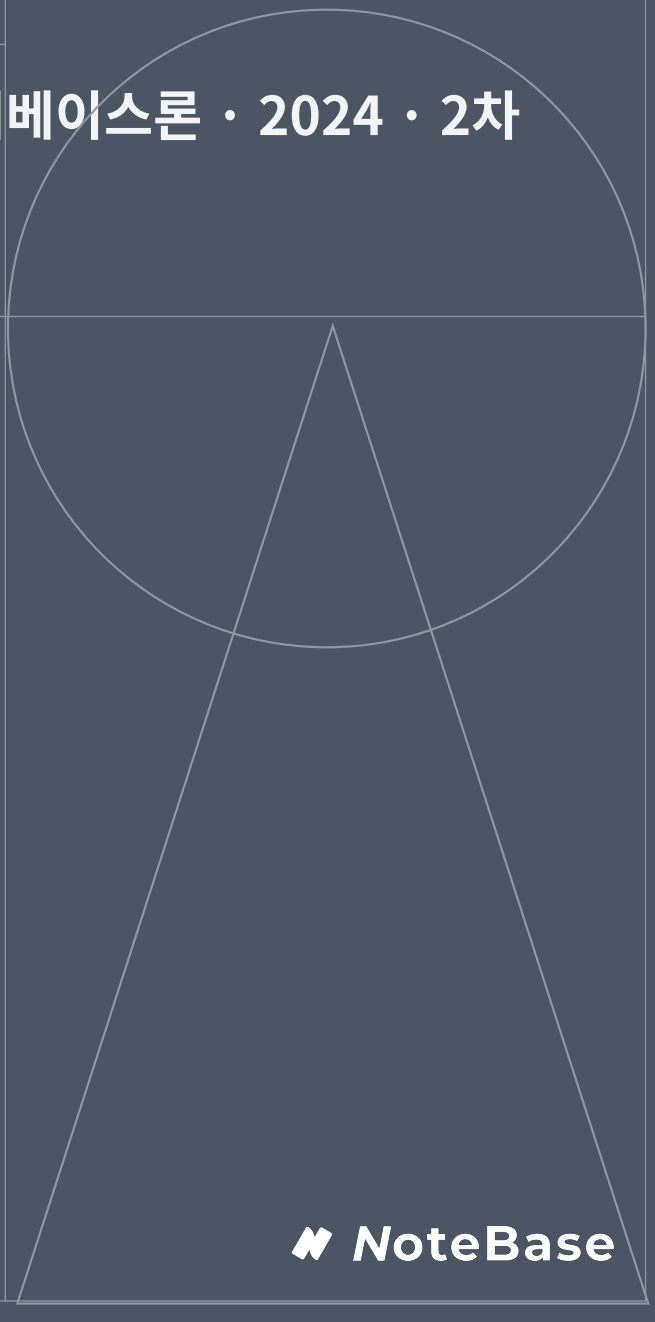
기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2024 · 2차



NoteBase

기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2024 · 2차

공무원/국가직/7급 · 데이터베이스론 · 2024 · 2차

문 1. 데이터베이스의 일반적인 특징으로 옳지 않은 것은?

① 데이터의 실시간 처리가 가능하다.

② 여러 사용자가 동시에 이용할 수 있다.

③ 사용자는 데이터의 저장 주소나 위치를 통해 데이터를 참조한다.

④ 동적인 특징이 있어 데이터를 계속 삽입, 삭제, 갱신하여 현재의 정확한 데이터를 유지한다.

정답 3번

출제 포인트

이 문제는 데이터베이스의 일반적 특징과 파일 처리 방식의 차이를 구분하는 문제이다.

핵심 개념

데이터베이스는 공용 데이터의 통합 저장소로서 실시간 처리, 동시 공유, 계속적 갱신이 가능하다.
데이터 독립성은 사용자가 물리적 저장 주소를 몰라도 논리적 구조를 통해 데이터를 사용할 수 있는 성질이다.

선지별 해설

① O
실시간 처리 가능성은 데이터베이스의 대표적 특징이다. 데이터가 갱신되면 그 결과를 즉시 업무에 반영할 수 있다.
② O
데이터베이스는 공용 데이터의 집합이므로 여러 사용자가 동시에 접근하여 사용할 수 있다.
③ X
틀린 부분은 "저장 주소나 위치를 통해"이다. 데이터베이스 사용자는 물리적 저장 위치를 직접 지정해 접근하는 것이 아니라 스키마와 질의 같은 논리적 구조를 통해 데이터를 참조한다.
올바르게는 "사용자는 데이터의 저장 주소나 위치를 몰라도 논리적 구조를 통해 데이터를 참조한다"이다.
④ O
데이터베이스는 동적인 특성을 가지므로 삽입, 삭제, 갱신이 계속 일어나며 최신의 정확한 데이터를 유지할 수 있다.

NoteBase

오답 포인트

③은 파일 처리 방식의 접근 방법을 데이터베이스의 특징처럼 바꾸어 놓은 함정이다. 저장 주소와 위치를 직접 알아야 한다는 표현이 보이면 데이터 독립성에 어긋나는지 먼저 확인하게 된다.

다시 나오면 이렇게 판단

데이터베이스 특징 문제는 공용성, 통합성, 운영성, 데이터 독립성을 먼저 떠올리면 판단이 빨라진다. 선지에 물리적 저장 위치를 직접 안다고 나오면 데이터베이스보다 파일 처리 방식에 가깝다고 판단한다.

암기 포인트

데이터베이스는 여러 사용자가 함께 쓰고 계속 갱신하지만, 사용자는 저장 위치까지 알 필요는 없다는 점이 핵심이다.

NoteBase

문 2. 데이터베이스 관리자(DBA)의 주요 업무와 관련된 설명으로 옳지 않은 것은?

① 데이터 사전과 같은 시스템 데이터베이스를 관리한다.

② DML을 사용하여 데이터 접근 권한을 사용자에게 부여한다.

③ 데이터베이스를 물리적으로 저장하기 위한 레코드 구조를 설계한다.

④ 사용자의 요구사항을 분석하여 데이터베이스의 구성요소를 결정한다.

정답 2번

출제 포인트

DBA의 업무 범위와 DML, DCL의 역할 차이를 구분하는 문제이다.

핵심 개념

DBA: 데이터베이스의 전체 구조, 저장 구조, 보안, 무결성, 권한, 백업과 회복을 총괄 관리하는 사람이다. / 결정 차이축: 데이터 자체를 조작하는지, 관리 권한을 통제하는지 / 키워드: 관리, 권한, 저장 구조
DML: SELECT, INSERT, UPDATE, DELETE처럼 데이터를 조회, 삽입, 수정, 삭제하는 언어이다. / 결정 차이축: 데이터 내용 조작 / 키워드: 데이터 조작
DCL: GRANT, REVOKE처럼 사용자 권한을 부여하거나 회수하는 언어이다. / 결정 차이축: 권한 통제 / 키워드: 권한 부여

선지별 해설

① O
데이터 사전은 DBMS가 관리하는 대표적 시스템 데이터베이스이므로 DBA의 관리 대상에 해당한다.

② X
틀린 부분은 "DML을 사용하여 데이터 접근 권한을 사용자에게 부여한다"는 표현이다.
권한 부여는 DML이 아니라 DCL의 GRANT 문으로 수행한다. DML은 데이터 내용 자체를 조작하는 언어이다.
틀린 표현 → 올바르게는 "DCL을 사용하여 데이터 접근 권한을 사용자에게 부여한다"이다.

③ O
DBA는 데이터베이스를 물리적으로 저장하기 위한 저장 구조와 접근 경로를 설계하고 관리하므로 레코드 구조와 저장 구조를 다루는 업무와 관련된다.

④ O
사용자의 요구사항을 반영하여 데이터베이스의 스키마와 구성 요소를 정하는 일은 데이터베이스 설계와 관리의 핵심 업무와 연결된다.

NoteBase

오답 포인트

DML과 DCL은 모두 SQL의 하위 범주이므로 이름이 비슷해 혼동하기 쉽다. 이 문항은 ②의 "권한 부여"라는 표현을 보고 바로 DCL을 떠올릴 수 있는지를 묻는 함정이다.

다시 나오면 이렇게 판단

선지에 권한, 보안, GRANT, REVOKE가 보이면 DCL인지부터 확인한다. 조회, 삽입, 수정, 삭제처럼 데이터 내용 자체를 바꾸는 작업이 나오면 DML로 판단한다. DBA 관련 선지는 권한 관리, 저장 구조 관리, 시스템 카탈로그 관리가 보이면 맞는 설명일 가능성이 크다.

암기 포인트

DML은 데이터 조작, DCL은 권한 통제이며 권한 부여는 GRANT이다.

NoteBase

문 3. 관계형 데이터베이스에 대한 설명으로 옳지 않은 것은?

① 릴레이션에서 속성의 순서는 중요하지 않지만, 투플의 순서는 중요하다.

② 제2정규형에서 기본키에 대해 이행적 종속성을 제거하면 제3정규형이 된다.

③ 각 속성은 특정 도메인에서 값을 가져야 하며, 속성 간 도메인은 다를 수 있다.

④ 모든 투플은 기본키에 의해 고유하게 식별되며, 기본키는 릴레이션 내에서 유일성이 보장된다.

정답 1번

출제 포인트

이 문제는 관계형 데이터베이스의 기본 성질과 정규화 단계, 특히 릴레이션의 순서 무의미성과 제3정규형의 조건을 구분하는 문제이다.

핵심 개념

릴레이션은 속성과 튜플의 집합으로 보므로 속성의 순서와 튜플의 순서는 모두 의미가 없다.
제2정규형은 부분 함수 종속을 제거한 상태이고, 제3정규형은 이행적 종속을 제거한 상태이다.
도메인은 속성이 가질 수 있는 값의 범위이므로 속성마다 서로 다른 도메인을 가질 수 있다.

선지별 해설

① X: 릴레이션에서는 속성의 순서뿐 아니라 튜플의 순서도 중요하지 않다. 틀린 부분은 "튜플의 순서는 중요하다"이다. 올바르게는 "튜플의 순서도 중요하지 않다"이다.
② O: 제2정규형에서 기본키에 대한 이행적 종속을 제거하면 제3정규형이 된다. 정규화 단계의 핵심 구분을 제대로 설명한 선지이다.
③ O: 각 속성은 정해진 도메인에서 값을 가져야 하며, 속성별 의미가 다르면 도메인도 서로 다를 수 있다. 예를 들어 학번과 이름은 같은 도메인을 쓰지 않는다.
④ O: 기본키는 릴레이션 내에서 튜플을 유일하게 식별하는 속성 또는 속성 집합이므로 유일성이 보장된다.

오답 포인트

①은 속성의 순서만 무의미하고 튜플의 순서는 의미가 있다고 뒤집어 놓은 함정이다. 관계형 모델은 집합 개념을 바탕으로 하므로 행과 열의 나열 순서에 의미를 두지 않는다는 점에서 자주 혼동이 발생한다.

다시 나오면 이렇게 판단

관계형 데이터베이스의 기본 성질을 묻는 선지는 먼저 집합적 성질인지, 정규화 단계인지, 키의 성질인지로 나누어 본다. 순서가 중요하다고 하면 관계형 모델의 기본 성질에 어긋나는지 먼저 점검하면 된다.

NoteBase

암기 포인트

릴레이션은 순서 무의미, 2NF는 부분 종속 제거, 3NF는 이행 종속 제거이다.

NoteBase

문 4. 외부 스키마에 대한 설명으로 옳지 않은 것은?

① 서브 스키마라고도 한다.

② 하나의 데이터베이스에 여러 개가 존재할 수 있다.

③ 데이터베이스가 외부 저장장치에 저장되는 방법을 정의한다.

④ 개별 사용자의 시각에서 본 데이터베이스의 논리적 구조이다.

정답 3번

출제 포인트

이 문제는 외부 스키마와 내부 스키마의 역할 차이, 그리고 외부 스키마의 사용자 관점 정의를 구별하는 문제이다.

핵심 개념

외부 스키마는 개별 사용자나 응용프로그램의 입장에서 필요한 데이터만 보이도록 정의한 사용자 관점의 논리적 구조이다. 키워드는 서브 스키마, 사용자 뷰이다.
개념 스키마는 데이터베이스 전체의 논리적 구조를 통합적으로 정의한 전체 관점의 스키마이다. 키워드는 전체 데이터베이스, 논리적 구조이다.
내부 스키마는 데이터가 저장장치에 어떤 구조와 방식으로 저장되는지를 정의한 물리적 구조이다. 키워드는 저장 방식, 물리적 구조이다.

