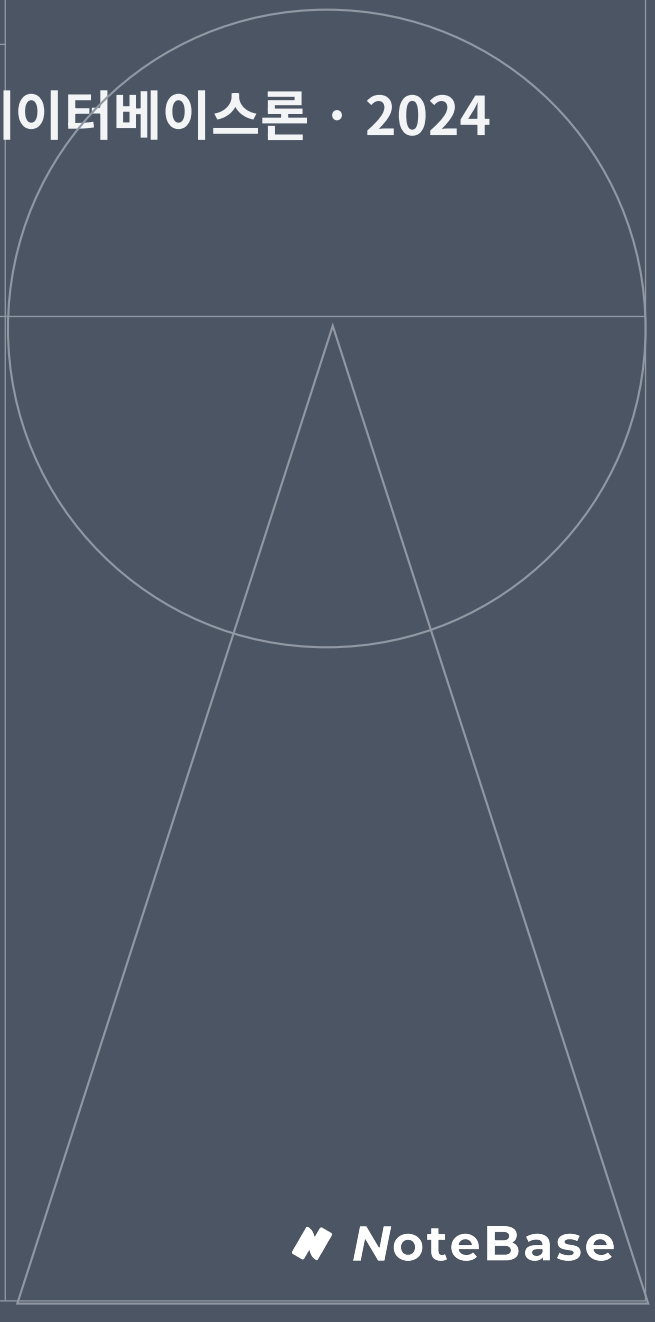
기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2024



NoteBase

기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2024

공무원/경찰/공채(경위) · 데이터베이스론 · 2024

문 1. 사용자가 원하는 데이터를 얻기 위해 데이터값을 사용하여 조건을 제시하면 이에 해당하는 데이터를 검색해 주는 데이터베이스 특징으로 가장 적절한 것은?

① 계속적인 변화

② 내용에 따른 참조

③ 동시 공유

④ 실시간 접근

정답 2번

간단한 해설

데이터의 저장 위치가 아니라 **데이터값과 조건을 기준으로 검색**하는 것이 '내용에 따른 참조'이다. 문제의 핵심은 접근 속도나 이용자 수가 아니라 검색 기준에 있다.

개념 구분

내용에 따른 참조: 데이터가 저장된 물리적 주소나 위치가 아니라 **데이터의 값과 조건**을 기준으로 원하는 데이터를 찾는 특성이다. 예를 들어 '점수가 80점 이상인 학생'이라는 조건으로 자료를 검색하는 경우가 이에 해당한다. 문제의 '데이터값을 사용하여 조건을 제시'한다는 표현이 결정적 단서이다.

계속적인 변화: 삽입·수정·삭제를 통해 데이터베이스의 내용이 현실의 상태를 반영하도록 갱신되는 특성이다. 검색의 기준이 값이라는 뜻과는 다르다.

동시 공유: 여러 사용자나 응용프로그램이 데이터베이스를 함께 이용할 수 있는 특성이다. 핵심은 **여러 사용자의 공동 이용**이지 검색 조건의 표현 방식이 아니다.

실시간 접근: 사용자의 요구에 필요한 데이터를 적절한 시간 안에 제공하는 특성이다. 응답의 신속성에 관한 것으로, 내용에 따른 참조와 구별한다.

따라서 데이터값으로 조건을 지정해 검색하는 특징은 ② '내용에 따른 참조'이다.

NoteBase

문 2. 데이터 사전에 대한 설명으로 가장 적절하지 않은 것은?

① 사용자가 접근할 수 있다.

② 스키마, 사상정보, 다양한 제약조건 등을 저장한다.

③ 데이터에 대한 데이터를 의미해 메타 데이터라고도 한다.

④ 데이터를 저장하는 데이터베이스의 일종으로 사용자 데이터베이스라고도 한다.

정답 4번

간단한 해설

데이터 사전은 실제 업무 데이터가 아니라 스키마·제약조건 등의 메타데이터를 관리하는 **시스템 데이터베이스**이다. 사용자 데이터베이스라고 부르는 것은 두 역할을 혼동한 것이다.

선지별 해설

① O: 사용자는 권한이 허용하는 범위에서 데이터 사전이나 이를 보여 주는 시스템 뷰를 조회할 수 있다. 다만 **조회할 수 있다는 것과 내용을 임의로 변경할 수 있다는 것은 다르다**. 데이터 사전의 정의 정보는 일반적으로 DBMS가 데이터 정의 작업 등에 맞추어 관리한다.

② O: 데이터 사전에는 테이블·열·자료형 같은 스키마 정보, 스키마 사이의 사상정보, 키와 무결성 제약조건 등 데이터베이스의 구조와 관리 정보가 저장된다. 이는 업무 데이터 자체가 아니라 **데이터를 정의하고 관리하기 위한 정보**이다.

③ O: 메타데이터는 '데이터에 대한 데이터'이다. 데이터 사전이 담고 있는 테이블 이름, 속성의 자료형, 제약조건 등의 정보가 이에 해당한다. 따라서 데이터 사전은 메타데이터를 저장·관리하는 체계로 이해한다.

④ X: 데이터 사전은 사용자 업무 데이터를 담는 사용자 데이터베이스와 구별되는 **시스템 데이터베이스 또는 시스템 카탈로그**에 해당한다. 고객의 실제 구매내역은 사용자 데이터이고, 구매내역 테이블의 열 이름·자료형·키 정의는 데이터 사전에 담기는 메타데이터이다.

NoteBase

문 3. 데이터베이스 구조에서 하위 스키마를 변경하더라도 상위 스키마가 영향을 받지 않는 특성으로 가장 적절한 것은?

① 최소성
② 원자성
③ 독립성
④ 유일성

정답 3번

간단한 해설

스키마를 단계별로 분리하여 하위 구조의 변경이 상위 구조로 전파되지 않게 하는 성질은 **데이터 독립성**이다. 변경되는 단계에 따라 논리적 독립성과 물리적 독립성으로 나뉜다.

개념 구분

데이터 독립성은 **하위 단계의 스키마가 바뀌어도 상위 단계의 스키마나 응용프로그램이 영향을 받지 않도록 하는 성질**이다. 데이터베이스의 3단계 구조는 사용자의 관점인 외부 스키마, 전체 논리 구조인 개념 스키마, 실제 저장 구조인 내부 스키마를 구분한다.

개념 스키마를 변경해도 외부 스키마나 응용프로그램에 영향을 주지 않는 것은 **논리적 데이터 독립성**이다. 내부 스키마의 저장 방식이나 접근 경로를 변경해도 개념 스키마에 영향을 주지 않는 것은 **물리적 데이터 독립성**이다. 각 단계 사이의 사상이 변경의 영향을 분리한다.

최소성은 후보키를 구성하는 속성 중 불필요한 속성이 없어야 한다는 성질이고, 유일성은 키 값으로 각 튜플을 구별할 수 있다는 성질이다. 원자성은 보통 트랜잭션의 연산이 전부 수행되거나 전혀 수행되지 않아야 한다는 성질을 뜻한다. 어느 것도 스키마 간 변경 영향의 분리를 뜻하지 않는다.

따라서 하위 스키마 변경으로부터 상위 스키마를 보호하는 특성은 ③ 독립성이다.

NoteBase

문 4. 데이터베이스 관리자의 역할을 모두 고른 것은?

가. 일관된 데이터베이스 상태를 정의
나. DBMS의 성능 검사 후 저장구조 변경
다. 데이터베이스의 개념적 스키마를 결정
라. 사용자의 요구사항을 분석하여 데이터베이스를 구성할 데이터 결정

① 가
② 가, 나
③ 가, 나, 다
④ 가, 나, 다, 라

정답 4번

간단한 해설

데이터베이스 관리자는 운영만 담당하는 것이 아니라 **데이터와 스키마의 결정, 무결성 관리, 저장구조 및 성능 관리**까지 담당한다. 사용자 요구 분석부터 일관된 상태의 유지까지 모두 이러한 관리 범위에 포함된다.

보기별 해설

가. O: 일관된 데이터베이스 상태란 데이터가 정해진 무결성 제약조건을 만족하는 상태이다. 데이터베이스 관리자는 기본키의 중복 금지, 참조 관계 등 **데이터가 지켜야 할 제약조건**을 정하고 유지하도록 관리한다. 이는 단순히 값이 자주 바뀌지 않는 상태를 뜻하지 않는다.

나. O: 데이터베이스 관리자는 응답시간, 처리량, 저장공간 사용 등 운영 성능을 점검한다. 병목이 발견되면 인덱스나 파일 배치 등 **저장구조를 조정하여 성능을 개선**할 수 있다. 따라서 성능 검사와 그 결과에 따른 저장구조 변경은 관리자의 역할에 해당한다.

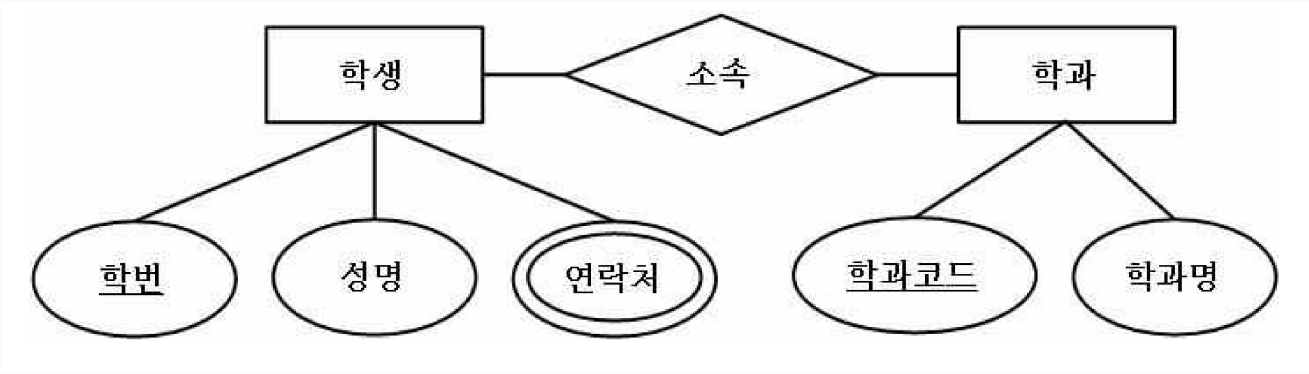
다. O: 개념적 스키마는 데이터베이스 전체에 어떤 데이터가 있고 그 관계와 제약이 무엇인지를 나타내는 통합된 구조이다. 데이터베이스 관리자는 조직 전체의 요구를 조정하여 이 구조를 결정·관리하는 역할을 맡는다. 특정 사용자의 관점인 외부 스키마나 물리적 저장방식인 내부 스키마와 구별된다.

