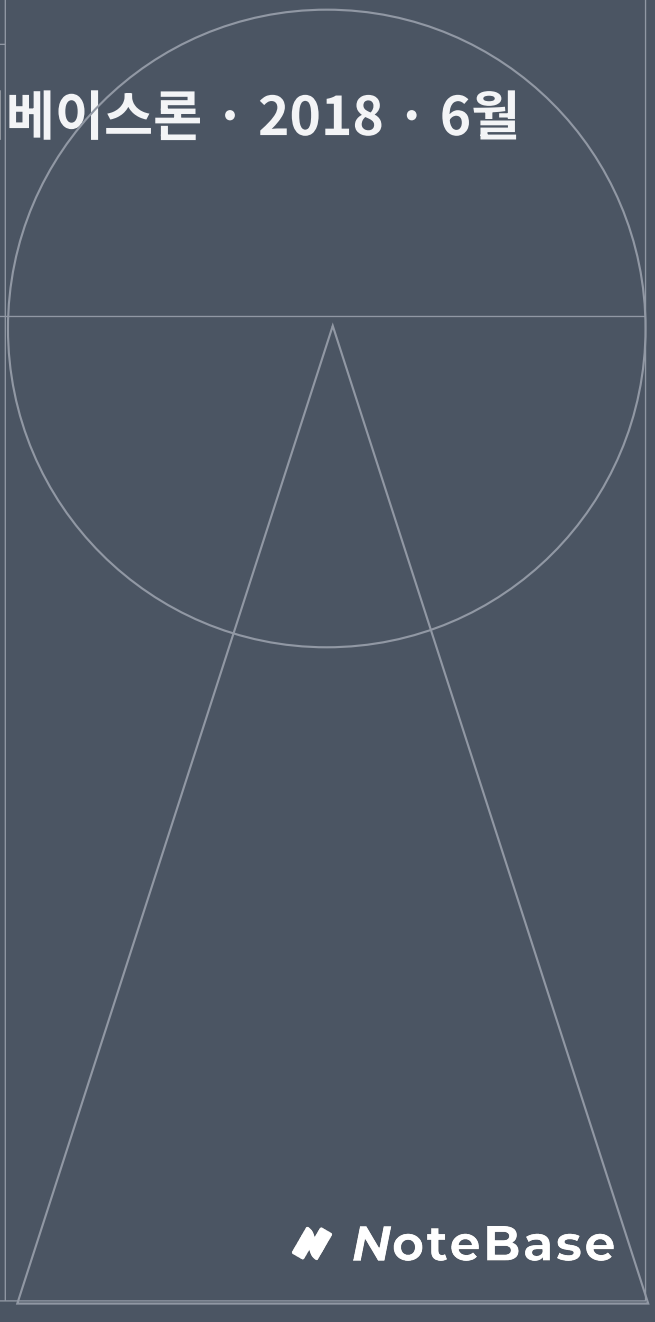
기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2018 · 6월



NoteBase

기출OX 해설집

공무원/서울시/7급 · 데이터베이스론 · 2018 · 6월

공무원/서울시/7급 · 데이터베이스론 · 2018 · 6월

문 1. <보기 1>은 의사들에 대한 정보를 담고 있는 doctor 테이블이다. 이 테이블에서 <보기 2>의 SQL을 수행할 때 출력되는 결과로 가장 옳은 것은?

<보기 1>

name	gender	major
박태웅	남	소화기
유성우	남	심장
김철수	남	대장
최태웅	남	간
이수정	여	위
조수진	여	심장
임석영	남	소화기
최인탁	남	간

<보기 2>

```
SELECT major, count(*) AS cnt
FROM doctor
GROUP BY major
HAVING major='소화기' OR major='심장';
```

① major: 소화기, cnt: 2; major: 심장, cnt: 2

② major: 소화기, count(*): 2; major: 심장, count(*): 2

③ major: 소화기; major: 심장

④ cnt: 2; cnt: 2

정답 1번

출제 포인트

이 문제는 GROUP BY로 전공별 그룹을 만든 뒤 HAVING으로 소화기와 심장 그룹만 남기고, SELECT 절의 별칭이 결과 열 이름으로 출력되는지를 묻는 문제이다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2018 · 6월

자료 읽는 순서

1. doctor 테이블에서 major 값이 무엇인지 행별로 확인한다.
2. GROUP BY major로 같은 major끼리 묶어 개수를 센다.
3. HAVING major='소화기' OR major='심장'으로 두 그룹만 남긴다.
4. SELECT major, count(*) AS cnt 이므로 결과 열 이름이 major와 cnt로 출력된다고 판단한다.

핵심 근거

표에서 소화기는 박태웅, 임석영으로 2명이고 심장은 유성우, 조수진으로 2명이다.
따라서 최종 결과는 소화기 2, 심장 2의 두 행이다.
또한 count(*)에 AS cnt를 주었으므로 결과 헤더는 count(*)가 아니라 cnt이다.

선지별 해설

① O
- 근거 위치: doctor 테이블의 major 열
- 소화기 2명, 심장 2명만 남고 결과 열 이름도 major, cnt로 표시되므로 맞다.

② X
- 근거 위치: SELECT 절의 count(*) AS cnt
- 집계값 2, 2 자체는 맞지만 결과 열 이름을 count(*)로 적은 점이 틀리다.
- 틀린 표현은 count(*)이고, 올바르게는 cnt이다.

③ X
- 근거 위치: SELECT 절의 major, count(*) AS cnt
- GROUP BY와 HAVING 이후에도 SELECT 절에 count(*) AS cnt가 있으므로 집계 열이 함께 출력되어야 한다.
- 틀린 표현은 cnt 열이 없는 결과이고, 올바르게는 major와 cnt 두 열이 함께 있어야 한다.

④ X
- 근거 위치: SELECT 절의 major, count(*) AS cnt
- 이 질의는 major와 cnt를 함께 조회하므로 cnt만 단독으로 출력되지 않는다.
- 틀린 표현은 major 열이 없는 결과이고, 올바르게는 소화기 2, 심장 2가 major와 함께 출력된다.

오답 포인트

집계값만 보고 ②를 고르기 쉽지만, 이 문항은 값뿐 아니라 결과 열 이름까지 함께 묻는다. AS cnt를 놓치면 count(*)가 그대로 출력된다고 착각하기 쉽다.

NoteBase

다시 나오면 이렇게 판단

SQL 결과 문제는 FROM으로 대상 표를 확인하고, GROUP BY로 묶인 기준을 찾고, HAVING으로 남는 그룹을 걸러낸 뒤, 마지막에 SELECT 절의 열 순서와 별칭이 결과 표 헤더에 어떻게 나타나는지를 확인하면 된다.

암기 포인트

GROUP BY는 그룹을 만들고 HAVING은 그룹을 거르며, SELECT의 AS 별칭은 결과 열 이름으로 출력된다.

NoteBase

문 2. <보기>의 스키마에서 자기 자신이 속한 학과 학생들의 평균 나이(AGE) 보다 나이가 많은 학생들의 이름(SNAME)을 보여 주는 SQL을 작성한 것으로 가장 옳은 것은? (단, 밑줄 친 속성은 기본키이며, STUDENT의 DID는 DEPARTMENT의 DID를 참조하는 외래키이다.)

<보기>

DEPARTMENT(DID, DNAME)

STUDENT(SID, SNAME, AGE, DID)

①
```
SELECT S.SNAME
FROM STUDENT S INNER JOIN
        (SELECT DID, DNAME
        FROM DEPARTMENT
        GROUP BY DID) D ON S.DID = D.DID
WHERE S.AGE > AVG(S.AGE);
```

②
```
SELECT S.SNAME
FROM STUDENT S
WHERE S.AGE >
        (SELECT AVG(AGE)
        FROM STUDENT
        WHERE DID = S.DID);
```

③
```
SELECT S.SNAME
FROM STUDENT S
GROUP BY S.DID
HAVING S.AGE > AVG(S.AGE);
```

④
```
SELECT *
FROM DEPARTMENT D, STUDENT S
WHERE D.DID = S.DID AND S.AGE > AVG(S.AGE)
GROUP BY D.DID;
```

정답 2번

출제 포인트

이 문제는 학생별 행을 유지한 채 자신이 속한 학과의 평균 나이와 비교하는 상관 서브쿼리를 적용할 수 있는지를 묻는 문제이다.

사례 정리

STUDENT의 DID가 학과 소속을 나타내므로 학생이 속한 학과는 STUDENT.DID로 식별된다.
구해야 하는 결과는 학과별 평균값 자체가 아니라, 같은 학과 평균보다 나이가 많은 학생의 이름 목록이다.
따라서 각 학생 행마다 같은 DID를 가진 학생 집합의 AVG(AGE)를 다시 계산하여 비교해야 한다.

핵심 개념

상관 서브쿼리는 내부 질의가 바깥 질의의 값을 참조하여 각 행마다 다시 수행되는 서브쿼리이다.
이 문제의 핵심은 바깥 학생 S의 DID를 내부 평균 계산에 연결하여 같은 학과 학생들만 대상으로 평균 나이를 구하는 것이다.

NoteBase

적용 기준

학과 평균을 구한다는 이유만으로 GROUP BY 결과만 만들면 학생 이름 목록을 바로 얻을 수 없다.
학생 이름을 출력하려면 학생 행을 유지해야 하므로, 바깥 학생의 DID와 내부 집계 대상의 DID를 연결하는 WHERE DID = S.DID 형태가 필요하다.

