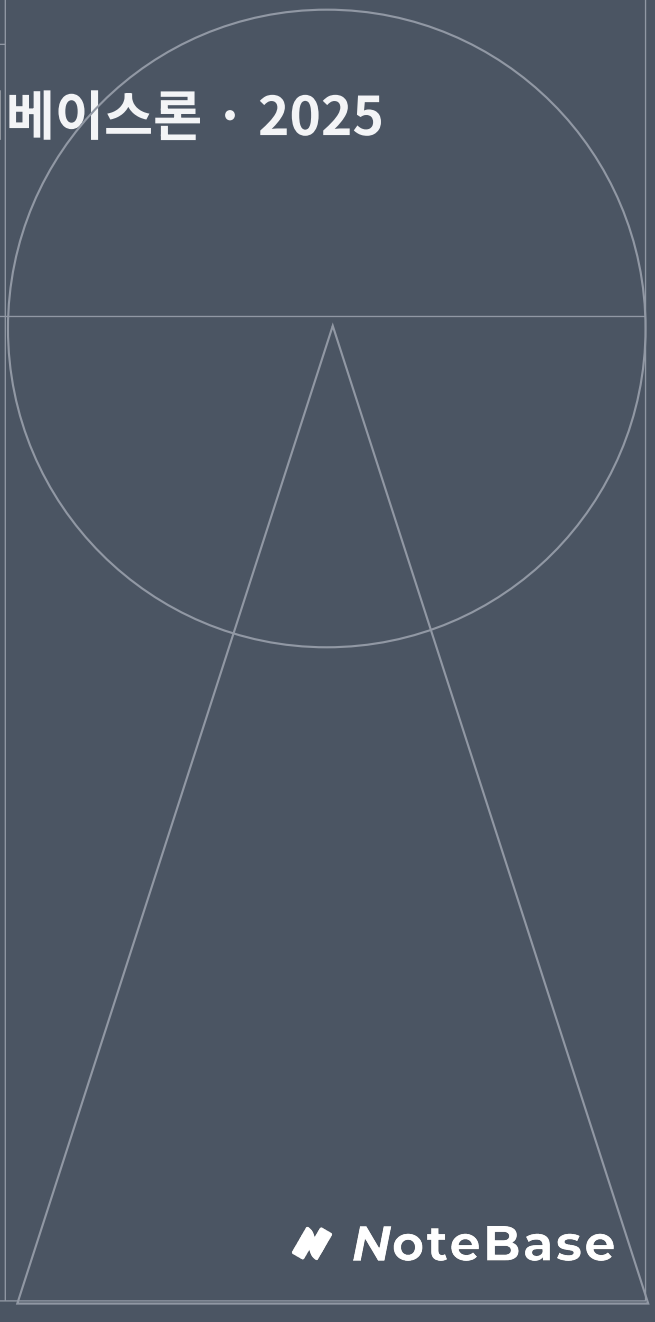
기출OX 해설집

공무원/국가직/9급 · 데이터베이스론 · 2025



NoteBase

기출OX 해설집

공무원/국가직/9급 · 데이터베이스론 · 2025

공무원/국가직/9급 · 데이터베이스론 · 2025

문 1. 다음 상황에 대한 트랜잭션(transaction)의 특성은?

제품의 재고량을 갱신하는 트랜잭션이 커밋(commit) 되기 전에 시스템 장애가 발생했다. 장애 후 데이터베이스는 트랜잭션이 수행한 변경 사항을 모두 취소하였고, 시스템은 장애 이전의 상태로 복구되었다.

① 일관성(consistency)
② 원자성(atomicity)
③ 지속성(durability)
④ 고립성(isolation)

정답 2번

간단한 해설

커밋 전에 장애가 발생하여 변경 사항을 **모두 취소**한 것은 트랜잭션을 전부 수행하거나 전혀 수행하지 않은 상태로 처리하는 **원자성**을 보여 준다.

핵심 풀이

원자성(atomicity)은 트랜잭션에 포함된 연산을 하나의 작업 단위로 취급하여, **전체가 반영되거나 아무것도 반영되지 않도록** 보장하는 특성이다. 제시된 상황에서는 재고량을 갱신하던 중 커밋 전에 장애가 발생하였으므로, 이미 수행한 변경도 모두 취소하여 해당 트랜잭션의 실행 전 상태로 되돌렸다. 일부 변경만 남기지 않았다는 점이 원자성에 해당한다.

지속성(durability)은 **커밋된 트랜잭션의 결과를 장애 후에도 유지**하는 특성이다. 이 상황은 커밋된 결과를 보존한 것이 아니라, 커밋되지 않은 변경을 모두 취소한 것이므로 지속성과 구별된다.

NoteBase

문 2. 데이터를 구조 형태에 따라 구분할 때 이에 대한 설명으로 옳지 않은 것은?

① XML 문서는 데이터를 교환하는 용도로 활용되고 있다.

② 정형 데이터(structured data)는 미리 정해진 구조에 따라 데이터를 저장하며 관계 데이터베이스에서 널리 사용된다.

③ 비정형 데이터(unstructured data)에는 사전에 데이터 타입이 정해지지 않은 텍스트도 있다.

④ 반정형 데이터(semi-structured data)는 데이터 구조와 내용을 별도의 파일 형태로 저장한다.

정답 4번

간단한 해설

반정형 데이터는 XML의 태그처럼 **데이터 내부에 구조 정보를 함께 표현**한다. 구조와 내용을 별도 파일로 나누어 저장하는 것이 반정형 데이터의 특징은 아니다.

선지별 해설

① O: XML은 태그로 데이터의 의미와 계층 관계를 표현하는 문서 형식이다. 서로 다른 시스템에서도 태그와 값을 해석할 수 있어 **데이터 교환**에 활용된다.

② O: 정형 데이터는 **미리 정해진 구조**에 맞추어 저장하는 데이터이다. 관계 데이터베이스의 테이블은 열마다 이름과 자료형을 정하고 각 행에 그 구조에 맞는 값을 저장한다. 예를 들어 학생 테이블에서 학번은 정수, 이름은 문자열로 정해 저장하는 방식이다.

③ O: 자유롭게 작성한 글처럼 **미리 정해진 항목 구조가 없는 텍스트**는 비정형 데이터에 해당한다. 여기서 구조가 없다는 것은 컴퓨터에서 문자열로 저장할 수 없다는 뜻이 아니라, 관계형 테이블처럼 내용을 일정한 열과 자료형에 맞추어 조직하지 않았다는 뜻이다.

