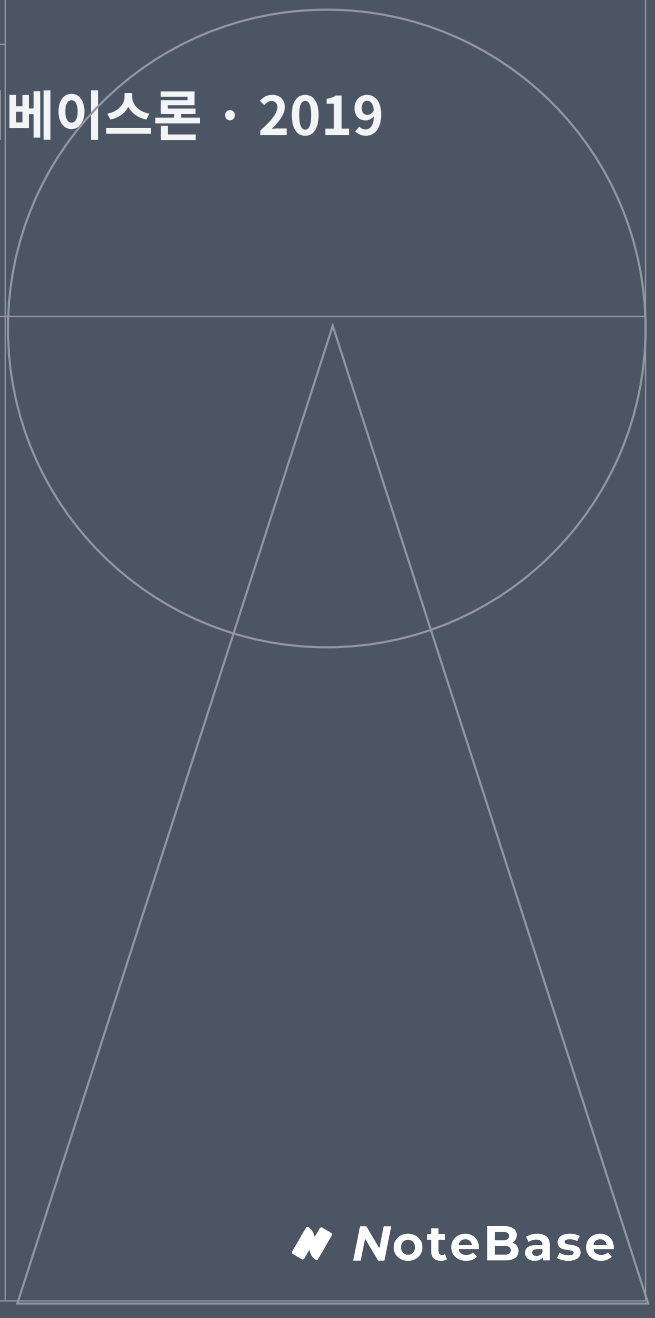
기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2019



NoteBase

기출OX 해설집

공무원/국가직/7급 · 데이터베이스론 · 2019

공무원/국가직/7급 · 데이터베이스론 · 2019

문 1. 관계형 데이터베이스에서 참조 무결성 제약조건을 만족하도록 외래키를 생성하고자 한다. 이와 관련된 SQL 명령문의 제약조건에 대한 설명으로 옳지 않은 것은?

① 'on delete set null'은 참조되는 테이블의 행이 삭제되면, 참조하는 테이블의 행에 있는 외래키 열에 null을 저장한다.

② 'on delete cascade'는 참조되는 테이블의 행이 삭제되면, 참조하는 테이블의 행에 있는 외래키 열을 삭제한다.

③ 'on delete set default'는 참조되는 테이블의 행이 삭제되면, 참조하는 테이블의 행에 있는 외래키 열에 사전에 정의된 default 값을 저장한다.

④ 'on delete no action'은 참조되는 테이블의 행을 삭제하려고 할 때, 참조하는 테이블의 행이 존재할 경우 삭제 명령이 수행되지 못하도록 한다.

정답 2번

출제 포인트

이 문제는 외래키의 참조 무결성 유지 옵션인 ON DELETE SET NULL, CASCADE, SET DEFAULT, NO ACTION의 동작 차이를 구분하는 문제이다.

핵심 개념

ON DELETE SET NULL은 부모 행이 삭제될 때 자식 행의 외래키 값을 NULL로 바꾸는 옵션이다.
ON DELETE CASCADE는 부모 행이 삭제될 때 그 부모를 참조하던 자식 행 자체를 함께 삭제하는 옵션이다.
ON DELETE SET DEFAULT는 부모 행이 삭제될 때 자식 행의 외래키 값을 미리 정의된 기본값으로 바꾸는 옵션이다.
ON DELETE NO ACTION은 참조 중인 자식 행이 남아 있으면 부모 행 삭제를 허용하지 않는 옵션이다.

NoteBase

선지별 해설

① O
참조되는 테이블의 행이 삭제되면 참조하는 테이블의 외래키 값에 NULL을 저장하는 설명이므로 맞다. 외래키 값만 바뀌고 자식 행 자체는 남는다.

② X
틀린 부분은 "참조하는 테이블의 행에 있는 외래키 열을 삭제한다"는 표현이다. ON DELETE CASCADE는 외래키 열만 지우는 것이 아니라 그 외래키를 가진 자식 행 전체를 함께 삭제한다.

③ O
참조되는 행이 삭제되면 자식 행의 외래키 값에 사전에 정의된 기본값을 넣는 동작을 설명하므로 맞다. 역시 삭제 대상은 열 자체가 아니라 외래키 값이다.

④ O
NO ACTION은 참조하는 자식 행이 존재하면 부모 행 삭제를 막는 설명이므로 맞다. 즉 참조 무결성을 깨뜨리는 삭제 명령이 수행되지 못하도록 한다.

오답 포인트

CASCADE를 보면 연쇄적으로 외래키 값만 바뀐다고 오해하기 쉽지만, 실제 연쇄 대상은 자식 행 전체이다. 이 문항은 "열을 삭제한다"와 "행을 삭제한다"를 바꿔 놓은 표현을 잡아내는 함정이다.

다시 나오면 이렇게 판단

삭제 옵션 문제는 부모 행 삭제 뒤 자식 쪽에 무엇이 남는지를 기준으로 판단한다. 값만 NULL로 바뀌면 SET NULL, 기본값으로 바뀌면 SET DEFAULT, 자식 행이 함께 사라지면 CASCADE, 삭제 자체가 막히면 NO ACTION이다.

암기 포인트

SET NULL은 값 변경, SET DEFAULT는 기본값 변경, CASCADE는 자식 행 삭제, NO ACTION은 부모 삭제 차단이다.

NoteBase

문 2. 다음 과제 테이블은 학년별 동아리에 가입한 학생 수와 제출한 과제 수를 저장하고 있다. '학생이 10명 이상 가입한 동아리에 대하여 동아리와 제출한 총 과제 수를 출력하시오'를 수행하기 위한 SQL문으로 옳은 것은?

과제			
학년	동아리	학생수	과제수
1	A	20	20
1	B	10	20
1	C	5	8
2	A	15	10
2	B	15	20
3	A	5	15
3	B	12	20
3	C	8	15

①
```
SELECT 동아리, SUM(과제수)
FROM 과제
GROUP BY 동아리
HAVING SUM(학생수) >= 10 ;
```

②
```
SELECT 동아리, SUM(과제수)
FROM 과제
WHERE SUM(학생수) >= 10 ;
```

③
```
SELECT 동아리, SUM(과제수)
FROM 과제
HAVING SUM(학생수) >= 10 ;
```

④
```
SELECT 동아리, SUM(과제수)
FROM 과제
WHERE 학생수 >= 10
GROUP BY 동아리 ;
```

정답 1번

NoteBase

출제 포인트

이 문제는 동아리별로 행을 묶은 뒤 총 학생수를 조건으로 걸고, 그 결과에서 총 과제 수를 구해야 하므로 `WHERE`와 `HAVING`의 차이와 `GROUP BY`의 위치를 묻는 문제이다.

자료 읽는 순서

1. 먼저 동아리별로 행을 묶어 총 학생수와 총 과제수를 계산한다.
2. 그다음 총 학생수가 10명 이상인 동아리만 남긴다.
3. 마지막으로 동아리와 SUM(과제수)를 출력한다.

선지별 해설

① O: 동아리별 합계를 구하면 A는 학생수 40명·과제수 45개, B는 37명·과제수 60개, C는 13명·과제수 23개이다. 총 학생수 조건은 그룹 결과에 걸어야 하므로 `GROUP BY 동아리 HAVING SUM(학생수) >= 10`이 맞다.
② X: `WHERE` 절에서는 집계함수 `SUM(학생수)`를 조건으로 사용할 수 없고, 이 문장은 `GROUP BY 동아리`도 빠져 있다. 틀린 표현은 `WHERE SUM(학생수) >= 10`이고, 올바르게는 `GROUP BY 동아리 HAVING SUM(학생수) >= 10`이다.
③ X: `HAVING`은 그룹에 대한 조건이므로 `GROUP BY 동아리` 없이 쓰면 동아리별 합계를 기준으로 판단할 수 없다. 틀린 표현은 `HAVING SUM(학생수) >= 10`만 두고 그룹화가 없는 부분이고, 올바르게는 `GROUP BY 동아리 HAVING SUM(학생수) >= 10`이다.
④ X: `WHERE 학생수 >= 10`은 각 행을 먼저 걸러 버리므로 A의 3학년 5명 행과 C의 두 행이 빠진다. 그 결과 A의 과제수는 45가 아니라 30으로 줄고 C도 아예 출력되지 않는다. 틀린 표현은 `WHERE 학생수 >= 10`이고, 올바르게는 동아리별 총 학생수를 구한 뒤 `HAVING SUM(학생수) >= 10`으로 거는 것이다.

오답 포인트

개별 행 조건과 그룹 조건을 섞으면 ④처럼 집계 전에 행이 탈락하여 총 과제 수가 달라진다. 또한 `SUM(학생수)` 같은 집계 결과는 `WHERE`가 아니라 `HAVING`에서 판단한다는 점을 놓치면 ②, ③처럼 문법과 의미를 동시에 틀리게 된다.

다시 나오면 이렇게 판단

조건의 대상이 한 행의 값이면 `WHERE`, 합계·평균·개수처럼 그룹을 만든 뒤의 값이면 `GROUP BY` 뒤 `HAVING`으로 판단한다. `SELECT`에 일반 열과 집계함수가 함께 나오면 일반 열은 `GROUP BY`에 포함되어야 한다.

암기 포인트

행 조건은 `WHERE`, 그룹 조건은 `HAVING`, 일반 열과 집계함수를 함께 고르면 `GROUP BY`가 필요하다.

NoteBase

문 3. 다음 관계형 데이터베이스의 세 가지 기능적 요소에 대한 설명에서 ㉠ ~ ㉢에 들어갈 용어를 바르게 연결한 것은?