NoteBase

선지별 해설

① O
외부 스키마는 사용자별로 필요한 부분만 본다는 뜻에서 서브 스키마라고도 한다. 사용자 뷰를 따로 정의한 것이므로 맞는 설명이다.

② O
하나의 데이터베이스에는 사용자나 응용 목적이 다를 수 있으므로 외부 스키마가 여러 개 존재할 수 있다.
사용자별 뷰가 각각 다를 수 있다는 점이 핵심이다.

③ X
틀린 부분은 "외부 저장장치에 저장되는 방법"이다.
실제로 외부 저장장치에 저장되는 방법을 정의하는 것은 내부 스키마이다. 외부 스키마는 저장 방식이 아니라 개별 사용자가 보는 논리적 뷰를 정의한다.

④ O
외부 스키마는 개별 사용자의 시각에서 본 데이터베이스의 논리적 구조를 의미한다. 전체 데이터베이스가 아니라 사용자별 관점이라는 점에서 맞는 설명이다.

오답 포인트

외부 스키마와 내부 스키마를 모두 "스키마"로만 기억하면 ③처럼 사용자 관점과 저장 관점을 뒤바꾸기 쉽다.
특히 "저장장치", "저장 방법", "물리적 구조"라는 표현이 나오면 내부 스키마 쪽 개념인지 먼저 점검해야 한다.

다시 나오면 이렇게 판단

선지에 사용자별 화면, 뷰, 서브 스키마, 개별 사용자 관점이 나오면 외부 스키마로 판단한다. 반대로 저장장치, 접근 경로, 레코드 배치, 물리적 저장 방식이 나오면 내부 스키마로 판단한다.

암기 포인트

외부 스키마는 사용자 뷰, 개념 스키마는 전체 논리, 내부 스키마는 물리 저장이다.

NoteBase

문 5. 다음과 같은 특징에 해당하는 키(key)는?

◯ 릴레이션에서 튜플을 식별할 수 있는 유일성(uniqueness)을 만족시키는 속성의 집합
◯ 유일성을 갖는 속성의 최소성(minimality)은 만족시키지 못함

① 슈퍼키(super key)
② 외래키(foreign key)
③ 기본키(primary key)
④ 후보키(candidate key)

정답 1번

출제 포인트

이 문제는 슈퍼키, 후보키, 기본키, 외래키의 정의를 구분하고 특히 유일성과 최소성의 관계를 판단하는 문제이다.

핵심 개념

슈퍼키는 릴레이션의 각 튜플을 유일하게 식별할 수 있는 속성 집합이며 유일성만 만족하면 된다.
후보키는 튜플을 유일하게 식별하면서도 더 이상 불필요한 속성을 포함하지 않는 최소성을 만족하는 키이다.
기본키는 여러 후보키 중 대표로 선택한 키이므로 후보키와 마찬가지로 유일성과 최소성을 만족한다.
외래키는 다른 릴레이션의 기본키를 참조하는 속성으로, 튜플의 고유 식별 자체보다는 릴레이션 간 참조 관계를 표현한다.

NoteBase

선지별 해설

① O
슈퍼키는 릴레이션에서 튜플을 식별할 수 있는 유일성을 만족하는 속성 집합이다. 불필요한 속성이 더 들어 있어도 되므로 최소성은 만족하지 못할 수 있다. 문제의 설명과 정확히 일치한다.

② X
틀린 부분은 "외래키"이다. 외래키는 다른 릴레이션의 기본키를 참조하는 키이지, 현재 릴레이션의 각 튜플을 유일하게 식별하는 키가 아니다. 올바르게는 "다른 릴레이션의 기본키를 참조하여 관계를 나타내는 속성"이다.

③ X
틀린 부분은 "기본키"이다. 기본키는 후보키 중 하나를 선택한 키이므로 유일성과 최소성을 모두 만족한다. 올바르게는 "기본키는 유일성과 최소성을 만족하는 후보키 중 대표 키"이다.

④ X
틀린 부분은 "후보키"이다. 후보키는 유일성과 최소성을 모두 만족해야 한다. 최소성을 만족시키지 못하는 키는 후보키가 아니라 슈퍼키이다. 올바르게는 "후보키는 유일성과 최소성을 모두 만족하는 속성 집합"이다.

오답 포인트

후보키와 기본키를 구분하지 못하거나, 슈퍼키도 최소성을 만족해야 한다고 착각하면 ③이나 ④로 흔들리기 쉽다. 이 문항의 함정은 "유일성은 만족하지만 최소성은 만족하지 못함"이라는 표현이 슈퍼키의 특징이라는 점을 놓치게 만드는 데 있다.

다시 나오면 이렇게 판단

먼저 튜플을 유일하게 식별하는지로 키 여부를 본다. 그다음 불필요한 속성이 섞여 있으면 슈퍼키, 최소성까지 만족하면 후보키, 그 후보키 중 대표로 선택된 것이 기본키, 다른 릴레이션의 키를 참조하면 외래키로 판단한다.

암기 포인트

슈퍼키는 유일성, 후보키는 유일성+최소성, 기본키는 선택된 후보키, 외래키는 참조키이다.

NoteBase

문 6. 두 릴레이션 R과 S의 차수(degree)와 카디널리티(cardinality)가 다음과 같다. R과 S에 이름과 도메인이 동일한 속성은 3개이고, 이들 공통 속성의 값이 모두 일치하는 투플의 조합이 5개일 때, 다음 설명 중 옳은 것은? (단, R과 S 사이에는 3개의 공통 속성 외에 도메인이 동일한 속성은 존재하지 않는다)

릴레이션	차수	카디널리티
R	5	5
S	10	10

① S와 R의 디비전(division), 즉 $S \div R$의 결과 릴레이션의 차수는 5이다.

② S와 R의 차집합(difference), 즉 $S - R$의 결과 릴레이션의 카디널리티는 5이다.

③ R과 S의 자연 조인(natural join), 즉 $R \bowtie_N S$의 결과 릴레이션의 차수는 12이다.

④ R과 S의 카티션 프로덕트(Cartesian product), 즉 $R \times S$의 결과 릴레이션의 차수는 12이고, 카디널리티는 50이다.

정답 3번

출제 포인트

자연 조인, 카티션 프로덕트, 차집합, 디비전의 성립 조건과 차수·카디널리티 계산 규칙을 구분하는 문제이다.

핵심 개념

디비전은 제수 릴레이션의 속성이 피제수 릴레이션에 모두 포함될 때만 정의되며 결과 차수는 피제수 차수에서 제수 차수를 뺀 값이다.
차집합은 두 릴레이션이 합집합 가능할 때만 정의되며, 차수와 대응 속성의 도메인이 같아야 한다.
자연 조인은 공통 속성의 값이 같은 튜플을 결합하고 공통 속성은 한 번만 남기므로 결과 차수는 두 차수의 합에서 공통 속성 수를 뺀 값이다.
카티션 프로덕트는 가능한 모든 튜플 쌍을 결합하므로 결과 차수는 두 차수의 합이고 카디널리티는 두 카디널리티의 곱이다.

NoteBase

선지별 해설

① X
'S ÷ R' 자체가 성립하지 않는다. 디비전은 제수 R의 모든 속성이 피제수 S에 포함되어야 하는데, 공통 속성은 3개뿐이므로 R의 5개 속성이 S에 모두 들어 있다고 볼 수 없다. 틀린 표현은 'S ÷ R'이고, 올바르게는 'R의 속성이 S에 모두 포함될 때만 S ÷ R이 정의되며 그때 결과 차수는 10-5=5이다.'이다.

② X
차집합 S-R은 두 릴레이션이 합집합 가능할 때만 정의된다. 그런데 R의 차수는 5이고 S의 차수는 10이므로 합집합 가능하지 않다. 틀린 표현은 'S-R의 결과 릴레이션'이고, 올바르게는 'R과 S는 차수가 달라 S-R은 정의되지 않는다.'이다.

③ O
자연 조인은 공통 속성 3개를 한 번만 남기므로 결과 차수는 5+10-3=12이다. 또한 공통 속성 값이 모두 일치하는 튜플 조합이 5개이므로 결과 카디널리티는 5가 되지만, 선지는 차수만 물었으므로 옳다.

④ X
카티션 프로덕트의 카디널리티는 모든 튜플 쌍의 수이므로 5×10=50이 맞다. 그러나 차수는 속성을 단순히 이어 붙인 값이므로 5+10=15이다. 틀린 표현은 '차수는 12'이고, 올바르게는 'R × S의 차수는 15이고 카디널리티는 50이다.'이다.

오답 포인트

자연 조인에서만 공통 속성을 한 번으로 줄인다는 점을 카티션 프로덕트에도 그대로 적용하면 ④를 고르기 쉽다. 또 디비전과 차집합은 계산식보다 먼저 성립 조건을 봐야 하는데, 이 조건을 건너뛰면 ①과 ②를 숫자만으로 판단하게 된다.

다시 나오면 이렇게 판단

관계대수 문항은 먼저 연산이 정의되는지부터 본다. 디비전은 제수 속성의 포함 관계를, 차집합은 합집합 가능 여부를, 자연 조인은 공통 속성 수를, 카티션 프로덕트는 차수 합과 카디널리티 곱을 순서대로 확인하면 된다.

암기 포인트

자연 조인 차수는 합에서 공통 속성 수를 빼고, 카티션 프로덕트 차수는 그냥 합이다. 차집합은 합집합 가능, 디비전은 속성 포함이 전제이다.

NoteBase

문 7. 다음 릴레이션에서 속성(A, B, C, D) 간 함수 종속을 화살표(→)로 표현할 때, 성립하는 것만을 모두 고르면? (단, 아래 투플을 기준으로 판단한다)

A	B	C	D
1	3	7	1
2	5	7	2
3	3	7	1
4	3	4	3
5	9	7	2

ㄱ. A →B

ㄴ. B →C

ㄷ. C →D

ㄹ. D →C

① ㄱ, ㄴ

② ㄱ, ㄷ

③ ㄱ, ㄹ

④ ㄴ, ㄹ

정답 3번

출제 포인트

동일한 결정자 값이 반복될 때 종속 속성 값이 항상 하나로 유지되는지로 함수 종속 성립 여부를 판정하는 문제이다.

자료 읽는 순서

1. 화살표 왼쪽 속성값이 중복되는 행을 찾는다.
2. 그 중복값에 대해 화살표 오른쪽 속성값이 하나인지 여러 개인지 본다.
3. ㄱ~ㄹ의 참거짓을 확정한 뒤 번호 조합과 연결한다.

NoteBase

핵심 근거

A값은 1, 2, 3, 4, 5로 모두 달라 같은 A에 서로 다른 B가 나타나는 반례가 없다. 따라서 ㄱ의 A→B는 성립한다.
B=3일 때 C가 7과 4로 갈리므로 ㄴ의 B→C는 성립하지 않는다.
C=7일 때 D가 1과 2로 갈리므로 ㄷ의 C→D는 성립하지 않는다.
D=1이면 C=7, D=2이면 C=7, D=3이면 C=4만 나타나므로 ㄹ의 D→C는 성립한다.

선지별 해설

① X
1번은 ㄱ, ㄴ의 조합이다. ㄱ은 O이지만 ㄴ은 B=3에서 C가 7과 4로 달라 X이다. 틀린 표현은 'B→C가 성립한다'이다. 올바르게는 'B=3이 여러 C값을 가지므로 B→C는 성립하지 않는다'이다.

② X
2번은 ㄱ, ㄷ의 조합이다. ㄱ은 O이지만 ㄷ은 C=7에서 D가 1과 2로 달라 X이다. 틀린 표현은 'C→D가 성립한다'이다. 올바르게는 'C=7이 여러 D값을 가지므로 C→D는 성립하지 않는다'이다.

③ O
3번은 ㄱ, ㄹ의 조합이다. ㄱ은 A값이 모두 달라 반례가 없으므로 O이다. ㄹ은 D값이 같을 때 C값이 항상 하나로 고정되므로 O이다. 따라서 성립하는 것만 모은 조합은 3번이다.

④ X
4번은 ㄴ, ㄹ의 조합이다. ㄹ은 O이지만 ㄴ은 B=3에서 반례가 생겨 X이다. 틀린 표현은 'B→C가 성립한다'이다. 올바르게는 'B가 3일 때 C가 7과 4로 나뉘므로 B→C는 성립하지 않는다'이다.