④ X: 반정형 데이터는 고정된 테이블 구조를 엄격히 따르지는 않지만, 태그나 키 등으로 **구조 정보를 데이터와 함께 표현**한다. 대표적으로 XML 문서는 태그가 항목의 의미와 계층 관계를 나타내고 그 안에 실제 값을 담는다. 외부에 스키마 파일을 둘 수도 있으나, 구조와 내용을 별도 파일로 분리해 저장하는 것이 반정형 데이터의 정의는 아니다.

NoteBase

문 3. 다음 릴레이션(relation) R과 S의 카티션 프로덕트(cartesian product) 연산 시 결과 릴레이션의 차수(degree)와 카디널리티(cardinality)값은?

R	
번호	이름
100	강감찬
200	이순신
300	한용운

S	
번호	나이
100	82
101	53
102	66

① 차수: 3, 카디널리티: 6

② 차수: 4, 카디널리티: 6

③ 차수: 3, 카디널리티: 9

④ 차수: 4, 카디널리티: 9

정답 4번

간단한 해설

카티션 프로덕트는 두 릴레이션의 **모든 튜플 쌍**을 결합하므로 속성 수는 더하고 튜플 수는 곱한다. R과 S는 각각 속성 2개, 튜플 3개이므로 결과의 차수는 4, 카디널리티는 9이다.

NoteBase

계산 과정

차수는 릴레이션의 속성 수, 카디널리티는 튜플 수이다. 그림에서 R은 번호·이름의 속성 2개와 튜플 3개, S는 번호·나이의 속성 2개와 튜플 3개를 가진다. 표의 제목과 속성명 행은 튜플 수에 포함하지 않는다.

차수 계산: 카티션 프로덕트는 R의 각 튜플에 S의 튜플을 이어 붙이므로 두 릴레이션의 속성을 모두 유지한다. 양쪽에 번호라는 같은 속성명이 있어도 **하나의 속성으로 합치지 않는다**. R.번호, R.이름, S.번호, S.나이로 구분하면 결과의 차수는 2 + 2 = 4이다.

카디널리티 계산: **번호 값의 일치 여부와 관계없이** R의 각 튜플을 S의 모든 튜플과 결합한다. 예를 들어 R의 (100, 강감찬)은 S의 (100, 82), (101, 53), (102, 66)과 각각 결합하여 결과 튜플 3개를 만든다. R의 나머지 두 튜플도 각각 3개씩 만들므로 결과의 카디널리티는 3 × 3 = 9이다.

결과 릴레이션은 속성 4개와 튜플 9개를 가지므로, 차수 4·카디널리티 9인 ④와 일치한다.

NoteBase

문 4. 트랜잭션에 대한 설명으로 옳지 않은 것은?

① 커밋된 후에도 해당 트랜잭션은 롤백(rollback)을 사용하여 취소가 가능하다.

② 현재 수행 중인 트랜잭션을 롤백하면 트랜잭션에서 SQL 문에 의해 수행된 모든 갱신이 취소(undo)된다.

③ 현재 수행 중인 트랜잭션을 커밋하면 트랜잭션이 수행한 갱신을 데이터베이스에 영구적으로 반영한다.

④ 트랜잭션은 데이터베이스 관리 시스템(database management system)에서 데이터를 다루는 논리적인 작업의 단위이다.

정답 1번

간단한 해설

커밋은 트랜잭션의 변경 내용을 확정하는 연산이므로, **커밋된 트랜잭션은 롤백으로 취소할 수 없다**. 롤백은 아직 커밋하지 않은 트랜잭션의 변경 내용을 취소한다.

선지별 해설

① X: 커밋은 트랜잭션의 변경 내용을 확정하고 해당 트랜잭션을 종료한다. 따라서 '커밋된 후에도 롤백으로 취소가 가능하다'는 설명은 잘못이며, **커밋하기 전의 변경 내용만 롤백으로 취소할 수 있다**.

② O: 현재 수행 중인 트랜잭션 전체를 롤백하면 그 트랜잭션에서 SQL 문으로 수행한 모든 갱신을 취소하여 트랜잭션 시작 전 상태로 되돌린다. 이는 트랜잭션의 작업을 **전부 반영하거나 전부 취소**하는 원자성과 연결된다.

③ O: 커밋은 트랜잭션이 수행한 갱신을 데이터베이스에 영구적으로 반영하는 연산이다. 성공적으로 커밋된 결과는 이후 시스템에 장애가 발생해도 보존되어야 하며, 이러한 성질을 **영속성**이라고 한다.

④ O: 트랜잭션은 데이터베이스에서 **하나의 논리적 작업으로 처리하는 단위**이며, 하나 또는 여러 SQL 문으로 구성될 수 있다. 예를 들어 계좌 이체에서는 출금과 입금을 하나의 트랜잭션으로 묶어 두 갱신을 모두 반영하거나 모두 취소하도록 처리한다.

NoteBase

문 5. '학생' 테이블에 대한 검색 권한을 사용자 '홍길동'에게 부여하는 SQL 문을 완성할 때 (가)와 (나)에 들어갈 내용을 바르게 연결한 것은?

(가) (나) ON 학생 TO 홍길동;

① (가): SELECT, (나): REVOKE

② (가): SELECT, (나): GRANT

③ (가): GRANT, (나): SELECT

④ (가): REVOKE, (나): SELECT

정답 3번

간단한 해설

권한을 부여할 때는 **GRANT**를 사용하고, 부여할 검색 권한은 **SELECT**로 지정한다. ON 뒤에는 대상 테이블을, TO 뒤에는 권한을 받을 사용자를 쓴다.

빈칸별 해설

(가) GRANT: 사용자에게 테이블에 대한 권한을 **부여하는 명령**이다. 문제는 홍길동에게 권한을 주는 상황이므로 GRANT를 쓴다. REVOKE는 부여된 권한을 회수할 때 사용하므로 이 자리에 들어갈 수 없다.

(나) SELECT: 테이블의 데이터를 **검색할 수 있는 권한**을 뜻한다. 이 문장에서 GRANT 바로 뒤는 부여할 권한을 지정하는 자리이므로 SELECT를 쓴다. ON 학생은 권한의 대상 테이블을, TO 홍길동은 권한을 받는 사용자를 나타낸다.

완성된 문장은 GRANT SELECT ON 학생 TO 홍길동;이다. (가)에 GRANT, (나)에 SELECT가 들어가는 조합이므로 ③과 일치한다.

