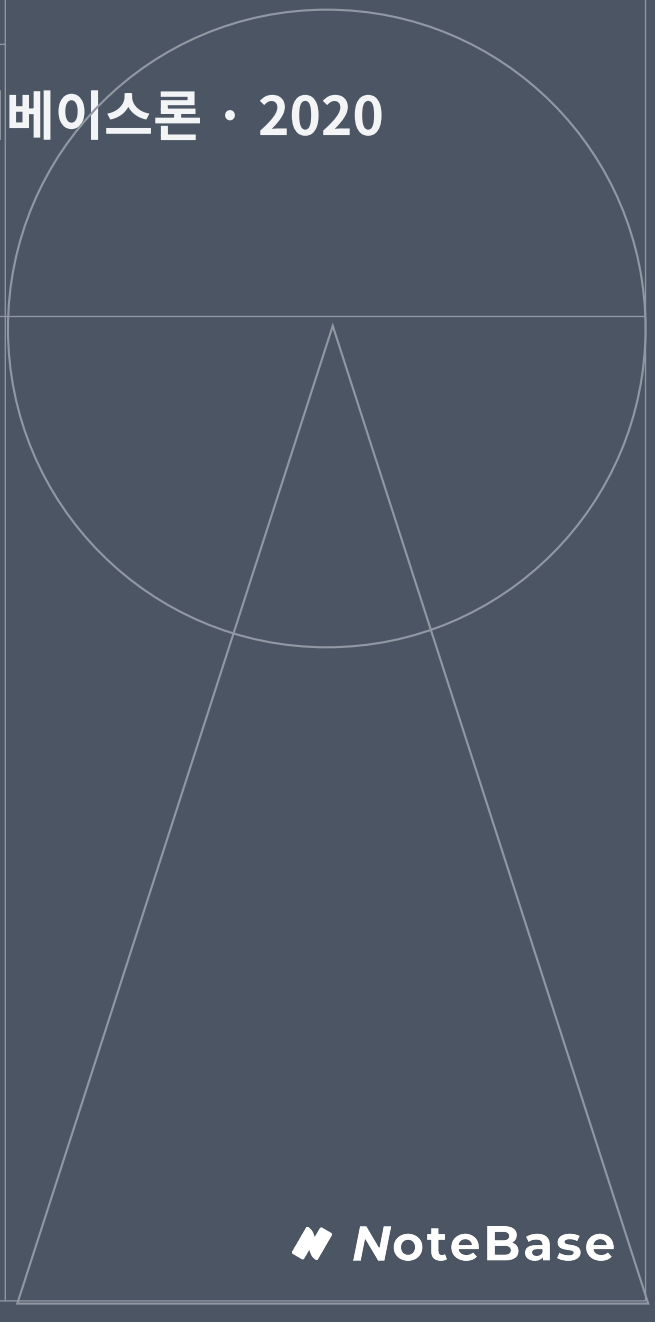
기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2020



NoteBase

기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2020

공무원/국가직/7급 · 데이터베이스론 · 2020

문 1. 관계 데이터 모델의 키와 제약 조건에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. R(A, B, C) 릴레이션에서 기본키가 복합키 (A, B)이고 B가 외래키라면, 참조 무결성에 의해 B는 널값을 가질 수 있다.
ㄴ. 주어진 릴레이션 R의 속성들의 부분집합 X에 대해, 어떤 튜플도 동일한 값을 가지지 않는다면, 이러한 속성의 집합 X를 그 릴레이션의 슈퍼키라고 한다.
ㄷ. 참조 무결성 제약은 참조할 수 없는 외래키 값을 가져서는 안 된다는 것을 의미한다.
ㄹ. 대체키와 외래키는 유일성과 최소성을 모두 만족해야 한다.

① ㄱ, ㄴ

② ㄱ, ㄹ

③ ㄴ, ㄷ

④ ㄷ, ㄹ

정답 3번

출제 포인트

이 문제는 기본키와 외래키의 널 허용 조건, 슈퍼키의 정의, 참조 무결성의 의미, 대체키와 외래키의 제약 차이를 구분할 수 있는지를 묻는 문제이다.

핵심 개념

기본키는 각 튜플을 유일하게 식별하는 키로서 유일성과 최소성을 만족하며 널값을 가질 수 없다.
슈퍼키는 튜플을 유일하게 식별할 수 있는 속성 집합으로, 최소성은 요구되지 않는다.
참조 무결성은 외래키 값이 참조 릴레이션의 기본키 값과 일치해야 한다는 제약이며, 외래키 자체에 유일성이나 최소성을 요구하지는 않는다.
대체키는 후보키 중 기본키로 선택되지 않은 키이므로 후보키와 마찬가지로 유일성과 최소성을 만족한다.

NoteBase

선지별 해설

1번 X
ㄱ이 틀리므로 오답이다. ㄱ의 틀린 부분은 "B는 널값을 가질 수 있다"이다. B가 외래키이더라도 기본키 (A, B)의 구성 속성이므로 기본키 제약에 따라 널값을 가질 수 없다. ㄴ은 어떤 두 튜플도 X에 대해 같은 값을 갖지 않으면 X가 슈퍼키라는 설명이므로 맞다.

2번 X
ㄱ과 ㄹ이 틀리므로 오답이다. ㄱ은 기본키를 구성하는 B가 널값을 가질 수 있다고 한 점이 틀렸고, 올바르게는 기본키 속성은 널값을 가질 수 없다. ㄹ의 틀린 부분은 "외래키는 유일성과 최소성을 모두 만족해야 한다"이다. 올바르게는 유일성과 최소성은 후보키·대체키의 성질이며 외래키의 필수 성질이 아니다.

3번 O
ㄴ과 ㄷ이 모두 옳으므로 정답이다. ㄴ은 같은 X값을 갖는 두 튜플이 없으면 X가 튜플을 유일하게 식별하므로 슈퍼키라는 뜻이다. ㄷ은 참조 무결성이 존재하지 않는 참조대상 값을 외래키로 둘 수 없다는 의미이므로 맞다.

4번 X
ㄷ은 맞지만 ㄹ이 틀리므로 오답이다. ㄹ의 틀린 부분은 외래키에도 대체키와 같은 유일성·최소성을 요구한 점이다. 올바르게는 대체키는 유일성과 최소성을 만족하지만 외래키는 참조 관계의 일치 여부가 핵심 제약이다.

오답 포인트

ㄱ에서는 "외래키는 널이 될 수 있다"는 일반 규칙만 기억하고, B가 동시에 기본키의 일부라는 점을 놓치기 쉽다. ㄹ에서는 대체키와 외래키를 모두 "키"라는 이유로 같은 제약으로 묶어 버리기 쉽다. 이 문제의 함정은 기본키 제약과 참조 무결성 제약, 슈퍼키와 후보키의 차이를 섞어 놓은 데 있다.

다시 나오면 이렇게 판단

키 관련 선지는 먼저 그 속성이 기본키 계열인지 외래키인지부터 구분한다. 기본키 계열이면 유일성과 최소성, 널 금지를 확인하고, 슈퍼키이면 유일 식별 가능성만 확인한다. 외래키이면 참조 대상 존재 여부를 보되 유일성이나 최소성까지 자동으로 붙이지 않는다.

암기 포인트

기본키는 널 금지, 슈퍼키는 최소성 불요, 외래키는 참조 일치만 본다.

NoteBase

문 2. 릴레이션 스키마의 표현은 릴레이션명(속성명1:도메인1, 속성명2:도메인2, ..., 속성명n:도메인n)이다. 이 표현을 따르는 릴레이션 스키마 A(a:int, x:int, c:int)와 B(b:int, x:int, d:int)에 대한 관계대수식 $\pi_{a,b}(\sigma_{a>10}(A \bowtie_{A.x=B.x} B))$와 동등한 관계대수식은?

① $\pi_{a,b}(\sigma_{a>10}(A) \times B)$

② $\sigma_{a>10}(\pi_{a,b}(A) \bowtie_{x} B)$

③ $\pi_{a,b}(A \bowtie_{a} \sigma_{a>10}(B))$

④ $\pi_{a,b}(\sigma_{a>10}(A) \bowtie_{A.x=B.x} B)$

정답 4번

출제 포인트

선택 조건 $a > 10$이 릴레이션 A의 속성만 사용하므로 조인 뒤의 선택을 조인 전 A로 밀어 넣을 수 있는지와, 조인 조건 $A.x = B.x$는 그대로 유지해야 하는지를 묻는 문제이다.

핵심 개념

선택 푸시다운은 선택 조건이 한 릴레이션의 속성만 참조할 때 그 릴레이션 앞으로 선택을 내려도 결과가 같다는 성질이다.
세타 조인은 $A.x = B.x$ 같은 조건을 만족하는 튜플만 결합하는 연산이다.
카테시안 곱은 조인 조건 없이 모든 튜플을 결합하므로 세타 조인과 결과가 다르다.

NoteBase

선지별 해설

① X
- 틀린 부분은 $\times B$이다.
- 원식은 $A.x = B.x$ 조건이 있는 조인인데, ①은 조건 없는 카테시안 곱으로 바꾸어 결과 튜플 수와 의미가 달라진다.
- 올바르게는 $\pi_{a,b}(\sigma_{a>10}(A) \bowtie_{A.x=B.x} B)$이다.

② X
- 틀린 부분은 $\pi_{a,b}(A)$이다.
- $\pi_{a,b}(A)$를 먼저 수행하면 조인에 필요한 속성 x가 사라져서 $\bowtie_x$를 수행할 수 없다.
- 올바르게는 x를 남긴 상태에서 조인한 뒤 $\pi_{a,b}$를 적용해야 하므로 $\pi_{a,b}(\sigma_{a>10}(A) \bowtie_{A.x=B.x} B)$가 된다.

③ X
- 틀린 부분은 $\sigma_{a>10}(B)$와 $\bowtie_a$이다.
- 스키마 $B(b:int, x:int, d:int)$에는 속성 a가 없으므로 B에 대해 $a>10$ 선택을 할 수 없고, a 기준 조인도 문제의 원식과 다르다.
- 올바르게는 조건 $a>10$을 A에 적용하고 $A.x = B.x$로 조인해야 하므로 $\pi_{a,b}(\sigma_{a>10}(A) \bowtie_{A.x=B.x} B)$이다.

④ O
- $a>10$은 A의 속성만 쓰는 조건이므로 $\sigma_{a>10}$를 조인 전 A에 먼저 적용해도 된다.
- 그 뒤 $A.x = B.x$ 조건으로 조인하고 마지막에 a, b만 투영하므로 원식 $\pi_{a,b}(\sigma_{a>10}(A \bowtie_{A.x=B.x} B))$와 동등하다.

오답 포인트

선택을 조인 앞으로 내릴 수 있다는 사실만 기억하고 조인 조건 $A.x = B.x$를 빠뜨리거나, 조인 키 x를 투영으로 먼저 지워 버리면 틀리기 쉽다. ②와 ③은 바로 그 함정을 노린 선지이다.

다시 나오면 이렇게 판단

먼저 선택 조건이 어느 릴레이션의 속성만 쓰는지 본다. 한 릴레이션의 속성만 쓰면 그 릴레이션 앞으로 선택을 내릴 수 있고, 조인 조건에 필요한 속성은 조인이 끝나기 전에 투영으로 제거하면 안 된다.

암기 포인트

한쪽 릴레이션의 속성만 쓰는 선택은 그쪽으로 내릴 수 있고, 조인 키는 조인이 끝날 때까지 남겨야 한다.

NoteBase

문 3. 스키마가 사원(사원번호, 직급, 보너스)인 사원 테이블의 인스턴스가 <보기 1>과 같을 때, <보기 2>의 ㉠, ㉡에 들어갈 말로 옳게 짝 지은 것은?

