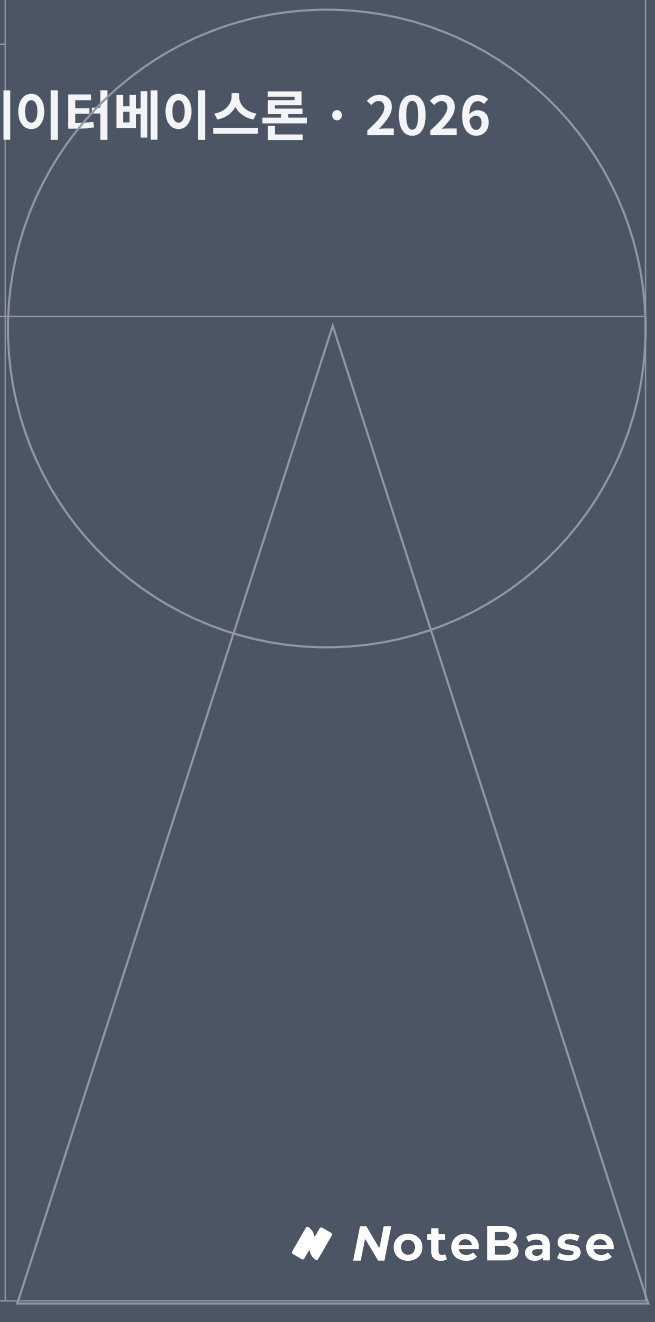
기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2026



NoteBase

기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2026

공무원/경찰/공채(경위) · 데이터베이스론 · 2026

문 1. DBMS의 장점을 모두 고른 것은?

가. 응용프로그램 개발 생산성이 높다.
나. 다양한 유형의 고장으로부터 데이터베이스를 회복할 수 있다.
다. 무결성이 향상된다.
라. 다수 사용자가 접근하므로 보안이 취약하다.
마. 다수 사용자를 위한 동시성 제어가 제공되지 않는다.

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

정답 4번

간단한 해설

DBMS는 다중 사용자 환경에서도 **접근 제어와 동시성 제어를 제공한다.** 따라서 여러 사용자가 접근한다는 이유로 보안이 취약하다거나 동시성 제어가 없다고 볼 수 없다.

NoteBase

보기별 해설

가. O: DBMS가 데이터의 저장·검색·갱신 기능을 제공하므로 응용프로그램마다 이를 직접 구현할 필요가 줄어 개발 생산성이 높아진다.

나. O: 로그와 백업 등을 이용한 회복 기능으로 트랜잭션 실패나 시스템·저장장치 장애가 발생했을 때 데이터베이스를 일관된 상태로 복구할 수 있다.

다. O: 기본키·외래키 등 무결성 제약조건을 검사하여 허용되지 않는 데이터 입력과 변경을 막으므로 데이터의 정확성과 일관성이 향상된다.

라. X: DBMS는 사용자 인증과 권한별 접근 제어를 통해 허가되지 않은 접근을 제한한다. 다수 사용자의 접근 자체가 보안 취약성을 뜻하는 것은 아니며, **보안을 통합적으로 관리**할 수 있다.

마. X: DBMS는 여러 트랜잭션이 동시에 실행될 때 갱신 손실 등의 문제를 막기 위한 **동시성 제어를 제공한다**.

따라서 DBMS의 장점에 해당하는 보기는 가, 나, 다이며, 이를 고른 선택지는 ④이다.

NoteBase

문 2. 데이터베이스 3단계 구조에서 데이터 독립성에 대한 설명으로 가장 적절한 것은?

외부 단계와 개념 단계 사이의 독립성으로 개념 스키마가 변경되어도 외부 스키마에는 영향을 미치지 않도록 지원한다.

① 물리적 데이터 독립성
② 논리적 데이터 독립성
③ 외부적 데이터 독립성
④ 개념적 데이터 독립성

정답 2번

간단한 해설

개념 스키마가 변경되어도 외부 스키마를 유지할 수 있게 하는 성질이 논리적 데이터 독립성이다. 물리적 데이터 독립성은 내부 저장 방식의 변경이 개념 스키마에 영향을 주지 않도록 하는 성질이다.

공통 해설

데이터 독립성은 하위 단계의 스키마를 변경해도 상위 단계의 스키마에 영향을 주지 않도록 하는 성질이다. 개념 스키마는 데이터베이스 전체의 논리적 구조를, 외부 스키마는 개별 사용자나 응용 프로그램이 보는 데이터 구조를 나타낸다. 이 둘 사이에서 **개념 스키마의 변경으로부터 외부 스키마를 독립**시키는 것이 논리적 데이터 독립성이므로 지문의 설명에 해당한다.

예를 들어 학생 정보에 새 속성을 추가하더라도, 학번과 이름만 조회하던 사용자의 외부 스키마는 그대로 유지할 수 있다. 개념 스키마와 외부 스키마 사이의 대응 관계를 조정하여 전체 논리 구조의 변경이 기존 사용자 관점에 영향을 주지 않게 하는 것이다.

같은 구분 기준에서 물리적 데이터 독립성은 **내부 스키마와 개념 스키마 사이**의 독립성이다. 파일의 저장 구조나 접근 경로를 바꾸어도 개념 스키마를 유지하는 경우가 이에 해당한다. 데이터 독립성의 표준적인 두 구분은 논리적·물리적 데이터 독립성이며, '외부적 데이터 독립성'과 '개념적 데이터 독립성'은 지문이 설명하는 성질의 명칭이 아니다.

NoteBase

문 3. 사원(사원번호, 이름, 상사사원번호) 테이블에서 사원번호, 이름, 상사사원이름을 검색하는 질의문의 빈칸에 가장 적절한 것은? (단, 사원번호는 기본키이며 상사사원번호는 상사의 사원번호를 참조하기 위한 외래키이다.)

SELECT A.사원번호, (㉠), (㉡)
FROM 사원 A, 사원 B
WHERE (㉢) = (㉣)

① (㉠): A.이름, (㉡): B.상사사원이름, (㉢): A.상사사원번호, (㉣): B.사원번호

② (㉠): A.이름, (㉡): B.상사사원이름, (㉢): A.사원번호, (㉣): B.상사사원번호

③ (㉠): A.이름, (㉡): B.이름, (㉢): A.사원번호, (㉣): B.사원번호

④ (㉠): A.이름, (㉡): B.이름, (㉢): A.상사사원번호, (㉣): B.사원번호

정답 4번

간단한 해설

사원 본인 A의 **상사사원번호**를 상사 B의 **사원번호**와 연결하는 셀프 조인이다. 본인의 이름은 A.이름, 상사의 이름은 B.이름에서 가져온다.

빈칸별 해설

같은 사원 테이블에 A와 B라는 별칭을 붙이는 '셀프 조인'이다. A는 조회할 사원, B는 그 사원의 상사 역할이며, SELECT의 A.사원번호와 함께 사원 본인의 이름과 상사의 이름을 가져온다.

㉠ A.이름: A가 나타내는 사원 본인의 이름을 조회하므로 A의 이름 속성을 사용한다.

㉡ B.이름: 조인으로 연결된 B가 상사의 행이므로 **B의 이름**이 상사사원이름이 된다. 테이블에 '상사사원이름'이라는 속성은 없으므로 B.상사사원이름을 사용할 수 없다.

㉢ A.상사사원번호: A의 상사를 찾는 값은 A 자신의 사원번호가 아니라 **상사의 사원번호를 저장한 외래키**이다. 이 값을 B의 사원번호와 비교해야 한다.

㉣ B.사원번호: A가 참조하는 상사의 행을 식별하려면 B의 기본키를 사용한다. 따라서 조인 조건은 A.상사사원번호 = B.사원번호이다. A.사원번호 = B.상사사원번호로 연결하면 B는 A를 상사로 둔 부하 사원이 되고, A.사원번호 = B.사원번호로 연결하면 같은 사원끼리 연결되어 상사의 이름을 얻지 못한다.

따라서 ㉠ A.이름, ㉡ B.이름, ㉢ A.상사사원번호, ㉣ B.사원번호의 조합이므로 ④와 일치한다.

NoteBase

문 4. 관계 데이터 모델에서 릴레이션의 특성으로 적절하게 설명한 것을 모두 고른 것은?

가. 릴레이션에는 중복된 튜플이 존재할 수 있다.
나. 릴레이션에 있는 튜플들의 순서는 의미가 없다.
다. 릴레이션의 속성 사이의 순서는 정해져 있다.
라. 모든 속성은 원자값을 갖는다.

① 나, 다
② 가, 라
③ 나, 라
④ 가, 다

정답 3번

간단한 해설

릴레이션은 튜플의 집합이므로 **튜플의 나열 순서는 의미가 없고**, 각 속성값은 **하나의 원자값**이어야 한다.

보기별 해설

가. X: 릴레이션은 튜플의 집합이므로 **동일한 튜플이 중복되어 존재할 수 없다**. 모든 속성값이 같은 두 행은 관계 데이터 모델에서 서로 다른 튜플로 취급하지 않는다.

나. O: 집합의 원소에 순서가 없듯이 릴레이션의 **튜플 사이에는 순서가 없다**. 튜플을 나열한 순서가 바뀌어도 포함된 튜플이 같으면 동일한 릴레이션이다.

다. X: 속성은 위치가 아니라 고유한 이름으로 구별하므로 **속성 사이의 순서는 의미가 없다**. 속성명과 그에 대응하는 값을 함께 유지한 채 열의 표시 순서를 바꾸어도 동일한 릴레이션이다.

라. O: 관계 데이터 모델에서 각 튜플의 속성값은 더 이상 나누지 않는 **하나의 원자값**이어야 한다. 따라서 한 속성 자리에 여러 값의 집합이나 반복되는 값의 목록을 넣지 않는다.

적절한 설명은 나, 라이며, 이를 모두 고른 선택지는 ③이다.

NoteBase

문 5. 관계 데이터 모델에서 참조 무결성 제약조건이 위배될 가능성이 가장 낮은 것은?

① 참조되는 테이블에 새로운 튜플이 삽입될 때

② 참조하는 테이블에 새로운 튜플이 삽입될 때

③ 참조되는 테이블에 있는 튜플을 삭제할 때

④ 참조하는 테이블에 있는 튜플을 변경할 때

정답 1번

간단한 해설

참조되는 테이블에 새 튜플을 삽입하면 **참조 가능한 키가 추가될 뿐**, 기존 외래 키의 참조 대상은 사라지지 않는다. 따라서 이 삽입은 참조 무결성을 위배할 가능성이 가장 낮다.

선지별 해설

참조 무결성은 참조하는 테이블의 외래 키 값이 참조되는 테이블의 키 값과 대응하도록 하는 제약이다. 외래 키에 NULL이 허용되는 경우에는 그 조건을 따른다. 각 연산이 참조 대상을 없애거나 존재하지 않는 참조 값을 만드는지를 비교한다.

① 참조되는 테이블에 새 튜플을 삽입하면 참조할 수 있는 키 값이 추가된다. 기존에 유효하던 외래 키가 참조 대상을 잃지 않으므로, 이 삽입 자체는 **참조 무결성을 위배하지 않는다**. 중복된 기본 키를 삽입하는 문제는 키의 유일성 등 다른 제약조건과 구별한다.

② 참조하는 테이블에 새 튜플을 삽입할 때 외래 키에 참조 대상 테이블에 없는 값을 넣으면 위반이 발생할 수 있다. 따라서 **새 외래 키 값에 대응하는 참조 키가 존재하는지** 확인해야 한다. NULL을 허용하는 외래 키라면 허용된 NULL은 별도로 처리한다.

③ 참조되는 테이블의 튜플을 삭제하면 다른 테이블의 외래 키가 그 튜플을 가리키고 있을 수 있다. 별도 조치 없이 삭제하면 **기존 외래 키의 참조 대상이 사라진다**. 데이터베이스는 삭제 거부나 연쇄 삭제 등 설정된 참조 동작으로 무결성을 유지한다.

