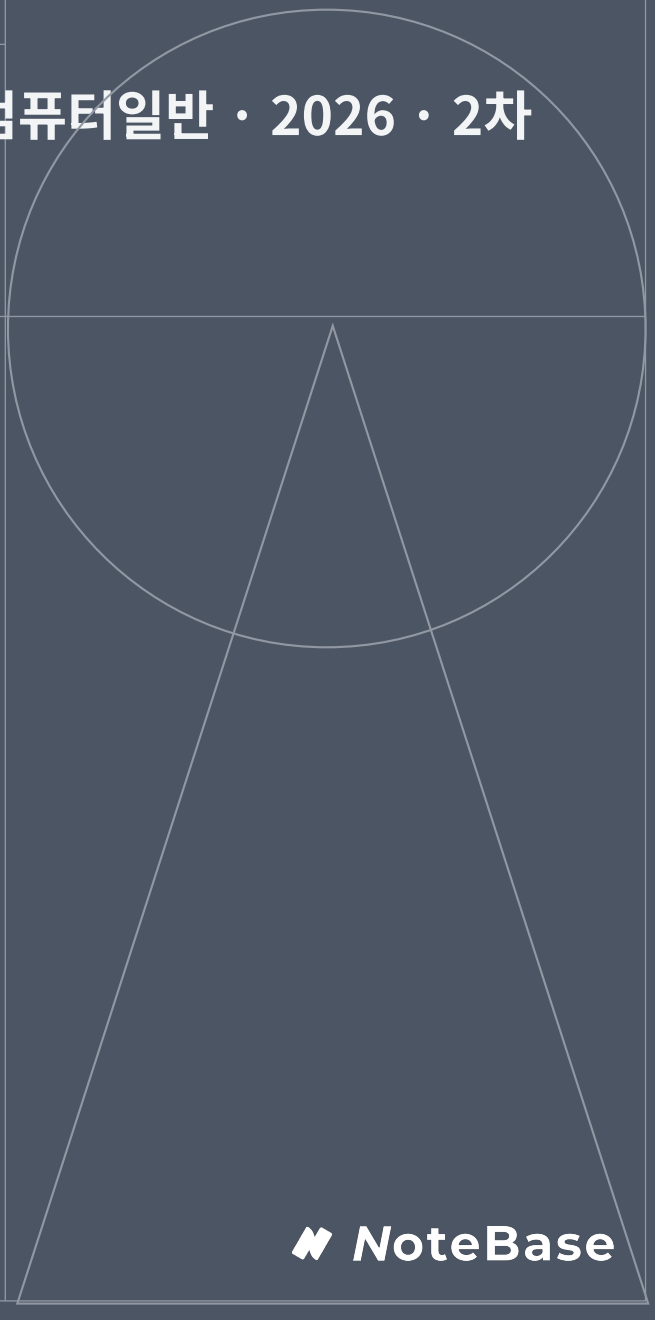
기출OX 해설집

공무원/해경/일반직 9급 · 컴퓨터일반 · 2026 · 2차



NoteBase

기출OX 해설집

공무원/해경/일반직 9급 · 컴퓨터일반 · 2026 · 2차

공무원/해경/일반직 9급 · 컴퓨터일반 · 2026 · 2차

문 1. 다음 중 클라우드에 관련된 내용으로 가장 옳은 것은?

① 민간 클라우드는 상용 서비스만 가능하며 지방 정부 등 공공영역의 서비스에 이용할 수 없다.

② 클라우드에서 시스템을 임차할 경우 IaaS, 소프트웨어 서비스를 임차할 경우 SaaS라 한다.

③ 클라우드에서 웹서비스를 구현할 경우 서버의 저장 데이터량에 따라 과금되며, 서버의 데이터 전송량에 따르는 과금 차이를 두지 않는다.

④ 클라우드는 일반적으로 물리적 시스템 단위로 임차 서비스를 제공한다.

정답 2번

출제 포인트

이 문제는 클라우드의 이용 범위, 서비스 모델, 과금 방식, 자원 제공 단위를 구별하는 문제이다.

핵심 개념

클라우드 컴퓨팅은 서버, 저장장치, 네트워크, 소프트웨어 등의 정보기술 자원을 인터넷을 통해 필요한 만큼 빌려 쓰는 방식이다.
IaaS는 서버·저장장치·네트워크 같은 인프라를 서비스로 제공하고, SaaS는 완성된 응용 소프트웨어를 서비스로 제공한다. 클라우드 자원은 가상화·자원 공유를 바탕으로 탄력적으로 할당되며, 요금은 저장 용량뿐 아니라 사용 시간, 처리량, 데이터 전송량 등 서비스별 측정 항목에 따라 달라질 수 있다.

NoteBase

선지별 해설

① X
틀린 표현은 '공공영역의 서비스에 이용할 수 없다'이다. 민간 사업자가 제공하는 클라우드도 요구되는 보안·인증 등의 조건을 충족하면 지방 정부를 포함한 공공영역에서 이용할 수 있다.

② O
서버·저장장치·네트워크 등 시스템 인프라를 임차하는 모델은 IaaS이고, 응용 소프트웨어를 서비스 형태로 임차하는 모델은 SaaS이므로 옳다.

③ X
틀린 표현은 '데이터 전송량에 따르는 과금 차이를 두지 않는다'이다. 클라우드 요금은 저장 데이터량뿐 아니라 데이터 전송량과 전송 방향 등에도 따라 달라질 수 있다.

④ X
틀린 표현은 '물리적 시스템 단위'이다. 클라우드는 일반적으로 가상화된 컴퓨팅 자원을 논리적 단위로 나누어 필요에 따라 제공한다.

오답 포인트

①의 '이용할 수 없다', ③의 '과금 차이를 두지 않는다', ④의 '물리적 시스템 단위'처럼 클라우드의 이용 범위와 운영 방식을 지나치게 단정한 표현이 함정이다. 특히 IaaS의 '시스템'은 물리 장비 한 대를 통째로 빌린다는 뜻이 아니라 가상화된 인프라 자원을 서비스로 이용한다는 뜻이다.

다시 나오면 이렇게 판단

무엇을 빌리는지를 기준으로 서비스 모델을 구분한다. 인프라를 빌리면 IaaS, 개발·실행 환경을 빌리면 PaaS, 완성된 소프트웨어를 빌리면 SaaS이다. 클라우드의 핵심 특성은 가상화, 탄력적 자원 할당, 사용량 기반 과금이므로 이를 부정하는 단정적 표현은 틀릴 가능성이 크다.

암기 포인트

IaaS는 인프라, PaaS는 개발 플랫폼, SaaS는 소프트웨어이며, 클라우드는 가상 자원을 필요한 만큼 사용하고 사용량에 따라 비용을 낸다.

NoteBase

문 2. 다음 중 플립플롭(Flip-Flop)을 사용하는 회로를 모두 고르면?

<보기>
㉠ 레지스터(Register)
㉡ 디코더(Decoder)
㉢ 무어 머신(Moore Machine)
㉣ 동기식 카운터(Synchronous Counter)

① ㉠, ㉡, ㉢
② ㉡, ㉢
③ ㉠, ㉢, ㉣
④ ㉠, ㉣

정답 3번

출제 포인트

플립플롭이 필요한 순차논리회로와 플립플롭을 사용하지 않는 조합논리회로를 구별하는 문제이다.

핵심 개념

플립플롭은 1비트의 상태를 저장하는 기억소자이다.
레지스터는 여러 비트를 저장하기 위해 여러 플립플롭을 묶은 순차논리회로이다.
디코더는 입력 코드에 따라 특정 출력선을 선택하는 조합논리회로이므로 기본적으로 플립플롭을 사용하지 않는다.
무어 머신은 현재 상태만으로 출력을 결정하며, 현재 상태를 저장하는 상태 레지스터에 플립플롭을 사용한다.
동기식 카운터는 모든 플립플롭이 같은 클록에 맞추어 상태를 바꾸는 순차논리회로이다.

선지별 해설

① X — ㉠ 레지스터와 ㉢ 무어 머신은 플립플롭을 사용하지만, ㉡ 디코더는 기억 기능이 없는 조합논리회로이다. 또한 플립플롭을 사용하는 ㉣ 동기식 카운터가 빠져 있다. 올바른 조합은 ㉠, ㉢, ㉣이다.
② X — ㉢ 무어 머신은 플립플롭을 사용하지만, ㉡ 디코더는 조합논리회로이므로 플립플롭을 사용하지 않는다. 또한 ㉠ 레지스터와 ㉣ 동기식 카운터가 빠져 있다. 올바른 조합은 ㉠, ㉢, ㉣이다.
③ O — ㉠ 레지스터는 데이터 저장에, ㉢ 무어 머신은 상태 저장에, ㉣ 동기식 카운터는 계수 상태 저장에 플립플롭을 사용한다.
④ X — ㉠ 레지스터와 ㉣ 동기식 카운터는 플립플롭을 사용하지만, 상태를 저장해야 하는 ㉢ 무어 머신이 빠져 있다. 올바른 조합은 ㉠, ㉢, ㉣이다.

