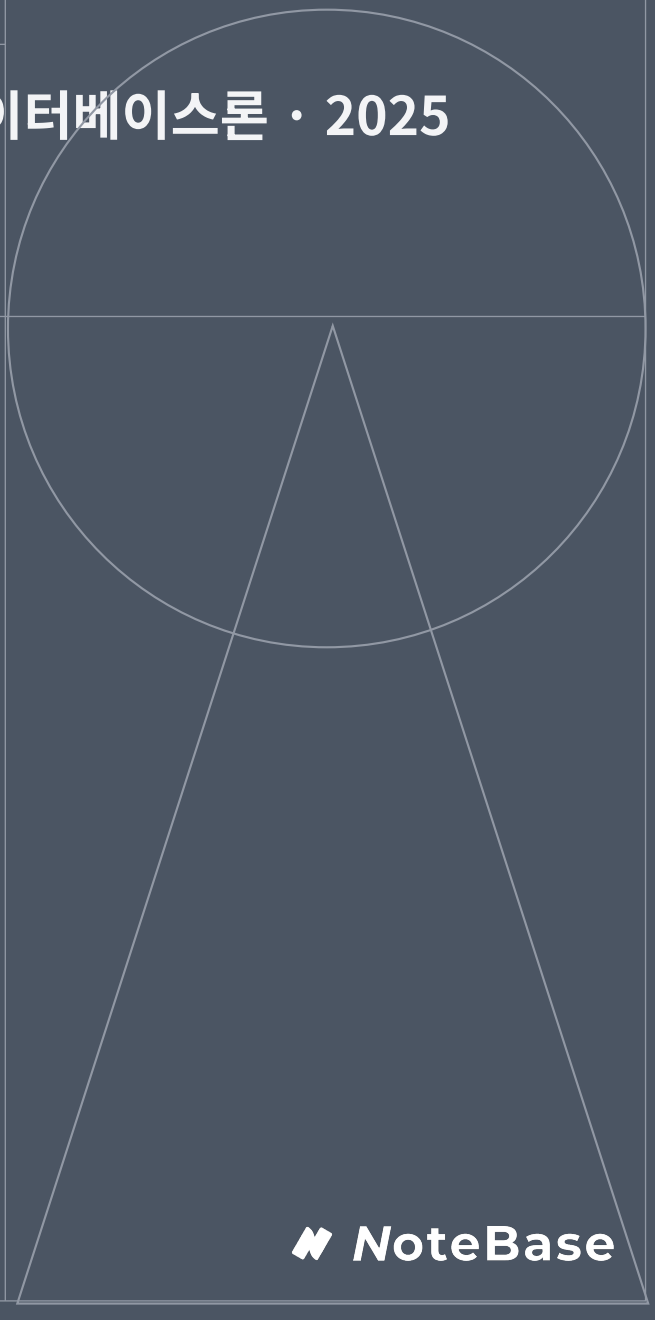
기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2025



NoteBase

기출OX 해설집

공무원/경찰/경력채용 · 데이터베이스론 · 2025

공무원/경찰/경력채용 · 데이터베이스론 · 2025

문 1. 데이터베이스 관리 시스템에 관한 설명으로 가장 적절하지 않은 것은?

① 데이터 종속성과 중복성 문제를 해결한다.

② 전체적으로 통제할 수 있는 프로그램들로 구성되어 응용 프로그램의 요청을 수행한다.

③ 데이터베이스가 접근 처리될 때마다 제어 기능을 통해 그 유효성을 검사함으로써 데이터의 정확성을 높인다.

④ 자료의 독립과 표준화로 인해 파일 시스템에 비해 초기 하드웨어 및 소프트웨어 구축 비용이 감소한다.

정답 4번

간단한 해설

데이터 독립성과 표준화는 DBMS의 장점이지만 **초기 하드웨어·소프트웨어 구축 비용 감소를 보장하지 않는다.** 도입·전환에 추가 비용이 발생할 수 있으므로 ④가 적절하지 않다.

선지별 해설

① O: DBMS는 데이터 구조와 저장 방식을 응용 프로그램에서 분리하고 데이터를 통합 관리하여 **종속성과 불필요한 중복을 줄이는 것**을 목표로 한다. 성능·가용성을 위한 의도적인 복제까지 모두 없앤다는 뜻은 아니다.

② O: DBMS는 데이터 정의·검색·갱신·접근 제어 등을 담당하는 **프로그램들의 집합**이다. 응용 프로그램의 요청을 해석하고 저장된 데이터에 대한 작업을 수행·관리한다.

③ O: 권한 검사와 무결성 제약 검사 등의 **제어 기능**을 통해 부적절한 접근이나 규칙을 위반하는 변경을 제한하여 정확성을 높인다. 이는 설정된 규칙에 따른 검사를 뜻하며, 모든 읽기마다 전체 데이터를 재검사하거나 현실의 사실 여부까지 자동으로 보장한다는 뜻은 아니다.

④ X: 데이터 독립성과 표준화가 유지보수를 돕더라도 **초기 구축 비용 감소가 자동으로 따라오는 것은 아니다.** DBMS 도입에는 요구 성능에 맞는 하드웨어, 소프트웨어, 데이터 전환·설정 등 추가 비용이 들 수 있다. 전통적인 파일 시스템과 비교한 일반적 단점으로 초기 도입 부담을 들며, 실제 비용은 제품과 구축 규모에 따라 달라진다.

NoteBase

문 2. 데이터베이스의 외부 스키마에 관한 설명으로 가장 적절한 것은?

① 내부 저장장치의 상세 구조를 명세한다.

② 하나의 데이터베이스에 여러 개 존재할 수 있다.

③ 조직 전체의 관점에서 데이터베이스 구조를 정의한다.

④ 외부 스키마와 개념 스키마 사이에는 물리적 데이터 독립성이 존재한다.

정답 2번

간단한 해설

외부 스키마는 **개별 사용자나 응용 프로그램의 데이터 관점**이다. 사용 목적에 따라 관점이 달라지므로 하나의 데이터베이스에 여러 외부 스키마를 둘 수 있다.

