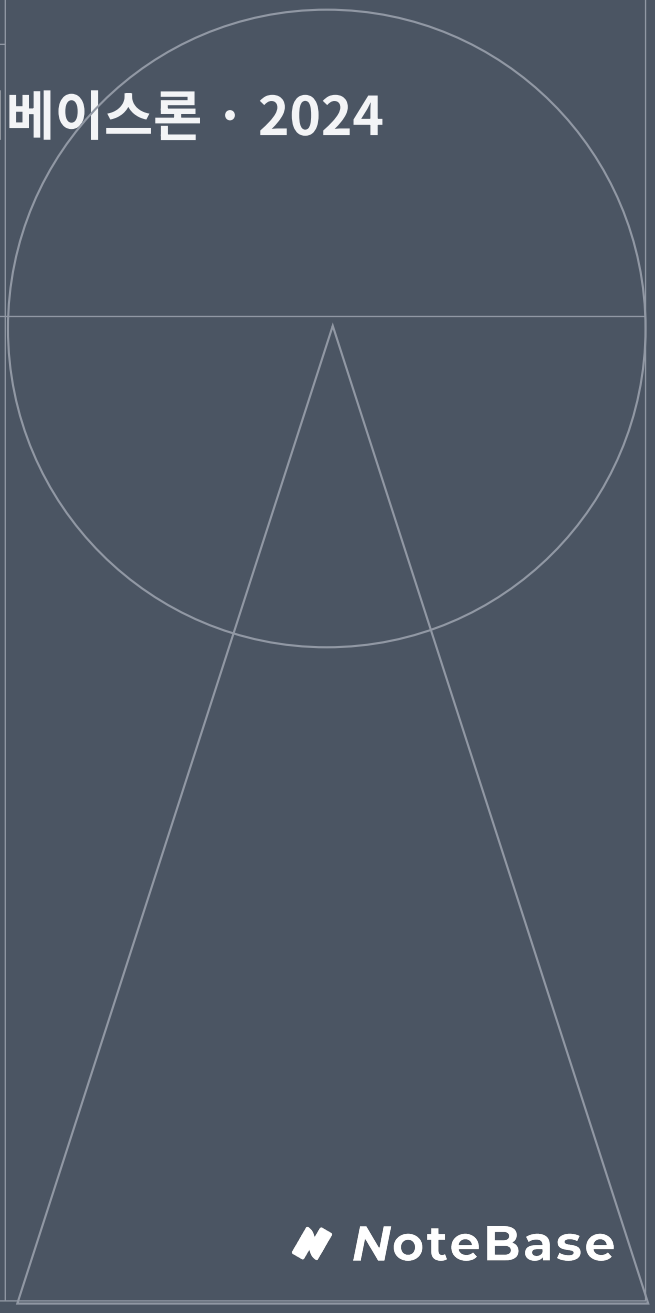
기출OX 해설집

공무원/국가직/9급 · 데이터베이스론 · 2024



NoteBase

기출OX 해설집

공무원/국가직/9급 · 데이터베이스론 · 2024

공무원/국가직/9급 · 데이터베이스론 · 2024

문 1. 데이터베이스 및 데이터베이스 관리시스템(DBMS)의 사용자와 관련된 설명으로 옳지 않은 것은?

① 일반 사용자(end user)는 응용 프로그램을 통해 데이터베이스에 접속하여 원하는 정보를 얻는 사용자를 말한다.

② 데이터베이스 관리자(DBA)는 일반 사용자가 편리하게 데이터베이스에 접근할 수 있도록 인터페이스를 개발한다.

③ 일반 사용자는 데이터의 삽입, 삭제, 갱신 작업을 수행하기도 한다.

④ 응용 프로그래머(application programmer)는 프로그래밍 언어를 이용하여 데이터베이스 응용 프로그램을 개발하는 사용자이다.

정답 2번

간단한 해설

일반 사용자가 데이터베이스를 편리하게 이용하도록 인터페이스를 개발하는 것은 **응용 프로그래머**의 역할이다. 데이터베이스 관리자(DBA)는 데이터베이스의 구조, 접근 권한, 백업·복구 등을 관리한다.

선지별 해설

① O: 일반 사용자는 데이터베이스에 저장된 정보를 업무에 이용하는 사용자이다. 응용 프로그램이 제공하는 **화면이나 메뉴를 통해** 조회를 요청하고 필요한 정보를 얻을 수 있다.

② X: 일반 사용자를 위한 인터페이스 개발은 **응용 프로그래머의 역할**이다. DBA의 주된 역할은 데이터베이스 구조 정의, 사용자 접근 권한 관리, 백업·복구, 성능 관리 등 데이터베이스의 운영과 통제이다. 선지의 '데이터베이스 관리자(DBA)'를 '응용 프로그래머'로 고쳐야 한다.

③ O: 일반 사용자의 작업은 조회에만 한정되지 않는다. **부여받은 권한 범위에서** 응용 프로그램을 이용하여 데이터를 삽입·삭제·갱신할 수도 있다. 예를 들어 고객 관리 프로그램에서 새 고객을 등록하거나 기존 고객의 주소를 바꾸는 작업이 이에 해당한다.

④ O: 응용 프로그래머는 **프로그래밍 언어로 데이터베이스 응용 프로그램을 개발**한다. 사용자 요청을 받아 DBMS에 조회나 변경을 요청하고 그 결과를 보여 주는 기능을 구현한다.

NoteBase

문 2. 데이터베이스 트랜잭션이 올바르게 수행된다는 것을 보장하기 위해 지원해야 하는 트랜잭션 성질에 대한 설명으로 옳지 않은 것은?

① 각 트랜잭션의 모든 연산이 정상적으로 수행 완료되거나, 아니면 어떠한 연산도 수행되지 않은 원래 상태가 되어야 한다.

② 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환한다.

③ 트랜잭션이 성공적으로 끝난 뒤에는, 시스템에 오류가 발생하더라도 트랜잭션에 의해 변경된 데이터베이스 내용은 계속 유지되어야 한다.

④ 트랜잭션이 실행되는 동안의 값은 다른 트랜잭션이 접근할 수 있어야 한다.

정답 4번

간단한 해설

④의 '접근할 수 있어야 한다'는 격리성에 어긋난다. **격리성**은 실행 중인 트랜잭션의 중간 결과에 다른 트랜잭션이 접근하지 못하도록 하여 서로 간섭하지 않게 하는 성질이다.

선지별 해설

① O: 원자성(Atomicity)에 대한 설명이다. 트랜잭션에 포함된 연산은 **모두 반영되거나 전혀 반영되지 않아야 한다**. 예를 들어 계좌 이체를 하나의 트랜잭션으로 처리할 때, 출금 후 입금에 실패하면 출금도 취소하여 이체 전 상태로 되돌려야 한다.

② O: 일관성(Consistency)에 대한 설명이다. 트랜잭션은 일관성 있는 데이터베이스 상태에서 시작하여 성공적으로 완료된 뒤에도 **무결성 제약조건을 만족하는 상태**를 유지해야 한다. 데이터 값이 바뀌지 않는다는 뜻이 아니라, 값이 바뀌어도 데이터베이스가 지켜야 할 규칙이 유지된다는 뜻이다.