오답 포인트

속성값이 모두 다른 경우를 보고 판단을 멈추거나, 일부 행만 보고 함수 종속을 성급히 인정하면 틀리기 쉽다.
특히 B=3과 C=7처럼 반복되는 값을 끝까지 모아 보지 않으면 ㄴ과 ㄷ을 참으로 착각하기 쉽다.

다시 나오면 이렇게 판단

함수 종속은 같은 왼쪽 값이 나오는 행끼리만 비교해서 오른쪽 값이 하나로 고정되는지 본다. 같은 왼쪽 값에 오른쪽 값이 두 개라도 나오면 그 종속은 즉시 깨진다.

암기 포인트

함수 종속 판단은 '왼쪽 중복 찾기, 오른쪽 분기 확인'으로 정리된다.

NoteBase

문 8. 다음과 같은 키 값의 삽입 순서에 따라 B-트리를 만들 때, 루트(root) 노드에 저장되는 키 값은? (단, 한 노드에 최대로 저장되는 키의 개수는 2이다)

9, 4, 1, 6, 2, 11, 10, 5, 12

① 4
② 6
③ 9
④ 11

정답 3번

출제 포인트

이 문제는 한 노드에 최대 2개의 키를 저장하는 B-트리에서 삽입 순서에 따른 노드 분할과 중간 키 승격 과정을 추적하는 문제이다.

올바른 순서

1. 9, 4를 삽입하면 루트는 [4, 9]가 된다.
2. 1을 삽입하면 [1, 4, 9]가 되어 분할되고, 중간 키 4가 승격되어 루트는 [4], 자식은 [1], [9]가 된다.
3. 6, 2, 11, 10, 5를 차례로 삽입하면 루트는 [4, 9], 자식은 [1, 2], [5, 6], [10, 11] 상태가 된다.
4. 12를 삽입하면 [10, 11, 12]가 분할되어 11이 부모로 올라가고, 기존 루트는 [4, 9, 11]이 된다.
5. 루트 [4, 9, 11]도 최대 키 수 2를 초과하므로 다시 분할되고, 중간 키 9가 새 루트가 된다.

왜 이 순서인가: B-트리에서는 넘친 노드의 중간 키가 상위 노드로 승격되므로, 마지막 루트 분할에서 가운데 있던 9가 최종 루트 값을 결정한다.

선지별 해설

① X이다. 4는 처음 [1, 4, 9]가 분할될 때 한 번 루트로 올라온 값일 뿐이다. 틀린 표현은 '최종 루트가 4이다'이며, 올바르게는 마지막 루트 분할 뒤 중간 키 9가 새 루트가 된다.
② X이다. 6은 중간 자식 노드 [5, 6]의 리프 키로 남는다. 틀린 표현은 '6이 루트로 승격된다'이며, 올바르게는 6은 어떤 분할에서도 중간 키가 되지 않아 루트가 되지 않는다.
③ O이다. 12를 삽입한 뒤 기존 루트가 [4, 9, 11]로 넘치고, 이 루트를 분할할 때 중간 키 9가 새 루트로 승격된다.
④ X이다. 11은 [10, 11, 12]가 분할될 때 부모로 한 번 올라가지만, 그 결과 기존 루트가 [4, 9, 11]이 되어 다시 분할된다. 틀린 표현은 '11이 최종 루트이다'이며, 올바르게는 11은 새 루트의 오른쪽 자식 노드 키로 남는다.

NoteBase

오답 포인트

분할 과정에서 부모로 한 번 승격된 키를 곧바로 최종 루트로 착각하기 쉽다. 이 문항에서는 11이 먼저 부모로 올라오므로 ④가 그럴듯해 보이지만, 바로 이어서 루트 자체가 다시 분할된다는 점을 놓치면 오답이 된다.

다시 나오면 이렇게 판단

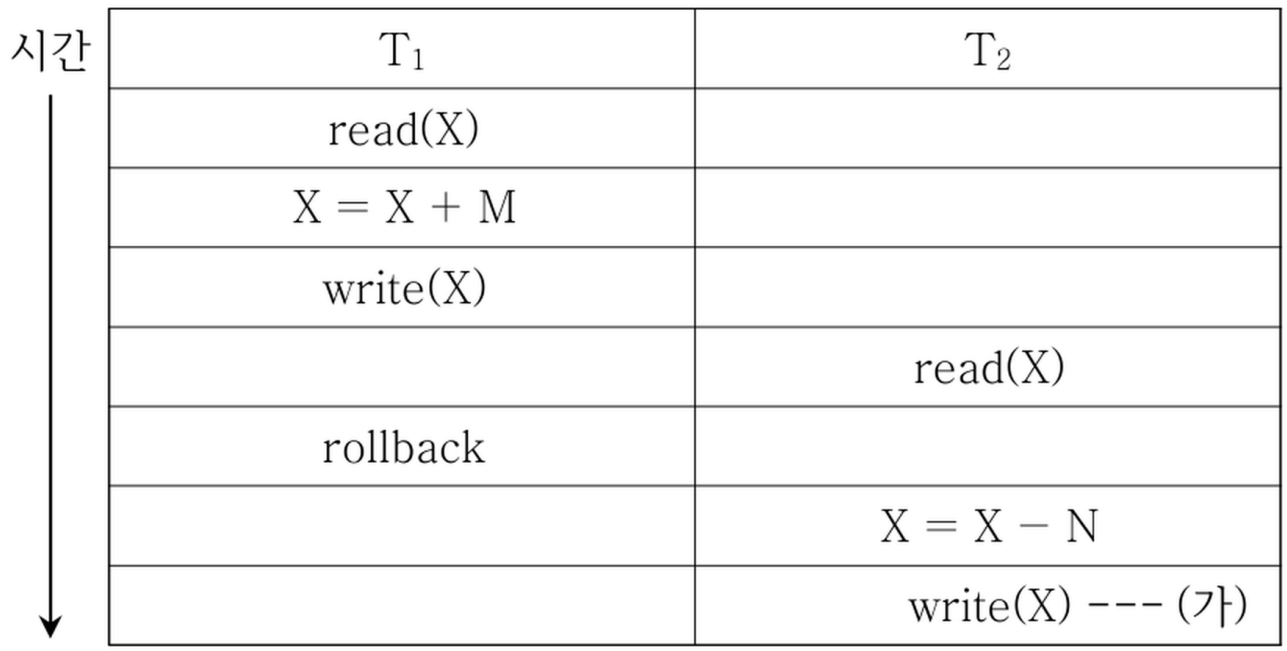
B-트리 삽입 문제는 각 단계마다 노드가 최대 키 수를 넘는지만 확인하면 된다. 넘치면 그 노드의 중간 키가 부모로 올라가고, 마지막에 루트가 넘치면 그때의 중간 키가 최종 루트가 된다.

암기 포인트

B-트리 분할의 핵심은 '넘치면 중간 키 승격'이며, 최종 루트는 마지막 루트 분할의 중간 키이다.

NoteBase

문 9. 트랜잭션 T_1, T_2가 수행되는 다음 스케줄에서 X, M, N의 초기값이 각각 100, 20, 50일 때, 트랜잭션 수행 후 (가) 위치에서 X의 최종 결과 값은?



① 50

② 70

③ 100

④ 120

정답 2번

출제 포인트

이 문제는 트랜잭션 스케줄에서 미확정 데이터 읽기와 롤백 이후의 최종 write 결과를 추적하는 문제이다.

올바른 순서

1. T1이 X=100을 읽는다. 2. T1이 로컬에서 X=100+20=120으로 계산하고 write하여 데이터베이스의 X가 120이 된다. 3. T2가 이 시점의 X=120을 읽는다. 4. T1이 rollback되어 데이터베이스의 X는 100으로 복구된다. 5. T2는 자신이 이미 읽어 둔 로컬 값 120에서 N=50을 빼 70을 만들고 write한다. 왜 이 순서인가: T2의 계산 입력값은 rollback 이후의 데이터베이스 값 100이 아니라 rollback 전에 read한 로컬 값 120이기 때문이다.

NoteBase

선지별 해설

① X: 50은 T2가 rollback 뒤의 데이터베이스 값 100을 읽었다고 본 값이다. 이 스케줄에서는 T2가 rollback 전에 이미 X=120을 read하였다. 올바르게는 T2의 계산은 120-50=70이다. ② O: T2는 T1의 write 결과 120을 읽은 뒤, T1이 rollback되어도 자신의 로컬 복사본 120으로 계속 계산하므로 최종 write 값은 70이 된다. ③ X: 100은 T1 rollback 직후 데이터베이스가 복구된 값일 뿐 (가) 위치의 최종 write 결과가 아니다. 올바르게는 그 뒤 T2가 70을 write한다. ④ X: 120은 T1이 X+M을 반영한 중간 값이다. 올바르게는 T2가 그 값에서 N=50을 빼므로 최종 값은 70이다.

오답 포인트

롤백이 일어나면 모든 트랜잭션의 읽기값까지 함께 100으로 되돌아간다고 오해하기 쉽다. 이 문제의 함정은 T2가 rollback 전에 이미 더티 데이터 120을 읽었다는 점이다.

다시 나오면 이렇게 판단

스케줄 문제는 각 트랜잭션의 read 시점에 로컬 변수에 무엇이 들어갔는지 먼저 적고, rollback은 데이터베이스 값만 되돌린다는 점과 이후 write가 어느 로컬 값을 쓰는지 순서대로 추적하면 된다.

암기 포인트

read한 뒤에는 로컬 복사본으로 계산하고, rollback은 이미 읽어 간 다른 트랜잭션의 로컬 값까지 자동으로 바꾸지 않는다.

NoteBase

문 10. 다음 두 릴레이션 R과 S의 세미조인(semi join), 즉 R ⋉ S의 결과 릴레이션의 투플과 속성의 수는?

R

A	B	C
1	2	3
6	3	8
9	4	8

S

B	D	E
2	3	4
2	4	5
7	8	10

① 투플 수: 1, 속성 수: 3

② 투플 수: 2, 속성 수: 3

③ 투플 수: 1, 속성 수: 5

④ 투플 수: 2, 속성 수: 5

정답 1번

출제 포인트

세미조인은 조인 조건을 만족하는 왼쪽 릴레이션의 튜플만 남기고, 속성도 왼쪽 릴레이션의 것만 유지하는 연산이라는 점을 묻는 문제이다.

자료 읽는 순서

1. 공통 속성 B를 기준으로 R과 S에서 일치하는 값을 찾는다.
2. R의 튜플 중 S와 매칭되는 튜플만 남긴다.
3. 세미조인은 R의 속성만 남기므로 A, B, C의 3개 속성인지 확인한다.

NoteBase

선지별 해설

① O: R의 B값은 2, 3, 4이고 S의 B값은 2, 2, 7이다. 공통값은 2뿐이므로 남는 R의 튜플은 (1, 2, 3) 1개이다. 세미조인은 R의 속성만 유지하므로 속성 수는 A, B, C의 3개이다.
② X: 튜플 수를 2개로 본 점이 틀리다. S에 B=2가 두 번 있어 조인 결과는 두 행이 될 수 있지만, 이를 R의 속성만 투영하면 같은 R 튜플 (1, 2, 3)만 남아 관계의 중복이 제거되므로 1개이다. 틀린 표현 '튜플 수 2' → 올바르게는 '튜플 수 1'이다.
③ X: 속성 수를 5개로 본 점이 틀리다. 5개는 양쪽 릴레이션의 속성을 함께 두는 일반 조인의 발상이고, 세미조인은 왼쪽 릴레이션 R의 속성만 남긴다. 틀린 표현 '속성 수 5' → 올바르게는 '속성 수 3'이다.
④ X: 튜플 수와 속성 수를 모두 잘못 판단했다. B=2가 S에 두 번 있어도 최종 관계에는 같은 R 튜플이 1개만 남고, 세미조인은 R의 속성 A, B, C만 유지한다. 틀린 표현 '튜플 수 2, 속성 수 5' → 올바르게는 '튜플 수 1, 속성 수 3'이다.

오답 포인트

세미조인을 일반 조인처럼 생각하면 속성을 5개로 세고, S에 같은 B값이 두 번 있다는 이유로 튜플 수도 2개로 세기 쉽다. 이 문제의 함정은 '조인으로 매칭은 여러 번 될 수 있어도, 세미조인 결과는 왼쪽 릴레이션의 튜플 존재 여부만 남긴다'는 점이다.

