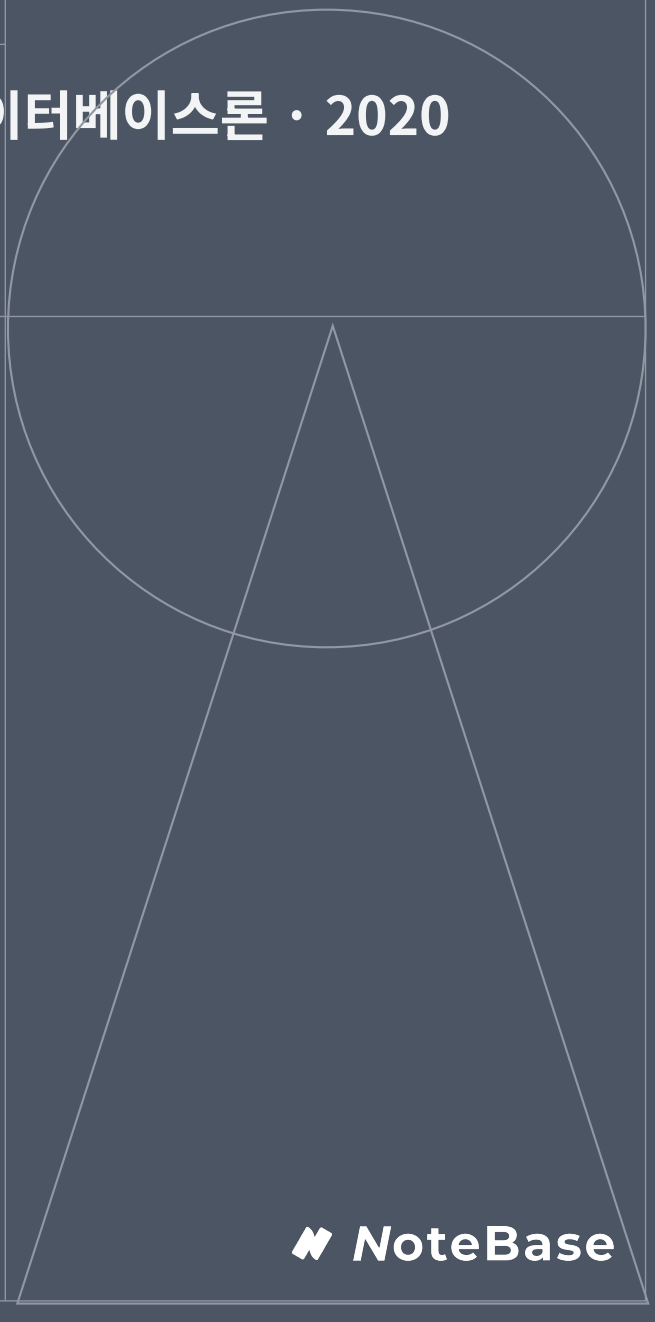
기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2020



NoteBase

기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2020

공무원/경찰/경력채용 · 데이터베이스론 · 2020

문 1. 데이터베이스 스키마에 대한 설명이다. 옳은 것을 모두 고른 것은?

㉠ 외부 스키마는 개별 사용자의 관점에서 보는 데이터베이스 구조에 관한 것으로 여러 개가 존재할 수 있다.
㉡ 내부 스키마는 스키마라고도 하며, 물리적 레코드 즉, 블록이나 페이지 또는 트랙이나 실린더에 관한 명세이다.
㉢ 개념 스키마는 개체, 관계, 제약조건, 보안정책, 무결성 규칙에 관한 명세를 포함한다.
㉣ 서브 스키마는 내부 저장 레코드 형식, 인덱스 유무, 내부 레코드의 물리적 순서 등에 관한 명세를 포함한다.

① ㉠㉡
② ㉠㉢
③ ㉡㉣
④ ㉢㉣

정답 2번

간단한 해설

외부는 사용자별 관점, 개념은 전체 논리 구조, 내부는 물리적 저장 구조이다. **서브 스키마는 외부 스키마이며, 통상 스키마라고 부르는 것은 개념 스키마**라는 용어 구분이 핵심이다.

NoteBase

보기별 해설

㉠ O: 외부 스키마는 특정 사용자나 응용 프로그램에 필요한 데이터와 그 구조를 표현한다. 사용자마다 필요한 관점이 다를 수 있으므로 **하나의 데이터베이스에 여러 외부 스키마**가 존재할 수 있다.

㉡ X: 내부 스키마가 물리적 저장 구조를 명세한다는 부분은 해당 수준의 설명이지만, 3단계 스키마 구분에서 보통 수식어 없이 '스키마'라고 부르는 것은 **개념 스키마**이다. 내부 스키마를 그 별칭으로 설명한 부분이 틀렸다. 저장 레코드·블록 등의 구성은 내부 수준에서 다룬다.

㉢ O: 개념 스키마는 데이터베이스 전체의 논리적 구조를 통합적으로 기술한다. 개체와 관계뿐 아니라 데이터가 지켜야 할 제약조건·무결성 규칙과 보안 관련 명세 등 **전체 데이터의 의미와 관리 규칙**을 포함한다.

㉣ X: 서브 스키마는 사용자 관점의 **외부 스키마**를 가리킨다. 내부 저장 레코드 형식, 인덱스, 물리적 순서 등 저장 구현에 관한 명세는 서브 스키마가 아니라 내부 스키마의 내용이다.

따라서 옳은 설명은 ㉠·㉢이며, ②와 일치한다.

NoteBase

문 2. 데이터베이스 관리 시스템(DBMS)의 주요 구성요소 중 동시성 제어, 회복작업 등의 역할을 수행하는 것으로 가장 적절한 것은?

① 트랜잭션 관리자

② DML 예비 컴파일러(Precompiler)

③ DDL 컴파일러

④ 저장 데이터 관리자

정답 1번

간단한 해설

동시성 제어와 회복은 트랜잭션의 안전한 실행과 장애 처리를 위한 기능이다. 이를 관리하는 구성요소는 트랜잭션 관리자이다.

핵심 풀이

트랜잭션 관리 계층은 여러 트랜잭션이 동시에 실행될 때 서로의 처리에 부적절하게 간섭하지 않도록 **동시성 제어**를 수행하고, 실패한 작업의 취소와 장애 후 복구 등 **회복**을 관리한다. 구현에 따라 동시성 제어 관리자와 회복 관리자가 별도 모듈로 나뉘더라도 두 기능은 트랜잭션 관리에 속한다.

DML 예비 컴파일러는 응용 프로그램에 포함된 데이터 조작문을 추출·변환하여 후속 컴파일을 준비한다.

DDL 컴파일러는 스키마 정의문을 처리한다.

저장 데이터 관리자는 저장된 데이터에 대한 물리적 접근을 담당한다. 문제에서 함께 제시한 동시성 제어와 회복을 포괄하는 구성요소와는 역할의 중심이 다르다.

따라서 가장 적절한 구성요소는 ① 트랜잭션 관리자이다.