③ O: 지속성(Durability)에 대한 설명이다. 트랜잭션이 성공적으로 완료되어 **커밋된 결과는 시스템 장애 후에도 유지**되어야 한다. 실패한 트랜잭션의 일부 변경을 취소하는 원자성과 달리, 지속성은 이미 확정된 변경 결과를 보존하는 성질이다.

④ X: 격리성(Isolation)은 여러 트랜잭션이 동시에 실행되어도 서로의 중간 처리 결과에 간섭받지 않고 독립적으로 수행되는 것처럼 보이도록 하는 성질이다. 따라서 '접근할 수 있어야 한다'는 '**중간 결과에 접근할 수 없어야 한다**'로 고쳐야 한다. 아직 완료되지 않은 변경 결과를 다른 트랜잭션이 이용하면, 그 변경이 나중에 취소될 때 잘못된 결과를 바탕으로 처리하게 될 수 있다.

NoteBase

문 3. 관계형 데이터베이스의 릴레이션에 대한 특성으로 옳지 않은 것은?

① 하나의 릴레이션에는 동일한 투플(tuple)이 존재할 수 없다.

② 릴레이션의 투플에 저장되는 값의 집합을 도메인(domain)이라 한다.

③ 하나의 릴레이션에서 투플의 순서는 의미가 없다.

④ 하나의 릴레이션에는 동일한 이름의 속성(attribute)을 가질 수 없다.

정답 2번

간단한 해설

도메인은 **하나의 속성이 가질 수 있는 원자값들의 집합**이다. 여러 속성의 값으로 이루어진 한 행이 투플이므로, 투플에 저장된 값의 집합을 도메인이라고 할 수 없다.

선지별 해설

① O: 릴레이션은 투플의 집합이다. 집합에서는 같은 원소를 중복하여 세지 않으므로, **모든 속성값이 같은 투플**이 하나의 릴레이션에 중복하여 존재할 수 없다.

② X: 도메인은 투플 전체에 저장된 값의 집합이 아니라, **하나의 속성이 가질 수 있는 원자값들의 집합**이다.
투플은 각 속성에 대응하는 값들로 구성된 한 행이다. 예를 들어 '학적상태' 속성이 '재학', '휴학', '졸업' 중 하나를 갖도록 정의되었다면, 이 세 값의 집합이 도메인이다. 현재 저장된 투플에 '휴학'이 없어도 도메인에는 포함된다.

③ O: 릴레이션은 순서가 없는 집합이므로 **투플의 나열 순서**는 의미가 없다. 행의 표시 순서를 바꾸어도 포함된 투플이 같으면 같은 릴레이션이다.

④ O: 하나의 릴레이션에서는 속성 이름으로 각 열을 구별하므로 **속성 이름이 서로 달라야 한다**. 같은 이름을 두 속성에 사용하면 그 이름이 어느 속성을 가리키는지 구별할 수 없다.

NoteBase

문 4. 데이터베이스 및 DBMS에 대한 설명으로 옳은 것은?

① 개념적 데이터 모델(conceptual data model)은 데이터가 컴퓨터의 저장 장치에 어떻게 저장되는가에 대한 세부 사항을 명시하는 개념을 제공한다.

② 데이터 정의어(DDL)를 이용하면 개념 스키마를 정의할 수 있다.

③ 데이터 조작어(DML)는 무결성 제약 조건을 정의하는 명령어를 포함한다.

④ DBMS는 데이터베이스 관리자나 일반 사용자가 아닌 응용 프로그래머를 위한 모듈은 제공하지 않는다.

정답 2번

간단한 해설

데이터 정의어(DDL)는 데이터베이스의 구조와 제약 조건을 기술하므로, 데이터베이스 전체의 논리적 구조를 나타내는 **개념 스키마를 정의**할 수 있다.

선지별 해설

① X: 개념적 데이터 모델은 사용자 관점에서 개체, 속성, 관계 등을 표현하며 저장 장치의 세부 구현은 다루지 않는다. 레코드의 저장 형식이나 접근 경로처럼 데이터가 실제로 저장되는 방식을 명시하는 것은 **물리적 데이터 모델**이다.

② O: 3단계 스키마 구조에서 개념 스키마는 데이터베이스 전체의 논리적 구조를 나타낸다. **데이터 정의어(DDL)**로 데이터의 구조, 자료형, 관계, 제약 조건 등을 정의할 수 있다. 예를 들어 CREATE TABLE로 학생 테이블의 학번·이름 열과 기본키를 선언하는 것은 스키마를 정의하는 작업이다.

③ X: 데이터 조작어(DML)는 이미 정의된 데이터베이스의 데이터를 검색·삽입·수정·삭제하는 데 사용한다. 무결성 제약 조건을 정의하는 명령어는 **DML이 아니라 DDL**에 속한다. 예를 들어 기본키나 외래키 제약 조건은 CREATE TABLE이나 ALTER TABLE로 정의한다.

④ X: DBMS는 데이터베이스 관리자와 일반 사용자뿐 아니라 **응용 프로그래머를 위한 인터페이스와 처리 기능도 제공**한다. 응용 프로그래머는 데이터베이스 접근 API나 프로그램 안에 삽입한 SQL 등을 이용해 데이터 처리를 요청하고, DBMS는 그 요청을 처리한다. 따라서 응용 프로그래머를 위한 모듈을 제공하지 않는다는 설명은 잘못이다.

NoteBase

문 5. 뷰에 대한 설명으로 옳지 않은 것은?

① 뷰를 생성하는 질의가 수행되면 결과 데이터가 물리적 저장 장치에 저장된다.

② 하나의 테이블로 여러 개의 상이한 뷰를 정의할 수 있다.

③ 뷰는 데이터의 접근을 제어함으로써 보안을 제공할 수 있다.

④ DROP 문을 이용해 뷰를 삭제할 수 있다.

정답 1번

간단한 해설

일반적인 뷰는 조회 결과를 물리적으로 저장하지 않고 **뷰를 정의한 질의문**을 저장한다. 결과 데이터를 별도로 저장하는 것은 **구체화된 뷰**이다.

선지별 해설

① X: 일반적인 뷰는 기본 테이블을 바탕으로 정의한 가상 테이블이다. 뷰를 생성하면 **결과 데이터가 아니라 뷰의 정의**가 저장되며, 뷰를 조회할 때 그 정의에 따라 기본 테이블에서 데이터를 가져온다. 결과를 물리적으로 저장하는 구체화된 뷰와 구별해야 한다.

② O: 하나의 테이블에서도 선택하는 열이나 행의 조건을 달리하여 **여러 뷰를 정의**할 수 있다. 예를 들어 같은 직원 테이블에서 이름과 부서만 보여 주는 뷰와 특정 부서 직원만 보여 주는 뷰를 각각 만들 수 있다.

③ O: 뷰는 사용자에게 필요한 열과 행만 노출하여 데이터 접근을 제한할 수 있다. 예를 들어 급여 열을 제외한 직원 뷰를 만들고, 사용자에게 기본 테이블 대신 **그 뷰에만 조회 권한**을 부여하면 급여를 직접 조회하지 못하도록 할 수 있다.

④ O: 뷰는 **DROP VIEW 문**으로 삭제할 수 있다. 이는 뷰의 정의를 제거하는 것이며, 그 뷰가 참조하던 기본 테이블과 그 안의 데이터 자체를 삭제하는 것은 아니다.

