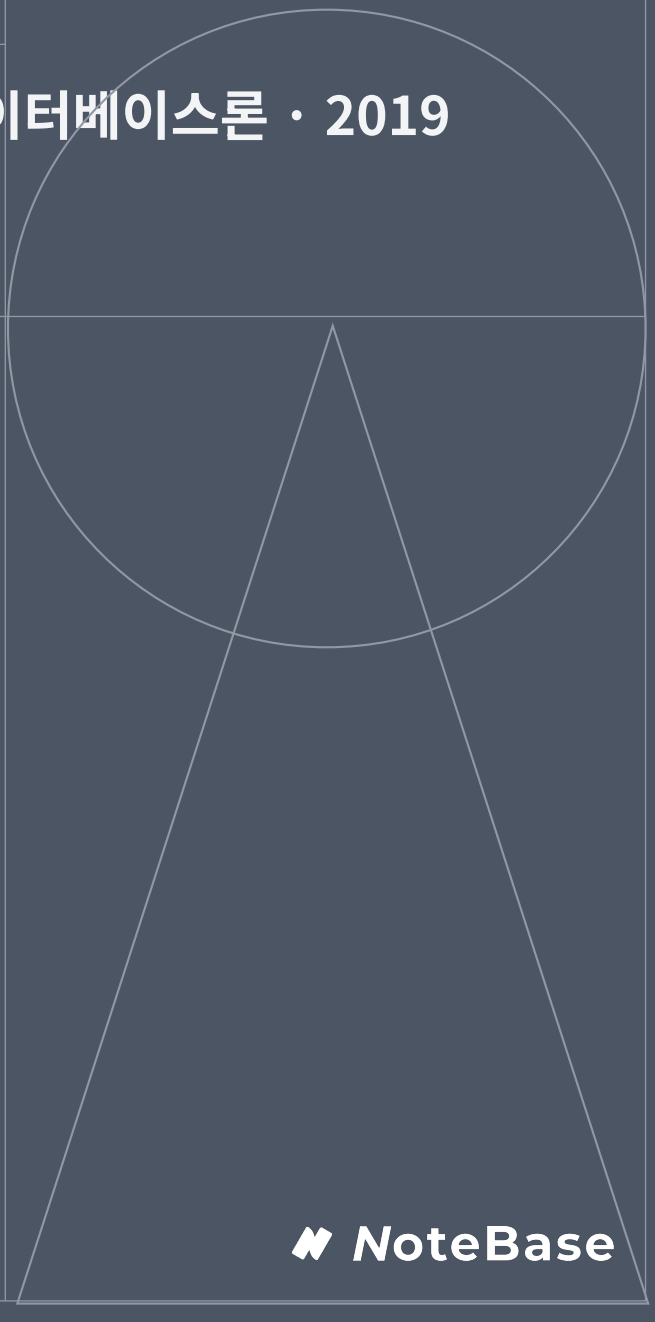
기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2019



NoteBase

기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2019

공무원/경찰/경력채용 · 데이터베이스론 · 2019

문 1. 다음은 데이터베이스 관리 시스템(DBMS)의 주요 구성요소에 대한 설명이다. 옳은 것을 모두 고른 것은?

㉠ 데이터정의어(DDL) 컴파일러는 DDL로 명세된 스키마 정의를 내부 형태로 처리하여 시스템 카탈로그에 저장한다.
㉡ 질의처리기는 질의문을 파싱, 분석, 컴파일하여 런타임 처리기를 통해 실행한다.
㉢ 데이터조작어(DML) 컴파일러는 DML 명령어를 추출 및 파싱하여 DB의 내용에 대한 정확성 및 안정성 유지 기능을 수행한다.
㉣ 트랜잭션 관리자는 무결성 제약조건을 준수하여 데이터 접근 권한 등의 통제수행, 동시성제어, 회복작업 등을 수행한다.

① ㉠㉡㉢
② ㉡㉣
③ ㉠㉡㉣
④ ㉢㉣

정답 3번

간단한 해설

DML 컴파일러는 명령을 분석하고 실행 가능한 형태로 변환하며, **데이터의 정확성·안정성 유지를 총괄하는 구성요소는 아니다**. 트랜잭션 관리자는 트랜잭션 실행을 통제하여 동시 실행과 장애 상황에서도 데이터의 일관성을 유지하도록 한다.

NoteBase

보기별 해설

㉠ O: 데이터정의어인 DDL은 테이블·속성·제약조건 등 데이터베이스 구조를 정의한다. DDL 컴파일러는 이 정의를 DBMS가 사용할 수 있는 내부 표현으로 처리하고, **스키마 정보를 시스템 카탈로그에 저장**한다.

㉡ O: 질의처리기는 질의문의 문법과 의미를 분석하고 실행 계획을 생성한다. 생성된 계획은 런타임 처리기 또는 질의 실행기를 통해 수행되므로, 질의문을 파싱·분석·컴파일하여 실행한다는 설명이 성립한다.

㉢ X: DML 컴파일러는 데이터 검색·삽입·수정·삭제 명령을 분석하고 실행 가능한 명령이나 실행 계획으로 변환한다. **데이터의 정확성·안정성 유지는 DML 컴파일러의 고유 기능이 아니다**. 이는 무결성 제약 검사, 권한 통제, 동시성 제어와 회복 등 DBMS의 실행 관리 기능을 통해 이루어진다. 프로그램에서 내장 DML을 추출하는 기능은 전처리기로 따로 구분하기도 한다.

㉣ O: 트랜잭션 관리자는 트랜잭션이 제약조건을 지키며 실행되도록 관리하고, 동시성 제어와 회복을 통해 데이터의 일관성을 보장한다. 권한 정보를 확인하여 허용된 연산의 실행을 통제하는 기능도 넓은 의미의 트랜잭션 실행 관리에 포함할 수 있다. 따라서 이 문항처럼 실행 통제 기능을 포괄하여 설명하면 성립한다. 다만 세부 DBMS 구성에서는 권한·무결성 검사기를 별도 모듈로 나눌 수 있으므로, **트랜잭션 관리자가 모든 검사를 단독 수행한다는 뜻은 아니다**.

옳은 보기는 ㉠·㉡·㉣이므로 이 조합을 제시한 ③에 해당한다.

NoteBase

문 2. 다음의 15개 숫자 데이터에서 이진 검색 알고리즘을 활용하여 숫자 2를 검색하고자 한다. 이에 대한 설명으로 가장 적절한 것은?

(정렬 전)

11	3	1	6	10	5	14	12	4	15	7	13	8	2	9

(정렬 후)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

① 정렬 후 데이터를 기준으로 2를 찾는 비교횟수는 4회이다.

② 정렬 후 데이터를 기준으로 비교횟수를 거듭할 때마다 검색 대상이 되는 데이터의 수가 약 반(1/2)으로 줄어든다.

③ 피보나치 수열에 따라 다음에 비교할 대상을 선정하여 검색한다.

④ 15개의 숫자 데이터들은 정렬되어 있지 않아도 항상 검색이 가능하다.

정답 2번

간단한 해설

이진 검색은 **정렬된 자료의 중앙 원소와 비교하여 후보 구간을 약 절반씩 줄인다**. 제시된 정렬 후 자료에서 2는 8→4→2의 순서로 총 3회 비교하여 찾는다.

선지별 해설

① X: 정렬된 1~15에서 중앙값 8과 먼저 비교한다. 2가 더 작으므로 1~7의 중앙값 4와 비교하고, 다시 1~3의 중앙값 2와 비교하여 찾는다. 따라서 비교 대상은 **8→4→2로 총 3회**이며 4회가 아니다. 여기서 비교 횟수는 탐색키와 중앙 원소를 대조한 횟수이다.

