데이터베이스론	국가 전산 7급	2018년 8월 18일

♤ 합격선/최종합격인원(74.16점/31명) - 채용예정인원 29명 / 필기합격 42명 ♤

1. 파일처리 시스템에서 데이터 중복의 단점에 대한 설명으로 옳지 않은 것은? [2018년 국가 7급]

① 같은 데이터가 여러 곳에 중복되어 있어 동일 수준의 보안이 가능하다.
② 데이터 저장공간에 대한 추가 비용이 소요된다.
③ 데이터 간의 불일치로 인해 데이터 일관성이 결여된다.
④ 데이터 갱신 시 중복된 모든 데이터를 찾아내어 갱신해야 하므로 갱신 비용이 추가된다.

♤ **파일처리 시스템에서 데이터 중복의 단점**

- 같은 데이터가 **여러 곳**에 중복되어 있어 **동일 수준의 보안**이 가능하다.(×)
  → 동일 수준의 보안이 **불가능**하다. 상황에 따라 보안 수준은 다를 수도 있다.

// 데이터 중복성

- 데이터 중복성은 하나의 시스템 내에 같은 데이터가 중복 저장되어 관리되는 것이다.
- 데이터 중복성의 문제점

경제성 (economics)	• 자료를 저장하기 위한 공간이 많이 필요하며, • 자료를 갱신하는 경우에 중복된 모든 자료에 대해서 작업을 해야 한다.
보안성 (security)	• 분산되어 있는 중복 자료에 대해 똑같은 보안등급 유지가 어렵다.
일관성 (consistency)	• 동일 사실을 나타내는 데이터는 모두 동일한 값을 가지도록 유지해야 한다. • 데이터가 여러 곳에 중복되어 있으면 동일한 값 유지가 어렵다.**(모순 현상)**
무결성 (integrity)	• 무결성은 데이터가 **정확한 값**을 유지하는 것을 의미한다. • 데이터는 실시간 변해간다. 중복된 모든 자료의 정확성 유지는 어렵다. • 데이터베이스에 저장된 값과 현실 세계의 **실제 값과 일치**하는 것이다.

// 데이터 종속성

- 데이터 종속성은 응용프로그램과 데이터 사이의 상호의존성이다.
- 데이터 구성 방법이나 접근 방법을 변경하면, 응용프로그램도 같이 변경시켜야 한다.
- 예 : 순차파일을 직접파일로 변경하면, 응용프로그램도 같이 변경시켜야 한다.
- 데이터 파일구조를 변경해야 할 때
  이미 개발된 모든 응용프로그램을 변경해야 하는 것은 매우 중대한 문제이다.

정답 : ①

2. 다음 데이터베이스 설계의 주요 단계를 순서대로 바르게 나열한 것은? [2018년 국가 7급]

ㄱ. 정규화	ㄴ. 물리적 설계
ㄷ. 요구사항 수집 및 분석	ㄹ. 개념적 설계
ㅁ. 데이터베이스 튜닝	ㅂ. 논리적 설계

① ㄱ, ㄷ, ㄹ, ㅂ, ㄴ, ㅁ
② ㄱ, ㄷ, ㅂ, ㄹ, ㅁ, ㄴ
③ ㄷ, ㄹ, ㅂ, ㄱ, ㄴ, ㅁ
④ ㄷ, ㅂ, ㄹ, ㄱ, ㅁ, ㄴ

♤ **데이터베이스 설계의 주요 단계**

ㄷ. 요구사항 수집 및 분석 : 사용자들이 원하는 데이터베이스가 무엇인지를 파악하는 단계
↓ 사용자와 응용프로그램의 요구사항을 분석한다.
↓ 데이터베이스에 저장될 자료의 범위가 확정된다.
↓
ㄹ. 개념적 설계 : 개념스키마 모델링(자료중심설계)
↓ 트랜잭션 모델링(처리중심설계)
↓ 특정 DBMS와는 독립적인 개념스키마를 설계한다.
↓
ㅂ. 논리적 설계 : 실제로 사용하게 될 DBMS가 취급하는 형태의 스키마를 설계
↓ 트랜잭션 인터페이스 설계(트랜잭션의 전체적인 골격을 정의)
↓
↓
ㄱ. 정규화 : 하나의 릴레이션에는 기본적으로 하나의 종속성이 표현되도록 릴레이션 분해
↓ 데이터 중복이 최소화되도록 데이터를 구조화한다.
↓ 이상(문제점)이 발생되지 않도록 한다.
↓
ㄴ. 물리적 설계 : 물리적 구조 설계(저장 레코드 양식, 레코드 집중, 접근 경로 설계)
↓ 완전한 트랜잭션 세부 설계(트랜잭션 처리도 고려)
↓
ㅁ. 데이터베이스 튜닝 : 보다 효율적인 데이터베이스가 되도록 개선한다.
데이터베이스가 가능한 빠르게 쿼리를 처리할 수 있도록 한다.
동시에 가능한 많은 동시 사용자를 지원할 수 있도록 한다.

