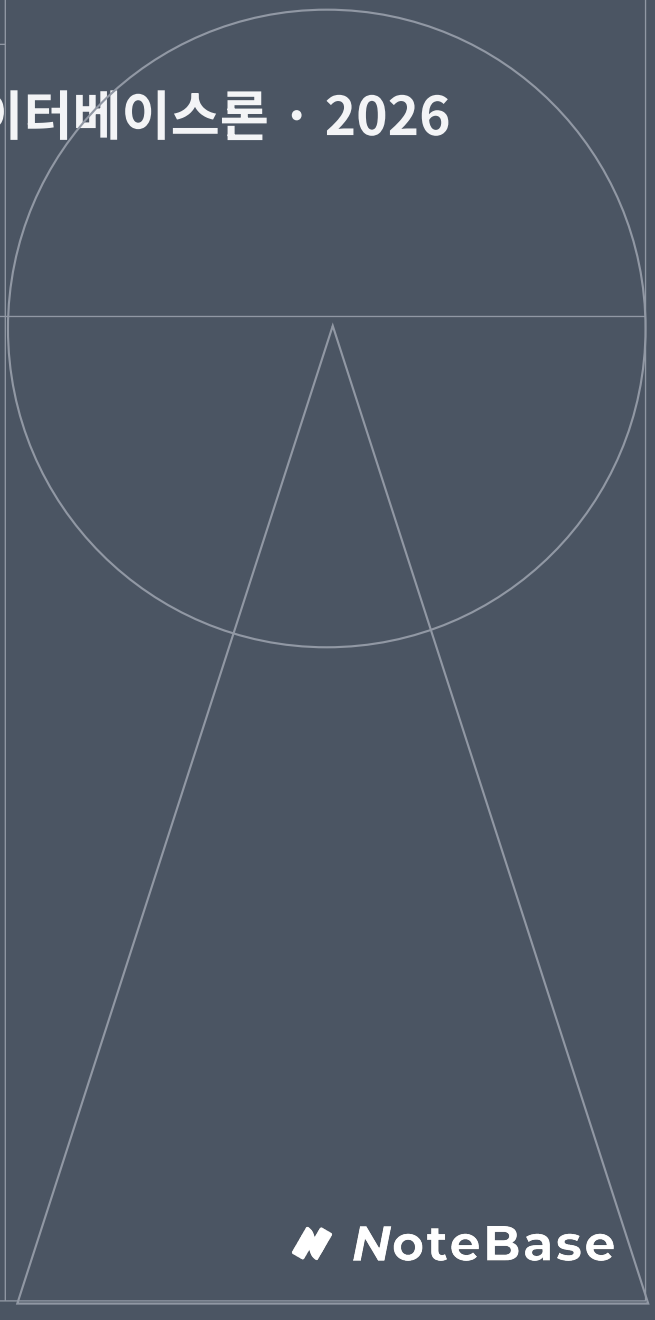
기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2026



NoteBase

기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2026

공무원/경찰/경력채용 · 데이터베이스론 · 2026

문 1. 데이터 언어 중 데이터 제어어(Data Control Language)의 기능에 관한 설명으로 가장 적절하지 않은 것은?

① 데이터베이스에 정확하고 유효한 데이터가 유지되도록 한다.

② 여러 사용자가 데이터에 동시 접근하여 처리할 수 있도록 한다.

③ 허가받지 않은 사용자가 데이터에 접근하는 것을 차단할 수 있다.

④ 데이터의 삽입, 삭제, 갱신, 검색 연산을 수행하고 처리할 수 있다.

정답 4번

간단한 해설

삽입·삭제·갱신·검색은 데이터를 직접 처리하는 **데이터 조작어(DML)의 기능**이다. 데이터 제어어(DCL)의 권한·안전한 이용 통제 기능과 구별해야 한다.

NoteBase

선지별 해설

① O(개론적 제어 기능의 분류): 정확하고 유효한 데이터를 유지하는 것은 **무결성 제어**의 목적이다. 이 문제의 넓은 DCL 분류에서는 데이터의 안전한 이용을 통제하는 기능으로 다룬다. 다만 실제 SQL에서 PRIMARY KEY·CHECK 등의 제약을 선언하는 작업은 DDL에 속하므로, 모든 무결성 기능을 DCL 명령으로만 구현한다는 뜻은 아니다.

② O(개론적 제어 기능의 분류): 여러 사용자가 동시에 작업할 때 서로의 처리가 부당하게 간섭하지 않도록 하는 **병행 제어**를 가리킨다. 동시 이용을 허용하면서 일관성을 지키는 통제 기능이라는 취지이다. 실제 잠금·격리 수준·트랜잭션 처리는 DBMS의 기능이며, 좁은 의미의 GRANT·REVOKE만으로 동시성 제어 전체를 수행하는 것은 아니다.

③ O: 데이터 접근 권한을 허가된 사용자에게 부여하고 불필요한 권한을 회수하는 것은 DCL의 대표적인 보안 기능이다. GRANT와 REVOKE로 접근 권한을 관리하여 권한이 없는 사용자의 데이터 이용을 제한할 수 있다.

④ X: 삽입·삭제·갱신·검색은 **데이터 조작어(DML)**가 담당하는 데이터 처리 연산이다. 각각 INSERT·DELETE·UPDATE·SELECT에 대응한다. DCL은 이런 연산을 누가 수행할 수 있는지 등을 통제하는 기능과 관련되며, 데이터 자체를 조작하는 연산과 구별한다.

NoteBase

문 2. 3단계 스키마와 데이터 독립성에 관한 설명으로 가장 적절하지 않은 것은?

① 개념 스키마는 조직 전체 관점의 스키마이며, 하나의 데이터베이스에 하나만 존재한다.

② 외부 스키마는 개별 사용자 관점의 스키마이며, 하나의 데이터베이스에 여러 개가 존재할 수 있다.

③ 논리적 데이터 독립성은 개념 스키마가 변경되어도 외부 스키마가 영향을 받지 않음을 의미한다.

④ 물리적 데이터 독립성은 외부 스키마가 변경되어도 개념 스키마가 영향을 받지 않음을 의미한다.

정답 4번

간단한 해설

물리적 데이터 독립성은 **내부 스키마 변경으로부터 개념 스키마를 보호**하는 성질이다. 개념 스키마 변경으로부터 외부 스키마를 보호하는 논리적 독립성과 구별해야 한다.

