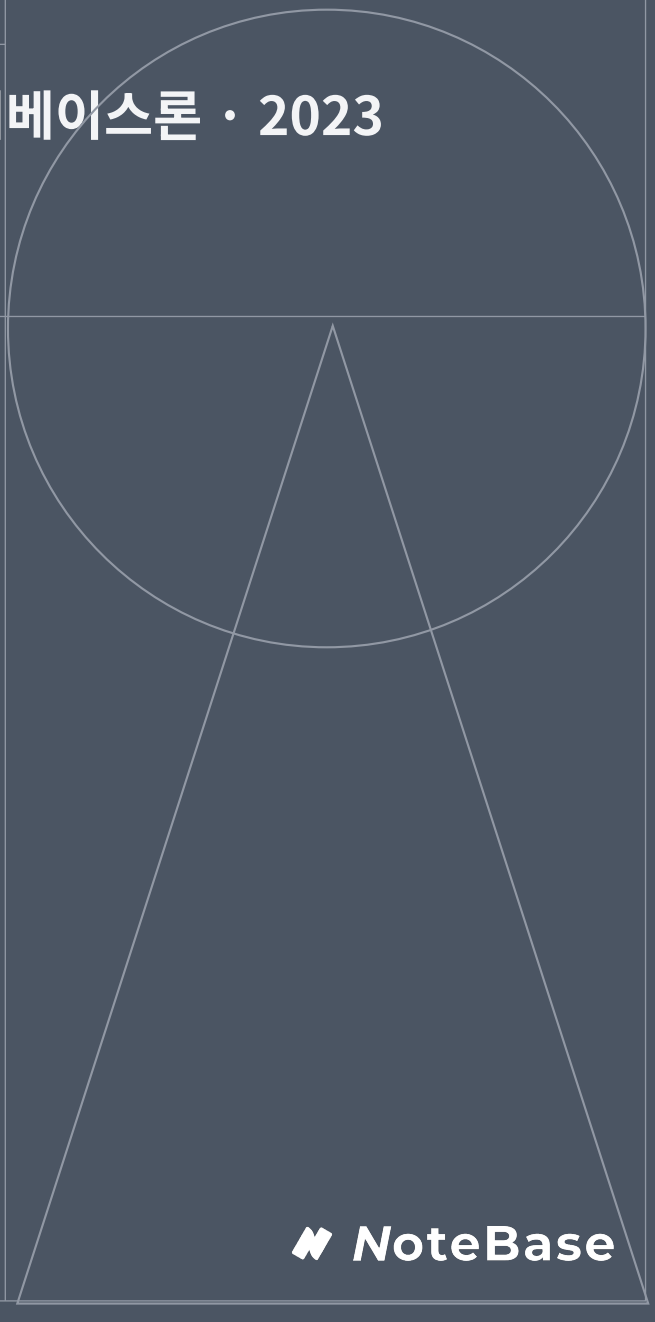
기출OX 해설집

공무원/국가직/9급 · 데이터베이스론 · 2023



NoteBase

기출OX 해설집

공무원/국가직/9급 · 데이터베이스론 · 2023

공무원/국가직/9급 · 데이터베이스론 · 2023

문 1. 다음에서 설명하는 데이터 마이닝 분석 기법은?

◯ 장바구니 분석이라고도 한다.
◯ 데이터 간의 발생 빈도를 분석하여 그 속에 숨겨진 규칙을 파악하는 방법으로, 상품이나 서비스 간의 관계를 분석하여 마케팅에 주로 활용한다.

① 분류 분석
② 군집 분석
③ 연관 분석
④ 회귀 분석

정답 3번

간단한 해설

'장바구니 분석'은 상품들이 **함께 구매되는 관계와 규칙**을 찾는 연관 분석을 뜻한다.

공통 해설

연관 분석은 거래 기록 등에서 항목들이 함께 나타나는 빈도를 바탕으로 항목 간 관계를 규칙으로 찾아내는 데이터 마이닝 기법이다. 예를 들어 구매 내역에서 빵과 우유가 함께 구매되는 경향을 찾아 상품 배치나 묶음 판매에 활용할 수 있다. 이처럼 **같은 거래에서 상품들이 함께 구매되는 관계**를 분석하므로 '장바구니 분석'이라고도 한다. 지문의 상품·서비스 간 발생 관계에서 숨은 규칙을 찾는다는 설명은 연관 분석의 목적에 해당한다.

NoteBase

문 2. 시스템 카탈로그에 대한 설명으로 옳지 않은 것은?

① 데이터 사전(data dictionary)이라고도 한다.

② 각 릴레이션의 튜플 수와 블록 수가 저장된다.

③ 일반 사용자도 시스템 카탈로그에 내용을 추가하거나 수정할 수 있다.

④ RDBMS마다 서로 다른 형태로 시스템 카탈로그를 제공한다.

정답 3번

간단한 해설

시스템 카탈로그의 내용을 **일반 사용자가 직접 추가·수정할 수는 없다**. 테이블 생성·변경에 따라 필요한 카탈로그 갱신은 DBMS가 수행한다.

선지별 해설

① O: 시스템 카탈로그는 테이블과 속성의 이름, 자료형, 제약조건 등 **데이터에 관한 정보인 메타데이터**를 저장한다. 데이터베이스의 구조와 관리 정보를 모아 두므로 데이터 사전이라고도 한다.

② O: 각 릴레이션의 튜플 수와 저장에 사용되는 블록 수 같은 **통계 정보**도 시스템 카탈로그에 저장된다. 튜플은 릴레이션의 행이고 블록은 디스크에서 데이터를 저장·입출력하는 단위이다. DBMS는 이러한 정보를 이용하여 질의 처리 비용을 추정하고 실행 계획을 선택한다.

③ X: 시스템 카탈로그는 DBMS가 관리하므로 일반 사용자가 그 내용을 직접 추가하거나 수정할 수 없다. 카탈로그가 임의로 바뀌면 실제 데이터베이스 구조와 관리 정보가 달라져 정상적인 처리를 방해할 수 있기 때문이다. 사용자가 테이블 생성·변경 명령을 내리면 **DBMS가 해당 메타데이터를 갱신**한다. 이는 사용자가 카탈로그 자체를 직접 수정하는 것과 구별된다.

④ O: 시스템 카탈로그는 공통으로 메타데이터를 관리하지만, 이를 구성하는 **시스템 테이블·뷰의 이름과 구조** 등은 RDBMS마다 다르다. 따라서 제품에 따라 시스템 카탈로그를 제공하는 형태가 달라진다.

NoteBase

문 3. 다음에서 설명하는 데이터베이스 무결성 제약조건(integrity constraint)은?

◯ 기본키 제약(primary key constraint)이라고도 한다.
◯ 기본키는 널(NULL) 값을 가져서는 안되며 릴레이션 내에 오직 하나의 값만 존재해야 한다.