NoteBase

문 3. 데이터베이스 설계 과정과 각 과정에서 수행한 작업의 결과가 가장 적절하지 않은 것은?

① 논리적설계 → 릴레이션 스키마

② 물리적설계 → 인덱스 구조

③ 요구사항분석 → 릴레이션 명세서

④ 개념적설계 → ER 다이어그램

정답 3번

간단한 해설

요구사항 분석에서는 필요한 데이터·기능·제약을 **요구사항 명세서**로 정리한다. 릴레이션 스키마를 구체화하는 논리적 설계와 구별해야 한다.

선지별 해설

① O: 논리적 설계는 개념적 모델을 선택한 데이터 모델에 맞는 논리적 구조로 변환하는 단계이다. 관계형 데이터베이스에서는 테이블의 속성·키·관계 등을 정한 **릴레이션 스키마**가 주요 결과물이다.

② O: 물리적 설계에서는 데이터의 실제 저장 방법과 접근 경로를 정한다. 조회 성능과 저장·갱신 비용을 고려해 결정하는 **인덱스 구조**는 이 단계의 결과에 해당한다.

③ X: 요구사항 분석은 사용자가 필요로 하는 데이터·처리 기능·제약 조건을 조사하여 **요구사항 명세서**로 정리하는 단계이다. 릴레이션의 구체적인 구조를 명세하는 것은 이후 논리적 설계에 해당하므로 요구사항 분석의 대표 결과물과 혼동하면 안 된다.

④ O: 개념적 설계는 업무에서 필요한 개체·속성·관계를 구현 세부사항과 분리해 표현하는 단계이다. 이를 시각적으로 모델링한 **ER 다이어그램**이 대표적인 결과물이다.

NoteBase

문 4. B-트리와 B^+-트리에 대한 설명이다. 옳은 것을 모두 고른 것은?

㉠ B-트리에서 새로운 키 값은 항상 단말 노드에 삽입된다.
㉡ B^+-트리에서 내부(또는 중간) 노드의 키 값이 단말 노드에 중복하여 나타날 수 있다.
㉢ B^+-트리에서 키 값의 삭제는 B-트리보다 간단하다.
㉣ B-트리에서 오버플로가 발생하여 단말 노드가 분할될 때, 중간 키 값이 부모 노드뿐만 아니라 새로 분할된 노드에도 저장된다.

① ㉠㉢㉣
② ㉠㉡㉢
③ ㉡㉢㉣
④ ㉠㉡㉣

정답 2번

간단한 해설

B-트리의 분할은 중간 키를 부모로 **이동**시키는 반면, B^+-트리의 단말 분할은 경계 키를 부모에 **복사**한다.
B^+-트리의 실제 데이터 항목은 단말 노드에서 관리한다.

보기별 해설

㉠ O: B-트리의 새 키 삽입은 탐색으로 찾은 단말 노드에서 시작한다. 단말 노드가 넘치면 분할하고 중간 키를 부모로 올리므로, 삽입 과정이 끝난 뒤에도 새 키가 반드시 단말에 남는다는 뜻은 아니다.

㉡ O: B^+-트리의 내부 노드에는 탐색 경로를 정하는 키가 있고, 단말 노드에는 실제 데이터 항목 또는 레코드 포인터가 있다. 단말 분할 때 경계 키의 복사본을 부모에 올릴 수 있어 **같은 키 값이 내부 노드와 단말 노드에 함께 존재**할 수 있다.

㉢ O: B^+-트리는 실제 데이터 항목이 단말에 모여 있어 삭제를 단말 노드에서 처리한다. B-트리에서 내부 키를 삭제할 때 필요한 대체 키 선택 절차와 구별되므로 기본 삭제 처리가 단순하다. 다만 B^+-트리에서도 최소 점유 조건을 어기면 재분배·병합과 내부 경계 키 조정이 필요하다.

㉣ X: B-트리의 노드 분할에서는 중간 키를 부모로 **이동**시키며, 그 키를 분할된 자식 노드에 다시 남기지 않는다. 단말에 키를 유지하면서 부모에 복사하는 것은 B^+-트리의 단말 분할 방식이다.

옳은 설명은 ㉠, ㉡, ㉢이므로 ②이다.

NoteBase

문 5. 다음 릴레이션의 키에 대한 설명 중 옳은 것은 모두 몇 개인가?

학생

학번	주민등록번호	학년
202001	830110-1111111	3
202002	851020-2222222	3
202003	860511-2345678	2

수강

학번	과목명
202001	정보보호
202002	정보보호
202002	사이버보안
202003	시스템보안

㉠ 학생 릴레이션의 슈퍼키는 (학번), (주민등록번호), (학번, 주민등록번호), (학번, 학년), (주민등록번호, 학년)만 가능하다.
㉡ 학생 릴레이션의 후보키는 (학번), (주민등록번호)이다.
㉢ 수강 릴레이션의 기본키는 (학번, 과목명)이다.
㉣ 학생 릴레이션에서 (학번)이 기본키로 설정되면 (주민등록번호)는 대체키가 된다.

① 1개
② 2개
③ 3개
④ 4개

정답 3번

간단한 해설

후보키를 포함하는 더 큰 속성 집합도 슈퍼키이다. **세 속성 전체의 조합도 슈퍼키**인데 이를 제외하고 다섯 집합만 가능하다고 한 설명이 틀렸다.

NoteBase

보기별 해설

㉠ X: 슈퍼키는 튜플을 유일하게 구별하면 되며 최소성은 요구하지 않는다. 제시된 다섯 집합 외에도 **(학번, 주민등록번호, 학년)**이 슈퍼키이므로 '만 가능하다'는 제한이 틀렸다. 세 속성 전체를 포함해도 기존 식별 능력은 사라지지 않는다.

㉡ O: 학생에서 학번과 주민등록번호는 각각 학생을 구별할 수 있고, 단일 속성이므로 더 줄여 유일성을 유지할 수 없다. 따라서 둘은 **유일성과 최소성을 함께 만족하는 후보키**이다. 학년은 같은 값이 반복되어 단독 후보키가 아니다.

㉢ O: 수강에서는 같은 학번이 여러 과목을 수강하고, 같은 과목명도 여러 학생에게 나타난다. 어느 한 속성만으로는 구별할 수 없지만 **(학번, 과목명)의 쌍**으로 각 수강 사실을 식별하므로 복합 기본키로 사용한다.

㉣ O: 여러 후보키 중 대표로 선택한 것이 기본키이며, 선택되지 않은 후보키는 대체키이다. 따라서 학번을 기본키로 선택하면 **남은 후보키인 주민등록번호가 대체키**가 된다.

옳은 설명은 ㉡·㉢·㉣의 3개이므로 ③과 일치한다.

NoteBase

문 6. 관계형 데이터베이스에서 릴레이션 인스턴스의 설명으로 옳은 것을 모두 고른 것은?

