장애가 이후에 발생하지 않는다는 보장도 아니다.

따라서 최종 결정을 받지 못한 참여자가 불확실 상태에서 기다려 블로킹이 발생할 수 있다는 ②가 적절하다.

NoteBase

문 34. 검사점(checkpoint) 회복 기법을 적용한 트랜잭션 수행 과정이 다음과 같을 때 데이터 복원을 위해 수행되는 작업으로 가장 적절하지 않은 것은? (단, T_c는 검사점, T_f는 장애시점을 의미한다.)



① T_1은 T_c 이후에 일어난 변경 부분에 대해서만 redo 한다.

② T_2는 T_c 이후에 일어난 변경 부분에 대해서만 undo 한다.

③ T_3는 처음부터 끝까지 redo 한다.

④ T_4는 처음부터 끝까지 undo 한다.

정답 2번

간단한 해설

검사점은 그때 실행 중이던 트랜잭션을 커밋시키는 절차가 아니다. 장애 때 미완료인 T_2는 **검사점 이전의 변경까지 포함하여 전체를 undo**해야 하므로, 검사점 이후만 취소한다는 설명이 틀리다.

NoteBase

선지별 해설

① O: T_1은 검사점 이전에 시작하여 검사점 이후, 장애 이전에 완료되었다. 이 문항의 검사점 회복 모형에서는 검사점까지의 변경은 저장되어 있으므로, 완료된 트랜잭션의 결과를 보장하기 위해 **검사점 이후의 변경을 redo**하면 된다.

② X: T_2는 검사점 이전에 시작했지만 장애 시점에도 완료되지 않았다. 검사점이 있었다고 미완료 트랜잭션의 변경이 커밋된 것은 아니므로, 원자성을 보장하려면 **검사점 이전을 포함하여 시작 이후의 변경 전체를 undo**해야 한다. 검사점 이후 부분만 취소하면 그 이전에 반영된 미완료 변경이 남게 된다.

③ O: T_3는 검사점 이후에 시작하여 장애 전에 완료되었다. 모든 변경이 검사점 이후에 이루어졌으므로, 완료 결과를 복원하기 위해 트랜잭션의 변경 전체를 redo한다. 그림에서 시작과 종료가 모두 두 기준 시점 사이에 있다는 점이 근거이다.

④ O: T_4는 검사점 이후에 시작하였고 장애 시점까지 완료되지 않았다. 따라서 장애 이전에 실제 수행된 변경을 **트랜잭션 시작부터 모두 undo**하여 원래 상태로 되돌린다. 아직 수행되지 않은 장애 이후 작업을 취소한다는 뜻은 아니다.

NoteBase

문 35. 다음 스케줄 S_1, S_2 중에서 직렬 가능한(serializable) 스케줄을 모두 고른 것은?

<스케줄 S_1>

시간 \ TID	T_1	T_2	T_3
1	read(X)		
2		read(Y)	
3			read(Z)
4	read(Y)		
5		read(Z)	
6			read(X)
7	write(X)		
8		write(Y)	
9			write(Z)

<스케줄 S_2>

시간 \ TID	T_1	T_2	T_3
1	read(X)		
2		read(Y)	
3	read(Y)		
4			read(Z)
5			read(X)
6	write(X)		
7			write(Z)
8		read(Z)	
9		write(Y)	

① S_1

② S_2

③ S_1, S_2

④ 없음

정답 2번

간단한 해설

S_1은 선행 그래프에 순환이 생기지만, S_2는 $\mathbf{T_3 \rightarrow T_1 \rightarrow T_2}$로 직렬화할 수 있다. 따라서 직렬 가능한 스케줄은 $\mathbf{S_2}$**만**이다.

NoteBase

계산 과정

서로 다른 트랜잭션이 같은 데이터에 접근하고 둘 중 하나 이상이 쓰기이면 충돌한다. 먼저 실행한 연산의 트랜잭션에서 나중 트랜잭션으로 간선을 그린 뒤, **선행 그래프에 순환이 없는지** 확인한다. 읽기와 읽기만의 순서는 간선을 만들지 않는다.