선지별 해설

① X: WHERE S.AGE > AVG(S.AGE)는 집계함수를 WHERE 절에서 직접 사용한 형태라 올바르지 않다. 또한 서브쿼리가 DEPARTMENT만 조회하므로 학과 학생들의 평균 나이를 계산하지 못한다. 틀린 표현은 AVG(S.AGE) 비교와 DEPARTMENT 서브쿼리이며, 올바르게는 STUDENT에서 같은 DID의 AVG(AGE)를 구하는 상관 서브쿼리로 써야 한다.
② O: 바깥 학생 S에 대해 내부 질의가 WHERE DID = S.DID로 같은 학과 학생만 모아 AVG(AGE)를 계산하므로, 각 학생의 나이를 자기 학과 평균과 정확히 비교할 수 있다. 요구 조건인 학생 이름 출력도 SELECT S.SNAME으로 맞게 처리한다.
③ X: GROUP BY S.DID는 학과별 묶음만 만들 뿐 학생 이름 S.SNAME과 개별 나이 S.AGE를 행 단위로 비교하는 질의가 아니다. 틀린 표현은 GROUP BY S.DID HAVING S.AGE > AVG(S.AGE)이며, 올바르게는 그룹 결과가 아니라 각 학생 행을 상관 서브쿼리의 평균과 비교해야 한다.
④ X: 조인 조건 D.DID = S.SID가 잘못되었고, 학생의 학과 연결은 S.DID = D.DID이어야 한다. 또한 S.AGE > AVG(S.AGE)를 WHERE에서 직접 쓸 수 없고 SELECT *와 GROUP BY D.DID의 조합도 요구 결과와 맞지 않는다. 올바르게는 STUDENT에서 같은 DID 평균을 구하는 상관 서브쿼리로 학생 이름만 추출해야 한다.

오답 포인트

학과별 평균이라는 표현만 보고 GROUP BY를 먼저 떠올리면 ③처럼 집단 요약 질의를 만들기 쉽다. 그러나 출력 대상이 학과가 아니라 학생 이름이므로 학생 행을 유지한 채 같은 학과 평균과 비교해야 한다는 점이 함정이다.

다시 나오면 이렇게 판단

출력 결과가 개별 행이고 비교 기준이 그 행이 속한 집단의 평균이면 먼저 행 유지가 필요한지 확인한다. 행 유지가 필요하면 바깥 행의 키와 내부 집계 대상의 키를 연결하는 상관 서브쿼리를 우선 판단 기준으로 잡는다.

암기 포인트

개별 행과 소속 집단 평균을 함께 묻는 SQL은 GROUP BY 결과표보다 상관 서브쿼리가 먼저이다.

NoteBase

문 3. 뷰(view)에 대한 설명으로 가장 옳은 것은?

① 사용자들의 데이터 접근을 제어하여 보안성을 높일 수 있다.

② 단일 기본 테이블에 집단 연산을 사용하지 않고 셀렉션과 프로젝션만을 사용하여 정의된 뷰에 대해서만 갱신할 수 있게 하는 것을 삽입 가능한 뷰라고 한다.

③ 뷰 메커니즘은 관계 모델에서 논리적 데이터의 종속성을 가진다.

④ 물리적으로 저장되어 있는 기본 테이블과 다르게 가상 테이블이므로, 갱신 연산에 제한을 받지 않는다.

정답 1번

출제 포인트

이 문제는 뷰의 기능, 갱신 가능 뷰의 조건, 그리고 뷰가 제공하는 논리적 데이터 독립성을 구분하는 문제이다.

핵심 개념

뷰는 기본 테이블에서 필요한 데이터만 논리적으로 보여 주는 가상 테이블이다.
기본 테이블은 실제로 데이터가 저장되는 테이블이고, 뷰는 저장된 질의 결과를 통해 외부에 보이는 형태를 단순화한다.
갱신 가능한 뷰는 보통 단일 기본 테이블에 대해 정의되고, 집단 연산이나 조인 없이 셀렉션과 프로젝션 중심으로 구성된 경우가 대표적이다.
뷰 메커니즘은 사용자에게 필요한 논리적 모습만 보여 주므로 논리적 데이터 독립성을 높이는 데 쓰인다.

NoteBase

선지별 해설

① O
사용자마다 필요한 열과 행만 보이게 하여 접근 범위를 제한할 수 있으므로 보안성을 높일 수 있다. 뷰의 대표적 목적 중 하나가 접근 제어이다.

② X
틀린 부분은 "삽입 가능한 뷰"라는 표현이다.
단일 기본 테이블에 대해 집단 연산 없이 셀렉션과 프로젝션만으로 정의되어 갱신할 수 있는 뷰는 일반적으로 갱신 가능한 뷰라고 본다. 삽입 가능 여부는 별도의 제약을 더 따진다.

③ X
틀린 부분은 "논리적 데이터의 종속성"이다.
뷰는 사용자 관점과 논리 구조를 분리하여 논리적 데이터 독립성을 높이는 메커니즘이다. 뷰의 목적은 종속성을 키우는 것이 아니다.

④ X
틀린 부분은 "갱신 연산에 제한을 받지 않는다"이다.
뷰는 가상 테이블이므로 오히려 갱신에 제약이 있다. 조인, 집단 연산, 파생 속성 등이 포함되면 갱신이 불가능하거나 제한되며, 조건을 만족하는 일부 뷰만 갱신 가능하다.

오답 포인트

②는 갱신 가능한 뷰와 삽입 가능한 뷰를 섞어 놓은 선지이고, ③은 독립성과 종속성을 뒤바꾼 선지이다. ④는 "가상 테이블"이라는 말 때문에 오히려 자유롭게 갱신된다고 착각하기 쉽지만 실제 시험에서는 반대로 갱신 제약을 연결하는 경우가 많다.

다시 나오면 이렇게 판단

뷰 문제가 나오면 먼저 보안성 향상, 논리적 데이터 독립성, 갱신 제약이라는 세 축으로 본다. 이어서 "단일 테이블인가", "집단 연산이나 조인이 있는가"를 확인하면 갱신 가능 여부를 빠르게 판단할 수 있다.

암기 포인트

뷰는 보안과 독립성에는 유리하고, 갱신에는 제약이 있는 가상 테이블이다.

NoteBase

문 4. DBMS에 대한 설명으로 가장 옳지 않은 것은?

① 삽입(insert), 삭제(delete), 갱신(update)과 같은 연산을 사용하여 데이터를 최신 값으로 유지하는 데 용이하다.

② 질의에 대하여 실시간으로 반영이 가능하다.

③ 같은 내용의 데이터를 다수의 사용자가 동시에 접근하여 이용하는 것이 가능하다.

④ 데이터베이스 내에 있는 데이터들은 사용자가 직접 데이터 주소를 참조함으로써 빠르게 찾을 수 있다.

정답 4번

출제 포인트

이 문제는 DBMS의 기능과 장점을 이해하고, 사용자의 데이터 접근 방식이 직접 주소 참조인지 질의 기반인지 구분하는 문제이다.

핵심 개념

DBMS는 사용자와 실제 저장 데이터 사이를 중재하면서 삽입, 삭제, 갱신, 검색을 관리하는 소프트웨어이다. 결정 차이축은 물리적 데이터 주소를 사용자가 직접 다루는지 여부이다. 키워드는 데이터 독립성, 질의 처리, 동시 공유이다.
직접 주소 참조 방식은 사용자가 물리적 저장 위치나 파일 구조를 알아야 데이터를 찾는 방식이다. DBMS는 이런 물리적 세부사항을 숨기고 논리적 질의로 접근하게 한다.

선지별 해설

① O: DBMS는 insert, delete, update 같은 연산으로 데이터를 쉽게 변경하고 최신 상태로 유지할 수 있다. 데이터 조작 기능을 설명한 선지이므로 옳다.
② O: DBMS에서는 데이터가 갱신되면 질의 결과에 그 내용이 바로 반영되어 현재 상태를 조회할 수 있다. 질의 처리와 갱신 반영 기능을 설명한 취지로 옳다.
③ O: DBMS는 여러 사용자가 같은 데이터를 동시에 공유하고 접근할 수 있도록 동시성 제어와 공유 기능을 제공한다. 다수 사용자 환경에 적합하다는 설명이므로 옳다.
④ X: 틀린 부분은 "사용자가 직접 데이터 주소를 참조"한다는 표현이다. DBMS에서는 사용자가 물리적 데이터 주소를 직접 참조하지 않고, 질의어와 스키마를 통해 논리적으로 데이터에 접근한다. 올바르게는 "데이터베이스 내의 데이터는 DBMS가 제공하는 질의 기능과 접근 경로를 통해 빠르게 찾을 수 있다"이다.

오답 포인트

④는 "빠르게 찾을 수 있다"는 부분만 보면 인덱스나 검색 속도 때문에 맞는 말처럼 보이지만, 함정은 그 앞의 "사용자가 직접 데이터 주소를 참조"라는 표현이다. DBMS의 핵심은 물리적 저장 위치를 사용자에게 숨기고 논리적 접근을 제공한다는 점이다.

NoteBase

다시 나오면 이렇게 판단

DBMS 관련 선지는 먼저 데이터 조작, 질의 처리, 동시 공유, 데이터 독립성 같은 핵심 기능인지 확인한다. 물리적 주소를 사용자가 직접 알아야 한다는 표현이 나오면 파일 처리 방식에 가까우므로 틀린 선지로 판단할 수 있다.

암기 포인트

DBMS는 논리적 질의로 접근하고, 직접 주소 참조는 파일 처리 방식의 특징이다.

NoteBase

문 5. 트랜잭션의 ACID에 대한 설명으로 가장 옳은 것은?

① 트랜잭션이 성공적으로 완료되면 해당 트랜잭션에 의해 변경된 데이터는 이후 다른 트랜잭션에 의해서 변경될 수 없다.

② 트랜잭션 다수가 동시에 수행될 때 각 트랜잭션의 결과가 서로에게 영향을 주는 경우가 발생할 수 있다.

③ 트랜잭션의 연산은 모두 데이터베이스에 반영되거나 취소되어야 하지만, 시스템 붕괴가 발생할 시 이를 지키지 않을 수 있다.

④ 트랜잭션이 성공적으로 수행되면 해당 데이터는 무결성이 유지되어야 하며 일관성 있는 상태이어야 한다.

정답 4번

출제 포인트

이 문제는 트랜잭션의 ACID 중 원자성, 일관성, 고립성, 지속성의 의미를 정확히 구별하는지를 묻는 문제이다.

핵심 개념

원자성은 트랜잭션의 연산이 전부 수행되거나 전부 취소되는 성질이다.
일관성은 트랜잭션 수행 전후에 데이터베이스가 무결성을 만족하는 일관된 상태를 유지하는 성질이다.
고립성은 동시에 실행되는 트랜잭션들이 서로 간섭하지 않고 결과적으로 직렬 실행과 같은 효과를 내는 성질이다.
지속성은 커밋이 완료된 결과가 시스템 장애가 발생해도 보존되는 성질이다.