NoteBase

문 6. 다음은 각 트랜잭션에서 구매된 물건들을 나타내는 장바구니 분석(market basket analysis) 데이터이다. 연관 규칙 기법으로 이 데이터를 분석할 때 옳지 않은 것은?

Transaction ID	Items
1	a, b, c
2	a, c
3	a, b, c, d
4	c, d
5	a, b, d

① 연관 규칙 {a, c} ⇒ {b}의 지지도(support)는 40%이다.

② 연관 규칙 {c} ⇒ {d}의 지지도(support)는 50%이다.

③ 연관 규칙 {c} ⇒ {a, b}의 신뢰도(confidence)는 50%이다.

④ 연관 규칙 {a} ⇒ {b}의 신뢰도(confidence)는 75%이다.

정답 2번

간단한 해설

지지도는 **전체 트랜잭션** 중 규칙의 양쪽 물건을 모두 구매한 비율이다. c와 d를 함께 구매한 것은 3번과 4번의 2건이므로, {c} ⇒ {d}의 지지도는 50%가 아니라 **40%**이다.

NoteBase

선지별 해설

연관 규칙의 지지도는 전체 트랜잭션 중 규칙의 왼쪽과 오른쪽 물건을 모두 포함한 비율이다. 신뢰도는 왼쪽 물건을 포함한 트랜잭션 중 오른쪽 물건까지 모두 포함한 비율이다. 따라서 지지도의 분모는 전체 거래 수이고, 신뢰도의 분모는 왼쪽 조건을 만족하는 거래 수이다. 표의 전체 트랜잭션은 5건이다.

① O: {a, c} ⇒ {b}의 지지도는 a, b, c를 모두 구매한 거래의 비율이다. 해당 거래는 1번과 3번의 2건이므로, 전체 5건을 기준으로 $\frac{2}{5} \times 100 = 40\%$이다.

② X: {c} ⇒ {d}의 지지도는 c와 d를 함께 구매한 거래 수를 **전체 거래 수 5건**으로 나누어 구한다. 해당 거래는 3번과 4번의 2건이므로 $\frac{2}{5} \times 100 = 40\%$이다. 50%는 c를 구매한 4건을 분모로 삼은 신뢰도이며, 선지의 지지도는 40%로 고쳐야 한다.

③ O: {c} ⇒ {a, b}의 신뢰도는 **c를 구매한 거래**를 기준으로 a와 b도 모두 구매한 비율이다. c를 구매한 거래는 1·2·3·4번의 4건이고, 이 중 a와 b를 모두 구매한 거래는 1번과 3번의 2건이다. 따라서 $\frac{2}{4} \times 100 = 50\%$이다.

④ O: {a} ⇒ {b}의 신뢰도는 a를 구매한 거래 중 b도 구매한 비율이다. a를 구매한 거래는 1·2·3·5번의 4건이고, 이 중 b도 구매한 거래는 1·3·5번의 3건이다. 따라서 $\frac{3}{4} \times 100 = 75\%$이다.

NoteBase

문 7. 개체-관계 모델(E-R Model)을 통해 학생들의 수강 신청 관리시스템 데이터베이스를 설계하고자 한다. 사상 원소 수(mapping cardinality)를 포함한 개체-관계 다이어그램(E-R Diagram)으로 표현할 때, 다음 <요구사항>에 부합하지 않는 다이어그램은? (단, 참여 제약 조건(participation constraint)은 고려하지 않는다)

<요구사항>
◯ 대학에는 여러 개의 학과가 있고 각 학과에는 여러 명의 학생이 소속되어 있다. 또한 각 학생은 필요에 따라 복수전공 등으로 여러 학과에 소속될 수 있다.
◯ 학생은 한 학기에 여러 개의 교과목을 수강하며, 교과목이 운영되려면 반드시 여러 명의 학생이 수강해야 한다.
◯ 학과에는 여러 명의 교수가 있으며, 각 교수는 단일 학과에 소속된다.
◯ 교수는 한 학기에 여러 교과목을 강의할 수 있고, 각 교과목은 한 명의 교수가 강의한다.

① 

② 

③ 

④ 

정답 3번

간단한 해설

한 학과에 여러 교수가 소속되고 각 교수는 단일 학과에 속하므로, 교수–학과는 **N:1 관계**이다. ③은 교수 쪽을 1, 학과 쪽을 N으로 표시하여 대응 방향을 뒤집었다.

NoteBase

선지별 해설

1:N 관계에서는 1 쪽 개체 하나에 N 쪽 개체 여러 개가 대응한다. N:M은 양쪽 모두 여러 개가 대응할 수 있는 다대다 관계이다. 이 문제에서는 참여 제약을 제외하고, 관계의 최대 대응 수를 판단한다.

① O: 학생 쪽 N, 학과 쪽 M으로 표시한 **다대다 관계**이다. 한 학과에 여러 학생이 소속되고, 한 학생도 복수전공 등으로 여러 학과에 소속될 수 있다는 요구사항에 부합한다.

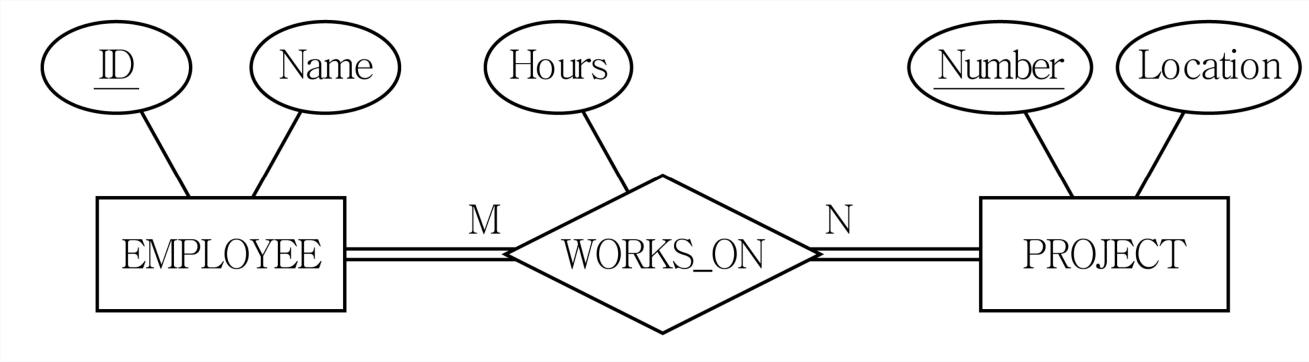
② O: 학생 쪽 N, 교과목 쪽 M으로 표시한 **다대다 관계**이다. 한 학생이 여러 교과목을 수강하고, 한 교과목을 여러 학생이 수강하므로 양쪽 모두 복수의 상대 개체와 연결될 수 있다.

③ X: 교수 쪽 1, 학과 쪽 N으로 표시되어 있다. 이는 교수 한 명이 여러 학과에 소속될 수 있고 한 학과에 소속되는 교수는 최대 한 명이라는 뜻이다. 요구사항은 한 학과에 여러 교수가 있고 각 교수는 단일 학과에 소속된다는 것이므로, **교수 쪽 N, 학과 쪽 1**로 고쳐야 한다.

④ O: 교수 쪽 1, 교과목 쪽 N으로 표시한 **일대다 관계**이다. 한 교수가 한 학기에 여러 교과목을 강의할 수 있고, 각 교과목의 담당 교수는 한 명이라는 요구사항에 부합한다.