S_1에서는 X에 대한 T_3의 읽기(시간 6)가 T_1의 쓰기(시간 7)보다 앞서므로 $T_3 \rightarrow T_1$이다. Y에서는 T_1의 읽기(시간 4)가 T_2의 쓰기(시간 8)보다 앞서 $T_1 \rightarrow T_2$이고, Z에서는 T_2의 읽기(시간 5)가 T_3의 쓰기(시간 9)보다 앞서 $T_2 \rightarrow T_3$이다.

따라서 S_1에는 **$T_3 \rightarrow T_1 \rightarrow T_2 \rightarrow T_3$의 순환**이 생긴다. 이 충돌 순서를 모두 만족하는 직렬 순서는 없다. 각 쓰기가 같은 트랜잭션의 선행 읽기를 갖는 이 스케줄에서는 읽기 대상 값의 출처도 이 순서를 요구하므로, 뷰 직렬화로도 이를 해결할 수 없다.

S_2에서는 X의 T_3 읽기(시간 5)와 T_1 쓰기(시간 6)로 $T_3 \rightarrow T_1$, Y의 T_1 읽기(시간 3)와 T_2 쓰기(시간 9)로 $T_1 \rightarrow T_2$, Z의 T_3 쓰기(시간 7)와 T_2 읽기(시간 8)로 $T_3 \rightarrow T_2$가 생긴다.

S_2의 그래프에는 순환이 없고 **$T_3 \rightarrow T_1 \rightarrow T_2$**의 직렬 순서가 모든 간선을 만족한다. 따라서 직렬 가능한 것은 S_2만이므로 ②이다.

NoteBase

문 36. 관계형 DBMS가 SQL 질의를 처리하는 단계를 순서대로 나열한 것으로 가장 적절한 것은?

가. 질의를 파싱하고 문법과 의미를 검증한다.
나. 여러 실행 방법의 비용을 비교해 최적의 실행 계획을 생성한다.
다. 질의를 관계 대수식(질의 트리)으로 변환한다.
라. 실행 계획에 따라 연산을 수행하여 결과를 반환한다.

① 가 → 나 → 다 → 라
② 가 → 나 → 라 → 다
③ 가 → 다 → 나 → 라
④ 나 → 가 → 다 → 라

정답 3번

간단한 해설

실행 방법의 비용을 비교하려면 먼저 질의를 **관계 대수식으로 표현**해야 한다. 따라서 다는 나보다 앞서며, 실제 결과를 반환하는 라는 실행 계획이 정해진 뒤에 온다.

올바른 순서

가 → 다 → 나 → 라

가. 먼저 SQL 문장을 파싱하여 구조를 분석하고 **문법과 의미를 검증**한다. 구문의 형식뿐 아니라 참조하는 테이블·속성 등이 유효한지도 확인해야 이후 처리에 사용할 수 있다.

다. 검증된 질의를 **관계 대수식 또는 질의 트리**로 변환한다. 선택·투영·조인 등 어떤 논리 연산이 필요한지를 표현하는 단계로, 구체적인 실행 방법의 비용을 비교하기 위한 바탕이 된다.

나. 질의의 논리적 표현을 대상으로 가능한 실행 방법을 검토하고 **비용을 비교하여 실행 계획**을 선택한다. 조인 순서나 접근 방법 등을 결정하므로, 관계 대수식으로 변환하는 단계보다 뒤에 온다.

라. 선택된 실행 계획에 따라 실제 연산을 수행하고 질의 결과를 반환한다. 실행할 계획이 먼저 정해져야 하므로 최적화 이후의 마지막 단계이다.

NoteBase

문 37. DBMS 유형에 따른 특징으로 가장 적절하지 않은 것은?

① 관계형 데이터베이스는 테이블 간 참조관계를 통해 데이터 간의 연결을 관리할 수 있다.

② NoSQL 데이터베이스는 정형, 반정형 데이터를 분산 관리할 수 있다.

③ 벡터 데이터베이스는 ACID 특성을 보장하면서 시스템 확장성을 제공할 수 있다.

④ 그래프 데이터베이스는 데이터를 노드와 엣지로 표현하여 연결된 정보를 효율적으로 저장하고 탐색할 수 있다.