<보기 1>

사원번호	직급	보너스
1000	과장	500
1001	과장	500
1002	사원	200

<보기 2>

이 사원 테이블에서 보너스는 직급별로 결정된다. 이때 직급과 보너스 속성은 모두 기본키인 사원번호에 의해 결정되지만 사원번호가 직급을 결정하고 직급이 보너스를 결정하는 (㉠) 관계를 가지고 있으므로, 이 테이블은 (㉡)이라고 할 수 있다.

① ㉠ 완전 함수 종속, ㉡ 제 2 정규형

② ㉠ 완전 함수 종속, ㉡ 제 3 정규형

③ ㉠ 이행적 함수 종속, ㉡ 제 2 정규형

④ ㉠ 이행적 함수 종속, ㉡ 제 3 정규형

정답 3번

출제 포인트

이 문제는 이행적 함수 종속과 완전 함수 종속을 구분하고, 그 결과로 제2 정규형과 제3 정규형을 판정하는 문제이다.

NoteBase

핵심 개념

완전 함수 종속: 어떤 속성이 기본키 전체에 종속되고 기본키의 일부에만은 종속되지 않는 관계이다. 키워드는 부분 종속 배제이다.
이행적 함수 종속: 기본키가 다른 비주요 속성을 거쳐 또 다른 비주요 속성을 결정하는 관계이다. 키워드는 기본키→비주요 속성→비주요 속성이다.
제2 정규형: 제1 정규형을 만족하면서 부분 함수 종속이 없는 상태이다. 기본키가 단일 속성이면 보통 자동으로 만족한다.
제3 정규형: 제2 정규형을 만족하면서 이행적 함수 종속이 없는 상태이다. 비주요 속성이 다른 비주요 속성을 결정하면 만족하지 못한다.

선지별 해설

① X
완전 함수 종속이 아니라 이행적 함수 종속이다. 사원번호가 직급을 결정하고 직급이 다시 보너스를 결정하므로 사원번호→보너스는 직급을 거쳐 성립한다.
틀린 표현은 '완전 함수 종속'이다. 올바르게는 '이행적 함수 종속'이다.

② X
㉠은 이행적 함수 종속이므로 첫 표현부터 틀리다. 또한 이행적 함수 종속이 남아 있으므로 이 테이블은 제3 정규형이 아니다.
틀린 표현은 '완전 함수 종속, 제3 정규형'이다. 올바르게는 '이행적 함수 종속, 제2 정규형'이다.

③ O
보너스는 직급별로 결정되므로 직급→보너스가 성립하고, 기본키인 사원번호가 직급을 결정하므로 이행적 함수 종속이 존재한다. 기본키가 사원번호 하나이므로 부분 함수 종속은 없어 제2 정규형은 만족한다.

④ X
이행적 함수 종속이라는 판단은 맞지만 제3 정규형이라는 판단이 틀리다. 비주요 속성인 직급이 다른 비주요 속성인 보너스를 결정하므로 제3 정규형을 만족하지 못한다.
틀린 표현은 '제3 정규형'이다. 올바르게는 '제2 정규형'이다.

오답 포인트

기본키가 사원번호 하나라는 이유만으로 완전 함수 종속이나 제3 정규형으로 곧바로 가면 함정에 걸린다. 이 문항의 핵심은 '보너스는 직급별로 결정된다'에서 직급→보너스를 읽어 내고, 그 순간 이행적 함수 종속과 제3 정규형 위반을 함께 판단하는 데 있다.

NoteBase

다시 나오면 이렇게 판단

기본키→중간 속성→다른 속성 구조가 보이면 먼저 이행적 함수 종속을 의심한다. 기본키가 단일 속성이면 제2정규형은 대체로 만족하므로, 남는 판단은 비주요 속성 사이 종속 때문에 제3 정규형이 깨지는지 여부이다.

암기 포인트

단일 기본키는 2NF를 쉽게 만족하고, 비주요 속성이 다른 비주요 속성을 결정하면 3NF가 깨진다.

NoteBase

문 4. 데이터베이스 시스템에서 데이터 저장 요구량이 빠르게 증가하고 있어서 많은 수의 디스크가 요구된다. 다수의 디스크 드라이브를 사용하여 저장 용량을 늘리고 읽기와 쓰기를 병렬로 수행하기도 하며, 디스크의 고장에 대비하기 위해 RAID(Redundant Arrays of Independent Disks)를 구성하여 활용한다. 이와 같은 RAID에서 1 TByte 디스크 드라이브 6개를 이용하여 RAID를 구성할 때, 구성된 RAID의 저장 용량이 가장 작은 구성 방법은?

① 레벨 0

② 레벨 1

③ 레벨 5

④ 레벨 6

정답 2번

출제 포인트

이 문제는 RAID 레벨별 중복 방식에 따라 실제 사용 가능한 저장 용량이 어떻게 달라지는지를 묻는 문제이다.

핵심 개념

RAID 0은 스트라이핑만 사용하므로 중복 저장이 없어 전체 디스크 용량을 그대로 사용한다.
RAID 1은 미러링으로 같은 데이터를 중복 저장하므로 이 시험의 전형적 용량 규칙에서는 실사용 용량이 1개 디스크 용량이 된다.
RAID 5는 패리티 1개 분량을 사용하므로 실사용 용량은 전체 디스크 수에서 1개를 뺀 크기이다.
RAID 6은 이중 패리티 2개 분량을 사용하므로 실사용 용량은 전체 디스크 수에서 2개를 뺀 크기이다.

NoteBase

선지별 해설

① X
틀린 부분은 "레벨 0"이다.
RAID 0은 장애 대비 기능이 없는 대신 6개 디스크를 모두 저장에 사용하므로 용량은 6TB이다.
올바르게는 가장 작은 저장 용량을 만드는 구성은 레벨 1이다.

② O
RAID 1은 미러링으로 동일 데이터를 중복 저장하므로 6개의 1TB 디스크를 써도 실사용 용량은 1TB로 본다.
보기 중 저장 용량이 가장 작으므로 정답이다.

③ X
틀린 부분은 "레벨 5"이다.
RAID 5는 패리티 1개 분량만 제외하므로 실사용 용량은 $6 - 1 = 5$TB이다.
올바르게는 레벨 5는 레벨 1보다 훨씬 큰 저장 용량을 제공한다.

④ X
틀린 부분은 "레벨 6"이다.
RAID 6은 이중 패리티 2개 분량을 제외하므로 실사용 용량은 $6 - 2 = 4$TB이다.
올바르게는 레벨 6도 용량이 줄어들지만 가장 작은 구성은 레벨 1이다.

오답 포인트

패리티가 들어가는 RAID 5, RAID 6이 용량을 많이 잃는다고 생각하면 오답이 된다. 실제로는 전체를 복제하는 미러링 RAID 1이 저장 용량 손실이 가장 크다.

다시 나오면 이렇게 판단

먼저 중복 저장 방식부터 구분한다. 중복이 없으면 전체 n개를 쓰고, 패리티 1개면 $n - 1$, 패리티 2개면 $n - 2$, 전체 미러링이면 1개 디스크 분으로 판단한다.

암기 포인트

RAID 용량은 0은 n, 1은 1, 5는 $n - 1$, 6은 $n - 2$로 정리된다.

NoteBase

문 5. 다음은 산업통계 테이블이다. 이 테이블을 대상으로 아래 결과를 출력하고자 한다. 이를 위한 SQL 질의문은?

산업통계

대분류	소분류	종사자수	평균연봉
제조업	반도체	10000	4000
제조업	스마트폰	5000	5000
제조업	가전	7000	6000
IT	SW개발	6000	4000
IT	게임개발	3000	3000
서비스업	미용	4000	3500
서비스업	유통	9000	4500

SQL 실행결과

대분류	소분류	종사자수	평균연봉
제조업	반도체	10000	4000
IT	SW개발	6000	4000
서비스업	유통	9000	4500

①
```
SELECT *
FROM 산업통계
WHERE 종사자수 >= 6000;
```

②
```
SELECT *
FROM 산업통계
WHERE 종사자수 = (SELECT MAX(종사자수)
                  FROM 산업통계);
```

③
```
SELECT A.*
FROM 산업통계 A,
            (SELECT 대분류, MAX(종사자수) AS 종사자수
             FROM 산업통계
             GROUP BY 대분류) B
WHERE A.종사자수 = B.종사자수 AND A.대분류 = B.대분류;
```

④
```
SELECT 대분류, 소분류, MAX(종사자수), 평균연봉
FROM 산업통계
GROUP BY 대분류;
```

NoteBase

정답 3번

출제 포인트

이 문제는 원표와 실행 결과를 대조하여 대분류별로 종사자수가 가장 큰 행만 남기는 SQL을 찾는 문제이다.

자료 읽는 순서

1. 실행 결과가 제조업, IT, 서비스업으로 대분류마다 1행씩만 남는지 확인한다.
2. 남은 각 행의 종사자수가 해당 대분류의 최대값인지 원표에서 대조한다.
3. 대분류별 최대 종사자수를 구한 뒤 그 값과 원표를 다시 연결하는 형태인지 선택지를 판단한다.

선지별 해설

① X: `WHERE 종사자수 >= 6000`은 단순 조건 필터이다. 원표에서 종사자수 6000 이상인 행을 모두 남기므로 제조업의 `가전(7000)`도 함께 출력된다. 틀린 표현은 `종사자수 >= 6000`만으로 결과표가 된다는 부분이다. 올바르게는 대분류별 최대 종사자수 행만 남겨야 한다.
② X: `(SELECT MAX(종사자수) FROM 산업통계)`는 전체 테이블의 최대값 10000만 구한다. 그래서 제조업 반도체 1행만 남고 IT와 서비스업은 빠진다. 틀린 표현은 `전체 최대값`으로 결과표를 만들 수 있다는 부분이다. 올바르게는 대분류별로 각각 최대값을 구해야 한다.
③ O: 서브쿼리 B가 `GROUP BY 대분류`로 대분류별 `MAX(종사자수)`를 구하고, 바깥 질의가 `A.대분류 = B.대분류`와 `A.종사자수 = B.종사자수`로 원표와 다시 연결한다. 그래서 제조업의 반도체, IT의 SW개발, 서비스업의 유통이 정확히 출력된다.
④ X: `GROUP BY 대분류`만 하면서 `소분류, 평균연봉`을 함께 그대로 선택하는 방식은 결과표 의도와 맞지 않는다. 집계 결과에서 필요한 것은 대분류별 최대 종사자수이고, 그 최대값에 대응하는 소분류와 평균연봉은 원표에서 다시 찾아와야 한다. 틀린 표현은 `소분류, 평균연봉`을 집계 없이 바로 선택한다는 부분이다. 올바르게는 대분류별 `MAX(종사자수)`를 구한 뒤 원표와 조인하여 해당 행 전체를 가져와야 한다.

오답 포인트

이 문항의 함정은 `조건에 맞는 행 걸러내기`와 `그룹별 최대 행 1개 뽑기`를 혼동하는 데 있다. ①은 단순 필터라서 같은 대분류에서 여러 행이 남을 수 있고, ②는 그룹별이 아니라 전체 최대 1행만 남긴다. ④는 집계값과 비집계 열을 한 번에 가져오면 된다고 착각하게 만든 선택지이다.

다시 나오면 이렇게 판단

실행 결과가 그룹마다 대표 1행씩 남아 있으면 먼저 `그룹별 집계인지`를 본다. 그다음 대표 기준이 최대값이나 최소값이면 `GROUP BY로 기준값을 구한 뒤 원표와 다시 연결하는가`를 확인하면 된다.