라. O: 어떤 데이터를 저장할지 정하려면 먼저 사용자의 업무와 정보 요구를 파악해야 한다. 데이터베이스 관리자는 요구사항을 분석하고 조정하여 **관리할 데이터의 범위와 내용**을 결정하는 데 관여한다. 실제 조직에서는 설계자나 분석가와 역할을 나누기도 하지만, 이 문항은 데이터베이스 관리자의 역할을 넓게 묻고 있다.

가, 나, 다, 라가 모두 관리자의 역할에 포함되므로 이들을 전부 고른 ④에 해당한다.

NoteBase

문 5. 다음 E-R 다이어그램에서 다중값 속성은?



① 학번

② 성명

③ 소속

④ 연락처

정답 4번

간단한 해설

E-R 다이어그램에서 다중값 속성은 **이중 타원**으로 나타낸다. 그림에서 연락처가 이중 타원으로 표시되어 있어 한 학생이 여러 연락처를 가질 수 있음을 뜻한다.

도식 해설

다중값 속성은 하나의 개체가 여러 개의 값을 가질 수 있는 속성이다. 이 E-R 표기법에서는 **이중 타원**으로 나타낸다. 그림에서 학생 개체에 연결된 연락처만 이중 타원으로 표시되어 있으므로, 한 학생이 여러 연락처를 가질 수 있음을 뜻한다.

학번: 밑줄 친 단일 타원으로 표시되어 있다. 밑줄은 학생을 식별하는 **키 속성**이라는 뜻이지 여러 값을 갖는다는 뜻이 아니다. 학과 쪽의 학과코드도 같은 방식으로 키 속성임을 나타낸다.

성명: 단일 타원으로 표시된 일반 속성이다. 이름이라는 내용만으로 다중값 여부를 추정하지 않고 그림의 표기를 기준으로 판단한다.

소속: 학생과 학과 사이를 연결하는 마름모로 표시되어 있다. 이는 두 개체 유형 사이의 **관계**이며 속성 표기가 아니다. 학생과 학과 자체는 각각 사각형으로 표시된 개체 유형이다.

따라서 이중 타원으로 나타낸 ④ 연락처가 다중값 속성이다.

NoteBase

문 6. 차수(degree)와 카디널리티(cardinality)에 대한 설명으로 적절한 것은 모두 몇 개인가?

가. 차수는 릴레이션에서 속성의 개수이다.
나. 유효한 릴레이션의 최소 차수는 0이다.
다. 유효한 릴레이션의 최소 카디널리티는 1이다.
라. 카디널리티는 한 릴레이션의 투플의 개수이다.

① 1개
② 2개
③ 3개
④ 4개

정답 2번

간단한 해설

차수는 **속성 수**, 카디널리티는 **투플 수**이다. 일반적인 테이블 정의에서 속성은 최소 하나 필요하지만 데이터 행은 없을 수 있으므로 최소 차수는 1, 최소 카디널리티는 0이다.

보기별 해설

가. O: 차수는 릴레이션을 구성하는 **속성, 즉 열의 개수**이다. 예를 들어 학번·이름·학과의 세 속성으로 이루어진 학생 릴레이션의 차수는 3이다. 저장된 학생 수가 늘어나더라도 속성 구성이 같으면 차수는 변하지 않는다.

나. X: 이 문제에서 사용하는 일반적인 관계형 데이터베이스의 테이블 정의에서는 유효한 릴레이션이 적어도 하나의 속성을 가져야 하므로 **최소 차수는 1**이다. 데이터 행이 없다는 것과 속성 자체가 없다는 것을 구별해야 한다.

다. X: 카디널리티는 현재 저장된 투플의 수이므로 **최솟값은 0**이다. 속성으로 정의된 구조는 유지하면서 투플을 하나도 담지 않은 빈 릴레이션도 가능하다. 따라서 유효하다는 이유만으로 데이터 행이 최소 한 개 필요하지는 않다.

라. O: 카디널리티는 릴레이션에 속하는 **투플, 즉 행의 개수**이다. 학생 정보가 20행 저장되어 있으면 카디널리티는 20이며, 투플을 삽입하거나 삭제하면 달라질 수 있다.

적절한 보기는 가, 라의 2개이므로 ②와 일치한다.

NoteBase

문 7. 릴레이션의 특징에 대한 설명으로 가장 적절하지 않은 것은?

① 한 릴레이션의 모든 속성값은 원자값이다.

② 한 릴레이션에 포함된 투플은 모두 상이하다.

③ 한 릴레이션에 포함된 투플 사이에는 순서가 있다.

④ 한 릴레이션을 구성하는 속성 사이에는 순서가 없다.

정답 3번

간단한 해설

릴레이션은 투플의 집합이므로 **투플 사이에 논리적인 순서가 없다**. 표시되는 행의 순서가 달라져도 투플 집합이 같으면 같은 릴레이션이다.

선지별 해설

① O: 관계형 모델의 기본 릴레이션에서 각 투플의 속성값은 해당 도메인의 **단일한 원자값**이다. 한 칸에 여러 값을 목록처럼 함께 저장하는 반복집합과 구별하며, 이 원자성은 제1정규형의 조건과 연결된다.

② O: 릴레이션은 투플의 집합이므로 **완전히 동일한 투플이 중복되지 않는다**. 일부 속성값이 같을 수는 있지만 모든 속성값이 같은 두 투플을 서로 다른 원소로 취급하지 않는다.

③ X: 릴레이션의 투플 사이에는 **논리적인 순서가 없다**. 화면에 표시되는 행 순서를 바꾸어도 같은 투플 집합이면 동일한 릴레이션이다. 조회 결과를 특정 순서로 표시하는 것과 릴레이션 자체에 순서가 존재한다는 것은 다르다.

④ O: 속성은 열의 위치가 아니라 이름으로 식별하므로 **속성 사이에도 논리적인 순서가 없다**. 열을 재배치하더라도 속성 이름과 그에 대응하는 값이 유지되면 릴레이션의 의미는 변하지 않는다.

NoteBase

문 8. <사원>, <부서> 릴레이션이 다음과 같을 때, 릴레이션 수(A), 속성 수(B), 투플 수(C)의 총합(A+B+C)은?

사원

사번	성명	부서번호
100	김철수	1
101	이영희	2
102	최고나	2
103	홍길동	1

부서

부서번호	부서명
1	영업부
2	총무부

① 11

② 12

③ 13

④ 14

정답 3번

간단한 해설

두 표의 릴레이션은 2개, 속성은 3 + 2 = 5개, 데이터 투플은 4 + 2 = 6개이다. **머리글은 투플에서 제외**하므로 총합은 2 + 5 + 6 = 13이다.

NoteBase

계산 과정

릴레이션은 표 하나, 속성은 열 하나, 투플은 실제 데이터를 담은 행 하나에 해당한다. 속성명이 적힌 머리글 행은 투플로 세지 않는다.

릴레이션 수 A: '사원'과 '부서'라는 두 개의 표가 있으므로 A = 2이다.

속성 수 B: 사원 릴레이션의 속성은 사번·성명·부서번호의 3개이고, 부서 릴레이션의 속성은 부서번호·부서명의 2개이다. 따라서 B = 3 + 2 = 5이다. **부서번호가 두 표에 공통으로 있어도 각 릴레이션의 속성으로 각각 센다**.

투플 수 C: 사원 릴레이션에는 사번 100~103의 데이터 4행이 있고, 부서 릴레이션에는 영업부와 총무부의 데이터 2행이 있다. 따라서 C = 4 + 2 = 6이다. 속성 이름이 있는 행은 데이터 행이 아니므로 제외한다.

총합은 A + B + C = 2 + 5 + 6 = 13이므로 ③에 해당한다.

NoteBase

문 9. 다음은 후보키에 대한 설명이다. 빈칸 ㉠, ㉡에 해당하는 가장 적절한 것은?

투플을 유일하게 구별하기 위해 꼭 필요한 최소한의 속성들로만 이루어지므로 (㉠) 중에서 (㉡)을 만족하는 것이 후보키가 된다.

① ㉠ 슈퍼키, ㉡ 최소성
② ㉠ 슈퍼키, ㉡ 유일성
③ ㉠ 대체키, ㉡ 최소성
④ ㉠ 대체키, ㉡ 유일성

정답 1번

간단한 해설

후보키는 투플을 유일하게 식별하는 **슈퍼키** 중에서 불필요한 속성이 없는 것이다. 이미 확보된 유일성에 더해 **최소성**을 만족해야 한다.

빈칸별 해설

㉠ 슈퍼키: 슈퍼키는 릴레이션의 각 투플을 유일하게 구별할 수 있는 속성 또는 속성들의 집합이다. 식별에 불필요한 속성을 함께 포함해도 유일성이 유지되면 슈퍼키가 된다. 후보키는 이 슈퍼키들 가운데 불필요한 속성이 없는 것이므로 빈칸에는 **슈퍼키**가 들어간다.

㉡ 최소성: 최소성은 키를 구성하는 속성을 하나라도 제거하면 더 이상 모든 투플을 유일하게 식별할 수 없다는 성질이다. 여기서 최소란 여러 후보키 중 속성 개수가 가장 적다는 뜻이 아니라 **그 키에서 불필요한 속성을 더 뺄 수 없다**는 뜻이다. 슈퍼키는 이미 유일성을 만족하므로 후보키가 되기 위해 추가로 요구되는 조건은 최소성이다.

따라서 ㉠은 슈퍼키, ㉡은 최소성이며 이 조합은 ①에 해당한다.

헷갈리는 개념

기본키는 여러 후보키 가운데 대표 식별자로 선택한 키이고, 대체키는 기본키로 선택되지 않은 나머지 후보키이다. 따라서 **대체키도 유일성과 최소성을 만족**하며, 슈퍼키에서 최소성을 만족하는 것을 대체키라고 부르는 것은 아니다.

NoteBase

문 10. 관계형 데이터베이스의 제약 조건에 대한 설명으로 가장 적절하지 않은 것은?

① 릴레이션 간에 독립성 유지

② 기본키의 값은 널을 가질 수 없음

③ 키 속성의 값은 반드시 도메인에 속하는 원자값

④ 후보키의 값은 릴레이션 내의 각 투플을 유일하게 식별

정답 1번

간단한 해설

관계형 데이터베이스의 제약 조건은 데이터의 정확성과 일관성을 유지하기 위한 조건이다. 릴레이션 사이에는 외래키로 연결된 관계가 존재할 수 있으며, 필요한 것은 독립성 보장이 아니라 **참조 관계의 일관성 유지**이다.

선지별 해설

① X: 릴레이션 사이의 독립성을 유지한다는 것은 관계형 데이터베이스의 무결성 제약 조건을 설명한 것이 아니다. 릴레이션들은 외래키 등을 통해 서로 관련될 수 있으며, **참조 무결성**은 이러한 참조 관계에서 일관성을 유지하도록 한다. 외래키에 널이 아닌 값이 들어 있다면 참조 대상 키 값이 실제로 존재해야 한다. 데이터 독립성은 데이터의 논리적·물리적 구조 변화가 다른 계층에 미치는 영향을 줄이는 개념으로, 무결성 제약과 구별된다.

