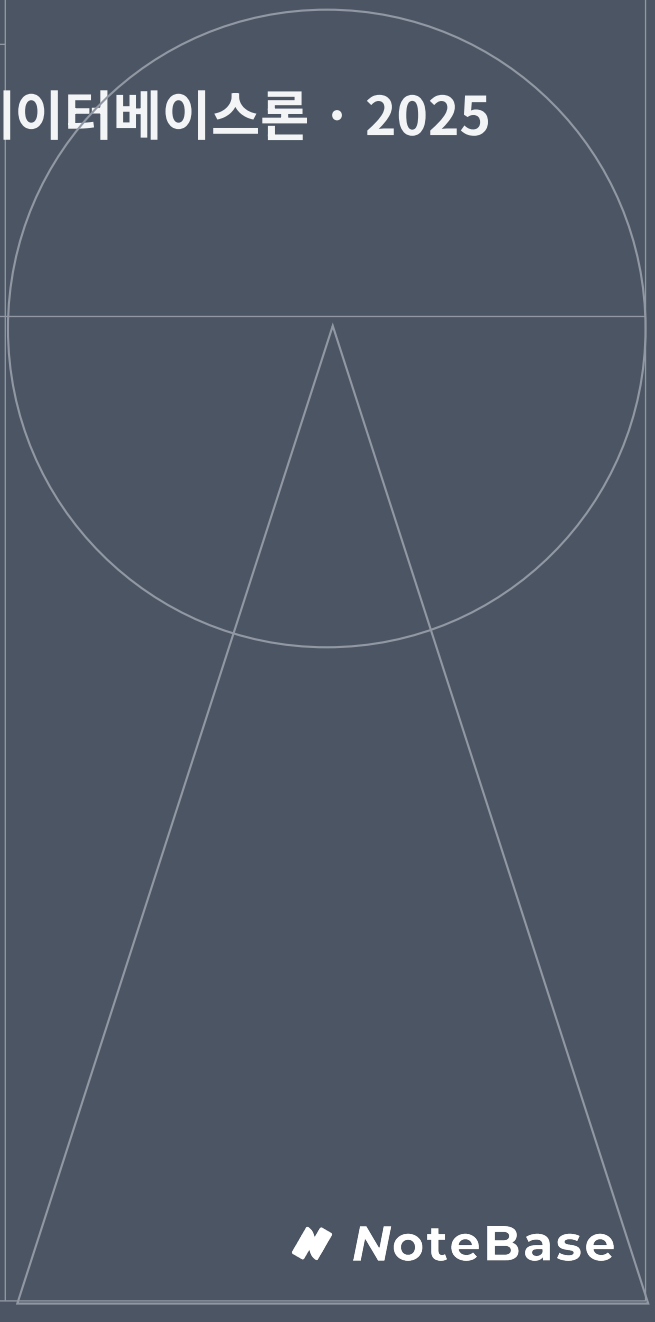
기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2025



NoteBase

기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2025

공무원/경찰/공채(경위) · 데이터베이스론 · 2025

문 1. 데이터베이스 관리 시스템이 갖는 특징으로 가장 적절하지 않은 것은?

① 데이터에 대한 중복 저장을 통제할 수 있다.

② 데이터베이스의 물리적 구조가 변경되면 응용 프로그램도 함께 수정해야 한다.

③ 다수 사용자가 데이터베이스를 동시에 접근할 수 있다.

④ 장애가 발생하더라도 데이터의 일관성과 무결성을 유지한다.

정답 2번

간단한 해설

DBMS는 **물리적 데이터 독립성**을 제공한다. 따라서 논리적 데이터 구조와 이용 방식이 유지되면 저장 위치나 인덱스 같은 물리적 구조가 바뀌어도 응용 프로그램을 함께 수정할 필요가 없다.

선지별 해설

① O: DBMS는 여러 응용 프로그램이 데이터를 통합적으로 공유하도록 관리하여 불필요한 중복 저장을 줄이고 통제할 수 있다. 이는 모든 중복을 반드시 없앤다는 뜻은 아니다. 성능이나 가용성 등을 위해 필요한 중복은 관리하되, 갱신 불일치가 생기지 않도록 통제하는 것이 핵심이다.

② X: DBMS는 **물리적 데이터 독립성**을 제공하여 저장장치, 파일 배치, 인덱스 같은 내부 저장구조의 변경이 응용 프로그램에 직접 영향을 주지 않도록 한다. 논리적 데이터 구조와 이용 방식이 유지된다면 물리적 구조가 바뀌었다는 이유만으로 응용 프로그램을 함께 수정할 필요는 없다.

③ O: DBMS는 여러 사용자나 트랜잭션이 데이터베이스에 동시에 접근할 수 있도록 지원한다. 이때 단순히 동시 실행만 허용하는 것이 아니라 잠금 등 동시성 제어를 통해 서로 충돌하는 읽기·쓰기 작업을 조정하고 일관성을 보호한다.

④ O: DBMS는 로그, 백업, 회복 기능 등을 이용하여 장애 후 데이터베이스를 일관된 상태로 복구한다. 예를 들어 완료되지 않은 트랜잭션의 변경을 취소하거나, 필요한 경우 완료된 변경을 다시 반영하여 중간 결과가 남는 것을 막는다. 장애가 전혀 발생하지 않거나 어떤 손상에서도 무조건 복구된다는 뜻이 아니라, **트랜잭션 관리와 회복 기능으로 일관성과 무결성을 보호**한다는 의미이다.

NoteBase

문 2. 관계 데이터 모델에서 릴레이션이 갖는 특징으로 가장 적절하지 않은 것은?

① 하나의 릴레이션에는 동일한 투플이 존재할 수 없다.

② 투플의 내용이 같더라도 투플의 순서가 다르면 서로 다른 릴레이션이다.

③ 하나의 릴레이션에서 서로 다른 속성의 도메인(domain)은 같을 수 있다.

④ 속성의 순서와 상관없이 동일한 속성들로 구성되면 동일한 릴레이션이다.

정답 2번

간단한 해설

릴레이션은 투플의 집합이므로 **투플의 순서에 의미가 없다**. 동일한 투플들을 어떤 순서로 나열하더라도 같은 릴레이션이다.

선지별 해설

① O. 관계 데이터 모델의 릴레이션은 **투플의 집합**이다. 집합은 같은 원소를 중복하여 갖지 않으므로 모든 속성값이 같은 동일 투플이 중복 존재하지 않는다. 실제 SQL 테이블에서 중복 행을 허용하는 경우와 이론적 릴레이션의 성질을 구별해야 한다.

② X. 집합의 원소에는 순서가 없으므로, 같은 투플들을 다른 순서로 나열해도 **같은 릴레이션**이다. 화면에 행을 표시하거나 물리적으로 저장하는 순서가 달라졌다고 논리적인 릴레이션이 달라지는 것은 아니다.

③ O. 도메인은 한 속성이 가질 수 있는 값들의 범위이다. 서로 다른 속성이 같은 도메인을 사용해도 된다. 예를 들어 '출발도시'와 '도착도시'는 이름과 역할이 다른 속성이지만 동일한 도시명 도메인을 사용할 수 있다.

④ O. 속성은 이름으로 구분하므로 **속성의 나열 순서 자체는 의미가 없다**. 각 속성에 대응하는 값과 투플 집합을 그대로 유지하면서 열의 순서만 바꾸면 같은 릴레이션이다. 이 설명은 속성 이름만 같으면 데이터가 달라도 같은 릴레이션이라는 뜻은 아니다.

NoteBase

문 3. 관계 데이터 모델의 참조 무결성 제약 조건에 대한 설명으로 가장 적절한 것은?

① 참조되는 릴레이션에 새로운 투플을 삽입할 때 참조 무결성 제약조건이 위배되지 않는다.

② 참조하는 릴레이션에 새로운 투플을 삽입할 때 참조 무결성 제약조건이 위배되지 않는다.

③ 참조하는 릴레이션에서 투플을 삭제할 때 참조 무결성 제약조건이 위배된다.

④ 참조되는 릴레이션에서 투플을 삭제할 때 참조 무결성 제약조건이 위배되지 않는다.

정답 1번

간단한 해설

정답은 ①이다. **부모에 투플을 추가**하는 것은 기존 자식의 참조 대상을 없애지 않는다. 반대로 자식의 삽입과 부모의 삭제는 참조 무결성 위반 가능성을 확인해야 한다.

선지별 해설

① O: 참조되는 릴레이션을 부모, 외래키를 가진 참조하는 릴레이션을 자식이라고 하자. 부모에 새로운 투플을 삽입하는 것은 참조 가능한 값을 추가하는 것이므로 **이 부모·자식 참조 관계의 무결성을 깨뜨리지 않는다**. 따라서 정답은 ①이다. 다만 기본키 중복이나 다른 외래키 등 별도 제약에 위배되는지는 따로 판단한다.

② X: 자식에 투플을 삽입할 때는 그 외래키 값에 대응하는 부모의 키가 존재하는지 확인해야 한다. 허용되는 NULL의 경우 등을 제외하고, 존재하지 않는 부모 키를 참조하면 참조 무결성을 위반한다. 따라서 항상 위배되지 않는다고 할 수 없다.

③ X: 자식 투플을 삭제하면 참조 자체가 제거되므로 해당 참조 관계에서 존재하지 않는 부모를 가리키는 문제가 생기지 않는다. 따라서 자식의 삭제가 참조 무결성을 위반한다고 설명한 것이 잘못이다.

④ X: 부모 투플을 삭제할 때 이를 참조하는 자식이 남으면 외래키의 참조 대상이 사라지므로 무결성 위반이 발생할 수 있다. 실제 DBMS는 삭제를 제한하거나, 정해진 동작에 따라 자식을 함께 삭제하거나 외래키를 NULL로 바꾸는 등의 방법으로 무결성을 유지한다. 참조하는 자식이 없다면 해당 관계에서는 삭제가 가능하다.

NoteBase

문 4. 3단계 데이터베이스 구조에 대한 설명으로 적절한 것만을 모두 고른 것은?

가. 사용자 관점의 내부 스키마(internal schema), 조직 전체 관점의 개념 스키마(conceptual schema), 저장구조 관점의 외부 스키마(external schema)로 구성된다.
나. 내부 스키마는 데이터베이스에 데이터가 물리적으로 저장되는 방식을 표현한다.
다. 외부 스키마는 다수 사용자들이 각자 데이터베이스를 바라보는 뷰를 의미한다.
라. 데이터베이스마다 다수의 개념 스키마가 존재할 수 있다.

① 가, 다
② 나, 다
③ 나, 라
④ 다, 라

정답 2번

간단한 해설

외부 스키마는 **사용자별 뷰**, 개념 스키마는 전체 데이터베이스의 통합된 논리구조, 내부 스키마는 물리적 저장구조를 나타낸다. 개념 스키마는 하나이며 사용자별로 여러 개가 될 수 있는 것은 외부 스키마이다.

보기별 해설

가. X: 사용자 관점은 **외부 스키마**, 저장구조 관점은 **내부 스키마**에 해당한다. 조직 전체의 데이터와 관계·제약조건을 통합한 논리적 구조를 나타내는 것이 개념 스키마이다. 보기에서는 외부와 내부의 관점을 서로 바꾸었다.

나. O: 내부 스키마는 데이터의 물리적 저장구조를 정의한다. 저장 레코드의 형식, 파일 구성, 인덱스와 접근 경로 등 실제 저장·접근 방식이 여기에 해당한다.

다. O: 외부 스키마는 특정 사용자나 사용자 집단이 필요로 하는 데이터베이스의 일부를 나타내는 뷰이다. 사용자마다 관심 데이터와 표현 방식이 다를 수 있으므로 하나의 데이터베이스에 여러 외부 스키마가 존재할 수 있다.

라. X: 3단계 구조에서 하나의 데이터베이스 전체를 통합하여 기술하는 개념 스키마는 **하나**이다. 여러 사용자 관점에 따라 다수가 존재할 수 있는 것은 외부 스키마이다.

적절한 보기는 나, 다이므로 정답은 ②이다.

NoteBase

문 5. 릴레이션 R과 S에 대해 다음 관계 대수를 수행한 결과의 투플 개수는? (단, ⋈ $_N$은 자연조인, ⟕은 왼쪽 외부조인이다)

R

A	B
a1	b1
a2	b2
a4	b2
a5	NULL

S

B	C
b1	c1
b2	c2
b3	c2
b4	c4

<관계 대수>

가. R ⋈ $_N$ S

나. R ⟕ S

① 가: 2, 나: 4

② 가: 3, 나: 3

③ 가: 3, 나: 4

④ 가: 4, 나: 3

정답 3번

간단한 해설

두 릴레이션의 공통 속성은 B이다. 자연 조인은 B가 일치하는 **세 투플**만 만들고, 왼쪽 외부조인은 연결되지 않는 R의 **(a5, NULL)도 보존**하므로 한 투플이 더 생긴다.