NoteBase

암기 포인트

그룹별 최대 행 전체가 필요하면 `GROUP BY + MAX`만으로 끝나지 않고, 집계 결과를 원표와 다시 연결해야 한다.

NoteBase

문 6. 버킷 용량 $C = 2$인 버킷들의 주소 공간이 $0 \sim (N - 1)$인 해시에서 키를 K로 하는 해시 함수 h(K)=K N이고, N 값은 5이다. 이때 일련의 K 값들이 3, 5, 7, 9, 11, 13, 18, 14 순서로 삽입된 후 4번 버킷의 값은? (단, 버킷에서의 충돌은 충돌 발생 버킷 주소 i에 대하여 (i+1) N를 통해 지정하는 선형탐색 개방 주소법을 이용하여 해결한다)

① 5, 14

② 3, 13

③ 9, 14

④ 9, 18

정답 4번

출제 포인트

이 문제는 버킷 용량이 2인 해시에서 선형탐색 개방 주소법으로 키들이 어느 버킷에 최종 저장되는지를 묻는다.

올바른 순서

1. 3, 5, 7, 9, 11은 각각 3번, 0번, 2번, 4번, 1번 버킷에 먼저 들어간다.
2. 13은 13 5 = 3이므로 3번 버킷의 두 번째 칸에 들어가서 3번 버킷이 [3, 13]이 된다.
3. 18은 18 5 = 3이지만 3번 버킷이 가득 차 있으므로 다음 버킷 4로 이동해 [9, 18]이 된다.
4. 14는 14 5 = 4이지만 4번 버킷도 가득 차 있으므로 0번 버킷으로 이동해 [5, 14]가 된다.

왜 이 순서인가: 선형탐색은 해시 주소 버킷이 가득 찼을 때만 다음 버킷으로 이동하므로, 먼저 들어간 18이 4번 버킷의 남은 한 칸을 차지하고 그 뒤의 14가 밀려난다.

선지별 해설

1번 X: 5, 14는 0번 버킷의 값이다. 14는 원래 4번 해시 주소를 가지지만 4번 버킷이 이미 [9, 18]로 가득 차서 다음 버킷 0으로 이동한다. 올바르게는 0번 버킷이 5, 14이다.

2번 X: 3, 13은 3번 버킷의 값이다. 두 키 모두 해시값이 3이어서 3번 버킷의 두 칸을 그대로 채운다. 올바르게는 3번 버킷이 3, 13이다.

3번 X: 9는 4번 버킷이 맞지만 14는 4번 버킷에 남지 못한다. 18이 먼저 4번 버킷의 빈 칸을 차지하므로 14는 선형탐색으로 0번 버킷으로 이동한다. 올바르게는 4번 버킷이 9, 18이다.

4번 O: 9는 9 5 = 4이므로 4번 버킷에 들어가고, 18은 18 5 = 3이지만 3번 버킷 [3, 13]이 가득 차서 다음 버킷 4에 저장된다. 따라서 4번 버킷의 값은 9, 18이다.

오답 포인트

18과 14의 삽입 순서를 무시하면 14가 4번 버킷에 남는다고 착각하기 쉽다. 버킷형 선형탐색은 같은 버킷의 남은 칸을 먼저 채우고, 가득 찬 뒤에만 다음 버킷으로 이동한다는 점이 함정이다.

NoteBase

다시 나오면 이렇게 판단

해시 문제는 각 키에 대해 해시값을 먼저 적고, 해당 버킷의 남은 칸 수를 확인한 뒤, 가득 찬 경우에만 한 버킷씩 순서대로 이동시키면 된다. 특히 나중에 들어오는 키는 먼저 들어온 충돌 키가 차지한 자리를 넘겨받을 수 없다는 점을 확인하면 된다.

암기 포인트

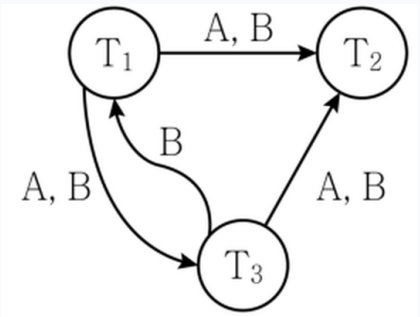
버킷 용량이 2이면 같은 해시 주소의 두 칸을 먼저 채우고, 그다음부터 선형탐색으로 다음 버킷에 넘긴다.

NoteBase

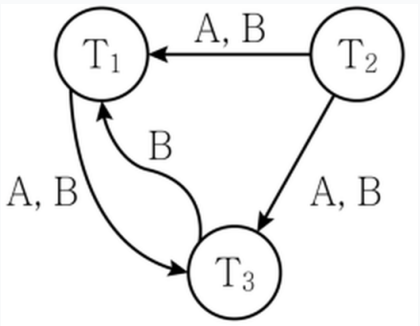
문 7. 다음 트랜잭션 스케줄의 선행그래프(precedence graph)로 옳은 것은? (단, t_i는 시간 단위, T_j는 트랜잭션, R(A)는 A항목 읽기, W(A)는 A항목 쓰기를 나타낸다)

시간	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
T_1	R(A)	W(A)				R(B)		W(B)		
T_2					R(A)					R(B)
T_3			R(A)	W(A)			R(B)		W(B)	

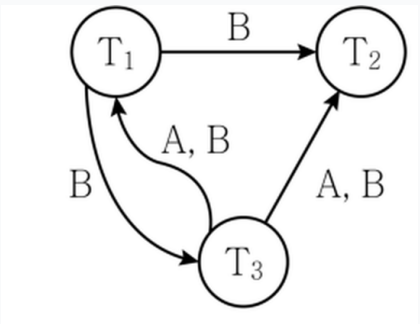
①



②

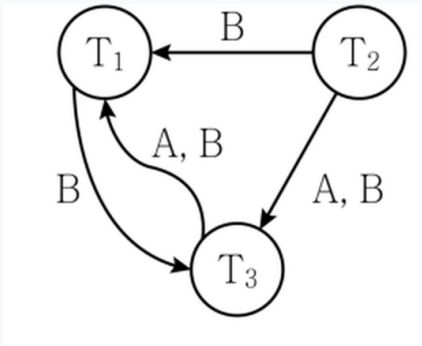


③



NoteBase

④



정답 1번

출제 포인트

이 문제는 같은 데이터 항목에 대한 충돌 연산의 시간 순서를 비교하여 선행그래프의 간선 방향을 정확히 정하는 문제이다.

선지별 해설

① O

- A항목에서는 T_1의 $W(A)$가 T_3의 $R(A)$, $W(A)$, T_2의 $R(A)$보다 먼저이므로 $T_1 \rightarrow T_3$, $T_1 \rightarrow T_2$가 생긴다. 또 T_3의 $W(A)$가 T_2의 $R(A)$보다 먼저이므로 $T_3 \rightarrow T_2$가 생긴다.
- B항목에서는 T_3의 $R(B)$가 T_1의 $W(B)$보다 먼저이므로 $T_3 \rightarrow T_1$이 생기고, T_1의 $W(B)$와 T_3의 $W(B)$는 모두 T_2의 $R(B)$보다 먼저이므로 $T_1 \rightarrow T_2$, $T_3 \rightarrow T_2$가 생긴다.
- 따라서 간선은 $T_1 \rightarrow T_2$(A,B), $T_1 \rightarrow T_3$(A,B), $T_3 \rightarrow T_1$(B), $T_3 \rightarrow T_2$(A,B)이고 ①이 맞다.

② X

- ②는 $T_2 \rightarrow T_1$(A,B)로 그렸지만 실제 충돌 순서는 T_1의 쓰기가 T_2의 읽기보다 앞서므로 방향은 $T_1 \rightarrow T_2$이다.
- 틀린 표현은 $T_2 \rightarrow T_1$이고, 올바르게는 $T_1 \rightarrow T_2$이다.

③ X

- ③은 $T_1 \rightarrow T_2$를 B만으로 표시했지만 A항목에서도 T_1의 $W(A)$가 T_2의 $R(A)$보다 먼저이므로 A에 의한 간선도 반드시 있어야 한다.
- 틀린 표현은 $T_1 \rightarrow T_2$의 라벨을 B만 둔 것이고, 올바르게는 $T_1 \rightarrow T_2$(A,B)이다.

④ X

- ④는 T_1과 T_2 사이의 B 간선 방향을 거꾸로 그렸다. T_1은 t_8에 $W(B)$를 하고 T_2는 t_{10}에 $R(B)$를 하므로 충돌 방향은 $T_1 \rightarrow T_2$이다.
- 틀린 표현은 $T_2 \rightarrow T_1$(B)이고, 올바르게는 $T_1 \rightarrow T_2$(A,B)이다.

오답 포인트

읽기-읽기는 충돌이 아니라는 점과 간선 방향을 항상 선행 연산에서 후행 연산으로 잡는다는 점이 함정이다. 특히 T_2는 A와 B를 읽기만 하므로, 앞서 쓰기를 수행한 T_1, T_3에서 T_2로 향하는 간선이 생긴다.

NoteBase

다시 나오면 이렇게 판단

항목 A와 B를 분리해서 시간순으로 본 뒤 읽기-쓰기, 쓰기-읽기, 쓰기-쓰기에만 표시를 하면 된다. 같은 항목에서 먼저 수행한 충돌 연산의 트랜잭션에서 나중에 수행한 트랜잭션으로 간선을 그리면 선행그래프가 정리된다.

암기 포인트

선행그래프는 같은 항목의 충돌 연산만 본다. 간선 방향은 항상 선행 연산 트랜잭션 → 후행 연산 트랜잭션이다.

NoteBase

문 8. 다음 SQL 구문에 대한 설명으로 옳지 않은 것은?

```
CREATE ASSERTION NO_DESIGNERS_IN_SEOUL
AS CHECK
(  NOT EXISTS
    ( SELECT *
      FROM EMPLOYEE E, DEPARTMENT D
      WHERE E.DEPTNO = D.DEPTNO
            AND E.JOB = 'DESIGNER'
            AND D.LOC = 'SEOUL' )
);
```

① 명시된 이벤트가 발생할 때마다 DBMS가 자동적으로 수행하는 구문이다.

② 대상 테이블과 관련된 SQL의 갱신문 수행 시 위의 구문이 검사된다.

③ 위의 SQL이 불필요할 경우 DROP문으로 제거할 수 있다.

④ 위의 SQL에서 명세하고 있는 조건에 따라 DB연산이 수행되지 않을 수 있다.

정답 1번

출제 포인트

이 문제는 SQL의 ASSERTION이 트리거처럼 사건이 발생할 때 절차를 실행하는 기능인지, 아니면 데이터베이스 전반의 무결성 조건을 선언하는 기능인지를 구분하는 문제이다.

핵심 개념

ASSERTION은 데이터베이스가 항상 만족해야 하는 무결성 제약을 선언하는 객체이다. 키워드는 조건 선언, 무결성 검사, 위반 시 갱신 거부이다.
TRIGGER는 특정 이벤트가 발생했을 때 자동으로 실행되는 절차적 동작이다. 키워드는 이벤트, 자동 실행, 행동 수행이다.
이 문장의 `CREATE ASSERTION ... AS CHECK (...)`는 서울 부서에 디자이너가 존재하지 않아야 한다는 전역 제약을 선언한 것이다.

NoteBase

선지별 해설

① X
'명시된 이벤트가 발생할 때마다 DBMS가 자동적으로 수행하는 구문'이라는 표현이 틀렸다. 이는 ASSERTION이 아니라 TRIGGER의 설명이다. 올바르게는 '명세한 조건이 항상 참이 되도록 DBMS가 관련 갱신 시 검사하는 무결성 제약 구문'이다.

