**- DBMS가 무엇인지, DB를 사용하는 이유에 대해 설명해주시겠어요?**

DBMS(Database Management System)는 데이터베이스 관리 시스템으로 여러 사용자가 데이터베이스에 접근해 사용할 수 있도록 하는 소프트웨어를 의미합니다.

DB를 사용하지 않고, OS에 종속적인 파일 시스템을 활용하면 프로그램의 확장성을 해칩니다.

또한 파일 시스템의 데이터 중복, 비일관성, 검색 등의 문제를 해결할 수 있습니다.

DB의 특징 :

1) 실시간 접근성 : 비정형적인 질의에 대해 실시간 처리에 의한 응답이 가능합니다.

2) 지속적인 변화 : DB의 상태는 동적이기 때문에, 새로운 데이터의 삽입, 삭제, 갱신으로 항상 최신의

데이터를 유지합니다.

3) 동시 공용 : 데이터베이스는 여러 사용자가 동시에 같은 내용의 데이터를 이용하도록 합니다.

4) 내용에 의한 참조 : 데이터베이스에 있는 데이터를 참조할 때는 주소나 위치가 아닌 사용자가

요구하는 데이터 내용 기반으로 찾습니다.

**- DB에서 '스키마'가 무엇일까요?**

데이터베이스의 구조와 제약 조건에 관한 전반적인 명세를 기술한 메타데이터 집합을 의미합니다.

DB에서 데이터 개체, 속성, 관계를 형식언어로 정의한 구조로, DB 사용자가 자료를 저장, 조회, 삭제, 변경할 때 DBMS는 자신이 생성한 데이터베이스 스키마를 참조해 명령을 수행하게 됩니다.

- 외부 스키마(External Schema) : 프로그래머나 사용자의 입장에서 데이터베이스의 모습으로 조직의

일부분을 정의한 것

- 개념 스키마(Conceptual Schema) : 모든 응용 시스템과 사용자들이 필요로하는 데이터를 통합한 조직

전체의 데이터베이스 구조를 논리적으로 정의한 것

- 내부 스키마(Internal Schema) : 전체 데이터베이스의 물리적 저장 형태를 기술하는 것

**- DB에서 '뷰'는 무엇인가요?**

- 뷰는 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 테이블에서 유도된,

메모리에는 물리적으로 존재하지 않는 가상테이블입니다.

- 뷰는 저장장치 내에 물리적으로 존재하진 않지만, 사용자에게 있는 것처럼 간주되며 데이터 보정작업,

처리과정 테스트 등 임시적인 작업을 위한 용도로 사용됩니다.

- 뷰는 JOIN문의 사용 최소화로 사용상의 편의성을 최대화 합니다.

**- DB에서 사용하는 '키'에 대해 설명하고, 키의 종류에 대해 설명이 가능할까요?**

> 슈퍼키 (Super Key)

테이블에 존재하는 필드들의 부분집합으로써, 유일성을 만족해야 하는 키입니다. 부분 집합이므로

필드 하나도 슈퍼키가 될 수 있고, 필드 여러 개가 묶인 키도 슈퍼키가 될 수 있습니다. 필드가

여러 개 묶인 키를 '복합키'라고도 부릅니다.

> 후보키 (Candidate Key)

후보키는 기본 키가 될 수 있는 후보가 되는 키들을 의미합니다. 슈퍼키 중에서 최소성을 만족하는

키를 의미합니다.

따라서 후보키는, 유일성 + 최소성을 만족하는 키들을 의미합니다.

- 유일성 : 모든 레코드에서 해당 필드에 중복된 값이 나타나지 않는다는 속성

- 최소성 : 최소한의 필드로 레코드를 유일하게 구별할 수 있어야 한다는 속성

> 기본키 (Primary Key)

기본 키는 테이블에서 특정 레코드를 구별하기 위해 후보 키 중에서 선택된 하나의 고유한 식별자를

의미합니다. 중복되는 값이나 NULL값은 가질 수 없습니다.

후보키 중에서 값이 자주 변경되지 않고, 값이 단순한 것을 선택하는 것이 좋습니다.

> 대체키 (Alternate Key)

후보 키 중에서 기본 키로 선택된 것들을 제외한 나머지 모든 것을 대체키라고 부릅니다.

> 외래키 (Foreign Key)

한 테이블의 키 중에서 다른 테이블의 레코드를 유일하게 식별할 수 있는 키를 의미합니다.

하나의 필드 또는 전체 필드의 부분집합으로 구성되고, 중복된 값과 NULL값을 가질 수 있으며

반드시 참조되는 테이블에서 유일한 값을 갖는 필드를 참조해야 합니다.