㉠ 어느 한 시점에 릴레이션이 포함하고 있는 튜플의 집합이다.
㉡ 릴레이션에서 동적인 성질을 의미한다.
㉢ 릴레이션의 내포(Intension)를 표현한 구조이다.
㉣ 시간에 따라 삽입연산, 삭제연산, 변경연산을 통해 릴레이션의 값을 변화시킨다.
㉤ 릴레이션 이름, 애트리뷰트 이름, 제약조건을 의미한다.

① ㉠㉡㉣
② ㉠㉢㉣
③ ㉡㉢㉤
④ ㉠㉡㉢

정답 1번

간단한 해설

릴레이션 인스턴스는 **특정 시점의 실제 튜플 집합**으로, 삽입·삭제·변경에 따라 달라진다. 이름·속성·제약 조건을 정하는 스키마, 즉 내포와 구별해야 한다.

보기별 해설

㉠ O: 릴레이션 인스턴스는 **특정 시점에 실제로 저장된 튜플의 집합**이다. 같은 스키마를 따르더라도 시점에 따라 인스턴스는 달라질 수 있다.

㉡ O: 인스턴스는 현재 저장된 데이터 값이므로 **동적인 성질**을 갖는다. 상대적으로 고정된 구조를 뜻하는 스키마와 대비된다.

㉢ X: 릴레이션의 **내포(Intension)는 스키마**가 나타내는 구조적 정의이다. 인스턴스는 그 정의를 따르는 실제 데이터인 외연(Extension)에 해당한다.

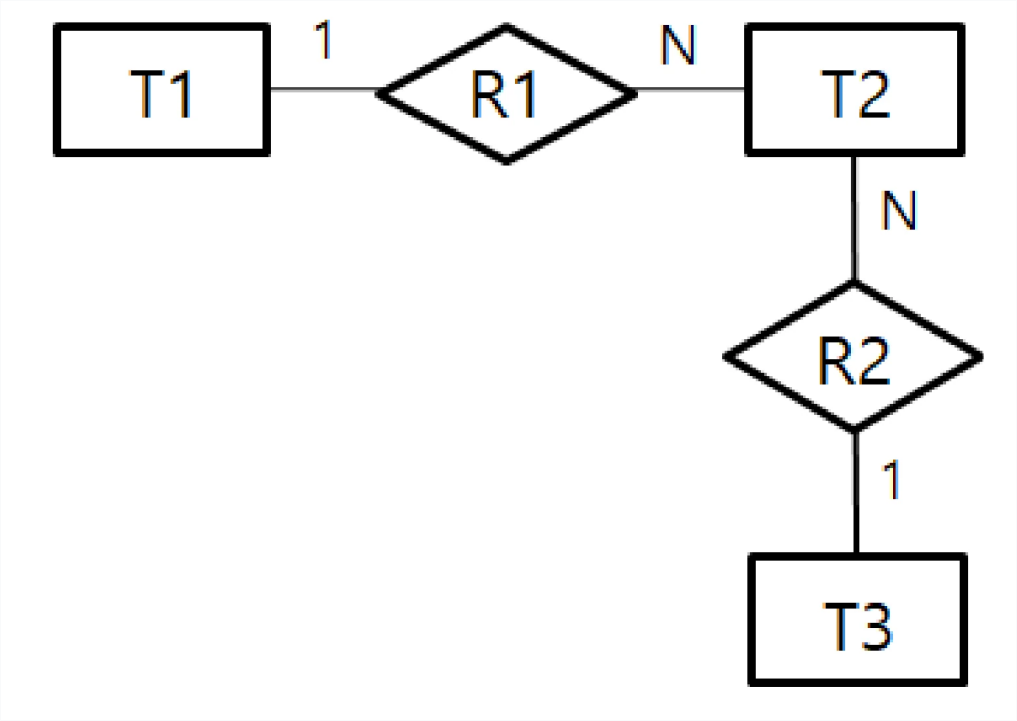
㉣ O: 튜플의 삽입·삭제·값 변경이 이루어지면 실제 저장 내용이 달라져 **새로운 시점의 인스턴스**가 된다. 여기서 값의 변화는 스키마 자체를 바꾸는 것과 구별한다.

㉤ X: 릴레이션 이름, 애트리뷰트 이름, 제약 조건은 **릴레이션의 구조와 규칙**을 정하는 스키마에 관한 설명이다. 특정 시점의 실제 튜플 집합을 뜻하지 않는다.

따라서 릴레이션 인스턴스에 대한 옳은 설명은 ㉠·㉡·㉣이며 ①과 일치한다.

NoteBase

문 7. 다음 ERD(Entity-Relationship Diagram)를 구성하는 3개의 릴레이션 T1, T2, T3(3개의 개체와 2개의 관계는 3개의 릴레이션으로 변환)에 대한 참조무결성 제약조건을 정의하였다고 가정하자. 정의한 참조무결성 제약조건의 영향을 받지 않는 연산들을 모두 고른 것은?



㉠ T1 테이블에 새로운 튜플을 삽입하는 삽입연산
㉡ T1 테이블에서 튜플을 검색하는 검색연산
㉢ T2 테이블에서 외래키값을 수정하는 변경연산
㉣ T2 테이블에서 기존 튜플을 삭제하는 삭제연산
㉤ T3 테이블에 새로운 튜플을 삽입하는 삽입연산
㉥ T3 테이블에서 기존 튜플을 삭제하는 삭제연산

① ㉠㉡㉢㉥
② ㉠㉡㉣㉤
③ ㉠㉢㉤㉥
④ ㉡㉢㉣㉤

정답 2번

간단한 해설

두 관계의 N측인 **T2가 T1과 T3를 외래키로 참조**한다. 부모 테이블의 삽입, 자식 테이블의 삭제, 단순 검색은 이 참조무결성을 깨뜨리지 않지만, 자식의 외래키 변경이나 부모의 삭제는 확인이 필요하다.

NoteBase

보기별 해설

그림에서 T1과 T3는 각각 1측이고 T2는 두 관계 모두에서 N측이다. 세 릴레이션으로 변환하면 **T2에 T1과 T3를 참조하는 외래키**를 둔다. 아래 O는 제시된 참조무결성 제약의 영향을 받지 않는 연산, X는 참조무결성 확인이나 참조 동작이 필요한 연산을 뜻한다. 기본키·도메인 등 다른 제약까지 없다는 뜻은 아니다.

㉠ O: T1은 참조되는 부모 테이블이다. T1에 새 튜플을 추가해도 기존 T2의 참조 대상이 사라지거나 잘못된 외래키 값이 생기지 않으므로, 이 참조무결성 제약의 영향을 받지 않는다.

㉡ O: 검색은 저장된 데이터를 변경하지 않는다. 따라서 T1에서 튜플을 읽는 것만으로 참조 관계를 위반하지 않는다.

㉢ X: T2의 외래키 값을 변경하면 새 값에 대응하는 T1 또는 T3의 참조 키가 존재하는지 확인해야 한다. **존재하지 않는 부모를 가리키는 값**으로 바꾸면 참조무결성에 위배된다. NULL 허용 여부도 정의된 제약을 따라야 한다.