선지별 해설

① X: 틀린 부분은 "이후 다른 트랜잭션에 의해서 변경될 수 없다"이다. 지속성은 커밋된 결과가 장애 후에도 보존된다는 뜻이지, 이후의 정상적인 다른 트랜잭션이 그 데이터를 다시 변경할 수 없다는 뜻은 아니다. 올바르게는 "커밋된 변경 내용은 장애가 발생해도 보존되지만, 이후 다른 트랜잭션에 의해 다시 변경될 수 있다"이다.
② X: 틀린 부분은 "각 트랜잭션의 결과가 서로에게 영향을 주는 경우가 발생할 수 있다"이다. 고립성은 동시 수행 중에도 각 트랜잭션이 서로 간섭하지 않도록 보장하는 성질이다. 올바르게는 "동시에 수행되더라도 결과는 서로 간섭하지 않아야 하며 직렬 수행과 같은 결과가 나와야 한다"이다.
③ X: 틀린 부분은 "시스템 붕괴가 발생할 시 이를 지키지 않을 수 있다"이다. 원자성과 지속성은 오히려 시스템 장애 상황에서도 회복 기법을 통해 커밋 또는 롤백 결과가 보장되도록 하는 성질이다. 올바르게는 "연산은 모두 반영되거나 모두 취소되어야 하며, 커밋된 결과는 시스템 붕괴 후에도 유지되어야 한다"이다.
④ O: 트랜잭션이 성공적으로 수행되면 데이터베이스는 무결성이 유지되는 일관된 상태여야 한다는 설명으로, 일관성의 의미를 가장 정확히 설명한 선지이다.

NoteBase

오답 포인트

①은 지속성을 "변경 불가"로 오해하게 만들고, ②는 고립성을 "동시 수행 중 상호 영향 허용"으로 바꿔 놓은 함정이다. ③은 장애 상황에서 ACID가 깨질 수 있다고 서술해 원자성과 지속성의 목적 자체를 뒤집어 놓은 문장이다.

다시 나오면 이렇게 판단

커밋 후 보존이면 지속성, 수행 전후 무결성 유지이면 일관성, 동시에 실행해도 서로 간섭하지 않으면 고립성, 전부 반영 또는 전부 취소이면 원자성으로 판단하면 된다.

암기 포인트

ACID는 원자성 전부-or-전무, 일관성 무결성 유지, 고립성 상호 간섭 배제, 지속성 커밋 결과 보존으로 정리된다.

NoteBase

문 6. 동시성 제어 기법에 대한 설명으로 가장 옳지 않은 것은?

① 2단계 로킹(locking)규약은 다른 트랜잭션이 로크(lock)를 해제한 경우에도 새로운 로크(lock)를 요청할 수 있다.

② 타임스탬프-순위 기법은 사전에 모든 트랜잭션 쌍 간의 순위를 정해놓음으로써 직렬성을 보장한다.

③ 로킹(locking)규약은 각 데이터 아이템에 언제 로크(lock)를 걸고 해제할 수 있는지를 나타내는 규칙이다.

④ 로킹(locking)의 단위가 속성이 되면 최대의 병행성을 제공 받을 수 있다.

정답 1번

출제 포인트

이 문제는 2단계 로킹 규약, 타임스탬프 순위 기법, 로킹 규약, 로크 단위의 차이를 구분하는 문제이다.

핵심 개념

2단계 로킹 규약은 로크를 획득하는 확장 단계와 로크를 해제하는 축소 단계를 분리하여, 한 번 해제를 시작하면 더 이상 새로운 로크를 얻지 못하게 하는 규약이다.
타임스탬프 순위 기법은 각 트랜잭션에 시간순서를 부여하고 그 순서를 어기는 연산을 거부하여 직렬성을 보장하는 기법이다.
로킹 규약은 데이터 항목에 대해 언제 로크를 걸고 언제 해제할 수 있는지를 정하는 규칙이다.
로크 단위는 속성처럼 세분될수록 병행성은 커지고, 대신 로크 관리 오버헤드는 증가한다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2018 · 6월

선지별 해설

① X
2단계 로킹 규약에서 틀린 부분은 '로크를 해제한 경우에도 새로운 로크를 요청할 수 있다'는 표현이다. 2단계 로킹은 해당 트랜잭션이 한 번이라도 로크 해제를 시작하면 축소 단계에 들어가므로 새로운 로크를 더 요청할 수 없다. 올바르게는 '로크 해제를 시작하기 전까지만 새로운 로크를 요청할 수 있다'이다.

② O
타임스탬프 순위 기법은 각 트랜잭션에 타임스탬프를 부여해 선후관계를 미리 정하고, 그 순서에 맞지 않는 읽기·쓰기 연산을 허용하지 않음으로써 직렬성을 보장한다. 표현은 다소 압축적이지만 취지는 옳다.

③ O
로킹 규약은 각 데이터 아이템에 대해 언제 로크를 설정하고 언제 해제할 수 있는지를 정한 규칙이다. 동시 수행 중 충돌을 통제하는 기본 원칙을 설명한 문장이다.

④ O
로킹 단위가 속성처럼 작아질수록 서로 다른 트랜잭션이 같은 레코드의 다른 속성에 동시에 접근할 수 있으므로 병행성은 가장 커진다. 다만 관리 오버헤드는 커지지만, 선지는 병행성만 묻고 있으므로 옳다.

오답 포인트

2단계 로킹은 '남이 로크를 풀었는가'가 아니라 '내가 이미 언락 단계에 들어갔는가'가 판단 기준이다. 또한 로크 단위가 작아지면 항상 효율이 좋아진다고 생각하기 쉬운데, 병행성 증대와 관리 오버헤드는 별개의 축이다.

다시 나오면 이렇게 판단

동시성 제어 문항은 각 기법이 무엇으로 직렬성을 보장하는지부터 구분하면 빠르다. 2단계 로킹은 획득 단계와 해제 단계를 분리하는 규약이고, 타임스탬프 기법은 시간순서를 강제하는 기법이며, 로크 단위 문장은 병행성과 오버헤드의 교환관계로 판단한다.

암기 포인트

2PL은 언락 시작 후 추가 로크 불가, 타임스탬프는 시간순 강제, 로크 단위가 작을수록 병행성은 증가한다.

NoteBase

문 7. 빅데이터의 특징과 처리에 대한 설명으로 옳지 않은 것을 <보기>에서 모두 고른 것은?

<보기>
ㄱ. NoSQL 데이터베이스는 관계형 데이터베이스보다 더 강한 일관성 모델을 제공한다.
ㄴ. MapReduce는 여러 노드(컴퓨터)에 분산된 데이터를 처리하기 위해 데이터를 한 노드로 집중시켜 처리하고 다시 분산 저장하는 과정을 지원한다.
ㄷ. 빅데이터의 특징을 정의하는 3V는 데이터의 규모(Volume), 다양성(Variety), 처리속도(Velocity)를 의미하며, 빅데이터의 유형은 정형, 반정형, 비정형의 데이터를 모두 포함한다.
ㄹ. 텍스트 마이닝(text mining)은 구조화된 데이터로부터 유용한 정보를 추출하는 기술이다.

① ㄱ, ㄷ
② ㄴ, ㄷ
③ ㄱ, ㄴ, ㄹ
④ ㄴ, ㄷ, ㄹ

정답 3번

출제 포인트

이 문제는 NoSQL의 일관성 특성, MapReduce의 분산 처리 방식, 빅데이터 3 V, 텍스트 마이닝의 대상 데이터를 구별할 수 있는지를 묻는 문제이다.

핵심 개념

NoSQL은 분산 확장성과 유연한 데이터 모델을 중시하는 비관계형 데이터베이스이며, 관계형 데이터베이스보다 항상 더 강한 일관성을 제공하는 개념이 아니다.
MapReduce는 데이터를 한 노드로 모으는 방식이 아니라 분산된 데이터를 여러 노드에서 병렬 처리한 뒤 결과를 집계하는 방식이다.
빅데이터의 3 V는 데이터의 규모(Volume), 다양성(Variety), 처리속도(Velocity)이며, 빅데이터는 정형·반정형·비정형 데이터를 모두 포함한다.
텍스트 마이닝은 주로 비정형 또는 반정형 텍스트 데이터에서 유용한 정보를 추출하는 기술이다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2018 · 6월

선지별 해설

① X
①은 ㄱ, ㄷ을 옳지 않은 것으로 묶었지만 ㄷ은 맞는 진술이다.
ㄱ은 "NoSQL 데이터베이스는 관계형 데이터베이스보다 더 강한 일관성 모델을 제공한다"는 부분이 틀리다. 일반적으로 관계형 데이터베이스는 ACID 기반의 강한 일관성을 중시하고, NoSQL은 확장성과 가용성을 위해 최종 일관성을 채택하는 경우가 많다.
ㄷ은 3 V의 의미와 빅데이터 데이터 유형 설명이 모두 맞다.
틀린 묶음은 ㄱ, ㄷ이 아니라 ㄱ, ㄴ, ㄹ이다.

② X
②는 ㄴ, ㄷ을 옳지 않은 것으로 묶었지만 역시 ㄷ이 맞는 진술이다.
ㄴ은 "데이터를 한 노드로 집중시켜 처리하고 다시 분산 저장한다"는 부분이 틀리다. 실제로는 각 노드에 분산된 데이터를 병렬로 처리하고 결과를 모아 집계한다.
ㄷ은 맞는 진술이므로 ②는 성립하지 않는다.
틀린 묶음은 ㄴ, ㄷ이 아니라 ㄱ, ㄴ, ㄹ이다.

③ O
③은 ㄱ, ㄴ, ㄹ을 옳지 않은 것으로 묶었고 정확하다.
ㄱ은 NoSQL의 일관성을 관계형 데이터베이스보다 더 강하다고 한 점이 틀렸고, ㄴ은 MapReduce를 중앙집중 처리처럼 설명한 점이 틀렸으며, ㄹ은 텍스트 마이닝의 대상 데이터를 구조화된 데이터라고 한 점이 틀렸다.
ㄹ은 "구조화된 데이터"가 아니라 비정형 또는 반정형 텍스트 데이터에서 유용한 정보를 추출하는 기술로 고쳐야 한다.

