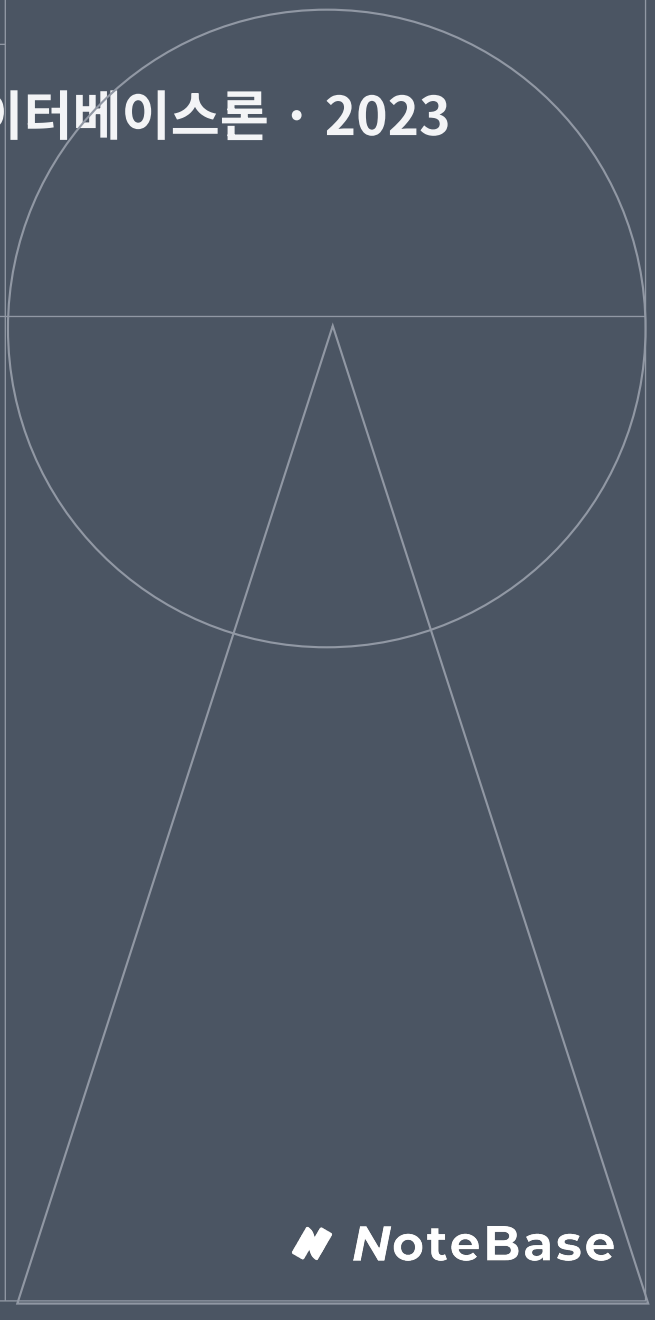
기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2023



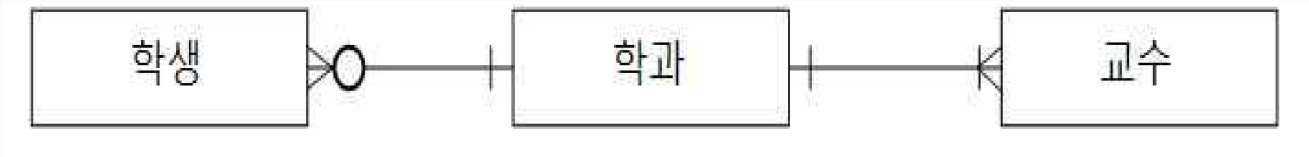
NoteBase

기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2023

공무원/경찰/경력채용 · 데이터베이스론 · 2023

문 1. 다음과 같이 설계된 ERD(Entity-Relationship Diagram)에 관한 설명으로 가장 적절하지 않은 것은?



① 학생은 두 학과에 소속될 수 있다.

② 학과에 속한 학생이 한 명도 없을 수 있다.

③ 학과에는 한 명 이상의 교수가 있다.

④ 교수는 한 학과에만 소속된다.

정답 1번

간단한 해설

관계선 끝의 기호는 반대편 개체 하나에 대응하는 개체 수를 뜻한다. 학과 쪽이 1로 표시되어 있으므로 **학생 한 명은 한 학과에만 소속**된다.

선지별 해설

① (X) 학생 한 명에 대응하는 학과의 수는 학과 쪽 끝의 막대로 표시된 1이다. 따라서 학생은 **한 학과에만 소속**되며, 두 학과에 동시에 소속될 수 있다는 설명은 도식과 다르다.

② (O) 학과 하나에 대응하는 학생 수는 학생 쪽의 원과 갈퀴 기호로 읽는다. 원은 최소 0, 갈퀴는 최대 다수를 뜻하므로 **학생이 없는 학과도 허용**된다.

③ (O) 학과 하나에 대응하는 교수 수는 교수 쪽의 막대와 갈퀴로 읽는다. 최소 1, 최대 다수이므로 학과에는 **한 명 이상의 교수**가 있어야 한다.

④ (O) 교수 한 명에 대응하는 학과 수는 학과 쪽 끝의 막대가 나타내는 1이다. 따라서 교수는 **한 학과에만 소속**된다.

NoteBase

문 2. 관계형 데이터베이스 키(key)에 관한 설명으로 가장 적절한 것은?

① 슈퍼키는 유일성과 최소성을 모두 가져야 한다.

② 후보키는 유일성만 가지고 최소성은 가지지 않는다.

③ 외래키는 NULL 값을 가질 수 있다.

④ 기본키는 키값의 변경이 자주 일어나는 속성으로 선정한다.

정답 3번

간단한 해설

슈퍼키는 유일성, 후보키는 **유일성과 최소성**을 갖는다. 기본키에는 NULL을 허용하지 않지만, 외래키는 별도 제약이 없다면 NULL을 가질 수 있다.

선지별 해설

① X: 슈퍼키는 튜플을 유일하게 식별할 수 있는 속성 집합이므로 유일성을 만족한다. 그러나 식별에 불필요한 속성이 포함될 수 있어 **최소성은 필수 조건이 아니다**. 최소성까지 만족하는 슈퍼키가 후보키이다.

② X: 후보키는 유일성과 최소성을 모두 만족한다. 최소성이란 속성 수가 모든 후보키 중 가장 작다는 뜻이 아니라, 그 후보키에서 어느 속성도 제거하면 더 이상 유일하게 식별할 수 없다는 뜻이다.

③ O: 외래키에는 별도로 NOT NULL 등의 제약이 없다면 **NULL이 허용될 수 있다**. NULL이 아닌 참조 값은 참조 대상 키와 일치해야 한다. 외래키가 동시에 기본키의 일부이거나 NOT NULL로 선언되면 NULL은 허용되지 않는다.