④ 참조하는 테이블의 튜플을 변경할 때 외래 키 값을 존재하지 않는 참조 키 값으로 바꾸면 위반이 발생할 수 있다. 외래 키와 무관한 속성의 변경은 보통 이 제약에 영향을 주지 않지만, **외래 키를 변경하는 경우**가 있으므로 위반 가능성이 있다.

참조되는 테이블에 새 튜플을 추가하는 연산은 참조 대상을 없애지도, 참조하는 쪽에 잘못된 외래 키를 만들지도 않으므로 위반 가능성이 가장 낮다. 따라서 요구에 해당하는 선택지는 ①이다.

NoteBase

문 6. 수강신청과 관련된 3개의 테이블에서 수강 테이블에 (25001, 102, NULL) 튜플이 삽입될 때 무결성 제약조건을 확인해야 할 내용으로 가장 적절하지 않은 것은? (단, 각 테이블에서 밑줄 친 속성은 기본키이고 수강 테이블에서 학번과 과목번호는 학생 테이블과 과목 테이블을 참조하기 위한 외래키이다.)

<학생>

학번	이름
25001	김기철
24099	임기영
23081	이찬수

<수강>

학번	과목번호	성적
25001	101	91
24099	104	85
25001	102	90
23081	102	88

<과목>

과목번호	과목명
101	인공지능
102	운영체제
103	자료구조
104	네트워크

① 학번 25001에 대한 참조 무결성 제약조건 확인

② 과목번호 102에 대한 참조 무결성 제약조건 확인

③ 삽입될 튜플의 각 속성 값에 대한 도메인 제약조건 확인

④ 성적 NULL에 대한 개체 무결성 제약조건 확인

정답 4번

간단한 해설

개체 무결성은 **기본키의 NULL을 금지**하는 제약이다. 성적은 기본키가 아니므로 성적 값이 NULL이라는 사실을 개체 무결성 위반으로 판단하지 않는다.

NoteBase

선지별 해설

수강 테이블의 기본키는 학번과 과목번호의 조합이며, 두 속성은 각각 학생과 과목 테이블을 참조하는 외래키이기도 하다. 같은 속성이 기본키와 외래키의 역할을 함께 할 수 있다.

① O: 학번은 학생 테이블의 학번을 참조하므로, 삽입하는 25001이 참조 대상에 존재하는지 확인해야 한다. 학생 테이블에 25001이 있으므로 이 값은 **참조 무결성을 충족**한다.

② O: 과목번호는 과목 테이블의 과목번호를 참조하므로, 삽입하는 102가 과목 테이블에 존재하는지 확인해야 한다. 과목 테이블에 운영체제의 과목번호 102가 있으므로 이 값도 참조 무결성을 충족한다.

③ O: 각 속성 값은 그 속성에 지정된 자료형과 허용 범위 등 **도메인 제약조건**에 맞아야 한다. 학번·과목번호·성적에 각각 적용되는 제약을 확인해야 하며, 성적의 NULL 허용 여부도 별도로 지정된 조건을 살펴야 한다.

④ X: 개체 무결성에서 NULL이 허용되지 않는 대상은 기본키를 구성하는 학번과 과목번호이다. **기본키가 아닌 성적**의 NULL을 개체 무결성 검사 대상으로 연결한 것이 부적절하다. 성적에 별도의 NOT NULL 제약이 있으면 그 제약을 확인해야 한다. 또한 기존 수강 테이블에 (25001, 102, 90)이 있으므로 새 튜플은 학번·과목번호의 조합이 중복되어 기본키의 유일성 제약을 위반한다.

NoteBase

문 7. 다음 키에 대한 설명으로 가장 적절한 것은?

기본키(primary key)가 보안을 필요로 하거나, 다수의 속성으로 구성되어 복잡하거나, 적절한 기본키가 없을 때 일련번호 같은 가상의 속성을 만들어 기본키로 설정하는 경우가 있다.

① 외래키(foreign key)
② 슈퍼키(super key)
③ 대체키(alternate key)
④ 대리키(surrogate key)

정답 4번

간단한 해설

기존 속성 대신 일련번호 같은 **인공적인 속성을 새로 만들어** 기본키로 사용하는 것이 **대리키**이다. 기존 식별값의 노출을 피하거나 복잡한 복합키를 간단하게 대체할 때 사용할 수 있다.

공통 해설

대리키(surrogate key)는 업무상 의미가 있는 기존 속성 대신, 레코드를 식별하기 위해 인위적으로 도입한 키이다. 지문에서는 단순히 레코드를 유일하게 구별할 수 있다는 성질보다 **일련번호 같은 가상의 속성을 새로 만든다**는 점이 핵심이다. 이처럼 키를 만드는 방식을 구체적으로 가리키는 명칭이 대리키이다.

기존 기본키가 민감한 정보를 담고 있으면 별도의 식별값으로 대신할 수 있고, 여러 속성을 묶은 복합키가 복잡하면 하나의 인공 식별값으로 간단히 만들 수 있다. 적절한 기존 식별자가 없는 경우에도 사용할 수 있다. 예를 들어 고객 테이블에 식별용 고객번호를 새로 추가하고 1001, 1002처럼 서로 다른 번호를 부여하여 기본키로 삼으면, 이 고객번호가 대리키이다.

NoteBase

문 8. 데이터 정의어(DDL)로 적절하지 않은 것을 모두 고른 것은?

가. MERGE
나. ALTER
다. DROP
라. CREATE
마. DELETE

① 가, 마
② 마
③ 가, 다
④ 나, 다, 라

정답 1번

간단한 해설

데이터 정의어(DDL)는 테이블 등 데이터베이스 객체의 **구조를 정의·변경·삭제**한다. MERGE와 DELETE는 구조를 정의하는 명령이 아니라 **저장된 데이터를 조작하는 DML**이다.

보기별 해설

가. MERGE: 데이터 조작어(DML)이다. 대상 테이블과 원본 데이터의 일치 여부에 따라 행을 삽입하거나 갱신하는 작업 등을 수행한다. **기존 테이블의 데이터를 조작**하므로 DDL에 해당하지 않는다.

나. ALTER: 데이터 정의어(DDL)이다. 테이블에 열을 추가하거나 열의 정의·제약조건을 바꾸는 등 **기존 데이터베이스 객체의 구조를 변경**한다.

다. DROP: 데이터 정의어(DDL)이다. 테이블·뷰 등 **데이터베이스 객체 자체를 제거**한다. 테이블의 행만 삭제하고 구조를 남기는 DELETE와 구별된다.

라. CREATE: 데이터 정의어(DDL)이다. 테이블·뷰 등 **새 데이터베이스 객체를 생성**하고 그 구조를 정의한다.

마. DELETE: 데이터 조작어(DML)이다. 조건에 맞는 행을 삭제하며, 조건을 지정하지 않으면 모든 행을 삭제할 수 있다. 행을 삭제하더라도 **테이블의 구조는 남는다**는 점에서 DROP과 다르다.

따라서 DDL에 해당하지 않는 것은 가와 마이며, 이 조합은 ①에 해당한다.

NoteBase

문 9. 상품 릴레이션과 판매 릴레이션을 통해 판매가 된 상품 정보를 검색하는 세미 조인(semi join) 결과 릴레이션에 대한 차수(degree)와 카디널리티(cardinality)의 합으로 가장 적절한 것은? (단, 각 릴레이션에서 밑줄 친 속성은 기본키이며 판매 릴레이션에서 상품번호는 상품 릴레이션을 참조하기 위한 외래키이다.)

<상품>

상품번호	상품명	가격
1	A1	1000
2	A2	1500
3	A3	1200
4	A4	1300

<판매>

판매번호	상품번호	주문일자	주문수량
1	2	2025-07-01	100
2	1	2025-07-03	150
3	2	2025-07-07	120
4	1	2025-07-10	130
5	4	2025-07-11	200
6	2	2025-07-13	180
7	4	2025-07-15	115

① 6

② 7

③ 10

④ 14

NoteBase

정답 1번

간단한 해설

세미 조인은 판매 내역이 있는 **상품의 속성만** 남긴다. 판매된 상품번호는 1, 2, 4이므로 차수는 3, 카디널리티는 3이며 합은 6이다.

계산 과정

세미 조인의 대상: 판매가 된 상품 정보를 구하므로 상품 릴레이션에서 판매 릴레이션의 상품번호와 일치하는 튜플을 선택한다. 세미 조인은 상대 릴레이션에 일치하는 튜플이 있는지를 확인하면서 **상품 릴레이션의 속성만 유지**한다.

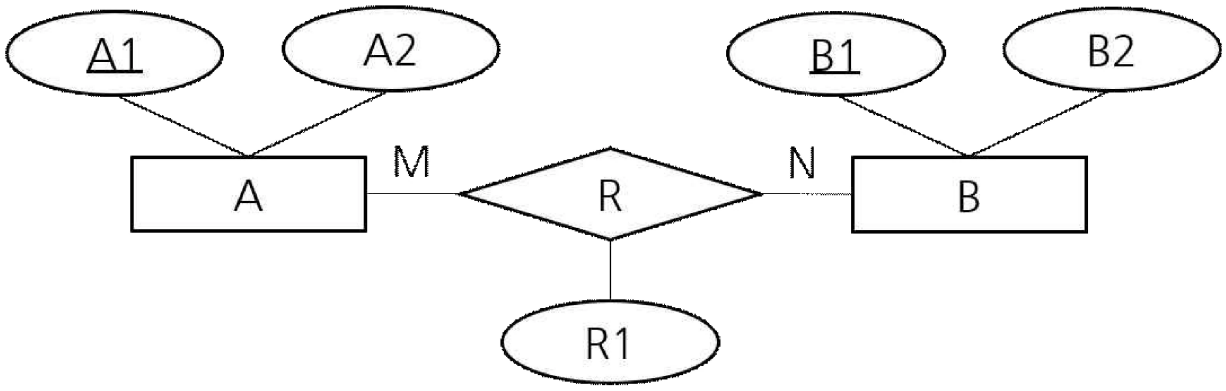
차수: 결과의 속성은 상품번호, 상품명, 가격이다. 차수는 속성의 개수이므로 3이다. 판매번호, 주문일자, 주문수량은 결과 속성에 추가하지 않는다.

카디널리티: 판매 릴레이션의 상품번호는 2, 1, 2, 1, 4, 2, 4이다. 이 중 서로 다른 상품번호는 1, 2, 4이므로 결과에는 (1, A1, 1000), (2, A2, 1500), (4, A4, 1300)의 3개 튜플이 남는다. **같은 상품이 여러 번 판매되어도** 해당 상품 튜플을 판매 횟수만큼 반복하지 않으며, 판매 내역이 없는 상품번호 3은 제외한다.

합계: 차수와 카디널리티의 합은 3 + 3 = 6이다. 따라서 6을 제시한 ①과 일치한다.

NoteBase

문 10. E-R 다이어그램을 관계 데이터 모델에 따라 릴레이션 스키마로 변환한 것으로 가장 적절한 것은? (단, 밑줄 친 속성은 기본키이다.)



① A(A1, A2), B(B1, B2), R(A1, B1, R1)

② A(A1, A2, R1), B(B1, B2)

③ A(A1, A2), B(B1, B2, R1)

④ A(A1, B1, A2, B2, R1)

정답 1번

간단한 해설

A와 B는 **다대다(M:N) 관계**이므로 관계 R을 별도의 릴레이션으로 만든다. R에는 두 개체의 기본키 A1·B1과 관계 속성 R1을 넣고, **A1과 B1의 조합**을 기본키로 삼는다.

NoteBase

도식 해설

그림에서 사각형 A·B는 개체, 마름모 R은 관계, 타원은 속성을 나타낸다. A와 R 사이의 M, R과 B 사이의 N은 하나의 A가 여러 B와 연결되고 하나의 B도 여러 A와 연결될 수 있는 다대다 관계를 뜻한다.

개체 A와 B는 각각 A(A1, A2), B(B1, B2)로 변환한다. 그림에서 밑줄 친 A1은 A의 기본키이고 B1은 B의 기본키이다. A2와 B2는 각 개체의 일반 속성이다.

다대다 관계 R은 **독립된 관계 릴레이션** R(A1, B1, R1)로 변환한다. A1은 A를, B1은 B를 참조하는 외래키이며, 두 속성을 묶은 (A1, B1)이 기본키가 된다. A1만으로는 여러 B와의 연결을 구별할 수 없고 B1만으로도 여러 A와의 연결을 구별할 수 있기 때문이다. R1은 개체 한쪽의 속성이 아니라 **A와 B의 연결에 붙는 속성**이므로 R에 저장한다.

②와 ③은 R1을 각각 A 또는 B에 넣고 그 개체의 기본키 일부로 삼았다. 이 구조에는 상대 개체의 키가 없어 어떤 A와 어떤 B가 연결되는지 저장할 수 없으며, 관계 속성을 한쪽 개체의 속성으로 잘못 옮긴다.

④는 개체 속성과 관계 속성을 하나의 릴레이션에 합쳤다. 다대다 연결마다 A2·B2가 반복 저장되어 개체 정보의 중복과 갱신 문제가 생기므로, 각 개체 릴레이션과 관계 릴레이션을 분리하는 표준 변환에 맞지 않는다.

따라서 기본키가 각각 A1, B1, (A1, B1)인 A(A1, A2), B(B1, B2), R(A1, B1, R1)을 제시한 ①이 해당한다.