㉣ O: T2는 다른 테이블을 참조하는 자식 테이블이다. T2의 튜플을 삭제하는 것은 참조 자체를 없애는 일이므로, 제시된 두 관계에서 참조 대상 없는 외래키를 남기지 않는다.

㉤ O: T3 역시 참조되는 부모 테이블이다. 새 튜플을 추가하는 것은 가능한 참조 대상을 늘리는 일이므로 기존 외래키 참조를 깨뜨리지 않는다.

㉥ X: 삭제하려는 T3의 튜플을 T2가 참조하고 있다면 참조 대상을 없애게 된다. 따라서 **삭제 제한이나 연쇄 삭제 등 정의된 참조 동작**의 영향을 받는다. 실제로 참조하는 T2 행이 없으면 삭제할 수 있다는 것과, 이 연산이 제약의 영향을 받지 않는다는 것은 다르다.

따라서 영향을 받지 않는 연산은 ㉠·㉡·㉣·㉤이며, 해당 조합은 ②이다.

NoteBase

문 8. 두 릴레이션 R과 S에 대한 관계대수 연산의 결과 릴레이션을 T1, T2라고 했을 때, 다음 중 연산의 결과로 옳은 것은? (단, c(T)는 릴레이션 T의 카디널리티, σ은 실렉트연산, Π는 프로젝트연산, ÷는 디비젼연산, ⋈_N는 자연조인연산을 의미한다.)

$T1 : R(A,B) \div (\Pi_B(\sigma_{D>2}(S)))$

$T2 : R(A,B) \bowtie_N (\Pi_{B,C}(\sigma_{D=4}(S)))$

R

A	B
a1	b1
a2	b2
a3	b2
a2	b3
a4	b1
a5	b2
a4	b2
a5	b3

S

B	C	D
b1	c1	1
b2	c3	3
b4	c1	2
b3	c2	4

① $c(T1) + c(T2) = 2$

② $c(T1) + c(T2) = 3$

③ $c(T1) + c(T2) = 4$

④ $c(T1) + c(T2) = 5$

정답 3번

간단한 해설

디비전 결과의 A값은 **a2, a5**로 2개이고, 자연조인 결과도 **2개 튜플**이다. 두 카디널리티의 합은 4이다.

NoteBase

계산 과정

T1의 나누는 릴레이션부터 구한다. S에서 D > 2를 만족하는 행은 (b2, c3, 3), (b3, c2, 4)이고, B로 프로젝트하면 **{b2, b3}**이다.

디비전은 나누는 릴레이션의 모든 B값과 대응하는 A값을 찾는다. R에서 b2와 b3을 **모두 가진 A값은 a2와 a5**이므로 T1 = {(a2), (a5)}, $c(T1) = 2$이다. b2만 있는 a3이나 b1 · b2만 있는 a4는 포함되지 않는다.

T2에서는 S에서 D = 4인 행 (b3, c2, 4)을 고른 뒤 B, C로 프로젝트하여 {(b3, c2)}를 얻는다. R과 자연조인하면 공통 속성 B가 b3인 (a2, b3), (a5, b3)이 연결되어 T2 = {(a2, b3, c2), (a5, b3, c2)}이다. 따라서 $c(T2) = 2$이다.

그러므로 $c(T1) + c(T2) = 2 + 2 = 4$이며, 해당하는 선지는 ③이다.

NoteBase

문 9. 관계대수의 연산자에 대한 설명으로 가장 적절하지 않은 것은?

① 합집합, 차집합, 카티션 프로덕트, 프로젝트, 실렉트는 기본 연산자이다.

② 프로젝트 → 실렉트 → 카티션 프로덕트 → 조인 순으로 연산자 우선순위가 낮아진다.

③ 세미조인은 교환법칙이 성립하지 않는다.

④ 조인, 교집합, 디비젼과 같은 복합 연산자는 기본 연산자로 변환하여 표현할 수 없다.

정답 4번

출제 포인트

관계대수의 기본 연산과 복합 연산을 구별하고, 연산자 우선순위와 세미조인의 교환법칙 성립 여부를 판단하는 문제이다.

핵심 개념

관계대수는 릴레이션을 입력받아 새로운 릴레이션을 만드는 연산 체계이다. 기본 연산은 다른 연산을 구성하는 바탕이고, 복합 연산은 기본 연산들을 조합하여 표현하는 연산이다. 실렉트는 조건에 맞는 행을 선택하고, 프로젝트는 필요한 열을 추출한다.

선지별 해설

① O. 합집합, 차집합, 카티션 프로덕트, 프로젝트, 실렉트는 관계대수의 기본 연산이다. 이 다섯 연산을 조합하여 조인, 교집합, 디비젼을 표현할 수 있다.
② O. 이 문항에서 적용하는 우선순위는 프로젝트, 실렉트, 카티션 프로덕트, 조인의 순으로 낮아진다. 다만 식에 괄호가 있으면 괄호로 지정한 연산 범위가 우선하므로, 이 순서를 모든 질의의 실제 실행 순서와 동일시해서는 안 된다.
③ O. 세미조인은 상대 릴레이션에 대응하는 튜플이 있는 왼쪽 릴레이션의 튜플만 반환한다. 피연산자를 바꾸면 반환하는 속성과 튜플의 기준도 바뀌므로 일반적으로 교환법칙이 성립하지 않는다.
④ X. 틀린 표현은 '기본 연산자로 변환하여 표현할 수 없다'이다. 올바르게는 '기본 연산자로 변환하여 표현할 수 있다'이다. 세타 조인은 카티션 프로덕트의 결과에 조인 조건으로 실렉트를 적용한 것으로, $R \bowtie_c S = \sigma_c(R \times S)$이다. 합병 가능한 두 릴레이션의 교집합도 $R \cap S = R - (R - S)$로 표현된다. 디비젼은 후보를 프로젝트로 구한 뒤, 카티션 프로덕트와 차집합으로 조건을 충족하지 못한 후보를 찾아 제외하여 표현할 수 있다.

오답 포인트

④의 '복합'은 기본 연산으로 표현할 수 없다는 뜻이 아니라 기본 연산의 조합이라는 뜻이다. 또한 일반적인 자연조인의 교환법칙을 왼쪽 릴레이션만 반환하는 세미조인에 그대로 적용하면 ③을 잘못 판단하게 된다.

NoteBase

다시 나오면 이렇게 판단

먼저 기본 연산 다섯 가지를 확인하고, 나머지 연산이 이들의 조합으로 표현되는지 판단한다. 교환법칙은 피연산자의 순서를 바꾸어도 반환하는 속성과 튜플이 같은지로 판단한다.

암기 포인트

기본 연산은 합집합, 차집합, 카티션 프로덕트, 프로젝트, 실렉트이다. 복합 연산은 기본 연산의 조합이고, 세미조인은 왼쪽 릴레이션만 반환한다.