NoteBase

문 8. 다음 개체-관계 다이어그램(E-R Diagram)을 관계형 데이터베이스의 스키마(schema)로 적절하게 나타낸 것은? (단, 밑줄이 표시된 속성은 키(key) 속성을 뜻한다)



① EMPLOYEE_WORKS_ON_PROJECT(ID, Name, Hours, Number, Location)

② EMPLOYEE(ID, Name, Hours)
PROJECT(Number, Location, Hours)

③ EMPLOYEE(ID, Name)
WORKS_ON(Hours)
PROJECT(Number, Location)

④ EMPLOYEE(ID, Name)
WORKS_ON(ID, Number, Hours)
PROJECT(Number, Location)

정답 4번

간단한 해설

EMPLOYEE와 PROJECT는 **다대다(M:N) 관계**이므로, 두 개체의 키를 함께 갖는 별도의 WORKS_ON 릴레이션이 필요하다. Hours는 직원이나 프로젝트 자체가 아니라 **특정 직원의 특정 프로젝트 참여**에 관한 속성이므로 WORKS_ON에 둔다.

NoteBase

공통 해설

그림에서 WORKS_ON 양쪽의 M과 N은 한 직원이 여러 프로젝트에 참여하고, 한 프로젝트에도 여러 직원이 참여할 수 있음을 뜻한다. 이러한 다대다 관계를 관계형 스키마로 변환할 때는 각 개체를 별도의 릴레이션으로 유지하고, 관계도 별도의 릴레이션으로 만든다. 관계 릴레이션에는 **양쪽 개체의 키와 관계 자체의 속성**을 함께 넣는다.

이 규칙을 적용하면 EMPLOYEE(ID, Name), PROJECT(Number, Location), WORKS_ON(ID, Number, Hours)가 된다. EMPLOYEE의 키는 ID이고 PROJECT의 키는 Number이다. WORKS_ON에서는 ID와 Number가 각각 EMPLOYEE와 PROJECT를 참조하는 외래키이며, **ID와 Number의 조합**이 키가 된다. ID만으로는 어느 프로젝트에 참여하는지, Number만으로는 어느 직원이 참여하는지 구별할 수 없기 때문이다.

Hours는 그림에서 WORKS_ON에 연결되어 있으므로 직원과 프로젝트의 쌍별로 저장한다. 예를 들어 같은 직원도 참여하는 프로젝트에 따라 작업 시간이 다를 수 있다. 따라서 Hours를 각 개체의 속성으로 옮기거나, WORKS_ON에 Hours만 두고 ID와 Number를 빠뜨리면 그 시간을 특정 참여 관계에 연결할 수 없다. 개체의 Name과 Location까지 하나의 참여 릴레이션에 모두 합치는 대신, 개체 정보는 각 개체 릴레이션에 두고 참여 정보만 WORKS_ON에 둔다.

NoteBase

문 9. 다음 '학생성적' 릴레이션과 각 투플의 레코드 번호를 참고하여 '성적' 속성에 대한 비트맵 인덱스를 적절히 표현한 것은?

	학생성적		
레코드번호	학생ID	성별	성적
0	88	남성	A
1	13	여성	D
2	87	여성	A
3	53	남성	C
4	75	남성	B

①

A	10100
B	00001
C	00010
D	01000

②

A	00110
B	00001
C	00100
D	01000

③

A	00010
B	00100
C	00011
D	00001

NoteBase

④

A	01111
B	11110
C	11101
D	10111

정답 1번

간단한 해설

비트맵 인덱스는 레코드 번호 순서대로 **해당 성적이면 1, 아니면 0**을 기록한다. 성적 A는 레코드 0번과 2번에 있으므로 비트열은 10100이다.

계산 과정

비트맵 인덱스는 속성의 서로 다른 값마다 비트열을 만든다. 각 비트는 하나의 레코드에 대응하며, 그 레코드의 속성값이 해당 값과 같으면 1, 다르면 0이다. 이 문제에서는 레코드가 5개이므로 성적 A·B·C·D마다 5비트가 필요하다. 비트 위치는 왼쪽부터 **레코드 번호 0 → 1 → 2 → 3 → 4**에 대응하며, 그 순서의 성적은 A → D → A → C → B이다.

A: 레코드 0번과 2번의 성적이 A이므로 첫 번째와 세 번째 비트를 1로 놓는다. 나머지는 0이므로 1·0·1·0·0, 즉 10100이다.

B: 레코드 4번만 성적이 B이므로 다섯 번째 비트만 1이다. 따라서 0·0·0·0·1, 즉 00001이다.

C: 레코드 3번만 성적이 C이므로 네 번째 비트만 1이다. 따라서 0·0·0·1·0, 즉 00010이다.

D: 레코드 1번만 성적이 D이므로 두 번째 비트만 1이다. 따라서 0·1·0·0·0, 즉 01000이다.

구성한 비트맵은 A: 10100, B: 00001, C: 00010, D: 01000이며, ①의 표현과 일치한다.

NoteBase

문 10. 다음 릴레이션에서 속성(A, B, C)들의 함수 종속을 화살표(→)로 표현할 때 옳지 않은 것은? (단, 아래 투플을 기준으로 판단하며, 새로운 투플은 추가되지 않는다고 가정한다)

A	B	C
c	d	i
f	j	g
h	j	g
f	c	b

① A → B는 성립하지 않는다.

② B → C는 성립한다.

③ (A, B) → C는 성립한다.

④ (B, C) → A는 성립한다.

정답 4번

간단한 해설

B와 C가 모두 (j, g)로 같은 두 투플에서 A는 각각 f와 h로 다르다. 따라서 **(B, C)가 같아도 A가 하나로 결정되지 않으므로** (B, C) → A는 성립하지 않는다.

NoteBase

선지별 해설

함수 종속 X → Y는 X 값이 같은 모든 투플에서 Y 값도 같다는 뜻이다. X가 여러 속성이면 그 속성들의 값이 모두 같은 투플끼리 비교한다. 이 문제는 새로운 투플이 추가되지 않는다고 했으므로, 표에 주어진 네 투플만을 기준으로 판단한다.

① O: A가 f인 두 투플은 (f, j, g)와 (f, c, b)이다. **A는 같지만 B는 j와 c로 다르므로** A → B는 성립하지 않는다. 따라서 성립하지 않는다는 선지의 설명이 맞다.