다시 나오면 이렇게 판단

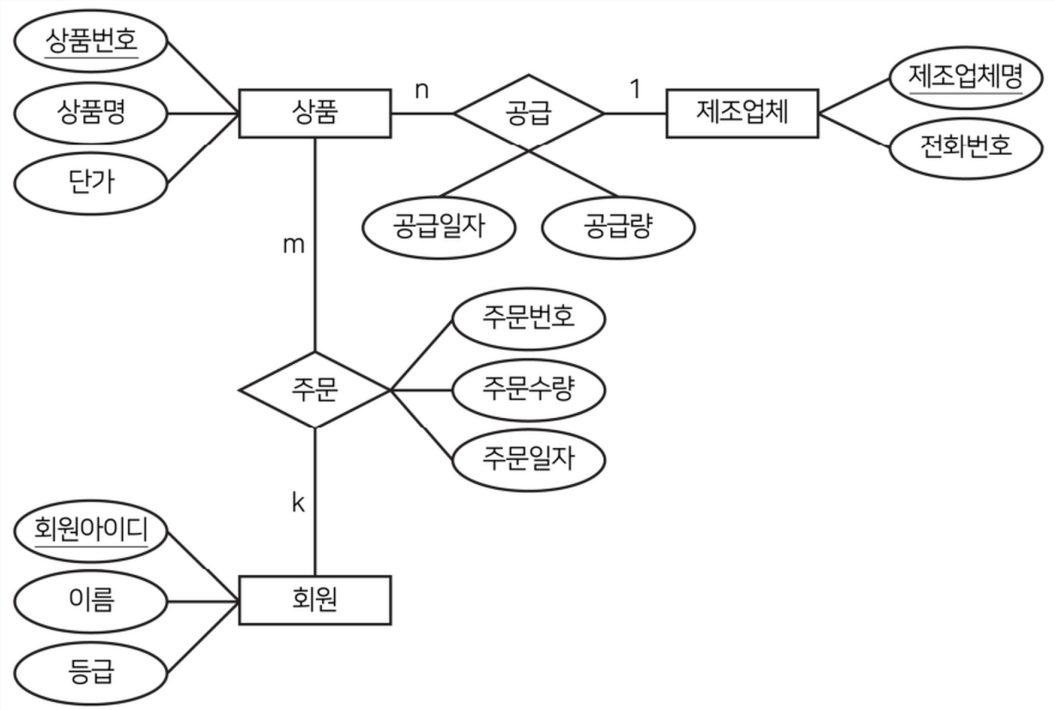
세미조인이 보이면 먼저 공통 속성으로 왼쪽 릴레이션에서 살아남는 튜플만 고른다. 그다음 결과 속성이 왼쪽 릴레이션의 속성만인지 확인한다. 같은 왼쪽 튜플이 여러 번 매칭되어도 최종 관계에서는 중복이 제거되는지 함께 본다.

암기 포인트

세미조인 = 왼쪽 튜플만 남기고, 속성도 왼쪽 것만 남긴다.

NoteBase

문 11. 다음 E-R 다이어그램을 릴레이션 스키마로 바르게 변환한 것은? (단, 밑줄로 표기된 속성은 기본키를 의미하며 k, m, n은 1보다 큰 자연수이다)



① 상품(상품번호, 상품명, 단가, 제조업체명, 공급일자, 공급량)
제조업체(제조업체명, 전화번호)
회원(회원아이디, 이름, 등급)
주문(회원아이디, 상품번호, 주문번호, 주문수량, 주문일자)

② 상품(상품번호, 상품명, 단가, 제조업체명, 공급일자, 공급량)
제조업체(제조업체명, 전화번호)
회원(회원아이디, 이름, 등급)
주문(주문번호, 주문수량, 주문일자)

③ 상품(상품번호, 상품명, 단가)
제조업체(제조업체명, 전화번호, 상품번호, 공급일자, 공급량)
회원(회원아이디, 이름, 등급)
주문(주문번호, 주문수량, 주문일자)

④ 상품(상품번호, 상품명, 단가)
제조업체(제조업체명, 전화번호, 상품번호, 공급일자, 공급량)
회원(회원아이디, 이름, 등급)
주문(회원아이디, 상품번호, 주문번호, 주문수량, 주문일자)

NoteBase

정답 1번

출제 포인트

이 문제는 E-R 다이어그램에서 $1:N$ 관계와 $M:N$ 관계를 구분하고, 관계 속성과 외래키를 어느 릴레이션에 배치하는지 판단하는 문제이다.

선지별 해설

① O
상품-제조업체의 공급은 상품 쪽이 n, 제조업체 쪽이 1인 관계이므로 $1:N$ 변환 규칙에 따라 N측인 상품 릴레이션에 1측 기본키인 제조업체명과 관계 속성 공급일자, 공급량을 포함한다.
주문은 회원과 상품 사이의 $M:N$ 관계이므로 별도 릴레이션을 만들고 참여 엔터티의 기본키 회원아이디, 상품번호와 관계 속성 주문번호, 주문수량, 주문일자를 함께 둔다.

② X
주문(주문번호, 주문수량, 주문일자)만 두면 어떤 회원이 어떤 상품을 주문했는지 연결 정보가 사라져 $M:N$ 관계를 표현할 수 없다.
틀린 표현은 주문 릴레이션에서 회원아이디와 상품번호를 뺀 점이다. 올바르게는 주문(회원아이디, 상품번호, 주문번호, 주문수량, 주문일자)이다.

③ X
공급은 $1:N$ 관계이므로 외래키와 관계 속성은 1측 제조업체가 아니라 N측 상품에 들어가야 한다.
또한 주문 릴레이션에서도 회원아이디와 상품번호가 빠져 회원과 상품의 연결을 표현하지 못한다.
틀린 표현은 제조업체(제조업체명, 전화번호, 상품번호, 공급일자, 공급량)와 주문(주문번호, 주문수량, 주문일자)이다. 올바르게는 상품(상품번호, 상품명, 단가, 제조업체명, 공급일자, 공급량), 주문(회원아이디, 상품번호, 주문번호, 주문수량, 주문일자)이다.

④ X
주문 릴레이션은 회원아이디와 상품번호를 포함해 $M:N$ 관계를 분해한 점은 맞지만, 공급 관계를 제조업체 릴레이션에 넣은 점이 잘못이다.
틀린 표현은 제조업체(제조업체명, 전화번호, 상품번호, 공급일자, 공급량)이다. 올바르게는 상품(상품번호, 상품명, 단가, 제조업체명, 공급일자, 공급량)이다.

오답 포인트

수험생이 가장 많이 틀리는 지점은 $1:N$ 관계의 관계 속성을 1측 릴레이션에 넣거나, $M:N$ 관계를 별도 릴레이션으로 만들면서 참여 엔터티의 기본키를 빼 버리는 점이다. 이 문항에서도 ②는 주문번호만 있으면 된다고 착각하게 만들고, ③과 ④는 공급 정보를 제조업체 쪽에 붙이도록 유도한다.

NoteBase

다시 나오면 이렇게 판단

먼저 관계의 카디널리티를 본다. 1 : *N*이면 1측 기본키와 관계 속성을 N측 릴레이션에 흡수하고, *M* : *N*이면 별도 릴레이션을 만든 뒤 양쪽 기본키와 관계 속성을 함께 둔다. 관계 자체에 식별용 속성이 있어도 참여 엔터티 키는 그대로 유지된다.

암기 포인트

1 : *N*은 N측 흡수, *M* : *N*은 별도 릴레이션, 관계 속성은 관계가 남는 곳에 둔다.

NoteBase

문 12. 트랜잭션에 대한 설명으로 옳지 않은 것은?

① 지속성(durability)은 완료된 트랜잭션이 갱신한 데이터를 데이터베이스에 영구적으로 보존하는 것을 의미한다.

② 원자성(atomicity)은 트랜잭션 연산이 데이터베이스에 모두 반영되지 않고 개별 반영되는 것을 의미한다.

③ 격리성(isolation)은 트랜잭션 수행 시 다른 트랜잭션의 작업이 끼어들지 못하도록 보장하는 것을 의미한다.

④ 일관성(consistency)은 트랜잭션을 수행하기 전과 수행한 후에 데이터베이스가 항상 일관된 상태를 유지하는 것을 의미한다.

정답 2번

출제 포인트

이 문제는 트랜잭션의 ACID 성질 가운데 원자성, 일관성, 격리성, 지속성의 정의를 정확히 구분하는지를 묻는 문제이다.

핵심 개념

원자성은 트랜잭션의 연산이 전부 수행되거나 전혀 수행되지 않는 성질이다.
일관성은 트랜잭션 수행 전후에 데이터베이스가 제약조건을 만족하는 일관된 상태를 유지하는 성질이다.
격리성은 동시에 수행되는 트랜잭션들이 서로의 중간 수행 결과에 간섭하지 않도록 하는 성질이다.
지속성은 성공적으로 완료된 트랜잭션의 결과가 장애가 발생해도 보존되는 성질이다.

선지별 해설

① O: 지속성은 완료된 트랜잭션의 갱신 결과가 데이터베이스에 영구적으로 남는 성질이므로 옳다.
② X: 틀린 부분은 "모두 반영되지 않고 개별 반영되는 것"이다. 원자성은 개별 반영이 아니라 연산이 모두 반영되거나 전혀 반영되지 않는 all-or-nothing 성질이다.
③ O: 격리성은 한 트랜잭션 수행 중 다른 트랜잭션의 작업이 끼어들어 중간 결과를 오염시키지 않도록 하는 성질이므로 옳다.
④ O: 일관성은 트랜잭션 전과 후에 데이터베이스가 항상 일관된 상태를 유지해야 한다는 뜻이므로 옳다.

오답 포인트

②는 원자성을 정반대로 뒤집어 놓은 선지이다. 원자성을 "개별 반영"으로 읽으면 부분 수행이 허용되는 것처럼 보이는데, 이는 원자성이 아니라 원자성 위반 상황이다.

다시 나오면 이렇게 판단

ACID는 키워드로 바로 연결해 판단한다. 원자성은 전부 또는 전무, 일관성은 제약 유지, 격리성은 간섭 차단, 지속성은 커밋 후 보존으로 대응시키면 선지의 O/X를 빠르게 가를 수 있다.

NoteBase

암기 포인트

Atomicity는 all or nothing, Consistency는 제약 유지, Isolation은 간섭 차단, Durability는 커밋 후 보존이다.

NoteBase

문 13. 다음과 같이 '학과'와 '학생' 테이블을 생성하고 UPDATE 질의문 (가)를 실행할 때, 이에 대한 설명으로 옳은 것은?

```
CREATE TABLE 학과 (
    학과명 VARCHAR(20), 전화번호 VARCHAR(20),
    PRIMARY KEY(학과명)
);

CREATE TABLE 학생 (
    학번 VARCHAR(20), 이름 VARCHAR(20),
    학과 VARCHAR(20), PRIMARY KEY(학번),
    FOREIGN KEY(학과) REFERENCES 학과(학과명)
);

UPDATE 학생 SET 학과 = 'AI소프트웨어학과' WHERE 학과
='소프트웨어학과';  --- (가)
```

① 질의문 (가)를 실행하면 개체 무결성 제약조건을 위반하게 된다.

② '학생' 테이블에 학과가 'AI소프트웨어학과'인 튜플이 존재하면 질의문 (가)는 실행되지 않는다.

③ 질의문 (가)는 '학생' 테이블에 있는 모든 튜플의 학과 값을 'AI소프트웨어학과'로 변경한다.

④ 학과명이 'AI소프트웨어학과'인 튜플이 '학과' 테이블에 존재하지 않으면, 질의문 (가)는 참조 무결성 제약조건을 위반한다.

정답 4번

출제 포인트

이 문제는 외래 키 속성을 UPDATE할 때 개체 무결성과 참조 무결성 중 어떤 제약이 적용되는지를 실제 SQL 문장에 대입해 판단하는 문제이다.

사례 정리

학과 테이블의 기본키는 학과명이다.
학생 테이블의 기본키는 학번이고, 학생.학과는 학과(학과명)를 참조하는 외래 키이다.
질의문 (가)는 학생 테이블에서 학과 값이 '소프트웨어학과'인 행만 'AI소프트웨어학과'로 바꾸는 UPDATE문이다.

NoteBase

핵심 개념

개체 무결성은 기본키가 NULL이 아니고 각 튜플을 유일하게 식별할 수 있어야 한다는 제약이다.
참조 무결성은 외래 키 값이 참조 테이블의 기본키 값으로 존재해야 한다는 제약이다.
이 문장에서는 학생.학과가 기본키가 아니라 외래 키이므로 핵심 판단 기준은 개체 무결성이 아니라 참조 무결성이다.