① 개체 무결성 제약조건
② 고유 무결성 제약조건
③ 도메인 무결성 제약조건
④ 참조 무결성 제약조건

정답 1번

간단한 해설

기본키로 각 행을 구별하려면 **NULL이 없어야 하고 값이 중복되지 않아야 한다**. 이처럼 기본키를 통한 행의 식별을 보장하는 제약이 개체 무결성 제약조건이다.

공통 해설

개체 무결성 제약조건은 릴레이션의 각 튜플, 즉 행을 기본키로 명확히 식별할 수 있도록 하는 규칙이며, 기본키 제약이라고도 한다. 기본키를 구성하는 속성에는 NULL이 허용되지 않고, 서로 다른 행의 기본키 값은 중복될 수 없다. 제시문의 '릴레이션 내에 오직 하나의 값만 존재'한다는 말은 릴레이션 전체에 값이 하나뿐이라는 뜻이 아니라, **같은 기본키 값이 여러 행에 나타날 수 없다**는 뜻이다.

예를 들어 학생 릴레이션에서 학번을 기본키로 정했다면 학번이 NULL인 행이나, 이미 존재하는 학번과 같은 학번을 가진 다른 행을 저장할 수 없다. 중복 금지만을 따로 보는 것이 아니라, **기본키의 NULL 금지와 유일성을 함께 요구**한다는 점이 제시된 조건을 개체 무결성과 연결하는 기준이다.

NoteBase

문 4. 데이터베이스 모델에서 사용되는 키(key)에 대한 설명으로 옳지 않은 것은?

① 후보키(candidate key)는 유일성과 최소성의 특성을 만족하는 속성 또는 속성들의 집합이다.

② 슈퍼키(super key)는 유일성의 특성을 만족하는 속성 또는 속성들의 집합이다.

③ 대체키(alternate key)는 기본키(primary key)로 선택되지 못한 외래키(foreign key)들이다.

④ 널(NULL) 값을 가질 수 있는 속성이 포함된 후보키는 기본키로 부적합하다.

정답 3번

간단한 해설

대체키는 기본키로 선택되지 않은 **후보키**이다. ③의 '외래키'는 '후보키'로 고쳐야 한다.

선지별 해설

① O: 유일성은 키 값으로 릴레이션의 각 튜플을 서로 구별할 수 있다는 뜻이다. 최소성은 유일성을 유지하면서 **속성을 하나라도 더 제거할 수 없다**는 뜻이다. 이 두 조건을 모두 만족하는 속성 집합이 후보키이며, 최소성이 모든 후보키 중 속성 수가 가장 적다는 뜻은 아니다.

② O: 슈퍼키는 각 튜플을 유일하게 식별할 수 있는 속성 집합으로, **최소성은 요구하지 않는다**. 예를 들어 학번만으로 학생을 구별할 수 있다면 학번은 후보키이자 슈퍼키이고, 학번에 이름을 추가한 속성 집합도 유일성을 유지하므로 슈퍼키이다. 다만 이름을 제거해도 식별할 수 있으므로 이 속성 집합은 후보키가 아니다.

③ X: 대체키는 여러 후보키 중 기본키로 선택된 하나를 제외한 **나머지 후보키**이다. 예를 들어 학번과 주민등록번호가 각각 후보키이고 학번을 기본키로 정했다면 주민등록번호는 대체키가 된다. 외래키는 릴레이션 사이의 참조 관계를 나타내는 키로, 기본키로 선택되지 않은 후보키를 뜻하지 않는다.

④ O: 기본키는 각 튜플을 식별하는 기준이므로 개체 무결성 제약에 따라 **기본키를 구성하는 모든 속성에 NULL이 허용되지 않는다**. 따라서 NULL을 허용하는 속성이 포함된 후보키를 기본키로 사용하려면 그 속성에도 NULL을 허용하지 않도록 해야 하며, NULL을 허용하는 상태로는 기본키로 부적합하다.

NoteBase

문 5. 릴레이션 R_1, R_2, R_3의 스키마는 R_1(A, B, C), R_2(C, D, E), R_3(E, F)이고, 기본키는 각각 A, C, E이다. 튜플의 수가 R_1은 100개, R_2는 150개, R_3는 75개라고 할 때, $R_1 \bowtie_N R_2 \bowtie_N R_3$의 결과에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. $R_1 \bowtie_N (R_2 \bowtie_N R_3)$의 결과와 동일하다.
ㄴ. 결과 릴레이션의 튜플은 최대 100개이다.
ㄷ. 결과 릴레이션의 속성은 8개이다.

① ㄱ
② ㄴ
③ ㄱ, ㄴ
④ ㄱ, ㄴ, ㄷ

정답 3번

간단한 해설

조인 상대의 C와 E가 각각 기본키이므로, 처음 릴레이션의 각 튜플에 연결되는 튜플은 단계마다 **최대 하나**이다.
자연조인은 같은 이름의 속성을 하나로 합치므로 결과의 속성은 **6개**이다.

NoteBase

보기별 해설

자연조인은 두 릴레이션에서 같은 이름의 속성값이 일치하는 튜플을 연결하고, 공통 속성은 결과에 한 번만 남기는 연산이다. 이 문제에서는 C와 E의 값이 각각 일치해야 한다.

ㄱ. O: 자연조인은 **결합법칙**이 성립한다. R_1과 R_2를 먼저 조인하면 C가 같은 튜플을 연결한 뒤 R_3와 E를 비교한다. R_2와 R_3를 먼저 조인하면 E가 같은 튜플을 연결한 뒤 R_1과 C를 비교한다. 어느 쪽도 C와 E가 모두 일치하는 조합만 남기므로 결과가 같다.

ㄴ. O: C는 R_2의 기본키이므로, R_1의 각 튜플과 C값이 같은 R_2의 튜플은 최대 하나이다. 따라서 첫 조인의 결과는 최대 100개이다. 이어서 E가 R_3의 기본키이므로, 첫 조인의 각 튜플에 연결되는 R_3의 튜플도 최대 하나이다. 두 단계 모두 **기본키와의 조인으로 튜플이 여러 개로 늘어나지 않는다**. 모든 튜플이 두 단계에서 일치하는 상대를 찾으면 최종 결과는 100개가 된다. 여러 튜플이 같은 E값으로 R_3의 한 튜플과 연결될 수 있으므로, R_3가 75개라는 사실이 결과를 75개 이하로 제한하지는 않는다. 일치하는 상대가 없으면 결과는 100개보다 적을 수 있다.

ㄷ. X: 8개는 세 릴레이션의 속성 수를 단순히 더한 값이다. 자연조인에서는 공통 속성 C와 E를 각각 한 번씩만 남기므로, 속성 수는 3 + 3 + 2 − 1 − 1 = 6개이다. 결과 스키마는 **(A, B, C, D, E, F)**이다.