NoteBase

문 6. SQL에서 UNION과 UNION ALL에 대한 설명으로 옳은 것은?

① UNION은 UNION ALL과 달리 ORDER BY 절을 자동으로 추가한다.

② UNION은 UNION ALL보다 연산 결과 릴레이션의 속성 개수가 크다.

③ UNION은 중복된 투플을 제거하며, UNION ALL은 중복된 투플을 포함한다.

④ UNION은 두 릴레이션에서 서로 대응되는 속성의 도메인(domain)이 다를 때만 사용할 수 있다.

정답 3번

간단한 해설

UNION은 두 조회 결과를 합치면서 **중복된 투플을 제거**하고, UNION ALL은 **중복을 그대로 유지**한다. 중복 여부는 특정 속성 하나가 아니라 투플 전체의 값으로 판단한다.

선지별 해설

① X: UNION이 ORDER BY 절을 자동으로 추가하는 것은 아니다. 중복 제거 과정에서 내부적으로 정렬이 사용될 수 있어도 결과의 출력 순서가 보장되지는 않는다. 원하는 순서로 출력하려면 **ORDER BY 절을 명시**해야 한다.

② X: UNION과 UNION ALL은 모두 두 조회 결과를 세로로 합치는 연산이므로 **결과의 속성 개수는 같다**. 두 입력의 속성 개수가 같아야 하며, 중복 제거 여부에 따라 달라질 수 있는 것은 속성 수가 아니라 투플 수이다.

③ O: UNION은 **모든 속성값이 같은 투플**을 중복으로 보아 하나만 남기고, UNION ALL은 중복된 투플도 모두 남긴다. 예를 들어 한 속성으로 된 두 조회 결과에 각각 값 1·2와 값 2·3이 있으면, UNION 결과에는 값 2가 한 번만 나타나고 UNION ALL 결과에는 두 번 나타난다.

④ X: UNION은 두 조회 결과의 속성 개수가 같고, 서로 대응되는 속성의 **자료형이 호환되어야** 사용할 수 있다. 대응 속성의 도메인이 다를 때만 사용하는 것이 아니라, 같은 도메인이거나 합칠 수 있는 호환 자료형이어야 한다. 이 조건은 UNION ALL에도 적용된다.

NoteBase

문 7. 다음 두 릴레이션 학생과 수강에서 아래 관계대수에 의해 생성된 결과로 옳은 것은? (단, 수강 릴레이션의 SID는 학생 릴레이션의 SID를 참조하는 외래키(foreign key)이다)

학생

SID	Name	Dept
101	홍길동	컴퓨터공학
102	이몽룡	경영학
103	성춘향	컴퓨터공학
104	장보고	경영학

수강

SID	CID	Grade
101	CS101	A
101	CS102	B
102	MG201	C
103	CS101	B
104	MG202	A

$$\pi_{\text{Name, CID}}(\sigma_{\text{Dept}='\text{컴퓨터공학}' \vee \text{Grade}='\text{B}'}(\text{학생} \bowtie_{\text{N}} \text{수강}))$$

①

Name	CID
홍길동	CS101
홍길동	CS102
성춘향	CS101

②

Name	CID
홍길동	CS102
성춘향	CS101

③

Name	CID
이몽룡	MG201
장보고	MG202

④

Name	CID
홍길동	CS102

정답 1번

간단한 해설

선택 조건은 컴퓨터공학 소속이거나 성적이 B인 경우이므로, **두 조건 중 하나만 충족해도** 포함된다. 홍길동은 성적 A인 CS101 수강 행까지 남아, 최종 결과는 홍길동의 두 수강 행과 성춘향의 한 수강 행이다.

NoteBase

계산 과정

1. 자연 조인: 학생과 수강의 공통 속성인 **SID가 같은 행끼리 연결**한다. 수강의 SID가 학생의 SID를 참조하므로 각 수강 행에 해당 학생의 Name과 Dept가 붙는다. 홍길동은 CS101·A와 CS102·B의 2개 행, 이몽룡은 MG201·C의 1개 행, 성춘향은 CS101·B의 1개 행, 장보고는 MG202·A의 1개 행으로 연결되어 총 5개 행이 생성된다.

2. 선택: 조인 결과에서 'Dept가 컴퓨터공학이거나 Grade가 B'인 행을 남긴다. 조건을 연결하는 ∨는 **둘 중 하나 이상이 참이면 되는 OR**이다. 홍길동은 컴퓨터공학 소속이므로 CS101·A와 CS102·B가 모두 남는다. 성춘향의 CS101·B도 조건을 충족한다. 이몽룡의 MG201·C와 장보고의 MG202·A는 경영학 소속이면서 성적도 B가 아니므로 제외된다. 이 단계에서 3개 행이 남는다.

3. 투영: 남은 행에서 Name과 CID 두 속성만 추출한다. 결과는 (홍길동, CS101), (홍길동, CS102), (성춘향, CS101)의 3개 튜플이다. 투영에서는 중복 튜플을 제거하지만, 여기서는 세 쌍이 서로 달라 모두 유지된다.

이 결과는 ①의 릴레이션과 일치한다.

NoteBase

문 8. NoSQL 데이터베이스에 대한 설명으로 옳은 것은?

① 문서 기반(document-based) 데이터베이스는 주로 노드(node)와 간선(edge)을 사용하여 데이터를 관리한다.

② 키-값(key-value) 데이터베이스는 분산하여 저장하는 시스템 환경에 적용하기에는 적합하지 않다.

③ 트랜잭션을 지원하여 ACID(Atomicity Consistency Isolation Durability)를 필수적으로 보장한다.

④ 문서 기반 데이터베이스는 반정형 데이터 형식을 지원한다.

정답 4번

간단한 해설

문서 기반 데이터베이스는 JSON처럼 필드 이름과 계층 구조를 가진 **반정형 데이터**를 저장할 수 있다. 모든 문서에 동일한 필드 구성을 강제하지 않아 서로 다른 구조의 데이터도 유연하게 다룬다.

선지별 해설

① X: 노드로 개체를, 간선으로 개체 사이의 관계를 표현하는 것은 **그래프 데이터베이스**이다. 문서 기반 데이터베이스는 필드와 값으로 구성된 문서를 단위로 데이터를 저장한다.