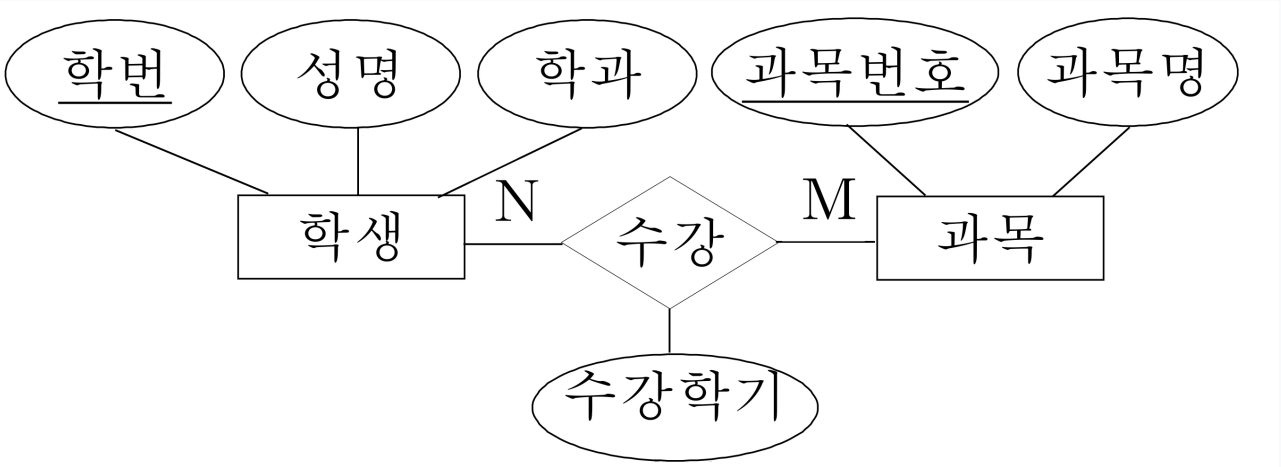
② O: 중앙 원소와 비교해 일치하지 않으면 정렬 순서를 이용하여 왼쪽 또는 오른쪽 구간만 남긴다. 따라서 탐색을 계속할 때마다 **후보 데이터 수가 약 절반**으로 줄어든다. 이 문제에서는 15개→7개→3개 구간으로 좁힌 뒤 2를 찾는다.

③ X: 이진 검색은 **현재 구간의 중앙 위치**를 비교 대상으로 삼는다. 피보나치 수를 이용해 다음 비교 위치와 구간을 정하는 것은 별도의 피보나치 검색이다.

④ X: 이진 검색으로 후보의 절반을 안전하게 제외하려면 **데이터가 검색 기준에 따라 정렬**되어 있어야 한다. 정렬되지 않은 자료에서 중앙값보다 작다는 이유만으로 한쪽 구간을 버리면 찾는 값이 있는 구간을 잘못 제외할 수 있다.

NoteBase

문 3. 다음 ERD(Entity Relationship Diagram)를 관계형 데이터베이스 스키마로 가장 적절하게 변환한 것은? (단, 밑줄(_____)은 기본키를 의미한다.)



① 학생(학번, 성명, 학과), 과목(과목번호, 과목명), 수강(학번, 과목번호)

② 학생(학번, 성명, 학과), 과목(과목번호, 과목명), 수강(학번, 과목번호)

③ 학생(학번, 성명, 학과), 과목(과목번호, 과목명), 수강(학번, 과목번호, 수강학기)

④ 학생(학번, 성명, 학과), 과목(과목번호, 과목명), 수강(학번, 과목번호, 수강학기)

정답 4번

출제 포인트

학생과 과목 사이의 다대다 관계인 수강을 별도 릴레이션으로 변환하고, 관계 속성인 수강학기를 보존하는 문제이다. 릴레이션은 데이터를 행과 열로 나타낸 표이며, 기본키는 각 행을 유일하게 구분하는 속성 또는 속성의 조합이다.

NoteBase

선지별 해설

① X: 수강(학번, 과목번호)에는 도식의 관계 속성인 수강학기가 빠져 있고, 수강 릴레이션의 기본키도 표시되어 있지 않다. 수강학기를 추가하고 학번·과목번호·수강학기를 묶어 기본키로 지정하는 형태로 고친다.
② X: 수강학기를 누락했으며 수강의 기본키를 학번 하나로 지정했다. 학번만 기본키이면 학생 한 명의 수강 행을 하나만 저장할 수 있어 여러 과목을 수강하는 다대다 관계를 표현하지 못한다. 수강학기를 추가하고 학번·과목번호·수강학기를 함께 기본키로 지정해야 한다.
③ X: 수강학기는 보존했지만 수강의 기본키를 학번과 수강학기로만 지정했다. 이 조합으로는 한 학생이 같은 학기에 여러 과목을 수강할 때 각 행을 구분하지 못한다. 기본키에 과목번호도 포함해야 한다.
④ O: 학생의 기본키는 학번, 과목의 기본키는 과목번호로 유지하고, 다대다 관계를 별도 수강 릴레이션으로 만들었다. 수강의 학번과 과목번호는 각각 학생과 과목을 참조하는 외래키가 되고, 관계 속성인 수강학기도 보존된다. 학번·과목번호·수강학기를 묶은 복합 기본키는 학생별·과목별·학기별 수강 기록을 구분하며, 같은 학생이 같은 과목을 다른 학기에 수강한 기록도 구분할 수 있다.

오답 포인트

별도 수강 릴레이션을 만드는 것만으로 변환이 끝나지 않는다. ①·②는 관계 속성인 수강학기를 누락했고, ②·③은 과목별 수강 기록을 구분하는 데 필요한 기본키 구성이 잘못되었다. 특히 ③은 수강학기가 있다는 이유로 고르기 쉽지만, 과목번호에 기본키 밑줄이 없다는 점이 결정적인 차이이다.

다시 나오면 이렇게 판단

먼저 각 개체의 기본키와 관계의 대응 수를 확인한다. 다대다 관계는 양쪽 개체의 키와 관계 속성을 담는 별도 릴레이션으로 변환한다. 이어 제시된 기본키로 한 학생의 여러 과목 수강 기록을 구분할 수 있는지 확인하고, 학기별 기록까지 구분하는 경우 학기 속성의 포함 여부도 확인한다.

암기 포인트

다대다 관계는 별도 릴레이션, 관계 속성은 그 릴레이션에 보존한다. 이 문항의 수강 기본키는 학번·과목번호·수강학기의 조합이다.

NoteBase

문 4. 다음은 무결성의 종류에 대한 설명이다. (가)~(라)에 해당되는 것으로 가장 알맞게 짝지은 것은?

(가) 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다.
(나) 속성 값은 그 속성의 도메인 범위 내에 존재해야 한다.
(다) 기본키에 속해 있는 속성은 널(NULL) 값을 가질 수 없다.
(라) 한 릴레이션에 같은 킷값을 가진 튜플들은 허용하지 않는다.

① (가): 개체 무결성, (나): 키 무결성, (다): 참조 무결성, (라): 도메인 무결성
② (가): 참조 무결성, (나): 도메인 무결성, (다): 개체 무결성, (라): 키 무결성
③ (가): 도메인 무결성, (나): 키 무결성, (다): 참조 무결성, (라): 개체 무결성
④ (가): 참조 무결성, (나): 개체 무결성, (다): 도메인 무결성, (라): 키 무결성

정답 2번

간단한 해설

외래키의 참조 대상 존재 여부는 참조 무결성, 속성의 허용 범위는 도메인 무결성이다. **기본키의 NULL 금지**는 개체 무결성이고, **키 값의 중복 금지**는 키 무결성이다.