옳은 보기는 ㄱ과 ㄴ이므로, 이에 해당하는 조합은 ③이다.

NoteBase

문 6. 데이터베이스의 스키마와 인스턴스에 관한 설명을 바르게 연결한 것은?

ㄱ. 데이터베이스의 논리적, 물리적 구조 정의
ㄴ. 특정 시점에 데이터베이스에 저장되어 있는 실제 값
ㄷ. DML(Data Manipulation Language)을 활용하여 생성 및 변환
ㄹ. DDL(Data Definition Language)을 활용하여 생성 및 변환

① 스키마: ㄱ, ㄴ, 인스턴스: ㄷ, ㄹ
② 스키마: ㄱ, ㄹ, 인스턴스: ㄴ, ㄷ
③ 스키마: ㄴ, ㄷ, 인스턴스: ㄱ, ㄹ
④ 스키마: ㄴ, ㄹ, 인스턴스: ㄱ, ㄷ

정답 2번

간단한 해설

스키마는 데이터베이스의 구조이고, **인스턴스는 특정 시점에 저장된 실제 데이터**이다. 구조를 정의·변경하는 DDL은 스키마에, 데이터를 삽입·수정하는 DML은 인스턴스에 대응한다.

구성 요소별 해설

ㄱ. 스키마: 데이터베이스의 논리적·물리적 구조를 정의한 것이다. 어떤 테이블과 열을 두고 각 열에 어떤 자료형을 사용할지 등 **데이터를 담는 구조**를 정하며, 그 안에 저장된 개별 값과 구별된다.

ㄴ. 인스턴스: **특정 시점에 저장되어 있는 실제 값**의 집합이다. 예를 들어 학생 테이블의 열 구성이 같더라도 학생 데이터가 추가되면 인스턴스는 달라진다.

ㄷ. 인스턴스: DML은 데이터 조작어로, 정의된 구조 안에서 데이터를 삽입·수정·삭제하는 데 사용한다. INSERT로 행을 추가하거나 UPDATE로 값을 바꾸는 것은 **구조가 아니라 저장된 데이터**를 바꾸므로 인스턴스의 생성·변환에 해당한다.

ㄹ. 스키마: DDL은 데이터 정의어로, 데이터베이스 객체의 구조를 정의하거나 변경하는 데 사용한다. CREATE TABLE로 테이블을 만들거나 ALTER TABLE로 열을 추가하는 것은 **데이터를 저장할 구조를 정의·변경**하므로 스키마에 해당한다.

스키마에 해당하는 것은 ㄱ·ㄹ, 인스턴스에 해당하는 것은 ㄴ·ㄷ이며, 이 분류에 대응하는 선택지는 ②이다.

NoteBase

문 7. 데이터베이스 회복(recovery) 기법에 대한 설명으로 옳은 것은?

① 트랜잭션이 갱신한 캐시 버퍼 페이지를 트랜잭션이 완료되기 전에 디스크에 기록할 수 있는 방식을 steal이라고 한다.

② 트랜잭션이 갱신한 모든 페이지가 트랜잭션이 완료되기 전에 즉시 디스크에 반영되는 것을 no-force 방식이라고 한다.

③ 지연갱신을 기반으로 하는 회복 기법에서 로그 레코드는 <트랜잭션 ID, 데이터 아이템, 변경 이전 값>의 형식을 갖는다.

④ 즉시갱신에 기반을 둔 회복 기법은 NO-UNDO/REDO에 의해 수행된다.

정답 1번

간단한 해설

steal은 트랜잭션이 **커밋하기 전에도 갱신된 버퍼 페이지를 디스크에 기록할 수 있는** 정책이다. 이후 트랜잭션이 취소되면 디스크에 반영된 변경을 되돌릴 수 있어야 한다.

선지별 해설

① O: steal은 아직 커밋하지 않은 트랜잭션이 갱신한 버퍼 페이지도 디스크에 기록하여 버퍼 공간을 재사용할 수 있게 하는 정책이다. 미완료 트랜잭션의 변경이 디스크에 남을 수 있으므로, 트랜잭션이 취소되거나 장애로 완료되지 못하면 그 변경을 되돌리는 **UNDO가 필요하다**.

② X: no-force는 트랜잭션의 **커밋 시 모든 갱신 페이지를 디스크에 기록하도록 강제하지 않는** 정책이다. 커밋 전에 모두 즉시 반영한다는 뜻이 아니다. 커밋 시 모든 갱신 페이지의 디스크 반영을 보장하는 정책은 force이다. no-force에서는 커밋한 트랜잭션의 변경이 디스크에 남지 않은 상태에서 장애가 날 수 있으므로, 로그를 이용해 변경을 재반영하는 REDO가 필요하다.

③ X: 지연갱신은 커밋 전에는 트랜잭션의 변경을 데이터베이스에 반영하지 않고, 커밋 후에 반영하는 방식이다. 따라서 미완료 트랜잭션의 변경을 되돌릴 필요는 없으며, 완료된 트랜잭션의 변경을 재반영할 수 있도록 갱신 로그에 **변경 이후 값**을 기록한다. 선지의 형식은 '<트랜잭션 ID, 데이터 아이템, 변경 이후 값>'으로 고쳐야 한다.

④ X: 즉시갱신은 트랜잭션이 커밋하기 전에도 변경을 데이터베이스에 반영할 수 있는 방식이므로, 미완료 트랜잭션의 변경을 취소하는 UNDO가 필요하다. 일반적인 steal/no-force 정책을 사용하는 즉시갱신 회복은 변경 이전 값과 이후 값을 로그에 기록하고 **UNDO/REDO**로 수행한다. UNDO는 미완료 트랜잭션의 변경을 되돌리고, REDO는 완료됐지만 디스크에 남지 않은 변경을 재반영한다. 선지의 NO-UNDO/REDO는 지연갱신에 해당한다.

NoteBase

문 8. SQL에서 다음 상품 테이블에 대한 ANY와 ALL 연산의 결과로 옳지 않은 것은?

상품

상품번호	가격
1	25000
2	30000
3	40000

① '40000 = ANY (SELECT 가격 FROM 상품)'은 참이다.

② '40000 <> ANY (SELECT 가격 FROM 상품)'은 참이다.

③ '30000 <> ALL (SELECT 가격 FROM 상품)'은 참이다.

④ '30000 > ALL (SELECT 가격 FROM 상품)'은 거짓이다.

정답 3번

간단한 해설

'30000 <> ALL (...)'이 참이려면 조회된 **모든 가격이 30000과 달라야** 한다. 상품 테이블에 가격이 30000인 행이 있으므로 이 조건식의 결과는 **거짓**이다.

NoteBase

선지별 해설

'SELECT 가격 FROM 상품'의 조회 결과는 25000, 30000, 40000이다. ANY는 각 값과의 비교 중 하나 이상이 참이면 참이고, ALL은 모든 값과의 비교가 참이어야 참이다. '<>'는 두 값이 서로 다름을 뜻한다.