선지별 해설

① X: 데이터의 물리적 저장 방식, 레코드 배치, 접근 경로 등 저장 구조를 명세하는 것은 **내부 스키마**이다. 외부 스키마는 사용자에게 필요한 데이터의 모습을 정의하며 저장장치의 상세 구조를 설명하는 단계가 아니다.

② O: 외부 스키마는 **사용자나 응용 프로그램별 관점**을 나타낸다. 인사 담당자와 급여 담당자가 필요로 하는 데이터가 다를 수 있으므로, 하나의 데이터베이스에 여러 외부 스키마가 존재할 수 있다.

③ X: 조직 전체의 데이터를 통합한 논리적 구조와 관계·제약조건을 정의하는 것은 **개념 스키마**이다. 외부 스키마는 그 전체 구조 가운데 특정 사용자가 필요로 하는 관점을 표현한다.

④ X: 외부 스키마와 개념 스키마 사이에서 다루는 것은 **논리적 데이터 독립성**이다. 개념 스키마를 변경하더라도 적절한 사상으로 기존 외부 관점을 유지할 수 있다는 뜻이다. 물리적 데이터 독립성은 내부 저장 구조의 변경이 개념 스키마에 영향을 주지 않도록 하는 것으로, 개념·내부 단계 사이에 해당한다.

따라서 외부 스키마가 여러 개 존재할 수 있다는 ②가 가장 적절하다.

NoteBase

문 3. 다음에서 설명하는 것으로 가장 적절한 것은?

· 데이터베이스에 저장된 데이터들에 대한 정의와 명세 등의 정보를 저장한다.
· 시스템 카탈로그라고도 한다.
· 시스템과 일반 사용자 모두 접근 가능하며 시스템은 내용의 추가 및 수정이 가능하고 일반 사용자는 검색만 가능하다.

① 뷰
② 데이터 사전
③ 데이터 디렉터리
④ 사용자 데이터베이스

정답 2번

간단한 해설

데이터의 정의·명세를 담는 정보는 메타데이터이며, **이를 관리하는 시스템 카탈로그가 데이터 사전**이다.
시스템이 내용을 유지하고 사용자는 허용 범위에서 조회한다.

개념 구분

데이터 자체가 아니라 테이블·속성·제약조건 등 데이터의 정의와 명세를 설명하는 정보는 **메타데이터**이다. 이를 관리하는 데이터 사전은 시스템 카탈로그라고도 하므로 ②가 해당한다. 스키마를 생성하거나 변경하면 DBMS가 그 정의에 맞게 사전 정보를 유지한다.

일반 사용자는 허용된 범위의 사전 정보를 조회하지만, 사전의 내부 내용을 업무 데이터처럼 직접 추가·수정하는 것은 아니다. 즉 **시스템의 관리와 사용자의 조회를 구분**한다. 사용자 데이터베이스는 실제 업무 데이터를 담고, 뷰는 질의로 정의한 가상 테이블이므로 정의·명세의 관리 저장소를 가리키는 이 설명과 다르다. 데이터 디렉터리는 저장 위치·접근 관리에 관한 개념과 구분하여 이해한다.

NoteBase

문 4. 데이터의 보안, 무결성, 회복과 밀접한 관련이 있는 데이터 언어로 가장 적절한 것은?

① 데이터 정의어(Data Definition Language)

② 데이터 조작어(Data Manipulation Language)

③ 데이터 제어어(Data Control Language)

④ 데이터 관리어(Data Management Language)

정답 3번

간단한 해설

보안·무결성·회복은 데이터의 **안전하고 일관된 이용을 제어**하는 기능과 관련된다. 정의·조작·제어로 나누는 개론적 분류에서는 이러한 통제 기능을 데이터 제어어(DCL)와 연결한다.

개념 구분

보안·무결성·회복이라는 단서는 데이터의 구조나 값 자체보다 **안전하고 일관된 이용을 통제하는 기능**을 가리킨다. 데이터 언어를 정의어·조작어·제어어로 구분하는 이 문제의 개론적 분류에서는 데이터 제어어(DCL)가 가장 적절하다. 대표적으로 GRANT와 REVOKE는 사용자의 접근 권한을 부여하거나 회수한다.

데이터 정의어(DDL)는 CREATE·ALTER·DROP 등으로 테이블과 스키마 구조를 정의·변경한다. 데이터 조작어(DML)는 INSERT·UPDATE·DELETE 등으로 저장된 데이터를 조작한다. 따라서 구조 정의나 값의 변경보다는 이용 통제에 초점이 있는 발문과 구별된다. '데이터 관리어(Data Management Language)'는 이 구분에서 DCL을 대신하는 분류명이 아니다.

세부 SQL 분류와는 범위를 구별해야 한다. COMMIT·ROLLBACK 같은 트랜잭션 제어는 TCL로 따로 구분하기도 하고, PRIMARY KEY·CHECK 같은 무결성 제약은 DDL로 선언한다. 따라서 보안·무결성·회복의 모든 기능을 오직 DCL 명령만으로 구현한다는 뜻은 아니다. 발문의 넓은 제어 기능에 대응하는 선택지는 ③ 데이터 제어어이다.

NoteBase

문 5. 다음은 릴레이션 직원, 거래처에 관한 ER-다이어그램을 작성한 것이다. 해당 릴레이션과 ER-다이어그램에 관한 설명으로 가장 적절하지 않은 것은? (단, 밑줄 속성은 기본키임)

<릴레이션>
직원(직원아이디, 이름, 직책)
거래처(거래처아이디, 직원아이디, 매출액)

<ER-다이어그램>



- 직원 : 단일 사각형
- 거래처 : 이중 사각형
- 관리 : 이중 마름모
- 직원-관리 : 단일선
- 거래처-관리 : 이중선

① 직원 개체는 강한 개체이다.

② 거래처 개체는 관리 관계에 선택적으로 참여한다.

③ 거래처 개체의 거래처아이디 속성은 부분키이다.