○ (㉠)는(은) SQL에서 삽입, 삭제, 갱신과 같은 데이터 변경문을 실행할 때 미리 명시된 조건을 만족하는 경우 특정한 동작을 자동으로 수행할 수 있도록 한다.
○ (㉡)는(은) 데이터베이스 내에 존재하는 작업 순서가 정해진 수행 단위로서 DBMS에서 컴파일된 후 실행된다.
○ (㉢)는(은) 데이터베이스에서 데이터를 신속하게 탐색할 수 있도록 만든 데이터 구조이다.

① ㉠: 인덱스, ㉡: 트리거(trigger), ㉢: 주장(assertion)
② ㉠: 주장, ㉡: 인덱스, ㉢: 저장 프로시저(stored procedure)
③ ㉠: 주장, ㉡: 인덱스, ㉢: 트리거
④ ㉠: 트리거, ㉡: 저장 프로시저, ㉢: 인덱스

정답 4번

출제 포인트

이 문제는 관계형 데이터베이스의 기능적 요소인 트리거, 저장 프로시저, 인덱스, 주장의 개념을 구별할 수 있는지를 묻는다.

핵심 개념

트리거는 삽입, 삭제, 갱신 같은 이벤트가 발생할 때 미리 정의된 조건에 따라 자동 실행되는 데이터베이스 객체이다.
저장 프로시저는 데이터베이스에 저장된 SQL 작업 절차를 하나의 실행 단위로 컴파일해 두고 호출하여 실행하는 프로그램 단위이다.
인덱스는 특정 속성값을 빠르게 찾도록 만드는 탐색용 데이터 구조이다.
주장은 데이터베이스 전체에 대해 항상 만족해야 하는 무결성 제약 조건이다.

NoteBase

선지별 해설

1번: X. ㉠을 인덱스로 둔 것이 틀렸다. ㉠은 데이터 변경문 실행 시 조건을 만족하면 자동 동작하는 기능이므로 트리거가 맞다. 또한 ㉡은 저장 프로시저가 아니라 트리거로 둔 것도 틀렸다. 올바르게는 ㉠ 트리거, ㉡ 저장 프로시저, ㉢ 주장이 아니라 인덱스이다.
2번: X. ㉠을 주장으로 둔 것이 틀렸다. 주장은 자동 실행 절차가 아니라 데이터베이스가 만족해야 할 제약 조건이다. ㉡을 인덱스로 둔 것도 틀렸다. 올바르게는 ㉠ 트리거, ㉡ 저장 프로시저, ㉢ 인덱스이다.
3번: X. ㉠을 주장으로, ㉡을 인덱스로, ㉢을 트리거로 연결한 부분이 모두 맞지 않는다. 자동 실행은 트리거, 컴파일 후 호출 실행은 저장 프로시저, 빠른 탐색용 구조는 인덱스이다.
4번: O. ㉠은 데이터 변경 이벤트에 반응해 자동 수행되므로 트리거이고, ㉡은 DBMS에 저장된 작업 절차를 컴파일 후 실행하므로 저장 프로시저이며, ㉢은 탐색 속도를 높이기 위한 데이터 구조이므로 인덱스가 맞다.

오답 포인트

트리거와 저장 프로시저를 모두 '미리 작성한 SQL 실행 기능'으로만 외우면 혼동한다. 자동 실행이면 트리거이고, 호출해서 실행하는 절차 단위이면 저장 프로시저이다. 주장과 인덱스도 각각 제약 조건과 탐색 구조라는 성격 차이를 구별해야 한다.

다시 나오면 이렇게 판단

문장에 '자동으로 수행'이 있으면 트리거를 먼저 떠올리고, '컴파일된 후 실행되는 작업 단위'이면 저장 프로시저로 본다. '빠르게 탐색'이라는 표현이 나오면 인덱스로 판단하고, '항상 만족해야 하는 조건'이면 주장으로 구분한다.

암기 포인트

트리거는 자동 실행, 저장 프로시저는 저장된 절차 호출 실행, 인덱스는 빠른 탐색, 주장은 무결성 조건이다.

NoteBase

문 4. 데이터베이스 보안과 관련한 설명으로 옳지 않은 것은?

① SQL 삽입(injection) 공격은 공격자가 악의적으로 만든 SQL 명령을 응용 프로그램이 수행하도록 하는 것이다.

② 데이터베이스 관리자(DBA)가 각 사용자에게 데이터베이스에 대한 접근 권한을 부여하거나 취소할 때 grant 명령을 사용한다.

③ 데이터베이스에 대한 권한은 역할(role)에도 부여할 수 있다.

④ 전자 서명(digital signature)은 공개키 암호화 기법의 특성을 이용하여 인증을 수행한다.

정답 2번

출제 포인트

이 문제는 데이터베이스 보안에서 SQL 삽입, 권한 부여·회수 명령, 역할(role), 전자서명의 기본 개념을 구분하는 문제이다.

핵심 개념

SQL 삽입은 입력값에 악성 SQL을 끼워 넣어 응용 프로그램이 의도하지 않은 SQL을 실행하게 만드는 공격이다.
GRANT는 권한 부여 명령이고, 권한 취소는 REVOKE 명령으로 수행한다.
역할(role)은 여러 권한을 묶어 사용자에게 간접 부여하기 위한 권한 집합이다.
전자서명은 공개키 암호 방식의 특성을 이용해 송신자 인증과 무결성 확인을 수행한다.

NoteBase

선지별 해설

① O
SQL 삽입 공격의 정의를 그대로 설명한 선지이다. 공격자가 만든 SQL 조각이 입력값을 통해 삽입되어 프로그램이 그 명령을 실행하게 만든다.

② X
틀린 부분은 "접근 권한을 부여하거나 취소할 때 grant 명령"이라는 표현이다.
권한을 부여할 때는 GRANT를 사용하지만, 취소할 때는 REVOKE를 사용한다. 올바르게는 "접근 권한을 부여할 때는 GRANT, 취소할 때는 REVOKE 명령을 사용한다"이다.

③ O
데이터베이스 권한은 사용자 개인뿐 아니라 역할(role)에도 부여할 수 있다. 역할에 권한을 묶어 두고 사용자는 그 역할을 부여받는 방식으로 관리한다.

④ O
전자서명은 공개키 암호 기술을 바탕으로 서명자의 신원을 확인하고 메시지가 변조되지 않았는지 검증하는 데 쓰인다. 인증 기능을 수행한다는 설명은 타당하다.

오답 포인트

GRANT와 REVOKE를 한 쌍으로 외우지 않으면 ②처럼 권한 부여와 권한 회수 명령을 뒤섞기 쉽다. 보안 용어 자체는 익숙해 보여도 명령어의 기능 방향이 반대로 출제되는 점이 함정이다.

다시 나오면 이렇게 판단

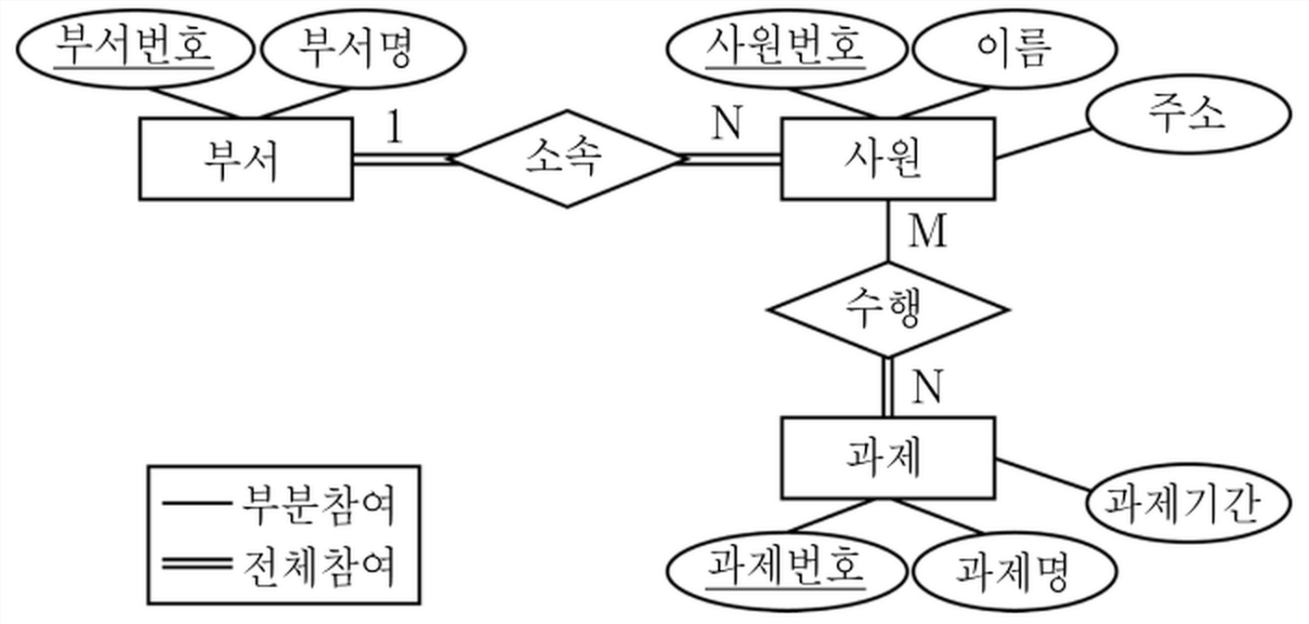
보안 문항에서는 먼저 용어의 한 줄 정의를 대응시키고, 다음으로 명령어의 기능 방향을 확인하면 된다. 특히 권한 관리 문제는 "부여는 GRANT, 회수는 REVOKE"처럼 짝을 맞춰 판단하면 오답을 줄일 수 있다.

암기 포인트

GRANT는 준다, REVOKE는 거둔다.

NoteBase

문 5. 다음 ERD(Entity-Relationship Diagram)에 대한 설명으로 옳지 않은 것은? (단, 속성 이름에 대한 밑줄은 기본키이다)



① 주어진 ERD를 릴레이션으로 사상(mapping)하면, 4개의 릴레이션이 생성된다.

② 모든 사원은 적어도 하나 이상의 과제를 수행하여야 한다.

③ 모든 부서는 적어도 1명 이상의 사원이 존재하여야 하며, 사원도 반드시 하나의 부서에 소속되어야 한다.

④ 사원은 2개의 부서에 동시에 소속될 수 없다.

정답 2번

출제 포인트

ERD의 카디널리티와 전체참여/부분참여를 읽어 각 개체의 참여 제약과 릴레이션 사상 결과를 해석하는 문제이다.

NoteBase

선지별 해설

① O
부서, 사원, 과제의 3개 개체는 각각 릴레이션이 되고, M:N 관계인 수행은 별도 릴레이션이 추가된다. 반면 1:N 관계인 소속은 N쪽인 사원 릴레이션에 부서번호를 외래키로 포함하면 되므로 총 4개 릴레이션이 생성된다.

② X
사원-수행 관계에서 사원 쪽은 부분참여이고 과제 쪽은 전체참여이다. 따라서 모든 사원이 과제를 하나 이상 수행해야 하는 것은 아니다. 틀린 표현은 "모든 사원은 적어도 하나 이상의 과제를 수행하여야 한다"이고, 올바르게는 "모든 과제는 적어도 한 명 이상의 사원에 의해 수행되어야 하지만 사원은 과제를 수행하지 않을 수도 있다"이다.

