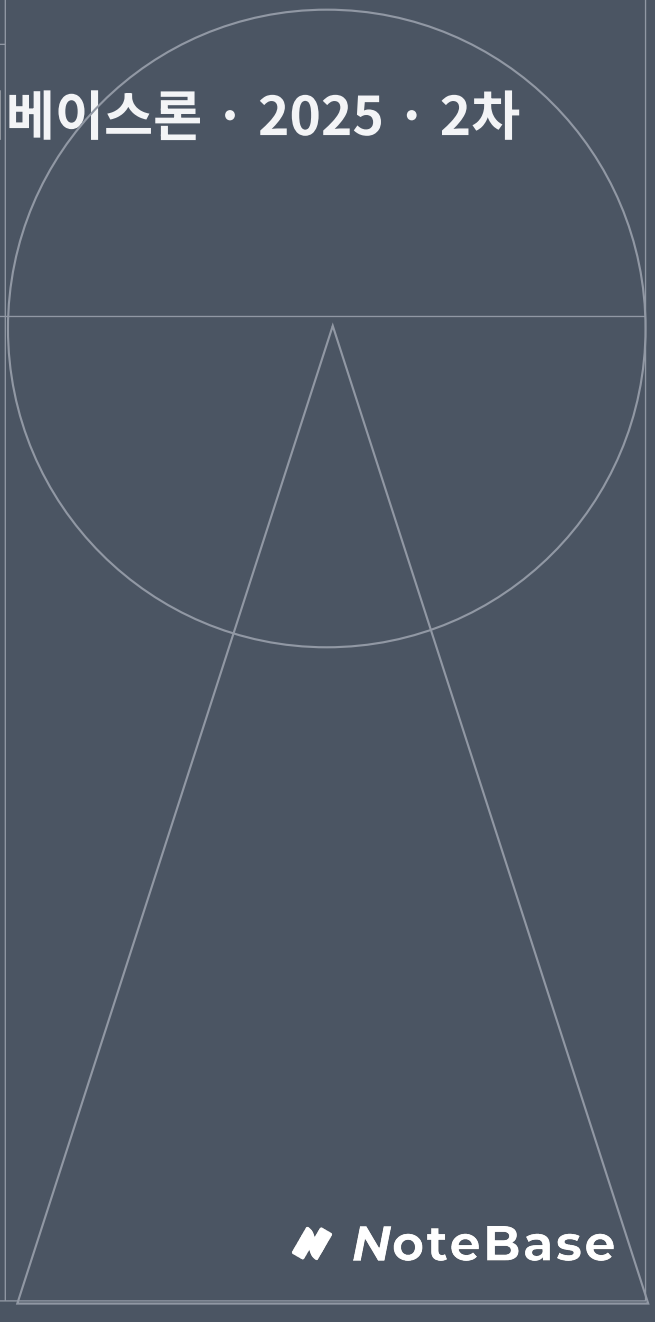
기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2025 · 2차



NoteBase

기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2025 · 2차

공무원/국가직/7급 · 데이터베이스론 · 2025 · 2차

문 1. 디스크에 저장된 사용자 데이터베이스나 시스템 카탈로그 접근에 대하여 책임지는 역할을 하는 데이터베이스 관리시스템의 구성 요소는?

① 질의 처리기

② 트랜잭션 관리자

③ 저장 데이터 관리자

④ 런타임 데이터베이스 처리기

정답 3번

출제 포인트

이 문제는 DBMS 구성요소 중 디스크에 저장된 데이터베이스와 시스템 카탈로그에 직접 접근하는 역할이 무엇인지를 구별할 수 있는지를 묻는다.

핵심 개념

질의 처리기는 사용자의 질의를 해석하고 실행 계획으로 바꾸는 구성요소이다. 트랜잭션 관리자는 동시성 제어와 회복을 통해 트랜잭션의 일관성을 보장하는 구성요소이다. 저장 데이터 관리자는 디스크에 저장된 사용자 데이터베이스와 시스템 카탈로그에 접근하여 실제 저장 데이터를 읽고 쓰는 구성요소이다. 런타임 데이터베이스 처리기는 실행된 질의를 따라 필요한 데이터 접근 명령을 수행하도록 연결하는 실행 단계 처리기이다.

NoteBase

선지별 해설

1번 X: 질의 처리기는 질의의 해석, 최적화, 실행 계획 수립이 중심 역할이다. 틀린 부분은 디스크 저장 데이터와 시스템 카탈로그에 대한 직접 접근 책임으로 본 점이다. 올바르게는 질의 처리기는 질의를 처리하는 상위 단계 구성요소이다.
2번 X: 트랜잭션 관리자는 원자성, 일관성, 고립성, 지속성과 같은 트랜잭션 성질을 유지하는 역할이 중심이다. 틀린 부분은 저장 데이터 접근 책임을 트랜잭션 관리자에 배정한 점이다. 올바르게는 트랜잭션 관리자는 접근의 순서와 회복을 관리한다.
3번 O: 저장 데이터 관리자는 디스크에 저장된 사용자 데이터베이스와 시스템 카탈로그에 접근하는 역할을 담당한다. 문제의 핵심 표현인 디스크에 저장된 데이터와 카탈로그 접근 책임에 정확히 대응한다.
4번 X: 런타임 데이터베이스 처리기는 실행 시점에 질의 실행을 진행시키는 구성요소로 이해하는 것이 맞다. 틀린 부분은 디스크 저장 데이터 접근의 주된 책임을 런타임 데이터베이스 처리기에 둔 점이다. 올바르게는 실제 저장 데이터 접근을 총괄하는 구성요소는 저장 데이터 관리자이다.

오답 포인트

질의 처리기와 런타임 데이터베이스 처리기는 모두 질의 실행과 관련되어 보여 혼동하기 쉽고, 트랜잭션 관리자도 데이터 접근과 함께 언급되는 경우가 많아 저장 데이터 관리자와 역할이 섞이기 쉽다. 이 문항은 접근 대상이 디스크의 사용자 데이터베이스와 시스템 카탈로그라는 표현을 보고 저장 계층 담당자를 고르는지가 함정이다.

다시 나오면 이렇게 판단

선지에서 묻는 역할이 질의 해석인지, 트랜잭션 통제인지, 실제 저장 데이터 접근인지로 먼저 나눈다. 디스크에 저장된 데이터베이스나 시스템 카탈로그 접근이 나오면 저장 데이터 관리자를 우선 떠올리면 된다.

암기 포인트

질의는 질의 처리기, 일관성은 트랜잭션 관리자, 디스크 저장 데이터 접근은 저장 데이터 관리자이다.

NoteBase

문 2. 다음은 데이터베이스 설계 단계이다. (가) ~ (다)에 들어갈 내용을 바르게 연결한 것은?

요구 조건 분석 → (가) → (나) → (다) → 구현

① (가): 개념적 설계, (나): 논리적 설계, (다): 물리적 설계
② (가): 논리적 설계, (나): 개념적 설계, (다): 물리적 설계
③ (가): 물리적 설계, (나): 개념적 설계, (다): 논리적 설계
④ (가): 물리적 설계, (나): 논리적 설계, (다): 개념적 설계

정답 1번

출제 포인트

이 문제는 데이터베이스 설계의 표준 진행 순서인 개념적 설계, 논리적 설계, 물리적 설계를 요구 조건 분석과 구현 사이에 올바르게 배치하는지를 묻는 문제이다.

올바른 순서

1. 개념적 설계 — 요구 사항을 엔터티, 속성, 관계 등 개념 모델로 추상화하는 단계이다.
2. 논리적 설계 — 개념 모델을 관계 스키마 등 DBMS 독립적인 논리 구조로 변환하는 단계이다.
3. 물리적 설계 — 논리 스키마를 실제 저장 구조, 인덱스, 접근 경로 등으로 구체화하는 단계이다.

왜 이 순서: 논리적 설계는 개념적 설계의 결과물을 바탕으로 하고, 물리적 설계는 논리적 설계에서 정리된 스키마를 실제 저장 방식으로 구현하기 위한 전제 단계이므로 개념적 설계 → 논리적 설계 → 물리적 설계 순서가 된다.

NoteBase

선지별 해설

① O
- (가)=개념적 설계, (나)=논리적 설계, (다)=물리적 설계로 연결한 것이다. 요구 조건 분석 뒤에는 먼저 현실 세계를 개념적으로 모델링하고, 그다음 이를 논리 스키마로 바꾸며, 마지막으로 저장 구조를 정한 뒤 구현으로 넘어간다.

② X
- 틀린 부분: (가)를 논리적 설계, (나)를 개념적 설계로 둔 부분이 틀리다.
- 올바르게는: 개념적 설계가 먼저이고 그 결과를 바탕으로 논리적 설계를 수행한다.

③ X
- 틀린 부분: (가)를 물리적 설계로 두고 (다)를 논리적 설계로 둔 부분이 틀리다.
- 올바르게는: 물리적 설계는 논리적 설계보다 뒤에 와야 하므로 개념적 설계 → 논리적 설계 → 물리적 설계 순서가 맞다.

④ X
- 틀린 부분: (가)를 물리적 설계, (다)를 개념적 설계로 둔 부분이 틀리다.
- 올바르게는: 개념적 설계가 가장 먼저 오고 물리적 설계가 가장 나중에 온다. 구현 직전 단계는 개념적 설계가 아니라 물리적 설계이다.

오답 포인트

개념적 설계와 논리적 설계를 뒤바꾸거나, 구현 직전에 오는 단계를 개념적 설계로 착각하는 경우가 많다. 이 문항의 함정은 세 단계의 이름을 모두 아는 수험생도 산출물의 추상화 수준을 기준으로 정리하지 않으면 ②, ④처럼 순서를 바꿔 고르기 쉽다는 점이다.

다시 나오면 이렇게 판단

데이터베이스 설계 순서 문제는 추상화 수준이 높은 단계에서 낮은 단계로 내려간다고 판단하면 된다. 현실 세계를 모델링하면 개념적 설계이고, 이를 스키마로 바꾸면 논리적 설계이며, 저장 방식과 접근 경로를 정하면 물리적 설계이고, 그 다음이 구현이다.

암기 포인트

요구 분석 뒤에는 개념, 논리, 물리 순서로 내려가고 물리 다음이 구현이다.

NoteBase

문 3. 직원(사번, 이름, 부서, 전화번호, 월급, 고용일) 테이블에 대한 유효한 표준 SQL문으로 옳지 않은 것은? (단, 밑줄 속성은 기본키이다)

①
```
SELECT 사번, 월급
FROM 직원
ORDER BY 월급;
```

②
```
SELECT *
FROM 직원
WHERE 이름 LIKE '김%';
```

③
```
SELECT AVG(월급)
FROM 직원
WHERE 부서 = '인사팀';
```

④
```
SELECT SUM(부서, 월급)
FROM 직원
GROUP BY 부서;
```

정답 4번

출제 포인트

이 문제는 SELECT문에서 ORDER BY, LIKE, 집계함수, GROUP BY의 문법 규칙을 정확히 구별할 수 있는지를 묻는다.

핵심 개념

ORDER BY는 조회된 결과를 정렬하는 절이다.
LIKE '김%'는 이름이 '김'으로 시작하는 행을 찾는 문자열 패턴 조건이다.
AVG, SUM 같은 집계함수는 한 개의 식을 입력으로 받아 전체 또는 그룹별 값을 계산한다.
SELECT절에 집계함수와 일반 컬럼을 함께 두면 일반 컬럼은 GROUP BY 기준과 맞아야 한다.

NoteBase

선지별 해설

1번 O: `SELECT 사번, 월급 FROM 직원 ORDER BY 월급;`은 사번과 월급을 조회한 뒤 월급 기준으로 정렬하는 문장이다. 조회 컬럼과 정렬 컬럼의 사용이 자연스럽고 문법상 문제도 없다.
2번 O: `SELECT * FROM 직원 WHERE 이름 LIKE '김%';`은 이름이 '김'으로 시작하는 직원을 찾는 조건문이다. `LIKE`와 `%`의 사용이 올바르므로 유효한 SQL문이다.
3번 O: `SELECT AVG(월급) FROM 직원 WHERE 부서 = '인사팀';`은 인사팀 행만 먼저 걸러 낸 다음 그 집합의 평균 월급을 구하는 문장이다. SELECT절에 집계값만 있으므로 GROUP BY가 없어도 문법상 유효하다.
4번 X: 틀린 부분은 `SUM(부서, 월급)`이다. `SUM`은 숫자형 식 하나만 인자로 받는데 `부서`와 `월급` 두 개를 함께 넣었으므로 표준 SQL 문법에 맞지 않는다. 부서별 월급 합계를 구하려면 `SELECT 부서, SUM(월급) FROM 직원 GROUP BY 부서;`로 고쳐야 한다.