④ X: 기본키는 후보키 가운데 대표 식별자로 선택하며, 가능하면 값의 변경이 적은 안정적인 속성을 고르는 것이 좋다. 기본키 값이 자주 바뀌면 이를 참조하는 데이터도 처리해야 하므로, 자주 변경되는 속성을 우선 선정한다는 설명은 부적절하다.

NoteBase

문 3. 데이터베이스와 DBMS(Database Management System)에 관한 설명으로 옳은 것을 모두 고른 것은?

㉠ 데이터베이스는 동일한 데이터를 다수 사용자가 공유해서 사용할 수 있다.
㉡ 데이터베이스는 정적 변화, 주소에 의한 참조 특징을 가진다.
㉢ DBMS는 파일 시스템이 가진 데이터 종속성과 중복성 문제해결을 위해 사용한다.
㉣ DBMS는 정의 기능, 조작 기능, 제어 기능을 가진다.

① ㉠㉡㉣
② ㉠㉢㉣
③ ㉠㉡㉢
④ ㉡㉢㉣

정답 2번

간단한 해설

데이터베이스는 공통 데이터를 공유하며 **계속적인 변화와 내용에 의한 참조**를 특징으로 한다. DBMS는 정의·조작·제어 기능으로 종속성과 불필요한 중복을 줄이므로 ㉠·㉢·㉣이 옳다.

보기별 해설

㉠ O: 데이터베이스는 여러 응용 프로그램과 사용자가 **공통 데이터를 공유**할 수 있도록 관리한다. 동시 접근 시에는 권한과 동시성 제어를 통해 충돌이나 비인가 접근을 방지한다.

㉡ X: 데이터베이스는 삽입·삭제·갱신을 반영하는 **계속적인 변화와 내용에 의한 참조**를 특징으로 한다. 사용자는 보통 물리적 저장 주소가 아니라 원하는 데이터의 값이나 조건을 지정하므로, 정적 변화와 주소에 의한 참조라는 설명은 틀리다.

㉢ O: DBMS는 데이터의 구조·저장과 응용 프로그램을 분리하고 통합 관리하여 **데이터 종속성과 불필요한 중복**을 줄인다. 이는 파일 시스템의 주요 문제를 개선하려는 것이며 모든 중복을 무조건 없앤다는 뜻은 아니다.

㉣ O: **정의 기능**은 스키마·자료형·제약 조건을 정하고, **조작 기능**은 검색·삽입·삭제·갱신을 수행한다. **제어 기능**은 권한·무결성·동시성·복구 등을 관리하여 데이터를 안전하고 일관되게 유지한다.

따라서 옳은 설명은 ㉠·㉢·㉣이며 ②와 일치한다.

NoteBase

문 4. 3단계 데이터베이스 구조에 관한 설명으로 가장 적절한 것은?

① 외부스키마는 사용자가 생각하는 개인적 데이터베이스 구조로 하나만 존재한다.

② 개념스키마는 조직 관점의 데이터베이스 구조로 여러 개 존재한다.

③ 내부스키마는 데이터베이스가 저장되는 방법으로 개체, 관계, 제약조건, 접근권한, 보안정책 등에 대한 명세를 포함한다.

④ 데이터베이스의 논리적 구조나 물리적 구조 변경 시 응용 프로그램이 직접 영향을 받지 않는 구조이다.

정답 4번

간단한 해설

3단계 구조는 외부·개념·내부 스키마를 분리하고 사상으로 연결하여 **데이터 독립성**을 제공한다. 이를 통해 논리·물리 구조 변경이 응용 프로그램에 직접 영향을 주는 것을 줄인다.

선지별 해설

① X: 외부 스키마는 특정 사용자나 사용자 집단이 필요로 하는 데이터의 관점이다. 서로 다른 업무 요구에 맞추어 **여러 외부 스키마**가 존재할 수 있으므로 하나만 존재한다는 설명은 틀리다.

② X: 개념 스키마는 조직 전체 관점에서 데이터베이스의 통합된 논리 구조를 정의한다. 3단계 구조에서는 **하나의 개념 스키마**를 중심으로 여러 외부 스키마를 연결한다.

③ X: 내부 스키마는 파일 구성, 레코드의 물리적 표현, 인덱스와 접근 경로 등 **물리적 저장 구조**를 다룬다. 선지에 열거된 개체·관계·논리적 제약과 조직 수준의 접근권한·보안정책 명세는 개념 수준의 설명과 혼동한 것이다.

④ O: 외부·개념·내부 수준 사이의 사상을 이용하여 **논리적·물리적 데이터 독립성**을 제공하는 구조이다. 개념 스키마의 변경을 외부 관점에 직접 전파하지 않는 것이 논리적 독립성이고, 내부 저장 구조의 변경을 개념 수준에 직접 전파하지 않는 것이 물리적 독립성이다. 필요한 사상을 조정하여 기존 응용 인터페이스를 유지한다는 뜻이지, 어떤 의미 변경도 무조건 응용 수정 없이 가능하다는 뜻은 아니다.

따라서 3단계 구조가 제공하는 데이터 독립성을 설명한 ④가 가장 적절하다.

NoteBase

문 5. 다음 중 인덱스에 관한 설명으로 옳은 것을 모두 고른 것은?

㉠ 인덱스는 레코드 접근을 위해 해당 인덱스를 찾고, 인덱스가 가리키는 주소를 사용하여 레코드에 접근한다.
㉡ 인덱스는 키값과 주소의 쌍으로 구성된다.
㉢ 인덱스는 한 단계로 구성되며 다단계 인덱스를 사용할 수 없다.
㉣ 인덱스를 사용한 데이터 접근방법은 인덱스를 사용하지 않은 경우보다 항상 빠른 접근이 가능하다.

① ㉠㉡
② ㉠㉢
③ ㉡㉢
④ ㉡㉣

정답 1번

간단한 해설

인덱스는 검색 키와 데이터 위치를 연결하며 다단계로 구성할 수 있다. 다만 추가 탐색 비용 때문에 **인덱스 사용이 항상 전체 탐색보다 빠르지는 않다**.

보기별 해설

㉠ (O) 일반적인 인덱스 기반 레코드 접근에서는 검색 키에 해당하는 인덱스 항목을 찾고, 그 항목의 위치 정보를 따라 실제 레코드에 접근한다. 핵심은 **인덱스 탐색 후 레코드 위치를 참조**하는 것이다. 필요한 값을 인덱스만으로 얻는 실행 방식도 있지만, 이 문장은 기본적인 레코드 접근 절차를 설명한다.

㉡ (O) 기본적인 인덱스 항목은 **검색 키값과 위치 정보의 쌍**으로 구성된다. 위치 정보는 구현에 따라 레코드나 데이터 블록 등을 가리키며, 키값으로 찾은 데이터를 실제 저장 위치와 연결한다.