③ O
부서-소속 관계는 부서와 사원 모두 전체참여이며 카디널리티는 1:N이다. 따라서 모든 부서에는 적어도 한 명 이상의 사원이 있어야 하고, 모든 사원도 반드시 하나의 부서에 소속되어야 한다.

④ O
소속 관계의 부서:사원 카디널리티가 1:N이므로 한 사원은 하나의 부서에만 소속된다. 따라서 한 사원이 두 개의 부서에 동시에 소속되는 것은 허용되지 않는다.

오답 포인트

전체참여 표시인 이중선이 어느 쪽에 붙어 있는지를 반대로 읽으면 ②를 쉽게 옳다고 오해한다. 수행 관계에서는 사원이 아니라 과제 쪽만 전체참여라는 점이 함정이다.

다시 나오면 이렇게 판단

ERD 문제는 먼저 관계의 카디널리티가 1:N인지 M:N인지 확인하고, 다음으로 이중선이 붙은 쪽이 전체참여인지 읽으면 된다. 그 뒤 1:N 관계는 외래키 포함으로, M:N 관계는 별도 릴레이션 생성으로 연결하면 선지를 빠르게 판단할 수 있다.

암기 포인트

1:N 관계는 보통 외래키로 흡수되고, M:N 관계는 별도 릴레이션이 되며, 이중선은 해당 개체의 전체참여를 뜻한다.

NoteBase

문 6. 다음 함수종속성 집합 FD의 최소커버(minimal cover) FD_{min}는?

$FD = \{Y \rightarrow X,\ Z \rightarrow XYW\}$

① $FD_{min} = \{Y \rightarrow X,\ Z \rightarrow X,\ Z \rightarrow Y,\ Z \rightarrow W\}$
② $FD_{min} = \{Y \rightarrow X,\ Z \rightarrow Y,\ Z \rightarrow W\}$
③ $FD_{min} = \{Y \rightarrow X,\ Z \rightarrow X,\ Z \rightarrow W\}$
④ $FD_{min} = \{Y \rightarrow X,\ Z \rightarrow X,\ Z \rightarrow Y\}$

정답 2번

출제 포인트

이 문제는 함수종속성 집합을 우변 단일 속성으로 분해한 뒤, 다른 종속성으로 유도되는 중복 종속성을 제거하여 최소커버를 구할 수 있는지를 묻는다.

핵심 개념

최소커버는 원래 함수종속성과 동등하면서 각 함수종속성의 우변이 한 속성이고, 좌변에 불필요한 속성이 없으며, 집합 안에 중복 함수종속성이 없는 형태이다.
이 문제에서는 $Y \rightarrow X, Z \rightarrow XYW$를 $Y \rightarrow X, Z \rightarrow X, Z \rightarrow Y, Z \rightarrow W$로 분해한 뒤 중복 여부를 본다.
여기서 $Z \rightarrow X$는 $Z \rightarrow Y$와 $Y \rightarrow X$로 유도되므로 제거할 수 있다.

선지별 해설

1번: X이다. $Z \rightarrow X$가 불필요하다. 이미 $Z \rightarrow Y$와 $Y \rightarrow X$가 있으므로 $Z \rightarrow X$는 추론된다. 틀린 표현은 최소커버에 $Z \rightarrow X$를 남겨 둔 부분이고, 올바르게는 $\{Y \rightarrow X, Z \rightarrow Y, Z \rightarrow W\}$이다.
2번: O이다. $Z \rightarrow XYW$를 분해하면 $Z \rightarrow X, Z \rightarrow Y, Z \rightarrow W$가 되고, 이 중 $Z \rightarrow X$는 $Z \rightarrow Y, Y \rightarrow X$로 대체되므로 제거된다. 따라서 $\{Y \rightarrow X, Z \rightarrow Y, Z \rightarrow W\}$가 최소커버이다.
3번: X이다. 이 집합에서는 $Z \rightarrow Y$를 직접 얻을 수 없고, $Z^+ = \{Z, X, W\}$에 Y가 포함되지 않는다. 틀린 표현은 $Z \rightarrow Y$를 빠뜨린 부분이고, 올바르게는 $Z \rightarrow Y$를 포함해야 한다.
4번: X이다. 이 집합에서는 $Z \rightarrow W$를 유도할 수 없어 원래 함수종속성 집합과 동등하지 않다. 또한 $Z \rightarrow X$도 중복이다. 틀린 표현은 $Z \rightarrow W$가 없는 부분이고, 올바르게는 $Z \rightarrow W$를 포함하고 중복인 $Z \rightarrow X$는 제거해야 한다.

오답 포인트

우변 분해까지만 하고 끝내면 1번처럼 중복 종속성을 남기기 쉽다. 반대로 3번이나 4번처럼 유도 가능성과 동등성을 끝까지 확인하지 않으면 필요한 종속성을 빠뜨리게 된다.

NoteBase

다시 나오면 이렇게 판단

먼저 우변을 한 속성씩 분해하고, 그다음 각 함수종속성을 하나씩 빼 본 뒤 나머지 종속성들로 다시 유도되는지 확인한다. 유도되면 중복이므로 제거하고, 유도되지 않으면 남겨 두면 된다.

암기 포인트

최소커버는 분해 후 중복 제거이다. 이 문제에서는 $Z \to Y$, $Y \to X$가 있으면 $Z \to X$는 지워진다.

NoteBase

문 7. 다음 인덱스 기법에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. B-트리 전체를 노드 내의 키 값에 따라 순차 검색하기 위해서는 트리의 각 노드를 전위 순회(preorder traversal)한다.
ㄴ. 밀집(dense) 인덱스는 희소(sparse) 인덱스에 비해 액세스 시간은 빠르지만 더 많은 공간을 필요로 한다.
ㄷ. B-트리에서 오버플로가 발생하여 리프 노드가 분할될 때, 중간 키 값이 부모 노드뿐만 아니라 새로 분할된 노드에도 저장된다.
ㄹ. 1,000명의 사원 정보가 저장된 '사원' 테이블의 '부서' 필드에 대하여, 30명이 소속된 '총무과' 사원에 대한 비트맵 인덱스를 구성할 경우 1,000비트가 필요하다.

① ㄱ, ㄷ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄴ, ㄹ

정답 4번

출제 포인트

이 문제는 B-트리 순회 방식, 밀집 인덱스와 희소 인덱스의 차이, B-트리와 B+트리의 분할 방식, 비트맵 인덱스의 비트 수를 구분하는 문제이다.

핵심 개념

B-트리는 키 순서 검색을 위해 중위 순회 방식으로 접근하는 균형 트리 인덱스이다.
밀집 인덱스는 모든 탐색 키 값 또는 레코드에 대해 인덱스 항목을 두므로 빠르지만 공간을 더 사용한다.
희소 인덱스는 일부 탐색 키에만 인덱스 항목을 두므로 공간은 적게 들지만 원하는 레코드까지 추가 탐색이 필요하다.
B-트리는 노드 분할 시 중간 키가 부모로 올라가고 원래 노드에서는 빠진다.
B+트리는 리프 분할 시 분리 키가 부모에 복사되고 리프에도 남을 수 있다.
비트맵 인덱스는 특정 값마다 전체 튜플 수만큼의 비트 벡터를 둔다.

NoteBase

기출OX 해설집 · 공무원/국가직/7급 · 데이터베이스론 · 2019

선지별 해설

① X
ㄱ은 틀리다. B-트리 전체를 키 값 순서대로 읽으려면 전위 순회가 아니라 중위 순회에 해당하는 방식으로 방문해야 한다.
ㄷ도 틀리다. B-트리에서 리프가 분할되면 중간 키는 부모 노드로 올라가고, 새로 분할된 자식 노드에 같은 키를 중복 저장하지 않는다. 이 설명은 B+트리의 리프 분할 특징에 가깝다.
따라서 ①의 조합은 옳지 않고, 올바른 조합은 ㄴ, ㄹ이다.

② X
ㄱ이 틀리므로 ㄱ, ㄹ 조합은 성립하지 않는다. ㄹ은 특정 부서값인 '총무과'에 대해 전체 1,000명 각각을 1비트로 표시하므로 1,000비트가 필요하다는 뜻에서 옳다.
따라서 ②의 잘못된 부분은 ㄱ을 옳은 설명으로 포함한 점이며, 올바르게는 ㄴ, ㄹ 조합이다.

③ X
ㄴ은 옳다. 밀집 인덱스는 희소 인덱스보다 인덱스 항목이 많아 공간을 더 쓰지만 원하는 레코드에 더 직접 접근할 수 있어 일반적으로 더 빠르다.
그러나 ㄷ은 틀리다. B-트리에서는 중간 키가 부모로 이동할 뿐 분할된 리프에도 함께 저장되지 않는다.
따라서 ③의 잘못된 부분은 ㄷ을 옳다고 본 점이며, 올바르게는 ㄴ, ㄹ 조합이다.

④ O
ㄴ은 밀집 인덱스와 희소 인덱스의 전형적인 공간-성능 교환관계를 설명하므로 옳다.
ㄹ은 비트맵 인덱스가 레코드 수만큼의 비트열을 사용한다는 점을 묻는 설명이므로 옳다. 1,000명 중 30명이 총무과이면 1,000비트 중 30비트만 1이고 나머지는 0이 된다.
따라서 옳은 것만 묶은 선택지는 ④이다.

오답 포인트

ㄷ에서 B-트리와 B+트리의 분할 방식을 뒤섞으면 쉽게 틀린다. 또한 순차 검색이라는 표현을 보고 트리 순회를 막연히 전위 순회로 연결하면 ㄱ을 맞는 설명으로 오인하기 쉽다.

다시 나오면 이렇게 판단

트리 인덱스 문제는 먼저 순서 검색이면 중위 순회 계열인지, 분할 설명이면 B-트리와 B+트리 중 어느 구조인지 구분하면 된다. 비트맵 인덱스 문제는 특정 값의 등장 횟수가 아니라 전체 레코드 수만큼 비트가 필요하다는 기준으로 판단한다.

암기 포인트

밀집은 빠르고 크며, 희소는 작고 느리다. B-트리는 중간 키를 올리고, 비트맵은 전체 행 수만큼 비트를 둔다.

NoteBase

문 8. 다음은 시스템 고장이 발생할 때 트랜잭션 T_1, T_2, T_3, T_4, T_5를 복구하기 위해 고장 전에 마지막으로 기록된 DBMS 로그이다. 이 로그를 사용하여 즉시 갱신 회복 기법의 undo-redo 알고리즘을 수행할 때, 회복과정에 대한 설명으로 옳지 않은 것은? (단, <checkpoint>는 검사점 기록 로그 레코드이며, <T_1, A, 200, 400>은 "T_1이 데이터 항목 A의 현재 값 200을 400으로 갱신한다"를 의미하는 로그 레코드이다. 표현되지 않은 로그의 다른 속성은 고려하지 않는다.)

