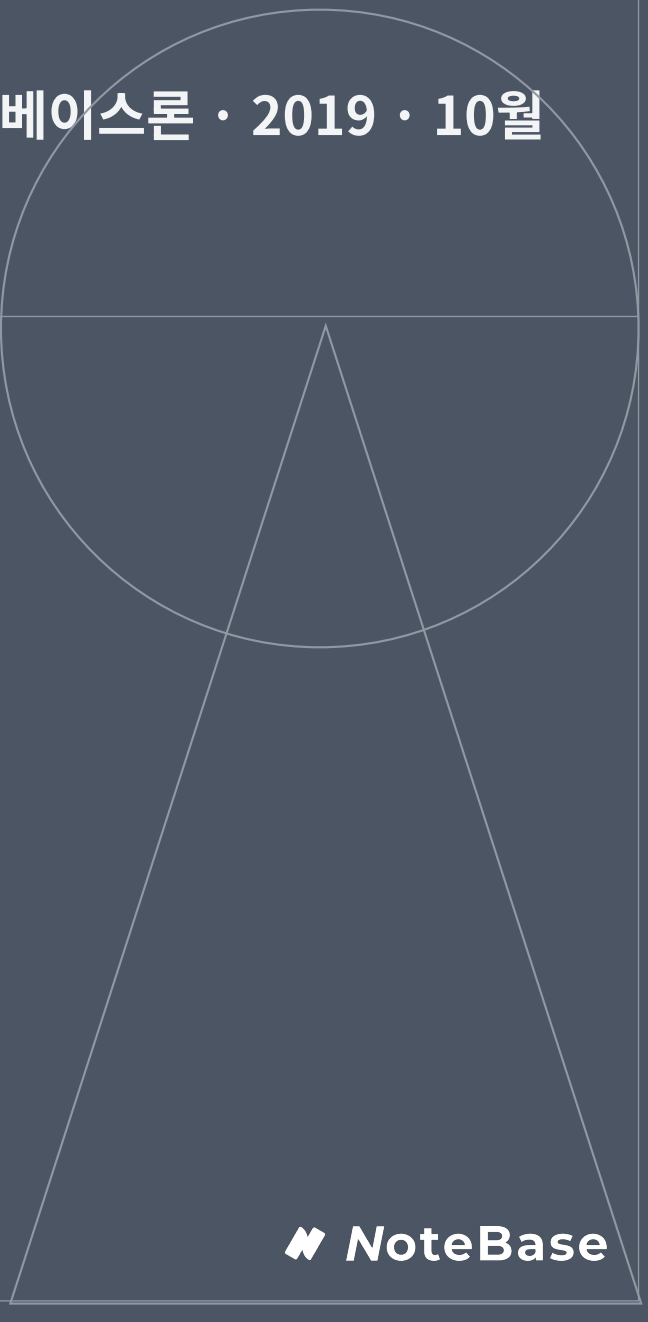
기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월



NoteBase

기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월

공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월

문 1. 관계형 모델에서 일반적인 릴레이션의 특성으로 가장 옳지 않은 것은?

① 릴레이션의 속성이 다른 속성의 특성을 상속할 수 있다.

② 하나의 릴레이션에서 속성 사이의 순서는 무의미하다.

③ 하나의 릴레이션에서는 동일한 튜플(tuple)이 존재할 수 없다.

④ 하나의 릴레이션에서 튜플 사이의 순서는 무의미하다.

정답 1번

출제 포인트

이 문제는 관계형 모델에서 릴레이션이 가지는 집합적 성질과 속성의 의미를 구분하는 문제이다.

핵심 개념

릴레이션은 같은 구조를 가진 튜플들의 집합이다.
속성은 릴레이션을 구성하는 열로서 각 속성은 자신의 도메인과 이름으로 구분되며, 다른 속성의 성질을 상속하는 개념이 아니다.
튜플은 릴레이션의 행이며, 릴레이션이 집합이므로 튜플의 중복과 순서는 인정되지 않는다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월

선지별 해설

① X
릴레이션의 속성은 서로 독립적인 의미를 가지는 열이다. 관계형 모델의 일반적 성질에는 속성이 다른 속성의 특성을 상속한다는 개념이 없다.
틀린 부분은 "다른 속성의 특성을 상속"이다.
올바르게는 "각 속성은 고유한 이름과 도메인을 가지며 다른 속성의 특성을 상속하지 않는다"이다.

② O
하나의 릴레이션에서 속성의 순서는 의미가 없다. 열의 위치가 바뀌어도 속성 이름과 도메인이 같으면 같은 릴레이션으로 본다.

③ O
릴레이션은 집합이므로 동일한 튜플이 중복되어 존재할 수 없다. 같은 행이 반복되면 집합의 성질에 어긋난다.

④ O
하나의 릴레이션에서 튜플 사이의 순서는 의미가 없다. 행이 저장되거나 출력되는 물리적 순서가 달라도 관계의 내용 자체는 변하지 않는다.

오답 포인트

①은 객체지향의 상속 개념을 관계형 모델의 속성 개념과 섞으면 흔들리기 쉽다. 이 문항은 순서 무의미성과 중복 불가라는 집합 성질은 맞게 두고, 상속이라는 낯선 표현만 틀리게 넣어 구분력을 묻는다.

다시 나오면 이렇게 판단

릴레이션 관련 선지를 보면 먼저 "집합인가"를 확인한다. 집합이면 튜플 중복이 없고 튜플 순서가 없다. 이어서 속성은 열의 이름과 도메인으로 식별되는 독립 요소이므로 상속, 우선순위 같은 표현이 나오면 관계형 모델의 일반 성질이 아닌지 의심하면 된다.

암기 포인트

릴레이션은 집합이라 순서와 중복이 없고, 속성은 상속하지 않는다.

NoteBase

문 2. 릴레이션 A에는 5개의 튜플이, 릴레이션 B에는 6개의 튜플이 있을 때, 두 릴레이션의 교차곱(cartesian product) 연산을 수행한 결과 릴레이션의 튜플 개수는?

① 5

② 6

③ 30

④ 5^6

정답 3번

출제 포인트

이 문제는 교차곱의 결과 튜플 수를 구하는 기본 계산 문제이다.

조건 정리

주어진 값: 릴레이션 A의 튜플 수는 5개, 릴레이션 B의 튜플 수는 6개이다.
구해야 할 값: 두 릴레이션의 교차곱 결과 튜플 수이다.
주의할 조건: 교차곱은 A의 각 튜플이 B의 모든 튜플과 한 번씩 결합된다.

사용할 공식

교차곱 결과 튜플 수는 $|A| \times |B|$이다.
이 공식을 쓰는 이유: 교차곱은 두 릴레이션의 가능한 모든 튜플 쌍을 만드는 연산이므로 한쪽의 튜플 수와 다른 쪽의 튜플 수를 곱해 전체 경우의 수를 구한다.

계산 과정

1. 릴레이션 A의 각 튜플 5개가 릴레이션 B의 6개 튜플과 각각 결합된다.
2. 전체 튜플 수는 5×6으로 계산된다.
3. 계산 결과는 30이다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월

선지 판단

① X
5는 릴레이션 A 자체의 튜플 수일 뿐 교차곱 결과가 아니다. 교차곱은 상대 릴레이션의 모든 튜플과 결합해야 하므로 틀린 표현은 '결과가 5개'이고, 올바르게는 '결과는 5 × 6 = 30개'이다.

② X
6은 릴레이션 B 자체의 튜플 수일 뿐 교차곱 결과가 아니다. 틀린 표현은 '결과가 6개'이고, 올바르게는 'A의 5개와 B의 6개를 모두 조합하므로 30개'이다.

③ O
교차곱의 결과 튜플 수는 두 릴레이션의 튜플 수를 곱한 값이므로 5 × 6 = 30이 맞다.

④ X
5^6은 거듭제곱 계산으로 교차곱의 정의와 맞지 않는다. 틀린 표현은 '5의 6제곱'이고, 올바르게는 'A의 튜플 수와 B의 튜플 수의 곱인 5 × 6'이다.

오답 포인트

교차곱을 한 릴레이션의 튜플 수만 세는 연산으로 착각하거나, 모든 조합이라는 뜻을 거듭제곱으로 오해하면 ①, ②, ④ 같은 오답에 걸리기 쉽다.

다시 나오면 이렇게 판단

두 집합이나 두 릴레이션의 모든 조합을 만드는 연산이 보이면 먼저 '한쪽 개수 × 다른 쪽 개수'인지 확인한 뒤 결과를 계산한다.

암기 포인트

교차곱은 모든 쌍 생성이므로 결과 튜플 수는 항상 '첫 릴레이션 개수 × 둘째 릴레이션 개수'이다.

NoteBase

문 3. <보기 1>은 구매내역을 담고 있는 buy 테이블이다. 이 테이블에 <보기 2>의 SQL을 수행할 때 출력되는 결과로 가장 옳은 것은?