② O
대상 테이블과 관련된 INSERT, UPDATE, DELETE 같은 갱신문이 수행되면 ASSERTION의 조건이 깨지는지 검사한다. 이 예시는 `EMPLOYEE`와 `DEPARTMENT`를 함께 참조하므로 두 테이블의 관련 갱신이 검사 대상이 된다.

③ O
ASSERTION은 스키마 객체이므로 불필요해지면 DROP문으로 제거할 수 있다. 표준 SQL 관점에서는 `DROP ASSERTION`으로 삭제하는 대상이다.

④ O
ASSERTION의 조건을 위반하는 결과가 생기면 해당 DB 연산은 허용되지 않을 수 있다. 즉 서울 부서에 디자이너가 존재하게 만드는 갱신은 거부될 수 있다.

오답 포인트

①은 '자동적으로 수행'과 '이벤트 발생'이라는 트리거의 핵심 표현을 ASSERTION 설명에 섞어 놓은 함정이다. ASSERTION은 동작을 실행하는 절차가 아니라 조건을 선언하고 위반 여부를 검사하는 제약이라는 점을 놓치면 혼동하기 쉽다.

다시 나오면 이렇게 판단

SQL 객체를 묻는 문제에서 '이벤트가 발생하면 실행된다'는 문구가 보이면 TRIGGER 쪽으로 판단한다. 반대로 `CHECK`, 무결성, 조건 유지, 위반 시 갱신 거부가 중심이면 ASSERTION 같은 제약 선언으로 판단할 수 있다.

암기 포인트

TRIGGER는 사건-행동이고, ASSERTION은 조건-검사이다.

NoteBase

문 9. XML Schema와 DTD에 대한 비교 중 옳지 않은 것으로만 묶은 것은?

	구분	XML Schema	DTD
ㄱ	구조	복잡함	상대적으로 간결함
ㄴ	문법	전용문법	XML문법
ㄷ	Namespace	지원함	지원하지 못함
ㄹ	확장성	문자열 치환을 통한 확장성	객체지향적인 확장성

① ㄱ, ㄴ

② ㄱ, ㄷ

③ ㄴ, ㄹ

④ ㄷ, ㄹ

정답 3번

출제 포인트

이 문제는 XML Schema와 DTD의 문법, Namespace 지원, 확장 방식의 차이를 구별하는 문제이다.

핵심 개념

XML Schema는 XML 문법으로 스키마를 정의하며 데이터형, Namespace, 타입 확장과 제한 같은 풍부한 표현 기능을 지원한다.
DTD는 XML과 별도의 전용 문법으로 문서 구조를 선언하며 엔티티를 통한 문자열 치환을 지원하지만 Namespace와 세밀한 데이터형 표현은 지원하지 못한다.

NoteBase

선지별 해설

① X
- ①은 ㄱ, ㄴ의 묶음이다.
- ㄱ은 맞다. XML Schema는 기능이 많아 구조가 상대적으로 복잡하고, DTD는 선언 방식이 비교적 간결하다.
- ㄴ은 틀리다. 틀린 부분은 'XML Schema 전용문법, DTD XML문법'이라는 표현이다. 올바르게는 'XML Schema는 XML 문법을 사용하고, DTD는 XML이 아닌 전용 문법을 사용한다.'이다.

② X
- ②는 ㄱ, ㄷ의 묶음이다.
- ㄱ은 맞고 ㄷ도 맞다. XML Schema는 Namespace를 지원하지만 DTD는 Namespace를 직접 지원하지 못한다.
- 따라서 ②는 옳지 않은 것만 묶은 선택지가 아니다.

③ O
- ③은 ㄴ, ㄹ의 묶음이다.
- ㄴ은 문법 설명이 서로 뒤바뀌어 틀렸다.
- ㄹ도 틀리다. 틀린 부분은 'XML Schema 문자열 치환을 통한 확장성, DTD 객체지향적인 확장성'이라는 표현이다. 올바르게는 'DTD는 엔티티를 이용한 문자열 치환 기능을 활용하고, XML Schema는 타입 확장과 제한을 통한 객체지향적 확장성을 가진다.'이다.

④ X
- ④는 ㄷ, ㄹ의 묶음이다.
- ㄷ은 맞고 ㄹ만 틀리다.
- 따라서 ④도 옳지 않은 것만 묶은 선택지가 아니다.

오답 포인트

XML Schema와 DTD의 문법 체계가 서로 반대라는 점과, DTD의 엔티티 치환 기능과 XML Schema의 타입 기반 확장성을 뒤섞으면 오답이 된다. 특히 'XML로 쓰는가, 전용문법인가'와 '엔티티 치환인가, 타입 확장인가'를 바꿔 적는 함정이 자주 출제된다.

다시 나오면 이렇게 판단

비교 문제에서는 먼저 문법 형식, Namespace 지원 여부, 확장 방식의 세 축으로 나누어 본다. XML 문법과 Namespace, 타입 확장은 XML Schema 쪽이고, 전용 문법과 엔티티 기반 문자열 치환은 DTD 쪽이면 정답을 빠르게 가를 수 있다.

암기 포인트

DTD는 전용문법·엔티티 치환, XML Schema는 XML문법·Namespace·타입 확장이다.

NoteBase

문 10. 즉시 갱신(Immediate Update) 전략을 이용하는 회복 시스템에서 <보기 1>의 로그 레코드 형식으로 <보기 2>의 로그 레코드가 형성되어 있다. 복구는 REDO 단계 실행 후 UNDO 단계를 실행한다. 이 시스템에서 복구 절차에 대한 설명으로 옳지 않은 것으로만 묶은 것은?

<보기 1>

◯ T_i: 트랜잭션 번호가 i인 트랜잭션

◯ <T_i, start>, <T_i, commit>, <T_i, abort>: T_i에 대한 시작, 완료, 철회를 기록하는 형식

◯ <T_i, rollback>: T_i가 실행 중 rollback이 발생함을 기록하는 형식

◯ <T_i, 데이터항목, 이전값, 이후값>: T_i가 데이터 항목을 변경할 때 기록하는 형식

◯ <T_i, 데이터항목, 이전값>: T_i가 철회되기 위하여 UNDO를 실행할 때 기록하는 형식

◯ <checkpoint, {트랜잭션리스트}>: 검사점 실행을 기록하는 형식

◯ {트랜잭션리스트}: 검사점 실행시점에 실행 중에 있던 트랜잭션들의 목록을 포함하는 트랜잭션 리스트

<보기 2>

로그번호	로그 레코드
1	<T_0, start>
2	<T_0, A, 100, 200>
3	<T_1, start>
4	<T_1, B, 200, 300>
5	<T_0, C, 300, 400>
6	<checkpoint, {T_0, T_1}>
7	<T_1, D, 500, 600>
8	<T_3, start>
9	<T_0, rollback>
10	<T_4, start>
11	<T_0, C, 300>
12	<T_3, E, 600, 700>
13	<T_0, A, 100>
14	<T_4, F, 700, 800>
15	<T_0, abort>
16	<T_4, G, 800, 900>
17	<T_3, H, 900, 950>
18	<T_4, commit>
	--고장 발생--

ㄱ. 복구가 시작되면 checkpoint 로그 레코드를 찾아 UNDO 대상 트랜잭션 리스트를 {T_0, T_1}으로 초기화한다.

ㄴ. T_0의 rollback을 처리하기 위하여 생성된 <T_0, C, 300>, <T_0, A, 100>은 복구과정에서 UNDO 단계 연산이 된다.

ㄷ. REDO 단계를 완료하면 UNDO 대상 트랜잭션 리스트는 {T_1, T_3}로 구성된다.

ㄹ. REDO 단계에서 로그 레코드 <T_1, D, 500, 600>, <T_0, C, 300>, <T_3, E, 600, 700>, <T_0, A, 100>, <T_4, F, 700, 800>, <T_4, G, 800, 900>, <T_3, H, 900, 950>은 REDO 연산의 대상이 된다.

ㅁ. REDO 단계에서는 REDO 연산에 따른 로그 레코드를 로그에 기록한다.

NoteBase

ㅂ. UNDO 단계에서는 <T_3, H, 900>, <T_3, E, 600>, <T_3, abort>, <T_1, D, 500>, <T_1, B, 200>, <T_1, abort>를 로그에 기록한다.

① ㄱ, ㅂ

② ㄴ, ㅁ

③ ㄷ, ㄹ

④ ㄹ, ㅂ

정답 2번

출제 포인트

즉시 갱신 회복에서 checkpoint 이후 REDO 대상과 crash 시점의 최종 UNDO 대상을 구분하고, rollback 중 생성된 보상 로그가 어떤 단계에서 처리되는지, 그리고 새 로그 레코드는 REDO가 아니라 UNDO 단계에서만 기록된다는 점을 묻는 문제이다.

올바른 순서

1. 최근 checkpoint를 찾고 트랜잭션 리스트 {T0, T1}를 복구의 초기 활성 트랜잭션 집합으로 둔다.
2. checkpoint 이후 로그를 앞으로 스캔하면서 갱신 로그와 rollback 중 생성된 보상성 로그를 REDO하고, start/commit/abort에 따라 활성 집합을 갱신한다. 이 전진 스캔에서 T3(log 8)와 T4(log 10)가 추가되고, <T0, abort>(log 15)로 T0가, <T4, commit>(log 18)로 T4가 제거된다.
3. 전진 스캔이 끝나면 commit이나 abort로 종료되지 않은 트랜잭션은 T1과 T3만 남는다.
4. 그다음 뒤로 스캔하면서 T3, T1의 갱신 로그를 UNDO하고 각 트랜잭션의 철회가 끝나면 abort를 기록한다.

왜 이 순서: <T0, C, 300>, <T0, A, 100>은 T0를 이미 되돌리면서 남긴 보상 로그이므로 복구 때 다시 UNDO할 대상이 아니라 REDO로 재현할 대상이다.

NoteBase

선지별 해설

① X (ㄱ, ㅂ)
ㄱ은 옳다. checkpoint 레코드의 트랜잭션 리스트 {T0, T1}에서 UNDO 대상을 초기화하는 설명은 맞다.
ㅂ도 옳다. UNDO 단계는 종료되지 않은 T3, T1을 역순으로 취소하며 보상 로그와 abort를 남긴다. <T3, H, 900>, <T3, E, 600>, <T3, abort>, <T1, D, 500>, <T1, B, 200>, <T1, abort>는 실제 UNDO 결과와 정확히 일치한다.
둘 다 옳으므로 옳지 않은 것만 묶은 조합이 아니다.

② O (ㄴ, ㅁ)
ㄴ은 옳지 않다. <T0, C, 300>, <T0, A, 100>은 T0의 rollback 중 생성된 보상 로그이고 T0는 log 15의 <T0, abort>로 이미 종료되었으므로, 복구 시 UNDO 대상이 아니라 REDO 단계에서 다시 적용되는 로그이다. 따라서 이들이 'UNDO 단계 연산이 된다'는 서술은 틀리다.
ㅁ은 옳지 않다. ㅁ은 'REDO 단계에서는 REDO 연산에 따른 로그 레코드를 로그에 기록한다'고 하는데, REDO 단계는 이미 로그에 기록되어 있는 갱신을 데이터베이스에 다시 적용(재현)할 뿐 새로운 로그 레코드를 기록하지 않는다. 복구 중 새 로그 레코드(보상 로그와 abort)를 기록하는 단계는 UNDO 단계이다.
따라서 옳지 않은 것만 묶은 조합은 ㄴ, ㅁ이다.

