↑ 목록으로 다음 단원 →

1단원. 데이터베이스 기초 개념

> 대상: 전공자 · 정보처리기사 취득자 | 목적: 인프라 직무(네트워크·클라우드·시스템·보안) 기술 면접 대비 중요도: ★★★ 최빈출 · ★★ 자주 출제 · ★ 기본 개념

Q1. 관계형 데이터베이스(**RDBMS**)와 **NoSQL**의 차이를 설명하세요. ★★★

답안 관계형 데이터베이스는 데이터를 행과 열로 구성된 테이블 형태로 저장하고, 스키마가 사전에 고정되어 있으며 **SQL**을 통해 데이터를 조작합니다. 테이블 간 관계는 외래키로 정의되고 **ACID** 트랜잭션을 보장해 정합성이 중요한 금융, 주문 시스템에 적합합니다. **NoSQL**은 스키마가 유연하거나 없으며 문서형(MongoDB), 키-값형(Redis), 컬럼형(Cassandra), 그래프형(Neo4j) 등 데이터 모델이 다양합니다. 대량의 비정형/반정형 데이터를 수평 확장(샤딩)으로 처리하는 데 강점이 있고, 강한 일관성보다 가용성과 확장성을 우선시하는 경우가 많습니다(BASE 특성). 면접에서는 "왜 이 프로젝트에 RDBMS 대신 NoSQL을 선택했는가"처럼 트레이드오프를 설명하는 능력을 자주 확인합니다.

관련 개념 스키마리스, 샤딩, CAP 이론, BASE, 수평/수직 확장

Q2. 정규화란 무엇이며 왜 필요한지 설명하세요. ★★★

답안 정규화는 데이터의 중복을 최소화하고 삽입·삭제·갱신 이상(anomaly)을 방지하기 위해 테이블을 분해하는 설계 기법입니다. 제1정규형(1NF)은 각 속성이 원자값을 가지도록 하고, 제2정규형(2NF)은 기본키의 일부에만 종속되는 부분 함수 종속을

제거합니다. 제3정규형(3NF)은 기본키가 아닌 속성이 다른 비키 속성에 종속되는 이행적 종속을 제거해 데이터 일관성을 높입니다. 정규화를 거치면 저장 공간이 절약되고 갱신 시 한 곳만 수정하면 되지만, 조인이 늘어나 조회 성능이 저하될 수 있어 실무에서는 조회 성능을 위해 의도적으로 반정규화를 병행하기도 합니다.

관련 개념 이상현상(삽입/삭제/갱신 이상), 함수적 종속, BCNF, 반정규화, 조인 비용

> **Q3.** 트랜잭션의 **ACID** 특성을 각각 설명하세요. ★★★

답안 원자성(**Atomicity**)은 트랜잭션 내 모든 연산이 전부 성공하거나 전부 실패해야 한다는 원칙입니다. 일관성(**Consistency**)은 트랜잭션 수행 전후로 데이터베이스가 정의된 제약조건을 항상 만족하는 상태를 유지해야 함을 의미합니다. 고립성(**Isolation**)은 동시에 실행되는 트랜잭션들이 서로의 중간 결과에 영향을 주지 않도록 격리되는 것이고, 지속성(**Durability**)은 커밋된 트랜잭션의 결과는 시스템 장애가 발생해도 영구적으로 보존되어야 함을 뜻합니다. 이 네 가지 특성은 은행 이체처럼 여러 연산이 하나의 논리적 단위로 묶여야 하는 상황에서 데이터 정합성을 보장하는 핵심 근거가 됩니다.

관련 개념 커밋/롤백, 트랜잭션 로그(WAL), 격리 수준, 데드락, 회복 기법(REDO/UNDO)

> **Q4.** 기본키(**Primary Key**)와 외래키(**Foreign Key**)의 차이를 설명하세요. ★★★

답안 기본키는 테이블 내에서 각 행(row)을 유일하게 식별하는 속성 또는 속성 집합으로, **NULL**을 허용하지 않으며 중복될 수 없습니다. 외래키는 다른 테이블의 기본키를 참조하는 속성으로, 두 테이블 간의 관계를 표현하며 참조 무결성을 보장하는 역할을 합니다. 예를 들어 주문(Order) 테이블의 고객ID는 고객(Customer) 테이블의 기본키를 참조하는 외래키이며, 이를 통해 존재하지 않는 고객의 주문이 생성되는 것을 방지합니다. 외래키에는 NULL이 허용될 수 있고, 부모 테이블의 데이터가

삭제될 때 자식 테이블의 처리 방식을 CASCADE, SET NULL, RESTRICT 등으로 지정할 수 있습니다.