② X: 키-값 데이터베이스는 고유한 키에 값을 연결하여 저장하고, 키로 해당 값을 조회한다. 키를 기준으로 데이터를 여러 서버에 나누어 저장할 수 있어 **분산 저장과 수평 확장에 적합하다**. 예를 들어 키의 해시값을 이용해 각 데이터를 저장할 서버를 정할 수 있다.

③ X: ACID는 트랜잭션의 원자성·일관성·격리성·지속성을 뜻한다. NoSQL이라는 분류 자체가 **ACID 보장을 필수로 요구하지는 않는다**. 트랜잭션 지원 범위와 일관성 보장 방식은 시스템마다 다르며, ACID 트랜잭션을 지원하는 NoSQL도 있으므로 트랜잭션을 전혀 지원하지 않는다고 이해해서도 안 된다.

④ O: 문서 기반 데이터베이스는 JSON 등의 **반정형 데이터 형식을 지원한다**. 반정형 데이터는 필드 이름이나 중첩 구조로 데이터의 의미와 구조를 표현하면서도, 관계형 테이블처럼 모든 데이터가 동일한 필드 구성을 따를 필요는 없다. 예를 들어 한 회원 문서에는 이름과 이메일을, 다른 회원 문서에는 이름과 전화번호를 저장할 수 있다.

NoteBase

문 9. 다음 릴레이션에서 아래 SQL문을 실행하면 나오는 투플의 개수는?

사원

사원번호	이름	부서번호	월급	직책
1	홍길동	10	5000	사원
2	이순신	20	3000	대리
3	강감찬	30	4000	과장
4	장보고	20	2000	부장
5	홍길동	20	5000	대리
6	안중근	10	2000	과장

```
SELECT 사원번호, 월급, 부서번호
FROM 사원
WHERE 부서번호 IN (SELECT 부서번호
                   FROM 사원
                   WHERE 이름 = '홍길동');
```

① 2
② 3
③ 4
④ 5

정답 4번

간단한 해설

하위 질의가 반환하는 부서번호는 10과 20이며, 바깥 질의는 **이 두 부서의 모든 사원**을 조회한다. 부서별 사원 수가 2명과 3명이므로 결과 투플은 5개이다.

NoteBase

계산 과정

하위 질의: 이름이 '홍길동'인 사원은 사원번호 1과 5이고, 각각 부서번호가 10과 20이다. 따라서 하위 질의의 결과는 부서번호 10, 20이다.

바깥 질의: IN은 부서번호가 하위 질의 결과 중 하나와 일치하는지 검사한다. 따라서 **이름과 관계없이 부서번호가 10 또는 20인 모든 사원**이 조회된다. 부서번호 10에는 사원번호 1·6의 2명, 부서번호 20에는 사원번호 2·4·5의 3명이 있다. 부서번호가 30인 사원번호 3은 제외된다.

투플 개수: SELECT는 조건을 만족한 각 사원에서 사원번호·월급·부서번호를 출력한다. 선택한 속성이 3개라는 것은 결과의 열 개수이며, 투플 개수는 조회된 행의 수이다. 두 부서에 속한 사원을 합하면 2 + 3 = 5개이므로 ④의 5에 해당한다.

NoteBase

문 10. 다음 트랜잭션 T1과 T2를 수행한 후 X와 Y의 최종값은? (단, X와 Y의 초깃값은 각각 100, 200이다)

시간 ↓

T1	T2
X = 20	
write(X)	
	read(X)
	X = X - 20
	write(X)
	read(Y)
	Y = Y + 20
	write(Y)

① X: 0, Y: 220

② X: 20, Y: 200

③ X: 80, Y: 220

④ X: 100, Y: 200

정답 1번

간단한 해설

T2는 X의 초깃값 100이 아니라 **T1이 기록한 20**을 읽어 20을 빼므로 X는 0이 된다. Y는 초깃값 200에 20을 더해 220이 된다.

NoteBase

계산 과정

표의 시간 화살표를 따라 위에서 아래로 실행한다. read는 데이터베이스의 값을 트랜잭션 내부로 읽고, write는 내부에서 계산한 값을 데이터베이스에 기록하는 연산이다. 처음 데이터베이스에는 X = 100, Y = 200이 저장되어 있다.

T1의 X 처리: X = 20은 **X에 20을 대입**하는 연산이다. 이어서 write(X)를 실행하므로 데이터베이스의 X는 100에서 20으로 바뀐다.

T2의 X 처리: read(X)는 앞서 T1이 기록한 **현재 값 20**을 읽는다. 여기에서 20을 빼면 20 - 20 = 0이며, write(X)를 실행한 뒤 데이터베이스의 X는 0이 된다.

T2의 Y 처리: 앞선 연산들은 Y를 바꾸지 않았으므로 read(Y)는 초깃값 200을 읽는다. 여기에 20을 더하면 200 + 20 = 220이며, write(Y)를 실행한 뒤 데이터베이스의 Y는 220이 된다.

최종값은 X = 0, Y = 220으로 ①과 일치한다.

NoteBase

문 11. 트리거(trigger)에 대한 설명으로 옳지 않은 것은?

① SQL 표준에 채택된 적은 없으나 실무상 유용하게 사용한다.

② 실행될 시점(event)과 이때 수행되어야 할 동작(action)으로 구성된다.

③ 무결성 제약 조건을 구현하는데 사용할 수 있다.

④ 데이터베이스에서 발생하는 특정 사건에 대하여 자동적으로 수행하는 구문이다.

정답 1번

간단한 해설

트리거는 **SQL:1999 표준에 포함**되었다. 따라서 'SQL 표준에 채택된 적은 없다'는 설명이 잘못되었다.

선지별 해설

① X: 트리거를 정의하는 CREATE TRIGGER는 **SQL:1999 표준에 포함**되어 있다. 실무에서 유용하게 사용한다는 부분은 맞지만, SQL 표준에 채택된 적이 없다는 부분은 잘못되었다.