선지별 해설

① O: 3단계 스키마 구조에서 개념 스키마는 **조직 전체 데이터베이스의 논리적 구조**를 통합하여 정의한다. 개별 사용자별 외부 스키마와 달리 이 구조에서는 하나의 데이터베이스에 하나의 개념 스키마를 둔다.

② O: 외부 스키마는 **개별 사용자나 응용 프로그램이 필요로 하는 데이터의 관점**을 정의한다. 사용자마다 필요한 부분과 표현이 다르므로 여러 외부 스키마가 존재할 수 있다.

③ O: 논리적 데이터 독립성은 **개념 스키마를 바꾸더라도 외부 스키마와 응용 프로그램을 바꾸지 않을 수 있는 능력**이다. 필요하면 외부·개념 스키마 사이의 사상을 조정하여 상위 관점을 유지한다.

④ X: 물리적 데이터 독립성은 **내부 스키마가 바뀌어도 개념 스키마가 영향을 받지 않는 것**이다. 예를 들어 저장 위치나 인덱스·파일 구성을 바꾸어도 논리적 테이블 구조를 유지하는 경우이다. 외부 스키마의 변경을 기준으로 설명한 것은 단계가 잘못되었다.

NoteBase

문 3. 관계 데이터 모델의 릴레이션 특성에 관한 설명으로 가장 적절한 것은?

① 하나의 릴레이션을 구성하는 속성들의 순서는 의미가 없다.

② 하나의 릴레이션을 구성하는 각각의 속성은 여러 개의 속성으로 분해가 가능하다.

③ 하나의 릴레이션에 포함된 속성 간 순서는 변경 없이 유지하여야만 병렬 처리가 가능하다.

④ 하나의 릴레이션에 동일한 투플이 여러 개 존재하는 경우에는 해당 투플의 순서를 변경하지 않아야만 관계 연산이 가능하다.

정답 1번

간단한 해설

릴레이션은 원자값을 가진 튜플의 집합이다. **속성과 튜플의 순서에는 논리적 의미가 없고**, 동일한 튜플은 중복되지 않는다.

선지별 해설

① (O) 관계 모델에서는 속성을 이름으로 구별하므로 **속성의 나열 순서는 논리적 의미를 바꾸지 않는다**. 속성과 해당 값의 대응을 그대로 유지한다면 열의 표시 순서를 바꾸어도 같은 정보를 표현한다.

② (X) 기본적인 관계 모델에서 각 속성값은 해당 도메인에서 하나의 **원자값**으로 취급한다. ER 모델의 복합 속성처럼 각각의 속성을 다시 여러 하위 속성으로 분해할 수 있다는 것을 릴레이션의 기본 특성으로 보지는 않는다.

③ (X) 속성 순서를 고정해야 병렬 처리가 가능하다는 제약은 관계 모델의 특성이 아니다. **논리적인 열 순서의 무의미성**과 DBMS가 선택하는 물리적 실행·병렬 처리 방식은 별개의 문제이다.

④ (X) 릴레이션은 튜플의 집합이므로 **동일한 튜플의 중복을 허용하지 않고 튜플의 순서도 의미가 없다**. 따라서 중복 튜플의 순서를 고정해야 관계 연산이 가능하다는 전제 자체가 틀리다. SQL의 일부 질의 결과가 중복을 허용하는 것과 순수 관계 모델은 구분한다.

NoteBase

문 4. 키(key)의 특성에 관한 설명으로 가장 적절하지 않은 것은?

① 기본키는 개체 무결성 제약조건에 따라 널(null) 값을 가질 수 없다.

② 외래키는 릴레이션 무결성 제약조건에 따라 널(null) 값을 가질 수 없다.

③ 슈퍼키는 유일성을 만족하는 속성(들)이며, 하나의 릴레이션에 여러 개 있을 수 있다.

④ 후보키는 유일성과 최소성을 만족하는 속성(들)이며, 하나의 릴레이션에 여러 개 있을 수 있다.

정답 2번

간단한 해설

외래키라는 이유만으로 NULL이 금지되는 것은 아니다. 별도의 NOT NULL 제약이나 기본키 참여 여부 등에 따라 NULL 허용 여부가 달라진다.

선지별 해설

① O: 기본키는 각 투플을 식별하므로 **NULL을 허용하지 않는다**. 이는 개체 무결성의 요구이며, 복합 기본키를 구성하는 속성에도 적용된다.

② X: 외래키에는 기본키처럼 일률적인 NULL 금지 규칙이 적용되지 않는다. **별도의 NOT NULL 등의 제약이 없다면 NULL을 허용할 수 있다**. 외래키가 주로 연결되는 것은 참조 무결성이며, 참조가 필요한 값은 참조 대상 키에 존재해야 한다. 외래키가 기본키의 일부이거나 NOT NULL로 선언되면 NULL이 금지되므로, 항상 허용된다는 뜻도 아니다.

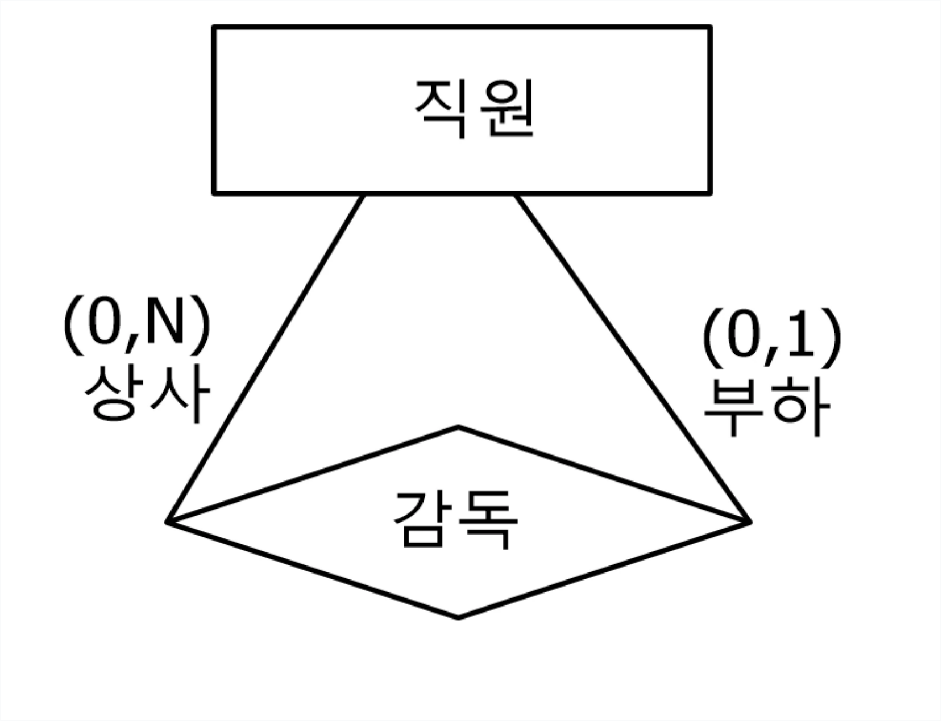
③ O: 슈퍼키는 투플을 **유일하게 식별하는 속성 집합**이다. 최소성은 요구하지 않으므로 후보키에 불필요한 속성을 더한 집합도 슈퍼키가 될 수 있다. 따라서 하나의 릴레이션에 여러 슈퍼키가 존재할 수 있다.