<보기 1>

userid	prodName	price	amount
KHD	운동화	50	2
KHD	노트북	1000	2
KYM	모니터	200	5
PSH	책	50	20
KYM	바지	100	5
PSH	책	50	10
LHJ	운동화	50	2
LHJ	노트북	1000	3

<보기 2>

```
SELECT userid AS [사용자], SUM(price*amount) AS [총 구매액]
FROM buy
GROUP BY userid
HAVING SUM(price*amount) > 2000
```

① 사용자: KHD, 총 구매액: 2100
사용자: PSH, 총 구매액: 3000

② 사용자: LHJ, 총 구매액: 2500
사용자: PSH, 총 구매액: 3000

③ 사용자: KHD, 총 구매액: 2100
사용자: LHJ, 총 구매액: 3100

④ 사용자: LHJ, 총 구매액: 3100

정답 3번

출제 포인트

이 문제는 buy 테이블에서 userid별 총구매액을 집계한 뒤 HAVING 조건으로 2000 초과 사용자만 남기는지를 묻는 문제이다.

NoteBase

자료 읽는 순서

1. 각 행의 구매액을 price*amount로 계산한다.
2. userid별로 구매액을 합산한다.
3. HAVING SUM(price*amount) > 2000 조건을 만족하는 사용자만 남긴다.

핵심 근거

KHD는 $50*2+1000*2=2100$이므로 출력된다.
KYM은 $200*5+100*5=1500$이므로 출력되지 않는다.
PSH는 $50*20+50*10=1500$이므로 출력되지 않는다.
LHJ는 $50*2+1000*3=3100$이므로 출력된다.

선지별 해설

① X: KHD 2100은 맞지만 PSH 3000이 틀리다. PSH의 실제 총구매액은 $50*20+50*10=1500$이다. 또한 결과에는 LHJ 3100이 포함되어야 한다.
② X: LHJ 2500과 PSH 3000이 모두 틀리다. LHJ의 실제 총구매액은 $50*2+1000*3=3100$이고, PSH의 실제 총구매액은 1500이다. 올바른 결과는 KHD 2100과 LHJ 3100이다.
③ O: KHD 2100과 LHJ 3100만 남긴 결과이므로 맞다. 두 사용자만 HAVING SUM(price*amount) > 2000 조건을 만족한다.
④ X: LHJ 3100 자체는 맞지만 KHD 2100이 누락되어 전체 결과가 틀리다. 올바르게는 KHD 2100과 LHJ 3100 두 행이 함께 출력된다.

오답 포인트

행별 구매액 계산과 사용자별 합계를 혼동하면 오답이 나온다. 특히 PSH는 같은 사용자 행이 두 개이므로 20개와 10개를 각각 계산한 뒤 합쳐야 하며, HAVING은 집계 후 조건이라는 점을 놓치기 쉽다.

다시 나오면 이렇게 판단

집계 SQL 문제는 SELECT의 집계식과 GROUP BY 기준열을 먼저 확인한 뒤, 각 행 값을 계산하고 그룹별 합계를 만든 다음 HAVING 조건으로 최종 출력 대상을 걸러내면 된다.

암기 포인트

GROUP BY는 묶는 기준이고 HAVING은 집계 결과를 거르는 조건이다.

NoteBase

문 4. 트리거(trigger)에 대한 설명으로 가장 옳지 않은 것은?

① 트리거의 특징은 데이터 변경문이 실행될 때 자동으로 실행되는 것이다.

② 트리거는 저장 프로시저(stored procedure)와는 달리 편리하며 시스템 성능을 항상 향상시킨다.

③ 트리거는 저장 프로시저와 달리 매개변수를 지정할 수 없다.

④ 트리거는 직접 실행시킬 수는 없고 테이블이나 뷰에 이벤트가 발생한 경우에만 실행되며 반환값을 사용할 수 없다.

정답 2번

출제 포인트

이 문제는 트리거와 저장 프로시저의 실행 방식, 매개변수 사용 가능 여부, 성능 특성을 구별하는지를 묻는 문제이다.

핵심 개념

트리거는 테이블이나 뷰에서 INSERT, UPDATE, DELETE 같은 이벤트가 발생할 때 자동으로 실행되는 데이터베이스 객체이다.
저장 프로시저는 필요할 때 명시적으로 호출하여 실행하는 프로그램 단위이며, 일반적으로 매개변수를 받을 수 있다.
트리거는 자동 실행 편의는 있으나 불필요하게 많이 사용하면 오히려 성능 저하와 유지보수 복잡성을 만들 수 있다.

선지별 해설

① O: 트리거의 핵심 특징은 데이터 변경문 실행 시 관련 이벤트에 반응하여 자동 실행된다는 점이므로 옳은 설명이다.
② X: 틀린 부분은 "시스템 성능을 항상 향상시킨다"는 표현이다. 트리거는 자동 처리에는 편리할 수 있지만 추가 작업이 함께 실행되므로 오히려 성능을 저하시킬 수도 있다. 올바르게는 "트리거는 자동 실행되어 편리할 수 있으나 시스템 성능을 항상 향상시키지는 않는다"이다.
③ O: 트리거는 저장 프로시저처럼 호출 시점에 매개변수를 넘겨 실행하는 방식이 아니므로 일반적으로 매개변수를 지정할 수 없다.
④ O: 트리거는 직접 호출하는 객체가 아니라 테이블이나 뷰의 이벤트 발생 시 실행되며, 함수처럼 반환값을 사용하는 객체도 아니므로 옳은 설명이다.

오답 포인트

트리거가 자동으로 실행된다는 이유만으로 성능에도 항상 유리하다고 일반화하면 ②에 걸리게 된다. 자동 실행과 성능 향상은 같은 개념이 아니며, 트리거는 숨은 부하를 만들 수 있다는 점이 함정이다.

NoteBase

다시 나오면 이렇게 판단

트리거 문제는 먼저 자동 실행 여부, 직접 호출 가능 여부, 매개변수 전달 가능 여부, 성능이 항상 좋아지는지 여부를 나누어 판단하면 된다. 특히 "항상" 같은 절대 표현이 붙으면 오답 가능성이 크다.

암기 포인트

트리거는 이벤트가 부르면 자동 실행, 프로시저는 사용자가 호출해 실행이다. 트리거는 자동화 도구이지 성능 보장 도구는 아니다.

NoteBase

문 5. 3단계 데이터베이스 구조와 데이터 독립성에 대한 설명으로 가장 옳지 않은 것은?

① 개념 스키마는 모든 응용시스템들이나 사용자들이 필요로 하는 조직 전체의 데이터베이스를 기술한 것이며 개체, 관계, 제약조건을 포함한다.

② 데이터 독립성은 스키마 사이의 사상에 의해 제공되며 사상 정보는 데이터 사전에 저장된다.

③ 데이터베이스 하나에는 개념 스키마가 하나만 존재하고, 내부 스키마는 여러 개 존재한다.

④ 물리적 데이터 독립성에 의해 내부 스키마가 변경되더라도 개념 스키마와 외부 스키마는 영향을 받지 않는다.

정답 3번

출제 포인트

이 문제는 3단계 데이터베이스 구조에서 외부 스키마, 개념 스키마, 내부 스키마의 역할과 개수, 그리고 데이터 독립성의 방향을 구별하는지를 묻는다.

핵심 개념

외부 스키마: 사용자나 응용 프로그램별 관점에서 본 데이터베이스 부분 구조이다. 한 데이터베이스에 여러 개가 존재할 수 있다.
개념 스키마: 조직 전체의 데이터베이스를 논리적으로 통합해 기술한 전체 구조이다. 개체, 관계, 제약조건을 포함하며 보통 하나만 존재한다.
내부 스키마: 데이터의 물리적 저장 구조와 접근 방법을 기술한 구조이다. 한 데이터베이스에 하나만 존재한다.
물리적 데이터 독립성: 내부 스키마가 바뀌어도 개념 스키마와 외부 스키마가 영향을 받지 않는 성질이다.

선지별 해설

1번: O. 개념 스키마는 조직 전체의 데이터베이스를 통합적으로 기술하는 논리적 스키마이므로 개체, 관계, 제약조건을 포함한다는 설명이 맞다.
2번: O. 데이터 독립성은 외부-개념, 개념-내부 스키마 사이의 사상으로 구현되며, 이런 스키마 정보와 사상 정보는 데이터 사전 또는 시스템 카탈로그에 저장된다.
3번: X. 틀린 부분은 "내부 스키마는 여러 개 존재한다"이다. 실제로는 데이터베이스 하나에 개념 스키마는 하나, 내부 스키마도 하나이며, 여러 개 존재할 수 있는 것은 외부 스키마이다.
4번: O. 물리적 데이터 독립성은 저장 구조나 접근 경로 같은 내부 스키마의 변화가 상위 수준인 개념 스키마와 외부 스키마에 영향을 주지 않게 하는 성질이므로 옳다.

오답 포인트

외부 스키마와 내부 스키마의 개수를 뒤바꾸면 쉽게 틀린다. 또한 물리적 데이터 독립성과 논리적 데이터 독립성의 방향을 섞으면 4번 같은 선지 판단이 흔들린다.

NoteBase

다시 나오면 이렇게 판단

사용자별 화면이나 관점이면 외부 스키마, 조직 전체의 논리 구조면 개념 스키마, 저장 방법이면 내부 스키마로 먼저 분류한다. 그다음 여러 개인 것은 외부 스키마뿐인지 확인하고, 물리적 독립성은 내부 변화가 상위 스키마에 번지지 않는 관계인지 판단한다.