정답 : ③

3. 함수적 종속성과 정규화에 대한 설명으로 옳지 않은 것은? [2018년 국가 7급]

① 함수적 종속성 X→Y는 릴레이션 내 임의의 두 튜플에서 속성 X의 값이 같을 경우, 속성 Y의 값도 항상 같음을 의미한다.
② 함수적 종속성은 릴레이션 상태(외연)의 특성이다.
③ 제2정규형에서 릴레이션의 모든 비주요 속성들은 릴레이션의 기본키에 완전하게 함수적으로 종속한다.
④ 정규화는 함수적 종속성과 기본키를 기반으로 주어진 릴레이션 스키마를 분석하는 과정이다.

♤ **함수적 종속성과 정규화**

• 함수적 **종속성**은 **릴레이션 상태(외연)의 특성**이다.(×)
→ 함수적 종속성은 튜플을 구성하는 **속성들 사이의 특성**이다.**(속성들 사이의 종속 관계)**

예	{직원ID} → {직원생일} {직원ID}는 정확히 하나의 {직원생일}을 가진다. {직원ID}와 {직원생일}은 함수종속 관계이다.

정답 : ②

4. 위치기반서비스(Location-Based Services)를 제공하는 시스템의 요소로 옳지 않은 것은?

① 푸시(push) 기반 서비스
② 최인접 주변 질의
③ TSQL2 언어
④ 윈도우 질의

♤ **위치기반서비스(Location-Based Services, LBS)**

// 위치기반서비스

• 무선인터넷 사용자에게 사용자의 변경 위치에 따른 특정 정보를 제공하는 서비스이다.
• 휴대폰 같은 이동통신망과 IT기술을 종합적으로 활용한 위치정보기반 서비스를 말한다.
• 위치기반서비스는 고객의 위치정보를 기반으로 다양한 정보를 제공할 수 있다.
• 위치기반서비스 앱 : 야놀자, 네이버지도, 카카오택시, 관광맛집 등

// 응용 서비스

• 윈도우(window) 질의 : 반경 10m 안의 주유소를 찾는 경우
• Nearest Neighbor(NN) : 가장 가까운 거리에 있는 주유소를 찾는 경우(최인접 주변 질의)
• k Nearest Neighbor(KNN) : 가까이 있는 k개의 주유소를 찾는 경우

- 지역적으로 분산된 자원의 관리 : 택시, 배달원, 대여 장비, 병원 등
- 사람이나 물건 위치 찾기 : 필요한 서비스 제공자(의사 등), 숙소 예약, 분실 휴대전화 등
- 목표 근접 시 알림 기능(push형 또는 pull형) : 친구, 데이트 상대 찾기 emd
- 목표 근접 시 자동 수행(push형 또는 pull형) : 위치 기반 자동

// 푸시(push)와 풀(pull) - 밀당(밀고/당기고)

- 푸시(push)와 풀(pull)은 경영 용어로 마케팅과 광고 영역에서 시작되었다.
- 푸시(push) 시스템 : 소비자는 상품 생산을 요청하지 않는다.
  → 생산과 분배의 결정은 오랜 기간의 예측에 기반한다.
- 풀(pull) 시스템 : 고객은 상품 생산을 요청한다.
  → 생산과 분배는 요청에 의해 조정된다.
  → 예 : 자동차를 고객에게 주문 받을 때에만 생산한다.

// 푸시(push) 기법 / 풀(pull) 기법

- 푸시 기법은 사용자가 정보를 일일이 요청하지 않아도 자동으로 사용자에게 정보를 제공하는 것이다.
- 풀 기법은 사용자가 원하는 정보를 직접 찾는 것으로 요청이 사용자에서 시작되는 정보 전달 방식이다.