㉢ (X) 인덱스는 한 단계로만 제한되지 않는다. 하위 인덱스 블록을 찾기 위한 상위 인덱스를 두는 **다단계 인덱스**를 사용할 수 있으며, B+ 트리도 계층적 탐색 구조의 대표적인 예이다.

㉣ (X) 인덱스를 먼저 탐색하고 데이터를 추가로 읽는 비용이 있으므로, 작은 테이블이나 대부분의 행을 읽는 질의에서는 전체 순차 탐색이 더 유리할 수 있다. 따라서 인덱스가 **항상 더 빠른 것은 아니다**.

옳은 보기는 ㉠과 ㉡이므로 ①이다.

NoteBase

문 6. 확장 ER 모델에 관한 설명으로 가장 적절하지 않은 것은?

① 세분화는 일반적인 개념에서 구체적인 하위개념을 구성하는 개념이다.

② 일반화는 원래의 개체 타입을 슈퍼 클래스로 가지는 서브 클래스를 정의하는 개념이다.

③ 집단화는 요소 객체들을 가지고 상위의 복합객체를 구축하는 개념이다.

④ 연관화는 독립된 클래스로부터 객체들을 연관시켜 하나의 상위 레벨 객체로 추상화하는 개념이다.

정답 2번

간단한 해설

세분화는 상위 타입에서 하위 타입을 나누는 과정이고, 일반화는 여러 타입의 공통 특성으로 상위 타입을 만드는 과정이다. 두 과정의 방향을 구별해야 한다.

선지별 해설

① O: 세분화 또는 특수화는 기존의 일반적인 상위 개체 타입에서 구별되는 특성을 가진 하위 타입을 나누는 과정이다. 예를 들어 직원에서 관리자와 기술자를 정의하는 것처럼 **상위에서 하위로 구체화**한다.

② X: 기존 개체 타입을 슈퍼클래스로 두고 그 아래 서브클래스를 정의하는 것은 세분화이다. 일반화는 여러 개체 타입의 공통 특성을 추출하여 **공통 슈퍼클래스를 구성**하는 반대 방향의 과정이다. 예를 들어 승용차와 화물차를 공통 차량 타입으로 묶는다.

③ O: 집단화는 구성 요소 객체들을 하나의 복합객체로 추상화하는 것이다. 예를 들어 엔진·바퀴 등의 구성 요소로 자동차 전체를 표현하는 부분과 전체의 관점에 해당한다.

④ O: 연관화는 서로 독립된 클래스의 객체들을 연결하여 그 연관을 하나의 상위 수준 추상화 대상으로 다루는 것이다. 이는 공통 특성을 묶어 상속 관계를 만드는 일반화와 다르며, 연관이 없어져도 참여 객체들이 독립적으로 존재할 수 있다.

NoteBase

문 7. ERD를 관계 스키마로 사상하는 방법에 관한 설명으로 가장 적절하지 않은 것은?

① 다대다(N:M) 관계는 새로운 관계를 생성하고, 새로운 관계의 기본키는 기존 두 관계의 기본키를 결합한 복합키를 사용한다.

② 일대다(1:N) 관계는 N측 개체의 기본키를 가져와 1측 개체 릴레이션에 포함시키고 외래키로 지정한다.

③ 일반화를 사용한 ER 모델의 하위개체는 상위 개체의 기본키를 하위 개체의 기본키로 재사용한다.

④ 다중값 속성은 새로운 릴레이션을 생성하고, 생성된 릴레이션의 기본키는 다중값 속성의 기본키와 원래 개체의 기본키 값으로 구성한다.

정답 2번

간단한 해설

일대다 관계에서는 **1측 기본키를 N측의 외래키**로 넣는다. 각 N측 행이 자신과 연결된 1측 개체를 참조하게 해야 하므로, ②의 설명은 키를 가져오는 방향과 저장 위치가 반대이다.

선지별 해설

① O: 두 개체 사이의 다대다 관계는 연결 릴레이션을 따로 만들어 표현한다. 표준 사상에서는 양쪽 개체의 기본키를 각각 외래키로 포함하고, **두 키의 조합**을 연결 릴레이션의 기본키로 삼아 개체 쌍을 식별한다. 예를 들어 학생과 과목 사이의 수강 관계는 학생번호·과목번호의 조합으로 식별할 수 있다.

② X: 외래키를 넣는 방향이 반대이다. 일대다 관계에서는 **1측의 기본키를 N측 릴레이션에 외래키로** 넣는다. 부서 1개에 사원이 여러 명 소속된다면 각 사원 행에 부서번호를 저장하여 소속 부서를 참조한다. N측 키를 1측에 넣는 방식으로는 한 부서의 여러 사원을 하나의 단일값 외래키로 표현할 수 없다.

③ O: 상위 개체와 하위 개체를 각각 릴레이션으로 사상하는 방식에서는 하위 개체도 같은 실제 대상을 나타내므로 상위 개체의 기본키를 재사용한다. 하위 릴레이션에서 이 키는 기본키이면서 상위 릴레이션을 참조하는 외래키가 된다. 다른 사상 전략도 있지만, 이 설명은 상·하위 릴레이션을 분리하는 표준 방식에 해당한다.

④ O(표준 사상 방식의 취지): 다중값 속성은 별도 릴레이션에 한 값씩 저장하고, **원래 개체의 기본키와 다중값 속성의 값**을 결합하여 각 행을 식별한다. 예를 들어 사원 전화번호는 사원번호·전화번호의 조합을 기본키로 삼는다. 원문의 '다중값 속성의 기본키'는 엄밀한 용어가 아니며, 다중값 속성 자체에 별도 기본키가 있다는 뜻으로 이해해서는 안 된다. 여기서는 해당 속성값을 키의 구성 요소로 사용한다는 취지로 구분해야 한다.

NoteBase

문 8. 정규화(normalization)에 관한 설명으로 가장 적절하지 않은 것은?

① 하나의 비구조적인 관계형 스키마를 두 개 이상의 구조적인 관계형 스키마로 분해하는 과정이다.

② 구조화된 릴레이션은 데이터의 중복성을 최소화하여 삽입, 삭제, 갱신 이상이 발생하지 않는 릴레이션이다.

③ 정규화 과정은 어떤 릴레이션 스키마가 어떤 정규형을 만족하는지 확인하는 일련의 테스트이다.

④ 함수적 종속성은 정규형과는 관련이 없다.

정답 4번

간단한 해설

정규화는 종속성을 분석하여 중복과 갱신 이상을 줄이는 설계 과정이다. 특히 **함수적 종속성은 2NF·3NF·BCNF의 판정 기준**이므로 정규형과 무관하다는 ④가 틀리다.

선지별 해설

① O: 여기서 비구조적이라는 표현은 관계형 스키마가 아예 없다는 뜻이 아니라 **중복과 이상을 일으키는 바람직하지 않은 구조**를 가리킨다. 정규화는 종속성을 분석하여 필요한 경우 이를 더 잘 구성된 여러 스키마로 무손실 분해하는 과정이다. 이미 목표 정규형을 만족하면 추가 분해가 필요하지 않다.