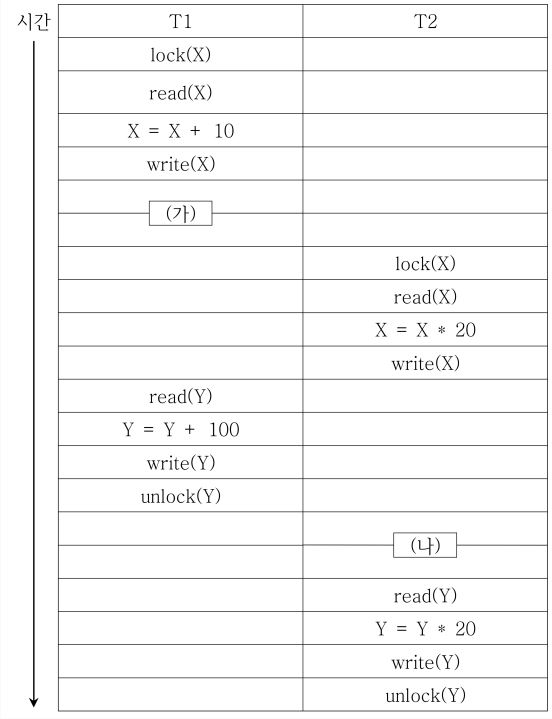
② O: 이벤트(event)는 INSERT·UPDATE·DELETE처럼 **트리거의 실행 계기가 되는 사건**이고, 동작(action)은 그 사건에 대응하여 수행할 처리이다. 예를 들어 UPDATE가 발생하면 변경 이력을 저장하도록 정의할 수 있다. 사건의 발생 전(BEFORE) 또는 후(AFTER)처럼 실행 시점을 지정하고, 동작을 수행할 조건을 추가할 수도 있다.

③ O: 트리거는 데이터가 변경될 때 제약 위반 여부를 검사하여 **무결성 제약 조건을 구현**할 수 있다. 예를 들어 급여를 음수로 변경하려는 UPDATE에 대해 오류를 발생시키도록 하면, 잘못된 값이 저장되는 것을 막을 수 있다.

④ O: 트리거는 데이터베이스에 정의해 둔 특정 사건이 발생하면 **DBMS가 자동으로 실행**한다. 사용자가 매번 별도로 호출하지 않아도, 지정한 사건과 조건에 따라 등록된 동작이 수행된다.

NoteBase

문 12. 다음 트랜잭션 T1과 T2에 대한 2단계 로킹 규약(2-phase locking protocol)을 준수하기 위해 (가)와 (나)에 적합한 코드는?



① (가)
lock(Y)
unlock(X)
(나)
lock(Y)
unlock(X)

② (가)
lock(Y)
unlock(X)
(나)
unlock(X)
lock(Y)

③ (가)
unlock(X)
lock(Y)
(나)
lock(Y)

NoteBase

unlock(X)
④ (가)
unlock(X)
lock(Y)
(나)
unlock(X)
lock(Y)

정답 1번

간단한 해설

2단계 로킹에서는 **잠금을 하나라도 해제한 뒤에는 새 잠금을 획득할 수 없다**. 따라서 X의 잠금을 풀기 전에 Y의 잠금을 확보해야 한다.

빈칸별 해설

lock은 데이터 항목의 잠금을 획득하고, unlock은 잠금을 해제하는 연산이다. 2단계 로킹은 각 트랜잭션을 잠금을 획득하되 해제하지 않는 확장 단계와, 잠금을 해제하되 새로 획득하지 않는 축소 단계로 구분한다. 첫 unlock이 두 단계의 경계이며, 이 규칙은 트랜잭션마다 적용된다.

(가): lock(Y) → unlock(X). T1은 이미 X의 잠금을 보유하고 있으며, 뒤에서 Y를 읽고 수정해야 하므로 Y의 잠금도 필요하다. **X의 잠금을 해제하기 전에 Y의 잠금을 획득**해야 한다. 반대로 unlock(X) → lock(Y)로 두면 축소 단계에 들어간 뒤 새 잠금을 획득하므로 규약을 위반한다. Y의 잠금을 확보해 두었으므로 X의 잠금을 해제한 뒤에도 Y를 읽고 수정할 수 있다.

(나): lock(Y) → unlock(X). T2도 X의 잠금을 보유한 상태에서 Y를 읽고 수정할 준비를 해야 한다. 그림에서는 T1의 unlock(Y) 다음에 (나)가 있으므로 이때 Y의 잠금을 획득할 수 있다. T1이 이미 잠금을 해제했더라도 T2가 잠금을 해제한 것은 아니므로, **T2는 아직 확장 단계**에 있을 수 있다. T2 역시 lock(Y)를 먼저 수행하고 unlock(X)로 축소 단계에 들어가야 하며, 두 연산을 바꾸면 규약을 위반한다.

(가)와 (나)에 모두 lock(Y) → unlock(X)를 넣는 조합은 ①에 해당한다.

NoteBase

문 13. 데이터베이스 접근제어(access control)에 대한 설명으로 옳은 것은?

① 강제 접근제어는 정보를 유출 시키는 트로이 목마 공격에 대하여 임의 접근제어보다 취약하다.

② 임의 접근제어는 데이터 소유자가 데이터에 대한 접근 권한을 제어하며, 주체 간 권한 위임이 가능하다.

③ 역할 기반 접근제어는 역할의 권한 변경 시 사용자마다 변경된 권한을 적용해야 한다.

④ Bell-LaPadula 모델은 허가 등급이 높은 주체가 허가 등급이 낮은 객체를 기록할 수 있다.

정답 2번

간단한 해설

임의 접근제어(DAC)는 **데이터 소유자가 접근 권한을 결정**하는 방식이다. 소유자는 다른 주체에게 권한을 부여할 수 있고, 권한을 받은 주체도 허용된 범위에서 이를 위임할 수 있다.

선지별 해설

① X: 정보를 유출하는 트로이 목마 공격에 더 취약한 것은 **임의 접근제어(DAC)**이다. DAC에서는 사용자 권한으로 실행된 악성 프로그램이 그 사용자가 읽을 수 있는 정보를 다른 객체로 복사하여 유출할 수 있다. 강제 접근제어(MAC)는 소유자의 의사와 관계없이 시스템의 보안 정책과 등급에 따라 정보 흐름을 제한하므로 이러한 유출을 억제한다.