로그 번호	로그 레코드
1	<T_1 start>
2	<T_1, A, 200, 400>
3	<T_1 commit>
4	<T_2 start>
5	<T_2, B, 900, 1900>
6	<checkpoint>
7	<T_3 start>
8	<T_3, A, 400, 700>
9	<checkpoint>
10	<T_5 start>
11	<T_5, C, 100, 5000>
12	<T_3 commit>
13	<T_4 start>
14	<T_4, D, 700, 1300>
15	<T_5 commit>
16	<T_4, E, 1300, 1500>
	--시스템 고장발생--

① T_1은 어떠한 undo와 redo 연산도 수행하지 않는다.

② 데이터 항목 A와 B의 값은 각각 700과 900으로 갱신된다.

③ 데이터 항목 C와 D의 값은 각각 5000과 700으로 갱신된다.

④ redo 연산은 T_5, T_3 순서로 수행된다.

정답 4번

출제 포인트

즉시 갱신 회복 기법의 undo-redo 알고리즘에서 마지막 checkpoint를 기준으로 redo 대상과 undo 대상을 구분하고 수행 방향을 판단하는 문제이다.

NoteBase

올바른 순서

1. 마지막 checkpoint인 9번 로그를 기준으로 회복 대상을 분류한다.
2. checkpoint 이후 commit된 트랜잭션인 T3, T5를 redo 대상으로 잡는다.
3. 장애 시점에 완료되지 않은 T4와 checkpoint 이전에 시작했지만 끝나지 않은 T2를 undo 대상으로 잡는다.
4. redo는 로그의 순방향으로, undo는 로그의 역방향으로 수행한다.

왜 이 순서: redo는 확정된 갱신을 다시 반영하는 과정이므로 앞에서 뒤로 적용하고, undo는 최근 갱신부터 이전값으로 되돌려야 하므로 뒤에서 앞으로 적용한다.

선지별 해설

① O

T1은 3번에서 이미 commit되었고 그 뒤 checkpoint가 기록되었으므로 회복 시 추가 undo나 redo 대상이 아니다.

② O

A는 T3의 갱신 $< T3, A, 400, 700 >$을 redo하므로 700이 된다. B는 commit되지 않은 T2의 갱신 $< T2, B, 900, 1900 >$을 undo하므로 900으로 돌아간다.

③ O

C는 commit된 T5의 갱신을 redo하므로 5000이 된다. D는 미완료인 T4의 갱신 $< T4, D, 700, 1300 >$을 undo하므로 700으로 돌아간다.

④ X

틀린 부분은 "redo 연산은 T5, T3 순서"라는 표현이다. redo는 commit 순서로 묶어서 처리하는 것이 아니라 로그를 앞에서 뒤로 읽으며 수행한다. 올바르게는 T3의 갱신 로그가 T5의 갱신 로그보다 먼저 나오므로 redo는 T3, T5 순서로 수행된다.

오답 포인트

checkpoint 이후에 commit된 것만 보고 T2를 회복 대상에서 빼거나, redo를 commit 완료 순서로 착각하면 ④를 판별하지 못한다. checkpoint 이전에 시작한 미완료 트랜잭션도 undo 대상이며, redo는 로그의 순방향이라는 점이 함정이다.

다시 나오면 이렇게 판단

즉시 갱신 회복 문제는 마지막 checkpoint를 찾은 뒤 미완료 트랜잭션은 undo, checkpoint 이후 commit된 트랜잭션은 redo로 먼저 분류한다. 그다음 redo는 로그의 순방향, undo는 로그의 역방향으로 적용하면 선지 판단이 정리된다.

NoteBase

암기 포인트

즉시 갱신 회복은 미완료는 undo, 완료는 redo이며 redo는 순방향, undo는 역방향이다.

NoteBase

문 9. <보기 1>에서 공급업체와 부품, 카탈로그 테이블을 생성하는 SQL문을 수행한 후 튜플을 삽입하여 세 테이블의 상태가 다음과 같을 때, <보기 2>의 SQL문을 수행한 결과로 옳은 것은?

<보기 1>

```
CREATE TABLE 공급업체 (
 업체번호 INT NOT NULL,
 업체명 VARCHAR(20),
 PRIMARY KEY(업체번호));
CREATE TABLE 부품 (
 부품번호 INT NOT NULL,
 부품명 VARCHAR(20),
 색상 VARCHAR(20),
 PRIMARY KEY(부품번호));
CREATE TABLE 카탈로그 (
 업체번호 INT NOT NULL,
 부품번호 INT NOT NULL,
 가격 INT,
 PRIMARY KEY(업체번호, 부품번호),
 FOREIGN KEY(업체번호) REFERENCES 공급업체(업체번호),
 FOREIGN KEY(부품번호) REFERENCES 부품(부품번호));
```

공급업체

업체번호	업체명
1	공급A
2	공급B
3	공급C

부품

부품번호	부품명	색상
1	부품1	빨강
2	부품2	파랑
3	부품3	노랑

카탈로그

업체번호	부품번호	가격
1	3	10000
2	1	20000
3	2	30000

<보기 2>

```
SELECT 업체번호, 업체명
FROM 공급업체
WHERE NOT EXISTS (
        SELECT 부품.부품번호
        FROM 부품
        WHERE 부품.색상='빨강' AND EXISTS (
                SELECT *
                FROM 카탈로그
                WHERE 카탈로그.부품번호=부품.부품번호
                AND 카탈로그.업체번호=공급업체.업체번호));
```

① 업체번호, 업체명

② 업체번호: 2, 업체명: 공급B

NoteBase

③ 업체번호: 1, 업체명: 공급A
업체번호: 3, 업체명: 공급C

④ 업체번호: 1, 업체명: 공급A
업체번호: 2, 업체명: 공급B
업체번호: 3, 업체명: 공급C

정답 3번

출제 포인트

이 문제는 상관 서브쿼리의 EXISTS, NOT EXISTS를 이용해 특정 조건의 부품을 공급하지 않는 업체를 찾을 수 있는지를 묻는다.

자료 읽는 순서

1. 부품 테이블에서 색상이 빨강인 부품번호를 찾는다.
2. 카탈로그 테이블에서 그 빨강 부품을 공급한 업체번호가 누구인지 확인한다.
3. 바깥 질의의 NOT EXISTS가 빨강 부품을 공급한 적이 없는 업체만 남긴다는 점을 선택지와 연결한다.

선지별 해설

1번: X. 결과가 공집합은 아니다. 부품 테이블에서 빨강은 부품번호 1이고, 카탈로그에서 부품번호 1을 공급한 업체는 업체번호 2뿐이다. 따라서 빨강 부품을 공급한 적이 없는 업체번호 1과 3은 남는다. 틀린 표현은 조회 결과가 없다는 점이며, 올바르게는 업체번호 1 공급A와 업체번호 3 공급C가 조회된다.
2번: X. 업체번호 2 공급B는 빨강 부품인 부품번호 1을 카탈로그에서 공급하고 있으므로 서브쿼리의 EXISTS가 참이 된다. 그러면 바깥쪽의 NOT EXISTS를 만족하지 못해 제외된다. 틀린 표현은 공급B가 결과에 포함된다는 점이며, 올바르게는 공급B는 결과에서 제외된다.
3번: O. 빨강 부품은 부품번호 1뿐인데, 카탈로그에서 이를 공급한 업체는 2뿐이다. 업체번호 1은 부품번호 3만, 업체번호 3은 부품번호 2만 공급하므로 빨강 부품 공급 이력이 없다. 따라서 결과는 업체번호 1 공급A와 업체번호 3 공급C이다.
4번: X. 모든 업체가 조회되려면 어떤 업체도 빨강 부품을 공급하지 않아야 한다. 그러나 업체번호 2는 빨강 부품번호 1을 공급한다. 틀린 표현은 세 업체 모두 결과에 포함된다는 점이며, 올바르게는 업체번호 2는 제외되고 1과 3만 조회된다.

오답 포인트

NOT EXISTS를 모든 부품을 공급하지 않는 업체로 잘못 읽으면 틀린다. 이 질의는 빨강 부품 중에서 해당 업체가 공급한 것이 하나라도 있으면 제외하는 구조이므로 조건의 기준이 전체 부품이 아니라 빨강 부품이라는 점이 함정이다.

NoteBase

다시 나오면 이렇게 판단

상관 서브쿼리에서 NOT EXISTS가 나오면 먼저 내부 조건으로 걸러지는 대상 집합을 찾고, 그 집합에 연결되는 행이 하나라도 있는 업체는 제외된다고 판단한다.

암기 포인트

NOT EXISTS는 조건을 만족하는 관련 행이 하나도 없을 때만 참이 된다.

NoteBase

문 10. 다음 SQL문에 대하여 질의 최적화를 수행하고자 한다. 이에 대한 설명으로 옳지 않은 것은?

```
SELECT 학생.이름, 교수.이름
FROM   학생, 학과, 교수
WHERE  학생.학과코드 = 학과.학과코드
  AND 교수.교수코드 = 학과.학과장코드
  AND 학과.학과코드 = 'CS' ;
```

① 카탈로그는 해당 테이블에 대한 통계 정보를 저장하고 있어 비용 계산에 활용될 수 있지만, 통계 정보를 실시간으로 갱신하려면 부하가 커서 주기적으로 갱신하기도 한다.

② 조인연산 학생 _학과코드 = 학과코드 학과는 기본적으로 중첩 루프(nested loop)를 이용하여 구현하지만, '학과.학과코드'에 인덱스가 구축되어 있다면 비용을 감소시킬 수 있다.

③ 경험적 질의 최적화 기법에서는 주어진 SQL문의 WHERE절에서 조인연산보다 학과.학과코드 = 'CS'를 먼저 수행하도록 한다.

④ $|R|$을 릴레이션 R의 튜플의 수로 정의할 때, $R \bowtie_{A=B} S$에 대하여 A가 R의 기본키이면 $|(R \bowtie_{A=B} S)| \leq |S|$이고 조인 선택률($js$)은 $js > 1/|R|$이다.

정답 4번

출제 포인트

선택 연산의 푸시다운, 인덱스 중첩 루프 조인, 조인 결과 카디널리티와 조인 선택률의 관계를 묻는 문제이다.

핵심 개념

카탈로그는 옵티마이저가 비용 추정을 위해 사용하는 메타데이터와 통계 정보 저장소이다 / 비용 계산 축: 통계 정보의 최신성과 유지 비용 / 키워드: 통계정보, 주기적 갱신

경험적 질의 최적화는 결과가 같은 연산 중 중간 결과를 줄이는 순서로 재배치하는 규칙 기반 최적화이다 / 결정 차이축: 조인 전 선택 연산 수행 여부 / 키워드: selection pushdown

조인 선택률은 $js = |R \bowtie S|/(|R||S|)$로 정의되는 비율이다 / 결정 차이축: 조인 결과 건수 상한과 선택률 부등호 / 키워드: 카디널리티 상한, $js \leq 1/|R|$

NoteBase

선지별 해설

① O
카탈로그에는 릴레이션 크기, 값 분포, 인덱스 유무 같은 통계가 저장되어 비용 기반 최적화에 활용된다.
통계를 매 갱신 시점마다 실시간 반영하면 오버헤드가 커질 수 있어 주기적으로 수집하는 방식도 맞다.

