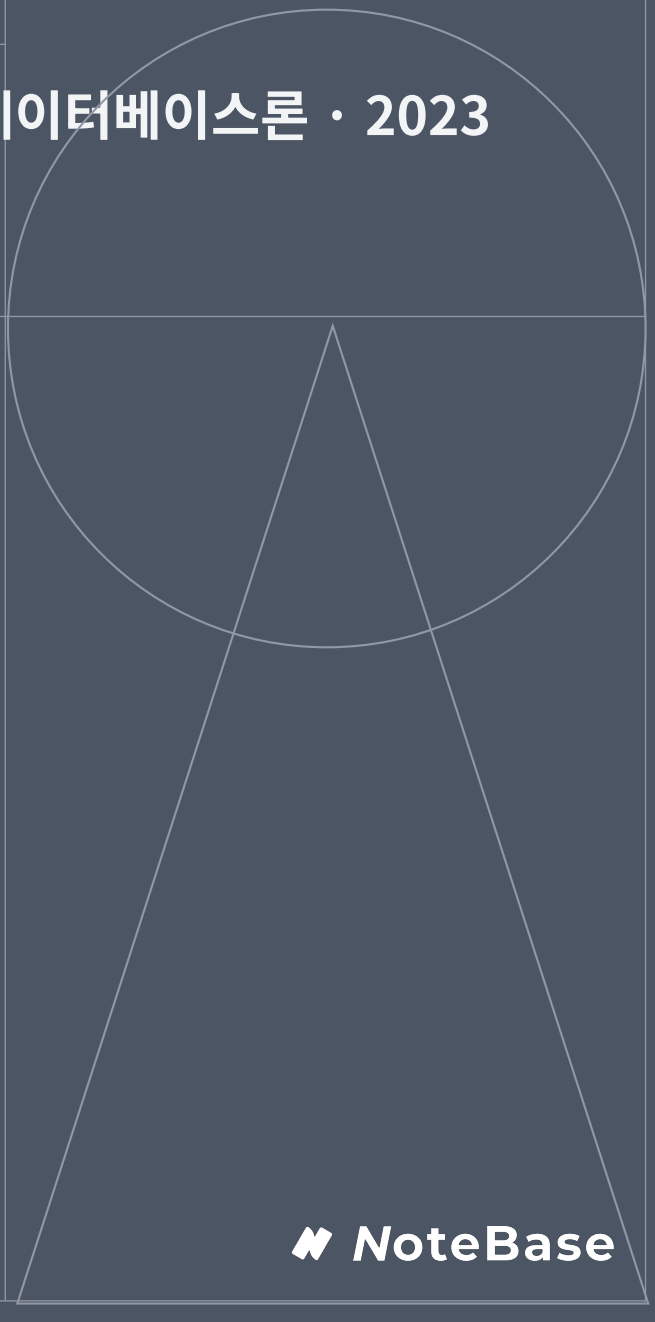
기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2023



NoteBase

기출OX 해설집

공무원/경찰/공채(경위) · 데이터베이스론 · 2023

공무원/경찰/공채(경위) · 데이터베이스론 · 2023

문 1. 데이터베이스의 논리적 설계 단계에서 수행하는 작업에 대한 설명으로 적절하지 않은 것은?

① 사용하려는 특정 목표 DBMS가 처리할 수 있는 논리적 스키마를 생성한다.

② 응용 프로그램 인터페이스를 설계한다.

③ E-R 다이어그램을 사용하여 DBMS에 독립적이고 고차원적인 표현 기법으로 기술한다.

④ 설계된 스키마를 정량적 정보와 성능평가 기준에 따라 평가하고 정제 등의 작업을 수행한다.

정답 3번

간단한 해설

E-R 다이어그램으로 현실의 개체와 관계를 DBMS에 독립적으로 표현하는 것은 **개념적 설계**이다. 논리적 설계는 그 결과를 목표 DBMS의 데이터 모델에 맞는 스키마로 바꾸는 단계이다.

선지별 해설

① O: 논리적 설계는 개념적 스키마를 **목표 DBMS가 지원하는 데이터 모델**로 변환한다. 관계형 DBMS라면 개체와 관계를 테이블로 대응시키고 속성·키·제약조건 등을 정한다. 디스크상의 실제 배치 같은 물리적 저장 세부사항을 결정하는 단계와는 구별된다.

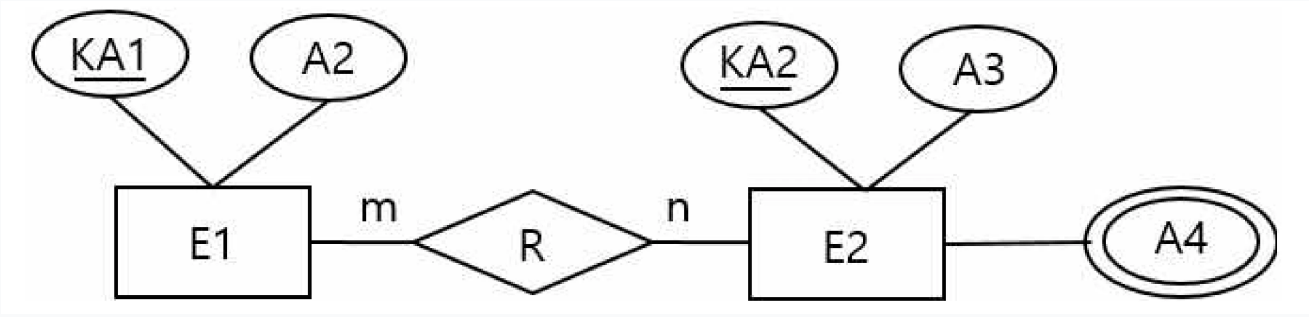
② O: 논리적 설계에서는 응용 프로그램의 트랜잭션이 어떤 데이터를 입력받아 조회·갱신하고 어떤 결과를 반환할지 등 **트랜잭션의 골격과 인터페이스**를 명세한다. 여기서 인터페이스 설계는 데이터베이스와 응용 프로그램 사이의 데이터 접근·입출력 관계를 정하는 것이며, 단순히 화면의 모양을 꾸미는 작업을 뜻하지 않는다.

③ X: E-R 다이어그램을 이용하여 개체·속성·관계를 특정 DBMS에 독립적인 고수준 모델로 나타내는 것은 **개념적 설계**에 해당한다. 논리적 설계에서는 이 개념적 모델을 목표 데이터 모델에 맞게 변환하여 논리적 스키마를 만든다.

④ O: 논리적 설계에서도 데이터량, 트랜잭션 처리빈도와 작업량 등 정량적 정보를 이용하여 스키마를 평가하고 정제한다. 예를 들어 자주 수행되는 작업이 접근해야 하는 논리적 레코드 수나 데이터 전송량을 검토하여 구조를 개선한다. **성능을 고려한다는 이유만으로 물리적 설계가 되는 것은 아니며**, 구체적인 파일 조직·인덱스·저장 배치의 결정은 물리적 설계에 해당한다.

NoteBase

문 2. 다음 E-R 다이어그램을 관계 데이터베이스 스키마로 변환할 때 가장 적절한 것은? (단, 밑줄은 기본키이다.)



① E1(KA1, A2), E2(KA2, A3, KA1), E21(KA2, A4)

② E1(KA1, A2, KA2), E2(KA2, A3), E21(KA2, A4)

③ E1(KA1, A2), E2(KA2, A3), E21(KA2, A4), R(KA1, KA2)

④ E1(KA1, A2, KA2), E2(KA2, A3, A4)

정답 3번

간단한 해설

E1과 E2 사이의 R은 **m:n 관계**이므로 별도 릴레이션으로 변환한다. E2의 A4는 이중 타원으로 표시된 **다중값 속성**이므로, KA2와 A4를 함께 기본키로 하는 별도 릴레이션이 필요하다.

NoteBase

도식 해설

개체 E1과 E2는 각각 E1(KA1, A2), E2(KA2, A3)로 변환한다. 그림에서 밑줄이 있는 KA1과 KA2는 각 개체의 기본키이다. A4는 다중값 속성이므로 E2의 일반 속성과 같은 방식으로 넣지 않는다.

관계 R의 양쪽에 m과 n이 표시되어 있으므로 E1의 한 개체가 여러 E2 개체와, E2의 한 개체가 여러 E1 개체와 연결될 수 있다. 이를 R(KA1, KA2)로 따로 만들고 **(KA1, KA2)를 복합 기본키**로 정한다. 각 속성은 각각 E1과 E2를 참조하는 외래키이기도 하다. 한쪽 릴레이션에 상대의 키 하나만 추가해서는 이 다대다 대응을 모두 표현할 수 없다.

A4의 이중 타원은 한 E2 개체가 여러 A4 값을 가질 수 있음을 뜻한다. 따라서 E21(KA2, A4)를 만들고 **(KA2, A4)를 복합 기본키**로 사용한다. KA2만 기본키로 삼으면 같은 E2 개체의 서로 다른 A4 값을 여러 행에 저장할 수 없기 때문이다. E21의 KA2는 E2를 참조하는 외래키이다.

①과 ②는 다대다 관계 R을 별도로 표현하지 않고 한쪽 개체 릴레이션에 외래키를 넣어 관계를 축소한다. 또한 ①의 E21은 KA2만 기본키이므로 A4의 다중값을 표현하지 못한다. ②는 E21의 복합 기본키는 맞지만 R의 변환이 잘못되었다.

④는 R을 E1의 KA2 하나로 표현하고, 다중값 A4를 E2의 일반 속성처럼 포함시킨다. 관계의 다중성과 A4의 여러 값을 모두 제대로 반영하지 못한다.

따라서 E1과 E2 외에 R(KA1, KA2)과 E21(KA2, A4)를 만들고, 두 릴레이션에서 각각 두 속성을 함께 기본키로 삼은 ③이 적절하다.

NoteBase

문 3. 데이터베이스 관리 시스템(DBMS)의 주요 구성요소 중 디스크에 저장된 사용자 데이터베이스나 시스템 카탈로그 접근을 담당하는 것으로 적절한 것은?

① 질의어 처리기

② 저장 데이터 관리자

③ 트랜잭션 관리자

④ 런타임 데이터베이스 처리기

정답 2번

간단한 해설

저장 데이터 관리자는 디스크에 있는 사용자 데이터베이스와 시스템 카탈로그에 대한 접근을 관리한다. 질의나 트랜잭션을 실행하는 처리기와 저장된 데이터를 읽고 쓰도록 지원하는 관리자의 역할을 구별한다.

개념 구분

저장 데이터 관리자: **디스크에 저장된 데이터에 대한 접근**을 관리하는 구성요소이다. 사용자 데이터베이스뿐 아니라 테이블·속성·제약조건 등의 메타데이터를 담은 시스템 카탈로그도 접근 대상이다. 운영체제의 기본 입출력 서비스를 이용하여 필요한 데이터를 읽고 쓰는 과정을 지원한다.

질의어 처리기: 사용자의 질의를 해석하고 적절한 실행 형태로 처리한다. 질의의 구문·의미 확인과 실행 계획 수립·최적화 등이 중심이므로, 이 문항에서 묻는 저장장치 접근 관리 역할과 구별된다.

트랜잭션 관리자: 트랜잭션의 실행을 관리하며 동시성 제어와 회복 등을 통해 여러 작업의 간섭이나 장애가 데이터의 일관성을 해치지 않도록 한다. 저장된 데이터를 읽고 쓰는 기능 자체보다 **트랜잭션의 안전한 실행**에 초점이 있다.

런타임 데이터베이스 처리기: 실행 단계에서 질의 계획·명령·트랜잭션을 수행한다. 저장 데이터 관리자와 협력하여 데이터를 이용하지만, 실행 연산을 수행하는 역할과 디스크상의 데이터 접근을 관리하는 역할은 구별된다.

따라서 제시된 역할에 해당하는 것은 저장 데이터 관리자인 ②이다.

NoteBase

문 4. 데이터베이스에서 한 조직의 고유 기능을 수행하기 위해 꼭 유지되어야 하는 것으로 적절한 것은?

① 공유 데이터

② 통합 데이터

③ 저장 데이터

④ 운영 데이터

정답 4번

간단한 해설

조직의 고유 업무를 수행하기 위해 계속 유지해야 하는 데이터는 **운영 데이터**이다. 발문은 공동 이용이나 저장 형태가 아니라 업무상 필요한 데이터의 성격을 묻고 있으므로 ④이다.

개념 구분

운영 데이터: **조직의 고유 업무를 계속 수행하기 위해 유지하는 데이터**를 뜻한다. 예를 들어 대학의 학적·수강 정보, 은행의 계좌·거래 정보가 해당한다. 발문의 '고유 기능을 수행하기 위해 꼭 유지'라는 표현은 데이터의 업무상 필요성과 이용 목적을 가리키므로 운영 데이터가 맞다. 여기서 '운영'은 운영체제용 데이터라는 뜻이 아니다.

공유 데이터: 여러 사용자나 응용프로그램이 공동으로 이용하는 데이터이다. 구분 기준은 **누가 함께 사용하는가**이다. 공유한다고 해서 모든 사람이 제한 없이 접근한다는 뜻은 아니며, 필요한 접근권한을 적용할 수 있다.

통합 데이터: 여러 곳에 흩어진 데이터를 하나의 일관된 체계로 모아 관리하는 데이터이다. 불필요한 중복을 줄이고 데이터 사이의 관계와 일관성을 관리한다는 점이 핵심이다. 통합을 중복이 단 한 건도 존재할 수 없다는 뜻으로 이해해서는 안 된다.

저장 데이터: 컴퓨터가 접근하여 처리할 수 있도록 전자적 저장장치에 기록된 데이터이다. 구분 기준은 데이터의 기록·보관 형태이다. 조직 업무에 반드시 필요한지라는 이용 목적을 직접 나타내는 용어는 아니다.

이 네 표현은 데이터베이스의 서로 다른 측면을 설명하며 한 데이터가 여러 성격을 함께 가질 수 있다. 이 문제에서 묻는 업무 수행상의 필수성을 나타내는 것은 운영 데이터이므로 ④이다.

NoteBase

문 5. 논리적 데이터 독립성에 대한 설명으로 적절하지 않은 것은?

① 데이터베이스의 논리적 구조가 변경될 때 필요하다.

② 외부 스키마와 개념 스키마 간의 사상(mapping)에 의해 제공된다.

③ 내부 스키마를 수정하더라도 개념 스키마에 영향을 미치지 않는다.

④ 개념 스키마를 수정하더라도 외부 스키마에 영향을 미치지 않는다.

정답 3번

간단한 해설

내부 스키마를 바꾸어도 개념 스키마가 유지되는 것은 **물리적 데이터 독립성**이다. 논리적 데이터 독립성은 **개념 스키마의 변경으로부터 외부 스키마를 보호**하는 성질이다.

