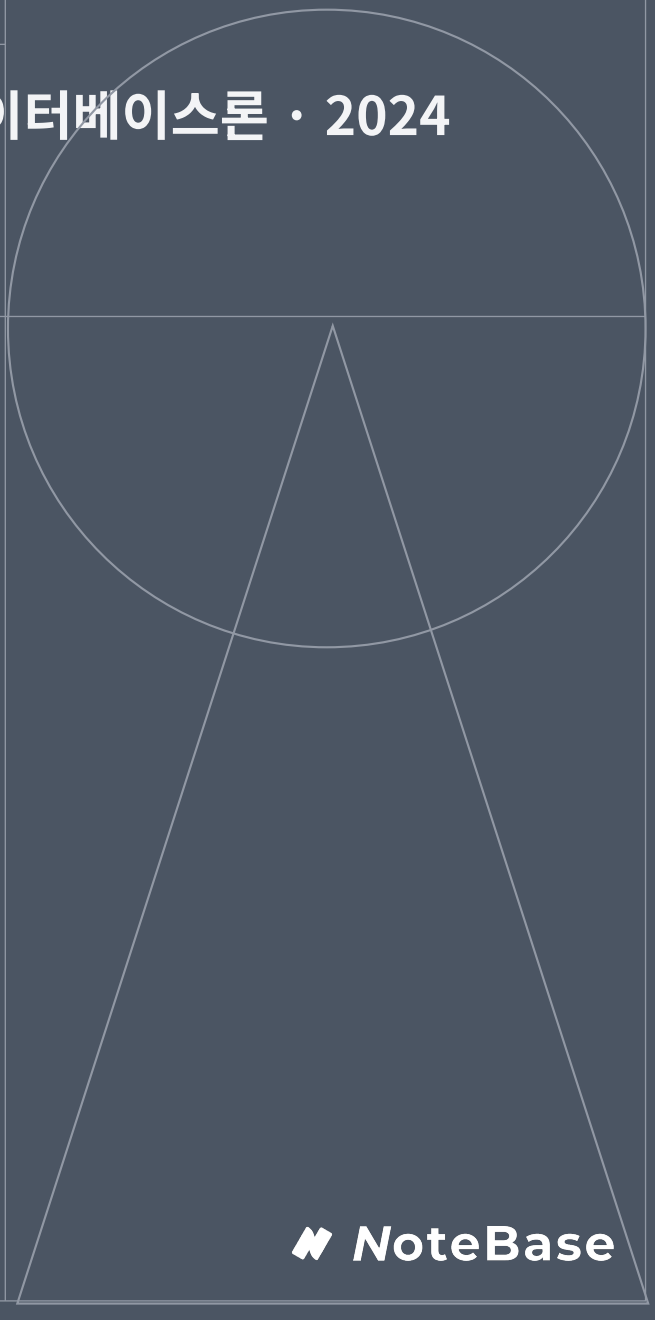
기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2024



NoteBase

기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2024

공무원/경찰/경력채용 · 데이터베이스론 · 2024

문 1. 데이터베이스의 특징에 관한 설명으로 가장 적절하지 않은 것은?

① 데이터의 주소나 위치를 기반으로 참조한다.

② 여러 사용자가 데이터에 동시 접근하여 사용할 수 있다.

③ 사용자의 데이터 요구에 실시간 응답이 가능하다.

④ 데이터에 대한 지속적 삽입, 삭제, 수정을 통해 데이터베이스의 상태가 계속적으로 변화한다.

정답 1번

간단한 해설

데이터베이스는 저장 주소가 아니라 **데이터의 내용과 조건으로 참조**한다. 동시 공유, 실시간 접근성, 계속적인 변화와 함께 구별해 두어야 한다.

선지별 해설

① X: 데이터베이스의 특징은 **내용에 의한 참조**이다. 사용자는 보통 물리적 저장 주소나 위치를 직접 지정하는 대신 데이터 값과 검색 조건으로 원하는 정보를 찾는다. DBMS 내부에서 주소나 포인터를 사용한다는 사실과 사용자 수준의 참조 방식을 구별해야 한다.

② O: 데이터베이스는 여러 사용자가 같은 데이터를 함께 이용하는 **동시 공유**를 지원한다. 충돌할 수 있는 읽기·쓰기 작업은 동시성 제어를 통해 조정하므로, 동시 공유가 무제한적인 동시 변경을 뜻하지는 않는다.

③ O: **실시간 접근성**은 사용자의 질의나 데이터 요구에 적시에 응답할 수 있도록 한다는 특징이다. 모든 질의가 항상 지연 없이 끝나거나 엄격한 시간 제한을 보장한다는 뜻은 아니며 실제 응답 시간은 질의·부하·시스템 성능에 따라 달라진다.

④ O: 데이터가 삽입·삭제·수정되면 저장된 내용이 달라져 **계속적인 변화**가 이루어진다. 데이터베이스는 현실의 변화가 반영된 최신 상태를 유지하도록 관리된다.

NoteBase

문 2. 데이터 언어 중 데이터 조작어(DML)에 관한 설명으로 가장 적절한 것은?

① 데이터 구조를 정의하는 명령을 사용한다.

② 데이터의 삽입, 삭제, 검색 및 수정에 관한 명령을 사용한다.

③ 데이터 무결성과 보안에 관한 명령을 사용한다.

④ 데이터 회복 및 병행제어를 위한 명령을 사용한다.

정답 2번

간단한 해설

DML은 구조가 아니라 그 안의 데이터를 다룬다. 데이터의 추가·제거·조회·수정과 테이블 구조 정의, 권한·트랜잭션 제어를 구별해야 한다.

개념 구분

데이터 조작어는 이미 정의된 테이블 등의 구조 안에서 **데이터를 삽입·삭제·검색·수정**하는 언어이다. 대표적으로 INSERT, DELETE, SELECT, UPDATE가 대응하므로 ②가 적절하다. 여기서는 검색을 DML에 포함하는 분류를 사용한다.

데이터 구조의 생성·변경·제거는 DDL의 역할이며 CREATE, ALTER, DROP 등이 대표적이다. 보안과 접근 권한은 GRANT·REVOKE 같은 제어 명령으로 다루고, 트랜잭션의 확정·취소는 COMMIT·ROLLBACK 등으로 다룬다. 무결성 제약은 테이블 정의 과정에서도 명세할 수 있으므로, 무결성에 관한 모든 명령을 하나의 분류에만 속한다고 단정하지 말고 데이터 자체의 조작과 구별한다.