④ O: 후보키는 **유일성과 최소성을 모두 만족**하는 속성 집합이다. 최소성이란 속성을 하나라도 제거하면 유일한 식별 능력을 잃는다는 뜻이다. 서로 다른 여러 속성 집합이 이 조건을 만족할 수 있으며, 그중 하나를 기본키로 선택한다.

따라서 외래키가 NULL을 가질 수 없다고 일반화한 ②가 가장 적절하지 않다.

NoteBase

문 5. 다음의 ER 다이어그램은 직원 개체(entity)에 대한 감독 관계(relationship)를 개체 타입의 각 참여 역할마다 정수의 쌍(min, max)을 연계시키는 방법으로 표현한 것이다. 이 다이어그램에 관한 설명으로 가장 적절하지 않은 것은?



① 감독 관계의 차수는 2이다.

② 한 직원은 여러 명의 부하를 둘 수 있다.

③ 상사가 없는 직원도 존재할 수 있다.

④ 직원이 상사로서 감독 관계에 참여하는 것은 선택적이다.

정답 1번

간단한 해설

직원이라는 **하나의 개체 타입**이 상사와 부하 역할로 참여하는 순환 관계이다. 개체 타입 수에 따른 분류에서는 차수가 1이며, 두 역할의 최소값이 모두 0이므로 각각의 참여는 선택적이다.

NoteBase

선지별 해설

① X: 이 문제처럼 참여하는 **개체 타입의 수로 차수를 분류**하면, 감독 관계에는 직원이라는 한 개체 타입만 참여하므로 차수는 1이다. 상사와 부하는 서로 다른 개체 타입이 아니라 같은 직원 타입의 두 역할이며, 이러한 관계를 단항 또는 순환 관계라고 한다.

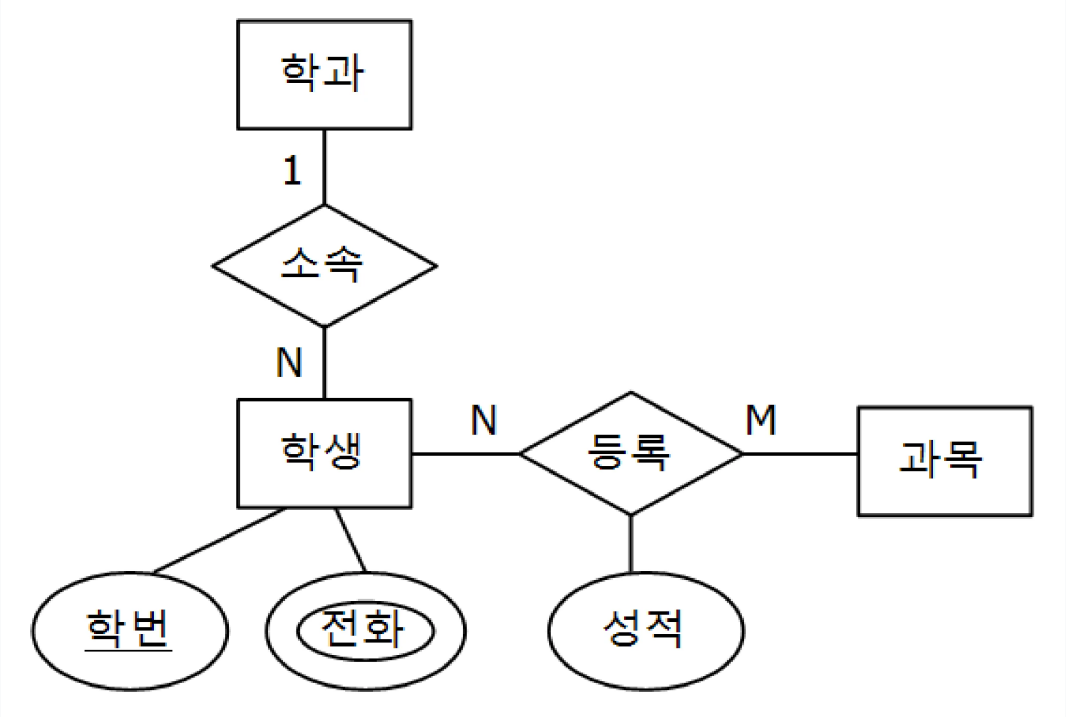
② O: 상사 역할의 최대 참여 횟수가 **N**이므로 한 직원이 상사로서 여러 감독 관계에 참여하여 여러 명의 부하를 둘 수 있다. 부하 역할의 최대값 1은 한 부하에게 허용되는 상사가 최대 한 명이라는 뜻이다.

③ O: 부하 역할의 최소 참여 횟수가 **0**이므로 어떤 직원은 부하로서 감독 관계에 전혀 참여하지 않을 수 있다. 따라서 상사가 없는 직원도 허용된다.

④ O: 상사 역할의 최소 참여 횟수도 **0**이므로 모든 직원이 반드시 상사로 참여할 필요는 없다. 부하가 없는 직원도 허용되므로 상사로서의 참여는 선택적이다.

NoteBase

문 6. 다음의 대학 관련 ER 다이어그램을 관계 데이터베이스 스키마로 사상할 때, 독립적인 릴레이션으로 표현되는 것으로 가장 적절하지 않은 것은?



① 학생 개체

② 등록 관계

③ 소속 관계

④ 전화 속성

정답 3번

간단한 해설

소속은 학과와 학생의 **1:N 관계**이므로, 학과의 키를 학생 릴레이션의 외래키로 넣어 표현하는 것이 기본 사상 방식이다. 별도 소속 릴레이션은 필요하지 않지만, N:M인 등록과 다중값인 전화는 별도 릴레이션으로 사상한다.