NoteBase

문 10. 다음 EMP 릴레이션에서 모든 사원에 대한 올바른 상하 관계를 갖는 결과 릴레이션(T)을 얻기 위한 SQL문으로 가장 옳은 것은? (단, 특정 사원의 관리자 애트리뷰트 값이 NULL인 것은 상위관리자가 없음을 의미한다.)

EMP

사원	관리자
김	NULL
이	김
박	김
최	박
하	박

결과 릴레이션(T)

사원	관리자	차상위관리자
김	NULL	NULL
이	김	NULL
박	김	NULL
최	박	김
하	박	김

① SELECT E1.사원, E1.관리자, E2.관리자 차상위관리자
FROM EMP E1, EMP E2;

② SELECT E1.사원, E1.관리자, E2.관리자 차상위관리자
FROM EMP E1, EMP E2
WHERE E1.관리자 = E2.사원;

③ SELECT E1.사원, E1.관리자, E2.관리자 차상위관리자
FROM EMP E1 JOIN EMP E2 ON (E1.관리자 = E2.사원);

④ SELECT E1.사원, E1.관리자, E2.관리자 차상위관리자
FROM EMP E1 LEFT OUTER JOIN EMP E2 ON (E1.관리자 = E2.사원);

정답 4번

간단한 해설

직접 관리자의 행을 찾는 **자기 조인**에 더해, 관리자가 NULL인 김까지 남겨야 한다. 따라서 사원 쪽 E1의 모든 행을 보존하는 LEFT OUTER JOIN을 사용한다.

NoteBase

핵심 풀이

자기 조인의 연결 기준: 같은 EMP를 사원 쪽 E1과 관리자 쪽 E2로 나누어 참조한다. **E1.관리자=E2.사원**으로 직접 관리자의 행을 찾으면 그 행의 E2.관리자가 차상위관리자이다. 최·하의 관리자는 박이고 박의 관리자는 김이므로 차상위관리자는 김이다.

모든 사원 보존: 김은 관리자 값이 NULL이므로 위 등호 조건을 만족하는 E2 행이 없다. 그러나 결과 T에는 김도 있어야 하므로 **E1을 보존하는 LEFT OUTER JOIN**이 필요하다. 매칭되지 않는 김의 오른쪽 값은 NULL로 채워진다. 이·박은 관리자 김의 행과 매칭되지만, 김의 관리자 값 자체가 NULL이므로 차상위관리자가 NULL이다.

다른 방식과의 차이: 조인 조건이 없는 곱집합은 5×5=25개 조합을 만들어 올바른 상하 관계를 보장하지 않는다. WHERE의 등호 조건을 쓰는 조인과 JOIN ON을 쓰는 내부 조인은 모두 매칭되지 않는 김을 제외하여 4행만 반환한다.

따라서 ④의 왼쪽 외부 자기 조인을 사용하면 (김,NULL,NULL), (이,김,NULL), (박,김,NULL), (최,박,김), (하,박,김)의 5행을 얻는다. ORDER BY가 없으므로 이 나열은 결과의 내용을 나타내며 출력 순서를 보장한다는 뜻은 아니다.

NoteBase

문 11. NoSQL에 대한 설명으로 가장 적절하지 않은 것은?

① 일관성(Consistency)과 가용성(Availability)을 강조한 모델이다.

② 기본 데이터 모델은 Key-Value이며, 데이터가 정규화되어 있지 않다.

③ 관계형 데이터 모델을 사용하지 않는다.

④ 빅데이터와 실시간 웹 응용에서 사용된다.

정답 1번

간단한 해설

NoSQL은 하나의 일관성·가용성 조합으로 정의되지 않는다. 분산 시스템에서는 **네트워크 분할 시 일관성과 가용성 사이의 선택**이 필요하며 제품마다 정책이 다르다.

선지별 해설

① X: NoSQL을 일관성과 가용성을 함께 강조하는 하나의 CA 모델로 규정할 수 없다. 분산 환경에서 **네트워크 분할이 발생하면 강한 일관성과 가용성을 동시에 보장할 수 없다**는 것이 CAP의 핵심이며, 제품별로 선택이 다르다. NoSQL 전체가 언제나 같은 두 속성만 선택한다고 외워서는 안 된다.

② O: 키-값은 대표적인 기본 NoSQL 모델이며, 조회 패턴에 맞춰 데이터를 한곳에 모으는 비정규화 설계가 널리 사용된다. 다만 **모든 NoSQL이 키-값 모델만 사용한다는 뜻은 아니다**. 문서·와이드컬럼·그래프 등도 있으며, 정규화 여부도 모델과 설계에 따라 달라진다.

③ O: NoSQL은 전통적인 관계형 테이블 모델 이외의 데이터 모델을 사용하는 비관계형 데이터베이스를 가리킨다. **비관계형 모델이라는 분류**와 SQL과 비슷한 질의 문법을 일부 지원하는지는 별개의 문제이다.

④ O: 유연한 데이터 구조와 분산 확장성을 활용하여 대량 데이터를 다루거나 낮은 지연이 필요한 웹 응용에 쓰인다. **빅데이터·실시간 웹 서비스는 대표적인 활용 분야**이며, 실제 제품 선택은 접근 패턴과 일관성 요구 등을 함께 고려한다.

NoteBase

문 12. 다음은 커서(Cursor)의 처리 단계를 설명한 것이다. (가)~(라)에 해당되는 커서 처리단계 명령문을 <보기>에서 골라 가장 적절하게 짝지어진 것은?

(가) 커서와 관련된 SQL 문장을 정의
(나) SQL문이 실행가능하도록 커서를 개방
(다) 커서가 가리키는 결과테이블의 한 튜플을 호스트변수로 가져옴
(라) SQL문의 작업을 종료한 후 커서를 폐쇄

<보기>
㉠ OPEN ㉡ DECLARE ㉢ FETCH ㉣ CLOSE

① (가): ㉡, (나): ㉠, (다): ㉢, (라): ㉣
② (가): ㉡, (나): ㉢, (다): ㉠, (라): ㉣
③ (가): ㉠, (나): ㉡, (다): ㉢, (라): ㉣
④ (가): ㉠, (나): ㉢, (다): ㉡, (라): ㉣

정답 1번

간단한 해설

커서의 일반적인 처리 흐름은 **DECLARE → OPEN → FETCH → CLOSE**이다. 각각 선언, 개방, 결과 행 인출, 폐쇄에 대응한다.

대응 관계 해설

(가) SQL 문장 정의 → **DECLARE(㉡)**: 커서 이름과 연결할 질의를 선언한다. 이 단계는 어떤 결과를 커서로 처리할 것인지 지정하는 단계이다.

(나) 커서 개방 → **OPEN(㉠)**: 선언한 커서를 열어 질의 결과를 가져올 수 있도록 준비한다. 삽입 SQL의 일반적인 커서 처리에서는 이때 질의 실행을 시작한다.

(다) 튜플을 호스트 변수로 가져오기 → **FETCH(㉢)**: 열린 커서에서 결과 행을 가져와 응용 프로그램의 호스트 변수에 저장한다. 필요한 행을 처리할 때까지 반복할 수 있다.