NoteBase

문 11. 릴레이션 R1과 R2를 이용하여 R3를 결과로 생성하는 관계 대수 연산으로 가장 적절한 것은?

<R1>

갑	을
101	A1
102	A2
103	A3
104	A4
105	A2

<R2>

을	병
A2	100
A3	120
A4	200
A5	180

<R3>

갑	을	병
101	A1	NULL
102	A2	100
103	A3	120
104	A4	200
105	A2	100
NULL	A5	180

① INNER JOIN
② RIGHT OUTER JOIN
③ LEFT OUTER JOIN
④ FULL OUTER JOIN

정답 4번

간단한 해설

R3에는 공통 속성 '을'이 일치하는 행뿐 아니라 **R1에만 있는 A1과 R2에만 있는 A5의 행도 모두** 남아 있다.
양쪽의 일치하지 않는 행을 보존하고 상대 릴레이션에 없는 속성을 NULL로 채우는 연산은 완전 외부 조인이다.

NoteBase

도식 해설

공통 속성 '을'의 값이 같은 행끼리 연결한다. R1의 속성은 '갑, 을', R2의 속성은 '을, 병'이며, R3는 '갑, 을, 병'으로 결합된 결과이다.

일치하는 행: A2는 R1에 (102, A2), (105, A2) 두 행이 있고 R2에는 (A2, 100)이 있으므로 (102, A2, 100), (105, A2, 100)이 각각 생성된다. A3와 A4도 연결되어 (103, A3, 120), (104, A4, 200)이 된다. 공통 속성값이 같더라도 '갑'이 다르므로 두 A2 행을 하나로 합치지 않는다.

R1에만 있는 행: (101, A1)은 R2에 대응하는 A1이 없지만 결과에 남아 있다. R2에서 가져올 '병'의 값이 없으므로 **(101, A1, NULL)**이 된다.

R2에만 있는 행: (A5, 180)은 R1에 대응하는 A5가 없지만 결과에 남아 있다. R1에서 가져올 '갑'의 값이 없으므로 **(NULL, A5, 180)**이 된다.

INNER JOIN은 일치하는 네 행만 남긴다. R1을 왼쪽 릴레이션으로 보았을 때 LEFT OUTER JOIN은 A1의 행까지 남기지만 A5의 행은 포함하지 않고, RIGHT OUTER JOIN은 A5의 행까지 남기지만 A1의 행은 포함하지 않는다.

R3는 양쪽의 불일치 행을 모두 포함한 여섯 행이므로 **FULL OUTER JOIN**에 해당하며, 선택지 ④와 일치한다.

NoteBase

문 12. 학생(학번, 학과, 신장) 테이블에서 평균 신장이 175 이상인 학과를 검색하는 질의문의 빈칸에 가장 적절한 것은?

SELECT 학과
FROM 학생
(㉠) 학과 (㉡)
(SELECT 학과
FROM 학생
(㉢) 학과 HAVING (㉣) >= 175);

① ㉠ WHERE ㉡ IN ㉢ GROUP BY ㉣ AVG(신장)

② ㉠ WHERE ㉡ NOT IN ㉢ GROUP BY ㉣ 신장

③ ㉠ HAVING ㉡ NOT IN ㉢ ORDER BY ㉣ AVG(신장)

④ ㉠ HAVING ㉡ IN ㉢ ORDER BY ㉣ 신장

정답 1번

간단한 해설

학과별 평균 신장을 비교하려면 **GROUP BY로 학과별 그룹을 만들고 HAVING에서 AVG(신장)을 검사**한다.
외부 질의는 WHERE와 IN으로 그 결과에 포함된 학과를 선택한다.

NoteBase

빈칸별 해설

안쪽 질의가 평균 신장 조건을 만족하는 학과의 집합을 구하고, 바깥쪽 질의가 그 집합에 속하는 학과를 검색한다. 네 빈칸은 이 두 질의의 역할을 기준으로 채운다.

㉠ WHERE: 바깥쪽 질의의 학생 행마다 학과가 안쪽 질의 결과에 포함되는지를 검사하므로 **행을 선택하는 WHERE**가 필요하다. HAVING은 그룹에 대한 조건을 지정하는 절이므로 이 자리를 대신하지 않는다.

㉡ IN: 학과가 안쪽 질의에서 구한 집합에 **포함되어야** 하므로 IN을 사용한다. NOT IN을 쓰면 조건을 만족하는 학과가 아니라 그 집합 밖의 학과를 고르게 된다.

㉢ GROUP BY: 학과별로 평균을 계산하려면 같은 학과의 학생들을 하나의 그룹으로 묶어야 하므로 GROUP BY 학과가 필요하다. ORDER BY는 결과의 정렬 순서를 지정하며 그룹을 만들지 않는다.

㉣ AVG(신장): 조건은 개별 학생의 신장이 아니라 **학과별 평균 신장이 175 이상**인지이다. 따라서 HAVING AVG(신장) >= 175로 각 그룹의 평균을 검사하며, 신장만 적으면 요구한 평균 조건을 표현하지 못한다.

따라서 ㉠ WHERE, ㉡ IN, ㉢ GROUP BY, ㉣ AVG(신장)의 조합인 ①과 일치한다.

NoteBase

문 13. 테이블을 생성할 때 참조되는 테이블의 튜플이 삭제되어도 참조 무결성 제약조건을 유지하기 위한 설정 방법으로 가장 적절하지 않은 것은?

① ON DELETE NO ACTION

② ON DELETE CASCADE

③ ON DELETE SET NULL

④ ON DELETE SET 0

정답 4번

간단한 해설

ON DELETE SET 0은 유효한 참조 동작이 아니다. 참조되는 행이 삭제될 때 외래키를 NULL로 바꾸려면 SET NULL을 사용하며, 0은 참조 관계를 없애는 특별한 값이 아니라 일반 데이터 값이다.

선지별 해설

참조 무결성은 외래키가 참조 대상의 유효한 키 값을 가리키도록 하는 제약이다. 부모 테이블의 행을 삭제할 때 자식 테이블에 존재하지 않는 부모를 가리키는 외래키가 남지 않도록 삭제 동작을 지정한다.

① O: ON DELETE NO ACTION은 자식 행을 자동으로 삭제하거나 외래키를 바꾸지 않는다. 제약조건을 검사할 때 삭제된 부모를 가리키는 자식 행이 남아 있으면 **참조 무결성 위반으로 삭제가 허용되지 않는다**. 무결성 검사를 생략한다는 뜻이 아니다.

② O: ON DELETE CASCADE는 부모 행을 삭제할 때 그 행을 참조하는 **자식 행도 함께 삭제**한다. 참조 대상이 사라진 외래키가 남지 않으므로 무결성을 유지한다.

③ O: ON DELETE SET NULL은 부모 행이 삭제되면 이를 참조하던 **자식 행의 외래키를 NULL로 변경**한다. 자식 행 자체는 남으며, 해당 외래키 열은 NULL을 허용해야 한다.

④ X: ON DELETE SET 0은 정의된 참조 동작이 아니다. NULL로 바꾸는 SET NULL이나 미리 정한 기본값으로 바꾸는 SET DEFAULT와 구별해야 한다. 기본값을 0으로 정하더라도 SET DEFAULT를 사용해야 하며, 변경된 값도 참조 무결성을 만족해야 한다. **0으로 바꾸는 것만으로 참조 무결성이 보장되지는 않는다**.

NoteBase

문 14. 관계 데이터 모델에서 정규형의 효용성에 관한 내용으로 가장 적절하지 않은 것은?

① 하나의 릴레이션 내에서 속성들 간의 관계를 표현한다.

② 불필요한 데이터의 종속과 중복을 제거한다.

③ 새로운 속성 추가 시 기존 속성과의 관계 수정을 최소화한다.

④ 질의 처리 시 릴레이션의 셀렉션 효율성을 최대화한다.

정답 4번

간단한 해설

정규화의 목적은 **데이터 중복과 갱신 이상을 줄이는 것**이다. 정규형을 만족한다고 해서 **셀렉션 효율성이 최대화되는 것은 아니며**, 질의 성능은 인덱스·파일 구성·실행 계획 등의 영향을 받는다.

선지별 해설

정규형은 속성 사이의 함수 종속 등 데이터의 의미적 제약을 기준으로 릴레이션이 갖추어야 할 조건이다.
정규화는 이 조건을 만족하도록 릴레이션을 분해하여 중복과 삽입·삭제·갱신 이상을 줄이는 과정이다.

① O: 정규형을 판단하려면 한 릴레이션 안에서 어떤 속성이 다른 속성의 값을 결정하는지 분석한다. 이처럼 **속성 사이의 종속 관계를 명확히 파악**하고 그 관계에 맞는 구조를 설계할 수 있다.

② O: 서로 다른 사실을 한 릴레이션에 섞으면 같은 값이 여러 튜플에 반복되고 수정할 때 일부만 바뀌는 문제가 생길 수 있다. 정규화는 부분 함수 종속·이행 함수 종속 등 정규형에 어긋나는 종속 구조를 분리하여 불필요한 중복과 이상 현상을 줄인다. 필요한 종속성 자체를 모두 없앤다는 뜻은 아니다.

③ O: 속성을 의미와 종속 관계에 맞게 나누어 두면 새 속성을 그 속성이 속하는 적절한 릴레이션에 추가할 수 있다. 따라서 **관련 없는 기존 속성들의 관계까지 변경할 필요를 줄여** 스키마 확장과 유지보수를 쉽게 한다. 새 속성의 종속 관계에 따라 구조 검토는 여전히 필요하다.

④ X: 셀렉션은 조건에 맞는 튜플을 선택하는 연산이다. 그 효율은 인덱스 유무, 조건의 선택도, 저장 방식, 질의 최적화 등에 좌우되며, **정규화만으로 최대 성능이 보장되지 않는다**. 정규화로 릴레이션이 분해되면 질의에 필요한 조인이 늘어날 수도 있다. 따라서 정규형의 효용성을 셀렉션 효율성의 최대화로 설명한 ④가 적절하지 않다.

NoteBase

문 15. 함수적 종속성 추론 규칙 중 의사 이행성 규칙 도출에 사용된 암스트롱 추론 규칙의 적용 내용으로 가장 적절한 것은?

$X \rightarrow Y$이고 $WY \rightarrow Z$이면 $WX \rightarrow Z$

① 재귀성 규칙과 부가성 규칙을 이용하여 도출이 가능하다.

② 재귀성 규칙만으로도 도출이 가능하다.

③ 부가성 규칙과 이행성 규칙을 이용하여 도출이 가능하다.

④ 이행성 규칙과 재귀성 규칙을 이용하여 도출이 가능하다.

정답 3번

간단한 해설

먼저 **부가성**으로 $X \rightarrow Y$의 양쪽에 W를 더하여 $WX \rightarrow WY$를 얻는다. 이를 주어진 $WY \rightarrow Z$와 **이행성**으로 연결하면 $WX \rightarrow Z$가 도출된다.

핵심 풀이

부가성 적용: 부가성 규칙은 함수적 종속성의 양쪽에 같은 속성 집합을 추가해도 종속성이 성립한다는 규칙이다. $X \rightarrow Y$에 W를 추가하면 $WX \rightarrow WY$가 된다. 여기서 WX와 WY는 각각 속성 집합의 합집합을 나타내며, 곱셈이 아니다.

이행성 적용: 이행성 규칙은 $A \rightarrow B$와 $B \rightarrow C$로부터 $A \rightarrow C$를 도출한다. 앞에서 얻은 $WX \rightarrow WY$와 주어진 $WY \rightarrow Z$는 **중간 속성 집합 WY가 같으므로** 연결할 수 있다. 따라서 $WX \rightarrow Z$가 성립한다.

재귀성과의 구분: 재귀성 규칙은 어떤 속성 집합이 그 부분집합을 함수적으로 결정한다는 규칙이다.
재귀성만으로는 주어진 두 종속성을 연결하는 위 과정을 수행할 수 없다. 재귀성과 부가성만 제시한 조합은 종속성을 연결하는 이행성이 빠져 있고, 이행성과 재귀성만 제시한 조합은 양쪽에 W를 더해 중간 집합을 맞추는 부가성이 빠져 있다.

따라서 의사 이행성 규칙은 **부가성 규칙과 이행성 규칙**을 적용하여 도출하며, 이는 ③의 내용과 일치한다.

NoteBase

문 16. 기본키가 {도크번호, 입항}인 선박관리 테이블이 만족하는 정규형으로 가장 적절한 것은?

<선박관리>

도크번호	입항	출항	목적	도크관리자	담당도선사
D1	11:00	11:45	선적	한경사	이순신
D2	11:50	12:45	하역	김경위	김구진
D1	12:00	12:45	주유	한경사	정약용
D2	13:00	15:00	정비	김경위	윤봉길

① 제1정규형

② 제2정규형

③ 제3정규형

④ BCNF

정답 1번

간단한 해설

각 칸에는 하나의 값이 들어 있지만, 도크관리자는 복합 기본키 전체가 아니라 **도크번호에만 종속**된다. 따라서 부분 함수 종속이 남아 있어 제2정규형을 만족하지 못하고 제1정규형에 해당한다.

NoteBase

핵심 풀이

제1정규형은 각 속성값이 원자값인 관계이다. 표의 도크번호·입항·출항·목적·도크관리자·담당도선사는 각 칸에 하나의 값을 가지므로 제1정규형 조건을 만족한다.