NoteBase

오답 포인트

디코더도 디지털 회로라는 이유만으로 플립플롭을 사용한다고 판단하기 쉽다. 그러나 디코더는 현재 입력만으로 출력이 정해지는 조합논리회로이고, 레지스터·무어 머신·동기식 카운터는 저장된 상태가 필요한 순차논리회로이다.

다시 나오면 이렇게 판단

회로가 이전 값이나 현재 상태를 기억해야 하는지를 확인한다. 상태 저장이 필요하면 플립플롭을 사용하는 순차논리회로이고, 현재 입력만으로 출력이 정해지면 조합논리회로이다.

암기 포인트

저장·상태·계수는 플립플롭, 입력 해독은 조합논리이다.

NoteBase

문 3. 다음 중 프로그램 카운터(Program Counter)의 역할에 대한 설명으로 가장 옳지 않은 것은?

① 다음에 실행할 명령어의 메모리 주소를 보관한다.

② 명령어를 인출(Fetch)한 후에는 일반적으로 값이 자동으로 증가한다.

③ 분기 명령어(Jump, Branch)가 실행되면 해당 목적지 주소로 값이 변경된다.

④ 현재 CPU에서 실행 중인 명령어의 실제 내용을 저장하고 있다.

정답 4번

출제 포인트

프로그램 카운터와 명령어 레지스터의 역할을 구별하는 문제이다.

핵심 개념

프로그램 카운터(PC)는 다음에 인출할 명령어의 메모리 주소를 보관하는 레지스터이다. 명령어 레지스터(IR)는 메모리에서 인출되어 현재 해독·실행 중인 명령어 자체를 보관한다.

선지별 해설

① O — 프로그램 카운터는 다음에 실행하기 위해 인출할 명령어의 메모리 주소를 보관한다.
② O — 명령어를 인출한 뒤에는 일반적으로 다음 순차 명령어를 가리키도록 프로그램 카운터 값이 명령어 길이만큼 자동 증가한다.
③ O — 분기 명령어의 분기 조건이 성립하거나 무조건 점프가 실행되면 프로그램 카운터는 분기 목적지 주소로 변경된다.
④ X — 틀린 표현은 '현재 CPU에서 실행 중인 명령어의 실제 내용을 저장'한다는 부분이다. 프로그램 카운터는 명령어의 주소를 저장하며, 인출된 명령어의 실제 내용은 명령어 레지스터에 저장된다.

오답 포인트

④는 프로그램 카운터와 명령어 레지스터를 바꾸어 설명한 선지이다. '주소'는 프로그램 카운터, '명령어 내용'은 명령어 레지스터라는 구분이 핵심이다.

다시 나오면 이렇게 판단

레지스터가 저장하는 대상부터 확인한다. 다음 명령어의 주소이면 프로그램 카운터이고, 인출된 명령어 자체이면 명령어 레지스터이다.

암기 포인트

PC는 다음 주소, IR은 현재 명령어이다.

NoteBase

문 4. 다음 <보기>에서 옳은 것을 모두 고르면?

<보기>
㉠ SRAM(Static RAM)은 전원이 공급되는 동안에는 데이터가 유지되므로 리프레시(Refresh) 동작이 필요 없다.
㉡ DRAM(Dynamic RAM)은 커패시터(Capacitor)에 전하를 저장한다.
㉢ SRAM은 DRAM보다 집적도가 높고 구조가 단순하다.
㉣ SRAM은 일반적으로 DRAM보다 속도가 매우 빠르다.

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

정답 3번

출제 포인트

이 문제는 SRAM과 DRAM의 저장 방식, 리프레시 필요 여부, 집적도 및 속도를 비교하는 문제이다.

핵심 개념

SRAM: 플립플롭 회로에 데이터를 저장하는 휘발성 메모리이다. 전원이 공급되는 동안 데이터가 유지되어 리프레시가 필요 없고, 빠르지만 셀 구조가 복잡하여 집적도가 낮다.
DRAM: 커패시터의 전하로 데이터를 저장하는 휘발성 메모리이다. 전하가 누설되므로 주기적인 리프레시가 필요하고, SRAM보다 느리지만 셀 구조가 단순하여 집적도가 높다.

NoteBase

선지별 해설

① X
㉠과 ㉡은 옳지만 ㉢은 틀리고, 옳은 ㉣이 빠져 있다. ㉢의 'SRAM은 DRAM보다 집적도가 높고 구조가 단순하다'가 잘못된 표현이다. 실제로는 SRAM이 DRAM보다 집적도가 낮고 셀 구조가 복잡하므로 옳은 조합은 ㉠, ㉡, ㉣이다.

② X
㉡과 ㉣은 옳지만 ㉢은 틀리고, 옳은 ㉠이 빠져 있다. SRAM은 전원이 공급되는 동안 데이터를 유지하므로 리프레시가 필요 없고, DRAM보다 집적도가 낮으며 구조가 복잡하다. 따라서 옳은 조합은 ㉠, ㉡, ㉣이다.

③ O
㉠은 SRAM에 리프레시가 필요 없다는 설명, ㉡은 DRAM이 커패시터에 전하를 저장한다는 설명, ㉣은 SRAM이 일반적으로 DRAM보다 빠르다는 설명으로 모두 옳다.

④ X
㉠과 ㉣은 옳지만 ㉢은 틀리고, 옳은 ㉡이 빠져 있다. ㉢은 '집적도가 높고 구조가 단순하다'를 SRAM이 아니라 DRAM의 특징으로 고쳐야 한다. 따라서 옳은 조합은 ㉠, ㉡, ㉣이다.

오답 포인트

㉢은 SRAM과 DRAM의 집적도와 구조적 특징을 서로 바꾸어 제시한 함정이다. SRAM은 속도가 빠른 대신 셀 구조가 복잡하고 집적도가 낮으며, DRAM은 상대적으로 느린 대신 셀 구조가 단순하고 집적도가 높다.

다시 나오면 이렇게 판단

저장 소자부터 구분한다. SRAM은 플립플롭을 사용하므로 리프레시가 필요 없고 빠르지만 저집적이며, DRAM은 커패시터를 사용하므로 리프레시가 필요하고 상대적으로 느리지만 고집적이다.

암기 포인트

SRAM은 빠름·무리프레시·저집적, DRAM은 커패시터·리프레시·고집적이다.

NoteBase

문 5. 다음은 캐시 메모리의 주소 지정 방식에 대한 설명이다. 주소 지정 방식이 가장 다른 하나는?

① 메인 메모리의 블록마다 캐시에 저장될 수 있는 위치가 한 곳으로 정해져 있어 빠르게 접근된다.

② 데이터를 찾기 위해 캐시 내 모든 라인의 태그(Tag)를 비교하여야 한다.

③ 충돌 미스(Conflict Miss)가 적어 캐시 히트율을 높이는 데 유리하다.

④ 메인 메모리의 블록을 캐시 내의 어느 라인(Line)에나 자유롭게 배치할 수 있다.

정답 1번

출제 포인트

직접 사상과 완전 연관 사상을 캐시 블록의 배치 가능 위치, 태그 비교 범위, 충돌 미스의 차이로 구별하는 문제이다.

핵심 개념

직접 사상은 메인 메모리의 각 블록이 들어갈 캐시 라인이 하나로 정해지는 방식이다. 구조가 단순하고 탐색이 빠르지만 같은 라인에 대응하는 블록끼리 충돌하기 쉽다.
완전 연관 사상은 메인 메모리의 블록을 캐시의 어느 라인에나 배치할 수 있는 방식이다. 배치가 자유로워 충돌 미스는 줄지만, 데이터를 찾을 때 모든 캐시 라인의 태그를 비교해야 한다.

선지별 해설

① O — 블록마다 들어갈 캐시 위치가 한 곳으로 정해진다는 설명은 직접 사상의 특징이다. 나머지 선지는 완전 연관 사상을 설명하므로 가장 다른 선지이다.
② X — 캐시의 모든 라인에 있는 태그를 비교한다는 것은 완전 연관 사상의 특징이다. '가장 다른 방식'의 설명이 아니며, 올바르게 분류하면 완전 연관 사상이다.
③ X — 완전 연관 사상은 블록을 어느 라인에나 둘 수 있어 특정 라인으로의 배치 충돌이 적고 캐시 히트율을 높이는 데 유리하다. '가장 다른 방식'의 설명이 아니며, 올바르게 분류하면 완전 연관 사상이다.
④ X — 메인 메모리 블록을 캐시의 어느 라인에나 자유롭게 배치할 수 있다는 것은 완전 연관 사상의 정의이다. '가장 다른 방식'의 설명이 아니며, 올바르게 분류하면 완전 연관 사상이다.