① O: '40000 = ANY (...)'는 조회된 가격 중 **40000과 같은 값이 하나 이상** 있는지를 검사한다. 상품번호 3의 가격이 40000이므로 조건식은 참이다.

② O: '40000 <> ANY (...)'는 조회된 가격 중 **40000과 다른 값이 하나 이상** 있는지를 검사한다. 25000과 30000은 40000과 다르므로 조건식은 참이다. 40000과의 비교가 거짓이어도 ANY 전체가 거짓이 되지는 않는다.

③ X: '30000 <> ALL (...)'는 조회된 가격이 모두 30000과 달라야 참이다. 25000, 40000과의 비교는 참이지만, 상품번호 2의 가격 30000과 비교하면 같은 값이므로 '30000 <> 30000'은 거짓이다. ALL은 **하나라도 거짓이면 거짓**이므로, 선지의 '참이다'를 '거짓이다'로 고쳐야 한다.

④ O: '30000 > ALL (...)'이 참이려면 30000이 조회된 모든 가격보다 커야 한다. 25000보다는 크지만 30000과는 같고 40000보다는 작으므로 조건식은 거짓이다. 따라서 조건식이 **거짓이라는 선지의 진술**은 성립한다.

NoteBase

문 9. SQL 명령어 중에서 데이터 제어어(DCL)만을 모두 고르면?

ㄱ. CREATE
ㄴ. GRANT
ㄷ. DROP
ㄹ. REVOKE

① ㄱ, ㄴ
② ㄴ, ㄹ
③ ㄷ, ㄹ
④ ㄴ, ㄷ, ㄹ

정답 2번

간단한 해설

데이터 제어어(DCL)는 데이터베이스의 **접근 권한을 관리**하는 명령어이다. GRANT는 권한을 부여하고, REVOKE는 부여된 권한을 회수한다.

보기별 해설

ㄱ. CREATE — 데이터 정의어(DDL): 테이블 등 데이터베이스 객체를 **생성**하는 명령어이다. 객체의 구조를 정의하므로 접근 권한을 관리하는 DCL과 구별된다.

ㄴ. GRANT — 데이터 제어어(DCL): 사용자 등에게 데이터베이스 객체에 대한 **접근 권한을 부여**하는 명령어이다. 예를 들어 특정 테이블을 조회할 수 있는 권한을 줄 때 사용한다.

ㄷ. DROP — 데이터 정의어(DDL): 테이블 등 데이터베이스 객체를 **제거**하는 명령어이다. 객체 자체를 없애는 것이므로 권한만 회수하는 REVOKE와 다르다.

ㄹ. REVOKE — 데이터 제어어(DCL): 사용자 등에게 **부여된 권한을 회수**하는 명령어이다. 테이블 조회 권한을 회수하더라도 테이블 자체를 제거하는 것은 아니다.

데이터 제어어에 해당하는 것은 ㄴ과 ㄹ이며, ②의 조합과 일치한다.

NoteBase

문 10. 역정규화(denormalization)에 대한 설명으로 옳지 않은 것은?

① 중복된 데이터를 일관성 있게 유지하기 위한 추가적인 작업이 필요하다.

② 갱신 시간이 중요한(time critical) 연산을 지원하기 위해 사용된다.

③ 더 높은 정규형 릴레이션들을 조인한 결과를 저장하여 더 낮은 정규형으로 되돌아가는 과정이다.

④ 여러 테이블을 결합하는 데 시간이 오래 걸리는 경우, 역정규화를 통해 검색 시간을 단축할 수 있다.

정답 2번

간단한 해설

②의 '갱신 시간이 중요한 연산'은 역정규화의 주된 적용 대상이 아니다. 역정규화는 중복 데이터의 갱신 부담을 감수하면서 **조인 비용을 줄여 검색 성능을 높이는** 데 사용된다.

선지별 해설

역정규화는 검색 성능 등을 높이기 위해 정규화된 데이터베이스에 중복을 허용하거나 분리된 테이블을 합치는 설계 방법이다.

① O: 역정규화로 같은 데이터를 여러 곳에 저장하면, 값이 바뀔 때 **중복된 데이터도 함께 수정**해야 한다. 예를 들어 고객명을 여러 주문 행에 중복 저장한 경우, 고객명이 변경되면 해당 주문 행들의 고객명도 갱신해야 일관성이 유지된다.

② X: 역정규화는 일반적으로 **갱신보다 검색 성능을 높이기 위해** 사용된다. 중복 데이터가 늘어나면 한 값의 변경을 여러 곳에 반영해야 하므로 갱신 비용이 커질 수 있다. 따라서 갱신 시간이 중요한 연산을 지원한다는 설명은 검색 시간이 중요한 연산을 지원한다는 내용으로 고쳐야 한다.

③ O: 높은 정규형으로 분해된 릴레이션들을 조인하고 그 결과를 하나의 릴레이션으로 저장하면, 분리했던 속성들이 다시 모이면서 **데이터 중복이 생기고 정규형이 낮아질 수 있다**. 이처럼 검색 성능 등을 위해 정규화된 구조를 의도적으로 되돌리는 것이 역정규화이다.

④ O: 여러 테이블의 데이터를 함께 조회할 때마다 조인하면 테이블을 읽고 연결하는 비용이 든다. 자주 사용하는 결합 결과를 미리 저장하면 **조회할 때 수행할 조인을 줄일 수 있어** 검색 시간이 단축될 수 있다.

NoteBase

문 11. 다음은 CLUB 테이블과 MEETING 테이블에 대한 SQL문이다. 실행 결과에 대한 설명으로 옳은 것은?

CLUB

CID	이름	도시	회비
10	Tennis	Atlanta	50
12	Music	New York	30
13	Art	Atlanta	40
15	Book	Dallas	10
17	Audio	Macon	20

MEETING

CID	날짜	참석수
10	2023-02-10	2
10	2023-04-10	1
13	2023-01-21	3
13	2023-02-21	4
15	2023-09-30	2

NoteBase

```
(SELECT CID
FROM CLUB
WHERE 회비<30)
UNION
(SELECT CID
FROM MEETING
WHERE 참석수>=3);
```

① '회비<30'와 '참석수>=3'를 동시에 만족하는 '회비'와 '참석수'가 검색된다.

② '회비<30'와 '참석수>=3'를 동시에 만족하는 'CID'가 검색된다.

③ 결과 테이블의 레코드 수는 총 4개이다.

④ 검색된 레코드의 합은 45이다.

정답 4번

간단한 해설

UNION은 두 조회 결과를 합치면서 **중복된 CID를 제거**한다. 최종 CID는 13, 15, 17이며, 이 값들의 합은 13 + 15 + 17 = 45이다.