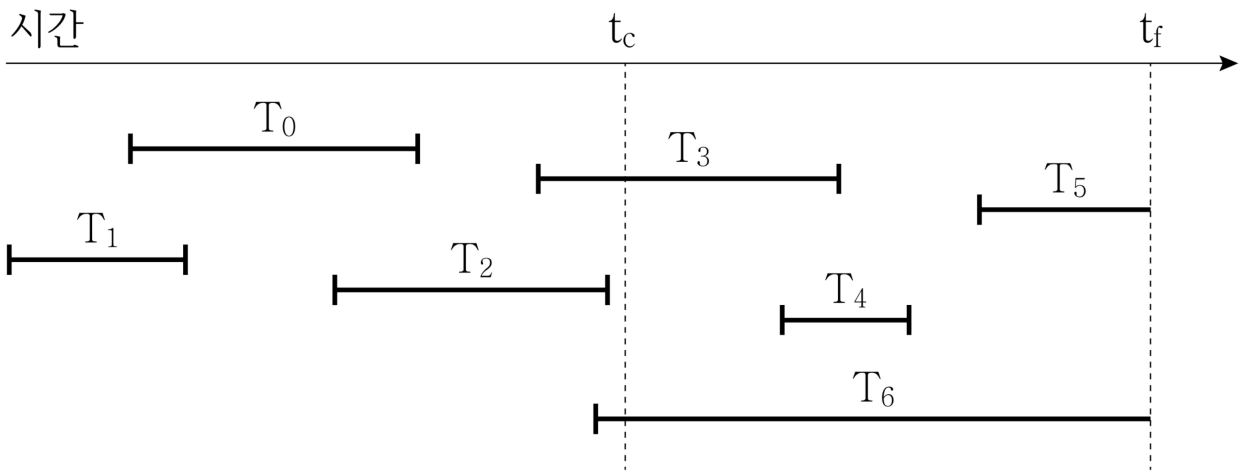
② O: B가 j인 두 투플에서 C는 모두 g이다. 나머지도 B가 d일 때 C는 i, B가 c일 때 C는 b로 대응한다. **같은 B 값에 서로 다른 C 값이 대응하는 경우가 없으므로** B → C는 성립한다.

③ O: (A, B)의 값은 (c, d), (f, j), (h, j), (f, c)로 모두 다르다. 따라서 **같은 (A, B)에 서로 다른 C가 대응하는 경우가 없어** (A, B) → C는 성립한다. A 값 f가 반복되더라도 B 값이 다르므로 같은 (A, B)로 보지 않는다.

④ X: 두 번째 투플 (f, j, g)와 세 번째 투플 (h, j, g)는 (B, C)가 모두 (j, g)이지만 A는 f와 h로 다르다. **같은 (B, C)가 서로 다른 A에 대응하므로** (B, C) → A는 성립하지 않는다. 선지의 '성립한다'를 '성립하지 않는다'로 고쳐야 한다.

NoteBase

문 11. 어떤 시스템에서 다음 그림과 같이 7개의 트랜잭션 T_0 ~ T_6이 수행 완료되었거나 수행 중이다. 시간 t_c 시점에서 검사점(checkpoint)이 실행되었고, 시간 t_f 시점에서 시스템 장애(failure)가 발생하여 정지되었다고 가정한다. 이때 재실행(redo)해야 하는 트랜잭션만을 모두 고르면?



① T_3, T_4

② T_5, T_6

③ T_0, T_1, T_2

④ T_3, T_4, T_5, T_6

정답 1번

간단한 해설

재실행(redo)은 **검사점 이후에 완료되어 장애 전에 확정된 변경**을 로그로 다시 반영하는 작업이다. 그림의 T_3, T_4는 이 구간에 완료되지만, T_5, T_6은 장애 당시 아직 수행 중이므로 구분해야 한다.

NoteBase

도식 해설

그림에서 시간은 오른쪽으로 흐르며, 점선 t_c는 검사점, t_f는 장애 시점이다. 오른쪽 끝에 완료 표시가 있는 트랜잭션은 그 지점에서 수행을 마쳤고, 완료 표시 없이 장애 시점까지 이어지는 트랜잭션은 미완료 상태이다. 이 문제의 검사점 회복에서는 완료 시점에 따라 회복 작업을 구분한다.

T_0, T_1, T_2: 모두 오른쪽 완료 표시가 t_c 왼쪽에 있다. **검사점 전에 완료된 변경**은 검사점에서 디스크 반영이 보장되므로 다시 실행할 필요가 없다.

T_3, T_4: 오른쪽 완료 표시가 모두 t_c와 t_f 사이에 있다. 완료된 변경이 아직 디스크에 모두 반영되지 않았을 수 있으므로, 로그에 기록된 변경을 다시 적용하는 **redo 대상**이다. T_3는 검사점 전에 시작했고 T_4는 검사점 후에 시작했지만, 둘 다 검사점 이후 장애 전에 완료되었다는 점이 결정적이다.

T_5, T_6: 오른쪽 완료 표시 없이 t_f까지 이어져 **장애 당시 미완료**이다. 이들의 변경을 완료된 결과로 확정해서는 안 되므로 redo 대상으로 고르지 않는다. 데이터베이스에 이미 반영된 변경이 있다면 undo로 취소한다.

따라서 재실행 대상은 T_3, T_4이며, ①의 구성과 일치한다.

NoteBase

문 12. 다음 두 릴레이션 'Emp'와 'Dep'에서 id(사번)와 d_code(부서코드)는 각 릴레이션의 기본키이다. 두 릴레이션에 대하여 다음 <관계대수>의 연산 결과와 동일한 결과를 나타내는 SQL문으로 옳은 것은? (단, 아래 릴레이션을 기준으로 판단하며, 새로운 투플은 추가되지 않는다고 가정한다)

Emp

id	name	d_code
10	홍길동	b
20	이순신	c
30	김유신	a

Dep

d_code	d_name
a	전산부
b	인사부
c	총무부

<관계대수>

$$Temp1 \leftarrow \sigma_{Emp.d_code=Dep.d_code}(Emp \times \sigma_{d_name='전산부'}(Dep))$$

$$Temp2 \leftarrow \rho_{E(id,\ name,\ code,\ d_code,\ d_name)}(Temp1)$$

$$\pi_{d_code}(Temp2)$$

①
```
SELECT d_code FROM Emp, Dep
WHERE d_name = '전산부';
```

②
```
SELECT d.d_code AS d_code FROM Emp AS e, Dep AS d
WHERE d_code = d_code HAVING d_name = '전산부';
```

③
```
SELECT d_code FROM Emp AS e, Dep AS d
WHERE e.d_code = d.d_code and e.d_name = '전산부';
```

④
```
SELECT d.d_code AS d_code
FROM Emp AS e INNER JOIN Dep AS d
ON e.d_code = d.d_code WHERE d_name = '전산부';
```

정답 4번

간단한 해설

관계대수는 **부서코드가 같은 사원과 부서를 연결**하고, 부서명이 '전산부'인 행의 부서코드를 출력한다. 주어진 자료에서 이 조건을 만족하는 사원은 김유신이므로 결과는 **d_code가 a인 한 행**이다.

NoteBase

핵심 풀이

Temp1: 먼저 Dep에서 부서명이 '전산부'인 행을 선택하면 부서코드가 a인 행만 남는다. 이를 Emp와 카티션 곱한 뒤 **Emp.d_code = Dep.d_code**인 행만 선택하므로, 부서코드가 a인 김유신의 행이 연결된다. Temp1의 유일한 투플은 속성 순서대로 (30, 김유신, a, a, 전산부)이다.

Temp2: 이름 변경 연산 ρ는 Temp1의 릴레이션 이름을 E로, 다섯 속성 이름을 차례로 id, name, code, d_code, d_name으로 바꾼다. 따라서 Emp에서 온 부서코드는 code가 되고, **Dep에서 온 부서코드는 d_code**가 된다. 속성 이름만 바뀌며 투플의 값은 그대로이다.