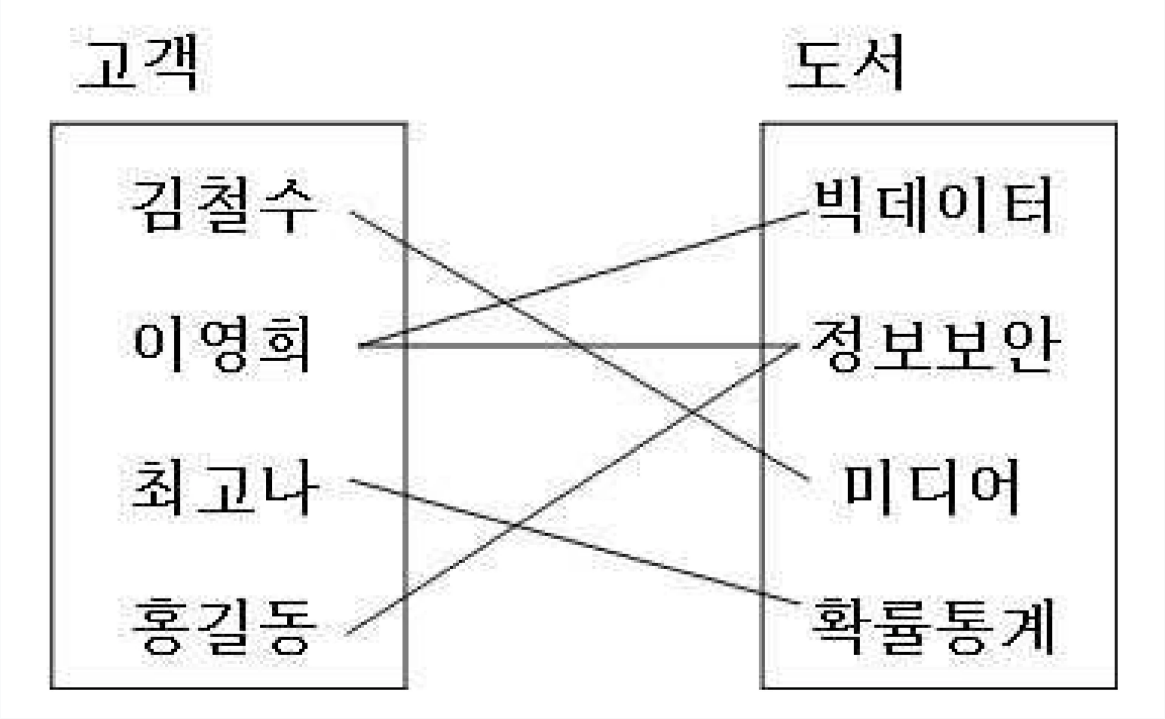
② O: 기본키는 각 투플을 식별하는 기준이므로 **널 값을 가질 수 없다**. 이것이 개체 무결성의 핵심이다. 기본키가 여러 속성으로 구성된 복합키라면 그 구성 속성 중 어느 하나에도 널을 허용하지 않는다.

③ O: 속성값은 해당 속성에 정해진 도메인의 허용 범위와 자료형을 따라야 한다. 기본적인 관계 모델에서는 각 속성값을 **더 분해하지 않는 원자값**으로 취급하며 키 속성도 예외가 아니다. 복합키가 존재할 수 있다는 사실은 구성 속성 하나가 여러 값을 동시에 담는다는 뜻이 아니다.

④ O: 후보키는 릴레이션 안의 각 투플을 **유일하게 식별**할 수 있는 최소한의 속성 집합이다. 따라서 서로 다른 투플이 같은 후보키 값을 가질 수 없다. 여기에 포함된 속성을 더 제거하면 유일한 식별이 불가능하다는 최소성도 갖는다.

NoteBase

문 11. <고객>, <도서> 개체의 관계가 다음과 같다. 관계의 유형으로 가장 적절한 것은?



① 다대다 관계

② 다대일 관계

③ 일대다 관계

④ 일대일 관계

정답 1번

간단한 해설

이영희는 두 도서와 연결되고, '정보보안'도 두 고객과 연결된다. **고객과 도서 양쪽 모두 여러 상대와 연결될 수 있으므로 다대다 관계**이다.

NoteBase

도식 해설

관계의 유형은 한쪽 개체 하나가 반대쪽의 몇 개체와 연결될 수 있는지를 양방향으로 확인해 결정한다. 모든 고객이나 모든 도서에 여러 연결이 있어야 다대다가 되는 것은 아니다.

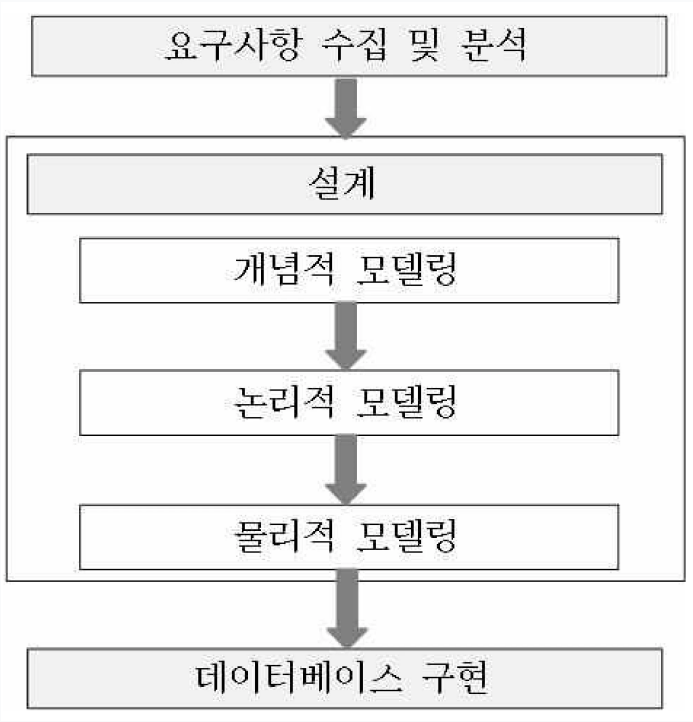
고객에서 도서를 본다. 그림에서 **이영희가 '빅데이터'와 '정보보안' 두 도서**에 연결되어 있으므로 고객 한 명이 여러 도서와 관계를 맺을 수 있다.

도서에서 고객을 본다. **'정보보안'은 이영희와 홍길동 두 고객**에 연결되어 있으므로 도서 하나도 여러 고객과 관계를 맺을 수 있다.

양쪽 모두 하나의 개체가 상대쪽의 여러 개체와 연결되므로 다대다 관계이며 ①과 일치한다. 일대다·다대일은 한쪽 방향만 다수이고, 일대일은 양쪽 모두 최대 하나씩 연결된다는 점에서 다르다.

NoteBase

문 12. 다음 데이터 모델링 과정에서 DBMS 선정을 고려해야 하는 가장 적절한 단계는?



① 요구사항 수집 및 분석

② 개념적 모델링

③ 논리적 모델링

④ 물리적 모델링

정답 3번

간단한 해설

개념 모델을 **목표 DBMS가 지원하는 데이터 모델의 논리적 구조**로 바꾸는 단계가 논리적 모델링이다. 따라서 이 단계에서 DBMS 선정을 고려하며, 저장 방법과 인덱스 설계는 그다음 물리적 모델링에서 구체화한다.

NoteBase

핵심 풀이

그림은 요구사항 수집 및 분석 이후 설계 단계에서 개념적 모델링, 논리적 모델링, 물리적 모델링을 거쳐 구현하는 흐름을 보여 준다. 이 문제는 전체 순서를 배열하는 것이 아니라 **DBMS 선택을 고려할 단계**를 식별하는 문제이다.

요구사항 수집 및 분석: 사용자가 어떤 데이터를 필요로 하고 어떤 업무·처리·제약이 있는지 파악한다. 설계의 기초를 정리하는 단계이며, 특정 DBMS의 구조로 변환하는 단계는 아니다.

개념적 모델링: 현실의 개체·속성·관계를 E-R 모델 등으로 표현한다. 업무의 의미를 기술하는 단계로서 **특정 DBMS에 독립적**인 개념 구조를 만든다.

논리적 모델링: 개념 모델을 실제 사용할 DBMS가 지원하는 데이터 모델에 맞는 논리적 스키마로 변환한다. 관계형 모델을 사용할 경우 릴레이션과 키·제약을 정하고 정규화 등을 수행한다. 따라서 **목표 DBMS와 그 데이터 모델을 고려**해야 하는 단계이다.

물리적 모델링: 정해진 논리 구조를 실제 저장장치에 효율적으로 구현하기 위해 파일 조직, 인덱스, 저장 구조와 접근 방법 등을 설계한다. 이 단계 역시 DBMS 특성의 영향을 받지만, DBMS를 고려하기 시작하는 논리적 모델링보다 뒤에 위치한다.

따라서 제시된 과정에서 DBMS 선정을 고려해야 하는 가장 적절한 단계는 ③ 논리적 모델링이다.

NoteBase

문 13. DB의 물리적 설계 단계에서 트랜잭션과 저장공간 설계 측면에서 고려할 사항으로 가장 적절하지 않은 것은?

① 응답시간 최소화

② 트랜잭션 처리도

③ 저장공간의 효율화

④ 트랜잭션 인터페이스 설계

정답 4번

간단한 해설

물리적 설계는 데이터를 실제로 저장하고 접근하는 방법을 정하면서 **응답시간·처리량·저장공간 효율**을 고려한다. 트랜잭션 인터페이스 설계는 사용자나 응용프로그램과의 입출력·호출 방식을 정하는 작업이므로 이 성능·저장구조 설계 기준과 구별된다.

선지별 해설

데이터베이스의 물리적 설계는 논리적 데이터 구조를 실제 저장장치에 구현하기 위해 파일 조직, 레코드 배치, 인덱스와 접근 경로 등을 정하는 단계이다.

① O: 응답시간은 요청을 보낸 시점부터 처리 결과를 받을 때까지 걸리는 시간이다. 물리적 설계에서는 자주 사용하는 트랜잭션의 접근 방식에 맞추어 인덱스 등을 선택하여 **응답시간을 줄이는 것**을 고려한다.

② O: 트랜잭션 처리도는 **단위 시간에 처리할 수 있는 트랜잭션의 수**를 나타내는 처리량이다. 저장구조와 접근 경로가 효율적이면 같은 시간에 더 많은 작업을 처리할 수 있으므로 중요한 설계 기준이다.

③ O: 저장공간의 효율화는 데이터와 인덱스 등에 필요한 공간을 적절하게 사용하는 것이다. 인덱스를 늘리면 검색은 빨라질 수 있지만 추가 공간과 갱신 비용이 필요하므로, 물리적 설계에서는 **성능과 저장공간 사이의 절충**을 고려한다.

④ X: 트랜잭션 인터페이스 설계는 트랜잭션의 입력·출력이나 사용자·응용프로그램과의 접점을 정하는 작업이다. 이는 **트랜잭션과 응용프로그램의 논리적 설계**에 해당하며, 물리적 저장구조를 평가하는 응답시간·트랜잭션 처리도·저장공간 효율과 같은 고려 기준이 아니다.

NoteBase

문 14. 다음은 EMPLOYEE와 DEPARTMENT 테이블을 정의하는 SQL이다. 밑줄 친 ㉠, ㉡에 가장 적절한 것은?

```
CREATE TABLE DEPARTMENT (
        dname   VARCHAR(15) NOT NULL,
        dnum    INT         NOT NULL,
        ____㉠____ ( dnum ) ) ;

CREATE TABLE EMPLOYEE (
        name    VARCHAR(15) NOT NULL,
        eno     CHAR(9)     NOT NULL,
        dno     INT         NOT NULL,
        ____㉠____ ( eno ),
        ____㉡____ ( dno ) REFERENCES
                        DEPARTMENT ( dnum ) ) ;
```

① ㉠: PRIMARY KEY, ㉡: FOREIGN KEY

② ㉠: FOREIGN KEY, ㉡: FOREIGN KEY

③ ㉠: PRIMARY KEY, ㉡: UNIQUE

④ ㉠: FOREIGN KEY, ㉡: UNIQUE

정답 1번

간단한 해설

각 테이블의 행을 식별하는 'dnum'과 'eno'에는 **PRIMARY KEY**를 지정한다. 'dno'가 DEPARTMENT의 'dnum'을 참조하도록 하는 선언은 **FOREIGN KEY**이다.

NoteBase

빈칸별 해설

㉠ PRIMARY KEY: 부서 테이블에서는 'dnum'이 각 부서를, 직원 테이블에서는 'eno'가 각 직원을 유일하게 식별하도록 지정한다. 따라서 두 위치에는 **기본키를 선언하는 PRIMARY KEY**가 들어간다. 기본키 값은 중복되거나 NULL일 수 없으며, 그림의 'PRIMARY KEY(dnum)'과 'PRIMARY KEY(eno)'는 각각 다른 테이블의 기본키 선언이다.

㉡ FOREIGN KEY: 직원 테이블의 'dno'는 해당 직원이 소속된 부서의 번호이다. 뒤에 'REFERENCES DEPARTMENT(dnum)'이 있으므로 이 열이 부서 테이블의 'dnum'을 참조하도록 **FOREIGN KEY**를 지정해야 한다. 'dno'에는 NOT NULL도 선언되어 있으므로 각 직원의 부서 번호는 반드시 실제 부서의 'dnum' 값 중 하나여야 한다. UNIQUE는 해당 테이블에서 값의 중복을 제한할 뿐 다른 테이블을 참조하는 관계를 선언하지 않는다.