NoteBase

계산 과정

1. 조인 기준 확인
R의 속성은 (A, B), S의 속성은 (B, C)이므로 공통 속성 B를 기준으로 연결한다. R에는 (a1, b1), (a2, b2), (a4, b2), (a5, NULL)이 있고, S에는 (b1, c1), (b2, c2), (b3, c2), (b4, c4)가 있다.

2. 가: 자연 조인
B가 b1인 투플끼리 연결하면 (a1, b1, c1)이 생성된다. R에서 B가 b2인 투플은 두 개이므로 S의 (b2, c2)와 각각 연결되어 (a2, b2, c2), (a4, b2, c2)가 생성된다. A 값이 서로 다르므로 이 둘은 중복 투플이 아니다. R의 B가 NULL인 투플은 일치하는 연결 상대가 없고, S의 b3·b4도 R에 상대가 없어 결과에 포함되지 않는다. 따라서 가의 투플 수는 $1 + 2 = 3$이다.

3. 나: 왼쪽 외부조인
왼쪽 외부조인은 자연 조인 결과에 더하여, 연결 상대가 없는 왼쪽 릴레이션 R의 투플도 보존한다. 따라서 (a5, NULL)에 S에서 채울 수 없는 C 값을 NULL로 보충한 (a5, NULL, NULL)을 추가한다. S에만 있는 b3·b4는 오른쪽의 미일치 투플이므로 추가하지 않는다. 나의 투플 수는 $3 + 1 = 4$이다.

따라서 **가: 3, 나: 4**이다.

NoteBase

문 6. 다른 관계대수 연산자를 이용하여 동일한 기능을 표현할 수 있는 관계대수 연산자로 가장 적절한 것은?

① 합집합(union)

② 교집합(intersect)

③ 셀렉트(select)

④ 프로젝트(project)

정답 2번

간단한 해설

교집합은 **차집합을 두 번 적용**하여 표현할 수 있다. R에서 S에 없는 튜플을 다시 제거하면 두 릴레이션에 공통인 튜플만 남으므로 $R \cap S = R - (R - S)$이다.

선지별 해설

관계대수에서는 셀렉트, 프로젝트, 합집합, 차집합, 카티션 곱 등을 기본 연산으로 두고, 이들을 조합해 표현할 수 있는 연산을 유도 연산으로 구분한다. 이 문항은 이러한 통상적인 기본·유도 연산의 구분을 묻는다.

① X: 합집합은 합병 가능한 두 릴레이션 중 어느 한쪽에 속하는 모든 튜플을 모으는 연산이다. 집합 의미의 관계대수에서는 중복 튜플을 제거한다. 합집합은 통상적인 기본 연산이므로 이 문항이 묻는 유도 연산에 해당하지 않는다.

② O: 교집합은 두 릴레이션에 공통으로 속한 튜플만 남긴다. 합병 가능한 릴레이션 R과 S에 대해 **차집합을 두 번 적용**하면 같은 결과를 얻는다. 즉 $R \cap S = R - (R - S)$이다. R에서 S에 없는 튜플을 먼저 구한 뒤, 이를 다시 R에서 제거하면 양쪽에 공통인 튜플만 남는다.

③ X: 셀렉트는 주어진 조건을 만족하는 튜플을 골라내는 연산으로, 릴레이션의 행을 선택한다. 예를 들어 나이가 20 이상인 튜플만 남기는 연산이다. 통상적인 관계대수 체계에서 조건에 따른 튜플 선택을 담당하는 기본 연산이며, 교집합과 같은 유도 연산으로 분류하지 않는다.

④ X: 프로젝트는 지정한 속성만 남기는 연산으로, 릴레이션의 열을 선택한다. 집합 의미의 관계대수에서는 선택된 속성값이 같은 결과 튜플을 하나로 합친다. 이는 셀렉트의 행 선택과 다르며, 속성을 추출하는 기본 연산이다.

NoteBase

문 7. 릴레이션 R의 카디널리티(cardinality)가 500이고 릴레이션 S의 카디널리티가 50일 때, 다음 SQL을 실행한 결과 투플(tuple)의 최대 개수는?

```
SELECT * FROM R, S;
```

① 50
② 500
③ 550
④ 25,000

정답 4번

간단한 해설

조건 없이 R과 S를 함께 나열한 FROM 절은 **카티션 곱**을 만든다. 각 릴레이션의 투플 수를 곱하면 결과는 **25,000개**이다.

계산 과정

카디널리티는 릴레이션에 들어 있는 **투플의 개수**이다. 따라서 R에는 500개, S에는 50개의 투플이 있다.

SELECT * FROM R, S;에는 두 릴레이션을 연결하는 조건이나 행을 거르는 WHERE 절이 없다. 그러므로 R의 각 투플을 S의 모든 투플과 짝짓는 **카티션 곱**을 구한다. 별도의 조건이 없으므로 같은 이름의 속성이 있어도 자동으로 자연 조인을 하지 않는다.

R의 투플 하나마다 S의 투플 50개가 결합하므로 결과 투플 수는 $500 \times 50 = 25,000$개이다. 따라서 최대 개수는 **25,000개**로 ④와 일치한다. 이 질의는 필터나 중복 제거를 지정하지 않았으므로 주어진 행 수에서 바로 그 개수의 조합을 만든다.

NoteBase

문 8. 개체-관계 모델(entity-relationship model)에서 관계에 대한 설명으로 가장 적절하지 않은 것은?

① 전체 참여(total participation)는 모든 개체 인스턴스가 관계에 반드시 참여해야 되는 것을 의미한다.

② 매핑 카디널리티(mapping cardinality)는 관계를 맺는 두 개체 집합에서 각 개체 인스턴스가 연관성을 맺고 있는 상대 개체 집합의 인스턴스 개수를 나타낸다.

③ 관계의 종속성이 있는 강한 개체와 약한 개체는 다대다 관계이다.

④ 순환 관계는 하나의 개체 타입이 자기 자신과 맺는 관계이다.

정답 3번

간단한 해설

정답은 ③이다. 소유자인 강한 개체와 약한 개체의 식별 관계는 전형적으로 **1:N 관계**이다. 약한 개체는 강한 개체에 종속되며, 다대다 관계로 설명하지 않는다.

선지별 해설

① O: 전체 참여는 해당 개체 집합의 모든 인스턴스가 그 관계에 적어도 한 번 참여해야 한다는 제약이다. 예를 들어 모든 부양가족이 반드시 어떤 직원에 소속되어야 한다면 부양가족은 그 소속 관계에 전체 참여한다.

② O: 매핑 카디널리티는 한 개체 인스턴스가 상대 개체 집합의 몇 개 인스턴스와 연결될 수 있는지를 나타내며, 이항 관계를 1:1, 1:N, N:1, M:N 등으로 구분한다. 참여 여부를 나타내는 전체·부분 참여와는 구별되는 제약이다.

③ X: 전형적인 강한 개체와 약한 개체의 **식별 관계는 강한 개체에서 약한 개체로 1:N**이다. 약한 개체 하나는 자신을 소유하는 강한 개체 하나에 종속되고, 강한 개체 하나에는 여러 약한 개체가 연결될 수 있다. 약한 개체는 소유자의 키와 자신의 부분키를 함께 이용해 식별하므로, 이를 다대다 관계라고 한 ③이 정답이다.

④ O: 순환 관계는 같은 개체 타입이 서로 다른 역할로 참여하는 관계이다. 예를 들어 직원 타입 안에서 한 직원이 다른 직원의 상사가 되는 관계가 이에 해당한다. '자기 자신'은 같은 개체 타입을 가리키며, 반드시 한 인스턴스가 자신과만 연결된다는 뜻은 아니다.

NoteBase

문 9. 다음과 같이 릴레이션 R을 릴레이션 R1, R2로 분해할 때, 분해된 릴레이션 R1, R2가 만족하는 가장 높은 수준의 정규형은?

R

A	B	C
100	a	s
100	a	t
100	b	s
100	b	t
200	c	x
200	c	y
200	c	z



R1

A	B
100	a
100	b
200	c

R2

A	C
100	s
100	t
200	x
200	y
200	z

① 제2정규형

② 제3정규형

③ 보이스/코드(Boyce/Codd) 정규형

④ 제4정규형

정답 4번

간단한 해설

같은 A에 속하는 B 값들과 C 값들이 독립적으로 조합되어 생기는 중복을 **(A, B)와 (A, C)로 분리**하였다. 이는 함수 종속만이 아니라 **비자명한 다치 종속에 따른 중복을 해소하는 제4정규형**의 분해이다.

NoteBase

선지별 해설

그림에서 A가 100이면 B의 a·b와 C의 s·t가 모두 조합되어 네 투플이 생긴다. A가 200이면 B의 c와 C의 x·y·z가 조합되어 세 투플이 생긴다. 이처럼 A에 대해 B와 C의 값 집합이 독립적인 다치 종속을 분리한 결과가 R1(A, B)와 R2(A, C)이다. 두 릴레이션을 A로 자연 조인하면 원래 일곱 투플이 복원된다.

① 제2정규형: 후보키의 일부에 비주요 속성이 종속되는 부분 함수 종속을 제거한 단계이다. 그림의 핵심은 부분 함수 종속이 아니라 서로 독립적인 다중 값 집합의 조합이므로, 이 단계만으로 분해의 특징과 가장 높은 수준을 설명할 수 없다.

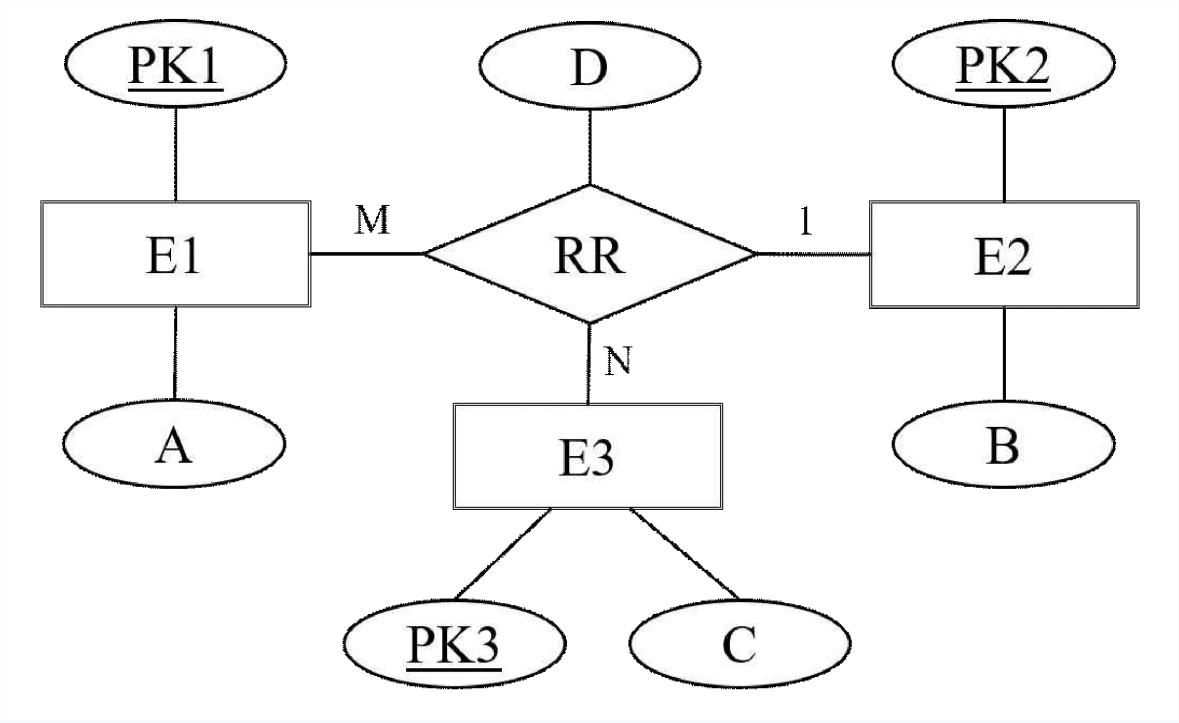
② 제3정규형: 비자명한 함수 종속에서 결정자가 슈퍼키이거나 종속되는 속성이 주요 속성이어야 한다. 일반적으로 비주요 속성의 이행적 함수 종속을 제거하는 단계로 설명하지만, **함수 종속과 다른 다치 종속에 따른 중복**까지 모두 제거하는 기준은 아니다.

③ 보이스/코드 정규형: 모든 비자명한 함수 종속의 결정자가 슈퍼키여야 한다. 제3정규형보다 강한 조건이지만, 슈퍼키가 아닌 속성에서 발생하는 비자명한 다치 종속은 별도로 남을 수 있어 그림과 같은 분해의 최종 기준으로는 부족하다.

④ 제4정규형: 모든 비자명한 다치 종속의 결정자가 슈퍼키여야 한다. 그림이 나타내는 $A \twoheadrightarrow B$, $A \twoheadrightarrow C$를 R1(A, B), R2(A, C)로 나누면 각 릴레이션에서는 해당 다치 종속이 전체 속성을 포괄하는 자명한 다치 종속이 된다. 따라서 제시된 보기 중 **분해된 두 릴레이션이 만족하는 가장 높은 단계는 제4정규형**이다.