// 푸시 서버(push server)

- 푸시(push)는 특정 정보를 밀어내 준다는 의미이다.
- 웹서버의 정보 중에서 사용자에게 원하는 정보를 밀어내 준다.(푸시 서버)
- **예** : 은행에서 고객의 입출금 내역을 스마트폰 푸시 알림으로 알려준다.
- 모바일기기가 등장하면서 기존 웹사이트를 통해 이용하던 서비스의 알림을 실시간으로 쉽고 편리하게 받을 수 있게 됐다.
- **예** : 온라인 서점이나 마켓 등 웹사이트에서 구매한 제품에 대한 정보와 포털사이트 메일 수신, 카페 공지 등 서비스에 대한 알림 메시지가 실시간으로 모바일기기로 전송된다.
- 메시지를 받은 사용자는 웹사이트에 접속, 로그인 등 절차를 밟지 않아도 웹사이트 내에서 이뤄지고 있는 상황을 쉽게 파악할 수 있게 된다.

// TSQL2 언어(temporal query language)

- TSQL2는 시간적 확장 언어이다.
- 시간 데이터 지원을 향상시키기 위해 SQL-92의 확장으로 TSQL2가 제안되었다.

정답 : ③

5. SQL 질의문은 동치인 관계대수식으로 변환할 수 있다. 다음 SQL 질의문을 관계대수식으로 변환한 것 중 옳은 것은? (단, 제품 테이블 P, 판매 테이블 S, 집계함수 표현을 위한 관계대수 연산자 기호는 F로 한다) [2018년 국가 7급]

스키마	· S(sId, sDate, pId, sAmount) · P(pId, pName)
SQL	SELECT SUM(sAmount) FROM P, S WHERE S.pId = P.pId AND P.pName='P1' GROUP BY sDate;

① $F_{SUM(sAmount)}(\pi_{P.pName}(\sigma_{P.pName='P1'} \bowtie_{S.pId=P.pId} P))$

② ${}_{sDate}F_{SUM(sAmount)}(\pi_{S.sAmount}(\sigma_{P.pName='P1'}(S \bowtie_{S.pId=P.pId} P)))$

③ $F_{SUM(sAmount)}(S \bowtie_{S.pId=P.pId}(\sigma_{P.pName='P1'}(P)))$

④ ${}_{sDate}F_{SUM(sAmount)}(\sigma_{P.pName='P1'}(P) \bowtie_{P.pId=S.pId} S)$

♤ SQL 질의문은 동치인 관계대수식으로 변환

// Schema 구조

제품 테이블 P		판매 테이블 S			
pId(제품번호)	pName(제품명)	sId(판매번호)	pId(제품번호)	sDate(판매일)	sAmount(판매량)
1	P1	100	1	8월 1일	1000
2	P2	200	1	8월 1일	2000
3	P3	300	2	8월 2일	3000

· ${}_{sDate}F_{SUM(sAmount)}(\sigma_{P.pName='P1'}(P) \bowtie_{P.pId=S.pId} S)$

↓ 속성 sDate를 기준으로 그룹화

↳ 제품 테이블 P와 판매 테이블 S를 조인

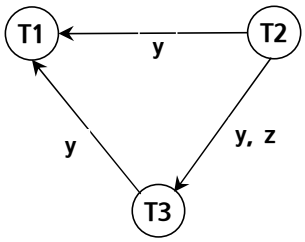
↱ 속성 sDate별 그룹화에 대한 sAmount의 총합을 구함

// 그룹화한 집계함수에 대한 관계대수 표현

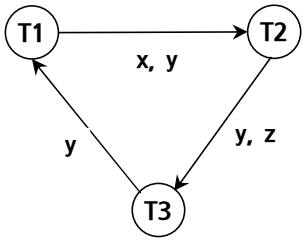
질의문	· 직원 테이블에 대한 각 부서별 급여 합을 구하라?
SQL	· Select Sum(급여) From 직원 Group By 부서;
관계대수	· ${}_{부서}F_{sum(급여)}$ (직원) ↓ 그룹화할 **속성(부서)**을 F 앞에 기술한다. · ${}_{부서}F_{sum(급여)}$ (직원) : Select Sum(급여)를 의미함 · 즉, $\Pi_{Sum(급여)}({}_{부서}F_{sum(급여)}$ (직원))처럼 **복잡**하게 표현하지 않지만 이런 의미이다.