오답 포인트

3번과 4번을 함께 보면 집계함수 관련 규칙이 함정이다. 많은 수험생이 집계가 나오면 무조건 GROUP BY가 필요하다고 생각하거나, GROUP BY만 있으면 집계함수 인자를 아무렇게나 써도 된다고 착각한다. 이 문항의 실제 함정은 GROUP BY의 존재가 아니라 `SUM`의 인자 형태이다.

다시 나오면 이렇게 판단

집계함수가 보이면 먼저 함수가 받는 인자 개수를 확인하고, 다음으로 SELECT절에 일반 컬럼이 섞였는지 본다. 일반 컬럼이 있으면 그 컬럼이 GROUP BY 기준과 일치하는지 확인하면 된다. WHERE는 집계 전에 행을 거르는 절이고 ORDER BY는 마지막에 결과를 정렬하는 절이라는 순서로 판단하면 흔들리지 않는다.

암기 포인트

집계함수는 한 식을 계산하고, 일반 컬럼과 함께 조회하면 일반 컬럼은 GROUP BY에 맞춘다.

NoteBase

문 4. 트랜잭션의 고립성 수준(isolation level) 중 반복할 수 없는 읽기(non-repeatable read) 문제가 있는 것만을 모두 고르면?

ㄱ. read committed
ㄴ. read uncommitted
ㄷ. repeatable read
ㄹ. serializable

① ㄱ, ㄴ
② ㄴ, ㄷ
③ ㄷ, ㄹ
④ ㄱ, ㄴ, ㄷ

정답 1번

출제 포인트

이 문제는 트랜잭션 격리수준별로 어떤 이상현상을 허용하는지를 구별할 수 있는지를 묻는다.

핵심 개념

read uncommitted는 커밋 전 데이터까지 읽을 수 있는 가장 낮은 격리수준으로 dirty read, non-repeatable read, phantom read가 모두 발생할 수 있다.
read committed는 커밋된 데이터만 읽으므로 dirty read는 막지만 같은 행을 다시 읽을 때 값이 바뀌는 non-repeatable read와 phantom read는 발생할 수 있다.
repeatable read는 같은 행을 다시 읽을 때 같은 값을 보장하므로 non-repeatable read는 막지만 phantom read는 발생할 수 있다.
serializable은 동시 실행 결과를 직렬 실행과 같게 맞추는 가장 높은 격리수준으로 non-repeatable read를 포함한 이상현상을 방지한다.

선지별 해설

1번 O: ㄱ read committed와 ㄴ read uncommitted는 모두 non-repeatable read가 발생할 수 있으므로 맞는 조합이다.
2번 X: 틀린 부분은 'ㄴ, ㄷ'이다. ㄷ repeatable read는 같은 행의 재조회 값을 보장하므로 non-repeatable read가 발생하지 않는다. 올바르게는 'ㄱ, ㄴ'이다.
3번 X: 틀린 부분은 'ㄷ, ㄹ'이다. ㄷ repeatable read와 ㄹ serializable은 모두 non-repeatable read를 방지한다. 올바르게는 'ㄱ, ㄴ'이다.
4번 X: 틀린 부분은 'ㄱ, ㄴ, ㄷ'이다. ㄷ repeatable read를 포함시킨 것이 잘못되었다. 올바르게는 'ㄱ, ㄴ'이다.

NoteBase

오답 포인트

repeatable read의 이름만 보고 모든 반복 조회 문제와 팬텀 읽기까지 모두 막는다고 섞어 기억하면 틀리기 쉽다. 이 문항은 '반복 불가능 읽기'만 묻는데 ㄷ을 끼워 넣는지 여부가 함정이다.

다시 나오면 이렇게 판단

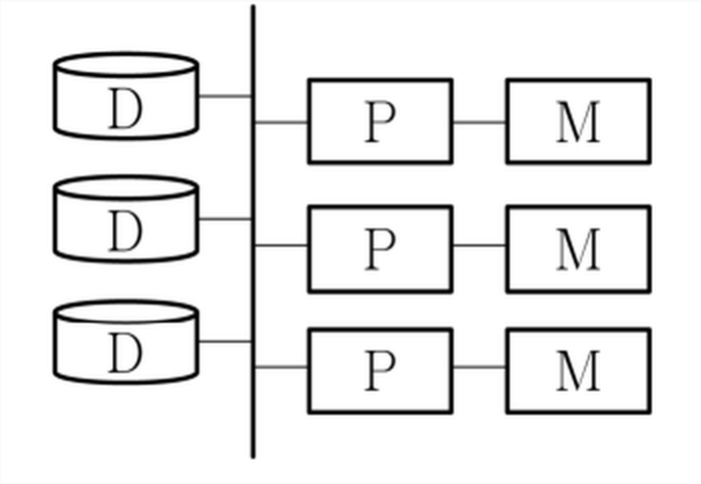
격리수준 문제는 dirty read, non-repeatable read, phantom read를 낮은 수준부터 하나씩 제거해 가는 순서로 판단하면 된다. non-repeatable read는 read committed까지는 허용되고 repeatable read부터는 차단된다고 보면 된다.

암기 포인트

RU는 다 허용, RC는 dirty만 차단, RR은 non-repeatable까지 차단, Serializable은 전부 차단이다.

NoteBase

문 5. 병렬 데이터베이스에서 다음과 같이 구성된 병렬 장치를 위한 구조 모델은? (단, D는 디스크, M은 메모리, P는 프로세서이다)



① 비공유(shared nothing) 모델

② 공유 디스크(shared disk) 모델

③ 공유 메모리(shared memory) 모델

④ 공유 프로세서(shared processor) 모델

정답 2번

출제 포인트

이 문제는 병렬 데이터베이스의 자원 공유 방식에서 디스크, 메모리, 프로세서 중 무엇이 공유되는지를 도식으로 판별하는 문제이다.

선지별 해설

① X: 비공유(shared nothing) 모델은 각 노드가 디스크, 메모리, 프로세서를 모두 독립적으로 가져야 한다. 이 도식은 디스크가 공통 연결부를 통해 여러 노드에 공유되므로 비공유 모델이 아니다. 틀린 표현은 '비공유'이고, 올바르게는 '디스크가 공유된다'이다.
② O: 공유 디스크(shared disk) 모델은 각 노드가 자신의 프로세서와 메모리를 가지면서 디스크를 공동으로 접근하는 구조이다. 그림이 바로 그 형태를 나타낸다.
③ X: 공유 메모리(shared memory) 모델이면 여러 프로세서가 하나의 공통 메모리를 함께 사용해야 한다. 이 도식은 메모리 M이 노드마다 따로 그려져 있으므로 공유 메모리 모델이 아니다. 틀린 표현은 '공유 메모리'이고, 올바르게는 '메모리는 공유되지 않고 디스크만 공유된다'이다.
④ X: 공유 프로세서 모델이라면 프로세서가 공통 자원으로 묶여 있어야 하나, 이 도식은 프로세서 P가 노드별로 따로 존재한다. 틀린 표현은 '공유 프로세서'이고, 올바르게는 '프로세서는 각 노드가 독립적으로 가진다'이다.

NoteBase

오답 포인트

수험생은 디스크가 여러 개 보인다는 이유만으로 비공유 모델로 오인하기 쉽다. 이 문제의 함정은 자원의 개수가 아니라 무엇이 공통 연결부에 붙어 있는지를 보아야 한다는 점이다. 메모리가 노드별로 분리된 사실을 놓치면 ③으로도 흔들리기 쉽다.

다시 나오면 이렇게 판단

병렬 데이터베이스 구조 문제는 먼저 공통으로 묶인 자원이 무엇인지 본다. 디스크만 공통이면 shared disk, 메모리까지 공통이면 shared memory, 디스크와 메모리와 프로세서가 모두 노드별 독립이면 shared nothing으로 판단된다.

암기 포인트

shared disk는 P와 M은 각자, D만 함께 쓰는 구조이다.

NoteBase

문 6. 반정형 데이터(semi-structured data)에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 동영상, 이미지, 음성 파일이 해당한다.
ㄴ. 데이터 구조를 파악하기 위한 과정이 필요하다.
ㄷ. 데이터 내용 안에 구조 정보가 함께 저장된다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 3번

출제 포인트

이 문제는 반정형 데이터와 비정형 데이터의 구별, 그리고 반정형 데이터의 구조 표현 방식을 묻는 문제이다.

핵심 개념

반정형 데이터는 고정된 테이블 스키마로 완전히 정리되지는 않지만 태그, 키-값, 메타데이터처럼 구조 정보가 데이터와 함께 들어 있는 데이터이다.
정형 데이터는 미리 정해진 스키마에 맞춰 행과 열로 저장되는 데이터이다.
비정형 데이터는 고정 구조가 거의 없어서 내용만으로는 바로 필드 구조를 나누기 어려운 데이터이며 동영상, 이미지, 음성 파일이 대표적이다.

선지별 해설

1번 X: 1번은 ㄱ, ㄴ 조합이다. ㄴ은 맞지만 ㄱ이 틀려서 오답이다. 틀린 표현은 ㄱ의 "동영상, 이미지, 음성 파일이 해당한다"이다. 반정형 데이터의 대표 예시는 XML, HTML, JSON 같은 태그 또는 키-값 구조 데이터이며, 동영상·이미지·음성 파일은 일반적으로 비정형 데이터이다.
2번 X: 2번은 ㄱ, ㄷ 조합이다. ㄷ은 맞지만 ㄱ이 틀려서 오답이다. 틀린 표현은 역시 ㄱ의 매체 예시 부분이다. 올바르게는 "데이터 내용 안에 구조 정보가 함께 저장된다"와 같은 설명이 반정형 데이터에 해당한다.
3번 O: 3번은 ㄴ, ㄷ 조합이다. ㄴ은 맞다. 정형 데이터는 저장 전에 고정된 스키마가 미리 정의되어 있어(schema-on-write) 구조를 따로 파악할 필요가 없지만, 반정형 데이터는 저장 시점에 고정 스키마가 강제되지 않고 데이터를 읽는 시점에 태그나 키-값을 해석해 구조를 파악해야 한다(schema-on-read). 따라서 데이터 구조를 파악하기 위한 별도의 과정이 필요하다는 ㄴ의 서술이 반정형 데이터의 특성에 부합한다. ㄷ도 태그나 메타데이터처럼 구조 정보가 내용과 함께 저장된다는 뜻이므로 맞다.
4번 X: 4번은 ㄱ, ㄴ, ㄷ 모두를 맞는 설명으로 본 선택지인데 ㄱ이 틀리므로 오답이다. 틀린 표현은 ㄱ 포함 부분이다. 올바른 조합은 ㄴ, ㄷ만 포함한 3번이다.

NoteBase

오답 포인트

동영상, 이미지, 음성 파일을 "파일 형태가 있으니 어느 정도 구조가 있다"고 생각하면 ㄱ을 맞는 설명으로 오해하기 쉽다. 이 문제의 핵심은 파일 형식 유무가 아니라 데이터 내부에 검색 가능한 구조 정보가 함께 표현되어 있는지 여부이다.

다시 나오면 이렇게 판단

먼저 예시가 XML, HTML, JSON처럼 태그·키-값 구조인지, 아니면 이미지·음성·영상처럼 내용 자체가 구조화되지 않은지 구분한다. 그다음 구조 정보가 데이터 내부에 함께 저장되는지와 읽는 시점에 구조 해석이 필요한지를 확인하면 반정형 데이터 여부를 판단할 수 있다.

암기 포인트

반정형 데이터는 구조 정보가 내용과 함께 있고, 비정형 데이터는 동영상·이미지·음성처럼 고정 구조가 거의 없다.

NoteBase

문 7. 데이터베이스의 시스템 카탈로그에 저장되는 데이터로 옳지 않은 것은?

① 속성의 데이터 타입

② 테이블에 저장된 레코드 값

③ 데이터에 대한 다양한 제약 조건

④ 테이블 이름, 인덱스(index) 이름, 뷰(view) 이름

정답 2번

출제 포인트

이 문제는 시스템 카탈로그가 저장하는 메타데이터와 실제 테이블에 저장되는 데이터 인스턴스를 구분하는 문제이다.

핵심 개념

시스템 카탈로그는 데이터베이스 객체의 정의와 구조를 저장하는 메타데이터 저장소이다. 테이블 이름, 속성 이름과 데이터 타입, 무결성 제약조건, 인덱스와 뷰 같은 객체 정의는 저장하지만, 각 행의 실제 값은 사용자 테이블에 저장된다.

