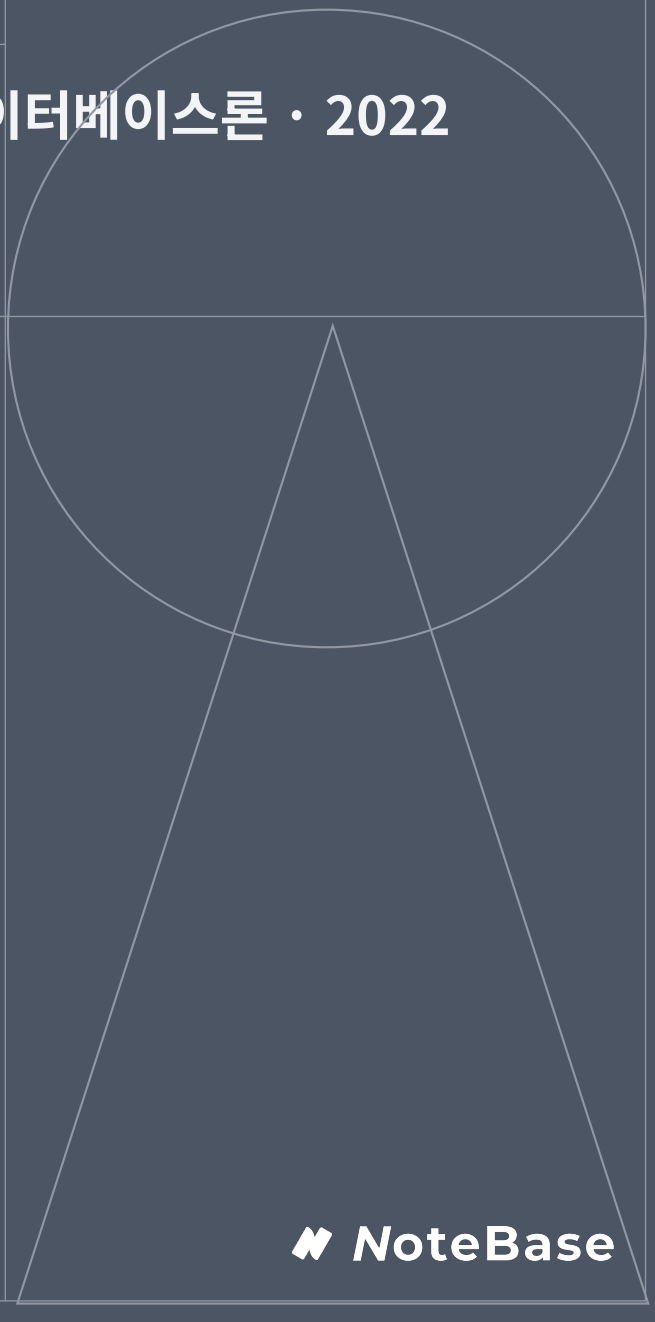
기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2022



NoteBase

기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2022

공무원/경찰/경력채용 · 데이터베이스론 · 2022

문 1. 3단계 스키마(3-level schema) 사이의 사상(mapping)에 필요한 정보를 컴파일하여 유지 관리하는 역할로 가장 적절한 것은?

① 질의어 처리기(query processor)

② 최적화기(optimizer)

③ 트랜잭션 관리자(transaction manager)

④ 데이터 사전(data dictionary)

정답 4번

간단한 해설

3단계 스키마의 구조와 상호 대응 관계는 메타데이터이다. 이를 유지·관리하는 **데이터 사전은 스키마 및 사상 정보의 저장소**이다.

개념 구분

외부·개념·내부 스키마 사이의 사상에는 각 단계의 구조와 대응 관계를 설명하는 메타데이터가 필요하다.
이러한 **스키마 정의와 사상 정보를 저장·유지하는 곳은 데이터 사전**이므로 ④가 해당한다. 정의를 분석·컴파일하는 처리와 그 결과를 보관하는 저장소는 구별해야 하며, 문제는 사상 정보의 유지·관리에 초점을 둔다.

질의어 처리기는 질의를 해석하고 실행하는 처리를 담당하고, 최적화기는 후보 실행 방법 중 효율적인 계획을 선택한다. 트랜잭션 관리자는 동시 실행과 장애 상황에서도 트랜잭션의 올바른 처리를 지원한다. 이들은 **질의·실행·트랜잭션 처리 기능**이며, 스키마와 사상 메타데이터를 보관하는 데이터 사전과 역할이 다르다.

NoteBase

문 2. 다음과 같은 ER 다이어그램의 '참여' 관계(relationship)에 관하여 <보기>의 내용을 가장 적절하게 짝지은 것은?



<보기>

㉠ 1개의 협력업체가 참여하는 사업의 수에는 제한이 없다.

㉡ 1개의 협력업체가 참여하는 사업의 수는 최대 1개로 제한된다.

㉢ 1개의 사업에 참여하는 협력업체의 수에는 제한이 없다.

㉣ 1개의 사업에 참여하는 협력업체의 수는 최대 1개로 제한된다.

① ㉠㉢

② ㉠㉣

③ ㉡㉢

④ ㉡㉣

정답 2번

간단한 해설

협력업체와 사업은 **1:n 관계**이므로, 협력업체 하나는 여러 사업에 참여할 수 있지만 사업 하나에 연결되는 협력업체는 최대 한 개이다. 양쪽의 수를 거꾸로 해석하지 않는 것이 핵심이다.

NoteBase

보기별 해설

㉠ O: 협력업체 쪽이 1, 사업 쪽이 n인 **일대다 관계**이다. 하나의 협력업체는 여러 사업과 연결될 수 있다. 수에 제한이 없다는 것은 이 그림에서 특정한 최대 사업 수를 정하지 않았다는 뜻이지, 실제 사업이 무한히 존재한다는 뜻은 아니다.

㉡ X: 협력업체 하나에 연결될 수 있는 사업은 n쪽이므로 최대 한 개로 제한되지 않는다. 협력업체 하나가 여러 사업에 참여할 수 있다고 고쳐야 한다.

