CHAPTER 02

데이터베이스 관리 시스템

Database Management System (DBMS)

Lecture Note

박상돈 조교수

대전대학교 컴퓨터공학과

목 차

01. DBMS 의 등장 배경

DBMS를 이해하려면, DBMS가 없던 시절을 먼저 이해해야 한다. DBMS가 등장하기 전인 대략 1960년대 이전에는 파일 시스템(File System)이라는 방식으로 데이터를 관리했다. 파일 시스템의 한계를 이해하면, DBMS가 왜 혁신적인 해결책이었는지가 자연스럽게 이해된다.

1.1 파일 시스템(File System)의 데이터 관리

파일 시스템(File System)이란, 과거에 데이터를 관리하기 위해 사용하던 소프트웨어로, 각 응용 프로그램이 독립적인 데이터 파일을 별도로 유지·관리하는 방식이다.

예시 — 온라인 쇼핑몰: 고객 관리 프로그램은 고객 파일(고객ID, 이름, 연락처)을, 주문 관리 프로그램은 주문 파일(주문ID, 고객ID, 이름)을, 배송 관리 프로그램은 배송 파일(배송ID, 고객ID, 이름, 주소)을 각각 별도로 관리한다. 이 구조에서 "고객 이름"이라는 데이터가 3개 파일에 중복 저장된다.

핵심 문제: 김철수라는 고객이 이름을 변경하여 "박철수"가 되었을 때, 고객 파일만 수정하고 주문·배송 파일을 수정하지 않으면 데이터 불일치(Inconsistency)가 발생한다. 고객 파일에는 "박철수", 주문·배송 파일에는 "김철수"로 남는다. 은행에서 계좌 잔액이 시스템마다 다르게 기록되면 금융 사고로 이어질 수 있다.

1.2 파일 시스템의 4 가지 주요 문제점

문제점 1: 데이터 중복성 (Redundancy)

같은 데이터가 여러 파일에 중복 저장되어 데이터 일관성(Consistency)과 무결성(Integrity) 유지가 어렵다. 한 파일만 수정하면 다른 파일과 불일치가 발생한다.

일관성(Consistency): 같은 데이터는 어디서 보든 같은 값이어야 한다. 고객 파일의 전화번호와 주문 파일의 전화번호가 다르면 일관성이 깨진 것이다.

무결성(Integrity): 데이터가 정확하고 유효한 상태를 유지해야 한다. 나이에 음수 값이 들어가면 안 되고, 학점에 F보다 낮은 값이 들어가면 안 된다. 파일 시스템에서는 이를 자동 검사하는 기능이 없어 프로그래머가 일일이 코드로 검증해야 했다.

문제점 2: 데이터 종속성 (Dependency)

응용 프로그램이 데이터 파일의 구조에 종속된다. 파일 구조가 변경되면 모든 관련 응용 프로그램을 함께 수정해야 하는 부담이 발생한다.

예시: 고객 파일의 구조가 "고객ID, 이름, 연락처"에서 "고객ID, 이름, 이메일, 연락처"로 변경되면, 기존 프로그램은 세 번째 필드를 연락처로 인식하는데 실제로는 이메일이 되어 오작동한다. 이 파일을 사용하는 모든 프로그램(고객 관리, 주문 관리, 배송 관리 등)의 소스 코드를 전부 수정해야 한다. 대기업의 레거시 시스템에는 이런 파일 기반 프로그램이 수천 개 있어, 하나의 파일 구조 변경이 수천 개 프로그램의 수정을 요구했다.

데이터 종속성 vs. 데이터 독립성

파일 시스템 = 데이터 종속성 (파일 구조가 바뀌면 프로그램도 수정해야 함)

DBMS = 데이터 독립성 (DB 구조가 바뀌어도 프로그램은 수정하지 않아도 됨)

이 대비는 시험 단골 출제 포인트이다!