적용 기준

이 사례에서는 UPDATE 대상 속성이 무엇인지 먼저 본다. 학생.학과는 외래 키이므로, 변경 후 값인 'AI소프트웨어학과'가 학과 테이블의 학과명에 실제로 존재하는지가 핵심이다. 존재하면 갱신할 수 있고, 존재하지 않으면 참조 무결성을 위반한다.

선지별 해설

① X
학생.학과는 학생 테이블의 기본키가 아니라 외래 키이므로 이 UPDATE가 곧바로 개체 무결성 제약을 위반하는 것은 아니다.
틀린 표현은 '개체 무결성 제약조건을 위반하게 된다'이다. 올바르게는 '참조 대상이 없을 때만 참조 무결성 제약조건을 위반할 수 있다'이다.

② X
학생 테이블에 이미 학과가 'AI소프트웨어학과'인 튜플이 있어도 같은 외래 키 값을 여러 행이 공유하는 것은 가능하므로 UPDATE 실행 자체가 막히지 않는다.
틀린 표현은 '질의문 (가)는 실행되지 않는다'이다. 올바르게는 '기존에 같은 학과 값을 가진 튜플이 있어도 참조 무결성만 만족하면 실행된다'이다.

③ X
WHERE 조건이 학과='소프트웨어학과'이므로 모든 튜플이 아니라 조건을 만족하는 튜플만 변경된다.
틀린 표현은 '학생 테이블에 있는 모든 튜플의 학과 값을'이다. 올바르게는 '학과 값이 소프트웨어학과인 튜플들의 학과 값만'이다.

④ O
변경 후 외래 키 값인 'AI소프트웨어학과'가 부모 테이블인 학과 테이블에 없으면 학생.학과가 존재하지 않는 학과명을 참조하게 되므로 참조 무결성 제약조건을 위반한다.

NoteBase

오답 포인트

기본키와 외래 키를 구분하지 않고 '무결성'이라는 단어만 보고 ①로 넘어가기 쉽다. 또한 UPDATE문을 보고 전체 행이 바뀐다고 착각하면 ③에 끌리기 쉽다. 이 문항은 변경되는 속성이 학생.학과라는 점과 WHERE 조건이 붙어 있다는 점을 동시에 보는지를 묻는다.

다시 나오면 이렇게 판단

UPDATE 문제는 먼저 변경 대상 속성이 기본키인지 외래 키인지 구분한다. 그다음 WHERE 조건으로 바뀌는 행의 범위를 확인하고, 외래 키라면 변경 후 값이 부모 테이블에 존재하는지만 보면 된다.

암기 포인트

기본키 변경은 개체 무결성, 외래 키 변경은 참조 무결성으로 판단한다.

NoteBase

문 14. 트랜잭션 T_1, T_2, T_3가 다음과 같은 순서로 스케줄링 될 경우, 이와 동치인 직렬 스케줄은?

시간 ↓

T_1	T_2	T_3
read(y)		
read(z)		
	read(x)	
	write(x)	
write(y)		
write(z)		
		read(z)
		read(x)
	read(y)	
	write(y)	

① $T_1 \rightarrow T_2 \rightarrow T_3$

② $T_1 \rightarrow T_3 \rightarrow T_2$

③ $T_2 \rightarrow T_1 \rightarrow T_3$

④ 동치인 직렬 스케줄이 없음

정답 1번

출제 포인트

이 문제는 충돌 연산으로 선행 그래프를 만들고 그 위상정렬로 동치인 직렬 스케줄을 찾는 문제이다.

올바른 순서

1. T1 - y와 z를 읽은 뒤 각각 y, z에 write를 수행한다.
2. T2 - x를 읽고 write한 뒤, 나중에 y를 읽고 write한다.
3. T3 - T1이 쓴 z와 T2가 쓴 x를 읽는다.

왜 이 순서: y에서는 T1의 read(y), write(y)가 T2의 read(y), write(y)보다 앞서므로 T1→T2가 되고, x에서는 T2의 write(x)가 T3의 read(x)보다 앞서므로 T2→T3가 되며, z에서는 T1의 write(z)가 T3의 read(z)보다 앞서므로 T1→T3가 된다.

NoteBase

선지별 해설

① O
T1→T2, T2→T3, T1→T3의 충돌 제약을 모두 만족하는 직렬 순서이다.

② X
틀린 부분: T3를 T2보다 먼저 둔 순서이다.
올바르게는: x에서 T2의 write(x)가 T3의 read(x)보다 먼저이므로 T2가 T3보다 앞서야 한다.

③ X
틀린 부분: T2를 T1보다 먼저 둔 순서이다.
올바르게는: y에서 T1의 read(y), write(y)가 T2의 read(y), write(y)보다 먼저이므로 T1이 T2보다 앞서야 한다.

④ X
틀린 부분: 동치인 직렬 스케줄이 없다고 본 점이다.
올바르게는: 선행 그래프가 T1→T2→T3의 비순환 그래프이므로 동치인 직렬 스케줄이 존재하고, 그 순서는 T1→T2→T3이다.

오답 포인트

같은 데이터 항목에서 read-write, write-read, write-write만 충돌인데 read-read까지 간선으로 잘못 잡으면 순서 제약을 과하게 만들어 오답으로 흐르기 쉽다. 또 간선은 보이는데 위상정렬까지 하지 않고 감으로 순서를 고르면 ②나 ③을 고르기 쉽다.

다시 나오면 이렇게 판단

스케줄 문제는 같은 데이터 항목끼리만 묶어서 충돌 연산을 찾고, 앞선 연산의 트랜잭션에서 뒤 연산의 트랜잭션으로 간선을 그린 뒤, 만들어진 선행 그래프를 위상정렬해서 직렬 순서를 판단한다.

암기 포인트

충돌 간선만 그리면 y에서 T1→T2, x에서 T2→T3, z에서 T1→T3가 되어 직렬 순서는 T1→T2→T3 하나로 정리된다.

NoteBase

문 15. 다음 '사원'과 '부서' 테이블에 대한 SQL문의 수행 결과가 다른 것은? (단, 밑줄로 표기된 속성은 기본키를 의미한다)

사원

사번	이름	부서번호	급여	직속상사
1	Kim	10	500	NULL
2	Kwon	20	500	3
3	Park	20	700	NULL
4	Choi	30	600	1
5	Cho	10	300	3
6	Lee	20	550	4
7	Yoon	30	400	3
8	Chae	20	650	1
9	Nam	10	430	4
10	Ha	30	200	NULL

부서

부서번호	부서이름	사무실위치
10	인사과	101호
20	개발팀	102호
30	총무과	103호

①
```
SELECT 이름
FROM 사원
WHERE 이름 LIKE '_a%' AND 급여 > 500;
```

②
```
SELECT E.이름
FROM 사원 AS E INNER JOIN 부서 AS D ON E.부서번호 = D.부서번호
WHERE D.사무실위치 = '102호' AND E.직속상사 is NULL;
```

③
```
SELECT 이름
FROM 사원
WHERE 이름 NOT IN (SELECT 이름 FROM 사원
WHERE 급여 < ANY (SELECT 급여 FROM 사원));
```

④
```
SELECT 이름
FROM 사원
WHERE 부서번호 IN (SELECT 부서번호 FROM 사원
GROUP BY 부서번호
HAVING COUNT(*) > 3);
```

정답 4번

출제 포인트

이 문제는 사원·부서 테이블을 기준으로 각 SQL문의 실제 반환 결과를 계산해 같은 결과인지 다른 결과인지를 비교하는 문제이다.

NoteBase

자료 읽는 순서

1. 이름 조건, 급여 조건, 부서번호 조건, NULL 조건을 각각 분리해 읽는다.
2. 각 SQL문이 최종적으로 어떤 사원 이름을 반환하는지 한 번씩 직접 적어 본다.
3. 반환 집합이 같은 문장끼리 묶고, 다른 집합을 반환하는 문장을 찾는다.

핵심 근거

사원 표에서 이름이 '_a%'이고 급여가 500 초과인 사람은 Park만 해당한다.
부서 표를 조인해 사무실위치가 '102호'이면서 직속상사가 NULL인 사람도 Park만 해당한다.
급여 < ANY (SELECT 급여 FROM 사원)은 최고 급여 700보다 작은 급여를 뜻하므로 Park를 제외한 전원이 서브쿼리에 들어가고, 바깥 질의 결과는 Park만 남는다.
반면 부서번호별 COUNT(*) > 3인 부서는 20번뿐이므로 ④는 20번 부서 사원인 Kwon, Park, Lee, Chae를 반환한다.

선지별 해설

① X
- 이름 LIKE '_a%'는 두 번째 글자가 a인 이름을 찾는다. Park, Nam, Ha가 후보이지만 급여 > 500 조건까지 만족하는 사람은 Park뿐이다.
- 실제 결과는 Park 1행이다.
- 틀린 표현을 바로잡으면, ①은 다른 결과를 내는 문장이 아니라 ②, ③과 같은 결과를 내는 문장이다.

② X
- 부서 사무실위치가 '102호'이면 부서번호 20이다. 20번 부서 사원은 Kwon, Park, Lee, Chae이다.
- 여기서 직속상사가 NULL인 사람은 Park만 해당하므로 실제 결과는 Park 1행이다.
- 틀린 표현을 바로잡으면, ②도 수행 결과가 다른 문장이 아니라 Park만 반환하는 문장이다.

③ X
- 급여 < ANY (SELECT 급여 FROM 사원)은 사원 급여 중 하나라도 더 큰 값이 있으면 참이라는 뜻이므로 최고 급여 700보다 작은 급여를 가진 사원을 찾는 조건과 같다.
- 따라서 서브쿼리에는 Park를 제외한 모든 이름이 들어가고, NOT IN 결과로 Park만 남는다.
- 틀린 표현을 바로잡으면, ③ 역시 다른 결과를 내는 문장이 아니라 Park만 반환하는 문장이다.

④ O
- 부서번호별 인원 수는 10번 부서 3명, 20번 부서 4명, 30번 부서 3명이다. HAVING COUNT(*) > 3을 만족하는 부서는 20번뿐이다.
- 따라서 바깥 질의는 20번 부서 사원인 Kwon, Park, Lee, Chae를 반환한다.
- ①, ②, ③은 모두 Park 1행인데 ④만 4행을 반환하므로 수행 결과가 다르다.

NoteBase

오답 포인트

③의 ANY를 SOME/EXISTS처럼 막연히 읽거나, ④의 HAVING COUNT(*) > 3을 3명 이상으로 잘못 읽으면 오답이 된다. 또한 ①의 LIKE '_a%'에서 밑줄이 한 글자 와일드카드라는 점을 놓치면 Park, Nam, Ha 후보를 먼저 추리지 못한다.

다시 나오면 이렇게 판단

SQL 결과 비교 문제는 문장 의미를 말로 바꾼 뒤 각 질의의 최종 반환 집합을 직접 써서 비교하면 된다. 특히 LIKE, NULL, ANY, GROUP BY-HAVING은 조건을 한글로 풀어 적으면 실수가 줄어든다.

암기 포인트

결과 비교형 SQL 문제는 조건 해석 후 반환 이름 집합을 직접 적어 보면 답이 고정된다.

NoteBase

문 16. 데이터베이스에 반영이 완료된 레코드만 읽게 하는 방법(READ COMMITTED)으로 해결할 수 있는 동시 처리(concurrent processing) 문제는?

① 오손 읽기(dirty read)
② 유령 읽기(phantom read)
③ 직렬 읽기(serialized read)
④ 반복 불가능 읽기(non-repeatable read)

정답 1번

출제 포인트

이 문제는 트랜잭션 격리수준인 READ COMMITTED가 어떤 동시성 이상 현상을 막고, 무엇은 아직 허용하는지를 구별하는 문제이다.

핵심 개념

오손 읽기는 아직 커밋되지 않은 다른 트랜잭션의 값을 읽는 현상이다. READ COMMITTED는 커밋이 끝난 데이터만 읽게 하므로 오손 읽기를 막는다. 반복 불가능 읽기는 같은 행을 두 번 읽는 사이 다른 트랜잭션이 커밋하여 값이 달라지는 현상이고, 유령 읽기는 같은 조건으로 다시 조회했을 때 새 행이 나타나거나 사라지는 현상이다.