NoteBase

선지별 해설

① O: 학생은 독립적인 개체 유형이므로 학생 릴레이션으로 사상한다. 그림에서 밑줄 친 학번이 기본키이며 학생 한 명을 식별한다. 학과와의 소속 관계는 이 학생 릴레이션에 학과를 참조하는 외래키를 포함하여 나타낼 수 있다.

② O: 등록은 학생과 과목 사이의 **N:M 관계**이므로 연결 릴레이션을 별도로 만든다. 학생의 키인 학번과 과목의 키를 외래키로 포함하고, 표준 사상에서는 두 키의 조합을 기본키로 삼는다. 관계에 붙은 성적 속성도 이 등록 릴레이션에 저장한다.

③ X: 소속은 학과 1개에 학생 여러 명이 연결되는 **1:N 관계**이다. 표준 외래키 사상에서는 1측 학과의 기본키를 N측 학생 릴레이션에 외래키로 넣으므로 별도 소속 릴레이션이 필요하지 않다. 별도 테이블로 구현하는 설계가 절대 불가능하다는 뜻이 아니라, 이 문제에서 적용하는 기본 사상 방식에서 다른 세 항목과 구별된다는 뜻이다.

④ O: 전화가 이중 타원으로 표시되어 있으므로 학생의 **다중값 속성**이다. 한 학생의 여러 전화번호를 원자값으로 저장하기 위해 학번과 전화번호를 담은 별도 릴레이션을 만든다. 학번은 학생을 참조하는 외래키이며, 학번과 전화번호의 조합으로 각 행을 식별할 수 있다.

NoteBase

문 7. 다음의 관계 대수 연산에 관한 설명으로 옳은 것만을 모두 고른 것은?

㉠ 프로젝트 연산의 결과 릴레이션의 투플 수는 항상 원래 릴레이션의 투플 수보다 같거나 작다.
㉡ 합집합, 차집합, 실렉트, 프로젝트, 조인 연산은 기본 연산(primitive operation)에 속한다.
㉢ 차집합 R-S의 카디널리티는 릴레이션 R이나 S의 어떤 카디널리티보다도 크지 않다.
㉣ R⋉S의 의미는 S와의 자연 조인에 참여할 수 있는 R의 투플만을 선택하는 것이다.

① ㉠㉡
② ㉠㉣
③ ㉡㉢
④ ㉠㉢㉣

정답 2번

간단한 해설

프로젝트는 중복 제거로 튜플 수가 늘지 않고, 세미조인은 대응이 있는 왼쪽 튜플만 남긴다. 차집합의 크기는 **왼쪽 릴레이션의 크기로만 상한이 보장**되며 오른쪽보다 클 수 있다.

보기별 해설

㉠ (O) 프로젝트는 필요한 속성만 남기며, 그 결과 같은 튜플이 생기면 집합 의미에 따라 중복을 제거한다. 따라서 **결과 튜플 수는 원래와 같거나 감소**하며 증가하지 않는다.

㉡ (X) 통상적인 기본 연산 집합에는 합집합·차집합·실렉트·프로젝트·카티션곱 등이 들어간다. 조인은 카티션곱에 조건을 적용하는 방식 등으로 표현할 수 있는 **유도 연산**이므로, 나열한 연산을 모두 기본 연산으로 묶은 설명은 틀리다.

㉢ (X) R-S는 R에 있으면서 S에는 없는 튜플의 집합이므로 **R의 카디널리티 이하**라는 보장은 있다. 그러나 S보다 클 수 있다. 예를 들어 같은 한 속성의 릴레이션 R={(1),(2),(3)}, S={(4)}이면 R-S는 3행이고 S는 1행이다.

㉣ (O) R⋉S는 왼쪽 세미조인으로, S와 자연조인 조건을 만족하는 대응이 있는 **R의 튜플만 남긴다**. 결과 속성은 R의 속성이며 S의 속성을 덧붙이는 일반 조인과 다르다.

따라서 옳은 것은 ㉠과 ㉣이므로 ②이다.

NoteBase

문 8. 다음의 두 릴레이션에 대해 주어진 관계 대수 연산을 수행한 후 결과 테이블 T의 카디널리티(cardinality)는? (단, 밑줄 속성은 기본키, π는 프로젝션 연산자, ÷는 디비전 연산자임)

프로젝트참여

eno	pno	hour
e1	p1	30
e2	p3	20
e3	p3	40
e1	p3	40
e2	p1	10

프로젝트

pno	pname
p1	차세대시스템
p3	AI에이전트

<관계 대수 연산>
$R \leftarrow \pi_{eno,pno}(\text{프로젝트참여})$
$S \leftarrow \pi_{pno}(\text{프로젝트})$
$T \leftarrow R \div S$

① 1
② 2
③ 3
④ 5

정답 2번

간단한 해설

디비전은 S의 **모든 프로젝트 p1·p3에 참여한 직원**을 구한다. e1과 e2가 조건을 만족하므로 결과의 카디널리티는 2이다.

NoteBase

계산 과정

1. 프로젝트참여에서 eno와 pno만 프로젝션하면 R={(e1,p1), (e2,p3), (e3,p3), (e1,p3), (e2,p1)}이다. hour 속성은 제거되므로 참여 시간의 크기는 이후 판정에 영향을 주지 않는다.

2. 프로젝트에서 pno만 프로젝션하면 S={p1,p3}이다. **R÷S는 S의 모든 pno와 짝을 이루는 eno**를 구한다. 프로젝트 중 하나에만 참여한 직원을 고르는 연산이 아니다.

3. e1은 p1·p3 모두에 참여하고, e2도 p1·p3 모두에 참여한다. e3은 p3에만 참여하므로 제외된다. 따라서 T의 eno 값은 {e1,e2}이며 **카디널리티는 2**이다.

따라서 결과 투플 수 2를 나타내는 ②가 정답이다.

NoteBase

문 9. 다음의 참조 무결성 조치 내용을 반영하도록 부서 릴레이션을 SQL문으로 정의할 경우 ㉠, ㉡에 들어갈 내용으로 가장 적절한 것은?