② O
중첩 루프 조인은 가장 기본적인 조인 구현 방식 가운데 하나이다.
내부 릴레이션인 학과의 조인 속성 학과.학과코드에 인덱스가 있으면 각 학생 튜플마다 학과를 빠르게 찾는 인덱스 중첩 루프 조인이 가능해져 비용을 줄일 수 있다.

③ O
경험적 최적화는 보통 선택 연산을 조인보다 먼저 내려 보내 중간 결과를 줄인다.
따라서 학과.학과코드 = 'CS'를 먼저 적용한 뒤 필요한 학과 튜플만 조인하는 설명은 옳다.

④ X
맞는 부분은 A가 R의 기본키이면 각 S 튜플이 R의 최대 한 튜플과만 매칭되므로 $|R \bowtie_{A=B} S| \le |S|$라는 점이다.
틀린 부분은 조인 선택률(js)은 js > 1/|R|라는 표현이다.
조인 선택률은 $js = |R \bowtie_{A=B} S|/(|R||S|)$이므로 위 상한을 대입하면 $js \le 1/|R|$가 된다.
틀린 표현은 js > 1/|R|이고, 올바르게는 $js \le 1/|R|$이다.

오답 포인트

4번은 앞문장 카디널리티 상한은 맞고 뒷문장 부등호만 뒤집어 놓아 전체를 맞는 설명처럼 보이게 만든 함정이다.
2번도 `기본적으로`라는 말을 보고 특정 구현만 가능하다고 오해하면 흔들릴 수 있으나, 문맥상 기본 조인 방식과 인덱스 활용 효과를 설명한 문장이다.

다시 나오면 이렇게 판단

질의 최적화 문제는 선택 연산의 선행 여부, 인덱스가 조인 비용에 미치는 영향, 조인 결과 건수 상한을 먼저 본다.
특히 결과 건수 상한이 주어지면 선택률은 그 상한을 $|R||S|$로 나눈 값 이하라는 방향으로 판단한다.

암기 포인트

선택은 조인 앞으로 보낸다, 인덱스는 중첩 루프를 줄인다, 기본키 조인은 결과 건수 상한이 생기므로 선택률 부등호는 대개 $\le$ 방향이다.

NoteBase

문 11. 두 릴레이션 T1, T2에 관계 대수 연산 ㉠ ~ ㉢을 수행한 결과 릴레이션이 각각 R1, R2, R3과 같다. 연산을 바르게 연결한 것은?

T_1

A	B
aa	10
bb	20
cc	30
dd	20

T_2

B	C
20	xx
30	yy
40	zz

$R_1 \leftarrow T_1$ ㉠ T_2

A	B	C
aa	10	
bb	20	
cc	30	
dd	20	
	20	xx
	30	yy
	40	zz

$R_2 \leftarrow T_1$ ㉡ T_2

A	B	C
aa	10	
bb	20	xx
cc	30	yy
dd	20	xx
	40	zz

$R_3 \leftarrow T_1$ ㉢ T_2

A	B	C
bb	20	xx
dd	20	xx
cc	30	yy
	40	zz

① ㉠: 완전 외부조인(⟗), ㉡: 외부 합집합(U+), ㉢: 왼쪽 외부조인(⟕)

② ㉠: 완전 외부조인(⟗), ㉡: 외부 합집합(U+), ㉢: 오른쪽 외부조인(⟖)

③ ㉠: 외부 합집합(U+), ㉡: 완전 외부조인(⟗), ㉢: 왼쪽 외부조인(⟕)

④ ㉠: 외부 합집합(U+), ㉡: 완전 외부조인(⟗), ㉢: 오른쪽 외부조인(⟖)

정답 4번

출제 포인트

외부 합집합과 완전 외부조인, 오른쪽 외부조인을 결과 릴레이션의 튜플 보존 방식과 NULL 채움 방식으로 구별하는 문제이다.

핵심 개념

외부 합집합(U+)은 서로 다른 속성을 가진 두 릴레이션을 공통 스키마로 늘려 부족한 속성은 NULL로 채운 뒤 행을 그대로 합치는 연산이다.
완전 외부조인은 조인 속성값이 같으면 한 행으로 결합하고, 양쪽 릴레이션의 미일치 행도 모두 NULL을 채워 남기는 연산이다.
오른쪽 외부조인은 조인 결과에 더해 오른쪽 릴레이션의 미일치 행만 NULL을 채워 보존하는 연산이다.
왼쪽 외부조인은 반대로 왼쪽 릴레이션의 미일치 행만 보존한다.

NoteBase

선지별 해설

① X
㉠을 완전 외부조인으로 본 부분이 틀리다. R1은 B=20, 30인 매칭 행을 하나로 합치지 않고 T1의 4개 행과 T2의 3개 행을 각각 NULL 확장만 해서 보여 주므로 외부 합집합이다.
㉢을 왼쪽 외부조인으로 본 부분도 틀리다. R3은 T1의 미일치 행 aa-10은 남지 않고, T2의 미일치 행 B=40, zz는 남으므로 오른쪽 외부조인이다.
올바르게는 ㉠ 외부 합집합, ㉡ 완전 외부조인, ㉢ 오른쪽 외부조인이다.

② X
㉢을 오른쪽 외부조인으로 본 부분은 맞지만, ㉠을 완전 외부조인으로 본 부분이 틀리다. R1은 조인된 행이 전혀 없고 두 릴레이션의 행을 NULL로 확장해 이어 붙인 모양이므로 외부 합집합이다.
틀린 표현은 '㉠ 완전 외부조인'이다. 올바르게는 '㉠ 외부 합집합'이다.

③ X
㉠을 외부 합집합, ㉡을 완전 외부조인으로 본 부분은 맞다. 틀린 부분은 ㉢을 왼쪽 외부조인으로 본 표현이다. R3에서는 오른쪽 릴레이션 T2의 미일치 행 B=40, zz가 남고, 왼쪽 릴레이션 T1의 미일치 행 aa-10은 남지 않는다. 따라서 왼쪽 외부조인이 아니라 오른쪽 외부조인이다.
틀린 표현은 '㉢ 왼쪽 외부조인'이다. 올바르게는 '㉢ 오른쪽 외부조인'이다.

④ O
R1은 외부 합집합이다. T1의 행들은 C가 NULL인 채 남고, T2의 행들은 A가 NULL인 채 남는다.
R2는 완전 외부조인이다. B=20, 30은 한 행으로 결합되고, T1의 aa-10과 T2의 40-zz 같은 미일치 행도 모두 남는다.
R3는 오른쪽 외부조인이다. 매칭 행에 더해 오른쪽 릴레이션 T2의 미일치 행 B=40, zz만 추가로 보존된다.

오답 포인트

외부 합집합과 외부조인을 모두 NULL이 생기는 연산으로만 외우면 혼동된다. 이 문제는 '매칭 행을 하나로 합치느냐'와 '미일치 행을 어느 쪽까지 살리느냐'를 동시에 물었다. 특히 R3에서 B=40, zz는 남고 aa-10은 사라진다는 점을 놓치면 왼쪽과 오른쪽을 뒤집기 쉽다.

다시 나오면 이렇게 판단

먼저 공통 속성값이 일치하는 행이 한 행으로 합쳐졌는지 본다. 합쳐지지 않았으면 외부 합집합이다. 합쳐졌다면 미일치 행이 양쪽 다 남는지, 왼쪽만 남는지, 오른쪽만 남는지를 확인해 완전 외부조인, 왼쪽 외부조인, 오른쪽 외부조인을 구분한다.

NoteBase

암기 포인트

외부 합집합은 행을 붙이고, 외부조인은 행을 합친다. 오른쪽 외부조인은 오른쪽의 고아 행을 남긴다.

NoteBase

문 12. SQL에서 뷰(view)의 역할에 대한 설명으로 옳지 않은 것은?

① 기본 테이블(base table)들만으로 작성된 질의를 간소화시킬 수 있다.

② 사용자의 접근권한에 따라 동일한 기본 테이블의 속성들을 선택적으로 제공할 수 있다.

③ 기본 테이블들만으로 작성된 질의 처리 성능을 향상시키기 위해 개발되었다.

④ 기본 테이블들의 물리적 구조를 변경시키지 않고 사용자가 원하는 새로운 가상 테이블을 생성시킬 수 있다.

정답 3번

출제 포인트

이 문제는 SQL 뷰의 주된 역할이 질의 단순화, 보안, 논리적 독립성 제공인지, 아니면 성능 향상 자체인지 구분하는 문제이다.

핵심 개념

뷰는 기본 테이블에 대한 질의를 이름 붙여 만든 가상 테이블이다. 결정 차이축은 실제 데이터를 물리적으로 저장하는지와 주된 목적이 무엇인지이다. 일반 뷰의 키워드는 가상, 질의 단순화, 접근 제어, 논리적 독립성이다. 반면 성능 최적화는 인덱스나 물질화 뷰 같은 별도 저장·최적화 구조와 더 직접적으로 연결된다.

선지별 해설

① O: 뷰는 기본 테이블만으로 작성한 복잡한 질의를 미리 정의한 가상 테이블 형태로 제공하므로 사용 시 질의를 간소화할 수 있다.
② O: 뷰는 필요한 속성만 노출하도록 정의할 수 있으므로 사용자 접근권한에 따라 동일한 기본 테이블의 일부 속성만 선택적으로 제공할 수 있다.
③ X: 틀린 부분은 "질의 처리 성능을 향상시키기 위해 개발되었다"이다. 일반 뷰의 주된 목적은 편의성, 보안성, 논리적 데이터 독립성 제공이다. 성능 향상은 인덱스나 물질화 뷰 같은 별도의 최적화 기법으로 다루는 것이 맞다.
④ O: 뷰는 기본 테이블의 물리적 구조를 바꾸지 않고 사용자 관점에서 필요한 형태의 가상 테이블을 정의할 수 있으므로 맞는 설명이다.

오답 포인트

일반 뷰와 물질화 뷰를 섞어서 생각하면 ③을 맞는 설명처럼 보기 쉽다. 이 문항은 "뷰가 가상 테이블로서 사용자에게 무엇을 편하게 해 주는가"와 "실행 성능을 직접 높이는 물리적 최적화 수단인가"를 구분하는지가 함정이다.

다시 나오면 이렇게 판단

보기에서 뷰의 역할이 질의 단순화, 필요한 열만 공개하는 보안, 사용자별 논리적 테이블 제공이면 맞는 설명으로 본다. 반대로 실행 성능 향상 자체가 뷰의 본래 목적이라고 서술하면 일반 뷰 기준으로 틀린 설명으로 판단한다.

NoteBase

암기 포인트

일반 뷰는 가상 테이블로서 편의·보안·논리적 독립성을 담당하고, 성능 최적화는 인덱스나 물질화 뷰와 연결된다.

NoteBase

문 13. 데이터베이스와 데이터베이스 관리시스템(DBMS)에 대한 설명으로 옳지 않은 것은?

① 데이터 종속성(data dependency)을 유지하여 데이터와 이를 이용하는 프로그램이 밀접하게 연결되어 동작하도록 도와준다.