오답 포인트

①의 '위치가 한 곳으로 정해져'는 직접 사상의 결정적 표현이다. 반면 ②의 '모든 라인의 태그 비교', ③의 '충돌 미스가 적음', ④의 '어느 라인에나 배치'는 모두 완전 연관 사상을 가리킨다. 장단점만 따로 외우면 ②와 ③을 다른 방식으로 오인하기 쉽다.

NoteBase

다시 나오면 이렇게 판단

배치 가능 위치를 먼저 확인한다. 한 곳으로 고정되면 직접 사상이고, 모든 라인에 배치할 수 있으면 완전 연관 사상이다. 이어서 직접 사상은 해당 라인의 태그만 비교하고 충돌이 많으며, 완전 연관 사상은 모든 태그를 비교하고 충돌이 적다는 점으로 검증한다.

암기 포인트

직접 사상은 한 블록·한 자리, 완전 연관 사상은 한 블록·모든 자리이다.

NoteBase

문 6. 다음 <보기>에서 NAND 플래시 메모리(NAND Flash Memory)가 사용되는 장치를 모두 고르면?

<보기>
㉠ SSD(Solid State Drive) 저장장치
㉡ USB 메모리 스틱(USB Flash Drive)
㉢ 스마트폰에 내장된 UFS/eMMC 등 내부 저장소
㉣ 마이크로프로세서 내부의 캐시 메모리

① ㉠
② ㉠, ㉡
③ ㉠, ㉡, ㉢
④ ㉠, ㉡, ㉢, ㉣

정답 3번

출제 포인트

NAND 플래시 메모리를 비휘발성 대용량 저장장치에 사용하는 경우와, SRAM을 고속 임시 저장장치에 사용하는 경우를 구별하는 문제이다.

핵심 개념

NAND 플래시 메모리는 전원이 꺼져도 데이터를 유지하는 비휘발성 메모리로, SSD·USB 플래시 드라이브·스마트폰의 UFS/eMMC 저장소 등에 사용된다.
마이크로프로세서 내부의 캐시 메모리는 처리 속도가 매우 빨라야 하므로 일반적으로 SRAM을 사용한다.

NoteBase

선지별 해설

① X
㉠의 SSD에는 NAND 플래시 메모리가 사용되지만 ㉡의 USB 메모리 스틱과 ㉢의 스마트폰 내부 저장소도 NAND 플래시 기반이다. 틀린 부분은 대상이 ㉠뿐이라는 점이며, 올바른 조합은 ㉠, ㉡, ㉢이다.

② X
㉠의 SSD와 ㉡의 USB 메모리 스틱은 모두 NAND 플래시를 사용하지만 ㉢도 포함해야 한다. 틀린 부분은 ㉢을 제외한 것이며, 올바른 조합은 ㉠, ㉡, ㉢이다.

③ O
㉠ SSD, ㉡ USB 메모리 스틱, ㉢ 스마트폰의 UFS/eMMC 내부 저장소는 모두 NAND 플래시 메모리를 저장 매체로 사용한다.

④ X
㉠, ㉡, ㉢은 맞지만 ㉣ 마이크로프로세서 내부의 캐시 메모리는 일반적으로 SRAM을 사용한다. 틀린 부분은 ㉣을 포함한 것이며, 올바른 조합은 ㉠, ㉡, ㉢이다.

오답 포인트

스마트폰의 UFS/eMMC도 이름에 '플래시'가 직접 드러나지 않지만 NAND 플래시 기반 저장장치이다. 반면 캐시는 메모리라는 공통점만 보고 포함하기 쉬우나, 비휘발성 대용량 저장보다 빠른 접근이 중요하여 일반적으로 SRAM을 사용한다.

다시 나오면 이렇게 판단

전원이 꺼져도 데이터를 보존하는 SSD·USB·스마트폰 내부 저장소는 NAND 플래시로 분류하고, CPU에 가까운 고속 임시 저장공간인 캐시는 SRAM으로 분류한다.

암기 포인트

SSD·USB·UFS/eMMC는 NAND, CPU 캐시는 SRAM이다.

NoteBase

문 7. 다음 중 컴퓨터 구조의 파이프라인(Pipeline)에 대한 설명으로 가장 옳지 않은 것은?

① 파이프라인은 한 개의 명령어가 실행되는데 걸리는 전체 실행 시간을 단축시킨다.

② 파이프라인으로 인한 해저드(Hazard)가 발생할 수 있으며, 이는 성능 저하의 원인이 된다.

③ 한 명령어를 단계별로 나누어 여러 명령을 겹쳐 실행함으로써 명령어 처리량(Throughput)을 높인다.

④ 파이프라인을 세분화하여 단계 개수를 늘리면 더 높은 클록 주파수(Clock Frequency)로 동작시킬 수 있다.

정답 1번

출제 포인트

파이프라이닝이 한 명령어의 지연시간을 줄이는 기술인지, 여러 명령어의 처리량을 높이는 기술인지 구분하는 문제이다.

핵심 개념

파이프라이닝은 명령어 처리를 여러 단계로 나누고, 서로 다른 명령어의 각 단계를 겹쳐 실행하는 방식이다. 핵심 효과는 일정 시간 동안 완료하는 명령어 수인 처리량의 증가이다. 한 명령어는 여전히 모든 단계를 거쳐야 하므로 그 명령어 하나의 전체 실행 시간이 원칙적으로 단축되는 것은 아니다.

선지별 해설

① X — 틀린 표현은 '한 개의 명령어가 실행되는데 걸리는 전체 실행 시간을 단축시킨다'이다. 파이프라이닝의 주된 효과는 여러 명령어를 겹쳐 실행하여 처리량을 높이는 것이며, 한 명령어의 전체 지연시간을 줄이는 것이 아니다. 올바르게는 '파이프라이닝은 여러 명령어의 실행을 중첩하여 전체 명령어 처리량을 높인다'이다.
② O — 데이터 의존성, 분기, 자원 충돌 등으로 해저드가 발생하면 파이프라인이 대기하거나 비워질 수 있어 처리량이 감소한다.
③ O — 한 명령어의 처리 과정을 단계별로 나누고 여러 명령어가 서로 다른 단계를 동시에 수행하게 하므로 단위 시간당 완료되는 명령어 수가 증가한다.
④ O — 파이프라인 단계를 더 세분화하면 각 단계의 조합논리 지연을 줄일 수 있으므로 더 짧은 클록 주기, 즉 더 높은 클록 주파수로 동작시킬 수 있다. 다만 실제 향상 폭은 단계 간 불균형과 파이프라인 레지스터 오버헤드 등의 제약을 받는다.

오답 포인트

①의 '한 개의 명령어 실행 시간'과 ③의 '명령어 처리량'을 같은 의미로 보는 것이 함정이다. 파이프라이닝은 개별 명령어를 더 빨리 끝내는 기술이라기보다 여러 명령어를 겹쳐 더 많이 끝내는 기술이다.

NoteBase

다시 나오면 이렇게 판단

파이프라인의 효과를 묻는 선지에서는 지연시간과 처리량을 먼저 구분한다. ‘단일 명령어의 전체 실행시간 단축’은 원칙적으로 틀리고, ‘여러 명령어의 중첩 실행과 처리량 증가’는 맞는 설명이다. 해저드는 파이프라인의 성능을 떨어뜨리는 요인이다.

암기 포인트

파이프라인은 한 명령어의 지연시간 단축이 아니라 여러 명령어의 처리량 향상이다.

NoteBase

문 8. 다음 중 타입 1(Bare-metal) 과 타입 2(Hosted) 하이퍼바이저(Hypervisor)에 대한 설명으로 가장 옳지 않은 것은?

① 타입 1 하이퍼바이저는 호스트 운영체제 없이 실행되므로, 타입 2보다 오버헤드가 적고 성능이 효율적이다.

② 타입 2 하이퍼바이저는 운영체제 위에서 실행되므로 다양한 시스템 명령을 사용할 수 있어 관리 편의성이 좋다.

③ 타입 1 하이퍼바이저는 일반적으로 기업용 서버나 데이터센터와 같이 높은 신뢰성과 성능이 요구되는 환경에 적절하다.

④ 타입 2 하이퍼바이저는 일반적으로 타입 1보다 보안성이 높으며, 시스템 장애 발생 소지가 적다.

정답 4번

출제 포인트

타입 1과 타입 2 하이퍼바이저를 구분하는 기준인 호스트 운영체제의 유무와 그에 따른 성능·관리·보안 특성을 묻는 문제이다.