기본키는 {도크번호, 입항}이다. 같은 도크번호가 여러 번 나타나더라도 입항 시각을 함께 사용하여 개별 입항 기록을 구별한다. 제2정규형이 되려면 제1정규형을 만족하면서, 키가 아닌 속성이 이 복합키의 일부에만 종속되는 부분 함수 종속이 없어야 한다.

표에서 D1의 두 기록은 입항 시각과 목적·담당도선사가 달라도 도크관리자가 모두 한경사이고, D2의 두 기록은 모두 김경위이다. 이 문항은 각 도크에 관리자가 정해지는 '도크번호 → 도크관리자' 관계를 나타낸다. 따라서 도크관리자는 **기본키의 일부인 도크번호만으로 결정**되어 제2정규형을 위반한다.

제3정규형과 BCNF는 제2정규형보다 강한 조건이므로 제2정규형을 만족하지 못하는 이 테이블은 두 정규형도 만족하지 못한다. 주어진 선택지에서 만족하는 정규형은 **제1정규형**이며, 선택지 ①과 일치한다.

NoteBase

문 17. 신상정보 릴레이션에 대한 설명으로 가장 적절하지 않은 것은? (단, 각 학생의 이메일은 고유하다. 한 학생은 다수의 과목을 수강할 수 있으며 각 수강 과목마다 학점을 받는다.)

<신상정보>

학번	이름	이메일	과목번호	학점
250726	최경위	kychoi@knpu.ac.kr	CS310	A0
250726	최경위	kychoi@knpu.ac.kr	CS313	B+
240727	한경감	kyhan@knpu.ac.kr	CS345	B0
240727	한경감	kyhan@knpu.ac.kr	CS310	A+

① 신상정보 릴레이션은 무손실 분해가 불가능하다.

② {학번→이름, 이메일}와 {이메일→학번, 이름} 뿐만 아니라 {(학번, 과목번호)→학점} 함수적 종속성이 모두 만족된다.

③ 한 학생이 여러 과목을 수강하면 (학번, 이름, 이메일)이 중복해서 나타날 수 있다.

④ 밑줄 친 속성을 키로 하여 릴레이션을 분할한 뒤 자연 조인하면 신상정보 릴레이션의 정보를 다시 얻어낼 수 있다.

정답 1번

간단한 해설

학생 정보와 수강 정보를 나누더라도 공통 속성인 학번으로 자연 조인하면 원래 튜플을 복원할 수 있으므로 **무손실 분해가 가능**하다. 학번이 학생의 이름과 이메일을 하나로 결정하기 때문에 불필요한 튜플이 생기지 않는다.

NoteBase

선지별 해설

① X: 무손실 분해는 나눈 릴레이션을 자연 조인했을 때 원래 릴레이션이 정확히 복원되는 분해이다. 이 릴레이션은 학생(학번, 이름, 이메일)과 수강(학번, 과목번호, 학점)으로 무손실 분해할 수 있다. 공통 속성인 **학번이 학생 릴레이션의 모든 속성을 결정**하므로, 조인할 때 수강 행마다 해당 학생 정보만 결합된다.

② O: 학번은 학생 한 명을 식별하므로 이름과 이메일을 결정한다. 문제에서 이메일도 학생마다 고유하므로 이메일로 학번과 이름을 결정할 수 있다. 또한 한 학생의 특정 과목에 대한 학점은 학번과 과목번호의 조합으로 결정된다. 따라서 제시된 세 함수적 종속성이 모두 성립한다.

③ O: 하나의 튜플이 학생의 과목별 수강 정보를 나타내므로, 같은 학생이 여러 과목을 수강하면 이름과 이메일 같은 학생 정보가 수강 행마다 반복된다. 그림에서도 학번 250726은 CS310과 CS313을 수강하여 같은 학번·이름·이메일이 두 행에 나타난다. 과목번호가 다르므로 이들은 서로 다른 튜플이다.

④ O: 그림에서 밑줄 친 학번과 과목번호는 원래 릴레이션의 복합키를 이룬다. 학생 릴레이션에서는 학번을 키로, 수강 릴레이션에서는 학번과 과목번호를 함께 키로 사용한다. 두 릴레이션을 **공통 속성인 학번으로 자연 조인**하면 이름·이메일과 과목번호·학점이 다시 결합되어 원래 신상정보를 얻는다.

NoteBase

문 18. 학생 릴레이션과 학과 릴레이션에 대한 자연 조인을 통해 역정규화된 학생정보 릴레이션을 새로 생성하였을 때, 이에 대한 설명으로 가장 적절하지 않은 것은? (단, 밑줄 친 속성은 기본키이다.)

학생(학번, 학과이름)
학과(학과이름, 학과전화번호)
학생정보(학번, 학과이름, 학과전화번호)

① 기존에 비해 데이터 중복과 갱신 이상이 더 발생한다.
② 생성된 릴레이션은 제3정규형을 만족한다.
③ 빈번한 검색 질의의 성능을 높이기 위한 변환이다.
④ 기존 릴레이션들에 비해 조인의 필요성이 감소한다.

정답 2번

간단한 해설

학생정보에는 학번 → 학과이름 → 학과전화번호의 **이행 함수 종속**이 존재한다. 기본키가 아닌 학과이름이 비주요 속성인 학과전화번호를 결정하므로 **제3정규형을 만족하지 않는다**.

NoteBase

선지별 해설

기존 학생 릴레이션의 기본키는 학번이고 학과 릴레이션의 기본키는 학과이름이다. 두 릴레이션을 공통 속성인 학과이름으로 자연 조인하면 학생정보(학번, 학과이름, 학과전화번호)가 되며 기본키는 학번이다. 같은 학과의 학생마다 해당 학과의 전화번호가 함께 저장된다.

① O: 여러 학생이 같은 학과에 속하면 학과전화번호가 학생별 튜플에 반복된다. 전화번호를 바꿀 때 해당 학과 학생들의 튜플을 모두 수정해야 하므로 **중복과 갱신 이상이 증가**한다. 일부 튜플만 수정하면 같은 학과의 전화번호가 서로 달라질 수 있다.

② X: 학번이 학과이름을 결정하고 학과이름이 학과전화번호를 결정하므로 학번에서 학과전화번호로의 이행 함수 종속이 생긴다. 학과이름은 학생정보의 슈퍼키가 아니고 학과전화번호도 후보키에 속하는 주요 속성이 아니어서 **제3정규형 조건에 어긋난다**. 따라서 생성된 릴레이션이 제3정규형을 만족한다는 ②가 적절하지 않다.

③ O: 학생과 그 학생의 학과전화번호를 자주 함께 조회한다면 기존 구조에서는 두 릴레이션을 조인해야 한다. 역정규화한 학생정보에서는 이 정보를 한 릴레이션에서 읽을 수 있어 **반복되는 조인 비용을 줄이는 것**이 목적이다. 이는 특정 검색 질의의 성능 개선을 위한 선택이며 모든 질의가 항상 빨라진다는 뜻은 아니다.

④ O: 자연 조인 결과를 새 릴레이션에 미리 저장했으므로 학생의 학과전화번호를 조회할 때마다 학생과 학과를 다시 조인할 필요가 줄어든다. 이처럼 조회 시 조인의 필요성을 낮추는 대신 데이터 중복과 일관성 관리 부담을 감수한다.

NoteBase

문 19. DBMS가 가변 길이 레코드 저장 방식을 사용할 때 블록 내에서 레코드를 관리하기 위해 사용되는 구조로 가장 적절한 것은?

① 계층적 구조(hierarchical structure)

② 슬롯 페이지 구조(slotted-page structure)

③ 오버플로 블록 구조(overflow-block structure)

④ 블록 강제 출력 구조(forced output-block structure)

정답 2번

간단한 해설

가변 길이 레코드는 크기가 서로 달라 블록 안의 위치를 고정된 간격으로 계산하기 어렵다. **슬롯에 각 레코드의 위치와 길이를 기록**하는 슬롯 페이지 구조로 레코드와 빈 공간을 관리한다.

개념 구분

슬롯 페이지 구조는 블록 안에 레코드 영역과 슬롯 정보를 두고, 각 슬롯으로 레코드의 시작 위치와 길이 등을 관리하는 방식이다. 레코드마다 길이가 달라도 슬롯을 통해 해당 레코드를 찾을 수 있다. 빈 공간을 모으기 위해 레코드를 블록 안에서 이동할 때는 **슬롯이 가리키는 위치를 갱신**하므로 슬롯을 통한 참조를 유지할 수 있다.

계층적 구조는 데이터를 부모와 자식의 상하 관계로 조직하는 데 초점이 있다. 블록 내부의 가변 길이 레코드 위치를 슬롯으로 관리하는 방식과 구별된다.

오버플로 블록은 원래 블록에 저장 공간이 부족할 때 추가 데이터를 수용하는 데 사용된다. 추가 블록을 활용하는 방식이라는 점에서 **한 블록 내부의 위치·길이 관리**를 묻는 문제의 핵심과 다르다.

블록의 강제 출력은 버퍼에 있는 블록을 디스크 등 저장장치에 쓰는 동작과 관련된다. 가변 길이 레코드를 블록 안에 배치하고 찾아가는 슬롯 관리 구조를 뜻하지 않는다.

따라서 제시된 기능에 해당하는 구조는 ② 슬롯 페이지 구조이다.

NoteBase

문 20. B^+-트리 인덱스를 이용하여 탐색 키 값에 대한 레코드 포인트를 획득하는 검색 연산을 기술한 알고리즘의 수행 순서로 가장 적절한 것은?

<용어>
V : 탐색키 값
K_i : 노드 내 i번째 키 값
P_i : 노드 내 i번째 포인터
N : 현재의 노드(초기의 N은 루트 노드)

<검색알고리즘>
가. 아래 순서로 다음 노드를 결정한다.
㉠ K_i = V일 경우, 포인터 P_{i+1}이 가리키는 노드를 N으로 결정
㉡ K_i > V일 경우, P_i가 가리키는 노드를 N으로 결정
㉢ V보다 큰 탐색키가 없는 경우, 노드 내의 NULL이 아닌 마지막 포인터가 가리키는 노드를 N으로 결정
나. N이 단말노드가 아니면 처음부터 반복하고, N이 단말 노드인 경우, 탐색키 값 K_i와 V가 같은 포인터 P_i를 반환한다.
다. 현재 탐색 대상인 N을 조사하여 V와 같거나 큰 탐색키 중 가장 작은 키 K_i를 찾는다.

① 다 → 나 → 가
② 나 → 가 → 다
③ 나 → 다 → 가
④ 다 → 가 → 나

정답 4번

간단한 해설

다음 자식 노드를 정하려면 먼저 현재 노드에서 **탐색값과 비교할 키**를 찾아야 한다. 그 비교 결과로 자식 노드로 이동한 뒤, **단말 노드인지 확인**하여 탐색을 계속할지 레코드 포인터를 반환할지 결정한다.

NoteBase

올바른 순서

다 → 가 → 나

다. 처음에는 루트 노드를 현재 노드 N으로 두고, N 안에서 탐색값 V 이상인 키 중 가장 작은 K_i를 찾는다. 이 비교 결과가 다음에 따라갈 포인터를 선택하는 기준이 된다. V 이상인 키가 없으면 마지막 포인터를 사용하는 경우로 이어진다.

가. 앞 단계에서 얻은 키의 비교 결과에 따라 **다음 자식 노드를 선택**한다. $K_i = V$이면 P_{i+1}을 따라가고, $K_i > V$이면 P_i를 따라간다. V보다 큰 키가 없고 앞의 일치 조건도 적용되지 않으면 NULL이 아닌 마지막 포인터를 따른다. 선택한 포인터가 가리키는 노드로 N을 갱신하므로, 키를 조사하는 다가 이 단계보다 먼저 수행되어야 한다.

나. 갱신된 N이 단말 노드인지 확인한다. 단말 노드가 아니면 다로 돌아가 키 조사와 자식 노드 선택을 반복한다. 단말 노드에 도달하면 **V와 같은 키에 대응하는 레코드 포인터**를 반환한다. 제시된 알고리즘에서는 그 키를 K_i, 반환하는 포인터를 P_i로 표시한다.

NoteBase

문 21. 정적 해싱이 가지는 단점을 해결하기 위한 확장성 해싱의 설명으로 가장 적절하지 않은 것은?

① 확장성 해싱 함수는 먼저 레코드의 키 값을 일정 길이의 비트 스트링(bit string)으로 만드는데 이를 보조키(auxiliary key)라고 한다.

② 파일의 크기와 관계없이 한 번의 디스크 접근만으로 특정 레코드를 검색할 수 있다는 것을 보장한다.

③ 디렉토리(directory)라는 버킷 주소 테이블(bucket address table)과 버킷으로 구성된 2단계 구조를 가진다.

④ 디스크에 저장된 디렉토리는 디렉토리 깊이 d(정수값)를 포함한 헤더와 데이터 레코드가 저장되어 있는 버킷에 대한 2^d개의 포인터로 구성된다.

정답 1번

간단한 해설

해시 함수가 키 값을 일정 길이의 비트열로 바꾼 결과는 **의사키(pseudokey)**이다. 이를 '보조키(auxiliary key)'라고 부른 부분이 잘못되었다.

출제 포인트

확장성 해싱의 의사키, 디렉토리와 버킷의 역할, 디렉토리 깊이에 따른 포인터 수를 구분하는 문제이다.