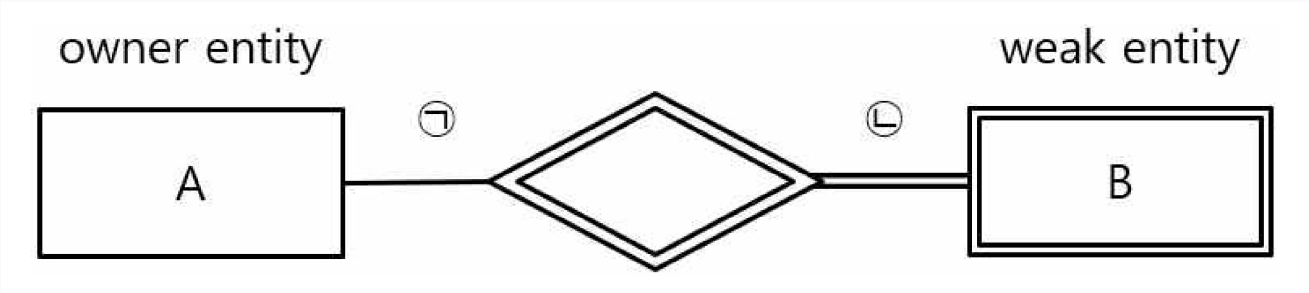
㉢ X: 사업 하나에서 반대편 협력업체 쪽을 보면 1이다. 따라서 하나의 사업에 연결되는 협력업체는 최대 한 개이며, 여러 협력업체가 같은 사업에 참여하는 다대다 관계가 아니다.

㉣ O: 협력업체 쪽의 1은 사업 하나에 연결될 수 있는 협력업체가 **최대 한 개**임을 나타낸다. 최대 개수 제약과 반드시 참여해야 한다는 최소 참여 제약은 구별해야 한다.

따라서 옳은 보기는 ㉠과 ㉣이며, 이에 해당하는 선택지는 ②이다.

NoteBase

문 3. 다음과 같은 ER 다이어그램의 약한 개체(weak entity) B에 관한 설명으로 가장 적절하지 않은 것은?



① 소유자 개체(owner entity) A와 일반화 관계(generalization relationship)를 통해 연결된다.

② 소유자 개체의 키를 약한 개체의 키에 포함시켜야 한다.

③ ㉠과 ㉡에는 차례대로 1과 n이 들어갈 수 있다.

④ B는 A에 존재 종속된다.

정답 1번

간단한 해설

소유자 A와 약한 개체 B를 연결한 이중 마름모는 **식별 관계**를 뜻한다. 상위·하위 개체를 묶는 일반화 관계와는 다르다.

선지별 해설

① X: 그림의 이중 마름모는 약한 개체 B와 소유자 A를 연결하는 **식별 관계**를 나타낸다. 일반화는 여러 하위 개체의 공통 특성을 상위 개체로 묶는 IS-A 관계이므로, 소유자와 약한 개체를 식별하는 이 관계와 다르다.

② O: B는 자신의 속성만으로 전체 개체를 유일하게 식별할 수 없다. 따라서 관계형 스키마로 옮길 때 **A의 키와 B의 부분키를 결합**하여 B의 키를 구성한다. 부분키는 같은 소유자에 속한 B들을 구별하는 값이다.

③ O: A 하나에 여러 B가 속하고 각 B는 하나의 소유자 A에 연결될 수 있으므로, A 쪽 ㉠에 1, B 쪽 ㉡에 n을 넣은 **1:n 식별 관계**가 가능하다. 개체 수 자체를 뜻하는 것이 아니라 두 개체 집합 사이의 대응 비율이다.

④ O: 약한 개체 B는 소유자 A 없이는 독립적으로 존재할 수 없는 **존재 종속성**을 가진다. 그림에서 B의 이중 사각형은 약한 개체를, 식별 관계와 B 사이의 이중선은 B의 전체 참여를 나타낸다.

NoteBase

문 4. 다음과 같은 방문접수대장 릴레이션(relation)에 관한 설명으로 가장 적절하지 않은 것은?

방문접수대장

방문자	방문부서	담당자	방문목적
홍길동	교통계	최일구	민원
박문수	외사계	오이팔	회의참석
박문수	형사계	권삼칠	면담
이순신	형사계	권삼칠	면담
최치원	교통계	최일구	민원

① 이순신은 박문수의 방문 후에 방문하였다.

② 박문수가 면담한 권삼칠과 이순신이 면담한 권삼칠이 동일 인물인지는 알 수 없다.

③ 방문자 속성과 담당자 속성은 동일 도메인을 가질 수 있다.

④ 차수(degree)는 4이다.

정답 1번

간단한 해설

릴레이션의 튜플에는 **논리적인 순서가 없으므로** 표에 표시된 위아래 순서로 방문 시각을 판단할 수 없다.
시간이나 순번을 나타내는 별도 속성이 필요하다.

NoteBase

선지별 해설

① X: 릴레이션의 튜플에는 **논리적인 순서가 없다**. 표에서 이순신 행이 박문수 행보다 아래에 표시되어 있다는 사실만으로 실제 방문 시각의 선후를 판단할 수 없다. 방문일시나 순번처럼 순서를 나타내는 별도 속성도 제시되어 있지 않다.

② O: 두 행의 담당자 이름이 모두 권삼칠이지만, **이름이 같다는 사실만으로 동일 인물임을 확정할 수 없다**. 담당자 식별번호나 이름의 유일성 제약이 없으므로 동명이인 가능성을 배제할 수 없다. 이는 두 사람이 반드시 다른 인물이라는 뜻도 아니다.

③ O: 속성 이름과 역할이 달라도 허용되는 값의 집합인 도메인은 같을 수 있다. 방문자와 담당자는 모두 사람의 이름을 값으로 가지는 도메인을 공유할 수 있으므로 적절하다.

④ O: 차수는 튜플 수가 아니라 **속성의 수**이다. 방문자, 방문부서, 담당자, 방문목적의 네 속성이 있으므로 차수는 4이다. 제시된 튜플 수인 카디널리티는 5이다.

따라서 표시된 행 순서를 실제 방문 순서로 해석한 ①이 가장 적절하지 않다.

NoteBase

문 5. 관계 연산식의 동등 관계가 성립하지 않는 것은? (단, R, S, T는 릴레이션이며 $\bowtie_N$, $\ltimes$는 자연 조인 연산자와 세미 조인 연산자를 의미함)

① $R \cap S = R - (R - S)$

② $(R \cup S) \cap T = R \cup (S \cap T)$

③ $R \bowtie_N S = (R \ltimes S) \bowtie_N S$

④ _조건1(_조건2(R)) = _조건2(_조건1(R))

정답 2번

간단한 해설

교집합을 합집합에 분배할 때는 **R과 S 양쪽 모두에 T와의 교집합**을 적용해야 한다. ②의 우변에는 R∩T가 아니라 R이 그대로 남아 동등 관계가 성립하지 않는다.

선지별 해설

① O: R−S는 R에는 있지만 S에는 없는 튜플이다. 이를 R에서 다시 빼면 **R과 S에 모두 속하는 튜플**만 남으므로 R∩S와 같다. 집합 연산은 합병 가능한 릴레이션에 적용한다.