**- 트랜잭션이 무엇인지 설명해주세요.**

- 데이터베이스의 상태를 변화시키는 하나의 논리적인 작업 단위를 의미합니다.

- 논리적인 작업의 쿼리 개수와 관계 없이 트랜잭션으로 묶어놓은 작업 단위 전체가 100% 적용되거나 아무것도 적용되지 않아야 함을 보장합니다.

**- 그렇다면 트랜잭션의 특징은 무엇이 있을까요?**

트랜잭션의 특징은 앞 글자를 따서 ACID라고 부릅니다.

- Atomicity (원자성) : 트랜잭션을 구성하는 연산 전체가 모두 정상적으로 실행되거나 모두 취소되어야 한다.

- Consistency (일관성) : 트랜잭션이 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로

유지한다.

- Isolation (고립성) : 두 개 이상의 트랜잭션이 동시에 발생할 때, 서로의 연산에 영향을 주면 안 된다.

- Durability (지속성,영구성) : 커밋된 트랜잭션의 내용은 영구히 반영된다.

**- 트랜잭션의 상태는 어떤 것들이 있을까요?**

트랜잭션의 상태는 활동, 장애, 철회, 부분완료, 완료가 있습니다.

- 활동 (Active) : 트랜잭션이 실행 중인 상태

- 장애 (Fail) : 트랜잭션이 실행에 오류가 발생해 중단한 상태

- 철회 (Aborted) : 트랜잭션이 비정상적으로 종료되어 Rollback(트랜잭션 연산 이전의 상태로 돌아감)

수행하는 상태

- 부분 완료 (Partitially Commit) : 트랜잭션이 마지막 연산까지 실행했지만, Commit 연산이 실행되기

직전의 상태

- 완료 (Committed) : 트랜잭션이 성공적으로 종료되어 commit 연산을 실행한 후의 상태

**- 트랜잭션의 연산 중, Commit과 Rollback에 대해 설명해주세요.**

- Commit : 트랜잭션이 성공해 트랜잭션 결과를 영구적으로 반영하는 연산입니다.

- Rollback : 트랜잭션의 실행을 취소했음을 알리는 연산이며, 트랜잭션이 수행한 결과를 원래의 상태(트랜잭션 연산 이전의 상태)로 원상 복귀시키는 연산을 의미합니다.

**- DB에서 동시성 제어란 무엇일까요?**

- DBMS는 다수의 사용자를 가정하기 때문에 다중 트랜잭션의 상호 간섭 작용 중, DB를 보호해야 합니다.

동시에 여러 개의 트랜잭션이 수행될 때, 트랜잭션들이 DB의 일관성을 파괴하지 않도록 트랜잭션 간의

상호작용을 제어하는 것을 의미합니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**- DB Lock에 대해 설명해주세요.**

- DB Lock은 트랜잭션 처리의 순차성을 보장하기 위한 방법으로, 데이터에 접근하기 전에 Lock을 요청해서 Lock이 허락되면 해당 데이터에 접근할 수 있도록 하는 기법입니다.

- 비관적 락 (Pessimistic Lock) : 데이터 갱신 시 충돌이 발생할 것으로 예상해 락을 거는 방식

- 낙관적 락 (Optimistic Lock) : 데이터 갱신 시 충돌이 발생하지 않을 것이라 가정해 락을 걸지 않고, 버전

관리 기능을 통해서 트랜잭션 격리성을 관리하는 방식.

비관적 락과 낙관적 락의 사용 기준은 '읽기와 수정 비율이 어디에 가까운지'입니다. 수정의 비율이 높다면 비관적 락을, 읽기의 비중이 높다면 Optimistic을 사용합니다.

Why?

- 데이터 수정 즉시 트랜잭션 충돌을 감지하고 롤백이 가능하므로 수정이 많은 경우 비관적 락을

사용하는 것이 좋습니다.

- 읽기 작업이 많다면 JPA의 버전관리 기능을 활용해 개발하는 것이 더욱 편리하기 때문에

낙관적 락을 사용하는 것이 좋습니다.

**- 무결성이 무엇인지, 무결성 제약 조건은 무엇인지 설명해주세요.**

- 데이터의 무결성은, 정확성 / 일관성 / 유효성이 유지되는 것을 의미합니다.

- 따라서 DB의 정확성, 일관성을 보장하기 위해 저장, 삭제, 수정을 제약하기 위한 조건이 무결성

제약 조건입니다.

- 개체 무결성 : 기본 키는 null, 중복 값 불가능

- 참조 무결성 : 외래 키는 null이거나 참조 테이블의 기본 키 값과 동일해야 함.