④ X
④는 ㄴ, ㄷ, ㄹ을 옳지 않은 것으로 묶었지만 ㄷ이 맞는 진술이고, 대신 ㄱ이 포함되어야 한다.
ㄴ과 ㄹ은 틀린 진술이지만 ㄷ은 3 V와 데이터 유형 설명이 모두 맞다.
틀린 묶음은 ㄴ, ㄷ, ㄹ이 아니라 ㄱ, ㄴ, ㄹ이다.

오답 포인트

NoSQL이 빅데이터 환경에서 많이 쓰인다는 이유만으로 관계형 데이터베이스보다 일관성이 더 강하다고 착각하기 쉽다. 또한 MapReduce를 대용량 데이터를 한곳에 모아 처리하는 기술로 오해하기 쉽고, 텍스트 마이닝을 구조화된 데이터 분석으로 혼동하면 ㄹ을 맞는 진술로 보기 쉽다. ㄷ은 3 V와 데이터 유형을 함께 묶어 제시해 길어 보이지만 내용 자체는 정확하므로 여기서 흔들리면 조합형 정답을 놓치게 된다.

NoteBase

다시 나오면 이렇게 판단

조합형 문항은 먼저 ㄱ, ㄴ, ㄷ, ㄹ을 각각 O/X로 독립 판단한 뒤 발문이 "옳은 것"인지 "옳지 않은 것"인지 확인하고 마지막에 조합으로 옮겨야 한다. 이 문제에서는 ㄱ X, ㄴ X, ㄷ O, ㄹ X로 표시하면 바로 ㄱ, ㄴ, ㄹ 조합을 고를 수 있다.

암기 포인트

NoSQL은 확장성과 유연성, MapReduce는 분산 병렬 처리, 3 V는 Volume·Variety·Velocity, 텍스트 마이닝은 비정형 텍스트 분석이다.

NoteBase

문 8. 릴레이션 R(A, B, C, D, E, F)에 대한 함수적 종속성이 <보기>와 같이 주어졌을 때 유추할 수 없는 함수적 종속성은?

<보기>
{C → BD, AF → E, B → F}

① C → F
② C → D
③ C → A
④ AB → E

정답 3번

출제 포인트

이 문제는 주어진 함수적 종속성으로 속성 폐포를 구해 어떤 함수적 종속성이 암스트롱 공리로 유추되는지를 판별하는 문제이다.

핵심 개념

함수적 종속성 추론은 주어진 종속성 집합으로 어떤 속성 집합의 폐포를 계산해 오른쪽 속성이 포함되는지를 확인하는 절차이다.
이 문항에서는 C^+와 AB^+를 보면 된다.
$C \rightarrow BD$이므로 C^+에는 우선 B와 D가 들어간다.
또 $B \rightarrow F$이므로 $C \rightarrow B \rightarrow F$가 되어 C^+는 {C, B, D, F}가 된다.
따라서 C로부터는 A를 얻을 수 없고, B로부터 F를 얻은 뒤 A와 함께 $AF \rightarrow E$를 적용하는 추론은 가능하다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2018 · 6월

선지별 해설

① O

$C \to B$와 $B \to F$를 연쇄 적용하면 $C \to F$가 된다. 추이 규칙으로 유추 가능하다.

② O

보기에서 이미 $C \to BD$가 주어졌으므로 분해 규칙에 의해 $C \to D$가 바로 성립한다.

③ X

C의 폐포는 {C, B, D, F}까지이며 A는 포함되지 않는다. 틀린 부분은 "$C \to A$"이다. 올바르게는 "주어진 종속성만으로는 $C \to A$를 유추할 수 없다"이다.

④ O

$B \to F$이므로 증대 규칙으로 $AB \to AF$가 된다. 여기에 $AF \to E$를 추이 적용하면 $AB \to E$가 된다.

오답 포인트

③은 C로부터 B와 F까지는 계속 확장되므로 A도 얻을 수 있다고 착각하기 쉽다는 점이 함정이다. 그러나 주어진 종속성들 가운데 A를 새롭게 만들어 내는 규칙은 없으므로 폐포에 A가 들어가지 않는다.

다시 나오면 이렇게 판단

함수적 종속성 추론 문제는 각 선택지의 왼쪽 속성 집합 폐포를 먼저 구해 본다. 오른쪽 속성이 폐포에 들어가면 유추 가능하고, 들어가지 않으면 유추 불가능하다고 판단한다.

암기 포인트

정답 찾기는 폐포로 끝난다. C의 폐포는 {C, B, D, F}이고 A가 없으므로 $C \to A$는 성립하지 않는다.

NoteBase

문 9. 색인(indexing) 기법에 대한 설명으로 가장 옳은 것은?

① 비트맵(bitmap) 색인의 크기는 데이터베이스 크기에 영향을 받지 않는다.

② 차수가 n인 B^+-tree는 루트를 제외한 모든 노드가 최소한 n/2 개의 포인터를 갖는다.

③ B^+-tree는 B-tree에서 탐색 키 중복을 제거한 것이다.

④ 색인 순차 파일의 성능은 데이터베이스 크기에 영향을 받지 않는다.

정답 2번

출제 포인트

이 문제는 비트맵 색인, B+-트리, 색인 순차 파일의 기본 성질을 구분하는 문제이다.

핵심 개념

비트맵 색인은 각 값에 대해 비트 벡터를 두는 색인이라 레코드 수가 늘어나면 비트 수가 함께 늘어난다.
B+-트리는 내부 노드는 탐색용 키만 두고 실제 레코드 포인터는 리프에 모아 두는 균형 트리이며, 비루트 노드는 최소 반수 이상 채우는 규칙을 가진다.
색인 순차 파일은 순차 저장 + 색인을 결합한 구조이므로 파일 크기와 오버플로 영역의 증가에 따라 탐색 성능이 영향을 받는다.

선지별 해설

① X: 비트맵 색인의 크기는 데이터베이스 크기와 무관하지 않다. 레코드 수가 증가하면 각 값에 대응하는 비트열 길이도 길어진다. 틀린 표현은 "영향을 받지 않는다"이다. 올바르게는 "비트맵 색인의 크기는 레코드 수와 속성의 값 분포에 영향을 받는다"이다.
② O: 차수가 n인 B+-트리는 루트를 제외한 노드가 최소 반수 이상 채워지는 균형 조건을 따른다. 시험 문맥에서는 비루트 노드가 최소한 n/2개의 포인터를 가진다는 진술로 정리한다.
③ X: B+-트리는 B-트리에서 탐색 키 중복을 제거한 구조가 아니다. 오히려 내부 노드의 분리 키가 리프의 키와 함께 나타나므로 탐색 키가 중복되어 보일 수 있다. 틀린 표현은 "중복을 제거"이다. 올바르게는 "B+-트리는 실제 레코드 포인터를 리프에만 두고 내부 노드는 탐색용 키를 유지하는 구조"이다.
④ X: 색인 순차 파일의 성능은 데이터베이스 크기에 영향을 받는다. 파일이 커지거나 오버플로 블록이 많아지면 접근 비용이 증가할 수 있다. 틀린 표현은 "영향을 받지 않는다"이다. 올바르게는 "색인 순차 파일의 성능은 파일 크기와 오버플로 정도에 영향을 받는다"이다.

오답 포인트

①과 ④는 모두 "영향을 받지 않는다"라는 단정 표현이 함정이다. 저장 구조나 색인 구조는 데이터 양이 커질수록 크기와 성능이 같이 변한다. ③은 B-트리와 B+-트리의 차이를 "중복 제거"로 잘못 외운 경우에 걸리기 쉽다.

NoteBase

다시 나오면 이렇게 판단

색인 문제에서는 먼저 각 구조가 무엇을 저장하는지와 데이터가 늘어날 때 무엇이 같이 증가하는지를 본다. B+-트리는 "내부 노드는 탐색, 리프는 실제 포인터"로 잡고, 비트맵 색인과 색인 순차 파일은 데이터 증가에 무관하다고 말하면 우선 의심한다.

암기 포인트

B+-트리는 내부 탐색·리프 저장, 비트맵과 색인 순차 파일은 데이터 규모의 영향을 받는다.

NoteBase

문 10. 관계형 모델에서 외래키(foreign key)에 대한 설명으로 가장 옳은 것은?

① 다른 릴레이션의 기본키(primary key)를 참조하는 키

② 기본키(primary key)가 아닌 후보키(candidate key)

③ 후보키(candidate key)가 다수일 때 기본키(primary key)로 선정되지 못한 키

④ 각 튜플을 고유하게 식별하지 못하는 키

정답 1번

출제 포인트

이 문제는 외래키, 기본키, 후보키, 대체키의 정의를 구분하는 문제이다.

핵심 개념

외래키는 다른 릴레이션의 기본키를 참조하여 릴레이션 사이의 관계를 표현하는 키이다.
후보키는 튜플을 유일하게 식별할 수 있는 최소 속성 집합이다.
기본키는 후보키 중 대표로 선택된 키이다.
대체키는 후보키가 여러 개일 때 기본키로 선택되지 않은 나머지 후보키이다.

선지별 해설

① O 다른 릴레이션의 기본키를 참조하는 키가 외래키이므로 옳다. 외래키는 참조 무결성을 유지하는 데 쓰인다.
② X 틀린 부분은 "기본키가 아닌 후보키"라는 표현만으로 외래키를 설명한 점이다. 기본키가 아닌 후보키는 외래키가 아니라 대체키에 해당한다.
③ X 틀린 부분은 "후보키가 다수일 때 기본키로 선정되지 못한 키"라는 표현이다. 올바르게는 후보키가 여러 개일 때 기본키로 선택되지 않은 키를 대체키라고 한다.
④ X 틀린 부분은 "각 튜플을 고유하게 식별하지 못하는 키"라는 표현이다. 올바르게는 키는 각 튜플을 고유하게 식별할 수 있어야 하며, 고유 식별이 되지 않으면 키가 아니다.