정답 3번

간단한 해설

ACID를 보장하면서 분산 확장성을 추구하는 것은 **NewSQL의 대표 특징**이다. 벡터 데이터베이스는 데이터를 수치 벡터로 저장하여 **유사도 검색**을 수행하는 데 특화되며 ACID 보장은 벡터 데이터베이스라는 유형만으로 정해지지 않는다.

출제 포인트

DBMS의 데이터 표현 방식과 핵심 기능을 구분하고, 벡터 데이터베이스와 NewSQL의 대표 특징을 비교하는 문제이다.

핵심 개념

관계형 데이터베이스: 데이터를 행과 열로 이루어진 테이블에 저장하며, 외래 키로 테이블 사이의 참조관계를 관리한다.
NoSQL 데이터베이스: 문서, 키-값 등 다양한 데이터 모델을 사용하며, 정형·반정형 데이터 등을 여러 서버에 분산하여 관리할 수 있다.
벡터 데이터베이스: 데이터의 특징을 수치의 묶음인 벡터로 저장하고, 벡터 사이의 유사도를 이용해 비슷한 데이터를 찾는다. 예를 들어 문서를 벡터로 표현하면 검색 문장과 의미가 비슷한 문서를 찾는 데 활용할 수 있다.
그래프 데이터베이스: 대상을 노드로, 대상 사이의 관계를 엣지로 표현하여 연결관계를 탐색한다.
NewSQL: 관계형 데이터베이스의 트랜잭션 보장과 분산 환경의 확장성을 함께 추구하는 DBMS이다. ACID는 트랜잭션의 원자성·일관성·격리성·지속성을 뜻한다.

NoteBase

선지별 해설

① O
관계형 데이터베이스에서는 외래 키가 다른 테이블의 키를 참조한다. 예를 들어 주문 테이블의 고객번호가 고객 테이블의 고객번호를 참조하면 주문과 고객의 연결을 관리할 수 있다.

② O
NoSQL은 반정형 데이터뿐 아니라 구조가 일정한 정형 데이터도 저장할 수 있다. 데이터를 여러 서버에 나누어 저장하는 분산 구성을 지원할 수 있으므로 선지의 설명은 적절하다.

③ X
'ACID 특성을 보장하면서 시스템 확장성을 제공'한다는 설명은 DBMS 유형을 구분할 때 NewSQL의 대표 특징에 해당한다. 벡터 데이터베이스의 대표 특징은 수치 벡터의 저장과 유사도 검색이다. 따라서 선지의 '벡터 데이터베이스'를 'NewSQL 데이터베이스'로 고치면 유형과 특징이 맞는다. 벡터 저장·검색 기능과 ACID 보장은 별개의 속성이므로, 특정 구현에서 함께 지원할 수 있다는 사실과 유형 자체의 대표 특징은 구분한다.

④ O
그래프 데이터베이스의 노드는 사람·상품 같은 대상을, 엣지는 대상 사이의 관계를 나타낸다. 엣지를 따라 탐색하면 친구 관계나 거래 관계처럼 연결된 정보를 효율적으로 찾을 수 있다.

오답 포인트

③의 '확장성'만 보고 벡터 데이터베이스의 특징으로 판단하기 쉽다. 여러 DBMS가 확장성을 지원할 수 있지만, 'ACID 보장과 분산 확장성의 결합'은 NewSQL을 구별하는 대표 설명이다. 또한 NoSQL이 정형 데이터를 저장할 수 없다고 오해하면 ②를 잘못 고르게 된다.

다시 나오면 이렇게 판단

데이터를 어떻게 표현하고 무엇을 중심으로 검색하는지 먼저 확인한다. 테이블과 외래 키는 관계형, 유연한 데이터 모델과 분산 관리는 NoSQL, 벡터 간 유사도 검색은 벡터, 노드와 엣지를 통한 관계 탐색은 그래프에 연결한다. ACID 보장과 분산 확장성을 함께 강조하면 NewSQL의 특징인지 확인한다.

암기 포인트

관계형은 테이블 참조, NoSQL은 유연한 모델과 분산, 벡터는 유사도, 그래프는 연결 탐색, NewSQL은 ACID와 확장성이다.