암기 포인트

외부는 여러 개, 개념과 내부는 하나이며, 물리적 데이터 독립성은 내부 변경이 개념·외부에 영향을 주지 않는 성질이다.

NoteBase

문 6. SCORE(이름(name), 번호(bunho), 국어(korean)) 테이블에서 이름(name)이 “김”으로 시작하는 학생 수를 구하기 위한 SQL문으로 옳은 것은?

① SELECT COUNT(*)
FROM SCORE
WHERE name = “김_”;

② SELECT COUNT(*)
FROM SCORE
WHERE name LIKE “김%”;

③ SELECT COUNT(*)
FROM SCORE
WHERE name LIKE “김_”;

④ SELECT COUNT(*)
FROM SCORE
WHERE name = “김%”;

정답 2번

출제 포인트

이 문제는 문자열 비교 연산자 `=`와 패턴 매칭 연산자 `LIKE`, 그리고 와일드카드 `%`와 `_`의 의미 차이를 묻는 문제이다.

핵심 개념

`=`는 문자열이 완전히 같은지를 비교하는 연산자이다.
`LIKE`는 문자열 패턴이 일치하는지를 비교하는 연산자이다.
`%`는 글자 수와 상관없이 0개 이상의 임의 문자열을 뜻하고, `_`는 정확히 1개의 임의 문자를 뜻한다.

선지별 해설

1번: X. `name = "김_"`는 이름이 문자 그대로 `김_`인 행만 찾는다. `=`는 와일드카드를 해석하지 않으므로 `김`으로 시작하는 모든 이름을 찾을 수 없다. 틀린 표현은 `=`이다. 올바르게는 `WHERE name LIKE "김%";`이다.
2번: O. `name LIKE "김%"`는 첫 글자가 `김`이고 그 뒤에 0개 이상의 문자가 오는 이름을 찾는다. 따라서 `김`, `김철수`, `김민지`처럼 `김`으로 시작하는 학생 수를 구하는 조건으로 맞다.
3번: X. `name LIKE "김_"`는 `김` 뒤에 정확히 한 글자만 오는 이름만 찾는다. 따라서 `김철`은 찾지만 `김철수`는 찾지 못한다. 틀린 표현은 `_`이다. 올바르게는 `WHERE name LIKE "김%";`이다.
4번: X. `name = "김%"`는 `%`를 패턴이 아니라 일반 문자로 취급하여 이름이 문자 그대로 `김%`인 경우만 찾는다. 틀린 표현은 `=`이다. 올바르게는 `WHERE name LIKE "김%";`이다.

NoteBase

오답 포인트

`LIKE`와 `=`를 같은 용도로 보거나 `%`와 `_`를 모두 막연히 와일드카드로만 외우면 틀리기 쉽다. 특히 3번은 `LIKE`를 썼다는 이유만으로 맞다고 보기 쉬우나 `_`는 한 글자만 허용한다는 점이 함정이다.

다시 나오면 이렇게 판단

문자열 검색 문제는 먼저 `=`인지 `LIKE`인지부터 구분한 다음, 패턴 기호가 `%`인지 `_`인지 확인하면 된다. 시작 문자열 뒤에 길이 제한 없이 이어지면 `%`, 정확히 한 글자만 더 오면 `_`로 판단한다.

암기 포인트

`=`는 완전일치, `LIKE`는 패턴일치, `%`는 여러 글자, `_`는 한 글자이다.

NoteBase

문 7. 지연 갱신(deferred update) 회복 기법에 대한 설명으로 가장 옳지 않은 것은?

① 장애 발생 시점에 따라 redo 연산과 undo 연산을 실행하여 데이터베이스를 복구한다.

② 로그 레코드에 변경 이전 값을 기록할 필요가 없다.

③ 트랜잭션이 수행되는 도중에 장애가 발생하면 로그에 기록된 내용을 버리기만 하면 된다.

④ 트랜잭션이 부분 완료된 후에 로그에 기록된 내용을 이용해 데이터베이스에 한번에 반영한다.

정답 1번

출제 포인트

이 문제는 지연 갱신과 즉시 갱신의 차이, 특히 장애 복구 시 undo가 필요한지와 로그에 무엇을 남기는지를 묻는 문제이다.

핵심 개념

지연 갱신은 트랜잭션이 완료되기 전에는 데이터베이스에 실제 갱신을 반영하지 않고 로그에만 기록하는 방식이다. 즉시 갱신은 트랜잭션 완료 전에도 데이터베이스에 일부 갱신이 반영될 수 있으므로 장애 시 undo와 redo가 모두 필요할 수 있다.

선지별 해설

1번: X. 틀린 부분은 "redo 연산과 undo 연산을 실행하여"이다. 지연 갱신에서는 완료 전 데이터베이스에 실제 갱신을 하지 않으므로 미완료 트랜잭션을 되돌릴 undo가 필요하지 않다. 올바르게는 "장애 발생 시 필요하면 redo만 수행하여 복구한다"이다.
2번: O. 지연 갱신은 실제 데이터베이스에 반영되기 전의 값을 되돌릴 일이 없으므로 로그에 변경 이전 값, 즉 before image를 기록할 필요가 없다.
3번: O. 트랜잭션 수행 도중 장애가 나면 아직 데이터베이스에 반영된 내용이 없으므로 그 트랜잭션의 로그 기록을 버리면 된다. undo 대상 자체가 없다.
4번: O. 지연 갱신은 트랜잭션이 부분 완료된 뒤 로그에 저장된 변경 내용을 이용해 데이터베이스에 반영하는 방식으로 이해하면 된다. 핵심은 실제 반영 시점이 완료 이후라는 점이다.

오답 포인트

즉시 갱신의 복구 방식인 "undo와 redo를 모두 고려"하는 특징을 지연 갱신에도 그대로 대입하면 1번을 맞는 설명처럼 보게 된다. 이 문제는 "실제 데이터베이스 반영이 언제 일어나는가"를 기준으로 두 기법을 구분하는지가 핵심 함정이다.

NoteBase

다시 나오면 이렇게 판단

먼저 완료 전 데이터베이스에 실제 갱신이 들어가는지를 본다. 완료 전 반영이면 즉시 갱신 쪽이라 undo 가능성을 본다. 완료 후 반영이면 지연 갱신 쪽이므로 미완료 트랜잭션에 대한 undo는 없고 필요하면 redo만 생각하면 된다.

암기 포인트

지연 갱신은 나중에 반영하므로 undo가 없고, 즉시 갱신은 먼저 반영될 수 있으므로 undo와 redo를 함께 본다.

NoteBase

문 8. 클러스터링 인덱스에 대한 설명으로 가장 옳지 않은 것은?

① 논리적으로 관련된 데이터를 물리적으로도 디스크상에 인접시켜 저장하는 것을 말한다.

② 수정 빈도수가 높은 칼럼을 사용하며 하나의 테이블에는 단 하나의 클러스터링 인덱스를 생성할 수 있다.

③ 클러스터링 인덱스는 인덱스 자체의 리프 페이지가 곧 데이터이다.

④ 클러스터링 인덱스를 생성할 때는 데이터 페이지 전체를 다시 정렬하게 된다.

정답 2번

출제 포인트

이 문제는 클러스터링 인덱스의 저장 방식과 생성·유지 특성을 정확히 구별하는지를 묻는 문제이다.

핵심 개념

클러스터링 인덱스는 논리적으로 인접한 데이터를 물리적으로도 가까이 배치하도록 정렬하는 인덱스이다. 한 테이블의 데이터 페이지 자체가 한 가지 물리적 순서만 가질 수 있으므로 보통 클러스터링 인덱스도 하나만 둘 수 있다. 대신 데이터 재배치 비용이 커서 수정이 잦은 칼럼에는 불리하다.

NoteBase

선지별 해설

① O
논리적으로 관련된 데이터를 디스크상에서도 인접하게 저장한다는 설명이므로 클러스터링 인덱스의 핵심 개념에 맞는다.

② X
틀린 부분은 수정 빈도수가 높은 칼럼을 사용한다는 표현이다.
클러스터링 인덱스는 데이터 페이지의 물리적 순서를 유지해야 하므로 삽입·삭제·갱신이 잦으면 재정렬과 페이지 분할 부담이 커진다. 올바르게는 조회 범위가 많고 순차 접근 이점이 큰, 비교적 안정적인 칼럼에 두는 편이 적절하다.
하나의 테이블에는 클러스터링 인덱스를 하나만 생성할 수 있다는 부분은 맞지만, 앞부분이 틀렸으므로 선지 전체는 틀리다.

③ O
클러스터링 인덱스에서는 인덱스의 리프 페이지가 별도 포인터 집합이 아니라 실제 데이터 페이지 역할을 한다. 따라서 리프 페이지가 곧 데이터라는 설명은 맞다.

④ O
클러스터링 인덱스를 생성하면 데이터 페이지를 해당 인덱스 키 순서에 맞게 다시 정렬하게 된다. 물리적 저장 순서를 바꾸는 인덱스라는 점을 설명한 것이므로 옳다.

오답 포인트

수험생은 ②의 뒤 문장인 하나의 테이블에는 단 하나의 클러스터링 인덱스를 생성할 수 있다는 사실이 맞기 때문에 선지 전체도 맞다고 넘기기 쉽다. 그러나 이 문항의 함정은 수정 빈도수가 높은 칼럼이라는 앞부분에 있다. 클러스터링 인덱스는 조회 성능은 좋지만 물리적 재배치 비용이 커서 갱신이 많은 칼럼과는 잘 맞지 않는다.