(라) 커서 폐쇄 → **CLOSE(㉣)**: 결과 처리를 마친 커서를 닫고 관련 자원을 해제한다. 커서를 닫는 것을 트랜잭션의 COMMIT과 동일시해서는 안 된다.

따라서 (가)~(라)의 대응은 ㉡, ㉠, ㉢, ㉣이며, 이를 올바르게 짝지은 것은 ①이다.

NoteBase

문 13. 관계 데이터베이스에서 정규화를 수행하여 얻을 수 있는 목적으로 가장 적절하지 않은 것은?

① 데이터베이스 연산에서 발생하는 여러 가지 이상(Anomaly) 현상을 제거할 수 있다.

② 개체들 간의 관계를 파악하기 용이하다.

③ 정규화를 통해 릴레이션을 분해하면 릴레이션의 수가 증가하므로, 조인연산으로 인해 검색성능이 보다 좋아진다.

④ 데이터의 중복성을 줄일 수 있다.

정답 3번

간단한 해설

정규화는 데이터 중복과 이상 현상을 줄이는 데 목적이 있다. 릴레이션을 분해하면 조인이 추가로 필요해질 수 있으므로, **조인 증가가 검색 성능 향상을 보장하지는 않는다**.

선지별 해설

① O: 여러 사실이 한 릴레이션에 뒤섞이고 중복되어 있으면 삽입·삭제·갱신 이상이 생길 수 있다. 종속 관계에 맞게 적절히 분해하면 **중복과 부적절한 종속에서 비롯되는 이상**을 제거하거나 줄일 수 있다. 이는 모든 종류의 프로그램 오류를 없앤다는 뜻은 아니다.

② O: 서로 다른 개체의 사실을 적절한 릴레이션으로 나누고 키를 통해 연결하면 각 개체가 가진 속성과 개체 사이의 관계가 명확해진다. 예를 들어 학생의 기본 정보와 수강 정보를 구분하면 학생과 수강의 연결을 키로 표현할 수 있다.

③ X: 분해된 릴레이션의 정보를 함께 검색하려면 조인이 추가로 필요할 수 있고, 그 처리 비용 때문에 **검색 성능이 오히려 낮아질 수도 있다**. 정규화는 이상과 불필요한 중복을 줄이는 설계 기법이지, 릴레이션과 조인이 늘수록 검색이 빨라지는 기법은 아니다. 실제 성능은 질의·인덱스·데이터 분포 등에 따라 달라진다.

④ O: 같은 사실을 여러 튜플에 반복 저장하는 대신 별도 릴레이션에 두고 키로 참조하면 **불필요한 데이터 중복**을 줄일 수 있다. 다만 관계를 연결하기 위한 키 값의 반복까지 모두 없애는 것은 아니다.

NoteBase

문 14. 트리거(Trigger)에 대한 설명으로 가장 적절하지 않은 것은?

① 조건이 만족될 때 자동으로 지정된 작업을 수행하게 만드는 일종의 프로시저이다.

② 테이블 정의 시 표현할 수 없는 기업의 비즈니스 규칙이나 복잡한 보안 요건들을 시행하는 역할을 수행할 수 있다.

③ 트리거는 무결성을 유지하기 위한 방법으로 사용된다.

④ 문장 트리거(Statement Trigger)는 조건을 만족하는 여러 개의 행에 대해 트리거를 반복적으로 여러 번 수행한다.

정답 4번

간단한 해설

문장 트리거와 행 트리거는 실행 단위가 다르다. **문장 트리거는 SQL 문장당 한 번**, 행 트리거는 해당 행마다 실행된다.

선지별 해설

① O: 트리거는 지정된 사건이 발생했을 때 DBMS가 자동으로 실행하는 저장된 처리 절차이다.
INSERT · UPDATE · DELETE 등의 사건에 연결하고, 추가 조건을 두었다면 그 조건을 만족할 때 작업을 수행하게 할 수 있다.

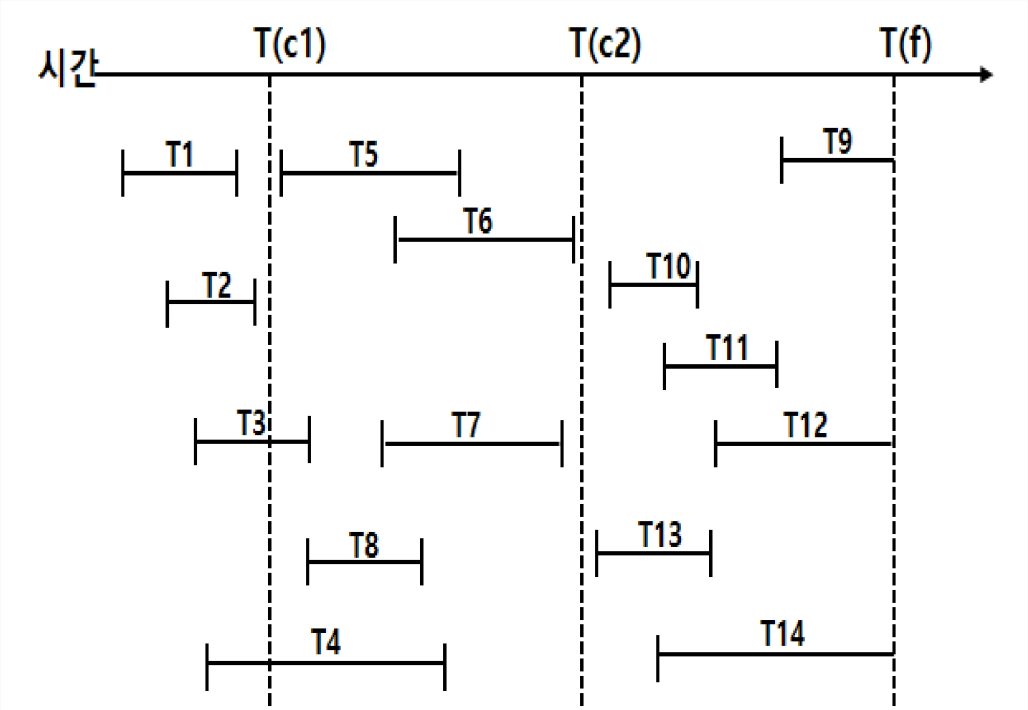
② O: 단순한 열 정의나 선언적 제약만으로 표현하기 어려운 업무 규칙을 트리거의 절차적 검사로 구현할 수 있다.
데이터 변경 시 추가 검증이나 감사 기록을 남기는 등 복잡한 보안 요구를 보완하는 데도 활용한다.

③ O: 트리거는 변경 전후의 데이터를 검사하거나 연관 데이터를 갱신하여 무결성을 유지하는 데 사용할 수 있다.
예를 들어 정해진 업무 규칙을 어기는 변경을 거부하도록 구현할 수 있다.