NoteBase

선지별 해설

① X: 두 SELECT문 모두 **CID만 조회**한다. 회비와 참석수는 WHERE절에서 행을 고르는 조건으로 사용될 뿐 결과 열에 포함되지 않는다. 또한 UNION은 두 조회 결과의 합집합을 구하므로 두 조건을 동시에 만족하는 행을 찾는 연산이 아니다.

② X: CLUB에서 회비<30인 행의 CID는 15, 17이고, MEETING에서 참석수>=3인 행의 CID는 13, 13이다. UNION은 **각 조회 결과 중 어느 한쪽에라도 포함된 CID**를 중복 없이 반환하므로 결과는 13, 15, 17이다. 두 조건을 동시에 만족하는 CID를 검색한다는 설명은 잘못이다.

③ X: 두 조회 결과를 단순히 이어 붙이면 CID가 15, 17, 13, 13으로 4개이지만, **UNION은 중복 행을 제거**한다. 따라서 CID 13은 한 번만 남아 결과 테이블의 레코드 수는 3개이다.

④ O: 결과 테이블에는 CID 열 하나에 13, 15, 17이 들어 있으므로 **검색된 CID 값의 합**은 13 + 15 + 17 = 45이다. SQL문 자체가 합계를 계산해 45를 반환하는 것은 아니며, 검색 결과에 포함된 값들을 더한 것이 45라는 뜻이다.

NoteBase

문 12. 다음 릴레이션 고객과 주문에 대한 완전외부조인(full outer join) 연산의 결과로 (가), (라)에 들어갈 내용은?

고객		
고객ID	고객이름	주소
ok	홍길동	서울
pk	강감찬	경기

주문		
주문번호	고객ID	주문제품
100	ok	운동화
101	ak	장갑

[완전외부조인 연산 결과]

고객ID	고객이름	주소	주문번호	주문제품
(가)	홍길동	서울	100	운동화
(나)	강감찬	경기	(라)	(라)
(다)	(라)	(라)	101	장갑

① (가): ak, (라): 0

② (가): ak, (라): NULL

③ (가): ok, (라): 0

④ (가): ok, (라): NULL

정답 4번

간단한 해설

홍길동의 고객ID는 주문번호 100의 고객ID와 같은 ok이므로 (가)는 **ok**이다. 완전외부조인은 대응하는 행이 없어도 양쪽 행을 모두 남기고 없는 쪽의 속성을 **NULL**로 채우므로 (라)는 NULL이다.

NoteBase

계산 과정

완전외부조인은 **조인 조건에 맞는 행을 결합하고, 짝이 없는 양쪽 행도 보존**하는 연산이다. 이 문제에서는 두 릴레이션의 공통 속성인 고객ID를 기준으로 행을 대응시키고, 대응하는 행이 없는 쪽의 속성에는 NULL을 넣는다.

(가): 고객 릴레이션에서 홍길동의 고객ID는 ok이고, 주문 릴레이션의 주문번호 100도 고객ID가 ok이다. 두 행을 결합하면 'ok, 홍길동, 서울, 100, 운동화'가 되므로 (가)는 ok이다.

(나)와 두 번째 행의 (라): 강감찬의 고객ID는 pk이지만 주문 릴레이션에는 고객ID가 pk인 행이 없다. 강감찬의 행은 보존하고 주문번호와 주문제품을 NULL로 채운다. 따라서 (나)는 pk이며 두 칸의 (라)는 모두 NULL이다.

(다)와 세 번째 행의 (라): 주문번호 101의 고객ID는 ak이지만 고객 릴레이션에는 고객ID가 ak인 행이 없다. 이 주문 행도 보존하므로 (다)는 ak이고, 대응하는 고객이름과 주소는 NULL이 된다. NULL은 **대응하는 값이 없음을 나타내는 표시**이며 숫자 0으로 채우는 것이 아니다.

따라서 (가)는 ok, (라)는 NULL로, ④의 내용과 일치한다.

NoteBase

문 13. 분산 데이터베이스에 대한 설명으로 옳지 않은 것은?

① 컴퓨터 네트워크를 통해서 데이터베이스 노드들이 연결된다.

② 수평적 단편화는 릴레이션을 속성 단위로 나눈다.

③ 다양한 데이터베이스 노드 내의 정보는 논리적으로 연관된다.

④ 각 노드들은 독립적으로 동작할 수 있는 자치성을 가진다.

정답 2번

간단한 해설

수평적 단편화는 릴레이션을 **튜플(행) 단위**로 나누는 방식이다. 속성(열) 단위로 나누는 것은 **수직적 단편화**이다.

선지별 해설

① O: 분산 데이터베이스는 여러 장소의 노드에 데이터를 나누어 저장하고, 노드들을 **컴퓨터 네트워크로 연결**한다. 노드 간 통신을 통해 다른 노드에 저장된 데이터에도 접근할 수 있다.

② X: 수평적 단편화는 릴레이션을 **튜플(행) 단위**로 나눈다. 예를 들어 직원 릴레이션에서 서울 지점 직원의 행과 부산 지점 직원의 행을 나누어 저장하는 방식이다. 선지의 '속성 단위로 나눈다'는 **수직적 단편화**에 대한 설명으로, 열을 기준으로 나누는 것을 뜻한다.

③ O: 각 노드의 데이터는 물리적으로 떨어져 있어도 **논리적으로 연관된 하나의 데이터베이스**를 구성한다. 서로 무관한 데이터베이스를 단순히 네트워크에 연결한 것만으로는 분산 데이터베이스가 되지 않는다.

④ O: 각 노드는 자신의 데이터를 관리하고 지역 작업을 처리할 수 있는 **자치성**을 가진다. 여러 노드의 데이터가 필요한 작업에서는 다른 노드와 협력하지만, 각 노드에서 수행하는 지역 작업은 독립적으로 처리할 수 있다.

NoteBase

문 14. 다음은 인덱스에 관한 내용이다. (가), (나)에 들어갈 용어를 바르게 연결한 것은?

○ (가)는/은 많은 수의 행을 가진 릴레이션을 위해 사용하는 기법으로, 하나 이상의 열에 대해 인덱스를 생성하며 적은 수의 유일한 값들을 갖는 열들에 적합하다.
○ (나)는/은 디렉터리(directory)와 버킷(bucket) 집합을 사용하는 기법으로, 데이터베이스가 증가하고 축소되는 변화에 유연하다.

① (가): 그리드 파일, (나): 비트맵 인덱스
② (가): 그리드 파일, (나): 확장성 해싱
③ (가): 비트맵 인덱스, (나): 그리드 파일
④ (가): 비트맵 인덱스, (나): 확장성 해싱

정답 4번

간단한 해설

비트맵 인덱스는 각 값에 해당하는 행을 비트로 표시하므로 **서로 다른 값의 수가 적은 열**에 적합하다. **확장성 해싱**은 디렉터리와 버킷을 조정하여 데이터의 증감에 대응한다.