선지별 해설

① O
속성의 데이터 타입은 스키마 정보이므로 시스템 카탈로그에 저장된다.

② X
틀린 부분은 "테이블에 저장된 레코드 값"이다.
실제로 레코드 값은 사용자 데이터 자체이므로 시스템 카탈로그가 아니라 해당 테이블의 데이터 영역에 저장된다.

③ O
데이터에 대한 다양한 제약 조건은 무결성 규칙에 해당하는 메타데이터이므로 시스템 카탈로그에 저장된다.

④ O
테이블 이름, 인덱스 이름, 뷰 이름은 데이터베이스 객체의 정의 정보이므로 시스템 카탈로그에 저장된다.

오답 포인트

시스템 카탈로그를 데이터베이스에 있는 모든 정보의 저장소로 넓게 이해하면 ②도 맞아 보이기 쉽다. 그러나 카탈로그는 실제 데이터가 아니라 데이터를 설명하는 구조 정보만 저장한다는 점이 핵심 함정이다.

NoteBase

다시 나오면 이렇게 판단

선지 내용이 객체의 이름, 속성, 타입, 제약조건처럼 스키마를 설명하면 카탈로그 대상이다. 반대로 각 행의 값, 실제 레코드 내용처럼 인스턴스 데이터 자체를 말하면 카탈로그 대상이 아니다.

암기 포인트

시스템 카탈로그는 메타데이터 저장소이고, 레코드 값은 사용자 데이터이다.

NoteBase

문 8. 빅데이터 분석 및 처리를 위한 기술인 하둡(Hadoop)에 대한 설명으로 옳은 것은?

① 맵리듀스(MapReduce) 기법을 사용하는 파일 시스템은 병렬 처리를 지원하지 않는다.

② 맵리듀스 기법에서 reduce() 함수의 입력 키 타입은 map() 함수의 출력 키 타입과 다르다.

③ 맵리듀스 기법의 map() 함수는 각 입력 레코드에 대해 출력이 없거나, 한 개 이상의 (키, 값) 쌍들을 출력한다.

④ 하둡 분산 파일 시스템(HDFS)에서 각 파일의 블록을 식별하기 위한 메타데이터를 저장하는 서버는 데이터노드(DataNode)이다.

정답 3번

출제 포인트

이 문제는 하둡의 저장 구조와 맵리듀스의 처리 흐름을 정확히 구분하는지를 묻는 문제이다.

핵심 개념

하둡은 대용량 데이터를 분산 저장하는 HDFS와 분산 처리하는 맵리듀스로 구성된다. map()은 입력 레코드를 중간 (키, 값) 쌍으로 바꾸는 단계이고, reduce()는 같은 키로 모인 값을 집계하는 단계이다. HDFS에서는 NameNode가 파일 이름, 블록 위치 같은 메타데이터를 관리하고 DataNode가 실제 데이터 블록을 저장한다.

선지별 해설

① X: 틀린 부분은 "병렬 처리를 지원하지 않는다"이다. 하둡은 여러 노드에 데이터를 분산 저장하고 맵리듀스로 병렬 처리한다. 올바르게는 "병렬 처리를 지원한다"이다.
② X: 틀린 부분은 "reduce() 함수의 입력 키 타입은 map() 함수의 출력 키 타입과 다르다"이다. reduce 입력 키는 shuffle과 sort를 거친 map 출력 키와 같은 타입이다. 올바르게는 "같다"이다.
③ O: map() 함수는 각 입력 레코드에 대해 조건에 따라 아무 출력도 하지 않을 수 있고, 한 개 이상 중간 (키, 값) 쌍을 출력할 수도 있다. 맵리듀스의 map 단계 특징을 정확히 설명한 선지이다.
④ X: 틀린 부분은 "메타데이터를 저장하는 서버는 데이터노드(DataNode)"이다. HDFS 메타데이터는 NameNode가 관리하고 DataNode는 실제 블록을 저장한다. 올바르게는 "메타데이터를 저장하는 서버는 네임노드(NameNode)"이다.

오답 포인트

하둡 문제에서는 HDFS와 맵리듀스를 한 덩어리로 외우면 ①과 ④에서 흔들리기 쉽다. 또 map 출력과 reduce 입력의 연결 관계를 놓치면 ②처럼 같은 타입이어야 하는 지점을 다르다고 착각하기 쉽다.

NoteBase

다시 나오면 이렇게 판단

하둡 관련 선지는 저장과 처리의 역할을 먼저 분리해서 본다. HDFS는 NameNode와 DataNode의 역할로, 맵리듀스는 map 출력이 reduce 입력으로 이어지는 흐름으로 판단하면 된다.

암기 포인트

NameNode는 메타데이터, DataNode는 블록 저장, map은 0개 이상 출력, reduce는 같은 키를 묶어 집계한다.

NoteBase

문 9. 다음 트랜잭션 T_1, T_2, T_3 스케줄의 충돌 직렬 가능성 검사 시 충돌하는 연산으로 옳지 않은 것은?

시간 ↓

T_1	T_2	T_3
	read(A)	
read(B)		
	write(A)	
	read(B)	
		read(A)
write(B)		
		write(A)
	write(B)	

① T_1의 read(B)와 T_2의 read(B)

② T_1의 write(B)와 T_2의 write(B)

③ T_2의 read(B)와 T_1의 write(B)

④ T_2의 write(A)와 T_3의 read(A)

정답 1번

출제 포인트

이 문제는 스케줄 표에서 같은 데이터 항목에 대한 연산쌍이 읽기-읽기인지, 읽기-쓰기 또는 쓰기-쓰기인지 구분하여 충돌 여부를 판단하는 문제이다. 발문은 '충돌하는 연산으로 옳지 않은 것'을 고르라는 것이므로, 실제로는 충돌하지 않는 연산쌍이 정답이 된다. 자료를 읽을 때는 각 연산이 접근하는 데이터 항목이 A인지 B인지 확인하고, 서로 다른 트랜잭션의 연산쌍만 비교하며, 같은 항목에 대해 읽기-읽기는 비충돌이고 읽기-쓰기·쓰기-읽기·쓰기-쓰기는 충돌로 판단한다.

NoteBase

선지별 해설

1번: O
T1의 read(B)와 T2의 read(B)는 서로 다른 트랜잭션이지만 같은 항목 B를 둘 다 읽기만 한다. 읽기-읽기 조합은 값을 변경하지 않으므로 충돌하지 않는다. 따라서 이 쌍은 '충돌하는 연산'이 아니어서 발문이 요구하는 '옳지 않은 것'에 해당하므로 정답이다.
2번: X
T1의 write(B)와 T2의 write(B)는 같은 항목 B에 대한 쓰기-쓰기 조합이다. 두 연산 모두 값을 변경하므로 이 쌍은 충돌한다. 충돌하는 연산이 맞으므로 발문이 요구하는 정답에는 해당하지 않는다.
3번: X
T2의 read(B)와 T1의 write(B)는 같은 항목 B에 대한 읽기-쓰기 조합이다. 한쪽이 값을 읽고 다른 쪽이 값을 변경하므로 이 쌍은 충돌한다. 충돌하는 연산이 맞으므로 정답에는 해당하지 않는다.
4번: X
T2의 write(A)와 T3의 read(A)는 같은 항목 A에 대한 쓰기-읽기 조합이다. write(A)가 변경한 값에 read(A)가 영향을 받으므로 이 쌍은 충돌한다. 충돌하는 연산이 맞으므로 정답에는 해당하지 않는다.

오답 포인트

같은 데이터 항목을 접근한다는 이유만으로 모두 충돌한다고 보면 1번에서 틀린다. 충돌 여부는 같은 항목인지와 함께 적어도 한쪽이 write인지까지 함께 봐야 한다.

다시 나오면 이렇게 판단

충돌 연산은 서로 다른 트랜잭션이 같은 데이터 항목을 접근하고 그중 적어도 하나가 write인 경우로 판단한다. 먼저 항목이 같은지 보고, 그다음 연산 종류가 read-read인지 아닌지를 확인하면 된다.

암기 포인트

같은 항목 + write 포함이면 충돌이고, read-read만 비충돌이다.

NoteBase

문 10. 관계형 데이터베이스의 질의 처리 방법에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 질의 최적화 모듈은 질의 처리 연산을 실행할 때 소요되는 정확한 비용을 계산한다.
ㄴ. 조인 연산 최적화를 위한 해시 조인 알고리즘은 동등 조인(equijoin)에 적용할 수 없다.
ㄷ. 질의 처리는 관계 대수 표현식으로의 변환, 질의 최적화, 실행 계획 순으로 진행한다.
ㄹ. 두 테이블 중 한 개 또는 두 개가 정렬되어 있는 경우에는 정렬 합병 조인 알고리즘이 유용하다.

① ㄱ, ㄴ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄷ, ㄹ

정답 4번

출제 포인트

이 문제는 질의 최적화의 비용 추정 성격과 조인 알고리즘의 적용 조건을 구별할 수 있는지를 묻는다.

핵심 개념

질의 최적화는 여러 실행 계획 중 예상 비용이 가장 낮은 계획을 고르는 과정이며, 비용은 통계 정보를 바탕으로 추정한다.
해시 조인은 조인 속성의 해시값으로 같은 키를 빠르게 매칭하는 방식이므로 등등 조인에 적합하다.
정렬 합병 조인은 입력이 이미 정렬되어 있거나 정렬 후 병합하는 편이 유리할 때 효과적이다.
질의 처리는 일반적으로 질의의 내부 표현 변환, 최적화, 실행 계획 선택과 실행의 흐름으로 이해할 수 있다.

선지별 해설

1번: X. ㄱ과 ㄴ을 모두 옳다고 본 조합이므로 틀리다. ㄱ의 틀린 표현은 "정확한 비용"이다. 질의 최적화기는 실제 실행 전 정확한 비용을 계산하는 것이 아니라 통계와 카디널리티를 바탕으로 비용을 추정한다. ㄴ의 틀린 표현은 "적용할 수 없다"이다. 해시 조인은 대표적으로 등등 조인에 적용된다. 따라서 올바른 조합은 "ㄱ, ㄴ"이 아니라 "ㄷ, ㄹ"이다.
2번: X. ㄹ은 옳지만 ㄱ이 틀리므로 조합 전체는 틀리다. ㄱ은 비용 추정을 정확 계산으로 바꿔 놓은 것이 오류이다. 따라서 "ㄱ, ㄹ"이 아니라 "ㄷ, ㄹ"이어야 한다.
3번: X. ㄷ은 옳지만 ㄴ이 틀리므로 조합 전체는 틀리다. 해시 조인은 등등 조인에 적용할 수 없다는 부분이 틀렸고, 실제로는 등등 조인에 잘 맞는 알고리즘이다. 따라서 "ㄴ, ㄷ"이 아니라 "ㄷ, ㄹ"이어야 한다.
4번: O. ㄷ과 ㄹ은 모두 옳다. ㄷ은 질의 처리의 큰 흐름을 내부 표현 변환, 최적화, 실행 계획의 수립과 실행으로 본 설명이고, ㄹ은 정렬된 입력에서 정렬 합병 조인이 유리하다는 점을 정확히 짚었다.

NoteBase

오답 포인트

질의 최적화기를 실제 실행 결과를 미리 아는 모듈처럼 생각하면 ㄱ에 속기 쉽다. 또 해시 조인을 비등가 조인 전용이거나 등등 조인에는 부적합한 방식으로 오해하면 ㄴ에 속기 쉽다.

