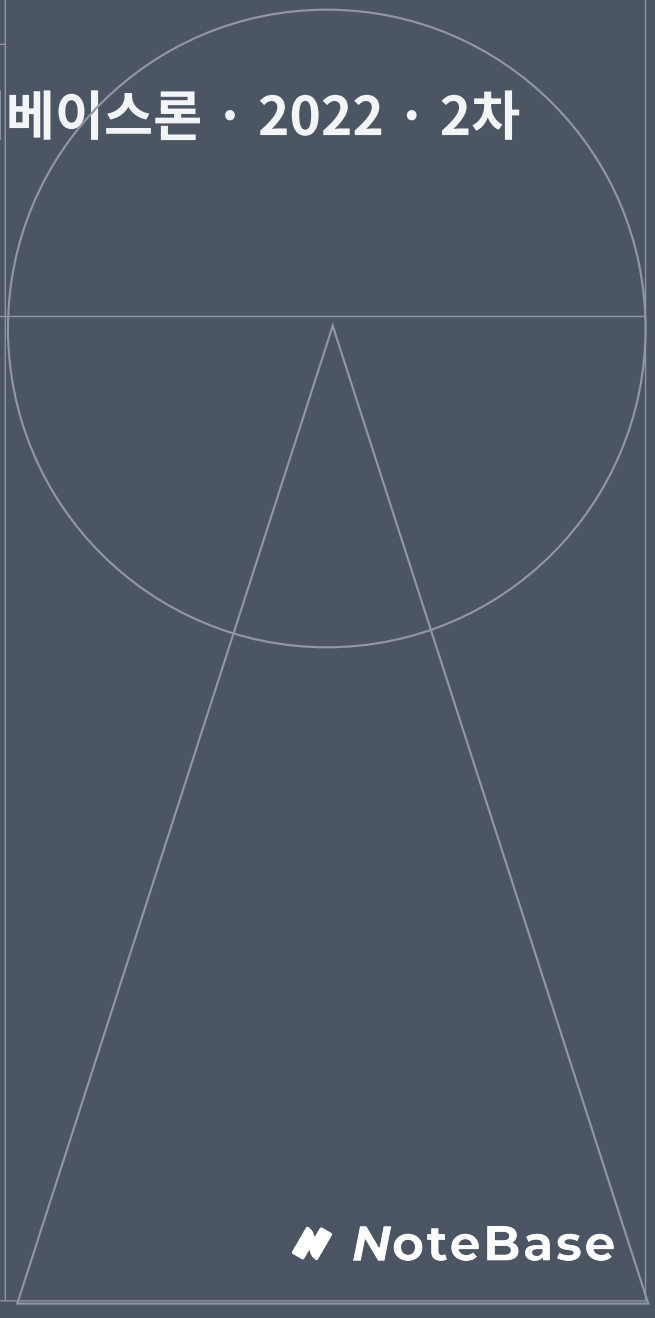
기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2022 · 2차



NoteBase

기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2022 · 2차

공무원/국가직/7급 · 데이터베이스론 · 2022 · 2차

문 1. 데이터베이스 관리 시스템(DBMS)에 대한 설명으로 옳지 않은 것은?

① 데이터 중복을 효율적으로 제어할 수 있다.

② 여러 사용자의 데이터 공유를 허용하지 않는다.

③ 사용자마다 데이터에 대한 적절한 권한을 부여하여 허가 받지 않은 접근을 통제할 수 있다.

④ 데이터베이스 시스템의 하드웨어나 소프트웨어의 고장에 대비하여 백업과 회복 기능을 제공한다.

정답 2번

출제 포인트

이 문제는 DBMS의 핵심 기능인 중복 제어, 데이터 공유, 권한 통제, 백업·회복 기능을 구분하는 문제이다.

핵심 개념

DBMS는 데이터를 통합 관리하여 중복을 줄이고 여러 사용자가 함께 사용할 수 있게 하며, 접근 권한 관리와 백업·회복 기능까지 제공하는 소프트웨어이다.

선지별 해설

① O: DBMS는 같은 데이터를 여러 파일에 중복 저장하지 않도록 통합 관리하므로 데이터 중복을 효율적으로 제어할 수 있다.
② X: 틀린 부분은 "여러 사용자의 데이터 공유를 허용하지 않는다"이다. DBMS의 대표 기능 중 하나가 여러 사용자의 동시 데이터 공유이므로, 올바르게는 "여러 사용자의 데이터 공유를 허용한다"이다.
③ O: DBMS는 사용자별 권한을 설정하여 허가되지 않은 접근을 막는 보안 통제 기능을 제공한다.
④ O: DBMS는 장애에 대비한 백업과 복구 기능을 제공하므로 하드웨어나 소프트웨어 고장 시 데이터베이스를 회복할 수 있다.

오답 포인트

발문이 "옳지 않은 것"이라는 점을 놓치면 공유 기능을 설명한 ②를 오히려 맞는 진술처럼 지나치기 쉽다. 이 문항은 DBMS의 대표 기능을 하나씩 나열해 두고 ②에서만 공유 기능을 부정형으로 뒤집어 놓은 함정이다.

NoteBase

다시 나오면 이렇게 판단

DBMS 관련 선지는 먼저 공유, 중복 제어, 보안, 백업·회복이라는 핵심 기능에 대입해 판단하면 된다. 이 핵심 기능을 부정하거나 반대로 서술한 선지는 오답 가능성이 높다.

암기 포인트

DBMS는 데이터를 공유하고, 중복을 줄이고, 권한을 통제하고, 장애 시 복구한다.

NoteBase

문 2. 외래키(foreign key)에 대한 설명으로 옳은 것은?

① 외래키는 중복값을 허용하지 않는다.

② 한 릴레이션의 외래키는 같은 릴레이션의 기본키를 참조할 수 있다.

③ 한 릴레이션의 외래키는 같은 릴레이션의 기본키의 일부가 될 수 없다.

④ 외래키의 도메인과 그 외래키가 참조하는 기본키의 도메인은 서로 다를 수 있다.

정답 2번

출제 포인트

이 문제는 외래키의 참조 대상, 값의 중복 가능성, 기본키와의 관계를 정확히 구별하는지를 묻는다.

핵심 개념

기본키는 릴레이션의 각 튜플을 유일하게 식별하는 속성 또는 속성 집합이다.
외래키는 다른 릴레이션이나 같은 릴레이션의 기본키를 참조하여 릴레이션 사이의 연결을 표현하는 속성 또는 속성 집합이다.
외래키는 참조 무결성을 위한 키이므로 값이 중복될 수 있고, 설계에 따라 기본키의 일부로도 사용될 수 있다.

선지별 해설

1번: X. 외래키 값은 여러 튜플에서 같은 참조 대상을 가리킬 수 있으므로 중복값이 허용된다. 틀린 표현은 "중복값을 허용하지 않는다"이다. 올바르게는 "외래키는 중복값을 가질 수 있다"이다.
2번: O. 외래키는 같은 릴레이션의 기본키를 참조할 수 있다. 조직도에서 사원 릴레이션의 상사번호가 같은 사원 릴레이션의 사원번호를 참조하는 자기 참조 외래키가 대표적이다.
3번: X. 외래키는 같은 릴레이션의 기본키 일부가 될 수 있다. 교차 릴레이션이나 약한 개체 릴레이션에서는 외래키 속성이 기본키를 구성하는 경우가 있다. 틀린 표현은 "기본키의 일부가 될 수 없다"이다. 올바르게는 "기본키의 일부가 될 수 있다"이다.
4번: X. 외래키와 그것이 참조하는 기본키는 같은 도메인, 즉 서로 호환되는 자료형과 값의 범위를 가져야 참조가 성립한다. 틀린 표현은 "서로 다를 수 있다"이다. 올바르게는 "서로 같아야 한다"이다.

오답 포인트

외래키를 키라는 말만 보고 기본키처럼 유일해야 한다고 오해하면 1번에 흔들린다. 또 자기 참조 외래키와 외래키의 기본키 포함 가능성을 함께 묻기 때문에 "다른 릴레이션만 참조한다" 또는 "기본키와 완전히 분리된다"라고 단정하면 2번과 3번을 동시에 틀리기 쉽다.

NoteBase

다시 나오면 이렇게 판단

외래키 문제는 먼저 참조 대상이 다른 릴레이션만인지 같은 릴레이션도 포함하는지 본다. 다음으로 유일성은 기본키의 성질인지 외래키의 성질인지 분리해서 본다. 마지막으로 외래키와 참조키의 도메인 일치, 기본키 포함 가능성을 확인하면 정답을 고를 수 있다.

암기 포인트

외래키는 참조용 키이므로 중복 가능, 자기 참조 가능, 기본키 포함 가능, 도메인은 참조키와 일치이다.

NoteBase

문 3. 인덱스에 대한 설명으로 옳지 않은 것은?

① B+-트리는 다단계 인덱스 구조이다.

② 기본키가 아닌 속성(필드)에도 인덱스를 생성할 수 있다.

③ 삽입과 삭제 연산이 빈번하게 발생하면 인덱스 유지를 위한 부하가 커진다.

④ 탐색하고자 하는 속성이 물리적으로 정렬되어 있지 않은 경우, 클러스터링 인덱스가 효율적이다.

정답 4번

출제 포인트

이 문제는 B\+-트리, 보조 인덱스, 인덱스 유지 비용, 클러스터링 인덱스의 적용 조건을 구별할 수 있는지를 묻는다.

핵심 개념

B\+-트리는 루트에서 리프까지 단계적으로 탐색하는 대표적 다단계 인덱스 구조이다.
클러스터링 인덱스는 레코드가 탐색 키 순서로 물리적으로 정렬되어 있을 때 범위 검색과 순차 접근에 유리한 인덱스이다.
보조 인덱스는 파일의 물리적 정렬 순서와 별개인 속성에도 만들 수 있지만, 삽입과 삭제가 많으면 인덱스 갱신 비용이 함께 증가한다.

선지별 해설

1번 O: B\+-트리는 상위 노드가 하위 노드를 가리키는 트리 구조이므로 여러 단계의 인덱스를 거쳐 탐색하는 다단계 인덱스 구조가 맞다.
2번 O: 인덱스는 기본키에만 한정되지 않는다. 검색이 자주 일어나는 비기본키 속성에도 보조 인덱스를 생성할 수 있다.
3번 O: 인덱스가 있으면 조회는 빨라질 수 있지만, 삽입과 삭제 때마다 인덱스 엔트리도 함께 수정해야 하므로 유지 부하가 커진다.
4번 X: 틀린 표현은 "물리적으로 정렬되어 있지 않은 경우, 클러스터링 인덱스가 효율적이다"이다. 클러스터링 인덱스는 데이터가 해당 속성 기준으로 물리적으로 정렬되어 있을 때 효율적이며, 정렬되어 있지 않다면 그 장점을 살리기 어렵다.

오답 포인트

클러스터링 인덱스를 "검색용 인덱스"라는 말만 보고 항상 효율적이라고 일반화하면 틀린다. 핵심은 인덱스가 아니라 데이터 파일의 물리적 정렬 여부이다.

NoteBase

다시 나오면 이렇게 판단

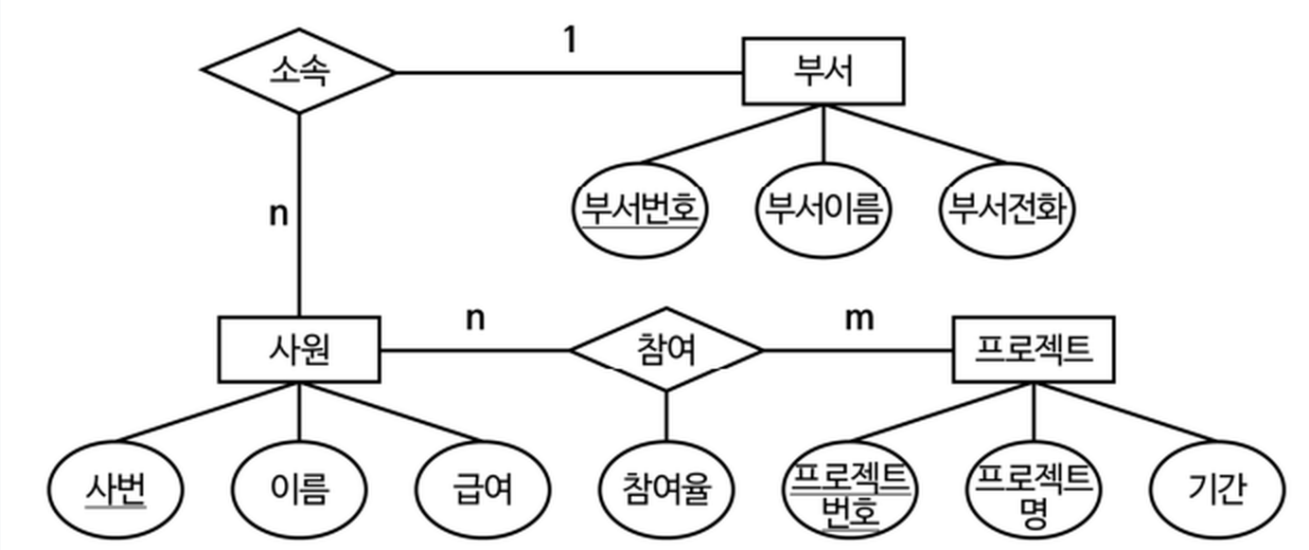
인덱스 문제는 먼저 해당 인덱스가 어떤 물리적 전제를 가지는지 본다. 클러스터링 인덱스가 나오면 데이터가 탐색 키 기준으로 실제 정렬되어 있는지를 먼저 확인하면 된다.