정답 : ④

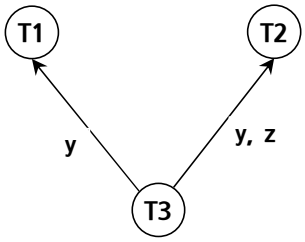
6. 다음 트랜잭션의 선행그래프 및 그와 동치인 직렬 가능한 스케줄의 쌍들이다. 이 쌍들 중 옳지 않은 것은? [2018년 국가 7급]



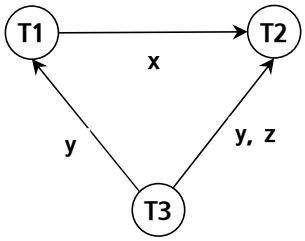
① 동치 직렬 스케줄 : T2→T3→T1



② 동치 직렬 스케줄 : T3→T1→T2



③ 동치 직렬 스케줄 : T3→T2→T1



④ 동치 직렬 스케줄 : T3→T1→T2

♤ **직렬 가능한 스케줄**

// 직렬 가능한 스케줄은 선행그래프에서

- 사이클이 없으면 : 직렬 가능
- 사이클이 있으면 : 직렬 불가능

- **항목 ②**의 선행그래프에는 사이클이 있다. 즉, **직렬 불가능 스케줄**이다.

정답 : ②

7. 다음 '사원' 테이블을 생성하는 SQL문에서 부서의 속성값을 '인사', '연구', '영업'으로 제한하고자 한다. ㉠, ㉡에 들어갈 내용으로 옳은 것은? [2018년 국가 7급]

```
CREATE TABLE 사원(
  사번 NUMBER NOT NULL,
  이름 CHAR(10),
  직급 CHAR(10),
  부서 CHAR(10)  ㉠  (부서  ㉡  ('인사', '연구', '영업')));
```

	㉠	㉡
①	UNIQUE	ON
②	CHECK	ON
③	UNIQUE	IN
④	CHECK	IN

♤ **제한조건 : CHECK ~ IN**

- 부서 CHAR(10) CHECK (부서 IN ('인사', '연구', '영업'))
  → 부서의 속성값을 '인사', '연구', '영업'으로 제한한다.
- 등급 CHAR(1) CHECK (등급 IN ('A', 'B', 'C', 'D'))
  → 등급의 속성값을 삽입할 때, 문자 A, B, C, D가 아닌 문자를 넣으면 오류 발생

// Unique

- Unique은 유일한 값으로 존재해야 하는 것을 의미한다.(중복 배제)
  → NULL 값에 대해서는 Unique 제약이 적용되지 않는다.(다수의 NULL 값 존재 가능)
  → NULL 값은 데이터로 인식하지 않는 원리이다.
- Unique 칼럼은 NULL 또는 유일한 값을 가질 수 있다.
- Unique 칼럼은 유일성을 보장한다.
- Unique 칼럼은 Primary Key로 인식되는 것은 아니다.
- Unique 칼럼으로 참조 무결성 지정은 불가능하다.
- 하나의 테이블에 Unique 칼럼은 여러 개 존재할 수 있다.
  → 하나의 테이블에 기본키는 하나만 존재

- Primary Key = Not Null + Uinque

정답 : ④

8. 2개의 계좌 A1, A2의 잔액이 각각 2,000,000원, 3,000,000원이라고 하자. 이 상태에서 각 계좌의 관리를 수행하는 트랜잭션 T1 및 T2의 실행이 다음과 같을 때, 발생한 현상으로 옳은 것은?

---

1 : T2가 계좌 A2에 2,500,000원을 더해 5,500,000원이 됨
2 : T1이 계좌 A1에 1,500,000원을 더해 3,500,000원이 됨
3 : T2가 계좌 A1의 잔액(3,500,000원)이 충분하다고 판단하여 A1에서 A2로 2,500,000원 이체를 실행함
4 : T1이 알 수 없는 이유로 트랜잭션의 수행이 실패함

---

① 오손판독(dirty read)
② 반복할 수 없는 판독(non-repeatable read)
③ 부정확한 요약(incorrect summary)
④ 유령판독(phantom read)

♤ **오손판독(dirty read)**

// 주어진 내용을 분석하면

T1	T2

	read(A2);
	A2 = A2 + 2,500,000;
	write(A2);

read(A1);
A1 = A1 + 1,500,000;
write(A1);	

	read(A1);
	A1 = A1 - 2,500,000;
	write(A1);
	read(A2);
	A2 = A2 + 2,500,000;
	write(A2);

rollback(복귀)	