문제점 3: 동시 공유·보안·회복 기능 부족

다수의 프로그램이 동시에 같은 파일을 공유하기 어려웠다. 한 프로그램이 파일을 잠가놓으면 다른 프로그램은 대기해야 했다. 보안은 파일 단위의 읽기/쓰기 권한 정도밖에 설정할 수 없어, "이 파일에서 급여 필드만 특정 사용자에게 숨기기" 같은 세밀한 접근 제어가 불가능했다. 장애 시 데이터 복구(Recovery) 기능도 부족하여, 시스템 장애로 파일이 손상되면 정교하게 복구할 방법이 마땅히 없었다.

문제점 4: 응용 프로그램 개발 난이도

데이터를 읽고, 쓰고, 삽입하고, 삭제하는 모든 로직을 프로그래머가 직접 코드로 작성해야 했다. "고객 파일에서 서울에 사는 고객만 찾기" 같은 간단한 요구도 파일을 한 줄씩 읽으면서 하나하나 비교하는 코드를 처음부터 작성해야 했다. DBMS 환경에서는 SELECT * FROM 고객 WHERE 주소 = '서울' 한 줄로 해결되는 일이다. 새로운 요구 사항이 생길 때마다 프로그램을 새로 개발해야 하므로 비용이 계속 증가했다.

암기법 — 중종동개

중복성 + 종속성 + 동시 공유 부족 + 개발 난이도

시험: "파일 시스템의 문제점을 설명하시오" → 이 4가지를 쓰고 각각 설명

02. DBMS 의 정의

2.1 DBMS 란 무엇인가?

> **DBMS의 정의**
>
> DBMS(Database Management System)란, 조직에 필요한 데이터를 통합 저장하고,
>
> 이에 대한 관리를 집중적으로 담당하는 소프트웨어 시스템이다.

이 정의의 핵심 키워드 3가지:

(1) 통합 저장: 파일 시스템처럼 각 프로그램이 따로 파일을 관리하는 것이 아니라, 모든 데이터를 하나의 통합된 데이터베이스에 모아서 저장한다.

(2) 관리를 집중적으로 담당: 모든 데이터 접근과 관리를 DBMS가 도맡아 한다. 응용 프로그램이 데이터를 직접 건드리지 않고, 반드시 DBMS를 통해서만 접근한다. DBMS가 중재자·문지기 역할을 한다.

(3) 소프트웨어 시스템: DBMS는 컴퓨터에 설치하는 소프트웨어이다. Oracle, MySQL, PostgreSQL, MongoDB 등이 모두 DBMS 소프트웨어이다.

파일 시스템 vs. DBMS 비교

구분	파일 시스템 (Before)	DBMS 도입 (After)
데이터 관리	각 프로그램이 별도 파일 관리	데이터를 통합 저장·관리
데이터 중복	중복 및 불일치 발생	중복 최소화, 일관성 보장

구조 변경	프로그램 수정 필요 (종속성)	프로그램 수정 불필요 (독립성)
동시 접근·보안·회복	기능 부족	DBMS 내장 기능으로 지원
데이터 처리	매번 직접 코딩	SQL로 CRUD 연산 용이

2.2 DBMS 기반 데이터 관리 구조와 데이터 독립성

DBMS 기반 구조는 4개의 계층으로 이루어진다. 맨 위에 사용자(여러 명), 그 아래에 응용 프로그램(여러 개), 그 아래에 DBMS(하나), 맨 아래에 통합 데이터베이스(하나). 모든 응용 프로그램은 반드시 DBMS를 통해서만 데이터에 접근할 수 있으며, 직접 데이터베이스에 접근하는 것은 허용되지 않는다.

데이터 독립성 (Data Independence)

데이터베이스의 구조가 변경되더라도 응용 프로그램에 영향을 주지 않는 성질.

DBMS가 응용 프로그램과 DB 사이를 중재하여 확보한다.