핵심 개념

타입 1(Bare-metal) 하이퍼바이저는 물리 하드웨어 위에서 직접 실행되어 가상 머신을 관리하는 소프트웨어이다. 호스트 운영체제를 거치지 않으므로 오버헤드가 작고 성능·안정성·보안성이 중요한 서버 환경에 적합하다.
타입 2(Hosted) 하이퍼바이저는 Windows나 Linux 같은 호스트 운영체제 위에서 일반 응용 프로그램처럼 실행된다. 설치와 사용이 편리하지만 호스트 운영체제 계층을 추가로 거치므로 일반적으로 타입 1보다 오버헤드와 공격 표면, 장애 의존성이 크다.

선지별 해설

① O — 타입 1은 호스트 운영체제 없이 하드웨어 위에서 직접 실행된다. 호스트 운영체제 계층을 거치지 않아 타입 2보다 일반적으로 오버헤드가 작고 성능 효율이 높다.
② O — 타입 2는 기존 운영체제 위에서 실행되므로 호스트 운영체제의 명령, 장치 드라이버, 관리 도구 등 익숙한 기능을 활용할 수 있어 설치와 관리가 편리하다.
③ O — 타입 1은 높은 성능과 신뢰성, 자원 격리가 요구되는 기업용 서버와 데이터센터에서 주로 사용된다.
④ X — 틀린 표현은 '타입 2가 타입 1보다 보안성이 높으며, 시스템 장애 발생 소지가 적다'이다. 타입 2는 호스트 운영체제에 의존하므로 호스트 운영체제의 취약점이나 장애가 가상 머신에도 영향을 줄 수 있다. 올바르게는 일반적으로 타입 1이 타입 2보다 공격 표면과 장애 의존성이 작아 보안성과 안정성 측면에서 유리하다고 해야 한다.

NoteBase

오답 포인트

타입 2의 관리 편의성을 보안성과 안정성의 우위로 바꾸어 생각하기 쉽다. ②의 '관리 편의성'은 장점이지만, ④의 '보안성이 높고 장애 소지가 적다'는 설명은 호스트 운영체제에 대한 추가 의존성을 거꾸로 서술한 것이다.

다시 나오면 이렇게 판단

하이퍼바이저 아래에 호스트 운영체제가 있는지를 먼저 확인한다. 하드웨어에 직접 설치되면 타입 1로서 성능·보안·안정성에 유리하고, 운영체제 위에 설치되면 타입 2로서 설치·사용·관리에 편리하지만 추가 오버헤드와 의존성이 생긴다.

암기 포인트

타입 1은 하드웨어 직결로 성능·안정성 중심, 타입 2는 운영체제 위 설치로 편의성 중심이다.

NoteBase

문 9. 다음 중 기아(Starvation) 현상이 발생할 수 있는 스케줄링 알고리즘들로만 묶인 것은?

① FCFS(First Come First Served), Round Robin

② SJF(Shortest Job First), Priority Scheduling

③ Round Robin, Multi Level Queue

④ FCFS, SRT(Shortest Remaining Time)

정답 2번

출제 포인트

이 문제는 CPU 스케줄링 알고리즘별로 어떤 프로세스가 무기한 대기하는 기아 현상이 발생할 수 있는지를 묻는다.

핵심 개념

기아는 준비 큐에 있는 프로세스가 다른 프로세스에 계속 밀려 CPU를 무기한 할당받지 못하는 현상이다.
SJF는 실행 시간이 짧은 작업을 우선하므로 짧은 작업이 계속 들어오면 긴 작업이 기아 상태에 빠질 수 있다.
Priority Scheduling은 우선순위가 높은 작업을 먼저 처리하므로 높은 우선순위의 작업이 계속 들어오면 낮은 우선순위의 작업에 기아가 발생할 수 있다.
FCFS는 도착 순서대로 처리하고, Round Robin은 준비된 프로세스에 시간 할당량을 순환 배분하므로 표준적인 전제에서는 기아가 발생하지 않는다.

선지별 해설

① X — FCFS는 먼저 도착한 순서대로 처리하고 Round Robin은 각 프로세스에 CPU 시간을 순환 배분하므로 둘 다 표준적인 전제에서는 기아를 일으키는 알고리즘이 아니다. 틀린 묶음이며, 올바르게는 FCFS와 Round Robin은 기아가 일반적으로 발생하지 않는 스케줄링 알고리즘이다.
② O — SJF에서는 긴 작업이 계속 뒤로 밀릴 수 있고, Priority Scheduling에서는 낮은 우선순위의 작업이 계속 뒤로 밀릴 수 있으므로 둘 다 기아가 발생할 수 있다.
③ X — Multi Level Queue는 고정 우선순위 방식으로 큐 사이를 선택하면 낮은 우선순위 큐에 기아가 발생할 수 있지만, Round Robin은 준비된 프로세스에 시간 할당량을 차례로 주므로 기아가 발생하지 않는다. 틀린 표현은 'Round Robin'이며, 이를 'SJF' 또는 'Priority Scheduling'처럼 기아가 가능한 알고리즘으로 바꾸어야 한다.
④ X — SRT는 남은 실행 시간이 짧은 프로세스를 선점하여 처리하므로 긴 프로세스가 계속 밀려 기아 상태에 빠질 수 있지만, FCFS는 도착 순서를 지키므로 기아가 발생하지 않는다. 틀린 표현은 'FCFS'이며, 이를 'Priority Scheduling'처럼 기아가 가능한 알고리즘으로 바꾸어야 한다.

NoteBase

오답 포인트

선점형이라는 이유만으로 기아가 발생한다고 판단하면 Round Robin을 잘못 고르게 된다. 기아 여부는 선점 가능성 자체가 아니라 특정 프로세스가 선택 기준에서 계속 밀릴 수 있는지로 판단한다.

다시 나오면 이렇게 판단

스케줄링 기준이 짧은 실행 시간이나 높은 우선순위처럼 특정 작업을 계속 앞세울 수 있으면 기아 가능성이 있다. 도착 순서나 공평한 순환 배분으로 결국 차례가 돌아오면 기아 가능성이 낮다.

암기 포인트

짧은 작업 우선인 SJF와 높은 우선순위 우선인 Priority Scheduling은 기아 가능, FCFS와 Round Robin은 기아 방지형으로 구분한다.

NoteBase

문 10. 다음 중 DBMS의 데이터 독립성에 대한 설명으로 옳지 않은 것을 모두 고르면?

<보기>
㉠ 데이터베이스 관리자가 데이터를 독점적으로 소유하는 것을 의미한다.
㉡ 각 부서가 서로 다른 DBMS를 사용하여 독립적으로 운영하는 것을 의미한다.
㉢ 네트워크 연결 없이도 데이터를 조회할 수 있는 능력을 말한다.
㉣ 하위 단계(물리적 구조)의 변경이 상위 단계(응용 프로그램)에 영향을 주지 않는 것이다.

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

정답 1번

출제 포인트

DBMS의 데이터 독립성이 하위 단계의 데이터 구조 변경이 상위 단계와 응용 프로그램에 미치는 영향을 줄이는 성질임을 이해하고, 이와 무관한 설명을 가려내는 문제이다.

핵심 개념

데이터 독립성: 한 단계의 스키마를 변경해도 그 위 단계의 스키마나 응용 프로그램을 변경할 필요가 없도록 하는 성질이다.
물리적 데이터 독립성: 저장 장치, 파일 구조, 인덱스 등 내부 스키마가 바뀌어도 개념 스키마와 응용 프로그램에 영향을 주지 않는 성질이다.
논리적 데이터 독립성: 개념 스키마가 바뀌어도 외부 스키마와 응용 프로그램에 영향을 주지 않는 성질이다.

NoteBase

선지별 해설

① O
㉠·㉡·㉢은 모두 데이터 독립성에 관한 설명이 아니고, ㉣은 물리적 데이터 독립성에 관한 설명이므로 옳지 않은 설명만 정확히 고른 조합이다.

② X
㉣을 옳지 않은 설명에 포함한 부분과 ㉢을 제외한 부분이 틀렸다. ㉣은 하위 단계인 물리적 구조의 변경이 상위 단계와 응용 프로그램에 영향을 주지 않는 물리적 데이터 독립성을 설명한다. 옳지 않은 설명의 올바른 조합은 ㉠·㉡·㉢이다.

③ X
㉣을 옳지 않은 설명에 포함한 부분과 ㉡을 제외한 부분이 틀렸다. 부서별로 서로 다른 DBMS를 독립 운영한다는 ㉡은 데이터 독립성이 아니라 시스템 운영 방식에 관한 설명이다. 옳지 않은 설명의 올바른 조합은 ㉠·㉡·㉢이다.

④ X
㉣을 옳지 않은 설명에 포함한 부분과 ㉠을 제외한 부분이 틀렸다. 데이터 독립성은 데이터베이스 관리자의 데이터 독점 소유를 뜻하지 않으므로 ㉠은 옳지 않다. 옳지 않은 설명의 올바른 조합은 ㉠·㉡·㉢이다.