NoteBase

문 10. 다음 E-R 다이어그램에서 관계 RR을 관계 데이터 모델로 변환할 때 가장 적절한 것은? (단, 밑줄 친 속성은 기본키이다)



① RR(PK1, PK3, PK2, D)

② RR1(PK1, PK3), RR2(PK2, D)

③ RR1(PK1, PK2, PK3), RR2(PK2, D)

④ RR1(PK2, PK3), RR2(PK1, PK3, D)

정답 1번

간단한 해설

삼항관계에서 E2 쪽의 1은 E1·E3의 한 쌍에 대응하는 E2가 최대 하나임을 뜻한다. 따라서 RR은 세 개체의 키와 관계 속성 D를 포함하되 **(PK1, PK3)를 기본키**로 한다.

NoteBase

도식 해설

그림의 RR은 E1·E2·E3를 연결하는 삼항관계이다. E1 쪽에 M, E2 쪽에 1, E3 쪽에 N이 표시되어 있고, D는 개체가 아니라 RR에 연결된 관계 속성이다.

E2 쪽의 1은 **E1과 E3의 한 쌍이 정해지면 연결되는 E2가 최대 하나**라는 뜻이다. 따라서 PK1과 PK3의 조합으로 PK2를 결정할 수 있다. 반면 E1과 E3 쪽은 다수 참여가 가능하므로 PK1이나 PK3 하나만으로 관계의 한 튜플을 식별할 수는 없다.

RR을 별도 릴레이션으로 변환하면 참여 개체의 키인 PK1·PK2·PK3와 관계 속성 D를 포함한다. 이때 **기본키는 (PK1, PK3)**이며 PK2는 기본키에서 제외된다. 세 개체의 키는 각각 해당 개체 릴레이션을 참조하는 외래키 역할도 한다. A·B·C는 각 개체의 속성이므로 RR에 옮기지 않는다.

②·③처럼 D를 PK2와만 묶으면 D가 E2 하나에 종속되는 것으로 바뀌어 원래 삼항관계의 속성이라는 의미를 보존하지 못한다. ③의 세 키 전체는 튜플을 식별할 수 있어도 PK2가 불필요하므로 최소성이 없다. ④ 역시 삼항관계를 두 이항 릴레이션으로 분해하여 원래의 세 개체 대응을 일반적으로 보존하지 못한다.

따라서 RR(PK1, PK3, PK2, D)에서 PK1·PK3에만 밑줄을 표시한 ①이 적절하다.

NoteBase

문 11. 다음 관계대수(relational algebra) 연산 중 합병 가능(union compatability) 조건을 필요로 하는 연산은 모두 몇 개인가?

가. 카티션 곱(cartesian product)
나. 교집합(intersect)
다. 완전 외부 조인(full outer join)
라. 차집합(set difference)
마. 프로젝트(project)

① 2
② 3
③ 4
④ 5

정답 1번

간단한 해설

교집합과 차집합은 두 릴레이션의 전체 투플을 비교하므로 **같은 속성 수와 호환되는 대응 도메인**이 필요하다.
카티션 곱·외부 조인·프로젝트에는 이 합병 가능 조건을 요구하지 않는다.

NoteBase

보기별 해설

합병 가능 조건은 두 릴레이션의 **속성 개수가 같고 대응하는 속성의 도메인이 호환**되어야 한다는 조건이다. 두 집합의 투플을 같은 형식으로 비교하거나 합치는 연산에서 필요하다. 아래 O/X는 이 조건을 요구하는지를 나타낸다.

가 X. 카티션 곱은 한 릴레이션의 각 투플에 다른 릴레이션의 각 투플을 이어 붙인다. 두 릴레이션의 속성 수나 대응 도메인이 같을 필요가 없다. 같은 이름의 속성은 필요에 따라 구별하여 표현한다.

나 O. 교집합은 두 릴레이션에 공통으로 존재하는 투플을 찾는다. 전체 투플의 동일 여부를 비교해야 하므로 **합병 가능 조건**이 필요하다.

다 X. 완전 외부 조인은 조인 조건에 맞는 투플을 결합하고, 일치 상대가 없는 양쪽 투플도 보존하며 없는 쪽의 속성값을 NULL로 채운다. 조인에 사용되는 속성은 비교 가능해야 하지만, 두 릴레이션 전체의 속성 수와 도메인이 같아야 하는 것은 아니다.

라 O. 차집합은 첫 번째 릴레이션에 있으면서 두 번째에는 없는 투플을 구한다. 같은 형식의 투플끼리 존재 여부를 비교하므로 **합병 가능 조건**이 필요하다.

마 X. 프로젝트는 하나의 릴레이션에서 지정한 속성을 선택하는 단항 연산이다. 두 릴레이션 사이의 합병 가능 조건은 적용되지 않는다.

조건이 필요한 연산은 나·라의 **2개**이므로 ①과 일치한다.

NoteBase

문 12. CAR 테이블에 대해 다음 SQL을 수행한 결과로 적절한 것은?

<CAR 테이블>

CAR_ID	SEATS
1	10
2	20
3	NULL

<SQL>
Q1. SELECT SEATS + 1 FROM CAR
WHERE CAR_ID = 3;
Q2. SELECT AVG(SEATS) FROM CAR;
Q3. SELECT COUNT(*) FROM CAR;

① Q1: NULL, Q2: 15, Q3: 3

② Q1: NULL, Q2: NULL, Q3: 2

③ Q1: 1, Q2: 10, Q3: 3

④ Q1: 1, Q2: 10, Q3: 2

정답 1번

간단한 해설

정답은 ①이다. NULL에 1을 더해도 NULL이고, AVG(SEATS)는 NULL을 제외해 15가 된다. COUNT(*)는 NULL 포함 여부와 관계없이 모든 행을 세므로 3이다.

NoteBase

계산 과정

CAR 테이블에는 세 행이 있으며, CAR_ID가 1·2·3인 행의 SEATS 값은 각각 10·20·NULL이다. NULL은 숫자 0이 아니라 값을 알 수 없거나 값이 없음을 나타내므로 연산과 집계에서 처리 방식을 구분해야 한다.

Q1은 CAR_ID = 3인 한 행을 선택하여 SEATS + 1을 계산한다. 해당 SEATS가 NULL이므로 **NULL + 1의 결과도 NULL**이다. NULL을 0으로 바꾸는 별도 처리가 없기 때문에 결과가 1이 되지 않는다.

Q2의 AVG(SEATS)는 NULL인 값을 제외하고 평균을 낸다. 따라서 유효한 값 10과 20 두 개를 사용하여 (10 + 20)/2 = 15를 얻는다. NULL을 포함한 세 행으로 나누어 10을 계산하거나, NULL이 하나 있다는 이유로 전체 평균을 NULL로 만들지 않는다.

Q3의 **COUNT(*)는 행 자체를 세므로 3**이다. SEATS가 NULL인 행도 존재하는 행이므로 포함한다. 반면 COUNT(SEATS)라면 SEATS가 NULL이 아닌 행만 세어 2가 된다.

세 결과는 **Q1: NULL, Q2: 15, Q3: 3**이므로 정답은 ①이다. ②는 평균과 행 수를 잘못 처리했고, ③은 NULL을 0으로 취급한 산술·평균 결과가 잘못되었다. ④는 여기에 COUNT(*)까지 COUNT(SEATS)처럼 처리한 오류가 있다.

NoteBase

문 13. 동시성 제어 없이 두 트랜잭션이 다음과 같이 실행될 때 발생하는 문제점으로 가장 적절한 것은?

시간 \ TID	T1	T2

1	read(X);
	X ← X + 1000;
	write(X);	

2		read(X);
		X ← X * 2;
		write(X);

3	read(Y); rollback;	

① 연쇄 복귀(cascading rollback)

② 모순성(inconsistency)

③ 갱신 분실(lost update)

④ 반복 불가능 읽기(unrepeatable read)

정답 1번

간단한 해설

T2가 **T1이 확정하지 않은 X의 변경값을 읽고** 실행된 뒤 T1이 취소되었다. T2의 처리도 취소해야 하므로, 한 트랜잭션의 취소가 다른 트랜잭션으로 이어지는 **연쇄 복귀**가 문제이다.

NoteBase

도식 해설

시간 1에서 T1은 X를 읽어 1000을 더하고 저장한다. 아직 커밋하지 않았으므로 이 변경값은 확정된 값이 아니다.

시간 2에서 T2는 그 X를 읽고 두 배로 만들어 저장한다. 초기값을 X_0라고 하면 T2의 결과는 $2(X_0 + 1000)$이다. 즉, T2는 T1의 미확정 변경에 의존하고 있다.

시간 3에서 T1은 Y를 읽은 뒤 롤백한다. T1이 만든 변경을 취소해야 하므로 그 변경값을 이용한 T2의 결과도 그대로 유지할 수 없다. T2 역시 롤백해야 하는 이 현상이 **연쇄 복귀(cascading rollback)**이다.

모순성은 동시 실행으로 일관되지 않은 데이터를 참조하는 문제를 가리키는 넓은 설명인 반면, 이 도식의 직접적인 핵심은 미확정값에 대한 의존으로 취소가 전파된다는 점이다.

갱신 분실은 다른 트랜잭션의 갱신을 덮어써 없애는 문제이지만, 여기서 T2는 T1의 변경값을 읽어 계산하였다.

반복 불가능 읽기는 같은 트랜잭션이 같은 데이터를 다시 읽을 때 값이 달라지는 문제이며, 도식에는 동일 데이터를 반복해서 읽는 과정이 없다.

NoteBase

문 14. 다음 조건을 만족하도록 EMPLOYEE 테이블을 생성하는 SQL에서 빈칸 ㉠~㉢에 적절한 것은?

<조건>
가. EMPNO는 기본키이다.
나. SALARY는 700 미만의 값만 가질 수 있다.
다. DNO는 NULL을 허용하지 않는다.
라. DNO는 DEPARTMENT 테이블의 DEPTNO를 참조하기 위한 외래키이다.

<SQL>

```
CREATE TABLE EMPLOYEE (
EMPNO INT,
SALARY INT ( ㉠ ) (SALARY < 700),
DNO INT ( ㉡ ),
PRIMARY KEY EMPNO,
FOREIGN KEY(DNO) ( ㉢ ) DEPARTMENT(DEPTNO) );
```

① ㉠: BETWEEN, ㉡: DEFAULT, ㉢: REFERENCES
② ㉠: BETWEEN, ㉡: NOT NULL, ㉢: CASCADE
③ ㉠: CHECK, ㉡: UNIQUE, ㉢: CASCADE
④ ㉠: CHECK, ㉡: NOT NULL, ㉢: REFERENCES

정답 4번

간단한 해설

값의 조건은 **CHECK**, NULL 금지는 **NOT NULL**, 외래키의 참조 대상 지정은 **REFERENCES**로 표현한다. 세 제약은 각각 급여의 범위, 부서번호의 필수 입력, 참조 무결성을 담당한다.

NoteBase

빈칸별 해설

기본키 조건: EMPNO를 기본키로 지정하면 각 직원을 유일하게 식별할 수 있어야 하므로 중복값과 NULL을 허용하지 않는다. 제시문에서 EMPNO의 기본키 지정은 이미 주어져 있고, 빈칸은 급여의 범위와 부서번호의 필수 입력·참조 관계를 완성하는 부분이다.

㉠ CHECK: 'SALARY < 700'이라는 조건을 제약조건으로 선언하려면 CHECK를 사용한다. 따라서 700 이상의 급여 값은 허용되지 않는다. BETWEEN은 범위 비교에 쓰는 술어이지 이 위치에서 조건식 전체를 제약조건으로 선언하는 키워드가 아니다. 또한 **CHECK만으로 NULL까지 금지되는 것은 아니므로** 값의 범위 제한과 NULL 금지를 구별해야 한다.

㉡ NOT NULL: DNO에 NULL을 허용하지 않으려면 NOT NULL을 지정한다. DEFAULT는 값을 생략했을 때 사용할 기본값을 정하는 기능이므로 명시적인 NULL 입력을 막는 제약과 다르다. UNIQUE는 중복값을 제한하는 제약이며 NOT NULL을 대신하지 않는다.

㉢ REFERENCES: 외래키인 DNO가 참조하는 대상은 REFERENCES 뒤에 지정한다. 'REFERENCES DEPARTMENT(DEPTNO)'는 DNO가 DEPARTMENT의 DEPTNO를 참조함을 뜻한다. DNO에 NOT NULL도 지정되어 있으므로 입력한 부서번호는 참조 대상에 존재해야 한다. CASCADE는 참조 대상의 삭제·갱신을 연쇄 반영하는 동작을 지정할 때 사용하며, 참조 대상 선언 키워드를 대신할 수 없다.

따라서 ㉠ CHECK, ㉡ NOT NULL, ㉢ REFERENCES의 조합인 ④와 일치한다.