오답 포인트

②와 ③은 모두 대체키의 설명을 외래키와 바꿔 놓은 함정이다. 외래키는 "다른 릴레이션을 참조한다"는 관계 중심 개념이고, 대체키는 "후보키들 사이의 선택 결과"라는 내부 분류 개념이라는 점에서 다르다.

다시 나오면 이렇게 판단

선지에 "다른 릴레이션의 키를 참조한다"가 나오면 외래키를 먼저 떠올리면 된다. 반대로 "후보키 중 선택되지 않음"이 나오면 대체키로 판단하면 된다. 또한 고유 식별이 되지 않으면 애초에 키가 아니라는 점을 함께 확인하면 된다.

NoteBase

암기 포인트

외래키는 밖을 참조하는 키이고, 대체키는 후보키 내부에서 탈락한 키이다.

NoteBase

문 11. B^+-tree에 대한 설명으로 가장 옳은 것은?

① 정적 색인 구조이다.

② 범위 질의(range query) 처리에 비효율적이다.

③ 리프노드(leaf node)가 다른 레벨에 존재할 수 있다.

④ 루트 외의 노드에는 언제나 절반 이상의 데이터가 들어 있다.

정답 4번

출제 포인트

이 문제는 B+트리의 핵심 성질인 동적 색인, 범위 질의 적합성, 리프 레벨 균형, 노드 점유율을 구분하는 문제이다.

핵심 개념

B+트리는 데이터베이스에서 많이 쓰는 동적 다단계 색인 구조로, 키는 정렬되어 있고 리프 노드가 순차 연결되어 범위 검색에 유리하다.
B-트리 계열은 균형 트리이므로 모든 리프 노드는 같은 레벨에 있다.
또한 루트를 제외한 노드는 삽입·삭제 후에도 최소 점유율을 유지하도록 분할과 병합이 일어난다.

선지별 해설

① X: B+트리는 정적 색인이 아니라 삽입과 삭제에 따라 구조가 변하는 동적 색인 구조이다. 틀린 표현은 "정적 색인 구조"이며, 올바르게는 "동적 다단계 색인 구조"이다.
② X: B+트리는 리프 노드들이 키 순서대로 연결되어 있어 범위 질의를 연속적으로 처리하기 쉽다. 틀린 표현은 "비효율적"이며, 올바르게는 "범위 질의 처리에 효율적"이다.
③ X: B+트리는 균형 트리이므로 모든 리프 노드가 동일한 깊이에 존재한다. 틀린 표현은 "다른 레벨에 존재할 수 있다"이며, 올바르게는 "모든 리프 노드는 같은 레벨에 존재한다"이다.
④ O: 루트를 제외한 노드는 보통 절반 이상 채워지도록 유지되므로 최소 점유율 조건을 만족한다. 따라서 루트 외의 노드에는 언제나 절반 이상의 데이터가 들어 있다는 설명이 가장 옳다.

오답 포인트

②와 ③은 B+트리의 장점을 반대로 뒤집어 놓은 선지이다. 범위 질의는 리프의 순차 연결 때문에 강점이고, 리프 레벨 동일성은 균형 트리의 기본 성질인데 이를 놓치면 오답으로 끌려가기 쉽다.

다시 나오면 이렇게 판단

B+트리 선지가 나오면 먼저 동적인지, 리프가 같은 레벨인지, 리프가 순차 연결되는지, 루트 외 노드의 최소 점유율이 유지되는지를 기준으로 판단하면 된다.

NoteBase

암기 포인트

B+트리 = 동적 균형 트리, 리프 동일 레벨, 리프 순차 연결, 범위 질의 강점이다.

NoteBase

문 12. <보기>의 로그에서 지연 갱신(deferred update) 회복기법을 이용할 때, 시스템 붕괴 후 복구 과정에서 redo되어야 하는 트랜잭션들을 모두 고른 것은?

<보기>
begin_transaction, T3
read, T3, A
write, T3, A, 5
begin_transaction, T1
commit, T3
begin_transaction, T2
checkpoint
read, T1, C
read, T2, B
write, T1, C, 10
begin_transaction, T4
commit, T1
read, T4, D
read, T2, A
write, T4, D, 20
write, T2, B, 15
write, T2, A, 30
commit, T2
read, T4, A ←시스템 붕괴

① T1

② T2

③ T1, T2

④ T1, T2, T3

정답 3번

출제 포인트

지연 갱신은 commit 이전의 갱신 내용을 데이터베이스에 반영하지 않으므로, 복구 때는 마지막 checkpoint 이후에 commit된 트랜잭션만 redo 대상으로 잡는지를 묻는 문제이다.

NoteBase

올바른 순서

1. 복구 기준점은 마지막 checkpoint로 잡는다.
2. checkpoint 이전에 이미 commit된 트랜잭션은 checkpoint 시점까지 반영이 끝난 것으로 보아 redo 대상에서 제외한다.
3. checkpoint 시점에 진행 중이었거나 checkpoint 이후에 시작된 트랜잭션 가운데 crash 이전에 commit된 트랜잭션만 고른다.
4. commit되지 않은 트랜잭션은 지연 갱신에서는 DB에 반영되지 않았으므로 redo도 undo도 하지 않는다.

왜 이 순서: 지연 갱신은 commit 여부가 실제 반영의 전제이므로, 먼저 checkpoint를 기준으로 범위를 줄인 뒤 commit 완료 여부를 따져야 redo 대상을 정확히 정할 수 있다.

선지별 해설

① X

T1은 checkpoint 이전에 시작했지만 checkpoint 이후에 commit되었으므로 redo 대상에 포함된다. 다만 T2도 checkpoint 이후 commit되었으므로 ①처럼 T1만 고르면 누락이 생긴다. 틀린 표현은 'T1만 redo된다'이고, 올바르게는 'T1과 T2가 redo된다'이다.

② X

T2는 checkpoint 이후 commit되었으므로 redo 대상이 맞다. 그러나 T1 역시 checkpoint 시점에는 미완료였고 crash 전 commit되었으므로 함께 redo해야 한다. 틀린 표현은 'T2만 redo된다'이고, 올바르게는 'T1과 T2가 redo된다'이다.

③ O

T1은 checkpoint 시점에는 진행 중이었고 이후 commit되었으므로 redo 대상이다. T2도 checkpoint 시점에 진행 중이었고 crash 전에 commit되었으므로 같은 이유로 redo 대상이다. 따라서 redo 대상은 T1, T2이다.

④ X

T3는 checkpoint 이전에 이미 commit되었다. 마지막 checkpoint 이후 redo 대상은 checkpoint 이후 commit된 트랜잭션이므로 T3는 다시 실행하지 않는다. 틀린 표현은 'T3도 redo된다'이고, 올바르게는 'T3는 redo 대상이 아니고 T1, T2만 redo된다'이다.

오답 포인트

checkpoint 이전에 시작했는지 여부만 보고 T1을 빼거나, commit만 보면 T3까지 포함하는 실수가 잦다. 이 문제의 함정은 시작 시점이 아니라 마지막 checkpoint 이후 commit되었는가에 있다.

NoteBase

다시 나오면 이렇게 판단

지연 갱신 로그 문제는 마지막 checkpoint를 먼저 찾고, 그 이후에 commit된 트랜잭션만 redo 후보로 잡는다. 그다음 crash 시점까지 commit되지 않은 트랜잭션을 제외하면 정답 집합이 정리된다.

암기 포인트

지연 갱신 복구는 undo보다 redo 중심이며, checkpoint 이후 commit된 것만 다시 반영한다.

NoteBase

문 13. <보기>의 릴레이션 Student (sno, sname, address, score, year, dept)에 대해, 여러 관계 대수식을 수행하려고 한다. 관계 대수식의 결과로 생성되는 릴레이션의 차수(degree)와 카디날리티(cardinality)가 바르게 짝지어지지 않은 것은?

<보기>

sno	sname	address	score	year	dept
10	홍길동	서울	50	4	100
11	김철수	대전	90	3	200
12	김영희	대구	55	2	100
13	황이수	부산	60	3	200
14	정순희	서울	70	1	300
15	이태백	서울	100	4	300
16	최지현	부산	70	2	100
17	김현정	인천	65	3	100

① $\sigma_{year>=3}(Student) - \sigma_{dept=100}(Student)$
차수 : 6 카디날리티 : 2

② $\Pi_{sno,sname,score}(\sigma_{score>60}(Student))$
차수 : 3 카디날리티 : 5

③ $\sigma_{year>=4}(Student) \cup \sigma_{year<=1}(Student)$
차수 : 6 카디날리티 : 3

④ $\sigma_{address='대전' \text{ and } year=3}(\Pi_{sno,address,score,year}(Student))$
차수 : 4 카디날리티 : 1

정답 1번

출제 포인트

이 문제는 선택, 차집합, 투영, 합집합 연산 뒤 결과 릴레이션의 차수와 카디널리티를 정확히 세는 문제이다.

조건 정리

주어진 Student 릴레이션의 속성은 sno, sname, address, score, year, dept로 6개이다.
차수는 결과 릴레이션에 남는 속성 수이다.
카디널리티는 결과 릴레이션에 남는 서로 다른 튜플 수이다.
선택 연산은 행만 거르므로 차수는 유지된다.
투영 연산은 지정한 속성만 남기므로 차수가 바뀐다.
차집합과 합집합은 같은 스키마끼리 수행하며 결과에서 중복 튜플은 하나로 본다.

NoteBase

사용할 공식

공식 대신 연산별 결과 규칙을 적용한다.
선택 σ 후 차수는 원래와 같고, 투영 Π 후 차수는 투영한 속성 개수와 같다.
카디널리티는 조건을 만족하는 튜플 수를 세고, 차집합은 앞 집합에서 뒤 집합을 뺀 수, 합집합은 중복을 제거한 전체 수로 판단한다.
이 규칙을 쓰는 이유는 이 문항이 수치 계산보다 관계대수 연산 결과의 속성 수와 튜플 수를 묻기 때문이다.