비유: 식당 주방의 구조가 바뀌어도(가스레인지→인덕션, 냉장고 위치 변경) 손님은 여전히 메뉴판 보고 "짜장면 하나요"라고 주문하면 된다. 주방장(=DBMS)이 주방 내부의 변화를 손님으로부터 숨겨주기 때문이다.

2.3 DBMS 의 3 대 주요 기능

DBMS의 핵심 기능은 정의(Definition), 조작(Manipulation), 제어(Control) 3가지이다.

기능	설명	사용 언어	대표 명령어
정의 기능 (Definition)	DB 구조를 생성·수정·삭제 데이터 형식, 구조, 제약 조건 명세 테이블, 뷰, 인덱스 등 스키마 정의	DDL (Data Definition Language)	CREATE TABLE ALTER TABLE DROP TABLE
조작 기능 (Manipulation)	데이터를 삽입·삭제·수정·검색 CRUD 연산을 체계적으로 수행 사용자-DB 간 인터페이스 제공	DML (Data Manipulation Language)	SELECT INSERT UPDATE DELETE
제어 기능 (Control)	데이터의 정확성·안전성 유지 동시성 제어, 보안, 무결성, 회복	DCL (Data Control Language)	GRANT REVOKE

제어 기능의 4 가지 세부 기능

세부 기능	설명
동시성 제어 (Concurrency Control)	여러 사용자가 동시에 같은 데이터에 접근할 때, 데이터가 꼬이지 않도록 관리. 수강 신청 시 잔여석 1석에 두 명이 동시 신청하면 한 명만 성공해야 함.
접근 권한 관리 및 보안 (Security)	누가 어떤 데이터에 접근할 수 있는지 세밀하게 제어. "이 사용자는 학생 테이블의 이름·학과만 볼 수 있고, 성적은 볼 수 없다" 같은 설정 가능.
무결성 제약 조건 검사 (Integrity)	데이터가 규칙에 맞는지 자동 검사. "학번은 중복 불가", "학년은 1~4", "학과는 존재하는 학과여야 함" 등의 규칙을 위반하면 입력 거부.
장애 시 데이터 회복 (Recovery)	시스템 장애 발생 시 트랜잭션 로그를 기반으로 UNDO/REDO 연산을 수행하여 정상 상태로 복구. (13주차에서 상세 학습)

핵심 대응 관계 (반드시 암기)

정의 기능 ↔ DDL (Data Definition Language) — 그릇을 만드는 단계

조작 기능 ↔ DML (Data Manipulation Language) — 그릇에 음식을 넣고 빼는 단계

제어 기능 ↔ DCL (Data Control Language) — 그릇과 음식을 안전하게 지키는 단계

DDL + DML + DCL = SQL (Structured Query Language)

시험 출제 포인트

"DBMS의 3대 기능을 설명하시오" → 정의(DDL), 조작(DML), 제어(DCL)를 쓰고 각각 설명

"DDL과 DML의 차이를 설명하시오" → DDL은 구조 정의, DML은 데이터 조작

03. DBMS 의 장·단점

3.1 DBMS 의 장점 (7 가지)

장점 1: 데이터 중복 통제

데이터를 통합 관리하여 중복을 최소화하고, 데이터 일관성(Consistency)을 유지한다. 저장 공간도 절약된다. 예를 들어 고객 이름을 고객 테이블 한 곳에만 저장하고, 주문·배송 테이블에서 필요하면 고객 테이블을 참조(조인)하는 방식을 사용한다. 수정할 때도 한 곳만 변경하면 모든 곳에 자동 반영된다.

장점 2: 데이터 독립성 확보

DBMS가 응용 프로그램과 DB 사이에서 중재자 역할을 하므로, DB 구조가 변경되어도 응용 프로그램을 수정할 필요가 없다. 파일 시스템에서는 필드 하나 추가에 관련 프로그램 전부를 수정해야 했지만, DBMS에서는 ALTER TABLE 한 줄로 해결된다.