관련 개념 참조 무결성, 후보키/대체키, 복합키, ON DELETE CASCADE, 자연키/대리키

> **Q5.** 인덱스(**Index**)란 무엇이며 어떻게 조회 성능을 향상시키는지 설명하세요. ★★★

답안 인덱스는 테이블의 특정 컬럼 값을 빠르게 찾을 수 있도록 별도로 정렬·구조화해 둔 자료구조로, 대부분 **B-Tree** 기반으로 구현됩니다. 인덱스가 없으면 조건에 맞는 행을 찾기 위해 테이블 전체를 스캔(Full Table Scan)해야 하지만, 인덱스가 있으면 트리 구조를 탐색해 O(log n) 수준으로 원하는 데이터에 접근할 수 있습니다. 다만 인덱스는 별도의 저장 공간을 차지하고, INSERT/UPDATE/DELETE 시마다 인덱스도 함께 갱신해야 하므로 쓰기 성능은 저하될 수 있습니다. 따라서 WHERE절, JOIN, ORDER BY에 자주 사용되는 컬럼에 선택적으로 인덱스를 생성하는 것이 중요하며, 카디널리티(값의 다양성)가 낮은 컬럼은 인덱스 효과가 떨어집니다.

관련 개념 B-Tree/B+Tree, 클러스터형/비클러스터형 인덱스, 카디널리티, 커버링 인덱스, 풀 스캔

> **Q6. ERD(Entity-Relationship Diagram)**란 무엇이며 데이터베이스 설계에서 어떤 역할을 하는지 설명하세요. ★★

답안 **ERD**는 데이터베이스에 저장할 개체(Entity)와 개체 간의 관계(Relationship), 속성(Attribute)을 시각적으로 표현한 다이어그램입니다. 개체는 테이블로, 속성은 컬럼으로, 관계는 외래키나 별도의 관계 테이블로 구현되는 것이 일반적입니다. ERD를 작성하면 실제 구현에 앞서 데이터 구조와 업무 규칙을 이해관계자들과 함께 검토할 수 있어 설계 오류를 조기에 발견할 수 있습니다. 관계는 1:1, 1:N, N:M 등의

카디널리티로 표현되며, N:M 관계는 실제 테이블 구현 시 중간에 연결 테이블(교차 테이블)을 두어 두 개의 1:N 관계로 분해합니다.

관련 개념 개체·속성·관계, 카디널리티(1:1, 1:N, N:M), 연결 테이블, 개념적/논리적/물리적 설계, 식별관계

> **Q7.** 트랜잭션의 상태(활성, 부분완료, 완료, 실패, 철회)를 설명하세요. ★★

답안 트랜잭션은 시작되면 활성(**Active**) 상태가 되어 연산을 수행하고, 마지막 연산까지 실행되면 부분완료(**Partially Committed**) 상태가 됩니다. 이후 변경 사항이 데이터베이스에 영구 반영되면 완료(**Committed**) 상태가 되고, 정상적으로 트랜잭션이 종료됩니다. 만약 실행 중 오류가 발생하면 실패(**Failed**) 상태가 되고, 지금까지의 변경 사항을 취소하는 철회(**Aborted/Rollback**) 과정을 거쳐 트랜잭션 시작 이전 상태로 복구됩니다. 이 상태 전이 모델은 회복 관리자가 시스템 장애 시 REDO/UNDO 연산을 판단하는 기준이 됩니다.

관련 개념 커밋 포인트, 롤백, 회복 관리자, 체크포인트, 로그 기반 회복

> **Q8.** 개체 무결성과 참조 무결성의 차이를 설명하세요. ★★

답안 개체 무결성(**Entity Integrity**)은 기본키를 구성하는 속성이 NULL 값을 가질 수 없고 중복되어서도 안 된다는 제약으로, 각 행을 유일하게 식별하기 위한 최소 조건입니다. 참조 무결성(**Referential Integrity**)은 외래키 값이 참조하는 테이블의 기본키 값으로 존재하거나 NULL이어야 한다는 제약으로, 서로 다른 테이블 간의 관계 일관성을 보장합니다. 예를 들어 주문 테이블의 고객ID가 실제 고객 테이블에 존재하지 않는 값을 가지면 참조 무결성이 깨진 것입니다. 이 두 무결성 제약은 DBMS가 자동으로 검증하도록 스키마 정의 시 PRIMARY KEY와 FOREIGN KEY 제약조건으로 명시합니다.