계산 과정

1. ①에서 year>=3인 튜플은 sno 10, 11, 13, 15, 17로 5개이고, dept=100인 튜플은 sno 10, 12, 16, 17로 4개이다. 따라서 차집합은 sno 11, 13, 15가 남아 카디널리티는 3이다.
2. ②에서 score>60인 튜플은 sno 11, 14, 15, 16, 17로 5개이다. sno, sname, score만 투영해도 중복이 생기지 않으므로 카디널리티는 5이고 차수는 3이다.
3. ③에서 year>=4인 튜플은 sno 10, 15로 2개이고, year<=1인 튜플은 sno 14로 1개이다. 두 집합이 겹치지 않으므로 합집합 카디널리티는 3이다.
4. ④에서 sno, address, score, year만 남기면 차수는 4이다. 그 뒤 address='대전' and year=3을 만족하는 튜플은 sno 11 한 개뿐이므로 카디널리티는 1이다.

NoteBase

선지별 해설

① X
- year>=3 결과는 5개, dept=100 결과는 4개이지만 차집합에서는 공통 튜플 sno 10과 17이 제거되고 sno 11, 13, 15가 남는다.
- 따라서 차수 6은 맞지만 카디널리티 2가 틀리다. 올바르게는 카디널리티 3이다.
- 이 선지는 잘못 짝지어진 보기이므로 정답이 된다.

② O
- score>60은 60 초과이므로 score가 60인 sno 13은 제외되고 sno 11, 14, 15, 16, 17만 남는다.
- sno, sname, score를 투영하면 속성 3개만 남아 차수는 3이고, sno가 모두 달라 중복 제거가 일어나지 않아 카디널리티는 5이다.

③ O
- year>=4는 sno 10, 15이고 year<=1은 sno 14이다.
- 합집합 후 튜플은 총 3개이므로 카디널리티 3이 맞다. 선택 연산 둘을 합집합해도 스키마는 그대로이므로 차수 6도 맞다.

④ O
- 먼저 sno, address, score, year를 투영했으므로 결과 속성은 4개이다.
- 그 결과에서 address='대전'이고 year=3인 튜플은 김철수 한 건뿐이므로 카디널리티 1이 맞다.

오답 포인트

①에서 두 선택 결과의 개수만 보고 5-4=1처럼 기계적으로 빼거나, 공통 튜플이 무엇인지 확인하지 않으면 오답이 나온다.
②에서는 score>60을 score>=60으로 잘못 읽어 sno 13을 포함시키기 쉽다.
투영 뒤 중복 제거 여부를 확인하지 않고 원래 튜플 수를 그대로 적는 실수도 자주 나온다.

다시 나오면 이렇게 판단

관계대수의 차수와 카디널리티 문제는 먼저 연산 순서를 적고, 선택이면 남는 튜플을 실제로 표시한 뒤, 투영이면 남는 속성 수와 중복 제거 여부를 따로 확인하는 방식으로 판단한다.

암기 포인트

선택은 차수 유지, 투영은 차수 변화, 카디널리티는 최종 서로 다른 튜플 수이다.

NoteBase

문 14. 개체 관계 모델(Entity-Relationship model)에 대한 설명으로 가장 옳지 않은 것은?

① 서브 타입 개체는 슈퍼 타입 개체로부터 속성은 물론 관계도 상속받는다.

② 하나의 속성이 여러 개의 속성값을 가질 수 있는 다중값 속성도 표시할 수 있다.

③ 약한 개체 타입(weak entity type)은 자신의 애트리뷰트만으로 키를 명세할 수 없다.

④ 하나의 개체에서 스스로 관계를 갖는 재귀적 관계는 표현할 수 없다.

정답 4번

출제 포인트

ER 모델에서 서브타입 상속, 다중값 속성, 약한 개체 타입, 재귀적 관계의 표현 가능 여부를 구분하는 문제이다.

핵심 개념

슈퍼타입과 서브타입: 서브타입은 슈퍼타입의 속성과 관계를 상속받는 하위 개체 타입이다.
다중값 속성: 하나의 개체가 같은 속성에 대해 여러 값을 가질 수 있음을 나타내는 속성이다.
약한 개체 타입: 자신의 속성만으로는 고유하게 식별되지 않아 소유자 개체의 키에 의존하는 개체 타입이다.
재귀적 관계: 하나의 개체 타입이 자기 자신과 맺는 관계이며 ER 모델에서 표현할 수 있다.

선지별 해설

① O: 서브타입 개체는 슈퍼타입 개체의 속성뿐 아니라 관계 타입도 상속받는다. 특수화된 하위 개체가 상위 개체의 구조를 이어받는다는 점에서 맞는 설명이다.
② O: 다중값 속성은 전화번호, 자격증처럼 한 개체가 동일 속성에 여러 값을 가질 수 있는 경우를 나타낸다. 하나의 속성이 여러 속성값을 가질 수 있다는 설명이 맞다.
③ O: 약한 개체 타입은 자신의 애트리뷰트만으로 키를 명세할 수 없다. 식별은 소유자 개체의 키와 약한 개체의 부분 키를 결합해서 이루어진다.
④ X: 틀린 표현은 "표현할 수 없다"이다. 올바르게는 하나의 개체 타입이 자기 자신과 관계를 맺는 재귀적 관계도 ER 모델에서 표현할 수 있다.

오답 포인트

약한 개체 타입과 재귀적 관계를 함께 묻기 때문에 식별 가능성 문제와 관계 표현 가능성 문제를 혼동하기 쉽다. ④는 ER 모델이 표현하지 못하는 것처럼 단정하는 문장이라서, ER 모델이 자기참조 관계도 다룰 수 있다는 점을 놓치면 오답이 된다.

다시 나오면 이렇게 판단

ER 모델 선지는 먼저 속성 표현, 식별 방식, 관계 표현 가능 여부로 나누어 판단하면 된다. 자기 자신과의 관계를 말하면 재귀적 관계를 떠올리고, 자신의 속성만으로 식별되지 않는다고 하면 약한 개체 타입을 떠올리면 된다.

NoteBase

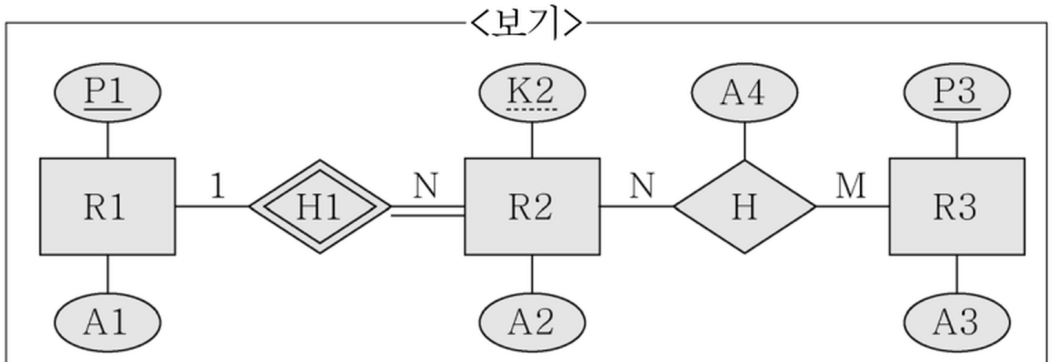
암기 포인트

서브타입은 상속, 약한 개체는 소유자 의존, 재귀적 관계는 자기참조 표현 가능으로 정리된다.

NoteBase

문 15. <보기>의 ERD를 표현한 릴레이션 스키마에 속하는 것을 모두 고른 것은? (단, R2(P2, A2, P1)에서 밑줄 친 속성은 기본키이며, P1(R2) → P1(R1)에서 R2의 P1은 R1의 P1을 참조하는 외래키이다.)

<보기>



ㄱ. R1(P1, A1)

ㄴ. R2(K2, P1, A2, P3) P1(R2) → P1(R1), P3(R2) → P3(R3)

ㄷ. R2(K2, A2, P1) P1(R2) → P1(R1)

ㄹ. R2(K2, P1, A2) P1(R2) → P1(R1)

ㅁ. R3(P3, A3, K2, P1) K2(R3) → K2(R2), P1(R3) → P1(R2)

ㅂ. R3(P3, A3)

ㅅ. H(K2, P1, P3, A4) K2(H) → K2(R2), P1(H) → P1(R2), P3(H) → P3(R3)

① ㄱ, ㅂ, ㅅ

② ㄷ, ㄹ, ㅂ

③ ㄱ, ㄹ, ㅁ

④ ㄷ, ㅂ, ㅅ

정답 1번

출제 포인트

이 문제는 ERD에서 약한 개체의 식별자 구성과 $N:M$ 관계에 붙은 관계 속성의 릴레이션 변환 규칙을 도식으로 읽어 내는 문제이다.

NoteBase

선지별 해설

1번 O: ㄱ, ㅂ, ㅅ의 조합이며 셋 다 올바른 릴레이션이다. `R1(P1, A1)`은 강한 개체 R1의 기본 릴레이션이므로 맞다. `R3(P3, A3)`도 강한 개체 R3의 기본 릴레이션이므로 맞다. R2와 R3 사이의 H는 $N:M$ 관계이고 관계 속성 A4를 가지므로, 참여 개체에 흡수하지 않고 `H(K2, P1, P3, A4)`처럼 별도 릴레이션으로 분해하며, 이때 기본키는 R2의 식별자 $\{K2, P1\}$과 R3의 기본키 P3을 합친 $\{K2, P1, P3\}$이다.
2번 X: ㄷ, ㄹ, ㅂ의 조합인데 ㄷ이 틀려 전체가 틀리다. H1이 이중 마름모로 그려진 식별 관계이므로 R2는 소유자 R1에 의존하는 약한 개체이고, 점선 밑줄로 표시된 부분키 K2만으로는 식별되지 않는다. 따라서 R2의 기본키는 부분키 K2와 소유자 키 P1을 함께 묶은 복합키 $\{K2, P1\}$이어야 한다. ㄷ의 `R2(K2, A2, P1)`은 K2에만 밑줄을 그어 K2를 단독 기본키로 삼았으므로 약한 개체 규칙에 어긋나 틀리다. 같은 조합의 ㄹ `R2(K2, P1, A2)`는 K2와 P1에 함께 밑줄을 그어 복합키로 표기했으므로 그 자체로는 올바른 표현이지만, 함께 묶인 ㄷ이 틀리므로 2번 전체가 틀리다.
3번 X: ㄱ, ㄹ, ㅁ의 조합인데 ㅁ이 틀려 전체가 틀리다. ㄱ과 ㄹ은 각각 R1과 약한 개체 R2를 바르게 나타내지만, ㅁ의 `R3(P3, A3, K2, P1)`은 $N:M$ 관계 H를 R3에 흡수해 R2의 키 K2, P1을 R3로 끌어들였고 관계 속성 A4도 빠졌으므로 틀리다. 올바르게는 `R3(P3, A3)`는 그대로 두고 관계 H를 `H(K2, P1, P3, A4)`로 따로 만든다.
4번 X: ㄷ, ㅂ, ㅅ의 조합인데 ㅂ과 ㅅ은 맞지만 ㄷ이 틀려 전체가 틀리다. 약한 개체 R2는 소유자 키 P1을 포함한 복합키로 식별되어야 하는데, ㄷ은 K2만 기본키로 두어 P1을 키에서 빠뜨렸다.