NoteBase

문 3. ER 다이어그램의 표기법에 관한 설명으로 가장 적절한 것은?

① 약한 관계타입은 이중 마름모로 표기한다.

② 강한 관계타입은 이중 타원으로 표기한다.

③ 강한 개체는 이중 사각형으로 표기한다.

④ 약한 개체는 점선 타원으로 표기한다.

정답 1번

간단한 해설

전통적인 ER 표기에서 약한 개체의 **식별 관계는 이중 마름모**, 약한 개체는 이중 사각형이다. 이중 타원은 다중값 속성, 점선 타원은 유도 속성과 구별한다.

선지별 해설

① O: 전통적인 Chen 계열 ER 표기에서 약한 개체를 소유 개체와 연결하는 **식별 관계는 이중 마름모**로 표시한다. 문제의 약한 관계타입은 이 식별 관계를 뜻한다.

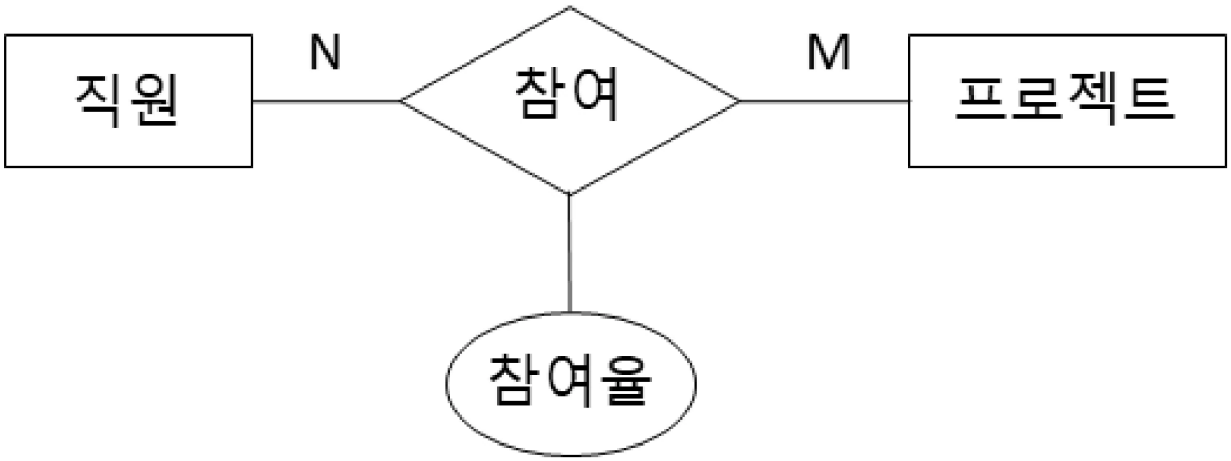
② X: 일반적인 관계타입은 **단일 마름모**로 표시한다. 이중 타원은 한 개체가 여러 값을 가질 수 있는 다중값 속성의 표기이다.

③ X: 강한 개체는 **단일 사각형**으로 표시한다. 이중 사각형은 자체 속성만으로 완전하게 식별되지 않아 소유 개체의 키에 의존하는 약한 개체를 나타낸다.

④ X: 약한 개체는 **이중 사각형**으로 표시한다. 점선 타원은 생년월일로 계산하는 나이처럼 다른 속성에서 계산할 수 있는 유도 속성을 뜻한다.

NoteBase

문 4. 다음의 ER 다이어그램을 관계 스키마로 사상하는 방법에 관한 설명으로 가장 적절한 것은?



① 직원 릴레이션의 기본키를 프로젝트 릴레이션에 포함시킨다.

② 프로젝트 릴레이션의 기본키를 직원 릴레이션에 포함시킨다.

③ 다대다(N:M) 관계 참여를 표현하는 새로운 릴레이션을 생성한다.

④ 속성 참여율을 프로젝트 개체를 표현한 릴레이션에 포함시킨다.

정답 3번

간단한 해설

N:M 관계는 양쪽 개체의 키를 연결하는 별도 릴레이션으로 표현한다. **참여율도 직원·프로젝트의 참여 관계에 속하므로** 그 릴레이션에 함께 둔다.

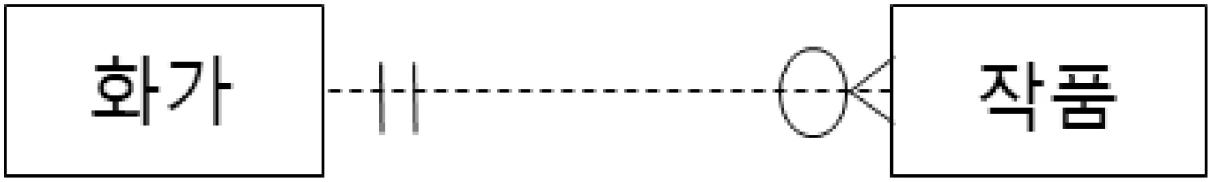
도식 해설

직원과 프로젝트 사이의 참여 관계는 N:M이다. 한 직원이 여러 프로젝트에 참여할 수 있고 한 프로젝트에도 여러 직원이 참여할 수 있으므로, **참여 관계를 별도 릴레이션으로 사상**한다. 예를 들어 참여(직원키, 프로젝트키, 참여율)를 두고 두 키를 각 개체 릴레이션의 외래키로 사용한다. 이 관계에서 직원과 프로젝트의 쌍이 참여 한 건을 식별하므로 두 키의 조합을 기본키로 구성할 수 있다.

참여율은 특정 직원이 특정 프로젝트에 참여하는 정도로, 직원이나 프로젝트 하나만의 속성이 아니라 **직원·프로젝트 쌍에 딸린 관계 속성**이다. 따라서 참여 릴레이션에 포함한다. 직원키를 프로젝트에 하나 넣거나 프로젝트키를 직원에 하나 넣는 방식만으로는 N:M의 여러 연결을 표현할 수 없고, 참여율을 프로젝트에만 넣어서도 직원별 차이를 담을 수 없다. 따라서 ③이 적절하다.