장점 3: 데이터 동시 공유

여러 사용자·프로그램이 동시에 같은 데이터를 공유할 수 있다. DBMS는 Lock(잠금) 메커니즘을 사용하여, 데이터를 수정하는 동안에는 다른 사용자의 동시 수정을 차단하고, 읽기만 하는 경우에는 여러 사용자의 동시 접근을 허용하는 세밀한 동시성 제어를 수행한다.

장점 4: 데이터 무결성 유지

데이터 연산 시 유효성을 자동 검사하여, 무결성 제약 조건을 만족하는 데이터만 DB에 저장된다. 대표적인 무결성 제약 조건으로는 유일성 제약(학번 중복 불가), 도메인 제약(학년은 1~4),

참조 무결성 제약(수강 테이블의 학번은 학생 테이블에 존재해야 함) 등이 있다.

장점 5: 데이터 표준화

데이터 접근 방식, 형식 및 구조를 조직 전체에 걸쳐 통일할 수 있다. 파일 시스템 시절에는 부서마다 저장 형식이 달랐지만(영업부: "성, 이름" 순 / 마케팅부: "이름 성" 순), DBMS에서는 조직 전체의 데이터 표준을 정하여 부서 간 데이터 교환이 원활해진다.

장점 6: 장애 시 회복 가능

트랜잭션 로그(Log)를 기반으로 장애 발생 이전 상태로 복구하는 기능을 제공한다. 예를 들어 은행 이체 도중 시스템이 멈추면, DBMS는 로그를 확인하여 이체 작업을 취소(UNDO)하거나, 완료된 작업을 디스크에 재기록(REDO)한다. (13주차에서 상세 학습)

장점 7: 응용 프로그램 개발 비용 절감

데이터 접근 로직을 DBMS가 담당하므로 개발자는 비즈니스 로직(실제 업무 규칙)에 집중할 수 있다. 파일 시스템에서 수십 줄로 작성하던 검색 코드를 SQL 한 줄로 대체할 수 있어, 개발 및 유지보수 비용이 크게 절감된다.

> **장점 7가지 암기법**
>
> 중복 통제, 독립성, 동시 공유, 무결성, 표준화, 회복, 개발 비용 절감
>
> 시험: "DBMS의 장점을 5가지 이상 서술하시오" → 최소 5가지는 확실히 쓸 수 있도록 준비

3.2 DBMS 의 단점 (3 가지)

단점	설명	보완
높은 비용	DBMS 소프트웨어 구매·설치 비용(Oracle 등 상용 DBMS는 수천만~수억 원), 고성능 서버 하드웨어 비용, DBA(전문 관리 인력) 비용 발생.	MySQL, PostgreSQL 등 오픈소스 DBMS로 소프트웨어 비용 절감 가능. 다만 하드웨어·인력 비용은 여전.
복잡한 백업과 회복	주기적 백업 필요. 장애 유형(HW/SW/사용자 실수)에 따라 회복 절차가 다르고 복잡하며, 전문 지식 필요. 백업 없이 장애 발생 시 데이터 영구 손실 위험.	체계적인 백업 정책 수립 및 DBA의 전문적 관리.
중앙 집중 관리의 취약점	모든 데이터가 하나의 DBMS에 집중되므로, DBMS 장애 시 전체 시스템 마비(Single Point of Failure). 파일 시스템에서는 한 파일에 문제가 생겨도 다른 파일은 정상이었음.	이중화(Replication) 기술로 DBMS를 복수 운영하여 하나가 장애여도 나머지가 대체.

시험 팁 — 장단점 비교 서술법

"DBMS의 장단점을 비교하여 설명하시오" 문제에서 장점과 단점을 대비시키면 좋은 답안.

예: "DBMS의 장점은 데이터를 중앙에서 통합 관리하여 중복을 최소화하는 것이다.

그러나 이 중앙 집중 구조가 단점이 되기도 한다. DBMS에 장애가 발생하면 전체 시스템이

영향을 받는 단일 장애 지점(Single Point of Failure) 문제가 있다."