오답 포인트

점선 밑줄 K2와 이중 마름모 H1을 보고도 약한 개체 규칙을 놓치면 K2만 기본키로 둔 ㄷ을 맞는 것으로 보기 쉽다. ㄴ의 `R2(K2, P1, A2, P3)`는 R3의 키 P3까지 R2에 넣었는데, R2와 R3의 관계는 별도 릴레이션 H로 처리되므로 P3은 R2에 들어가지 않아 틀리다. 또 $N:M$ 관계에 속성 A4가 붙어 있는데도 이를 R3에 흡수해 ㅁ을 고르기 쉽다. 이 문항은 약한 개체의 복합식별자와 $N:M$ 관계의 별도 릴레이션 생성 여부를 동시에 묻는 함정이 있다.

다시 나오면 이렇게 판단

도식 문제에서는 먼저 약한 개체인지부터 본다. 부분키가 보이면 소유자 기본키가 함께 들어가 복합키가 된다. 다음으로 관계의 카디널리티를 본다. $N:M$이고 관계 속성이 있으면 참여 개체 릴레이션에 흡수하지 않고 관계 자체를 별도 릴레이션으로 만든다.

암기 포인트

약한 개체는 소유자 키와 함께 식별하고, $N:M$ 관계에 속성이 있으면 관계 릴레이션을 따로 둔다.

NoteBase

문 16. <보기>와 같이 릴레이션 R과 함수 종속성 F가 정의되어 있다. 제3정규형(3NF)을 만족하도록 분해한 릴레이션들을 나타낸 것으로 가장 옳은 것은?

<보기>
R(A, B, C, D, E)
F = {BE → A, E → D, D → C}

① (D, C), (E, D), (B, E, A)
② (D, C), (B, E, A, D)
③ (D, C), (B, E, A), (A, E)
④ (E, D, C), (B, E, A)

정답 1번

출제 포인트

이 문제는 함수 종속성의 최소 커버를 구한 뒤 3NF 합성 결과가 어떤 릴레이션 집합이 되는지를 판단하는 문제이다.

핵심 개념

3NF 합성은 최소 커버의 각 함수 종속성 $X \to Y$에 대해 보통 XY 형태의 릴레이션을 만들고, 후보키를 포함하는 릴레이션이 없으면 후보키 릴레이션을 추가하는 방식이다.
이 문제의 최소 커버는 그대로 $\{BE \to A,\ E \to D,\ D \to C\}$이다.
후보키는 BE이다. $BE^{+} = \{A, B, C, D, E\}$가 되지만 B와 E는 다른 속성으로 유도되지 않으므로 BE가 최소 후보키이다.

선지별 해설

1번(O): 최소 커버의 각 종속성으로부터 (D,C), (E,D), (B,E,A)를 만들면 된다. (B,E,A) 안에 후보키 BE가 포함되어 무손실 분해 조건도 만족하고, 각 함수 종속성도 그대로 보존된다.
2번(X): (B,E,A,D)에서는 $BE \to A$, $E \to D$가 성립한다. 이 릴레이션의 후보키는 여전히 BE인데, E는 슈퍼키가 아니고 D도 주요 속성이 아니므로 $E \to D$ 때문에 3NF를 만족하지 못한다. 틀린 표현은 (B,E,A,D)로 합쳐도 3NF라는 점이며, 올바르게는 (E,D)를 별도 릴레이션으로 둔다.
3번(X): (A,E)를 추가해도 필요한 함수 종속성 $E \to D$를 보존하지 못한다. 틀린 표현은 (A,E)가 (E,D)를 대신할 수 있다는 점이며, 올바르게는 (E,D)가 반드시 포함되어야 한다.
4번(X): (E,D,C)에서는 $E \to D$, $D \to C$가 함께 성립한다. 이 릴레이션의 후보키는 E인데 D는 슈퍼키가 아니고 C도 주요 속성이 아니므로 $D \to C$가 3NF 위반이 된다. 틀린 표현은 D와 C를 E와 함께 한 릴레이션에 두어도 된다는 점이며, 올바르게는 (E,D)와 (D,C)로 나누어야 한다.

NoteBase

오답 포인트

후보키 BE를 찾지 않고 모양만 비슷한 릴레이션을 고르면 2번과 4번처럼 부분 릴레이션 내부의 이행 종속을 놓치기 쉽다. 3번은 불필요한 릴레이션을 추가했지만 정작 필요한 종속성 $E \rightarrow D$를 잃는 함정이다.

다시 나오면 이렇게 판단

먼저 최소 커버를 구하고 후보키를 찾는다. 그다음 각 함수 종속성마다 대응하는 릴레이션이 살아 있는지, 합쳐 놓은 릴레이션 내부에서 결정자가 슈퍼키가 아닌 종속성이 새로 남지 않는지를 확인하면 된다.

암기 포인트

3NF 합성은 최소 커버별 릴레이션 생성, 후보키 포함 확인, 합친 뒤 생기는 이행 종속 재점검의 순서로 정리된다.

NoteBase

문 17. <보기>의 릴레이션 R에 존재하는 함수 종속성을 나타낸 것으로 가장 옳은 것은?

<보기>

R

A	B	C	D	E
1	b	10	2	a
2	a	10	2	a
1	b	10	3	a
1	b	20	3	b

① A → BC

② C → DE

③ DE → C

④ BC → D

정답 3번

출제 포인트

이 문제는 릴레이션 인스턴스에서 결정자 값이 같을 때 종속자 값이 항상 하나로 정해지는지 확인하여 함수 종속성을 판별하는 문제이다.

자료 읽는 순서

1. 왼쪽 속성값이 같은 튜플끼리 묶는다.
2. 그 묶음에서 오른쪽 속성값이 하나로 유지되는지 본다.
3. 하나라도 다른 값이 나오면 해당 함수 종속성은 성립하지 않는다고 판단한다.

NoteBase

선지별 해설

① X: A가 1인 튜플은 1행, 3행, 4행이다. 이때 B는 모두 b이지만 C는 10, 10, 20으로 달라지므로 A → BC는 성립하지 않는다. 틀린 표현은 "A → BC"이고, 올바르게는 "A → B는 성립하지만 A → C는 성립하지 않는다"이다.
② X: C가 10인 튜플은 1행, 2행, 3행이다. 이때 E는 모두 a이지만 D는 2, 2, 3으로 달라지므로 C → DE는 성립하지 않는다. 틀린 표현은 "C → DE"이고, 올바르게는 "C → E는 성립하지만 C → D는 성립하지 않는다"이다.
③ O: 같은 DE 값을 가진 튜플을 보면 (2, a)는 1행과 2행에서 모두 C가 10이고, (3, a)는 3행에서 C가 10, (3, b)는 4행에서 C가 20이다. 같은 DE에서 서로 다른 C가 나온 경우가 없으므로 DE → C는 성립한다.
④ X: BC가 (b, 10)인 튜플은 1행과 3행이다. 이때 D가 2와 3으로 달라지므로 BC → D는 성립하지 않는다. 틀린 표현은 "BC → D"이고, 올바르게는 "BC가 같아도 D는 달라질 수 있다"이다.

오답 포인트

반복되는 결정자 값이 있는지 먼저 보지 않고 속성 이름의 조합만 보고 판단하면 틀리기 쉽다. 특히 ①은 A가 1일 때 B만 보고 맞다고 착각하기 쉽고, ②는 C가 10일 때 E가 모두 a라는 점 때문에 D까지 함께 결정된다고 오해하기 쉽다.

다시 나오면 이렇게 판단

함수 종속성 문제는 왼쪽 속성값이 같은 행을 먼저 찾고, 그 행들에서 오른쪽 속성값이 하나라도 달라지는지 확인하여 판단한다. 같은 결정자에서 다른 종속값이 한 번이라도 나오면 그 종속성은 바로 깨진다.

암기 포인트

함수 종속성은 "같은 왼쪽 값이면 오른쪽 값도 항상 같아야 한다"는 규칙으로 판단한다.

NoteBase

문 18. 무결성 규정(integrity rule)에 대한 설명으로 가장 옳지 않은 것은?

① 트리거(trigger)를 사용하면 하나의 릴레이션이 변경될 때, 다른 릴레이션도 변경될 수 있다.

② 무결성 규정은 시스템 카탈로그(system catalog)나 데이터 사전에 저장된다.

③ 갱신 연산을 할 때, 릴레이션 무결성 규정은 트랜잭션이 완전히 수행된 이후에 적용된다.

④ SQL에서 CREATE 문으로 도메인을 생성할 때, CHECK 구문을 통해 도메인 제약 조건을 지정할 수 있다.

정답 3번

출제 포인트

무결성 규정의 저장 위치, 트리거와의 관계, 제약 검사 시점, 도메인 제약 지정 방법을 구별하는 문제이다.

핵심 개념

무결성 규정은 데이터베이스가 항상 만족해야 하는 일관성 조건이다.
이 규정은 시스템 카탈로그나 데이터 사전에 메타데이터로 저장된다.
도메인 제약은 속성값의 허용 범위를 제한하고, 갱신 연산은 이 제약을 위반하면 허용되지 않는다.

선지별 해설

① O
트리거는 특정 릴레이션의 삽입·삭제·수정 이벤트에 반응해 다른 릴레이션에 추가 갱신을 수행할 수 있다.
따라서 한 릴레이션의 변경이 다른 릴레이션의 변경으로 이어질 수 있다는 설명은 옳다.