④ 거래처 개체는 약한 개체이다.

정답 2번

간단한 해설

거래처와 관리 관계를 잇는 **이중선은 필수적인 전체 참여**를 의미한다. 선택적 참여라는 ②가 틀리며, 이중 사각형의 거래처는 직원의 키에 의존하는 약한 개체이다.

NoteBase

선지별 해설

① O: 직원은 **단일 사각형**으로 표시되고 직원아이디가 기본키이다. 다른 개체의 키를 결합하지 않고 자신의 키로 식별되는 강한 개체이다.

② X: 거래처와 관리 관계 사이의 **이중선은 전체 참여, 즉 필수 참여**를 나타낸다. 따라서 모든 거래처는 관리 관계에 참여해야 하며 선택적으로 참여한다는 설명은 틀리다. 직원 쪽의 단일선과 구별해야 한다.

③ O: 거래처의 기본키는 그림에서 밑줄 친 **(거래처아이디, 직원아이디)의 조합**이다. 거래처아이디는 동일한 소유 직원에 속한 거래처들을 구별하는 부분키이고, 소유 개체의 직원아이디와 함께 완전한 식별자를 구성한다.

④ O: 거래처는 **이중 사각형**으로 표시된 약한 개체이다. 관리의 이중 마름모는 식별 관계이며, 거래처의 식별에 강한 개체인 직원의 키가 필요함을 보여 준다.

NoteBase

문 6. 다음의 릴레이션 R1, R2의 속성 중 외래키로 가장 적절한 것은? (단, 밑줄 속성은 기본키임)

<릴레이션>
R1(고객번호, 이름, 등급)
R2(주문번호, 고객번호, 제품번호)

① R2의 고객번호
② R2의 주문번호
③ R2의 제품번호
④ R1의 등급

정답 1번

간단한 해설

외래키는 다른 릴레이션의 키를 참조하여 두 릴레이션을 연결한다. 제시된 구조에서는 **주문의 고객번호가 고객 릴레이션의 기본키를 참조**하는 관계가 자연스럽다.

핵심 풀이

R1은 고객번호를 기본키로 하여 고객을 식별한다. R2의 고객번호는 주문한 고객을 나타내므로, **R1의 기본키인 고객번호를 참조하는 외래키**로 지정하기에 가장 적절하다. 따라서 선택지는 ①이다. 이 참조를 설정하면 R2의 NULL이 아닌 고객번호는 R1에 존재하는 고객번호와 일치해야 한다.

R2의 주문번호는 제시된 스키마에서 주문 자체를 식별하는 기본키이다. 기본키가 외래키를 겸하는 설계도 가능하지만, 여기에는 주문번호의 참조 대상이 주어져 있지 않다. 제품번호도 별도의 제품 릴레이션이 있다면 외래키가 될 수 있으나 제시된 R1에는 그 참조 대상이 없다. R1의 등급 역시 이 두 릴레이션 사이의 키 참조를 나타내지 않는다. 속성 이름이 같다는 사실만으로 자동으로 외래키 제약이 생기는 것은 아니며, 실제 데이터베이스에서는 참조 관계를 명세해야 한다.

NoteBase

문 7. 다음의 릴레이션에서 위반한 제약조건으로 가장 적절한 것은? (단, 밑줄 속성은 기본키임)

고객아이디	고객이름	나이
P001	김순경	25
NULL	이순경	28
P003	박순경	26
NULL	최순경	30

① 개체 무결성 제약조건

② 참조 무결성 제약조건

③ 도메인 무결성 제약조건

④ 함수 무결성 제약조건

정답 1번

간단한 해설

고객아이디가 기본키인데 일부 행의 값이 NULL이다. **기본키의 NULL을 금지하는 개체 무결성**을 위반한 것이다.

핵심 풀이

밑줄이 있는 고객아이디가 기본키이다. 그런데 두 번째와 네 번째 행의 고객아이디가 NULL이므로 **기본키에는 NULL을 허용하지 않는 조건**을 위반한다. 기본키는 각 튜플을 식별해야 하므로 값이 누락된 상태를 허용하지 않으며, 해당 제약은 개체 무결성인 ①이다.

참조 무결성은 외래키와 참조 대상 사이의 일치 여부에 관한 조건이고, 도메인 무결성은 속성값의 자료형·허용 범위 등에 관한 조건이다. 제시된 표의 결정적 근거는 다른 테이블과의 관계나 나이 범위가 아니라 **기본키 열에 NULL이 존재한다는 점**이다. NULL이 두 번 나타난다는 이유로 두 NULL이 서로 같은 값이라 판단하는 것이 아니라, 각 NULL 자체가 기본키의 필수값 조건에 어긋난다고 이해한다.

NoteBase

문 8. 릴레이션의 특성에 관한 설명으로 가장 적절하지 않은 것은?

① 하나의 릴레이션에는 동일한 투플이 존재할 수 없다.

② 하나의 릴레이션에서 투플 사이의 순서가 바뀌면 다른 릴레이션으로 변한다.

③ 속성 값으로 원자 값만 사용할 수 있다.

④ 하나의 릴레이션에서 속성 사이의 순서는 무의미하다.

정답 2번

간단한 해설

릴레이션은 **순서 없는 투플의 집합**이다. 투플의 나열 순서만 바꾸어도 같은 투플 집합이면 동일한 릴레이션이다.

선지별 해설

① O: 관계 모델의 릴레이션은 투플의 집합이므로 **모든 속성 값이 같은 중복 투플**을 허용하지 않는다. 일부 속성 값이 같은 서로 다른 투플은 가능하다. 중복 행을 허용하는 SQL 테이블·질의 결과와 순수한 관계 모델을 구분해야 한다.

② X: 릴레이션에서 **투플의 순서는 의미가 없다**. 같은 투플들을 위아래로 다르게 나열해도 투플 집합이 같으면 같은 릴레이션이다. 화면 표시 순서나 물리적 저장 순서는 릴레이션의 동일성을 결정하지 않는다.