NoteBase

문 15. DID가 'B041'인 도서를 구매하지 않은 고객의 고객번호(CID)와 고객이름(CNAME)을 검색하기 위한 다음 SQL에서 빈칸 ㉠, ㉡에 적절한 것은? (단, 스키마에서 밑줄 친 속성은 기본키이고 ORDER에서 CCID는 CUSTOMER의 CID를 참조하기 위한 외래키이다)

<스키마>

CUSTOMER(CID, CNAME, PHONE)

ORDER(ORDID, DID, CCID, ODATE)

<SQL>

```
SELECT CID, CNAME
FROM CUSTOMER
WHERE CID ( ㉠ )
  (SELECT CCID FROM ORDER WHERE ( ㉡ ) );
```

① ㉠: IN, ㉡: CCID = CID

② ㉠: IN, ㉡: DID = 'B041'

③ ㉠: NOT IN, ㉡: DID = 'B041'

④ ㉠: NOT IN, ㉡: CCID = CID

정답 3번

간단한 해설

하위질의에서 DID = 'B041'인 주문의 고객번호를 구한 뒤, **NOT IN으로 그 목록에 없는 고객**을 찾는다. 다만 이 방식이 의도대로 작동하려면 **하위질의 결과에 NULL이 없어야 한다**.

NoteBase

빈칸별 해설

㉠ NOT IN: 하위질의가 반환한 구매 고객번호 목록에서 바깥 질의의 CID를 제외하는 연산이다. 하위질의 결과에 NULL이 없다면, 목록에 없는 고객만 남으므로 해당 도서를 구매하지 않은 고객을 찾을 수 있다. IN을 쓰면 반대로 목록에 있는 구매 고객을 찾는다. 단, **외래키라는 조건만으로 CCID의 NULL이 금지되지는 않는다**. 하위질의 결과에 NULL이 섞이면 목록의 다른 값과 일치하지 않는 CID도 NOT IN 조건이 참이 아니라 UNKNOWN이 되어 조회되지 않는다. 문제에 NULL 배제 조건은 명시되어 있지 않으므로, 이를 주어진 제약으로 단정해서는 안 된다.

㉡ DID = 'B041': ORDER에서 **조회 대상 도서의 주문만 먼저 선택**하는 조건이다. SELECT CCID는 그 주문을 한 고객번호를 반환한다. CCID = CID는 현재 고객과 주문을 연결할 뿐 도서를 B041로 제한하지 않으므로, 이를 넣으면 도서 종류와 관계없이 주문이 있는지를 검사하게 된다.

㉠에 NOT IN, ㉡에 DID = 'B041'을 넣는 조합이 ③이다. 이 조합은 B041 구매 고객을 구해 제외하려는 구성이며, 위의 NULL 처리 한계에 유의해야 한다.

NoteBase

문 16. 비어있는 B-트리에 다음과 같은 순서로 키가 삽입된 후 루트 노드에 존재하는 키 값은? (단, 차수는 3이다)

10, 22, 13, 17, 1, 3, 12, 29, 5

① 10

② 12

③ 13

④ 17

정답 3번

간단한 해설

29를 삽입한 뒤 루트가 넘치면서 **13이 새 루트**로 올라간다. 마지막 5의 삽입은 왼쪽 하위 노드만 분할하므로 최종 루트의 키는 13이다.

NoteBase

계산 과정

차수 3인 B-트리에서 한 노드의 자식은 최대 3개, 키는 최대 2개이다. 키를 정렬하여 삽입한 결과 한 노드에 키가 3개가 되면 **가운데 키를 부모로 올리고 양쪽을 분할**한다. 부모도 넘치면 같은 처리를 반복한다. 아래에서 [ ]는 한 노드의 키를 나타낸다.

10, 22를 차례로 삽입하면 루트는 [10, 22]이다. 13을 넣으면 [10, 13, 22]로 넘치므로 13을 새 루트로 올린다. 이때 루트 [13]의 왼쪽 자식은 [10], 오른쪽 자식은 [22]이다.

17은 오른쪽 자식에 들어가 [17, 22]가 된다. 1은 왼쪽 자식에 들어가 [1, 10]이 된다. 이어서 3을 넣으면 왼쪽 노드가 [1, 3, 10]으로 넘치므로 3을 루트로 올린다. 루트는 [3, 13], 그 자식들은 왼쪽부터 [1], [10], [17, 22]이다.

12는 3과 13 사이의 자식에 들어가 [10, 12]가 된다. 29는 오른쪽 자식에 들어가 [17, 22, 29]를 만들므로 22를 부모로 올린다. 부모인 기존 루트도 [3, 13, 22]로 넘쳐 **가운데 키 13이 새 루트**가 된다. 새 루트의 왼쪽 내부 노드는 [3]이고 자식은 [1], [10, 12]이며, 오른쪽 내부 노드는 [22]이고 자식은 [17], [29]이다.

마지막 5는 루트 13보다 작고 내부 노드의 키 3보다 크므로 잎 [10, 12]에 들어간다. [5, 10, 12]를 분할하면서 10을 부모 [3]으로 올리면 부모는 [3, 10]이 된다. 키가 2개이므로 이 부모는 더 분할되지 않으며 최상위 루트 [13]도 그대로 유지된다.

따라서 최종 루트 노드의 키는 13이고, 정답은 ③이다.

NoteBase

문 17. 다음 뷰(view)에 대한 설명으로 적절한 것만을 모두 고른 것은?

가. 다수의 테이블에 대한 조인을 통해 뷰를 생성할 수 있다.
나. 뷰는 일반 테이블과 같이 항상 삽입과 삭제가 가능하다.
다. 기존 뷰를 사용하여 새로운 뷰를 생성하는 것이 불가능하다.
라. 뷰가 정의된 기본 테이블이 제거되면 뷰도 자동적으로 제거된다.
마. 사용자별로 필요한 데이터를 제공하기 때문에 뷰에 권한 설정이 불가능하다.

① 가, 나
② 가, 라
③ 나, 다
④ 라, 마

정답 2번

간단한 해설

뷰는 **기본 테이블에 의존하는 가상 테이블**이므로, 기본 테이블이 제거되면 기존 정의대로 데이터를 조회할 수 없다. 라는 이러한 종속성을 나타내는 설명으로 인정되지만, 실제 뷰 객체의 자동 삭제 여부는 **DBMS와 삭제 옵션에 따라 다르다**.

NoteBase

보기별 해설

가. O: 뷰는 질의의 결과를 테이블처럼 사용할 수 있도록 정의한 가상 테이블이다. 질의에 **여러 테이블의 조인**을 사용할 수 있으므로, 학생 테이블과 학과 테이블을 연결하여 학생 이름과 학과 이름을 보여 주는 뷰를 만들 수 있다.

나. X: 뷰를 통한 삽입과 삭제에는 **뷰의 정의에 따른 제약**이 있다. 예를 들어 학과별 학생 수를 집계한 뷰에 한 행을 삽입하더라도, 실제로 어떤 학생 행을 기본 테이블에 추가해야 하는지 정할 수 없다. 따라서 삽입과 삭제가 '항상 가능하다'는 표현은 잘못이며, 해당 연산이 허용되는 조건을 만족하는 뷰에서 가능하다.

다. X: **기존 뷰를 바탕으로 새로운 뷰를 정의할 수 있다**. 기존 뷰를 새 뷰의 정의 질의에서 참조하면 된다. 예를 들어 재학생만 보여 주는 뷰에서 특정 학과의 재학생만 골라 보여 주는 뷰를 다시 만들 수 있다.

라. O: 뷰는 정의에 사용한 기본 테이블에 의존하므로, 그 테이블이 제거되면 기존 정의대로 데이터를 조회할 수 없다. 이 문항에서는 기본 테이블과 종속 뷰가 함께 제거된다는 설명을 적절한 것으로 인정한다. 다만 **뷰 객체의 자동 삭제는 모든 DBMS의 공통 동작이 아니다**. PostgreSQL에서는 CASCADE를 지정하면 종속 뷰도 삭제되지만, 기본값인 RESTRICT에서는 종속 뷰가 있으면 기본 테이블의 삭제를 거부한다. Oracle에서는 기본 테이블이 삭제되어도 뷰 객체는 남고 무효 상태가 된다. 따라서 뷰의 종속성과 실제 객체의 삭제 여부를 구분해야 한다.

마. X: 뷰에도 **사용자별 접근 권한을 설정할 수 있다**. 필요한 행이나 열만 보이도록 뷰를 정의하고 해당 뷰에 대한 조회 권한 등을 부여하여 접근 범위를 제한한다. 사용자별로 필요한 데이터를 제공한다는 특성은 권한 설정을 불가능하게 하는 이유가 아니라, 뷰를 보안에 활용하는 이유이다.

이 문항에서 적절한 것으로 인정하는 보기는 가와 라이며, 이 조합은 ②에 해당한다.

NoteBase

문 18. 다음에서 데이터 정의어(data definition language)에 해당하는 것은 모두 몇 개인가?

가. ALTER
나. COMMIT
다. RENAME
라. REVOKE
마. TRUNCATE
바. DROP

① 3
② 4
③ 5
④ 6

정답 2번

간단한 해설

정답은 ②이다. DDL은 **ALTER, RENAME, TRUNCATE, DROP의 4개**이다. COMMIT은 트랜잭션 제어, REVOKE는 권한 제어에 해당한다.

NoteBase

보기별 해설

가. ALTER는 DDL이다. 테이블의 열이나 제약조건 등 이미 정의된 데이터베이스 객체의 구조를 변경한다.

나. COMMIT은 DDL이 아니다. 트랜잭션의 변경 내용을 확정하는 명령으로, 기능별 분류에서는 트랜잭션 제어어(TCL)에 해당한다.

다. RENAME은 DDL이다. 테이블 등 데이터베이스 객체의 이름을 바꾸는 정의 작업이다. 구체적인 구문은 DBMS에 따라 다를 수 있다.

라. REVOKE는 이 문제의 분류에서 DDL이 아니다. 사용자에게 부여했던 접근 권한 등을 회수하는 명령이므로 **데이터 제어어(DCL)**로 구분한다.

마. TRUNCATE는 DDL로 분류한다. 테이블 정의는 남겨 두고 테이블의 데이터를 일괄 제거하는 명령이다. 행을 조건에 따라 삭제하는 DML인 DELETE와 구별해야 한다. 다만 롤백 가능 여부 등 세부 동작은 DBMS에 따라 다르므로 DDL이라는 분류만으로 단정하지 않는다.

바. DROP은 DDL이다. 테이블이나 뷰 등 데이터베이스 객체 자체를 제거한다. 테이블을 대상으로 하면 그 구조와 데이터가 함께 제거되므로, 구조를 남기는 TRUNCATE와 구별된다.

DDL에 해당하는 것은 **가·다·마·바의 4개**로 정답은 ②이다. ①의 3개는 DDL 하나를 빠뜨린 수이며, ③의 5개와 ④의 6개는 트랜잭션·권한 제어 명령까지 포함한 수이다. 이 문항은 SQL 명령을 기능별로 나누는 일반적인 학습 분류를 따른다.

NoteBase

문 19. DNAME이 '총무팀'인 부서에서 근무하는 직원아이디(EID), 직원이름(ENAME), 부서이름(DNAME)을 검색하기 위한 다음 SQL에서 빈칸 ㉠, ㉡에 적절한 것은? (단, 스키마에서 밑줄 친 속성은 기본키이고 EMPLOYEE의 DNO는 DEPARTMENT의 DID를 참조하기 위한 외래키이다)

<스키마>
EMPLOYEE(EID, ENAME, MANAGER, DNO)
DEPARTMENT(DID, DNAME, ADDRESS)
<SQL>
SELECT E.EID AS EMPID, E.ENAME AS EMPNAME,
D.DNAME AS DEPTNAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE (㉠) = '총무팀' AND (㉡);

① ㉠: D.DNAME, ㉡: E.EID = D.DID
② ㉠: D.DID, ㉡: E.DNO = D.DNAME
③ ㉠: D.DNAME, ㉡: E.DNO = D.DID
④ ㉠: D.DID, ㉡: E.DNO = D.DID

정답 3번

간단한 해설

'총무팀'은 부서번호가 아니라 **부서이름 D.DNAME**과 비교해야 한다. 직원과 소속 부서는 **외래키 E.DNO와 참조 대상 기본키 D.DID**를 연결한다.

NoteBase

빈칸별 해설

E는 EMPLOYEE의 별칭이고 D는 DEPARTMENT의 별칭이다. SELECT절은 이미 직원아이디 E.EID, 직원이름 E.ENAME, 부서이름 D.DNAME을 선택하고 있으므로, WHERE절에서 해당 부서와 그 부서 소속 직원을 정확히 걸러야 한다.

㉠: D.DNAME
조건의 오른쪽 값 '총무팀'은 부서이름이다. 따라서 부서이름 속성 D.DNAME과 비교해야 한다. D.DID는 부서를 식별하는 기본키인 부서번호이므로 여기에 부서이름을 대응시키면 문제의 검색 조건과 달라진다.

㉡: E.DNO = D.DID
문제에서 EMPLOYEE의 DNO는 DEPARTMENT의 DID를 참조하는 외래키라고 명시하였다. 따라서 직원의 소속 부서번호 E.DNO와 부서 테이블의 기본키 D.DID를 같게 연결해야 한다. E.EID는 직원 자체의 기본키이므로 부서번호와 연결할 속성이 아니며, D.DNAME도 외래키가 참조하는 대상이 아니다.