NoteBase

문 5. 다음은 화랑 데이터베이스에서 화가와 작품의 관계를 까마귀발(crow's foot)을 이용하여 ER 다이어그램으로 표현한 것이다. 이 다이어그램에 관한 설명으로 가장 적절한 것은? (단, 점선은 비식별 관계를 의미함)



① 하나의 작품을 여러 화가가 공동으로 제작할 수 있다.

② 한 화가는 최대 N개의 작품을 제작한다.

③ 해당 화가 정보가 없는 작품 정보도 입력될 수 있다.

④ 해당 작품 정보가 없는 화가 정보는 존재할 수 없다.

정답 2번

간단한 해설

작품 쪽의 **원과 까마귀발은 0개 이상 여러 작품**을 뜻하므로, 한 화가는 작품이 없어도 되고 여러 작품을 제작할 수도 있다. 반대로 작품 하나에는 화가 쪽 두 세로선에 따라 정확히 한 화가가 연결된다.

NoteBase

선지별 해설

관계선의 각 끝 기호는 반대편 개체 한 건에 대해 그쪽 개체가 몇 건 연결될 수 있는지를 나타낸다. 화가 쪽은 정확히 1, 작품 쪽은 0 이상 다수이다. 점선의 비식별 여부와 최소·최대 참여 수는 구별해서 읽는다.

① X: 화가 쪽의 두 세로선은 작품 하나에 연결되는 화가가 **정확히 1명**이라는 뜻이다. 따라서 이 모델은 여러 화가가 하나의 작품을 공동 제작하는 관계를 나타내지 않는다. 공동 제작까지 표현하려면 작품에 여러 화가가 연결될 수 있는 관계가 필요하다.

② O: 작품 쪽의 원과 까마귀발은 화가 한 명에 연결되는 작품이 **0개 이상 여러 개**일 수 있음을 뜻한다. 따라서 한 화가는 다수의 작품을 제작할 수 있다. 여기서 N은 관계의 '다수'를 나타내는 기호이며, 그림이 특정 숫자의 상한을 제시한다는 뜻은 아니다.

③ X: 작품에서 화가를 바라보면 화가 쪽 표시는 필수 1명이다. 따라서 이 관계 제약을 만족하는 작품 정보에는 반드시 대응하는 화가 정보가 있어야 한다. 점선이 비식별 관계라는 사실만으로 화가 연결을 생략할 수 있는 것은 아니다.

④ X: 화가에서 작품을 바라보면 작품 쪽 원이 최소 참여 수 0을 허용한다. 따라서 **아직 작품이 연결되지 않은 화가 정보도 존재할 수 있다.** 화가 한 명당 작품이 반드시 1개 이상이어야 한다면 원 대신 최소 1을 뜻하는 표시가 필요하다.

NoteBase

문 6. 관계 데이터 모델의 키(key)에 관한 설명으로 가장 적절하지 않은 것은?

① 대체키와 슈퍼키는 최소성을 만족한다.

② 슈퍼키는 유일성을 만족한다.

③ 기본키는 널(NULL)값을 가질 수 없다.

④ 외래키는 참조무결성 제약조건과 관계가 있다.

정답 1번

간단한 해설

슈퍼키는 유일성을 만족하지만 최소성까지 요구되지는 않는다. 대체키는 후보키의 일종이므로 유일성과 최소성을 모두 만족한다.

선지별 해설

① X: 대체키는 후보키 중 기본키로 선택되지 않은 것이므로 유일성과 최소성을 만족한다. 그러나 **슈퍼키에는 최소성이 요구되지 않는다**. 예를 들어 학번만으로 학생을 식별할 수 있다면 학번과 이름을 합친 집합도 슈퍼키이지만, 이름을 빼도 식별되므로 최소성은 없다. 최소성을 만족하는 슈퍼키가 후보키이다.

② O: 슈퍼키는 튜플을 유일하게 식별하는 속성 집합이므로 **유일성**을 만족한다. 식별에 불필요한 속성을 포함할 수 있다는 점에서 후보키보다 넓은 개념이다.

③ O: 기본키는 각 튜플의 식별자이며 **NULL을 허용하지 않는다**. 여러 속성을 결합한 기본키라면 그 구성 속성 모두 NULL이 아니어야 한다. 이는 개체무결성과 연결된다.

④ O: 외래키는 참조 대상의 키와 대응하여 **참조무결성**을 유지한다. NULL 허용 여부와 복합 외래키의 일치 규칙 등 제약조건에 따라 예외가 처리되며, 참조가 필요한 값은 참조 대상에 존재해야 한다.

따라서 슈퍼키도 최소성을 만족한다고 일반화한 ①이 가장 적절하지 않다.

NoteBase

문 7. 다음의 학생, 수강신청 테이블에서 학생⟕수강신청의 연산 결과에 관한 설명으로 가장 적절한 것은? (단, ⟕는 왼쪽 외부조인 연산자임)

학생

학번	이름	나이
1	김수사	28
2	이수사	29
3	오수사	30

수강신청

수강번호	학번	과목
S01	1	개인정보
S02	3	정보보안

① 카디널리티는 3이다.

② 카디널리티는 2이다.

③ 차수는 3이다.

④ 차수는 4이다.

정답 1번

간단한 해설

왼쪽 외부조인은 일치하지 않는 왼쪽 행도 보존한다. 수강신청이 없는 학번 2의 학생도 결과에 포함되므로, 학번 1·2·3에 대응하는 세 행이 남는다.

선지별 해설

① O: 공통 속성 학번을 기준으로 왼쪽의 학생을 보존한다. 학번 1은 S01, 학번 3은 S02와 각각 연결되고, 학번 2도 수강신청이 없다는 이유로 제거되지 않는다. 학번 2의 수강번호와 과목을 NULL로 채운 행을 포함하므로 **결과는 세 행**이고 카디널리티는 3이다.

② X: 두 행만 남는 것은 일치하는 학번 1과 3만 유지하는 내부조인의 결과이다. 왼쪽 외부조인은 학번 2의 학생 행도 남기므로 카디널리티는 2가 아니라 3이다.

③ X: 차수는 행 수가 아니라 **속성의 수**이다. 공통 학번을 하나로 합쳐 표현하는 외부조인 결과는 학번·이름·나이·수강번호·과목의 다섯 속성을 가지므로 차수는 3이 아니다.