② O: 잘 구조화된 릴레이션이라는 용어는 **불필요한 중복과 그로 인한 삽입·삭제·갱신 이상을 없애도록 설계한 릴레이션**을 뜻한다. 다만 어느 낮은 정규형 하나를 만족했다는 사실만으로 모든 종류의 이상이 사라진다고 일반화해서는 안 된다. 적용한 정규형과 고려한 종속성에 따라 제거되는 문제가 달라진다.

③ O: 정규화에서는 후보키와 종속성 등을 확인하여 스키마가 **각 정규형의 조건을 만족하는지 검사**한다. 만족하지 않으면 필요한 분해를 수행하고 분해된 스키마를 다시 검사한다.

④ X: 함수적 종속성은 **2NF·3NF·BCNF 등을 판정하는 핵심 기준**이다. 예를 들어 부분 함수 종속은 2NF, 이행적 종속의 문제는 3NF와 관련되며, BCNF는 모든 비자명한 함수 종속의 결정자가 슈퍼키인지 확인한다. 따라서 정규형과 관련이 없다는 설명은 틀리다.

NoteBase

문 9. 질의 최적화 방법 중 경험적 최적화에 관한 설명으로 가장 적절하지 않은 것은?

① 선택 연산을 먼저 수행하도록 처리한다.

② 인덱스는 사용하지 않도록 처리한다.

③ 카티션곱과 선택 연산을 결합하여 조인 연산으로 변환한다.

④ 연산의 중간 결과 데이터 양을 줄이는 연산을 먼저 수행하도록 처리한다.

정답 2번

간단한 해설

경험적 최적화는 조기 선택, 조인 변환, 중간 결과 축소 등의 규칙을 활용한다. **인덱스 사용을 일괄 배제**하는 것은 적절한 최적화 규칙이 아니다.

선지별 해설

① O: 결과의 의미를 보존할 수 있는 범위에서 선택 연산을 아래로 내려 **가능한 일찍 행을 걸러 내는 것**은 대표적인 경험적 규칙이다. 이후 조인 등이 처리할 튜플 수를 줄일 수 있다. 모든 조건을 어떤 연산보다도 무조건 앞당길 수 있다는 뜻은 아니다.

② X: 경험적 최적화가 **인덱스 사용을 배제하는 것은 아니다**. 조건에 맞는 인덱스가 있으면 검색이나 조인의 접근 경로로 활용할 수 있다. 인덱스가 항상 유리한 것도 아니지만, 사용하지 않도록 하는 것을 일반적인 최적화 규칙으로 삼지는 않는다.

③ O: 카티션곱 뒤에 두 릴레이션 사이의 조인 조건을 검사하는 선택 연산이 있으면, 이를 **조건 조인으로 결합**할 수 있다. 거대한 카티션곱 결과를 먼저 생성하는 비효율을 피하는 방향의 변환이다.

④ O: 선택이나 불필요한 열을 제거하는 프로젝트 등으로 **중간 결과의 크기를 줄이면** 후속 연산의 입출력·메모리·계산 부담을 줄일 수 있다. 단, 최종 결과와 후속 연산에 필요한 열·조건을 보존하는 동치 변환이어야 한다.

따라서 인덱스를 사용하지 않도록 한다는 ②가 가장 적절하지 않다.

NoteBase

문 10. 다음 직원테이블과 부서테이블에 관한 수행 연산과 수행 시 위배되는 무결성 제약조건으로 가장 적절한 것은? (단, 밑줄 속성은 기본키이고, 직원 테이블의 부서번호 속성은 부서 테이블의 부서번호 속성을 외래키로 참조함)

직원

사번	이름	부서번호
230101	A	01
230102	B	01
230201	A	02
230301	C	03

부서

부서번호	부서이름
01	경무과
02	교육과
03	기획과

① 테이블: 직원, 연산: INSERT, 튜플: (230304, D, 04), 위배 제약조건: 개체 무결성

② 테이블: 부서, 연산: INSERT, 튜플: (04, 정보통신과), 위배 제약조건: 개체 무결성

③ 테이블: 직원, 연산: DELETE, 튜플: (230101, A, 01), 위배 제약조건: 참조 무결성

④ 테이블: 부서, 연산: DELETE, 튜플: (03, 기획과), 위배 제약조건: 참조 무결성

정답 4번

간단한 해설

참조 무결성은 외래키의 참조 대상이 유지되어야 한다는 조건이다. **직원이 참조 중인 부서만 삭제하면 대응 부서가 사라진다.**

NoteBase

선지별 해설

① (X) 새 사번 230304는 기존 사번과 중복되지 않고 NULL도 아니므로 개체 무결성 위반이 아니다. 문제는 직원의 부서번호 04에 대응하는 부서가 없다는 점이다. 따라서 위배되는 것은 **참조 무결성**이다.

② (X) 부서 테이블의 기본키는 부서번호이고 기존 값은 01·02·03이다. 새 부서번호 04는 중복도 NULL도 아니므로 **개체 무결성을 위배하지 않는다**. 직원을 배치하기 전에 부서를 먼저 등록하는 것도 가능하다.

③ (X) 직원은 부서를 참조하는 쪽이다. 직원 (230101, A, 01)을 삭제해도 나머지 직원들의 부서번호는 기존 부서와 연결되어 있으므로 **제시된 외래키의 참조 무결성 위반은 없다**. 부서 01에 직원 230102가 남아 있다는 사실과도 모순되지 않는다.

④ (O) 부서 03은 직원 230301이 참조하고 있다. 이 직원을 그대로 둔 채 부서 (03, 기획과)만 삭제하면 직원의 외래키가 가리킬 부서가 없어져 **참조 무결성을 위배**한다. 연쇄 삭제나 NULL 설정 같은 별도 조치가 제시되지 않은 이 문제에서는 해당 삭제와 제약조건의 연결이 옳다.

NoteBase

문 11. 다음 릴레이션 R과 S에서 실행 가능한 관계대수 연산들을 짝지은 것으로 가장 적절한 것은?

릴레이션	R (A, B, C) S (B, D)
제약조건	밑줄 속성은 기본키 릴레이션 R과 S의 모든 속성들의 도메인은 동일

① 합집합, 교집합

② 교집합, 선택 연산

③ 자연조인, 카티션곱

④ 합집합, 동등조인

정답 3번

간단한 해설

합집합·교집합은 두 릴레이션의 **속성 수와 대응 도메인**이 맞아야 한다. R과 S는 속성 수가 다르지만 공통 속성 B를 통한 자연조인과 모든 튜플 쌍을 결합하는 카티션곱은 가능하다.