다시 나오면 이렇게 판단

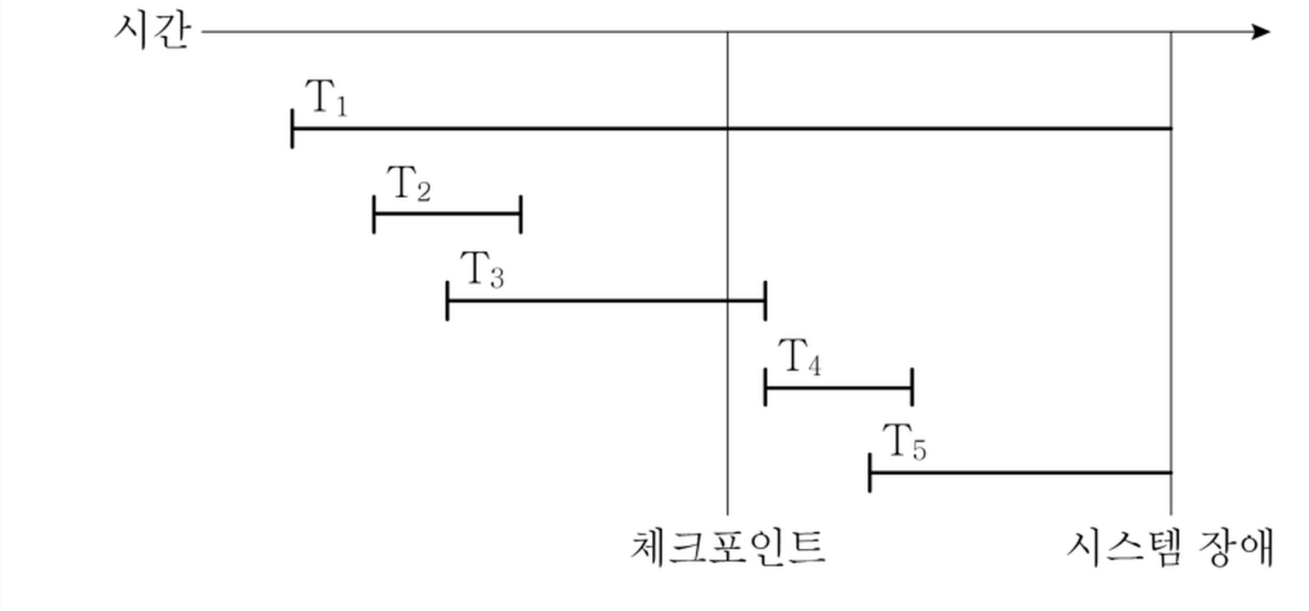
질의 처리 문제는 먼저 최적화기의 역할이 "정확 계산"인지 "비용 추정"인지부터 구분하고, 다음으로 각 조인 알고리즘의 전형적 적용 조건을 대입해 판단하면 된다. 해시 조인은 등등 조인, 정렬 합병 조인은 정렬된 입력이라는 연결고리를 먼저 확인하면 된다.

암기 포인트

최적화기는 비용을 추정하고, 해시 조인은 등등 조인에 강하며, 정렬 합병 조인은 정렬된 입력에서 유리하다.

NoteBase

문 11. 체크포인트 회복 기법을 사용하는 시스템에서 트랜잭션이 수행되는 중 다음과 같이 장애가 발생하였을 때, 취소(UNDO) 연산이 필요한 트랜잭션만을 모두 고르면?



① T_5

② T_1, T_5

③ T_4, T_5

④ T_1, T_3, T_5

정답 2번

출제 포인트

이 문제는 체크포인트 회복에서 시스템 장애 시점까지 종료되지 않은 트랜잭션만 UNDO 대상이라는 점을 도식으로 판별할 수 있는지를 묻는다.

선지별 해설

1번 X: T5는 UNDO 대상이 맞지만 T1도 시스템 장애 시점까지 종료되지 않았으므로 빠져 있다. 틀린 표현은 'T5만'이며, 올바르게는 'T1, T5'이다.
2번 O: T1과 T5는 모두 장애 시점에 미완료 상태이므로 UNDO 대상이다. T1은 체크포인트 이전에 시작했어도 끝나지 않았고, T5도 체크포인트 이후 시작했지만 끝나지 않았다.
3번 X: T4는 체크포인트 이후 시작했지만 시스템 장애 전에 종료된 완료 트랜잭션이므로 UNDO 대상이 아니다. 틀린 표현은 'T4'이며, 올바르게는 'T1, T5'이다.
4번 X: T3는 체크포인트를 지나 수행되었지만 시스템 장애 전에 종료된 완료 트랜잭션이므로 UNDO 대상이 아니다. 틀린 표현은 'T3'이며, 올바르게는 'T1, T5'이다.

NoteBase

오답 포인트

체크포인트 이후에 시작한 트랜잭션만 UNDO한다고 오해하거나, 체크포인트를 지난 트랜잭션은 모두 UNDO한다고 섞어 판단하면 틀린다. 이 문항의 함정은 체크포인트 시점이 아니라 시스템 장애 시점의 완료 여부를 기준으로 보아야 한다는 점이다.

다시 나오면 이렇게 판단

시간축 문제는 각 트랜잭션이 시스템 장애 시점 전에 종료되었는지부터 확인한다. 장애 시점까지 살아 있으면 UNDO, 장애 전에 정상 종료되었으면 UNDO 대상이 아니다.

암기 포인트

UNDO는 장애 시점 미완료 트랜잭션, REDO는 장애 전 커밋 완료 후 반영 확인 대상이라는 구분이 핵심이다.

NoteBase

문 12. '선수' 테이블에서 검색 결과가 다음과 같이 생성되는 SQL문의 (가)에 들어갈 내용은?

선수

선수번호	이름	포지션	점수	팀번호
2123	장영실	CF	30	2
3321	이순신	RB	12	1
2276	정약용	LW	6	1
1925	허준	RW	24	1
9842	김유신	GK	10	2

검색 결과

팀번호	합계점수	최대점수
2	40	30

```
SELECT 팀번호, SUM(점수) AS 합계점수, MAX(점수) AS
       최대점수
FROM 선수
(가)
```

① GROUP BY 팀번호;

② GROUP BY 선수번호;

③ GROUP BY 팀번호 HAVING AVG(점수) >= 15;

④ GROUP BY 팀번호 HAVING MAX(점수) >= 20;

정답 3번

출제 포인트

이 문제는 팀번호로 그룹화한 뒤 집계 결과에서 원하는 팀만 남기도록 HAVING 조건을 고르는 문제이다.

NoteBase

자료 읽는 순서

1. 팀번호를 기준으로 먼저 그룹을 나누어야 합계점수와 최대점수가 팀 단위로 계산된다.
2. 각 팀의 SUM, MAX, AVG를 직접 계산한다.
3. 검색 결과에 팀번호 2만 남는 조건이 무엇인지 HAVING 절로 판단한다.

핵심 근거

팀 1의 점수는 12, 6, 24이므로 합계는 42, 최대는 24, 평균은 14이다.
팀 2의 점수는 30, 10이므로 합계는 40, 최대는 30, 평균은 20이다.
검색 결과는 팀번호 2, 합계점수 40, 최대점수 30 한 행만 보여 주므로 팀번호로 그룹화한 뒤 평균점수가 15 이상인 그룹만 남겨야 한다.

선지별 해설

① X: `GROUP BY 팀번호`만 사용하면 팀 1과 팀 2가 모두 출력된다. 팀 1도 합계점수 42, 최대점수 24로 한 행이 생성된다. 틀린 표현은 조건 없이 그룹화만 하면 된다는 점이며, 올바르게는 `GROUP BY 팀번호 HAVING AVG(점수) >= 15`처럼 팀 1을 제외하는 조건이 추가되어야 한다.
② X: `GROUP BY 선수번호`는 선수별로 한 행씩 묶이므로 SUM과 MAX가 각 선수 자신의 점수와 같아진다. 따라서 팀번호 2 한 행만 나오는 팀별 집계 결과를 만들 수 없다. 틀린 표현은 선수번호로 묶는 부분이며, 올바르게는 팀번호로 묶어야 한다.
③ O: 팀번호로 그룹화하면 팀 1과 팀 2의 집계값을 각각 계산할 수 있다. 이때 팀 1의 평균은 14, 팀 2의 평균은 20이므로 `HAVING AVG(점수) >= 15`를 적용하면 팀 2만 남는다. 그 결과 팀번호 2, 합계점수 40, 최대점수 30이 정확히 만들어진다.
④ X: `HAVING MAX(점수) >= 20`은 팀 1의 최대점수 24와 팀 2의 최대점수 30이 모두 조건을 만족하므로 두 팀이 모두 출력된다. 틀린 표현은 최대점수 조건만으로 팀 2만 남는다고 보는 점이며, 올바르게는 평균점수 조건으로 팀 1을 걸러야 한다.

오답 포인트

이 문항은 정답 팀의 집계값만 맞추면 되는 문제가 아니라 다른 팀이 함께 출력되는지도 반드시 확인해야 한다는 점이 함정이다. 팀 2의 최대점수 30만 보고 ④를 고르면 팀 1의 최대점수 24도 함께 조건을 만족한다는 사실을 놓치게 된다.

다시 나오면 이렇게 판단

GROUP BY 문제가 나오면 먼저 묶는 기준이 무엇인지 정하고, 각 그룹의 SUM, MAX, AVG를 직접 계산한다. 그 다음 결과에 남아야 하는 그룹과 제외되어야 하는 그룹을 비교하여 HAVING 조건이 정확히 어느 그룹을 제거하는지 판단한다.

NoteBase

암기 포인트

GROUP BY는 그룹을 만들고 HAVING은 집계된 그룹을 거른다.

NoteBase

문 13. 트랜잭션 T_1, T_2 스케줄이 다음과 같을 때 발생하는 문제는?

시간 ↓

T_1	T_2
	sum = sum + 0
	read(A)
	sum = sum + A
read(X)	
X = X − 100	
write(X)	
	read(X)
	sum = sum + X
	read(Y)
	sum = sum + Y
read(Y)	
Y = Y + 100	
write(Y)	

① 오손 읽기(dirty read)

② 갱신 손실(lost update)

③ 팬텀 문제(phantom problem)

④ 부정확한 요약(incorrect summary)

정답 4번

출제 포인트

이 문제는 스케줄 표에서 합계를 계산하는 트랜잭션이 갱신 중인 여러 항목을 섞어 읽을 때 발생하는 부정확한 요약 현상을 구별하는 문제이다.

NoteBase

선지별 해설

1번 X: 오손 읽기는 다른 트랜잭션이 아직 확정하지 않은 값을 읽는 현상 자체를 가리킨다. 이 문항의 핵심은 합계 계산 결과가 갱신 전후 값을 섞어 읽어 틀어진다는 점이다. 틀린 표현은 '오손 읽기'이고, 올바르게는 '여러 항목의 값을 비일관적으로 읽어 생긴 부정확한 요약'이다.
2번 X: 갱신 손실은 두 트랜잭션이 같은 데이터 항목을 각각 갱신한 뒤 한쪽 결과가 다른 쪽에 덮여 사라질 때 발생한다. 여기서는 T2가 X나 Y를 다시 기록하지 않으므로 T1의 갱신이 소실되지 않는다. 틀린 표현은 '갱신 손실'이고, 올바르게는 '요약 트랜잭션이 갱신 중간 상태를 읽은 현상'이다.
3번 X: 팬텀 문제는 같은 검색 조건으로 다시 조회했을 때 다른 트랜잭션의 삽입이나 삭제 때문에 행 집합이 달라지는 현상이다. 이 문항에는 범위 검색이나 행의 추가·삭제가 없다. 틀린 표현은 '팬텀 문제'이고, 올바르게는 '합계 계산 중 일부 값만 갱신 후 상태를 읽은 문제'이다.
4번 O: 부정확한 요약은 합계, 평균, 개수 같은 집계 연산이 동시 갱신 때문에 일관되지 않은 값을 읽어 잘못된 결과를 내는 현상이다. T2는 A와 Y는 갱신 전 기준, X는 갱신 후 기준으로 읽으므로 전체 합이 실제 데이터베이스의 어떤 한 시점 상태와도 일치하지 않는다.

오답 포인트

T2가 T1이 기록한 X를 읽는 장면만 보고 오손 읽기로 바로 가면 틀리기 쉽다. 이 문항은 한 항목의 미확정 읽기 여부보다 합계 결과가 일관된 데이터 상태를 반영하는지 여부를 묻는 문제라는 점이 함정이다.

다시 나오면 이렇게 판단

동시성 이상 문제는 먼저 각 트랜잭션이 읽기만 하는지, 쓰기까지 하는지, 집계 연산인지, 범위 검색인지부터 구분한다. 합계 트랜잭션이 여러 항목을 읽는 사이 다른 트랜잭션이 값을 옮기거나 바꾸면 부정확한 요약을 우선 의심할 수 있다.

암기 포인트

집계 중간에 갱신 전 값과 갱신 후 값을 섞어 읽으면 부정확한 요약이다.

NoteBase

문 14. 다음의 요구사항을 반영하여 도서 대출 관리용 관계 데이터 모델을 생성할 때, 이에 대한 설명으로 옳지 않은 것은?

◯ 회원은 회원번호, 이름, 생년월일, 주소를 가지며, 회원번호를 기본키로 한다.
◯ 도서는 도서번호, 제목, 저자, 출판연도를 가지며, 도서번호를 기본키로 한다.
◯ 대출은 대출번호, 회원번호, 도서번호, 대출일을 가지며, 대출번호를 기본키로 한다.
◯ 반납은 반납번호, 대출번호, 반납일을 가지며, 반납번호를 기본키로 한다.