선지별 해설

1번: O. 오손 읽기는 미커밋 데이터를 읽는 현상인데 READ COMMITTED는 커밋 완료 레코드만 읽게 하므로 이 문제를 해결한다.
2번: X. 유령 읽기는 조건 검색 결과의 행 집합이 바뀌는 현상이라 READ COMMITTED만으로는 막지 못한다. 틀린 표현은 READ COMMITTED로 해결된다고 본 점이고, 올바르게는 더 강한 격리수준에서 통제하는 문제이다.
3번: X. 직렬 읽기는 READ COMMITTED가 해결하는 동시성 이상 현상의 이름이 아니다. 틀린 표현은 직렬 읽기를 이상 현상처럼 제시한 점이고, 올바르게는 직렬 가능성 또는 SERIALIZABLE이 더 강한 격리 보장의 개념이다.
4번: X. 반복 불가능 읽기는 같은 행을 다시 읽을 때 값이 달라지는 현상으로 READ COMMITTED에서는 여전히 발생할 수 있다. 틀린 표현은 READ COMMITTED로 해결된다고 본 점이고, 올바르게는 이 수준이 오손 읽기만 방지한다고 이해하는 편이 맞다.

오답 포인트

유령 읽기와 반복 불가능 읽기를 모두 READ COMMITTED가 막는다고 섞어 외우면 틀린다. 이 문항의 함정은 커밋된 데이터만 읽는다는 문장을 곧 모든 읽기 이상 현상 차단으로 과대해석하게 만드는 데 있다.

NoteBase

다시 나오면 이렇게 판단

격리수준 문제가 나오면 먼저 미커밋 데이터 접근 여부를 본다. 미커밋 값을 읽는 현상이면 오손 읽기이고 READ COMMITTED가 막는다. 같은 행 값 변화면 반복 불가능 읽기, 조회 결과 행 집합 변화면 유령 읽기로 구분한다.

암기 포인트

READ COMMITTED는 오손 읽기 차단, 반복 불가능 읽기와 유령 읽기는 잔존이다.

NoteBase

문 17. 다음 테이블은 쇼핑몰의 판매자가 판매하는 상품과 해당 상품을 생산한 제조사에 대한 정보를 담고 있다. 테이블의 기본키가 (판매자ID, 상품)일 때, 다음 설명 중 옳은 것만을 모두 고르면? (단, 아래 튜플을 기준으로 판단한다)

판매자ID	상품	제조사
0001	컴퓨터	A
0001	모니터	B
0002	컴퓨터	A
0002	모니터	C
0003	모니터	C
0003	컴퓨터	D

ㄱ. 이 테이블은 제2정규형의 조건을 만족한다.

ㄴ. 이 테이블은 제3정규형의 조건을 만족하지 않는다.

ㄷ. 제조사 E가 모니터를 생산한다는 정보를 판매자ID 없이 튜플로 추가하더라도 삽입 이상이 발생하지 않는다.

ㄹ. 판매자ID가 0001인 판매자가 모니터를 더 이상 판매하지 않아 해당 튜플을 삭제한다면 삭제 이상이 발생할 수 있다.

ㅁ. 제조사 B가 생산하는 상품이 모니터에서 노트북으로 변경된다면 갱신 이상이 발생할 수 있다.

① ㄱ, ㄷ

② ㄱ, ㄹ

③ ㄴ, ㅁ

④ ㄹ, ㅁ

정답 2번

출제 포인트

이 문제는 현재 튜플만을 기준으로 함수 종속을 읽어 제2정규형·제3정규형 충족 여부와 삽입·삭제·갱신 이상의 발생 가능성을 구별하는지를 묻는다.

NoteBase

핵심 개념

제2정규형은 복합 기본키의 일부에만 비주요 속성이 종속되지 않으면 만족한다.
제3정규형은 비주요 속성이 다른 비주요 속성에 이행 종속되지 않아야 하며, 우변이 후보키 속성이면 허용된다.
현재 튜플 기준으로는 판매자ID만으로도, 상품만으로도 제조사가 정해지지 않으므로 제조사는 기본키 전체에 종속된다. 또한 제조사별 상품이 하나씩이어서 제조사 → 상품이 성립하지만, 이때 상품은 후보키 속성으로 볼 수 있어 제3정규형은 유지된다.

선지별 해설

1번 X: 1번은 ㄱ, ㄷ의 조합인데 ㄱ은 맞고 ㄷ이 틀리므로 오답이다. ㄱ은 상품만으로도, 판매자ID만으로도 제조사가 결정되지 않으므로 비주요 속성 제조사는 복합키 (판매자ID, 상품) 전체에 완전 함수 종속되어 제2정규형을 만족한다. ㄷ의 틀린 표현은 "삽입 이상이 발생하지 않는다"이다. 이 테이블은 제조사 E가 모니터를 생산한다는 사실만 따로 넣으려 해도 기본키의 일부인 판매자ID가 없어서 튜플을 넣을 수 없으므로, 올바르게는 "삽입 이상이 발생할 수 있다"이다.
2번 O: 2번은 ㄱ, ㄹ의 조합이고 둘 다 맞으므로 정답이다. ㄱ은 앞서 본 것처럼 부분 함수 종속이 없어 제2정규형을 만족한다. ㄹ은 판매자ID가 0001인 판매자의 모니터 판매 튜플 (0001, 모니터, B)을 삭제하면 제조사 B가 모니터를 생산한다는 정보 자체가 함께 사라지므로 삭제 이상이 발생할 수 있다.
3번 X: 3번은 ㄴ, ㅁ의 조합인데 둘 다 틀리므로 오답이다. ㄴ의 틀린 표현은 "제3정규형을 만족하지 않는다"이다. 현재 튜플 기준으로 제조사 → 상품이 성립하더라도 상품은 후보키를 이루는 주요 속성이므로, 올바르게는 "제3정규형을 만족한다"이다. ㅁ의 틀린 표현은 "갱신 이상이 발생할 수 있다"이다. 현재 데이터에서는 제조사 B가 한 번만 나타나므로 모니터를 노트북으로 바꿀 때 한 튜플만 수정하면 되어, 올바르게는 "현재 튜플 기준으로는 갱신 이상이라고 보기 어렵다"이다.
4번 X: 4번은 ㄹ, ㅁ의 조합인데 ㄹ은 맞고 ㅁ이 틀리므로 오답이다. ㄹ은 삭제 시 제조사 B와 상품 모니터의 대응 정보가 함께 소실될 수 있으므로 맞다. 반면 ㅁ은 제조사 B 관련 정보가 중복 저장되어 있지 않다는 점을 놓친 판단이다. 올바르게는 "제조사 B가 한 번만 등장하므로 이 변경만으로는 갱신 이상이 발생하지 않는다"이다.

오답 포인트

현재 튜플만 기준으로 판단한다는 조건을 놓치면 ㅁ을 습관적으로 맞다고 보기 쉽다. 또 제조사 → 상품이 보인다고 해서 바로 제3정규형 위반으로 단정하면 ㄴ을 잘못 고르게 된다. 제3정규형은 우변이 주요 속성이면 허용된다는 점이 함정이다.

다시 나오면 이렇게 판단

먼저 기본키의 일부만으로 비주요 속성이 정해지는지 확인해 제2정규형을 판정한다. 다음으로 현재 튜플에서 보이는 추가 함수 종속이 있더라도 그 우변이 주요 속성인지 확인해 제3정규형 위반인지 가른다. 마지막으로 어떤 사실이 여러 튜플에 중복 저장되는지, 한 튜플 삭제나 삽입 불가가 독립 정보 손실로 이어지는지를 보면 이상 현상을 판정할 수 있다.

NoteBase

암기 포인트

2정규형은 부분 종속 금지이고, 3정규형은 비주요 속성의 이행 종속 금지이다. 삭제 이상은 유일한 사실이 튜플 삭제와 함께 사라질 때 생기고, 갱신 이상은 같은 사실이 여러 튜플에 중복될 때 생긴다.

NoteBase

문 18. 릴레이션 R(A, B, C, D)에 다음과 같은 함수 종속성의 집합 F가 존재할 때, 후보키에 해당하는 것과 릴레이션 R이 만족하는 정규형만을 모두 고르면? (단, 이 릴레이션의 기본키는 (A, B)이며, 모든 속성의 도메인은 원자값(atomic value)으로 구성된다)

F = {(A, B) → C, (A, B) → D, C → B, D → A}
(가) (B, C) (나) (B, D) (다) (C, D)
ㄱ. 제1정규형
ㄴ. 제2정규형
ㄷ. 제3정규형
ㄹ. 보이스/코드 정규형

① 후보키: (가), (다) / 정규형: ㄱ, ㄴ, ㄷ
② 후보키: (나), (다) / 정규형: ㄱ, ㄴ, ㄷ
③ 후보키: (가), (나) / 정규형: ㄱ, ㄴ, ㄷ, ㄹ
④ 후보키: (나), (다) / 정규형: ㄱ, ㄴ, ㄷ, ㄹ

정답 2번

출제 포인트

속성 폐포로 후보키를 판별하고, 주요 속성 여부를 기준으로 제3정규형과 보이스/코드 정규형을 구분하는 문제이다.

핵심 개념

후보키는 모든 속성을 결정하면서 더 이상 줄일 수 없는 최소 슈퍼키이다.
제3정규형은 모든 함수 종속 $X \rightarrow Y$에서 X가 슈퍼키이거나 Y가 주요 속성이면 만족한다.
보이스/코드 정규형은 모든 함수 종속의 결정자 X가 반드시 슈퍼키여야 하므로 제3정규형보다 더 엄격하다.
이 문항에서는 기본키 (A, B) 외에도 (B, D), (C, D)가 후보키가 되어 모든 속성 A, B, C, D가 주요 속성이 된다.

NoteBase

선지별 해설

① X
- 후보키 부분이 틀리다. $(B, C)^+$는 $C \to B$만 적용되어 여전히 B, C만 알 수 있으므로 A와 D를 결정하지 못한다. 따라서 (가)는 후보키가 아니다.
- 정규형 부분인 ㄱ, ㄴ, ㄷ은 맞다. 올바르게는 후보키가 (나), (다)인 선택지여야 한다.

② O
- $(B, D)^+$는 $D \to A$로 A를 얻고, 이어서 $(A, B) \to C$로 C를 얻어 모든 속성을 결정한다. 따라서 (나)는 후보키이다.
- $(C, D)^+$는 $C \to B, D \to A$를 적용하면 A, B, C, D 전체를 얻으므로 (다)도 후보키이다.
- 후보키가 (A, B), (B, D), (C, D)이므로 모든 속성이 주요 속성이다. 따라서 제1정규형, 제2정규형, 제3정규형은 만족하지만, $C \to B, D \to A$에서 결정자 C, D가 슈퍼키가 아니어서 보이스/코드 정규형은 아니다.

③ X
- 후보키 부분이 틀리다. (가)는 후보키가 아니고, 올바르게는 (나), (다)이다.
- 정규형 부분도 틀리다. ㄹ까지 포함했지만 $C \to B, D \to A$가 존재하므로 이 릴레이션은 보이스/코드 정규형이 아니다. 올바르게는 ㄱ, ㄴ, ㄷ만 만족한다.

④ X
- 후보키 부분인 (나), (다)는 맞다.
- 그러나 정규형 부분이 틀리다. 보이스/코드 정규형은 모든 결정자가 슈퍼키여야 하는데 C와 D는 슈퍼키가 아니다. 올바르게는 ㄱ, ㄴ, ㄷ만 만족한다.

오답 포인트

기본키가 (A, B)라고 주어졌다고 해서 후보키를 하나만 있다고 단정하면 (나), (다)를 놓치게 된다. 또 결정자가 슈퍼키가 아니면 곧바로 제3정규형도 깨진다고 오해하기 쉬운데, 이 문제는 종속자 A, B가 모두 주요 속성이므로 제3정규형은 유지된다.

다시 나오면 이렇게 판단

먼저 각 보기의 속성 집합에 대해 폐포를 계산해 후보키 여부를 가린다. 그다음 후보키들의 합집합으로 주요 속성을 정하고, 결정자가 슈퍼키인지와 종속자가 주요 속성인지를 차례로 확인하면 제3정규형과 보이스/코드 정규형을 구분할 수 있다.

암기 포인트

제3정규형은 결정자가 슈퍼키이거나 종속자가 주요 속성이면 되고, 보이스/코드 정규형은 결정자가 항상 슈퍼키여야 한다.