<참조 무결성 조치>
- 직원 테이블에서 관리자인 직원의 직원번호 변경 시 부서 테이블의 해당 관리자번호도 동일하게 변경
- 직원 테이블에서 관리자인 직원의 삭제 시 부서 테이블의 해당 부서의 관리자번호를 디폴트값으로 설정

<SQL 정의문>

```
CREATE TABLE 부서
(...,
 관리자번호 INT NOT NULL DEFAULT 1,
 FOREIGN KEY(관리자번호) REFERENCES 직원(직원번호)
    ON DELETE ㉠
    ON UPDATE ㉡ );
```

① ㉠: NO ACTION, ㉡: CASCADE

② ㉠: SET NULL, ㉡: RESTRICT

③ ㉠: CASCADE, ㉡: SET DEFAULT

④ ㉠: SET DEFAULT, ㉡: CASCADE

정답 4번

간단한 해설

삭제 시 외래 키를 기본값으로 바꾸는 조치는 **SET DEFAULT**, 참조 대상 키의 변경을 외래 키에 전파하는 조치는 **CASCADE**이다. 빈칸 앞의 ON DELETE와 ON UPDATE를 기준으로 구별한다.

NoteBase

빈칸별 해설

㉠ SET DEFAULT: ON DELETE는 참조 대상인 직원 행이 삭제될 때의 조치를 지정한다. 부서 행을 삭제하는 것이 아니라 해당 관리자번호를 기본값으로 바꾸라는 조건이므로 **SET DEFAULT**가 필요하다. 제시된 DEFAULT 1에 따라 값은 1로 설정되며, 이 값도 외래 키 제약을 만족해야 한다. SET NULL은 기본값 설정과 다르고 NOT NULL 조건에도 맞지 않는다.

㉡ CASCADE: ON UPDATE는 참조 대상인 직원번호가 변경될 때의 조치를 지정한다. 이를 참조하는 부서의 관리자번호도 같은 새 번호로 변경하려면 **CASCADE**를 사용한다. RESTRICT나 NO ACTION으로 제약 위반을 막는 것은 변경된 번호를 전파하는 조치가 아니다.

따라서 ㉠은 SET DEFAULT, ㉡은 CASCADE이며 이 조합에 해당하는 것은 ④이다.

NoteBase

문 10. SQL에서 사용하는 집계 함수(aggregate function)에 관한 설명으로 가장 적절한 것은?

① 집계 함수를 사용하는 경우, 서브 쿼리와 반드시 함께 사용해야 한다.

② 널(null) 값이 포함된 특정 속성에 대해 SUM(속성 이름)을 실행하는 경우, 해당 속성의 실행 결과는 무조건 0이다.

③ 릴레이션에 대해 COUNT(*)를 실행하는 경우, 널(null) 값인 투플을 포함하여 모든 투플의 개수를 출력한다.

④ 특정 속성에 대해 중복 값의 개수를 출력하려는 경우, DISTINCT(속성 이름)과 같이 사용해야 한다.

정답 3번

간단한 해설

COUNT(*)는 속성값이 아니라 행을 세므로 **NULL을 포함한 행도 집계**한다. SUM과 COUNT(속성)은 NULL을 제외하고, DISTINCT는 중복을 제거한다.

선지별 해설

① (X) 집계 함수는 SELECT SUM(금액) FROM 거래처럼 **하나의 질의 안에서 바로 사용**할 수 있다. 서브쿼리는 집계 결과를 다른 질의에 활용하는 등의 경우에 사용할 수 있지만 필수 조건은 아니다.

② (X) SUM은 입력 중 NULL을 제외한 값을 더한다. 예를 들어 값이 10, NULL, 20이면 합은 30이므로 **NULL이 포함되었다고 결과가 0이 되지 않는다**. 더할 비NULL 값이 하나도 없으면 일반적으로 NULL을 반환하며, 0이 필요하면 별도의 대체 처리를 한다.

③ (O) COUNT(*)는 특정 속성값이 아니라 **집계 대상의 행 자체를 센다**. 따라서 속성에 NULL이 들어 있는 행도 포함한다. 반면 COUNT(속성)은 그 속성값이 NULL인 행을 제외한다.

④ (X) COUNT(DISTINCT 속성)은 **중복을 제거한 비NULL 값의 종류 수**를 센다. 중복 값이 나타난 횟수나 중복된 행 수를 세는 것과 다르다. 예를 들어 5,5,7,NULL에서는 COUNT(속성)=3, COUNT(DISTINCT 속성)=2이다.

NoteBase

문 11. SQL 뷰(view)에 관한 설명으로 가장 적절하지 않은 것은?

① 뷰는 다른 테이블로부터 유도된 가상 테이블이다.

② 뷰를 제거하려면 DROP문을 사용한다.

③ 모든 뷰는 삽입, 삭제, 갱신 연산에 아무런 제약이 없다.

④ 하나의 뷰는 다른 뷰를 기반으로 정의될 수 있다.

정답 3번

간단한 해설

집계나 그룹화 등으로 결과를 만든 뷰는 기초 행의 변경과 대응시키기 어려워 **삽입·삭제·갱신이 제한될 수 있다.** 따라서 모든 뷰를 아무 제약 없이 갱신할 수 있다는 설명은 틀리다.

선지별 해설

① O: 일반적인 뷰는 저장된 질의로 정의되는 **가상 테이블**이다. 기본 테이블 등의 데이터를 선택·결합한 결과를 테이블처럼 사용할 수 있으며, 일반 뷰는 그 결과 행을 독립적인 데이터 사본으로 저장하는 물리화 뷰와 구별된다.

② O: 뷰의 정의를 제거할 때는 **DROP VIEW**를 사용한다. 이는 뷰를 삭제하는 것이며, 단순히 뷰를 제거한다고 기초 테이블의 데이터까지 삭제되는 것은 아니다. 다른 객체가 의존하고 있다면 삭제에 제약이 있을 수 있다.

③ X: **모든 뷰가 제약 없이 갱신 가능한 것은 아니다.** 집계·그룹화·중복 제거 등을 포함한 뷰는 결과 행을 어느 기초 행의 변경으로 대응시킬지 모호하여 삽입·삭제·갱신이 제한될 수 있다. 단순한 뷰 중 갱신 가능한 것도 있지만, 뷰 정의와 DBMS의 지원 조건을 확인해야 한다.

④ O: 뷰 정의의 질의에서 다른 뷰를 참조할 수 있으므로 하나의 뷰를 또 다른 뷰 위에 정의할 수 있다. 이 경우 상위 뷰는 하위 뷰의 정의와 그 기초 데이터에 의존하므로, 참조 대상의 변경·삭제가 영향을 줄 수 있다.