② 데이터베이스 사용자들에게 공용 데이터에 대한 다양한 관점을 제공해 준다.

③ 데이터베이스 시스템의 자기 기술성은 데이터베이스 구조와 제약조건에 대한 정의를 가지고 있음을 의미한다.

④ 분산 데이터베이스 시스템에 포함된 각 지역의 DBMS는 지역 자치성(local autonomy)을 가질 수 있다.

정답 1번

출제 포인트

이 문제는 데이터 종속성과 데이터 독립성, 뷰, 자기 기술성, 분산 데이터베이스의 지역 자치성을 구별할 수 있는지를 묻는다.

핵심 개념

데이터 종속성은 데이터 구조나 접근 방식이 응용 프로그램에 직접 묶여 한쪽이 바뀌면 다른 쪽도 함께 수정되어야 하는 상태이다.
데이터 독립성은 데이터의 논리적 구조나 물리적 구조가 바뀌어도 응용 프로그램에 미치는 영향을 최소화하는 성질이다.
자기 기술성은 데이터베이스가 스키마와 제약조건 같은 메타데이터를 시스템 카탈로그에 함께 저장하는 성질이다.
지역 자치성은 분산 데이터베이스의 각 지역 DBMS가 자체적인 운영과 관리 권한을 가질 수 있는 성질이다.

선지별 해설

1번: X. 틀린 부분은 '데이터 종속성(data dependency)을 유지하여'라는 표현이다. DBMS는 데이터와 프로그램을 밀접하게 묶는 방향이 아니라 데이터 독립성을 제공하여 데이터 구조 변경이 프로그램에 미치는 영향을 줄인다.
2번: O. DBMS는 뷰를 통해 같은 공용 데이터에 대해서도 사용자나 응용 목적에 따라 서로 다른 외부 스키마를 제공할 수 있다.
3번: O. 데이터베이스 시스템의 자기 기술성은 데이터 자체뿐 아니라 데이터베이스 구조와 제약조건에 대한 정의도 카탈로그에 저장하고 있음을 뜻한다.
4번: O. 분산 데이터베이스 시스템에서는 각 지역의 DBMS가 지역 자원을 독자적으로 관리할 수 있는 지역 자치성을 가질 수 있다.

NoteBase

오답 포인트

데이터 종속성과 데이터 독립성을 반대로 외우면 1번을 맞는 설명처럼 보게 된다. DBMS의 핵심은 데이터와 프로그램의 결합을 강화하는 것이 아니라 결합을 약하게 만드는 데 있다.

다시 나오면 이렇게 판단

선지에 종속성, 독립성, 뷰, 카탈로그, 지역 자치성 같은 핵심 용어가 나오면 먼저 그 용어의 정의를 확인한 뒤, DBMS가 프로그램과 데이터를 더 강하게 묶는지 더 느슨하게 분리하는지를 기준으로 판단한다.

암기 포인트

DBMS는 데이터 종속성을 유지하는 시스템이 아니라 데이터 독립성을 높이는 시스템이다.

NoteBase

문 14. 다음 트랜잭션 스케줄 S_1, S_2, S_3, S_4 중 회복 불가능한 스케줄은? (단, $r_i(X)$와 $w_i(X)$는 각각 트랜잭션 T_i의 데이터 항목 X에 대한 읽기연산과 쓰기연산이고, c_i와 a_i는 각각 T_i의 완료연산과 철회연산이다)

① S_1: $r_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; a_1; $r_2(Y)$; c_2;

② S_2: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; c_2; a_1;

③ S_3: $r_1(X)$; $w_1(X)$; $r_2(Y)$; $w_2(Y)$; $w_2(X)$; c_2; c_1;

④ S_4: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; a_1; a_2;

정답 2번

출제 포인트

회복 가능 스케줄은 어떤 트랜잭션이 다른 트랜잭션의 미완료 갱신값을 읽었을 때, 그 값을 읽은 트랜잭션이 원작성자보다 먼저 완료되면 안 된다는 판정 기준을 묻는 문제이다.

핵심 개념

회복 가능 스케줄은 읽은 값의 원작성자 트랜잭션이 먼저 완료된 뒤에만 그 값을 읽은 트랜잭션이 완료되는 스케줄이다.
회복 불가능 스케줄은 미완료 갱신값을 읽은 트랜잭션이 원작성자보다 먼저 완료되어, 원작성자가 철회되면 읽은 결과를 되돌릴 수 없는 스케줄이다.
연쇄 복귀 가능 스케줄은 미완료 값을 읽을 수는 있지만 원작성자가 철회되면 읽은 쪽도 함께 철회하여 회복하는 스케줄이다.
각 스케줄은 읽기 의존관계(더티 리드)가 있는지 확인하고, 있다면 읽은 트랜잭션의 완료가 원작성자의 완료·철회보다 앞서는지를 따져 판정한다.

NoteBase

선지별 해설

① X: S1은 r1(X); r2(X); r1(Y); w2(X); w1(Y); a1; r2(Y); c2 순서이다. T1은 X를 쓰지 않고 Y만 w1(Y)로 쓰므로, T2의 r2(X)는 초기 X를 읽고 r2(Y)는 T1이 a1으로 철회된 뒤의 복원된 Y를 읽는다. 즉 T1의 미완료 갱신값을 읽는 더티 리드가 없다. 읽기 의존이 없어 T1의 철회가 T2에 영향을 주지 않으므로 회복 가능한 스케줄이며 회복 불가능이 아니다.

② O: S2는 r1(X); w1(X); r2(X); r1(Y); w2(X); c2; a1 순서이다. T2의 r2(X)는 T1의 w1(X)가 만든 미완료 값을 읽는 더티 리드이다. 그런데 T2가 c2로 먼저 완료된 뒤에야 T1이 a1으로 철회되므로, 더티 리드를 한 T2가 원작성자 T1보다 먼저 완료된다. 따라서 T1이 철회되어도 이미 완료된 T2의 결과를 되돌릴 수 없는 회복 불가능한 스케줄이다.

③ X: S3는 r1(X); w1(X); r2(Y); w2(Y); w2(X); c2; c1 순서이다. T2는 Y만 r2(Y)로 읽고 X는 읽지 않은 채 w2(X)로 덮어쓰기만 한다. 읽기 의존이 전혀 없으므로 c2가 c1보다 앞서더라도 되돌려야 할 더티 리드가 없다. 따라서 완료 순서와 무관하게 회복 가능한 스케줄이며 회복 불가능이 아니다.

④ X: S4는 r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); a1; a2 순서이다. T2의 r2(X)는 T1의 미완료 값을 읽는 더티 리드이지만, T1(a1)과 T2(a2)가 모두 철회되어 완료된 트랜잭션이 하나도 없다. 원작성자보다 먼저 완료된 트랜잭션이 없으므로 연쇄 복귀로 함께 되돌리면 되는 회복 가능한 스케줄이며 회복 불가능이 아니다.

오답 포인트

미완료 값을 읽었다는 사실만 보고 곧바로 회복 불가능으로 판단하면 ④에 끌리기 쉽다. 회복 불가능 여부는 더티 리드 자체가 아니라, 그 값을 읽은 트랜잭션이 원작성자보다 먼저 완료되었는지까지 확인해야 한다.

다시 나오면 이렇게 판단

먼저 어떤 읽기 연산이 다른 트랜잭션의 쓰기 결과를 읽었는지 찾고, 그 읽기 의존관계가 있으면 읽힌 값을 쓴 트랜잭션의 완료 또는 철회보다 읽은 쪽의 완료가 앞서는지를 본다. 읽은 쪽이 먼저 완료되면 회복 불가능이고, 함께 철회되거나 원작성자가 먼저 완료되면 회복 가능이다.

암기 포인트

회복 불가능은 더티 리드 자체가 아니라 더티 리드 후 선완료가 핵심이다.

NoteBase

문 15. 릴레이션 스키마 R(A, B, C, D)는 다음 함수적 종속성 집합 FD를 만족한다. R을 BCNF 정규화하였을 때 분해된 두 릴레이션 스키마 R1과 R2로 옳은 것은?

FD = {A→BCD, B→ACD, D→C}

① R1(B,C,D), R2(A,B)

② R1(A,C,D), R2(B,C,D)

③ R1(A,B,D), R2(C,D)

④ R1(A,B,C), R2(C,D)

정답 3번

출제 포인트

후보키를 찾고, BCNF를 위반하는 함수적 종속성을 골라 표준 분해 $R_1 = X \cup Y, R_2 = R - (Y - X)$를 적용하는 문제이다.

핵심 개념

BCNF는 모든 비자명한 함수적 종속성에서 결정자 X가 슈퍼키여야 하는 정규형이다.
이 문항에서는 $A^+ = ABCD, B^+ = ABCD$이므로 A와 B는 후보키이고, $D^+ = DC$이므로 D는 슈퍼키가 아니다.
따라서 $D \rightarrow C$만 BCNF 위반 종속성이고, 이를 기준으로 분해하면 $R_1(C, D), R_2(A, B, D)$가 된다.

NoteBase

선지별 해설

① X
R1(B,C,D)에서는 $D \to C$가 그대로 남아 있는데 D는 BCD의 슈퍼키가 아니므로 BCNF가 아니다.
틀린 표현은 분해 결과를 $R1(B, C, D), R2(A, B)$로 본 점이다. 올바르게는 $D \to C$로 분해하여 $R1(C, D), R2(A, B, D)$가 된다.

② X
R1(A,C,D)와 R2(B,C,D) 모두에서 $D \to C$가 남아 있고, D는 각 릴레이션의 슈퍼키가 아니므로 둘 다 BCNF가 아니다.
틀린 표현은 위반 종속성을 제거하지 못한 분해라는 점이다. 올바르게는 위반 FD인 $D \to C$를 기준으로 $R1(C, D), R2(A, B, D)$로 분해해야 한다.

③ O
$D \to C$가 BCNF 위반이므로 $X \cup Y = \{C, D\}, R - (Y - X) = \{A, B, D\}$로 분해하면 $R1(A, B, D), R2(C, D)$가 된다.
$R1(A, B, D)$에서는 A와 B가 여전히 키이고, $R2(C, D)$에서는 D가 키이므로 두 릴레이션 모두 BCNF를 만족한다.

④ X
R1(A,B,C)와 R2(C,D)는 각각만 보면 BCNF처럼 보일 수 있지만, 공통 속성 C가 어느 한쪽 릴레이션의 키가 아니어서 원래 릴레이션을 손실 없이 복원하는 분해가 아니다.
틀린 표현은 C만으로 조인해도 된다고 본 점이다. 올바르게는 위반 FD의 결정자 D를 공통 속성으로 남기는 $R1(A, B, D), R2(C, D)$ 분해이다.

오답 포인트

A와 B가 후보키라는 사실만 보고 BCNF 위반이 없다고 착각하거나, $D \to C$를 찾고도 표준 분해식을 적용하지 않아 공통 속성을 C로 잡는 실수가 자주 나온다. 이 문항의 함정은 '부분 릴레이션이 그럴듯해 보이는가'가 아니라 '위반 FD가 제거되었는가'와 '손실 없는 분해인가'를 함께 보는 점이다.