③ X (ㄷ, ㄹ)
ㄷ은 옳다. 전진 스캔이 끝나면 commit이나 abort로 종료되지 않은 트랜잭션은 T1, T3뿐이므로 UNDO 대상 리스트는 {T1, T3}가 된다.
ㄹ도 옳다. checkpoint 이후의 갱신 로그와 rollback 중 생성된 <T0, C, 300>, <T0, A, 100>은 모두 REDO 대상이다.
둘 다 옳으므로 ③은 오답 조합이다.

④ X (ㄹ, ㅂ)
ㄹ은 옳다(③ 참조). REDO 단계는 checkpoint 이후의 갱신 결과를 모두 재현하므로 제시된 로그들이 REDO 대상이 된다.
ㅂ도 옳다(① 참조). UNDO 단계가 실제로 기록하는 보상 로그·abort와 일치한다.
둘 다 옳으므로 옳지 않은 것만 묶은 조합이 아니다.

오답 포인트

첫째, REDO 단계가 로그 레코드를 새로 기록한다고 착각하기 쉽다(ㅁ). REDO는 로그에 이미 남은 갱신을 다시 적용만 하고, 새 로그(보상 로그·abort)를 기록하는 것은 UNDO 단계이다. 둘째, rollback 중 생성된 <T0, C, 300>, <T0, A, 100>을 다시 UNDO해야 한다고 보는 것(ㄴ)이다. 이 로그들은 이미 되돌린 결과를 담은 보상 로그이고 T0는 abort로 종료되었으므로 REDO 대상이다.

NoteBase

다시 나오면 이렇게 판단

즉시 갱신 복구 문제는 checkpoint 이후를 먼저 앞으로 읽어 종료된 트랜잭션과 미종료 트랜잭션을 가른 뒤, rollback 중 남겨진 보상 로그인지 아직 철회되지 않은 갱신 로그인지를 구분한다. 또 새 로그 레코드는 REDO가 아니라 UNDO 단계에서만 남는다는 점을 함께 기억하면 판단이 정리된다.

암기 포인트

REDO는 기록된 결과를 다시 적용만 하고 로그를 새로 남기지 않는다. 새 로그(보상 로그·abort)는 UNDO 단계에서 기록하며, 끝까지 종료되지 않은 트랜잭션만 나중에 UNDO한다.

NoteBase

문 11. 데이터베이스에 대한 설명으로 옳지 않은 것은?

① 데이터 종속성이란 응용프로그램과 데이터 간의 상호의존관계를 의미하며, 이는 응용프로그램이 접근하려는 데이터의 구성 방법에 맞게 작성되어야 함을 의미한다.

② 데이터 중복성이란 한 시스템 내에 내용이 같은 데이터가 중복 저장 관리되는 것을 의미하며, 경제성 문제는 발생하나 보안성은 향상된다.

③ 논리적 데이터 독립성이란 기존 응용프로그램에 영향을 주지 않고 데이터베이스의 논리적 구조를 변경시킬 수 있는 능력을 의미한다.

④ 데이터 일관성이란 여러 개의 데이터가 모두 하나의 사실을 나타낸다면 논리적으로 그 내용이 모두 동일해야 함을 의미한다.

정답 2번

출제 포인트

이 문제는 데이터 종속성, 데이터 중복성, 논리적 데이터 독립성, 데이터 일관성의 정의와 효과를 구별하는 문제이다.

핵심 개념

데이터 종속성: 응용프로그램이 데이터의 논리적 구조나 저장 구조에 의존하는 성질이다. 키워드는 프로그램이 데이터 구조 변경의 영향을 받는다는 점이다.
데이터 중복성: 같은 데이터가 여러 곳에 중복 저장되는 상태이다. 키워드는 저장 낭비, 갱신 이상, 불일치 위험이다.
논리적 데이터 독립성: 응용프로그램에 큰 영향을 주지 않고 데이터베이스의 논리적 구조를 변경할 수 있는 성질이다. 키워드는 외부 스키마와 개념 스키마의 분리이다.
데이터 일관성: 여러 데이터 사이에 모순이 없어 같은 사실에 대해 동일한 결과를 유지하는 성질이다. 키워드는 모순 없음이다.

선지별 해설

① O: 데이터 종속성은 프로그램과 데이터가 상호 의존하여 데이터 구조가 바뀌면 프로그램도 영향을 받는 관계를 뜻한다. 선지 설명은 이 정의와 맞다.
② X: 데이터 중복성은 같은 데이터가 중복 저장되는 현상이라는 정의 자체는 맞지만, '보안성은 향상된다'가 틀린 표현이다. 실제로는 중복이 늘수록 관리 지점이 많아져 보안 유지와 일관성 관리가 더 어려워질 수 있다.
③ O: 논리적 데이터 독립성은 개념 스키마 수준의 논리 구조를 바꾸더라도 기존 응용프로그램을 크게 수정하지 않도록 하는 성질이다. 선지 설명이 맞다.
④ O: 데이터 일관성은 동일한 사실을 표현하는 데이터들 사이에 모순이 없어야 한다는 뜻이다. 선지 설명이 맞다.

NoteBase

오답 포인트

데이터 중복성을 단순히 복사본이 많아 안전하다고 오해하면 ②를 고르지 못한다. 이 개념은 백업의 장점이 아니라 중복 저장으로 인한 저장 낭비, 갱신 이상, 불일치 위험을 떠올려야 하는 표현이다.

다시 나오면 이렇게 판단

정의형 문항에서는 각 용어의 핵심 효과를 한 단어로 연결하면 판단이 빨라진다. 종속성은 의존, 중복성은 낭비와 불일치, 논리적 독립성은 논리 구조 변경 허용, 일관성은 모순 없음으로 연결하면 된다.

암기 포인트

종속성은 프로그램이 데이터에 끌려가는 성질이고, 독립성은 그 영향을 줄이는 성질이다. 중복성은 저장 낭비와 불일치 위험, 일관성은 같은 사실에 대한 모순 없음이다.

NoteBase

문 12. 트랜잭션의 특성에 대한 설명으로 옳지 않은 것은?

① S은행 계좌에서 K은행 계좌로 이체하는 도중에 오류가 발생하면 수행된 트랜잭션을 취소하여 원래의 상태로 만들어야 한다.

② S은행 계좌와 K은행 계좌 금액의 합은 S계좌에서 K계좌로 이체한 후에도 그 금액의 합이 같아야 한다.

③ S은행 계좌에서 K은행 계좌로의 이체가 성공적으로 완료된 후에는 항상 이체가 완료된 상태를 유지하여야 한다.

④ S은행 계좌에서 K은행 계좌로 이체하는 트랜잭션을 수행하는 동안 다른 트랜잭션이 관여할 수 없으나 참조할 수는 있다.

정답 4번

출제 포인트

이 문제는 트랜잭션의 ACID 특성인 원자성, 일관성, 지속성, 고립성의 의미를 구분할 수 있는지를 묻는 문제이다.

핵심 개념

원자성은 트랜잭션이 전부 수행되거나 전부 취소되는 성질이다.
일관성은 트랜잭션 수행 전후에 데이터베이스가 정해진 제약과 의미를 계속 만족하는 성질이다.
지속성은 성공적으로 완료된 결과가 장애가 발생해도 보존되는 성질이다.
고립성은 동시에 수행되는 다른 트랜잭션이 중간 수행 결과에 간섭하거나 그 미완료 상태를 보지 못하게 하는 성질이다.

NoteBase

선지별 해설

① O
이체 도중 오류가 나면 이미 수행한 작업까지 포함해 전체를 취소하여 원래 상태로 되돌려야 한다. 이는 일부만 반영되는 것을 막는 원자성의 설명과 맞다.

② O
계좌이체 전후에 두 계좌 금액의 총합이 같아야 한다는 설명은 데이터의 의미와 제약이 유지되어야 한다는 일관성의 설명과 맞다.

③ O
이체가 성공적으로 완료된 뒤에는 그 결과가 계속 유지되어야 한다. 이는 커밋된 결과가 장애 후에도 보존되는 지속성의 설명과 맞다.

④ X
틀린 부분은 "다른 트랜잭션이 관여할 수 없으나 참조할 수는 있다"는 표현이다. 고립성은 다른 트랜잭션이 현재 트랜잭션의 미완료 중간 상태를 참조하거나 간섭하지 못하게 하는 성질이다. 올바르게는 "트랜잭션 수행 중에는 다른 트랜잭션이 그 중간 결과를 참조하거나 영향을 주어서는 안 된다"이다.

오답 포인트

④는 고립성을 약하게 이해하면 헷갈리기 쉽다. '관여는 못하지만 참조는 가능하다'라고 보면 단순 읽기는 허용된다고 착각하기 쉬우나, ACID에서 고립성은 미완료 중간 결과의 노출 자체를 막는 개념으로 정리한다.

다시 나오면 이렇게 판단

오류 시 전체 취소이면 원자성, 수행 전후 값의 의미와 제약 유지이면 일관성, 성공 결과의 보존이면 지속성, 동시에 수행될 때 서로의 중간 결과를 보지 못하게 하면 고립성으로 판단한다.

암기 포인트

원자성은 전부 아니면 전무, 일관성은 합과 제약 유지, 지속성은 커밋 결과 보존, 고립성은 중간 결과 차단이다.

NoteBase

문 13. 다음의 사원 테이블에 대한 SQL 질의문의 결과는? (단, 열 A, B는 모두 INTEGER 타입이고, 값이 표시되어 있지 않은 열은 NULL이다)

사원	
A	B
1000	500
1000	1000
2000	1000
	4000

```
사원 테이블에 대한 SQL 질의문

    SELECT COUNT( A ), COUNT( DISTINCT A ),
           MAX( A+B )
    FROM 사원;
```

① 3 2 3000

② 3 2 4000

③ 4 3 3000

④ 4 3 4000

정답 1번

출제 포인트

COUNT, COUNT(DISTINCT), MAX가 NULL과 중복값을 어떻게 처리하는지를 함께 묻는 문제이다.

조건 정리

행은

1행: A=1000, B=500

2행: A=1000, B=1000

3행: A=2000, B=1000

4행: A=NULL, B=4000이다.

주의할 조건은 COUNT(A)는 NULL을 제외하고, COUNT(DISTINCT A)는 NULL과 중복을 제외하며,

MAX(A+B)는 각 행의 A+B 결과 중 NULL이 아닌 값만 비교한다는 점이다.

NoteBase

계산 과정

1. COUNT(A)는 A가 NULL이 아닌 행만 세므로 3이다.
2. DISTINCT A 값은 1000, 2000 두 개이므로 COUNT(DISTINCT A)는 2이다.
3. A+B는 1500, 2000, 3000, NULL이 된다. 4행은 A가 NULL이어서 A+B도 NULL이다.
4. MAX(A+B)는 NULL을 제외한 값 중 최댓값이므로 3000이다.

선지별 해설

① O: 결과는 3, 2, 3000이다. COUNT(A)=3, COUNT(DISTINCT A)=2, MAX(A+B)=3000이므로 맞는 선지이다.
② X: 틀린 표현은 마지막 값 4000이다. 올바르게는 3000이다. B가 4000인 행은 A가 NULL이므로 A+B가 4000이 아니라 NULL이 되고, MAX는 이 NULL 결과를 최댓값으로 쓰지 않는다.
③ X: 틀린 표현은 첫째 값 4와 둘째 값 3이다. 올바르게는 3과 2이다. COUNT(A)는 NULL인 4행을 세지 않고, COUNT(DISTINCT A)는 1000의 중복을 하나로 보며 NULL도 세지 않는다.
④ X: 틀린 표현은 첫째 값 4, 둘째 값 3, 마지막 값 4000이다. 올바르게는 3, 2, 3000이다. 이 선지는 NULL 제외 규칙과 DISTINCT의 중복 제거 규칙을 모두 잘못 적용한 경우이다.