선지별 해설

① O: 논리적 데이터 독립성은 테이블의 구성이나 속성 등 데이터베이스의 **전체 논리 구조**를 바꾸면서도 기존 사용자 관점과 응용 프로그램을 유지하려 할 때 필요하다. 논리 구조를 나타내는 개념 스키마와 사용자별 데이터 관점을 나타내는 외부 스키마를 분리하는 데 의미가 있다.

② O: 외부 스키마와 개념 스키마 사이의 사상은 사용자에게 보이는 데이터가 전체 논리 구조의 어느 부분에 대응하는지를 정한다. 개념 스키마가 바뀌면 **사상을 조정하여 변경을 흡수**함으로써 기존 외부 스키마와 응용 프로그램을 유지할 수 있다.

③ X: 내부 스키마를 변경하여도 개념 스키마가 영향을 받지 않는 것은 **물리적 데이터 독립성**이다. 예를 들어 파일 저장 방식이나 인덱스를 바꾸어도 논리적 테이블 구조를 유지하는 경우이며, 개념 스키마와 내부 스키마 사이의 사상이 변경을 흡수한다.

④ O: 논리적 데이터 독립성은 개념 스키마를 바꾸더라도 외부 스키마를 유지할 수 있는 능력이다. 예를 들어 기존 사용자에게 필요하지 않은 속성을 전체 테이블에 추가하여도 기존 사용자 뷰는 그대로 둘 수 있다. 다만 필요한 데이터 자체를 없애는 등 기존 사용자 요구를 충족할 수 없는 모든 변경까지 무조건 감출 수 있다는 뜻은 아니다.

NoteBase

문 6. 참조 무결성 규칙에 대한 설명으로 적절하지 않은 것은?

① 릴레이션은 참조할 수 없는 외래키 값을 가져서는 안 된다.

② 참조하는 릴레이션과 참조되는 릴레이션은 반드시 서로 다른 릴레이션이 되어야 하는 것은 아니다.

③ 데이터 형식을 통해 유형을 제한하거나 CHECK 제약조건 및 규칙을 통해 형식을 제한한다.

④ 키 값이 변경되면 해당키 값에 대한 모든 참조가 데이터베이스 전체에서 일관되게 변경되고 유지되어야 한다.

정답 3번

간단한 해설

데이터형이나 CHECK로 속성값의 허용 범위·조건을 제한하는 것은 **도메인 무결성**에 관한 설명이다. 참조 무결성은 외래키와 참조 대상 키 사이의 대응이 일관되게 유지되도록 하는 규칙이다.

선지별 해설

참조 무결성은 외래키가 가리키는 대상이 유효하도록 유지하는 규칙이다. 예를 들어 주문에 상품번호를 저장할 때, 존재하지 않는 상품을 가리키는 번호를 넣지 못하도록 한다. 속성값 자체의 형식이나 허용 범위를 제한하는 도메인 무결성과 구분한다.

① O: 외래키가 값을 가지면 참조 대상 릴레이션의 해당 키 값과 대응해야 한다. 존재하지 않는 대상을 가리키는 외래키 값을 허용하면 참조 관계가 깨진다. 다만 **NULL이 허용된 외래키의 NULL**은 존재하지 않는 특정 키 값을 가리키는 것과 구별하며, 별도의 NOT NULL 제약이 없으면 허용될 수 있다.

② O: 외래키가 같은 릴레이션의 키를 참조하는 **자기 참조**도 가능하다. 예를 들어 사원 릴레이션에서 상사번호가 같은 릴레이션의 사원번호를 참조할 수 있다. 참조하는 릴레이션과 참조되는 릴레이션이 반드시 서로 달라야 하는 것은 아니다.

③ X: 데이터형과 CHECK 등을 사용해 값의 종류·범위·조건을 제한하는 것은 **도메인 무결성**의 설명이다. 예를 들어 나이에 음수가 들어가지 못하게 하는 조건이 이에 해당한다. 참조 무결성은 이런 값 자체의 조건이 아니라 외래키와 참조 대상 키의 유효한 대응을 보장한다.

④ O: 참조 대상 키가 변경될 때도 관련 외래키를 그대로 방치하여 참조가 깨지게 해서는 안 된다. ON UPDATE CASCADE를 지정하면 연결된 외래키 값도 함께 갱신된다. 다만 이는 **항상 자동 연쇄 갱신을 해야 한다는 뜻은 아니다**. 설정된 참조 동작에 따라 무결성을 깨는 키 변경을 거부하는 방식도 가능하며, 핵심은 변경 후에도 참조 관계가 일관되어야 한다는 점이다.

NoteBase

문 7. 두 릴레이션 R(A,B,C)과 S(D,E,F)에서 현재 존재하는 값만을 고려할 때 키(key)에 대한 설명으로 적절한 것은 모두 몇 개인가? (단, NULL은 널 값이다.)

R

A	B	C
1	a	x
2	a	y
3	b	x

S

D	E	F
a	1	2
b	2	2
c	NULL	2

가. A는 R의 후보키가 될 수 있다.

나. B는 R의 기본키가 될 수 없다.

다. BC는 R의 대체키가 될 수 있다.

라. ABC는 R의 수퍼키가 될 수 있다.

마. E는 S의 외래키가 될 수 있다.

바. F는 S의 외래키가 될 수 있다.

① 3개

② 4개

③ 5개

④ 6개

정답 4번

간단한 해설

A와 BC는 각각 유일성·최소성을 충족하고, 중복된 B는 기본키가 될 수 없으며 ABC는 수퍼키가 될 수 있다. E의 NULL과 F의 중복도 **외래키가 되는 데 그 자체로 장애가 되지 않는다**. 따라서 여섯 보기 모두 적절하여 ④이다.

NoteBase

보기별 해설

문제는 현재 표에 있는 값만으로 키가 될 수 있는지를 묻는다. R에는 (1, a, x), (2, a, y), (3, b, x)가 있고, S에는 (a, 1, 2), (b, 2, 2), (c, NULL, 2)가 있다. 앞으로 들어올 모든 데이터에 대한 업무 규칙까지 표만으로 확정하는 문제는 아니다.

가. O: R의 A 값은 1, 2, 3으로 모두 달라 각 튜플을 유일하게 식별한다. A는 하나의 속성이므로 식별력을 유지하면서 더 뺄 속성도 없다. 따라서 현재 값에서 **유일성과 최소성을 갖춘 후보키**가 될 수 있다.

나. O: R의 B 값은 a, a, b로 첫째 행과 둘째 행에서 a가 중복된다. B만으로 두 튜플을 구별할 수 없어 후보키가 아니므로, 후보키 중에서 선택하는 기본키가 될 수도 없다.

다. O: BC의 값 쌍은 (a, x), (a, y), (b, x)로 서로 다르다. B만 보면 a가 중복되고 C만 보면 x가 중복되므로 두 속성 중 어느 하나도 제거할 수 없다. 따라서 BC도 후보키이며, **A를 기본키로 선택하면 BC는 대체키**가 될 수 있다. 후보키의 최소성은 모든 후보키 중 속성 수가 가장 적어야 한다는 뜻이 아니라, 그 후보키에서 불필요한 속성을 더 뺄 수 없다는 뜻이다.

라. O: 수퍼키는 튜플의 유일한 식별을 보장하면 되며 최소성은 요구하지 않는다. 이미 A만으로 튜플을 구별할 수 있으므로 A를 포함한 ABC도 수퍼키가 될 수 있다. 다만 B와 C를 빼도 식별 가능하므로 ABC 자체는 최소성을 갖춘 후보키가 아니다.

마. O: S의 E 값 중 NULL이 아닌 값은 1, 2이며 모두 R의 A 값에 존재한다. A를 R의 기본키로 정하면 E는 이를 참조하는 외래키가 될 수 있다. **외래키에는 별도의 NOT NULL 제약이 없다면 NULL을 허용**할 수 있으므로, 셋째 행의 NULL이 외래키 가능성을 배제하지 않는다.

바. O: S의 F 값은 모두 2이며, 2는 R의 A에 존재한다. 따라서 F도 A를 참조하는 외래키가 될 수 있다. 외래키는 참조하는 행이 존재해야 한다는 조건이지, 참조하는 쪽의 값까지 유일해야 한다는 조건은 아니므로 **외래키 값의 중복은 허용**된다.

가·나·다·라·마·바이 모두 적절하므로 6개인 선택지 ④이다.

NoteBase

문 8. 다음 릴레이션에서 '데이터베이스' 책을 100번 고객이 주문한 날짜를 구하기 위한 관계대수식으로 적절하지 않은 것은?

고객(고객번호(PK), 이름)
책(책번호(PK), 제목)
주문(고객번호(FK), 책번호(FK), 주문일)

① _주문일(_제목='데이터베이스' _고객번호=100(고객주문책))
② _주문일(_제목='데이터베이스'((_고객번호=100(고객))주문책))
③ _주문일((_고객번호=100(고객))주문(_제목='데이터베이스'(책)))
④ _주문일(_제목='데이터베이스'(_고객번호=100(고객주문책)))

정답 1번

간단한 해설

'데이터베이스'라는 제목과 고객번호 100이라는 조건은 **동시에 충족**해야 한다. 두 조건을 논리합으로 연결하는 ①의 형태는 이 요구를 충족하지 못하며, 논리곱 또는 선택 연산의 연속 적용으로 표현해야 한다.

NoteBase

핵심 풀이

요구하는 결과는 제목이 '데이터베이스'인 책을 고객번호 100인 고객이 주문한 주문일이다. 따라서 **제목 조건과 고객번호 조건을 모두 만족**해야 하며, 두 조건은 논리곱 으로 결합한다. 선택 연산 σ는 조건에 맞는 행을 남기고, 투영 연산 Π는 필요한 속성만 남긴다.

고객과 주문은 고객번호로, 주문과 책은 책번호로 자연조인한다. R=고객주문책이라 두면 올바른 식은 _주문일(_제목='데이터베이스'고객번호=100(R))처럼 표현할 수 있다. 고객·책의 정보를 주문과 연결한 뒤 조건을 만족하는 행에서 주문일만 추출한다.

②는 고객번호 조건을 고객 릴레이션에 먼저 적용하고 조인한 뒤 제목 조건을 적용한다.

③은 고객번호 조건과 제목 조건을 각각 해당 릴레이션에 미리 적용한 뒤 조인한다.

④는 세 릴레이션을 조인한 뒤 두 선택 연산을 차례로 적용한다.

해당 릴레이션의 속성만 사용하는 선택 조건은 자연조인 전에 적용할 수 있고, 두 선택을 연속 적용하면 두 조건을 모두 만족하는 행만 남으므로 세 식은 요구한 결과를 구한다.

①은 선택 연산자들을 로 연결한 형태로, 표준적인 단일 선택 조건이나 선택의 연속 적용을 나타내지 못한다. 그 기호를 두 조건의 논리합으로 읽더라도 한 조건만 만족하는 행을 허용하므로 요구와 맞지 않는다. 예를 들어 다른 고객의 '데이터베이스' 책 주문이나 100번 고객의 다른 제목 책 주문까지 포함할 수 있다.

따라서 적절하지 않은 식은 ①이다.

NoteBase

문 9. 릴레이션 R과 S에 대한 자연조인 $R \bowtie_N S$와 동등하지 않은 관계대수식은?

① $S \bowtie_N (S \ltimes R)$

② $(S \ltimes R) \bowtie_N R$

③ $R \bowtie_N (S \ltimes R)$

④ $(R \ltimes S) \bowtie_N S$

정답 1번

간단한 해설

반조인 $S \ltimes R$는 R과 연결되는 **S의 튜플과 속성만** 남긴다. 이를 다시 S와 조인해도 R의 고유 속성을 얻을 수 없으므로, 양쪽 릴레이션의 속성을 결합하는 자연조인과 일반적으로 같지 않다.

핵심 풀이

자연조인은 공통 속성의 값이 같은 튜플들을 결합하고 양쪽 속성을 합친다. 왼쪽 반조인 $R \ltimes S$는 그 결과를 R의 속성에 투영하므로, S와 연결되는 R의 튜플만 남기고 S에만 있는 속성은 포함하지 않는다. 관계대수에서는 튜플을 집합으로 다룬다.

①의 식: $S \ltimes R$는 R과 조인되는 S의 튜플만 남긴 것으로, 속성은 S와 같다. 따라서 이를 S와 자연조인해도 $S \bowtie_N (S \ltimes R) = S \ltimes R$이다. 양쪽 피연산자에 모두 S의 속성만 있으므로 **R에만 있는 속성을 복구할 수 없다**. 예를 들어 설명용 릴레이션 R(A, B)에 튜플 (1, 2), S(B, C)에 튜플 (2, 3)이 있다면 원래 자연조인 결과는 (A, B, C)의 (1, 2, 3)이지만, ①의 결과는 (B, C)의 (2, 3)이다. 특정 자료에서 우연히 같은 결과가 나올 수 있어도 일반적인 동치식은 아니다.

②의 식: $S \ltimes R$에서 제외된 S의 튜플은 애초에 R의 어떤 튜플과도 조인되지 않는다. 이들을 제거한 뒤 R과 조인해도 **원래 조인에 참여하는 모든 튜플 쌍이 유지**된다. 따라서 $(S \ltimes R) \bowtie_N R = S \bowtie_N R = R \bowtie_N S$이다.

③의 식: 자연조인은 **교환법칙**이 성립하므로 ②의 두 피연산자 위치를 바꾸어도 결과가 같다. 따라서 $R \bowtie_N (S \ltimes R) = R \bowtie_N S$이다. 여기서는 S의 속성만 남은 중간 결과를 R과 조인하므로 양쪽의 필요한 속성이 모두 포함된다.

④의 식: $R \ltimes S$는 S와 조인할 상대가 없는 R의 튜플만 제거한다. 남은 R의 튜플을 S와 조인하면 원래 조인 결과를 그대로 얻는다. 따라서 $(R \ltimes S) \bowtie_N S = R \bowtie_N S$이다. 반조인으로 한쪽을 줄인 뒤 **상대편 원래 릴레이션과 조인**하는 구조이다.

원래 자연조인과 일반적으로 동등하지 않은 식은 ①이다.

NoteBase

문 10. 관계대수 연산자에 대한 설명으로 적절하지 않은 것은?

① 릴레이션 R에 실렉트(select) 연산을 수행한 결과로 얻은 릴레이션의 차수는 R의 차수와 같다.

② 차집합과 디비전(division)은 교환법칙과 결합법칙이 성립하지 않는다.

③ 복합 연산인 디비전은 기본 연산인 실렉트, 차집합, 카티션 프로덕트(cartesian product)로 대체할 수 있다.

④ 카티션 프로덕트와 외부 합집합(outer union)은 릴레이션이 합병 가능(union compatible)하지 않아도 연산이 가능하다.