NoteBase

문 12. 릴레이션 처리 시 발생할 수 있는 이상(anomaly) 현상에 관한 설명으로 가장 적절하지 않은 것은?

① 릴레이션에 포함된 중복 투플 중 일부 투플의 속성 값만 갱신되었을 때 정보의 모순이 발생할 수 있다.

② 릴레이션에서 하나의 투플이 삭제될 때 꼭 필요한 데이터까지도 삭제되어 정보의 손실이 발생할 수 있다.

③ 릴레이션에 데이터를 삽입할 때 원하지 않은 데이터도 함께 삽입되어야 하는 문제가 발생할 수 있다.

④ 릴레이션의 모든 투플에 표준화 처리를 하지 않을 경우, 데이터 분산 처리에 영향을 주는 문제가 발생할 수 있다.

정답 4번

간단한 해설

릴레이션 설계의 대표적 이상은 **갱신·삭제·삽입 이상**이다. 투플의 표준화 누락이 분산 처리에 영향을 준다는 설명은 이 분류에 해당하지 않는다.

선지별 해설

① O(갱신 이상의 취지): 여러 투플에 반복 저장된 동일 사실을 일부에서만 고치면 값이 서로 달라지는 **갱신 이상**이다. 원문의 “중복 투플”은 엄밀하게는 “동일 정보가 중복 저장된 투플들”로 이해해야 한다. 관계 모델에서 모든 속성 값이 동일한 투플의 중복을 허용한다는 뜻은 아니다.

② O: 한 사실을 지우려고 투플을 삭제했는데 다른 필요한 사실까지 사라지는 것은 **삭제 이상**이다. 예를 들어 마지막 수강 기록을 삭제하면서 그 행에 함께 기록된 강좌 정보까지 잃는 경우이다.

③ O: 원하는 사실을 저장하려면 무관한 다른 사실까지 함께 채워야 하거나, 그 정보가 없어 삽입하지 못하는 것은 **삽입 이상**이다. 예를 들어 수강생이 없다는 이유로 새 강좌 정보를 독립적으로 넣지 못하는 경우이다.

④ X: 릴레이션 설계에서 말하는 대표적 이상은 **갱신·삭제·삽입 이상**이다. 모든 투플의 표준화 처리 여부가 분산 처리에 영향을 준다는 설명은 이러한 이상의 정의가 아니다. 정규화는 종속성과 구조를 검토하여 중복 및 이상을 줄이는 과정이지, 투플마다 분산 처리를 위한 표준화 작업을 하는 것이 아니다.

따라서 관계 설계의 이상 현상을 설명하지 않은 ④가 가장 적절하지 않다.

NoteBase

문 13. 병행 제어를 하지 않은 다수의 트랜잭션이 동일한 데이터를 동시에 처리하는 경우 발생할 수 있는 문제로 가장 적절하지 않은 것은?

① 즉시 갱신(immediate update)

② 갱신 분실(lost update)

③ 연쇄 복귀(cascading rollback)

④ 모순성(inconsistency)

정답 1번

간단한 해설

즉시 갱신은 갱신·회복 방식이고, 갱신 분실·연쇄 복귀·모순성은 부적절한 병행 실행에서 발생할 수 있는 문제이다. 커밋 전에 변경을 반영하는 방식 자체와 트랜잭션 간 간섭으로 생긴 문제를 구별해야 한다.

개념 구분

즉시 갱신은 **트랜잭션이 커밋되기 전에도 데이터베이스에 변경을 반영할 수 있게 하는 갱신·회복 방식**이다. 실패 시 미완료 변경을 되돌릴 수 있도록 로그와 UNDO 등의 회복 수단을 마련한다. 병행 처리의 부작용을 가리키는 용어가 아니며, 적절한 병행 제어와 함께 사용할 수 있다.

갱신 분실은 두 트랜잭션의 갱신이 겹쳐 한쪽 결과가 다른 쪽에 덮여 사라지는 문제이다.

연쇄 복귀는 미완료 트랜잭션의 변경값에 의존한 다른 트랜잭션까지 함께 롤백해야 하는 현상이다.

모순성은 관련 데이터를 읽거나 갱신하는 작업이 부적절하게 끼어들어 서로 맞지 않는 상태나 결과가 생기는 문제이다. 이들은 병행 제어가 없을 때 발생할 수 있는 문제라는 점에서 즉시 갱신과 구별된다.

NoteBase

문 14. 다음 직원 테이블에 SQL문을 실행한 결과의 카디널리티(cardinality)로 가장 적절한 것은? (단, 밑줄 속성은 기본키임)

직원

사번	부서	급여
G1	A	300
G2	A	350
G3	B	280
G4	B	320
G5	C	400

<SQL문>

```
SELECT 사번
FROM 직원 e
WHERE 급여>(SELECT AVG(급여) FROM 직원
                    WHERE 부서 = e.부서);
```

① 1

② 2

③ 3

④ 4

정답 2번

간단한 해설

각 직원의 급여를 소속 부서의 평균과 비교한다. **A부서의 G2와 B부서의 G4만 평균을 초과**하므로 결과 카디널리티는 2이다.

NoteBase

계산 과정

서브쿼리의 부서=e.부서는 현재 바깥 행과 같은 부서의 직원들을 고른다. 따라서 비교 기준은 전체 평균이 아니라 **각 직원이 속한 부서의 평균 급여**이다.

A부서의 평균은 $\frac{300+350}{2}=325$이다. G1의 300은 평균보다 작고 G2의 350은 크므로 **G2만 선택**된다.

B부서의 평균은 $\frac{280+320}{2}=300$이다. G3의 280은 평균보다 작고 G4의 320은 크므로 **G4만 선택**된다.

C부서에는 G5만 있어 평균이 400이다. 조건은 크거나 같음이 아니라 급여>평균이므로 400>400은 거짓이고 G5는 제외된다.

결과는 G2·G4의 **2행**이므로 카디널리티는 2, ②이다.

NoteBase

문 15. 고객작품 릴레이션의 후보키와 함수종속이 다음과 같을 때, 이 릴레이션이 속한 정규형으로 가장 적절한 것은? (단, 모든 속성 값은 원자 값임)

고객작품(고객번호, 고객명, 가격, 작품번호, 작품명, 화가)
후보키: (고객번호, 작품번호)
함수종속: (고객번호, 작품번호) → 가격
고객번호 → 고객명
작품번호 → (작품명, 화가)

① 1NF
② 2NF
③ 3NF
④ BCNF

정답 1번

간단한 해설

원자 값 조건으로 1NF는 만족하지만, **고객번호 → 고객명**과 **작품번호 → (작품명, 화가)**가 복합 후보키에 대한 부분 함수 종속이므로 2NF는 만족하지 않는다. 따라서 가장 높은 정규형은 1NF이다.