① '회원', '도서', '대출', '반납'이 주요 릴레이션이다.
② '도서'와 '반납' 릴레이션 간의 관계는 일대다(1 : n) 관계이다.
③ '대출'과 '반납' 릴레이션 간의 관계는 다대다(m : n) 관계이다.
④ '대출' 릴레이션의 회원번호는 '회원' 릴레이션의 회원번호를 참조하는 외래키이다.

정답 3번

출제 포인트

이 문제는 릴레이션의 기본키·외래키와 릴레이션 사이의 대응관계를 속성 연결 기준으로 판단할 수 있는지를 묻는다.

핵심 개념

기본키는 각 튜플을 유일하게 식별하는 속성이다. 외래키는 다른 릴레이션의 기본키를 참조하는 속성이다. 관계 차수는 어떤 릴레이션의 기본키가 다른 릴레이션에서 몇 번 반복될 수 있는지로 판단한다.

선지별 해설

1번 O: 회원, 도서, 대출, 반납은 각각 독립된 속성과 기본키를 가진 릴레이션으로 모델링되어 있으므로 주요 릴레이션으로 보는 것이 맞다.
2번 O: 반납은 대출번호를 통해 대출과 연결되고, 대출은 도서번호를 통해 도서와 연결된다. 한 권의 도서는 여러 번 대출·반납될 수 있으므로 도서와 반납의 대응은 1:n으로 볼 수 있다.
3번 X: 틀린 부분은 "다대다(m:n)"이다. 반납 릴레이션의 대출번호는 하나의 대출만 가리키므로 반납 한 건이 여러 대출에 대응하지 않는다. 올바르게는 대출과 반납은 1:1에 가까운 관계이며, 업무 의미까지 반영하면 대출 1건에 대해 반납은 0 또는 1건이다.
4번 O: 대출 릴레이션의 회원번호는 회원 릴레이션의 회원번호를 참조하여 어떤 회원이 대출했는지 연결하므로 외래키가 맞다.

NoteBase

오답 포인트

대출과 반납이 시간상 여러 번 발생한다는 사실만 보고 3번을 다대다로 착각하기 쉽다. 하지만 관계는 발생 횟수가 아니라 참조 속성의 연결 방식으로 판단한다. 반납 한 건은 하나의 대출번호만 가진다는 점이 핵심이다.

다시 나오면 이렇게 판단

먼저 각 릴레이션의 기본키를 찾고, 그다음 다른 릴레이션에 들어 있는 참조 속성을 따라 부모-자식 방향을 확인하면 된다. 부모 키 1개가 자식 쪽에 여러 번 나타나면 1:n이고, 자식 한 행이 여러 부모를 동시에 가리키지 못하면 m:n이 아니다.

암기 포인트

기본키는 자기 식별, 외래키는 연결, 관계판단은 속성 참조 방향으로 본다.

NoteBase

문 15. 표준 RAID(Redundant Array of Independent Disks) 1에 대한 설명으로 옳은 것은?

① 데이터를 디스크에 분산 저장하고 오류 검출 및 정정이 가능한 해밍코드를 사용한다.

② 미러링(mirroring) 기법을 활용하여 한 디스크 장애 시 미러 디스크에서 읽을 수 있다.

③ 데이터를 디스크에 교차 배치하여 읽기와 쓰기 성능이 향상되나, 장애 복구 기능은 제공되지 않는다.

④ 블록 단위로 데이터를 분할하여 여러 디스크에 분산 저장하고, 장애 시 패리티 비트를 통해 복구한다.

정답 2번

출제 포인트

이 문제는 RAID 1의 핵심 방식인 미러링과 RAID 0, RAID 2, RAID 5의 저장 방식 차이를 구별하는 문제이다.

핵심 개념

RAID 1은 동일한 데이터를 두 개 이상의 디스크에 그대로 복제하는 미러링 방식이다. 결정 차이축은 중복 저장 방식이며, 키워드는 미러링과 장애 허용이다. RAID 0은 데이터를 여러 디스크에 교차 배치하는 스트라이핑 방식으로 성능 향상이 목적이지만 중복이 없다. RAID 2는 비트 단위 분산과 해밍코드 기반 오류 정정을 사용한다. RAID 5는 블록 단위 분산 저장과 패리티를 사용해 디스크 장애를 복구한다.

선지별 해설

① X: 해밍코드를 사용해 오류 검출 및 정정을 하는 방식은 RAID 2의 설명이다. 틀린 부분은 "해밍코드"이다. RAID 1은 해밍코드를 쓰지 않고 동일 데이터를 복제하는 미러링 방식이다.
② O: RAID 1은 미러링을 사용하므로 한 디스크에 장애가 나도 복제된 다른 디스크에서 계속 읽을 수 있다. 장애 허용성을 확보하는 대표적 특징이다.
③ X: 데이터를 교차 배치해 성능을 높이지만 복구 기능이 없다는 설명은 RAID 0에 해당한다. 틀린 부분은 "교차 배치"와 "장애 복구 기능은 제공되지 않는다"이다. RAID 1은 교차 배치가 아니라 미러링을 사용하며, 한 디스크 장애 시에도 데이터를 읽을 수 있다.
④ X: 블록 단위 분산 저장과 패리티로 복구하는 방식은 RAID 5의 설명이다. 틀린 부분은 "패리티 비트"이다. RAID 1은 패리티로 복구하지 않고 같은 데이터를 복제해 장애에 대비한다.

오답 포인트

출제자는 RAID별 대표 키워드를 서로 바꾸어 놓았다. ①의 해밍코드, ③의 교차 배치, ④의 패리티를 RAID 1의 특징처럼 보면 틀리기 쉽다. RAID 1은 성능 최적화나 오류 정정 코드보다 미러링 자체가 핵심이라는 점이 함정 회피 포인트이다.

NoteBase

다시 나오면 이렇게 판단

선지에서 미러링, 복제, 동일 데이터 이중 저장이 나오면 RAID 1을 먼저 떠올리면 된다. 해밍코드가 나오면 RAID 2, 교차 배치만 있고 중복이 없으면 RAID 0, 블록 단위와 패리티가 나오면 RAID 5로 구분할 수 있다.

암기 포인트

RAID 1은 미러링, RAID 0은 스트라이핑, RAID 2는 해밍코드, RAID 5는 블록 분산과 패리티이다.

NoteBase

문 16. 분산 데이터베이스 시스템에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 지역 트랜잭션과 전역 트랜잭션으로 구분할 수 있다.
ㄴ. 분산 투명성에는 위치 투명성, 중복 투명성, 단편화 투명성이 포함된다.
ㄷ. 사용자는 원하는 데이터가 어느 지역에 저장되어 있는지를 명확하게 알 필요가 있다.
ㄹ. 한 릴레이션의 속성을 분할하여 두 개 이상의 서로 다른 장소에 저장하는 것은 수평적 단편화이다.

① ㄱ, ㄴ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄷ, ㄹ

정답 1번

출제 포인트

이 문제는 분산 데이터베이스의 트랜잭션 구분, 분산 투명성, 단편화 유형을 정확히 구별하는지를 묻는 문제이다.

핵심 개념

지역 트랜잭션은 한 지역 데이터베이스 안에서만 수행되는 트랜잭션이고, 전역 트랜잭션은 둘 이상의 지역 사이트에 걸쳐 수행되는 트랜잭션이다. 분산 투명성은 사용자가 데이터의 분산 저장 방식을 의식하지 않고 이용하게 하는 성질로 위치 투명성, 중복 투명성, 단편화 투명성 등을 포함한다. 수평적 단편화는 튜플을 나누는 방식이고, 수직적 단편화는 속성을 나누는 방식이다.

선지별 해설

① O: ㄱ과 ㄴ만 옳다. ㄱ은 분산 데이터베이스의 트랜잭션을 지역 트랜잭션과 전역 트랜잭션으로 구분한다는 설명이므로 맞다. ㄴ도 분산 투명성의 대표 요소로 위치 투명성, 중복 투명성, 단편화 투명성을 드는 설명이므로 맞다.
② X: 틀린 부분은 ㄹ을 옳다고 본 점이다. ㄹ의 '속성을 분할'은 수평적 단편화가 아니라 수직적 단편화이다. 올바르게는 '한 릴레이션의 속성을 분할하여 서로 다른 장소에 저장하는 것은 수직적 단편화이다'가 된다.
③ X: 틀린 부분은 ㄷ을 옳다고 본 점이다. 분산 데이터베이스는 위치 투명성을 제공하므로 사용자가 데이터가 어느 지역에 저장되어 있는지 명확히 알 필요는 없다. 올바르게는 '사용자는 원하는 데이터의 저장 지역을 몰라도 된다'가 된다.
④ X: ㄷ과 ㄹ을 모두 옳다고 본 점이 틀리다. ㄷ은 위치 투명성에 반하고, ㄹ은 속성 분할을 수평적 단편화로 잘못 설명했다. 올바르게는 ㄷ은 거짓이고 ㄹ도 거짓이다.

NoteBase

오답 포인트

ㄹ의 '속성'이라는 단어를 놓치면 수평적 단편화와 수직적 단편화를 뒤집기 쉽다. 또한 ㄷ은 분산 데이터베이스의 핵심 장점인 위치 투명성을 정반대로 서술했는데, '사용자가 알아야 한다'라는 표현 때문에 익숙하지 않으면 맞는 말처럼 보일 수 있다.

다시 나오면 이렇게 판단

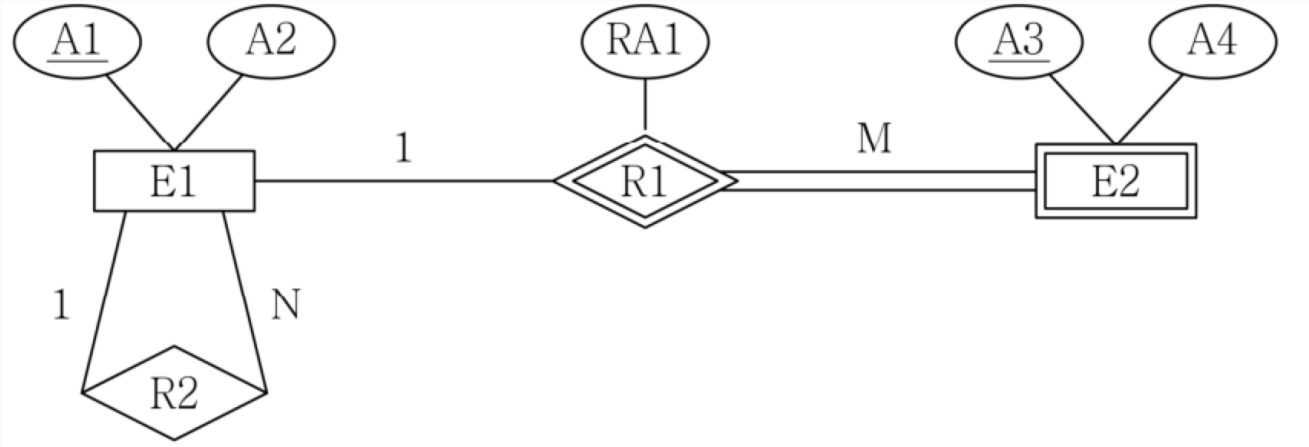
분산 데이터베이스 문제에서는 먼저 투명성이 사용자에게 무엇을 숨겨 주는지 본다. 위치를 숨기면 저장 위치를 알 필요가 없다. 다음으로 단편화는 '행을 나누면 수평, 열(속성)을 나누면 수직'이라는 기준으로 즉시 판별한다.

암기 포인트

분산 투명성은 위치를 숨기고, 속성 분할은 수직 단편화이다.

NoteBase

문 17. 다음 개체-관계 다이어그램(E-R diagram)을 관계형 데이터베이스 스키마로 변환하면? (단, 밑줄 속성은 기본키이고, RA2와 'E2' 릴레이션의 A1은 'E1' 릴레이션의 A1을 참조하는 외래키이다)



① E1(A1, A2, RA2)
E2(A3, A4, RA1)

② E1(A1, A2, RA1, RA2)
E2(A3, A4)

③ E1(A1, A2, RA1, RA2)
E2(A1, A3, A4)

④ E1(A1, A2, RA2)
E2(A1, A3, A4, RA1)

정답 4번

출제 포인트

이 문제는 약한 개체와 식별 관계, 그리고 1:N 자기참조 관계를 관계형 스키마로 변환하는 규칙을 묻는 문제이다.