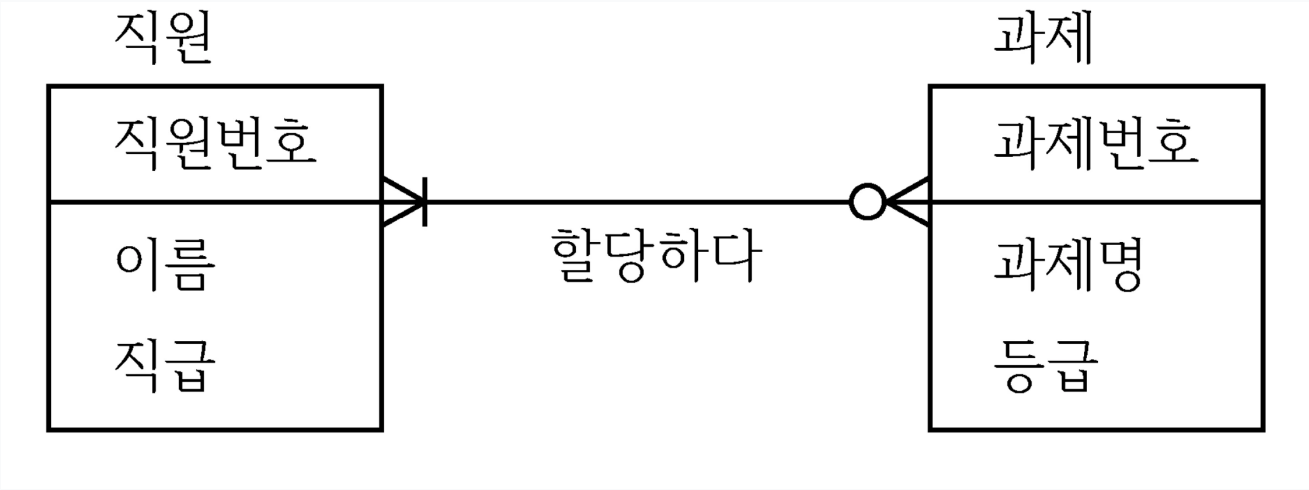
② O: 임의 접근제어는 **데이터 소유자의 재량**으로 다른 주체의 읽기·쓰기 등의 접근 권한을 정한다. 소유자가 다른 사용자에게 권한을 부여할 수 있으며, 권한을 받은 주체도 위임이 허용된 범위에서 다른 주체에게 권한을 넘길 수 있다.

③ X: 역할 기반 접근제어(RBAC)는 사용자에게 권한을 각각 직접 부여하는 대신, **역할에 권한을 부여하고 사용자에게 역할을 할당**한다. 따라서 역할의 권한을 변경하면 그 역할을 통해 접근하는 사용자에게 변경된 권한이 적용되므로, 사용자마다 권한을 따로 수정할 필요가 없다.

④ X: Bell-LaPadula 모델은 기밀성을 보호하기 위해 **하위 등급 객체에 쓰는 것을 금지**한다. 높은 등급으로 동작하는 주체가 낮은 등급의 객체에 기밀 정보를 기록하면 낮은 등급의 사용자에게 정보가 유출될 수 있기 때문이다. 기본 규칙은 상위 등급 읽기 금지(No Read Up)와 하위 등급 쓰기 금지(No Write Down)이다.

NoteBase

문 14. 다음 개체 관계 다이어그램(entity relationship diagram)에 대한 설명으로 옳지 않은 것은?



① 각 과제는 최소한 한 명의 직원에게 할당되어야 한다.

② 동일한 과제가 여러 명에게 할당될 수도 있다.

③ 각 직원은 최소한 한 과제를 할당받아야 한다.

④ 어떤 직원은 과제를 여러 개 할당받을 수도 있다.

정답 3번

간단한 해설

과제 쪽의 원 표시는 직원 한 명에게 연결되는 과제가 **0개일 수도 있음**을 뜻한다. 따라서 과제를 하나도 할당받지 않은 직원도 허용된다.

NoteBase

선지별 해설

이 다이어그램은 까마귀발 표기법을 사용한다. 한쪽 개체 하나에 연결되는 상대 개체의 수는 상대 개체 쪽 끝의 기호로 읽는다. 원은 최소 0, 세로선은 최소 1을 나타내며, 까마귀발은 여러 개까지 연결될 수 있음을 나타낸다.

① O: 직원 쪽 끝의 세로선은 최소 1명을 뜻한다. 따라서 각 과제에는 **최소 한 명의 직원**이 연결되어야 하며, 담당 직원이 없는 과제는 허용되지 않는다.

② O: 직원 쪽 끝의 까마귀발은 한 과제에 연결될 수 있는 직원이 여러 명임을 뜻한다. 따라서 **동일한 과제를 여러 직원에게** 할당할 수 있다.

③ X: 과제 쪽 끝의 원은 직원 한 명이 할당받는 과제의 최소 개수가 0임을 뜻한다. '최소한 한 과제를 할당받아야 한다'가 아니라 **과제를 하나도 할당받지 않을 수도 있다**로 고쳐야 한다.

④ O: 과제 쪽 끝의 까마귀발은 직원 한 명에게 연결될 수 있는 과제가 여러 개임을 뜻한다. 따라서 **한 직원에게 여러 과제**를 할당할 수 있다.

NoteBase

문 15. 다음 트랜잭션 T1과 T2에서 발생하는 문제로 옳은 것은?

시간 ↓

T1	T2
read(X)	
X = X + 100	
	read(X)
	X = X * 2
write(X)	
	write(X)

① 모순성(inconsistency)

② 갱신 분실(lost update)

③ 삽입 이상(insertion anomaly)

④ 연쇄 복귀(cascading rollback)

정답 2번

간단한 해설

T2는 T1이 변경한 값을 저장하기 전에 X를 읽어 이전 값을 기준으로 계산한다. 이후 T2의 쓰기가 **T1의 갱신 결과를 덮어쓰므로** 갱신 분실이 발생한다.

NoteBase

핵심 풀이

그림의 시간은 위에서 아래로 흐른다. T1은 X를 읽어 100을 더하지만, 아직 write(X)를 수행하지 않은 상태에서 T2가 X를 읽는다. 따라서 두 트랜잭션은 **동일한 갱신 전 값**을 기준으로 각각 계산한다. 처음 값을 a라고 하면 T1이 계산한 값은 $a + 100$, T2가 계산한 값은 $2a$이다.

이후 T1의 write(X)가 $a + 100$을 저장하고, T2의 write(X)가 그 자리에 $2a$를 저장한다. T2는 T1의 변경을 반영하지 않은 값으로 덮어쓰기 때문에 **T1이 100을 더한 갱신 효과가 사라진다**. 이처럼 한 트랜잭션의 갱신 결과가 다른 트랜잭션의 쓰기로 소실되는 현상이 갱신 분실(lost update)이다.