④ X: 공통 학번을 하나로 표현하더라도 학생의 세 속성에 수강번호와 과목 두 속성이 추가되므로 차수는 5이다. SQL의 ON 방식처럼 양쪽 학번 열을 모두 유지하는 표기에서는 여섯 열이 되지만, 어느 경우에도 차수가 4가 되는 것은 아니다.

NoteBase

문 8. 다음에서 설명하는 트랜잭션의 특징으로 가장 적절한 것은?

수행 중인 트랜잭션(T1)이 수행 완료(commit)할 때까지, 다른 트랜잭션(T2)은 수행 중인 트랜잭션(T1)의 중간 연산결과에 접근이 불가하다.

① 원자성
② 격리성
③ 일관성
④ 지속성

정답 2번

간단한 해설

T2가 T1의 미확정 중간 결과를 보지 못하게 하는 것은 **격리성**이다. 연산을 전부 반영하거나 취소하는 원자성, 커밋 결과를 보존하는 지속성과 구별한다.

구성 요소별 해설

T1의 중간 연산결과: T1이 아직 커밋하지 않은 변경은 나중에 철회될 수도 있는 **미확정 결과**이다. 따라서 다른 트랜잭션이 이를 확정된 데이터처럼 이용하면 문제가 생길 수 있다.

T2의 접근 제한: T2가 T1의 미확정 결과를 보지 못하도록 두 트랜잭션의 실행을 분리하는 성질은 **격리성**이다. 이 설명은 특히 미확정 값을 읽는 부정 판독을 막는 상황을 보여 준다.

커밋 시점의 의미: 커밋은 T1의 결과가 확정되는 경계이다. 중간 결과를 읽지 못한다는 것이 T2의 모든 작업을 멈춘다는 뜻은 아니며, 구현에 따라 T2는 이전에 커밋된 버전을 읽을 수도 있다. 또한 이 조건 하나만으로 최고 격리 수준의 모든 성질이 보장된다고 판단해서는 안 된다.

다른 ACID 요소와 구별: 원자성은 연산의 전부 수행 또는 전부 취소, 일관성은 정해진 무결성 규칙을 만족하는 상태의 유지, 지속성은 커밋된 결과의 영구 보존에 관한 성질이다. 제시문처럼 **다른 트랜잭션의 중간 결과 접근을 제한**하는 성질은 격리성이므로 ②가 가장 적절하다.

NoteBase

문 9. <보기>의 SQL 명령어 중 데이터 정의어(DDL)로 옳은 것을 모두 고른 것은?

<보기>
㉠ CREATE ㉡ ALTER ㉢ DROP
㉣ SELECT ㉤ UPDATE ㉥ DELETE

① ㉠㉡㉢
② ㉠㉢㉤
③ ㉡㉣㉤
④ ㉡㉤㉥

정답 1번

간단한 해설

DDL은 데이터베이스 객체의 구조를 정의·변경·제거한다. **CREATE·ALTER·DROP은 객체 정의**를 다루고, SELECT·UPDATE·DELETE는 데이터 조회·조작을 다룬다.

보기별 해설

㉠ (O) CREATE는 테이블·뷰 등의 **데이터베이스 객체를 정의하여 생성**하는 명령이다. 데이터의 저장 구조를 정의하는 DDL에 해당한다.

㉡ (O) ALTER는 기존 테이블의 열이나 제약조건 등 **객체의 구조·정의를 변경**하는 명령이다. 저장된 행의 값을 수정하는 UPDATE와 구별한다.

㉢ (O) DROP은 테이블·뷰 등 **객체 자체의 정의를 제거**하는 명령이다. 행만 삭제하는 DELETE와 달리 DDL에 해당한다.

㉣ (X) SELECT는 저장된 **데이터를 조회**하는 명령으로, 객체 구조를 정의하는 DDL이 아니다. 분류 체계에 따라 DML의 조회 기능으로 보거나 DQL로 별도 구분한다.

㉤ (X) UPDATE는 기존 행의 **속성값을 수정**하는 데이터 조작 명령이다. 테이블 구조를 변경하는 것이 아니므로 DDL이 아니다.

㉥ (X) DELETE는 테이블에서 조건에 맞는 **행을 삭제**하는 데이터 조작 명령이며 테이블 정의는 남는다.

따라서 DDL은 ㉠·㉡·㉢이고 조합은 ①이다.

NoteBase

문 10. 다음의 직원과 부양가족 테이블에 대해 <보기>와 같은 SQL 질의문 실행시 질의 결과 테이블로 가장 적절한 것은?

직원

직원번호	이름	부서번호
100	김경장	D1
200	박경사	D2
300	이순경	D2
400	송경위	D1
500	윤경정	D3

부양가족

직원번호	이름
100	최바름
300	정열심
500	윤어른
500	손열정
100	김철수

<보기>

```
SELECT 이름
FROM 직원
WHERE (SELECT COUNT(*)
FROM 부양가족
WHERE 직원.직원번호=직원번호) >= 2;
```

① 이름: 김경장
이름: 이순경
이름: 윤경정

② 이름: 김경장
이름: 송경위

③ 이름: 박경사
이름: 송경위

④ 이름: 김경장
이름: 윤경정

정답 4번

간단한 해설

상관 서브쿼리가 직원별 부양가족 수를 세어 **2명 이상인 직원**을 고른다. 김경장과 윤경정이 각각 2명이므로 두 직원의 이름이 출력된다.

NoteBase

계산 과정

1. 상관 서브쿼리의 조건을 해석한다. 직원.직원번호는 바깥 직원 테이블의 현재 행을 가리키고, 안쪽의 한정되지 않은 직원번호는 부양가족 테이블의 열이다. 따라서 COUNT(*)는 **현재 직원번호와 일치하는 부양가족 행 수**를 센다.

2. 직원별 인원을 계산한다. 직원번호 100 김경장은 최바름·김철수로 2명, 200 박경사는 0명, 300 이순경은 정열심으로 1명, 400 송경위는 0명, 500 윤경정은 윤어른·손열정으로 2명이다.