문 38. 다음 중 NoSQL 데이터 모델에 대한 옳은 설명을 모두 고른 것으로 가장 적절한 것은?

가. 키-값(key-value) 저장소는 단순한 키로 값을 빠르게 조회하는 데 적합하다.
나. 문서(document) 저장소는 JSON, XML 같은 반정형 문서를 저장하고 질의한다.
다. 그래프(graph) 데이터베이스는 개체 간의 관계(연결)를 표현하고 탐색하는 데 강점이 있다.
라. 칼럼 패밀리(column-family) 저장소는 행 단위로만 저장하여 열 기반 집계에 불리하다.

① 가, 라
② 나, 다
③ 다, 라
④ 가, 나, 다

정답 4번

간단한 해설

키-값·문서·그래프 저장소의 특징을 설명한 가·나·다는 옳다. 칼럼 패밀리 저장소를 **행 단위로만 저장하는 방식**으로 한정한 라는 틀리다.

보기별 해설

가 (O). 키-값 저장소는 **키를 식별자로 사용하여 대응하는 값을 조회**한다. 복잡한 테이블 간 조인보다 키를 알고 있는 항목의 빠른 저장·조회에 적합하다.

나 (O). 문서 저장소는 JSON·XML 등과 같은 구조를 가진 문서를 저장하고, 문서의 필드나 속성을 대상으로 질의할 수 있다. 문서마다 구조가 달라질 수 있는 **반정형 데이터**를 다루는 데 적합하다.

다 (O). 그래프 데이터베이스는 개체를 노드, 관계를 엣지로 나타낸다. 관계를 따라가는 탐색을 통해 연결 정보나 경로를 처리하는 데 강점이 있다.

라 (X). 칼럼 패밀리 저장소는 관련 열들을 칼럼 패밀리로 묶어 관리하며, 필요한 열이나 열 묶음 중심으로 데이터를 다룰 수 있다. 따라서 **행 단위로만 저장한다는 단정이 틀리다**. 다만 칼럼 패밀리 방식과 분석용 칼럼 지향 저장소는 같은 개념이 아니며, 모든 구현에서 열 기반 집계가 항상 빠르다고 단정하는 것도 적절하지 않다.

옳은 설명은 **가, 나, 다**이므로 ④이다.

NoteBase

문 39. 다음 분류 모델의 혼동 행렬(confusion matrix)에서 정밀도(precision)와 재현율(recall)로 가장 적절한 것은?

< 혼동 행렬 >

실제 \ 예측	양성	음성
양성	60	40
음성	20	80

① 정밀도: 0.6, 재현율: 0.6

② 정밀도: 0.6, 재현율: 0.75

③ 정밀도: 0.75, 재현율: 0.6

④ 정밀도: 0.75, 재현율: 0.75

정답 3번

간단한 해설

정밀도는 양성 예측 80개 중 참양성 60개의 비율로 **0.75**이다. 재현율은 실제 양성 100개 중 참양성 60개의 비율로 **0.6**이다.

NoteBase

계산 과정

행렬 읽기: 행은 실제 분류, 열은 예측 분류이다. 실제 양성을 양성으로 예측한 참양성(TP)은 60, 실제 음성을 양성으로 예측한 거짓양성(FP)은 20이다. 실제 양성을 음성으로 예측한 거짓음성(FN)은 40, 실제 음성을 음성으로 예측한 참음성(TN)은 80이다.