정답 3번

간단한 해설

디비전은 **프로젝트·차집합·카티션 프로덕트**로 표현할 수 있다. 실렉트는 행을 고르지만 디비전 결과에 필요한 열만 남기지 못하므로, ③의 '실렉트'를 '프로젝트'로 바꾸어야 한다.

선지별 해설

① O: 실렉트는 조건을 만족하는 **튜플, 즉 행을 선택**하는 연산이다. 속성인 열의 구성은 그대로이므로 결과 릴레이션의 차수는 원래 R과 같다. 선택된 행이 없더라도 결과의 스키마는 유지되며, 행 수인 카디널리티만 0이 될 수 있다.

② O: 차집합은 앞 릴레이션에서 뒤 릴레이션에 포함된 튜플을 제거하므로 순서를 바꾸거나 괄호의 위치를 바꾸어도 결과가 같다는 법칙이 일반적으로 성립하지 않는다. 디비전도 제수의 모든 튜플과 대응하는 값을 피제수에서 찾는 **방향성이 있는 연산**이다. 제수의 속성은 피제수에 포함되어야 하며, 두 피연산자를 뒤집거나 나누는 순서를 바꾸면 결과 속성이나 연산 가능 여부 자체가 달라질 수 있어 교환법칙과 결합법칙을 일반 법칙으로 적용할 수 없다.

③ X: 필요한 기본 연산은 실렉트가 아니라 **프로젝트**, 그리고 차집합과 카티션 프로덕트이다. 서로 겹치지 않는 속성 집합 X, Y에 대하여 R(X, Y)를 S(Y)로 나누면 결과는 X 속성만 가진다. 그 식은 $R \div S = \pi_X(R) - \pi_X((\pi_X(R) \times S) - R)$이다. 먼저 R의 X 값 후보와 S의 모든 Y 값을 조합한 뒤 R에 없는 조합을 찾고, 그런 누락 조합이 있는 X 후보를 제거한다. 이 과정에서 X 열만 추출하기 위해 프로젝트가 필요하다.

④ O: 일반적인 합집합은 두 릴레이션의 차수가 같고 대응 속성의 도메인이 호환되는 **합병 가능성**을 요구한다. 카티션 프로덕트는 양쪽 튜플의 모든 조합을 연결하므로 이 조건을 요구하지 않으며, 같은 속성명이 있으면 이름을 구별해 준다. 외부 합집합은 일부 속성이 호환되는 릴레이션들의 공통 속성과 서로 다른 속성을 함께 보존하고 필요한 곳에 NULL을 채우는 확장 연산이므로, 일반 합집합처럼 전체 속성이 합병 가능할 필요는 없다.

NoteBase

문 11. 데이터 정의어(DDL)에 대한 설명으로 적절하지 않은 것은?

① CREATE TABLE 문에서 PRIMARY KEY 옵션은 개체 무결성을, FOREIGN KEY 옵션은 참조 무결성을 유지하기 위한 기본적인 제약조건을 명시하는 역할을 한다.

② DROP 문의 옵션으로 RESTRICT를 명시하는 경우, 삭제할 대상이 다른 곳에서 참조 중이면 삭제하지 않는다.

③ CREATE ASSERTION 문은 기본 제약조건 외 추가의 제약조건을 명시하여 해당 제약조건을 위반하는 연산이 수행되지 않도록 한다.

④ CREATE INDEX 문으로 하나의 테이블에 여러 개의 인덱스를 지정할 수 있으며 갱신 성능 향상에 도움이 된다.

정답 4번

간단한 해설

인덱스는 주로 **검색할 행을 빠르게 찾는 데** 도움을 준다. 데이터가 변경되면 관련 인덱스도 관리해야 하므로 **갱신 유지 비용이 추가**되며, 여러 인덱스가 갱신 성능을 높인다고 일반화할 수 없다.

선지별 해설

DDL은 테이블·인덱스 등의 데이터베이스 객체와 제약조건을 정의하거나 변경·제거하는 언어이다. 기본키와 외래키 등은 데이터의 무결성을 유지하고, 인덱스는 주로 데이터 접근 효율을 높이기 위한 구조이다.

① O: PRIMARY KEY는 기본키의 유일성과 NULL 금지를 통해 각 행을 식별하게 하므로 개체 무결성을 유지한다. FOREIGN KEY는 외래키와 참조 대상 키의 유효한 대응을 강제하여 **참조 무결성**을 유지한다.

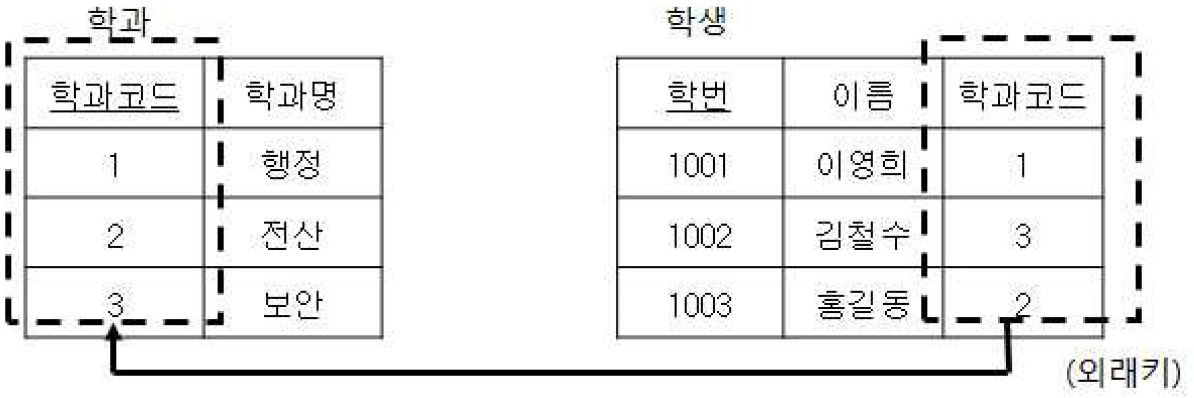
② O: DROP에 RESTRICT를 지정하면 다른 객체가 삭제 대상에 의존하는 경우 삭제를 거부한다. 예를 들어 다른 테이블의 외래키가 참조하는 테이블을 함부로 제거하지 못하게 한다. 의존 객체도 함께 처리하는 CASCADE와 구분된다.

③ O: CREATE ASSERTION은 SQL 표준에서 데이터베이스가 만족해야 할 일반적인 무결성 조건을 별도로 선언하는 기능이다. 특정 열이나 한 테이블의 기본 제약조건만으로 표현하기 어려운 조건도 명시할 수 있으며, 제약을 위반하는 데이터 변경을 허용하지 않는 것이 목적이다. 다만 실제 문법의 지원 여부는 DBMS마다 다르다.

④ X: 하나의 테이블에 여러 인덱스를 만들 수 있다는 부분은 맞지만, 인덱스의 일반적인 장점은 **검색 성능 향상**이다. 삽입·삭제 또는 인덱스가 가리키는 값의 변경 시 관련 인덱스도 유지해야 하므로 갱신 비용이 늘 수 있다. 인덱스가 UPDATE의 대상 행을 찾는 데 도움을 주는 경우도 있지만, 이를 인덱스가 많을수록 갱신 자체가 빨라진다는 뜻으로 해석해서는 안 된다.

NoteBase

문 12. 그림과 같이 학생 테이블을 생성할 때 학과코드의 외래키 정의에서 ON DELETE 구문을 지정하지 않으면 기본으로 선택되는 옵션으로 적절한 것은? (단, 밑줄은 기본키이다.)



① CASCADE

② NO ACTION

③ SET NULL

④ SET DEFAULT

정답 2번

간단한 해설

ON DELETE를 생략하면 기본 옵션은 **NO ACTION**이다. 참조하는 학생이 남아 있는 학과를 삭제하여 참조무결성이 깨지면 삭제가 실패하며, 학생을 자동 삭제하거나 외래키를 NULL로 바꾸지는 않으므로 ②이다.

NoteBase

개념 구분

그림에서 학과 테이블의 '학과코드'는 기본키이며, 학생 테이블의 '학과코드'가 이를 참조하는 외래키이다. ON DELETE는 학생 행 자체를 삭제할 때가 아니라 **참조되는 학과 행을 삭제할 때의 처리 방식**을 정한다.

NO ACTION: ON DELETE를 생략했을 때의 기본 동작이다. 학생 행을 자동으로 삭제하거나 외래키 값을 다른 값으로 바꾸지 않는다. 대신 **외래키 제약 검사 시점에 참조무결성을 위반하면 삭제가 실패**한다. 예를 들어 학과코드 3을 참조하는 학생 1002가 남아 있는데 학과코드 3인 학과를 삭제하려 하면 제약 위반이다. '아무 조치도 하지 않는다'를 참조무결성 위반을 무시하고 삭제한다는 뜻으로 이해해서는 안 된다.

CASCADE: 학과 행을 삭제하면 그 학과를 참조하는 학생 행도 함께 삭제한다. 예를 들어 학과코드 3인 학과 삭제에 따라 학생 1002의 행도 삭제되는 방식이다. 이러한 연쇄 삭제는 명시적으로 지정해야 하며 기본 옵션이 아니다.

SET NULL: 삭제되는 학과를 참조하던 학생의 학과코드를 NULL로 바꾼다. 학생 행은 남고 참조값이 비워지는 것이므로 CASCADE와 다르다. 해당 외래키 열에 NULL이 허용되어야 하며, ON DELETE 생략만으로 자동 적용되지 않는다.

SET DEFAULT: 삭제되는 학과를 참조하던 학생의 학과코드를 그 열의 기본값으로 바꾼다. **바뀐 기본값도 외래키 제약을 충족**해야 하므로, 기본값이 참조할 수 없는 학과코드라면 문제가 해결되지 않는다. 이것도 별도로 지정하는 처리 방식이다.

따라서 지정하지 않았을 때 선택되는 옵션은 NO ACTION인 ②이다.

NoteBase

문 13. 뷰(view)에 대한 설명으로 적절하지 않은 것은?

① 뷰는 기본 테이블을 기반으로 만들어지며, 뷰를 기반으로 새로운 뷰를 만들 수 있다.

② 뷰의 생성, 변경을 위해 CREATE, ALTER 문 사용이 가능하다.

③ 뷰의 내용에 대한 삽입, 수정, 삭제 연산이 항상 허용되는 것은 아니다.

④ 모든 뷰에 대한 검색이 가능하다.

정답 2번

간단한 해설

②의 ALTER에 의한 뷰 정의 변경은 **표준 SQL 기능이 아니라 DBMS별 확장 기능**이다. 표준 SQL 명령으로는 기존 뷰를 DROP VIEW로 제거한 뒤 CREATE VIEW로 다시 정의할 수 있다. 다만 SQL Server처럼 ALTER VIEW를 지원하는 제품도 있으므로, 모든 DBMS에서 뷰 변경이 불가능하다는 뜻은 아니다.

선지별 해설

① O: 뷰는 기본 테이블이나 다른 뷰를 조회하는 질의로 정의하는 가상 테이블이다. 따라서 **이미 만든 뷰를 기반으로 새로운 뷰를 정의**할 수 있다. 예를 들어 직원 테이블에서 개발부 직원만 보여 주는 뷰를 만든 뒤, 그 뷰에서 이름만 조회하는 새 뷰를 만들 수 있다.

② X: 표준 SQL을 기준으로 보면, CREATE VIEW는 뷰 생성 명령이지만 **ALTER VIEW를 통한 뷰 정의 변경은 표준 SQL 기능이 아니다**. 뷰를 정의하는 조회식을 바꾸려면 기존 뷰를 DROP VIEW로 제거하고, 새 조회식으로 CREATE VIEW를 실행하여 다시 정의할 수 있다. 다만 SQL Server는 ALTER VIEW로 조회식을 변경하는 기능을 제공한다. 따라서 이 선지는 표준 SQL 기준으로 구분해야 하며, 실제 DBMS에서도 ALTER VIEW를 전혀 사용할 수 없다고 일반화해서는 안 된다.

③ O: 뷰를 통한 삽입·수정·삭제가 가능하려면 그 연산을 **기본 테이블의 변경으로 명확하게 대응**시킬 수 있어야 한다. 집계 등으로 이러한 대응이 어려운 뷰에는 변경 연산이 제한된다. 예를 들어 부서별 급여 합계만 보여 주는 뷰에서 합계값을 바꾸려 해도, 어느 직원의 급여를 얼마씩 바꿀지 정해지지 않는다. 따라서 모든 뷰에서 삽입·수정·삭제가 항상 허용되는 것은 아니다.

④ O: 정상적으로 정의된 뷰에 필요한 조회 권한이 있으면, 뷰를 SELECT 문의 대상으로 삼아 검색할 수 있다. 뷰는 정의된 질의의 결과를 테이블처럼 제공하므로, 집계 등으로 **변경 연산이 제한된 뷰도 검색은 가능**하다. 여기서 모든 뷰를 검색할 수 있다는 것은 누구나 권한 없이 조회할 수 있다는 뜻은 아니다.

NoteBase

문 14. 다음 함수종속을 갖는 사원특강(사원번호, 특강과목, 강사코드) 릴레이션에 대한 설명으로 적절하지 않은 것은? (단, 밑줄은 기본키이다.)

함수종속 : (사원번호, 특강과목) → 강사코드
강사코드 → 특강과목
기본키 : (사원번호, 특강과목)

① 사원특강 릴레이션은 제2정규형을 만족한다.

② 사원특강 릴레이션은 제3정규형을 만족한다.

③ 사원특강 릴레이션에서 이상 현상이 발생하므로 분해가 필요하다.

④ R1(사원번호, 특강과목)과 R2(강사코드, 특강과목)로 분해하는 것이 가장 좋다.

정답 4번

간단한 해설

제시된 분해는 사원과 강사의 연결 정보를 잃어, 다시 조인하면 **원래 없던 사원·강사 조합**을 만들 수 있다.
강사코드가 특강과목을 결정하므로 **(사원번호, 강사코드)와 (강사코드, 특강과목)**으로 분해해야 무손실 조인이 가능하다.

NoteBase

선지별 해설

사원번호를 S, 특강과목을 C, 강사코드를 I라고 하면 주어진 함수종속은 $SC \rightarrow I$와 $I \rightarrow C$이다. (S, C)는 I를 결정하고, (S, I)는 I를 통해 C를 결정하므로 두 쌍 모두 후보키이다. 따라서 세 속성은 모두 어떤 후보키에 포함되는 주요 속성이다.

① O: 제2정규형은 제1정규형인 릴레이션에서 비주요 속성이 후보키의 일부에만 종속되는 부분 함수종속을 배제한다. 이 릴레이션에는 **비주요 속성이 없으므로** 제2정규형을 만족한다. 강사코드만으로 특강과목을 결정할 수 있어도, 특강과목이 주요 속성이므로 제2정규형 위반이 되지 않는다.