따라서 ㉠은 PRIMARY KEY, ㉡은 FOREIGN KEY이며 이 조합은 ①에 해당한다.

NoteBase

문 15. 다음 테이블에 대해, 모든 사원들의 부서(dno) 평균급여(salary)가 3,000,000원 이상인 부서를 검색하는 SQL 명령문이다. 밑줄 친 ㉠, ㉡에 가장 적절한 것은?

EMPLOYEE

eno	ename	salary	dno
1001	홍길동	3000000	1
1002	김철수	2800000	2
1003	이영희	3200000	2

```
SELECT   dno, AVG( salary ) AS avgsal
FROM     EMPLOYEE
   ㉠    dno
   ㉡    AVG( salary ) >= 3000000 ;
```

① ㉠: WHERE, ㉡: AND

② ㉠: WHERE, ㉡: HAVING

③ ㉠: GROUP BY, ㉡: AND

④ ㉠: GROUP BY, ㉡: HAVING

정답 4번

간단한 해설

부서별 평균급여를 구하려면 **GROUP BY**로 같은 부서의 사원들을 묶어야 한다. 이렇게 계산한 평균급여에 조건을 적용해 부서를 골라내는 절은 **HAVING**이다.

NoteBase

빈칸별 해설

㉠ GROUP BY: 부서별 평균급여를 구하려면 dno 값이 같은 사원들을 하나의 그룹으로 묶어야 한다. GROUP BY dno를 사용하면 AVG(salary)가 **각 부서에 속한 사원들의 급여 평균**을 계산한다. 예를 들어 제시된 테이블에서 dno가 2인 김철수와 이영희는 같은 그룹에 속한다. WHERE는 개별 행을 조건에 따라 걸러내는 절이므로 부서별 그룹을 만드는 역할을 하지 않는다.

㉡ HAVING: AVG(salary)에 대한 조건은 개별 사원의 급여가 아니라 **그룹별로 계산한 평균급여**에 적용된다. 따라서 그룹화와 집계 후 조건을 만족하는 부서만 남기는 HAVING을 사용한다. WHERE는 그룹화 전에 개별 행을 거르므로 이 집계 조건을 직접 지정하는 자리에 쓸 수 없다. AND는 여러 조건을 연결하는 논리 연산자이므로 HAVING 절 자체를 대신하지 못한다.

㉠에 GROUP BY, ㉡에 HAVING을 넣는 조합은 ④와 일치한다.

NoteBase

문 16. 다음의 R, S 테이블에 대해 SQL ‘가’와 ‘나’의 실행결과 레코드 수의 합은?

R

A	B	C
1	a	10
1	a	11
2	b	25
3	b	22
5	c	17

S

A	D	E
1	g	x
1	g	y
2	h	w
4	k	w
6	m	z

가. SELECT DISTINCT A FROM R ;
나. SELECT R.A, R.C, S.E
FROM R LEFT OUTER JOIN S
ON R.A = S.A ;

① 10
② 11
③ 12
④ 13

정답 2번

간단한 해설

가에서 DISTINCT로 중복을 제거하면 4행이다. 나에서는 A = 1인 행들이 4개 조합을 만들고, **왼쪽 R의 미대응 행도 보존**되어 7행이므로 합은 11이다.

NoteBase

계산 과정

가. DISTINCT는 조회한 열의 중복값을 제거한다. R의 A값은 1, 1, 2, 3, 5이므로 중복된 1을 한 번만 남기면 **1, 2, 3, 5의 4개 레코드**가 반환된다.

나. LEFT OUTER JOIN은 조인조건에 맞는 행을 결합하되, **오른쪽에 대응하는 행이 없어도 왼쪽 R의 행을 남긴다**. R.A = S.A를 기준으로 각 A값의 결합 수를 센다.

A = 1: R에 2행, S에도 2행이 있으므로 모든 대응 조합이 만들어져 2 × 2 = 4행이다. 선택된 R.C와 S.E의 쌍은 (10, x), (10, y), (11, x), (11, y)이다.

A = 2: R과 S에 각각 1행이 있어 1행이 반환된다. A = 3과 A = 5는 R에만 존재하지만 왼쪽 외부조인이므로 각각 1행씩 남고 S.E는 NULL이 된다. S에만 있는 A = 4와 A = 6은 결과에 추가되지 않는다.

따라서 나의 결과는 4 + 1 + 1 + 1 = 7행이고, 두 SQL 결과 레코드 수의 합은 4 + 7 = 11로 ②와 일치한다.

NoteBase

문 17. SQL에서 뷰의 개념에 대한 설명으로 가장 적절하지 않은 것은?

① 다른 테이블이나 기존의 뷰로부터 유도된 테이블이다.

② 뷰는 SELECT, DELETE, INSERT, UPDATE 명령문을 이용하여 정의한다.

③ 뷰에 대한 갱신 연산은 제한을 받는다.

④ 뷰를 이용하여 복잡한 질의를 간단한 질의로 표현할 수 있다.

정답 2번

간단한 해설

뷰는 **CREATE VIEW 문에 SELECT 질의를 지정**하여 정의한다. INSERT·UPDATE·DELETE는 데이터를 변경하는 명령이며, 뷰를 정의하는 명령이 아니다.

선지별 해설

뷰는 저장된 질의를 바탕으로 다른 테이블이나 뷰의 데이터를 보여 주는 가상 테이블이다. 일반적인 뷰는 별도의 데이터 집합 자체가 아니라 질의 정의를 저장한다.

① O: 뷰는 하나 이상의 기본 테이블이나 이미 정의된 다른 뷰에 대한 질의로부터 유도된다. 사용자는 이 **유도된 결과를 하나의 테이블처럼** 조회할 수 있다.

② X: 뷰의 정의에는 **CREATE VIEW와 SELECT 질의**를 사용한다. SELECT는 뷰가 보여 줄 행과 열을 정하며, DELETE·INSERT·UPDATE는 데이터 삭제·삽입·변경을 수행하는 명령이다. 뷰를 정의하는 것과, 허용되는 뷰를 통해 원본 데이터를 변경하는 것은 서로 다른 작업이다.

③ O: 뷰를 갱신하려면 그 변경을 기본 테이블의 어떤 행에 반영해야 하는지 명확해야 한다. 예를 들어 집계 결과를 보여 주는 뷰는 집계값을 바꾸어도 원본의 어떤 값을 바꿔야 할지 모호하므로 **일반적으로 갱신이 제한**된다. 모든 뷰가 갱신 가능한 것은 아니다.

④ O: 복잡한 조인이나 조건을 뷰의 정의에 넣어 두면, 사용자는 뷰 이름을 대상으로 간단한 SELECT 문을 작성할 수 있다. 따라서 뷰는 **복잡한 질의를 감추고 재사용**하게 하여 질의 작성을 단순화한다.

NoteBase

문 18. 다음과 같이 릴레이션들이 주어졌을 때, 관계대수식 $AB \div C$ 연산의 결과 투플 수는?

AB

A#	B#
a1	b1
a1	b2
a1	b4
a2	b1
a2	b2
a2	b4
a2	b6
a3	b3
a4	b1
a4	b2
a4	b4

C

B#
b2
b4

① 1

② 2

③ 3

④ 4

정답 3번

간단한 해설

디비전은 C에 있는 **b2와 b4 모두에 대응하는 A#**를 찾는다. 이를 만족하는 값은 a1, a2, a4이므로 결과 투플은 3개이다.

NoteBase

계산 과정

디비전의 조건: 관계대수의 디비전은 나누는 릴레이션에 있는 **모든 값과 대응하는** 나머지 속성값을 찾는다.
C에는 B# 값 b2와 b4가 있으므로 AB에서 b2 및 b4와 모두 짝을 이루는 A#를 구한다. 결과에는 C의 속성인 B#를 제외한 A#만 남는다.

a1 확인: AB에 (a1, b2)와 (a1, b4)가 모두 있으므로 a1은 결과에 포함된다. 추가로 b1과 연결되어 있어도 조건을 만족하는 데 지장이 없다.

a2 확인: (a2, b2)와 (a2, b4)가 모두 있으므로 a2도 포함된다. b1·b6과의 추가 대응은 배제 사유가 아니다. 디비전은 B# 집합이 C와 정확히 같을 것을 요구하는 것이 아니라 **C의 모든 값을 포함**할 것을 요구한다.

a3 확인: a3은 b3하고만 연결되어 있어 b2·b4와의 대응이 없다. 따라서 제외된다.

a4 확인: (a4, b2)와 (a4, b4)가 모두 존재하므로 a4는 포함된다.

결과는 A# 값 a1, a2, a4로 이루어진 3개의 투플이다. 따라서 결과 투플 수는 3으로 ③과 일치한다.

NoteBase

문 19. 관계형 데이터모델의 관계대수 연산자에 대한 설명으로 가장 적절하지 않은 것은?

① $\Pi_{attr}(R)$ 연산의 결과 투플 수는 항상 R의 투플 수보다 작거나 같다.

② $\sigma(R)$ 연산의 결과 릴레이션 차수는 릴레이션 R의 차수와 같다.

③ 합집합, 교집합, 차집합 연산은 합집합 호환성이 만족되어야 한다.

④ 합집합, 교집합, 차집합 연산은 결합법칙, 교환법칙이 성립한다.

정답 4번

간단한 해설

합집합과 교집합은 교환법칙과 결합법칙이 성립하지만, **차집합은 일반적으로 성립하지 않는다**. 차집합은 어느 집합에서 무엇을 빼는지와 연산을 묶는 순서에 따라 결과가 달라진다.

선지별 해설

① O: 프로젝션은 지정한 속성들만 남기는 연산이다. 관계대수에서는 릴레이션을 집합으로 다루므로, 일부 속성을 제거한 뒤 같아진 투플은 하나로 합쳐진다. 새 투플을 추가하지 않고 **중복이 생기면 제거**하므로 결과 투플 수는 원래 투플 수보다 작거나 같다.

② O: 셀렉션은 조건을 만족하는 투플을 선택하는 연산이며 속성을 제거하거나 추가하지 않는다. 따라서 결과의 투플 수는 줄어들 수 있지만, 속성의 개수인 **차수는 그대로 유지**된다.

③ O: 합집합·교집합·차집합은 두 릴레이션의 투플을 같은 구조에서 비교하는 연산이다. 따라서 두 릴레이션의 차수가 같고 대응하는 속성의 도메인이 같아야 하는 **합집합 호환성**이 요구된다.

④ X: 합집합과 교집합은 교환법칙과 결합법칙이 성립하지만, **차집합은 두 법칙이 일반적으로 성립하지 않는다**. 차집합은 앞 집합에 속하면서 뒤 집합에 속하지 않는 원소를 남기므로 순서가 중요하다. 예를 들어 $A = \{1\}$, $B = \{2\}$이면 $A - B = \{1\}, B - A = \{2\}$로 서로 다르다. 또한 $A = B = C = \{1\}$이면 $(A - B) - C = \emptyset$이지만 $A - (B - C) = \{1\}$이므로 묶는 순서에 따라서도 결과가 달라진다.

NoteBase

문 20. 다음 릴레이션 R에서 성립하는 함수 종속성으로 가장 적절하지 않은 것은?

R

A	B	C
b	c	h
e	i	f
g	i	f
e	b	a

① A → B

② B → C

③ AB → C

④ AC → B

정답 1번

간단한 해설

A값이 e인 두 투플에서 B값이 각각 i와 b로 다르다. 함수 종속성이 성립하려면 **같은 결정자 값에 같은 종속자 값**이 대응해야 하므로 A → B는 성립하지 않는다.

NoteBase

선지별 해설

함수 종속성 X → Y는 두 투플의 X값이 같으면 Y값도 같아야 한다는 조건이다. 이 문제에서는 제시된 릴레이션 인스턴스의 모든 행을 비교한다.

① X: A값이 e인 두 행은 각각 (e, i, f)와 (e, b, a)이다. **같은 A값 e에 서로 다른 B값 i와 b**가 대응하므로 A → B는 성립하지 않는다.