두 조건을 AND로 결합하면 부서이름이 총무팀이면서 해당 부서에 속한 직원만 남는다. 따라서 대응 관계는 **㉠: D.DNAME, ㉡: E.DNO = D.DID**이다.

NoteBase

문 20. 다음과 같이 트랜잭션이 실행 중 시스템 장애가 발생하였다. 즉시 갱신과 로그를 사용하는 회복 기법이 성공적으로 수행되었을 때 데이터 항목 A, B, C의 최종값은? (단, 변경 로그 레코드 형식은 <트랜잭션식별자, 데이터항목, 이전값, 새값>이다)

<start_transaction, T1>
<T1, C, 100, 20>
<commit, T1>
<checkpoint>
<start_transaction, T4>
<T4, B, 20, 15>
<T4, A, 30, 20>
<commit, T4>
<start_transaction, T2>
<T2, B, 15, 25>
<start_transaction, T3>
<T3, A, 20, 40>
<T2, C, 20, 30>
... 시스템 장애 발생 ...

① A: 20, B: 15, C: 20

② A: 20, B: 25, C: 30

③ A: 40, B: 25, C: 30

④ A: 20, B: 25, C: 20

정답 1번

간단한 해설

T1과 T4는 커밋했으므로 결과를 보존하고, 커밋하지 못한 T2·T3의 변경은 이전값으로 되돌린다. 회복 후 값은 **A=20, B=15, C=20**이다.

NoteBase

계산 과정

즉시 갱신에서는 커밋 전의 변경도 저장장치에 반영될 수 있다. 따라서 장애 회복 시 **커밋한 트랜잭션의 결과는 보존하고, 미완료 트랜잭션의 변경은 취소**해야 한다. 변경 로그의 이전값은 UNDO에, 새값은 REDO에 사용한다.

T1은 C를 100에서 20으로 바꾼 뒤 커밋했고, 그 뒤 체크포인트가 있다. 따라서 T1의 결과 C=20은 보존한다. 이 문제의 기본 체크포인트 회복 방식에서는 체크포인트 이전에 완료된 T1을 다시 취소하지 않는다.

T4는 체크포인트 이후 B를 20에서 15로, A를 30에서 20으로 변경하고 커밋했다. 따라서 T4의 **커밋된 새값 B=15, A=20**을 REDO로 보장한다.

T2와 T3에는 커밋 기록이 없으므로 UNDO 대상이다. 로그를 역순으로 따라가면 T2의 C 변경을 30에서 이전값 20으로, T3의 A 변경을 40에서 이전값 20으로, T2의 B 변경을 25에서 이전값 15로 되돌린다. 실제 디스크 반영 여부와 관계없이 회복 결과에서는 미완료 변경이 남지 않아야 한다.

결국 **A=20, B=15, C=20**이 되어 ①과 일치한다. 로그에 가장 나중에 기록된 새값을 그대로 최종값으로 선택하면 미커밋 변경까지 인정하게 되므로 잘못이다.

NoteBase

문 21. 관계형 데이터베이스에서 역정규화(denormalization)에 대한 설명으로 적절한 것은 모두 몇 개인가?

가. 정규형 수준을 높이기 위해 역정규화를 수행한다.
나. 역정규화는 개념적 설계 단계에서 수행한다.
다. 릴레이션의 데이터 중복이 감소되고 데이터베이스의 크기도 작아진다.
라. 둘 이상의 릴레이션에 대한 조인 결과를 자주 이용하는 경우 역정규화를 통해 질의응답 시간을 단축할 수 있다.

① 1
② 2
③ 3
④ 4

정답 1번

간단한 해설

역정규화는 조회 성능을 위해 **일부 중복을 허용**하는 설계 조정이다. 자주 사용하는 조인을 줄일 수 있지만 정규형 수준을 높이거나 중복을 줄이는 작업은 아니다.

NoteBase

보기별 해설

가. X: 역정규화는 정규화된 구조를 성능 요구에 맞게 조정하면서 일부 중복을 의도적으로 허용하는 것이다. 정규형 수준을 높이는 작업은 정규화이며, 역정규화는 오히려 **정규화 제약을 일부 완화**하여 조회 비용 등을 줄이려는 작업이다.

나. X: 개념적 설계에서는 업무에 필요한 개체와 관계 등 데이터의 의미를 표현한다. 역정규화는 이러한 초기 모델링이 아니라, 정규화된 논리적 구조가 마련된 뒤 실제 접근 패턴과 성능을 고려하는 물리적 설계·튜닝 과정에서 검토한다.

다. X: 역정규화는 테이블 병합, 중복 속성이나 미리 계산한 값을 저장하는 방법 등을 사용하므로 일반적으로 **데이터 중복과 저장공간이 늘어날 수 있다.** 중복된 값을 일관되게 갱신하는 추가 관리도 필요하다. 중복을 줄여 저장공간을 절약하는 것이 역정규화의 일반적인 목적은 아니다.

라. O: 자주 함께 조회하는 데이터를 병합하거나 조인 결과를 미리 저장하면 질의 때마다 수행하는 조인과 데이터 접근을 줄일 수 있다. 따라서 반복적인 조인 질의의 응답시간을 단축할 수 있다. 다만 갱신 비용과 중복 데이터의 일관성 관리 부담도 함께 비교하여 적용해야 한다.

적절한 보기는 라 하나이므로 개수 1을 나타내는 ①과 일치한다.

NoteBase

문 22. 다음 데이터베이스 설계 단계를 수행하는 순서대로 나열할 때, 3번째로 수행되는 것은?

가. 개념적 설계
나. 요구조건 분석
다. 데이터베이스 튜닝
라. 물리적 설계
마. 논리적 설계

① 가
② 다
③ 라
④ 마

정답 4번

간단한 해설

요구사항을 개념적 스키마로 표현한 뒤, 이를 목표 데이터 모델에 맞는 **논리적 구조로 변환**한다. 구체적인 저장 구조와 접근 경로를 정하는 물리적 설계는 그다음 단계이다.

NoteBase

올바른 순서

나 → 가 → 마 → 라 → 다

첫째, 나의 요구조건 분석에서는 어떤 데이터를 저장하고 어떤 업무·조회·갱신을 지원해야 하는지 파악한다. 요구사항이 정해져야 이를 반영하는 데이터 구조를 설계할 수 있다.

둘째, 가의 개념적 설계에서는 요구사항을 개체·속성·관계 등으로 표현하여 E-R 모델과 같은 개념적 스키마를 만든다. 특정 저장 방식보다는 업무의 의미와 데이터 사이의 관계에 초점을 둔다.

셋째, **마의 논리적 설계**에서는 개념적 스키마를 목표 데이터 모델에 맞는 논리적 구조로 변환한다. 관계형 데이터베이스라면 테이블·키·제약조건 등을 정하고 정규화 등을 검토한다.

넷째, 라의 물리적 설계에서는 파일 조직, 인덱스, 저장 구조와 접근 경로 등 실제 저장·처리 방법을 결정한다. 논리적 구조를 먼저 정한 뒤 성능과 저장 효율을 고려해 구체화한다.

다섯째, 다의 데이터베이스 튜닝에서는 구현·운영 중 관찰한 성능을 바탕으로 질의, 인덱스와 설정 등을 조정한다. 제시 목록에는 구현 단계가 따로 없지만, 일반적으로 물리적 설계를 구현한 뒤 운영 성능을 측정하고 튜닝한다.

NoteBase

문 23. X, Y, Z가 릴레이션 R의 속성일 때 함수 종속성에 대한 추론 규칙으로 적절하지 않은 것은?

① $X \rightarrow Y$이고 $Y \rightarrow Z$이면 $X \rightarrow Z$이다.

② $X \rightarrow Y$이면 $XZ \rightarrow YZ$이다.

③ $X \rightarrow Y$이고 $X \rightarrow Z$이면 $Y \rightarrow Z$이다.

④ $X \rightarrow Y$이고 $WY \rightarrow Z$이면 $WX \rightarrow Z$이다.

정답 3번

간단한 해설

같은 X가 Y와 Z를 각각 결정한다는 사실만으로 **Y가 Z를 결정한다고 추론할 수 없다**. 서로 다른 X에 같은 Y와 서로 다른 Z가 대응할 수 있기 때문이다.

선지별 해설

함수 종속성 $X \rightarrow Y$는 X 값이 같은 두 튜플의 Y 값도 같아야 한다는 뜻이다. XY는 속성 집합 X와 Y를 합친 것을 나타낸다.

① O: 이행 규칙이다. X 값이 같으면 Y 값이 같고, Y 값이 같으면 Z 값도 같으므로 $X \rightarrow Z$를 추론할 수 있다.

② O: 증대 규칙이다. 두 튜플의 XZ 값이 같으면 X 값과 Z 값이 각각 같다. $X \rightarrow Y$에 의해 Y 값도 같으므로 두 튜플의 YZ 값이 같아져 $XZ \rightarrow YZ$가 성립한다.

③ X: X가 Y와 Z를 각각 결정하더라도 **Y가 Z를 결정하는 것은 아니다**. 예를 들어 (X, Y, Z)가 (1, a, p), (2, a, q)인 두 튜플을 생각하면 X 값별로 Y·Z가 정해져 두 전제는 성립하지만, 같은 Y 값 a에 서로 다른 Z 값 p와 q가 대응하여 $Y \rightarrow Z$는 성립하지 않는다. 두 전제에서 합집합 규칙으로 도출할 수 있는 것은 $X \rightarrow YZ$이다.

④ O: 가이행 규칙이다. $X \rightarrow Y$에 W를 추가하는 증대 규칙을 적용하면 $WX \rightarrow WY$이다. 여기에 $WY \rightarrow Z$와 이행 규칙을 적용하면 $WX \rightarrow Z$를 얻는다.

NoteBase

문 24. 릴레이션 R(A, B, C, D, E, F)의 함수적 종속성이 다음과 같을 때 제2정규형을 만족하도록 가장 적절하게 분해한 것은?

$AB \rightarrow C, A \rightarrow D, B \rightarrow E, B \rightarrow F$

① R1(A, B, C), R2(A, D), R3(B, E, F)

② R1(A, B, C, D), R2(B, E, F)

③ R1(A, B, E, F), R2(A, D)

④ R1(A, B, C, D), R2(B, E), R3(B, F)

정답 1번

간단한 해설

후보키는 AB이고, D는 A에, E·F는 B에만 종속된다. 제2정규형을 위해 **후보키 일부에 대한 부분 함수 종속을 분리**하여 ABC, AD, BEF로 나눈다.

핵심 풀이

1. 후보키 판별
주어진 종속성에서 A와 B는 다른 속성으로부터 유도되지 않으므로 후보키에 모두 필요하다. $AB \rightarrow C$, $A \rightarrow D$, $B \rightarrow E$, $B \rightarrow F$를 적용하면 AB로 모든 속성을 결정할 수 있으므로 후보키는 AB이다.

2. 부분 함수 종속 분리
제2정규형은 비주요 속성이 후보키 전체에 완전 함수 종속되어야 한다. C는 AB 전체에 종속되지만, D는 A에만 종속되고 E·F는 B에만 종속된다. 따라서 $A \rightarrow D$와 $B \rightarrow EF$를 각각 별도 릴레이션으로 분리하고, $AB \rightarrow C$는 함께 남긴다.

3. 분해 결과 확인
R1(A,B,C), R2(A,D), R3(B,E,F)로 분해한다. 밑줄은 기본키를 표시한다. R1의 C는 복합키 AB 전체에 종속되고, R2와 R3에서는 각각 A와 B가 단일 속성 키이므로 해당 부분 함수 종속이 남지 않는다. 분해 후에도 주어진 함수 종속을 각 릴레이션에서 확인할 수 있다.

②와 ④의 R1(A, B, C, D)에는 $A \rightarrow D$라는 부분 함수 종속이 남는다. ③은 속성 C를 누락했을 뿐 아니라 R1(A, B, E, F)에 $B \rightarrow E$, $B \rightarrow F$라는 부분 함수 종속을 그대로 남기므로 적절한 분해가 아니다.

NoteBase

문 25. 다음 SQL을 빈번하게 수행할 때, 생성하면 가장 효과적인 인덱스(index)로 적절한 것은?

```
SELECT E.ENAME, C.CNAME
FROM EMP E, CUSTOMER C
WHERE E.CID = C.CUSTID AND C.AGE = 20;
```

① E.CID에 정의된 인덱스와 C.AGE에 정의된 인덱스

② E.NAME에 정의된 인덱스

③ E.CID에 정의된 인덱스와 C.CUSTID에 정의된 인덱스

④ (C.AGE, C.CUSTID)에 정의된 복합 인덱스

정답 1번

간단한 해설

C.AGE 인덱스는 20세 고객을 찾는 데, E.CID 인덱스는 그 고객번호와 연결되는 직원을 찾는 데 쓰일 수 있다.
나이 조건 검색과 직원 쪽 조인 탐색을 함께 지원하는 조합이다.