② X: 올바른 분배 법칙은 $(R \cup S) \cap T = (R \cap T) \cup (S \cap T)$이다. 제시된 우변은 R에 속하지만 T에는 없는 튜플까지 포함하므로 동등하지 않다. 예를 들어 같은 스키마에서 R={a}, S=∅, T=∅이면 좌변은 ∅이지만 우변은 {a}이다.

③ O: R⋉S는 R의 튜플 중 S와 자연 조인할 상대가 있는 튜플만 남기는 세미 조인이다. 미리 제거된 튜플은 원래 자연 조인에서도 결과를 만들지 못하므로, **세미 조인 결과를 다시 S와 자연 조인해도 $R \bowtie_N S$와 같다.**

④ O: 조건2로 선택한 뒤 조건1로 선택하든 그 순서를 바꾸든 **조건1과 조건2를 모두 만족하는 튜플**만 남는다. 따라서 두 선택 연산의 적용 순서를 바꾸어도 결과는 같다.

NoteBase

문 6. 다음과 같은 성적 테이블에 필요한 DBMS 연산을 정의(definition) 연산과 조작(manipulation) 연산으로 구별할 때 가장 적절한 것은?

성적

학번	외국어	수학	경제학
S100	98	90	95
S200	88	85	90
S300	65	60	70
S400	95	95	95

① 학번이 S200인 학생의 수학 성적을 변경하기 위해서는 정의 연산이 필요하다.

② 학번이 S500인 학생의 성적을 추가하기 위해서는 정의 연산이 필요하다.

③ 학생들의 체육 성적을 추가하기 위해서는 정의 연산이 필요하다.

④ 성적 테이블을 생성하기 위해서는 조작 연산이 필요하다.

정답 3번

간단한 해설

테이블 구조를 만들거나 바꾸면 정의 연산, 기존 구조 안에서 행이나 값을 다루면 조작 연산이다. 주어진 성적 테이블에 없는 체육 과목을 새 열로 추가하려면 구조 변경이 필요하다.

선지별 해설

① X: S200 행의 수학 성적 85를 다른 값으로 바꾸는 것은 기존 데이터의 수정이다. 테이블 구조는 그대로이므로 **UPDATE에 해당하는 조작 연산**이 필요하다.

② X: S500 학생의 성적을 추가하는 것은 기존의 학번·외국어·수학·경제학 열에 새 행을 넣는 작업이다. 새로운 속성을 정의하는 것이 아니므로 INSERT에 해당하는 조작 연산이 필요하다.

③ O: 제시된 테이블은 과목별 성적을 각각 열로 저장하며, 현재 체육 열이 없다. 같은 구조로 체육 성적도 저장하려면 먼저 **체육이라는 새 열을 추가하는 정의 연산**이 필요하다. SQL에서는 ALTER TABLE로 열을 추가할 수 있고, 이후 실제 점수를 입력·수정하는 작업은 조작 연산이다.

④ X: 성적 테이블 자체를 만드는 것은 열 이름·자료형 등 저장 구조를 정하는 일이다. CREATE TABLE에 해당하는 정의 연산이며, 이미 만들어진 테이블의 데이터를 다루는 조작 연산과 구별된다.

NoteBase

문 7. 다음과 같은 사원 테이블에 <보기>와 같은 SQL 문을 수행하면 결과의 카디널리티(cardinality)는 얼마인가?

사원

사원번호	입사년도	부서	호봉
E100	2008	기획실	9
E200	2010	창원공장	7
E300	2017	영업1팀	5
E400	2010	판교연구소	8
E500	2011	홍보실	6
E600	2017	창원공장	4
E700	2018	판교연구소	3
E800	2019	창원공장	3
E900	2013	판교연구소	6
E1000	2019	판교연구소	3

<보기>

```
SELECT 호봉
FROM 사원
WHERE 입사년도 >= 2015 AND 부서 IN
( SELECT 부서 FROM 사원 WHERE 입사년도 = 2010 );
```

① 2

② 3

③ 4

④ 5

정답 3번

간단한 해설

안쪽 질의가 찾는 부서는 창원공장과 판교연구소이다. 이 두 부서에서 2015년 이후 입사한 사원은 4명이며, **DISTINCT가 없으므로 같은 호봉도 중복을 유지**하여 결과는 4행이다.

NoteBase

계산 과정

카디널리티는 질의 결과의 행 수이다. 먼저 안쪽 질의의 입사년도 = 2010을 만족하는 사원은 E200과 E400이므로, 부서 목록은 창원공장·판교연구소이다. 바깥 질의의 IN은 사원의 부서가 이 목록 중 하나와 일치하는지 검사한다.

바깥 질의에서는 부서 조건과 입사년도 >= 2015를 **동시에** 만족해야 한다. 해당 사원은 E600(2017, 창원공장, 호봉 4), E700(2018, 판교연구소, 호봉 3), E800(2019, 창원공장, 호봉 3), E1000(2019, 판교연구소, 호봉 3)의 4명이다. E300은 입사년도 조건은 만족하지만 영업1팀이므로 제외된다. E200·E400·E900은 부서는 맞지만 입사년도 조건을 만족하지 않는다.

SELECT 호봉에는 DISTINCT가 없으므로 결과에는 호봉 4인 행 1개와 호봉 3인 행 3개가 남는다. 서로 다른 호봉 값의 수인 2가 아니라, **중복을 포함한 행 수 1 + 3 = 4**를 센다. 따라서 결과의 카디널리티는 4이며 ③과 일치한다.

NoteBase

문 8. A회사에서 사원(사번, 이름, 소속, 나이) 테이블로부터 영업 사원 뷰(view)를 <보기>와 같이 정의하였다. 영업사원 뷰에 관한 설명으로 가장 적절하지 않은 것은? (단, 밑줄 친 애트리뷰트는 기본키임)

<보기>

```
CREATE VIEW 영업사원(사번, 이름, 소속, 나이)
AS SELECT * FROM 사원
WHERE 소속 = '영업부'
WITH CHECK OPTION;
```

① SELECT 문을 사용하여 A회사 '영업부' 사원의 수를 검색할 수 있다.

② UPDATE 문을 사용하여 홍길동 영업사원의 소속을 '영업부'에서 '홍보부'로 변경할 수 있다.

③ '영업부' 사원 중에 40세 이상의 사원들만을 포함하는 선임영업사원 뷰를 다시 정의할 수 있다.

④ 영업부 부장에게 영업사원 뷰에 대한 검색 권한을 부여하고 사원 테이블 자체에 대한 권한은 부여하지 않음으로써 영업부 사원에 대한 정보만 접근 가능하도록 정보 접근을 제한할 수 있다.

정답 2번

간단한 해설

WITH CHECK OPTION 때문에 영업사원 뷰를 통한 갱신 후에도 소속이 영업부여야 한다. 홍보부로 변경하면 뷰의 조건을 위반하므로 거부된다.