다시 나오면 이렇게 판단

클러스터링 인덱스 선지는 먼저 물리적 정렬이 일어나는지, 리프 페이지가 실제 데이터인지, 테이블당 하나만 가능한지를 확인한다. 그다음 수정이 잦은 칼럼에 유리하다고 하면 바로 틀린 선지로 판단할 수 있다.

암기 포인트

클러스터링 인덱스는 물리적 정렬, 리프 페이지는 데이터, 테이블당 하나, 갱신 많은 칼럼에는 불리하다.

NoteBase

문 9. ERD(Entity–Relationship Diagram)를 관계형 모델로 변환하는 방법 중 가장 옳지 않은 것은?

① 1:N 관계(relationship) → 외래키(foreign key) 또는 관계 릴레이션

② M:N 관계(relationship) → 2개의 외래키(foreign key)를 가지는 관계 릴레이션

③ 다치 속성(multivalued attribute) → 외래키(foreign key)를 가지는 릴레이션

④ 복합 속성(composite attribute) → 유도 속성(derived attribute)들의 집합

정답 4번

출제 포인트

이 문제는 ERD의 관계, 다치 속성, 복합 속성을 관계형 모델의 릴레이션과 외래키 구조로 어떻게 변환하는지를 구분하는 문제이다.

핵심 개념

1:N 관계는 보통 N쪽 릴레이션에 1쪽 기본키를 외래키로 포함시켜 표현하며, 관계 자체의 속성이 중요하면 별도 관계 릴레이션으로 둘 수 있다.
M:N 관계는 한 릴레이션에 직접 넣기 어려우므로 양쪽 엔티티의 기본키를 외래키로 갖는 별도 관계 릴레이션으로 변환한다.
다치 속성은 한 엔티티가 같은 속성값을 여러 개 가질 수 있는 경우이므로 원래 엔티티의 키를 외래키로 가진 별도 릴레이션으로 분리한다.
복합 속성은 하나의 속성이 여러 하위 속성으로 분해되는 것이므로 유도 속성의 집합이 아니라 단순 속성들의 집합이다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월

선지별 해설

① O

1:N 관계는 다수 측에 일수 측의 기본키를 외래키로 두어 표현하는 것이 기본이다. 관계 자체를 독립적으로 관리해야 하거나 관계 속성이 있으면 관계 릴레이션으로 분리할 수도 있으므로 선지의 설명이 타당하다.

② O

M:N 관계는 어느 한쪽에 외래키 하나만 두어서는 표현되지 않으므로 두 엔티티의 키를 각각 외래키로 갖는 별도 관계 릴레이션으로 변환한다. 따라서 설명이 맞다.

③ O

다치 속성은 한 엔티티가 여러 값을 가지는 속성이므로 별도 릴레이션으로 분해한다. 이 릴레이션은 원래 엔티티의 키를 외래키로 가지며, 보통 그 키와 다치 속성값을 함께 사용해 튜플을 식별한다.

④ X

복합 속성은 이름, 주소처럼 하나의 속성이 여러 하위 속성으로 나뉘는 구조이다. 틀린 표현은 '유도 속성들의 집합'이다. 올바르게는 '복합 속성은 단순 속성 또는 하위 속성들의 집합'이다. 유도 속성은 다른 속성값으로부터 계산되는 속성이므로 복합 속성과 개념이 다르다.

오답 포인트

복합 속성, 다치 속성, 유도 속성을 모두 '여러 값과 관련된 속성'처럼 느껴서 섞기 쉽다. 이 문항은 4번에서 복합 속성을 유도 속성으로 바꿔치기하여, '분해되는 속성'과 '계산되는 속성'의 차이를 헷갈리게 만든다.

다시 나오면 이렇게 판단

ERD 속성을 보면 먼저 세 가지를 구분한다. 여러 값이 반복되면 다치 속성이므로 별도 릴레이션 분리, 하나의 속성이 세부 항목으로 나뉘면 복합 속성이므로 하위 속성 분해, 다른 값으로 계산되면 유도 속성이므로 계산 결과 속성으로 판단한다. 관계는 1:N이면 외래키 흡수, M:N이면 별도 관계 릴레이션 생성으로 정리하면 된다.

암기 포인트

1:N은 외래키, M:N은 관계 릴레이션, 다치 속성은 별도 릴레이션, 복합 속성은 하위 속성 분해이다.

NoteBase

문 10. <보기 1>은 두 개의 릴레이션 R(a, b, c)과 S(d, e, f)를 보여주고 있다. 두 릴레이션에 대해 <보기 2>의 관계대수식을 수행한 결과로 가장 옳은 것은?

<보기 1>

R		
a	b	c
P1	B1	50
P2	B3	150
P3	B5	30
P4	B9	80

S		
d	e	f
P1	E3	2000
P1	E6	4000
P3	E6	6000
P3	E9	3000

<보기 2>

$R \ltimes_{(R.a = S.d)} S$

※ 연산 기호 "⋉"는 세미조인(semi-join) 기호임.

① a: P1, b: B1, c: 50
a: P3, b: B5, c: 30

② a: P1, b: B1, c: 50, d: P1, e: E3, f: 2000
a: P1, b: B1, c: 50, d: P1, e: E6, f: 4000
a: P3, b: B5, c: 30, d: P3, e: E6, f: 6000
a: P3, b: B5, c: 30, d: P3, e: E9, f: 3000

③ a: P1, b: B1, c: 50, e: E3, f: 2000
a: P1, b: B1, c: 50, e: E6, f: 4000
a: P3, b: B5, c: 30, e: E6, f: 6000
a: P3, b: B5, c: 30, e: E9, f: 3000

④ a: P2, b: B3, c: 150
a: P4, b: B9, c: 80

정답 1번

출제 포인트

이 문제는 표로 주어진 두 릴레이션에 세미조인 $R \ltimes_{R.a = S.d} S$을 적용했을 때, 매칭되는 왼쪽 릴레이션의 튜플만 남는다는 점을 묻는 문제이다.

NoteBase

자료 읽는 순서

1. 조인 조건이 R.a=S.d인지 확인한다.
2. S.d에 존재하는 값과 같은 a 값을 R에서 찾는다.
3. 세미조인은 조인 여부만 검사하고 결과에는 R의 속성 a, b, c만 남긴다는 점을 적용한다.

선지별 해설

1번: O. S.d 값은 P1, P1, P3, P3이므로 R에서 매칭되는 a 값은 P1과 P3이다. 세미조인은 매칭된 R의 튜플만 남기고 S의 속성은 붙이지 않으므로 결과는 (P1, B1, 50), (P3, B5, 30)이다.
2번: X. 이 결과는 매칭된 각 쌍마다 R과 S의 속성 a, b, c, d, e, f를 모두 붙인 조인 결과이다. 틀린 표현은 세미조인 결과라는 점이고, 올바르게는 조건조인 $R \bowtie_{R.a=S.d} S$ 결과이다.
3번: X. 이 결과도 매칭된 각 쌍을 유지하면서 S의 조인 속성 d만 제외한 조인 결과이다. 틀린 표현은 세미조인 결과라는 점이고, 올바르게는 조인 결과에서 중복되는 조인 속성만 제거한 형태이다.
4번: X. 이 결과는 S와 매칭되지 않는 R의 튜플 P2, P4만 남긴 것이다. 틀린 표현은 세미조인 결과라는 점이고, 올바르게는 매칭 실패한 튜플만 모은 반대 방향의 결과이다.

오답 포인트

세미조인을 일반 조인과 혼동하면 2번이나 3번처럼 매칭된 모든 쌍을 그대로 펼친 결과를 고르게 된다. P1과 P3가 S에서 두 번씩 나타나더라도 세미조인은 조인 결과를 확장하는 연산이 아니라, R에서 매칭 여부만 검사해 해당 튜플을 남기는 연산이라는 점이 함정이다.

다시 나오면 이렇게 판단

세미조인 기호가 보이면 먼저 조인 조건으로 왼쪽 릴레이션에서 매칭되는 튜플을 찾고, 그 다음 결과 속성이 왼쪽 릴레이션의 속성만 남는지 확인한다. 오른쪽 릴레이션의 속성이 붙어 있으면 조인이고, 매칭되지 않은 튜플만 남아 있으면 오답이다.

암기 포인트

세미조인 = 매칭 검사는 양쪽으로 하고, 결과는 왼쪽 릴레이션만 남긴다.

NoteBase

문 11. <보기>와 같이 테이블 student를 생성하고 레코드를 삽입한 후, 각각의 쿼리를 실행할 때 결과 값이 가장 작은 것은?