③ O: 이 문항이 전제한 기본적인 1정규형 관계 모델에서는 각 속성 위치에 해당 도메인의 **하나의 원자 값**을 둔다. 하나의 칸에 반복 그룹이나 값의 집합을 넣는 구조와 구별한다. 원자성은 해당 모델에서 하나의 값으로 취급한다는 뜻이지 문자열을 문자 단위로 분석할 수 없다는 뜻은 아니다.

④ O: 속성은 이름으로 식별하므로 **속성 이름과 값의 대응을 유지**한 채 열의 표시 순서만 바꾸어도 릴레이션의 의미는 같다. 단순히 열 이름을 그대로 두고 서로 다른 속성의 값만 바꾸는 것은 열 순서 변경이 아니다.

따라서 투플 순서가 바뀌면 다른 릴레이션이 된다는 ②가 가장 적절하지 않다.

NoteBase

문 9. 다음의 조건에 따라 제품관리 테이블에 관한 SQL 코드를 작성하려 할 때, (가), (나)에 들어갈 코드로 가장 적절한 것은? (단, 밑줄 속성은 기본키임)

<조건>
- 제조하는 제품의 수가 2개 이상인 제조업체를 그룹으로 지정
- 각 그룹의 제조업체, 제조업체가 생산한 제품의 수(제품수), 매출의 평균(평균매출액)을 출력

<제품관리>

제품번호	제품	제조업체	매출
PR1	수건	A	1000
PR2	컵	B	2000
PR3	양말	A	1500
PR4	식기	B	2300
PR5	의자	C	800

<코드>

```
SELECT 제조업체, (가)(*) AS 제품수, (나)(매출) AS 평균매출액
FROM 제품관리
GROUP BY 제조업체 HAVING (가)(*)>=2;
```

① (가): COUNT (나): AVG

② (가): COUNTS (나): AVERAGE

③ (가): DISTINCT (나): AVG

④ (가): DISTINCT ALL (나): AVERAGE

정답 1번

간단한 해설

제품 수를 세는 집계 함수는 **COUNT(*)**, 매출 평균을 구하는 함수는 **AVG(매출)**이다. 제조업체별로 묶은 뒤 HAVING COUNT(*) >= 2를 적용하면 제품이 2개인 A와 B만 남는다.

NoteBase

빈칸별 해설

(가) COUNT: 제조업체별 제품 수는 각 그룹에 속한 행 수이므로 **COUNT(*)**로 구한다. 제품번호가 기본키인 표에서 각 행이 제품 하나를 나타낸다. 같은 (가)가 들어가는 HAVING COUNT(*) >= 2는 그룹을 만든 뒤 제품이 2개 이상인 업체만 남긴다. DISTINCT는 중복 제거 지정어이지 이 자리에 넣어 행 수를 세는 집계 함수가 아니며, 표준 함수명은 COUNTS가 아니라 COUNT이다.

(나) AVG: 각 업체의 매출 평균은 **AVG(매출)**로 구한다. 제시된 표에는 매출의 NULL이 없으므로 각 그룹의 매출 합계를 제품 수로 나눈 값과 같다. 일반적으로 AVG는 NULL이 아닌 입력값을 대상으로 평균을 구하며, 이 집계 함수명은 AVERAGE가 아니라 AVG이다.

GROUP BY 제조업체를 적용하면 A는 PR1·PR3, B는 PR2·PR4, C는 PR5로 묶인다. A와 B는 각각 제품 2개여서 HAVING 조건을 통과하고, C는 1개여서 제외된다. 평균매출액은 A가 $\frac{1000+1500}{2}=1250$, B가 $\frac{2000+2300}{2}=2150$이다.

따라서 (가)는 COUNT, (나)는 AVG이며 두 빈칸이 모두 일치하는 선택지는 ①이다.

NoteBase

문 10. 순수 관계 연산자 중 셀렉트(σ)에 관한 설명으로 가장 적절한 것은?

① 연산 결과로 조건을 만족하는 투플들이 출력된다.

② 연산을 위해 반드시 두 개의 릴레이션이 필요하다.

③ 일반 집합 연산자 중 합집합과 동일한 결과를 출력한다.

④ 연산 결과로 조건을 만족하는 속성들이 출력되며, 중복된 투플이 포함될 수 있다.

정답 1번

간단한 해설

셀렉트 σ는 **하나의 릴레이션에서 조건을 만족하는 행을 선택**하는 단항 연산이다. 열을 선택하는 프로젝트 π와 구별하면 ①이 옳다.

선지별 해설

① O: 셀렉트 σ는 하나의 릴레이션에서 **주어진 조건을 만족하는 튜플, 즉 행**을 골라낸다. 예를 들어 학년=2라는 조건을 적용하면 2학년인 행만 남고 속성 구성은 그대로 유지된다.

② X: 셀렉트는 **하나의 릴레이션을 입력으로 받는 단항 연산**이다. 두 릴레이션이 필요한 합집합·차집합·조인 등의 이항 연산과 구별해야 한다.

③ X: 합집합은 합병 가능한 두 릴레이션 중 어느 한쪽에 속하는 튜플을 모으지만, 셀렉트는 **한 릴레이션의 행을 조건으로 거른다.** 특정 자료에서 결과가 우연히 같을 수 있어도 두 연산이 동일한 의미를 갖는 것은 아니다.

④ X: 셀렉트는 속성인 열이 아니라 **조건에 맞는 행을 선택**한다. 열을 선택하는 연산은 프로젝트 π이다. 또한 순수 관계대수의 릴레이션은 집합이므로 중복 튜플을 포함하지 않으며, SQL의 중복 허용 결과와 혼동하면 안 된다.

NoteBase

문 11. 다음은 합격자 테이블을 생성하는 SQL 코드이다. 아래 코드에 관한 설명으로 가장 적절하지 않은 것은?

```
CREATE TABLE 합격자(
합격번호 VARCHAR(10) NOT NULL,
합격점수 INT DEFAULT 0,
시험응시번호 VARCHAR(10) NOT NULL,
PRIMARY KEY(합격번호),
UNIQUE(시험응시번호)
);
```

① 합격번호 속성을 합격자 테이블의 기본키로 지정한다.