NoteBase

문 19. '학생' 릴레이션의 과목번호는 '과목' 릴레이션의 과목번호를 참조하는 외래키이다. 다음 설명 중 옳지 않은 것은? (단, 밑줄로 표기된 속성은 기본키를 의미한다)

학생

학번	이름	과목번호
1111	을지문덕	200
2222	신사임당	100
3333	이율곡	300
4444	유관순	400
5555	김만덕	400

과목

과목번호	과목명	담당교수	개설학과
100	인공지능개론	안창호	AI소프트웨어학과
200	빅데이터분석	김구	컴퓨터공학과
300	머신러닝	윤봉길	AI소프트웨어학과
400	생성형AI	김좌진	AI소프트웨어학과

① "DELETE FROM 과목 WHERE 과목번호 = 400;"을 실행하면 참조 무결성 제약조건을 위반한다.

② "UPDATE 학생 SET 과목번호 = 500 WHERE 학번 = 3333;"을 실행하면 참조 무결성 제약조건을 위반한다.

③ "INSERT INTO 학생(학번, 이름, 과목번호) VALUES (5555, '홍길동', 200);"을 실행하면 기본키 제약조건을 위반한다.

④ "INSERT INTO 과목(과목번호, 과목명, 담당교수, 개설학과) VALUES (NULL, '데이터베이스', '안중근', 'AI소프트웨어학과');"를 실행하면 참조 무결성 제약조건을 위반한다.

NoteBase

정답 4번

출제 포인트

이 문제는 기본키와 외래키가 걸린 두 릴레이션에서 각 SQL 문이 기본키 제약, 개체 무결성, 참조 무결성 중 무엇을 위반하는지를 구분할 수 있는지를 묻는 문제이다.

사례 정리

학생 릴레이션의 기본키는 학번이고, 과목번호는 과목 릴레이션의 과목번호를 참조하는 외래키이다.
과목 릴레이션의 과목번호 400은 학생 릴레이션의 학번 4444, 5555가 이미 참조하고 있다.
학생 릴레이션에는 학번 5555가 이미 존재하고, 과목 릴레이션에는 과목번호 500이 존재하지 않는다.

핵심 개념

기본키 제약은 기본키 값의 유일성을 요구하는 제약이다.
개체 무결성은 기본키가 NULL이 될 수 없다는 제약이다.
참조 무결성은 외래키 값이 참조 릴레이션의 기본키 값으로 존재해야 한다는 제약이다.

적용 기준

삭제나 갱신 뒤에 외래키가 고아 값이 되는지 보면 참조 무결성 위반 여부를 판단할 수 있다.
삽입 시 기본키가 중복되면 기본키 제약 위반이고, 기본키 자리에 NULL이 들어가면 개체 무결성 위반이다.

선지별 해설

① O
`과목`에서 과목번호 400을 삭제하면 `학생`의 두 튜플이 더 이상 존재하지 않는 과목번호 400을 참조하게 된다. 따라서 참조 무결성 제약조건을 위반한다.

② O
학번 3333의 과목번호를 500으로 바꾸면 `학생.과목번호=500`이 되는데, `과목` 릴레이션에는 과목번호 500이 없다. 외래키가 존재하지 않는 기본키 값을 가리키므로 참조 무결성 제약조건을 위반한다.

③ O
새 튜플의 학번 5555는 이미 `학생` 릴레이션에 존재한다. 따라서 학번의 유일성이 깨져 기본키 제약조건을 위반한다.

④ X
과목번호는 `과목` 릴레이션의 기본키이므로 NULL을 넣을 수 없다. 이 문장은 위반 종류를 참조 무결성이라고 한 부분이 틀렸다. 올바르게는 기본키의 NULL 금지를 어겨 개체 무결성 제약조건을 위반한다고 보아야 한다.

NoteBase

오답 포인트

④는 외래키, 참조, 무결성이라는 단어에 끌리면 참조 무결성으로 오인하기 쉽다. 그러나 NULL이 들어간 위치가 외래키가 아니라 기본키 `과목번호`라는 점이 핵심이다.

다시 나오면 이렇게 판단

제약조건 문제는 먼저 값이 들어가거나 바뀌는 속성이 기본키인지 외래키인지부터 확인한 뒤, 중복이면 기본키 제약, NULL이면 개체 무결성, 존재하지 않는 참조값이면 참조 무결성으로 분류하면 된다.

암기 포인트

기본키 중복은 기본키 제약, 기본키 NULL은 개체 무결성, 외래키의 미존재 참조는 참조 무결성이다.

NoteBase

문 20. RAID에 대한 설명으로 옳지 않은 것은?

① RAID 0은 여러 개의 디스크에 데이터를 분산 저장한다.

② RAID 1은 디스크의 모든 데이터를 별도의 디스크에 복제하여 유지한다.

③ RAID 5는 데이터와 패리티 정보를 여러 디스크에 분산 저장한다.

④ RAID 6은 데이터를 저장하는 디스크와 패리티를 저장하는 디스크를 분리하여 사용한다.

정답 4번

출제 포인트

이 문제는 RAID 0, 1, 5, 6의 저장 방식과 패리티 배치 방식을 구별할 수 있는지를 묻는 문제이다.

핵심 개념

RAID 0은 여러 디스크에 데이터를 스트라이핑하여 성능을 높이는 방식이고 중복성은 없다.
RAID 1은 동일 데이터를 다른 디스크에 그대로 복제하는 미러링 방식이다.
RAID 5는 데이터와 패리티를 여러 디스크에 분산 저장하는 방식이다.
RAID 6은 RAID 5처럼 분산 저장을 하되 이중 패리티를 여러 디스크에 분산 저장하는 방식이다.

선지별 해설

1번: O. RAID 0은 데이터를 여러 디스크에 분산하여 저장하는 스트라이핑 방식이다. 성능 향상에는 유리하지만 장애 대비용 중복 정보는 없다.
2번: O. RAID 1은 한 디스크의 데이터를 다른 디스크에 동일하게 복제하여 유지하는 미러링 방식이다.
3번: O. RAID 5는 데이터 블록과 패리티 정보를 여러 디스크에 분산 저장한다. 전용 패리티 디스크를 따로 두지 않는 점이 핵심이다.
4번: X. 틀린 부분은 "데이터를 저장하는 디스크와 패리티를 저장하는 디스크를 분리하여 사용한다"는 표현이다. RAID 6은 전용 패리티 디스크를 분리하는 방식이 아니라 데이터와 이중 패리티를 여러 디스크에 분산 저장하는 방식이다.

오답 포인트

RAID 4의 전용 패리티 디스크 개념을 RAID 6과 혼동하면 4번을 고르지 못한다. RAID 5와 RAID 6은 모두 패리티를 분산 저장하고, RAID 6만 이중 패리티를 사용한다는 점이 함정이다.

다시 나오면 이렇게 판단

RAID 문제는 먼저 중복성 유무를 보고, 다음으로 미러링인지 스트라이핑인지, 마지막으로 패리티가 전용인지 분산인지와 패리티 개수가 하나인지 둘인지 순서대로 판단하면 된다.

NoteBase

암기 포인트

RAID 0은 성능, RAID 1은 복제, RAID 5는 분산 패리티, RAID 6은 분산 이중 패리티이다.

NoteBase

문 21. 즉시 갱신 전략을 이용하는 회복 시스템에서 다음과 같은 로그 기록을 사용하여 복구 과정을 수행한 내용 중 옳지 않은 것은?

<T1, start>
<T1, A, 30, 60>
<T2, start>
<T1, commit>
<T3, start>
<T4, start>
<checkpoint, {T2, T3, T4}>
<T3, C, 15, 45>
<T3, commit>
<T5, start>
<T2, B, 10, 50>
<T4, A, 60, 90>
<T4, abort>
<T5, D, 20, 40>
-- 시스템 장애 발생 --

① 트랜잭션 T1은 복구 작업을 수행할 필요가 없다.

② 트랜잭션 T2는 undo를 수행하여 B값으로 10을 저장한다.

③ 트랜잭션 T3는 undo를 수행하여 C값으로 15를 저장한다.

④ 트랜잭션 T5는 undo를 수행하여 D값으로 20을 저장한다.

정답 3번

출제 포인트

이 문제는 즉시 갱신 회복에서 체크포인트 이후 각 트랜잭션을 redo 대상, undo 대상, 복구 제외 대상으로 구분하는 절차를 묻는 문제이다.

NoteBase

올바른 순서

1. 체크포인트 시점의 활성 트랜잭션 집합을 확인한다. 이 로그에서는 {T2, T3, T4}이다.
2. 체크포인트 이후 로그를 앞으로 읽으며 commit, abort, start를 반영해 crash 시점 상태를 분류한다. T3는 commit, T4는 abort, T5는 start 상태가 추가된다.
3. commit이 끝난 T3는 redo 대상이다. 즉시 갱신에서는 커밋된 트랜잭션의 변경이 디스크에 반영되지 않았을 수 있으므로 재실행해 준다.
4. crash 시점 미완료인 T2와 T5는 undo 대상이다. 즉시 갱신에서는 부분 반영 가능성이 있으므로 이전 값으로 되돌린다.
5. 체크포인트 이전에 이미 commit한 T1은 복구 대상이 아니다. T4는 abort가 기록되었으므로 중단 처리에서 이미 rollback된 것으로 본다.

왜 이 순서인가: 체크포인트 이후에 누가 끝났는지와 crash 시점에 누가 미완료인지가 먼저 정해져야 redo와 undo 대상을 정확히 나눌 수 있기 때문이다.

선지별 해설

① O

T1은 체크포인트 이전에 이미 commit되었고, 체크포인트는 그 이전 완료 트랜잭션의 반영 상태를 안정화한 기준점으로 본다. 따라서 복구 작업 대상에서 제외된다.

② O

T2는 체크포인트 시점에도 활성 상태였고 crash 전까지 commit 기록이 없다. 따라서 loser 트랜잭션이므로 undo 대상이며, 로그 <T2, B, 10, 50>에 따라 B값은 이전 값 10으로 복원된다.

③ X

T3는 체크포인트 이후 <T3, commit>이 있으므로 undo 대상이 아니라 redo 대상이다. 틀린 표현은 'undo를 수행하여 C값으로 15를 저장한다'이다. 올바르게는 redo를 수행하여 C값 45가 유지되도록 한다.

④ O

T5는 체크포인트 이후 시작되었지만 commit 없이 장애가 발생했다. 따라서 undo 대상이며, 로그 <T5, D, 20, 40>에 따라 D값은 이전 값 20으로 되돌린다.

오답 포인트

가장 흔한 함정은 '체크포인트에 포함된 트랜잭션은 모두 undo한다'라고 단순화하는 것이다. T3처럼 체크포인트 시점에는 활성 상태였더라도 이후 commit이 기록되면 redo 대상이 된다. 또한 abort가 기록된 T4와 crash 시점 미완료인 T2, T5를 같은 미완료 집합으로 뭉개면 복구 절차가 흐려진다.

다시 나오면 이렇게 판단

즉시 갱신 로그 문제는 '체크포인트의 활성 집합 확인 -> 이후 commit/abort 여부 확인 -> crash 시점 미완료는 undo, commit 완료는 redo, 체크포인트 이전 완료는 제외' 순서로 판단하면 된다. 보기에서 undo와 redo를 뒤바꾸는 표현이 나오면 그 선지를 먼저 의심하면 된다.

NoteBase

암기 포인트

즉시 갱신 복구는 winner redo, loser undo, checkpoint 이전 완료는 제외이다.

NoteBase

문 22. 데이터베이스 관리자(DBA)와 일반 사용자(Kim, Lee, Park, Choi, Cho, Hong)가 데이터베이스 권한 관리를 위한 SQL문을 다음 순서대로 모두 수행하였다. 이에 대한 설명으로 옳은 것은? (단, SQL문 앞의 'OOO>'는 해당 문장을 수행한 사람을 의미하고, 각 SQL문의 문법(syntax) 오류는 발생하지 않는다)

```
DBA> GRANT SELECT ON 고객 TO Kim WITH
     GRANT OPTION;
DBA> GRANT SELECT ON 고객 TO Lee WITH
     GRANT OPTION;
Kim> GRANT SELECT ON 고객 TO Park WITH
     GRANT OPTION;
Lee> GRANT SELECT ON 고객 TO Choi WITH
     GRANT OPTION;
DBA> REVOKE SELECT ON 고객 FROM Kim
     CASCADE;
DBA> REVOKE SELECT ON 고객 FROM Lee
     RESTRICT;
```

① Kim은 고객 테이블에 대한 검색(SELECT) 권한을 가진다.

② Lee는 고객 테이블에 대한 검색(SELECT) 권한을 가진다.

③ Park은 "GRANT SELECT ON 고객 TO Cho"를 실행할 수 있다.

④ Choi는 "GRANT SELECT ON 고객 TO Hong"를 실행할 수 없다.