대응 관계 해설

(가) 참조 무결성: 외래키를 통해 다른 튜플을 참조할 때 **참조 대상이 존재**해야 한다. 따라서 대응하는 참조 키 값이 없는 외래키 값은 허용하지 않는다. 다만 해당 외래키에 NULL이 허용되면 참조하지 않음을 나타내는 NULL은 가능하다.

(나) 도메인 무결성: 도메인은 속성이 취할 수 있는 값의 집합이다. 자료형과 허용 범위 등의 조건에 맞는 값만 저장하도록 제한하므로 **속성 값의 허용 범위**에 관한 설명이다.

(다) 개체 무결성: 기본키는 각 튜플을 식별하므로 **기본키를 구성하는 모든 속성이 NULL이 아니어야** 한다. 여러 속성으로 이루어진 복합 기본키도 각 구성 속성에 이 조건이 적용된다.

(라) 키 무결성: 키는 서로 다른 튜플을 유일하게 식별해야 하므로 **같은 키 값의 중복을 허용하지 않는다**.
복합키에서는 개별 속성 하나의 값이 아니라 키를 이루는 값의 조합이 유일해야 한다.

따라서 (가) 참조 무결성, (나) 도메인 무결성, (다) 개체 무결성, (라) 키 무결성으로 연결되며, 이 조합은 ②이다.

NoteBase

문 5. 다음의 두 릴레이션 R과 S에 대한 조인(Join) 결과로 옳지 않은 것은?

R

A	B	C
2	나	b
3	다	c
4	라	d
1	가	a

S

B	C	D
나	b	2
나	c	3
라	d	4
가	a	1

① ① R ⋈ S (Natural Join)

A	B	C	D
2	나	b	2
4	라	d	4
1	가	a	1

② ② R ⋈ S (Left Outer Join)

A	B	C	B	C	D
2	나	b	나	b	2
3	다	c	NULL	NULL	NULL
4	라	d	라	d	4
1	가	a	가	a	1

③ ③ R ⋈ S (Right Outer Join)

A	B	C	B	C	D
2	나	b	나	b	2
NULL	NULL	NULL	나	c	3
4	라	d	라	d	4
1	가	a	가	a	1

④ ④ R ⋈ S (Full Outer Join)

A	B	C	B	C	D
3	다	c	NULL	NULL	NULL
NULL	NULL	NULL	나	c	3

정답 4번

간단한 해설

완전 외부 조인은 양쪽에서 짝이 없는 행뿐 아니라 **조인 조건을 만족한 행도 함께 보존**한다. 공통으로 결합되는 세 행을 누락한 결과는 완전 외부 조인이 아니다.

NoteBase

선지별 해설

① O: 자연 조인은 공통 속성 B와 C가 모두 같은 행을 결합하고 공통 열을 한 번만 표시한다. 일치하는 (B, C)는 (나, b), (라, d), (가, a)이므로 결과는 (2, 나, b, 2), (4, 라, d, 4), (1, 가, a, 1)의 세 행이다. **R의 (다, c)와 S의 (나, c)는 B가 달라 결합되지 않는다**.

② O: 왼쪽 외부 조인은 일치하는 세 쌍에 더하여 **왼쪽 R의 미일치 행도 보존**한다. 따라서 R의 (3, 다, c)에 대응하는 S의 B·C·D를 NULL로 채운 (3, 다, c, NULL, NULL, NULL)을 추가하여 네 행이 된다.

③ O: 오른쪽 외부 조인은 일치하는 세 쌍에 더하여 **오른쪽 S의 미일치 행도 보존**한다. S의 (나, c, 3)에 대응하는 R의 A·B·C를 NULL로 채운 (NULL, NULL, NULL, 나, c, 3)을 추가하여 네 행이 된다.

④ X: 완전 외부 조인은 **일치하는 세 행과 양쪽 미일치 두 행을 모두 포함**하므로 총 다섯 행이어야 한다. 열 순서가 (R.A, R.B, R.C, S.B, S.C, S.D)일 때 (2, 나, b, 나, b, 2), (4, 라, d, 라, d, 4), (1, 가, a, 가, a, 1)과 ②·③에서 설명한 NULL 확장 행이 함께 남는다. 제시된 표는 미일치 두 행만 남기고 정상적으로 결합된 세 행을 빠뜨렸다.

NoteBase

문 6. 다음의 릴레이션 tbl_salary(index, name, address, age, salary)에 대해, 여러 관계 대수식을 수행하려고 한다. 관계 대수식의 결과로 생성되는 릴레이션의 차수(Degree)와 카디날리티(Cardinality)로 옳지 않은 것은?

index	name	address	age	salary
1	차	서울	35	350
2	경	대전	27	270
3	강	경남	31	320
4	김	경기	28	270
5	박	경기	29	290
6	최	부산	20	230
7	오	전북	23	250

① 대수식: σaddress='경기'(tbl_salary) ∪ σaddress='서울'(tbl_salary) (결과) 차수: 5, 카디날리티: 3

② 대수식: σsalary>=300(tbl_salary) - σaddress='경기'(tbl_salary) (결과) 차수: 5, 카디날리티: 2

③ 대수식: Πname,address(σage>29(tbl_salary) ∪ σage<=23(tbl_salary)) (결과) 차수: 3, 카디날리티: 4

④ 대수식: Πname,age,salary(σage>28(tbl_salary)) (결과) 차수: 3, 카디날리티: 3

정답 3번

간단한 해설

차수는 속성 수, 카디날리티는 튜플 수이다. name과 address만 추출하는 프로젝션의 결과는 **차수 2**이며, 해당 네 행의 이름·주소 쌍은 서로 달라 카디날리티는 4로 유지된다.

NoteBase

선지별 해설

① O: 주소가 경기인 행은 index 4, 5이고 서울인 행은 index 1이다. 합집합 결과는 서로 다른 3개 행이며, 속성을 제거하는 연산이 없으므로 원래의 5개 속성이 유지된다. 차수 5, 카디날리티 3이다.

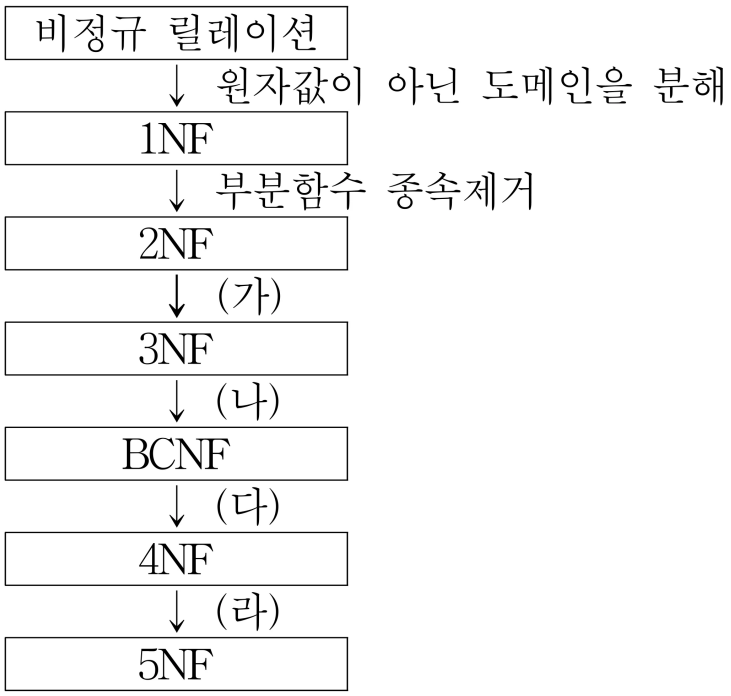
② O: salary가 300 이상인 행은 index 1, 3이다. 주소가 경기인 행은 index 4, 5이므로 앞의 두 행 중 차집합에서 제거될 행이 없다. 속성 5개와 행 2개가 남아 차수 5, 카디날리티 2이다.