- 트랜잭션 T1이 갱신 기록한 데이터 A1 값을 T2가 읽고 갱신 기록하였다.
- 트랜잭션 T1이 최종적으로 정상 완료되지 않고 철회(rollback, 복귀)되었다.
  → 잘못된 데이터를 읽은 트랜잭션 T2도 당연히 복귀되어야 한다.(오손판독, 연쇄복귀)
- 오손판독은 완료되지 않은 트랜잭션에서 쓴(write) 데이터를 다른 트랜잭션에서 읽는 것

정답 : ①

9. 로킹(locking)을 이용한 DBMS 동시성 제어기법의 설명으로 옳은 것은? [2018년 국가 7급]

① 2단계로킹프로토콜(two-phase locking protocol)의 확대단계에서는 로크를 해제할 수 없지만, 축소 단계에는 로크를 해제한 후 같은 항목에 대하여 다시 로크를 획득할 수 있다.

② 로킹 기법을 사용하여 트랜잭션의 동시성을 제어하면 항상 직렬가능(serializable)한 스케줄을 보장한다.

③ 다중단위로킹(multiple granularity locking) 프로토콜에서 SIX(intention-shared-exclusive) 로크가 걸려있는 노드에 IS(intention-shared) 로크를 추가로 걸 수 있다.

④ 2단계로킹프로토콜을 활용하면 불필요한 교착상태(deadlock)를 회피할 수 있어서 트랜잭션의 철회(rollback)를 막을 수 있다.

♤ **로킹을 이용한 DBMS 동시성 제어기법**

① 2단계로킹프로토콜의 확대단계에서는 로크를 해제할 수 없지만, 축소단계에는 로크를 해제한 후 같은 항목에 대하여 다시 로크를 획득할 수 있다.**(×)**
→ **확장단계** : 트랜잭션은 lock만 수행할 수 있다.(unlock은 수행할 수 없다)
→ **수축단계** : 트랜잭션은 unlock만 수행할 수 있다.(lock은 수행할 수 없다)

② 로킹 기법을 사용하여 트랜잭션의 동시성을 제어하면 **항상 직렬가능**(serializable)한 스케줄을 보장한다.**(×)** → **단순 로킹 기법**만으로는 **직렬가능**한 스케줄을 보장하지 못한다.

③ 다중단위로킹 프로토콜에서 **SIX**(intention-shared-exclusive) 로크가 걸려있는 노드에 **IS**(intention-shared) 로크를 추가로 걸 수 있다.(○)

④ 2단계로킹프로토콜을 활용하면 불필요한 **교착상태**(deadlock)를 회피할 수 있어서 트랜잭션의 철회(rollback)를 막을 수 있다.**(×)** → 2단계로킹규약은 **교착상태 발생** 가능성은 있다.
→ 2단계로킹규약은 **직렬가능성**을 **보장**할 수 있는 규약이다.

// 다중단위로킹에서 로크 양립성 행렬 (호환성 행렬)

추가 잠금 요청 / 현재 잠금 상태	IS	IX	S	SIX	X
잠금 없음	**가능**	**가능**	**가능**	**가능**	**가능**
배타(X) 잠금	불가	불가	불가	불가	불가
공유(S) 잠금	**가능**	불가	**가능**	불가	불가
의도-공유(IS) 잠금	**가능**	**가능**	**가능**	**가능**	불가
의도-배타(IX) 잠금	**가능**	**가능**	불가	불가	불가
공유-의도-배타(SIX) 잠금	**가능**	불가	불가	불가	불가

- SIX 로크가 걸려있는 노드에 추가로 IS 로크를 걸 수 **있다. – 로크 중복 가능**
- SIX 로크가 걸려있는 노드에 추가로 IX 로크를 걸 수 **없다.**라는 의미이다.

// 의도로크 – 3가지 형태

의도-공유 **(IS 로크)**	앞으로, 자식노드들에게 **공유로크**를 걸겠다는 의미이다.
의도-배타 **(IX 로크)**	앞으로, 자식노드들에게 **배타로크**나 **공유로크**를 걸겠다는 의미이다.
공유-의도-배타 **(SIX 로크)**	현재 특정 **노드**를 **루트**로 하는 서브트리가 **공유로크**가 걸려 있는데, 앞으로, 자식노드들에게 **배타로크**로 변경하겠다는 의미한다.