② O: B값 i가 두 번 나타나지만 두 행의 C값은 모두 f이다. 나머지 B값 c와 b도 각각 C값 h와 a에 하나씩 대응하므로 **같은 B값에 다른 C값이 나오는 반례가 없다**. 따라서 제시된 R에서 B → C가 성립한다.

③ O: AB는 A와 B를 함께 묶은 결정자이다. 각 행의 AB값은 (b, c), (e, i), (g, i), (e, b)로 모두 다르므로 같은 AB값에 서로 다른 C값이 대응하는 경우가 없다. 따라서 **A와 B를 함께 보면** C가 결정되어 AB → C가 성립한다. B → C가 성립한다는 사실로도 이를 확인할 수 있다.

④ O: 각 행의 AC값은 (b, h), (e, f), (g, f), (e, a)로 모두 다르다. A값 e는 중복되지만 C값이 f와 a로 구별되므로 **AC 조합이 같은 두 행에서 B가 달라지는 반례가 없다**. 따라서 제시된 R에서 AC → B가 성립한다.

NoteBase

문 21. 릴레이션 R(A, B, C)는 다음과 같은 함수 종속성이 성립한다. 릴레이션 R의 후보키는?

$AB \rightarrow C, \quad C \rightarrow A$

① A
② C
③ AB
④ AC

정답 3번

간단한 해설

AB는 $AB \rightarrow C$에 의해 모든 속성을 결정하고, A나 B를 하나라도 빼면 전체 속성을 결정하지 못한다. 따라서 **유일성과 최소성을 모두 만족하는 후보키**이다.

계산 과정

후보키는 모든 속성을 결정하는 유일성을 가지면서 어떤 속성도 더 뺄 수 없는 최소 속성집합이다. 함수 종속성을 반복 적용해 구한 속성집합의 폐포가 {A, B, C} 전체가 되는지 확인한다.

B는 두 함수 종속성의 오른쪽 어디에도 나타나지 않으므로, B를 포함하지 않은 속성집합에서는 B를 새로 얻을 수 없다. 따라서 **후보키에는 반드시 B가 포함**되어야 한다. 선택지의 A, C, AC는 이 조건을 만족하지 못한다.

AB에서 출발하면 A와 B가 이미 있고, $AB \rightarrow C$를 적용해 C를 얻는다. 따라서 $(AB)^+ = \{A, B, C\}$이므로 AB는 모든 속성을 결정하는 슈퍼키이다.

AB에서 B를 빼면 $A^+ = \{A\}$이고, A를 빼면 $B^+ = \{B\}$이다. 어느 쪽도 전체 속성을 결정하지 못하므로 **AB는 더 줄일 수 없어 최소성도 만족**한다.

따라서 제시된 선택지 중 후보키는 AB인 ③이다. BC도 $C \rightarrow A$에 의해 전체 속성을 결정하는 후보키이지만, 선택지에는 제시되어 있지 않다.

NoteBase

문 23. 데이터베이스를 파일에 저장하는 기법과 관련된 설명으로 가장 적절하지 않은 것은?

① 파일은 일반적으로 동일한 유형의 레코드들의 모임으로 이루어진다.

② DBMS는 디스크로부터 사용자가 원하는 데이터를 포함하고 있는 블록을 주기억장치로 가져온다.

③ 버퍼는 성능향상을 위해 자주 참조되는 디스크 블록들을 저장하는 데 사용되는 주기억장치 공간이다.

④ 한 파일에 속하는 블록들은 반드시 인접해 있어야 한다.

정답 4번

간단한 해설

파일을 구성하는 블록들은 논리적으로 연결되어 있으면 되므로 **디스크에서 반드시 인접할 필요는 없다**. 연결 정보나 인덱스를 이용하면 떨어져 있는 블록들도 하나의 파일로 관리할 수 있다.

선지별 해설

① O: 레코드는 하나의 대상에 관한 여러 필드의 모음이며, 파일은 일반적으로 **같은 유형의 레코드들을 모아** 저장한 단위이다. 예를 들어 직원 파일은 이름·직원번호 등의 공통 필드를 갖는 직원 레코드들의 모음으로 구성할 수 있다.

② O: 디스크와 주기억장치 사이의 입출력은 보통 블록 단위로 이루어진다. 따라서 필요한 데이터가 메모리에 없으면 DBMS는 그 데이터를 포함한 **블록을 주기억장치로 읽어** 처리한다. 이미 버퍼에 있는 블록이라면 디스크에서 다시 읽을 필요 없이 이용할 수 있다.

③ O: 버퍼는 디스크 블록을 일시적으로 보관하는 주기억장치 영역이다. 자주 사용하는 블록을 버퍼에 유지하면 같은 데이터를 이용할 때마다 디스크에 접근하는 일을 줄일 수 있어 **디스크 입출력 비용을 감소**시킨다.

④ X: 한 파일의 블록들이 논리적으로 같은 파일에 속한다는 사실이 디스크에서도 연속된 위치에 있어야 함을 뜻하지는 않는다. 연속 할당을 사용할 수 있지만 연결 정보나 인덱스를 통해 **서로 떨어진 블록들을 하나의 파일로 관리**할 수도 있다. 따라서 반드시 인접해야 한다는 설명은 잘못되었다.

NoteBase

문 24. 다음은 RAID에 대한 설명이다. 적절한 설명은 모두 몇 개인가?

가. 블록레벨 스트라이핑(striping)은 데이터 블록들을 비트별로 여러 디스크에 분산시킨다.
나. 신뢰성 향상을 위해 반사(mirroring), 패리티(parity) 기법을 사용한다.
다. 레벨 1은 디스크 오류 복구가 가장 쉽고 레벨 0은 가장 좋은 쓰기 성능을 갖는다.
라. 레벨 5는 데이터를 모든 디스크에 분산시키고, 패리티를 일부 디스크에 분산시킨다.

① 1개
② 2개
③ 3개
④ 4개

정답 2번

간단한 해설

블록레벨 스트라이핑의 분산 단위는 비트가 아니라 **블록**이다. RAID 5의 패리티도 일부 고정 디스크에만 두는 것이 아니라 **전체 디스크에 걸쳐 분산**하므로 두 설명을 구별해야 한다.

NoteBase

보기별 해설

가. X: 블록레벨 스트라이핑은 데이터를 **블록 단위**로 나누어 여러 디스크에 분산 저장한다. 비트별로 나누는 것은 비트레벨 스트라이핑이다. 스트라이핑이라는 공통점만으로 분산 단위를 같게 보면 안 된다.

나. O: 미러링은 같은 데이터를 다른 디스크에도 복제하고, 패리티는 원래 데이터를 복원할 수 있는 추가 정보를 저장한다. 둘 다 **중복 정보를 이용해 디스크 고장에 대비**하는 방법이다. 다만 RAID 0처럼 이러한 중복성을 제공하지 않는 레벨도 있다.

다. O: RAID 1은 정상 디스크에 동일한 데이터 사본이 있어 고장난 디스크의 내용을 복사하여 복구하기가 단순하다. RAID 0은 데이터를 분산하여 병렬로 쓰면서 **미러링의 중복 쓰기나 패리티 계산·갱신 부담이 없다**. 따라서 같은 종류의 디스크 등 조건을 같게 둔 전형적인 RAID 레벨 비교에서 쓰기 성능에 가장 유리한 것으로 설명한다. 실제 성능은 장치 수·컨트롤러·작업 부하에 따라 달라질 수 있으며, RAID 0 자체에는 디스크 고장 복구를 위한 중복 정보가 없다.

라. X: RAID 5는 블록 단위로 데이터를 분산하고 **패리티 블록도 전체 구성 디스크에 분산**한다. 한 스트라이프의 패리티는 그 스트라이프의 한 디스크에 놓이지만, 스트라이프가 달라지면 패리티를 맡는 디스크도 달라진다. 따라서 일부 고정 디스크에만 패리티를 배치한다고 이해하면 잘못이다.

적절한 보기는 나, 다의 2개이므로 ②와 일치한다.

NoteBase

문 25. 인덱스에 대한 설명으로 가장 적절하지 않은 것은?

① 클러스터링(clustering) 인덱스는 범위 질의에 유용하다.

② 보조(secondary) 인덱스는 탐색키 값에 따라 정렬되지 않은 데이터 파일에 대해 정의된다.

③ 밀집(dense) 인덱스는 모든 데이터 블록의 최대 탐색키 값들을 모아서 인덱스 엔트리로 유지하는 인덱스이다.

④ 클러스터링 인덱스는 희소 인덱스일 경우가 많다.

정답 3번

간단한 해설

밀집 인덱스는 **모든 탐색키 값**에 대한 엔트리를 갖는다. 블록별 대표키만 모아 유지하는 방식은 희소 인덱스에 해당하므로 두 개념을 바꾸어 설명한 부분이 잘못이다.

선지별 해설

① O: 클러스터링 인덱스는 데이터 레코드가 탐색키 순서에 따라 모여 저장되는 구조와 연결된다. 특정 구간에 속하는 값들이 인접한 블록에 모이므로 시작 위치를 찾은 뒤 연속적으로 읽기 쉬워 **범위 질의**에 유용하다.

② O: 보조 인덱스는 데이터 파일의 실제 정렬 순서와 다른 탐색키에 대해 추가로 만드는 접근 경로이다. 따라서 그 탐색키의 값 순서로 데이터 파일이 정렬되어 있지 않아도 레코드를 찾을 수 있도록 키와 위치 정보를 관리한다.

③ X: 밀집 인덱스는 데이터 파일에 존재하는 **모든 탐색키 값**에 대한 인덱스 엔트리를 유지한다. 같은 키 값의 레코드가 여러 개라면 그 레코드들을 찾을 수 있는 포인터 구조를 사용한다. 블록마다 최대값 등의 대표키만 골라 엔트리로 유지하는 방식은 전체 탐색키 값을 담는 밀집 인덱스가 아니라 **희소 인덱스**에 해당한다.

④ O: 데이터가 탐색키 순서대로 정렬되어 있으면 각 블록의 대표키 등 일부 키만으로 목표 블록을 찾아 그 안을 검색할 수 있다. 따라서 클러스터링 인덱스는 희소하게 구성되는 경우가 많으며, 모든 키의 엔트리를 두지 않아도 정렬된 데이터의 위치를 좁힐 수 있다.

NoteBase

문 26. 다음과 같이 인덱스가 생성되어 있을 때, 생성된 인덱스가 사용되기에 가장 적절하지 않은 질의는?

```
CREATE INDEX EmpIndex
        ON EMPLOYEE ( dno, salary ) ;
```

① SELECT * FROM EMPLOYEE WHERE salary = 5000000 ;

② SELECT * FROM EMPLOYEE WHERE dno = 3 AND salary = 5000000 ;

③ SELECT * FROM EMPLOYEE WHERE dno = 2 ;

④ SELECT * FROM EMPLOYEE WHERE dno >= 2 AND salary >= 4000000 AND salary <= 5000000 ;

정답 1번

간단한 해설

복합 인덱스 (dno, salary)는 **dno가 선두 열**이다. salary만 제한하는 질의는 선두 열의 정렬 구조로 검색 범위를 좁히기 어려워 이 인덱스 활용에 가장 불리하다.

NoteBase

선지별 해설

주어진 복합 인덱스의 열 순서는 (dno, salary)이다. 일반적인 B-tree 방식에서는 **dno를 먼저 정렬하고 같은 dno 안에서 salary를 정렬**하므로, 선두 열의 조건이 검색 범위를 좁히는 데 중요하다.

① salary = 5000000만 지정하고 **선두 열 dno의 조건이 없다**. 같은 급여 값이 여러 dno 구간에 흩어질 수 있어, 이 인덱스로 하나의 좁은 연속 구간을 바로 찾기 어렵다. 네 질의 중 일반적인 선두 열 활용 원칙에 가장 부합하지 않는다.

② dno = 3과 salary = 5000000을 모두 동등조건으로 지정한다. 선두 열 dno를 고정한 뒤 그 안에서 salary 값까지 찾으므로 **복합 키 전체를 이용해 범위를 좁힐 수 있다**.

③ dno = 2는 선두 열만 지정한 조건이다. salary 조건이 없어도 dno가 2인 연속된 인덱스 구간을 탐색할 수 있으므로 인덱스 활용에 적합하다.