오답 포인트

4행의 B=4000만 보고 MAX(A+B)를 4000으로 착각하기 쉽다. 그러나 집계 전에 먼저 A+B를 계산해야 하고, NULL과의 산술 결과는 NULL이 된다. 또 COUNT와 COUNT(DISTINCT)에서 NULL 제외와 중복 제거를 동시에 놓치면 4와 3으로 잘못 세기 쉽다.

다시 나오면 이렇게 판단

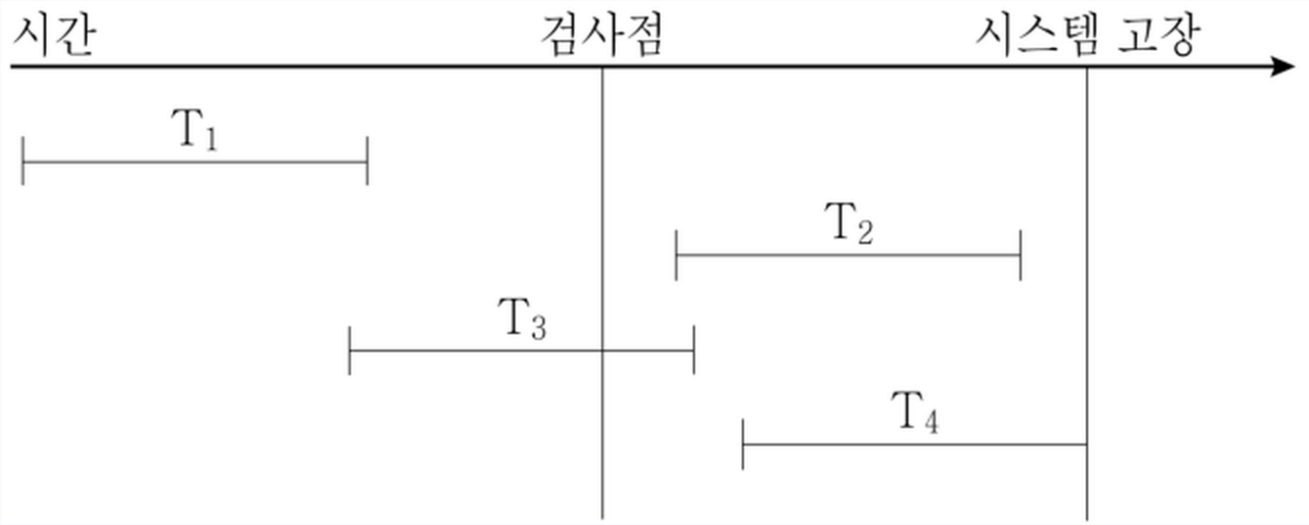
집계 함수 문제는 먼저 각 함수가 NULL을 세는지 제외하는지 확인하고, DISTINCT는 중복 제거 뒤 개수를 센다. 식이 들어간 MAX나 MIN은 열 하나를 바로 보는 것이 아니라 각 행의 식 결과를 먼저 만든 뒤 NULL이 아닌 값끼리 비교한다고 판단하면 된다.

암기 포인트

COUNT는 NULL을 세지 않고, COUNT(DISTINCT)는 중복과 NULL을 제외하며, NULL이 들어간 산술식 결과는 NULL이 된다.

NoteBase

문 14. 다음은 트랜잭션의 수행 중에 장애가 발생할 경우의 회복(Recovery) 기법을 적용하여 트랜잭션의 상태를 복구시키는 사례이다. 로그는 그대로 기록을 유지하면서, 회복 관리자가 정하는 일정한 시간 간격으로 검사점을 생성하는 Checkpoint 회복기법을 적용하는 경우에 트랜잭션들의 상태를 설명한 것으로 옳지 않은 것은?



① T_1는 회복작업을 할 필요가 없다.

② T_2는 트랜잭션 전체에 대하여 UNDO 연산을 수행해야 한다.

③ T_3는 검사점 이후에 발생된 변경 부분에 대해서만 REDO 연산을 수행해야 한다.

④ T_4는 트랜잭션 전체에 대하여 UNDO 연산을 수행해야 한다.

정답 2번

출제 포인트

이 문제는 체크포인트 회복에서 도식에 표시된 각 트랜잭션의 종료 시점과 시스템 고장 시점을 비교하여 무조치, REDO, UNDO 대상을 구분하는 문제이다.

NoteBase

선지별 해설

① O
T1은 검사점 이전에 이미 종료된 트랜잭션이므로 체크포인트 시점에 반영이 끝난 것으로 본다. 따라서 회복 작업 대상이 아니다.

② X
T2는 검사점 이후에 시작하여 시스템 고장 전에 정상 종료된 트랜잭션이므로 UNDO 대상이 아니라 REDO 대상이다. 틀린 표현은 "UNDO 연산"이며, 올바르게는 "트랜잭션 전체에 대하여 REDO 연산을 수행해야 한다"이다.

③ O
T3는 검사점 시작 당시 수행 중이었고 시스템 고장 전에는 종료되었다. 체크포인트 이전 변경 내용은 반영된 것으로 보고, 체크포인트 이후에 발생한 변경 부분만 REDO 하면 된다.

④ O
T4는 시스템 고장 시점까지 완료되지 않은 미완료 트랜잭션이므로 수행한 작업을 취소해야 한다. 따라서 트랜잭션 전체에 대하여 UNDO 연산을 수행해야 한다.

오답 포인트

체크포인트 이후에 끝난 트랜잭션을 모두 UNDO 대상으로 착각하기 쉽다. 회복에서는 고장 시점에 완료되지 않았으면 UNDO, 고장 전까지 완료되었으면 REDO라는 구분이 먼저이고, 체크포인트를 걸친 경우에만 REDO 범위를 검사점 이후로 줄여서 본다.

다시 나오면 이렇게 판단

도식 문제는 먼저 각 트랜잭션이 검사점 이전에 끝났는지, 검사점 이후 고장 전에 끝났는지, 고장 시점까지 미완료인지로 나누어 판단한다. 검사점 이전 종료는 무조치, 고장 전 완료는 REDO, 고장 시 미완료는 UNDO로 정리하면 된다. 검사점을 걸친 완료 트랜잭션은 검사점 이후 변경분만 REDO 대상이 된다.

암기 포인트

체크포인트 회복은 검사점 전 종료는 무조치, 검사점 후 완료는 REDO, 고장 시 미완료는 UNDO로 정리된다.

NoteBase

문 15. 다음은 테이블을 생성하는 SQL 구문이다. 이 테이블에 대한 연산을 설명하는 것으로 옳지 않은 것은?

```
CREATE TABLE 직원 (
        직원번호 NUMBER NOT NULL,
        직원이름 CHAR(10) UNIQUE,
        부서번호 NUMBER
                CHECK(부서번호 IN (1,2,3,4)) DEFAULT 1,
        PRIMARY KEY(직원번호),
        FOREIGN KEY(부서번호) REFERENCES
                부서(부서번호) ON DELETE CASCADE );
```

① 동명이인의 직원 이름을 추가했더니 입력되지 않았다.

② 부서가 없는 신입사원을 입력했더니 부서번호가 널값이 되었다.

③ 부서 테이블에서 특정 부서를 삭제했더니 해당 부서에 속한 직원들도 삭제되었다.

④ 직원 번호에 동일한 번호를 입력했더니 처리되지 않았다.

정답 2번

출제 포인트

이 문제는 CREATE TABLE 문에 선언된 PRIMARY KEY, UNIQUE, DEFAULT, CHECK, FOREIGN KEY ON DELETE CASCADE 제약의 의미를 구분하는 문제이다.

핵심 개념

UNIQUE는 중복값 입력을 금지하는 제약이다.
PRIMARY KEY는 행을 유일하게 식별하며 중복과 NULL을 모두 허용하지 않는 제약이다.
DEFAULT는 값을 생략했을 때 자동으로 들어가는 기본값이다.
FOREIGN KEY ... ON DELETE CASCADE는 부모 테이블의 참조 행이 삭제되면 자식 테이블의 관련 행도 함께 삭제되는 제약이다.

NoteBase

선지별 해설

① O
직원이름에 UNIQUE가 걸려 있으므로 같은 이름이 이미 저장되어 있으면 같은 이름을 다시 입력할 수 없다.

② X
틀린 부분은 '부서번호가 널값이 되었다'이다.
이 컬럼은 DEFAULT 1로 정의되어 있으므로 부서번호 값을 생략하고 입력하면 NULL이 아니라 1이 들어간다.

③ O
부서번호는 부서 테이블의 부서번호를 참조하는 외래키이고 ON DELETE CASCADE가 지정되어 있으므로 부서 테이블에서 특정 부서를 삭제하면 그 부서를 참조하던 직원 행도 함께 삭제된다.

④ O
직원번호는 PRIMARY KEY이므로 같은 번호를 중복 입력할 수 없다. 이미 존재하는 직원번호를 다시 넣으면 제약 위반으로 처리되지 않는다.

오답 포인트

DEFAULT와 NULL 허용을 같은 의미로 오해하면 ②를 고르지 못한다. 이 문항은 '입력하지 않았을 때' 적용되는 DEFAULT와 'NULL을 허용하는가'를 구분하는지를 노린 문제이다.

다시 나오면 이렇게 판단

테이블 정의 문제가 나오면 각 열에 붙은 제약을 하나씩 분리해서 본다. UNIQUE는 중복 금지, PRIMARY KEY는 중복과 NULL 금지, DEFAULT는 값 생략 시 자동 입력, ON DELETE CASCADE는 부모 삭제 시 자식도 삭제로 바로 연결하면 된다.

암기 포인트

DEFAULT는 생략값, PRIMARY KEY는 유일+NOT NULL, CASCADE는 부모 삭제 시 자식도 삭제이다.

NoteBase

문 16. 두 트랜잭션 스케줄 s_1, s_2에 대하여, s_1의 비연쇄(Cascadeless) 스케줄 여부와 s_2의 s_1에 대한 뷰 동치 여부로 옳게 짝 지어진 것은? (단, r_i, w_i, c_i는 각각 트랜잭션 T_i의 읽기, 쓰기, 완료 연산을 의미한다)

s_1: $r_1(a)\ r_1(b)\ w_1(a)\ w_1(b)\ c_1\ r_2(b)\ r_2(c)\ w_2(b)\ c_2\ r_3(c)\ r_3(b)\ c_3$
s_2: $r_1(a)\ r_1(b)\ r_2(b)\ r_2(c)\ r_3(c)\ w_1(a)\ w_1(b)\ w_2(b)\ r_3(b)\ c_1\ c_2\ c_3$

① s_1은 비연쇄 스케줄인가?: 비연쇄 스케줄, s_2는 s_1과 뷰 동치인가?: 뷰 동치
② s_1은 비연쇄 스케줄인가?: 비연쇄 스케줄, s_2는 s_1과 뷰 동치인가?: 뷰 동치 아님
③ s_1은 비연쇄 스케줄인가?: 비연쇄 스케줄 아님, s_2는 s_1과 뷰 동치인가?: 뷰 동치
④ s_1은 비연쇄 스케줄인가?: 비연쇄 스케줄 아님, s_2는 s_1과 뷰 동치인가?: 뷰 동치 아님

정답 2번

출제 포인트

이 문제는 비연쇄 스케줄과 뷰 동치의 판단 기준을 각 읽기 연산의 원천값과 커밋 시점으로 구분할 수 있는지를 묻는 문제이다.