② O: 제3정규형은 모든 비자명한 함수종속에서 결정자가 슈퍼키이거나 종속되는 속성이 주요 속성이어야 한다. $SC \rightarrow I$의 SC는 후보키이고, $I \rightarrow C$의 C는 주요 속성이다. 따라서 **I가 슈퍼키가 아니어도 제3정규형은 만족**한다. 모든 결정자가 슈퍼키여야 하는 BCNF와 구별된다.

③ O: 같은 강사를 여러 사원이 수강하면 그 강사의 특강과목 정보가 반복 저장된다. 강사의 과목 정보를 바꿀 때 여러 행을 함께 고쳐야 하므로 갱신 이상이 생길 수 있다. 또한 수강 사원 없이 강사와 과목 정보만 저장하기 어렵고, 마지막 수강 사원을 삭제하면 강사·과목 정보도 잃을 수 있다. 슈퍼키가 아닌 I가 C를 결정하므로 BCNF를 만족하지 않아 이러한 중복을 줄이는 분해가 필요하다.

④ X: (S, C)와 (I, C)의 공통 속성 C는 어느 릴레이션 전체도 결정하지 못하므로 이 분해는 무손실 조인을 보장하지 못한다. 예를 들어 원래 (사원1, 과목1, 강사1), (사원2, 과목1, 강사2)만 있었어도 분해 후 과목으로 조인하면 사원1·강사2와 사원2·강사1의 잘못된 조합이 추가된다. 대신 R1(사원번호, 강사코드)과 R2(강사코드, 특강과목)로 분해한다. 공통 속성인 **강사코드가 R2 전체를 결정**하므로 무손실 분해이며, R1의 기본키는 두 속성의 조합, R2의 기본키는 강사코드이다. 다만 원래의 $SC \rightarrow I$는 이 두 릴레이션의 개별 제약만으로 보존되지 않으므로, 그 제약을 유지하려면 추가 검사가 필요하다.

NoteBase

문 16. 관계형 모델에서 키에 대한 설명으로 적절한 것은 모두 몇 개인가?

가. 수퍼키는 기본키가 되기 위한 필요충분조건이다.
나. 외래키와 참조된 기본키는 동일한 도메인을 가져야 하며 이름이 같을 필요는 없다.
다. 외래키가 기본키로 사용되는 경우는 없다.
라. 후보키와 외래키는 유일성과 최소성을 모두 만족해야 한다.

① 1개
② 2개
③ 3개
④ 4개

정답 1번

간단한 해설

수퍼키는 유일성을 갖추지만 **최소성을 반드시 갖추는 것은 아니다**. 외래키는 참조 관계를 나타내므로 유일성·최소성을 필수로 요구하지 않으며, 별도의 기본키 조건도 만족하면 기본키를 겸할 수 있다.

보기별 해설

가. X: 기본키는 튜플을 유일하게 식별하므로 반드시 수퍼키이지만, 수퍼키에는 불필요한 속성이 포함될 수 있다. 후보키는 수퍼키에 **최소성**까지 갖춘 것이고 기본키는 후보키 중에서 선택한다. 따라서 수퍼키라는 조건은 기본키가 되기 위한 필요조건이지만 충분조건은 아니다.

나. O: 외래키의 값은 참조 대상 기본키의 값과 대응해야 하므로, 관계형 모델에서 대응하는 속성들은 **동일한 도메인**을 가져야 한다. 그러나 외래키 속성과 참조된 기본키 속성의 이름까지 같을 필요는 없다. 예를 들어 사원의 '소속부서번호'가 부서의 '부서번호'를 참조할 수 있다.

다. X: 한 속성 집합이 **기본키와 외래키를 동시에** 맡을 수 있다. 예를 들어 학생 테이블의 기본키가 학번이고 장학생 테이블도 학번을 기본키로 사용하면서 학생의 학번을 참조한다면, 장학생의 학번은 자기 테이블의 튜플을 식별하는 기본키이자 학생 테이블을 참조하는 외래키이다.

라. X: **유일성과 최소성을 모두 요구하는 것은 후보키**이다. 외래키는 참조 관계를 나타내는 속성 집합이므로 참조하는 쪽에서 값이 중복될 수 있으며, 외래키라는 이유만으로 튜플 식별을 위한 최소성도 요구되지 않는다. 예를 들어 여러 사원이 같은 부서번호를 외래키 값으로 가질 수 있다.

적절한 설명은 나 하나이므로 1개이며, ①에 해당한다.

NoteBase

문 17. 함수 종속성에 대한 추론 규칙으로 적절하지 않은 것은?

① X → Y 이고 X → Z이면 X → YZ 이다.

② X → YZ 이면 X → Y 와 X → Z 이다.

③ X → Y 이면 WX → WY 이고 X → WY 이다.

④ W → X 이고 XY → Z 이면 WY → Z 이다.

정답 3번

간단한 해설

③은 증대 규칙으로 성립하는 WX → WY에 근거 없는 X → WY를 덧붙였다. **X → Y만으로는 X가 W까지 결정한다고 할 수 없다.**

선지별 해설

함수 종속성 X → Y는 X 값이 같은 튜플들의 Y 값도 같다는 뜻이다. XY, WX 등은 속성 집합의 합집합을 나타낸다.

① O 합집합 규칙이다. **X가 Y와 Z를 각각 결정하면**, X 값이 같은 튜플들은 Y와 Z 값이 모두 같으므로 X → YZ가 성립한다.

② O 분해 규칙이다. X가 YZ 전체를 결정하면 **그 일부인 Y와 Z도 각각 결정**하므로 X → Y와 X → Z가 성립한다.

③ X X → Y의 양쪽에 W를 더한 WX → WY는 증대 규칙에 따라 성립한다. 그러나 왼쪽의 W를 없앤 **X → WY는 일반적으로 성립하지 않는다**. 예를 들어 설명용 두 튜플의 (X, Y, W) 값이 (1, a, 10), (1, a, 20)이면 X → Y는 만족하지만 X → W와 X → WY는 만족하지 않는다. 따라서 뒤의 X → WY를 삭제해야 옳다. 이를 유지하려면 X → W 같은 추가 조건이 필요하다.

④ O 의사 이행 규칙이다. W → X의 **양쪽에 Y를 더하면 WY → XY**가 된다. 여기에 XY → Z를 적용하면 이행 규칙에 따라 WY → Z가 성립한다.

NoteBase

문 18. 정규화에 대한 설명으로 적절한 것은 모두 몇 개인가?

가. 정규화를 통해 종속성을 모두 제거하게 된다.
나. 정규화를 통해 릴레이션을 분해하면 검색 연산시간이 감소한다.
다. 데이터베이스 연산의 이상 현상을 없애기 위함이다.
라. 릴레이션을 바람직한 형태로 점차 유도하는 스키마 변환 과정을 거친다.

① 1개
② 2개
③ 3개
④ 4개

정답 2번

간단한 해설

정규화는 데이터 중복으로 생기는 삽입·삭제·갱신 이상을 줄이기 위해 릴레이션을 적절한 정규형으로 변환하는 과정이다. **이상 현상의 방지와 스키마 구조의 개선**이라는 목적과 방법이 판단 기준이다.

보기별 해설

가. X: 정규화는 **모든 종속성을 없애는 과정이 아니다**. 함수 종속성 등을 분석하여 중복과 이상을 일으키는 구조를 적절히 분해한다. 예를 들어 후보키가 다른 속성을 결정하는 함수 종속성은 정상적인 릴레이션에서도 존재한다. 분해할 때에는 무손실 조인을 확보하고 필요한 종속성의 보존도 고려한다.

나. X: 정규화 후 검색 연산시간이 반드시 감소하지는 않는다. 릴레이션이 분해되면 여러 테이블을 다시 조인해야 하는 질의의 처리 비용이 늘어날 수 있다. 정규화의 주된 목적은 **검색 속도의 일률적 향상이 아니라 중복과 이상 현상의 개선**이다.

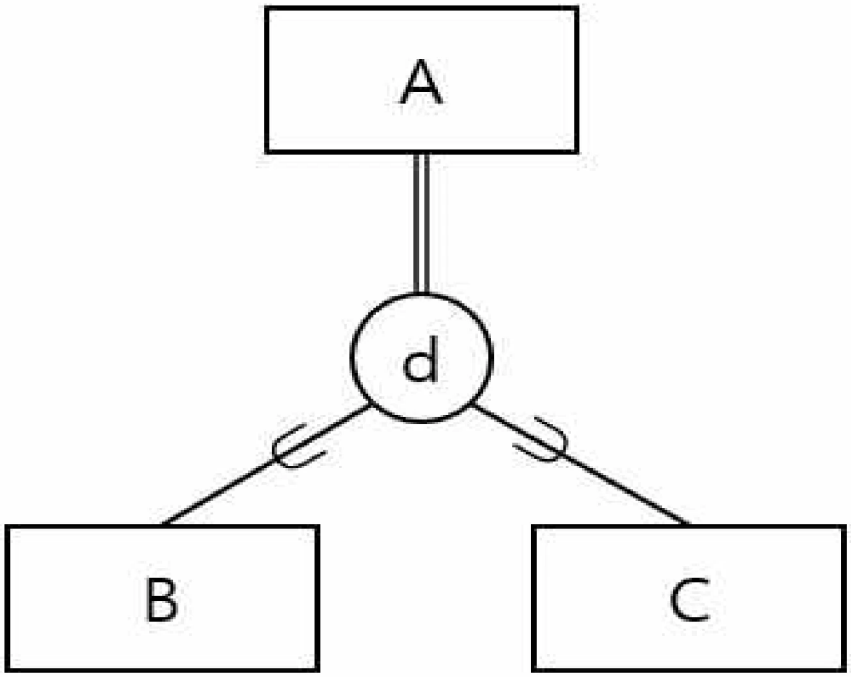
다. O: 정규화는 부적절한 데이터 중복으로 발생하는 삽입·삭제·갱신 이상을 방지하거나 제거하기 위한 설계 방법이다. 같은 사실이 여러 행에 반복되어 일부만 수정되거나, 한 사실을 삭제하면서 다른 필요한 사실까지 잃는 문제를 줄이도록 데이터를 분리한다.

라. O: 정규화는 속성 사이의 종속 관계를 분석하고 정규형의 조건에 맞도록 릴레이션을 변환하는 과정이다. 제1정규형·제2정규형·제3정규형 등의 기준에 따라 중복과 이상을 줄일 수 있는 **바람직한 스키마 구조**로 유도한다.

적절한 보기는 다와 라의 2개이므로 ②에 해당한다.

NoteBase

문 19. 다음 EER(Enhanced ER) 다이어그램은 ISA 관계를 표현한 것이다. 이중선은 전체 참여(total participation)를, 'd'는 분할(disjoint)을 의미한다. 이 다이어그램에 대한 설명으로 적절한 것을 모두 고른 것은? (단, 각 엔티티에 대한 속성은 적절하게 정의되어 있다고 가정한다.)



가. B 엔티티와 C 엔티티가 A 엔티티로 일반화될 수 있다.
나. B 엔티티도 아니고 C 엔티티도 아닌 A 엔티티가 존재할 수 있다.
다. B와 C 엔티티가 중복(overlap)된다면 A 엔티티는 테이블로 변환할 수 없다.
라. A 엔티티를 테이블로 표현하지 않고 A 엔티티의 속성을 상속받아 B와 C 엔티티만 테이블로 표현할 수 있다.

① 가, 나
② 가, 라
③ 나, 다
④ 가, 다, 라

정답 2번

간단한 해설

전체 참여이므로 모든 A 개체는 B 또는 C에 속하고, 분할 조건에 따라 **둘 중 정확히 하나에 속한다**. 따라서 B와 C를 상위 A로 일반화할 수 있으며, A의 속성을 상속한 B·C 테이블만으로도 개체 정보를 표현할 수 있다.

NoteBase

보기별 해설

그림에서 A는 상위 엔티티, B와 C는 하위 엔티티이다. 이중선은 모든 A 개체가 적어도 한 하위 엔티티에 속한다는 전체 참여이고, 원 안의 d는 B와 C에 동시에 속할 수 없다는 분할 조건이다. 개체 집합으로 나타내면 $A = B \cup C$, $B \cap C = \varnothing$이다.

가. O: 일반화는 하위 엔티티들의 공통 특성을 묶어 상위 엔티티를 만드는 과정이다. B와 C는 모두 A의 한 종류이므로 **B와 C의 공통 속성과 성질을 A로 일반화**할 수 있다. 반대로 A를 B와 C로 구분하는 과정은 특수화이다.

나. X: B도 C도 아닌 A 개체가 존재하면 전체 참여 조건을 위반한다. 이중선에 따라 **모든 A 개체가 B 또는 C에 포함**되어야 하며, d 조건까지 있으므로 동시에 둘에 속하지도 않는다. 어느 하위 엔티티에도 속하지 않는 A 개체를 허용하는 것은 부분 참여의 경우이다.

다. X: 하위 엔티티 간 중복이 허용되어도 상위 A를 테이블로 변환할 수 있다. A 테이블에 공통 속성을 저장하고 B·C 테이블이 A의 기본키를 참조하게 하면, 같은 A 개체의 키가 두 하위 테이블에 모두 나타나는 방식으로 중복 소속을 표현할 수 있다. 중복 여부는 하위 엔티티의 소속 제약이며, A의 테이블 변환 가능 여부를 결정하지 않는다. 다만 현재 그림은 d이므로 실제로는 중복을 허용하지 않는다.

라. O: 전체 참여이므로 A에만 속하여 하위 테이블에 저장되지 않는 개체가 없다. 따라서 B와 C의 테이블에 각각 A의 기본키와 공통 속성, 해당 하위 엔티티의 고유 속성을 넣으면 A 테이블을 별도로 두지 않아도 된다. 또한 분할 조건에 따라 **각 개체를 한 하위 테이블에만 저장**하므로 동일 개체의 공통 속성을 두 곳에 중복 저장할 필요가 없다.

적절한 보기는 가와 라이므로 ②와 일치한다.

NoteBase

문 20. 릴레이션 R(A,B,C,D)에 대한 함수종속이 다음과 같을 때, R의 수퍼키로만 나열된 것은?

ACD → B, B → A

① ABD, BCD

② ABC, ABD

③ BCD, ABCD

④ ABC, ABCD

정답 3번

간단한 해설

BCD는 B → A에 따라 전체 속성을 결정하고, ABCD는 처음부터 전체 속성을 포함한다. **수퍼키는 최소성을 요구하지 않으므로** 두 집합 모두 수퍼키이다.

계산 과정

속성 집합 X가 수퍼키인지 판단하려면 주어진 함수종속을 반복 적용하여 속성 폐포를 구한다. 폐포란 X로부터 결정할 수 있는 모든 속성의 집합이며, $X^+ = \{A, B, C, D\}$이면 수퍼키이다. **수퍼키에는 최소성을 요구하지 않는다**.