**- JOIN은 어떨 때 사용하는 것인지 설명해주세요.**

두 개 이상의 테이블이나 데이터베이스를 연결해 데이터를 검색하는 방법입니다. 적어도 하나의 컬럼을 공유하고 있어야 사용이 가능합니다.

**- Inner join과 Outer join의 차이를 설명해주세요.**

- inner join은 서로 연관된 내용만 검색하는 조인 방법입니다. A와 B의 교집합을 말합니다.

- outer join은 한 쪽에는 데이터가 있고 한 쪽에는 데이터가 없는 경우, 데이터가 있는 쪽의 내용을 전부 출력하는 방법입니다. A와 B의 합집합을 말합니다.

**- Group By의 역할에 대해 설명해주세요.**

- GROUP BY 는 GROUP BY 명령어를 통해 특정 컬럼을 기준으로 연산한 결과를 집계 키로 정의하여 그룹을

짓는 역할을 합니다.

- 집합 연산자는 COUNT, SUM, AVG, MAX, MIN 등이 있고, DISTINCT와 같이 중복 데이터를 제거하는 특징이

있습니다.

**- DELETE, TRUNCATE, DROP의 차이에 대해 설명해주세요.**

- DELETE는 데이터는 지우지만 테이블 용량은 줄어들지 않고 원하는 데이터만 골라서 지울 수 있습니다.

삭제 후 되돌릴 수 있습니다.

- TRUNCATE는 전체 데이터를 한번에 삭제하는 방식입니다. 테이블 용량이 줄어들고 인덱스 등도 삭제되지만

테이블은 삭제할 수 없고, 삭제 후 되돌릴 수 없습니다.

- DROP은 테이블 자체를 완전히 삭제하는 방식(공간, 인덱스, 객체 모두 삭제)입니다. 삭제 후 되돌릴 수

없습니다.

**- 트리거가 무엇인지 설명해주세요.**

트리거는 특정 테이블에 반응해 INSERT, DELETE, UPDATE 같은 DML이 수행되었을 때, 데이터베이스에서 자동으로 동작하도록 작성된 프로그램(메서드 형식의 쿼리)으로, 사용자가 직접 호출하지 않고 DB에서 자동적으로 호출된다는 특징이 있습니다.

DML이 실행되고, 자동으로 정의된 프로시저가 실행됩니다.

(프로시저 : 쿼리문을 마치 하나의 메서드 형식으로 만들고 어떤 동작을 일괄적으로 처리하는 용도.)

**- DML, DDL, DCL이 각각 무엇인지 설명해주세요.**

- DDL(정의어) : 데이터베이스 구조를 정의, 수정, 삭제하는 언어 (ALTER, CREATE, DROP)

- DML(조작어) : DB내의 자료 검색, 삽입, 갱신, 삭제를 위한 언어 (SELECT, INSERT, UPDATE, DELETE)

- DCL(제어어) : 데이터에 대해 무결성 유지, 병행 수행 제어, 보호와 관리를 위한 언어

(COMMIT, ROLLBACK, GRANT, REVOKE)

**- DB 인덱스에 대해 설명해주세요.**

- Index는 테이블을 처음부터 끝까지 검색하지 않고, 인덱스를 검색해서 해당 자료의 테이블을 접근하는

방법을 의미합니다.

- Index는 항상 정렬된 상태를 유지하기 때문에 원하는 값을 검색하는 데 빠른 장점을 가지고 있지만,

새로운 값을 추가하거나 삭제, 수정하는 경우에는 쿼리문 실행 속도가 느려집니다.

(인덱스도 변경하고 정렬해야 하므로 성능 저하)

- 즉, 인덱스는 데이터의 저장 성능을 희생하는 대신, 데이터의 검색 속도를 높이는 기능입니다.

**- 정규화를 왜 하는지, 각 정규화 단계는 어떤 것을 의미하는지, 정규화의 장단점에 대해 설명해주세요.**

하나의 릴레이션에 하나의 의미만 존재하도록 릴레이션을 분해하는 과정이며, 데이터의 일관성 및 최소한의 데이터 중복, 최대한의 데이터 유연성을 위해 정규화를 진행합니다.

즉, 이상현상이 존재하는 테이블(릴레이션)을 분해해 여러 개의 테이블(릴레이션)으로 분리하는 과정을 정규화라고 할 수 있습니다.

이상현상 : 테이블을 설계할 때 잘못 설계해 데이터를 삽입, 삭제, 수정할 때 생기는 논리적 오류를 의미.