<보기>

```
CREATE TABLE student(
stuNum      CHAR(10) PRIMARY KEY,
stuName     CHAR(50) UNIQUE,
stuAge      INT,
stuProgram  CHAR(50) NOT NULL
)
INSERT INTO student (stuNum, stuName, stuAge, stuProgram)
VALUES ('A001', 'park', 50, 'math')
INSERT INTO student (stuNum, stuName, stuAge, stuProgram)
VALUES ('A002', 'yoon', 50, 'science')
INSERT INTO student (stuNum, stuProgram)
VALUES ('A003', 'kor')
INSERT INTO student (stuNum, stuName, stuAge, stuProgram)
VALUES ('A004', 'jang', 50, 'eng')
INSERT INTO student (stuNum, stuName, stuAge, stuProgram)
VALUES ('A005', 'oh', 50, 'eng')
INSERT INTO student (stuNum, stuName, stuProgram)
VALUES ('A006', 'kim', 'kor')
```

①
```
SELECT COUNT(*)
FROM student
WHERE stuNum IS NOT NULL
```

②
```
SELECT COUNT(*)
FROM student
WHERE stuName IS NOT NULL
```

③
```
SELECT COUNT(stuProgram)
FROM student
```

④
```
SELECT COUNT(*)
FROM student
WHERE stuAge >= 20 OR stuAge < 20
```

정답 4번

출제 포인트

이 문제는 COUNT(*)와 COUNT(컬럼)의 차이, 그리고 NULL이 비교식과 WHERE 조건에서 제외된다는 점을 묻는 문제이다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월

조건 정리

삽입된 행은 총 6개이다.
stuNum은 기본키이므로 모든 행에서 NULL이 될 수 없다.
stuName은 A003에서만 NULL이다.
stuProgram은 NOT NULL 제약이 있으므로 모든 행에서 값이 존재한다.
stuAge는 A003, A006에서 NULL이고 나머지 4개 행은 50이다.

사용할 원리

COUNT(*)는 조건을 만족하는 행 수를 센다.
COUNT(컬럼)은 그 컬럼 값이 NULL이 아닌 행만 센다.
WHERE절에서 NULL과의 비교 결과는 참이 아니므로 결과 집합에 포함되지 않는다.

선지별 해설

① X
WHERE stuNum IS NOT NULL은 6개 행 모두 만족하므로 결과는 6이다. 가장 작은 값이 아니다.
틀린 표현은 ①이 최소값이라는 판단이다. 올바르게는 ①의 결과는 6이다.

② X
stuName이 NULL이 아닌 행은 A001, A002, A004, A005, A006의 5개이므로 결과는 5이다. 가장 작은 값이 아니다.
틀린 표현은 ②가 최소값이라는 판단이다. 올바르게는 ②의 결과는 5이다.

③ X
COUNT(stuProgram)은 stuProgram이 NULL이 아닌 행 수를 센다. stuProgram은 NOT NULL 제약이 있으므로 6개 모두 집계되어 결과는 6이다. 가장 작은 값이 아니다.
틀린 표현은 ③이 다른 COUNT 결과보다 작을 수 있다는 판단이다. 올바르게는 ③의 결과는 6이다.

④ O
stuAge >= 20 OR stuAge < 20은 stuAge가 숫자인 경우 사실상 모든 비NULL 나이를 포함하지만, stuAge가 NULL인 행은 두 비교가 모두 참이 아니어서 제외된다. 따라서 50인 4개 행만 집계되어 결과는 4이다. 네 쿼리 중 가장 작다.

오답 포인트

④에서 '20 이상 또는 20 미만이면 전부 세어진다'고만 보고 NULL 행까지 포함시키는 실수가 가장 흔하다.
SQL의 NULL은 비교 결과가 참도 거짓도 아닌 상태가 되어 WHERE절에서 빠진다. 또 COUNT(*)와 COUNT(컬럼)을 같은 것으로 보면 ③을 6이 아닌 다른 값으로 오해할 수 있다.

NoteBase

다시 나오면 이렇게 판단

먼저 각 컬럼의 NULL 가능 여부와 제약조건을 표시한 뒤 COUNT(*)인지 COUNT(컬럼)인지 구분한다. 그다음 WHERE절에 비교식이 있으면 NULL 행이 참으로 남는지부터 확인하면 된다.

암기 포인트

COUNT(*)는 행 수이고 COUNT(컬럼)은 NULL이 아닌 값 수이다. NULL은 비교식에서 자동으로 제외된다.

NoteBase

문 12. 데이터 웨어하우스(data warehouse)에 대한 설명으로 가장 옳은 것은?

① 스타 스키마(star schema)는 한 개의 사실 테이블(fact table)과 정규화가 되어 있지 않은 차원 테이블(dimension table)들로 구성되며 추가적인 정규화가 필요하다.

② 드릴다운(drill-down)은 동일한 차원 또는 큐브 내에서 상세한 형태의 데이터 수준에서 보다 더 요약된 데이터를 탐색하는 연산이다.

③ 데이터 마트(data mart)는 데이터 웨어하우스에 비해 데이터 용량에서 차이가 있으며 사용자 지원을 위해 온라인 트랜잭션 작업을 수행한다.

④ 데이터 웨어하우스에 데이터가 적재되면 일괄처리 작업에 의한 갱신 이외에는 삽입이나 삭제 등의 변경이 수행되지 않는다.

정답 4번

출제 포인트

이 문제는 데이터 웨어하우스의 핵심 특성과 관련 개념인 스타 스키마, 드릴다운, 데이터 마트의 차이를 정확히 구분하는지를 묻는 문제이다.

핵심 개념

스타 스키마는 하나의 사실 테이블을 중심으로 비정규화된 차원 테이블이 연결되는 구조이다.
스노우플레이크 스키마는 차원 테이블을 추가로 정규화한 구조이다.
드릴다운은 요약된 수준에서 더 상세한 수준으로 내려가며 보는 연산이고, 롤업은 상세한 수준에서 요약된 수준으로 올라가는 연산이다.
데이터 마트는 전사적 데이터 웨어하우스보다 범위가 작고 특정 부서나 주제 중심으로 구성되며, 주된 활용은 OLAP 분석 지원이다.
데이터 웨어하우스는 한 번 적재된 데이터가 자주 변경되지 않는 비휘발성(non-volatile) 특성을 가진다.

NoteBase

선지별 해설

① X: 틀린 부분은 "추가적인 정규화가 필요하다"이다. 스타 스키마의 차원 테이블은 의도적으로 비정규화되어 단순한 조인과 빠른 분석을 지원한다. 추가 정규화가 필요한 구조는 스타 스키마가 아니라 스노우플레이크 스키마이다.
② X: 틀린 부분은 "상세한 형태의 데이터 수준에서 보다 더 요약된 데이터를 탐색하는"이다. 이는 드릴다운이 아니라 롤업의 설명이다. 드릴다운은 요약 데이터에서 세부 데이터로 내려가며 분석하는 연산이다.
③ X: 틀린 부분은 "온라인 트랜잭션 작업을 수행한다"이다. 데이터 마트는 데이터 웨어하우스와 마찬가지로 의사결정 지원과 분석을 위한 저장소이므로 주 용도는 OLAP이다. OLTP는 일상적 거래 처리를 위한 운영 데이터베이스의 성격에 가깝다.
④ O: 데이터 웨어하우스는 일괄 적재 이후에 일반적인 운영 DB처럼 빈번한 삽입, 수정, 삭제가 일어나지 않는 비휘발성 특성을 가진다. 따라서 배치에 의한 갱신 외에는 데이터 변경이 거의 수행되지 않는다는 설명이 옳다.

오답 포인트

①은 스타 스키마와 스노우플레이크 스키마를 뒤바꾸는 함정이고, ②는 드릴다운과 롤업의 방향을 반대로 제시한 함정이다. ③은 데이터 마트도 분석용 저장소라는 점을 흐리고 OLTP와 OLAP를 혼동하게 만드는 선지이다.

다시 나오면 이렇게 판단

스키마 문제는 차원 테이블이 비정규화인지 추가 정규화인지부터 구분한다. OLAP 연산 문제는 드릴다운은 요약에서 상세로, 롤업은 상세에서 요약으로 방향만 먼저 잡으면 된다. 데이터 마트와 데이터 웨어하우스는 모두 분석용이라는 점을 기준으로 OLTP 표현을 경계하면 된다.

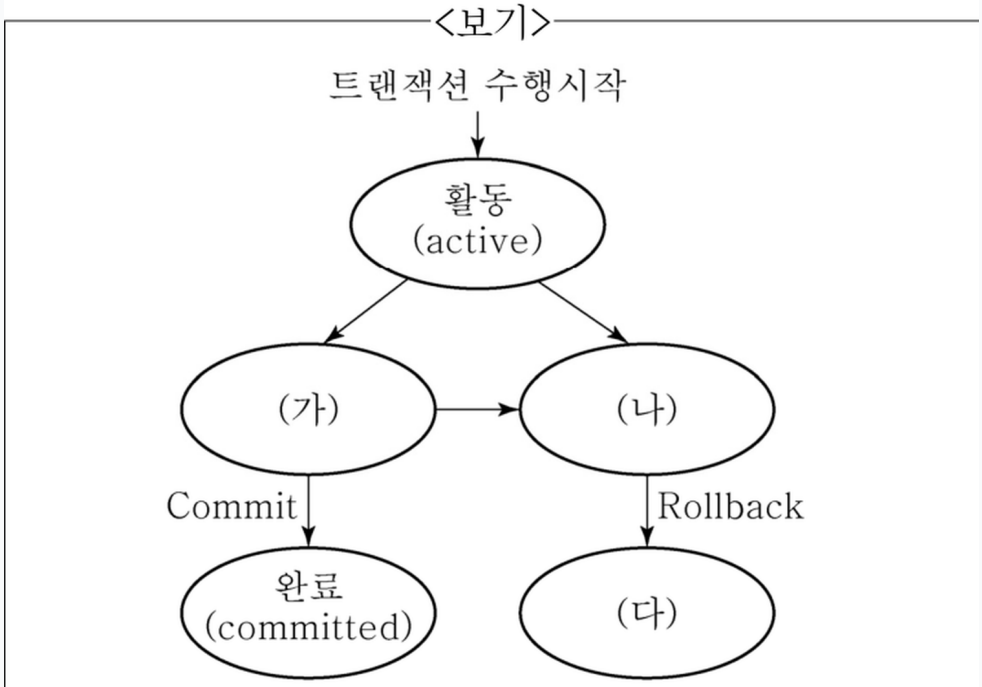
암기 포인트

스타 스키마는 비정규화, 스노우플레이크는 정규화이다. 드릴다운은 아래로 내려감, 롤업은 위로 올림이다. 데이터 웨어하우스와 데이터 마트는 OLAP 중심, 비휘발성이다.