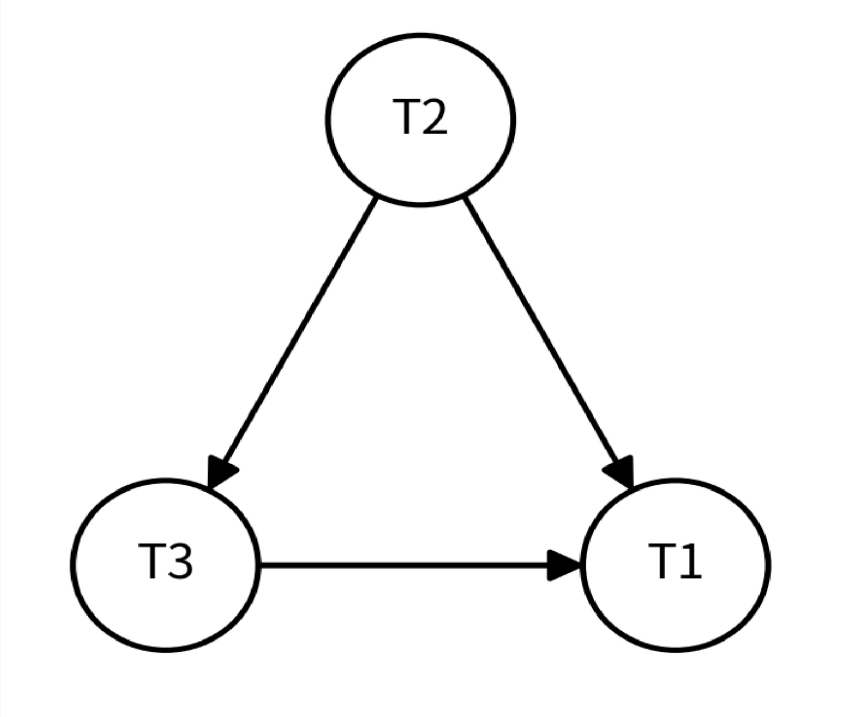
정밀도: **양성으로 예측한 것 중 실제 양성인 비율**이다. 양성 예측은 참양성 60과 거짓양성 20을 합한 80이므로, 정밀도는 $\frac{TP}{TP+FP} = \frac{60}{60+20} = 0.75$이다.

재현율: **실제 양성 중 양성으로 찾아낸 비율**이다. 실제 양성은 참양성 60과 거짓음성 40을 합한 100이므로, 재현율은 $\frac{TP}{TP+FN} = \frac{60}{60+40} = 0.6$이다.

따라서 정밀도 0.75, 재현율 0.6을 제시한 ③이 적절하다. 참음성 80은 두 지표의 분모에 들어가지 않으며, 예측 양성과 실제 양성 중 어느 집합을 기준으로 삼는지를 구별해야 한다.

NoteBase

문 40. 다음 선행 그래프(precedence graph)가 표현하는 충돌 직렬 가능한(conflict-serializable) 스케줄에 대한 설명으로 가장 적절한 것은? (단, 노드는 트랜잭션이고 화살표는 선행 관계이다.)



① 그래프에 사이클이 있어 충돌 직렬 가능하지 않다.

② 충돌 직렬 가능하지만 등가 직렬 순서는 유일하지 않다.

③ 충돌 직렬 가능하며 등가 직렬 순서는 T1 → T3 → T2이다.

④ 충돌 직렬 가능하며 등가 직렬 순서는 T2 → T3 → T1이다.

정답 4번

간단한 해설

선행 그래프에는 방향 순환이 없어 **충돌 직렬 가능**하다. 모든 선행 관계를 만족하는 유일한 직렬 순서는 $\mathbf{T_2 \rightarrow T_3 \rightarrow T_1}$이다.

NoteBase

계산 과정

화살표는 앞 트랜잭션이 뒤 트랜잭션보다 먼저 실행되어야 함을 뜻한다. 그림의 선행 관계는 $\mathbf{T_2 \rightarrow T_3}$, $\mathbf{T_2 \rightarrow T_1}$, $\mathbf{T_3 \rightarrow T_1}$이다.

처음에 들어오는 간선이 없는 노드는 T_2뿐이므로 T_2를 먼저 둔다. T_2와 그 간선을 제거하면 T_3만 진입 차수가 0이 되므로 T_3를 다음에 두고, 마지막에 T_1을 둔다.

모든 노드를 이러한 위상 정렬로 나열할 수 있으므로 그래프에는 방향 순환이 없고 **충돌 직렬 가능**하다. T_2가 T_3보다 앞서고 T_3가 T_1보다 앞서야 하므로 등가 직렬 순서도 유일하다.

따라서 등가 직렬 순서는 $\mathbf{T_2 \rightarrow T_3 \rightarrow T_1}$이며 ④가 적절하다. 삼각형 모양이라는 이유만으로 사이클이 있는 것은 아니며, 화살표 방향을 따라 출발점으로 돌아올 수 있어야 사이클이다.

NoteBase