② O
무결성 규정은 데이터 정의의 일부이므로 시스템 카탈로그 또는 데이터 사전에 저장된다. 제약 조건을 DBMS가 관리하려면 이런 메타정보 형태로 보관되어야 한다.

③ X
틀린 부분은 "트랜잭션이 완전히 수행된 이후에 적용된다"이다. 무결성 규정은 갱신 결과가 반영되는 과정에서 위반 여부를 검사하며, 위반 시 연산이 거부되거나 트랜잭션이 완료되지 않는다. 올바르게는 "무결성 규정은 갱신 연산 중 또는 커밋 시점까지 검사되어 위반 상태의 반영을 막는다"이다.

④ O
표준 SQL에서는 CREATE DOMAIN으로 도메인을 정의할 때 CHECK 절을 사용해 허용값 범위나 조건을 지정할 수 있다. 따라서 도메인 제약 조건을 CHECK 구문으로 지정할 수 있다는 설명은 옳다.

NoteBase

오답 포인트

③은 지연 검사 개념과 사후 적용 개념을 혼동시키는 선지이다. 일부 제약은 커밋 시점까지 미룰 수 있어도, 위반 상태를 그대로 확정한 뒤 나중에 적용하는 규칙은 아니다.

다시 나오면 이렇게 판단

무결성 문제는 "어디에 저장되는가", "언제 검사되는가", "무엇을 제한하는가"를 나누어 본다. 특히 "트랜잭션이 끝난 뒤에 적용된다"처럼 제약 검사를 사후 처리로 서술하면 대체로 틀린 진술이다.

암기 포인트

무결성 규정은 카탈로그에 저장되고, 갱신 시 검사되며, 도메인 제약은 CHECK로 건다.

NoteBase

문 19. 로킹(locking) 기법에 대한 설명으로 가장 옳지 않은 것은?

① 하나의 트랜잭션이 어떤 데이터를 액세스 하고 있는 경우 그 데이터를 다른 트랜잭션이 접근할 수 없도록 하는 것이 로킹(locking) 기법이다.

② 로크(lock)는 테이블(table), 튜플(tuple), 속성(attribute) 단위로 설정할 수 있으며, 이를 로킹(locking) 단위라고 한다.

③ 로킹(locking) 단위가 커지게 되면 병행성 수준이 높아지는 대신 로킹(locking) 오버헤드가 증가한다.

④ 둘 이상의 트랜잭션들이 서로 맞물려 상대방의 잠금이 해제될 때까지 무한정 대기 상태가 되는 것을 교착상태(deadlock)라고 한다.

정답 3번

출제 포인트

이 문제는 로킹의 개념, 로킹 단위와 병행성·오버헤드의 관계, 교착상태의 의미를 구별하는 문제이다.

핵심 개념

로킹은 동시 실행되는 트랜잭션들의 데이터 접근을 제어하여 일관성을 유지하는 기법이다.
로킹 단위는 테이블, 튜플, 속성처럼 잠금을 거는 대상의 크기이다.
로킹 단위가 작을수록 병행성은 높아지지만 잠금 관리 비용은 커지고, 로킹 단위가 클수록 병행성은 낮아지지만 잠금 관리 비용은 줄어든다.
교착상태는 둘 이상의 트랜잭션이 서로가 가진 자원을 기다리며 무한 대기하는 상태이다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2018 · 6월

선지별 해설

① O
하나의 트랜잭션이 잠금을 보유하면 다른 트랜잭션의 충돌 접근을 제한하여 데이터의 일관성을 지키는 것이 로킹의 기본 취지이므로 옳다.

② O
록은 테이블, 튜플, 속성 등 여러 수준에 설정할 수 있고, 이렇게 잠금을 거는 대상의 크기를 로킹 단위라고 하므로 옳다.

③ X
틀린 부분은 "로킹 단위가 커지게 되면 병행성 수준이 높아지고 로킹 오버헤드가 증가한다"는 표현이다.
실제로는 로킹 단위가 커지면 한 번에 묶이는 데이터 범위가 넓어져 병행성은 낮아지고, 관리해야 할 잠금 수가 줄어 로킹 오버헤드는 감소한다.

④ O
둘 이상의 트랜잭션이 서로 상대방의 잠금이 풀리기만 기다리며 진행하지 못하는 상태를 교착상태라고 하므로 옳다.

오답 포인트

로킹 단위가 크면 관리가 편해지므로 성능이 좋아질 것이라고 단순 연결하면서 병행성까지 같이 높아진다고 착각하기 쉽다. 이 문항은 "병행성"과 "잠금 관리 오버헤드"가 서로 반대 방향으로 움직인다는 점을 묻는다.

다시 나오면 이렇게 판단

로킹 단위가 커지는지 작아지는지를 먼저 본다. 단위가 크면 병행성은 낮아지고 오버헤드는 줄어들며, 단위가 작으면 병행성은 높아지고 오버헤드는 커진다고 판단하면 된다.

암기 포인트

큰 락은 병행성 감소·오버헤드 감소, 작은 락은 병행성 증가·오버헤드 증가이다.

NoteBase

문 20. <보기>의 릴레이션 R과 S에서 주어진 관계 대수식의 결과로 가장 옳지 않은 것은?

<보기>

R

A	B	C
a1	2	c4
a2	2	c2
a3	5	c1

S

B	D
2	d1
4	d2

(단, 사용된 외부조인 기호는 아래와 같다.)

=x : 왼쪽 외부조인

x= : 오른쪽 외부조인

=x= : 완전 외부조인

① $\pi_{A,C}(R =x S)$

A	C
a1	c4
a2	c2
a3	c1

② $\sigma_{B \neq 2}(R =x= S)$

A	B	C	B	D
a3	5	c1	^	^
^	^	^	4	d2

③ $\pi_{A,C}(R\ x=\ \sigma_{B=2}(S))$

A	C
a1	c4
a2	c2

④ $\sigma_{B=2}(\pi_{A,C}(R) =x S)$

A	C	B	D
a1	c4	2	d1
a2	c2	2	d1

NoteBase

정답 4번

출제 포인트

공통 속성 B를 기준으로 외부조인 결과를 만든 뒤, 투영과 선택을 적용했을 때 어떤 튜플이 남는지 판단하는 문제이다.

자료 읽는 순서

1. 여기서 외부조인은 자연조인을 바탕으로 하므로, 공통 속성이 있으면 그 값이 같은 튜플을 잇고 공통 속성이 하나도 없으면 조인 조건이 사라져 카테시안 곱이 된다. R와 S의 공통 속성은 B이므로 B값이 같은 튜플을 먼저 매칭한다.
2. 왼쪽 외부조인은 왼쪽 릴레이션의 모든 튜플, 오른쪽 외부조인은 오른쪽 릴레이션의 모든 튜플, 완전 외부조인은 양쪽의 모든 튜플을 보존한다.
3. 조인 결과를 만든 뒤에 투영과 선택을 순서대로 적용한다.

NoteBase

기출OX 해설집 · 공무원/서울시/7급 · 데이터베이스론 · 2018 · 6월

선지별 해설

① O
R 왼쪽 외부조인 S에서 B=2인 두 행은 S의 (2,d1)과 매칭되고, B=5인 (a3,5,c1)은 오른쪽 속성만 비어 있는 채 유지된다.
따라서 A,C만 투영하면 (a1,c4), (a2,c2), (a3,c1)이 남으므로 제시 결과가 맞다.

② O
R 완전 외부조인 S에서는 B=2로 매칭된 (a1,2,c4,2,d1), (a2,2,c2,2,d1) 두 행과, 매칭되지 않은 R의 (a3,5,c1) 행, S의 (4,d2) 행이 모두 남고, 매칭되지 않은 쪽은 상대 속성이 널(^)로 채워진다.
여기서 $B \neq 2$를 적용하면 B가 2인 두 행이 빠지고 (a3,5,c1,^,^)와 (^,^,^,4,d2)만 남으므로 제시 결과가 맞다.

③ O
$\sigma_{B=2}(S)$는 (2,d1) 하나만 남긴다.
R 오른쪽 외부조인 $\sigma_{B=2}(S)$에서는 오른쪽 릴레이션의 B=2 행이 보존되고, 이에 매칭되는 R의 두 행 (a1,2,c4), (a2,2,c2)가 모두 나온다.
따라서 A,C만 투영하면 (a1,c4), (a2,c2)가 남으므로 맞다.

④ X
$\pi_{A,C}(R)$는 (a1,c4), (a2,c2), (a3,c1) 세 행이며 이 투영으로 B 속성이 사라진다.
이때 투영 결과의 속성 $\{A, C\}$와 S의 속성 $\{B, D\}$ 사이에는 공통 속성이 하나도 없으므로, 자연조인을 바탕으로 한 왼쪽 외부조인은 조인 조건이 사라져 두 릴레이션의 카테시안 곱과 같아진다.
따라서 세 행과 S의 두 행이 모두 짝지어져 $3 \times 2 = 6$개의 튜플이 만들어지고, 여기서 $\sigma_{B=2}$는 S에서 온 B가 2인 튜플만 남긴다.
그 결과는 (a1,c4,2,d1), (a2,c2,2,d1), (a3,c1,2,d1)의 세 개여야 하는데, 제시된 표는 (a3,c1,2,d1)을 빠뜨려 두 개만 두었으므로 틀리다.

오답 포인트

④는 원래 R에 있던 B값 2만 머릿속에 남겨 두고 판단하면 틀리기 쉽다. 그러나 투영 $\pi_{A,C}$를 먼저 적용한 순간 B 속성은 사라지므로, 다음 조인은 더 이상 B를 기준으로 이루어지지 않는다.

다시 나오면 이렇게 판단

자연조인이나 외부조인 문제는 각 단계가 끝난 뒤 남아 있는 속성 집합을 다시 확인하면 된다. 조인 속성이 투영으로 삭제되었으면 다음 조인은 그 속성으로 묶이지 않는다고 판단한다.

암기 포인트

자연조인은 현재 남아 있는 공통 속성으로만 수행된다. B를 투영으로 지우면 다음 조인은 B로 묶이지 않는다.

NoteBase