NoteBase

선지별 해설

① X: E1(A1, A2, RA2)는 자기참조 1:N 관계를 N측 외래키 RA2로 흡수한 형태라서 맞다. 그러나 E2를 E2(A3, A4, RA1)로 두면 약한 개체가 소유자 키 A1을 받지 못한다. 틀린 표현은 "E2(A3, A4, RA1)"이다. 올바르게는 "E2(A1, A3, A4, RA1)"이며 기본키는 (A1, A3)이다.
② X: 식별 관계 R1의 속성 RA1은 약한 개체 E2를 식별하는 쪽에 함께 저장된다. 틀린 표현은 "E1(A1, A2, RA1, RA2)"와 "E2(A3, A4)"이다. 올바르게는 E1은 "E1(A1, A2, RA2)", E2는 "E2(A1, A3, A4, RA1)"이다.
③ X: E2에 A1을 넣은 점은 맞지만, R1의 속성 RA1을 E1에 두고 E2에서 뺀 점이 틀리다. 틀린 표현은 "E1(A1, A2, RA1, RA2)"와 "E2(A1, A3, A4)"이다. 올바르게는 RA1을 E2에 포함하여 "E2(A1, A3, A4, RA1)"로 두어야 한다.
④ O: E1의 RA2는 자기참조 1:N 관계 R2에서 N측이 갖는 자기 외래키이고, E2의 A1은 약한 개체가 소유자 E1을 참조하기 위한 외래키이다. 따라서 E1은 "E1(A1, A2, RA2)", E2는 "E2(A1, A3, A4, RA1)"가 되며 E2의 기본키는 (A1, A3)이다.

오답 포인트

1:N 관계라는 표기만 보고 RA1을 E1 쪽으로 옮기기 쉽지만, 이 문항의 핵심은 단순 1:N이 아니라 약한 개체를 위한 식별 관계라는 점이다. 약한 개체 E2는 반드시 소유자 키 A1을 포함해야 하고, 자기참조 1:N 관계 R2는 같은 E1 릴레이션 안에 자기 외래키 RA2를 추가한다.

다시 나오면 이렇게 판단

이중 사각형과 이중 마름모가 보이면 약한 개체와 식별 관계부터 확인한다. 약한 개체 릴레이션에는 소유자 기본키를 포함하고, 관계 속성이 있으면 그 식별 관계를 흡수하는 약한 개체 쪽에 함께 둔다. 1:N 관계는 N측에 외래키를 두며, 자기참조이면 같은 릴레이션 안에 자기 외래키가 추가된다.

암기 포인트

약한 개체는 소유자 키를 받아 (소유자 키 + 자신의 키)로 식별하고, 자기참조 1:N은 같은 릴레이션에 자기 외래키를 둔다.

NoteBase

문 18. 다음에서 설명하는 테이블에서 모든 회원의 이름과 추천인의 이름을 함께 출력하기 위한 SQL문은? (단, 추천인이 없는 경우 추천인의 이름은 NULL 값으로 출력한다)

◯ '회원' 테이블의 속성은 (아이디, 이름, 등급, 추천인아이디)이다.
◯ 밑줄 속성은 기본키이고 추천인아이디는 '회원' 테이블의 아이디를 참조하는 외래키이다.
◯ 추천인아이디를 제외한 나머지 속성은 NOT NULL로 정의된다.
◯ 추천인이 없는 회원인 경우 추천인아이디 속성은 NULL로 저장된다.

①
```
SELECT e.이름, s.이름 AS 추천인이름
FROM 회원 e LEFT OUTER JOIN 회원 s
   ON e.추천인아이디 = s.아이디;
```

②
```
SELECT e.이름, s.이름 AS 추천인이름
FROM 회원 e, 회원 s
WHERE e.추천인아이디 = s.아이디;
```

③
```
SELECT 이름, 추천인이름
FROM 회원 e
WHERE EXISTS
      (SELECT 이름 AS 추천인이름
       FROM 회원
       WHERE e.아이디 = 추천인아이디);
```

④
```
SELECT e.이름, s.이름 AS 추천인이름
FROM 회원 e, 회원 s
WHERE e.추천인아이디 = s.아이디
   AND e.추천인아이디 IS NOT NULL;
```

정답 1번

출제 포인트

이 문제는 자기조인에서 추천인이 없는 회원까지 포함하려면 외부조인을 사용해야 한다는 점을 묻는 문제이다.

핵심 개념

자기조인은 같은 테이블을 서로 다른 별칭으로 나누어 연결하는 조인이다.
LEFT OUTER JOIN은 왼쪽 테이블의 모든 행을 유지하고, 오른쪽에서 일치하는 행이 없으면 오른쪽 열 값을 NULL로 채운다.
INNER JOIN은 조인 조건을 만족하는 행만 남기므로 추천인아이디가 NULL인 회원은 결과에서 제외된다.

NoteBase

선지별 해설

① O: 회원 e를 기준으로 회원 s를 e.추천인아이디 = s.아이디로 LEFT OUTER JOIN 하므로 모든 회원이 남고, 추천인이 없으면 s.이름이 NULL로 출력된다.
② X: 틀린 부분은 암시적 내부조인으로만 연결한 점이다. WHERE e.추천인아이디 = s.아이디는 일치하는 행만 남기므로 추천인이 없는 회원이 빠진다. 올바르게는 모든 회원을 남기기 위해 LEFT OUTER JOIN을 사용해야 한다.
③ X: 틀린 부분은 EXISTS로 추천인 존재 여부를 검사하면서 추천인이름까지 함께 출력할 수 있다고 본 점이다. EXISTS는 행의 존재만 판단하는 조건식이고, 바깥 SELECT의 추천인이름도 이 형태로는 만들어지지 않는다. 올바르게는 추천인 회원 행의 이름을 가져오도록 자기조인을 해야 한다.
④ X: 틀린 부분은 내부조인 뒤에 e.추천인아이디 IS NOT NULL을 추가한 점이다. 이 조건은 추천인이 없는 회원을 더 확실히 제외할 뿐이며 NULL 이름으로 남기지 못한다. 올바르게는 내부조인이 아니라 LEFT OUTER JOIN으로 연결해야 한다.

오답 포인트

발문의 핵심은 '모든 회원'과 '추천인이 없는 경우 추천인이름은 NULL'이다. 이 문구를 놓치면 같은 테이블을 연결한다는 점만 보고 ②나 ④ 같은 내부조인을 고르기 쉽다. ③도 추천인 존재 여부를 찾는 구문처럼 보여 헷갈리지만, 존재 여부 확인과 이름 출력은 다른 문제이다.

다시 나오면 이렇게 판단

기준이 되는 행을 모두 보여 주어야 하고 대응되는 상대 행이 없어도 NULL로 남겨야 하면 기준 테이블을 왼쪽에 둔 LEFT OUTER JOIN을 선택한다. 일치하는 행만 필요할 때만 INNER JOIN을 사용한다.

암기 포인트

모든 회원 유지 + 상대 없으면 NULL이면 회원 LEFT JOIN 추천인이다.

NoteBase

문 19. ‘R1’과 ‘R2’ 릴레이션이 다음과 같을 때 관계 대수 연산 적용에 대한 설명으로 옳지 않은 것은? (단, ‘R1’의 B와 ‘R2’의 B의 도메인은 동일하다)

R1	
A	B
a1	b1
a2	b2
a3	b3

R2		
B	C	D
b1	c1	d1
b3	c2	d3
b4	c1	d5

① $R1 \cup R2$ 연산은 수행될 수 없다.

② $R2 \div R1$ 연산 결과의 차수는 2이고 카디널리티는 2이다.

③ $\sigma_{R1.B='b2'}(R1 \times \Pi_{B,D}(R2))$ 연산 결과의 카디널리티는 3이다.

④ $\sigma_{A='a1'}(R1 \times R2)$와 $\sigma_{A='a1'}(R1 \bowtie_{R1.B=R2.B} R2)$ 연산 결과의 차수는 동일하다.

정답 2번

출제 포인트

이 문제는 릴레이션의 차수와 카디널리티, 그리고 합집합·카티션곱·조인·나눗셈의 적용 조건을 표에서 확인하는 문제이다.

자료 읽는 순서

1. R1의 속성은 A,B이고 튜플 수는 3이며, R2의 속성은 B,C,D이고 튜플 수는 3임을 확인한다.
2. 각 관계대수 연산이 먼저 정의되는지 판단하고, 정의되면 선택과 투영이 차수와 카디널리티에 어떤 영향을 주는지 계산한다.
3. 세타 조인은 공통 속성을 자동으로 합치지 않고, 자연 조인만 중복 속성을 합친다는 점을 보기와 연결한다.

NoteBase

선지별 해설

① O: 합집합은 두 릴레이션의 차수가 같고 대응 속성의 도메인이 같아야 한다. R1은 2차, R2는 3차이므로 R1 ∪ R2는 수행될 수 없다.
② X: 나눗셈은 제수 릴레이션의 속성 집합이 피제수 릴레이션의 속성 집합에 포함될 때 정의된다. R1의 속성은 A,B이고 R2의 속성은 B,C,D이므로 R1 전체가 R2에 포함되지 않아 R2 ÷ R1 자체가 성립하지 않는다. 틀린 표현은 '결과의 차수는 2이고 카디널리티는 2이다'이고, 올바르게는 'R2 ÷ R1 연산은 정의되지 않는다'이다.
③ O: $\Pi_{B,D}(R2)$는 $(b1, d1)$, $(b3, d3)$, $(b4, d5)$로 3개 튜플을 갖는다. $\sigma_{R1.B='b2'}(R1)$은 $(a2, b2)$ 한 개이므로 카티션곱의 카디널리티는 $1 \times 3 = 3$이다.
④ O: $\sigma_{A='a1'}(R1 \times R2)$는 선택만 한 것이므로 차수는 $2 + 3 = 5$이다. $\sigma_{A='a1'}(R1 \bowtie_{R1.B=R2.B} R2)$도 세타 조인이어서 두 B 속성이 모두 남아 차수는 5이다. 따라서 두 연산 결과의 차수는 동일하다.

오답 포인트

②에서 공통 속성 B가 있다는 이유만으로 나눗셈이 가능하다고 서두르면 틀리기 쉽다. ④에서도 조인 기호를 보고 자연 조인처럼 중복 B가 제거된다고 착각하기 쉬우나, 보기의 연산은 조건부 세타 조인이므로 차수가 줄지 않는다.

다시 나오면 이렇게 판단

관계대수 선지는 먼저 연산 가능 여부를 본다. 그다음 선택은 차수는 그대로 두고 튜플 수만 줄이며, 투영은 속성 수를 줄이고, 카티션곱은 차수를 더하고 카디널리티를 곱하며, 세타 조인은 자연 조인과 달리 중복 속성을 자동 제거하지 않는다고 판단한다.

암기 포인트

합집합은 같은 차수, 나눗셈은 제수 속성 포함, 세타 조인은 중복 속성 유지가 핵심이다.

NoteBase

문 20. 다음 <정보>를 이용하여 아래에 주어진 <관계 대수>를 세미조인(semijoin) 방식으로 처리하고자 할 때, 가장 먼저 수행되는 연산은? (단, 속성 이름의 괄호 안 숫자는 바이트 크기를 나타낸다)

<정 보>
◯ 'EMP'와 'DEPT' 테이블이 분산 저장되어 있다.
◯ 사이트 1에 'EMP' 테이블이 저장되며, SSN(10), NAME(30), DNO(5), ADDR(55)의 속성값으로 구성된 100 바이트 크기의 투플 10,000개를 포함한다.
◯ 사이트 2에 'DEPT' 테이블이 저장되며, DNUM(5), DNAME(25), MGRSSN(10), DATE(10)의 속성값으로 구성된 50 바이트 크기의 투플 100개를 포함한다.
<관계 대수>
$\Pi_{NAME,\,DNAME}(EMP \bowtie_{DNO\,=\,DNUM} DEPT)$

① $\Pi_{DNO}(EMP)$
② $\Pi_{DNUM}(DEPT)$
③ $\Pi_{DNO,\,NAME}(EMP)$
④ $\Pi_{DNUM,\,DNAME}(DEPT)$

정답 2번

출제 포인트

이 문제는 분산 데이터베이스에서 세미조인으로 조인을 처리할 때 어떤 투영 연산을 가장 먼저 수행해야 전송량을 최소화하는지를 묻는 문제이다.