BCD: 처음에 B, C, D를 알고 있고, 함수종속 B → A에 따라 A를 추가할 수 있다. 따라서 $(BCD)^+ = \{A, B, C, D\}$이므로 BCD는 수퍼키이다.

ABCD: 시작부터 릴레이션의 모든 속성을 포함하므로 $(ABCD)^+ = \{A, B, C, D\}$이다. BCD에 불필요한 A가 더해져 최소성을 갖추지 못하더라도 **유일 식별 능력은 유지**되므로 ABCD 역시 수퍼키이다.

ABD: B → A를 적용해도 이미 있는 A만 얻으며, C가 없어서 ACD → B로 새로운 속성을 얻을 수도 없다. 따라서 $(ABD)^+ = \{A, B, D\}$로 C를 결정하지 못하므로 수퍼키가 아니다.

ABC: B → A를 적용해도 속성이 늘지 않고, D가 없어서 ACD → B로도 추가할 속성이 없다. 따라서 $(ABC)^+ = \{A, B, C\}$로 D를 결정하지 못하므로 수퍼키가 아니다.

주어진 함수종속의 오른쪽에는 C와 D가 없으므로, 두 속성은 다른 속성에서 새로 얻을 수 없다는 점도 확인할 수 있다. 선택지에 등장한 집합 중 수퍼키인 것은 BCD와 ABCD이며, 이 둘만 나열한 것은 ③이다.

NoteBase

문 21. 타임 스탬프를 이용한 동시성 제어 기법에 대한 설명으로 적절하지 않은 것은?

① 타임스탬프는 시스템이 트랜잭션에 부여하기 위해 생성한 고유한 값이다.

② 트랜잭션들의 타임스탬프로 직렬가능성을 결정한다.

③ 로킹(locking)이 필요 없으므로 교착상태(deadlock)가 발생하지 않는다.

④ 연쇄복귀(cascading rollback)를 방지한다.

정답 4번

간단한 해설

기본 타임스탬프 순서 규약은 트랜잭션의 직렬 실행 순서를 지키지만, **미커밋 값을 읽는 것까지 금지하지는 않는다**. 그 값을 쓴 트랜잭션이 중단되면 이를 읽은 트랜잭션도 복귀해야 할 수 있으므로 연쇄복귀가 가능하다.

선지별 해설

타임스탬프를 이용한 동시성 제어는 트랜잭션에 부여한 순서 값에 따라 충돌 연산의 실행 순서를 관리한다. 여기서는 별도의 커밋 대기 조건을 추가하지 않은 기본 타임스탬프 순서 규약을 기준으로 판단한다.

① O: 타임스탬프는 시스템이 각 트랜잭션에 부여하는 **고유한 순서 값**이다. 시스템 시계나 증가하는 논리적 카운터 등을 이용하며, 서로 다른 트랜잭션의 선후를 구분할 수 있어야 한다. 반드시 달력의 날짜·시각 형식이어야 하는 것은 아니다.

② O: 타임스탬프 순서가 트랜잭션의 **직렬 실행 순서의 기준**이 된다. 더 작은 타임스탬프를 가진 트랜잭션이 먼저 실행된 직렬 스케줄과 동등하도록 충돌하는 읽기·쓰기를 제어한다. 각 데이터의 읽기·쓰기 타임스탬프와 요청한 트랜잭션의 값을 비교하여 이 순서를 위반하는 연산을 허용하지 않는다.

③ O: 기본 타임스탬프 순서 규약에서는 읽기·쓰기 전에 잠금을 얻고 기다리는 대신, 순서 위반이 발생하면 해당 트랜잭션을 중단하고 재시작한다. 따라서 **잠금 해제를 서로 기다리는 대기 고리**가 형성되지 않아 교착상태가 발생하지 않는다. 다만 반복적인 중단·재시작으로 어떤 트랜잭션이 오래 완료되지 못하는 기아현상까지 자동으로 방지하는 것은 아니다.

④ X: 기본 타임스탬프 순서 규약은 직렬가능성을 보장하지만 **연쇄복귀 방지를 보장하지는 않는다**. 예를 들어 먼저 시작한 트랜잭션이 쓴 미커밋 값을 뒤의 트랜잭션이 읽은 후, 앞의 트랜잭션이 중단되면 그 값을 읽은 뒤의 트랜잭션도 복귀해야 할 수 있다. 연쇄복귀를 막으려면 커밋되지 않은 값을 다른 트랜잭션이 읽지 못하게 하는 추가 규약이 필요하다. 이러한 제한을 둔 강화된 변형과 기본 규약을 구분해야 한다.

연쇄복귀를 방지한다고 단정한 ④가 적절하지 않다.

NoteBase

문 22. 2단계 로킹(two-phase locking)에 대한 설명으로 적절하지 않은 것은?

① 확장단계에서 트랜잭션은 lock만 수행할 수 있고 unlock은 수행할 수 없다.

② 축소단계에서 트랜잭션은 unlock만 수행할 수 있고 lock은 수행할 수 없다.

③ 직렬 가능한 스케줄을 생성함으로써 교착상태를 피할 수 있다.

④ 2단계 로킹 규약은 직렬 가능성을 보장하는 충분조건이지 필요조건은 아니다.

정답 3번

간단한 해설

2단계 로킹은 트랜잭션의 실행 결과가 직렬 실행과 동등하도록 하지만, **서로 보유한 잠금을 기다리는 교착상태까지 막지는 못한다**. 직렬 가능성과 교착상태 방지는 서로 다른 성질이다.

선지별 해설

① O: 확장단계에서는 필요한 잠금을 새로 획득할 수 있지만, 이미 획득한 잠금을 해제할 수 없다. 첫 잠금 해제가 일어나기 전까지가 확장단계이며, 여기서 'lock만 수행'한다는 것은 **잠금 획득·해제 연산 사이의 제한**을 뜻한다. 허용된 잠금 아래의 읽기·쓰기까지 금지한다는 뜻은 아니다.

② O: 축소단계에서는 보유한 잠금을 해제할 수 있지만 새로운 잠금을 획득할 수 없다. 즉, **한 번 잠금을 해제한 뒤에는 다시 잠금을 얻지 않는다**는 규칙이 두 단계를 구분한다.

③ X: 기본적인 2단계 로킹은 충돌 직렬 가능성을 보장하지만 **교착상태의 부재까지 보장하지는 않는다**. 예를 들어 한 트랜잭션이 A의 배타잠금을, 다른 트랜잭션이 B의 배타잠금을 가진 채 서로 상대가 가진 잠금을 기다리면 교착상태가 생길 수 있다. 두 트랜잭션 모두 아직 잠금을 해제하지 않은 확장단계이므로 이러한 대기는 2단계 로킹 규칙과 양립한다.

④ O: 모든 트랜잭션이 2단계 로킹 규약을 지키면 생성되는 스케줄은 충돌 직렬 가능하므로 충분조건이다. 그러나 직렬 가능한 모든 스케줄이 반드시 이 규약으로만 생성되는 것은 아니다. 예를 들어 트랜잭션들이 실제로 하나씩 차례로 실행된다면, 각 트랜잭션 내부에서 잠금 해제 후 다른 잠금을 획득하여 2단계 로킹을 따르지 않아도 전체 스케줄은 직렬 스케줄이다.

NoteBase

문 23. 조인 알고리즘에 대한 설명으로 가장 적절하지 않은 것은?

① 블록 중첩루프 조인(block nested-loop join)은 조인 대상 릴레이션을 모두 메모리에 올릴 수 없는 경우에 조인 대상 릴레이션에 대해서 블록 단위로 릴레이션을 처리하여 블록의 접근비용을 줄인다.

② 인덱스를 사용한 중첩루프 조인(indexed nested-loop join)은 외부 반복문의 조인 속성에 인덱스가 구축되어 있을 경우 효과적이다.

③ 합병조인(merge join)은 조인 속성에 같은 값을 갖는 모든 투플들의 그룹이 주메모리 내에 한번에 올라갈 수 있을 때 매우 효율적이다.

④ 해시조인(hash join)은 조인 대상 릴레이션의 조인 속성에 대해 해시함수를 이용하여 동일한 해시 값을 가지는 집합으로 분할한다.

정답 2번

간단한 해설

인덱스 중첩루프 조인은 외부 릴레이션의 각 투플에 대해 **내부 릴레이션의 조인 속성 인덱스**로 대응 투플을 찾는다. 외부 릴레이션의 인덱스를 핵심 조건으로 제시한 부분이 잘못되었다.

NoteBase

선지별 해설

조인은 조건에 맞는 두 릴레이션의 투플을 연결하는 연산이다. 중첩루프 조인에서 외부 릴레이션은 반복의 기준이 되는 쪽이고, 내부 릴레이션은 그 기준값에 맞는 투플을 반복해서 찾는 쪽이다. 각 알고리즘은 내부 탐색이나 디스크 입출력 비용을 줄이는 방법이 다르다.

① O: 블록 중첩루프 조인은 외부 릴레이션의 투플을 하나씩 따로 처리하는 대신 블록이나 메모리에 들어가는 블록 묶음으로 읽는다. 내부 릴레이션의 블록을 읽었을 때 메모리에 있는 여러 외부 투플과 함께 비교하므로 **읽은 블록을 여러 비교에 재사용**할 수 있다. 두 릴레이션 전체가 메모리에 들어가지 않아도 반복적인 블록 접근 비용을 줄일 수 있다.

② X: 외부 투플의 조인 속성값을 검색키로 삼아 **내부 릴레이션의 인덱스**를 탐색한다. 내부 릴레이션의 조인 속성에 유용한 인덱스가 있으면 매번 내부 전체를 순차 탐색하지 않고 일치하는 투플에 접근할 수 있다. 따라서 효과적인 인덱스의 위치는 외부 반복문의 릴레이션이 아니라 내부에서 검색하는 릴레이션이다.

③ O: 합병조인은 두 입력이 조인 속성 순서로 정렬되어 있다는 전제에서 같은 키 값을 갖는 그룹을 맞추며 진행한다. 정렬되지 않았다면 먼저 정렬해야 한다. 중복 키의 그룹이 메모리에 들어가면 그 그룹을 유지한 채 대응 투플들을 결합할 수 있어 **그룹의 반복적인 디스크 재읽기**를 줄인다. 필요한 정렬 비용과 키의 중복 정도가 실제 성능에 영향을 준다.

④ O: 해시조인은 조인 속성에 같은 해시함수를 적용하여 같은 키 값을 가진 투플들이 대응하는 파티션에 들어가도록 한다. 이후 각 파티션 안에서 해시 테이블을 구성하고 상대 투플을 탐색하여 동등 조건을 만족하는 쌍을 결합한다. 서로 다른 값도 같은 해시값을 가질 수 있으므로, **해시값의 일치만으로 조인하지 않고 실제 키 값도 비교**해야 한다.

NoteBase

문 24. 로그 레코드를 주메모리 내 로그 버퍼에 임시로 저장하였다가, 하나의 출력 연산으로 로그 버퍼 내에 있는 모든 로그 레코드를 저장장치에 출력하고자 한다. 장애 발생 시 트랜잭션의 원자성을 보장하기 위한 작업으로 가장 적절하지 않은 것은?

① 트랜잭션 T의 로그 레코드 <T, commit>이 로그 버퍼에 저장되면 트랜잭션 T는 커밋(commit) 상태가 된다.

② 트랜잭션 T와 관련된 모든 로그 레코드가 <T, commit>보다 먼저 안전한 저장장치에 출력되어야 한다.

③ 변경된 데이터 블록 D의 데이터 항목과 관련된 모든 로그 레코드가 블록 D보다 먼저 안전한 저장장치에 출력되어야 한다.

④ 로그 레코드를 출력해야 할 필요가 있는 경우 로그 레코드가 한 블록을 채우기에 부족하면, 메모리 내의 모든 로그 레코드를 한꺼번에 모아 안전한 저장장치에 출력한다.

정답 1번

간단한 해설

로그 버퍼는 주메모리에 있으므로 장애 때 내용이 사라질 수 있다. 커밋 로그를 버퍼에 넣는 것만으로는 부족하며, **관련 로그와 커밋 로그가 안전한 저장장치에 기록된 뒤** 커밋을 확정해야 한다.

선지별 해설

① X: 주메모리의 로그 버퍼에 커밋 레코드를 기록했더라도 아직 영속적으로 보존된 것은 아니다. 이 상태에서 장애가 나면 커밋 사실과 변경 내역을 잃을 수 있으므로, **T의 선행 로그와 커밋 로그까지 안전한 저장장치에 출력**한 뒤 커밋을 확정해야 한다. 커밋을 위해 변경된 모든 데이터 블록 자체를 즉시 저장할 필요는 없으며, 필요한 변경은 보존된 로그로 복구할 수 있다.

② O: 커밋 레코드가 안전하게 남아 있다면 그 트랜잭션을 복구하는 데 필요한 앞선 로그도 남아 있어야 한다. 따라서 T의 변경 로그를 커밋 레코드에 앞선 순서로 출력하여, **커밋 기록만 있고 선행 변경 기록은 없는 상태**가 생기지 않게 해야 한다. 순서를 보존한 로그 묶음을 한 번에 출력할 수 있으므로 각 레코드를 별도의 출력 연산으로 저장해야 한다는 뜻은 아니다.

③ O: 변경된 데이터 블록보다 관련 로그를 먼저 안전한 저장장치에 기록하는 것이 **선행 기록, 즉 WAL 원칙**이다. 데이터만 먼저 저장되고 복구용 로그가 사라지면 미완료 트랜잭션의 변경을 되돌리는 등의 회복 작업을 할 수 없으므로, 블록 D를 출력하기 전에 D의 변경을 설명하는 로그가 보존되어야 한다.

④ O: 커밋이나 데이터 블록 출력 때문에 로그를 강제로 저장해야 할 때는 **로그 블록이 가득 찰 때까지 기다리지 않는다**. 이 문제의 일괄 출력 방식에서는 현재 버퍼에 모인 로그를 함께 출력하여 앞선 기록을 빠짐없이 보존한다. 블록의 일부가 비어 있어도 복구에 필요한 로그의 저장 시점을 지키는 것이 우선이다.

NoteBase

문 25. 그림자 페이징(shadow paging) 회복기법에 대한 설명으로 적절하지 않은 것을 모두 고른 것은?

가. 트랜잭션이 완료되기 전에는 그림자 페이지를 변경할 필요가 전혀 없다.
나. 트랜잭션의 취소 연산이 매우 간단하다.
다. 데이터베이스 페이지의 물리적 위치가 변경되므로 페이지의 지역성이 감소할 수 있다.
라. 페이지 테이블의 크기가 클수록 그림자 페이지 테이블 관리를 위한 오버헤드가 감소한다.
마. 병행수행되는 트랜잭션의 숫자가 많을수록 상대적으로 좋은 성능을 갖는다.