오답 포인트

‘독립성’이라는 일상적 표현 때문에 ㉡의 부서별 독립 운영이나 ㉢의 네트워크 없는 조회를 데이터 독립성으로 오해하기 쉽다. DBMS에서 데이터 독립성의 핵심은 소유권·조직 운영·통신 연결이 아니라 스키마 변경의 영향 차단이다.

다시 나오면 이렇게 판단

선지에서 ‘하위 단계의 구조가 바뀌어도 상위 단계나 응용 프로그램이 바뀌지 않는다’는 관계를 먼저 찾는다. 소유권, 부서별 시스템 분리, 네트워크 연결 여부를 말하면 데이터 독립성과 무관한 설명으로 판단한다.

암기 포인트

데이터 독립성은 ‘데이터를 따로 보유한다’가 아니라 ‘아래를 바꿔도 위는 그대로’이다.

NoteBase

문 11. 다음 중 운영체제의 교착 상태(Deadlock)가 발생하는 4가지 필요조건에 대한 설명으로 가장 옳지 않은 것은?

① 상호 배제(Mutual Exclusion): 자원은 한 번에 한 프로세스만 사용할 수 있으며, 다른 프로세스가 해당 자원을 사용하려면 사용 중인 프로세스가 자원을 놓아줄 때까지 기다려야 한다.

② 비선점(Non-Preemptive): 프로세스가 새로운 자원을 요청하기 전에 다른 프로세스가 점유하고 있는 자원을 확인하여, 자원을 대기하지 않도록 하는 것을 의미한다.

③ 점유 및 대기(Hold and Wait): 최소한 하나의 자원을 점유하고 있는 프로세스가 다른 프로세스에 의해 점유된 추가 자원을 할당받기 위해 대기하는 상태이다.

④ 순환 대기(Circular Wait): 대기 프로세스의 집합이 순환 형태로 서로가 가진 자원을 기다리며 끝없이 대기하는 상태이다.

정답 2번

출제 포인트

교착 상태가 발생하기 위한 네 가지 필요조건인 상호 배제, 점유 및 대기, 비선점, 순환 대기의 정확한 뜻을 구별하는 문제이다.

핵심 개념

상호 배제는 한 자원을 동시에 하나의 프로세스만 사용하는 조건이다. 점유 및 대기는 자원을 가진 채 다른 자원을 기다리는 조건이다. 비선점은 이미 할당된 자원을 운영체제나 다른 프로세스가 강제로 빼앗을 수 없고, 점유한 프로세스가 스스로 반납해야 하는 조건이다. 순환 대기는 프로세스들이 원형으로 연결되어 다음 프로세스가 가진 자원을 서로 기다리는 조건이다. 이 네 조건이 동시에 성립해야 교착 상태가 발생할 수 있다.

선지별 해설

① O — 상호 배제는 한 자원을 한 번에 하나의 프로세스만 사용할 수 있어 다른 프로세스가 그 자원의 반환을 기다리는 조건이다.
② X — 틀린 표현은 '새로운 자원을 요청하기 전에 다른 프로세스가 점유하고 있는 자원을 확인하여, 자원을 대기하지 않도록 하는 것'이다. 비선점은 이미 할당된 자원을 강제로 회수할 수 없으며 점유한 프로세스가 자발적으로 놓아주어야 한다는 뜻이다.
③ O — 점유 및 대기는 프로세스가 하나 이상의 자원을 보유한 상태에서 다른 프로세스가 가진 추가 자원을 기다리는 조건이다.
④ O — 순환 대기는 각 프로세스가 다음 프로세스가 점유한 자원을 기다리는 관계가 원형으로 형성된 상태이다.

NoteBase

오답 포인트

②는 '비선점'을 자원 요청 전에 가용 여부를 확인하는 방식으로 바꾸어 설명했다. 비선점의 판단 기준은 사전 확인 여부가 아니라 이미 할당된 자원을 강제로 빼앗을 수 있는지 여부이다.

다시 나오면 이렇게 판단

자원 하나를 혼자 쓰는지는 상호 배제, 가진 채 더 기다리는지는 점유 및 대기, 강제 회수가 불가능한지는 비선점, 기다림의 고리가 형성되는지는 순환 대기로 판단한다.

암기 포인트

교착 상태의 네 필요조건은 '혼자 사용·쥔 채 대기·강제 회수 불가·원형 대기'이다.

NoteBase

문 12. 다음 중 페이징(Paging)과 세그멘테이션(Segmentation)의 가상기억공간 관리 방법에 대한 설명으로 가장 옳지 않은 것은?

① 페이징 기법은 물리 메모리를 고정 크기로 분할하여 관리한다.

② 세그멘테이션 기법은 프로그램을 논리적 단위인 세그먼트로 나누어 관리하며, 각 세그먼트는 서로 다른 크기를 가질 수 있다.

③ 페이징에서는 페이지 테이블을 통해 가상 주소의 페이지 번호를 물리 주소의 프레임 번호로 매핑한다.

④ 페이징 기법은 외부 단편화는 발생하지만 내부 단편화가 발생하지 않는다.

정답 4번

출제 포인트

이 문제는 페이징과 세그멘테이션의 분할 단위, 주소 변환 방식, 단편화 특성을 구분하는 문제이다.

핵심 개념

페이징은 가상 메모리를 같은 크기의 페이지로, 물리 메모리를 같은 크기의 프레임으로 나누어 페이지 테이블로 대응시키는 방식이다. 고정 크기 단위이므로 외부 단편화는 발생하지 않지만, 할당된 마지막 페이지에 남는 공간 때문에 내부 단편화가 발생할 수 있다.
세그멘테이션은 프로그램을 코드·데이터·스택과 같은 논리적 단위의 세그먼트로 나누는 방식이다.
세그먼트마다 크기가 다를 수 있으며 외부 단편화가 발생할 수 있다.

선지별 해설

① O — 페이징은 물리 메모리를 동일한 고정 크기의 프레임으로 분할하여 관리한다. 가상 메모리의 페이지 크기와 물리 메모리의 프레임 크기는 같다.
② O — 세그멘테이션은 프로그램을 의미 있는 논리적 단위로 나누며, 각 세그먼트는 내용에 따라 서로 다른 크기를 가질 수 있다.
③ O — 페이지 테이블은 가상 주소의 페이지 번호에 대응하는 물리 메모리의 프레임 번호를 제공한다. 페이지 내부 변위는 그대로 결합되어 물리 주소가 만들어진다.
④ X — 틀린 부분은 '외부 단편화는 발생하지만 내부 단편화가 발생하지 않는다'이다. 올바르게는 페이징에서는 외부 단편화가 발생하지 않지만 내부 단편화가 발생할 수 있다.

오답 포인트

④는 페이징의 단편화 특성을 정확히 반대로 바꾼 선지이다. 고정 크기 블록을 사용하는 페이징은 흩어진 빈 공간 때문에 할당하지 못하는 외부 단편화를 없애지만, 페이지 단위로 할당하고 남는 공간에서 내부 단편화가 생길 수 있다.

NoteBase

다시 나오면 이렇게 판단

분할 단위가 고정 크기이면 페이징, 프로그램의 논리적 단위이며 크기가 가변적이면 세그멘테이션으로 판단한다. 단편화는 '페이징은 내부, 세그멘테이션은 외부'의 대응으로 구분한다.

암기 포인트

페이징은 고정 크기·페이지 테이블·내부 단편화, 세그멘테이션은 논리 단위·가변 크기·외부 단편화이다.

NoteBase

문 13. 다음 중 운영체제의 페이지 교체 및 가상 메모리 관리에서 지역성(Locality)에 대한 설명으로 가장 옳지 않은 것은?

① 시간 지역성(Temporal Locality)은 참조된 메모리 주소가 가까운 시간 내에 다시 참조될 가능성이 높은 성질이다.

② 공간 지역성(Spatial Locality)은 어떤 메모리 주소가 참조되면, 그 주변의 인접 주소들이 곧 참조될 가능성이 높은 성질이다.

③ 지역성의 특성을 잘 이용하여 페이지 부재(Page Fault)를 낮추어 시스템 성능을 향상시킬 수 있다.

④ 프로그램이 한번 실행될 때 메모리의 모든 영역에 대한 참조가 이루어지므로, 이를 페이지 교체 알고리즘에 반영한다.

정답 4번

출제 포인트

이 문제는 메모리 참조의 지역성 원리와 시간 지역성·공간 지역성의 의미를 구별하는 문제이다.

핵심 개념

지역성은 프로그램이 실행되는 동안 짧은 기간에는 전체 메모리가 아니라 일부 주소와 그 주변을 집중적으로 참조하는 성질이다.
시간 지역성은 최근 참조한 주소를 가까운 시간 안에 다시 참조할 가능성이 높은 성질이고, 공간 지역성은 참조한 주소와 가까운 주소를 곧 참조할 가능성이 높은 성질이다. 운영체제는 이러한 경향을 이용해 필요한 페이지를 메모리에 유지하고 페이지 부재를 줄인다.