NoteBase

선지별 해설

① O: 영업사원 뷰에는 소속이 영업부인 사원만 나타난다. SELECT COUNT(*) FROM 영업사원;과 같이 조회하면 영업부 사원의 수를 구할 수 있다.

② X: **WITH CHECK OPTION**은 이 뷰를 통해 삽입·갱신한 행이 뷰의 조건을 계속 만족하도록 검사한다. 영업사원 뷰를 대상으로 홍길동의 소속을 홍보부로 바꾸면 소속 = 영업부 조건을 위반하므로 허용되지 않는다. 이는 권한을 가진 사용자가 기본 테이블을 직접 갱신하는 경우까지 이 뷰 옵션이 금지한다는 뜻은 아니다.

③ O: 기존 뷰를 바탕으로 **나이 >= 40** 조건을 추가한 새 뷰를 정의할 수 있다. 예를 들어 선임영업사원을 SELECT * FROM 영업사원 WHERE 나이 >= 40으로 정의하면 영업부이면서 40세 이상인 사원만 포함한다.

④ O: 기본 테이블의 직접 조회 권한 대신 필요한 행만 보이는 뷰의 조회 권한을 부여하여 **접근 범위를 제한**할 수 있다. 이는 뷰 소유자 권한으로 기본 테이블을 참조하는 일반적인 뷰 권한 모델에 따른 활용이며, 실제 보안 설정에서는 다른 권한 경로나 DBMS별 보안 옵션도 함께 점검한다.

따라서 제시된 영업사원 뷰를 통한 갱신으로 허용되지 않는 것은 ②이다.

NoteBase

문 9. 다음과 같은 거래장부 테이블이 속하는 가장 높은 수준의 정규형으로 적절한 것은? (단, 현재 테이블의 값에 기반해서 함수 종속성을 판단한다고 가정하며, 기본키는 (거래번호, 상품코드)임)

거래장부

거래번호	고객번호	거래일자	상품코드	단가	수량	고객주소
101011	C3456	220101	G121	1000	3	세종 34-5
101011	C3456	220101	G178	3500	2	세종 34-5
101012	C8923	220101	G121	1000	1	용인 13-4
101013	C2877	220102	G121	1000	2	천안 32-1
101013	C2877	220102	G445	3500	1	천안 32-1
101014	C2877	220102	G121	1000	3	천안 32-1
101015	C8923	220103	G178	3500	3	용인 13-4

① 제1정규형

② 제2정규형

③ 제3정규형

④ BCNF

정답 1번

간단한 해설

각 속성값은 원자값이지만, 거래번호만으로 고객번호·거래일자가 결정되고 상품코드만으로 단가가 결정된다.
복합 기본키에 대한 부분 함수 종속이 있으므로 제2정규형이 아닌 제1정규형까지 만족한다.

NoteBase

핵심 풀이

각 셀에는 하나의 값이 들어 있고 반복되는 값 목록이 한 셀에 묶여 있지 않으므로 **제1정규형의 원자값 조건**을 만족한다. 고객주소가 지역명과 번지로 표현되어 있다는 사실만으로 원자값 조건을 위반하는 것은 아니다. 주소를 하나의 속성값으로 취급하는 도메인에서 판단한다.

기본키는 (거래번호, 상품코드)이다. 그런데 거래번호 101011의 두 행은 고객번호와 거래일자가 같고, 거래번호 101013의 두 행도 각각 동일하다. 주어진 값에서 거래번호만으로 고객번호·거래일자 등을 결정할 수 있다. 또한 상품코드 G121은 모두 단가 1000, G178은 모두 단가 3500이므로 상품코드만으로 단가가 결정된다. 이는 **복합키의 일부에 대한 부분 함수 종속**이므로 제2정규형을 만족하지 않는다. 따라서 더 높은 제3정규형이나 BCNF에도 해당하지 않으며, 가장 높은 수준은 제1정규형인 ①이다.

NoteBase

문 10. 다음 중 함수 종속성(functional dependency)에 관한 설명으로 가장 적절하지 않은 것은?

① 함수 종속성은 다치 종속성(multi-valued dependency)의 일종이다.

② 릴레이션 R(X, Y)에서 X가 후보키이면 함수 종속성 X → Y가 성립한다.

③ 릴레이션 R(A, B, C)에 함수 종속성 A → B가 존재하고 B → C가 존재하면 A → C도 동시에 존재한다.

④ 릴레이션 R(A, B, C)에 함수 종속성 A → B가 존재하면 2개 테이블 R1(A, B)와 R2(B, C)로 무손실 분해(nonloss decomposition)할 수 있다.

정답 4번

간단한 해설

④의 분해에서는 공통 속성 B가 어느 한쪽 테이블을 함수적으로 결정한다는 보장이 없어 **조인 시 가짜 튜플**이 생길 수 있다. A→B를 이용하려면 (A,B)와 (A,C)로 분해하여 공통 속성 A가 (A,B)를 결정하도록 해야 한다.

선지별 해설

① O: 함수 종속성 X→Y가 성립하면 같은 X에 대해 Y가 하나로 정해지므로 다치 종속성 X↠Y도 성립한다. 따라서 **함수 종속성은 다치 종속성의 특수한 경우**이다. 다치 종속성이 있다고 항상 함수 종속성이 성립하는 것은 아니다.

② O: 후보키는 튜플을 유일하게 식별하는 최소 속성 집합이므로 릴레이션의 모든 속성을 함수적으로 결정한다. 따라서 X가 후보키이면 **X→Y**가 성립한다.

③ O: 함수 종속성의 **이행 규칙**에 따라 A→B와 B→C가 성립하면 A→C도 성립한다. 같은 A를 가진 튜플은 B가 같고, 그 B에 의해 C도 같아야 하기 때문이다.

④ X: 두 테이블의 공통 속성은 B이다. 함수 종속성에 의한 이항 무손실 분해가 보장되려면 **B가 분해된 테이블 중 적어도 하나의 모든 속성을 결정**해야 하는데 A→B만으로는 B→A나 B→C가 보장되지 않는다. 예를 들어 R={(a1,b,c1),(a2,b,c2)}는 A→B를 만족하지만, R1(A,B)와 R2(B,C)를 자연 조인하면 (a1,b,c2), (a2,b,c1)도 생긴다. 따라서 제시한 분해는 무손실이라고 할 수 없으며, A→B를 이용한 무손실 분해는 R1(A,B)와 R2(A,C)이다.

NoteBase

문 11. 관계 데이터베이스 시스템에서 무결성 규정을 명세하는 방법으로 가장 적절하지 않은 것은?