3. **2명 이상**이라는 조건을 적용하면 김경장과 윤경정만 남는다. 바깥 SELECT가 출력하는 열은 직원의 이름이므로 결과는 이름 열의 두 행, 김경장과 윤경정이다. ORDER BY가 없으므로 출력 순서 자체는 보장되지 않는다.

따라서 결과의 두 이름이 일치하는 ④가 정답이다.

NoteBase

문 11. 다음은 학생 테이블로부터 '학생2' 뷰(view)를 생성하는 정의문의 일부이다. ㉠의 문장에 관한 설명으로 가장 적절하지 않은 것은? (단, 밑줄 속성은 기본키임)

테이블	학생(학생번호, 이름, 학과, 지도교수, 학년, 이수학점)
SQL문	CREATE VIEW 학생2(이름, 학과, 이수학점) (㉠);

① AS SELECT문을 적고 일반 검색문처럼 사용할 수 있다.

② DISTINCT 키워드가 포함되어 정의된 경우, '학생2' 뷰의 변경이 가능하다.

③ GROUP BY 절이 포함되어 정의된 경우, '학생2' 뷰의 변경이 불가하다.

④ 집계함수가 포함되어 정의된 경우, '학생2' 뷰의 변경이 불가하다.

정답 2번

간단한 해설

DISTINCT·GROUP BY·집계함수는 결과 행과 원본 행의 단순한 대응을 깨뜨린다. 별도 갱신 처리를 두지 않은 일반적인 뷰에서는 **DISTINCT를 포함한 경우에도 자동 갱신이 제한**되므로 ②가 틀리다.

선지별 해설

판단 범위: 여기서 뷰의 변경은 뷰를 통한 데이터의 INSERT·UPDATE·DELETE를 뜻하며, 뷰 정의문 자체를 교체하는 것을 뜻하지 않는다. 별도 트리거나 갱신 규칙을 작성하지 않은 **일반적인 자동 갱신 가능성**을 기준으로 판단한다.

① O: CREATE VIEW 학생2(이름, 학과, 이수학점) 뒤에는 **AS SELECT 이름, 학과, 이수학점 FROM 학생**과 같은 조회문을 적을 수 있다. 조회 결과의 세 열이 앞에서 지정한 뷰의 세 열에 대응한다.

② X: DISTINCT는 같은 값을 가진 여러 원본 행을 하나의 결과 행으로 합칠 수 있어 **결과 행과 갱신 대상 원본 행의 대응**이 모호해진다. 따라서 DISTINCT를 포함한 뷰는 일반적으로 자동 갱신할 수 없다.

③ O: GROUP BY로 정의한 뷰는 그룹별 결과를 나타내므로 결과 행을 변경할 때 원본의 어느 행들을 어떻게 바꿀지 자동으로 정하기 어렵다. 따라서 **그룹화된 뷰는 일반적인 자동 갱신 대상이 아니다.**

④ O: SUM·AVG·COUNT 등의 집계 결과는 여러 원본 값으로 계산된 값이다. 예를 들어 합계를 바꾸더라도 개별 학생의 이수학점을 어떻게 배분해 바꿀지는 정해지지 않으므로 **집계 뷰는 일반적으로 자동 갱신할 수 없다.**

NoteBase

문 12. 릴레이션 R의 함수 종속이 다음과 같을 때, R이 만족하는 정규형으로 옳은 것은? (단, 밑줄 속성은 기본키임)

R(학번, 과목번호, 이름, 성적)
함수종속: {학번, 과목번호} → 성적, 학번 → 이름

R

학번	과목번호	이름	성적
202401	A1	김정보	100
202401	A2	김정보	99
202402	A1	황정보	100

① 제1정규형
② 제2정규형
③ 제3정규형
④ BCNF

정답 1번

간단한 해설

이름은 복합키 전체가 아니라 **학번만으로 결정되는 부분 함수 종속**을 갖는다. 따라서 원자값을 저장하는 제1정규형은 만족하지만 제2정규형부터는 만족하지 않는다.

NoteBase

핵심 풀이

R의 기본키는 학번과 과목번호의 조합이다. 표에서 각 칸에는 하나의 학번·과목번호·이름·성적 값이 들어 있으며 반복 목록이나 집합을 저장하지 않으므로 제1정규형에 해당한다.

제2정규형이 되려면 비주요 속성이 후보키의 일부에만 종속되는 **부분 함수 종속이 없어야** 한다. 그런데 주어진 학번 → 이름에서 이름은 복합키 전체가 아니라 그 일부인 학번만으로 결정된다. 이는 제2정규형을 위반하는 부분 함수 종속이다. 반면 성적은 주어진 종속에 따라 학번과 과목번호의 조합으로 결정된다.

제2정규형을 만족하지 않으므로 그보다 강한 제3정규형과 BCNF도 만족하지 않는다. 표에서 같은 학번의 이름이 여러 과목 행에 반복되는 현상은 이 부분 종속으로 생기는 중복을 보여 준다. 다만 함수 종속의 근거는 몇 개의 예시 행에서 추측한 우연한 규칙이 아니라 문제에 명시된 종속이다.

따라서 제시된 정규형 중 R이 만족하는 것은 제1정규형이며 ①과 일치한다.

NoteBase

문 13. 다음의 함수 종속이 성립하는 릴레이션 R을 BCNF 정규형으로 분해한 결과로 가장 적절한 것은? (단, 밑줄 속성은 기본키임)

R(A, B, C, D, E)
함수종속: A→BCDE, B→ACDE, E→D

① R1(A, B, C), R2(D, E)

② R1(A, B, C, D), R2(D, E)

③ R1(B, C, D, E), R2(A, B)

④ R1(A, B, C, E), R2(D, E)

정답 4번

간단한 해설

BCNF에서는 모든 비자명한 함수 종속의 결정자가 슈퍼키여야 한다. **E는 전체를 식별하지 못하면서 D를 결정**하므로, D와 E를 별도 릴레이션으로 분리하고 나머지에는 연결 속성 E를 남긴다.

핵심 풀이

A와 B는 각각 모든 속성을 결정하므로 후보키이다. 반면 E가 결정할 수 있는 속성은 E 자신과 D뿐이므로 E는 원래 릴레이션의 슈퍼키가 아니다. 따라서 **슈퍼키가 아닌 E가 결정자인 종속** $E \rightarrow D$가 BCNF를 위반한다.