> 이상현상의 종류

삽입 이상 : 불필요한 데이터를 추가해야만 삽입이 가능한 상황

갱신 이상 : 전체의 데이터 중 일부만 변경하여 데이터가 불일치하는 상황

삭제 이상 : 삭제로 인해 꼭 필요한 데이터까지 삭제되는 상황

> 정규화

1 정규화 : 테이블의 컬럼이 하나의 값을 갖도록 테이블을 분해하는 것(도메인이 원자값만 포함)

2 정규화 : 기본키의 부분집합이 결정자가 안되도록 분리(완전 함수적 종속)

3 정규화 : 이행종속 제거 (a -> b, b -> c 일때, a->c 가 성립되는 것을 분리, 즉, a-> b랑 b->c 테이블로 분리)

BCNF 정규화 : 값을 정하는 결정자가 후보키가 되도록 테이블 분해(모든 결정키가 후보키)

> 정규화의 장단점

- 데이터베이스 변경 시 발생하는 이상현상을 해결할 수 있고, 구조확장 시 정규화된 데이터베이스는

구조 변경하지 않거나 일부만 변경해도 됩니다.

- 릴레이션 분해로 인한 JOIN 연산이 많아져 질의에 대한 응답시간이 느려집니다. 다만 중복속성을 제거하고

테이블의 데이터 용량이 최소화 되는 효과가 있어 속도가 빨라질 수도, 느려질 수도 있습니다.

**- 역정규화(반정규화)는 무엇인지 설명해주세요.**

- 정규화의 단점으로, 릴레이션 간의 JOIN 연산이 늘어나 질의 성능이 저하될 우려가 있습니다.

읽기 작업이 많이 필요한 DB의 전반적인 성능을 향상시키기 위해 역정규화를 진행합니다.

**- Connection Pool이 무엇인지 설명해주세요.**

- Client의 요청에 따라 각 Application의 Thread에서 DB에 접근하기 위해서는 Connection이 필요합니다.

커넥션 풀은, 미리 일정 수의 Connection을 생성해 Pool에 보관해 두는 것을 의미합니다.

- 요청이 올 때마다 Connection을 생성하는 방식은, 연결량이 많을 때 서버에 과부하가 걸리게 되는 현상이

발생합니다.

- 미리 만들어 Pool에 저장하면 이러한 현상을 방지할 수 있습니다.

- Connection Pool을 사용하면, 생성 및 소멸에 대한 시간 소요가 없어지기 때문에 효율적이고, 한 번에

사용할 수 있는 커넥션 수가 제어되어 Application이 쉽게 죽지 않습니다.

**- RDB와 NoSQL의 차이에 대해서 설명하실 수 있나요?**

1) RDB(관계형 데이터베이스)

1. 정해진 스키마에 따라 데이터를 '테이블'에 저장하는 데이터베이스

2. 데이터 구조가 보장되어있고, 중복을 피할 수 있습니다.

3. SQL을 사용해 RDBMS에서 데이터를 저장, 수정, 삭제 및 검색할 수 있습니다.

4. 수직적 확장이 가능합니다. (서버의 성능을 향상시키는 것)

5. 관계를 맺고 있는 데이터의 변경이 잦은 경우 / 변경될 여지가 없고, 명확한 스키마가 사용자와

데이터에게 중요한 경우에 사용합니다.

2) NoSQL

1. 스키마가 없거나, 느슨한 스키마로 데이터 간의 관계 없이 자유로운 형태로 데이터를 저장하는

데이터베이스 (비정형)

2. 유연하기 때문에 데이터 조정과 새로운 필드 생성이 자유롭습니다.

3. 중복을 계속해서 업데이트 해줘야 하는 단점이 있습니다.

4. 수평적 확장으로 트래픽 분산 및 대용량 처리가 가능합니다.

5. 읽기를 자주하지만 데이터 변경은 자주 없는 경우 / 비정형의 데이터 구조를 가진 경우 /

막대한 양의 데이터에 의해 데이터베이스를 수평적으로 확장해야 하는 경우에 사용합니다.

**- Redis의 특징에 대해 설명해주세요.**

Redis는 NoSQL로 싱글쓰레드 인메모리 DB입니다. key-value 형태로 데이터를 저장하며 주로 캐시에 사용됩니다.

스탭샷을 통해 디스크에 백업할 수 있고, AOF를 통해 모든 작업을 log 파일에 기록하고 서버가 재실행 되면

순차적으로 연산해 데이터를 복구할 수 있습니다. 또한 자료구조를 지원합니다.

**- DB 클러스터링에 대해 설명해주세요.**

DB 클러스터링은 DB 스토리지는 공유하고, 동일한 DB 서버를 여러 대 놓는 방식입니다. 동기 방식으로 동기화를 진행합니다.