암기 포인트

클러스터링 인덱스는 물리적 정렬이 전제이고, 보조 인덱스는 정렬과 별개로 생성할 수 있다.

NoteBase

문 4. 다음 E-R 다이어그램을 관계 데이터 모델 스키마로 변환할 때, 도출된 릴레이션으로 옳지 않은 것은?



① 사원(사번, 이름, 급여, 부서번호)

② 부서(부서번호, 부서이름, 부서전화)

③ 참여(사번, 프로젝트번호, 참여율)

④ 프로젝트(프로젝트번호, 프로젝트명, 기간, 사번)

정답 4번

출제 포인트

이 문제는 E-R 다이어그램의 1:N 관계와 M:N 관계를 관계 스키마로 변환할 때 외래키를 어느 릴레이션에 두고 언제 별도 릴레이션을 만드는지를 묻는다.

핵심 개념

1:N 관계는 N쪽 릴레이션에 1쪽의 기본키를 외래키로 넣고, M:N 관계는 관계 자체를 별도 릴레이션으로 만들어 양쪽 기본키와 관계의 속성을 함께 둔다.

NoteBase

선지별 해설

1번: O. 소속은 부서 쪽이 1, 사원 쪽이 n인 1:N 관계이므로 N쪽인 사원 릴레이션에 부서의 기본키인 부서번호를 외래키로 둔다. 따라서 사원(사번, 이름, 급여, 부서번호)는 옳다.
2번: O. 부서는 강한 개체이며 속성이 부서번호, 부서이름, 부서전화로 보인다. 개체는 그대로 하나의 릴레이션이 되므로 부서(부서번호, 부서이름, 부서전화)는 옳다.
3번: O. 참여는 사원과 프로젝트 사이의 n:m 관계이고 관계 속성으로 참여율이 있다. 따라서 참여 자체를 별도 릴레이션으로 만들고 양쪽 기본키인 사번, 프로젝트번호와 관계 속성 참여율을 포함하는 참여(사번, 프로젝트번호, 참여율)가 된다.
4번: X. 프로젝트는 개체이므로 프로젝트 릴레이션에는 프로젝트의 속성만 들어가고, 사번은 프로젝트에 직접 붙지 않는다. 사번은 프로젝트와 사원의 M:N 관계를 표현하는 참여 릴레이션에 들어가야 한다. 올바르게는 프로젝트(프로젝트번호, 프로젝트명, 기간)과 참여(사번, 프로젝트번호, 참여율)로 나누어야 한다.

오답 포인트

참여가 M:N 관계인데도 이를 별도 릴레이션으로 만들지 않고 프로젝트 쪽에 사번을 붙이면 4번처럼 틀린다. 소속의 1:N 관계 처리와 참여의 M:N 관계 처리를 서로 섞는 함정을 노린 문항이다.

다시 나오면 이렇게 판단

먼저 관계의 카디널리티를 읽는다. 1:N이면 1쪽의 기본키를 N쪽에 외래키로 넣고, M:N이거나 관계 자체의 속성이 있으면 관계명을 별도 릴레이션으로 분리하여 양쪽 기본키와 관계 속성을 함께 적는다.

암기 포인트

1:N은 N쪽 외래키, M:N은 관계 릴레이션 생성이다.

NoteBase

문 5. 다음 상점 테이블과 상품 테이블을 생성하는 SQL 구문에 대한 설명으로 옳지 않은 것은?

```
CREATE TABLE 상점(
        ID              INTEGER,
        이름            VARCHAR(200),
        업종코드        INTEGER,
        주소            VARCHAR(200),
        PRIMARY KEY(ID));
CREATE TABLE 상품(
        ID              INTEGER,
        이름            VARCHAR(200),
        상점_ID         INTEGER,
        수량            INTEGER,
        단가            INTEGER,
        PRIMARY KEY(ID),
        FOREIGN KEY(상점_ID) REFERENCES 상점(ID));
```

① 상점 테이블의 ID 속성에 동일한 값을 갖는 두 개 이상의 투플이 존재할 수 없다.

② 상점 테이블과 상품 테이블에 대해 자연 조인(natural join) 연산을 수행할 수 있다.

③ 상품 테이블의 상점_ID 속성에 NULL 값이 아닌 상점 테이블의 ID 속성에 존재하지 않는 값을 저장할 수 있다.

④ 상점 테이블의 이름 속성과 주소 속성에 길이가 200 미만인 문자열을 입력하더라도, 두 속성에서 저장 공간의 낭비가 발생하지 않는다.

정답 3번

출제 포인트

이 문제는 기본 키, 외래 키, NATURAL JOIN, VARCHAR의 의미를 구분하는 문제이다.

핵심 개념

기본 키는 각 튜플을 유일하게 식별하는 속성이며 중복과 NULL을 허용하지 않는다.
외래 키는 다른 테이블의 기본 키 값을 참조하는 속성이며, NULL은 가능해도 존재하지 않는 부모 키 값은 저장할 수 없다.
NATURAL JOIN은 두 테이블에서 이름이 같은 모든 속성을 기준으로 자동 조인하는 연산이다.
VARCHAR(200)은 최대 길이가 200이라는 뜻의 가변 길이 문자열 타입이어서 실제 입력 길이에 맞춰 저장된다.

NoteBase

선지별 해설

① O
상점 테이블의 ID에는 PRIMARY KEY 제약이 걸려 있으므로 동일한 값을 가진 두 개 이상의 튜플이 존재할 수 없다. 기본 키는 유일성을 만족해야 한다.

② O
상점 테이블과 상품 테이블에는 이름이 같은 속성인 ID와 이름이 공통으로 있으므로 NATURAL JOIN 자체는 수행할 수 있다. 다만 이 경우 조인 기준은 상점.ID=상품.ID와 상점.이름=상품.이름이 되며, 상점_ID를 기준으로 자동 조인하는 것은 아니다.

③ X
틀린 부분은 "NULL 값이 아닌 상점 테이블의 ID 속성에 존재하지 않는 값을 저장할 수 있다"는 표현이다. 상품.상점_ID는 FOREIGN KEY로 상점.ID를 참조하므로 NULL이 아닌 값이라면 반드시 상점 테이블의 ID에 실제로 존재해야 한다.

④ O
VARCHAR(200)은 200자를 고정 저장하는 타입이 아니라 최대 200자까지 저장하는 가변 길이 타입이다. 따라서 이름과 주소에 200자 미만 문자열을 넣는다고 해서 CHAR처럼 남는 길이만큼 고정 공간을 낭비하는 구조는 아니다.

오답 포인트

②에서 NATURAL JOIN이 상점_ID를 기준으로 연결된다고 착각하거나, ④에서 VARCHAR를 CHAR와 같은 고정 길이 타입으로 오해하면 틀리기 쉽다. ③은 외래 키가 NULL을 허용할 수 있다는 점과 존재하지 않는 부모 키를 허용하지 않는다는 점을 섞어 놓은 함정이다.

다시 나오면 이렇게 판단

PRIMARY KEY가 보이면 중복과 NULL 금지를 먼저 확인하고, FOREIGN KEY가 보이면 참조 대상에 없는 비NULL 값 저장 가능 여부를 판단한다. NATURAL JOIN은 이름이 같은 속성 전체로 조인된다는 점을 확인하고, VARCHAR는 고정 길이인지 가변 길이인지부터 구분한다.

암기 포인트

PRIMARY KEY는 유일성, FOREIGN KEY는 참조 무결성, NATURAL JOIN은 동명 속성 자동 조인, VARCHAR는 가변 길이이다.

NoteBase

문 6. 다음 함수 종속 관계가 있는 릴레이션 '학생(학번, 과목번호, 이름, 학과이름, 학과전화번호, 성적)'에 대한 설명으로 옳지 않은 것은? (단, 학생 릴레이션은 이미 제1정규형을 만족한다)

학번 → {이름, 학과이름, 학과전화번호}
학과이름 → 학과전화번호
{학번, 과목번호} → 성적

① 함수적 종속관계 X→Y가 성립하더라도, Y→X가 항상 성립하는 것은 아니다.

② 학생 릴레이션을 분해하더라도 무손실분해(lossless decomposition)이어야 한다.

③ 학생 릴레이션이 제2정규형을 만족하도록 분해하면, (학번, 이름, 학과이름, 학과전화번호), (학번, 과목번호, 성적)이다.

④ 학생 릴레이션이 제3정규형을 만족하도록 분해하면, (학번, 이름), (학과이름, 학과전화번호), (학번, 과목번호, 성적)이다.

정답 4번

출제 포인트

이 문제는 함수 종속에 따른 부분 함수 종속과 이행 함수 종속을 구분하여 제2정규형과 제3정규형 분해를 판단하는 문제이다.

핵심 개념

부분 함수 종속은 기본키가 복합키일 때 기본키의 일부에만 속성이 종속되는 경우이다. 이 문제에서는 학번이 복합키 \학번, 과목번호\의 일부인데 이름, 학과이름, 학과전화번호가 학번에만 종속된다.
이행 함수 종속은 기본키가 아닌 속성을 거쳐 다른 속성이 종속되는 경우이다. 이 문제에서는 학번 → 학과이름, 학과이름 → 학과전화번호이므로 학번 → 학과전화번호가 이행적으로 성립한다.
제2정규형은 부분 함수 종속을 제거한 형태이고, 제3정규형은 이행 함수 종속까지 제거한 형태이다.

NoteBase

선지별 해설

① O
함수적 종속관계에서 X → Y가 성립해도 역으로 Y → X가 항상 성립하는 것은 아니다. 한 학생의 학번은 이름을 결정할 수 있어도 같은 이름이 여러 학생에게 나타날 수 있으므로 이름이 학번을 결정한다고 볼 수 없는 것과 같다.

② O
정규화로 릴레이션을 분해할 때는 원래 정보를 조인으로 복원할 수 있어야 하므로 무손실분해가 되어야 한다. 손실분해가 되면 조인 결과에 가짜 튜플이 생기거나 원래 정보가 사라질 수 있다.

③ O
제2정규형은 부분 함수 종속만 제거하면 된다. 이름, 학과이름, 학과전화번호는 학번에만 종속되므로 (학번, 이름, 학과이름, 학과전화번호)와 (학번, 과목번호, 성적)으로 나누면 부분 함수 종속이 제거된다.

④ X
틀린 부분은 (학번, 이름)으로만 분해한 표현이다. 제3정규형으로 분해하려면 이행 함수 종속인 학과이름 → 학과전화번호만 분리해야 하므로 학번이 결정하는 학과이름은 학생 정보 릴레이션에 남아 있어야 한다. 올바른 분해는 (학번, 이름, 학과이름), (학과이름, 학과전화번호), (학번, 과목번호, 성적)이다.

오답 포인트

④는 학과전화번호를 별도 릴레이션으로 분리한 점만 보고 맞다고 판단하기 쉽다. 그러나 제3정규형 분해에서는 이행 종속만 제거하는 것이므로 학번이 직접 결정하는 학과이름까지 삭제하면 안 된다는 점이 함정이다.

다시 나오면 이렇게 판단

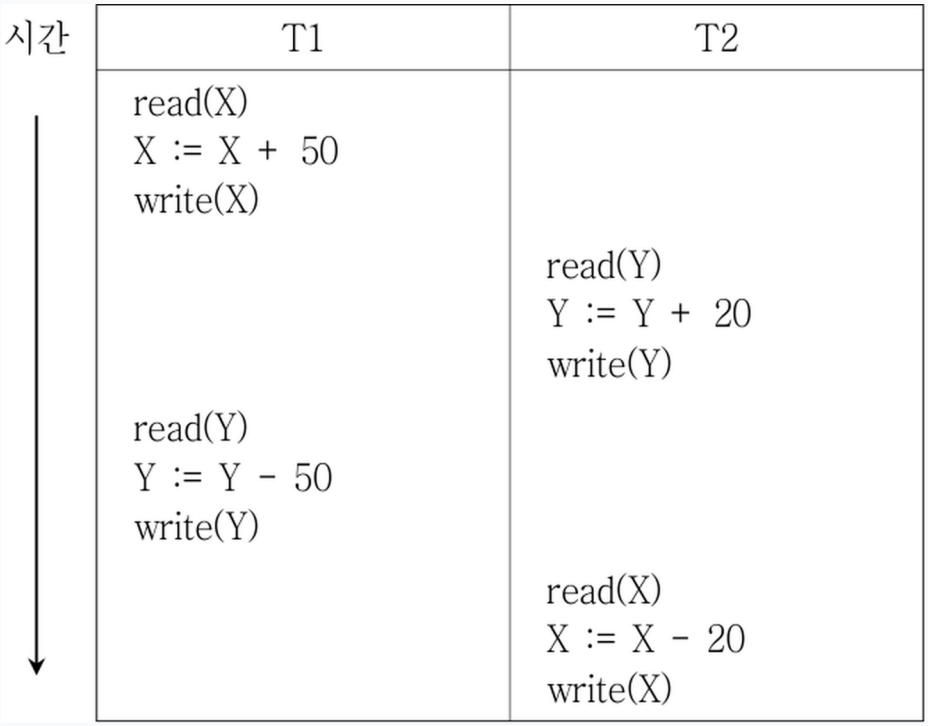
먼저 후보키를 찾고, 기본키 일부에만 종속되는 속성이 있으면 제2정규형 분해 대상으로 본다. 그다음 기본키가 아닌 속성이 또 다른 비주요 속성을 결정하면 그 부분만 따로 분리하여 제3정규형 여부를 판단한다.