④ X: 문장 트리거는 **해당 SQL 문장 단위로 한 번** 실행되며, 변경 대상 행 수만큼 반복 실행되는 것이 아니다.
행마다 실행되는 것은 행 트리거(Row Trigger)이다. 예를 들어 한 UPDATE 문이 10개 행을 변경할 때, 그 문장에 지정된 문장 트리거는 1회, 행 트리거는 각 행에 대해 실행된다.

NoteBase

문 15. 다음의 검사점(Checkpoint)이 있는 로그(Log)에서 시스템 장애가 발생하는 경우 재수행(Redo) 연산을 해야 하는 트랜잭션을 모두 고른 것은? (단, T(c1), T(c2)는 검사점, T(f)는 장애시점을 의미한다.)



① T1, T2, T3, T4

② T5, T6, T7, T8, T10, T11, T13

③ T10, T11, T13

④ T9, T12, T14

정답 3번

간단한 해설

마지막 검사점 **T(c2) 이후에 완료되고 장애 T(f) 전에는 커밋된 트랜잭션**을 찾는다. 도식에서 이에 해당하는 것은 T10·T11·T13이다.

NoteBase

계산 과정

기준 검사점: 장애 T(f)에 가장 가까운 **마지막 검사점 T(c2)**를 기준으로 삼는다. 이 도식은 검사점 이전에 완료된 결과가 안정 저장장치에 반영된 기본 검사점 복구 모델을 전제로 해석한다. 도중에 완료를 표시하는 오른쪽 끝막대와 장애선까지 이어지는 실행 구간을 구별한다.

검사점 이전 완료: T1·T2는 T(c1) 전에 끝났고, T3·T4·T5·T6·T7·T8도 T(c2) 전에 끝났다. 이 8개 트랜잭션의 결과는 마지막 검사점에 반영된 것으로 보므로 이번 장애의 Redo 대상에서 제외한다.

검사점 이후 완료: **T10·T11·T13**은 T(c2) 이후에 시작해 T(f) 전에 완료되었다. 커밋된 결과의 지속성을 보장하기 위해 로그의 갱신 내용을 다시 적용하는 Redo 대상으로 분류한다.

장애 시 미완료: T9·T12·T14는 완료 막대 없이 T(f)까지 이어진다. 이 3개는 커밋 완료 집합에 넣지 않는다. 기본 즉시 갱신 복구에서 이미 반영된 미완료 변경을 되돌리는 Undo와 완료 결과를 복원하는 Redo를 구별해야 한다.

대상 집계: 전체 14개는 검사점 이전 완료 8개, 이후 완료 3개, 장애 시 미완료 3개로 나뉜다. 따라서 Redo 집합은 **{T10, T11, T13}**이며 ③과 일치한다.

NoteBase

문 16. 다음은 병행수행을 위한 로킹(Locking) 기법에서 로크적용을 위한 행렬을 테이블로 표현한 것이다. 테이블에서 ㉠, ㉡에 들어갈 내용으로 가장 적절한 것은? (단, S는 공용로크, X는 전용로크, N은 로크없음을 의미한다.)

구분	N	S	X
N	허용	허용	허용
S	허용	㉠	대기
X	허용	㉡	대기

① ㉠: 허용, ㉡: 허용

② ㉠: 허용, ㉡: 대기

③ ㉠: 대기, ㉡: 허용

④ ㉠: 대기, ㉡: 대기

정답 2번

간단한 해설

동일 데이터에 대한 **공용로크끼리는 양립하지만 전용로크와 공용로크는 충돌**한다. S–S는 허용하고 X–S는 대기시킨다.

빈칸별 해설

㉠: S와 S가 만나는 칸이므로 **허용**이다. 공용로크는 같은 데이터를 읽는 트랜잭션끼리 공유할 수 있다. 서로 값을 변경하지 않는 읽기 작업끼리는 양립하므로, 기존 공용로크가 있다는 이유만으로 다른 공용로크를 막지 않는다.

㉡: X와 S가 만나는 칸이므로 **대기**이다. 전용로크는 해당 데이터의 변경을 위해 배타적 접근을 요구하므로 다른 트랜잭션의 공용로크와 양립하지 않는다. 같은 데이터에서 S–X뿐 아니라 X–S도 충돌하며, 기존 충돌 로크가 해제될 때까지 새 요청은 기다린다.

N은 충돌하는 로크가 없는 상태이고, X–X도 서로 양립하지 않는다. 따라서 ㉠은 허용, ㉡은 대기로 ②와 일치한다.

NoteBase

문 17. 관계 데이터베이스에서 테이블 조작을 위한 권한부여 명령을 다음과 같이 수행하였다. 명령을 수행한 후의 테이블에 대한 권한을 설명한 것 중 가장 적절하지 않은 것은? (단, [ ]은 사용자를 의미한다.)

[DBA] GRANT SELECT ON T1 TO USER1
WITH GRANT OPTION;
[USER1] GRANT SELECT ON T1 TO USER2
WITH GRANT OPTION;
[USER2] GRANT SELECT ON T1 TO USER3
WITH GRANT OPTION;
[USER3] GRANT SELECT ON T1 TO USER4;
[USER2] REVOKE SELECT ON T1 FROM USER3
CASCADE;

① USER1은 T1 테이블에 대해 SELECT 연산을 수행할 수 있다.

② USER2는 T1 테이블에 대해 SELECT 연산을 수행할 수 있다.

③ USER3은 T1 테이블에 대해 SELECT 연산을 수행할 수 없다.

④ USER4는 T1 테이블에 대해 SELECT 연산을 수행할 수 있다.

정답 4번

간단한 해설

권한은 DBA에서 USER1, USER2, USER3을 거쳐 USER4로 전달되었다. USER3의 SELECT를 CASCADE로 회수하면 **USER3에게 의존하는 USER4의 권한도 함께 회수**되며, 상위 부여자인 USER1과 USER2의 권한은 남는다.

NoteBase

선지별 해설

① O: USER1은 DBA에게 SELECT 권한을 직접 받았다. 마지막 REVOKE는 USER2가 USER3에게 준 권한을 취소하는 것이므로, 그보다 상위 경로에 있는 USER1의 권한은 유지된다.

② O: USER2는 USER1에게 SELECT 권한과 재부여 권한을 받았다. USER3에게 부여했던 권한을 회수하더라도 USER2 자신의 권한을 반납하는 것은 아니므로 SELECT를 계속 수행할 수 있다.

③ O: 마지막 명령의 직접 회수 대상은 USER3이다. REVOKE SELECT이므로 SELECT 권한 자체가 회수되며, 이에 딸린 재부여 권한도 사라진다. 제시된 부여 경로에서는 USER3이 더 이상 SELECT를 수행할 수 없다.

④ X: USER4의 SELECT 권한은 USER3의 권한에 의존한다. **CASCADE는 회수 대상에게 의존하여 재부여된 권한도 연쇄 회수**하므로, USER3에게서 받은 USER4의 권한도 사라진다. 제시된 명령에는 별도의 독립 부여 경로가 없으므로 USER4는 SELECT를 수행할 수 없다.