② 합격점수 속성에 값을 입력하지 않으면 기본값으로 0이 지정된다.

③ 시험응시번호 속성을 합격자 테이블의 외래키로 지정한다.

④ 합격번호 속성은 NULL이 아닌 가변길이의 문자열 데이터이다.

정답 3번

간단한 해설

UNIQUE는 중복 금지, FOREIGN KEY는 참조 관계를 위한 제약이다. 이 코드의 시험응시번호에는 UNIQUE와 NOT NULL만 선언되어 있으며 외래키 참조는 없다.

선지별 해설

① O: PRIMARY KEY(합격번호)는 합격번호를 합격자 테이블의 기본키로 지정한다. 이 속성은 각 행을 유일하게 식별하며, 중복 값과 NULL을 허용하지 않는다.

② O: 합격점수에는 DEFAULT 0이 선언되어 있으므로 행을 삽입할 때 이 열의 값을 생략하거나 DEFAULT를 지정하면 0을 사용한다. **명시적으로 NULL을 넣는 것과 값을 생략하는 것은 다르며**, 이 코드의 일반적인 DEFAULT 선언이 명시적 NULL까지 자동으로 0으로 바꾸는 것은 아니다.

③ X: UNIQUE(시험응시번호)는 시험응시번호의 **중복을 금지하는 유일성 제약**이다. 다른 테이블을 참조하도록 하는 외래키 선언이 아니며, 제시된 코드에는 FOREIGN KEY나 REFERENCES가 없다. 이 열에는 NOT NULL도 있으므로 NULL 없이 유일한 값을 가져야 한다.

④ O: VARCHAR(10)은 지정된 길이 한도 안에서 길이가 달라질 수 있는 문자열을 저장하는 자료형이며, NOT NULL은 NULL 저장을 금지한다. 따라서 합격번호는 NULL이 아닌 가변길이 문자열이고, 기본키 제약에 의해 유일성도 갖는다.

NoteBase

문 12. 뷰에 관한 설명으로 가장 적절하지 않은 것은?

① 사용자들의 데이터 접근을 제어하여 보안성을 높일 수 있다.

② 뷰를 삭제하면 참조한 기본 테이블도 삭제된다.

③ WITH CHECK OPTION을 사용하여 뷰를 정의하면 삽입 또는 갱신되는 투플에 대해 제한을 둘 수 있다.

④ GROUP BY를 사용한 뷰에 대해 투플의 삽입 또는 갱신이 불가능하다.

정답 2번

간단한 해설

뷰는 기본 테이블을 바탕으로 정의한 가상 테이블이다. **뷰 정의를 삭제해도 참조하던 기본 테이블과 데이터는 남는다**.

선지별 해설

① (O) 필요한 행과 열만 보여 주는 뷰를 정의하고 그 뷰에 대한 권한을 부여하면 **사용자에게 노출되는 데이터를 제한**할 수 있다. 보안 효과를 얻으려면 기본 테이블의 직접 접근 권한도 함께 적절히 관리해야 한다.

② (X) DROP VIEW는 뷰의 정의를 제거하는 것이지, 뷰가 참조하던 **기본 테이블을 삭제하는 것이 아니다**. 다른 뷰 등 종속 객체에 영향을 줄 수는 있지만, 참조 대상인 기본 테이블까지 거꾸로 삭제된다는 설명은 틀리다.

③ (O) WITH CHECK OPTION은 갱신 가능한 뷰를 통해 삽입·수정한 행이 **뷰의 정의 조건을 계속 만족**하도록 검사한다. 조건을 벗어나 뷰에서 보이지 않게 되는 데이터를 해당 경로로 만드는 것을 제한한다.

④ (O) GROUP BY로 여러 기본 행을 묶은 결과는 일반적인 자동 갱신 가능 뷰가 아니다. 결과 한 행의 변경을 어떤 기본 행에 반영할지 직접 결정하기 어려워 **통상적인 뷰 삽입·갱신이 제한**된다. 별도 트리거 등으로 구현한 특별한 변경 처리는 이 기본 갱신 가능성 판단과 구별한다.

NoteBase

문 13. 질의 최적화 방법으로 가장 적절하지 않은 것은?

① 최소 비용을 가진 질의문 계획을 선정하기 위해 디스크 입출력 비용, 저장 비용 등을 고려하여야 한다.

② 셀렉트(σ) 연산에 대해 교환법칙이 성립한다.

③ 카티션 프로덕트(X) 연산에 대해 교환법칙이 성립한다.

④ 실제 주어진 질의문에 대해 가능한 모든 질의문 계획을 모두 생성하여야 한다.

정답 4번

간단한 해설

실용적인 질의 최적화는 **모든 실행 계획을 반드시 생성하지 않는다**. 비용 추정과 동치 변환을 활용하면서 가지치기나 경험적 탐색으로 계획 탐색의 부담을 줄인다.

선지별 해설

① O: 비용 기반 최적화는 실행 계획별 **디스크 입출력·CPU·중간 결과 저장 등의 예상 비용**을 비교한다. 데이터 양과 통계, 접근 경로 등을 이용하여 비용을 추정하며, 비용 모형이 실제 실행 시간을 완벽하게 예측한다는 뜻은 아니다.

② O: 같은 릴레이션에 조건 p와 q의 선택 연산을 연속 적용하면, p를 먼저 적용하든 q를 먼저 적용하든 두 조건을 모두 만족하는 투플이 남는다. 이처럼 **선택 연산의 적용 순서를 교환**할 수 있어, 중간 결과를 더 많이 줄이는 조건을 먼저 처리하는 변환에 활용한다.

③ O: 이름으로 속성을 구분하는 관계대수에서는 **카티션 프로덕트의 두 피연산자 순서를 교환**해도 동일한 속성 값의 조합을 얻는다. 표현상 열 순서가 달라질 수 있으므로 속성의 대응을 유지하여 비교하며, 같은 이름의 속성은 구분되도록 한정하거나 재명명한다.