③ X: age가 29보다 큰 행은 index 1, 3이고 23 이하인 행은 index 6, 7이다. 합집합의 네 행에서 name과 address만 추출하면 (차, 서울), (강, 경남), (최, 부산), (오, 전북)이 남는다. 중복이 없으므로 카디날리티는 4이지만, **남긴 속성은 두 개이므로 차수는 2**이다. 차수 3을 2로 고쳐야 한다.

④ O: age가 28보다 큰 행은 index 1, 3, 5이다. 여기서 name, age, salary를 추출하면 (차, 35, 350), (강, 31, 320), (박, 29, 290)으로 중복 없는 세 행이 된다. 차수 3, 카디날리티 3이다.

NoteBase

문 7. 다음은 정규화 과정을 도식화한 것이다. 정규화 과정 중 (가)~(라)에 해당되는 과정을 <보기>에서 골라 알맞게 나열한 것은?



<보기>

㉠ 후보키를 통하지 않은 조인종속 제거

㉡ 결정자가 후보키가 아닌 함수종속 제거

㉢ 이행함수 종속제거

㉣ 함수종속이 아닌 다치종속 제거

① (가): ㉢, (나): ㉡, (다): ㉣, (라): ㉠

② (가): ㉡, (나): ㉠, (다): ㉢, (라): ㉣

③ (가): ㉡, (나): ㉣, (다): ㉢, (라): ㉠

④ (가): ㉢, (나): ㉣, (다): ㉠, (라): ㉡

정답 1번

간단한 해설

정규화 단계별 핵심은 **3NF의 이행종속, BCNF의 키가 아닌 결정자, 4NF의 다치종속, 5NF의 조인종속**을 구별하는 것이다. 그림의 각 화살표가 도달하는 정규형에 맞춰 보기의 과정을 대응시킨다.

NoteBase

빈칸별 해설

(가) 2NF→3NF: ㉢ **이행함수 종속제거**에 해당한다. 2NF에서 비주요 속성이 키에 이행적으로 종속되어 생기는 중복을 해소하는 단계이다. 3NF의 엄밀한 조건은 비자명한 함수종속 X→A마다 X가 슈퍼키이거나 A가 주요 속성인 것이다.

(나) 3NF→BCNF: ㉡ **결정자가 후보키가 아닌 함수종속 제거**에 대응한다. 보기의 표현은 키가 아닌 결정자로 인한 문제를 제거한다는 취지이며, 엄밀한 BCNF 조건은 모든 비자명한 함수종속의 결정자가 슈퍼키여야 한다는 것이다. 최소성을 갖춘 슈퍼키가 후보키이므로 두 용어를 무조건 같은 뜻으로 쓰지는 않는다.

(다) BCNF→4NF: ㉣ **함수종속이 아닌 다치종속 제거**에 해당한다. 함수종속만으로 설명되지 않는 독립적인 다중값의 조합에서 생기는 중복을 분해로 해소한다. 모든 다치종속을 없앤다는 뜻이 아니라, 비자명한 다치종속의 결정자가 슈퍼키여야 한다는 조건을 충족시키는 것이다.

(라) 4NF→5NF: ㉠ **후보키를 통하지 않은 조인종속 제거**에 해당한다. 후보키로부터 함의되지 않는 비자명한 조인종속으로 생기는 중복을 해소하여, 적절히 분해한 릴레이션의 조인으로 원래 관계를 손실 없이 재구성하도록 한다.

따라서 (가)~(라)는 ㉢, ㉡, ㉣, ㉠에 대응하며 ①과 일치한다.

NoteBase

문 8. 다음의 릴레이션 R에 존재하는 함수 종속성을 나타낸 것으로 옳은 것은?

R

A	B	C	D	E
1	c	20	a	1
1	b	30	b	3
2	c	20	a	2
2	c	20	a	3
2	a	30	c	2
3	b	20	b	2
3	b	30	b	1

① B→DE

② A→BC

③ AB→E

④ CD→B

정답 4번

간단한 해설

주어진 표에서 **(C, D)가 같은 행들은 B도 모두 같다**. 반면 다른 선지에는 결정자 값이 같은데 종속 속성 값이 달라지는 행이 존재한다.

NoteBase

선지별 해설

함수 종속 X→Y는 X 값이 같은 모든 튜플에서 Y 값도 같다는 뜻이다. 여기서는 제시된 릴레이션 상태의 모든 행을 비교하며, 이 표만으로 앞으로 허용될 모든 데이터의 업무 제약까지 확정하는 것은 아니다.

① X: B가 c인 첫째 행과 셋째 행에서 E는 각각 1과 2이다. **B가 같아도 E가 다르므로** B→DE는 성립하지 않는다. 주어진 표에서 B→D는 성립하지만, D와 E를 모두 결정한다는 주장과는 다르다.

② X: A가 1인 첫째 행과 둘째 행에서 (B, C)는 각각 (c, 20), (b, 30)이다. A가 같은데 B와 C가 달라지므로 A→BC는 성립하지 않는다. A만으로는 B와 C의 값을 결정할 수 없다.

③ X: 셋째 행과 넷째 행은 (A, B)가 모두 (2, c)이지만 E가 각각 2와 3이다. **결정자 AB의 값이 같을 때 E가 달라지는 반례**가 있으므로 AB→E는 성립하지 않는다.

④ O: (C, D)가 (20, a)인 세 행은 B가 모두 c이고, (30, b)인 두 행은 B가 모두 b이다. 나머지 (30, c)는 B가 a, (20, b)는 B가 b로 각각 하나씩 대응한다. 따라서 **같은 (C, D)에 서로 다른 B가 대응하는 경우가 없어** 주어진 표에서 CD→B가 성립한다.

NoteBase

문 9. 다음의 tbl_student 테이블을 이용하여 아래의 SQL을 수행하였을 때 예상되는 결과로 옳은 것은?

id	name	grade	subject	score
2017001	Ryu	2	math	60
2017002	Cho	1	kor	80
2019006	Kim	1	kor	55
2018002	Yang	3	eng	85
2018004	Park	2	math	45
2016003	Choi	3	eng	55
2016006	Kang	3	eng	60

```
SELECT count(*) FROM tbl_student
GROUP BY subject HAVING count(*) > 2;
```

① 0

② 1

③ 2

④ 3

정답 4번

간단한 해설

과목별 행 수는 math 2, kor 2, eng 3이다. HAVING 조건을 만족하는 **eng 그룹의 COUNT(*) 값 3**이 출력된다.

NoteBase

계산 과정

GROUP BY subject로 과목별 그룹을 만든 뒤 COUNT(*)로 각 그룹의 행 수를 센다. 표에서 math는 Ryu·Park의 **2행**, kor는 Cho·Kim의 **2행**, eng는 Yang·Choi·Kang의 **3행**이다.

HAVING COUNT(*) > 2는 집계 결과가 2보다 큰 그룹만 남긴다. 따라서 math와 kor는 제외되고 **eng 그룹만 남는다**.

SELECT COUNT(*)가 출력하는 값은 남은 그룹의 행 수인 **3**이다. 결과 행 자체는 1행이지만, 문제의 질의가 출력하는 값은 결과 행의 개수가 아니라 해당 그룹의 집계값이다.