선지별 해설

① O — 시간 지역성의 올바른 설명이다. 반복문이나 반복 호출처럼 최근 사용한 명령어나 데이터를 가까운 시간 안에 다시 참조하는 경우가 이에 해당한다.
② O — 공간 지역성의 올바른 설명이다. 배열을 순차적으로 읽는 경우처럼 한 주소가 참조되면 그 인접 주소도 곧 참조될 가능성이 높다.
③ O — 지역성을 활용하면 현재 자주 쓰이는 페이지와 인접 페이지를 메모리에 유지할 수 있으므로 페이지 부재가 줄고 시스템 성능이 향상될 수 있다.
④ X — 틀린 부분은 '메모리의 모든 영역에 대한 참조가 이루어진다'이다. 실제로는 프로그램이 일정 시간 동안 메모리의 일부 영역을 집중적으로 참조하며, 페이지 교체 알고리즘은 이러한 지역성을 반영해 앞으로 사용될 가능성이 높은 페이지를 유지한다.

NoteBase

오답 포인트

④는 프로그램 전체 실행 과정에서 여러 영역을 사용할 수 있다는 사실을 '한 번 실행되면 모든 영역을 참조한다'로 과장한다. 지역성의 핵심은 모든 영역의 균등한 참조가 아니라 특정 시점에 일부 영역으로 참조가 몰리는 현상이다.

다시 나오면 이렇게 판단

'최근 사용한 같은 주소를 다시 사용'하면 시간 지역성, '참조한 주소의 주변을 이어서 사용'하면 공간 지역성으로 판단한다. 전체 메모리를 고르게 또는 모두 참조한다는 표현은 지역성의 정의와 반대이다.

암기 포인트

시간 지역성은 같은 곳을 다시, 공간 지역성은 가까운 곳을 이어서 참조한다.

NoteBase

문 14. 다음 <보기> 자료구조에서 각각 효율적으로 자료가 배치되었을 때 검색 효율성이 가장 높은 것부터 낮은 순서대로 바르게 나열된 것은?

<보기>
㉠ 연결 리스트(Linked List)
㉡ 이진 탐색 트리(Binary Search Tree)
㉢ 해시테이블(Hash Table)

① ㉠ → ㉡ → ㉢
② ㉡ → ㉢ → ㉠
③ ㉢ → ㉡ → ㉠
④ ㉢ → ㉠ → ㉡

정답 3번

출제 포인트

효율적으로 구성된 해시테이블, 이진 탐색 트리, 연결 리스트의 일반적인 검색 시간 복잡도를 비교하여 빠른 순서대로 배열하는 문제이다.

올바른 순서

㉢ 해시테이블 → ㉡ 이진 탐색 트리 → ㉠ 연결 리스트이다.
해시 충돌이 적은 해시테이블은 평균 $O(1)$, 균형 잡힌 이진 탐색 트리는 $O(\log n)$, 연결 리스트는 순차 검색이 필요하므로 $O(n)$의 검색 시간이 걸린다. 따라서 자료 수가 커질수록 일반적인 검색 효율은 해시테이블, 이진 탐색 트리, 연결 리스트 순이다.

선지별 해설

① X — 연결 리스트를 가장 먼저 둔 것이 틀렸다. 연결 리스트는 임의의 값을 찾을 때 노드를 처음부터 차례로 확인하므로 검색이 $O(n)$이다. 올바른 순서는 ㉢ → ㉡ → ㉠이다.
② X — 이진 탐색 트리를 해시테이블보다 앞에 둔 것이 틀렸다. 효율적으로 구성된 경우 이진 탐색 트리는 $O(\log n)$, 해시테이블은 평균 $O(1)$이므로 올바른 순서는 ㉢ → ㉡ → ㉠이다.
③ O — 해시테이블의 평균 $O(1)$, 균형 잡힌 이진 탐색 트리의 $O(\log n)$, 연결 리스트의 $O(n)$ 순으로 검색 효율이 낮아진다.
④ X — 연결 리스트를 이진 탐색 트리보다 앞에 둔 것이 틀렸다. 연결 리스트의 순차 검색 $O(n)$보다 균형 잡힌 이진 탐색 트리의 검색 $O(\log n)$이 효율적이므로 올바른 순서는 ㉢ → ㉡ → ㉠이다.

NoteBase

오답 포인트

자료 삽입·삭제의 편의성과 검색 효율을 혼동하면 연결 리스트의 순위를 높게 판단하기 쉽다. 이 문항은 검색만 비교하며, '효율적으로 자료가 배치'되었다는 조건에 따라 해시테이블은 충돌이 적고 이진 탐색 트리는 균형이 잡힌 상태로 본다.

다시 나오면 이렇게 판단

검색 문제에서는 각 자료구조의 일반적인 탐색 방식을 먼저 확인한다. 해시 함수로 바로 위치를 찾으면 평균 $O(1)$, 탐색 범위를 절반씩 줄이면 $O(\log n)$, 처음부터 순차적으로 훑으면 $O(n)$으로 판단한다.

암기 포인트

효율적 검색 순서는 해시 $O(1)$ → 균형 이진 탐색 트리 $O(\log n)$ → 연결 리스트 $O(n)$이다.

NoteBase

문 15. 다음 중 데이터베이스의 스키마(Schema)에 대한 설명으로 가장 옳은 것은?

① 내부 스키마는 인덱스 구조나 데이터 저장 방식 등 성능 최적화와 관련된 물리적 관점을 다룬다.

② 개념 스키마는 개별 사용자의 관점에서 필요한 데이터만을 정의하며, 사용자마다 여러 개가 존재할 수 있다.

③ 외부 스키마는 물리적 저장 장치에서 데이터가 실제로 어떻게 저장되는지를 정의하는 단계이다.

④ 외부 스키마와 내부 스키마 사이에는 데이터 종속성이 있으며, 이는 사용자의 뷰를 개별적으로 생성하기 때문이다.

정답 1번

출제 포인트

데이터베이스의 3단계 스키마에서 외부·개념·내부 스키마가 각각 사용자 관점, 전체 논리 관점, 물리적 저장 관점 중 무엇을 나타내는지 구별하는 문제이다.

핵심 개념

외부 스키마: 개별 사용자나 응용 프로그램이 필요로 하는 데이터의 논리적 모습을 정의하는 사용자 뷰이며 여러 개가 존재할 수 있다.
개념 스키마: 데이터베이스 전체의 논리적 구조와 제약 조건을 통합하여 정의하며 하나만 존재한다.
내부 스키마: 데이터의 실제 저장 방식, 파일 조직, 인덱스 등 물리적 구조를 정의하며 하나만 존재한다.
한 줄 대비: 외부는 사용자별 뷰, 개념은 전체 논리 구조, 내부는 물리적 저장 구조이다.

선지별 해설

① O — 내부 스키마는 인덱스 구조, 레코드 배치, 파일 조직과 같은 물리적 저장 방법을 다룬다. 이러한 요소는 저장 공간과 접근 성능의 최적화에 직접 관련되므로 옳다.
② X — 틀린 표현은 '개념 스키마'와 '개별 사용자의 관점', '여러 개'이다. 올바르게는 외부 스키마가 사용자별로 필요한 데이터 뷰를 정의하여 여러 개 존재할 수 있고, 개념 스키마는 데이터베이스 전체의 논리 구조를 정의하여 하나만 존재한다.
③ X — 틀린 표현은 '외부 스키마'이다. 올바르게는 내부 스키마가 물리적 저장 장치에 데이터가 실제로 저장되는 방식을 정의하며, 외부 스키마는 개별 사용자의 논리적 뷰를 정의한다.
④ X — 틀린 표현은 '외부 스키마와 내부 스키마 사이에는 데이터 종속성이 있다'이다. 올바르게는 외부-개념 사상과 개념-내부 사상을 통해 논리적 데이터 독립성과 물리적 데이터 독립성을 제공하여 한 단계의 변경이 다른 단계에 미치는 영향을 줄인다. 사용자별 뷰를 만드는 것은 외부 스키마의 역할이지 데이터 종속성의 근거가 아니다.

NoteBase

오답 포인트

②와 ③은 외부·개념·내부 스키마의 역할을 서로 바꾸어 놓은 선지이다. 특히 '사용자마다 여러 개'는 외부 스키마, '전체 논리 구조'는 개념 스키마, '실제 저장 방식'은 내부 스키마의 표지어이다. ④는 단계 간 사상이 데이터 종속성을 강화하는 것이 아니라 데이터 독립성을 제공한다는 점을 뒤집었다.

다시 나오면 이렇게 판단

선지의 관점과 핵심 명사를 먼저 확인한다. 사용자·뷰는 외부 스키마, 전체 데이터베이스·논리 구조·제약 조건은 개념 스키마, 파일·레코드·인덱스·저장 장치는 내부 스키마로 판단한다. 단계 간 변경 영향의 최소화가 나오면 데이터 독립성과 연결한다.