④ X: 가능한 모든 실행 계획을 반드시 생성해야 하는 것은 아니다. 계획 수가 급격히 증가하므로 **가지치기·동적 계획법·경험적 탐색** 등으로 탐색 비용을 줄인다. 제한된 시간 안에 좋은 계획을 찾는 것이 중요하며, 실용 최적화가 항상 전역 최적 계획을 보장하는 것도 아니다.

따라서 모든 가능한 계획을 모두 생성해야 한다는 ④가 가장 적절하지 않다.

NoteBase

문 14. 다음의 함수 종속을 만족하는 릴레이션 R의 정규형으로 가장 적절한 것은? (단, 밑줄 속성은 기본키임)

<릴레이션>
R(사번, 직급, 연봉)

사번	직급	연봉
S1	사원	2500
S2	대리	3000
S3	사원	2500
S4	과장	3500

<함수종속>
사번 → 직급
사번 → 연봉
직급 → 연봉

① 제2정규형

② 제3정규형

③ BCNF

④ 제5정규형

정답 1번

간단한 해설

후보키 사번은 단일 속성이므로 부분 함수 종속이 없어 제2정규형을 만족한다. 그러나 **사번→직급→연봉**의 이행 종속 때문에 제3정규형은 만족하지 않는다.

NoteBase

핵심 풀이

키와 제2정규형: 사번이 기본키이고, 주어진 종속성으로 사번→직급·연봉이 성립한다. 다른 속성들로는 사번을 결정할 수 없으므로 제시된 종속성 아래의 후보키는 사번이다. 값들이 원자값인 제1정규형이며, 후보키가 단일 속성이어서 **비주요 속성의 부분 함수 종속이 없다.** 따라서 제2정규형을 만족한다.

제3정규형 검사: **사번→직급→연봉**의 이행 함수 종속이 존재한다. 직급은 사번을 결정하지 못하는 비슈퍼키이고 연봉도 후보키에 속하지 않는 비주요 속성이므로, 직급→연봉은 제3정규형 조건을 위반한다.

더 높은 정규형과 구별: 직급→연봉의 결정자인 직급이 슈퍼키가 아니므로 BCNF도 만족하지 않는다.
제3정규형조차 충족하지 못한 이 스키마를 제5정규형으로 판단할 수도 없다. 이 판단은 표의 우연한 값 일치가 아니라 **명시된 함수 종속**에 근거한다.

따라서 제시된 선택지 중 충족하는 가장 높은 정규형은 **제2정규형**이며 ①이 가장 적절하다.

NoteBase

문 15. 체크포인트를 이용한 데이터 복구에 관한 설명으로 가장 적절하지 않은 것은?

① 체크포인트 이전에 커밋된 트랜잭션에 대해 아무것도 할 필요가 없다.

② 체크포인트 이전에 트랜잭션이 시작되어 체크포인트 이후에 커밋된 트랜잭션에 대해 REDO를 수행하여야 한다.

③ 체크포인트 이전에 트랜잭션이 시작되어 시스템 장애가 일어났을 때 커밋이 이루어지지 않은 트랜잭션에 대해 REDO를 수행하여야 한다.

④ 체크포인트 이후에 트랜잭션이 시작되어 시스템 장애가 일어났을 때 커밋이 이루어지지 않은 트랜잭션에 대해 UNDO를 수행하여야 한다.

정답 3번

간단한 해설

기본 UNDO/REDO 회복에서 장애 시 **미커밋 트랜잭션은 UNDO로 취소**한다. 체크포인트 전후의 시작 시점만으로 REDO 대상을 정하는 것이 아니라, 완료된 결과를 보존해야 하는지 커밋 여부를 확인한다.

선지별 해설

보기는 체크포인트 이전의 완료 결과가 반영되어 있고, 커밋 전 변경이 저장될 수 있는 기본 UNDO/REDO 회복 모형을 전제로 구분한다. 커밋된 작업은 필요한 경우 재실행하고 미완료 작업은 취소한다. 지연 갱신의 NO-UNDO 방식이나 ARIES의 이력 재현 단계까지 모든 회복 구현에 동일하게 적용되는 절차라는 뜻은 아니다.

① O: 체크포인트 이전에 커밋된 트랜잭션의 변경은 검사점 처리로 디스크에 안전하게 반영되어 있다고 보는 기본 회복 모형이다. 따라서 그 트랜잭션을 다시 REDO하거나 UNDO할 필요가 없다.

② O: 체크포인트 때는 실행 중이었지만 장애 전에는 커밋된 트랜잭션이다. 완료된 변경이 디스크에 모두 남아 있지 않을 수 있으므로, 로그를 이용해 **REDO하여 커밋된 결과를 복원**한다. 시작 시점이 체크포인트 전이라는 이유로 제외하지 않는다.

③ X: 장애 시 커밋되지 않은 트랜잭션의 변경은 완료된 결과로 유지하면 안 된다. 이 문제의 기본 UNDO/REDO 회복 분류에서는 **REDO가 아니라 UNDO**로 미완료 변경을 취소한다. 체크포인트 이전에 시작했다는 사실보다 장애 시점의 커밋 여부가 중요하다.

④ O: 체크포인트 이후 시작했더라도 장애 시 미완료라면 취소 대상이다. 즉시 갱신처럼 커밋 전 변경이 디스크에 반영될 수 있는 방식에서는 해당 변경을 UNDO하여 원자성을 보장한다. 체크포인트 이후에 시작했다는 이유만으로 미완료 변경을 남겨 두지 않는다.

NoteBase

문 16. 다음은 ㉠~㉤ 순서대로 실행된 권한 부여 및 회수에 관한 SQL 코드이다. 각 명령문에 관한 설명으로 가장 적절하지 않은 것은?