① 인덱스(index) 생성

② 도메인(domain) 제약조건 정의

③ 트리거(trigger) 생성

④ 주장(assertion) 생성

정답 1번

간단한 해설

무결성 규정은 데이터가 만족해야 할 조건을 정하고, 일반 인덱스는 주로 데이터 접근을 빠르게 한다. 조건의 명세와 검색을 위한 구조의 생성을 구별해야 한다.

개념 구분

인덱스는 검색할 데이터를 빠르게 찾기 위한 **접근 구조**이다. 일반적인 인덱스 생성 자체는 어떤 데이터가 유효한지를 정하는 무결성 규정의 명세가 아니므로, 이 구분에 해당하는 것은 ①이다. 다만 UNIQUE 인덱스는 중복을 막아 유일성을 강제할 수 있으므로, 인덱스가 무결성과 전혀 관계없다고 일반화해서는 안 된다.

도메인 제약조건은 속성에 허용되는 자료형·값의 범위 등을 정한다.

트리거는 특정 데이터 변경 사건이 발생할 때 검사를 수행하고 부적절한 변경을 거부하는 등의 방식으로 무결성을 유지할 수 있다.

주장은 데이터베이스가 항상 만족해야 하는 조건을 명세하는 방법이며, SQL 표준의 CREATE ASSERTION이 이에 해당한다. 실제 DBMS의 주장 지원 여부는 제품에 따라 다르다.

NoteBase

문 12. 비용 기반(cost-based) 질의 최적화(query optimization) 방법에 관한 설명으로 가장 적절하지 않은 것은?

① 각 질의 실행 전략에 대해 비용을 예측한다.

② 모든 가능한 실행 전략을 대상으로 가장 효율적인 전략의 선택을 보장한다.

③ 비용 예측을 위해 DBMS의 시스템 카탈로그(catalog)를 사용한다.

④ 대용량 데이터베이스에서는 디스크 접근 비용 최소화가 매우 중요하다.

정답 2번

간단한 해설

비용 기반 최적화는 후보 실행 계획의 **추정 비용**을 비교해 낮은 비용의 계획을 선택한다. 탐색 범위와 추정 오차의 한계가 있으므로 모든 가능한 계획 중 실제 최적 계획의 선택을 보장하지는 않는다.

선지별 해설

① O: 비용 기반 최적화는 검토하는 후보 실행 계획마다 접근 경로·조인 순서·조인 방식 등에 따른 **예상 실행 비용을 산정하여 비교**한다. 모든 계획을 실제로 실행해 시간을 재는 방식은 아니다.

② X: 가능한 계획의 수가 매우 많아 탐색 범위를 제한하거나 휴리스틱을 사용할 수 있고, 비용 자체도 통계와 모델에 근거한 추정치이다. 따라서 **모든 가능한 계획을 탐색하여 실제로 가장 빠른 계획을 선택한다고 보장할 수 없다.** 목표는 합리적인 최적화 시간 안에 비용이 낮은 계획을 찾는 것이다.

③ O: 시스템 카탈로그에 저장된 **테이블 크기·튜플 수·인덱스 정보·값의 분포 등 통계와 메타데이터**를 이용한다. 이를 바탕으로 조건의 선택도와 중간 결과 크기를 추정하고 각 연산의 비용을 계산한다.

④ O: 디스크 기반 대용량 데이터베이스에서는 많은 페이지를 읽고 쓰는 I/O가 주요 비용이므로 **불필요한 디스크 접근을 줄이는 것**이 중요하다. 다만 비용 모델은 CPU·메모리·캐시 효과 등도 고려하며, 디스크 비용만이 항상 유일한 기준이라는 뜻은 아니다.

NoteBase

문 13. 트랜잭션 T1과 T2가 다음과 같은 시간 순으로 실행될 경우 발생하는 문제점을 방지할 수 있는 SQL 표준 트랜잭션 고립 수준을 <보기>에서 모두 고른 것은?

시간	트랜잭션 T1	트랜잭션 T2
↓	select avg(salary) from employee;	
		update employee set salary = salary * 1.05;
	select avg(salary) from employee;	

<보기>

㉠ READ UNCOMMITTED

㉡ READ COMMITTED

㉢ REPEATABLE READ

㉣ SERIALIZABLE

① ㉠㉡㉢㉣

② ㉡㉢㉣

③ ㉢㉣

④ ㉣

정답 3번

간단한 해설

T1의 두 평균 조회 사이에 T2가 기존 급여를 변경하면 반복 조회 결과가 달라질 수 있다. SQL 표준에서 **반복 불가능 읽기는 REPEATABLE READ 이상**에서 방지한다.

NoteBase

보기별 해설

㉠ (X) READ UNCOMMITTED는 다른 트랜잭션의 미커밋 변경도 읽을 수 있는 수준이다. T1이 급여 평균을 다시 읽을 때 T2의 변경에 영향을 받을 수 있으므로 **반복 읽기의 일관성을 보장하지 않는다**.

㉡ (X) READ COMMITTED는 미커밋 값을 읽는 더티 리드를 막지만, T1의 두 조회 사이에 T2의 급여 변경이 커밋되면 두 번째 조회에서 달라진 평균을 읽을 수 있다. 즉 **커밋된 변경으로 인한 반복 불가능 읽기**까지 방지하지는 않는다.

㉢ (O) REPEATABLE READ는 이미 읽은 데이터가 다른 트랜잭션의 변경 때문에 재조회 시 달라지는 **반복 불가능 읽기를 방지**한다. 기존 직원들의 급여를 변경하는 이 사례에서 T1의 반복 조회가 서로 다른 변경 전후 값을 읽는 문제를 막는다.

㉣ (O) SERIALIZABLE은 동시 실행의 결과가 트랜잭션을 순서대로 실행한 결과와 같도록 하는 수준이다. 반복 불가능 읽기를 포함한 동시 실행 이상을 방지하므로 이 문제에도 적용된다.

따라서 해당하는 수준은 **㉢과** ㉣이며 조합은 ③이다.

NoteBase

문 14. 데이터베이스 로그(log)에 관한 설명으로 가장 적절한 것은?

① 지연갱신 방법을 사용하는 경우 변경 연산을 위해 반드시 필요한 로그 레코드 정보는 <트랜잭션 식별자, 데이터항목, 변경이전값, 변경이후값>이다.

② 데이터베이스 버퍼의 블록을 디스크에 기록하는 것과 로그 버퍼의 블록을 로그 파일에 기록하는 것은 어떤 순서로 실행되어도 상관없다.

③ 백업된 시스템 로그에 기록되어 있는 모든 완료된 트랜잭션의 실행 결과는 데이터베이스를 UNDO 하는데 사용될 수 있다.

④ 로그 레코드 버퍼링은 여러 로그 레코드를 로그 버퍼에 한꺼번에 모아 저장하여 성능을 향상시킨다.