따라서 SQL 실행 결과 값은 ④ 3이다.

NoteBase

문 10. 다음의 SQL에서 명령어 짝의 사용이 적절하지 않은 것을 모두 고른 것은?

㉠ UPDATE ... / AS ...
㉡ CREATE VIEW ... / SET ...
㉢ SELECT.../FROM.../WHERE.../GROUP BY ... /HAVING ...

① ㉠
② ㉠㉡
③ ㉡㉢
④ ㉠㉢

정답 2번

간단한 해설

주요 구문의 연결어를 구별하면, UPDATE에서 수정값을 지정하는 절은 **SET**이고 CREATE VIEW에서 정의 조회문을 연결하는 것은 **AS**이다. 별칭을 붙이는 AS의 용도와 혼동하지 않는다.

보기별 해설

㉠ X: 수정할 열과 값을 지정하는 UPDATE의 기본 짝은 **SET**이다. 기본 구조는 UPDATE 테이블명 SET 열명 = 값이며, AS로 SET을 대신할 수 없다. 일부 DBMS에서는 UPDATE 대상 테이블의 별칭을 붙일 때 AS를 쓸 수 있지만, 이는 여기서 묻는 값 지정 절과는 다른 용도이다.

㉡ X: 뷰를 정의하는 조회문을 연결하는 CREATE VIEW의 기본 짝은 **AS**이다. CREATE VIEW 뷰명 AS SELECT ... 형태이며, 뷰 정의 조회문 앞에 SET을 쓰는 것이 아니다.

㉢ O: SELECT에 조회할 항목, FROM에 대상 테이블, WHERE에 행 조건, GROUP BY에 그룹 기준, HAVING에 그룹 조건을 적는 구성은 올바르다. 이는 SQL 문장의 작성 순서이며, 논리적 처리 순서를 뜻하는 것은 아니다.

기본 구문에서 짝이 잘못 연결된 것은 ㉠과 ㉡이므로 ②이다.

NoteBase

문 11. 다음의 뷰에 대한 설명으로 적절하지 않은 것을 모두 고른 것은?

㉠ 관계형 데이터베이스에서 뷰는 사용자의 전체 외부 스키마 대신에 하나의 가상 릴레이션(Virtual Relation)을 의미한다.
㉡ 뷰에 대한 검색은 기본 테이블과 동일하기 때문에 삽입, 삭제, 갱신의 경우 제약을 받지 않는다.
㉢ 뷰가 기본키를 포함한 행과 열의 부분 집합으로 구성되어도 갱신 연산은 수행이 불가능하다.
㉣ 데이터베이스의 구조가 바뀌어도 뷰를 정의하는 질의만 변경하면 되므로, 기존의 응용 프로그램을 다시 작성할 필요성을 줄일 수 있다.
㉤ 뷰는 독자적으로 인덱스를 가질 수 있으며, 뷰의 정의를 쉽게 변경할 수 있다.

① ㉠㉡㉤
② ㉡㉢㉤
③ ㉠㉢㉣
④ ㉢㉣㉤

정답 2번

간단한 해설

뷰의 검색과 갱신 가능성은 별개이며, **단순 뷰는 조건에 따라 갱신할 수 있다**. 일반적인 가상 뷰는 자체 저장 인덱스를 갖지 않고, 정의 변경도 아무 제약 없이 이루어지는 것은 아니다.

NoteBase

보기별 해설

㉠ O: 관계형 데이터베이스의 뷰는 질의로 정의한 하나의 가상 릴레이션이다. 사용자가 보는 외부 스키마 전체와 뷰 하나를 동일시하지 않으며, 여러 뷰가 사용자 관점을 구성할 수 있다.

㉡ X: 뷰를 기본 테이블처럼 검색할 수 있다는 사실이 **삽입·삭제·갱신까지 무제한으로 가능함을 뜻하지는 않는다**. 집계나 그룹화 등으로 정의된 뷰는 변경할 기본 행을 직접 대응시키기 어려워 갱신에 제약이 있다.

㉢ X: 기본 테이블의 행과 열 일부를 선택한 단순 뷰는 갱신할 수 있다. 기본키를 포함하여 원래 행을 식별할 수 있고 갱신 가능 조건을 충족하면 기본 테이블의 변경으로 연결되므로 **갱신이 항상 불가능한 것은 아니다**.

㉣ O: 기본 테이블의 구조가 바뀌어도 뷰가 제공하는 열과 의미를 유지하도록 정의를 조정할 수 있다면, 응용 프로그램은 기존 뷰를 계속 이용할 수 있다. 이처럼 뷰는 논리적 데이터 독립성을 높여 프로그램 수정 필요성을 줄인다. 모든 구조 변경을 무조건 숨길 수 있다는 뜻은 아니다.

㉤ X: 이 문항의 일반적인 가상 뷰는 자체 데이터를 저장하지 않으므로 **독자적인 저장 인덱스를 갖지 않는다**. 뷰 정의의 변경도 의존 객체와 DBMS의 지원·제약을 고려해야 하므로 언제나 쉽게 바꿀 수 있다고 일반화할 수 없다. SQL Server의 인덱싱된 뷰처럼 결과를 물리적으로 저장하는 특수 기능은 일반 가상 뷰와 구별된다.

따라서 적절하지 않은 설명은 ㉡·㉢·㉤이며, 해당 조합은 ②이다.

NoteBase

문 12. 물리적 데이터베이스 설계 시 고려해야 할 사항으로 가장 적절하지 않은 것은?

① 저장 레코드 형식 설계 및 저장 레코드 집중화

② 수행될 질의와 트랜잭션의 예상 빈도

③ 질의문의 조건 절에서 자주 사용하는 애트리뷰트에 대한 인덱스 생성

④ 클러스터 인덱스를 여러 개의 인덱스로 정의

정답 4번

간단한 해설

한 테이블의 행을 여러 독립적인 클러스터 키 순서로 동시에 조직할 수는 없다. **클러스터 인덱스는 테이블당 하나**이며, 여러 검색 경로는 비클러스터 인덱스와 구별한다.

선지별 해설

① O: 물리적 설계는 논리적으로 정의된 데이터를 실제 저장장치에 어떻게 배치할지 결정한다. 레코드의 저장 형식과 함께 자주 접근하는 레코드를 가까이 배치하는 집중화는 **저장 공간과 디스크 입출력 효율**에 영향을 주므로 고려 대상이다.

② O: 같은 데이터 구조라도 자주 실행되는 검색과 갱신의 비율에 따라 유리한 저장 방식이 달라진다. 따라서 **질의·트랜잭션의 예상 빈도와 접근 패턴**을 분석하여 인덱스와 파일 조직을 선택해야 한다.

③ O: 조건절에서 자주 검색하는 속성은 인덱스 후보가 된다. 다만 **검색 이득과 갱신·저장 비용을 함께 평가**해야 하며, 자주 쓰인다는 이유만으로 모든 속성에 무조건 인덱스를 만드는 것은 아니다.

④ X: 클러스터 인덱스는 테이블 행의 저장 조직을 하나의 키 순서에 맞추는 인덱스이므로 **한 테이블에 여러 독립 클러스터 인덱스를 정의할 수 없다**. 여러 검색 경로는 비클러스터 인덱스로 제공할 수 있다. 여러 열을 묶은 하나의 복합 클러스터 인덱스와 여러 개의 클러스터 인덱스는 다르다.

NoteBase

문 13. 데이터베이스 설계 단계의 튜닝과정에서 고려해야 할 사항에 대한 설명으로 가장 적절하지 않은 것은?

① 성능 향상을 위해 릴레이션을 수직 또는 수평으로 분할할 수 있다.