다시 나오면 이렇게 판단

먼저 각 결정자의 closure를 구해 후보키와 비후보키를 나눈다. 그다음 BCNF 위반 FD를 하나 찾으면 그 FD $X \to Y$에 대해 $X \cup Y$와 $R - (Y - X)$로 분해한다. 마지막으로 각 분해 릴레이션 안에서 남은 FD들의 결정자가 모두 슈퍼키인지 확인하면 된다.

NoteBase

암기 포인트

BCNF는 '모든 결정자가 슈퍼키'이다. 이 문항은 A, B는 키이고 D는 키가 아니므로 $D \to C$를 잡아 ABD, CD로 분해하는 문제이다.

NoteBase

문 16. 다음과 같은 두 트랜잭션 T_1과 T_2에 대하여 현재 T_1은 x-lock(X)를, T_2는 s-lock(Y)를 성공적으로 수행한 상태이다. 이후 두 트랜잭션이 각각 순서대로 마지막 연산까지 수행하는 과정에 대한 설명으로 옳은 것은? (단, read(X)와 write(X)는 각각 트랜잭션의 데이터 항목 X에 대한 읽기연산과 쓰기연산이고, x-lock(X)와 s-lock(X)는 각각 X에 대한 배타로크를 거는 연산과 공유로크를 거는 연산이며, unlock(X)는 X에 대한 로크를 해제하는 연산이다)

T_1	T_2
x-lock(X)	s-lock(Y)
read(X)	read(Y)
X = X + 500	x-lock(X)
write(X)	unlock(Y)
x-lock(Y)	read(X)
unlock(X)	X = X + Y
Y = Y + 500	unlock(X)
unlock(Y)	

① T_1은 성공적으로 완료되지만 T_2는 교착 상태에 빠진다.

② T_1은 교착 상태에 빠지지만 T_2는 성공적으로 완료된다.

③ T_1과 T_2가 모두 성공적으로 완료된다.

④ T_1과 T_2가 모두 교착 상태에 빠진다.

정답 4번

출제 포인트

이 문제는 락 호환성과 unlock 위치를 따라가며 두 트랜잭션의 대기 관계가 교착 상태로 닫히는지를 묻는 문제이다.

NoteBase

올바른 순서

1. T1은 이미 x-lock(X)을 가진 상태에서 read(X), X = X + 500, write(X)까지 수행한다.
2. T2는 이미 s-lock(Y)을 가진 상태에서 read(Y)까지 수행한다.
3. T1의 다음 연산은 x-lock(Y)인데, Y에는 아직 T2의 s-lock(Y)가 걸려 있으므로 T1은 대기한다.
4. T2의 다음 연산은 x-lock(X)인데, X에는 아직 T1의 x-lock(X)가 걸려 있으므로 T2도 대기한다.

왜 이 순서: T1은 x-lock(Y)를 얻기 전에는 unlock(X)로 갈 수 없고, T2는 x-lock(X)를 얻기 전에는 unlock(Y)로 갈 수 없어서 서로가 서로의 해제를 선행조건으로 만든다.

선지별 해설

① X

T1은 성공적으로 완료되지 않는다. T1의 unlock(X)는 x-lock(Y) 뒤에 배치되어 있는데, x-lock(Y) 단계에서 T2의 s-lock(Y)에 막혀 멈춘다. 틀린 표현은 "T1은 성공적으로 완료된다"이다. 올바르게는 T1도 완료되지 못하고 대기 상태가 된다.

② X

T2도 성공적으로 완료되지 않는다. T2의 unlock(Y)는 x-lock(X) 뒤에 배치되어 있는데, x-lock(X) 단계에서 T1의 x-lock(X)에 막혀 멈춘다. 틀린 표현은 "T2는 성공적으로 완료된다"이다. 올바르게는 T2도 완료되지 못하고 대기 상태가 된다.

③ X

두 트랜잭션이 모두 완료되려면 둘 중 하나가 먼저 필요한 락을 얻어 다음 unlock까지 진행해야 한다. 그러나 T1은 Y를, T2는 X를 기다리며 둘 다 다음 단계로 넘어가지 못한다. 틀린 표현은 "모두 성공적으로 완료된다"이다. 올바르게는 둘 다 교착 상태에 빠진다.

④ O

T1은 T2가 잡은 Y의 공유락이 풀리기를 기다리고, T2는 T1이 잡은 X의 배타락이 풀리기를 기다린다. 각 트랜잭션의 unlock이 추가 락 획득 뒤에 놓여 있으므로 서로 영원히 기다리는 교착 상태가 된다.

오답 포인트

공유락은 약한 락이라고 보고 T2가 곧 Y를 풀 것이라고 착각하기 쉽다. 그러나 이 문제의 핵심은 락의 종류보다 unlock의 위치이다. T2는 x-lock(X)를 먼저 얻어야 unlock(Y)로 갈 수 있고, T1도 x-lock(Y)를 먼저 얻어야 unlock(X)로 갈 수 있어서 둘 다 먼저 풀어 줄 수 없다.

NoteBase

다시 나오면 이렇게 판단

락 문제는 현재 보유 락, 다음에 요청할 락, unlock이 나오는 위치를 순서대로 적어 보면 된다. 한 트랜잭션이 가진 락을 풀기 전에 상대가 가진 다른 락을 추가로 얻어야 하고, 상대도 대칭적으로 같은 구조이면 wait-for graph가 원형이 되어 교착 상태가 된다.

암기 포인트

서로 가진 락을 풀기 전에 상대 락을 먼저 더 요구하면 교착 상태가 된다.

NoteBase

문 17. 빅데이터(big data)에 대한 설명으로 옳지 않은 것은?

① 디지털 환경에서 생성되는 데이터로 규모가 방대하고, 정형, 반정형, 비정형 등 다양한 형태의 데이터를 포함한다.

② NoSQL 시스템은 반구조적이고 자기 기술적인 데이터를 허용하므로 대개는 스키마를 요구하지 않는다.

③ NoSQL의 키-값(key-value) 데이터 모델은 키와 값의 쌍으로 저장하며, 값은 이미지나 동영상 등 다양한 형태의 데이터가 될 수 있다.

④ 빅데이터 분석 과정에서 추출된 정보를 시각화하는 기술로 Hadoop의 맵 리듀스(MapReduce)를 사용한다.

정답 4번

출제 포인트

이 문제는 빅데이터, NoSQL, 키-값 모델, MapReduce의 역할을 구분할 수 있는지를 묻는 문제이다.

핵심 개념

빅데이터는 규모가 크고 생성 속도가 빠르며 형태가 다양한 데이터와 그 처리 환경을 뜻한다.
NoSQL은 관계형 고정 스키마보다 유연한 데이터 모델을 사용하여 대량의 비정형·반정형 데이터를 분산 저장·처리하는 계열이다.
키-값 모델은 키 하나에 값 하나를 대응시키는 단순한 구조이며, 값은 문자열뿐 아니라 이미지·동영상 같은 바이너리 데이터도 될 수 있다.
MapReduce는 대용량 데이터를 여러 노드에서 나누어 처리하는 분산 처리 모델이지, 시각화 도구가 아니다.

NoteBase

선지별 해설

① O
빅데이터는 디지털 환경에서 대규모로 생성되며 정형, 반정형, 비정형 데이터를 모두 포함할 수 있다. 선지의 설명은 빅데이터의 대표적 특징인 규모와 다양성을 올바르게 설명한다.

② O
NoSQL은 관계형 데이터베이스처럼 엄격한 고정 스키마를 전제로 하지 않는 경우가 많다. 반구조적이고 자기 기술적인 데이터를 유연하게 저장할 수 있으므로 선지의 취지는 타당하다.

③ O
키-값 데이터 모델은 키와 값의 쌍으로 저장하는 구조이다. 이때 값은 단순 문자열에 한정되지 않고 문서, 이미지, 동영상 등 다양한 형태의 데이터가 될 수 있다.

④ X
틀린 부분은 "시각화하는 기술로 Hadoop의 맵 리듀스(MapReduce)를 사용한다"이다. MapReduce는 빅데이터를 분산 병렬 처리하는 프로그래밍 모델이며, 시각화 기술이 아니다. 올바르게는 "빅데이터의 분산 처리 기술로 Hadoop의 MapReduce를 사용한다" 또는 "분석 결과의 시각화에는 별도의 시각화 도구를 사용한다"이다.

오답 포인트

④는 "분석"과 "시각화"를 한 덩어리로 묶어 MapReduce의 역할을 바꿔 놓은 선지이다. Hadoop, HDFS, MapReduce는 저장·분산 처리 쪽 개념이고, 시각화는 결과 표현 단계의 개념이라는 점을 분리하지 않으면 틀리기 쉽다.

다시 나오면 이렇게 판단

빅데이터 관련 선지는 먼저 데이터의 성격을 설명하는지, 저장 모델을 설명하는지, 분산 처리 방식을 설명하는지, 시각화 단계를 설명하는지 역할부터 나누어 본다. 처리 엔진을 시각화 도구로 바꾸어 적어 놓으면 오답으로 판단된다.

암기 포인트

빅데이터는 다양한 대용량 데이터, NoSQL은 유연한 저장 모델, MapReduce는 분산 처리이다.

NoteBase

문 18. XML(eXtensible Markup Language)에 대한 설명으로 옳은 것은?

① DTD(Document Type Definition) 문서는 XML 문서를 작성하기 위한 구문 규칙과 동일한 방식으로 작성된다.

② DTD(Document Type Definition) 문서는 XML 스키마 문서와 달리 네임스페이스(namespace)를 지원한다.

③ XQuery 질의문은 FOR, LET, WHERE, RETURN 절로 구성되고 둘 이상의 변수에 대한 조인연산을 작성할 수 있다.

④ XML 스키마는 XML 문서의 요소 및 속성을 데이터베이스 테이블에 맵핑(mapping)할 때 문자열 데이터를 제외한 다른 데이터형(data type)을 지정할 수 없다.

정답 3번

출제 포인트

이 문제는 DTD, XML 스키마, XQuery의 기능과 표현 방식 차이를 구분하는 문제이다.

핵심 개념

DTD는 XML 문서의 요소 구조와 허용 관계를 정의하는 문서이지만 문법 자체는 XML과 다른 비XML 형식이다. 키워드는 구조 정의, 비XML 문법이다.
XML 스키마는 XML 문서의 구조와 데이터형을 XML 문법으로 정의하는 규격이다. 키워드는 XML 문법, 네임스페이스 지원, 다양한 데이터형이다.
XQuery는 XML 데이터를 검색하고 가공하는 질의 언어이며 FLWOR 표현식으로 여러 변수에 대한 조건과 결과 생성을 기술할 수 있다. 키워드는 질의, FLWOR, 조인이다.