정답 4번

간단한 해설

로그 버퍼링은 여러 로그 레코드를 모아 기록하여 디스크 입출력 부담을 줄인다. 버퍼링을 하더라도 관련 로그를 데이터보다 먼저 영구 기록하는 WAL 원칙은 유지해야 한다.

선지별 해설

① X: 기본적인 지연갱신은 커밋 전의 변경을 데이터베이스에 반영하지 않으므로 장애 회복에서 미완료 변경을 UNDO할 필요가 없다. 변경 재현을 위한 **변경이후값**을 기록하는 REDO 로그가 핵심이며, 변경이전값까지 반드시 필요하다고 한 설명은 부적절하다.

② X: 선행 기록 로그(WAL) 원칙에서는 변경된 데이터 블록을 영구 저장장치에 기록하기 전에 **관련 로그를 먼저 영구 저장**해야 한다. 데이터만 기록되고 회복에 필요한 로그가 유실되는 상황을 막아야 하므로 기록 순서는 임의가 아니다.

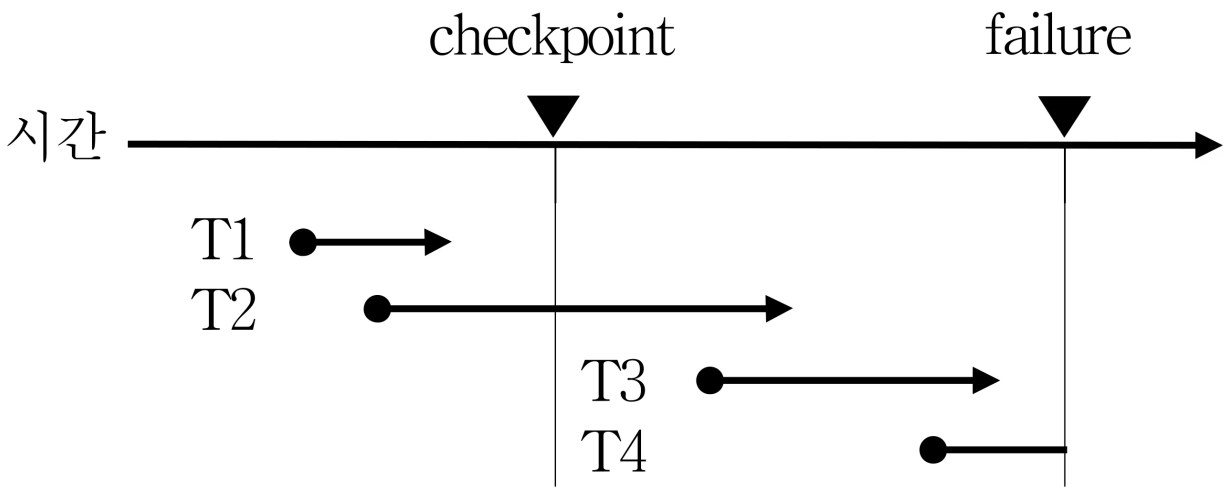
③ X: 일반적인 장애 회복에서 커밋이 완료된 트랜잭션의 변경은 필요한 경우 **REDO로 재현**하여 보존한다. UNDO는 커밋되지 않은 변경을 취소하는 데 사용되므로, 완료된 트랜잭션의 결과를 모두 UNDO에 사용한다는 설명은 회복 목적을 잘못 연결한 것이다.

④ O: 로그 레코드를 생성할 때마다 각각 디스크에 쓰기보다 **로그 버퍼에 모아 묶어서 기록**하면 입출력 횟수와 비용을 줄일 수 있다. 다만 버퍼링하더라도 WAL의 선행 기록 순서와 커밋의 지속성을 위한 로그 기록 요건은 지켜야 한다.

따라서 로그 버퍼링을 통한 성능 향상을 설명한 ④가 가장 적절하다.

NoteBase

문 15. 다음과 같이 검사점(checkpoint) 이후 시스템 장애(failure)가 발생하였다. 지연갱신 또는 즉시갱신에 따라 UNDO와 REDO되는 트랜잭션들로 가장 적절한 것은?



① 기준: 지연갱신, UNDO: 없음, REDO: T2, T3
기준: 즉시갱신, UNDO: T4, REDO: T2, T3

② 기준: 지연갱신, UNDO: 없음, REDO: T3
기준: 즉시갱신, UNDO: T4, REDO: T3

③ 기준: 지연갱신, UNDO: T4, REDO: T3
기준: 즉시갱신, UNDO: 없음, REDO: T3

④ 기준: 지연갱신, UNDO: T4, REDO: T2, T3
기준: 즉시갱신, UNDO: 없음, REDO: T2, T3

정답 1번

간단한 해설

검사점 이후 완료된 트랜잭션은 REDO 대상으로, 장애 때 미완료인 트랜잭션은 갱신 방식에 따라 UNDO 여부를 판단한다. **지연갱신은 UNDO가 없고, 즉시갱신은 미완료 변경을 되돌린다.**

NoteBase

도식 해설

그림에서 T1은 검사점 전에 완료되었으므로 이 기본 검사점 회복 모형에서는 다시 처리하지 않는다. T2는 검사점 전에 시작했지만 검사점 뒤에 완료되었고, T3도 검사점 뒤에 완료되었다. 따라서 **REDO 대상은 T2와 T3**이다. 시작 시점만 보고 T2를 제외하면 안 된다.

지연갱신은 커밋 전의 변경을 데이터베이스에 반영하지 않으므로, 장애 당시 미완료인 T4에 대해 되돌릴 데이터베이스 변경이 없다. 따라서 UNDO 대상은 없고, 완료된 T2와 T3를 REDO한다.

즉시갱신은 미완료 트랜잭션의 변경도 데이터베이스에 반영될 수 있으므로, 장애 당시 미완료인 **T4를 UNDO**한다. 완료된 T2와 T3는 REDO 대상이다.

두 방식을 이렇게 짝지은 선택지는 ①이다.

NoteBase

문 16. 로킹(locking) 기법에 관한 설명으로 가장 적절하지 않은 것은?

① 디스크 블록 단위보다는 레코드 단위에서 로킹 오버헤드가 크다.

② 로킹 기법을 사용하여 트랜잭션의 동시성을 제어하더라도 항상 직렬가능(serializable) 스케쥴을 보장하지는 않는다.

③ COMMIT 명령문은 로킹으로 발생한 교착상태를 해지시키는데 사용된다.

④ 2단계 로킹 규약에서 트랜잭션 T가 완료되거나 철회될 때까지 T가 보유한 배타 로크(exclusive lock)를 해제하지 않으면 연쇄적 롤백(cascading rollback)을 방지할 수 있다.