정답 2번

출제 포인트

GRANT OPTION으로 전파된 권한이 REVOKE의 CASCADE와 RESTRICT에 따라 어떻게 소멸하거나 유지되는지를 실행 순서대로 판단하는 문제이다.

NoteBase

올바른 순서

1. DBA가 Kim에게 SELECT 권한을 WITH GRANT OPTION으로 부여한다.
2. DBA가 Lee에게 SELECT 권한을 WITH GRANT OPTION으로 부여한다.
3. Kim이 Park에게 SELECT 권한을 WITH GRANT OPTION으로 부여한다.
4. Lee가 Choi에게 SELECT 권한을 WITH GRANT OPTION으로 부여한다.
5. DBA가 Kim의 권한을 CASCADE로 취소하므로 Kim이 받은 권한과 Kim이 Park에게 준 권한이 함께 사라진다.
6. DBA가 Lee의 권한을 RESTRICT로 취소하려 하지만 Lee가 Choi에게 준 권한이 남아 있으므로 취소가 허용되지 않아 Lee와 Choi의 권한이 유지된다.

왜 이 순서인가: CASCADE는 하위 전파 권한까지 연쇄 삭제하지만, RESTRICT는 의존하는 하위 권한이 있으면 취소 자체가 성립하지 않기 때문이다.

선지별 해설

① X
Kim은 5단계에서 DBA가 Kim에게 부여한 SELECT 권한을 CASCADE로 취소하면서 권한을 잃는다.
틀린 표현: "Kim은 고객 테이블에 대한 검색 권한을 가진다"
올바르게는: "Kim은 REVOKE ... CASCADE 이후 고객 테이블에 대한 검색 권한을 가지지 않는다"

② O
Lee는 6단계에서 Choi에게 다시 부여한 권한이 남아 있어 REVOKE ... RESTRICT가 허용되지 않으므로 SELECT 권한을 계속 가진다.

③ X
Park은 Kim으로부터 받은 권한이므로 5단계의 CASCADE 취소에 연쇄적으로 함께 삭제된다. 따라서 그 이후에는 다른 사용자에게 권한을 줄 수 없다.
틀린 표현: "Park은 GRANT SELECT ON 고객 TO Cho를 실행할 수 있다"
올바르게는: "Park은 CASCADE 취소 후 권한 자체가 사라져 GRANT SELECT ON 고객 TO Cho를 실행할 수 없다"

④ X
Choi는 Lee로부터 WITH GRANT OPTION으로 권한을 받았다. 6단계의 RESTRICT 취소가 실패하므로 Choi의 권한과 GRANT OPTION도 유지된다.
틀린 표현: "Choi는 GRANT SELECT ON 고객 TO Hong을 실행할 수 없다"
올바르게는: "Choi는 권한과 GRANT OPTION이 유지되므로 GRANT SELECT ON 고객 TO Hong을 실행할 수 있다"

NoteBase

오답 포인트

REVOKE가 실행되었다는 사실만 보고 Lee의 권한도 사라진다고 판단하면 틀린다. 이 문항의 함정은 CASCADE와 RESTRICT를 같은 효과로 보는 데 있다. Kim 경로는 하위 전파 권한까지 함께 끊기지만, Lee 경로는 하위 권한인 Choi의 권한이 남아 있어 RESTRICT 취소가 거부된다는 점이 핵심이다.

다시 나오면 이렇게 판단

권한 문제는 부여 경로를 화살표로 그린 뒤 REVOKE 방식을 확인하면 된다. CASCADE가 붙으면 해당 사용자와 그 사용자로부터 파생된 하위 권한까지 함께 지워지고, RESTRICT가 붙으면 하위 권한이 하나라도 남아 있으면 취소가 되지 않는다고 판단하면 된다.

암기 포인트

CASCADE는 아래로 같이 삭제되고, RESTRICT는 아래 가지가 남아 있으면 삭제가 막힌다.

NoteBase

문 23. 뷰(view)에 대한 설명으로 옳지 않은 것은?

① 사용자의 요구에 맞게 가공하여 만들어 쓸 수 있다.

② 사용자별로 데이터 접근을 제한하여 보안성을 높일 수 있다.

③ 자주 사용되는 질의를 뷰로 미리 정의해 놓고 재사용할 수 있다.

④ 뷰의 기본적인 활용 용도는 검색(SELECT)이며, 뷰를 통한 원본 테이블의 데이터 조작(INSERT 등)은 불가능하다.

정답 4번

출제 포인트

이 문제는 뷰의 활용 목적과 갱신 가능성에 대한 기본 개념을 정확히 구분하는지를 묻는 문제이다.

핵심 개념

뷰는 하나 이상의 기본 테이블에서 필요한 데이터만 조회하도록 정의한 가상 테이블이다.
갱신 가능한 뷰는 단순한 구조처럼 일정한 조건을 만족하면 뷰를 통해서도 INSERT, UPDATE, DELETE가 가능한 뷰이다.
핵심 차이축은 '조회용으로 많이 쓴다'와 '항상 조회만 가능하다'의 구분이다.

선지별 해설

① O: 뷰는 사용자의 요구에 맞게 필요한 열이나 행만 보여 주도록 가공하여 만들 수 있으므로 옳다.
② O: 사용자별로 필요한 데이터만 보이게 하거나 일부 열을 감추는 방식으로 접근을 제한할 수 있으므로 보안성 향상에 활용할 수 있다.
③ O: 자주 사용하는 질의를 뷰로 정의해 두면 같은 SELECT 문을 반복 작성하지 않고 재사용할 수 있으므로 옳다.
④ X: 틀린 부분은 '원본 테이블의 데이터 조작(INSERT 등)은 불가능하다'이다. 뷰의 주된 용도가 검색인 것은 맞지만, 단순 뷰처럼 갱신 가능한 조건을 만족하면 뷰를 통해서도 INSERT, UPDATE, DELETE가 가능하다.
올바르게는 '뷰는 주로 검색에 사용되지만, 갱신 가능한 뷰는 원본 테이블에 대한 데이터 조작도 가능할 수 있다'이다.

오답 포인트

④는 '뷰=가상 테이블=조회 전용'이라는 기억을 절대 표현인 '불가능하다'로 바꾸어 함정을 만든 선지이다.
수험생이 뷰의 일반적 용도와 갱신 가능 여부의 예외 조건을 분리하지 않으면 쉽게 틀릴 수 있다.

NoteBase

다시 나오면 이렇게 판단

뷰 관련 선지에서는 먼저 보안 강화, 질의 단순화, 재사용성처럼 대표 용도를 확인하고, 그다음 '항상', '불가능', '반드시' 같은 절대 표현이 붙으면 갱신 가능한 뷰 예외가 있는지를 따져 판단한다.

암기 포인트

뷰는 가상 테이블이고, 주용도는 조회이지만 단순 뷰는 갱신도 가능할 수 있다.

NoteBase

문 24. NoSQL 데이터베이스에 대한 설명으로 옳지 않은 것은?

① NoSQL 시스템으로 MongoDB 등이 있다.

② 비정형 데이터를 포함하는 빅데이터를 효율적으로 저장하고 처리할 수 있다.

③ 대표적인 데이터 모델에는 키-값, 문서 기반, 컬럼 기반, 그래프 기반 데이터 모델이 있다.

④ CAP 이론은 병행성(Concurrency), 가용성(Availability), 분할 감내성(Partition tolerance)을 모두 보장할 수는 없다는 성질을 말한다.

정답 4번

출제 포인트

이 문제는 NoSQL의 대표적 데이터 모델과 활용 특징, 그리고 CAP 이론의 핵심 용어를 정확히 구분하는지를 묻는 문제이다.

핵심 개념

NoSQL은 전통적인 관계형 모델 외의 다양한 데이터 모델을 사용하는 비관계형 데이터베이스 계열이다.
스키마가 유연하고 대규모 분산 저장에 적합하다는 점이 핵심이다.
CAP 이론은 분산 시스템에서 일관성, 가용성, 분할 감내성 세 성질을 동시에 완전하게 만족시키기 어렵다는 정리이다.
대표 모델은 키-값, 문서 기반, 컬럼 기반, 그래프 기반으로 구분하는 것이 일반적이다.

선지별 해설 ①

O이다. MongoDB는 대표적인 문서 기반 NoSQL 데이터베이스이므로 NoSQL 시스템의 예로 드는 것이 맞다.

선지별 해설 ②

O이다. NoSQL은 정형 데이터뿐 아니라 반정형·비정형 데이터를 포함한 대용량 데이터를 분산 환경에서 저장하고 처리하는 데 적합하다.

선지별 해설 ③

O이다. NoSQL의 대표적 데이터 모델로 키-값, 문서 기반, 컬럼 기반, 그래프 기반 모델을 드는 것이 일반적이므로 서술이 맞다.

선지별 해설 ④

X이다. 틀린 부분은 "병행성(Concurrency)"이다. CAP 이론의 C는 병행성이 아니라 일관성(Consistency)이다. 따라서 이 선지는 "CAP 이론은 일관성, 가용성, 분할 감내성을 모두 동시에 완전하게 보장할 수 없다는 성질을 말한다"로 고쳐야 맞다.

NoteBase

오답 포인트

④는 CAP의 C를 Concurrency로 바꿔 넣은 함정이다. 약어 문제에서는 영어 첫 글자만 보고 익숙한 컴퓨터 용어로 착각하기 쉬우므로, 분산 시스템 문맥에서 C는 일관성이라는 점을 정확히 기억해야 한다.

다시 나오면 이렇게 판단

NoSQL 문제는 먼저 대표 모델이 무엇인지 확인하고, CAP 문제가 나오면 C-A-P를 일관성-가용성-분할 감내성으로 바로 대응시키면 판단이 빨라진다. 약어의 풀네임이 하나라도 바뀌면 그 선지는 틀린 선지로 본다.

암기 포인트

NoSQL 대표 모델은 키-값·문서·컬럼·그래프이고, CAP는 Consistency-Availability-Partition tolerance이다.

NoteBase

문 25. 분산 데이터베이스 시스템에 대한 설명으로 옳지 않은 것은?

① 중앙 집중식 데이터베이스 시스템에 비해 확장이 용이하다.

② 중앙 집중식 데이터베이스 시스템에 비해 더 많은 구축 비용이 소요된다.

③ 한 지역에서 장애가 발생하더라도 전체 시스템이 작업을 계속 수행할 수 있다.

④ 수직적 단편화는 관계대수 연산인 셀렉트(select)를 통해 릴레이션을 속성 단위로 분할하여 저장하는 것을 말한다.

정답 4번

출제 포인트

이 문제는 분산 데이터베이스 시스템의 장단점과 단편화 방식, 특히 수평적 단편화와 수직적 단편화의 구분을 묻는 문제이다.

핵심 개념

분산 데이터베이스 시스템은 여러 지역에 분산된 데이터베이스를 네트워크로 연결해 하나의 시스템처럼 운영하는 구조이다.
수평적 단편화는 릴레이션을 튜플 단위로 나누는 방식이며 관계대수의 셀렉트(select)를 사용한다.
수직적 단편화는 릴레이션을 속성 단위로 나누는 방식이며 관계대수의 프로젝트(project)를 사용한다.

선지별 해설

① O
중앙 집중식 데이터베이스보다 노드를 추가하는 방식으로 시스템 확장이 가능하므로 확장성이 좋다.

② O
여러 지역의 장비, 통신망, 분산 제어 기능이 필요하므로 중앙 집중식보다 구축과 운영 비용이 커지기 쉽다.

③ O
한 지역에 장애가 발생해도 다른 지역 노드가 계속 동작할 수 있어 전체 시스템의 가용성과 신뢰성이 높다.

④ X
틀린 부분은 "수직적 단편화는 관계대수 연산인 셀렉트(select)를 통해 릴레이션을 속성 단위로 분할한다"는 표현이다.
실제로는 수직적 단편화는 프로젝트(project) 연산으로 속성 단위 분할을 수행한다.
셀렉트(select)는 수평적 단편화에서 튜플 단위 분할에 사용된다.

NoteBase

오답 포인트

수평적 단편화와 수직적 단편화를 튜플 기준인지 속성 기준인지로 구분하지 않고, select와 project를 서로 바꾸어 외우면 ④를 맞는 설명처럼 보게 된다.

다시 나오면 이렇게 판단

단편화 문제가 나오면 먼저 분할 기준이 행인지 열인지부터 본다. 행이면 수평적 단편화와 select, 열이면 수직적 단편화와 project로 연결하면 된다.

암기 포인트

수평은 튜플-select, 수직은 속성-project이다.

NoteBase