암기 포인트

외부는 사용자별 여러 뷰, 개념은 전체 논리 구조 하나, 내부는 물리 저장 구조 하나이다. 외-개 사상은 논리적 독립성, 개-내 사상은 물리적 독립성과 연결된다.

NoteBase

문 16. 다음 중 데이터베이스의 뷰(View)에 대한 설명으로 가장 옳은 것은?

① 뷰는 실제 데이터를 물리적으로 저장하고 있는 테이블이므로, 생성 시 많은 저장 공간이 필요하다.

② 뷰를 사용하면 사용자에게 필요한 일부만 보여 줄 수 있어, 데이터 보안성을 높일 수 있다.

③ 뷰는 오직 하나의 기본 테이블(Base Table)로부터만 생성될 수 있으며, 여러 테이블을 조인하여 만들 수 없다.

④ 기본 테이블의 구조가 변경되면 뷰를 사용하더라도 응용 프로그램의 쿼리문 수정이 필요하다.

정답 2번

출제 포인트

데이터베이스 뷰의 저장 방식, 보안 기능, 생성 범위, 논리적 데이터 독립성을 구별하는 문제이다.

핵심 개념

뷰는 하나 이상의 기본 테이블에서 필요한 데이터를 조회하도록 정의한 가상 테이블이다. 일반적인 뷰는 실제 데이터 자체가 아니라 질의를 정의한 내용만 저장하며, 사용자에게 특정 행이나 열만 공개하여 접근 범위를 제한할 수 있다. 또한 응용 프로그램과 기본 테이블 사이에 논리적 인터페이스를 제공하므로 기본 테이블의 변화가 응용 프로그램에 미치는 영향을 줄일 수 있다.

선지별 해설

① X — 틀린 표현은 '실제 데이터를 물리적으로 저장하고 있는 테이블'과 '많은 저장 공간이 필요하다'이다. 일반적인 뷰는 가상 테이블로서 질의 정의를 저장하고, 조회 결과 데이터 자체를 별도로 물리적 저장하지 않는다.
② O — 뷰를 이용하면 사용자별로 필요한 행이나 열만 보이게 하고 나머지 데이터는 감출 수 있으므로 데이터 보안성을 높일 수 있다.
③ X — 틀린 표현은 '오직 하나의 기본 테이블로부터만'과 '여러 테이블을 조인하여 만들 수 없다'이다. 뷰는 하나의 기본 테이블뿐 아니라 여러 기본 테이블을 조인한 조회 결과로도 만들 수 있다.
④ X — 틀린 표현은 '응용 프로그램의 쿼리문 수정이 필요하다'이다. 기본 테이블의 구조가 바뀌더라도 뷰가 기존 인터페이스를 유지하도록 정의되면 응용 프로그램의 쿼리문을 수정하지 않을 수 있으며, 이것이 뷰가 제공하는 논리적 데이터 독립성의 장점이다.

오답 포인트

①은 일반 뷰를 조회 결과를 실제로 저장하는 물리적 테이블처럼 표현한 함정이다. ③은 뷰의 원본을 한 테이블로 제한하고, ④는 뷰가 기본 테이블의 구조 변화를 감추어 줄 수 있다는 장점을 반대로 서술했다.

NoteBase

다시 나오면 이렇게 판단

뷰 관련 선지는 '가상 테이블인가', '행·열 제한으로 보안을 높이는가', '여러 테이블의 조인이 가능한가', '응용 프로그램과 기본 테이블 사이의 논리적 독립성을 높이는가'를 차례로 확인한다.

암기 포인트

뷰는 가상 테이블이며, 제한 공개로 보안을 높이고 여러 테이블을 조인할 수 있으며 논리적 데이터 독립성을 지원한다.

NoteBase

문 17. 소프트웨어 모듈에서 결합도(Coupling)와 응집도(Cohesion)에 대한 설명으로 가장 옳지 않은 것은?

① 좋은 소프트웨어 설계는 강한 응집과 약한 결합을 목적해야 한다.

② 응집도는 모듈 내부의 요소들이 얼마나 밀접하게 관련되어 있는지를 나타내는 척도이다.

③ 결합도가 높아질수록 모듈 간의 연결성이 높아져 유지보수가 쉬워지고 재사용성이 좋아진다.

④ 기능적 응집도는 응집도의 단계 중 가장 높은 수준이다.

정답 3번

출제 포인트

모듈 내부의 관련성을 나타내는 응집도와 모듈 사이의 의존성을 나타내는 결합도의 바람직한 방향을 구별하는 문제이다.

핵심 개념

응집도는 하나의 모듈을 구성하는 요소들이 같은 목적을 위해 얼마나 밀접하게 관련되는지를 나타낸다. 응집도가 높을수록 모듈의 기능이 명확해진다.
결합도는 서로 다른 모듈이 얼마나 강하게 의존하는지를 나타낸다. 결합도가 낮을수록 한 모듈의 변경이 다른 모듈에 미치는 영향이 작아진다.
좋은 모듈 설계의 기본 방향은 강한 응집과 약한 결합이다.

선지별 해설

① O — 모듈 내부에서는 관련 기능을 하나로 모아 응집도를 높이고, 모듈 사이에서는 의존성을 줄여 결합도를 낮추는 것이 좋은 설계의 방향이다.
② O — 응집도는 한 모듈 내부의 구성 요소들이 단일 목적과 얼마나 밀접하게 관련되는지를 측정하는 척도이다.
③ X — 틀린 표현은 '결합도가 높아질수록 유지보수가 쉬워지고 재사용성이 좋아진다'이다. 결합도가 높으면 한 모듈의 변경이 다른 모듈에 연쇄적으로 영향을 주므로 유지보수가 어려워지고 독립적 재사용도 제한된다.
올바르게는 결합도가 낮아질수록 유지보수성과 재사용성이 좋아진다.
④ O — 기능적 응집은 모듈의 모든 요소가 하나의 명확한 기능을 수행하기 위해 협력하는 형태로, 응집도 단계 중 가장 높은 수준이다.

오답 포인트

③은 '모듈 간 연결성이 높다'를 협업이 잘된다는 긍정적 의미로 오해하게 만든다. 그러나 결합도에서 연결성이 높다는 것은 모듈 간 의존성이 크다는 뜻이므로 유지보수성과 재사용성에 불리하다.

NoteBase

다시 나오면 이렇게 판단

내부 요소의 관련성을 묻는다면 응집도, 모듈 사이의 의존성을 묻는다면 결합도로 판단한다. 설계 품질은 응집도는 높을수록, 결합도는 낮을수록 좋다는 방향으로 판별한다.

암기 포인트

모듈 안은 강하게 묶고, 모듈 사이는 느슨하게 연결한다. 즉, 강한 응집과 약한 결합이다.

NoteBase

문 18. 다음 중 AI와 관련된 내용으로 가장 옳지 않은 것은?

① ChatGPT와 같이 인터넷을 통한 클라우드 서비스 AI모델은 보안 위험성이 있다.

② 허깅페이스(Hugging Face)에서 다운로드 가능하도록 공개된 LLM(Large Language Model)은 격리된 폐쇄망 내부에서도 구동 가능하다.

③ LLM은 텍스트 형태의 입력을 받으며, 이미지, 소리 등 기타 데이터 처리 과정에 응용하지 못한다.

④ RAG(Retrieval-Augmented Generation)는 AI 모델에 대하여 일정 지식정보를 추가하여 답변 결과를 전문화 하기 위한 방법이다.

정답 3번

출제 포인트

클라우드 AI의 보안 위험, 공개 LLM의 로컬 구동, 멀티모달 처리, RAG의 개념을 구별하는 문제이다.

핵심 개념

LLM은 대규모 언어 데이터를 학습하여 문장을 이해하고 생성하는 모델이다. 최근의 AI 시스템은 LLM을 중심으로 이미지·음성 처리 모듈을 결합하거나 멀티모달 모델로 구성하여 텍스트 이외의 데이터도 처리할 수 있다. RAG는 질문과 관련된 외부 자료를 검색한 뒤 그 내용을 생성 모델의 입력 문맥에 제공하여 답변의 전문성과 근거성을 높이는 방식이다.

선지별 해설

① O — 인터넷을 통해 외부 클라우드 AI 서비스를 이용하면 입력 데이터가 외부 서버로 전송될 수 있으므로 기밀정보 유출, 접근 통제, 저장·처리 위치 등의 보안 위험을 고려해야 한다.
② O — 모델 가중치와 실행 프로그램을 미리 내려받아 내부 장비에 갖추면 인터넷과 분리된 폐쇄망에서도 공개 LLM을 로컬로 구동할 수 있다.
③ X — 틀린 표현은 '이미지, 소리 등 기타 데이터 처리 과정에 응용하지 못한다'이다. 올바르게는 이미지·음성을 인식하는 모듈과 LLM을 결합하거나 멀티모달 모델을 사용하면 텍스트뿐 아니라 이미지와 소리도 입력·처리할 수 있다.
④ O — RAG는 관련 지식정보를 외부 저장소에서 검색하여 모델의 입력 문맥에 추가함으로써 특정 분야에 더 전문적이고 근거 있는 답변을 생성하게 한다. 모델 자체를 반드시 재학습시키는 방식은 아니다.