빈칸별 해설

(가) 비트맵 인덱스: 열의 각 값에 대해 비트열을 만들고, 각 행이 그 값을 가지면 1, 아니면 0으로 표시한다. 예를 들어 세 행의 값이 A, B, A라면 A의 비트열은 101, B의 비트열은 010이다. 행이 많더라도 **서로 다른 값의 수가 적으면** 관리할 비트열의 수가 적어 효율적이다. 여러 조건은 비트열의 AND·OR 연산으로 결합할 수 있다. 따라서 '적은 수의 유일한 값들을 갖는 열'이라는 단서는 비트맵 인덱스에 해당한다.

(나) 확장성 해싱: 키의 해시값 중 일부 비트를 이용하여 디렉터리 항목을 찾고, 그 항목이 가리키는 버킷에 접근한다. 버킷이 가득 차면 **해당 버킷을 분할**하고, 필요할 때 디렉터리도 확장한다. 데이터가 줄면 조건에 따라 버킷을 병합하고 디렉터리를 축소할 수 있어 데이터의 증감에 유연하다. 그리드 파일도 디렉터리로 버킷을 찾지만, 여러 검색 키의 값 범위를 기준으로 다차원 공간을 분할한다는 점에서 구별된다.

(가)에 비트맵 인덱스, (나)에 확장성 해싱을 넣으면 ④의 조합과 일치한다.

NoteBase

문 15. 다음 도서 테이블에 대한 SQL문을 수행하였을 때, 결과 테이블의 튜플 수는?

도서

도서번호	도서명	저자명	출판사
1	국어 기초	Anderson	Blue
2	국어 심화	Bella	White
3	국어 심화	Anderson	White
4	영어 기초	Anderson	White
5	영어 심화	Cooper	Green
6	영어 심화	Cooper	Green
7	수학 기초	Anderson	Green
8	수학 기초	Frank	White
9	수학 심화	Davis	Blue
10	수학 심화	Frank	Red

```
SELECT DISTINCT 출판사
FROM 도서
WHERE 저자명 = 'Anderson';
```

① 1
② 2
③ 3
④ 4

정답 3번

간단한 해설

저자가 Anderson인 행의 출판사는 Blue, White, White, Green이다. **DISTINCT가 중복된 White를 하나로 처리**하므로 결과 테이블에는 3개의 튜플이 남는다.

NoteBase

계산 과정

1. 조건에 맞는 행 선택: WHERE 저자명 = 'Anderson'은 저자가 Anderson인 행만 남긴다. 도서번호 1, 3, 4, 7이 해당하므로 이 단계에서는 4개의 행이 선택된다.

2. 출판사 추출과 중복 제거: SELECT는 선택된 행에서 출판사 열만 추출한다. 그 값은 Blue, White, White, Green이다. DISTINCT는 **출력하는 열의 값이 같은 행을 하나로 합친다**. 도서번호 3과 4는 서로 다른 도서이지만 출력할 출판사 값은 모두 White이므로 결과에서는 하나의 행으로 남는다.

3. 튜플 수 계산: 중복을 제거한 출판사는 Blue, White, Green의 세 가지이다. 튜플은 테이블의 한 행이므로 결과 테이블의 튜플 수는 3개이며, ③의 값과 일치한다.

NoteBase

문 16. 다음 릴레이션 R1과 R2에 대한 디비전 연산(R1 ÷ R2)의 결과는?

R1		
A1	A2	A3
s1	t1	u1
s1	t2	u1
s2	t1	u1
s2	t1	u2
s2	t2	u2
s3	t1	u1
s3	t1	u2
s4	t1	u1
s5	t2	t2

R2	
A2	A3
t1	u1
t1	u2

①

A1
s1

②

A1
s2
s3

NoteBase

③

A1
s2
s4
s5

④

A1
s1
s2
s3
s4
s5

정답 2번

간단한 해설

디비전은 R2의 **모든 튜플과 짝을 이루는** A1 값을 구하는 연산이다. (t1, u1)과 (t1, u2)를 모두 갖는 값은 s2와 s3이므로, 결과에는 이 두 값만 남는다.

NoteBase

계산 과정

연산 기준: R1의 속성은 A1·A2·A3이고 R2의 속성은 A2·A3이므로, R1 ÷ R2의 결과 속성은 **R2에 없는 A1**이다. 각 A1 값에 대해 R2의 모든 (A2, A3) 튜플과 결합한 튜플이 R1에 존재하는지 확인한다.

필요한 튜플: R2에는 (t1, u1)과 (t1, u2)가 있다. 따라서 어떤 A1 값이 결과에 포함되려면 그 값에 대응하는 (A2, A3)에 **이 두 쌍이 모두 있어야 한다**.

s1: (t1, u1)과 (t2, u1)은 있지만 (t1, u2)가 없다. 필요한 두 쌍 중 하나가 빠져 있으므로 제외한다.

s2: (t1, u1)과 (t1, u2)가 모두 있으므로 포함한다. 추가로 있는 (t2, u2)는 포함 여부에 영향을 주지 않는다. 디비전은 **필요한 쌍을 모두 갖추면 되며, 다른 쌍이 있어도 허용**하기 때문이다.

s3: (t1, u1)과 (t1, u2)가 모두 있으므로 포함한다.

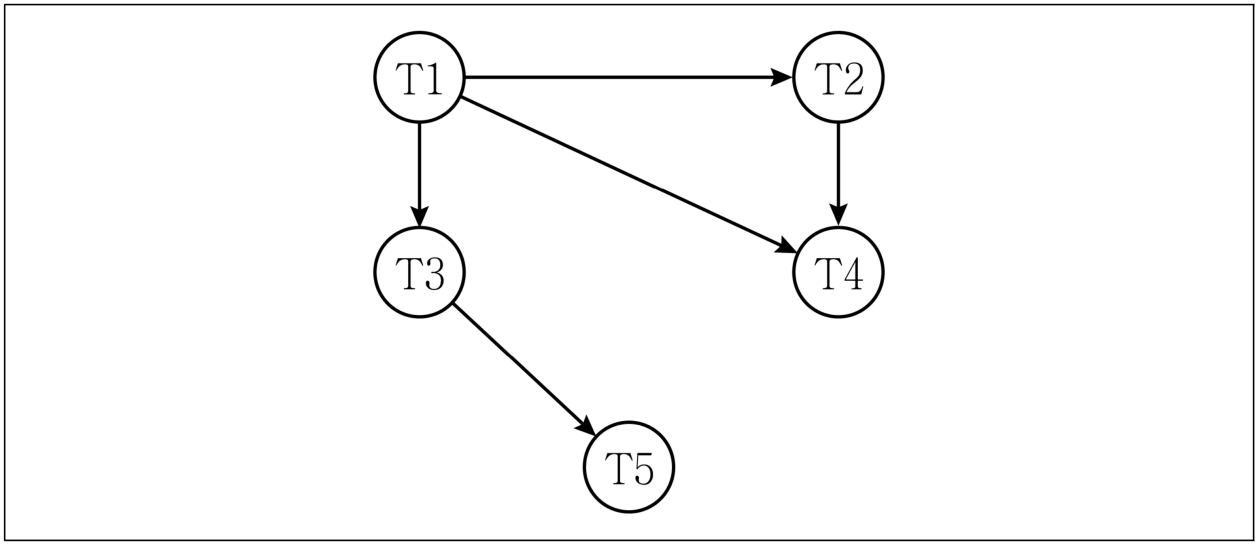
s4: (t1, u1)만 있고 (t1, u2)가 없으므로 제외한다.

s5: A2 값이 t2인 튜플만 있으므로, R2가 요구하는 (t1, u1)과 (t1, u2)가 모두 없다. 따라서 제외한다.