㉠ DBA : GRANT INSERT, DELETE ON STUDENT TO U1;
㉡ DBA : GRANT SELECT ON STUDENT TO U2 WITH GRANT OPTION;
㉢ U2 : GRANT SELECT ON STUDENT TO U3;
㉤ DBA : REVOKE SELECT ON STUDENT FROM U2 CASCADE;

① ㉠에 의해 U1은 STUDENT 릴레이션에 대한 삽입과 삭제 권한을 갖게 된다.
② ㉡에 의해 U2는 STUDENT 릴레이션에 대한 검색 권한을 갖게 된다.
③ ㉢에 의해 U3는 U2로부터 부여받은 검색 권한을 다른 사용자에게 전파할 수 있다.
④ ㉤에 의해 U2로부터 받은 U3의 권한이 회수된다.

정답 3번

간단한 해설

권한을 사용하는 능력과 다시 부여하는 능력은 다르다. U3에게 전달된 SELECT에는 WITH GRANT OPTION이 없으므로 재부여는 할 수 없으며, U2의 권한을 CASCADE로 회수하면 그에 의존한 U3의 권한도 회수된다.

선지별 해설

① O: ㉠은 STUDENT에 대한 INSERT와 DELETE를 U1에게 부여한다. 따라서 U1은 이 명령에 의해 행의 삽입과 삭제 권한을 얻는다. 마지막 SELECT 회수 명령은 U1의 이 권한을 회수하지 않는다.

② O: ㉡은 U2에게 SELECT 권한을 부여하므로 U2는 이 단계에서 STUDENT를 검색할 수 있다. WITH GRANT OPTION도 포함되어 있어 해당 검색 권한을 다른 사용자에게 다시 부여할 수 있다. 이후 ㉤에서 이 권한이 회수되는 것과 ㉡ 자체의 효과를 구별해야 한다.

③ X: ㉢에는 **WITH GRANT OPTION이 없다**. 따라서 U3는 U2로부터 SELECT 사용 권한만 받을 뿐, 이 부여를 근거로 다른 사용자에게 전파할 권한은 받지 않는다. U2가 재부여 권한을 갖고 있다고 해서 그 권한까지 U3에게 자동으로 전달되지는 않는다.

④ O: ㉤은 U2의 SELECT 권한을 회수하면서 **CASCADE로 종속된 권한 부여도 함께 회수**한다. U3가 U2에게 받은 SELECT 권한은 이 경로에 의존하므로 회수된다. 제시된 명령과 무관한 다른 부여 경로가 있다면 그 별도 권한까지 무조건 소멸하는 것은 아니다.

NoteBase

문 17. 다음의 트랜잭션 스케줄에서 충돌 직렬 가능 스케줄인 것은?

① Read T1(X);
Read T2(X);
Write T2(X);
Write T1(X);
Read T3(X);

② Write T1(X);
Read T3(X);
Read T2(X);
Write T2(X);
Read T1(X);

③ Read T3(X);
Read T1(X);
Write T1(X);
Read T2(X);
Write T2(X);

④ Read T3(X);
Read T2(X);
Write T2(X);
Write T3(X);
Read T1(X);

정답 3번

간단한 해설

서로 다른 트랜잭션이 같은 항목을 다루고 하나 이상이 쓰기이면 선행 연산에서 후행 연산으로 간선을 만든다.
충돌 그래프에 사이클이 없어야 충돌 직렬 가능하며, ③은 T3→T1→T2로 정렬된다.

NoteBase

선지별 해설

① (X) T1의 읽기가 T2의 쓰기보다 앞서 T1→T2가 생긴다. 반대로 T2의 읽기·쓰기는 T1의 쓰기보다 앞서 T2→T1도 생긴다. **T1→T2→T1의 사이클**이 있어 충돌 직렬 가능하지 않다. 두 쓰기가 뒤의 T3 읽기보다 앞선 간선도 있지만 이 사이클은 없어지지 않는다.

② (X) 처음 T1의 쓰기가 T2의 읽기·쓰기보다 앞서 T1→T2가 생기고, T2의 쓰기가 마지막 T1의 읽기보다 앞서 T2→T1이 생긴다. 따라서 **T1과 T2 사이에 양방향 의존성**이 존재하여 충돌 직렬 가능하지 않다. T1→T3와 T3→T2도 형성된다.

③ (O) T3의 읽기가 T1·T2의 쓰기보다 앞서 T3→T1, T3→T2가 생기고, T1의 읽기·쓰기가 T2의 쓰기·읽기보다 앞서 T1→T2가 생긴다. 역방향 간선이나 사이클은 없으므로 **T3→T1→T2 순서로 직렬화**할 수 있다.

④ (X) T3의 읽기가 T2의 쓰기보다 앞서 T3→T2가 생긴다. T2의 읽기·쓰기는 T3의 쓰기보다 앞서 T2→T3도 생기므로 **T3→T2→T3의 사이클**이 존재한다. T1이 마지막에 읽는다고 해서 앞의 사이클이 해소되지는 않는다.

NoteBase

문 18. 인덱스 기법에 관한 설명으로 가장 적절한 것은?

① B 트리 전체를 노드 내의 키 값에 따라 순차 검색하기 위해서는 트리의 각 노드를 전위 순회하여야 한다.

② B 트리에서 오버플로우가 발생하여 리프 노드가 분할될 때, 중간 키 값이 부모 노드뿐만 아니라 새로 분할된 노드에도 중복 저장된다.

③ B+ 트리의 삽입에서 오버플로우가 발생한 경우, 분할이 일어나지 않고 새로운 추가 노드에 키 값이 삽입된다.

④ B+ 트리에서 모든 리프 노드는 같은 레벨에 존재한다.

정답 4번

간단한 해설

B+트리는 **모든 리프 노드의 깊이가 같은 균형 트리**이므로 ④가 옳다. B-트리의 중간 키 이동과 B+트리 리프의 경계 키 복사도 구별해야 한다.

선지별 해설

① X: B-트리의 키를 정렬된 순서로 방문하려면 자식 서브트리와 노드 안의 구분 키를 차례로 섞어 방문하는 **중위 순회에 해당하는 방식**을 사용한다. 노드의 키를 자식보다 먼저 방문하는 전위 순회는 키 순서를 보장하지 않는다.