핵심 개념

확장성 해싱은 데이터가 늘어날 때 버킷을 분할하고 필요하면 디렉토리를 확장하는 동적 해싱 방식이다. 버킷은 레코드를 담는 저장 공간이고, 디렉토리는 해당 버킷의 주소를 찾는 포인터 테이블이다.
의사키는 원래 키를 해시 함수로 변환한 비트열이다. 디렉토리 깊이 d는 버킷 주소를 찾는 데 사용하는 비트 수이므로 디렉토리 항목은 2^d개이다. 예를 들어 d = 2이면 00, 01, 10, 11에 대응하는 포인터 4개가 있다. 여러 포인터가 같은 버킷을 가리킬 수도 있으므로 포인터 수와 실제 버킷 수는 반드시 같지는 않다.

NoteBase

선지별 해설

① X
키 값을 일정 길이의 비트열로 바꾼다는 설명은 맞지만, 그 결과의 명칭이 틀렸다. '보조키(auxiliary key)'를 '의사키(pseudokey)'로 고친다. 의사키의 일부 비트를 이용해 디렉토리 항목을 선택한다.

② O
디렉토리가 메모리에 있고 레코드가 해당 버킷에 저장된 경우, 메모리에서 버킷 주소를 찾은 뒤 그 버킷을 읽는 한 번의 디스크 접근으로 검색한다. 이 선지의 한 번이라는 설명은 이 전제에서 성립한다. 디렉토리까지 디스크에서 읽어야 하면 추가 접근이 필요하므로 모든 저장 조건에서 한 번을 보장한다는 뜻으로 일반화할 수는 없다.

③ O
디렉토리에서 버킷 주소를 찾고, 그 주소가 가리키는 버킷에서 레코드를 찾는다. 따라서 디렉토리와 버킷으로 이루어진 2단계 구조라는 설명이 맞다.

④ O
디렉토리에는 깊이 d를 기록한 헤더와 버킷을 가리키는 2^d개의 포인터가 있다. d비트로 표현할 수 있는 비트 조합이 2^d개이기 때문이다. 포인터는 레코드 자체가 아니라 레코드가 저장된 버킷의 주소를 담는다.

오답 포인트

①은 비트열을 만드는 과정은 맞게 설명하고 명칭만 '보조키'로 바꾸었다. 과정이 익숙하다는 이유로 용어까지 맞다고 판단하기 쉽다. 또한 ②의 디스크 접근 횟수는 디렉토리의 메모리 상주 여부와 함께 이해해야 한다.

다시 나오면 이렇게 판단

키를 해시 함수로 변환한 값은 의사키, 버킷 주소를 담는 테이블은 디렉토리, 실제 레코드를 담는 공간은 버킷으로 구분한다. 깊이가 d이면 디렉토리 포인터는 2^d개이며, 검색의 디스크 접근 횟수는 디렉토리를 메모리에서 이용할 수 있는지에 따라 판단한다.

암기 포인트

키 → 의사키 → 디렉토리 → 버킷. 디렉토리 깊이 d → 포인터 2^d개.

NoteBase

문 22. 디스크를 병렬로 연결하거나 물리적으로 이중화하여 얻을 수 있는 효과로 가장 적절하지 않은 것은?

① 스트라이핑 기법으로 디스크를 병렬 구동하면 데이터 입출력을 더욱 빠르게 할 수 있다.

② 디스크 병렬 구동 시 프로세서 연산 속도가 병렬화에 비례하여 향상되므로 프로세서는 같은 시간에 더 많은 작업을 처리할 수 있다.

③ 미러링 기법이나 디스크 이중화 기법으로 디스크를 병렬 구동하면 어느 한 디스크에 장애가 발생하더라도 장애를 극복할 수 있다.

④ 수행 중, 때로는 에러 감지 코드를 또 다른 디스크에 저장함으로써 런타임 데이터 저장 신뢰도를 높일 수 있다.

정답 2번

간단한 해설

디스크 병렬화는 여러 디스크의 입출력을 동시에 수행하여 **저장장치의 처리량**을 높이는 기법이다. **프로세서 자체의 연산 속도**가 디스크 수에 비례하여 빨라지는 것은 아니다.

선지별 해설

여러 디스크를 함께 사용하는 목적에는 입출력 성능 향상과 장애에 대비한 신뢰성 향상이 있다. 데이터를 분산하는 스트라이핑과 같은 데이터를 복제하는 미러링은 목적과 효과가 다르며, 디스크의 처리 성능과 CPU의 연산 속도도 구별해야 한다.

① O: 스트라이핑은 데이터를 여러 디스크에 나누어 저장하고 각 디스크가 입출력을 병렬로 수행하게 한다. 따라서 작업을 여러 장치에 분산하여 **데이터 전송 처리량을 높일 수 있다**. 다만 실제 향상 정도는 요청 크기와 접근 방식 등의 영향을 받는다.

② X: 디스크 병렬화는 저장장치의 입출력 병목을 완화하지만 CPU의 클록이나 연산 장치 자체를 병렬화하지는 않는다. 입출력 대기 시간이 줄어 전체 작업 처리량이 좋아질 수는 있어도 **프로세서 연산 속도가 디스크 병렬화에 비례해 향상되지는 않는다**. 따라서 두 성능을 혼동한 ②가 적절하지 않다.

③ O: 미러링은 동일한 데이터를 둘 이상의 디스크에 복제해 둔다. 한 디스크가 고장 나도 **정상인 복제본으로 데이터를 읽고 서비스를 계속**할 수 있어 장애에 대한 내성이 높아진다. 단순 스트라이핑만으로 이러한 복제 효과가 생기는 것은 아니다.

④ O: 데이터와 함께 오류 검출용 중복 정보를 별도의 디스크에 저장하면 읽은 데이터가 손상되었는지 검사하는 데 활용할 수 있다. 이처럼 **검사용 중복 정보를 추가**하면 데이터 저장의 신뢰성을 높일 수 있다. 오류 검출과 복구는 구별되며, 실제 복구 가능 여부는 사용한 코드와 디스크 구성에 달려 있다.

NoteBase

문 23. 비신장 조직 기반 고정 길이 레코드 저장 방식에서 하나의 블록에 저장 가능한 레코드 수로 가장 적절한 것은? (단, 레코드 길이는 408바이트, 블록 크기는 4,096바이트, 블록 헤더 길이는 24바이트이다.)

① 9개

② 10개

③ 11개

④ 12개

정답 1번

간단한 해설

블록 헤더를 제외한 저장 공간은 4,072바이트이다. 비신장 방식은 **레코드를 블록 경계에서 나눌 수 없으므로** 408바이트 레코드를 9개까지 저장한다.

계산 과정

사용 가능한 공간: 블록 전체 크기 4,096바이트에서 헤더 24바이트를 제외한다. 헤더는 블록 관리 정보이므로 레코드를 저장할 공간에 포함하지 않는다.
4,096 − 24 = 4,072바이트

레코드 수: 비신장 조직에서는 각 레코드가 **한 블록 안에 완전히 들어가야** 한다. 따라서 사용 가능한 공간을 레코드 길이 408바이트로 나눈 값에서 소수 부분을 버린다.
$\left\lfloor \frac{4,072}{408} \right\rfloor$ = 9개

공간 확인: 9개가 차지하는 공간은 9 × 408 = 3,672바이트이며, 남는 공간은 4,072 − 3,672 = 400바이트이다. 잔여 공간 400바이트는 레코드 하나의 길이 408바이트보다 작다. 10개를 넣으면 10 × 408 = 4,080바이트가 필요하여 사용 가능한 공간을 8바이트 초과한다.

따라서 한 블록에 저장 가능한 레코드 수는 9개이며 ①과 일치한다.

NoteBase

문 24. 밀집 인덱스를 탐색하여 데이터 블록에 접근하기 위한 총 소요 시간으로 가장 근접한 것은?

<밀집 인덱스 속성 및 값>

밀집 인덱스 속성	값
블록 크기	4,096바이트
인덱스 내 키 필드 길이	11바이트
인덱스 내 블록 포인터 길이	4바이트
접근 대상 데이터 레코드 개수	500,000개
하나의 블록을 읽는 시간	5ms
데이터 블록에 접근하는 방법	이진 키 탐색법

① 60ms

② 180ms

③ 360ms

④ 720ms

정답 1번

간단한 해설

인덱스 엔트리는 키 11바이트와 포인터 4바이트를 합친 15바이트이므로 인덱스는 약 1,832개 블록으로 구성된다. 이진 탐색에 약 11회, 실제 데이터 블록 읽기에 1회가 필요하므로 **총 12회 × 5ms = 60ms**이다.

NoteBase

계산 과정

밀집 인덱스는 데이터 레코드마다 인덱스 엔트리를 둔다. 한 엔트리는 키 필드와 데이터 블록 포인터로 구성되므로 크기는 11 + 4 = 15바이트이고, 500,000개 레코드에 대응하는 엔트리도 500,000개이다.

블록 하나에 들어가는 엔트리 수는 블록 크기를 엔트리 크기로 나눈 뒤 버린 값이다. 엔트리 하나가 블록에 온전히 들어가야 하므로 소수 부분을 올리지 않는다.
$\left\lfloor \frac{4{,}096}{15} \right\rfloor = 273$개

인덱스를 저장할 블록 수는 엔트리 수를 블록당 엔트리 수로 나눈 뒤 올린 값이다. 마지막 블록이 가득 차지 않아도 블록 하나가 필요하다.
$\left\lceil \frac{500{,}000}{273} \right\rceil = 1{,}832$개

인덱스 블록을 이진 탐색하면 탐색 범위가 매번 약 절반으로 줄어든다. 약 1,832개 블록에서 해당 엔트리를 찾는 데 필요한 블록 접근 횟수는 약 11회이다.
$\lceil \log_2 1{,}832 \rceil = 11$회

찾은 엔트리의 포인터로 **실제 데이터 블록을 한 번 더 읽어야** 한다. 따라서 총 블록 읽기 횟수는 11 + 1 = 12회이고, 블록당 5ms가 걸리므로 총 소요 시간은 12 × 5 = 60ms이다. 선택지 ①의 60ms와 일치한다.

NoteBase

문 25. 릴레이션을 정규화할 때 각 단계에 있는 ㉠~㉢과 연관된 정규화 연산을 가~라 항목에서 가장 적절하게 선택한 것을 모두 고른 것은?

<정규화 과정>



<정규화 연산>

가. 이행 종속성을 제거

나. 부분 종속성을 제거

다. 나열 및 반복 그룹을 제거

라. 후보키가 아닌 결정자를 제거

① ㉠-나

② ㉠-나, ㉢-라

③ ㉢-라

④ ㉡-나, ㉢-가

정답 2번

간단한 해설

제1정규형에서 제2정규형으로 갈 때는 **부분 종속성**을 제거하고, 제3정규형에서 BCNF로 갈 때는 **후보키가 아닌 결정자에 의한 종속**을 분리한다. 도식의 ㉠은 나, ㉢은 라에 대응한다.

NoteBase

대응 관계 해설

정규화는 함수 종속 관계에 따라 릴레이션을 분해하여 중복과 갱신 이상을 줄이는 과정이다. 그림의 각 화살표에서 제거해야 할 종속성을 그 화살표의 출발·도착 정규형으로 판단한다.

㉠-나: 그림에서 ㉠은 제1정규형에서 제2정규형으로 가는 단계이다. 제2정규형에서는 후보키에 속하지 않는 속성이 복합 후보키의 일부에만 종속되는 **부분 함수 종속성**을 제거한다. 키 전체에 완전 함수 종속하도록 릴레이션을 분해하는 단계이므로 나와 대응한다.

㉡-가: 그림에서 ㉡은 제2정규형에서 제3정규형으로 가는 단계이다. 후보키에 속하지 않는 속성이 다른 비주요 속성을 거쳐 키에 간접적으로 종속되는 **이행 함수 종속성**을 제거한다. 따라서 부분 종속성을 제거하는 나가 아니라 가에 대응한다.

㉢-라: 그림에서 ㉢은 제3정규형에서 BCNF로 가는 단계이다. BCNF는 모든 비자명한 함수 종속에서 결정자가 슈퍼키가 되도록 요구한다. 보기의 '후보키가 아닌 결정자를 제거'는 키가 아닌 결정자 때문에 생기는 종속 관계를 **별도 릴레이션으로 분리**한다는 뜻이며, 속성이나 데이터를 무조건 삭제한다는 뜻이 아니다. 따라서 라와 대응한다.

다. 나열 및 반복 그룹을 제거하는 것은 속성값을 원자값으로 만들어 제1정규형을 갖추는 과정이다. 그림은 이미 제1정규형인 릴레이션에서 시작하므로 ㉠~㉢에 해당하지 않는다.

올바른 대응은 ㉠-나, ㉡-가, ㉢-라이다. 제시된 선택지에서 옳은 대응만 묶은 것은 ㉠-나와 ㉢-라를 제시한 ②이다.

NoteBase

문 26. 비순서 히프파일(heap file)이 다른 파일 구조에 비해 상대적으로 효율성이 높은 연산을 모두 고른 것은?