핵심 풀이

R은 A·B·C의 세 속성, S는 B·D의 두 속성을 가진다. 합집합과 교집합에는 합병 가능성이 필요하므로 대응 속성의 도메인뿐 아니라 **속성 수도 같아야 한다**. 모든 속성의 도메인이 같다는 조건만으로는 부족하며, 주어진 R과 S를 그대로 합집합 또는 교집합할 수 없다.

자연조인은 공통 속성 B의 값이 같은 튜플을 연결하므로 실행 가능하다. 중복된 B는 하나만 남아 결과의 속성은 A·B·C·D가 된다.

카티션곱도 두 릴레이션의 모든 튜플 쌍을 결합할 수 있으며, 두 B는 R.B와 S.B처럼 구별하거나 이름을 바꾸어 표현한다. 결과에는 양쪽 속성 다섯 개가 유지된다.

따라서 두 연산이 모두 가능한 조합은 **자연조인과 카티션곱**인 ③이다. 선택 연산은 각 릴레이션에 적용할 수 있고 동등조인도 가능하지만, 이를 짝지은 ②에는 불가능한 교집합이, ④에는 불가능한 합집합이 포함되어 있다.

NoteBase

문 12. NoSQL 시스템에 관한 설명으로 가장 적절하지 않은 것은?

① 일관성보다는 가용성과 확장성에 중점을 둔다.

② 데이터 구조를 명세하기 위해 스키마를 포함한다.

③ SQL과 같은 강력한 질의를 필요로 하지 않는다.

④ 구조와 특성에 따라 문서기반, 키-값 저장소, 칼럼 기반, 그래프 기반의 NoSQL 시스템으로 분류한다.

정답 2번

간단한 해설

NoSQL의 일반적 특징은 관계형 데이터베이스처럼 데이터 구조를 미리 고정하기보다 **유연한 스키마를 사용한다**는 점이다. 따라서 구조를 명세하는 스키마의 포함을 NoSQL의 일반적 특징으로 제시한 설명이 가장 적절하지 않으며, 이는 선택적인 스키마 검증 기능까지 불가능하다는 뜻은 아니다.

선지별 해설

① O: 전통적인 NoSQL 시스템은 대규모 데이터를 여러 서버에 분산하여 처리하고 서버를 추가하는 방식으로 확장하는 데 중점을 둔다. 이를 위해 즉각적인 강한 일관성의 일부를 완화하고 **가용성과 수평적 확장성을 우선**하는 설계를 널리 사용한다. 다만 모든 NoSQL 제품이 강한 일관성을 지원하지 않는다는 뜻은 아니다.

② X: 이 문항에서 비교하는 일반적 특성은 관계형 데이터베이스의 사전 정의된 구조와 NoSQL의 유연한 데이터 모델이다. NoSQL은 저장되는 데이터마다 필드나 자료형이 달라질 수 있는 **스키마리스 또는 유연한 스키마**를 특징으로 하므로, 데이터 구조 명세를 위한 스키마의 포함을 일반적 특징으로 내세우는 것은 적절하지 않다. 올바른 설명은 고정된 스키마를 전제하지 않고 데이터 구조를 유연하게 수용한다는 것이다. 일부 제품의 선택적인 스키마 검증이나 애플리케이션 차원의 구조 설계까지 배제하는 것은 아니다.

③ O: NoSQL은 관계형 데이터베이스의 SQL처럼 복잡한 조인 등을 제공하는 범용 질의 언어를 필수 전제로 하지 않는다. 키를 통한 조회, 문서 조건 검색, 그래프 탐색 등 **각 데이터 모델에 맞는 접근 방식**을 사용할 수 있다. SQL과 유사한 언어나 강력한 질의 기능을 지원하는 제품도 있으므로, 그러한 기능을 사용할 수 없다는 의미는 아니다.

④ O: NoSQL은 데이터 모델에 따라 문서 기반, 키-값 저장소, 칼럼 기반, 그래프 기반 등으로 분류한다. 문서 기반은 구조화된 문서 단위로, 키-값 저장소는 키와 대응 값으로 데이터를 다룬다. 칼럼 기반은 칼럼 또는 칼럼 패밀리 중심으로 데이터를 조직하고, 그래프 기반은 노드와 관계를 중심으로 연결된 데이터를 표현한다.

문 13. 다음 데이터베이스 보안에 관한 설명으로 옳은 것을 모두 고른 것은?

㉠ 데이터베이스 보안은 권한을 부여받은 사용자에게만 데이터를 이용할 수 있도록 한다.
㉡ GRANT, REVOKE 등의 명령어를 이용한 접근 권한 관리 기법은 임의 접근제어 기법 중 하나이다.
㉢ 뷰를 이용한 접근제어는 강제 접근제어 기법 중 하나이다.
㉣ 추론제어는 숨겨진 통로와 같이 정보가 흘러서 권한을 갖지 않는 사용자들에게 도달하는 것을 방지하는 제어수단이다.

① ㉠㉡
② ㉡㉣
③ ㉠㉢
④ ㉢㉣

정답 1번

간단한 해설

권한에 따른 접근 통제와 GRANT·REVOKE를 통한 임의 접근제어에 관한 설명은 옳다. 뷰 사용 자체가 강제 접근제어인 것은 아니며, 은닉 채널을 통한 정보 이동의 통제는 추론제어가 아니라 **흐름제어**이다.

보기별 해설

㉠ O: 데이터베이스 보안은 **허가된 사용자만 허가된 범위의 데이터와 연산에 접근**하도록 통제한다. 사용자 인증과 접근 권한 관리는 이러한 비인가 접근 방지의 수단이다.

㉡ O: GRANT는 권한을 부여하고 REVOKE는 부여된 권한을 회수한다. 객체 소유자 등 권한을 가진 주체가 다른 사용자에게 권한을 부여·회수하는 방식은 **임의 접근제어(DAC)**에 해당한다.

㉢ X: 필요한 행·열만 보이는 뷰를 만들고 그 뷰에 이용 권한을 부여하는 것은 일반적인 임의 접근제어의 수단이다. **강제 접근제어(MAC)**는 보안 등급·레이블과 시스템 정책으로 접근을 강제하는 방식이다. 뷰를 MAC 환경에서도 사용할 수는 있지만, 뷰를 사용한다는 사실 자체가 MAC을 뜻하지 않는다.

㉣ X: 권한 없는 대상에게 정보가 흘러가는 것을 막고 은닉 채널을 통제하는 것은 **흐름제어**의 설명이다. 추론제어는 허용된 질의 결과나 공개된 정보를 결합하여 비공개 정보를 유추하지 못하도록 제한하는 것이다.

따라서 옳은 설명은 ㉠, ㉡이며, 이를 묶은 ①이 정답이다.