연쇄 복귀는 다른 트랜잭션이 기록했지만 아직 확정하지 않은 값을 읽은 뒤, 그 값을 기록한 트랜잭션의 취소 때문에 이를 읽은 트랜잭션도 취소해야 하는 현상이다. 그림에서 T2의 read(X)는 T1의 write(X)보다 먼저이므로 T1이 기록한 미확정 값을 읽은 상황이 아니며, 취소 연산도 제시되어 있지 않다.

NoteBase

문 16. 참조 무결성 제약조건(referential integrity constraint)에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 두 릴레이션의 연관된 투플들 사이의 일관성을 유지하는 데 사용된다.
ㄴ. 외래키는 참조되는 테이블의 기본키 값이 삭제될 경우, 해당키를 외래키로 가지는 모든 투플을 삭제해야만 한다.
ㄷ. 외래키 값은 참조되는 릴레이션의 후보키(candidate key) 값과 일치하거나 NULL이어야 한다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 2번

간단한 해설

참조되는 행을 삭제할 때 관련 행까지 지우는 **연쇄 삭제는 여러 처리 방식 중 하나**이다. 삭제를 거부하거나, NULL이 허용되는 외래키를 NULL로 바꾸는 방식으로도 참조 무결성을 유지할 수 있다.

보기별 해설

ㄱ. O: 참조 무결성은 외래키를 통해 연결된 릴레이션 사이에서 **존재하지 않는 대상을 참조하지 않도록** 하는 제약조건이다. 예를 들어 사원 릴레이션의 부서번호가 부서 릴레이션을 참조한다면, NULL이 아닌 사원의 부서번호는 실제로 존재하는 부서번호여야 한다. 이로써 서로 연관된 투플 사이의 일관성을 유지한다.

ㄴ. X: 참조되는 기본키 값이 삭제될 때, 이를 외래키로 가진 모든 투플을 **반드시 삭제해야 하는 것은 아니다**. 관련 투플까지 삭제하는 것은 CASCADE 방식이다. 참조하는 투플이 있으면 삭제를 거부하는 RESTRICT 방식도 있고, 외래키에 NULL이 허용되면 그 값을 NULL로 바꾸는 SET NULL 방식도 있다. 따라서 '모든 투플을 삭제해야만 한다'는 '설정된 참조 동작에 따라 삭제를 제한하거나 연쇄 삭제 등을 수행한다'로 고쳐야 한다.

ㄷ. O: 외래키의 NULL이 아닌 값은 **참조 대상으로 지정된 후보키 값과 일치**해야 한다. 후보키는 투플을 유일하게 식별하는 데 필요한 최소한의 속성 집합이며, 기본키는 후보키 중 하나를 선택한 것이다. 외래키는 기본키뿐 아니라 참조 대상이 될 수 있도록 유일성이 보장된 다른 후보키도 참조할 수 있다. NULL은 참조할 값이 없음을 나타내므로 허용되지만, 외래키에 NOT NULL 제약조건이 별도로 있으면 사용할 수 없다.

옳은 보기는 ㄱ, ㄷ이며, 이 조합은 ②에 해당한다.

NoteBase

문 17. 다음 정규화(normalization)에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 제1정규형에서는 릴레이션에 속한 모든 속성의 도메인이 원자값(atomic value)으로만 구성된다.
ㄴ. 제2정규형에서는 이행적 함수 종속이 존재하지 않는다.
ㄷ. 보이스-코드 정규형(BCNF)은 함수 종속의 모든 결정자가 후보키이다.
ㄹ. 제2정규형 이상에서는 다치 종속(multi valued dependency)을 고려하지 않아도 된다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄹ
④ ㄱ, ㄷ, ㄹ

정답 2번

간단한 해설

제2정규형이 제거하는 것은 **부분 함수 종속**이며, 비주요 속성의 이행적 함수 종속은 남을 수 있다. 또한 제2정규형이나 BCNF를 만족해도 다치 종속에 따른 중복이 남을 수 있으므로, 이를 다루는 **제4정규형**을 따로 고려한다.

NoteBase

보기별 해설

ㄱ. O: 제1정규형은 모든 속성의 도메인이 **원자값으로만 구성**되는 조건이다. 원자값은 해당 데이터베이스에서 더 나누어 다루지 않는 하나의 값이므로, 한 속성 값에 여러 값을 집합이나 반복 항목으로 넣지 않는다.

ㄴ. X: 제2정규형은 제1정규형을 만족하면서 후보키에 속하지 않는 속성인 비주요 속성이 후보키 전체에 완전 함수 종속되는 상태이다. 즉 후보키의 일부에만 의존하는 **부분 함수 종속을 제거**하며, 이행적 함수 종속까지 제거하는 것은 아니다. 예를 들어 학번이 유일한 후보키이고 학번 → 학과, 학과 → 학과사무실이라면, 후보키가 단일 속성이므로 부분 함수 종속은 없지만 학과사무실은 학과를 거쳐 학번에 이행적으로 종속된다. 이러한 비주요 속성의 이행적 함수 종속을 제거하는 단계가 제3정규형이다.

ㄷ. O: '모든 결정자가 후보키'는 BCNF를 설명할 때 쓰는 표현이다. 엄밀한 조건은 **비자명한 함수 종속의 결정자가 슈퍼키**여야 한다는 것이다. 비자명한 함수 종속은 결정자에 포함되지 않은 속성을 결정하는 종속이며, 슈퍼키는 튜플을 유일하게 식별하는 속성 집합이다. 후보키는 슈퍼키 중 불필요한 속성을 더 뺄 수 없는 최소의 집합이므로, 후보키 자체나 후보키를 포함하는 속성 집합이 이 조건을 충족한다.

ㄹ. X: 제2정규형 이상이라는 이유만으로 **다치 종속을 무시할 수는 없다**. 다치 종속은 한 결정자 값에 대응하는 여러 값이 나머지 속성 값과 독립적으로 조합되는 관계이다. 제2정규형이나 BCNF를 만족해도 이러한 조합으로 인한 중복이 남을 수 있다. 이를 다루는 제4정규형에서는 비자명한 다치 종속의 결정자가 슈퍼키여야 한다.

옳은 보기는 ㄱ, ㄷ이며, 이 조합은 ②에 해당한다.

NoteBase

문 18. 다음 릴레이션 R에 대한 SQL문의 실행 결과는?