가. 조건 검색 연산(conditional search)
나. 순차 검색 연산(sequential search)
다. 삽입 연산(insertion operation)
라. 삭제 연산(deletion operation)

① 나, 라
② 다, 라
③ 가, 나
④ 다

정답 4번

간단한 해설

비순서 히프파일은 레코드의 정렬 순서를 유지하지 않으므로 새 레코드를 빈 공간이나 파일 끝에 추가하는 **삽입 연산**에 유리하다. 검색이나 삭제는 대상 레코드를 먼저 찾아야 하므로 같은 이점이 없다.

NoteBase

보기별 해설

가. X: 비순서 히프파일은 탐색 키 순서로 정렬되어 있지 않으므로, 조건에 맞는 레코드를 찾을 때 파일을 차례로 조사해야 한다. 키를 이용해 탐색 범위를 줄이는 정렬 파일이나 인덱스·해시 기반 구조에 비해 조건 검색이 유리하다고 볼 수 없다.

나. X: 순차 검색은 레코드를 차례로 읽는 연산이다. 히프파일도 순차적으로 읽을 수 있지만, 전체 자료를 찾으려면 필요한 레코드를 계속 조사해야 하므로 **비순서 저장 자체가 검색량을 줄이지는 않는다**. 정렬 파일도 순차 탐색이 가능하며, 히프파일이 이러한 검색에서 특별히 높은 효율성을 갖는 것은 아니다.

다. O: 삽입할 레코드를 키 순서에 맞는 위치로 옮겨 넣을 필요가 없다. 빈 공간이 있는 페이지나 파일 끝에 레코드를 추가하면 되므로 **정렬 상태를 유지하기 위한 이동 부담**이 작다. 이는 비순서 히프파일의 대표적인 장점이다.

라. X: 삭제할 위치가 이미 알려졌다면 해당 공간을 비어 있는 것으로 표시할 수 있지만, 일반적인 삭제 연산에는 대상 레코드를 찾는 과정이 포함된다. 비순서 히프파일은 이 검색에 비용이 들기 때문에 삭제가 다른 파일 구조보다 특별히 효율적이라고 할 수 없다.

따라서 상대적으로 효율성이 높은 연산은 다의 삽입 연산이며, '다'만을 제시한 ④와 일치한다.

NoteBase

문 27. 릴레이션 R에서 함수 종속성 F가 주어졌을 때 R이 제3정규형을 만족하도록 수행해야 할 내용으로 가장 적절한 것은?

$R(StID, Advisor, Room)$
$F = \{StID \rightarrow Advisor,\ Advisor \rightarrow Room\}$

① R에서 StID→Advisor만 유지하고 Room은 제거한다.

② R로부터 StID→Room을 새로 추가한다.

③ R을 R1(StID, Advisor)과 R2(Advisor, Room)로 분해한다.

④ 함수 종속성은 후보키에 기반하므로 R의 분해는 불필요하다.

정답 3번

간단한 해설

StID가 Advisor를 결정하고 Advisor가 Room을 결정하므로 Room은 후보키 StID에 **이행적으로 종속**된다. 이 종속 구조를 해소하려면 **R1(StID, Advisor)과 R2(Advisor, Room)로 분해**하여 지도교수의 방 정보를 별도로 저장한다.

NoteBase

핵심 풀이

주어진 함수 종속성은 StID → Advisor와 Advisor → Room이다. StID를 알면 Advisor를 알고 다시 Room을 알 수 있으므로 StID는 세 속성을 모두 결정하는 후보키이다. 반면 Advisor만으로는 StID를 결정할 수 없어 Advisor는 원래 릴레이션 R의 슈퍼키가 아니다.

제3정규형에서는 비자명한 함수 종속성의 결정자가 슈퍼키이거나, 종속되는 속성이 후보키에 포함되는 주요 속성이어야 한다. Advisor → Room에서는 Advisor가 슈퍼키가 아니고 Room도 주요 속성이 아니므로 **제3정규형 조건을 위반**한다. 이는 StID → Advisor → Room으로 이어지는 이행 함수 종속이다.

R을 R1(StID, Advisor)과 R2(Advisor, Room)로 분해하면 R1의 기본키는 StID, R2의 기본키는 Advisor가 된다. 각 함수 종속성의 결정자가 해당 릴레이션의 키가 되므로 제3정규형 조건을 만족한다. 두 릴레이션의 공통 속성 Advisor가 R2를 결정하므로 **자연 조인으로 원래 정보를 손실 없이 복원**할 수 있고, 주어진 두 함수 종속성도 각 릴레이션에서 유지된다.

①처럼 Room을 단순히 제거하면 방 정보를 잃는다.

②의 StID → Room은 이미 주어진 종속성에서 이행적으로 도출되며, 이를 명시한다고 잘못된 종속 구조가 사라지지는 않는다.

④는 Advisor가 R의 후보키나 슈퍼키가 아니라는 점을 놓친 설명이다.

따라서 두 종속 관계를 R1(StID, Advisor)과 R2(Advisor, Room)로 분리하는 ③이 해당한다.

NoteBase

문 28. 뷰에 대한 설명으로 적절한 것을 모두 고른 것은?

가. 시스템 관리 효율을 위해 생성된 뷰를 통해 새로운 뷰를 정의할 수 없다.
나. 일관성을 확보하기 위해 하나의 테이블에는 하나의 뷰만 정의할 수 있다.
다. 두 개 이상의 테이블을 조인하여 뷰를 정의할 수 있다.
라. 기본 테이블에 직접 접근할 수 있는 권한을 부여하지 않고 뷰를 통해 데이터에 접근할 수 있기 때문에 보안 기능을 제공한다.

① 가, 나
② 가, 라
③ 나, 다
④ 다, 라

정답 4번

간단한 해설

뷰는 **여러 테이블을 조인한 질의**로 정의할 수 있다. 기본 테이블에 대한 직접 접근 권한 대신 필요한 데이터만 보여 주는 뷰의 접근 권한을 부여하면 **데이터 노출 범위를 제한**할 수 있다.

NoteBase

보기별 해설

뷰는 저장된 질의를 바탕으로 사용자에게 테이블처럼 보이는 데이터의 논리적 표현이다. 기본 테이블의 전체 구조를 그대로 노출하지 않고 필요한 행·열이나 여러 테이블의 결합 결과를 보여 줄 수 있다.

가. X: **이미 정의된 뷰를 이용하여 새로운 뷰를 정의할 수 있다**. 뷰도 질의의 데이터 원본으로 사용할 수 있으므로, 기존 뷰의 결과에서 필요한 행이나 열을 다시 선택하는 뷰를 만들 수 있다.

나. X: 하나의 기본 테이블을 바탕으로 **여러 개의 뷰**를 정의할 수 있다. 예를 들어 같은 직원 테이블에서 부서별 직원만 보여 주는 뷰와 일부 열만 보여 주는 뷰를 각각 만들 수 있다. 일관성을 위해 뷰를 하나로 제한하는 것은 아니다.

다. O: 뷰를 정의하는 질의에는 두 개 이상의 테이블을 결합하는 조인을 사용할 수 있다. 여러 테이블에서 필요한 정보를 하나의 가상 테이블처럼 제공할 수 있다.

라. O: 기본 테이블에 대한 직접 접근을 제한하고 뷰에 대한 접근 권한을 부여하여, 사용자에게 **허용된 행과 열만 노출**할 수 있다. 이러한 권한 설정을 이용하면 기본 테이블 전체를 공개하지 않고 필요한 데이터에 접근하게 하는 보안 기능을 제공한다.

적절한 보기는 다와 라이므로 ④와 일치한다.

NoteBase

문 29. 트랜잭션의 특징을 적절하게 설명한 것을 모두 고른 것은?

가. 트랜잭션 수행 도중 장애가 발생하면 일부 연산만이라도 완료하는 것을 보장한다.
나. 완료된 트랜잭션 내에서 갱신된 사항은 트랜잭션을 재수행하여 원자성을 보장한다.
다. 동시성 제어를 통해 트랜잭션의 고립성을 보장한다.
라. 완료된 트랜잭션의 수행 결과는 향후 시스템의 고장이 발생할 경우 손실될 수 있다.

① 가, 다
② 가, 라
③ 나, 다
④ 나, 라

정답 3번

간단한 해설

회복은 성공한 트랜잭션의 변경을 **전부 반영**하고 실패한 트랜잭션의 변경은 전부 제거하여 일부만 수행된 상태가 남지 않게 한다. 동시성 제어는 동시에 실행되는 트랜잭션 사이의 간섭을 조정하여 **고립성**을 보장한다.

NoteBase

보기별 해설

가. X: 원자성은 트랜잭션의 연산이 전부 수행되거나 전혀 수행되지 않은 것처럼 처리되는 성질이다. 장애가 발생했다고 일부 연산의 완료만 보장하는 것이 아니다. 완료되지 못한 트랜잭션은 이미 반영한 변경을 취소하는 등 **부분 실행의 효과가 남지 않도록** 복구한다.

나. O: 완료된 트랜잭션의 갱신 중 저장장치에 아직 반영되지 않은 부분은 로그를 이용한 재수행(REDO)으로 복구한다. 성공한 트랜잭션의 효과가 일부만 반영된 상태로 남지 않게 하여 원자성의 '전부 수행' 결과를 확보하며, 완료된 결과를 보존하는 지속성도 함께 보장한다. 여기서 재수행은 **기록된 갱신 내용을 다시 반영**하는 회복 절차이다.

다. O: 고립성은 동시에 실행되는 트랜잭션들이 서로의 중간 처리에 부당하게 영향을 주지 않도록 하는 성질이다. 동시성 제어는 잠금이나 실행 순서 조정 등을 통해 충돌하는 데이터 접근을 제어하여, 병행 실행의 결과가 올바른 순차 실행과 일치하도록 한다.

라. X: 지속성은 성공적으로 완료된 트랜잭션의 결과가 이후 장애가 발생하더라도 유지되어야 한다는 성질이다. 로그와 회복 절차를 이용하여 **완료된 변경을 복구·보존**해야 하므로, 시스템 고장 시 결과의 손실을 허용한다는 설명은 지속성에 어긋난다.

적절한 보기는 나와 다이므로 선택지 ③과 일치한다.

NoteBase

문 30. 갱신 전략에 따른 회복 기법에 대한 설명으로 가장 적절하지 않은 것은?

① 즉시 갱신을 수행한 트랜잭션이 완료되기 전에 장애가 발생할 경우 UNDO 연산을 수행한다.

② 즉시 갱신을 수행한 트랜잭션이 완료된 후에 장애가 발생할 경우 REDO 연산을 수행할 필요가 없다.

③ 지연 갱신을 수행한 트랜잭션이 완료되기 전에 장애가 발생할 경우 UNDO 연산을 수행할 필요가 없다.

④ 지연 갱신을 수행한 트랜잭션이 완료된 후에 장애가 발생할 경우 REDO 연산을 수행한다.

정답 2번

간단한 해설

즉시 갱신에서도 트랜잭션이 완료되었다는 사실만으로 **모든 변경이 디스크에 반영되었다고 볼 수는 없다**. 장애로 유실된 완료 트랜잭션의 변경은 로그를 이용한 **REDO**로 다시 반영해야 한다.

선지별 해설

UNDO는 완료되지 않은 트랜잭션의 변경을 취소하여 원래 값으로 되돌리는 연산이고, REDO는 완료된 트랜잭션의 변경을 다시 반영하는 연산이다. 즉시 갱신은 완료 전에도 데이터베이스에 변경을 반영할 수 있으며, 지연 갱신은 완료가 확정될 때까지 데이터베이스 반영을 미룬다.

① O: 즉시 갱신에서는 아직 완료되지 않은 트랜잭션의 변경이 데이터베이스에 반영될 수 있다. 이 상태에서 장애가 발생하면 해당 트랜잭션을 취소해야 하므로 로그의 갱신 전 값을 이용하여 **미완료 변경을 UNDO**한다.

② X: 즉시 갱신은 완료 전 반영을 허용한다는 뜻이지, 완료 시 모든 변경 페이지를 반드시 디스크에 기록한다는 뜻은 아니다. 완료된 트랜잭션의 변경 중 디스크에 남지 않은 부분은 장애 후 **로그의 갱신 후 값으로 REDO**해야 한다. 따라서 완료 후에는 REDO가 필요 없다고 단정한 ②가 적절하지 않다.

③ O: 지연 갱신에서는 트랜잭션이 완료되기 전의 변경을 데이터베이스에 반영하지 않는다. 완료 전 장애가 발생해도 데이터베이스에는 취소할 변경이 없으므로 **UNDO가 필요 없다**. 해당 미완료 트랜잭션의 로그를 데이터베이스에 적용하지 않으면 된다.

④ O: 지연 갱신에서도 완료는 확정되었지만 변경을 데이터베이스에 모두 반영하기 전에 장애가 발생할 수 있다. 회복 시 완료된 트랜잭션의 로그에 기록된 변경을 REDO하여 그 결과가 데이터베이스에 남도록 한다. 이 때문에 지연 갱신은 일반적으로 UNDO 없이 REDO를 사용하는 회복 방식으로 설명한다.

NoteBase

문 31. 동시성 제어를 수행하지 않을 때 발생하는 문제에 대한 설명에서 빈칸에 가장 적절한 것은?

가. (㉠)은(는) 트랜잭션 T1이 갱신한 내용을 다른 트랜잭션 T2가 덮어씀으로써 트랜잭션 T1의 갱신이 무효가 되는 것을 의미한다.
나. (㉡)은(는) 트랜잭션 T1이 데이터 X을 한 번 읽고 두 번째 읽기 연산을 하는 사이에 다른 트랜잭션 T2가 X를 변경하면 T1은 X에 대해 다른 값을 읽는 것을 의미한다.