선지별 해설

① X. 틀린 부분은 "XML 문서를 작성하기 위한 구문 규칙과 동일한 방식으로 작성된다"이다. DTD는 XML과 같은 태그 문법으로 쓰는 것이 아니라 별도의 선언 문법으로 구조를 정의한다. 올바르게는 "DTD는 XML 문서의 구조를 정의하지만 문서 자체와는 다른 문법으로 작성된다"이다.
② X. 틀린 부분은 "XML 스키마 문서와 달리 네임스페이스를 지원한다"이다. 네임스페이스 지원은 XML 스키마의 장점이고 DTD는 네임스페이스 표현이 제한적이다. 올바르게는 "XML 스키마는 네임스페이스를 지원하지만 DTD는 이를 제대로 지원하지 못한다"이다.
③ O. XQuery의 대표 표현식은 FOR, LET, WHERE, RETURN을 중심으로 작성되며, 여러 변수 바인딩을 이용해 둘 이상의 XML 집합 사이 조건 비교와 조인 연산을 기술할 수 있다.
④ X. 틀린 부분은 "문자열 데이터를 제외한 다른 데이터형을 지정할 수 없다"이다. XML 스키마는 문자열뿐 아니라 정수, 날짜, 불리언 등 다양한 단순 데이터형과 복합형을 지정할 수 있다. 올바르게는 "XML 스키마는 요소와 속성에 다양한 데이터형을 지정할 수 있다"이다.

NoteBase

오답 포인트

DTD와 XML 스키마를 둘 다 구조 정의 도구로만 외우면 문법 형식과 데이터형 지원 여부를 뒤바꾸기 쉽다. 또한 XQuery의 FLWOR 표현식을 단순 반복문처럼만 보면 여러 변수 기반 조인 가능성을 놓치기 쉽다.

다시 나오면 이렇게 판단

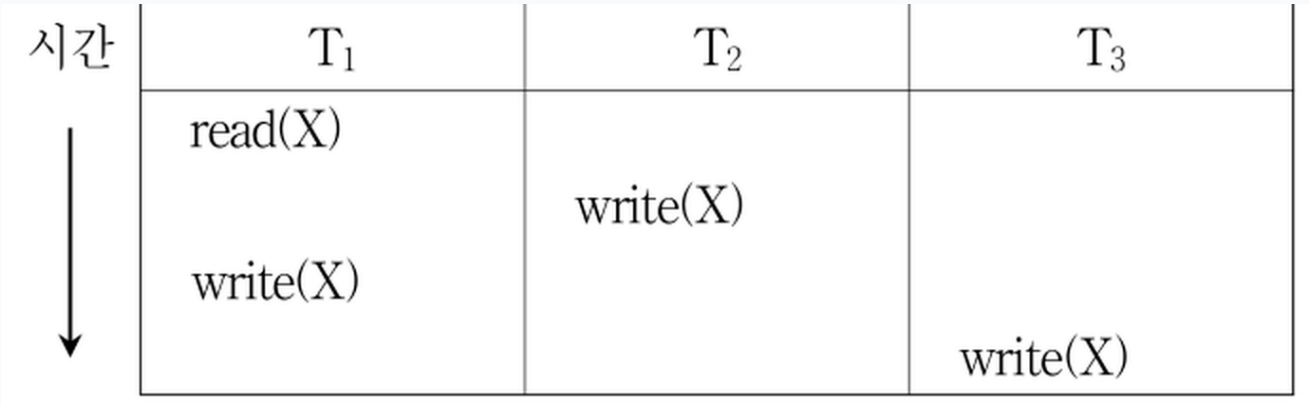
XML 관련 선지는 먼저 구조 정의 도구인지 질의 언어인지 나눈다. 구조 정의 도구라면 DTD와 XML 스키마 중 어느 쪽이 XML 문법, 네임스페이스, 데이터형 지원을 갖는지 확인하고, XQuery가 나오면 여러 변수 바인딩과 조건 결합이 가능한 질의 언어인지 판단한다.

암기 포인트

DTD는 비XML 문법, XML 스키마는 데이터형과 네임스페이스, XQuery는 FLWOR와 조인이다.

NoteBase

문 19. 3개의 트랜잭션 T_1, T_2, T_3에 대하여, 다음과 같은 스케줄의 뷰(view)/충돌(conflict) 직렬 가능성(serializability)을 설명한 것으로 옳은 것은?



① 뷰 직렬 가능하며, 동시에 충돌 직렬 가능하다.

② 뷰 직렬 가능하지 않으며, 충돌 직렬 가능하지 않다.

③ 뷰 직렬 가능하지만, 충돌 직렬 가능하지 않다.

④ 뷰 직렬 가능하지 않지만, 충돌 직렬 가능하다.

정답 3번

출제 포인트

이 문제는 한 스케줄이 뷰 직렬 가능하지만 충돌 직렬 가능하지는 않을 수 있다는 점을 묻는 문제이다.

핵심 개념

뷰 직렬 가능성은 각 read가 어떤 write의 값을 읽는지와 최종 write 트랜잭션이 같은 직렬 스케줄이 존재하면 성립한다.
충돌 직렬 가능성은 충돌 연산의 선후관계로 만든 precedence graph에 cycle이 없을 때만 성립한다.
이 스케줄에서는 T1의 read(X)는 초기값을 읽고, 최종 write는 T3가 수행한다. 따라서 T1-T2-T3의 직렬 스케줄과 읽기 관계와 최종 write를 같게 만들 수 있어 뷰 직렬 가능하다.
반면 r1(X)와 w2(X) 때문에 T1→T2, w2(X)와 w1(X) 때문에 T2→T1이 생겨 cycle이 만들어지므로 충돌 직렬 가능하지 않다.

NoteBase

선지별 해설

① X
뷰 직렬 가능성은 성립하지만 충돌 직렬 가능성은 성립하지 않는다.
틀린 부분은 "동시에 충돌 직렬 가능하다"이다.
올바르게는 "뷰 직렬 가능하지만, 충돌 직렬 가능하지 않다"이다.

② X
이 스케줄은 T1이 초기값을 읽고 T3가 최종 write를 하므로 T1-T2-T3 직렬 스케줄과 같은 view를 만들 수 있다.
틀린 부분은 "뷰 직렬 가능하지 않다"이다.
올바르게는 "뷰 직렬 가능하다"이다.

③ O
T1의 read(X)는 어떤 write의 값도 읽지 않고 초기값을 읽는다. 또 마지막 write(X)는 T3가 수행한다. 이 관계는 T1-T2-T3 직렬 스케줄과 같다.
하지만 충돌 그래프에서는 T1→T2와 T2→T1이 동시에 생겨 cycle이 있으므로 충돌 직렬 가능하지 않다.

④ X
충돌 직렬 가능성은 precedence graph에 cycle이 없을 때만 성립하는데, 이 스케줄은 T1과 T2 사이에 상호 간선이 생긴다.
틀린 부분은 "충돌 직렬 가능하다"이다.
올바르게는 "충돌 직렬 가능하지 않다"이다.

오답 포인트

뷰 직렬 가능성과 충돌 직렬 가능성을 같은 뜻으로 보면 ①이나 ④로 흔들리기 쉽다. 이 문항은 blind write가 포함된 스케줄에서는 뷰 직렬 가능성이 충돌 직렬 가능성보다 넓다는 점을 노린 문제이다.

다시 나오면 이렇게 판단

먼저 precedence graph에서 cycle 존재 여부를 확인해 충돌 직렬 가능성을 판단한다. 그다음 초기값을 읽는 read가 있는지, 각 read가 누구의 값을 읽는지, 최종 write가 누구인지 확인해 같은 view를 만드는 직렬 순서가 있는지 따지면 된다.

암기 포인트

충돌 직렬 가능이면 항상 뷰 직렬 가능이지만, 뷰 직렬 가능이라고 해서 항상 충돌 직렬 가능한 것은 아니다.

NoteBase

문 20. 관계형 데이터베이스 역정규화(denormalization)에 대한 설명으로 옳지 않은 것만을 모두 고르면?

ㄱ. 릴레이션들은 역정규화한 후 정규형 수준이 높아진다.
ㄴ. 데이터베이스의 데이터 무결성을 강화할 목적으로 개발되었다.
ㄷ. 릴레이션들의 데이터 중복을 줄임으로써 데이터베이스의 크기를 감소시킨다.
ㄹ. 둘 이상의 릴레이션들에 대하여 조인한 결과를 빈번하게 이용하는 경우, 역정규화 함으로써 질의응답 시간이 단축될 수 있다.

① ㄱ, ㄴ, ㄷ
② ㄱ, ㄴ, ㄹ
③ ㄴ, ㄷ, ㄹ
④ ㄱ, ㄴ, ㄷ, ㄹ

정답 1번

출제 포인트

이 문제는 역정규화의 목적이 무결성 강화가 아니라 성능 개선이며, 그 과정에서 정규형 수준과 데이터 중복이 어떻게 바뀌는지를 구별하는 문제이다.

핵심 개념

정규화는 데이터 중복과 갱신 이상을 줄여 무결성을 높이기 위한 설계이다.
역정규화는 조회 성능을 높이기 위해 중복을 일부 허용하거나 릴레이션을 통합하여 조인을 줄이는 설계이므로, 일반적으로 정규형 수준은 낮아지고 저장 공간은 늘 수 있다.

선지별 해설

① O: 옳지 않은 설명은 ㄱ, ㄴ, ㄷ이고 ㄹ만 옳으므로 정답이다. ㄱ의 '정규형 수준이 높아진다'는 틀린 표현이며, 올바르게는 역정규화 후 정규형 수준은 낮아지거나 중복 허용이 커진다. ㄴ의 '데이터 무결성을 강화할 목적'도 틀렸고, 올바르게는 조회 성능 개선과 조인 비용 감소가 목적이다. ㄷ의 '데이터 중복을 줄인다'와 '크기를 감소시킨다'도 틀렸고, 올바르게는 중복이 늘어 데이터베이스 크기가 커질 수 있다. ㄹ은 조인 결과를 자주 사용할 때 역정규화로 질의응답 시간을 줄일 수 있다는 설명이므로 옳다.
② X: ㄹ을 옳지 않은 것에 포함한 점이 틀리다. ㄹ은 역정규화의 대표적 목적을 설명한 옳은 진술이다. 올바르게는 ㄱ, ㄴ, ㄷ이다.
③ X: ㄱ을 빠뜨린 점이 틀리다. ㄱ도 역정규화의 효과를 반대로 서술한 오답이므로 포함되어야 한다. 올바르게는 ㄱ, ㄴ, ㄷ이다.
④ X: ㄹ까지 포함한 점이 틀리다. ㄹ은 옳은 설명이므로 제외되어야 한다. 올바르게는 ㄱ, ㄴ, ㄷ이다.

NoteBase

오답 포인트

정규화와 역정규화의 방향을 뒤집어 외우면 틀리기 쉽다. '중복 감소, 무결성 강화, 저장 공간 감소'는 정규화의 특징이고, 역정규화는 그 반대로 성능을 위해 중복을 감수하는 설계라는 점이 함정이다.

다시 나오면 이렇게 판단

선지에 중복 감소, 무결성 강화, 정규형 상승이 나오면 정규화 쪽 설명인지 먼저 본다. 반대로 조인 감소, 조회 성능 향상, 중복 허용이 나오면 역정규화 설명으로 판단할 수 있다.

암기 포인트

정규화는 무결성·중복 감소, 역정규화는 성능·조인 감소이다.

NoteBase