NoteBase

문 13. <보기>의 트랜잭션 상태 흐름도에서 (가), (나), (다)에 들어갈 용어로 옳은 것은?

<보기>



① (가) 중지(stopped), (나) 부분 완료(partially committed), (다) 철회(aborted)

② (가) 부분 완료(partially committed), (나) 철회(aborted), (다) 실패(failed)

③ (가) 중지(stopped), (나) 철회(aborted), (다) 실패(failed)

④ (가) 부분 완료(partially committed), (나) 실패(failed), (다) 철회(aborted)

정답 4번

출제 포인트

이 문제는 트랜잭션 상태 흐름도에서 commit 직전의 부분 완료, 오류 발생 직후의 실패, rollback 완료 후의 철회를 구분하는 문제이다.

NoteBase

선지별 해설

① X: (가)를 stopped로 본 점이 틀리다. active에서 모든 연산을 끝내고 commit 직전으로 가는 상태는 partially committed이다. (나)도 partially committed가 아니라 오류가 발생한 failed이다. (다)가 aborted라는 점만 맞고 전체 조합은 틀리다. 올바르게는 (가) 부분 완료, (나) 실패, (다) 철회이다.
② X: (가)를 partially committed로 본 점은 맞다. 그러나 rollback은 failed 상태에서 수행되므로 (나)는 aborted가 아니라 failed이다. rollback이 끝난 결과 상태가 (다)이므로 (다)는 failed가 아니라 aborted이다. 올바르게는 (가) 부분 완료, (나) 실패, (다) 철회이다.
③ X: (가)를 stopped로 본 점이 틀리다. 이 도식에서 committed로 내려가는 직전 상태는 partially committed이다. 또한 rollback 직전 상태인 (나)는 aborted가 아니라 failed이고, rollback 후 상태인 (다)는 failed가 아니라 aborted이다. 올바르게는 (가) 부분 완료, (나) 실패, (다) 철회이다.
④ O: active에서 마지막 문장 실행까지 마친 뒤 commit이 확정되기 직전 상태가 (가)인 partially committed이다. 그 상태나 active에서 오류가 나면 (나)인 failed로 간다. failed 상태에서 rollback이 끝나면 (다)인 aborted가 된다.

오답 포인트

출제자는 failed와 aborted의 시점을 뒤바꾸는 함정을 넣었다. 오류가 발생한 직후는 failed이고, rollback 처리까지 끝난 뒤가 aborted이다. 또한 commit 화살표 바로 위 상태를 stopped로 착각하면 (가)를 잘못 고르게 된다.

다시 나오면 이렇게 판단

트랜잭션 상태도는 화살표의 앞뒤 시점을 먼저 읽으면 된다. commit으로 내려가기 직전 노드는 partially committed이고, rollback 화살표가 시작되는 노드는 failed이며, rollback 화살표가 도착한 결과 노드는 aborted이다.

암기 포인트

commit 전은 partially committed, 오류 직후는 failed, rollback 후는 aborted이다.

NoteBase

문 14. 어떤 트랜잭션이 자체는 문제가 없지만 다른 트랜잭션의 간섭이 있을 때, 잘못된 답을 발생시키는 경우가 아닌 것은?

① 갱신 분실(lost update)

② 모순성(inconsistency)

③ 직렬 스케줄(serial schedule)

④ 연쇄 복귀(cascading rollback)

정답 3번

출제 포인트

트랜잭션 스케줄에서 간섭으로 생기는 이상 현상과 직렬 스케줄의 의미를 구별하는 문제이다.

핵심 개념

갱신 분실은 둘 이상의 트랜잭션이 같은 데이터를 갱신하는 과정에서 나중 값이 먼저 값을 덮어써 이전 갱신이 사라지는 현상이다.
모순성은 한 트랜잭션의 일부 갱신만 반영된 중간 상태를 다른 트랜잭션이 읽어 일관되지 않은 값을 보는 현상이다.
직렬 스케줄은 트랜잭션들을 겹치지 않게 하나씩 순서대로 실행하는 스케줄로, 간섭에 따른 이상 현상을 피하는 기준 형태이다.
연쇄 복귀는 아직 완료되지 않은 트랜잭션의 값을 다른 트랜잭션이 읽은 뒤 원래 트랜잭션이 롤백되어 뒤따라 롤백이 퍼지는 현상이다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월

선지별 해설

① X
갱신 분실은 다른 트랜잭션의 간섭 때문에 한 갱신 결과가 사라지는 대표적 이상 현상이다.
틀린 부분은 '아닌 것의 답으로 보는 판단'이다.
올바르게는 갱신 분실은 간섭이 있을 때 잘못된 답을 만드는 경우에 해당한다.

② X
모순성은 부분 갱신 상태를 읽어 데이터의 일관성이 깨지는 현상으로, 역시 간섭 때문에 잘못된 결과를 만든다.
틀린 부분은 '아닌 것의 답으로 보는 판단'이다.
올바르게는 모순성은 간섭이 있을 때 발생하는 비정상 현상이다.

③ O
직렬 스케줄은 트랜잭션을 순차적으로 실행하므로 서로 간섭하지 않는다.
따라서 잘못된 답을 발생시키는 경우가 아니라 정합성을 보장하는 기준 스케줄이다.

④ X
연쇄 복귀는 미완료 트랜잭션의 값을 읽은 뒤 그 트랜잭션이 취소되면서 관련 트랜잭션까지 연달아 롤백되는 현상이다.
틀린 부분은 '아닌 것의 답으로 보는 판단'이다.
올바르게는 연쇄 복귀는 간섭으로 인한 비정상 처리의 한 형태이다.

오답 포인트

이 문항은 보기들이 모두 트랜잭션 처리 용어라서 용어를 아는지만으로는 흔들리기 쉽다. 직렬 스케줄만 이상 현상이 아니라 이상 현상을 막는 실행 방식이라는 점을 놓치면 ①, ②, ④와 같은 anomaly 용어와 섞여 오답을 고르기 쉽다.

다시 나오면 이렇게 판단

보기의 용어가 이상 현상인지 스케줄 방식인지부터 나누면 된다. 갱신 분실, 모순성, 연쇄 복귀처럼 간섭의 결과를 나타내는 말은 이상 현상이고, 직렬 스케줄처럼 실행 방식을 나타내는 말은 이상 현상을 피하는 기준으로 본다.

암기 포인트

갱신 분실·모순성·연쇄 복귀는 이상 현상이고, 직렬 스케줄은 이상 현상을 막는 순차 실행 방식이다.

NoteBase

문 15. <보기 1>의 릴레이션 R에서 성립하는 함수 종속성을 <보기 2>에서 선택한 것으로 가장 옳은 것은?

<보기 1>

A	B	C
b	c	h
e	i	f
g	i	f
e	b	a

<보기 2>

ㄱ. A → B

ㄴ. B → C

ㄷ. (A, B) → C

ㄹ. (B, C) → A

① ㄱ, ㄴ

② ㄴ, ㄷ

③ ㄱ, ㄴ, ㄷ

④ ㄴ, ㄷ, ㄹ

정답 2번

출제 포인트

주어진 릴레이션 인스턴스에서 같은 결정자 값이 반복될 때 종속 속성 값이 달라지는지 확인하여 함수 종속 $A \rightarrow B, B \rightarrow C, (A, B) \rightarrow C, (B, C) \rightarrow A$의 성립 여부를 판정하는 문제이다.

자료 읽는 순서

1. 각 함수 종속의 왼쪽 속성값이 중복되는 행이 있는지 찾는다.
2. 중복되는 행이 있으면 오른쪽 속성값이 모두 같은지 확인한다.
3. 참인 보기 ㄱ~ㄹ만 남겨 선택지 1~4와 연결한다.

NoteBase

핵심 근거

ㄱ은 X이다. A가 e인 두 행에서 B가 i와 b로 달라 $A \to B$는 성립하지 않는다.
ㄴ은 O이다. B가 i인 두 행에서 C가 모두 f이므로 $B \to C$는 성립한다.
ㄷ은 O이다. (A,B) 쌍이 (b,c), (e,i), (g,i), (e,b)로 모두 달라 같은 (A,B)에서 다른 C가 나오는 반례가 없다.
ㄹ은 X이다. (B,C)=(i,f)인 두 행에서 A가 e와 g로 달라 $(B, C) \to A$는 성립하지 않는다.