최종 결과와 SQL 대응: 프로젝션 π는 Temp2에서 d_code 열만 추출하므로 결과는 열 이름이 d_code이고 값이 a인 한 행이다. ④의 INNER JOIN은 ON e.d_code = d.d_code로 같은 부서코드끼리 연결하고, WHERE d_name = '전산부'로 전산부 소속 행만 남긴다. 이어 SELECT d.d_code AS d_code가 Dep의 부서코드를 d_code라는 이름으로 출력하므로 관계대수의 결과와 일치한다. d_name은 Dep에만 있으므로 WHERE절에서 테이블 별칭 없이 사용해도 구별된다. **주어진 자료에서는 조건에 맞는 행이 하나뿐**이므로 DISTINCT가 없어도 프로젝션 결과와 중복 차이가 생기지 않는다.

다른 SQL문과의 구분: Emp와 Dep에는 모두 d_code가 있으므로, 두 테이블을 함께 조회할 때는 e.d_code나 d.d_code처럼 소속을 밝혀야 한다. ①과 ②의 한정되지 않은 d_code는 어느 테이블의 열인지 모호하며, 특히 ②의 d_code = d_code는 두 테이블의 부서코드를 명시적으로 연결한 조건이 아니다. ①에는 부서코드를 연결하는 조건 자체도 없다. ③의 e.d_name은 Emp에 존재하지 않는 열이므로 부서명 조건에는 d.d_name을 써야 하며, SELECT의 d_code도 소속을 명확히 해야 한다.

NoteBase

문 13. 다음 릴레이션 '프로젝트'에 대하여 아래와 같은 SQL문을 수행한 결과는?

프로젝트		
관리번호	관리자사번	연구비
1	1	500
2	1	2200
3	2	900
4	3	400
5	4	1500
6	1	1700
7	2	3000
8	3	1700
9	2	550
10	3	2000

```
SELECT 관리자사번, SUM(연구비) AS 연구비총액
FROM 프로젝트
WHERE 연구비 > 1000
GROUP BY 관리자사번
HAVING COUNT(*) > 1;
```

①

관리자사번	연구비총액
1	3900
3	3700

②

관리자사번	연구비총액
1	3900
2	3000
3	3700

NoteBase

③

관리자사번	연구비총액
1	4400
2	4450
3	4100

④

관리자사번	연구비총액
1	3900
2	3000
3	3700
4	1500

정답 1번

간단한 해설

연구비가 1000을 초과하는 행만 남긴 뒤, **남은 행이 2개 이상인 관리자**만 출력한다. 관리자사번 1과 3이 이 조건을 충족하며, 연구비총액은 각각 3900과 3700이다.

계산 과정

행 선택: WHERE 연구비 > 1000은 그룹을 만들기 전에 **개별 행의 연구비**를 검사한다. 관리번호 1·3·4·9의 행은 제외되고, 관리번호 2·5·6·7·8·10의 행만 남는다.

관리자사번 1: 남은 행은 관리번호 2와 6의 2개이다. GROUP BY 관리자사번으로 두 행을 같은 그룹에 묶으며, 연구비 합계는 2200 + 1700 = 3900이다.

관리자사번 2: 남은 행은 관리번호 7의 1개이다. 연구비 합계는 3000이다. 관리번호 3과 9의 연구비는 WHERE 조건에서 제외되었으므로 합계에 포함하지 않는다.

관리자사번 3: 남은 행은 관리번호 8과 10의 2개이다. 연구비 합계는 1700 + 2000 = 3700이다.

관리자사번 4: 남은 행은 관리번호 5의 1개이며, 연구비 합계는 1500이다.

그룹 선택과 출력: HAVING COUNT(*) > 1은 **WHERE를 통과한 행의 개수가 2개 이상인 그룹**만 남긴다. 따라서 관리자사번 2와 4의 그룹은 제외된다. SELECT는 관리자사번과 각 그룹의 연구비 합계를 출력하고, AS 연구비총액은 합계 열의 이름을 지정한다. 결과는 (관리자사번 1, 연구비총액 3900), (관리자사번 3, 연구비총액 3700)의 두 행으로, ①과 일치한다.

NoteBase

문 14. 다음 두 릴레이션 '고객'과 '주문'에 대하여 아래 SQL문의 수행 결과값은?

고객

ID	Name	나이
good	김유신	20
best	유관순	30
better	강감찬	50
nice	이순신	20

주문

ID	고객ID	제품
1001	good	brush
1002	better	book
1003	best	umbrella
1004	nice	cup

```
SELECT SUM(나이) FROM 고객
WHERE ID NOT IN (SELECT 고객ID FROM 주문 WHERE
제품 LIKE '%u%');
```

① 50
② 70
③ 80
④ 100

정답 1번

간단한 해설

LIKE '%u%'는 **u가 포함된 제품명**을 찾고, NOT IN은 해당 제품을 주문한 고객을 제외한다. 남는 고객은 better(강감찬) 한 명이므로 나이의 합은 **50**이다.

NoteBase

계산 과정

1. 부분 질의의 고객ID 구하기: LIKE에서 %는 길이가 0 이상인 임의의 문자열을 뜻하므로, 제품 LIKE '%u%'는 **제품명에 u가 포함된** 행을 찾는다. '주문'에서 brush, umbrella, cup이 이에 해당하며, 부분 질의가 반환하는 고객ID는 각각 good, best, nice이다. book에는 u가 없어 better는 이 결과에 포함되지 않는다.

2. 바깥 질의의 고객 선택: ID NOT IN은 '고객'의 ID가 **부분 질의 결과에 없는** 행만 남긴다. good(김유신), best(유관순), nice(이순신)는 제외되고, better(강감찬)만 남는다. 이 고객의 나이는 50이다.

3. 나이 합계 계산: SUM(나이)는 조건을 통과한 고객 행들의 나이를 합산한다. 남은 행이 better 한 행이므로 합계는 50이며, 이는 ①의 값과 일치한다.

NoteBase

문 15. 다음 두 릴레이션 '학생'과 '그룹'을 왼쪽 외부 조인(left outer join)한 결과(학생 ⟕ 그룹)로 옳은 것은?

학생	
SID	GID
100	A
200	B
300	A
400	F

그룹	
GID	Score
A	90
B	80
C	70
D	60
E	50

①

SID	GID	Score
100	A	90
200	B	80
300	A	90

②

SID	GID	Score
100	A	90
200	B	80
300	A	90
400	F	NULL

③

SID	GID	Score
100	A	90
200	B	80
300	A	90
NULL	C	70
NULL	D	60
NULL	E	50

④

SID	GID	Score
100	A	90
200	B	80
300	A	90
400	F	NULL
NULL	C	70
NULL	D	60
NULL	E	50

정답 2번

간단한 해설

왼쪽 외부 조인은 **왼쪽 '학생' 릴레이션의 모든 튜플을 보존**한다. GID가 F인 학생 400도 결과에 남으며, '그룹'에 F가 없으므로 **Score만 NULL**로 채운다.

NoteBase

계산 과정

조인 기준: 두 릴레이션의 공통 속성은 그룹 식별자인 GID이다. GID가 같은 튜플을 결합하여 학생의 SID·GID에 그룹의 Score를 붙인다. 왼쪽 외부 조인이므로 **일치하는 그룹이 없어도 학생 튜플을 유지**한다.