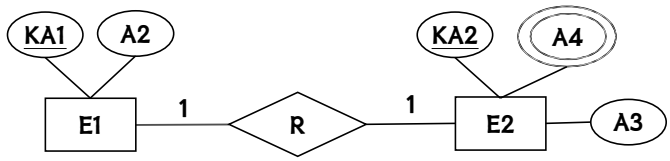
- **의도로크**는 앞으로 자식노드들에게 **로크**를 요구할 것이라는 것을 나타낸다.
- 어떤 노드에 **의도로크**가 걸리면, 자식노드 중에서 명시적으로 로크가 걸린 것을 의미한다.
- 즉, **의도로크**는 부모노드에 로크를 걸면, 자식노드에 **묵시적**으로 로크를 건 효과가 있다.

// 병행제어(동시성제어) 정리

구분	내용	직렬성 보장	교착상태 방지	연쇄복귀 방지
잠금 **(locking)**	• 공유-잠금 방식(S, shared mode) • 배타-잠금 방식(X, exclusive mode)	×	×	×
기본 2PL **(basic 2PL)**	• 확장단계 : lock 수행 단계 • 수축단계 : unlock 수행 단계 • **교착상태/연쇄복귀 문제가 발생**한다.	○	×	×
엄밀 2PL **엄격 2PL** **(strict 2PL)**	• **연쇄복귀 문제가 발생되지 않는다.** • 트랜잭션이 완료할 때까지 **모든 배타-잠금**을 유지 • 즉, 트랜잭션 완료까지 **모든 x-lock**을 **unlock** 없이 유지	○	×	○
엄중 2PL **(rigorous 2PL)**	• 트랜잭션이 완료할 때까지 **모든 잠금(공유, 배타)**을 유지 • 즉, 트랜잭션 완료까지 **모든** lock은 unlock 없이 유지 • 트랜잭션은 종료까지 확장단계에 있다. • 엄중 2PL은 엄밀 2PL보다 더 **제한적**이다.	○	×	○
정적 2PL **(static 2PL)** **또는** **보수적 2PL** **(conservative)**	• 필요한 모든 읽기집합과 쓰기집합을 미리 선언한다. • 만약, 미리 선언한 데이터 중에 하나라도 로크를 획득할 수 없으면 트랜잭션은 로크를 획득할 수 있을 때까지 대기한다.(봉쇄) • 일단, 트랜잭션이 시작되면 축소단계에 있게 된다. • 읽기, 쓰기 집합을 미리 선언해야 하므로 현실적으로 적용하기가 어렵다.(비현실적)	○	○	○
타임스탬프	• 타임스탬프를 이용한다. – **타임스탬프 순서 기법** • 타임스탬프 순으로 트랜잭션들의 수행순서를 **미리** 결정	○	○	×
낙관적 **(optimistic)**	• 낙관적 병행제어는 일단 트랜잭션을 실행하겠다는 것! • 트랜잭션이 실행되는 동안은 어떤 검사도 실시하지 않음 • 트랜잭션이 실행 종료한 후에 직렬성 위배 여부 조사 • 3단계 : 판독, 확인, 기록 – 타임스탬프 이용	○	○	○

정답 : ③

10. 다음 ERD(entity-relationship diagram)를 관계형 데이터베이스 스키마로 변환한 것으로 옳지 않은 것은?(단, 밑줄은 기본키이다) [2018년 국가 7급]



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) R(KA1, KA2) E21(KA2, A4)
④ E1(KA1, A2, KA2) E2(KA2, A3, A4)

♤ **ERD를 관계형 데이터베이스 스키마로 변환**

④ E1(KA1, A2, KA2) E2(KA2, A3, A4) **(×)**
→ ERD에서 이중원은 다중 값을 갖는 속성을 의미한다.
→ 다중 값을 갖는 속성이 있으면 새로운 스키마를 별도로 만들어야 한다.
→ 원래의 스키마에서는 다중값속성은 빠진다.
→ 생성된 스키마의 기본키는 {개체의 기본키, 다중값속성}으로 구성한다. E21(KA2, A4)

다중 값 속성	• 다중 값을 갖는 속성은 **이중원**으로 표시한다. • 다중 값 속성은 여러 개의 속성 값을 가지는 속성이다.(집합이나 리스트 형태) • 예를 들면, 한 사람은 여러 개의 전화를 가질 수 있다.

// 다음과 같은 스키마 구조를 가진다.(1대1의 관계)

E1(신랑)	
KA1(주민번호)	A2(이름)
11	갑돌이
12	김중배
13	견우

E2(신부)		
KA2(주민번호)	A3(이름)	KA1(주민번호)
21	갑순이	11
22	심순애	12
23	직녀	13

E21(신부_1)	
KA2(주민번호)	A4(차번호)
21	1000
21	1001
:	:

정답 : ④