NoteBase

선지별 해설

이 질의는 CUSTOMER에서 **C.AGE=20으로 대상을 좁힌 뒤 E.CID=C.CUSTID로 EMP와 연결**할 수 있다. 따라서 고객의 나이 조건 검색과 그 고객에 대응하는 직원 검색을 함께 지원하는 인덱스 조합을 비교한다.

① C.AGE 인덱스로 나이가 20인 고객을 찾고, 각 고객의 CUSTID로 E.CID 인덱스를 탐색하여 대응하는 EMP 행을 찾을 수 있다. **선택 조건과 상대 테이블의 조인 탐색을 모두 지원**한다는 점에서 이 질의에 가장 직접적인 조합이다.

② E.NAME은 제시된 WHERE 절의 검색 조건이나 조인 조건에 사용되지 않는다. SELECT 절의 출력 속성도 E.ENAME이므로, 제시된 E.NAME 인덱스는 나이 조건 처리와 조인 탐색을 직접 개선하는 근거가 없다.

③ E.CID와 C.CUSTID의 인덱스는 조인 키 검색을 지원한다. 그러나 C.AGE=20을 직접 찾는 인덱스가 없으므로, 나이에 따른 사전 필터링을 지원하는 ①과 차이가 있다.

④ (C.AGE, C.CUSTID) 복합 인덱스는 선두 속성인 AGE로 고객을 찾고 그 고객의 조인 키를 얻는 데 유용하다. 다만 이 선택지만으로는 **EMP 쪽 E.CID의 인덱스 탐색**까지 지원하지 않는다. 따라서 제시된 두 처리 단계에 각각 대응하는 ①이 인정 정답이다.

실제 실행시간은 데이터 분포, 기존 인덱스, 테이블 크기와 옵티마이저의 조인 방식에 따라 달라진다. 여기서는 다른 조건이 제시되지 않은 상태에서 질의의 선택·조인 조건에 직접 대응하는 인덱스 구성을 비교한다.

NoteBase

문 26. 트랜잭션 T1, T2, T3의 스케줄과 동등한(equivalent) 충돌직렬가능(conflict serializable) 스케줄에 해당하는 것은?

시간 \ TID	T1	T2	T3
1	read(X);		
2		read(Y);	
3	read(Y);		
4			read(X);
5			read(Z);
6	write(X);		
7			write(Z);
8		read(Z);	
9		write(Y);	

① 없음

② T1 → T2 → T3

③ T2 → T3 → T1

④ T3 → T1 → T2

정답 4번

간단한 해설

같은 데이터에 대한 연산 중 **적어도 하나가 쓰기**인 충돌의 순서를 보존해야 한다. X에서는 T3가 T1보다 앞서고, Y에서는 T1이 T2보다 앞서야 하므로 직렬 실행 순서가 결정된다.

NoteBase

올바른 순서

T3 → T1 → T2

충돌은 서로 다른 트랜잭션이 같은 데이터 항목에 접근하고, 두 연산 중 적어도 하나가 쓰기인 경우에 발생한다. 먼저 실행된 충돌 연산의 트랜잭션에서 뒤의 트랜잭션으로 선행 관계를 연결한다. 아래 선행 관계들에는 순환이 없으므로 충돌 동등한 직렬 스케줄이 존재한다.

T3: 시간 4의 T3 read(X)가 시간 6의 T1 write(X)보다 먼저이므로 **T3가 T1보다 앞서야 한다.** 또한 시간 7의 T3 write(Z)가 시간 8의 T2 read(Z)보다 먼저이므로 T3는 T2보다도 앞서야 한다. 시간 1의 T1 read(X)와 시간 4의 T3 read(X)는 모두 읽기이므로 서로 충돌하지 않아 반대 방향의 선행 관계를 만들지 않는다.

T1: 시간 3의 T1 read(Y)가 시간 9의 T2 write(Y)보다 먼저이므로 **T1이 T2보다 앞서야 한다.** T1은 X에 관한 선행 관계 때문에 T3 뒤에, Y에 관한 선행 관계 때문에 T2 앞에 놓인다. 시간 2의 T2 read(Y)와 시간 3의 T1 read(Y)는 읽기끼리이므로 T2를 먼저 놓아야 하는 근거가 되지 않는다.

T2: Z에 대해서는 T3의 쓰기 뒤에 읽고, Y에 대해서는 T1의 읽기 뒤에 써야 하므로 마지막에 놓인다. 이렇게 배치하면 서로 다른 트랜잭션 사이의 X·Y·Z 관련 충돌 연산 순서를 모두 보존한다.

NoteBase

문 27. 디스크(disk)에 레코드를 배치하는 방법으로 가장 적절하지 않은 것은?

① 하나의 파일에서 레코드들을 저장하는 블록들은 반드시 인접할 필요가 없다.

② 입출력 속도를 향상시키기 위해 레코드들을 저장하는 블록들이 인접하도록 재배치할 수 있다.

③ 파일 내의 각 레코드 크기는 모두 동일해야 한다.

④ 하나의 레코드 길이가 블록 크기를 초과할 경우 두 개 이상의 블록에 저장될 수 있다.

정답 3번

간단한 해설

파일의 레코드가 항상 같은 크기일 필요는 없다. **가변 길이 레코드**를 사용하면 하나의 파일에도 서로 다른 크기의 레코드를 저장할 수 있다.

선지별 해설

① O: 파일의 논리적 레코드 순서와 디스크의 물리적 블록 위치는 반드시 일치하지 않는다. 블록 위치를 관리하는 정보가 있으면 같은 파일의 레코드들을 비인접 블록에도 저장할 수 있다.

② O: 레코드를 저장한 블록을 인접하게 재배치하면 순차적으로 읽을 때 디스크 헤드의 이동을 줄일 수 있다. 따라서 자기 디스크에서는 블록이 흩어져 있을 때보다 입출력 효율을 높일 수 있다.

③ X: 파일은 고정 길이 레코드뿐 아니라 **가변 길이 레코드**도 저장할 수 있다. 가변 길이 필드나 선택적 필드를 포함하면 레코드마다 크기가 달라질 수 있으며, 길이 정보·구분자·슬롯 정보 등을 이용하여 각 레코드의 위치와 경계를 관리한다.

④ O: **스팬드(spanned) 방식**에서는 한 레코드를 여러 블록에 나누어 저장할 수 있다. 레코드가 블록 크기를 초과하면 조각을 둘 이상의 블록에 저장하고 이어지는 부분의 위치를 관리한다. 이와 달리 언스팬드(unspanned) 방식은 한 레코드가 블록 경계를 넘지 않도록 한다.

NoteBase

문 28. 동시성 제어를 위한 다중모드 로크(multi-mode lock)에 대한 설명으로 적절한 것만을 모두 고른 것은?

가. 트랜잭션 T는 항목 X에 읽기 연산을 수행하기 전에 공유 로크(shared lock) 또는 배타 로크(exclusive lock)를 수행해야 한다.
나. 트랜잭션 T는 항목 X에 대한 모든 읽기 연산과 쓰기 연산을 끝낸 후 반드시 로크를 해제해야 한다.
다. 트랜잭션 T는 항목 X에 쓰기 연산을 수행하기 전에는 반드시 배타 로크를 수행해야 한다.
라. 트랜잭션 T가 항목 X에 배타 로크를 보유하고 있다면 다른 트랜잭션 S의 공유 로크를 허용한다.

① 나, 다
② 다, 라
③ 가, 나, 다
④ 가, 나, 라

정답 3번

간단한 해설

읽기는 공유 또는 배타 로크로 보호할 수 있지만, 쓰기는 **배타 로크가 필요**하다. 또한 배타 로크가 걸린 항목에는 다른 트랜잭션의 **공유 로크도 허용되지 않는다**.

보기별 해설

가. O: 공유 로크는 읽기를 허용하고, 배타 로크는 읽기와 쓰기를 모두 허용한다. 따라서 트랜잭션이 X를 읽기 전에는 공유 로크나 배타 로크 중 하나를 획득해야 한다.

나. O: X에 대한 읽기·쓰기가 끝나면 보유한 로크를 해제하여 다른 트랜잭션이 접근할 수 있도록 해야 한다. 다만 이 설명을 '마지막 접근 직후 즉시 해제해야 한다'는 뜻으로 해석해서는 안 된다. 엄격한 2단계 로킹 등에서는 정해진 규약에 따라 트랜잭션 종료까지 로크를 유지할 수 있다.

다. O: 쓰기는 데이터 값을 변경하므로 다른 트랜잭션의 접근과 충돌할 수 있다. 따라서 X에 쓰기 연산을 수행하려면 반드시 **배타 로크를 획득**해야 하며, 공유 로크만 보유한 상태로는 쓸 수 없다.

라. X: 배타 로크는 다른 트랜잭션의 공유 로크 및 배타 로크 모두와 양립할 수 없다. T가 X의 배타 로크를 보유하면 S의 공유 로크 요청도 허용되지 않고 대기해야 한다. 서로 다른 트랜잭션끼리 동시에 보유할 수 있는 것은 **공유 로크와 공유 로크**의 조합이다.

따라서 적절한 보기는 **가, 나, 다**이다.

NoteBase

문 29. 레코드 크기가 200바이트, 레코드 개수가 20,000개, 블록 크기가 1,024바이트일 때, 블로킹 인수(blocking factor)와 필요한 블록의 개수는?

① 블로킹 인수: 5, 블록 개수: 2,000

② 블로킹 인수: 5, 블록 개수: 4,000

③ 블로킹 인수: 10, 블록 개수: 2,000

④ 블로킹 인수: 10, 블록 개수: 4,000

정답 2번

간단한 해설

한 블록에는 200바이트 레코드가 **5개** 들어간다. 20,000개를 5개씩 저장하므로 필요한 블록은 **4,000개**이다.

계산 과정

블로킹 인수는 한 블록에 들어가는 레코드 수이다. 레코드를 여러 블록에 걸쳐 나누어 저장하지 않는 경우에는 **완전한 레코드가 들어가는 정수 개수**만 세므로, 블록 크기를 레코드 크기로 나눈 값의 소수 부분을 버린다.

블로킹 인수는 $\left\lfloor \frac{1,024}{200} \right\rfloor = \lfloor 5.12 \rfloor = 5$이다. 레코드 5개는 1,000바이트를 차지하고 24바이트가 남는다. 6개는 1,200바이트이므로 한 블록에 들어가지 않는다.

필요한 블록 수는 전체 레코드 수를 블로킹 인수로 나누어 올림한다. 마지막 블록에 일부 레코드만 들어가더라도 블록 하나가 필요하기 때문이다. 이 문제에서는 $\left\lceil \frac{20,000}{5} \right\rceil = 4,000$개로 나누어떨어진다.

따라서 블로킹 인수 5, 필요한 블록 수 4,000개인 ②와 일치한다.

NoteBase

문 30. 로그 우선 기록(write-ahead logging)에 관련된 설명으로 가장 적절하지 않은 것은?

① 로그 레코드를 주 메모리(main memory)의 버퍼 블록에 모아 한번에 출력하는 로그 레코드 버퍼링(log record buffering) 기법을 사용하면 로그의 안정적인 관리가 보장된다.

② 트랜잭션 T는 <T, commit> 로그 레코드를 안전한 저장장치에 출력시켜야만 완료 상태가 된다.

③ <T, commit> 로그 레코드를 출력하기 전에 T에 관련된 모든 로그 레코드를 안전한 저장장치에 출력해야 한다.

④ 데이터베이스 버퍼 블록을 출력하기 전에 이 버퍼 블록의 데이터와 관련된 모든 로그 레코드가 안전한 저장장치에 출력되어야 한다.

정답 1번

간단한 해설

메모리 버퍼에만 있는 로그는 장애 시 사라질 수 있으므로 **버퍼링만으로 안정성이 보장되지 않는다**. 변경 데이터보다 복구 로그를 먼저 안전한 저장장치에 기록해야 한다.

선지별 해설

① X. 주기억장치의 로그 버퍼는 휘발성이므로 장애가 나면 아직 출력하지 않은 로그가 사라질 수 있다. 로그 버퍼링은 여러 기록을 모아 쓰면서 입출력 부담을 줄이는 기법이지, 그 자체로 **로그의 영속성을 보장하는 기법은 아니다**. 필요한 로그를 안전한 저장장치에 기록하는 절차가 따로 필요하다.

② O. 정상적인 내구성 보장 방식에서는 트랜잭션의 **커밋 로그가 안전한 저장장치에 기록**되어야 완료를 확정할 수 있다. 커밋 정보를 메모리에만 둔 채 성공을 알리면 장애 후 그 트랜잭션이 완료되었는지 보장할 수 없다.

③ O. 커밋을 확정하기 전에 해당 트랜잭션의 변경 내용을 복구하는 데 필요한 앞선 로그도 안전한 저장장치에 있어야 한다. **변경 로그와 커밋 로그의 기록 순서**를 보장해야 장애 후 커밋된 결과를 재현할 수 있다. 반드시 로그 하나마다 별도의 입출력을 해야 한다는 뜻은 아니다.