이 종속을 분리하여 R1(A, B, C, E)와 R2(D, E)를 만든다. R1에서는 A 또는 B가 전체 속성을 결정하고, R2에서는 E가 키이므로 두 릴레이션은 BCNF이다. 공통 속성 E가 R2 전체를 결정하므로 다시 조인해도 가짜 튜플이 생기지 않는 **무손실 분해**이다. 제시된 기본키 A와 E도 적절하므로 선택지는 ④이다.

①은 두 릴레이션에 공통 속성이 없어 원래 대응 관계를 복원할 수 없다.

②는 공통 속성이 D이지만 D가 어느 쪽 릴레이션도 결정하지 않아 무손실성이 보장되지 않는다.

③은 R1(B, C, D, E)에 슈퍼키가 아닌 E에 의한 종속이 그대로 남으므로 BCNF가 아니다.

NoteBase

문 14. B+-트리에 해당하는 사항으로 가장 적절한 것은?

① 각 노드가 최소한 2/3 이상 차 있어야 한다.

② 리프 노드의 오버플로우 발생시 두 개의 노드로 분할하고 중간 키값은 리프 레벨에 남겨두지 않고 분할된 노드의 부모 노드로 올려 보내 삽입한다.

③ 리프가 아닌 노드로 구성되는 인덱스 세트와 리프 노드로만 구성된 순차 세트로 구성된다.

④ 내부 노드는 서브트리에 관한 포인터, 탐색키 값, 해당 탐색키 값을 갖는 레코드에 관한 포인터 등으로 구성된다.

정답 3번

간단한 해설

B+ 트리는 비리프의 인덱스 세트로 탐색하고 리프의 순차 세트에서 데이터 항목에 도달한다. **리프가 모든 데이터 항목의 키를 유지하며 순차 연결**된다는 점이 핵심이다.

선지별 해설

① (X) 일반적인 B+ 트리는 루트의 예외와 정수 단위의 올림 규칙을 고려하여 비루트 노드가 대략 **절반 이상 차도록** 유지한다. 모든 노드가 최소 2/3 이상 차야 한다는 조건은 B+ 트리의 기본 조건이 아니다.

② (X) 리프가 넘쳐 분할될 때는 경계 키를 부모의 탐색 기준으로 **복사하여 올리고 리프에도 남겨 둔다**. 리프에는 실제 데이터 항목을 찾는 데 필요한 키가 유지되어야 한다. 내부 노드 분할 때 키를 위로 이동시키는 처리와 혼동하면 안 된다.

③ (O) B+ 트리는 탐색 경로를 제공하는 비리프의 **인덱스 세트**와 데이터 항목을 순서대로 보유하는 리프의 **순차 세트**로 구분된다. 리프들을 연결하여 순차 접근과 범위 검색을 효율적으로 수행한다.

④ (X) B+ 트리의 내부 노드는 탐색 키와 자식 서브트리를 가리키는 포인터로 경로를 안내한다. 실제 레코드를 가리키는 데이터 포인터는 일반적으로 **리프 노드에 저장**하며, 내부 노드마다 레코드 포인터를 함께 두는 구조로 설명하지 않는다.

NoteBase

문 15. 질의 최적화에 관한 설명으로 가장 적절한 것은?

① 조인(⋈) 연산을 카티션 프로덕트(×)와 셀렉트(σ)로 대체한다.

② 셀렉트(σ) 연산들을 질의 트리의 위로 이동하고 보다 제한적인 (restrictive) 셀렉트 연산을 나중에 적용한다.

③ 일련의 프로젝트(π) 연산들이 연쇄적으로 나타나면 마지막 연산을 제외하고 모두 무시할 수 있다.

④ 합집합(∪), 교집합(∩), 차집합(−)은 교환 법칙이 성립한다.

정답 3번

간단한 해설

단순 투영이 연속되면 최종 출력 속성을 정하는 **가장 바깥쪽 투영**만 남길 수 있다. 앞선 투영이 그 속성을 포함하는 유효한 연쇄라는 조건에서 ③이 옳다.

선지별 해설

① X: 일반적인 휴리스틱 최적화에서는 **카티션 프로덕트 뒤의 조인 조건 선택을 조인으로 결합**하여 불필요한 중간 결과를 피한다. 세타 조인을 카티션 프로덕트와 선택으로 표현하는 것은 대수적으로 가능하지만, 이를 실제 수행 순서로 풀어 큰 곱집합을 만드는 것이 보편적인 최적화 원칙은 아니다.

② X: 선택 연산은 가능한 한 **질의 트리의 아래쪽, 원본 릴레이션 가까이**로 내려 먼저 수행한다. 다른 조건이 같다면 더 많은 행을 걸러내는 제한적인 선택을 일찍 적용하여 후속 연산의 입력 크기를 줄이는 것이 기본 원칙이다.

③ O: 유효한 단순 투영 연쇄에서는 최종적으로 필요한 속성만 남기는 **마지막 실행 투영, 즉 가장 바깥쪽 투영**만 유지할 수 있다. 예를 들어 $A \subseteq B$이면 $\pi_A(\pi_B(R)) = \pi_A(R)$이다. 중간에 다른 연산이 끼거나 필요한 속성이 이미 제거된 잘못된 식까지 무조건 적용하는 규칙은 아니다.

④ X: 합집합과 교집합에는 교환 법칙이 성립하지만 **차집합에는 일반적으로 성립하지 않는다.** 예를 들어 R={a}, S={b}이면 R−S={a}, S−R={b}로 서로 다르다.

NoteBase

문 16. 로킹(locking) 기법에 관한 설명으로 가장 적절하지 않은 것은?

① 단순한 로킹 규약의 적용만으로는 모순성이 발생할 수 있다.

② 2단계 로킹 규약(2PL)을 적용하면 직렬가능성이 보장된다.

③ 2단계 로킹 규약은 확장 단계(growing phase)와 축소 단계(shrinking phase)로 구성된다.

④ 2단계 로킹 규약에서는 교착 상태(deadlock)가 발생하지 않는다.

정답 4번

간단한 해설

2PL은 직렬가능성을 보장하지만 교착 상태까지 막지는 않는다. 잠금을 획득하는 확장 단계와 해제하는 축소 단계를 분리해도, 서로의 잠금을 기다리는 순환 대기는 가능하다.