NoteBase

문 14. 테이블 S, T, U를 다음과 같이 운영할 수 있도록 하는 생성문에서 ㉠~㉣의 내용으로 가장 적절한 것은? (단, 밑줄 속성은 기본키이고, Tname 속성은 CHAR(64)로 한정함)

S

Sno	Sname
105	홍길동
…	…
128	김하늘

T

Tno	Tname
16	정수기
…	…
70	NULL

U

Sno	Tno
105	16
…	…
113	62

```
CREATE TABLE S (
        Sno (    ㉠    ) PRIMARY KEY,
        Sname CHAR(64) NOT NULL
      );
CREATE TABLE T (
        Tno INT(4) PRIMARY KEY,
        Tname (    ㉡    )
      );
CREATE TABLE U (
        Sno CHAR(4),
        Tno (    ㉢    ),
        PRIMARY KEY (    ㉣    ),
        FOREIGN KEY(Sno) REFERENCES S(Sno),
        FOREIGN KEY(Tno) REFERENCES T(Tno)
      );
```

① ㉠: CHAR(4), ㉡: CHAR(64) NOT NULL, ㉢: INT(4), ㉣: Sno, Tno

② ㉠: CHAR(4), ㉡: CHAR(64), ㉢: INT(4), ㉣: Sno, Tno

③ ㉠: INT(4), ㉡: CHAR(64) NOT NULL, ㉢: CHAR(4), ㉣: Tno, Sno

④ ㉠: INT(4), ㉡: CHAR(64), ㉢: CHAR(4), ㉣: Tno, Sno

정답 2번

간단한 해설

외래키의 자료형을 참조 대상 열과 맞추고, 예시의 NULL을 저장하려면 **Tname에 NOT NULL을 붙이지 않아야** 한다. U의 밑줄 친 Sno와 Tno는 두 열을 묶은 복합 기본키이다.

NoteBase

빈칸별 해설

㉠ CHAR(4): U의 Sno는 이미 CHAR(4)로 선언되어 있고 S(Sno)를 참조한다. 따라서 참조 대상인 S의 Sno도 이 외래키와 호환되도록 CHAR(4)로 선언한다. 표의 값이 105처럼 숫자로 보인다는 이유만으로 정수형을 선택하면 안 된다.

㉡ CHAR(64): 문제에서 Tname의 형식을 CHAR(64)로 한정했고, T의 예시에는 Tname이 NULL인 행이 있다. 따라서 **NOT NULL을 붙이지 않아야** 그 행을 저장할 수 있다. Tno가 기본키라는 사실은 별도 열인 Tname의 NULL까지 금지하지 않는다.

㉢ INT(4): U의 Tno는 T(Tno)를 참조하며, T의 Tno는 INT(4)로 선언되어 있다. 따라서 제시된 생성문의 표기에 맞춰 U의 Tno도 INT(4)로 둔다. 이와 같은 MySQL식 표기에서 괄호의 4를 정수의 자릿수 제한이나 저장 바이트 수로 해석하지 않는다.

㉣ Sno, Tno: U에서 밑줄 친 두 속성은 함께 **복합 기본키**를 이룬다. 따라서 PRIMARY KEY (Sno, Tno)로 선언하여 두 값의 조합을 유일하게 식별한다. 두 열을 각각 독립된 기본키로 선언하는 것은 아니다. 키의 나열 순서를 바꿔도 이 조합의 유일성 조건은 같으므로, 순서만으로 다른 선택지를 배제하지 않는다.

㉠ CHAR(4), ㉡ CHAR(64), ㉢ INT(4), ㉣ Sno, Tno가 모두 맞는 조합은 ②이다.

NoteBase

문 15. 데이터베이스에서 트랜잭션에 대한 동시성 제어(concurrency control)를 하지 않았을 때 발생하는 문제로 가장 적절하지 않은 것은?

① 갱신 손실(lost update)

② 임시 갱신(temporary update) 또는 오손 읽기(dirty read)

③ 반복할 수 없는 읽기(unrepeatable read)

④ 충돌 동치(conflict equivalence)

정답 4번

간단한 해설

갱신 손실·오손 읽기·반복할 수 없는 읽기는 동시 실행에 따른 이상 현상이다. 충돌 동치는 이상 현상이 아니라 두 실행 스케줄 사이의 관계를 나타내는 개념이다.

개념 구분

충돌 동치는 같은 트랜잭션들의 연산으로 구성된 두 스케줄에서 **충돌하는 모든 연산 쌍의 상대적 순서가 같은 관계**이다. 서로 다른 트랜잭션이 같은 데이터 항목에 접근하며 적어도 하나가 쓰기일 때 충돌한다. 충돌 동치는 스케줄의 성질을 비교하는 개념이지, 동시성 제어가 없어서 생기는 이상 현상 자체가 아니므로 ④가 해당한다.

갱신 손실은 다른 트랜잭션의 쓰기가 앞선 갱신을 덮어 그 효과가 사라지는 현상이다.

오손 읽기는 아직 커밋하지 않은 다른 트랜잭션의 값을 읽는 현상으로, 그 트랜잭션이 취소되면 유효하지 않은 값을 사용하게 된다.

반복할 수 없는 읽기는 같은 트랜잭션이 같은 데이터를 다시 읽었을 때 다른 트랜잭션의 커밋된 변경 때문에 값이 달라지는 현상이다.

이들은 동시 실행을 적절히 제어하지 않으면 발생할 수 있는 문제이다.

NoteBase

문 16. 다음 지역테이블과 제품테이블에 관한 수행 연산과 연산 결과 카디널리티로 가장 적절한 것은? (단, × : 카티션곱, ⟗_N : 자연조인, ⟕ : 왼쪽 외부조인, ⟖ : 오른쪽 외부조인 연산자임)

지역

지역번호	지역명
1	서울
2	경기
3	충청
4	경상

제품

제품번호	제품명	지역번호
1	B01	2
2	A01	1
3	B02	2
4	A02	1
5	C01	3
6	F01	6

① 연산: 지역 × 제품, 카디널리티: 10

② 연산: 지역 ⟗_N 제품, 카디널리티: 4

③ 연산: 지역 ⟕ 제품, 카디널리티: 5

④ 연산: 지역 ⟖ 제품, 카디널리티: 6

정답 4번

간단한 해설

지역번호로 연결되는 제품은 5개이며, 제품 F01은 대응 지역이 없다. 오른쪽 외부조인은 **미매칭 제품까지 보존하여 총 6행**을 만든다.

NoteBase

선지별 해설

① (X) 카티션곱은 지역의 각 행과 제품의 모든 행을 조합한다. 지역은 4행, 제품은 6행이므로 결과는 **4×6=24행**이며 10행이 아니다.

② (X) 자연조인은 공통 속성인 지역번호가 같은 행을 연결한다. 지역번호 1의 제품 2개, 2의 제품 2개, 3의 제품 1개가 연결되어 **2+2+1=5행**이다. 제품 F01의 지역번호 6은 지역 테이블에 없어 제외된다.