연산 결과는 A1 열에 s2와 s3을 갖는 릴레이션이며, ②의 표와 일치한다.

NoteBase

문 17. 다음은 5개 트랜잭션의 충돌 직렬가능(conflict serializable) 스케줄에 대한 우선순위 그래프이다. 이에 대한 설명으로 옳은 것만을 모두 고르면?



ㄱ. 동등한 직렬 스케줄은 6개이다.

ㄴ. 모든 동등한 직렬 스케줄은 T1에서 시작하고 T5에서 종료한다.

ㄷ. T2와 T3은 동시에 수행할 수 있지만, T5는 T4가 수행된 후에 수행해야 한다.

ㄹ. T2와 T5는 같은 데이터 항목에 대한 write 연산이 없다.

① ㄱ, ㄷ

② ㄱ, ㄹ

③ ㄴ, ㄷ

④ ㄴ, ㄹ

정답 2번

간단한 해설

우선순위 그래프의 화살표는 동등한 직렬 스케줄에서 지켜야 할 선후 관계이다. T1 이후에는 **T2 → T4와 T3 → T5**만 유지하면 되므로 T5가 반드시 마지막일 필요는 없다. 또한 **T2와 T5 사이에 간선이 없으므로** 두 트랜잭션이 같은 데이터 항목을 쓰는 충돌도 없다.

NoteBase

보기별 해설

우선순위 그래프에서 A → B는 A의 연산이 B의 연산보다 먼저 수행되는 충돌 관계를 나타내며, 동등한 직렬 스케줄에서도 A가 B보다 앞서야 한다. 충돌은 서로 다른 트랜잭션이 같은 데이터 항목에 접근하고 적어도 하나가 write인 경우에 발생한다. 동등한 직렬 스케줄은 그래프의 모든 화살표 방향을 지키도록 트랜잭션을 나열한 순서이다.

ㄱ. O: T1은 T2·T3·T4보다 앞서고, T3을 거쳐 T5보다도 앞서므로 항상 첫 번째이다. 남은 네 자리에는 **T2가 T4보다 앞서고 T3이 T5보다 앞선다**는 조건만 적용된다. 이 네 자리 중 T2가 첫째이면 T4의 위치에 따라 3가지, 둘째이면 첫째는 T3이고 나머지 두 자리의 배치가 2가지, 셋째이면 T3 → T5 → T2 → T4의 1가지이다. 따라서 동등한 직렬 스케줄은 3 + 2 + 1 = 6개이다.

ㄴ. X: 모든 동등한 직렬 스케줄이 T1에서 시작하지만, **마지막은 T4 또는 T5**가 될 수 있다. 예를 들어 T1 → T2 → T3 → T5 → T4도 모든 화살표의 선후 관계를 지킨다. 따라서 T5에서 반드시 종료한다는 부분이 잘못되었다.

ㄷ. X: T2와 T3 사이에는 어느 방향으로도 간선이나 경로가 없으므로, T1 이후 두 트랜잭션 사이의 선후 제약 없이 동시에 수행할 수 있다. 그러나 T4와 T5 사이에도 선후 제약이 없다. **T5가 기다려야 하는 것은 T4가 아니라 T3**이며, T4보다 먼저 수행될 수도 있다.

ㄹ. O: T2와 T5가 같은 데이터 항목에 write 연산을 한다면 write-write 충돌이 발생하여 두 트랜잭션 사이에 어느 한 방향의 간선이 있어야 한다. 그런데 그래프에는 **T2와 T5를 직접 연결하는 간선이 없다**. 따라서 두 트랜잭션이 공통으로 쓰는 데이터 항목은 없다. 이는 각각 서로 다른 항목에 write하는 것까지 금지한다는 뜻은 아니다.

옳은 보기는 ㄱ과 ㄹ이며, 이 조합은 ②에 해당한다.

NoteBase

문 18. 다음 SQL문을 통해 생성된 학생 테이블에 새로운 속성으로 '나이'를 추가하고자 한다. 이를 위한 SQL문으로 옳은 것은? (단, '나이'의 데이터 타입은 CHAR(3)이다)

```
CREATE TABLE 학생 (
    학번        CHAR(10)    PRIMARY KEY,
    이름        CHAR(20),
    주소        CHAR(50),
    전화번호    CHAR(13)    );
```

① INSERT INTO 학생 VALUES 나이 CHAR(3);

② ALTER TABLE 학생 INSERT 나이 CHAR(3);

③ ALTER TABLE 학생 ADD CONSTRAINT 나이 CHAR(3);

④ ALTER TABLE 학생 ADD 나이 CHAR(3);

정답 4번

간단한 해설

기존 테이블에 새 속성인 열을 추가하려면 **ALTER TABLE의 ADD**를 사용한다. 테이블명 뒤에 ADD와 속성명, 데이터 타입을 차례로 쓰므로 'ALTER TABLE 학생 ADD 나이 CHAR(3);'이 된다.

핵심 풀이

제시된 CREATE TABLE문은 학번·이름·주소·전화번호를 속성으로 갖는 학생 테이블을 생성한다. 여기에 나이를 추가하는 것은 학생 데이터 한 행을 넣는 작업이 아니라 **테이블의 구조를 변경**하는 작업이므로 ALTER TABLE을 사용한다.

새 열을 추가하는 기본 형식은 'ALTER TABLE 테이블명 ADD 속성명 데이터타입;'이다. 테이블명은 학생, 속성명은 나이, 데이터 타입은 CHAR(3)이므로 'ALTER TABLE 학생 ADD 나이 CHAR(3);'으로 작성한다.

INSERT INTO는 테이블에 **새 행을 삽입**할 때 사용한다. 새 열을 정의하는 명령이 아니며, ALTER TABLE 뒤에서도 열 추가에 INSERT를 사용하지 않는다.

ADD CONSTRAINT는 기본키·외래키 등의 **제약조건을 추가**할 때 사용한다. 이 문제는 제약조건이 아니라 나이라는 열과 그 데이터 타입을 정의하는 작업이므로 ADD 뒤에 바로 '나이 CHAR(3)'을 쓴다.