① 가, 나, 다
② 나, 마
③ 가, 다, 라
④ 라, 마

정답 4번

간단한 해설

그림자 페이징은 변경 전 페이지를 보존하여 취소를 단순하게 하지만, **페이지 테이블이 커지면 관리 비용이 증가**한다. 여러 갱신 트랜잭션의 페이지 테이블 전환을 조정하기도 어려워, 병행수행 수가 많을수록 유리하다고 볼 수 없다.

NoteBase

보기별 해설

그림자 페이징은 변경 전 상태를 가리키는 그림자 페이지 테이블과 작업 중인 현재 페이지 테이블을 분리한다. 변경은 새 페이지에 기록하고, 커밋할 때 변경 페이지와 현재 테이블을 먼저 영속화한 뒤 기준 포인터를 새 테이블로 전환한다.

가. O: 트랜잭션 수행 중에는 **변경 전 상태를 담은 그림자 페이지를 보존**한다. 수정할 페이지는 새 공간에 복사하여 그 복사본을 갱신하고, 현재 페이지 테이블만 새 위치를 가리키게 한다. 따라서 취소 시 돌아갈 기준인 그림자 페이지와 그림자 페이지 테이블을 수행 도중 바꿀 필요가 없다.

나. O: 취소할 때는 **현재의 미커밋 변경을 버리고 보존된 상태를 사용**하면 된다. 기존 페이지를 덮어쓰지 않았으므로 로그의 이전 값을 따라 변경 하나하나를 역연산하는 과정이 필요하지 않다. 다만 더 이상 사용하지 않는 새 페이지의 공간 회수는 별도로 필요할 수 있다.

다. O: 변경된 내용을 원래 페이지에 덮어쓰지 않고 **다른 물리적 위치의 새 페이지에 기록**하므로, 관련된 페이지들이 디스크에서 서로 떨어질 수 있다. 이와 같은 단편화는 가까운 위치의 페이지를 연속해서 접근하는 지역성을 낮출 수 있다.

라. X: 페이지 테이블이 클수록 복사·저장·관리해야 할 항목이 많아져 일반적으로 **관리 오버헤드가 증가**한다. 트리 구조를 이용해 변경된 경로만 복사하는 등 비용을 줄이는 구현은 가능하지만, 테이블이 커진다는 이유 자체로 오버헤드가 감소하지는 않는다.

마. X: 기본 그림자 페이징은 **많은 갱신 트랜잭션의 병행수행에 유리한 방식이 아니다**. 여러 트랜잭션이 서로 다른 현재 페이지 테이블을 갱신하면 다른 트랜잭션의 변경을 잃지 않도록 커밋과 테이블 전환을 조정해야 하므로 확장이 복잡해진다. 병행수행을 지원하는 개선은 가능하지만, 트랜잭션 수가 많을수록 성능이 좋아진다고 일반화할 수 없다.

적절하지 않은 보기는 라·마이므로 ④에 해당한다.

NoteBase

문 26. SQL 표준에서 정의하고 있는 4가지 트랜잭션 격리 수준(isolation level)에 대한 설명으로 적절하지 않은 것은?

① 트랜잭션의 동시 실행성 측면에서 가장 낮은 성능을 보이는 격리수준은 SERIALIZABLE이다.

② 오손읽기(dirty read)가 발생할 수 있는 격리 수준은 READ UNCOMMITTED뿐이다.

③ 반복 가능한 읽기를 지원하는 격리 수준은 REPEATABLE READ와 SERIALIZABLE 2가지 뿐이다.

④ 유령읽기(phantom read)가 발생할 수 있는 격리 수준은 READ UNCOMMITTED와 READ COMMITTED뿐이다.

정답 4번

간단한 해설

SQL 표준에서 유령읽기는 READ UNCOMMITTED, READ COMMITTED뿐 아니라 **REPEATABLE READ에서도 허용된다.** 따라서 적절하지 않은 설명은 ④이다.

선지별 해설

① O SERIALIZABLE은 직렬 실행과 같은 결과를 요구하여 동시 실행에 가장 강한 제약을 둔다. 격리성과 동시 실행성의 일반적인 상충 관계를 비교한 설명이다. 실제 처리 속도가 모든 제품·작업에서 반드시 가장 느리다는 뜻은 아니다.

② O 오손읽기는 다른 트랜잭션이 아직 커밋하지 않은 데이터를 읽는 것이다. SQL 표준의 네 수준 중 이를 허용하는 것은 READ UNCOMMITTED뿐이다.

③ O 이미 읽은 행을 다시 읽을 때 다른 트랜잭션의 커밋 때문에 값이 달라지는 비반복 읽기를 방지하는 수준은 REPEATABLE READ와 SERIALIZABLE이다. 이때 반복 가능한 읽기와 검색 조건에 해당하는 행 집합의 불변은 구분해야 한다.

④ X 유령읽기는 같은 검색 조건으로 다시 조회했을 때 다른 트랜잭션의 커밋으로 결과 행 집합이 달라지는 현상이다. 표준상 REPEATABLE READ에서도 허용되므로, 제시된 두 수준에 **REPEATABLE READ를 추가**해야 옳다. SERIALIZABLE에서는 허용되지 않는다.

여기서는 SQL 표준의 최소 보장 수준을 기준으로 판단한다. 개별 DBMS는 더 강하게 구현할 수 있으며, 예를 들어 PostgreSQL의 REPEATABLE READ는 유령읽기도 방지한다.

NoteBase

문 27. 동시간대 함께 구매가 빈번한 상품을 묶음으로 판매하거나 인접한 진열대에 두어 매출을 올리는 데이터 마이닝 분석 기법으로 가장 적절한 것은?

① 분류분석

② 연관분석

③ 군집분석

④ 회귀분석

정답 2번

간단한 해설

연관분석은 거래 자료에서 함께 구매되는 상품들의 관계를 찾는 기법이다. 발견한 공동 구매 패턴을 묶음 판매나 인접 진열에 활용한다는 점이 결정적인 단서이다.

핵심 풀이

연관분석: 거래 자료에서 **함께 나타나는 항목들의 관계**를 찾는다. 장바구니 분석이 대표적이며, 특정 상품을 구매한 거래에서 다른 상품도 함께 구매되는 경향을 분석한다. 문항의 '함께 구매가 빈번한 상품'이라는 단서를 이용하여 묶음 판매와 진열 위치를 정하는 것이 이에 해당한다. 함께 구매된다는 관계가 곧 두 상품 사이의 인과관계를 뜻하는 것은 아니다.

분류분석: 미리 정해진 범주를 기준으로 자료가 어느 범주에 속하는지 판별한다. 예를 들어 이미 구분된 정상 거래·부정 거래 자료를 학습하여 새 거래의 범주를 예측하는 방식이다.

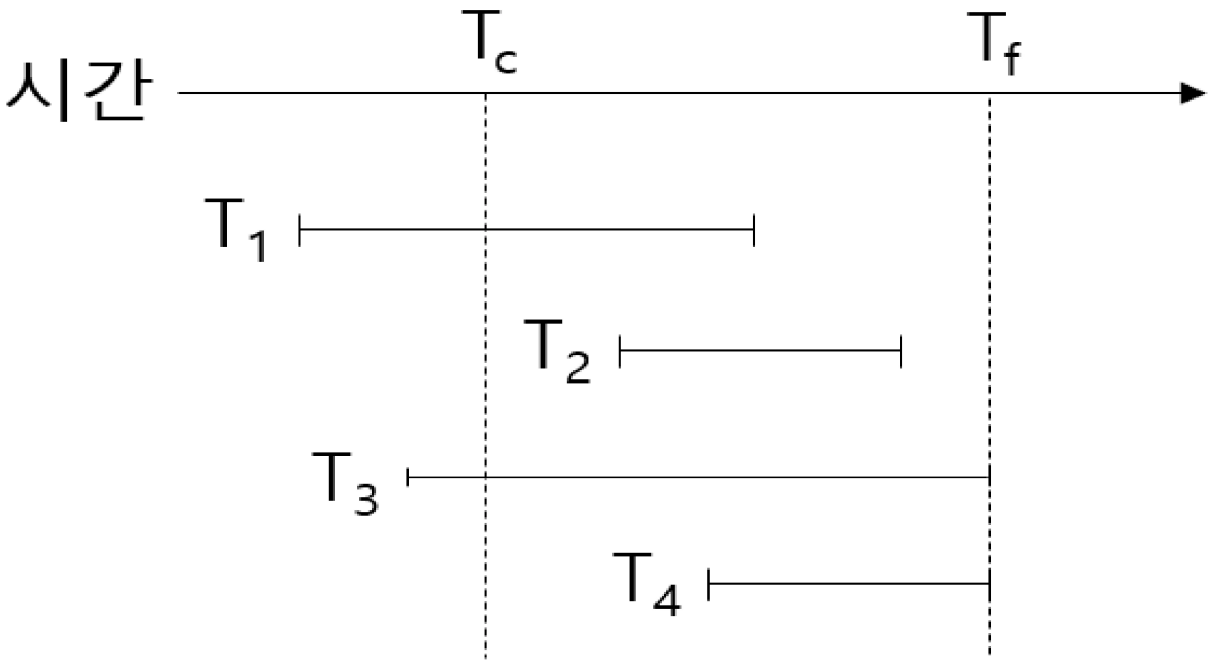
군집분석: 정답 범주가 미리 주어지지 않은 자료를 특성의 유사성에 따라 여러 집단으로 묶는다. 이 문항은 비슷한 고객이나 상품 집단을 찾는 것보다 **서로 함께 구매되는 상품의 관계**를 찾는 데 초점이 있다.

회귀분석: 설명변수와 결과변수의 관계를 이용하여 매출액 등 수치형 값을 설명하거나 예측한다. 문항은 매출액 자체의 예측식을 구하는 것이 아니라 공동 구매 패턴을 찾아 판매 전략에 활용하는 사례이다.

따라서 가장 적절한 분석 기법은 연관분석인 ②이다.

NoteBase

문 28. 검사점(checkpoint) 회복 기법과 관련하여 트랜잭션 수행 과정이 다음과 같을 때 데이터 복원을 위해 수행되는 작업으로 적절하지 않은 것은? (단, T_c는 검사점, T_f는 장애시점을 의미한다.)



① T_1은 T_c 이후에 일어난 변경 부분에 대해서만 redo 한다.

② T_2는 처음부터 끝까지 redo 한다.

③ T_3은 T_c 이후에 일어난 변경 부분에 대해서만 undo 한다.

④ T_4는 처음부터 끝까지 undo 한다.

정답 3번

간단한 해설

검사점은 저장을 위한 기준 시점이며 **진행 중인 트랜잭션의 커밋을 뜻하지 않는다**. 장애 때 미완료인 T_3은 검사점 이전의 변경까지 포함하여 시작 이후의 모든 변경을 undo 해야 한다.

NoteBase

선지별 해설

① O: 그림에서 T_1은 검사점 T_c 이전에 시작하여 T_c 이후, 장애시점 T_f 이전에 정상 완료되었다. 이 문제의 검사점 회복 방식에서는 검사점까지의 변경이 저장장치에 반영되어 있으므로, 완료된 T_1에 대해서는 **검사점 이후의 변경만 redo**하여 확정된 결과를 복원한다.

② O: T_2는 검사점 이후에 시작하여 장애 전에 정상 완료되었다. 수행된 모든 변경이 검사점 이후에 속하므로, 로그를 따라 **처음부터 완료 시점까지 redo**한다. 이는 완료된 트랜잭션의 결과가 장애로 유실되지 않도록 하는 작업이다.

③ X: T_3은 검사점 이전에 시작했지만 장애시점까지 정상 완료되지 않았다. 검사점 이전의 변경이 저장장치에 기록되었더라도 커밋되지 않은 트랜잭션의 일부 결과를 남겨 둘 수는 없다. 따라서 검사점 이후만 되돌리는 것이 아니라 **T_3의 시작부터 장애 때까지 수행된 모든 변경을 undo**하여 원자성을 보장한다.

④ O: T_4는 검사점 이후에 시작하여 장애시점까지 완료되지 않았다. 따라서 T_4가 시작한 뒤 장애 때까지 수행한 **전체 변경을 undo**한다. 여기서 '끝까지'는 장애 이후의 미실행 작업이 아니라 장애 전에 실제로 수행된 마지막 변경까지를 뜻한다.

NoteBase

문 29. 트랜잭션의 특성에 대한 설명으로 적절하지 않은 것은?

① 트랜잭션이 수행되는 도중에는 데이터베이스가 일관된 상태를 갖지 않을 수 있으나 트랜잭션이 완료되면 일관된 상태로 변환된다.

② 각 트랜잭션은 시스템 내에서 동시에 수행되고 있는 다른 트랜잭션에 영향을 줄 수 없으나 참조할 수는 있다.

③ 트랜잭션들이 동시에 수행되더라도 그 결과는 어떤 순서에 따라 트랜잭션들을 하나씩 순서대로 수행한 결과와 같아야 한다.

④ 성공적으로 트랜잭션이 완료되면 이 트랜잭션이 갱신한 것은 그 후에 시스템이 고장 난 경우에도 손실되지 않아야 한다.

정답 2번

간단한 해설

트랜잭션의 고립성은 다른 트랜잭션이 실행 도중의 **미완료 중간 결과를 참조하여 영향을 받지 않도록** 하는 성질이다. 동시에 실행 중인 트랜잭션의 중간 결과를 자유롭게 참조할 수 있다는 설명은 고립성과 맞지 않는다.

NoteBase

선지별 해설

트랜잭션은 하나의 논리적 작업을 이루는 연산 묶음이다. 기본 특성인 ACID는 전부 수행하거나 전부 취소하는 원자성, 유효한 상태를 유지하는 일관성, 동시 실행의 간섭을 막는 고립성, 확정된 결과를 보존하는 지속성을 뜻한다.

① O: 올바르게 작성된 트랜잭션은 일관된 데이터베이스 상태에서 출발하여 성공적으로 완료되면 다시 일관된 상태를 만들어야 한다. 계좌 이체에서 한 계좌의 차감 뒤 다른 계좌의 증가가 아직 이루어지지 않은 중간 단계처럼, 작업 도중에는 전체 업무 규칙이 일시적으로 성립하지 않을 수 있다. 이 중간 상태를 다른 트랜잭션에 부적절하게 노출하지 않는 것이 고립성의 역할이다.

② X: 고립성은 다른 트랜잭션이 아직 완료하지 않은 연산의 **중간 결과를 참조하여 의존하지 않도록** 한다. 미확정 변경을 읽으면 그 변경이 취소될 때 잘못된 값을 이용한 작업이 남을 수 있다. 다른 트랜잭션이 확정한 데이터를 정해진 동시성 제어 규칙에 따라 읽는 것까지 금지한다는 뜻은 아니다.