암기 포인트

제2정규형은 부분 종속 제거, 제3정규형은 이행 종속 제거이다.

NoteBase

문 7. 은행계좌 X와 Y에 대한 트랜잭션 T1과 T2가 수행되는 다음 스케줄에 대한 설명으로 옳지 않은 것은? (단, 두 트랜잭션 수행 전 X계좌 잔액은 20 이상의 양수이고, Y계좌 잔액은 50 이상의 양수이다)



① 사이클을 포함한 스케줄이다.

② 2단계 로킹 규약(two-phase locking protocol)으로 구현할 수 있다.

③ 두 트랜잭션 수행 전과 수행 완료 후 X계좌와 Y계좌 잔액의 합(X+Y)은 같다.

④ 두 트랜잭션 수행 완료 후 X계좌 잔액이 180이면, 두 트랜잭션 수행 전 X계좌 잔액은 150이다.

정답 2번

출제 포인트

이 문제는 인터리빙된 트랜잭션 스케줄에서 충돌 직렬가능성, 2단계 로킹 가능 여부, 계좌 합 보존을 함께 판정하는 문제이다.

사례 정리

T1은 X에 50을 더한 뒤 Y에서 50을 빼므로 Y에서 X로 50을 이체한다.
T2는 Y에 20을 더한 뒤 X에서 20을 빼므로 X에서 Y로 20을 이체한다.
스케줄 순서는 T1의 X 처리 → T2의 Y 처리 → T1의 Y 처리 → T2의 X 처리이다.

NoteBase

기출OX 해설집 · 공무원/국가직/7급 · 데이터베이스론 · 2022 · 2차

핵심 개념

사이클이 있는 선행 그래프는 충돌 직렬가능하지 않다.
2단계 로킹은 락을 해제한 뒤 새로운 락을 다시 얻을 수 없으므로, 서로 다른 데이터 항목을 교차 접근하는 스케줄은 그대로 재현되지 않을 수 있다.
두 트랜잭션이 모두 단순 이체이므로 각 트랜잭션 자체는 X+Y의 합을 보존한다.

적용 기준

X에서의 충돌로 T1→T2, Y에서의 충돌로 T2→T1 간선이 생기는지를 먼저 본다.
그다음 각 트랜잭션이 X와 Y를 어떤 순서로 잠가야 하는지 따져 2단계 로킹으로 같은 인터리빙이 가능한지 판정한다.
마지막으로 각 트랜잭션의 순효과를 합산해 최종 잔액을 계산한다.

선지별 해설

① O
X에서는 T1이 write(X)를 먼저 하고 T2가 나중에 read(X), write(X)를 하므로 T1→T2 간선이 생긴다. Y에서는 T2가 write(Y)를 먼저 하고 T1이 나중에 read(Y), write(Y)를 하므로 T2→T1 간선이 생긴다. 선행 그래프에 사이클이 있으므로 맞는 설명이다.

② X
이 스케줄은 2단계 로킹으로 그대로 구현할 수 없다. T1이 먼저 X를 갱신한 뒤 나중에 Y를 읽고 써야 하고, T2는 먼저 Y를 갱신한 뒤 나중에 X를 읽고 써야 한다. 따라서 중간 시점에는 T1이 X 락을, T2가 Y 락을 잡은 상태가 되는데, 이후 각각 상대 항목의 락을 추가로 필요로 하므로 교착 상태가 되거나 같은 순서를 유지할 수 없다. 틀린 표현은 '2단계 로킹 규약으로 구현할 수 있다'이고, 올바르게는 '이 스케줄은 2단계 로킹으로는 그대로 구현할 수 없다'이다.

③ O
T1의 순효과는 X에 +50, Y에 -50이므로 합은 변하지 않는다. T2의 순효과는 Y에 +20, X에 -20이므로 역시 합은 변하지 않는다. 따라서 수행 전후의 X+Y는 같다.

④ O
최종 X는 T1 때문에 50 증가하고 T2 때문에 20 감소하므로 처음 X보다 30 크다. 따라서 수행 완료 후 X가 180이면 수행 전 X는 150이다.

오답 포인트

②는 '교착 상태가 생길 수 있다'와 '2단계 로킹으로 구현 가능하다'를 혼동하면 틀리기 쉽다. 사이클이 있다고 해서 항상 값의 합이 깨지는 것은 아니고, 직렬가능성 여부와 보존량 계산은 별개의 판단이라는 점도 함정이다.

NoteBase

다시 나오면 이렇게 판단

트랜잭션 스케줄 문제는 각 데이터 항목별 충돌 순서로 선행 그래프를 먼저 그린다. 그다음 락 획득 순서를 대입해 2단계 로킹 가능 여부를 보고, 마지막에 각 트랜잭션의 순증감만 합산하면 잔액 관련 선지를 빠르게 판정할 수 있다.

암기 포인트

X 충돌은 T1→T2, Y 충돌은 T2→T1이므로 사이클이 생긴다. 이체형 트랜잭션은 각 항목 증감만 더하면 X+Y 보존 여부를 바로 판정할 수 있다.

NoteBase

문 8. 200개의 트랜잭션에 대한 장바구니 분석을 통해 다음 조건을 만족하는 '기저귀 → 맥주' 연관 규칙(association rule)을 도출하였다. 이 규칙의 지지도(support)와 신뢰도(confidence)를 순서대로 바르게 나열한 것은?

맥주는 50개의 트랜잭션에서 구매되었다.
기저귀는 40개의 트랜잭션에서 구매되었다.
기저귀와 맥주가 동시에 구매된 트랜잭션 수는 20개이다.

① 지지도(%): 10, 신뢰도(%): 50
② 지지도(%): 40, 신뢰도(%): 45
③ 지지도(%): 45, 신뢰도(%): 40
④ 지지도(%): 50, 신뢰도(%): 10

정답 1번

출제 포인트

연관 규칙 기저귀 → 맥주의 지지도와 신뢰도를 구분해 계산하는 문제이다.

조건 정리

주어진 값: 전체 트랜잭션 200개, 맥주 구매 50개, 기저귀 구매 40개, 기저귀와 맥주를 동시에 구매한 트랜잭션 20개이다.
구해야 할 값: 규칙 기저귀 → 맥주의 지지도와 신뢰도이다.
주의할 조건: 지지도는 전체 거래 대비 동시 구매 비율이고, 신뢰도는 선행항인 기저귀 구매 거래 대비 동시 구매 비율이다.

사용할 공식

지지도는 기저귀와 맥주를 함께 산 거래 수전체 거래 수 × 100이다.
신뢰도는 기저귀와 맥주를 함께 산 거래 수기저귀를 산 거래 수 × 100이다.
이 공식을 쓰는 이유는 규칙 $X \rightarrow Y$에서 지지도는 X와 Y가 함께 나타난 비율을, 신뢰도는 X가 나타났을 때 Y도 함께 나타날 조건부 비율을 뜻하기 때문이다.

계산 과정

지지도는 $\frac{20}{200} \times 100 = 10\%$이다.
신뢰도는 $\frac{20}{40} \times 100 = 50\%$이다.
따라서 순서대로 나열하면 10%, 50%이다.

NoteBase

선지별 해설

① O: 지지도 10%, 신뢰도 50%로 계산 결과와 정확히 일치한다.
② X: 지지도 40%, 신뢰도 45%는 둘 다 계산 결과와 다르다. 지지도를 전체 200개가 아니라 다른 기준으로 잘못 잡았고, 신뢰도도 $\frac{20}{40}$이 아니라 임의의 비율로 잘못 계산한 값이다. 올바르게는 지지도 10%, 신뢰도 50%이다.
③ X: 지지도 45%, 신뢰도 40%는 지지도와 신뢰도를 모두 잘못 계산한 값이다. 특히 신뢰도를 후행항이나 다른 수치와 혼동한 흔적이 있다. 올바르게는 지지도 10%, 신뢰도 50%이다.
④ X: 지지도 50%, 신뢰도 10%는 두 값을 서로 뒤바꾸고 분모도 잘못 잡은 값이다. 50%는 신뢰도이고 10%는 지지도이다. 올바르게는 지지도 10%, 신뢰도 50%이다.

오답 포인트

이 문항의 함정은 지지도와 신뢰도의 분모가 다르다는 점이다. 지지도는 전체 트랜잭션 200개를 분모로 쓰고, 신뢰도는 선행항인 기저귀 구매 40개를 분모로 쓴다. 선행항과 전체 건수를 뒤섞으면 ④처럼 두 값을 뒤바꾼 오답에 걸리기 쉽다.

다시 나오면 이렇게 판단

연관 규칙 문제는 먼저 화살표의 왼쪽이 선행항인지 확인한 뒤, 지지도는 전체 대비 공동발생 비율로 계산하고 신뢰도는 선행항 대비 공동발생 비율로 계산하면 된다.

암기 포인트

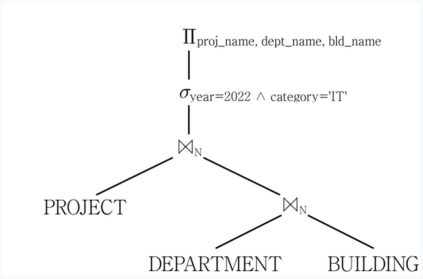
지지도는 전체 분모, 신뢰도는 선행항 분모이다.

NoteBase

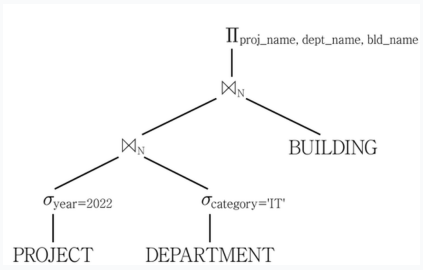
문 9. 다음 릴레이션에 대하여 질의를 수행하고자 한다. 질의 수행 계획(query evaluation plan)의 의미가 다른 것은?

PROJECT (proj_id, proj_name, dept_id, year)
DEPARTMENT (dept_id, dept_name, bld_id, category)
BUILDING (bld_id, bld_name, location, capacity)

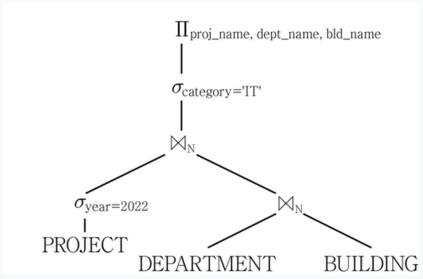
①



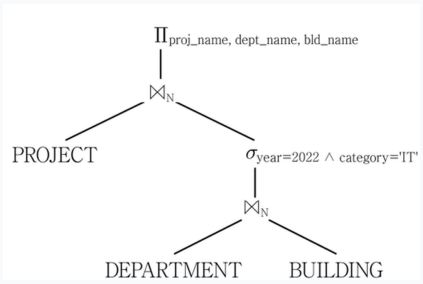
②



③



④



정답 4번

출제 포인트

이 문제는 질의 수행 계획에서 선택 연산의 푸시다운 가능 범위와 조인 트리의 등가성을 판단할 수 있는지를 묻는다.

NoteBase

선지별 해설

1번: X. PROJECT, DEPARTMENT, BUILDING을 먼저 조인한 뒤 `year=2022`와 `category='IT'`를 한꺼번에 적용한 형태이다. 두 조건이 각각 PROJECT와 DEPARTMENT의 속성만 사용하므로 전체 조인 뒤에 적용해도 원래 질의와 의미가 같다.
2번: X. `σ_year=2022`를 PROJECT에, `σ_category='IT'`를 DEPARTMENT에 미리 내려 보낸 뒤 조인한 형태이다. 선택 연산 푸시다운이 성립하므로 원래 질의와 의미가 같다.
3번: X. `σ_year=2022`는 PROJECT에 먼저 적용하고, `σ_category='IT'`는 DEPARTMENT와 BUILDING의 조인 결과에 적용한 형태이다. `category`는 DEPARTMENT의 속성이므로 DEPARTMENT가 포함된 조인 결과에 적용해도 의미가 유지된다.
4번: O. `σ_year=2022 ∧ category='IT'`를 DEPARTMENT와 BUILDING의 조인 결과에 적용한 점이 다르다. `year`는 PROJECT의 속성이므로 이 부분트리에서는 참조할 수 없다. 올바르게는 `σ_year=2022`를 PROJECT에 적용하거나, 세 릴레이션을 조인한 뒤 전체 결과에 선택 조건을 적용해야 한다.

오답 포인트

선택 연산은 조건에 등장한 속성이 그 중간 결과에 모두 존재할 때만 아래로 내릴 수 있다. 조건이 단순하다는 이유만으로 임의의 서브트리에 미는 순간 질의 의미가 달라진다.