일치하는 튜플 결합: 학생 (100, A)와 (300, A)는 그룹 (A, 90)에 각각 연결되어 (100, A, 90), (300, A, 90)이 된다. 학생 (200, B)는 그룹 (B, 80)에 연결되어 (200, B, 80)이 된다. 두 학생이 같은 그룹 A에 속해도 SID가 다르므로 별개의 튜플로 남는다. 여기까지 일치하는 학생 튜플은 3개이다.

일치하지 않는 학생 보존: 학생 (400, F)에 대응하는 GID가 '그룹'에 없다. 이때 **학생의 SID 400과 GID F는 그대로 두고**, 오른쪽에서 가져올 Score만 NULL로 채워 (400, F, NULL)을 추가한다. NULL은 대응하는 값이 없음을 나타내며 0으로 바꾸지 않는다.

일치하지 않는 그룹 처리: 그룹 C·D·E는 대응하는 학생이 없다. 왼쪽 외부 조인은 오른쪽 릴레이션에만 있는 튜플까지 보존하지 않으므로 이 세 그룹을 위한 행은 추가하지 않는다.

결과 확인: 일치하는 학생 3개에 일치하지 않는 학생 1개를 더하므로 결과 튜플 수는 3 + 1 = 4개이다. SID, GID, Score 열에 위 네 튜플을 담은 결과는 ②의 표와 일치한다.

NoteBase

문 16. 다음에서 설명하는 모든 조건을 만족하는 릴레이션 R의 정규형은?

◯ 릴레이션 R의 함수 종속 관계에서 기본키가 아닌 모든 속성들이 기본키에 이행적 함수 종속이 아니다.
◯ 릴레이션 R에 속한 모든 속성의 도메인이 원잣값만으로 되어 있으며, 기본키가 아닌 모든 속성이 기본키에 완전 함수 종속이다.
◯ 릴레이션 R의 함수 종속 관계에서 결정자(determinant) 중에 후보키가 아닌 것이 포함되어 있을 수 있다.

① 제2정규형
② 제3정규형
③ 제4정규형
④ BCNF

정답 2번

간단한 해설

원잣값과 완전 함수 종속이라는 제2정규형의 조건에 **비키 속성의 이행적 함수 종속 배제**를 더한 설명이다.
제3정규형은 BCNF와 달리, 후보키가 아닌 결정자가 남아 있을 수 있다.

NoteBase

구성 요소별 해설

함수 종속은 어떤 속성의 값이 정해지면 다른 속성의 값도 하나로 정해지는 관계이며, 이때 값을 결정하는 쪽을 결정자라고 한다. 후보키는 각 튜플을 유일하게 식별하는 최소 속성 집합이고, 기본키는 후보키 중에서 선택한 키이다.

◯ 첫째 조건 — 이행적 함수 종속 배제: 기본키로 다른 비키 속성을 결정하고 그 속성을 거쳐 다시 비키 속성을 결정하는 간접적인 종속을 배제한다는 뜻이다. 제2정규형에서는 이런 종속이 남을 수 있지만, 제3정규형은 이를 제거한다. 따라서 첫째 조건은 **제2정규형에서 제3정규형으로 나아가는 조건**이다.

◯ 둘째 조건 — 원잣값과 완전 함수 종속: 모든 속성이 더 나누지 않는 하나의 값인 원잣값을 가지는 것은 제1정규형의 조건이다. 여기에 비키 속성이 기본키 전체에 종속되고 **기본키의 일부에만 종속되는 부분 함수 종속이 없어야** 제2정규형이 된다. 완전 함수 종속은 키의 일부만으로는 해당 속성의 값을 결정할 수 없다는 뜻이다. 따라서 둘째 조건은 제3정규형의 바탕이 되는 제2정규형을 설명한다.

◯ 셋째 조건 — 후보키가 아닌 결정자의 허용: 제3정규형에서는 결정자가 슈퍼키가 아니더라도, 결정되는 속성이 **어느 후보키에 포함되는 속성**이면 해당 비자명한 함수 종속을 허용한다. 여기서 슈퍼키는 릴레이션의 모든 속성을 결정하여 튜플을 유일하게 식별하는 속성 집합이다. 반면 BCNF는 모든 비자명한 함수 종속의 결정자가 슈퍼키일 것을 요구한다. 따라서 제3정규형이라고 해서 BCNF까지 보장되지는 않는다. 제4정규형은 BCNF보다 강한 정규형이므로 이 조건들만으로는 보장되지 않는다.

둘째 조건의 제2정규형에 첫째 조건의 이행적 함수 종속 배제를 더하고, 셋째 조건처럼 BCNF보다 완화된 결정자 조건을 허용하는 정규형은 ②의 제3정규형이다.

NoteBase

문 17. U1 ~ U5는 데이터베이스 일반 사용자들의 계정명이며, 사용자 U1은 '성적' 테이블을 생성한 이 테이블의 소유자이다. 다음과 같은 순서대로 5번까지 각 사용자들이 SQL문을 수행하였다. 이에 대한 설명으로 옳은 것은?

순번	수행한 사용자	수행한 SQL문
1	U1	GRANT SELECT, INSERT ON 성적 TO U2 WITH GRANT OPTION;
2	U1	GRANT SELECT, INSERT ON 성적 TO U3 WITH GRANT OPTION;
3	U3	GRANT SELECT ON 성적 TO U4;
4	U1	REVOKE SELECT ON 성적 FROM U2;
5	U4	GRANT SELECT ON 성적 TO U5;

① 사용자 U1은 데이터베이스 관리자(DBA)가 아니므로 다른 사용자에게 성적 테이블에 대한 SELECT 및 INSERT 권한을 부여할 수 없다.

② 사용자 U2는 성적 테이블에 대한 SELECT 권한이 있다.

③ 사용자 U4는 성적 테이블에 대한 SELECT 권한이 있다.

④ 사용자 U5는 성적 테이블에 대한 SELECT 권한이 있다.

정답 3번

간단한 해설

U4는 **U1 → U3 → U4 경로**로 SELECT 권한을 받았으므로, U2의 권한이 회수되어도 조회할 수 있다. 다만 U4는 **WITH GRANT OPTION 없이** 권한을 받아 U5에게 이를 다시 부여할 수 없다.

NoteBase

선지별 해설

① X: U1은 DBA가 아니어도 **성적 테이블의 소유자**이므로 다른 사용자에게 해당 테이블의 권한을 부여할 수 있다. 따라서 1번과 2번 SQL문으로 U2와 U3에게 SELECT·INSERT 권한을 부여할 수 있으며, WITH GRANT OPTION을 붙였으므로 이들도 받은 권한을 다른 사용자에게 부여할 수 있다.

② X: U2는 1번 SQL문으로 SELECT·INSERT 권한을 받지만, 4번 SQL문의 REVOKE SELECT에 의해 **SELECT 권한이 회수된다**. 따라서 5번까지 수행한 뒤에는 성적 테이블을 조회할 수 없다. 이때 회수 대상에 포함되지 않은 INSERT 권한은 유지된다.

③ O: U3는 2번 SQL문으로 SELECT 권한과 그 권한을 다른 사용자에게 부여할 수 있는 자격을 받았다. 따라서 3번 SQL문으로 U4에게 SELECT 권한을 부여할 수 있다. 4번 SQL문은 U2의 SELECT 권한만 회수하므로, **U3를 거쳐 U4에게 부여된 SELECT 권한은 유지된다**.