② X: B-트리의 노드 분할에서는 **중간 키를 부모로 이동**시키므로 분할된 자식에 같은 중간 키를 중복 저장하지 않는다. B+트리의 리프 분할에서 데이터 키를 리프에 남기고 경계 키를 부모로 복사하는 방식과 구별해야 한다.

③ X: B+트리에서 노드가 가득 찬 상태로 삽입되어 오버플로우가 생기면 기본적으로 **노드를 분할하고 부모에 구분 키를 반영**한다. 새 노드가 생기는 것은 분할 과정의 일부이지, 분할 없이 임의의 추가 노드에만 키를 넣는다는 뜻이 아니다.

④ O: B+트리는 균형 탐색 트리이므로 **모든 리프 노드가 같은 레벨**에 있다. 삽입·삭제 시 분할·병합 등의 조정을 통해 이 균형을 유지하며, 루트가 분할되면 트리 높이가 증가한다.

NoteBase

문 19. 분산 데이터베이스의 특성 중 투명성(Transparency)에 관한 설명으로 가장 적절하지 않은 것은?

① 데이터베이스가 분산된 각 지역의 시스템이나 통신망에 이상이 생겨도 데이터의 무결성을 보존할 수 있어야 하는 개념을 장애 투명성(Failure Transparency)이라고 한다.

② 사용자가 데이터의 중복성, 백업상황 등에 대해 고려하지 않고도 데이터를 활용할 수 있어야 하는 개념을 중복 투명성(Replication Transparency)이라고 한다.

③ 여러 사용자나 프로그램이 동시에 분산 데이터베이스에 대한 트랜잭션을 수행해도 결과에 이상이 없어야 하는 개념을 병행 투명성(Concurrency Transparency)이라고 한다.

④ 사용자가 데이터의 물리적 위치를 고려하여 데이터를 활용할 수 있어야 하는 개념을 위치 투명성(Location Transparency)이라고 한다.

정답 4번

간단한 해설

위치 투명성은 물리적 저장 위치를 몰라도 데이터를 이용할 수 있게 하는 성질이다. 사용자에게 위치를 고려하도록 요구하는 개념이 아니다.

선지별 해설

① O: **장애 투명성**은 사이트나 통신망 장애에 대한 처리·복구를 시스템이 담당하여 트랜잭션의 원자성과 데이터의 무결성을 지키도록 하는 성질이다. 장애가 발생해도 일부 변경만 확정된 불일치 상태를 남기지 않아야 한다. 모든 장애에서 중단이나 대기 없이 서비스를 계속한다는 무조건적 보장은 아니다.

② O: **중복·복제 투명성**은 사용자가 복제본의 수나 배치·관리 상황을 직접 알아야만 데이터를 이용할 수 있는 것은 아니라는 뜻이다. 시스템이 복제 데이터의 접근과 일관성 관리를 맡아 논리적으로 하나의 데이터처럼 이용하도록 한다. 여기서 백업상황이라는 표현은 복제본 관리의 세부 사항을 의식하지 않는다는 취지이며, 복제와 시점 백업이 동일한 기술이라는 뜻은 아니다.

③ O: **병행 투명성**은 여러 트랜잭션이 동시에 실행되더라도 서로의 간섭으로 갱신 손실 등 부정합이 발생하지 않도록 동시성 제어를 제공하는 성질이다. 필요한 격리·일관성 조건을 보존하는 것이 핵심이며, 대기나 재시도 자체가 전혀 없다는 뜻은 아니다.

④ X: **위치 투명성**은 사용자가 데이터의 물리적 위치를 고려하지 않고도 접근할 수 있게 하는 것이다. 데이터가 어느 사이트에 저장되어 있는지에 관한 세부 사항을 감추므로, 물리적 위치를 고려해야 한다는 설명은 반대이다.

따라서 위치 투명성의 의미를 거꾸로 설명한 ④가 가장 적절하지 않다.

NoteBase

문 20. 다음은 각 트랜잭션에서 구매한 물건들을 나타내는 구매 데이터이다. 연관 규칙 기법으로 구매 데이터를 분석할 때 가장 적절하지 않은 것은?

트랜잭션 아이디	구매 항목
1	A, B, C
2	C, D
3	A, B, D
4	A, C
5	A, B, C, D

① 연관 규칙 A ⇒ B의 지지도는 40%이다.

② 연관 규칙 C ⇒ D의 지지도는 40%이다.

③ 연관 규칙 A ⇒ B의 신뢰도는 75%이다.

④ 연관 규칙 C ⇒ B의 신뢰도는 50%이다.

정답 1번

간단한 해설

지지도는 전체 거래를, 신뢰도는 조건부 항목이 포함된 거래를 분모로 삼는다. A와 B를 함께 구매한 거래는 전체 5건 중 3건이므로 **A ⇒ B의 지지도는 60%**이다.

선지별 해설

① X: 지지도는 **전체 거래 중 두 항목을 모두 포함한 거래의 비율**이다. A와 B를 함께 구매한 거래는 1, 3, 5번으로 3건이므로, 지지도는 3/5 × 100 = 60%이다. 40%가 아니라 60%로 고쳐야 한다.

② O: C와 D를 함께 구매한 거래는 2, 5번으로 2건이다. 전체 5건을 분모로 삼으므로 C ⇒ D의 지지도는 2/5 × 100 = 40%이다.

③ O: 신뢰도는 **조건부 항목을 포함한 거래 중 결과 항목도 포함한 거래의 비율**이다. A를 구매한 거래는 1, 3, 4, 5번으로 4건이고, 이 중 B도 구매한 거래는 1, 3, 5번으로 3건이므로 신뢰도는 3/4 × 100 = 75%이다.

④ O: C를 구매한 거래는 1, 2, 4, 5번으로 4건이다. 이 중 B도 구매한 거래는 1, 5번으로 2건이므로 C ⇒ B의 신뢰도는 2/4 × 100 = 50%이다.

NoteBase