R

A	B
1	5
1	7
2	1
2	3
2	2
3	12
4	5
4	4
5	6
5	5

```
SELECT COUNT(*) AS T
FROM (
        SELECT A
        FROM R
        GROUP BY A
        HAVING SUM(B) > 10
      ) AS S;
```

①

S
3

②

S
4

③

T
3

NoteBase

④

T
4

정답 3번

간단한 해설

HAVING 조건을 통과하는 그룹은 A가 1, 3, 5인 **3개**이다. 바깥쪽 COUNT(*)는 이 그룹들로 이루어진 S의 행 수를 세고, AS T는 결과 열 이름을 **T**로 지정한다.

계산 과정

1. 그룹 구성: GROUP BY A는 릴레이션 R에서 **A값이 같은 행끼리** 묶는다. A값은 1, 2, 3, 4, 5이므로 총 5개 그룹이 만들어진다.

2. 그룹별 합계: SUM(B)는 각 그룹에 속한 B값을 더한다. A = 1이면 5 + 7 = 12, A = 2이면 1 + 3 + 2 = 6, A = 3이면 12, A = 4이면 5 + 4 = 9, A = 5이면 6 + 5 = 11이다.

3. 그룹 선별: HAVING SUM(B) > 10은 **그룹별 B의 합계가 10을 초과하는 그룹만** 남긴다. 합계가 각각 12, 12, 11인 A = 1, A = 3, A = 5가 조건을 만족한다. 안쪽 SELECT A는 통과한 그룹마다 A값 하나를 출력하므로, S는 A값 1, 3, 5를 가진 3개 행으로 이루어진다.

4. 최종 집계: 바깥쪽 COUNT(*)는 **원래 R의 행 수가 아니라 S의 행 수**를 센다. 따라서 집계값은 3이고, AS T에 따라 결과 열 이름은 T이다. AS S는 안쪽 조회 결과에 붙인 테이블 별칭이다. 실행 결과는 T열에 값 3이 들어 있는 한 행이며, ③의 결과표와 일치한다.

NoteBase

문 19. 인덱스(index)에 대한 설명으로 옳지 않은 것은?

① 테이블에서 한 개 이상의 속성을 이용하여 생성한다.

② 빠른 검색과 효율적인 레코드 접근을 가능하게 한다.

③ 희소 인덱스(sparse index)는 그 인덱스가 클러스터링 인덱스인 경우 사용할 수 없다.

④ 데이터에서 수정, 삭제 등의 변경이 발생하면 인덱스를 재구성할 수 있다.

정답 3번

간단한 해설

③의 '클러스터링 인덱스인 경우 사용할 수 없다'는 설명은 반대이다. 희소 인덱스는 일부 검색 키 값만 인덱스에 두는 방식으로, 데이터가 **검색 키 순서로 정렬된 클러스터링 인덱스에서 사용할 수 있다**.

선지별 해설

① O: 인덱스는 테이블의 **하나 이상의 속성을 검색 키**로 삼아 생성한다. 학번 하나로 만들 수도 있고, 학과와 학번을 함께 묶은 복합 인덱스로 만들 수도 있다.

② O: 인덱스는 검색 키 값과 레코드에 접근할 수 있는 위치 정보를 제공한다. 검색 조건에 맞는 인덱스를 이용하면 **테이블 전체를 훑는 작업을 줄여** 원하는 레코드에 효율적으로 접근할 수 있다.

③ X: 희소 인덱스는 모든 검색 키 값에 항목을 두지 않고 일부 값에만 항목을 둔다. 데이터 파일이 검색 키 순서로 정렬되어 있으면, 인덱스로 탐색 시작 위치를 찾은 뒤 그곳부터 순차적으로 확인하여 해당 레코드를 찾을 수 있다. 따라서 **클러스터링 인덱스에서도 희소 인덱스를 사용할 수 있다**. 예를 들어 각 데이터 블록의 첫 검색 키 값만 인덱스에 기록해도, 대상 레코드가 들어 있는 블록을 찾아 그 안을 탐색할 수 있다.

④ O: 데이터가 변경되면 인덱스도 **변경된 데이터와 일치하도록 유지**해야 한다. 검색 키 값이 수정되면 해당 인덱스 항목을 바꾸고, 레코드가 삭제되면 그 레코드를 가리키던 정보를 제거하거나 조정한다. 이 과정에서 필요한 경우 인덱스 구조를 재구성할 수 있다.

NoteBase

문 20. 다음 뷰(view)에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 실체화된 뷰(materialized view)는 뷰 정의 쿼리 결과를 물리적으로 저장해 두고 일정 주기로 갱신할 수 있다.
ㄴ. 뷰는 데이터 보안 강화, 복잡한 쿼리 단순화를 위해 사용할 수 있다.
ㄷ. 뷰에 GROUP BY 절을 포함하는 경우에는 뷰를 갱신할 수 있다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 1번

간단한 해설

ㄷ의 GROUP BY는 여러 원본 행을 하나의 그룹으로 묶으므로, 그 결과를 바꿀 때 어느 원본 행을 수정할지 일반적으로 정할 수 없다. 따라서 GROUP BY를 포함한 뷰는 **일반적인 직접 갱신이 제한된다**.

보기별 해설

ㄱ. O: 실체화된 뷰는 뷰를 정의한 질의의 **실행 결과를 물리적으로 저장**한다. 저장된 결과는 원본 데이터의 변경을 반영하도록 일정 주기로 새로 고칠 수 있다. 질의 정의만 저장하는 일반 뷰와 달리 결과 자체를 보관하므로, 갱신 시점 사이에는 원본의 최신 상태와 차이가 날 수 있다.

ㄴ. O: 뷰를 통해 사용자에게 필요한 행과 열만 보여 주고 접근 권한을 제한하면 **민감한 데이터의 노출을 줄일 수 있다**. 또한 복잡한 조인이나 검색 조건을 뷰 정의에 담아 두면, 사용자는 뷰를 조회하여 같은 질의를 매번 직접 작성하는 부담을 줄일 수 있다.

ㄷ. X: GROUP BY를 포함한 뷰는 여러 원본 행을 그룹으로 묶으므로 **일반적으로 직접 갱신할 수 없다**. 예를 들어 부서별 급여 합계를 보여 주는 뷰에서 합계를 바꾸려 해도, 어느 직원의 급여를 얼마나 바꿀지 그 결과만으로는 정할 수 없다. 따라서 'GROUP BY 절을 포함하는 경우에는 뷰를 갱신할 수 있다'는 설명은 일반적인 뷰 갱신 조건에 맞지 않는다.

옳은 보기는 ㄱ과 ㄴ이며, 이 조합은 ①에 해당한다.

NoteBase