NoteBase

문 19. 도서관 데이터베이스의 도서 테이블이 '도서ID', '도서명', '대출횟수'를 포함하여 총 30개의 속성으로 구성되어 있다. 100번 이상 대출된 도서만을 대상으로 별도의 서비스를 준비하기 위해 다음과 같이 '인기도서' 뷰를 생성하였을 때 이에 대한 설명으로 옳지 않은 것은? (단, 도서 테이블의 기본키는 '도서ID'이며, 도서 테이블에 '도서ID'가 123인 레코드는 저장되어 있지 않다)

```
CREATE VIEW 인기도서
AS SELECT 도서ID, 도서명, 대출횟수
        FROM 도서 WHERE 대출횟수 >= 100
WITH CHECK OPTION;
```

① '인기도서' 뷰를 정의할 때 도서 테이블에서 보안이 필요한 속성들은 제외함으로써 보안성을 높일 수 있다.

② '인기도서' 뷰를 삭제하더라도 도서 테이블은 삭제되지 않는다.

③ 'SELECT 도서명 FROM 인기도서 WHERE 대출횟수 <= 500;'을 수행하면 100번 이상 500번 이하 대출된 도서의 '도서명'이 검색된다.

④ 'INSERT INTO 인기도서 VALUES(123, '데이터베이스', 5);'를 수행하면 도서 테이블에 레코드가 추가된다.

정답 4번

간단한 해설

WITH CHECK OPTION은 뷰를 통한 삽입·수정 결과가 **뷰의 검색 조건을 만족하도록** 강제한다. 대출횟수 5는 '100번 이상'이라는 조건에 맞지 않으므로, 해당 삽입은 거부되어 도서 테이블에 레코드가 추가되지 않는다.

NoteBase

선지별 해설

① O: 뷰는 기본 테이블에서 필요한 행과 열을 골라 보여 주는 가상 테이블이다. 제시된 '인기도서' 뷰는 도서 테이블의 30개 속성 중 도서ID·도서명·대출횟수만 제공한다. 보안이 필요한 속성을 제외하고 사용자가 **허용된 뷰를 통해서만 접근**하도록 하면, 그 속성을 노출하지 않아 보안성을 높일 수 있다.

② O: 뷰를 삭제하는 것은 **뷰의 정의를 제거**하는 것이다. 기반이 되는 도서 테이블 자체나 그 안에 저장된 레코드를 삭제하는 것이 아니므로, '인기도서' 뷰를 삭제해도 도서 테이블은 남는다.

③ O: '인기도서' 뷰에는 대출횟수가 100번 이상인 도서만 나타난다. 이 뷰를 조회하면서 대출횟수 500번 이하라는 조건을 추가하면 **두 조건을 모두 만족**하는 도서만 선택된다. 따라서 100번 이상 500번 이하 대출된 도서의 도서명이 검색되며, 100번과 500번도 포함된다.

④ X: 뷰 정의 끝의 WITH CHECK OPTION은 뷰를 통해 레코드를 삽입하거나 수정할 때, 그 결과가 **뷰의 WHERE 조건을 만족해야 한다**는 제약이다. 삽입하려는 레코드의 대출횟수는 5이므로 '대출횟수 100번 이상' 조건을 만족하지 않는다. 도서ID 123이 기존에 없어 기본키 중복은 발생하지 않더라도, 이 조건 위반으로 삽입이 거부되어 도서 테이블에 레코드가 추가되지 않는다.

NoteBase

문 20. 다음은 A고객의 포인트 1000점을 B고객에게 전송하는 SQL문이다. 이 SQL문을 하나의 트랜잭션으로 처리할 때 이에 대한 설명으로 옳지 않은 것은? (단, A고객과 B고객의 'ID'는 각각 123과 456이며, 트랜잭션 수행 전 '포인트'는 각각 5000과 200이다)

```
UPDATE 고객
      SET 포인트 = 포인트 - 1000
      WHERE ID = 123;      ..... (가)
UPDATE 고객
      SET 포인트 = 포인트 + 1000
      WHERE ID = 456;      ..... (나)
```

① 동시 공유라는 데이터베이스의 특성으로 인해 (가)까지 수행된 결과를 다른 트랜잭션에서 접근할 수 있도록 허용해야 한다.

② (가)까지 수행한 직후 시스템이 다운되었다면, 시스템이 재가동된 후에 고객 테이블에서 A고객의 포인트는 5000점으로 되돌려야 한다.

③ 트랜잭션이 (나)까지 성공적으로 수행된 후에 고객 테이블에서 A고객과 B고객의 포인트 합은 5200점이다.

④ 트랜잭션이 (나)까지 성공적으로 완료되면 A고객과 B고객의 포인트 변경 결과는 장애가 발생하더라도 손실되지 않도록 보장되어야 한다.

정답 1번

간단한 해설

(가)는 A고객의 포인트만 차감된 중간 상태이므로, 동시 공유를 이유로 다른 트랜잭션에 공개해야 하는 것은 아니다. 트랜잭션의 **격리성**에 따라 다른 트랜잭션이 미완료 작업의 중간 결과를 이용하지 않도록 해야 한다.

NoteBase

선지별 해설

① X: 동시 공유는 여러 트랜잭션이 데이터베이스를 함께 이용할 수 있다는 뜻이며, 미완료 트랜잭션의 중간 결과까지 공유해야 한다는 뜻은 아니다. (가) 직후에는 A고객의 포인트만 5000점에서 4000점으로 줄고 B고객은 아직 200점이므로, 포인트 전송이 끝나지 않았다. **격리성**에 따라 다른 트랜잭션이 이러한 중간 결과를 이용하지 않도록 해야 한다.

② O: 하나의 트랜잭션으로 묶인 두 UPDATE문은 모두 반영되거나 모두 취소되어야 한다. 이를 **원자성**이라고 한다. (가) 직후 장애가 발생하면 B고객에게 1000점을 더하는 작업이 수행되지 않았으므로, A고객의 차감도 취소하여 포인트를 원래의 5000점으로 복구해야 한다.

③ O: (가)에서 A고객의 포인트는 5000 - 1000 = 4000점이 되고, (나)에서 B고객의 포인트는 200 + 1000 = 1200점이 된다. 두 고객의 합은 4000 + 1200 = 5200점으로, 수행 전의 5000 + 200 = 5200점과 같다. 고객 사이에서 포인트를 옮기는 작업이므로 **전체 포인트 합이 보존**된다.

④ O: 트랜잭션이 성공적으로 완료되어 커밋되면 그 변경 결과는 이후 장애가 발생하더라도 유지되어야 한다. 이를 **지속성**이라고 한다. 따라서 A고객 4000점, B고객 1200점이라는 완료 결과가 장애로 사라지지 않도록 보장해야 한다.

NoteBase