④ dno >= 2는 선두 열의 범위조건이므로 해당 dno 범위의 인덱스를 탐색할 수 있다. 다만 dno가 여러 값에 걸치므로 뒤의 salary 범위조건이 ②와 같이 전체 검색 구간을 좁혀 주는 것은 아니며, 탐색한 항목을 추가로 걸러내는 데 활용될 수 있다.

따라서 선두 열 조건이 없는 ①이 가장 적절하지 않다. 실제 사용 여부는 데이터 분포와 DBMS의 최적화 방식에 따라 달라지므로, salary만으로 인덱스를 사용하는 실행계획이 모든 경우에 불가능하다는 뜻은 아니다.

NoteBase

문 27. 비용 기반(cost-based) 질의 최적화(query optimization) 방법을 위해 비용 산정 시 고려하는 구성요소로 가장 적절하지 않은 것은?

① 보조 기억 장치의 접근 비용

② 백업(backup)을 위한 저장 비용

③ 주기억 장치 연산을 수행하는 계산 비용

④ 질의 전송 및 결과 반환을 위한 통신 비용

정답 2번

간단한 해설

비용 기반 질의 최적화는 실행계획별 **입출력·계산·통신 비용**을 추정하여 비교한다. 백업용 저장 비용은 별도의 운영·복구 관리 비용으로, 일반적인 개별 질의 실행계획의 비교 요소가 아니다.

선지별 해설

① O: 질의를 실행하려면 테이블이나 인덱스의 블록을 읽고 필요한 경우 중간 결과를 디스크에 기록해야 한다. 따라서 보조기억장치에 대한 **입출력 횟수와 접근 비용**은 실행계획을 비교하는 주요 요소이다. 인덱스 사용 여부나 조인 방식에 따라 이 비용이 달라진다.

② X: 백업은 장애에 대비해 데이터를 별도로 보존하는 관리 작업이다. 백업용 저장공간의 비용은 데이터베이스 운영비용에 속하지만, 일반적인 비용 기반 질의 최적화에서 **개별 질의 실행계획을 비교하는 비용 요소는 아니다**. 질의 실행 중 임시 결과를 저장하는 비용과 백업 보존 비용을 구별해야 한다.

③ O: 메모리로 읽은 데이터에 대해 비교, 정렬, 해시 계산, 조건 검사 등을 수행할 때 처리 자원이 소요된다. 이러한 **CPU 연산 비용**도 실행계획의 비용을 구성한다. 디스크 입출력이 중요하더라도 계산 비용이 항상 무시되는 것은 아니다.

④ O: 특히 분산 데이터베이스에서는 질의를 다른 노드에 보내고 필요한 데이터나 결과를 전달하는 과정에서 통신이 발생한다. 전송량과 네트워크 지연 등이 실행시간에 영향을 미치므로 **통신 비용**도 고려할 수 있다. 단일 장치 안에서 처리되는 질의에서는 그 비중이 작거나 별도로 고려되지 않을 수 있다.

NoteBase

문 28. 동시성 제어가 제대로 이루어지지 않은 경우에 발생하는 문제로 가장 적절하지 않은 것은?

① 갱신 손실 문제가 발생할 수 있다.

② 오손 읽기(dirty read) 문제가 발생할 수 있다.

③ 연쇄 삭제 문제가 발생할 수 있다.

④ 부정확한 요약 문제가 발생할 수 있다.

정답 3번

간단한 해설

연쇄 삭제는 참조 무결성 규칙에 따라 관련 행을 함께 삭제하는 동작으로, 동시성 제어 실패의 대표적인 이상 현상이 아니다. 동시성 문제와 관련되는 것은 연쇄 삭제가 아니라 **미확정 데이터를 읽은 거래들의 연쇄 복귀**이다.

선지별 해설

동시성 제어는 여러 트랜잭션이 함께 실행되어도 서로의 읽기와 쓰기가 부적절하게 간섭하지 않도록 조정하는 기능이다.

① O: 갱신 손실은 여러 트랜잭션이 같은 데이터를 읽고 각각 갱신하면서 **나중의 쓰기가 앞선 갱신을 덮어쓰는 현상**이다. 예를 들어 두 거래가 같은 잔액을 기준으로 계산한 뒤 결과를 차례로 저장하면 한 거래의 증감분이 반영되지 않을 수 있다.

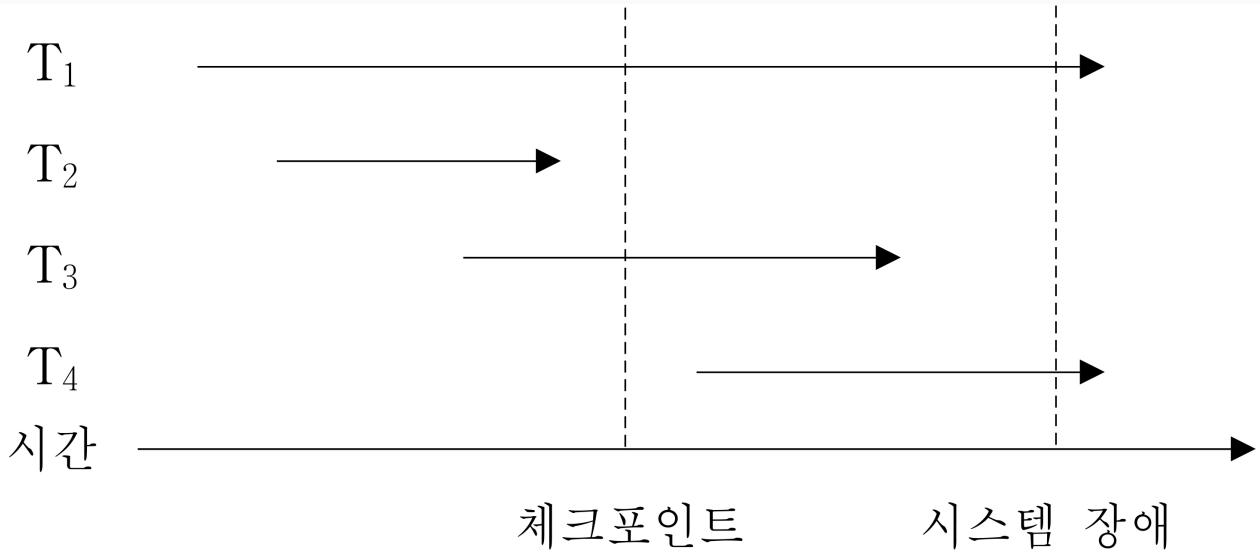
② O: 오손 읽기는 다른 트랜잭션이 변경했지만 **아직 커밋하지 않은 값**을 읽는 것이다. 그 트랜잭션이 나중에 롤백되면 실제로 확정되지 않은 값을 사용한 셈이 되어 일관성이 깨질 수 있다.

③ X: 연쇄 삭제는 부모 행이 삭제될 때 외래키로 참조하는 자식 행도 삭제하도록 정한 동작으로, 일반적으로 참조 무결성의 삭제 규칙과 관련된다. 동시성 제어 실패로 인한 이상 현상으로 구분하는 것은 아니다. 이와 혼동하기 쉬운 **연쇄 복귀**는 한 트랜잭션의 롤백 때문에 그 미확정 값을 읽은 다른 트랜잭션들도 롤백해야 하는 현상이다.

④ O: 부정확한 요약은 한 트랜잭션이 합계나 평균을 구하는 동안 다른 트랜잭션이 일부 데이터를 변경하여, **변경 전 값과 변경 후 값을 섞어 읽는 현상**이다. 이 때문에 계산된 요약값이 어느 한 일관된 시점의 데이터 상태도 반영하지 못할 수 있다.

NoteBase

문 29. 다음 그림은 트랜잭션(T□)의 로그 기록과 체크포인트 및 시스템 장애 시점을 나타낸다. 시스템 장애가 발생한 후 로그 기록에 따라 회복하는 방법 중 가장 적절하지 않은 것은?



① 즉시갱신 방법을 사용한 경우, T_1은 REDO가 필요하다.

② 즉시갱신 방법을 사용한 경우, T_2는 아무 작업이 필요없다.

③ 지연갱신 방법을 사용한 경우, T_3는 REDO가 필요하다.

④ 지연갱신 방법을 사용한 경우, T_4는 아무 작업이 필요없다.

정답 1번

간단한 해설

T_1은 장애가 발생했을 때 미완료 상태이다. 즉시갱신에서는 미완료 변경이 데이터베이스에 반영되었을 수 있으므로 **REDO가 아니라 UNDO**로 그 영향을 취소해야 한다.

NoteBase

선지별 해설

REDO는 완료된 트랜잭션의 갱신 결과를 다시 반영하는 작업이고, UNDO는 미완료 트랜잭션이 반영한 변경을 취소하는 작업이다. 그림에서는 T_2가 체크포인트 전에 완료되고, T_3는 체크포인트 후 장애 전에 완료되며, T_1·T_4는 장애 시점에 미완료 상태이다.

① X: T_1은 장애 시점까지 끝나지 않았다. 즉시갱신은 트랜잭션이 완료되기 전에도 데이터베이스에 변경을 반영할 수 있으므로, 미완료 T_1의 영향을 **UNDO로 취소**해야 한다. 완료되지 않은 결과를 REDO로 확정하여 남기는 것이 아니다.

② O: T_2는 체크포인트 전에 완료되었다. 문제에서 전제하는 체크포인트 회복 방식에서는 그 이전에 완료된 갱신 결과가 안정적으로 저장되어 있으므로 T_2에 대해 다시 REDO나 UNDO를 할 필요가 없다.

③ O: T_3는 체크포인트를 지난 뒤 장애 발생 전에 완료되었다. 지연갱신은 완료 전에는 실제 데이터베이스를 갱신하지 않으며, 완료된 뒤에도 변경 사항이 모두 반영되었는지 보장되지 않을 수 있다. 따라서 로그를 이용한 **REDO로 완료 결과를 재반영**한다.

④ O: T_4는 체크포인트 이후 시작했지만 장애 시점에 완료되지 않았다. 지연갱신 방식에서는 이 미완료 트랜잭션의 변경이 실제 데이터베이스에 반영되지 않았으므로 취소할 데이터 변경이 없다. 미완료 결과를 REDO하지도 않으므로 별도의 회복 작업이 필요하지 않다.

NoteBase

문 30. 데이터베이스 로그(log)에 대한 설명으로 가장 적절한 것은?

① 트랜잭션의 동시성 제어를 위해 작성하는 파일이다.

② 로그 기록 시 변경 후의 값만을 저장한다.

③ 트랜잭션 수행이 종료될 때만 기록한다.

④ 트랜잭션의 종료 또는 중단 여부를 판단할 수 있다.

정답 4번

간단한 해설

로그에는 트랜잭션의 시작·갱신뿐 아니라 **커밋이나 중단에 관한 기록**도 남으므로 정상 완료 또는 중단 여부를 판단할 수 있다.

선지별 해설

① X: 로그의 기본 목적은 트랜잭션의 수행과 데이터 변경을 기록하여 **장애가 발생했을 때 복구**하는 것이다. 여러 트랜잭션의 충돌을 조절하는 동시성 제어는 잠금, 타임스탬프 등의 기법으로 수행하므로 로그를 단순히 동시성 제어용 파일로 설명하는 것은 부적절하다.

② X: 로그에 변경 후 값만 저장한다고 일반화할 수 없다. 복구 방식에 따라 **변경 전 값은 UNDO, 변경 후 값은 REDO**에 이용되며, 갱신 전후의 값을 모두 기록할 수 있다. 필요한 로그 내용은 사용하는 복구 방식에 따라 달라진다.

③ X: 로그는 트랜잭션이 종료될 때만 기록하는 것이 아니다. **시작, 데이터 갱신, 커밋, 중단** 등의 수행 과정에 관한 기록이 남아야 장애 시 어느 작업까지 진행되었는지 판단할 수 있다.

④ O: 로그의 커밋 기록과 중단 기록 등을 확인하면 트랜잭션이 정상 완료되었는지 중단되었는지를 파악할 수 있다. 장애 시에는 이러한 기록을 바탕으로 완료된 트랜잭션의 결과를 재실행하거나 미완료 트랜잭션의 변경을 취소하는 복구 처리를 결정한다.