다시 나오면 이렇게 판단

각 선택 조건이 어느 릴레이션의 속성을 참조하는지 먼저 표시한 뒤, 그 속성이 모두 살아 있는 노드까지만 푸시다운이 가능하다고 판단한다.

암기 포인트

`σ`는 조건 속성이 존재하는 곳까지만 내려가고, 조인 순서는 바뀌어도 조건의 소속 릴레이션은 바뀌지 않는다.

NoteBase

문 10. 릴레이션 $R(B, C, G, H, I, S)$이 함수 종속성 집합 F를 만족할 때, R의 수퍼키(superkey)로 옳은 것은?

$F = \{B \rightarrow H, CG \rightarrow H, CG \rightarrow I, S \rightarrow B, S \rightarrow C\}$

① H
② S
③ CG
④ GS

정답 4번

출제 포인트

이 문제는 함수 종속 집합으로부터 속성 폐포를 계산하여 어떤 속성집합이 슈퍼키인지 판정하는 문제이다.

조건 정리

릴레이션은 R(B, C, G, H, I, S)이다.
주어진 함수 종속은 $B \rightarrow H, CG \rightarrow H, CG \rightarrow I, S \rightarrow B, S \rightarrow C$이다.
슈퍼키는 어떤 속성집합 X의 폐포 X^+가 R의 모든 속성을 포함할 때 성립한다.

사용할 공식

슈퍼키 판정 기준은 $X^+ = R$이다.
이 기준을 쓰는 이유는 시작 속성집합으로부터 함수 종속을 반복 적용해 모든 속성을 유도할 수 있어야 전체 튜플을 유일하게 식별할 수 있기 때문이다.

계산 과정

1. $H^+ = \{H\}$이므로 H만으로는 다른 속성을 전혀 유도하지 못한다.
2. $S^+ = \{S, B, C, H\}$이다. $S \rightarrow B, S \rightarrow C, B \rightarrow H$까지는 나오지만 G와 I는 나오지 않는다.
3. $(CG)^+ = \{C, G, H, I\}$이다. $CG \rightarrow H, CG \rightarrow I$는 가능하지만 S와 B는 나오지 않는다.
4. $(GS)^+$를 구하면 먼저 S로부터 B와 C가 나오므로 $\{G, S, B, C\}$가 된다. 이어서 C와 G가 함께 있으므로 $CG \rightarrow H, CG \rightarrow I$를 적용해 H와 I도 나온다. 따라서 $(GS)^+ = \{G, S, B, C, H, I\} = R$이다.

NoteBase

선지별 해설

① X

H의 폐포는 $H^+ = \{H\}$이다. H 단독으로는 다른 어떤 속성도 결정하지 못하므로 슈퍼키가 아니다. 틀린 표현은 H가 전체 속성을 결정한다는 판단이며, 올바르게는 H는 키가 아니고 R 전체를 결정하지 못한다.

② X

S의 폐포는 $S^+ = \{S, B, C, H\}$이다. B, C, H까지는 얻지만 G가 없어서 $CG \rightarrow I$를 적용할 수 없고 I도 결정하지 못한다. 틀린 표현은 S만으로 모든 속성을 결정한다는 판단이며, 올바르게는 S는 G와 I를 결정하지 못하므로 슈퍼키가 아니다.

③ X

CG의 폐포는 $(CG)^+ = \{C, G, H, I\}$이다. H와 I는 얻지만 S와 B는 나오지 않으므로 전체 속성을 포함하지 못한다. 틀린 표현은 CG가 충분한 결정자라는 판단이며, 올바르게는 CG는 S와 B를 결정하지 못해 슈퍼키가 아니다.

④ O

GS의 폐포는 $(GS)^+ = \{G, S, B, C, H, I\} = R$이다. S로부터 B와 C를 얻고, 얻어진 C와 기존 G를 묶어 H와 I까지 유도할 수 있으므로 GS는 슈퍼키이다.

오답 포인트

S에서 B와 C가 나오므로 ②를 성급하게 고르기 쉽다. 그러나 슈퍼키 판단은 일부 속성이 아니라 전체 속성 도달 여부로 결정되며, 이 문제에서는 G가 출발 집합에 없으면 I까지 연결되지 않는다는 점이 함정이다.

다시 나오면 이렇게 판단

슈퍼키 문제는 먼저 기준을 $X^+ = R$로 고정한 뒤, 출발 집합에서 새로 얻는 속성을 한 번씩만 적어 가며 폐포를 끝까지 확장하면 된다. 중간에 핵심 연결 속성이 빠지면 전체 속성에 도달할 수 없는지 바로 확인할 수 있다.

암기 포인트

슈퍼키는 폐포가 R 전체가 되는 집합이며, 이 문제에서는 S가 C를 만들고 G와 합쳐져 $CG \rightarrow H, I$가 완성된다는 흐름이 핵심이다.

NoteBase

문 11. 관계 데이터베이스 관리 시스템(RDBMS)과 NoSQL에 대한 설명으로 옳은 것은?

① RDBMS는 NoSQL보다 약한 스키마를 요구한다.

② NoSQL은 RDBMS보다 엄격한 일관성 모델을 보장한다.

③ NoSQL은 RDBMS보다 정형 데이터를 저장하기에 적합하다.

④ NoSQL 데이터 모델로 키-값(key-value), 문서 기반(document-based), 그래프 기반(graph-based) 모델이 있다.

정답 4번

출제 포인트

이 문제는 RDBMS와 NoSQL의 핵심 차이인 스키마 엄격성, 일관성 특성, 적합한 데이터 형태, 대표 데이터 모델을 구별하는 문제이다.

핵심 개념

RDBMS는 테이블 간 관계를 바탕으로 정형 데이터를 관리하는 관계형 데이터베이스 관리 시스템으로, 미리 정의한 스키마와 강한 일관성이 핵심이다.
NoSQL은 비관계형 데이터 저장 방식의 묶음으로, 대규모 분산 환경과 유연한 스키마에 강점이 있고 key-value, document-based, graph-based 같은 다양한 모델을 포함한다.
구별 축은 정형 데이터 중심의 엄격한 구조인지, 비정형·반정형 데이터까지 유연하게 다루는 분산 지향 구조인지이다.

선지별 해설

① X: 틀린 부분은 'RDBMS는 NoSQL보다 약한 스키마를 요구한다'는 표현이다. RDBMS는 테이블 구조와 속성 타입을 사전에 엄격하게 정의하는 편이다. 올바르게는 'NoSQL이 RDBMS보다 상대적으로 유연한 스키마를 허용한다'이다.
② X: 틀린 부분은 'NoSQL은 RDBMS보다 엄격한 일관성 모델을 보장한다'는 표현이다. 일반적으로 RDBMS는 ACID 특성을 바탕으로 강한 일관성을 제공하고, NoSQL은 확장성과 가용성을 위해 일관성을 완화하는 경우가 많다. 올바르게는 'RDBMS가 NoSQL보다 일반적으로 더 엄격한 일관성 모델을 제공한다'이다.
③ X: 틀린 부분은 'NoSQL은 RDBMS보다 정형 데이터를 저장하기에 적합하다'는 표현이다. 정형 데이터와 복잡한 관계를 안정적으로 다루는 데에는 RDBMS가 더 적합하다. 올바르게는 'NoSQL은 비정형 또는 반정형 데이터와 대규모 분산 저장에 더 적합한 경우가 많다'이다.
④ O: NoSQL의 대표적 데이터 모델에는 key-value, document-based, graph-based가 있다. 이 선택지는 NoSQL의 분류 예시를 정확하게 제시하므로 옳다.

NoteBase

오답 포인트

이 문항의 함정은 RDBMS와 NoSQL의 특징을 서로 뒤바꾸어 제시한 데 있다. 특히 ①의 '약한 스키마', ②의 '엄격한 일관성', ③의 '정형 데이터 적합'은 모두 RDBMS 쪽 성격과 NoSQL 쪽 성격을 바꿔 놓은 표현이다.

다시 나오면 이렇게 판단

RDBMS가 나오면 관계형 테이블, 엄격한 스키마, 강한 일관성, 정형 데이터 적합성을 먼저 떠올리면 된다. NoSQL이 나오면 유연한 스키마, 분산 처리, 비정형·반정형 데이터, 다양한 데이터 모델을 연결해 판단하면 된다.

암기 포인트

RDBMS는 정형·관계·엄격, NoSQL은 유연·분산·다양한 모델이다.

NoteBase

문 12. 관계 데이터베이스에서 NULL 값에 대한 설명으로 옳지 않은 것은?

① 한 개 이상의 NULL 값을 포함한 산술 연산 결과는 NULL이다.

② 릴레이션 투플의 삽입 연산 수행 시, 결과 릴레이션의 속성은 NULL 값을 가질 수 있다.

③ NULL 값과 다른 값에 대해 대소 비교 연산자 '<'를 수행한 결과는 참(true) 혹은 거짓(false)이다.

④ NULL 값을 가지지 않은 속성들만으로 구성된 두 릴레이션에 대해 외부 조인(outer join) 연산을 수행한 경우, 결과 릴레이션의 속성은 NULL 값을 가질 수 있다.

정답 3번

출제 포인트

이 문제는 관계 데이터베이스에서 NULL의 의미와 NULL이 산술 연산, 비교 연산, 외부 조인 결과에 어떻게 작용하는지를 구별할 수 있는지를 묻는다.

핵심 개념

NULL은 0이나 공백이 아니라 값이 없거나 알 수 없음을 나타내는 특수 표시이다.
산술 연산에서 피연산자 중 하나가 NULL이면 결과도 보통 NULL이 된다.
일반 비교 연산자 =, <, >로 NULL을 비교하면 참이나 거짓이 아니라 미정 값으로 처리된다.
외부 조인은 매칭되지 않은 튜플의 속성을 채우기 위해 결과 릴레이션에 NULL을 만들어 낼 수 있다.

선지별 해설

1번: O. 산술 연산에 NULL이 포함되면 결과값을 확정할 수 없으므로 결과는 NULL로 처리된다.
2번: O. 릴레이션에 튜플을 삽입할 때 특정 속성값이 주어지지 않았거나 알 수 없으면 그 속성은 NULL 값을 가질 수 있다.
3번: X. 틀린 부분은 "참(true) 혹은 거짓(false)이다"라는 표현이다. NULL과 다른 값을 일반 대소 비교 연산자 `<`로 비교한 결과는 참이나 거짓으로 확정되지 않고 unknown으로 처리된다. 올바르게는 "NULL 값과 다른 값에 대해 대소 비교 연산자 `<`를 수행한 결과는 unknown이다"이다.
4번: O. 두 릴레이션의 원래 속성들이 NULL을 갖지 않더라도 외부 조인은 매칭되지 않은 쪽의 속성 자리를 NULL로 채우므로 결과 릴레이션의 속성이 NULL 값을 가질 수 있다.

오답 포인트

NULL을 0이나 빈값처럼 일반 값으로 생각하면 3번처럼 비교 결과도 참 또는 거짓이라고 착각하기 쉽다. 이 문항의 함정은 NULL이 포함된 비교는 2값 논리가 아니라 3값 논리로 처리된다는 점이다.

NoteBase

다시 나오면 이렇게 판단

NULL이 보이면 먼저 일반 값이 아니라는 점을 떠올린다. 그다음 산술 연산인지, 비교 연산인지, 조인 결과인지 구분하면 산술은 결과가 NULL로 흐르고, 비교는 unknown으로 흐르며, 외부 조인은 결과에 NULL을 만들어 낼 수 있다고 판단할 수 있다.

암기 포인트

NULL은 값이 없음을 뜻하고, 산술 결과는 NULL, 비교 결과는 unknown, 외부 조인 결과에는 NULL이 생길 수 있다.

NoteBase

문 13. 다음 중 데이터베이스 관리자(DBA)의 역할만을 모두 고르면?

ㄱ. 데이터베이스 스키마 정의
ㄴ. 저장 구조에 대한 접근 방식 결정
ㄷ. 응용 프로그램 개발
ㄹ. 데이터베이스 시스템 모니터링 및 성능 분석

① ㄱ, ㄷ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄹ

정답 4번

출제 포인트

이 문제는 데이터베이스 관리자(DBA)의 역할과 응용 프로그램 개발자의 역할을 구분할 수 있는지를 묻는 문제이다.

핵심 개념

DBA는 데이터베이스의 스키마 정의, 저장 구조와 접근 방식 결정, 권한 관리, 성능 모니터링처럼 데이터베이스 운영 전반을 관리하는 사람이다.
응용 프로그램 개발은 데이터베이스를 사용하는 프로그램을 작성하는 업무로, 일반적으로 DBA의 직접 역할이 아니라 응용 프로그래머의 역할이다.

NoteBase

선지별 해설

① X
ㄱ은 DBA의 역할이 맞지만, ㄷ이 틀렸다. 응용 프로그램 개발은 DBA의 역할이 아니라 응용 프로그램 개발자의 역할이다. 올바르게는 ㄱ, ㄴ, ㄹ이다.

② X
ㄱ과 ㄹ은 맞지만, ㄴ이 빠져 있어서 틀렸다. 저장 구조에 대한 접근 방식 결정은 DBA가 담당하는 핵심 역할이다. 올바르게는 ㄱ, ㄴ, ㄹ이다.