② 성능 향상을 위해 롤백 세그먼트를 하나로 통합하여 구성할 수 있다.

③ 자주 함께 조인되는 테이블들을 역정규화하여 조인 경로를 줄일 수 있다.

④ 분산 데이터베이스를 적용했을 경우, 원격 데이터베이스를 이용할 때 성능 저하가 예상된다면 스냅샷을 이용한 복제 테이블 생성을 고려할 수 있다.

정답 2번

간단한 해설

롤백 세그먼트는 트랜잭션 부하를 **분산하여 경합을 줄이는 방향**으로 관리한다. 하나로 통합하는 것은 경합을 집중시킬 수 있어 일반적인 성능 개선 방안으로 보기 어렵다.

선지별 해설

① O: 접근 패턴에 맞추어 행을 나누는 수평 분할이나 열을 나누는 수직 분할을 적용하면, 조회할 데이터 범위나 읽기 부담을 줄일 수 있다. 분할 후 조인·통합 비용도 함께 고려해야 하며, 분할 자체가 항상 성능 향상을 보장하지는 않는다.

② X: 전통적인 수동 롤백 세그먼트 관리에서는 여러 트랜잭션의 부하를 적절한 수의 세그먼트에 **분산하여 경합을 줄이는 것**이 중요하다. 하나로 통합하면 경합이 집중될 수 있으므로, 이를 일반적인 성능 향상 방안으로 제시하는 것은 부적절하다. 현대 DBMS의 자동 UNDO 관리와는 구별하여 이해한다.

③ O: 자주 함께 조회되는 정보를 역정규화하면 반복적인 조인과 접근 경로를 줄일 수 있다. 대신 데이터 중복, 갱신 비용, 일관성 유지 부담이 늘 수 있으므로 조회 이득과 비교하여 결정한다.

④ O: 원격 테이블에 반복 접근하는 비용이 크다면 스냅샷이나 구체화된 뷰 형태의 복제본을 가까운 위치에 두어 조회 비용을 줄일 수 있다. 다만 복제본의 갱신 주기, 최신성 요구와 동기화 비용을 함께 고려해야 한다.

따라서 설계 단계의 일반적인 튜닝 방안으로 가장 적절하지 않은 것은 ②이다.

NoteBase

문 14. 다음의 RAID(Redundant Array of Inexpensive Disks)에 대한 설명으로 적절한 것을 모두 고른 것은?

㉠ RAID는 디스크 시스템의 성능을 향상시키기 위해 여러 개의 디스크를 병렬로 연결하여 하나의 고성능 논리적 디스크처럼 동작한다.
㉡ RAID에서 디스크 시스템의 성능 향상은 데이터를 다중 디스크에 투명하게 분산시키는 미러링(Mirroring) 기법을 통해 구현한다.
㉢ RAID 0(Stripping)은 데이터의 빠른 입출력을 위해 데이터를 여러 드라이브에 분산 저장하므로 읽기 및 쓰기 성능이 뛰어나지만 복구를 위한 추가적인 정보를 기록하지 않아 장애가 발생하면 데이터는 모두 손실된다.
㉣ RAID 3(Pararell Transfer with Parity)은 RAID 0의 빠른 속도와 RAID 1의 안정적인 복구 기능을 합쳐 놓은 방식이다.
㉤ RAID 5(Independent Data Disks with Distributed Parity Blocks)는 패리티 정보를 모든 드라이브에 나눠 기록하기 때문에 패리티를 담당하는 디스크가 병목현상을 일으키지 않아 다중 처리 시스템에 보다 적합하다.

① ㉠㉡㉢
② ㉠㉢㉤
③ ㉠㉡㉣
④ ㉢㉣㉤

정답 2번

간단한 해설

데이터를 나누어 저장하는 것은 **스트리핑**, 같은 데이터를 복제하는 것은 미러링이다. RAID 3은 전용 패리티를, RAID 5는 분산 패리티를 사용하며, RAID 0에는 장애 복구용 중복 정보가 없다.

NoteBase

보기별 해설

㉠ O: RAID는 여러 물리 디스크를 묶어 하나의 논리적 저장장치처럼 사용한다. 데이터를 분산하여 병렬 입출력을 활용하거나, 구성 수준에 따라 중복 정보를 두어 가용성을 높인다.

㉡ X: 데이터를 여러 디스크에 나누어 배치하는 기법은 **스트리핑(striping)**이다. 미러링은 같은 데이터의 복사본을 별도 디스크에 저장하는 기법이다. 미러링이 읽기 성능에 도움을 줄 수는 있지만, 분산 배치 자체를 미러링이라고 부르지는 않는다.

㉢ O: RAID 0은 스트리핑을 사용하지만 복사본이나 패리티 같은 복구용 중복 정보를 저장하지 않는다. 구성 디스크가 고장 나면 배열 전체의 데이터 이용과 완전한 복구가 보장되지 않는다. 여기서 전체 손실은 논리적 배열의 손실 위험을 뜻하며, 정상 디스크의 모든 비트까지 물리적으로 지워진다는 의미는 아니다.

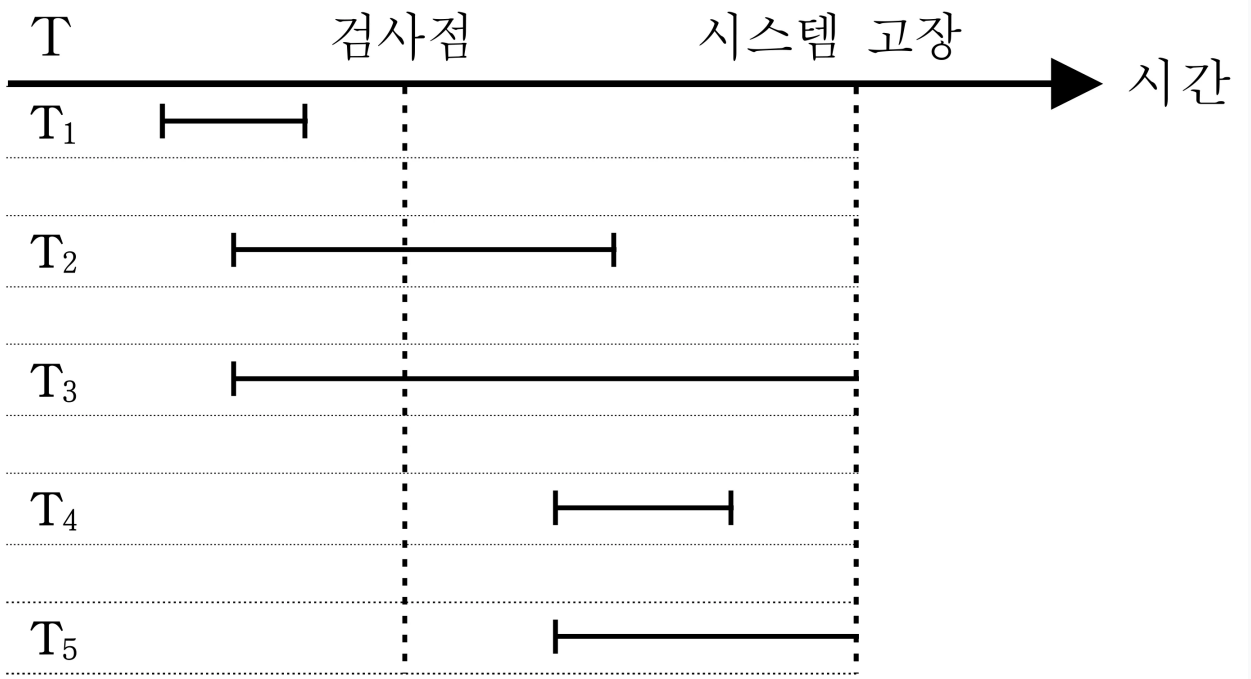
㉣ X: RAID 3은 데이터를 바이트 단위로 분산하고 **전용 패리티 디스크**를 두어 장애에 대응하는 방식이다. RAID 1의 미러링을 결합한 구조가 아니다. RAID 0의 스트리핑과 RAID 1의 미러링을 결합하는 대표적인 구성은 RAID 10이다.