NoteBase

문 31. NoSQL은 성능과 모델링 파워의 유연성 및 복잡한 질의 기능을 강조하고 있다. 다음 중 NoSQL의 특징으로 가장 적절하지 않은 것은?

① 빅데이터를 효율적으로 처리하기 위해 등장한 데이터 저장 기술이다.

② NoSQL은 하나의 데이터베이스나 네트워크 시스템을 여러개의 작은 조각으로 나누어 관리하는 기술을 지원한다.

③ NoSQL은 RDBMS보다 엄격한 일관성 모델을 보장한다.

④ NoSQL 데이터 모델로 키-값, 문서기반, 그래프 기반 등의 모델이 있다.

정답 3번

간단한 해설

NoSQL은 확장성과 유연한 데이터 모델을 중시하는 여러 시스템을 포괄하며, RDBMS보다 **더 엄격한 일관성을 일률적으로 보장하지 않는다**. 최종적 일관성을 활용하는 경우도 있고 강한 일관성을 제공하는 경우도 있어 제품과 설정을 구분해야 한다.

선지별 해설

① O: NoSQL은 대규모 데이터와 다양한 형식의 데이터를 효율적으로 저장·처리하려는 요구 속에서 발전하였다. 고정된 관계형 테이블만을 전제로 하지 않는 모델과 **분산 저장·수평 확장**을 활용하여 데이터량과 처리량 증가에 대응하는 것이 중요한 특징이다.

② O: 큰 데이터 집합을 작은 조각인 샤드로 나누고 여러 서버에 분산하는 **샤딩**을 설명한 것이다. 여러 NoSQL 시스템은 이를 통해 데이터와 요청 부하를 분산한다. 이때 데이터베이스의 논리적 데이터를 분할하는 것이 핵심이며 네트워크 회선을 단순히 잘게 나누는 기술이라는 뜻은 아니다.

③ X: NoSQL이라는 이유만으로 RDBMS보다 **더 엄격한 일관성을 보장하는 것은 아니다**. 분산 환경에서 가용성·확장성을 중시하여 복제본의 상태가 시간이 지나 수렴하는 최종적 일관성을 활용하는 시스템도 있다. 다만 모든 NoSQL이 일관성을 약하게만 보장하는 것도 아니며, 실제 보장 수준은 제품과 설정·연산에 따라 달라진다.

④ O: 키-값 모델은 키를 이용해 대응 값을 찾고, 문서 모델은 문서 단위로 데이터를 저장하며, 그래프 모델은 노드와 연결 관계를 중심으로 표현한다. 이 밖에 넓은 열 모델도 있으며, NoSQL은 하나의 동일한 저장 구조가 아니라 **여러 비관계형 데이터 모델**을 포괄한다.

NoteBase

문 32. 데이터 웨어하우스의 특징에 대한 설명으로 가장 적절하지 않은 것은?

① 누구나 쉽게 이해하도록 데이터를 고객, 제품 등과 같이 주제 중심으로 분류 가능하다.

② 과거와 현재의 경향에 대한 분석 및 예측을 지원하기 위한 데이터 공간이다.

③ 일관성 있는 데이터 정의, 관계성, 키 구조 등 하나의 의미로 통합 저장한다.

④ 일반적으로 데이터 추가, 삭제, 갱신이 매우 빈번하게 이루어지므로 쓰기 전용 형태로 저장한다.

정답 4번

간단한 해설

데이터 웨어하우스는 분석을 위해 이력 데이터를 보존하는 **비휘발성·읽기 중심**의 저장소이다. 운영 데이터베이스처럼 데이터를 빈번하게 수정·삭제하는 쓰기 전용 저장소가 아니다.

선지별 해설

데이터 웨어하우스는 여러 원천의 데이터를 통합하여 조직의 분석과 의사결정을 지원하는 저장소로, 주제지향성·통합성·시계열성·비휘발성을 대표적 특징으로 가진다.

① O: 데이터 웨어하우스는 개별 업무 처리 절차보다 고객·제품·매출 등 **분석할 주제 중심**으로 데이터를 구성한다. 여러 업무에 흩어진 자료를 같은 주제에서 살펴볼 수 있어 분석에 적합하다.

② O: 데이터 웨어하우스는 과거 자료를 현재 자료와 함께 보존하여 기간별 변화나 추세를 분석할 수 있도록 한다. 이러한 **시계열성**은 과거와 현재의 비교 및 미래 예측을 위한 자료 기반을 제공한다.

③ O: 서로 다른 시스템에서 가져온 데이터는 이름, 코드, 단위, 키 구조 등이 다를 수 있다. 데이터 웨어하우스는 이 차이를 조정하여 **일관된 정의와 의미로 통합**하므로 서로 비교하고 결합할 수 있다.

④ X: 데이터 웨어하우스는 일상적인 업무 트랜잭션마다 데이터를 수정·삭제하는 저장소가 아니라 **이력을 안정적으로 유지하며 조회·분석하는 저장소**이다. 새 데이터를 적재하거나 관리상 정정하는 작업은 있을 수 있지만, 이를 빈번한 갱신 중심이나 쓰기 전용이라고 설명할 수는 없다.

NoteBase

문 33. OLAP 연산기능 중 요약된 형태의 데이터에서 상세 데이터로 접근하는 기능으로 가장 적절한 것은?

① Drill-down
② Pivoting
③ Roll-up
④ Slicing

정답 1번

간단한 해설

Drill-down은 연도별 자료에서 월별 자료로 내려가듯 **요약 수준을 낮추어 더 상세한 데이터에 접근**하는 연산이다.

개념 구분

OLAP은 다차원 데이터를 여러 관점과 집계 수준에서 분석한다. 이 문제의 판단 기준은 데이터를 표시하는 방향이 아니라 **요약 수준에서 상세 수준으로 내려가는가**이다.

Drill-down은 계층을 따라 더 세부적인 수준의 데이터로 들어가는 연산이다. 예를 들어 연도별 매출에서 분기별·월별 매출로 내려가 세부 내용을 확인한다.

Roll-up은 반대로 월별 매출을 분기별·연도별로 묶는 것처럼 **상위 수준으로 집계**한다. 따라서 문제에서 요구한 접근 방향과 반대이다.

Pivoting은 행·열이나 분석 축을 바꾸어 같은 데이터를 다른 관점으로 보는 연산이다.

Slicing은 특정 연도처럼 한 차원의 값을 고정해 다차원 데이터의 단면을 선택하는 연산이다. 두 연산 모두 요약 자료를 더 세부적인 계층으로 내려가는 기능 자체를 뜻하지 않는다.

따라서 요약된 형태에서 상세 데이터로 접근하는 기능은 Drill-down인 ①이다.

NoteBase

문 34. 다음과 같은 권한 부여 및 권한 취소와 관련된 SQL 명령어를 순서대로 실행한 후, 각 사용자의 권한과 관련한 설명으로 가장 적절하지 않은 것은?

```
GRANT insert, delete ON
EMPLOYEE, DEPARTMENT TO Kim ;

GRANT select ON EMPLOYEE, DEPARTMENT
TO Lee WITH GRANT OPTION ;

GRANT select ON EMPLOYEE TO Hong ;

REVOKE select ON EMPLOYEE FROM Lee ;

GRANT update ON EMPLOYEE TO Hong ;
```

① Kim은 EMPLOYEE, DEPARTMENT 릴레이션에서 검색할 수 있는 권한을 지니고 있다.

② Lee는 EMPLOYEE의 검색 권한을 지니고 있지 않다.

③ Lee는 DEPARTMENT 릴레이션만을 검색할 수 있는 권한을 지니고 있다.

④ Hong은 EMPLOYEE 릴레이션의 검색 및 갱신의 권한을 지니고 있다.

정답 1번

간단한 해설

Kim에게 부여된 **INSERT·DELETE는 SELECT 권한을 포함하지 않는다**. 따라서 제시된 명령만으로 Kim이 EMPLOYEE와 DEPARTMENT를 검색할 수 있다고 볼 수 없다.

NoteBase

선지별 해설

제시된 명령에 따른 권한의 변화를 추적하며, 별도로 주어지지 않은 기존 권한이나 재부여 사실은 가정하지 않는다.

① X: Kim에게 부여된 것은 EMPLOYEE와 DEPARTMENT에 대한 **INSERT와 DELETE 권한**이다. INSERT는 행 삽입, DELETE는 행 삭제를 허용하며, SELECT로 검색할 권한을 자동으로 포함하지 않는다. 제시된 명령에는 Kim에게 SELECT를 부여하는 문장이 없으므로 두 릴레이션을 검색할 권한이 있다는 설명은 잘못되었다.

② O: Lee는 두 릴레이션에 대한 SELECT 권한을 받았지만, 이후 'REVOKE select ON EMPLOYEE FROM Lee'가 실행되어 **EMPLOYEE의 검색 권한이 취소**되었다. 함께 부여되었던 WITH GRANT OPTION은 권한을 다른 사용자에게 다시 부여할 수 있게 하는 옵션이지, 본인의 권한 취소를 막는 옵션이 아니다.

③ O: 취소 명령의 대상은 EMPLOYEE에 대한 SELECT뿐이다. 따라서 Lee가 처음 부여받은 **DEPARTMENT의 SELECT 권한은 유지**된다. 제시된 두 릴레이션 가운데 최종적으로 Lee가 검색할 수 있는 것은 DEPARTMENT이다.

④ O: Hong은 'GRANT select ON EMPLOYEE TO Hong'으로 EMPLOYEE의 검색 권한을 받았고, 마지막에 UPDATE 권한도 받았다. Lee의 권한을 취소하는 문장은 Hong의 권한을 취소하지 않으므로 **SELECT와 UPDATE가 모두 유지**된다. 주어진 명령만으로 Hong이 Lee에게서 권한을 재부여받았다고 가정하지 않는다.

NoteBase

문 35. 다음 빅데이터에 대한 설명으로 가장 적절하지 않은 것은?

① 빅데이터는 수치, 문자, 영상 등 형태가 다양하며 방대한 규모의 정형 및 비정형 데이터를 포함한다.

② 빅데이터는 데이터로부터 가치를 추출하고 그 결과를 분석하는 기술을 의미하기도 한다.

③ 빅데이터의 3V 특징은 가시성(visibility), 속도(velocity), 다양성(variety)이다.

④ 빅데이터 처리과정은 크게 수집, 저장, 처리, 분석, 표현(시각화), 이용, 폐기 순으로 수행된다.

정답 3번

간단한 해설

빅데이터의 기본적인 3V는 **규모(Volume), 속도(Velocity), 다양성(Variety)**이다. 가시성(Visibility)은 기본 3V에 해당하지 않는다.

선지별 해설

① O: 빅데이터에는 정해진 표 구조의 정형 데이터뿐 아니라 문서·영상·음성 같은 비정형 데이터도 포함된다. 수치·문자·이미지 등 **형식이 다양하고 규모가 크다**는 특징 때문에 저장·처리 방식에서도 기존 방식과 다른 요구가 생긴다.

② O: 빅데이터라는 용어는 데이터 자체만을 가리키기도 하지만, 넓게는 대량의 데이터를 수집·저장·처리·분석하여 **유용한 정보와 가치를 도출하는 기술과 활동**을 포함한다. 따라서 가치 추출과 결과 분석 기술을 의미한다는 설명도 적절하다.

③ X: 기본적인 3V는 **규모(Volume)·속도(Velocity)·다양성(Variety)**이다. 규모는 데이터의 양, 속도는 생성·유입과 처리의 빠르기, 다양성은 데이터 형식과 출처의 다채로움을 뜻한다. 가시성(Visibility)은 여기서 규모를 잘못 바꾼 표현이다.

④ O: 데이터의 일반적인 수명주기는 수집한 자료를 저장하고 처리·분석한 뒤, 결과를 시각화하여 이해하고 이용하며 더 이상 필요하지 않으면 폐기하는 흐름으로 설명할 수 있다. 각 단계는 **데이터 확보 → 의미 도출 → 활용 → 종료**로 이어진다. 실제 시스템에서는 저장·처리·분석이 반복되거나 실시간으로 겹칠 수 있으므로 제시된 순서를 모든 구현의 절대적인 단일 절차로 이해할 필요는 없다.