핵심 풀이

1NF 충족: 모든 속성 값이 원자 값이라는 조건이 있으므로 제1정규형을 만족한다.

2NF 위반: 후보키는 (고객번호, 작품번호)인데, 고객명은 고객번호만으로 결정되고 작품명과 화가는 작품번호만으로 결정된다. 이들은 후보키에 포함되지 않는 속성이 후보키의 일부에 종속되는 **부분 함수 종속**이다.

최종 판단: 2NF가 되려면 1NF를 만족하면서 이러한 부분 함수 종속이 없어야 한다. 가격이 복합 후보키에 종속된다는 사실만으로 다른 속성의 부분 종속이 사라지지는 않는다. 따라서 주어진 릴레이션의 가장 높은 정규형은 ① **1NF**이며, 2NF를 전제로 하는 3NF와 그보다 강한 BCNF도 만족하지 않는다.

NoteBase

문 16. 병행 제어 기법 중 2단계 로킹 규약(2-phase locking protocol)에 관한 설명으로 가장 적절한 것은?

① 교착 상태가 발생하지 않음을 보장할 수 있다.

② 확장 단계에서는 lock만 수행 가능하다.

③ 일부 트랜잭션만 해당 규약을 준수하여도 스케줄의 직렬 가능성을 항상 보장할 수 있다.

④ 데이터를 점유한 하나의 트랜잭션은 전용 로크(exclusive-lock) 수행을 통해 다른 트랜잭션과 데이터 사용 권한을 모두 공유할 수 있다.

정답 2번

간단한 해설

2단계 로킹의 **확장 단계는 로크 획득만 허용하고 해제는 금지**한다. 첫 해제 후의 축소 단계에서는 새 로크를 획득할 수 없으며, 이 규약 자체가 교착 상태까지 방지하는 것은 아니다.

선지별 해설

① X: 기본 2단계 로킹은 교착 상태를 방지하지 않는다. 예를 들어 두 트랜잭션이 서로 다른 항목의 전용 로크를 가진 채 상대가 가진 항목의 로크를 기다리면 순환 대기가 생길 수 있다. **직렬 가능성 보장과 교착 상태 방지**는 다른 문제이며, 교착 상태에는 별도 예방·탐지·해결 기법이 필요하다.

② O: 로크 연산의 관점에서 **확장 단계에서는 새 로크를 획득할 수 있지만 해제할 수 없다.** 첫 로크를 해제하면 축소 단계로 들어가며, 그 뒤에는 로크를 해제할 수만 있고 새 로크를 획득할 수 없다. ‘lock만’은 읽기·쓰기 등의 데이터 처리까지 금지한다는 뜻이 아니라 lock과 unlock의 단계 구분이다.

③ X: 2단계 로킹으로 스케줄의 충돌 직렬 가능성을 보장하려면 관련 트랜잭션들이 모두 규약과 로크 호환성 규칙을 지켜야 한다. 일부만 준수하면 다른 트랜잭션의 연산이 비직렬적인 충돌을 만들 수 있으므로 **항상 보장**할 수 없다.

④ X: 전용 로크는 같은 데이터 항목에 대한 다른 트랜잭션의 공유 로크나 전용 로크와 호환되지 않는다. 여러 트랜잭션이 읽기 목적으로 함께 보유할 수 있는 것은 **공유 로크(shared lock)**이다. 전용 로크는 배타적 접근을 위한 것이므로 사용 권한을 모두 공유한다는 설명은 반대이다.

NoteBase

문 17. 릴레이션의 열(column) A에 대해 이 열의 각 고유 값별로 그에 대응하는 개별 인덱스가 데이터베이스 내에 유지되는 인덱스를 사용한다고 할 때, 다음과 같은 방식으로 질의 처리를 지원하는 인덱스 구조로 가장 적절한 것은? (단, v는 A의 검색 조건으로 사용된 값을 의미함)

· 질의 조건 $A = v$의 경우, 값 v에 해당하는 인덱스는 이 조건을 만족하는 행들에 대한 레코드의 위치 값을 리턴한다.
· 질의 조건 $A = v$를 만족하는 행들의 개수를 구할 경우, 해당 인덱스 벡터에서 1인 엔트리의 개수를 구한다.

① 비트맵 인덱스

② 해시 인덱스

③ B-트리 인덱스

④ 함수 기반 인덱스

정답 1번

간단한 해설

고유 값별 비트 벡터를 두고 조건을 만족하는 행을 1로 표시하는 구조는 비트맵 인덱스이다. 1의 위치로 행을 찾고, 1의 개수를 세어 조건에 맞는 행 수를 구한다.

핵심 풀이

비트맵 인덱스는 열의 각 고유 값에 대해 **행별 해당 여부를 0과 1로 나타낸 비트 벡터**를 관리한다. 값 v의 벡터에서 해당 행의 값이 v이면 1, 아니면 0으로 표시한다. 따라서 $A = v$ 검색은 1인 비트에 대응하는 행의 위치를 찾는 방식으로 처리할 수 있다.

조건을 만족하는 행의 수는 해당 벡터에서 **1인 비트의 개수**와 같다. 해시값으로 버킷을 찾는 해시 인덱스, 정렬된 키를 트리로 탐색하는 B-트리 인덱스, 함수·표현식의 계산 결과를 키로 삼는 함수 기반 인덱스와 달리, 제시된 설명의 핵심은 고유 값별 비트 벡터와 1의 개수 세기이다.

NoteBase

문 18. 차수가 p인 B-트리 계열의 인덱스의 구조에 관한 설명에서 B^+-트리에 해당하는 내용만을 모두 고른 것은?