오답 포인트

③은 LLM이라는 이름의 '언어'에만 주목하여 활용 범위를 텍스트로 제한한 함정이다. ④의 '지식정보를 추가'는 모델 가중치에 지식을 새로 학습시킨다는 뜻이 아니라, 검색한 자료를 답변 생성 시 입력 문맥으로 제공한다는 뜻으로 이해한다.

NoteBase

다시 나오면 이렇게 판단

‘텍스트만 가능’, ‘이미지·음성에는 응용 불가’처럼 AI의 처리 대상을 절대적으로 제한하는 표현을 먼저 점검한다. RAG는 외부 지식 검색과 문맥 제공, 파인튜닝은 모델 가중치 조정이라는 차이로 구별한다.

암기 포인트

멀티모달은 텍스트·이미지·음성을 함께 처리하고, RAG는 검색한 지식을 문맥으로 보강한다.

NoteBase

문 19. 다음 <보기>의 특성과 소프트웨어 개발 생명주기 모델이 가장 옳게 짝지어진 것은?

<보기>
㉠ 순차적 단계 진행
㉡ 위험 분석 수행
㉢ 반복적 점진 개발
㉣ 테스트 단계 강조
㉤ 상세 문서 작성 우선
㉥ 유연한 요구사항 변경

① 폭포수 모델: ㉠, V-모델: ㉥, 나선형 모델: ㉡, 애자일 모델: ㉤
② 폭포수 모델: ㉠, V-모델: ㉣, 나선형 모델: ㉡, 애자일 모델: ㉥
③ 폭포수 모델: ㉢, V-모델: ㉣, 나선형 모델: ㉠, 애자일 모델: ㉥
④ 폭포수 모델: ㉢, V-모델: ㉥, 나선형 모델: ㉠, 애자일 모델: ㉤

정답 2번

출제 포인트

소프트웨어 개발 생명주기 모델을 구분하는 핵심 기준인 순차성, 테스트 대응, 위험 분석, 요구사항 변화 대응을 묻는 문제이다.

핵심 개념

폭포수 모델: 요구분석부터 유지보수까지 단계를 순차적으로 진행하는 계획 중심 모델이다. 키워드는 순차적 단계 진행(㉠)이다.
V-모델: 개발 단계와 그에 대응하는 테스트 단계를 V자 형태로 연결하여 검증과 확인을 강조하는 모델이다. 키워드는 테스트 단계 강조(㉣)이다.
나선형 모델: 반복 개발 과정마다 위험을 식별하고 분석하여 위험을 줄이는 모델이다. 키워드는 위험 분석 수행(㉡)이다.
애자일 모델: 짧은 반복과 고객 피드백을 통해 변화하는 요구사항에 유연하게 대응하는 모델이다. 키워드는 유연한 요구사항 변경(㉥)이다.

NoteBase

선지별 해설

① X
폭포수 모델－㉠과 나선형 모델－㉡은 맞지만, V-모델－㉤과 애자일 모델－㉣이 틀렸다. V-모델은 테스트 단계를 강조하는 ㉢, 애자일 모델은 요구사항 변경에 유연한 ㉤과 연결해야 한다. 애자일은 상세 문서 작성을 작동하는 소프트웨어와 변화 대응보다 우선하지 않는다.

② O
폭포수 모델－㉠, V-모델－㉢, 나선형 모델－㉡, 애자일 모델－㉤이 각 모델의 대표 특성과 모두 정확히 대응한다.

③ X
V-모델－㉢과 애자일 모델－㉤은 맞지만, 폭포수 모델－㉥과 나선형 모델－㉠이 틀렸다. 폭포수 모델은 순차적 단계 진행인 ㉠, 나선형 모델은 위험 분석 수행인 ㉡과 연결해야 한다. 반복적·점진적 개발인 ㉥은 애자일의 대표 특성이다.

④ X
네 연결이 모두 틀렸다. 폭포수 모델－㉥은 폭포수 모델－㉠으로, V-모델－㉤은 V-모델－㉢로, 나선형 모델－㉠은 나선형 모델－㉡으로 고쳐야 한다. 애자일 모델－㉣도 틀리며, 애자일은 상세 문서 작성 우선이 아니라 유연한 요구사항 변경인 ㉤과 연결한다.

오답 포인트

㉥의 반복적·점진적 개발과 ㉤의 유연한 요구사항 변경은 모두 애자일과 관련되어 혼동하기 쉽다. 이 문항의 선지에서는 애자일의 직접 대응 특성으로 ㉤을 사용하며, V-모델은 이름의 모양보다 개발 단계별 테스트 대응을 강조한다는 점이 핵심이다.

다시 나오면 이렇게 판단

모델 이름마다 대표 키워드를 먼저 고정한다. 순차 진행은 폭포수, 개발 단계와 테스트 단계의 대응은 V-모델, 위험 분석은 나선형, 반복·피드백·변화 대응은 애자일로 판단한다.

암기 포인트

폭포수는 순차, V는 테스트, 나선은 위험, 애자일은 변화 대응이다.

NoteBase

문 20. 다음 <보기>에서 자바 코드의 실행 결과로 출력되는 값은?

< 보 기 >

```
class Calculator {
    public int compute(int n) {
        return n * 2;
    }
}

class AddCalculator extends Calculator {
    @Override
    public int compute(int n) {
        return super.compute(n) + 10;
    }
}

class SubCalculator extends Calculator {
    @Override
    public int compute(int n) {
        return super.compute(n) - 5;
    }
}

public class Main {
    public static void main(String[] args) {
        Calculator[] calcs = {
            new AddCalculator(),
            new SubCalculator(),
            new Calculator()
        };

        int total = 0;
        for (Calculator c : calcs) {
            total += c.compute(10);
        }

        System.out.println(total);
    }
}
```

① 45

② 55

③ 60

④ 65

정답 4번

출제 포인트

상위 클래스형 배열에 서로 다른 하위 클래스 객체가 들어 있을 때 오버라이딩된 메서드가 실제 객체의 자료형에 따라 동적으로 호출된다는 점과 각 반환값의 합계를 구하는 문제이다.

조건 정리

Calculator.compute(10)은 10을 2배 하여 20을 반환한다.
AddCalculator와 SubCalculator는 compute 메서드를 오버라이딩하며, 두 메서드 안의 super.compute(10)은 상위 클래스 Calculator의 메서드를 호출한다.
calcs에는 AddCalculator 객체, SubCalculator 객체, Calculator 객체가 차례로 들어 있다.

NoteBase

사용할 원리

향상된 for문에서 c의 선언형은 Calculator이지만, c.compute(10)은 각 원소가 실제로 가리키는 객체의 오버라이딩 메서드를 호출한다. 따라서 세 객체의 반환값을 각각 구한 뒤 total에 모두 더한다.

계산 과정

1. AddCalculator 객체: super.compute(10) + 10 = 20 + 10 = 30이다.
2. SubCalculator 객체: super.compute(10) - 5 = 20 - 5 = 15이다.
3. Calculator 객체: compute(10) = 20이다.
4. total = 30 + 15 + 20 = 65이므로 65가 출력된다.

선지 판단

① X — 45는 AddCalculator와 SubCalculator의 반환값 30과 15만 더한 값으로, 배열의 세 번째 Calculator 객체가 반환하는 20을 누락한 결과이다. 출력값은 45가 아니라 65이다.
② X — 세 객체가 반환하는 값은 각각 30, 15, 20이므로 합계는 55가 아니라 65이다.
③ X — 반복문은 배열의 세 원소를 모두 처리하며 합계는 60이 아니라 65이다.
④ O — 30 + 15 + 20 = 65이므로 실제 출력값과 일치한다.

오답 포인트

c의 선언형이 Calculator라는 이유로 모든 호출에 Calculator.compute가 실행된다고 보면 안 된다. 또한 ①의 45는 마지막 Calculator 객체의 반환값 20을 합산하지 않을 때 나오는 값이다.

다시 나오면 이렇게 판단

배열의 각 원소에 대해 실제 객체의 클래스를 먼저 확인하고, 오버라이딩된 메서드 내부의 super 호출을 계산한 다음, 반복문의 누적 연산을 원소 수만큼 빠짐없이 적용한다.

암기 포인트

오버라이딩 메서드는 실제 객체 기준으로 호출되고, super는 바로 위 상위 클래스의 메서드를 호출한다.

NoteBase