선지별 해설

1번 X: ㄱ, ㄴ을 고른 선택지이다. ㄴ은 맞지만 ㄱ은 A=e에서 B가 i와 b로 갈려 성립하지 않는다. 올바르게는 ㄱ을 제외하고 ㄴ, ㄷ만 골라야 한다.
2번 O: ㄴ, ㄷ을 고른 선택지이다. B=i일 때 C가 모두 f라서 ㄴ이 맞고, 같은 (A,B) 조합이 반복되지 않아 ㄷ도 맞다.
3번 X: ㄱ, ㄴ, ㄷ을 고른 선택지이다. ㄴ과 ㄷ은 맞지만 ㄱ이 틀려 전체 선택지가 틀리다. 올바르게는 ㄱ을 빼고 ㄴ, ㄷ만 선택해야 한다.
4번 X: ㄴ, ㄷ, ㄹ을 고른 선택지이다. ㄴ과 ㄷ은 맞지만 ㄹ은 (B,C)=(i,f)에서 A가 e와 g로 달라 틀리다.
올바르게는 ㄹ을 제외한 ㄴ, ㄷ만 성립한다.

오답 포인트

복합 속성 함수 종속은 같은 왼쪽 값이 두 번 이상 나와야만 반례를 직접 볼 수 있다는 점을 놓치기 쉽다. 그래서 (A,B) 조합이 모두 다르면 $(A, B) \to C$를 틀렸다고 오해하거나, 반대로 (B,C)=(i,f)가 두 번 나온 사실을 놓쳐 ㄹ까지 맞다고 착각하기 쉽다.

다시 나오면 이렇게 판단

함수 종속 문제는 왼쪽 속성값이 같은 행끼리만 묶어서 본다. 같은 왼쪽 값에서 오른쪽 값이 하나라도 달라지면 X이고, 끝까지 반례가 없으면 O로 판단한다.

암기 포인트

함수 종속 판정은 같은 결정자끼리 비교하고, 다른 종속값이 나오면 바로 반례가 된다.

NoteBase

문 16. 릴레이션 R(A, B, C, D, E, F)에 대한 함수적 종속성이 <보기>와 같이 주어졌을 때 최소성 커버(minimal cover)에 포함되었다고 가장 보기 어려운 함수적 종속성은?

<보기>
FD = {ABC → CDEF, C → E, A → B, D → F}

① C → E

② AB → E

③ D → F

④ AC → D

정답 2번

출제 포인트

이 문제는 함수적 종속성을 최소성 커버로 바꿀 때 우변 분해, 좌변의 불필요한 속성 제거, 중복 종속성 제거를 할 수 있는지를 묻는다.

핵심 개념

최소성 커버는 각 함수적 종속성의 우변이 1개 속성이고, 좌변에 불필요한 속성이 없으며, 어떤 종속성을 빼도 전체 의미가 달라지는 함수적 종속성 집합이다.
이 문제의 한 최소성 커버는 {AC→D, C→E, A→B, D→F}이다. ABC→CDEF를 분해하면 ABC→D, ABC→E, ABC→F가 나오고, A→B 때문에 ABC→D에서 B는 불필요하여 AC→D가 된다. ABC→E는 C→E로 대체되고, ABC→F는 AC→D와 D→F로 유도되어 제거된다.

선지별 해설

1번 O: C→E는 원래 주어진 종속성이며, 이를 제거하면 C만으로 E를 얻을 다른 경로가 없다. 따라서 최소성 커버에 남는다.
2번 X: 틀린 부분은 "AB→E"이다. AB의 폐포는 A→B를 적용해도 여전히 {A, B}에 머물러 E가 나오지 않는다. 올바르게는 E는 C→E로 얻고, 최소성 커버에도 C→E가 포함된다.
3번 O: D→F는 원래 주어진 종속성이고 좌변도 더 줄일 수 없다. 이를 제거하면 D로부터 F를 얻을 방법이 없어지므로 최소성 커버에 포함된다.
4번 O: ABC→D에서 B는 A→B 때문에 불필요하다. 따라서 좌변 축소를 하면 AC→D가 되고, 이것이 최소성 커버에 포함된다.

NoteBase

오답 포인트

ABC→CDEF를 분해한 뒤 생기는 종속성을 그대로 외우면 2번도 그럴듯해 보인다. 그러나 최소성 커버에서는 새 종속성을 덧붙이는 것이 아니라 기존 종속성을 더 짧게 줄이거나 중복을 제거한다는 점이 함정이다. AB만으로는 C가 없으므로 E에 도달할 수 없다.

다시 나오면 이렇게 판단

먼저 우변을 한 속성씩 분해하고 자명한 종속성을 지운다. 다음으로 좌변 속성을 하나씩 빼 보아도 같은 우변이 나오면 그 속성은 불필요하다고 본다. 마지막으로 어떤 종속성이 다른 종속성들로 유도되면 제거한다. 이 절차를 거친 뒤 남는 종속성만 최소성 커버에 포함된다.

암기 포인트

최소성 커버는 우변 1개, 좌변 최소, 중복 제거이다. 이 문항에서는 AC→D, C→E, A→B, D→F만 남는다.

NoteBase

문 17. SQL 구문을 통해 권한을 부여할 때, 권한을 받은 사용자가 부여받은 권한을 다른 사용자에게 부여할 수 있게 지정하는 키워드는?

① WITH GRANT OPTION

② WITH REVOKE OPTION

③ WITH CHECK OPTION

④ WITH ASSIGNMENT OPTION

정답 1번

출제 포인트

이 문제는 SQL의 권한 부여 구문에서 권한을 다시 다른 사용자에게 재부여할 수 있게 하는 옵션을 구분하는 문제이다.

핵심 개념

GRANT는 사용자에게 권한을 부여하는 명령이며, WITH GRANT OPTION은 그 사용자가 받은 권한을 다시 다른 사용자에게 부여할 수 있게 하는 재부여 옵션이다.
REVOKE는 이미 준 권한을 취소하는 명령이다.
WITH CHECK OPTION은 뷰를 통해 삽입·수정된 행이 뷰의 조건을 계속 만족하도록 강제하는 옵션이다.

선지별 해설

① O: WITH GRANT OPTION은 권한을 받은 사용자가 그 권한을 다른 사용자에게 다시 부여할 수 있게 하는 올바른 키워드이다.
② X: 틀린 부분은 WITH REVOKE OPTION이라는 표현이다. REVOKE는 권한을 회수할 때 사용하는 명령이며, 권한 재부여 옵션은 WITH GRANT OPTION이다.
③ X: 틀린 부분은 WITH CHECK OPTION이라는 표현이다. WITH CHECK OPTION은 뷰의 갱신 가능 조건을 검사하는 옵션이지 권한을 재부여하는 옵션이 아니다.
④ X: 틀린 부분은 WITH ASSIGNMENT OPTION이라는 표현이다. 권한을 다른 사용자에게 넘길 수 있게 하는 표준 키워드는 WITH GRANT OPTION이다.

오답 포인트

GRANT, REVOKE, CHECK처럼 SQL 권한과 뷰 옵션에서 자주 보이는 단어를 섞어 놓아 헷갈리게 만든 문제이다. 특히 ③은 OPTION이라는 형태가 같아서 그럴듯해 보이지만, 적용 대상이 권한이 아니라 뷰 제약이라는 점이 핵심 차이이다.

NoteBase

다시 나오면 이렇게 판단

문장에서 핵심 동사가 권한을 부여한다인지, 권한을 취소한다인지, 뷰 조건을 검사한다인지부터 구분하면 된다. 다른 사용자에게 다시 줄 수 있게 한다는 표현이 보이면 WITH GRANT OPTION으로 판단할 수 있다.

암기 포인트

권한을 준다는 GRANT, 다시 주게 하는 것도 GRANT OPTION이다.

NoteBase

문 18. <보기 1>은 고객과 주문 정보를 담고 있는 customer, orders 테이블이다. 이 테이블에서 <보기 2>의 SQL을 수행할 때 출력되는 결과로 가장 옳은 것은?

<보기 1>

customer

cid	name	age
c002	A	50
c004	B	25
c010	C	33

orders

cid	pid	qty
c002	p12	10
c004	p04	5
c010	p08	45
c004	p09	50

<보기 2>

```
SELECT name AS cname
FROM customer
WHERE NOT EXISTS (
  SELECT *
  FROM orders
  WHERE qty = 5 AND customer.cid = orders.cid
)
```

① name: B

② cname: B

③ name: A, C

④ cname: A, C

정답 4번

출제 포인트

이 문제는 상관 서브쿼리의 NOT EXISTS 조건과 SELECT 절의 열 별칭을 함께 읽어 최종 결과 행과 열 이름을 판단하는 문제이다.