④ X: U4는 3번 SQL문으로 SELECT 권한을 받았지만 **WITH GRANT OPTION을 받지 않았다**. 조회 권한이 있다는 것과 그 권한을 다른 사용자에게 부여할 수 있다는 것은 다르다. 따라서 U4가 실행한 5번 SQL문은 권한 부족으로 실패하며, U5는 SELECT 권한을 얻지 못한다.

NoteBase

문 18. NoSQL의 특징으로 옳은 것만을 모두 고르면?

ㄱ. 반정형 데이터(semi-structured data)를 처리하기에는 적합하지 않다.
ㄴ. 트랜잭션의 ACID 성질을 완벽히 보장하는 것이 중요하다.
ㄷ. 문서-기반 모델은 NoSQL에서 사용되는 데이터 모델의 한 종류이다.
ㄹ. 스키마를 요구하지 않는 경우가 있다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄹ
④ ㄷ, ㄹ

정답 4번

간단한 해설

NoSQL은 **문서 기반 모델**을 사용할 수 있으며, 데이터 구조를 미리 고정하는 **스키마를 요구하지 않는 경우**도 있다.

보기별 해설

ㄱ. X: 반정형 데이터는 JSON처럼 필드나 계층 구조는 있지만 모든 자료가 동일한 고정 구조를 따르지는 않는 데이터이다. NoSQL은 유연한 데이터 모델을 사용하므로 반정형 데이터를 처리하기에 **적합하다**.

ㄴ. X: ACID는 트랜잭션의 원자성·일관성·격리성·지속성을 뜻한다. NoSQL은 분산 환경의 확장성이나 가용성 등을 중시하여 일부 ACID 성질의 보장 수준을 완화할 수 있으므로, **ACID의 완전한 보장이 공통적인 핵심 특징은 아니다**. 다만 ACID 트랜잭션을 지원하는 NoSQL 제품도 있으므로, ACID를 전혀 지원하지 않는다고 단정해서도 안 된다.

ㄷ. O: 문서 기반 모델은 관련 필드와 값을 하나의 문서로 묶어 저장하는 모델이다. JSON과 같은 형식으로 중첩된 데이터도 표현할 수 있으며, **NoSQL의 데이터 모델 중 하나**이다.

ㄹ. O: 일부 NoSQL은 저장 전에 모든 데이터에 적용할 **고정된 스키마를 미리 정의하도록 요구하지 않는다**. 예를 들어 한 문서에는 이름과 나이를, 다른 문서에는 이름과 이메일을 저장할 수 있다. 이는 데이터에 구조가 전혀 없다는 뜻이 아니라, 문서마다 필드 구성이 달라도 된다는 뜻이다.

옳은 보기는 ㄷ, ㄹ이며, 이 조합에 해당하는 선택지는 ④이다.

NoteBase

문 19. B^+**-트리 인덱스에 대한 설명으로 옳은 것만을 모두 고르면?**

ㄱ. 루트(root)를 포함한 내부 노드(internal node)는 데이터 파일 레코드를 가리키는 포인터를 갖는다.
ㄴ. 트리의 루트(root)에서 단말 노드(leaf node)까지 모든 경로의 길이가 같은 균형 트리(balanced tree) 형태이다.
ㄷ. 단말 노드(leaf node)는 검색키 값을 기초로 선형 순서로 되어 있어 데이터 레코드들을 그 키 값의 순서에 따라 능률적으로 순차 접근할 수 있게 해 준다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 3번

간단한 해설

B^+-트리의 내부 노드에 있는 포인터는 데이터 레코드가 아니라 **하위 노드**를 가리킨다. 모든 단말 노드는 같은 깊이에 있고, 키 순서로 연결되어 있어 **순차 접근**에 효율적이다.

보기별 해설

ㄱ. X: 루트를 포함한 내부 노드는 검색할 하위 노드를 결정하는 검색키와 **하위 노드를 가리키는 포인터**를 갖는다. 데이터 파일의 레코드를 가리키는 포인터는 단말 노드에 있다. 따라서 내부 노드가 데이터 레코드에 직접 연결된다는 설명을, 내부 노드를 거쳐 단말 노드에 도달한 뒤 레코드에 접근한다는 내용으로 고쳐야 한다.

ㄴ. O: B^+-트리는 **모든 단말 노드가 같은 깊이**에 있는 균형 트리이다. 따라서 루트에서 어느 단말 노드로 내려가더라도 거치는 간선의 수가 같아, 특정 가지에만 검색 경로가 길게 치우치지 않는다.

ㄷ. O: 단말 노드의 검색키는 정렬되어 있고, 단말 노드들도 **키 값의 순서로 연결**되어 있다. 처음 필요한 키가 있는 단말 노드를 찾은 뒤에는 연결된 단말 노드를 차례로 따라가며 레코드에 접근할 수 있으므로, 매번 루트부터 다시 검색하지 않고 순차 접근이나 범위 검색을 효율적으로 수행할 수 있다.

옳은 보기는 ㄴ, ㄷ이므로 ③의 조합에 해당한다.

NoteBase

문 20. 다음은 '학원생' 릴레이션이다. 후보키(candidate key)에 해당하는 것은? (단, 아래 투플을 기준으로 판단하며, 투플은 추가·변경·삭제되지 않는다고 가정한다)

학원생					
학원생이름	전화번호	나이	학교이름	학년	반
홍길동	1237－6542	19	A고	3	1
이순신	3218－8465	18	B고	2	2
홍길동	4567－8910	20	A고	1	1
유관순	3247－6512	18	C고	3	1

① {전화번호}

② {학원생이름}

③ {전화번호, 학년}

④ {학년, 반}

정답 1번

간단한 해설

전화번호는 네 투플에서 모두 달라 각 투플을 구별하며, **전화번호 하나만으로 식별이 가능**하다. 따라서 {전화번호}는 유일성과 최소성을 모두 갖춘 후보키이다.

NoteBase

공통 해설

후보키는 유일성과 최소성을 모두 만족하는 속성 집합이다. 유일성은 모든 투플을 서로 구별할 수 있다는 뜻이고, 최소성은 **유일성을 유지하면서 속성을 더 제거할 수 없다**는 뜻이다. 문제에서 투플이 추가·변경·삭제되지 않는다고 했으므로 제시된 네 투플만을 기준으로 판단한다.

{전화번호}: 1237-6542, 3218-8465, 4567-8910, 3247-6512가 모두 달라 유일성을 만족한다. 단일 속성으로 이루어져 있고 이를 제거하면 네 투플을 구별할 수 없으므로 최소성도 만족한다.

{학원생이름}: 홍길동이라는 이름을 가진 투플이 두 개 있다. 이름만으로는 두 투플을 구별할 수 없으므로 유일성을 만족하지 못한다.

{전화번호, 학년}: 전화번호가 모두 달라 유일성은 만족한다. 그러나 **학년을 제거해도 전화번호만으로 식별할 수 있으므로** 최소성을 만족하지 못한다. 따라서 슈퍼키이지만 후보키는 아니다.

{학년, 반}: 첫 번째 홍길동과 유관순의 학년·반이 모두 (3, 1)이다. 두 속성을 함께 사용해도 이 두 투플을 구별할 수 없으므로 유일성을 만족하지 못한다.

따라서 제시된 선택지 중 후보키에 해당하는 속성 집합은 {전화번호}이다.

NoteBase