올바른 순서

1. DEPT에서 $\Pi_{DNUM}(DEPT)$를 수행한다. — 작은 릴레이션에서 조인 속성만 뽑는다.
2. 그 결과를 EMP가 있는 사이트로 보낸다. — EMP에서 매칭되는 부서번호만 남길 기준 집합이 된다.
3. EMP를 줄인 뒤 필요한 속성만 정리한다. — 큰 릴레이션 전체를 보내지 않고 조인 대상만 축소한다.
4. 축소된 결과와 DEPT를 결합한 뒤 최종적으로 NAME, DNAME만 투영한다. — 최종 결과 속성은 맨 마지막에 정리된다.

왜 이 순서: 세미조인은 상대적으로 작은 릴레이션의 조인 속성을 먼저 보내서 큰 릴레이션을 줄이는 방식이므로, 뒤 단계의 축소 작업은 앞 단계의 $\Pi_{DNUM}(DEPT)$ 결과가 있어야 가능하다.

NoteBase

선지별 해설

① X. $\Pi_{DNO}(EMP)$를 먼저 하면 큰 릴레이션 EMP에서 10,000개의 튜플에 대해 조인 속성을 뽑아 보내는 흐름이 된다. 세미조인의 시작점은 작은 쪽의 조인 속성만 먼저 보내는 것이다. 틀린 표현은 'EMP를 먼저 투영한다'는 점이다. 올바르게는 DEPT에서 $\Pi_{DNUM}(DEPT)$를 먼저 수행한다.

② O. DEPT는 100개 튜플, 조인 속성 DNUM의 크기는 5바이트이므로 먼저 $\Pi_{DNUM}(DEPT)$를 만들면 전송 대상이 매우 작아진다. 이 결과를 EMP 쪽으로 보내야 큰 릴레이션 EMP를 세미조인으로 줄일 수 있다.

③ X. $\Pi_{DNO, NAME}(EMP)$는 NAME까지 포함하므로 초기 축소 단계에 불필요한 속성을 함께 다루게 된다. NAME은 최종 출력에 필요하지만, 가장 먼저 필요한 것은 EMP를 거를 기준이 되는 DEPT의 조인 속성 집합이다. 틀린 표현은 'NAME까지 포함한 EMP 투영을 먼저 한다'는 점이다. 올바르게는 먼저 $\Pi_{DNUM}(DEPT)$를 수행하고, EMP 축소 후에 필요한 속성을 남긴다.

④ X. $\Pi_{DNUM, DNAME}(DEPT)$는 DNAME이 최종 출력에는 필요하지만 첫 단계의 세미조인 필터링에는 필요하지 않다. 처음에는 조인 성립 여부만 판단하면 되므로 DNUM만 있으면 충분하다. 틀린 표현은 'DNAME까지 함께 투영한다'는 점이다. 올바르게는 처음에는 $\Pi_{DNUM}(DEPT)$만 수행하고 DNAME은 최종 조인 결과를 만들 때 사용한다.

오답 포인트

최종 결과에 필요한 속성을 먼저 뽑아야 한다고 생각하면 ③이나 ④로 흔들리기 쉽다. 세미조인의 목적은 결과 속성 정리가 아니라 전송량 감소이므로, 첫 단계에서는 '최종 출력 속성'이 아니라 '작은 릴레이션의 조인 속성'만 본다.

다시 나오면 이렇게 판단

세미조인 절차형 문제는 먼저 어떤 릴레이션이 더 작은지, 그다음 조인 성립에 꼭 필요한 속성이 무엇인지 순서대로 본다. 작은 릴레이션의 조인 속성만 먼저 투영한 뒤 그것으로 큰 릴레이션을 줄이는 흐름이면 정답 방향이 맞다.

암기 포인트

세미조인의 시작은 작은 쪽 조인키 투영이다. 결과 속성보다 조인키가 먼저이다.

NoteBase

문 21. 인덱스에 대한 설명으로 옳은 것은?

① SQL문에서 조인 연산에 자주 사용되는 속성은 인덱스로 적합하다.

② 클러스터링(clustering) 인덱스는 검색 효율을 위해 테이블당 2개 이상 활용된다.

③ 밀집(dense) 인덱스는 데이터 파일의 레코드 그룹에 대해 생성되는 하나의 엔트리이다.

④ 인덱스를 구성하기 위한 자료구조로는 이진 탐색 트리(binary search tree)가 가장 많이 활용된다.

정답 1번

출제 포인트

이 문제는 인덱스의 용도와 클러스터링 인덱스, 밀집 인덱스, 인덱스 자료구조의 차이를 정확히 구별할 수 있는지를 묻는다.

핵심 개념

인덱스는 원하는 레코드를 빠르게 찾기 위한 보조 접근 경로이다. 클러스터링 인덱스는 데이터가 인덱스 순서에 맞추어 물리적으로 정렬된 인덱스이므로 한 테이블에는 보통 하나만 둘 수 있다. 밀집 인덱스는 모든 탐색키 값 또는 레코드에 대응하는 엔트리를 두는 인덱스이고, 성긴 인덱스는 블록이나 레코드 그룹 대표값만 둔다. 실제 DB 인덱스 구조로는 이진 탐색 트리보다 B-트리나 B+트리가 더 널리 쓰인다.

선지별 해설

1번: O. 조인 조건이나 검색 조건에 자주 등장하는 속성은 탐색 횟수를 줄이는 데 직접 도움이 되므로 인덱스 후보로 적절하다. 특히 조인에 반복적으로 사용되는 속성은 인덱스를 통해 매칭 대상을 빠르게 찾을 수 있다.
2번: X. 틀린 부분은 "테이블당 2개 이상 활용된다"이다. 클러스터링 인덱스는 데이터의 물리적 저장 순서를 결정하므로 한 테이블에는 일반적으로 하나만 둘 수 있다. 올바르게는 "클러스터링 인덱스는 테이블당 보통 1개만 둘 수 있다"이다.
3번: X. 틀린 부분은 "레코드 그룹에 대해 생성되는 하나의 엔트리"이다. 그것은 성긴 인덱스의 설명에 가깝다. 밀집 인덱스는 각 탐색키 값 또는 각 레코드에 대응하는 엔트리를 갖는다. 올바르게는 "밀집 인덱스는 모든 탐색키 값 또는 레코드에 대해 엔트리를 둔다"이다.
4번: X. 틀린 부분은 "이진 탐색 트리(binary search tree)가 가장 많이 활용된다"이다. 실제 데이터베이스 인덱스는 균형 유지와 디스크 접근 효율 때문에 B-트리나 B+트리가 주로 사용된다. 올바르게는 "인덱스 자료구조로는 B-트리나 B+트리가 대표적으로 활용된다"이다.

오답 포인트

3번의 "레코드 그룹당 하나의 엔트리"를 밀집 인덱스로 착각하거나, 4번에서 일반 자료구조인 이진 탐색 트리를 데이터베이스 인덱스의 대표 구조로 그대로 연결하면 틀리기 쉽다. 2번도 클러스터링이라는 이름 때문에 여러 개를 둘 수 있다고 오해하기 쉽지만 물리적 정렬 기준은 하나뿐이다.

NoteBase

다시 나오면 이렇게 판단

인덱스 문제는 먼저 용도를 확인하고, 다음으로 물리적 정렬을 바꾸는지 여부로 클러스터링 인덱스를 판단한다. 이어서 엔트리가 모든 값에 붙는지, 대표값만 붙는지로 밀집 인덱스와 성긴 인덱스를 구별하면 된다. 자료구조를 묻는 선지는 BST인지 B-트리 계열인지부터 구분하면 된다.

암기 포인트

조인 속성은 인덱스 후보, 클러스터링 인덱스는 1개, 밀집 인덱스는 전부에 엔트리, DB 인덱스 구조는 B+트리이다.

NoteBase

문 22. 데이터베이스 관리시스템의 로그 레코드 버퍼링(log record buffering)에 대한 설명으로 옳지 않은 것은?

① 로그 레코드가 상당 기간 메인 메모리에 있으면 시스템 장애 시 해당 로그 레코드가 손실될 수 있다.

② 트랜잭션의 원자성을 보장하기 위한 회복 기법으로 로그 우선 기록 규약(write-ahead log protocol)을 지원해야 한다.

③ 시스템 장애 시 복구를 위해 변경 사항을 즉시 디스크에 기록하며, 이는 시스템의 성능을 최적화하는 데 중요한 역할을 한다.

④ 트랜잭션이 수행되는 동안 발생하는 변경 사항을 일시적으로 메모리 버퍼에 저장하여, 디스크 출력 횟수를 줄이고 시스템 성능을 향상시킨다.

정답 3번

출제 포인트

이 문제는 로그 레코드 버퍼링의 목적과 로그 우선 기록 규약의 의미를 구분할 수 있는지를 묻는다.

핵심 개념

로그 레코드 버퍼링은 로그 레코드를 메모리 버퍼에 모아 두었다가 적절한 시점에 디스크로 출력하여 I/O를 줄이는 기법이다. 로그 우선 기록 규약은 관련 데이터 페이지가 디스크에 기록되기 전에 해당 로그가 먼저 안정 저장장치에 기록되어야 한다는 규칙이다.

선지별 해설

1번 O: 로그 레코드가 오랫동안 메인 메모리에만 머물면 시스템 장애 시 아직 디스크로 flush되지 않은 로그가 사라질 수 있다. 로그 버퍼링의 대표적 위험을 설명한 문장으로 타당하다.
2번 O: 트랜잭션의 원자성을 보장하려면 장애 후 undo/redo가 가능해야 하므로 WAL을 따라야 한다. 즉 데이터보다 로그가 먼저 기록되어야 복구가 가능하다.
3번 X: 틀린 표현은 "변경 사항을 즉시 디스크에 기록"과 "성능을 최적화"이다. 로그 레코드 버퍼링은 모든 변경 로그를 즉시 디스크에 쓰는 방식이 아니라 메모리 버퍼에 모아 두었다가 한꺼번에 기록하여 성능을 높이는 방식이다. 올바르게는 "변경 사항에 대한 로그 레코드를 버퍼에 저장했다가 WAL을 지키는 범위에서 디스크에 기록하며, 이로써 복구 가능성과 성능을 함께 확보한다"이다.
4번 O: 트랜잭션 수행 중 발생한 변경 내용을 담은 로그 레코드를 일시적으로 메모리 버퍼에 저장하면 디스크 출력 횟수를 줄일 수 있다. 로그 버퍼링의 목적이 바로 이 성능 향상에 있다.

오답 포인트

복구를 위해서는 모든 로그를 즉시 디스크에 써야 한다고 단순화하면 3번에 속게 된다. WAL은 즉시성 자체를 강제하는 규칙이 아니라 데이터 기록보다 로그 기록의 선행 순서를 요구하는 규칙이다.

NoteBase

다시 나오면 이렇게 판단

로그 버퍼링 문제가 나오면 먼저 저장 대상이 로그인지 데이터인지 구분하고, 다음으로 "즉시 디스크 기록"인지 "버퍼에 모아 기록"인지 확인한다. 성능 향상과 관련된 설명은 대체로 버퍼링 쪽이고, 복구 안전성과 관련된 설명은 WAL의 선행 기록 규칙과 연결된다.

암기 포인트

로그 버퍼링은 모아서 쓰는 것이고, WAL은 데이터보다 먼저 쓰는 규칙이다.

NoteBase

문 23. 고객(고객번호, 고객이름, 나이, 등급, 직업, 적립금) 테이블에 대하여 다음과 같이 4개의 뷰를 생성하였을 때, 데이터 삽입이 가능한 뷰는? (단, 밑줄 속성은 기본키이며, 기본키 이외에 NOT NULL 속성은 없다)

```
CREATE VIEW 고객1(번호, 이름, 등급)
AS SELECT DISTINCT 고객번호, 고객이름, 등급
    FROM 고객
    WHERE 적립금 > 20000;

CREATE VIEW 고객2(고객이름, 등급, 적립금)
AS SELECT 고객이름, 등급, 적립금
    FROM 고객
    WHERE 직업 = '학생';

CREATE VIEW 고객3(등급, 평균나이)
AS SELECT 등급, AVG(나이) AS 평균나이
    FROM 고객
    GROUP BY 등급;

CREATE VIEW 고객4
AS SELECT 고객번호, 고객이름, 직업, 적립금
    FROM 고객
    WHERE 나이 >= 30 AND 등급 = 'GOLD';
```

① 고객1

② 고객2

③ 고객3

④ 고객4

정답 4번

출제 포인트

이 문제는 삽입 가능한 뷰의 조건과 삽입이 불가능하게 만드는 요소인 DISTINCT, GROUP BY, 집계함수를 구별할 수 있는지를 묻는다.