정답 3번

간단한 해설

교착상태는 보통 **희생 트랜잭션의 중단·롤백**으로 잠금을 해제하여 해결한다. COMMIT은 정상적인 변경 확정 명령이므로 이를 교착상태 해소 명령으로 설명한 ③이 틀리다.

선지별 해설

① O: 같은 범위의 데이터를 처리할 때 레코드 단위 로킹은 블록 단위보다 더 많은 잠금을 획득·관리해야 하므로 **잠금 관리 오버헤드가 일반적으로 크다.** 대신 서로 다른 레코드에 대한 접근을 허용하여 동시성을 높일 수 있다.

② O: 잠금을 사용한다는 사실만으로 직렬 가능성이 보장되지는 않는다. 해제 후 다시 잠금을 얻는 등 규칙 없이 사용하면 비직렬가능 스케줄이 생길 수 있다. **2단계 로킹처럼 적절한 규약**을 모든 트랜잭션이 따르면 충돌 직렬 가능성이 보장된다.

③ X: COMMIT은 트랜잭션의 변경을 확정하는 명령이지 교착상태 해소 명령이 아니다. 교착상태에서는 서로 상대방의 잠금 해제를 기다려 진행하지 못하므로, 보통 **희생 트랜잭션을 중단·롤백하여 잠금을 해제**하고 순환 대기를 끊는다. 정상적인 COMMIT이 잠금을 해제한다는 사실과 교착상태 복구 방법을 혼동하면 안 된다.

④ O: 배타 잠금을 커밋 또는 철회 시점까지 유지하는 것은 **엄격한 2단계 로킹**이다. 다른 트랜잭션이 미확정 변경 값을 읽지 못하게 하므로, 앞선 트랜잭션의 철회 때문에 그 값을 읽은 트랜잭션까지 취소해야 하는 연쇄적 롤백을 방지한다. 다만 교착상태까지 자동으로 방지하는 것은 아니다.

NoteBase

문 17. 분산 데이터베이스 질의 처리에 관한 설명 중 가장 적절하지 않은 것은?

① 분산 질의 처리의 목표는 데이터 전송량 최소화를 포함한다.

② 한 질의를 원격 사이트에서 실행 가능하도록 분해하고, 결과를 결합한 후 질의의 최종 결과를 만드는 전략을 고려한다.

③ 분산 조인 연산의 처리를 위해서는 전송될 데이터의 분량과 각 사이트의 상대적 처리속도를 고려한다.

④ 세미 조인 연산의 활용은 원격 사이트에 있는 릴레이션에서 조인에 참여하는 튜플의 수가 많을수록 효과적이다.

정답 4번

간단한 해설

세미 조인은 원격 데이터를 먼저 걸러 **조인에 필요한 튜플만 전송**하도록 돕는다. 따라서 다른 조건이 같을 때 조인에 참여하지 않는 튜플을 많이 제거할 수 있을수록 통신량 절감에 유리하다.

선지별 해설

① O: 분산 환경에서는 사이트 사이의 데이터 이동에 통신 비용이 발생한다. 따라서 불필요한 튜플이나 속성을 보내지 않도록 **전송량을 줄이는 것**이 질의 처리의 중요한 목표에 포함된다. 다만 전체 비용에는 각 사이트의 처리 비용도 포함된다.

② O: 전역 질의를 각 원격 사이트에서 수행할 수 있는 부분 작업으로 나누고, 필요한 중간 결과를 전달·결합하여 최종 결과를 얻는다. 어떤 연산을 어느 사이트에서 수행할지도 실행 전략의 일부이다.

③ O: 분산 조인에서는 전체 릴레이션을 옮길지, 일부만 줄여서 옮길지와 함께 조인을 수행할 사이트를 정해야 한다. 전송할 데이터의 크기뿐 아니라 사이트별 처리 성능도 총비용과 응답 시간에 영향을 준다.

④ X: 세미 조인은 조인에 필요한 속성 값을 먼저 보내 원격 릴레이션에서 **최종 조인에 참여하지 않을 튜플을 제거**하는 데 활용한다. 다른 조건이 같다면 참여하는 튜플의 비율이 작아 제거되는 양이 클수록 전송량 절감 효과가 커진다. 참여 튜플이 많을수록 효과적이라는 설명은 반대이며, 실제 이득은 조인 속성 값의 선행 전송 비용과 추가 처리 비용까지 비교해야 한다.

NoteBase

문 18. 데이터 웨어하우스(data warehouse)에 관한 설명으로 가장 적절한 것은?

① 스타 스키마(star schema)는 하나의 차원 테이블과 여러 개의 사실 테이블로 구성된 관계 데이터베이스 스키마이다.

② 롤업(roll-up)은 요약된 형태의 데이터 수준에서 더욱 상세한 정보를 탐색하는 연산이다.

③ ETL은 데이터를 추출(extract), 변환(transform), 적재(load)하는 업무를 의미한다.

④ 데이터 마트(data mart)는 데이터 웨어하우스에서 효과적으로 온라인 트랜잭션을 지원하기 위해 구축된 전사적 조직을 위한 저장소이다.

정답 3번

간단한 해설

ETL은 **데이터 추출·변환·적재**를 뜻한다. 원천 데이터를 분석에 적합하게 정리하여 데이터 웨어하우스 등의 목적 저장소에 넣는 과정이다.

선지별 해설

① X: 기본적인 스타 스키마는 **중심의 사실 테이블과 이를 둘러싼 여러 차원 테이블**로 구성된다. 사실 테이블은 매출액 같은 측정값과 차원 키를 담고, 차원 테이블은 상품·시간·지역 같은 분석 기준을 담는다. 선지는 사실과 차원의 위치·구성을 뒤바꾸었다.

② X: 롤업은 일별 데이터를 월별·연도별로 합산하는 것처럼 **더 높은 요약 수준**으로 집계하는 연산이다. 반대로 요약 수준에서 더 상세한 수준으로 내려가 탐색하는 것은 드릴다운이다.

③ O: ETL은 **추출(Extract) → 변환(Transform) → 적재(Load)**를 뜻한다. 원천 시스템에서 데이터를 가져오고, 형식을 맞추거나 정제·통합한 다음 목적 저장소에 적재하는 데이터 통합 과정이다.

④ X: 데이터 마트는 전사 전체가 아니라 **특정 부서나 업무 주제의 분석 요구**에 맞춘 데이터 저장소이다. 데이터 웨어하우스의 일부로 만들거나 다른 원천에서 구축할 수 있으며, 주된 목적은 온라인 트랜잭션 처리보다 분석과 의사결정 지원이다.

따라서 ETL의 의미를 올바르게 설명한 ③이 가장 적절하다.

NoteBase

문 19. NOSQL의 특징과 처리에 관한 설명으로 옳은 것을 <보기>에서 모두 고른 것은?