③ X
ㄴ은 맞지만, ㄷ을 포함한 점이 틀렸다. 또한 ㄱ과 ㄹ이 빠져 있어서 완전한 답이 아니다. 올바르게는 ㄱ, ㄴ, ㄹ이다.

④ O
ㄱ의 데이터베이스 스키마 정의, ㄴ의 저장 구조에 대한 접근 방식 결정, ㄹ의 데이터베이스 시스템 모니터링 및 성능 분석은 모두 DBA의 역할이므로 옳다.

오답 포인트

응용 프로그램 개발도 데이터베이스와 관련된 작업이므로 ㄷ을 DBA 역할로 착각하기 쉽다. 그러나 DBA는 데이터베이스 자체의 구조와 운영을 관리하고, 응용 프로그램 개발은 그 데이터베이스를 활용하는 프로그램을 만드는 업무라는 점이 핵심 함정이다.

다시 나오면 이렇게 판단

선지를 볼 때 데이터베이스 자체를 설계·운영·통제하는 일인지, 데이터베이스를 사용하는 응용 프로그램을 만드는 일인지를 먼저 구분하면 된다. 전자는 DBA 역할이고 후자는 응용 프로그램 개발자의 역할이다.

암기 포인트

DBA는 스키마·저장 구조·성능 관리이고, 응용 프로그램 개발은 DBA 업무가 아니다.

NoteBase

문 14. 관계 데이터 모델의 릴레이션에 대한 설명으로 옳지 않은 것은?

① 릴레이션 R에 포함된 투플들 사이에는 순서가 없다.

② 릴레이션 R의 모든 투플은 서로 다른 값을 가지고 있다.

③ 릴레이션의 인스턴스는 시간에 따라 변하지 않는 정적인 성질을 가지고 있다.

④ 릴레이션 스키마에 정의된 속성의 전체 개수를 릴레이션의 차수(degree)라고 한다.

정답 3번

출제 포인트

이 문제는 릴레이션의 집합적 성질과 릴레이션 스키마, 릴레이션 인스턴스의 차이를 구별할 수 있는지를 묻는다.

핵심 개념

릴레이션 스키마는 속성들의 이름과 구조를 정의한 틀이고, 릴레이션 인스턴스는 특정 시점에 실제로 저장된 튜플들의 집합이다.
릴레이션은 집합이므로 튜플의 순서가 없고 중복 튜플을 허용하지 않는다.
릴레이션의 차수는 스키마에 정의된 속성 수이다.

선지별 해설

1번: O. 릴레이션은 튜플들의 집합이므로 집합의 성질에 따라 튜플 사이에 순서가 없다. 조회 결과에서 보이는 정렬 순서는 저장 구조가 아니라 출력 방식의 문제이다.
2번: O. 릴레이션은 중복 튜플을 허용하지 않으므로 모든 튜플은 서로 구별된다. 다만 이것은 튜플 전체가 중복될 수 없다는 뜻이지, 각 속성값이 모두 달라야 한다는 뜻은 아니다.
3번: X. 틀린 부분은 '릴레이션의 인스턴스는 시간에 따라 변하지 않는 정적인 성질'이라는 표현이다. 실제로는 릴레이션 인스턴스가 삽입, 삭제, 갱신에 따라 시간에 따라 변하는 동적인 상태이고, 비교적 변하지 않는 것은 릴레이션 스키마이다.
4번: O. 릴레이션 스키마에 정의된 속성의 전체 개수를 릴레이션의 차수 degree라고 한다. 행의 개수는 카디널리티이고 차수와 구별된다.

오답 포인트

릴레이션 스키마와 릴레이션 인스턴스를 뒤바꾸면 3번에서 틀리기 쉽다. 또 2번의 '서로 다른 값'을 각 속성값이 모두 달라야 한다는 뜻으로 과하게 해석하면 중복 튜플 금지라는 핵심을 놓치게 된다.

다시 나오면 이렇게 판단

릴레이션 관련 선지는 먼저 스키마인지 인스턴스인지 구분한 뒤, 집합의 성질 순서 없음과 중복 없음, 그리고 차수와 카디널리티의 구별을 대입하면 빠르게 판단된다.

NoteBase

암기 포인트

스키마는 틀, 인스턴스는 현재 데이터, 차수는 열 수, 카디널리티는 행 수이다.

문 15. 데이터베이스 회복에 대한 설명으로 옳은 것은?

① 지연 갱신 회복기법에서는 UNDO 연산이 필요 없다.

② REDO 연산을 수행할 때는 로그(log)가 필요 없다.

③ 검사점(checkpoint) 연산은 장애가 발생할 때 실행된다.

④ 즉시 갱신 회복기법에서는 트랜잭션이 완료한 이후 데이터베이스를 갱신한다.

정답 1번

출제 포인트

이 문제는 지연 갱신과 즉시 갱신의 차이, 그리고 로그와 체크포인트의 역할을 구별할 수 있는지를 묻는다.

핵심 개념

지연 갱신은 트랜잭션이 완료되기 전에는 데이터베이스에 실제 갱신을 반영하지 않는 회복 기법이다. 따라서 미완료 트랜잭션을 되돌리는 UNDO가 필요 없다.
즉시 갱신은 트랜잭션 완료 전에도 데이터베이스에 갱신이 반영될 수 있는 회복 기법이다. 따라서 장애 시 UNDO와 REDO가 모두 필요할 수 있다.
로그는 갱신 내역을 기록해 UNDO와 REDO의 근거가 되는 정보이다.
체크포인트는 정상 수행 중 일정 시점의 회복 기준점을 남겨 장애 복구 범위를 줄이는 연산이다.

선지별 해설

1번: O. 지연 갱신 회복기법에서는 트랜잭션이 완료되기 전의 변경 내용이 데이터베이스에 반영되지 않는다. 따라서 장애가 발생해도 미완료 트랜잭션이 데이터베이스를 더럽히지 않으므로 UNDO 연산이 필요 없다.
2번: X. 틀린 부분은 "로그가 필요 없다"이다. REDO는 장애 이전에 완료된 트랜잭션의 갱신 내용을 다시 반영하는 연산이므로 어떤 연산을 재실행해야 하는지 판단하려면 로그가 필요하다.
3번: X. 틀린 부분은 "장애가 발생할 때 실행된다"이다. 체크포인트는 장애가 난 뒤에 실행하는 연산이 아니라 정상 운영 중 주기적으로 실행하여 회복 시점의 기준을 남기는 연산이다.
4번: X. 틀린 부분은 "트랜잭션이 완료한 이후 데이터베이스를 갱신한다"이다. 즉시 갱신은 트랜잭션 완료 이전에도 데이터베이스를 갱신할 수 있는 방식이다. 트랜잭션 완료 이후에만 반영하는 것은 지연 갱신의 특징이다.

오답 포인트

지연 갱신과 즉시 갱신의 반영 시점을 뒤바꾸면 1번과 4번을 동시에 헷갈리기 쉽다. 또한 체크포인트를 장애 발생 후 조치로 오해하면 3번에 끌리기 쉽다.

NoteBase

다시 나오면 이렇게 판단

먼저 데이터베이스 반영 시점이 완료 전인지 완료 후인지를 본다. 완료 후 반영이면 지연 갱신이므로 UNDO가 불필요하고, 완료 전 반영이면 즉시 갱신이므로 로그 기반의 UNDO와 REDO 가능성을 함께 떠올리면 된다. 체크포인트는 장애 후 조치가 아니라 평상시 회복 기준점이라는 점으로 판단한다.

암기 포인트

지연 갱신은 완료 후 반영이라 UNDO가 없고, 즉시 갱신은 완료 전 반영이라 로그와 UNDO/REDO를 함께 생각한다.

NoteBase

문 16. 다음 Student, Course, Enrollment 테이블에 대한 SQL 구문들을 수행하였을 때, COUNT(*) 값은? (단, Enrollment 테이블 생성 시 sno 속성과 cno 속성은 각각 Student 테이블과 Course 테이블 기본키의 외래키이며, 두 외래키에 모두 ON DELETE CASCADE 조건이 적용된다)

Student

sno	sname	year	dept
100	John	4	Com
200	David	3	EE
300	Tom	1	Com
400	Mark	4	Com
500	Simon	2	ME

Course

cno	cname	credit	dept	professor
C123	Java	3	Com	Andy
C312	Data Structure	3	Com	Joseph
C324	File System	3	Com	Helen
C413	Database	3	Com	Luna
E412	OS	3	EE	Elsa

NoteBase

Enrollment

sno	cno	grade	midterm	final
200	C123	B	85	80
300	C312	A	90	95
400	C312	A	90	95
500	C312	B	85	80
300	C324	C	75	75
400	C324	A	95	90
100	C413	A	90	95
300	C413	A	95	90
400	C413	B	80	85
100	E412	A	95	95
400	E412	C	65	75

SQL 구문

```
DELETE FROM Student WHERE sname= 'Simon';
DELETE FROM Course WHERE cname= 'OS';
SELECT COUNT(*) FROM Enrollment;
```

① 6

② 7

③ 8

④ 9

정답 3번

출제 포인트

이 문제는 ON DELETE CASCADE가 걸린 외래키에서 부모 행을 삭제할 때 Enrollment 테이블의 행 수가 얼마나 줄어드는지를 표 기준으로 계산하는 문제이다.

NoteBase

자료 읽는 순서

1. Enrollment의 현재 행 수를 먼저 센다. 현재 행 수는 11행이다.
2. Student에서 sname이 Simon인 학생의 sno를 찾는다. sno는 500이고, Enrollment에서 sno=500인 행은 (500, C312) 1행이다.
3. Course에서 cname이 OS인 과목의 cno를 찾는다. cno는 E412이고, Enrollment에서 cno=E412인 행은 (100, E412), (400, E412) 2행이다.
4. 두 삭제는 서로 겹치지 않으므로 총 3행이 연쇄 삭제되고, 11-3=8이 된다.

선지별 해설

1번(X): COUNT(*)를 6으로 보려면 Enrollment에서 5행이 삭제되어야 한다. 실제 삭제는 sno=500인 1행과 cno=E412인 2행뿐이므로 총 3행만 삭제된다. 틀린 표현은 결과를 6으로 본 점이며, 올바르게는 COUNT(*) 값이 8이다.
2번(X): 7이 되려면 4행이 삭제되어야 한다. 그러나 Student 삭제로 1행, Course 삭제로 2행이 삭제되고 두 집합이 겹치지 않아 총 3행만 사라진다. 틀린 표현은 삭제 행 수를 4행으로 본 점이며, 올바르게는 3행 삭제 후 8행이 남는다.
3번(O): Enrollment는 처음 11행이다. Simon 삭제로 (500, C312) 1행이 연쇄 삭제되고, OS 삭제로 (100, E412), (400, E412) 2행이 연쇄 삭제되어 총 3행이 사라지므로 COUNT(*)는 8이다.
4번(X): 9가 되려면 삭제가 2행만 일어나야 한다. 하지만 OS 과목 삭제만으로도 2행이 지워지고, 여기에 Simon의 수강 1행이 추가로 삭제된다. 틀린 표현은 최종 행 수를 9로 본 점이며, 올바르게는 8이다.

오답 포인트

Student 삭제와 Course 삭제를 따로 보다가 한쪽 연쇄 삭제를 빠뜨리거나, OS 삭제로 사라지는 2행만 세고 Simon의 1행을 놓치면 오답이 된다. 반대로 같은 행이 중복 삭제된다고 착각해도 틀린다. 이 문항에서는 (500, C312)와 E412 관련 2행이 서로 다른 행이다.

다시 나오면 이렇게 판단

외래키에 ON DELETE CASCADE가 나오면 먼저 자식 테이블의 현재 행 수를 세고, 각 DELETE문마다 어떤 부모키가 지워지는지 찾은 뒤 자식 테이블에서 대응 행을 표시한다. 마지막에는 중복 여부를 확인한 후 전체에서 한 번만 빼면 된다.

암기 포인트

CASCADE 문항은 부모 삭제 키를 찾고, 자식 테이블에서 해당 키 행 수를 센 뒤, 겹침 없이 총 삭제 행 수를 빼면 된다.

NoteBase

문 17. 다음 직원 테이블과 부서 테이블로 구성된 데이터베이스에서 '기본 인덱스를 사용하여 여러 개의 레코드를 검색하는 방식'으로 구현되는 SELECT문으로 옳은 것은? (단, 직원 테이블의 부서번호 속성은 부서 테이블 기본키의 외래키이다)

◯ 직원(직원번호, 이름, 봉급, 부서번호)
◯ 부서(부서번호, 부서명, 관리자)

① `SELECT * FROM 직원 WHERE 부서번호 = 5;`

② `SELECT * FROM 부서 WHERE 부서번호 > 5;`

③ `SELECT * FROM 직원 WHERE 직원번호 = 100;`

④ `SELECT * FROM 직원 WHERE 부서번호 = 5 AND 봉급 > 30000;`

정답 2번

출제 포인트

기본 인덱스는 기본키 순으로 정렬된 파일에서 시작 위치를 찾은 뒤 범위 조건에 맞는 여러 레코드를 연속해서 읽는 데 강하다는 점을 묻는 문제이다.