③ (X) 왼쪽 외부조인은 자연조인의 매칭 5행에 왼쪽 지역 테이블의 미매칭 행도 포함한다. 제품이 없는 지역번호 4(경상)에 대해 제품 속성을 NULL로 채운 1행을 추가하므로 **5+1=6행**이며 5행이 아니다.

④ (O) 오른쪽 외부조인은 오른쪽 제품 테이블의 모든 행을 유지한다. 매칭 5행에 지역번호 6을 가진 제품 F01의 미매칭 1행을 추가하므로 **5+1=6행**이다. 이 행의 대응 지역명 등은 NULL로 채워진다.

NoteBase

문 17. 다음 정규화와 관련된 ㉠~㉤의 내용으로 가장 적절한 것은?

· 비정규형 릴레이션에서 릴레이션의 속성이 모두 (㉠)이 되도록 릴레이션을 분해해서 얻은 릴레이션을 제1정규형 릴레이션이라 한다.
· 키에 대해 부분함수적 종속이 존재하는 릴레이션을 분해해서 키에 대해 (㉡)만이 존재하는 릴레이션을 제2정규형 릴레이션이라 한다.
· 키에 대해 (㉢)이 존재하는 릴레이션을 분해하여 키에 대해 (㉣)이 존재하지 않는 릴레이션을 제3정규형 릴레이션이라 한다.
· 릴레이션에서 결정자가 (㉤)가 아닌 함수적 종속을 제거하여, 모든 결정자가 (㉤)인 릴레이션을 BCNF 릴레이션이라 한다.

① ㉠: 다중값, ㉡: 이행함수적 종속, ㉢: 완전함수적 종속, ㉤: 기본키

② ㉠: 다중값, ㉡: 완전함수적 종속, ㉢: 이행함수적 종속, ㉤: 후보키

③ ㉠: 원자값, ㉡: 완전함수적 종속, ㉢: 이행함수적 종속, ㉤: 후보키

④ ㉠: 원자값, ㉡: 이행함수적 종속, ㉢: 완전함수적 종속, ㉤: 기본키

정답 3번

간단한 해설

1NF는 원자값, 2NF는 비주요 속성의 후보키에 대한 완전 함수 종속, 3NF는 전형적인 이행 종속의 제거와 연결된다. BCNF 빈칸의 선택지상 대응어는 후보키이지만, **엄밀한 조건은 비자명한 함수 종속의 결정자가 슈퍼키**라는 것이다.

NoteBase

빈칸별 해설

㉠ 원자값: 제1정규형은 각 속성 값이 해당 도메인에서 **더 나누지 않는 하나의 값**이 되도록 한다. 한 칸에 여러 값을 반복 집합으로 넣는 다중값과 구별한다.

㉡ 완전함수적 종속: 제2정규형은 제1정규형을 만족하면서 비주요 속성이 후보키의 진부분집합에 종속되는 부분함수 종속을 없앤다. 따라서 비주요 속성은 **후보키 전체에 완전 함수 종속**되어야 한다.

㉢ 이행함수적 종속: 제3정규형으로 진행할 때 제거하는 전형적인 문제는 키→비키 속성→다른 비키 속성의 **이행적 종속**이다. 엄밀한 3NF 조건은 모든 비자명한 함수 종속 X→A에 대해 X가 슈퍼키이거나 A가 주요 속성인 것이다. 따라서 모든 형태의 이행 관계 자체를 무조건 금지한다는 뜻은 아니다.

㉣ 후보키: 제시된 선택지의 전통적인 요약 표현에 대응하는 말은 후보키이며, 임의로 선정한 하나의 기본키만을 뜻하지 않는다. 다만 **BCNF의 엄밀한 조건은 모든 비자명한 함수 종속의 결정자가 슈퍼키인 것**이다. 최소 결정자를 중심으로 설명하면 후보키라고 표현하지만, 후보키에 속성을 더한 슈퍼키까지 배제해서는 안 된다.

따라서 빈칸은 원자값·완전함수적 종속·이행함수적 종속·후보키의 순서로 대응하며 ③이 가장 적절하다.

NoteBase

문 18. 다음은 관계형 데이터베이스 스키마와 SQL 질의문이다. 질의에 관한 설명으로 가장 적절한 것은? (단, 밑줄 속성은 기본키임)

데이터베이스 스키마	부품(부품번호, 부품이름, 재고수량, 단가) 공급자(공급자번호, 공급자상호, 주소) 주문(공급자번호, 부품번호, 주문날짜, 주문수량)
SQL 질의문	SELECT 공급자상호, 부품이름, SUM(주문수량)*단가 FROM 부품, 주문, 공급자 WHERE 부품.부품번호=주문.부품번호 AND 주문.공급자번호=공급자.공급자번호 AND 단가 >= 1000 AND 단가 <= 150000 GROUP BY 공급자.공급자번호, 부품.부품번호 HAVING SUM(주문수량) >= 100;

① 질의에 있는 SUM(재고수량)*단가는 모든 주문에 대한 총액을 계산한다.

② 단가가 150000 보다 큰 부품을 검색한다.

③ 공급자별로 주문수량의 합이 100 이상인 부품을 검색한다.

④ 부품, 공급자, 주문 테이블에 대해 자연조인이 불가능하여 동등조인으로 명세하였다.

정답 3번

간단한 해설

WHERE로 단가 범위를 제한한 뒤, **공급자·부품별**로 주문을 묶어 주문수량을 합산한다. HAVING은 그 합이 100 이상인 그룹만 남긴다.

NoteBase

선지별 해설

① X: 실제 질의의 식은 SUM(재고수량)*단가가 아니라 SUM(주문수량)*단가이다. 또한 공급자번호와 부품번호별로 그룹을 나누므로, 이 식은 해당 공급자·부품 그룹의 주문수량 합에 그 부품의 단가를 곱한 금액이지 모든 주문을 통틀어 구한 하나의 총액이 아니다.