NoteBase

문 18. 데이터베이스와 웹페이지 간의 데이터 교환을 위해 사용되는 XML(eXtensible Markup Language)에 대한 설명으로 가장 적절하지 않은 것은?

① XSLT는 XML 문서를 HTML, XML, PDF 등 다른 문서 형태로 변환하고자 할 때 사용된다.

② XML Schema는 키와 참조 무결성 제약을 지원하며, 사용자가 원하는 임의의 복합 타입을 정의할 수 있다.

③ SAX는 XML 문서의 파싱에 사용되며 XML 문서 전체를 트리 형태로 표현하기 위한 API 표준이다.

④ XQuery 질의문은 FOR, LET, WHERE, RETURN 절들로 구성되고 조인연산을 작성할 수 있다.

정답 3번

간단한 해설

SAX는 XML을 읽으면서 발생하는 이벤트를 처리하는 **이벤트 기반 API**이다. 문서 전체를 트리로 표현하는 API는 **DOM**이므로 두 방식의 설명을 구별해야 한다.

선지별 해설

① O: XSLT는 XML을 다른 XML 구조나 HTML 등으로 변환하는 데 사용한다. PDF를 만드는 경우에는 보통 **XSLT로 XSL-FO를 생성하고 별도의 포매터로 PDF를 출력**한다. 따라서 PDF 변환 과정에 XSLT가 사용된다는 취지이며, XSLT 자체가 PDF 렌더러라는 뜻은 아니다.

② O: XML Schema는 key와 keyref로 키의 유일성과 참조 관계를 검증할 수 있다. 또한 complexType으로 요소와 속성의 구조를 조합한 **사용자 정의 복합 타입**을 만들 수 있다. '임의'는 스키마 문법과 제약 안에서 필요한 구조를 정의할 수 있다는 뜻이다.

③ X: SAX는 요소 시작·종료와 문자 데이터 등을 읽을 때 응용 프로그램의 처리 함수를 호출하는 **이벤트 기반 API**이다. XML 문서 전체를 노드 트리로 표현하고 탐색·수정하는 표준 API는 **DOM**이다. SAX의 이벤트로 응용 프로그램이 별도 트리를 만들 수는 있지만, SAX 자체가 전체 문서 트리 표현 API인 것은 아니다.

④ O: XQuery의 FLWOR 표현식은 for로 항목을 순회하고 let으로 값을 바인딩하며 where로 조건을 걸고 return으로 결과를 구성한다. 여러 입력의 항목을 조건으로 연결하여 조인을 표현할 수 있다. 다만 모든 XQuery가 이 네 절을 반드시 포함하는 것은 아니며, order by 등 다른 절도 사용할 수 있다.

NoteBase

문 19. 빅데이터의 특징과 처리 기술에 대한 설명으로 가장 적절하지 않은 것은?

① 정형 데이터를 제외한 반정형, 비정형 데이터를 실시간으로 처리한다.

② 분산 시스템에서 대용량 데이터 처리를 지원하는 오픈소스로 하둡(Hadoop)이 있다.

③ 데이터 마이닝은 데이터에 숨겨진 규칙과 패턴을 찾아 유용한 정보를 추출하는 것이다.

④ 텍스트 마이닝은 비구조화된 데이터로부터 유용한 정보를 추출할 수 있는 기술이다.

정답 1번

간단한 해설

빅데이터에는 **정형 데이터도 포함**되며, 처리 방식은 배치와 실시간 처리 등으로 다양하다. 반정형·비정형 데이터만 실시간으로 처리한다고 한정할 수 없다.

선지별 해설

① X: 빅데이터의 다양성은 **정형·반정형·비정형 데이터 모두**를 포함한다. 정형 데이터를 제외하지 않으며, 처리 방식도 목적과 지연 허용 범위에 따라 배치 처리나 실시간·스트림 처리를 선택하므로 언제나 실시간 처리만 하는 것은 아니다.

② O: 하둡은 여러 컴퓨터로 구성된 클러스터에서 대규모 데이터를 분산 저장·처리하도록 지원하는 오픈소스 프레임워크이다. HDFS와 MapReduce 등이 대표적인 구성요소이다.

③ O: 데이터 마이닝은 데이터에 숨어 있는 패턴과 관계를 찾아 유용한 정보를 얻는 분석 과정이다. 분류, 예측, 군집화, 연관 규칙 분석 등이 활용된다.

④ O: 텍스트 마이닝은 문서와 같은 **비구조화된 텍스트**에서 유용한 정보를 추출하는 기술이다. 자연어 처리 등을 이용해 주요 개체, 주제, 관계 등을 분석하며, 모든 종류의 비정형 데이터를 텍스트 마이닝만으로 처리한다는 뜻은 아니다.

따라서 빅데이터의 대상과 처리 방식을 잘못 한정한 ①이 가장 적절하지 않다.

NoteBase

문 20. OLAP(On-Line Analytical Processing)의 다차원 모델에서의 연산에 대한 설명으로 가장 적절하지 않은 것은?

① 드릴다운(Drill-down)은 요약된 형태의 데이터 수준에서 더 자세한 자료를 보고자 할 때 사용하는 연산이다.

② 피보팅(Pivoting)은 하나의 차원 구조로부터 다른 차원 구조로 변환하는 연산이다.

③ 슬라이싱(Slicing)은 한 차원을 잘라보고 동시에 다른 차원을 자르면서 데이터의 범위를 좁혀가는 연산이다.

④ 드릴스루(Drill-through)는 상세 데이터에서 요약 데이터를 보고자 할 때 사용하는 연산이다.

정답 4번

간단한 해설

상세 자료를 더 높은 수준으로 집계해 보는 것은 **롤업 또는 드릴업**이다. 드릴스루는 요약 결과의 근거가 되는 상세 원천 데이터로 이동하는 기능이다.

선지별 해설

① O: 드릴다운은 요약 수준에서 **더 상세한 계층 수준**으로 내려가 자료를 보는 연산이다. 예를 들어 연도별 매출에서 분기별·월별 매출로 내려가 분석한다.

② O: 피보팅은 **분석 차원의 배치나 축을 바꾸어**다른 관점에서 데이터를 보는 연산이다. 행·열에 놓인 차원을 바꾸는 것이 대표적이며, 원천 데이터의 값 자체를 수정하는 연산은 아니다.

③ O: 차원의 특정 값을 고정해 단면을 보고, 다른 차원의 값도 고정해 범위를 더 좁히는 **슬라이싱의 넓은 용례**로 이해할 수 있다. 한 차원만 고르는 슬라이스와 여러 차원의 값 범위를 고르는 다이스를 엄격히 구별하는 설명도 있으므로, 모든 다차원 범위 선택을 슬라이싱이라고 일반화해서는 안 된다.

④ X: 상세 데이터에서 더 높은 요약 수준으로 올라가는 것은 **롤업(Roll-up), 즉 드릴업**이다. 드릴스루는 요약된 분석 결과에서 그 근거가 되는 원천의 상세 레코드로 이동하여 확인하는 기능이므로 방향과 대상이 다르다.

NoteBase