③ O: 기본적인 고립성의 목표는 동시 실행 결과가 트랜잭션들을 어떤 순서로 하나씩 실행한 결과와 같게 하는 **직렬 가능성**이다. 물리적으로 한 번에 하나만 실행해야 하는 것은 아니며, 연산이 교차 실행되어도 결과가 적절한 직렬 실행과 같으면 된다. 실제 DBMS의 완화된 격리수준을 모두 이 수준과 동일하다고 보는 것은 아니다.

④ O: 지속성은 **성공적으로 커밋된 결과**가 이후 시스템 장애로 사라지지 않아야 한다는 성질이다. DBMS는 로그와 복구 절차 등을 통해 확정된 변경을 보존하거나 복구한다. 아직 커밋하지 않은 중간 변경까지 반드시 보존해야 한다는 의미는 아니다.

NoteBase

문 30. 트랜잭션 T1, T2, T3에 대한 스케줄 S1, S2 중에서 직렬 가능한(serializable) 스케줄을 모두 고른 것은? (단, r은 read 연산을, w는 write 연산을 의미한다.)

<트랜잭션>
T1: r1(x); r1(y); w1(x);
T2: r2(y); r2(z); w2(y);
T3: r3(x); r3(z); w3(z);

<스케줄>
S1: r1(x); r2(y); r3(x); r1(y); r2(z); r3(z); w1(x); w2(y); w3(z);
S2: r1(x); r2(y); r1(y); r3(x); r3(z); w1(x); w3(z); r2(z); w2(y);

① 없음
② S1
③ S2
④ S1, S2

정답 3번

간단한 해설

S1의 우선순위 그래프에는 T3 → T1 → T2 → T3의 사이클이 있다. S2는 **T3 → T1 → T2** 순서로 직렬화할 수 있으므로 정답은 ③ **S2**이다.

NoteBase

계산 과정

충돌은 서로 다른 트랜잭션이 같은 데이터에 접근하고, 두 연산 중 적어도 하나가 쓰기일 때 발생한다. 먼저 실행되는 연산의 트랜잭션에서 나중 연산의 트랜잭션으로 간선을 그린다. 읽기끼리는 충돌하지 않는다.

S1

x: r3(x)가 w1(x)보다 먼저이므로 T3 → T1
y: r1(y)가 w2(y)보다 먼저이므로 T1 → T2
z: r2(z)가 w3(z)보다 먼저이므로 T2 → T3
따라서 T3 → T1 → T2 → T3의 사이클이 있어 충돌 직렬 가능하지 않다. 또한 세 읽기는 각각 상대 트랜잭션의 쓰기 전 값을 읽으므로, 같은 읽기 결과를 보존하는 직렬 순서에도 위 세 순서 제약이 필요하다. 이를 동시에 만족하는 직렬 순서는 없다.

S2

x: r3(x)가 w1(x)보다 먼저이므로 T3 → T1
y: r1(y)가 w2(y)보다 먼저이므로 T1 → T2
z: w3(z)가 r2(z)보다 먼저이므로 T3 → T2
사이클이 없으며 **T3 → T1 → T2** 순서의 직렬 실행과 동등하다. S1과 달리 z의 충돌 순서가 T3 → T2가 된 것이 핵심이다.

직렬 가능한 것은 S2뿐이므로 정답은 ③이다.

NoteBase

문 31. OLAP의 특징으로 가장 적절하지 않은 것은?

① ‘why’보다 ‘what’에 중점을 두어 분석한다.

② 분석을 위해 활용되는 정보의 형태는 다차원이다.

③ 최종사용자는 중간매개자(정보시스템부서)나 매개체(리포트) 없이 온라인상에서 직접 데이터에 접근한다.

④ 최종사용자의 기업 상황 이해와 의사결정을 위한 전략적 방향 설정에 활용된다.

정답 1번

간단한 해설

OLAP은 다차원 자료를 반복적으로 조회·분석하여 단순한 현황 확인을 넘어 **결과가 나타난 원인과 배경을 탐색**하게 한다. 따라서 'why보다 what에 중점'이라는 설명은 OLAP의 분석 지향을 거꾸로 제시한 것이다.

선지별 해설

OLAP은 온라인 분석 처리로, 사용자가 축적된 데이터를 여러 차원과 집계 수준에서 대화식으로 분석하여 의사결정에 활용하는 기술이다.

① X: OLAP은 단순히 '무슨 일이 일어났는가(what)'라는 집계 결과를 확인하는 데 그치지 않고, **'왜 그런 결과가 나왔는가(why)'를 탐색하는 분석**에 중점을 둔다. 예를 들어 매출 감소를 확인한 뒤 지역·상품·기간별로 세분하여 어떤 부분이 감소에 기여했는지 살펴본다. 이는 사용자가 원인과 배경을 분석한다는 뜻이며, OLAP만으로 인과관계가 자동 입증된다는 뜻은 아니다.

② O: OLAP은 매출액 등의 측정값을 시간·지역·상품과 같은 **여러 차원으로 분석**한다. 상세 수준으로 내려가는 드릴다운, 상위 수준으로 집계하는 롤업, 특정 조건을 선택하는 슬라이스·다이스 등을 통해 같은 정보를 다양한 관점에서 살펴본다. 다차원은 분석의 논리적 관점이며, 실제 저장 방식이 언제나 다차원 배열로 고정된다는 뜻은 아니다.

③ O: 최종사용자는 분석 도구에서 조건이나 관점을 바꾸면서 **대화식으로 직접 조회·분석**할 수 있다. 매번 정보시스템부서에 정형 리포트 작성을 의뢰하고 그 결과만 기다리는 방식과 구별된다. 여기서 직접 접근한다는 것은 분석 도구·서버나 접근권한 통제가 없다는 의미가 아니라, 사용자가 분석을 주도한다는 의미이다.

④ O: OLAP은 과거와 현재의 실적을 여러 관점에서 비교하여 추세·차이·문제 영역을 파악하게 하므로 **경영 의사결정과 전략 수립**을 지원한다. 예를 들어 지역별 수익성을 비교하여 자원 배분이나 판매전략을 검토하는 데 활용할 수 있다. 개별 주문의 등록·수정처럼 일상 거래를 처리하는 OLTP와 주된 목적이 다르다.

OLAP의 분석 지향을 'why보다 what에 중점'이라고 설명한 ①이 가장 적절하지 않다.

NoteBase

문 32. 다차원 데이터의 지원을 위한 SQL 구문으로 가장 적절하지 않은 것은?

① CUBE

② ROLLUP

③ GROUPING SETS

④ EXCEPTION

정답 4번

간단한 해설

CUBE, ROLLUP, GROUPING SETS는 여러 차원과 집계 수준을 함께 다루는 GROUP BY 확장 기능이다. EXCEPTION은 **오류에 대한 예외 처리**에 사용되므로 다차원 집계 지원 구문에 해당하지 않는다.

선지별 해설

① O: CUBE는 지정한 집계 차원의 **가능한 모든 조합**에 대한 그룹을 만든다. 예를 들어 GROUP BY CUBE(A, B)는 (A, B), (A), (B), ()에 따른 집계를 제공한다. 여기서 ()는 모든 행을 하나의 그룹으로 보는 전체 집계이다.

② O: ROLLUP은 지정한 열의 순서에 따라 세부 집계에서 상위 수준의 소계와 전체 집계로 올라간다. 예를 들어 GROUP BY ROLLUP(A, B)는 (A, B), (A), ()에 따른 집계를 제공하며, B만의 집계는 따로 만들지 않는다. 따라서 **열의 순서에 따른 계층적 집계**에 적합하다.

③ O: GROUPING SETS는 필요한 그룹화 조합을 직접 지정한다. 예를 들어 GROUP BY GROUPING SETS((A), (B))는 A별 집계와 B별 집계를 함께 구한다. 이 예에서는 지정하지 않은 (A, B) 집계나 전체 집계를 자동으로 추가하지 않는다.

④ X: EXCEPTION은 PL/pgSQL 같은 절차형 언어에서 실행 중 발생한 오류를 처리하는 데 사용하는 구문이다. **다차원 집계 수준이나 그룹 조합을 생성하는 구문이 아니다**. 다차원 집계에는 CUBE, ROLLUP, GROUPING SETS 같은 GROUP BY 확장 기능을 사용한다.

NoteBase

문 33. 데이터베이스 시스템의 성능평가 표준인 TPC 벤치마크의 설명으로 적절하지 않은 것은?

① 릴레이션의 투플 수는 고정된 숫자 대신 초당 요청된 트랜잭션 수의 배수로 정의한다.

② 성능의 측정 기준은 처리율이며, TPS(Transactions Per Second)로 나타낸다.

③ 비즈니스 관련 응용에서는 성능을 TPS당 가격으로 측정하기도 한다.

④ 의사결정 질의의 성능 측정을 위해 특화된 벤치마크는 TPC-C이다.

정답 4번

간단한 해설

TPC-C는 주문 입력 등 **온라인 트랜잭션 처리(OLTP)**를 평가하는 벤치마크이다. 복잡한 의사결정 지원 질의를 평가하는 TPC-D, TPC-H 등의 벤치마크와 구분된다.

선지별 해설

TPC는 데이터베이스 시스템을 정해진 작업 부하와 조건으로 비교하기 위한 여러 벤치마크를 마련한다. 처리량과 가격 대비 성능은 중요한 평가 요소이지만, 실제 지표의 명칭·시간 단위·데이터 규모 규칙은 벤치마크마다 다르다.

① O: TPC-A와 같은 전통적인 트랜잭션 처리 벤치마크에서는 **요청 부하와 데이터 규모를 함께 증가**시키도록 규정한다. 예를 들어 TPC-A는 설정한 초당 트랜잭션 수에 비례하여 필요한 계좌·창구·지점 레코드 수를 정한다. 고성능 시스템도 지나치게 작은 고정 데이터만으로 시험하지 않도록 하기 위한 것이다. 모든 TPC 벤치마크가 동일한 배수 규칙을 사용하는 것은 아니다.

② O: 초당 완료한 트랜잭션 수인 TPS는 트랜잭션 처리율을 나타내는 대표적인 지표이며, TPC-A는 이를 바탕으로 tpsA를 사용한다. 이는 응답시간처럼 한 작업이 걸리는 시간과 구별되는 처리량 지표이다. 다만 **모든 TPC 지표가 TPS인 것은 아니며**, TPC-C는 분당 신규 주문 트랜잭션 처리량인 tpmC를 사용한다.

③ O: 시스템 가격을 처리율로 나눈 가격 대비 성능 지표를 사용하면 단순히 가장 빠른 시스템뿐 아니라 같은 처리 능력을 확보하는 데 드는 비용도 비교할 수 있다. TPS당 가격이 낮을수록 같은 기준의 처리 능력을 더 적은 비용으로 얻는다는 의미이다. 이는 개별 트랜잭션 한 건에 청구되는 요금과는 다르며, 동일한 벤치마크와 가격 산정 기준으로 비교해야 한다.

④ X: TPC-C는 여러 종류의 주문·결제 관련 트랜잭션을 혼합하여 **OLTP 성능**을 측정한다. 의사결정 지원 질의 평가를 위한 대표적인 벤치마크는 TPC-D와 그 뒤의 TPC-H·TPC-DS 등이다. 이들은 대량의 데이터를 대상으로 하는 복잡한 분석 질의에 초점을 두므로 TPC-C의 목적과 다르다.

NoteBase

문 34. 트리거(trigger)와 주장(assertion)에 대한 설명으로 가장 적절하지 않은 것은?

① 트리거는 데이터베이스의 무결성을 유지하기 위한 도구이다.

② 주장은 트리거보다 더 절차적이다.

③ 주장이 복잡하면 유효성 검사에 상당한 오버헤드가 발생한다.

④ 트리거는 'Event–Condition–Action' 규칙으로 명시된다.

정답 2번

간단한 해설

주장은 만족해야 할 조건을 선언하고, 트리거는 사건 발생 시 수행할 동작을 지정한다. 따라서 **주장이 트리거보다 더 절차적이라는 ②가 틀리다.**

선지별 해설

① O 트리거는 지정한 데이터베이스 사건이 발생할 때 자동으로 실행되는 동작을 정의한다. 부적절한 변경을 거부하거나 관련 데이터를 함께 갱신하도록 하여 무결성을 유지하는 데 사용할 수 있다. 트리거의 용도가 무결성 유지에만 한정되는 것은 아니다.

② X 주장은 데이터베이스가 만족해야 할 조건을 선언적으로 명시한다. 반면 트리거는 어떤 사건이 발생했을 때 조건을 검사하고 어떤 동작을 수행할지 지정하므로 더 절차적이다. 따라서 **주장은 트리거보다 더 선언적이다**라고 고쳐야 한다.

③ O 복잡한 주장은 여러 행이나 테이블을 조회하고 집계·비교해야 할 수 있다. 데이터 변경 후 조건을 만족하는지 확인하는 과정에서 상당한 검사 비용이 발생할 수 있다. 선언적으로 작성한다는 것과 검사 비용이 작다는 것은 별개다.

④ O 트리거는 ECA 규칙으로 설명한다. Event는 실행 계기가 되는 삽입·삭제·갱신 등의 사건, Condition은 동작 실행 여부를 결정하는 조건, Action은 조건이 만족될 때 수행하는 동작이다. 별도 조건을 지정하지 않는 경우에는 해당 사건에 대해 항상 참인 조건으로 이해할 수 있다.

NoteBase

문 35. RAID에 대한 설명으로 가장 적절하지 않은 것은?

① 레벨 1은 장애 복구 능력이 요구되는 경우에 사용되며 레벨 0보다 빠른 데이터 입출력이 가능하다.

② 레벨 5는 하나의 패리티 드라이브 사용으로 인한 병목 현상을 개선한 것이다.

③ 레벨 6은 2차 패리티 정보를 넣어 레벨 5보다 높은 장애 대비 능력을 제공한다.

④ 레벨 0+1은 레벨 0과 레벨 1의 장점을 결합한 방식이다.

정답 1번

간단한 해설

RAID 1은 미러링으로 장애에 대비하지만, RAID 0보다 입출력이 항상 빠른 것은 아니다. **데이터를 분산하는 스트라이핑**과 **복제본을 유지하는 미러링**의 목적을 구분해야 한다.

선지별 해설

① X. RAID 1은 같은 데이터를 두 디스크에 복제하는 미러링 방식으로, 한 디스크가 고장 나도 다른 복제본으로 서비스를 계속할 수 있다. 그러나 RAID 0보다 입출력이 빠르다고 일반화할 수는 없다. RAID 0은 데이터를 여러 디스크에 나누어 병렬로 처리하고 복제·패리티 부담이 없다. RAID 1도 읽기 성능이 개선될 수 있지만 쓰기는 복제본에도 반영해야 하므로, 장애 복구 능력과 처리 속도를 같은 장점으로 묶어 판단하면 안 된다.