- Active-Active 방식

- 여러 대의 DB서버가 트래픽을 분산해서 받습니다.

- 여러 대의 서버가 DB 스토리지를 공유하기 때문에 병목이 생길 수 있습니다.

- Active-Standby 방식

- 한쪽은 Standby 상태로 두어 Active 상태의 서버가 죽으면 FailOver되어 전환하는 방식

- FailOver이 이루어지는 동안 손실이 발생합니다.

**- DB 레플리케이션에 대해 설명해주세요.**

DB 레플리케이션은 DB 서버와 DB 스토리지를 다중화하는 방식으로 DB 클러스터링과는 차이가 있습니다.

- 조회작업은 Slave에서 INSERT, DELETE, UPDATE 작업은 Master에서 수행하면서 트래픽을 분산합니다.

- Slave로 데이터를 옮길 때 비동기 방식으로 동작하기 때문에 일관성 있는 데이터를 얻지 못할 수도 있습니다.

- Master 노드가 다운되면 복구 및 대처가 까다롭습니다.

**- 수직 파티셔닝과 수평 파티셔닝(샤딩)에 대해 설명해주세요.**

서비스의 크기가 점점 커지고 DB에 저장하는 규모가 대용량화 되면서 용량의 한계와 성능의 저하를 가져오게 되었습니다.

이러한 이슈를 해결하기 위해 테이블을 파티션이라는 작은 단위로 나눠 관리하는 파티셔닝 기법이 나왔습니다.

큰 테이블이나 인덱스를 관리하기 쉬운 Partition이라는 작은 단위로 물리적으로 분할하는 것을 의미합니다.

1) 수직 파티셔닝

큰 Table이나 인덱스를 관리하기 쉬운 단위로 분리하는 방식

예를 들어, 사람이라는 테이블이 너무 커지게 되면서 사람을 북유럽인, 아시아인, 서유럽인 이렇게 테이블로 분리하는 것이라고 볼 수 있다.

장점

1. Insert 시 분리된 파티션으로 분산시켜 경합을 줄임
2. 읽기/쓰기 향상
3. 파티션 별로 백업 및 복구 가능
4. 데이터 전체 검색 시 필요한 부분만 탐색해서 성능 증가

단점

1. 인덱스와 테이블을 별도로 파티셔닝 할 수 없다. 즉, 테이블과 인덱스를 같이 파티셔닝 해야한다.
2. 테이블 간 조인 비용 증가

2) 수평 파티셔닝 (샤딩)

같은 테이블 스키마를 가진 데이터를 다수의 데이터베이스에 분산하여 저장하는 방식

데이터를 잘 분산시키기 위해 고려해야할 것이 Shard Key이고 이를 정하는 방식으로 Hash Sharding, Dynamic Sharding, Entity Group 방식이 있다.

샤딩은 복잡도가 매우 높아지므로 다른 방식을 우선적으로 고려해야 한다.

**- SQL Injection이 무엇인지 설명해주세요.**

악의적인 사용자가 보안상의 취약점을 이용해, 임의의 SQL문을 주입하고 실행되게 하여 DB가 비정상적인 동작을 하도록 조작하는 행위를 의미합니다.

- Error based SQL Injection, Union based SQL Injection, Boolean based SQL Injection, Time based SQL Injection 등 다양한 공격기법이 있습니다.

- 대응 방안으로는 입력 값에 대한 검증, Prepared Statement 구문 사용, Error Message 노출 금지, 웹 방화벽 사용 등의 방안이 있습니다.

**- ORM이 무엇인지 설명해주세요.**

- ORM은 객체와 관계형 데이터베이스 매핑의 줄임말입니다. (Object-RDBMS Mapping)

- OOP에서 사용하는 객체라는 개념을 구현한 클래스와 RDB에서 사용하는 데이터 테이블을 서로 매핑하는

것을 의미합니다.

**- 옵티마이저(Optimizer)에 대해 아는대로 설명해주세요.**

- 옵티마이저는 SQL을 가장 빠르고 효율적으로 수행할 최적의 처리 경로를 생성해주는 DBMS 내부의 핵심

엔진입니다.

- 컴퓨터의 CPU = DBMS의 옵티마이저. 개발자가 SQL을 작성하면 옵티마이저가 실행 계획을 세우고,

최적의 효율을 갖는 계획을 판별해 쿼리를 수행합니다.

- 옵티마이저가 항상 최적의 실행 경로를 보장하는 것은 아니기 때문에, 직접 최적의 실행 경로를

작성해주는 것을 '힌트'라고 부릅니다.