① ㉠ 갱신 손실 ㉡ 부정확한 요약
② ㉠ 갱신 손실 ㉡ 반복할 수 없는 읽기
③ ㉠ 오손 읽기 ㉡ 부정확한 요약
④ ㉠ 오손 읽기 ㉡ 반복할 수 없는 읽기

정답 2번

간단한 해설

다른 트랜잭션의 덮어쓰기로 앞선 갱신 결과가 사라지는 현상은 **갱신 손실**이다. 같은 트랜잭션이 같은 데이터를 다시 읽었는데 다른 트랜잭션의 변경으로 값이 달라지는 현상은 **반복할 수 없는 읽기**이다.

빈칸별 해설

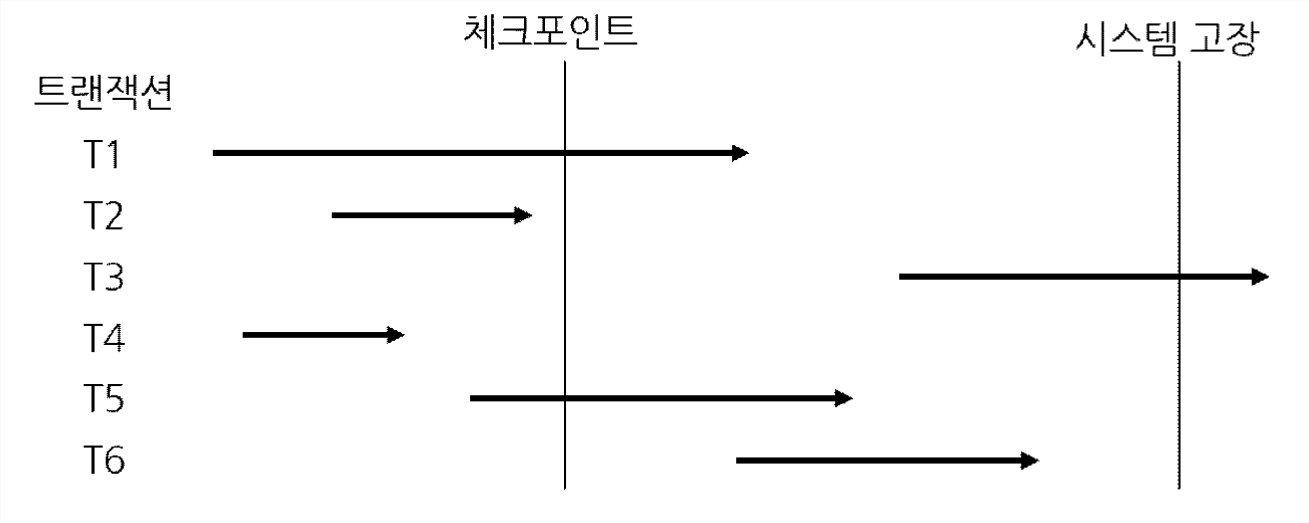
㉠ 갱신 손실: T1이 데이터를 변경한 뒤 T2가 그 갱신 결과를 반영하지 않은 값으로 다시 덮어쓰면 T1의 변경 효과가 사라진다. 가의 결정적 단서는 **다른 트랜잭션의 덮어쓰기로 갱신이 무효가 된다**는 점이다. 확정되지 않은 값을 읽는 오손 읽기와 달리, 쓰기 결과가 소실되는 문제이다.

㉡ 반복할 수 없는 읽기: T1이 X를 읽고 같은 X를 다시 읽는 사이에 T2가 X를 바꾸면 두 번의 읽기 결과가 달라질 수 있다. 나에서는 **같은 데이터에 대한 재읽기 결과가 달라진다**는 점이 핵심이다. 오손 읽기는 다른 트랜잭션이 아직 확정하지 않은 값을 읽는 현상이며, 부정확한 요약은 여러 데이터의 합계 등을 구하는 도중 다른 갱신이 끼어들어 일관된 시점의 결과를 얻지 못하는 현상이다.

따라서 ㉠은 갱신 손실, ㉡은 반복할 수 없는 읽기이며, 이 조합을 제시한 ②와 일치한다.

NoteBase

문 32. 즉시 갱신을 수행하는 DBMS에서 다음과 같이 체크포인트가 수행되었다고 할 때 REDO를 수행할 트랜잭션을 모두 고른 것은?



① T2, T4

② T3

③ T1, T5, T6

④ T1, T2, T4, T6

정답 3번

간단한 해설

이 문항의 체크포인트 회복 방식에서는 **체크포인트 이후부터 고장 전까지 완료된 트랜잭션**의 결과를 REDO로 다시 반영한다. 체크포인트 이전에 완료된 작업은 복구 대상에서 제외하고, 고장 시 미완료인 작업은 UNDO로 취소한다.

NoteBase

도식 해설

즉시 갱신 방식은 트랜잭션이 완료되기 전에도 변경 내용을 데이터베이스에 반영할 수 있다. 따라서 고장 복구 때 완료된 작업의 결과를 되살리는 REDO와 미완료 작업의 변경을 취소하는 UNDO를 구분한다. 그림에서는 화살표의 끝이 체크포인트와 시스템 고장 세로선 중 어느 구간에 있는지 확인한다.

T1: 체크포인트 전에 시작했지만 그 이후, 시스템 고장 전에 완료되었다. **시작 시점이 아니라 완료 시점**이 체크포인트 이후이므로 REDO 대상이다.

T2: 화살표가 체크포인트 세로선 전에 끝난다. 체크포인트 이전에 이미 완료된 결과이므로 이 문제의 복구 과정에서 REDO할 필요가 없다.

T3: 체크포인트 이후에 시작했고 화살표가 시스템 고장 세로선을 넘어 이어진다. 고장 순간에는 **아직 완료되지 않은 트랜잭션**이므로 REDO가 아니라 UNDO 대상이다.

T4: 체크포인트 이전에 시작하고 완료되었다. T2와 마찬가지로 체크포인트 이전에 완료된 결과이므로 REDO 대상이 아니다.

T5: 체크포인트 전에 시작하여 체크포인트 이후, 시스템 고장 전에 완료되었다. T1과 같은 경우이므로 REDO 대상이다.

T6: 체크포인트 이후에 시작하고 시스템 고장 전에 완료되었다. 마지막 체크포인트 이후에 완료된 작업의 결과를 복구해야 하므로 REDO 대상이다.

따라서 REDO 대상은 T1, T5, T6이며 ③과 일치한다.

NoteBase

문 33. 2PL(Two Phase Locking)을 이용한 동시성 제어에 대한 설명으로 가장 적절한 것은?

① 기본적(basic) 2PL의 수축 단계에서는 공유 로크(shared lock)만을 획득할 수 있다.

② 엄격한(strict) 2PL은 트랜잭션이 완료되거나 철회될 때까지 트랜잭션이 보유한 공유 로크는 해제할 수 없지만 배타 로크(exclusive lock)는 해제할 수 있다.

③ 보수적(conservative) 2PL은 트랜잭션이 접근하려는 모든 항목들에 로크를 획득할 수 없다면 읽기 항목에 대해서만 공유 로크를 획득한다.

④ 엄중한(rigorous) 2PL은 트랜잭션이 완료되거나 철회될 때까지 트랜잭션이 보유한 어떠한 로크도 해제하지 않는다.

정답 4번

간단한 해설

엄중한(rigorous) 2PL은 트랜잭션이 완료되거나 철회될 때까지 **공유 로크와 배타 로크를 모두 유지**한다. 엄격한(strict) 2PL이 종료 시점까지 유지하도록 요구하는 것은 배타 로크이다.

선지별 해설

2PL은 로크를 획득하는 확장 단계와 해제하는 수축 단계를 구분하여 충돌 직렬 가능성을 보장하는 동시성 제어 규약이다. 공유 로크는 읽기에, 배타 로크는 다른 트랜잭션의 충돌 접근을 막는 갱신 등에 사용한다.

① X: 기본적 2PL의 수축 단계에서는 로크를 해제할 수 있지만 **공유·배타 로크 모두 새로 획득할 수 없다**. 새 로크의 획득은 확장 단계에서만 가능하며, 하나라도 로크를 해제하면 수축 단계로 넘어간다.

② X: 엄격한 2PL은 트랜잭션이 완료되거나 철회될 때까지 **배타 로크를 유지**한다. 공유 로크는 기본 2PL 규칙을 지키면서 그 전에 해제할 수 있다. 선지는 종료까지 유지해야 할 로크의 종류를 반대로 설명했다.

③ X: 보수적 2PL은 트랜잭션 실행에 필요한 로크를 **실행 전에 모두 확보**한다. 필요한 로크 전체를 확보할 수 없으면 일부 공유 로크만 보유하고 시작하지 않고, 전체를 확보할 수 있을 때까지 기다린다. 로크 일부를 보유한 채 다른 로크를 기다리는 상황을 막아 교착 상태를 예방한다.

④ O: 엄중한 2PL은 **공유 로크와 배타 로크 모두**를 트랜잭션의 완료 또는 철회 시점까지 해제하지 않는다. 배타 로크의 유지에 초점을 둔 엄격한 2PL보다 로크 유지 범위가 넓다.

NoteBase

문 34. 데이터베이스 보안을 위한 접근 제어 방법에 대한 설명으로 가장 적절한 것은?

① 임의 접근 제어는 데이터와 사용자들을 다양한 보안 등급으로 분류하고 조직에 적합한 보안 정책을 적용하여 다단계 보안을 시행한다.

② 강제 접근 제어는 사용자들에게 특정 테이블, 레코드, 속성을 임의 지정 모드로 접근할 수 있는 권한을 부여한다.

③ 임의 접근 제어는 접근 권한을 가진 사용자가 정보에 접근하면 정보가 어떻게 사용되는지 통제하지 않는다.

④ 강제 접근 제어는 임의 접근 제어보다 융통성이 높아 관계형 DBMS에 많이 사용된다.

정답 3번

간단한 해설

임의 접근 제어는 사용자에게 객체별 접근 권한을 부여하는 방식이다. 일단 허용된 접근으로 읽은 정보가 다른 곳으로 복사되거나 전달되는 등의 **후속 정보 흐름까지 강제적으로 통제하지는 못한다**.

선지별 해설

임의 접근 제어(DAC)는 객체의 소유자 등 권한을 가진 주체가 사용자별 접근 권한을 정하는 방식이다. 강제 접근 제어(MAC)는 사용자와 데이터에 부여된 보안 등급 및 시스템의 보안 정책에 따라 접근을 통제하며, 사용자가 그 정책을 임의로 바꿀 수 없다.

① X: 사용자와 데이터를 보안 등급으로 나누어 다단계 보안 정책을 적용하는 것은 **강제 접근 제어**이다. 임의 접근 제어는 등급에 의한 일률적 정보 흐름 통제보다 사용자별 객체 접근 권한을 중심으로 한다.

② X: 특정 테이블·레코드·속성에 어떤 사용자가 어떤 방식으로 접근할 수 있는지를 개별 권한으로 지정하는 설명은 **임의 접근 제어**에 해당한다. 강제 접근 제어에서는 개별 사용자의 임의 판단보다 시스템이 정한 보안 정책이 우선한다.

③ O: 임의 접근 제어는 사용자가 해당 정보에 접근할 권한이 있는지를 확인한다. 그러나 읽기를 허용받은 사용자가 정보를 복사하여 다른 객체에 저장하는 등의 후속 사용까지 정보 흐름 정책으로 강제 통제하지는 못한다. 이와 같은 **접근 허용 이후의 정보 유출 통제 한계**를 설명한 ③이 적절하다.

④ X: 일반적으로 개별 객체에 권한을 부여하고 변경하기 쉬운 **임의 접근 제어가 더 융통성이 높다**. 관계형 DBMS에서 사용되는 사용자별 권한 부여·회수는 그 대표적인 방식이다. 강제 접근 제어는 정해진 보안 정책을 엄격하게 적용하므로 상대적으로 유연성이 낮다.

NoteBase

문 35. 데이터베이스에서 수행하는 권한 관리에 대한 설명으로 가장 적절한 것은?

① 권한을 사용자에게 허가하고 취소할 때에는 GRANT 문과 DROP 문을 사용한다.

② SELECT, INSERT, DELETE, UPDATE에 대한 권한은 부여 가능하지만 REFERENCES에 대한 권한은 부여할 수 없다.

③ 테이블 또는 뷰의 소유자는 해당 테이블 또는 뷰에 대한 모든 권한을 갖는다.

④ 일반 사용자는 허가받은 권한을 다른 사용자에게 부여할 수 없다.

정답 3번

간단한 해설

테이블이나 뷰의 **소유자**는 해당 객체에 대한 권한을 가지며, 이를 다른 사용자에게 허가할 수 있는 기본 주체이다. 권한 취소에는 DROP이 아닌 REVOKE를 사용하고, 일반 사용자도 권한 재부여를 허용받으면 다른 사용자에게 전달할 수 있다.

선지별 해설

① X: 권한의 허가는 GRANT, 권한의 취소는 **REVOKE**로 수행한다. DROP은 테이블·뷰 같은 데이터베이스 객체 자체를 삭제하는 명령으로, 사용자에게 허가한 권한만 취소하는 명령이 아니다.