핵심 개념

- 비연쇄 스케줄: 어떤 트랜잭션이 다른 트랜잭션이 쓴 값을 읽더라도 그 쓰기 트랜잭션이 먼저 커밋된 뒤에만 읽는 스케줄이다. 결정 차이축: 미완료 데이터 읽기 여부 / 키워드: commit 후 read
- 뷰 동치: 각 읽기 연산이 같은 초기값 또는 같은 트랜잭션의 쓴 값을 읽고, 각 데이터 항목의 최종 쓰기 트랜잭션도 같은 스케줄이다. 결정 차이축: read-from 관계와 최종 write 일치 / 키워드: same read-from, same final write

NoteBase

선지별 해설

① X
s1이 비연쇄 스케줄이라는 판단은 맞지만, "뷰 동치"가 틀린 표현이다. s1에서는 r2(b)가 w1(b) 뒤, c1 뒤에 수행되어 T1이 쓴 b를 읽는다. 반면 s2에서는 r2(b)가 w1(b)보다 먼저 수행되어 초기 b를 읽는다. 따라서 두 스케줄은 같은 값을 읽지 않으므로 뷰 동치가 아니다. 올바르게는 "비연쇄 스케줄 / 뷰 동치 아님"이다.

② O
s1은 다른 트랜잭션이 쓴 값을 읽는 연산이 모두 커밋 뒤에 나온다. r2(b)는 c1 뒤에 T1의 값을 읽고, r3(b)는 c2 뒤에 T2의 값을 읽으므로 비연쇄 스케줄이다. 또한 s1과 s2는 최종 쓰기 트랜잭션은 같아도 r2(b)의 읽기 원천이 달라서 뷰 동치가 아니다.

③ X
"비연쇄 스케줄 아님"이 틀린 표현이다. s1에서는 더티 리드가 없다. T2는 T1이 커밋한 뒤 b를 읽고, T3는 T2가 커밋한 뒤 b를 읽는다. 따라서 올바르게는 "비연쇄 스케줄"이다. 또한 s2는 s1과 뷰 동치도 아니다.

④ X
"비연쇄 스케줄 아님"이 틀린 표현이다. s2가 s1과 뷰 동치가 아니라는 판단은 맞지만, s1 자체는 비연쇄 스케줄이다. 올바르게는 "비연쇄 스케줄 / 뷰 동치 아님"이다.

오답 포인트

최종 쓰기 트랜잭션만 같으면 뷰 동치라고 오해하기 쉽다. 이 문항의 함정은 b에 대한 최종 쓰기는 둘 다 T2이지만, r2(b)가 s1에서는 T1의 값을 읽고 s2에서는 초기값을 읽는다는 점이다. 비연쇄 여부도 단순히 write가 있는지만 보지 말고 그 write를 읽는 시점이 commit 뒤인지까지 확인해야 한다.

다시 나오면 이렇게 판단

비연쇄 여부는 읽기 연산마다 바로 앞의 write 주체와 그 write 트랜잭션의 commit 위치를 먼저 대조하면 된다. 뷰 동치는 각 읽기 연산이 초기값을 읽는지, 어떤 트랜잭션의 값을 읽는지, 마지막 write가 누구인지의 세 가지를 순서대로 맞춰 보면 된다.

암기 포인트

비연쇄는 commit 후 read이고, 뷰 동치는 same read-from과 same final write이다.

NoteBase

문 17. 다음 시스템 카탈로그(System Catalog)에 대한 설명으로 옳은 것으로만 묶은 것은?

ㄱ. 시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스이기 때문에 일반 사용자는 SQL을 이용하여 내용을 검색해 볼 수 없다.
ㄴ. 개체들은 기본 테이블, 뷰, 인덱스, 데이터베이스, 패키지, 접근 권한 등이 있다.
ㄷ. 모든 데이터베이스에서 요구하는 정보는 동일하므로 데이터베이스의 종류에 관계없이 동일한 구조로 필요한 정보를 제공한다.
ㄹ. 카탈로그에 저장된 정보를 메타 데이터(Meta-Data)라고도 한다.
ㅁ. 트랜잭션 관리자를 통하여 사용자의 요구를 효율적인 형태로 변환하고 질의를 처리하는 좋은 전략을 모색할 수 있다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄹ
④ ㄴ, ㅁ

정답 3번

출제 포인트

이 문제는 시스템 카탈로그의 성격과 메타데이터의 의미, 그리고 질의 처리 구성요소의 역할을 구별하는 문제이다.

핵심 개념

시스템 카탈로그는 데이터베이스 내부 객체에 대한 설명 정보를 저장하는 시스템 데이터베이스이다.
메타데이터는 데이터 자체가 아니라 테이블, 뷰, 인덱스, 권한 같은 데이터에 대한 데이터이다.
질의 최적화와 질의 처리 전략 선택은 질의 처리기와 옵티마이저의 역할이고, 트랜잭션 관리자는 동시성 제어와 회복 같은 트랜잭션 제어를 담당한다.

NoteBase

선지별 해설

① X
- ①은 ㄱ, ㄴ의 조합인데 ㄴ은 맞지만 ㄱ이 틀리므로 오답이다.
- ㄱ에서 틀린 부분은 "일반 사용자는 SQL을 이용하여 내용을 검색해 볼 수 없다"는 표현이다.
- 시스템 카탈로그는 권한이 허용된 범위에서 SQL로 조회할 수 있으므로 올바르게는 "시스템 카탈로그는 SQL을 이용해 조회할 수 있다"이다.

② X
- ②는 ㄱ, ㄷ의 조합인데 ㄱ과 ㄷ이 모두 틀리므로 오답이다.
- ㄱ의 오표현은 "검색해 볼 수 없다"이고, 올바르게는 "SQL로 조회할 수 있다"이다.
- ㄷ의 틀린 부분은 "모든 데이터베이스에서 요구하는 정보는 동일하므로"와 "동일한 구조로"이다.
- 시스템 카탈로그가 담는 세부 항목과 구조는 DBMS마다 다르므로 올바르게는 "데이터베이스 종류에 따라 카탈로그 구조와 제공 정보가 달라질 수 있다"이다.

③ O
- ③은 ㄴ, ㄹ의 조합이고 ㄴ과 ㄹ이 모두 옳으므로 정답이다.
- ㄴ은 시스템 카탈로그의 관리 대상 예시를 묻는 내용으로, 기본 테이블, 뷰, 인덱스, 권한 등 데이터베이스 객체와 관련 정보가 포함된다는 점에서 맞다.
- ㄹ은 카탈로그에 저장된 정보가 데이터에 대한 설명 정보라는 뜻이므로 메타데이터라고 부른다는 점에서 맞다.

④ X
- ④는 ㄴ, ㅁ의 조합인데 ㄴ은 맞지만 ㅁ이 틀리므로 오답이다.
- ㅁ에서 틀린 부분은 "트랜잭션 관리자를 통하여 사용자의 요구를 효율적인 형태로 변환하고 질의를 처리하는 좋은 전략을 모색"한다는 표현이다.
- 질의 변환과 최적화 전략 탐색은 질의 처리기와 옵티마이저의 역할이고, 트랜잭션 관리자는 트랜잭션의 원자성, 일관성, 격리성, 지속성과 관련된 제어를 담당한다.

오답 포인트

ㄱ은 시스템 카탈로그가 시스템용 정보라는 이유로 조회 자체가 불가능하다고 착각하면 틀리기 쉽다. ㅁ은 질의 최적화 기능과 트랜잭션 관리 기능을 섞어 놓은 함정이다. ㄷ은 "모든 DBMS에서 동일하다"라는 절대 표현이 나오면 먼저 의심해야 한다.

다시 나오면 이렇게 판단

시스템 카탈로그 문제는 먼저 "무엇을 저장하는가", "사용자가 조회할 수 있는가", "DBMS마다 구조가 같은가", "어느 관리자가 담당하는가"를 나누어 본다. 객체 설명 정보이면 메타데이터, 질의 최적화이면 질의 처리기, 동시성 제어와 회복이면 트랜잭션 관리자로 판단하면 된다.

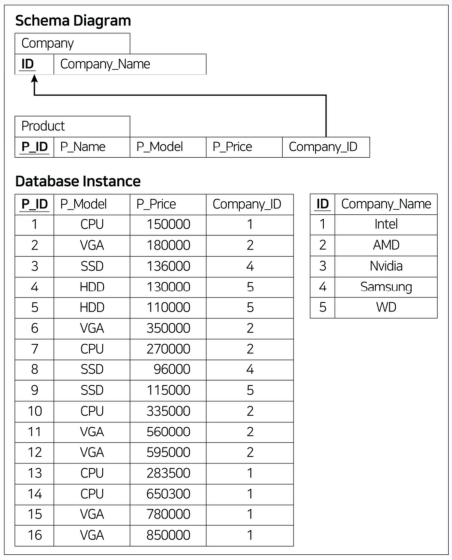
NoteBase

암기 포인트

카탈로그는 메타데이터, 질의 최적화는 옵티마이저, 트랜잭션 관리자는 동시성 제어와 회복이다.

NoteBase

문 18. 다음은 제품 데이터베이스의 스키마 다이어그램과 인스턴스를 나타낸 것이다. 데이터베이스를 구성하는 이 두 테이블을 대상으로 아래 결과를 출력하고자 한다. 이를 위한 SQL 질의문은?



Database Instance

P_ID	P_Model	P_Price	Company_ID
1	CPU	150000	1
2	VGA	180000	2
3	SSD	136000	4
4	HDD	130000	5
5	HDD	110000	5
6	VGA	350000	2
7	CPU	270000	2
8	SSD	96000	4
9	SSD	115000	5
10	CPU	335000	2
11	VGA	560000	2
12	VGA	595000	2
13	CPU	283500	1
14	CPU	650300	1
15	VGA	780000	1
16	VGA	850000	1

ID	Company_Name
1	Intel
2	AMD
3	Nvidia
4	Samsung
5	WD

SQL 실행 결과

P_ID	P_Model	P_Price	Company_ID
6	VGA	350000	2
11	VGA	560000	2
12	VGA	595000	2
15	VGA	780000	1
16	VGA	850000	1

①
```
SELECT *
FROM Product Pro
WHERE Pro.P_Model = 'VGA' AND Pro.P_Price IN
            (SELECT P.P_Price
             FROM Product P LEFT JOIN Company C
                ON (C.ID = P.Company_ID)
             WHERE C.Company_Name = 'AMD');
```

②
```
SELECT *
FROM Product Pro
WHERE Pro.P_Model = 'VGA' AND Pro.P_Price NOT IN
            (SELECT P.P_Price
             FROM Product P LEFT JOIN Company C
                ON (C.ID = P.Company_ID)
             WHERE C.Company_Name = 'AMD');
```

③
```
SELECT *
FROM Product Pro
WHERE Pro.P_Model = 'VGA' AND Pro.P_Price > ANY
            (SELECT P.P_Price
             FROM Product P LEFT JOIN Company C
                ON (C.ID = P.Company_ID)
             WHERE C.Company_Name = 'AMD');
```

④
```
SELECT *
FROM Product Pro
WHERE Pro.P_Model = 'VGA' AND Pro.P_Price > ALL
            (SELECT P.P_Price
             FROM Product P LEFT JOIN Company C
                ON (C.ID = P.Company_ID)
             WHERE C.Company_Name = 'AMD');
```

정답 3번

NoteBase

출제 포인트

이 문제는 Company_Name이 AMD인 행들의 가격 집합을 구한 뒤, `IN`, `NOT IN`, `> ANY`, `> ALL`이 각각 어떤 VGA 행을 남기는지 데이터로 판정하는 문제이다.