㉤ O: RAID 5는 블록 단위로 데이터를 분산하고 패리티도 여러 디스크에 분산한다. 따라서 한 전용 패리티 디스크에 쓰기가 집중되는 병목을 완화하여 독립적인 여러 요청을 처리하는 데 유리하다. 다만 패리티 계산·갱신 비용까지 없어지는 것은 아니다.

적절한 설명은 ㉠, ㉢, ㉤이므로 ②이다.

NoteBase

문 15. 다음의 검사점(Checkpoint)이 있는 로그(Log)에서 시스템 고장이 발생하는 경우, 트랜잭션 T에 대한 Undo-list와 Redo-list를 바르게 나열한 것은?



① Undo-list: {T3, T5}, Redo-list: {T2, T4}

② Undo-list: {T3, T4, T5}, Redo-list: {T1, T2}

③ Undo-list: {T1, T3, T5}, Redo-list: {T2, T4}

④ Undo-list: {T3, T5}, Redo-list: {T1, T2, T4}

정답 1번

간단한 해설

고장 시 미완료인 트랜잭션은 UNDO, 검사점 이후 완료된 트랜잭션은 REDO로 분류한다. **T3·T5는 취소하고 T2·T4는 재수행**하며, 검사점 전 완료된 T1은 제외한다.

NoteBase

구성 요소별 해설

T1: 검사점 전에 시작하여 검사점 전에 완료되었다. 이 문제의 기본 검사점 회복 모형에서는 완료된 결과가 검사점에서 반영되어 있으므로 **UNDO와 REDO 모두의 대상에서 제외**한다.

T2: 검사점 전에 시작했지만 검사점 이후, 고장 이전에 완료되었다. 완료된 결과를 보존해야 하므로 **REDO 대상**이다. 시작 시점이 검사점 이전이라는 이유로 제외하지 않는다.

T3: 검사점 전에 시작하여 고장 시점까지 완료되지 않았다. 미완료 트랜잭션의 변경이 남지 않도록 **UNDO 대상**으로 분류한다. 필요한 취소 로그는 검사점 이전 시작 지점까지 확인할 수 있다.

T4: 검사점 이후에 시작하여 고장 전에 완료되었다. 성공적으로 완료된 변경을 복구해야 하므로 **REDO 대상**이다.

T5: 검사점 이후에 시작했으나 고장 시점까지 완료되지 않았다. 검사점 당시 실행 중 목록에 없었더라도 **고장 시 미완료이므로 UNDO 대상**이다.

따라서 Undo-list는 {T3, T5}, Redo-list는 {T2, T4}이며, ①과 일치한다.

NoteBase

문 16. 트랜잭션의 병행제어를 위해 사용하는 로킹(Locking)기법에 대한 설명 중 가장 적절하지 않은 것은?

① 데이터 A에 대하여 트랜잭션 T의 배타 잠금(Lock)이 설정되어 있다면 그 잠금이 해제될 때까지는 다른 어떤 트랜잭션도 데이터 A에 대하여 잠금을 설정하지 못하고 대기해야 한다.

② 특정 스케줄에 참여하고 있는 모든 트랜잭션들이 2단계 로킹 기법(2PL: 2Phase Locking)을 준수한다면, 그 스케줄은 직렬 가능하므로 항상 옳다고 간주할 수 있다.

③ 엄격한 2PL 기법의 경우 엄격한 관리를 통하여 교착상태(Deadlock)를 막을 수 있다.

④ 2PL은 크게 확장단계(Growing Phase)와 수축단계(Shrinking Phase)로 구성된다.

정답 3번

간단한 해설

엄격한 2PL은 배타 잠금을 트랜잭션 종료까지 유지하여 연쇄 복귀를 방지하지만, **서로의 잠금을 기다리는 교착상태는 발생할 수 있다**. 직렬 가능성·회복 특성과 교착상태 예방은 구별해야 한다.

선지별 해설

① O: 공유 잠금과 배타 잠금을 사용하는 기본 로킹 모형에서 배타 잠금은 다른 트랜잭션의 **공유 잠금·배타 잠금 모두와 양립하지 않는다**. 따라서 T가 A의 배타 잠금을 보유하는 동안 다른 트랜잭션의 A에 대한 잠금 요청은 허용되지 않고 대기한다.

② O: 모든 트랜잭션이 2PL을 준수하면 그 스케줄은 **충돌 직렬 가능**하다. 원문의 '항상 옳다'는 각 트랜잭션 자체가 올바르다는 전제에서 병행 실행이 어떤 직렬 실행과 동등하다는 뜻이다. 모든 업무 오류나 교착상태가 사라지거나 기본 2PL만으로 회복 가능성까지 보장된다는 뜻은 아니다.

③ X: 엄격한 2PL은 배타 잠금을 커밋 또는 중단까지 유지하여 미완료 변경을 다른 트랜잭션이 읽거나 덮어쓰는 일을 막지만, **교착상태까지 예방하지는 않는다**. 예를 들어 T1이 A를, T2가 B를 잠근 뒤 서로 상대방의 자원 잠금을 기다리면 엄격한 2PL에서도 순환 대기가 생긴다. 교착상태의 예방이나 탐지·복구는 별도로 필요하다.

④ O: 확장단계에서는 새 잠금을 획득할 수 있지만 기존 잠금을 해제하지 않는다. 수축단계에서는 잠금을 해제할 수 있지만 **새 잠금을 획득할 수 없다**. 한 번 해제를 시작하면 다시 확장단계로 돌아가지 않는 것이 2PL의 핵심이다.

NoteBase

문 17. 테이블 student에 대한 권한을 부여하는 ㉠과 회수하는 ㉡을 SQL로 바르게 표현한 것은?

㉠ 테이블 student에 대한 SELECT, INSERT 권한을 Kim과 Lee에게 부여한다.
㉡ 테이블 student에 대한 SELECT, INSERT 권한을 Lee로부터 회수한다.

① ㉠ GRANT SELECT, INSERT ON student TO Kim, Lee;
㉡ REVOKE SELECT, INSERT ON student FROM Lee;

② ㉠ AUTHORIZE SELECT, INSERT ON student TO Kim, Lee;
㉡ CANCEL SELECT, INSERT ON student FROM Lee;

③ ㉠ AUTHORIZE student TO Kim, Lee ON SELECT, INSERT;
㉡ CANCEL student TO Lee ON SELECT, INSERT;

④ ㉠ GRANT student TO Kim, Lee ON SELECT, INSERT;
㉡ REVOKE student TO Lee ON SELECT, INSERT;

정답 1번

간단한 해설

권한 부여는 **GRANT … ON … TO**, 권한 회수는 **REVOKE … ON … FROM**을 사용한다. 권한 목록 뒤에 대상 테이블, 마지막에 사용자를 지정한다.

대응 관계 해설

㉠ 권한 부여: **GRANT 권한 목록 ON 객체 TO 사용자 목록**의 순서로 작성한다. 따라서 student 테이블의 조회·삽입 권한을 Kim과 Lee에게 부여하는 문장은 GRANT SELECT, INSERT ON student TO Kim, Lee;이다.