② X: SELECT·INSERT·DELETE·UPDATE뿐 아니라 **REFERENCES 권한도 부여할 수 있다**.
REFERENCES는 다른 테이블에서 외래키 제약을 정의할 때 해당 테이블이나 열을 참조할 수 있게 하는 권한이다. 데이터를 조회하는 SELECT 권한과는 목적이 다르다.

③ O: SQL 권한 관리의 기본 원칙에서 테이블이나 뷰의 소유자는 해당 객체에 대한 모든 권한을 가진다.
소유자는 자신의 객체에 대해 다른 사용자가 수행할 수 있는 연산을 GRANT로 허가하는 등 권한을 관리한다.
따라서 이 설명이 가장 적절하다.

④ X: 일반 사용자라도 **WITH GRANT OPTION**을 포함하여 권한을 허가받으면 그 권한을 다른 사용자에게 다시 부여할 수 있다. 재부여 범위는 자신이 받은 권한과 부여 옵션의 범위 안으로 제한되며, 그러한 옵션 없이 권한을 받은 경우에는 임의로 전달할 수 없다.

NoteBase

문 36. 릴레이션 R(A, B, C)과 S(D, C)의 튜플이 각각 100개, 70개라고 할 때 관계 대수식을 만족할 것이라 예상되는 최대 결과 개수가 가장 작은 것은? (단, 릴레이션 R과 S에서 A와 D는 기본키이고 S 릴레이션에서 C는 R 릴레이션을 참조하기 위한 외래키이다.)

① $R \bowtie S$

② $\pi_D(S)$

③ $\sigma_{A=3}(R)$

④ $S \times R$

정답 3번

간단한 해설

A는 R의 기본키이므로 A = 3인 튜플은 **최대 1개**이다. 다른 연산은 여러 튜플을 결과로 낼 수 있으므로 기본키의 특정 값을 선택하는 연산의 최대 결과 개수가 가장 작다.

선지별 해설

① 자연 조인: $R \bowtie S$는 공통 속성 C의 값이 같은 튜플을 결합한다. 결과 수는 C의 값 분포에 따라 달라지지만, 각 S 튜플에 대응하는 R 튜플이 하나씩 있는 경우에는 70개를 얻을 수 있다. 이는 주어진 기본키와 외래키 조건을 만족하면서 가능한 경우이므로, 이 연산의 최대 결과 수가 1개로 제한되지는 않는다.

② 프로젝션: $\pi_D(S)$는 S에서 D 속성만 남긴다. 관계 대수의 프로젝션은 중복 튜플을 제거하지만 **D가 기본키이므로 70개 값이 모두 다르다**. 따라서 결과 수는 70개이다.

③ 선택: $\sigma_{A=3}(R)$는 R에서 A = 3인 튜플만 남긴다. A는 기본키이므로 같은 A값을 가진 튜플이 둘 이상 존재할 수 없다. A = 3인 튜플이 없으면 0개, 있으면 1개이므로 **최대 결과 수는 1개**이다.

④ 카티션 곱: $S \times R$는 S의 각 튜플을 R의 모든 튜플과 짝짓는다. 별도의 결합 조건으로 걸러 내지 않으므로 결과 수는 $70 \times 100 = 7000$개이다.

따라서 ①은 70개인 결과도 가능하고, ②는 70개, ④는 7000개인 데 비해 ③은 최대 1개이므로 ③의 최대 결과 개수가 가장 작다.

NoteBase

문 37. 관계형 DBMS의 질의 최적화에 대한 설명으로 적절한 것을 모두 고른 것은?

가. 사용자가 입력한 SELECT 문은 관계 대수식의 동등성을 이용하여 관계 연산자들을 재배열할 수 있다.
나. 질의 트리에서 SELECT 문의 FROM 절에 포함된 테이블(들)은 질의 트리의 단말 노드에 저장한다.
다. 경험 기반 질의 최적화는 각 질의 트리의 비용을 예상하기 위해서 비용 기반 모델을 사용한다.
라. 질의 수행은 질의 트리의 루트 노드에서 시작하여 단말 노드에서 종료한다.

① 가, 나
② 가, 다
③ 나, 라
④ 다, 라

정답 1번

간단한 해설

질의는 **결과를 보존하는 관계대수 동치 규칙**으로 연산 순서를 바꿀 수 있다. 질의 트리에서는 FROM 절의 테이블이 **입력 자료를 제공하는 단말 노드**가 된다.

NoteBase

보기별 해설

질의 최적화는 같은 결과를 내는 여러 실행 방법 중 처리량이나 입출력 비용을 줄일 수 있는 방법을 찾는 과정이다. 관계대수식과 질의 트리는 수행할 연산 및 입력 관계를 나타내는 데 사용한다.

가. O: 관계대수의 동치 규칙을 적용하면 **질의 결과를 바꾸지 않고** 연산자들을 재배열할 수 있다. 예를 들어 특정 테이블에만 관련된 선택 조건을 조인보다 먼저 적용하면 이후 처리할 행 수를 줄일 수 있다. 다만 모든 연산 순서를 아무 조건 없이 교환할 수 있다는 뜻은 아니다.

나. O: FROM 절에 나타나는 기본 테이블은 질의의 입력 관계이므로 질의 트리의 **단말 노드**로 표현한다. 선택·프로젝션·조인 등의 관계 연산은 그 입력을 처리하는 내부 노드에 놓인다.

다. X: 각 실행 계획의 비용을 모형으로 추정하여 비교하는 것은 **비용 기반 최적화**의 특징이다. 경험 기반 최적화는 선택 연산을 가능한 한 먼저 적용하는 것처럼 일반적으로 유리하다고 알려진 규칙으로 질의 트리를 변환한다.

라. X: 질의 트리의 연산 결과는 **단말 노드에서 루트 노드 방향**으로 만들어진다. 입력 테이블을 읽고 하위 연산의 결과를 상위 연산에 전달하여 마지막에 루트의 최종 결과를 얻으므로, 제시된 방향은 거꾸로이다.

적절한 보기는 가와 나이므로 ①과 일치한다.

NoteBase

문 38. 분산 데이터베이스 특성을 나타내는 CAP 이론에 기반하여 관계형 DBMS가 갖는 특징으로 가장 적절한 것은?

① 가용성과 투명성
② 일관성과 분할 감내
③ 일관성과 가용성
④ 투명성와 분할 감내

정답 3번

간단한 해설

전통적인 관계형 DBMS를 CAP 관점에서 단순화하여 분류하면 **일관성(C)과 가용성(A)**을 갖춘 CA에 해당한다. 다만 이는 네트워크 분할 상황까지 두 속성을 동시에 보장한다는 뜻은 아니다.

개념 구분

CAP는 **일관성(Consistency), 가용성(Availability), 분할 감내(Partition tolerance)**를 뜻한다. 일관성은 읽기가 완료된 최신 쓰기를 반영하는 등 데이터가 하나의 일관된 상태로 관찰되는 성질이고, 가용성은 정상 동작 중인 노드가 받은 요청에 응답하는 성질이다. 분할 감내는 노드 사이 통신이 끊어져도 시스템이 동작을 계속하는 성질이다.

이 문항은 전통적인 관계형 DBMS를 CA로 설명하는 분류를 사용한다. 네트워크 분할을 견디면서 C와 A를 모두 보장하는 체계로 보는 것이 아니라, **통신 분할이 없는 환경에서 일관성과 가용성을 제공**하는 성격에 초점을 둔다.

'일관성과 분할 감내'는 CP 조합이다. 통신 분할 중에도 일관성을 유지하려면 일부 요청을 기다리게 하거나 거부하여 가용성을 제한할 수 있다. '투명성'은 분산 시스템에서 사용될 수 있는 별도 개념이지만 **CAP의 세 구성 요소에는 포함되지 않는다**.

관계형이라는 데이터 모델만으로 실제 모든 DBMS의 CAP 특성이 고정되는 것은 아니며, 복제·분산 구성과 장애 처리 방식에 따라 달라진다. 이 문항의 전통적 CA 분류에 해당하는 조합은 일관성과 가용성이므로 ③과 연결된다.

NoteBase

문 39. 분산 데이터베이스에서 제공하는 투명성에 대한 설명으로 가장 적절하지 않은 것은?

① 단편화 투명성(fragmentation transparency)은 하나의 릴레이션이 수평 단편화 및 수직 단편화를 통해 분산되어 있지만 하나의 릴레이션으로 취급하게 되는 것을 말한다.

② 위치 투명성(location transparency)은 동일한 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 한 곳에 데이터가 존재하는 것처럼 사용하는 것을 말한다.

③ 장애 투명성(failure transparency)은 특정 사이트에 고장이 발생하여도 시스템은 작업을 중지하지 않고 계속해서 작업을 수행할 수 있어야 하는 것을 말한다.

④ 병행 투명성(concurrency transparency)은 다수의 트랜잭션들이 동시에 수행되더라도 그 트랜잭션의 결과는 일관성을 유지해야 하는 것을 말한다.

정답 2번

간단한 해설

여러 곳에 저장된 동일 데이터의 복제본을 하나의 데이터처럼 사용하게 하는 것은 **중복 투명성**이다. **위치 투명성**은 사용자가 데이터의 실제 저장 사이트를 몰라도 접근할 수 있게 한다.

NoteBase

선지별 해설

분산 데이터베이스의 투명성은 사용자가 데이터의 분산 방식이나 시스템 내부 처리를 일일이 알지 않아도 하나의 논리적 데이터베이스처럼 이용하도록 하는 성질이다. 무엇을 사용자에게 드러내지 않는지에 따라 투명성의 종류를 구별한다.

① O: 수평 단편화는 릴레이션을 행 단위로, 수직 단편화는 속성인 열 단위로 나누는 방식이다. 단편화 투명성은 사용자가 **단편의 존재와 분할 방식을 의식하지 않고** 원래의 논리적 릴레이션에 질의하도록 한다. 필요한 단편의 선택과 결합은 시스템이 처리한다.

② X: 동일한 데이터의 여러 복제본이 있어도 하나의 데이터처럼 사용하게 하는 것은 **중복 투명성 또는 복제 투명성**이다. 위치 투명성은 데이터가 어느 사이트에 저장되어 있는지를 사용자가 알거나 지정할 필요가 없다는 뜻이다. 복제본의 존재와 저장 위치를 혼동한 ②가 적절하지 않다.

③ O: 장애 투명성은 특정 사이트나 구성 요소에 장애가 생겨도 장애 처리의 세부 사항을 사용자에게 드러내지 않고 작업을 지속하도록 하는 성질이다. 시스템은 복제본 활용이나 회복 절차 등을 통해 **부분 장애의 영향을 감추고 서비스 연속성을 유지**한다.

④ O: 병행 투명성은 여러 트랜잭션이 동시에 접근하더라도 서로의 간섭 때문에 데이터의 일관성이 깨지지 않게 하는 성질이다. 시스템이 동시성 제어를 수행하여 사용자가 다른 트랜잭션과의 실행 조정을 직접 처리하지 않아도 일관된 결과를 얻도록 한다.

NoteBase

문 40. 트랜잭션 실행 과정에서 오손 읽기 문제와 반복할 수 없는 읽기 문제가 발생하지 않는 고립성 수준을 모두 고른 것은?

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

① 가, 나
② 가, 라
③ 나, 다
④ 다, 라

정답 2번

간단한 해설

오손 읽기를 막는 것만으로는 같은 데이터를 다시 읽을 때 값이 달라지는 현상까지 막을 수 없다. **REPEATABLE READ와 SERIALIZABLE**은 커밋되지 않은 변경을 읽지 않으며, 한 트랜잭션 안에서 같은 데이터를 반복하여 읽을 때의 일관성도 보장한다.

NoteBase

보기별 해설

오손 읽기는 다른 트랜잭션이 아직 커밋하지 않은 변경을 읽는 현상이다. 반복할 수 없는 읽기는 같은 트랜잭션이 같은 데이터를 다시 읽었을 때 다른 트랜잭션의 갱신·커밋 때문에 값이 달라지는 현상이다. 두 현상을 모두 방지하는지를 기준으로 각 수준을 구분한다.

가. SERIALIZABLE: 두 현상을 모두 방지한다. 여러 트랜잭션이 동시에 실행되더라도 결과가 어떤 직렬 실행과 같도록 보장하는 수준이므로 **오손 읽기와 반복할 수 없는 읽기 모두** 발생하지 않는다.

나. READ COMMITTED: 오손 읽기는 방지하지만 반복할 수 없는 읽기는 허용한다. 커밋된 값만 읽더라도 첫 번째 읽기와 두 번째 읽기 사이에 다른 트랜잭션이 값을 바꾸고 커밋하면 나중 읽기에서 바뀐 값을 볼 수 있다.

다. READ UNCOMMITTED: 두 현상 모두 허용한다. 다른 트랜잭션이 아직 커밋하지 않은 변경도 읽을 수 있어 오손 읽기가 발생할 수 있고, 동일 데이터를 반복해서 읽을 때 값이 유지된다는 보장도 없다.

라. REPEATABLE READ: 두 현상을 모두 방지한다. 커밋되지 않은 변경을 읽지 않을 뿐 아니라, 같은 트랜잭션에서 이미 읽은 데이터를 다시 읽을 때 **다른 트랜잭션의 갱신 때문에 값이 달라지지 않도록** 보장한다.

두 문제를 모두 방지하는 수준은 가와 라이므로 선택지 ②와 일치한다.

NoteBase