핵심 개념

기본 인덱스는 파일이 기본키 기준으로 정렬되어 있을 때 사용하는 희소 인덱스이다.
기본키에 대한 동등 조건은 보통 한 레코드를 찾는 데 쓰이고, 기본키에 대한 범위 조건은 시작 지점을 찾은 뒤 뒤쪽 레코드를 순차적으로 읽어 여러 개를 검색한다.
외래키나 중복값이 나올 수 있는 비정렬 속성 검색은 기본 인덱스 검색 방식과 다르다.

NoteBase

선지별 해설

① X. `직원 WHERE 부서번호 = 5`는 직원 테이블의 부서번호로 찾는 조건이다. 문제에서 직원 테이블의 부서번호는 부서 테이블 기본키의 외래키라고 했으므로 직원 테이블에서는 기본 인덱스 대상 기본키가 아니다. 틀린 표현은 `직원 WHERE 부서번호 = 5`이다. 기본 인덱스로 여러 레코드를 찾는 형태로 바꾸려면 기본키로 정렬된 테이블의 기본키 범위 조건이어야 한다.
② O. 부서 테이블의 부서번호는 기본키이므로 기본 인덱스의 대상이 된다. `부서번호 > 5`는 범위 조건이어서 인덱스로 시작 위치를 찾은 뒤 그 이후의 여러 부서 레코드를 연속해서 읽는 방식으로 구현된다.
③ X. `직원번호 = 100`은 직원번호가 기본키라고 보더라도 동등 조건이므로 한 레코드를 찾는 검색이다. 틀린 표현은 `직원번호 = 100`이다. 여러 레코드를 기본 인덱스로 찾으려면 `직원번호 >= 100`처럼 기본키 범위 조건이어야 한다.
④ X. `직원 WHERE 부서번호 = 5 AND 봉급 > 30000`은 직원 테이블의 외래키 부서번호와 비키 속성 봉급을 함께 거는 조건이다. 기본 인덱스로 여러 레코드를 찾는 전형적 형태가 아니다. 틀린 표현은 `부서번호 = 5 AND 봉급 > 30000`이다. 기본 인덱스 검색으로 맞추려면 기본키가 있는 테이블에서 그 기본키에 범위 조건을 걸어야 한다.

오답 포인트

`여러 개의 레코드`라는 말을 보고 ①이나 ④처럼 결과가 여러 행일 수 있는 조건을 고르기 쉽다. 그러나 이 문제의 핵심은 결과 행 수가 아니라 `기본 인덱스`로 구현되는가이다. ①과 ④는 직원 테이블의 외래키 부서번호를 사용하므로 기본 인덱스 범위 검색과 연결되지 않고, ③은 기본키를 써도 동등 조건이라 단건 검색에 가깝다.

다시 나오면 이렇게 판단

먼저 조건 속성이 그 테이블의 기본키인지 본다. 그다음 조건이 `=`인지 범위 조건인지 본다. 기본키에 대한 범위 조건이면 기본 인덱스로 여러 레코드를 순차 검색하는 형태이고, 기본키가 아니거나 동등 조건이면 다른 선택지를 의심한다.

암기 포인트

기본 인덱스는 기본키 범위 검색, 외래키 검색은 기본 인덱스가 아니다.

NoteBase

문 18. 다음 수강 릴레이션에 대한 설명으로 옳지 않은 것은? (단, 수강 릴레이션에 기본키가 설정되어 있으며, 한 강사는 한 과목만 강의할 수 있고, 한 학생은 서로 다른 한 개 이상의 과목들을 수강할 수 있다)

수강		
학번	강사명	과목명
100	John	데이터베이스
100	Marry	네트워크
200	Jeremy	데이터베이스
300	Iaan	인공지능
300	John	데이터베이스

① {강사명, 과목명}은 후보키가 아니다.

② 강사 ‘James’가 강의할 ‘빅데이터’ 과목을 신설하는 경우 수강 학생이 없이 튜플을 삽입할 수 있다.

③ 학번이 100인 학생이 ‘네트워크’ 과목 수강을 취소하여 그 수강 튜플이 삭제되는 경우, ‘네트워크’ 과목의 강사명도 함께 사라진다.

④ ‘데이터베이스’ 과목의 강사를 ‘John’에서 ‘Tony’로 변경한다면, ‘John’이 강의하는 과목을 수강하는 모든 학생의 강사명을 수정해야 한다.

정답 2번

출제 포인트

이 문제는 수강 릴레이션의 후보키와 함수 종속을 바탕으로 삽입 이상, 삭제 이상, 갱신 이상을 판단할 수 있는지를 묻는 문제이다.

사례 정리

기본키가 설정된 수강 릴레이션은 학생의 수강 사실을 저장한다.
한 강사는 한 과목만 강의할 수 있으므로 강사명은 과목명을 함수적으로 결정한다.
한 학생은 서로 다른 여러 과목을 수강할 수 있으므로 학번만으로는 튜플이 식별되지 않는다.

NoteBase

핵심 개념

후보키는 튜플을 유일하게 식별하는 최소 속성 집합이다.
삽입 이상은 아직 발생하지 않은 사실 때문에 새 정보를 저장하지 못하는 현상이고, 삭제 이상은 어떤 튜플을 지우면서 다른 정보까지 함께 사라지는 현상이며, 갱신 이상은 같은 사실이 여러 튜플에 반복되어 여러 곳을 함께 수정해야 하는 현상이다.

적용 기준

이 사례에서는 기본키에 학생 식별 속성이 포함되어야 하고, 강사명 → 과목명 함수 종속이 성립한다는 점을 함께 본다.
따라서 학생이 정해지지 않은 새 강좌 정보는 독립적으로 저장하기 어렵고, 한 과목의 강사 정보가 여러 수강 튜플에 반복되면 삭제 이상과 갱신 이상이 생긴다.

선지별 해설

1번: O. {강사명, 과목명}은 후보키가 아니다. 같은 강사와 같은 과목에 대해 학번 100과 300이 함께 존재하므로 {강사명, 과목명}만으로는 각 튜플을 유일하게 구분할 수 없다.
2번: X. 틀린 표현은 '수강 학생이 없이 튜플을 삽입할 수 있다'이다. 이 릴레이션은 수강 사실을 저장하는 릴레이션이므로 기본키에도 학생 식별이 필요하다. 따라서 학생이 없는 상태에서 과목과 강사 정보만으로 튜플을 삽입할 수 없다. 올바르게는 '강사 James가 강의할 빅데이터 과목을 신설하더라도 아직 수강 학생이 없으면 이 릴레이션에 해당 정보를 독립적으로 삽입할 수 없다'이다.
3번: O. 학번 100의 네트워크 수강 튜플을 삭제하면 현재 인스턴스에서 네트워크 과목과 그 강사 Marry에 대한 정보도 함께 사라진다. 수강 정보와 과목·강사 정보가 한 릴레이션에 섞여 있어 나타나는 삭제 이상이다.
4번: O. 데이터베이스 과목의 강사를 John에서 Tony로 바꾸면 그 과목을 수강하는 학번 100과 300의 튜플에서 강사명을 모두 수정해야 한다. 같은 강사 정보가 여러 튜플에 중복 저장되어 나타나는 갱신 이상이다.

오답 포인트

2번은 '과목 신설'이라는 말 때문에 별도 과목 테이블에 넣는 상황으로 착각하기 쉽다. 그러나 문제의 대상은 수강 릴레이션 하나뿐이므로 학생이 없는 과목 정보를 이 릴레이션에 바로 넣을 수 없다는 점이 핵심이다.

다시 나오면 이렇게 판단

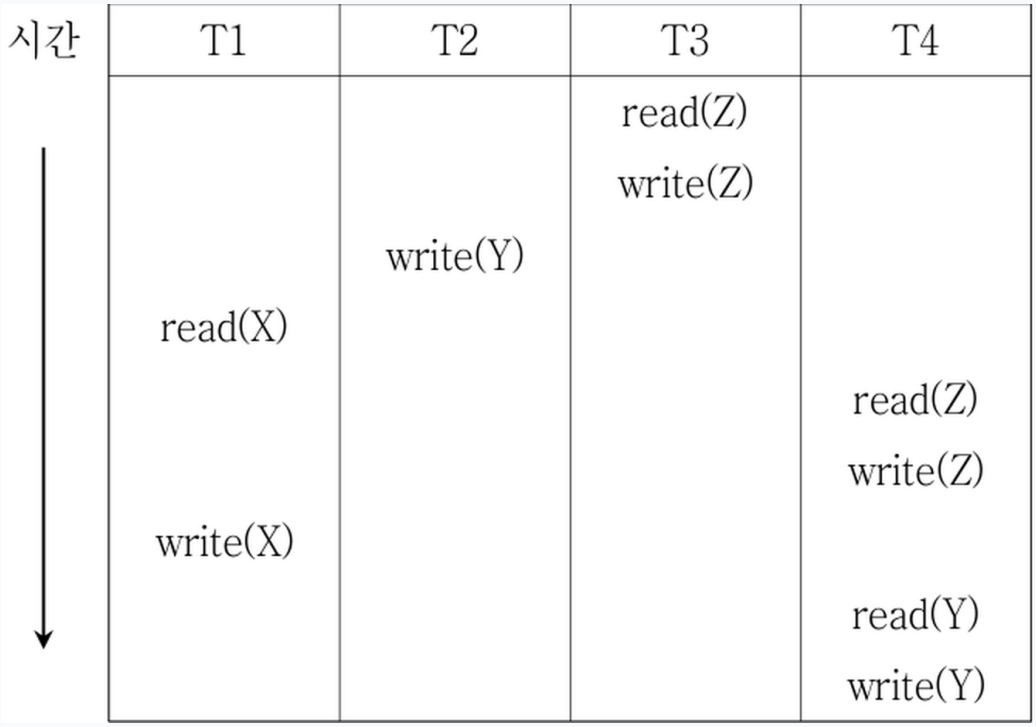
릴레이션 이상 문제는 먼저 이 릴레이션이 무엇을 한 튜플로 저장하는지 확인한 뒤, 기본키에 꼭 필요한 속성이 무엇인지 본다. 그다음 학생 없이 넣을 수 있는지, 한 튜플 삭제로 다른 사실이 함께 없어지는지, 같은 사실을 여러 튜플에서 동시에 고쳐야 하는지를 차례로 점검하면 된다.

암기 포인트

수강 릴레이션에 과목 정보가 중복되면 학생이 없을 때는 삽입 이상, 마지막 수강 삭제 때는 삭제 이상, 강사 변경 때는 갱신 이상이 생긴다.

NoteBase

문 19. 다음과 같이 실행되고 있는 트랜잭션 T1~T4의 스케줄과 동등한 (equivalent) 충돌 직렬가능(conflict serializable) 스케줄이 아닌 것은?



① T1 → T3 → T2 → T4

② T2 → T3 → T4 → T1

③ T2 → T1 → T4 → T3

④ T3 → T2 → T4 → T1

정답 3번

출제 포인트

이 문제는 스케줄에서 같은 데이터 항목에 대한 충돌 순서를 보고 선행 그래프를 만든 뒤, 그 간선을 모두 만족하는 직렬 순서를 고르는 문제이다. 이 스케줄에서는 Z에 대해 T3→T4, Y에 대해 T2→T4의 선행 관계가 생기고, T1은 다른 트랜잭션과 충돌하지 않아 위치가 자유롭다.

NoteBase

선지별 해설

① O: T3가 T4보다 앞에 있고 T2도 T4보다 앞에 있으므로 충돌 간선 T3→T4, T2→T4를 모두 만족한다. T1은 X만 다루므로 어느 위치에 있어도 무방하다.
② O: T2가 먼저 와서 Y에 대한 선행 관계를 지키고, T3가 T4보다 먼저 와서 Z에 대한 선행 관계도 지킨다. 따라서 원 스케줄과 동등한 충돌 직렬가능 스케줄이다.
③ X: 잘못된 부분은 T4가 T3보다 앞에 놓인 순서이다. 원 스케줄에서는 T3의 write(Z)가 T4의 read(Z), write(Z)보다 먼저 실행되므로 반드시 T3→T4가 되어야 한다. 올바르게는 T3가 T4보다 먼저 오는 직렬 순서여야 한다.
④ O: T3가 T4보다 앞에 있고 T2도 T4보다 앞에 있으므로 필요한 두 간선을 모두 만족한다. T1은 다른 트랜잭션과 충돌이 없어 마지막에 와도 문제없다.

오답 포인트

T1의 read(X), write(X)가 눈에 띄어서 T1과 다른 트랜잭션 사이에도 순서 제약이 있다고 오해하기 쉽다. 그러나 충돌 직렬가능성은 같은 데이터 항목에 대한 read/write, write/read, write/write만 본다. 이 문항의 실제 제약은 Z에서 생긴 T3→T4와 Y에서 생긴 T2→T4 두 개뿐이다.

다시 나오면 이렇게 판단

시간 순서대로 같은 데이터 항목을 공유하는 연산만 골라 트랜잭션 간 간선을 만든다. 그다음 모든 간선을 보존하는 직렬 순서인지 확인하면 된다. 간선이 없는 트랜잭션은 어느 위치로 옮겨도 된다.

암기 포인트

충돌 직렬가능성은 같은 항목의 충돌만 간선으로 남기고, 직렬 순서는 그 간선 방향만 지키면 된다.