자료 읽는 순서

1. Company 테이블에서 AMD의 ID가 2임을 확인한다.
2. Product 테이블에서 Company_ID가 2인 행의 P_Price 집합이 180000, 270000, 335000, 350000, 560000, 595000임을 읽는다.
3. 전체 VGA 행의 가격이 180000, 350000, 560000, 595000, 780000, 850000임을 확인한다.
4. 각 비교 연산이 이 VGA 가격 집합에서 어떤 행을 남기는지 대조한다.

핵심 근거

서브쿼리는 AMD 회사의 제품 가격 집합을 반환한다.
`> ANY`는 그 집합의 값 중 하나보다만 크면 되므로, 결과적으로 최솟값 180000보다 큰 값이면 조건을 만족한다.
따라서 VGA 중 180000만 제외되고 350000, 560000, 595000, 780000, 850000이 남아 SQL 실행 결과와 일치한다.

NoteBase

선지별 해설

① X
- `IN`은 VGA 가격이 AMD 가격 집합에 포함되는 행만 고른다.
- 근거 값은 180000, 350000, 560000, 595000이므로 결과는 P_ID 2, 6, 11, 12가 된다.
- 제시된 실행 결과에는 780000, 850000이 포함되고 180000은 제외되므로 일치하지 않는다.
- 틀린 표현은 `IN`이다. 올바르게는 AMD 가격 집합의 어떤 값보다 큰지를 비교하는 `> ANY`여야 한다.

② X
- `NOT IN`은 VGA 가격이 AMD 가격 집합에 없는 행만 고른다.
- VGA 중 AMD 집합에 없는 값은 780000, 850000뿐이므로 결과는 P_ID 15, 16만 남는다.
- 제시된 실행 결과의 350000, 560000, 595000이 빠지므로 일치하지 않는다.
- 틀린 표현은 `NOT IN`이다. 올바르게는 AMD 가격 집합에 포함 여부가 아니라 그 집합의 어떤 값보다 큰지를 묻는 `> ANY`이다.

③ O
- AMD 가격 집합의 최솟값은 180000이다.
- `Pro.P_Price > ANY (집합)`은 집합의 값 중 하나보다만 크면 되므로, 여기서는 180000보다 큰 가격이면 성립한다.
- 전체 VGA 가격 180000, 350000, 560000, 595000, 780000, 850000 중 180000만 제외되어 P_ID 6, 11, 12, 15, 16이 남는다.
- 이는 제시된 SQL 실행 결과와 정확히 일치한다.

④ X
- `> ALL`은 AMD 가격 집합의 모든 값보다 커야 하므로, 사실상 최댓값 595000보다 커야 한다.
- 그러면 VGA 중 780000, 850000만 남아 P_ID 15, 16만 출력된다.
- 제시된 실행 결과의 350000, 560000, 595000은 조건을 만족하지 못하므로 일치하지 않는다.
- 틀린 표현은 `> ALL`이다. 올바르게는 최댓값 초과가 아니라 집합의 어떤 한 값 초과를 뜻하는 `> ANY`이다.

오답 포인트

`ANY`와 `ALL`을 막연히 대소 비교로만 외우면 틀리기 쉽다. 이 문항은 AMD 회사의 VGA만 뽑는 것이 아니라 AMD 회사의 가격 집합을 기준으로 전체 VGA를 다시 거르는 문제인데, 이 기준 집합과 최종 검색 대상 테이블을 같은 집합으로 착각하면 ①이나 ④로 오인하기 쉽다.

다시 나오면 이렇게 판단

서브쿼리가 반환하는 값을 먼저 실제 집합으로 적어 두고, `IN`은 포함 여부, `NOT IN`은 미포함 여부, `> ANY`는 집합 최솟값보다 큰지, `> ALL`은 집합 최댓값보다 큰지로 바꾸어 판단하면 된다.

NoteBase

암기 포인트

`> ANY`는 최솟값 초과, `> ALL`은 최댓값 초과로 정리된다.

NoteBase

문 19. 다음 데이터베이스 시스템의 질의최적화에 대한 설명으로 옳지 않은 것으로만 묶은 것은?

ㄱ. DBMS는 SQL 질의문을 내부 표현(internal representation) 형태로 변환하여 질의최적화를 한다.
ㄴ. DBMS는 질의최적화 과정에서 데이터사전에 저장된 정보를 활용한다.
ㄷ. 규칙기반 최적화 기법(rule-based optimization)이 질의 실행계획(query execution plan)을 생성하는 데 비용이 거의 소요되지 않기 때문에 최근에 많이 활용되고 있다.
ㄹ. 비용기반 최적화 기법(cost-based optimization)은 생성 가능한 모든 후보 질의실행계획 중에서 비용이 가장 적게 드는 질의실행계획을 생성한다.
ㅁ. 조인 방법(join method) 중에서 해시 조인방법은 비동등 조인 질의(non-equi join query)에 적용하지 않는다.

① ㄱ, ㄴ
② ㄴ, ㄷ
③ ㄷ, ㄹ
④ ㄹ, ㅁ

정답 3번

출제 포인트

이 문제는 규칙기반 최적화와 비용기반 최적화의 차이, 그리고 해시 조인의 적용 범위를 구별하는지를 묻는다.

핵심 개념

SQL 질의는 내부 표현 형태로 바뀐 뒤 최적화가 수행된다. 규칙기반 최적화는 미리 정한 휴리스틱 규칙으로 실행계획을 고르는 방식이고, 비용기반 최적화는 데이터 사전의 통계 정보를 이용해 후보 실행계획의 비용을 추정하는 방식이다. 해시 조인은 등가 조인에 적합하고 비동등 조인에는 일반적으로 적용되지 않는다.

NoteBase

선지별 해설

1번: X. ㄱ과 ㄴ이 모두 옳다. ㄱ은 SQL 질의문을 내부 표현으로 변환한 뒤 최적화를 수행한다는 설명이므로 맞고, ㄴ은 질의최적화에서 데이터 사전과 카탈로그 통계 정보를 활용한다는 설명이므로 맞다. 따라서 1번은 '옳지 않은 것만 묶은 것'이 아니다. 올바르게는 ㄱ, ㄴ은 옳은 설명이다.
2번: X. ㄴ은 옳지만 ㄷ은 틀리므로 옳지 않은 것만 묶은 선택지가 아니다. ㄷ의 틀린 표현은 '최근에 많이 활용되고 있다'이다. 규칙기반 최적화는 최적화 비용은 작지만 최근의 주류 최적화 방식은 비용기반 최적화이다.
3번: O. ㄷ과 ㄹ이 옳지 않은 설명이므로 정답이다. ㄷ은 규칙기반 최적화가 최근에 많이 활용된다고 한 부분이 틀렸고, ㄹ은 '생성 가능한 모든 후보 질의실행계획 중에서'라는 절대적 표현이 틀리다. 비용기반 최적화는 통상 통계 정보를 바탕으로 탐색한 후보들 중 추정 비용이 가장 작은 계획을 선택한다.
4번: X. ㄹ은 틀리지만 ㅁ은 옳다. ㅁ은 해시 조인이 동등 조건을 전제로 하는 조인 방식이어서 비동등 조인에는 일반적으로 적용되지 않는다는 설명이므로 맞다. 따라서 4번은 옳지 않은 것만 묶은 선택지가 아니다. 올바르게는 ㅁ은 옳은 설명이다.

오답 포인트

ㄷ의 '최근에 많이 활용'과 ㄹ의 '모든 후보' 같은 절대적 표현이 함정이다. 규칙기반 최적화는 오래된 휴리스틱 방식이고, 비용기반 최적화도 실제로는 탐색 가능한 후보 공간 안에서 비용을 비교한다는 점을 놓치면 2번이나 4번으로 흔들린다.

다시 나오면 이렇게 판단

이런 조합형 문항은 먼저 ㄱ~ㅁ을 각각 O/X로 독립 판정한 뒤 보기 1~4를 맞춰 본다. 최적화 문항에서는 '최근', '모든'처럼 범위를 과장하는 표현이 붙으면 먼저 의심하고, 조인 방식은 등가 조인인지 비동등 조인인지부터 구분한다.

암기 포인트

규칙기반은 휴리스틱, 비용기반은 통계와 비용, 해시 조인은 등가 조인이라는 축으로 기억하면 된다.

NoteBase

문 20. 트랜잭션 고립 수준 중 REPEATABLE READ 모드에서 발생할 수 있는 문제를 모두 고르면?

ㄱ. 유령데이터 읽기
ㄴ. 반복불가능 읽기
ㄷ. 오손읽기

① ㄱ
② ㄷ
③ ㄱ, ㄴ
④ ㄴ, ㄷ

정답 1번

출제 포인트

이 문제는 트랜잭션 고립 수준인 REPEATABLE READ에서 허용되는 이상 현상과 차단되는 이상 현상을 구분하는 문제이다.

핵심 개념

유령데이터 읽기는 같은 조건으로 다시 조회했을 때 다른 트랜잭션의 삽입 또는 삭제 때문에 결과 집합의 행 수가 달라지는 현상이다.
반복불가능 읽기는 같은 행을 두 번 읽었을 때 중간 갱신 때문에 값이 달라지는 현상이다.
오손읽기는 아직 커밋되지 않은 데이터를 읽는 현상이다.
REPEATABLE READ는 오손읽기와 반복불가능 읽기는 막지만, 조건 검색 결과의 행 집합 변화인 유령데이터 읽기는 허용할 수 있다.

NoteBase

선지별 해설

① O
- ①은 ㄱ만 고른 선택지이다.
- REPEATABLE READ에서는 같은 행의 재조회 값은 고정되지만 범위 조건으로 조회한 결과 집합은 다른 트랜잭션의 삽입·삭제로 달라질 수 있으므로 유령데이터 읽기는 발생할 수 있다.

② X
- ②는 ㄷ만 고른 선택지이다.
- 틀린 부분은 오손읽기가 REPEATABLE READ에서 발생한다고 본 점이다.
- 오손읽기는 커밋 전 데이터를 읽는 현상인데, REPEATABLE READ에서는 허용되지 않는다.
- 올바르게는 발생 가능한 것은 ㄱ만이므로 ①이 맞다.

③ X
- ③은 ㄱ, ㄴ을 고른 선택지이다.
- 틀린 부분은 ㄴ인 반복불가능 읽기를 포함한 점이다.
- REPEATABLE READ는 같은 행을 다시 읽어도 값이 바뀌지 않도록 보장하므로 반복불가능 읽기는 발생하지 않는다.
- 올바르게는 ㄴ을 제외하고 ㄱ만 남겨야 한다.

④ X
- ④는 ㄴ, ㄷ을 고른 선택지이다.
- 틀린 부분은 반복불가능 읽기와 오손읽기가 모두 발생한다고 본 점이다.
- REPEATABLE READ에서는 ㄴ과 ㄷ이 모두 차단되고, 범위 검색에서의 유령데이터 읽기만 남는다.
- 올바르게는 ㄴ, ㄷ이 아니라 ㄱ만 선택해야 한다.

오답 포인트

반복불가능 읽기와 유령데이터 읽기를 둘 다 '다시 읽었더니 달라지는 현상'으로만 외우면 ③으로 흔들리기 쉽다. 같은 행의 값이 바뀌는 것은 반복불가능 읽기이고, 조건 검색 결과의 행 집합이 바뀌는 것은 유령데이터 읽기라는 축으로 나누어야 한다.

다시 나오면 이렇게 판단

고립 수준 문제는 먼저 오손읽기 차단 여부를 보고, 다음으로 같은 행 값 변화 차단 여부를 본다. 그 뒤 범위 조회 결과 집합 변화까지 막는지를 따지면 READ COMMITTED, REPEATABLE READ, SERIALIZABLE을 구분할 수 있다.

NoteBase

암기 포인트

REPEATABLE READ는 행 값 재읽기는 고정하고, 팬텀은 남을 수 있다.

NoteBase