04. DBMS 의 발전 과정

DBMS는 1960년대에 처음 등장하여 현재까지 약 60년간 네 세대에 걸쳐 진화해왔다. 전체적인 발전 방향은 "복잡한 구조 → 단순한 테이블 → 객체 지원 → 비정형 데이터 대응"으로 요약된다.

세대	시기	유형	핵심 특징	대표 제품
1세대	1960s	네트워크 DBMS 계층 DBMS	네트워크/트리 구조 복잡하고 구조 변경 어려움 표준 질의 언어 없음	IDS (네트워크) IMS-IBM (계층)
2세대	1980s	관계 DBMS (RDBMS)	테이블(행·열) 기반 단순 구조 SQL 표준 언어 도입 ACID 트랜잭션, 데이터 독립성 현재까지 DBMS의 주류	Oracle, MySQL PostgreSQL MS SQL Server MariaDB
3세대	1990s	객체지향 DBMS 객체관계 DBMS	OOP 개념을 DB에 결합 OODBMS: 순수 객체 모델 (시장 실패) ORDBMS: 관계형+객체 하이브리드	O2, GemStone Oracle, PostgreSQL
4세대+	2010s~	NoSQL NewSQL	비정형 데이터·빅데이터 대응 수평 확장(Scale-Out) NewSQL: ACID+확장성 결합	MongoDB, Redis Cassandra, Neo4j Google Spanner CockroachDB

4.1 1 세대: 네트워크·계층 DBMS (1960s)

네트워크 DBMS: 데이터 간의 관계를 간선(Edge)으로 표현하는 그래프 구조. 다대다(M:N) 관계

표현이 가능하지만, 구조가 매우 복잡하고 변경이 극도로 어려웠다. 대표 제품: IDS (1960년 초반).

계층 DBMS: 트리(Tree) 구조로 부모-자식 관계를 표현. 1:N 관계만 표현 가능하고 M:N 관계는 불가. 대표 제품: IBM의 IMS (1960년 후반). IMS는 놀랍게도 아직 일부 대형 은행·항공사에서 사용 중이다.

1세대의 공통 한계: (1) 데이터 구조가 복잡하여 일반 사용자가 사용하기 어려움 (2) 구조 변경이 매우 어려워 유연성 부족 (3) SQL 같은 표준 질의 언어가 없어 호환성 부재.

4.2 2 세대: 관계 DBMS — RDBMS (1980s)

관계 데이터 모델(Relational Data Model)을 기반으로, 데이터를 테이블(Table) 형태로 구성하는 DBMS이다. 1970년 E.F. Codd가 IBM에서 관계형 모델을 제안하고, 1980년대에 상용화되면서 DB의 주류가 되었다.

RDBMS의 5가지 핵심 특징:

(1) 단순한 테이블 구조 — 행과 열로 이루어진 직관적인 표 형태. (2) SQL(Structured Query Language) 표준 언어 사용 — DDL, DML, DCL 기능을 모두 제공. 어떤 RDBMS에서든 같은 SQL로 데이터를 다룰 수 있는 호환성. (3) 데이터 독립성이 우수 — 구조 변경의 영향 최소화. (4) ACID 트랜잭션 지원 — 데이터 처리의 안정성과 정확성 보장. (5) 현재까지 가장 널리 사용되는 모델.

주요 RDBMS 제품

제품	유형	특징
Oracle	상용(유료)	세계 시장 점유율 1위. 대기업·금융·공공기관에서 주로 사용. 안정성과 성능 우수하나 비용 매우 높음.
MySQL	오픈소스(무료)	세계에서 가장 많이 사용되는 오픈소스 DBMS. Oracle사 소유. 페이스북, 트위터, 유튜브 등 대형 서비스에서 사용.
PostgreSQL	오픈소스(무료)	가장 진보적인 오픈소스 DBMS. 기능이 매우 풍부하고 최근 인기 급상승. 객체관계(ORDBMS) 기능도 지원.
MS SQL Server	상용(유료)	마이크로소프트 제품. 윈도우 환경에서 주로 사용.
MariaDB	오픈소스(무료)	MySQL에서 분기(Fork)된 DBMS. MySQL 원래 창시자가 Oracle 인수에 반발하여 별도 개발. 한국에서도 많이 사용.