NoteBase

문 36. 데이터마이닝(data mining)의 연관규칙을 이용하여 다음과 같은 구매 품목들을 분석하였다. 빵과 우유 사이의 연관규칙에 대한 신뢰도(confidence)와 지지도(support) 계산 결과로 가장 적절한 것은?

트랜잭션 ID	구매품목
1	기저귀, 빵, 우유
2	기저귀, 빵, 소고기, 우유
3	사이다, 콜라, 소고기, 맥주
4	콜라, 빵, 우유
5	빵, 콜라, 우유
6	맥주, 사이다, 빵

① '우유 → 빵'에 대한 신뢰도는 4/6이고 지지도는 4/5이다.

② '우유 → 빵'에 대한 신뢰도는 4/5이고 지지도는 4/6이다.

③ '빵 → 우유'에 대한 신뢰도는 4/5이고 지지도는 4/6이다.

④ '빵 → 우유'에 대한 신뢰도는 4/4이고 지지도는 5/6이다.

정답 3번

간단한 해설

신뢰도의 분모는 선행 품목의 구매 건수, 지지도의 분모는 전체 거래 건수이다. 빵 구매는 5건, 빵과 우유의 동시 구매는 4건, 전체 거래는 6건이므로 두 비율을 구분해 계산한다.

NoteBase

계산 과정

전체 거래는 6건이다. 빵을 구매한 거래는 1·2·4·5·6번의 5건, 우유를 구매한 거래는 1·2·4·5번의 4건이다. 빵과 우유를 함께 구매한 거래도 1·2·4·5번의 4건이다.

신뢰도는 선행 품목을 구매한 거래 중 후행 품목도 구매한 비율이다. 따라서 '빵 → 우유'의 신뢰도는 $\frac{4}{5}$, '우유 → 빵'의 신뢰도는 $\frac{4}{4} = 1$이다. 방향을 바꾸면 분모가 빵 구매 건수에서 우유 구매 건수로 달라진다.

지지도는 전체 거래 중 두 품목을 함께 구매한 비율이므로 $\frac{4}{6} = \frac{2}{3}$이다. 두 품목의 동시 구매 건수와 전체 거래 건수는 규칙의 방향을 바꾸어도 같으므로 지지도도 같다.

따라서 '빵 → 우유'의 신뢰도는 $\frac{4}{5}$, 지지도는 $\frac{4}{6}$인 ③이 적절하다.

①은 신뢰도와 지지도의 분모를 잘못 적용했고,

②는 '우유 → 빵'의 신뢰도를 잘못 계산했다.

④의 신뢰도는 역방향 규칙의 값이며, 지지도도 동시 구매가 아닌 빵 구매 건수를 분자로 삼아 잘못 계산했다.

NoteBase

문 37. 다음 로크 호환성 테이블은 표준 로킹 기법에서 트랜잭션 T_1이 항목 x에 대해 공유 로크(shared lock) 또는 배타 로크(exclusive lock)를 획득하였을 때, 다른 트랜잭션 T_2가 해당 로크를 요청할 때 획득 가능 여부를 나타낸다. 이때, 빈칸 ㉠~㉣에 가장 적절한 것은?

트랜잭션 T_2 \ 트랜잭션 T_1	공유 로크	배타 로크
공유 로크	㉠	㉡
배타 로크	㉢	㉣

① ㉠: 예, ㉡: 예, ㉢: 예, ㉣: 예

② ㉠: 예, ㉡: 예, ㉢: 예, ㉣: 아니오

③ ㉠: 예, ㉡: 예, ㉢: 아니오, ㉣: 아니오

④ ㉠: 예, ㉡: 아니오, ㉢: 아니오, ㉣: 아니오

정답 4번

간단한 해설

표준 공유·배타 로킹에서는 **공유 로크끼리만 동시에 허용**된다. 같은 데이터에 대한 조합에 배타 로크가 하나라도 포함되면 다른 트랜잭션의 로크 요청과 충돌한다.

NoteBase

도식 해설

표의 열은 먼저 로크를 획득한 T_1의 로크 종류이고, 행은 T_2가 새로 요청한 로크 종류이다. 공유 로크는 읽기용으로 여러 트랜잭션이 함께 보유할 수 있지만, 배타 로크는 다른 트랜잭션의 공유·배타 로크와 모두 충돌한다.

㉠ 예: T_1이 공유 로크를 보유하고 T_2도 공유 로크를 요청한다. 두 트랜잭션이 같은 항목을 읽는 것은 서로의 데이터를 변경하지 않으므로 **공유 로크끼리는 호환**된다.

㉡ 아니오: T_1이 배타 로크를 보유한 상태에서 T_2가 공유 로크를 요청한다. 변경 중인 항목을 다른 트랜잭션이 읽지 못하도록 해야 하므로 공유 로크를 즉시 부여할 수 없다.

㉢ 아니오: T_1이 공유 로크를 보유한 상태에서 T_2가 배타 로크를 요청한다. 다른 트랜잭션이 읽고 있는 항목을 변경하면 간섭이 생기므로 배타 로크를 즉시 부여할 수 없다.

㉣ 아니오: T_1과 T_2가 같은 항목의 배타 로크를 동시에 보유할 수는 없다. **배타 로크는 다른 트랜잭션의 모든 로크와 비호환**이므로 기존 로크가 해제될 때까지 기다려야 한다.

따라서 ㉠만 '예'이고 ㉡·㉢·㉣은 '아니오'이므로 ④에 해당한다.

NoteBase

문 38. 아래와 같이 트랜잭션 T_1, T_2, T_3가 병행 수행되는 스케줄을 가진 경우, 직렬 불가능 스케줄은?

①

시간 \ TID	T_1	T_2	T_3
1	read(x)		
2	write(x)		
3			read(y)
4			write(y)
5		read(x)	
6		write(x)	
7	read(y)		
8	write(y)		

②

시간 \ TID	T_1	T_2	T_3
1	read(x)		
2	write(x)		
3	read(y)		
4		read(x)	
5		write(x)	
6	write(y)		
7			read(y)
8			write(y)

③

시간 \ TID	T_1	T_2	T_3
1	read(x)		
2	write(x)		
3			read(y)
4		read(x)	
5			write(y)
6	read(y)		
7		write(x)	
8	write(y)		

④

시간 \ TID	T_1	T_2	T_3
1	read(x)		
2			read(y)
3			write(y)
4		read(x)	
5		write(x)	
6	write(x)		
7	read(y)		
8	write(y)		

정답 4번

간단한 해설

정답은 ④이다. ④에서는 x에 대한 충돌 때문에 $T_1 \rightarrow T_2$와 $T_2 \rightarrow T_1$이 동시에 요구된다. **선행 그래프에 순환이 생겨** 직렬화할 수 없다.

NoteBase

선지별 해설

서로 다른 트랜잭션이 같은 데이터에 접근하고 그중 하나 이상이 쓰기이면 두 연산은 충돌한다. 먼저 실행된 연산의 트랜잭션에서 나중 트랜잭션으로 간선을 그린 선행 그래프에 **순환이 없으면 충돌 직렬 가능**하다.

① (O) x의 충돌 순서는 $T_1 \rightarrow T_2$, y의 충돌 순서는 $T_3 \rightarrow T_1$이다. 순환이 없으므로 $T_3 \rightarrow T_1 \rightarrow T_2$ 순서의 직렬 스케줄과 동등하다.

② (O) x에서 $T_1 \rightarrow T_2$, y에서 $T_1 \rightarrow T_3$가 성립한다. T_2와 T_3 사이에는 충돌이 없어 순환이 없으며, T_1을 먼저 실행한 뒤 나머지 두 트랜잭션을 어느 순서로 실행해도 된다.

③ (O) 연산이 서로 끼어 있어도 x에서는 $T_1 \rightarrow T_2$, y에서는 $T_3 \rightarrow T_1$만 요구된다. 따라서 $T_3 \rightarrow T_1 \rightarrow T_2$ 순서로 직렬화할 수 있다.

④ (X) T_1의 시점 1 read(x)가 T_2의 시점 5 write(x)보다 앞서므로 $T_1 \rightarrow T_2$가 생긴다. 반대로 T_2의 시점 4 read(x)와 시점 5 write(x)가 T_1의 시점 6 write(x)보다 앞서므로 $T_2 \rightarrow T_1$도 생긴다. **두 트랜잭션 사이에 순환이 발생하므로 직렬 불가능**하다. y에서 생기는 $T_3 \rightarrow T_1$ 간선으로도 이 순환은 해소되지 않는다.

NoteBase

문 39. 다음 설명에 해당하는 데이터마이닝 기법으로 가장 적절한 것은?

· 일정한 집단에 대한 특정 정의를 통해 추론한다.
· 과거 데이터를 이용하여 모델을 생성하고 새로운 데이터에 대한 값을 예측한다.
· 지도학습 형식을 지닌다.

① 군집화(clustering)
② 분류(classification)
③ 연관규칙(association rule)
④ 유사도 검색(similarity search)

정답 2번

간단한 해설

분류는 소속 집단이 알려진 과거 데이터로 모델을 학습한 뒤, 새 데이터가 어느 집단에 속하는지 예측하는 **지도학습** 기법이다.

공통 해설

분류(classification)는 **미리 정해진 범주와 그 범주가 표시된 학습 데이터**를 이용해 새 데이터의 소속 범주를 예측하는 기법이다. 입력의 특성과 범주 사이의 관계를 모델로 학습하므로 지도학습에 해당한다. 예를 들어 스팸 여부가 표시된 과거 이메일로 모델을 학습한 뒤, 새 이메일을 스팸 또는 정상으로 판별한다.

제시문의 '일정한 집단'은 예측할 범주에, '과거 데이터를 이용하여 모델을 생성'한다는 것은 범주가 알려진 자료로 학습하는 과정에 대응한다. 여기서 새 데이터에 대해 예측하는 값은 연속적인 수치가 아니라 **소속 범주**이다. 군집화처럼 범주 표시 없이 유사한 자료를 묶거나, 연관규칙을 찾거나, 유사한 자료를 검색하는 것이 아니라, 이미 정해진 범주에 새 자료를 배정하는 분류를 설명한다.

NoteBase

문 40. 하둡 분산 파일 시스템(HDFS, Hadoop Distributed File System) 개발 목적으로 가장 적절하지 않은 것은?

① 하드웨어의 오류 자동 감지 및 회복

② 대화형 실시간 처리

③ 대용량 데이터 집합

④ 간단한 일관성 모델

정답 2번

간단한 해설

HDFS는 대화형 서비스의 짧은 응답시간보다 **대용량 데이터의 높은 처리량**을 중시한다. 따라서 대화형 실시간 처리보다 배치·연속적 데이터 접근을 위한 분산 파일 시스템으로 이해해야 한다.

선지별 해설

① O: 여러 장치로 구성된 대규모 분산 환경에서는 일부 하드웨어의 고장을 예외가 아닌 예상 가능한 상황으로 본다. HDFS는 장애를 감지하고 필요한 데이터 복제 등을 수행하는 **자동 회복과 내결함성**을 주요 설계 목표로 삼는다.

② X: HDFS는 사용자 요청에 즉시 응답하는 대화형 저지연 처리보다 **대용량 데이터의 높은 처리량**을 우선한다. 대량 데이터를 연속적으로 읽는 배치 처리에 적합하도록 설계되었으므로 대화형 실시간 처리를 핵심 개발 목적으로 든 설명은 부적절하다.

③ O: HDFS는 큰 파일과 대규모 데이터 집합을 여러 노드에 나누어 저장하고 접근하도록 설계되었다. **데이터 규모와 저장 노드 수의 확장**을 지원하는 것이 주요 목적이다.

④ O: 파일을 한 번 기록한 뒤 여러 번 읽는 접근 형태를 중심으로 하여 **일관성 관리의 복잡성을 줄인다**. 파일의 임의 위치를 빈번히 수정하는 범용 접근보다 단순한 모델을 사용해 처리량을 높인다. 추가 쓰기 등을 지원한다는 사실과 모든 위치의 자유로운 갱신을 기본으로 하지 않는다는 점은 구별해야 한다.

NoteBase