NoteBase

문 20. 트랜잭션 T1과 T2가 병행 수행될 때, 정확한 결과를 보장할 수 있는 스케줄만을 모두 고르면? (단, 두 트랜잭션 수행 전 A와 B의 값은 각각 50이다)

ㄱ.

시간	T1	T2
		read(A)
		A := A×2
		write(A)
		read(B)
		B := B×2
		write(B)
	read(A)	
	A := A+100	
	write(A)	
	read(B)	
	B := B+100	
	write(B)	

ㄴ.

시간	T1	T2
	read(A)	
	A := A+100	
		read(A)
		A := A×2
		write(A)
	write(A)	
		read(B)
		B := B×2
	read(B)	
	B := B+100	
	write(B)	
		write(B)

ㄷ.

시간	T1	T2
	lock(A)	
	read(A)	
	A := A+100	
	write(A)	
	unlock(A)	
		lock(A)
		read(A)
		A := A×2
		write(A)
		unlock(A)
		lock(B)
		read(B)
		B := B×2
		write(B)
		unlock(B)
	lock(B)	
	read(B)	
	B := B+100	
	write(B)	
	unlock(B)	

ㄹ.

시간	T1	T2
	lock(A)	
	read(A)	
	A := A+100	
	write(A)	
	lock(B)	
	unlock(A)	
		lock(A)
		read(A)
		A := A×2
		write(A)
	read(B)	
	B := B+100	
	write(B)	
	unlock(B)	
		lock(B)
		unlock(A)
		read(B)
		B := B×2
		write(B)
		unlock(B)

① ㄱ, ㄴ

② ㄱ, ㄹ

③ ㄴ, ㄷ

④ ㄷ, ㄹ

정답 2번

출제 포인트

이 문제는 스케줄이 두 트랜잭션의 연산 순서를 각 데이터 항목마다 일관되게 유지하여 직렬 스케줄과 같은 결과를 내는지를 도식으로 판별하는 문제이다.

NoteBase

선지별 해설

1번 X: ㄱ은 T2가 A와 B를 모두 먼저 수행하고 그다음 T1이 수행되는 직렬 스케줄이므로 정확한 결과를 보장한다. 반면 ㄴ은 A에서 T1이 50을 읽어 150을 계산한 뒤 T2가 같은 50을 읽어 100을 기록하고, 마지막에 T1이 150을 써서 T2의 갱신이 사라진다. B도 마지막 쓰기 순서 때문에 T1의 갱신이 사라져 직렬 결과와 같지 않다. 'ㄱ, ㄴ'이 아니라 올바르게는 'ㄱ, ㄹ'이다.
2번 O: ㄱ은 T2 전체 후 T1 전체가 수행되는 직렬 스케줄이어서 결과가 T2→T1 순서와 같다. ㄹ은 T1이 A와 B의 lock을 모두 확보한 뒤에 unlock을 시작하고, T2도 필요한 lock을 모두 확보하기 전에는 unlock을 시작하지 않으므로 두 트랜잭션이 같은 직렬 순서로 직렬가능하다. 실제 결과도 A와 B 모두 T1→T2 순서가 적용되어 직렬 결과와 일치한다.
3번 X: ㄴ은 lost update가 발생하여 정확한 결과를 보장하지 못한다. ㄷ도 A에서는 T1→T2 순서가 적용되지만 B에서는 T2→T1 순서가 적용되어 한 스케줄 안에서 데이터 항목별 순서가 뒤바뀐다. 따라서 하나의 직렬 순서와 동등하지 않다. 'ㄴ, ㄷ'이 아니라 올바르게는 'ㄱ, ㄹ'이다.
4번 X: ㄹ은 정확한 결과를 보장하지만 ㄷ은 보장하지 못한다. ㄷ의 잘못된 부분은 A 처리 후 lock을 해제한 뒤 다시 B에 대한 lock을 잡는 점이다. 이 때문에 두 트랜잭션의 순서가 A와 B에서 서로 다르게 섞인다. 'ㄷ, ㄹ'이 아니라 올바르게는 'ㄱ, ㄹ'이다.

오답 포인트

lock이 보인다는 이유만으로 항상 정확하다고 보면 틀린다. 핵심은 lock의 존재 자체가 아니라 unlock을 시작한 뒤 새로운 lock을 다시 잡는지, 그리고 A와 B에 대해 두 트랜잭션의 선후관계가 끝까지 같은지이다.

다시 나오면 이렇게 판단

각 데이터 항목 A, B에 대해 T1과 T2의 적용 순서를 따로 적어 보고 그 순서가 모든 항목에서 동일한지 확인하면 된다. 모든 항목이 T1→T2 또는 T2→T1 중 하나로 통일되면 직렬가능하고, 항목마다 순서가 섞이면 정확한 결과를 보장하지 못한다.

암기 포인트

직렬가능성은 모든 데이터 항목에서 트랜잭션의 선후관계가 한 줄로 같아야 성립한다.

NoteBase

문 21. 분산 데이터베이스 시스템에 대한 설명으로 옳지 않은 것은?

① 사용자들은 원하는 데이터가 어느 지역에 저장되어 있는지를 명확하게 알 수 있다.

② 데이터베이스를 여러 지역에 중복하여 분산 저장하면 가용성(availability)이 높아진다.

③ 지역 사이트들도 어느 정도 자치성(autonomy)을 가지고 독립적으로 동작할 수 있다.

④ 수평적 단편화(horizontal fragmentation)는 릴레이션 투플들을 몇 개의 부분집합(단편)으로 분할한 후, 각 부분집합을 서로 다른 사이트에 저장하는 것이다.

정답 1번

출제 포인트

분산 데이터베이스 시스템의 핵심 특성인 위치 투명성, 가용성, 자치성, 수평적 단편화의 의미를 구분하는 문제이다.

핵심 개념

위치 투명성은 사용자가 데이터의 실제 저장 위치를 몰라도 동일하게 접근할 수 있는 성질이다. 가용성은 일부 장애가 있어도 데이터 접근이나 서비스 지속 가능성이 높게 유지되는 성질이다. 자치성은 각 지역 사이트가 어느 정도 독립적으로 운영되고 제어될 수 있는 성질이다. 수평적 단편화는 릴레이션의 튜플을 행 기준으로 나누어 여러 사이트에 분산 저장하는 방식이다.

선지별 해설

① X: 틀린 부분은 "사용자들은 원하는 데이터가 어느 지역에 저장되어 있는지를 명확하게 알 수 있다"이다. 분산 데이터베이스의 장점은 위치를 사용자가 명확히 아는 데 있지 않고, 위치를 몰라도 동일하게 접근할 수 있는 위치 투명성에 있다. 올바르게는 "사용자는 데이터가 어느 지역에 저장되어 있는지를 몰라도 된다"이다.
② O: 동일 데이터를 여러 지역에 중복 저장하면 한 사이트에 장애가 나더라도 다른 복제본으로 서비스를 계속할 수 있어 가용성이 높아진다.
③ O: 분산 데이터베이스에서는 각 지역 사이트가 어느 정도 자치성을 가지고 독립적으로 운영될 수 있다. 이는 중앙 집중식 시스템과 구별되는 대표 특성이다.
④ O: 수평적 단편화는 릴레이션을 튜플, 즉 행 단위의 여러 부분집합으로 나누는 방식이다. 각 단편을 서로 다른 사이트에 저장하는 설명이므로 옳다.

오답 포인트

위치 투명성을 "데이터 위치를 잘 아는 것"으로 뒤집어 이해하면 ①을 맞는 설명으로 오해하기 쉽다. 이 문항은 투명성이 사용자의 인지 확대가 아니라 위치 은닉이라는 점을 묻는다.

NoteBase

다시 나오면 이렇게 판단

분산 데이터베이스 선지는 먼저 투명성, 복제, 자치성, 단편화의 정의를 짧게 대입해 판단한다. 저장 위치를 사용자가 알아야 한다는 표현이 나오면 투명성 개념과 반대인지 먼저 확인하면 된다.

암기 포인트

위치 투명성은 위치를 드러내는 개념이 아니라 위치를 숨기는 개념이다. 수평 단편화는 행 분할, 수직 단편화는 열 분할이다.

NoteBase

문 22. 다음 두 트랜잭션이 테이블 R(A, B)에 대해 병행 수행되고 있다. 트랜잭션 T2에서 “select sum(B) from R;”의 결과가 항상 동일한 값을 출력하도록 보장하는 트랜잭션 고립성 수준(isolation level)은? (단, 테이블 R의 속성 A, B의 도메인은 정수이다)

T1	T2
begin transaction; insert into R values (3, 150); commit;	begin transaction; select sum(B) from R; select sum(B) from R; commit;

① 직렬가능(SERIALIZABLE)

② 반복 가능한 읽기(REPEATABLE READ)

③ 커밋된 데이터 읽기(READ COMMITTED)

④ 커밋되지 않은 데이터 읽기(READ UNCOMMITTED)

정답 1번

출제 포인트

이 문제는 두 번의 동일한 집계 질의가 항상 같은 결과를 내도록 보장하는 격리 수준, 즉 팬텀 읽기까지 막는 수준을 고를 수 있는지를 묻는다.

사례 정리

T2는 하나의 트랜잭션 안에서 `select sum(B) from R;`를 두 번 수행한다.
T1은 그 사이에 `(3, 150)` 행을 삽입하고 커밋한다.
판단의 핵심은 두 번째 조회 시 새로 삽입된 행이 집계 대상에 들어올 수 있는지 여부이다.

핵심 개념

직렬가능은 동시에 수행된 결과가 어떤 직렬 수행 결과와 같도록 보장하는 격리 수준이다.
이 문제의 핵심 현상은 기존 행의 값 변경이 아니라 새 행이 조회 범위에 끼어드는 팬텀 읽기이다.
반복 가능한 읽기는 이미 읽은 행의 재조회 일관성은 보장하지만, 범위 안에 새로 들어오는 행까지 항상 막지는 못한다.

NoteBase

적용 기준

이 사례에서는 같은 트랜잭션 안에서 테이블 전체에 대한 합계를 두 번 읽으므로 읽는 행의 집합 자체가 변하지 않아야 한다.
따라서 새 삽입 행을 팬텀으로 보지 않게 해야 하며, 이를 항상 보장하는 수준은 직렬가능이다.

선지별 해설

1번 O: 직렬가능은 T2가 읽는 범위에 대한 팬텀 삽입까지 배제하므로 두 번의 `sum(B)`가 항상 같은 결과를 낸다. T1의 삽입은 T2와 완전히 분리된 직렬 순서로 처리된 것과 같은 효과를 가져야 한다.
2번 X: 반복 가능한 읽기는 같은 행을 다시 읽을 때 값이 바뀌지 않는 수준에 가깝고, 새 행 삽입으로 조회 집합이 늘어나는 팬텀은 항상 막지 못한다. 틀린 표현은 "항상 동일한 값"을 보장한다는 부분이며, 올바르게는 팬텀까지 막아야 항상 동일한 집계값이 보장된다.
3번 X: 커밋된 데이터 읽기는 각 질의 시점에 커밋된 데이터만 읽게 할 뿐이다. 따라서 첫 번째 조회 뒤 T1이 커밋하면 두 번째 조회에서는 새 행 `(3, 150)`이 보일 수 있어 합계가 달라진다. 올바르게는 커밋 여부만이 아니라 동일 범위 재조회 안정성까지 보장해야 한다.
4번 X: 커밋되지 않은 데이터 읽기는 미커밋 변경까지 볼 수 있는 가장 약한 수준이므로 결과 일관성이 가장 약하다. T1의 삽입이 커밋 전이든 커밋 후이든 T2의 두 집계값이 달라질 수 있다. 올바르게는 이 수준은 동일 결과 보장과 거리가 가장 멀다.

오답 포인트

반복 가능한 읽기와 직렬가능을 같은 수준으로 착각하면 2번으로 흐르기 쉽다. 이 문항은 기존 행의 재읽기가 아니라 새 행 삽입으로 집계 대상 집합이 바뀌는 팬텀 문제라는 점이 함정이다.

다시 나오면 이렇게 판단

같은 트랜잭션에서 범위 조회나 집계 조회를 반복하는데 중간에 다른 트랜잭션의 삽입이 개입하면 먼저 팬텀 가능성을 본다. 새 행이 들어와도 결과가 항상 같아야 하면 직렬가능을 고른다.

암기 포인트

집계값 반복 조회의 안정성까지 묻는 순간 핵심은 팬텀이며, 팬텀을 확실히 막는 선택지는 직렬가능이다.

NoteBase

문 23. 다음 관계 대수식에서 동등 규칙(equivalence rule)이 성립하는 것만을 모두 고르면? (단, E, E_1, E_2는 관계대수식을, θ, θ_1, θ_2는 술어(predicate) 조건을, L_1, L_2는 속성을 의미한다)

ㄱ. $\Pi_{L_1}(\Pi_{L_2}(E)) \equiv \Pi_{L_2}(E)$
ㄴ. __1 _2(E) ≡ __2(__1(E))
ㄷ. $\sigma_\theta(E_1 - E_2) \equiv \sigma_\theta(E_1) - E_2$
ㄹ. $E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$

① ㄱ, ㄷ
② ㄴ, ㄷ
③ ㄴ, ㄹ
④ ㄱ, ㄴ, ㄹ

정답 2번

출제 포인트

이 문제는 관계대수의 대표적인 동등 규칙, 즉 중첩 투영의 조건, 선택 연산의 분해, 차집합과 선택의 결합, 조인의 교환 가능성을 구별할 수 있는지를 묻는다.