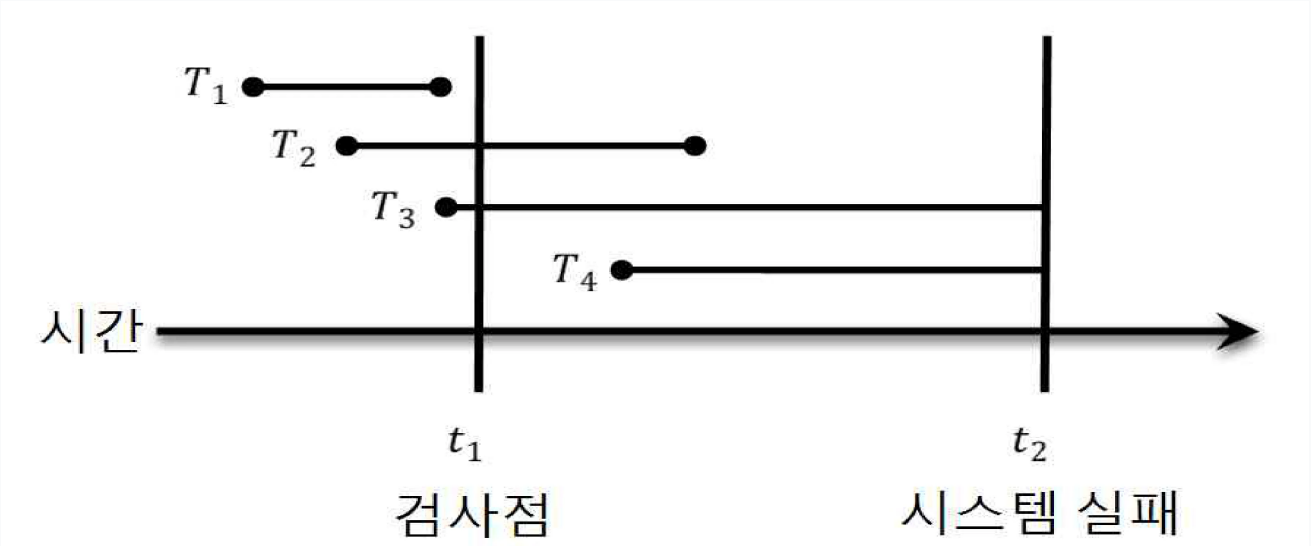
② X: WHERE는 단가가 1000 이상이고 150000 이하인 부품을 남긴다. 두 경계값을 모두 포함하므로 150000보다 큰 부품을 검색한다는 설명은 반대이다.

③ O: GROUP BY가 **공급자번호·부품번호의 조합**으로 주문들을 묶고, HAVING이 **주문수량 합이 100 이상인 그룹**만 남긴다. 따라서 단가 조건을 만족하는 부품 중에서 각 공급자별 주문수량 합계가 100 이상인 부품을 검색한다. 공급자상호와 부품이름, 단가는 각각 그룹에 포함된 기본키로 결정된다.

④ X: 부품과 주문은 부품번호를, 주문과 공급자는 공급자번호를 공통으로 가지므로 이 속성들을 이용한 자연조인이 가능하다. 질의에서 등호 조건으로 조인을 명시했다는 사실은 자연조인이 불가능하다는 뜻이 아니다.

NoteBase

문 19. 다음 트랜잭션들의 상태에서 지연 갱신을 기반으로 하는 회복기법 적용 시, 회복 연산으로 가장 적절하지 않은 것은?



① 트랜잭션 T_1은 아무 작업도 하지 않는다.

② 트랜잭션 T_2는 REDO 연산을 수행한다.

③ 트랜잭션 T_3는 UNDO 연산을 수행한다.

④ 트랜잭션 T_4는 아무 작업도 하지 않는다.

정답 3번

간단한 해설

지연 갱신에서는 **커밋 전 변경이 데이터베이스에 반영되지 않으므로 UNDO가 필요 없다.** 장애 시 미완료인 트랜잭션은 REDO 대상도 아니며, 검사점 이후 완료된 트랜잭션의 변경만 REDO로 반영한다.

NoteBase

선지별 해설

지연 갱신은 커밋 전 변경을 데이터베이스에 반영하지 않는 NO-UNDO/REDO 방식이다. 그림의 왼쪽 세로선은 검사점, 오른쪽 세로선은 시스템 실패 시점이며, 그 전에 끝점이 있는 트랜잭션은 완료된 것으로 읽는다.

① O: 그림에서 T_1은 검사점 t_1 전에 완료되었다. 문제의 검사점 회복 모형에서는 그때까지 완료된 변경이 이미 안전하게 반영되어 있으므로, 시스템 실패 후 이 트랜잭션에 대해 REDO나 UNDO를 수행할 필요가 없다.

② O: T_2는 검사점 당시 실행 중이었지만 시스템 실패 t_2 전에는 완료되었다. 검사점 이후 커밋된 변경을 데이터베이스에 확실히 반영하기 위해 로그의 새 값을 이용하는 **REDO**를 수행한다.

③ X: T_3는 검사점 전에 시작했으나 시스템 실패 때까지 완료되지 않았다. **지연 갱신은 미완료 트랜잭션의 변경을 데이터베이스에 반영하지 않으므로 UNDO가 필요 없다.** 커밋되지 않았으므로 REDO 대상도 아니며, 이 트랜잭션의 변경에 대한 데이터베이스 회복 연산은 수행하지 않는다.

④ O: T_4는 검사점 이후 시작하여 시스템 실패 때까지 완료되지 않았다. 지연 갱신에서는 이 변경도 데이터베이스에 반영되지 않았으므로 UNDO나 REDO를 하지 않는다. 이는 데이터베이스 회복 연산이 없다는 뜻이지, 장애 후 업무상 트랜잭션을 다시 실행할 수 없다는 뜻은 아니다.

NoteBase

문 20. 뷰(view)에 관한 설명으로 가장 적절하지 않은 것은?

① 뷰에 대한 INSERT, DELETE, UPDATE 연산은 항상 실행된다.

② 뷰는 항상 최신 정보를 가진다.

③ 질의 수정(query modification) 뷰는 뷰에 대한 질의를 기본 테이블에 대한 질의로 수정 또는 변경한다.

④ SQL에서 뷰를 정의하는 명령은 CREATE VIEW 문이다.

정답 1번

간단한 해설

뷰는 질의로 정의된 가상 테이블이며 정의 방식에 따라 갱신이 제한된다. **집계 등으로 만든 뷰는 기본 행에 대한 변경을 직접 결정하기 어려워** 모든 데이터 변경 연산이 항상 허용되지는 않는다.

선지별 해설

① (X) 뷰의 갱신 가능 여부는 정의 방식에 따라 달라진다. 집계나 그룹화 등으로 만든 뷰에서는 결과 행의 변경을 기본 테이블의 어떤 행에 어떻게 반영할지 바로 결정할 수 없어 갱신이 제한된다. 따라서 INSERT·DELETE·UPDATE가 **모든 뷰에서 항상 실행되는 것은 아니다**.

② (O) 일반적인 비물리화 뷰는 결과를 별도로 고정 저장하지 않고 조회할 때 기본 테이블에서 계산하므로 **조회 시 보이는 기본 데이터의 상태**를 반영한다. 여기서 최신이라는 표현은 트랜잭션의 가시성 범위 안에서 이해하며, 별도로 갱신해야 하는 물리화 뷰와는 구별한다.

③ (O) 질의 수정 방식은 뷰에 정의된 질의와 사용자의 조회 조건을 결합하여 **기본 테이블에 대한 질의로 변환**한다. 사용자는 뷰를 대상으로 질의하지만 실제 처리는 그 정의에 따라 기본 데이터에서 수행된다.

④ (O) SQL에서 일반 뷰의 정의에는 **CREATE VIEW** 문을 사용한다. 뷰 이름과 그 내용을 정하는 조회식을 지정하여 가상 테이블을 정의한다.

NoteBase