<보기>
㉠ 관계 데이터베이스와 같이 엄격한 스키마(schema)를 필요로 한다.
㉡ 관계 데이터베이스보다 높은 수준의 일관성(consistency)을 강조한다.
㉢ NOSQL 데이터베이스는 키-값 저장소, 문서기반 데이터베이스, 컬럼기반 데이터베이스, 그래프 데이터베이스 등으로 구분될 수 있다.
㉣ CAP 이론에서 정의한 분산 컴퓨팅 환경의 세가지 특징 중 가용성(availability)과 분할허용성(partition tolerance)을 강조한다.

① ㉠㉡
② ㉠㉢
③ ㉡㉢
④ ㉢㉣

정답 4번

간단한 해설

NoSQL은 유연한 데이터 모델과 분산 확장을 지향하며, 키-값·문서·와이드 컬럼·그래프 등의 모델로 나뉜다. **AP 지향 설계는 가용성과 분할허용성을 중시**하지만, 이를 모든 NoSQL에 적용되는 보장으로 이해해서는 안 된다.

NoteBase

보기별 해설

㉠ X: NoSQL은 관계형 테이블의 고정된 구조와 달리 **유연한 데이터 모델·스키마**를 활용하는 경우가 많다. 따라서 엄격한 스키마를 일률적으로 요구한다는 설명은 부적절하다. 다만 스키마가 유연하다는 것이 모든 제품에서 데이터 검증 규칙을 전혀 둘 수 없다는 뜻은 아니다.

㉡ X: 관계 데이터베이스보다 더 높은 일관성을 공통 목표로 삼는 것이 NoSQL의 특징은 아니다. 대규모 분산 처리와 확장성·가용성을 위해 일관성을 완화하거나 조정 가능한 일관성을 제공하는 설계도 있다. 반대로 강한 일관성을 지원하는 NoSQL도 있으므로 NoSQL을 항상 약한 일관성으로 단정해서도 안 된다.

㉢ O: 대표적인 데이터 모델은 키로 값을 조회하는 키-값형, 문서 단위로 저장하는 문서형, 컬럼 패밀리 등을 이용하는 와이드 컬럼형, 정점과 간선으로 관계를 표현하는 그래프형으로 구분할 수 있다. 여기서 컬럼기반은 이러한 NoSQL 모델 분류의 문맥이다.

㉣ O(일반적인 AP·BASE 지향 설계의 취지): 분산 환경에서 서비스 응답을 계속 제공하려는 NoSQL 설계는 **가용성과 분할허용성**을 중시한다. Cassandra가 대표적인 예이다. 다만 원문은 이러한 설계 경향을 요약한 것으로 이해해야 하며, 모든 NoSQL이 AP를 선택한다는 보편 명제는 아니다. 네트워크 분할 시 일관성을 우선하는 CP 계열과 조정 가능한 일관성도 존재한다. CAP는 분할이 발생한 상황에서 강한 일관성과 가용성을 동시에 보장할 수 없다는 제약을 설명한다.

이러한 설계 경향의 의미와 한계를 전제로 고르면 ㉢·㉣이며, 조합은 ④와 일치한다.

NoteBase

문 20. 데이터베이스의 성능 튜닝에 관한 설명으로 옳지 않은 것을 <보기>에서 모두 고른 것은?

<보기>
㉠ 한 릴레이션에 관련된 모든 인덱스를 집중인덱스(clustered-index)로 구성한다.
㉡ 합계와 같은 집계 질의를 위해 실체화 뷰(materialized view)를 생성한다.
㉢ 릴레이션의 속성에 대한 실렉트(σ) 조건이 질의에서 자주 사용되는 경우에는 릴레이션을 속성 값에 따라 수직 분할한다.
㉣ explain 등의 명령을 사용하여 사용자가 질의에 사용 중인 수행계획을 파악한다.
㉤ 클라이언트-서버 시스템에서 빈번하게 반복 사용되는 작업에 대해 저장 프로시져(stored procedure)를 사용한다.
㉥ 데이터의 일관성을 체계적으로 유지하기 위해서는 실체화 뷰보다는 역정규화 릴레이션(denormalized relation)을 사용한다.

① ㉠㉡㉤
② ㉠㉢㉥
③ ㉡㉣㉤
④ ㉢㉣㉥

정답 2번

간단한 해설

집중 인덱스를 모두 중복 구성할 수 없고, 속성 값에 따른 행 분리는 수직이 아닌 **수평 분할**이다. 또한 역정규화 자체가 실체화 뷰보다 일관성 관리에 유리한 것은 아니므로 ㉠·㉢·㉥이 틀리다.

NoteBase

보기별 해설

㉠ X: 집중 인덱스는 데이터 행의 저장 순서와 연결되므로 하나의 테이블에 **여러 개의 서로 다른 집중 인덱스를 둘 수 없다.** 필요한 나머지 접근 경로는 비집중 인덱스 등으로 구성한다.

㉡ O: 실체화 뷰는 질의 결과를 미리 저장하므로 반복되는 합계·평균 등의 **집계 계산을 줄이는 데 활용**할 수 있다. 원본 변경에 따른 뷰 갱신 비용과 최신성 요구도 함께 고려해야 한다.

㉢ X: 속성 값에 대한 선택 조건에 따라 해당 행들을 나누는 것은 **수평 분할**이다. 수직 분할은 자주 함께 사용하는 열을 묶는 등 속성 집합을 기준으로 나누는 방법이므로 두 분할의 기준을 혼동하였다.

㉣ O: EXPLAIN 등의 명령으로 **접근 경로·조인 순서·연산 방식 등 실행 계획**을 확인하여 병목을 분석할 수 있다. 일반 EXPLAIN이 실제 실행 시간까지 반드시 측정하는 것은 아니며, 실제 실행 통계는 별도 옵션이나 도구를 사용한다.

㉤ O: 반복 작업을 서버의 저장 프로시저로 묶으면 **클라이언트와 서버 사이의 반복 통신**을 줄이고 처리 로직을 재사용할 수 있다. 실행 계획 재사용 등은 DBMS 구현에 따라 달라지므로 무조건적인 성능 향상을 뜻하지는 않는다.

㉥ X: 역정규화 릴레이션은 중복 값을 직접 관리해야 하므로 갱신 시 일관성 유지 부담이 커질 수 있다. 원본과의 대응 관계 및 갱신 체계를 DBMS가 관리하는 **실체화 뷰를 활용하는 편이 체계적인 유지에 유리**하다. 단, 실체화 뷰도 갱신 방식과 시점을 정해야 하며 언제나 자동으로 최신 상태가 되는 것은 아니다.

따라서 옳지 않은 설명은 ㉠·㉢·㉥이며 ②와 일치한다.

NoteBase