㉠ 각 내부 노드의 형태는 $\langle P_1, K_1, P_2, K_2, \ldots, P_{q-1}, K_{q-1}, P_q \rangle$이다. 여기서 $q \le p$이고, 각 P_i는 트리 포인터, 각 K_i는 키 값이다.
㉡ 각 리프 노드의 형태는 $\langle \langle K_1, Pr_1 \rangle, \langle K_2, Pr_2 \rangle, \ldots, \langle K_{q-1}, Pr_{q-1} \rangle, P_{next} \rangle$이다. 여기서 $q \le p$이고, 각 Pr_i는 키 값 K_i를 가지고 있는 레코드에 대한 포인터이고, P_{next}는 다음 리프 노드를 가리킨다.
㉢ 리프 노드에 키 값을 삽입할 때 오버플로가 일어나면 이 노드를 2개의 노드로 분할하여 각 노드의 키 값들을 절반씩 저장하고, 중간 키 값은 리프 노드에 남겨두지 않고 분할된 노드의 부모 노드로 올려보내 삽입한다.

① ㉡
② ㉠㉡
③ ㉠㉢
④ ㉡㉢

정답 2번

간단한 해설

B^+-트리는 내부 노드에 탐색용 키와 자식 포인터를, 리프에 키와 레코드 포인터 및 다음 리프 연결을 둔다. **리프 분할 시 구분 키는 부모에 복사하고 리프에도 유지**하므로 ㉢은 틀리다.

보기별 해설

㉠ O: 내부 노드는 **검색 경로를 정하는 키와 자식 노드를 가리키는 트리 포인터**로 구성된다. 자식 포인터가 q개이면 그 사이를 구분하는 키는 q−1개이며, 최대 자식 수가 p라는 표기에 맞는다.

㉡ O: 리프에는 **키와 해당 레코드에 대한 포인터**를 저장하고, 다음 리프를 가리키는 연결 포인터를 둔다. 이 연결을 따라 키 순서대로 이동할 수 있어 범위 검색에 활용된다.

㉢ X: B^+-트리의 **리프 분할에서는 부모에 구분 키를 복사**하며, 그 키와 레코드 포인터는 리프에도 남긴다. 리프에서 중간 키를 없애고 부모로만 옮긴다는 설명은 맞지 않는다. 내부 노드 분할에서 구분 키를 올리는 경우와 리프 분할을 구별해야 한다.

따라서 B^+-트리에 해당하는 것은 **㉠과** ㉡이므로 정답은 ②이다.

NoteBase

문 19. 두 릴레이션 R과 S에 대한 조인(join) 연산의 구현 방법 중에서 정렬-합병 조인(sort-merge join)에 관한 설명으로 가장 적절한 것은?

① R의 조인 속성 A와 S의 조인 속성 B에 대하여 동일한 해시함수 h를 사용하여 R과 S의 레코드들을 분할하고 버켓을 통해 일치하는 레코드를 찾아 결합한다.

② 외부 루프에서 R의 각 레코드에 대해 내부 루프에서 S로부터 모든 레코드를 검색하고 두 레코드가 조인 조건을 만족하는지 테스트한다.

③ R과 S의 레코드들이 각각 조인 속성 A와 B의 값에 따라 물리적으로 정렬되어 있으면 가장 효율적으로 적용할 수 있다.

④ 조인 속성 두 개 중 하나에 인덱스가 존재할 경우 내부 루프에서 인덱스를 사용하여 조건을 만족하는 레코드를 직접 검색하여 조인 효율을 향상시킨다.

정답 3번

간단한 해설

정렬-합병 조인은 **조인 키로 정렬된 두 입력을 비교하며 결합**한다. 이미 정렬된 입력을 이용하면 정렬 비용을 줄일 수 있다.

선지별 해설

① X: 같은 해시함수로 조인 키를 분할하고 대응 버킷에서 일치하는 레코드를 찾는 것은 **해시 조인**의 설명이다. 같은 해시값이라고 반드시 같은 키는 아니므로 실제 조인 조건도 확인한다.

② X: R의 각 레코드마다 S의 모든 레코드를 조사하는 것은 **단순 중첩 루프 조인**이다. 두 입력을 조인 키 순서로 진행하며 합병하는 방법과 다르다.

③ O: **정렬-합병 조인**은 두 입력을 조인 속성 순서로 정렬한 뒤 비교하며 일치하는 키의 레코드를 결합한다. 이미 정렬되어 있으면 추가 정렬 비용을 줄일 수 있어 유리하다. “가장 효율적으로”는 이런 유리한 입력 조건을 뜻하며, 데이터 크기·선택도 등에 관계없이 항상 다른 모든 조인보다 빠르다는 보장은 아니다. 인덱스 등으로 정렬된 입력을 얻을 수도 있다.

④ X: 바깥 레코드의 조인 키를 이용하여 안쪽 릴레이션의 인덱스로 일치 행을 찾는 것은 **인덱스 중첩 루프 조인**이다. 인덱스 탐색을 반복하는 점이 핵심이며 정렬된 두 입력을 합병하는 방식이 아니다.

따라서 정렬-합병 조인의 특징을 설명한 ③이 가장 적절하다.

NoteBase

문 20. 검색엔진을 이용한 순위 검색에서 어떤 검색 질의에 대해 데이터베이스에 존재하는 실제 관련된(relevant) 문서의 수가 10개, 이 검색 질의로 검색된(retrieved) 문서가 5개, 이 검색된 문서 중에서 관련된 문서가 3개라고 할 때, 이 검색 결과에 대한 정밀도(precision)와 재현율(recall)로 가장 적절한 것은?

① 정밀도: 30%, 재현율: 60%

② 정밀도: 50%, 재현율: 30%

③ 정밀도: 50%, 재현율: 60%

④ 정밀도: 60%, 재현율: 30%

정답 4번

간단한 해설

정밀도는 검색된 5개를, 재현율은 실제 관련 문서 10개를 분모로 한다. **검색된 관련 문서 3개를 각각 나누면 60%와 30%**이다.

계산 과정

관련 문서는 전체 10개이고, 검색된 5개 중 관련 문서는 3개이다. 따라서 관련 문서의 검색 성공 수는 3, 검색 결과 중 비관련 문서는 5-3=2, 놓친 관련 문서는 10-3=7이다. 두 지표의 분자는 모두 **검색된 관련 문서 3개**이다.

정밀도는 검색된 문서 중 실제 관련된 문서의 비율이다. 분모는 검색 결과 전체 5개이므로 **정밀도=** $\frac{3}{5} \times 100 = 60\%$이다.

재현율은 전체 관련 문서 중 검색에 성공한 문서의 비율이다. 분모는 실제 관련 문서 전체 10개이므로 **재현율=** $\frac{3}{10} \times 100 = 30\%$이다.

따라서 정밀도60%·재현율30%인 ④가 적절하다.

NoteBase