4.3 3 세대: 객체지향·객체관계 DBMS (1990s)

1990년대 C++, Java 등 객체지향 프로그래밍(OOP) 언어가 유행하면서, 프로그래밍 언어의 객체 모델과 데이터베이스의 관계 모델 사이의 불일치(Impedance Mismatch) 문제를 해결하기 위해 등장했다.

객체지향 DBMS (OODBMS): 관계형 모델을 버리고 데이터를 객체(Object)로 저장·관리. 상속, 캡슐화, 다형성 등 OOP 특성을 직접 지원. 멀티미디어, CAD 등 복잡한 데이터에 적합했으나, 복잡도가 높고 표준 질의 언어가 없어 시장에서 널리 보급되지 못했다. 대표 제품: O2, ONTOS, GemStone.

객체관계 DBMS (ORDBMS): 기존 관계형 모델을 유지하면서 객체지향 기능을 추가한 하이브

리드 모델. 테이블 구조를 그대로 사용하되, 사용자 정의 타입과 확장 SQL을 지원. 전환 비용이 낮아 현실적으로 성공. 대표 제품: Oracle, PostgreSQL (현재 가장 인기 있는 DBMS 중 2개가 사실 ORDBMS).

4.4 4 세대+: NoSQL DBMS (2010s~)

등장 배경

2000년대 중후반 SNS의 폭발적 성장(페이스북, 트위터, 유튜브 등)으로 사진, 동영상, 검색 로그 등 대량의 비정형 데이터가 생산되었다. 기존 RDBMS는 고정 스키마의 정형 데이터에 최적화되어 있어, 대규모 비정형 데이터 처리에 성능·확장성 한계가 있었다.

NoSQL (Not Only SQL)

비정형 데이터 처리에 특화된 DBMS.

고정된 스키마 없이 유연한 데이터 모델을 제공한다.

"SQL만이 전부가 아니다, SQL을 넘어서자"라는 의미.

NoSQL 의 4 가지 유형

유형	구조	특징	대표 제품
Key-Value	키 하나에 값 하나 대응 (사전처럼)	가장 단순. 속도 매우 빠름. 캐시 용도로 널리 사용.	Redis, DynamoDB
Document	JSON 형태의 문서로 저장	유연한 스키마. 각 문서가 서로 다른 구조 가능. NoSQL 중 가장 인기.	MongoDB, CouchDB

Column-Family	열(Column) 기반 저장	대량 데이터의 빠른 읽기/쓰기. 넷플릭스, 스포티파이에서 사용.	Cassandra, HBase
Graph	노드와 간선의 그래프 구조	관계 기반 분석에 강력. 소셜 네트워크 친구 추천 등.	Neo4j, ArangoDB

NoSQL의 공통 특징: 수평 확장(Scale-Out) 용이, 유연한 스키마, 대용량 데이터 처리에 최적화.

한계: ACID 트랜잭션 완전 지원이 어려워 금융 거래 같은 미션 크리티컬 업무에는 부적합. 기존 RDBMS와의 통합이 까다로움.

NoSQL 핵심 한 줄 요약

NoSQL은 "RDBMS가 잘 못하는 것"을 잘하기 위해 등장했지만,

그 대가로 "RDBMS가 잘하는 것"의 일부를 포기한 것이다.

NewSQL DBMS (2011~)

RDBMS의 ACID 트랜잭션 보장 능력과 NoSQL의 수평 확장성·유연성을 결합한 차세대 DBMS이다. 테이블 기반 + 표준 SQL 사용 + ACID 완전 지원 + 수평 확장(Scale-Out)이라는 "두 마리 토끼"를 동시에 잡는 것이 목표이다.