자료 읽는 순서

1. customer와 orders를 customer.cid = orders.cid로 연결해 각 고객별 주문을 확인한다.
2. qty = 5인 주문이 존재하는 고객을 찾는다. 표에서는 c004만 해당한다.
3. NOT EXISTS이므로 그 고객을 제외하고, SELECT name AS cname 이므로 출력 값은 name이지만 열 이름은 cname으로 판단한다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월

핵심 근거

orders 테이블에서 qty = 5인 행은 cid가 c004인 한 건이다.
따라서 customer에서 c004(B)는 제외되고 c002(A), c010(C)만 남는다.
또한 SELECT name AS cname 이므로 결과 열 이름은 name이 아니라 cname이다.

선지별 해설

① X
- 열 이름이 틀렸다. SELECT 절에서 name AS cname으로 별칭을 주었으므로 결과 열 제목은 name이 아니라 cname이다.
- 행도 틀렸다. B는 qty = 5인 주문이 존재하는 c004이므로 NOT EXISTS 조건을 만족하지 못한다.
- 틀린 표현은 'name / B'이다. 올바르게는 'cname / A, C'이다.

② X
- 열 이름 cname은 맞지만 행이 틀렸다. B는 orders에서 qty = 5인 주문이 있는 고객이라 제외된다.
- 틀린 표현은 'B'이다. 올바르게는 'A, C'이다.

③ X
- 행 A, C는 맞지만 열 이름이 틀렸다. SELECT name AS cname의 결과 열 이름은 cname이다.
- 틀린 표현은 'name'이다. 올바르게는 'cname'이다.

④ O
- qty = 5인 주문이 있는 고객은 c004(B)뿐이므로 B는 제외된다.
- 남는 고객 이름은 A, C이고, 결과 열 이름은 cname이므로 표시는 'cname / A, C'가 된다.

오답 포인트

NOT EXISTS를 보고 결과에 qty = 5인 고객 B가 남는다고 거꾸로 해석하거나, SELECT name AS cname에서 실제 데이터 값과 열 별칭을 혼동하면 ②나 ③으로 흔들리기 쉽다.

다시 나오면 이렇게 판단

상관 서브쿼리는 먼저 내부 조건에 걸리는 바깥 행이 누구인지 찾고, NOT EXISTS면 그 행을 제외한다고 판단한다. 마지막에는 SELECT 절의 별칭이 결과 열 제목을 바꾼다는 점을 따로 확인한다.

암기 포인트

NOT EXISTS는 조건에 맞는 행을 제외하고, AS는 결과 열 이름만 바꾼다.

NoteBase

문 19. 분산 데이터베이스(distributed database)에 대한 설명으로 가장 옳지 않은 것은?

① 분산된 데이터베이스의 양이 줄어들어 데이터베이스 설계 및 구축의 비용이 감소한다.

② 동일한 데이터가 저장된 여러 지역에서 병렬 처리를 수행할 수 있어 데이터 처리 성능이 향상된다.

③ 중복되는 데이터의 일치성을 보장하기 위한 추가적인 부하가 발생한다.

④ 데이터 처리 요청이 한 지역에 집중되지 않고 여러 지역에 분산되므로 처리 부담을 줄일 수 있다.

정답 1번

출제 포인트

이 문제는 분산 데이터베이스의 대표적인 장점과 단점을 구별할 수 있는지를 묻는 문제이다.

핵심 개념

분산 데이터베이스는 논리적으로는 하나의 데이터베이스이지만 물리적으로는 여러 지역에 데이터를 분산 저장하여 관리하는 시스템이다. 장점은 병렬 처리, 부하 분산, 가용성 향상이고, 단점은 중복 데이터 관리, 일관성 유지, 분산 제어와 통신에 따른 복잡성 및 비용 증가이다.

선지별 해설

1번 X: 틀린 부분은 "데이터베이스의 양이 줄어들어"와 "비용이 감소한다"는 표현이다. 분산 데이터베이스는 데이터를 여러 지역에 분산하거나 일부 중복 저장하므로 관리해야 할 요소가 단순히 줄어드는 구조가 아니다. 실제로는 분산 제어, 통신, 동기화, 일관성 관리가 추가되어 설계 및 구축 비용이 커지기 쉽다.
2번 O: 동일한 데이터의 복제본이나 분산된 데이터를 여러 지역에서 나누어 처리할 수 있으므로 병렬 처리가 가능하다. 그 결과 응답 시간 단축과 처리 성능 향상을 기대할 수 있다.
3번 O: 분산 데이터베이스는 데이터 중복을 활용할 수 있지만, 그만큼 각 지역의 데이터가 같은 상태를 유지하도록 동기화해야 한다. 따라서 일관성 보장을 위한 추가 제어와 통신 부하가 발생한다.
4번 O: 분산 데이터베이스는 요청을 여러 지역 시스템에 나누어 처리할 수 있으므로 특정 지역이나 서버에 부하가 집중되는 현상을 완화할 수 있다. 이는 부하 분산이라는 대표적인 장점에 해당한다.

오답 포인트

분산 데이터베이스를 단순히 "나누어 저장하므로 양이 줄고 비용도 줄어든다"고 연결하면 틀린다. 분산 시스템은 성능과 가용성 면에서는 유리하지만, 구축과 관리의 복잡성은 오히려 커진다. 또한 발문이 "가장 옳지 않은 것"인지 먼저 확인하지 않으면 장점 선지를 골라 버리기 쉽다.

다시 나오면 이렇게 판단

분산 데이터베이스 문제는 먼저 장점과 단점을 나누어 판단한다. 병렬 처리, 부하 분산, 가용성 향상은 장점이고, 일관성 유지, 통신 오버헤드, 설계와 운영의 복잡성 증가는 단점으로 보면 빠르게 정리된다.

NoteBase

암기 포인트

분산 데이터베이스는 성능과 가용성은 높이고, 관리 복잡성과 비용은 높인다.

NoteBase

문 20. 메인 메모리 데이터베이스(main memory database) 시스템에 대한 설명으로 가장 옳지 않은 것은?

① 데이터의 저장과 연산이 메모리에서 수행되며 메모리의 데이터를 일정 주기로 디스크에 기록해야 한다.

② 데이터와 로그를 메인 메모리에 저장하므로 디스크 입출력이 발생하지 않는다.

③ 백업 매체는 디스크 기반 데이터베이스와 같이 디스크를 사용한다.

④ 디스크 기반 데이터베이스보다 트랜잭션 처리 속도 면에서 더 좋은 성능을 보인다.

정답 2번

출제 포인트

이 문제는 메인 메모리 데이터베이스의 핵심 특성과 디스크 기반 데이터베이스와의 차이를 묻는 문제이다.

핵심 개념

메인 메모리 데이터베이스는 주기억장치를 주 저장소로 사용하여 데이터 접근과 연산을 수행하는 데이터베이스이다. 핵심 차이축은 주 저장 위치와 입출력 경로이다. 메인 메모리 데이터베이스는 주 작업이 메모리에서 이루어져 빠르지만, 장애 복구와 영속성 확보를 위해 로그, 체크포인트, 백업 등에서는 보조 저장장치를 함께 사용한다. 디스크 기반 데이터베이스는 디스크를 중심 저장소로 사용하므로 디스크 입출력 비용의 영향이 더 크다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2019 · 10월

선지별 해설

① O
데이터의 저장과 연산이 메모리에서 수행된다는 설명은 메인 메모리 데이터베이스의 핵심 특징에 해당한다. 또한 휘발성 메모리의 한계를 보완하려면 체크포인트나 스냅샷 형태로 메모리의 내용을 주기적으로 디스크에 기록해야 하므로 옳다.

② X
틀린 부분은 '디스크 입출력이 발생하지 않는다'는 표현이다. 메인 메모리 데이터베이스는 주된 처리 경로가 메모리일 뿐이며, 장애 복구와 영속성 보장을 위해 로그 기록, 체크포인트, 백업 과정에서 디스크 입출력이 발생한다. 올바르게는 '디스크 기반 데이터베이스보다 디스크 입출력이 크게 줄어들 수 있지만, 디스크 입출력이 전혀 없는 것은 아니다'이다.

③ O
메인 메모리 데이터베이스도 장애 복구를 위한 백업 매체가 필요하므로 일반적으로 디스크 같은 비휘발성 저장장치를 사용한다. 따라서 디스크 기반 데이터베이스와 마찬가지로 백업 매체로 디스크를 사용할 수 있다.

④ O
주 데이터 접근과 연산이 메모리에서 이루어지므로 디스크 접근 지연이 크게 줄어든다. 따라서 일반적으로 디스크 기반 데이터베이스보다 트랜잭션 처리 성능 면에서 유리하다.

오답 포인트

메인 메모리 데이터베이스를 '모든 것이 메모리 안에서만 끝나는 시스템'으로 과장해서 이해하면 ②에 끌리기 쉽다. 이 문항의 함정은 '디스크 입출력이 줄어든다'와 '디스크 입출력이 전혀 없다'를 바꿔 놓은 데 있다.

다시 나오면 이렇게 판단

메인 메모리 데이터베이스 관련 선지는 '주 저장과 주 연산은 메모리에서 수행된다'와 '영속성과 복구를 위해 보조 저장장치를 완전히 배제하지는 않는다'를 함께 떠올리면 판단이 정리된다. 성능 향상과 디스크 완전 제거는 다른 주장이라는 점을 구분하면 된다.

암기 포인트

메인 메모리 DB는 처리 중심은 메모리, 영속성 보장은 디스크이다.

NoteBase