관련 개념 도메인 무결성, 제약조건(CHECK, UNIQUE, NOT NULL), 캐스케이드 옵션, 무결성 위반 처리

> **Q9.** 뷰(**View**)란 무엇이며 어떤 장단점이 있는지 설명하세요. ★★

답안 뷰는 하나 이상의 실제 테이블을 기반으로 정의된 가상 테이블로, 실제 데이터를 저장하지 않고 조회할 때마다 정의된 쿼리를 실행해 결과를 보여줍니다. 뷰를 사용하면 복잡한 조인이나 서브쿼리를 감춰 사용자가 단순화된 인터페이스로 데이터에 접근할 수 있고, 특정 컬럼이나 행만 노출시켜 보안성을 높일 수 있습니다. 반면 뷰는 매번 기반 쿼리를 다시 실행하므로 복잡한 뷰는 조회 성능이 저하될 수 있고, 일반적으로 여러 테이블을 조인한 뷰는 삽입·수정·삭제 연산에 제약이 따릅니다. 자주 조회되지만 실시간성이 크게 중요하지 않은 경우 결과를 실제로 저장하는 구체화 뷰 **(Materialized View)**를 고려하기도 합니다.

관련 개념 가상 테이블, 구체화 뷰, 뷰 갱신 제약, 데이터 은닉, 논리적 독립성

> **Q10.** 이상현상(삽입 이상, 삭제 이상, 갱신 이상)을 예를 들어 설명하세요. ★★

답안 정규화되지 않은 테이블에서는 세 가지 이상현상이 발생할 수 있습니다. 삽입 이상은 예를 들어 학생-수강 정보가 한 테이블에 있을 때, 아직 수강 신청을 하지 않은 학생을 등록하려면 수강 정보를 NULL로 채워야 하는 문제입니다. 삭제 이상은 특정 학생의 유일한 수강 기록을 삭제하면 그 학생의 다른 정보(학과 등)까지 함께 사라지는 문제입니다. 갱신 이상은 동일한 정보가 여러 행에 중복 저장되어 있을 때 일부 행만 수정하면 데이터 불일치가 발생하는 문제입니다. 이러한 이상현상은 정규화를 통해 테이블을 적절히 분해함으로써 해결할 수 있습니다.

관련 개념 정규화, 함수적 종속, 데이터 중복, 무결성 유지, 테이블 분해

> **Q11.** 슈퍼키, 후보키, 대체키, 기본키의 차이를 설명하세요. ★

답안 슈퍼키(**Super Key**)는 튜플을 유일하게 식별할 수 있는 속성의 집합으로, 불필요한 속성이 포함되어도 무방합니다. 후보키(**Candidate Key**)는 슈퍼키 중에서 최소성을 만족하는, 즉 더 이상 속성을 제거하면 유일성을 잃는 키입니다. 후보키가 여러 개 존재할 수 있으며, 이 중 대표로 선택된 것이 기본키(**Primary Key**)이고 선택되지 않은 나머지 후보키들은 대체키(**Alternate Key**)라고 부릅니다. 예를 들어 학생 테이블에서 학번과 주민등록번호가 모두 후보키가 될 수 있다면, 학번을 기본키로 정하고 주민등록번호는 대체키가 됩니다.

관련 개념 최소성, 유일성, 복합키, 자연키/대리키, 무결성 제약

> **Q12. CAP** 이론이란 무엇이며 데이터베이스 선택에 어떤 영향을 주는지 설명하세요. ★★

답안 **CAP** 이론은 분산 시스템에서 일관성(**Consistency**), 가용성(**Availability**), 분단 허용성(**Partition Tolerance**) 세 가지를 동시에 모두 만족할 수 없고, 네트워크 분단이 발생했을 때 최대 두 가지만 선택 가능하다는 이론입니다. 분산 환경에서는 네트워크 분단이 항상 발생할 수 있다고 가정해야 하므로, 실질적으로는 일관성과 가용성 중 하나를 우선시하는 트레이드오프를 하게 됩니다. 예를 들어 카산드라(Cassandra)는 가용성을 우선시하는 AP 시스템에 가깝고, 전통적인 RDBMS나 일부 NoSQL(HBase 등)은 일관성을 우선시하는 CP 시스템에 가깝습니다. 실무에서는 서비스 요구사항(강한 일관성이 필요한 결제 시스템 vs 높은 가용성이 필요한 SNS 피드)에 따라 적합한 데이터베이스를 선택합니다.

관련 개념 분산 시스템, 최종적 일관성(Eventual Consistency), AP/CP/CA, 복제(Replication), 파티션 분단