핵심 개념

삽입 가능한 뷰는 한 개의 기본 테이블에서 유도되고, 행의 대응이 1대1로 유지되며, 기본키처럼 반드시 넣어야 할 값이 뷰를 통해 입력 가능해야 한다. DISTINCT, GROUP BY, AVG 같은 중복 제거나 집계가 들어가면 원본 행이 직접 결정되지 않아 삽입할 수 없다.

NoteBase

선지별 해설

1번(X): 고객1은 SELECT DISTINCT를 사용하므로 뷰의 한 행이 원본 테이블의 어느 한 행에 정확히 대응되지 않는다. 틀린 표현은 DISTINCT를 둔 채 삽입 가능하다고 보는 점이다. 올바르게는 DISTINCT가 포함된 뷰는 삽입이 불가능하다.
2번(X): 고객2는 고객번호가 빠져 있다. 문제에서 밑줄 속성인 고객번호는 기본키이므로 삽입 시 반드시 값이 필요하다. 틀린 표현은 기본키 없이도 삽입 가능하다고 보는 점이다. 올바르게는 기본키 고객번호를 입력할 수 없는 이 뷰는 삽입이 불가능하다.
3번(X): 고객3은 GROUP BY와 AVG(나이)를 사용한 집계 뷰이다. 집계 결과 한 행은 여러 원본 행을 묶어 만든 값이므로 새 행을 어느 원본 튜플로 넣을지 결정할 수 없다. 틀린 표현은 집계 뷰를 삽입 가능하다고 보는 점이다. 올바르게는 GROUP BY나 집계함수가 있는 뷰는 삽입이 불가능하다.
4번(O): 고객4는 한 개의 기본 테이블에서 단순 SELECT로 만든 뷰이고 DISTINCT, GROUP BY, 집계함수가 없다. 또한 기본키인 고객번호를 포함하므로 기본 테이블에 새 행을 대응시켜 삽입할 수 있다. WHERE 조건은 행의 범위를 제한할 뿐, 이런 단순 뷰 자체를 삽입 불가능하게 만들지는 않는다.

오답 포인트

WHERE 조건이 붙으면 무조건 삽입이 안 된다고 오해하기 쉽다. 이 문제의 핵심 함정은 조건절 자체가 아니라 DISTINCT, GROUP BY, AVG 같은 비갱신 요소와 기본키 포함 여부이다.

다시 나오면 이렇게 판단

먼저 한 개의 기본 테이블에서 단순 선택한 뷰인지 본다. 그다음 DISTINCT, GROUP BY, 집계함수 유무를 확인하고, 마지막으로 기본키처럼 반드시 필요한 값을 뷰에서 입력할 수 있는지 확인하면 된다.

암기 포인트

단순 SELECT + 기본키 포함은 삽입 가능, DISTINCT·GROUP BY·집계함수는 삽입 불가이다.

NoteBase

문 24. 다음 '고객' 테이블과 '주문' 테이블에서 주어진 SQL문과 동일한 결과를 출력하는 SQL문은? (단, 밑줄 속성은 기본키이며, '주문' 테이블의 고객번호는 '고객' 테이블의 번호를 참조하는 외래키이다)

고객

번호	이름	성별
1001	김홍도	M
1002	허균	M
1003	유관순	F

주문

고객번호	제품번호	개수
1001	AA	1
1003	AA	1
1001	BA	2

```
SELECT 이름, 제품번호, 개수
FROM 고객, 주문
WHERE 번호 = 고객번호
ORDER BY 이름, 제품번호;
```

①
```
SELECT 이름, 제품번호, 개수
FROM 고객 OUTER JOIN 주문
   ON 번호 = 고객번호
ORDER BY 이름, 제품번호;
```

②
```
SELECT 이름, 제품번호, 개수
FROM 고객 RIGHT OUTER JOIN 주문
   ON 번호 = 고객번호
ORDER BY 이름, 제품번호;
```

③
```
SELECT 이름, 제품번호, 개수
FROM 고객 LEFT OUTER JOIN 주문
   ON 번호 = 고객번호
ORDER BY 이름, 제품번호;
```

④
```
SELECT 이름, 제품번호, 개수
FROM 고객, 주문
WHERE 번호 IN (SELECT 고객번호 FROM 주문)
ORDER BY 이름, 제품번호;
```

NoteBase

정답 2번

출제 포인트

이 문제는 고객과 주문의 조인 결과를 표 데이터에 맞춰 비교하고, 내부 조인과 외부 조인의 결과 집합 차이를 구분하는 문제이다.

자료 읽는 순서

1. 고객 테이블에서 고객번호 1001, 1002, 1003과 이름을 확인한다.
2. 주문 테이블에서 주문이 있는 고객번호가 1001, 1003뿐임을 확인한다.
3. 제시 SQL은 WHERE 번호 = 고객번호 조건이 있는 내부 조인이라는 점을 확인한다.
4. 따라서 결과는 주문이 있는 고객만 나오고, 주문이 없는 1002 허균은 제외된다고 판단한다.

선지별 해설

① X: OUTER JOIN은 일치하지 않는 행도 남기는 외부 조인이다. 이 데이터에서는 주문이 없는 고객 1002 허균 행이 NULL 값과 함께 추가될 수 있으므로 제시 SQL의 내부 조인 결과와 같지 않다. 틀린 표현은 내부 조인과 동일하다고 보는 점이며, 올바르게는 일치 행만 원하면 INNER JOIN 또는 동등한 조인 조건을 사용해야 한다.
② O: 고객 RIGHT OUTER JOIN 주문 ON 번호 = 고객번호는 주문 테이블의 모든 행을 기준으로 남긴다. 문제에서 주문.고객번호는 고객.번호를 참조하는 외래키이므로 주문의 3개 행은 모두 고객과 매칭된다. 따라서 결과는 김홍도-AA-1, 김홍도-BA-2, 유관순-AA-1의 3행으로 제시 SQL과 동일하다.
③ X: 고객 LEFT OUTER JOIN 주문은 고객 테이블의 모든 행을 남긴다. 따라서 주문이 없는 허균도 제품번호와 개수가 NULL인 행으로 포함되어 결과가 4행이 된다. 틀린 표현은 내부 조인과 같은 결과가 나온다고 보는 점이며, 올바르게는 LEFT OUTER JOIN은 왼쪽 테이블의 미매칭 행까지 포함한다.
④ X: FROM 고객, 주문에 WHERE 번호 IN (SELECT 고객번호 FROM 주문)만 쓰면 고객과 주문 사이의 연결 조건인 번호 = 고객번호가 없다. 그래서 주문 이력이 있는 고객 1001과 1003이 주문 테이블의 모든 행 3개와 각각 결합되어 총 6행의 카티션 곱 일부가 만들어진다. 틀린 표현은 서브쿼리만으로 조인이 대체된다고 보는 점이며, 올바르게는 동일 결과를 내려면 번호 = 고객번호 조건이 반드시 필요하다.

오답 포인트

외부 조인이라는 이름만 보고 ①과 ③을 같은 내부 조인처럼 처리하거나, ④에서 서브쿼리가 있으니 자동으로 고객과 주문이 연결된다고 착각하기 쉽다. 이 문항의 함정은 어떤 테이블의 모든 행을 보존하는지와 조인 조건 자체가 존재하는지를 분리해서 보지 않는 데 있다.

다시 나오면 이렇게 판단

먼저 기준 SQL이 내부 조인인지 외부 조인인지 확인하고, 다음으로 어느 테이블의 모든 행을 남기는지 본다. 그 뒤 조인 조건이 실제로 두 테이블을 연결하는지 확인하면 동일 결과 여부를 빠르게 판정할 수 있다.

NoteBase

암기 포인트

INNER JOIN은 일치 행만, LEFT/RIGHT OUTER JOIN은 한쪽 전체 행까지, 조인 조건이 없으면 카티션 곱이 기본이다.

NoteBase

문 25. 다음 '부품상태' 테이블에서 도출할 수 있는 함수 종속을 화살표(→)로 표현할 때 옳은 것만을 표현한 것은? (단, 밑줄 속성은 기본키이며, 아래 투플을 기준으로 판단한다)

부품상태

상태번호	상태	재고위치	부품번호	수량
S1	20	서울	P1	300
S1	20	서울	P2	200
S1	20	서울	P3	400
S1	20	서울	P4	200
S1	20	서울	P5	100
S1	20	서울	P6	100
S2	10	부산	P1	300
S2	10	부산	P2	400
S3	10	부산	P2	200
S4	20	서울	P2	200
S4	20	서울	P4	300
S4	20	서울	P5	400

① {상태번호} → {재고위치}
{부품번호, 수량} → {상태}

② {부품번호} → {상태번호}
{상태번호, 부품번호} → {상태}

③ {부품번호} → {수량}
{상태번호, 상태} → {부품번호}

④ {상태번호} → {상태, 재고위치}
{상태번호, 부품번호} → {수량}

정답 4번

출제 포인트

이 문제는 주어진 튜플 집합만을 기준으로 속성 집합 사이의 함수 종속이 성립하는지를 판별하는 문제이다.

자료 읽는 순서

1. 밑줄 친 속성이 기본키이므로 {상태번호, 부품번호}가 각 튜플을 식별한다는 점을 확인한다.
2. 각 보기의 좌변 값이 같은 행들을 묶어 우변 값이 하나로만 나오는지 확인한다.
3. 같은 좌변에서 우변이 둘 이상 나오면 함수 종속이 깨진다고 판단한다.

NoteBase

핵심 근거

{상태번호, 부품번호}는 기본키이므로 현재 표에서는 수량을 포함한 다른 속성을 결정한다.
상태번호가 S1이면 항상 상태 20, 재고위치 서울이고, S2는 10과 부산, S3는 10과 부산, S4는 20과 서울로 고정되어 {상태번호} -> {상태, 재고위치}가 성립한다.
반면 부품번호 P2는 S1, S2, S3, S4에 모두 나타나고 수량도 200과 400으로 달라져 부품번호 단독 종속은 성립하지 않는다.

선지별 해설

① X: {상태번호} -> {재고위치}는 각 상태번호마다 재고위치가 하나로만 나타나므로 성립한다. 그러나 {부품번호, 수량} -> {상태}는 (P1, 300)이 S1에서는 상태 20, S2에서는 상태 10으로 나타나 성립하지 않는다. 틀린 표현은 '{부품번호, 수량} -> {상태}'이며, 올바르게는 '{부품번호, 수량}만으로는 {상태}를 결정할 수 없다'이다.
② X: {부품번호} -> {상태번호}는 P2가 S1, S2, S3, S4에 모두 나타나므로 성립하지 않는다. {상태번호, 부품번호} -> {상태}는 기본키가 상태를 결정하므로 성립한다. 틀린 표현은 '{부품번호} -> {상태번호}'이며, 올바르게는 '{부품번호}만으로는 {상태번호}를 결정할 수 없다'이다.
③ X: {부품번호} -> {수량}는 P2의 수량이 200과 400으로 달라 성립하지 않는다. {상태번호, 상태} -> {부품번호}는 (S1, 20)에서 P1, P2, P3, P4, P5, P6이 모두 가능하므로 성립하지 않는다. 틀린 표현은 '{부품번호} -> {수량}'과 '{상태번호, 상태} -> {부품번호}'이며, 올바르게는 '{부품번호}만으로는 {수량}을 결정할 수 없고, {상태번호, 상태}만으로는 {부품번호}를 결정할 수 없다'이다.
④ O: {상태번호} -> {상태, 재고위치}는 같은 상태번호에서 상태와 재고위치가 각각 하나로만 정해지므로 성립한다. {상태번호, 부품번호} -> {수량}은 밑줄 친 기본키가 각 튜플을 유일하게 식별하므로 성립한다.

오답 포인트

부품번호가 같으면 상태번호나 수량도 같을 것이라고 섣불리 묶으면 틀리기 쉽다. 특히 P2처럼 여러 상태번호에 반복 등장하는 값을 하나만 보고 일반화하면 함수 종속을 잘못 판단하게 된다.

다시 나오면 이렇게 판단

함수 종속은 같은 좌변값을 가진 행들을 모아 우변이 하나로만 정해지는지 보는 방식으로 판단한다. 기본키를 포함한 좌변은 우선 성립 후보로 보고, 부품번호나 상태처럼 반복되는 값은 실제 중복 행을 대조해 깨지는 사례가 있는지 확인한다.

암기 포인트

같은 좌변이면 우변은 하나로 정해지고, 기본키는 나머지 속성을 결정한다.

NoteBase