구분	RDBMS	NoSQL	NewSQL
데이터 모델	테이블 (관계형)	다양 (KV, Doc 등)	테이블 (관계형)
스키마	고정 스키마	유연한 스키마	고정 스키마
ACID 트랜잭션	완전 지원	부분 지원	완전 지원

확장성	수직 확장 (Scale-Up)	수평 확장 (Scale-Out)	수평 확장 (Scale-Out)
SQL 지원	표준 SQL	자체 API	표준 SQL
대표 제품	Oracle, MySQL	MongoDB, Redis	Spanner, CockroachDB

대표적인 NewSQL 제품으로는 Google Spanner(구글의 전 세계 분산 DB), VoltDB, CockroachDB, TiDB 등이 있으며, 클라우드 네이티브 환경에서 주목받고 있다.

면접 단골 질문

"RDBMS와 NoSQL의 차이를 설명하시오"

→ RDBMS: 고정 스키마, ACID 보장, SQL 사용, 수직 확장

→ NoSQL: 유연한 스키마, ACID 부분 지원, 자체 API, 수평 확장

→ NewSQL: 양쪽의 장점 결합 (ACID + 수평 확장)

4.5 현재 DBMS 인기도 (DB-Engines Ranking)

분류	1위	2위
유료 DBMS	Oracle	MS SQL Server
오픈소스 DBMS	MySQL	PostgreSQL
NoSQL DBMS	MongoDB	Redis

취업 준비 시 참고: RDBMS에서는 MySQL 또는 PostgreSQL, NoSQL에서는 MongoDB, 캐시 용도로 Redis를 알고 있으면 대부분의 웹 개발 포지션에서 통한다. (참고 사이트: db-

engines.com/en/ranking)

2 장 최종 요약

섹션 1: DBMS 의 등장 배경

파일 시스템의 4가지 문제점: 데이터 중복성, 데이터 종속성, 동시 공유·보안·회복 부족, 응용 프로그램 개발 난이도 (암기법: **중종동개**)

섹션 2: DBMS 의 정의

DBMS = 조직의 데이터를 통합 저장·관리하는 소프트웨어. 핵심 개념: 데이터 독립성(DB 구조 변경 시 프로그램 수정 불필요).

3대 기능: 정의(DDL) + 조작(DML) + 제어(DCL) = SQL

섹션 3: DBMS 의 장·단점

장점 7가지: 중복 통제, 독립성, 동시 공유, 무결성, 표준화, 회복, 개발 비용 절감

단점 3가지: 높은 비용, 복잡한 백업/회복, 중앙 집중 취약점(Single Point of Failure)

섹션 4: DBMS 의 발전 과정

1세대(네트워크·계층, 1960s) → 2세대(관계형/RDBMS, 1980s) → 3세대(객체지향·객체관계, 1990s) → 4세대+(NoSQL·NewSQL, 2010s~)

핵심 비교: RDBMS(ACID+SQL, 수직확장) vs. NoSQL(유연+수평확장, ACID 부분지원) vs. NewSQL(ACID+수평확장)

시험 대비 체크리스트

파일 시스템의 4가지 문제점을 쓸 수 있는가?

DBMS의 정의를 한 문장으로 쓸 수 있는가?

데이터 독립성이 무엇인지 설명할 수 있는가?

DBMS의 3대 기능과 각각에 대응하는 언어(DDL, DML, DCL)를 쓸 수 있는가?

DBMS의 장점 7가지와 단점 3가지를 나열할 수 있는가?

DBMS의 세대별 발전 과정을 설명할 수 있는가?

RDBMS와 NoSQL의 차이를 설명할 수 있는가?

다음 시간 예고 — 2장 연습문제 풀이

파일 시스템의 문제점, DBMS의 3대 기능, RDBMS와 NoSQL의 비교가 중점적으로 출제

오늘 수업 내용을 한번 쭉 복습하고 오시면 좋겠습니다.