선지별 해설

① O: 데이터를 사용하기 전에 잠그고 사용 후 해제한다는 규칙만으로는 전체 트랜잭션의 실행 순서가 직렬가능해지는 것이 보장되지 않는다. 잠금을 일찍 풀고 다른 잠금을 다시 얻는 사이에 다른 트랜잭션이 끼어들면 일관되지 않은 결과가 생길 수 있다.

② O: 모든 트랜잭션이 적절한 잠금 모드로 2PL을 따르면 **충돌 직렬가능성**이 보장된다. 잠금의 획득과 해제 순서를 제한하여 충돌 연산들의 순서가 어떤 직렬 실행과 같도록 만들기 때문이다.

③ O: 확장 단계에서는 잠금을 획득할 수 있지만 해제하지 않고, 축소 단계에서는 잠금을 해제할 수 있지만 새로 획득하지 않는다. **한 번 잠금을 해제한 뒤에는 새 잠금을 얻을 수 없다**는 것이 기본 규칙이다.

④ X: 기본 2PL도 교착 상태가 발생할 수 있다. 예를 들어 한 트랜잭션이 A의 배타 잠금을 보유한 채 B를 기다리고, 다른 트랜잭션이 B의 배타 잠금을 보유한 채 A를 기다리면 순환 대기가 된다. 둘 다 확장 단계일 수 있으므로 2PL 위반 없이 발생하며, 별도의 교착 예방·탐지·해결 대책이 필요하다.

NoteBase

문 17. 다음의 R, S 테이블에서 R-S 연산 결과의 카디널리티는? (단, -는 차집합 연산자임)

R

학번	이름	학과
1	김순경	정보보호
2	이순경	사이버수사
3	오순경	사이버수사

S

학번	이름	학과
1	김순경	정보보호
2	이순경	사이버수사
5	황순경	정보보호

① 0

② 1

③ 2

④ 3

정답 2번

간단한 해설

R에서 S와 공통인 두 튜플을 제외하면 **(3, 오순경, 사이버수사)**만 남는다. 따라서 결과의 카디널리티는 1이다.

계산 과정

1. 차집합 R−S는 **R에는 있고 S에는 없는 튜플**만 남긴다. 같은 튜플인지는 학번·이름·학과의 전체 값으로 비교한다.

2. R의 (1, 김순경, 정보보호)와 (2, 이순경, 사이버수사)는 S에도 있으므로 제외한다. (3, 오순경, 사이버수사)는 S에 없으므로 남는다. S에만 있는 (5, 황순경, 정보보호)는 R−S에 추가하지 않는다.

3. 결과는 {(3, 오순경, 사이버수사)}이다. 카디널리티는 열 개수가 아니라 **튜플 개수**이므로 1이다.

따라서 카디널리티 1을 나타내는 ②가 정답이다.

NoteBase

문 18. 로그를 이용한 회복 기법에서 장애 발생시 로그의 내용이 다음과 같을 경우 회복 작업에 관한 설명으로 가장 적절하지 않은 것은? (단, Ti는 트랜잭션 번호를 의미함)

<T1, start>
<T1, commit>
<T2, start>
<checkpoint {T2}>
<T3, start>
<T3, commit>
<T4, start>

① T1은 즉시 갱신시 아무런 조치도 필요하지 않다.

② T2는 즉시 갱신시 undo를 해야 한다.

③ T3은 지연 갱신시 redo를 해야 한다.

④ T4는 지연 갱신시 undo를 해야 한다.

정답 4번

간단한 해설

지연 갱신에서는 **커밋 전 변경을 데이터베이스에 반영하지 않아 UNDO가 필요 없다.** T4는 시작 기록만 있는 미완료 트랜잭션이므로 REDO 대상도 아니다.

NoteBase

선지별 해설

checkpoint {T2}는 검사점 당시 T2가 실행 중이었음을 나타낸다. 이후 로그를 보면 T3는 완료되었고 T2와 T4는 완료되지 않았다. 즉시 갱신과 지연 갱신은 미완료 변경이 데이터베이스에 반영될 수 있는지에서 다르다.

① O: T1의 commit 기록은 checkpoint보다 앞에 있다. 문제의 검사점 회복 모형에서 검사점 이전에 완료된 변경은 이미 안전하게 반영되어 있으므로 즉시 갱신에서도 T1에 대한 REDO나 UNDO가 필요 없다.

② O: T2는 검사점의 실행 중 트랜잭션 목록에 있으며, 장애 전까지 commit 기록이 없다. 즉시 갱신은 커밋 전 변경도 데이터베이스에 반영될 수 있으므로, **미완료 T2의 변경을 UNDO**하여 취소해야 한다.

③ O: T3는 검사점 이후 시작하여 장애 전에 commit되었다. 지연 갱신에서도 완료된 변경이 디스크에 모두 반영되지 않았을 수 있으므로, 로그의 새 값을 이용하여 **REDO**한다. 커밋된 작업의 결과를 유지하기 위한 회복이다.

④ X: T4는 start만 있고 commit 기록이 없는 미완료 트랜잭션이다. 지연 갱신은 미완료 트랜잭션의 변경을 데이터베이스에 반영하지 않으므로 **UNDO할 변경이 없다.** T4는 REDO 대상도 아니며, 그 변경에 대한 데이터베이스 회복 연산은 수행하지 않는다. 즉시 갱신의 미완료 트랜잭션 처리와 혼동하면 안 된다.

NoteBase

문 19. SQL에서 권한 부여 명령어에 관한 설명으로 가장 적절하지 않은 것은?

① GRANT 명령을 이용하여 권한을 부여하고, REVOKE 명령을 이용하여 권한을 취소한다.

② 사용자 계정 PUBLIC을 지정하면 해당 권한을 데이터베이스의 모든 사용자에게 할당한다.

③ GRANT 명령에서 WITH GRANT OPTION을 지정하면 부여한 권한을 다른 사용자에게 다시 부여할 수 있다.

④ SELECT, UPDATE, INSERT 권한의 경우 특정 테이블의 특정 속성을 명시해 부여할 수 있다.

정답 4번

출제 포인트

SQL 권한의 부여와 회수, 권한을 받는 대상, 재부여 가능 여부 및 열 단위 권한의 지원 범위를 구분하는 문제이다.