④ O. 변경된 데이터 페이지를 저장장치에 내보내기 전에 그 변경에 대응하는 로그를 먼저 안전하게 기록해야 한다. 이것이 **로그 우선 기록**의 핵심이다. 데이터만 반영되고 복구 로그가 없는 상태가 생기는 것을 막는다.

NoteBase

문 31. 다음에서 팬텀 문제(phantom problem)가 발생할 수 있는 것으로 적절한 것만을 모두 고른 것은?

가. READ UNCOMMITTED
나. READ COMMITTED
다. REPEATABLE READ
라. SERIALIZABLE

① 가, 다
② 나, 라
③ 가, 나, 다
④ 가, 나, 다, 라

정답 3번

간단한 해설

팬텀 문제는 같은 검색 조건을 다시 적용했을 때 **조건에 맞는 행의 집합이 달라지는 현상**이다. SQL 표준의 격리수준 구분에서 REPEATABLE READ까지는 이를 허용할 수 있지만, **SERIALIZABLE은 방지**해야 한다.

보기별 해설

SQL 표준의 격리수준별 보장 범위를 기준으로 판단한다. 팬텀 문제의 예는 다른 트랜잭션이 조건에 맞는 행을 삽입·삭제해, 같은 조건으로 다시 조회했을 때 결과 행 집합이 달라지는 경우이다.

가. O: READ UNCOMMITTED는 미확정 데이터의 읽기까지 허용할 수 있는 가장 낮은 격리수준이다. 반복 불가능 읽기나 팬텀 문제를 방지하도록 보장하지 않으므로 팬텀 문제가 발생할 수 있다.

나. O: READ COMMITTED는 다른 트랜잭션의 미확정 변경값을 읽는 더티 리드를 방지한다. 그러나 반복 조회 사이에 다른 트랜잭션이 확정한 삽입·삭제가 반영될 수 있어 팬텀 문제는 발생할 수 있다.

다. O: REPEATABLE READ는 더티 리드와 반복 불가능 읽기를 방지하지만, 표준상 **검색 조건에 해당하는 행 집합의 변화까지 방지할 의무는 없다**. 따라서 팬텀 문제가 허용될 수 있다. 다만 일부 DBMS는 이 수준에서도 팬텀을 방지하는 더 강한 보장을 제공하므로, 모든 구현에서 반드시 발생한다는 뜻은 아니다.

라. X: SERIALIZABLE은 동시 실행의 결과가 트랜잭션을 순차 실행한 결과와 같도록 보장하는 수준이다. 따라서 표준의 팬텀 문제를 허용하지 않으며, '발생할 수 있는 수준'에 포함되지 않는다.

따라서 해당하는 보기는 **가, 나, 다**이다.

NoteBase

문 32. EMP(ENO, ENAME, DNO) **테이블에 대해 트랜잭션 T1, T2가 다음과 같이 실행될 때 발생 가능한 문제점을 해결하는 방법으로 가장 적절하지 않은 것은?**

TID / 시간	T1	T2
1	START TRANSACTION SELECT ENAME FROM EMP WHERE DNO = 1;	
2		START TRANSACTION INSERT INTO EMP VALUES(10, '홍길동', 1); COMMIT;
3	SELECT ENAME FROM EMP WHERE DNO = 1; COMMIT;	

① EMP 테이블의 DNO 속성에 인덱스가 정의되어 있으면, 트랜잭션 T1이 시간 1에 DNO 인덱스에서 값이 1인 엔트리에 공유 로크를 설정한다.

② 트랜잭션 T1이 시간 1에 EMP 테이블에 공유 로크를 설정한다.

③ 트랜잭션들이 2단계 로킹 프로토콜을 준수한다.

④ 트랜잭션 고립 수준을 SERIALIZABLE로 설정한다.

정답 3번

간단한 해설

T1의 두 조회 사이에 조건에 맞는 새 행이 삽입되면 **팬텀 읽기**가 발생할 수 있다. 기존 레코드에 대한 2단계 로킹만으로는 신규 행의 삽입까지 막지 못하므로 테이블·검색 조건·인덱스 키 범위 등을 보호해야 한다.

NoteBase

선지별 해설

T1은 시간 1에 DNO = 1인 사원명을 조회한다. T2가 시간 2에 (10, '홍길동', 1)을 삽입하고 커밋하면, T1의 시간 3 조회에 처음에는 없던 행이 추가될 수 있다. 이는 기존 행의 값이 바뀌는 문제가 아니라 같은 검색 조건에 맞는 행 집합이 달라지는 **팬텀 읽기**이다.

① O: 여기서 인덱스 엔트리는 DNO = 1에 해당하는 레코드 집합을 관리하는 검색키 항목이다. 이 항목을 공유 로크로 보호하면 T2가 같은 값 1을 가진 새 레코드를 추가하면서 필요한 인덱스 항목 변경과 충돌한다. 이를 통해 삽입을 지연시킬 수 있다. 단순히 이미 존재하는 개별 행만 잠그는 것과는 다르며, 실제 구현에서는 해당 검색키의 신규 삽입까지 보호하는 잠금이어야 한다.

② O: EMP 테이블 전체의 공유 로크를 유지하면 다른 트랜잭션이 그 테이블에 새 행을 삽입하기 위해 요구하는 쓰기 잠금과 충돌한다. 따라서 T1이 조회하는 동안 T2의 삽입을 막아 팬텀을 방지할 수 있지만, 잠금 범위가 넓어 동시성이 낮아진다.

③ X: 2단계 로킹은 잠금을 획득하는 확장 단계와 해제하는 축소 단계를 구분하는 규칙이다. **기존 레코드만 잠그는 2단계 로킹만으로는** 아직 존재하지 않는 새 행의 삽입까지 막지 못한다. 팬텀을 방지하려면 테이블이나 검색 조건에 해당하는 인덱스 키·범위 등 삽입과 충돌할 대상을 함께 보호해야 한다. 2단계 로킹 자체가 이런 잠금 대상을 자동으로 보장하는 것은 아니다.

④ O: SERIALIZABLE은 트랜잭션들이 직렬로 실행된 것과 같은 효과를 보장하는 고립 수준이므로 팬텀 읽기를 허용하지 않는다. 구현에 따라 범위 잠금 등으로 충돌 작업을 대기시키거나 직렬화 충돌을 감지하여 트랜잭션을 재시도하게 할 수 있다.

NoteBase

문 33. 경험적 질의 최적화(heuristic query optimization)에서 관계 대수 연산의 동등 변환 규칙을 활용하여 초기 질의 트리(query tree)를 더 효율적으로 변환하기 위한 방법으로 가장 적절하지 않은 것은?

① 셀렉트(select) 연산은 트리에서 가능한 한 아래로 이동한다.

② 카티션 곱(cartesian product) 연산은 후속의 조인 조건을 나타내는 셀렉트 연산과 결합시켜 조인(join) 연산으로 변환한다.

③ 연산의 결과가 많아 선택도(selectivity)가 가장 큰 셀렉트 연산이 먼저 실행되도록 한다.

④ 프로젝트(project) 연산의 속성들을 나눠서 트리에서 가능한 한 아래로 이동하고 필요한 만큼의 프로젝트 연산을 생성한다.

정답 3번

간단한 해설

경험적 최적화에서는 **결과 튜플을 더 적게 남기는 셀렉트**를 먼저 적용하여 중간 결과를 줄인다. 결과가 많이 남는 큰 선택도의 조건을 우선한다는 설명은 반대이다.

선지별 해설

경험적 질의 최적화는 최종 결과를 바꾸지 않는 동등 변환을 이용해 중간 결과의 크기와 처리량을 줄이는 것을 기본 원칙으로 한다. 질의 트리의 아래쪽은 원본 릴레이션에 가까우므로 해당 연산이 더 일찍 수행된다.

① O: 셀렉트를 가능한 한 아래로 이동하면 조건에 맞지 않는 튜플을 먼저 제거하여 이후 조인이나 정렬 등의 입력 크기를 줄일 수 있다. 단, 그 조건에 필요한 속성이 존재하고 연산의 의미가 보존되는 범위에서만 이동해야 한다.

② O: 두 릴레이션의 카티션 곱을 구한 뒤 조인 조건으로 셀렉트하는 연산은 해당 조건의 조인과 동등하다. 따라서 이를 조인 연산으로 바꾸면 **모든 튜플 쌍을 중간 결과로 만드는 낭비**를 줄일 수 있다. 논리적으로 동등한 결과를 더 적절한 조인 처리 방법으로 계산하는 것이다.

③ X: 이 문항에서 선택도는 전체 입력 중 조건을 만족하여 남는 튜플의 비율을 뜻한다. 경험적 기준에서는 결과가 많은 연산이 아니라 **결과 튜플을 더 적게 남기는 셀렉트**를 먼저 적용하는 것이 유리하다. 선택도가 작은 조건을 먼저 처리하면 후속 연산의 입력이 줄어든다. 개별 조건의 계산 비용 등을 모두 평가하는 비용 기반 최적화와는 구별해야 한다.

④ O: 프로젝트를 아래로 이동하여 불필요한 속성을 일찍 제거하면 중간 튜플의 폭과 데이터 이동·저장량을 줄일 수 있다. 여러 입력 릴레이션에 필요한 속성 집합을 나누어 프로젝트를 적용할 수 있지만, **후속 조인·셀렉트와 최종 출력에 필요한 속성**은 반드시 남겨야 한다.

NoteBase

문 34. 다음에서 로그를 이용한 회복 기법에 해당되는 것만을 모두 고른 것은?

가. 지연 갱신(deferred update) 회복 기법
나. 즉시 갱신(immediate update) 회복 기법
다. 검사 시점(checkpoint) 회복 기법
라. 그림자 페이징(shadow paging) 회복 기법

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

정답 2번

간단한 해설

지연 갱신·즉시 갱신·검사 시점은 로그를 활용하여 변경을 재실행·취소하거나 회복 범위를 줄인다. **그림자 페이징은 보존한 페이지 매핑**을 이용하므로 기본적인 로그 회복 방식과 구별된다.

보기별 해설

가 O. 지연 갱신은 트랜잭션이 커밋하기 전에는 변경 내용을 데이터베이스에 반영하지 않고 로그에 기록하는 방식이다. 장애가 나면 커밋한 트랜잭션의 변경을 **로그로 REDO**하여 반영할 수 있다. 미커밋 변경은 데이터베이스에 반영되지 않았으므로 기본 방식에서는 UNDO가 필요하지 않다.

나 O. 즉시 갱신은 커밋 전에도 변경 내용이 데이터베이스에 반영될 수 있다. 따라서 로그의 이전값을 이용해 미완료 변경을 취소하고, 새값을 이용해 완료된 변경을 재실행하는 등 **UNDO·REDO 회복**에 로그를 사용한다.

다 O. 검사 시점은 체크포인트 정보를 기록하여 장애 회복 시 검토할 로그 범위와 대상 트랜잭션을 줄이는 방식이다. 로그 기반 회복을 보조하여 회복 시간을 단축하며, **검사 시점 이후 필요한 로그**를 활용한다. 모든 방식에서 과거 로그를 무조건 전부 버려도 된다는 뜻은 아니다.

라 X. 그림자 페이징은 변경 전 페이지를 가리키는 그림자 페이지 테이블을 보존하고, 변경분은 별도 페이지에 쓰는 방식이다. 실패하면 기존 그림자 매핑으로 돌아가고, 커밋하면 새 매핑을 확정한다. 기본 원리는 **페이지 사본과 매핑 전환**이며 변경 로그를 이용한 회복 기법과 구별된다.

로그를 이용하는 것은 가·나·다이므로 ②와 일치한다.

NoteBase

문 35. 시스템 카탈로그(system catalog)에 대한 설명으로 가장 적절하지 않은 것은?

① 시스템 카탈로그는 데이터베이스 관리자가 생성하고 주기적으로 갱신한다.

② SELECT 문을 사용해 카탈로그 내용을 검색하는 것이 가능하다.

③ 테이블에 대한 다양한 통계 정보를 포함하며, 질의 최적화를 위해 활용된다.

④ 데이터 사전(data dictionary)이라고도 한다.

정답 1번

간단한 해설

시스템 카탈로그는 스키마·제약조건·권한 등 데이터베이스 자체에 관한 정보를 담는다. 이를 생성·유지하는 주체는 **DBA가 아니라 DBMS**이며, 객체 정의가 바뀌면 관련 메타데이터도 관리된다.

선지별 해설

시스템 카탈로그는 실제 업무 데이터와 구별되는 메타데이터 저장소이다. 관리자가 사용하는 정보이지만, 관리자가 일반 업무 테이블처럼 직접 작성해 주기적으로 갱신하는 자료는 아니다.

① X: 시스템 카탈로그는 **DBMS가 생성하고 유지·관리**한다. 테이블 생성·변경·삭제 등의 명령을 처리하면서 관련 메타데이터를 갱신한다. DBA가 스키마 변경이나 통계 수집을 지시할 수는 있지만, 이를 카탈로그 자체를 직접 생성하고 주기적으로 갱신하는 주체라고 설명한 것은 부정확하다.