② O. RAID 4처럼 패리티를 하나의 전용 디스크에 집중하면 쓰기 요청이 그 디스크에 몰려 병목이 생길 수 있다. RAID 5는 패리티 블록을 여러 디스크에 분산하여 이 병목을 완화한다. 다만 패리티 계산과 갱신에 따른 쓰기 부담까지 없어지는 것은 아니다.

③ O. RAID 6은 서로 독립적인 두 종류의 패리티 정보를 저장하므로 디스크 두 개가 고장 나도 데이터를 복구할 수 있다. 디스크 한 개의 고장에 대비하는 RAID 5보다 장애 허용 능력이 높다. 두 번째 패리티는 첫 번째 패리티의 단순 복사본이 아니다.

④ O. RAID 0+1은 먼저 RAID 0 방식으로 스트라이핑한 디스크 집합을 만들고, 그 집합을 RAID 1 방식으로 미러링한다. 데이터 분산을 통한 처리 성능과 복제를 통한 장애 대비라는 두 방식의 장점을 결합한다. 미러링한 디스크 쌍들을 다시 스트라이핑하는 RAID 1+0과는 구성 순서가 다르다.

NoteBase

문 36. 분산 데이터베이스에 대한 설명으로 적절하지 않은 것은?

① 데이터의 이용빈도에 따라 한 사이트에서 다른 사이트로 데이터 저장 위치가 변경되더라도 응용 프로그램에 영향을 주지 않는다.

② 시스템의 확장이 필요한 경우 서비스를 중단시키지 않고 새로운 노드를 추가하는 것도 가능하다.

③ 여러 사이트가 공조해서 처리해야 하는 경우 메시지 교환과 조정을 위한 추가적인 작업으로 부하가 발생한다.

④ 할당 스키마는 데이터베이스가 전혀 분산되어 있지 않은 것처럼 분산 데이터베이스에 할당되어 있는 모든 데이터를 정의한다.

정답 4번

간단한 해설

분산 여부를 숨기고 전체 데이터의 논리적 구조를 정의하는 것은 **전역 개념 스키마**이다. 할당 스키마는 각 데이터 단편이나 복제본을 **어느 사이트에 저장할지** 정의한다.

NoteBase

선지별 해설

분산 데이터베이스는 논리적으로 관련된 데이터를 여러 사이트에 저장하고 하나의 체계로 관리한다. 사용자에게 분산을 숨기는 투명성과 확장성의 이점이 있지만, 사이트 사이의 통신과 조정 비용이 발생한다.

① O: **위치 투명성**은 데이터가 어느 사이트에 저장되는지를 응용 프로그램이 직접 의식하지 않게 하는 성질이다. 논리적 이름과 실제 위치의 대응을 시스템이 관리하면, 이용빈도 등에 따라 데이터를 다른 사이트로 옮겨도 응용 프로그램의 데이터 참조 방식은 유지할 수 있다. 위치 변경을 숨기는 구조가 전제된다.

② O: 분산 구조에서는 새 노드를 추가하여 저장공간과 처리능력을 늘리는 **점진적 확장**이 가능하다. 온라인 노드 추가와 데이터 재배치를 지원하는 시스템이라면 기존 서비스를 계속 제공하면서 확장할 수 있다. 모든 구현에서 아무 조정이나 비용 없이 무중단 확장이 자동 보장된다는 뜻은 아니다.

③ O: 여러 사이트에 걸친 질의나 트랜잭션은 데이터를 주고받고 처리 결과를 조정해야 하므로 **통신·조정 오버헤드**가 발생한다. 예를 들어 여러 사이트가 한 트랜잭션의 커밋 여부를 일치시키려면 추가 메시지를 교환해야 한다. 분산에 따른 병렬 처리 이익과 별개로 이러한 부하를 고려해야 한다.

④ X: 전체 데이터베이스를 분산되지 않은 하나의 논리적 데이터베이스처럼 정의하는 것은 **전역 개념 스키마**이다. 할당 스키마는 데이터 단편이나 그 복제본을 어느 사이트에 배치하는지 정의한다. 전역 스키마는 전체 데이터의 논리적 구조, 단편화 스키마는 데이터를 나누는 방법, 할당 스키마는 나눈 데이터의 저장 위치를 나타낸다는 차이가 있다.

전역 개념 스키마의 역할을 할당 스키마의 역할로 설명한 ④가 적절하지 않다.

NoteBase

문 37. 정보검색 엔진에서 주어진 검색 키워드에 관련된 문서의 수와 검색된 문서의 수가 아래 표와 같을 때, 가. ~ 다. 중 검색 성능을 바르게 계산한 것은 모두 몇 개인가?

		검색된(retrieved) 문서 수	
		YES	NO
관련된(relevant) 문서 수	YES	70	70
	NO	30	30

가. 정확률(precision) = 50%

나. 재현율(recall) = 70%

다. 정확도(accuracy) = 50%

① 0개

② 1개

③ 2개

④ 3개

정답 2번

간단한 해설

정확도는 **관련 문서를 검색한 경우와 무관한 문서를 검색하지 않은 경우**를 모두 올바른 결정으로 센다. 이 표에서는 70 + 30 = 100개가 전체 200개 중 올바른 결정이므로 정확도는 50%이다.

NoteBase

보기별 해설

표의 행은 관련 여부, 열은 검색 여부를 나타낸다. 관련 있고 검색된 문서는 70개, 관련 있지만 검색되지 않은 문서는 70개, 관련 없지만 검색된 문서는 30개, 관련 없고 검색되지 않은 문서는 30개이다. 전체 문서는 200개이다.

가. X: 정확률은 **검색된 문서 중 관련 문서의 비율**이다. 검색된 문서는 70 + 30 = 100개이고 그중 관련 문서는 70개이므로 $\text{precision} = \frac{70}{70+30} \times 100\% = 70\%$이다. 따라서 50%가 아니라 70%이다.

나. X: 재현율은 **전체 관련 문서 중 검색된 문서의 비율**이다. 관련 문서는 70 + 70 = 140개이고 그중 검색된 문서는 70개이므로 $\text{recall} = \frac{70}{70+70} \times 100\% = 50\%$이다. 따라서 70%가 아니라 50%이다.

다. O: 정확도는 전체 문서 중 검색 여부를 올바르게 결정한 문서의 비율이다. 관련 문서를 검색한 70개와 관련 없는 문서를 검색하지 않은 30개가 올바른 결정에 해당하므로 $\text{accuracy} = \frac{70+30}{70+70+30+30} \times 100\% = 50\%$이다.

바르게 계산한 보기는 다의 1개이므로 ②에 해당한다.

NoteBase

문 38. 빅데이터에 대한 설명으로 적절하지 않은 것을 모두 고른 것은?

가. 맵리듀스(MapReduce)는 분산 시스템에서 대용량 데이터를 처리하기 위해 개발된 소프트웨어 프레임워크이다.
나. NoSQL은 RDBMS보다 강한 일관성 모델을 제공한다.
다. NoSQL은 쿼리 디자인이 중요하며 빅데이터 처리에 반드시 필요하다.
라. Hadoop 시스템은 Java로 구현된 맵리듀스의 오픈 소스다.

① 가, 나
② 가, 라
③ 나, 다
④ 다, 라

정답 3번

간단한 해설

NoSQL이 RDBMS보다 항상 더 강한 일관성을 제공하는 것은 아니며, 빅데이터 처리에 반드시 필요한 것도 아니다. 적절하지 않은 보기는 **나·다이므로** ③이다.

NoteBase

보기별 해설

가. O 맵리듀스는 대규모 데이터를 나누어 **여러 노드에서 처리하고 결과를 모으는** 분산 처리 방식이다. 이를 구현한 프레임워크는 Map 작업의 병렬 실행, 중간 결과의 정렬·그룹화, Reduce 작업 등을 관리한다.

나. **X** NoSQL이라는 분류만으로 RDBMS보다 더 강한 일관성을 보장하지 않는다. 분산 확장성과 가용성을 위해 최종 일관성을 제공하는 경우가 있으며, 강한 일관성의 지원 여부와 범위는 제품·설정에 따라 다르다. 따라서 **일관성 모델은 시스템과 설정에 따라 달라진다**고 고쳐야 한다. 모든 NoSQL이 강한 일관성을 지원하지 못한다고 단정해서도 안 된다.

다. **X** 어떤 질의를 자주 수행하는지에 맞춘 쿼리·데이터 모델 설계는 중요하다. 그러나 NoSQL이 빅데이터 처리의 필수 조건은 아니다. 예를 들어 분산 파일에 저장한 데이터를 맵리듀스로 처리할 수 있다. **NoSQL은 요구에 따라 선택하는 기술**이라고 고쳐야 한다.

라. O Hadoop은 Java로 구현된 **맵리듀스 처리 기능을 제공하는 오픈 소스** 분산 처리 체계다. 맵리듀스의 대표적인 오픈 소스 구현이라는 설명으로 이해하면 된다. 다만 Hadoop 전체에는 HDFS 등도 포함되며, 이용자가 작성하는 맵리듀스 응용 프로그램까지 반드시 Java여야 하는 것은 아니다.

적절하지 않은 보기는 나·다이고, 이를 모두 고른 것은 ③이다.

NoteBase

문 39. 반정규화(denormalization)를 사용한 성능향상에 대한 설명으로 가장 적절하지 않은 것은?

① 1:N 관계 테이블을 통합한다.

② 집계함수를 미리 계산하여 저장한다.

③ 자주 이용되는 칼럼들을 모아 별도의 테이블을 생성한다.

④ 정보의 중복을 최대한 제거하여 데이터 접근 성능을 높인다.

정답 4번

간단한 해설

반정규화는 조회 성능 등을 위해 **중복 저장이나 미리 계산한 값**을 의도적으로 허용하는 설계이다. 중복을 최대한 제거하는 것은 정규화의 방향이며, 반정규화의 핵심 방법과 반대이다.

선지별 해설

반정규화는 정규화된 구조를 출발점으로 실제 접근 패턴에 맞게 테이블을 통합하거나 조회용 데이터를 별도로 저장하는 등 성능을 조정하는 작업이다. 조인·계산·입출력을 줄일 수 있지만, 저장 공간과 갱신 비용이 증가하고 복제된 값의 일관성을 관리해야 한다.

① O: 1:N 관계의 두 테이블을 자주 함께 조회한다면 통합하여 **반복적인 조인을 줄일 수 있다**. 이때 1 쪽의 속성이 N 쪽 행마다 반복될 수 있으므로, 중복을 허용하는 반정규화에 해당한다. 통합하면 항상 빨라지는 것은 아니며 행 크기 증가와 갱신 부담을 함께 고려해야 한다.

② O: 합계·건수 등의 집계 결과를 미리 계산해 저장하면 조회할 때마다 원본의 많은 행을 읽어 다시 계산하지 않아도 된다. 대신 원본 데이터가 변할 때 **저장된 집계값도 갱신**해야 하며, 그렇지 않으면 실제 데이터와 집계 결과가 어긋난다.

③ O: 자주 함께 이용하는 칼럼을 모아 조회용 테이블을 구성하면 필요한 데이터에 적은 접근으로 도달하게 할 수 있다. 예를 들어 자주 쓰는 일부 칼럼을 별도로 저장하면 불필요한 칼럼을 읽거나 여러 테이블을 반복 조인하는 비용을 줄일 수 있다. 원본과 중복된 칼럼을 유지하는 방식이라면 두 저장 위치의 값을 일관되게 갱신해야 한다.

④ X: **정보의 중복을 최대한 제거**하여 갱신 이상을 줄이고 논리적 구조를 정리하는 것은 정규화의 방향이다. 반정규화는 성능상 필요한 경우 중복 칼럼·조회용 테이블·집계값 등을 허용한다. 다만 중복을 늘리는 것 자체가 목적은 아니며, 실제 성능 이익이 유지·관리 비용보다 큰지 확인하여 적용한다.

NoteBase

문 40. ODBC에 대한 설명으로 적절하지 않은 것은?

① 자바 기반의 클래스와 인터페이스를 제공한다.

② 응용프로그램이 특정업체 인터페이스에 종속되지 않는다.

③ 프로그램 내에서 ODBC 함수를 사용하면 Oracle, DB2 등의 DB 액세스가 가능하다.

④ Microsoft사에서 개발한 표준 인터페이스이다.

정답 1번

간단한 해설

자바 클래스와 인터페이스로 데이터베이스에 접근하도록 하는 것은 **JDBC**이다. ODBC는 DBMS별 드라이버를 통해 여러 데이터베이스에 공통 함수 호출 방식으로 접근하는 API이다.

선지별 해설

ODBC는 Open Database Connectivity의 약자로, 응용 프로그램이 공통 API와 DBMS별 드라이버를 이용하여 데이터베이스에 접근하게 한다.

① X: 자바 기반의 클래스와 인터페이스로 데이터베이스 접근 기능을 제공하는 것은 **JDBC**이다. 자바의 java.sql 패키지에는 Connection·Statement·ResultSet 인터페이스와 DriverManager 클래스 등이 있다. ODBC는 이러한 자바 클래스 체계가 아니라 표준 함수 호출을 중심으로 한 데이터베이스 접근 API이다. 다른 언어에서 연결 도구를 통해 ODBC를 이용할 수 있다는 사실과 ODBC 자체가 자바 API라는 주장은 다르다.

② O: 응용 프로그램은 DBMS 업체마다 다른 전용 호출 대신 **공통 ODBC API**를 사용한다. DBMS별 차이는 해당 드라이버가 처리하므로 응용 프로그램의 특정 업체 인터페이스에 대한 의존성을 줄인다. 다만 SQL 문법의 확장이나 DBMS별 지원 기능 차이까지 모두 없어지는 것은 아니다.

③ O: Oracle이나 DB2에 맞는 **ODBC 드라이버**와 접속 설정이 갖추어지면, 프로그램은 ODBC 함수로 연결하고 SQL을 실행하여 결과를 가져올 수 있다. 드라이버 관리자가 응용 프로그램과 적절한 드라이버 사이의 호출을 연결한다. 해당 데이터베이스에 대한 접속·이용 권한도 필요하다.

④ O: ODBC는 **Microsoft가 개발한 표준 데이터베이스 접근 인터페이스**이다. 여러 DBMS에 공통 호출 방식으로 접근하게 하려는 규격이며 SQL 호출 수준 인터페이스 표준과 관련된다. Microsoft 제품에서만 사용할 수 있거나 Microsoft만 드라이버를 만들 수 있다는 뜻은 아니다.

JDBC의 자바 클래스·인터페이스 특징을 ODBC의 특징으로 설명한 ①이 적절하지 않다.

NoteBase