㉡ 권한 회수: **REVOKE 권한 목록 ON 객체 FROM 사용자 목록**의 순서로 작성한다. 따라서 Lee에게 부여한 해당 권한을 회수하는 문장은 REVOKE SELECT, INSERT ON student FROM Lee;이다. 이 문장에서 Kim은 회수 대상이 아니다.

AUTHORIZE·CANCEL을 사용하거나 객체와 권한의 위치를 뒤바꾼 문장이 아니라, 두 구문을 모두 올바르게 표현한 ①이 정답이다.

NoteBase

문 18. 분산 데이터베이스 시스템의 장점에 대한 설명으로 가장 적절하지 않은 것은?

① 데이터 처리량을 증가시켜야 할 경우 중앙 집중 시스템은 보다 큰 용량의 시스템으로 대체하는 것이 일반적이지만 분산 시스템에서는 새로운 노드를 추가함으로써 가능하게 할 수 있다.

② 어느 한 사이트의 장애나 개별적인 통신 장애가 있을 경우 성능이 저하되기는 하지만 그 기능을 계속 수행할 수 있다.

③ 중앙집중형 데이터베이스 시스템보다 구현이 간단하기 때문에 소프트웨어 개발비용을 더 많이 줄일 수 있다.

④ 임의의 질의어가 여러 사이트에 있는 데이터를 이용한다면, 여러 개의 부 질의로 분할하여 병렬로 실행할 수 있다.

정답 3번

간단한 해설

분산 데이터베이스는 노드 추가, 장애 대응, 병렬 처리에 장점이 있지만 **사이트 간 조정과 일관성·복구 관리로 구현이 복잡해진다**. 구현이 간단해 개발비용이 줄어든다는 설명은 이러한 특성과 반대이다.

선지별 해설

① O: 중앙 집중 시스템의 용량 증설은 한 장비의 성능을 높이는 수직 확장으로 접근하는 경우가 많다. 분산 시스템은 **새 노드를 추가하는 수평 확장**을 통해 저장량이나 처리량을 늘릴 수 있다. 실제 확장 효과는 데이터 분산과 부하 균형, 통신 비용에 영향을 받는다.

② O: 분산·복제와 장애 대응을 적절히 구성하면 일부 사이트나 통신 경로가 고장 나도 **정상 사이트에서 처리 가능한 기능을 계속 수행**할 수 있다. 이는 가용성 측면의 장점이다. 다만 모든 장애에서 모든 기능을 보장한다는 뜻은 아니며, 필요한 데이터의 위치·복제 구성·일관성 정책에 따라 일부 작업은 중단될 수 있다.

③ X: 분산 데이터베이스는 여러 사이트 사이의 통신, 동시성 제어, 분산 트랜잭션, 복제 일관성 및 장애 복구를 다뤄야 한다. 따라서 일반적으로 **구현과 관리가 더 복잡**하며, 구현이 간단해서 소프트웨어 개발비용이 줄어든다는 설명은 장점으로 보기 어렵다.

④ O: 여러 사이트의 데이터를 사용하는 질의는 **부질의로 나누어 병렬 처리**하고 결과를 결합할 수 있다. 실행 시간 단축 가능성이 있지만, 부질의 간 의존 관계와 데이터 전송 비용 때문에 모든 질의가 같은 정도로 빨라지는 것은 아니다.

NoteBase

문 19. 다음은 데이터마이닝의 응용 프로그램에 대한 설명이다. (가)~(라)에 해당되는 것으로 알맞게 짝지은 것은?

(가) 데이터를 서로 다른 부류 혹은 클래스로 나누어주는 모델을 학습하는 과정
(나) 유사한 레코드들은 같은 그룹에 배치하고 서로 다른 레코드들은 다른 그룹에 배치시키고, 그룹들 간에는 서로 겹치지 않음
(다) 연속성을 갖는 변수의 미래 값을 예측하는 방법
(라) 인간이 경험으로부터 학습해 가는 두뇌의 신경망 활동에 착안한 기법

① (가): classification rule, (나): clustering, (다): regression, (라): neural networks
② (가): clustering, (나): classification rule, (다): association rule, (라): regression
③ (가): classification rule, (나): association rule, (다): regression, (라): neural networks
④ (가): clustering, (나): classification rule, (다): neural networks, (라): association rule

정답 1번

간단한 해설

정해진 범주를 예측하는 분류와 유사성으로 그룹을 찾는 군집화를 구별한다. **연속적인 수치 예측**은 회귀, 두뇌의 신경 연결에서 착안한 모형은 신경망이다.

대응 관계 해설

(가) classification rule: 분류는 학습한 모델을 이용해 데이터를 정해진 범주·클래스에 배정한다. **범주를 구별하는 모델을 학습**한다는 점이 핵심이며, 유사한 데이터끼리 그룹 자체를 찾는 군집화와 구별한다.

(나) clustering: 군집화는 데이터의 유사성을 기준으로 그룹을 구성한다. 제시문의 서로 겹치지 않는 그룹은 각 레코드를 하나의 군집에 배정하는 방식이다. 이는 함께 나타나는 항목 사이의 연관 규칙을 찾는 association rule과 다르다.

(다) regression: 회귀는 변수 사이의 관계를 학습하여 **연속적인 수치값**을 예측하는 방법이다. 미래 값 예측에도 활용되며, 범주 이름을 출력하는 분류와 예측 대상이 다르다.

(라) neural networks: 신경망은 생물학적 신경 연결에서 착안한 계산 모형으로, 연결된 노드와 가중치를 이용하여 데이터의 패턴을 학습한다. 뇌의 신경망 활동에서 착안했다는 단서가 대응된다.

따라서 (가) classification rule, (나) clustering, (다) regression, (라) neural networks로 연결한 ①이다.

NoteBase

문 20. 다음과 같은 과목수강 트랜잭션에 대해 연관규칙 탐사를 수행한 경우, {OS, PL}->{DB}에 대한 지지도(Support)와 신뢰도(Confidence)는 각각 얼마인가?

TID	OS	PL	CA	DB
1	1	0	0	1
2	0	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	0
6	1	1	0	1

① 지지도 0.67, 신뢰도 0.5

② 지지도 0.75, 신뢰도 0.5

③ 지지도 0.5, 신뢰도 0.75

④ 지지도 0.5, 신뢰도 0.67

정답 3번

간단한 해설

OS·PL·DB를 모두 수강한 트랜잭션은 3개이고, OS·PL을 함께 수강한 트랜잭션은 4개이다. **지지도는 전체 6개를 기준**으로 하여 $\frac{3}{6} = 0.5$, **신뢰도는 선행항을 만족한 4개를 기준**으로 하여 $\frac{3}{4} = 0.75$이다.

NoteBase

계산 과정

표에서 1은 해당 과목을 수강함을 뜻한다. 전체 트랜잭션은 6개이며, 선행항인 OS와 PL을 모두 수강한 것은 TID 3·4·5·6의 4개이다. 그중 DB까지 수강한 것은 TID 3·4·6의 3개이다. CA의 수강 여부는 이 규칙의 조건에 포함되지 않는다.

지지도는 **전체 트랜잭션 중 선행항과 후행항을 모두 포함하는 비율**이다. 따라서 $\text{Support} = \frac{3}{6} = 0.5$이다.

신뢰도는 **선행항을 만족한 트랜잭션 중 후행항도 만족하는 비율**이다. 따라서 $\text{Confidence} = \frac{3}{4} = 0.75$이다. 지지도와 분자는 같지만 분모는 전체 6개가 아니라 OS·PL을 함께 수강한 4개이다.

계산 결과는 지지도 0.5, 신뢰도 0.75이므로 ③과 일치한다.

NoteBase