핵심 개념

투영은 속성을 줄이는 연산이므로 중첩 투영의 동등식은 바깥 투영이 남길 속성 집합을 기준으로 성립한다.
선택은 술어의 논리곱을 연속된 선택으로 분해할 수 있으므로 __1 _2(E) ≡ __2(__1(E))가 성립한다.
차집합 뒤의 선택은 남아 있는 투플만 다시 거르는 것이므로 $\sigma_\theta(E_1 - E_2) \equiv \sigma_\theta(E_1) - E_2$로 정리할 수 있다.
일반적인 자연조인 해석에서는 $E_1 \bowtie E_2$와 $E_2 \bowtie E_1$ 사이에 교환법칙이 성립한다.

선지별 해설

1번 X: ㄷ은 맞지만 ㄱ이 틀리므로 오답이다. 틀린 표현은 $\Pi_{L_1}(\Pi_{L_2}(E)) \equiv \Pi_{L_2}(E)$ 부분이다. 올바르게는 $L_1 \subseteq L_2$일 때 $\Pi_{L_1}(\Pi_{L_2}(E)) \equiv \Pi_{L_1}(E)$이다.
2번 O: ㄴ과 ㄷ을 성립하는 동등 규칙으로 보면 정답 키와 일치한다. ㄴ은 논리곱 조건을 단계별 선택으로 나눈 형태이고, ㄷ은 차집합 결과가 원래 E_1의 투플 중 E_2에 없는 것들만 남긴다는 점을 이용한 등식이다.
3번 X: ㄴ은 맞다. 다만 ㄹ의 $E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$는 일반적인 자연조인 해석에서는 교환법칙이 성립하므로, 문항의 기호 해석과 제공된 정답 키 사이에 충돌 여지가 있다. 제공된 정답 키 기준으로는 ㄹ을 포함한 3번이 아니라 ㄴ, ㄷ만 묶은 2번으로 정리된다.
4번 X: ㄱ이 틀리므로 오답이다. 틀린 표현은 중첩 투영의 결과를 안쪽 투영 $\Pi_{L_2}(E)$와 같다고 본 부분이다. 올바르게는 바깥 투영이 최종 결과를 결정하므로 ㄱ을 빼고, 정답 키 기준으로는 ㄴ과 ㄷ만 남는다.

NoteBase

오답 포인트

가장 흔한 함정은 중첩 투영에서 바깥 투영과 안쪽 투영의 역할을 뒤집어 보는 것이다. 또 ⋈를 자연조인으로 읽으면 ㄹ도 성립해 보여 정답 키와 충돌처럼 느껴질 수 있다는 점이 이 문항의 혼선 포인트이다.

다시 나오면 이렇게 판단

투영이 중첩되면 최종적으로 남는 속성 집합이 무엇인지부터 본다. 선택은 논리곱이면 연속 선택으로 쪼갤 수 있는지 보고, 차집합이 있으면 남는 투플이 결국 E_1 쪽 투플이라는 점을 이용해 등식을 판단한다.

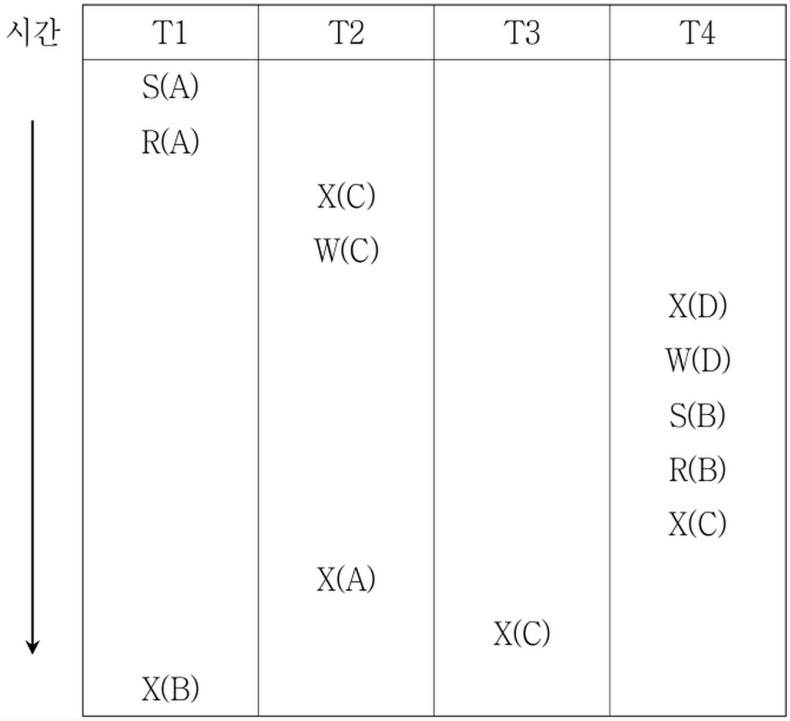
암기 포인트

투영은 바깥쪽이 최종 속성을 정하고, 선택은 논리곱으로 분해되며, 차집합 뒤 선택은 왼쪽 릴레이션 기준으로 생각한다.

NoteBase

문 24. 트랜잭션 T1 ~ T4가 다음 연산과 조건에 따라 수행하는 아래 스케줄에 대한 설명으로 옳은 것은?

○ S(A): 데이터 A에 읽기 연산(R(A))을 위한 공용 로크 (shared lock) 요청
○ X(A): 데이터 A에 쓰기 연산(W(A))을 위한 전용 로크 (exclusive lock) 요청
○ 로크를 요청한 트랜잭션은 요청한 로크가 허용될 때까지 대기한다.
○ 트랜잭션이 소유한 로크는 트랜잭션이 완료될 때 해제된다.



① 모든 트랜잭션 T1 ~ T4가 교착상태에 빠졌다.
② 트랜잭션 T3을 철회(abort)하면 교착상태가 해결될 수 있다.
③ 교착상태를 해결하기 위해 철회할 트랜잭션으로 T4를 선택하는 것이 가장 좋다.
④ 트랜잭션 T2를 철회하면, 트랜잭션 T3은 C에 대한 전용 로크를 획득할 수 있다.

정답 4번

출제 포인트

이 문제는 로크 스케줄에서 각 트랜잭션의 대기 관계를 따라가며 교착상태의 구성원과 철회 후 진행 가능성을 판단하는 문제이다.

자료 읽는 순서

1. 각 트랜잭션이 이미 보유한 로크를 확인한다.
2. 뒤에서 요청한 로크가 누구의 로크 때문에 대기하는지 연결한다.
3. 대기 간선이 순환하는지 보고, 철회 시 어느 대기가 풀리는지 추적한다.

NoteBase

선지별 해설

1번 X: 교착상태의 순환은 T1 → T4 → T2 → T1이다. T1은 B에 대한 X(B)를 원하지만 T4의 S(B) 때문에 기다리고, T4는 C에 대한 X(C)를 원하지만 T2의 X(C) 때문에 기다리며, T2는 A에 대한 X(A)를 원하지만 T1의 S(A) 때문에 기다린다. T3는 C에 대한 X(C)를 기다리기만 할 뿐 다른 트랜잭션이 T3를 기다리지는 않으므로 교착상태의 구성원이 아니다. 틀린 표현은 "모든 트랜잭션 T1~T4"이고, 올바르게는 "T1, T2, T4가 교착상태에 빠졌다"이다.
2번 X: T3를 철회해도 T1, T2, T4 사이의 순환 대기는 그대로 남는다. T3는 교착 고리 바깥에서 C를 기다리는 대기자일 뿐이므로, T3를 없애도 T1은 여전히 B를 기다리고 T4는 여전히 C를 기다리며 T2는 여전히 A를 기다린다. 틀린 표현은 "T3을 철회하면 교착상태가 해결"이고, 올바르게는 "교착 순환에 들어 있는 T1, T2, T4 중 하나를 철회해야 해결된다"이다.
3번 X: T4를 철회하면 교착상태는 해소될 수 있지만, 그것이 "가장 좋다"고 단정할 정보는 없다. 희생자 선택은 일반적으로 수행한 작업량, 복구 비용, 재시작 비용 같은 추가 기준이 있어야 판단할 수 있다. 틀린 표현은 "T4를 선택하는 것이 가장 좋다"이고, 올바르게는 "교착 순환에 속한 트랜잭션 중 하나를 철회하면 해결되며, 최적 희생자는 추가 기준으로 정한다"이다.
4번 O: T2를 철회하면 T2가 보유하던 X(C)가 해제되어 C를 둘러싼 대기가 풀리기 시작한다. 그 결과 T4가 진행하여 작업을 마치고 로크를 해제한 뒤, T3도 결국 C에 대한 X(C)를 획득할 수 있다. T2 철회는 T1 → T4 → T2 → T1의 순환을 끊는 올바른 해결 방법이다.

오답 포인트

T3도 C를 기다리므로 교착상태에 포함된다고 착각하기 쉽다. 그러나 교착상태는 서로가 서로를 기다리는 순환이어야 하므로, 단순 대기와 순환 대기를 구분해야 한다.

다시 나오면 이렇게 판단

로크 스케줄 문제는 wait-for graph를 머릿속에 그리면 정리가 쉽다. 누가 누구의 로크 때문에 멈췄는지를 화살표로 연결해 순환이 생기면 교착상태이고, 순환 밖에서 한쪽으로만 기다리는 트랜잭션은 단순 대기이다.

암기 포인트

교착상태는 순환 대기이고, 단순 대기는 교착상태가 아니다. 해결은 순환 안의 트랜잭션을 끊어야 한다.

NoteBase

문 25. 다음은 OLAP(Online Analytical Processing)의 기능에 대한 설명이다. (가), (나)에 들어갈 용어를 바르게 연결한 것은?

○ (가)은 교차 테이블(cross-tab)에서 데이터를 분석하는 차원을 다양하게 변경하는 기능이다.
○ 집계 연산에 대해 사용자가 원하는 크기의 데이터로 보여주기 위해, 미세한 단위부터 큰 단위로 변화시키는 연산을 (나)이라고 한다.

① (가): 피보팅(pivoting), (나): 롤업(roll-up)
② (가): 피보팅(pivoting), (나): 드릴다운(drill-down)
③ (가): 슬라이싱(slicing), (나): 롤업(roll-up)
④ (가): 슬라이싱(slicing), (나): 드릴다운(drill-down)

정답 1번

출제 포인트

이 문제는 OLAP의 기본 연산인 피보팅, 슬라이싱, 롤업, 드릴다운의 의미와 방향을 구별할 수 있는지를 묻는다.

핵심 개념

피보팅은 교차 테이블의 행·열·축 배치를 바꾸어 같은 데이터를 다른 차원 관점에서 보는 기능이다.
슬라이싱은 특정 차원값을 고정하여 데이터의 일부만 잘라 보는 기능이다.
롤업은 세부 수준의 데이터를 상위 집계 수준으로 올려 더 큰 단위로 요약하는 연산이다.
드릴다운은 상위 집계 수준에서 하위 세부 수준으로 내려가 더 미세하게 보는 연산이다.

선지별 해설

1번: O. (가)의 설명은 피보팅이다. 교차 테이블에서 분석 축을 바꾸어 차원 배치를 재구성하는 기능이기 때문이다. (나)의 설명도 롤업이다. 미세한 단위에서 큰 단위로 올라가며 집계 수준을 높이는 연산이기 때문이다.
2번: X. (가)는 피보팅이 맞지만 (나)가 틀렸다. 드릴다운은 큰 단위에서 작은 단위로 내려가 세부 데이터를 더 자세히 보는 연산이다. 틀린 표현은 '드릴다운'이며, 올바르게는 '롤업'이다.
3번: X. (가)가 틀렸다. 슬라이싱은 차원을 다양하게 변경하는 기능이 아니라 특정 차원값을 선택해 데이터 일부만 보는 기능이다. 틀린 표현은 '슬라이싱'이며, 올바르게는 '피보팅'이다. (나)는 롤업이 맞다.
4번: X. (가)와 (나)가 모두 틀렸다. 슬라이싱은 부분 추출이고 드릴다운은 큰 단위에서 작은 단위로 내려가는 연산이다. 틀린 표현은 '슬라이싱, 드릴다운'이며, 올바르게는 '피보팅, 롤업'이다.

NoteBase

오답 포인트

피보팅과 슬라이싱을 둘 다 화면 조작 기능으로만 외우면 헷갈린다. 차원 배치를 바꾸면 피보팅이고, 특정 차원값만 남겨 부분 집합을 만들면 슬라이싱이다. 또 롤업과 드릴다운은 집계 방향이 반대인데, '미세한 단위부터 큰 단위로'라는 표현이 롤업의 결정 단서이다.

다시 나오면 이렇게 판단

먼저 차원을 바꾸는지, 일부를 잘라내는지를 구분한다. 그다음 집계 수준이 세부에서 요약으로 올라가면 롤업, 요약에서 세부로 내려가면 드릴다운으로 판단한다.

암기 포인트

피보팅은 축 회전, 슬라이싱은 부분 절단, 롤업은 아래에서 위로 집계, 드릴다운은 위에서 아래로 세분이다.

NoteBase