② O: 시스템 카탈로그의 정보는 시스템 테이블이나 뷰를 통해 제공되며, 필요한 권한이 있으면 SELECT 문으로 조회할 수 있다. 다만 조회할 수 있다는 것이 카탈로그를 임의로 직접 수정해도 된다는 뜻은 아니다.

③ O: 테이블의 행 수, 값의 분포, 인덱스 관련 정보 등 통계정보는 질의 최적화에 사용된다. DBMS는 이러한 정보를 바탕으로 검색 방법이나 조인 순서 등 실행계획의 비용을 추정한다. 통계 수집 시점과 방식은 DBMS 및 설정에 따라 달라질 수 있다.

④ O: 시스템 카탈로그는 데이터베이스 객체의 구조와 관련 정보를 모아 둔다는 점에서 데이터 사전이라고도 한다. 데이터 자체를 설명하는 메타데이터를 저장한다는 의미이다.

NoteBase

문 36. 분산 데이터베이스의 데이터 단편화(data fragmentation)에 대한 설명으로 가장 적절하지 않은 것은?

① 전역 릴레이션에 속한 데이터는 어느 한 단편에는 반드시 속해야 한다.

② 단편화된 전역 릴레이션은 그 단편들로부터 원래의 전역 릴레이션으로 원상회복이 가능해야 한다.

③ 수평 단편화는 릴레이션을 행 단위로 분할하는 것이다.

④ 수직 단편화는 전역 릴레이션을 재구성하기 위해 하나의 수직 단편에만 기본키를 포함해야 한다.

정답 4번

간단한 해설

수직 단편화에서는 나뉜 속성들을 원래 튜플로 다시 연결해야 한다. 이를 위해 **각 수직 단편에 기본키를 포함**하며, 하나의 단편에만 두지 않는다.

선지별 해설

① O: 완전성 조건에 관한 설명이다. 전역 릴레이션의 데이터를 단편으로 나누더라도 **원래 데이터가 누락되어서는 안 된다**. 따라서 각 데이터는 적어도 하나의 단편에 포함되어야 한다.

② O: 재구성 가능성 조건에 관한 설명이다. 단편들을 적절한 관계 연산으로 결합하여 원래 릴레이션을 복원할 수 있어야 한다. 수평 단편은 합집합으로, 수직 단편은 공통 키를 이용한 조인으로 재구성한다.

③ O: 수평 단편화는 **행, 즉 튜플을 기준으로 분할**한다. 예를 들어 직원 릴레이션을 근무 지역에 따라 나누면 각 단편은 같은 속성 구조를 유지하면서 서로 다른 직원 행들을 담는다.

④ X: 수직 단편화는 속성, 즉 열을 기준으로 분할하며, 원래 같은 튜플에 속했던 값들을 다시 연결할 식별자가 필요하다. 따라서 기본키를 하나의 단편에만 두는 것이 아니라 **각 수직 단편에 기본키를 포함**시켜 키를 기준으로 조인할 수 있게 해야 한다. 기본키의 반복 포함은 재구성을 위한 것으로, 키 이외의 속성을 나누는 것과 구별한다.

NoteBase

문 37. OLAP의 다차원 집계 질의에 대한 다음 설명에서 빈칸 ㉠, ㉡에 가장 적절한 것은?

가. (㉠)은 상세한 질의에서부터 보다 일반적인 질의를 한다.
나. (㉡)은 큐브 데이터에 대해 차원의 값을 어떤 범위로 제한해서 부분 큐브 데이터를 생성한다.

① ㉠: 드릴 다운(drill-down), ㉡: 다이싱(dicing)
② ㉠: 드릴 다운, ㉡: 슬라이싱(slicing)
③ ㉠: 롤업(roll-up), ㉡: 다이싱
④ ㉠: 롤업, ㉡: 슬라이싱

정답 3번

간단한 해설

상세 데이터를 더 일반적인 수준으로 집계하는 것은 **롤업**이다. 차원 값의 범위를 제한하여 부분 큐브를 만드는 것은 **다이싱**이다.

빈칸별 해설

가의 ㉠은 **롤업(roll-up)**이다. 상세한 수준의 데이터를 더 높은 계층으로 집계하여 요약한다. 예를 들어 일별 매출을 월별 또는 연도별 매출로 묶는 방향이다. 드릴 다운은 반대로 연도별 요약에서 월별·일별 세부 내용으로 내려가는 연산이다.

나의 ㉡은 **다이싱(dicing)**이다. 차원의 값을 범위나 값의 집합으로 제한하여 원래 큐브의 일부인 부분 큐브를 만든다. 예를 들어 지역을 서울·부산, 기간을 1~3월로 제한해 해당 구간의 매출 큐브를 얻는 방식이다.

슬라이싱(slicing)은 보통 한 차원을 특정 값으로 고정하여 단면을 선택하는 연산이다. 예를 들어 연도를 2025년으로 고정하여 그 해의 지역·상품별 매출을 보는 것이다. **상위 수준 집계인지, 범위 제한인지, 한 값의 단면 선택인지**를 구분한다.

따라서 ㉠은 롤업, ㉡은 다이싱으로 ③과 일치한다.

NoteBase

문 38. 분산 컴퓨팅 환경의 세 가지 특징을 정의하는 CAP 이론과 관련된 설명으로 가장 적절하지 않은 것은?

① CAP은 일관성(consistency), 가용성(availability), 분할감내(partition-tolerance)를 의미한다.

② 분산 데이터 저장소에 네트워크 분할 또는 장애 발생 시 CAP의 세 가지 특성 중 두 개만 보장 가능하다.

③ 은행 데이터베이스 시스템은 가용성 우선(basically available), 부드러운 상태(soft state), 궁극적 일관성(eventually consistent)를 의미하는 BASE 특성을 만족해야 한다.

④ 호텔 예약 사이트를 통해 사용자가 특정 날짜에 특정 호텔 룸 예약을 진행하다가 일시적으로 연결이 끊어진 경우에, 해당 호텔에 대한 이중 예약을 허용한다면 이것은 가용성을 우선적으로 고려한 것이다.

정답 3번

간단한 해설

은행의 이체·잔액 처리에서는 **ACID에 따른 정확한 트랜잭션 처리**가 중요하다. 가용성을 우선하며 일시적인 불일치를 허용하는 BASE를 은행 데이터베이스의 필수 특성으로 제시한 것은 부적절하다.

선지별 해설

① O: C는 일관성, A는 가용성, P는 분할감내를 뜻한다. CAP의 일관성은 여러 노드가 하나의 최신 데이터 사본처럼 동작하는 성질이고, 가용성은 정상 동작 중인 노드가 받은 요청에 응답하는 성질이다. 분할감내는 노드 집단 사이의 통신이 끊기는 상황을 견디는 것을 말한다.

② O: 네트워크가 분할되면 서로 통신할 수 없는 노드들이 최신 상태를 확인하면서 모든 요청에도 응답하는 것을 동시에 보장할 수 없다. 따라서 핵심은 **분할 중 일관성과 가용성을 모두 완전히 보장할 수 없다는 것**이다. '세 가지 중 두 가지'는 이를 요약한 표현이며, 분할이 없는 정상 상태에서도 언제나 둘만 가능하다는 뜻은 아니다.

③ X: BASE의 약어 풀이 자체는 가용성 우선, 부드러운 상태, 궁극적 일관성을 가리킨다. 그러나 은행의 계좌 이체·잔액 갱신 같은 핵심 거래에서는 일부 처리만 반영되거나 서로 다른 잔액을 기준으로 거래되는 것을 막기 위해 **원자성·일관성·격리성·지속성의 ACID 트랜잭션 보장**이 중요하다. 은행 데이터베이스가 반드시 BASE를 만족해야 한다는 설명은 부적절하다. 은행의 모든 부가 서비스가 같은 일관성 정책을 사용해야 한다는 뜻은 아니며, CAP의 일관성과 ACID의 무결성 유지 의미도 완전히 같지는 않다.

④ O: 통신 단절 중 다른 노드의 최신 예약 상태를 확인하지 못해도 각 노드가 예약 요청을 계속 받아 이중 예약을 허용한다면, 전역적으로 일치하는 예약 상태보다 요청 처리를 우선한 것이다. 이는 일관성을 일부 희생하고 가용성을 우선한 사례이다. 반대로 이중 예약을 막기 위해 확인이 불가능한 예약 요청을 중단하면 가용성을 제한하는 선택이 된다.

NoteBase

문 39. 조인(join) 연산을 구현하는 기법으로 가장 적절하지 않은 것은?

① 중첩 루프(nested loop)

② 인덱스 검사(index lookup)

③ 해시 검사(hash lookup)

④ 의미 검사(semantic lookup)

정답 4번

간단한 해설

조인을 실행하려면 두 릴레이션에서 **조인 조건을 만족하는 투플을 찾아 연결**해야 한다. 중첩 루프·인덱스 탐색·해시 탐색은 이에 쓰이지만, 의미 검사는 일반적인 물리적 조인 구현 기법이 아니다.

선지별 해설

조인의 구현 기법은 논리적으로 주어진 조인 조건을 실제 데이터 탐색·비교 절차로 수행하는 방법이다.

① O: 중첩 루프는 한 릴레이션의 각 투플에 대해 다른 릴레이션의 투플을 순회하면서 조인 조건을 검사한다. 개념적으로 단순하고 다양한 조인 조건에 적용할 수 있지만, 탐색 범위가 크면 비교 비용이 증가한다.

② O: 인덱스 검사는 한쪽 릴레이션의 조인 속성 값으로 다른 쪽의 인덱스를 탐색해 일치하는 투플을 찾는 방법이다. 예를 들어 인덱스 중첩 루프 조인은 안쪽 릴레이션 전체를 매번 순회하는 대신 적절한 인덱스를 이용한다.

③ O: 해시 검사는 조인 속성에 해시 함수를 적용해 탐색 범위를 줄이는 방법이다. 해시 조인은 보통 한쪽 입력으로 해시 테이블을 만들고 다른 쪽 입력의 값으로 이를 탐색하며, 주로 동등 조인에 사용된다. 해시 충돌이 있을 수 있으므로 같은 해시값이라는 이유만으로 연결하지 않고 실제 조인 조건도 확인한다.

④ X: **의미 검사(semantic lookup)는 일반적인 물리적 조인 구현 기법이 아니다**. 질의의 테이블·속성 참조나 자료형 등이 올바른지 확인하는 의미 분석과, 두 릴레이션의 투플을 실제로 연결하는 조인 실행은 구분해야 한다.

NoteBase

문 40. 다음 장바구니 트랜잭션에서 빈발항목(frequent itemset)에 해당하는 것으로만 구성된 것은? (단, 최소 지지도는 55%이다)

트랜잭션 ID	구매품목
1	양파, 감자, 마늘
2	감자, 양파
3	맥주, 감자, 두부
4	맥주, 감자, 마늘
5	양파, 맥주

① {감자}, {마늘}

② {양파}, {맥주}

③ {맥주}, {감자, 양파}

④ {감자, 마늘, 맥주}

정답 2번

간단한 해설

5개 거래에서 최소 지지도 55%를 충족하려면 **적어도 3개 거래**에 나타나야 한다. {양파}와 {맥주}는 각각 3개 거래에 포함되어 지지도가 60%이다.

NoteBase

계산 과정

항목집합의 지지도는 전체 거래 중 그 집합의 항목을 모두 포함한 거래의 비율이다. 총 5개 거래이므로 지지도=해당 항목집합을 포함한 거래 수5×100\%이다. 최소 지지도 55%를 충족하려면 5 × 0.55 = 2.75건 이상, 즉 **최소 3건**에 나타나야 한다.

{감자}: 거래 1·2·3·4에 포함되므로 $\frac{4}{5} \times 100\% = 80\%$이다. 기준을 충족한다.

{마늘}: 거래 1·4에 포함되므로 $\frac{2}{5} \times 100\% = 40\%$이다. 기준에 미달한다.

{양파}: 거래 1·2·5에 포함되므로 $\frac{3}{5} \times 100\% = 60\%$이다. 기준을 충족한다.

{맥주}: 거래 3·4·5에 포함되므로 $\frac{3}{5} \times 100\% = 60\%$이다. 기준을 충족한다.

{감자, 양파}: 두 품목을 **한 거래에서 함께 구매**한 거래는 1·2이므로 $\frac{2}{5} \times 100\% = 40\%$이다. 각 품목이 따로 빈발하더라도 그 조합까지 빈발하는 것은 아니다.

{감자, 마늘, 맥주}: 세 품목이 모두 포함된 거래는 4뿐이므로 $\frac{1}{5} \times 100\% = 20\%$이다. 기준에 미달한다.

따라서 제시된 항목집합 중 빈발항목집합은 {감자}, {양파}, {맥주}이다. 선택지 ①에는 {마늘}, ③에는 {감자, 양파}, ④에는 {감자, 마늘, 맥주}가 포함되어 조건을 충족하지 않는다. 빈발항목집합만으로 구성된 것은 {양파}, {맥주}를 제시한 ②이다.

NoteBase