핵심 개념

권한은 사용자가 데이터베이스 객체에 수행할 수 있는 작업의 허용 범위이다. GRANT는 권한을 부여하고 REVOKE는 부여된 권한을 회수한다.
PUBLIC은 모든 사용자를 가리키는 특별한 권한 부여 대상이다. WITH GRANT OPTION은 권한을 받은 사용자가 그 권한을 다른 사용자에게 다시 부여할 수 있게 하는 옵션이다.
속성은 테이블의 열을 뜻한다. 열 단위 권한을 부여할 수 있는 작업의 종류는 DBMS에 따라 다르다.

NoteBase

선지별 해설

이 문항에서 O는 적절한 설명, X는 가장 적절하지 않아 선택할 설명을 뜻한다.
① O. GRANT는 사용자에게 권한을 부여하고, REVOKE는 이미 부여한 권한을 회수한다. 두 명령의 기능을 올바르게 구분한 설명이다.
② O. 권한 부여 대상을 PUBLIC으로 지정하면 모든 사용자에게 해당 권한이 부여된다. 다만 PUBLIC은 일반 사용자 한 명의 계정이 아니라 모든 사용자를 나타내는 특별한 대상이라는 점에서 “사용자 계정”이라는 표현은 엄밀하지 않다.
③ O. WITH GRANT OPTION을 붙여 권한을 부여하면 그 권한을 받은 사용자가 다른 사용자에게 같은 권한을 다시 부여할 수 있다. 재부여할 수 있는 범위는 자신이 해당 옵션과 함께 받은 권한의 범위이다.
④ X. UPDATE와 INSERT를 특정 열에 한정하여 부여할 수 있다는 부분은 타당하다. 그러나 SELECT까지 같은 방식으로 부여할 수 있다고 DBMS 구분 없이 묶어 설명한 부분은 일반화에 한계가 있다. Oracle에서는 열을 명시한 권한 부여가 INSERT, UPDATE, REFERENCES에 허용되고 SELECT에는 허용되지 않는다. 이 방식에서는 필요한 열만 포함한 뷰에 SELECT 권한을 부여하여 조회 범위를 제한할 수 있다. 따라서 “SELECT, UPDATE, INSERT 모두 특정 속성을 명시해 부여할 수 있다”는 표현은 “UPDATE와 INSERT는 열을 지정해 부여할 수 있으며, SELECT의 열 단위 권한 부여는 DBMS의 지원 여부에 따라 달라진다”로 고친다. PostgreSQL처럼 SELECT의 열 단위 부여를 지원하는 DBMS도 있으므로, SELECT는 언제나 열 단위로 부여할 수 없다고 단정하지 않는다.

오답 포인트

④는 열 단위로 부여할 수 있는 UPDATE와 INSERT에 SELECT를 함께 묶었다. 테이블 단위 권한과 열 단위 권한을 구분하면서, 특정 DBMS의 제한을 모든 SQL의 보편 규칙으로 확대하지 않는 것이 핵심이다. ②의 PUBLIC도 일반 계정이라는 표현보다 모든 사용자에게 권한을 부여한다는 기능에 초점을 맞춘다.

다시 나오면 이렇게 판단

권한 문제는 명령의 기능, 권한을 받는 대상, 재부여 옵션, 권한의 적용 범위를 차례로 구분한다. 열 단위 권한이 나오면 해당 작업이 그 DBMS에서 지원되는지 판단한다. 특정 DBMS가 제시되지 않은 문항에서는 지원 범위를 일괄적으로 단정한 설명의 한계를 함께 살핀다.

암기 포인트

GRANT는 부여, REVOKE는 회수, PUBLIC은 모든 사용자, WITH GRANT OPTION은 받은 권한의 재부여이다. 열 단위 권한의 지원 범위는 DBMS마다 다르다.

NoteBase

문 20. NoSQL의 특징에 관한 설명으로 가장 적절하지 않은 것은?

① 데이터 양이 증가함에 따라 저장 공간과 노드를 추가할 수 있는 수평적 확장성(scalability)을 제공한다.

② 반구조적(semi-structured)이며 자기기술적(self-describing)인 데이터를 허용함으로써 스키마를 요구하지 않는다.

③ CAP(Consistency, Availability, Partition tolerance) 이론의 일치성은 관계 데이터베이스 수준으로 강하게 보장하고 가용성과 분할 허용성은 약하게 보장한다.

④ SQL과 같은 강력한 질의어를 요구하지 않는다.

정답 3번

간단한 해설

NoSQL은 분산 확장성과 유연한 데이터 모델을 중시하지만, 모든 제품이 같은 일치성 정책을 쓰는 것은 아니다. CAP는 **네트워크 분할 상황에서의 일치성과 가용성 간 제약**을 설명한다.

선지별 해설

① O: 많은 NoSQL 시스템은 데이터를 여러 노드에 분산하여 저장하고 처리한다. 서버 한 대의 성능만 높이는 수직 확장과 달리, 노드를 추가하는 수평 확장으로 데이터 증가에 대응하도록 설계된다.

② O: 문서형 등에서 말하는 스키마리스는 모든 레코드에 동일한 고정 스키마를 미리 강제하지 않는다는 뜻이다. 필드 이름과 값을 함께 담는 자기기술적 데이터로 서로 다른 구조를 유연하게 저장할 수 있다. **데이터 구조가 전혀 없거나 스키마 검증을 사용할 수 없다는 뜻은 아니다.**

③ X: NoSQL을 강한 일치성만 보장하고 가용성과 분할 허용성은 약하게 보장하는 방식으로 일괄 설명할 수 없다. CAP의 핵심은 **네트워크 분할 중 강한 일치성과 모든 요청에 대한 가용성을 동시에 보장할 수 없다는 것**이다. 시스템은 설계와 설정에 따라 분할 상황에서 일치성을 우선하거나 가용성을 우선한다. NoSQL이 모두 약한 일치성만 제공한다는 반대 일반화도 옳지 않다.

④ O: NoSQL은 SQL을 필수 공통 인터페이스로 요구하지 않으며, 키 기반 접근이나 문서 질의 등 모델별 API와 질의 방식을 사용할 수 있다. 이는 SQL과 유사한 언어나 풍부한 질의 기능을 제공하는 NoSQL 제품이 없다는 뜻은 아니다.

NoteBase